ระบบตรวจสอบเครือข่ายแบบอีเทอร์เนต

Ethernet Network Monitoring System

โดย

นาย สันทัด เกียรติสุไพบูลย์

นาย ชัยรัตน์ โล้วโสภณกุล

วัน เดือน ปี 31 ก.ค. 2540

เลขทะเบียน 037092

เลขเรียกหนังสือ T 38185 ส583 ร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

037092

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจสอบเครือข่ายแบบอีเทอร์เนต
Ethernet Network Monitoring System

ผู้จัดทำ

1. นายสันทัด เกียรติสุไพบูลย์ 35104465
2. นายชัยรัตน์ โล้วโสภณกุล 35104524

อาจารย์ที่ปรึกษา

(อ. บรรจง ปียธำรง)

อาจารย์ที่ปรึกษา

(อ. สมศักดิ์ วลัยรัชต์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบตรวจสอบเครือข่ายแบบอีเทอร์เนต

นาย สันทัด เกียรติสุไพบูลย์ 35104465
นาย ชัยรัตน์ โล้วโสภณกุล 35104524

อาจารย์ที่ปรึกษา
อ. บรรจง ปิยะธำรง
อ. สมศักดิ์ วลัยรัชต์

ปีการศึกษา 2538

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ จัดทำขึ้นเพื่อวัตถุประสงค์ ในการศึกษาและวิเคราะห์การทำงานของระบบเครือข่ายคอมพิวเตอร์ เพื่อช่วยเป็นแนวทางการออกแบบ จัดการดูแลและบำรุงรักษาระบบเครือข่ายให้สามารถทำงานได้อย่างมีประสิทธิภาพ

การศึกษาจะเน้นหนักไปทางระบบเครือข่ายที่มีใช้อยู่จริงตาม มาตรฐานอีเทอร์เนต ซึ่งเป็นมาตรฐานหนึ่งในระบบเครือข่ายท้องถิ่น (Local Area Network) ได้รับการพัฒนาขึ้นภายใต้ข้อกำหนดพื้นฐาน OSI ทำให้มีความยืดหยุ่นในการเชื่อมต่อสูง ไม่ผูกติดกับผู้ผลิตรายใดรายหนึ่ง ทำให้ผู้ออกแบบหรือผู้บริหารระบบเครือข่าย มีทางเลือกในการวาง และจัดการระบบมากมาย และทางเลือกนี้เอง จึงเป็นเหตุให้ต้องหาเครื่องมือแสดงประสิทธิภาพการใช้งาน ไม่ว่าจะเป็นเรื่องโพรโตคอล หรือสภาพความคับคั่งของข้อมูลบนอุปกรณ์การสื่อสาร เพื่อช่วยในการตัดสินใจ แต่ในปัจจุบัน เครื่องมือที่ใช้วัดประสิทธิภาพนี้ ยังมีราคาสูงมาก ซึ่งอาจจะเป็นอุปสรรคในการจัดหา

ดังนั้นทางคณะผู้จัดทำจึงได้ จัดทำโปรแกรมเพื่อช่วยในงานดังกล่าว โดยในการใช้งานต้องการอุปกรณ์เพียงเล็กน้อย คือเครื่องไมโครคอมพิวเตอร์ และเนตเวอร์กการ์ด ซึ่งเป็นอุปกรณ์พื้นฐานที่มีอยู่แล้วบนระบบเนตเวอร์ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Ethernet Network Monitoring System

SANTAD KIATSUPAIBUL 35104465

CHAIRUTTHANA LOWSOPHONKUL 35104524

ADVISOR

BUNJONG PIYATHAMRONG

SOMSAK WALAIRUTCHATA

SEMESTER 1995

ABSTRACT

The objective of this thesis is to analize the computer network, guiding to get the best performance, by showing how to design, maintain and manage it.

We will focus only in Ethernet LAN, the flexible and economic network system. the Ethernet is developed under the OSI Specification, so make it compatible with many vendor products. Not depending on any product, we have many choices to implement the network. The buyer should have some tools helping him to decide to get the best product. That can monitor the traffic on Network segment , show the protocol usage. But now the available is too expensive to afford.

For these reason,we presented some programmed tools, working so well by using a little bit hardware (MicroComputer and Network Card). We hope that it can help you to get the best one.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้าที่
บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎีและหลักการ	
2.1 ระบบเครือข่ายคอมพิวเตอร์ตามมาตรฐาน OSI	3
2.2 รายละเอียดที่ควรสนใจในระบบเครือข่ายท้องถิ่น	9
2.3 มาตาฐาน IEEE กับระบบเครือข่ายท้องถิ่น	14
2.4 รูปแบบของ ODI กับมาตรฐาน OSI	17
2.5 ระบบเครือข่ายแบบอีเทอร์เนต	19
2.6 ชุดอินเตอร์เนตโพรโตคอล	23
2.7 ชุดเนตแวร์โพรโตคอล	38
2.8 เปรียบเทียบการทำงานของ TCP/IP กับ SPX/IPX	44
บทที่ 3 การคำนวณและการสร้าง	
3.1 การทำงานและประโยชน์ของแพคเก็ตไดรเวอร์	45
3.2 การเรียกใช้โปรแกรมแพคเก็ตไดรเวอร์	47
3.3 การเขียนโปรแกรมติดต่อกับแพคเก็ตไดรเวอร์	49
3.4 การเขียนโปรแกรมติดต่อกับแพคเก็ตไดรเวอร์ เพื่อรับเฟรมข้อมูล	60
3.5 ส่วนของข้อมูลที่แพคเก็ตไดรเวอร์ รับได้	67
บทที่ 4 การทดลองและผลการทดลอง	
4.1 ตัวอย่างเฟรมข้อมูล ที่ได้รับจากแพคเก็ตไดรเวอร์	69
4.2 การนำเฟรมข้อมูล ที่ได้จากแพคเก็ตไดรเวอร์ มาใช้ประโยชน์	72
4.3 โปรแกรม Ethernet Monitor	74
4.3.1 ความสามารถของโปรแกรม Ethernet Monitor	75
4.3.2 ข้อกำหนดของโปรแกรม Ethernet Monitor	87
บทที่ 5 บทวิจารณ์และสรุป	88

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

System Flowchart

Programming Code

กิตติกรรมประกาศ

หนังสืออ้างอิง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

รูปที่		หน้าที่
	บทที่ 2 ทฤษฎีและหลักการ	
2.1	แสดงการใช้เฮดเดอร์ในระดับชั้นต่างๆ	4
2.2	แสดงเนตเวอร์กโทโพโลยีแบบต่างๆ	9
2.3	แสดงการแบ่งเวลาใช้สายสัญญาณแบบ CSMA / CD	10
2.4	แสดงการแบ่งเวลาใช้สายสัญญาณแบบ Token-Passing	11
2.5	แสดงลักษณะของสัญญาณ	12
2.6	แสดงชนิดของสายสัญญาณ	12
2.7	แสดงความสัมพันธ์ระหว่างมาตรฐาน OSI กับมาตรฐาน IEEE	14
2.8	แสดงตัวอย่างมาตรฐาน IEEE แบ่งออกเป็นระดับชั้นต่างๆ	15
2.9	แสดงการเปรียบเทียบมาตรฐาน OSI กับรูปแบบของ ODI	17
2.10	แสดงลักษณะของเฟรมอีเทอร์เนตและเฟรม IEEE 802.3	19
2.11	แสดงการเปรียบเทียบมาตรฐาน OSI กับชุดอินเตอร์เนตโพรโตคอล	24
2.12	แสดงฟิลด์ต่างๆของ IP แพคเก็ต	25
2.13	แสดงค่าต่างๆในฟิลด์ Type Of Service	26
2.14	แสดงการจัดแบ่งชั้นของค่าตำแหน่งที่อยู่ IP	28
2.15	แสดงฟิลด์ต่างๆของ TCP แพคเก็ต	30
2.16	แสดงฟิลด์ต่างๆของ UDP แพคเก็ต	34
2.17	แสดงการทำงานของโพรโตคอลที่อยู่ในระดับชั้นบน	36
2.18	แสดงการเปรียบเทียบมาตรฐาน OSI กับชุดเนตแวร์โพรโตคอล	38
2.19	แสดงฟิลด์ต่างๆของ IPX แพคเก็ต	39
2.20	แสดงฟิลด์ต่างๆของ SPX แพคเก็ต	42
	บทที่ 3 การคำนวณและการสร้าง	
3.1	แสดงการทำงานของแพคเก็ตไดรเวอร์	45
3.2	แสดงการเขียนโปรแกรมติดต่อ เพื่อรับเฟรมข้อมูลจากแพคเก็ตไดรเวอร์	59
3.3	แสดงส่วนของข้อมูลที่แพคเก็ตไดรเวอร์รับได้	67

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 การทดลองและผลการทดลอง

4.1 แสดงข้อมูลที่รับได้จากการใช้คำสั่ง ls ซึ่งใช้ชุดอินเตอร์เนตโพรโตคอล 70

4.2 แสดงข้อมูลที่รับได้จากการใช้คำสั่ง dir ซึ่งใช้ชุดเนตแวร์โพรโตคอล 71



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่	หน้าที่
บทที่ 2 ทฤษฎีและหลักการ	
2.1 สรุปรายละเอียดมาตรฐาน OSI แยกตามระดับชั้น	8
2.2 แสดงมาตรฐาน IEEE แบบต่างๆ	16
2.3 แสดงรายละเอียดมาตรฐานแบบต่างๆของ IEEE 802.3	22
2.4 แสดงค่าต่างๆในฟิลด์ Type Of Service	26
2.5 แสดงค่าและความหมายในฟิลด์ Flags	27
2.6 แสดงค่าหมายเลขพอร์ตต่างๆในส่วนฟิลด์ Destination และ Source Port	31

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1 บทนำ

ในยุคโลกาภิวัฒน์ ระบบเครือข่ายคอมพิวเตอร์ ได้เข้ามามีบทบาทในด้านต่างๆ อย่างมากมาย ไม่ว่าจะด้านการดำรงชีวิต ด้านการศึกษา หรือทางด้านธุรกิจ ดังจะเห็นได้จากโครงการต่างๆเช่น ระบบการศึกษาทางไกลผ่านเครือข่ายคอมพิวเตอร์ ระบบอินเตอร์เนต ที่ช่วยให้การศึกษาค้นคว้าในปัจจุบันเป็นไปได้อย่างสะดวกรวดเร็ว และมีประสิทธิภาพ การรื้อปรับระบบองค์กร หรือที่เราได้ยินกันในชื่อว่า **การรีเอนจีเนียริง** การทำระบบสำนักงานอัตโนมัติ ระบบจดหมายอิเลกทรอนิกส์ ก็ล้วนแล้วแต่เกี่ยวข้องกับระบบเครือข่ายคอมพิวเตอร์ ด้วยกันทั้งสิ้น

ดังนั้นเมื่อมีการใช้ระบบเครือข่ายคอมพิวเตอร์กันอย่างแพร่หลาย จึงเกิดปัญหาเกี่ยวกับการใช้งานระบบเครือข่ายขึ้น จึงต้องมีการปรับแต่งและแก้ไข เพื่อให้ได้ประสิทธิภาพสูงที่สุด

ปริญญานิพนธ์ฉบับนี้ จัดทำขึ้นเพื่อวัตถุประสงค์ 3 ประการ

1. เพื่อให้ความรู้เบื้องต้นในด้านระบบเครือข่าย โดยจะเน้นเฉพาะเครือข่ายแบบอีเทอร์เนต และจะอธิบายอ้างอิงตามมาตรฐานของ OSI ซึ่งจะช่วยให้เข้าใจได้ง่ายขึ้น
2. เพื่อให้ความรู้ในส่วนการทำงานของโพรโตคอล ที่มีการใช้งานอย่างแพร่หลายในเครือข่ายแบบอีเทอร์เนตคือ **ชุดเนตแวร์โพรโตคอล** และ**ชุดอินเตอร์เนตโพรโตคอล** โดยได้เขียนโปรแกรมขึ้นมาทำงานบนระบบเครือข่ายจริง เพื่อให้เห็นภาพการทำงานของโพรโตคอลทั้งสองชัดเจนยิ่งขึ้น
3. สามารถนำความรู้ที่ได้ไปใช้ประโยชน์ เพื่อให้ระบบเครือข่ายคอมพิวเตอร์มีประสิทธิภาพสูงขึ้น

รูปแบบการนำเสนอของปริญญานิพนธ์ฉบับนี้ ได้จัดแบ่งออกเป็นหัวข้อต่างๆ แยกออกเป็นบท โดยแต่ละบทจะมีเนื้อหาดังนี้

บทที่ 2 จะกล่าวถึงข้อมูลทางทฤษฎี โดยในหัวข้อแรกเป็นส่วนของมาตรฐาน OSI ในหัวข้อต่อๆ ไปคือ 2.2-2.5 กล่าวถึงมาตรฐานต่างๆในระดับ 2 ชั้นล่างของ OSI ส่วนในตอนท้าย 2.6-2.8 กล่าวถึงรายละเอียดของชุดเนตแวร์โพรโตคอลและชุดอินเตอร์เนตโพรโตคอล

เหตุที่แบ่งหัวข้อในลักษณะนี้ เพื่อให้ทำความเข้าใจได้ง่าย เนื่องจากว่าทั้งชุดเนตแวร์โพรโตคอลและชุดอินเตอร์เนตโพรโตคอล ไม่ได้กำหนดรายละเอียดในส่วนระดับ 2 ชั้นล่างของ OSI แต่ไปใช้มาตรฐานต่างๆที่ได้มีการพัฒนากันมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3 จะกล่าวถึงการนำ **โปรแกรมแพคเก็ตไดรเวอร์ (Packet Driver)** มาใช้งานเพื่อให้การเขียนโปรแกรมติดต่อกับระบบเครือข่ายทำได้ง่ายขึ้น เนื่องจากจะช่วยให้ไม่ต้องสนใจในรายละเอียดส่วนการควบคุมเนตเวอร์กการ์ด โดยในหัวข้อที่ 3.1 ถึง 3.2 จะอธิบายให้เข้าใจถึงการทำงานของแพคเก็ตไดรเวอร์ หัวข้อที่ 3.3-3.4 จะเป็นการแนะนำวิธีการเขียนโปรแกรมติดต่อกับแพคเก็ตไดรเวอร์ ส่วนในหัวข้อสุดท้ายจะแสดงรูปแบบของข้อมูลที่รับมาได้

บทที่ 4 จะกล่าวถึงการนำเอาข้อมูลที่ได้จากแพคเก็ตไดรเวอร์ มาช่วยเพิ่มประสิทธิภาพการทำงานของระบบเครือข่าย โดยจะกล่าวถึงการนำมาประยุกต์ใช้ในโปรแกรม Ethernet Monitor ด้วย

บทที่ 5 จะเสนอแนวทางในการนำความรู้ที่ได้ ไปใช้ประโยชน์ต่อไป

ในส่วนภาคผนวก ได้รวบรวมข้อมูลที่เป็นประโยชน์ เพื่อนำไปใช้ในการอ้างอิงส่วนอื่นๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2 ทฤษฎีและหลักการ

2.1 ระบบเครือข่ายคอมพิวเตอร์ตามมาตรฐาน OSI

ในยุคต้นๆของระบบเครือข่ายคอมพิวเตอร์ ระบบเครือข่ายมีให้เลือกเพียงไม่กี่ชนิดจากผู้ผลิต ได้แก่ IBM, HP, DEC, HoneyWell อีกทั้งแต่ละระบบยังไม่สามารถเชื่อมต่อกันได้เพราะเป็นแบบปิด (Proprietary) ดังนั้นเมื่อเลือกซื้อระบบใดแล้ว ผู้ซื้อจะต้องผูกติดกับผู้ขายทำให้เกิดปัญหาหลายๆอย่างตามมา

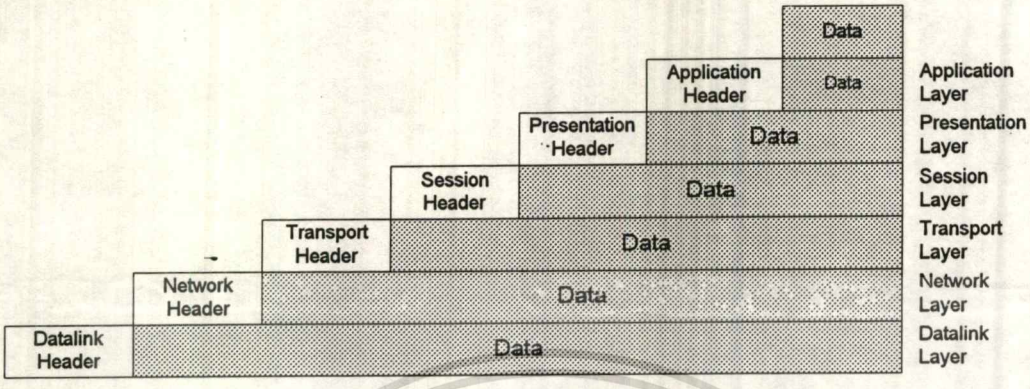
จึงทำให้ระบบ OSI ได้รับการพัฒนาขึ้นในปี ค.ศ.1978 โดย International Organize of Standard (ISO) เพื่อให้มีมาตรฐานในรูปแบบของระบบเปิด (Open System) และเป็นตัววัดความแตกต่างระหว่างระบบการติดต่อสื่อสาร โดยระบบเครือข่ายใดที่พัฒนาได้สอดคล้องกันกับขอบเขตข้อบังคับของ OSI เสมือนกับระบบเหล่านั้นพูดจาภาษาเดียวกัน กล่าวคือใช้การเชื่อมต่อแบบเดียวกัน ทำให้ระบบเครือข่าย จากหลายผู้ผลิตสามารถเชื่อมต่อกันได้ ทำให้มีทางเลือกเพิ่มขึ้น อีกทั้งระบบยังจะมีความยืดหยุ่นสูง เพิ่มการแข่งขันทำให้มีราคาถูกลง

รูปแบบมาตรฐาน OSI ได้กำหนดโดยใช้หลักการ 5 อย่างดังนี้ คือ

1. ชั้น (Layer) จะถูกสร้างขึ้นก็ต่อเมื่อมีความแตกต่างกัน
2. แต่ละชั้นจะต้องถูกกำหนดหน้าที่ไว้อย่างชัดเจน
3. หน้าที่ของแต่ละชั้นจะต้องนิยามได้ในโพรโตคอลมาตรฐาน
4. ขอบเขตของแต่ละชั้นจะต้องมีข้อมูลไหลข้ามผ่านน้อยที่สุด
5. หน้าที่ที่แตกต่างกันจะถูกกำหนดแยกกันในแต่ละชั้น แต่จำนวนชั้นนั้นจะต้องไม่มากเกินกว่าที่สถาปัตยกรรม จะรองรับได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบมาตรฐาน OSI ประกอบไปด้วยชั้นต่างๆ 7 ชั้น ดังต่อไปนี้



รูปที่ 2.1 แสดงการใช้เฮดเดอร์ในระดับชั้นต่างๆ

ชั้นที่ 1 ชั้นฟิสิคัล (Physical Layer)

ในชั้นฟิสิคัล จะเป็นการส่งและรับบิตข้อมูล ผ่านสื่อกลางที่ใช้สื่อสาร หน่วยของข้อมูลในชั้นนี้จะเป็นมีหน่วยเป็น **บิต** โดยสนใจเพียงแต่การรับและการส่งเท่านั้น จะไม่สนใจในความหมายของตัวกลุ่มบิต โดยจะอธิบายรายละเอียดทางกลศาสตร์และหลักการในการส่งบิต อาทิเช่น การกำหนดลักษณะขาสัญญาณของอุปกรณ์ที่ใช้ในการเชื่อมต่อ ขนาดแรงดันไฟฟ้าของสัญญาณที่ใช้

คุณสามารถเทียบการทำงานในชั้นนี้กับการส่งโทรเลข ที่ประกอบด้วยข้อมูลที่เป็นจุดและขีด ซึ่งทั้งจุดและขีดจะเป็นอิสระต่อกัน ไม่สามารถเข้าใจความหมายได้ หากยังไม่รับการแปลความหมายเสียก่อน ซึ่งนั่นจะเป็นหน้าที่ของระดับชั้นที่อยู่สูงขึ้นไป

โพรโตคอลที่รู้จักกันดีในชั้นนี้คือ RS-232

ชั้นที่ 2 ชั้นดาตาลิงค์ (Data Link Layer)

ชั้นดาตาลิงค์ จะรับข้อมูลที่เป็น `1` และ `0` จากชั้นฟิสิคัล มาจำแนกออกเป็นกลุ่มของบิต โดยจะเรียกกลุ่มของบิตเหล่านั้นว่า **เฟรม (Frame)** ในชั้นดาตาลิงค์จะรวมกฎเกณฑ์การควบคุมการขอใช้ระบบเครือข่าย คือเมื่อไรที่เครื่องลูกข่ายสามารถส่งข้อมูลได้ จะจัดการอย่างไรกรณีโหนดเครื่องลูกข่ายล้ม จะตรวจสอบความผิดพลาดที่เกิดขึ้นได้อย่างไร

เราเปรียบการทำงานของชั้นดาตาลิงค์ได้กับการแปลความหมายข้อมูลจุดและขีดในโทรเลข ซึ่งจะต้องแบ่งกลุ่มข้อมูลออกเป็นกลุ่มๆ เพื่อใช้แทนความหมายของคำหรือประโยค ในชั้นนี้จะมีการเพิ่มส่วนเฮดเดอร์ลงไปในตัวข้อมูล โดยมักจะเป็นหมายเลขที่อยู่ (address) ของทั้งฝ่ายส่งและฝ่ายรับข้อมูล มาตรฐานโทเกนริงและอีเทอร์เนตก็มีการเพิ่มเฮดเดอร์ในลักษณะนี้ หมายเลขที่อยู่นี้จะใช้เพื่อการค้นหาเส้นทางให้กับเฟรมข้อมูล และเพื่อให้เครื่องปลายทางทราบแหล่งที่มาของข้อมูลชุดนั้น หมายเลขที่อยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งอยู่ในชั้นนี้มักเป็น *หมายเลขที่อยู่ทางฟิสิคัล (Physical Address)* เพราะมันได้มาจากตัวฮาร์ดแวร์ซึ่งกำหนดโดยบริษัทผู้ผลิตเนตเวอร์กการ์ดนั้นๆ และเก็บถาวรอยู่บน ROM ของการ์ดนั้นเอง หมายเลขที่อยู่นี้จะเป็นหมายเลขเฉพาะของแต่ละเนตเวอร์กการ์ด จะไม่มีหมายเลขใดที่ซ้ำซ้อนกันเลย

ชั้นที่ 3 ชั้นเนตเวอร์ก (Network Layer)

ในชั้นเนตเวอร์ก รับหน้าที่เกี่ยวกับการค้นหาเส้นทางเพื่อส่งข้อมูลในระบบเครือข่ายคอมพิวเตอร์ที่มีความซับซ้อน หน่วยข้อมูลที่เรียกกันในชั้นนี้คือ **แพคเก็ต (Packet)**

ในระบบเครือข่ายที่ไม่ซับซ้อน หมายเลขที่อยู่ทางฟิสิคัล ก็เพียงพอแล้วในการติดต่อกัน แต่ถ้าหากว่าระบบเครือข่ายมีความซับซ้อนมากขึ้น จำเป็นจะต้องมีข้อมูลเพิ่มขึ้น เพื่อช่วยให้การทำงานสะดวกและมีประสิทธิภาพมากขึ้น ระบบเครือข่ายที่มีความซับซ้อนจะเรียกกันว่า **อินเตอร์เนตเวอร์ก (Internetwork)** หรือ **อินเตอร์เนต (Internet)**

ในอินเตอร์เนต จะมีการกำหนด *หมายเลขที่อยู่แบบลอจิคัล (Logical Address)* สำหรับแต่ระบบเครือข่ายย่อย โดยการกำหนดหมายเลขนี้เป็นหน้าที่ของชั้นเนตเวอร์ก เช่นเนตแวร์จะกำหนดหมายเลขที่อยู่แบบลอจิคัลของเครือข่ายย่อยแต่ละเครือข่าย โดยใช้เลขฐานสิบหก จำนวนแปดหลัก

ในชั้นเนตเวอร์ก จะรับผิดชอบหน้าที่การค้นหาเส้นทางให้กับตัวแพคเก็ต ซึ่งจะทำให้ในการทำงานในระดับชั้นที่สูงขึ้นไป ไม่ต้องมากังวลในส่วนนี้อีก ไม่ต้องสนใจรายละเอียดว่าแต่ละแพคเก็ตจะเดินทาง จากตำแหน่งที่อยู่ต้นทาง ไปยังที่อยู่ปลายทางได้อย่างไร

เปรียบเทียบกับระบบโทรเลข ในกรณีที่เป็นการส่งโทรเลขข้ามประเทศ จำเป็นต้องส่งข้อมูลในโทรเลขนั้น ผ่านสถานีย่อยๆไปเรื่อยๆ จากต้นทางจนถึงปลายทาง ผู้ส่งเพียงแต่ระบุปลายทางที่ชัดเจนให้กับเจ้าหน้าที่ที่รับโทรเลขของสถานีต้นทาง หลังจากนั้นก็ปล่อยให้เป็นหน้าที่ของเจ้าหน้าที่ว่าจะส่งต่อข้อมูลดังกล่าว ผ่านไปยังสถานีไหน ข้อมูลนั้นจะถูกส่งผ่านไปเรื่อยๆตามสถานีย่อยระหว่างทาง จนถึงปลายทาง

ตัวอย่างโพรโตคอลที่รู้จักกันในชั้นนี้คือ X.25

ชั้นที่ 4 ชั้นทรานสปอร์ต (Transport layer)

รับผิดชอบด้าน**ความน่าเชื่อถือ (Reliability)** ในการส่งข้อมูลของโพรเซส (Process) ที่ทำงานกันระหว่างเครื่องคอมพิวเตอร์ต้นทางและปลายทาง ย้ำว่าเป็นการติดต่อกันเองของโพรเซส ไม่ใช่เป็นการติดต่อกันกับหมายเลขที่อยู่เนตเวอร์ก ข้อมูลที่อยู่ในชั้นนี้มีหน่วยเป็น **เซกเมนต์ (Segment)**

การทำงานในชั้นนี้จะตรวจสอบข้อมูล ไม่ให้เกิดมีความผิดพลาด, ลำดับการรับส่งข้อมูลจะต้องถูกต้อง, ไม่มีข้อมูลใดๆสูญหาย, หรือซ้ำซ้อน ความน่าเชื่อถือนี้ไม่ได้หมายถึงจะป้องกันไม่ให้เกิดความผิด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พลาดขึ้น แต่หมายถึงการที่เมื่อเกิดมีความผิดพลาดขึ้นแล้ว สามารถรับรู้ได้ทันที ซึ่งจะต้องทำการแก้ไขที่ระดับชั้นนี้ ไม่เช่นนั้นแล้วถ้าผ่านขึ้นไปในระดับชั้นที่สูงขึ้นไปจะทำให้ได้รับข้อมูลที่ผิดๆไป

ระดับชั้นทรานสปอร์ตนี้รับผิดชอบในการนำเอาชุดข้อมูลที่มีความยาวมากๆ มาตัดแบ่งให้มีขนาดพอเหมาะพร้อมที่จะผ่านไปให้ระดับชั้นเนตเวอร์กทำงานได้ รวมถึงควบคุมลำดับการส่งข้อมูล การแก้ไขข้อผิดพลาด การจัดลำดับข้อมูลใหม่ และส่งสัญญาณตอบรับเมื่อได้รับข้อมูลจากเครื่องที่กำลังติดต่ออยู่ด้วย

เจ้าหน้าที่โทรเลข จะเป็นผู้ตรวจสอบว่าเกิดความผิดพลาดในการส่งข้อมูลโทรเลขหรือไม่ อาทิ เช่นเนื้อความไม่สมบูรณ์ หรือไม่ได้รับเครื่องหมายที่ใช้แสดงถึงการสิ้นสุดเนื้อข้อมูล เจ้าหน้าที่จะต้องบอกให้ทำการส่งข้อมูลชุดนั้นๆใหม่

ชั้นที่ 5 ชั้นเซสชัน (Session Layer)

ในชั้นนี้จะดูแลการติดต่อกันของเครื่องคอมพิวเตอร์ 2 เครื่องในเรื่องของ **การสร้าง** (Establishing), **การสอดคล้องกัน** (Synchronizing) **และการบอกเลิกการติดต่อ** (Terminating)

การที่คนเราจะพูดคุยสนทนากันได้นั้น จำเป็นที่จะต้องมีการสร้างข้อกำหนดเช่น กำหนดว่าจะใช้ภาษาใดเป็นสื่อกลาง และจะต้องใช้มารยาททางการสื่อสารที่ถูกต้อง เมื่อสนทนาเสร็จเรียบร้อยแล้ว จะต้องบอกกล่าวเลิกการสนทนา

เจ้าหน้าที่โทรเลข จะปฏิบัติตามกฎเกณฑ์ของการติดต่อที่วางเอาไว้อย่างเคร่งครัด ตั้งแต่ตอนเริ่มส่งข้อมูล ขณะส่งข้อมูล และเมื่อส่งข้อมูลแล้วเสร็จ

การสื่อสารของคอมพิวเตอร์มีลักษณะที่คล้ายคลึงกับของคนเรา ที่จะต้องมีการสร้างการติดต่อก่อน ตกลงกันว่าจะใช้โพรโตคอลอะไร รูปแบบการสื่อสารแบบไหน หากเกิดความผิดพลาดขึ้นแล้ว จะต้องแก้ไขอย่างไร เมื่อไม่ต้องการสื่อสารแล้ว ก็มีขั้นตอนการบอกเลิกการสื่อสารเช่นเดียวกัน

การสื่อสารสามารถทำได้ทั้งแบบ 2 ทิศทางพร้อมกัน (Full-duplex) หรือทีละทิศทาง (Half-duplex) ระดับชั้นเซสชัน จะรับผิดชอบดูแลให้การสื่อสารระหว่างต้นทางและปลายทางเป็นไปอย่างสอดคล้องกัน

โพรโตคอลที่รู้จักกันดีในชั้นนี้คือ Remote Procedure Calls (RPCs) โดยจะนำไปใช้กับทั้งเนตแวร์ และชุดโพรโตคอลอื่นๆอีกมากมาย รวมถึงชุดอินเทอร์เนตโพรโตคอลด้วย

ชั้นที่ 6 ชั้นพรีเซนเตชัน (Presentation Layer)

ระดับชั้นพรีเซนเตชัน จะดูแลให้การสื่อสารระหว่างเครื่องคอมพิวเตอร์ 2 เครื่องถูกต้องตามกฎเกณฑ์ที่วางเอาไว้ งานที่รับผิดชอบส่วนใหญ่จะเป็นการแปลงข้อมูลที่ได้รับจากระดับชั้นล่างซึ่งเป็นรูปแบบมาตรฐาน ไปเป็นรูปแบบข้อมูลที่คอมพิวเตอร์หรือตัวแอพพลิเคชันนั้นๆ นำไปใช้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชั้นพรีเซนเตชัน เปรียบเทียบได้กับการที่มนุษย์มี **ล่าม** ทำหน้าที่แปลภาษาหนึ่งมาเป็นภาษาที่สามารถเข้าใจได้ หน้าที่อื่นๆก็มีอาทิเช่น **การเข้ารหัสข้อมูล** (Data encryption) **และการบีบอัดข้อมูล** (Data compression) แต่ในระดับชั้นอื่นก็อาจมีความสามารถในงานเหล่านี้ได้เช่นเดียวกัน

การที่มีชื่อว่าระดับชั้นพรีเซนเตชัน จึงมักทำให้เกิดความเข้าใจผิดว่าจะดูแลงานการแสดงผลข้อมูลให้กับผู้ใช้ แต่แท้ที่จริงแล้วจะเป็นการแสดงผลข้อมูลให้กับระดับชั้นบน คือชั้นแอพพลิเคชัน

ชั้นที่ 7 ชั้นแอพพลิเคชัน (Application Layer)

ชั้นแอพพลิเคชัน จะเป็นระดับชั้นที่ติดต่อใกล้กับผู้ใช้มากที่สุด ระดับชั้นนี้จะไม่ใช่ระดับชั้นที่ประกอบด้วยโปรแกรมประมวลผลคำ, โปรแกรมทางด้านสเปรดชีต ความเข้าใจผิดนี้น่าจะมีสาเหตุมาจากที่มีชื่อว่า *ชั้นแอพพลิเคชัน* นั่นเอง

แต่ระดับชั้นนี้ จะทำให้การเข้าใช้ระบบเครือข่ายของโปรแกรมที่ผู้ใช้ใช้อยู่ มีรูปแบบที่มีลักษณะเหมือนกัน เช่นโปรแกรมประมวลผลคำ ไม่จำเป็นจะต้องเข้าใจถึงระบบเครือข่าย แต่ชั้นแอพพลิเคชันจะทำให้มองเห็นระบบเครือข่ายมีลักษณะเหมือนกับ DOS ทุกๆโปรแกรมก็จะมองระบบเครือข่ายเปรียบเหมือนเป็น DOS รายละเอียดในส่วนของระบบเครือข่ายจะถูกซ่อนไว้ โพรโตคอลในระดับชั้นนี้ที่รู้จักกันดีคือ NFS ทำให้ผู้ใช้สามารถใช้ไฟล์ที่อยู่บนเครื่องอื่น ในระบบเครือข่ายได้ หน่วยของข้อมูลจะเป็น **ข้อความ** (Message)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Layer	Information Units	Examples	Functions	Hardware
Application	Messages	Telnet S.400, NFS, NCOPY	Application interfaces	Gateways
Presentation		OSI Pres. protocol Sun's XDR, Netware password, encryption	Language translator	Gateways
Session		Sun's RPC OSI Sess. protocol Novell's TTS	Dialog managemet	Gateways
Transport	Segments	TCP, SPX OSI's TP4	Reliable data delivery	Gateways multiplexors
Network	Packets	IP, IPX	Routing over segments, message fragmentation and reassembly	Routers
Data-Link	Frames	Ethernet, Token Ring, ARCnet	Package and unpackage data	Bridges
Physical	Bits	RS232, V.28	Move bits across medium	Repeater

ตารางที่ 2.1 สรุปรายละเอียดมาตรฐาน OSI แยกตามระดับชั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 รายละเอียดที่ควรสนใจในระบบเครือข่ายท้องถิ่น

ในระบบเครือข่ายท้องถิ่นมีสิ่งที่ควรให้ความสนใจดังนี้

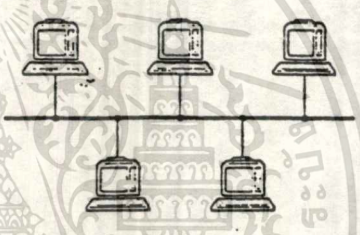
- เนตเวอร์กโทโพโลยี (Network Topology)
- วิธีการแบ่งเวลาใช้สายสัญญาณ (Access Method)
- ลักษณะของสัญญาณ (Signalling Technique)
- ชนิดของสายสัญญาณ (Transmission Medium)

เนตเวอร์กโทโพโลยี

เนตเวอร์กโทโพโลยีในระบบเครือข่ายท้องถิ่น โดยทั่วไปมีอยู่ 3 แบบใหญ่คือ

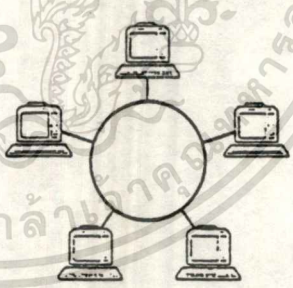
- Bus

  The bus network is the most prevalent LAN configuration, used in both Ethernet (CSMA/CD) and Token Passing bus.



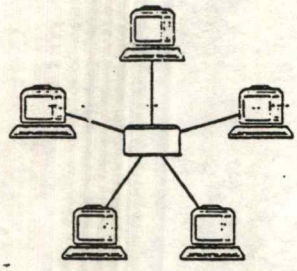
- Ring

  With ring topologies used in Token Passing networks, all networked devices are connected in a closed loop.



- Star

  In a star topology, networked devices are connected directly to a central network device, which is usually active.



รูปที่ 2.2 แสดงเนตเวอร์กโทโพโลยีแบบต่างๆ

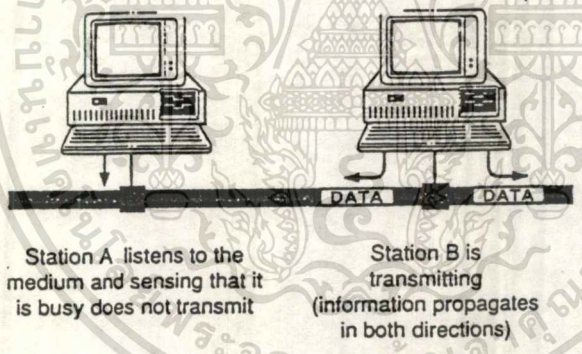
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. **แบบดาว (Star)** คือทุกเครื่องต่อกับอุปกรณ์ที่อยู่ตรงกลางเพียงตัวเดียว เมื่อเครื่องหนึ่งในระบบจะติดต่อกับเครื่องอื่นๆ ก็ต้องผ่านตัวกลางนี้ก่อน
2. **แบบบัส (Bus)** คือใช้สายต่อเครื่องเข้าหาสายใหญ่ที่อยู่ตรงกลางหรือบัส เมื่อเครื่องหนึ่งจะติดต่อกับเครื่องอื่นๆ ก็สามารถส่งข้อมูลออกมาบนบัสและไปถึงอีกเครื่องหนึ่งได้โดยตรง
3. **แบบวงแหวน (Ring)** ตัวกลางจะร้อยเป็นวงผ่านทุกๆเครื่องในระบบจนครบ การส่งข้อมูลจะส่งออกมาในวงแหวน แต่ละเครื่องที่อยู่ระหว่างทางก็จะช่วยๆกันส่งข้อมูลที่ผ่านมาต่อเนื่องกันไปจนกว่าจะวนไปถึงปลายทางที่ต้องการ

วิธีการแบ่งเวลาใช้สายสัญญาณ

เนื่องจากแต่ละเครื่องในระบบเครือข่ายท้องถิ่น ใช้สายสัญญาณชุดเดียวกันในการติดต่อ จึงต้องมีวิธีการที่จะแบ่งเวลาในการใช้สายนี้ให้ทั่วถึง โดยไม่ต้องรอกันนานเกินไป ซึ่งโดยทั่วไปจะมี 2 แบบ คือ

1. CSMA/CD (Carrier Sense Multiple Access / Collision Detection)



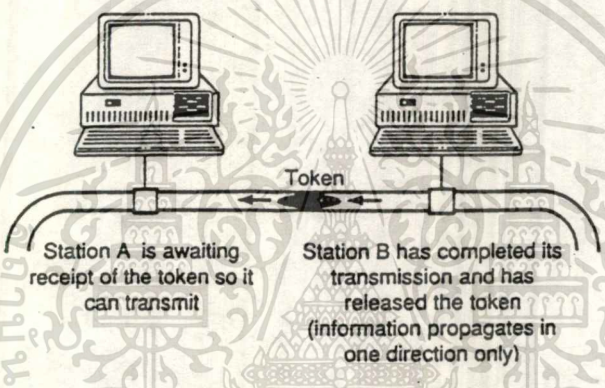
รูปที่ 2.3 แสดงการแบ่งเวลาใช้สายสัญญาณแบบ CSMA / CD

ใช้ในกรณีที่ทุกเครื่องต่ออยู่กับสายชุดเดียวกัน คือสำหรับระบบที่เป็นบัสนั่นเอง โดยที่ขณะใดขณะหนึ่งคอมพิวเตอร์แต่ละเครื่องจะคอย "ฟัง" ว่าสายว่างหรือไม่ (carrier sense) ถ้าว่างก็จะเริ่มส่งสัญญาณออกมา ถ้าสายว่างข้อมูลก็จะไปถึงผู้รับได้เลย แต่การเริ่มส่งสัญญาณนี้อาจเกิดขึ้นจากหลายๆสถานีพร้อมกันได้ ผลก็คือสัญญาณที่ได้จะชนกันในสายทำให้ข้อมูลใช้ไม่ได้ ถึงตรงนี้แต่ละเครื่องจะต้องสามารถตรวจจับการชนกันหรือ collision detection ได้ จากนั้นแต่ละเครื่องที่ส่งข้อมูลออกมาชนกันก็จะหยุดส่งและรอ โดยนับถอยหลังในเวลาที่สุ่มขึ้นมาให้แตกต่างกันระหว่างแต่ละเครื่อง เมื่อครบเวลาที่นับของแต่ละเครื่องแล้วก็ค่อยส่งข้อมูลออกมาใหม่ ซึ่งเนื่องจากเวลาที่ใช้รอแตกต่างกัน การส่งครั้งใหม่จึงไม่มีโอกาสจะชนกันระหว่างคู่เดิมอีก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อดีของ CSMA/CD คือใช้ง่ายและแพร่หลายในระบบเครือข่ายท้องถิ่นทั่วไป โดยเฉพาะที่เป็นอีเทอร์เนต แต่ข้อจำกัดคือมีแนวโน้มที่จะทำงานช้าลงอย่างรวดเร็ว เมื่อจำนวนเครื่องคอมพิวเตอร์ในระบบหรือปริมาณการส่งข้อมูลเพิ่มสูงขึ้นมากๆ เนื่องจากเกิดการชนกันของสัญญาณมาก ทำให้ต้องเสียเวลาส่งใหม่บ่อยครั้ง และช่วงเวลาที่เกิดการชนกันก็จะเสียไปเปล่าๆ ไม่ได้ข้อมูลผ่านออกไป นอกจากนี้ยังไม่สามารถจำกัดช่วงเวลานานที่สุดที่ข้อมูลจะถูกส่งออกไปได้ ว่าหลังจากเริ่มส่งแล้วจะส่งออกไปได้ภายในเวลานานเท่าไร เช่น ภายใน 250 มิลลิวินาที เป็นต้น เพราะอาจเกิดการชนกันกับข้อมูลอื่นๆหลายครั้งจนต้องรอนานกว่านั้นก็ได้ ทำให้ไม่เหมาะกับงานบางประเภทที่ต้องใช้ข้อมูลตามเวลาจริง(real time) เช่น งานควบคุมภาพเคลื่อนไหวที่ต้องการความสม่ำเสมอ

2. Token-Passing



รูปที่ 2.4 แสดงการแบ่งเวลาใช้สายสัญญาณแบบ Token - Passing

ใช้กับการต่อสายหรือโทโพโลยีได้หลายแบบ ไม่ว่าจะเป็นแบบบัส, แบบดาว หรือแบบวงแหวน โดยในขณะใดขณะหนึ่งจะมีคอมพิวเตอร์เพียงเครื่องเดียวในระบบเครือข่าย ที่มีสิทธิในการส่งข้อมูลโดยมีรหัสที่เรียกว่า โทเกน(token) เก็บไว้ เมื่อส่งข้อมูลออกไปเสร็จแล้วก็จะส่งรหัสโทเกนนี้ออกไปให้เครื่องอื่นๆ ตามลำดับที่กำหนดไว้ล่วงหน้า เครื่องอื่นๆเมื่อได้รับรหัสแล้ว ถ้าเครื่องไหนยังไม่ต้องการส่งข้อมูลก็จะส่งรหัสต่อไปเลย แต่ถ้าต้องการส่งข้อมูลก็ให้ส่งข้อมูลออกมาก่อน แล้วค่อยส่งรหัสตามออกไปให้เครื่องอื่นตามลำดับ ด้วยเครื่องจะได้รับสิทธิในการส่งข้อมูล (โดยมีรหัสโทเกนส่งมาถึง) 1 ครั้งภายใน 1 รอบการทำงานหรือ 1 ห้วงเวลาที่กำหนด ทำให้สามารถจำกัดเวลาได้ว่าจะส่งข้อมูลออกได้ไปภายในเวลาไม่เกินกี่ มิลลิวินาที

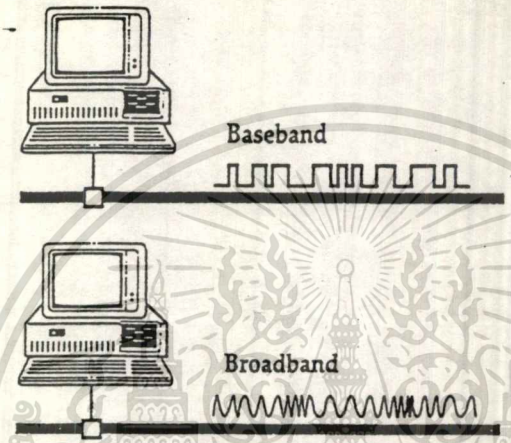
ในทางปฏิบัติระบบเครือข่ายท้องถิ่น ที่มีผู้ผลิตออกจำหน่ายในตลาดจะใช้วิธีการหลายๆ แบบมาประกอบกัน บางแบบอาจใช้โทโพโลยีแบบบัส แต่ใช้การควบคุมแบบโทเกนพาสซิง เช่น อาร์กเนต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(ARCnet) บางแบบอาจใช้บัสร่วมกับ CSMA/CD เช่น อีเทอร์เนต หรือบางแบบก็ใช้แบบวงแหวนร่วมกับโท เกนพาสชิงเช่น โทเกนริง

ลักษณะของสัญญาณ

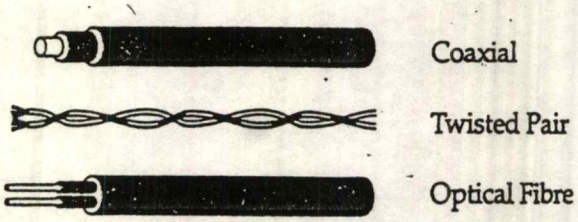
ลักษณะของสัญญาณแบ่งออกได้เป็น 2 ประเภท



รูปที่ 2.5 แสดงลักษณะของสัญญาณ

1. **Baseband** เป็นการส่งสัญญาณดิจิตอลออกไปบนสายสัญญาณโดยตรงเลย ทำให้ในขณะหนึ่งๆ สามารถมีสัญญาณได้เพียงช่อง (Channel) เดียว
2. **Broadband** เป็นการส่งสัญญาณที่เป็นอนาลอกออกไปบนสายสัญญาณ (โดยการมอดูเลตสัญญาณดิจิตอลกับคลื่นพาหะ) ทำให้ในขณะเวลาหนึ่งๆ สามารถที่จะมีข้อมูลของหลายๆงานอยู่บนสายสัญญาณได้ โดยใช้เทคนิคการมอดูเลตแบบความถี่ (FDM)

ชนิดของสายสัญญาณ



รูปที่ 2.6 แสดงชนิดของสายสัญญาณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชนิดของสายสัญญาณที่มักจะนำมาใช้กับระบบเครือข่ายท้องถิ่นคือ

- Broadband coaxial cable
- Baseband coaxial cable
- Twisted pair wire

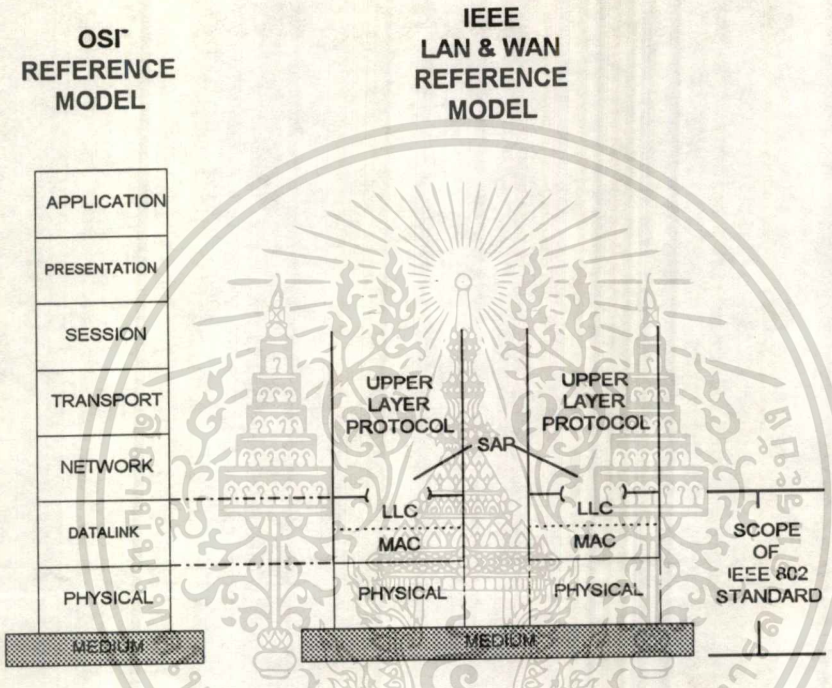
แต่ปัจจุบันได้มีการนำเอาเทคโนโลยีทางด้านสายสัญญาณแบบใยแก้ว (Fiber Optics) เข้ามาใช้กันอย่างแพร่หลาย ซึ่งตัวกลางชนิดนี้มีความสามารถในการรับส่งข้อมูลได้ในความเร็วสูง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 มาตรฐาน IEEE กับระบบเครือข่ายท้องถิ่น

Institute of Electrical and Electronic Engineers (IEEE) ได้ประกาศโครงการ 802 ขึ้นในเดือนกุมภาพันธ์ 1980 โดยครอบคลุมมาตรฐานระบบเครือข่ายท้องถิ่นที่มีความเร็วในการติดต่อไม่เกิน 20 ล้านบิตต่อวินาที และเรียกมาตรฐานดังกล่าวนี้ว่า **"IEEE 802"** หมายเลข 802 ก็คือช่วงเวลาที่ได้ประกาศใช้มาตรฐานนี้



รูปที่ 2.7 แสดงความสัมพันธ์ระหว่างมาตรฐาน OSI กับมาตรฐาน IEEE

จะสังเกตได้ว่า IEEE จะเน้นเฉพาะส่วนที่เกี่ยวข้องกับฮาร์ดแวร์ ซึ่งอยู่ในชั้นฟิสิคัล และดาตาลิงค์ของ OSI อาทิเช่นลักษณะของการ์ดแลน รูปแบบการเชื่อมต่อสาย

มาตรฐาน IEEE ได้แบ่งชั้นดาตาลิงค์ของ OSI ออกเป็น 2 ชั้นย่อยคือ ชั้นล่างจะเป็น **Media Access Protocol (MAC) และชั้นบนจะเป็น Logical Link Control(LLC)** ในชั้น MAC จะดูแลควบคุมวิธีแบ่งกันใช้งานอุปกรณ์ทางฮาร์ดแวร์ที่จำเป็นต้องใช้ร่วมกัน มาตรฐานโทเกนริง และมาตรฐานอีเทอร์เนต ต้องใช้ MAC ที่แตกต่างกันเนื่องจากว่าโทเกนริงและอีเทอร์เนต มีวิธีการแบ่งการใช้งานฮาร์ดแวร์ที่แตกต่างกัน

แต่ทุกๆมาตรฐานของ IEEE จะใช้ชั้น LLC เหมือนกันหมดคือ **IEEE 802.2** การใช้มาตรฐานเดียวกันนี้คือใช้ IEEE 802.2 กับระบบเครือข่ายทุกรูปแบบมีข้อได้เปรียบอย่างหนึ่งคือ โพรโตคอลที่อยู่ใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชั้นบนขึ้นไป ไม่ต้องสนใจว่ามาตรฐานในชั้นล่างจะมีรูปแบบเป็นเช่นไร การติดต่อกับโพรโตคอลในชั้นล่างจะใช้รูปแบบเดียวกันหมด การใช้งานสามารถทำได้ง่ายขึ้น

จะสังเกตได้อีกอย่างหนึ่งว่าการติดต่อกับโพรโตคอลชั้นบนของ LLC มาตรฐาน IEEE จะใช้ผ่าน **Link Service Access Points (LSAPs)** ตัว LSAPs จะเป็นที่อยู่ทางลอจิคัลของชั้นดาตาลิงค์ หนึ่งที่อยู่ ทางฟิสิคัล (ที่อยู่ MAC) สามารถจะมีที่อยู่ LSAPs ได้หลายตัว ทำให้สามารถติดต่อกันได้ในแบบหลายจุด (Multiple end-point connection) ระหว่างโหนด 2 โหนดบนระบบเครือข่าย

ชั้น LLC สามารถเลือกได้ว่าจะให้เป็นการติดต่อแบบ**วงจรเสมือน (Virtual Circuit)** ซึ่งเป็นแบบ Connection-Oriented หรือแบบ**ดาตาแกรม (Datagram)** ซึ่งเป็นแบบ Connectionless-Oriented หรืออาจจะใช้แบบผสมกันทั้ง 2 แบบก็ได้ แบ่งออกได้เป็น 3 แบบดังนี้

แบบที่ 1 ดาตาแกรมในรูปแบบ Unacknowledge ทุกๆแพคเก็ตที่ส่งออกไป จะมีข้อมูลที่อยู่ครบถ้วนทั้งที่อยู่ของทางฝ่ายรับ และที่อยู่ของทางฝ่ายส่ง ไม่มีการตรวจสอบว่าแพคเก็ตที่ส่งออกจะส่งถึงผู้รับหรือไม่ ลำดับของแพคเก็ตที่ส่งไปนั้น เมื่อถึงมือผู้รับแล้วจะถูกต้องหรือไม่ ทำงานได้ทั้งในแบบจุดต่อจุด (point-to-pont) แบบหลายจุด (Multipoint) และแบบกระจายทุกจุด (Broadcast)

แบบที่ 2 แบบวงจรเสมือน มีการควบคุมการส่งแพคเก็ต ควมคุมลำดับของแพคเก็ตที่จะต้องตรงกันทั้งฝ่ายรับและฝ่ายส่ง มีการป้องกันการเกิดความผิดพลาด ก่อนที่จะส่งข้อมูลในรูปแบบนี้ ต้องสร้างการติดต่อขึ้นมาก่อน และเมื่อการติดต่อจะสิ้นสุดลง จะต้องบอกเลิกการติดต่อเช่นเดียวกัน

