
# โปรแกรมสื่อสารข้อมูลระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง

นายปิยะนิส ปิยะรัชชนันทน์ 34501011
นางสาวรักชนก จันทรธาดาพร 34501014
นายสมภพ ตาลสอน 34501021



ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต
สาขาคณิตศาสตร์ประยุกต์
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2537

# Data Communication between 2 Computer

Mr. Piyanit Piyaratchanan

Ms. Rakchanok Jantharathadaporn

Mr. Sompop Talsorn

A Special Project Submitted in Partial Fulfillment ot the

the Degree of Bachelor of Science

Department of Applied Mathematics and Computer Science

King Mongkut's Institute of Technology Ladkrabang

1994

| | |
|---|---|
| หัวข้อโครงงานพิเศษ | โปรแกรมการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง |
| โดย | นายปิยะนิส ปิยะรัชชนันทน์ |
| | นางสาวรักชนก จันทรธาดาพร |
| | นายสมภพ ตาลสอน |
| ภาควิชา | คณิตศาสตร์และวิทยาการคอมพิวเตอร์ |
| อาจารย์ที่ปรึกษา | อ.สิริลักษณ์ เตียพิริยะกิจ |

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยี-พระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับโครงงานพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

.....

( รศ. ภัคคินี ชิตสกุล )

รักษาการแทนหัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

.....

( รศ. วิเชียร ศรีเสือขาม )

ประธานกรรมการการสอบปัญหาพิเศษ

.....

( ผศ. พัชรินทร์ เหมโชติ )

กรรมการสอบปัญหาพิเศษ

.....

( อ. สิริลักษณ์ เตียพิริยะกิจ )

กรรมการและอาจารย์ที่ปรึกษาปัญหาพิเศษ

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

# บทคัดย่อ

ในปัจจุบันนี้ การทำงานของผู้ใช้ไม่ได้จำกัดเฉพาะ การใช้งานข้อมูลบนเครื่องคอมพิวเตอร์ของตนเองเท่านั้น เพราะในบางครั้งผู้ใช้อาจจำเป็น ต้องใช้ข้อมูลจากผู้ใช้คนอื่น ๆ เพื่อนำมาประมวลผลร่วมกับข้อมูลของตนเอง ซึ่งในหน่วยงานที่มีขนาดเล็ก อาจไม่จำเป็นต้องมีเครือข่ายคอมพิวเตอร์ เช่น ระบบ LAN ( Local Area Network ) เนื่องจากเป็นการเปลืองงบประมาณ ดังนั้นโปรแกรมนี้อาจเป็นทางเลือกหนึ่งสำหรับผู้ใช้ ในการแลกเปลี่ยนข้อมูลอย่างง่าย ๆ อีกทั้ง ถ้าแฟ้มข้อมูลนั้นมีขนาดใหญ่ จะมีการบีบอัดข้อมูล เพื่อให้สามารถส่งข้อมูลได้รวดเร็วขึ้นโดย โปรแกรมบีบอัดข้อมูลที่ได้จัดเตรียมไว้

วัตถุประสงค์ของปัญหาพิเศษนี้ เพื่อศึกษา หลักการสื่อสารข้อมูลเบื้องต้น , หลักการเขียนโปรแกรมฝังตัวในหน่วยความจำ , หลักการบีบอัดข้อมูลโดยวิธีของ Huffman และ ศึกษาแนวความคิดในการเขียนโปรแกรม editor และนำหลักการดังกล่าวมาพัฒนา โปรแกรมอำนวยความสะดวก ( Utility Program ) โดยที่โปรแกรมทั้งหมดอยู่ในสภาวะฝังตัวในหน่วยความจำ ( Terminate and Stay resident )

## Abstract

Today computer user not only use his data in data processing but also use another user's data together with his data as well. Any small organization may not have to use network system such as LAN ( Location Area Network ) for reason of exession . So Using this program is one way for the user to transfer data between computer . Data compression is one utility of the program to help the data transfer faster .

Object to study is the basic principle of data communication , resident programming with C , data compression by Huffman , make editor program and use the previous knowledge in creating a utility resident program .

# กิตติกรรมประกาศ

ในการจัดทำปัญหาพิเศษเรื่องการติดต่อสื่อสารระหว่างคอมพิวเตอร์สองเครื่อง สามารถสำเร็จลุล่วงไปด้วยดี คณะผู้จัดทำต้องขอขอบพระคุณอาจารย์ สิริลักษณ์ เตียพิริยะกิจ อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษา ในการแก้ปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ จนการทำปัญหาพิเศษครั้งนี้สำเร็จด้วยดี รวมทั้งเพื่อนๆ และน้องๆ ทุกคนที่ให้ความช่วยเหลือในด้านต่างๆ เกี่ยวกับการทำปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2538

# สารบัญ
# CONTENTS

# บทที่ 1
# บทนำ

ปัจจุบันเทคโนโลยีทางด้านการสื่อสารข้อมูล ระหว่างเครื่องไมโครคอมพิวเตอร์ได้รับการพัฒนาอย่างรวดเร็ว และมีการใช้งานอย่างกว้างขวาง เนื่องจากประโยชน์ที่ได้รับจากเทคโนโลยีนี้มีมาก จึงทำให้ทุกฝ่ายได้ให้ความสนใจและความสำคัญในการที่จะพัฒนาการสื่อสารข้อมูล ซึ่งในปัจจุบันไม่ว่าจะเป็นสำนักงาน มหาวิทยาลัย หรือองค์กรต่างๆ เป็นต้น ต่างก็ได้รับความสะดวกสบายจากการใช้เทคโนโลยีนี้ เช่น การใช้ระบบ LAN ( Local Area Network ) ภายในสำนักงาน

การจะนำเอาเทคโนโลยีนี้มาประยุกต์ใช้ รวมถึงการบำรุงรักษาหรือปรับปรุงระบบให้ใช้เหมาะสมกับงานนั้น ก็ต้องอาศัยความรู้ความเข้าใจถึงหลักการ การทำงานของเทคโนโลยีการสื่อสารข้อมูลระหว่างเครื่องคอมพิวเตอร์ 2 เครื่องเป็นพื้นฐาน

## 1.1 หลักการทำงานของโปรแกรมการสื่อสารข้อมูลระหว่างเครื่องไมโครคอมพิวเตอร์ 2 เครื่อง

หลักการทำงานของการสื่อสารข้อมูลระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง สามารถแบ่งได้เป็น 3 ส่วน กล่าวโดยสรุปคือ

- ส่วนการเตรียมข้อมูล
- ส่วนการส่งข้อมูล
- ส่วนรับข้อมูล

ในส่วนการเตรียมข้อมูลในโครงงานนี้ จะใช้โปรแกรม Editor ที่เขียนขึ้นเป็นโปรแกรมจัดการ และข้อมูลที่ต้องการส่ง ( Transfer ) จะผ่านการบีบอัดข้อมูล ( Data Compression ) เพื่อให้การส่งข้อมูลรวดเร็ว และเป็นการประหยัดเนื้อที่ฮาร์ดิสก์ ( Hard Disk ) จากนั้นจึงส่งไปตามสาย RS 232C ซึ่งเป็นพอร์ตแบบอนุกรม ( Serial Port ) และทำการส่งแบบ Asynchronous ส่วนทางด้านรับข้อมูลก็จะทำการขยายข้อมูล ( Decompression ) ให้ได้ข้อมูลเดิมกลับมา ซึ่งโปรแกรมนี้จะถูกฝังตัวในหน่วยความจำ ( Resident Program ) เพื่อที่ว่าโปรแกรมอื่นไม่สามารถโหลด ( Load ) ทับได้ อีกทั้งยังทำให้การเรียกใช้โปรแกรมการสื่อสารนี้เป็นไปได้อย่างสะดวกและรวดเร็วอีกด้วย

1.2 วัตถุประสงค์ของการวิจัย

- เรียนรู้และเข้าใจ ถึงหลักการทำงานของ การสื่อสารข้อมูลระหว่างเครื่องคอมพิวเตอร์ เพื่อเป็นพื้นฐานไปสู่ความเข้าใจการสื่อสารที่เป็นระบบเครือข่าย
- เรียนรู้และเข้าใจถึงหลักการบีบอัดข้อมูล ( Data Compression )
- สามารถนำโปรแกรมสื่อสารนี้ไปใช้งานได้

1.3 ขั้นตอนการวิจัย

- ศึกษารายละเอียดการสื่อสารข้อมูลระหว่างเครื่องไมโครคอมพิวเตอร์ ทั้งทางด้านฮาร์ดแวร์ ( Hardware ) และ ซอฟท์แวร์ ( Software )
- ศึกษารายละเอียดการบีบอัดข้อมูล ( Data Compression ) และ อัลกอริทึมของฮัฟฟ์แมน ( Huffman Algorithm )
- ศึกษาการทำงานของสายส่ง RS - 232C
- สร้างโปรแกรมสื่อสารข้อมูล ระหว่างเครื่องไมโครคอมพิวเตอร์ ผ่านทางพอร์ตอนุกรม ( Serial Port ) ของสายส่ง RS - 232C
- ดำ เนินการติดตั้งเชื่อมต่อสายส่งข้อมูล RS - 232C และทดสอบการใช้งาน
- สร้างโปรแกรม Editor เพื่อใช้ติดต่อกับผู้ใช้ ( User Interface ) ให้มีลักษณะเป็นโปรแกรมฝังตัวในหน่วยความจำ ( Resident Program )
- สร้างโปรแกรมบีบอัดข้อมูล ( Data Compression )
- นำโปรแกรมมารวมกัน ( Link ) เพื่อทำการทดสอบการทำงานของโปรแกรมโดยรวมทั้งหมด

1.4 ขอบเขตการวิจัย

1.4.1 ในการทำโครงงานพิเศษนี้จะพัฒนาบนเครื่องไมโครคอมพิวเตอร์

1.4.2 โปรแกรมการสื่อสารข้อมูลส่งผ่านข้อมูลทาง สายส่ง RS - 232C ระหว่างเครื่องไมโครคอมพิวเตอร์ 2 เครื่อง

1.4.3 สามารถใช้โปรแกรมการสื่อสารข้อมูลนี้ ได้กับเครื่อง IBM PC , PC / XT , PC / AT หรือ เครื่องที่เข้ากันได้กับเครื่อง IBM ( IBM Compatible ) ทั่วไป ที่มีหน่วยความจำอย่างต่ำ 640 กิโลไบต์

1.4.4 สามารถใช้งานได้กับวงจรแสดงผลต่อไปนี้ HERCULES, CGA , EGA , VGA

1.4.5 ทำงานภายใต้ระบบปฏิบัติการ DOS ตั้งแต่รุ่น ( Version ) 2.0 ขึ้นไป

1.5 <u>เครื่องมือที่ใช้ในการทำวิจัย</u>

1.5.1 ฮาร์ดแวร์ ( Hardware )

- เครื่องคอมพิวเตอร์ 2 เครื่อง
- ฮาร์ดดิสก์ ( Hard Disk ) ที่มีเนื้อที่ในฮาร์ดดิสก์ ไม่น้อยกว่า 150,000 ไบต์ ( Byte )
- สายส่ง RS - 232C

1.5.2 ซอฟท์แวร์ ( Software )

- ระบบปฏิบัติการ DOS ตั้งแต่รุ่น 2.0 ขึ้นไป
- ตัวแปลโปรแกรม ( Compiler ) TURBO C

1.6 <u>ประโยชน์ที่คาดว่าจะได้รับ</u>

- เรียนรู้และเข้าใจถึงการสื่อสารข้อมูลระหว่างเครื่องไมโครคอมพิวเตอร์
- สามารถสื่อสารข้อมูลระหว่างคอมพิวเตอร์ โดยผ่านทางโปรแกรมประยุกต์ ( Application Program ) ที่เขียนขึ้นมาได้
- สามารถนำโปรแกรมบีบอัดข้อมูล ( Data Compression ) มาประยุกต์ใช้งานได้
- สามารถนำโปรแกรมสื่อสารข้อมูลนี้ไปใช้งานได้

# บทที่ 2
# ทฤษฎีและหลักเกณฑ์ที่เกี่ยวข้อง

## SCREEN - EDITOR

ถึงแม้ว่าในปัจจุบันจะพบว่ามี Text Editor อยู่เป็นจำนวนมากบน Operating System ต่างๆ แต่โครงงานนี้ก็ได้จัดทำ Text Editor ขึ้นมาด้วยเหตุผล คือ

1. โครงงานที่มีลักษณะเป็นการ Programming โดยทั่วไป ต้องการ Text Editor ไว้ใช้งาน อาทิเช่น Database Manager นั้นต้องการ Editor เพื่อให้ผู้ใช้ ( user ) ใช้สร้างการสอบถาม ( queries ) โปรแกรมการสื่อสารนี้ ต้องการ Editor เพื่อไว้ใช้งานด้านการสร้างรายการส่ง ( script ) ซึ่งรายการส่งนี้จะถูก Execute ต่อมาโดยอัตโนมัติ

2. Text Editor ต่างๆ ที่ถูกสร้างขึ้นมานั้น อาจใช้งานได้ แต่อาจจะไม่เป็นที่พอใจของผู้ใช้ ดังนั้นจึงควรสร้าง Editor ขึ้นมาเพื่อรองรับความต้องการของผู้ใช้ เช่น การกำหนดฟังก์ชันคีย์ต่าง ๆ ให้มีหน้าที่ตามความต้องการของผู้ใช้ เป็นต้น

### หน้าที่ของโปรแกรม Editor

คือ ให้ผู้ใช้สามารถป้อนข้อมูลในแบบเต็มหน้าจอ ( Full Screen Edit ) ได้โดยให้ผู้ใช้สามารถเลื่อนตัวชี้จอภาพ ( Cursor ) ไปได้ทั่วทั้งจอภาพ เพื่อการแทรก-ลบ ข้อมูลในแต่ละบรรทัด หรือแทรก-ลบ-ย้ายข้อมูล ทั้งบรรทัดหรือหลายๆ บรรทัด ซึ่งการดำเนินการทั้งหมดนี้ มีพื้นฐานอยู่ที่การจัดเก็บข้อมูลกับ การแทรก-ลบ ข้อมูลทั้งบรรทัด ซึ่งการสร้างโปรแกรม Editor อาศัยการจัดเก็บข้อมูล เป็นแบบ อะเรย์ของพอยเตอร์ ไปยัง ชุดตัวอักษร ( String ) ในฮีพ ( Heap ) ซึ่งเรียกสั้นๆ ว่า " พอยเตอร์ไปยังสตริง "

### พอยเตอร์ไปยัง String

การจัดเก็บข้อมูลแบบนี้จะกำหนดเป็น อะเรย์ของพอยเตอร์ไปยังสตริง จำนวนช่องของอะเรย์คือจำนวนบรรทัดตามแต่ผู้เขียนโปรแกรมจะกำหนด

สำหรับความยาวของสตริง คือ จำนวนอักษรในแต่ละบรรทัด ขึ้นอยู่กับงานสำหรับทางด้านเอกสาร ( Document ) ส่วนมากจะอยู่ในช่วง 65 - 80 ตัวอักษร การกำหนดตัวแปร row เป็นอะเรย์ของพอยเตอร์ไปยังสตริงในฮีฟอีกทอดหนึ่งในแบบ 1 พอยเตอร์ ต่อ 1 สตริง ซึ่งในการเขียนโปรแกรม สามารถแยกออกได้เป็น 2 แนวความคิด คือ

- แบบที่หนึ่ง ให้กำหนดหน่วยความจำในฮีฟ 2000 สตริงในตอนต้น

- แบบที่สอง ให้กำหนดสตริงในฮีฟ เมื่อต้องการใช้ และยกเลิกเมื่อไม่ต้องการใช้ ซึ่งเหมาะกับโปรแกรมที่ทำหน้าที่ Editor อย่างเดียว

## Editor โดยทั่วไปควรมีคุณสมบัติ

1. พิมพ์ตัวอักษร
2. ลบตัวอักษร
3. ลบบรรทัดทั้งบรรทัด
4. เลื่อนลงจากหน้าจอ 1 บรรทัด ( Scroll Down )
5. เลื่อนขึ้นจากหน้าจอ 1 บรรทัด ( Scroll Up )
6. เลื่อนขึ้น 1 หน้า ( Page Up )
7. เลื่อนลง 1 หน้า ( Page Down )
8. กลับสู่ต้นแฟ้มข้อมูล
9. ไปสู่ท้ายแฟ้มข้อมูล
10. สามารถทำการค้นหาชุดตัวอักษร ( String ) ที่ต้องการได้
11. สามารถทำการค้นหาและแทนที่ชุดตัวอักษร ( String ) ที่ต้องการได้
12. สามารถพิมพ์แฟ้มข้อมูลจาก Editor ไปสู่เครื่องพิมพ์ได้

## แนวทางการสร้าง Editor ในโครงงานนี้

สิ่งสำคัญที่สุดในการเขียน Editor นี้คือ การรักษาการเชื่อมโยงที่ตรงกันระหว่าง แฟ้มข้อมูล ( File ) และจอภาพ ( Screen ) เมื่อทำการสร้าง Editor นี้มีความจำเป็นที่จะต้องรักษาการเชื่อมโยง ( Link ) ของสองสิ่งเข้าด้วยกัน ซึ่งเราจะใช้สิ่งที่เรียกว่า ตำแหน่งปัจจุบันของจอภาพ ( Current locaion ) ซึ่งก็คือ คู่ลำดับตำแหน่งต่างๆ บนจอภาพ และตำแหน่งปัจจุบันในแฟ้มข้อมูล ( pointer ที่ชี้อยู่ในแฟ้มข้อมูล ) ซึ่งจะต้องทำการรักษาการเชื่อมโยงของสองสิ่งนี้ด้วยกันเสมอ กล่าวคือ เช่น การเลื่อนตำแหน่งตัวชี้บนจอภาพ ( Cursor 1 ตำแหน่ง จะทำให้ตำแหน่งในแฟ้มข้อมูล ( File Pointer ) ถูกเลื่อนไป 1 ตำแหน่งเช่นกัน เป็นต้น ดังนั้นอาจจะมีปัญหาที่ว่า แฟ้มข้อมูล ( File ) นั้นมีลักษณะเป็น 1 มิติ แต่จอภาพนั้นมีลักษณะเป็น 2 มิติ และจะพบปัญหาอีกอย่างคือ แฟ้มข้อมูลทั้งแฟ้มจะถูกเก็บอยู่ในหน่วยความจำตลอดเวลา แต่จอภาพนั้นสามารถแสดงผลแฟ้มข้อมูลได้เพียงบางส่วนเท่านั้น

