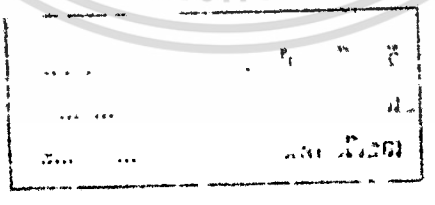
การส่งข้อมูลภาพผ่านเครือข่ายท้องถิ่น

TRANSMISSION PICTURE DATA IN LOCAL AREA NETWORK

โดย

นาย คงพันธ์ ฉมารัตน์ 34101042

นาย ปรัชญา ปัณฑุยากร 34104211



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์

สาขา วิศวกรรมอิเล็กทรอนิกส์

ภาควิชา อิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์เรื่อง การส่งข้อมูลภาพผ่านเครือข่ายท้องถิ่น

TRANSMISSION PICTURE DATA IN LOCAL AREA NETWORK

โดย นาย คงพันธ์ ฉมารัตน์ 34101042

นาย ปรัชญา ปัณฑุยากร 34104211

ชื่ออาจารย์ที่ปรึกษา รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์ปีการศึกษา 2537

ภาค วิชาอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การส่งข้อมูลภาพผ่านเครือข่ายท้องถิ่น**

TRANSMISSION PICTURE DATA IN LOCAL AREA NETWORK

ผู้จัดทำ นาย คงพันธ์ ฉมารัตน์ รหัสประจำตัว 34101042

นาย ปรัชญา ปัณฑุยากร รหัสประจำตัว 34104211

..... อาจารย์ที่ปรึกษา

(รศ.ดร. มนัส สังวรศิลป์)

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการศึกษาการส่งข้อมูลผ่านระบบเครือข่ายท้องถิ่น (Local Area Network) ด้วยความเร็วสูง ซึ่งเป็นการนำโปรโตคอล IPX (Internetwork Packet Exchange Protocol) มาประยุกต์ใช้ในการเขียนโปรแกรม โดยที่เป็นการส่งข้อมูลในแบบจุดต่อจุด (Peer-to-peer) ซึ่งมีความเร็วสูงในการส่ง โดยข้อมูลที่ใช้ในการศึกษาจะเป็นข้อมูลที่ได้จาก TARGA Card ซึ่งมีความคมชัดของภาพ และจำนวนสีมากกว่า VGA Card

ปัจจัยที่สำคัญต่อความเร็วในการส่งข้อมูลภาพ คือ แบบของระบบเครือข่าย และขนาดของข้อมูลภาพ การศึกษาได้ใช้ระบบเครือข่ายท้องถิ่นแบบ Ethernet ตามมาตราฐาน IEEE 802.3 ที่มีการส่งข้อมูลแบบเบสแบนด์ ที่มีความเร็วในการส่งข้อมูล 10 เมกะบิตต่อวินาที ซึ่งเป็นการยากที่จะเปลี่ยนความเร็วของระบบเครือข่าย ดังนั้นขนาดของข้อมูลเป็นสิ่งที่ต้องพิจารณามากกว่า เนื่องจากขนาดของข้อมูลภาพมีขนาดใหญ่ จึงได้มีการนำวิธีการลดขนาดข้อมูลมาช่วย เพื่อให้ข้อมูลที่ทำการส่งมีขนาดเล็กลง ทำให้ความเร็วในการส่งสูงขึ้น

ABSTRACT

This thesis aims to study on data transmission system through Local Area Network,by using Internetwork Packet Exchange as a programmer's tool. The peer-to-peer model is applied to this extreme high speed transmission. The study selected data from Targa Card,which provides not only superior vision but wider range of colours in comparing to VGA Card.

The most significant factors influenced on this speed transmission are the model of network and data from the Targa Card. The study particularly used the Ethernet Local Area Network under IEEE 802.3 standard. The network is able to transmit data with 10 Megabits per second. This high speed limit can not apply for extra large size of data,therefore, compression size of data is crucial in accelerating speed of transmission.

คำนำ

ปัจจุบันนี้การติดต่อสื่อสารข้อมูลเป็นสิ่งที่จำเป็นอย่างยิ่ง ในหน่วยงานต่าง ๆ ได้มีการนำคอมพิวเตอร์มาช่วยอำนวยความสะดวกในการทำงาน และด้วยความก้าวหน้าทางเทคโนโลยี ได้มีการเชื่อมคอมพิวเตอร์เข้าไว้เป็นระบบเครือข่าย (Network) ซึ่งทำให้ประสิทธิภาพในการสื่อสารข้อมูลดียิ่งขึ้น เราสามารถติดต่อส่งข้อมูลถึงกันโดยใช้เครื่องคอมพิวเตอร์ส่วนตัว (PC) ไม่ว่าจะเป็นข้อมูลภาพ ข้อมูลเสียง และข้อมูลอื่น ๆ

วิทยานิพนธ์เล่มนี้เป็นการศึกษาการส่งข้อมูลภาพผ่านเครือข่ายท้องถิ่น (Local Area Network : LAN) โดยเน้นที่ความเร็วสูง ทำการส่งข้อมูลโดยใช้โปรโตคอล IPX (Internetwork Packet Exchange Protocol) และเพื่อให้การส่งใช้เวลาน้อยที่สุด ได้มีการนำวิธีการลดขนาดข้อมูลมาทำการลดข้อมูลก่อนการส่ง ข้อมูลภาพที่ใช้ในการศึกษาในที่นี้ ได้มาจาก TARGA Card ซึ่งมีความคมชัด และสีมากกว่า VGA Card

ในวิทยานิพนธ์เล่มนี้ได้มีการแสดงให้เห็นถึงการทำงานโดยละเอียดของการทดลอง โดยมีการแสดงภาพโดยใช้ภาพถ่ายจากหน้าจอภาพ เพื่อใช้ในการอธิบาย เนื่องจากว่าวิทยานิพนธ์เรื่องนี้ยังไม่ได้เคยมีการศึกษามาก่อน และการหาข้อมูลทำได้ค่อนข้างลำบาก ดังนั้นถ้าหากว่าวิทยานิพนธ์เล่มนี้มีข้อผิดพลาดประการใด ทางผู้ศึกษาขออภัยมา ณ ที่นี้ด้วย

ด้วยความเคารพอย่างสูง
นาย คงพันธ์ ฉมารัตน์
นาย ปรัชญา ปัณฑุยากร

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมาในการทำการศึกษา

ปัจจุบันวิวัฒนาการทางเทคโนโลยี ได้ก้าวไปอย่างรวดเร็ว โลกในปัจจุบันนี้อาจจะกล่าวได้ว่าเป็นโลกของข้อมูล ในการแข่งขันกันทางธุรกิจ ผู้ใดสามารถได้รับข่าวสารที่ถูกต้อง รวดเร็ว ก็จะเป็นผู้ที่ได้เปรียบ และในทางการแพทย์ ความรวดเร็วในการติดต่อสื่อสารที่รวดเร็วยิ่งมีความสำคัญมาก เพราะแต่ล่ะนาที อาจจะหมายถึงชีวิตคน

ดังนั้นวิทยานิพนธ์เล่มนี้จึงได้จัดทำขึ้นมา โดยต้องการศึกษาการส่งข้อมูลผ่านเครือข่ายท้องถิ่น (Local Area Network) ที่รวดเร็ว และมีความถูกต้อง ซึ่งข้อมูลที่ใช้ในการศึกษานี้จะเป็นข้อมูลภาพ ซึ่งข้อมูลภาพจะมีขนาดใหญ่ จึงสามารถใช้ในการศึกษาได้เป็นอย่างดี

1.2 วัตถุประสงค์ของการศึกษา

วิทยานิพนธ์เล่มนี้มีวัตถุประสงค์ดังต่อไปนี้

1. ศึกษาการส่งข้อมูลที่รวดเร็ว และถูกต้อง
2. ศึกษาข้อมูลภาพที่ได้จาก TARGA Card ซึ่งเป็นการ์ดที่มีความคมชัดและสีสันมาก
3. ประยุกต์การนำวิธีการนำภาพขนาดใหญ่ มาทำการส่งโดยแยกออกเป็นข้อมูลย่อย (Packet) แล้วทำการส่ง และ ทำการรวบรวมให้ได้ข้อมูลภาพที่เหมือนต้นฉบับทุกประการ
4. ศึกษาวิธีการใช้งาน TARGA Card
5. ศึกษาวิธีเขียนโปรแกรมโดยใช้โปรโตคอล IPX (Internetwork Packet Exchange Protocol)
6. ศึกษาวิธีการลดขนาดข้อมูล
7. นำโปรแกรมที่ได้จากการศึกษาและจัดทำ นำไปประยุกต์ใช้งาน

1.3 วิธีการศึกษา

โครงงานนี้ได้มีการศึกษาทางด้านซอฟแวร์เพียงอย่างเดียว โดยแบ่งเป็นดังนี้

1. ศึกษาการเขียนโปรแกรมโดยใช้โปรโตคอล IPX
2. ศึกษาการเขียนโปรแกรมควบคุม TARGA Card
3. ศึกษาการเขียนโปรแกรมลดขนาดข้อมูล
4. นำโปรแกรมที่เขียนทั้งสองอย่างมารวมกัน โดยเขียนเป็น รายการ (Menu) เพื่อให้ง่ายในการใช้งาน

1.4 ขอบเขตของวิทยานิพนธ์

1. สามารถส่งข้อมูลที่ได้จาก TARGA Card ผ่านเครือข่ายท้องถิ่น (Local Area Network : LAN)

โดยที่การส่งได้รับภาพที่เหมือนต้นฉบับทุกประการ

2. เนื่องจากว่า TARAGA Card ได้มีฟังก์ชั่นที่อำนวยความสะดวกในการลดขนาดข้อมูล และมีอัตราการลดขนาดข้อมูล ที่มากกว่าที่ได้ศึกษาโดยวิธีของ Huffman โดยได้มีการเปรียบเทียบได้ผลดังนี้

โดยใช้ฟังก์ชั่นของ TARGA Card สามารถลดข้อมูลได้ประมาณ 40 %

โดยใช้วิธีของ Huffman สามารถลดข้อมูลได้ประมาณ 18 %

และความเร็วในการลดขนาดข้อมูลใช้เวลาใกล้เคียงกัน ดังนั้นผู้ศึกษาจึงได้ใช้ฟังก์ชั่นของ TARGA Card ในการลดขนาดข้อมูล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

เพื่อประโยชน์ในทางการแพทย์ ในการส่งภาพที่ได้จากการถ่ายภาพของผู้ป่วยเพื่อประกอบในการวินิจฉัยโรค และสามารถส่งไปยังแพทย์ท่านอื่นเพื่อใช้ในการวินิจฉัยเช่นกัน

เพื่อประโยชน์ทางธุรกิจ ในการส่งภาพของสินค้าในการเจรจาทางการค้า

เพื่อประโยชน์ในการวางแนวทางให้รุ่นน้องในการศึกษา และพัฒนาทางด้านนี้ต่อไป

บทที่ 2
OSI โมเดล

2.1 ความเป็นมาของ OSI

ในช่วงปลายปี ค.ศ. 1981 IEEE และสามาคม European Computer Manufactures Accociation (ECMA) ได้ร่วมกันกำหนดมาตราฐานการสื่อสารคอมพิวเตอร์ระบบเปิด (Open System Interconnection หรือ OSI) โดยอ้างอิงโมเดลขององค์การระหว่างประเทศเพื่อทำให้เป็นมาตรฐาน (International Standards Organization หรือ ISO) ซึ่งมีอยู่ทั้งหมด 7 ชั้น (Layers) ดังตารางที่ 2.1

ตารางที่ 2.1.1 โมเดล OSI และมาตราฐาน IEEE 802

ชั้น	โมเดล OSI	มาตราฐาน IEEE 802
7	Application	N/A
6	Presentation control	N/A
5	Session control	N/A
4	Transport end-to-end control	N/A
3	Network control	N/A
2	Logical link control	802.3, 802.4
	medium access control (MAC)	802.5
1	Physical control	802.3, 802.4
		802.5

ชั้นที่ 1 (Physical control) และชั้นที่ 2 (Logical link control control และ medium access control) ก็เป็นมาตราฐานของ IEEE ด้วยเช่นกัน ชั้นที่ 1 กำหนดชนิดของสายส่งข้อมูลที่ใช้ซึ่งมีผลต่อราคาในการติดตั้งและการขยายระบบต่อไปในอนาคต ชั้นที่ 1 ถูกนำไปใช้ในฮาร์ดแวร์ ส่วนชั้นที่ 2 จะนำไปร่วมกันทั้งฮาร์ดแวร์และซอฟแวร์ ซึ่งขึ้นอยู่กับอุปกรณ์หรือการ์ดการเชื่อมโยงเครือข่าย (Network interface card : NIC) ของผู้ผลิตแต่ล่ะราย

ส่วนอีก 5 ชั้นที่เหลือนั้น IEEE ไม่ได้กำหนดเป็นมาตราฐานไว้ แม้ว่ามาตราฐานบางอย่างของ ISO จะมีอยู่แล้วในระบบเครือข่าย เช่นมาตราฐาน Manufacturing Automation Protocol (MAP) เป็นต้น ชั้นเหล่านี้จะไปใช้ในซอฟต์แวร์ทั้งหมด

มาตราฐาน 4 อัน ของ IEEE จะอยู่ที่ 2 ชั้นล่างสุดของ OSI ในขณะนี้ขอบเขตของ ISO ที่ชั้นที่ 1 และ 2 จะเหมือนกับมาตราฐานของ IEEE แต่ถ้าการกำหนดของ ISO เสร็จสิ้นแล้วก็ยังไม่รู้ว่า ชั้นที่ 2 และ 3 จะเหมือนกับมาตราฐานของ IEEE มีดังนี้

802.2 Data or logical link control

802.3 Carrier Sense Multiple Access with Collision Detect (CSMA/CD) bus LANs เช่น Ethernet และ StarLAN

802.4 Token-passing bus LANs เช่ย MAP,ARCnet

802.5 Token-passing ring LANs เช่น ระบบเครือข่าย IBM Token-Ring

มาตราฐาน 802.6 พัฒนาขึ้นสำหรับเครือข่ายแบบ MAN โดยใช้เทคโนโลยีของ CATV และ 802.7 ครอบคลุมถึงระบบเครือข่ายแบบบรอดแบนด์(Broad band) รูปแบบของระบบเครือข่ายที่กว้างกว่านี้ สามารถมองเห็นเหมือนสะพานสื่อสาร(Bridge) ที่ต่อออกจาก LAN ได้

ในเอกสารของ IEEE 802.1 ได้บรรยายถึงความสัมพันธ์ระหว่างมาตราฐานต่างๆ รวมทั้งตำแหน่งในโมเดล OSI ของ ISO ด้วย นอกจากนี้เอกสารฉบับนี้ยังได้รวมถึงความ สัมพันธ์ของมาตราฐาน 802 กับโปรโตคอล(Protocol) ในชั้นที่สูงขึ้นไปเป็นการจัดการระบบเครือข่ายและการติดต่อกันในระหว่างระบบเครือข่าย

ส่วนบนของชั้นที่ 2 ของ ISO จะสัมพันธ์กับมาตรฐาน 802.2 ของ IEEE logical link control เป็นได้ ทั้งแบบ connection-oriented (class II) การบริการแบบ connectionless จะใช้เมื่อชั้นสูงในชั้น logical link ที่ไม่ต้องการตรวจสอบการส่ง ส่วนแบบ connection-oriented จะมีการบริการคล้ายกับโปรโตคอล Syschronous data link control (SDLC) หรือ high level data link control (HDLC) รวมทั้งการส่งหน่วยของข้อมูลในชั้น logical link control และเทคนิคของการแก้ไขจากข้อผิดพลาด (error recovery)

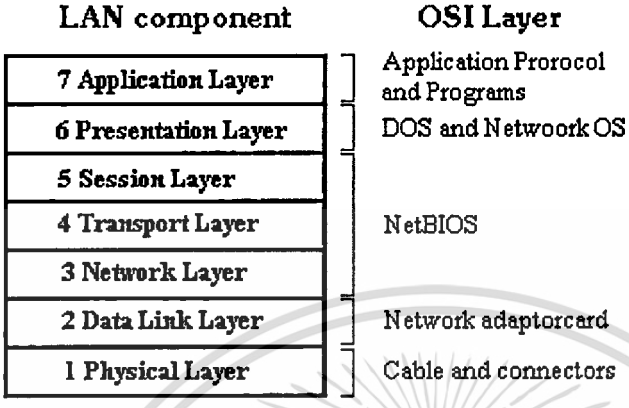
ชั้นย่อย medium access control (MAC) และ ชั้น network control จะติดต่อกับชั้น logical link control โดยชุดของการบริการพื้นฐาน 3 แบบ คือ request ซึ่งจะถามถึงการบริการที่จะเริ่มต้น (initiate) (สัญญาณจากชั้นที่ติดกันกับ logical link control) indication จะเป็นสัญญาณตอบรับจาก logical link control ไปยังชั้นที่ติดกันเกี่ยวกับการขอหรือให้สัญญาณกับ logical link control และ confirm ให้สัญญาณตอบรับจาก Logical link control ไปยังชั้นที่ติดกันเกี่ยวกับผลของการขอครั้งก่อน

สำหรับบริการแบบ connectionless ของ logical link control จะไม่ต้องสัญญาณตอบรับข้อมูล (acknowledge) ในการส่งข้อมูล แต่ถ้าเป็นการบริการแบบ connection-oriented การร้องขอจะมีการจัดตั้งการเชื่อมต่อสัญญาณ (connection establishment) ส่งข้อมูล การสิ้นสุดการเชื่อมต่อ (connection termination) การรีเซ็ตการเชื่อมต่อ (connection reset) และการควบคุมการไหลของข้อมูลในการเชื่อมต่อ (connection flow control)

เราสามารถสร้างระบบเครือข่ายท้องถิ่นได้หลายแบบด้วยกัน แต่การแบ่งชนิดของระบบเครือข่ายท้องถิ่นแต่ล่ะระบบโดยปกติจะดูจากองค์ประกอบ 3 อย่างคือ ชนิดของสายส่งข้อมูล (cable types) วิธีการเชื่อมต่อระบบ (topologies) วิธีการส่งข้อมูล (access methods)

2.2 โมเดล OSI กับระบบเครือข่ายท้องถิ่น

โมเดล OSI สามารถนำมาใช้กับระบบเครือข่ายคอมพิวเตอร์ได้ทุกชนิดรวมทั้งสถาปัตยกรรมของเครื่องมินิ เมนเฟรม และระบบเครือข่ายท้องถิ่นด้วย รูปที่ 2.1 จะแสดงถึงความสัมพันธ์ขององค์ประกอบในระบบเครือข่ายท้องถิ่นโมเดล OSI



รูปที่ 2.2.1 โมเดล OSI กับองค์ประกอบในระบบเครือข่าย

ชั้นที่ 1 รวมถึงสื่อที่ใช้ในการส่งข้อมูล (สายโทรศัพท์ สายโคแอกเซียล หรือเส้นใยนำแสง) รวมถึงตังเชื่อมต่อ ตัวขยายสัญญาณ และอุปกรณ์อื่นๆในการรับและส่งบิตของข้อมูล

ชั้นที่ 2 นำไปใช้งานบน Network Interface Card (NIC) ความรับผิดชอบภายในระบบเครือข่ายท้องถิ่นที่ชั้นนี้ คือ การกำหนดข้อมูลจากชั้นที่สูงขึ้นไปให้อยู่ในลำดับที่กำหนด Ethernet, Token Ring,StarLAN,ARCNET และรูปแบบเฟรมอื่นๆจะแตกต่างกันออกไปซึ่งแต่ล่ะเฟรมจะเป็นข้อมูลพื้นฐานของระบบเครือข่ายท้องถิ่น ฟังก์ชั่นอื่นๆในชั้นนี้ก็คือ การกำหนดตำแหน่งของต้นทางปลายทาง เพื่อระบุผู้ส่งและผู้รับของเฟรมทั้ง cyclical redundancy check (CRC) ในการควบคุมข้อผิดพลาด

ชั้นที่ 3 ถึงชั้นที่ 5 นำไปใช้งานใน NetBIOS (Network Basic Input/Output System) ที่พัฒนาขึ้นโดย IBM และ Sytek NetBIOS จะมีฟังก์ชั่นสำหรับ session layer (ชั้นที่ 5) เพื่อจัดการเชื่อมต่อทางตรรกระหว่างสถานีผู้ใช้ในระบบเครือข่ายท้องถิ่น ส่วนฟังก์ชั่นใน transport layer (ชั้นที่ 4) และ netwoek layer (ชั้นที่ 3) ไม่ต้องใช้ในระบบเครือข่ายท้องถิ่น แต่จะพบได้ใน WAN NetBIOS อาจจะอยู่ใน ROM บนการ์ดเชื่อมโยงระบบเครือข่ายหรือ จำลองอยู่ในซอฟต์แวร์ที่สถานีผู้ใช้และศูนย์บริการข้อมูล

ชั้นที่ 6 นำไปใช้งานในระบบปฏิบัติการของระบบเครือข่าย (NOS) และการจำลองในดอสที่สัมพันธ์กัน ตัวอย่างเช่น Netware ของ Novell,โปรแกรม PC LAN ของ IBM 3+ และ 3+Open ของ 3 Com,ViaNet ของ Western Digital,VINES ของ Banyan

ชั้นที่ 7 โปรแกรมโปรโตคอลและโปรแกรมประยุกต์

บทที่ 3
ระบบเครือข่ายท้องถิ่น

3.1 การใช้งานระบบเครือข่ายแบบ Ethernet

ระบบเครือข่ายแบบ Ehternet เป็นการใช้งานระบบเครือข่ายตามมาตรฐาน IEEE 802.3 ที่มีการส่งแบบเบสแบนด์ที่ความเร็ว 10 เมกะบิทต่อวินาที มีโทโพโลยีแบบบัส และใช้โปรโตคอลแบบ CSMA/CD ข้อมูลที่ถูกส่งในระบบเครือข่าย Ehternet จะส่งเป็นกลุ่มข้อมูลการสื่อสารของเฟรม ซึ่งรูปแบบของเฟรมมีตามรูปที่ 3.1

Preamble	Destn. Address	Source Address	Type Field	Data Field	Frame Check Sequence
8 ไบท์	6 ไบท์	6 ไบท์	2 ไบท์	46-1500 ไบท์	4 ไบท์

Ethernet Frame Format

Preamble	Start Delm	Destn. Address	Source Address	Type Field	Data Field	Pad Byte	Frame Check Sequence
7 ไบท์	1 ไบท์	2 หรือ	2 หรือ	2 ไบท์	0-n ไบท์	0-p ไบท์	4 ไบท์