แบบที่ 3 ดาตาแกรมในรูปแบบ Acknowledge เป็นรูปแบบผสมระหว่างแบบดาตาแกรมและแบบวงจรเสมือน โดยเมื่อเกิดความผิดพลาดในการส่งข้อมูล จะแก้ไขด้วยการให้ส่งข้อมูลชุดนั้นใหม่

Logical Link Control (LLC)	IEEE 802.2 Type 1 Unacknowledged, Datagram service Type 2 Virtual - Circut service Type 3 Acknowledged, Datagram service					
Medium Access Control (MAC)	IEEE 802.3	CSMA/CD Medium Access Control	IEEE 802.4	Token-Bus Medium Access Control	IEEE 802.5	Token-Ring Medium Access Control
Physical Medium		Baseband Coaxial 1,10 MBPS Baseband Twisted Pair 10 MBPS Broadband Coaxial 10 MBPS		Broadband Coaxial 1,5,10 MBPS Carrier band 1,5,10 MBPS Optical Fiber 5,10,20 MBPS		Shielded Twisted Pair 1, 4 MBPS Unshielded Twisted Pair

รูปที่ 2.8 แสดงตัวอย่างมาตรฐาน IEEE แบ่งออกเป็นระดับชั้นต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IEEE Standard	Meaning
IEEE 802.1	LAN bridging
IEEE 802.2	Logical Link Control (LLC)
IEEE 802.3	Standardization of Ethernet Technology, Includes 100BASE-TX, 100BASE-TF, and 100BASE-T4(Fast Ethernet)
IEEE 802.4	Token Bus standard
IEEE 802.5	Token Ring standard
IEEE 802.6	Metropolitan Area Network (MAN)
IEEE 802.7	Broadband technical advisory
IEEE 802.8	Fiber-optic technical advisory
IEEE 802.9	Integrated Voice/Data (VID)
IEEE 802.10	LAN security
IEEE 802.11	Wireless LANs
IEEE 802.12	100BASE-VG (100VG-anyLAN)

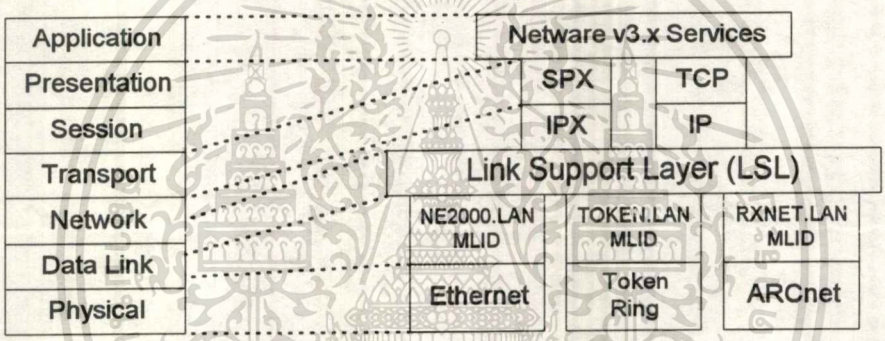
ตารางที่ 2.2 แสดงมาตรฐาน IEEE แบบต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 รูปแบบของ ODI กับมาตรฐาน OSI

ก่อนอื่นจะต้องเข้าใจว่าไม่ว่าจะเป็นมาตรฐานของ ODI , เนตแวร์ หรือของผู้ค้ารายใดๆ ก็ไม่สามารถที่จะจัดแบ่งระดับชั้นตามรูปแบบของ OSI ได้อย่างสมบูรณ์เลย

ODI มีความสามารถทำให้เนตเวอร์กการ์ดหลากหลายรุ่น หลากหลายผู้ค้า สามารถใช้งานได้กับหลากหลาย**โพรโตคอลแสตกค์ (Protocol Stack)** ทั้ง TCP/IP, OSI, SPX/IPX หรือ Apple Talk โดยก่อนหน้าที่จะมีรูปแบบ ODI หรือมีการใช้ NDIS แพคเก็ตไดรเวอร์(หรือที่ปัจจุบันเรียกกันว่า Crynwr ไดรเวอร์) จำเป็นที่จะต้องใช้ไดรเวอร์ของเนตเวอร์กการ์ดสำหรับแต่ละโพรโตคอลแสตกค์ ทำให้เป็นการยากที่จะใช้หลายๆไดรเวอร์ในเครื่องๆเดียว ส่งผลให้ยากต่อการทำให้เครื่องหนึ่งเครื่องทำงานได้มากกว่า 1 โพรโตคอลแสตกค์



รูปที่ 2.9 แสดงการเปรียบเทียบมาตรฐาน OSI กับรูปแบบของ ODI

หลักสำคัญของ ODI คือการใช้ **Link Support Layer (LSL)** และ **Multiple Link Interface Driver (MLID)** จากรูปจะสังเกตได้ว่ามาตรฐานอีเทอร์เนต, โทเกนริงและอาร์กเนต จะสัมพันธ์กับการทำงานในชั้นที่ 1 และ 2 ของ OSI โดยมี NE2000.LAN, TOKEN.LAN และ RXNET.LAN เป็นชื่อของไดรเวอร์ MLID ของแต่ละเนตเวอร์กการ์ด และมีการทำงานสัมพันธ์กับชั้นที่ 2 คือดาตาลิงค์ของ OSI ไดรเวอร์เหล่านี้ติดต่อโดยตรงกับ LSL ตัว LSL จะไม่สอดคล้องกับรูปแบบของ OSI ทีเดียวนัก เพราะทำงานครอบคลุมระหว่างชั้นดาตาลิงค์และเนตเวอร์ก จะสังเกตได้ว่า LSL จะเป็นตัวเชื่อมต่อกันระหว่างโพรโตคอลในชั้นบนกับมาตรฐานแบบต่างๆในชั้นล่าง

หลักสำคัญของ ODI อยู่ที่ LSL มันช่วยทำให้มองเห็นฟิสิคัลเนตเวอร์กการ์ด เป็นเนตเวอร์กการ์ดเสมือน การทำงานที่อยู่ในระดับชั้นที่สูงขึ้นไป ไม่จำเป็นต้องเข้าใจในรายละเอียดส่วนการทำงานในชั้นด้านล่าง ไม่ต้องแยกแยะวิธีการใช้งานที่แตกต่างกันในเนตเวอร์การ์ดแต่ละชนิด การใช้งานมองเห็นเป็นรูปเดียวกันหมด เป็นรูปแบบมาตรฐาน หากมีการผลิตเนตเวอร์กการ์ดแบบใหม่ออกมา เพียงแต่เขียน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไดรเวอร์ให้เข้ากันได้กับ LSL ก็เป็นอันเรียบร้อย ตัวรายละเอียดที่อยู่ในชั้นสูงขึ้นไปไม่ต้องการการแก้ไขอย่างไร

ในตัว MLID แต่ละตัว ก็จะใช้งานได้หลายโพรโตคอล ตราบใดก็ตามที่โพรโตคอลเหล่านั้น ยังใช้งานผ่านตัว LSL อยู่ โดย MLID ก็จะไปใช้งานโพรโตคอลเหล่านั้นโดยเรียกผ่านตัว LSL อีกทีหนึ่ง

ในกรณีที่ได้รับข้อมูลมา ตัวไดรเวอร์จะส่งข้อมูลนั้นให้กับ LSL ทันที ไม่ต้องมีการจำแนกอย่างไร ปล่อยให้เป็นหน้าที่ของ LSL ทำการจำแนกข้อมูลนั้น ส่งไปให้โพรโตคอลแสตกค์ที่สอดคล้องกัน ที่ทำเช่นนี้ได้เพราะว่า LSL จะเก็บข้อมูลของทั้งตัว MLIDs และโพรโตคอลแสตกค์ ที่มันทำงานด้วย โดยเมื่อไม่ว่า MLIDs ตัวใดถูกเรียกใช้ หรือโพรโตคอลแสตกค์ใดถูกเรียกใช้ จะมีการเก็บ **ค่าหมายเลขทางลอจิคัล** ไว้ที่ตัว LSL ในส่วนของดาตาเซกเมนต์ (Data Segment)

ส่วนของ LSL นี้จะเป็นของแต่ละรูปแบบระบบปฏิบัติการ หมายความว่า LSL ของระบบปฏิบัติการหนึ่งๆไม่สามารถใช้ได้กับระบบปฏิบัติการอื่นๆ ใน ดอส, โอเอสทู, เนตแวร์ ต่างก็ต้องมี LSL เฉพาะที่สัมพันธ์กับระบบปฏิบัติการของตนเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 ระบบเครือข่ายแบบอีเทอร์เน็ต

ระบบเครือข่ายแบบอีเทอร์เนต เป็นการใช้งานระบบเครือข่ายตามมาตรฐาน IEEE 802.3 ที่มีการส่งข้อมูลแบบเบสแบนด์ ความเร็ว 10 ล้านบิตต่อวินาที มีโทโพโลยีแบบบัส และมีวิธีการใช้สายสัญญาณในแบบ CSMA / CD ข้อมูลที่ถูกส่งในระบบเครือข่ายอีเทอร์เนต จะอยู่ในรูปแบบของเฟรม 2 ประเภทคือ เฟรมอีเทอร์เนต และ เฟรม IEEE 802.3

Ethernet

8 bytes	6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes
Preamble	Dest. Address	Source Address	Type	Data	CRC

IEEE 802.3

7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes
Preamble	Start of Frame	Dest. Address	Source Address	Length	802.2 Header and data	CRC

รูปที่ 2.10 แสดงลักษณะของเฟรมอีเทอร์เนต และ เฟรม IEEE 802.3

Preamble and Start of Frame Delimiters

ในส่วนของฟิลด์นี้ ในเฟรมอีเทอร์เนตจะใช้ 8 ไบต์ เป็น Preamble ทั้งหมดโดยจะเป็นชุดของตัวเลข 1 และ 0 สลับกัน (10101010)

ส่วนของในเฟรมของ IEEE 802.3 จะใช้เพียง 7 ไบต์เท่านั้นที่เป็น Preamble ส่วนในไบต์ที่ 8 จะเรียกว่าเป็น Start of Frame Delimiters โดยใน 2 บิตสุดท้ายของไบต์นี้จะเป็น 11 (10101011)

Destination Address

จะเป็นหมายเลขที่อยู่ทางฟิสิคัล ของเนตเวอร์กการ์ดปลายทาง โดยหมายเลขปลายทางนี้ยังสามารถที่จะระบุได้เป็นแบบจุดต่อจุด แบบหลายจุด หรือแบบกระจายทุกจุด

Source Address

จะเป็นหมายเลขที่อยู่ของเนตเวอร์กการ์ดต้นทาง มีขนาด 6 ไบต์ โดย 3 ไบต์แรกนั้น IEEE จะเป็นผู้กำหนดให้กับผู้ผลิตแต่ละรายไป และอีก 3 ไบต์หลังนั้นผู้ผลิตจะเป็นผู้กำหนดเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
037092

Length Versus Type

ในเฟรม IEEE 802.3 จะใช้เป็นฟิลด์ความยาว (length) ใช้ระบุความยาวเป็นจำนวนไบต์ของข้อมูลที่อยู่ในฟิลด์ข้อมูล แต่ในเฟรมอีเทอร์เนต จะใช้ระบุประเภท (type) ของเฟรมนั้นๆ

Data and Padding

ในเฟรม 802.3 จะใช้ฟิลด์นี้ในการเก็บข้อมูล โดยตัวเลขที่อยู่ในฟิลด์ความยาว จะระบุขนาดของข้อมูลที่อยู่ในฟิลด์นี้ ถ้าหากว่าขนาดของข้อมูลนั้นสั้นเกินไป คือความยาวในฟิลด์นี้ไม่ถึง 46 ไบต์ (ซึ่งจะทำให้ขนาดของเฟรมไม่ถึง 64 ไบต์) ทำให้เมื่อเกิดข้อผิดพลาดขึ้นกับเฟรมนั้นๆ หรือหากเกิดการชนกันของข้อมูลจะไม่สามารถตรวจพบได้ทัน ก็จะมีการใส่ค่าเพิ่ม (padding) ให้กับฟิลด์นี้

Frame Check Sequence

ใช้ในการตรวจเช็คความถูกต้อง ของเฟรมข้อมูล โดยใช้วิธีการ CRC

เนตเวอร์กการ์ดแต่ละตัวในระบบเครือข่ายอีเทอร์เนต จะมีหมายเลขที่อยู่ขนาด 6 ไบต์ ซึ่งหมายเลขนี้จะได้รับการกำหนดมาจากโรงงานที่ผลิตอุปกรณ์นั้นๆเลย โดยการควบคุมของ IEEE นิวยอร์ค สหรัฐอเมริกา

เมื่อเนตเวอร์กการ์ดหนึ่งๆได้ส่งเฟรมข้อมูลเข้าไปในระบบเครือข่ายแล้ว จะต้องรออย่างน้อย 9.6 ไมโครวินาที ก่อนที่จะมีการส่งข้อมูลชุดต่อไป ซึ่งช่วงเวลานี้เรียกว่า Inter-Packet-Gab (IPG) ซึ่ง IPG จะช่วยให้มีเวลาเตรียมพร้อมที่จะรับข้อมูล โดยที่ไม่เกิดปัญหา ถ้าเนตเวอร์กการ์ดพยายามที่จะส่งข้อมูล แต่พบว่าขณะนั้นสายสัญญาณยังไม่ว่างจะทำการรอ หรือถ้าหลังจากที่ส่งข้อมูลไปแล้ว พบว่าข้อมูลชุดนั้นเกิดการชนกัน ก็จะทำการรอเป็นช่วงเวลาที่สุ่มขึ้นมา ก่อนที่จะส่งข้อมูลชุดนั้นใหม่ ในการส่งข้อมูลแต่ละชุดนั้น จะพยายามส่งสูงสุดไม่เกิน 16 ครั้ง ถ้าหากการส่งยังไม่เป็นผลสำเร็จ จะรายงานความผิดพลาด

ความต้องการขั้นพื้นฐานของระบบเครือข่ายอีเทอร์เนต คือเมื่อเกิดการชนกันของข้อมูล ทุกๆเครื่องลูกข่ายจะต้องรู้ถึงการชนกันนั้น ซึ่งเป็นตัวกำหนดให้ความยาวของเฟรมต่ำสุดที่จะทำให้เครื่องลูกข่ายสามารถรับข้อมูลได้คือ 576 บิต (72 ไบต์) และสูงสุดไม่เกิน 12,208 บิต (1526 ไบต์)

ถ้าขนาดเฟรมข้อมูลที่มีการส่งทุกเฟรม มีความยาวต่ำสุดเท่าที่มีได้จะสามารถส่งได้ด้วยความเร็ว 7,619,047 บิตต่อวินาที หรือส่งด้วยความสามารถ (Utilization) 76 % แต่ถ้าทุกเฟรมใช้ความยาวสูงสุดแล้ว จะสามารถส่งได้ด้วยความเร็ว 9,523,212 บิตต่อวินาที หรือส่งด้วยความสามารถ 95.23 %

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มาตรฐานการส่งข้อมูลของ IEEE 802.3 ที่ได้รับการปรับปรุงใหม่

เดิมทีเดียวมาตรฐานอีเทอร์เนตจะใช้สายส่งข้อมูลแบบโคแอกเชียล โดยมี 2 แบบคือ

- สายส่งข้อมูลแบบบาง (RG-58 , 50 โอห์ม) สามารถใช้กับเซกเมนต์ที่มีความยาวไม่เกิน 185 เมตร
- สายส่งข้อมูลแบบหนา (RG-11 , 50 โอห์ม) สามารถใช้กับเซกเมนต์ที่มีความยาวไม่เกิน 500 เมตร ถ้าใช้อุปกรณ์ทวนสัญญาณ (Repeater) จะทำให้สามารถเพิ่มความยาวได้ออกไปถึง 3,000 เมตร

ตามมาตรฐานการส่งข้อมูล IEEE 802.3 ที่ปรับปรุงใหม่ขึ้นนี้ จะส่งข้อมูลด้วยความเร็ว 10 ล้านบิตต่อวินาที ยกเว้นมาตรฐาน 1BASE5 ที่มีการส่งข้อมูลด้วยความเร็วเพียง 1 ล้านบิตต่อวินาที แต่มีระยะทางไกลกว่า มาตรฐานที่ได้รับความนิยมมากได้แก่ 10Base5, 10Base2 และ 10BaseT ตามมาตรฐานที่ได้รับการปรับปรุงใหม่มีดังนี้

10Base5 เป็นการส่งข้อมูลโดยใช้สายส่งข้อมูลแบบ RG-11 หรือสายโคแอกเชียลแบบหนา ความยาวสูงสุดของเซกเมนต์ 500 เมตร (1640 ฟุต) สามารถต่อตัวทรานซีฟเวอร์ (Transceiver) ในแต่ละเซกเมนต์ได้สูงสุด 100 ตัว จำนวนสูงสุดของ DTE ในระบบเครือข่ายคือ 1,024 ตัว ระยะห่างต้องไม่ต่ำกว่า 2.5 เมตร การส่งข้อมูลเป็นแบบ Baseband

10Base2 เป็นการส่งข้อมูลโดยใช้สายส่งข้อมูลแบบ RG-58A/U หรือสายโคแอกเชียลแบบบางนั่นเอง ความยาวสูงสุดของเซกเมนต์ต้องไม่เกิน 185 เมตร (600 ฟุต) สามารถต่อตัวทรานซีฟเวอร์ได้สูงสุด 30 ตัว และระยะห่างระหว่างตัวทราฟซีสเวอร์กับเซกเมนต์ต้องไม่ต่ำกว่า 0.5 เมตร การส่งข้อมูลเป็นแบบ Baseband

10BaseT ใช้สายส่งข้อมูลแบบ 20 ถึง 24 AWG UTP หรือสายส่งข้อมูลแบบคู่บิตเกลียว และใช้ตัวเชื่อมต่อแบบ RJ-45 ระยะห่างระหว่างตัวทรานซีฟเวอร์กับฮับ (Hub) จะต้องไม่ยาวเกินกว่า 100 เมตร (328 ฟุต) ฮับ 1 ตัวมีเครื่องลูกข่ายต่ออยู่ได้ไม่เกิน 1023 เครื่อง

10Broad36 ใช้สายส่งข้อมูลแบบโคแอกเชียล (RG-59/U CATV Type) มีความยาว แต่ละเซกเมนต์สูงสุด 3,600 เมตร ใช้วิธีการส่งข้อมูลแบบ Broadband

1Base5 ใช้สายส่งข้อมูลแบบคู่บิตเกลียวที่มีความยาวเซกเมนต์สูงสุดไม่เกิน 500 เมตร ส่งข้อมูลด้วยความเร็ว 1 ล้านบิตต่อวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Rate	Topology	Media	Segment Length in Meters	Data MBPS
Ethernet	BUS	50 ohm thick coax	500	10
10Base5	BUS	50 ohm thick coax	500	10
10Base2	BUS	50 ohm thin coax	185	10
10BaseT	STAR	Unshielded twisted pair	100	10
10Broad36	BUS	75 ohm coax	1800	10
1Base5	STAR	Unshielded twisted pair	250	1

ตารางที่ 2.3 แสดงรายละเอียดมาตรฐานแบบต่างๆของ IEEE 802.3

การตั้งชื่อมาตรฐานต่างๆของ IEEE 802.3 นี้เป็นการตั้งเพื่อให้สื่อความหมาย โดยตัวเลขด้านหน้าสุด จะแสดงความเร็วในการส่งผ่านข้อมูลบนตัวกลางชนิดนั้นๆ ส่วนคำที่อยู่ตรงกลางจะหมายถึงลักษณะของสัญญาณ (Baseband & Broadband) ส่วนตัวสุดท้ายคือความยาวที่มากที่สุดที่จะสามารถเชื่อมต่อได้ของตัวกลางชนิดนั้นๆโดยใช้หน่วยเป็น 100 เมตร

หมายเหตุ ***T ใน 10BaseT*** *หมายถึง Twisted Pair*

36 ใน 10Broad36 *ได้มากจากข้อกำหนดที่ว่า 10Broad36 จะต้องประกอบด้วย 2 เซกเมนต์ และแต่ละเซกเมนต์จะมีความยาวมากที่สุดได้ไม่เกิน 1,800 เมตร*

2 ใน 10Base2 *ได้มาจากการประมาณค่าของค่าจากค่าจริงๆคือ 185 เมตร จึงประมาณเป็น 200 เมตรแทน*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 ชุดอินเตอร์เนตโพรโตคอล

ชุดอินเตอร์เนตโพรโตคอล (Internet Protocol Suite) นั้น คนทั่วไปมักจะรู้จักกันในนามของ**โพรโตคอล TCP/IP (Transmission Control Protocol and Internet Protocol)** ทั้งๆที่ในความเป็นจริงแล้ว การใช้งานของโพรโตคอลในกลุ่มนี้ประกอบด้วยโพรโตคอลมากมาย ไม่ได้มีเพียงแค่ 2 ชนิดนี้เท่านั้น

ถ้าหากคุณได้ศึกษาในรายละเอียดของอินเตอร์เนตโพรโตคอลแล้ว จะสังเกตได้ว่า**ไม่มี**การกำหนดรายละเอียดในส่วนของมาตรฐานใน 2 ชั้นล่างคือ **ชั้นดาตาลิงค์ และชั้นฟิสิคัล** เลย ทำให้ชุดอินเตอร์เนตโพรโตคอล สามารถทำงานได้กับระบบเครือข่ายหลายๆชนิด ไม่ว่าจะเป็นอีเทอร์เนต, โทเกนริง, โทเกนบัส, HDLC หรือ X.25 แต่อาจจะต้องมีการปรับแต่งอะไรเล็กน้อย ด้วยความสามารถนี้เองที่ทำให้ชุดอินเตอร์เนตโพรโตคอล ได้รับการยอมรับความสามารถในการการรวมหลายๆระบบ เข้าไว้ด้วยกัน

มีการนำชุดอินเตอร์เนตโพรโตคอลมาใช้ในการเชื่อมระบบเครือข่ายท้องถิ่นเข้าด้วยกัน เพราะสามารถใช้งานได้กับมาตรฐานหลายชนิด อีกทั้งมีความสามารถทำงานได้หลายอย่างไม่ว่าจะเป็น File Tranfer Protocol (FTP), Simple Mail Tranfer Protocol (SMTP) และ Terminal Service (Telnet) ปัจจุบันชุดอินเตอร์เนตโพรโตคอล ได้เป็นมาตรฐานของระบบ Interconnection Lan ทั้งในหน่วยงานของรัฐและเอกชน

ประวัติความเป็นมาของชุดอินเตอร์เนตโพรโตคอล

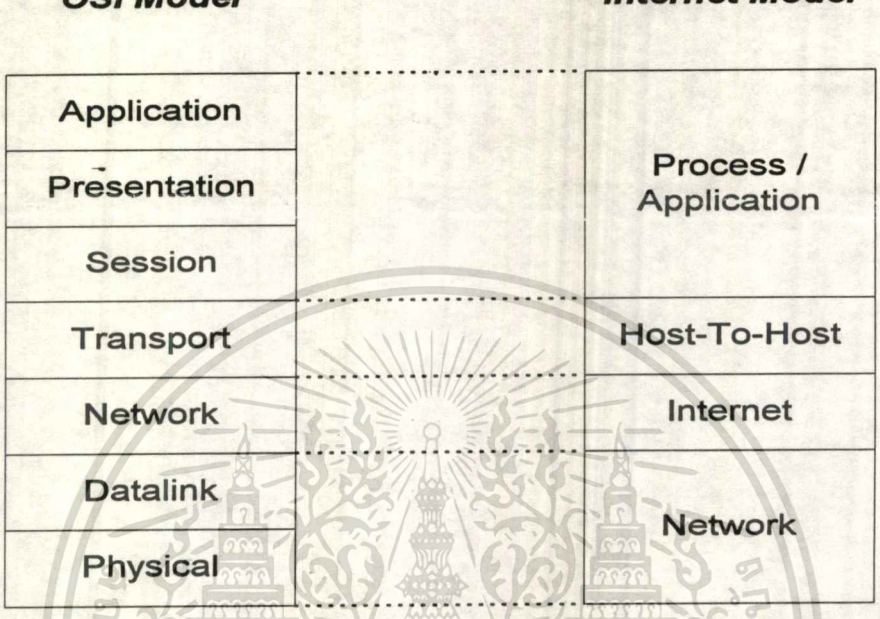
ในช่วงปลายทศวรรษที่ 1960 มหาวิทยาลัยในสหรัฐอเมริกา และสถาบันวิจัยต่างๆ มีความต้องการที่จะเชื่อมต่อระบบเครือข่ายคอมพิวเตอร์เข้าด้วยกัน เพื่อใช้ติดต่อส่งข้อมูลข่าวสารระหว่างกัน แต่ติดขัดปัญหาที่ระบบคอมพิวเตอร์ของแต่ละหน่วยงานมีความแตกต่างกัน สถาบันวิจัยโครงงานขั้นสูง (Advanced Research Project Agency) หรือ ARPA จึงจัดตั้ง ARPANET ขึ้น เพื่อศึกษาแนวทางและออกแบบพัฒนารูปแบบในการเชื่อมต่อระบบเครือข่ายที่มีอยู่ในแต่ละหน่วยงานเข้าด้วยกัน โดยเริ่มแรกนั้นใช้รูปแบบของ packet switching โดยใช้โครงข่ายโทรศัพท์เป็นช่องทางการเชื่อมต่อหลัก

ARPA รู้จักกันในนามของ DARPA (Department of Defence Advanced Research Project Agency) เพราะก่อตั้งโดยรัฐบาลสหรัฐ เพื่อค้นคว้างานทางด้านที่เกี่ยวกับการทหาร ซึ่งในปัจจุบันนี้ ARPANET ก็ได้เป็นเครือข่ายย่อยหนึ่งใน ระบบเครือข่ายอินเตอร์เนต พร้อมกับเครือข่ายย่อยๆอื่นๆ ทางด้านงานวิจัยและทางอุตสาหกรรมอีกมากมายทั่วโลก

ส่วนหนึ่งที่ทำให้ชุดอินเตอร์เนตโพรโตคอล ได้รับการยอมรับอย่างแพร่หลายคือ ได้มีการนำเอาผลการวิจัยและการพัฒนา ออกเผยแพร่ให้กับบุคคลที่ต้องการในนามของ 4.2 BSD (Berkeley System Distribution) โดยผู้ใช้ไม่ต้องเสียค่าใช้จ่ายใดๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เปรียบเทียบรายละเอียดของชุดอินเตอร์เนตโพรโตคอล กับรูปแบบของ OSI



รูปที่ 2.11 แสดงการเปรียบเทียบมาตรฐาน OSI กับชุดอินเตอร์เนตโพรโตคอล

ชั้นที่ 1: Network

จะครอบคลุมการทำงานในส่วน 2 ชั้นล่างของ OSI คือชั้นฟิสิคัล และดาตาลิงค์ โดยข้อกำหนดจะขึ้นอยู่กับชนิดของระบบเครือที่ใช้ อาทิเช่น อีเทอร์เนต , X.25

ชั้นที่ 2: Internet

จะครอบคลุมการทำงานในส่วนชั้นเนตเวอร์กของ OSI โดยจะมีโพรโตคอล IP และ ICMP ทำงานในชั้นนี้

ชั้นที่ 3: Host-to-Host

จะครอบคลุมการทำงานในส่วนชั้นทรานสปอร์ตของ OSI และรวมถึงการทำงานบางอย่างของชั้นเซสชันด้วย มีโพรโคคอล TCP และ UDP อยู่ในชั้นนี้

ชั้นที่ 4: Process/Application

เป็นชั้นสูงที่สุด ครอบคลุมหน้าที่การทำงานของชั้นแอพพลิเคชัน, ชั้นพรีเซนเตชัน และบางส่วนของชั้นเซสชัน โดยมี TELNET (Virtual Terminal Emulation), FTP (File Transfer Protocol) ,TFTP (Trivial File Transfer Protocol) และ SMTP (Simple Mail Transfer Protocol) ทำงานในชั้นนี้

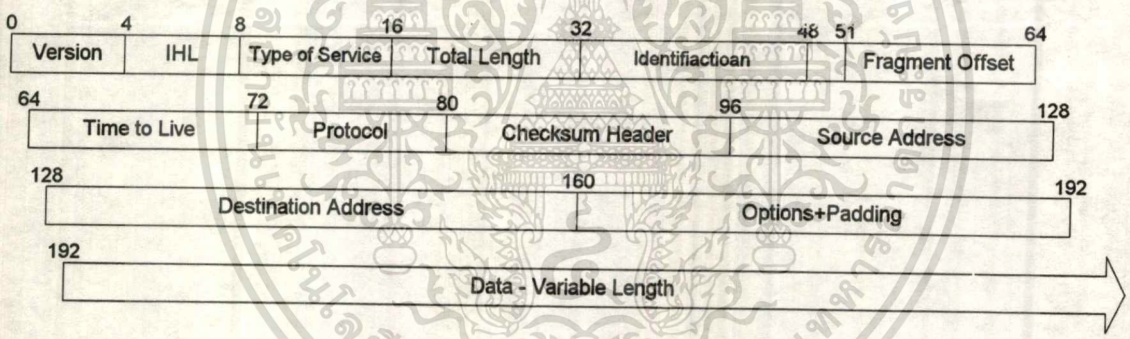
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IP แพคเก็ต

IP เป็นโพรโตคอลที่อยู่ในชั้นเนตเวอร์กของ OSI หรือก็คือชั้นอินเตอร์เนตของชุดอินเตอร์เนตโพรโตคอล ทำหน้าที่รับผิดชอบเกี่ยวกับการค้นหาเส้นทาง(Routing) บนระบบเครือข่าย และหากในกรณีแพคเก็ต มีขนาดใหญ่เกินไป IP ก็จะทำหน้าที่แบ่งแพคเก็ตนั้นๆ ออกเป็นชิ้นย่อยๆ (fragments) แล้วจึงส่งผ่านไปยังระบบเครือข่ายต่อไป และเมื่อไปถึงปลายทาง จึงทำการจัดเรียงลำดับที่ถูกต้องใหม่ เนื่องจากว่าแต่ละแพคเก็ตที่ส่งมานั้น สามารถใช้เส้นทางคนละเส้นทางได้ โดยในส่วนตัวแพคเก็ตเองจะมีส่วนที่ใช้ในการจัดเรียงลำดับของข้อมูล คืนตามรูปแบบเดิมรวมไว้ด้วย

แต่การค้นหาเส้นทางของแพคเก็ต จะต้องใช้ข้อมูลจาก**ตารางการค้นหาเส้นทาง (Routing Information Table)** ซึ่งในการสร้างและปรับปรุงค่าต่างๆ ในตาราง จะต้องใช้โพรโตคอลชนิดอื่นเข้าช่วย ซึ่งจะขอยกตัวอย่างเพียงโพรโตคอลเดียวคือ **Routing Information Protocol (RIP)** ในส่วนตอนท้ายของบทนี้

รายละเอียดในแต่ละฟิลด์ของ IP แพคเก็ต



รูปที่ 2.12 แสดงฟิลด์ต่างๆของ IP แพคเก็ต

Version

ขนาด 4 บิต

เป็นฟิลด์ที่ระบุหมายเลขเวอร์ชันของแพคเก็ต ซึ่งทั้งฝ่ายส่งและฝ่ายรับ รวมถึงตัวค้นหาเส้นทางจะต้องรู้จักเวอร์ชันของแพคเก็ตนั้นๆด้วย ที่ใช้กันในปัจจุบันจะเป็นเวอร์ชัน 4

IP Header Length (IHL)

ขนาด 4 บิต

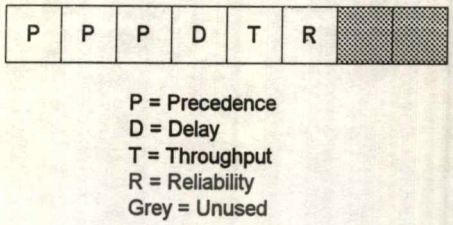
IHL จะระบุความยาวของเฮดเดอร์ของดาตาแกรม ในหน่วยของเวิร์ด (ขนาดเท่ากับ 32 บิต) โดยทั่วไปแล้วเฮดเดอร์จะมีความยาว 5 เวิร์ด (ถ้าไม่มีการใช้ส่วนฟิลด์ Options)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Type of Service

ขนาด 8 บิต

จะเป็นส่วนที่ระบุว่าจะให้โพรโตคอลที่อยู่ในชั้นสูงขึ้นไป จัดการกับแพคเก็ตนั้นๆอย่างไร โดยจะแบ่งออกเป็นบิตต่างๆดังนี้



รูปที่ 2.13 แสดงค่าต่างๆในฟิลด์ Type of Service

Precedence

จะใช้ 3 บิตแรกเป็นตัวกำหนดความสำคัญ (Priority) ของแพคเก็ต โดย `0` จะมีความสำคัญต่ำสุด ส่วน `7` จะมีความสำคัญสูงสุด

Delay

อยู่ในตำแหน่งบิตที่ 4 บิตนี้จะถูกกำหนดเป็น `1` ถ้าต้องการการส่งแพคเก็ตแบบที่ไม่ต้องมีการหน่วงเวลา ถ้าหากเป็น `0` จะเป็นการส่งในรูปแบบปกติ

Throughput

ถ้าหากต้องการให้การส่งแพคเก็ตมีประสิทธิภาพสูง ให้กำหนดบิตนี้เป็น `1` ถ้าหากเป็น `0` จะเป็นการส่งในรูปแบบปกติ

Reliability

บิตนี้จะกำหนดเป็น `1` หากเป็นการส่งข้อมูลที่ต้องการให้มีความเชื่อถือสูง อาทิเช่นในระบบธนาคาร หากกำหนดเป็น `0` จะเป็นการส่งตามรูปแบบปกติ

ส่วน 2 บิตท้ายสุดนั้น ยังไม่มีการนำมาใช้ โดยทั่วไปจะกำหนดให้เป็น `0`

Bits	Description	Values
0-2	Precedence	0-7
3	Delay	0-Normal 1-Low
4	Throughput	0-Normal 1-High
5	Reliablity	0-Normal 1-High
6-7	Reserved for future use	

ตารางที่ 2.4 แสดงค่าต่างๆในฟิลด์ Type of Service

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Length

ขนาด 16 บิต

จะเป็นฟิลด์ที่แสดงถึงขนาดความยาวของแพคเก็ตนั้นๆ มีหน่วยเป็นจำนวนไบต์

Identification

ขนาด 8 บิต

ใช้เป็นฟิลด์ระบุหมายเลขของชิ้นส่วนย่อย เมื่อตอนที่ผู้รับได้ชิ้นส่วนย่อยนั้นแล้ว ก็จะได้ทราบว่าเป็นของแพคเก็ตไหน

Flags

ขนาด 3 บิต

Bit 1	Bit 2	Bit 3
Reserved	0 - May Fragment	0 - Last Fragment
	1 - Do not Fragment	1 - More Fragments

ตารางที่ 2.5 แสดงค่าและความหมายในฟิลด์ Flags

ใช้เป็นฟิลด์ในการระบุตำแหน่งของชิ้นส่วนย่อย เพื่อให้ผู้รับทราบว่าเป็นตำแหน่งที่เท่าไรของแพคเก็ตนั้น

Fragment Offset

ขนาด 13 บิต

เป็นลำดับของชิ้นส่วนย่อยที่เท่าไร ของตัวแพคเก็ตที่จะส่งในกรณีที่แพคเก็ตนั้นมีขนาดใหญ่เกินจำเป็นที่จะต้องทอนเป็นส่วนย่อยๆ เพื่อให้ผู้รับสามารถนำชิ้นส่วนย่อยๆเหล่านี้ กลับมาจัดเรียงลำดับได้อย่างถูกต้อง

Time-To-Live

ขนาด 8 บิต

เป็นฟิลด์ที่ใช้กำหนดอายุของแพคเก็ต ว่าจะมีอายุได้ยืนนานสักเท่าไร เมื่อได้ส่งออกไปในระบบเครือข่ายแล้ว โดยค่าในฟิลด์ดังกล่าวจะลดลงไปเรื่อยๆ ทีละ 1 เมื่อได้ผ่านตัวค้นหาเส้นทาง 1 ครั้ง แต่ในบางครั้งอาจจะลดลงได้มากกว่าหนึ่ง ในกรณีที่ตัวค้นหาเส้นทางนั้นๆ ทำงานช้า ค่าสูงสุดที่สามารถจะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดได้คือ 255 หรือเทียบเท่ากับ 4.25 นาที การส่งข้อมูลที่สำเร็จ แพคเก็ตจะต้องถึงปลายทาง ก่อนที่ค่าในฟิลด์นี้จะเป็น `0`

Protocol

ขนาด 8 บิต

กำหนดชนิดของโพรโตคอลที่ใช้ในชั้นที่สูงขึ้นไป เพื่อให้ทางฝ่ายรับส่งผ่านแพคเก็ตจากชั้นอินเทอร์เนตชั้นไปสู่ชั้นโฮสต์ต่อโฮสต์ได้อย่างถูกต้อง หากเป็น TCP จะมีค่าเป็น `6` ส่วน UDP จะมีค่าเป็น `17

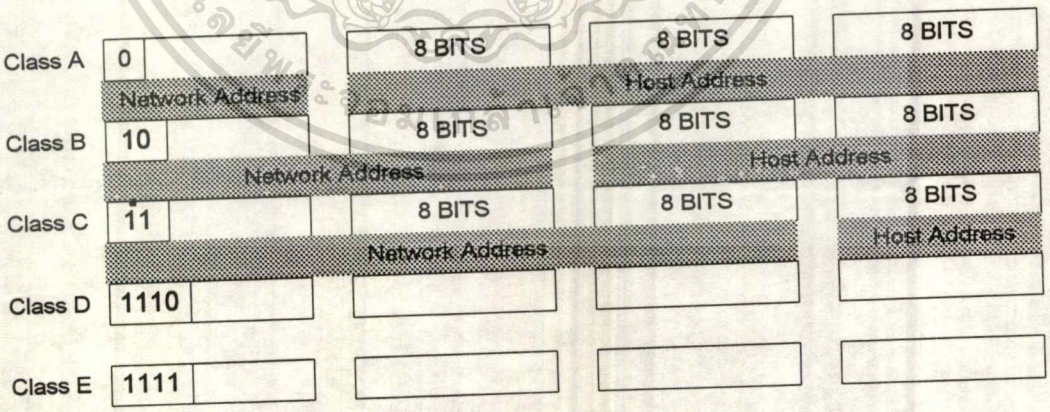
Header Checksum

ขนาด 16 บิต

ใช้เพื่อตรวจสอบความถูกต้องของข้อมูลที่อยู่ในแพคเก็ต ในการส่งบนระบบเครือข่าย ทั้งนี้ค่าดังกล่าวจะต้องคำนวณใหม่ทุกครั้งที่มีการค้นหาเส้นทาง เพราะค่าของฟิลด์ Time-To-Live เปลี่ยนไป

Source and Destination Addresses

ในส่วนของ IP แพคเก็ต จะมีฟิลด์เก็บหมายเลขที่อยู่ต้นทาง และปลายทางไว้ ซึ่งหมายเลขที่อยู่นี้จะเป็นแบบลอจิคัลหรือที่คนส่วนใหญ่มักจะเรียกกันว่าเป็น *IP Address* โดยจะเป็นชุดของตัวเลข 4 ไบต์ แต่ละไบต์จะคั่นด้วย `.` สามารถแบ่งออกได้เป็น 5 ชั้น ดังนี้



รูปที่ 2.14 แสดงการจัดแบ่งชั้นของค่าตำแหน่งที่อยู่ IP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Class A

จะเริ่มจากเลข 0-127 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย `0` ในส่วนไบต์แรกจะใช้ระบุหมายเลขที่อยู่เนตเวอร์ก อีก 3 ไบต์หลังจะใช้ระบุหมายเลขที่อยู่โฮสต์ ใน Class A นี้จะใช้กับเนตเวอร์ก ที่ภายในประกอบด้วยโฮสต์ จำนวนมาก

Class B

จะเริ่มจากเลข 128-191 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย "10" ในส่วนของสองไบต์แรกจะระบุหมายเลขที่อยู่เนตเวอร์กและอีกสองไบต์หลัง จะใช้ระบุหมายเลขที่อยู่โฮสต์ มักใช้กับเครือข่ายขนาดกลาง

Class C

จะเริ่มจากเลข 192-223 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย "110" ในส่วนสามไบต์แรกใช้ระบุ หมายเลขที่อยู่เนตเวอร์กและไบต์สุดท้ายจะใช้ระบุหมายเลขที่อยู่โฮสต์ จึงใช้กับเครือข่ายที่มีขนาดเล็ก จำนวนเครื่องโฮสต์ ไม่มาก

Class D

จะเริ่มจากเลข 224-239 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย "1110" ใช้กับแพคเก็ตที่ต้องการส่งเป็นแบบหลายจุด

Class E

จะเริ่มจากเลข 240-255 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย "1111" ยังไม่นำมาใช้จริง กำลังอยู่ในช่วงศึกษา

Options

ขนาด ไม่เกิน 320 บิต

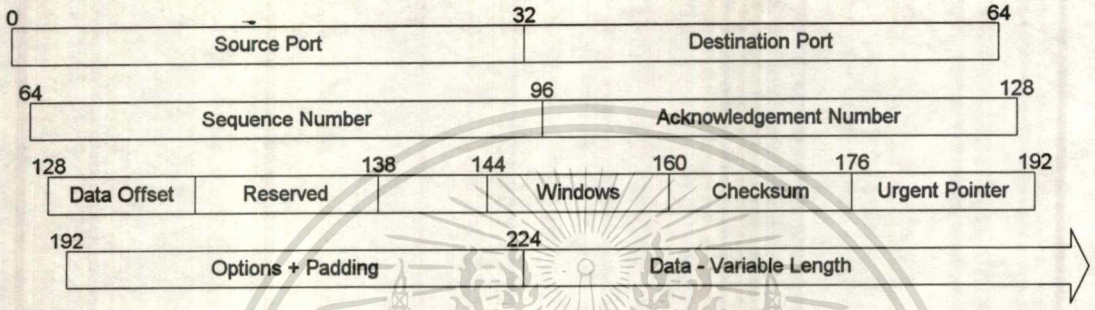
เป็นส่วนออฟชันที่เพิ่มขึ้นมา ใช้กับเฉพาะการใช้งานบางประเภทเช่น ใช้เพื่อระบุการค้นหาเส้นทางของแพคเก็ต ใช้ระบุค่าเวลา (TimeStamp) ใช้ในด้านการรักษาความปลอดภัยของกระทรวงกลาโหมสหรัฐฯ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TCP แพคเก็ต

TCP เป็นโพรโตคอล ที่ทำหน้าที่ในส่วนของชั้นทรานสปอร์ต ทำงานเป็นแบบ Connection-Oriented ใช้รองรับการทำงานโพรโตคอล Telnet, FTP และ SMTP ในระดับชั้นบน

รายละเอียดในแต่ละฟิลด์ของ TCP แพคเก็ต



รูปที่ 2.15 แสดงฟิลด์ต่างๆของ TCP แพคเก็ต

Source Port

ขนาด 16 บิต

เป็นฟิลด์ที่ระบุหมายเลยของโพรโตคอลในชั้นบน ไม่ว่าจะเป็น Telnet, FTP หรือ SMTP เพื่อให้ส่งผ่านข้อมูลในระหว่างระดับชั้นถูกต้อง

Destination Port

ขนาด 16 บิต

ลักษณะเหมือนกับฟิลด์ Source Port แต่เปลี่ยนจากต้นทางมาเป็น ปลายทางเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Port Number	Description
0	Reserved
5	Remote Job Entry
7	Echo
9	Discard
11	Systat
13	Daytime
15	Netstat
17	Quotd (Quote of the day)
20	Ftp_data
21	Ftp (Control)
23	Telnet
25	SMTP
37	Time
53	Name Server
102	ISO-TSAP
103	X.400
104	X.400 sending service
111	Sun RPC
139	NetBIOS session source
160-223	Reserved

ตารางที่ 2.6 แสดงค่าหมายเลขพอร์ตต่างๆในส่วนของฟิลด์ Destination และ Source Port

Sequence Number

ขนาด 32 บิต

ใช้เพื่อทำให้การส่งแพคเก็ต เป็นไปตามลำดับถูกต้อง ตามรูปแบบของวงจรเสมือน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Acknowledgement Number

ขนาด 32 บิต

Acknowledgement Number นำมาใช้เพื่อการตอบรับการได้รับแพคเก็ตแล้ว โดยกำหนดบิต ACK ในฟิลด์ Flag เซต ในฟิลด์นี้จะระบุหมายเลขลำดับไบต์ ของข้อมูลที่คาดว่าจะได้รับในครั้งต่อไป ตัวอย่างเช่น สมมติว่ามีการติดต่อกันระหว่างคู่โฮสต์คู่หนึ่ง ซึ่งต้องมีการส่งแพคเก็ตติดต่อกันจำนวนมาก และแต่ละแพคเก็ตมีขนาดเท่ากับ 20 ไบต์ หมายเลขลำดับของการส่งจะเป็น 40, 60 ,80 เพราะฉะนั้นเมื่อสิ้นสุดการส่งแพคเก็ตชุดที่ 3 แล้ว แพคเก็ตตอบรับ(Acknowledge Packet) ซึ่งผู้รับจะส่งตอบรับกลับไปหาผู้ส่ง จะมีหมายเลข Acknowledge เท่ากับ 100 ซึ่งหมายความว่า ข้อมูลก่อนหน้านี้ (40 ถึง 99) ได้รับเรียบร้อยแล้ว (ที่เป็น 100 เพราะว่า 100 เป็นหมายเลขลำดับต่อไปถัดจาก 80)

Data Offset

ขนาด 4 บิต

เพื่อบอกความยาวของเฮดเดอร์ TCP ในหน่วยเวิร์ด ซึ่งจะเปลี่ยนแปลงได้ตามขนาดของฟิลด์ Option

Reserved

ขนาด 6 บิต

สำรองใช้ในอนาคต

Flags

ขนาด 6 บิต (1 บิต ต่อ 1 Flag)

ในฟิลด์ Flags นี้จะเก็บข้อมูลควบคุมการติดต่อของ TCP ดังนี้

- URG (Urgent) จะเซตเพื่อบอกว่าฟิลด์ Urgent Pointer ทำงานอยู่ด้วย
- ACK (Acknowledge) จะเซตเพื่อบอกว่าเป็นแพคเก็ตตอบรับ โดยมีหมายเลขลำดับการตอบรับอยู่ในฟิลด์ Acknowledge Number
- PUSH ถ้าเป็นฝ่ายส่งจะบอกให้ส่งผ่านข้อมูล ไปยังโพรโตคอลชั้นล่างทันที แต่ถ้าเป็นผู้รับจะทำงานกลับกันคือ ส่งผ่านไปยังระดับชั้นบนทันที
- SYN (Synchronize) เป็นแพคเก็ตเพื่อใช้สร้างวงจรเสมือนระหว่างผู้รับและผู้ส่ง โดยผู้ส่งจะต้องส่งแพคเก็ตที่มีบิต Syn เซต หมายเลขลำดับเป็น A ฝ่ายผู้รับก็จะส่งแพคเก็ตที่มีบิต Syn เซต และบิต ACK เซตเพื่อตอบรับกลับ โดยมีหมายเลขการตอบรับกลับเป็น (A)+1 ฝ่ายผู้ส่งครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แรกจะตอบรับกลับโดยส่งแพคเก็ตที่มีเฉพาะบิต ACK เซตและหมายเลขลำดับการตอบรับเป็น (B)+1

- FIN (Finish) เพื่อบอกว่า ส่งข้อมูลครบหมดแล้ว การติดต่อที่ได้สร้างไว้ยกเลิกได้

Window

ขนาด 16 บิต

เพื่อบอกว่าขนาดจำนวนไบต์ที่ต้องการจะได้รับกลับมา

Checksum

ขนาด 16 บิต

เพื่อตรวจสอบความถูกต้องของข้อมูล ถ้าหากว่าค่าที่อยู่ในฟิลด์นี้ กับที่ได้มาจากการคำนวณของผู้รับข้อมูลมีขนาดไม่เท่ากัน ข้อมูลชุดดังกล่าวจะถูกยกเลิกไป

Urgent Pointer

ขนาด 16 บิต

จะเป็นค่าออฟเซตที่ชี้ไปยังตำแหน่ง *ข้อมูลรีบด่วน* ของแพคเก็ต

Option

ขนาด ไม่แน่นอน

เป็นออฟชันที่เพิ่มขึ้นมา เพื่อเพิ่มความสามารถในการทำงาน อาทิเช่น ขยายขนาดเซกเมนต์ของ TCP

Data

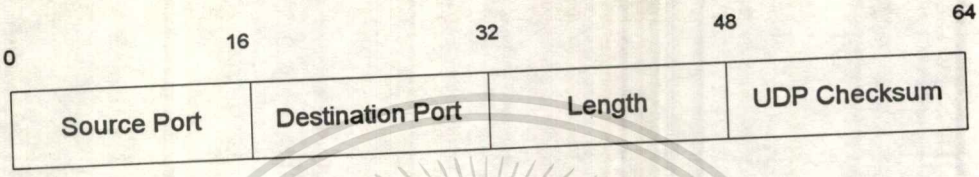
เก็บข้อมูลส่วนที่จะส่งต่อไปยังโพรโตคอลในชั้นบนต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

UDP แพคเก็ต

นำมาใช้งานแทนที่ TCP มีการทำงานแบบ **Connectionless-Oriented** ไม่ต้องมีการตอบรับข้อมูล ทำให้ลดภาระ (Overhead) ลงได้มาก หน้าที่หลักของ UDP เพียงเพื่อรับและส่งผ่านข้อมูลไปยังโพรโตคอลในชั้นบนขึ้นไป

รายละเอียดในแต่ละฟิลด์ของ UDP แพคเก็ต



รูปที่ 2.16 แสดงฟิลด์ต่างๆของ UDP แพคเก็ต

Source Port

ขนาด 16 บิต

เป็นฟิลด์ที่ระบุหมายเลยของโพรโตคอลในชั้นบน ไม่ว่าจะเป็น Telnet, FTP หรือ SMTP เพื่อให้ส่งผ่านข้อมูลในระหว่างระดับชั้นถูกต้อง

Destination Port

ขนาด 16 บิต

ลักษณะเหมือนกับ Source Port แต่เปลี่ยนจากต้นทางมาเป็น ปลายทางเท่านั้น

Length

ขนาด 16 บิต

ใช้บอกความยาวของ UDP แพคเก็ต มีหน่วยเป็นจำนวนไบต์

UDP Checksum

เป็นเพียงออฟชัน เพื่อตรวจสอบความถูกต้อง เมื่อผู้รับได้รับข้อมูล

หมายเหตุ *TCP ทำงานแบบควบคุมลำดับการรับส่งข้อมูล และต้องสร้างการติดต่อกันก่อน แต่ UDP ไม่ต้อง ทำให้ทำงานได้เร็วกว่า เนื่องจากมีภาระน้อย*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ICMP แพคเก็ต

ICMP จะช่วยการทำงานของ IP แพคเก็ต โดยจะให้ข้อมูลที่เป็นประโยชน์เช่น เมื่อเกิดปัญหาขึ้น หรือข้อมูลเพื่อการควบคุมการทำงานกับ IP แพคเก็ต โดยจะส่งมาเป็นข้อความ (Message) ในเรื่องต่างๆ ดังนี้

Echo Request/Reply

PING ซึ่งเป็นยูทิลิตี้ที่ใช้กันอย่างแพร่หลาย ใช้โพรโตคอล ICMP เพื่อตรวจสอบว่าตำแหน่งปลายทางของแพคเก็ต สามารถติดต่อได้หรือไม่

Destination Unreachable

ICMP จะแสดงข้อความดังกล่าว เมื่อตัวค้นหาเส้นทางไม่สามารถหา ตำแหน่งเครื่องที่อยู่ปลายทางได้

Time Exceeded

จะส่งข้อความดังกล่าวกลับมายังผู้ส่งแพคเก็ต เมื่อกรณีที่ฟิลด์ Time-To-Live มีค่าเป็นศูนย์

Redirect

ถ้าหาก ICMP พบเส้นทางที่ดีกว่าเส้นทางเดิมของตัวค้นหาเส้นทาง จะส่งข้อความดังกล่าวกลับมายังตัวค้นหาเส้นทาง