**การแทรกและการลบตัวอักษร ( Inserting And Deleting Character )**

จากที่ทราบแล้วว่ามี แนวความคิดมากมายที่สามารถใช้สร้าง Editor ได้ แต่ Editor เหล่านี้จะต้องมีลักษณะพื้นฐาน 2 สิ่งที่เหมือนกันคือ การแทรกตัวอักษร และ การลบตัวอักษร และ Editor ที่ได้จัดทำขึ้นนี้ ตัวอักษรทั้งหมดจะถูกจัดเก็บในหน่วยความจำ ดังนั้นแต่ละครั้งที่ผู้ใช้พิมพ์ตัวอักษร ตำแหน่งของตัวอักษรที่อยู่ในปัจจุบัน และถัดจากนี้ไปจะถูกเลื่อนไป 1 ตำแหน่ง และตัวอักษรที่ถูกพิมพ์จะถูกแทรกเข้าไป และเมื่อตัวอักษรถูกลบ ตัวอักษรตัวถัดไปจากตำแหน่งที่ถูกลบทั้งหมดจะถูกเลื่อนขึ้นมา 1 ตำแหน่ง แต่ละครั้งที่มีการแทรกหรือลบข้อมูล จอภาพต้องทำการเปลี่ยนแปลงผลที่เกิดขึ้นจากการกระทำเหล่านี้

**เงื่อนไขขอบเขตของ Editor ( Boundary Condition )**

ตัวอักษรที่ถูกพิมพ์ทั้งหมดจะถูกเก็บอยู่ในหน่วยความจำ และหน่วยความจำนี้จะมีตำแหน่งเริ่มต้นและตำแหน่งสุดท้าย ดังนั้นจะเห็นได้ว่า โปรแกรมที่สร้าง Editor ขึ้นมาต้องมีการป้องกัน ตำแหน่งปัจจุบัน ( Current Location ) ที่กล่าวไปแล้วนั้น จากการเลื่อนกลับหรือการคอยหลังเกินจุดเริ่มต้นของหน่วยความจำที่กำหนดไว้ หรืออาจเป็นการเลื่อนตำแหน่งปัจจุบันเลย ตำแหน่งสุดท้ายของหน่วยความจำไป ดังนั้นผู้เขียนโปรแกรม Editor จะต้องใส่ใจถึงขอบเขตเหล่านี้

**หน้าที่คีย์ต่างๆ ที่กำหนดใน Editor**

| **คีย์** | **หน้าที่** |
|---|---|
| ลูกศรซ้าย ( Left Arrow ) | เลื่อนตำแหน่งปัจจุบัน ( Current Location ) ไปทางซ้าย 1 ตัวอักษร |
| ลูกศรขวา ( Right Arrow ) | เลื่อนตำแหน่งปัจจุบัน ( Current Locaion ) ไปทางขวา 1 ตัวอักษร |
| ลูกศรขึ้น ( Up Arrow ) | เลื่อนตำแหน่งปัจจุบัน ( Current Location ) ขึ้น 1 บรรทัด |
| ลูกศรลง ( Down Arrow ) | เลื่อนตำแหน่งปัจจุบัน ( Current Location ) ลง 1 บรรทัด |
| โฮม ( Home ) | เลื่อนตำแหน่งปัจจุบัน ( Current Location ) ไปที่ต้นบรรทัด |
| เอ็น ( End ) | เลื่อนตำแหน่งปัจจุบัน ( Current Location ) ไปที่ปลายสุดของบรรทัด |
| เดล ( Del ) | ลบตัวอักษร 1 ตัว ณ ตำแหน่งปัจจุบันของตัวชี้จอภาพ ( Cursor ) |
| แบคสเปซ ( Backspace ) | ลบตัวอักษร 1 ตัว ณ ตำแหน่งทางซ้ายของตัวชี้จอภาพ ( Cursor ) |
| F3 | อ่านแฟ้มข้อมูลจากภายนอกไปสู่หน่วยความจำ |
| F2 | จัดเก็บแฟ้มข้อมูล |

| คีย์ | หน้าที่ |
|---|---|
| F1 | ส่วนช่วยเหลือคีย์ต่างๆ ของ Editor |
| F10 | ส่วนเมนูบาร์ |
| CTRL - Y | ลบบรรทัด ณ ตำแหน่งตัวชี้จอภาพ ( Cursor ) ทั้งบรรทัด |
| CTRL - QF | ค้นหาชุดตัวอักษร ( String ) ที่ต้องการ 1 ครั้ง |
| CTRL - QA | ค้นหาและแทนที่ชุดตัวอักษร ( String ) ที่ต้องการ |
| CTRL - L | ทำการค้นหาต่อไป |

และคีย์ F2 , F3 , CTRL - QF , CTRL - QA , CTRL - L สามารถเลือกทำงานได้จากเมนูบาร์ โดยการกด F10 และเมนูบาร์จะขึ้นมา จากนั้นทำการเลือกการทำงานตามที่ต้องการได้

## Terminate and stay resident

จากที่ทราบมาแล้วว่า ดอส ( Dos ) เป็นระบบปฏิบัติการ ( Operating System ) ที่ถูกใช้งานอย่างแพร่หลายทั่วโลก มีผู้ประมาณว่ามีระบบที่ติดตั้งระบบปฏิบัติการดอสถึง 100 ล้านเครื่อง เนื่องมาจากการใช้งานง่าย ดอสจึงประสพความสำเร็จทางด้านนี้ ดอสเป็นระบบปฏิบัติการแบบ ผู้ใช้คนเดียว ( Single user ) , ทำงานทีละงาน ( Single Tasking ) ซึ่งจะเห็นว่าเป็นลักษณะระบบปฏิบัติการของคอมพิวเตอร์ส่วนบุคคล ( Personal Computer ) อย่างไรก็ตามได้มีการศึกษาและนำประโยชน์ของลักษณะที่ดอสไม่ได้แจ้งไว้ หรือลักษณะ Undocumented ของ Dos ซึ่งบริษัทไมโครซอฟท์เองไม่ประสพความสำเร็จในการพยายามที่จะซ่อนความลับนี้ไว้ ดังนั้นเราจึงสามารถนำดอสเข้าไปทำงานในหลายๆ งานได้ในระบบปฏิบัติการ ( Multi - Tasking )

โปรแกรมฝังตัวในหน่วยความจำ จะสามารถทำงานได้ โดยการใช้อินเตอร์รัพต์ สำหรับโปรแกรมฝังตัวที่มีลักษณะโต้ตอบ ( Interactive ) จะเห็นว่าต้องใช้อินเตอร์หมายเลข 9 ( keystoke interrupt ) เป็นต้น

**รายละเอียดเกี่ยวกับอินเตอร์รัพต์**

ไมโครโปรเซสเซอร์ 8086 ได้ตระเตรียมอินเตอร์รัพต์เวกเตอร์ จำนวน 256 ตัว ไว้ให้ใช้งาน ซึ่งบางอินเตอร์รัพต์เวกเตอร์นั้นก็ไม่ได้ถูกใช้งานแต่อย่างใด เก็บค่าแอดเดรส ( address ) 0 ไว้ ค่าอินเตอร์รัพต์เวกเตอร์เป็นค่าที่ถูกเก็บอยู่ในตารางอินเตอร์รัพต์เวกเตอร์ ซึ่งตารางนี้มีขนาด 1 กิโลไบต์ ( 1024 ไบต์ ) แต่ละอินเตอร์รัพต์เวกเตอร์จะกินเนื้อที่ 4 ไบต์ ดังนั้นจึงมีอินเตอร์รัพต์เวกเตอร์ได้ทั้งหมด 256 ตัว ซึ่งค่าที่ถูกเก็บอยู่ในอินเตอร์รัพต์เวกเตอร์ เป็นค่าเซกเมนต์ ( Segment ) และออฟเซต ( Offset ) จะถูกกำหนดเป็น 2 ไบต์ ในรูทีนการบริการอินเตอร์รัพต์ ( interrupt service routine ) อาทิเช่น อินเตอร์รัพต์หมายเลข 0 จะอยู่ ณ ตำแหน่งหน่วยความจำที่ 0000 : 0000 และตำแหน่งสิ้นสุดที่ 0000 : 0003 ซึ่งไบต์แรก จะเก็บค่าเซกเมนต์รูทีนการบริการอินเตอร์รัพต์ และ 2 ไบต์หลังเก็บค่าออฟเซตแอดเดรสของรูทีนการบริการอินเตอร์รัพต์ และอินเตอร์รัพต์เวกเตอร์ที่ 1 จะอยู่ ณ ตำแหน่งหน่วยความจำที่ 0000 : 0004 - 0000 : 0007 ซึ่งค่าที่ถูกเก็บไว้มีความหมายดังที่กล่าวไปแล้ว

โปรแกรมฝังตัวในหน่วยความจำนี้จะยึดเอาอินเตอร์รัพต์ 1 ตัว หรือมากกว่า ไว้ทำงาน ( ให้อินเตอร์รัพต์เวกเตอร์เก็บค่า แอดเดรสของโปรแกรม )

**การออกแบบพื้นฐานของโปรแกรมฝังตัวในหน่วยความจำ**

โปรแกรมฝังตัวในหน่วยความจำนี้ สามารถแบ่งเป็นส่วนใหญ่ ๆ ได้ 3 ส่วน

1. ส่วนการติดตั้ง ( Installation )
2. ส่วนรูทีนการบริการอินเตอร์รัพต์ ( Interrupt Service Routine )
3. ส่วนแอพลิเคชัน ( งานที่ต้องการให้โปรแกรมทำ )

**ส่วนการติดตั้งโปรแกรมฝังตัวไว้ในหน่วยความจำ**

เมื่อโปรแกรมฝังตัวเริ่มการทำงาน ส่วนการติดตั้งนี้ จะทำหน้าที่ 3 ประการ คือ

1. ทำการเริ่มต้น ( Initialize ) ทุกสิ่งทุกอย่างที่ส่วนแอพลิเคชันนั้นต้องการ เช่น การโหลดไฟล์ที่ถูกใช้ในส่วนแอพลิเคชัน ( กล่าวคือไม่ต้องการโหลดไฟล์ใหม่ในขณะทำงานอยู่ในส่วนแอพลิเคชัน )

2. ทำการกำหนดค่าแอดเดรสของโปรแกรมฝังตัวลงใน อินเตอร์รัพต์เวคเตอร์ที่ต้องการ ถ้าอินเตอร์รัพต์เวกเตอร์ใดถูกใช้งานอยู่ก่อนแล้ว ต้องทำการเก็บค่าเดิมไว้ เมื่อหลุดจากการทำงานของโปรแกรมฝังตัวแล้ว ก็ต้องกำหนดค่าแอดเดรสนี้ กลับคืนในอินเตอร์รัพต์ -

เวกเตอร์เดิม ( การหลุดจากการทำงาน หมายถึง การสิ้นสุดการทำงานของโปรแกรมตัวจริง กล่าวคือ หน่วยความจำที่จองไว้ทั้งหมดจะถูกส่งกลับคืนให้แก่ระบบ )

3. จัดการเรียกอินเตอร์รัพต์ที่หมายเลข 21 ฟังก์ชันที่ 0 x 31 ( 31H ) จะยึดหน่วยความจำ ( Keep ) ไว้ตลอดการทำงานของโปรแกรม หน่วยความจำส่วนนี้จะไม่ถูกใช้งานโดยโปรแกรมอื่นใดอีก

## ส่วนทำหน้าที่เป็นรูทีนให้บริการอินเตอร์รัพต์ ( Interrupt Service Routine )

โปรแกรมฝังตัวที่มีลักษณะเป็นแบบโต้ตอบ ( Interactive ) จะกระตุ้นให้ทำงาน ( Active ) โดยการกดคีย์พิเศษที่ได้กำหนดไว้ ดังนั้นหนทางในการจัดการก็คือ การยึดเอาคีย์สโตรกอินเตอร์รัพต์ ( Keystroke Intertupt ) หมายเลข 9 ที่มีอยู่ก่อนเดิม มาใช้งาน เพราะว่า โปรแกรมฝังตัวต้องยึด คีย์พิเศษที่ต้องการ ( Hot key ) ที่ระดับต่ำสุด กล่าวคือ ต้องสร้างรูทีนที่ทำหน้าที่แทนอินเตอร์รัพต์ผ่านไบออส ( Bios keyboad interrupt ) ซึ่งก็คืออินเตอร์รัพต์หมายเลข 9 นั่นเอง และก่อนที่จะทำเช่นนั้น ต้องทำการเก็บแอดเดรสของ อินเตอร์รัพต์หมายเลข 9 เดิมก่อน

## ส่วนแอพลิเคชัน

เป็นส่วนงานที่ต้องการให้โปรแกรมฝังตัวทำงาน เช่น ส่วนเอดิเตอร์ ( Editor ) เป็นการส่งข้อมูล เป็นต้น

## การทำงานของโปรแกรมเรสซิเดนต์

หลังจากที่ดอสโหลดโปรแกรมเรสซิเดนต์ ลงในหน่วยความจำแล้ว ก็จะเริ่มรันโปรแกรม ซึ่งจะเปลี่ยนค่าในตารางอินเตอร์รัพต์เวคเตอร์ ตรงหมายเลขอินเตอร์รัพต์ที่ต้องการเปลี่ยน ให้ชี้ไปยังโปรแกรมบริการที่อยู่ในโปรแกรมเรสซิเดนต์ ซึ่งทำหน้าที่แทน โปรแกรมบริการอินเตอร์รัพต์เดิม จากนั้นก็จะฝังตัวเองลงในหน่วยความจำ โปรแกรมส่วนที่ฝังตัวอยู่นี้จะไม่ถูกโหลด ( Load ) โดยโปรแกรมอื่น การทำงานของโปรแกรมเรสซิเดนต์ก็มีเพียงเท่านี้ จากนั้นก็รออินเตอร์รัพต์ ซึ่งถ้ามีอินเตอร์รัพต์ตรงกับหมายเลขอินเตอร์รัพต์ที่ถูกเปลี่ยน ซีพียูก็จะกระโดดไปทำโปรแกรมอินเตอร์รัพต์ที่ฝังตัวโดยโปรแกรมเรสซิเดนต์

### การแก้ไขค่าในตารางอินเตอร์รัพต์เวคเตอร์

การเปลี่ยนหรือแก้ไขค่าในตารางอินเตอร์รัพต์เวคเตอร์สามารถทำได้ตรงๆ แต่วิธีที่นิยม คือ เรียกผ่านดอสทางอินเตอร์รัพต์ 21h ( int 21h ) โดยใช้ฟังก์ชัน 25h ในการเซตค่า และฟังก์ชัน 35h

### ขบวนการของอินเตอร์รัพต์

อินเตอร์รัพต์มีทั้งฮาร์ดแวร์อินเตอร์รัพต์ และซอฟต์แวร์อินเตอร์รัพต์ ทั้งสองจะแตกต่างกันคือ ซอฟแวร์อินเตอร์รัพต์นั้นจะเกิดขึ้นเมื่อต้องการเรียกใช้จากโปรแกรมอื่นๆ แต่ในขณะที่ฮาร์ดแวร์อินเตอร์รัพต์จะเกิดได้ทุกเวลา ที่มีการร้องขอจากอุปกรณ์รอบข้างที่ต่อกับซีพียู ดังนั้นโปรแกรมบริการอินเตอร์รัพต์จะต้องเก็บสภาวะแวดล้อมของงานหรือโปรแกรมอื่นในขณะนั้นเอาไว้ก่อน หลังจากที่ทำโปรแกรมบริการอินเตอร์รัพต์เสร็จแล้วจึงค่อยคืนสถานะเดิมกลับคืน

ในขบวนการอินเตอร์รัพต์เองนั้น ก็ได้ทำการเก็บสถานะบางอย่างเอาไว้แล้ว เมื่อเกิดอินเตอร์รัพต์ขึ้น ซีพียู 8088 จะเก็บรีจิสเตอร์แฟล็ก ( Flag ), CS และ IP ลงในสแต็ก ( Stack ) ดังนั้นการออกจากโปรแกรมอินเตอร์รัพต์ จึงต้องใช้อินเตอร์รัพต์รีเทิร์น ( IRET ) เพื่อที่จะดึงแฟล็กออกจากสแต็กด้วย

## ฮาร์ดแวร์ที่เกี่ยวข้องกับการสื่อสารข้อมูล

### พอร์ต RS 232C

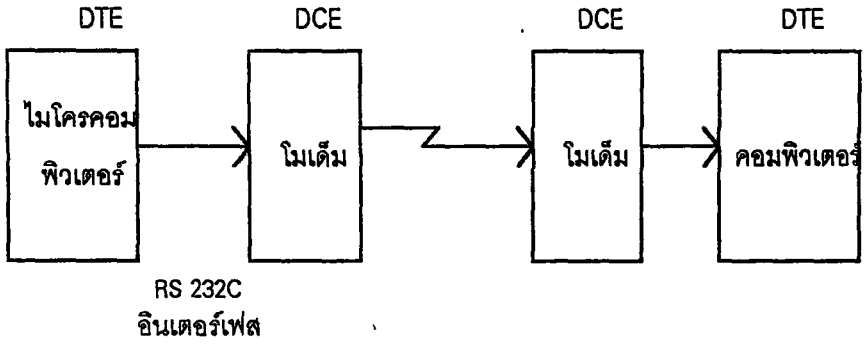
โดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรม เรียกชื่อกันว่า RS 232C อยู่ในตัวเองอยู่แล้ว หลายเครื่องไม่มีมากับเครื่อง อย่างเช่น IBM PC จำเป็นจะต้องมีการ์ดที่เรียกว่า อะซิงโครนัสอะแดปเตอร์ ( Asynchronous Communication ) มาเสียบใส่

พอร์ต RS 232C นี้ทำหน้าที่รับและส่งข้อมูลในแบบอนุกรมเรียกว่า Universal Asynchronous Adapter

