การ์ดไอเอสเอลีนุกซ์เซิร์ฟเวอร์

LINUX SERVER ON ISA CARD

นายนุกูล พิมเสน
นายมานต ชัยสิรินิวัฒน์
นายศักดิ์ชัย เส้งสุย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

การ์ดไอเอสเอลีนุกซ์เซิร์ฟเวอร์

LINUX SERVER ON ISA CARD

โดย

นายนุกูล พิมเสน

นายมานต ชัยสิรินิวัฒน์

นายศักดิ์ชัย เส้งสุย

อาจารย์ที่ปรึกษา

อาจารย์อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การ์ดไอเอสเอลีนุกซ์เซิร์ฟเวอร์

LINUX SERVER ON ISA CARD

ผู้จัดทำ

1. นายนุกูล พิมเสน รหัสประจำตัว 40013254
2. นายมานต ชัยสิรินิวัฒน์ รหัสประจำตัว 40013260
3. นายศักดิ์ชัย เส้งสุย รหัสประจำตัว 40013272

อาจารย์ที่ปรึกษา

(อาจารย์อภิเนตร อูนากูล)

การ์ดไอเอสเอลีนุกซ์เซิร์ฟเวอร์

นายนุกูล พิมเสน 40013254
นายมานต ชัยสิรินิวัฒน์ 40013260
นายศักดิ์ชัย เส้งสุย 40013272
อาจารย์อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

ปัจจุบันคำว่า "อินเทอร์เน็ต" คงมีน้อยคนนักที่ไม่รู้จัก หรือไม่ใช้งานมัน ถ้าเกี่ยวข้องกับ "คอมพิวเตอร์" โดยอาจจะเห็นได้ว่าคนส่วนมากใช้คอมพิวเตอร์เพื่อการใช้งานอินเทอร์เน็ต ถ้ารู้จักใช้มันในทางที่ถูกต้องเหมาะสมจะสามารถใช้ประโยชน์จากมันได้อย่างมากมาย ซึ่งเป็นการเสียโอกาสอย่างยิ่ง ถ้าไม่มีโอกาสได้ใช้ความก้าวหน้าทางวิทยาการด้านนี้

จำนวนของเยาวชนไทยในวัยศึกษาที่มีโอกาสได้รู้จักอินเทอร์เน็ตยังมีอยู่น้อย ซึ่งไม่ได้เกิดจากพวกเขาไม่มีความสามารถในการเรียนรู้ แต่มาจากความขาดแคลนของอุปกรณ์ต่างๆ ที่จะนำมาใช้ในการเข้าถึงอินเทอร์เน็ต และอุปกรณ์เหล่านั้นเป็นหน้าที่ของใครที่จะจัดเตรียมไว้ให้ เป็นหน้าที่ของพวกเขาหรือที่จะจัดหามาใช้ที่บ้านเอง หรือเป็นหน้าที่ของสถานศึกษาที่ต้องจัดเตรียมไว้ให้ ถ้าเป็นอย่างแรก ความเป็นสถานศึกษาคงมีบทบาทในการเรียนรู้ลดลงไปมาก จึงทำให้ยังมีเยาวชนที่ห่างไกลอีกมากที่ยังขาดโอกาสเหล่านี้อยู่ อาจพูดตรงๆว่าพวกเขากลายเป็นคนล้าหลังไปจากพวกเราหลายก้าว และนั่นคือ "ช่องว่าง"

เราเลือกที่จะนำคอมพิวเตอร์ ที่ใช้โพรเซสเซอร์ ตระกูล 486 หรือ เพนเทียม กับระบบปฏิบัติการ "ลีนุกซ์" และอุปกรณ์เก็บข้อมูลที่มีความทนทานสูงประหยัดพลังงาน ซึ่งแน่นอนว่าไม่ใช่ ฮาร์ดดิสก์ เราใช้ แฟลช เมโมรี แล้วทำให้อุปกรณ์ที่กล่าวมาเป็นอุกรณ์ที่ใช้ในการเข้าถึงอินเทอร์เน็ต โดยสร้างการ์ดที่ใช้ แฟลช เมโมรีแล้วติดตั้งลีนุกซ์ลงบนแฟลช เมโมรีนั้น แล้วนำการ์ดนั้นติดตั้งลงในคอมพิวเตอร์ทั่วไป ทางสล๊อตไอเอสเอ และนั่นเราสร้างอุปกรณ์ที่สามารถเข้าถึงอินเทอร์เน็ตได้ ด้วยความสามารถของลีนุกซ์เอง จากนั้นเราใช้ ไมโครคอนโทรลเลอร์ เชื่อมต่อเข้าไปในอินเทอร์เน็ต โดยผ่านเซิร์ฟเวอร์ที่สร้างขึ้นมา ซึ่งท้ายที่สุดจะได้ระบบเครือข่ายที่สามารถสื่อสารผ่านอินเทอร์เน็ต ด้วยค่าใช้จ่ายที่ไม่สูงเลย โดยหวังไว้ว่าจะนำไปปรับปรุงเพิ่มเติมให้ใช้ประโยชน์ได้อย่างจริงจังต่อไป

LINUX SERVER ON ISA CARD

Nukoon Phimsen

Manut Chaisiriniwat

Sakchai Sengsui

Apinetr Unakul

Advisor

ABSTRACT

Nowadays the word "Internet" have a few people don't know or don't relate with it, if related with computer. Will be say that the most people used computer for used internet. If recognize to use it in appropriate way will make a lot of useful it. If have not chance to use this technology that will cause have lose chance.

Number of young in educational age that have chance to know internet is too little. That have not cause from they have not capable for study. But cause from have not enough many device for internet access. And have some question that whom is responsibility to provide it? Is it they or educational institute responsibility? If is first answer educational institute may be reduce the role of education a lot. That cause far away young have not this chance. May be straightforward say that they become to lag behind a lot from us. That is "space".

We choose computer that use processor 486.or Pentium family with Linux OS. And use storage with very rugged and low power. That isn't har disk sure. It is flash memory. Then make that to be device for internet access. By make card that have install flash and embed linux into that flash then install this card into general computer via ISA slot. That is we make device that have capable of internet access due to capable of Linux itself. Then we use micro-controller connect to internet via our server that we make first. Final we have network that can communicate to internet while have whole cost is low. We hope to be improve addition for useful it in the feature.

กิตติกรรมประกาศ

การทำปริญญานิพนธ์ครั้งนี้มิอาจเสร็จได้ด้วยดี ถ้าหากว่าไม่ได้รับการช่วยเหลือจากผู้มีพระคุณที่ให้การสนับสนุนทางด้านต่างๆ หลายท่านด้วยกัน เพื่อให้การทำปริญญานิพนธ์นี้สำเร็จได้ด้วยดี ซึ่งในที่นี้จะขอกล่าวถึงเพียงสองท่าน ซึ่งมีบทบาทอย่างมากในการทำวิทยานิพนธ์ครั้งนี้ ท่านแรกคงเป็นใครไปไม่ได้ ถ้าไม่ใช่อาจารย์ที่ปรึกษาวิทยานิพนธ์ คือท่านอาจารย์อภิเนตร อูนากูล ที่คอยเอาใจใส่ ให้คำปรึกษาที่มีประโยชน์อย่างมาก จึงขอขอบพระคุณเป็นอย่างสูง และอีกท่านที่อยากจะกล่าวถึง คือคุณขนิษฐา ประสารสุข ที่ช่วยสนับสนุนการทำงานต่างๆให้ราบรื่นตลอดมา จึงขอขอบพระคุณเป็นอย่างสูง

และต้องขอบคุณห้องวิจัย ESL ที่เป็นสถานที่ที่เราใช้ทำปริญญานิพนธ์รวมถึงอุปกรณ์ต่างๆที่เราต้องใช้ประกอบการทำปริญญานิพนธ์ครั้งนี้ด้วย

สุดท้ายที่ขาดไม่ได้เลยก็คือคุณพ่อคุณแม่ที่เลี้ยงดูเรามาและคอยให้กำลังใจเราเสมอมา

นายนุกูล พิมเสน
นายมานต ชัยสิรินิวัฒน์
นายศักดิ์ชัย เส้งสุย

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VII
สารบัญภาพ	VIII
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของงานวิจัย	2
1.3 ลักษณะของงานวิจัย	2
บทที่ 2 ทฤษฎีและหลักการที่เกี่ยวข้อง	5
2.1 โอเอสไอ โมเดล และ อินเทอร์เน็ตโพรโตคอล	5
2.1.1 โอเอสไอ โมเดล	5
2.1.2 ทีซีพีไอพีโพรโตคอล (TCP/IP Protocol)	8
2.2 การสื่อสารแบบ เน็ตเวิร์ก ด้วย RS-485	9
2.3 กระบวนการ POST และ บูตระบบปฏิบัติการ	10
2.4 การโหลดระบบปฏิบัติการ	11
2.5 ลีนุกซ์กับการจัดการระบบเน็ตเวิร์ก	12
2.5.1 หลักการที่ใช้ในการจัดการระบบเน็ตเวิร์ก	12
2.5.2 การประยุกต์ใช้งาน sk_buff	13
2.5.3 รูทีน (Routine) ที่สนับสนุนการทำงานในระดับสูง	15
2.5.4 อุปกรณ์เน็ตเวิร์ก	16
2.5.5 โครงสร้างพื้นฐานของระบบเน็ตเวิร์กในลีนุกซ์	17
2.6 ทฤษฎีของหน่วยความจำแฟลช	17
2.6.1 การอิมพลีเมนต์เรซิเดนต์แฟลชอาร์เรย์ (Implementing Resident Flash Array)	17
2.6.2 เรซิเดนต์แฟลชดิสก์ (Resident Flash Disk)	19
2.6.3 ความต้องการทางด้านซอฟต์แวร์ของ RFD	19
2.6.4 การอิมพลีเมนต์ FTL	21
2.6.5 ความต้องการทางด้าน ฮาร์ดแวร์ของ RFD	24
2.6.6 การออกแบบหน่วยความจำแฟลช ภายใต้ สถาปัตยกรรมของอินเทล	25
2.6.7 สองแนวทางของคีย์ (Key) เทคโนโลยี ที่เป็นทางเลือก	28

2.6.8 ดิสก์ออนชิป (Disk on Chip) 29
บทที่ 3 แนวปฏิบัติงานวิจัย 35
3.1 การ์ดไอเอสเอลีนุกซ์ 35
3.1.1 การทดลองสร้างการ์ดจากคอมแพคแฟลช 35
3.1.2 การทดลองสร้างการ์ดจากดิสก์ออนชิป 36
3.2 การออกแบบการ์ดในส่วนของการสื่อสาร RS-485 38
3.3 การเขียนโปรแกรมดีไวซ์ไดรเวอร์หรือเคอร์เนลโมดูลบนระบบปฏิบัติการลีนุกซ์ 40
3.3.1 การเขียนโปรแกรมโมดูล 40
3.3.2 การสร้างเน็ตเวิร์กดีไวซ์ไดรเวอร์โมดูล 41
3.3.3 การใช้งานอินเทอร์รัปต์ 43
3.3.4 การใช้งานแรมบนอุปกรณ์ 43
3.3.5 การใช้งานไอโอ 44
3.4 การอิมพลีเมนต์เครือข่ายท้องถิ่นด้วย RS-485 44
3.4.1 ปัญหาที่เจอในระบบ 44
3.4.2 การออกแบบ 45
3.4.3 ดาต้าลิงก์ 45
3.4.4 เอสแอลไอพี (SLIP : Serial Line Internet Protocol) 49
3.4.5 การทำงานในแต่ละสถานะ 51
บทที่ 4 ผลจากการวิจัย 58
4.1 การทดสอบบอร์ดของ MCS-51 58
4.1.1 ทดสอบ สเตติกแรม จาก พีซี 58
4.1.2 ทดสอบ สเตติกแรม จาก MCS-51 58
4.1.3 ทดสอบการทำงานของ 58
4.1.4 การทดสอบการทำงานเบื้องต้น 58
4.1.5 การทำการทดสอบการรับส่ง แพ็คเก็จ จริงๆ 58
4.2 การทดสอบเครือข่ายของ MCS-51 58
4.2.1 การทำงานบนระบบลีนุกซ์ 58
4.2.2 การทำงานของอุปกรณ์ที่ต่อเป็นลูกข่าย 62
บทที่ 5 บทวิจารณ์และสรุปผล 63
5.1 สรุปผลงาน 63
5.2 ประโยชน์จากโครงการ 63
5.3 ปัญหาที่พบจากโครงการ 63
5.4 แนวทางการพัฒนาต่อ 64
5.5 การประยุกต์ใช้ 64

5.6 สรุปภาพรวมของโครงการ 65
ภาคผนวก 66
ภาคผนวก.ก ลักษณะภายนอกและรายละเอียดขาของดิสก์ออนชิป 66
ภาคผนวก.ข ลักษณะภายนอกของคอมแพคแฟลชและคำสั่งของเอทีเอ 67
ภาคผนวก.ค เฮดเดอร์ของโพรโตคอลทีซีพีไอพี 69
คำอธิบายศัพท์ 72
บรรณานุกรม 73

สารบัญภาพ

หน้าที่

รูปที่ 1 เลเยอร์ทั้ง 7 เลเยอร์ของโอเอสไอโมเดลมีดังต่อไปนี้ 6

รูปที่ 2 แสดง 4 เลเยอร์ของทีซีพีไอพีโพรโตคอล 8

รูปที่ 3 บล็อกไดอะแกรมการสื่อสารโดยใช้ RS-485 10

รูปที่ 4 รูปแสดงการโหลดระบบปฏิบัติการลีนุกซ์ 12

รูปที่ 5 เริ่มการจับจองหน่วยความจำโดยฟังก์ชัน alloc_skb() 15

รูปที่ 6 เมื่อได้มีการสงวนหน่วยความจำบางส่วนโดยใช้ skb_reserve() 15

รูปที่ 7 หลังจากการใส่ข้อมูลลงในบัฟเฟอร์ 15

รูปที่ 8 จากการเรียกใช้ฟังก์ชัน skb_put() 15

รูปที่ 9 จากการเรียกใช้ฟังก์ชัน skb_push() 15

รูปที่ 10 โครงสร้างพื้นฐานระบบเน็ตเวิร์กในลีนุกซ์ 17

รูปที่ 11 การเปรียบเทียบระหว่างสถาปัตยกรรม เอมเบ็ดเด็ด กับ พีซี 18

รูปที่ 12 ประเภทของ เรซิเดนต์แฟลชอาร์เรย์ 18

รูปที่ 13 การ แมพบล็อก ของ แฟลช เป็น บล็อกอ่านเขียน(เซ็กเตอร์) 21

รูปที่ 14 การอิมพลีเมนต์เอฟทีแอล 23

รูปที่ 15 แมพของหน่วยความจำระบบ 25

รูปที่ 16. การอินเทอร์เฟชหน่วยความจำแฟลชกับสถาปัตยกรรมอินเทล 26

รูปที่ 17 การเพจหน่วยความจำแฟลชไปสู่ พื้นที่ของเอ็กแพนชันรอม 27

รูปที่ 18 การ เชื่อมต่อ หน่วยความจำแฟลชไปยัง ไอเอสเอบัส 28

รูปที่ 19 การ เปรียบเทียบการอิมพลีเมนต์แบบ เอทีเอ และ เอฟทีแอล 29

รูปที่ 20 การแมพดิสก์ออนชิปลงสู่แมพของหน่วยความจำระบบ 31

รูปที่ 21 ไดอะแกรมของดิสก์ออนชิป 32

รูปที่ 22 หน้าที่การทำงานของ ทรูเอฟเอฟเอส 32

รูปที่ 23 บล็อกไดอะแกรมของคอมแพคแฟลช 34

รูปที่ 24.การเชื่อมต่อพื้นฐานสำหรับคอมแพคแฟลช 35

รูปที่ 25 การอินเทอร์ดิสก์ออนชิป 36

รูปที่ 26 วงจรการใช้งานดิสก์ออนชิปบนไอเอสเอ 37

รูปที่ 27 วงจรการใช้งานดิสก์ออนชิปบนแบบหลายตัวเพื่อเพิ่มความจุ 37

รูปที่ 28 การ์ดไอเอสเอในส่วนที่ติดต่อกับ MCS-51 39

รูปที่ 29 การเปรียบเทียบระหว่างโอเอสไอโมดูลกับเน็ตเวิร์ก RS-485 45

รูปที่ 30 รูปแสดงสเตทไดอะแกรม 46

รูปที่ 31 แพ็กเก็จของอีเทอร์เน็ต 48

รูปที่ 32 รูปแสดงการส่งข้อมูลแบบเอสแอลไอพี 50

รูปที่ 33 โฟลวชาร์ตแสดงการรับข้อมูล 51
รูปที่ 34 รูปโฟลวชาร์ตแสดงการทำงานของเซิร์ฟเวอร์ซิงโครไนซ์ 52
รูปที่ 35 รูปโฟลวชาร์ตแสดงการซิงโครไนซ์ของไคลเอ็นต์ 53
รูปที่ 36 รูปโฟลวชาร์ตแสดงคอนเน็กชันสเตท 54
รูปที่ 37 รูปแสดงการทำงานของทรานสมิทชันสเตท 55
รูปที่ 38 รูปโฟลวชาร์ตแสดงการทำงานของดิสคอนเน็กสเตท 56
รูปที่ 39 รูปโฟลวชาร์ต แสดงการทำงานของพาสซิฟสเตท 57
รูปที่ 40 ผลจากการติดตั้งโมดูล 59
รูปที่ 41 การใช้คำสั่ง ping เพื่อทดสอบการทำงานโมดูล 59
รูปที่ 42 การติดตามผลการทำงานของโมดูล 60
รูปที่ 43 แพ็กเกจของดิสก์ออนชิป 66
รูปที่ 44 แสดงลักษณะทางภายนอกของคอมแพคแฟลช 67
รูปที่ 45 โครงสร้างของไอพีเฮดเดอร์ 69
รูปที่ 46 โครงสร้างของทีซีพีเฮดเดอร์ 69
รูปที่ 47 โครงสร้างของยูดีพีเฮดเดอร์ 70

ในฐานะที่พวกเราเลือกที่จะเป็นวิศวกรคอมพิวเตอร์ ที่มีความรู้ด้านนี้อยู่ระดับหนึ่ง พอเพียงที่จะสร้างสรรค์อุปกรณ์พื้นฐานที่ใช้ในการเข้าถึงเครือข่ายของอินเทอร์เน็ตได้ เพื่อที่จะนำไปปรับปรุงใช้งานให้สามารถนำไปใช้ประโยชน์ได้อย่างมีประสิทธิภาพต่อไปด้วยต้นทุนรวมต่ำ เราจึงเลือกทำงานวิจัยนี้ขึ้นมา

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาถึงแนวทางในการสร้างเซิร์ฟเวอร์ (Server) สำหรับเครือข่ายระดับท้องถิ่น เพื่อที่ใช้ติดต่อสื่อสารกับเครือข่ายของอินเทอร์เน็ต โดยมีข้อจำกัดอยู่ที่ค่าใช้จ่าย คือต้องใช้ค่าใช้จ่ายน้อยที่สุดเท่าที่จะทำได้และระดับความสามารถของ เซิร์ฟเวอร์อยู่ในระดับของ 80486 – เพนเทียม (Pentium) เพื่อเป็นการนำวิทยาการที่ถือได้ว่าหมดประโยชน์แล้วกลับมาทำให้เกิดประโยชน์

1.2.2 ศึกษาแนวทางในการสร้างหน่วยความจำสำรองที่มีความทนทานสูง ประหยัดพลังงาน ไม่ต้องการการบำรุงรักษามากนัก และมีความเชื่อถือได้สูง มาทดแทนการใช้งานฮาร์ดดิสก์ (Hard Disk) ที่ใช้อยู่ในคอมพิวเตอร์ทั่วๆไปในปัจจุบัน เพื่อความเชื่อถือได้สูงสุดของเซิร์ฟเวอร์

1.2.3 ศึกษาวิธีการติดตั้งใช้งานระบบปฏิบัติการลีนุกซ์ลงบนหน่วยความจำสำรองที่ใช้ทดแทนฮาร์ดดิสก์ เพื่อที่จะใช้ลีนุกซ์ (Linux) ในการพัฒนาแอพพลิเคชัน (Application) สำหรับการติดต่อสื่อสารผ่านเครือข่ายอินเทอร์เน็ตและเครือข่ายท้องถิ่นที่จะสร้างขึ้นมา

1.2.4 ศึกษาแนวทางในการสร้างเครือข่ายระดับท้องถิ่นที่ใช้อุปกรณ์ระดับพื้นฐาน ในระดับของไมโครคอนโทรลเลอร์ (Micro Controller) รวมถึงศึกษาการสร้าง ดาต้าลิงก์เลเยอร์โพรโตคอล (Data Link Layer Protocol) ที่จะนำมาใช้งานบนเครือข่ายที่สร้างขึ้นมานี้ เพื่อนำเสนอว่าสามารถใช้อุปกรณ์ที่มีความสามารถต่ำกว่าคอมพิวเตอร์ส่วนบุคคลอยู่มาก ในการใช้งานแอพพลิเคชันของอินเทอร์เน็ต

1.3 ลักษณะของงานวิจัย

ข้อจำกัดของงานวิจัยนี้อยู่ที่งบประมาณ ตามที่กล่าวไว้ตามหัวข้อ 1.1 ข้างต้น ดังนั้นเราจำเป็นต้องเลือกใช้อุปกรณ์ที่มีต้นทุนต่ำ มีความทนทานสูง ไม่ต้องการการดูแลบำรุงรักษามากนัก เนื่องจากจุดมุ่งหมายเพื่อนำไปใช้กับผู้ด้อยโอกาส ซึ่งอยู่ห่างไกลความเจริญ โดยที่งานวิจัยนี้มีลักษณะโดยรวมใหญ่ๆอยู่ 2 ขั้นคือ

1.3.1 สร้างอุปกรณ์ที่ใช้ในการเข้าถึงอินเทอร์เน็ต

โดยอุปกรณ์นี้จะทำหน้าที่เชื่อมต่อกับเครือข่ายของอินเทอร์เน็ต เพื่อทำให้อุปกรณ์ที่มาเชื่อมต่อเป็นลูกข่ายสามารถสื่อสารผ่านเครือข่ายของอินเทอร์เน็ตได้ โดยขออ้างถึงอุปกรณ์นี้ด้วยคำว่า "เซิร์ฟเวอร์" ซึ่งประกอบไปด้วยส่วนประกอบที่เราต้องสร้างขึ้นมาและที่มีอยู่แล้วดังต่อไปนี้คือ

เคลื่อนไหวในขณะทำงาน จึงทำให้ประหยัดพลังงานกว่ามาก รวมถึงทำให้เกิดความเชื่อถือได้มากกว่า ฮาร์ดดิสก์อยู่มาก ระบบปฏิบัติการที่ติดตั้งลงบนการ์ดนี้ คือ ลีนุกซ์ เนื่องมาจากลักษณะของลีนุกซ์เองมีข้อได้เปรียบกว่า ระบบปฏิบัติการอื่นหลายอย่างด้วยกัน โดยที่มีความสามารถที่ระบบปฏิบัติการทั่วๆไปมี ขณะที่ต้องการทรัพยากรของระบบต่ำกว่าระบบปฏิบัติการทั่วไป และอีกเหตุผลหนึ่งคือ เป็นระบบปฏิบัติการแบบ เปิดเผยโค้ด จึงทำให้มีความยืดหยุ่นหรืออิสระในการพัฒนามาก แล้วติดตั้งซอฟต์แวร์อื่นๆสำหรับลีนุกซ์ ที่จำเป็นสำหรับการให้บริการเพื่อการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต ลงไปบนการ์ด ซึ่งการ์ดนี้สามารถนำไปติดตั้งกับอุปกรณ์ ตามหัวข้อ 1.2.1.1 และ 1.2.1.2 ใดๆเพื่อทำหน้าที่เป็นเซิร์ฟเวอร์ให้กับลูกข่ายอื่นๆต่อไป

1.3.2 สร้างอุปกรณ์ที่ใช้ควบคุมลูกข่ายและตัวอย่างลูกข่าย

โดยในส่วนนี้เพื่อที่จะนำเสนอว่าในการติดต่อสื่อสารผ่านเครือข่ายของอินเทอร์เน็ตนั้น เราสามารถใช้อุปกรณ์ที่มีความสามารถอยู่ระดับหนึ่ง (ด้อยกว่าคอมพิวเตอร์ส่วนบุคคลอยู่มาก) สร้างเป็นเครือข่ายเพื่อที่จะสื่อสารผ่านอินเทอร์เน็ต สำหรับใช้งานลักษณะใดลักษณะหนึ่งที่ต้องการได้ โดยมีส่วนประกอบดังต่อไปนี้คือ

1.3.2.1 **การ์ดควบคุม** : คือส่วนที่ติดตั้งลงในเซิร์ฟเวอร์อีกที เป็นการ์ดที่อินเทอร์เฟซ (Interface) กับเซิร์ฟเวอร์ด้วยไอเอสเอเช่นกัน ซึ่งประกอบไปด้วย ไมโครคอนโทรลตระกูล MCS-51, แรม(RAM) สำหรับเป็นบัฟเฟอร์ (Buffer) และพอร์ต (Port) สำหรับเชื่อมต่อกับเครือข่ายของ RS-485 เป็นการ์ดที่ควบคุมการสื่อสารระหว่างเครือข่ายของ RS-485 กับเครือข่ายของอินเทอร์เน็ต

1.3.2.2 **บอร์ดลูกข่าย** : เป็นบอร์ด (Board) ที่ประกอบไปด้วยไมโครคอนโทรลเลอร์ ตระกูล MCS-51, แรมสำหรับเป็นบัฟเฟอร์และพอร์ตสำหรับเชื่อมต่อกับ เครือข่ายของ RS-485 เช่นกัน โดยบอร์ดนี้จะเชื่อมต่อเข้าสู่เครือข่ายด้วย RS-485 เป็นลูกข่ายของระบบ โดยสามารถติดต่อสื่อสารผ่านเครือข่ายอินเทอร์เน็ต เพื่อประยุกต์ใช้ทำงานบางอย่างตามความเหมาะสม ด้วยอินเทอร์เน็ตโพรโตคอล เพื่อแสดงให้เห็นว่า การประยุกต์ใช้ แอพพลิเคชันของอินเทอร์เน็ตไม่จำเป็นต้องใช้ความสามารถระดับคอมพิวเตอร์ส่วนบุคคลเสมอไป

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 โอเอสไอ โมเดล และ อินเทอร์เน็ตโพรโตคอล

ในการสร้างหรือใช้งานเครือข่ายพื้นฐานนั้น ต้องอาศัยมาตรฐานของการแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์โดยไม่ขึ้นอยู่กับสถาปัตยกรรม โดยในหัวข้อนี้จะอ้างถึงโมเดลอ้างอิงที่ถูกบัญญัติมาเพื่อเป็นโมเดลอ้างอิงในการเชื่อมต่อระบบเปิดใดๆ เข้าด้วยกัน และ ทีซีพีไอพี โพรโตคอล (TCP/IP Protocol) ที่เป็นชุดของ โพรโตคอล ที่พัฒนามาเพื่อการทำงานร่วมกันระหว่างคอมพิวเตอร์

2.1.1 โอเอสไอ โมเดล (OSI Model)

เป็นโมเดลที่ถูกพัฒนามาจาก International Standards Organization (ISO) ซึ่งถูกเรียกว่า "OSI (Open systems Interconnection) Reference Model" เพราะมีการเกี่ยวข้องกับการเชื่อมต่อระหว่างระบบเปิด (Open system) เข้าด้วยกัน ซึ่งก็คือระบบนั้นเปิดไว้สำหรับการติดต่อกับระบบอื่นๆ

โอเอสไอโมเดลถูกโมเดลให้เป็นชั้นๆหรือเลเยอร์ (Layers) อิสระจากกัน 7 เลเยอร์ โดยมีหลักการที่ใช้ในการโมเดลจนได้เป็น 7 เลเยอร์ดังนี้

- แต่ละเลเยอร์ควรจะถูกสร้างมาจากระดับของบทสรุปที่ต้องการที่แตกต่างกัน
- แต่ละเลเยอร์ควรจะกระทำตามหน้าที่ถูกนิยามไว้ดีแล้ว
- หน้าที่ของแต่ละเลเยอร์ควรคัดเลือกมาจากการมองมาจากโพรโตคอลที่เป็นมาตรฐานสากลมาแล้ว
- ขอบเขตของเลเยอร์ควรจะคัดเลือกเพื่อการโฟลว (Flow) ของข้อมูลผ่านอินเทอร์เฟซของแต่ละ เลเยอร์ต่ำที่สุด
- จำนวนของเลเยอร์ควรจะมากเพียงพอกับที่ ฟังก์ชันที่ต่างกันในเลเยอร์เดียวกันไม่ทำงานไปด้วยกันโดยไม่จำเป็นและน้อยพอกับที่สถาปัตยกรรมจะไม่กลายเป็นซับซ้อนเกินไป

