อีเตอร์เน็ตคาเมล่าโมดูล

Ethernet Camera Module

โดย

นาย คณะวัติ เนื่องวงษา

นาย สำรวย ชำนาญกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ภาควิชา
วิศวกรรมโทรคมนาคม

ผ่านการตรวจชิ้นงานแล้ว
(ลงชื่อ).....ผู้ตรวจ

อีเตอร์เน็ตคาเมล่าโมดูล

Ethernet Camera Module

โดย

นาย คณะวัดิ เนื่องวงษา 47015736

นาย สำรวย ชำนาญกิจ 47015752

อาจารย์ที่ปรึกษา

รศ.ดร. ปราโมทย์ วาดเขียน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง อีเธอร์เน็ตคาเมล่าโมดูล

Ethernet Camera Module

ผู้จัดทำ

1. นาย คณะวัติ เนื่องวงษา 47015736
2. นาย สำรวย ชำนาญกิจ 47015752

.....ปราโมทย์..... อาจารย์ที่ปรึกษา

(รศ.ดร. ปราโมทย์ วาดเขียน)

อีเตอร์เน็ตคาเมล่าโมดูล

Ethernet Camera Module

โดย นาย คณะวัติ เนื่องวงษา 47015736

นาย สำรวย ชำนาญกิจ 47015752

อาจารย์ที่ปรึกษา รศ.ดร. ปราโมทย์ วาดเขียน

บทคัดย่อ :

โครงงานนี้เป็นการศึกษาและสร้างอีเตอร์เน็ตคาเมล่าโมดูล เพื่อใช้ส่งข้อมูลภาพจากกล้อง USBผ่านระบบเครือข่ายท้องถิ่น โดยตัวกล้องกับตัวอีเตอร์เน็ตโมดูลจะถูกสร้างขึ้นเป็นบอร์ดและเชื่อมต่อกันผ่านพอร์ต USB โดยจะถูกควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์ โมดูลดังกล่าวจะถูกสร้างขึ้นโดยใช้ไอซีเบอร์ SL811HST เป็นตัวหลักในการทำงาน และรับสัญญาณภาพจากกล้อง USB โดยผู้ใช้สามารถที่จะรับข้อมูลภาพนั้นได้ตลอดเวลา

Abstract :

This project studies and constructs the Ethernet Camera Module to transmit images from the USB camera via LAN system. The camera and the ethernet module are constructed into the board module where the connection between them is accomplished via the USB port. The mentioned module is constructed by using IC SL811HST which primarily operates and receives images from USB camera . The users can monitor the images at all time.

กิตติกรรมประกาศ

คณะผู้จัดทำขอกราบขอบพระคุณ รศ.ดร. ปราโมทย์ วาดเขียน อาจารย์ที่ปรึกษาโครงงาน คณะอาจารย์ทุกท่าน ที่กรุณาให้ความช่วยเหลือพร้อมทั้งแนวความคิดและคำแนะนำต่างๆในการทำโครงงาน ให้สำเร็จลุล่วงไปด้วยดี

ขอขอบพระคุณ เจ้าหน้าที่ ภาควิชาวิศวกรรมโทรคมนาคม ทุกท่านที่กรุณา ให้ความช่วยเหลือ ที่เป็นประโยชน์ต่อการทำโครงงาน

นาย คณะวัติ เนื่องวงษา

นาย สำรวย ชำนาญกิจ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญตาราง

สารบัญตาราง(ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เป็นที่ทราบกันดีแล้วว่าปัจจุบันนี้ ประเทศของเรา ประสพกับปัญหา ทั้งด้านเศรษฐกิจ คนว่างงาน และการติดการพนันของบางกลุ่มคนซึ่งสาเหตุที่กล่าวมาทำให้เกิดกลุ่มมิจฉาชีพจึงทำให้อุปกรณ์รักษาความปลอดภัย หรือการเฝ้าระวังทรัพย์สินหรือ อื่น ๆ นั้นมีความสำคัญมากในปัจจุบัน เลยเกิดแนวคิดที่จะสร้างอุปกรณ์ตัวนี้ขึ้นมาชื่อว่า "อีเตอร์เน็ตคาเมล่าโมดูล" มาทำหน้าที่นี้ โดยอุปกรณ์ตัวนี้จะใช้กล้องยูเอสบี (USB: Universal Serial Bus) เชื่อมต่อกับโมดูลและส่งภาพผ่านระบบเครือข่ายท้องถิ่นซึ่งให้ความสะดวกกับการรักษาความปลอดภัย หรือการเฝ้าระวังทรัพย์สิน เพราะผู้ใช้สามารถเฝ้าดูทรัพย์สินได้ทุกที่ทุกเวลาภายในระบบเครือข่ายท้องถิ่นหรือภายในองค์กร โดยสามารถเข้าไปดูได้ผ่านระบบ คอมพิวเตอร์

สำหรับสาเหตุที่ผู้จัดทำได้ทำอุปกรณ์ที่ใช้พอร์ตยูเอสบี เพราะว่าการเชื่อมต่อผ่านพอร์ตยูเอสบี ผู้ใช้งาน สามารถเชื่อมต่ออุปกรณ์ผ่านพอร์ตยูเอสบีได้โดยไม่ยุ่งยาก เพียงแค่เสียบสายเข้าระบบปฏิบัติการก็จะถามหาซอฟต์แวร์ ซึ่งมีขั้นตอนการติดตั้งเพียงเล็กน้อย ก็สามารถใช้งานได้แล้ว แต่ความสะดวกสบายที่ได้รับของผู้ใช้งานปลายทางนั้น แท้ที่จริงแล้วมีเบื้องหลังการทำงานที่ยุ่งยาก ที่มีเทคโนโลยีแฝงอยู่มากมาย รวมทั้งซอฟต์แวร์ที่ซับซ้อนมากพอสมควร แต่เมื่อเทียบกับพอร์ตอนุกรมและพอร์ตขนานแล้วก็ถือว่า คุ้มค่าเนื่องจากยูเอสบี เป็นระบบบัสที่มีสายเพียง 2 เส้นแต่สามารถรองรับอุปกรณ์ที่เข้ามาเชื่อมต่อได้มากถึง 128 อุปกรณ์ ทั้งอุปกรณ์ที่ใช้พลังงานไม่มากนั้นสามารถใช้พลังงานจากบัสได้โดยไม่ต้องอาศัยพลังงานจากภายนอกอีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

- ศึกษาการทำงานของพอร์ตยูเอสบี
- ศึกษาการทำงานของไมโครคอนโทรลเลอร์
- ประยุกต์การใช้งานไมโครคอนโทรลเลอร์ร่วมกับยูเอสบีโฮสต์คอนโทรลเลอร์
- ศึกษาและสร้างการเชื่อมต่อผ่านทางระบบเครือข่ายท้องถิ่นได้
- เพื่อเป็นแนวทางในการนำระบบนี้ไปประยุกต์ใช้งานในรูปแบบต่าง ๆ

1.3 ขอบเขตของโครงงาน

- สร้างยูเอสบีโฮสต์คอนโทรลเลอร์
- เชื่อมต่ออุปกรณ์ ยูเอสบีร่วมกับเครือข่ายท้องถิ่น

1.4 วิธีดำเนินการ

ด้านฮาร์ดแวร์

- ศึกษาและออกแบบวงจรการทำงานของยูเอสบีโมดูล เบอร์ SL811HST
- ศึกษาและออกแบบวงจรการทำงานของไมโครคอนโทรลเลอร์
- ประยุกต์การใช้งานร่วมกันระหว่างไมโครคอนโทรลเลอร์กับยูเอสบีโมดูลเบอร์ SL811HST

ด้านซอฟต์แวร์

- ไดรเวอร์อุปกรณ์ ยูเอสบี (USB device drivers)
- ไดรเวอร์ยูเอสบี (USB drivers)
- ไดรเวอร์โฮสต์คอนโทรลเลอร์ (host controller drivers)

1.5 ประโยชน์ที่ได้รับ

- สามารถนำไมโครคอนโทรลเลอร์มาประยุกต์ใช้งานได้
- สามารถเขียนโปรแกรมติดต่อระหว่างไมโครคอนโทรลเลอร์และยูเอสบีโมดูลเบอร์ SL811HST ได้
- ได้รับความรู้เกี่ยวกับการทำงานของระบบยูเอสบี

บทที่ 2
ทฤษฎีหรือหลักการ

2.1 ระบบเครือข่ายอีเตอร์เน็ต (Ethernet)

หมายถึง มาตรฐาน ในการต่อเชื่อมคอมพิวเตอร์หลายเครื่อง (ตั้งแต่สองเครื่องขึ้นไป) เข้าด้วยกันเป็นระบบเครือข่าย สำหรับปฏิบัติการในแต่ละจุด (เช่นตามบ้าน หรือ สำนักงานต่าง ๆ) หรือ local area network ซึ่งนิยมเรียกกันสั้น ๆ ว่า LAN รวมทั้งระบบการสื่อสารที่ช่วยให้คอมพิวเตอร์เหล่านั้น สามารถใช้ข้อมูลและโปรแกรมต่างๆ ร่วมกันได้ด้วย หากมี คอมพิวเตอร์หลายเครื่อง ในห้องเดียวกัน หรืออาคารเดียวกัน ระบบเครือข่าย อีเตอร์เน็ต จะสามารถช่วยเพิ่ม ประสิทธิภาพ การทำงาน ของคอมพิวเตอร์เหล่านั้นได้ เพราะหลังจากการสร้างระบบเครือข่าย อีเตอร์เน็ต ขึ้นมาแล้วจะสามารถ ถ่ายโอนข้อมูลระหว่าง PC และเครื่อง server ได้รวดเร็วขึ้นมาก อีกทั้งยังสามารถ สั่งพิมพ์งาน ผ่านพริ้นเตอร์ หรือใช้โปรแกรมต่าง ๆ รวมทั้งระบบ ต่อเชื่อมอินเตอร์เน็ต ร่วมกัน ระหว่าง คอมพิวเตอร์ทุกเครื่อง ในเครือข่ายนั้นด้วย จนถึงปัจจุบันระบบเครือข่ายนี้ ก็ยังนับเป็น ระบบยอดนิยม สำหรับธุรกิจน้อยใหญ่ ทำให้เครือข่าย อีเตอร์เน็ต กลายเป็นระบบ มาตรฐานอย่างหนึ่งในการสร้างระบบ เครือข่ายคอมพิวเตอร์ในปัจจุบัน แต่ระบบเครือข่าย อีเตอร์เน็ต ก็เป็นสิ่งที่มากกว่าแค่อุปกรณ์ฮาร์ดแวร์ เพราะมันยังรวมถึง รูปแบบในการสื่อสาร และการถ่ายโอนข้อมูลต่าง ๆ ของคอมพิวเตอร์ ที่เชื่อมโยงกัน ทั้งนี้ คอมพิวเตอร์ ที่ต่อเชื่อม ด้วยระบบ อีเตอร์เน็ต นี้ จะส่งข้อมูล ไปตามสาย ในรูปแบบ ของกลุ่มข้อมูลขนาดเล็ก ที่เรียกว่า packets โดยใน packet นั้น นอกจากมีข้อมูลต่างๆ แล้ว ยังมีข้อมูลเกี่ยวกับ ที่อยู่ของคอมพิวเตอร์ ที่เกี่ยวข้องกับการรับ-ส่งข้อมูลต่างๆ ด้วย

2.1.1 การต่อเชื่อมเข้าเป็นเครือข่าย อีเตอร์เน็ต

เราสามารถ เชื่อมต่อ คอมพิวเตอร์ ภายในห้องต่างๆ เข้าด้วยกัน รวมไปถึง คอมพิวเตอร์ ที่อยู่ต่างชั้นกัน ได้ด้วย สาย Cable Network ซึ่งเป็น สายใยแก้วนำแสง เพื่อเป็นเส้นทาง ในการส่งผ่านข้อมูล โดยอาศัย ฮับ (Hub) หรือ สวิตช์(Switch) เป็นศูนย์กลาง ในการรับ หรือ กระจายข้อมูล ไปสู่เครื่องต่างๆ ในขณะที่ บางหน่วยงาน ที่มีคอมพิวเตอร์ เป็นจำนวนมาก อาจจะต้องอาศัย Server เป็นตัวควบคุม ร่วมกับ ฮับ หรือ สวิตช์ด้วย

2.1.2 ลักษณะพิเศษของระบบเครือข่าย อีเตอร์เน็ต

- เป็นระบบเครือข่ายที่มีความเร็วในการส่งข้อมูลในรูปแบบดิจิตอลที่มีความเร็ว ตั้งแต่ 10 Mbps จนถึง 1,000 Mbps (1 Gbps)
- เป็นเครือข่ายที่มีขนาดเส้นผ่าศูนย์กลาง ตั้งแต่ 205 เมตรจนถึง 4,000 เมตร

ใช้โปรโตคอลการทำงานที่เรียกว่า CSMA/CD (Carrier Sense Multiple Access with Collision Detect) ซึ่งเป็นมาตรฐานของ IEEE802.3 นอกจากนี้ก็ยังมีมาตรฐาน IEEE802.3u สำหรับ 100

Mbps Fast Ethernet และ IEEE2.3 สำหรับ Gigabit Ethernet รวมทั้ง IEEE802.3ab สำหรับ Gigabit Ethernet ที่ใช้สายทองแดง

- หนึ่งเครือข่าย อีเดอร์เน็ต สามารถมีอุปกรณ์เชื่อมต่อ เช่น คอมพิวเตอร์ลูกข่าย อุปกรณ์ Repeater เป็นต้น ได้มากมายถึง 1,024 รายการหรือเรียกว่า Node
- เป็นเครือข่ายได้สามารถใช้สายสัญญาณได้หลายแบบ เช่น สาย Coaxial ทั้งแบบหนาและแบบบาง สาย Twisted Pair ทั้งแบบ Shield และ Unshield รวมทั้งสาย Optical Fiber แบบและขนาดต่างๆ นอกจากนี้ยังสามารถใช้สื่อที่ใช้รับส่งข้อมูลแบบไร้สาย เช่น คลื่นวิทยุที่มีความถี่ Spread Spectrum รวมทั้งไมโครเวฟ (Microwave) ที่ใช้ความถี่ในช่วง 14 GHz และอินฟราเรด (infrared) เป็นต้น
- เป็นระบบเครือข่ายที่มีการเชื่อมต่อในรูปแบบ Bus และ Star Topology
- อุปกรณ์ที่ใช้มีราคาประหยัด
- มีความน่าเชื่อถือสูง โดยเฉพาะหากใช้สื่อที่เป็นสาย Optical Fiber
- มีเครื่องมือในรูปแบบของซอฟต์แวร์ที่ใช้บริหารจัดการเครือข่ายมากมายที่ทำงานภายใต้ SNMP (Simple Network Management Protocol)

2.1.2.1 ส่วนประกอบหลักที่สำคัญของเครือข่าย อีเดอร์เน็ต

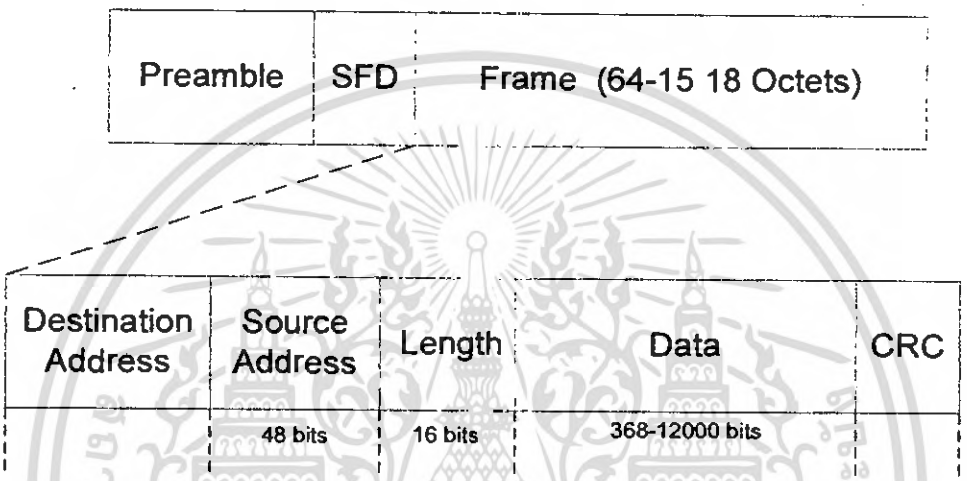
ระบบเครือข่าย อีเดอร์เน็ต มีส่วนประกอบหลักซึ่งเมื่อทำงานด้วยกันแล้วก็จะเป็นเครือข่ายที่มีประสิทธิภาพการทำงานสูงดังนี้

- ตัวเฟรมเป็นชุดรูปแบบของบิตข้อมูลข่าวสารที่ใช้ส่งผ่านมาบนระบบ หากไม่มีเฟรมเราจะไม่สามารถสื่อสารข้อมูลบนเครือข่ายได้โดยเด็ดขาด การรับส่งข้อมูลข่าวสารบนเครือข่าย อีเดอร์เน็ต จะต้องเป็นไปในรูปแบบเฟรมมาตรฐาน 2 แบบ และเป็นแบบใดแบบหนึ่งเท่านั้น (การ์ด LAN เป็นผู้สร้างเฟรมนี้ขึ้นมา)
- ชุดโปรโตคอลที่ใช้ในการควบคุมการแอกเซสเข้าไปที่เครือข่าย (Media Access Control Protocol) ซึ่งประกอบด้วยชุดของกฎกติกาที่อยู่ใน Ethernet Interface (เช่น การ์ด LAN เป็นต้น) ซึ่งเป็นกฎมาตรฐานที่จะยอมให้คอมพิวเตอร์ต่างๆสามารถเข้ามาที่เครือข่าย และแบ่งใช้ทรัพยากรต่างๆ บนเครือข่ายได้อย่างมีประสิทธิภาพ
- อุปกรณ์ที่ใช้รับส่งสัญญาณบนเครือข่าย (Signaling Components) ประกอบด้วยชุดของอุปกรณ์ที่ใช้เชื่อมต่อและส่งสัญญาณเพื่อการรับส่งข้อมูลภายในเครือข่าย
- สื่อที่ใช้ในการรับส่งสัญญาณข้อมูลบนเครือข่าย (Physical Medium) ประกอบด้วยสายสัญญาณรวมทั้งอุปกรณ์ทางฮาร์ดแวร์อื่นๆ ที่จะช่วยในการนำพาข้อมูลข่าวสารต่างๆในรูปแบบดิจิตอลวิ่งไปมาบนเครือข่าย

2.1.2.2 เฟรมบนระบบ อีเตอร์เน็ต

หัวใจสำคัญของระบบ อีเตอร์เน็ต ได้แก่ เฟรมข้อมูลทางข่าวสารและอุปกรณ์ทางฮาร์ดแวร์ที่ใช้เชื่อมต่อสื่อสารบนเครือข่าย ซึ่งได้แก่ การ์ด Ethernet LAN สายสัญญาณและอุปกรณ์เสริมอื่นๆ ที่จะช่วยนำพาข้อมูลในรูปแบบของบิตทางดิจิตอลที่เรียกว่า เฟรม วิ่งไปมาระหว่างคอมพิวเตอร์บนเครือข่าย

เฟรมข้อมูลสำหรับระบบ อีเตอร์เน็ต ประกอบขึ้นด้วยกลุ่มของบิตที่เป็นข้อมูลและข่าวสารสำคัญ แบ่งออกเป็นขนาดสัดส่วนที่แน่นอนที่เรียกว่าช่อง Field



รูปที่ 2.1 ลักษณะโครงสร้างของเฟรมข้อมูล

แสดงให้เห็นรูปแบบของเฟรมข้อมูลที่ใช้บน อีเตอร์เน็ต ได้แก่ Ethernet Frame ตามมาตรฐาน IEEE802.3 ส่วน Ethernet II Frame ซึ่งทั้งสองเฟรมจะมีความแตกต่างกันเล็กน้อย ทำให้เครือข่ายที่ใช้เฟรมแตกต่างกันนี้อาจไม่สามารถเข้ากันได้ หมายความว่าระบบเครือข่าย อีเตอร์เน็ต ของท่านจะต้องเลือกใช้อุปกรณ์เครือข่ายที่คอยสนับสนุนเฟรมอย่างใดอย่างหนึ่งเท่านั้น แต่ก็เป็นเรื่องที่ดีที่ผู้ผลิตอุปกรณ์ทางฮาร์ดแวร์ที่ใช้เชื่อมต่อเครือข่าย อีเตอร์เน็ต มีการใช้มาตรฐาน IEEE802.3 เป็นหลัก อย่างไรก็ดีมีผู้ที่ผลิตอุปกรณ์สนับสนุนเฟรมทั้งสองแบบในตัวเดียวกัน ต่อไปเราจะมาทำความเข้าใจเฟรมมาตรฐานของ IEEE802.3

Preamble	Destination MAC Address (6 Byte)	Source MAC Address (6 Byte)	Type (2 Byte)	Data Field (1500 Byte Max)	Frame Check Sequence (4 Byte)

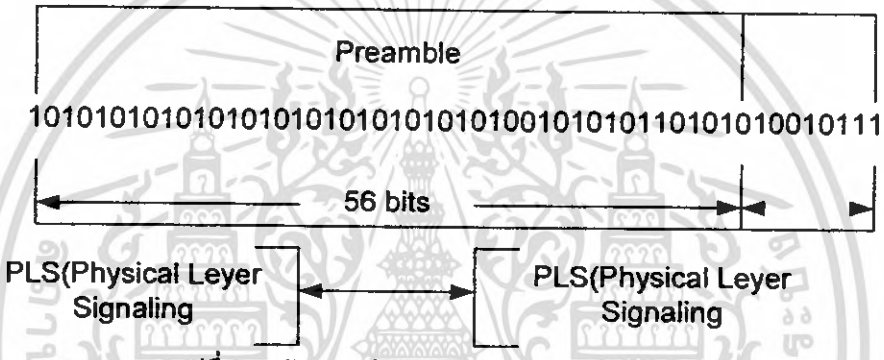
ตารางที่ 2.1 ลักษณะของ Ethernet II Frame

Preamble : ช่อง Preamble ประกอบด้วยบิตข่าวสารที่เป็นเลข 1 และ 0 สลับกัน และสิ้นสุดที่ 11 ซึ่งเป็นบิตที่ 63 และ 64 เป็นบิตข่าวสารที่ยังไม่ใช่ข้อมูลจริงของผู้ส่ง Preamble ประกอบด้วยข่าวสารที่มีขนาด 7

หรือ 8 ไบต์ จุดประสงค์ของข่าวสารนี้ก็เพื่อใช้สร้างจังหวะการรับข้อมูลให้แก่ผู้รับ โดยที่ส่วนนี้จะไปถึงตัวผู้รับก่อน ทำให้เครื่องคอมพิวเตอร์ของผู้รับสามารถปรับจังหวะความเร็วให้เข้ากับผู้ส่งได้ (Synchronize) สำหรับเฟรมแบบ Ethernet II จะมีขนาด 8 ไบต์ และถ้าเป็นมาตรฐาน IEEE802.3 แล้ว ช่องนี้จะถูกแบ่งออกเป็น 2 ส่วน ได้แก่

- Preamble
- Start of Frame Delimeter

จุดประสงค์อีกประการหนึ่งของ Preamble ได้แก่การทำให้แน่ใจว่าระยะเวลาความห่างระหว่างเฟรมที่ 1 กับเฟรมถัดมาจะมีความห่างอยู่ที่ 9.6 ไมโครวินาทีโดยมาตรฐาน ทั้งนี้ก็เพื่อการปฏิบัติการตรวจสอบความผิดพลาดและกู้สถานการณ์เมื่อเกิดปัญหา

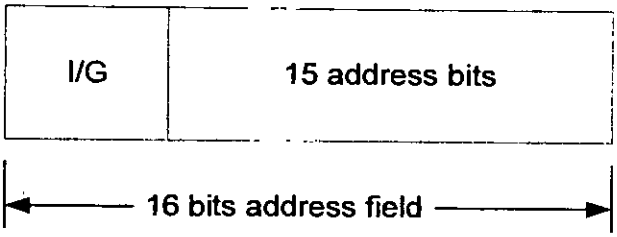


รูปที่ 2.2 ลักษณะส่วนการทำงานภายในของ Preamble

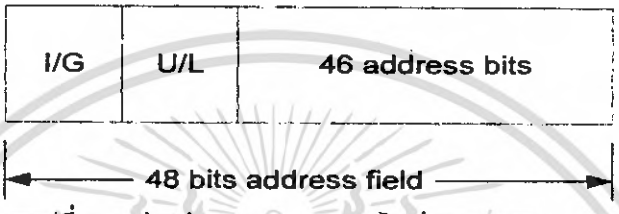
ช่อง Preamble รวมทั้ง Start of Frame Delimeter นี้จะถูกถอดออกไปเป็นอันดับแรกหลังจากคอมพิวเตอร์ของผู้รับได้เฟรมข้อมูลนี้เป็นที่เรียบร้อยแล้ว โดยการ์ด LAN ของผู้รับจะทำการตรวจสอบว่าจำนวนบิตในช่อง Preamble มีครบ 64 บิตหรือไม่ หากไม่ครบก็แสดงว่าเป็นเฟรมที่ไม่สมบูรณ์ โดยอาจเป็นเฟรมที่หลงเหลือมาจากการชนกับเครือข่ายก็ได้ ลักษณะเช่นนี้ การ์ด LAN ของผู้รับจะปฏิเสธที่จะรับและโยนทิ้ง (Frame Drop) ในทันที โดยสามารถเปรียบเทียบกับ Preamble ไม่สมบูรณ์ได้กับรถยนต์หรูๆที่มีกันชนหน้าบุบ ซึ่งเจ้าของที่เป็นผู้รับทำใจไม่ได้เนื่องจากเป็นรถใหม่

ช่อง Destination Address : ในช่อง Destination Address ประกอบด้วยข้อมูลข่าวสารเกี่ยวกับแอดเดรสหรือที่อยู่ของผู้รับปลายทาง ดูเผินๆแล้วเป็นช่องที่เรียบง่ายไม่มีอะไรมาก แต่โดยความจริงแล้วช่องนี้ถูกแบ่งออกเป็นช่องย่อยๆที่เรียกว่า Sub-fields ซึ่งเก็บข้อมูลข่าวสารที่ใช้ดูแลการทำงานของเครือข่าย ทั้งนี้ขึ้นอยู่กับแอดเดรสที่ปรากฏอยู่ในช่องนี้ไม่ว่าช่องแอดเดรสนี้จะมีแอดเดรสที่ถูกเจาะจงเป็นรายบุคคลหรือเป็นกลุ่มของผู้รับก็ตาม

ช่องข่าวสารขนาด 2 ไบต์



ช่องข่าวสารขนาด 6 ไบต์



รูปที่ 2.3 ช่องข่าวสาร ขนาด 2, 6 ไบต์

หมายเหตุ: หากค่าบิตในช่องย่อย I/G มีค่าเป็น "0" ก็หมายถึงแอดเดรสที่ระบุตัวคอมพิวเตอร์ผู้รับอย่างเฉพาะเจาะจง แต่ถ้ามีค่าเป็น "1" ก็หมายถึงคอมพิวเตอร์ที่ระบุแอดเดรสเป็นกลุ่ม ไม่เจาะจงเครื่องใดเครื่องหนึ่ง หากค่าบิตในช่อง U/L มีค่าเป็น "0" ก็หมายความว่าแอดเดรสนี้ถูกกำหนดมาตรฐานโดย IEEE และถ้ามีค่าเป็น "1" ก็จะหมายถึงเป็นแอดเดรสมาตรฐานเฉพาะองค์กรที่ระบุมาตรฐานในซอฟต์แวร์ซึ่งแอดเดรสมาตรฐานที่เราใช้กันอยู่ทุกวันนี้ถูกควบคุมโดย IEEE

จากกรอบแผนผัง เป็นการแสดงรูปภาพในช่องย่อยๆของ Destination Address ซึ่งมีทั้งแบบขนาด 2 ไบต์สำหรับเฟรมแบบ IEEE802.3 เท่านั้น ส่วนแบบ 6 ไบต์สำหรับเฟรมแบบอีเตอร์เน็ต และ IEEE802.3 ผู้ใช้งานสามารถเลือกใช้ช่องย่อยใน Destination Address ทั้งแบบ2 ไบต์และ 6 ไบต์ อย่างใดอย่างหนึ่งและเนื่องจากอุปกรณ์ที่ใช้เป็นแบบ IEEE802.3 ดังนั้นทุกสถานีเครือข่ายทุกเครื่องจะต้องใช้โครงสร้างการอ้างอิงแอดเดรสแบบเดียวกันเสมอ เช่น หากเลือกใช้แบบ 2 ไบต์ก็หมายความว่าคอมพิวเตอร์ทุกเครื่องที่เชื่อมต่อเข้ากับเครือข่ายจะต้องใช้แบบ 2 ไบต์ทั้งหมด ทุกวันนี้เครือข่ายเกือบทั้งหมดใช้มาตรฐานแบบ 6 ไบต์เพื่อการอ้างอิงแอดเดรสถึงกัน แต่ก็ยังรวมช่องขนาด 2 ไบต์นี้เข้าไปเป็นทางเลือกอีกด้วย

ช่อง I/G : มีการจัดตั้งค่าบิตขึ้นในช่องนี้โดยการ์ด LAN ซึ่งหากค่านี้ถูกตั้งค่าไว้ที่ "0"ก็แสดงว่าตัวแอดเดรสที่ระบุอยู่ในช่อง Destination Address นั้นเป็นแอดเดรสที่ระบุตัวคอมพิวเตอร์ผู้รับแบบเฉพาะเจาะจง แต่ถ้าถูกตั้งค่าเป็น "1" ก็แสดงว่าแอดเดรสในช่อง Destination Address นี้เป็นแอดเดรสที่ใช้ติดต่อผู้รับที่เป็นกลุ่มคอมพิวเตอร์ทั้งหลาย เราเรียก Group Address ตัวอย่างของ Group Address ได้แก่ "FFFFFFFFFFFF" ซึ่งถือว่าเป็น Broadcasting Address หรือแอดเดรสที่ไม่เจาะจงผู้รับ โดยผู้รับเป็นกลุ่มหรือทั้งหมดก็สามารถรับข้อมูลข่าวสารนี้ได้

ช่อง Source Address : สำหรับช่อง Source Address นี้มีไว้เพื่อแสดงตัวสถานีเครือข่ายต้นทางที่เป็นต้นทางส่งข้อมูลข่าวสารเข้ามาและเช่นเดียวกับช่อง Destination Address กล่าวคือ ช่อง Source Address สามารถมีช่องย่อยได้ทั้งแบบ 2 ไบต์หรือ 6 ไบต์อย่างใดอย่างหนึ่ง ช่องย่อย Source Address แบบ 2 ไบต์ใช้กับมาตรฐาน IEEE802.3 และต้องใช้แอดเดรสปลายทางขนาด 2 ไบต์เท่านั้น รวมทั้งทุกสถานีลูกข่ายจะต้องใช้ช่องแอดเดรสขนาด 2 ไบต์ส่วนช่องย่อยขนาด 6 ไบต์ สามารถใช้ได้ทั้งมาตรฐาน อีเดอร์เน็ต ทั่วไปและ IEEE802.3 และเมื่อมีการเลือกใช้ช่องย่อย 6 ไบต์ก็จะมีการกำหนดให้ 3 ไบต์แรกเป็นแอดเดรสที่ IEEE กำหนดให้ผู้ผลิตต่างๆซึ่งแอดเดรสนี้จะถูกฝังตัวอยู่ในไมโครชิป บนการ์ด LAN ส่วนที่เหลืออีก 3 ไบต์ก็จะเป็นแอดเดรสที่ผู้ผลิตการ์ด LAN แต่ละแห่งนำไปกำหนดกันเองต่อไป

ผู้ผลิตการ์ด LAN	รหัสผู้ผลิตขนาด 3 ไบต์
Cisco	00-00-0C
Cabletron	00-00-1D
Intel	00-AA-00
3 Com	02-60-8C
Hewlett Packard	08-00-09
Sun	08-00-20
DEC	08-00-2B
Shiva	00-80-D3
Xerox	00-00-AA
IBM	08-00-5A
AT&T	80-00-10

ตารางที่ 2.2 เป็นตัวอย่างของ Source Address ที่แสดงรหัสแอดเดรสของผู้ผลิตดังนี้คือ

ช่องย่อย U/L : ช่องย่อย U/L มีไว้สำหรับช่องขนาด 6 ไบต์เท่านั้น ค่าที่ถูกตั้งไว้ในช่องย่อยนี้เป็นการบ่งบอกให้ทราบว่าแอดเดรสที่ปรากฏอยู่ในช่อง Destination Address นี้เป็นแอดเดรสที่ถูกกำหนดมาตรฐานโดย IEEE หรือองค์กรอย่างเฉพาะเจาะจง