IEEE 802.3 Frame Format

รูปที่ 3.1.1 รูปแบบกลุ่มสื่อสารของ IEEE 802.3 MAC

อุปกรณ์แต่ล่ะตัวในระบบเครือข่าย Ethenet จะมีตำแหน่งของอุปกรณ์ขนาด 48 บิท ซึ่งตำแหน่งของอุปกรณ์นี้สร้างมาจากโรงงานเลย โดย IEEE ในนิวยอร์ค สหรัฐอเมริกาจะควบคุมการกำหนดตำแหน่งเหล่านี้

เมื่ออุปกรณ์นี้ได้ส่งกลุ่มข้อมูลสื่อสารไปในระบบเครือข่ายแล้ว จะรออย่างน้อย 9.6 ไมโครวินาที ก่อนที่ส่งข้อมูลอีก ซึ่งช่วงเวลานี้เรียกว่าเป็น Inter-Packet-Gab (IPG) ซึ่ง IPG จะช่วยให้อุปกรณ์รับส่งมีเวลาเตรียมพร้อมที่ระรับกลุ่มข้อมูลสื่อสารโดยที่ไม่เกิดปัญหา ถ้าอุปกรณ์พยายามส่งข้อมูลและพบว่าสายไม่ว่ามันจะรอ ถ้าหลังจากที่ส่งกลุ่มข้อมูลไปแล้วเกิดการชนกันขึ้น อุปกรณ์รับส่งจะรอเป็นช่วงเวลาสุ่มสูงสุด 16 ครั้ง ถ้ายังส่งไม่ได้จะรายงานว่าเกิดข้อผิดพลาดขึ้น

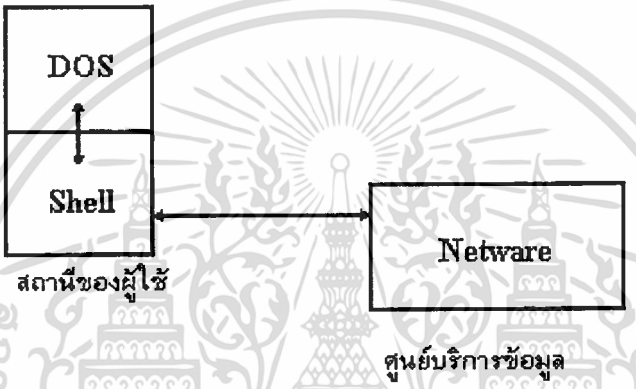
ความต้องการพื้นฐานของระบบเครือข่าย Ehternet ก็คือ เมื่อเกิดการชนกันของข้อมูลขึ้น ทุก ๆ สถานีจะต้องรับรู้ถึงการชนนั้น ก่อนที่สถานีที่เกี่ยวข้องกับการชนนั้นจะส่งเฟรมขนาดต่ำสุดคือ 572 บิต ออก ถ้ากฎเกณฑ์เหล่านีไม่ดีแล้ว อาจทำให้บางสถานีรับข้อมูลได้แต่บางสถานีไม่สามารถรับได้ ข้อมูล LLC จะมีความยาว 368 บิต (46 ไบท์) และ 12000 บิต (1500 ไบท์) ถ้าข้อมูลมูขนาดสั้นเกินไป จะเติม 0 ให้ครบ 368 บิท ซึ่งหมายความว่าถ้ากลุ่มข้อมูลใช้ขนาดต่ำสุดในการส่งแล้วสามารถส่งได้ด้วยความเร็ว 7,619,047 บิตต่อวินาที หรือส่งด้วย utilization 76% ส่วนที่เหลือในกลุ่มข้อมูลสื่อสารของ Ethernet ก็คือ

IPG และ Overhead อื่นๆ ถ้ากลุ่มข้อมูลสื่อสารใช้ขนาดสูงสุดในการส่งแล้ว จะสามารถส่งได้ด้วยความเร็ว 9,523,212 บิตต่อวินาที หรือมี Utilization 95.23 %

3.2 การทำงานของ Netware

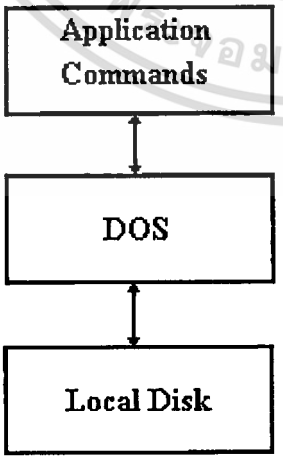
ระบบปฏิบัติการ Netware สนับสนุนการใช้งานเครื่องไมโครคอมพิวเตอร์ที่ต่างกัน หรือมีระบบปฏิบัติการที่ต่างกันโดย ไม่ต้องมีการแบ่งเนื้อที่บนดิสก์ไฟล์ทั้งหมดและ สถานีผู้ใช้ทุกสถานีสามารถใช้งานไดเรกทอรีร่วมกันได้

Netware จะอยู่ที่ศูนย์บริการข้อมูลและดอสจะอยู่ที่สถานีผู้ใช้ Netware Shell จะอยู่ที่สถานีของผู้ใช้เพื่อให้ดอสติดต่อกับ NetWare ได้



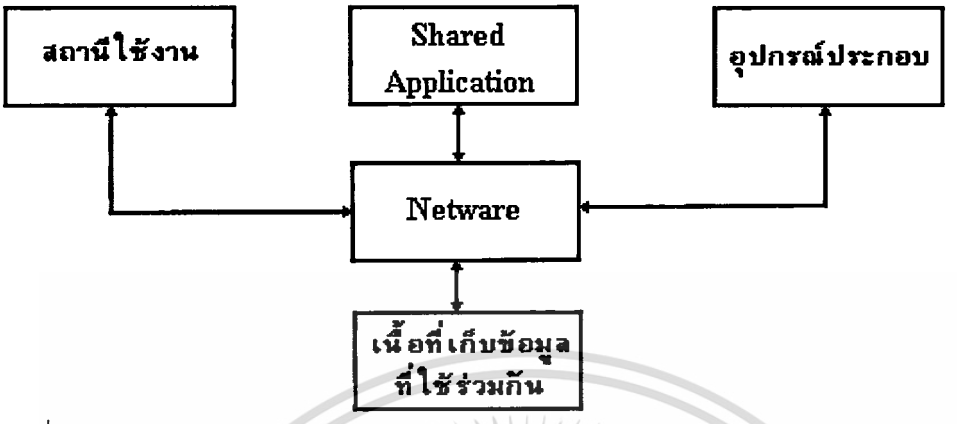
รูปที่ 3.2.1 Shell ของ Netware ทำหน้าที่ติดต่อระหว่างดอสของ สถานีผู้ใช้งานกับศูนย์บริการข้อมูล

ดอส Netware และ Shell จะทำงานร่วมกันเพื่อช่วยในการทำงานข้อมูลและฮาร์ดแวร์ร่วมกัน ดอสออกแบบให้ประมวลผลโปรแกรมที่สถานีผู้ใช้ในสภาพแวดล้อมแบบผู้ใช้คนเดียว (single-user environment)



รูปที่ 3.2.2 การทำงานของ DOS (สภาวะแวดล้อมแบบมีผู้ใช้คนเดียว)

Netware ออกแบบมาให้ประมวลผลในสภาพแวดล้อมที่มีผู้ใช้หลายคน(multiuser environment) ซึ่งจะช่วยในการใช้วานอุปกรณ์ต่างๆร่วมกัน ใช้งานโปรแกรมร่วมกัน ควบคุมไดเรกทอรีที่ซับซ้อนและ file allocation tables (FAT) ที่ศูนย์บริการข้อมูล



รูปที่ 3.2.3 การทำงานของ Netware (สภาพแวดล้อมแบบมีผู้ใช้หลายคน)

การรักษาความปลอดภัยและการจัดการไฟล์ที่ดอสไม่ได้จัดการให้จะถูกจัดการโดย Netware ที่ศูนย์ข้อมูลโดย Netware กำหนดบุคคลที่มีคุณสมบัติดังนี้

-ผู้ที่สามารถใช้งานศูนย์บริการข้อมูลได้

-ผู้ที่สามารถใช้งานไฟล์ได้

-ผู้ที่สามารถใช้งานไฟล์ร่วมกันได้

Netware นั้นไม่ใช้ดอสในสภาพแวดลอมของระบบเครือข่าย แต่ดอสจะถูกใช้ที่สถานีผู้ใช้เท่านั้น เพื่อที่การทำงานบนไดรฟ์ที่เป็นของตัวเองรวมทั้งโปรแกรมของผู้ใช้

Netaware ควบคุมการสร้าง และการจัดการข้อมูลบนศูนย์บริการข้อมูล ในขณะที่ Netware Shell จะช่วยให้ดอสและ Netware สามารถติดต่อกันได้

Netware Shell ซึ่งมีขนาดประมาณ 40 กิโลไบท์ จะอยู่ที่สถานีผู้ใช้และดักการใช้อินเทอร์รัปต์ 21h (การขอข้อมูลหรือคำสั่งของ Netware ที่อยู่บนศูนย์บริการข้อมูล)

ดอสจะเปรียบเทียบได้กับผู้ที่พูดและเข้าใจเฉพาะภาษาอังกฤษ Netware เป็นผู้ที่พูดและเข้าใจภาษาไทย shell จะเป็นล่ามระหว่างดอสและ Netware ทำให้ดอสและ Netware สามารถติดต่อกันได้

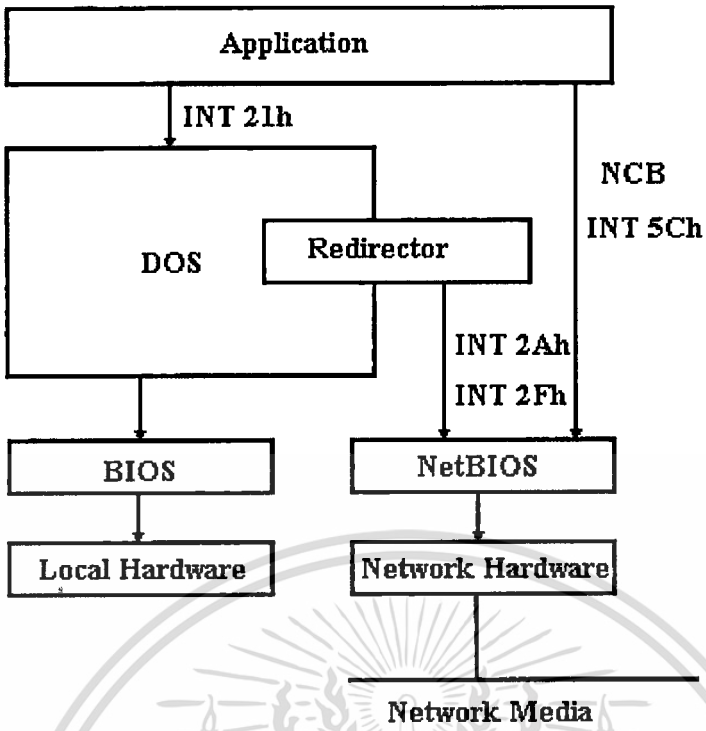
Shell ของ Netware ใช้เฉพาะ IPX ในการติดต่อกับศูนย์บริการข้อมูลเพราะในตัว Shell เองจะมีรูทีนในการส่งข้อมูลซ้ำถ้าเกิดการผิดพลาด และมีการแก้ไขจากข้อผิดพลาดต่างๆ ในการส่งอยู่แล้ว SPX ไม่ได้เรียกใช้โดย Shell แต่ถูกโหลดที่สถานีของผู้ใช้เพื่อช่วยให้โปรแกรมของผู้ใช้สามารถติดต่อกับ transport layer ได้ โปรแกรมของผู้ใช้จะต้องใช้ SPX เพื่อให้แน่ใจว่าในการส่งข้อมูล และควบคุมลำดับการทำงาน ถ้าโปรแกรมมีการติดต่อกับจุดหรือโหนดอื่นเป็นเวลานานหรือไม่รู้เวลาการตอบสนองที่แน่นอน เช่น การติดต่อสื่อสารแบบประตูสื่อสารและฐานข้อมูล,Optical-disk หรือ archive severs

Novell มีตัวจำลอง NetBOIS ในการติดต่อระหว่างโปรแกรมในระบบเครือข่ายซึ่งจะอยู่เหนือโปรโตคอล IPX อีกทีหนึ่ง โปรแกรมจะใช้การบริการของ NetBIOS ในการติดต่อสื่อสารแบบ interprocess กับจุดอื่นของ Netware เท่านั้น

เหนือจากชั้นการติดต่อสื่อสาร IPX/SPX จะเป็น Shell ของ Netware Novell และผู้ผลิตรายอื่น ๆ ใช้ Shell ของระบบเครือข่าย เพื่อเพิ่มประสิทธิภาพและเพิ่มฟังก์ชั่นการทำงานต่างๆ เมื่อ Shell ถูกโหลดมันจะเปลี่ยนเวกเตอร์การอินเตอร์รัปต์ (INT 21h) และจบงานแบบที่ค้างอยู่ในหน่วยความจำ (terminate but stay resident) การเรียกใช้งาน INT 21h จากโปรแกรมใดๆจะผ่านไปยัง Shell ก่อน ถ้าเป็นการเรียกใช้ทรัพยากรที่เป็นของตัวเอง (local) การขอผ่านไปให้ฟังก์ชั่นดอสตามปกติ แต่ถ้าพบว่าเป็นการขอทรัพยากรของระบบเครือข่ายก็จะจัดการการขอนี้เอง

3.3 NetBIOS

ซอฟต์แวร์ในตัวเครื่องที่ทำหน้าที่ในเชื่อมต่อระหว่างฮาร์ดแวร์กับผู้ใช้เรียกว่า BIOS (Basic Input/Output System) และ NetBIOS (Network Basic Input/Output System) เป็นส่วนขยายของระบบนี้เพื่อช่วยด้านการเชื่อมต่อด้านการสื่อสารกับฮาร์ดแวร์ของระบบเครือข่าย NetBIOS
จัดการการเชื่อมโยง ที่เรียกว่า virtual link
ระหว่างจุดหรือสถานีบนระบบเครือข่ายและส่งผ่านข้อมูลข้ามระหว่าง virtual link นั้น
นั่นคือเป็นการจัดการการติดต่อสื่อสารข้อมูลแบบจุดต่อจุด (peer-to-peer) ระหว่างสถานีบนระบบเครือข่าย ซึ่ง NetBIOS เป็นพื้นฐานของซอฟต์แวร์ของระบบเครือข่ายท้องถิ่น เช่น โปรแกรม IBM PC LAN และใช้กับบริการพิเศษบนระบบเครือข่ายเช่น Eicon SDLC gateway เป็นต้น



รูปที่ 3.3.1 ลักษณะการทำงานของ NetBIOS

บทที่ 4

การส่งและรับข้อมูลโดยใช้โปรโตคอล IPX

การรับส่งข้อมูลในระบบ LAN ของ NETWARE นั้น ข้อมูลจะมีการรับส่งในลักษณะของ Packet โดยโปรโตคอลที่งานคือ โปรโตคอล IPX / SPX ซึ่งได้ถูกพัฒนามาจาก XNS (Xerox Network Standard Protocol)

4.1 รูปแบบของการติดต่อสื่อสาร

การติดต่อสื่อสารสามารถแบ่งออกได้เป็นสองแบบ คือ

1.Datagram (Conectionless) คือวิธีการส่งข้อมูล โดยที่ไม่มีการเชื่อมต่อเส้นทางไว้ล่วงหน้า วิธีนี้จึงเป็นวิธีการส่งที่รวดเร็ว เพราะไม่ต้องเสียเวลาในการสร้างเส้นทาง Packet จะถูกส่งออกไปเรื่อย ๆ ดังนั้น Packet อาจจะเดินทางไปยังจุดหมายด้วยเส้นทางที่ต่างกันได้ และถ้าหากว่าเส้นทางไหนเกิดชำรุด เป็นไปได้ว่าข้อมูลอาจจะมีการสูญหาย

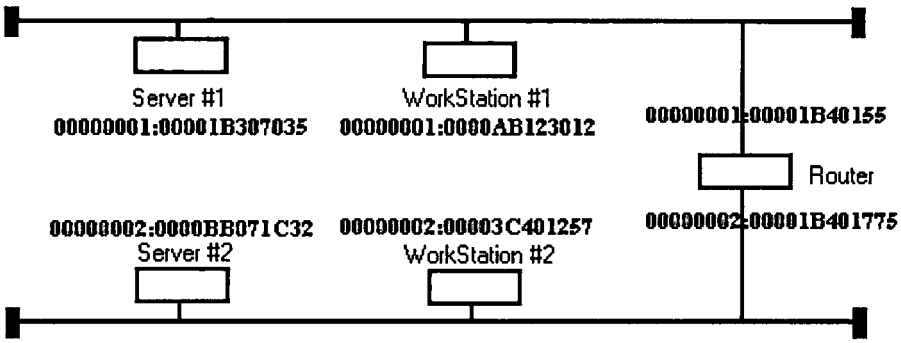
2.Session (Oriented connection) คือวิธีการที่มีการสร้างเส้นทางในการติดต่อสื่อสารล่วงหน้าก่อนมีการส่งข้อมูล ดังนั้น จึงต้องเสียเวลาส่วนหนึ่งในการสร้างเส้นทาง แต่วิธีนี้ข้อมูลมักจะได้รับครบถ้วน เพราะมีการส่งข้อมูลเป็นลำดับออกไป

4.2 ลักษณะ Address ใน Network

Workstation แต่ล่ะเครื่องจะต้องมี Address เฉพาะของตัวเองที่ไม่เหมือนกัน โดยจะประกอบไปด้วย

1.Network Address จะเป็นตัวบอกหมายเลขของ Network ของ Workstation ที่ติดตั้งอยู่

2.Node Address จะเป็นตัวบอกหมายเลข Address ของ Worlstation ที่ติดตั้งอยู่ที่กล่าวมาถ้าเกิดเป็นการติดต่อสื่อสารภายใน Segment เดียวกัน (ภายใต้ Server เดียวกัน) Network Address ของ Workstation แต่ละตัวจะมีค่าเหมือนกัน แต่ถ้าหากว่า เป็นการติดต่อสื่อสารคนล่ะ Segment กันจะต้องมีการระบุ Network Address ของผู้รับ และ ผู้ส่ง ซึ่งจะเป็นคนล่ะหมายเลขกัน จากรูปที่ 4.1 แสดงการระบุหมายเลขของ Network Address และ Node Address ที่อยู่ต่าง Segment กัน และเชื่อมต่อกันด้วย Bridge หรือ Router



รูปที่ 4.2.1 แสดงตัวอย่างของ Internetwork Address

4.3 ลักษณะของ Socket

ในการติดต่อระหว่าง Workstation แต่ละ Workstation ใน Network เนื่องจากว่าแต่ละ Workstation อาจมีการปฏิบัติงานโปรแกรมหลายโปรแกรมพร้อมกันอยู่ เพื่อให้สะดวกในการแยกว่า Packet เป็นของโปรแกรมใด จะต้องมีการกำหนดหมายเลข Socket ของแต่ล่ะโปรแกรม

โดย Network ได้มีการสงวน Socket หมายเลข 0000h-3FFFh และ 8000h-FFFFh ไว้ใช้เอง ดังนั้น หมายเลขของ Socket ที่สามารถกำหนดได้เองก็คือ 4000h-7FFFh โดยชนิดของ Socket จะมี 2 แบบดังนี้

1.Socket ชั่วคราว คือ หลังจากใช้งานเสร็จจะมีการปิด หรือลบ Socket ออก

2.Socket ถาวร คือ จะใช้ในโปรแกรมฝังตัว (Resident Program) จะใช้งานจนกว่าจะมีการปิด หรือยกเลิกอย่างชัดเจน

หมายเลข Socket ใน Workstation หนึ่งๆ อาจถูกใช้โดย Workstation อื่นใน Network ด้วยก็ได้ แต่ไม่สามารถเปิดหมายเลขซ้ำซ้อนในเครื่องเดียวกันได้

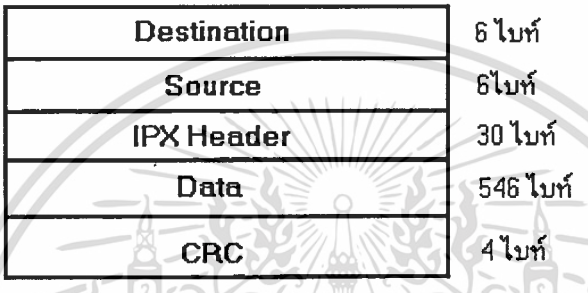
การติดต่อสื่อสารภายใต้โปรโตคอล IPX (Internetwork Packet Exchange Prorocol) เป็นการติดต่อสื่อสารในลักษณะ Peer-to-peer หมายความว่าเป็นการติดต่อโดยที่แต่ล่ะ Workstation สามารถติดต่อกันได้โดยตรงโดยที่ไม่ต้องผ่าน Fileserver ซึ่งก็หมายความว่าไม่จำเป็นที่จะต้องมีตัวกลางในการควบคุม

เนื่องจากว่าข้อมูลไม่จำเป็นต้องผ่าน Fileserver ดังนั้นเวลาที่ใช้ในการส่งจะน้อย เนื่องจากว่า Packet ของข้อมูลมีขนาดเล็ก คือ ไม่จำเป็นต้องมี Header ในส่วนของ Fileserver

4.4 โปรโตคอล IPX (Internetwork Packet Exchange Protocal)

โปรโตคอล IPX เทียบได้กับ Layer ที่ 3 ของ OSI Model การใช้โปรโตคอล IPX จะเป็นการติดต่อสื่อสารแบบ Datagram ดังนั้นจึงไม่มีการสร้างเส้นทางในการส่งข้อมูลก่อน ระหว่างผู้ส่งและผู้รับ ดังนั้นจึงไม่ต้องเสียเวลาในการสร้างเส้นทาง แต่ในการส่งแบบ Datagram เนื่องจากว่าไม่มีการสร้างเส้นทางก่อน Packet แต่ละ Packet จะเดินทางไปยังจุดหมายเอง จึงไม่อาจรับประกันได้ว่า ข้อมูลจากต้นทาง จะสามารถส่งไปถึงปลายทางได้ครบถ้วน ในได้มีการทดสอบแล้วว่า สามารถรับส่งข้อมูลได้ถูกต้องถึง 95%