ตัดสินใจ ว่าแพ็กเก็ต (Packet) จะถูกส่งมาจากแหล่งที่มา (Source) ไปยังจุดหมาย (Destination) อย่างไร

2.1.1.4 **ทรานสปอร์ตเลเยอร์ (Transport Layer)** : เตรียมการสำหรับการควบคุมการสื่อสารระหว่างปลายด้านหนึ่งกับอีกด้านหนึ่ง หน้าที่พื้นฐานของเลเยอร์นี้คือรับข้อมูลจากเลเยอร์บนแล้วแบ่งเป็นหน่วยที่เล็กกว่า ก่อนที่จะส่งผ่านไปให้ เน็ตเวิร์กเลเยอร์และทำให้แน่ใจว่าชิ้นของข้อมูลที่รับมาทั้งหมดนั้นถูกต้องที่ปลายอีกด้านหนึ่ง ภายใต้สภาวะปกติเลเยอร์นี้จะสร้าง การเชื่อมต่อของเน็ตเวิร์กแยกกันสำหรับแต่ละการเชื่อมต่อของทรานสปอร์ตที่ถูกต้องโดยการเลเยอร์บน โปรแกรมบนเครื่องต้นทางสนทนา (Conversation) กับโปรแกรมที่เหมือนกันบนเครื่องปลายทางโดยใช้ เมสเซจเฮดเดอร์ (Message Header) และเมสเซจควบคุม (Control Message)

2.1.1.5 **เซสชันเลเยอร์ (Session Layer)** : เนื่องจากในระบบระบบหนึ่งหรือในหนึ่งโหนดนั้นได้มีการใช้งานระบบเน็ตเวิร์กด้วยกันดังนั้นเลเยอร์นี้จึงเป็นส่วนที่จะช่วยในการจัดการควบคุมการใช้งานระบบเน็ตเวิร์ก ร่วมกันเช่นในระบบยูนิกซ์ (UNIX) จะมีโพรเซสจำนวนมากที่ต้องการใช้งานในระบบเน็ตเวิร์กโดยในการทำงานกับระบบเน็ตเวิร์กของแต่ละโพรเซสนั้นจะไม่มีการข้องเกี่ยวหรือรบกวนระหว่างโพรเซสด้วยกัน เช่นระบบการทำงานของซ็อกเก็ต (Socket) ในระบบยูนิกซ์

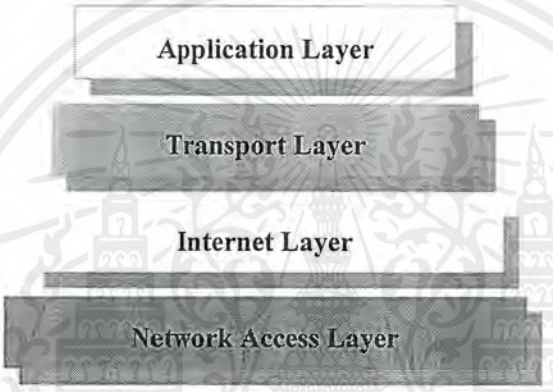
2.1.1.6 **พรีเซนเทชันเลเยอร์ (Presentation Layer)** : การแปลงข้อมูลพรีเซนเทชันเลเยอร์เกี่ยวข้องกับซีนแทก (Syntax) และ ซีแมนติค (Semantic) ของข้อมูลที่ถูกส่ง โดยมีหน้าที่ทั่วๆไปเกี่ยวกับการเข้ารหัสและถอดรหัสตามมาตรฐานที่กำหนดไว้ พรีเซนเทชันเลเยอร์ยังเกี่ยวข้องกับการนำเสนอของอินฟอร์เมชันอื่นด้วย เช่นการบีบอัดข้อมูลเพื่อลดจำนวนของบิตที่จะถูกส่งออกไปและการเข้ารหัสลับเพื่อความปลอดภัยของข้อมูล

2.1.1.7 **แอพพลิเคชันเลเยอร์ (Application Layer)** : เตรียมการบริการต่างๆให้กับ แอพพลิเคชัน แอพพลิเคชันเลเยอร์ บรรจุโพรโตคอลที่ถูกต้องการร่วมกันไว้หลากหลาย เช่นเวอร์ชวลเทอร์มินอล (Virtual Terminal) เนื่องจากมีเทอร์มินอลที่แตกต่างกันหลากหลายชนิด จึงต้องนิยามเวอร์ชวลเทอร์มินอลมา เพื่อสามารถแสดงข้อมูลที่เทอร์มินอลแต่ละชนิดได้อย่างถูกต้อง หน้าที่อื่นของ แอพพลิเคชันเลเยอร์เช่นการโอนย้ายไฟล์ (File Transfer) ระบบไฟล์ที่แตกต่าง

กันย่อมมีวิธีการให้ชื่อไฟล์ วิธีการนำเสนอข้อมูลและอื่นๆที่ต่างกัน ในการโอนย้ายระหว่างระบบที่ต่างกันนี้ต้องการการจัดการกับความไม่เข้ากันได้นี้ งานอื่นที่อยู่ในเลเยอร์นี้ได้แก่อิเล็กทรอนิกส์เมล์ (Electronic Mail) และอื่นๆ

2.1.2 ทีซีพีไอพีโพรโตคอล (TCP/IP Protocol)

เป็นชุดของ โพรโตคอลที่ถูกพัฒนามาเพื่อยอมให้มีการทำงานร่วมกันระหว่างคอมพิวเตอร์เพื่อการแชร์ (Share) ทรัพยากร ผ่านทางเครือข่าย ซึ่งถูกพัฒนาโดยกลุ่มของศูนย์นักวิจัย ARPAnet (Advanced Research Projects Agency) แต่ละโพรโตคอลใน ทีซีพีไอพีโพรโตคอลสามารถนำเสนอได้ภายใต้เลเยอร์ใดเลเยอร์หนึ่งของ 4 เลเยอร์ ดังต่อไปนี้



รูปที่ 2 แสดง 4 เลเยอร์ของทีซีพีไอพีโพรโตคอล

2.1.2.1 เน็ตเวิร์กแอ็กเซสเลเยอร์ (Network Access Layer) : มีการทำงานอยู่ในระดับ ฮาร์ดแวร์ มีหน้าที่หลักๆ ได้แก่จัดโครงสร้างของข้อมูลให้เหมาะสำหรับการส่งไปบนเครือข่ายและทำการแมพ (Map) ลอจิคัลแอดเดรส (Logical Address) ไปเป็น ฟิสิกส์คัลแอดเดรสของอุปกรณ์ (Physical Device Address)

2.1.2.2 อินเทอร์เน็ตเลเยอร์ (Internet Layer) : จัดเตรียมโพรโตคอลหลักสองโพรโตคอล คือ อินเทอร์เน็ตโพรโตคอล (IP Protocol) และ อินเทอร์เน็ตคอนโทรลเมสเซจโพรโตคอล (ICMP Protocol) ซึ่งอินเทอร์น็ตโพรโตคอลจะเกี่ยวข้องกับ การโฟลวของข้อมูลผ่านเครือข่าย ส่วนอินเทอร์เน็ตคอนโทรลเมสเซจโพรโตคอลถูกใช้สำหรับควบคุมการโฟลวของข้อมูล

2.1.2.3 ทรานสปอร์ทเลเยอร์ (Transport Layer) : เหมือนๆกับอินเทอร์เน็ตเลเยอร์ที่จัดเตรียมโพรโตคอลหลักไว้สองโพรโตคอล คือ ทรานสมิตชันคอนโทรลโพรโตคอล (TCP Protocol) และ ยูสเซอร์ดาต้าแกรมโพรโตคอล (UDP Protocol) ซึ่ง ทรานสมิตชัน

คอนโทรลโพรโตคอล จะทำหน้าที่สำหรับการส่งมอบ (Deliver) ข้อมูลของเครือข่ายเป็นส่วนใหญ่แต่ยังมีบางแอพพลิเคชัน ที่ใช้ความได้เปรียบของ ยูสเซอร์ดาต้าแกรมโพรโตคอล (UDP) ตรงที่มีโอเวอร์เฮด (Overhead) น้อยกว่าทำหน้าที่นี้แทน

2.1.2.4 **แอพพลิชันเลเยอร์ (Application Layer)** : จะครอบคลุมทุกๆแอพพลิเคชันโปรแกรมที่ใช้ข้อมูลที่มีการส่งมอบจาก ทีซีพีไอพี ซึ่งแอพพลิเคชันที่ใช้อยู่ทั่วๆไปได้แก่ จดหมายอิเล็กทรอนิกส์ และ รีโมทล็อกอิน (Remote Login) ซึ่งกลายมาเป็นแอพลิเคชันมาตรฐาน รวมถึงแอพพลิเคชันอื่นๆและแอพลิเคชันเฉพาะเจาะจงเฉพาะงานที่มีการใช้บริการจาก ทีซีพีไอพีด้วย

2.2 การสื่อสารแบบ เน็ตเวิร์ก ด้วย RS-485

ปัจจุบันงานทางด้านไมโครคอนโทรลเลอร์มักจะใช้ระบบมัลติโพรเซสเซอร์ (Multi-Processor) กันมากขึ้น โดยเฉพาะอย่างยิ่งงานที่ค่อนข้างซับซ้อน และมักจะใช้เครื่องไมโครคอมพิวเตอร์เป็นศูนย์กลางการควบคุมระบบมัลติโพรเซสเซอร์ คือการกระจายหน่วยควบคุมเป็นหน่วยย่อย ซึ่งแต่ละหน่วยจะมีไมโครคอนโทรลเลอร์อยู่ในส่วนที่ตัวเองดูแล โดยอาศัยการสื่อสารของแต่ละส่วนเข้าด้วยกัน เพื่อรับและส่งข้อมูลรวมทั้งเพื่อให้สามารถควบคุมจากส่วนกลางด้วย ซึ่งจะใช้ RS-485 ในการสื่อสาร

RS-485 เป็นการสื่อสารอนุกรมเช่นเดียวกับ RS-232 แต่มีข้อแตกต่างกันที่ RS-232 เป็นแบบจุดต่อจุด (Point-to-Point) แต่ RS-485 จะเป็นหลายจุด (Multipoint) คือสามารถเชื่อมต่อไปยังหลายๆ โหนดจากจุดเดียวกันซึ่งต่างจาก RS-232 โดย RS-485 จะใช้สายสัญญาณเพียงแค่ 2 เส้น เป็นทั้งรับและส่งในตัวเองการทำงานจะเป็นแบบ "Half-Duplex" ซึ่งจะต้องมีการควบคุมการรับส่ง ข้อดีของ RS-485 คือสายสัญญาณ 2 เส้นจะใช้สัญญาณไฟที่แตกต่างกัน ซึ่งจะหักล้างกัน นั่นคือเมื่อส่ง 0 สายเส้นหนึ่งจะเป็น +5 โวลต์ ในขณะอีกเส้นหนึ่งเป็น –5 โวลต์ และเมื่อส่ง 1 ระดับไฟจะสลับขั้วกัน จากคุณสมบัตินี้ทำให้ RS-232 สามารถส่งสัญญาณได้ไกลถึง 1.2 กิโลเมตร และสามารถต่อพ่วงตัวรับตัวส่งได้พร้อมๆถึง 32 จุดหรือโหนด ดังบล็อกไดอะแกรม

6. ถ้าต้องการบูต ระบบปฏิบัติการ "PnP" โดยทั่วไปจะมีการอนุญาตอุปกรณ์เฉพาะที่ต้องการใช้ในกระบวนการอ่านระบบปฏิบัติการสู่หน่วยความจำ เช่น คีย์บอร์ด (Keyboard), การ์ดแสดงผล (Display Adapter) เป็นต้น
7. ระบบปฏิบัติการจะถูกบูตในหน่วยความจำและการควบคุมจะถูกส่งต่อให้ระบบปฏิบัติการ

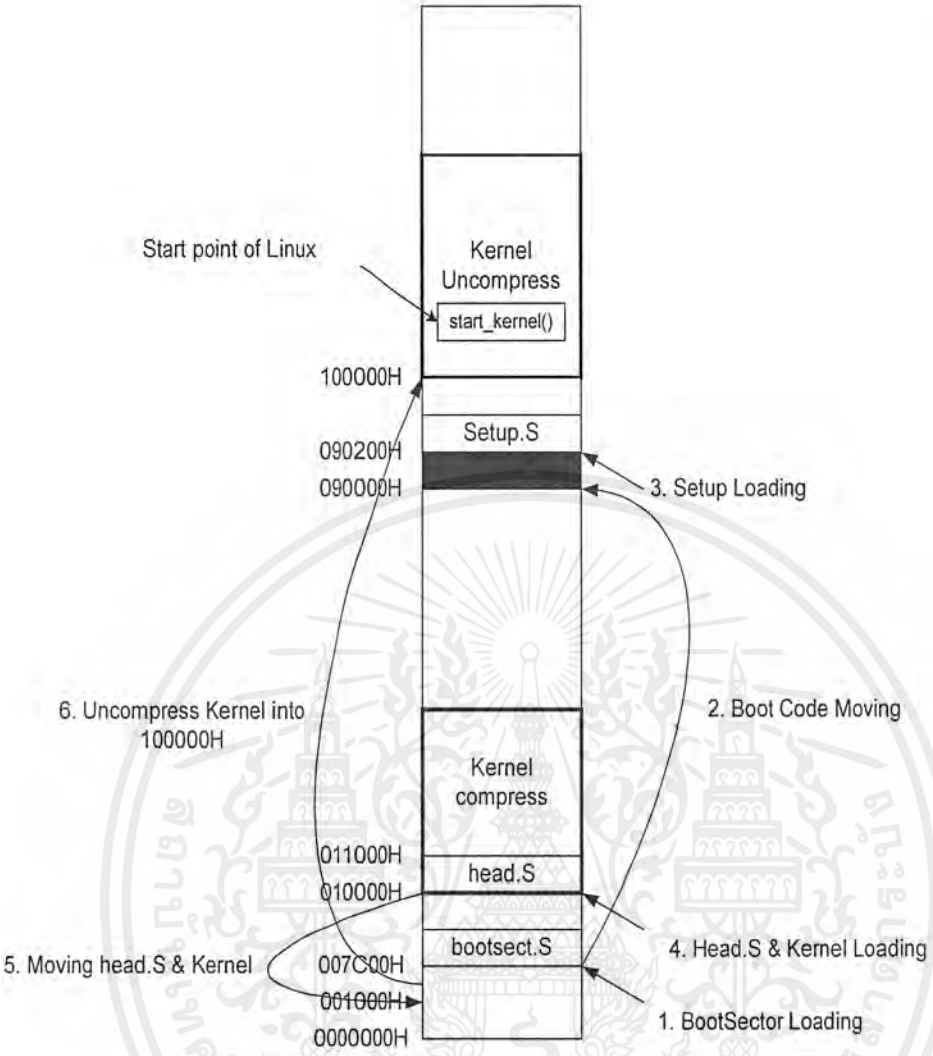
2.4 การโหลดระบบปฏิบัติการลีนุกซ์

เริ่มจากระบบคอมพิวเตอร์ได้เริ่มทำงานในโพรเซสเซอร์ตระกูล X86 จะเริ่มด้วยการรีเซต (Reset) ตัวเองซึ่งโพรเซสเซอร์จะอยู่ใน เรียลโหมด (Real Mode) และจะกระทำคำสั่งที่ตำแหน่ง FFFF0H ซึ่งจะเป็นตำแหน่งของไบออสและจะติดตั้งค่าของอินเทอร์รัปต์เว็กเตอร์ (Interrupt Vector) ที่ตำแหน่ง 00000H หลังจากนั้นจะโหลดโค้ด (Code) จากอุปกรประเภทบูตดีไวซ์ไว้ยังตำแหน่ง 07C00H และกระโดดไปทำงานที่ตำแหน่งนั้น

จากที่กล่าวมาข้างต้นนั้นยังไม่ได้เข้ายังระบบของลีนุกซ์ ซึ่งเป็นเพียงการเริ่มทำงานทั่วๆ ไปของการเข้าสู่ระบบของระบบปฏิบัติการทั่วๆไป ในส่วนการเข้าสู่ระบบลีนุกซ์จะเริ่มจากโค้ดที่อยู่ในไดเรกทอรี (Directory) /boot ไฟล์ (File) bootsect.S จากข้างต้นที่กล่าวมา โค้ดส่วนนี้จะเป็นส่วนที่โหลดจากไบออสไปยัง 90000H โดยการทำงานทั้งหมดที่ผ่านมาจะเป็นการทำงานที่อยู่ในเรียลโหมด โดย bootsect.S จะทำการโหลด โค้ดที่ติดตั้งระบบรวมทั้งโค้ดที่เป็นเคอร์เนล (Kernel) ในตำแหน่ง 10000H ต่อไป

การโหลดส่วนที่ติดตั้งระบบจะถูกโหลดไปที่ตำแหน่ง 90200H ซึ่งถัดจาก bootsect.S ไป 2 Kbyte โค้ดที่ทำการคือ /boot/Setup.S และ /boot/head.S ซึ่ง head.S จะทำการลิงก์ (Link) มาจาก zBoot/inflate.c, zBoot/unzip.c และ zBoot/misc.c โดยส่วนที่จะมีการทำงานเป็นส่วนแรกคือ Setup.S ในโค้ดนี้จะทำการติดตั้งระบบเช่น การติดตั้งค่าเริ่มต้นให้กับอุกรณ์ต่างๆ เช่น อินเทอร์รัปต์คอนโทรลเลอร์ เมื่อสิ้นสุดการติดตั้งแล้ว Setup.S จะย้ายเคอร์เนลจากตำแหน่ง 10000H ไปยัง 01000H จากนั้นจะทำการเปลี่ยนโหมดการทำงานจากเรียลโหมดไปยังโปรเท็กต์ (Protect Mode) ขั้นตอนต่อไปจะเป็นขั้นตอนการเข้าสู่ระบบปฏิบัติการลีนุกซ์

เนื่องจากโค้ดที่เป็นระบบปฏิบัติการนั้นจะถูกย่อยขนาดอยู่จึงไม่สามารถทำงานได้โดยตรง ดังนั้นจึงมีโค้ดส่วนหนึ่งในการขยายโค้ดตรงส่วนนี้โดย /boot/head.S การทำงานของ head.S จะขยายโค้ดของเคอร์เนลที่แท้จริงถัดจากจุดเริ่มต้นไป 01000H ไปจนสุดโค้ด โดยโค้ดนี้จะถูกขยายไปยังตำแหน่ง 100000H และย้ายการทำงานไปยังตำแหน่งนั้น จากนั้นจะติดตั้งส่วนที่ใช้ในการจักการหน่วยความจำหลัก (Memory Management) เช่นการติดตั้งค่า IDT, GDT และ LDT เมื่อติดตั้งส่วนการจัดการหน่วยความจำ เมื่อเสร็จสิ้นแล้วจะเริ่มเข้าสู่เคอร์เนลโดยการเรียกฟังก์ชัน start_kernel() ใน /init/main.c



รูปที่ 4 รูปแสดงการโหลดระบบปฏิบัติการลีนุกซ์

ในส่วนต่อไปก็จะเป็นการทำงานของตัวลีนุกซ์ โดยลีนุกซ์จะเรียกการทำงานไปยัง /etc/init หรือ /bin/init หรือ /sbin/init โดยการทำงานส่วนนี้ถ้าเป็นระบบโดยทั่วไปจะมีการอ่าน หรือทำการเรียกโปรแกรม คือจะเข้าถึงดิสก์ แต่ในระบบลีนุกซ์สามารถที่จะจำลองดิสก์บนหน่วยความจำหลักได้ (Ramdisk)

2.5 ลีนุกซ์กับการจัดการระบบเน็ตเวิร์ก

การจัดการระบบเน็ตเวิร์กในลีนุกซ์ไม่มีการสร้างโดยอ้างอิงตามมาตราฐาน "Berkely Socket API" ซึ่งมีจุดเริ่มต้นมาจากระบบยูนิกซ์ BSD (4.2/4.3/4.4 BSD) ในหัวข้อนี้จะกล่าวถึงการจัดการหน่วยความจำที่ใช้ในระบบเน็ตเวิร์กสำหรับเน็ตเวิร์กเลเยอร์และดีไวซ์ไดรเวอร์ภายใต้การทำงานของเคอร์เนลลีนุกซ์ซึ่งเป็นหัวใจสำคัญของระบบ

2.5.1 หลักการที่ใช้ในการจัดการระบบเน็ตเวิร์ก

โครงสร้างหลักของโค้ดของระบบเน็ตเวิร์กเริ่มจากการติดตั้งการทำงาน จนกระทั่งการสร้างและใช้งานซ็อกเก็ต

2.5.1.1 **อุปกรณ์หรือส่วนต่อเชื่อมระบบ (Device or Interface)** : อุปกรณ์ต่อเชื่อมระบบเน็ตเวิร์กจะแสดงถึงการรับ/ส่งแพ็กเก็ต โดยปกติโค้ดที่ใช้ติดต่อกับอุปกรณ์เช่นอีเทอร์เน็ตการ์ด แต่ในบางดีไวซ์จะเป็นการใช้ซอร์ฟแวร์ทั้งหมดเช่นลูปแบ็คดีไวซ์

2.5.1.2 **โพรโตคอล (Protocol)** : ในแต่ละโพรโตคอลจะเปลี่ยนได้กลับภาษาต่างที่อยู่ในเน็ตเวิร์ก บางโพรโตคอลจะมีการใช้งานโดยผู้ผลิตเอง และในบางโพรโตคอลจะออกแบบไม่เฉพาะวัตถุประสงค์ และในลีนุกซ์นั้น โพรโตคอลได้แยกออกเป็นแต่ละโมดูล โดยมีซ็อกเก็ตเลเยอร์ในการเตรียมบริการในส่วนนี้

2.5.1.3 **ซ็อกเก็ต (Socket)** : ซ็อกเก็จคือจุดต่อเชื่อมระหว่างเน็ตเวิร์กที่เตรียมไว้ในไอโอไฟล์ กับผู้ใช้งาน ในเคอร์เนลแต่ละซ็อกเก็ตนั้นจะรวมการทำงานระหว่างซ็อกเก็ตระดับสูงกับโพรโตคอลในระดับต่ำ

2.5.1.4 **ซ็อกเก็ตบัฟเฟอร์ (sk_buff)** : เป็นหน่วยความจำบัฟเฟอร์ที่ใช้ในเน็ตเวิร์กเลเยอร์ ในการควบคุมนี้จะถูกเตรียมโดยไลบรารี (Library) ระดับล่างให้มีการเรียกใช้จากเน็ตเวิร์กโพรโตคอล

2.5.2 การประยุกต์ใช้งาน sk_buff

วัตถุประสงค์หลักของ sk_buff คือการเตรียมระบบการจัดการและควบคุมการใช้งานบัฟเฟอร์ สำหรับทุกๆเน็ตเวิร์กเลเยอร์ โดยเริ่มจากการเตรียมการทำงานของ sk_buff ระดับสูงและการจัดการซ็อกเก็ตเพื่ออำนวยความสะดวกให้กับทุกโพรโตคอล

ส่วนของ sk_buff นั้นคือหน่วยความจำส่วนหนึ่งที่ถูกนำมาใช้เก็บข้อมูลที่เป็นโครงสร้างเพื่อใช้ในการควบคุมการทำงาน โดยจะมีฟังก์ชันอยู่สองส่วนหลักๆที่เตรียมไว้ในไลบรารี ส่วนแรกคือ ส่วนที่จัดการลิงก์ลิสต์ (Link List) ของ sk_buff และส่วนที่สองจะเป็น การจัดการการใช้งานหน่วยความจำ ส่วนของบัฟเฟอร์นั้นจะใช้ลิงก์ในการเพิ่มข้อมูลที่ท้ายสุดของลิงก์และย้ายข้อมูลออกจากส่วนหน้าสุดของลิงก์ข้อมูล โดยจะเกิดจากการทำงานของระบบเน็ตเวิร์กเมื่อเกิดการทำงาน ดังตัวอย่างโค้ดดังนี้

```
void append_frame(char *buf, int len)
{
    struct sk_buff *skb=alloc_skb(len, GFP_ATOMIC);
    if(skb==NULL)
        my_dropped++;
    else
    {
        skb_put(skb,len);
        memcpy(skb->data,data,len);
```

```
            skb_append(&my_list, skb);
        }
}
```

ฟังก์ชัน append_frame() เป็นการเพิ่มข้อมูลลงในบัฟเฟอร์ซึ่งจะถูกเรียกการใช้งานจากระบบ

```
void process_queue(void)
{
        struct sk_buff *skb;
        while((skb=skb_dequeue(&my_list))!=NULL)
        {
            process_data(skb);
            kfree_skb(skb, FREE_READ);
        }
}
```

ฟังก์ชัน precess_queue() เป็นฟังก์ชันที่จัดการย้ายข้อมูลออกจากคิว (Queue) เมื่อมีการส่งข้อมูลไปยังโพรโตคอล ซึ่งทั้งฟังก์ชัน append_frame() และ process_queue() จะคล้ายกับโค้ด netif_rx() และ net_bh() ในไฟล์ net/core/dev.c แต่จะมีความซับซ้อนมากกว่า แต่การทำงานจะมีลักษณะคล้ายกัน โดยจะส่งแพ็กเก็ตไปยังโพรโตคอลที่ถูกต้องและจากโพรโตคอลก็จะส่งข้อมูลไปยังระดับผู้ใช้อีกต่อไปอย่างถูกต้อง ในตัวอย่าง append_frame() จะใช้ฟังก์ชัน skb_put() เพื่อใช้สงวนพื้นที่ส่วนหนึ่งของบัฟเฟอร์สำหรับข้อมูลเพื่อที่จะใช้ส่งข้อมูลลงไปยังเลเยอร์ล่างต่อไป

ใน append_frame() ที่ฟังก์ชัน alloc_skb() จะจองหน่วยความจำเท่ากับจำนวนค่าของตัวแปร len หรือจำนวนความยาวข้อมูล ดังรูปที่ 1 โดยข้อมูลทีส่วนหัวและส่วนข้อมูลจะเท่ากับศูนย์และจะมีความยาวเท่ากับตัวแปร len

ฟังก์ชัน skb_put() ในรูปที่ 4 จะนำข้อมูลไปไว้ยังส่วนท้ายของบัฟเฟอร์โดยใช้ฟังก์ชัน memcpy() ในการทำงานในระดับเน็ตเวิร์กส่วนมากนั้นจะในการส่งไปยังที่จุดเริ่มต้นของเฟรมโดยลำดับจากการเพิ่มเฮดเดอร์ลงแพ็กเก็ต เช่นฟังก์ชัน skb_push() ที่จะยอมให้มีการส่งข้อมูลผ่านหน่วยความจำไปยังส่วนเริ่มต้นของบัฟเฟอร์ที่สงวนไว้ดังรูปที่ 5

หลังจากที่บัฟเฟอร์ได้มีการจับจองหน่วยความจำบางส่วนของหน่วยความจำ ต่อไปจะเป็นฟังก์ชันที่ใช้สงวนหน่วยความจำส่วนหนึ่งในบัฟเฟอร์ซึ่งสามารถที่จะเรียนฟังก์ชันนี้ก่อนที่จะใส่ข้อมูลลองบัฟเฟอร์โดยใช้ฟังก์ชัน skb_reserve() ในรูปที่ 2 โดยมีตัวอย่างดังนี้

```
skb=alloc_skb(len+headspace, GFP_KERNEL);
skb_reserve(skb, headspace);
skb_put(skb,len);
memcpy_fromfs(skb->data,data,len);
pass_to_m_protocol(skb);
```

ในระบบยูนิกซ์เช่น BSD จะใช้การเชื่อมโยงบัฟเฟอร์ขนาดเล็กๆเข้าด้วยกันสำหรับเน็ตเวิร์กบัฟเฟอร์ แต่ในระบบของลีนุกซ์เลือกที่จะใช้ลิเนียบัฟเฟอร์หรือบัฟเฟอร์ที่มีความต่อเนื่องกันเพราะว่าการใช้ลิเนียบัฟเฟอร์ (Linear Buffer) นั้นจะสามารถที่จะช่วยให้การทำงานมีประสิทธิภาพมากขึ้น

รูปที่ 5 เริ่มการจับจองหน่วยความจำโดยฟังก์ชัน alloc_skb()

รูปที่ 6 เมื่อได้มีการสงวนหน่วยความจำบางส่วนโดยใช้ skb_reserve()