ช่องแสดง Type : ช่องแสดง Type มีขนาด 2 ไบต์ ใช้กับ Ethernet Frame เท่านั้น โดยช่องนี้ใช้เพื่อแสดงว่าโปรโตคอลการทำงานของเฟรมนี้เป็นแบบใด จุดประสงค์คือเพื่อต้องการให้ทราบว่าข้อมูลที่อยู่ในเฟรมนี้ จะทำงานภายใต้โปรโตคอลใด ซึ่งผู้รับจะได้เตรียมการแปลความหมายที่อยู่ในช่องข้อมูล (Data Field) ได้ถูกต้องภายใต้ระบบเครือข่าย อีเธอร์เน็ต เราสามารถใช้โปรโตคอลได้หลายตัวพร้อมกันบนเครือข่าย LAN และบริษัท XEROX ทำหน้าที่เป็นผู้ให้บริการ กำหนดหมายเลขของแอดเดรสที่เป็นลิขสิทธิ์ให้แก่ผู้ผลิตการ์ด LAN ต่างๆรวมทั้งการกำหนดค่าที่ใช้แสดงแทนโปรโตคอลที่ใช้ในช่อง Type แห่งนี้

ช่อง Length : ช่องนี้มีขนาดความยาวเพียง 2 ไบต์ใช้ได้กับเฟรมมาตรฐาน IEEE802.3 เท่านั้นเป็นช่องที่ใช้แสดงขนาดจำนวนของไบต์ที่มีปรากฏอยู่ในช่อง Data ภายใต้มาตรฐาน อีเดอร์เน็ต และ IEEE802.3 ขนาดของเฟรมจะมีขนาดเล็กที่สุดไม่ต่ำกว่า 64 ไบต์ นับตั้งแต่ชุดแรกสุดคือ Preamble จนถึงช่องสุดท้ายได้แก่ FCS และการกำหนดให้มีขนาดเล็กที่สุดไม่น้อยกว่า 64 ไบต์นี้จุดประสงค์ก็เพื่อให้แน่ใจว่าช่วงระยะการส่งข้อมูลมีมากพอที่จะทำให้การ์ด LAN สามารถตรวจพบการเกิด Collision บนสายสัญญาณที่มีขนาดความยาวที่สุดของเครือข่าย หากขนาดเฟรมเล็กกว่า 64 ไบต์ก็อาจเกิดปัญหา Collision ซึ่งจะนำไปสู่ปัญหาบนเครือข่ายได้บนพื้นฐานของเฟรมขนาดเล็กที่สุดคือ 64 ไบต์ และมีการใช้ช่องแอดเดรสขนาด 2 ไบต์หมายความว่าช่องสำหรับ Data จะต้องมีขนาดเล็กสุดไม่ต่ำกว่า 46 ไบต์ (เมื่อหักขนาดและจำนวนช่องต่างๆออกไปหมดแล้ว)เมื่อใดที่ข้อมูลมีขนาดเล็กกว่า 46 ไบต์ช่องของ Data ที่อยู่ในเฟรมจะถูกใส่ค่าเพิ่มให้ได้อย่างน้อยเป็น 46 ไบต์

ช่อง Data (Data Field) : ดังที่ได้กล่าวมาแล้วว่าช่องของ Data อย่างน้อยต้องมีขนาดไม่เล็กกว่า 46 ไบต์ เพื่อให้แน่ใจว่าเฟรมมีขนาดไม่ต่ำกว่า 64 ไบต์ซึ่งหมายความว่าการแหร่ข้อมูลขนาดหนึ่งไม่ว่า 1 หรือ 10 ไบต์ก็ตามต้องมาจาก 46ไบต์นี้แต่ถ้าข้อมูลในช่องนี้เล็กกว่า 46ไบต์ แน่นอนว่าต้องมีการเพิ่มไบต์ลงไปอีกเพื่อให้ได้ขนาด 46 ไบต์อยู่ดี ขนาดของข้อมูลที่อยู่ใน Data จะต้องมีขนาดสูงสุดไม่เกิน 1,500 ไบต์

ช่องตรวจสอบความผิดพลาดของข้อมูลในเฟรม (Frame Check Sequence) : ช่อง Frame Check Sequence นี้ใช้ได้ในเฟรมมาตรฐาน ทั้งเฟรมมาตรฐาน ทั้ง อีเดอร์เน็ต และ IEEE802.3 เป็นช่องที่ประกอบด้วยข้อมูลที่ใช้เป็นกลไกในการตรวจสอบความผิดพลาดของข้อมูลภายในเฟรม หลักการทำงานมีอยู่ว่าก่อนที่เครื่องผู้ส่งจะส่งข้อมูลออกไปที่เครือข่าย การ์ด LAN ของมันจะคำนวณค่าต่างๆในช่องต่างๆซึ่งครอบคลุมตั้งแต่ช่อง Address ต่างๆของtype และช่อง Length รวมทั้งช่อง Data การคำนวณค่าแบบนี้เรียกว่า Cyclic Redundancy Check (CRC) ซึ่งหลังจากที่ได้คำนวณค่าเสร็จสิ้นแล้ว ผลลัพธ์ที่คำนวณได้มีขนาด 4 ไบต์จะถูกนำไปใส่ไว้ในช่อง Frame Check Sequence แห่งนี้

เมื่อเฟรมถูกส่งมาถึงผู้รับแล้ว ตัวการ์ด LAN ของผู้รับจะทำการตรวจสอบค่าที่อยู่ในช่อง Preamble เพื่อดูว่ามีความถูกต้องหรือไม่ เพื่อให้แน่ใจว่าเฟรมที่ทำการตรวจสอบอยู่นี้ไม่ได้เป็นเฟรมที่เหลือรอดจาก Collision หรือการชนกันของสัญญาณในเครือข่าย และหาก Preamble ไม่มีปัญหาเรื่องความถูกต้องก็จะมีการคำนวณค่าที่อยู่ในช่อง Frame Check Sequence ต่อไป หากมีความผิดพลาดกล่าวคือ ค่าที่ได้ไม่ต้องเท่ากับค่าที่ได้จากการที่คำนวณได้จากต้นทาง แสดงว่าเป็นเฟรมที่มีปัญหา ซึ่งก็มักเป็นปัญหาจากสายสัญญาณ รวมทั้งสัญญาณรบกวน ซึ่งในที่สุดการ์ด LAN ก็จะปฏิเสธที่จะรับเฟรมที่เข้ามานี้ในที่สุด

2.1.2.3 ข้อกำหนดเกี่ยวกับขนาดของ Data Frame

ขนาดของ Data Frame มีมาตรฐานดังต่อไปนี้

- ขนาดเล็กที่สุด ต้องไม่น้อยกว่า 64 ไบต์ โดยมี 12 ไบต์สำหรับแอดเดรส 2 ไบต์สำหรับช่อง Length 46 ไบต์สำหรับเก็บข้อมูล และ 4 ไบต์สำหรับตรวจสอบความผิดพลาดข้อมูล หรือFrame Check Sequence
- ขนาดใหญ่ที่สุด ต้องไม่เกิน 1,518 ไบต์ โดยแบ่งออกเป็น 12 ไบต์สำหรับแอดเดรส 2 ไบต์สำหรับ Length 1,500 ไบต์สำหรับข้อมูล และ 4 ไบต์สำหรับช่องตรวจสอบความผิดพลาดข้อมูล
- เฟรมที่มีขนาดเล็กที่สุด 64 ไบต์นี้เทียบกับ 512 บิต (Bit/Byte) โดยที่ระบบ อีเธอร์เน็ต มีค่า Bit Time อยู่ที่ 0.1 ไมโครวินาที (Bit Time เป็นช่วงเวลาที่ใช้ในการส่งข้อมูลขนาด 1 บิต) ดังนั้นการส่งข้อมูลขนาดเล็กที่สุดคือ 64 ไบต์จะต้องใช้เวลาอยู่ที่ 51.2 ไมโครวินาที

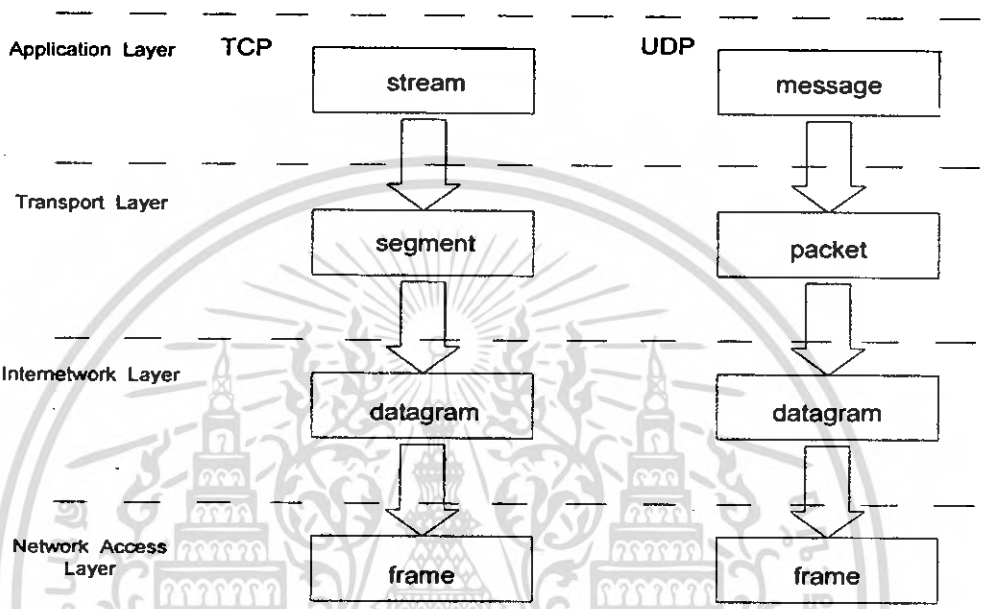
2.1.3 โครงสร้างของโปรโตคอล TCP/IP

โปรโตคอล TCP/IP มีการจัดกลไกการทำงานเป็นชั้นหรือ Layer เรียงต่อกัน โดยในแต่ละ Layer จะมีการทำงานเทียบได้กับ OSI model มาตรฐาน แต่บาง Layer ของโปรโตคอล TCP/IP จะทำงานเทียบกับ OSI หลาย Layer ปนกัน ซึ่งในแต่ละ Layer ของโปรโตคอล TCP/IP จะประกอบด้วย

- Process Layer
- Host – to – Host Layer
- Internetwork Layer
- Network Interface Layer

โดยเมื่อเทียบกับมาตรฐาน OSI model ซึ่งเราจะเห็นว่าบางกลไกของโปรโตคอล TCP/IP เทียบได้กับมาตรฐาน OSI model สองชั้น หรือบางกลไกก็จะทำงานคาบเกี่ยวกันระหว่างบางชั้นของ OSI model ตัวอย่างเช่น กลไกการทำงานของโปรโตคอล TCP/IP ในส่วน Network Interface Layer เมื่อเทียบกับมาตรฐาน OSI model จะเทียบได้กับ Data Link Layer และ Physical Layer 2 ชั้นรวมกัน เป็นต้น ในแต่ละกลไกของโปรโตคอล TCP/IP และโปรโตคอลอื่น ๆ ในชุดของ TCP/IP ร่วมทำงานอยู่ด้วย สำหรับการศึกษาโปรโตคอล TCP/IP นั้นเราจะไม่อ้างอิง OSI Reference Model นี้เพราะจะเข้าใจได้ยาก ดังนั้นเราจึงจะสร้าง Model ขึ้นมาใหม่ ลักษณะการทำงานของ model ในลักษณะนี้คือ ข้อมูลจะถูกส่งลงมาจากชั้นข้างบนลงมายังชั้นข้างล่างสุดซึ่งมีหน้าที่จัดการเกี่ยวกับการส่งข้อมูลผ่านสายสัญญาณไปยังจุดหมายปลายทาง เมื่อข้อมูลไปถึงจุดหมายแล้วก็จะย้อนกลับจากชั้นล่างขึ้นไปชั้นบนสุด ซึ่งเป็นชั้นที่โปรแกรมใช้งานต่างๆ ทำงานอยู่ ขณะที่ข้อมูลถูกส่งผ่านจากชั้นบนลงมายังชั้นล่าง แต่ละชั้นจะทำการเพิ่มข้อมูลควบคุมเข้าไปเพื่อให้การส่งข้อมูลถูกต้อง และเป็นการส่งพารามิเตอร์ที่จำเป็นไปให้กับชั้นของมันในเครื่องปลายทาง ข้อมูลควบคุมเหล่านี้เราเรียกว่า Header แต่ละชั้นจะมี Header ที่มีรูปแบบเป็นของตัวเอง การเพิ่ม Header เข้าไปกับข้อมูล

นั้นเราเรียกว่า Data Encapsulation หน่วยของข้อมูลที่จะทำการส่งนั้นจะมีชื่อเรียกต่างกันเมื่อมันเดินทางผ่านแต่ละ Layer ซึ่งจะเห็นว่าการส่งใน TCP/IP นั้นมีอยู่ 2 โปรโตคอล ย่อยคือ TCP กับ UDP ชื่อของข้อมูลที่ผ่าน โปรโตคอล ทั้งสองนั้นแตกต่างกันในชั้นบนๆ เท่านั้น ชื่อสำคัญๆ ที่เราจะต้องพบและใช้ต่อไปบ่อยๆ ได้แก่ datagram และ frame



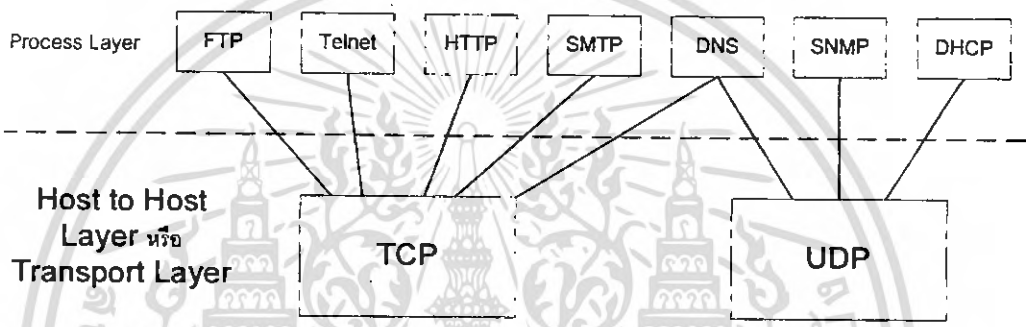
รูปที่ 2.4 Data Structures

2.1.4 Process Layer

การแสดงลำดับชั้นการทำงานของโปรโตคอล TCP/IP เทียบกับมาตรฐาน OSI model นั้น ในชั้นบนสุดเรียกว่า Process Layer ทำงาน 2 หน้าที่เทียบได้กับ Application Layer และ Presentation Layer ในชั้นนี้จะรองรับการทำงานของแอพพลิเคชันต่าง ๆ ที่ทำงานเป็นโปรเซส อยู่ในเครื่องเซิร์ฟเวอร์ที่ให้บริการและเครื่องที่ขอใช้บริการ หรือ client ซึ่งจะติดต่อกันผ่านโปรโตคอลเฉพาะแอพพลิเคชันอีกทีหนึ่ง ตัวอย่างเช่น เมื่อผู้ใช้งานอินเตอร์เน็ตต้องการโอนถ่ายไฟล์หรือ download ข้อมูลจากเครื่องเซิร์ฟเวอร์ที่ให้บริการ โดยอาจจะเรียกใช้โปรแกรม ftp client ทั่วไป เช่น โปรแกรม WS_ftp ติดต่อกับโปรเซส ftp ที่กำลังให้บริการอยู่ที่เครื่องเซิร์ฟเวอร์ จากนั้นตัวโปรเซส ftp ก็จะเรียกใช้โปรโตคอล FTP (File Transfer Protocol) เพื่อทำการโอนถ่ายไฟล์นี้ หรือถ้าผู้ใช้ต้องการเรียกใช้งานคอมพิวเตอร์ที่อยู่ห่างไกลออกไปด้วยการใช้โปรแกรม telnet ที่เครื่องเซิร์ฟเวอร์ให้บริการ ตัวโปรเซส Telnet ที่ทำงานอยู่ก็จะเรียกใช้โปรโตคอล Telnet เพื่อติดต่อกัน หรือในกรณีที่มีการเรียกใช้โปรแกรม web browser เช่น Netscape Navigator เพื่อเรียกดูเว็บไซท์ CNN ที่เครื่องซึ่งให้บริการเว็บของ CNN ก็จะมีโปรเซส HTTP (HyperText Transfer Protocol) ทำงานอยู่และจะติดต่อกับผู้ใช้ผ่านโปรโตคอล HTTP เป็นต้น

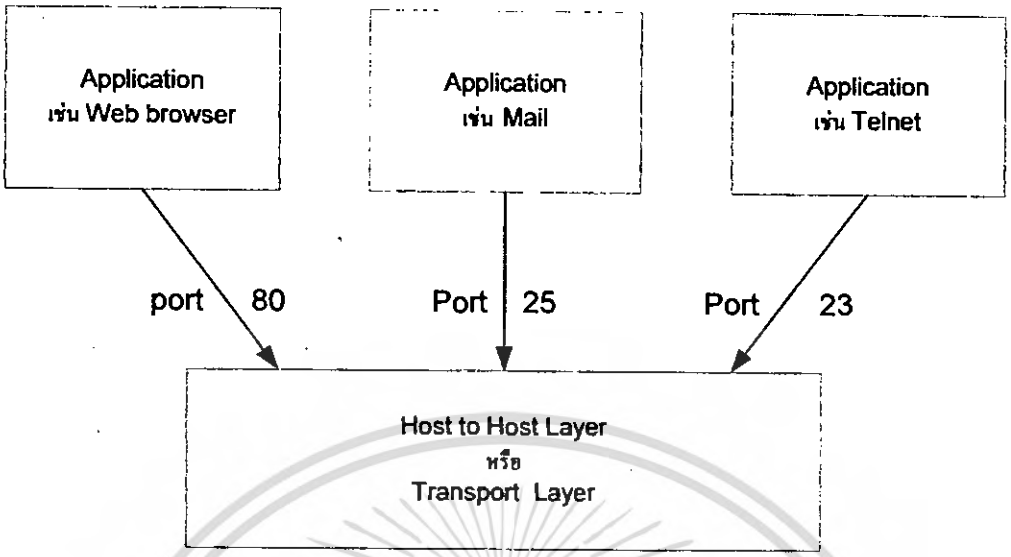
2.1.5 Host – To – Host Layer

การทำงานที่ชั้นของ Host – to – Host Layer นี้จะมีบทบาทในการจัดการต่อจาก Process Layer บางครั้งเรามักเรียกชั้น Host – to – Host ว่าเป็น Transport Layer ซึ่งไม่ใช่ชั้นของ Transport Layer ใน มาตรฐาน OSI model การทำงานของ Host – to Host Layer นี้จะมีการสร้าง connection หรือการเชื่อมต่อกัน ระหว่างแอพพลิเคชันกับ Host – to – Host Layer โดยจุดที่เชื่อมกันเพื่อรับส่งข้อมูลที่เรียกว่า port หรือ socket (คำว่า port ในที่นี้ไม่ได้หมายถึง port ทางฮาร์ดแวร์) และในแต่ละแอพพลิเคชันก็จะสร้างการเชื่อมต่อผ่าน port ได้พร้อมกันหลายแอพพลิเคชัน ซึ่งการใช้งาน port ของแต่ละแอพพลิเคชันที่อยู่ในชั้น Process Layer จะ แตกต่างกันตามหมายเลขที่กำหนดไว้ และแต่ละโปรโตคอลจะมีการใช้งาน port หมายเลขต่าง ๆ ไม่ซ้ำกัน



รูปที่ 2.5 แสดงการใช้งาน port ของแต่ละโปรโตคอล

เมื่อแอพพลิเคชันทำงานผ่านโปรโตคอลในชั้น Proccss Layer จะมีการส่งผ่านข้อมูลไปยัง Host – to - Host Layer ที่ชั้นนี้จะมีการเชื่อมต่อผ่าน port ที่กำหนด ทำให้การรับส่งข้อมูลในแต่ละโปรโตคอลทำได้ ถูกต้อง ถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลายโปรเซสที่แตกต่างกันก็ตาม หรือมี ผู้ใช้บริการเข้ามาใช้งานพร้อมกันจำนวนมากและหลายแอพพลิเคชันในเวลาเดียวกัน ในชั้น Host – to – Host หรือ Transport Layer ของ TCP/IP นี้ จะมีโปรโตคอลทำงานอยู่ 2 โปรโตคอลที่แตกต่างกัน คือ โปรโตคอล TCP และโปรโตคอล UDP (User Datagram Protocol) ในการส่งผ่านข้อมูลลงไปที่ชั้นถัด ๆ ไป เราจะเห็นว่า โปรโตคอล TCP และ UDP จะถูกผนึกเข้าไปในโปรโตคอล IP อีกทีหนึ่งและส่งต่อไปยังเครือข่ายอินเตอร์เน็ต ต่อไป



รูปที่ 2.6 แสดงการส่งข้อมูลจาก Application ไปยัง Host – to – Host Layer

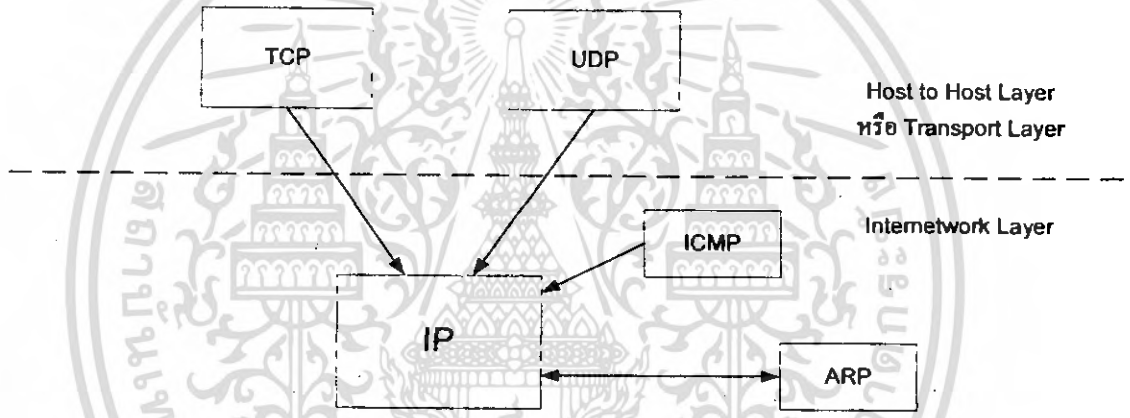
2.1.5.1 โปรโตคอล TCP

โปรโตคอล TCP (Transmission Control Protocol) เป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อย ๆ ก่อน แล้วจึงส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูล datagram ดังนั้นแอพพลิเคชันหรือโปรเซสใดที่อาศัยการส่งผ่านข้อมูลด้วยโปรโตคอล TCP จะต้องใช้หน่วยความจำและขนาดของช่องสัญญาณ (bandwidth) มากกว่า UDP

การติดต่อระหว่างกันจะต้องเป็นแบบ connection – oriented คือต้องมีการสร้างการติดต่อกันเป็น session ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (full duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้วจึงเริ่มการสนทนา เช่น พูดคำว่า "สวัสดี" หรือ "ฮัลโหล" กันก่อนเพื่อให้แน่ใจว่าฝ่ายตรงข้ามพร้อมจะติดต่อด้วย จากนั้นจึงเริ่มต้นติดต่อกัน และเมื่อต้องการจะเลิกการติดต่อก็จะมีการพูดคำว่า "สวัสดี" ให้ฝ่ายตรงข้ามทราบว่าจะเลิกการติดต่อและวางสายไป ซึ่งในระหว่างการติดต่อกันนั้น แม้ว่าฝ่ายใดฝ่ายหนึ่งหรือทั้งสองฝ่ายจะเงียบไป คือไม่พูดอะไรเป็นเวลานาน ๆ แต่การเชื่อมโยงระหว่างทั้งสองด้านยังคงมีอยู่ไม่ขาดไปจนกว่าฝ่ายใดฝ่ายหนึ่งจะวางสาย เช่นเดียวกับการติดต่อกันด้วยกลไกโปรโตคอล TCP เมื่อแอพพลิเคชันต้องการส่งผ่านข้อมูลจะใช้โปรโตคอลที่เหมาะสมในชั้น Process Layer ติดต่อไปและมีการสร้างช่องส่งข้อมูลผ่านพอร์ต ที่กำหนดเพื่อส่งผ่านข้อมูลไปยังโปรโตคอล TCP

2.1.5.2 โปรโตคอล UDP

ใน Host – to – Host Layer นอกจากจะมีโปรโตคอล TCP ทำงานแล้ว ก็ยังมีโปรโตคอล UDP (User Datagram Protocol) ในการส่งข้อมูลแต่ละครั้งและไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้แอพพลิเคชันหรือโปรเซสใดที่ต้องอาศัยโปรโตคอล UDP จะเป็นแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน (connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการ โดยไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโปรโตคอล TCP และไม่มีการตรวจสอบความถูกต้องครบถ้วนในการรับส่งข้อมูลนั้น ๆ ด้วย เนื่องจากโปรโตคอล UDP ไม่มีสัญญาณสอบทานข้อมูล (acknowledgement) ในการส่งข้อมูลแต่ละครั้ง และไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้แอพพลิเคชันหรือโปรเซสใดที่ต้องอาศัยโปรโตคอล UDP ในการส่งผ่านข้อมูลก็อาจจะต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง



รูปที่ 2.7 แสดงกลไกการส่งข้อมูลด้วยโปรโตคอล UDP

2.1.5.3 UDP Header

16-bit Source port	16-bit Destination port
length	checksum
Data	

ตารางที่ 2.3 UDP Header

ตำแหน่ง	ชื่อ	อธิบาย
บิต 0-15	Source port number	หมายเลขพอร์ตต้นทางที่ส่งดาต้าแกรมนี้ มีความยาว 16 บิต
บิต 16-31	destination port number	หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับดาต้าแกรม มีความยาว 16 บิตเช่นกัน
บิต 32-47	UDP length	ความยาวของดาต้าแกรม ทั้งส่วน Header และ data นั่น หมายความว่า ค่าที่น้อยที่สุดในฟิลด์นี้คือ 8 ซึ่งเป็นขนาดของ Header
บิต 48-63	Checksum	เป็นตัวตรวจสอบความถูกต้องของ UDP datagram และจะนำข้อมูลบางส่วนใน IP Header มาคำนวณด้วย

ตารางที่ 2.4 รายละเอียด UDP Header

2.1.5.4 UDP Checksum

Checksum เป็น เลข 16 บิตถูกคำนวณด้วยวิธี one's complement โดยนำ Pseudo Header และข้อมูลทั้งหมดใน UDP Datagram มาคำนวณ Pseudo Header เป็นข้อมูลที่อยู่ในส่วนของ IP Header ประกอบด้วยฟิลด์ source IP address , destination IP address , zero , protocol , UDP length

16-bit Source IP address		
16-bit Destination IP address		
zero	8-bit protocol (17 for UDP)	16-bit length

ตารางที่ 2.5 Pseudo Header

หากค่า Checksum ที่คำนวณออกมาเป็น 0 ค่า checksum จะถูกเซ็ตเป็น 1 ทั้งหมดแทน (มีค่าเท่ากันในระบบ 1's complement) ทั้งนี้เพราะในบางแอพพลิเคชันที่ไม่ต้องการตรวจสอบค่า checksum ในระดับ UDP จะเซ็ตค่านี้เป็น 0 (disable checksum)

2.1.6 Internetwork Layer

ในระดับล่างต่อมาในชั้น Internetwork Layer มีหน้าที่ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมีโปรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใด ๆ บนอินเตอร์เน็ต คือ โปรโตคอล IP (Internet Protocol) นอกจากนี้ในชั้น Internetwork Layer ยังมีโปรโตคอลทำงานอยู่ด้วยอีก 2 ชนิดคือ

โปรโตคอล Internet Control Message Protocol (ICMP) และโปรโตคอล Address Resolution Protocol (ARP) แต่ที่ใช้ในการทำโครงงานนี้ คือ โปรโตคอล IP

2.1.6.1 โปรโตคอล IP

โปรโตคอล IP ทำหน้าที่ให้บริการส่งผ่านข้อมูลที่มาจาก Host – to– Host Layer เพื่อส่งข้ามไปยังเครือข่ายใด ๆ ได้อย่างถูกต้อง แม้ว่าจะมีเครือข่ายเชื่อมต่อกันอยู่ในอินเตอร์เน็ตเป็นล้าน ๆ เครือข่ายก็ตาม เนื่องจากโปรโตคอล IP มีข้อมูลตำแหน่ง IP ปลายทางที่จะส่งข้อมูลไปให้โดยทำงานร่วมกับอุปกรณ์ router เพื่อส่งข้อมูลข้ามเครือข่ายออกไปได้ ตัวโปรโตคอล IP จะทำงานแบบ packet switching คือมีการส่งข้อมูลผ่านสวิตช์ (switch) ไปยังปลายทาง โดยข้อมูลจะเดินทางไปยังเครือข่ายต่าง ๆ ผ่านสวิตช์นี้ไปเรื่อง ๆ จนกว่าจะถึงปลายทาง ตัววงจรผ่านหรือ switch นี้อาจเป็น gateway หรือ router ในระบบเครือข่ายก็ได้ ซึ่งในข้อมูลของโปรโตคอล IP จะมีข้อมูลของหมายเลข IP ที่จะส่งข้อมูลไปและเมื่อถึงเครือข่ายปลายทางแล้ว จะมีกลไกแปลงหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์ประจำเครื่องที่ถูกต้องอีกทีหนึ่งด้วยโปรโตคอล ARP

2.1.6.2 IP Datagram

IP Datagram ประกอบด้วย IP Header และ IP Payload

IP Header เป็นขนาดที่เปลี่ยนแปลงได้ระหว่าง 20 และ 60 ไบต์ ในการเพิ่มขึ้น 4 ไบต์ มันจะจัดเตรียมการสนับสนุน routing , การแสดงตัว payload , การชี้ให้เห็นถึงขนาด IP Header และ Datagram , การสนับสนุน fragmentation

Version	IP Header Length	Type of Service	Total Length	Identifier	Flags	Fragment Offset
4 bit	4 bit	8 bit	16 bit	16 bit	3 bit	13 bit

Time – to – Live	Protocol	Header Checksum	Source IP Address	Destination IP Address	IP Option and Padding
8 bit	8 bit	16 bit	32 bit	32 bit	32 bit

ตารางที่ 2.6 แสดงโครงสร้าง IP Header

IP Payload เป็นขนาดที่เปลี่ยนแปลงโดยมีลำดับจาก 8 ไบต์ (68 ไบต์ IP Datagram กับ 60 ไบต์ IP Header) ถึง 65,515 ไบต์ (65,535 ไบต์ IP Datagram กับ 20 ไบต์ IP Header)

2.1.6.3 โปรโตคอล

โปรโตคอล Field มีขนาดยาว 1 ไบต์ และใช้เป็นตัวบ่งบอกในการบรรจุข้อมูลไปยัง Layer ที่สูงขึ้น กับใน IP Payload และยังเป็นตัวบ่งบอกลูกข่ายโปรโตคอลที่ชัดเจน โดยปกติค่าของ IP Protocol Field จะเป็น 1 สำหรับ ICMP , 6 สำหรับ TCP และ 17 (0x11) สำหรับ UDP โปรโตคอล field จะแสดงตัวได้อย่างมากมาย ดังนั้น payload สามารถที่จะผ่านไปยัง Layer ที่สูงขึ้นไปได้ถูกต้อง โดยจะได้รับที่ destination

Value	Protocol
0	Reserved
1	Internet Control Message Protocol (ICMP)
2	Internet Group Management Protocol (IGMP)
4	IP in IP encapsulation
6	Transmission Control Protocol (TCP)
8	Exterior Gateway Protocol (EGP)
17	User Datagram Protocol (UDP)
46	Resource Reservation Protocol (RSVP)
47	Generic Routing Protocol (GRE)
50	IP Security Encapsulating Security Payload (ESP)
51	IP Security Authentication Header (AH)
89	Open Shortest Path First (OSPF)

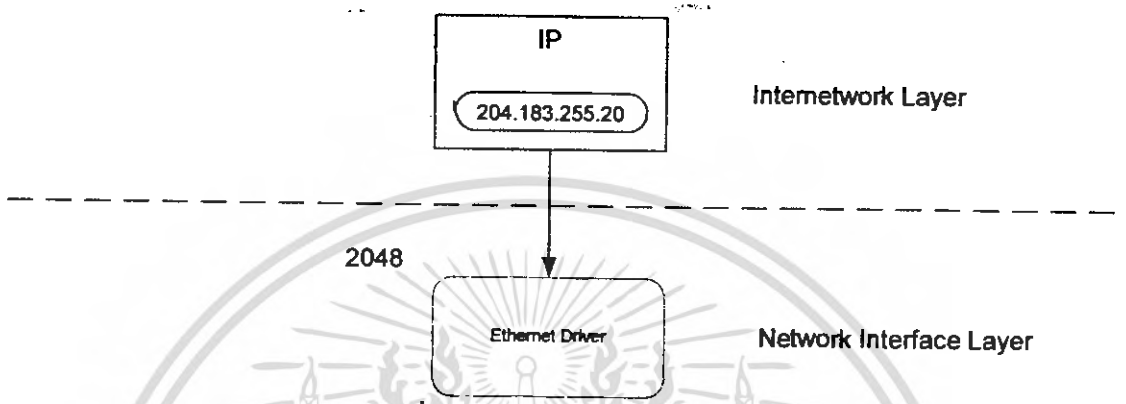
ตารางที่ 2.7 ค่าต่างๆของ IP Protocol Field