ฟอร์แมตของ IPX Packet เป็นดังนี้



รูปที่ 4.4.1 แสดงฟอร์แมตของ IPX Packet

ตารางที่ 4.4.1 ส่วนประกอบของ IPX Header เป็นดังนี้

ชื่อ	หน้าที่
Checksum	มาจาก XNS ซึ่งเลิกใช้แล้ว มีค่าเป็น FFFFH เสมอ
Length	เก็บความยาวของ IPX Header บวกข้อมูลใน Packet
Transport Control	มีค่าเริ่มค้นเป็น 00H และจะเพิ่มค่าเมื่อ Packet วิ่งผ่าน Router นั้นคือฟิลด์นี้จะ เป็นตัวนับจำนวน Router ที่ Packet วิ่งผ่าน
Packet Type	บอกชนิดของ Packet
Destination Network	หมายเลขเครือข่ายของเครื่องปลายทาง
Destination Node	แอดเดรสขนาด 48 บิต ของเครื่องปลายทาง
Destination Socket	หมายเลขซ็อกเก็ตของเครื่องปลายทาง
Source Network	หมายเลขเครือข่ายของเครื่องต้นทาง
Source Node	แอดเดรสขนาด 48 บิท ของเครื่องต้นทาง
Source Socket	หมายเลขซ็อกเก็ตของเครื่องต้นทาง

ตารางที่ 4.4.2 ชนิดของแพ็กเกตมีดังนี้

ชนิด	ความหมาย
0	Unknown Packet Type
1	Routing Information Packet
2	Echo Packet
3	Error Packet
4	Packet Exchange Packet
5	Sequenced Packet Protocol Packet
16-31	Experimental Protocol
17	NetWare Core Protocol

ในการส่งโดยใช้โปโตคอล IPX เราจะกำหนดให้เป็นแบบชนิดที่ 4 เท่านั้น

4.5 บล็อกควบคุมเหตุการณ์(ECB : Event Control Blocks)

บล็อกควบคุมเหตุการณ์ (ECB) เป็นโครงสร้างที่ใช้ในการเรียงลำดับเหตุการณ์(event) ในการรับและส่งแพ็กเกต โดยมีโครงสร้างดังนี้

ตารางที่ 4.5.1 แสดงหน้าที่ของบล็อกควบคุมเหตุการณ์

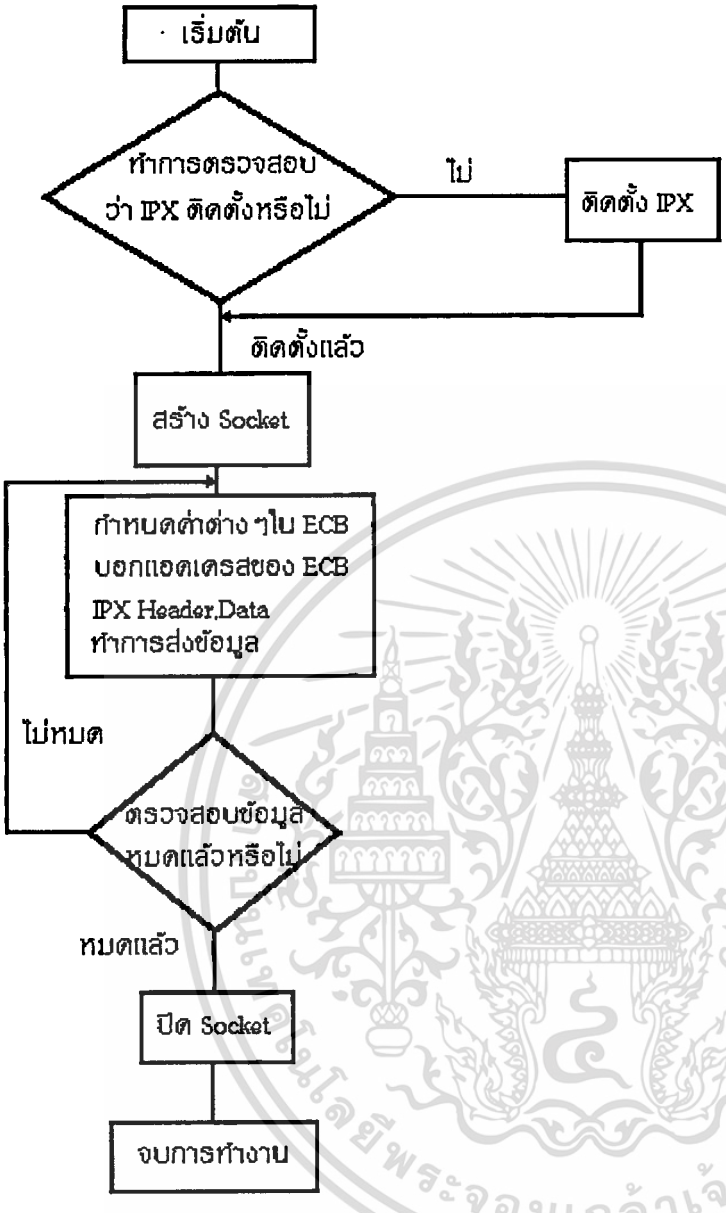
ชื่อ	หน้าที่
Link Address	จะถูกใช้โดย IPX ใช้ในการบอกสถานะ ECB เพื่อทำการรับส่งข้อมูล
ESR Address	แอดเดรสของ routine ที่จะถูกเรียกให้ทำงาน เมื่อแพ็กเกตถูกส่งมาถึงเครื่องรับ หรือกรณีที่แพ็กเกตถูกส่งไปจนสำเร็จ
In Use Flag	แสดงสถานะของ ECB
Completion Code	แสดงสถานะสุดท้ายของ ECB เมื่อเสร็จสิ้นการรับส่งข้อมูล
Socket Number	หมายเลขซ็อกเก็ตที่ใช้ในการรับส่งข้อมูล
IPX Workspace	ถูกใช้โดย IPX
Driver Workspace	ถูกใช้โดย IPX สำหรับ Network Driver
Immediate Address	ค่าแอดเดรสของ Bridge ในกรณีที่เป็นการส่งข้อมูลระหว่าง 2 Network หรือเป็นค่าแอดเดรสของเครื่องรับในกรณีที่ส่งข้อมูลใน Network เดียวกัน

Fragment Count,Fragment Address1,Fragment Size 1, Fragment Size2,Fragment Address2,agment Size Fragment Size2	แพ็กเกตที่ใช้ส่ง สามารถแบ่งออกได้เป็นส่วนๆ โดย Fragment count แสดงจำนวนของแพ็กเกตที่ถูกแบ่ง Fragment Size ใช้บอกขนาดของแต่ล่ะส่วนที่ถูกแบ่งส่วน Fragment Address แสดง pointer ที่ชี้ไปยัง Buffer ของแต่ล่ะส่วน

4.6 ลักษณะการส่งข้อมูลโดยใช้โปรโตคอล IPX

เนื่องจากว่าใน Workstation แต่ล่ะตัว อาจจะมีการปฏิบัติงานของโปรแกรมหลายโปรแกรมพร้อมกัน ดังนั้นในการส่งข้อมูล ข้อมูลจะถูกทำการแบ่งแล้วส่งออกไปเป็น Packet ดังนั้นระหว่าง Workstation จะต้องมีการระบุว่า Packet เป็นของโปรแกรมไหน โดยข้อมูลของแต่ละโปรแกรมจะผ่านทาง Socket โดยที่แต่ล่ะโปรแกรมจะต้องมี Socket เฉพาะตัวไม่ซ้ำกัน ดังนั้นแสดงว่าก่อนเริ่มการติดต่อจะต้องมีการระบุหมายเลข Socket ของแต่ล่ะโปรแกรม และเมื่อทำการ กำหนดหมายเลข Socket เรียบร้อยแล้ว ขั้นตอนต่อมาคือการกำหนดที่อยู่ของปลายทางหรือ Destination Address ซึ่งจะประกอบด้วย Network Address และ Node Address ของเครื่องรับปลายทาง และถ้าในกรณีที่เครื่องรับและเครื่องส่งอยู่คนล่ะ Segment กัน จะต้องมีการระบุ Immediate Address ของ Bridge หรือ Router ด้วย โดยในระหว่างการส่ง จะต้องมีการควบคุมการส่งโดยใช้ ECB(Event control blocks) เป็นตัวควบคุม และเมื่อทำการส่งข้อมูลเสร็จแล้ว จะต้องมีการยกเลิกการส่งข้อมูลด้วย

Flowchart แสดงการทำงานของการส่งข้อมูล



รูปที่ 4.6.1 Flow Chart แสดงขั้นตอนการส่งข้อมูล

ขั้นตอนการส่งข้อมูลโดยใช้โปรโตคอล IPX

1.ทำการตรวจสอบว่า IPX ทำการติดตั้งหรือไม่ ถ้ายังไม่ติดตั้งให้ทำการติดตั้ง

2.สร้าง Socket ขอทางด้านผู้ส่ง โดยค่าจะมีค่า ตั้งแต่ 4000h-7FFFh

3.ทำการหา Connection number ของผู้รับปลายทาง

4.ทำการหา Network address และ Node address ของเครื่องรับปลายทางจาก Connection number ที่หาได้จากข้อ 3

5.ทำการตรวจสอบว่า ระหว่างเครื่องรับและเครื่องส่ง มี Bridge หรือ Router อยู่หรือไม่ ถ้าหากมี ค่าของ Immediate address จะเป็นค่าแอดเดรสของ Bridge หรือ Router แต่ถ้าหากว่าไม่มี ค่าของ Immediate address จะเป็นค่าแอดเดรสของเครื่องรับปลายทาง

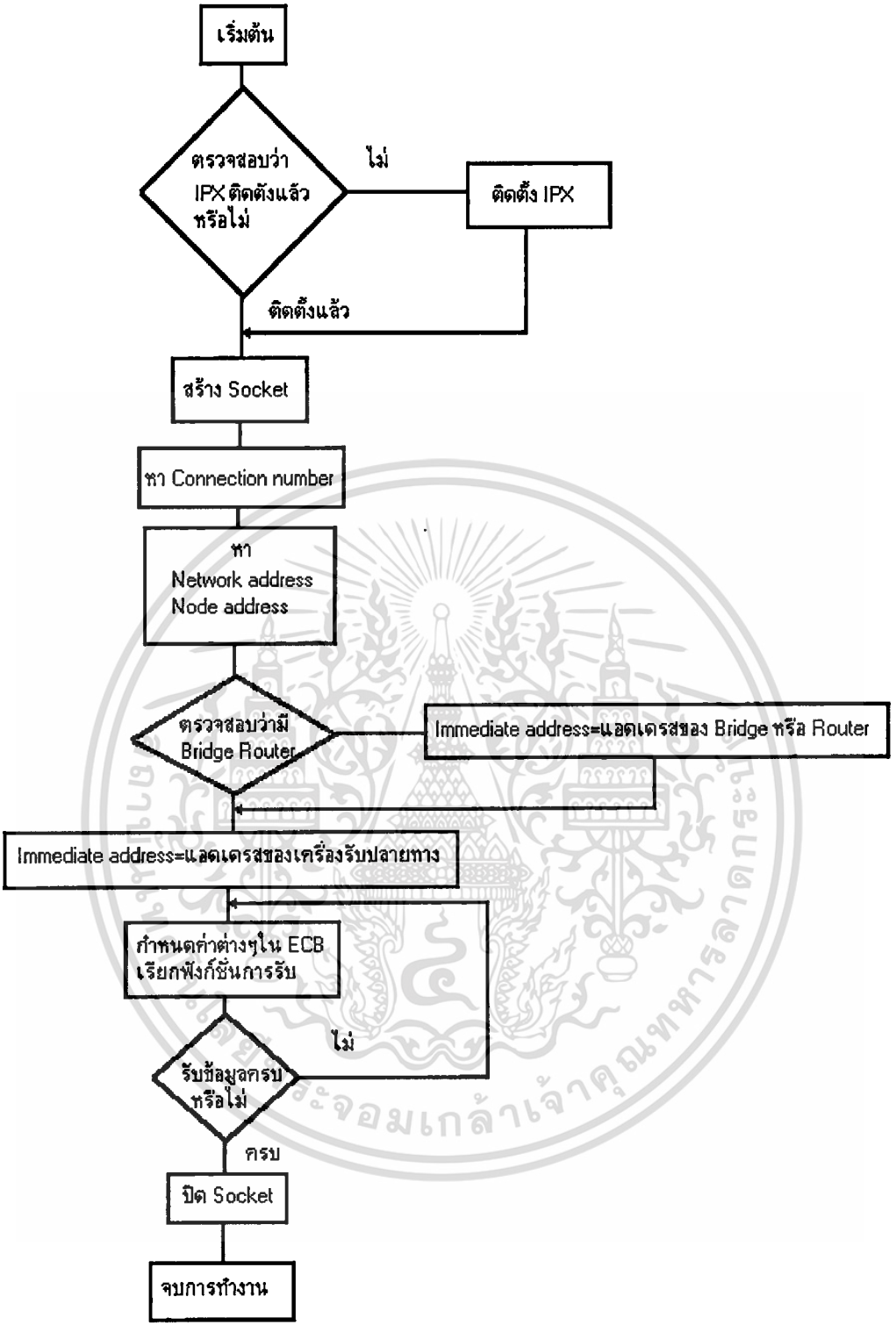
6.ก่อนทำการส่งข้อมูล ถ้าหากว่าข้อมูลมีจำนวนมากจะไม่สามารถส่งไปได้ในแพ็กเกตเดียว ดังนั้นจะต้องทำการส่งไปทีละส่วน แล้วทำการรวบรวมที่เครื่องรับปลายทาง

7.กำหนดค่าต่างของ ECB และกำหนด Socket ของเครื่องรับปลายทางให้อยู่ในค่าตั้งแต่ 4000h-7FFFh บอกแอดเดรสของ ECB , IPX Header และ Data ทำการเรียกฟังก์ชั่นการส่งข้อมูล เพื่อทำการส่งข้อมูล

8.เมื่อข้อมูลยังส่งไม่ครบทำขั้นตอนที่ 7 ซ้ำจนกว่าข้อมูลจะส่งหมด

9.ทำการปิด Socket

10.จบการส่งข้อมูล



รูปที่ 4.6.2 Flow Chart แสดงขั้นตอนการรับข้อมูล

ขั้นตอนการรับข้อมูลโดยใช้โปรโตคอล IPX

1.ทำการตรวจสอบว่า IPX ทำการติดตั้งหรือไม่ ถ้ายังไม่ติดตั้งให้ทำการติดตั้ง

2.สร้าง Socket ขอทางด้านผู้รับ โดยค่าจะมีค่า ตั้งแต่ 4000h-7FFFh

3.ทำการกำหนดค่าต่างๆใน ECB และทำการกำหนดแอดเดรสของ ECB และ IPX Header ทำการเรียกฟังก์ชั่นการรับเพื่อทำการรับข้อมูล

4.เมื่อทำการรับข้อมูลไม่ครบให้กลับไปทำขั้นตอนที่ 3 จนกว่าจะรับข้อมูลครบ

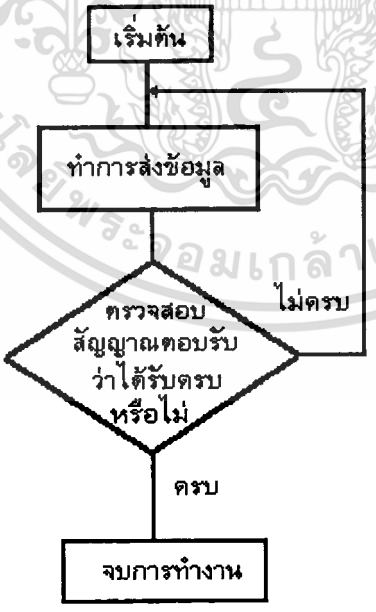
5.ปิด Socket

6.จบการรับข้อมูล

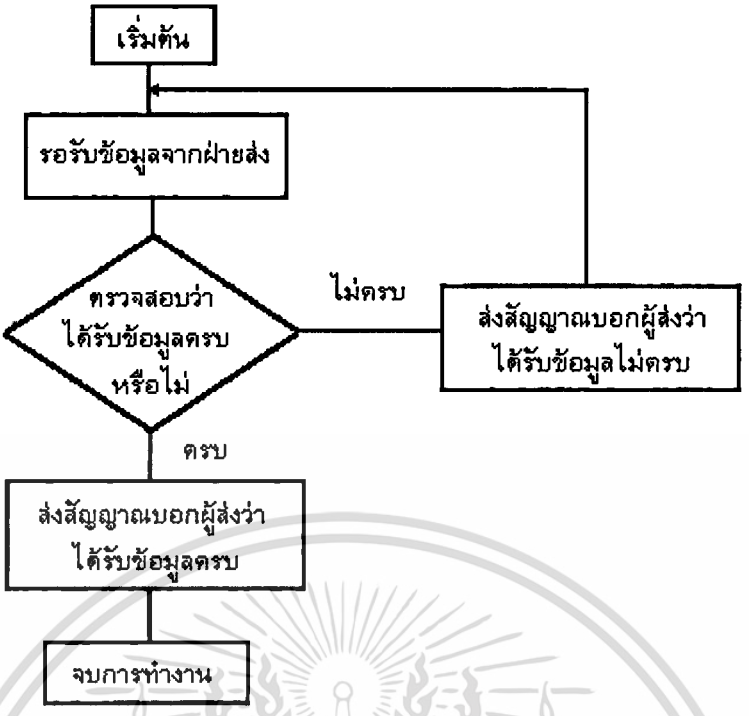
4.7 การตราจสอบข้อผิดพลาดในการรับส่งข้อมูล

เนื่องจากว่าเป็นการส่งข้อมูลในลักษณะ Datagram ดังนั้นโอกาสที่ฝ่ายรับได้รับข้อมูลไม่ครบถ้วนอาจจะเกิดได้ ดังนั้นจึงได้หาทางแก้ไขโดยการทางด้านฝ่ายผู้ส่งทำการบอกจำนวน Packet ว่าได้ส่งไปจำนวนเท่าไหร่ จากนั้นทางด้านฝ่ายรับจะทำการนับจำนวน Packet ที่ตนได้รับ แล้วทำการเปรียบเทียบกับจำนวนที่ทางฝ่ายส่งบอกมาว่าตรงกันหรือไม่ ถ้าไม่ตรงทางด้านฝ่ายรับจะทำการส่งสัญญาณไปทางด้านฝ่านส่งเพื่อบอกให้ฝ่ายส่งทำการส่งข้อมูลทั้งหมดมาใหม่อีกครั้ง แต่ถ้าหากว่าทางด้านฝ่ายรับได้รับข้อมูลครบแล้ว ก็จะต้องทำการส่งสัญญาณบอกทางด้านฝ่ายส่งด้วยว่า ได้รับข้อมูลครบถ้วนเพื่อเป็นการยืนยันกับฝ่ายส่ง

รูปดังต่อไปนี้เป็นการแสดงวิธีการตรวจสอบว่าได้รับข้อมูลครบหรือไม่



รูปที่ 4.7.1 Flow Chart แสดงการตรวจสอบการผิดพลาดทางด้านฝ่ายส่ง



รูปที่ 4.7.2 Flow Chart แสดงการตรวจสอบการผิดพลาดทางด้านฝ่ายรับ

บทที่ 5 โปรแกรมควบคุมกราฟฟิกส์

5.1 การควบคุมการแสดงผล

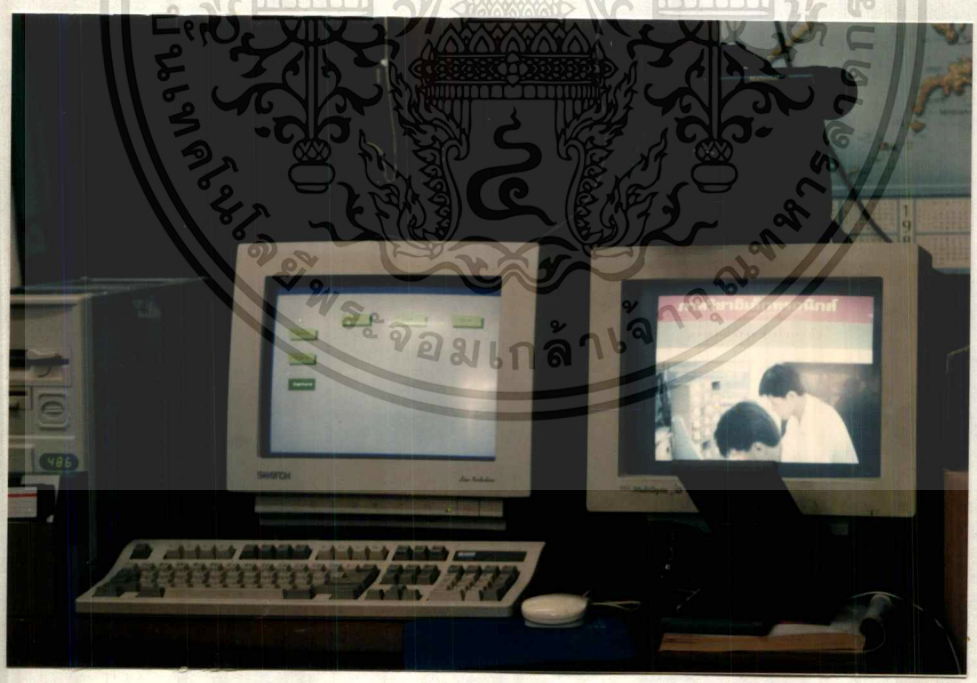
ในการทำงานส่วนของกราฟฟิกส์นี้ จะเป็นเรื่องเกี่ยวกับการประมวลผลรูปภาพ (Image Processing) ซึ่งส่วนประกอบของการควบคุมและการแสดงผล และหน้าที่ของแต่ละส่วนมีดังนี้

1. การ์ด VGA ทำหน้าที่ ควบคุมจอภาพที่เราใช้เป็น จอแสดงเมนูสำหรับควบคุมการทำงานของระบบทั้งหมด

2. การ์ด TARGA ทำหน้าที่ ควบคุมจอภาพที่เราใช้เป็นจอแสดงภาพที่เราทำการประมวลผลภาพ

3. กล้อง V.D.O. ทำหน้าที่ นำสัญญาณภาพที่เป็นสัญญาณ Analog จากภายนอกระบบมาประมวลผล

4. เครื่องคอมพิวเตอร์ (PC) เป็นส่วนสำคัญที่สุดนั่นคือเป็นส่วนกลางในการควบคุมส่วนประกอบทั้งหมดทั้ง 3 ส่วน ให้มีการทำงานร่วมกัน และยังทำหน้าที่ประมวลผลข้อมูลของระบบทั้งหมด