รูปที่ 7 หลังจากการใส่ข้อมูลลงในบัฟเฟอร์

รูปที่ 8 จากการเรียกใช้ฟังก์ชัน skb_put()

รูปที่ 9 จากการเรียกใช้ฟังก์ชัน skb_push()

2.5.3 รูทีน (Routine) ที่สนับสนุนการทำงานในระดับสูง

รูปแบบโครงสร้างของการจับจองและจัดการบัฟเฟอร์สำหรับซ็อกเก็ตโดยจะมีการทำงานตามสัญญาณ (Signal) และทางเลือกในการทำงานนั้นจะมีอยู่สองส่วนที่จะต้องสร้างให้ง่ายต่อการใช้งานกับโพรโตคอลโดยส่วนมาก

ฟังก์ชัน sock_queue_rev_skb() ฟังก์ชันนี้จะใช้ในการควบคุมข้อมูลที่เข้ามาดังตัวอย่าง

```
sk=my_find_socket(whatever);
if(sock_queue_rcv_skb(sk,skb)==-1)
```

```
{
        myproto_stats.dropped++;
        kfree_skb(skb,FREE_READ);
        return;
}
```

ในฟังก์ชันนี้จะป้องกันการอ่านข้อมูลจากซ็อกเก็ตที่มีขนาดใหญ่ หลังจากอ่านข้อมูลเรียบร้อยแล้วฟังก์ชันจะส่งต่อไปยังโปรแกรมในระดับสูงเพื่อให้โปรแกรมระดับสูงสามารถอ่านข้อมูลได้เร็วเพียงพอ อย่างการทำงานของ TCP/IP หรือโพรโตคอลที่ส่งข้ามระบบเน็ตเวิร์กจะบอกให้เครื่องที่ทำการส่งแบ่งข้อมูลที่จะส่งให้เหมาะสม

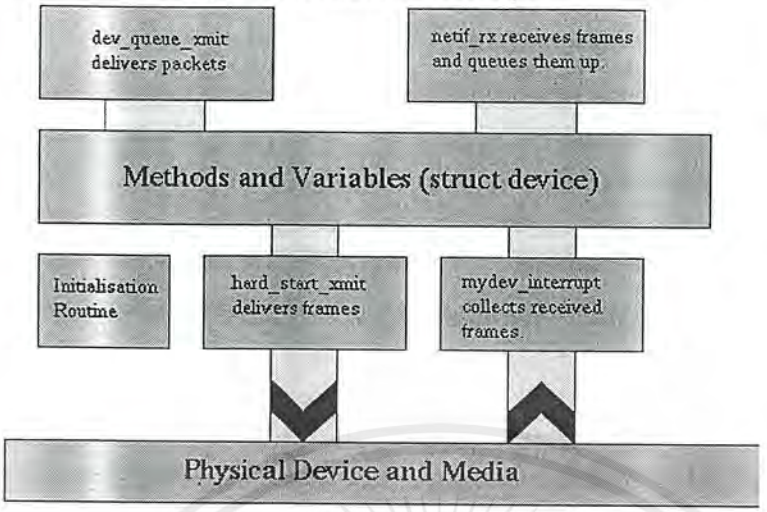
ในฝั่งส่ง ฟังก์ชัน sock_alloc_send_skb() จะมีสัญญาณที่ควบคุมการส่ง ไม่ว่าจะในการรับส่งแบบรอหรือไม่รอรับการตอบสนอง จนกระทั่งข้อมูลที่อยู่ในบัฟเฟอร์ที่จะส่งจะไม่สามารถส่งข้อมูลทั้งหมดในหน่วยความจำในอุปกรณ์ที่มีความเร็วในการส่งที่ช้าได้ โดยทั้งหมดนี้โพรโตคอลระดับสูงส่วนมากจากเป็นตัวจดการดังตัวอย่าง

```
skb=sock_alloc_send_skb(sk,....)
if(skb==NULL)
        return -err;
skb->sk=sk;
skb_reserve(skb, headroom);
skb_put(skb,len);
memcpy(skb->data, data, len);
protocol_do_something(skb);
```

2.5.4 อุปกรณ์เน็ตเวิร์ก

ในอุปกรณ์เน็ตเวิร์กบนระบบลีนุกซ์จะมีการติดต่อผ่านหลายฟังก์ชันที่เหมือนกัน โดยจะเป็นไปตามหลักการเชิงวัตถุ (Object-Oriented) โดยจะให้แต่ละดีไวซ์เป็นวัตถุและมีลำดับของฟังก์ชันหรือเมธอด (Method) เป็นส่วนหนึ่งของโครงสร้างข้อมูล ในแต่ละเมธอดจะถูกเรียกโดยดีไวซ์ของมันเอง ดังในไฟล์ drivers/net/skeleton.c จะบรรจุโครงร่างของเน็ตเวิร์กดีไวซ์ไดรเวอร์

2.5.5 โครงสร้างพื้นฐานของระบบเน็ตเวิร์กในลีนุกซ์



รูปที่ 10 โครงสร้างพื้นฐานระบบเน็ตเวิร์กในลีนุกซ์

ในแต่ละเน็ตเวิร์กดีไวซ์จะเริ่มการส่งข้อมูลของเน็ตเวิร์กบัฟเฟอร์จากโพรโตตอลระดับสูงอุปกรณ์ตัวนำ และจะถูกรับรวมทั้งถอดรหัสจากการตอบสนองอุปกรณ์ฮาร์ดแวร์ ในเฟรมที่เข้ามาจะกลับมาสู่เน็ตเวิร์กบัฟเฟอร์ และจะถูกกำหนดชื่อที่ใช้อ้างเฟรมโดยโพรโตคอลและส่งไปยัง netif_rx() โดยส่งต่อไปยังโพรโตคอลเพื่อประมวลผลต่อไป

ในแต่ละเมธอดจะเตรียมการควบคุมการหยุด เริ่ม ควบคุม และ ห่อหุ้มแพ็กเก็ต ทั้งหมดนี้จะมีการควบคุมข้อมูลต่างๆ ที่เป็นกลุ่มของแต่ละดีไวซ์โดยการจัดการแต่ละดีไวซ์เอง

2.6 ทฤษฎีของหน่วยความจำแฟลช

2.6.1 การอิมพลีเมนต์เรซิเดนต์แฟลชอาร์เรย์ (Implementing Resident Flash Array)

ความเปลี่ยนแปลงของ ที่เป็นความต้องการของสถาปัตยกรรมคอมพิวเตอร์ ที่เป็น ระบบเอมเบ็ดเด็ด (Embedded) กับสถาปัตยกรรม คอมพิวเตอร์ส่วนบุคคล (PC) หลักๆ คือ

- สถาปัตยกรรมไม่ได้ถูกจำกัดมาจากสิ่งที่ถูกทำขึ้นมาก่อนด้วย สถาปัตยกรรมของหน่วยความจำที่มีอยู่แล้ว เพื่อที่จะได้มาซึ่งความต้องการของ ผู้ใช้ขั้นสุดท้าย ของระบบเอมเบ็ดเด็ด
- เป็นสถาปัตยกรรมที่อิสระในการเข้าร่วม และรับเอาความก้าวหน้าของเทคโนโลยีทั้งทางด้าน ซอฟต์แวร์ และ ฮาร์ดแวร์ ไปพร้อมๆ กับการปกป้อง ผู้ใช้ขั้นสุดท้าย ในเรื่องการลงทุนทางด้าน ฮาร์ดแวร์ พื้นฐานในเบื้องต้น

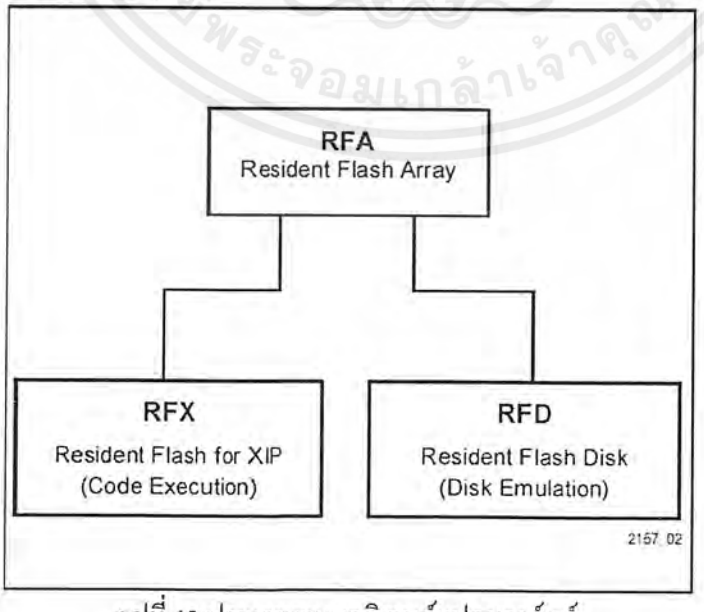
ซึ่งการทำให้สถาปัตยกรรมนี้ตอบสนองกับความต้องการดังข้างต้นนั้น ต้องการสิ่งที่เป็นสื่อในการเก็บ ข้อมูลที่แตกต่างไปจากพีซีทั่วๆ ไปอันได้แก่ อุปกรณ์ เช่น รอม (ROM), ฟลอปปี้

ดิสก์ (Floppy Disk) หรือ ฮาร์ดดิสก์ ซึ่งทางออก ก็คือหน่วยความจำแฟลช ซึ่งดูการเปรียบเทียบของทั้งสองสถาปัตยกรรมได้จากรูปที่ 1

APPLICATION	DATA MANIPULATION	CODE EXECUTION	FILE & CODE STORAGE
 Desktops	DRAM	DRAM/ROM	FDD/HDD
 Embedded	DRAM	FLASH	FLASH - Resident Disk - Flash Card - Flash Drive

รูปที่ 11 การเปรียบเทียบระหว่างสถาปัตยกรรม เอมเบ็ดเด็ด กับ พีซี

หน่วยความจำแฟลช จะถูกฝังลงไปใน ระบบ เอมเบ็ดเด็ด ซึ่งในการออกแบบ หน่วยความจำแฟลช ทั่วๆ ไปจะถูกจัดให้เป็น อาร์เรย์ ซึ่งถูกเรียกว่า เรซิเดนต์แฟลชอาร์เรย์ (RFA) ซึ่ง RFA ยังแบ่งออกเป็น 2 ประเภทด้วยกันตามรูปที่ 2 ประเภทแรกเพื่อที่จะเก็บ โค้ด ที่ใช้ในการ เอ็กซีคิวต์ (Execute) หรือ ที่เรียกว่า eXecute-In-Place (XIP) เช่น "ROMed DOS" ใน เมโมรีแมพ (Menory Map) ของระบบ การใช้งานในลักษณะนี้ ของ RFA เรียกว่า Resident Flash for XIP (RFX) ประเภทที่สองของ RFA ทำในลักษณะ เก็บ ไฟล์ และ โปรแกรม เหมือนหน้าที่การทำงานของ ฮาร์ดดิสก์ ทั่วๆไป ซึ่งในการใช้งานในลักษณะนี้ของ RFA เรียกว่า เรซิเดนต์แฟลชดิสก์ (Resident Flash Disk (RFD))



รูปที่ 12 ประเภทของ เรซิเดนต์แฟลชอาร์เรย์

จะเห็นว่าการทำงานทั้งสองแบบ จะแตกต่างกันโดยสิ้นเชิง โดยที่แบบแรกจะทำงานในพื้นที่ของหน่วยความจำหลักในระบบ โดยแบบที่ 2 จะทำงานในลักษณะหน่วยความจำสำรองหรือหน่วยความจำภายนอก (External Memory) ซึ่งจะเห็นว่าพื้นที่แบบที่สองนั้นจะสามารถที่จะมีเพิ่มเติมได้ไม่จำกัด (สมมติไม่มีข้อจำกัดของ แฟลชเทคโนโลยี และระบบที่จะมาจัดการกับ แฟลช เช่น ระบบปฏิบัติการ) โดยที่แบบแรกจะมีปัญหาเรื่อง พื้นที่ที่จะขยายได้ลำบากเนื่องจากข้อกำหนดของพื้นที่ของหน่วยความจำระบบ ดังนั้นจะเห็นว่าเราสามารถเพิ่มเติมความสามารถของระบบโดยเพิ่มเติม ซอฟต์แวร์ ให้กับระบบ ได้ง่ายถ้าใช้การทำงานแบบที่สอง หรือ แบบ RFD ซึ่งทำให้เราเลือกที่จะใช้งานในแบบที่สอง จึงไม่ขอกล่าวถึงแบบแรก

2.6.2 เรซิเดนต์แฟลชดิสก์ (Resident Flash Disk)

โซลิดเสต็ดดิสก์ (Solid-State Disk) จะกลายเป็นความจริงด้วยราคาที่ลดลงของ หน่วยความจำแฟลช ระบบต่างๆย่อมเลือกที่จะใช้ โซลิดเสต็ดมีเดีย (Solid-State Media) ที่มีความสามารถหลายอย่าง และมีความเชื่อถือได้ซึ่งจะแตกต่างจาก แมกเนติกมีเดีย (Magnetic Media) คือพวกฮาร์ดดิสก์หรือฟลอปปี้ดิสก์

ข้อได้เปรียบหลายประการของ โซลิดเสต็ดมีเดีย ที่เหนือกว่า แมกเนติกมีเดีย ได้แก่

- ต้องการกำลังงานต่ำ
- ประสิทธิภาพสูง
- น้ำหนักเบาและมีขนาดเล็ก
- มีความเชื่อถือได้สูง
- มีความทนทานสูง
- อัตราการโอนถ่ายข้อมูลสูงและการเข้าถึงเร็ว

โดยส่วนมากแล้วงานทางด้าน เอมเบ็ดเด็ด ไม่ต้องการความจุที่สูงมากนักเหมือนที่มีในฮาร์ดดิสก์ทั่วๆไป ซึ่ง แฟลชอาร์เรย์ ก็เป็นอุปกรณ์เก็บข้อมูลที่ให้ประโยชน์คุ้มค่ากับราคาเมื่อเทียบกับ ฮาร์ดดิสก์และเมื่อสถาปัตยกรรมพื้นฐานของระบบ พีซี กลายเป็นการเจาะจงไปใช้เฉพาะงานและเปลี่ยนแปลงไปใช้เพื่อการ เอ็กซีคิวต์ กับ ซอฟต์แวร์ ที่เฉพาะทางโดยตรง ดังนั้นจึงความต้องการของการ สตอเรจ จึงลดลง ทำให้การเปลี่ยนแปลงไปเป็น โซลิดเสต็ดหน่วยความจำแฟลช สตอเรจ กลายมาเป็นทางเลือกที่น่าสนใจ

2.6.3 ความต้องการทางด้านซอฟต์แวร์ของ RFD

มีความแตกต่างคร่าวๆ ระหว่าง หน่วยความจำแฟลช กับ แมกเนติกมีเดีย และซอฟต์แวร์ที่ใช้ในการจัดการกับความแตกต่างกันนี้

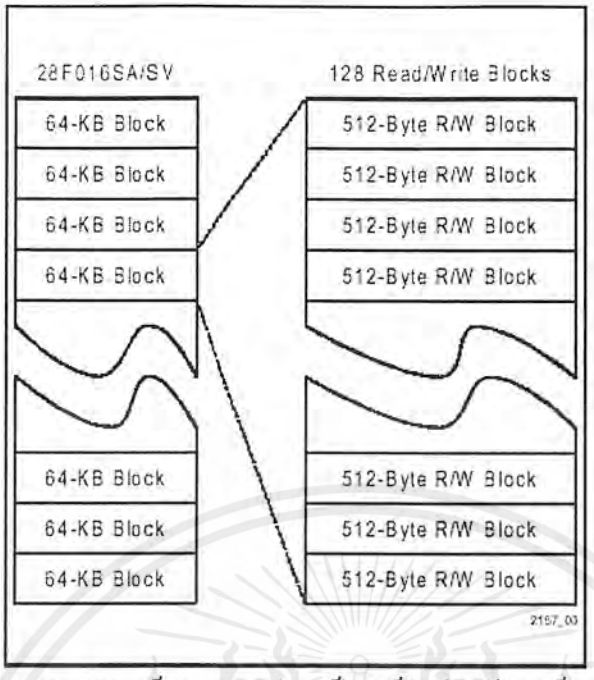
2.6.3.1 แมกเนติกมีเดีย : ระบบที่มีพื้นฐานของ DOS มีความสามารถเขียนและลบแบบสุ่มเมื่อมีการเข้าถึง แมกเนติกมีเดีย ซึ่งหมายถึงว่าเซ็กเตอร์ (Sector) ของ แมกเนติกมีเดีย สามารถที่จะ ถูกโปรแกรมเตรียมไว้หรือ โปรแกรม เข้าไปใหม่ และลบได้เวลาไหนก็ได้ File

Allocation Table (FAT) ถูกใช้ในการจัดการ ไฟล์ ซึ่งตารางนี้จะถูกปรับปรุง ได้เวลาไหนก็ได้เมื่อมีการ สร้าง แก้ไขหรือลบ

2.6.3.2 **แฟลชมีเดีย :** สถาปัตยกรรมของ หน่วยความจำแฟลช มีความแตกต่างอย่างมากจาก แมกเนติกมีเดีย ซึ่ง หน่วยความจำแฟลช ถูกแบ่งย่อยออกเป็น Erase Blocks เมื่อ หน่วยความจำแฟลช ถูกลบอย่างสมบูรณ์แล้ว ทุกๆ บิต จะถูก เซต ให้เป็น "1" อุปกรณ์ที่ถูกลบอย่างสมบูรณ์แล้วจะยอมให้เขียนลงไปได้ตำแหน่งไหนก็ได้ แต่อย่างไรก็ตามถ้าครั้งใดที่มี บิตใดๆในบล็อกถูกเขียนให้เป็น "0" ไว้แล้ว ต้องมีการลบทั้ง บล็อก ก่อนที่ บิต เหล่านั้นจะสามารถเขียนกลับให้เป็น "1" ซึ่งลักษณะสมบัติดังกล่าวนี้ของ หน่วยความจำแฟลช จะป้องกันการใช้งาน FAT โดยตรง และต้องการ แฟลชมีเดียแมเนเจอร์ (Flash Media Manager) เพื่อที่จะจัดการกับไฟล์ ใน FRD

2.6.3.3 **แฟลชมีเดียแมเนเจอร์ :** ในปัจจุบันมี แฟลชมีเดียแมเนเจอร์ อยู่หลากหลาย ซึ่ง แต่ละ แฟลชมีเดียแมเนเจอร์ จะมีลักษณะสำคัญที่ปรับปรุงไปทางด้านการนำไปประยุกต์ใช้งานที่เจาะจง สำหรับการนำไปแทน ดิสก์ไดรฟ์บนสภาวะแวดล้อม DOS แนะนำให้เลือกใช้ Flash Translation Layer (FTL)

2.6.3.4 **Flash Translation Layer** : Flash Translation Layer เป็น เซ็กเตอร์เบสมีเดียแมเนเจอร์ (Sector-Base Media Manager) ซึ่งจะใช้ ระบบไฟล์แบบเซ็กเตอร์ ที่มีอยู่ เช่น FAT ของ DOS ในการจัดการกับระดับบน โดยการ แปลง (Translate) จะรับการร้องขอจาก DOS (หรือ ระบบปฏิบัติการใดๆ ที่นี้ยกตัวอย่าง DOS เพื่อจะเห็นได้ชัดเจน) FTL จะแสดงเป็น ไดร์ฟแบบเซ็กเตอร์ ปกติกับ ซอฟต์แวร์ระดับบน ซึ่งซอฟต์แวร์ ในระดับบนจะคาดหวังที่จะแก้ไข เซ็กเตอร์ เหล่านี้ได้ ณ เวลาใดๆ แฟลชมีเดีย มีความต้องการในการลบบล็อก ที่ต้องการจะ แก้ไขก่อนที่จะมีการ แก้ไข ไฟล์ ใน บล็อก นั้นลงไปได้ เมื่อ ซอฟต์แวร์ระดับบนต้องการที่จะเขียน เซ็กเตอร์ใหม่ FTL จะ รีแมพ (Remap) การร้องขอนั้นไปยัง พื้นที่ว่างของ แฟลช ซึ่งการ รีแมพเซ็กเตอร์นี้กระทำเป็น บล็อกอ่านเขียนลอจิคัล (Logical Read/Write Blocks) แทน ฟิสิกส์คัลเซ็กเตอร์ ซึ่ง FTL จะแบ่งย่อย บล็อก ของ แฟลช ไปเป็น บล็อกอ่านเขียน (Read/Write Blocks) ที่เล็กกว่า ซึ่งในแต่ละ บล็อกอ่านเขียน นี้จะมีขนาดเป็นเซ็กเตอร์ (ทั่วๆไปเท่ากับ 512 bytes) ตามตัวอย่างรูปที่ 3



รูปที่ 13 การ แมพบล็อก ของ แฟลช เป็น บล็อกอ่านเขียน(เซ็กเตอร์)

2.6.4 การอิมพลีเมนต์ FTL

มีการอิมพลีเมนต์ เบื้องต้น 3 อย่างที่ถูกเตรียมไว้ให้จาก ผู้พัฒนา FTL ในปัจจุบัน (ดูจากรูปที่4.) ซึ่งการ อิมพลีเมนต์ นั้นถูกแบ่งเป็น ชั้นๆ ซึ่งมีรายละเอียดคร่าวๆดังนี้

2.6.4.1 FTL สำหรับ card service :

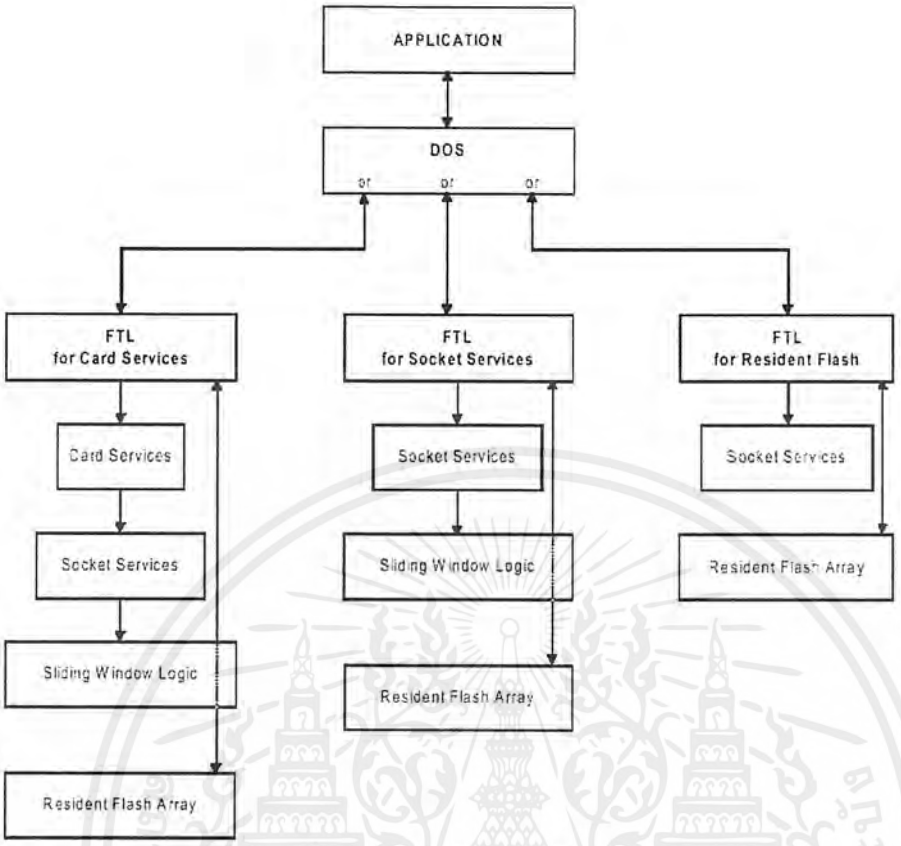
- **Card Services** : คือ ชั้นที่มีความรับผิดชอบในการจับจอง ทรัพยากร ของระบบสำหรับ การ์ดพีซีเอ็มซีไอเอ (PCMCIA Card) ด้วย ไคลเอ็นต์ไดรเวอร์ (Client Driver) ที่เหมาะสม Card Service จัดการกับการสนับสนุนสำหรับเมโมรี หรือ ไอโอ (I/O) การ์ด ใดๆ ซึ่ง ซอฟต์แวร์ ในชั้นนี้ต้องใช้ในระบบที่ต้องการ การสนับสนุนสำหรับ ไอโอ การ์ด
- **Socket Services :** เป็นการกำหนดมาตรฐานของ พีซีเอ็มซีไอเอ ซึ่งดั้งเดิมได้ถูกพัฒนาขึ้นเพื่อการคอนฟิกกูเร พีซีเอ็มซีไอเอซ็อกเก็ต ซึ่ง พีซีเอ็มซีไอเอซ็อกเก็ต จะสอบถาม การ์ดที่ใส่เข้าไป สำหรับ ข้อมูลแอตทริบิวต์ (Attribute Information) ของ การ์ด ที่ใส่ เนื่องจาก RFD เป็นแบบ "Non-Removable Card"

ดังนั้นทุกๆ แอตทริบิวต์อินฟอร์เมชัน เกี่ยวกับ RFD จะคงที่ RFD Socket Service ควรจะง่ายในการรวมค่าคงที่เหล่านี้ทุกค่าไว้ด้วยกัน

- **Sliding Window Logic** : เป็นฮาร์ดแวร์ที่ควบคุม วินโดว์ (Window)ในการ เข้าถึงแฟลช ระบบที่ต้องการที่จะสนับสนุน พีซีเอ็มซีไอเอเมโมรีการ์ด ต้องการพีซีเอ็มซีไอเอซ็อกเก็ตคอนโทรลเลอร์เช่น 82365SL ซึ่ง คอนโทรลเลอร์นี้จัดการการเข้าถึง แฟลช ด้วย Sliding Window ซึ่ง RFD สามารถที่จะเข้าถึงด้วย Sliding Window
- **Resident Flash** : ขออ้างถึงส่วนประกอบที่เป็น ฟิสิกส์คัล ของหน่วยความจำแฟลชทั่วๆไป

2.6.4.2 **FTL สำหรับ Socket Services** : ข้อแตกต่างเพียงอย่างเดียวระหว่าง FTL สำหรับ Socket Services และ FTL สำหรับ Card Services คือ การกำจัด ชั้นของ Card Services ออกไป ซึ่งในการเอา ชั้นของ Card Services ออกนั้นเป็นการสละการสนับสนุน สำหรับ ไอโอการ์ด ออกไป ถ้าระบบใช้เพียงแต่ เมโมรีการ์ด หรือ RFD ไม่มีความต้องการในการใช้ Card Services และสามารถรักษา Memory Space ที่มีค่าอย่างยิ่งโดยไม่ติดตั้งซอฟต์แวร์ Card Service

2.6.4.3 **FTL สำหรับ Resident Flash** : มีความเป็นไปได้ในการเข้าถึง เรซิเดนต์แฟลช โดยไม่ต้องใช้ Sliding Window ซึ่งทำได้โดยการ แมพ เรซิเดนต์แฟลช ไปเป็น Extended Memory วิธีการนี้เป็นแนวทางที่สิ้นเปลืองน้อยที่สุด โดยการกำจัด ลอจิกคสำหรับการทำ Sliding Window แนวทางนี้มีความต้องการออกแบบ FTL ที่พิเศษในการเข้าถึง หน่วยความจำแฟลชนั่นคือการ แมพ ไปสู่ Extended Memory แบบลิเนีย



รูปที่ 14 การอิมพลีเมนต์เอฟทีแอล

2.6.4.4 **Wear Leveling** : Wear Leveling คือ เทคนิคที่ถูกใช้โดยผู้พัฒนา FTL เพื่อที่จะขยายอายุการใช้งานของอุปกรณ์แฟลช โดยการกระจายการลบไปยังทุกๆ บล็อก ของ อุปกรณ์ สมมติว่าไม่มี Wear Leveling แล้วมีการ ปรับปรุงไฟล์ เดิมทุกๆชั่วโมง จะทำให้มีการเขียนลงบางตำแหน่งของ บล็อกของแฟลช ทุกๆ ชั่วโมง จะทำให้บล็อก นั้นมี Cycle Rate เท่ากับ 1 Cycle ต่อชั่วโมง โดยที่ บล็อกอื่นเป็น ศูนย์ เนื่องจาก บล็อก ต้องมีการลบในทุกครั้งที่มีการเขียนไฟล์ แต่ถ้า ระบบไฟล์มีการ อิมพลีเมนต์ Wear Leveling บนทุกๆ บล็อก ของ อุปกรณ์จะทำให้มีการเขียน ไฟล์ลง บล็อก ที่ต่างกันในทุกๆ ชั่วโมง จึงทำให้มี Cycle Rate เท่ากับ 1 Cycle ต่อจำนวนบล็อก ชั่วโมง Flash Device ส่วนใหญ่จะลบได้ อย่างน้อย 100,000 ครั้งต่อ บล็อกและระบบไฟล์ FTL ส่วนใหญ่ที่อิมพลีเมนต์ Wear Leveling จะมี cycle rate อย่างต่ำ 1,000,000 ครั้งต่อ บล็อก