2.1.6.4 การกำหนด IP address ให้กับอุปกรณ์

ต้องกำหนดหมายเลข IP address ให้กับจุดเชื่อมต่อเข้ากับเครือข่ายทุก จุดเชื่อมต่อหรือ Interface อาจจะหมายถึง Network Interface card (Lan การ์ด) ที่ติดตั้งในเซิร์ฟเวอร์หรือ WAN port , Ethernet port ที่ Router ใช้เชื่อมต่อเข้ากับเครือข่ายเป็นต้น การกำหนดหมายเลข IP address ให้กับจุดเชื่อมต่อนี้ทำให้เราเข้าใจได้ว่าในบางอุปกรณ์ที่มีจุดเชื่อมต่อเข้ากับเครือข่ายมากกว่าหนึ่งจุด ต้องกำหนดหมายเลข IP address ให้ครบ

2.1.6.5 การ Bind IP address

เมื่อได้กำหนดหมายเลข IP address ให้กับจุดเชื่อมต่อเช่น Lan card แล้วที่เครื่องเซิร์ฟเวอร์จะต้องมีการ bind หรือผนวกค่า IP address ดังกล่าวเข้ากับ Ethernet driver เพื่ออ้างอิงหมายเลข IP กับฮาร์ดแวร์ให้ทำหน้าที่ติดต่อส่งข้อมูลในระดับ Network Interface ได้ต่อไป ดังตัวอย่างในรูปที่ 2.8



รูปที่ 2.8 แสดงการ Bind IP address

จากรูปจะแสดงค่า bind IP address 204.183.255.20 เข้ากับ Ethernet driver โปรโตคอล IP จะใช้ค่า IP address นี้ในการติดต่อกันและผ่านฮาร์ดแวร์ที่ถูก bind ไว้อีกต่อหนึ่ง ค่าหมายเลขฮาร์ดแวร์ก็ได้แก่ MAC address ที่มีประจำอยู่บน Lan card ซึ่งจะไม่ได้ใช้งานอ้างอิงโดยตรงแต่จะผ่านหมายเลข IP address แทน

2.1.6.6 โปรโตคอล ARP

โปรโตคอล ARP (Address Resolution Protocol) ถูกเรียกใช้งานโดยโปรโตคอล IP เพื่อช่วยแปลงหมายเลข IP ไปเป็นหมายเลขฮาร์ดแวร์ปลายทาง ตัวอย่างเช่น เว็บเซิร์ฟเวอร์เครื่องหนึ่งเชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ต และในการเชื่อมต่อนี้ต้องอาศัย Network Interface Card (NIC) หรือ LAN card ติดตั้งอยู่ที่ LAN card นี้เองจะมีหมายเลขเฉพาะประจำฮาร์ดแวร์ที่ไม่ซ้ำกับใคร เพื่อใช้อ้างอิงการส่งข้อมูลในเครือข่าย แต่เมื่อมาใช้งานในโปรโตคอล TCP/IP ก็จะต้องมีการกำหนดหมายเลข IP address ประจำตัวเพื่อใช้อ้างอิงกัน และโปรโตคอล ARP จะทำหน้าที่แปลงค่าหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์จริงให้ในระดับการทำงานที่ Internetwork Layer นี้ ซึ่งกลไกการแปลงนี้เรียกว่า address resolution

2.1.6.7 โปรโตคอล ARP ย้อนกลับ RARP (Reverse Address Resolution Protocol)

วิธีการ ARP ช่วยแก้ปัญหาในการค้นหาที่อยู่ ของข้อมูลที่ใช้การกำหนดที่อยู่ฮาร์ดแวร์ แบบ IP แต่ถ้าทราบที่อยู่แบบ ฮาร์ดแวร์ แล้วต้องการแปลงที่อยู่เป็น IP จะทำอย่างไร ปัญหานี้มักเกิดขึ้นกับเครื่อง Computer ที่เริ่มทำงานด้วยการอ่านข้อมูลทั้งหมดจากเครื่อง Host เครื่องประเภทนี้จะทราบเพียงที่อยู่ ของตนเองจากอุปกรณ์สื่อสารเครือข่ายเท่านั้น การค้นหาคำตอบสามารถทำได้โดยวิธีควบคุมการสื่อสารแบบ ARP

ย้อนกลับ หรือ RARP (Reverse Address Resolution Protocol) วิธีการนี้ Computer ที่เพิ่งจะเริ่มทำงาน (หรือเครื่องใดก็แล้วแต่) จะส่งคำถามออกไปในทำนอง "ที่อยู่ขนาด 48 Bits แบบ ฮาร์ดแวร์ ของฉันคือ 14.04.05.18.01.25 มีใครทราบที่อยู่ IP ของฉันบ้าง" เครื่องที่ให้บริการ RARP จะตรวจดูข้อมูลในตารางข้อมูลของตนเองแล้วจึงส่งหมายเลข IP กลับไปให้ วิธีการนี้ช่วยให้เกิดความอ่อนตัวและเพิ่มประสิทธิภาพในการใช้หมายเลข IP เนื่องจากผู้ใช้ไม่มีหมายเลข IP เป็นของตนเอง ผู้ควบคุมระบบสามารถกำหนดหมายเลข IP ใดๆที่ไม่มีผู้ใช้งานในขณะนั้นให้ใช้ได้ หมายเลข IP ในที่นี้จึงเป็นเสมือนสมบัติส่วนกลางที่ทุกคนใช้ร่วมกัน

2.1.7 Network Interface Layer

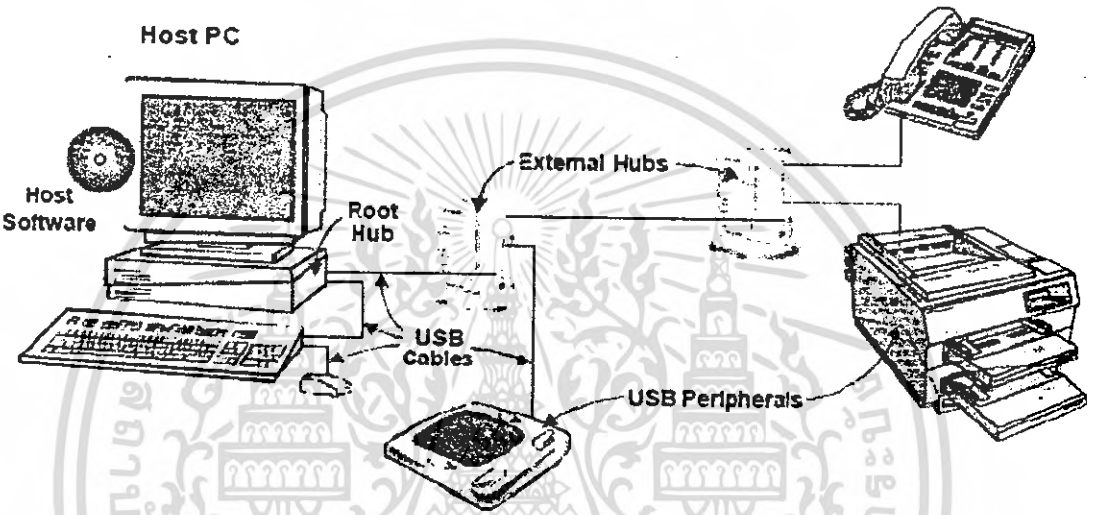
การทำงานระดับล่างสุดต่อจาก Internetwork Layer จะเป็นการแปลงข้อมูล IP datagram ให้อยู่ในรูปที่เหมาะสม และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่ายต่อไป ซึ่งในชั้น Network Interface Layer นี้เมื่อเทียบกับมาตรฐาน OSI model แล้วจะเป็นการรวม 2 Layer เข้าด้วยกันคือ Data link layer และ Physical Layer กล่าวโดยสรุปคือ การทำงานในชั้นต่าง ๆ ตามโครงสร้างของโปรโตคอล TCP/IP

2.1.7.1 MAC Address

เนื่องจากคอมพิวเตอร์แต่ละเครื่องสามารถแชร์ข้อมูลกันได้ในระบบเครือข่ายเดียวกัน ดังนั้นแต่ละเครื่องควรมีสิ่งที่ชี้ลักษณะเฉพาะตัวของมัน เช่น เราต้องมีบัตรประจำตัวประชาชน ซึ่งในทางคอมพิวเตอร์นี้เราจะใช้เลขฐาน 16 จำนวน 12 digits เป็นตัวบ่งชี้ลักษณะเฉพาะนั้น ๆ ซึ่งเราเรียกว่า MAC Address เนื่องจาก MAC Address เป็นตัวบ่งชี้ลักษณะเฉพาะของแต่ละ machine ดังนั้นจึงต้องเป็นค่าที่ไม่ซ้ำกัน (unique) MAC Address เป็นเลข 48 bits โดยแบ่งออกเป็น 2 ส่วน โดย 24 bits แรกเป็นค่าที่แสดงถึงบริษัทที่ผลิตการ์ดนั้น ๆ ส่วน 24 bits หลังเป็น serial number ที่ทางบริษัทกำหนดให้ ซึ่งแต่ละตัวต้องไม่ซ้ำกัน เราเรียกเลข 24 bit นี้ว่า OUI (Organizationally Unique Identifier) ซึ่ง OUI จะใช้เพียง 22 bits เท่านั้น ส่วนอีก 2 bits ที่เหลือจะถูกใช้เพื่อวัตถุประสงค์อื่น โดย bit หนึ่งจะใช้เพื่อแสดงว่า address นั้นเป็น broadcast/multicast address ส่วนอีก bit หนึ่งนั้นไว้แสดงว่า adapter นั้นถูกกำหนด locally administered address ซึ่ง admin ของระบบจะทำการกำหนด MAC Address เพื่อความเหมาะสมของนโยบายระบบ

2.2 จุดกำเนิดของยูเอสบี

พอร์ตแต่ละชนิดของคอมพิวเตอร์ได้รับการออกแบบมา เพื่องานเฉพาะ ทำ ให้อุปกรณ์แต่ละตัวต้องเลือกใช้พอร์ตต่าง ๆ ชนิดกันไป ส่งผลให้การนำอุปกรณ์ต่าง ๆ มา เชื่อมต่อกับ เครื่องคอมพิวเตอร์เป็นเรื่องที่ต้องให้ความสนใจเนื่องจากอุปกรณ์แต่ละชนิดมีคอนเน็กเตอร์ที่ใช้เชื่อมต่อแตกต่างกันไปตามชนิดของพอร์ต เช่น พอร์ตอนุกรมที่ใช้ต่อคีย์บอร์ด และ เมาส์ (ทั้ง PS/2 และ พอร์ตอนุกรมมาตรฐาน) พอร์ต ขนานสำหรับต่อเครื่องพิมพ์ พอร์ตเชื่อมต่อจอภาพและอื่น ๆ ทำให้ผู้ใช้ที่ไม่มีความรู้ในการติดตั้งหรือเรียกกันว่า เอ็นด์ยูสเซอร์ (End user) พบกับความยากลำบากในการเรียนรู้เรื่องราวเหล่านี้



รูปที่ 2.9 การเชื่อมต่ออุปกรณ์ต่าง ๆ ด้วย ยูเอสบี

นอกจากนั้นการติดตั้งอุปกรณ์ต่าง ๆ เพิ่มเข้าไปในเครื่องคอมพิวเตอร์จะเกิดปัญหาการแย่งกันใช้สัญญาณ IRQ (Interrupt Request) ซึ่งเป็นตัวจำกัดจำนวนอุปกรณ์ที่มาต่อ ทำให้เกิดแนวคิดที่จะกำหนดมาตรฐานที่สร้างเป็นพอร์ตที่ทำหน้าที่เชื่อมต่อทั้งหมดอยู่ในรูปแบบเดียวกันง่ายต่อการใช้งานของผู้ใช้ทั่ว ๆไป และไม่มีข้อจำกัดในการใช้ IRQ คำตอบของแนวคิดนี้คือ พอร์ต ยูเอสบีนั่นเอง

พอร์ตยูเอสบี ถ้าแปลแบบตรงตัวก็จะได้ความหมายว่า บัสอนุกรมอเนกประสงค์ คุณสมบัติต่าง ๆ ที่ทำให้สามารถกำหนดชื่อนี้ให้กับพอ์ตชนิดนี้ได้มีดังนี้

- ใช้คอนเน็กเตอร์เพียงชนิดเดียว ซึ่งมี 2 รูปแบบสำหรับการเชื่อมต่ออุปกรณ์ทุก ๆ ชนิดเข้ากับคอมพิวเตอร์
- สามารถเชื่อมต่อหลาย ๆ ชนิดรวมเข้าสู่คอนเน็กเตอร์เดียวกัน สูงสุด 128 ตัว
- ไม่เกิดการขัดแย้งกันของการเข้าใช้ทรัพยากรของระบบ (IRQ)
- ตรวจสอบการเชื่อมต่อและตั้งค่าการทำงาน อัตโนมัติ ระหว่างที่เครื่องกำลังทำงานอยู่ (Hot attachment)

- ความเร็วในการถ่ายทอดจะขึ้นอยู่กับมาตรฐาน อันมีรายละเอียดดังนี้

  มาตรฐาน USB 1.0/1.1 : มีอัตราการถ่ายทอดข้อมูลความเร็วต่ำ (low speed) เท่ากับ 1.5 เมกะบิตต่อวินาที (Mbit/sec) และความเร็วเต็มที่ (full speed) เท่ากับ 12 เมกะบิตต่อวินาที (Mbit/sec)

  มาตรฐาน USB 2.0 : จะมีอัตราเร็วในการถ่ายทอดเพิ่มขึ้นอีก 1 ระดับคือ ความเร็วสูง (high speed) ซึ่งมีความเร็วสูง ถึง 480 เมกะบิตต่อวินาที (Mbit/sec)

- ที่ขาพอร์ต ยูเอสบีมีแรงดันไฟตรง +5V จ่ายออกมา ทำให้อุปกรณ์ต่อพ่วงที่ใช้ พลังงานไม่มาก นักสามารถใช้แรงดันจากพอร์ต นี้เป็นไฟเลี้ยงเพื่อทำงานได้ โดยไม่ต้องอาศัยแหล่งจ่ายจากภายนอกเพิ่มเติมอีก

2.2.1 การส่งข้อมูลภายในบัสยูเอสบี 1.0/1.1

ยูเอสบีเป็นการส่งข้อมูลที่มีรูปแบบ การเชื่อมต่อใน ระบบบัสคือ อุปกรณ์ทุก ๆ ตัวจะต้องส่งสัญญาณรวมกันไปในสายส่งสัญญาณเพียงคู่เดียว ดังนั้นอุปกรณ์ทุก ๆ ตัวที่เชื่อมต่อกับบัสจะต้องส่งข้อมูลเรียงลำดับกันไป เพื่อไม่ให้เกิด การชนกัน ของข้อมูล และ เนื่องจากยูเอสบี เป็นระบบบัสที่ ใช้สายสัญญาณ เพียงคู่เดียว (2 เส้น) ทำให้ช่วงเวลาหนึ่ง ๆ จะมีข้อมูลวิ่งไปได้ทิศทางเดียวเท่านั้น ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันได้หรือที่เรียกกันว่า การส่งข้อมูลแบบฮาร์ฟดูเพล็กซ์ (Half duplex)

จังหวะการรับส่งข้อมูล ของระบบบัสยูเอสบี ทั้งหมดจะถูกควบคุมจาก โฮสต์ ซึ่งก็คือ คอมพิวเตอร์ที่เป็นจุดรวมของอุปกรณ์ทุกตัวที่เชื่อมต่ออยู่นั่นเอง ดังนั้นจึงไม่สามารถเชื่อมต่อเครื่องคอมพิวเตอร์ 2 เครื่องให้รับส่งข้อมูลถึงกันได้โดยตรง เพราะถ้าคอมพิวเตอร์ทั้งสองเครื่องทำหน้าที่เป็นโฮสต์ทั้งคู่จะเกิดการชนกันของข้อมูล เนื่องจากแต่ละเครื่องก็จะพยายามกำหนดจังหวะในการรับส่งของตัวเองขึ้นมาดังนั้นจะ เชื่อมต่อ คอมพิวเตอร์ สอง เครื่องเข้า ด้วย กัน ผ่านยูเอสบี จะต้องมีอุปกรณ์ที่เป็นตัวกลางเพื่อซิงโครไนซ์ตัวเองเข้ากับโฮสต์ทั้งสองให้ได้

การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรม โดยทุก ๆ 1 มิลลิวินาที จะเกิดการรับส่งข้อมูลขึ้น1 เฟรม ในแต่ละเฟรมจะแบ่งออกเป็นแพ็กเก็ต (packet) เริ่มต้นการทำงานของแต่ละเฟรมโดยโฮสต์จะส่งสัญญาณเริ่มต้นหรือ SOF (Start Of Frame) ออกไปเพื่อให้อุปกรณ์ทุกตัวรู้จังหวะการเริ่มเฟรม หลังจากนั้น

SOF	ส่งเสียงไปยังโทรศัพท์	รับเสียงจากโทรศัพท์	ส่งข้อมูลไปยังโมเด็ม	รับข้อมูลจากโมเด็ม	ส่งเสียงไปยังลำโพง	รับข้อมูลจากเมาส์	รับข้อมูลจากคีย์บอร์ด	ส่งข้อมูลไปยังเครื่องพิมพ์	รับข้อมูลจากสแกนเนอร์

← 1 มิลลิวินาที →

ตารางที่ 2.8 การจัดลำดับการรับส่งข้อมูลของอุปกรณ์แต่ละตัว

โฮสต์ก็จะเริ่มส่งหรือรับข้อมูลต่าง ๆ ตามที่ได้จัดลำดับความ สำคัญไว้อุปกรณ์ต่าง ๆ ที่อยู่ภายในบัสจะต้องทำงานตามจังหวะที่โฮสต์กำหนดไว้เท่านั้นการส่ง ข้อมูลกลับไปยังโฮสต์จะสามารถทำได้ก็ต่อเมื่อได้รับการถามหรือร้องขอจากโฮสต์

แต่เนื่องจากแต่ละเฟรมข้อมูล จะต้อง รับส่งภายใน 1 มิลลิวินาทีนั้น หมายความว่าข้อมูล ของอุปกรณ์ทุก ๆ ตัวที่เชื่อมต่อกับบัสจะต้องถูกกำหนดให้ขนาดไม่ใหญ่เกินกว่าที่จะสามารถรับส่งได้ภายใน 1 มิลลิวินาที และเล็กพอที่จะให้อุปกรณ์ทุก ๆ ตัวสามารถ ใช้งานบัสไปพร้อม ๆ กันได้ดังนั้นในระบบบัส ยูเอสบี จึงจำเป็นต้องอาศัยซอฟต์แวร์ที่เข้ามาจัดการ ในด้านนี้ และยังต้องอาศัย ฮาร์ดแวร์ที่จะคอย กระจายการส่งและรวบรวมการรับข้อมูลจาก อุปกรณ์ทุก ๆ ตัวในระบบ ซึ่งซอฟต์แวร์และ ฮาร์ดแวร์ ที่จำเป็น ต่อส่วน ประกอบสำหรับระบบยูเอสบี

2.2.1 .1 ส่วนประกอบทางซอฟต์แวร์

ไดรเวอร์อุปกรณ์ยูเอสบี

ไดรเวอร์อุปกรณ์ ยูเอสบี คือโปรแกรมเก็บข้อมูลที่จำเป็นในการติดต่อไปยังอุปกรณ์แต่ละตัวเมื่อโปรแกรมใดมีความต้องการจะติดต่อกับอุปกรณ์ต่าง ๆจะต้องแจ้งความต้องการนั้นมายังไดรเวอร์อุปกรณ์ยูเอสบี เนื่องจากไดรเวอร์นี้จะรู้ว่าถ้าต้องการติดต่อกับอุปกรณ์จะต้องติดต่อผ่านเอ็นด์พอยต์ ไหนด้วยรูปแบบใด (การทำงานของอุปกรณ์ ยูเอสบี จะต้องติดต่อส่งผ่านเอ็นด์พอยต์ของตัวอุปกรณ์ ซึ่งอุปกรณ์ต่าง ๆ จะมีชนิดและจำนวนเอ็นด์พอยต์ที่ต่างกัน) ดังนั้นอุปกรณ์แต่ละตัวจะมี ไดรเวอร์อุปกรณ์ ยูเอสบีเฉพาะตัวซึ่งเมื่อถึงคราวต้องนำอุปกรณ์นั้นมาต่อใช้งานกับคอมพิวเตอร์จริง ๆ ก็ต้องนำไดรเวอร์ตัวเดียวกันนั้นมาติดตั้งเพิ่มเข้ากับระบบปฏิบัติการในคอมพิวเตอร์เพื่อให้ระบบรู้จักและติดต่อใช้งานอุปกรณ์ที่ติดตั้งมาใหม่นี้ได้

แต่ในบางอุปกรณ์ที่เป็นอุปกรณ์พื้นฐานของเครื่องคอมพิวเตอร์ เช่น เมาส์ คีย์บอร์ดจะมีการบรรจุไดรเวอร์อุปกรณ์ ยูเอสบี ของอุปกรณ์เหล่านี้ไว้ภายในไบออส (BIOS) ของเครื่องคอมพิวเตอร์เรียบร้อยแล้ว จึงไม่ต้องติดตั้งไดรเวอร์เพิ่มเติมสำหรับอุปกรณ์เหล่านี้เพียงเข้าไปเปิดการทำงานไบออสก็จะทำให้เครื่องคอมพิวเตอร์รู้จักอุปกรณ์เหล่านี้เอง

ไดรเวอร์ยูเอสบี

การทำงานของยูเอสบีนั้นเป็นการต่อรวมกันของอุปกรณ์หลาย ๆ ชนิดบนสายสัญญาณเพียงคู่เดียว ดังนั้น การส่งข้อมูลของอุปกรณ์ แต่ละชนิดจะต้อง มีการแบ่งสรรปันส่วนกันไป อย่างเหมาะสมพอดี เพื่อให้อุปกรณ์ทุกตัวสามารถทำงานไปได้พร้อม ๆ กันและแน่นอนว่าต้องมีซอฟต์แวร์เข้ามาทำหน้าที่นี้ซึ่งก็คือ ไดรเวอร์ยูเอสบีนั่นเอง ไดรเวอร์อุปกรณ์ ยูเอสบี ของอุปกรณ์แต่ละตัวจะส่งการร้องขอเพื่อการติดต่อ (request) ลงมายัง ไดรเวอร์ยูเอสบี และเมื่อไดรเวอร์ยูเอสบีรับทราบความต้องการการติดต่อของ อุปกรณ์ครบทุก ๆตัวที่เชื่อมต่ออยู่กับบัสแล้ว ก็พิจารณาว่าในรองรับการส่งข้อมูลหนึ่ง ๆ นั้นอุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้

มากเพียงใด หากปริมาณ ข้อมูลมาก ก็จะทำการ แบ่งออก เป็นส่วน ๆ แล้วเก็บไว้เพื่อรอส่งรอบ ถัดไปโดย ปริมาณข้อมูลที่ส่งได้ของอุปกรณ์แต่ละตัวจะถูกพิจารณาจากชนิดการถ่ายทอดข้อมูล

ไดรเวอร์โฮสต์คอนโทรลเลอร์

หลังจากไดรเวอร์ยูเอสบีพิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้าง มันจะส่งข้อมูลของอุปกรณ์แต่ละตัวที่จะติดต่อในรอบการติดต่อนั้นมายังไดรเวอร์โฮสต์คอนโทรลเลอร์จากนั้นไดรเวอร์โฮสต์คอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลของอุปกรณ์แต่ละ ชนิดลง เป็นเฟรม ข้อมูลเพิ่ม เติมส่วนประกอบต่าง ๆ ของเฟรมข้อมูลให้ครบตามมาตรฐานการถ่ายทอดข้อมูลแบบ ยูเอสบีแล้วส่งข้อมูลไปยังฮาร์ดแวร์ยูเอสบีโฮสต์คอนโทรลเลอร์เพื่อส่งข้อมูลทั้งหมดออกไปยังอุปกรณ์ต่าง ๆ

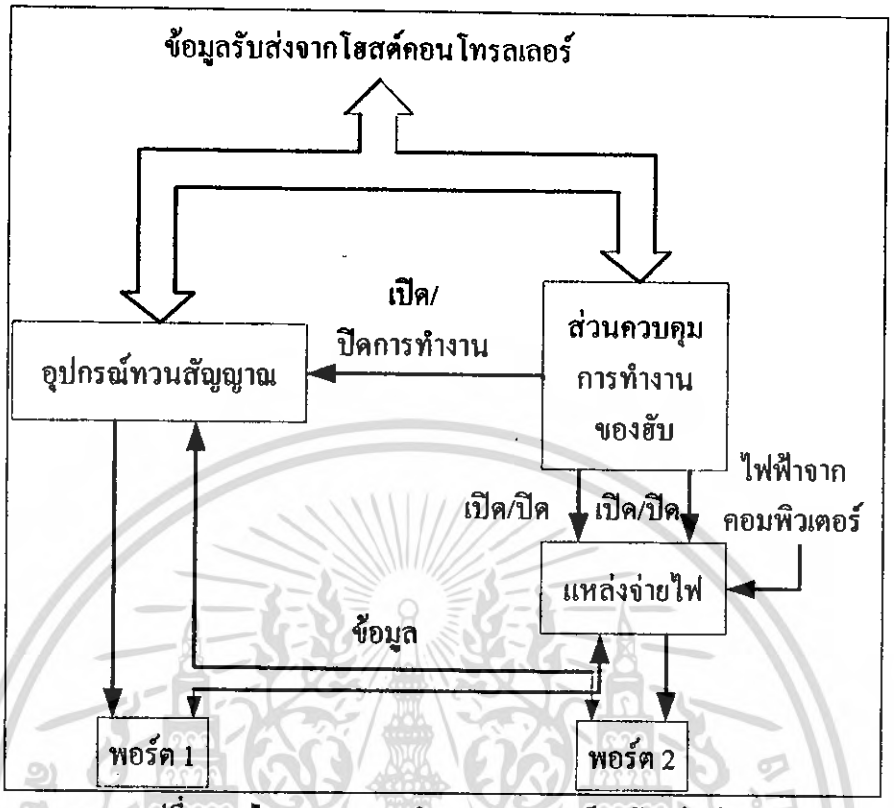
2.2.1.2 ส่วนประกอบทางฮาร์ดแวร์

จากส่วนประกอบ 3 ส่วนที่ผ่านมาเป็นส่วนประกอบด้านซอฟต์แวร์ซึ่งจะคอยควบคุมจัดการการทำงานของอุปกรณ์ที่นำมาต่อรวมกันทั้งหมดให้เป็นไปอย่างราบรื่น ในส่วนถัดไปจะเป็นหน้าที่ของส่วนประกอบทางด้านฮาร์ดแวร์ที่ทำหน้าที่ รับส่งสัญญาณกับตัวอุปกรณ์ต่าง ๆ โดยส่วนประกอบตัวแรก ที่จะกล่าวถึงคือยูเอสบีโฮสต์คอนโทรลเลอร์ (USB host controller) ยูเอสบีรูตฮับ (USB root hub)

ยูเอสบีโฮสต์คอนโทรลเลอร์/ ยูเอสบีรูตฮับ

ยูเอสบีโฮสต์คอนโทรลเลอร์ มีหน้าที่สร้างสัญญาณข้อมูลทางไฟฟ้า แล้วส่งต่อไปยังรูตฮับเพื่อกระจายไปยังอุปกรณ์ต่าง ๆ โดยมันจะสร้างข้อมูลการติดต่อรูปแบบต่าง ๆ ตามที่ไดรเวอร์โฮสต์คอนโทรลเลอร์กำหนดมาให้จากนั้นจะแปลงข้อมูลที่ส่งแบบขนาน มาเป็นอนุกรม เพื่อใช้ในการส่งต่อไปเมื่อสัญญาณที่ต้องการส่งมาถึงรูตฮับ รูตฮับจะส่งสัญญาณนั้นออกไปยังบัสเพื่อส่งต่อไปยังอุปกรณ์ต่าง ๆ นอกจากนั้นรูตฮับยังมีหน้าที่สำคัญอีก 4 อย่างคือ

- ควบคุมการใช้พลังงานของอุปกรณ์ที่มาต่อ
- ตรวจสอบการเชื่อมต่อของอุปกรณ์ว่ามีอุปกรณ์ต่ออยู่หรือไม่
- เปิดหรือ enable การใช้งานพอร์ตเมื่อมีอุปกรณ์ต่ออยู่ และปิดหรือ disable การใช้งานเมื่อปลดอุปกรณ์ออกไปแล้ว
- รายงานสถานะของแต่ละพอร์ตเมื่อไดรเวอร์โฮสต์คอนโทรลเลอร์ร้องขอมา



รูปที่ 2.10 ไดอะแกรมการทำงานของ ยูเอสบีรูตฮับอย่างง่าย

ยูเอสบีฮับ

หน้าที่หลักของ ยูเอสบีฮับ คือขยายการเชื่อมต่อให้อุปกรณ์จำนวนมากสามารถเชื่อมต่อเข้ากับระบบบัสได้ โดยการทำงานหลักของ ยูเอสบีฮับ นั้นมีอยู่ 2 ส่วนคือ ทำหน้าที่เป็นตัวทวนสัญญาณ (repeater) และตัวจัดการพลังงาน (Power management)

ในส่วนของการทวนสัญญาณ ยูเอสบีฮับจะต้องรับสัญญาณจากโฮสต์มาแล้วส่งกระจายออกไปยังพอร์ตทุก ๆ พอร์ต และ รับสัญญาณจาก แต่ละพอร์ต แล้วจับรวมกัน เพื่อส่งกลับไปโฮสต์สำหรับส่วนของการจัดการพลังงานนั้นมีหน้าที่เหมือนกับรูตฮับก็คือตรวจสอบว่ามีการต่ออยู่ของอุปกรณ์ที่พอร์ตใดบ้างหากมีการต่ออยู่ก็เปิดการใช้งานที่พอร์ตนั้น ๆ หากไม่มี การต่อก็จะปิดการใช้งาน และป้องกันอุปกรณ์ที่ต่ออยู่ในแต่ละพอร์ตไม่ให้ดึงกระแสไฟฟ้าเกินกว่าที่กำหนด

อุปกรณ์ยูเอสบี

ส่วนประกอบทาง ฮาร์ดแวร์ส่วน สุดท้ายที่ ต้องรู้จักคือ อุปกรณ์ ยูเอสบีซึ่งก็คืออุปกรณ์ต่าง ๆ ที่เชื่อมต่อกับคอมพิวเตอร์ด้วยพอร์ตยูเอสบีนั่นเองสามารถแบ่งออกเป็น 2 ชนิด (ยกเฉพาะมาตรฐานยูเอสบี1.0/1.1) ตามความเร็วในการถ่ายทอดข้อมูลคือ

- อุปกรณ์ความเร็วต่ำ (low speed device) ถ่ายทอดด้วยความเร็ว1.5 เมกะบิตต่อวินาที (Mb/s)
- อุปกรณ์ความเร็วเต็มที่ (full speed device) รับส่งข้อมูลด้วยความเร็ว 12 เมกะบิตต่อวินาที (Mb/s) ปัจจุบันนี้อุปกรณ์ ยูเอสบีที่จำหน่ายตามท้องตลาดมีจำนวนมาก อาทิ คีย์บอร์ด, เมาส์, จอยสติ๊ก เหล่านี้คืออุปกรณ์ความเร็วต่ำส่วนมอนิเตอร์, ลำโพง, เครื่องพิมพ์, กล้องถ่ายรูปดิจิตอล, ซีดีไดรฟ์, เครื่องเล่น MP3 จัดเป็นอุปกรณ์ ยูเอสบีความเร็วสูงอุปกรณ์บางตัวจะบรรจุความสามารถของ ยูเอสบีฮับไปด้วยสามารถทำให้อุปกรณ์อื่น ๆ มาเชื่อมต่อได้ เหมือนกับ การต่อเข้ากับฮับ อุปกรณ์เหล่านี้เรียกว่า Compound USB device ตัวอย่างของอุปกรณ์ที่มีฮับอยู่ภายในได้แก่ มอนิเตอร์ หรือเครื่องพิมพ์ เป็นต้น นอกจากแบ่งชนิดของอุปกรณ์ตามความเร็วในการถ่ายทอดข้อมูลแล้วอาจแบ่งกลุ่มตามการใช้พลังงานของอุปกรณ์เองก็ได้ ซึ่งสามารถแบ่งได้อีก 2 ชนิด คือ