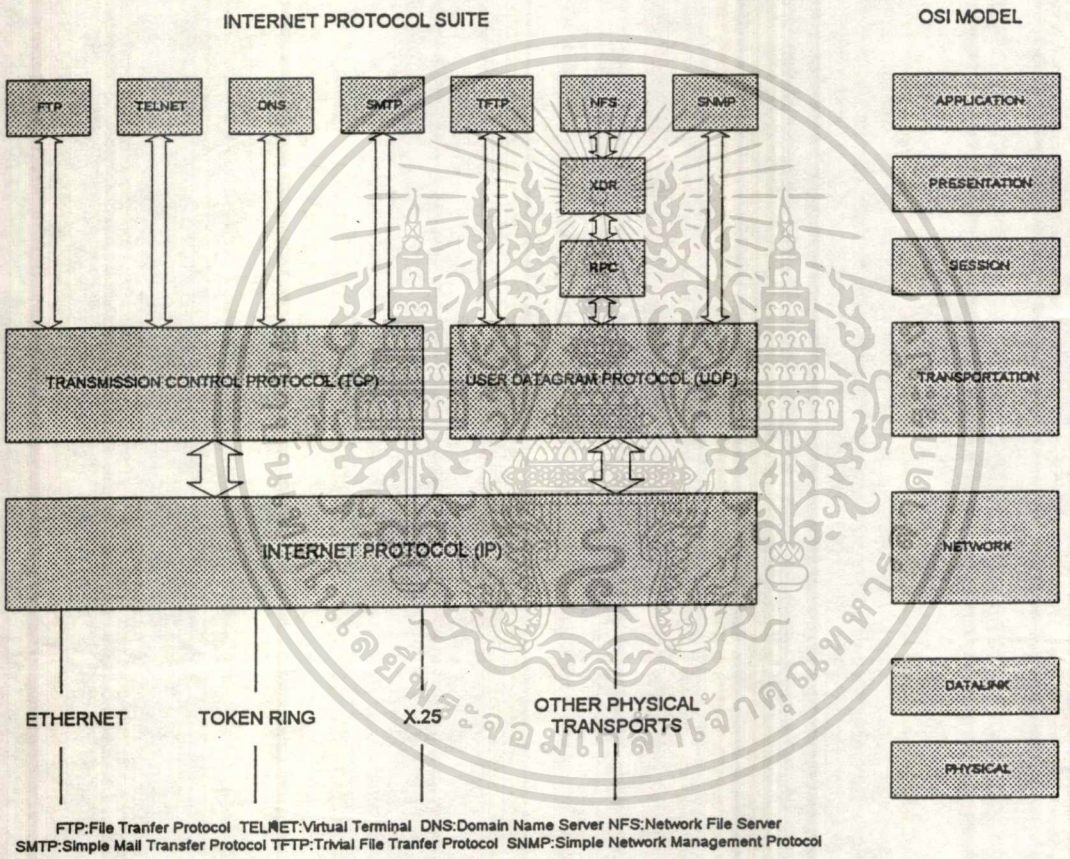
Source Quench

ถ้าหากว่าฝ่ายส่ง ส่งข้อมูลเร็วเกินกว่าที่ฝ่ายรับจะทำงานได้ทัน ICMP จะส่งข้อมูลดังกล่าวเตือนกลับมายังฝ่ายส่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรโตคอลซึ่งทำงานในระดับชั้นสูงขึ้นไป

เป็นโพรโตคอลซึ่งมีหน้าที่การทำงานอยู่ตั้งแต่ในระดับชั้นเซสชัน จนถึงชั้นแอพพลิเคชันตามมาตรฐานของ OSI การทำงานของแต่ละโพรโตคอลในระดับชั้นนี้ จะต้องอาศัยการทำงานของโพรโตคอลที่อยู่ในระดับล่าง ไม่ว่าจะเป็น IP , TCP หรือ UDP



รูปที่ 2.17 แสดงการทำงานของโพรโตคอลที่อยู่ในระดับชั้นบน

Remote Terminal Emulation - TELNET

เป็นโพรโตคอลที่จะจำลองเครื่องคอมพิวเตอร์ ทำงานเป็นเทอร์มินัลของเครื่องโฮสต์ ได้ ซึ่งส่วนใหญ่แล้วเครื่องคอมพิวเตอร์จะมีความสามารถสูงกว่า ที่ตัวเทอร์มินัลเองจำเป็นต้องใช้อยู่แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

File Transport Protocol(FTP) และ Trivial File Transfer Protocol (TFTP)

FTP จะทำให้สามารถโอนถ่ายแฟ้มข้อมูลต่างๆระหว่างเครื่องโฮสต์ได้ การทำงานของ FTP จะต้องทำการ login ก่อนและจะมีการตรวจสอบสิทธิที่ของผู้ใช้นั้นๆ บนแฟ้มข้อมูลก่อน ส่วนการทำงานของ TFTP ไม่ต้องทำการ login ทำให้ลดภาระในส่วนนี้ไปได้ แต่ยังคงต้องมีการตรวจสอบสิทธิต่างๆก่อนเช่นเดียวกัน

Simple Mail Transfer Protocol (SMTP)

เป็นโพรโตคอลที่เครื่องโฮสต์ ใช้จัดการกับจดหมายอิเลกทรอนิกส์

Routing Information Protocol (RIP)

RIP เป็นโพรโตคอลที่ใช้ดูแลและจัดการข้อมูลที่เกี่ยวกับการค้นหาเส้นทาง โดยจะมีการส่ง RIP ออกไปตามช่วงเวลาที่กำหนด เพื่อที่จะทำการเปลี่ยนแปลงข้อมูลของตารางการค้นหาเส้นทาง ซึ่งเก็บอยู่ในเครื่องโฮสต์และตัวค้นหาเส้นทาง เพื่อทำให้การค้นหาเส้นทางมีประสิทธิภาพสูงสุด

Network File System (NFS)

เป็นโพรโตคอลที่พัฒนาขึ้นโดย Sun Microsystems บนแพลตฟอร์ม(Platform) ONC เพื่อเมาท์ hard drive ของเครื่องที่อยู่บนเครือข่าย ให้ทำงานได้เสมือนเป็นโลคัลไดเรกทอรีหนึ่ง เพื่อที่จะใช้ทรัพยากรเครื่องที่ทำการเมาท์ได้ ถึงแม้ว่าจะเป็นระบบปิด(proprietary) โดยจะต้องใช้ 3 โพรโตคอลประกอบกัน คือ NFS, XDR (External Data Representation) และ RPC(Remote Procedure Call)

eXternal Data Representation (XDR)

XDR จะเป็นกลุ่มของไลบรารีรูทีน ซึ่งเรียกใช้ได้โดยภาษาซี มีความสามารถในการใช้ข้อมูลของระบบอื่นโดยไม่ต้องกังวลในเรื่องของแพลตฟอร์มที่แตกต่างกัน

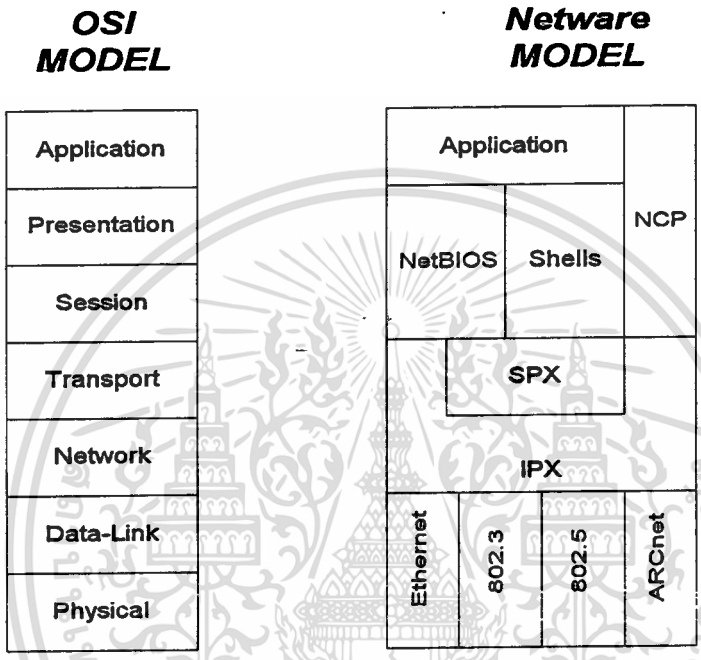
Remote Procedure Call (RPC)

ให้ระบบสามารถใช้ข้อมูล ได้ทั้งจากระบบของตนเอง, จากตัวไคลเอนต์ (Client) หรือจากระบบอื่นๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 ชุดเนตแวร์โพรโตคอล

เนตแวร์ใช้โพรโตคอล XNS (Xerox Netware System) ซึ่งพัฒนาโดยบริษัท XEROX สถาปัตยกรรมของโพรโตคอล XNS สามารถเทียบเคียงได้กับ OSI ดังนี้



รูปที่ 2.18 แสดงการเปรียบเทียบมาตรฐาน OSI กับชุดเนตแวร์โพรโตคอล

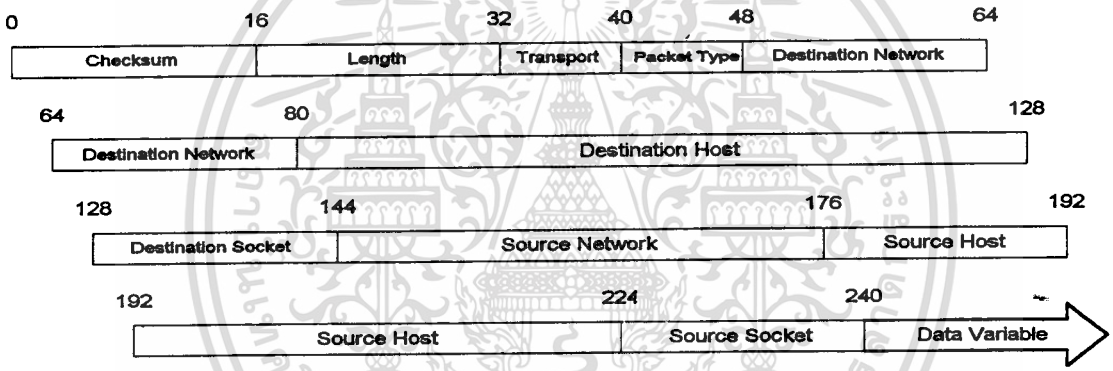
ระดับชั้นฟิสิคัลและดาตาลิงค์ จะใช้มาตรฐานของอีเทอร์เนต, IEEE 802.3, 802.5 และอาร์กเนต ระดับชั้นเนตเวอร์กเทียบได้กับ IPX (Internet Packet Exchange) ชั้นทรานสปอร์ตเทียบได้กับ Sequenced Packet Exchange (SPX) ชั้นเซสชันจะเป็น NetBIOS (Network Basic Input Output System) ชั้นพรีเซนเตชันและชั้นแอพพลิเคชันเปรียบได้กับ Netware Core Protocol (NCP) และ Netware Value-Added Services

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IPX แพคเก็ต

Internet Packet Exchange (IPX) เป็นโพรโตคอลที่ทำงานแบบ Connectionless รับหน้าที่การค้นหาเส้นทางให้กับแพคเก็ต จากที่อยู่ต้นทางไปยังที่อยู่ปลายทาง เดิมที IPX ทำงานได้เหมือนกับ IDP (Internetwork Datagram Protocol) ของ XNS แต่เนื่องจากต้องทำงานการค้นหาเส้นทางด้วย ทางบริษัทโนเวลล์ (Novell) จึงพัฒนา IPX เวอร์ชันใหม่ขึ้นมา ให้มีความสามารถในการค้นหาเส้นทางด้วย โดยจะเป็น RIP ของ IPX โดยทำงานคล้ายคลึงกับ RIP ของชุดอินเตอร์เนตโพรโตคอล เนตแวร์เซอร์ฟเวอร์และตัวค้นหาเส้นทาง จะส่งข้อมูลที่อยู่กระจายออกไปยังโหนดอื่นๆ ทุกๆช่วงเวลาที่กำหนดเพื่อเป็นข้อมูลในการจัดทำตารางการค้นหาเส้นทาง

รายละเอียดในแต่ละฟิลด์ของ IPX แพคเก็ต



รูปที่ 2.19 แสดงฟิลด์ต่างๆของ IPX แพคเก็ต

Checksum

ขนาด 16 บิต

ใช้ตรวจสอบความถูกต้องของตัวเฮดเดอร์ XNS แต่ฟิลด์นี้จะไม่ใช้ในกรณีที่เป็นระบบเครือข่ายเดี่ยว เพราะถือว่าในระบบเครือข่ายเดี่ยว มีความน่าเชื่อถือสูงอยู่แล้ว

Length

ขนาด 16 บิต

จะแสดงความยาวทั้งหมดของ IPX แพคเก็ต โดยความยาวนั้นจะมีค่าตั้งแต่ 30 ไบต์ไปจนถึง 576 ไบต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Transport Control

ขนาด 8 บิต

ใช้กำหนดอายุของแพคเก็ตที่จะส่งออกไปบนระบบเครือข่าย ผู้ส่งจะกำหนดค่าในฟิลด์นี้เป็นศูนย์ เมื่อแพคเก็ตนี้ผ่านตัวค้นหาเส้นทางแต่ละตัว ด้วยค้นหาเส้นทางจะเพิ่มค่าในฟิลด์นี้ทีละ 1 และเมื่อไรก็ตามที่ค่าในฟิลด์นี้เป็น 16 แพคเก็ตชิ้นนั้นจะถูกยกเลิกทิ้ง

Packet Type

ขนาด 8 บิต

จะเป็นตัวระบุโพรโตคอลในชั้นบน ที่ IPX จะส่งผ่านข้อมูลขึ้นไปให้ ค่าที่ใช้ในฟิลด์นี้มีดังนี้

0 Unknown Packet

1 Routing Information Protocol (RIP)

2 Echo Packet

3 Error Packet

4 Packet Exchange Packet (PEP)

5 Sequenced Packet Protocol (SPP)

17 Netware Core Protocol (NCP)

Destination Network

ขนาด 32 บิต

จะเป็นที่อยู่ของระบบเครือข่ายปลายทาง หากมีค่าเป็นศูนย์ หมายความว่าเครื่องต้นทางและเครื่องปลายทางอยู่บนระบบเครือข่ายเดียวกัน

Destination Host

ขนาด 48 บิต

จะเป็นที่อยู่ทางฟิสิคัลของเครื่องปลายทาง ถ้าหากใช้มาตรฐาน IEEE 802.3 หรือ IEEE 802.5 จะใช้ครบหมดทุกบิต แต่ถ้าใช้อาร์กเนต ซึ่งที่อยู่ในมาตรฐานนี้จะใช้ไม่ครบทุกไบต์ หากไบต์ไหนไม่มีการใช้จะเติมเลขศูนย์ที่หน้าไบต์เหล่านั้น แต่หากเป็นการส่งข้อมูลแบบกระจายทุกจุด ค่าในฟิลด์นี้จะเป็น หนึ่งทุกบิต

Destination Socket

ขนาด 16 บิต

จะระบุโพรเซสของเครื่องปลายทางที่ใช้งานกับแพคเก็ตนี้ ซึ่งมีค่าต่างๆดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

0451H File server packet

0452H Service advertising packet

0453H Routing Information packet

0455H NetBIOS packet

0456H Diagnostic packet

IPX จะใช้ฟิลด์ Destination Socket และ Source Socket ระบุโพรเซสที่จะใช้งานในระดับชั้นบนขึ้นไป

Source Network

ขนาด 32 บิต

ลักษณะเช่นเดียวกันกับตัว Destination Network แต่จะเป็นที่อยู่ระบบเครือข่ายของต้นทางแทน ถ้าหากมีค่าเป็นศูนย์ จะเป็นระบบเครือข่ายที่ไม่รู้จัก

Source Host

ขนาด 48 บิต

จะเป็นที่อยู่ทางฟิสิคัลของเครื่องที่เป็นฝ่ายส่งข้อมูล มีรูปแบบคล้ายกับฟิลด์ Destination Host และจะใส่ศูนย์ ลงไปแทนไบต์ที่ไม่มีการใช้งาน ถ้าหากเป็นการส่งข้อมูลแบบกระจายทุกจุด ทุกบิตในฟิลด์นี้จะเป็นหนึ่ง

Source Socket

ขนาด 16 บิต

มีรูปแบบและการทำงานเช่นเดียวกับ Destination Socket

Data

ขนาด ไม่คงที่

จะเป็นฟิลด์ที่เก็บข้อมูล เพื่อส่งผ่านขึ้นไปยังระดับชั้นบนในกรณีที่เป็นผู้รับข้อมูล หรือใช้เก็บข้อมูลที่ส่งผ่านลงมาจากระดับชั้นบนในกรณีที่เป็นฝ่ายส่ง

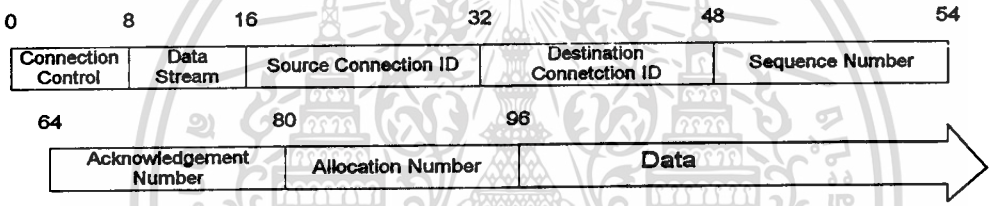
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SPX แพคเก็ต

Sequenced Packet Exchange (SPX) ทำงานแบบ Connection-Oriented มีการตรวจสอบการส่งข้อมูลระหว่างเครื่องลูกข่ายกับเครื่องเซอร์ฟเวอร์ ตัวอย่างการนำโพรโตคอลนี้คือ RConsole, การพิมพ์ทางไกล(Remote printing) และ SNA เกตเวย์ เพราะการใช้งานเหล่านี้จำเป็นต้องใช้ความน่าเชื่อถือสูงระหว่างที่มีการรับส่งข้อมูล

การทำงานของ SPX มีรูปแบบเป็นวงจรเสมือน ในระหว่างแต่ละคู่ของเครื่องที่มีการติดต่อ จะมีหมายเลขเฉพาะ เรียกว่า Connection ID ทำให้ในหนึ่งช่องการติดต่อ (Socket) สามารถมีได้หลายคู่ Connection ID

รายละเอียดในแต่ละฟิลด์ของ SPX แพคเก็ต



รูปที่ 2.20 แสดงฟิลด์ต่างๆของ SPX แพคเก็ต

Connection Control

ขนาด 8 บิต

ประกอบด้วยค่าแฟลก แบบ 1 บิตจำนวน 4 แฟลก ซึ่งใช้ควบคุมการรับส่งของข้อมูล

01H-08H ยังไม่ได้มีการนำมาใช้

10H สิ้นสุดข้อความ

20H เตรียมพร้อม

40H ต้องการการตอบรับกลับ

80H เป็นแพคเก็ตของระบบ

Datastream Type

ขนาด 8 บิต

เพื่อแสดงประเภทของข้อมูลในด้านที่อยู่ของแพคเก็ต มีลักษณะคล้ายกับฟิลด์ Type ของอีเทอร์เนต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

00H-FDH กำหนดโดยตัวไคลเอนต์

FEH บอกสิ้นสุดการติดต่อ

FFH ตอบรับการสิ้นสุดการติดต่อ

Source Connection ID

ขนาด 16 บิต

ค่าในฟิลด์นี้ จะถูกกำหนดโดยฝ่ายส่งแพคเก็ต

Destination Connectin ID

ขนาด 16 บิต

ใช้เพื่อแยกความแตกต่างของข้อมูล ในกรณีที่มีคู่การติดต่อหลายตัว ติดต่อผ่านทางช่องการติดต่อเดียว

Sequence Number

ขนาด 16 บิต

จะเป็นค่าของหมายเลขแพคเก็ตที่คาดว่าจะได้รับในครั้งต่อไปจากคู่ติดต่ออีกฝ่ายหนึ่ง เพื่อเพิ่มความน่าเชื่อถือให้กับการรับส่งข้อมูล

Allocation Number

ขนาด 16 บิต

แสดงหมายเลขของแพคเก็ต ซึ่งจะสัมพันธ์กับขนาดของบัฟเฟอร์(Buffer) ที่ทางฝ่ายรับเตรียมไว้ในการรับแพคเก็ต ฝ่ายส่งจะสามารถส่งแพคเก็ตมาให้ฝ่ายรับเรื่อยๆ จนกว่าค่า Sequence Number กับ Allocation Number เท่ากัน

Data

เก็บข้อมูลที่จะใช้ส่งผ่านขึ้นไปให้กับระดับการทำงานชั้นบนต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 เปรียบเทียบการทำงานของ TCP/IP และ SPX/IPX

สิ่งที่เหมือนกันของ IP และ IPX

- ทำงานอยู่ที่ระดับชั้นเนตเวอร์กของ OSI
- การทำงานเป็นแบบดาตาแกรม
- มีโพรโตคอลช่วยในการค้นหาเส้นทางคือ RIP
- การรับส่งข้อมูลจะเป็นแบบ Connectionless
- มีการกำหนดอายุของแพคเก็ตที่ส่งออกไปในระบบเครือข่าย

สิ่งที่เหมือนกันของ SPX และ TCP

- ทำงานอยู่ที่ระดับชั้นทรานสปอร์ตของ OSI
- มีหน้าที่เพื่อเพิ่มความน่าเชื่อถือในการรับส่งข้อมูล โดยใช้การส่งข้อมูลตอบรับเมื่อได้รับข้อมูล เป็นการทำงานแบบวงจรเสมือน
- รูปแบบการทำงานเป็นแบบ ควบคุมการรับส่งข้อมูล

สิ่งที่มีเฉพาะใน TCP/IP แต่ไม่มีใน SPX/IPX

- ข้อมูลของ TCP/IP ได้มีการเผยแพร่ออกสู่สาธารณะ
- การรับส่งข้อมูลเป็นแบบ Sliding วินโดวส์
- IP จะใช้ฟิลด์ Time-To-Live เป็นตัวกำหนดอายุของแพคเก็ตในเครือข่าย
- ที่อยู่แบบ IP จะรวมทั้งที่อยู่ของระบบเครือข่ายและเครื่อง ไว้ในข้อมูลตัวเดียว

สิ่งที่มีเฉพาะใน SPX/IP แต่ไม่มีใน TCP/IP

- ข้อมูลของ SPX/IPX ไม่ได้นำออกเผยแพร่สู่สาธารณะ
- การรับส่งข้อมูลจะเป็นแบบ นับจำนวนบัฟเฟอร์ที่ว่าง พร้อมที่จะรับข้อมูล
- IPX จะใช้ฟิลด์ Transport Control กำหนดอายุของแพคเก็ต
- ที่อยู่แบบ IPX จะแยกกันเป็น 2 ตัวคือที่อยู่ของระบบเครือข่าย และที่อยู่ของเครื่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

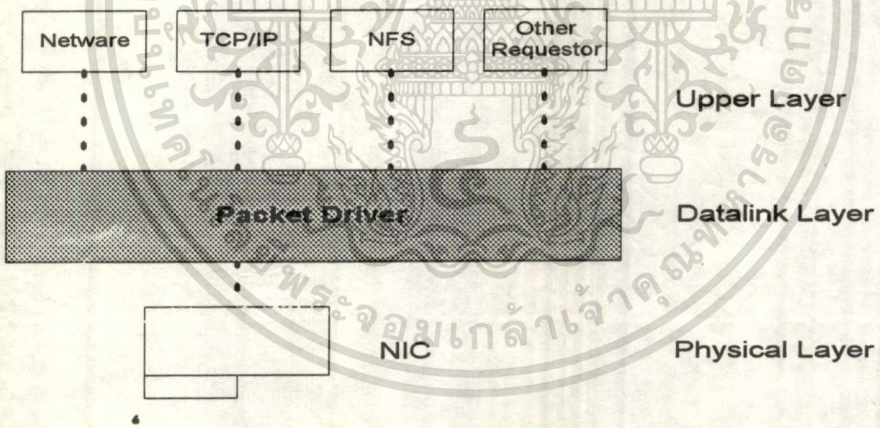
บทที่ 3 การคำนวณและการสร้าง

3.1 การทำงานและประโยชน์ของแพคเก็ตไดรเวอร์

ในองค์กรขนาดใหญ่ที่ใช้ระบบเครือข่ายคอมพิวเตอร์ โดยเฉพาะที่มีอีเทอร์เนตเป็นเครือข่ายหลัก เพราะว่าเป็นมาตรฐานที่มีความยืดหยุ่นสูง มักจะพบกับปัญหาว่า เมื่อทำงานกับแอพพลิเคชันตัวไหนแล้ว จำเป็นต้องเรียกใช้ไดรเวอร์เนตเวอร์กการ์ดเฉพาะสำหรับแต่ละแอพพลิเคชัน ทำให้ไม่สามารถทำงานกับแอพพลิชันหลายๆ ตัวพร้อมกันได้ เมื่อต้องการใช้แอพพิลเคชันตัวใหม่ ต้องเอาไดรเวอร์ตัวเก่าออกก่อน แล้วจึงเรียกใช้ไดรเวอร์ตัวใหม่ ไม่สะดวกในการใช้งานเป็นอย่างมาก หากต้องการทำงานโดยใช้ชุดอินเตอร์เนตโพรโตคอล ก็ไม่สามารถใช้ชุดเนตแวร์โพรโตคอลได้

จึงมีความคิดในการทำไดรเวอร์ของเนตเวอร์กการ์ด ที่มีใช้รูปแบบมาตรฐานในการติดต่อสามารถทำให้ทุกแอพพลิเคชัน ติดต่อกับเนตเวอร์กการ์ดได้ ในเวลาเดียวกัน ไม่ต้องเปลี่ยนไดรเวอร์กลับไปกลับมา โดยใช้ชื่อไดรเวอร์นี้ว่า **"แพคเก็ตไดรเวอร์"** (Packet Driver)

ลักษณะการทำงานของแพคเก็ตไดรเวอร์



รูปที่ 3.1 แสดงการทำงานของแพคเก็ตไดรเวอร์

แพคเก็ตไดรเวอร์จะเข้าไปควบคุมตัวเนตเวอร์กการ์ด โดยจะทำงานฝังอยู่ในหน่วยความจำ ทำหน้าที่จัดเตรียมแพคเก็ตให้กับโพรโตคอลในชั้นสูงขึ้นไป โดยเมื่อได้รับเฟรมข้อมูลเข้ามา แพคเก็ตไดรเวอร์จะตรวจสอบดูฟิลด์ Type ของเฟรมนั้นๆ (ศึกษาได้จากภาคผนวก) ว่าตรงกับชนิดของโพรโตคอลที่มีการติดต่ออยู่หรือไม่ ถ้าตรงก็จะส่งผ่านข้อมูลในเฟรมนั้นขึ้นไปให้ หากไม่ตรงกับโพรโตคอลใดเลย เฟรมนั้นจะถูกยกเลิกทิ้ง เพื่อเตรียมพร้อมสำหรับรับเฟรมต่อไป

ชนิดของเฟรมที่แพคเก็ตไดรเวอร์ ทำงานได้คือ **อีเทอร์เนต** และ IEEE 802.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประโยชน์ที่ได้จากการใช้แพคเก็ตไดรเวอร์

1. สามารถทำงานหลายแอพพลิเคชันได้ในเวลาเดียวกัน และบนเนตเวอร์กการ์ดตัวเดียวกัน โดยแอพพลิเคชันแต่ละตัวนั้นทำงานโดยติดต่อผ่าน แพคเก็ตไดรเวอร์
2. เนตเวอร์การ์ดเพียงอันเดียวก็เพียงพอแล้ว ไม่จำเป็นต้องเตรียมเนตเวอร์กการ์ด เพื่อการทำงานสำหรับแอพพลิเคชันแต่ละตัว
3. เพิ่มความเร็วในการกรองแพคเก็ต ที่ไม่ต้องการออก โดยเฉพาะแพคเก็ตประเภทส่งแบบกระจาย
4. จากเหตุผลในข้อ 3 ทำให้ลดภาระการทำงานของ CPU ลงไม่ต้องไปคอยจัดการกับแพคเก็ตเหล่านั้นอีก
5. เมื่อต้องการทำงานแอพพลิเคชันใหม่ ไม่จำเป็นต้องเปลี่ยนค่ากำหนดเดิมที่ใช้ หรือบูตเครื่องใหม่อีก
6. ขณะที่ทำงานกับตัวเนตแวร์เซอร์ฟเวอร์ ยังทำงานกับชุดอินเตอร์เนตโพรโตคอล หรือชุดโพรโตคอลอื่นๆได้อีกด้วย โดยสามารถทำงานกับตัวเนตแวร์เซอร์ฟเวอร์ได้ตามปกติ
7. แพคเก็ตไดรเวอร์ช่วยลดภาระให้กับตัวไคลเอนต์ (ส่วนโพรโตคอลที่อยู่ในระดับชั้นสูงขึ้นไป) โดยจะเรียกใช้ไดรเวอร์เมื่อมีแพคเก็ตที่เกี่ยวข้องกับตัวเองเท่านั้น ซึ่งจะช่วยให้แอพพลิเคชันที่ทำงานช้า ไม่ต้องรับภาระในการดูแลจัดการแพคเก็ตอีก
8. เมื่อเรียกใช้จะฝังตัวอยู่ในหน่วยความจำ แต่มีขนาดเล็ก ไม่เปลืองเนื้อที่หน่วยความจำมากนัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 การเรียกใช้โปรแกรมแพคเก็ตไดรเวอร์

1. เลือกตัวโปรแกรมแพคเก็ตไดรเวอร์ ให้ตรงกับเนตเวอร์กการ์ดที่ใช้อยู่ อาทิเช่น 3C5XX จะเป็นแพคเก็ตไดรเวอร์ที่ใช้กับเนตเวอร์การ์ดของบริษัท 3COM แต่ถ้าเป็น NE2000 ก็จะใช้กับ เนตเวอร์กการ์ดที่เป็นแบบ NE2000 Compatible
2. เรียกใช้โปรแกรมแพคเก็ตไดรเวอร์ ที่ได้มา โดยมีรูปแบบดังนี้

   การใช้: *NE2000 [options] packet_int_no [hardware_irq[io_addr]]*

   (รูปแบบนี้อาจจะเปลี่ยนไปได้ ตามแต่ละชนิดของเนตเวอร์กการ์ด)

   ซึ่งทำให้โปรแกรมแพคเก็ตไดรเวอร์เข้าไปฝังตัวอยู่ในหน่วยความจำ

   Options

   - -d ใช้กับเครื่องลูกข่ายที่จำเป็นต้องบูตเครื่องโดยใช้ PROMBoot บนเนตเวอร์กการ์ด
   - -n ตัวเนตแวร์เชลล์ (Netware Shell) สามารถใช้ได้งานได้กับทั้งเฟรมแบบอีเทอร์เนต II และ IEEE 802.3 บนมาตรฐานอีเทอร์เนต แต่ก็มีเชลล์ (Shell) บางตัวที่จะต้องใช้กับเฟรมที่เป็นแบบอีเทอร์เนต II เท่านั้น หากใช้ออฟชัน -n แพคเก็ตไดรเวอร์ จะแปลงข้อมูลทั้งหมดให้อยู่ในรูปแบบ เฟรมอีเทอร์เนต II เพื่อให้ทำงานกับเชลล์เหล่านั้นได้ (และสามารถแปลงกลับให้มีรูปแบบเหมือนเดิมจากเฟรมอีเทอร์เนต II ไปเป็น IEEE 802.3 ได้ด้วย)
   - -p สามารถใช้ออฟชันนี้เพื่อเพิ่มความปลอดภัยในการใช้ข้อมูล เพื่อเป็นการยกเลิกโหมดการทำงานแบบ *promiscuous* แต่ถ้าหากต้องการความปลอดภัยจริงๆแล้ว ไม่ควรใช้เพียงแค่ออฟชันนี้
   - -w ใช้เมื่อต้องการใช้งานบนวินโดวส์ แต่ขณะนี้ได้มีการพัฒนา **winpkt** ขึ้นมาเพื่อทำงานบนวินโดวส์โดยเฉพาะ ซึ่งจะทำงานได้มีประสิทธิภาพกว่า
   - -i ใช้เมื่อโปรแกรมที่ทำงานติดต่อกับแพคเก็ตไดรเวอร์ ต้องตรวจสอบโหมดการทำงานโดยจำกัดโหมด ว่าต้องเป็นโหมดสำหรับเฟรม IEEE 802.3 เท่านั้น ถึงแม้ว่าตัวแพคเก็ตไดรเวอร์จะทำงานได้กับเฟรมทั้งแบบอีเทอร์เนต และ IEEE 802.3 แต่จะรายงานโหมดการทำงานของตนเอง เพียงโหมดใดโหมดหนึ่งเท่านั้น การใช้ออฟชันนี้จะทำให้ตัวแพคเก็ตไดรเวอร์ รายงานกับโปรแกรมที่ติดต่ออยู่ว่าเป็นโหมดสำหรับเฟรม 802.3 หากไม่ใช้ออฟชันนี้ แพคเก็ตไดรเวอร์จะรายงานว่าทำงานเป็นโหมดอีเทอร์เนต II อยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

packet_int_no

จะเป็นตำแหน่งอินเทอร์รัพต์ของซอฟท์แวร์ที่แพคเก็ตไดรเวอร์ จะใช้เพื่อติดต่อกับโปรแกรมของเรา หมายเลขอินเทอร์รัพต์นี้ เลือกใช้ได้ตั้งแต่ 0X60 ถึง 0X7E แต่จะต้องไม่ไปทับซ้อนกับการทำงานของโปรแกรมอื่นๆ

ตำแหน่งที่ต่ำกว่า 0X60 ใช้โดยโปรแกรม DBase

ตำแหน่งที่ 0X67 ใช้ในการติดต่อกับ EMS

ตำแหน่งที่ 0X70-0X77 ใช้เป็น IRQ8 - IRQ15 โดยอินเทอร์รัพต์คอนโทรลเลอร์ตัวที่สอง (Second interrupt controller)

hardware_irq

ต้องกำหนดให้ตรงกับอินเทอร์รัพต์ของเนตเวอร์กการ์ดใช้อยู่ สังเกตได้จากตำแหน่งของ *จัมป์เปอร์ (Jumper)* บนเนตเวอร์กการ์ด

I/O Address

เป็นหมายเลขของตำแหน่ง I/O ซึ่งต้องกำหนดให้ตรงกับที่ใช้อยู่ ของเนตเวอร์กการ์ด

หมายเหตุ *แพคเก็ตไดรเวอร์ จะทำหน้าที่เป็น รูทีนบริการการอินเทอร์รัพต์ (Interrupt Service Routine) ของเนตเวอร์กการ์ด เพราะฉะนั้นการเรียกใช้งาน จะต้องไม่มีโปรแกรมใดที่ทำหน้าที่ในลักษณะเดียวกันนี้ ทำงานอยู่ก่อน อาทิเช่น ไดรเวอร์ของเนตเวอร์กการ์ด*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 การเขียนโปรแกรมติดต่อกับแพคเก็ตไดรเวอร์

การทำงานของแพคเก็ตไดรเวอร์ แบ่งออกเป็น 3 ระดับ

1. Basic Packet Driver มีฟังก์ชันการทำงานให้เรียกใช้น้อยที่สุด แต่สามารถนำไปใช้ได้ง่าย อีกทั้งยังใช้ทรัพยากรของเครื่องน้อย การทำงานในระดับชั้นนี้คือ การส่งข้อมูลแบบกระจาย และการรับแพคเก็ตข้อมูล
2. Extended Packet Driver ครอบคลุมฟังก์ชันการทำงานของ Basic Packet Driver แต่ความสามารถในการส่งข้อมูลแบบกระจายเป็นกลุ่ม (Multicast) และเก็บรวบรวมสถิติการใช้งานแพคเก็ตไดรเวอร์
3. High Performance มีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพการทำงานของแพคเก็ตไดรเวอร์

การติดต่อกับแพคเก็ตไดรเวอร์

การติดต่อกับแพคเก็ตไดรเวอร์ ทำได้โดยการเรียกใช้ **ซอฟท์แวร์อินเทอร์รัพต์** ในช่วง 0X60 ถึง 0X80 โดยที่จะเรียกใช้การทำงานใดนั้นขึ้นกับค่าที่ผ่านให้ทางรีจิสเตอร์ AH

ตัวอย่างการเรียกใช้ฟังก์ชันที่จะแสดงต่อไปนี้ เขียนอยู่ในรูปแบบของ Microsoft C และ ภาษา Assembly แพคเก็ตไดรเวอร์จะเก็บค่าของรีจิสเตอร์ต่างๆที่ต้องนำไปใช้งาน ในขณะที่เข้าไปทำงานนั้นๆ และจะคืนค่าดังกล่าวเมื่อทำงานเสร็จ ค่าของรีจิสเตอร์ที่ต้องเก็บคือ DS, SS, SP และแฟลก

ฟังก์ชันการทำงานของแพคเก็ตไดรเวอร์

1. Driver_info( )

```
      driver_info (handle)          AH == 1, AL=255
            int handle;             BX     / * Optional */

error return:
      carry flag set
      error code                    DH
      possible errors:
            BAD_HANDLE                      /* Older drivers only */

non-error return:
      carry flag clear
      version                       CH
      class                         DX
      type                          CX
      number                        CL
      name                          DS:SI
      funtionality                  AL
                                    1==basic functions present.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                                    2==basic and exteded present.
                                    5==basic and high-performance.
                                    6==basic, high-performance,
                                       extend.
                                    255==not installed.
```

จะแสดงข้อมูลต่างๆที่เกี่ยวข้องกับแพคเก็ตไดรเวอร์ที่มีการเรียกใช้ ค่าเวอร์ชันจะเป็นที่สมมติขึ้นเพื่อให้ทราบถึงความสามารถในการทำงานได้ ในเวอร์ชันก่อนหน้าเวอร์ชัน 1.07 การเรียกใช้ฟังก์ชันนี้จะต้องส่งค่า handle ให้ด้วย ซึ่งค่า handle นี้ได้มาจากการทำงานในฟังก์ชัน access_type

2. Access_type( )

```
      int access_type( if_class, if_type, if_number, type, typelen,
receiver)                              AH=2
      int if_class;                    AL
      int if_type;                     BX
      int if_number;                   DL
      char far *type;                  DS:SI
      unsigned typelen;                CX
      int (far *receiver)( );          ES:DI

error return:
      carry flag set
      error code                       DH
      possible errors:
            NO_CLASS
            NO_TYPE
            NO_NUMBER
            BAD_TYPE
            NO_SPACE
            TYPE_INUSE

non-error return:
      carry flag clear
      handle                           AX

receiver call:
      (*receiver)(handle, flag, len [,buffer])
      int handle;                      BX
      int flag;                        AX
      unsigned len;                    CX
if AX==1
      char far *buffer;                DX:SI
```

ใช้เพื่อตรวจสอบข้อมูลแพคเก็ตว่าเป็น type ตามที่ตรวสอบหรือไม่ ค่าของ type ที่ส่งผ่านให้นั้นจะเป็นตัวชี้ที่ชี้ไปยังตำแหน่งส่วนที่แสดง type ของแพคเก็ตนั้นๆ ค่า typelen จะเป็นขนาดความยาวของแพคเก็ต ค่า receiver จะเป็นตัวชี้ตำแหน่งของส่วนการทำงาน เมื่อได้รับแพคเก็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อได้รับแพคเก็ต ส่วนของ receiver จะถูกเรียกทำงาน 2 ครั้ง ครั้งที่ 1 จะเรียกเพื่อให้แอพพลิเคชัน เตรียมสำรองเนื้อที่บัฟเฟอร์ไว้ สำหรับเก็บแพคเก็ต สังเกตได้จากค่าในรีจิสเตอร์ AX=0 แอพพลิเคชันจะคืนค่าตำแหน่งบัฟเฟอร์โดยใช้ ES:DI เป็นตัวชี้ ถ้าหากว่าแอพพลิเคชันไม่สามารถหาเนื้อที่ได้เพียงพอ ก็จะส่งค่า 00:00 ผ่านทาง ES:DI ซึ่งจะทำให้แพคเก็ตไดรเวอร์ ทิ้งแพคเก็ตที่ได้รับมานั้น และจะไม่ทำการเรียก receiver ทำงานเป็นครั้งที่ 2

ค่าความยาวของแพคเก็ตไดรเวอร์ที่เข้ามานั้น จะเก็บอยู่ใน CX เมื่อตอนเรียกให้ receiver ครั้งแรก เพื่อให้แอพพลิเคชันทราบว่า จะต้องสำรองเนื้อที่ขนาดเท่าไร

ในการเรียก receiver ครั้งที่ 2 จะเป็นการบอกให้ทราบว่า ได้ทำสำเนาแพคเก็ตลงในบัฟเฟอร์เรียบร้อยแล้ว สังเกตได้จากรีจิสเตอร์ AX จะมีค่าเท่ากับ 1

3. Release_type( )

```
      int release_type(handle)        AH==3
            int handle;               BX

error return:
      carry flag set
      error code                      DH
      possible errors:
            BAD_HANDLE

non-error return:
      carry flag clear
```

ฟังก์ชันนี้จะเป็นการบอกเลิกค่า handle ที่ได้มาจากการทำงานในฟังก์ชัน access_type( )

4. Send_pkt( )

```
      int send_pkt(buffer, length)    AH==4
            char far *buffer;         DS:SI
            unsigned length;          CX

error return:
      carry flag set
      error code                      DH
      possible errors:
            CANT_SEND

non-error return:
      carry flag clear
```

เป็นฟังก์ชันสำหรับส่งแพคเก็ต แอพพลิเคชันที่เรียกใช้ฟังก์ชันนี้ จะต้องเตรียมข้อมูลแพคเก็ตให้พร้อมสรรพ รวมถึงในส่วนที่เป็น MAC เฮดเดอร์ที่แสดงค่าหมายเลขที่อยู่ต่างๆด้วย เพราะอยู่นอกเหนือการทำงานของแพคเก็ตไดรเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. Terminate( )

```
      terminate(handle)              AH==5
            int handle;              BX

error return:
      carry flag set
      error code                     DH
      possible errors:
            BAD_HANDLE
            CANT_TERMINATE

non-error return:
      carry flag clear
```

ทำงานสัมพันธ์กับค่า handle ซึ่งหากการทำงานไม่เกิดปัญหาใดๆ จะเป็นการคืนหน่วยความจำที่ถูกใช้โดยแพคเก็ตไดรเวอร์ ให้กับดอส

6. Get_address( )

```
      get_address(handle, buf, lan) AH==6
            int handle;             BX
            char far *buf           ES:DI
            int len;                CX

error return:
      carry flag set
      error code                    DH
      possible errors:
            BAD_HANDLE
            NO_SPACE

non-error reutrn:
      carry flag clear
      length                        CX
```

เพื่อเก็บค่าหมายเลขที่อยู่ของเครือข่าย ลงในบัฟเฟอร์ โดยจะมีความยาวเท่ากับค่า len และจะส่งคืนค่าจำนวนไบต์ที่ได้รับการเก็บลงในบัฟเฟอร์ผ่านทางรีจิสเตอร์ CX ถ้าหากว่าเกิดความผิดพลาด NO_SPACE ขึ้น แสดงว่าเนื้อที่บัฟเฟอร์ ไม่เพียงพอในการจัดเก็บ

7. Reset_interface( )

```
      reset_interface(handle)       AH==7
            int handle;             BX

error return:
      carry flag set
      error code                    DH
      possible errors:
            BAD_HANDLE
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            CANT_RESET

non-error return:
      carry flag clear
```

เป็นฟังก์ชันการทำงานเพื่อรีเซตแพคเก็ตไดรเวอร์ คือยกเลิกแพคเก็ตที่เตรียมส่ง เตรียมพร้อมให้เริ่มรับแพคเก็ตใหม่ กำหนดโหมดการรับแพคเก็ตไว้ที่โหมด 3 (รับเฉพาะแพคเก็ตของตน และแบบกระจายทุกจุด)

8. Get_parameters( )

```
High-performance driver function
      get_parameters( )                AH==10

error return:
      carry flag set
      error code                       DH
      possible errors:
            BAD_COMMAND                /* No high-performance support */

non-error return:
      carry flag clear
      struct param far *;              ES:DI

struct param {
  unsigned char major_rev;          /* Revision of Packet Driver spec */
  unsigned char minor_rev;          /*    this driver conforms to. */
  unsigned char length;             /* Length of structure in bytes */
  unsigned char addr_len;           /* Length of a MAC-layer address */
  unsigned short mtu;               /* MTU. including MAC headers */
  unsigned short multicast_avai;    /* Buffer size for multicast addr */
  unsigned short rcv_bufs;          /* (# of back-to-back MTU rcvs)-1 */
  unsigned short xmt_bufs;          /* (#of successive xmits)-1 */
  unsigned short int_num;           /* Interrupt # to hook for post-EOI
                                       processing. 0==none */
};
```

ใช้เพื่อปรับเพิ่มประสิทธิภาพการทำงานของแอพพลิเคชันที่ใช้งานแพคเก็ตไดรเวอร์ โดยฟังก์ชันนี้จะคืนค่าต่างๆ ที่จำเป็นต่อการปรับแต่งแอพพลิเคชันให้เหมาะสมกับการทำงานของแพคเก็ตไดรเวอร์นั้นกลับมาทาง param ฟังก์ชันนี้ได้เริ่มมีในแพคเก็ตไดรเวอร์ตั้งแต่เวอร์ชัน 1.09 (ตรวจสอบโดยใช้ driver_info ( ))

9. As_send_pkt( )

```
High performace driver function
      int as_send_packet( buffer, length, upcall)  AH==11
      char far *buffer;              DS:SI
      unsigned length;               CX
      int (far *upcall)( );          ES:DI
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
error return:
      carry flag set
      error code                    DH
      possible errors:
            CANT_SEND               /* transmit error, re-entered,
                                       etc. */
            BAD_COMMAND             /* Level 0 or 1 driver */

non-error return:
      carry flag clear

buffer available upcall:
      (*upcall)(buffer, result)
        int result;             AX  /* 0 for copy ok */
        char far *buffer;       ES:DI /* from as_send_pkt() call */
```

การทำงานคล้ายคลึงกับ send_pkt( ) แต่ต่างกันตรงที่บัฟเฟอร์ที่ใช้เก็บข้อมูลแพคเก็ตนั้น หากเป็น send_pkt( ) จะต้องรอจนกว่าจะทำงานในส่วนของ send_pkt( ) เสร็จเสียก่อน จึงจะสามารถนำไปใช้ได้ ส่วนหากเป็น as_send_pkt( ) สามารถนำไปใช้ได้ทันที ที่เรียกใช้ฟังก์ชันนี้ ฟังก์ชันนี้ได้เริ่มมีในแพคเก็ตไดรเวอร์ตั้งแต่เวอร์ชัน 1.09 (ตรวจสอบโดยใช้ driver_info ( ))

10. Set_rcv_mode( )

```
Extended driver function
      set_rcv_mode( handle, mode)   AH==20
            int handle;             BX
            int mode;               CX

error return:
      carry flag set
      error code                    DH
      possible errors:
            BAD_HANDLE
            BAD_MODE

non-error return:
      carry flag clear
```

จะต้องผ่านค่า handle ให้กับฟังก์ชันนี้ เพื่อกำหนดโหมดในการรับแพคเก็ต โหมดในการรับแพคเก็ตมีดังนี้

1. ไม่รับแพคเก็ตใดๆ
2. รับเฉพาะที่ส่งมายังเครื่องนี้
3. โหมด 2 + แบบที่ส่งกระจายทุกจุด
4. โหมด 3 + แบบที่ส่งกระจายเป็นกลุ่ม ตามที่กำหนดไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. โหมด 3 + แบบที่ส่งกระจายเป็นกลุ่ม ทุกๆกลุ่ม
6. รับทุกๆแพคเก็ต

แพคเก็ตไดรเวอร์บางตัว ไม่สามารถทำการรับแพคเก็ตในบางโหมดได้ ในส่วนการรับแพคเก็ตแบบกระจายเป็นกลุ่ม สามารถกำหนดรายละเอียดได้ (ฟังก์ชัน get_multicast_lis( ) และ set_multicast_list) และหากไม่มีการกำหนดค่าโหมดการทำงานเป็นอื่น แพคเก็ตไดรเวอร์จะกำหนดให้ค่าโหมด 3 เป็นค่าเริ่มต้น

11. Get_rcv_mode( )

```
Extended driver function
      get_rcv_mode( handle, mode)    AH==21
            int handle;              BX

error return:
      carry flag set
      error code                     DH
      possible errors:
            BAD_HANDLE

non-error return:
      carry flag clear
      mode                           AX
```

แสดงค่าโหมดการรับแพคเก็ต ที่ได้กำหนดไว้ณ เวลาปัจจุบัน

12. Set_multicast_list( )

```
Extended driver function
      set_multicast_list( addrlst, len)   AH==22
            char far *addrlst;            ES:DI
            int len;                      CX

error return:
      carry flag set
      error code
      possible errors:
            NO_MULTICAST
            NO_SPACE
            BAD_ADDRESS

non-error return:
      carry flag clear
```

เป็นฟังก์ชันเพื่อกำหนดค่าของหมายเลขที่อยู่ ซึ่งจะใช้เป็นหมายเลขที่อยู่ในการทำงานแบบรับส่งแพคเก็ตเป็นกลุ่ม อาจจะใช้ร่วมกับ get_multicast_list( ) เพื่อนำค่าที่กำหนดไว้ในปัจจุบัน มาปรับเปลี่ยนแก้ไข

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

13. Get_multicast_list( )

```
Extended driver function
      get_multicast_list( )          AH==23

error return:
      carry flag set
      error code                     DH
      possible errors:
            NO_MULTICAST
            NO_SPACE

non-error return:
      carry flag clear
      char far *addrlst;             ES:DI
      int len;                       CX
```

เป็นฟังก์ชันเพื่อแสดงค่าของ multicast list ที่ใช้อยู่ในปัจจุบัน

14. Get_statistics( )

```
Extended driver function
      get_statistics( handle)        AH==24
            int handle;              BX

error return:
      carry flag set
      error code                     DH
      possible errors:
            BAD_HANDLE

non-error return:
      carry flag clear
      char far *stats;               DS:SI

struct statistics {
  unsigned long packet_in      /* Totals across all handles */
  unsigned long packet_out;
  unsigned long bytes_in;      /* Including MAC headers */
  unsigned long bytes_out;
  unsigned long errors_in;     /* Totals across all error types */
  unsigned long errors_out;
  unsigned long packets_lost; /* No buffer from receiver( ), card */
                               /* out of resources, etc */
};
```

แสดงค่าสถิติของแพคเก็ตที่มีการรับส่ง ผ่านแพคเก็ตไดรเวอร์

15. Set_address( )

```
Extended driver function
      set_address( addr, len)        AH==25
            char far *addr;          ES:DI
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            int len;                    CX

error return:
      carry flag set
      error code                        DH
      possible errors:
            CANT_SET
            BAD_ADDRESS

non-error return:
      carry flag clear
      length                            CX
```

ใช้เพื่อกำหนดหมายเลขที่อยู่ให้มีรูปแบบ สัมพันธ์กับการทำงานของโพรโตคอลที่อยู่ในระดับชั้นบน

สรุปหมายเลขฟังก์ชันเพื่อใช้ติดต่อกับแพคเก็ตไดรเวอร์

หมายเลขฟังก์ชันในรูปแบบเลขฐานสิบต่อไปนี้ จะใช้ติดต่อกับแพคเก็ตไดรเวอร์ เพื่อให้ทำงานตามที่ต้องการ โดยหมายเลขดังกล่าวจะถูกส่งไปยังแพคเก็ตไดรเวอร์ ผ่านทางรีจิสเตอร์ AH

driver_info	1
access_type	2
release_type	3
send_pkt	4
terminate	5
get_address	6
reset_interface	7
+get_parameters	10
+as_send_pkt	11
*set_rcv_mode	20
*get_rcv_mode	21
*set_multicast_list	22
*get_multicast_list	23
*get_statistics	24
*set_address	25

+ เป็นฟังก์ชันการทำงานของ High-performance packet driver

* เป็นฟังก์ชันการทำงานของ Extended packet driver

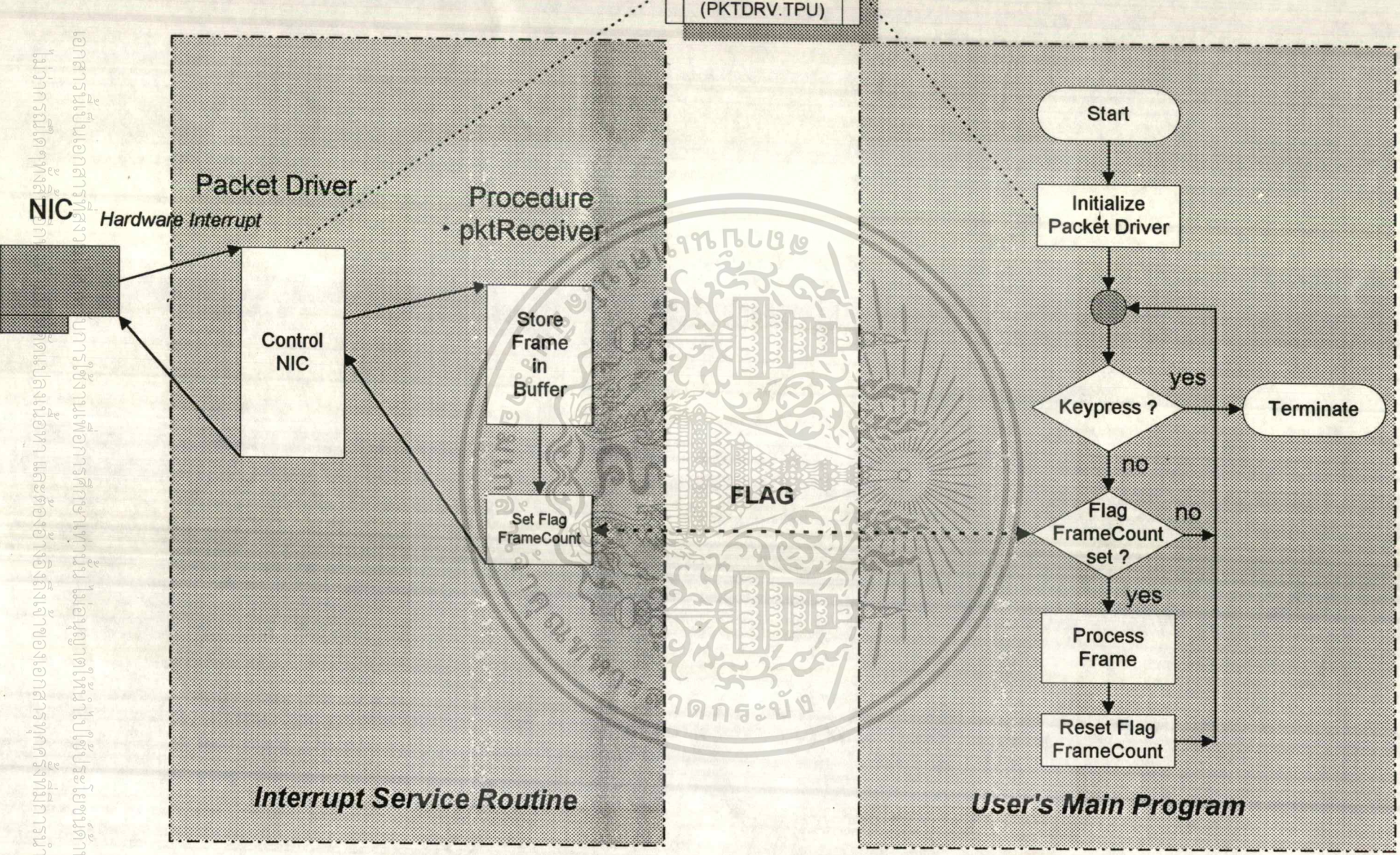
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่าในช่วงของ 128 ถึง 255 (0X80 ถึง 0xFF) สำหรับให้ผู้ใช้กำหนดฟังก์ชันการทำงานเอง