2.6.4.5 **FTL Installation** : ผู้พัฒนาFTL ส่วนใหญ่ ยอมให้มีความยืดหยุ่นในการ โหลด FTL ได้หลายวิธี เช่น โหลด เป็นดีไวซ์ไดรเวอร์, TSR หรือ ROM BIOS Extension

2.6.5 ความต้องการทางด้าน ฮาร์ดแวร์ของ RFD

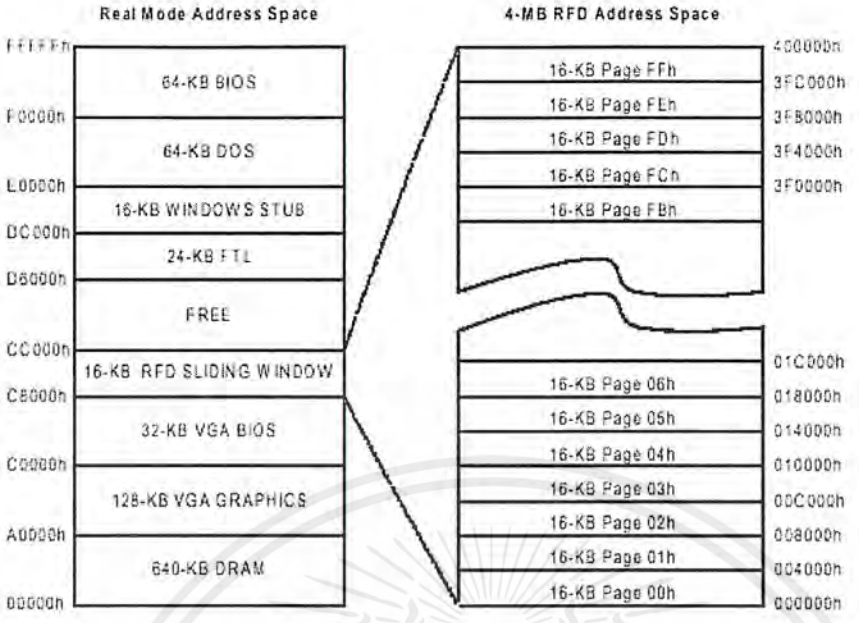
อธิบายถึงความต้องการ ทางด้าน ฮาร์ดแวร์ สำหรับการ อิมพลีเมนต์ RFD แนวคิดนี้สามารถที่จะรวมเข้ากับ Chip-Set หรือ ASIC Design เพื่อที่จะได้ใช้ข้อได้เปรียบของหน่วยความจำแฟลชให้กับระบบฝังตัว

2.6.5.1 **หน่วยความจำแฟลช** : หน่วยความจำแฟลช คือ หน่วยความจำประเภทหนึ่งที่มีข้อเด่นอยู่หลายประการเหนือกว่า หน่วยความจำชนิดอื่น เช่น แรม, รอม, พรอม (PROM), อีพรอม (EPROM) ฯลฯ ข้อดีเช่นสามารถอ่านเขียนได้เหมือน แรมโดยที่ ไม่จำเป็น ต้อง มีไฟเลี้ยงในการเก็บรักษาข้อมูล มีการอ่านเขียนเป็น บล็อก ซึ่งมีผู้ผลิตหลายรายที่ผลิต หน่วยความจำแฟลช ออกจำหน่าย ผู้นำทางด้านนี้คือ อินเทล (Intel), ซัมซุง (Samsung), เอเอ็มดี (AMD)

2.6.5.2 **แฟลชคอนโทรลลอจิก** : ลอจิกต่างๆที่ ต้องการในการ อินเทอร์เฟซ หน่วยความจำแฟลช กับระบบ มีส่วนหลักๆได้แก่

- **บัฟเฟอร์** : ป้องกันการช่วงชิง บัส ข้อมูล เมื่อ Cycle การอ่านถูกตามด้วย Cycle การเขียน ต้องการ ลอจิก Enable สำหรับบัฟเฟอร์ ด้วย
- **WE# Control** : ส่วนเพิ่มเติมในการควบคุม บัฟเฟอร์ คือ ต้องการที่จะ Pull High WE# ไว้ก่อน ซึ่งระยะเวลาในการ Pull High ขึ้นอยู่กับ Specification ของ หน่วยความจำแฟลชที่ใช้อยู่
- **Byte Writes** : แนวทาง FTL ส่วนใหญ่ต้องการความสามารถในการทำ Byte write ซึ่ง RFD ควรจะ ถูก อิมพลีเมนต์ ด้วย 8 บิต บัสข้อมูล

2.6.5.3 **Sliding Window ลอจิก** : รูปที่ 15 แสดง เมโมรีแมพ ทั่วๆไป สำหรับ ระบบ รวมถึง หน่วยความจำ RFD ด้วย ใน เรียลโหมด นั้น การ เข้าถึงหน่วยความจำ ต้องอยู่ภายใต้ ขอบเขต 1 Mbytes ในการเข้าถึง RFD เมโมรีอาร์เรย์ จำเป็นต้อง แมพเมโมรี นั้นเป็น เพจ (Pages) และอ่าน เพจ เหล่านั้นโดยตำแหน่งของ RFD Sliding Window ภายใต้ ขอบเขตของ 1 Mbytes



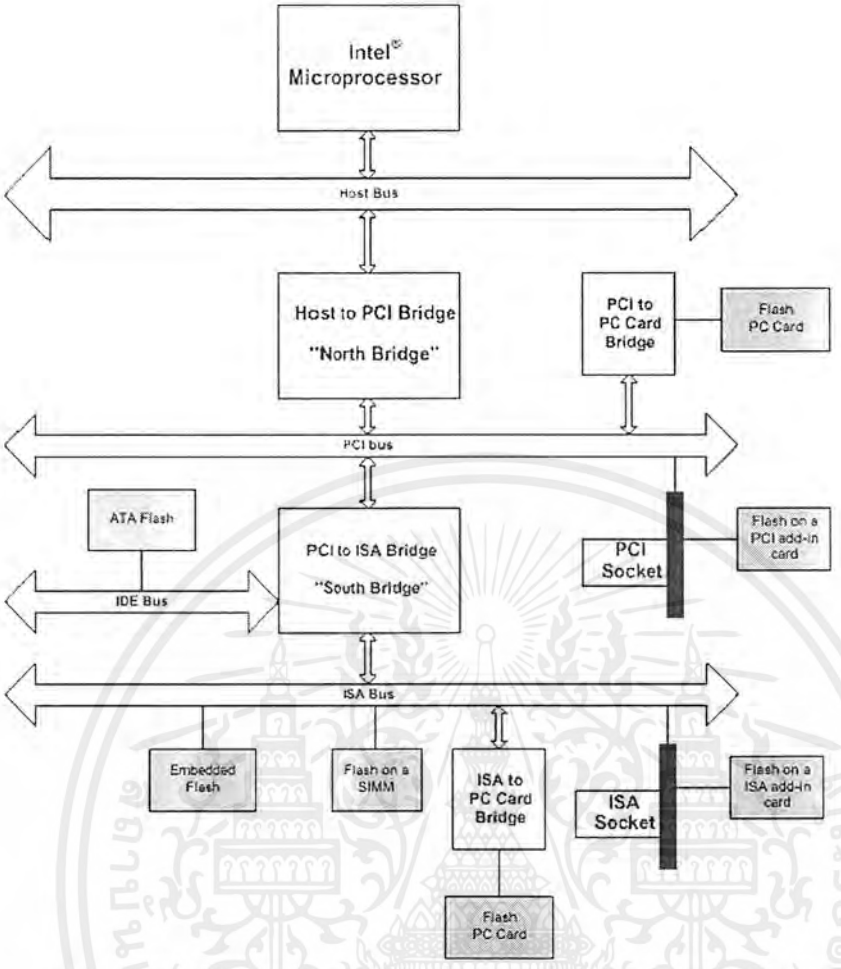
รูปที่ 15 แมพของหน่วยความจำระบบ

2.6.6 การออกแบบหน่วยความจำแฟลช ภายใต้ สถาปัตยกรรมของอินเทล

โพรเซสเซอร์ และ Chip-Set ของอินเทลได้ จัดเตรียมจำนวนของการการ อินเทอร์เฟซ กับ หน่วยความจำแฟลชไว้แล้ว ขึ้นอยู่กับการ ออกแบบ อินเทอร์เฟซ ซึ่ง หน่วยความจำแฟลชควรจะเก็บ ไบออส, ระบบปฏิบัติการ, ข้อมูล หรือ สิ่งเหล่านี้รวมกัน มีอยู่หลากหลายวิธีในการเชื่อมต่อหน่วยความจำแฟลชเข้ากับ สถาปัตยกรรมของ อินเทล เพื่อให้ได้มาซึ่ง หน่วยเก็บข้อมูลที่ทนทานและไม่สูญหายของข้อมูล โดยแสดง 6 วิธีในการ อินเทอร์เฟซ หน่วยความจำแฟลชกับ สถาปัตยกรรม ของ อินเทล ตามรูปที่ 6 ซึ่งได้แก่

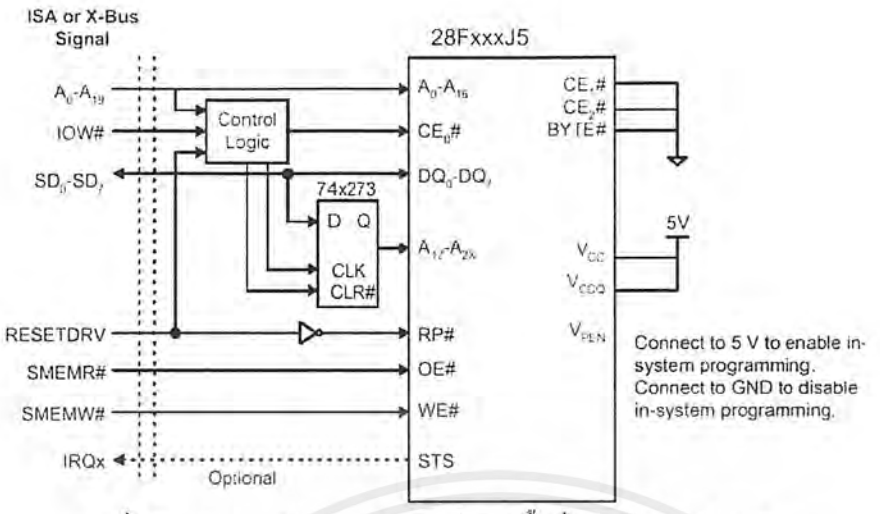
- **ISA bus** :เมื่อ หน่วยความจำแฟลช ถูก อมเบ็ดเด็ด ไว้บนแผงวงจร
- **ISA bus** : เมื่อ เป็นลักษณะของ Add-In การ์ด
- **ISA bus** : เมื่อ เป็นลักษณะของ SIMM
- **PCI bus** : เมื่อ เป็นลักษณะของ Add-In การ์ด
- **ISA/ISA bus** : เมื่อ เป็นลักษณะของ พีซีเอ็มซีไอเอ อินเทอร์เฟซ

ในที่นี้จะขออธิบายการทำงานของ การ อินเทอร์เฟซ กับ ไอเอสเอบัส เนื่องจากเป็นวิธีที่ง่ายที่สุดต่อการอิมพลีเมนต์ โดยมีวิธีการพื้นฐานในการ อินเทอร์เฟซ อยู่ 2 วิธีการที่ ง่ายต่อการ อิมพลีเมนต์ คือ วิธีการ เพจ หน่วยความจำแฟลช ไปสู่ พื้นที่ของExpansion ROM หรือการต่อ หน่วยความจำแฟลชเป็น แบบ ลีเนีย เมโมรี ไปยัง ไอเอสเอบัส



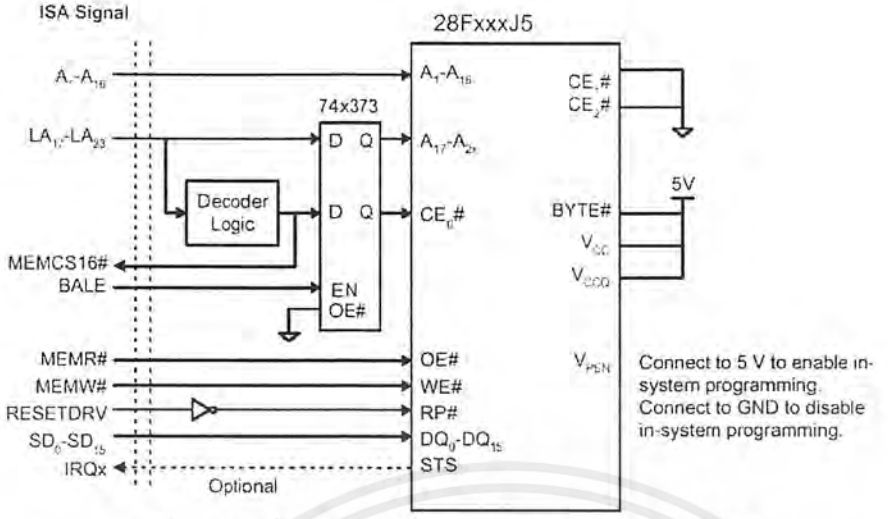
รูปที่ 16. การอินเทอร์เฟซหน่วยความจำแฟลชกับสถาปัตยกรรมอินเทล

2.6.6.1 การเพจหน่วยความจำแฟลช ไปสู่พื้นที่ Expansion ROM :ระบบเหล่านั้นซึ่งไม่สามารถ Extended Memory บน ไอเอสเอบัส แต่ยังสามารถใช้ อาร์เรย์ ของ แฟลช ขนาดใหญ่ๆ ได้อยู่ ช่วง แอดเดรส ของ ไอเอสเอ จาก 0C0000H – 0xDFFFFH ถูกออกแบบ มาสำหรับ Expansion ROMs ซึ่ง รอม นี้จะรวมไว้กับ Expansion Boards เมื่อระบบบูต, ไบออส จะ แสกน ช่วงแอดเดรส นี้เพื่อที่จะหาอุปกรณ์ที่ต้องการ Initial และมีความสามารถในการบูตได้ รูปที่ 17 แสดงการอิมพลีเมนต์ เทคนิคนี้



รูปที่ 17 การเพจหน่วยความจำแฟลชไปสู่ พื้นที่ของเอ็กแพนชันรอม

2.6.6.2 การเชื่อมต่อหน่วยความจำแฟลชเป็น ลีเนียเมโมรีบนไอเอสเอบัส : ไอเอสเอบัส ถูกออกแบบมาเพื่อที่จะ เป็น บัส ที่ขยาย เมโมรี และ ไอโอ รูปที่ 18 แสดงการเชื่อมต่อทั่วๆไประหว่าง หน่วยความจำแฟลช กับ ISA bus เทคนิคนี้ใช้งานมานานพอๆ กับ แนวทางของการทำงาน Extended Memory ของระบบ (100000H-FFFFFFH) กับ ไอเอสเอ วิธีการนี้มีประโยชน์ในลักษณะที่ ข้อมูลที่เก็บอยู่ในหน่วยความจำแฟลชจะมีอยู่เสมอ ดังนั้นข้อมูลที่เก็บอยู่ใน หน่วยความจำแฟลช สามารถที่จะ เข้าถึง ณ เวลาใดๆ เท่าที่ต้องการ แต่ พีซี Chip-Set/ไบออส ทั่วๆไปได้รวมแนวทางนี้ของ การเข้าถึง Extended Memory ถึง ดีแรม (DRAM) ซึ่ง เทคนิคนี้จะจำกัดขนาดของ แฟลช ที่ 16 Mbyte ซึ่ง หน่วยความจำแฟลช จะถูก แมพ แบบลีเนีย เหนือ ตำแหน่งของ เมโมรี สูงสุด ภายใต้เมโมรีแมพ



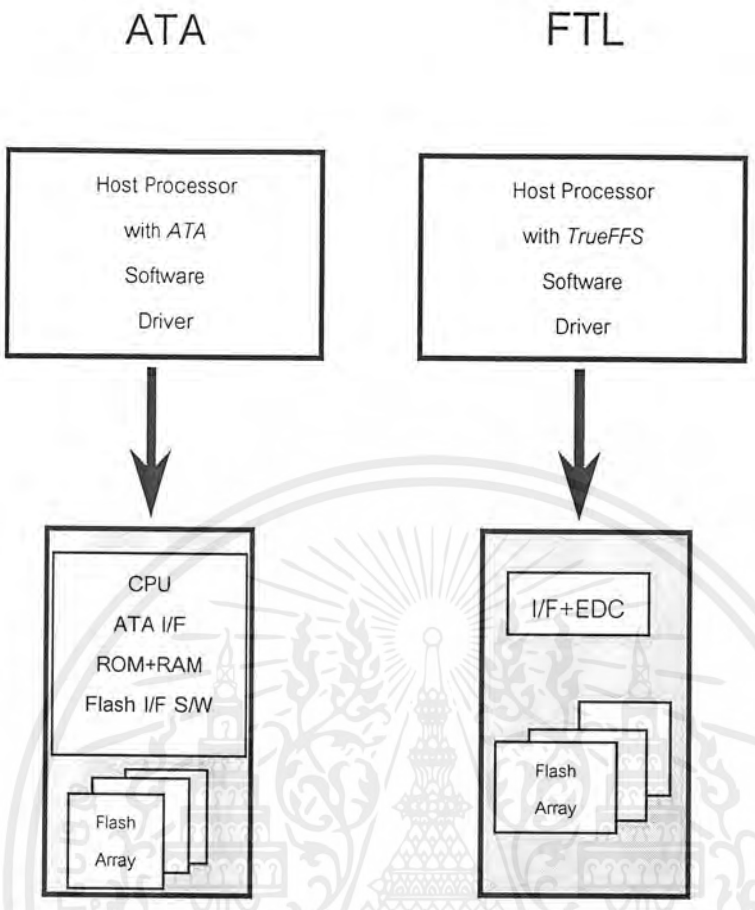
รูปที่ 18 การ เชื่อมต่อ หน่วยความจำแฟลชไปยัง ไอเอสเอบัส

2.6.7 สองแนวทางของคีย์ (Key) เทคโนโลยี ที่เป็นทางเลือก

รูปที่ 19 แสดงการเปรียบเทียบการ อิมพลีเมนต์ ระหว่าง ATA และ FTL

2.6.7.1 **ATA** : จะเป็นการทำงานทางด้าน ฮาร์ดแวร์ เป็นหลัก โดยมีข้อได้เปรียบทางด้านไม่ต้องการซอฟต์แวร์ เพิ่มเติม และง่ายต่อการอินเทอร์เฟซ ด้วยแต่ เนื่องจากเป็นการอิมพลีเมนต์ ด้วย ฮาร์ดแวร์ เป็นหลัก จึงเป็นการ Fixed ใจเรื่องของการ อินเทอร์เฟซ เกินไป

2.6.7.2 **FTL** : จะเป็นการทำงานด้วย ซอฟต์แวร์ เป็นหลัก โดยจะเป็นการจำลองการทำงานของ ATA เพื่อให้ จำลองการทำงานของฮาร์ดดิสก์ได้อย่างสมบูรณ์ ซึ่งมีข้อได้เปรียบอยู่หลายอย่าง คือ ต้นทุนต่ำเนื่องจากไม่ต้องการ ฮาร์ดแวร์ เช่น แรมหรือ คอนโทรลเลอร์, สิ้นเปลืองพลังงานต่ำกว่ามาก และมีประสิทธิภาพสูงเหมาะสมกับงานที่มีข้อจำกัดเรื่องพื้นที่



รูปที่ 19 การ เปรียบเทียบการอิมพลีเมนต์แบบ เอทีเอ และ เอฟทีแอล

2.6.8 ดิสก์ออนชิป (Disk on Chip)

2.6.8.1 ลักษณะสมบัติ

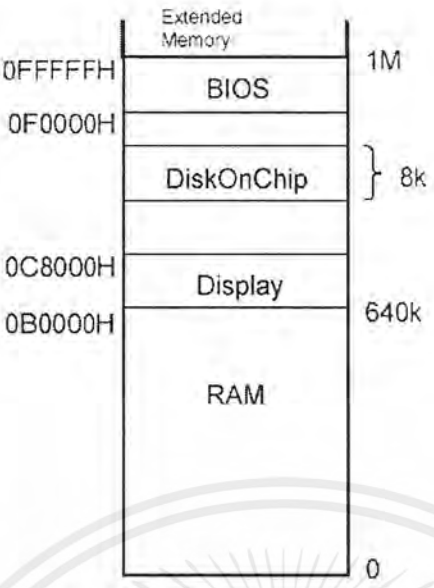
- เป็น ชิป เดี่ยว ง่ายต่อการรวมเข้าไปในระบบ
- ใช้กำลังงานต่ำ 3.3 โวลต์ หรือ 5 โวลต์
- Small form factor (รุ่นตัวถัง 32 ขา TSOP-II หรือ DIP)
- EDC/ECC สำหรับความเชื่อถือได้ของข้อมูลสูง
- มีความสามารถในการบูต
- สนับสนุนระบบปฏิบัติการอย่างกว้างขวาง เช่น DOS, Windows, pSOS+, QNX, VxWorks, Linux, ฯลฯ
- ขนาดของเมโมรีวินโดว์ เล็ก (8K)

- Embedded *TrueFFS* Technology เตรียมไว้สำหรับ จำลอง ความสามารถ อ่านและเขียน , บูต ของ ฮาร์ดดิสก์ ไว้อย่างสมบูรณ์
- ใช้เทคนิค Wear Leveling รุ่นที่ 3
- การ แมพ Bad Block อย่างอัตโนมัติ
- การกู้คืนจาก Power Loss
- สนับสนุน โพรเซสเซอร์สำคัญๆ เช่น x86, MediaGX, PowerPC, 68K, MIPS, SHx, StrongARM ฯลฯ

2.6.8.2 การประยุกต์ใช้งาน

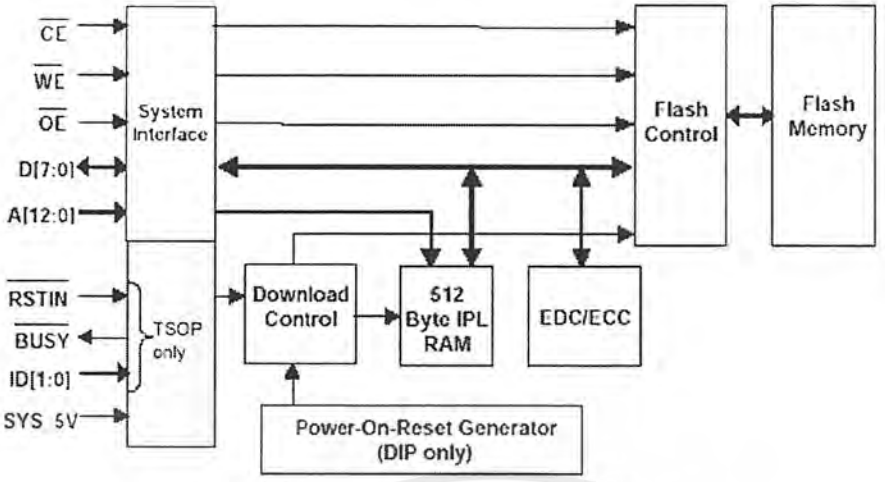
- Embedded Systems
- อุปกรณ์ เพื่อการเข้าถึง อินเทอร์เน็ต
- Set-Top Box หรือ Web TV, Web Browser
- Thin Client, Network Computers
- Routers, Networking
- Web-Phones, Car-PC, DVD, HPC
- Point Of Sale, Industrial PC's
- Telecom, Medical

2.6.8.3 **การออกแบบเพื่อใช้งานดิสก์ออนชิปบนสถาปัตยกรรมพีซี** : เมื่อต้องการใช้งาน ดิสก์ออนชิป ภายใต้สถาปัตยกรรม พีซี ดิสก์ออนชิป ควรจะถูกเชื่อมต่อไปยัง เมโมรีวินโดว์ 8K ภายในช่วงของ เมโมรี Expansion BIOS ซึ่งปกติจะอยู่ที่ 0C8000H – 0EFFFH ระหว่างการ บูต ระบบ ดิสก์ออนชิป จะ โหลด เฟิร์มแวร์ (Firmware) ของมันลงสู่ เมโมรี ของ ระบบและติดตั้งตัวมันเองเป็น ดิสก์ไดรฟ์ ของระบบ เมื่อ ระบบปฏิบัติการ ถูก โหลดดิสก์ออนชิป จะถูกรู้จักเป็น ดิสก์มาตรฐาน ธรรมดา รูปที่ 20 แสดงการ แมพ ดิสก์ออนชิป ลงใน เมโมรีแมพ ของระบบหลังจากการ รีเซต ระบบ ไบออส จะ เอ็กซีคิวต์ POST (POWER ON SELF-TEST) เป็น อันดับแรก จากนั้น ไบออส จะค้นหาอุปกรณ์ Expansion ROM เมื่อ เจอ ดิสก์ออนชิป จะทำให้ ไบออส เอ็กซีคิวต์ Initial Program Loader (IPL) โค้ด ที่อยู่ใน ดิสก์ออนชิป เอง ซึ่ง โค้ด นี้เองที่ โหลด TrueFFS ลง เมโมรี ของระบบเพื่อ ติดตั้ง ดิสก์ออนชิป เป็น ดิสก์ ของ ระบบ จากนั้นจะโอน การควบคุมกลับไปให้ ไบออส ของ ระบบอีกที



รูปที่ 20 การแมพดิสก์ออนชิปลงสู่แมพของหน่วยความจำระบบ

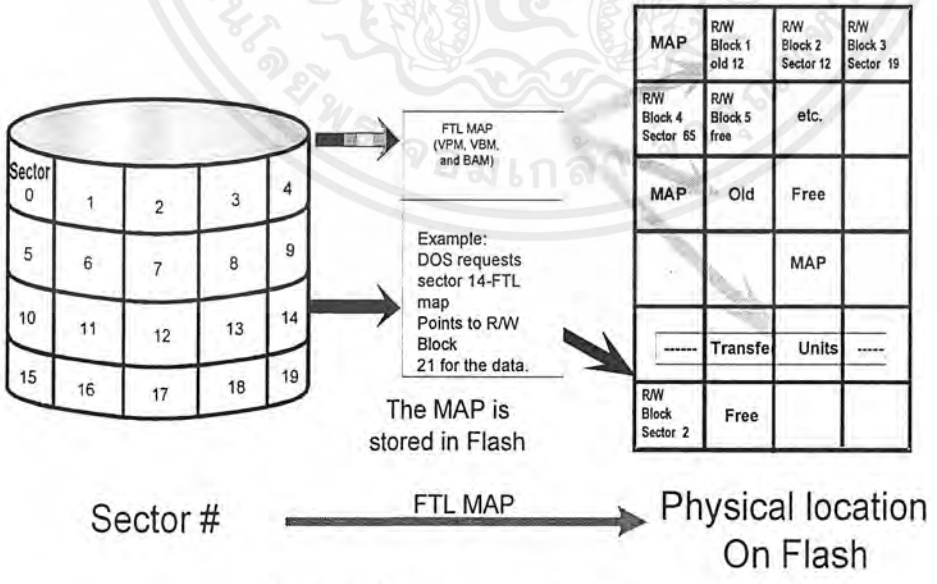
2.6.8.4 **ขั้นตอนการทำงานภายในตัวดิสก์ออนชิป** : ในช่วงของ การ Power-up จะทำให้ ดิสก์ออนชิป ดาวน์โหลด IPL ขนาด 512 bytes แรกของ หน่วยความจำแฟลชภายในตัวมันลงสู่ แรม ภายในของมันเอง จากนั้น ดิสก์ออนชิป จะทำการตรวจสอบความถูกต้องของข้อมูล ด้วย EDC ภายในตัวมัน ถ้าพบความผิดพลาด ดิสก์ออนชิป จะ ดาวน์โหลด สำเนาซ้ำซ้อนของ IPL ที่เก็บไว้ใน เพจ ถัดไปของหน่วยความจำแฟลชภายใน ซึ่งกระบวนการ ดาวน์โหลด ทั้งหมดจะใช้ เวลาน้อยกว่า 1 mS ซึ่งจะไม่ยอมให้มีการ เข้าถึง ดิสก์ออนชิป เมื่อการ ดาวน์โหลด IPL ยังไม่สมบูรณ์ เมื่อกระบวนการ ดาวน์โหลด เสร็จสิ้นสมบูรณ์แล้ว ระบบโฮสต์ (Host System) (พีซีหรืออื่นๆ) จะเอ็กซีคิวต์โค้ด IPL จาก แรม ภายใน ดิสก์ออนชิป ซึ่ง โค้ดนี้จะ โหลด TrueFFS จาก หน่วยความจำแฟลช ลงสู่ หน่วยความจำของระบบ