  อุปกรณ์ใช้ไฟเลี้ยงจากบัส (bus powered device) คืออุปกรณ์ใช้ไฟเลี้ยงจากบัสโดยตรงไม่ต้องใช้ไฟเลี้ยงจากภายนอกเพิ่มเติม

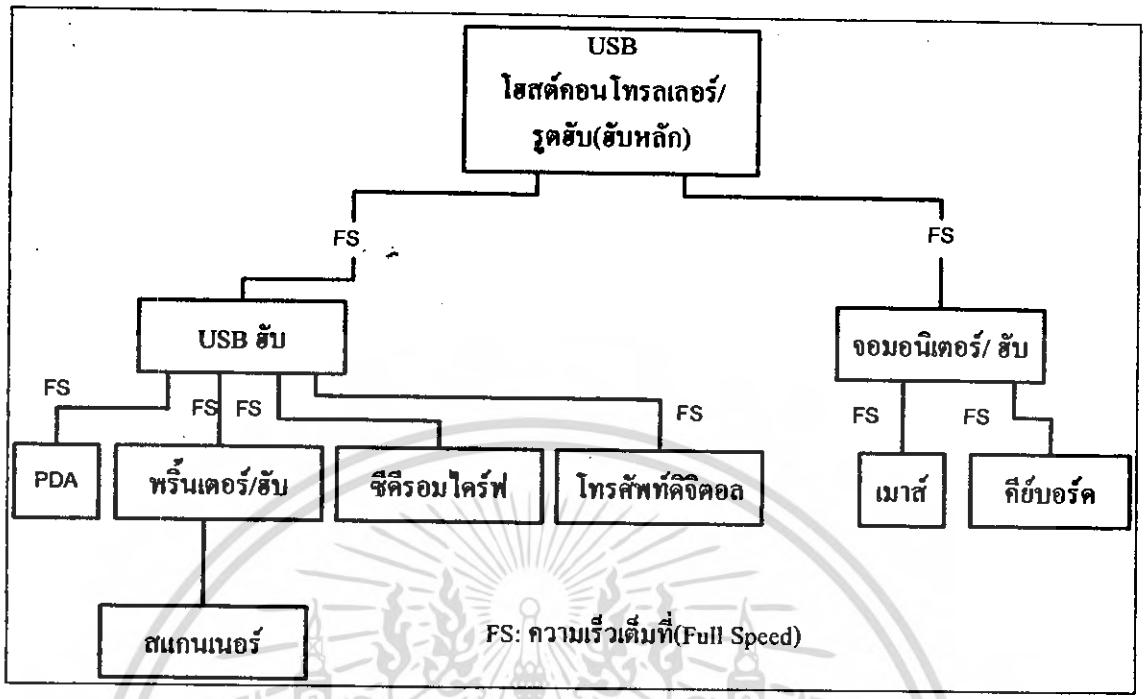
  อุปกรณ์ใช้ไฟเลี้ยงจากตัวเอง (self powered device) คืออุปกรณ์ที่มีไฟเลี้ยงในตัวไม่ต้องอาศัยไฟเลี้ยงจากบัส

2.2.2 โครงสร้างการเชื่อมโยง (Topology)

โครงสร้างการเชื่อมต่อของยูเอสบีนั้นเป็นแบบสตาร์ (STAR) โดยฮับจะทำหน้าที่เป็นจุดศูนย์กลางเชื่อมต่อไปยังอุปกรณ์ต่าง ๆ ในแต่ละระดับชั้น สายเชื่อมต่อแต่ละเส้นเป็นการต่อแบบจุดต่อจุด(Point to Point) เนื่องจาก "พอร์ต ยูเอสบี มีรูปแบบการเชื่อมต่อเป็นแบบระบบบัส แต่มีโครงสร้างการเชื่อมต่อแบบสตาร์" ซึ่งอาจสร้างความสับสนในจุดนี้ได้จึงขอขยายความหมายและเปรียบเทียบให้เห็นความแตกต่างตรงจุดนี้จากคำว่า

โครงสร้างการเชื่อมต่อ : จะอธิบายถึงลักษณะการต่อสายของสัญญาณแต่ละตัวเข้าด้วยกันซึ่งก็คือลักษณะการต่อสายไฟระหว่างกันทำอย่างไร นั่นเอง

รูปแบบการเชื่อมต่อ : นั้นจะอธิบายถึง การเดินทาง ของสัญญาณข้อมูล ภายในสาย ที่เชื่อมต่อกันอยู่จากอุปกรณ์ตัวหนึ่งไปยังอุปกรณ์แต่ละตัวที่เชื่อมต่ออยู่ความหมายโดยรวมก็คือ พอร์ตยูเอสบี ต่อสายเข้าด้วยกันแบบสตาร์แต่รับส่งแบบบัส



รูปที่ 2.11 แสดงโครงข่ายการเชื่อมต่อยูเอสบี

2.2.3 การติดต่อระหว่างอุปกรณ์และโฮสต์

การถ่ายทอดสัญญาณ (Transfer type) ของบัสยูเอสบีนั้นแบ่งออกเป็น 4 ชนิดตามขนาดชนิดของข้อมูลและจังหวะการส่งข้อมูลดังนี้

- การถ่ายทอดสัญญาณแบบไอโซโครนัส (Isochronous transfer)
- การถ่ายทอดสัญญาณแบบบัลก์ (Bulk transfer)
- การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์ (Interrupt transfer)
- การถ่ายทอดสัญญาณควบคุม (Control transfer)

2.2.3.1 การถ่ายทอดแบบไอโซโครนัส

การส่งข้อมูลชนิดนี้มีการส่งหรือรับข้อมูลในทุก ๆ เฟรมข้อมูล (ทุก ๆ 1 มิลลิวินาที) ทำให้เกิดการรับส่งข้อมูลด้วยอัตราเร็วคงที่ แต่นั้นก็หมายความว่า การรับส่งข้อมูลแบบนี้จะต้องจองการรับส่งข้อมูลในแต่ละเฟรม 1 มิลลิวินาทีไว้ตลอดเวลา ประเภทงานที่ใช้การรับส่งข้อมูลชนิดนี้คืองาน ที่ต้องการอัตราเร็วการส่งข้อมูลคงที่ตลอดเวลา เช่น การส่ง ข้อมูลเสียงไปยังลำโพง ยูเอสบี เพราะว่างาน ลักษณะนี้ปลายทางจะต้องนำข้อมูลที่ได้รับแปลงกลับเป็นข้อมูลอะนาล็อก ซึ่งจะเกิดความผิดเพี้ยนทางความถี่ หากข้อมูลขาดช่วงไม่สม่ำเสมอก่อน เริ่มการทำงาน อุปกรณ์จะต้อง ทำการซิงโครไนซ์ การทำงานของตัวเองเข้ากับสัญญาณเริ่มต้น

ของแต่ละเฟรม ของ ยูเอสบีเพื่อให้อุปกรณ์สามารถรับหรือส่งข้อมูลในทุก ๆ 1 มิลลิวินาทีได้ทัน ทำให้เกิดการรับ-ส่งข้อมูลที่ถูกต้องการซิงโครไนซ์เข้ากับเฟรมนั้นมีอยู่ 3 วิธีดังนี้

อะซิงโครนัส (asynchronous) การซิงโครไนซ์วิธีนี้ สัญญาณ นาฬิกาของ อุปกรณ์ไม่จำเป็นต้องเข้าจังหวะสัญญาณ ข้อมูลของ ยูเอสบีสัญญาณ นาฬิกาที่ควบคุมการ ทำงานของอุปกรณ์จะเป็นอิสระไม่ขึ้นอยู่กับสัญญาณ SOF แต่อัตราการส่งข้อมูล จะต้องกำหนด ไว้ตายตัวที่ค่าใดค่าหนึ่ง กำหนดต่อนเริ่มต้นการเชื่อมต่อ ไม่สามารถเปลี่ยน แปลงความเร็วได้ ตัวอย่างของอุปกรณ์ที่ซิงโครไนซ์ด้วยวิธีนี้คือ ซีดีรอมในโหมดของการเล่นเพลงเพราะว่าซีดีเพลงมีอัตราการสุ่มข้อมูลคงที่เท่ากับ 44.1 kHz และนอกจากซีดีรอมแล้วอีกตัวหนึ่งที่เห็นได้ชัดเจนคือ ไมโครโฟนยูเอสบีที่มี อัตราการสุ่ม (Sampling Rate) เป็น อิสระไม่ขึ้นอยู่กับสัญญาณ SOF ของยูเอสบี

ซิงโครนัส (synchronous) วิธีนี้ใช้สัญญาณนาฬิกาที่ใช้ในอุปกรณ์ กับสัญญาณ SOF ของยูเอสบีจะต้องเข้าจังหวะกันโดยก่อนเริ่มทำงานอุปกรณ์จะต้องมีการปรับสัญญาณนาฬิกาของตัวเองให้เข้าจังหวะกับ SOF ตัวอย่างอุปกรณ์ที่ทำงานในลักษณะนี้คือ ไมโครโฟนยูเอสบี ที่มีอัตราการสุ่มข้อมูลเข้าจังหวะกับสัญญาณ SOF นั่นเอง อีกตัวอย่างหนึ่งก็คือ โมเด็ม ISDN ความเร็ว 64 กิโลบิตต่อวินาทีจะต้องเกิดการซิงโครไนซ์สัญญาณ SOF ของโฮสต์เข้ากับสัญญาณนาฬิกาของระบบ ISDN ก่อนจึงจะเกิดการถ่ายทอดข้อมูลที่ความเร็วคงที่ 64 กิโลบิตต่อวินาที ได้

อะแดปตีฟ (Adaptive) วิธีนี้เป็นวิธีที่อุปกรณ์มีความสามารถมากที่สุด เพราะอุปกรณ์สามารถปรับความเร็วในการถ่ายทอดข้อมูลได้ในช่วงที่กว้างมากไม่ถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่งสามารถเปลี่ยนความเร็วได้ในระหว่างการใช้งานสัญญาณนาฬิกาภายในระบบของอุปกรณ์ จะเข้าจังหวะกับสัญญาณข้อมูลที่ส่งในสาย (แตกต่างจากวิธีซิงโครนัสที่เข้าจังหวะกับสัญญาณ SOF) ตัวอย่างของอุปกรณ์ที่ทำงานลักษณะนี้คือ ซีดีรอมเล่นเพลงที่มีอุปกรณ์พิเศษในการเปลี่ยนแปลงอัตราการสุ่มข้อมูล (Sample Rate Converter: SRC) ซึ่งด้วยอุปกรณ์ตัวนี้จะทำให้ความเร็วในการสุ่มข้อมูลไม่จำเป็นต้องคงที่เท่ากับ 44.1 kHz แต่สามารถปรับเปลี่ยนไปตามขอบเขตการทำงานของตัว SRC

2.2.3.2 การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์

การสร้างข้อมูลชนิดนี้สร้างขึ้นเพื่อเลียนแบบการสร้างสัญญาณอินเตอร์รัปต์ ของอุปกรณ์ให้แก่ระบบ วิธีการเลียนแบบ อาศัยการวนอ่านจากตัวอุปกรณ์ต่าง ๆ ในระยะเวลาอย่างสม่ำเสมอหรือเรียกว่า โพลลิ่ง (Polling) โดยอัตราการวนอ่านข้อมูลต้องไม่ช้าหรือเร็วเกินไปเพราะจะทำให้เกิดการสูญเสียข้อมูลที่อ่านไม่ทันได้ในขณะเดียวกันจะต้องไม่เร็วเกินไปจนเกินเข้าไปในแบนด์วิทซ์ข้อมูลมากเกินความจำเป็น ถ้าหากอุปกรณ์ไม่มีข้อมูลที่ต้องการส่ง ก็จะส่งสัญญาณ No Acknowledge (NAK)

2.2.3.3 การถ่ายทอดสัญญาณแบบบัลก์

เป็นการส่งข้อมูลผ่านพอร์ต ยูเอสบีสำหรับงานที่ไม่ต้องการอัตราเร็วในการถ่ายทอดสัญญาณที่คงที่

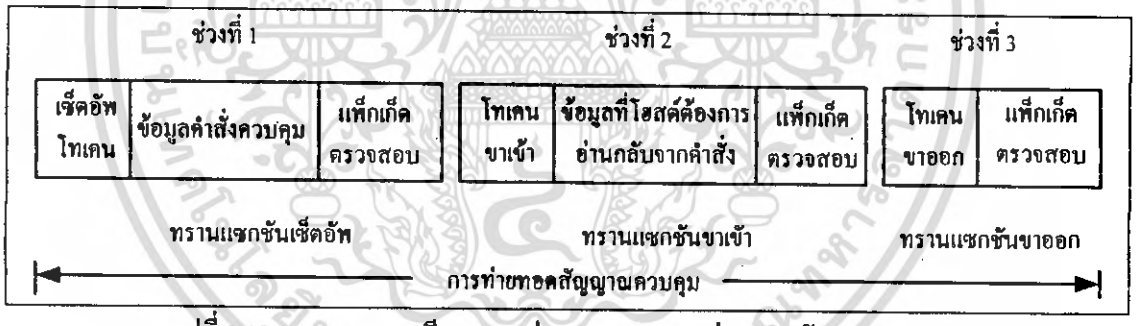
อุปกรณ์ที่เห็นได้ชัดเจนที่สุดสำหรับ การถ่ายทอด ในลักษณะนี้คือ เครื่องพิมพ์เพราะข้อมูลที่ผู้ใช้งานต้องการส่งไปยังเครื่องพิมพ์นั้นไม่ต้องการความเร็วในการส่งที่ สม่ำเสมอ การส่งช้าหรือเร็วก็จะส่งผลกระทบในด้านเวลาที่ต้องการใช้มากขึ้นเท่านั้น แต่ข้อมูลที่ได้ก็ยังไม่เสียหายแต่อย่างใด

2.2.3.4 การถ่ายทอดสัญญาณควบคุม

การรับส่งข้อมูลแบบนี้อาจถือว่ามีความสำคัญที่สุดก็ว่าได้ เพราะใช้สำหรับส่งคำสั่งควบคุมการทำงานทั้งหมด ของอุปกรณ์ทุก ๆ ตัว ตั้งแต่เริ่มแรก เมื่ออุปกรณ์ถูกเชื่อมต่อเข้ากับระบบ เกิดการอ่านข้อมูลดีสคริปเตอร์ต่าง ๆ เมื่อได้รับข้อมูลมาโฮสต์จะพิจารณาว่าสามารถรองรับได้หรือไม่ เมื่อพิจารณาว่าได้ ก็จะตั้งค่าการทำงานต่าง ๆ และเริ่มการทำงาน

การถ่ายทอดสัญญาณทั้ง 3 แบบที่ผ่านมาล้วนแล้วเป็นการรับส่งข้อมูลที่เกิดจากฟังก์ชันการทำงานของตัวอุปกรณ์ทั้งหมดแต่สำหรับการถ่ายทอดสัญญาณควบคุมนั้นเป็นการส่งข้อมูลควบคุมตัวอุปกรณ์ซึ่งความผิดพลาดในการส่งข้อมูลเป็นเรื่องที่ยอมรับไม่ได้ดังนั้นการส่งข้อมูลชนิดนี้จะถูกออกแบบมาพิเศษกว่าการส่งชนิดอื่น ๆ เพื่อป้องกันความผิดพลาดที่จะเกิดขึ้น

การถ่ายทอดสัญญาณควบคุมประกอบด้วย การส่งข้อมูล 3 ช่วง ช่วงที่หนึ่งคือ ช่วงเตรียมตัวตั้งค่า (Setup stage) ช่วงที่ 2 คือช่วงข้อมูล (data stage) และช่วงสุดท้ายคือ ช่วงตรวจสอบ (handshake stage)



รูปที่ 2.12 แสดงรายละเอียดของแต่ละสเตจของการถ่ายทอดสัญญาณควบคุม

2.2.4 โครงสร้างระดับล่างของการรับส่งข้อมูล

โครงสร้างระดับล่างสุดของ ยูเอสบีคือ จัดข้อมูลเป็นแพ็กเก็ต ซึ่งตัวแพ็กเก็ตแบ่งได้หลายชนิดเมื่อนำแพ็กเก็ตมาต่อเรียงกันก็จะได้ทรานแซกชันขึ้นมาดังนั้น ทรานแซกชัน จึงมีหลาย ชนิดขึ้นอยู่กับชนิด ของแพ็กเก็ตที่ประกอบขึ้นมา แต่ละทรานแซกชันประกอบขึ้นจากแพ็กเก็ตต่าง ๆ ซึ่งโดยส่วนมากประกอบขึ้นด้วย 3 เฟสหรือแพ็กเก็ต โดยมีรายละเอียดดังนี้

- **Token packet phase:** เป็นเฟสเริ่มต้นของทุกทรานแซกชันภายในจะบรรจุข้อมูลที่ บอกถึงชนิดของทรานแซกชันรวมไปถึงแอดเดรสของอุปกรณ์ที่ติดต่อด้วย
- **Data packet phase :** ทรานแซกชัน ส่วนใหญ่จะมีเฟส ข้อมูลรวมอยู่ด้วยเพื่อส่งหรือรับด้วยข้อมูลปริมาณมาก ๆ โดยข้อมูลที่สามารถส่งหรือรับได้มากสุดต่อครั้งเท่ากับ 1024 ไบต์
- **Handshake packet phase :** เนื่องจากทุก ๆ ครั้งที่เกิดการส่งข้อมูลยกเว้นการส่งแบบไอโซโครนัสจะมีการรับประกันความถูกต้องของการส่งดังนั้นเฟสนี้จึงเป็นข้อมูลที่ฝ่ายรับส่งกลับไปยังฝ่ายส่งเพื่อตรวจสอบความถูกต้องของการสื่อสาร

รูปที่ 2.13 ส่วนประกอบของแต่ละทรานแซกชัน

2.2.5 การจัดการกับอุปกรณ์บนบัส ยูเอสบี ที่มีความเร็วต่างกัน

อุปกรณ์ยูเอสบีแบ่งตามความเร็วของการถ่ายทอดข้อมูลได้ 2 ชนิดคือ อุปกรณ์ความเร็วเต็มที่จะถ่ายทอดข้อมูลที่อัตรา 12 Mb/s และอุปกรณ์ความเร็วต่ำถ่ายทอดข้อมูลที่อัตราเร็ว 1.5 Mb/s แต่เนื่องจากอุปกรณ์ทั้งสองประเภทนี้อยู่บนบัสเดียวกันทั้งหมด ดังนั้นพอร์ตของฮับที่ให้อุปกรณ์ยูเอสบีเข้ามาเชื่อมต่อจะต้องรองรับการทำงานได้ทั้ง 2 แบบ และด้วยความที่เป็นระบบบัส อุปกรณ์ทุก ๆ ตัวจะได้รับข้อมูลทุก ๆ แพ็กเก็ตที่ส่งเข้ามาไม่ว่าเป็นการส่งด้วยความเร็วสูงหรือต่ำดังนั้นจึงต้องจัดการจราจรระหว่างข้อมูลที่ส่งด้วยโดยตั้งเป็นข้อกำหนดว่า พอร์ตของอุปกรณ์ความเร็วต่ำจะต้องไม่เปิดทำงานจนกว่าจะได้รับสัญญาณ preamble จากโฮสต์โดยหลังจากได้รับสัญญาณ preamble แล้ว ฮับจะเปลี่ยนความเร็วในการถ่ายทอดความเร็วไปสู่โหมดความเร็วต่ำแล้วเปิดพอร์ตทำงานที่ความเร็วต่ำทำให้อุปกรณ์ที่เชื่อมต่อด้วยความเร็วต่ำได้รับข้อมูลในทางกลับกัน หลังจากได้รับสัญญาณ preamble แล้วอุปกรณ์ความเร็วเต็มที่จะทราบทันทีว่าข้อมูลที่ตามหลังมาอยู่ในโหมดความเร็วต่ำ ไม่ต้องอ่านและตีความข้อมูลเหล่านั้น

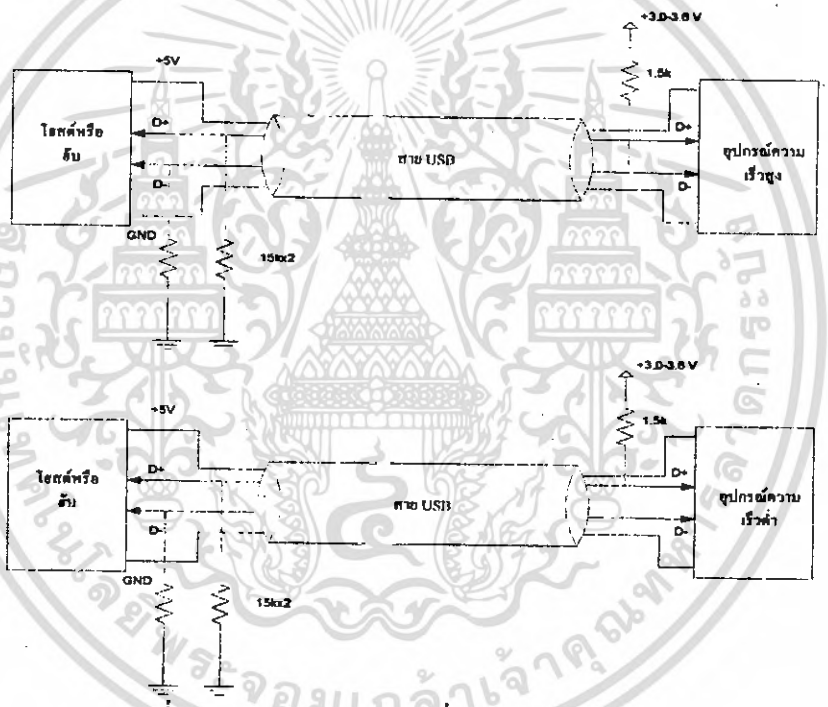
2.2.6 การตรวจสอบการเชื่อมต่อและความเร็วของอุปกรณ์

คุณสมบัติเด่นข้อหนึ่งของอุปกรณ์ ยูเอสบี คือ สามารถตรวจสอบการเชื่อมต่อของอุปกรณ์ตัวใหม่และตั้งค่าเริ่มต้นที่จำเป็นโดยอัตโนมัติ แต่เนื่องจากอุปกรณ์ ยูเอสบีแบ่งออกได้เป็นอุปกรณ์ความเร็วสูงและ ต่ำถ้าส่งข้อมูลผิดความเร็วก็ไม่สามารถสื่อสารกับอุปกรณ์นั้น ๆ ได้นั้นหมายความว่าจะต้องมีรูปแบบพิเศษที่ใช้แจ้งสถานการณ์เชื่อมต่อหรือปลดออกรวมทั้งระบุความเร็วของอุปกรณ์นั้น ๆ ได้ด้วย

การตรวจสอบการเชื่อมต่อของ ยูเอสบีนั้นจะตรวจสอบการเปลี่ยนแปลงของระดับสัญญาณในสายสัญญาณ D- และ D+ โดยสายข้อมูลทั้งสองที่ด้านฮับจะถูกต่อด้วยตัวต้านทาน $15k\Omega$ พูลดาวน์ไว้ทั้งสองเส้น ซึ่งส่งผลให้สายสัญญาณทั้งสองมีระดับแรงดันเป็น 0 V ในขณะไม่มีอุปกรณ์ใด ๆ ต่ออยู่

ที่ด้านอุปกรณ์ สำหรับอุปกรณ์ความเร็วต่ำสายสัญญาณ D- จะต่อตัวต้านทาน $1.5k\Omega$ พูลอัปกับแรงดัน 3.0 – 3.6 V ส่วนอุปกรณ์ความเร็วสูงสายสัญญาณที่ต่อพูลอัปคือ สายสัญญาณ D+ เมื่อมีอุปกรณ์ตัวใหม่ถูกต่อเข้ากับฮับจะเกิดการแบ่งแรงดันจากตัวต้านทานทั้งสอง ทำให้แรงดันของสายสัญญาณเพิ่มจาก 0V ขึ้นไปที่ 90% ของไฟเลี้ยงที่พูลอัป

คำนวณได้จาก $\dfrac{15x10^3}{(1.5+15)x10^3 xVcc}$



รูปที่ 2.14 การตรวจสอบการเชื่อมต่อของระบบ ยูเอสบี

การเปลี่ยนแรงดันนี้ทำให้ฮับรู้ว่ามีอุปกรณ์ตัวใหม่เข้ากับระบบ และทำให้รู้ว่าอุปกรณ์ถูกปลดออกจากระบบเช่นเดียวกัน โดยถ้าสัญญาณที่เกิดการเปลี่ยนแปลงแรงดันที่เป็นสัญญาณ D- หมายความว่าอุปกรณ์ที่นำมาต่อเป็นอุปกรณ์ความเร็วต่ำ แต่ถ้าสัญญาณ D+ เปลี่ยนแปลงหมายความว่าเป็นอุปกรณ์ความเร็วสูงนำเข้ามาต่อกับระบบ

2.2.7 การส่งสัญญาณในบัส ยูเอสบี

สัญญาณที่ปรากฏบนสายสัญญาณระหว่าง รูตฮับและอุปกรณ์จะส่งไปในแบบสัญญาณผลต่าง เพื่อลดการกระจายของสนามแม่เหล็กไฟฟ้า (EMI: Electromagnetic Interference) เนื่องจากตามธรรมชาติของสัญญาณไฟฟ้า เมื่อมีการเปลี่ยนแปลงของระดับสัญญาณด้วยความเร็วมาก ๆ (ก็คือการส่งข้อมูลที่ความเร็วมากๆ) จะทำให้เกิดการแพร่กระจายของสนามแม่เหล็กไฟฟ้าและสนามไฟฟ้าออกมารอบ ๆ สายส่งสัญญาณ ซึ่งอาจรบกวนการทำงานของอุปกรณ์ต่าง ๆ รอบข้างได้ด้วยการส่งสัญญาณแบบนี้จะทำให้การเปลี่ยนแปลงของสัญญาณในสายตัวนำเกิดขึ้นพร้อมกันและเกิดในลักษณะตรงข้าม ทำให้สนามแม่เหล็กไฟฟ้าเกิดการหักล้างกัน ไม่แพร่ออกมาภายนอก

2.2.7.1 กระบวนการกำหนดการทำงานของอุปกรณ์

เมื่อเริ่มการทำงานของยูเอสบีหรือเมื่อมีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้ามาในระบบตามลำดับขั้นตอนคร่าวๆ ในการทำงานจะเป็นดังนี้

- ฮับตรวจสอบพบว่ามีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้าสู่ระบบแล้วแจ้งผลกลับไปยังโฮสต์คอนโทรลเลอร์
- โฮสต์คอนโทรลเลอร์สั่งให้ฮับเปิดการทำงานของแหล่งจ่ายไฟโหมดประหยัดพลังงาน เพื่ออุปกรณ์ที่อาศัยพลังงานจากบัสสามารถทำงานได้
- โฮสต์คอนโทรลเลอร์สั่งให้ฮับรีเซ็ตพอร์ตที่อุปกรณ์ตัวใหม่เข้ามาเชื่อมต่อเพื่อให้อุปกรณ์ รีเซ็ตค่าแอดเดรสและเอนด์พอยต์ของตัวเองเป็นค่าเริ่มต้น (default)
- โฮสต์อ่านดีสคริปเตอร์ต่าง ๆ จากตัวอุปกรณ์และพิจารณาว่าทรัพยากรของระบบเพียงพอต่อความต้องการของตัวอุปกรณ์หรือไม่ ทรัพยากรในที่นี้คือ พลังงานไฟฟ้าและปริมาณข้อมูลที่จะส่งของตัวอุปกรณ์ หากพิจารณาแล้วว่าไม่สามารถทำงานได้ก็จะสั่งให้ฮับปิดการทำงานของพอร์ตนั้น
- เมื่อโฮสต์พิจารณาแล้วว่าสามารถให้บริการแก่อุปกรณ์ที่เข้ามาเชื่อมต่อได้ จะควบคุมให้แหล่งจ่ายไฟตามที่อุปกรณ์ต้องการ รวมไปถึงการตั้งค่าแอดเดรสและกำหนดค่าต่าง ๆ
- หลังจากตั้งค่าเรียบร้อย คอมพิวเตอร์ก็จะรู้จักกับอุปกรณ์ตัวใหม่และสามารถติดต่อกับอุปกรณ์ได้ทันทีโดยไม่ต้องปิดและเปิดคอมพิวเตอร์ใหม่

2.2.8 คำร้องขอของอุปกรณ์ ยูเอสบี (USB Device Request)

อุปกรณ์ ยูเอสบีทุกตัวจะต้องตอบสนองต่อคำร้องขอที่ถูกส่งมาจากโฮสต์บนดีฟอลต์คอนโทรลไปป์ของอุปกรณ์ คำร้องขอเหล่านี้จะถูกสร้างขึ้นและส่งไปโดยใช้การส่งถ่ายข้อมูลแบบคอนโทรล คำร้องขอและพารามิเตอร์ต่างๆ ของมันจะอยู่ในเซ็ตอัพแพ็กเก็ตซึ่งถูกส่งไปยังอุปกรณ์ ตามปกติเซ็ตอัพแพ็กเก็ตทุกชุดจะมีขนาด 8 ไบต์ ซึ่งความหมายและหน้าที่ของฟิลด์ต่างๆ ภายในแพ็กเก็ตสามารถอธิบายได้ดังนี้

bmRequest Type : ฟิลด์นี้เกิดจากการนำบิตข้อมูลหลายกลุ่มซึ่งมีความหมายของตัวเองมาเรียงต่อกัน หรือเรียกว่า บิตแมพ (Bitmap) สำหรับหน้าที่ของฟิลด์นี้คือเป็นตัวกำหนดทิศทางของการส่งถ่ายข้อมูลในดาต้าสเตจ ชนิดของคำร้องขอ และผู้รับ สถานะของบิตแสดงทิศทางจะไม่มีผลเมื่อฟิลด์ wLength มีค่าเป็น 0 ซึ่งเป็นการบ่งบอกว่าไม่มีดาต้าสเตจในการส่งถ่ายข้อมูลคอนโทรลนั้นๆ

bRequest : เป็นตัวกำหนดคำร้องขอ ซึ่งการเปลี่ยนแปลงบิต Type ในฟิลด์ bmRequest Type จะส่งผลกระทบต่อความหมายของฟิลด์นี้ ในข้อกำหนด ยูเอสบีจะมีการกำหนดค่าที่ใช้กับฟิลด์นี้เพียงฟิลด์เดียวเท่านั้น ในกรณีที่บิต Type ถูกรีเซ็ตเป็น 0 จะหมายถึงคำร้องขอนั้นเป็นคำร้องขอมาตรฐาน

wValue : ค่าที่อยู่ในฟิลด์นี้จะมีการเปลี่ยนแปลงสอดคล้องกับคำร้องขอ หน้าที่ของมันคือใช้ในการผ่านค่าพารามิเตอร์ของคำร้องขอบางตัวไปยังอุปกรณ์

wIndex : อยู่ในฟิลด์นี้จะมีการเปลี่ยนแปลงสอดคล้องกับคำร้องขอ หน้าที่ของมันคือใช้ในการผ่านค่าพารามิเตอร์ไปยังอุปกรณ์ของคำร้องขอบางตัว ตามปกติ wIndex จะใช้ในการร้องขอที่ส่งไปยังเอนด์พอยต์หรืออินเตอร์เฟสซึ่งมีรูปแบบดังรูป

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
สงวนไว้ใช้งาน(รีเซ็ตเป็น 0)								ทิศทาง	สงวนไว้ใช้งาน (รีเซ็ตเป็น 0)			หมายเลขเอนด์พอยต์			

ตารางที่ 2.9 รูปแบบของ wIndex เมื่อใช้กำหนดเอนด์พอยต์

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
สงวนไว้ใช้งาน(รีเซ็ตเป็น 0)								หมายเลขอินเตอร์เฟส							

ตารางที่ 2.10 รูปแบบของ wIndex เมื่อใช้กำหนดอินเตอร์เฟส

wLength : ใช้สำหรับระบุความยาวของข้อมูลที่ส่งถ่ายในดาต้าเฟส ถ้าฟิลด์นี้มีค่าเป็น 0 จะหมายถึงการส่งถ่ายข้อมูลคอนโทรลนั้นไม่มีข้อมูลในดาต้าเฟส

ออฟเซ็ต	ฟิลด์	ขนาด	ค่า	คำอธิบาย
0	bmRequestType	1	บิตแมพ	D7 ทิศทางการถ่ายโอนข้อมูลของคาต้าเฟส 0 = โฮสต์ไปยังอุปกรณ์ 1 = อุปกรณ์ไปยังโฮสต์ D6...5 ชนิดของคำร้องขอ (Type) 0 = คำร้องขอมาตรฐาน 1 = คำร้องขอของคลาส 2 = คำร้องขอของผู้ผลิต 3 = สงวนไว้ใช้งาน D4...0 เกี่ยวกับผู้รับ 0 = อุปกรณ์ 1 = อินเตอร์เฟส 2 = เอนด์พอยต์ 3 = อื่นๆ 4...31= สงวนไว้ใช้งาน
1	bRequest	1	ค่า	คำร้องขอ
2	wValue	2	ค่า	ใช้ในการผ่านค่าพารามิเตอร์ไปยังอุปกรณ์ เปลี่ยนแปลงไปตามชนิดของคำร้องขอ
4	wIndex	2	ตัวเลขดัชนีหรือออฟเซ็ต	ใช้ในการผ่านค่าตัวเลขดัชนีหรือค่าออฟเซ็ตซึ่ง เปลี่ยนแปลงตามชนิดของคำร้องขอ
6	wLength	2	จำนวน	จำนวนไบต์ที่ต้องการถ่ายโอนในคาต้าเฟส