สรุปหมายเลขแสดงข้อผิดพลาด

1. BAD_HANDLE	ค่า handle ผิดพลาด
2. NO_CLASS	ไม่พบ class ดังกล่าว
3. NO_TYPE	ไม่พบ type ดังกล่าว
4. NO_NUMBER	ไม่พบ No ดังกล่าว
5. BAD_TYPE	ไม่รู้จักชนิดแพคเก็ตนั้น
6. NO_MULTICAST	ไม่สามารถทำงานแบบส่งกระจายเป็นกลุ่มได้
7. CANT_TERMINATE	ไม่สามารถออกจากโปรแกรมแพคเก็ตไดรเวอร์
8. BAD_MODE	กำหนดโหมดที่ใช้ในการรับแพคเก็ตผิดพลาด
9. NO_SPACE	การทำงานผิดพลาด อันมีสาเหตุมาจากเนื้อที่ไม่เพียงพอ
10. TYPE_INUSE	ชนิดแพคเก็ตนั้น กำลังทำงานอยู่
11. BAD_COMMAND	ไม่รู้จักคำสั่งนั้นๆ
12. CANT_SEND	ไม่สามารถส่งแพคเก็ตออกไปได้ (มักมีสาเหตุมาจากฮาร์ดแวร์)
13. CANT_SET	ไม่สามารถแก้ไขค่าหมายเลขที่อยู่ของฮาร์ดแวร์ได้
14. BAD_ADDRESS	ค่าหมายเลขที่อยู่ผิดพลาด
15. CANT_RESET	ไม่สามารถทำการรีเซตได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.2 แสดงการเขียนโปรแกรมติดต่อเพื่อรับเฟรมข้อมูลจากแพคเก็ตไดรเวอร์

3.4 วิธีการเขียนโปรแกรมติดต่อกับแพคเก็ตไดรเวอร์ เพื่อรับเฟรมข้อมูล

การเขียนโปรแกรมเพื่อรับเฟรมข้อมูล โดยทำงานติดต่อกับแพคเก็ตไดรเวอร์ จะแบ่งการทำงานออกเป็น 2 ส่วนคือ

1. ส่วนรูทีนบริการอินเทอร์รัพต์ (Interrupt Service Routine)
2. ส่วนโปรแกรมของผู้ใช้ (User's Main Program)

และทั้งสองส่วนนี้จะติดต่อกันโดยใช้ **แฟลก (Flag)**

1. ส่วนรูทีนบริการอินเทอร์รัพต์ (Interrupt Service Routine) แบ่งออกเป็น 2 ส่วนย่อยคือ

1.1 ส่วนของแพคเก็ตไดรเวอร์ (Packet Driver)

ทำหน้าที่เป็น *รูทีนบริการอินเทอร์รัพต์ของฮาร์ดแวร์* ควบคุมเนตเวอร์กการ์ด เมื่อมีอินเทอรัพต์เกิดขึ้นกับเนตเวอร์กการ์ด อาทิเช่น เมื่อตอนที่เนตเวอร์กการ์ดรับข้อมูลเข้ามา ก็จะเข้าไปทำในรูทีนดังกล่าวนี้

1.2 โพรซีเดอร์ที่ทำหน้าที่ส่งสัญญาณผ่านแฟลก (Procedure pktReceiver)

ทำหน้าที่ติดต่อกับส่วนโปรแกรมหลักของผู้ใช้ โดยจะถูกเรียกใช้โดยแพคเก็ตไดรเวอร์

โพรซีเดอร์นี้ จะถู กเรียกใช้ 2 ครั้งโดยแพคเก็ตไดรเวอร์ ทุกครั้งที่มีเฟรมเข้ามา 1 เฟรม

ครั้งที่ 1. แพคเก็ตไดรเวอร์จะเรียกใช้โพรซีเดอร์ครั้งแรกนี้ เพื่อให้ทำการ **สำรองเนื้อที่บัฟเฟอร์** รอไว้ โดยตรวจสอบได้จากค่าที่ผ่านมาทางรีจิสเตอร์ AX จะเท่ากับศูนย์

AX=0

CX=ขนาดของเฟรมที่รวมส่วนเฮดเดอร์ ประกอบด้วยฟิลด์ที่อยู่ปลายทาง, ต้นทางและชนิดแพคเก็ต นั่นหมายความว่าขนาดเฟรมจะเท่ากับขนาดข้อมูลในฟิลด์ข้อมูล บวก 18 ไบต์

คุณจะต้องส่งตัวชี้ (pointer) ของตำแหน่งบัฟเฟอร์ที่ว่าง โดยผ่านทางรีจิสเตอร์ ES:DI ไปให้กับแพคเก็ตไดรเวอร์ หากไม่ต้องการจะรับเฟรมข้อมูลนั้นให้ใส่ค่า 0000:0000 ที่รีจิสเตอร์ ES:DI

การสำรองเนื้อที่บัฟเฟอร์จะต้องทำตั้งแต่ตอนเรียกโปรแกรม โดยมักจะใช้ในรูปแบบของข้อมูลแบบอาร์เรย์(Array) ถ้าไม่เช่นนั้นแล้วจะไม่สามารถสำรองเนื้อที่ได้ เพราะไม่สามารถเรียกใช้การทำงานของ ดอสได้ (Int 21H) ขณะกำลังอยู่ในรูทีนบริการอินเทอร์รัพต์ (Interrupt Service Routine)

ครั้งที่ 2. แพคเก็ตไดรเวอร์จะเรียกใช้โพรซีเดอร์นี้อีกครั้งหนึ่งเมื่อข้อมูลเฟรมนั้นๆ **ได้จัดเก็บลงในบัฟเฟอร์ที่สำรองไว้ในตอนแรกเรียบร้อยแล้ว** โดยตรวจสอบได้จากค่าที่ผ่านมาทางรีจิสเตอร์ AX จะเท่ากับหนึ่ง เป็นการบอกให้ทราบว่าเฟรมข้อมูลทั้งหมด โดยรวมส่วนเฮด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เดอร์ (ขนาด 6+6+2 ไบต์) ได้รับการทำสำเนามาเก็บไว้อยู่ในบัฟเฟอร์ เพื่อให้เรานำไปใช้งานได้แล้ว การทำงานในส่วนโพรซีเดอร์นี้เป็นส่วนหนึ่งของรูทีนบริการอินเทอร์รัพต์ จึงไม่ควรให้ทำงานอยู่นาน ไม่เช่นนั้นแล้วจะไม่สามารถรับเฟรมข้อมูลถัดไปได้ทัน เนื่องจากอินเทอร์รัพต์จะถูกปิดกั้นไว้ไม่ให้ไปทำงานในส่วนรูทีนบริการอินเทอร์รัพต์ซ้ำซ้อนกัน จะสามารถทำงานในรูทีนอีกครั้งหนึ่งได้ ก็ต่อเมื่อออกจากรูทีนนั้นเสียก่อน

2. **ส่วนโปรแกรมหลักของผู้ใช้** (User's Main Program) แบ่งออกเป็น 3 ส่วนย่อย

2.1 **ส่วนการกำหนดค่าเริ่มต้นให้กับแพคเก็ตไดรเวอร์** (Initialize Packet Driver)

ทำหน้าที่กำหนดรายละเอียดต่างๆ ในการทำงานของแพคเก็ตไดรเวอร์ ที่สำคัญคือกำหนดให้ *ตัวชี้ตำแหน่งการทำงาน* เมื่อมีแพคเก็ตเข้ามา ชี้มายังตำแหน่งโพรซีเดอร์ที่ทำหน้าที่ส่งสัญญาณแฟลก

2.2 **ส่วนการประมวลผลเฟรม** (Procedure Process)

โดยจะเข้ามาทำงานในส่วนนี้ได้ก็ต่อเมื่อได้รับ *สัญญาณแฟลก* ซึ่งหมายถึงข้อมูลของเฟรมได้มาเก็บอยู่ในบัฟเฟอร์เรียบร้อยแล้ว แต่จะนำไปใช้งานอย่างไรบ้างขึ้นอยู่กับผู้ใช้ อาทิเช่นนำข้อมูลที่อยู่ในบัฟเฟอร์ แสดงผลออกทางหน้าจอคอมพิวเตอร์

2.3 **ส่วนที่จะทำงานเมื่อออกจากโปรแกรม** (Terminate)

ซึ่งจะคืนค่าต่างๆให้กับตัวแพคเก็ตไดรเวอร์ ที่สำคัญคือจะคืนค่าของ *ตัวชี้* ที่ชี้มายังโพรซีเดอร์ที่ทำหน้าที่ส่งสัญญาณแฟลก

ลำดับขั้นตอนในการทำงานจะเป็นดังนี้

1. เรียก **แพคเก็ตไดรเวอร์** ให้ฝังอยู่ในหน่วยความจำเสียก่อน ตัวแพคเก็ตไดรเวอร์จะเข้าไปควบคุมการทำงานของเนตเว่อร์กการ์ด
2. เรียกใช้ **โปรแกรมหลักของผู้ใช้** โดยลำดับขั้นตอนในโปรแกรมหลักของผู้ใช้จะเป็นดังนี้
   - ทำการ **กำหนดค่าเริ่มต้น** ของแพคเก็ตไดรเวอร์ ซึ่งจะทำให้ ตัวชี้ตำแหน่งการทำงานเมื่อมีแพคเก็ตเข้ามา ชี้มาที่ส่วนโพรซีเดอร์ที่ทำหน้าที่ส่งสัญญาณแฟลก
   - หลังจากนั้นตัวโปรแกรมหลักนี้ จะไป **หยุดรอ** ว่าเมื่อไรจะมีเฟรมของข้อมูลส่งมา
   - เมื่อมีเฟรมของข้อมูลส่งมาแล้ว จึงไปทำการ **ประมวลผลเฟรมข้อมูล** ชุดนั้นๆ
3. ในกรณีที่ **ไม่มีเฟรมข้อมูล** ส่งเข้ามา
   - ส่วนของโพรซีเดอร์ที่ทำหน้าที่ส่งสัญญาณแฟลก จะไม่ทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ส่วนของ โปรแกรมหลักของผู้ใช้ จะไปหยุดรอจนกว่าจะมีเฟรมส่งเข้ามา

4. ในกรณีที่ **มีเฟรมข้อมูล** ส่งเข้ามา
   - ส่วนของรูทีนบริการอินเทอร์รัพต์จะทำงาน โดยเริ่มจาก **รูทีนบริการอินเทอร์รัพต์ของฮาร์ดแวร์** *(แพคเก็ตไดรเวอร์)* ก่อน และจะเข้าไปทำ **โพรซีเดอร์ที่ทำหน้าที่ส่งสัญญาณแฟลก** *(โพรซีเดอร์ pktReceiver)* ซึ่งจะไป**เซตค่า แฟลก** FrameCount เพื่อไปบอกโปรแกรมหลักของผู้ใช้ให้เริ่มทำงานได้
   - ส่วน *โปรแกรมหลักของผู้ใช้* เมื่อเห็นว่าค่า แฟลก FrameCount เซตแล้วซึ่งหมายความว่าข้อมูลในเฟรมได้มาเก็บอยู่ในบัฟเฟอร์เรียบร้อยแล้ว จะเข้าไปทำงานใน ส่วนประมวลผลเฟรม และเมื่อทำงานเสร็จแล้ว จะกลับไป **รีเซตค่าแฟลก** เพื่อให้พร้อมรับเฟรมข้อมูลตัวถัดไป
5. ก่อนออกจากโปรแกรมหลักของผู้ใช้ จะทำการคืนค่าที่กำหนดไว้กับแพคเก็ตไดรเวอร์ เมื่อตอนเริ่มต้นโปรแกรม โดยการเรียกใช้ **ส่วนการออกจากโปรแกรม**

โพรซีเดอร์ทำหน้าที่ส่งสัญญาณแฟลก (โพรซีเดอร์ pktReceiver)

```
VAR LastFrame   : Array[00..1520] of Byte; { บัฟเฟอร์สำหรับเก็บเฟรมข้อมูลที่ส่งเข้ามา }
    FrameCount  : Word;                    { แฟลกที่ใช้บอกว่าเฟรมข้อมูลเก็บอยู่ในบัฟเฟอร์แล้ว }
    FrameLength : Word;                    { ความยาวของเฟรมสุดท้ายที่ส่งเข้ามา }

$S-PROCEDURE pktReceiver; ASSEMBLER;
ASM
  PUSH AX                  { เก็บค่ารีจิสเตอร์ลงแสตกค์ }
  PUSH BX
  PUSH CX
  PUSH DX

  CMP  AX,0001             { AX=1 หมายถึงเฟรมข้อมูลได้ทำสำเนาไว้ในบัฟเฟอร์เสร็จแล้ว }
  JZ   @@FrameCopied
  CMP  AX,0000             { AX=0 หมายถึงให้เตรียมเนื้อที่บัฟเฟอร์ไว้ }
  JZ   @@AllocMemory
  JMP  @@EXIT              { ในกรณีที่ค่าในรีจิสเตอร์ AX เป็นอย่างอื่นให้ออกจากโปรแกรม }

@@AllocMemory:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
MOV  DX,0                  { ES:DI = 0000:0000, ไม่ต้องการรับแพคเก็ต }
MOV  ES,DX
MOV  DI,0

MOV  DX,SEG FrameCount     { กำหนดค่าส่วนของดาตาเซกเมนต์ }
MOV  DS,DX
MOV  DX,FrameCount         { ตรวจสอบว่าบัฟเฟอร์ว่างหรือไม่ ( FrameCount = 0 ) }
CMP  DX,0
 JNZ @@Exit          { กรณีที่บัฟเฟอร์ไม่ว่างต้องออกจากโปรแกรม จึงให้ค่าใน ES:DI = 0000:0000 }

MOV  DX,SEG LastFrame  { ส่งตำแหน่งบัฟเฟอร์ที่พร้อมจะเก็บเฟรมข้อมูลให้กับแพคเก็ตไดรเวอร์ }
MOV  ES,DX             { โดยส่งผ่านทาง ES:DI }
MOV  DI,OFFSET LastFrame

MOV  DX,SEG FrameLength
MOV  DS,DX
MOV  SI,OFFSET FrameLength
MOV  WORD PTR DS:[SI],CX   { เพื่อเก็บค่าความยาวของเฟรม }

JMP  @@Exit

@@FrameCopied:

MOV  DX,SEG FrameCount     { กำหนดดาตาเซกเมนต์ }
MOV  DS,DX
MOV  FrameCount,1          { เซตค่าแฟลกให้เป็นหนึ่ง }

{ ขณะนี้บัฟเฟอร์มีข้อมูลของเฟรมแล้ว ( FrameCount = 1). }
{  และจะไม่รับเฟรมใดๆอีกจนกว่าจะรีเซต แฟลก ( FrameCount= 0) }

@@Exit:

 POP  DX                   { คืนค่ารีจิสเตอร์ทั้งหมด }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  POP  CX
  POP  BX
  POP  AX
END;
$S+
```

ส่วนโปรแกรมหลักของผู้ใช้

```
BEGIN
  InitPktDriver; { เรียกใช้โปรแกรมส่วนกำหนดค่าเริ่มต้นให้กับแพคเก็ตไดรเวอร์ }
  REPEAT
    IF (FrameCount = 1) THEN          { ถ้าแฟลกเซต }
      ProcessFrame; { เรียกใช้โปรแกรมส่วนการประมวลเฟรมข้อมูล }
  UNTIL KeyPressed;
  TerminatePktDriver; { เรียกใช้โปรแกรมส่วนออกจากโปรแกรมหลักของผู้ใช้ }
END.
```

โพรซีเดอร์ทำหน้าที่กำหนดค่าเริ่มต้นให้กับแพคเก็ตไดรเวอร์ (Initialize Packet Driver)

```
 VAR pktDriver      : TPKTDRVR;    { ประกาศให้รู้จักออปเจกต์ของแพคเก็ตไดรเวอร์ }
    pktDriverInfo  : TDRVRINFO;   { ประกาศเรคอร์ดเพื่อรับข้อมูลจากแพคเก็ตไดรเวอร์ }
    pktDriverAccess: TACCESSTYPE; { ประกาศเรคอร์ดเพื่อใช้ติดต่อกับแพคเก็ตไดรเวอร์ }

PROCEDURE InitPktDriver;
BEGIN
  WriteLn('Initializing packet driver....');
  IF (pktDriver.GetStatus <> INITIALIZED) THEN
  BEGIN
    WriteLn('Could not initialze packet driver...');
    WriteLn('Aborting...');
    Halt($FF);
  END
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ELSE
BEGIN

  { พบแพคเก็ตไดรเวอร์ฝังตัวในหน่วยความจำ แสดงข้อมูลบางส่วน }
  WriteLn('Packet driver found:');
  pktDriver.DriverInfo(pktDriverInfo);

  WriteLn('Name    =',StrPas(pktDriverInfo.PName));
  WriteLn('Version = ',pktDriverInfo.Version);
  WriteLn('IF-Type = ',pktDriverInfo.Type_);
  Write  ('Func    = ');

  CASE pktDriverInfo.Functionality OF
    01 : WriteLn('Basic functions present.');
    02 : WriteLn('Basic & extended functions present.');
    05 : WriteLn('Basic & high-performance functions present.');
    06 : WriteLn('Basic, high-performance & extended functions present.');
  END;
END;

        { ใส่ข้อมูลที่จะใช้ติดต่อกับแพคเก็ตไดรเวอร์ }
WITH pktDriverAccess DO
BEGIN
  if_class  := pktDriverInfo.Class;
  if_type   := ANYTYPE;
  if_number := 0;
  type_     := @TypeField;
  typelen   := 0;
  receiver  := @pktReceiver; { บอกให้แพคเก็ตไดรเวอร์นำตัวชี้ตำแหน่งการทำงาน }
                             { เมื่อได้รับแพคเก็ตข้อมูล มาชี้ที่โพรซีเดอร์ pktReceiver }
END;

{ ติดต่อกับแพคเก็ตไดรเวอร์ }
pktDriver.AccessType(pktDriverAccess);
WriteLn('Handle  = ',pktDriver.GetHandle);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
WriteLn;

        { บอกโหมดในการรับข้อมูล โดยในที่นี้จะเซตเป็นโหมด promiscuous }
        { ซึ่งทำให้สามารถรับข้อมูลได้ทุกเฟรม โดยรวมข้อมูลที่ส่งแบบกระจายด้วย }
        { จึงควรระวัง หากนำไปใช้กับระบบเครือข่ายที่มีการรับส่งเฟรมจำนวนมาก }

 pktDriver.SetRCVmode(6);
END;
```

โพรซีเดอร์ ส่วนประมวลผลเฟรม (Process Frame)

```
BEGIN
        { นำเฟรมของข้อมูลที่ได้มาประมวลผล }
  .
  .
  .
  FrameCount := 0; { รีเซตแฟลก เพื่อให้พร้อมรับข้อมูลเฟรมต่อไป }
END;
```