รูปที่ 21 ไดอะแกรมของดิสก์ออนชิป

2.6.8.5 **EDC/ECC** : ดิสก์ออนชิป ใช้ Reed – Solomon ECC/EDC อัลกอริธึม (Algorithm) เพื่อที่จะแน่ใจความเชื่อถือได้ของข้อมูลสูง ในแต่ละครั้งที่มีการ เขียนข้อมูลลง แฟลช จะมีการเขียน 6 bytes โค้ด ลงไปด้วย ซึ่งแต่ละครั้งที่มีการอ่านข้อมูลกลับมาจาก แฟลช จะมีการคำนวณ 6 bytes โค้ดใหม่ TrueFFS ใช้ โค้ด นี้ในการตรวจสอบความถูกต้อง

2.6.8.6 **TrueFFS** : TrueFFS คือ FTL Flash File System Management ซึ่งก็คือการ อิมพลีเมนต์ หน้าที่การทำงานของ ATA ด้วย ซอฟต์แวร์ ซึ่งมีหน้าที่การทำงานตาม รูปที่ 22



รูปที่ 22 หน้าที่การทำงานของ ทรูเอฟเอฟเอส

2.6.9 คอมแพคแฟลช (Compact Flash)

คอมแพคแฟลช เป็น เมโมรีการ์ด โซลิดเสต็ดแฟลชเมโมรีสนับสนุนมาตรฐานการใช้งานกับ มาตรฐานของ พีซีการ์ดเอทีเอ (PC Card ATA) และสนับสนุนการทำงานใน โหมด ของ True IDE ซึ่ง สอดคล้องกับการทำงานของ IDE ดิสก์ไดรฟ์ การเข้าถึงหน่วยความจำแฟลช ทุกอย่างภายใน การ์ด นั้นจะต้องผ่านทาง อินเทลลิเจนต์คอนโทรลเลอร์ (Intelligent Controller) บน การ์ด

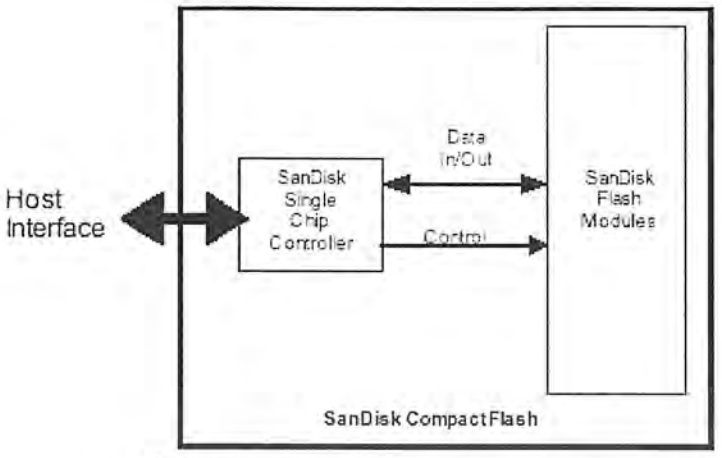
2.6.9.1 ลักษณะสมบัติ

- ความจุเพิ่มได้ถึง 48 Mbytes
- เข้ากันได้กับ โพรโตคอล พีซีการ์ดเอทีเอ
- เข้ากันได้กับโหมดของ True IDE
- กินพลังงานต่ำ
- ทนทานสูง น้ำหนักเบา
- ทำงานที่ 5โวลต์ หรือ 3โวลต์
- แก้ไขความผิดพลาดอัตโนมัติ
- สนับสนุน คำสั่ง เพาเวอร์ดาวน์ และ โหมด พักผ่อน (Sleep Mode)

2.6.9.2 การประยุกต์ใช้งาน

- เก็บรูปภาพในกล้องดิจิตอล
- พีซี มือถือ
- Palm พีซี
- PDA
- Smart Cellular Phones

2.6.9.3 โครงสร้างภายในของ คอมแพคแฟลช : รูปที่ 23 แสดงโครงสร้างภายในที่ประกอบไปด้วยส่วนประกอบสำคัญที่เป็นฮาร์ดแวร์คือ ส่วนที่เป็น อินเทลลิเจนต์คอนโทรลเลอร์ ซึ่งทำหน้าที่ในการจัดการกับ หน่วยความจำแฟลช ภายใน คอมแพคแฟลช ทุกอย่าง โดยการติดต่อหรือการเข้าถึง หน่วยความจำแฟลช ภายใน คอมแพคแฟลช ต้องทำผ่านทาง คอนโทรลเลอร์ ตัวนี้ด้วยคำสั่งที่เรียกว่าคำสั่ง ATA



รูปที่ 23 บล็อกไดอะแกรมของคอมแพคแฟลช

2.6.9.4 **ลักษณะทาง ฟิสิกส์คัลของ คอมแพคแฟลช** : รูปที่ 44 เป็นลักษณะทาง กายภาพของ คอมแพคแฟลช ซึ่งจะมีขนาดเล็กกว่า พีซีการ์ดครึ่งหนึ่ง

2.6.9.5 **การเข้าถึง หน่วยความจำแฟลช ภายในตัว คอมแพคแฟลช** : การเข้าถึง หน่วยความจำแฟลช ภายใน คอมแพคแฟลช จะต้องกระทำผ่านรีจิสเตอร์ ATA ภายใน คอนโทรลเลอร์ ของ คอมแพคแฟลช ด้วยคำสั่งของ ATA ซึ่ง ตารางที่ 2 แสดงรีจิสเตอร์ ATA พื้นฐานที่ใช้ในการเข้าถึงและ ตารางที่ 3 แสดง รายละเอียดของคำสั่ง ATA

2.6.9.6 **การเชื่อมต่อพื้นฐานสำหรับ คอมแพคแฟลช** : รูปที่ 24 แสดงการต่อใช้งาน คอมแพคแฟลช พื้นฐานโดยใช้ ข้อมูล ขนาด 8 บิต ซึ่งทำงานใน เมโมรีโหมด จะเห็นว่าสัญญาณ ที่ใช้ควบคุมการทำงานของ คอมแพคแฟลช ในการต่อใช้งานในลักษณะนี้จะ สอดคล้องกับการใช้งาน หน่วยความจำทั่วๆ ไป เพียงแต่การเข้าถึงต้องเข้าถึงผ่านทางรีจิสเตอร์ ATA ด้วยคำสั่ง ATA ดังแสดงตามหัวข้อที่ผ่านมา

บทที่ 3

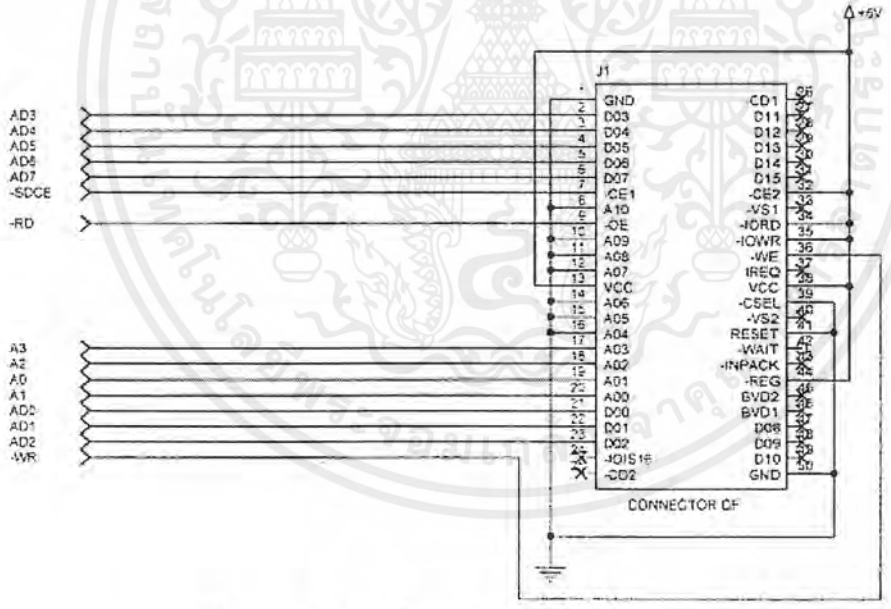
แนวปฏิบัติงานวิจัย

3.1. การ์ดไอเอสเอลีนุกซ์

การออกแบบ การ์ดไอเอสเอลีนุกซ์การ์ดหรืออุปกณ์ที่ทำหน้าที่เก็บระบบปฏิบัติการนั้น ในขั้นแรกมีความตั้งใจที่จะใช้หน่วยความจำแฟลชที่ เป็นชิปเดี่ยวๆ มาประกอบเป็นการ์ด แต่เนื่องจากว่ามีเทคโนโลยีที่มีอยู่ในปัจจุบันนั้น มีอุปกรณ์ที่เป็นหน่วยความจำแฟลชอยู่หลากหลาย ที่มีขนาดกะทัดรัดแต่มีความจุสูงเพียงพอกับการใช้งาน เช่นคอมแพคแฟลชและดิสก์ออนชิปเป็นต้น ซึ่งจากการเปรียบเทียบระหว่างราคากับหน่วยความจำแฟลชที่เป็นชิปจะมีราคาประหยัดกว่ามาก ในขณะที่ความจุสูง ซึ่งจะสอดคล้องกับวัตถุประสงค์ของการวิจัยด้วยคือต้องใช้งบประมาณให้น้อยที่สุดเท่าที่จะทำได้รวมถึงประหยัดพื้นที่และพลังงานด้วย

การทดลองการสร้างการ์ดจากหน่วยความจำแฟลชแบบคอมแพคแฟลชและดิสก์ออนชิป

3.1.1. การทดลองสร้างการ์ดจากคอมแพคแฟลช



รูปที่ 24.การเชื่อมต่อพื้นฐานสำหรับคอมแพคแฟลช

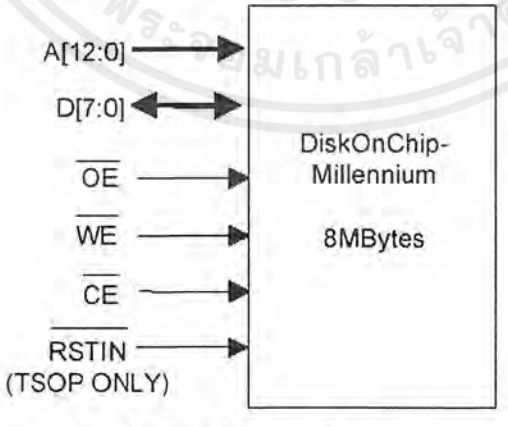
หน่วยความจำแฟลชแบบคอมแพคแฟลชเป็นหน่วยความจำที่น่าสนใจในการนำมาใช้เป็นหน่วยความจำสำรองแทนฮาร์ดิสก์เพราะมีข้อดีหลายๆอย่างรวมถึงถูกใช้เป็นหน่วยความจำสำหรับอุกกรณ์หลายๆชนิดในปัจจุบันอย่างแพร่หลายจึงทำให้หาซื้อได้ง่าย ในการวิจัยครั้งนี้มีการทดลองเพื่อการเข้าถึงคอมแพคแฟลชด้วยไมโครคอนโทรเลอร์ MCS-51 ดูก่อนเนื่องจากง่ายต่อการทดลองซึ่งมีการอินเทอร์เฟซง่ายๆตาม รูปที่ 24

ในการทดลองการเข้าถึงคอมแพคแฟลชมีการเข้าถึงที่ง่ายเหมือนๆกับการเข้าถึงแรมโดยการเข้าถึงทำได้โดยผ่านไมโครคอนโทลเลอร์ที่อยู่ภายในคอมแพคแฟลชเองจึงทำให้การเข้าถึงเพียงแต่ส่งคำสั่งไปให้กับคอมมานรีจิสเตอร์ จากนั้นก็ทำการอ่านข้อมูลจากรีจิสเตอร์ข้อมูล ซึ่งจะเห็นว่าการเข้าถึงทำได้ง่ายมากและไม่เปลืองโอเวอร์เฮดของระบบ

ความสามารถที่น่าสนใจของคอมแพคแฟลชอีกอย่างหนึ่งก็คือความสามารถในการต่อเข้าไปแทนฮาร์ดดิสก์ได้โดยตรงโดยที่ทำการต่อวงจรเพิ่มเล็กน้อย แต่ว่าจะผิดวัตถุประสงค์ของการวิจัย และคอมแพคแฟลชยังมีข้อเสียอยู่อย่างหนึ่งที่ทำให้การวิจัยครั้งนี้ไม่ตัดสินใจใช้คอมแพคแฟลชและไม่ทดลองนำคอมแพคแฟลชไปอินเทอร์เฟซกับไอเอสเอต่อคือ การอินเทอร์เฟซกับคอมแพคแฟลชต้องใช้ซ็อกเก็ตของคอมแพคแฟลชซึ่งมีขนาดเท่ากับ ซ็อกเก็ตของพีซีเอ็มซีไอเอ ซึ่งมีขนาดเล็กและการเชื่อมต่อวงจรค่อนข้างละเอียด ซึ่งซ็อกเก็ตของคอมแพคแฟลชเองก็มีราคาแพงและหาซื้อลำบากซึ่งในการทดลองนี้ใช้ซ็อกเก็ตของพีซีเอ็มซีไอเอแทนเนื่องจากพอหาได้ ดังนั้นจึงทำให้ค่าใช้จ่ายโดยรวมที่ใช้ในการสร้างการ์ดด้วยคอมแพคแฟลชจึงมีราคาสูง เนื่องจากราคาของซ็อกเก็ตสูงและการประกอบวงจรต้องใช้ความละเอียดสูงเพราะซ็อกเก็ตของคอมแพคแฟลชหรือพีซีเอ็มซีไอเอก็ตามมีขนาดเล็กและละเอียดมากจึงไม่เป็นทางเลือกที่คุ้มค่าและเหมาะสมสำหรับการวิจัยนี้

3.1.2. การทดลองสร้างการ์ดจากดิสก์ออนชิป

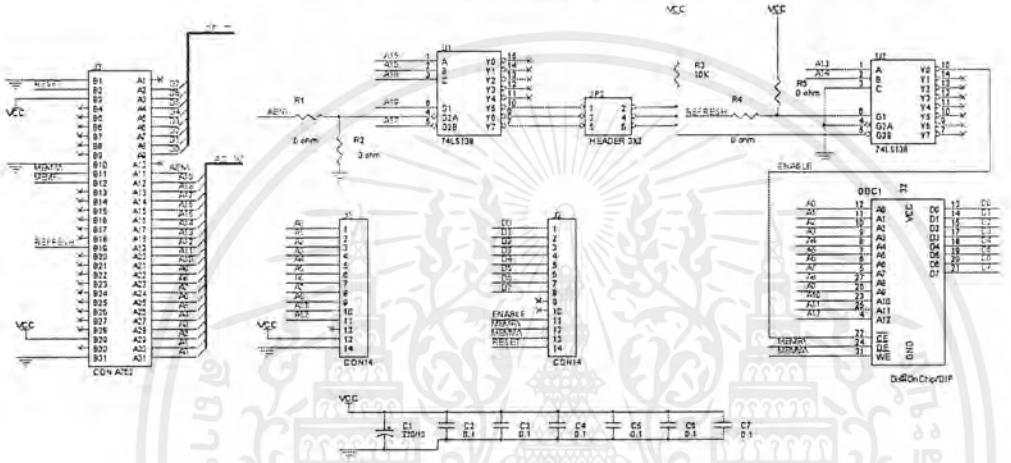
อุปกรณ์ที่เป็นหน่วยความจำแฟลชอีกประเภทคือดิสก์ออนชิปเป็นอุปกรณ์ที่มีต้นทุนของหน่วยความจำพอๆกับคอมแพคแฟลชเมื่อเทียบกับขนาดความจุแต่เนื่องจากข้อเสียของคอมแพคแฟลชจากที่กล่าวมาจาก 3.1.1 และจากการคำนวณจากค่าใช้จ่ายรวมของการ์ดเมื่อเปรียบเทียบทั้งสองชนิดแล้วจะพบว่าดิสก์ออนชิปมีค่าใช้จ่ายรวมต่ำกว่ามากจึงเป็นทางเลือกที่ดีที่สุดสำหรับการวิจัย



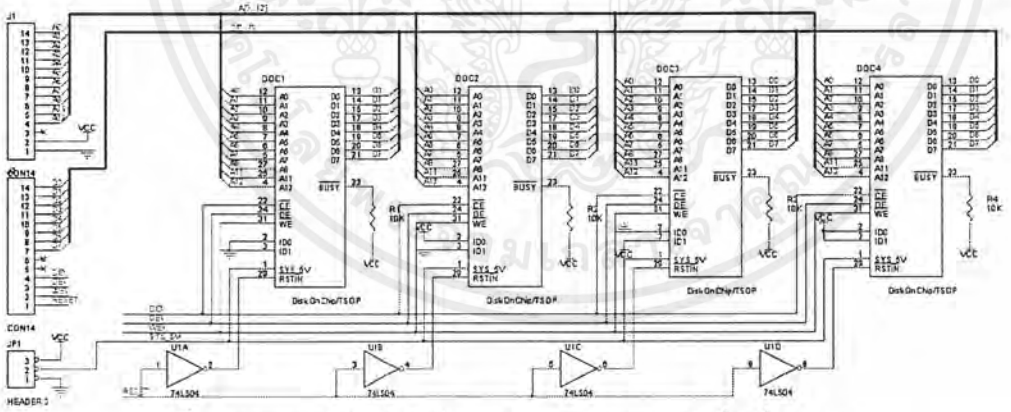
รูปที่ 25 การอินเทอร์ดิสก์ออนชิป

3.1.2.1. การออกแบบเพื่อใช้งานดิสก์ออนชิป

- ด้านฮาร์ดแวร์ : การใช้งาน ดิสก์ออนชิป ในระบบ ดิสก์ออนชิป จะถูกเชื่อมต่อด้วยมาตรฐานของการอินเทอร์เฟซหน่วยความจำ ด้วยสัญญาณที่ใช้สำหรับ การ อินเทอร์เฟซหน่วยความจำ ด้วย โดยทั่วๆไป ดิสก์ออนชิป จะถูก แมพ ไปไว้ในตำแหน่งที่ว่างขนาด 8 K ใดๆ ในแมพของ หน่วยความจำ ซึ่ง รูปที่ 25 แสดง การ อินเทอร์เฟซ พื้นฐานสำหรับ การใช้งาน ดิสก์ออนชิป รูปที่ 26 แสดงวงจรที่ใช้ในการวิจัย



รูปที่ 26 วงจรการใช้งานดิสก์ออนชิปบนไอเอสเอ



รูปที่ 27 วงจรการใช้งานดิสก์ออนชิปบนแบบหลายตัวเพื่อเพิ่มความจุ

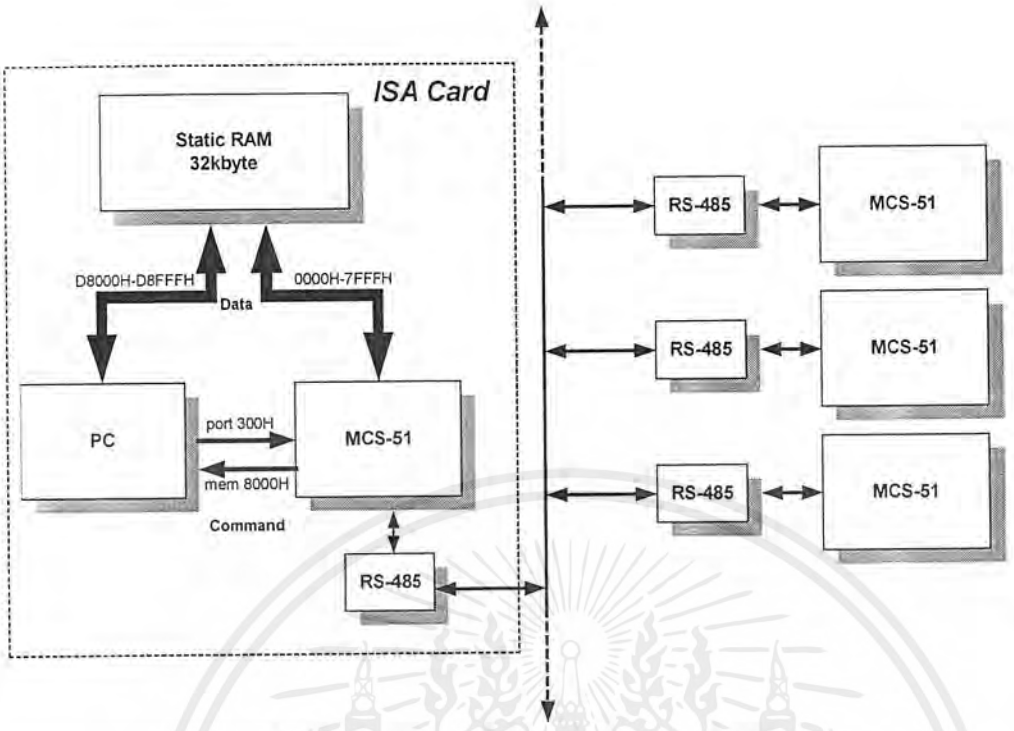
- การใช้งาน ดิสก์ออนชิปหลายๆ ตัว : ดิสก์ออนชิปที่มีแพ็กเกจ แบบ TSOP สามารถต่อเข้าด้วยกันถึง 4 ตัวตามวงจรตาม รูปที่ 27 ด้วยการขนานเข้าด้วยกันโดยไม่ต้องใช้ วงจรสำหรับ ดีโค้ด (Decode) เพิ่มเติม โดยใช้ ขา ID ของดิสก์ออนชิปเอง ในการระบุเจาะจงไปยัง ดิสก์ออนชิปแต่ละ

ตัวที่ต่อเข้าด้วยกัน โดยยังคงใช้ วินโดว์ของหน่วยความจำ เพียง 8K เท่าเดิม

3.1.2.2. ด้าน ซอฟต์แวร์ : ดิสก์ออนชิปใช้ มาตรฐาน TrueFFS เป็น Flash Memory Management ภายใต้การ ควบคุมของ TrueFFS ดิสก์ออนชิปทำงาน เหมือนกับอุปกรณ์เก็บข้อมูลมาตรฐานทั่วๆ ไป ซึ่งจำลองการทำงานของ ฮาร์ดดิสก์ ให้กับระบบอย่างสมบูรณ์ ซึ่งจะทำให้ การใช้งานภายใต้ระบบปฏิบัติการต่างๆ ดิสก์ออนชิปจะเข้าถึง เหมือนกับอุปกรณ์ประเภทบล็อก อื่นๆ ด้วย มาตรฐาน ระบบไฟล์ ที่มีอยู่ ซึ่งจะทำให้ โปรแกรมประยุกต์ สามารถ อ่านหรือเขียนจาก เซกเตอร์ใดๆ บน ดิสก์ออนชิปก็ได้ ซึ่ง หน่วยความจำแฟลชภายใน ดิสก์ออนชิปจะถูก เข้าถึง ด้วย TrueFFS อีกทีผ่านทาง 8K พื้นที่หน่วยความจำ ของระบบ

3.2. การออกแบบการ์ดในส่วนของการสื่อสาร RS-485

จาก รูปที่ 28 สำหรับส่วนที่จะใช้ในการสื่อสารระหว่างพีซีกับ MCS-51 นั้น ในส่วนของพีซี จะติดต่อผ่านทางไอดอสเอสล๊อต ตาม โดยจะติดต่อกับ MCS-51 ซึ่งทำหน้าที่เป็น มาสเตอร์ (Master) หรือว่าเป็นศูนย์กลาง ในการติดต่อกับ MCS-51 ตัวอื่น ในการติดต่อสื่อสารจะมีส่วนที่เป็น คำสั่ง และส่วนที่เป็น ข้อมูล ในส่วนที่เป็น คำสั่งทางฝั่งของ พีซี จะรับและส่งโดยใช้ ไอโอพอร์ต 300H ในการติดต่อกับ MCS-51 โดย พอร์ต นี้จะเป็นทั้ง อินพุต และเอาท์พุต ใน พอร์ตเดียวกัน ในส่วนของฝั่ง MCS-51 ตัว มาสเตอร์ จะใช้ เมโมรีตำแหน่ง 8000H ในการรับ คำสั่งจากพีซี และส่ง คำสั่ง ไปยัง พีซี ซึ่ง ตำแหน่งนี้จะเป็นทั้ง อินพุต และ เอาท์พุต เช่นเดียวกับการ เข้าถึงหน่วยความจำ



รูปที่ 28 การ์ดไอเอสเอในส่วนที่ติดต่อกับ MCS-51

ในส่วนของ ข้อมูล จะแลกเปลี่ยนกันโดยใช้ หน่วยความจำซึ่งเป็น สเตติกแรม (Static RAM) ร่วมกัน โดยทางฝั่งของ พีซี จะทำการแมพ สเตติกแรม ไว้ที่ ตำแหน่ง D8000H-D8FFFH ซึ่งสามารถอ้าง หน่วยความจำได้ขนาด 2 Kbyte แรกของ สเตติกแรม ส่วนทางฝั่งของ MCS-51 สามารถอ้าง หน่วยความจำได้ทั้ง 32 Kbyte ของ สเตติกแรม โดยใช้ ตำแหน่ง ตั้งแต่ 0000H-7FFFH ส่วน ตำแหน่งที่ 8000H จะใช้เป็น คำสั่งของตำแหน่งดังนั้นการรับข้อมูล ระหว่าง พีซี กับ MCS-51 จึงจำต้องมีส่วนที่เป็น คำสั่ง ในการควบคุมการรับส่ง ข้อมูล โดยปกติแล้วได้ทำการออกแบบให้ MCS-51 เป็นผู้ครอบครอง สเตติกแรม ไว้ก่อนเมื่อ พีซี ต้องการใช้ สเตติกแรม ในการรับส่งข้อมูล ก็จำเป็นที่ต้องร้องขอมาที่ MCS-51 ก่อน รอจนกว่า MCS-51 จะ รับรู้ (Acknowledge) จึงจะสามารถใช้ static RAM ได้ แต่ในกรณีที่เกิดการผิดพลาดขึ้นโดย พีซี ใช้ สเตติกแรม พร้อมกับ MCS-51 ในกรณีนี้ได้ทำการออกแบบให้ พีซี มี ลำดับความสำคัญ (Priority) ที่สูงกว่า MCS-51 คือส่วนของ MCS-51 จะถูกตัดออกจากระบบ

ในส่วนของคำสั่งซึ่งทางฝั่ง พีซี ใช้ ไอโอพอร์ต ในการติดต่อกับ MCS-51 เมื่อ พีซี ส่งคำสั่งออกไปได้มีการออกแบบให้เกิดการอินเทอร์รัปต์ ขึ้นที่ MCS-51 ซึ่งใช้ขา INT0 ทำให้ MCS-51 ไม่จำเป็นต้องเขียนโปรแกรมให้วนลูปรอรับ คำสั่งจาก พีซี อยู่ตลอดเวลา ในส่วนของ พีซี ก็เช่นเดียวกันเมื่อ MCS-51 ส่ง คำสั่ง มาก็จะเกิด อินเทอร์รัปต์ ขึ้นโดยได้ออกแบบโดยใช้ขา Interrupt Request(IRQ) โดยในวงจรที่ออกแบบได้ใช้ IRQ7 ซึ่งว่างอยู่

ส่วนของ MCS-51 ที่ติดต่อสื่อสารกันแบบมัลติโพรเซสเซอร์นั้น ได้ออกแบบให้ติดต่อกันผ่านทางขา RXD และ TXD เหมือนการสื่อสารอนุกรมทั่วๆไป แต่ได้มีการเพิ่มส่วนของ RS-485 ซึ่งเป็นส่วนที่ใช้ขยายสัญญาณให้แรงขึ้นสามารถส่งไปได้ไกล อีกทั้งยังสามารถต่อไปยัง

โหนดอื่นๆได้หลายๆ โหนดที่จุดเดียวกัน โดย MCS-51 ส่วนที่เป็น สลาฟ (Slave) จะสามารถติดต่อกับ พีซี โดยต้องผ่านตัว มาสเตอร์ อีกที่หนึ่ง และ พีซี จะติดต่อกับ สลาฟ ก็ต้องผ่าน มาสเตอร์เช่นกัน