### มาตรฐาน RS 232C

มาตรฐาน RS 232C ได้จัดพิมพ์ขึ้นเมื่อ ปี ค.ศ. 1969 โดยสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์แห่งสหรัฐอเมริกา RS ( Recommened Standard ) ส่วน 232 เป็นหมายเลขบ่งบอกของมาตรฐานตัวนี้ C เป็นหมายเลขของฉบับท้ายสุดของมาตรฐานตัวนี้ จุดประสงค์ของมาตรฐานตัวนี้ก็เพื่อบรรยายคุณลักษณะของการเชื่อมต่ออุปกรณ์รับส่งข้อมูลปลายทาง ( Data Terminal Equipment DTE ) กับอุปกรณ์สื่อสารข้อมูล ( Data Communication Equipment DCE ) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE ก็หมายถึงตัวไมโครคอมพิวเตอร์และ DCE ก็หมายถึงโมเดม

อุปกรณ์อื่น ๆ เช่น เครื่องพิมพ์ที่รับสัญญาณแบบอนุกรมอาจจะเป็นได้ทั้ง DTE และ DCE ขึ้น




รูปที่ 2.1 การใช้ RS 232C เชื่อมต่ออุปกรณ์

อยู่กับผู้ผลิต ข้อแตกต่างของ DTE และ DCE จะเห็นได้จาก รูปที่ 2.1 จากรูปนี้เราจะเห็นได้ว่า RS 232C มีส่วนสำคัญอย่างใหญ่หลวงสำหรับการสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์

ความจริงอีกประการหนึ่งของ RS 232C ก็คือ ความเร็วและระยะทางของการเชื่อมต่อ RS 232C สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้จาก 0 - 20,000 บิตต่อวินาที ซึ่งเพียงพอสำหรับไมโครคอมพิวเตอร์ที่มีขนาดบอร์ดเรท ( baud rate ) 110 ถึง 9,600 บอร์ด ความยาวของสายเชื่อมต่อโดยสัญญาณตามมาตรฐานของ RS 232C จำกัดอยู่แค่ 50 ฟุต ซึ่งเพียงพอสำหรับการสื่อสารไมโครคอมพิวเตอร์กับอุปกรณ์รอบนอก

## ลักษณะของสัญญาณ RS 232C

เพื่อเป็นหลักประกันว่าข้อมูลถูกส่งออกไปอย่างถูกต้อง และอุปกรณ์ถูกควบคุมอย่างถูกต้อง จำเป็นจะต้องมีข้อตกลงกันในเรื่องของสัญญาณที่ใช้ มาตรฐาน RS 232C กำหนดย่านของแรงดันไฟฟ้าในสัญญาณเพื่อสนองจุดประสงค์ข้างบน ดังแสดงในตารางที่ 2.1 และรูป 2.2

ตารางที่ 2.1

มาตรฐานของการใช้แรงดันไฟฟ้า

| แรงดันไฟฟ้า | สถานภาพลอจิก | สถานภาพของสัญญาณ | ฟังก์ชันในการควบคุม |
|---|---|---|---|
| บวก | 0 | สเปซ | ออน |
| ลบ | 2 | มาร์ค | ออฟ |

+5 _____

ย่านแรงดันไฟฟ้าบวก

+3 _____

ย่านเปลี่ยนแปลงระดับ

-3 _____

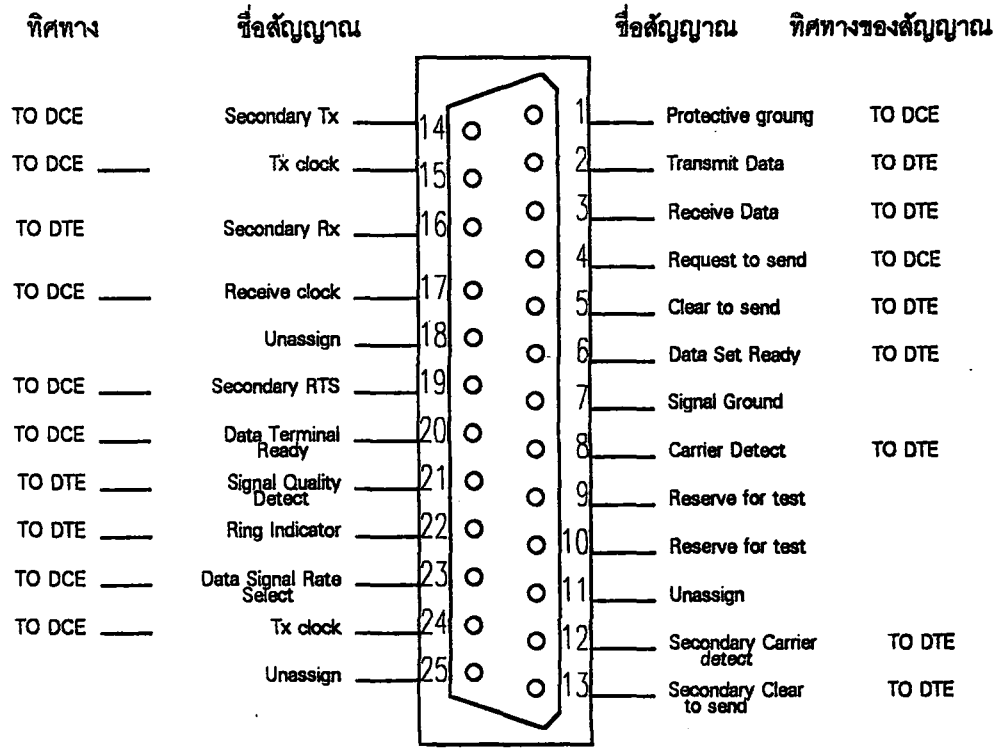
ย่านแรงดันไฟฟ้าลบ

-15 _____

รูปที่ 2.2 ย่านของแรงดันไฟฟ้าที่ใช้ในสัญญาณ RS 232C

## การกำหนดจุดข้อต่อของ RS 232C

ในทางฟิสิกส์แล้ว มาตรฐานของ RS 232C กำหนดข้อต่อแบบ DB - 25 แต่ละขาของข้อต่อกำหนดไว้ดังในรูปที่ 2.3




DTE = Data Terminal Equipment

DCE = Data Communication Equipment ( Modem )

รูปที่ 2.3 การกำหนดของข้อต่อ RS 232C

**สัญญาณต่างๆ ถูกมอบหมายให้ทำหน้าที่ดังนี้**

**Transmit Data** ( TD ขาที่ 20 )

เป็นสัญญาณที่ส่งออกจาก DTE ( หรือตัวไมโครคอมพิวเตอร์ ) ไปยังโมเด็มหรือต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น หรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออกสถานภาพของลอจิกที่ขานี้จะมีค่าเท่ากับ " 1 " หรือเทียบเท่ากับสต๊อปบิต ( stop bit )

**Receive Data** ( RD ขาที่ 3 )

เป็นทางของสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์ เมื่อไม่มีสัญญาณรับเข้ามา ขานี้จะมีสถานภาพทางลอจิก เป็น " 1 "

**Request To Send** ( RTS ขาที่ 4 )

ใช้สำหรับส่งสัญญาณไปยังโมเด็มหรือเครื่องพิมพ์เป็นการเรียกร้องที่จะส่งสัญญาณมาทางขา 2 สัญญาณนี้ใช้คู่กับ CTS ( Clear To Send ) อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับก็ส่งสัญญาณไปที่สาย CTS

**Clear To Send** ( CTS ขาที่ 5 )

ดังอธิบายไว้ใน RTS เมื่อสัญญาณนี้อยู่ในสถานะออฟ ( negative voltage หรือ ลอจิก " 1 " ) หมายความว่า อุปกรณ์รับกำลังบอกว่าพร้อมที่จะรับข้อมูลแล้ว

**Data Set Ready** ( DSR ขาที่ 6 )

เมื่อสัญญาณสายนี้อยู่ในสถานะออน ( หรือลอจิก 0 ) เป็นการบอกไมโครคอมพิวเตอร์หรือฝ่ายส่งว่า โมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้วและพร้อมที่จะส่งได้แล้ว โมเด็มที่มีการหมุนหมายเลขอัตโนมัติจะส่งสัญญาณสายนี้ไปบอกให้คอมพิวเตอร์รู้ว่าต่อโทรศัพท์ได้สำเร็จแล้ว

**Signal Ground** ( SG ขาที่ 7 )

SG ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุกๆ สายของสัญญาณ จะมีแรงดันเป็น " 0 " เมื่อเทียบกับสัญญาณตัวอื่น

Carrier Detect ( CD ขาที่ 8 )

โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออน ( ลอจิก " 0 " ) ไปบอกไมโครคอมพิวเตอร์ เมื่อได้รับสัญญาณจากโมเด็มของอีกฝ่ายหนึ่ง สัญญาณนี้จะนำไปจุด LED บอกว่าได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่งแล้ว ไฟ LED จะอยู่บนหน้าปัดของโมเด็มเอง

Data Terminal Ready ( DTR ขาที่ 20 )

คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน ( ลอจิก " 0 " ) เมื่อพร้อมที่จะติดตามกับโมเด็ม โมเด็มส่วนมากจะไม่รายงานสถานภาพของตัวเอง ( CD, USR และ CTS ) ให้คอมพิวเตอร์รู้ หากคอมพิวเตอร์ไม่เปิดสัญญาณ DTR

Ring Indicator ( RI ขาที่ 22 )

สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบโต้อัตโนมัติ ( Auto - answer ) สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งมา และออฟระหว่างเสียงดังของกระดิ่ง

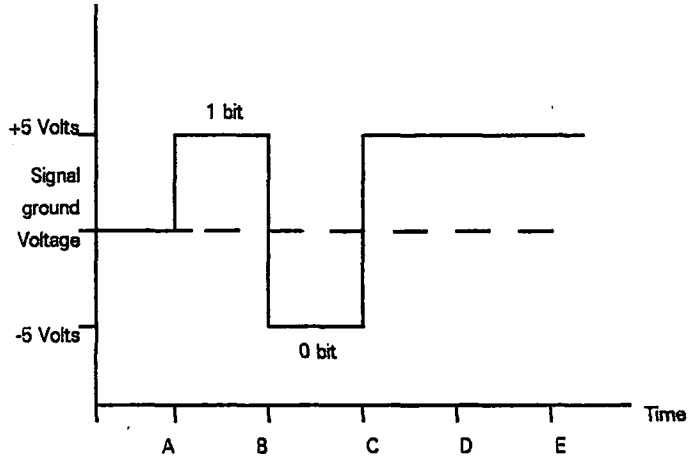
## การสื่อสารข้อมูล แบบ Asynchronous ผ่านพอร์ตอนุกรม

การสื่อสารข้อมูลผ่านเครื่องคอมพิวเตอร์ คือ การรับส่งข้อมูลระหว่างคอมพิวเตอร์เครื่องหนึ่ง กับ คอมพิวเตอร์อีกเครื่องหนึ่ง คอมพิวเตอร์เก็บข้อมูลไว้ในรูปสัญญาณไฟฟ้า โดยลักษณะของสัญญาณไฟฟ้าที่เก็บจะมีอยู่ 2 สภาวะ คือ สภาวะ " 0 " และ " 1 " เรียกสภาวะของข้อมูลนี้ว่า หนึ่งบิต( bit ) ข้อมูลหนึ่งบิตที่มีสภาวะ " 0 " หรือ " 1 " นี้อาจอยู่ในรูปของสัญญาณไฟฟ้า 0 Volt และ +5 Volt หรือ -12 Volt กับ +12 Volt ก็ได้ ขอให้มีความแตกต่างกันจนคอมพิวเตอร์แยกออกได้ว่า สัญญาณนี้คือ สภาวะ " 0 " หรือ " 1 " ก็พอ

### หลักการพื้นฐานการสื่อสารข้อมูล

ถ้าคอมพิวเตอร์เครื่องหนึ่ง ต้องการติดต่อสื่อสารรับส่งข้อมูล กับอีกเครื่องหนึ่ง สิ่งที่จะต้องกระทำคือ การเชื่อมโยงให้เกิดช่องสัญญาณระหว่างเครื่องทั้งสอง ถ้าเป็นการส่งในลักษณะทางเดียวในสายสัญญาณช่องเดียว เรียก การส่งแบบนี้ว่า ซิมเพล็กซ์ ( Simplex ) แต่ถ้าสายสัญญาณช่องเดียว แต่ผลัดกันรับส่งโดยตลอดเวลา คนละขณะ จะเรียกการรับส่งแบบนี้ว่า ฮาว์ฟดูเพล็กซ์ ( Half Duplex ) แต่ถ้าสามารถส่งสัญญาณไปกลับในเวลาเดียวกันหรือพร้อมกัน เรียกว่า ฟูลดูเพล็กซ์ ( Full Duplex )

การรับส่งข้อมูล คือ การเปลี่ยนแปลงแรงดันไฟฟ้าภายในสายสัญญาณ ให้มีค่าต่างๆ กัน ตามลักษณะของข้อมูลที่จะส่งนั่นเอง ตัวอย่างแรงดันไฟฟ้าสายสัญญาณ เป็นดังรูป 2.4




รูปที่ 2.4 ตัวอย่างแรงดันไฟฟ้าในสายสัญญาณ

## การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรม หรือการสื่อสารผ่านพอร์ตอนุกรมนี้ มีวิธีการ 2 วิธี คือ

1. การสื่อสารแบบ Asynchronous
2. การสื่อสารแบบ Synchronous

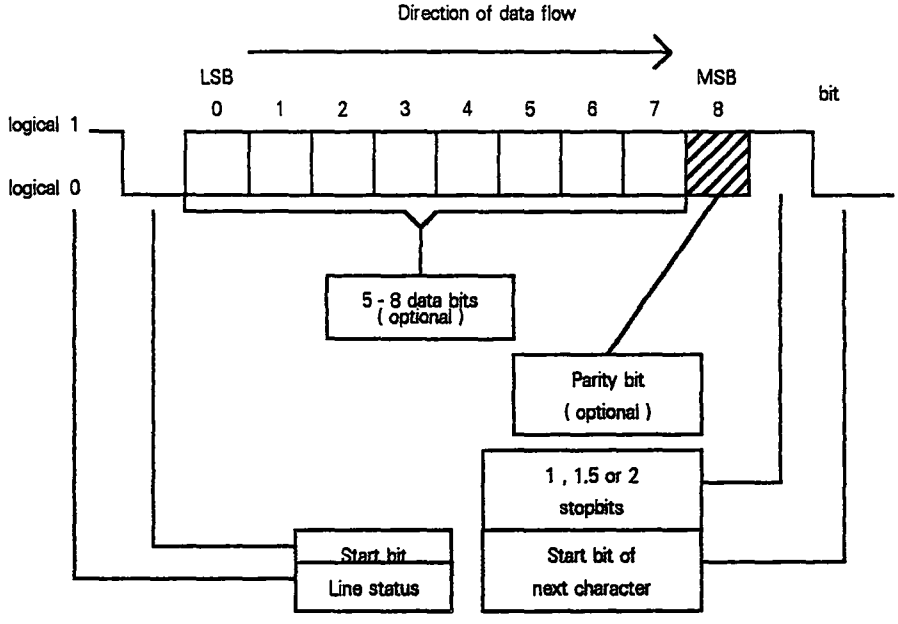
เนื่องจากโครงงานนี้ใช้การสื่อสารแบบ Asynchronous จึงจะขอกล่าวถึงการส่งข้อมูลแบบ Asynchronous เท่านั้น

## การสื่อสารแบบ Asynchronous

การส่งแบบ Asynchronous นี้ รูปแบบของการส่งจะประกอบด้วยบิตเริ่มต้น เรียกว่า Start Bit บิตข้อมูล เรียกว่า Data Bit , พาริตี้ เรียกว่า Parity Bit และบิตสิ้นสุดเรียกว่า Stop Bit ขณะที่สถานะของการส่งข้อมูลเป็นแบบว่าง ( idle ) คือ ไม่มีสัญญาณส่งออกมา จะมีสัญญาณหรือมีแรงดัน ( กระแส ) ตลอดเวลา เพื่อความแน่ใจว่าฝ่ายรับยังติดต่ออยู่กับฝ่ายส่ง

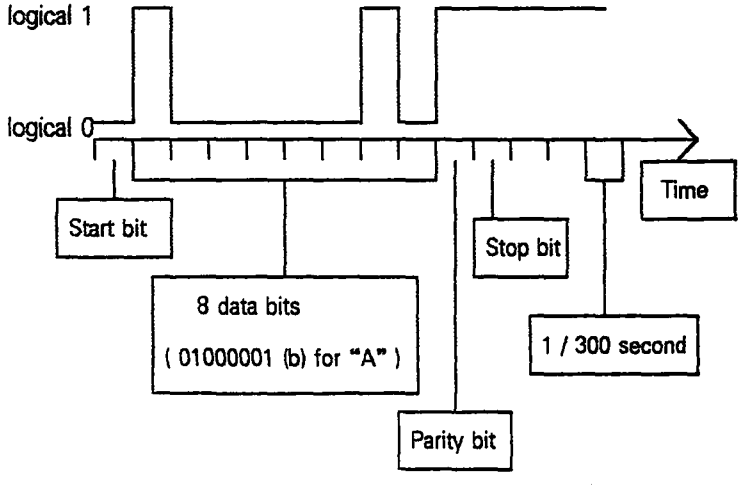
เมื่อเริ่มจะส่งข้อมูล สัญญาณของ Asynchronous จะเป็น " 0 " หนึ่งช่วงสัญญาณนาฬิกา บิตนี้คือ Start Bit หลังจากนี้จะตามด้วยบิตที่เป็นข้อมูล คือ Data Bit สำ หรับ 1 ตัวอักษร อาจมีขนาดตั้งแต่ 5 บิต จนถึง 8 บิต โดยที่ไม่มีค่านัยสำคัญน้อยที่สุด ( LSB ) จะถูกส่งออกไปก่อน ไล่ไปจนถึงบิตที่มีค่านัยสำคัญสูงสุด ( MSB ) หลังจากจบบิตข้อมูล หรือ

Data Bit แล้ว อาจมีการใส่ Parity Bit ซึ่งอาจเป็นแบบคี่ ( Odd ) หรือแบบคู่ ( Even ) รูปแบบของการส่ง เป็นดังรูป 2.5