รูปที่ 5.1.1 ส่วนประกอบของการประมวลผลภาพ

นั่นคือ ในการทำงาน เราจะสั่งงานจากจอภาพที่ควบคุมโดยการ์ด VGA ซึ่งจอภาพนี้จะมีเมนูควบคุมการทำงานของระบบทั้งหมด เช่น

- Clear จอภาพที่ควบคุมโดยการ์ด TARGA
- สั่งให้การ์ด TARGA รับสัญญาณภาพที่เป็นสัญญาณ Anaolog จากกล้อง V.D.O. ซึ่ง เป็นภาพเคลื่อนไหว (Interactive) มาแสดงผลออกทางจอภาพ
- จับภาพ (Capture) นั่นคือการนำข้อมูลเกี่ยวกับสัญญาณภาพ Interactive ในช่วง เวลาที่เราเลือกเก็บลงในหน่วยความจำของการ์ด TARGA เพื่อนำไปประมวลผลต่อไป
- Save , Load ข้อมูลภาพลง หรือ ออกจาก Harddisk
- ประมวลผลภาพที่เราต้องการ เช่น การแยกสี
- ทำการส่งข้อมูลที่ต้องการ ผ่านโครงข่าย LAN ไปให้กับ User ต่าง ๆ

5.2 การเขียนโปรแกรมเมนูควบคุมการทำงาน

จากที่กล่าวมาข้างต้น เราทราบว่าเมนูควบคุมการทำงานของระบบนี้ เราใช้จอภาพที่ควบคุมโดยการ์ด VGA ดังนั้นเราจึงต้องเขียนโปรแกรมควบคุมการ์ด VGA ซึ่งเคยกล่าวไว้แล้วในส่วนของเทอมที่แล้ว

การเขียนโปรแกรมเมนูนี้ จะเขียนอยู่ใน Graphics Mode ซึ่ง Function ต่าง ๆ ที่ใช้ใน Graphics Mode นี้ จะถูกรวมอยู่ใน <graphics.h> เช่น คำสั่งการวาดเส้นตรง (lineto) , คำสั่งการเขียนตัวอักษร (outtextxy) , คำสั่งการวาดสี่เหลี่ยม (rectangle) , คำสั่งการระบายสี (fillrectangle) , คำสั่งการเลือกสี (setcolor) เป็นต้น นอกจากนี้ในการสั่งงานเราจะใช้ mouseดังนั้นจึงต้องมีการเขียนโปรแกรมควบคุม mouse ด้วย

เมนูที่สร้างขึ้นนี้จะเป็นในลักษณะที่มีการเคลื่อนไหว ซึ่งมีความซับซ้อนพอสมควรดังนั้น จึงมีการสร้าง function ขึ้นมาใหม่เพื่ออำนวยความสะดวก ในส่วนนี้จึงขออธิบายเพียง Function ต่าง ๆ ที่สร้างขึ้นมา และ การเขียนโปรแกรมควบคุม mouse เท่านั้น

5.2.1 Function ต่าง ๆ ที่สร้างขึ้น

1. void Fillbox(int x1,int y1,int x2,int y2,int color) ทำหน้าที่วาดกล่องและระบายสีภายในกล่อง โดยตัวแปร x1,y1,x2,y2 เป็นตัวกำหนดคู่ลำดับของมุมซ้ายบนและมุมขวาบนของกล่องตามลำดับ ส่วน Color เป็นตัวกำหนดสีที่จะระบายลงในกล่อง

```
void Fillbox(int x1,int y1,int x2,int y2,int color)
 {
        setcolor(color);
        setfillstyle(1,color);
        bar(x1,y1,x2,y2);
 }
```

2. void Choice(int x,int y,char text[10]) เป็น Function ที่ใช้สำหรับวาดรูปปุ่มที่ใช้สำหรับเลือก โดยตัวแปร x,y ใช้กำหนดคู่อันดับของตำแหน่งที่ต้องการ ส่วน text[10] เป็นข้อความบนปุ่มนั้น

```
void Choice(int x,int y,char text[10])
  {
        Fillbox(x+7,y+7,x+7+width,y+7+hight,GREEN);
        Fillbox(x,y,x+width,y+hight,LIGHTGREEN);
        setcolor(RED);
        outtextxy(x,y+12,text);
  }
```

3. void Full_Menu(void) เป็น Function ที่ใช้วาดเมนูหลัก

```
void Full_Menu(void)
  {
        Choice(xmenu,ymenu,menu[0]);
        Choice(xmenu+skip,ymenu,menu[1]);
        Choice(xmenu+(skip*2),ymenu,menu[2]);
  }
```

4. void Move(int x,int y,int num,int init) เป็นการวาดเมนูตามแนวดิ่ง ซึ่งจะเป็นลักษณะ

Interactive โดยที่ x,y เป็นค่าที่ใช้กำหนดตำแหน่งเริ่มต้น ค่า num เป็นค่าของจำนวนปุ่มในแนวนอน ส่วนค่า init เป็นค่าเริ่มต้นของตัวอักษรที่อยู่ใน array ซึ่งจะนำไปเขียนไว้บนปุ่ม

```
void Move(int x,int y,int num,int init)
{
  int y1,i,j;
  y1 = y+7;
  j = 0;

     do
            {
          for(i=0;i<=step;i++)
         {
         Clearpos(x+7,y1,width,hight);
         y1++;
         Fillbox(x+7,y1,x+width+7,y1+hight,GREEN);
         delay(3);
       }
         Fillbox(x,y1-7,x+width,y1-7+hight,LIGHTGREEN);
         setcolor(RED);
         outtextxy(x,y1-7+12,ymenu1[init+j]);
         y1=y1+hight;
         j++;
         num--;
       } while(num != 0);
}
```

5. ส่วนเมนูย่อยเป็นการกำหนดต่าที่แน่นอนตายตัว จึงมีการกำหนดไว้ที่ต้นโปรแกรมดังตัวอย่าง

```
#define    Menu_Y_1()   Move(xmenu,ymenu,3,0)
#define    Menu_Y_2()   Move(xmenu+skip,ymenu,3,3)
```

6. ในโหมด Graphics การรับค่าข้อมูลจากการ key เข้า ถ้าเราใช้คำสั่ง scanf แล้ว จะทำให้สีต่าง ๆ ที่บริเวณนั้นเสียไป ดังนั้นเราจึงแก้ปัญหาโดยการใช้คำสั่ง getch() แทน โดยการรับค่าเข้ามาทีละ 1 ตัว แล้วเก็บค่าลงไปใน array ดังตัวอย่างข้างล่าง

```
void InData(int x,int y,char text[10])
{   int i;
        char ch;
        int x1=x+7+width;
        int y1=y+7+hight;
    for(i=0;i<=300;i++)
    {
        Fillbox(x1,y+7,x1+i,y1,GREEN);
    }
        for(i=0;i<=35;i++)
    {
        Fillbox(x+7,y1,x1+300,y1+i,GREEN);
    }
        Fillbox(x,y,x+width+300,y+hight+35,LIGHTGREEN);
    setcolor(BLACK);
    rectangle(x+5,y+5,x+width+295,y+hight+30);
    outtextxy(x+10,y+15,text);
        Fillbox(x+10,y+35,x+360,y+50,BLUE);

   for(i=0;i<50;i++)
   {
        buffer[i]=0;
```

```
}
i=0;
do
{
  ch = getch();
        if(ch!=27&&ch!=13&&ch!=8)
  {
          Fillbox(x+10,y+35,x+360,y+50,BLACK);
    buffer[i]=ch;
          i++;
    setcolor(WHITE);
          moveto(x+12,y+40);
    outtext(strcat(buffer,"_"));
  }
  if(ch==8&&i>0)
        {
          Fillbox(x+10,y+35,x+360,y+50,BLACK);
    i--;
    buffer[i]=0;
          setcolor(WHITE);
          moveto(x+12,y+40);
    outtext(strcat(buffer,"_"));
         }
  if(ch==27)
  {
       for(i=0;i<49;i++)
       {
       buffer[i]=0;
       }
```

```
        }

    }while(ch!=13&&ch!=27);
    buffer[i]=0;
}
```

7. void WinText(char text[50]) เป็น Function ที่ใช้เขียนคำแนะนำต่าง ๆ เราจะใช้หลัก การในการนับตัวอักษรเพื่อคำนวณหาความยาวของ Window เพื่อทำให้ขนาดของ Window ยาวพอดีกับความยาวของข้อความ

```
void WinText(char text[50])
{
    len=((strlen(text))*8)+20;
    ywin=380;
    xwin=(getmaxx()/2)-(len/2);
    Fillbox(xwin,ywin,xwin+len,ywin+30,MAGENTA);
    Fillbox(xwin+5,ywin+5,xwin+len-5,ywin+25,LIGHTMAGENTA);
    setcolor(BLACK);

    outtextxy(xwin+12,ywin+13,text);
}
```

5..2.2 การเขียนโปรแกรมควบคุม mouse

สิ่งแรกที่ Software ต้องการสำหรับการทำงานกับ mouse ก็คือ เราต้องทำให้ Softwareได้ รับรู้ว่ามี mouse อยู่ ซึ่งแน่นอนว่าการต่ออุปกรณ์ mouse เพียงอย่างเดียวย่อมไม่สามารถบอก Software ได้ โปรแกรมเท่านั้นที่คุยกับโปรแกรมรู้เรื่อง เราจึงจำเป็นต้องติดตั้ง mouse driver เพื่อ บอกให้ Software ทราบก่อน

mouse driver คือโปรแกรมที่สามารถเข้าใจถึงสัญญาณที่มาจากพอร์ตที่อุปกรณ์ mouse ต่ออยู่ได้ mouse driver จะแปลงสัญญาณเหล่านี้ไปเป็นการกระทำที่เหมาะสมไปบนจอภาพ

ต่อจากนี้เราจะมาดูว่าเราจะเขียนโปรแกรมอย่างไรเพื่อที่จะใช้ควบคุม mouse ได้

<u>การใช้อินเตอร์รัปต์ไบออส</u>

ในการติดต่อกับ mouse เราจะเรียกใช้บริการของอินเตอร์รัปต์เบอร์ 33 h และเราจะให้ค่าผ่าน ทางรีจิสเตอร์ AX เพื่อเป็นการเรียกใช้บริการต่าง ๆ ส่วนรีจิสเตอร์ BX,CX,DX ในการส่งและรับค่าระ - หว่าง Software กับ mouse ดังตัวอย่าง

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

main()
{
  int x,y,buttons;
  union REGS regs;

  regs.x.ax = 0;
  /* initialize mouse and get initialization status */
  int86( 0x33 , &regs , &regs ) ;
  if ( regs.x.ax == 0 )
  {
     printf("No Mouse Available\n");
     exit(1);
  }
  clrscr();
  gotoxy( 40,1 );
  printf("press both buttons to exit");
  regs.x.ax = 1 /* show mouse pointer */
```

```
  int86( 0x33 , &regs , &regs );
  do
  {
     regs.x.ax = 3;
     int86( 0x33 , &regs , &regs );
     buttons = regs.x.bx &3;
     x = regs.x.cx;
     y = regs.x.dx;
     gotoxy(1,1);
     printf("x=%3d,y=%3d" ,x,y);
  } while( buttond != 3);
  clrscr();
  gotxy(1,1);
  printf("have a mice day");
}
```

จากตัวอย่าง เราเรียกใช้อินเตอร์รัปต์เบอร์ 33h หลังจากให้ค่า 0 แก่รีจิสเตอร์ AX เพื่อบ่งบอกถึงการใช้บริการที่ 0 ของอินเทอร์รัปต์เบอร์ 33h ซึ่งมีหน้าที่ตรวจสอบว่า mouse driver ได้ถูกโหลดเข้ามาไว้ในหน่วยความจำแล้วหรือยัง ถ้ายังบริการนี้ก็จะส่งค่า 0 กลับมาในรีจิสเตอร์ AX ซึ่งในที่นี้เป็นสมาชิกตัวหนึ่งของ union regs

จากนั้นจึงเป็นการเรียกใช้บริการที่ 1 อินเตอร์รัปต์เบอร์ 33h โดยใส่ค่า 1 ในรีจิสเตอร์ AX แล้ว execute อินเตอร์รัปต์เบอร์ 33h จะทำงานด้วยฟังก์ชัน int86() ซึ่งทำหน้าที่แสดง mouse cursor ให้ปรากฏขึ้น

ลูป do while จะ execute จนกว่าเราจะคลิกปุ่มบน mouse ทั้ง 2 ปุ่มพร้อม ๆ กันภาย ในลูปนี้เราให้ค่า 3 แก่รีจิสเตอร์ AX เพื่อเรียกใช้บริการที่ 3 ของอินเตอร์รัปต์เบอร์ 33h ซึ่งมีหน้าที่อยู่ 2อย่าง คือ บอกเราว่าปุ่มใดบน mouse ถูกคลิกและให้ตำแหน่งบนจอภาพที่ mouse cursor อยู่ และเช่นเดิมใช้ฟังก์ชัน int86() ในการ execute อินเตอร์รัปต์เบอร์ 33h

ค่าของปุ่มที่ถูกคลิกที่ส่งกลับจาก บริการที่ 3 ของอินเตอร์รัปต์เบอร์ 33h จะถูกเก็บไว้ในรีจิสเตอร์ BX ซึ่งใน 2 byte นี้จะมีเพียง 3 bit ต่ำเท่านั้นที่มีความสำคัญ ถ้า bit ที่ 0 ถูกเซ็ตจะบ่งบอกว่าปุ่มซ้ายถูกคลิก หากbit ที่ 1 ถูกเซ็ตจะบ่งบอกว่าปุ่มขวาถูกคลิก ส่วน bit ที่ 2 ถูกเซ็ตจะบ่งบอกว่า

ปุ่มกลางถูกคลิก ด้วยการกระทำทางบิตแบบ AND กับค่าในรีจิสเตอร์ BX ด้วย 3 ทำให้เราสามารถล่วงรู้ถึงค่าใน 2 บิตต่ำคือบิตที่ 0 และบิตที่ 1 ได้ ถ้าทั้งคู่ถูกเซ็ตอยู่จะได้ค่าเป็น 3 ซึ่งทำให้รู้ว่ามีการคลิกทั้งปุ่มซ้ายและปุ่มขวาพร้อมกัน เราได้เก็บผลการ AND ไว้ในตัวแปร buttons เพื่อใช้เป็นเงื่อนไขในการออกจากลูป do while

ค่าในรีจิสเตอร์ CX และรีจิสเตอร์ DX จะเป็นค่าที่มาจากการส่งกลับของบริการที่ 3 ของ อินเตอร์รัปต์เบอร์ 33h ซึ่งเป็นค่าบรรทัดและค่าคอลัมน์ที่ mouse cursor อยู่ตามลำดับ ค่า 2 ค่านี้จะเปลี่ยนไปในแต่ละครั้งที่มีการเคลื่อนย้าย mouse

การให้บริการอินเตอร์รัปต์เบอร์ 33h

จากที่กล่าวมาแล้วจะเห็นว่าในการเขียนโปรแกรมควบคุม mouse น้น เราต้องใช้บริการของอินเตอร์รัปต์เบอร์ 33h ในการเลือกใช้บริการที่ 3 ของอินเตอร์รัปต์เบอร์ 33h จะทำได้ โดย การส่งค่าผ่านไปทางรีจิสเตอร์ AX ส่วนรีจิสเตอร์ BX,CX,DX จะทำหน้าที่ส่งและรับข้อมูลระหว่างSoftware กับ mouse ดังตัวอย่างที่กล่าวมาแล้ว ต่อไปนี้จะเป็นหน้าที่และค่าในรีจิสเตอร์ต่าง ๆ ของแต่ละบริการของอินเตอร์รัปต์เบอร์ 33h ที่สำคัญ ๆ

1. บริการที่ 0 ของอินเตอร์รัปต์เบอร์ 33h ทำหน้าที่ตรวจสอบว่า mouse driver ถูกโหลดเข้ามาในหน่วยความจำหรือยัง

ค่าที่ป้อนเข้า

AX = 0 เรียกใช้บริการที่ 0 ของอินเตอร์รัปต์เบอร์ 33h

ค่าที่ส่งกลับ

AX = 0 เมื่อไม่มีการโหลด mouse driver ลงในหน่วยความจำ

AX = 1 เมื่อมีการโหลด mouse driver ลงในหน่วยความจำแล้ว

2. บริการที่ 1 ของอินเตอร์รัปต์เบอร์ 33h ทำหน้าที่แสดงตำแหน่งของ mouse cursor

ค่าที่ป้อนเข้า

AX = 1 เรียกใช้บริการที่ 1 ของอินเตอร์รัปต์เบอร์ 33h

ค่าที่ส่งกลับ

ไม่มี

3. บริการที่ 2 ของอินเตอร์รัปต์เบอร์ 33h ทำหน้าที่ทำให้ mouse cursor หายไปี่

ค่าที่ป้อนเข้า

AX = 2 เรียกใช้บริการที่ 2 ของอินเตอร์รัปต์เบอร์ 33h

ค่าที่ส่งกลับ

ไม่มี

4. บริการที่ 3 ของอินเตอร์รัปต์เบอร์ 33h ทำหน้าที่ 2 อย่างคือ บอกสถานะของ mouse ว่าปุ่มไหนถูกคลิก และ บอกตำแหน่งของ mouse cursor

ค่าที่ป้อนเข้า

AX = 3 เรียกใช้บริการที่ 3 ของอินเตอร์รัปต์เบอร์ 33h

ค่าที่ส่งกลับ

BX - ถ้าบิตที่ 0 ถูกเซ็ตแสดงว่าปุ่มซ้ายถูกคลิก

- ถ้าบิตที่ 1 ถูกเซ็ตแสดงว่าปุ่มขวาถูกคลิก

CX - บอกค่าตำแหน่งของ mouse cursor ในแกน X

DX - บอกค่าตำแหน่งของ mouse cursor ในแกน Y

5. บริการที่ 7 และ 8 ของอินเตอร์รัปต์เบอร์ 33h ทำหน้าที่ กำหนดขอบเขตของ mouse cursor

ค่าที่ป้อนเข้า

AX = 7 กำหนดขอบเขตของ mouse cursor ในแนวดิ่ง

CX - กำหนดขอบเขตของ mouse cursor ทางด้านซ้าย

DX - กำหนดขอบเขตของ mouse cursor ทางด้านขวา

AX = 8 กำหนดขอบเขตของ mouse cursor ในแนวนอน

CX - กำหนดขอบเขตของ mouse cursor ทางด้านบน

DX - กำหนดขอบเขตของ mouse cursor ทางด้านล่าง

ค่าที่ส่งกลับ

ไม่มี

5.3 การประมวลผลภาพ (Image Processing)

จากที่กล่าวมาข้างต้นเป็นส่วนของเมนูซึ่งเราจะสั่งงานทางจอภาพที่ควบคุมโดยการ์ด VGA แต่ในส่วนการประมวลผลภาพเราจะใช้จอภาพที่ควบคุมโดยการ์ด TARGA ซึ่งมีความสามารถและจำนวนสีมากกว่าการ์ด VGA และที่สำคัญการ์ด TARGA สามารถต่อเชื่อมกับกล้อง V.D.O. ได้โดยที่สามารถแสดงภาพที่อยู่ในลักษณะ Interactive

5.3.1 โหมดการแสดงผลของการ์ด TARGA

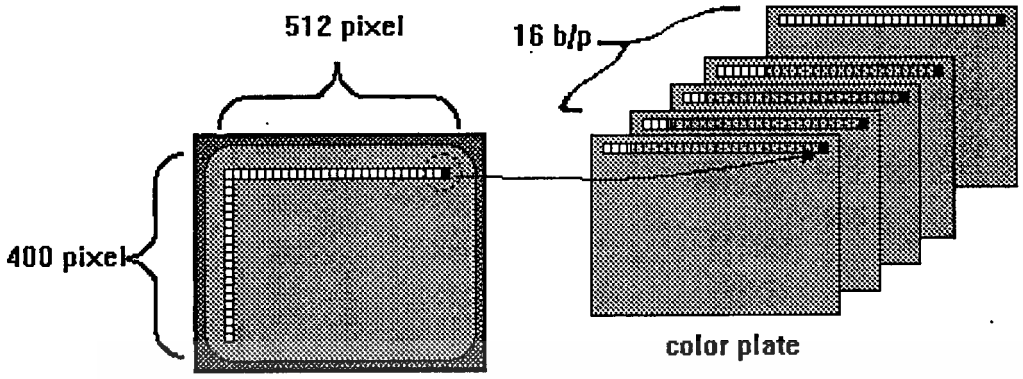
เนื่องจากจอภาพที่ใช้แสดงผลมีหลายชนิด ดังนั้นในการจอแสดงผลเหล่านี้ด้วยการ์ด TARGA ตัวการ์ด TARGA จึงต้องสามารถควบคุมจอแสดงผลที่ต่างกันได้ทุกชนิด ดังนั้นจึงต้องมีการกำหนดโหมดการทำงานของการ์ด TARGA ให้ตรงกับชนิดของจอที่ใช้แสดงผล

โหมดการทำงานที่ต่างกันของการ์ด TARGA จะหมายถึงการกำหนดค่าความแตกต่างของจอชนิดต่าง ๆ เช่น จำนวนจุดภาพในแนวนอน (Horizontal Resolution) , จำนวนจุดภาพในแนวตั้ง (Vertical Resolution) , อัตราส่วนมุมมองของจอภาพ (Aspect Ratio) , จำนวนบิตต่อจุดภาพ (Pixel Depth) , รูปแบบการ Scan ภาพ (Display) ซึ่งค่าของโหมดการทำงานต่าง ๆ ได้แสดงไว้ดังตารางที่ 5.1

ในส่วนที่เราใช้งานอยู่นี้การ์ด TARGA จะทำงานอยู่ในโหมดที่ 2 ซึ่งมีจำนวนจุดภาพในแนวนอน (Horizontal Resolution) = 512 จุด มีจำนวนจุดภาพในแนวตั้ง (Vertical Resolution) = 400 จุด มีจำนวนบิตต่อจุดภาพ (Pixel Depth) = 16 บิตต่อจุดภาพ (bit per pixel) และมีการ scan ภาพแบบ Non - Interlaced ซึ่งมีรูปแบบดังภาพที่ 5.3.1.1

ตารางที่ 5.3.1.1 ค่าโหมดต่าง ๆ ของการ์ด TARGA

Mode	Horizontal Resolution	Vertical Resolution	Aspect Ratio	Pixel Depth	Display
1	512	400	1.071	16 b/p	Interlaced
2	512	400	1.042	16 b/p	Non-Interlaced
3	512	400	1.071	32 b/p	Interlaced
4	512	400	1.042	32 b/p	Non-Interlaced
5	512	476	1.269	16 b/p	Interlaced
6	512	476	1.240	16 b/p	Non-Interlaced
7	512	476	1.269	32 b/p	Interlaced
8	512	476	1.240	32 b/p	Non-Interlaced
9	512	486	1.266	16 b/p	Interlaced
10	512	486	1.266	16 b/p	Non-Interlaced
11	512	486	1.266	32 b/p	Interlaced
12	512	486	1.266	32 b/p	Non-Interlaced



รูปที่ 5.3.1.1 แสดงการทำงานของการ์ด TARGA โหมดที่ 2

จากที่กล่าวมาจะเห็นว่าใน 1 จุดภาพจะมีค่าเท่ากับ 16 บิต โดยมี 1 บิตเป็นบิตควบคุมและอีก 15 บิตที่เหลือเป็นค่าของสีต่าง ๆ นั่นคือใน 1 จุดภาพสามารถแสดงสีที่ต่างกันถึง 2 ยกกำลัง 15 สี

5.3.2 พื้นฐานการเขียนโปรแกรมควบคุมการ์ด TARGA

ก่อนที่เราจะเขียนโปรแกรมควบคุมการ์ด TARGA นั้น ก่อนอื่นเราจะต้องรู้พื้นฐานการทำงานของการ์ด TARGA และการเข้าถึงการ์ด TARGA เสียก่อน

ในการเขียนโปรแกรมควบคุมการ์ด TARGA นี้เราจะอาศัย TARGA Graphics Toolkit (TGT) ซึ่งเป็น library ของ Software routines ซึ่งประกอบไปด้วย Function ต่าง ๆ ที่ใช้ในการสร้าง Graphics ต่าง ๆ เช่น การวาดเส้นตรง , การวาดวงรี , การวาดสี่เหลี่ยม เป็นต้น นอกจากนี้ TGT ยังทำหน้าที่ในการ Interface กันระหว่างการ์ด TARGA และ MS-DOS อีกด้วย

ในการที่ติดต่อระหว่างการ์ด TARGA กับ CPU นั้นเราจะต้องมีการกำหนดตำแหน่งของ I/O และ memory location ของการ์ด TARGA ให้กับ CPU ได้รับรู้ ซึ่งเราสามารถทำได้โดยการเซ็ต device driver ของ MS-DOS ซึ่งเราจะต้องทำการ install ไฟล์ที่ชื่อ TARGAP.SYS แล้วเข้าไปแก้ไฟล์ CONFIG.SYS ดังตัวอย่างนี้