3.3. การเขียนโปรแกรมดีไวซ์ไดรเวอร์หรือเคอร์เนลโมดูลบนระบบปฏิบัติการลีนุกซ์

การเขียนโปรแกรมดีไวซ์ไดรเวอร์มีอยู่หลายส่วนด้วยกันในที่จนี้จะกล่าวถึง การเขียนโปรแกรมเน็ตเวิร์กดีไวซ์ไดรเวอร์ โดยจะเริ่มจากการโปรแกรมประเภทโมดูล

3.3.1. การเขียนโปรแกรมโมดูล

ในการเขียนโมดูลนั้นมีอยู่สองส่วนหลักๆด้วยกันคือ

3.3.1.1. **int init_modules(void)** เป็นฟังก์ชันที่จะถูกเรียกใช้งานจากระบบเมื่อมีการติดตั้งโมดูล โดยฟังก์ชันนี้จะใช้ในการติดตั้งค่าหรือเตรียมพร้อมการทางานของโมดูลนั้น การเรียกใช้งานโมดูลจะใช้คำสั่ง insmod <modules.o>

3.3.1.2. **void cleanup_modules(void)** เป็นฟังก์ชันที่จะถูกเรียกใช้งานเมื่อได้มีการกำจัดโมดูลออกจากระบบ ในฟังก์ชันนี้โดยส่วนมากจะทำหน้าที่จัดการให้ระบบมีสภาพเหมือนเดิม โดยการใช้คำสั่ง rmmod <modules>

ตัวอย่าง hello.c

```
#define MODULES
#include <linux/modules.h>
int init_modules(void)
{
        printk("initialize hello !\n");
        return 0;
}
void cleanup_modules(void)
{
        printk("cleanup hello moduel\n");
}
```

การคอมไพล์และการใช้งาน

```
#gcc –c hello.c
#insmod hello.o
```

```
initialize hello !
#rmmod hello
cleanup hello module
```

3.3.2. การสร้างเน็ตเวิร์กดีไวซ์ไดรเวอร์โมดูล

เป็นโมดูลที่จะให้บริการกับระบบเกี่ยวกับ การสื่อสารผ่านระบบเน็ตเวิร์กที่อุปกรณ์นั้นอยู่ โดยดีไวซ์นั้นจะมีมาตรฐานการติดต่อกับระบบของลีนุกซ์ดังนี้

เริ่มจากการสร้างตัวแปรสตรัคเจอร์ที่ใช้ติดต่อกับเคอร์เนล คือ struct device ซึ่งจะมีรายละเอียดดังนี้

```
/* หน้าที่ 310 – 318  The Device Struct Detail */
ตัวอย่างการติดตั้งค่าให้กับสตรัคเจอร์ดีไวซ์
static struct device dev_n485=
{
        "n485", // กำหนดชื่อให้กับดีไวซ์
0, 0, 0, 0, 0, 0, 0, 0, 0,
NULL,
n485_init // กำหนดฟังก์ชันที่ใช้ทำหน้าที่ติดตั้งค่าเริ่มต้น
};
```

จากนั้นจะใช้ฟังก์ชัน void register_netdev(struct device) ในการติดตั้งไดรเวอร์นี้ เช่น

```
        if(result = register_netdev(&dev_n485))
        {
                printk("net485: cann't register net device\n");
                return -1;
        }
```

เมื่อมีการติดตั้งเรียบร้อยแล้วและไดรเวอร์จะเริ่มทำงานโดยการที่เคอร์เนลจะเรียนฟังก์ชัน n485_init(struct device) จากในสตรัคเจอร์ n485_dev {} ซึ่งในฟังก์ชัน init n485_init(struct device) จะมีการติดตั้งค่าต่างๆดังโค้ด

```
int n485_init(struct device *dev)
{
        net485_t *net485=0;
        dev->open = n485_open;
        dev->stop = n485_close;
        //dev->set_config = n485_config;
        dev->hard_start_xmit = n485_start_xmit;
```

```
        dev->do_ioctl = n485_ioctl;
        dev->get_stats = n485_getstats;
        //dev->rebuild_header = n485_rebuild_header;
        dev->flags |= IFF_NOARP;

        dev_init_buffers(dev);

        dev->mtu = NET485_MTU;
        dev->type = ARPHRD_SLIP;
        dev->tx_queue_len = 5;

        dev->priv = kmalloc(sizeof(net485_t), GFP_KERNEL);
        if (dev->priv == NULL)
                return -ENOMEM;
        memset(dev->priv, 0, sizeof(net485_t));
        net485 = (net485_t*)dev->priv;
        net485->stats = kmalloc(sizeof(struct net_device_stats),GFP_KERNEL);
        if(net485->stats == NULL)
        {
                kfree(dev->priv);
                dev->priv = NULL;
                return -ENOMEM;
        }
        return 0;
}
```

ในฟังก์ชัน void n485_init(struct dev) จะทำหน้าที่ติดตั้งฟังก์ชันพอยน์เตอร์ และค่าต้องให้กับเน็ตเวิร์กดีไวซ์นี้ ซึ่งมีฟังก์ชันดังนี้

```
int n485_init(struct device *dev);
static int n485_open(struct device *dev);
static int n485_close(struct device *dev);
static int n485_start_xmit(struct sk_buff *skb,struct device *dev);
static int n485_rebuild_header(void *buf, struct device *dev,unsigned long raddr,struct sk_buff
*skb);
//static int n485_hard_header(struct sk_buff *skb,struct device *dev,unsigned short type,
```

```
//void *daddr,void *sddr,unsigned len);
static struct net_device_stats* n485_getstats(struct device *dev);
static int n485_config(struct device *dev,struct ifmap *map);
static int n485_ioctl(struct device *dev, struct ifreq *ifr,int cmd);
//static void n485_multicast(struct device *dev);
//static int n485_macaddr(struct device *dev,void *addr);
```

จากข้างต้นที่กล่าวมาเมื่อได้เริ่มติดตั้งโมดูลโดยใช้คำสั่ง insmod แล้ว ให้ใช้คำสั่ง ifconfig เพื่อดูข้อมูลต่างของไดรเวอร์ดังนี้ ifconfig n485 จะปรากฏว่าพบดีไวซ์ที่ได้ทำการกำหนดไว้ในตอนแรก และต่อมาให้ทำการติดตั้งดีไวซ์ n485 เข้ากับระบบของเน็ตเวิร์ก โดยใช้คำสั่ง ifconfig เช่นเดิม ดังนี้ ifcofnig n485 <IP> netmask <NETMASK> up จากนั้นให้ทำการทดลองใช้คำสั่ง ping ที่ <IP> นี้ จะพบว่ามีการตอบรับกลับมา

จากขั้นตอนที่กล่าวมา ก็จะสามารถที่มีอุปกรณ์ที่ได้มีการติดต่อกับระบบเน็ตเวิร์กบนระบบของลีนุกซ์ได้แล้ว แต่ในดีไวซ์ไดรเวอร์แต่ละตัวก็จะมีการทำงานที่รองรับกับอุปกรณ์นั้นๆ ดังนั้นในส่วนต่อไปจะเป็นการเขียนโปรแกรมประเภทโมดูลที่สามารถติดต่อกับอุปกรณ์ได้ ซึ่งในโครงงานนี้จะมีการใช้อินเทอร์รัปต์ และการเข้าถึงข้อมูลบนแรมที่อยู่ในอุปกรณ์ รวมทั้งการใช้ไอโอ ที่เข้าถึงอุปกรณ์บนระบบปฏิบัติการลีนุกซ์

3.3.3. การใช้งานอินเทอร์รัปต์

ในการใช้งานนั้นต้องมีการสร้างฟังก์ชันที่รองรับการเกดอินเทอร์รัปต์ เมื่อฟังก์ชันพร้อมแล้วในระบบของลีนุกซ์ได้มีฟังก์ชันที่สนับสนุนในการติดต่ออินเทอร์รัปต์กับระบบคือ request_irq() โดยฟังก์ชันนี้จะติดตั้งฟังก์ชันที่ต้องการเข้ากับระบบอินเทอร์รัปต์ โดยมีการใช้ดังนี้ Result = request_irq(IRQ485,net485irq,0,"net485",NULL); จากนั้นให้เรียก enable_irq(IRQ485) เพื่ออนุญาตได้เกิดการอินเทอร์รัปต์ได้

3.3.4. การใช้งานแรมบนอุปกรณ์

ในโครงงานนี้นั้นใช้การติดต่ออุปกรณ์แบบไอเอสเอ ดังนั้นแรมของอุปกรณ์ต่างๆจะมาอยู่ที่ตำแหน่ง A0000H ถึง 100000H แต่อุปกรณ์ที่ใช้ได้ออกแบบให้อยู่ที่ D8000H และต่อจากนั้นไปอีก 2 กิโลไบต์ ในการใช้การแรมตรงส่วนนี้จะต่อในฟังก์ชัน ioremap() ในการแมพแอดเดรสฟิสิกส์คัลให้อยู่ที่เวอร์ชวลเมโมรี (Virtual Memory) จึงจะสามารถใช้งานแอดเดรสตรงส่วนนี้ได้ แต่ในเคอร์เนลรุ่นใหม่ๆนั้นได้มีการ ioremap() อยู่แล้วตั้งแต่การเริ่มต้นทำงานของระบบ จึงสามารถใช้งานได้เลยซึ่งอยู่ที่แอดเดรส D8000H

3.3.5. การใช้งานไอโอ

ไอโอในระบบของลีนุกซ์นั้นสามารถที่จะกำหนดสิทธิในการใช้งานได้ ซึ่งจะใช้ในเอ็กซีคิวต์ไฟล์ทั่วไปนั้นจะต้องมีการกำหนดสิทธิ์ก่อนโดยใช้คำสั่ง ioprem() แต่ถ้าเป็นการใช้งานในโมดูลแล้วไม่จำเป็นต้องใช้ ioperm() ซึ่งสามารถใช้งานได้เลย โดยใช้ฟังก์ชัน inb() หรือ outb()

3.4. การอิมพลีเมนต์เครือข่ายท้องถิ่นด้วย RS-485

เทคโนโลยีที่ใช้ในการสื่อสารคือ RS-485 เป็นมาตรฐานที่ใช้ในการสื่อสารข้อมูลอนุกรมแบบหลายจุดต่อเชื่อม (Multi-point) โดยมีลักษณะดังรูป แล้วอุปกรณ์ที่นำมาใช้ในโครงงานนี้ก็คือไอซีเบอร์ SN75176BP โดยจะเป็นการสื่อสารแบบรับส่งไม่พร้อมกัน (Half duplex) และไม่มีระบบการตรวจสอบการชนกันของข้อมูล ดังนั้นการชนกันของข้อมูลจึงเป็นปัญหาที่สำคัญสำหรับการทำอุปกรณ์ตัวนี้มาใช้งาน ซึ่งในหัวข้อต่อไปจะเป็นการออกแบบระบบที่ป้องกันการชนกันของข้อมูลที่อาจจะทำให้ระบบนั้นเสียหายได้

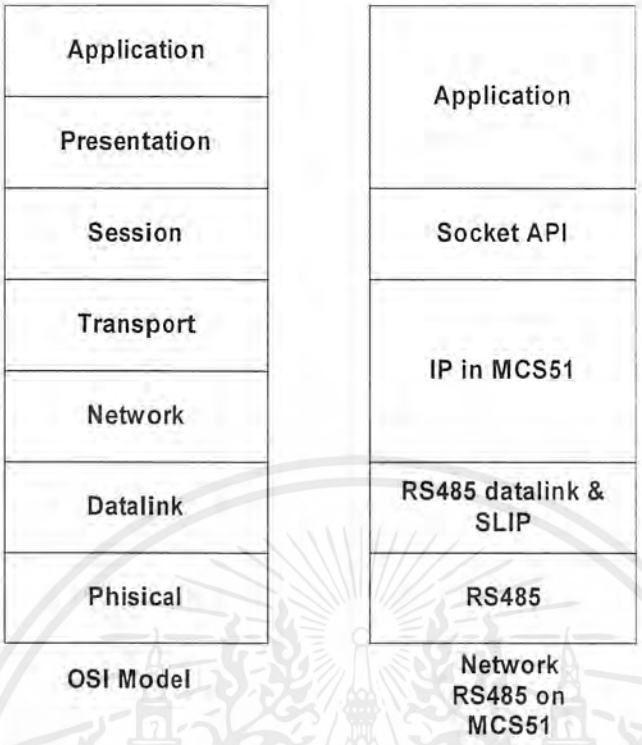
3.4.1. ปัญหาที่เจอในระบบ

3.4.1.1. เนื่องด้วยอุปกรณ์ที่นำมาใช้นั้นไม่สามารถที่จะตรวจสอบการชนกันของข้อมูลหรือการส่งข้อมูลออกมาพร้อมกันจากหลายส่วนจึงทำให้ระบบหรือข้อมูลเกิดการเสียหายได้ รวมทั้งการที่ทำให้ข้อมูลเสียหายและระบบต้องสามารถทำงานต่อไปได้ และสามารถที่จะนำข้อมูลที่สมบูรณ์มาใช้งานต่อไปได้

3.4.1.2. อุปกรณ์ที่นำมาเป็นส่วนประมวลผลหลักนั้นเป็นอุปกรณ์ที่มีความสามารถต่ำซึ่งก็คือตะกูล MCS-51 ในโครงงานนี้ใช้ก็คือเบอร์ 89C52 ซึ่งมีการอ้างหน่วยความจำได้น้อยเพียง 64 กิโลไบต์

3.4.1.3. การออกแบบระบบนี้ได้มีการอ้างอิงกับโมเดลอ้างอิง (OSI model) ซึ่งมีการแบ่งการทำงานออกเป็นหลายเลเยอร์ด้วยกันจึงทำให้ต้องมีส่วนที่ต้องมีการทำงานไปพร้อมกัน (Concurrently) จึงทำให้ต้องมีการออกแบบการทำงานของ MCS-51 ให้เหมาะสม

3.4.1.4. ความเร็วในการสื่อสารข้อมูลนั้น (baud rate : bps) ค่อนข้างต่ำในโครงงานใช้ที่ 19.2 กิโลบิต/วินาที



รูปที่ 29 การเปรียบเทียบระหว่างโอเอสไอโมดูลกับเน็ตเวิร์ก RS-485

3.4.2. การออกแบบ

การออกแบบแบ่งออกเป็นส่วนหลักๆได้ 3 ส่วนคือ

3.4.2.1. ดาต้าลิงก์ (Datalink) : ใน RS-485 ซื่อเป็นส่วนสำคัญในการสื่อสารข้อมูลที่ต้องรับประกันความถูกต้องของข้อมูลที่ได้รับก่อนที่จะมีการส่งต่อไปให้กับเลเยอร์ที่สูงกว่าอีกต่อไป

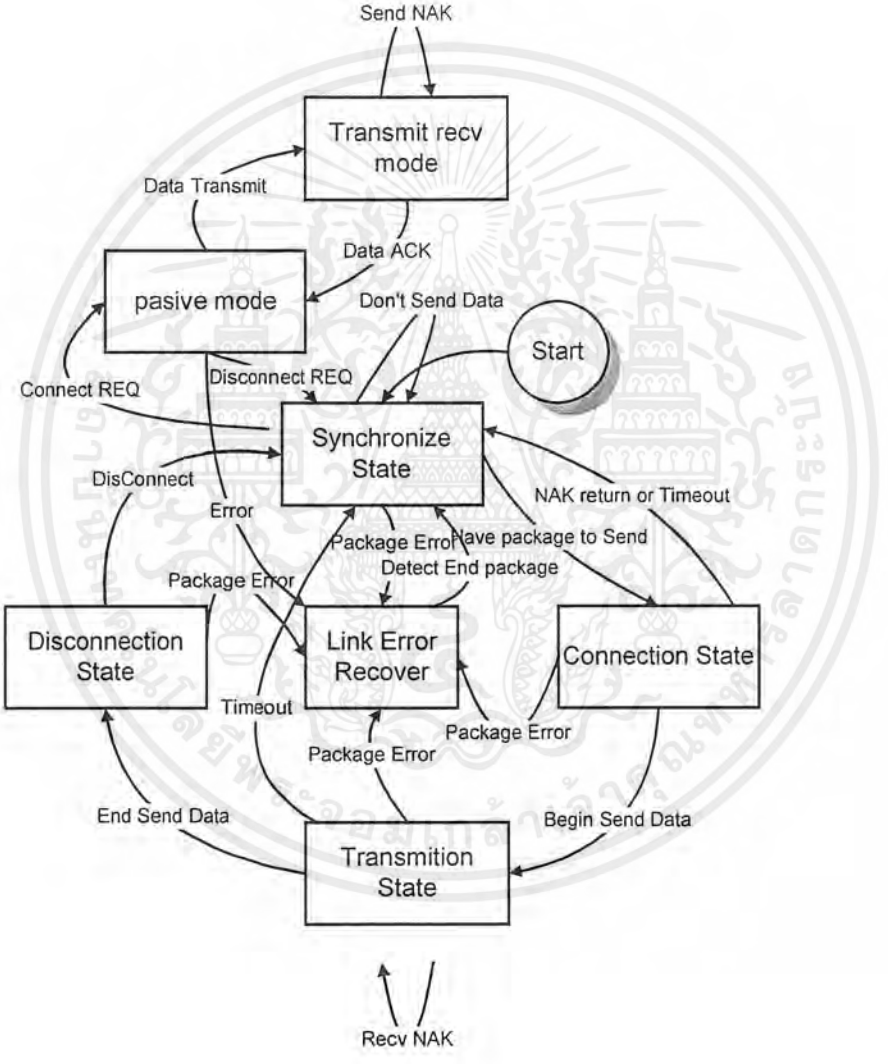
3.4.2.2. อินเทอร์เน็ตโพรโตคอล (Internet Protocol: IP) : เป็นส่วนที่จัดการเกี่ยวกับการสื่อสารในระดับอินเทอร์เน็ต

3.4.2.3. ซ็อกเก็ตฟังก์ชันที่รองรับการติดต่อจากแอพพลิเคชัน (Socket API) : เป็นฟังก์ที่ให้บริการกับโปแกรมอื่นในการใช้งานอินเทอร์เน็ตโพรโตคอล

3.4.3. ดาต้าลิงก์

ในการทำงานของดาต้าลิงก์จะมีการทำงานเป็นลักษณะของพูลลิ่ง (Pooling) โดยมีเซิร์ฟเวอร์ที่ทำหน้าที่การควบคุมการรับส่งข้อมูลของแต่ละอุปกรณ์ที่ต่ออยู่กับเครือข่าย

3.4.3.1. **สเตทไดอะแกรม** : ในส่วนสถานะของดาต้าลิงก์จะมีสเตทของการทำแบบสัมพันธ์กัน (Synchronization State) โดยมีตัวที่จัดการควบคุม ซึ่งสเตทนี้เป็นสเตทแรกของระบบเมื่อเริ่มมีการทำงาน โดยสเตทของเซิร์ฟเวอร์กับไคลเอ็นต์จะมีการเปลี่ยนแปลงของสเตทที่เหมือนกันแต่จะแตกต่างกันตรงสเตทของการทำงานแบบสัมพันธ์กับ โดยเซิร์ฟเวอร์จะมีการทำงานเป็นลักษณะแอ็กทิฟ และส่วนของไคลเอ็นต์จะมีงานในลักษณะพาสซิฟ ดังในสเตทไดอะแกรม



รูปที่ 30 รูปแสดงสเตทไดอะแกรม

3.4.3.2. **การเปลี่ยนแปลงของแต่ละสถานะ** : เริ่มต้นการทำงานระบบของดาต้าลิงก์จะอยู่ที่สถานะซิงโครไนซ์สเตท (Synchronize State) ทั้งทางเซิร์ฟเวอร์และไคลเอ็นต์ แต่ในสถานะนี้ทั้งสองจะมีการทำงานที่แตกต่างกันโดยเซิร์ฟเวอร์จะส่งเมสเซจ (Message) ไปยังไคลเอ็นต์โดยจะมีไอดี (ID: Identify) เพื่อบอกว่าเมสเซจนั้นเป็นของไคลเอ็นต์ ตัวใด โดยไอดีของไคลเอ็นต์นั้นจะเริ่มตั้งแต่ 2 จนถึง 14 ซึ่งไอดีหมายเลขหนึ่ง จะเป็นไอดีของ เซิร์ฟเวอร์และไอดี 0 จะไม่มีการใช้งาน ส่วนไอดี 15 เป็นไอดีที่ใช้ในการบรอดคาสต์ (Broadcast) เซิร์ฟเวอร์จะส่งเมสเซจเพื่อถาม กับไคลเอ็นต์ว่าต้องการส่งข้อมูลหรือไม่ถ้าไคลเอ็นต์ต้องการส่ง จะส่งแอ็กโนวเลจ (Acknowlage) ให้กับเซิร์ฟเวอร์ถ้าไม่ต้องการส่งไคลเอ็นต์จะตอบด้วยเนกาทิฟแอ็กโนวเลจ (Nagative Acknowlage) และถ้าไม่มีไคลเอ็นต์ที่ไม่มีไอดีนั้นเซิร์ฟเวอร์จะรอจนกว่าจะหมดเวลา (Timeout) การที่จะรอให้มีการไทม์เอาต์ของไคลเอ็นต์จะทำให้ระบบมีประสิทธิภาพต่ำลง จึงได้มีการส่งเมสเซจทุกไอดีในทุกๆ 1 นาที นอกจากนั้นแล้วจะส่งเฉพาะไอดีที่เจอเท่านั้น

เมื่อไคลเอ็นต์ต้องการที่จะส่งข้อมูลไคลเอ็นต์จะรอจนกว่าเซิร์ฟเวอร์ส่งเมสเซจมาที่ไคลเอ็นต์ตัวนั้น ไคลเอ็นต์ก็จะส่งเมสเซจ ACK ให้กับเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์จะส่งเมสเซจอนุญาตในไคลเอ็นต์นั้นสามารถทำการส่งข้อมูลได้ เมื่อไคลเอ็นต์ได้รับการอนุญาตแล้วก็เปลี่ยนเป็นสถานะเป็นคอนเน็กชันสเตท ไคลเอ็นต์นั้นจะร้องขอการต่อเชื่อมไปยังอุปกรณ์ปลายทางที่ต้องการซึ่งรวมทั้งเซิร์ฟเวอร์ด้วย ในอุปกรณ์ปลายทางนั้นจะมีการตรวจสอบสถานะของระบบตัวเองว่าพร้อมที่จะรับข้อมูลหรือไม่ ถ้าไม่พร้อมไคลเอ็นต์ที่ร้องขอจะส่งเมสเซจสิ้นสุดการติดต่อให้กับเซิร์ฟเวอร์แล้วเปลี่ยนสถานะเป็นซิงโครไนซ์สเตทเหมือนเดิม และถ้าอุปกรณ์ที่ถูกร้องขอการต่อเชื่อมแล้ว และระบบของตัวเองพร้อมที่จะรับข้อมูลด้วยแล้วก็จะส่งเมสเซจแอกโนวเลจกลับไปยังไคลเอ็นต์ ไคลเอ็นต์นั้นก็จะเปลี่ยนสถานะไปยังทรานส์เฟอร์สเตท และอุปกรณ์ที่ถูกร้องขอจะเปลี่ยนตัวเองให้อยู่ที่พาสซิฟสเตท จากนั้นไคลเอ็นต์ก็จะทำการส่งข้อมูลไป

เมื่ออุปกรณ์ที่ถูกร้องขอรับข้อมูลได้แล้วจะส่งเมสเซจแอกโนวเลจกลับมา และถ้าข้อมูลนั้นตรวจสอบได้ว่าข้อมูลเสียหายจะส่งเมสเซจให้มีการส่งข้อมูลใหม่อีกครั้ง ในขณะที่มีการรับข้อมูลได้เรียบร้อย ไคลเอ็นต์จะส่งเมสเซจดิสคอนเน็กเพื่อบอกว่าสิ้นสุดการส่งข้อมูลให้กับอุปกรณ์นั้น ไคลเอ็นต์จะเปลี่ยนสเตทให้อยู่ในดิสคอนเน็กสเตท และส่งเมสเซจให้กับเซิร์ฟเวอร์ว่าไม่ต้องการส่งข้อมูลแล้ว เพื่อเป็นการคืนระบบให้กับเซิร์ฟเวอร์ควบคุมต่อไป

ในการรับส่งข้อมูลนั้นเป็นไปได้อาจจะมีการเกิดความเสียหายกับข้อมูลได้ ดังนั้นจึงได้มีการตรวจสอบการเสียหายของข้อมูลซึ่งจะใช้วิธีการของเช็กซัม (Check Sum) รวมกับการใช้โพรโตคอลเอสแอลไอพี (SLIP: Serial Line Internet Protocol) ช่วยในการบอกการสิ้นสุดของข้อมูล ซึ่งจะกล่าวในหัวข้อต่อไป

3.4.3.3. **โพรโตคอลที่ใช้ในดาต้าลิงก์** : โพรโตคอลที่นำมาใช้ในดาต้าลิงก์นั้นจะลักษณะคล้ายกับแพ็คเก็จของอีเธอร์เน็ต (Ethernet) โดยจะมีดังนี้

Type	Source ID	Dist ID	Lenght	Data	Check Sum

รูปที่ 31 แพ็กเก็จของอีเทอร์เน็ต

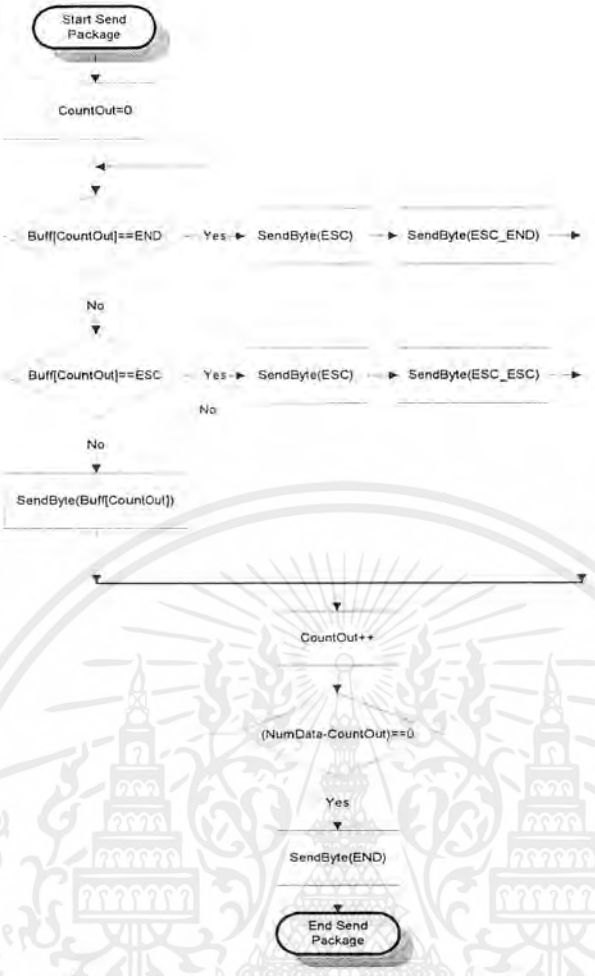
- Type: บอกถึงชนิดของแพ็คเก็จ (8 บิต)
- Source ID: หมายเลขของอุปกรณ์ต้นทาง (4 บิต)
- Dist ID: หมายเลขของอุปกรณ์ปลายทาง (4 บิต)
- Length: ขนาดของความยาวข้อมูล ในฟิลด์ Data (8 บิต)
- Data: ข้อมูลที่อยู่ในเพ็คเก็จ (Lemght ไบต์)
- Check Sum: ข้อมูลที่ใช้ตรวจสอบความถูกต้องของเพ็คเก็จ (8 บิท)

3.4.3.4. โพรโตคอลนี้สามารถที่จะตรวจสอบการผิดพลาดของข้อมูลได้โดยการตรวจสอบเช็กซัมว่าตรงกันหรือไม่ แต่โพรโตคอลไม่สามารถที่จะช่วยในการกู้ระบบขึ้นมาจากการผิดพลาดได้ เช่นในการรับส่งข้อมูลแต่ละเพ็คเก็จนั้นทุกครั้งจะมีตรวจของข้อมูลในไบต์ที่สามที่เป็นไบต์ที่บอกถึงความยาวของข้อมูลจากนั้นก็จะนับข้อมูลที่เข้ามาจนครบตามจำนวนที่บ่งบอกในไบต์ที่สามโดยรวมเช็คซัมด้วยอีกหนึ่ง ดังนั้นแต่ถ้าในระหว่างการส่งได้เกิดการสูญหายของข้อมูลไป N ไบต์จะทำให้การรับข้อมูลผิดพลาดไป N ไบต์ ซึ่งจะได้ข้อมูลของเพ็คเก็จต่อไป N ไบต์เช่นกันและจะทำให้ไม่สามารถที่จะกู้ระบบคืนมาได้ ในเหตุการณ์นี้สามารถแก้ได้โดยการให้มีช่วงเวลาระหว่างเพ็คเก็จ แต่จะเป็นการเพิ่มความลำบากให้กับอุปกรณ์ที่มีเทคโนโลยีไม่สูง ดังนั้นจึงได้มีการประยุกต์ในโพรโตคอลในระดับสูงที่ชื่อว่าเอสแอลไอพี (SLIP) ในการจุดสิ้นสุดของแต่ละเพ็คเก็จเข้ามาใช้งาน