รูปที่ 2.5 รูปแบบการส่งข้อมูลเป็นบิต

ตัวอย่างการส่งตัวอักษร ' A ' เป็น ASCII ออกทางพอร์ตอนุกรม แบบ Asynchronous เป็นดังรูป 2.6




รูปที่ 2.6 Transmitting A character : 8-bit word length, 1 stop bit , odd parity and 300 baud

การรับส่งข้อมูลแบบ Asynchronous ออกทางพอร์ตอนุกรมนั้น จะต้องมีการตั้งค่าสถานะต่างๆ ของทางฝ่ายรับและฝ่ายส่ง ให้ตรงกันด้วย ค่าสถานะต่างๆ มีดังนี้

- ความเร็วของการรับส่งข้อมูล ( Baud Rate ) ที่นิยมใช้กันตั้งแต่ 300 , 1200 , 2400 , 3600 , 4800 , 9600 และ 19200 baud

- ความยาวของข้อมูล 1 ตัวอักษร ( Data Bit ) หมายถึง จำนวนบิตในหนึ่งตัวอักษร มีได้ตั้งแต่ 5 บิต , 6 บิต , 7 บิต และ 8 บิต

- บิตพาริตี้ ( Parity Bit ) ถ้ามีการใช้ Parity ในการตรวจสอบข้อผิดพลาด สามารถใช้ได้ทั้งแบบ even และ odd

- จำนวนบิตสิ้นสุด ( Stop Bit ) สามารถกำหนดจำนวนบิตสิ้นสุดได้ว่า จะใช้ 1 บิต , 1 - 5 บิต หรือ 2 บิต

## Data Compression

Data Compression คือ กรรมวิธีต่างๆ ที่สามารถช่วยลดขนาดของข้อมูล ทั้งนี้เพื่อประหยัดเนื้อที่ ที่ใช้ในการจัดเก็บข้อมูล และยังทำให้การโอนย้าย(Transfer) ข้อมูลระหว่างเครื่องคอมพิวเตอร์ เป็นไปได้อย่างรวดเร็ว

การทำ Data Compression ในปัญหาพิเศษนี้ เลือกใช้อัลกอริทึม ของฮัฟฟ์แมน ( Huffman Algorithm ) เนื่องจากเป็นอัลกอริทึมที่เข้าใจง่าย และมีประสิทธิภาพไม่น้อยไปกว่าอัลกอริทึมอื่นๆ เมื่อพิจารณาตามความเหมาะสมของการใช้งาน ในลักษณะนี้

### ที่มาของฮัฟฟ์แมนอัลกอริทึม

ฮัฟฟ์แมนอัลกอริทึม เกิดจากแนวความคิดของ ชานนอน ในปี ค.ศ. 1948 และฟาโน ในปี ค.ศ. 1949 ต่อมาในปี ค.ศ. 1952 ฮัฟฟ์แมนได้ปรับปรุงวิธีการบางส่วนใหม่ ให้เป็นวิธีการที่รับประกันได้ว่า สามารถกำจัดส่วนที่ไม่จำเป็นออกไปจากข้อมูลได้มากที่สุด ( Minimum Redundancy ) วิธีการนี้จึงได้รับการยอมรับอย่างมาก แม้ในปัจจุบันนี้ ก็ยังมีการนำไปใช้งานโปรแกรมอัดข้อมูลหลายโปรแกรม เช่น โปรแกรมจัดเก็บแฟ้มข้อมูลพีเคแวร์ ( PKWARE File Archives ) , โปรแกรมอรรถประโยชน์คอมเพลส ( Compress Utility ) ของโปรแกรมจัดการระบบยูนิกส์ เป็นต้น

ต่อมาก็ได้มีการพัฒนาการอัดข้อมูลขึ้นอีก ตามความเหมาะสมของงานที่ใช้และต้องการความประหยัดเนื้อที่ของอุปกรณ์

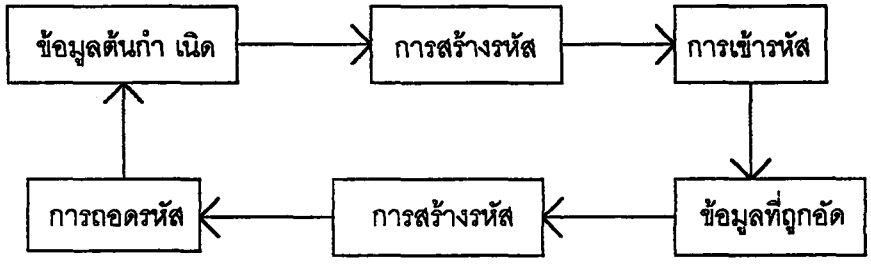
**อัลกอริทึมการอัดข้อมูล ( Data Compression Algorithm )**

อัลกอริทึมการอัดข้อมูล ประกอบด้วย 3 อัลกอริทึม คือ

1 . อัลกอริทึมการสร้างรหัส ( Code Construction Algorithm ) เป็นขั้นตอนที่ศึกษาและรวบรวมลักษณะของข้อมูล แล้วนำลักษณะของข้อมูลที่รวบรวมได้มาใช้ในการสร้างรหัสใหม่

2 . อัลกอริทึมการเข้ารหัส ( Encoding Algorithm ) เป็นขั้นตอนการแทนรหัสเดิมในแฟ้มข้อมูลต้นกำเนิด ( Source File ) ด้วยรหัสใหม่ที่ได้จากอัลกอริทึมการสร้างรหัสเมื่อจบขั้นตอนการเข้ารหัสจะได้แฟ้มข้อมูลที่ถูกอัดแล้ว ( Compressed File )

3 . อัลกอริทึมการถอดรหัส ( Decoding Algorithm ) เป็นขั้นตอนการถอดรหัสข้อมูลที่ถูกอัดให้กลับคืนเป็นข้อมูลต้นกำเนิด




รูปที่ 2.7 อัลกอริทึมการอัดข้อมูล

วิธีการของฮัฟฟ์แมน อัลกอริทึมการสร้างรหัส และอัลกอริทึมการเข้ารหัส จะทำแยกจากกันทีละขั้นตอน นั่นคือ ขั้นตอนการสร้างรหัสใหม่จากค่าความน่าจะเป็นของรหัสที่เกิดขึ้นทั้งหมดในข้อความ ต่อจากนั้นจึงเข้ารหัสข้อมูลด้วยรหัสใหม่ที่สร้างขึ้น จะได้แฟ้มข้อมูลที่ถูกอัดแล้ว เมื่อต้องการแฟ้มข้อมูลต้นกำเนิดมาใช้งานก็สามารถทำได้ โดยใช้อัลกอริทึมการถอดรหัส

**ตัวอย่าง 1** : จากข้อมูล " DEAAABBCDABBCCCAAEDB"

◆ สร้างรหัสใหม่โดยใช้อัลกอริทึมของฮัฟฟ์แมนได้ดังนี้

| รหัส | ความน่าจะเป็น | รหัสใหม่ |
|---|---|---|
| A | 0.30 | 00 |
| B | 0.25 | 01 |
| C | 0.20 | 10 |
| D | 0.15 | 111 |
| E | 0.10 | 110 |

◆ เข้ารหัสข้อมูลโดยการแทนที่รหัสเดิมด้วยรหัสใหม่ จะได้ข้อมูลที่ถูกอัดคือ " 1111100000000101... " ( เป็นข้อมูลไบนารี )

◆ ถอดรหัสข้อมูลโดยการแทนที่รหัสใหม่ด้วยรหัสเดิม ได้ข้อมูลต้นกำเนิดคือ " DEAAABBCDABBCCCAAEDB "

**คุณสมบัติของรหัส**

รหัสใหม่ที่นำมาใช้ในการอัดข้อมูลต้องมีคุณสมบัติดังต่อไปนี้

1 . สามารถถอดรหัสได้รหัสเดียว ( Unique Decodable ) หมายความว่า รหัสใหม่สำหรับแต่ละรหัสจะต้องไม่ซ้ำกัน และในการถอดรหัส จะได้ความหมายของรหัสเพียงรหัสเดียว โดยไม่สามารถถอดรหัสได้ในความหมายของรหัสอื่นอีก เช่น จาก ตัวอย่างที่ 1 รหัส " 111 " ถอดรหัสได้เป็น D เท่านั้น เพราะไม่มีรหัส " 1 ", " 11 " หรือ " 1111 " รวมอยู่ในกลุ่มรหัสใหม่

2 . รหัสที่มีคุณสมบัติถอดได้รหัสเดียว จะเป็นรหัสที่มีคุณสมบัติ เป็นรหัสนำหน้า ( Prefix Code ) หมายความว่า รหัสแต่ละรหัสจะต้องไม่นำหน้าด้วยรหัสใดๆ ที่มีอยู่ เช่น จากตัวอย่างที่ 1 รหัส " D " ( คือ " 111 " ) มีคุณสมบัติรหัสนำหน้า เนื่องจากส่วนหน้าของรหัส ( คือ " 1 " หรือ " 11 " ) ไม่ใช่รหัสใดๆ ที่มีอยู่

3 . ในการถอดรหัสของข้อมูล ที่เข้ารหัสด้วยรหัสที่มีคุณสมบัติเป็นรหัสนำหน้า ( Prefix Code ) จะสามารถถอดรหัสได้ทันที ( Instantaneously Decodable ) โดยไม่ต้องอ่านข้อมูลล่วงหน้าไปก่อน เช่น ในการถอดรหัสข้อมูล " 1000000001 " โดยใช้รหัส { 1 , 00 , 100000 } รหัสแรกของข้อมูลคือ 1 ซึ่งความเป็นจริงแล้ว ไม่สามารถ กำหนดลงไปได้จนกว่า จะอ่านข้อมูลถึงรหัสสุดท้ายของข้อมูล เพราะรหัสแรกของข้อมูลจะเป็น 100000 ทันที ถ้าจำนวนของ 0 เป็นจำนวนคี่ สำหรับรหัสตามตัวอย่างที่ 1 เป็นรหัสที่สามารถถอดรหัสได้ทันที ( Lelewer and Hirschberg 1987:264 )

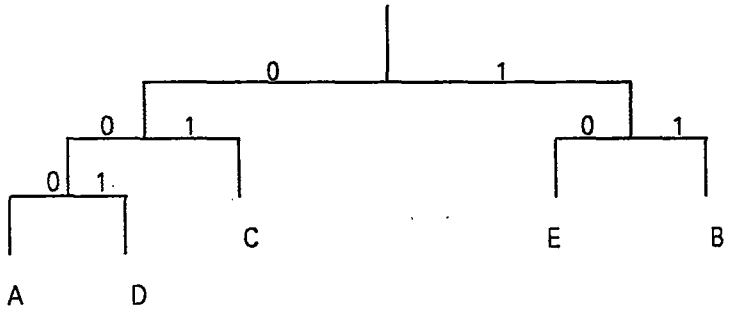
**ประสิทธิภาพการอัด** ( Compression Efficiency )

ประสิทธิภาพการอัดสามารถวัดได้จาก ค่าอัตราส่วนการอัด ( Compression Ratio ) เป็นการหาอัตราส่วนของขนาดข้อมูลที่สามารถลดลงได้ทั้งหมด เมื่อเปรียบเทียบกับขนาดของข้อมูลเดิม

$$\text{อัตราส่วนการอัด} = \frac{\text{ขนาดข้อมูลต้นกำเนิด} - \text{ขนาดข้อมูลที่ผ่านการอัด}}{\text{ขนาดข้อมูลต้นกำเนิด}}$$

**อัลกอริทึมของฮัฟฟ์แมน**

ดังที่ได้กล่าวมาแล้วว่า การสร้างรหัสของสัญลักษณ์แต่ละตัวจะมีคุณสมบัติ Unique Prefix และสามารถแทนได้ด้วยไบนารีทรี ( Binary tree ) ซึ่งก็คือ tree ที่มี Children อยู่เพียง 2 Childrens ในแต่ละโหนดเท่านั้น โดย Left Child จะแทนด้วยรหัส 0 และ Right Child แทนด้วยรหัส 1 ดังรูปที่ 2.8




รูปที่ 2.8 แสดง Simple Code Tree

**ขั้นตอนการเข้ารหัสฮัฟฟ์แมน** ( Huffman Coding )

การเข้ารหัส ( Coding ) ฮัฟฟ์แมนใช้ไบนารีทรีช่วย ( Binary Tree ) ซึ่งจะสร้างจากล่างขึ้นบน แต่ละโหนดจะมี weight โดยพิจารณาจาก ความถี่หรือความน่าจะเป็นของการปรากฏอักษรนั้นๆ การสร้างไบนารีทรีทำได้ตามขั้นตอนดังนี้ :

- นำ อักษรที่มี weight ต่ำที่สุด 2 ตัว บรรจุลงโหนดที่ว่าง 2 โหนด
- Parent node สำหรับ 2 โหนด ที่สร้างขึ้นมี weight เท่ากับผลรวมของ 2 child node
- เลือกอักษรที่เหลือที่มี weight ต่ำที่สุดอีกครั้ง ทำการเปรียบเทียบกับ Parent node ก่อนหน้านี้ นำ Parent node ที่มี weight มากกว่าอยู่ทางด้านขวา ถ้าน้อยอยู่ด้านซ้าย
- Parent node ใหม่ที่ได้กำหนดให้มี weight เท่ากับ ผลรวมของทั้งสอง Parent nodes เก่า ย้อนกลับไปพิจารณาขั้นตอนแรกใหม่อีก ทำจนกระทั่งหมดข้อความที่ต้องการอัด ( Compress ) ในที่สุดก็จะได้ไบนารีทรีของฮัฟฟ์แมน

จากไบนารีทรีนี้ สามารถสร้างรหัสสำหรับอักษรต่างๆ ได้ โดยอ่านจาก Leaf node ไป Root node จากรูปที่ 2.7 สามารถอ่านรหัสได้ดังนี้

A : 0

B : 11

C : 010

D : 001

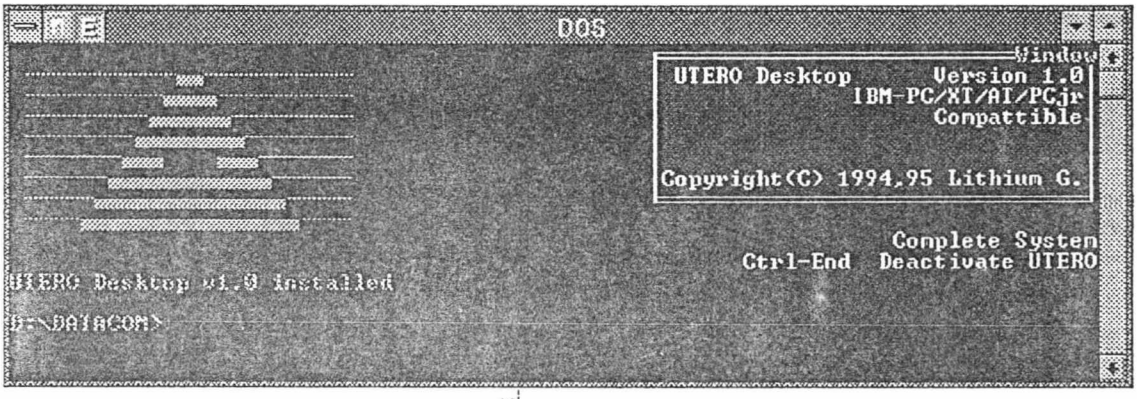
E : 10

# บทที่ 3
# การวิจัยและดำเนินการ

จากการระบบปฏิบัติการที่ดอสพร้อมพ์ ( DOS PROMPT ) สามารถเข้าสู่โปรแกรมการสื่อสารข้อมูล ชื่อ UTERO ดังนี้

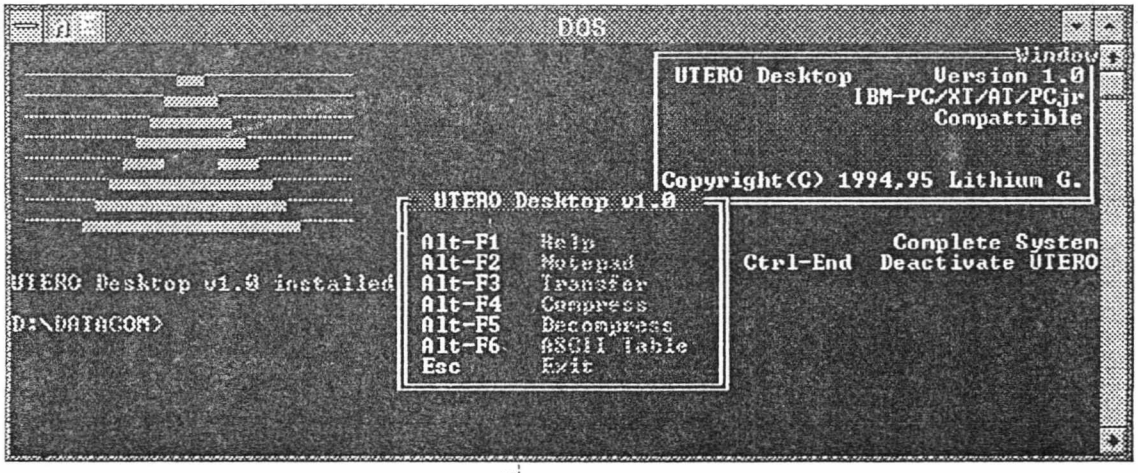
```
C:\UTERO
```

โปรแกรม UTERO.EXE ก็จะเริ่มปฏิบัติการ ( Excecute ) ดังรูป 3.1




รูปที่ 3.1

จากนั้นกดปุ่ม Ctrl + tab เพื่อเข้าสู่การทำงานของโปรแกรมการส่งข้อมูล ซึ่งจะเกิดเมนูการทำงาน แล้วสามารถเรียกใช้งานได้ตามคีย์ที่กำหนด ดังรูป 3.2