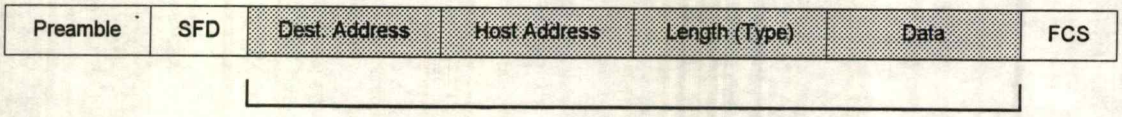
โพรซีเดอร์ ทำหน้าที่คืนค่าต่างๆ ให้กับโปรแกรมแพคเก็ตไดรเวอร์ (Terminate)

```
PROCEDURE TerminatePktDriver;
BEGIN
  pktDriver.ReleaseType; { คืนค่าต่างๆของแพคเก็ตไดรเวอร์ }
END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 ส่วนของข้อมูลที่แพคเก็ตไดรเวอร์ รับได้

ลักษณะของข้อมูลที่รับได้ จะเป็น **เฟรมอีเทอร์เนต** และ IEEE 802.3 ซึ่งทำงานอยู่ในระดับชั้นดาตาลิงค์ของ OSI



รูปที่ 3.3 แสดงส่วนของข้อมูลที่แพคเก็ตไดรเวอร์รับได้

ฟิลด์ Preamble
ขนาด 7 ไบต์
จะเก็บค่าเลขฐานสอง 10101010 ในทุกๆไบต์

ฟิลด์ SFD
ขนาด 1 ไบต์
จะเก็บค่าเลขฐานสอง 10101011

แพคเก็ตไดรเวอร์จะคอยตรวจสอบสัญญาณที่ส่งผ่านเนตเวอร์กการ์ด หากพบสัญญาณที่เหมือนกับค่าที่อยู่ในสองฟิลด์นี้ผ่านมาเมื่อไร แสดงว่าเป็นเฟรมข้อมูล

ฟิลด์ Destination Address และ Source Address
ขนาด 6 ไบต์
จะเก็บหมายเลขที่อยู่ทางฟิสิคัลของเครื่องปลายทางและเครื่องต้นทางของเฟรมข้อมูลนั้น บางครั้งเรียกว่า **หมายเลขที่อยู่อีเทอร์เนต** (Ethernet Address)

ฟิลด์ Length
ขนาด 2 ไบต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าหากค่าที่อยู่ในฟิลด์นี้น้อยกว่า 5EEh ซึ่งหมายความว่า เฟรมอีเทอร์เนตนี้ใช้ **มาตรฐาน IEEE 802.3** ค่าที่อยู่ในฟิลด์นี้จะแสดงความยาวของฟิลด์ Data

แต่ถ้าหากมีค่ามากกว่า 5EEh ซึ่งหมายความว่า เฟรมอีเทอร์เนตนี้ใช้ **มาตรฐานอีเทอร์เนตปกติ** ค่าในฟิลด์นี้เป็นชนิด (Type) ของเฟรม จะไม่แสดงความยาวของฟิลด์ Data อีกต่อไป

ฟิลด์ Data

ขนาด 46 - 1500 ไบต์

เก็บข้อมูลที่จะถูกส่งผ่านขึ้นไป ให้กับการทำงานในระดับชั้นบน

ฟิลด์ FCS

เพื่อตรวจสอบความถูกต้องของข้อมูลที่รับเข้ามา

ข้อควรระวัง *การแสดงผลข้อมูลที่ประเภทเวิร์ด (Word) ในเครื่องคอมพิวเตอร์ที่ใช้ CPU ของ Intel จะสลับตำแหน่งไบต์สูงกับไบต์ล่างกัน อาทิเช่น เฟรมประเภท DOD IP ซึ่งจะต้องมีหมายเลข 0800h แต่เมื่อแสดงผลจะเป็น 0080h*

หมายเหตุ *หมายเลขที่อยู่อีเทอร์เนตนี้ จะประกอบด้วยข้อมูล 2 กลุ่มโดย 3 ไบต์แรกเพื่อใช้แสดงหมายเลขผู้ผลิตเนตเวอร์กการ์ดนั้นๆ (ศึกษาข้อมูลเพิ่มเติมได้จากภาคผนวก) ส่วน 3 ไบต์หลังผู้ผลิตการ์ดจะเป็นผู้กำหนด โดยจะไม่ให้มีหมายเลขซ้ำซ้อนกัน*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 การทดลองและผลการทดลอง

4.1 ตัวอย่างเฟรมข้อมูลที่ได้รับจากแพคเก็ตไดรเวอร์

ในที่นี้จะขอยกตัวอย่างเฟรมข้อมูลที่ไดรับจากแพคเก็ตไดรเวอร์เพียง 2 เฟรมเท่านั้น เฟรมที่หนึ่งจะเป็นเฟรมข้อมูลในชุดของอินเตอร์เนตโพรโตคอล ส่วนในเฟรมที่สองจะเป็นเฟรมข้อมูลในชุดของเนตแวร์โพรโตคอล

1. เฟรมข้อมูลในชุดอินเตอร์เนตโพรโตคอล (รูปที่ 4.1)

เป็นเฟรมข้อมูลที่เกิดจากคำสั่ง LS

ในระดับชั้นดาตาลิงค์ ใช้เฟรมตามมาตรฐานอีเทอร์เนต

สังเกตได้จากฟิลด์ type (length) มีค่าเท่ากับ 0800H (มากกว่า 05EEH)

ในระดับชั้นเนตเวอร์ก ใช้โพรโตคอล IP

สังเกตได้จากฟิลด์ type มีค่าเท่ากับ 0800H

ในระดับชั้นทรานสปอร์ต ใช้โพรโตคอล TCP

สังเกตได้จากฟิลด์ protocol มีค่าเท่ากับ 6H

ในระดับชั้นที่สูงขึ้นไป ใช้โพรโตคอล Telnet

สังเกตได้จากฟิลด์ Destination Port มีค่าเท่ากับ 17H

2. เฟรมข้อมูลในชุดเนตแวร์โพรโตคอล (รูปที่ 4.2)

เป็นเฟรมข้อมูลที่เกิดจากการใช้คำสั่ง DIR

ในระดับชั้นดาตาลิงค์ ใช้เฟรมตามมาตรฐาน IEEE 802.3

สังเกตได้จากฟิลด์ type (length) มีค่าเท่ากับ 002A (น้อยกว่า 05EEH)

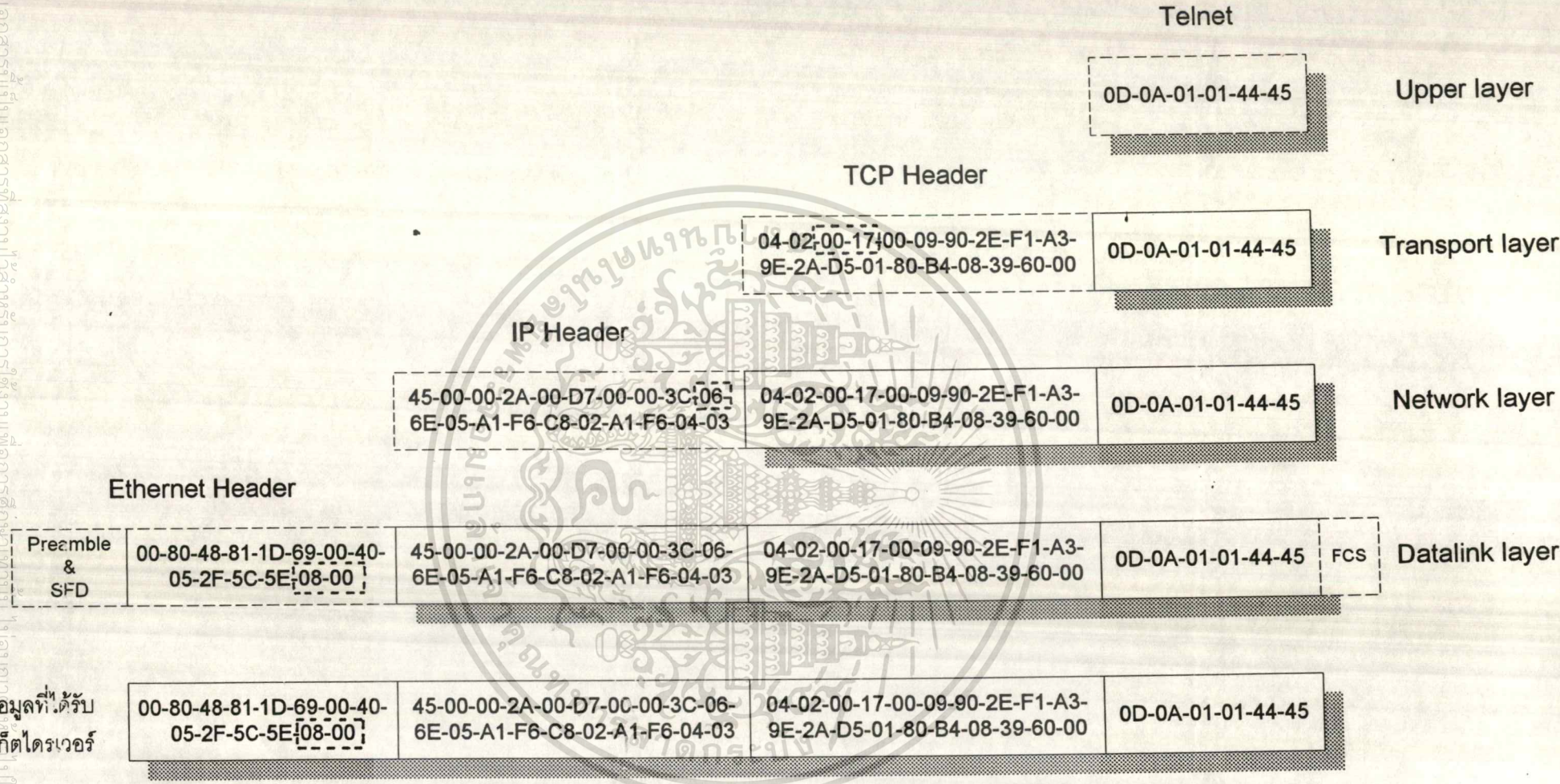
ในระดับชั้นเนตเวอร์ก ใช้โพรโตคอล IPX

เป็นข้อกำหนดของเนตแวร์ ซึ่งทุกแพคเก็ตจะต้องใช้โพรโตคอล IPX (รูปที่ 2.18)

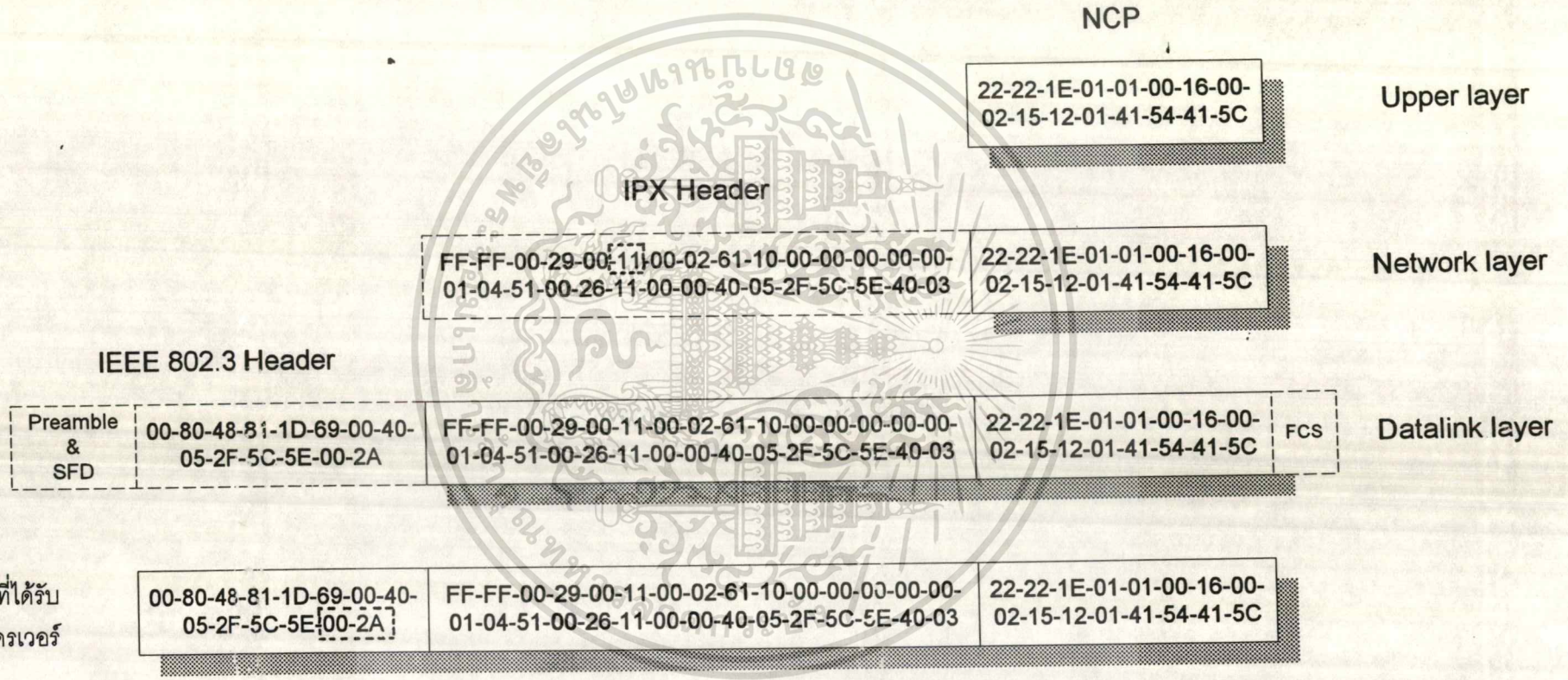
ในระดับชั้นที่สูงขึ้นไป ใช้โพรโตคอล NCP

สังเกตได้จากฟิลด์ Packet Type มีค่าเท่ากับ 11H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.1 แสดงข้อมูลที่รับได้ จากการใช้คำสั่ง ls
ซึ่งใช้ชุดอินเตอร์เนตโพรโตคอล



รูปที่ 4.2 แสดงข้อมูลที่รับได้ จากการใช้คำสั่ง dir
ซึ่งใช้ชุดเนตแวร์โพรโตคอล

4.2 การนำเฟรมข้อมูลที่ได้ จากแพคเก็ตไดรเวอร์มาใช้ประโยชน์

เนื่องจากข้อมูลที่ได้รับจากแพคเก็ตไดรเวอร์ มาจากตัวฮาร์ดแวร์โดยตรง จึงทำให้ได้ข้อมูลที่สมบูรณ์ ประกอบด้วย ข้อมูลที่ทำงานอยู่ในระดับชั้นบนอย่างครบถ้วน

ถ้าหากจะอธิบายเปรียบเทียบกับรูปแบบของ OSI แล้ว ข้อมูลที่ได้รับจะอยู่ในระดับชั้นดาตาลิงค์ ข้อมูลในระดับชั้นที่อยู่สูงขึ้นไป ตั้งแต่ชั้นเนตเวอร์กจนถึงแอพพลิเคชันจะถูก**ห่อรวมอยู่** (Capsulation) ซึ่งเราสามารถนำมาประโยชน์ได้ดังนี้

ตรวจสอบความเร็วในการรับส่งข้อมูล

ความเร็วในการรับส่งข้อมูล นับว่าเป็นหัวใจของระบบเครือข่าย เราสามารถใช้แพคเก็ตไดรเวอร์วัดความเร็ว ที่ใช้ในการติดต่อกันของเครื่องลูกข่ายต่างๆ โดยให้แพคเก็ตไดรเวอร์คอยตรวจจับเฟรมที่มีการรับส่งกัน แต่ทุกครั้งที่ได้รับเฟรม จะบันทึก**เวลาในขณะนั้น** (TimeStamp) เก็บไว้ด้วย หลังจากนั้นจึงกลับมาตรวจสอบค่าเวลาที่บันทึกไว้ เพื่อตรวจดูที่หน่วงออกไปไปในการรับส่งเฟรม ถ้าหากว่าอยู่ในเกณฑ์ที่ไม่น่าพอใจ จะต้องดำเนินการแก้ไข

เราสามารถใช้ข้อมูลในฟิลด์หมายเลขที่อยู่ ทั้งที่เป็น หมายเลขที่อยู่แบบฟิสิคัล และ หมายเลขที่อยู่แบบลอจิคัล เพื่อตรวจสอบความเร็วในการรับส่งข้อมูล เฉพาะเครื่องที่เราสนใจ โดยอาจจะเป็นระหว่างตัวไคลเอนต์กับโฮสต์ หรือระหว่างโฮสต์กับโฮสต์ก็ได้

ช่วยในการแบ่งเครือข่ายออกเป็นเซกเมนต์ย่อยๆ ทำได้อย่างถูกต้อง

ในเครือข่ายขนาดใหญ่ที่ใช้มาตรฐานอีเทอร์เนตนั้น เราสามารถเพิ่มประสิทธิภาพในด้านความเร็วของการรับส่งข้อมูล ได้โดย **การกำหนดเซกเมนต์**

การกำหนดเซกเมนต์ ก็คือการแบ่งระบบเครือข่ายที่มีขนาดใหญ่ออกเป็นเครือข่ายย่อยๆ แยกตามความถี่ที่มีการติดต่อกัน ถ้าเครื่องใดมีการติดต่อรับส่งกันมาก จะถูกกำหนดให้รวมอยู่ในเซกเมนต์เดียวกัน และระหว่างเซกเมนต์จะเชื่อมต่อกันด้วย **บริดจ์** (Bridge) หรือ **ตัวค้นหาเส้นทาง** (Router) ตัวบริดจ์หรือตัวค้นหาเส้นทาง จะทำหน้าที่กันข้อมูลที่เป็นการติดต่อภายในเซกเมนต์ของตนเองไม่ให้ผ่านออกไปรบกวนการทำงานของเซกเมนต์อื่น ทำให้ลดโอกาสที่จะเกิด **การชนกันของแพคเก็ต** (Collision) ลง ความเร็วของการรับส่งข้อมูลก็จะเพิ่มขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลที่ได้จากแพคเก็ตไดรเวอร์จะเก็บ **หมายเลขที่อยู่** ไว้ซึ่งมีทั้งที่เป็นแบบฟิสิคัล และที่เป็นแบบลอจิคัล เราสามารถเก็บรวบรวมเป็นสถิติ แล้วนำมาแจกแจงความถี่ตามความถี่ของการติดต่อกัน เพื่อดูว่า ในระบบเครือข่าย มีการรับส่งข้อมูลระหว่างกันเป็นเช่นไร คู่ไหนมาก คู่ไหนน้อย เพื่อใช้ประกอบการตัดสินใจในการจัดแบ่งเซกเมนต์

ช่วยให้การตัดสินใจ เลือกพัฒนาระบบเป็นไปอย่างถูกต้อง

เราสามารถนำแพคเก็ตไดรเวอร์มาช่วยประกอบการตัดสินใจ ในการเลือกลงทุน เพื่อพัฒนาระบบคอมพิวเตอร์ ที่มีใช้อยู่ในองค์กร โดยใช้หลักการที่ว่า ***หากระบบใดมีคนใช้มาก ก็น่าจะคุ้มค่าต่อการลงทุนพัฒนา*** อาทิเช่นถ้าระบบเครือข่ายของบริษัทมี 2 ระบบคือระบบที่ใช้ชุดเนตแวร์โพรโตคอล และระบบที่ใช้ชุดอินเตอร์เนตโพรโตคอล เราก็นำข้อมูลที่ได้จากเฟรมที่ได้รับมาจากแพคเก็ตไดรเวอร์มาจำแนกเป็นข้อมูลทางสถิติ ของการใช้โพรโตคอลทั้งสองชนิด ซึ่งจะทำให้มองเห็นภาพการใช้งานในระบบทั้งสอง ชัดเจนยิ่งขึ้น

เพิ่มประสิทธิภาพการทำงาน โดยใช้การจัดแบ่งเวลาการใช้งาน

ในเฟรมที่เราได้รับมาจากแพคเก็ตไดรเวอร์ จะมีฟิลด์ซึ่งใช้เก็บค่าความยาวของเฟรมนั้นๆอยู่ หากเรานำค่าในฟิลด์ดังกล่าวมาเก็บสะสม และจำแนกออกตามช่วงเวลาต่างๆ เช่นทุกๆหนึ่งชั่วโมง หรือทุกๆครึ่งชั่วโมง จะทำให้ทราบถึง **การใช้งานระบบเครือข่าย** (Utilization) จำแนกตามช่วงเวลา ทราบว่าในวันหนึ่งๆ ช่วงเวลาไหนมีการใช้ระบบเครือข่ายสูง ช่วงเวลาไหนไม่ค่อยมีการใช้ระบบเครือข่าย หากช่วงเวลาไหนมีการใช้ระบบเครือข่ายสูง ประสิทธิภาพโดยรวมย่อมจะต้องลดลงอันเนื่องมาจากโอกาสที่จะ **เกิดการชนกันของแพคเก็ต** ก็จะมีสูงขึ้นด้วย

ข้อมูลที่ได้นั้นพอที่จะนำไปกำหนดเป็นนโยบายของหน่วยงานต่างๆได้ อาทิเช่นพยายามกระจายงานที่ต้องใช้ระบบเครือข่าย เฉลี่ยให้ได้ดุลมากที่สุด **ไม่มากระจุกกันในช่วงเวลาใดเวลาหนึ่ง** งานที่สามารถจะทำได้โดยไม่ต้องการความรีบเร่ง เช่นการสำรองข้อมูล ควรให้ย้ายไปทำช่วงตอนหลังเลิกงานแล้ว เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 โปรแกรม Ethernet Monitor

Ethernet Monitor เป็นโปรแกรมจำลองการทำงานของอุปกรณ์ LAN Analyzer มีความสามารถในการตรวจจับ วิเคราะห์ และจำลองแพคเก็ต ที่มีการใช้งานบนระบบเครือข่ายที่เป็นแบบ **อีเทอร์เนต** โดยจะเน้นรายละเอียดเฉพาะใน **ชุดอินเตอร์เนตโพรโตคอล** และ **ชุดเนตแวร์โพรโตคอล**

การตรวจจับแพคเก็ตนั้น จะตรวจจับได้เฉพาะแพคเก็ตที่มีการรับส่งผ่านบนเซกเมนต์ (Segment) นั้น สามารถกระทำได้ทั้งในแบบ **ออนไลน์** (On-line) ซึ่งจะนำแพคเก็ตที่มีการรับส่ง ในขณะนั้นๆมาแสดงผล หรือในแบบ **ออฟไลน์** (Off-line) ซึ่งจะจัดเก็บแพคเก็ตที่ได้ ลงในบัฟเฟอร์หรือแฟ้มข้อมูล ที่ได้เตรียมไว้แล้วก่อน โดยสามารถจะนำกลับมาแสดงผลได้ในภายหลัง การนำแพคเก็ตมาวิเคราะห์นั้น จะเป็นการพิจารณาในแง่ของชนิดโพรโตคอลที่มีการใช้งาน ความคับคั่งที่เกิดขึ้น (Traffic) ของสายสัญญาณในช่วงเวลาต่างๆ โดยจะนำเสนอในรูปแบบของกราฟแท่ง เพื่อให้ง่ายต่อการทำความเข้าใจ และยังมีความสามารถในการจำลองแพคเก็ต ส่งเข้าไปในระบบเครือข่าย เพื่อทดสอบความสามารถการรองรับการทำงาน ของระบบเครือข่าย กรณีที่หากว่า มีแพคเก็ตข้อมูลเพิ่มขึ้น

โปรแกรม Ethernet Monitor นี้จะทำงานร่วมกันกับโปรแกรม **แพคเก็ตไดรเวอร์** (Packet Driver) โดยจะต้องเรียกโปรแกรมแพคเก็ตไดรเวอร์ ให้ฝังตัวอยู่ในหน่วยความจำก่อน ซึ่งจะช่วยทำหน้าที่เกี่ยวกับการจัดการและการติดต่อกับเนตเวอร์กการ์ด ให้กับโปรแกรม Ethernet Monitor

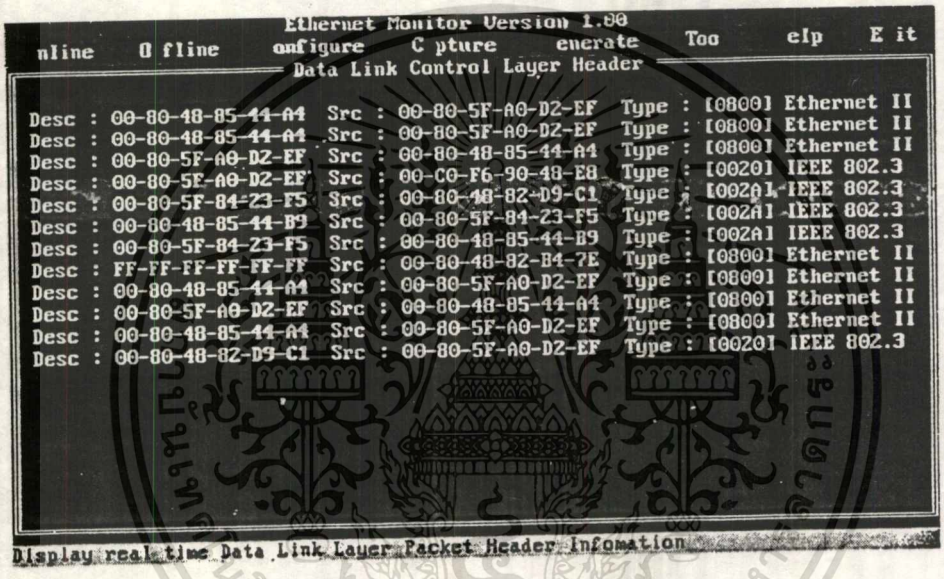
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.1 ความสามารถของโปรแกรม Ethernet Monitor

- **การตรวจจับแพคเก็ตในแบบออนไลน์** (On - line)

เป็นการตรวจจับแพคเก็ต ที่มีการรับส่ง ณ ขณะเวลานั้นๆ โดยสามารถแสดงผลได้ในหลายรูปแบบดังนี้

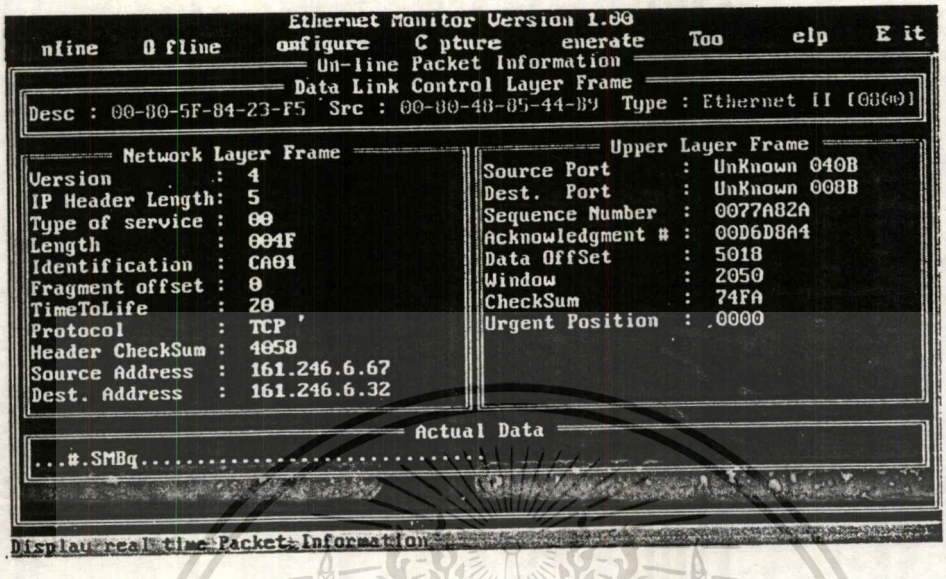
แสดงเฉพาะส่วนข้อมูลระดับชั้นดาตาลิงค์



แสดงผลเฉพาะในส่วน **เฮดเดอร์** (Header) ของข้อมูลที่อยู่ในระดับชั้น Datalink หรือก็คือเฮดเดอร์ของเฟรมชนิด **อีเทอร์เนต** และ IEEE 802.3 นั่นเอง ซึ่งจะประกอบด้วยฟิลด์หมายเลขที่อยู่ทางฟิสิคัลของเครื่องปลายทาง ต้นทาง และชนิดของเฟรมนั้นๆ ในกรณีที่เป็นเฟรมอีเทอร์เนต หรือถ้าหากเป็นเฟรมชนิด IEEE 802.3 ก็จะแสดงความยาวของเฟรมนั้นแทน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

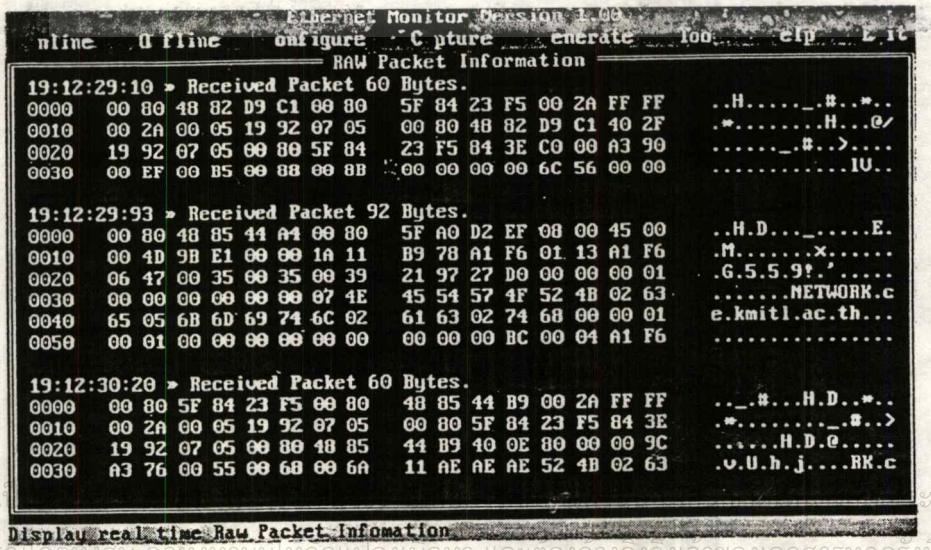
แสดง Packet Information



จะแสดงแพคเก็ตที่จับได้ในขณะเวลานั้นๆ จำแนกออกเป็นโพรโตคอล ในระดับชั้นต่างๆ ของชุดอินเตอร์เนตโพรโตคอล และชุดเนตแวร์โพรโตคอล

- ตารางทางด้านบน จะจำแนกชนิดของเฟรมข้อมูลในระดับชั้นดาตาลิงค์
- ตารางกลางในคอลัมน์ซ้าย จะเป็นรายละเอียดข้อมูลในส่วนที่เป็นเฮดเดอร์ ที่อยู่ในระดับชั้นเนตเวอร์ก
- ตารางกลางในคอลัมน์ขวา จะเป็นรายละเอียดข้อมูลในส่วนที่เป็นเฮดเดอร์ ที่อยู่ในระดับชั้นทรานสปอร์ตขึ้นไป จนถึงชั้นแอพพลิเคชัน
- ตารางทางด้านล่าง แสดงเฉพาะส่วนที่เป็นตัวข้อมูล

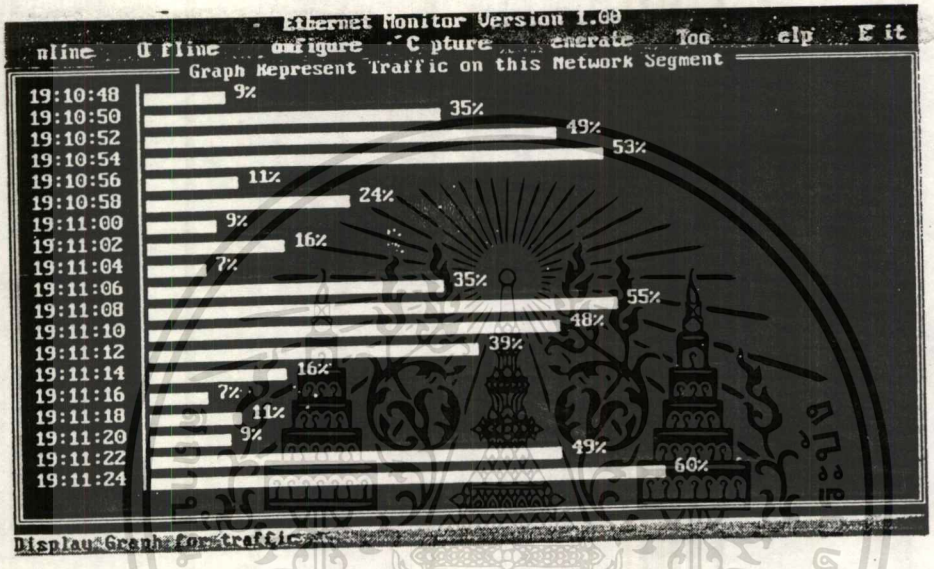
แสดงข้อมูลแบบ Raw Packet



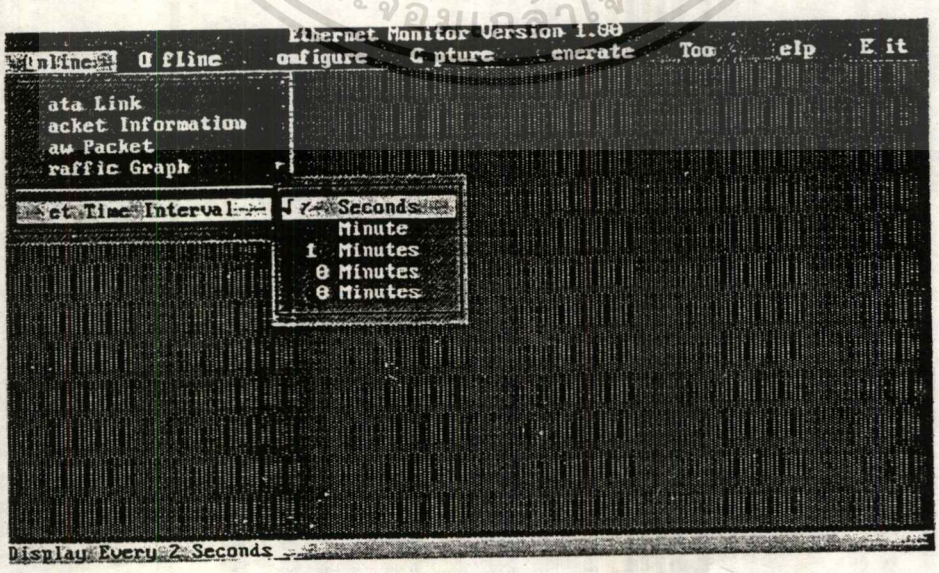
เอกสารนี้เ ะโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะแสดงเฟรมข้อมูลดิบ ในรูปแบบของ **เลขฐาน 16** ซึ่งยังไม่ได้นำไปแปลงเป็นความหมาย ข้อมูลในแต่ละฟิลด์ แต่ละบรรทัดจะแสดงข้อมูลความยาวเท่ากับ 16 ไบต์ ส่วนในคอลัมน์ทางขวา จะแสดงข้อมูลนั้นในรูปแบบของ **อักขระ (Alphabet)** และในแต่ละเฟรม จะมีการแสดงเวลา ขณะที่ได้รับเฟรมนั้น รวมถึงความยาวของ เฟรมนั้นๆด้วย

แสดง Traffic Graph



ซึ่งจะแสดงกราฟเปรียบเทียบความคับคั่ง (Traffic) ของสายสัญญาณในระบบเครือข่าย ในช่วงเวลาต่างๆ โดยที่ค่าช่วงห่างของเวลาที่จะแสดงผลในกราฟ สามารถกำหนดได้ตั้งแต่ 2 วินาที , 1 นาที , 15 นาที , 30 นาที และ 60 นาที



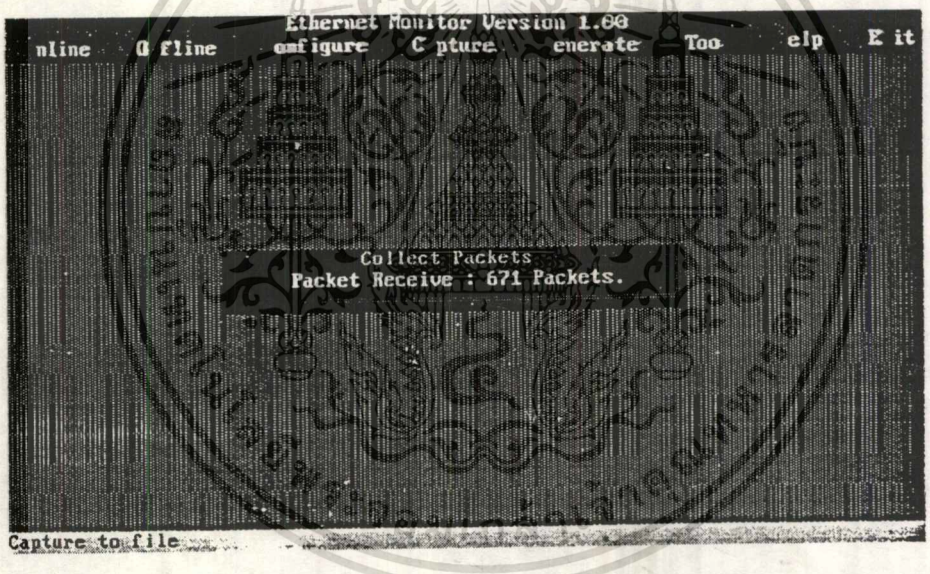
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หมายเหตุ *การตรวจจับแพคเก็ตแบบออนไลน์นี้ สามารถเลือกให้แสดงผลเฉพาะแพคเก็ต ที่มีรูปแบบตามที่กำหนดไว้ได้ (หัวข้อ การกำหนดชนิดของแพคเก็ต)*

- **การตรวจจับแพคเก็ตในแบบออฟไลน์** (Off - line)

จะเป็นการตรวจจับแพคเก็ต ที่มีการรับส่ง ณ ขณะเวลานั้น แต่แทนที่จะนำออกมาแสดงผลในทันที กลับเป็นการจัดเก็บแพคเก็ตที่ได้รับเหล่านั้น ลงในแฟ้มข้อมูล หรือในบัฟเฟอร์ที่ได้จัดเตรียมไว้แล้ว โดยสามารถที่จะกำหนดจำนวนแพคเก็ตที่ต้องการจะจัดเก็บได้ หรือไม่เช่นนั้นก็จะจัดเก็บจนกว่าจะเต็มขนาด ของเนื้อที่ที่ได้จัดเตรียมไว้

การจัดเก็บแพคเก็ตลงในแฟ้มข้อมูล



ซึ่งจะจัดเก็บแพคเก็ตที่ได้รับเหล่านั้น ลงในแฟ้มข้อมูลที่ชื่อ DATA.DAT และจะปรากฏข้อความแสดงจำนวนแพคเก็ตที่ได้รับการจัดเก็บไปแล้ว หากว่า ต้องการหยุดการจัดเก็บไว้ที่จำนวนตัวเลขขณะนั้น ให้กดคีย์ *ESC* แต่ถ้าหากไม่มีการกดคีย์ โปรแกรม Ethernet Monitor ก็จะจัดเก็บจนเต็มขนาดที่ได้จัดเตรียมไว้

หมายเหตุ *หากต้องการจะกำหนดเป็นชื่อแฟ้มข้อมูลอื่นที่ไม่ใช่ DATA.DAT ให้เลือก Save as แทนการเลือก Save ซึ่งจะปรากฏไดอะล็อกบล็อคเพื่อให้ใส่ชื่อแฟ้มข้อมูลใหม่*

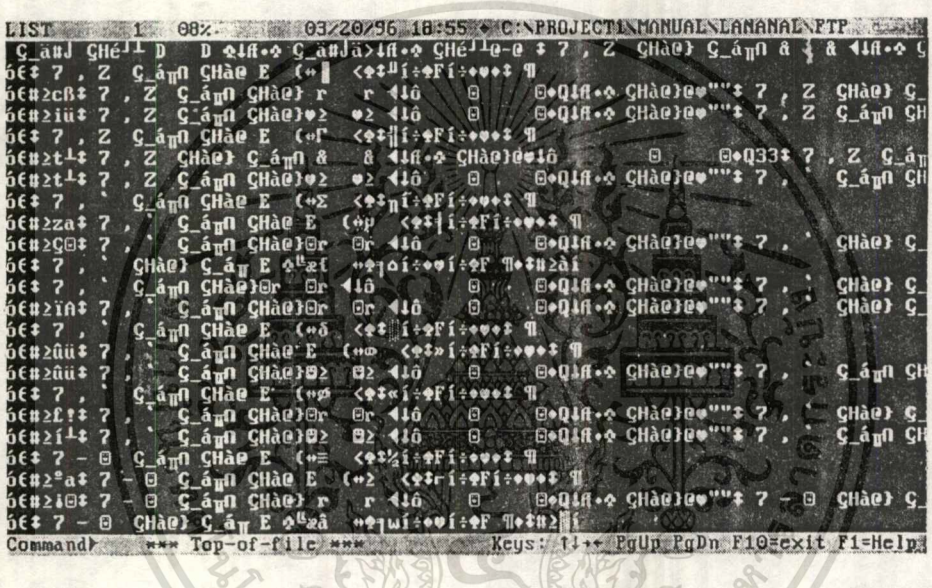
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจัดเก็บแพคเก็ตลงในบัฟเฟอร์

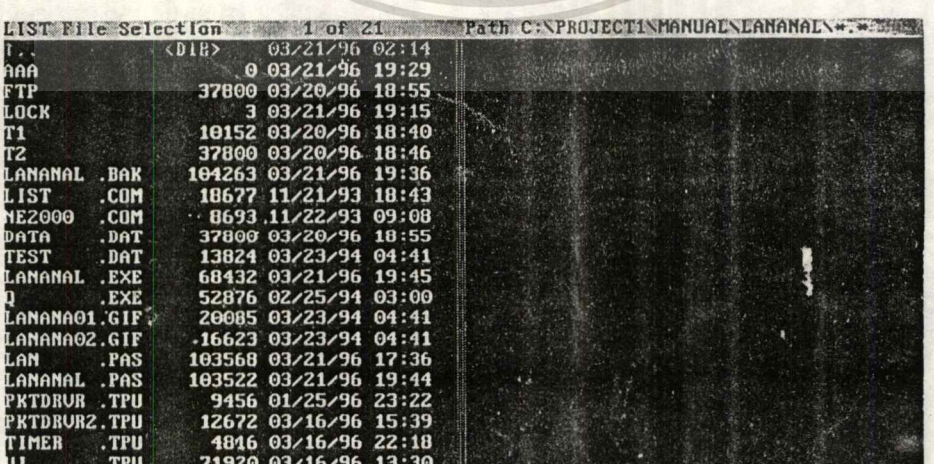
มีรูปแบบคล้ายคลึงกับการจัดเก็บแพคเก็ตลงในแฟ้มข้อมูล เพียงแต่ปลี่ยนจากแฟ้มข้อมูลซึ่งต้องมีการบันทึกลงบนดิสค์ มาเป็นบัฟเฟอร์หรือ หน่วยความจำส่วนหนึ่งที่ได้สำรองไว้แทน

ขณะที่มีการจัดเก็บแพคเก็ตลงในบัฟเฟอร์ จะมีข้อความแสดงจำนวนแพคเก็ตที่ได้มีการจัดเก็บไปแล้ว เราสามารถกำหนดจำนวนแพคเก็ตที่ต้องการจะจัดเก็บได้ โดยการกดคีย์ *ESC* ณ ขณะเวลาที่แสดงหมายเลขจำนวนแพคเก็ตนั้น ไม่เช่นนั้นแล้วจะจัดเก็บจนกว่าจะเต็มขนาดของบัฟเฟอร์

การตรวจสอบข้อมูลที่อยู่ในแฟ้มข้อมูล

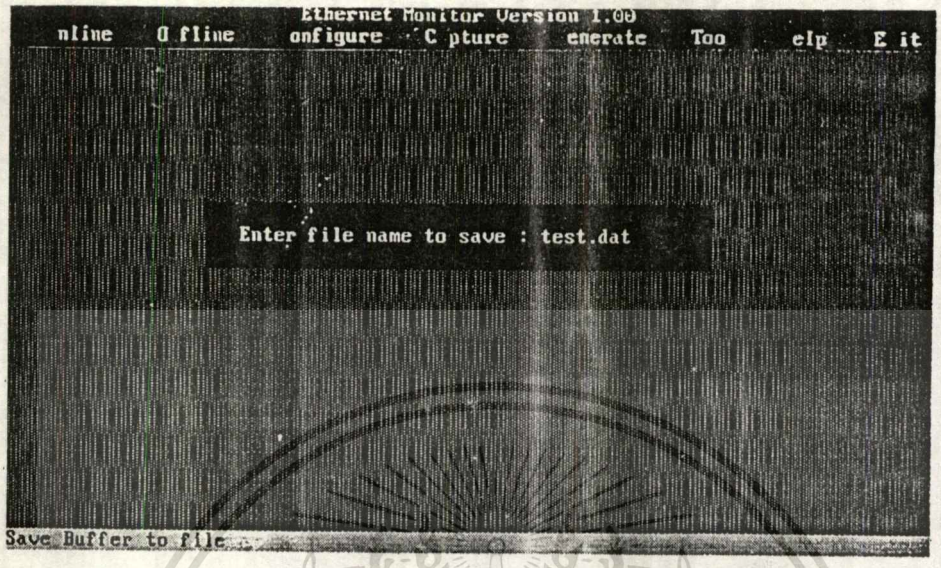


ซึ่งจะแสดงข้อมูลที่อยู่ในแฟ้มข้อมูลที่มีการทำงานครั้งล่าสุด แต่ถ้าหากว่าในขณะนั้น ยังไม่มีการทำงานใดๆกับแฟ้มข้อมูล ก็จะแสดงชื่อไฟล์ต่างๆ ที่อยู่ในไดเรกทอรีนั้นออกมาแทน



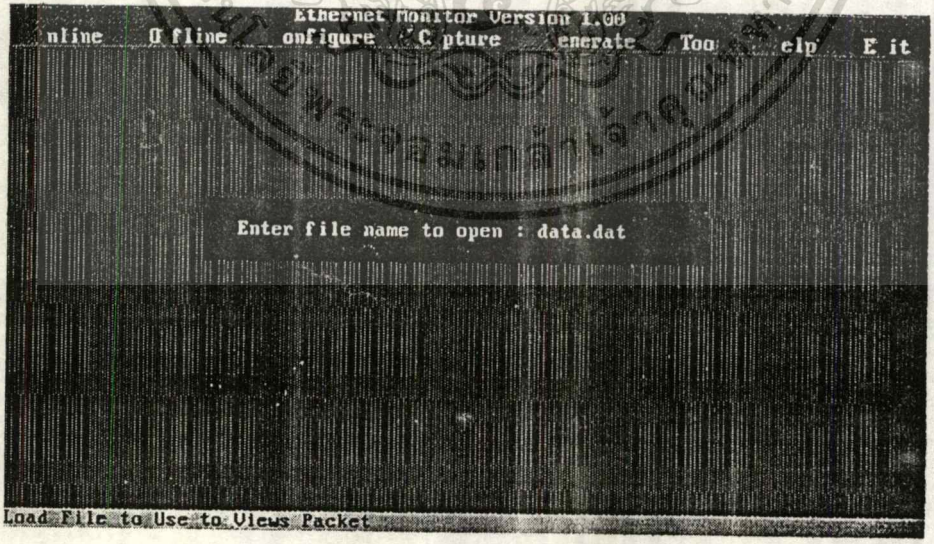
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำสำเนาข้อมูลแพคเก็ตจากบัฟเฟอร์ ลงในแฟ้มข้อมูล



ทั้งนี้เพื่อให้สามารถรวบรวม และจัดทำข้อมูลของการใช้งานแพคเก็ตบนระบบเครือข่าย ในรูปแบบของสถิติ นำมาใช้อ้างอิงในโอกาสที่เหมาะสมได้ต่อไป

การทำสำเนาข้อมูลแพคเก็ตจากแฟ้มข้อมูลลงในบัฟเฟอร์

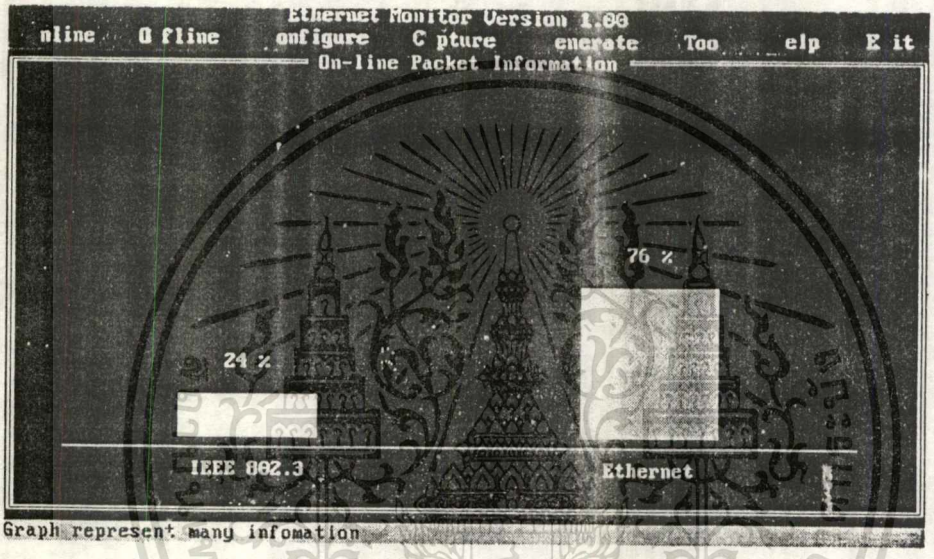


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การวิเคราะห์ข้อมูลแพคเก็ต

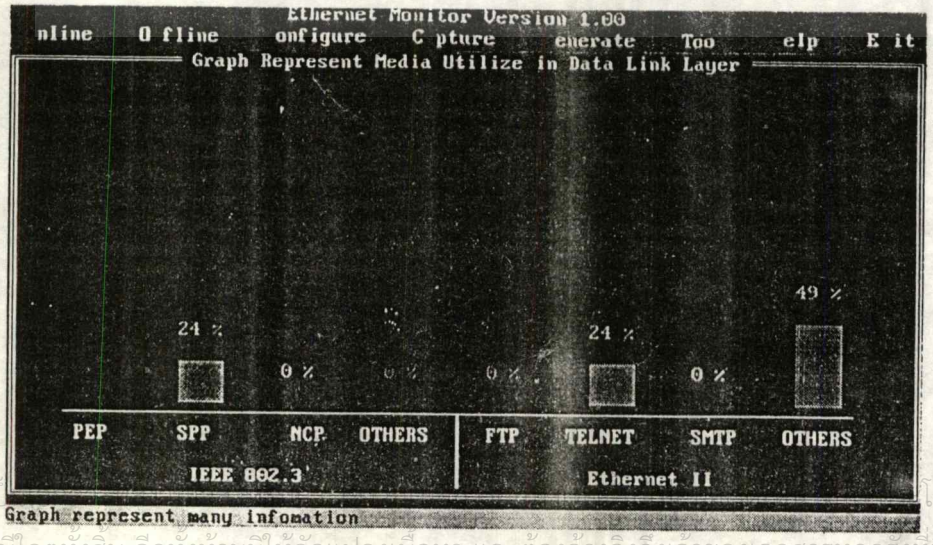
เป็นการนำ **ข้อมูลแพคเก็ตที่อยู่ในบัฟเฟอร์** มาวิเคราะห์ในรูปแบบของ กราฟแท่ง เพื่อให้ง่ายต่อการทำความเข้าใจ โดยจะเป็นการพิจารณาในเรื่องของชนิดโพรโตคอลที่มีการใช้งาน ซึ่งสามารถนำมาแสดงได้ใน 2 ระดับชั้นคือ

การวิเคราะห์ข้อมูลจำนวนแพคเก็ตในระดับชั้นดาตาลิงค์



จะเป็นการแสดงกราฟเปรียบเทียบจำนวนการใช้ เฟรมระหว่างชนิด IEEE 802.3 และชนิดมาตรฐาน Ethernet

การวิเคราะห์ข้อมูลจำนวนแพคเก็ตในระดับชั้นเนตเวอร์ก จนถึงชั้นแอพพลิเคชัน



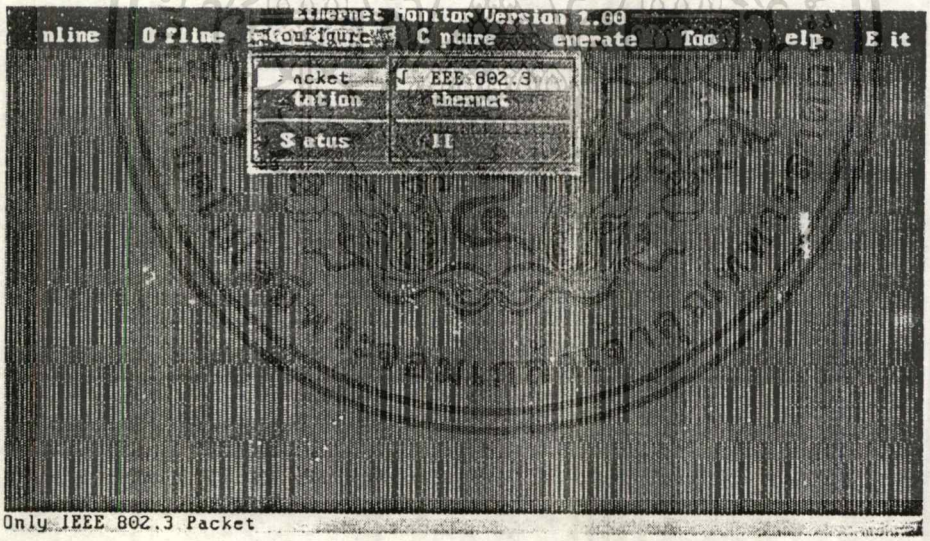
เอกสารนี้เ โยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะเป็นกราฟเปรียบเทียบระหว่างการใช้งาน **ชุดเนตแวดร์โพรโตคอล** ซึ่งใช้เฟรม IEEE 802.3 ประกอบด้วยโพรโตคอล PEP (Packet Exchange Packet), SPP (Sequenced Packet Protocol) และ NCP (Netware Core Protocol) กับการใช้งาน **ชุดอินเตอร์เนตโพรโตคอล** ซึ่งใช้เฟรม **Ethernet II** ประกอบด้วยโพรโตคอล FTP (File Transfer Protocol) , TELNET (Termianl Emulator) และ SMTP (Simple Mail Transfer Protocol) หากว่าแพคเก็ตไหนไม่สามารถจะจำแนกได้ จะถูกรวมไว้ใน OTHERS

- **การกำหนดชนิดของแพคเก็ต**

ในการ **ตรวจจับแพคเก็ตแบบออนไลน์** สามารถที่จะเลือกให้แสดงผลเฉพาะแพคเก็ตที่ต้องการได้ โดยข้อมูลที่จะนำมาเลือกนี้ จะใช้ประกอบกัน 2 ส่วนคือ ข้อมูลในส่วนที่เป็น **ชนิดของเฟรมในระดับชั้นดาตาลิงค์** และ **หมายเลขที่อยู่ทางฟิสิคัลของเครื่องต้นทาง และเครื่องปลายทาง**

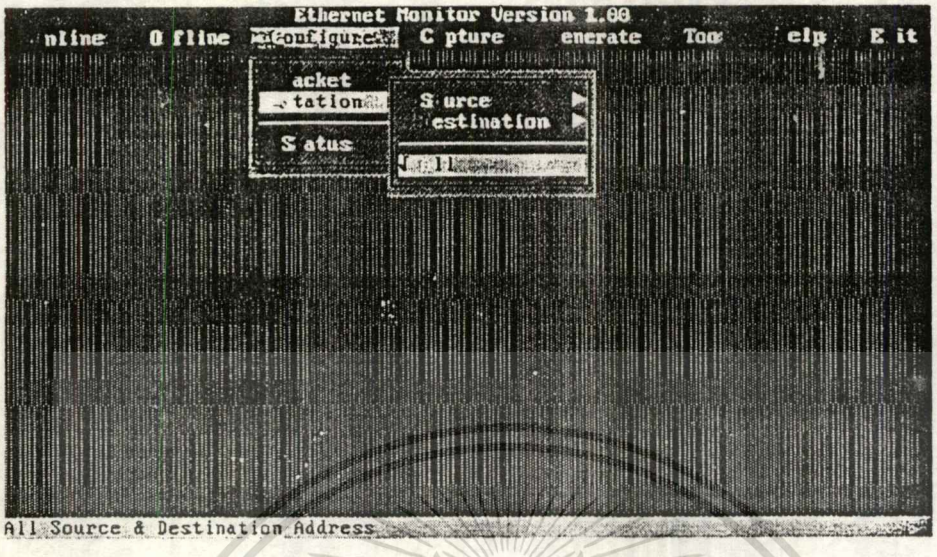
การกำหนดชนิดของเฟรมในระดับชั้นดาตาลิงค์



สามารถเลือกแพคเก็ตที่จะนำมาแสดงผลได้ระหว่างแพคเก็ตที่ใช้ **เฟรม** IEEE 802.3 และ Ethernet หรือว่าจะเลือก **ทั้งสองชนิด** (All)

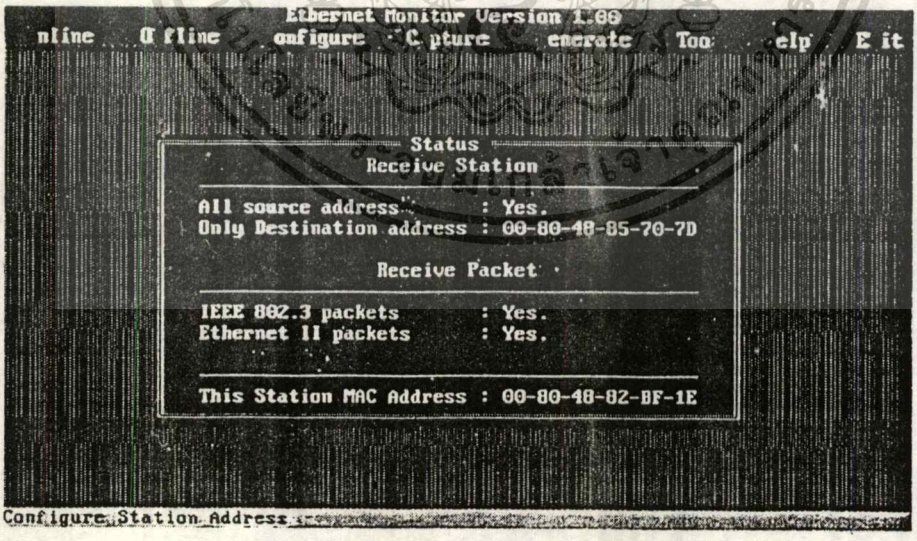
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การกำหนดหมายเลขที่อยู่ทางฟิสิคัล ของเครื่องต้นทางและเครื่องปลายทาง



สามารถกำหนดค่าหมายเลขที่อยู่ ของแพคเก็ตที่ต้องการจะนำมาแสดงผล ทั้งที่เป็นหมายเลขที่อยู่ของ **เครื่องต้นทาง** และ **เครื่องปลายทาง** อีกทั้งยังสามารถเลือก All ได้ หากไม่ต้องการที่จะกำหนดค่าใดๆ ซึ่งจะทำให้รับได้ทุกๆแพคเก็ต

การแสดงสถานะของค่าที่กำหนดในขณะนั้น



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลในส่วน 2 บรรทัดแรกจะเป็น ค่าที่กำหนดหมายเลขที่อยู่ทางฟิสิคัล ในอีก 2 บรรทัดถัดมา จะแสดงชนิดของเฟรม ส่วนในบรรทัดสุดท้าย จะเป็นหมายเลขที่อยู่ทางฟิสิคัล ของเครื่องที่ทำงานโปรแกรม Ethernet Monitor นี้อยู่

- **การจำลองแพคเก็ต ส่งเข้าไปในระบบเครือข่าย**

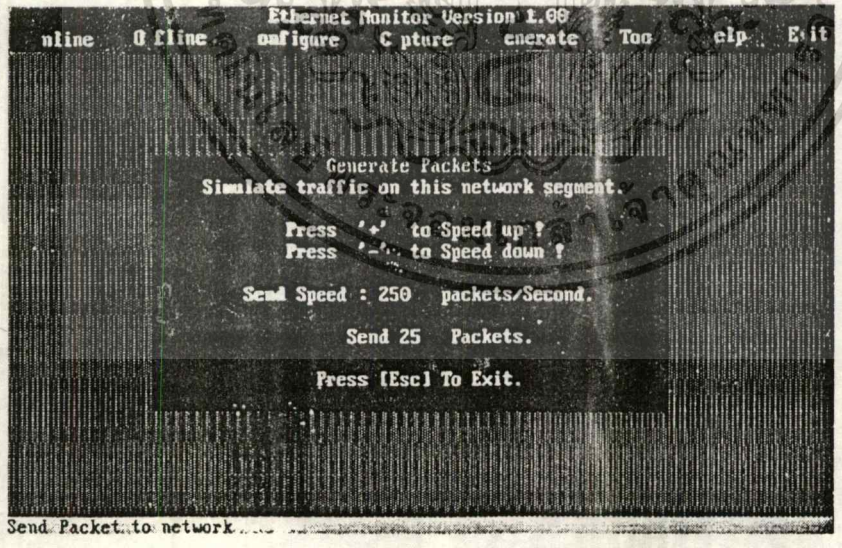
การจำลองแพคเก็ต ส่งเข้าไปในระบบเครือข่ายนี้ มีวัตถุประสงค์เพื่อตรวจสอบประสิทธิภาพในการทำงานของระบบเครือข่าย ว่าจะลดลงเพียงใด กรณีที่เกิด **ความคับคั่ง** (Traffic) ของแพคเก็ตข้อมูลเพิ่มขึ้น

โดยแพคเก็ตข้อมูลที่จะส่งนี้ จะอยู่ในรูปแบบของเฟรมประเภท IEEE 802.3 โดยที่

- ฟิลด์ Source Address จะเป็นหมายเลขที่อยู่ทางฟิสิคัล ของเครื่องที่ทำการส่งแพคเก็ต
- ฟิลด์ Destination Address จะเป็นหมายเลขที่อยู่ทางฟิสิคัล ของเครื่องปลายทาง
- ฟิลด์ Length จะเก็บค่าความยาวของเฟรม ในที่นี้จะเป็น 0x005A

โดยมีเนื้อข้อมูลดังนี้

" This is a test message generate from Ethernet Monitor Version 1.00 Revision A !!!!! "

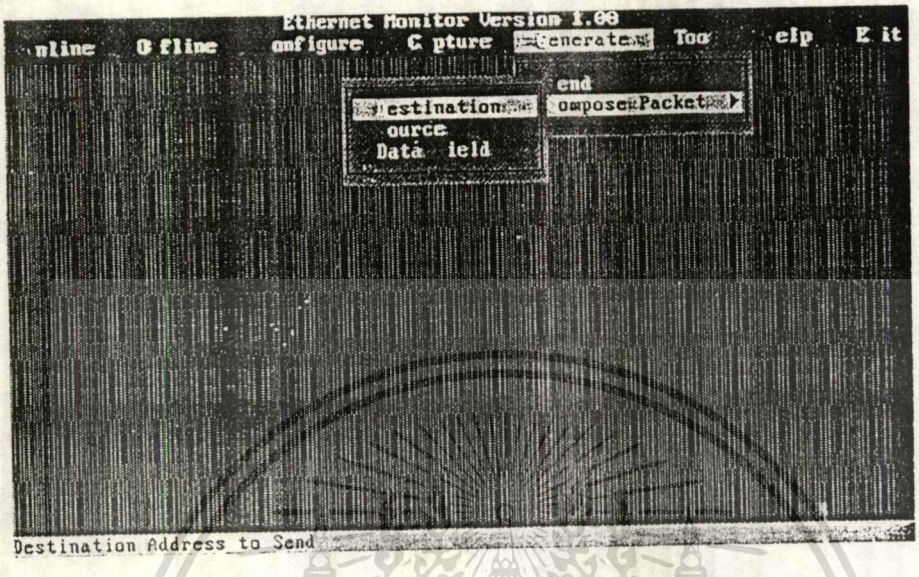


สามารถที่จะกำหนดอัตราเร็วในการส่งแพคเก็ตได้ตั้งแต่ 1 จนถึง 1000 แพคเก็ตต่อวินาที โดยการใช้คีย์ ' + ' เพื่อเพิ่มอัตราเร็ว และใช้คีย์ ' - ' เพื่อลดอัตราเร็ว

ในบรรทัดล่าง แสดงจำนวนแพคเก็ตทั้งหมดที่ได้ส่งออกไป ในระบบเครือข่ายแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

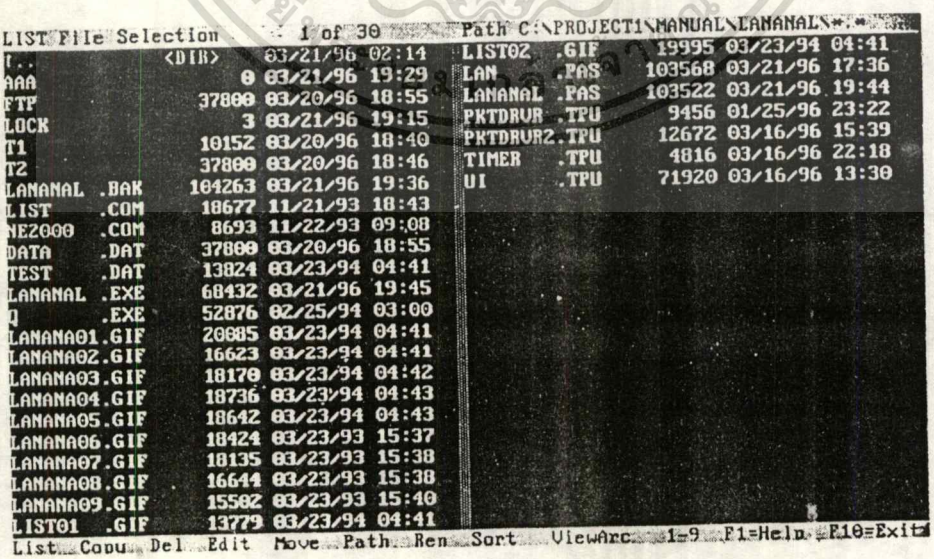
และยังสามารถแก้ไข ข้อมูลที่อยู่ในฟิลด์ต่างๆ (Destination, Source Address & Data) ของเฟรม ได้ด้วย



- เครื่องมือเสริม

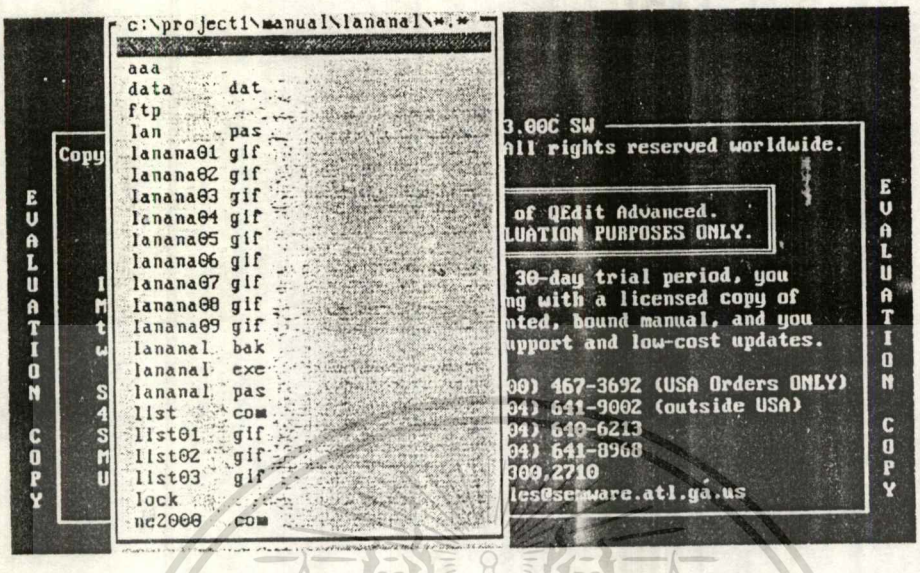
ในโปรแกรม Ethernet Monitor ได้รวมเครื่องมือเสริมที่จะช่วยในการเปิดดู หรือทำการแก้ไขแฟ้มข้อมูลแบบตัวอักษร (Text File) ไว้ด้วย

โดยหากต้องการเพียงจะเปิดดูแฟ้มข้อมูลเท่านั้น ให้เรียกโปรแกรม List ขึ้นมาทำงาน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แต่ถ้าหากว่าต้องการจะทำการแก้ไขข้อมูลด้วย ให้เรียกโปรแกรม Qedit ขึ้นมาทำงาน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.2 ข้อกำหนดของโปรแกรม Ethernet Monitor

1 สามารถตรวจจับแพคเก็ต เฉพาะที่ส่งผ่านบนเซกเมนต์ ซึ่งเครื่อง Ethernet Monitor ทำงานอยู่ ไม่สามารถตรวจจับแพคเก็ตที่อยู่ในเซกเมนต์อื่นได้

2 เนื่องจาก Ethernet Monitor จะต้องใช้งานร่วมกับแพคเก็ตไดรเวอร์ แต่ตัวแพคเก็ตไดรเวอร์นั้นจะแยกออกเป็นของแต่ละผู้ผลิตเนตเวอร์กการ์ด เช่นแพคเก็ตไดรเวอร์จำพวก 3CXXX จะใช้งานกับเนตเวอร์กการ์ดของบริษัท 3COM หากเป็น NE2000 ก็จะใช้งานกับเนตเวอร์กการ์ดที่เป็นแบบ NE2000 Compatible เพราะฉะนั้นจึงจำเป็นต้องหาแพคเก็ตไดรเวอร์ที่ตรงกันกับเนตเวอร์กการ์ดที่มีอยู่ (ตรวจสอบรายชื่อแพคเก็ตไดรเวอร์ที่มีอยู่ทั้งหมดได้ จากแฟ้มข้อมูลชื่อ MANIFEST.DOC ในแผ่นดิสก์ที่ได้แนบมาด้วย)

3 ระบบเครือข่ายที่จะสามารถใช้งาน Ethernet Monitor ได้คือเครือข่ายแบบ อีเทอร์เนต (IEEE 802.3) เท่านั้น ไม่สามารถใช้งานได้กับเครือข่ายที่เป็นแบบ โทเกนบัส หรือ โทเกนริง

4 จุดมุ่งหมายประการสำคัญของ Ethernet Monitor คือเพื่อใช้ประกอบการศึกษาระบบเครือข่ายคอมพิวเตอร์แบบอีเทอร์เนต โดยเน้นไปที่การทำงานของโพรโตคอล ทางผู้เขียนจึงไม่รับรองความแม่นยำ ในการทำงานของโปรแกรม แต่สามารถนำไปใช้งานเพื่อมองดูภาพรวม การใช้งานในระบบเครือข่ายที่ต้องการได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5 บทวิจารณ์และสรุป

จากการศึกษาและทำงานเกี่ยวกับระบบเครือข่ายคอมพิวเตอร์ ในแง่การพัฒนาและปรับปรุงระบบเครือข่ายของประเทศไทยนั้น ทำได้ค่อนข้างลำบาก เนื่องจากยังขาดแคลนบุคลากรที่มีความรู้ความสามารถทางด้านนี้เป็นอย่างมาก โดยหากมีผู้รู้ ก็จะจำกัดความรู้นั้นอยู่เพียงกลุ่มคนแคบๆ ไม่ได้มีการกระจายความรู้ออกไปเท่าที่ควร จึงทำให้เป็นอุปสรรคแก่ผู้เริ่มต้นทำการศึกษาหาความรู้ทางด้านนี้ เพราะนอกจากปัญหาที่หาผู้ที่มีความรู้ พอที่จะซักถามได้ยากแล้ว ยังมีปัจจัยอื่นๆอีกหลายอย่างเช่น แหล่งความรู้เพิ่มเติมทางด้านเอกสารหรือตำราที่เป็นภาษาไทยไม่ค่อยมี ทำให้ผู้เริ่มต้นไม่รู้ว่าควรจะเริ่มต้นจากจุดไหนดี

ดังนั้นในปริญญานิพนธ์ฉบับนี้จึงได้รวบรวม ความรู้และข้อมูลที่ควรทราบจากแหล่งข้อมูลต่างๆ เช่น จากตำราภาษาต่างประเทศ เอกสารแนะนำและคู่มือการใช้อุปกรณ์เครือข่าย ข้อมูลเผยแพร่จากระบบเครือข่ายอินเตอร์เนต เพื่อเป็นแนวทางในการศึกษาค้นคว้าแก่ผู้ที่สนใจต่อไป

ข้อเสนอแนะ

จากการศึกษาค้นคว้าทางด้านระบบเครือข่าย โดยใช้แพคเก็ตไดรเวอร์ เป็นเครื่องมือช่วยในการแสดงการทำงานของระบบเครือข่าย ทำให้สามารถเข้าใจการทำงานของระบบเครือข่ายในการรับส่งข้อมูลได้ดีขึ้น แต่จากการใช้งานจะพบว่ายังมีอุปสรรค ที่อาจจะทำให้การทำงานไม่เป็นไปตามที่ต้องการ จึงเสนอแนะแนวทางในการแก้ไขปัญหาดังนี้

1. เครื่องที่นำมาใช้งานต้องมีความเร็วสูง ข้อมูลที่ได้รับมาจะมีความสมบูรณ์ ถูกต้องแค่ไหนจะขึ้นอยู่กับความเร็วในการประมวลผล

2. อัลกอลิทึมที่ใช้ในการประมวลผล ต้องกระชับ คือจะต้องทำงานเฉพาะในส่วนที่จำเป็นเท่านั้น เพื่อช่วยลดเวลาในการประมวลผล

3. การตรวจสอบความถูกต้องของโปรแกรม ในการที่จะบอกในการที่จะบอกได้ว่าโปรแกรมทำงานได้ถูกต้องเพียงใด ต้องทำการทดสอบในระบบเครือข่ายที่มีลักษณะปิด กล่าวคือการส่งข้อมูลและปริมาณข้อมูลจะต้องควบคุมได้ และรู้ปริมาณของข้อมูล เพื่อจะนำมาใช้วิเคราะห์ความถูกต้องได้

4. การทำงานยังทำงานอยู่บนสิ่งแวดล้อมของดอส หากต้องการนำไปพัฒนาบนวินโดวส์ โปรแกรมแพคเก็ตไดรเวอร์สามารถเรียกใช้ออฟชัน -w แต่อาจจะไม่มีเสถียรภาพเท่าที่ควร ควรใช้ .winpkt แทนซึ่งพัฒนาสำหรับวินโดวส์โดยเฉพาะ

5. โปรแกรมแพคเก็ตไดรเวอร์ มีข้อจำกัดคือ ไม่สามารถรับหรือส่งข้อมูลข้ามเซกเมนต์ได้

6. จากการที่สามารถรับและส่งข้อมูลใดๆก็ได้ ทำให้เปรียบเสมือนดาบสองคม เพราะระบบเครือข่ายนั้น มีผู้อื่นใช้งานด้วย ทั้งขึ้นอยู่กับผู้ใช้จะนำไปใช้ในทางสร้างสรรค์ หรือทำลาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แนวทางในการนำไปใช้งานต่อไป

จากการทำงานของแพคเก็ตไดรเวอร์ ที่ควบคุมเนตเวอร์กการ์ดได้ถึงขั้นจัดการเฟรม ทำให้สามารถส่ง และรับข้อมูลได้ ทั้งๆที่ข้อมูลนั้นไม่เป็นของตนเอง ทำให้มองเห็นแนวทางในการพัฒนาต่อไปได้ดังนี้

1. ทำการตรวจสอบและป้องกัน ไม่ให้มีการติดต่อกับแหล่งข้อมูลหรือทำงานบางประเภทที่ไม่อนุญาต หากตรวจพบว่ามีความพยายามดังกล่าวแล้ว ให้ทำการสร้างข้อมูลขึ้นมารบกวน ไม่ให้ทำการติดต่อต่อไปได้ ตัวอย่างเช่นหากมีผู้พยายามจะโอนถ่ายข้อมูลที่เป็นภาพลามกอนาจาร จากแหล่งข้อมูลในเครือข่ายอินเตอร์เนต การโอนถ่ายข้อมูลนั้นหากทำงานในแบบที่เป็น Connection-Oriented จะต้องมีลำดับการรับส่งข้อมูลระหว่างเครื่องทั้งสอง การสร้างข้อมูลขึ้นมารบกวน ก็จะเป็นการไปแทรกลำดับให้เกิดความสับสน จนไม่สามารถติดต่อต่อไปได้

2. ใช้เพื่อตรวจสอบโพรโตคอลที่มีการใช้งานอยู่ภายในองค์กร ว่ามีความปลอดภัยเพียงพอหรือไม่ อาทิเช่นจากที่ได้ศึกษาการทำงานของชุดอินเตอร์เนตโพรโตคอล การรับส่งข้อมูลนั้นจะไม่มีการเข้ารหัสใดๆเลย จึงควรที่จะหาแนวทางการป้องกันและแก้ไขเพื่อเพิ่มความปลอดภัยต่อไป

3. ใช้เก็บรวบรวมข้อมูลการใช้งานระบบเครือข่ายภายในองค์กร และนำไปจัดทำเป็นสถิติในรูปแบบต่างๆ เพื่อหาแนวทางในการเพิ่มประสิทธิภาพของระบบเครือข่ายให้สูงขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

ตารางแสดง Ethernet DIX Type

Type	Description
0600	XNS
0601	XNS Address Translation
0800	DOD IP
0801	X.75 internet
0802	NBS internet
0803	ECMA internet
0804	Chaosnet
0805	X.25 Level 3
0806	ARP
0807	XNS Compatibility
081C	Symbolics private
0888	Xyplex
0889	Xyplex
088A	Xyplex
0900	Ungermann-Bass net debugr
0A00	Xerox PUP
0A01	Xerox PUP Address Translation
0BAD	Banyan Systems
0BAF	Banyan Echo
1000	Berkeley trailer negotiation
1001	Berkeley trailer encapsulation
1002	Berkeley trailer encapsulation
1003	Berkeley trailer encapsulation
1004	Berkeley trailer encapsulation
1005	Berkeley trailer encapsulation
1006	Berkeley trailer encapsulation
1007	Berkeley trailer encapsulation
1008	Berkeley trailer encapsulation
1009	Berkeley trailer encapsulation

Type	Description
100A	Berkeley trailer encapsulation
100B	Berkeley trailer encapsulation
100C	Berkeley trailer encapsulation
100D	Berkeley trailer encapsulation
100E	Berkeley trailer encapsulation
100F	Berkeley trailer encapsulation
1234	DCA - Multicast
1600	VALID system protocol
1989	Artificial Horizons (dogfight simulator)
3C00	3Com NBP
3C01	3Com NBP
3C02	3Com NBP
3C03	3Com NBP
3C04	3Com NBP
3C05	3Com NBP
3C06	3Com NBP
3C07	3Com NBP
3C08	3Com NBP
3C09	3Com NBP
3C0A	3Com NBP
3C0B	3Com NBP
3C0C	3Com NBP
3C0D	3Com NBP
4242	PCS Basic Block Protocol
4321	THD - Diddle
5208	BBN Simnet Private
6000	DNA experimental
6001	DNA Dump/Load -MOP-
6002	DNA Remote Console -MOP-

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Type	Description
6003	DNA IV Routing Layer
6004	DEC: Local Area Transport
6005	DEC: Diagnostics
6006	DEC: Customer Use
6007	DEC: LAVC
6008	DEC Unassigned (AMBER?)
6009	DEC Unassigned (MUMPS?)
6010	3Com
6011	3Com
6012	3Com
6013	3Com
6014	3Com
7000	Ungermann-Bass download
7001	Ungermann-Bass NIUs
7002	Ungermann-Bass diagnostic/loopback
7003	Ungermann-Bass
7005	Ungermann-Bass Bridge
7007	OS/9 Microware
7009	OS/9 Net ?
7020	Sintrom (was LRT)
7021	Sintrom (was LRT)
7022	Sintrom (was LRT)
7023	Sintrom (was LRT)
7024	Sintrom (was LRT)
7025	Sintrom (was LRT)
7026	Sintrom (was LRT)
7027	Sintrom (was LRT)
7028	Sintrom (was LRT)
7029	Sintrom (was LRT)
7030	Racal-Interlan
7031	Prime NTS
7034	Cabletron
8003	Cronus VLN

Type	Description
8004	Cronus Direct
8005	HP Probe
8006	Nestar
8008	AT&T/Standford
8010	Excelan
8013	Silicon Graphics diagnostic
8014	Silicon Graphics network games
8015	Silicon Graphics
8016	Silicon Graphics XNS Nameserver
8019	Apollo DOMAIN
802E	Tymshare
802F	Tigan
8035	Reverse ARP
8036	Aeonic Systems
8037	IPX (Netware)
8038	DEC: bridge
8039	DEC: DSM/DDP
803A	DEC: (Argonaut console ?)
803B	DEC: (VAXELN ?)
803C	DEC: (NMSV? DNA Naming?)
803D	DEC: encryption
803E	DEC: distributed time service
803F	DEC: LAN Traffic Monitor
8040	DEC: NetBIOS Datagrams
8041	DEC: Local Area System Transport
8042	DEC Unassigned
8044	Planning Research Corp.
8046	AT&T
8047	AT&T
8048	DEC: DECamds
8049	ExperData
805B	VMTP/RFC-1045
805O	Stanford V-Kernel, version 6.0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Type	Description
805D	Evans & Sutherland
8060	Little Machine
8062	Counterpoint Computers
8065	University of Mass. at Amherst
8066	University of Mass. at Amherst
8067	Veeco Integrated Automation
8068	General Dynamics
8069	AT&T
806A	Autophon
806C	ComDesign
806D	Compugraphic Corp.
806E	Landmark Graphics Corp.
806F	Landmark Graphics Corp.
8070	Landmark Graphics Corp.
8071	Landmark Graphics Corp.
8072	Landmark Graphics Corp.
8073	Landmark Graphics Corp.
8074	Landmark Graphics Corp.
8075	Landmark Graphics Corp.
8076	Landmark Graphics Corp.
8077	Landmark Graphics Corp.
807A	Matra
807B	Dansk Data Elektronic
807C	Merit Internodal
807D	Vitalink Communications
807E	Vitalink Communications
807F	Vitalink Communications
8080	Vitalink TransLAN III Mgmt
8081	Counterpoint Computers
8082	Counterpoint Computers
8083	Counterpoint Computers
8088	Xyplex
8089	Xyplex

Type	Description
808A	Xyplex
809B	EtherTalk (AppleTalk)
809C	Datability
809D	Datability
809E	Datability
809F	Spider Systems
80A3	Nixdorf
80A4	Siemens Gammasonics
80A5	Siemens Gammasonics
80A6	Siemens Gammasonics
80A7	Siemens Gammasonics
80A8	Siemens Gammasonics
80A9	Siemens Gammasonics
80AA	Siemens Gammasonics
80AB	Siemens Gammasonics
80AC	Siemens Gammasonics
80AD	Siemens Gammasonics
80AE	Siemens Gammasonics
80AF	Siemens Gammasonics
80B0	Siemens Gammasonics
80B1	Siemens Gammasonics
80B2	Siemens Gammasonics
80B3	Siemens Gammasonics
80C0	DCA: Data Exchange Cluster
80C1	DCA: Data Exchange Cluster
80C2	DCA: Data Exchange Cluster
80C3	DCA: Data Exchange Cluster
80C6	Pacer Software
80C7	Appplitek Corp.
80C8	Intergraph Corp.
80C9	Intergraph Corp.
80CA	Intergraph Corp.
80CB	Intergraph Corp.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Type	Description
80CC	Intergraph Corp.
80CD	Harris Corporation
80CE	Harris Corporation
80CF	Taylor Instrument
80D0	Taylor Instrument
80D1	Taylor Instrument
80D2	Taylor Instrument
80D3	Rosemount Corp.
80D4	Rosemount Corp.
80D5	IBM SNA Service on Ethernet
80DD	Varian Associates
80DE	TRFS (Integrated Solutions ...)
80DF	TRFS (Integrated Solutions ...)
80E0	Allen-Bradley
80E1	Allen-Bradley
80E2	Allen-Bradley
80E3	Allen-Bradley
80E4	Datability
80E5	Datability
80E6	Datability
80E7	Datability
80E8	Datability
80E9	Datability
80EA	Datability
80EB	Datability
80EC	Datability
80ED	Datability
80EE	Datability
80EF	Datability
80F0	Datability
80F2	Retix
80F3	AppleTalk AARP
80F4	Kinetics
80F5	Kinetics
80F7	Apollo Computers
80FF	Wellfleet
8100	Wellfleet
8101	Wellfleet
8102	Wellfleet
8103	Wellfleet
8107	Symbolics
8108	Symbolics
8109	Symbolics
812B	Talaris
8130	Waterloo Microsystems
8131	VG Laboratory Systems
8137	Novell NetWare
8138	Novell ?
8139	KTI
813A	KTI
813B	KTI
813C	KTI
813D	KTI
814C	SNMP over Ethernet
814F	Technically Elite Concepts
817D	XTP
81D6	Lantastic
8582	Kalpana
8888	HP LanProbe ?
9000	Loopback
9001	3Com: XNS Mngmt
9002	3Com: TCP/IP Mngmt
9003	3Com: loopback detection
AAAA	DECNET ???
FF00	BBN VITAL-LanBridge

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

ตารางแสดง IEEE MAC Address ของผู้ผลิต

ค่า 3 ไบต์แรกของ Ethernet MAC Address

MAC	Vendor
00-00-02	BBN
00-00-0C	Cisco
00-00-0E	Fujitsu
00-00-0F	NeXT
00-00-10	Sytek/Hughes LAN Systems
00-00-11	Tektronics
00-00-15	Datapoint
00-00-18	Webster
00-00-1A	AMD ?
00-00-1B	Novell/Eagle Technology
00-00-1D	Cabletron
00-00-20	Data Industrier AB
00-00-21	SC&C
00-00-22	Visual Technology
00-00-23	ABB
00-00-29	IMC
00-00-2A	TRW
00-00-3C	Auspex
00-00-3D	ATT
00-00-44	Castelle
00-00-46	Bunker Ramo
00-00-49	Apricot
00-00-4B	APT
00-00-4F	Logicraft
00-00-51	Hob Electronic
00-00-52	ODS
00-00-55	AT&T
00-00-5A	SK/Xerox
00-00-5D	RCF
00-00-5E	IANA
00-00-61	Gateway
00-00-62	Honeywell
00-00-65	Network General
00-00-69	Silicon Graphics
00-00-6B	MIPS
00-00-6E	Artisoft
00-00-77	MIPS/Interphase
00-00-78	Labtam
00-00-7A	Ardent
00-00-7B	Research Machines
00-00-7D	Cray Research/Harris
00-00-7F	Linotronic
00-00-80	Dowty Network Services
00-00-81	Synoptics
00-00-84	Aquila ?
00-00-86	Gateway
00-00-89	Cayman Systems
00-00-8A	Datahouse Information Systems
00-00-8E	Jupiter ? Solbourne ?
00-00-93	Proteon
00-00-94	Asante
00-00-95	Sony/Tektronics
00-00-97	Epoch
08-00-46	Sony
08-00-4E	BICC
08-00-5A	IBM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MAC	Vendor
00-00-98	CrossCom
00-00-9F	Ameristar Technology
00-00-A0	Sanyo Electronics
00-00-A2	Wellfleet
00-00-A3	NAT
00-00-A4	Acorn
00-00-A5	Compatible Systems Corporation
00-00-A6	Network General
00-00-A7	NCD
00-00-A8	Stratus
00-00-A9	Network Systems
00-00-AA	Xerox
00-00-C0	Western Digital
00-00-C9	Emulex
00-AA-00	Intel
00-DD-00	Ungermann-Bass
00-DD-01	Ungermann-Bass
02-07-01	MICOM/Interlan
02-60-8C	3Com
08-00-02	3Com-Bridge
08-00-05	Symbolics
08-00-06	Siemens Nixdorf
08-00-07	Apple
08-00-09	HP
08-00-0A	Nestar Systems
08-00-0B	Unisys
08-00-10	AT&T
08-00-11	Tektronics
08-00-14	Excelan
08-00-20	Sun
08-00-2B	DEC
08-00-38	Bull
08-00-39	Spider

MAC	Vendor
08-00-69	Silicon Graphics
08-00-6E	Excelan
08-00-7C	Vitalink
08-00-89	Kinetics
08-00-90	Retix

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค

ตารางแสดงค่า ซอฟต์แวร์อินเทอร์รัพท์ ช่วง 0x60 ถึง 0x80

60 -- -- reserved for user interrupt

60 -- -- FTP Driver - PC/TCP Packet Driver Specification

60 01 FF FTP Driver - DRIVER INFO

60 02 -- FTP Driver - ACCESS TYPE

60 03 -- FTP Driver - RELEASE TYPE

60 04 -- FTP Driver - SEND PACKET

60 05 -- FTP Driver - TERMINATE DRIVER FOR HANDLE

60 06 -- FTP Driver - GET ADDRESS

60 07 -- FTP Driver - RESET INTERFACE

60 11 -- 10-NET - LOCK AND WAIT

60 12 -- 10-NET - LOCK

60 13 -- 10-NET - UNLOCK

60 20 -- FTP Driver - SET RECEIVE MODE

60 21 -- FTP Driver - GET RECEIVE MODE

60 24 -- FTP Driver - GET STATISTICS

61 -- -- reserved for user interrupt

62 -- -- reserved for user interrupt

63 -- -- reserved for user interrupt

64 -- -- reserved for user interrupt

65 -- -- reserved for user interrupt

66 -- -- reserved for user interrupt

67 -- -- LIM EMS

...

67 DE 00 Virtual Control Program Interface - INSTALLATION CHECK

...

68 01 -- APPC/PC

...

69 -- -- unused

6A -- -- unused

6B -- -- unused

6C -- -- system resume vector (CONVERTIBLE)

6C -- -- DOS 3.2 Realtime Clock update

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6D -- -- VGA - internal

6E -- -- unused

6F -- -- Novell NetWare - PCOX API (3270 PC terminal interface)

6F 00 -- 10-NET - LOGIN

...

70 -- -- IRQ8 - AT/XT286/PS50+ - REAL-TIME CLOCK

71 -- -- IRQ9 - AT/XT286/PS50+ - LAN ADAPTER 1

72 -- -- IRQ10 - AT/XT286/PS50+ - RESERVED

73 -- -- IRQ11 - AT/XT286/PS50+ - RESERVED

74 -- -- IRQ12 - PS50+ - MOUSE INTERRUPT

75 -- -- IRQ13 - AT/XT286/PS50+ - 80287 ERROR

76 -- -- IRQ14 - AT/XT286/PS50+ - FIXED DISK

77 -- -- IRQ15 - AT/XT286/PS50+ - RESERVED

78 -- -- not used

79 -- -- not used

7A -- -- Novell NetWare - LOW-LEVEL API

7A -- -- AutoCAD Device Interface

7B -- -- not used

7C -- -- not used

7D -- -- not used

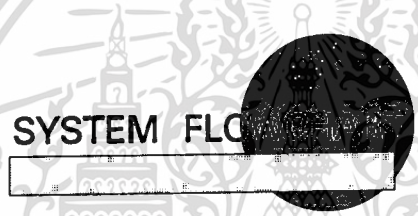
7E -- -- not used

7F -- -- HDILOAD.EXE - 8514/A VIDEO CONTROLLER INTERFACE

7F -- -- HLLAPI (High-Level Language API)

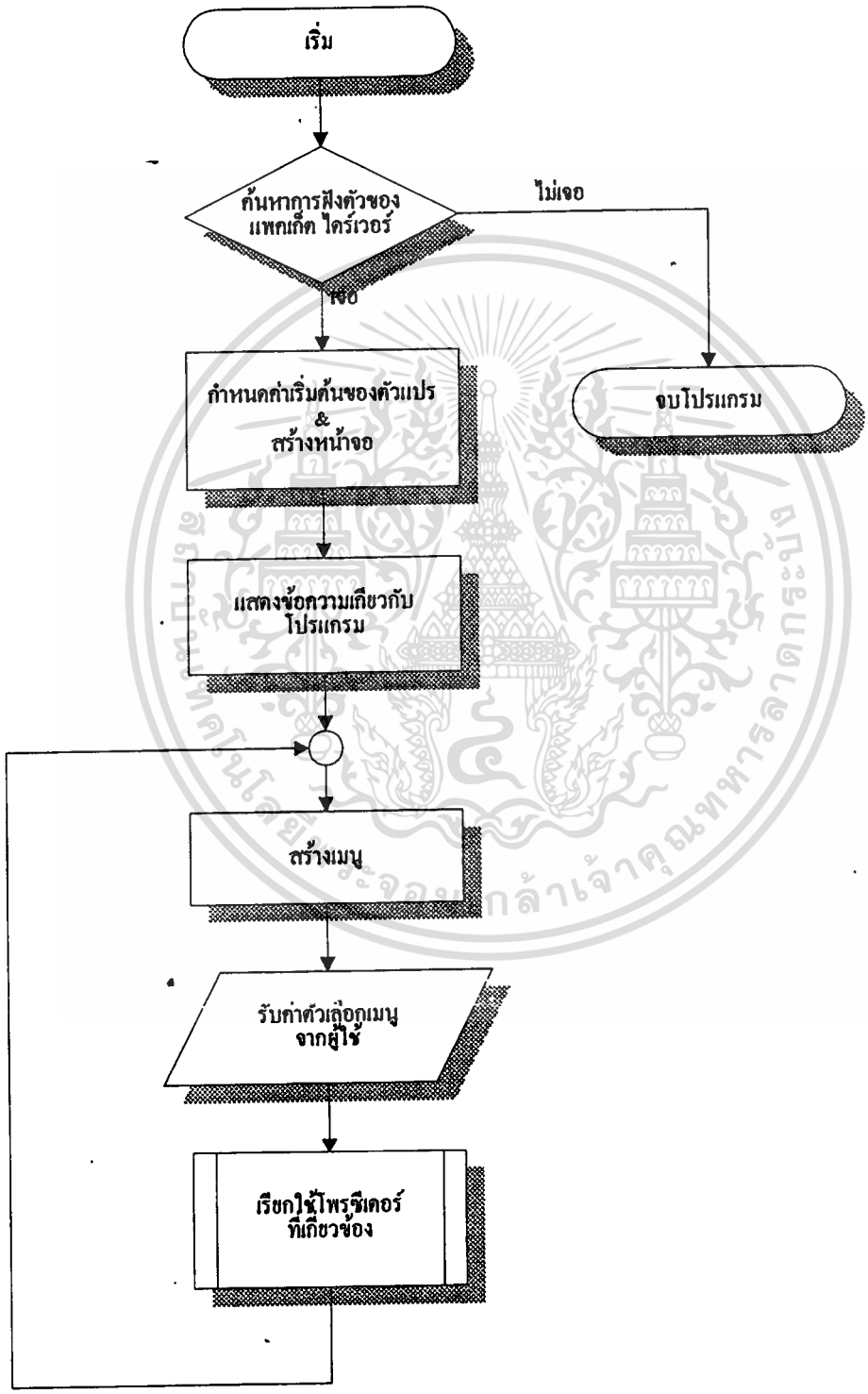
80 -- -- reserved for BASIC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



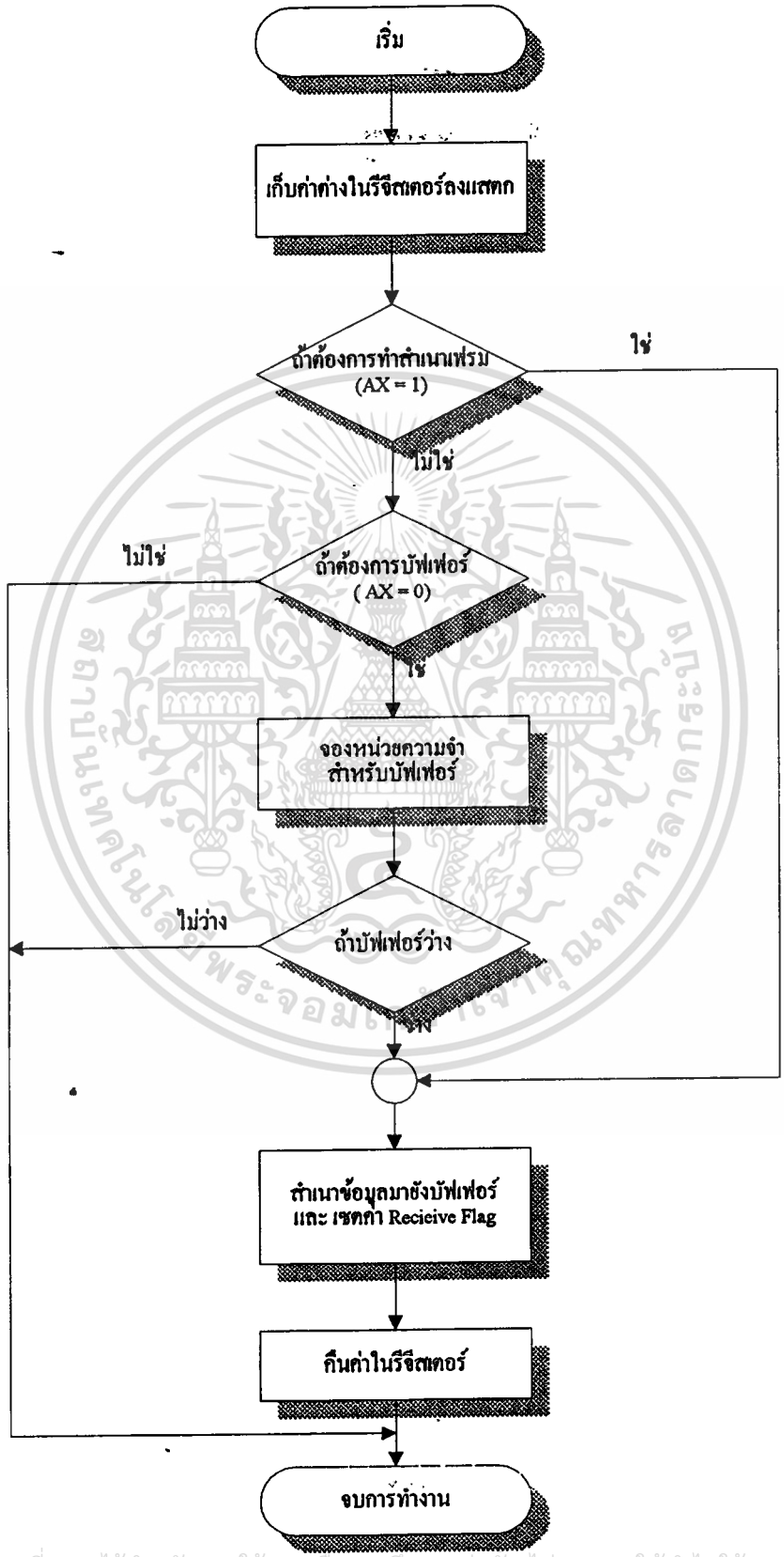
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมหลัก ของ Ethernet Monitor



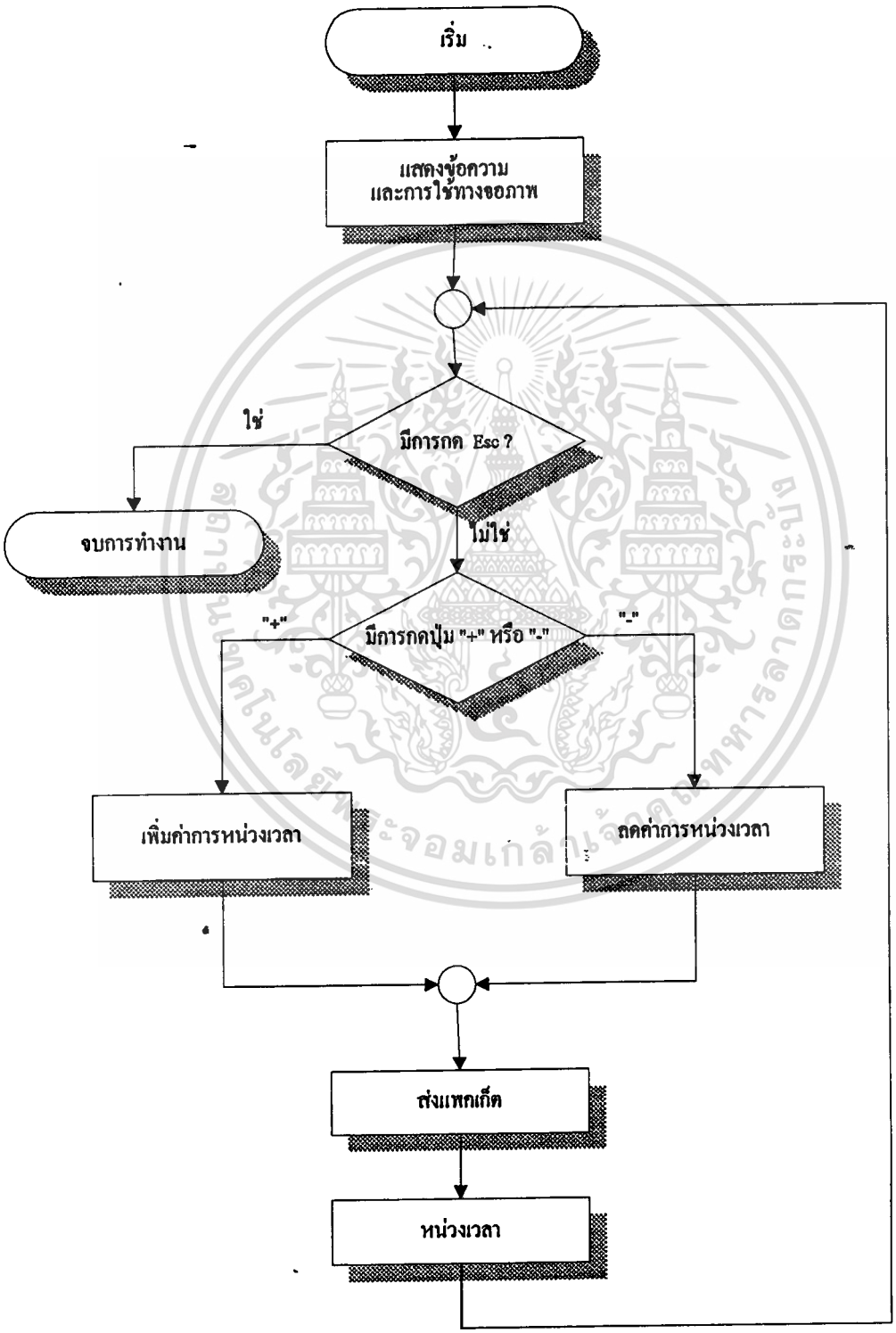
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์ในการรับแพคเก็ตจากแพคเก็ตไดร์เวอร์



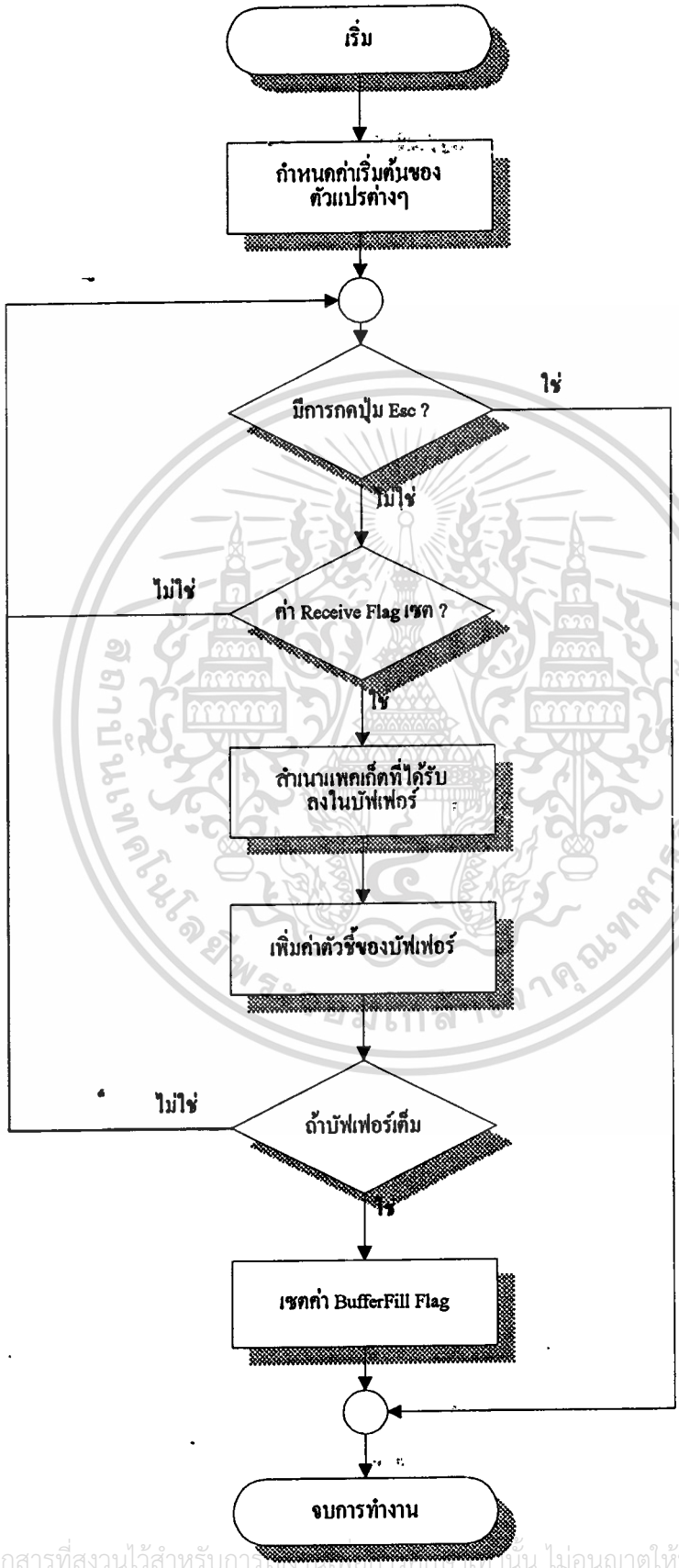
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่งแพคเก็ตลงในระบบเครือข่าย



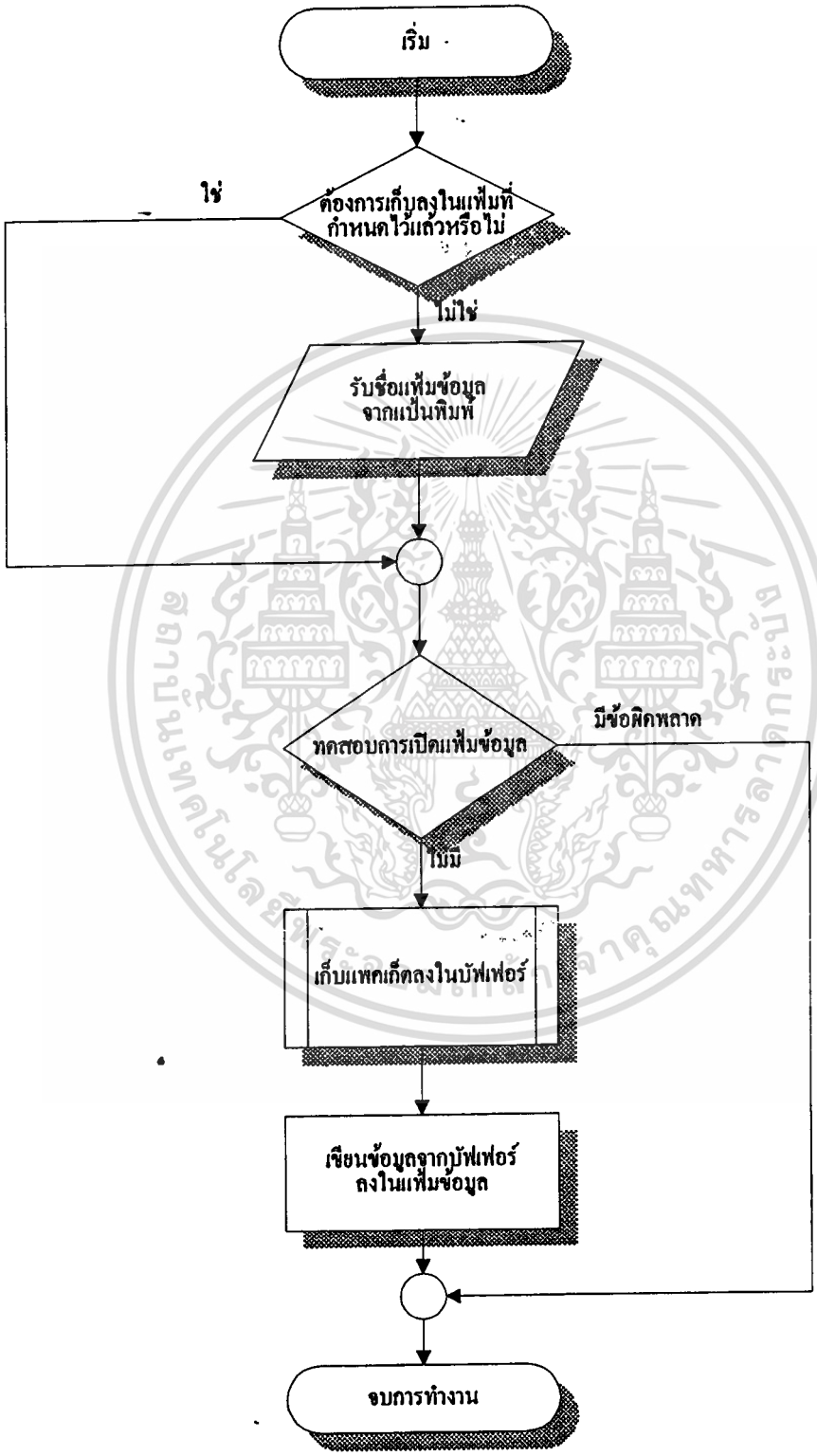
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์จับแพคเก็ตลงบัฟเฟอร์



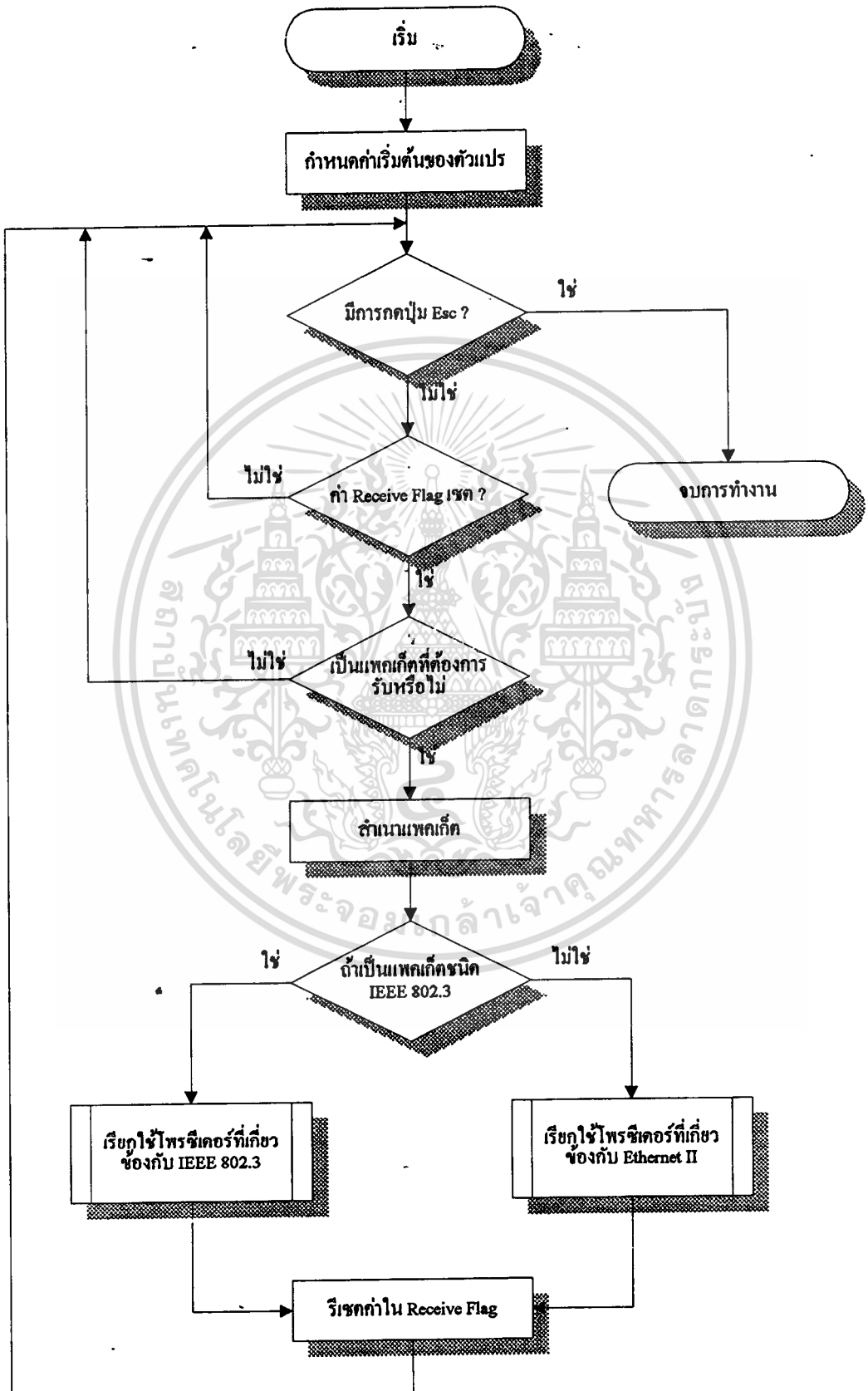
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการ[illegible]เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์จับแพคเก็ตลงแฟ้มข้อมูล



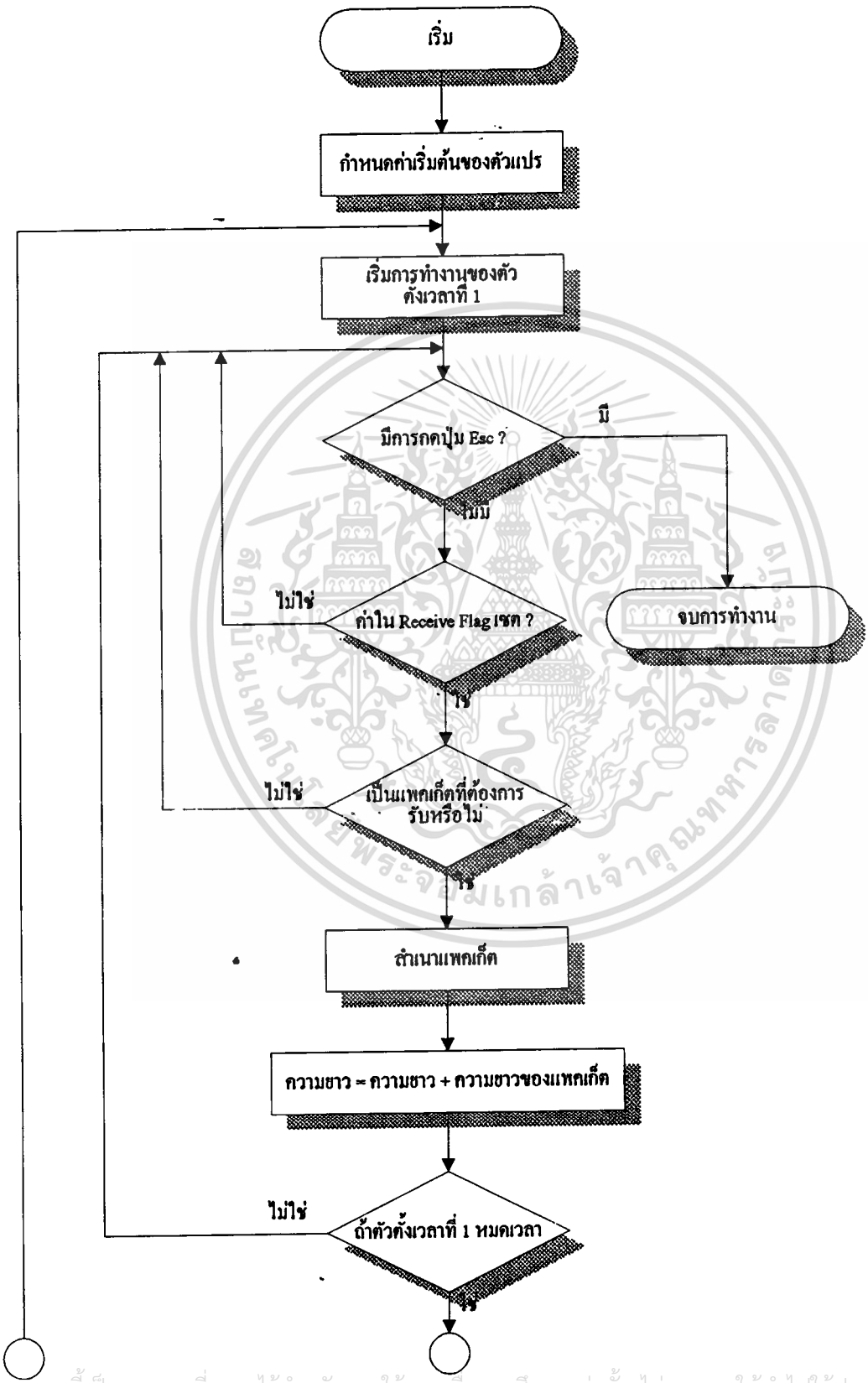
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์แสดงข้อมูลของแพคเก็ตแบบออนไลน์



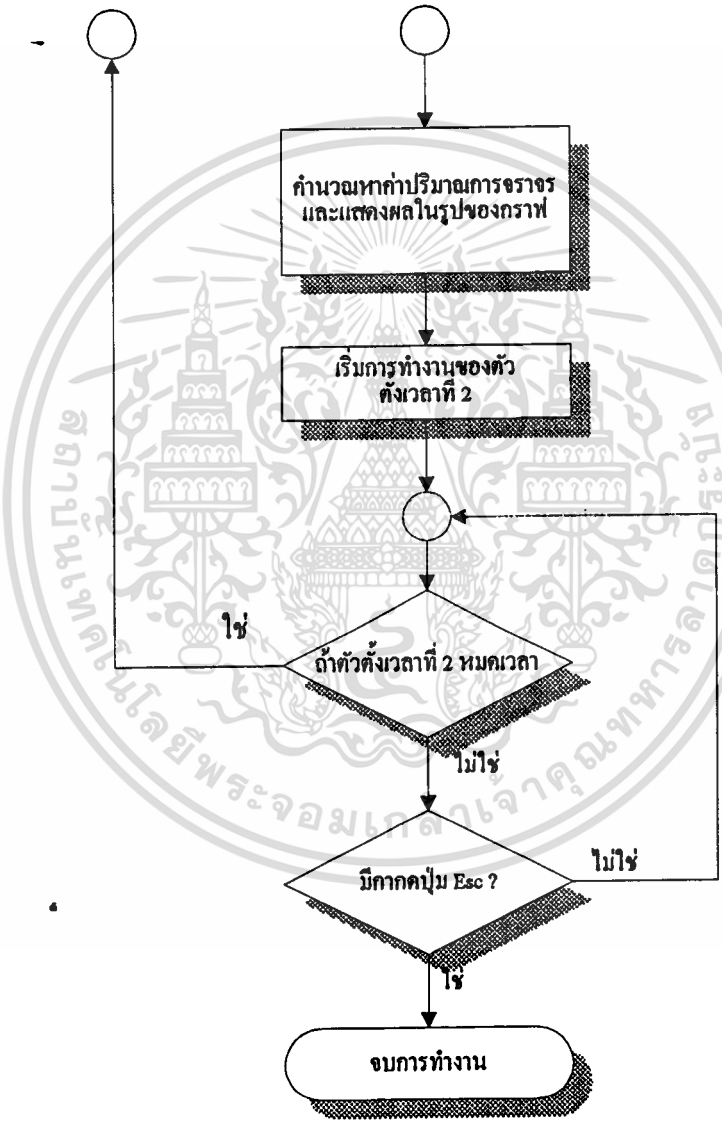
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์แสดงปริมาณความคับคั่ง



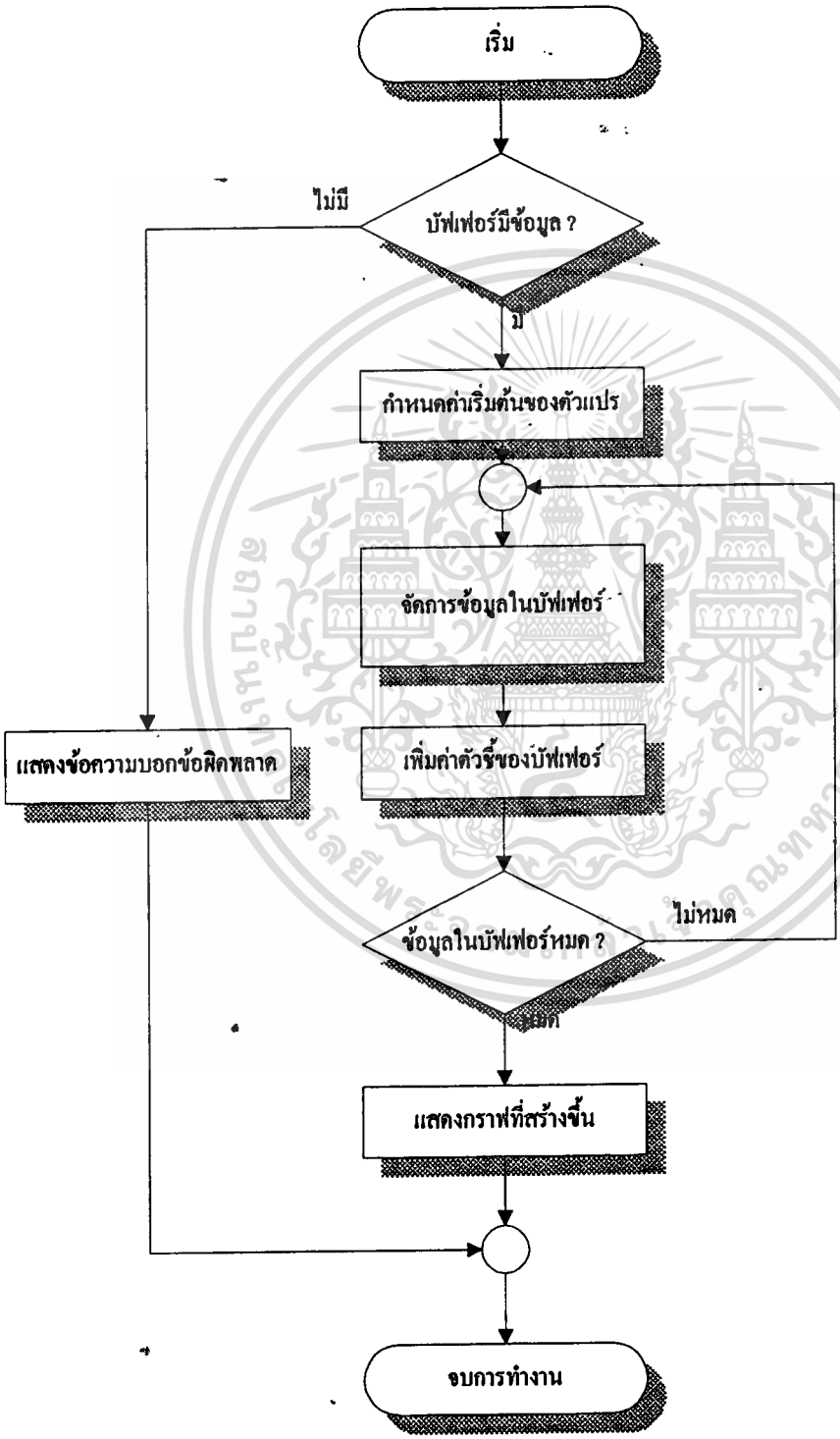
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์แสดงปริมาณความคับคั่ง (ต่อ)



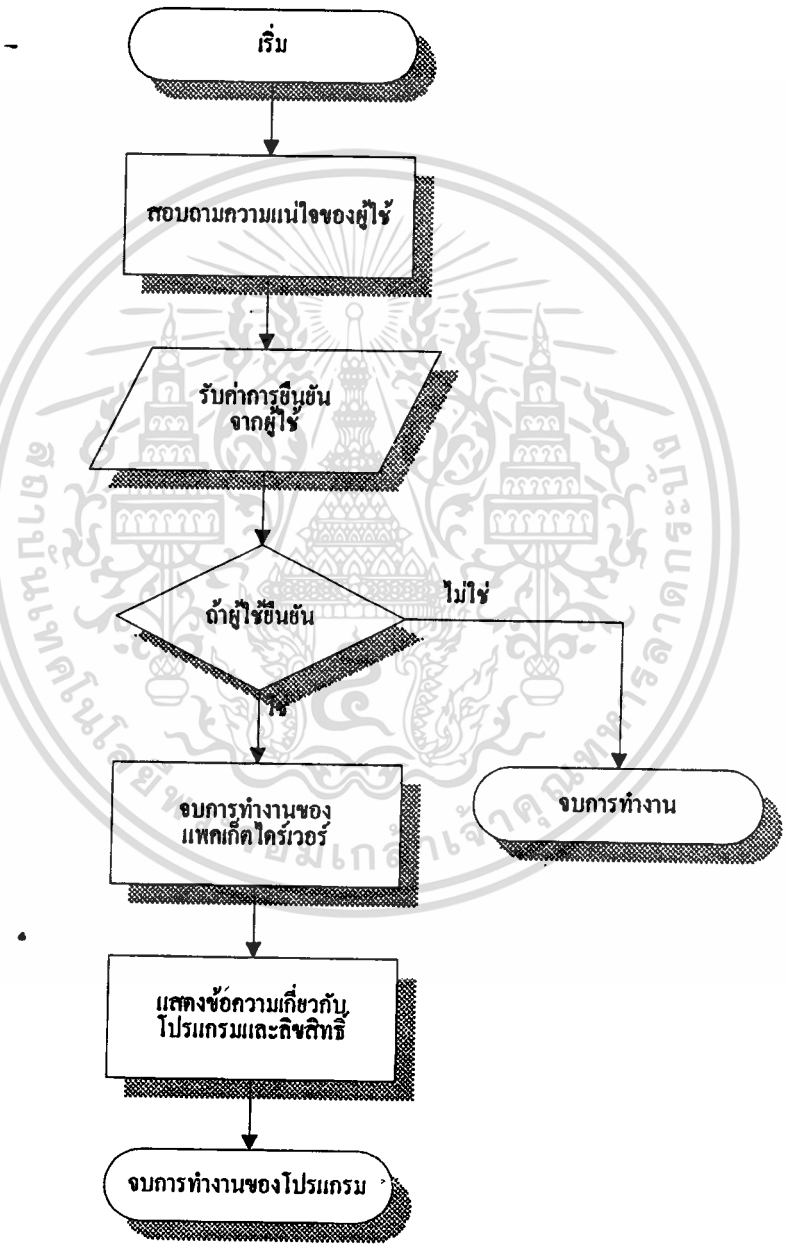
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์แสดงผลกราฟจากบัฟเฟอร์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรซีเดอร์จบการทำงานของโปรแกรม



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Programmin

File Name	Description
ETHERMON.PAS	Ethernet Monitor To Monitor, Analyze and Generate Ethernet Traffic
TIMER.PAS	Unit File used for Timer & Time Stamp
PKTDRVR.PAS	Unit File used to interface with Crynwr Packet Drivers
PACKET.INC	Include file for constants used with PKTDRVR.PAS
NE2000.ASM	Packet Driver Assembly Code for NE2000 Compatible

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ETHERMON.PAS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PROGRAM Ethernet_Monitor;
{
```