ตารางที่ 2.11 เซ็ตอัพแพ็กเก็ต

2.2.10 คำร้องขอมาตรฐาน (Standard Request)

คำร้องขอมาตรฐานเป็นสิ่งที่ใช้กับอุปกรณ์ ยูเอสบีได้ทุกตัว คำร้องขอมาตรฐานเหล่านี้สามารถส่งไปยังผู้รับที่แตกต่างกัน เช่นคำร้องขอ Get_Status สามารถส่งไปยังอุปกรณ์อินเตอร์เฟส หรือ เอนด์พอยต์ก็ได้ โดยเมื่อมีการส่งไปยังอุปกรณ์ค่าที่ได้รับกลับมาจะเป็นแฟล็กซึ่งบอกสถานะของรีโมตเวคอัพและลักษณะการใช้กำลังงานของอุปกรณ์ (ใช้กำลังงานจากตัวเองหรือจากระบบบัส) ถ้าคำร้องขอเดียวกันนี้ถูกส่งไปยัง

อินเตอร์เฟสค่าที่ส่งกลับออกมาจะเป็น 0 เสมอ และถ้าคำร้องขอนี้ถูกส่งไปยังเอนด์พอยต์ค่าที่ส่งกลับมาจะเป็นสถานะของแฟล็ก หยุดชั่วคราวที่เอนด์พอยต์นั้นๆในข้อกำหนดยูเอสบี

2.2.11 คำร้องขอมาตรฐานสำหรับอุปกรณ์

Get Status : เป็นคำร้องขอที่ใช้สำหรับอ่านค่าสถานะของอุปกรณ์ (ในกรณีที่คำร้องขอถูกส่งไปยังอุปกรณ์) ซึ่งอุปกรณ์จะตอบสนองต่อคำร้องขอนี้โดยการส่งค่ากลับออกมาขนาด 2 ไบต์ในคาต้าสเตจมีรูปแบบดังนี้

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
สงวนไว้ใช้งาน(รีเซ็ตเป็น 0)														รีโมตเวคอัพ	ใช้กำลังงานจากตัวเอง

ตารางที่ 2.12 รูปแบบของข้อมูลที่อุปกรณ์ตอบสนองต่อคำร้องขอของ Get Status

ถ้า D0 ถูกเซ็ตเป็น 1 หมายถึง อุปกรณ์ใช้กำลังงานจากตัวเอง และถ้าเป็น 0 จะหมายถึงอุปกรณ์ใช้กำลังงานจากระบบบัส บิตรีโมตเวคอัพ (D1) ใช้สำหรับแสดงว่าในขณะนั้นอุปกรณ์ได้เปิดการทำงานในลักษณะรีโมตเวคอัพไว้หรือไม่ ค่าดีฟอลต์หรือค่าเริ่มต้นของบิตนี้คือ 0 นั้นคืออุปกรณ์ไม่สามารถทำรีโมตเวคอัพได้ ถ้า D1 ถูกเซ็ตเป็น 1 หมายถึงอุปกรณ์สามารถทำรีโมตเวคอัพได้ ซึ่งอุปกรณ์สามารถกระตุ้นโฮสต์ในขณะที่โฮสต์อยู่ในโหมดซัซเพนด์ได้ การเปลี่ยนแปลงบิตรีโมตเวคอัพสามารถทำได้โดยใช้คำร้องขอ Set_Feature และ Clear_Feature โดยใช้ตัวเลือกฟีเจอร์ (ความสามารถพิเศษของอุปกรณ์) DEVICE_REMOTE_WAKEUP (01h) ในกรณีที่อุปกรณ์ถูกรีเซ็ตบิตนี้จะถูกรีเซ็ตให้มีค่าเป็น 0

Clear Feature : ใช้ในการเคลียร์หรือปิดการทำงานของฟีเจอร์ ซึ่งตัวเลือกฟีเจอร์ที่ใช้สนับสนุนอุปกรณ์จะมีอยู่ด้วยกัน 2 ตัวคือ DEVICE_REMOTE_WAKEUP และ TEST_MODE แต่ตัวเลือกฟีเจอร์ TEST_MODE ซึ่งสนับสนุนเฉพาะในอุปกรณ์ที่เป็นไฮสปีดจะไม่สามารถปิดการทำงานได้ด้วยการใช้คำร้องขอ Clear_Feature แต่ต้องทำการปลดและจ่ายไฟให้อุปกรณ์ใหม่เมื่อต้องการออกจากโหมดทดสอบ

Set Feature : ใช้ในการเปิดการทำงานของฟีเจอร์ซึ่งตัวเลือกฟีเจอร์ที่ใช้สำหรับอุปกรณ์จะมีอยู่ด้วยกัน 2 ตัวคือ DEVICE_REMOTE_WAKEUP และ TEST_MODE สำหรับฟีเจอร์ TEST_MODE ได้ถูกกำหนดให้ใช้สำหรับกรณีที่ผู้ใช้เป็นอุปกรณ์เท่านั้น (bmRequestType = 0) โดยที่ไบต์บนของฟิลด์ wIndex จะรีเซ็ตให้เป็น 0 ส่วนไบต์ล่างจะเป็นโค้ดสำหรับเลือกโหมดการทดสอบ การเซ็ตให้เป็น TEST_MODE นี้จะทำให้อุปกรณ์เข้าสู่โหมดการทดสอบและอุปกรณ์จะสามารถออกจากโหมดการทดสอบได้ด้วยการจ่ายไฟให้กับอุปกรณ์ใหม่อีกครั้ง

Set Address : คำร้องขอนี้จะใช้ในขณะทำการอิวนิเมอเรทเพื่อกำหนดแอดเดรสให้กับอุปกรณ์ยูเอสบี แต่ละตัวโดยที่ไม่ซ้ำกัน แอดเดรสจะถูกระบุมาใน wValue และสามารถมีค่าสูงสุดได้เพียง 127 เท่านั้น คำร้องขอนี้ใช้ได้เฉพาะกับอุปกรณ์ตัวนั้นและการเซ็ตแอดเดรสจะกระทำหลังจากสเตตัสสเตจเสร็จสิ้นลง ซึ่งต่างจากคำร้องขออื่นๆ ที่ต้องทำให้เสร็จสิ้นลงก่อนถึงสเตตัสสเตจ

Get_Descriptor : คำร้องขอนี้ใช้ในกรณีที่โฮสต์ต้องการร้องขอดีสคริปเตอร์ โดยในไบต์สูงของฟิลด์ wValue จะเป็นตัวกำหนดชนิดของดีสคริปเตอร์และตัวชี้ดีสคริปเตอร์จะกำหนดไว้ในไบต์ต่ำ สำหรับตัวชี้ดีสคริปเตอร์นั้นจะใช้ในการเลือกดีสคริปเตอร์ในกรณีที่มีดีสคริปเตอร์ชนิดเดียวกันหลายตัวอยู่ภายในอุปกรณ์ (Configuration Descriptor , String Descriptor) ในการร้องขอคอนฟิกิวเรชันดีสคริปเตอร์แต่ละครั้งนั้นอุปกรณ์จะส่งคอนฟิกิวเรชันดีสคริปเตอร์รวมถึงอินเตอร์เฟสดีสคริปเตอร์และเอนด์พอยต์ดีสคริปเตอร์ทั้งหมดที่อยู่ในคอนฟิกิวเรชันนั้นกลับออกมา

ฟิลด์ **wIndex** ใช้สำหรับระบุหมายเลขภาษาของ String Descriptor แต่สำหรับดีสคริปเตอร์ชุดอื่นๆ จะรีเซ็ตเป็น 0

ฟิลด์ **wLength** ใช้ระบุจำนวนไบต์ของดีสคริปเตอร์ที่จะส่งกลับ คำร้องขอนี้จะสนับสนุนดีสคริปเตอร์ 3 ชนิดนี้ได้แก่ Device Descriptor (รวมถึง Device Qualifier Descriptor) Configuration Descriptor (Other Speed Configuration Descriptor) และ String Descriptor ส่วนดีสคริปเตอร์ที่เหลือ (Interface Descriptor และ Endpoint Descriptor) จะไม่สามารถเข้าถึงได้โดยตรงจากคำร้องขอ Get_Descriptor

Set Descriptor : คำร้องขอนี้ใช้ในกรณีที่โฮสต์ต้องการเพิ่มดีสคริปเตอร์ตัวใหม่เข้าไป หรือแก้ไขดีสคริปเตอร์ที่มีอยู่ก่อนหน้านี้ในไบต์สูงของฟิลด์ wValue จะเป็นตัวกำหนดชนิดของดีสคริปเตอร์และตัวชี้ดีสคริปเตอร์จะกำหนดไว้ในไบต์ต่ำ สำหรับตัวชี้ดีสคริปเตอร์จะใช้ในการเลือกดีสคริปเตอร์ในกรณีที่มีดีสคริปเตอร์ชนิดเดียวกันหลายตัวอยู่ภายในอุปกรณ์ (Configuration Descriptor , String Descriptor)

ฟิลด์ **wIndex** ใช้สำหรับระบุภาษาของ String Descriptor แต่สำหรับดีสคริปเตอร์ชุดอื่นๆ จะรีเซ็ตเป็น 0

ฟิลด์ **wLength** ใช้ระบุจำนวนไบต์ของดีสคริปเตอร์ที่จะส่งจากโฮสต์ไปยังตัวอุปกรณ์คำร้องขอนี้อนุญาตให้ใช้กับดีสคริปเตอร์ 3 ชนิดได้แก่ Device Descriptor (Device Qualifier Descriptor) Configuration Descriptor (Other Speed Configuration Descriptor) และ String Descriptor ส่วนดีสคริปเตอร์ที่เหลือ (Interface Descriptor และ Endpoint Descriptor) จะไม่สามารถเข้าถึงได้โดยตรงจากคำร้องขอ Set_Descriptor

Get configuration : ใช้สำหรับร้องขอคอนฟิกิวเรชันที่ใช้อยู่ในปัจจุบันของอุปกรณ์ โดยจะมีไบต์ข้อมูลส่งกลับมาในระหว่างดาต้าสเตจเพื่อเป็นการบอกสถานะของอุปกรณ์ ถ้าได้ค่า 0 ออกมาแสดงว่าไม่สามารถตั้งค่าให้กับอุปกรณ์ได้ และถ้าได้ค่าที่ไม่เป็น 0 ส่งกลับออกมาแสดงว่าอุปกรณ์ถูกตั้งค่าเรียบร้อยแล้ว

Set Configuration : ใช้สำหรับเซ็ตคอนฟิกิวเรชันของอุปกรณ์ โดยใช้ไบต์ล่างของ wValue เป็นตัวระบุคอนฟิกิวเรชันที่ต้องการ

2.2.12 คำร้องขอมาตรฐานของอินเตอร์เฟส.

Get Status : เป็นคำร้องขอที่ใช้สำหรับอ่านสถานะของอินเตอร์เฟส (ในกรณีที่คำร้องขอถูกส่งไปยังอินเตอร์เฟส) ซึ่งข้อมูลที่ถูกส่งกลับมาจากการร้องขอไปยังอินเตอร์เฟสจะมีขนาด 2 ไบต์ซึ่งทั้ง 2 ไบต์จะมีค่าเป็น 00h

Clear Feature และ Set Feature : ใช้สำหรับเซ็ตหรือเคลียร์ฟีเจอร์สำหรับอินเตอร์เฟส แต่ในปัจจุบันข้อกำหนด ยูเอสบี ยังไม่มีการกำหนดฟีเจอร์ที่เป็นของอินเตอร์เฟสเอาไว้

Get Interface : คำร้องขอนี้ใช้สำหรับค่ากลุ่มของอินเตอร์เฟส (Alternate Setting) ที่อุปกรณ์ใช้อยู่ในขณะนั้น

Set Interface : คำร้องขอนี้จะถูกใช้ในกรณีที่โฮสต์ต้องการจะเปลี่ยนทางเลือกของอินเตอร์เฟส ไปใช้อินเตอร์เฟสอีกกลุ่มหนึ่ง

2.2.13 คำร้องขอมาตรฐานสำหรับเอนด์พอยต์

Get Status : ใช้สำหรับอ่านค่าสถานะของเอนด์พอยต์ (HALT/STALL) ซึ่งค่าที่รับเข้ามา

Clear Feature/ Set Feature : ใช้สำหรับเคลียร์หรือเซ็ตฟีเจอร์ของเอนด์พอยต์ ในข้อกำหนด ยูเอสบี ได้กำหนดตัวเลือกฟีเจอร์ของเอนด์พอยต์ไว้เพียงตัวเดียวเท่านั้น นั่นคือ ENDPOINT_HALT (00h) สำหรับบัลก์เอนด์พอยต์และอินเตอร์รัปต์เอนด์พอยต์จะต้องสนับสนุนเงื่อนไขการเกิดการหยุดชั่วคราว (HALT) ซึ่งสาเหตุของการเกิดหยุดชั่วคราวมี 2 อย่าง สาเหตุแรกเกิดจากปัญหาทางด้านการสื่อสาร เช่นอุปกรณ์ไม่ได้รับแฮนด์เช็กแพ็กเก็ต ส่วนกรณีที่สองเกิดจากการที่อุปกรณ์ได้รับคำร้องขอ Set_Feature ให้ทำการหยุดชั่วคราวเอนด์พอยต์ซึ่งการส่งคำร้อง ขอ Clear_Feature ไปยัง เอนด์พอยต์ที่ ถูกทำให้หยุด ชั่วคราวจะเป็นการนำ เอนด์พอยต์ออกจากสภาวะหยุดชั่วคราวซึ่งเกิดเนื่องจากการใช้คำร้องขอ Set_Feature ได้

Synch Frame : ใช้สำหรับเซ็ตและรายงานการเข้าจังหวะเฟรมของเอนด์พอยต์ ในการส่งถ่ายข้อมูลแบบไอโซโครนัสนั้น ดีไวซ์เอนด์พอยต์อาจจะส่งคำร้องขอคาด้าแพ็กเก็ตขนาดต่างๆกันหลายชุด เช่น เอนด์พอยต์อาจจะส่งลำดับของไบต์ข้อมูลขนาด 8, 16, 32, 64 ไบต์ออกมาซึ่งคำร้องขอ Synch_Frame นี้จะช่วยทำให้โฮสต์และเอนด์พอยต์ตกลงกันได้ว่าเฟรมไหนเป็นเฟรมเริ่มต้นของลำดับไบต์ข้อมูล

2.2.14 ยูเอสบีดีสคริปเตอร์ (USB DESCRIPTOR)

ดีสคริปเตอร์เป็น โครงสร้างข้อมูลที่อธิบายถึงความสามารถ และวิธีการใช้งานของอุปกรณ์ ยูเอสบี โดยที่อุปกรณ์ ยูเอสบี ทุกตัวจะมีลำดับชั้นของดีสคริปเตอร์ซึ่งเป็นตัวที่ใช้อธิบายข้อมูลให้แก่โฮสต์ เช่น อุปกรณ์ตัวนี้คืออุปกรณ์อะไร ใครคือผู้ผลิต เวอร์ชันของข้อกำหนด ยูเอสบี ที่อุปกรณ์ตัวนั้นสนับสนุน การตั้งค่าต่างๆสามารถตั้งได้กี่ทาง จำนวนและชนิดของเอนด์พอยต์ที่ใช้เป็นต้น

ดีสคริปเตอร์ที่ใช้ในข้อกำหนด ยูเอสบี 2.0 ได้แก่

- ดีไวซ์ดีสคริปเตอร์ (Device Descriptor)
- ดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์ (Device Qualifier Descriptor)
- คอนฟิกิวเรชันดีสคริปเตอร์ (Configuration Descriptor)
- อัทเธอร์สปีดคอนฟิกิวเรชันดีสคริปเตอร์ (Other Speed Configuration Descriptor)
- อินเตอร์เฟสดีสคริปเตอร์ (Interface Descriptor)
- เอนด์พอยต์ดีสคริปเตอร์ (Endpoint Descriptor)
- สตริงดีสคริปเตอร์ (String Descriptor)

รูปแบบของยูเอสบี ดีสคริปเตอร์ (USB Descriptor)

ดีสคริปเตอร์ทุกชนิดจะสร้างขึ้นมาจากรูปแบบเดียวกัน โดยไบต์แรกเป็นตัวที่ใช้ระบุความยาวของดีสคริปเตอร์ และไบต์ที่สองเป็นตัวที่ใช้บอกชนิดของดีสคริปเตอร์ ถ้าความยาวของดีสคริปเตอร์น้อยกว่าที่กำหนดไว้ในข้อกำหนด ยูเอสบี โฮสต์จะไม่สนใจดีสคริปเตอร์ตัวนั้น แต่ถ้าความยาวของดีสคริปเตอร์มีค่ามากกว่าที่ต้องการ โฮสต์จะไม่สนใจในส่วนที่เกินมา และมองหาดีสคริปเตอร์ตัวถัดไปที่อยู่ต่อจากจุดสุดท้ายของความยาวจริงของดีสคริปเตอร์ตัวนั้น

ออฟเซ็ต	ฟิลด์	ขนาด	ค่า	คำอธิบาย
0	bLength	1	ตัวเลข	ขนาดของดีสคริปเตอร์ (ไบต์)
1	bDescriptorType	1	ค่าคงที่	ชนิดของดีสคริปเตอร์
2	bcdUSB	2	BCD	จุดเริ่มต้นของพารามิเตอร์สำหรับดีสคริปเตอร์
...	...	...	...	...

ตารางที่ 2.13 รูปแบบของ ยูเอสบี ดีสคริปเตอร์

2.2.15 ดีไวซ์ดีสคริปเตอร์ (Device Descriptor)

ดีไวซ์ดีสคริปเตอร์เป็นตัวที่ใช้อธิบายข้อมูลพื้นฐานของอุปกรณ์ซึ่งโฮสต์จะทำการอ่านดีสคริปเตอร์ชุดนี้จากอุปกรณ์ที่ต่ออยู่ในระบบเป็นชุดแรก ดีไวซ์ดีสคริปเตอร์ประกอบด้วยฟิลด์ทั้งหมด 14 ฟิลด์ ซึ่งเป็นข้อมูลเกี่ยวกับตัวดีสคริปเตอร์เอง ตัวอุปกรณ์ที่ใช้ดีสคริปเตอร์ชุดนี้อธิบายคอนฟิกิวเรชัน และคลาสของอุปกรณ์ ซึ่งรายละเอียดของฟิลด์แต่ละฟิลด์สามารถอธิบายโดยแบ่งตามกลุ่มหน้าที่ของมันได้ดังนี้

กลุ่มที่ใช้อธิบายตัวดีสคริปเตอร์

bLength : ความยาวเป็นไบต์ของดีสคริปเตอร์

bDescriptorType : ค่าคงที่ซึ่งบอกชนิดว่าเป็นดีไวซ์ดีสคริปเตอร์

กลุ่มที่ใช้อธิบายตัวอุปกรณ์

bcdUSB : เป็นตัวที่ใช้แสดงเวอร์ชันสูงสุดของ ยูเอสบี ที่อุปกรณ์ตัวนั้นสนับสนุนค่านี้จะเป็นรหัส บีซีดี ในรูปแบบ JJMNh โดยที่ JJ เป็นตัวเลขแสดงเวอร์ชันหลัก (M) คือตัวเลขแสดงเวอร์ชันรอง และ N คือตัวเลขย่อยของเวอร์ชันรอง เช่น ยูเอสบี 2.0 สามารถเขียนได้เป็น 0200h, ยูเอสบี 1.1 สามารถเขียนได้เป็น010h และ ยูเอสบี 1.0 สามารถเขียนได้เป็น 0100h แต่ถ้าหากมีเวอร์ชัน 2.1.3 จะสามารถเขียนได้เป็น 0213h

idVendor : ผู้ที่เป็นสมาชิกของกลุ่มผู้สร้างมาตรฐาน ยูเอสบี หรือผู้ที่จ่ายค่าธรรมเนียมในการดำเนินการรายอื่นๆจะได้รับสิทธิ์ให้ใช้หมายเลขผู้ผลิตหรือเวนเดอร์ไอดี ที่ไม่ซ้ำกับผู้อื่นซึ่งดีไวซ์ดีสคริปเตอร์ของอุปกรณ์ ยูเอสบี ที่ใช้ในการค้าจะต้องมีเวนเดอร์ไอดีนี้ ส่วนทางด้านโฮสต์จะมีไฟล์ INF ซึ่งบรรจุค่าดังกล่าวอยู่และระบบปฏิบัติการวินโดวส์จะใช้ค่าหมายเลขนี้ช่วยในการตัดสินใจเลือกโหมดดีไวซ์ไดร์ฟเวอร์ที่เหมาะสมกับอุปกรณ์ตัวนั้น

idProduct : ผู้ผลิตอุปกรณ์จะระบุหมายเลขผลิตภัณฑ์หรือโพรดักซ์ไอดี เพื่อใช้สำหรับจำแนกผลิตภัณฑ์แต่ละตัว ค่านี้จะมีอยู่ที่ดีไวซ์ดีสคริปเตอร์และไฟล์ ไอเอ็นเอฟ ของอุปกรณ์ซึ่งอยู่บนโฮสต์ด้วย และวินโดวส์จะใช้ค่านี้ช่วยในการตัดสินใจในการเลือกโหลดดีไวซ์ไดร์ฟเวอร์ตัวที่เหมาะสมด้วย ค่าโปรดักซ์ไอดีจะใช้งานร่วมกับเวนเดอร์ไอดีซึ่งผู้ผลิตต่างๆ สามารถกำหนดค่าโปรดักซ์ไอดีของตนเองได้ ดังนั้นผู้ผลิตต่างๆที่มีเวนเดอร์ไอดีของตนเองจึงสามารถใช้ค่าโปรดักซ์ไอดีเหมือนกันได้

iProduct : ตัวเลขดัชนีซึ่งชี้ไปยังสตริงซึ่งอธิบายถึงผลิตภัณฑ์ ซึ่งถ้าไม่ได้ใช้งานจะถูกกำหนดให้เป็น 0

iSerialNumber : ตัวเลขดัชนีซึ่งชี้ไปยังสตริงซึ่งเก็บค่าซีเรียลนัมเบอร์ของอุปกรณ์เอาไว้ ซึ่งถ้าไม่ได้ใช้งานค่านี้จะถูกกำหนดให้เป็น 0 ซีเรียลนัมเบอร์เป็นค่าที่มีประโยชน์มาก ในกรณีที่ผู้ใช้อาจจะมีอุปกรณ์ที่เหมือนกันอยู่มากกว่าหนึ่งตัวในระบบบัส และโฮสต์ต้องการติดตามการทำงานของอุปกรณ์แต่ละตัว นอกจากนี้ซีเรียลนัมเบอร์ยังทำให้โฮสต์สามารถหาได้ว่าอุปกรณ์ทุกตัวที่เหมือนกันซึ่งมีค่าเวนเดอร์ไอดีและโปรดักซ์ไอดีเหมือนกันเหล่านี้ ตัวไหนเป็นอุปกรณ์ที่ติดตั้งเข้าไปใหม่ในระบบและตัวไหนติดตั้งอยู่ก่อนหน้านี้แล้ว

กลุ่มที่ใช้อธิบายคอนฟิกิวเรชัน

บีนัมเบอร์คอนฟิกิวเรชัน จำนวนของคอนฟิกิวเรชันที่อุปกรณ์สนับสนุน

บีแมกแพ็กเก็ตไซด์0 ค่าขนาดสูงสุดของแพ็กเก็ตที่ใช้สำหรับเอนด์พอยด์ 0 โฮสต์จะใช้ข้อมูลนี้กับคำร้องขอที่ตามมา อุปกรณ์โลว์สปีดจะมีค่าเท่ากับ 8 ส่วนอุปกรณ์ฟูลสปีดสามารถมีค่าได้เท่ากับ 8, 16, 32 หรือ 64 และสำหรับอุปกรณ์ไฮสปีดจะต้องมีค่าเท่ากับ 64

กลุ่มที่ใช้อธิบายคลาส

บีดีไวซ์คลาส ค่านี้จะเป็นชื่อของคลาสที่อุปกรณ์ถูกจัดให้อยู่ ค่าตั้งแต่ 1 ถึง FEh จะถูกสงวนไว้สำหรับคลาสที่ ยูเอสบี กำหนดขึ้นเช่น คลาสสำหรับฮับ พริ้นเตอร์ และ อุปกรณ์ทางการสื่อสารต่างๆ ส่วนค่า FFh หมายถึงคลาสที่ถูกกำหนดมาจากผู้ผลิตเอง อุปกรณ์บางตัวจะระบุคลาสที่มันอยู่ไว้ในอินเตอร์เฟสดีสคริปเตอร์ซึ่งในกรณีนี้ฟิลด์บีดีไวซ์คลาส ในดีไวซ์ดีสคริปเตอร์จะมีค่าเป็น 0 อุปกรณ์ ยูเอสบี ทุกตัวไม่จำเป็นต้องจัดอยู่ในคลาสใดคลาสหนึ่ง

บีดีไวซ์ซับคลาส ฟิลด์นี้ใช้เพื่อกำหนดคลาสย่อย หรือซับคลาส ให้กับอุปกรณ์ที่จัดอยู่ในคลาสใดคลาสหนึ่ง ถ้าฟิลด์บีดีไวซ์ซับคลาส เป็น 0 ซับคลาสต้องมีค่าเป็น 0 ด้วย ถ้าบีดีไวซ์ซับคลาส มีค่าอยู่ระหว่าง 1 และ FEh ซับคลาสจะต้องมีค่าตามที่กำหนดไว้ในข้อกำหนด ยูเอสบี และถ้าบีดีไวซ์ซับคลาส มีค่าเท่ากับ (FFh) ซับคลาสจถูกกำหนดจากผู้ผลิต

บีดีไวซ์โปรโตคอล ฟิลด์นี้ใช้ระบุโปรโตคอลซึ่งถูกกำหนดโดยฟิลด์คลาสหรือซับคลาสถ้าบีดีไวซ์คลาส มีค่าอยู่ระหว่าง 1 ถึง FEh โปรโตคอลจะต้องเป็นไปตามที่กำหนดไว้ในข้อกำหนด ยูเอสบี

ออฟเซ็ต	ฟิลด์	ขนาด	ค่า	คำอธิบาย
0	bLength	1	ตัวเลข	ขนาดเป็นไบต์ของดีสคริปเตอร์ (12 ไบต์)
1	bDescriptorType	1	ค่าคงที่	บอกชนิดว่าเป็นดีไวซ์ดีสคริปเตอร์ (01h)
2	bcdUSB	2	BCD	เวอร์ชันของข้อกำหนดที่อุปกรณ์ยอมให้ใช้งานได้
4	bDviceClass	1	คลาส	รหัสคลาส (ถูกกำหนดโดยกลุ่มผู้สร้างมาตรฐาน ยูเอสบี) - ถ้ามีค่าเป็น 0 : อินเตอร์เฟสแต่ละตัวภายในคอนฟิกิวเรชันจะระบุรหัสคลาสของมันเองและอินเตอร์เฟสต่างๆจะทำงานอย่างเป็นอิสระต่อกัน - ถ้ามีค่าเป็น (FFH) : รหัสคลาสจะระบุโดยผู้ผลิต - ถ้าเป็นค่าอยู่ระหว่าง 1 และ (FEH) : อุปกรณ์จะรองรับข้อกำหนดคลาสที่แตกต่างกันบนอินเตอร์เฟสที่แตกต่างกันและอินเตอร์เฟสอาจทำงานไม่เป็นอิสระต่อกัน
5	bDeviceSubClass	1	ซับคลาส	ซับคลาส (ถูกกำหนดโดยกลุ่มผู้สร้างมาตรฐาน ยูเอสบี)
6	bDeviceProtocol	1	โปรโตคอล	โปรโตคอล (ถูกกำหนดโดยกลุ่มผู้สร้างมาตรฐานยูเอสบี)
7	bMaxPacketSize(0)	1	ตัวเลข	ขนาดสูงสุดของแพ็กเก็ตสำหรับเอนด์พอยต์ศูนย์ (ค่าที่เป็นไปได้คือ 8,16,32 64)
8	idVendor	2	ID	เลขประจำตัวผู้ผลิต (ถูกกำหนดโดยกลุ่มผู้สร้างมาตรฐาน ยูเอสบี)
10	idProduct	2	ID	เลขประจำตัวผู้ผลิต (ถูกกำหนดโดยกลุ่มผู้สร้างมาตรฐาน ยูเอสบี)
12	bcdDevice	2	BCD	เวอร์ชันของอุปกรณ์
14	iManufacturer	1	ตัวเลขดัชนี	ตัวเลขดัชนีของสตริงดีสคริปเตอร์แสดงชื่อผู้ผลิต
15	iProduct	1	ตัวเลขดัชนี	ตัวเลขดัชนีของสตริงดีสคริปเตอร์แสดงชื่อผลิตภัณฑ์
16	iSerialNumber	1	ตัวเลขดัชนี	ตัวเลขดัชนีของสตริงดีสคริปเตอร์แสดงรุ่นของผลิตภัณฑ์
17	bNumConfigurations	1	เลขจำนวนเต็ม	จำนวนของคอนฟิกิวเรชัน

ตารางที่ 2.14 ดีไวซ์ดีสคริปเตอร์

2.2.16 ดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์ (Device Qualifier Descriptor)

อุปกรณ์ซึ่งสนับสนุนทั้งโหมดการทำงานที่ฟูลสปีดและไฮสปีดจะต้องมีดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์ ถ้าอุปกรณ์สลับความเร็วอาจทำให้ฟิลด์บางฟิลด์ในดีไวซ์ดีสคริปเตอร์เกิดการเปลี่ยนแปลงได้ ดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์จะใช้สำหรับเก็บฟิลด์ต่างๆ เหล่านี้ซึ่งเป็นของโหมดความเร็วที่ไม่ได้ใช้อยู่ในปัจจุบันเอาไว้ ค่าต่างๆที่อยู่ภายในดีไวซ์ดีสคริปเตอร์และดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์จะมีการสลับกันโดยขึ้นอยู่กับว่าในขณะนั้นความเร็วใดถูกเลือกใช้งานอยู่

ดีสคริปเตอร์ชนิดนี้จะมีฟิลด์อยู่ 9 ฟิลด์ ดีสคริปเตอร์จะประกอบไปด้วยข้อมูลเกี่ยวกับตัวดีสคริปเตอร์เอง ตัวอุปกรณ์ที่มันอธิบายอยู่ คอนฟิกิวเรชันและคลาสของอุปกรณ์ ฟิลด์ต่างๆ เหล่านี้จะเหมือนกับที่ใช้ในดีไวซ์ดีสคริปเตอร์ แต่จะแตกต่างกันอยู่เพียงแต่มันใช้อธิบายอุปกรณ์ที่ความเร็วซึ่งในขณะนั้นไม่ได้ใช้งานเท่านั้น

ค่าเวนเดอร์ไอดี และโปรดักซ์ไอดี เวอร์ชันของอุปกรณ์ และสตริงเกี่ยวกับผู้ผลิต ผลิตภัณฑ์และซีเรียลนัมเบอร์จะไม่มีการเปลี่ยนแปลงดังนั้นดีไวซ์ควอลิไฟเออร์จึงไม่มีค่านี้รวมอยู่ด้วย ต่อไปจะอธิบายแต่ละฟิลด์โดยแบ่งตามกลุ่มหน้าที่ของมัน

กลุ่มที่ใช้อธิบายตัวดีสคริปเตอร์

bLength : ความยาวเป็นไบต์ของดีสคริปเตอร์

bDescriptorTyp : ค่าคงที่ซึ่งบอกชนิดว่าเป็นดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์

กลุ่มที่ใช้อธิบายตัวอุปกรณ์

bcdUSB : เป็นตัวที่ใช้แสดงเวอร์ชันสูงสุดของ ยูเอสบี ที่อุปกรณ์ตัวนั้นสนับสนุน ซึ่งจะต้องมีค่าอย่างน้อยเท่ากับ (0200h) เนื่องจากดีไวซ์ควอลิไฟเออร์จะใช้ในการสลับโหมดความเร็วการทำงานระหว่างฟูลสปีดและไฮสปีด