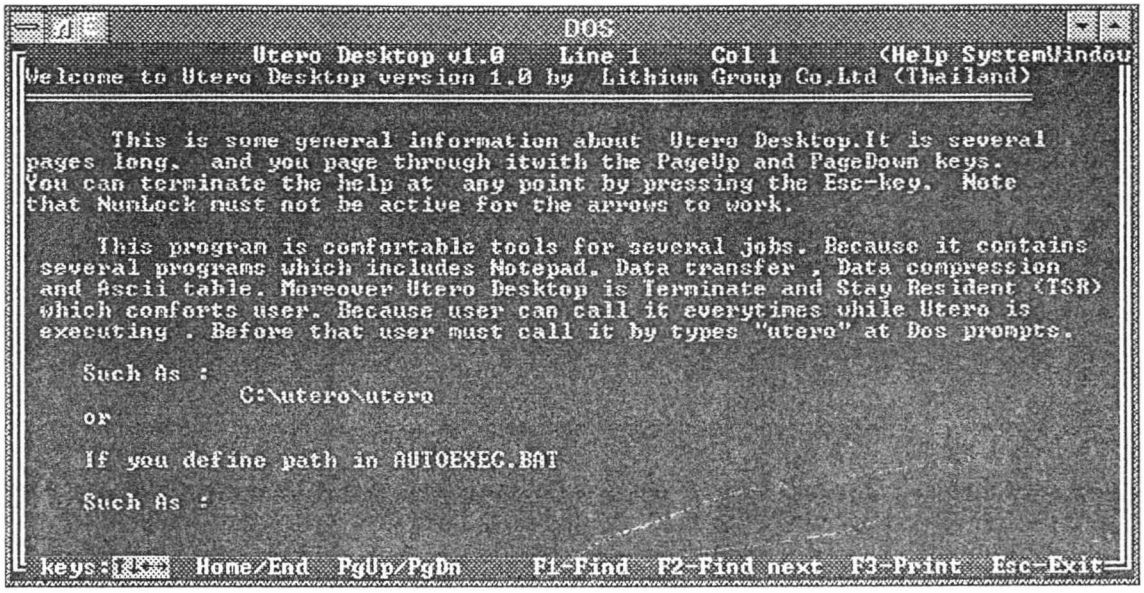



รูปที่ 3.2

## การทำงานของเมนูหลัก

Alt - F1

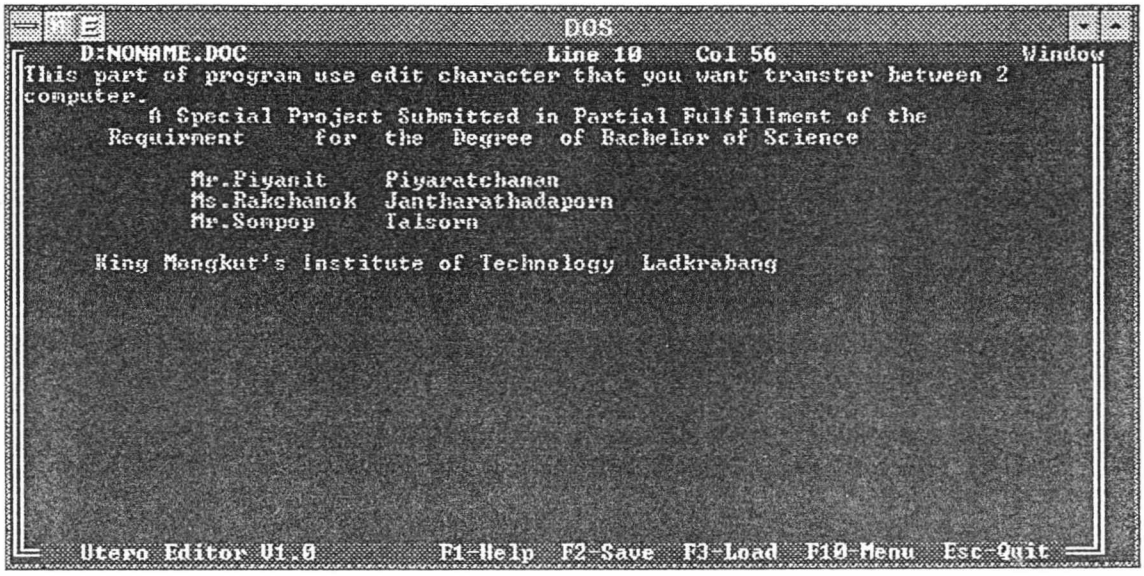
เพื่ออำนวยความสะดวกแก่ผู้ใช้ ( User ) ซึ่งผู้ใช้สามารถทำความเข้าใจการทำงานของโปรแกรมได้ด้วยตนเอง โดยการอ่านจาก Help ดังรูป 3.3




รูปที่ 3.3

Alt - F2

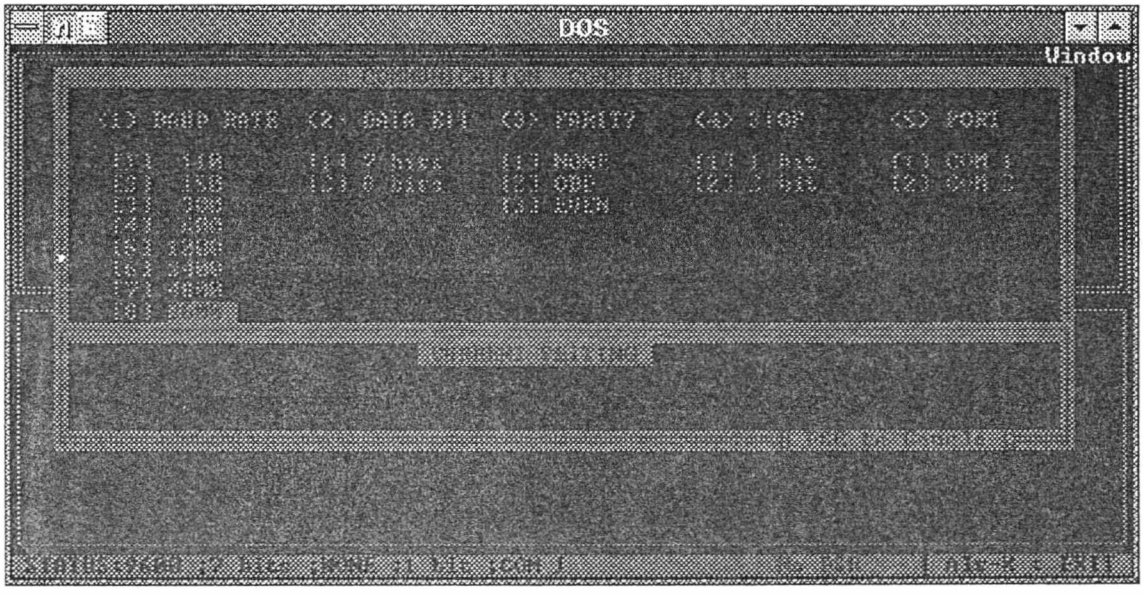
เป็นการเข้าสู่โปรแกรมเอดิเตอร์ ( Editor ) **ซึ่งมีคุณสมบัติดังที่กล่าวมาแล้วในบทที่ 2 และสามารถออกจากโปรแกรมได้โดย กดปุ่ม** Esc ดังรูป 3.4




รูปที่ 3.4

Alt - F3

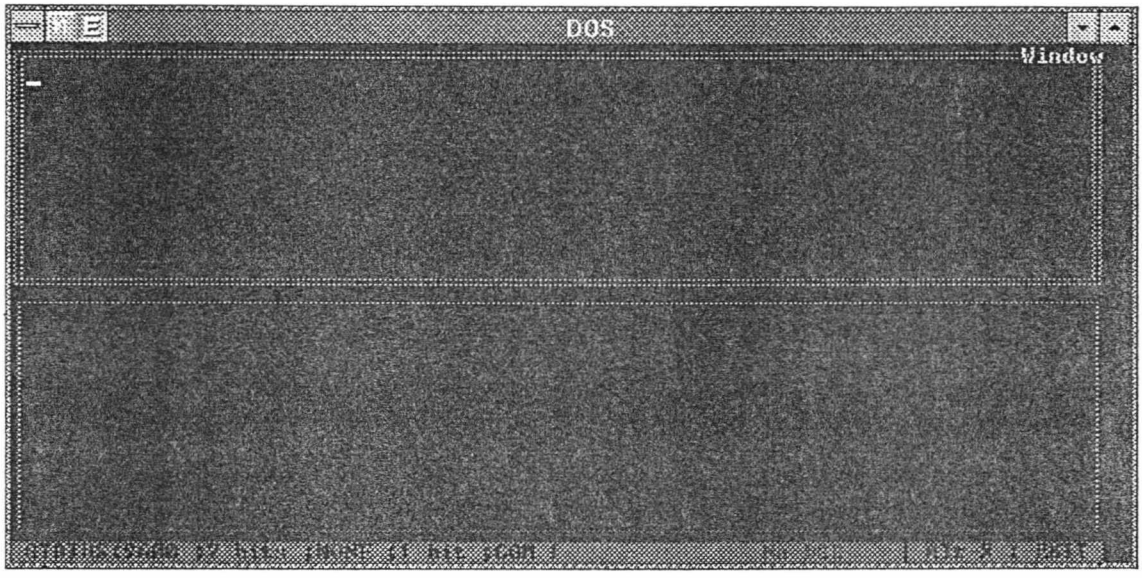
สำหรับนำข้อมูลที่ได้จากการพิมพ์ไว้ในส่วน Editor มาทำการส่งข้อมูลข้ามเครื่องไปอีกเครื่องหนึ่ง ซึ่งก่อนที่เราจะทำการส่งข้อมูล เราสามารถกำหนด Configuration ตามต้องการ โดยการกด F10 ดังรูป 3.5




รูปที่ 3.5

ซึ่งการส่งข้อมูลมีอยู่ด้วยกัน 2 ลักษณะ คือ

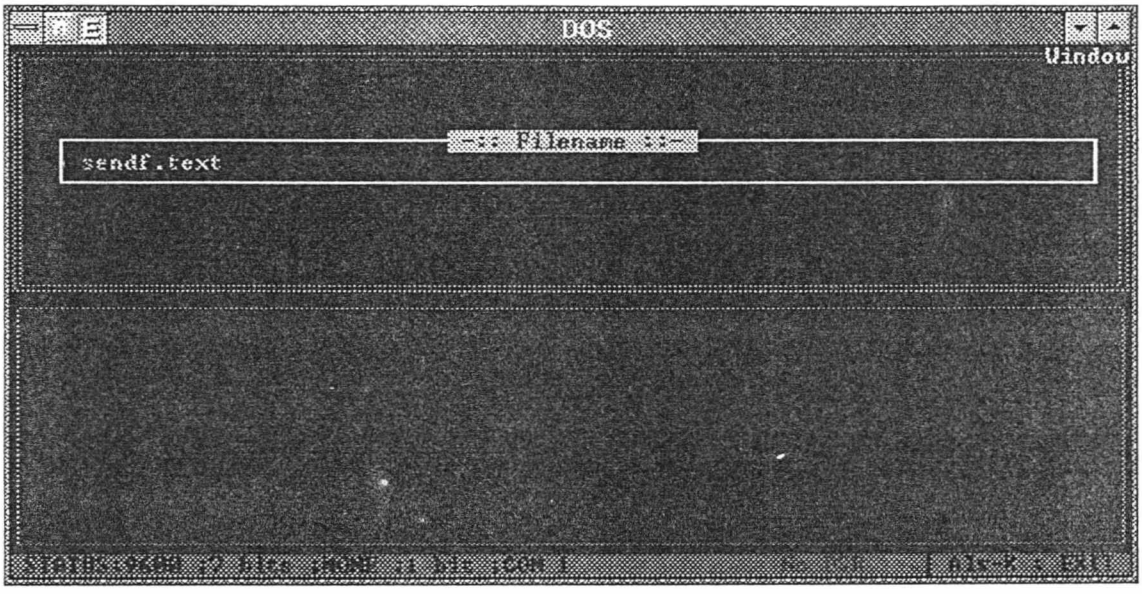
- การส่งแบบเป็นตัวอักษร ( Character )




รูปที่3.6

จากรูป 3.6 บล็อก ( block ) บนเป็นส่วนของข้อมูลที่เราต้องการส่งให้กับอีกฝ่ายหนึ่ง และบล็อกล่างจะเป็นส่วนที่อีกฝ่ายต้องการส่งให้กับเรา

- การส่งแบบเป็นไฟล์ นำข้อความที่พิมพ์ไว้ ซึ่งเราตั้งชื่อไว้แล้ว มาเป็นส่วนที่ต้องการส่งไฟล์ โดยการกด F1 จะได้ดังรูป 3.7




รูปที่ 3.7

Alt - F4

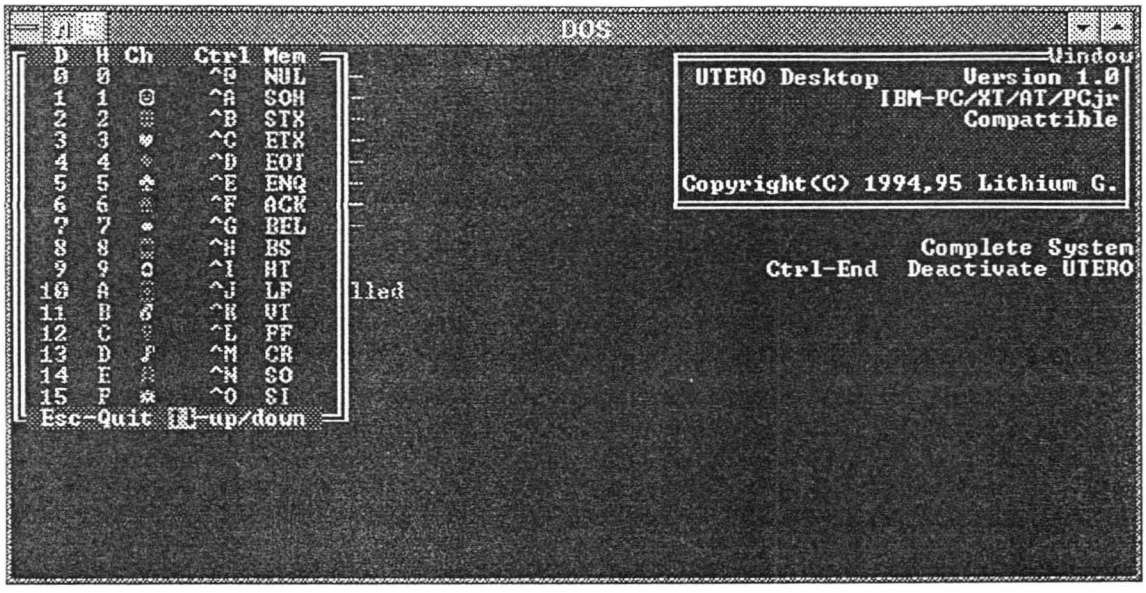
เป็นส่วนของการบีบอัดข้อมูล โดยการใช้อัลกอริทึมของฮัฟฟ์แมน ( Huffman Algorithm ) ก่อนทำการส่งข้อมูลไปอีกเครื่องหนึ่ง

Alt - F5

ส่วนขยายข้อมูลที่ถูกบีบอัด ( Decompression ) หลังจากที่อีกฝ่ายหนึ่งบีบอัดข้อมูลส่งมา

Alt - F6

เป็นส่วนที่แสดงรหัสแอสกีต่าง ๆ ดังรูปที่ 3.8




รูปที่ 3.8

ถ้าต้องการจะออกจากโปรแกรม จึงกดปุ่ม Esc ก็จะเป็นการสิ้นสุดการทำงานของโปรแกรม พร้อมทั้งจะทำการปลดปล่อยหน่วยความจำที่จองไว้ด้วย

# บทที่ 4
# ผลการวิจัยและวิจารณ์

จากโปรแกรมที่ได้จัดทำขึ้นนี้ ค่อนข้างทำงานได้ตามที่คาดหวังไว้ กล่าวคือเป็นโปรแกรมที่ใช้งานง่าย ไม่ซับซ้อน มีการติดต่อกับผู้ใช้ ( User Interface ) ที่เข้าใจได้ง่าย โดยมีลักษณะการติดต่อกับผู้ใช้คล้าย โปรแกรม SIDEKICK ของบริษัท บอร์แลนด์ ซึ่งเป็นที่คุ้นเคยกัน

ผลที่ได้จากการศึกษาพบว่า การเขียนโปรแกรมฝังตัวในหน่วยความจำนั้น จะมีข้อจำกัดในการเขียนค่อนข้างมาก อาทิเช่น การอินเตอร์รัพต์ของดอส บางฟังก์ชันไม่สามารถทำงานได้ขณะอยู่ในสภาวะการฝังตัว เช่น อินเตอร์รัพต์การจัดสรรหน่วยความจำ ( อินเตอร์รัพต์หมายเลข 21H ฟังก์ชันที่ 48H ) การใช้อินเตอร์รัพต์เพื่อสิ้นสุดโปรแกรม ( Exit ) เป็นต้น

ในส่วนของเอดิเตอร์นั้น สามารถทำงานได้ตามที่คาดหวังไว้ ซึ่งมีคุณสมบัติพื้นฐานการทำงานที่คล้ายกับ โปรแกรมเอดิเตอร์โดยทั่วไป แต่ทว่ายังมีข้อบกพร่องอีกบางจุด อาทิ โปรแกรมเอดิเตอร์ที่จัดทำนี้ เป็นเอดิเตอร์ที่ใช้ 80 คอลัมน์ ( Documentd Editor ) ซึ่งตามปกติแล้วเอดิเตอร์โดยทั่วไปจะมีจำนวนคอลัมน์ที่มากกว่านี้ ซึ่งจะเห็นว่าค่อนข้างน้อยเมื่อเทียบกันแล้ว และยังวาดการกำหนดบล็อก ( block ) ในเอดิเตอร์ เพื่อทำการ ตัด , ก๊อปปี้ และ แปะ เพราะระยะเวลาในการจัดทำค่อนข้างน้อยจึงไม่เพียงพอ

การส่งแฟ้มข้อมูล ( File Transfer ) ทำงานได้ค่อนข้างดี แต่ยังมีข้อจำกัด คือใช้ได้กับเฉพาะพอร์ตอนุกรม ( Serial Port ) เท่านั้น ไม่สามารถใช้กับพอร์ตแบบขนานได้ และยังเป็นการส่งในรูปแบบ Synchronous อีกด้วย