Ethernet Monitor Version 1.00
Purpose to Monitor,Analyze and Generate Ethernet Traffic.
Project : Ethernet Network Monitoring System *Authorized By : Mr.Santad Kiatsupaibul ID. 35104465* *Mr.Chairutthana Lowsophonkul ID 35104524* *Advisors : Mr.Bunjong Piyatamrong* *Mr.Somsak Walairachata*
Faculty of Engineering *KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG*
Copyright (C) 1995-1996 Department of Computer Engineering
! ATTENTION ! *The author can not be held responsible for any damages resulting out of the use of this software !!!*

```
}
{$A+,B-,D+,E-,F+,G-,I-,L+,N-,O-,R-,S-,V+,X+}
{$M $2048 $0 $8196}

USES   PKTDRVR,TIMER,STRINGS,CRT,DOS,UI;

CONST  MaxEthernetCapacity = 1250000; { Max. theroetical ethernet capacity }
       DefaultFileName     = 'DATA.DAT';
       MaxPacketBuffer     = 700;
       FullBlock           = '';

TYPE
       { 6 Byte ethernet address }
       EthernetAddr = Array[00..05] of Byte;
       Data48       = Array[00..05] of Byte;
       Data32       = Array[00..03] of Byte;
       Data16       = Array[00..01] of Byte;
       Data8        = Array[00..00] of Byte;
       PacketFrame  = Array[00..1524] of Byte;

       MACHeader    = RECORD { IEEE 802.3 header }
                        DestAddr   : EthernetAddr;
                        SourceAddr : EthernetAddr;
                        TypeLen    : Word;
                      END;

       IPXFrame     = RECORD { IPX Frame }
                        DestAddr   : EthernetAddr;
                        SourceAddr : EthernetAddr;
                        TypeLen    : Word;
                          CheckSum : Data16;
                          Length   : Data16;
                          Transport : Data8;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                PacketType: Data8;
                DestNet   : Data32;
                DestHost  : EthernetAddr;
                DestSock  : Data16;
                SourceNet : Data32;
                SourceHost: EthernetAddr;
                SourceSock: Data16;
                IPXData   : Array[00..1480] of Byte; {576 Theroetical}
              END;

NCPFrame    = RECORD { NCP Packet }
                RequestType  : Word;
                SequenceNum  : Word;
                TaskNumber   : Word;
                Reserved     : Word;
                ServiceCode  : Word;
                NCPData      : Array[00..1480] of Char;
              END;

EtherIIFrame = RECORD { Ethernet II Frame }
               DestAddr     : EthernetAddr;
               SourceAddr   : EthernetAddr;
                 TypeII     : Data16;
                 Version    : Data8;
                 Typeserv   : Data8;
                 Length     : Data16;
                 Identification : Data16;
                 Flag       : Data16;
                 TTL        : Data8;
                 Protocol   : Data8;
                 HChkSum    : Data16;
                 SourceAdd  : Data32;
                 DestinAdd  : Data32;
                 EtherIIData: Array[00..1490] of Byte; {576 Theroetical}
               END;

TCPFrame    =  RECORD  {TCP Frame}
                 SourcePort   : Data16;
                 DestPort     : Data16;
                 SequenceNum  : Data32;
                 Acknowledge  : Data32;
                 DataOffset   : Data16;  {Contain Dataoffset + Reserve + Flag}
                 Window       : Data16;
                 CheckSum     : Data16;
                 UrgentPosite : Data16;
                 TCPData      : Array[00..1470] of Char;
               END;

TPacketInfo =  RECORD
                 Header : MACHeader;
                 Data   : Array[00..31] OF Byte;
                 Hour   : Word;
                 Minute : Word;
                 Second : Word;
                 Sec100 : Word;
               END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
TPacketFile  = File of TPacketInfo;

TPacketIEEE802 = RECORD
                    DestAddr    : EthernetAddr;
                    SourceAddr  : EthernetAddr;
                    TypeLen     : Word;
                    CheckSum    : Data16;
                    Length      : Data16;
                    Transport   : Data8;
                    PacketType  : Data8;
                    DestNet     : Data32;
                    DestHost    : EthernetAddr;
                    DestSock    : Data16;
                    SourceNet   : Data32;
                    SourceHost  : EthernetAddr;
                    SourceSock  : Data16;
                    Hour        : Word;
                    Minute      : Word;
                    Second      : Word;
                    Sec100      : Word;
                 END;

TPacketEtherII = RECORD
                    DestAddr     : EthernetAddr;
                    SourceAddr   : EthernetAddr;
                    TypeII       : Word;

                    Version      : Data8;
                    Typeserv     : Data8;
                    Length       : Data16;
                    Identification : Data16;
                    Flag         : Data16;
                    TTL          : Data8;
                    Protocol     : Data8;
                    HChkSum      : Data16;
                    SourceAdd    : Data32;
                    DestinAdd    : Data32;
                    SourcePort   : Data16;
                    DestPort     : Data16;
                    Hour         : Word;
                    Minute       : Word;
                    Second       : Word;
                    Sec100       : Word;
                 END;

TSendPacket  =   RECORD
                    DestAddr     : EthernetAddr;
                    SourceAddr   : EthernetAddr;
                    TypeLen      : Word;
                    Data         : ARRAY[01..256] OF Char;
                 END;


VAR
   PacketBuffer    : ARRAY[00..MAXPACKETBUFFER] of TPacketInfo;
   RcvRawData      : Array[00..1524] of Char;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pktDriver        : TPKTDRVR;          { Instance of the TPKTDRVR object }
pktDriverInfo    : TDRVRINFO;         { record for driver information }
pktDriverAccess  : TACCESSTYPE;       { record used for accessing packet driver }
pktStatistics    : TSTATISTICS;       { record used for satistics of packet }

Rcvpacket        : PacketFrame;       { Rcv buffer }

RcvIEEE802       : TPacketIEEE802;    { record used for IEEE802.3 type }
RcvNCP           : NCPFrame;
RcvIPX           : IPXFrame;
RcvTCP           : TCPFrame;

RcvEthernetII    : TpacketEtherII;    { record used for Ethernet II type }
RcvEtherLength   : TpacketEtherII;
RcvEtherII       : EtherIIFrame;

RcvLength        : Word;              { Length of packet }
RcvHeader        : MACheader;         { Header           }

SendPacket       : TSendPacket;       { Send packet buffer }
TypeField        : Word;              { type field of Send packet }

PacketCount      : Word;              { Packet counter   }
PacketDelay      : Word;
TotalPacket      : LongInt;

CurTraffic       : Real;              { Used for evalute traffic }
CumTraffic       : LongInt;

I,tInt           : Integer;           { Used for handle file operation }
FrameType        : String;
FileName         : String[13];

AllStation       : Boolean;           { Stored Information of Received Packet }
AllSource        : Boolean;
AllDestination   : Boolean;
AllPacket        : Boolean;
IsEtherII        : Boolean;
IsIEEE802        : Boolean;
SourceAddr       : EthernetAddr;
DestinationAddr  : EthernetAddr;
SendSourceAddr   : EthernetAddr;
SendDestAddr     : EthernetAddr;
BufferCount      : Integer;
Mode             : Integer;
BufferFill       : Boolean;

Timer1           : _TIMER;            { Timer for Trick Monitor Traffic }
Timer2           : _TIMER;
Interval         : LongInt;

MyEthernetAddr   : EthernetAddr;      { My Station Ethernet Address }
SendDataCount    : Byte;              { Counter Size of Send fields }

Menu             : PullDownType;      { Menu Section }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    MenuSelect      : Word;

    Window          : WindowType;       { Window Type }
    C               : Char;             { Temporary Valiable }
    Key             : Char;
    J               : Integer;
    tF              : Text;
    Byt             : Byte;
    tStr,ttStr      : String;
```

```
{ ┌──────────────────────────────────────────────────────────┐
  │ PROCEDURE pktReceiver                                    │
  ├──────────────┬───────────────────────────────────────────┤
  │ Description  : │ Receive Packets                         │
  └──────────────┴─────────────────────────────────────────── }
```

```
{$S-}
PROCEDURE pktReceiver; ASSEMBLER;
ASM
        PUSH AX                         { Push registers onto stack }
        PUSH BX
        PUSH CX
        PUSH DX

        CMP  AX,0001                    { AX=1 means frame copied }
        JZ   @@FrameCopied
        CMP  AX,0000                    { AX=0 means allocate memory please }
        JZ   @@AllocMemory
        JMP  @@EXIT                     { Invalid register contents for AX so exit}

@@AllocMemory:

        MOV  DX,0                       { ES:DI = 0000:0000, we don't want the packet }
        MOV  ES,DX
        MOV  DI,0                       { We don't grab the packet }

        MOV  DX,SEG PacketCount         { Set correct data segment }
        MOV  DS,DX
        MOV  DX,PacketCount
        CMP  DX,0

        JNZ  @@Exit                     { buffer is not free ! }

        MOV  DX,SEG Rcvpacket
        MOV  ES,DX
        MOV  DI,OFFSET Rcvpacket

        MOV  DX,SEG RcvLength
        MOV  DS,DX
        MOV  SI,OFFSET RcvLength
        MOV  WORD PTR DS:[SI],CX        { Store length of frame in PacketLength }

        JMP  @@Exit

@@FrameCopied:

        MOV  DX,SEG PacketCount         { Set correct data segment }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        MOV  DS,DX
        MOV  PacketCount,1              { Set Flag to 1 }

@@Exit:

        POP  DX                         { Pop registers from stack }
        POP  CX
        POP  BX
        POP  AX
END;
{$S+}

{
  PROCEDURE InitPktDriver;
  Description   :  Initialize packet driver.
}
PROCEDURE InitPktDriver;
BEGIN
     WriteLn('Initializing packet driver....');
     IF (pktDriver.GetStatus <> INITIALIZED) THEN
      BEGIN
          WriteLn('Could not initialze packet driver...');
          WriteLn('Aborting...');
          Halt($FF);
      END
     ELSE
      BEGIN
      { Packet driver found. Show some information }
          WriteLn('Packet driver found:');
          pktDriver.DriverInfo(pktDriverInfo);
          WriteLn('Name    = ',StrPas(pktDriverInfo.PName));
          WriteLn('Version = ',pktDriverInfo.Version);
          WriteLn('IF-Type = ',pktDriverInfo.Type_);
          Write  ('Func    = ');

          CASE pktDriverInfo.Functionality OF
            01 : WriteLn('Basic functions present.');
            02 : WriteLn('Basic & extended functions present.');
            05 : WriteLn('Basic & high-performance functions present.');
            06 : WriteLn('Basic, high-performance & extended functions present.');
          END;
      END;

      { Fill in information used for accessing packet driver }
      WITH pktDriverAccess DO
      BEGIN
          if_class  := pktDriverInfo.Class;
          if_type   := ANYTYPE;
          if_number := 0;
          type_     := @TypeField;
          typelen   := 0;
          receiver  := @pktReceiver; { receiver procedure }
      END;

      { Access packet driver }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    pktDriver.AccessType(pktDriverAccess);
    WriteLn('Handle  = ',pktDriver.GetHandle);
    WriteLn;

    { Setting packet driver to promiscuous mode }
    pktDriver.SetRcvmode(6);
END;
```

{

PROCEDURE TerminatePktDriver;	
Description :	Releases the handle and terminates packet driver.

}

```
PROCEDURE TerminatePktDriver;
BEGIN
    pktDriver.ReleaseType;
END;
```

{

FUNCTION ByteToHEXASCII (...) : String;	
Description :	Converts Byte to a HEX-ASCII-String requests.

}

```
FUNCTION ByteToHEXASCII(tByte : Byte) : String;
CONST
    HEXChars: array [0..15] of char = '0123456789ABCDEF';
VAR  Nibble1 : Byte;
    Nibble2 : Byte;
    tStr    : String;
BEGIN
    Nibble1 := (tByte AND $0F);          { AND 00001111b }
    Nibble2 := (tByte AND $F0) SHR 4;    { AND 11110000b }
    tStr := HEXChars[Nibble2]+HEXChars[Nibble1];
    ByteToHEXASCII := tStr;
END;
```

{

FUNCTION WordToHEXASCII (...) : String;	
Description :	Converts Word to a HEX-ASCII-String requests.