3.4.4. เอสแอลไอพี (SLIP : Serial Line Internet Protocol)

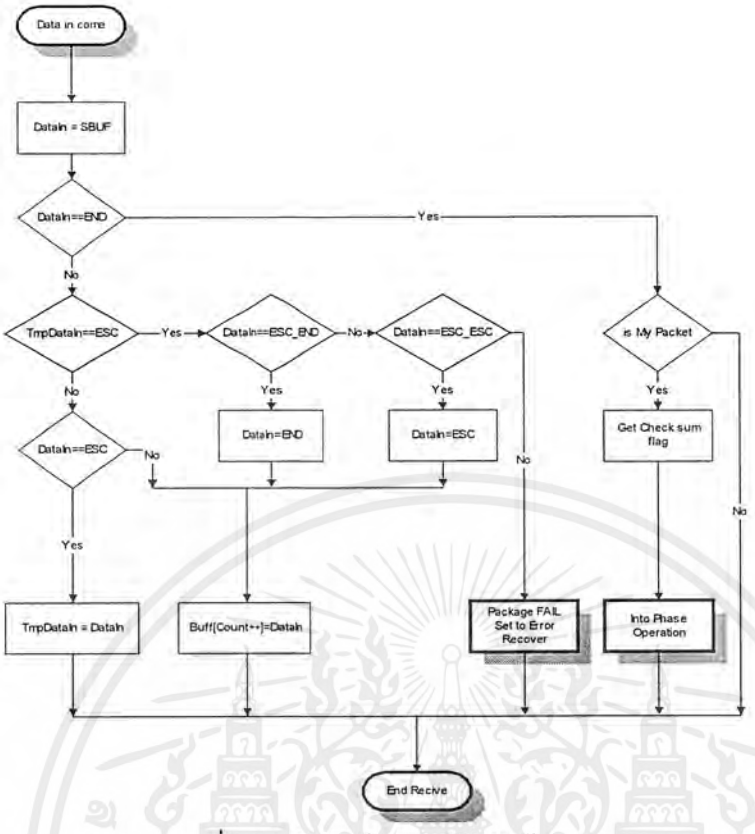
โพรโตคอลเอสแอลไอพีนั้นจะใช้ในการห่อหุ้มอินเทอร์เน็ตโพรโตคอลและส่งไปในการสื่อสารแบบอนุกรมซึ่งโดยทั่วไปจะใช้ส่งแบบจุดต่อจุด (Point to Point) โพรโตคอลเอสแอลไอพีจะมีคาเร็กเตอร์พิเศษอยู่ 4 ตัวด้วยกัน END, ESC, ESC_END และ ESC_ESC โดยจะมีรหัส 192, 219, 220 และ 221 ในเลขฐานสิบตามลำดับ

เพ็คเก็จที่เป็นแบบเอสแอลไอพีจะมีคาเร็กเตอร์ END ที่ท้ายเพ็คเก็จ ดังนั้นเมื่อทำการอ่านเพ็คเก็จแล้วเจอคาเร็กเตอร์ END แสดงว่าได้มีการจบชุดของข้อมูลหรือหนึ่งเพ็คเก็จแล้ว ดังนั้นในการส่งข้อมูลในเพ็คเก็จนั้นๆจะต้องไม่มีคาเร็กเตอร์นี้อยู่ ในการส่งจึงมีการเปลี่ยนจะคาเร็กเตอร์ END เป็น ESC แล้วจะตามด้วยคาเร็กเตอร์ ESC_END เพิ่มเข้าไป และถ้าในข้อมูลมีคาเร็กเตอร์ ESC อยู่ก็จะเป็น ESC และตามด้วยคาเร็กเตอร์ ESC_ESC เพิ่มเข้าไป ดังนั้นจากกลไกการส่งแล้วจะทำให้ไม่มีคาเร็กเตอร์ END อยู่ในเพ็คเก็จเลยนอกจากไบต์สุดท้ายที่แสดงถึงการจบของเพ็คเก็จ แต่ก็จะมีข้อเสียคือ ถ้าในเพ็ตเก็จมีคาเร็กเตอร์ END และ ESC อยู่เป็นจำนวนมากก็จะทำให้มีการส่งข้อมูลเป็นจำนวนมากเช่นกัน การส่งข้อมูลแบบเอสแอลไอพีดังโฟลวชาร์ตดังนี้



รูปที่ 32 รูปแสดงการส่งข้อมูลแบบเอสแอลไอพี

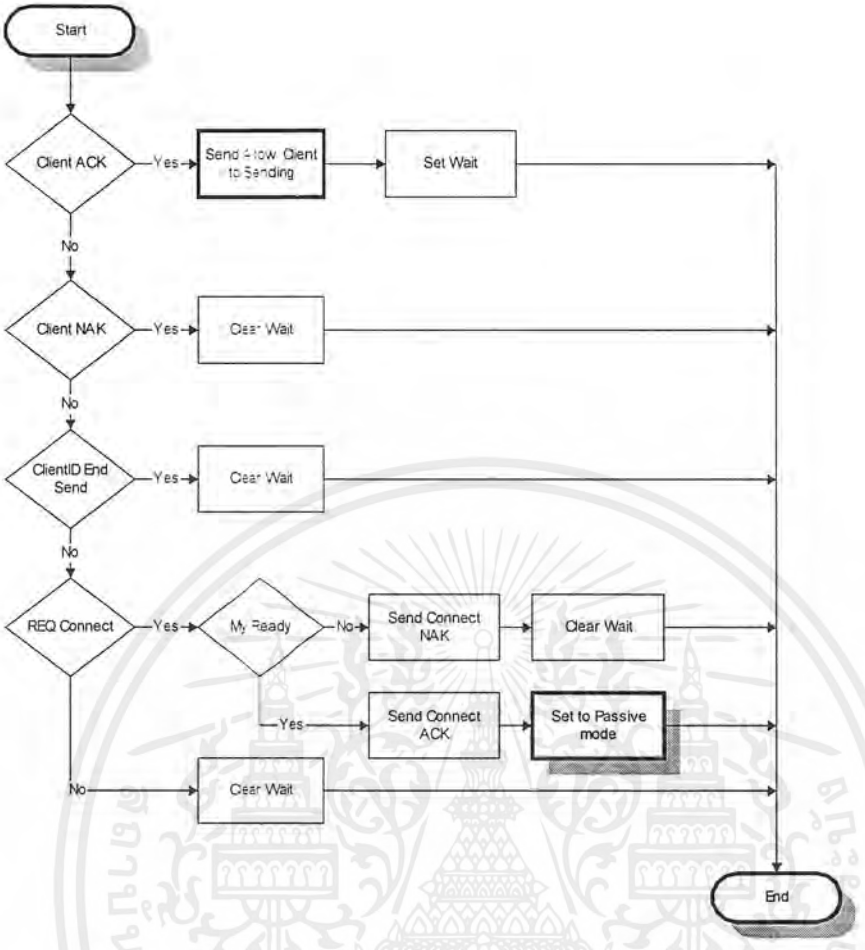
ในส่วนของการรับข้อมูลนั้นจะมีการทำงานที่ตรงกันข้ามกับการส่งดังนี้ ในการส่งข้อมูลนั้นจะส่งในลักษณะไบต์ ดังนั้นก็จะเกิดอินเทอร์รัปต์หนึ่งครั้งก็จะทำการเก็บลงบัฟเฟอร์ แต่ก่อนที่จะมีการเก็บนั้นจะมีการตรวจสอบข้อมูลแต่ละไบต์ว่าเป็น END หรือ ESC หรือไม่ ถ้าเป็น ESC ก็จะรอรับข้อมูลไบต์ถัดไปว่าเป็น ESC_END หรือ ESC_ESC ถ้าเป็น ESC_END ก็จะเปลี่ยนเป็น END และถ้าเป็น ESC_ESC ก็จะเปลี่ยนเป็น ESC แล้วจึงเก็บลงบัฟเฟอร์ และถ้าไม่ใช่ ESC_END หรือ ESC_ESC ก็แสดงว่าเกิดการผิดพลาด จากนั้นถ้ารับข้อมูลมาเป็น END ก็จะแสดงว่าเจอการส่งข้อมูลในหนึ่งเพ็คเก็จดังโฟรวชาร์ต



รูปที่ 33 โฟลวชาร์ตแสดงการรับข้อมูล

3.4.5. การทำงานในแต่ละสถานะ

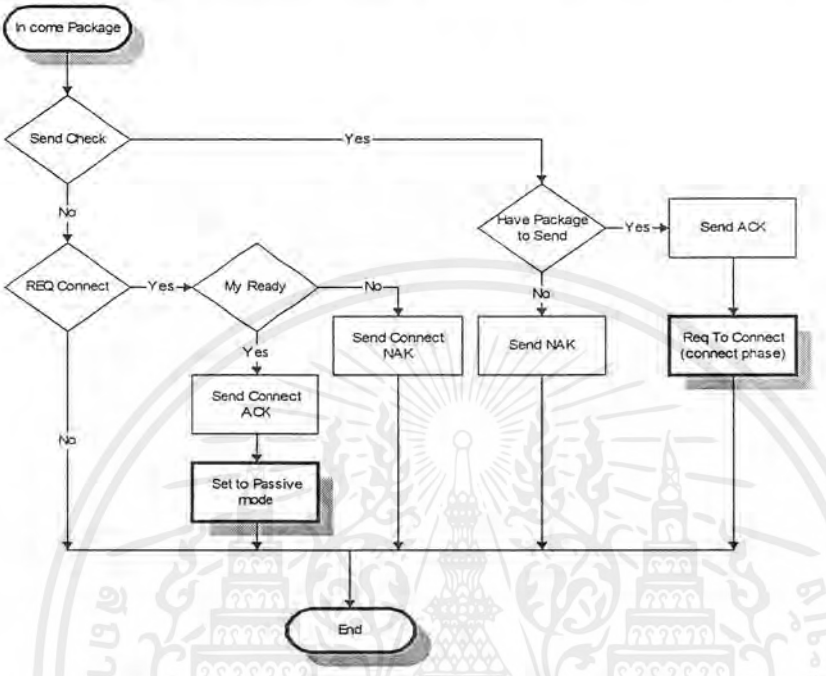
3.4.5.1. เซิร์ฟเวอร์ซิงโครไนซ์สเตท :ในสถานะนี้เป็นการทำงานของเซิร์ฟเวอร์โดยเฉพาะ โดยจะมีการรอรับเพ็คเก็จไทป์หลายไทป์ด้วยกัน และมีการทำงานดังต่อไปนั้นดังรูปที่ 34



รูปที่ 34 รูปโฟลวชาร์ตแสดงการทำงานของเซิร์ฟเวอร์ซิงโครไนซ์

- แอ็กโนวเลจจากไคลเอ็นต์ เซิร์ฟเวอร์จะส่งเมสเซจอนุญาตให้กับไคลเอ็นต์ และกำหนดให้เซิร์ฟเวอร์รอการดิสคอนเน็กจากไคลเอ็นต์
- เนกาทิฟแอ็กโนวเลจ เป็นการบ่งบอกจากไคลเอ็นต์ ว่าไคลเอ็นต์ไม่ต้องการการส่งข้อมูล และจะกำหนดให้เคลียร์การรอ
- เอนด์ออฟเซ็นต์ ไคลเอ็นต์จะบอกกับเซิร์ฟเวอร์ว่าสิ้นสุดการส่งข้อมูล และจะกำหนดให้เคลียร์การรอ
- ร้องขอการติดต่อ เมื่อไคลเอ็นต์ต้องการจะติดต่อกับเซิร์ฟเวอร์ ซึ่งจะเป็นลักษณะเดียวกันกับการร้องขอการติดต่อกับอุปกรณ์ตัวอื่น และจะตรวจสอบระบบของตัวเองว่าพร้อมที่จะรับการติดต่อหรือไม่ ถ้าพร้อมก็จะส่งเมสเซจแอ็กโนวเลจกลับไป และเปลี่ยนสถานะตัวเองให้อยู่ในพาสซิฟสเตท และถ้าไม่พร้อมก็จะส่งเมสเซจเนกาทิฟแอ็กโนวเลจ จากนั้นจะเคลียร์การรอ

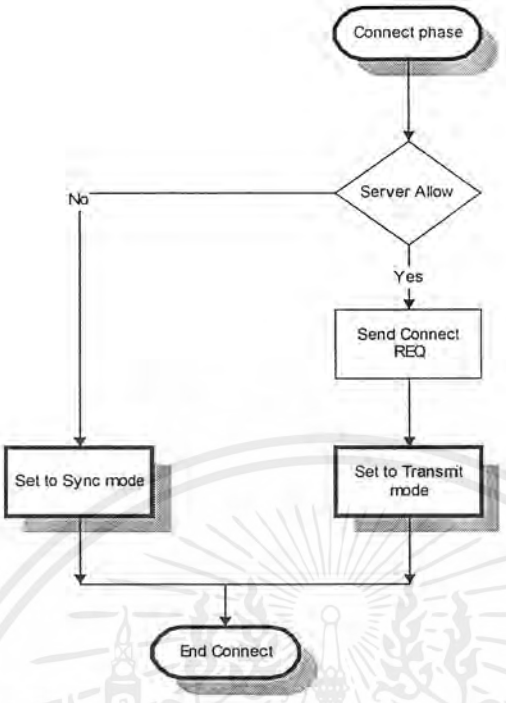
3.4.5.2. **ไคลเอ็นต์ซิงโครไนท์สเตท** : จะมีลักษณะการทำงานคล้ายกับเซิร์ฟเวอร์แต่จะเป็นลักษณะของพาสซิฟ โดยจะรอรับการซิงโครไนซ์จะเซิร์ฟเวอร์ และจะมีการทำงานดังต่อไปนี้



รูปที่ 35 รูปโฟลวชาร์ตแสดงการซิงโครไนซ์ของไคลเอ็นต์

- เมื่อได้รับแพ็คเก็จ ซิงโครไนซ์จากเซิร์ฟเวอร์ ไคลเอ็นต์จะตรวจสอบระบบตัวเองว่ามีข้อมูลที่ต้องการส่งหรือไม่ ถ้าไม่ไคลเอ็นต์จะส่งเนกาทีฟแอ็กโนวเลจกลับไป และถ้ามีก็จะส่งแอ็กโนวเลจให้กลับเซิร์ฟเวอร์ และจะเปลี่ยนสถานะตัวเองให้เป็นคอนเน็กชันสเตท
- เมื่อได้รับการร้องขอการติดต่อ จะมีการตรวจสอบระบบตัวเองว่าพร้อมที่จะรับการติดต่อหรือไม่ ถ้าไม่จะส่งเนกาทีฟแอ็กโนวเลจออกไป และถ้าพร้อมจะส่งแอ็กโนวเลจ และเปลี่ยนสถานะเป็นพาสซิฟสเตท

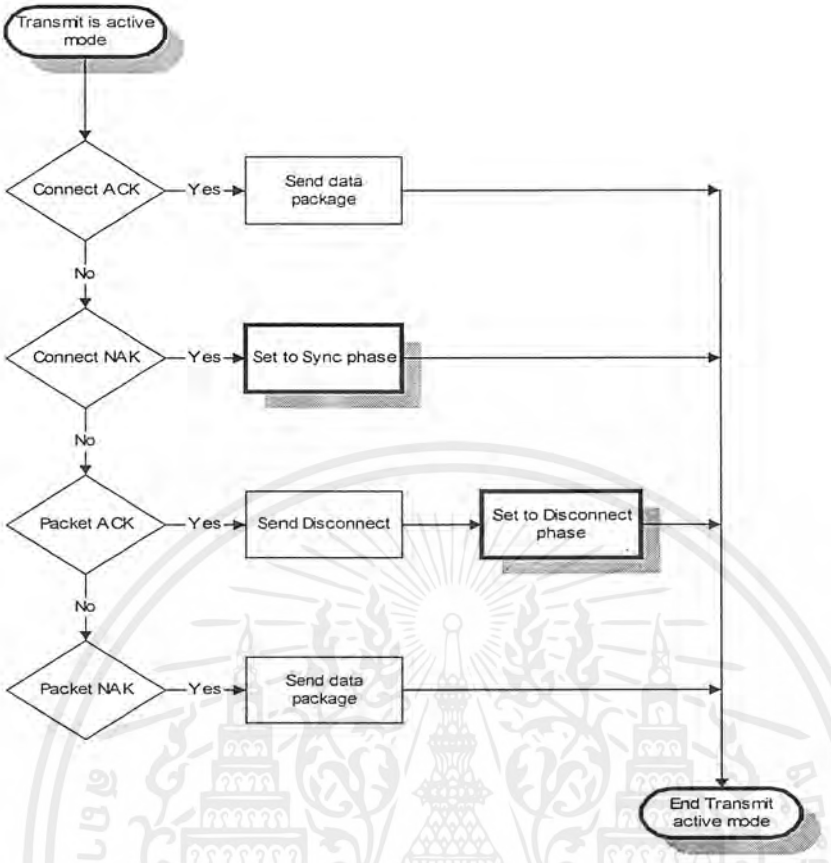
3.4.5.3. **คอนเน็กชันสเตท** : ในสถานะนี้จะมีทั้งในเซิร์ฟเวอร์และในไคลเอ็นต์ โดยสถานะนี้จะรอการอนุญาตจากเซิร์ฟเวอร์ เมื่อได้รับการอนุญาตแล้ว จะส่งการคอนเน็กไปให้กับอุปกรณ์ที่ต้องการ และเปลี่ยนไปอยู่ที่ทรานสมิทชันสเตท และถ้าไม่มีการอนุญาตก็จะเปลี่ยนกลับไปอยู่ที่ซิงโครไนซ์สเตทอย่างเดิม



รูปที่ 36 รูปโฟลวชาร์ตแสดงคอนเน็กชันสเตท

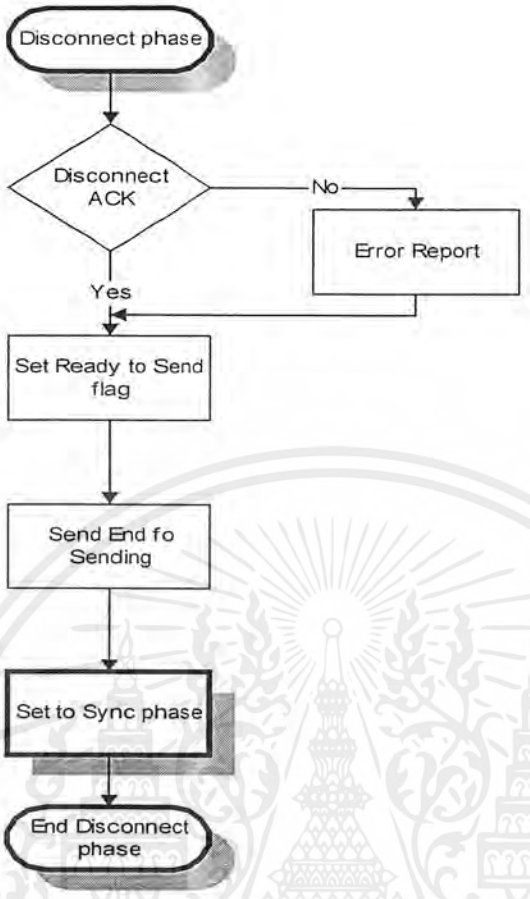
3.4.5.4. ทรานสมิทชันสเตท : สถานะนี้เป็นสถานะที่ใช้ส่งข้อมูลเมื่อได้มีการร้องขอการติดต่อและได้รับการอนุญาตแล้ว

- เมื่อได้รับการแอ็กโนวเลจจากอุปกรณ์ที่ได้ร้องขอไป ก็จะเริ่มส่งข้อมูลได้ทันที
- เมื่อได้รับเนกาทีฟแอ็กโนวเลจ แสดงว่าอุปกรณ์นั้นยังไม่สามารถที่จะรับการติดต่อก็จะเปลี่ยนกลับไปที่ชิงโครไนซ์สเตท
- เมื่อข้อมูลได้มีการส่งไปเรียบร้อยแล้วมีการส่งเมสเซจแพ็คเก็จแอ็กโนวเลจกลับมาแสดงว่าการส่งข้อมูลนั้นเรียบร้อยแล้วก็จะหนดสถานะให้อยู่ที่ดิสคอนเน็กสเตท
- ถ้าได้แพ็คเก็จเนกาทีฟแอ็กโนวเลจกลับมาแสดงว่าไม่สามารถส่งข้อมูลได้สมบูรณ์จะมีการส่งข้อมูลใหม่อีกครั้ง



รูปที่ 37 รูปแสดงการทำงานของทรานสมิทชันสเตท

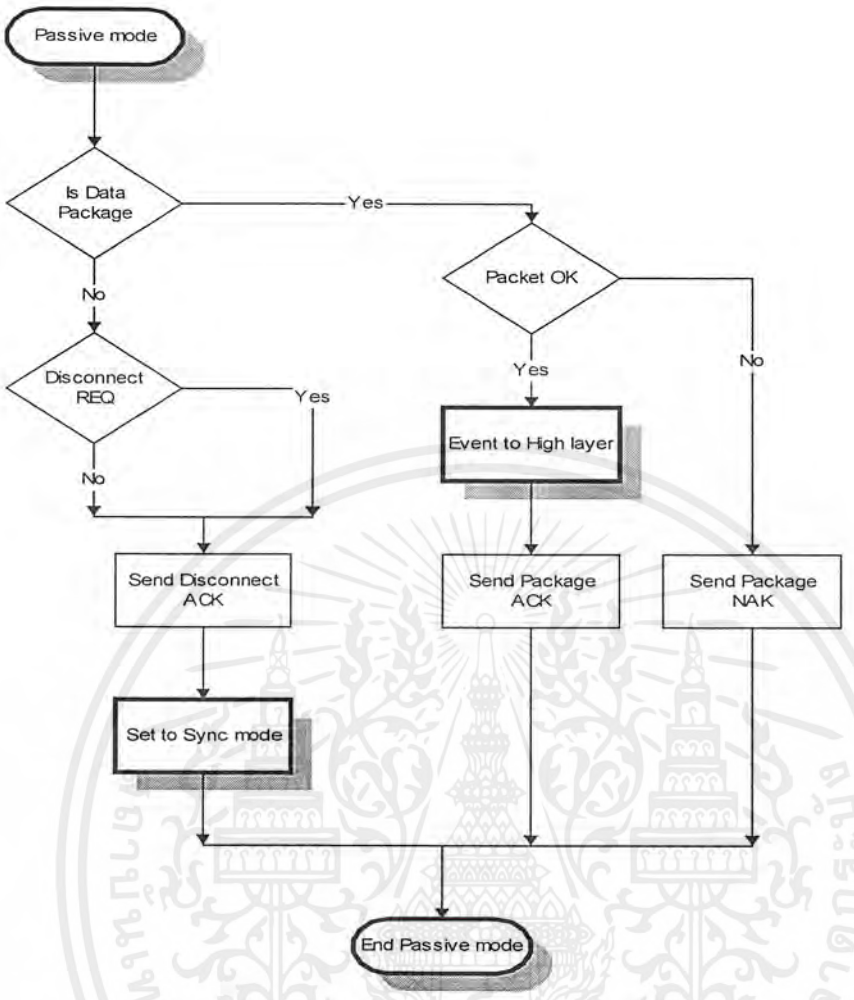
3.4.5.5. **ดิสคอนเน็กสเตท** : ในสถานะนี้จะรอรับดิสคอนเน็กแอ็กโนวเลจ และจะส่งเมสเซจสิ้นสุดการส่งไปยังเซิร์ฟเวอร์จึงกลับเข้าสู่ซิงโครไนซ์สเตท



รูปที่ 38 รูปโฟลวชาร์ตแสดงการทำงานของดิสคอนเน็กสเตท

3.4.5.6. พาสซิฟสเตท : การเข้าสู่สถานะนี้เนื่องจากได้มีการร้องขอการติดต่อจากอุปกรณ์อื่นโดยมีสถานะการทำงานดังนี้

- ถ้าแพ็คเก็จที่เข้ามาเป็นข้อมูลและได้ตรวจสอบว่าเป็นแพ็คเก็จที่สมบูรณ์ก็จะมีการส่งการทำงานหรือสัญญาณให้กับการทำงานในระดับสูงต่อไป และถ้าข้อมูลนั้นมีการผิดพลาดก็จะส่งเนกาทีฟแอ็กโนวเลจกลับไปเพื่อเป็นการบอกให้ส่งข้อมูลเดิมอีกครั้ง
- ถ้าเป็นแพ็คเก็จดิสคอนเน็ก ก็จะส่งดิสคอนเน็กแอ็กโนวเลจกลับไปแล้วกลับเข้าสู่ซิงโครไนซ์สเตท



รูปที่ 39 รูปโฟลวชาร์ต แสดงการทำงานของพาสซิฟสเตท

บทที่ 4

ผลจากการวิจัย

4.1. การทดสอบบอร์ดของ MCS-51

4.1.1. ทดสอบ สเตติกแรม จาก พีซี

เพื่อดูว่า พีซี สามารถอ่านเขียนข้อมูลได้ถูกต้องหรือไม่ โดยในตอนแรกจะทดสอบบนดอสโดยใช้โปรแกรม Debug ซึ่งเพียงแค่เขียนข้อมูลลงไปแล้วอ่านขึ้นมาดูว่าถูกต้องหรือไม่

4.1.2. ทดสอบ สเตติกแรม จาก MCS-51

โดยเขียนโปรแกรมให้ MCS-51 อ่านเขียน สเตติกแรม แล้วเอาออกแสดงผลทางพอร์ต ซึ่งที่พอร์ตดี (Port D) ก็จะมี LED ต่ออยู่เพื่อแสดงผลและดูว่าผลที่ได้ถูกต้องหรือไม่

4.1.3. ทดสอบการทำงานของ พอร์ต

ที่ใช้ในการรับส่งคำสั่งว่าสามารถทำงานได้ถูกต้องหรือไม่ โดยต้องทดสอบควบคู่ไปกับหน่วยความจำตำแหน่ง 8000H ของ MCS-51 ที่ใช้ในการรับส่ง คำสั่ง ด้วย โดยผลัดกันรับผลัดกันส่งแล้วดูว่าผลลัพธ์ถูกต้องหรือไม่

4.1.4. การทดสอบการทำงานเบื้องต้น

โดยเขียนโปรแกรมให้ทั้งสองฝั่งสามารถติดต่อกันได้ สามารถ ซิงโครไนซ์กันได้

4.1.5. การทำการทดสอบการรับส่ง แพ็คเก็จ จริงๆ

ซึ่งทำบนลีนุกซ์โดยทดลองกับคำสั่งง่ายๆเช่นคำสั่ง ping 192.168.1.2 แล้วดูว่า แพ็คเก็จ ที่พีซี ส่งออกไป MCS-51 ได้รับถูกต้องหรือไม่

4.2. การทดสอบเครือข่ายของ MCS-51

การทดลองแบ่งเป็นสองส่วนหลักคือ

4.2.1. การทำงานบนระบบลีนุกซ์

ในส่วนนี้จะทดลองการ์ดที่ติดต่อระหว่างเน็ตเวิร์กกับระบบลีนุกซ์แบ่งเป็นส่วนๆ ดังนี้