จากการศึกษาพบว่า ไม่สามารถทำการบีบอัดข้อมูลได้ ถ้ายังคงอยู่ในสถานะการเป็นโปรแกรมฝังตัว เพราะว่าในส่วนของการบีบอัดข้อมูลนั้น จะมีการจัดสรรหน่วยความจำขณะทำการรันโปรแกรมด้วย ( Dynamic Memory Allocating ) ซึ่งจากการศึกษาพบว่า ไม่สามารถจัดการกับหน่วยความจำเช่นนี้ได้ เพราะระบบปฏิบัติการดอส ไม่ช่วยเหลือการจัดสรรนี้ แต่เมื่อส่วนของการบีบอัดข้อมูลนี้อยู่ในสถานะรับตามปกติ จะสามารถทำงานบีบอัดข้อมูลได้ตามปกติ ซึ่งข้อบกพร่อง ในการบีบอัดข้อมูลด้วยวิธีของฮัฟฟ์แมน ( Huffman ) นี้ คือ จะทำการบีบอัดข้อมูลได้เฉพาะแฟ้มข้อมูลตัวอักษรเท่านั้น ( Text File )

ข้อบกพร่องโดยรวมอย่างหนึ่งของโปรแกรมนี้ คือ การที่โปรแกรมมีขนาดที่ค่อนข้างใหญ่ ดังนั้น จึงกินเนื้อที่หน่วยความจำค่อนข้างมากตามไปด้วย คือกินหน่วยความจำประมาณ 128

กิโลไบต์ ซึ่งบางครั้งถ้าผู้ใช้ต้องการโหลดโปรแกรมอื่นใดที่มีขนาดใหญ่ ๆ แล้วอาจจะทำให้โปรแกรมนั้นไม่สามารถทำงานได้ ซึ่งตามปกติแล้ว การเขียนโปรแกรมฝังตัวในหน่วยความจำนี้ มักจะใช้ภาษาแอสเซมบลีในการเขียน ( ภาษาเครื่อง ) แต่โปรแกรมที่จัดทำขึ้นเขียนด้วยภาษาซี ซึ่งเป็นภาษาระดับสูง ทำให้โปรแกรมที่ได้มีขนาดใหญ่กว่าโปรแกรม ที่เขียนด้วยภาษาแอสเซมบลี

# บทที่ 5
# บทสรุปการศึกษาพัฒนาและข้อเสนอแนะ

## สรุปผลการศึกษาพัฒนา

ในปัญหาพิเศษฉบับนี้ จุดมุ่งหมายหลักก็คือการสร้างเครื่องมืออำนวยความสะดวก ซึ่งประกอบไปด้วย ส่วนเอดิเตอร์, การส่งไฟล์ข้ามเครื่อง (file transfer) และการบีบอัดข้อมูล (data compression) จากการศึกษาพบว่า โปรแกรมที่จัดทำขึ้นนี้ยังทำงานได้ค่อนข้างจำกัด ควรจะปรับปรุงในบางจุดดังนี้

ในส่วนของเอดิเตอร์ จำนวนคอลัมน์ที่สามารถเก็บข้อมูลได้ ควรจัดการให้มีคอลัมน์มากกว่าเดิมคือมากกว่า 80 คอลัมน์ นอกจากนี้ควรเพิ่ม ให้มีการกอปปี้ข้อความ, การตัดแปะ, การค้นหาคำ เพื่อความสะดวกรวดเร็วในการใช้

ส่วนของการส่งแฟ้มข้อมูล ( file transfer ) ควรให้ทำงานได้ทั้งบนพอร์ตอนุกรมและพอร์ตขนาน ให้มีการส่งไฟล์ได้รวดเร็วขึ้น และส่งแฟ้มข้อมูลที่มีขนาดใหญ่เกิน 64 กิโลไบต์ได้

ส่วนของการบีบอัดข้อมูล (data compression) ควรใช้อัลกอริธึมอื่นๆ ช่วยในการลดขนาดข้อมูล ซึ่งจะทำให้สามารถบีบอัดข้อมูลได้ทั้งไฟล์ตัวอักษร ( text file ) และ binary file และทำให้การบีบอัดข้อมูลเป็นไปได้รวดเร็ว และบีบอัดแฟ้มข้อมูลขนาดใหญ่ได้

โปรแกรมที่ได้จัดทำขึ้นนี้มีขนาดใหญ่ ต้องใช้เนื้อที่ในหน่วยความจำมาก เมื่อเขียนโปรแกรมแบบฝังตัว อาจทำให้โปรแกรมมีปัญหาได้ ดังนั้นในการเขียนโปรแกรมฝังตัวควรเขียนเป็นภาษาแอสเซมบลี้เพื่อให้โปรแกรมมีขนาดเล็ก