}

```
FUNCTION WordToHEXASCII(tWord : Word) : String;
VAR  tStr : String;
BEGIN
    tStr := ByteToHexASCII(Hi(tWord));
    tStr := tStr+ByteToHexASCII(Lo(tWord));
    WordToHexASCII := tStr;
END;
```

{

FUNCTION GetEthernetAddress	
Description :	Converts a 6 byte ethernet address into a dash separated string.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
FUNCTION GetEthernetAddress(tEtherAddr : EthernetAddr) : String;
VAR tString : String;
    tDigit  : String[02];
    tCount  : Byte;
    AddrLen : Byte;
BEGIN
     AddrLen := SizeOf(EthernetAddr)-1;
     tString := '';

     FOR tCount := 0 TO AddrLen DO
     BEGIN
          tString := tString + ByteToHexASCII(tEtherAddr[tCount]);
          IF (tCount < AddrLen) THEN tString := tString + '-';
     END;
     GetEthernetAddress := tString;
END;
```

{

PROCEDURE TranslateEthernetAddress(..);	
Description :	*Translates a dash separated ethernet address into 6 Bytes.*

}

```
PROCEDURE TranslateEthernetAddress(tStr : String; VAR tEtheraddr : EthernetAddr);
VAR tDigit : String;
    tByte  : Byte;
    Err    : Integer;
    tVal   : Byte;
BEGIN
     tDigit :='';
     FOR tByte := 1 TO Length(tStr) DO
     BEGIN
          IF (Copy(tStr,tByte,1) <> '-') THEN tDigit := tDigit + Copy(tStr,tByte,1);
     END;
     FOR tByte := 0 TO 5 DO
     BEGIN
          Val('$'+Copy(tDigit,(tByte*2)+1,2),tVal,Err);
          tEtheraddr[tByte] := tVal;
     END;
END;
```

{

FUNCTION SwapWord;	
Description :	*Swaps the Hi and Lo byte of a word.*

}

```
FUNCTION SwapWord(sWord : Word) : Word;
VAR tWord : Word;
BEGIN
     tWord := (Lo(sWord) SHL 8) + Hi(sWord);
     SwapWord := tWord;
END;
```

{

FUNCTION GetFragment(...) : Byte;

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Description :	Converts Low 13 bits of Word to Word

```
FUNCTION GetFragment(tWord : Data16) : Word;
VAR  ttWord        :  Word;
BEGIN
     MOVE(tword,ttWord,2);
     ttWord := SwapWord(ttWord);
     GetFragment := (ttWord AND $1F);
END;
```

PROCEDURE HideCurSor;	
Description :	HideCurSor

```
PROCEDURE HideCurSor;
VAR       pkRegs         :  Registers;
BEGIN
     WITH pkRegs DO
     BEGIN
          AX := $0100;
          CX := $1300;
          Intr($10,pkRegs);
     END;
END;
```

PROCEDURE ShowCurSor;	
Description :	ShowCurSor

```
PROCEDURE ShowCurSor;
VAR       pkRegs         :  Registers;
BEGIN
     WITH pkRegs DO
     BEGIN
          AX := $0100;
          CX := $1213;
          Intr($10,pkRegs);
     END;
END;
```

FUNCTION CompareEthernetAddr(Addr1,Addr2 : EthernetAAddr) : Boolean;	
Description :	Compare 2 Ethernet Address Reture TRUE if they are EQUAL

```
FUNCTION CompareEthernetAddr(Addr1,Addr2 : EthernetAddr) : Boolean;
VAR       Result : Boolean;
          Count  : Byte;
BEGIN
     Result := TRUE;
     Count  := 0;
     REPEAT
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        IF (Addr1[Count] <> Addr2[Count]) THEN Result := FALSE;
        INC (Count);
    UNTIL Not(Result) OR (Count = 6);
    CompareEthernetAddr  := Result;
END;
```

{

FUNCTION Get4HighBit(...) : Byte;	
Description :	Converts High-Nibble of Byte to Byte

}

```
FUNCTION Get4HighBit(tByte : Data8) : Byte;
VAR      Nibble,ttByte :  Byte;
BEGIN
     MOVE(tByte,ttByte,1);
     Get4Highbit := (ttByte AND $F0) SHR 4;  { AND 11110000b }
END;
```

{

FUNCTION Get4LowBit(...) : Byte;	
Description :	Converts Low-Nibble of Byte to Byte

}

```
FUNCTION Get4LowBit(tByte : Data8) : Byte;
VAR      Nibble,ttByte :  Byte;
BEGIN
     MOVE(tByte,ttByte,1);
     Get4Lowbit := (ttByte AND $0F) ;  { AND 00001111b }
END;
```

{

FUNCTION Get8bit(tdata8 : data8) : String;	
Description :	Convert Array of Data 8 bit -> String

}

```
FUNCTION Get8bit(tdata8 : data8) : String;
VAR tString : String;
    tCount  : Byte;
    AddrLen : Byte;
BEGIN
     AddrLen := SizeOf(data8)-1;
     tString := '';
     FOR tCount:=0  TO AddrLen DO
     BEGIN
          tString := tString + ByteToHexASCII(tdata8[tCount]);
     END;
     Get8bit := tString;
END;
```

{

FUNCTION GetData16toASCII(tdata16 : data16) : String;	
Description :	Convert Array of Data 16 bit -> String

}

```
FUNCTION GetData16ToASCII(tdata16 : data16) : String;
VAR    tWord : Word ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BEGIN
     Move(tdata16,tWord,2);
     GetData16ToASCII := WordToHEXASCII(SwapWord(tWord));
END;
```

PROCEDURE Get16bit(tdata16 : data16) : String;	
Description :	Convert Array of Data 16 bit -> String

```
FUNCTION Get16bit(tdata16 : data16) : String;
VAR tString : String;
    tCount  : Byte;
    AddrLen : Byte;
BEGIN
     AddrLen := SizeOf(data16)-1;
     tString := '';
     FOR tCount:=0  TO AddrLen DO
     BEGIN
          tString := tString + ByteToHexASCII(tdata16[tCount]);
          IF (tCount < AddrLen) THEN tString := tString + '-';
     END;
     Get16bit := tString;
END;
```

FUNCTION Get32bit(tdata32 : data32) : String;	
Description :	Convert Array of Data 32 bit -> String

```
FUNCTION Get32bit(tdata32 : data32) : String;
VAR tString,ttString : String[8];
    tWord            : Array[00..01] of Word;
    tCount           : Byte;
BEGIN
     Move(tData32,tWord,4);
     tString := WordToHEXASCII(SwapWord(tWord[0]));
     ttString := tString + WordToHEXASCII(SwapWord(tWord[1]));
     Get32Bit := ttString;
END;
```

FUNCTION PrintIP(tdata32 : data32) : String;	
Description :	Convert Array of Data 32 bit -> String

```
FUNCTION PrintIP(tdata32 : data32) : String;
VAR tString  : String;
    tString1 : String;
    tDigit   : String[02];
    tCount   : Byte;
    AddrLen  : Byte;
BEGIN
     AddrLen := SizeOf(tdata32)-1;
     tString := '';
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
     FOR tCount := 0 TO AddrLen DO
     BEGIN
          Str(tdata32[tCount],tString1);
          tString := tString + tString1;
          IF (tCount < AddrLen) THEN tString := tString + '.';
     END;
     PrintIP := tString;
END;
```

{

PROCEDURE MakeFrame(X1,Y1,X2,Y2,Color,Border : Integer);	
Description :	MakeFrame Border. Border = 1 ====> Single 2 ====> Double

}

```
PROCEDURE MakeFrame(X1,Y1,X2,Y2,Color,Border : Integer);
VAR       I,J,K,L : Integer;
BEGIN
     TextColor(Color);
     IF Border = 1 THEN
      BEGIN
           K := X2-X1-1;
           L := Y2-Y1-1;

           GoToXY(X1,Y1);Write('┌');
           FOR I := 1 TO K DO
               Write('─');
           Write('┐');

           FOR J := 1 TO L DO
           BEGIN
                GoToXY(X1,Y1+J);Write('│');
                GoToXY(X2,Y1+J);Write('│');
           END;

           GoToXY(X1,Y2);Write('└');
           FOR I := 1 TO K DO
               Write('─');
           Write('┘');
      END
      ELSE
      BEGIN
           K := X2-X1-1;
           L := Y2-Y1-1;

           GoToXY(X1,Y1);Write('╒');
           FOR I := 1 TO K DO
               Write('═');
           Write('╕');

           FOR J := 1 TO L DO
           BEGIN
                GoToXY(X1,Y1+J);Write('│');
                GoToXY(X2,Y1+J);Write('│');
           END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        GoToXY(X1,Y2);Write('╚');
        FOR I := 1 TO K DO
            Write('═');
        Write('╝');
    END;
    TextColor(7);
END;
```

{

PROCEDURE PutBar(X,Y,W,H,Color:Integer;Percent:Real);	
Description :	-Create Color Bar Input : Location,Color,Value,High

}

```
PROCEDURE PutBar(X,Y,W,H,Color:Integer;Percent:Real);
VAR     I,J : Integer;
BEGIN
    TextColor(Color);
    FOR I := 0 TO H-1 Do
    BEGIN
        GotoXY(X,Y-I);
        FOR J := 1 TO W DO
        BEGIN
            Write(FullBlock)
        END;
    END;
    X := X + (W DIV 2) - 2;
    GotoXY(X,Y-H-1);Write(TRUNC(Percent),'%');
END;
```

{

PROCEDURE GetMyEthernetAddr;	
Description :	Get my Station Address

}

```
PROCEDURE GetMyEthernetAddr;
VAR     cWord : Word;
BEGIN
    pktDriver.GetAddress(@MyEthernetAddr,SizeOf(MyEthernetAddr),cWord);
END;
```

{

PROCEDURE Message(sTitle,sStr:String);	
Description :	Create Pop up window for display message Input : Title,Message

}

```
PROCEDURE Message(sTitle,sStr:String);
VAR  DialogRT    : Word;
    DialogPopup : DialogType;
BEGIN
    HideCursor;
    DialogPopup.Init(20,10,40,5,stitle);
    DialogPopup.AddText(1,0,3,1,$70,sStr);
    DialogPopup.AddButton(2, 0, 17, 3, RetDlgOK, ' OK ', 'OK !');
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    DialogRT := DialogPopup.Run;
    DialogPopup.Done;
END;
```

{

PROCEDURE Warning;	
Description :	Display Warning Dialog

}

```
PROCEDURE Warning;
VAR  DialogRT    : Word;
     DialogPopup : DialogType;
BEGIN
     Hidecursor;
     DialogPopup.Init(10,9,60,7,' Warning ! ');
     DialogPopup.AddText(1,0,5,2,$70,'Do not use this program to Destroy System Security.');
     DialogPopup.AddText(2,0,3,3,$70,'The author can not be held responsible for any damaged.');
     DialogPopup.AddButton(3,0,12,5,RetDlgOK,' ACCEPT ', 'I accepted the condition !');
     DialogPopup.AddButton(4,0,35,5,RetDlgCancel,' Not ACCEPT ', 'I don''t accept,Exit Program.');
     DialogRT := DialogPopup.Run;
     DialogPopup.Done;
     IF DialogRT = $FF01 THEN
     BEGIN
          TerminatePktDriver;
          ClrScr;
          WriteLn('
          WriteLn('          Ethernet Monitor Version 1.00                    ');
          WriteLn('
          WriteLn(' Authorized: Santad Kiatsupaibul          ID 35104465    ');
          WriteLn('             Chairutthana Lowsophonkul    ID 35104524    ');
          WriteLn('
          WriteLn(' Advisor   : Bunjong Piyatamrong                         ');
          WriteLn('             Somsak Walairachata                         ');
          WriteLn('
          WriteLn('       Project of Computer Engineering KMITL             ');
          WriteLn('          Copyright (C) 1995-1996                        ');
          WriteLn('
          ShowCurSor;
          Halt(0);
     END;
END;
```

{

PROCEDURE GenPacket;	
Description :	Create Color Bar Input : Location,Color,Value,High

}

```
PROCEDURE GenPacket;
VAR      Window    : WindowType;
         tFileName : String[13];
         ch        : Char;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BEGIN
     Window.bX:=15; Window.bY:=8; Window.bW:=50; Window.bH:=12;
     Window.bAttr := 6;
     Window.bBorderAttr:=1;
     Window.bTitleAttr:=6;
     Window.bBStyle:=2;
     Window.sTitle:=' Generate Packets ';
     WindowOpen(Window);
     HideCursor;
     TextColor(WHITE);

     GotoXY(5,1);Write('Simulate traffic on this network segment.');
     GotoXY(13,3);Write('Press        to Speed up !  ');
     GotoXY(13,4);Write('Press        to Speed down !');
     GotoXY(9,6);Write('Send Speed :        packets/Second.');
     GotoXY(19,8);Write('Send       Packets.');
     GotoXY(16,10);Write('Press        To Exit.');
     TextColor(YELLOW);
     GotoXY(20,3);Write('''+''');
     GotoXY(20,4);Write('''-''');
     GotoXY(22,10);Write('[Esc]');

     PacketCount := 0; PacketDelay := 10;
     REPEAT
          Delay(PacketDelay);
          IF KeyPressed THEN Key := ReadKey;
          CASE Key OF
            '-' : BEGIN IF (PacketDelay < 1000)THEN Inc(PacketDelay,1); Key := #00; END;
            '+' : BEGIN IF (PacketDelay > 1   ) THEN Dec(PacketDelay,1); Key := #00; END;
          END;
     { Send packet }
     IF SendDataCount < 47 THEN
     BEGIN
          FOR I := SendDataCount to 60 DO
              SendPacket.Data[I] := #0;
     END;
     pktDriver.SendPkt(@SendPacket,13+SendDataCount);  { SendDataCount + 14 - 1 }
     Inc(PacketCount);
     GotoXY(22,6);Write(1000 Div PacketDelay,' ');
     GotoXY(24,8);Write(PacketCount);

     UNTIL (Key = #27);

     Key := #0;
     WindowClose;
END;
```

{

PROCEDURE SetTimeInterval(sTime:Integer);	
Description :	Set time duration Input : time

}

```
PROCEDURE SetTimeInterval(sTime:Integer);
BEGIN
     Interval := sTime;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
END;
```

{

PROCEDURE SetStatus	
Description :	Maintain Status for receive packet

}

```
PROCEDURE SetStatus;
BEGIN
     IF AllStation AND AllPacket THEN Mode := 1;
     IF AllStation AND IsEtherII AND NOT(IsIEEE802) THEN Mode := 2;
     IF AllStation AND NOT(IsEtherII) AND IsIEEE802 THEN Mode := 3;
     IF AllSource AND NOT(AllDestination) AND AllPacket THEN Mode := 5;
     IF AllSource AND NOT(AllDestination) AND IsEtherII AND NOT(IsIEEE802)
        THEN Mode := 6;
     IF AllSource AND NOT(AllDestination) AND NOT(IsEtherII) AND IsIEEE802
        THEN Mode := 7;
     IF NOT(AllSource) AND AllDestination AND AllPacket THEN Mode := 9;
     IF NOT(AllSource) AND AllDestination AND IsEtherII AND NOT(IsIEEE802)
        THEN Mode := 10;
     IF NOT(AllSource) AND AllDestination AND NOT(IsEtherII) AND IsIEEE802
        THEN Mode := 11;
     IF NOT(AllSource) AND NOT(AllDestination) AND AllPacket THEN Mode := 13;
     IF NOT(AllSource) AND NOT(AllDestination) AND IsEtherII AND NOT(IsIEEE802)
        THEN Mode := 14;
     IF NOT(AllSource) AND NOT(AllDestination) AND NOT(IsEtherII) AND IsIEEE802
        THEN Mode := 15;
     IF NOT(IsEtherII) AND NOT(IsIEEE802) THEN Mode := 16
END;
```

{

FUNCTION TestOpenFile(Name:String):Boolean;	
Description :	Test File is open able Input : File name to test

}

```
FUNCTION TestOpenFile(Name:String):Boolean;
VAR
  tFile : TPacketFile;
BEGIN
     Assign(tFile,Name);              { Open Data File For Flush Data }
     {$I-} Rewrite(tFile); {$I+}
     IF (IOResult <> 00) THEN
       BEGIN
          Message(' Error ','Can not open file,please check it !');
          TestOpenFile := FALSE;
       END
     ELSE
       BEGIN
          Close(tFile);
          TestOpenFile := TRUE;
     END;
END;
```

{

PROCEDURE CapturePacket;

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Description :	Capture Packet to file.

```
PROCEDURE CapturePacket;
VAR
      Hr,Minu,Scd,Scd100 : Word;
      tInfo  : TPacketInfo;
      tFile  : TPacketFile;
      I,J    : Integer;
      Window : WindowType;
      tInt   : Integer;
BEGIN
     I := 0;
     TextColor(YELLOW);
     Window.bX:=20; Window.bY:=12; Window.bW:=40; Window.bH:=3;
     Window.bAttr := 6;
     Window.bBorderAttr:=1;
     Window.bTitleAttr:=6;
     Window.bBStyle:=2;
     Window.sTitle:=' Collect Packets ';
     WindowOpen(Window);
     HideCursor;

     REPEAT
          IF KeyPressed THEN Key := ReadKey;
          IF ( PacketCount = 1 ) THEN       { Receive Packet }
          BEGIN
               GetTime(Hr,Minu,Scd,Scd100);
               Move(Rcvpacket,PacketBuffer[I],576);
               PacketBuffer[I].Hour   := Hr;
               PacketBuffer[I].Minute := Minu;
               PacketBuffer[I].Second := Scd;
               PacketBuffer[I].Sec100 := Scd100;
               I:=I+1;
               RcvLength   := 0;
               PacketCount := 0;              { Reset flag for new frames }

               GotoXY(6,1);Write('Packet Receive : ',I,' Packets.');
          END;
     UNTIL (Key = #27) OR (I = MaxPacketBuffer);   { Stop when ESC or buffer full }
     Key := #0;

     GotoXY(9,1);Write('Now flushing buffer to disk.');
     Assign(tFile,FileName);             { Open Data File For Flush Data }
     {$I-} Rewrite(tFile); {$I+}
     IF (IOResult <> 00) THEN
     BEGIN
          WindowClose;
          Message(' Error ','Can not open file,please check it !');
          Exit;
     END;

     FOR J:= 0 TO I-1 DO
     BEGIN
          Write(tFile,PacketBuffer[J]);
     END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    WindowClose;
    BufferCount := I;
    Close(tFile);
    BufferFill := TRUE;
END;
```

{

PROCEDURE CaptureToBuffer;	
Description :	Capture Packet to Memory Buffer

}

```
PROCEDURE CaptureToBuffer;
VAR
     Hr,Minu,Scd,Scd100 : Word;
     tFile1 : TEXT;
     tInfo  : TPacketInfo;
     tFile  : TPacketFile;
     I,J    : Integer;
     Window : WindowType;
BEGIN
    I := 0;
    TextColor(YELLOW);
    Window.bX:=20; Window.bY:=12; Window.bW:=40, Window.bH:=3;
    Window.bAttr := 6;
    Window.bBorderAttr:=1;
    Window.bTitleAttr:=6;
    Window.bBStyle:=2;
    Window.sTitle:=' Collect Packets ';
    WindowOpen(Window);
    HideCursor;

    REPEAT
        IF KeyPressed THEN Key := ReadKey;
        IF ( PacketCount = 1 ) THEN      { Receive Packet }
        BEGIN
            GetTime(Hr,Minu,Scd,Scd100);
            Move(Rcvpacket,PacketBuffer[I],576);
            PacketBuffer[I].Hour   := Hr;
            PacketBuffer[I].Minute := Minu;
            PacketBuffer[I].Second := Scd;
            PacketBuffer[I].Sec100 := Scd100;
            Inc(I);
            GotoXY(6,1);Write('Packet Receive : ',I,' Packets.');
            RcvLength   := 0;
            PacketCount := 0;            { Reset flag for new frames }
        END;
    UNTIL (Key = #27) OR (I = MaxPacketBuffer);   { Stop when ESC or buffer full }
    Key := #0;
    BufferCount := I;
    BufferFill  := TRUE;
    WindowClose;
END;
```

{

PROCEDURE BufferToDisk

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Description :	Save Buffer to Disk

}

```
PROCEDURE BufferToDisk;
VAR     Window    : WindowType;
        tFileName : String[13];
        tFile     : TPacketFile;
        J         : Byte;
BEGIN
     Window.bX:=18; Window.bY:=10; Window.bW:=44; Window.bH:=3;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=1;
     Window.bTitleAttr:=6;
     Window.bBStyle:=2;
     Window.sTitle:='';

     IF NOT(BufferFill) THEN
     BEGIN
          Message(' Error ','Packet must be captured to Buffer !');
          Exit;
     END;
     WindowOpen(Window);
     TextColor(Yellow);
     Write('  Enter file name to save : ');
     ReadLn(tFileName);
     WindowClose;
     IF tFileName ='' THEN EXIT;
     IF NOT(TestOpenFile(tFileName)) THEN EXIT;

     FileName :=  tFileName;
     Assign(tFile,FileName);              { Open Data File For Flush Data }
     {$I-} Rewrite(tFile); {$I+}

     IF (IOResult <> 00) THEN
     BEGIN
          Message(' Error ','Can not open file,please check  !');
          Exit;
     END;

     TextColor($F5);
     Window.bX:=20; Window.bY:=12; Window.bW:=40; Window.bH:=3;
     Window.bAttr := 6;
     Window.bBorderAttr:=1;
     Window.bTitleAttr:=6;
     Window.bBStyle:=2;
     Window.sTitle:=' Collect Packets ';
     WindowOpen(Window);
     HideCursor;
     GotoXY(7,1);Write('Now flushing buffer to disk.');
     FOR J:= 0 TO BufferCount-1 DO
     BEGIN
          Write(tFile,PacketBuffer[J]);
     END;
     Close(tFile);
     WindowClose;
END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

{

PROCEDURE LoadBackGround	
Description :	Load Dune BackGround

}

```
PROCEDURE LoadBackGround;
BEGIN
     For I := 0 To 1919 Do
         Mem[$B800:$A0+(I*2)] := 176;
     FOR I := 0 To 1919 Do
         Mem[$B800:$A1+(I*2)] := $07;
END;
```

{

PROCEDURE ExitProgram	
Description :	Exit Program

}

```
PROCEDURE ExitProgram;
VAR  DialogRT   : Word;
     DialogPopup : DialogType;
BEGIN
     HideCursor;
     DialogPopup.Init(24,10,32,5,'');
     DialogPopup.AddText(1, 0, 6,1, $70, 'Do you want to EXIT ?');
     DialogPopup.AddButton(2, 0, 7, 3, RetDlgOK, ' OK ', 'Yes, I want to exit.');
     DialogPopup.AddButton(3, 0, 17, 3, RetDlgCancel, 'Cancel', 'No, I don''t');
     DialogRT := DialogPopup.Run;
     IF DialogRT = $FF00 THEN
     BEGIN
          TerminatePktDriver;
          ClrScr;
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          WriteLn('
          ShowCurSor;
          Halt(0);
     END;
     DialogPopup.Done;
END;
```

Ethernet Monitor Version 1.00
Authorized: Santad Kiatsupaibul ID 35104465 Chairutthana Lowsophonkul ID 35104524
Advisor : Bunjong Piyatamrong Somsak Walairachata
Project of Computer Engineering KMITL Copyright (C) 1995-1996

{

Function IsPacketVerify(PacketHeader : MacHeader) : Boolean;	
Description :	Is Received packet match requirement ? Input : Packet Header

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Function IsPacketVerify(PacketHeader : MacHeader) : Boolean;
BEGIN
     IsPacketVerify := FALSE;
     CASE Mode OF
       1  :  IsPacketVerify := TRUE;
       2  :  IF (SwapWord(PacketHeader.TypeLen) >  $5EE) THEN IsPacketVerify := TRUE;
       3  :  IF (SwapWord(PacketHeader.TypeLen) <= $5EE) THEN IsPacketVerify := TRUE;
       5  :  IF CompareEthernetAddr(PacketHeader.DestAddr,DestinationAddr)
                THEN IsPacketVerify := TRUE;
       6  :  IF CompareEthernetAddr(PacketHeader.DestAddr,DestinationAddr) AND
                (SwapWord(PacketHeader.TypeLen) > $5EE) THEN IsPacketVerify := TRUE;
       7  :  IF CompareEthernetAddr(PacketHeader.DestAddr,DestinationAddr) AND
                (SwapWord(PacketHeader.TypeLen) <= $5EE) THEN IsPacketVerify := TRUE;
       9  :  IF CompareEthernetAddr(PacketHeader.SourceAddr,SourceAddr)
                THEN IsPacketVerify := TRUE;
      10  :  IF CompareEthernetAddr(PacketHeader.SourceAddr,SourceAddr) AND
                (SwapWord(PacketHeader.TypeLen) > $5EE) THEN IsPacketVerify := TRUE;
      11  :  IF CompareEthernetAddr(PacketHeader.SourceAddr,SourceAddr) AND
                (SwapWord(PacketHeader.TypeLen) <= $5EE) THEN IsPacketVerify := TRUE;
      13  :  IF CompareEthernetAddr(PacketHeader.SourceAddr,SourceAddr) AND
                CompareEthernetAddr(PacketHeader.DestAddr,DestinationAddr)
                THEN IsPacketVerify := TRUE;
      14  :  IF CompareEthernetAddr(PacketHeader.SourceAddr,SourceAddr) AND
                CompareEthernetAddr(PacketHeader.DestAddr,DestinationAddr) AND
                (SwapWord(PacketHeader.TypeLen) > $5EE)
                THEN IsPacketVerify := TRUE;
      15  :  IF CompareEthernetAddr(PacketHeader.SourceAddr,SourceAddr) AND
                CompareEthernetAddr(PacketHeader.DestAddr,DestinationAddr) AND
                (SwapWord(PacketHeader.TypeLen) <= $5EE)
                THEN IsPacketVerify := TRUE;
     END;
END;
```

PROCEDURE SetFileName	
Description :	*Get file name from user and save buffer to its*

```
PROCEDURE SetFileName;
VAR     Window    : WindowType;
        tFileName : String[13];
BEGIN
     Window.bX:=18; Window.bY:=10; Window.bW:=44; Window.bH:=3;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=1;
     Window.bTitleAttr:=6;
     Window.bBStyle:=2;
     Window.sTitle :='';
     WindowOpen(Window);

     TextColor(Yellow);
     Write('  Enter file name to save : ');
     Readln(tFileName);
     WindowClose;
     IF tFileName = '' THEN EXIT;
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    IF NOT(TestOpenFile(tFileName)) THEN
     BEGIN
         EXIT
     END
    ELSE
     BEGIN
         FileName :=  tFileName;
         CapturePacket;
     END;
END;
```

{

PROCEDURE SetDefaultFileName;	
Description :	Save buffer to default File name

}

```
PROCEDURE SetDefaultFileName;
BEGIN
     FileName := DefaultFileName;
     CapturePacket;
END;
```

{

PROCEDURE ExecFile(CommanLine,Param:String);	
Description :	Execute external file Input : command line and parameter

}

```
PROCEDURE ExecFile(CommanLine,Param:String);
VAR     Window     : WindowType;
BEGIN
     Window.bX:=1; Window.bY:=1; Window.bW:=80; Window.bH:=25;
     Window.bAttr := 0;
     Window.bBorderAttr:=0;
     Window.bTitleAttr:=0;
     Window.bBStyle:=1;
     Window.sTitle:='';
     WindowOpen(Window);
     TextColor(Yellow);
     GotoXY(35,12);Write('Loading ...');

     SwapVectors;
     Exec(CommanLine,Param);
     SwapVectors;
     IF DosError <> 0 THEN{ Error? }
         WriteLn('Dos error #', DosError)
     ELSE
         WriteLn('Exec successful. ','Child process exit code = ',DosExitCode);
     WindowClose;
END;
```

{

PROCEDURE About(Mode : Boolean);	
Description :	Show About Message

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    Input : TRUE > Until keypress;FALSE > Until timeout
}
PROCEDURE About(Mode : Boolean);
VAR     Window     : WindowType;
BEGIN
     Window.bX:=12; Window.bY:=5; Window.bW:=57; Window.bH:=17;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=BROWN;
     Window.bTitleAttr:=15;
     Window.bBStyle:=2;
     Window.sTitle:=' Ethernet Monitor Version 1.00 ';
     WindowOpen(Window);
     HideCursor;

     TextColor(LIGHTCYAN);
     WriteLn('');
     WriteLn('      Project Ethernet Network Monitoring System');
     TextColor(DARKGRAY);
     WriteLn(' ──────────────────────────────────────────────────');
     TextColor(LIGHTGREEN);
     WriteLn('                          Authorized By');
     TextColor(YELLOW);
     WriteLn('       Mr.Santad Kiatsupaibul         ID 35104465');
     WriteLn('       Mr.Chairutthana Lowsophonkul   ID 35104524');
     TextColor(DARKGRAY);
     WriteLn(' ──────────────────────────────────────────────────');
     TextColor(LIGHTGREEN);
     WriteLn('                          Advisor ');
     TextColor(YELLOW);
     WriteLn('                  Mr.Bunjong Piyatamrong');
     WriteLn('                  Mr.Somsak Walairachata');
     TextColor(DARKGRAY);
     WriteLn(' ──────────────────────────────────────────────────');
     TextColor(LIGHTMAGENTA);
     WriteLn('');
     WriteLn('    Copyright (C) 1995-1998 By Computer Engineering');
     Write  ('    King Mongkut''s Institute of Technology Ladkrabang');

     IF Mode THEN ReadKey ELSE
     BEGIN
          FOR I:= 0 TO 300 DO
             IF KeyPressed THEN
              BEGIN
                  WindowClose;
                  Exit;
              END
             ELSE Delay(10);
     END;
     WindowClose;
END;
```

```
{
  FUNCTION LogicToEng(Logics:Boolean):String;

  Description   :  Change Logic to Yes or No String
                   Input : Boolean
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
FUNCTION LogicToEng(Logics:Boolean):String;
VAR      tString : String[4];
BEGIN
     IF Logics THEN LogicToEng := 'Yes.'
     ELSE LogicToEng := 'No. ';
END;

{
  PROCEDURE DisplayStatus(delaytime:Boolean);
  Description   :  Show current status of receive require
                   Input : TRUE > until timeout;FALSE >  until Keypres
}
PROCEDURE DisplayStatus(delaytime:Boolean);
VAR      Window     : WindowType;
BEGIN
     SetStatus;
     Window.bX:=14; Window.bY:=7; Window.bW:=52; Window.bH:=14;
     Window.bAttr := 0;
     Window.bBorderAttr:=6;
     Window.bTitleAttr:=15;
     Window.bBStyle:=2;
     Window.sTitle:=' Status ';
     WindowOpen(Window);

     TextColor(WHITE);
     GotoXY(19,1);Write('Receive Station');
     TextColor(LIGHTGRAY);
     GotoXY(4,2) ;Write('──────────────────────────────────────────────');
     TextColor(WHITE);
     GotoXY(4,3);
     IF AllSource THEN
      BEGIN
         Write('All source address       : ');
         TextColor(YELLOW);
         Write(LogicToEng(AllSource));
      END
     ELSE
      BEGIN
         TextColor(WHITE);
         Write('Only source address      : ');
         TextColor(YELLOW);
         Write(GetEthernetAddress(SourceAddr));
      END;
    GotoXY(4,4);
    IF AllDestination THEN
     BEGIN
        TextColor(WHITE);
        Write('All destination address  : ');
        TextColor(YELLOW);
        Write(LogicToEng(AllDestination));
     END
    ELSE
     BEGIN
        TextColor(WHITE);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        Write('Only Destination address : ');
        TextColor(YELLOW);
        Write(GetEthernetAddress(DestinationAddr));
      END;
    TextColor(WHITE);
    GotoXY(20,6);Write('Receive Packet');
    TextColor(LIGHTGRAY);
    GotoXY(4,7);Write('------------------------------------------------');
    TextColor(WHITE);
    GotoXY(4,8);Write('IEEE 802.3 packets      : ');
    TextColor(YELLOW);
    Write(LogicToEng(IsIEEE802));
    TextColor(WHITE);
    GotoXY(4,9);Write('Ethernet  packets       : ');
    TextColor(YELLOW);
    Write(LogicToEng(IsEtherII));
    TextColor(LIGHTGRAY);
    GotoXY(4,11);Write('------------------------------------------------');
    TextColor(WHITE);
    GotoXY(4,12);Write('This Station MAC Address : ');
    TextColor(YELLOW);
    Write(GetEthernetAddress(MyEthernetAddr));

    IF DelayTime THEN Delay(2500)
    ELSE  ReadKey;
    WindowClose;
END;
```

{

PROCEDURE DisplayIPXInfo;	
Description :	Display information of IPX Frame.

}

```
PROCEDURE DisplayIPXInfo;
VAR   tType  : Byte;
      tWord  : Word;
      ttType : Byte;
BEGIN
    GotoXY(2,5);
    IF (RcvIPX.CheckSum[1] <> $FF) THEN
          Write('CheckSum       : ',GetData16ToASCII(RcvIPX.CheckSum))
    ELSE
          Write('CheckSum       : Disabled (FFFF)     ');

    GotoXY(2,6);Write('Length          : ',GetData16ToASCII(RcvIPX.Length),'                ');
    GotoXY(2,7);Write('Transport       : ',Get8Bit(RcvIPX.Transport),'     ');

    GotoXY(2,8);Move(RcvIPX.PacketType,ttType,SizeOf(ttType));
       CASE ttType OF
          $00 : Write('Packet_Type  : Unknown Frame (',Get8Bit(RcvIPX.PacketType),') ');
          $01 : Write('Packet_Type  : RIP Packet (',Get8Bit(RcvIPX.PacketType),')     ');
          $02 : Write('Packet_Type  : Echo Packet (',Get8Bit(RcvIPX.PacketType),')    ');
          $03 : Write('Packet_Type  : Error Packet (',Get8Bit(RcvIPX.PacketType),')   ');
          $04 : Write('Packet_Type  : PEP Packet (',Get8Bit(RcvIPX.PacketType),')     ');
          $05 : Write('Packet_Type  : SPP Packet (',Get8Bit(RcvIPX.PacketType),')     ');
          $11 : BEGIN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            Write('Packet_Type  :  NCP Packet (',Get8Bit(RcvIPX.PacketType),')  ');
            Move(RcvIPX.IPXData,RcvNCP,SizeOf(RcvIPX.IPXData));
            GoToXY(41,5);Write('Request Type : ',WordToHEXASCII(RcvNCP.RequestType));
            GoToXY(41,6);Write('Sequence Number :',WordToHEXASCII(RcvNCP.SequenceNum));
            GoToXY(41,7);Write('Task Number :  ',WordToHEXASCII(RcvNCP.TaskNumber));
            GoToXY(41,8);Write('Reserved :  ',WordToHEXASCII(RcvNCP.Reserved));
            GoToXY(41,9);Write('Service Code :  ',WordToHEXASCII(RcvNCP.ServiceCode));
            GoToXY(41,10);Write('                                    ');
            GoToXY(41,11);Write('                                    ');
            GoToXY(41,12);Write('                                    ');
            GoToXY(41,13);Write('                                    ');
            GoToXY(2,18);
                  Move(RcvIPX.Length,tWord,2);
                  tWord := SwapWord(tWord);
                  I := 0;
                  While (( I < (tWord-42)) AND ( I < 75 ))  DO
                  BEGIN
                       CASE RcvNCP.NCPData[I] OF
                          ' '..'~' :  Write(RcvNCP.NCPData[I]);
                       ELSE
                          Write('.');
                       END;
                          I:=I+1;
                  END;
                  While I < 75 DO BEGIN Write(' ');Inc(I); END;
             END;
   ELSE;
   Write('Packet_Type    :  N/A (',Get8Bit(RcvIPX.PacketType),')          ');
   END;

GotoXY(2,9);Write('Dest. Network  :  ',Get32Bit(RcvIPX.DestNet),'          ');
GotoXY(2,10);Write('Dest. Host     :  ',GetEthernetAddress(RcvIPX.DestHost),' ');
GotoXY(2,11);Move(RcvIPX.DestSock,tWord,2);
   tWord := SwapWord(tWord);
   CASE tWord OF
      $0451 :Write('Dest. Socket   :  File Server       ');
      $0452 :Write('Dest. Socket   :  Service Advertising');
      $0453 :Write('Dest. Socket   :  Routing Information');
      $0455 :Write('Dest. Socket   :  NetBIOS Packet    ');
      $0456 :Write('Dest. Socket   :  Diagnostic Packet ');
   ELSE
   Write(          'Dest. Socket   :  Normal Socket     ');
   END;

GotoXY(2,12);Write('Source Network :  ',Get32Bit(RcvIPX.SourceNet),'          ');
GotoXY(2,13);Write('Source Host    :  ',GetEthernetAddress(RcvIPX.SourceHost));
GotoXY(2,14);Move(RcvIPX.SourceSock,tWord,2);
   tWord := SwapWord(tWord);
   CASE tWord OF
      $0451 :Write('Source Socket  :  File Server       ');
      $0452 :Write('Source Socket  :  Service Advertising');
      $0453 :Write('Source Socket  :  Routing Information');
      $0455 :Write('Source Socket  :  NetBIOS Packet    ');
      $0456 :Write('Source Socket  :  Diagnostic Packet ');
   ELSE
      Write(       'Source Socket  :  Normal Socket     ')
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    END;
  GoToXY(2,15);Write('                                        ');
  IF ( tType <> $11 ) THEN
  BEGIN
        GoToXY(41,5); Write('                                 ');
        GoToXY(41,6); Write('                                 ');
        GoToXY(41,7); Write('                                 ');
        GoToXY(41,8); Write('                                 ');
        GoToXY(41,9); Write('                                 ');
        GoToXY(41,10);Write('                                 ');
        GoToXY(41,11);Write('                                 ');
        GoToXY(41,12);Write('                                 ');
        GoToXY(41,13);Write('                                 ');
  END;
  IF ttType <> $11 THEN
  BEGIN
       GoToXY(2,18); Write('                                  ');
                     Write('                                   ');
  END;
END;
```

{

PROCEDURE DisplayEtherII;	
Description :	Display Information of Ethernet II Frame.

}

```
PROCEDURE DisplayEthernetII;
VAR   tByte : Byte;
      tWord : Word;
BEGIN
  GotoXY(2,5);Write('Version          :  ',Get4Highbit(RcvEtherII.version));
  GotoXY(2,6);Write('IP Header Length:  ',Get4Lowbit(RcvEtherII.version));
  GotoXY(2,7);Write('Type of service :  ',Get8bit(RcvEtherII.typeserv));
  GotoXY(2,8);Write('Length          :  ',GetData16ToASCII(RcvEtherII.length));
  GotoXY(2,9);Write('Identification  :  ',GetData16ToASCII(RcvEtherII.Identification));
  GotoXY(2,10);Write('Fragment offset :  ',GetFragment(RcvEtherII.Flag));
  GotoXY(2,11);Write('TimeToLife      :  ',Get8bit(RcvEtherII.TTL));
  GotoXY(2,12);BEGIN
               Move(RcvEtherII.protocol,tByte,1);
                 CASE tByte OF
                      06  : BEGIN
                            Move(RcvEtherII.EtherIIData,RcvTCP,SizeOf(RcvTCP));
                            Write('Protocol         :  TCP                ');
                            Move(RcvTCP.SourcePort,tWord,2);
                            tWord := SwapWord(tWord);
                            GoToXY(41,5);
                            CASE tWord OF
                                 05 :Write('Source Port       :  Remote Job Entry ');
                                 07 :Write('Source Port       :  Echo             ');
                                 09 :Write('Source Port       :  DisCard          ');
                                 11 :Write('Source Port       :  Systat           ');
                                 13 :Write('Source Port       :  DayTime          ');
                                 15 :Write('Source Port       :  NetStat          ');
                                 17 :Write('Source Port       :  Quote of the Day ');
                                 20 :Write('Source Port       :  FTP Data         ');
                                 21 :Write('Source Port       :  FTP Control      ');
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            23 :Write('Source Port      :  TELNET          ');
            25 :Write('Source Port      :  SMTP            ');
            37 :Write('Source Port      :  Time            ');
            53 :Write('Source Port      :  Name Server     ');
        ELSE
            Write('Source Port : UnKnown ',
                  GetData16ToASCII(RcvTCP.SourcePort))
        END;
        Move(RcvTCP.DestPort,tWord,2);
        tWord := SwapWord(tWord);
        GoToXY(41,6);
        CASE tWord OF
            05 :Write('Dest.  Port      :  Remote Job Entry ');
            07 :Write('Dest.  Port      :  Echo             ');
            09 :Write('Dest.  Port      :  DisCard          ');
            11 :Write('Dest.  Port      :  Systat           ');
            13 :Write('Dest.  Port      :  DayTime          ');
            15 :Write('Dest.  Port      :  NetStat          ');
            17 :Write('Dest.  Port      :  Quote of the Day ');
            20 :Write('Dest.  Port      :  FTP Data         ');
            21 :Write('Dest.  Port      :  FTP Control      ');
            23 :Write('Dest.  Port      :  TELNET           ');
            25 :Write('Dest.  Port      :  SMTP             ');
            37 :Write('Dest.  Port      :  Time             ');
            53 :Write('Dest.  Port      :  Name Server      ');
        ELSE
            Write('Dest.  Port      :  UnKnown ',
                  GetData16ToASCII(RcvTCP.DestPort))
        END;

        GoToXY(41,7);Write('Sequence Number  :  ',
                           Get32Bit(RcvTCP.SequenceNum));
        GoToXY(41,8);Write('Acknowledgment # :  ',
                           Get32Bit(RcvTCP.Acknowledge));
        GoToXY(41,9);Write('Data OffSet      :  ',
                           GetData16ToASCII(RcvTCP.DataOffSet));
        GoToXY(41,10);Write('Window          :  ',
                           GetData16ToASCII(RcvTCP.Window));
        GoToXY(41,11);Write('CheckSum        :  ',
                           GetData16ToASCII(RcvTCP.CheckSum));
        GoToXY(41,12);Write('Urgent Position :  ',
                           GetData16ToASCII(RcvTCP.UrgentPosite));
        GoToXY(2,18);
        Move(RcvEtherII.Length,tWord,2);
        tWord := SwapWord(tWord);
        I := 0;
        While (( I < (tWord-40)) AND ( I < 75 ))  DO
        BEGIN
            CASE RcvTCP.TCPData[I] OF
                ' '..'~' :  Write(RcvTCP.TCPData[I]);
            ELSE
                Write('.');
            END;
            I:=I+1;
        END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    While I < 75 DO BEGIN Write(' ');Inc(I); END;
                    END;
                17  : Write('Protocol        :  UDP                ');

                ELSE
                    Write(  'Protocol        :  UnKnown Protocol   ');
                    GoToXY(41,5); Write('                                  ');
                    GoToXY(41,6); Write('                                  ');
                    GoToXY(41,7);Write('                                  ');
                    GoToXY(41,8);Write('                                  ');
                    GoToXY(41,9);Write('                                  ');
                    GoToXY(41,10);Write('                                 ');
                    GoToXY(41,11);Write('                                 ');
                    GoToXY(41,12);Write('                                 ');
                END;
            END;
    GotoXY(2,13);Write('Header CheckSum :  ',GetData16ToASCII(RcvEtherII.HChkSum));
    GotoXY(2,14);Write('                                    ');
    GotoXY(2,15);Write('                                    ');
    GotoXY(2,14);Write('Source Address  :  ',PrintIP(RcvEtherII.SourceAdd),'   ');
    GotoXY(2,15);Write('Dest. Address   :  ',PrintIP(RcvEtherII.DestinAdd),'   ');
    IF (tByte <> $6) THEN
    BEGIN
        GoToXY(2,18); Write('                                  ');
                      Write('                                  ');
    END;
END;
```

{

PROCEDURE DisplayAllInformation;	
Description :	Display all packet information

}

```
PROCEDURE DisplayAllInformation;
VAR    Window    :  WindowType;
BEGIN
     Window.bX:=1; Window.bY:=3; Window.bW:=80; Window.bH:=22;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=WHITE;
     Window.bTitleAttr:=YELLOW;
     Window.bBStyle:=2;
     Window.sTitle:=' On-line Packet Information ';
     WindowOpen(Window);
     HideCursor;
     TextColor(WHITE);

     MakeFrame(1,1,78,3,14,2);
     MakeFrame(1,4,39,16,13,2);
     MakeFrame(40,4,78,16,13,2);
     MakeFrame(1,17,78,19,10,2);

     TextColor(15);
     GoTOXY(24,1);WriteLn(' Data Link Control Layer Frame ');
     GoTOXY(9,4);WriteLn(' Network Layer Frame ');
     GoTOXY(51,4);WriteLn(' Upper Layer Frame ');
     GoTOXY(34,17);WriteLn(' Actual Data ');
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    TextColor(14);
    GoToXY(2,2);Write('Desc :                    Src :                    Type :');
    TextColor(11);

    REPEAT
        IF KeyPressed THEN Key := ReadKey;

        IF (PacketCount = 1) THEN
        BEGIN
            Move(Rcvpacket,RcvHeader,SizeOf(RcvHeader));
            IF IsPacketVerify(RcvHeader) THEN
            BEGIN
                TextColor(LIGHTRED);
                GoToXY(9,2);Write(GetEthernetAddress(RcvHeader.DestAddr));
                GoToXY(34,2);Write(GetEthernetAddress(RcvHeader.SourceAddr));
                GoToXY(72,2);Write('[',WordToHexASCII(SwapWord(RcvHeader.TypeLen)),']');

                IF (SwapWord(RcvHeader.TypeLen) <= $5EE) THEN
                  BEGIN
                     Move(Rcvpacket,RcvIPX,SizeOf(Rcvpacket));
                     GoToXY(60,2);Write('IEEE 802.3 ');
                     TextColor(LIGHTCYAN);
                     DisplayIPXInfo;
                  END
                ELSE
                  IF (SwapWord(RcvHeader.TypeLen) = $800) THEN
                  BEGIN
                     Move(Rcvpacket,RcvEtherII,SizeOf(Rcvpacket));
                     GoToXY(60,2);Write('Ethernet II');
                     TextColor(WHITE);
                     DisplayEthernetII;
                  END;
            END;
        END;
        RcvLength   :=0;
        PacketCount := 0; { Reset flag that new frames can be received }

  UNTIL (Key = #27);
  Key := #0;
  WindowClose;
END;
```

{

PROCEDURE GetAssignAddr(VAR Addr:EthernetAddr);	
Description :	Get Ethernet Address from user Output : Ethernet Address

}

```
PROCEDURE GetAssignAddr(VAR Addr:EthernetAddr);
VAR     Window    : WindowType;
        tAddr     : String[17];
BEGIN
     Window.bX:=15; Window.bY:=7; Window.bW:=50; Window.bH:=3;
     Window.bAttr := BROWN;
     Window.bBorderAttr:=6;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Window.bTitleAttr:=15;
    Window.bBStyle:=2;
    Window.sTitle:='';
    WindowOpen(Window);

    TextColor(YELLOW);
    GotoXY(5,1);Write('Input Focus Address : ');
    TextColor(7);
    Write('00-00-00-00-00-00');
    TextColor(BROWN);
    ShowCursor;
    GotoXY(27,1);ReadLn(tAddr);
    HideCursor;
    TranslateEthernetAddress(tAddr,Addr);
    WindowClose;
END;
```

{

PROCEDURE SetAllDestinationAddr(Logics:Boolean);	
Description :	Set receive packet destination address Input : TRUE > all addr ; FALSE > get from assign

}

```
PROCEDURE SetAllDestinationAddr(Logics:Boolean);
BEGIN
    IF NOT(Logics) THEN
    BEGIN
        AllStation := FALSE;
        GetAssignAddr(DestinationAddr);
    END;
    IF Logics AND AllSource THEN ALLStation := TRUE;
    AllDestination := Logics;
    DisplayStatus(TRUE);
END;
```

{

PROCEDURE SetAllSourceAddr(Logics:Boolean);	
Description :	Set receive packet source address Input : TRUE > all addr ; FALSE > get from assign

}

```
PROCEDURE SetAllSourceAddr(Logics:Boolean);
BEGIN
    IF NOT(Logics)  THEN
    BEGIN
        AllStation := FALSE;
        GetAssignAddr(SourceAddr);
    END;
    IF Logics AND AllDestination THEN ALLStation := TRUE;
    AllSource := Logics;
    DisplayStatus(TRUE);
END;
```

{

PROCEDURE SetAllStation(Logics:Boolean);	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Description :	set receive packet for all address Input : Boolean

```
PROCEDURE SetAllStation(Logics:Boolean);
BEGIN
     AllStation     := Logics;
     AllSource      := Logics;
     AllDestination := Logics;
     DisplayStatus(TRUE);
END;
```

PROCEDURE SetEtherII;	
Description :	Set receive packet for Ethernet II

```
PROCEDURE SetEtherII;
BEGIN
     IsEtherII       := NOT(IsEtherII);
     IF (IsEtherII AND IsIEEE802) THEN AllPacket := TRUE
     ELSE AllPacket := FALSE;
     IF IsEtherII THEN Menu.SetState(312, $01) ELSE Menu.SetState(312, $0);
     DisplayStatus(TRUE);
END;
```

PROCEDURE SetIEEE802;	
Description :	Set receive packet for IEEE8002

```
PROCEDURE SetIEEE802;
BEGIN
     IsIEEE802       := NOT(IsIEEE802);
     IF (IsEtherII AND IsIEEE802) THEN AllPacket := TRUE
     ELSE AllPacket := FALSE;
     DisplayStatus(TRUE);
END;
```

PROCEDURE SetAllPacket;	
Description :	Set receive packet for All Packet

```
PROCEDURE SetAllPacket;
BEGIN
     AllPacket       := TRUE;
     IsEtherII       := TRUE;
     IsIEEE802       := TRUE;
     DisplayStatus(TRUE);
END;
```

PROCEDURE SetSendSourceAddr;	
Description :	Get source address for send packet

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PROCEDURE SetSendSourceAddr;
BEGIN
     GetAssignAddr(SendPacket.SourceAddr);
END;
```

{

PROCEDURE SetSendDestinationAddr;	
Description :	Get destination address for send packet

}

```
PROCEDURE SetSendDestinationAddr;
BEGIN
     GetAssignAddr(SendPacket.DestAddr);
END;
```

{

PROCEDURE SetSendData	
Description :	Set Send Data field of send packet

}

```
PROCEDURE SetSendData;
VAR     Window     : WindowType;
        I          : Word;
        Data       : ARRAY[0..255] of Char;
BEGIN
     Window.bX:=5; Window.bY:=7; Window.bW:=70; Window.bH:=5;
     Window.bAttr := 7;
     Window.bBorderAttr:=6;
     Window.bTitleAttr:=15;
     Window.bBStyle:=2;
     Window.sTitle:='';
     WindowOpen(Window);

     TextColor(7);
     SendDataCount := 0;
     Data[0] := 'A';
     ShowCursor;
     GotoXY(2,2);Write('Data : ');

     REPEAT
          Inc(SendDataCount);
          Data[SendDataCount] := Readkey;
          Write(Data[SendDataCount]);
     UNTIL NOT((Data[SendDataCount] <> #13) AND (SendDataCount <= 255));
     FOR I:= 1 TO SendDataCount DO
          SendPacket.Data[I] := Data[I];
     SendPacket.TypeLen := SwapWord(SendDataCount-1+14);

     WindowClose;
END;
```

{

PROCEDURE DisplayDataLink;	
Description :	Show packet information in Datalink layer

}

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PROCEDURE DisplayDataLink;
VAR  Window    : WindowType;
BEGIN
     Window.bX:=1; Window.bY:=3; Window.bW:=80; Window.bH:=22;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=WHITE;
     Window.bTitleAttr:=YELLOW;
     Window.bBStyle:=2;
     Window.sTitle:=' Data Link Control Layer Header ';
     WindowOpen(Window);
     HideCursor;
     TextColor(WHITE);

     REPEAT
          IF KeyPressed THEN Key := ReadKey;

          IF (PacketCount = 1) THEN
          BEGIN
               Move(Rcvpacket,RcvHeader,SizeOf(RcvHeader));
               IF IsPacketVerify(RcvHeader) THEN
               BEGIN
                   Write(#10,#13,' Desc : ',GetEthernetAddress(RcvHeader.DestAddr),'Src:'
                        ,GetEthernetAddress(RcvHeader.SourceAddr),'  Type : ','['
                        ,WordToHexASCII(SwapWord(RcvHeader.TypeLen)),'] ');
                   IF (SwapWord(RcvHeader.TypeLen) <= $5EE) THEN Write('IEEE 802.3 ')
                    ELSE  Write('Ethernet II');
               END;
          END;
          PacketCount := 0;
          RcvLength   := 0;
     UNTIL (Key = #27);
     Key := #0;
     WindowClose;
END;
```