กลุ่มที่ใช้อธิบายคอนฟิกิวเรชัน

bNumConfiguration : จำนวนของคอนฟิกิวเรชันที่อุปกรณ์สนับสนุน

bMaxPacketSize (0) : ค่าขนาดสูงสุดของแพ็กเก็ตที่ใช้สำหรับเอนด์พอยต์ 0

กลุ่มที่ใช้อธิบายคลาส

bDeviceClass : เป็นชื่อของคลาสที่อุปกรณ์ถูกจัดให้อยู่

bDeviceSubClass: ใช้กำหนดคลาสย่อยหรือซับคลาสให้แก่อุปกรณ์ที่ถูกจัดอยู่ในคลาสใดคลาสหนึ่ง

bDeviceProtocol : ใช้ระบุโปรโตคอลซึ่งถูกกำหนดโดยฟิลด์คลาสหรือซับคลาส

Reserved : ฟิลด์ที่สงวนไว้ใช้ในอนาคต

ออฟเซ็ต	ฟิลด์	ขนาด	ค่า	คำอธิบาย
0	bLength	1	ตัวเลข	ขนาดเป็นไบต์ของดีสคริปเตอร์
1	bDescriptorType	1	ค่าคงที่	บอกชนิดว่าเป็นดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์
2	bcdUSB	2	BCD	เวอร์ชันของข้อกำหนดที่อุปกรณ์ยอมให้ใช้งานได้
4	bDeviceClass	1	คลาส	รหัสคลาส
5	bDeviceSubClass	1	ซับคลาส	รหัสซับคลาส
6	bDeviceProtocol	1	โปรโตคอล	รหัสโปรโตคอล
7	bMaxPacketSize (0)	1	ตัวเลข	ขนาดสูงสุดของแพ็กเก็ตสำหรับเอนด์พอยต์ 0
8	bNumConfiguration	1	เลขจำนวนเต็ม	จำนวนของคอนฟิกิวเรชัน
9	Reserved	1	-	สงวนไว้ใช้งานในอนาคต

ตารางที่ 2.15 ดีไวซ์ควอลิไฟเออร์ดีสคริปเตอร์

2.2.17 คอนฟิกิวเรชันดีสคริปเตอร์ (Configuration Descriptor)

หลังจากที่โฮสต์ได้รับดีไวซ์ดีสคริปเตอร์เข้าไปแล้ว ต่อมาโฮสต์จะทำการรับคอนฟิกิวเรชันดีสคริปเตอร์ อินเตอร์เฟสดีสคริปเตอร์ และเอนด์พอยต์ดีสคริปเตอร์ของอุปกรณ์ อุปกรณ์แต่ละตัวจะมีคอนฟิกิวเรชันดีสคริปเตอร์อยู่อย่างน้อย 1 ชุดซึ่งใช้อธิบายถึงจุดเด่นและความสามารถของอุปกรณ์ ตามปกติคอนฟิกิวเรชันหรือค่าที่ใช้สำหรับการทำงานของอุปกรณ์สามารถมีได้มากกว่า 1 ชุดซึ่งแต่ละชุดจะสนับสนุนโหมดการทำงานที่แตกต่างกัน แต่ในขณะใดขณะหนึ่งจะมีการใช้งานคอนฟิกิวเรชันได้เพียงชุดเดียวเท่านั้น คอนฟิกิวเรชันแต่ละชุดจะต้องมีคอนฟิกิวเรชันดีสคริปเตอร์ของมันเอง ซึ่งคอนฟิกิวเรชันดีสคริปเตอร์นี้จะมีข้อมูลเกี่ยวกับการใช้พลังงานของอุปกรณ์และอินเตอร์เฟสที่อุปกรณ์ตัวนั้นสนับสนุนนอกจากนี้คอนฟิกิวเรชันดีสคริปเตอร์แต่ละชุดยังมีดีสคริปเตอร์ย่อยๆ อยู่ภายในตัวมันอีก ซึ่งประกอบด้วยอินเตอร์เฟสดีสคริปเตอร์อย่างน้อย 1 ชุด และอาจมีเอนด์พอยต์ดีสคริปเตอร์เพิ่มเติมด้วย

ทางฝั่งโฮสต์จะทำการเลือกคอนฟิกิวเรชันโดยใช้คำร้องขอ Set_Configuration และทำการอ่านหมายเลขคอนฟิกิวเรชันที่ใช้อยู่ในปัจจุบันด้วยคำร้องขอ Get_Configuration เมื่อคอนฟิกิวเรชันดีสคริปเตอร์

ถูกอ่านเข้ามา มันจะส่งค่าลำดับชั้นของคอนฟิกิวเรชันทั้งหมดซึ่งจะรวมถึงอินเตอร์เฟสดีสคริปเตอร์และเอนด์พอยต์ดีสคริปเตอร์ที่เกี่ยวข้องทั้งหมดออกมา โดยที่ฟิลด์ ดับเบิลยู โททอลเลงค์ จะเป็นตัวที่บอกถึงจำนวนไบต์ทั้งหมดของดีสคริปเตอร์ในลำดับชั้นรวมกัน คอนฟิกิวเรชันดีสคริปเตอร์จะประกอบด้วยฟิลด์ทั้งหมด 8 ฟิลด์ แต่ละฟิลด์จะมีข้อมูลเกี่ยวกับตัวดีสคริปเตอร์ คอนฟิกิวเรชัน และการใช้พลังงานของคอนฟิกิวเรชันนั้นๆ ต่อไปจะอธิบายแต่ละฟิลด์โดยแบ่งตามกลุ่มหน้าที่ของมัน

ออฟเซ็ต	ฟิลด์	ขนาด	ค่า	คำอธิบาย
0	bLength	1	ตัวเลข	ขนาดของดีสคริปเตอร์ในหน่วยไบต์
1	bDescriptorType	1	ค่าคงที่	คอนฟิกิวเรชันดีสคริปเตอร์
2	wTotalLength	2	ตัวเลข	ความยาวทั้งหมดของข้อมูล
4	bNumberInterface	1	ตัวเลข	จำนวนอินเตอร์เฟส
5	bConfigurationValue	1	ตัวเลข	ค่าที่ใช้เป็นเหมือนพารามิเตอร์เพื่อใช้เลือกคอนฟิกิวเรชันนี้
6	iConfiguration	1	ตัวเลขดัชนี	ตัวเลขดัชนีของสตริงดีสคริปเตอร์ที่อธิบายคอนฟิกิวเรชันนี้
7	bmAttributes	1	บิตแมพ	D7 ใช้กำลังงานจากบัส (ยูเอสบี 1.0) D6 ใช้พลังงานจากตัวอุปกรณ์เอง D5 รีโมตเวคอัพ D4...0 สงวนไว้ (0)
8	bMaxPower	1	มิลลิแอมป์	กระแสไฟฟ้าสูงสุดที่อุปกรณ์ดึงจากระบบบัส

ตารางที่ 2.16 ดีสคริปเตอร์ต่างๆที่ถูกอ่านเข้ามาพร้อมกับคอนฟิกิวเรชันดีสคริปเตอร์

กลุ่มที่ใช้อธิบายตัวดีสคริปเตอร์

bLength : ความยาวเป็นไบต์ของดีสคริปเตอร์

bDescriptorType : ค่าคงที่ซึ่งบอกชนิดว่าเป็นคอนฟิกิวเรชันดีสคริปเตอร์

wTotalLength : จำนวนของข้อมูลที่อุปกรณ์ส่งกลับออกมา ประกอบด้วยจำนวนไบต์ของคอนฟิกิวเรชันดีสคริปเตอร์ อินเตอร์เฟสดีสคริปเตอร์และเอนด์พอยต์ดีสคริปเตอร์

กลุ่มที่ใช้อธิบายคอนฟิกิวเรชัน

bConfigurationValue : เป็นหมายเลขของคอนฟิกิวเรชันซึ่งใช้กับคำร้องขอ Get_Configuration และ Set_Configuration

bMaxPacketSize (0) : ค่าขนาดสูงสุดของแพ็กเก็ตที่ใช้สำหรับเอนด์พอยต์ 0

iConfiguration : ฟิลด์นี้เป็นตัวเลขดัชนีที่ชี้ไปยังสตริงซึ่งใช้อธิบายคอนฟิกิวเรชัน

bNumberInterface : เป็นตัวระบุจำนวนของอินเตอร์เฟสซึ่งคอนฟิกิวเรชันนี้โดยค่าต่ำสุดคือ 1

กลุ่มที่ใช้อธิบายการใช้พลังงาน

bmAttribute: เป็นตัวกำหนดพารามิเตอร์ด้านพลังงานสำหรับคอนฟิกิวเรชันนี้ ถ้าอุปกรณ์เป็นแบบใช้พลังงานจากตัวมันเองจะเซ็ตบิต D6 เป็น 1 สำหรับ D7 จะถูกใช้ใน ยูเอสบี1.0 เพื่อเป็นการบอกว่าอุปกรณ์เป็นแบบใช้พลังงานจากบัส แต่ตั้งแต่ ยูเอสบี เวอร์ชัน 1.1 เป็นต้นไปบิตนี้จะสงวนไว้และเซ็ตให้เป็น 1 สำหรับอุปกรณ์ที่ใช้พลังงานบางส่วนจากบัส ไม่ว่าจะเป็นอุปกรณ์แบบใช้พลังงานจากบัสหรือใช้พลังงานจากตัวเอง มันจะรายงานค่าความสิ้นเปลืองพลังงานที่ฟิลด์ บีแมกพาวเวอร์ นอกจากนี้อุปกรณ์สามารถสนับสนุนรีโมตเวคอัพ ซึ่งอนุญาตให้อุปกรณ์กระตุ้นโฮสต์เมื่อโฮสต์เข้าสู่โหมดซัซเพนด์ได้ด้วย

bMaxPower : เป็นตัวกำหนดพลังงานสูงสุดซึ่งอุปกรณ์ดึงจากบัสค่าของฟิลด์นี้ 1 หน่วยเท่ากับ 2 มิลลิแอมป์ เช่น ถ้าบีแมกพาวเวอร์มีค่าเท่ากับ 50 จะหมายถึง 100 มิลลิแอมป์ เป็นต้น ในข้อกำหนด ยูเอสบี อนุญาตให้อุปกรณ์แบบใช้พลังงานจากบัสสูง จะดึงกระแสจากบัสได้ไม่เกิน 500 มิลลิแอมป์ ถ้าอุปกรณ์สูญเสียกำลังงานจากภายนอก มันจะไม่สามารถดึงกระแสได้เกินที่ระบุไว้ในบีแมกพาวเวอร์ ซึ่งทำให้การทำงานบางส่วนที่ต้องใช้แหล่งจ่ายไฟภายนอกไม่สามารถใช้งานได้

2.2.18 อัทเธอร์สปีดคอนฟิกิวเรชันดีสคริปเตอร์ (Other Speed Configuration Descriptor)

ดีสคริปเตอร์ชนิดนี้มีใช้เฉพาะกับอุปกรณ์ซึ่งสนับสนุนทั้งฟูลสปีดและไฮสปีด โครงสร้างของ อัทเธอร์สปีดคอนฟิกิวเรชันดีสคริปเตอร์จะเหมือนกับคอนฟิกิวเรชันดีสคริปเตอร์ชนิดนี้ใช้อธิบายคอนฟิกิวเรชันของโหมดความเร็วที่ไม่ได้ใช้ในปัจจุบัน อัทเธอร์สปีดคอนฟิกิวเรชันดีสคริปเตอร์จะมีดีสคริปเตอร์ย่อยเช่นเดียวกับคอนฟิกิวเรชันดีสคริปเตอร์ ส่วนโครงสร้างของดีสคริปเตอร์ประกอบไปด้วย 8 ฟิลด์

2.2.19 อินเตอร์เฟสดีสคริปเตอร์ (Interface Descriptor)

อินเตอร์เฟสในที่นี้หมายถึงกลุ่มของเอนด์พอยต์ซึ่งถูกใช้โดยฟังก์ชัน ซึ่งอินเตอร์เฟสดีสคริปเตอร์จะมีข้อมูลของเอนด์พอยต์ซึ่งอินเตอร์เฟสนั้นสนับสนุน

คอนฟิกิวเรชันแต่ละชุดสามารถ มีอินเตอร์เฟสได้มากกว่า 1 ชุด ซึ่งแต่ละชุดสามารถใช้งานในเวลาเดียวกันได้อย่างอิสระ สำหรับอินเตอร์เฟสแต่ละชุดนั้นจะต้องมีอินเตอร์เฟสดีสคริปเตอร์ของมันเอง และมีดีสคริปเตอร์ย่อยเป็นเอนด์พอยต์ดีสคริปเตอร์สำหรับอธิบายเอนด์พอยต์แต่ละตัวซึ่งอินเตอร์เฟสนั้นสนับสนุน

ออฟเซ็ต	ฟิลด์	ขนาด	ค่า	คำอธิบาย
0	bLength	1	ตัวเลข	ขนาดของดีสคริปเตอร์ในหน่วยไบต์
1	bDescriptorType	1	ค่าคงที่	อินเตอร์เฟสดีสคริปเตอร์
2	bInterfaceNumber	1	ตัวเลข	จำนวนของอินเตอร์เฟส
3	bAlternateSetting	1	ตัวเลข	ค่าที่ใช้สำหรับเลือกกลุ่มของอินเตอร์เฟส
4	bNumEndPoint	1	ตัวเลข	จำนวนของเอนด์พอยต์ที่ใช้สำหรับอินเตอร์เฟสนี้
5	bInterfaceClass	1	คลาส	คลาส (ถูกกำหนดโดยกลุ่มผู้สร้างมาตรฐาน ยูเอสบี)
6	bInterfaceSubClass	1	ซับคลาส	ซับคลาส (ถูกกำหนดโดยกลุ่มผู้สร้างมาตรฐานยูเอสบี)
7	bInterfaceProtocol	1	โปรโตคอล	โปรโตคอล
8	iInterface	1	ตัวเลขดัชนี	ตัวเลขดัชนีของสตริงดีสคริปเตอร์ที่อธิบายอินเตอร์เฟสนี้

ตารางที่ 2.17 อินเตอร์เฟสดีสคริปเตอร์

อุปกรณ์ที่มีคอนฟิกิวเรชันซึ่งมีอินเตอร์เฟสหลายชุด โดยที่แต่ละชุดทำงานในเวลาเดียวกันจะเรียกว่าอุปกรณ์คอมโพสิท ซึ่งในการใช้งานนั้นโฮสต์จะโหลดดีไวซ์ไดร์ฟเวอร์สำหรับแต่ละอินเตอร์เฟสขึ้นมาใช้งาน

นอกจากนี้ภายในคอนฟิกิวเรชันอาจมีการสนับสนุนอินเตอร์เฟสได้หลายกลุ่ม ซึ่งแยกจากกันเป็นอิสระ โดยแต่ละกลุ่มอาจประกอบไปด้วยอินเตอร์เฟสอีกหลายชุด โฮสต์จะร้องขอให้มีการเปลี่ยนไปใช้อินเตอร์เฟสอีกกลุ่มหนึ่งโดยการใช้คำร้องขอ Set_Interface และทำการอ่านหมายเลขอินเตอร์เฟสโดยใช้คำร้องขอ Get_Interface

อินเตอร์เฟสดีสคริปเตอร์ประกอบไปด้วย 9 ฟิลด์ ส่วนความหมายของแต่ละฟิลด์ซึ่งแบ่งตามกลุ่มลักษณะการทำงานของมันสามารถอธิบายได้ดังนี้

กลุ่มที่ใช้อธิบายตัวดีสคริปเตอร์

bLength : ความยาวเป็นไบต์ของดีสคริปเตอร์

bDescriptorType : ค่าคงที่ซึ่งบอกชนิดว่าเป็นอินเตอร์เฟสดีสคริปเตอร์

กลุ่มที่ใช้อธิบายอินเตอร์เฟส

iInterface : เป็นตัวเลขดัชนีที่ชี้ไปที่สตริงที่ใช้ในการอธิบายอินเตอร์เฟส

bInterfaceNumber : หมายเลขของอินเตอร์เฟส ในอุปกรณ์คอมโพสิทที่คอนฟิกิวเรชันของมันมีอินเตอร์เฟสหลายชุดซึ่งสามารถทำงานในเวลาเดียวกันได้ โดยแต่ละอินเตอร์เฟสนั้นจะต้องมีค่าไม่ซ้ำกัน สำหรับค่าเริ่มต้นของฟิลด์นี้คือ 0

bAlternateSetting : ในกรณีที่คอนฟิกิวเรชันสนับสนุนอินเตอร์เฟสหลายกลุ่มซึ่งแต่ละกลุ่มจะต้องมีค่า บีอินเตอร์เฟสนัมเบอร์ ในอินเตอร์เฟสดีสคริปเตอร์เหมือนกันแต่จะต้องมีค่า บีอัลเทอร์เนทเซ็ตติ้ง ต่างกัน คำร้องขอ{Get_Interface} จะรับเอาค่าของ บีอัลเทอร์เนทเซ็ตติ้ง ที่ถูกเซ็ตอยู่ในปัจจุบันกลับมา ส่วนคำร้องขอ {Set_Interface} จะใช้สำหรับเลือกกลุ่มของอินเตอร์เฟส

bNumberEndpoint : เป็นตัวบอกถึงจำนวนเอนด์พอยต์ซึ่งถูกใช้ในอินเตอร์เฟสนี้ ค่านี้ไม่นับรวมเอนด์พอยต์ 0 นอกจากนี้ บีนัมเบอร์เอนด์พอยต์ ยังใช้ในการบอกถึงจำนวนเอนด์พอยต์ดีสคริปเตอร์ที่จะตามมาอีกด้วย

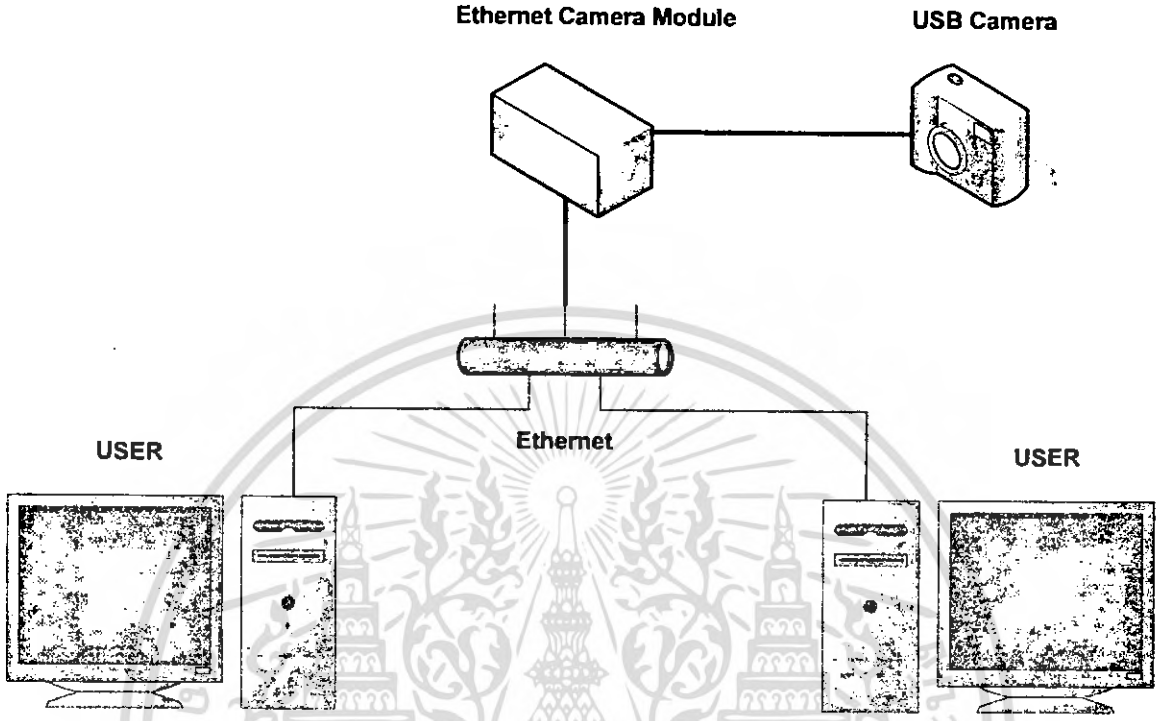
bInterfaceClass : เหมือนกับ บีดีไวซ์คลาส แต่จะใช้สำหรับอุปกรณ์ที่ระบุคลาสด้วย อินเตอร์เฟส ค่าตั้งแต่ 1ถึง FEh จะถูกสงวนไว้สำหรับคลาสที่ ยูเอสบี กำหนดขึ้น ส่วนค่า FFh หมายถึงค่าที่กำหนดขึ้นมาจากผู้ผลิตเอง และ 0 จะเป็นค่าที่สงวนไว้

bInterfaceSubClass : ใช้กำหนดซับคลาสให้กับอุปกรณ์ที่ถูกจัดให้อยู่ในคลาสใดคลาสหนึ่ง เหมือนกับ บีอินเตอร์เฟสคลาส แต่จะใช้สำหรับอุปกรณ์ที่ระบุคลาสด้วยอินเตอร์เฟส ถ้าฟิลด์ บีอินเตอร์เฟสคลาส เป็น 0 ซับคลาสต้องมีค่าเป็น 0 ด้วยถ้า บีอินเตอร์เฟสคลาส มีค่าอยู่ระหว่าง 1 และ FEh ซับคลาสจะต้องมีค่าตามที่ระบุไว้ในข้อกำหนด ยูเอสบี และถ้า บีอินเตอร์เฟสซับคลาส มีค่าเท่ากับ (FFh) ซับคลาสจะถูกกำหนดจากผู้ผลิต

bInterfaceProtocol : เหมือนกับ บีดีไวซ์โปรโตคอล แต่จะใช้สำหรับอุปกรณ์ที่ระบุคลาสด้วยอินเตอร์เฟส ฟิลด์นี้ใช้ระบุโปรโตคอลซึ่งถูกกำหนดโดยฟิลด์คลาสหรือซับคลาส ถ้า บีอินเตอร์เฟสโปรโตคอล มีค่าอยู่ระหว่าง 1 และ FEh โปรโตคอลจะต้องเป็นไปตามข้อกำหนด ยูเอสบี

บทที่ 3
การคำนวณและการสร้าง

3.1 โครงสร้างโดยรวมของระบบ

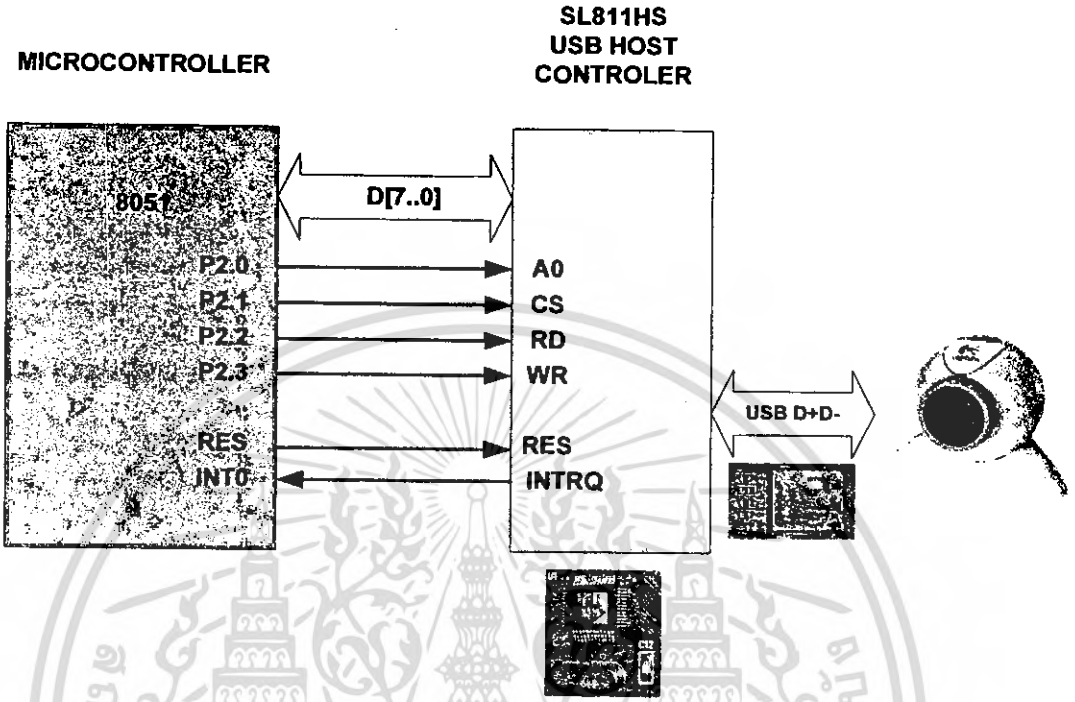


รูปที่ 3.1 โครงสร้างโดยรวมของระบบ

3.2 การทำงานของระบบ

โดยเริ่มจาก "อีเตอร์เน็ตคาเมล่าโมดูล" ทำหน้าที่ควบคุมการทำงานของอุปกรณ์คือกล้องยูเอสบี โดยโมดูลตัวนี้จะมีส่วนที่ทำหน้าที่เป็น ยูเอสบีโฮสต์คอนโทรลเลอร์เพื่อเป็นตัวจัดการให้รูปแบบการรับส่งข้อมูลเป็นไปตามมาตรฐานการติดต่อแบบ ยูเอสบี เพื่อที่จะสามารถติดต่อกับอุปกรณ์ ยูเอสบี ได้ และโมดูลนี้ก็มีส่วนของการจัดการข้อมูลให้เป็นไปตามมาตรฐานของระบบเครือข่ายท้องถิ่นเพื่อส่งข้อมูลภาพผ่านระบบเครือข่ายท้องถิ่นซึ่งเป็นระบบเครือข่ายที่ได้รับความนิยมมากในปัจจุบันให้มีความสะดวกกับผู้ใช้การเฝ้าระวังทรัพย์สินของตัวเอง เพราะผู้ใช้สามารถเฝ้าดูทรัพย์สินได้ทุกที่ทุกเวลาภายในระบบเครือข่ายท้องถิ่นหรือภายในองค์กร

3.3 โครงสร้างของอีเตอร์เน็ตคาเมล่าโมดูล



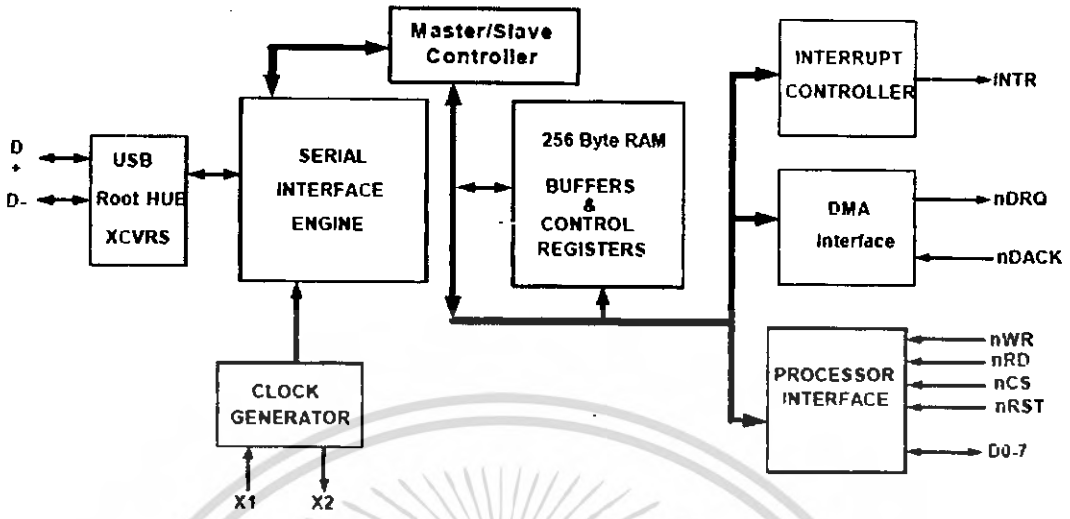
รูปที่ 3.2 โครงสร้างของอีเตอร์เน็ตคาเมล่าโมดูล

การทำงานของระบบโดยมีอุปกรณ์หลักคือ ไอซี SL811HS ซึ่งเป็น ยูเอสบีโฮสต์คอนโทรลเลอร์มาทำหน้าที่นี้แค่การทำงานจะต้องมีการควบคุมการทำงานให้เป็นไปตามมาตรฐานของอุปกรณ์ที่เข้ามาเชื่อมต่อโดยจะต้องใช้ซอฟต์แวร์ที่บรรจุในตัวอุปกรณ์ในการจัดการเรียกว่า "เฟิร์มแวร์" บล็อกไดอะแกรมการทำงานของไอซี SL811HS

3.4 บล็อกไดอะแกรม

ไอซี SL811HS เป็นทั้ง ยูเอสบี โฮสต์/เสลฟ คอนโทรลเลอร์ ที่มีความสามารถในการสื่อสารได้ดีทั้งความเร็วต่ำหรือความเร็วสูง SL811HS สามารถเชื่อมต่อกับอุปกรณ์ เช่น ไมโครโปรเซสเซอร์ ไมโครคอนโทรลเลอร์ หรือเชื่อมต่อโดยตรงกับระบบบัส เช่น ISA, PCMCIA และอื่น ๆ SL811HS ยูเอสบี โฮสต์คอนโทรลเลอร์มีความสอดคล้องกับยูเอสบีสเปกซ์ซิฟิเคชัน 1.1

SL811HS ยูเอสบี โฮสต์/เสลฟ คอนโทรลเลอร์มีฟังก์ชันการรับส่งทั้งความเร็วต่ำและความเร็วเต็มที่อยู่ร่วมกัน โดย SL811HS สนับสนุนและทำงานในโหมดความเร็วเต็มที่ ที่ความเร็ว 12 เมกะบิตต่อวินาทีและในโหมดความเร็วต่ำที่ 1.5 เมกะบิตต่อวินาที

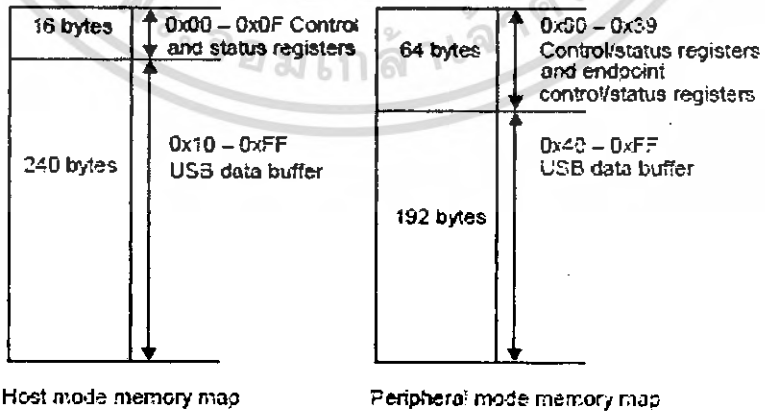


รูปที่ 3.3 SL811HS USB Host/Slave Controller Functional Block Diagram

3.5 บัฟเฟอร์ เมมโมรี่

SL811HS มีความจุภายใน 256 ไบต์ ใช้สำหรับยูเอสบีดาต้าบัฟเฟอร์ รีจิสเตอร์ควบคุม สถานะ รีจิสเตอร์ เมื่อต้องการทำเป็นโหมดมาสเตอร์ จะกำหนด เมมโมรี่ 16 ไบต์สำหรับเป็นรีจิสเตอร์ และใช้ เมมโมรี่ที่เหลือ 240 ไบต์ใช้สำหรับยูเอสบีดาต้าบัฟเฟอร์ เมื่อต้องการทำเป็นโหมดเสลฟ จะใช้ 64 ไบต์แรกเป็น เอนด์พอยต์ คอนโทรล และ รีจิสเตอร์ส่วน 192 ไบต์ที่เหลือเป็นที่ว่างสำหรับ เอนด์พอยต์บัฟเฟอร์ และสำหรับส่งผ่านข้อมูล

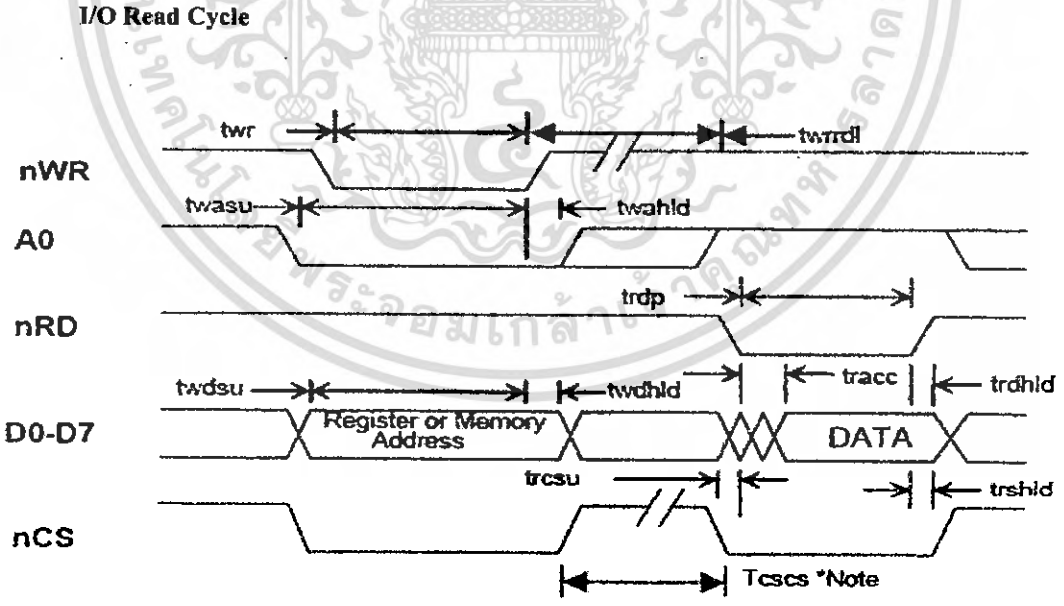
การเข้าถึงรีจิสเตอร์และดาต้าเมมโมรี่จะผ่านทางดาต้าบัส 8 บิตของไมโครคอนโทรลเลอร์ภายนอก และอื่น ๆ หรือเข้าสู่แอดเดรสโดยตรง และยังมีโหมดการทำงานแบบ Auto Address Incriment จะใช้เมื่อเป็นการเขียนและอ่านตำแหน่งแอดเดรสโดยตรง การปรับเปลี่ยนบัฟเฟอร์สำหรับการส่งผ่าน โดยสามารถทำได้ในรีจิสเตอร์คอนโทรลดังนั้นตำแหน่งและการแบ่งขนาดสามารถจัดสรรในเมมโมรี่บัฟเฟอร์ได้