# ภาคผนวก

```
/*****************************************
*     Main Program for Editor : PCTSR.C     *
*****************************************/
#define _STRUCT_WINDOW_
#define WIN_PROTOTYPE

#include "window.h"
#include <stdio.h>
#include <dos.h>
#include <process.h>

#define SCREEN 1
#define MEMORY 0
#define STK_SIZE 0x2000     /* 8192 bytes */

void interrupt new_int8(void);
void interrupt new_int28(void);
void interrupt tsr_keystroke(void);
void activate_sctsr(void);
void _cursor_pos(void);
void cursor_off(void);
void cursor_on(void);
int load_logo(char *filename);
void display_menu(void);
int load_to_chunk(char *filename);

void interrupt (*old_int80)();
void interrupt (*old_int9)();
void interrupt (*old_int28)();
void interrupt (*old_int8)();

char far *dos_active;
char busy,d_key;
unsigned char stack[STK_SIZE];
unsigned int sp,ss;
```

```
char video;
int old_row,old_col;

int main(void)
{
  union REGS reg;
  struct SREGS s_reg;

  old_int80 = getvect(0x80);
  set_vid_mem();
  if (!old_int80)  {
     clrscr();
     if (!load_logo("logo.txt")) {
          make_window(0,"  Intermediate window  ","",0,0,42,5,3);
          window_act(0);
          write_string(2,1,"Define path to logo.txt for logo of ",15);
          write_string(2,2,"\"Lithium Group\"  or use backup",15);
          write_string(2,4,"      Press any key to continue...",2);
          while (!kbhit());
          deactivate(0);
     }
     _window("","",47,0,79,7,3);
     write_string(48,1," UTERO Desktop       Version 1.0",31);
     write_string(48,2,"              IBM-PC/XT/AT/PCjr",31);
     write_string(48,3,"                   Compattible",31);
     write_string(48,4,"                               ",31);
     write_string(48,5,"                               ",31);
     write_string(48,6,"Copyright(C) 1994,95 Lithium G.",31);

     write_string(65,9,"Complete System",15);
     write_string(54,10,"Ctrl-End  Deactivate UTERO",15);
  }

  if (!old_int80)
     setvect(0x80,new_int8);
```

```
  else {
    printf("UTERO Desktop v1.0 already loaded - Press Ctrl-Tab for activate\n");
    exit(0);
  }

  reg.h.ah = 0x34;
  int86x(0x21,&reg,&reg,&s_reg);
  dos_active = (char far *) MK_FP(s_reg.es,reg.x.bx);


  old_int9 = getvect(0x9);
  old_int28 = getvect(0x28);
  old_int8 = getvect(0x8);   /* timer interrupt */

  setvect(0x9,tsr_keystroke);
  setvect(0x28,new_int28);
  setvect(0x8,new_int8);

  gotoxy(1,12);
  fprintf(stdout,"UTERO Desktop v1.0 installed\n");
  keep(0,9000);
}

void display_menu(void)
{
  make_window(1,"  UTERO Desktop v1.0  ","",28,7,52,16,2);
  window_act(1);
  window_cls(1);
  write_string(29,8,"                          ",15);
  write_string(29,9," Alt-F1",15);write_string(36,9,"   Help          ",11);
  write_string(29,10," Alt-F2",15);write_string(36,10,"   Notepad       ",11);
  write_string(29,11," Alt-F3",15);write_string(36,11,"   Transfer      ",11);
  write_string(29,12," Alt-F4",15);write_string(36,12,"   Compress      ",11);
  write_string(29,13," Alt-F5",15);write_string(36,13,"   Decompress    ",11);
  write_string(29,14," Alt-F6",15);write_string(36,14,"   ASCII Table   ",11);
```

```
  else {
    printf("UTERO Desktop v1.0 already loaded - Press Ctrl-Tab for activate\n");
    exit(0);
  }

  reg.h.ah = 0x34;
  int86x(0x21,&reg,&reg,&s_reg);
  dos_active = (char far *) MK_FP(s_reg.es,reg.x.bx);


  old_int9 = getvect(0x9);
  old_int28 = getvect(0x28);
  old_int8 = getvect(0x8);   /* timer interrupt */

  setvect(0x9,tsr_keystroke);
  setvect(0x28,new_int28);
  setvect(0x8,new_int8);

  gotoxy(1,12);
  fprintf(stdout,"UTERO Desktop v1.0 installed\n");
  keep(0,9000);
}

void display_menu(void)
{
  make_window(1,"  UTERO Desktop v1.0  ","",28,7,52,16,2);
  window_act(1);
  window_cls(1);
  write_string(29,8,"                          ",15);
  write_string(29,9," Alt-F1",15);write_string(36,9,"   Help          ",11);
  write_string(29,10," Alt-F2",15);write_string(36,10,"   Notepad       ",11);
  write_string(29,11," Alt-F3",15);write_string(36,11,"   Transfer      ",11);
  write_string(29,12," Alt-F4",15);write_string(36,12,"   Compress      ",11);
  write_string(29,13," Alt-F5",15);write_string(36,13,"   Decompress    ",11);
  write_string(29,14," Alt-F6",15);write_string(36,14,"   ASCII Table   ",11);
```

```
  else {
    printf("UTERO Desktop v1.0 already loaded - Press Ctrl-Tab for activate\n");
    exit(0);
  }

  reg.h.ah = 0x34;
  int86x(0x21,&reg,&reg,&s_reg);
  dos_active = (char far *) MK_FP(s_reg.es,reg.x.bx);


  old_int9 = getvect(0x9);
  old_int28 = getvect(0x28);
  old_int8 = getvect(0x8);   /* timer interrupt */

  setvect(0x9,tsr_keystroke);
  setvect(0x28,new_int28);
  setvect(0x8,new_int8);

  gotoxy(1,12);
  fprintf(stdout,"UTERO Desktop v1.0 installed\n");
  keep(0,9000);
}

void display_menu(void)
{
  make_window(1,"  UTERO Desktop v1.0  ","",28,7,52,16,2);
  window_act(1);
  window_cls(1);
  write_string(29,8,"                          ",15);
  write_string(29,9," Alt-F1",15);write_string(36,9,"   Help          ",11);
  write_string(29,10," Alt-F2",15);write_string(36,10,"   Notepad       ",11);
  write_string(29,11," Alt-F3",15);write_string(36,11,"   Transfer      ",11);
  write_string(29,12," Alt-F4",15);write_string(36,12,"   Compress      ",11);
  write_string(29,13," Alt-F5",15);write_string(36,13,"   Decompress    ",11);
  write_string(29,14," Alt-F6",15);write_string(36,14,"   ASCII Table   ",11);
```

```
write_string(29,15," Esc  ",15);write_string(36,15,"  Exit        ",11);

cursor_off();
do {
  key.int_key = bioskey(0);
  if ((key.uch_key[1] >= 104 && key.uch_key[1] <= 109) || key.uch_key[0] == 27)  {
       if (!key.uch_key[0])  {
         switch(key.uch_key[1])  {
           case 104 : deactivate(1);  /* Alt-F1  Help */
                      help_system();
                      break;

           case 105 : deactivate(1);  /* Alt-F2  Notepad */
                      notepad();
                      break;

           case 106 : deactivate(1);  /* Alt-F3  Laplink */
                      tllink();
                      break;

           case 107 : deactivate(1);  /* Alt-F4  Compress */
                      break;

           case 108 : deactivate(1);  /* Alt-F5  Decompress */
                      break;

           case 109 : deactivate(1);  /* Alt-F6  IBM ASCII table */
                      table();
                      break;
       } /* switch */
  }
  else if (key.uch_key[0] == 27)
         deactivate(1);
  break;
  } /* if */
```

```
      else
       continue;
   } while (1);
   cursor_on();
}

int load_logo(char *filename)
{
  FILE *fp;
  int ch;
  int row,col;

  if ((fp = fopen(filename,"rb")) == NULL)
     return 0;
  for (row = 1,col = 1; !feof(fp) ; col++ ) {
    ch = fgetc(fp);
    if (ferror(fp))
       return 0;
    if (ch == 26)    /* ^Z  because binary view also see ^Z and then end of file */
       break;
    if (ch == '\r') {
      ch = fgetc(fp);    /* '\n' */
      row++;
      col = 0;
    }
    else
      write_char(col,row,ch,11);
  }
  if (fclose(fp))
       return 0;
  return 1;
}

void interrupt new_int8(void)
{
```

```
  (*old_int8)();
  if (!*dos_active && !busy && d_key)
     activate_sctsr();
}

void interrupt new_int28(void)
{
  (*old_int28)();
  if (!busy && d_key)
    activate_sctsr();
}

void interrupt tsr_keystroke(void)
{
  int far * head = (int far *) MK_FP(0x0000,0x041A);
  unsigned char far * t = (char far *) MK_FP(0x0000,0x041A);

  (*old_int9)();

  if (*t != *(t+2))  {
    t += *t - 30 + 5;
     if ( *t == 148 || *t == 117) {
          switch(*t)  {
            case 148 : d_key = 1;          /* Ctrl-Tab Activate */
                       break;

            case 117 : d_key = 2;          /* Ctrl-End Deactivate */
                       break;
          }

          *(head+1) = *head;   /* clear buffer */
    }
  }
}
```

```
void activate_sctsr(void)
{

  disable();
  ss = _SS;
  sp = _SP;
  _SS = _DS;
  _SP = (unsigned) &stack[STK_SIZE - 2];
  enable();

  video = video_mode();
  if (!busy && (video == 7 || video == 2 || video == 3))  {
     busy = !busy;
     switch(d_key)  {
       case 1 : display_menu();
                    break;
     }
     busy = !busy;
   }

  if (d_key == 2)
     _cursor_pos();

  disable();
  _SP = sp;
  _SS = ss;
  enable();

  if (d_key == 2) {
     setvect(8,old_int8);
     setvect(9,old_int9);
     setvect(0x28,old_int28);
     setvect(0x80,old_int80);
     freemem(_psp);
   }
```

```
  else
    d_key = 0;
}

void _cursor_pos(void)
{
  union REGS regs;

  regs.h.bh = 0;
  regs.h.ah = 3;
  int86(16,&regs,&regs);

  old_row = regs.h.dh;
  old_col = regs.h.dl;
}
```

```
/************************************
* Program TRANSF.C for tranmit data *
************************************/
#define _TSREDIT_
#define WIN_PROTOTYPE

#include "window.h"
#include "tsredit.h"
#include <dos.h>
#include <bios.h>
#include <conio.h>
#include <stdlib.h>
#include <io.h>
#include <fcntl.h>
#include <stat.h>

#define video     0x10
#define comm      0x14
#define ictl      0x20              /* 8259 interrupt controller port */
#define eoi         0x20            /* end of interrupt signal */
#define ictl_1    0x21              /* 8259_1 interrupt controller port */
#define  UP       0x4800
#define  LEFT     0x4B00
#define  RIGHT    0x4D00
#define  DOWN     0x5000
#define ESC       0x001B
#define ENTER     0x000D
#define true      1
#define false     0
#define right     79
#define BOTT      24
#define MAX_BUFF  1024
#define MAX       1440
```

```
void status_bar(void);
void status_data(void);
void status_parity(void);
void status_stop(void);
void clrmscr(void);
void scroll_up(void);
void scroll_down(void);
void beep(void);
void keep_char(void);
void inc_row(void);
int  Open_file(char m);
void Rec_file(void);
int dc_report(void);
char get_data_serialport(void);
void Screen(int x1,int y1,int x2,int y2,char ch,int color);
int GetKey(void);
void get_crtseg(void);
void dc_config(void);
void clrtermscr(void);
unsigned int filesize(void);
void Send_file(void);
void baud_rate_nui(void);
void disable_port(void);
void config_nui(void);
void _frame (char x1,char y1,char x2,char y2,char type,char attrb);
void putchxy (char x,char y,char chr,char attrb);
void tllink(void);

union BIOS_KEY {
   int int_key;
   unsigned char uch_key[2];
};
```

```
static void (interrupt far *old_comm_isr)(void);
static void interrupt far new_comm_isr(void);
static char *line_message[] = {
                                    " Line ok   ",
                                    " Line error",
                                    " No CTS    ",  /* clear to send */
                                    " No DSR    ",  /* data set ready */
                                   };
static char *bdrate[] = {
                          " 110 ",
                          " 150 ",
                          " 300 ",
                          " 600 ",
                          "1200 ",
                          "2400 ",
                          "4800 ",
                          "9600 ",
                         };
static char *data[] = {
                          "7 bits ",
                          "8 bits ",
                         };
static char *parity[]={
                          "NONE ",
                          "ODD  ",
                          "EVEN ",
                         };
static char *stop[] = {
                          "1 bit ",
                          "2 bit ",
                         };
static char *comm_ch[] ={
                           "COM 1 ",
```

```
                        "COM 2 ",
                       };
char com_buf[MAX_BUFF];
char r , c, r1, c1, rs, cs, top=1, bottom=10,col=1,row=13;
char iid, attribute;
char    old_buf_port,
          old_ier_port,
          old_iir_port,
          old_lcr_port,
          old_mcr_port,
          old_lsr_port,
          old_msr_port,
          old_ictl_1 ;
char buff_ch[MAX+MAX];
int  fp;
char far *crtseg_far;
char word_length=0, wdlen  = 0x02,
    stop_bit=0,    stpbit = 0x01,
    parity_bit=0,  prbit  = 0x00,
    baud_rat=7;
char ox  = 11, oy  = 12,ox1 = 25,
    oy1 = 5,  ox2 = 39,oy2 = 5,
    ox3 = 53, oy3 = 5, ox4 = 67,oy4 = 5;

int in_p, out_p, tmp, crtseg;
int esc_state=0;
int terminate;
int buf_port, h_dvs_port, l_dvs_port, ier_port;
int iir_port, lcr_port, mcr_port, lsr_port, msr_port;
int com_channel, baud;
int count;

void beep()
```

```
{
        sound(MAX_BUFF);
        delay(32);
        sound(666);
        delay(32);
        nosound();
}

int GetKey()
{
  int c;

  c = getch();
  if (!(c & 0x00FF))
    c = getch() << 8;
        return (c);
}

void dsp_str(char x, char y, char attribute, char *string)
{
    int offset = (int) ((x<<1) + (y*160));

        while (*string != '\0')
            {
             pokeb(crtseg, offset++, *string++);
             pokeb(crtseg, offset++, attribute);
            }
}

void Screen(int x1,int y1,int x2,int y2,char ch,int color)
{
 unsigned char far *scr;
 int i,j;
```

```
        scr =(unsigned char far *) (crtseg_far +(x1<<1)+(y1*160));
        for (j=0;j<(y2-y1);j++) {
            for (i=0;i<(x2-x1);i++) {
                *scr++=ch;
                *scr++=color;
            }
       scr+=(160-2*(x2-x1));
       }
}
void putchxy (x,y,chr,attrb)
char x,y,chr,attrb;
{ unsigned char far *Segvdo;

    Segvdo=(unsigned char far *) (crtseg_far +(unsigned int)(x<<1)+(y*160));
    *Segvdo++=chr;
    *Segvdo=attrb;
}

void _frame (char x1,char y1,char x2,char y2,char type,char attrb)
{ char type_box [4][6] = { { 0xDA,0xBF,0xC0,0xD9,0xC4,0xB3 },
                           { 0xC9,0xBB,0xC8,0xBC,0xCD,0xBA },
                           { 0xD5,0xB8,0xD4,0xBE,0xCD,0xB3 },
                           { 0xD6,0xB7,0xD3,0xBD,0xC4,0xBA } };
  char tx=x1+1,ty=y1+1;

  putchxy (x1,y1,type_box [type][0],attrb);
  putchxy (x2,y1,type_box [type][1],attrb);
  putchxy (x1,y2,type_box [type][2],attrb);
  putchxy (x2,y2,type_box [type][3],attrb);

  do {
     putchxy (tx,y1,type_box [type][4],attrb);
     putchxy (tx,y2,type_box [type][4],attrb);
```

```
  } while (++tx<x2);

  do {
     putchxy (x1,ty,type_box [type][5],attrb);
     putchxy (x2,ty,type_box [type][5],attrb);
  } while (++ty<y2);
}

 void config_nui()
 {
   switch(word_length)
   {
     case 0: wdlen = 0x02; break;
     case 1: wdlen = 0x03; break;
   }
   switch(stop_bit)
   {
     case 0: stpbit = 0x00; break;
     case 1: stpbit = 0x04; break;
   }
   switch(parity_bit)
   {
     case 0: prbit = 0x00;  break;
     case 1: prbit = 0x08;  break;
     case 2: prbit = 0x18;  break;
   }
   baud = baud_rat;
 }

void status_baud(int rate)
{
          dsp_str(8,24, 0x30, bdrate[rate]);
}
```

```
void status_data()
{
        dsp_str(14,24,0x30,data[word_length]);
}

void status_parity()
{
        dsp_str(22,24,0x30,parity[parity_bit]);
}

void status_stop()
{
        dsp_str(28,24,0x30,stop[stop_bit]);
}

void status_com(int cc)
{
          dsp_str(35, 24, 0x30, comm_ch[cc]);
}

void status_bar()
{
 char *title[2] = {
                    " STATUS:    ;     ;    ;    ;    ",
                    "                          Alt-X : EXIT ",
                    };

          dsp_str(0, 24, 0x30, title[0]);
          dsp_str(40,24, 0x30, title[1]);
 }

 void ststus_line(unsigned char msr,unsigned char lsr)
 {
```

```
  int i;
   if((lsr & 0x0e) == 0x00) i = 0;
    else i =1 ;
   if((msr & 0x10) != 0x10) i = 2;
   if((msr & 0x20) != 0x20) i = 3;
   dsp_str(54,24,0x34,line_message[i]);
}

void keep_char(void)
{
  int offset;
  count  = 0;
  offset   = (int) ((r*160) + (c*2));
  do {
    buff_ch[count++]   = peek(crtseg,offset++);
    offset++;
  } while (count != 79-c);
  buff_ch[count]=0x20;
  buff_ch[count+1]='\0';
}

static void interrupt far new_comm_isr(void)
{
          iid = inportb(iir_port);
          enable();
          if ((iid & 0x04) == 0x04)
            {
              com_buf[in_p++] = (inportb(buf_port));
          if (in_p == MAX_BUFF)
              in_p = 0;
            }
          ststus_line(inportb(msr_port),inportb(lsr_port));
          disable();
```

```
        outportb(ictl,eoi);
}

void clrmscr(void)
{
        status_bar();
    status_baud(baud);
    status_data();
    status_parity();
    status_stop();
        status_com(com_channel);
    ststus_line(inportb(msr_port),inportb(lsr_port));
}

void clrtermscr(void)
{
        clrscr();
    clrmscr();
}

void goto_row_col(char row, char col)
{
        _AH = 2;
        _BH = 0;
        _DH = row;
        _DL = col;
        geninterrupt(video);
}

void scroll_up(void)
{
        _AH = 6;
        _AL = 1;
```

```
        _BH = 7;
        _CH = top;
        _CL = 1;
        _DH = bottom;
        _DL = 78;
        geninterrupt(video);
}

void scroll_down(void)
{
        _AH = 7;
        _AL = 1;
        _BH = 7;
        _CH = top;
        _CL = 1;
        _DH = bottom;
        _DL = 78;
        geninterrupt(video);
}

void get_crtseg()
{
    int eqmt;

        eqmt = biosequip() & 0x0030;
        if (eqmt == 0x0030) {
          crtseg = 0xb000;
          crtseg_far = (char far *) 0xb0000000;
        }
        else {
          crtseg = 0xb800;
          crtseg_far = (char far *) 0xb8000000;
        }
```

```
}

void dsp_char(char d_c_in)
{

        _AH = 9;
        _AL = d_c_in;
        _BH = 0;
        _BL = attribute;
        _CX = 1;
        geninterrupt(video);
}

void send_char(char s_buf)
{
          while (((inportb(lsr_port)) & 0x60) != 0x60)
            continue;
          outportb(buf_port, s_buf);
}

void inc_row(void)
{
        if (r == bottom)
         scroll_up();
        else
         r++;
}

void dec_row()
{
        if (r == top)
         scroll_down();
        else
```

```
        r--;
}

char get_data_serialport(void)
{
    char out_b;

    out_b = com_buf[out_p++];      /*    buffer   1024  byte    */
    if (out_p == MAX_BUFF) {
        out_p = 0;
    }
    return(out_b);
}



void process_char(unsigned char c_buf)
{
   int i;

   switch (esc_state)          {
     case 0 : if ((c_buf==0x03)||(c_buf==0x0)) {
                    beep();
              } else
              switch (c_buf) {
                case 0x08 :                    /* BSP  */
                      if (c == 1)
                        beep();
                        else {
                            keep_char();
                            c--;
                                    dsp_str(c,r,0x07,buff_ch);
                            goto_row_col(r, c);
```

```
                    send_char(0x1b);        /* ESC   */
                    send_char(0x50);    /*  P   */
                    }
                break;
        case 0x0d:              /* CR  */
                c = 1;      /*  new line  */
                inc_row();
                goto_row_col(r,c);
                send_char(0x1b);      /* ESC */
                send_char(0x45);      /* "E" */
                break;
        case 0x1b:              /* ESC  */
                esc_state++;
                break;
        default :       /* any other character */
                dsp_char(c_buf);
                c++;
                if (c == right) {
                  c = 1; /*  79  */
                  inc_row();
                  }
                goto_row_col(r, c);
                  } /* end switch (c_buf) */
   break;
case 1:
          switch (c_buf)
          {
     case 0x41:                         /* "A" UP */
                    if (r > 13)
                      r--;
                    break;
        case 0x42:                      /* "B" DOWN */
                    if (r < 22)
```

```
                r++;
                break;
        case 0x43:                      /* "C" RIGHT */
                if (c < 78)
                 c++;
                break;
        case 0x44:                      /* "D" LEFT */
                if (c > 1)
                 c--;
                break;
        case 0x45:                              /* "E" ENTER */
                if (r==22) {
              top =13;bottom=22;
              scroll_up();
              top =1; bottom =10;
                   }else r++;
                   c = 1;
                   break;
        case 0x48:                              /* "H" HOME */
                   c = 1;r= 13;
                   break;
      case 0x4C:                        /* "L" INS */
                   top = r;
                   scroll_down();
                   top = 13;
                   break;
      case 0x50:                  /* "P" BSP */
                   keep_char();
             c--;
                   dsp_str(c,r,0x07,buff_ch);
                   break;
      case 0x46:                  /* "F" FILE */
             gettext(15,8,65,12,buff_ch);
```

```
                        if (Open_file(1)) {
                                send_char(0x0d);
                                Rec_file();
                                if (close(fp))
                                    err_win();
                        }
                        puttext(15,8,65,12,buff_ch);
                    break;
                } /* end switch (c_buf) */
                goto_row_col(r,c);
            esc_state = 0;
                break;
        } /* end switch (esc_state) */
}

void Rec_file(void)
{
  unsigned int size=0;
  char ch;

    while (!dc_report());
          size=((unsigned int)get_data_serialport())<<8;
    while (!dc_report());
          size+=(unsigned int)get_data_serialport() & 0x00ff;

   do {
        if (dc_report()) {
       ch=get_data_serialport();
          if (write(fp,&ch,sizeof(char)) == -1)
              err_win();
          size--;
        }
   } while (size);
```

```
}

void dc_config()
{
 unsigned char l_b[8] = {
                        0x17,       /* 1047, 0417h = 110 baud */
                        0x00,       /*  768, 0300h = 150     */
                        0x80,       /*  384, 0180h = 300     */
                        0xc0,       /*  192, 00c0h = 600     */
                        0x60,       /*   96, 0060h = 1200    */
                        0x30,       /*   48, 0030h = 2400    */
                        0x18,       /*   24, 0018h = 4800    */
                        0x0c,       /*   12, 000ch = 9600    */
                        };
 unsigned char h_b[8] = {
                        0x04,       /* 1047, 0417h = 110 baud */
                        0x03,       /*  768, 0300h = 150     */
                        0x01,       /*  384, 0180h = 300     */
                        0x00,       /*  192, 00c0h = 600     */
                        0x00,       /*   96, 0060h = 1200    */
                        0x00,       /*   48, 0030h = 2400    */
                        0x00,       /*   24, 0018h = 4800    */
                        0x00,       /*   12, 000ch = 9600    */
                        };

        if (com_channel == 0)
          tmp = 0x03f8;
        else
          tmp = 0x02f8;
        disable();
        l_dvs_port = buf_port = tmp++;
        h_dvs_port = ier_port = tmp++;
        iir_port = tmp++;
```

```
    lcr_port = tmp++;
    mcr_port = tmp++;
    lsr_port = tmp++;
    msr_port = tmp;
  old_buf_port  = inportb(buf_port);
    old_ier_port  = inportb(ier_port);
    old_iir_port  = inportb(iir_port);
    old_lcr_port  = inportb(lcr_port);
    old_mcr_port  = inportb(mcr_port);
    old_lsr_port  = inportb(lsr_port);
    old_msr_port  = inportb(msr_port);
    old_ictl_1    = inportb(ictl_1);
    outportb(lcr_port, 0x80);
    outportb(l_dvs_port, l_b[baud]);