```
DEVICE=C:\TARGA\TARGAP.SYS
```

Header Files ที่ใช้ในการควบคุมการ์ด TARGA

ในการเขียนโปรแกรมด้วย Borlandc ++ เพื่อที่จะประยุกต์การทำงานของ TGT เราต้องอาศัย Header Files เหล่านี้ซึ่งจะรวบรวม structure definitions ต่าง ๆ ที่จำเป็นสำหรับ routines ต่าง ๆ ของ TGT โดย Header Files ได้ถูกจัดไว้ดังนี้

TARGAPL.H - กำหนดค่าความสัมพันธ์ระหว่างการ์ด TARGA กับ DOS driver

TARGRAF.H - กำหนดค่าความสัมพันธ์ขององค์ประกอบ Griphics ใน TGT

TAG.H - กำหนดค่าความสัมพันธ์ไฟล์ภาพ .TGA

FIXEDPT.H - กำหนดค่าความสัมพันธ์ของตำแหน่งข้อมูลชนิดต่าง ๆ

ในการเริ่มต้นโปรแกรมเราจะต้อง #include<targraf.h> ที่ต้นโปรแกรม ซึ่งค่าความสัมพันธ์ต่าง ๆ ที่สำคัญได้ถูกรวมอยู่ในไฟล์นี้หมดแล้ว

Library Files

ใน TGT จะประกอบไปด้วย library ที่สำคัญ 2 ตัวคือ LTPLIB.LIB และ TARGRAF.LIB ซึ่งมีหน้าที่ต่างกันดังนี้

LTPLIB.LIB (Large model Targa Plus LIBrary) library นี้จะทำหน้าที่เกี่ยวกับการ Interface ระหว่างการ์ด TARGA กับ DOS device driver ซึ่งจะมีความสามารถในการอ่านและเขียนรีจิสเตอร์ที่ใช้ควบคุมการ์ด TARGA เกี่ยวกับค่าสีในแต่ละ pixel

TARGRAF.LIB จะจัดการเกี่ยวกับ color control , ค่าตัวแปรต่าง ๆ ใน Graphics และการเขียนข้อความ

ลักษณะของสีในการ์ด TARGA

สีถูกให้ความหมายภายใน TGT ไว้แตกต่างกัน 2 อย่าง มีโครงสร้างเรียกว่า grafport ซึ่งสามารถให้นิยามได้ทั้ง foreground drawing color และ background color background color ถูกใช้ในการอธิบายค่าที่ถูกเขียนเข้าไปใน frame buffer ในระหว่าง erase operation. function อย่างเช่น EraseOval และ EraseRect ใช้ background color เมื่อปฏิบัติขั้นตอนกระบวนการของมัน foreground color ถูกใช้โดย functions เหล่านี้ ได้แก่ FrameRect,LineTo,StrokeOval และ PaintRect ดังที่กล่าวไว้ แต่ละสีสามารถ

อธิบายได้แตกต่างกัน 2 แบบ ไม่ว่า foreground color หรือ background color ค่าเหล่านี้จะแทนด้วยค่า 32 bit ค่า 32 bit ทั้งหมดจะถูกใช้ ถ้า TGT กำลังทำงานใน mode 32 bit ต่อ pixel จะใช้เพียง least significant 16 เท่านั้น ใน mode 16 bit ต่อ pixel และจะใช้เพียง least significant 8 เท่านั้น ใน mode 8 bit ต่อ pixel ค่าเหล่านี้จะตามการแทน pixel ซึ่งถูกอธิบายโดย hardware สำหรับแต่ละ mode เหล่านี้ เช่น ใน mode 16 bit ต่อ pixel บิตที่ 15 (MSB ของค่า 16 bit) เป็น overlay (alpha) control bit แล้วตามด้วย red component data 5 bit,green component data 5 bit,blue component data 5 bit ใน mode 32 bit per pixel เราจะมี component information 8 bits สำหรับแต่ละ pixel ในลำดับ alpha,red,green, blue (จาก most significant ไป least significant)

การคัดสีใน fashion แบบนี้มักจะสับสน ดังนั้นจึงมีโครงสร้าง data ที่แยกออกมาในการอธิบายค่า component ของ 1 สี ในความลึก pixel (pixel depth) ที่ไม่ขึ้นกับ fashion โครงสร้าง data RGB Color ให้คำจำกัดความสีในรูปขององค์ประกอบสีแดง,เขียว,และ น้ำเงิน ของมันแต่ละองค์ประกอบเป็นค่า normalized 16 bit เนื่องจากค่าถูก normalize การเปลี่ยนจาก RGB Color เป็น pixel (index) value จะใช้ประโยชน์ของ MSB ขององค์ประกอบแรก ตัวอย่างเช่น ถ้าเราเปลี่ยนสี RGB เป็น 16 bit color (pixel) value, จะใช้เพียงค่า most significant 5 bits ของแต่ละองค์ประกอบ RGB เท่านั้น ส่วน Overlay หรือ alpha ของ color (pixel) value จะนำมาจากอีกสมาชิกหนึ่งของโครงสร้าง port ,16 bit foreground alpha value หรือ 16 bit background alpha value สำหรับตัวอย่างนี้ในการเปลี่ยนจะใช้เพียง normalized alpha value เท่านั้น functions จะถูกใช้ set foreground และ background color ในอีกรูปแบบหนึ่ง และใช้เปลี่ยบระหว่างรูปแบบองค์ประกอบทั้งหมด foreground color สามารถถูก SetFore Color แต่ ผู้เรียกจะต้องรู้ความลึก pixel ปัจจุบัน เพื่อให้ได้รับผลตามต้องการ foreground color สามารถถูก set โดยไม่ขึ้นกับความลึก pixel ด้วย โดยการใช้ SetRGBForeColor และ SetForeAlpha และ การกระทำเหล่านี้จะเปลี่ยน ค่าองค์ประกอบที่ถูก normalize ให้เป็น foreground color value ที่เหมาะสม โดยมีขึ้นอยู่กับความลึก pixel ปัจจุบัน ให้เราพูดว่าเราต้องการสร้าง foreground color เป็นสีขาว โดยไม่คำนึงถึงความลึก pixel ซึ่งเราดำเนินอยู่ เราสามารถทำได้โดยลำดับ code เหล่านี้

```
RGBColor        rgbColor;
rgbColor.red=0xffff;
rgbColor.green=0xffff;
rgbColor.blue=0xffff;
```

SetRGBForeColor(rgbColor)

SetForeAlpha(0); /* insure alpha compnent is zero */

ถ้าเรารู้ว่ากำลังดำเนินการอยู่ที่ 32 bits per pixel เราสามารถทำผลเดียวกันนี้โดย

SetForeColor(0x00ffffffL);

แต่ถ้าเรากำลังใช้งานอยู่ที่ 16 bits per pixel การเรียกจะเป็น

SetForeColor(0x7fffL);

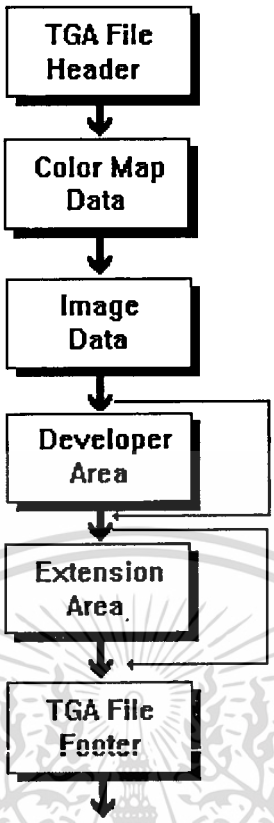
เนื่องจากการจำกัดความยาวของ RGB Color ถูก normalize แต่ล่ะองค์ประกอบจะเป็น 0xff00 และผลเหมือนเดิม เพราะ บิตจำนวนมากที่ใช่จะเกิดขึ้นที่ 32 bit per pixel depth (ใช้ most significant 8 bits ของแต่ล่ะองค์ประกอบ) ฟังก์ชั่นสองฟังก์ชั่นถูกใช้ในการเปลี่ยนระหว่าง pixel color value และ RGB Component value : RGB ToIndex และ IndexToRGB truncation error ทำให้การเปลี่ยนไม่สมมาตร การทดลองที่ดี คือ การนิยาม RGB Color ,เปลี่ยนมันเป็น pixel (index) value โดยใช้ RGBToIndex แล้วเปลี่ยนผลอันนั้นกลับเป็น RGB color โดยใช้ IndexToRGB ข้อแตกต่างระหว่าง RGB color จะแสดง error ที่เกิดขึ้นระหว่างการเปลี่ยน

5.3.3 การเขียนโปรแกรมเกี่ยวกับ Truevision TGA files

ในการเก็บภาพจากหน้าจอเพื่อนำมาทำการประมวลผล เราจะเก็บค่าสีต่าง ๆ และจัดรูปแบบให้อยู่ใน Format ของ TGA files (มีนามสกุลเป็น .TGA) ทั้งนี้เนื่องจาก Format ของ TGA files มี option และ mode ต่าง ๆ มากมายในการนำไปประยุกต์ใช้ , TGA files รองรับภาพ images ที่มีค่าของสีตั้งแต่ 1 , 8 , 16 , 24 และ 32 บิต และที่สำคัญ TGA files สามารถบอกสถานะของตัวมันเองได้ว่ามีการ Compress หรือไม่ ซึ่งจะเป็นประโยชน์อย่างมากในการส่งข้อมูลภาพผ่าน LAN นอกจากนี้มันยังสามารถเก็บค่าเริ่มต้นและจุดสิ้นสุดของ Images ได้ และยังสามารถกลับภาพไปทางซ้าย ขวาได้

โครงสร้างของ Targa Files

ในการเขียนโปรแกรมควบคุม TGA files เราจำเป็นที่จะต้องรู้โครงสร้างของ TGA files เพื่อที่จะนำไปประยุกต์ใช้ในการประมวลผลได้



รูปที่ 5.3.3.1 โครงสร้างของ TGA files

จากรูที่ 5.3.3.1 จะเห็นว่าโครงสร้างของ TGA files จะแบ่งออกเป็นส่วน ๆ โดยแต่ละส่วนจะมีหน้า ที่แตกต่างกันออกไป ซึ่งต่อไปนี้จะขออธิบายเฉพาะส่วนที่สำคัญเท่านั้น ได้แก่

ส่วนของ Header File

และส่วนของ Color Map Data

ใน file ที่มีนามสกุล " .TGA " หรือมี format เป็น TGA files นั้น ในส่วนต้นของ file นี้ ค่าแต่ละค่าจะบ่งบอกถึงลักษณะเฉพาะของ file นั้น ๆ ตามโครงสร้างข้างล่างนี้

```
typedef struct {
        char identsize;
        char colormaptype;
        unsigned int colormapstart;
        unsigned int colormaplenght;
        char colormapbits;
        unsigned int xstart;
```

```
unsigned int ystart;
unsigned int width,depth;
char bits;
char descriptor;
} TGAHEAD ;
```

ในโครงสร้างของ TGAHEAD ฟิลด์ identsize จะแสดงจำนวนไบต์ที่แบ่งส่วนของ Header กับส่วนอื่น ๆ ส่วนใหญ่แล้วจะมีค่าเป็น 0

ฟิลด์ colormaptype จะบอกชนิดของ color map ที่ใช้ใน file นั้น โดยจะมีค่าเป็น 0 เมื่อเป็น file ที่ใช้กับ Monochrome หรือ RGB color information และจะเป็น 1 ถ้าเป็น color map และเมื่อเป็น color map ฟิลด์ colormapstart จะระบุค่าของ color map index ของ palette ข้อมูลสีอันแรก ส่วนฟิลด์ colormaplength จำนวนสีที่ทั้งหมดที่จะใส่ลงไป ซึ่งจะมีค่าตั้งแต่ 0 ถึง 256 นั่นคือในหนึ่ง palette สีจะมีจำนวนสีได้มากที่สุด 256 สี

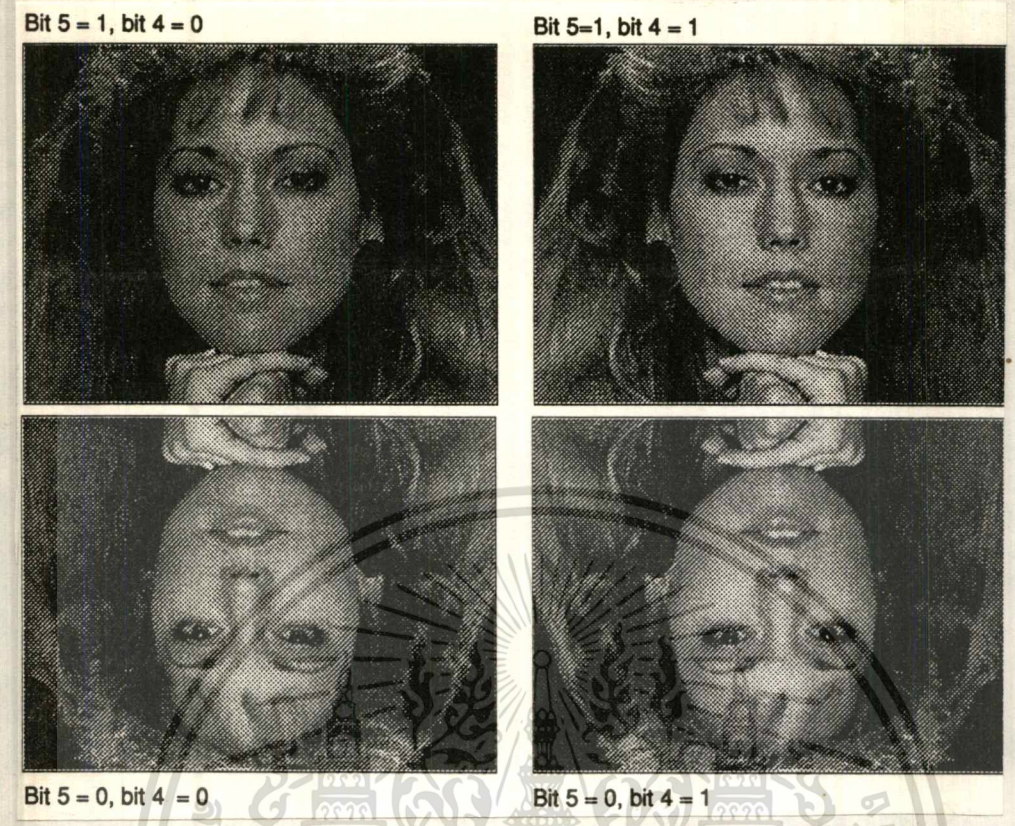
ฟิลด์ imagetype จะบอกชนิดของการเก็บ file นั้น ๆ ซึ่งมีค่าต่าง ๆ ดังนี้

1 - Uncompressed palette-driven image
2 - Uncompressed RGB image
3 - Uncompressed monochrome image
9 - Run-length encode palette-driven image
10 - Run-length encode RGB image
11 - Run-length encode monochrome image

ฟิลด์ xstart,ystart จะบอกค่าตำแหน่งของ image ที่จะไปปรากฏบน screen

ฟิลด์ width,depth จบอกค่ามิติของ image ในแต่ละ pixel ค่าของบิตจะบ่งบอกถึงจำนวนบิตของสีที่ใช้ ซึ่งจะมีค่า 1 , 8 , 16 , 24 หรือ 32

ฟิลด์ descriptor จะเป็นค่า flags ที่จะบ่งชี้ว่าข้อมูล image นั้นจะมีลักษณะอย่างไร โดยที่มีค่าเพียง 2 บิตที่มีความสำคัญ ซึ่งสามารถอธิบายได้ดังรูปที่ 5.3.3.2



รูปที่ 5.3.3.2 รูปแสดงหน้าที่ของฟิลด์ descriptor

เมื่อเราทราบโครงสร้างของ Header Files แล้วเราก็สามารถจะอ่านค่าต่าง ๆ ของ TGA files ได้และยังสามารถกำหนดรูปแบบต่าง ๆ ของ file ได้อีกด้วย