รูปที่ 3.4 Memory Map

CHIP SELECT (NCS) - เอ็นซีเอสใช้ในการเปิดการทำงานในการเชื่อมต่อของ SL811HS และใช้ในการอ่านหรือเขียน สัญญาณ เอ็นซีเอส เป็นสัญญาณที่สำคัญในการส่งผ่านข้อมูลให้เป็นไปตามที่ต้องการสำหรับ "ชิปนี้" เมื่อต้องการใช้ก็จะส่งสัญญาณ "0"มาที่ขานี้ซึ่งจะใช้ร่วมกับสัญญาณอ่านหรือเขียน สัญญาณ เอ็นซีเอส จะต้องถูกยืนยันโดยทางฝั่งประมวลผลอย่างน้อยที่สุด 65 นาโนวินาทีระหว่างการส่งผ่านตามลำดับ ซึ่งจะทำให้การส่งผ่านประสบผลสำเร็จ โดยหน่วยประมวลผลจะต้องการเฟิร์มแวร์เข้ามาจัดการให้ได้ตามเวลาที่วางไว้หรือสถานการณ์รอเพื่อให้ เอ็นซีเอส ทำตามกระบวนการที่สมบูรณ์ เอ็นซีเอส ก็สามารถที่จะทำการยืนยันอย่างต่อเนื่องได้ (ถ้าใช้ชิปตัวเดียวสามารถต่อลง กราวด์ได้)

READ (NRD) - เอ็นอาร์ดี เป็นสัญญาณที่ทำงานที่ แอคทีฟ Low โดยที่ฝั่งตัวประมวลผลจะเป็นตัวใช้เพื่อที่จะใช้ในการอ่าน รีจิสเตอร์ หรือเมมโมรี่ ก่อนที่จะทำการอ่านจะต้องชี้ตำแหน่งแอดเดรสที่จะทำการอ่านจะต้องระบุไปใน SL811HS ระหว่างการอ่าน เอ็นซีเอส ต้องยืนยันตามลำดับทำให้ SL811HS เข้าใจว่ามีการอ่านที่ขานี้เมื่อยืนยันขา เอ็นอาร์ดี โดยที่ความกว้างของพัลส์ที่น้อยที่สุดของ เอ็นอาร์ดี เท่ากับ 65 นาโนวินาที หลังจากการยืนยันขาเอ็นอาร์ดี 65 นาโนวินาที สัญญาณ D[7:0] จะเปลี่ยนจากโหมด hi-z ไปเป็นโหมดไดร์ฟวิ่ง และส่งข้อมูลบนบัสจนกระทั่งถึง 5 นาโนวินาที หลังจากที่เอ็นอาร์ดี ไม่ได้ยืนยัน โดยที่การทิ้งช่วงเวลาที่น้อยที่สุดในระหว่างการอ้างอิงเอ็นอาร์ดี คือ 85 นาโนวินาที

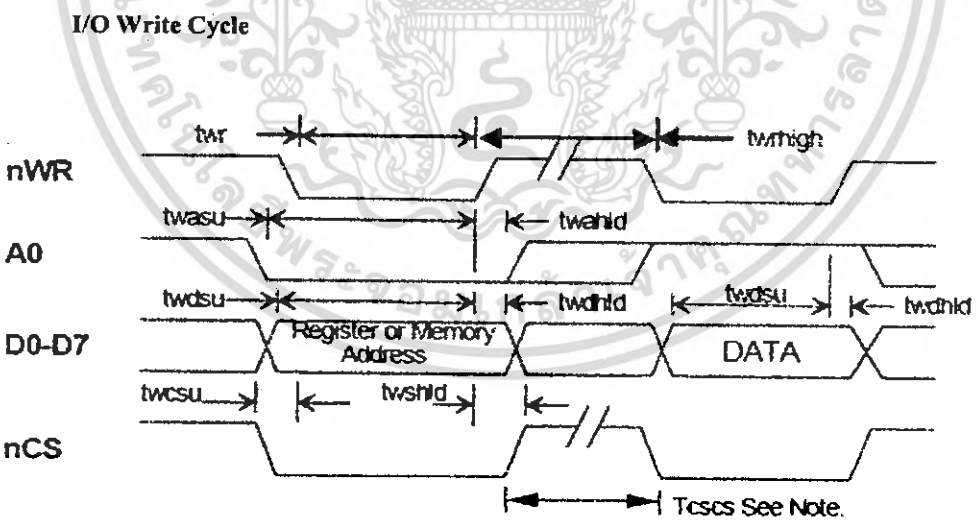


รูปที่ 3.5 I/O Read Cycle from Register or Memory Buffer

Parameter	Description	Min.	Typ.	Max.
t_{WR}	Write pulse width	85 ns		
t_{RD}	Read pulse width	85 ns		
t_{WCSU}	Chip select set-up to nWR	0 ns		
t_{WASU}	A0 address set-up time	85 ns		
t_{WAHLD}	A0 address hold time	10 ns		
t_{WDSU}	Data to Write HIGH set-up time	85 ns		
t_{WDHLD}	Data hold time after Write HIGH	5 ns		
t_{RACC}	Data valid after Read LOW	25 ns		85 ns
t_{RDHLD}	Data hold after Read HIGH	40 ns		
t_{RCSU}	Chip select LOW to Read LOW	0 ns		
t_{RSHLD}	NCS hold after Read HIGH	0 ns		
T_{CSCS}*	nCS inactive to nCS asserted	85 ns		
t_{WRRDL}	nWR HIGH to nRD LOW	85ns		

ตารางที่ 3.1 I/O Read Cycle from Register or Memory Buffer

WRITE (nWR) -เอ็นดับเบิลยูอาร์ เป็นสัญญาณที่ทำงานที่ แอคทีฟ Low โดยที่ฝั่งตัวประมวลผลจะเป็นผู้ใช้ในการเขียนตำแหน่ง รีจิสเตอร์ หรือ เมมโมรี่โดย เอ็นดับเบิลยูอาร์ จะใช้ร่วมกับ เอ็นซีเอส จะต้องยืนยันตามลำดับสำหรับ SL811HS จะยืนยันสัญญาณ เอ็นดับเบิลยูอาร์ ให้อยู่ที่ Low น้อยที่สุดคือ 65 นาโนวินาทีข้อมูลจะถูกเขียนจากฝั่งประมวลผลไปยัง SL811HS ข้อมูลยังคงอยู่บนบัส 5 นาโนวินาทีหลังยกเลิกการยืนยันและทิ้งช่วงเวลาที่น้อยที่สุดก่อนเขียนข้อมูล เอ็นดับเบิลยูอาร์ อย่างน้อย 85 นาโนวินาที



รูปที่ 3.6 I/O Write Cycle to Register or Memory Buffer

Parameter	Description	Min.	Typ.	Max.
t_{WR}	Write pulse width	85 ns		
t_{WCSU}	Chip select set-up to nWR LOW	0 ns		
t_{WSHLD}	Chip select hold time After nWR HIGH	0 ns		
t_{WASU}	A0 address set-up time	85 ns		
t_{WAHLD}	A0 address hold time	10 ns		
t_{WDSU}	Data to Write HIGH set-up time	85 ns		
t_{WDHLD}	Data hold time after Write HIGH	5 ns		
t_{CSCS}	nCS inactive to nCS* asserted	85 ns		
t_{WRHIGH}	NWR HIGH	85 ns		

ตารางที่ 3.2 I/O Write Cycle to Register or Memory Buffer

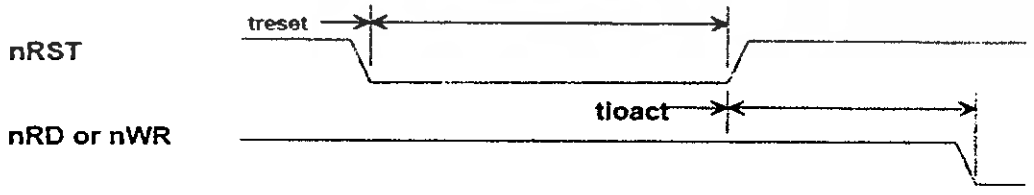
ADDRESS (A0) –A0 เป็นสัญญาณที่ถูกควบคุมโดยฝั่งตัวประมวลผล และถูกใช้ร่วม กับเอ็นดับเบิลยูอาร์ และ เอ็นอาร์ดี เป็นสัญญาณถ้า A0 มีค่าเป็น Low เป็นข้อมูลที่เขียนไปจะมองว่าเป็นตัวชี้ตำแหน่งแอดเดรสหรือ ข้อมูล ในขณะ A0 มีค่าเป็น High ข้อมูลที่เขียนไปจะมองว่าเป็นข้อมูล

INTERRUPT (INTRQ) - อินเตอร์รัปต์ เป็นการขัดจังหวะของสัญญาณที่ถูกยืนยันด้วยสถานะ High โดย SL811HS/S ระหว่างในเหตุการณ์ขัดจังหวะ ตัวอย่างนี้อาจจะรวมไปถึง การเชื่อมต่อของอุปกรณ์เข้ามาใหม่ อินเตอร์รัปต์ จะถูกยืนยันโดยสถานะ High จนกระทั่งเหตุการณ์การขัดจังหวะโดยการ เขียนความสัมพันธ์ของอินเตอร์รัปต์รีจิสเตอร์ ใน SL811HS/S

DATA BUS (D[7:0]) – SL811HS จะส่งข้อมูลเป็นแบบสองทิศทาง บนดาต้าบัสใช้ในการส่งผ่านข้อมูล ทั้งอินพุตและเอาต์พุต ในรีจิสเตอร์ หรือเมมโมรี่ โดยที่ปกติดาต้าบัสจะมีสถานะ High อิมพีแดนซ์ ถ้าไม่มีการยืนยันของ เอ็นซีเอส และเอ็นอาร์ดี ในการส่งผ่านระหว่างขา D [7:0]

RESET (NRST) - จะทำการรีเซ็ตเมื่อมีสถานะ Low ที่ขณะดอนจ่ายพลังงาน โดยจะทำการที่ฝั่งตัวประมวณผล หรือ วงจรภายนอก เอ็นอาร์เอสที จะถูกยืนยันเป็นเวลา 16 สัญญาณคล็อก

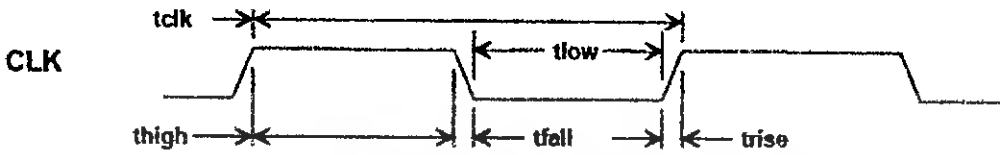
RESET TIMING



รูปที่ 3.7 RESET TIMING

Parameter	Description	Min.	Typ.	Max.
t_{RESET}	nRst Pulse width	16 clocks		
t_{IOACT}	nRst HIGH to nRD or nWR active	16 clocks		

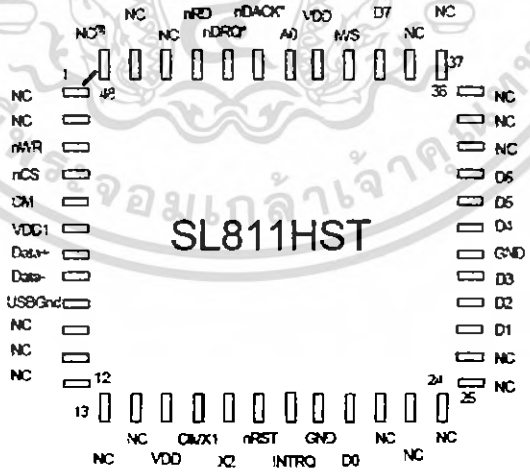
ตารางที่ 3.3 Clock Timing Specifications



รูปที่ 3.8 CLOCK TIMING

Parameter	Description	Min.	Typ.	Max.
t_{CLK}	Clock Period (48 MHz)	20.0 ns	20.8 ns	
t_{HIGH}	Clock HIGH Time	9 ns		11 ns
t_{LOW}	Clock LOW Time	9 ns		11 ns
t_{RISE}	Clock rise Time			5.0 ns
t_{FALL}	Clock fall Time			5.0 ns
	Clock Duty Cycle	45%		55%

ตารางที่ 3.4 CLOCK TIMING



รูปที่ 3.9 SL811HST-AC USB Host/Slave Controller Pin Layout

3.6 อธิบายการทำงานของ {SL811HST-AC USB Host Controller}

SL811HST-AC 48-pin TQFP เป็นอุปกรณ์ที่ต้องการแหล่งจ่ายพลังงาน 3.3 โวลต์ดีซี ต้องการคริสตอลหรือคล็อกจากภายนอก 12 หรือ 48 เมกะเฮิร์ด

Pin No.	Pin Type	Pin Name	Pin Description
1	NC	NC	NC
2	NC	NC	NC
3	IN	nWR	**Write Strobe Input.** An active LOW input used with nCS to Write to registers/data memory.
4	IN	nCS	**Active LOW SL811HST-AC Chip select**. Used with nRD and nWr when accessing SL811HT.
5	IN	CM	**Clock Multiply. Select** 12-MHz/48-MHz Clock Source.[9]
6	VDD1	+3.3 VDC	**Power for USB Transceivers.** V_{DD1} may be connected to V_{CC}.
7	BIDIR	DATA +	USB Differential Data Signal HIGH Side.
8	BIDIR	DATA -	USB Differential Data Signal LOW Side.
9	GND	USB GND	Ground Connection for USB.
10	NC	NC	NC
11	NC	NC	NC
12	NC	NC	NC
13	NC	NC	NC
14	NC	NC	NC
15	VDD	+3.3 VDC	SL811HST-AC Device V_{DD} Power[10]
16	IN	CLK/X1	Clock or External Crystal X1 connection.[11]
17	OUT	X2	External Crystal X2 connection.
18	IN	nRST	SL811HST-AC Device active low reset input.
19	OUT	INTRQ	Active HIGH Interrupt Request output to external controller.
20	GND	GND	SL811HST-AC Device Ground.
21	BIDIR	D0	**Data 0.** Microprocessor Data/(Address) Bus.
22	NC	NC	NC
23	NC	NC	NC
24	NC	NC	NC
25	NC	NC	NC
26	NC	NC	NC
27	BIDIR	D1	**Data 1.** Microprocessor Data/(Address) Bus.
28	BIDIR	D2	**Data 2.** Microprocessor Data/(Address) Bus
29	BIDIR	D3	**Data 3.** Microprocessor Data/(Address) Bus.
30	GND	GND	SL811HST-AC Device Ground
31	BIDIR	D4	**Data 4.** Microprocessor Data/(Address) Bus.
32	BIDIR	D5	**Data 5.** Microprocessor Data/(Address) Bus.

Notes:
9. The CM Clock Multiplier pin should be tied HIGH for a 12-MHz clock source and tied to ground for a 48-MHz clock source. In SL11H, this pin was designated as ALE input pin.
10. VDD can be derived from the USB supply. See diagram.
11. The X1/X2 Clock requires external 12- or 48-MHz matching crystal or clock source.

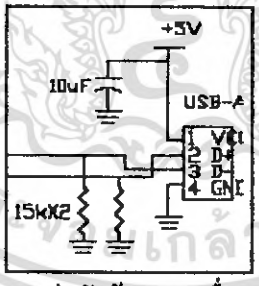
ตารางที่3.5 SL811HST-AC Pin Assignments and Definitions

Pin No.	Pin Type	Pin Name	Pin Description
33	BIDIR	D6	**Data 6**. Microprocessor Data/(Address) Bus.
34	NC	NC	NC
35	NC	NC	NC
36	NC	NC	NC
37	NC	NC	NC
38	NC	NC	NC
39	BIDIR	D7	**Data 7**. Microprocessor Data/(Address) Bus.
40	IN	M/S	**Master/Slave Mode Select**. "1" selects Slave. "0" = Master.
41	VDD	+3.3 VDC	SL811HST-AC Device V_{DD} Power.
42	IN	A0	**A0 = "0."** Selects address pointer. Reg.A0 = "1." Selects data buffer or register.[12]
43	IN	nDACK	**DMA Acknowledge**. An active LOW input used to interface to an external DMA controller. DMA is enabled only in slave mode. In host mode, pin should be tied HIGH (logic "1").
44	OUT	nDRQ	**DMA Request**. An active LOW output used with an external DMA controller. nDRQ and nDACK form the handshake for DMA data transfers. In host mode, pin must be left unconnected.
45	IN	nRD	**Read Strobe Input**. An active LOW input used with nCS to Read registers/data memory.
46	NC	NC	NC
47	NC	NC	NC
48	NC	NC	NC

Notes:
12. The A0 Address bit is used to access address register or data registers in I/O Mapped or Memory Mapped applications.

ตารางที่ 3.5 SL811HST-AC Pin Assignments and Definitions (ต่อ)

SL811HST เป็นเวอร์ชัน 48 ขาโครงสร้างภายนอกได้ประยุกต์มาจากแพ็กเก็ต 28 ขา โดยรูปที่ 3.11 เป็นการต่อการทำงานของยูเอสบีเอ็มเบดเด็ด โฮสต์ ต่อรวมกับไมโครคอนโทรลเลอร์8051 โดยจากวงจรในส่วนของ SL811HST จะใช้สัญญาณคล็อก 12-/48 เมกะเฮิร์ต จากคริสตอลและจะต้องอาศัยแหล่งจ่าย 3.3โวลต์ การเลี้ยงวงจรการทำงาน จะมีส่วนของการตรวจสอบการเชื่อมต่อดังรูป 3.10



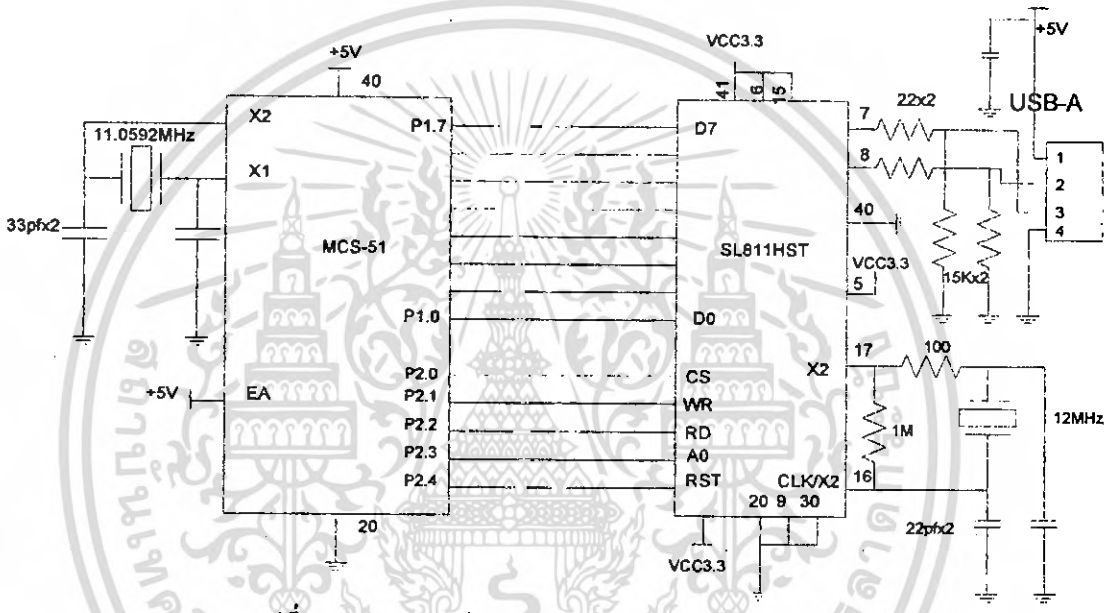
รูปที่ 3.10 แสดงการต่อตัวต้านทานเพื่อตรวจสอบสถานการณ์เชื่อมต่อ

การตรวจสอบการเชื่อมต่อของ ยูเอสบี นั้นจะตรวจสอบการเปลี่ยนแปลงของระดับสัญญาณในสายสัญญาณ D- และ D+ โดยสายข้อมูลทั้งสองที่ด้านฮับจะถูกต่อด้วยตัวต้านทาน15 กิโลโอห์ม พูลดาวน์ไว้ทั้งสองเส้น ซึ่งส่งผลให้สายสัญญาณทั้งสองมีระดับแรงดันเป็น 0 V ในขณะไม่มีอุปกรณ์ใด ๆ ต่ออยู่

ที่ด้านอุปกรณ์ สำหรับอุปกรณ์ความเร็วต่ำสายสัญญาณ D- จะต่อตัวต้านทาน1.5 กิโลโอห์ม พูลอัปกับแรงดัน 3.0 – 3.6 โวลต์ ส่วนอุปกรณ์ความเร็วสูงสายสัญญาณที่ต่อพูลอัปคือ สายสัญญาณ D+ เมื่อมี

อุปกรณ์ตัวใหม่ถูกต่อเข้ากับโฮสต์จะเกิดการแบ่งแรงดันจากตัวต้านทานทั้งสอง ทำให้แรงดันของสายสัญญาณเพิ่มจาก 0โวลต์ ขึ้นไปที่ 90% ของไฟเลี้ยงที่พูลอัป

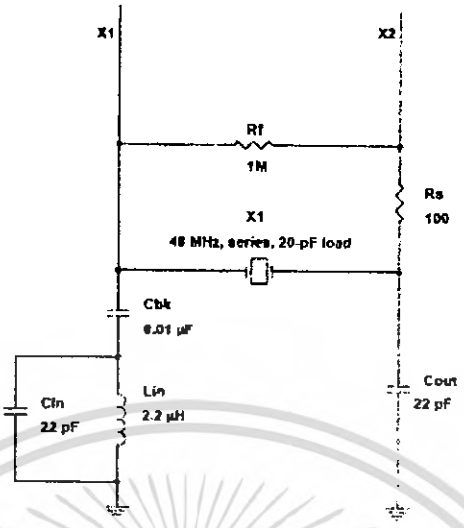
คำนวณได้จาก $\frac{15x10^3}{(1.5+15)x10^3 xVcc}$



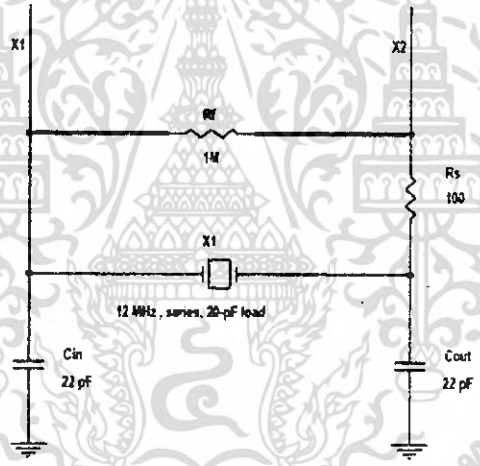
รูปที่ 3.11 แสดงการต่อวงจรระหว่าง 8051 และ SL811HST

3.7 พีแอลแอล คล็อค เจนเนอเรเตอร์ (PLL Clock Generator)

สามารถใช้สัญญาณจากคริสตอล 12 เมกะเฮิร์ตหรือสัญญาณจากคริสตอล ภายนอก 48 เมกะเฮิร์ต โดยสามารถเลือกใช้โดยการเซ็ตที่ขา ซีเอ็มของ SL811HST โดยถ้าใช้ 12 เมกะเฮิร์ต ขานี้จะต่อกับ VDC แต่ถ้าใช้สัญญาณจากคริสตอล 48 เมกะเฮิร์ต หรือสัญญาณคล็อกภายนอกขานี้ต่อลง กราวด์ ทั้ง 2 ขา นี้มีเงื่อนไขการต่อแบบง่ายที่สุดดังแสดงการต่อวงจรและอุปกรณ์ดังรูปที่ 3.12 และ 3.13 ถ้าใช้สัญญาณคล็อกจากภายนอกสามารถทำได้โดยสามารถใช้คล็อกแทนวงจรคล็อกโดยการต่อสัญญาณคล็อกโดยตรงกับขาอินพุต $X1$ เมื่อใช้สัญญาณคล็อกจากภายนอก ขา $X2$ จะไม่ถูกใช้งาน



รูปที่ 3.12 Full-Speed 48-MHz Crystal Circuit



รูปที่ 3.13 Optional 12-MHz Crystal Circuit

3.8 SL811HS Registers

การดำเนินการและคอนโทรลของ SL811HS ถูกจัดการผ่านรีจิสเตอร์ภายใน เมื่อทำงานในโหมดโฮสต์ ตำแหน่ง 16 ไบต์แรกจะถูกกำหนดเป็นพื้นที่ของรีจิสเตอร์ แต่ถ้าทำงานในโหมด เสลฟตำแหน่ง 64 ไบต์แรกจะถูกกำหนดเป็นพื้นที่ของรีจิสเตอร์ ตาราง 3.6 แสดงแผนที่หน่วยความจำและรีจิสเตอร์ของทั้งคู่ทั้ง SL11H และ SL811HS ใน โหมด SL11H จะแสดงสำหรับผู้ที่เคยใช้และต้องการเปลี่ยนมาใช้ SL811HS

โหมดรีจิสเตอร์ (SL811HS Master Mode Registers)

รีจิสเตอร์ใน SL811HS ถูกแบ่งออกเป็น 2 กลุ่มโดยกลุ่มแรกจะถูกอ้างอิงเป็น ยูเอสบีคอนโทรล รีจิสเตอร์เหล่านี้จะทำการเปิดและเตรียมสถานะ สำหรับควบคุมการสื่อสารของ ยูเอสบี และการไหลของข้อมูล ส่วนกลุ่มที่ 2 เป็นรีจิสเตอร์สำหรับเตรียมการควบคุมและสถานการณ์ทำงานทั้งหมด

รีจิสเตอร์ (Register Values on Power-up and Reset)

รีจิสเตอร์ดังต่อไปนี้จะมีค่าเริ่มต้นเป็นศูนย์ตอนจ่ายไฟและรีเซ็ต

- USB-A/USB-B Host Control Register [00H, 08H] bit 0 only
- Control Register 1 [05H]
- USB Address Register [07H]
- Current Data Set/Hardware Revision/SOF Counter LOW

Bit Position	Bit Name	Function
7	Preamble	ถ้าบิต = "1" จะส่ง preamble token ไปก่อนเพื่อบอกว่าแพ็กเก็ตที่ตามมาจะเป็นข้อมูลโหมดความเร็วต่ำ ถ้าบิต = "0" จะไม่สร้างพรีแอมเบิล
6	Data Toggle Bit	"0" ถ้า DATA0, "1" ถ้า DATA1(ใช้สำหรับ OUT tokens ในโฮสต์โหมด)
5	Sync SOF	"1" ซิงโครไนซ์กับการส่งผ่าน SOF เมื่อการกระทำใน โหมดความเร็วสูง เท่านั้น
4	ISO	เมื่อ เซ็ต "1" อนุญาตทำงานในโหมดไอโซโครนัส สำหรับแพ็กเก็ตนี้
3	Reserved	บิต 3 สงวนไว้ใช้งานในอนาคต
2	Direction	เมื่อ = "1" ส่ง เมื่อ = "1" รับ
1	Enable	ถ้า Enable = "1" อนุญาตให้มีการส่งผ่านเกิดขึ้น ถ้า Enable = "1" การติดต่อระหว่าง USB ทำไม่ได้ บิต Enable จะใช้ร่วมกับ Arm bit (บิต0 ของรีจิสเตอร์นี้)
0	Arm	อนุญาตให้ส่งผ่านเมื่อเซ็ต "1" เคลียร์ 0 เมื่อส่งผ่านสมบูรณ์(เมื่อยืนยันการอินเตอร์รัปต์)

ตารางที่ 3.6 USB-A/USB-B Host Control Register Definition [Address 00h, 08h]

USB-A/USB-B Host Base Address [Address = 01h, 09h]

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
HBADD7	HBADD6	HBADD5	HBADD4	HBADD3	HBADD2	HBADD1	HBADD0

ตารางที่ 3.7 USB-A/USB-B Host Base Address Definition [Address 01h, 09h]

เป็นตำแหน่งแอดเดรสเมมโมรี่บัฟเฟอร์ ของ SL811HS สำหรับ ยูเอสบี อ่านและเขียน เมื่อมีการส่งผ่านข้อมูลออก

USB-A/USB-B Host Base Length [Address = 02h, 0Ah]

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
HBL7	HBL6	HBL5	HBL4	HBL3	HBL2	HBL1	HBL0

ตารางที่ 3.8 USB-A / USB-B Host Base Length Definition [Address 02h, 0Ah]

เป็นรีจิสเตอร์สำหรับจำกัดขนาดแพ็กเก็ตสูงสุดในการส่งผ่านระหว่าง SL811HSและอุปกรณ์ ยูเอสบี โดยพื้นฐานการออกแบบขนาดความยาวแพ็กเก็ตที่สามารถส่งผ่านได้ใน SL811HS จะออกแบบขนาดดาต้าแพ็กเก็ตที่ใช้ส่งและรับ ดังตัวอย่างต่อไปนี้ สำหรับความเร็วเต็มที่โหมดบัลก์ ขนาดความยาวแพ็กเก็ตสูงสุด 64 ไบต์โหมดไอเอสโอ ขนาดความยาวแพ็กเก็ตสูงสุด 1023 ไบต์ แต่ SL811HS ขนาดความยาว 8 บิตเท่านั้น ขนาดแพ็กเก็ตสูงสุดสำหรับ โหมดไอเอสไอ ใช้ไบต์ที่ 16 – 256 ของ SL811HS

USB-A/USB-B USB Packet Status (Read) and Host PID, Device Endpoint (Write) [Address = 03h, 0Bh]

รีจิสเตอร์นี้ใช้ได้ 2 โหมดที่ไม่อิสระต่อกันไม่ว่าเป็นการเขียนหรืออ่าน เมื่ออ่าน รีจิสเตอร์นี้จะจัดการสถานะและจำกัดขนาดข่าวสารและเปรียบเทียบจนแพ็กเก็ตสุดท้ายของการส่งและรับ รีจิสเตอร์นี้จะอ่านไม่สมบูรณ์จนกระทั่งหลังการอินเตอร์รัปต์สิ่งที่จะทำให้เกิดรีจิสเตอร์ที่ทันสมัยขึ้น จะกำหนดตามรีจิสเตอร์ดังต่อไปนี้

Bit Position	Bit Name	Function
7	STALL	ฝั่งอุปกรณ์จะส่ง STALL กลับมาเพื่อบอกว่าการส่งเกิดการผิดพลาด
6	NAK	ฝั่งอุปกรณ์จะส่ง STALL กลับมาเพื่อบอกว่าไมสามารถรับข้อมูลได้ชั่วขณะ
5	Overflow	ความยาวสูงสุดเกินกว่าเงื่อนไขที่จะรับได้
4	Setup	บิตนี้ไม่สามารถปรับใช้งานได้ สำหรับการทำงานของโฮสต์แต่ SETUP packet โฮสต์จะเป็นผู้สร้างขึ้นมา
3	Sequence	ลำดับ บิต "0" ถ้า DATA0 "1" ถ้า DATA1
2	Time-out	การเกิด Time-out, time-out กำหนดที่ 18-bit time โดยปราศจากการตอบสนองของอุปกรณ์(ในอุปกรณ์ความเร็วเต็มที่)
1	Error	ตรวจสอบความผิดพลาดในระหว่างส่งผ่าน จะประกอบด้วย CRC5, CRC16 และ PID errors
0	ACK	Transmission Acknowledge

ตารางที่ 3.9 USB-A/USB-B USB Packet Status Register Definition when READ [Address 03h, 0Bh]

PID [3:0]: 4-bit PID Field (See Table Below), EP [3:0]: 4-bit Endpoint Value in Binary.