    outportb(h_dvs_port, h_b[baud]);
    outportb(lcr_port, wdlen|stpbit|prbit);          /* no parity, 1 1/2 */
    if (com_channel == 0)
       {
        old_comm_isr = getvect(12);
        setvect(12,new_comm_isr);
       }
    else
       {
        old_comm_isr = getvect(11);
        setvect(11,new_comm_isr);
       }
    outportb(mcr_port, 0x0b);            /* set out2, RTS and DTR */
    outportb(ier_port, 0x0d);            /* enable rx, line, modem interrupt */
    tmp = inportb(ictl_1);
    if (com_channel == 0)
        tmp &= 0xef;
    else
        tmp &= 0xf7;
```

```
        outportb(ictl_1, tmp);
        enable();
}

void disable_port()
{
        clrscr();
        outportb(buf_port,old_buf_port);
        outportb(ier_port,old_ier_port);
        outportb(iir_port,old_iir_port);
        outportb(lcr_port,old_lcr_port);
        outportb(mcr_port,old_mcr_port);
        outportb(lsr_port,old_lsr_port);
        outportb(msr_port,old_msr_port);
        outportb(ictl_1,old_ictl_1);
        if (com_channel == 0)
          setvect(12, old_comm_isr);
        else
          setvect(11, old_comm_isr);
}

int dc_report()
{
        if (in_p == out_p)
          return(0);
        else
          return(1);
}

int Open_file(char m)
{
  char * msg[] ={" CAN'T OPEN INPUT FILE  ",
                 " CAN'T OPEN OUTPUT FILE "};
```

```
  fname[0] = '\0';
  win_get_string(fname," -:: Filename ::- ");
  _fmode = O_BINARY;
  if (m) {
    if ((fp = creat(fname,S_IREAD | S_IWRITE)) == -1) {
        dsp_str(30,10,0x87,msg[m]);
        beep();
          GetKey();
          return 0;
    }
    else
          return 1;
  }
  else  {
    if ((fp = _open(fname,O_RDONLY)) == -1) {
       dsp_str(30,10,0x87,msg[m]);
          beep();
          GetKey();
          return 0;
    }
    else
          return 1;
  }
}

unsigned int filesize()
{
  unsigned int f;

  lseek(fp,0,SEEK_END);
  f = (unsigned int) tell(fp);
  lseek(fp,0,SEEK_SET);
```

```
  return (f);
}

void Send_file(void)
{
  char ch;
    do {
        if (read(fp,&ch,sizeof(char)) == -1)
          err_win();
      send_char(ch);
    } while (!eof(fp) && (!kbhit()));
}

void baud_rate_nui(void)
{
 char x,y;
 int  key_buff;
  cursor_off();
  _frame(3,1,75,18,2,0x34);
  _frame(3,13,75,18,2,0x34);
  dsp_str(24,1,0x34," COMMUNICATION  CONFIGURATION ");

/************************
*      show baud rate
/************************/
  dsp_str(6,3,0x03,"(1) BAUD RATE ");
  dsp_str(6,5,0x03," [1] ");  dsp_str(11,5,0x03,bdrate[0]);
  dsp_str(6,6,0x03," [2] ");  dsp_str(11,6,0x03,bdrate[1]);
  dsp_str(6,7,0x03," [3] ");  dsp_str(11,7,0x03,bdrate[2]);
  dsp_str(6,8,0x03," [4] ");  dsp_str(11,8,0x03,bdrate[3]);
  dsp_str(6,9,0x03," [5] ");  dsp_str(11,9,0x03,bdrate[4]);
  dsp_str(6,10,0x03," [6] "); dsp_str(11,10,0x03,bdrate[5]);
  dsp_str(6,11,0x03," [7] "); dsp_str(11,11,0x03,bdrate[6]);
```

```
  dsp_str(6,12,0x03," [8] "); dsp_str(11,12,0x03,bdrate[7]);

/************************
*   show data bits
/************************/
  dsp_str(20,3,0x03," (2) DATA BIT ");
  dsp_str(20,5,0x03," [1] ");  dsp_str(25,5,0x03,data[0]);
  dsp_str(20,6,0x03," [2] ");  dsp_str(25,6,0x03,data[1]);

/************************
*   show parity bit
/************************/
  dsp_str(34,3,0x03," (3) PARITY");
  dsp_str(34,5,0x03," [1] ");  dsp_str(39,5,0x03,parity[0]);
  dsp_str(34,6,0x03," [2] "); dsp_str(39,6,0x03,parity[1]);
  dsp_str(34,7,0x03," [3] "); dsp_str(39,7,0x03,parity[2]);

/************************
*   show stop bit
/************************/
  dsp_str(48,3,0x03," (4) STOP");
  dsp_str(48,5,0x03," [1] ");  dsp_str(53,5,0x03,stop[0]);
  dsp_str(48,6,0x03," [2] ");  dsp_str(53,6,0x03,stop[1]);

/************************
*   show  com port
/************************/
  dsp_str(62,3,0x03," (5) PORT");
  dsp_str(62,5,0x03," [1] ");  dsp_str(67,5,0x03,comm_ch[0]);
  dsp_str(62,6,0x03," [2] ");  dsp_str(67,6,0x03,comm_ch[1]);

  dsp_str(55,18,0x34,"ч ESC to Cancle Ж");
```

```
/**********************
*     defualt config
/**********************/
  dsp_str(29,14,0x34," CURRENT SETTING ");

    do {
      dsp_str(ox,oy,0x34,bdrate[baud_rat]);
      key_buff = GetKey();
      dsp_str(ox,oy,0x03,bdrate[baud_rat]);
      switch (key_buff) {
          case UP   : if (oy < 6 ) {
                              oy =12;baud_rat = 7;
                              } else { oy--;baud_rat--; }
                    break;
          case DOWN : if (oy > 11) {
                              oy =5; baud_rat = 0;
                              }else { oy++; baud_rat++; }
                     break;
          case ENTER: dsp_str(11,16,0x03,bdrate[baud_rat]);
                    break;
          case ESC  :return;
          }  /* END OF SWITCH */
      }while (key_buff!=ENTER);
/** RECEIVE DATA BIT **/
    do {
      dsp_str(ox1,oy1,0x34,data[word_length]);
      key_buff = GetKey();
      dsp_str(ox1,oy1,0x03,data[word_length]);
      switch (key_buff) {
          case UP   :
          case DOWN : if (oy1 == 6 ) {
                              oy1 =5;word_length = 0;
                              } else { oy1=6;word_length=1; }
```

```
                    break;
            case ENTER: dsp_str(25,16,0x03,data[word_length]);
                    break;
            case ESC  :return;
            }  /* END OF SWITCH */
        }while (key_buff!=ENTER);
/** RECEIVE PARITY **/
    do {
        dsp_str(ox2,oy2,0x34,parity[parity_bit]);
        key_buff = GetKey();
        dsp_str(ox2,oy2,0x03,parity[parity_bit]);
        switch (key_buff) {
            case UP   : if (oy2 < 6 ) {
                                oy2=7;parity_bit=2;
                                }else { oy2--;parity_bit--;}
                    break;
            case DOWN : if (oy2 > 6 ) {
                                oy2 =5;parity_bit = 0;
                                } else { oy2++;parity_bit++; }
                    break;
            case ENTER: dsp_str(39,16,0x03,parity[parity_bit]);
                    break;
            case ESC  :return;
            }  /* END OF SWITCH */
        }while (key_buff!=ENTER);

/** RECEIVE STOP BIT **/
    do {
        dsp_str(ox3,oy3,0x34,stop[stop_bit]);
        key_buff = GetKey();
        dsp_str(ox3,oy3,0x03,stop[stop_bit]);
        switch (key_buff) {
            case UP   :
```

```
            case DOWN : if (oy3 == 6 ) {
                                    oy3 =5;stop_bit = 0;
                                    } else { oy3=6;stop_bit=1; }
                        break;
            case ENTER: dsp_str(53,16,0x03,stop[stop_bit]);
                        break;
            case ESC  :return;
            } /* END OF SWITCH */
        }while (key_buff!=ENTER);

/** RECEIVE PORT **/
    do {
        dsp_str(ox4,oy4,0x34,comm_ch[com_channel]);
        key_buff = GetKey();
        dsp_str(ox4,oy4,0x03,comm_ch[com_channel]);
        switch (key_buff) {
            case UP   :
            case DOWN : if (oy4 == 6 ) {
                                    oy4 =5;com_channel = 0;
                                    } else { oy4=6;com_channel=1; }
                        break;
            case ENTER: dsp_str(67,16,0x03,comm_ch[com_channel]);
                        break;
            case ESC  :return;
            } /* END OF SWITCH */
        }while (key_buff!=ENTER);
}

void  tllink(void)
{
    unsigned int F_size;
    int key_buf;
    int ocurx,ocury;
```

```
ocurx = wherex(); ocury = wherey();
save_window(0,0,79,24,wp);
r = c = 1;
in_p = out_p = 0;
attribute = 7;
baud = 7;                              /* 9600 */
com_channel = 0;                       /* COM1          */
get_crtseg();
clrscr();
cursor_off();
dc_config();
cursor_on();
clrtermscr();
_frame(0,0,79,11,1,0x03);
_frame(0,12,79,23,0,0x03);
terminate = false;
goto_row_col(r, c);
while (!terminate)          {
        if (dc_report()) {
        r1 = r; c1 = c;
        r= row;c= col;
           top = 13;bottom = 22;
           goto_row_col(r,c);
           process_char(get_data_serialport());
           row =r;col = c;
           r = r1; c = c1;
           goto_row_col(r,c);
        top = 1;bottom = 10;
         }
         if (bioskey(1)) {
           key_buf = bioskey(0);
           switch (key_buf) {
               case 0x3B00:                    /* F1   */
```

```
                gettext(15,8,65,12,buff_ch);
                if (Open_file(0)) {
                  send_char(0x1b);    /* ESC   */
                  send_char(0x46);    /*  "F"  */
                      while (!kbhit())
                        if (dc_report())
                          if (get_data_serialport() == 0x0d) {
                                F_size=filesize();
                                send_char((char) (F_size>>8));
                                send_char((char) (F_size & 0xFF));
                                dsp_str(30,10,0x47,"  SEND FILE NOW  ");
                                Send_file();
                                dsp_str(30,10,0x47,"   END OF FILE   ");
                          }
                }
                if (fp != -1)
                  if (close(fp))
                       err_win();
    puttext(15,8,65,12,buff_ch);
                break;
case 0x4400:                    /* F10  */
    gettext(4,2,76,20,buff_ch);
    Screen(4,2,76,20,0x20,0x03);
                baud_rate_nui();    /* set config  */
    config_nui();
    dc_config();
    puttext(4,2,76,20,buff_ch);
    clrmscr();
    cursor_on();
    break;
case UP    :
    if ( r > top) r--;
                send_char(0x1b);          /* ESC   */
```

```
            send_char(0x41);    /* "A"          */
            break;
case DOWN  :
    if ( r < bottom) r++;
            send_char(0x1b);          /* ESC   */
            send_char(0x42);          /* "B"   */
            break;
case LEFT  :
    if ( c >1) c--;
            send_char(0x1b);          /* ESC   */
            send_char(0x44);          /* "D"   */
            break;
case RIGHT :
    if ( c < 78 ) c++;
            send_char(0x1b);          /* ESC   */
            send_char(0x43);          /* "C"   */
            break;
case 0x4700:                  /* home */
    c = r = 1;
    send_char(0x1b);          /* ESC   */
            send_char(0x48);          /* "H"   */
            break;
case 0x5200:                          /* ins   */
    top =r;
    scroll_down();
    top = 1;
            send_char(0x1b);          /* ESC   */
            send_char(0x4c);          /* "L"   */
            break;
case 0x2d00:                  /*  Alt-X  */
    terminate = true;
            break;
default    :
```

```
                    process_char((char) key_buf);
                        if ((key_buf!=0xE08)&&(key_buf!=0x1c0d))
                    send_char((char) key_buf);
                        break;
                }
              goto_row_col(r,c);
            }
            }
  disable_port();
  restore_window(0,0,79,24,wp);
  _gotoxy(ocurx-1,ocury-1);
}
```

```
/*****************************************
* Program  HUFF.C  for  Compress Data  *
*****************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <dos.h>
#include <io.h>
#include "c:\project\update\main.h"
#include "c:\project\update\bitio.h"
#include "c:\project\update\errhand.h"

typedef struct tree_node {
        unsigned  int  count ;
        unsigned  int  saved_count;
        int  child_0;
        int  child_1;
}NODE;

typedef struct code {
        unsigned int code;
        int code_bits;
}CODE;

#define END_OF_STREAM 256

void count_bytes(int input, unsigned long *long_counts );
void scale_counts(unsigned long *long_counts, NODE *nodes);
int build_tree(NODE *nodes);
void convert_tree_to_code(NODE *nodes,
                          CODE *codes,
```

```
                                unsigned int code_so_far,
                                int bits,
                                int node);
void output_counts(BIT_FILE *output, NODE *nodes);
void input_counts(BIT_FILE *input, NODE *nodes);
void print_model(NODE *nodes, CODE *codes);
void compress_data(int input, BIT_FILE *output, CODE *codes );
void expand_data(BIT_FILE *input,int output,NODE *nodes,
                        int root_node );
void print_char(int c);

void CompressFile(int input,BIT_FILE * output);
void ExpandFile(BIT_FILE *input,int output);

char * CompressionName = "static order 0 model with Huffman coding";
char * Usage = "infile outfile [-d]\n\n\Specifying -d will dump the modeling data\n";

void CompressFile(int input,BIT_FILE * output)
{
  unsigned long * counts;
  NODE * nodes;
  CODE * codes;
  int root_nodes;

  if ((counts = (unsigned long *) calloc(256,sizeof(unsigned long))) == NULL) {
      fatal_error("Heap memory not enough !");
      return;
  }
  if ((nodes = (NODE *)  calloc(514,sizeof(NODE))) == NULL) {
      fatal_error("Heap memory not enough !");
      return;
  }
  if ((codes = (CODE *) calloc(257,sizeof(CODE))) == NULL) {
```

```
        fatal_error("Heap memory not enough !");
        return;
    }
    count_bytes(input,counts);
    scale_counts(counts,nodes);
    output_counts(output,nodes);
    root_nodes = build_tree(nodes);
    convert_tree_to_code(nodes,codes,0,0,root_nodes);
    compress_data(input,output,codes);
    free((unsigned long *) counts);
    free((NODE *) nodes);
    free((CODE *) codes);
}

void ExpandFile(BIT_FILE *input,int output)
{
    NODE * nodes;
    int root_node;

    if ((nodes = (NODE *) calloc(514,sizeof(NODE))) == NULL) {
        fatal_error( "Error allocting nodes array\n" );
        return;
    }
    input_counts(input,nodes);
    root_node = build_tree(nodes);
    expand_data(input,output,nodes,root_node);
    free(nodes);
}

void output_counts(BIT_FILE *output, NODE *nodes)
{
    int first;
    int last;
```

```
int next;
int i;

first = 0;
while (first < 255 && nodes[first].count == 0)
        first++;
for ( ; first < 256 ; first = next) {
    last = first + 1;
    for ( ; ; ) {
        for ( ; last < 256 ; last++)
          if (nodes[last].count == 0)
                break;
        last--;
        for ( next = last + 1; next < 256 ; next++)
          if (nodes[ next ].count != 0)
                break;
        if (next > 255)
           break;
        if ((next - last) > 3)
           break;
        last = next;
    };

   if (write(output->f_handle,&first,1) == -1) {
        fatal_error("Failure writing byte counts\n");
        return;
   }
   if (write(output->f_handle,&last,1) == -1) {
        fatal_error("Failure writing byte counts\n" );
        return;
   }
   for ( i = first ; i <= last ; i++) {
        if (write(output->f_handle,&nodes[i].count,1) == -1) {
```

```
                fatal_error("Failure writing byte counts\n" );
                return;
            }
        }
    }
    {
        int p = 0;

        if (write(output->f_handle,&p,1) == -1) {
            fatal_error("Failure writing byte counts\n");
            return;
        }
    }
}

void input_counts(BIT_FILE * input,NODE * nodes)
{
    unsigned char first;
    unsigned char last;
    int i;
    unsigned char c;

    for (i = 0 ; i < 256 ; i++)
        nodes[i].count = 0;
    if ((read(input->f_handle,&first,1)) == -1) {
        fatal_error("Error reading byte counts\n");
        return;
    }
    if ((read(input->f_handle,&last,1)) == -1) {
        fatal_error("Error reading byte counts\n");
        return;
    }
    for ( ; ; ) {
```

```
    for ( i = first ; i <= last ; i++ )
        if  ((read(input->f_handle,&c,1)) == -1) {


            fatal_error( "Error reading byte counts\n" );
            return;
        }
        else
            nodes[i].count = (unsigned int) c;
    if ((read(input->f_handle,&first,1)) == -1) {
        fatal_error( "Error reading byte counts\n" );
        return;
    }
    if (first == 0)
        break;
    if ((read(input->f_handle,&last,1)) == -1) {
        fatal_error( "Error reading byte counts\n" );
        return;
    }
  }
  nodes[END_OF_STREAM].count = 1;
}

#ifndef SEEK_SET
#define SEEK_SET 0
#endif

void count_bytes(int input,unsigned long * counts)
{
    long input_marker;
    int c = 0;

    input_marker = tell(input);
    do {
      if ((read(input,&c,1)) == -1) {
```

```
            fatal_error("Failure from reading file");
            return;
        }
        else
            counts[c]++;
    } while (!eof(input));
    lseek(input,input_marker,SEEK_SET);
}

void scale_counts(unsigned long * counts,NODE * nodes)
{
  unsigned long max_count;
  int i;

  max_count = 0;
  for (i = 0 ; i < 256 ; i++)
          if (counts[i] > max_count)
              max_count = counts[i];
  if (max_count == 0) {
      counts[0] = 1;
      max_count = 1;
  }
  max_count = max_count / 255;
  max_count = max_count + 1;
  for (i = 0 ; i < 256 ; i++)   {
     nodes[i].count = (unsigned int) (counts[i] / max_count);
     if (nodes[i].count == 0 && counts[i] != 0)
             nodes[i].count = 1;
  }
  nodes[END_OF_STREAM].count = 1;
}

int build_tree(NODE *nodes)
```

```
{
    int next_free;
    int i;
    int min_1;
    int min_2;

    nodes[513].count = 0xffff;
    for (next_free = END_OF_STREAM + 1;  ; next_free++)   {
        min_1 = 513;
        min_2 = 513;
        for ( i = 0 ; i < next_free ; i++ )
            if (nodes[i].count != 0)
              {
                if (nodes[i].count < nodes[min_1].count)
                  {
                      min_2 = min_1;
                      min_1 = i;
                  } else if (nodes[i].count < nodes[min_2].count )
                          min_2 = i;
              }
            if (min_2 == 513)
               break;
            nodes[next_free].count = nodes[min_1].count + nodes[min_2].count;
            nodes[min_1].saved_count = nodes[ min_1 ].count;
            nodes[min_1].count = 0;
            nodes[min_2].saved_count = nodes[ min_2 ].count;
            nodes[min_2].count = 0;
            nodes[next_free].child_0 = min_1;
            nodes[next_free].child_1 = min_2;
     }
    next_free--;
    nodes[next_free].saved_count = nodes[next_free].count;
    return (next_free);
```

```
}

void convert_tree_to_code( NODE *nodes, CODE *codes,
                           unsigned int code_so_far, int bits, int node )
{
  if ( node <= END_OF_STREAM )
    {
      codes[ node ].code = code_so_far;
      codes[ node ].code_bits = bits;
      return;
    }
  code_so_far <<= 1;
  bits++;
  convert_tree_to_code( nodes, codes, code_so_far, bits,
                        nodes[ node ].child_0 );
  convert_tree_to_code( nodes, codes, code_so_far | 1,
                        bits, nodes[ node ].child_1);
}

void print_char( int c )
{
  if ( c >= 0x20 && c < 127 )
    printf( "'%c'", c );
  else
    printf( "%3d", c );

}

void compress_data(int input,BIT_FILE *output,CODE *codes)
{
  int c = 0;

  do {
```

```
    if (read(input,&c,1) == -1) {
        fatal_error("Failure in reading file");
        return;
    }
    OutputBits(output,(unsigned long) codes[c].code,
                codes[c].code_bits );
  } while (!eof(input));

  OutputBits (output,(unsigned long) codes[END_OF_STREAM].code,
                codes[END_OF_STREAM].code_bits );
}

void expand_data(BIT_FILE *input,int output, NODE *nodes, int root_node)
{
  int node;

  for ( ; ; )  {
    node = root_node;
    do {
        if (InputBit(input))
            node = nodes[node].child_1;
        else
            node = nodes[node].child_0;
    } while (node > END_OF_STREAM);

    if (node == END_OF_STREAM)
        break;
    if ((write(output,&node,1)) == -1) {
        fatal_error("Error trying to write byte to output");
        return;
    }
  }
}
```

```
/*************************************
* Program MAIN-C.C  for Compression  *
*************************************/
#define WIN_PROTOTYPE
#define _STRUCT_WINDOW_

#include "window.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <dos.h>
#include <io.h>
#include "c:\project\update\main.h"
#include "c:\project\update\bitio.h"
#include "c:\project\update\errhand.h"

#define  PATH_FILE 50

long file_size(char *name);
void compress(char * n_fname,char * c_fname);

void  get_file_compress(void)
{
   char org_file[PATH_FILE];
   char cps_file[PATH_FILE];
   int ox,oy;

   ox = wherex(); oy = wherey();
   org_file[0] = '\0';
   cps_file[0] = '\0';
   win_get_string(org_file," -:: Originated Filename ::- ");
   win_get_string(cps_file," -:: Compressed Filename ::- ");
```

```
   compress(org_file,cps_file);
   _gotoxy(ox-1,oy-1);
}

void compress(char * n_fname,char * c_fname)
{
  BIT_FILE *output;
  int input;

  setbuf(stdout,NULL);
  _fmode = O_BINARY;
  if ((input = _open(n_fname,O_RDONLY)) == -1) {
      fatal_error("Can't open file %s",n_fname);
      return;
  }
  output = (BIT_FILE *) OpenOutputBitFile(c_fname);
  if (output == NULL) {
       fatal_error("Heap memory not enough !");
       return;
  }
  CompressFile(input,output);
  CloseOutputBitFile(output);
  if (close(input)) {
    fatal_error("Can't close compress file");
    return;
  }
}

#ifndef  SEEK_END
#define  SEEK_END 2
#endif

long file_size(char *name)
```

```
{
  long eof_ftell;
  int  file;

  if ((file = _open(name,O_RDONLY)) == -1) {
     fatal_error("Can't open file ");
     return(0L);
  }
  if (file == -1)
       return(0L);
  lseek(file,0L,SEEK_END);
  eof_ftell = tell(file);
  close(file);
  return(eof_ftell);
}
```

```
/***************************************
*  Program  MAIN-E.C  for  Encode      *
***************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <bios.h>
#include <fcntl.h>
#include <io.h>
#include <stat.h>
#include "c:\project\update\main.h"
#include "c:\project\update\bitio.h"
#include "c:\project\update\errhand.h"

#define PATH_FILE 50


void decompress(char *n_fname,char *c_fname);
extern void ExpandFile(BIT_FILE * input,int output);

void get_file_decompress(void)
{
  char org_file[PATH_FILE];
  char cps_file[PATH_FILE];
  int ox,oy;

  ox = wherex(); oy = wherey();
  org_file[0] = '\0';
  cps_file[0] = '\0';

  win_get_string(cps_file," -:: Compressed File ::- ");
  win_get_string(org_file," -:: Originated File ::- ");
  decompress(org_file,cps_file);
```

```
  _gotoxy(ox-1,oy-1);
}

void decompress(char *n_fname,char *c_fname)
{
  int output;
  BIT_FILE *input;

  setbuf(stdout,NULL);
  input = OpenInputBitFile(c_fname);
  if (input == NULL) {
     fatal_error("Heap memory not enough !");
     return;
  }
  _fmode = O_BINARY;
  if ((output = creat(n_fname,S_IREAD | S_IWRITE)) == -1) {
       fatal_error("Failure in opening %s for output\n",n_fname);
       return;
  }
  ExpandFile(input,output);
  CloseInputBitFile(input);
  close(output);
}
```

# บรรณานุกรม


1. Larry, E. Jordan & Bruce Churchill , in Communications and Networking for the IBM PC . Prentice Hall, 1983.
2. Ken Kreehmer, V.22bit : The Modem Specification that's got trouble , Data Communication., McGraw - Hill Magazine , Apil , 1985.
3. Harvey A.Freeman and Kenneth J. Thurber tutorial Micro Computer Networks IEEE Computer society, IEEE Catalog Number EH190-9, 1981.
4. Herbert Schildt, C Power User's Guide , Osborne McGraw - Hill , Berkeley , California, USA , 1988.
5. Dr. James T.Smith, Advance turbo C , Intertext Publications McGraw - Hill book Company , 1989
6. Mark Nelson, The Data Compression Book, pp. 29 - 80, M&T Publishing, Inc., 1992.
7. เรวดี ลิมปโชติกุล ลำดับชั้นของวิธีการจัดข้อมูล วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิทยาศาสตร์ บัณฑิตวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย 2534