```
17C8:0100  1A 00 0A 00 00 00 00 00-00 00 00 00 80 00 80 00   ................
17C8:0110  18 00 54 72 75 65 76 69-73 69 6F 6E 28 52 29 20   ..Truevision(R)
17C8:0120  53 61 6D 70 6C 65 20 49-6D 61 67 65 87 00 00 FF   Sample Image....
17C8:0130  87 00 FF 00 87 FF 00 00-87 00 00 00 87 00 00 FF   ................
17C8:0140  87 00 FF 00 87 FF 00 00-87 FF FF FF 87 00 00 FF   ................
17C8:0150  87 00 FF 00 87 FF 00 00-87 00 00 00 87 00 00 FF   ................
17C8:0160  87 00 FF 00 87 FF 00 00-87 FF FF FF 87 00 00 FF   ................
17C8:0170  87 00 FF 00 87 FF 00 00-87 00 00 00 87 00 00 FF   ................
```

7-5 Peeking at the beginning of a Truevision Targa test file.

รูปที่ 5.3.3.3 ตัวอย่างของ Header Files

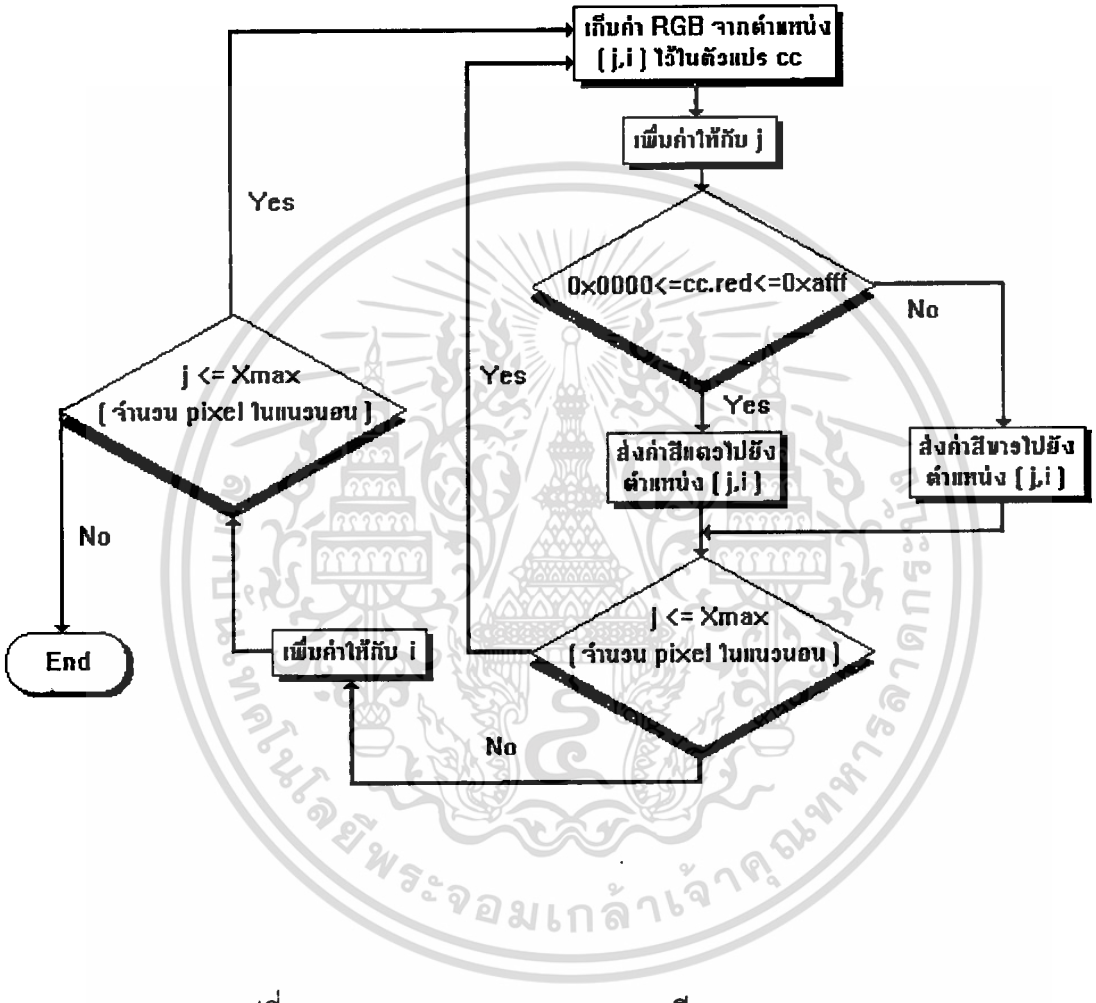
Byte#	Data (7 6 5 4 3 2 1 0)	Details
0	ID Length	# of bytes used for Image ID, max 255
1	Color Map Type	0: no color map, 1: color map in file
2	Image Type	7 types defined, specifies color and compression schemes
3–4	Color Map Specification First Entry Index	Color table index to associate with first entry of file color map
5–6	Color Map Specification Color map Length	Total number of color map entries in file
7	Color Map Specification Color map Entry Size	Number of bits per entry, typically 15, 16, 24, or 32 bits
8–9	*X*-origin of Image	Horizontal coordinate for the lower-left corner of the image
10–11	*Y*-origin of Image	Vertical coordinate for the lower-left corner of the image
12–13	Image Width	In pixels
14–15	Image Height	In pixels
16	Pixel Depth	Bits per pixel, typically 8, 16, 24, or 32
17	0 | 0 | Origin | Alpha bits	See below
18–	Image ID	Optional ASCII text, 0–255 bytes

Origin — Bottom left, bottom right, top left, top right;
Alpha bits — Attribute bits per pixel, either Alpha channels bits or overlay bits

รูปที่ 5.3.3.4 TGA File Header

5.3.4 การเขียนโปรแกรมประมวลผลภาพ

ในการเขียนโปรแกรมในส่วนนี้ จะใช้หลักเกณฑ์คล้ายกับการเขียนโปรแกรมควบคุมการ์ด VGA จะต่างกันตรงที่เราจะใช้ชุดคำสั่งของ TGT ในการควบคุมการ์ด TARGA ฉะนั้นในส่วนนี้จะขออธิบายเฉพาะบางส่วนของโปรแกมโดยจะอธิบายเป็น Flow Chart ประกอบกับโปรแกรมที่อยู่ในส่วนภาคผนวก

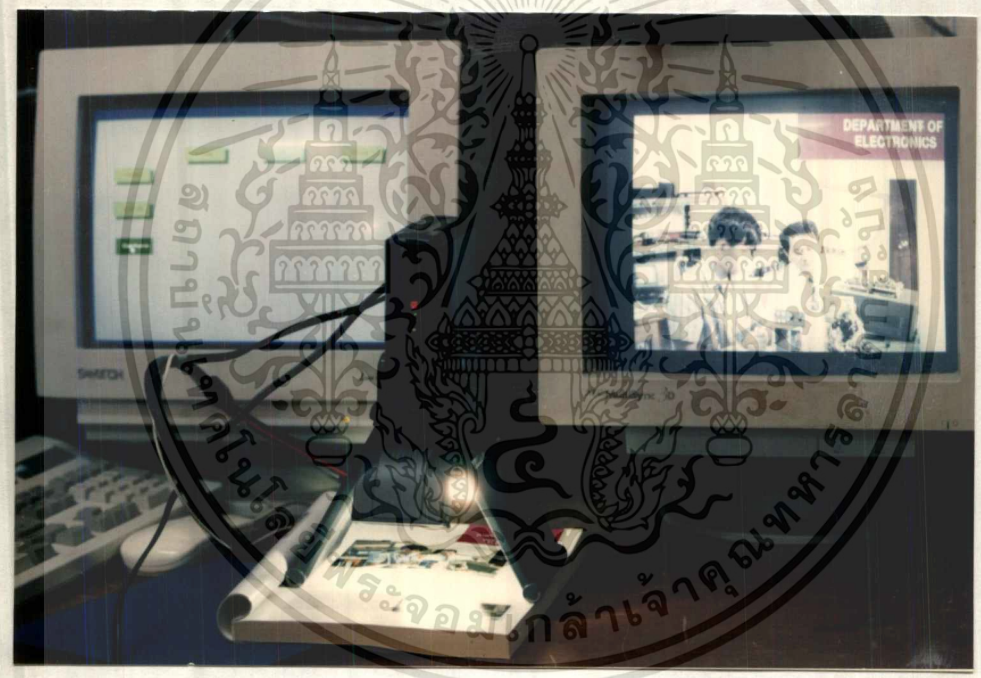


รูปที่ 5.3.4.1 Flow Chart แสดงการแยกสีแดงออกจากภาพ

บทที่ 6 ผลการทดลอง

6.1 ส่วนของการประมวลผลภาพ

ในการทดลองถ่ายภาพจากกล้อง V.D.O. เพื่อนำสัญญาณภาพที่มีลักษณะเป็น Interactive ไปแสดงบนจอภาพที่ควบคุมด้วยการ์ด TARGA ได้ผลออกมาตามรูปที่ 6.1.1 ซึ่งจะได้ภาพที่เป็นลักษณะ Interactive ตามต้องการ ส่วนรูปที่ 6.1.2 แสดงถึงการจับภาพในช่วงเวลาขณะใดขณะหนึ่งซึ่งถ้าเราสามารถตั้งกล้องให้อยู่นิ่งได้เราก็จะได้ผลของภาพที่คมชัดดังรูปที่ 6.1.2

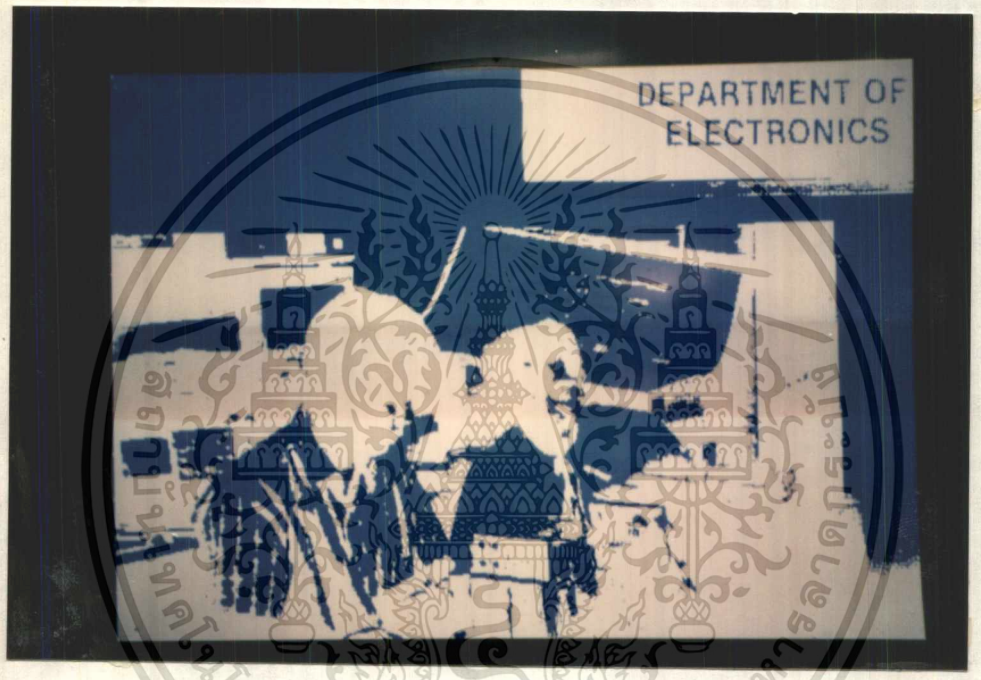


รูปที่ 6.1.1 แสดงการถ่ายภาพจาก V.D.O. ลงสู่การ์ด TARGA

รูปที่ 6.1.2 แสดงการจับภาพให้หยุดนิ่ง ณ. ช่วงเวลาใด ๆ

ส่วนในรูปที่ 6.1.3 แสดงถึงการแยกองค์ประกอบสีจากภาพที่ได้จากรูป 6.1.2 ออกเป็น สีแดง สีเขียว สีน้ำเงิน ซึ่งจะเห็นได้ว่าส่วนใดในรูปที่ 6.1..2 ที่มีสีดำนั่นคือมีองค์ประกอบของสีทั้งสามน้อย เราจึงตัดค่าสีนั้นทิ้งไป ดังจะเห็นในรูปที่ 6.1..3 ในส่วนที่เป็นสีขาว

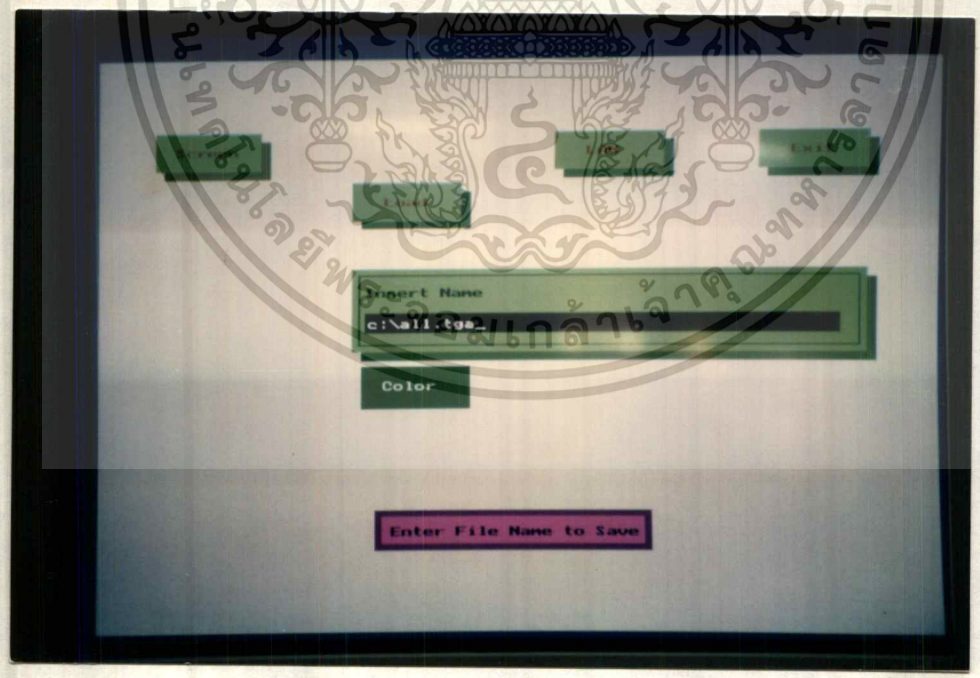
รูปที่ 6.1.3 การแยกองค์ประกอบสี



รูปที่ 6.1.3 การแยกองค์ประกอบสี (ต่อ)

รูปที่ 6.1.4 แสดงถึงการเลือกบันทึกรูปภาพด้วยขนาดและตำแหน่งที่ต้องการ ซึ่งจะได้ผลดังรูปที่ 6.1.6

รูปที่ 6.1.4 แสดงการเลือกตำแหน่งและขนาดการบันทึก



รูปที่ 6.1.5 แสดงเมนู (Menu) การควบคุม

รูปที่ 6.1.6 ภาพที่ใช้ในการเตรียมส่งผ่านข้อมูล

6.2 การทดลองส่งภาพผ่านเครือข่ายท้องถิ่น

ในการส่งภาพตามรูปที่ 6.1.6 ซึ่งมีขนาด832,150 ไบท์ ส่งผ่านเครือข่ายท้องถิ่นแล้วผลปรากฏว่า ทางด้านผู้รับสามารถได้รับข้อมูลขนาด 832,150 ไบท์ ซึ่งมีขนาดเท่ากับต้นฉบับที่ส่งมาจากด้านผู้ส่ง และเมื่อนำไปอ่านด้วยการ์ด TARGA ได้ผลดังรูปที่ 6.2.1

รูปที่ 6.2.1 ภาพที่ได้จากส่งมาจากผู้ส่งโดยผ่านเครือข่ายท้องถิ่น

บทที่ 7
สรุปและวิจารณ์ผลการทดลอง

7.1 ส่วนประมวลผลภาพ

ในส่วนการประมวลผลภาพด้วยการ์ด TARGA ส่วนใหญ่ให้ผลการทดลองเป็นไปตามความคาดหวัง เพราะส่วนฮาร์ดแวร์ของการ์ด TARGA มีการออกแบบให้ควบคุมได้สะดวก ปัญหาที่มีส่วนใหญ่คือ การศึกษาการทำงานของการ์ด TARGA เพราะเนื่องจากการ์ด TARGA มีความสามารถในการทำงานสูง จึงมีความซับซ้อนในการเขียนโปรแกรมควบคุม เพื่อให้การ์ด TARGA ทำงานได้ตามความต้องการ

7.2 ส่วนรับส่งข้อมูล

ในส่วนการรับส่งข้อมูลสามารถทำการส่งได้ครบถ้วน มีปัญหาที่ก่อนการส่งจะต้องมีการส่งข้อความไปยังด้านผู้รับ เพื่อให้ผู้ปฏิบัติงานโปรแกรมรับ ถึงจะทำการส่งข้อมูลไปได้ แต่ถ้าหากว่า ทางด้านผู้รับปฏิบัติงานโปรแกรมรับ แต่ทางด้านผู้ส่งไม่ส่งมาอาจเนื่องจากปัญหาบางประการจะทำให้ทางด้านผู้รับไม่สามารถออกจากโปรแกรมรับได้ เพราะ การรอรับข้อมูลจะเป็นการทำให้อินเตอรัปไม่มีผล (disable) ดังนั้นจึงไม่ยืดหยุ่นเท่าที่ควร และการส่งข้อความจะไม่สามารถปรากฏขึ้นหน้าจอ ถ้าอยู่ในโหมดกราฟฟิกส์

กิตติกรรมประกาศ

เนื่องจากว่าอุปกรณ์ในการทดลองนั้น ค่อนข้างมีราคาสูงและ เรื่องนี้ไม่ได้วิทยานิพนธ์เล่มใดทำการศึกษามาก่อน ดังนั้น วิทยาพนธ์เล่มนี้จะไม่สามารถสำเร็จลงได้ถ้าไม่ได้รับการช่วยเหลือทางด้านอุปกรณ์ คำแนะนำ ข้อมูล และวิธีการใช้จาก

รศ. ดร. มนัส สังวรศิลป์

อ. เทอดศักดิ์ ลิ่วหาทอง

ขอขอบคุณผู้ที่ให้คำแนะนำ และ เอื้อเฟื้ออุปกรณ์ในการพิมพ์ตลอดเวลาที่ได้จัดทำวิทยานิพนธ์เล่มนี้ คือ

ฝ่ายข้อมูลและบริการลูกค้า (STC)

ฝ่ายคอมพิวเตอร์ (FIO , ICS)

บริษัท เชลล์แห่งประเทศไทย จำกัด

ขอขอบคุณในความกรุณาเป็นอย่างสูง

นาย คงพันธ์ ฉมารัตน์

นาย ปรัชญา ปัณฑุยากร

ผู้ศึกษา

บรรณานุกรม

1. **Network Programming in C.** Barry Nance,QUE Corporation.,1990.
2. **Programmer's Guide to NetWare,** Charles G.Rose, MaGraw-Hill Inc.,1990.
3. **Graphics File Formats reference and guide.** C.Wayne Brown and Barry J.Shepherd., Prentice Hall Inc.,1995.
4. **Supercharged Bitmapped Graphics.** Steve Rimmer, MaGraw-Hill Inc.,1992.
5. **PC System Programming.** Michael Tischer, Abacus Inc.,1991.
6. **ระบบเครือข่ายคอมพิวเตอร์** LAN,อัครเสน สมุทรผ่อง,จักร พิชัยศรทัต,บริษัท ซีเอ็ดยูเคชั่น จำกัด.,2535.
7. **แอดวานซ์ดอสด้วยภาษาซี,**วิริ พงษ์แจ้ง,กิตติ องค์คุณารักษ์,บริษัท ซีเอ็ดยูเคชั่น จำกัด., 2538

ภาคผนวก

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<dos.h>
#include<string.h>
#include<io.h>
#include<fcntl.h>
#include<stdlib.h>
#include<ctype.h>
#include<math.h>
#include<dir.h>
#include<alloc.h>
#include"targraf.h"

#define  LEFT   0
#define  RIGHT  1          // traga mouse
#define  BOTH   2

#define  ENABLE  1
#define  DISABLE 0

#define  Menu_Y_1()  Move(xmenu,ymenu,3,0)
#define  Menu_Y_2()  Move(xmenu+skip,ymenu,3,3)
#define  ClearWin()  Clearpos(xwin,ywin,len,30)

void Initial(void);
void Fillbox(int x1,int y1,int x2,int y2,int color);
void Title(int back);
void Sound(int freq,int sec);
int Option1(void);
void Choice(int x,int y,char text[10]);
void Choose(int x,int y,char text[10]);
void Clearpos(int x,int y,int w,int h);
void Full_Menu(void);
void Main_Menu(void);
void Mouse(void);
void ClearMouse(void);
void Move(int x,int y,int num,int init);
void InData(int x,int y,char text[10]);
void Pop(int x,int y,char text[20]);
void WinText(char text[50]);
void ClearTga(void);
void TextTga(char text[20],int posx,int posy,int num);
void MouseTga(void);
void MouseBoxTga(void);
void BoxTga(int x1,int y1,int x2,int y2);

int  Xmax,Xmin,Ymax,Ymin;
int  Height,Width;       // TARGA
int  Xpos,Ypos;          // position at traga
int  X,Y;                // traga mouse
int  Botton;

int  Xbox1,Ybox1,Xbox2,Ybox2;//select rectangle

Rect  saverec;
Rect  r;
Point p={0,0};
Point P;

char font[14][12] = {
                        "CRT.TSF",
                        "CRTND.TSF",
                        "GOTHIC.TSF",
                        "GREEK.TSF",
                "ROMAN.TSF",
                                "ROMANI.TSF",
                        "ROMANIM.TSF",
                                "ROMANISM.TSF",
                        "ROMANSM.TSF",
                        "SCRIPT.TSF",
                        "SCRIPTF.TSF",
                        "SYMBOL.TSF",
                        "SYSTEM.TSF",
                                "SYSTEMB.TSF"
                };
```

```
RGBColor   white = { 0xffff, 0xffff, 0xffff };


char option[4][8];
union REGS regs;
int op;

char buffer[50];

char menu[3][10] = {  "  Screen",     //
                                        "  File",    //
                                        "  Exit"       //    main menu
                                } ;           //

char ymenu1[6][10] = {  "  Clear",
                                        "  V.D.O",
                                        "  Capture",      //  Y-menu-1
                                        "  Load",
                                        "  Save ",
                                        "  Color "
                                       };

int xmenu=50;
int ymenu=80;        // main menu
int skip=150;

int width=80;        // size of choice
int hight=30;

int xms,yms;          // mouse

int step = 40;   //  far of menu y

int capture=DISABLE;
int save   =ENABLE;
int clear  =ENABLE;
int vdo    =ENABLE;
int load   =ENABLE;
int color  =DISABLE;

int xwin,ywin,len;          //window text

void main()
{
                /*                      {
                                        int result;
                                        result = spawnlp(P_WAIT,"TMODE.EXE","","4","6","0","0","0",NULL);
                                        if(result == -1)
                                        {
                                                printf("I can't run tmode 4 6 0 0 0");
                                                exit(0);
                                        }
                                        printf("I can run tmode 4 6 0 0 0");
                }*/

                if( InitGraphics() < 0 )
                {
                                puts("targa driver not available");
                                exit(-1);
                }
                EnableDisplay();
                SetNoninterlaced(true);
                EnableLUT();
                SetDisplayMode(dispModeIndependent);
                GetOrigin( &Xmin, &Ymin );
                GetPortSize( &Width, &Height );
                SetRect( &r, Xmin, Ymin , Xmin+Width, Ymin+Height );
                Ymax = Ymin+Height;
                Xmax = Xmin+Width;
                EraseRect(r);
                GetPic("pong2.tga",0,p);
                TextTga("Prachya Pantuyakorn",55,Ymax/2,5);
                //              getch();
                 //             Clear()
 Initial();
  Option1();
   Title(op);
   Main_Menu();
   closegraph();
```

```
}

void Initial(void)
{
  /* request auto detection */
  int gdriver = DETECT, gmode, errorcode;

  /* initialize graphics mode */
  initgraph(&gdriver, &gmode, "");

  /* read result of initialization */
  errorcode = graphresult();

  if (errorcode != grOk)  /* an error occurred */
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
                getch();
                exit(1);            /* return with error code */
  }

 }

void Fillbox(int x1,int y1,int x2,int y2,int color)
{
 setcolor(color);
 setfillstyle(1,color);
 bar(x1,y1,x2,y2);
}

void Title(int back)
{  int i,x;
//  Sound(2000,1);
  for(i=0;i<getmaxx()/2;i++)
   { Fillbox(0,0,i*2,20,BLUE);
    Fillbox(getmaxx()-(i*2),getmaxy()-20,getmaxx(),getmaxy(),BLUE);
   }
  //  Sound(3000,1);
              x=getmaxx()/10;
    for(i=0;i<getmaxy()/12+5;i++)
      {
                Fillbox(0,i*12,x,20+(i*12),BLUE);
                Fillbox(0,0,x,i*12,back);
                Fillbox(x*2,i*12,x*3,20+(i*12),BLUE);
                Fillbox(x*2,0,x*3,i*12,back);
                Fillbox(x*4,i*12,x*5,20+(i*12),BLUE);
                Fillbox(x*4,0,x*5,i*12,back);
                Fillbox(x*6,i*12,x*7,20+(i*12),BLUE);
                Fillbox(x*6,0,x*7,i*12,back);
                Fillbox(x*8,i*12,x*9,20+(i*12),BLUE);
                Fillbox(x*8,0,x*9,i*12,back);
                /*********************************/
                Fillbox(x,getmaxy()-(20+i*12),x*2,getmaxy()-(i*12),BLUE);
                Fillbox(x,getmaxy()-(i*12),x*2,getmaxy(),back);
                Fillbox(x*3,getmaxy()-(20+i*12),x*4,getmaxy()-(i*12),BLUE);
                Fillbox(x*3,getmaxy()-(i*12),x*4,getmaxy(),back);
                Fillbox(x*5,getmaxy()-(20+i*12),x*6,getmaxy()-(i*12),BLUE);
                Fillbox(x*5,getmaxy()-(i*12),x*6,getmaxy(),back);
                Fillbox(x*7,getmaxy()-(20+i*12),x*8,getmaxy()-(i*12),BLUE);
                Fillbox(x*7,getmaxy()-(i*12),x*8,getmaxy(),back);
                Fillbox(x*9,getmaxy()-(20+i*12),getmaxx(),getmaxy()-(i*12),BLUE);
                Fillbox(x*9,getmaxy()-(i*12),getmaxx(),getmaxy(),back);

      }
 Fillbox(0,0,getmaxx(),20,BLUE);
 Fillbox(0,0,5,getmaxy(),BLUE);
 Fillbox(getmaxx()-5,0,getmaxx(),getmaxy(),BLUE);
 Fillbox(0,getmaxy()-5,getmaxx(),getmaxy(),BLUE);

 }

 void Sound(int freq,int sec)
 {
   sound(freq);
   delay(sec);
   nosound();
 }
```

```
int Option1(void)
{
 int x,y,loop;

 regs.x.ax = 0;
 int86(0x33,&regs,&regs);
 strcpy(option[0],"WHITE");
 strcpy(option[1],"YELLOW");
 strcpy(option[2],"GRAY");
 strcpy(option[3],"EXIT");
 outtextxy(50,200,"Select Color of Screen");
 rectangle(80,220,170,320);
 outtextxy(100,235,option[0]);
 outtextxy(100,250,option[1]);
 outtextxy(100,265,option[2]);
 outtextxy(100,280,option[3]);
 if (regs.x.ax == 0)
 {
   outtextxy(getmaxx()/2,getmaxy()/2,"No Mouse Available");
              exit(1);
 }
 regs.x.ax=1;
 int86(0x33,&regs,&regs);
  do
    { loop = 1;
              regs.x.ax = 3;
              int86(0x33,&regs,&regs);
              x = regs.x.cx;
              y = regs.x.dx;
              if(regs.x.bx&1)
               {
                if(x>80&&x<170)
                 {
                              if(y>235 && y<250)
                                 { op=WHITE;loop=0;}
                              if(y>250 && y<265)
                                        { op=YELLOW;loop=0; }
                              if(y>265 && y<280)
                                { op=LIGHTGRAY;loop=0; }
                              if(y>280 && y<295)
                                        { closegraph();
                                   exit(1); }

                              }
               }
    } while(loop!=0);
 regs.x.ax = 2;
 int86(0x33,&regs,&regs);

 return(op);
}
void Choice(int x,int y,char text[10])
{
   Fillbox(x+7,y+7,x+7+width,y+7+hight,GREEN);
   Fillbox(x,y,x+width,y+hight,LIGHTGREEN);
   setcolor(RED);
              outtextxy(x,y+12,text);
}
void Choose(int x,int y,char text[10])
{
              Clearpos(x,y,width+7,hight+7);
   Fillbox(x+7,y+7,x+width+7,y+hight+7,GREEN);
   setcolor(WHITE);
              outtextxy(x+7,y+12+7,text);

}
void Clearpos(int x,int y,int w,int h)
{
    Fillbox(x,y,x+w,y+h,op);
}
void Full_Menu(void)
{
               Choice(xmenu,ymenu,menu[0]);
               Choice(xmenu+skip,ymenu,menu[1]);
               Choice(xmenu+(skip*2),ymenu,menu[2]);
}

void Main_Menu(void)
{
```

```
RGBColor cc;
RGBColor rgb;
int i,j;
int loop,loop1,loop2;
char *filename,c;
unsigned int odmd;
            Full_Menu();
regs.x.ax=1;
int86(0x33,&regs,&regs);
            do
              { loop = 1;
            Mouse();
            if(regs.x.bx&1)
                        {
                         if( yms>=ymenu && yms<=ymenu+hight )
                         {
                          if(xms>=xmenu && xms<=xmenu+width)
                                    {
                                     ClearMouse();
                                     Full_Menu();
                                     Choose(xmenu,ymenu,menu[0]);
                                     Clearpos(xmenu,ymenu+hight+10,300,300);
                                     //Sound(800,200);
                                     Menu_Y_1();
                                     xms=0;
                                     yms=0;
                                    if (capture==DISABLE)
                                    {
                                     Choose(xmenu,ymenu+(step*3)+(hight*2),ymenu1[2]);
                                    }       // END IF CAPTURE DISABLE
                                    if (vdo==DISABLE)
                                    {
                                     Choose(xmenu,ymenu+(step*2)+(hight),ymenu1[1]);
                                    }       // END IF V.D.O DISABLE
                                    if (clear==DISABLE)
                                    {
                                     Choose(xmenu,ymenu+(step*1),ymenu1[0]);
                                    }       // END IF CLEAR DISABLE

                                    do
                                    {
                                     Mouse();
                                     loop1=1;
                                     if(regs.x.bx&1)
                                     {
                                              if (xms>=xmenu && xms<=xmenu+width)
                                              {
                                               if (yms>=ymenu+step && yms<=ymenu+step+hight&&clear==ENABLE)
                                               {
                                               ClearMouse();
                                               xms=0;
                                               yms=0;
                                               Choose(xmenu,ymenu+step,ymenu1[0]);
                                               ClearTga();
                                               //Sound(800,200);
                                               loop1=0;
                                               }    // BUTTON Y 1-1
                                               if (yms>=ymenu+(step*2)+hight && yms<=ymenu+(step*2)+(2*hight)&&vdo==ENABLE)
                                               {
                                               ClearMouse();
                                               xms=0;
                                               yms=0;
                                               Choose(xmenu,ymenu+(step*2)+hight,ymenu1[1]);
                                                        if(SelectLiveSource(2))
                                                        {
                                                        GetDisplayMode(&odmd);
                                                        SetNoninterlaced(false);
                                                        EnableGenlock();
                                                        SetDisplayMode(dispModeLive);
                                                        }       // END IF LIVE VDO

                                               //Sound(800,200);
                                                capture=ENABLE;
                                                vdo=DISABLE;
                                                clear=DISABLE;
                                                load=DISABLE;
                                                loop1=0;
                                               }                // BUTTON Y 1-2
                                               if (yms>=ymenu+(step*3)+(hight*2) && yms<=ymenu+(step*3)+(3*hight)&&capture==ENABLE)
```

```
                        {
                        ClearMouse();
                        xms=0;
                        yms=0;
                        Choose(xmenu,ymenu+(step*3)+(hight*2),ymenu1[2]);
                        //Sound(800,200);
                        GrabFrame();
                        SetDisplayMode(odmd);
                        DisableGenlock();
                         capture=DISABLE;
                         vdo=ENABLE;
                         clear=ENABLE;
                         save=ENABLE;
                         load=ENABLE;
                         loop1=0;
                        }    // BUTTON Y 1-3

                      } else loop1=0;     // END IF
                }       // END BUTTON LEFT  Y 1