{

PROCEDURE LoadBufferFile;	
Description :	Load File fill in buffer

}

```
PROCEDURE LoadBufferFile;
VAR     Window    : WindowType;
        I         : Integer;
        tFileName : String[13];
        tFile     : TPacketFile;
BEGIN
     Window.bX:=18; Window.bY:=10; Window.bW:=44; Window.bH:=3;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=1;
     Window.bTitleAttr:=6;
     Window.bBStyle:=2;
     Window.sTitle:='';
     WindowOpen(Window);

     TextColor(Yellow);
     Write('  Enter file name to open : ');
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
     ReadLn(tFileName);
     WindowClose;

     FileName :=  tFileName;
     Assign(tFile,FileName);              { Open Data File For Flush Data }
     {$I-} Reset(tFile); {$I+}
     IF (IOResult <> 00) THEN
     BEGIN
          Message(' Error ','Can not open file,please check  !');
          WindowClose;
          Exit;
     END;

     I := 0;
     WHILE NOT Eof(tFile) DO
     BEGIN
          Read(tFile,PacketBuffer[I]);
          Inc(I);
     END;
     BufferFill := TRUE;
     BufferCount := I;
     Close(tFile);
END;

{
  PROCEDURE DisplayRawPacket;
  Description   :  Show Raw Packet Infomation
}
PROCEDURE DisplayRawPacket;
VAR  Window     : WindowType;
     I,J,K,L    : Word;
BEGIN
     Window.bX:=1; Window.bY:=3; Window.bW:=80; Window.bH:=22;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=WHITE;
     Window.bTitleAttr:=YELLOW;
     Window.bBStyle:=2;
     Window.sTitle:=' RAW Packet Information ';
     WindowOpen(Window);
     HideCursor;
     TextColor(WHITE);
     K := 0;L := 0;

     REPEAT
          IF KeyPressed THEN Key := ReadKey;
          IF (PacketCount = 1) THEN
          BEGIN
               Move(Rcvpacket,RcvHeader,SizeOf(RcvHeader));
               IF IsPacketVerify(RcvHeader) THEN
               BEGIN
                    Move(Rcvpacket,RcvRawData,RcvLength);
                    WriteLn(#10,#13,' ',GetSystemTimeAndMilliSec,
                            ' » Received Packet',RcvLength,'Bytes.');
                    WHILE (K <= RcvLength) DO
                    BEGIN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    Write(' ',WordToHexASCII(L*16),'  ');
                    Inc(L);
                    FOR J:= 1 TO 8 DO
                    BEGIN
                        Write(' ',ByteToHexASCII(RcvPacket[K]));
                        Inc(K);
                    END;
                    Write('  ');
                    FOR J:= 1 TO 8 DO
                    BEGIN
                        Write(' ',ByteToHexASCII(RcvPacket[K]));
                        Inc(K);
                    END;
                    Write('    ');
                    I := K-16;
                    FOR J:= 1 TO 16 DO
                    BEGIN
                        CASE RcvRawData[I] OF
                            ' '..'~' : Write(RcvRawData[I])
                        ELSE
                            Write('.')
                        END;
                        Inc(I);
                    END;
                    WriteLn;
                END;
            END;
        END;
        K := 0;  L := 0;
        RcvLength   := 0;
        PacketCount := 0;

    UNTIL (Key = #27);
    Key := #0;
    WindowClose;
END;
```

{

PROCEDURE DisplayTrafficGraph;	
Description :	*Show traffic level on setting time*

}

```
PROCEDURE DisplayTrafficGraph;
VAR  Window          : WindowType;
     PacketCnt       : Word;
     CumTraffic      : LongInt;
     CurTraffic      : Real;
     TrafficLevel    : Real;
     tKey            : Char;
BEGIN
     Window.bX:=1; Window.bY:=3; Window.bW:=80; Window.bH:=22;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=WHITE;
     Window.bTitleAttr:=YELLOW;
     Window.bBStyle:=2;
     Window.sTitle:=' Graph Represent Traffic on this Network Segment ';
```

เอกสารนี้ใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    WindowOpen(Window);
    HideCursor;
    TextColor(WHITE);

    PacketCnt  := 0;
    CumTraffic := 0;

    Timer1.Init(1);
    IF Interval <> 1 THEN Timer2.Init(Interval);
    Timer1.Start;

    REPEAT
          IF KeyPressed THEN Key := ReadKey;

          IF (PacketCount = 1) THEN
          BEGIN
               Move(Rcvpacket,RcvHeader,SizeOf(RcvHeader));
               IF IsPacketVerify(RcvHeader) THEN
               BEGIN
                  Inc(CumTraffic,RcvLength);
                  RcvLength  := 0;
                  PacketCount := 0;
               END;
          END;

          IF Timer1.Timeout THEN
          BEGIN
               TrafficLevel := (CumTraffic / (MaxEthernetCapacity)*75);
               CurTraffic   := (TrafficLevel / 75) * 100;
               TextColor(YELLOW);
               Write(' ',GetSystemTime,' |');
               TextColor(WHITE);
               FOR I := 1 TO Trunc(TrafficLevel) DO
               BEGIN
                    Write('■');
               END;
               WriteLn(' ',Trunc(CurTraffic),'%');
               CumTraffic := 0;
               IF Interval <> 1 THEN
               BEGIN
                    Timer2.Start;
                    While NOT(Timer2.Timeout) DO
                    BEGIN
                         IF Keypressed THEN tKey := ReadKey;
                         IF tKey = #27 THEN
                         BEGIN
                              WindowClose;
                              EXIT;
                         END;
                    END;
               END;
               Timer1.Start;
          END;

    UNTIL (Key = #27);
    Key := #0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    WindowClose;
END;
```

{

PROCEDURE DisplayGraphDLC;	
Description :	Show Graph Utilized by IEEE802.3 VS Ethernet II

}

```
PROCEDURE DisplayGraphDLC;
VAR  Window          : WindowType;
     tLongInt        : LongInt;
     IEEE802Count    : LongInt;
     EthernetIICount : LongInt;
     IEEE802Percent  : Real;
     EthernetIIPercent:Real;
     IEEE802High     : Integer;
     EthernetIIHigh  : Integer;
     tWord           : Word;
BEGIN
     Window.bX:=1; Window.bY:=3; Window.bW:=80; Window.bH:=22;
     Window.bAttr := YELLOW;
     Window.bBorderAttr:=WHITE;
     Window.bTitleAttr:=YELLOW;
     Window.bBStyle:=2;
     Window.sTitle:=' On-line Packet Information ';
     WindowOpen(Window);
     HideCursor;
     TextColor(WHITE);

     IF NOT(BufferFill) THEN
     BEGIN
          WindowClose;
          Message(' Error ','Packet must be captured to buffer');
          EXIT;
     END;

     IEEE802Count := 0;
     EthernetIICount := 0;

     FOR I := 0 TO BufferCount-1 DO
     BEGIN
          IF (SwapWord(PacketBuffer[I].Header.TypeLen) <= $5EE) THEN
             BEGIN
                  IEEE802Count := IEEE802Count + PacketBuffer[I].Header.TypeLen;
             END
          ELSE
             BEGIN
                  Move(PacketBuffer[I],RcvEtherLength,SizeOf(PacketBuffer[I]));
                  Move(RcvEtherLength.Length,tWord,SizeOf(tWord));
                  EthernetIICount := EthernetIICount + tWord;
             END;
     END;

     tLongInt := IEEE802Count + EthernetIICount + 1;

     IEEE802Percent := TRUNC(IEEE802Count * 100 / tLongInt);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    EthernetIIPercent := TRUNC(EthernetIICount * 100 / tLongInt);

    IEEE802High       := TRUNC(IEEE802Percent/10);
    EthernetIIHigh    := TRUNC(EthernetIIPercent/10);

    IF (IEEE802Percent+EthernetIIPercent <> 0) THEN
    BEGIN
      IF ((IEEE802Percent+EthernetIIPercent) <> 100) THEN
      BEGIN
         IF (IEEE802Percent >= EthernetIIPercent) THEN
             IEEE802Percent := IEEE802Percent +
                               (100 - IEEE802Percent - EthernetIIPercent)
         ELSE
             EthernetIIPercent := EthernetIIPercent +
                                  (100 - IEEE802Percent - EthernetIIPercent);
      End;
    End;
    GotoXY(5,18);
    FOR I:= 0 TO 68 DO Write('-');
    GotoXY(16,19);Write('IEEE 802.3                                    Ethernet');
    PutBar(15,17,12,IEEE802High,YELLOW,IEEE802Percent);
    PutBar(50,17,12,EthernetIIhigh,LIGHTGREEN,EthernetIIPercent);

    readkey;
    WindowClose;
END;
```

{

PROCEDURE DisplayGraphProtocol;	
Description :	*Show Graph for protocols utilized*

}

```
PROCEDURE DisplayGraphProtocol;
VAR    Window          : WindowType;
       IEEE802Count    : LongInt;
       EthernetIIcount : LongInt;
       ProtocolCount   : ARRAY[0..7] OF LongInt;
       ProtocolPercent : ARRAY[0..7] OF Real;
       ProtocolHigh    : ARRAY[0..7] OF Integer;
       tWord,ttWord    : Word;
       tLongint        : LongInt;
       tByte           : Byte;
BEGIN
    Window.bX:=1; Window.bY:=3; Window.bW:=80; Window.bH:=22;
    Window.bAttr := YELLOW;
    Window.bBorderAttr:=WHITE;
    Window.bTitleAttr:=YELLOW;
    Window.bBStyle:=2;
    Window.sTitle:=' Graph Represent Media Utilize in Data Link Layer ';
    WindowOpen(Window);
    HideCursor;
    TextColor(WHITE);

    IF NOT(BufferFill) THEN
    BEGIN
        WindowClose;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      Message(' Error ','Packet must be captured to buffer');
      EXIT;
  END;

  IEEE802Count := 0;
  EthernetIICount := 0;

  FOR I:= 0 TO 7 DO
      ProtocolCount[I]  :=  0;

  FOR I := 0 TO BufferCount-1 DO
  BEGIN
       IF (SwapWord(PacketBuffer[I].Header.TypeLen) <= $5EE) THEN
          BEGIN
               Move(PacketBuffer[I],RcvIEEE802,SizeOf(RcvIEEE802));
               Move(RcvIEEE802.PacketType,tByte,1);
               CASE  tByte OF
                 4   :  ProtocolCount[0]  := ProtocolCount[0] +
                                             PacketBuffer[I].Header.TypeLen;
                 5   :  ProtocolCount[1]  := ProtocolCount[1] +
                                             PacketBuffer[I].Header.TypeLen;
                 17  :  ProtocolCount[2]  := ProtocolCount[2] +
                                             PacketBuffer[I].Header.TypeLen;
               ELSE
                 ProtocolCount[3]    := ProtocolCount[3] +
                                        PacketBuffer[I].Header.TypeLen;
               END;
               IEEE802Count := IEEE802Count + PacketBuffer[I].Header.TypeLen;
          END
       ELSE
          BEGIN
          IF  (PacketBuffer[I].Header.TypeLen =  $8) THEN
          BEGIN
               Move(PacketBuffer[I],RcvEtherLength,SizeOf(PacketBuffer[I]));
               Move(RcvEtherLength.Length,ttWord,SizeOf(ttWord));
               Move(PacketBuffer[I],RcvEthernetII,SizeOf(RcvEtherII));
               Move(RcvEthernetII.Protocol,tByte,1);
               IF (tByte = 6) THEN
               BEGIN
                    Move(RcvEthernetII.DestPort,tWord,SizeOf(tWord));
                    tWord := SwapWord(tWord);
                    CASE  tWord OF
                      20..21  :  ProtocolCount[4] :=  ProtocolCount[4] + ttWord;
                      23      :  ProtocolCount[5] :=  ProtocolCount[5] + ttWord;
                      25      :  ProtocolCount[6] :=  ProtocolCount[6] + ttWord;
                    ELSE
                       ProtocolCount[7] :=  ProtocolCount[7] + ttWord;
                    END;
               END;
               IF (tByte <> 6) THEN ProtocolCount[7] :=  ProtocolCount[7] + ttWord;
               EthernetIICount := EthernetIICount + ttWord;
          END;
          END;
  END;

  tLongInt := IEEE802Count + EthernetIICount + 1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    FOR I:= 0 TO 7 DO
        ProtocolPercent[I] := TRUNC(ProtocolCount[I] * 100 /tLongint);
    FOR I := 0 TO 7 DO
        ProtocolHigh[I] := TRUNC(ProtocolPercent[I]/10);

    GotoXY(5,17);
    FOR I:= 0 TO 68 DO Write('─');
    GotoXY(39,17);Write('┬');
    GotoXY(6,18);Write('PEP      SPP      NCP    OTHERS │ FTP     TELNET    SMTP    OTHE
    GotoXY(5,19);FOR I:= 0 TO 68 DO Write('─');
    GotoXY(39,19);Write('┼');
    GotoXY(6,20);Write('          IEEE 802.3              │          Ethernet II');
    FOR I:= 0 TO 7 DO
        PutBar(6+I*9,16,4,ProtocolHigh[I],I+1,ProtocolPercent[I]);
    readkey;
    WindowClose;
END;
```

{

PROCEDURE MakeSendPacket;	
Description :	Create Simulate Send Packet

}

```
PROCEDURE MakeSendPacket;
BEGIN
    GetMyEthernetAddr;
    For I := 0 TO 5 Do
        SendPacket.Destaddr[I] := $AA;
    SendPacket.SourceAddr      := MyEthernetAddr;
    SendPacket.TypeLen         := $5E00;
    tStr := 'this a test message generate from Ethernet Monitor Version 1.00 Revision A !!!
    Move(tStr,SendPacket.Data,76);
    SendPacket.Data[1] := 'T';
END;
```

{

PROCEDURE InitialEnvironment;	
Description :	Intialize Start Environment for program

}

```
PROCEDURE InitialEnvironment;
BEGIN
    {Menu Initial}
    bBStyle  := 1;
    MenuNormAttr := $2f;
    MenuBrightAttr := $24;
    MenuSelectAttr := $70;
    MenuSelectBrightAttr := $7e;
    MenuBorderAttr := $20;

    {Window Initial}
    Window.bX:=0; Window.bY:=0; Window.bW:=79; Window.bH:=24;
    Window.bAttr := 7;
    Window.bBorderAttr:=0;
    Window.bTitleAttr:=0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Window.bBStyle:=1;
    Window.sTitle:='';

    {Station Initial}
    AllStation      := TRUE;
    AllSource       := TRUE;
    AllDestination  := TRUE;
    AllPacket       := TRUE;
    IsEtherII       := TRUE;
    IsIEEE802       := TRUE;
    Mode            := 1;
    BufferFill      := FALSE;

    {Timer Initial}
    Interval        := 1;
END;

{
        MAIN PROGRAM FOR Ethernet Monitor VERSION 1.00
}
BEGIN

    ClrScr;
    HideCursor;

    pktDriver.ScanForPktDriver;
    InitPktDriver;
    Delay(1500);
    ClrScr;

    InitialEnvironment;
    MakeSendPacket;
    TextColor($e);
    TextBackGround($6);
    GotoXY(1,1);Write('     Ethernet Monitor Version 1.00          ');

    LoadBackGround;
    TextBackGround(BLACK);
    About(FALSE);

    REPEAT
        Menu.Init(1,2);
        Menu.Add('&Online' , 1,'Online Monitoring Packet',0);

        Menu.Add('&Data Link'          ,11,'Display real time Data Link
                                            Layer Packet Header Infomation',1);
        Menu.Add('&Packet Information',12,'Display real time Packet Information',1);
        Menu.Add('&Raw Packet'         ,13,'Display real time Raw Packet Infomation',1);
        Menu.Add('&Traffic Graph'      ,14,'Display Graph for traffic',1);
        Menu.AddBar(1);
        Menu.Add('&Set Time Interval' ,15,'Set Time Interval for Traffic Graph',1);

        Menu.Add('&2  Seconds' ,151,'Display Every 2 Seconds',15);
        IF Interval = 1 THEN Menu.SetState(151,$01) ELSE Menu.SetState(151,$0);
        Menu.Add('&1  Minute'  ,152,'Display Every  1 Minute',15);
        IF Interval = 57 THEN Menu.SetState(152,$01) ELSE Menu.SetState(152,$0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
IF Interval = 897 THEN Menu.SetState(153,$01) ELSE Menu.SetState(153,$0);
Menu.Add('&30 Minutes' ,154,'Display Every  30 Minutes',15);
IF Interval = 1797 THEN Menu.SetState(154,$01) ELSE Menu.SetState(154,$0);
Menu.Add('&60 Minutes' ,155,'Display Every  60 Minutes',15);
IF Interval = 3597 THEN Menu.SetState(155,$01) ELSE Menu.SetState(155,$0);

Menu.Add('O&ffline'  , 2,'Offline Monitoring Packet',0);
Menu.Add('&Graph'    ,21,'Graph represent many infomation',2);
Menu.Add('&Load File',22,'Load File to Use to Views Packet',2);

Menu.Add('&Data Link'  ,211,
         'Graph represent utilize of medea IEEE802.3 VS Ethernet',21);
Menu.Add('&Protocol'   ,212,'Graph represent percentage of protocols',21);

Menu.Add('&Configure', 3,'Configure Packet',0);
Menu.Add('&Packet'   ,31,'Configure Packet Type',3);
Menu.Add('&Station'  ,32,'Configure Station Address',3);
Menu.AddBar(3);
Menu.Add('S&tatus'   ,33,'Display Status of Packets to Monitor',3);

Menu.Add('&IEEE 802.3' ,311,'Only IEEE 802.3 Packet',31);
IF IsIEEE802 THEN Menu.SetState(311,$01) ELSE Menu.SetState(311,$0);
Menu.Add('&Ethernet '  ,312,'Only Ethernet Packet',31);
IF IsEtherII THEN Menu.SetState(312,$01) ELSE Menu.SetState(312,$0);
Menu.AddBar(31);
Menu.Add('&All'        ,313,'Display all Packet',31);
IF AllPacket THEN Menu.SetState(313,$01) ELSE Menu.SetState(313,$0);

Menu.Add('S&ource'     ,321,'Specific Source Station Address',32);
Menu.Add('&Destination',322,'Specific Destination Station Address',32);
Menu.AddBar(32);
Menu.Add('&All'        ,323,'All Source & Destination Address',32);
IF AllStation THEN Menu.SetState(323,$01) ELSE Menu.SetState(323,$0);

Menu.Add('A&ll'   ,3211,'All Source Station',321);
IF AllSource THEN Menu.SetState(3211,$01) ELSE Menu.SetState(3211,$0);
Menu.Add('Ass&ign',3212,'Only Assign Source Station',321);
IF AllSource THEN Menu.SetState(3212,$0) ELSE Menu.SetState(3212,$01);

Menu.Add('A&ll'   ,3221,'All Destination Station',322);
IF AllDestination THEN Menu.SetState(3221,$01) ELSE Menu.SetState(3221,$0);
Menu.Add('Ass&ign',3222,'Only Assign Destinattion Station',322);
IF AllDestination THEN Menu.SetState(3222,$0) ELSE Menu.SetState(3222,$01);

Menu.Add('C&apture'    , 4,'Capture Packets to Buffer',0);
Menu.Add('&File'       ,41,'Capture to file',4);

Menu.Add('&Save'   ,411,'Capture to default file name',41);
Menu.Add('Sa&ve as',412,'Capture to assign file name',41);

Menu.Add('&Buffer'     ,42,'Capture to memory buffer',4);
Menu.Add('&View'       ,43,'View Saved file',4);
Menu.AddBar(4);
Menu.Add('&Save Buffer',44,'Save Buffer to file',4);

Menu.Add('&Generate'   , 5,'Generate Packet to Networks',0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Menu.Add('&Send'           ,51,'Send Packet to network',5);
Menu.Add('&Compose Packet',52,'Config send packet',5);

Menu.Add('&Destination',521,'Destination Address to Send',52);
Menu.Add('&Source'     ,522,'Source Address to Emulate',52);
Menu.Add('Data &Field' ,523,'Data to send',52);

Menu.Add('Too&l'      , 6,'Miscellaneous Tools',0);
Menu.Add('L&ist'      ,61,'Text Viewer',6);
Menu.Add('&Q Edit'    ,62,'Text Editor',6);

Menu.Add('&Help'    , 7,'Help Ethernet Monitor',0);
Menu.Add('&Help'    ,71,'Help Ethernet Monitor',7);
Menu.Add('&Warning',72,'Warning for use Ethernet Monitor !',7);
Menu.AddBar(7);
Menu.Add('Abo&ut',73,'About Ethernet Monitor',7);

Menu.Add('E&xit',8,'Exit Program',0);

MenuSelect := Menu.Get(C);

Menu.Done;

TextColor(WHITE);
TextBackGround(BLACK);

Case MenuSelect OF
  1       :;
  11      :       DisplayDataLink;
  12      :       DisplayAllInformation;
  13      :       DisplayRawPacket;
  14      :       DisplayTrafficGraph;
  15      :;
  151     :       SetTimeInterval(1);
  152     :       SetTimeInterval(57);
  153     :       SetTimeInterval(897);
  154     :       SetTimeInterval(1797);
  155     :       SetTimeInterval(3597);
  2       :;
  21      :;
  211     :       DisplayGraphDLC;
  212     :       DisplayGraphProtocol;
  22      :       LoadBufferFile;
  3       :;
  31      :;
  311     :       SetIEEE802;
  312     :       SetEtherII;
  313     :       SetAllPacket;
  32      :;
  321     :;
  3211    :       SetAllSourceAddr(TRUE);
  3212    :       SetAllSourceAddr(FALSE);
  322     :;
  3221    :       SetAllDestinationAddr(TRUE);
  3222    :       SetAllDestinationAddr(FALSE);
  323     :       SetAllStation(TRUE);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            33      :       DisplayStatus(FALSE);
            4       :;
            41      :;
            411     :       SetDefaultFileName;
            412     :       SetFileName;
            42      :       CaptureToBuffer;
            43      :       ExecFile('LIST.COM',FileName);
            44      :       BufferToDisk;
            5       :;
            51      :       GenPacket;
            52      :;
            521     :       SetSendDestinationAddr;
            522     :       SetSendSourceAddr;
            523     :       SetSendData;
            6       :;
            61      :       ExecFile('LIST.COM','');
            62      :       ExecFile('Q.EXE','');
            7       :;
            71      :       ExecFile('LIST.COM','HELP.TXT');
            72      :       Warning;
            73      :       About(TRUE);
            8       :       ExitProgram;
        END;
    UNTIL False;
END.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TIMER.PAS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
```

Filename : TIMER.PAS	Program / Unit : [U]
Description :	
Compiler : Turbo Pascal 6.0	
OS-Version : MS-DOS 5.0	
Last edit : 27-JAN-93	
Author : Oliver Rehmann	
Copyright : Oliver Rehmann	

```
}

UNIT TIMER;

INTERFACE

USES DOS;

TYPE  _TIMER = OBJECT
                 Starttime   : LONGINT;
                 Duration    : LONGINT;
                 _Timeout    : BOOLEAN;

                 CONSTRUCTOR Init(tDuration : LONGINT);
                 PROCEDURE   Start;
                 FUNCTION    Timeout : BOOLEAN;

               END;

FUNCTION  GetSystemTime : String;

IMPLEMENTATION

FUNCTION  GetSystemTime : String;
VAR tmp     : String;
    Time    : String;
    TimeTmp : Byte;
    Hour,Minute,Second,Sec100 : Word;
BEGIN
     GetTime(Hour,Minute,Second,Sec100);
     Str(Hour:2,tmp);
       If Pos(' ',tmp)<>0 then Time :=   '0'+tmp[2] else Time :=      tmp;
     Time := Time+':';
     Str(Minute:2,tmp);
       If Pos(' ',tmp)<>0 then Time := Time+'0'+tmp[2] else Time := Time+tmp;
     Time := Time+':';
     Str(Second:2,tmp);
       If Pos(' ',tmp)<>0 then Time := Time+'0'+tmp[2] else Time := Time+tmp;
     TimeTmp := Hour; GetSystemTime := Time
END;

FUNCTION GetSysTime : LONGINT;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
VAR Hour,Minute,Second,Sec100 : Word;
    tSysTime                  : LONGINT;
BEGIN
  tSysTime    := 0;
  GetTime(Hour,Minute,Second,Sec100);
  tSysTime    := Hour*3600;                { each hour has 3600 seconds }
  tSysTime    := tSysTime + (Minute*60);   { each min  has 60 seconds   }
  tSysTime    := tSysTime + Second;
  GetSysTime := tSysTime;
END;

CONSTRUCTOR _TIMER.Init(tDuration : LONGINT);
BEGIN
  Duration := tDuration;
  _Timeout := FALSE;
END;

PROCEDURE _TIMER.Start;
BEGIN
  _Timeout := FALSE;
  Starttime := GetSysTime;
END;

FUNCTION _TIMER.Timeout : BOOLEAN;
BEGIN
  IF ((Starttime + Duration) < GetSysTime) THEN _Timeout := TRUE;
  Timeout := _Timeout;
END;

BEGIN
END.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PKTDRVR.PAS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
UNIT PKTDRVR;
{
```

Filename : *PKTDRVR.PAS*	*Program / Unit : [U]*
Description : *Turbo Pascal*	
Object to interface with Crynrware packet drivers.	
Compiler : *Turbo Pascal 7.0*	
OS-Version : *MS-DOS 6.0*	
Last edit : *08-Oct-93*	
Version : *1.0*	
Author : *Oliver Rehmann*	
Copyright : *(C) 1993 Oliver Rehmann*	
Released to public domain.	
The author can not be held responsible for any damages resulting	
from the use of this software.	

```
}

INTERFACE

USES DOS,OBJECTS;

{$I PACKET.INC}

CONST
     Pkt_Sig  : String[08] = 'PKT DRVR';
     ParamLen : Byte       = 14;

TYPE TPKTSTATUS    = (NO_PKTDRVR,INITIALIZED,NOT_INITIALIZED);
     TACCESSTYPE   = RECORD
                       if_class      : Byte;    { Interface class  }
                       if_type       : Word;    { Interface Type   }
                       if_number     : Byte;    { Interface number }
                       type_         : Pointer;
                       typelen       : Word;    { length of type_, set to 0 if
                                                  you want to receive all pkts }
                       receiver      : Pointer; { receive handler }
                     END;

       TPKTPARAMS   = RECORD
                       major_rev     : Byte; { Major revision ID of packet specs }
                       minor_rev     : Byte; { Minor revision ID of packet specs }
                       length        : Byte; { Length of structure in Bytes      }
                       addr_len      : Byte; { Length of a MAC address           }
                       mtu           : Word; { MTU, including MAC headers        }
                       multicast_aval: Word; { buffer size for multicast addr.   }
                       rcv_bufs      : Word; { (# of back-to-back MTU rcvs) - 1   }
                       xmt_bufs      : Word; { (# of successive xmits) - 1       }
                       int_num       : Word; { Interrupt # to hook for post-EOI
                                               processing, 0 == none }
                     END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
TDRVRINFO    = RECORD
                 Version       : Word; { Packet driver version   }
                 Class         : Byte; { Driver class  }
                 Type_         : Word; { Driver type   }
                 Number        : Byte; { Driver number }
                 pName         : Pointer;
                 Functionality : Byte; { How good is this driver }
               END;

TSTATISTICS = RECORD
                 packets_in    : LongInt;
                 packets_out   : LongInt;
                 bytes_in      : LongInt;
                 bytes_out     : LongInt;
                 errors_in     : LongInt;
                 errors_out    : LongInt;
                 packets_lost  : LongInt;
               END;

TPKTDRVR = OBJECT(TOBJECT)

PRIVATE          pktInt        : Integer;
                 pktHandle     : Integer;
                 pktRecvHandler : Pointer;
                 pktStatus     : TPKTSTATUS;
                 pktError      : Byte;
                 pktRegs       : Registers;

                 pktAccessInfo : TACCESSTYPE;

                 PROCEDURE  TestForPktDriver;

PUBLIC           CONSTRUCTOR Init(IntNo : Integer);
                 DESTRUCTOR  Done; VIRTUAL;

                 PROCEDURE  ScanForPktDriver;

                 FUNCTION   GetStatus                                  : TPKTSTATUS;
                 FUNCTION   GetError                                   : Byte;
                 FUNCTION   GetHandle                                  : Word;

                 PROCEDURE  GetAccessType  (VAR pktAccessType : TACCESSTYPE);
                 PROCEDURE  DriverInfo     (VAR pktInfo       : TDRVRINFO  );

                 PROCEDURE  AccessType     (VAR pktAccessType : TACCESSTYPE);
                 PROCEDURE  ReleaseType;
                 PROCEDURE  TerminateDriver;

                 PROCEDURE  GetAddress     (Buffer : Pointer;BufLen : Word;
                                            VAR BufCopied : Word);

                 PROCEDURE  ResetInterface;
                 PROCEDURE  GetParameters  (VAR pktParams : TPKTPARAMS);

                 PROCEDURE  SendPkt        (Buffer : Pointer;BufLen : Word );
                 PROCEDURE  As_SendPkt     (Buffer : Pointer;BufLen : Word;
                                            Upcall : Pointer );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    PROCEDURE   SetRCVmode      (Mode   : Word);
    FUNCTION    GetRCVmode              : Word;

    PROCEDURE   SetMulticastList(VAR mcList : Pointer; VAR mcLen : Word);
    PROCEDURE   GetMulticastList(VAR mcList : Pointer; VAR mcLen : Word);

    PROCEDURE   GetStatistics   (VAR pktStatistics : TSTATISTICS       );
    PROCEDURE   SetAddress      (Address : Pointer; VAR AddrLen  : Word);

    END;

IMPLEMENTATION
```

{

CONSTRUCTOR Init	
Description :	*Initializes packet driver object*
Creation date :	*13-SEPT-93*

}

```
CONSTRUCTOR TPKTDRVR.Init(IntNo : Integer);
BEGIN
  Inherited Init;

  pktInt    := IntNo;
  pktStatus := NOT_INITIALIZED;
  FillChar(pktAccessInfo,SizeOf(pktAccessInfo),#00);

  TestForPktDriver;
END;
```

{

DESTRUCTOR Done	
Description :	*Calls inherited destructor of TOBJECT*
Creation date :	*13-SEPT-93*

}

```
DESTRUCTOR TPKTDRVR.Done;
BEGIN

  { Release allocated handle }
  IF (pktStatus = INITIALIZED) THEN
  BEGIN
    ReleaseType;
  END;

  Inherited Done;
END;
```

{

FUNCTION GetStatus : TPKTSTATUS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Description :	Returns current object status
Creation date :	13-SEPT-93

```
FUNCTION TPKTDRVR.GetStatus : TPKTSTATUS;
BEGIN
  GetStatus := pktStatus;
END;
```

FUNCTION GetAccessType	
Description :	Returns information about type of interface, class, interface number.
Creation date :	13-SEPT-93

```
PROCEDURE TPKTDRVR.GetAccessType(VAR pktAccessType : TACCESSTYPE);
BEGIN
  pktAccessType := pktAccessInfo;
END;
```

PROCEDURE TestForPktDriver;	
Description :	Tests if the assigned interrupt points to a valid packet driver.
Creation date :	13-SEPT-93

```
PROCEDURE TPKTDRVR.TestForPktDriver;
VAR tPointer  : Pointer;
    Signature : String[08];
    I         : Integer;
BEGIN
  Signature := '';
  GetIntVec(pktInt,tPointer);
  FOR I := 3 TO 10 DO
  BEGIN
    Signature := Signature + Chr(Mem[Seg(tPointer^):Ofs(tPointer^)+I]);
  END;
  IF (POS(Pkt_Sig,Signature) = 0) THEN
    pktStatus := NO_PKTDRVR
  ELSE
    pktStatus := INITIALIZED;
END;
```

PROCEDURE ScanForPktDriver	
Description :	Scans interrupts ($60-$7F) for a packet driver. Stops if it has found a valid driver.
Creation date :	13-SEPT-93

เอกสารนี้ใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PROCEDURE TPKTDRVR.ScanForPktDriver;
VAR I : Integer;
BEGIN
  I := $60; { Lower range of possible pktdrvr interrupt }
  REPEAT
    pktInt := I;
    TestForPktDriver;
    Inc(I);
  UNTIL (I = $80) OR (pktStatus = INITIALIZED);
END;
```

{

PROCEDURE DriverInfo	
Description :	Returns some information on the assigned packet driver
Creation date :	13-SEPT-93

}

```
PROCEDURE TPKTDRVR.DriverInfo(VAR pktInfo : TDRVRINFO);
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := DRIVER_INFO;
    AL := $FF;
    BX := pktHandle;
    Intr(pktInt,pktRegs); { Call Packet Driver }
    IF (pktRegs.Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
    BEGIN
      pktError := 0;
      IF (pktError = NO_ERROR) THEN
      BEGIN
        pktInfo.Version      := BX;
        pktInfo.Class        := CH;
        pktInfo.Type_        := DX;
        pktInfo.Number       := CL;
        pktInfo.pName        := Ptr(DS,SI);
        pktInfo.Functionality := AL;
      END;
    END;
  END;
END;
```

{

PROCEDURE AccessType	
Description :	Accesses the packet driver.
Creation date :	13-SEPT-93

}

```
PROCEDURE TPKTDRVR.AccessType(VAR pktAccessType : TACCESSTYPE);
BEGIN
  WITH pktRegs DO
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  BEGIN
    AH := ACCESS_TYPE;
    AL := pktAccessType.if_class;
    BX := pktAccessType.if_type;
    CX := pktAccessType.typelen;
    DL := pktAccessType.if_number;
    DS := Seg(pktAccessType.type_^);
    SI := Ofs(pktAccessType.type_^);
    ES := Seg(pktAccessType.receiver^);
    DI := Ofs(pktAccessType.receiver^);
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError    := DH
    ELSE
    BEGIN
      pktError      := 0;
      pktHandle     := AX;
      pktAccessInfo := pktAccessType;
    END;
  END;
END;
```

{

PROCEDURE ReleaseType	
Description :	*Releases a specific type handle.*
Creation date :	*13-SEPT-93*

}

```
PROCEDURE TPKTDRVR.ReleaseType;
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := RELEASE_TYPE;
    BX := pktHandle;
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
      pktError := 0;
  END;
END;
```

{

PROCEDURE SendPkt	
Description :	*Sends a data packet by accessing the packet driver.*
Creation date :	*13-SEPT-93*

}

```
PROCEDURE TPKTDRVR.SendPkt(Buffer : Pointer;BufLen : Word);
BEGIN
  WITH pktRegs DO
  BEGIN
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    AH := SEND_PKT;
    CX := BufLen;
    DS := Seg(Buffer^);
    ES := DS;
    SI := Ofs(Buffer^);
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
      pktError := 0;
  END;
END;
```

{

PROCEDURE TerminateDriver	
Description :	*Terminates the Driver associated with pktHandle*
Creation date :	*13-SEPT-93*

}

```
PROCEDURE   TPKTDRVR.TerminateDriver;
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := TERMINATE;
    BX := pktHandle;
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
      pktError := 0;
  END;
END;
```

{

PROCEDURE GetAddress	
Description :	*Gets the hardware address from the driver.*
Creation date :	*13-SEPT-93*

}

```
PROCEDURE TPKTDRVR.GetAddress (Buffer : Pointer;BufLen : Word; VAR BufCopied : Word);
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := GET_ADDRESS;
    BX := pktHandle;
    CX := BufLen;
    ES := Seg(Buffer^);
    DI := Ofs(Buffer^);
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError  := DH
    ELSE
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    BEGIN
      pktError  := 0;
      BufCopied := CX;
    END;
  END;
END;
```

{

PROCEDURE ResetInterface	
Description :	*Does what it says.*
Creation date :	*13-SEPT-93*

}

```
PROCEDURE TPKTDRVR.ResetInterface;
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := RESET_INTERFACE;
    BX := pktHandle;
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
      pktError := 0;
  END;
END;
```

{

PROCEDURE GetParameters	
Description :	*Gets specific parameters from the driver.* *Not all drivers support this function.*
Creation date :	*13-SEPT-93*

}

```
PROCEDURE TPKTDRVR.GetParameters(VAR pktParams : TPKTPARAMS);
VAR b : Byte;
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := GET_PARAMETERS;
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
    BEGIN
      pktError := 0;
      FOR b := 0 TO ParamLen-1 DO  { Copy contents of structure }
        Mem[Seg(pktParams):Ofs(PktParams)+b] := Mem[ES:DI+b];
    END;
  END;
END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

{

PROCEDURE AsSendPkt	
Description :	Sends a data packet by accessing the packet driver. Upcall is called when order was placed.
Creation date :	17-SEPT-93

}

```
PROCEDURE TPKTDRVR.As_SendPkt(Buffer : Pointer;BufLen : Word;Upcall : Pointer);
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := AS_SEND_PKT;
    CX := BufLen;
    DS := Seg(Buffer);
    SI := Ofs(Buffer);
    ES := Seg(Upcall^);
    DI := Ofs(Upcall^);
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
      pktError := 0;
  END;
END;
```

{

PROCEDURE SetRCVmode	
Description :	Sets the receive mode of the driver. For meaning of mode see "packet_d.109"
Creation date :	17-SEPT-93

}

```
PROCEDURE TPKTDRVR.SetRCVmode(Mode : Word);
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := SET_RCV_MODE;
    BX := pktHandle;
    CX := Mode;
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
      pktError := 0;
  END;
END;
```

{

FUNCTION GetRCVmode : Word	
Description :	Gets the receive mode from the driver. For meaning of mode see "packet_d.109"
Creation date :	17-SEPT-93

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
FUNCTION TPKTDRVR.GetRCVmode : Word;
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := GET_RCV_MODE;
    BX := pktHandle;
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError   := DH
    ELSE
    BEGIN
      pktError   := 0;
      GetRCVmode := AX;
    END;
  END;
END;
```

{

PROCEDURE SetMulticastList	
Description :	*Assigns a multicast list to the driver.*
Creation date :	*17-SEPT-93*

}

```
PROCEDURE TPKTDRVR.SetMulticastList(VAR mcList : Pointer; VAR mcLen : Word);
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := SET_MULTICAST_LIST;
    CX := mcLen;
    ES := Seg(mcList^);
    DI := Ofs(mcList^);
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
      pktError := 0;
  END;
END;
```

{

PROCEDURE GetMulticastList	
Description :	*Retrieves a pointer to a multicast list.*
Creation date :	*17-SEPT-93*

}

```
PROCEDURE TPKTDRVR.GetMulticastList(VAR mcList : Pointer; VAR mcLen : Word);
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := GET_MULTICAST_LIST;
    Intr(pktInt,pktRegs);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
    BEGIN
      pktError := 0;
      mcList   := Ptr(ES,DI);
      mcLen    := CX;
    END;
  END;
END;
```

{

PROCEDURE GetStatistics	
Description :	*Retrieves statistics about received and sent packets, bytes, errors.*
Creation date :	*17-SEPT-93*

}

```
PROCEDURE TPKTDRVR.GetStatistics(VAR pktStatistics : TSTATISTICS);
VAR b : Byte;
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := GET_STATISTICS;
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
    ELSE
    BEGIN
      pktError := 0;
      FOR b := 0 TO SizeOf(TSTATISTICS)-1 DO  { Copy contents of structure }
        Mem[Seg(pktStatistics):Ofs(pktStatistics)+b] := Mem[DS:SI+b];
    END;
  END;
END;
```

{

PROCEDURE SetAddress	
Description :	
Creation date :	*17-SEPT-93*

}

```
PROCEDURE TPKTDRVR.SetAddress(Address : Pointer; VAR AddrLen : Word);
BEGIN
  WITH pktRegs DO
  BEGIN
    AH := SET_ADDRESS;
    CX := AddrLen;
    ES := Seg(Address^);
    DI := Ofs(Address^);
    Intr(pktInt,pktRegs);
    IF (Flags AND Carry_Flag) = Carry_Flag THEN
      pktError := DH
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    ELSE
    BEGIN
      pktError := 0;
      AddrLen  := CX;
    END;
  END;
END;
```

{

PROCEDURE GetError	
Description :	*Returns the last error. 0 means NO ERROR*
Creation date :	*17-SEPT-93*

}

```
FUNCTION TPKTDRVR.GetError  : Byte;
BEGIN
  GetError := pktError;
END;
```

{

PROCEDURE GetHandle	
Description :	*Returns the accessed handle.*
Creation date :	*17-SEPT-93*

}

```
FUNCTION TPKTDRVR.GetHandle : Word;
BEGIN
  GetHandle := pktHandle;
END;

BEGIN
END.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
```

Filename : PACKET.INC Description : Include file for constants	Program / Unit : [ ]
Compiler : TURBO PASCAL 7.0 OS-Version : MS-DOS 6.0 Last edit : 17-SEPT-93	
Author : Oliver Rehmann Copyright : Oliver Rehmann	

```
}
{$IFNDEF CL_ETHERNET}

CONST

{ Packet driver interface classes }
CL_NONE          = 0;
CL_ETHERNET      = 1;
CL_PRONET_10     = 2;
CL_IEEE8025      = 3;
CL_OMNINET       = 4;
CL_APPLETALK     = 5;
CL_SERIAL_LINE   = 6;
CL_STARLAN       = 7;
CL_ARCNET        = 8;
CL_AX25          = 9;
CL_KISS          = 10;
CL_IEEE8023      = 11;
CL_FDDI          = 12;
CL_INTERNET_X25  = 13;
CL_LANSTAR       = 14;
CL_SLFP          = 15;
CL_NETROM        = 16;
NCLASS           = 17;

{ Packet driver interface types (not a complete list) }
TC500            = 1;
PC2000           = 10;
WD8003           = 14;
PC8250           = 15;
ANYTYPE          = $ffff;

{ Packet driver function call numbers. From Appendix B. }
DRIVER_INFO              = 1;
ACCESS_TYPE              = 2;
RELEASE_TYPE             = 3;
SEND_PKT                 = 4;
TERMINATE                = 5;
GET_ADDRESS              = 6;
RESET_INTERFACE          = 7;
GET_PARAMETERS           = 10;
AS_SEND_PKT              = 11;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SET_RCV_MODE            = 20;
GET_RCV_MODE            = 21;
SET_MULTICAST_LIST      = 22;
GET_MULTICAST_LIST      = 23;
GET_STATISTICS          = 24;
SET_ADDRESS             = 25;

{ Packet driver error return codes. From Appendix C. }

NO_ERROR        = 0;
BAD_HANDLE      = 1;    { invalid handle number }
NO_CLASS        = 2;    { no interfaces of specified class found }
NO_TYPE         = 3;    { no interfaces of specified type found }
NO_NUMBER       = 4;    { no interfaces of specified number found }
BAD_TYPE        = 5;    { bad packet type specified }
NO_MULTICAST    = 6;    { this interface does not support multicast }
CANT_TERMINATE  = 7;    { this packet driver cannot terminate }
BAD_MODE        = 8;    { an invalid receiver mode was specified }
NO_SPACE        = 9;    { operation failed because of insufficient space }
TYPE_INUSE      = 10;   { the type had previously been accessed, and not released }
BAD_COMMAND     = 11;   { the command was out of range, or not  implemented }
CANT_SEND       = 12;   { the packet couldn't be sent (usually  hardware error) }
CANT_SET        = 13;   { hardware address couldn't be changed (> 1 handle open) }
BAD_ADDRESS     = 14;   { hardware address has bad length or format }
CANT_RESET      = 15;   { couldn't reset interface (> 1 handle open) }

CARRY_FLAG      = 1;

{$ENDIF }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

NE2000.ASM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
;
;  Filename      : NE2000.ASM        Program / Driver
;  Description   : Packet Driver
;                  Assembly Code for NE2000 Compatible
;
;
;  Compiler      : TURBO ASSEMBLER
;  OS-Version    : MS-DOS 6.0
;  Last edit     : 17-SEPT-93
;
;  Author        : Oliver Rehmann
;  Copyright     : Oliver Rehmann
;

version equ     4
;History:76,1 0

;  The following people have contributed to this code: David Horne, Eric
;  Henderson, and Bob Clements.

;  Copyright, 1988-1992, Russell Nelson, Crynwr Software

;   This program is free software; you can redistribute it and/or modify
;   it under the terms of the GNU General Public License as published by
;   the Free Software Foundation, version 1.
;
;   This program is distributed in the hope that it will be useful,
;   but WITHOUT ANY WARRANTY; without even the implied warranty of
;   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;   GNU General Public License for more details.
;
;   You should have received a copy of the GNU General Public License
;   along with this program; if not, write to the Free Software
;   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

        include defs.asm

code.   segment word public
        assume  cs:code, ds:code

;*****************************************************************************
;
;       NE2000 controller board offsets
;       IO port definition (BASE in io_addr)
;*****************************************************************************
NE_DATAPORT     EQU     10h             ; NE2000 Port Window.
NE_RESET        EQU     1fh             ; Issue a read for reset
EN_OFF          equ     0h

        include 8390.inc

; Shared memory management parameters

SM_TSTART_PG    equ     040h    ; First page of TX buffer
SM_RSTART_PG    equ     046h    ; Starting page of RX ring
SM_RSTOP_PG     equ     080h    ; Last page +1 of RX ring
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pause_  macro
;       jmp     $+2
;
; The reason for the pause_ macro is to establish a minimum time between
; accesses to the card hardware. The assumption is that the fetch and execution
; of the jmp $+2 instruction will provide this time. In a fast cache machine
; this may be a false assumption. In a fast cache machine, there may be
; NO REAL TIME DIFFERENCE between the two I/O instruction streams below:
;
;       in      al,dx           in      al,dx
;       jmp     $+2
;       in      al,dx           in      al,dx
;
; To establish a minimum delay, an I/O instruction must be used. A good rule of
; thumb is that ISA I/O instructions take ~1.0 microseconds and MCA I/O
; instructions take ~0.5 microseconds. Reading the NMI Status Register (0x61)
; is a good way to pause on all machines.
;
; The National 8390 Chip (NIC) requires 4 bus clocks between successive
; chip selects (National DP8390 Data Sheet Addendum, June 1990 -- it took them
; long enough to figure this out and tell everyone) or the NIC behaves badly.
; Therefor one I/O instruction should be inserted between each successive
; NIC I/O instruction that could occur 'back - to - back' on a fast cache
; machine.
;   - gft - 910529
;
        push    ax
        in      al, 61h
        pop     ax
;
endm

reset_8390      macro
        loadport
        setport NE_RESET
        in      al,dx
        longpause
        out     dx,al           ; should set command 21, 80

        endm

terminate_board macro
        endm

        public  int_no, io_addr
int_no          db      2,0,0,0         ;must be four bytes long for get_number.
io_addr         dw      0300h,0         ; I/O address for card (jumpers)

        public  driver_class, driver_type, driver_name, driver_function, parameter_list
driver_class    db      BLUEBOOK, IEEE8023, 0           ;from the packet spec
driver_type     dw      54.             ;from the packet spec
driver_name     db      'NE2000',0      ;name of the driver.
driver_function db      2
parameter_list  label   byte
        db      1       ;major rev of packet driver
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        db      9       ;minor rev of packet driver
        db      14      ;length of parameter list
        db      EADDR_LEN       ;length of MAC-layer address
        dw      GIANT   ;MTU, including MAC headers
        dw      MAX_MULTICAST * EADDR_LEN       ;buffer size of multicast addrs
        dw      0       ;(# of back-to-back MTU rcvs) - 1
        dw      0       ;(# of successive xmits) - 1
int_num dw      0       ;interrupt # to hook for post-EOI
                        ;processing, 0 == none,

        extrn   is_186: byte            ;=0 if 808[68], =1 if 80[123]86.

;
;       Block input routine
;       CX = byte count, es:di = buffer location, ax = buffer address

        public  block_input
block_input:
        push    ax              ; save buffer address
        loadport
        setport EN_CCMD
        pause_
        mov     al,ENC_NODMA+ENC_PAGE0+ENC_START
        out     dx,al
        setport EN0_RCNTLO      ; remote byte count 0
        pause_
        mov     al,cl
        out     dx,al
        setport EN0_RCNTHI
        pause_
        mov     al,ch
        out     dx,al
        pop     ax              ; get our page back
        setport EN0_RSARLO
        pause_
        out     dx,al           ; set as hi address
        setport EN0_RSARHI
        pause_
        mov     al,ah
        out     dx,al
        setport EN_CCMD
        pause_
        mov     al,ENC_RREAD+ENC_START  ; read and start
        out     dx,al
        setport NE_DATAPORT
        pause_
        cmp     is_186,0
        jnz     read_186
read_loop:
        in      al,dx           ; get a byte
        stosb                   ; save it
        loop    read_loop
        ret
read_186:
        shr     cx,1            ; word count
        .286
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        rep     insw
        .8086
        jnc     read_186_1              ;is there an extra byte?
        in      ax,dx                   ;yes, read the next word and store a
        stosb                           ;  byte.
read_186_1:
        ret
;
;       Block output routine
;       CX = byte count, ds:si = buffer location, ax = buffer address

block_output:
        assume  ds:nothing
        push    ax              ; save buffer address
        inc     cx              ; make even
        and     cx,0fffeh
        loadport
        setport EN_CCMD
        pause_
        mov     al,ENC_NODMA+ENC_START
        out     dx,al           ; stop & clear the chip
        setport ENO_RCNTLO      ; remote byte count 0
        pause_
        mov     al,cl
        out     dx,al
        setport ENO_RCNTHI
        pause_
        mov     al,ch
        out     dx,al
        pop     ax              ; get our page back
        setport ENO_RSARLO
        pause_
        out     dx,al           ; set as lo address
        setport ENO_RSARHI
        pause_
        mov     al,ah
        out     dx,al
        setport EN_CCMD
        pause_
        mov     al,ENC_RWRITE+ENC_START ; write and start
        out     dx,al
        setport NE_DATAPORT
        pause_
        cmp     is_186,0
        jnz     write_186
write_loop:
        lodsb                   ; get a byte
        out     dx,al           ; save it
        loop    write_loop
        jmp     short block_output_1
write_186:
        shr     cx,1            ; word count
        .286
        rep     outsw
        .8086
block_output_1:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        mov     cx,0
        setport EN0_ISR
tx_check_rdc:
        in      al,dx
        test    al,ENISR_RDC    ; dma done ???
        jnz     tx_start
        loop    tx_check_rdc
        stc
        ret
tx_start:
        clc
        ret


        include 8390.asm

        public  usage_msg
usage_msg       db      "usage: NE2000 [options] <packet_int_no> <int_level> <io_addr>",
                db       CR,LF,'$'

        public  copyright_msg
copyright_msg   db      "Packet driver for NE2000, version "
                db      '0'+(majver / 10),'0'+(majver mod 10),".",'0'+version,
                db      ".",'0'+dp8390_version,CR,LF,'$'

int_no_name     db      "Interrupt number ",'$'
io_addr_name    db      "I/O port ",'$'

        extrn   set_recv_isr: near

;enter with si -> argument string, di -> word to store.
;if there is no number, don't change the number.
        extrn   get_number: near

;enter with dx -> name of word, di -> dword to print.
        extrn   print_number: near

        public  parse_args
parse_args:
;exit with nc if all went well, cy otherwise.
        mov     di,offset int_no
        call    get_number
        mov     di,offset io_addr
        call    get_number
        clc
        ret

        extrn   etopen_diagn: byte

init_card:
;get the board data. This is (16) bytes starting at remote
;dma address 0. Put it in a buffer called board_data.
        assume  ds:code

        or      endcfg,ENDCFG_WTS
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        loadport
        mov     al,endcfg
        setport EN0_DCFG
        pause_
        out     dx,al

        mov     cx,10h          ; get 16 bytes,
        movseg  es,ds
        mov     di,offset board_data

        setport EN_CCMD
        pause_
        mov     al,ENC_NODMA+ENC_PAGE0+ENC_START
        out     dx,al
        setport EN0_RCNTLO      ; remote byte count 0
        pause_
        mov     al,20h          ; count is actually doubled.
        out     dx,al
        setport EN0_RCNTHI
        pause_
        xor     al,al           ; high byte of count is zero.
        out     dx,al

        mov     ax,0            ; from address 0

        setport EN0_RSARLO
        pause_
        out     dx,al           ; set as hi address
        setport EN0_RSARHI
        pause_
        mov     al,ah
        out     dx,al
        setport EN_CCMD
        pause_
        mov     al,ENC_RREAD+ENC_START  ; read and start
        out     dx,al
        setport NE_DATAPORT
        pause_
sp_read_loop:
        in      al,dx           ; get a byte
        stosb                   ; save it
        loop    sp_read_loop

        push    ds              ; Copy from card's address to current address
        pop     es

        mov si, offset board_data       ; address is at start
        mov di, offset rom_address
        mov cx, EADDR_LEN       ; Copy one address length
        rep     movsb           ; ..

        clc
        ret

        public  print_parameters
print_parameters:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
;echo our command-line parameters
        mov     di,offset int_no
        mov     dx,offset int_no_name
        call    print_number
        mov     di,offset io_addr
        mov     dx,offset io_addr_name
        call    print_number
        ret

code    ends

        end
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ เพราะได้รับคำแนะนำอันอบอุ่น รวมถึงความช่วยเหลือในด้านการจัดหาอุปกรณ์ต่างๆที่จำเป็นจากอาจารย์หลายๆท่าน โดยเฉพาะอย่างยิ่ง อาจารย์บรรจง ปิยะธำรง, อาจารย์วัชระ ฉัตรวิริยะ และอาจารย์สมศักดิ์ วลัยรัชต์ และต้องขอกล่าวคำขอโทษต่ออาจารย์ประสาน ดังติสานนท์ ที่ทำให้อาจารย์ไม่ได้รับความสะดวกในการติดตั้งระบบเครือข่ายของเราในช่วงแรก ขอขอบคุณพี่อุดม และวุฒิเจตน์ อร่ามดิลกรัตน์ ในการประสานงานจัดหาอุปกรณ์ต่างๆให้ และท้ายที่สุดคือพี่อัศวิน เชาวนกฤษณกุล ที่เอื้อเฟื้อรูทีนในการทำเมนูแบบพูลดาวน์ ซึ่งทำให้พวกเราเขียนโปรแกรมได้สะดวกยิ่งขึ้น

ขอบคุณมาก

ผู้จัดทำปริญญานิพนธ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

หนังสืออ้างอิง

ต้น ตัณฑ์สุทธิวงศ์, สุพจน์ ปุณณชัยยะ, "คู่มือใช้งาน NETWARE ฉบับสมบูรณ์", สำนักพิมพ์ Provision, 212 หน้า , 2537

Black Uyless D. , "Data Networks", Prentice-Hall, 899 p., 1989

Debra R.Niedermiller-Chaffins, Drew Heywood, "Netware Training Guide: Networking Technologies", New Riders Publishing, USA, 1993

Karanjit Siyan, "Netware Professional Reference 4th ", New Riders Publishing, 1414 p., 1995

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า