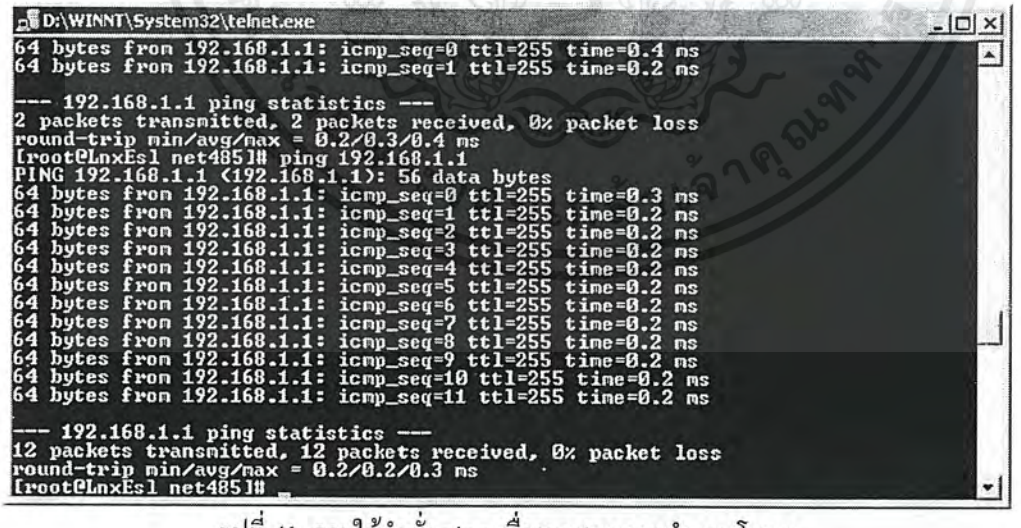
4.2.1.1. การทดลองกับโปรแกรมโมดูลที่ได้พัฒนากับระบบอินเทอร์เน็ต : โดยการตรวจสอบการเพิ่มคำสั่ง printk() เพื่อใช้แสดงผลที่ฟังก์ชัน n485_hard_xmit() เนื่องจากฟังก์ชันนี้จะทำงานก็ต่อเมื่อเลเยอร์บนหรือเน็ตเวิร์กเลเยอร์หรือเลเยอร์ไอพีนั้นต้องการส่งข้อมูลลงสู่ระบบเน็กเวิร์ก RS-485 โดยมีลำดับการปฏิบัติดังนี้

- ทำการติดตั้งโมดูลโดยใช้คำสั่ง insmod net485.o โดย net485.o เป็นไฟล์ที่ได้คอมไพล์มาจากโค้ด

- ตรวจสอบว่าดีไวซ์โมดูลได้มีการติดตั้งเข้าสู่ระบบโดยการใช้คำสั่ง ifconfig n485 จากนั้นผลจะแสดงข้อมูลของดีไวซ์โมดูลออกมา
- จากนั้นจะติดตั้งดีไวซ์โมดูลเข้าสู่ระบบเน็ตเวิร์ก โดยใช้คำสั่ง ifconfig n485 192.168.1.1 netmask 255.255.255.0 up แล้วใช้คำสั่ง ifconfig อีกครั้ง จะเห็นผลดังรูปที่ 40

รูปที่ 40 ผลจากการติดตั้งโมดูล

- ต่อมาให้ทดลองโดยใช้คำสั่ง ping 191.168.1.1 จะได้ผลของการตอบสนองตาม รูปที่ 41



รูปที่ 41 การใช้คำสั่ง ping เพื่อทดสอบการทำงานโมดูล

- จากนั้นให้ใช้คำสั่ง ping 191.168.1.2 เพื่อทำการทดลองส่งข้อมูลให้กับเน็ตเวิร์กวงเดียวกัน และให้ใช้คำสั่ง more ดูที่ไฟล์ /var/log/message จะปรากฏดัง รูปที่ 42

รูปที่ 42 การติดตามผลการทำงานของโมดูล

ให้สังเกตฟังก์ชันที่มีคำว่า watch: socket buffer จนกระทั่ง watch: end นั้นจะมีตัวเลขที่เรียงอยู่ 3-4 บรรทัด นั่นหมายถึงแพ็คเก็จที่ส่งผ่านมาจากเน็ตเวิร์กเลเยอร์ โดยใน 20 ไบต์แรกนั้นจะเป็นเฮดเดอร์ของอินเทอร์เน็ตโพรโตคอลโดยห่อหุ้มแพ็กเก็จแบบไอซีเอมพี (ICMP)

4.2.1.2. การทดลองโปรแกรมโมดูลกับการ์ดติดต่อกับระบบเน็ตเวิร์ก : ในการทดลองจะมีการเพิ่มการทำงานในส่วนของอินเตอร์รัปต์ที่ใช้ติดต่อกับ MCS-51 เพื่อทำการส่งผ่านข้อมูลสู่ระบบเน็ตเวิร์ก RS-485 โดยในเทอร์มินัลต่อเข้ากับเน็กเวิร์ก RS-485 ดังนั้นข้อมูลจะแสดงผลที่เหมือนกันกับที่แสดงดังรูปก่อนหน้า

4.2.1.3. การทดลองการทำงานระหว่างระบบเน็ตเวิร์กกับระบบอินเทอร์เน็ต : โดยการใช้คำสั่ง ping หรือใช้การเขียนโปรแกรมเพื่อเปิดซ็อกเก็ตแบบ UDP แล้วส่งข้อมูลมายังไอพีที่อยู่ในเน็ตเวิร์ก RS-485 ดังตัวอย่าง

```
include <iostream.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
```

```
#include <stdio.h>

int main(int argc, char *argv[])
{
 cout << "this is test udp socket server\n" << endl;
        int ssock,csock;
        int slen,clen;
        int listenerr;
        char inbuf='1';
        struct sockaddr_in saddr;
 struct sockaddr_in caddr;

        if((ssock = socket(AF_INET, SOCK_DGRAM, 0))==-1)
        {
                cout << "socket error" << endl;
                return -1;
        }

        saddr.sin_family = AF_INET;
        saddr.sin_addr.s_addr = inet_addr("192.168.1.2");
        saddr.sin_port = xxxx;
        slen = sizeof(saddr);

        bind(ssock,(struct sockaddr *)&saddr,slen);
        cout << inbuf <<endl;

        read(ssock,&inbuf,1);
        cout << inbuf <<endl;

        close(csock);
 return EXIT_SUCCESS;
}
```

โดยจะมีผลลัพธ์เป็นแพ็กเก็จไอพีที่เป็นชนิด UDP โดยมีข้อมูลที่ส่งติดมาด้วย

4.2.2. การทำงานของอุปกรณ์ที่ต่อเป็นลูกข่าย

4.2.2.1. **การทดลองกับระบบเน็ตเวิร์ก RS-485** : โดยการใช้เทอร์มินัลเพื่อดูและตรวจสอบผลที่ออกมาโดยจะต้องถูกต้องตามลักษณะข้อมูลที่ได้ออกแบบใน[xxxxx]

4.2.2.2. **การทดลองกับระบบอินเทอร์เน็ต** : เมื่อใช้คำสั่ง ping หรือเปิดซ็อกเก็ตไปยังไอพีที่ต้องการที่อยู่ในเน็ตเวิร์ก RS-485 ก็จะมีการตอบสนองกลับ

บทที่ 5

บทวิจารณ์และสรุปผล

5.1. สรุปผลงาน

จากการพัฒนาลีนุกซ์การ์ดเพื่อใช้เป็นเกตเวย์ให้กับอุปกรณ์ที่มีความสามารถต่ำสามารถสรุปได้ดังนี้

5.1.1. ได้มีการสร้างการ์ดที่ติดต่อไอเอสเอบัสโดยสามารถที่จะโหลดระบบปฏิบัติการลีนุกซ์ได้ โดยใช้หน่วยความจำแบบแฟลชเป็นที่เก็บระบบปฏิบัติการ

5.1.2. ได้มีการพัฒนาโปรแกรมโมดูลที่ติดต่อระหว่างระบบเน็ตเวิร์กในลีนุกซ์กับเน็ตเวิร์ก RS-485 โดยมี MCS 51 เป็นตัวจัดการ

5.1.3. สร้างการ์ดที่มีการจัดการและควบคุมการสื่อสารในระบบเน็ตเวิร์ก RS-485

5.1.4. อุปการณ์ที่สามารถสื่อสารผ่านอินเทอร์เน็ตได้โดยใช้อินเทอร์เน็ตโพรโตคอลผ่านระบบเน็ตเวิร์ก RS-485

5.2. ประโยชน์จากโครงการ

5.2.1. ความรู้ในการทำงานของระบบเน็ตเวิร์กในระบบการสร้าง เช่นการทำงานของระบบปฏิบัติการลีนุกซ์กับการจัดการหน่วยความจำกับเน็ตเวิร์ก

5.2.2. สามารถใช้กับอุปกรณ์ที่มีความสามารถต่ำช่วยให้ประหยัดค่าใช้จ่าย

5.2.3. ในปัจจุบันการสื่อสารนั้นมีความสำคัญต่อชีวิตประจำวัน ดังนั้นการที่อุปกรณ์นั้นสามารถที่จะสื่อสารอย่างมีระบบและความถูกต้องโดยใช้หลักการของโอดอสไอโมเดล จำช่วยให้การสื่อสารนั้นมีประสิทธิภาพกับอุปกรณ์ในระดับต่ำได้

5.2.4. เนื่องจากการสื่อสารเป็นปัจจัยที่สำคัญแล้ว ปัจจุบันอินเทอร์เน็ตเป็นรูปแบบการสื่อสารหนึ่งที่แพร่หลายมากจึงทำให้สามารถนำไปประยุกต์ใช้งานกับชีวิตประจำวันได้

5.3. ปัญหาที่พบจากโครงการ

5.3.1. เนื่องจากการใช้ต้นทุนที่ต่ำจึงจำเป็นที่จะต้องใช้อุปกรณ์ที่มีความสามารถต่ำทำให้มีความยากลำบากในการพัฒนา

5.3.2. ความเร็วในการสื่อสารนั้นเป็นสิ่งที่จำเป็นต่อการออกแบบและพัฒนาระบบการสื่อสาร เช่นเมื่อมีการสื่อสารที่มีความเร็วต่ำจะทำให้ระบบการทำงานแบบ TCP นั้นมีปัญหาหรือไม่สามารถที่จะใช้งานได้เลย เนื่องจากระบบของ TCP จะเกิดการไทม์เอาต์ทำให้หลุดการติดต่อ

5.3.3. การสื่อสารผ่านอินเทอร์เน็ตนั้นจะใช้หน่วยความจำที่มากพอสมควรโดยอย่างน้อยประมาณ 32 – 64 กิโลไบต์ ดังนั้น MCS-51 จึงถือได้ว่ายังเป็นอุปกรณ์ที่มีความสามารถต่ำเกิน

ที่จะนำมาประยุกต์ใช้งานกับอินเทอร์เน็ตหรือถ้าจำเป็นต้องมีการใช้งานก็ต้องมีการออกแบบไว้เป็นอย่างดี

5.3.4. จากหัวข้อที่กล่าวมาในเรื่องของการใช้งานหน่วยความจำนั้นจำเป็นที่ต้องมีการจัดการหน่วยความจำ (Memory Manage or Buffer Manage) เนื่องจากมีการใช้งานหน่ายความจำที่ไม่คงที่ (Dynamic Memory) ซึ่งเป็นอีกเรื่องที่ยากลำบากต่อการใช้งานกับ MCS-51

5.3.5. ในการออกแบบโดยการอ้างอิงตามโอเอสไอโมเดลซึ่งมีลักษณะเลเยอร์ดังนั้นจะมีการทำงานที่มีลักษณะขนานกัน (Concurrency or Multitasking) ซึ่งก็เป็นอีกจุดที่ยากต่อการใช้ MCS-51 เช่นกัน หรือถ้ามีการพัฒนาให้มีการทำงานขนานกันได้ก็อาจทำให้ระบบที่จะนำมาใช้งานกันอินเทอร์เน็ตนั้นด้อยลงตามไปเช่นกัน

5.3.6. ความเร็วในการทำงานของอุปกรณ์ซึ่งมีความสำคัญต่อการทำงานกับระบบโดยรวมโดยเฉพาะเซิร์ฟเวอร์ที่ต้องมีการควมคุมการทำงานหรือความคุมการสื่อสารในระบบให้ดีแล้วยังต้องมีประสิทธิภาพที่มากเป็นพิเศษด้วย

5.4. แนวทางการพัฒนาต่อ

5.4.1. ในโครงการนี้ในเรื่องคอมพิวเตอร์ส่วนบุคคลให้เป็นตัวต่อเชื่อมระบบกับอินเทอร์เน็ตจึงจำเป็นต้องยอมสละเครื่องคอมพิวเตอร์ ดังนั้นถ้าได้มีการพัฒนาให้มีอุปกรณ์ที่สามารถทำหน้าที่นี้โดยเฉพาะ เช่นเป็นจุดต่อเชื่อมกับเน็ตเวิร์กอีเธอร์เน็ต (Ethernet) หรือสามารถที่จะหมุนโมเดมเพื่อต่อการติดต่อเข้ากับไอเอสพี (Internet Service Provider : ISP) ซึ่งอาจจะทำให้ลดต้นทุนและค่าใช้จ่ายได้ในบางส่วน

5.4.2. ทำการออกแบบโครงสร้างโดยรวมใหม่โดยอาจให้ MCS-51 นั้นมีการทำงานในเฉพาะอินเทอร์เน็ตโพรโตคอลและทำการออกแบบการติดต่อกับอุปกรณ์อื่นที่ต้องการใช้งานอินเทอร์เน็ตรวมทั้งซอฟต์แวร์ด้วย ซึ่งเปรียบได้กับการ์ดแลนกับเครื่องคอมพิวเตอร์

5.4.3. พัฒนาซอฟต์แวร์ไลบรารีให้มีการใช้งานที่มาตรฐานเช่น Socket API

5.4.4. ออกแบบและพัฒนาส่วนที่จัดการหน่วยความจำให้เหมาะสมกับระบบไม่ให้ระบบนั้นทำงานหนักเพราะการทำงานนี้มากเกินไปและมีการจัดการหน่วยความจำที่อ่อนตัว

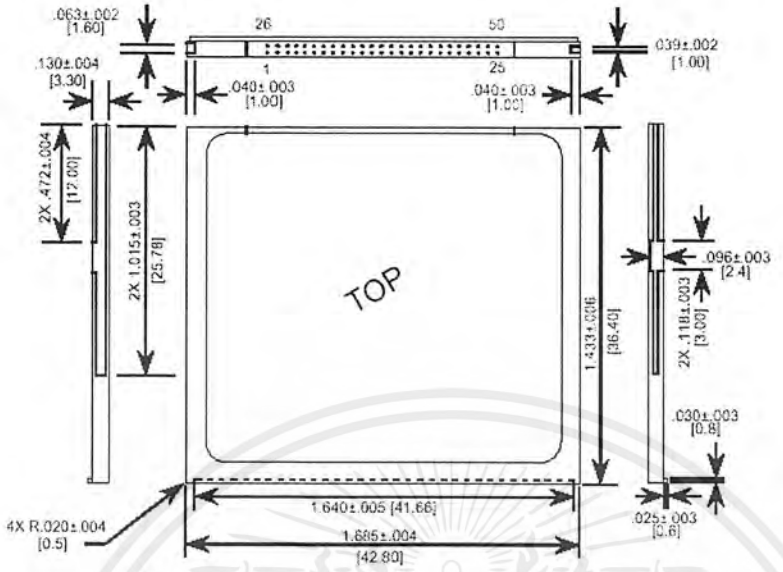
5.5. การประยุกต์ใช้

ในการประยุกต์ในงานนั้นในโครงการนี้อาจจะนำไปประยุกต์ใช้ได้บางส่วน เช่นการสื่อสารผ่านอินเทอร์เน็ตแต่ยังไม่สามารถใช้งานได้ไม่เต็มที่เท่าที่ควร จึงต้องมีการพัฒนาต่อดังเช่นหัวข้อ 5.4 จึงสามารถนำไปใช้งานได้อย่างเต็มที่ ดังนั้นโครงการนี้เป็นเพียงการสร้างและสะสมความรู้หรือประสบการณ์ในการพัฒนาต่อในขั้นต่อๆไป

5.6. สรุปภาพรวมของโครงการ

โดยรวมของโครงการนั้นเป็นการพัฒนาและออกแบบระบบที่มีความสามารถต่ำให้มีการใช้งานระบบการสื่อสารที่เป็นมาตรฐานที่แพร่หลายได้ โดยมีค่าใช้จ่ายที่ต่ำ โดยใช้อุปกรณ์นั้นสื่อสารผ่านอินเทอร์เน็ตโดยมีเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการลีนุกซ์ที่บรรจุอยู่บนหน่วยความจำชนิดแฟลชโดยติดต่อผ่านการ์ดไอเอสเอ แต่อย่างไรก็ตามถือได้ว่าโครงการไม่ประสบความสำเร็จเท่าที่ควรเนื่องจากปัญหาของอุปกรณ์ที่มีความสามารถต่ำ

ภาคผนวก.ข ลักษณะภายนอกของคอมแพคแฟลชและคำสั่งของเอทีเอ

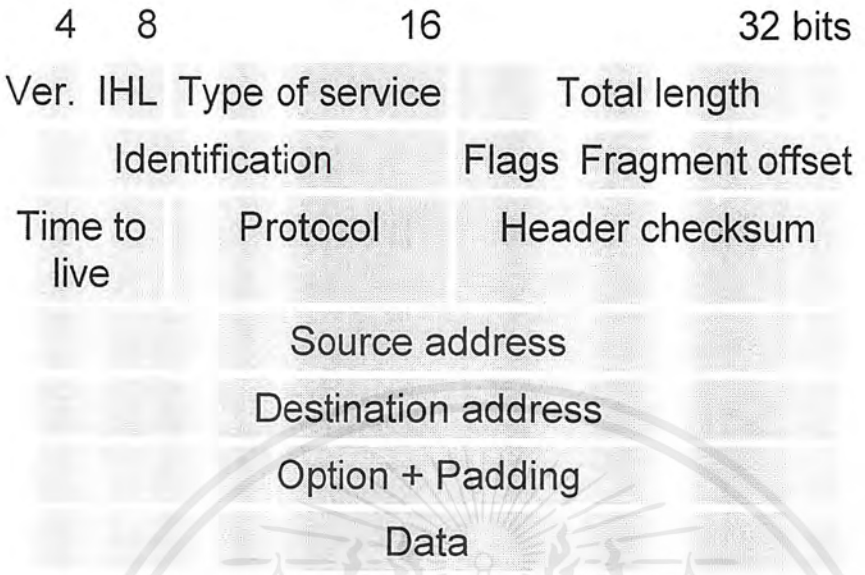


รูปที่ 44 แสดงลักษณะทางภายนอกของคอมแพคแฟลช

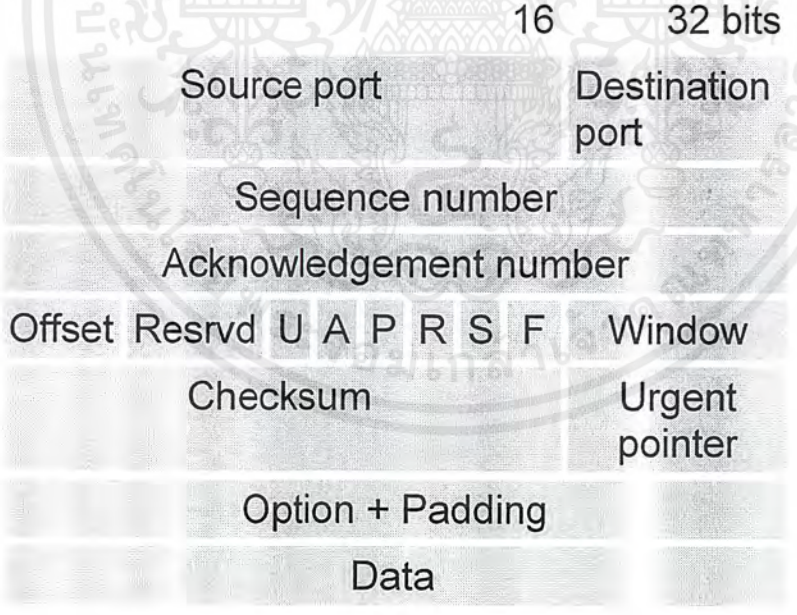
Offset	-OE=0	-WE=0
0	Read Data	Write Data
1	Error	Features
2	Sector Count	Sector Count
3	Sector No. or bits 7-0 of Logical Block Address (LBA)	Sector No. or bits 7-0 of Logical Block Address (LBA)
4	Cylinder Low or LBA 15-8	Cylinder Low or LBA 15-8
5	Cylinder High or LBA 23-16	Cylinder High or LBA 23-16
6	Drive/Head or Drive/LBA 27-24	Drive/Head or Drive/LBA 27-24
7	Status	Command

ตารางที่ 2 เอทีเอรีจิสเตอร์ พื้นฐานที่ใช้ในการ เข้าถึง คอมแพคแฟลช

ภาคผนวก.ค เฮดเดอร์ของโพรโตคอลทีซีพีไอพี



รูปที่ 45 โครงสร้างของไอพีเฮดเดอร์



รูปที่ 46 โครงสร้างของทีซีพีเฮดเดอร์

รูปที่ 47 โครงสร้างของยูดีพีเฮดเดอร์

คำอธิบายศัพท์

80486 : โพรเซสเซอร์ตระกูลหนึ่งของอินเทล เป็นพรเซสเซอร์ขนาด 32 บิต

กิโล (Kilo) : เป็นค่าของปริมาณ มีค่าเท่ากับ สิบยกกำลังสาม

กิกะ (Giga) : เป็นค่าของปริมาณ มีค่าเท่ากับ สิบยกกำลังหก

การ์ดแลน (LAN Card) : คือการ์ดที่ทำหน้าที่ต่อเข้ากับเครือข่ายคอมพิวเตอร์

บิต (Bit) : หน่วยที่เล็กที่สุด มีค่าเป็นศูนย์หรือหนึ่ง

บูต (Boot) : กระบวนการปลุกเครื่อง

ไบต์ (Byte) : กลุ่มของบิตจำนวนแปดบิต

เคอร์เนล (Kernel) : เป็นแก่นของระบบปฏิบัติการที่ทำหน้าที่ในการจัดการทรัพยากรต่างๆของระบบ

โค้ด (Code) : คือคำสั่งที่ใช้ควบคุมการทำงานของโพรเซสเซอร์หรือ รหัส

ซอฟต์แวร์ (Software) : เป็นชุดของโค้ดที่ใช้ทำให้เกิดผลลัพธ์บางอย่าง

เซิร์ฟเวอร์ (Server) : เป็นคอมพิวเตอร์ที่มีหน้าที่ให้บริการต่างๆกับลูกข่าย

โซลิดเสต็ดมีเดีย (Solid-State Media) : คือสื่อที่ใช้ในการเก็บข้อมูลที่ทำมาจากสารกึ่งตัวนำ

เฟิร์มแวร์ (Firmware) : คือคำสั่งที่ฝังไว้ในอุปกรณ์เพื่อควบคุมการทำงานของอุกรณ์นั้น

เพนเทียม (Pentium) : เป็นโพรเซสเซอร์ตระกูลถัดจาก 80486 ของอินเทลที่ใช้สถาปัตยกรรม Super Scalar

โพรเซสเซอร์ (Processor) : คืออุปกรณ์ที่ทำหน้าที่ประมวลผลในคอมพิวเตอร์

โพรโตคอล (Protocol) : คือข้อตกลงในการแลกเปลี่ยนข้อมูล

โปรเทกต์โหมด (Protect Mode) : คือโหมดการทำงานของโพรเซสเซอร์โหมดหนึ่งที่สามารถอ้างหน่วยความจำได้เกินหนึ่งเมกะไบต์

ไมโครคอนโทรลเลอร์ (Micro-Controller) : เป็นอุปกรณ์ที่มีลักษณะคล้ายโพรเซสเซอร์แต่รวมไอโอ, หน่วยความจำม, และส่วนอื่นๆที่จำเป็นสำหรับโพรเซสเซอร์ไว้ในตัวเอง

แมกเนติกมีเดีย (Magnetic Media) :

โมเดม (Modem) : คืออุปกรณ์ที่ทำหน้าที่แปลงสัญญาณอนาล็อกเป็นดิจิตัลและดิจิตัลเป็นอนาล็อกเพื่อให้คอมพิวเตอร์สามารถสื่อสารผ่านสายโทรศัพท์ได้

แรมดิสก์ (RAM Disk) : คือการมใช้หน่วยความจำของระบบจำลองเป็นดิสก์ไดรฟ์

ระบบปฏิบัติการ (Operating System) : คือโปรแกรมที่ทำหน้าที่จัดถารทรัพยากรต่างๆของระบบคอมพิวเตอร์

เรียลโหมด (Real Mode) : คือโหมดการทำงานโหมดหนึ่งของโพรเซสเซอร์ที่ทำอ้างหน่วยความจำได้ไม่เกินหนึ่งเมกะไบต์

ลีนุกซ์ (Linux) : คือระบบปฏิบัติการชนิดหนึ่งที่เป็นแบบผู้ใช้หลายคนเป็นระบบปฏิบัติการแบบเผยแพร่โค้ด

ยูนิกต์ (UNIX) : คือระบบปฏิบัติการชนิดหนึ่งที่เป็นแบบผู้ใช้หลายคนเป็นระบบปฏิบัติการแรกที่สนับสนุนทีซีพีไอพีโพรโตคอล

เวอร์ชวลเมโมรี (Virtual Memory) : คือการจำลองให้ดิสก์ทำงานเป็นหน่วยความจำ

อินเทอร์เน็ต (Internet) : คือเครือข่ายคอมพิวเตอร์สากล

อินเทอร์รัปต์ (Interupt) : คือกระบวนการในการขัดจังหวะโพรเซสเซอร์จากการทำงานปกติให้ไปทำงานบางอย่าง

ไอซี (IC : Integrated Circute) : คือชิปที่รวมวงจรอิเล็กทรอนิกส์ไว้

BIOS (Basic Input Output System) : เป็นชุดของคำสั่งที่ใช้ในการควมคุมการทำงานของอุปกรณ์รอบข้างภายในคอมพิวเตอร์ในระดับต่ำ

H : บ่งบอกว่าเลขนั้นเป็นเลขฐานสิบหก

Half Duplex : คือการสื่อสารแบบสองทางที่สามารถส่งและรับได้เพียงฝ่ายเดียว

MCS-51 : คือไมโครคอนโทรลเลอร์ตระกูล 51 มีขนาดแปดบิต

POST (Power on Self Test) : คือกระบวนการในการทดสอบอุปกรณ์ต่างๆที่ต่ออยู่ในคอมพิวเตอร์ขณะบูตเครื่อง

PnP (Plug and Play) : คืออุปกรณ์ที่สามารถจับจองทรัพยากรได้เองโดยไม่ต้องมีการเซตค่าต่างๆให้คือติดตั้งแล้วใช้งานได้เลย

RS232 : คือมาตรฐานการสื่อสารแบบอนุกรมนิยมใช้บนพีซี เป็นแบบจุดต่อจุด

RS-485 : คือมาตรฐานการสื่อสารแบบอนุกรมที่เป็นแบบหลายจุดโดยใช้สายเชื่อมต่อเพียงคู่เดียว

บรรณานุกรม

[1] Don Anderson, John Swindle, Tom Eisa System Shanley, (1995) "Isa System Architecture", 3rd edition : Addison-Wesley,1995.

[2] Tom Shanley, Don Anderson, Inc. MindShare, (1995) "Plug and Play System Architecture", Addison-Wesley,1995.

[3] Tom Shanley , (1995) : "80486 System Architecture", 3rd edition : Addison-Wesley,1995.

[4] Brian Dipert., (1994) : "Designing With Flash Memory", Coriolis Group (Sd),1994.

[5] Alessandro Rubini, Andy Oram, (1998)"Linux Device Drivers (Nutshell Handbook)", O'Reilly & Associates,1998.

[6] Paul G. Sery,(1998) : "Linux Network Toolkit", Bk&Cd-Rom edition,1998.

[7] Michael Beck , Harald Bohme, Mirko Dziadzka, Ulrich Kunitz, Robert Magnus, Harold Bohme(1997) :"Linux Kernel Internals", 2nd Bk&cdr edition, Addison-Wesley,1997.

[8] Kevin Reichard,(1997):"The Linux Internet Server ",IDG Books Worldwide,1997.

[9] Alessandro rubini.(1999):"Linux device drivers",O'REILLY,1999

[10] Thomas Herbert, (1999) "Embedded System Programming : Introduction to TCP/IP,Part 1", A Miller Freeman Publication, Vol.12, No.13, December 1999

[11] Thomas Herbert, (2000) "Embedded System Programming : Embedding TCP/IP", A Miller Freeman Publication, Vol.13, No.1, January 2000

[12] W.Richad Stevens, (1994), "TCP/IP Illustrated Vol.1", Addison Wesley,1994

เวบไซด์และนิวส์กรุ๊ป

news:comp.arch.embedded

news:comp.realtime

http://www.pcengines.com/embchip.htm

http://linuxembedded.com

http://smalllinux.netpedia.net/links/index.html

http://metalab.unc.edu/mdw

http://www.pein.com/index.html

http://www.edtn.com/embedsys/main1.html

http://www.sandpile.org

http://www.embeddedeurope.com

http://www.emjembedded.com

http://www.m-sys.com

http://www.intel.com

http://www.amd.com