               } while(loop1!=0);  // END DO MENU Y 1-1
               Clearpos(xmenu,ymenu,width*4,step*6);
               Full_Menu();
               }          // END IF MENU Y-1
if(xms>=xmenu+skip && xms<=xmenu+skip+width)
               { ClearMouse();
                Full_Menu();
                Choose(xmenu+skip,ymenu,menu[1]);
                Clearpos(xmenu,ymenu+hight+10,300,300);
                //Sound(800,200);
                Menu_Y_2();
                xms=0;
                yms=0;
               if (save==DISABLE)
               {
                Choose(xmenu+skip,ymenu+(step*2)+hight,ymenu1[4]);
               }       // END IF SAVE DISABLE
               if (load==DISABLE)
               {
                Choose(xmenu+skip,ymenu+(step*1),ymenu1[3]);
               }       // END IF SAVE DISABLE
               if (color==DISABLE)
               {
                Choose(xmenu+skip,ymenu+(step*3)+(2*hight),ymenu1[5]);
               }       // END IF COLOR DISABLE

               do
               {
                Mouse();
                loop2=1;
                if(regs.x.bx&1)
                {
                        if (xms>=xmenu+skip && xms<=xmenu+width+skip)
                        {
                         if (yms>=ymenu+step && yms<=ymenu+step+hight&&load==ENABLE)
                         {
                         ClearMouse();
                         xms=0;
                         yms=0;
                         Choose(xmenu+skip,ymenu+step,ymenu1[3]);
                         WinText("Enter File Name to Load");
                         InData(xmenu+skip,ymenu+step,"Insert Name");
                         ClearWin();
                         WinText("Click 'Left' for New Position 'Right' for Origin");
                         MouseTga();
                         if (Botton==LEFT)
                         {
                                 Xpos=X;
                                 Ypos=Y;
                         }
                         if (Botton==RIGHT)
                         {
                                 Xpos=0;
                                 Ypos=0;
                         }

                         filename=buffer;
                         SetPt(&P,Xpos,Ypos);
                         GetPic(filename,1,P);
                         //Sound(800,200);
```

```
color=ENABLE;
loop2=0;
}    // BUTTON Y 2-1
if (yms>=ymenu+(step*2)+hight && yms<=ymenu+(step*2)+(2*hight)&&vdo==ENABLE)
{
ClearMouse();
xms=0;
yms=0;
Choose(xmenu+skip,ymenu+(step*2)+hight,ymenu1[4]);
for(;;)
{
                WinText("Press 'A' to Save All 'S' to Select Area");
                c=getch();
                ClearWin();
                if(c=='s'||c=='S')
                 {
                WinText("Click 'Right' to Quit");
                MouseBoxTga();
                ClearWin();
                WinText("Press 'Y' to O.K. 'N' to Quit");
                c=getch();
                ClearWin();
                if(c=='y'||c=='Y')
                {
                 SetRect( &saverec, Xbox1, Ybox1 , Xbox2, Ybox2 );
                 break;
                }
                if(c=='n'||c=='N')
                {
                 BoxTga(Xbox1,Ybox1,Xbox2,Ybox2);
                 loop2=0;
                 break;
                }
                 }
                 if(c=='a'||c=='A')
                  {
                SetRect( &saverec, Xmin, Ymin , Xmin+Width, Ymin+Height );
                break;
                 }
                 //Sound(800,200);
 }    // forever loop
if(loop2!=0)
                 {
                  WinText("Enter File Name to Save");
                  InData(xmenu+skip,ymenu+(step*2)+hight,"Insert Name");
                  BoxTga(Xbox1,Ybox1,Xbox2,Ybox2);
                  filename=buffer;
                  PutPic(filename,saverec);
                 }
 loop2=0;
 }          //  BUTTON Y 2-2

 if (yms>=ymenu+(3*step)+(2*hight) && yms<=ymenu+(3*step)+(3*hight)&&color==ENABLE)
 {
 ClearMouse();
 xms=0;
 yms=0;
 Choose(xmenu+skip,ymenu+(3*step)+(2*hight),ymenu1[5]);
 WinText("Press 'R' or 'G' or 'B' for Image Color");
 c=getch();
 if (c=='r' || c=='R')
 {
                for(i=0;i<=Ymax;i++)
                 {
                  for(j=0;j<=Xmax;j++)
                  {
                                GetRGBPixel(j,i,&cc);
                                if (cc.red>=0x0000&& cc.red<=0x0fff)
                                { rgb.red=0xffff;rgb.green=0xffff;rgb.blue=0xffff;}
                                else
                                { rgb.red=cc.red;rgb.green=0x0000;rgb.blue=0x0000;}
                                SetRGBPixel(j,i,rgb);

                  }.
                 }

 }

 if (c=='g' || c=='G')
 {
```

```
                                        for(i=0;i<=Ymax;i++)
                                        {
                                         for(j=0;j<=Xmax;j++)
                                         {
                                                GetRGBPixel(j,i,&cc);
                                                if (cc.green>=0x0000&&cc.green<=0x0fff)
                                                { rgb.red=0xffff;rgb.green=0xffff;rgb.blue=0xffff;}
                                                else
                                                { rgb.red=0x0000;rgb.green=cc.green;rgb.blue=0x0000;}
                                                SetRGBPixel(j,i,rgb);
                                         }
                                        }

                                }
                                if (c=='b' || c=='B')
                                {
                                        for(i=0;i<=Ymax;i++)
                                        {
                                         for(j=0;j<=Xmax;j++)
                                         {
                                                GetRGBPixel(j,i,&cc);
                                                if (cc.blue>=0x0000&& cc.blue<=0x0fff)
                                                { rgb.red=0xffff;rgb.green=0xffff;rgb.blue=0xffff;}
                                                else
                                                { rgb.red=0x0000;rgb.green=0x0000;rgb.blue=cc.blue;}
                                                SetRGBPixel(j,i,rgb);
                                         }
                                        }

                                }

                                ClearWin();
                                getch();
                                color = DISABLE;
                                loop2=0;
                                }     // BUTTON Y 2-3


                                }  else loop2=0;      // END IF
                          }         // END BUTTON LEFT  Y 2
                          }    while(loop2!=0);  // END DO MENU Y 2
                          Clearpos(xmenu,ymenu,width*7,step*6);
                          ClearWin();
                          Full_Menu();
                           }        // END IF MENU Y-2
                    if(xms>=xmenu+(skip*2) && xms<=xmenu+(skip*2)+width)
                           { Full_Menu();
                            Choose(xmenu+(skip*2),ymenu,menu[2]);
                            //Sound(2000,500);
                            closegraph();
                            exit(1);
                            loop=0;
                           }        // END IF MENU Y-3

                    }          // END IF MENU X-AXIS
                   //else Sound(100,600);
            }               // END IF MOUSE LEFT BUTTON

    } while(loop!=0);       // END DO
}                   // END Main_Menu()

void ClearMouse(void)
{
  regs.x.ax = 2;
  int86(0x33,&regs,&regs);
}

void Move(int x,int y,int num,int init)
{
  int y1,i,j;
  y1 = y+7;
  j = 0;

    do
                {
                for(i=0;i<=step;i++)
                {
                Clearpos(x+7,y1,width,hight);
                y1++;
```

```
            Fillbox(x+7,y1,x+width+7,y1+hight,GREEN);
            delay(3);
            }
            Fillbox(x,y1-7,x+width,y1-7+hight,LIGHTGREEN);
            setcolor(RED);
            outtextxy(x,y1-7+12,ymenu1[init+j]);
            y1=y1+hight;
            j++;
            num--;
    } while(num != 0);
//  Choose(x,y," ^^^ ");
}


void Mouse(void)
{
 regs.x.ax=1;
 int86(0x33,&regs,&regs);


            regs.x.ax = 3;
            int86(0x33,&regs,&regs);
            xms = regs.x.cx;
            yms = regs.x.dx;
}
void ClearTga(void)
 {
            SetRect( &r, Xmin, Ymin , Xmin+Width, Ymin+Height );
            EraseRect(r);
}

void TextTga(char text[20],int posx,int posy,int num)
{
            LoadStrokeFont(font[num]);
            MoveTo(posx,posy);
            DrawString(text);
            DisposeStrokeFont();
}
void MouseTga(void)
{
    long oldpix[85];
    int cnt=0;
             unsigned int cntx,cnty;
    union REGS rg;
    X=10,Y=10;
             for(cntx=X-4;cntx<=X+4;cntx++)
    {
            for(cnty=Y+4;cnty>=Y-4;cnty--)
            {
             GetPixel(cntx,cnty,&oldpix[cnt]);
             cnt++;
            }
     }
     cnt=0;
             for(;;)
              {
               rg.x.ax = 11;
               int86(0x33,&rg,&rg);

               if(rg.x.cx || rg.x.dx)
               {
                cnt=0;
                        for(cntx=X-4;cntx<=X+4;cntx++)
                {
                        for(cnty=Y+4;cnty>=Y-4;cnty--)
                        {
                         SetPixel(cntx,cnty,oldpix[cnt]);
                         cnt++;
                        }
                }

                        X+=(signed)rg.x.cx/3;
                        Y-=(signed)rg.x.dx/3;

                                if(X>508) X=508;
                                if(X<5) X=5;
                                if(Y>=Ymax) Y=Ymax;
                                if(Y<5) Y=5;
                                cnt=0;
```

```
                        for(cntx=X-4;cntx<=X+4;cntx++)
                        {
                          for(cnty=Y+4;cnty>=Y-4;cnty--)
                                {
                                GetPixel(cntx,cnty,&oldpix[cnt]);
                                cnt++;
                                }
                        }
                                SetForeColor(0x00000000L);
                                MoveTo(X-1,Y+4); LineTo(X-1,Y-4);
                                MoveTo(X+1,Y+4); LineTo(X+1,Y-4);
                                MoveTo(X-4,Y+1); LineTo(X+4,Y+1);
                                MoveTo(X-4,Y-1); LineTo(X+4,Y-1);

                                SetForeColor(0x00ffffffL);
                                MoveTo(X,Y+4); LineTo(X,Y-4);
                                MoveTo(X-4,Y); LineTo(X+4,Y);

            }
                                rg.x.ax=3;
                                rg.x.bx=0;
                                int86(0x33,&rg,&rg);
                                if(rg.x.bx&2)
                                 {
                                   rg.x.ax=0; int86(0x33,&rg,&rg);
                                   cnt=0;
                                                for(cntx=X-4;cntx<=X+4;cntx++)
                                   {
                                    for(cnty=Y+4;cnty>=Y-4;cnty--)
                                    {
                                                SetPixel(cntx,cnty,oldpix[cnt]);
                                                cnt++;
                                    }
                                   }
                                   Botton=RIGHT;
                                   break;
                                 }
                        //      if(!yes) MoveTo(X,Y);
                                if(rg.x.bx&1)
                                {
                                   rg.x.ax=0; int86(0x33,&rg,&rg);
                                   cnt=0;
                                   for(cntx=X-4;cntx<=X+4;cntx++)
                                                {
                                    for(cnty=Y+4;cnty>=Y-4;cnty--)
                                    {
                                                SetPixel(cntx,cnty,oldpix[cnt]);
                                                cnt++;
                                    }
                                   }
                                   Botton=LEFT;
                                   break;
                                }

            }
};          // END MOUSETGA

void BoxTga(int x1,int y1,int x2,int y2)
{
  int cnt1=0;
  long c;

  if (x2>=x1)
  {
   for(cnt1=0;cnt1<=x2-x1;cnt1++)
              {
               GetPixel(x1+cnt1,y1,&c);
               SetPixel(x1+cnt1,y1,c^0x00000fff);
    GetPixel(x1+cnt1,y2,&c);
    SetPixel(x1+cnt1,y2,c^0x00000fff);
   }
  }
  if (x2<x1)
  {
   for(cnt1=0;cnt1<=x1-x2;cnt1++)
   {
    GetPixel(x1-cnt1,y1,&c);
    SetPixel(x1-cnt1,y1,c^0x00000fff);
```

```
    GetPixel(x1-cnt1,y2,&c);
    SetPixel(x1-cnt1,y2,c^0x00000fff);
   }
  }
  if (y1>=y2)
  {
   for(cnt1=0;cnt1<=y1-y2;cnt1++)
              {
    GetPixel(x1,y1-cnt1,&c);
    SetPixel(x1,y1-cnt1,c^0x00000fff);
    GetPixel(x2,y1-cnt1,&c);
    SetPixel(x2,y1-cnt1,c^0x00000fff);
   }
  }
  if (y1<y2)
  {
   for(cnt1=0;cnt1<=y2-y1;cnt1++)
   {
    GetPixel(x1,y1+cnt1,&c);
    SetPixel(x1,y1+cnt1,c^0x00000fff);
    GetPixel(x2,y1+cnt1,&c);
    SetPixel(x2,y1+cnt1,c^0x00000fff);
   }
  }

}
void InData(int x,int y,char text[10])
{  int i;
   char ch;
   int x1=x+7+width;
   int y1=y+7+hight;
   for(i=0;i<=300;i++)
   {
    Fillbox(x1,y+7,x1+i,y1,GREEN);
   }
   for(i=0;i<=35;i++)
   {
    Fillbox(x+7,y1,x1+300,y1+i,GREEN);
   }
   Fillbox(x,y,x+width+300,y+hight+35,LIGHTGREEN);
   setcolor(BLACK);
   rectangle(x+5,y+5,x+width+295,y+hight+30);
   outtextxy(x+10,y+15,text);
   Fillbox(x+10,y+35,x+360,y+50,BLUE);

  for(i=0;i<50;i++)
  {
            buffer[i]=0;
  }
  i=0;
  do
  {
   ch = getch();
   if(ch!=27&&ch!=13&&ch!=8)
   {
    Fillbox(x+10,y+35,x+360,y+50,BLACK);
    buffer[i]=ch;
    i++;
    setcolor(WHITE);
    moveto(x+12,y+40);
    outtext(strcat(buffer,"_"));
   }
   if(ch==8&&i>0)
   {
             Fillbox(x+10,y+35,x+360,y+50,BLACK);
    i--;
    buffer[i]=0;
             setcolor(WHITE);
    moveto(x+12,y+40);
    outtext(strcat(buffer,"_"));
   }
   if(ch==27)
   {
            for(i=0;i<49;i++)
            {
            buffer[i]=0;
            }
   }
```

```
  }while(ch!=13&&ch!=27);
  buffer[i]=0;
}

void Pop(int x,int y,char text[20])
{
 int ph=200;
 int pw=100;
 int i;

}
void WinText(char text[50])
{
  len=((strlen(text))*8)+20;
  ywin=380;
  xwin=(getmaxx()/2)-(len/2);
  Fillbox(xwin,ywin,xwin+len,ywin+30,MAGENTA);
  Fillbox(xwin+5,ywin+5,xwin+len-5,ywin+25,LIGHTMAGENTA);
  setcolor(BLACK);

  outtextxy(xwin+12,ywin+13,text);
}
void MouseBoxTga(void)
{
        long old;
        int j=0,m=0;
        unsigned int cntx,cnty,cnt=5;
        union REGS rg;
        X=10,Y=10;
        Xbox1=0;
        Xbox2=0;
        Ybox1=0;
        Ybox2=0;

        for(cnty=0;cnty<=5;cnty++)
        {
                for(cntx=0;cntx<=cnt;cntx++)
                {
                 GetPixel(X+cntx,Y-cnty,&old);
                 SetPixel(X+cntx,Y-cnty,old^0x00ffffff);
                }
                cnt--;
        }
                cnt=5;
       for(;;)
        {
                rg.x.ax = 11;
                int86(0x33,&rg,&rg);

                if(rg.x.cx || rg.x.dx)
                {
        for(cnty=0;cnty<=5;cnty++)
        {
                for(cntx=0;cntx<=cnt;cntx++)
                {
                 GetPixel(X+cntx,Y-cnty,&old);
                 SetPixel(X+cntx,Y-cnty,old^0x00ffffff);
                }
                cnt--;
        }
                cnt=5;

                X+=(signed)rg.x.cx/3;
                Y-=(signed)rg.x.dx/3;

                        if(X>508) X=508;
                        if(X<5) X=5;
                        if(Y>=Ymax) Y=Ymax;
                        if(Y<5) Y=5;

        for(cnty=0;cnty<=5;cnty++)
        {
                for(cntx=0;cntx<=cnt;cntx++)
                {
                 GetPixel(X+cntx,Y-cnty,&old);
                 SetPixel(X+cntx,Y-cnty,old^0x00ffffff);
                }
                cnt--;
        }
```

```
                cnt=5;


                }
                        rg.x.ax=3;
                        rg.x.bx=0;
                        int86(0x33,&rg,&rg);
                        if(rg.x.bx&1&&j==0)
                        {
                         j=1;
                         BoxTga(Xbox1,Ybox1,Xbox2,Ybox2);
                         Xbox1=X;
                         Ybox1=Y;
                         Xbox2=X-1;
                         Ybox2=Y+1;


                                for(;;)
                                 {
                        rg.x.ax=3;
                        rg.x.bx=0;
                        int86(0x33,&rg,&rg);

                                 if(rg.x.bx&1)
                                        {
                                         BoxTga(Xbox1,Ybox1,Xbox2,Ybox2);
                                         Xbox2=X;
                                         Ybox2=Y;
                                         BoxTga(Xbox1,Ybox1,Xbox2,Ybox2);

                rg.x.ax = 11;
                int86(0x33,&rg,&rg);

                        if(rg.x.cx || rg.x.dx)
                        {
                                        for(cnty=0;cnty<=5;cnty++)
                                        {
                                                for(cntx=0;cntx<=cnt;cntx++)
                                                {
                                                        GetPixel(X+cntx,Y-cnty,&old);
                                                        SetPixel(X+cntx,Y-cnty,old^0x00ffffff);
                                                 }
                                                cnt--;
                                        }
                                                cnt=5;

                                                X+=(signed)rg.x.cx/3;
                                                Y-=(signed)rg.x.dx/3;

                                                if(X>508) X=508;
                                                if(X<5) X=5;
                                                if(Y>=Ymax) Y=Ymax;
                                                if(Y<5) Y=5;

                                                        for(cnty=0;cnty<=5;cnty++)
                                                        {
                                                         for(cntx=0;cntx<=cnt;cntx++)
                                                         {
                                                                GetPixel(X+cntx,Y-cnty,&old);
                                                                SetPixel(X+cntx,Y-cnty,old^0x00ffffff);
                                                          }
                                                                cnt--;
                                                        }
                                                                cnt=5;
}
                                        }
                                        else
                                        { j=0;
                                          break;
                                         }
                                }
                        }
                        if(rg.x.bx&2)
                         {
                                rg.x.ax=0; int86(0x33,&rg,&rg);
        for(cnty=0;cnty<=5;cnty++)
        {
                for(cntx=0;cntx<=cnt;cntx++)
```

```
                {
                 GetPixel(X+cntx, Y-cnty,&old);
                 SetPixel(X+cntx, Y-cnty,old^0x00ffffff);
                }
                cnt--;
        }
                cnt=5;

                                        j=0;
                                       break;
                                }

        }

}       // END MOUSEBOXTGA
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <mem.h>
#include <conio.h>
//#include "ipx.h"
#include <fcntl.h>
#include <io.h>
#define  IPXINSTALL 0xff
#define  GET_CON 0xdc
#define  send_nm[]
typedef struct
        { unsigned int  checksum;
          unsigned int  length;
          unsigned char transport_control;
          unsigned char packet_type;
          unsigned char dest_network_number[4];
          unsigned char dest_node_address[6];
                  unsigned int  dest_socket;
                  unsigned char source_network_number[4];
                  unsigned char source_node_address[6];
                  unsigned int  source_socket;
       }IPXHEADER;

typedef struct
        { void far     *link_address;
          void far     (*esr)(void);
          unsigned char in_use;
                  unsigned char completion_code;
                  unsigned int  socket_number;
                  unsigned char workspace[4];
          unsigned char driver_workspace[12];
          unsigned char immediate_address[6];
          unsigned int  packet_count;
          struct
                  {  void far *address;
             unsigned int length;
           } packet[2];
        }ECB;

typedef struct
                {  unsigned char network_num[4];
           unsigned char node_address[6];
        } NET_ADDRESS;

typedef struct
                {  unsigned int  len;
                   unsigned char count;
                   unsigned char connection_num[100];
                } CON_ADDRESS;

void far (*ipx_spx)(void);

void ipx_install(void)
{
    asm mov ax,0x7a00;
    asm int 0x2f;
    if(_AL != 0xff)
      {  printf("\n IPX not installed.");
                exit(0);
      }
    //ipx_spx = MP_FP(_ES,_IS);
    printf("\n IPX is installed now.");
}

void create_socket(unsigned int socket)
{
    asm mov al,0x00;
    asm mov bx,0x00;
    asm mov dx,socket;
    asm int 0x7a;
    if (_AL != 0x00)
                {  printf("\n Can't open socket ");
                   exit(0);
                }
    printf("\n socket complte ");
}

void get_connection_number(char *name,CON_ADDRESS  *con_buff_ptr)
{  struct
     {  unsigned int  len;
                unsigned char buff_type;
                unsigned int  object_type;
                unsigned char object_name_len;
```

```
            unsigned char name[47];
    } request_buff;

    request_buff.len = 51;
    request_buff.buff_type      = 0x15;
    request_buff.object_type    = 0x0100;
    request_buff.object_name_len = (unsigned char) strlen(name);
    strcpy(request_buff.name,name);
    con_buff_ptr->len = 101;

    _SI = FP_OFF ((void far*)&request_buff);
    _DS = FP_SEG ((void far*)&request_buff);
    _DI = FP_OFF ((void far*)con_buff_ptr);
    _ES = FP_SEG ((void far*)con_buff_ptr);

    asm mov ah,0xe3;
    asm int 0x21;

    if (_AL != 0)
            { printf("\n Unable get connetion number.");
              exit(0);
            }
    if (con_buff_ptr->count == 0)
            { printf("\n User ID not  available");
              exit(0);
            }
    if (con_buff_ptr->count > 1)
            { printf("\n There are user id logging in more than one");
              exit(0);
            }
}

unsigned char get_local_con_number(void)
{
    asm mov ah,0xdc;
    asm mov al,0x00;
    asm int 0x21;
    return _AL;
}

void get_internet_add(unsigned char number,NET_ADDRESS *net_buff_ptr)
{  struct
    { unsigned int  len;
            unsigned char type;
            unsigned char con_number;
    } request_buff;
   struct
    { unsigned int  len;
            unsigned char network_num[4];
            unsigned char node_address[6];
            unsigned int  socket;
    } reply_buff;

    request_buff.len       = 2;
    request_buff.type      = 0x13;
    request_buff.con_number = number;
    reply_buff.len         = 12;

    _SI = FP_OFF ((void far*)&request_buff);
    _DS = FP_SEG ((void far*)&request_buff);
    _DI = FP_OFF ((void far*)&reply_buff);
    _ES = FP_SEG ((void far*)&reply_buff);

    asm mov ah,0xe3;
    asm int 0x21;

    if (_AL != 0)
            { printf("\n Unable get internet address.");
              exit(0);
            }
    memcpy(net_buff_ptr->network_num,reply_buff.network_num,4);
    memcpy(net_buff_ptr->node_address,reply_buff.node_address,6);
}

void get_local_target(NET_ADDRESS net_buff,unsigned char *imme_add_ptr)
{  struct
    { unsigned char network_num[4];
            unsigned char node_address[6];
            unsigned int  d_socket;
    } request_buff;

    memcpy(request_buff.network_num,net_buff.network_num,4);
    memcpy(request_buff.node_address,net_buff.node_address,6);
    request_buff.d_socket  = 0x5555;
```

```
    _SI = FP_OFF ((void far*)&request_buff);
    _DS = FP_SEG ((void far*)&request_buff);
    _DI = FP_OFF ((void far*)imme_add_ptr);
    _ES = FP_SEG ((void far*)imme_add_ptr);

    asm mov bx,0x02;
    asm int 0x7a;

    if (_AL != 0)
              { printf("\n Unable get local target.");
                exit(0);
              }
}

void get_local_con_number1(unsigned char *user_num)
{
   asm mov ah,0xdc;
   asm mov al,0x00;
   asm int 0x21;
   *user_num = _AL;
}

void  get_user_info(unsigned char user_num,unsigned char *name)
{  struct
    { unsigned int   len;
              unsigned char  type;
              unsigned char  num;
    } request_buff;
  struct
    { unsigned int   len;
              unsigned char  object_id[4];
              unsigned int   object_end;
              unsigned char  object_name[48];
              unsigned char  log_time[7];
               } reply_buff;

              request_buff.len  = 2;
              request_buff.type  = 0x16;
              request_buff.num  = user_num;

              reply_buff.len  = 61;

    _SI = FP_OFF ((void far*)&request_buff);
    _DS = FP_SEG ((void far*)&request_buff);
    _DI = FP_OFF ((void far*)&reply_buff);
    _ES = FP_SEG ((void far*)&reply_buff);

    asm mov ah,0xe3;
    asm int 0x21;

    if (_AL != 0)
              { printf("\n Unable get user_info.");
                exit(0);
              }
               memset(name,0,48);
    memcpy(name,reply_buff.object_name,48);
}


void ipx_breath(void)
{   asm mov bx,0x000a;
    asm int 0x7a;
}


void ipx_send( ECB         *ecb_send_ptr,
               IPXHEADER    *head_send_ptr,
               NET_ADDRESS  net_buff,
               unsigned char *imme_add_ptr,
               unsigned char *packet_ptr)
{
 ecb_send_ptr->in_use=1;


              memset(ecb_send_ptr,0,sizeof(ECB));
              ecb_send_ptr->socket_number    = 0x4444;
              memcpy(ecb_send_ptr->immediate_address,imme_add_ptr,6);
              ecb_send_ptr->packet_count     = 2;
              ecb_send_ptr->packet[0].address = head_send_ptr;
              ecb_send_ptr->packet[0].length  = sizeof(IPXHEADER);
              ecb_send_ptr->packet[1].address = packet_ptr;
              ecb_send_ptr->packet[1].length  = 500;

              head_send_ptr->packet_type = 4;
              memcpy(head_send_ptr->dest_network_number,net_buff.network_num,4);
```

```
            memcpy(head_send_ptr->dest_node_address,net_buff.node_address,6);
            head_send_ptr->dest_socket  = 0x5555;

            _ES = FP_SEG ((void far*) ecb_send_ptr);
            _SI = FP_OFF ((void far*) ecb_send_ptr);
            asm mov bx,0x03;
            asm int 0x7a;
            ipx_breath();


//          printf("\n completion code: %x  ",ecb_send_ptr->completion_code);
//  ipx_breath();
}

void send( ECB          *ecb_send_ptr,
           IPXHEADER    *head_send_ptr,
           NET_ADDRESS   net_buff,
           unsigned char *imme_add_ptr,
//         unsigned char *data_ptr,
           char *name)
{char bname[12];
 unsigned int  i,x;          /*   send   */
 unsigned char  packet[500],len[3],lenp[5],sender_num,sender_name[48];
 FILE *input,*output;
unsigned long int size,handle;
unsigned int nx,n,np;
int num,check;
memset(bname,0,12);
strcpy(bname,name);
handle = open(name,O_CREAT);
size=filelength(handle);
n=size/500;
nx=size%500;
get_local_con_number1(&sender_num);
get_user_info(sender_num,sender_name);
printf("\nSender name is %s\n",sender_name);
np=n+2;
printf("np=%d\n",np);
printf("nx=%d\n",nx);
   if ((input = fopen(name, "rb"))
     == NULL)
  {
    printf("Cannot open input file %s.\n",name);
    exit(1);
  }

  if (n<1) {     /*  1   */
  memset(packet,0,500);
  fread(packet,500,1,input);
          ipx_send(ecb_send_ptr,head_send_ptr,net_buff,imme_add_ptr,packet);
          ipx_breath();
          delay(30);
    }           /*    1     */

   if (n>=1) {          /*    A     */
  while (n) {                 /*    B   */
              memset(packet,0,500);
              fread(packet,500,1,input);
          ipx_send(ecb_send_ptr,head_send_ptr,net_buff,imme_add_ptr,packet);
          ipx_breath();
          delay(30);
              n--;
              }       /*     B     */
 //         printf("the last");
          delay(30);
          memset(packet,0,500);
          fread(packet,500,1,input);
          ipx_send(ecb_send_ptr,head_send_ptr,net_buff,imme_add_ptr,packet);
          ipx_breath();
          delay(30); } /*      A   */
   memset(packet,0,500);
   memset(packet,0x1b,8);
   memset(len,0,3);
   memset(lenp,0,5);
   itoa(nx,len,10);
   itoa(np,lenp,10);
   check=8;
   for(num=0;num<=2;num++) {
   packet[check]=len[num];
   check++;
   }
   for(num=0;num<=4;num++) {
   packet[check]=lenp[num];
   check++;
   }
```

```
  check=18;
  for(num=0;num<12;num++){
  packet[check]=bname[num];
  check++;
  }
  check=40;
  for(num=0;num<48;num++){
  packet[check]=sender_name[num];
  check++;
  }
  ipx_send(ecb_send_ptr,head_send_ptr,net_buff,imme_add_ptr,packet);
  ipx_breath();
  fclose(input);
}                 /*  send  */
int check()
{ ECB           *ecb_read;
  IPXHEADER      *head_read;
  unsigned int    listen_socket;
  unsigned char packet[500],yyy[8],nnn[8];
  int i,yes,no;
    clrscr();
    ipx_install();
    listen_socket=0x5555;
    create_socket(listen_socket);
    memset(yyy,0xaa,8);
    memset(nnn,0xbb,8);
    memset(packet,0,500);
    memset(ecb_read,0,sizeof(ECB));
    memset(head_read,0,sizeof(IPXHEADER));

    ecb_read->socket_number    = 0x5555;
    ecb_read->packet_count     = 2;
    ecb_read->packet[0].address = head_read;
    ecb_read->packet[0].length  = sizeof(IPXHEADER);
    ecb_read->packet[1].address = packet;
    ecb_read->packet[1].length  = 500;

     _ES = FP_SEG ((void far*)ecb_read);
     _SI = FP_OFF ((void far*)ecb_read);

     asm mov bx,0x0004;
     asm int 0x7a;

              while(ecb_read->in_use)
              {  ipx_breath();
              }
     if(ecb_read->completion_code != 0)
              {  printf("\n Listen error!! ");
                 printf("\n completion code : %x",ecb_read->completion_code);
                 exit(0);
              }
yes=strncmp(yyy,packet,8);
no=strncmp(nnn,packet,8);
if(yes==0){
printf("######  Data send correct  ######\n");
}
else
{
printf(" ! Data retransmiting !\n");
}
//printf("yes=%d\n",yes);
//printf("no=%d\n",no);
return(yes);
}
void Send_main(void)
{ unsigned char     imme_add[6],
                              user_name[47];
  unsigned int      send_socket;
  int repeat;
  CON_ADDRESS       con_buff;
  NET_ADDRESS       net_buff;
  ECB               ecb_send,ecb_read;
  IPXHEADER         head_send,head_read;
  unsigned char name[12],file_name[12];
  printf("\nEnter destination  ");
  gets(name);
  printf("\nEnter file name to send  ");
  gets(file_name);
    clrscr();
    ipx_install();

    send_socket=0x4444;
    create_socket(send_socket);
    get_connection_number(name,&con_buff);
```

```
    printf("\n con_number of %s is %d ",name,con_buff.connection_num[0]);

    get_internet_add(con_buff.connection_num[0],&net_buff);

    get_local_target(net_buff,imme_add);
    {
    repeat=1;
    while(repeat){
    send(&ecb_send,&head_send,net_buff,imme_add,file_name);
    repeat=check();}
    }

}
 void ipx_listen( ECB          *ecb_read_ptr,
                  IPXHEADER    *head_read_ptr,
                  unsigned char *packet_ptr
                )
{
     memset(ecb_read_ptr,0,sizeof(ECB));
     memset(head_read_ptr,0,sizeof(IPXHEADER));

     ecb_read_ptr->socket_number    = 0x5555;
     ecb_read_ptr->packet_count     = 2;
     ecb_read_ptr->packet[0].address = head_read_ptr;
     ecb_read_ptr->packet[0].length  = sizeof(IPXHEADER);
     ecb_read_ptr->packet[1].address = packet_ptr;
     ecb_read_ptr->packet[1].length  = 500;

     _ES = FP_SEG ((void far*) ecb_read_ptr);
     _SI = FP_OFF ((void far*) ecb_read_ptr);

     asm mov bx,0x0004;
     asm int 0x7a;

              while(ecb_read_ptr->in_use)
               {  ipx_breath();
               }
     if(ecb_read_ptr->completion_code != 0)
               {  printf("\n Listen error!! ");
                  printf("\n completion code : %x",ecb_read_ptr->completion_code);
                  exit(0);
               }
}
void send1( ECB         *ecb_send_ptr,
            IPXHEADER   *head_send_ptr,
            NET_ADDRESS  net_buff,
            unsigned char *imme_add_ptr,
            int i)

{ unsigned char  packet[500];
unsigned char *yes="Y";
unsigned char *no="N";
FILE *input,*output;
//int num,check;
   memset(packet,0,500);
   if(i==1)
   {
   memset(packet,0xaa,8);
   }
   if(i==0)
   {
   memset(packet,0xbb,8);
   }
   ipx_send(ecb_send_ptr,head_send_ptr,net_buff,imme_add_ptr,packet);
   ipx_breath();
   fclose(input);
}                  /*  send  */

void check_r(int i,char *name1)
{ unsigned char     imme_add[6],
                    user_name[47];
  unsigned int      send_socket;
  CON_ADDRESS       con_buff;
  NET_ADDRESS       net_buff;
  ECB               ecb_send,ecb_read;
  IPXHEADER         head_send,head_read;


//    printf("\Loop send1\n");
//    printf("\ni=%d\n",i);
    ipx_install();
//    printf("\nname1=%s\n",*name1);

    send_socket=0x4444;
    create_socket(send_socket);
```

```
    get_connection_number(name1,&con_buff);
    printf("\n con_number of %s is %d ",name1,con_buff.connection_num[0]);

    get_internet_add(con_buff.connection_num[0],&net_buff);

    get_local_target(net_buff,imme_add);
    send1(&ecb_send,&head_send,net_buff,imme_add,i);
}



void listen( ECB        *ecb_read_ptr,
             IPXHEADER    *head_read_ptr)

{ CON_ADDRESS       con_buff;
  NET_ADDRESS       net_buff;
 unsigned char      imme_add[6];
 unsigned char   packet[500],end[8],temp2[500],temp1[500],lenr[3],lenpr[5];
 unsigned int       send_socket;
  int           i,check,num;
  unsigned int  nr,jj,np,n,temp;
  char *rname[12],sender_name[48];
  FILE *output;
    if ((output = fopen("temp.zzz", "wb"))
     == NULL)
  {
    printf("Cannot open output file %s.\n");
    exit(1);
  }

    memset(sender_name,0,48);
    i=1;n=0;
    memset(end,0x1b,8);
    while(i)                /*   0   */
              {  memset(packet,0,500);
                 ipx_listen(ecb_read_ptr,head_read_ptr,packet);
                 memset(temp1,0,500);
                 memcpy(temp1,packet,500);
                 n=n+1;
                 i = strncmp(end,packet,8);
                 if(i) {     /*     1    */
                            if (n>1){          /*    2    */
                            fwrite(temp2,500,1,output);}   /*   2   */
                            memset(temp2,0,500);
                            memcpy(temp2,temp1,500);
                            }                  /*    1    */
                 else
                           {                   /*    3    */
                           check=8;
                            for(num=0;num<=2;num++)
                             { lenr[num]=packet[check];
                                       check++;}
                            for(num=0;num<=4;num++)
                             { lenpr[num]=packet[check];
                                       check++;}
                          check=18;
                            for(num=0;num<12;num++)
                              {rname[num]=&packet[check];
                              check++;
                              }
                          check=40;
                            for(num=0;num<8;num++)
                            {sender_name[num]=packet[check];
                            check++;
                            }

                           nr=atoi(lenr);
                          np=atoi(lenpr);
                          fwrite(temp2,nr,1,output);
                          printf("\nSender name is %s\n",sender_name);
                          printf("??? np=%d\n",np);
                          printf("??? n=%d\n",n);
                          printf("temp=%d\n",temp);
                           if(np!=n) {        /*     4     */
                           printf("\np=%d   n=%d",np,n);
                           printf("\nData received NOT COMPLETE !\n");
                           fclose(output);
                           check_r(0,sender_name);

                           }
                           else   {         /*  5  */
                           printf("\n## Data received COMPLETE ##\n");
                           printf("File name is %s\n",*rname);
```

```
                        printf("\np=%d  n=%d",np,n);
                        fclose(output);
                        check_r(1,sender_name);
                        }
                        } /*   3  */
                        } /*   0  */
                rename("temp.zzz",*rname);
                }

void listen_main(void)
{ ECB           ecb_read;
  IPXHEADER     head_read;
  unsigned int  listen_socket;

    clrscr();
    ipx_install();

    listen_socket=0x5555;
    create_socket(listen_socket);
    listen(&ecb_read,&head_read);
}
void send_message( unsigned char  number,
                   unsigned char  *mes
                 )
{  struct
        { unsigned int   len;
          unsigned char  type;
          unsigned char  con_count;
          unsigned char  con_number;
          unsigned char  m_len;
          unsigned char  message[55];    //LIMIT AT 45 BYTE
        }request_buff;
   struct
        { unsigned int   len;
          unsigned char  con_count;
          unsigned char  result;
        }reply_buff;



        request_buff.type      = 0x00;
        request_buff.con_count  = 1;
        request_buff.con_number = number;
        request_buff.m_len     = 55;
        memcpy(request_buff.message,mes,55);
        request_buff.len       = (request_buff.m_len+4);

        reply_buff.len       =0x04;

        _SI = FP_OFF ((void far*)&request_buff);
        _DS = FP_SEG ((void far*)&request_buff);
        _DI = FP_OFF ((void far*)&reply_buff);
        _ES = FP_SEG ((void far*)&reply_buff);
        asm mov ah,0xe1;
        asm int 0x21;

}

void setup ( unsigned char number,
             unsigned char *name
           )
{ unsigned char  packet[500],
                 response[8];

  unsigned char  *call;

  unsigned int   a;

  call = (unsigned char *) calloc(56,1);
  if (call == NULL)
    {   printf("\nCAN'T ALLOCATE MEMORY (call)");
        exit(0);
    }
  sprintf(call,"Please  choose  RECEIVE botton ");
  strupr(name);
  strcat(call,name);

    a = strlen(call);
    memset(&(call[a]),0x20,(55-a));
    call[55]=0;
    send_message(number,call);

}
```