PID TYPE	D7-D4
SETUP	1101 (D Hex)
IN	1001 (9 Hex)
OUT	0001 (1 Hex)
SOF	0101 (5 Hex)
PREAMBLE	1100 (C Hex)
NAK	1010 (A Hex)
STALL	1110 (E Hex)
DATA0	0011 (3 Hex)
DATA1	1011 (B Hex)

ตารางที่ 3.10 USB-A / USB-B Host PID and Endpoint Register when WRITTEN [Address 03h, 0Bh]

USB-A/USB-B Host Transfer Count Register (Read), USB Address (Write) [Address = 04h, 0Ch]

รีจิสเตอร์นี้มี 2 ฟังก์ชันที่แตกต่างกันที่ไม่อิสระต่อกันทั้งการอ่านและเขียน เมื่ออ่าน รีจิสเตอร์นี้บรรจุหมายเลขของไบต์ที่เหลืออยู่ (จากค่าความยาวพื้นฐานของโฮสต์) หลังจากแพ็กเก็ตนี้ส่งผ่านเสร็จ ตัวอย่างเช่น ถ้ารีจิสเตอร์เบสเลงค์ เซ็ตที่ 0x40 และไอเอ็น โทเคน ถูกส่งให้อุปกรณ์เสริม ถ้าหลังการส่งผ่านสมบูรณ์ค่าของโฮสต์ทรานเฟอร์ คือ 0x10 เพราะฉะนั้นจำนวนไบต์ที่ส่งผ่านสมบูรณ์คือ 0x30 นั้นสามารถบอกได้ว่าส่งได้ต่ำกว่า

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
HTC7	HTC6	HTC5	HTC4	HTC3	HTC2	HTC1	HTC0

ตารางที่ 3.11 USB-A / USB-B Host Transfer Count Register when READ [Address 04h, 0Ch]

Bit 7	Bit 6	Bit 5	Bit 4	Bit3	Bit 2	Bit 1	Bit 0
0	DA6	DA5	DA4	DA3	DA2	DA1	DA0

ตารางที่ 3.12 USB-A / USB-B USB Address when WRITTEN [Address 04h, 0Ch]

Bit Position	Bit Name	Function
7	Reserved	0
6	Suspend	"1" enable, "0" = disable.
5	USB Speed	"1" เซ็ตอัพสำหรับความเร็วเต็มที่ "0" เซ็ตอัพสำหรับความเร็วต่ำ
4	J-K state force	See the table below.
3	USB Engine Reset	รีเซ็ต USB = "1." ทำงานปกติเซ็ต "0".
2	Reserved	0
1	Reserved	0
0	SOF ena/dis	"1" = enable auto Hardware SOF generation; "0" = disable.

ตารางที่ 3.13 Control Register 1 [Address 05h]

Bit 4	Bit 3	Function
0	0	Normal operating mode
0	1	Force USB Reset, D+ and D– are set LOW (SE0)
1	0	Force J-State, D+ set HIGH, D– set LOW[2]
1	1	Force K-State, D– set HIGH, D+ set LOW[3]

Notes:
2. Force K-State for low speed.
3. Force J-State for low speed.

ตารางที่ 3.14 Control Register 1 Address 05h – Bits 3 and 4

อินเตอร์รัปต์อีนาเบิลรีจิสเตอร์ (Interrupt Enable Register [Address = 06h])

SL811HS ทำการร้องขอโดยการส่งอินเตอร์รัปต์ออกมา ซึ่งสามารถถูกกระตุ้นสำหรับเงื่อนไขจำนวนมากโดยใช้ อินเตอร์รัปต์อีนาเบิลรีจิสเตอร์

Bit Position	Bit Name	Function
7	Reserved	0
6	Device Detect / Resume	Enable Device Detect/Resume Interrupt. เมื่อบิต 6 ของรีจิสเตอร์05h เท่ากับ "1" บิต6 ของรีจิสเตอร์นี้จำทำให้ Resume Detect Interrupt. มิฉะนั้น, บิตนี้ถูกใช้เพื่อทำให้อุปกรณ์อยู่ในสถานะตรวจจับโดยการกำหนดใน อินเตอร์รัปต์สเตตัสรีจิสเตอร์
5	Inserted/Removed	อนุญาตให้อุปกรณ์ภายนอก Insert/Remove Detection ใช้ในการ enable/disable อุปกรณ์

ตารางที่ 3.15 อินเตอร์รัปต์อีนาเบิลรีจิสเตอร์ [Address 06h]

4	SOF Timer	1 = ทำให้อินเตอร์รัปต์ SOF Timer
3	Reserved	0
2	Reserved	0
1	USB-B DONE	USB-B Done Interrupt. (See USB-A Done interrupt).
0	USB-A DONE	USB-A Done Interrupt เมื่อทำอินเตอร์รัปต์จะส่งสัญญาณ triggered ออกมาที่บิตนี้

ตารางที่ 3.15 อินเตอร์รัปต์อีนาเบิลรีจิสเตอร์ [Address 06h](ต่อ)

อินเตอร์รัปต์สเตตัสรีจิสเตอร์(Interrupt Status Register, Address [Address = 0Dh])

รีจิสเตอร์อินเตอร์รัปต์สเตตัสรีจิสเตอร์ เป็นการอ่าน/เขียนเป็นการเตรียมการสำหรับอินเตอร์รัปต์ สามารถเคลียร์โดยการเขียนที่รีจิสเตอร์นี้ ถ้าเคลียร์อินเตอร์รัปต์โดยเจาะจงให้เซ็ต "1" เหมือนกันทุกบิต

Bit Position	Bit Name	Function
7	D+	ค่าของขา Data+ ถ้าอุปกรณ์ความเร็วต่ำบิตนี้ = "0", ถ้าอุปกรณ์ความเร็วเต็มที่บิตนี้ = "1"
6	Device Detect /Resume	Device Detect/Resume Interrupt. บิต 6 จะใช้ร่วมกันระหว่าง Device Detection status และ Resume detection interrupt. เมื่อบิต 6 ของรีจิสเตอร์ 05h เซ็ต 1 บิตนี้บิตอินเตอร์รัปต์จะเริ่มตรวจสอบอุปกรณ์ใหม่ มิฉะนั้นบิตของมันจะถูกใช้เพื่อแสดงการปรากฏตัวของอุปกรณ์ "1" = อุปกรณ์ "Not present" และ "0" = อุปกรณ์"Present"ในโหมดนี้บิทนี้ถูกตรวจสอบคล้ายกับบิต 5 เพื่อกำหนดไม่ว่าอุปกรณ์ได้ถูกใส่หรือเอาออก
5	Insert/Remove	Device Insert/Remove Detection. โหมด บิตนี้เซ็ต "1" เมื่อการส่งผ่านจาก SE0 to IDLE (device inserted) หรือจาก IDLE to SE0 (device removed) ที่เกิดขึ้นบนบัส
4	SOF timer	1 = Interrupt on SOF Timer.
3	Reserved	0
2	Reserved	0
1	USB-B	USB-B Done Interrupt. (ดูการอธิบายที่ [address 06h])
0	USB-A	USB-A Done Interrupt. (ดูการอธิบายที่ [address 06h])

ตารางที่ 3.16 อินเตอร์รัปต์สเตตัสรีจิสเตอร์ Interrupt Status Register [Address 0Dh]

Current Data Set Register/Hardware Revision/SOF Counter LOW [Address = 0Eh]

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Hardware Revision				Reserved			

Bit Position	Bit Name	Function
7-4	Hardware Revision	SL11H Read = 0H, SL811HS rev1.2 Read = 1H, SL811HS rev1.5 Read = 2.
3-2	Reserved	Read will be zero.
1-0	Reserved	Reserved for slave.

ตารางที่ 3.17 Hardware Revision when READ [Address 0Eh]

เมื่อเขียนรีจิสเตอร์นี้จะตั้งค่า กับอุปกรณ์ที่ต่ออยู่ทั้งหมด ตัวนับจะใช้สัญญาณคล็อก 12 เมกะเฮิร์ต และไม่ขึ้นอยู่กับความถี่คลิสดอล จะเซ็ตอัพในช่วงเวลา 1 มิลลิวินาที การติดตั้งซอฟต์แวร์ค่าจะต้องเหมาะสมกับเคาต์เตอร์รีจิสเตอร์

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0

ตารางที่ 3.18 SOF Counter LOW Address when WRITTEN [Address 0Eh]

SOF Counter HIGH/Control Register 2 [Address = 0Fh]

เมื่ออ่านรีจิสเตอร์นี้จะคืนค่าของตัวนับ เอสโอเอฟ ที่หารโดย 64 กลับมา ควรใช้ซอฟต์แวร์เพื่อกำหนดรีจิสเตอร์นี้ ในวิธีนี้ ผู้ใช้จะสามารถหลีกเลี่ยงเงื่อนไขที่ไม่จำเป็นเกี่ยวกับ ยูเอสบี ได้ ยกตัวอย่างเช่น เพื่อกำหนด แบนด์วิดธ์ในการส่งเฟรมออกจำนวนคล็อกสูงสุดในการTicks ใน 1200 เฟรม/มิลลิวินาที ค่าที่ได้หลังจากการอ่านในรีจิสเตอร์ 0FH คือ (Count *64) * 84 ns = เวลาที่เหลืออยู่ในเฟรม

Value of register 0FH Available bit times left are between

BBH 12000 bits to 11968 (187 × 64) bits : BAH 11968 bits to 11904 (186 × 64) bits

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C13	C12	C11	C10	C9	C8	C7	C6

ตารางที่ 3.19 SOF High Counter when READ [Address 0Fh]

Bit Position	Bit Name	Function
7	SL811HS Master/Slave selection	Master = 1, Slave = 0.
6	SL811HS D+/D– Data Polarity Swap	"1" = change polarity (low-speed) "0" = no change of polarity (full-speed)
5-0	SOF HIGH Counter Register	Write a value or read it back to SOF HIGH Counter Register.

ตารางที่ 3.20 Control Register 2 when WRITTEN [Address 0Fh]

บทที่ 4

การทดลองและผลการทดลอง

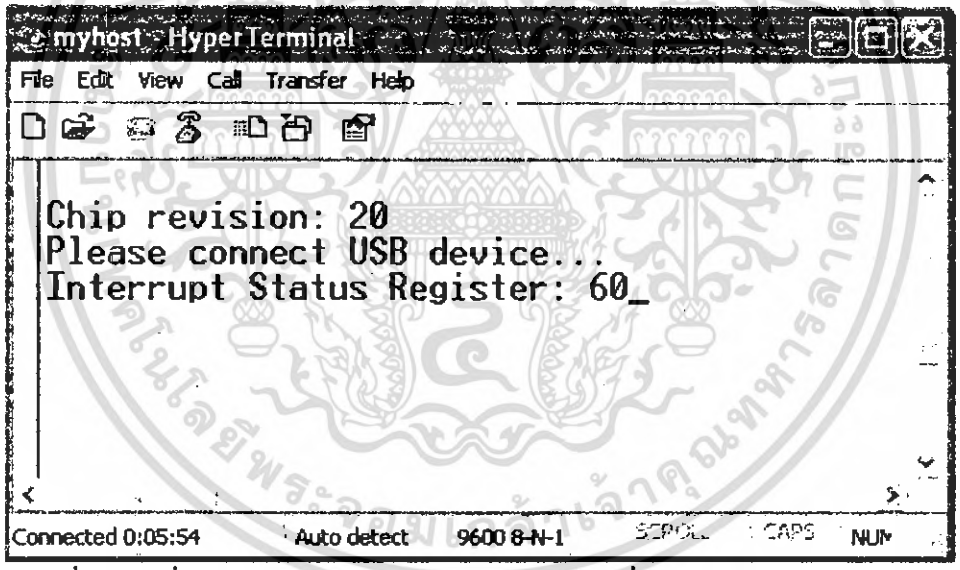
4.1 กล่าวนำ

บทนี้จะกล่าวถึงรายละเอียดผลการทำงานของชิ้นงาน โดยในขั้นแรกจะทำการทดสอบการทำงานของอุปกรณ์เบื้องต้นของโมดูล SL811HST โดยจะนำโมดูล SL811HST มาต่อเข้ากับไมโครคอนโทรลเลอร์เพื่อตรวจสอบการทำงานแล้วใช้โปรแกรมไฮเปอร์เทอร์มินอลของวินโดวส์แสดงผลเพื่อตรวจสอบข้อมูลเบื้องต้นของแต่ละรีจิสเตอร์รวมทั้งตรวจสอบสถานะการเชื่อมต่อของอุปกรณ์ยูเอสบีโดยผลที่แสดงออกมาจะอยู่ในรูปแบบของเลขฐานสิบหกขนาดหนึ่งไบต์

4.2 ทดลองการตรวจสอบการเชื่อมต่อและความเร็วของอุปกรณ์

เนื่องจากอุปกรณ์ยูเอสบีแบ่งออกเป็นอุปกรณ์ความเร็วเต็มที่และความเร็วต่ำถ้าส่งข้อมูลผิดความเร็วก็จะไม่สามารถสื่อสารกับอุปกรณ์นั้นๆ ได้ การตรวจสอบสถานะการเชื่อมต่อสามารถตรวจสอบได้โดยการอ่านค่าจาก รีจิสเตอร์ 0Dh (Interrupt Status Register) นอกจากนี้ยังสามารถอ่านค่าเวอร์ชันของชิปได้ด้วยเพื่อเป็นข้อพิจารณาในการนำไปใช้งาน

4.2.1 เมื่อไม่มีการต่ออุปกรณ์ยูเอสบี



รูปที่ 4.1 ค่าที่อ่านได้จากโปรแกรมไฮเปอร์เทอร์มินอล เมื่อไม่มีการต่ออุปกรณ์ยูเอสบี

โดยที่ Chip revision: 20 คือค่าที่อ่านได้จากรีจิสเตอร์ฮาร์ดแวร์รีวิชั่น (รีจิสเตอร์ 0Eh) ซึ่งค่าที่อ่านออกมาเท่ากับ 20h แสดงว่ายูเอสบีโฮสต์คอนโทรลเลอร์ที่ใช้เป็น SL811HST รีวิชั่น 1.5 ซึ่งได้รับการพัฒนามาจากรีวิชั่น 1.2 สามารถพิจารณาค่าที่อ่านออกมาได้ตังตารางที่ 4.1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	0	0	0	0	0
Hardware Revision				Reserved			

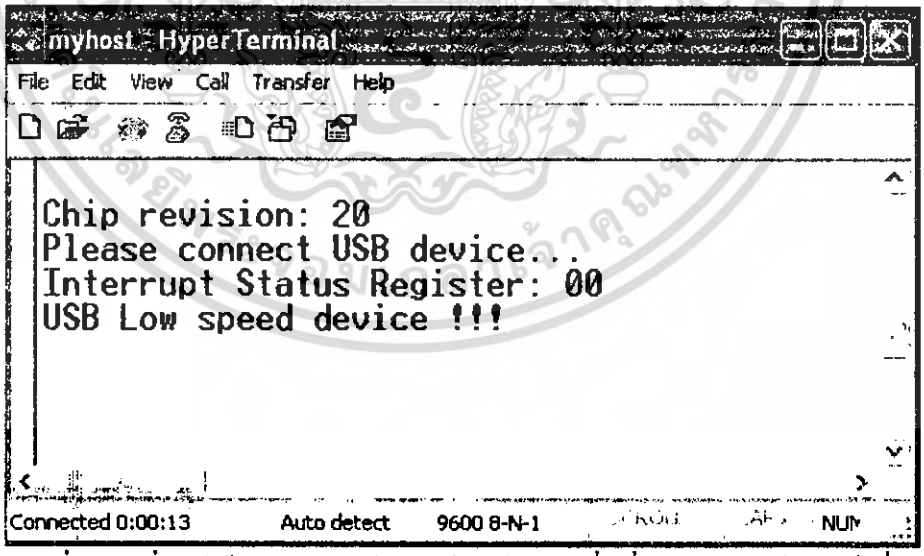
ตารางที่ 4.1 Hardware Revision when Read [Address 0EH]

และ Interrupt Status Register: 60 คือค่าที่อ่านได้จากรีจิสเตอร์ 0Dh ซึ่งค่าที่อ่านออกมาเท่ากับ 60h แสดงให้เห็นว่าบิต 6 มีค่าเท่ากับ 1 เป็นการแสดงให้เห็นว่าไม่มีการเชื่อมต่ออุปกรณ์ยูเอสบีอยู่บนบัส (Device "Not Present") สามารถพิจารณาค่าที่อ่านออกมาได้ดังตารางที่ 4.2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	1	0	0	0	0	0
D+	Device Detect/Resume	Insert/Remove	SOF timer	Reserved	Reserved	USB - B	USB - A

ตารางที่ 4.2 Interrupt Status Register, Address [Address 0DH]

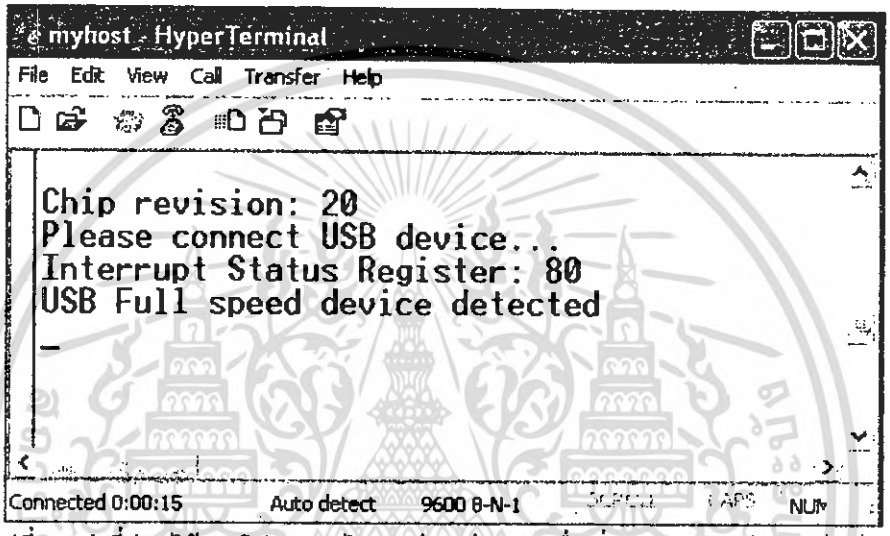
4.2.2 เมื่อมีการต่ออุปกรณ์ยูเอสบี ความเร็วต่ำ



รูปที่ 4.2 ค่าที่อ่านได้จากโปรแกรมไฮเปอร์เทอร์มินอล เมื่อเชื่อมต่ออุปกรณ์ความเร็วต่ำ

โดย Interrupt Status Register: 00 คือ ค่าที่อ่านได้จากรีจิสเตอร์ 0Dh ซึ่งค่าที่อ่านออกมาเท่ากับ 00h ซึ่งค่าที่อ่านออกมาเท่ากับ 80h โดยบิต 6 ของรีจิสเตอร์ 0Dh จะเป็น 0 เป็นการแสดงว่ามีการเชื่อมอยู่อุปกรณ์ ยูเอสบีอยู่บนบัส (Device "Present") และบิต 7 ของรีจิสเตอร์ 0Dh จะเป็น 1 ซึ่งเป็นการแสดงอุปกรณ์ที่เชื่อมอยู่บนบัสเป็นอุปกรณ์ความเร็วต่ำโดยที่มีอัตราการรับ – ส่งข้อมูล 1.5 เมกะบิตต่อวินาทีสามารถพิจารณาค่าที่อ่านออกมาได้ดังตารางที่ 4.2

4.2.3 เมื่อมีการต่ออุปกรณ์ยูเอสบี ความเร็วเต็มที่

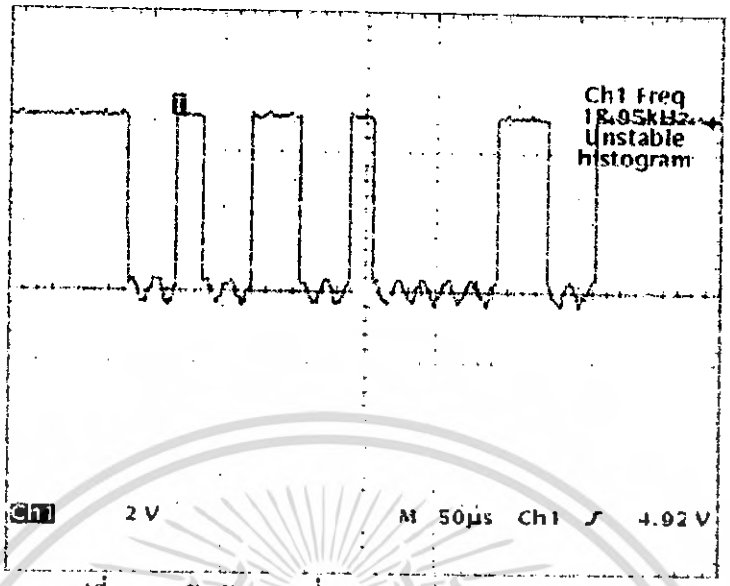


รูปที่ 4.3 ค่าที่อ่านได้จากโปรแกรมไฮเปอร์เทอร์มินอล เมื่อเชื่อมต่ออุปกรณ์ความเร็วเต็มที่

โดย Interrupt Status Register: 80 คือค่าที่อ่านได้จากรีจิสเตอร์ 0Dh ซึ่งค่าที่อ่านออกมาเท่ากับ 80h โดยที่ บิต 7 เป็น 1 แสดงว่ามีการต่ออุปกรณ์ยูเอสบีอยู่บนบัสเป็นอุปกรณ์ความเร็วเต็มที่ โดยที่มีอัตราการรับ - ส่งข้อมูล 12 เมกะบิตต่อวินาทีสามารถพิจารณาค่าที่อ่านออกมาได้ดังตารางที่ 4.2

4.3 ทดลองการวัดสัญญาณจาก พอร์ตอนุกรมของไมโครคอนโทรลเลอร์

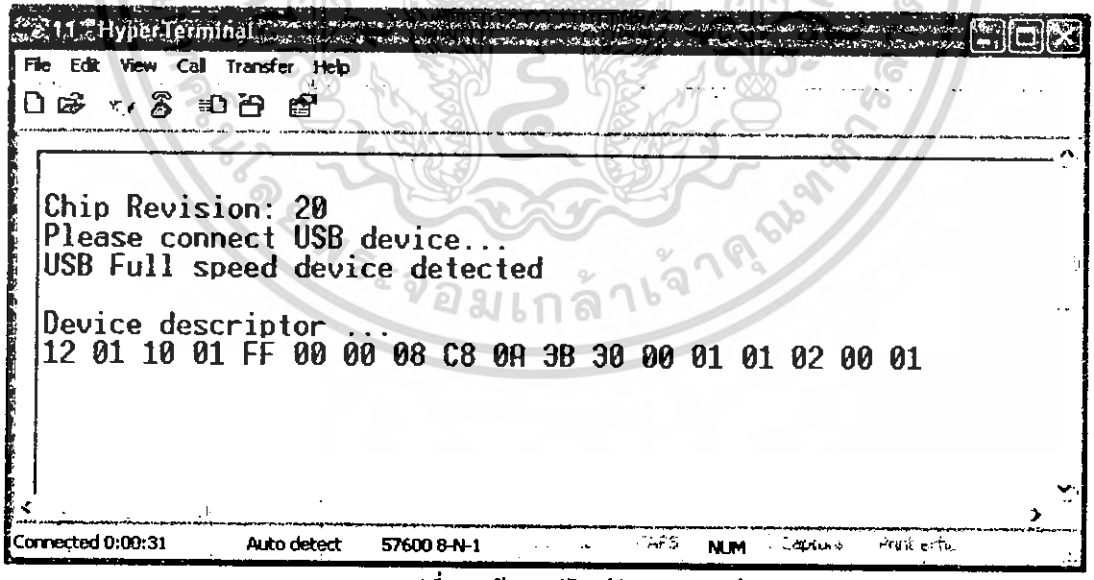
ส่วนนี้เป็นส่วนของสัญญาณที่วัดจากขา Tx ของไมโครคอนโทรลเลอร์ โดยค่าที่ส่งคือ 20 เมื่อแปลงเป็น ASSCII CODE จะได้32h และ30h ตามลำดับโดยการอ่านจะเริ่มจากด้านซ้ายก่อนโดยจะเรียงลำดับดังนี้ start bit 01001100 stop bit start bit 00001100stop bit โดย start bit="1" และ stop bit ="0" และจะเริ่มจากบิตล่างก่อนดังรูป



รูปที่ 4.4 ระดับสัญญาณที่ขา Tx ของไมโครคอนโทรลเลอร์

4.4 การร้องขอข้อมูลดีไวซ์ดีสคริปเตอร์ของ กล้องยูเอสบี

เมื่อเราต้องการร้องขอข้อมูลดีไวซ์ดีสคริปเตอร์ของ กล้องยูเอสบีโดยจะใช้คำร้องขอ GET_DESCRIPTOR(06h) โดยมีทิศทาง อุปกรณ์ไปยังโฮสต์(รายละเอียดการร้องขอในบทที่ 2) ซึ่งค่าดีไวซ์ดีสคริปเตอร์ที่กล้องยูเอสบีส่งกลับมามีด้วยกัน 18 ไบต์ดังนี้



รูปที่ 4.5 ข้อมูลดีไวซ์ดีสคริปเตอร์

Device Descriptor: มีรายละเอียดจะเรียงลำดับแต่ละไบต์ดังนี้

bLength 18

bDescriptorType 1

bcdUSB 1.10

bDeviceClass 255 //Vendor Specific Class

bDeviceSubClass 0

bDeviceProtocol 0

bMaxPacketSize0 8

idVendor 0x0ac8 //Z-Star Microelectronics Corp.

idProduct 0x303b

bcdDevice 1.00

iManufacturer 1 //Vimicro Corp.

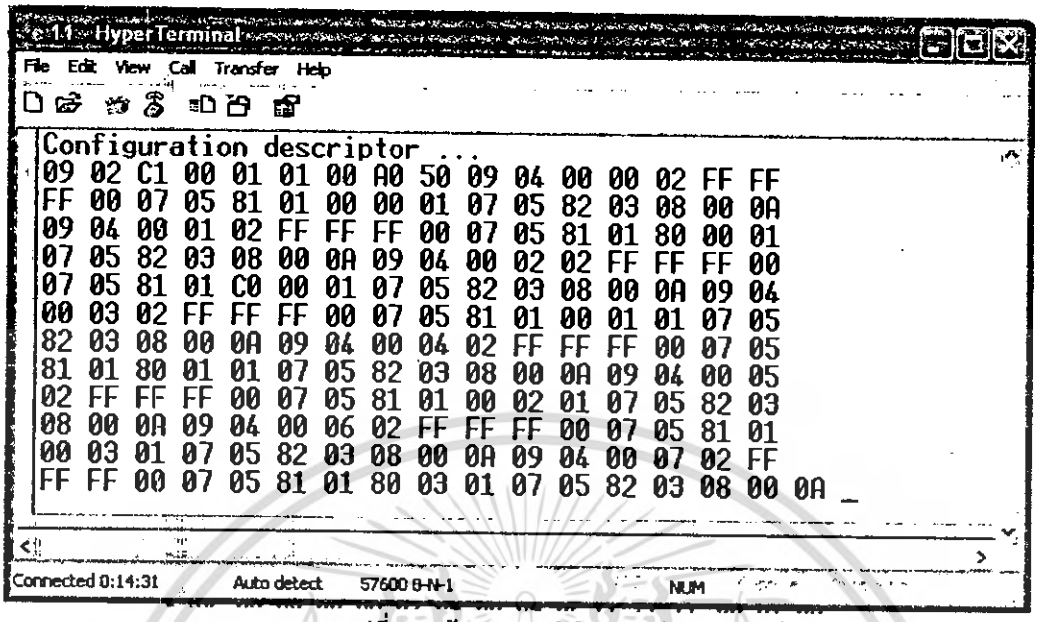
iProduct 2 //PC Camera

iSerial 0

bNumConfigurations 1

4.5 การร้องขอข้อมูลคอนฟิกิวเรชันดีสคริปเตอร์

เป็นดีสคริปเตอร์ที่อธิบายถึงความสามารถ และวิธีการใช้งานอุปกรณ์ยูเอสบี โดยอุปกรณ์ยูเอสบีทุกตัวจะมีลำดับชั้นของดีสคริปเตอร์ซึ่งเป็นตัวอธิบายข้อมูลให้แก่โฮสต์ เช่น อุปกรณ์ตัวนี้คืออุปกรณ์อะไร ใครเป็นผู้ผลิต และเวอร์ชันของอุปกรณ์ยูเอสบี ที่อุปกรณ์ตัวนี้สนับสนุน และการตั้งค่าต่าง ๆ ตั้งได้กี่ทาง จำนวนและชนิดของเอนด์พอยต์ที่ใช้ โดยมีข้อมูลและรายละเอียดดังนี้



รูปที่ 4. 6 ข้อมูลคอนฟิกิวเรชันดีสคริปเตอร์

โดยข้อมูลมี 193 ไบต์ ประกอบไปด้วย คอนฟิกิวเรชันดีสคริปเตอร์ อินเตอร์เฟสดีสคริปเตอร์ และ เอนด์พอยต์ดีสคริปเตอร์ โดยรายละเอียดต่าง ๆ ของดีสคริปเตอร์มีอธิบายไว้ในบทที่ 2 แต่ในที่นี้จะอธิบายคร่าว ๆ ยกตัวอย่าง 09 ไบต์แรก คือความยาวของ DescriptorType ไบต์ที่ สองคือ 02 บ่งบอกว่าเป็น Descriptor ชนิด Configuration Descriptor โดยเรียงตามลำดับไบต์ดังต่อไปนี้

1. **Configuration Descriptor(02h):**	2. **Interface Descriptor(04h):**
bLength 9	bLength 9
bDescriptorType 2	bDescriptorType 4
wTotalLength 193	bInterfaceNumber 0
bNumInterfaces 1	bAlternateSetting 0
bConfigurationValue 1	bNumEndpoints 2
iConfiguration 0	bInterfaceClass 255 Vendor Specific Class
bmAttributes 0x0A	bInterfaceSubClass 255 Vendor Specific Subclass
MaxPower 160mA	bInterfaceProtocol 255 Vendor Specific Protocol
	iInterface 0

3. Endpoint Descriptor(05h):	**6. Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x81 EP 1 IN	bEndpointAddress 0x81 EP 1 IN
bmAttributes 1	bmAttributes 1
Transfer Type Isochronous	Transfer Type Isochronous
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x0000 1x 0 bytes	wMaxPacketSize 0x0080 1x 128 bytes
bInterval 1	bInterval 1
4. Endpoint Descriptor:	**7. Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x82 EP 2 IN	bEndpointAddress 0x82 EP 2 IN
bmAttributes 3	bmAttributes 3
Transfer Type Interrupt	Transfer Type Interrupt
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x0008 1x 8 bytes	wMaxPacketSize 0x0008 1x 8 bytes
bInterval 10	bInterval 10
5. Interface Descriptor:	**8. Interface Descriptor:**
bLength 9	bLength 9
bDescriptorType 4	bDescriptorType 4
bInterfaceNumber 0	bInterfaceNumber 0
bAlternateSetting 1	bAlternateSetting 2
bNumEndpoints 2	bNumEndpoints 2
bInterfaceClass 255 Vendor Specific Class	bInterfaceClass 255 Vendor Specific Class
bInterfaceSubClass 255 Vendor Specific Subclass	bInterfaceSubClass 255 Vendor Specific Subclass
bInterfaceProtocol 255 Vendor Specific Protocol	bInterfaceProtocol 255 Vendor Specific Protocol
iInterface 0	iInterface 0

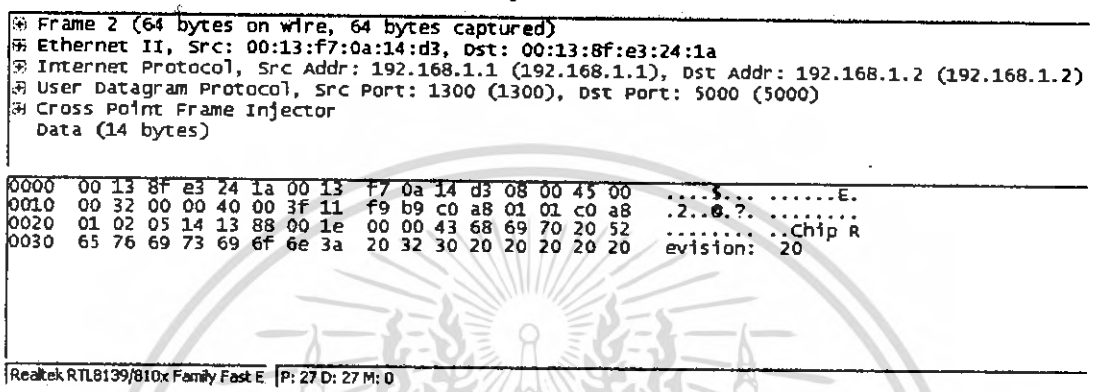
9. Endpoint Descriptor:	**12. Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x81 EP 1 IN	bEndpointAddress 0x81 EP 1 IN
bmAttributes 1	bmAttributes 1
Transfer Type Isochronous	Transfer Type Isochronous
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x00c0 1x 192 bytes	wMaxPacketSize 0x0100 1x 256 bytes
bInterval 1	bInterval 1
10. Endpoint Descriptor:	**13.Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x82 EP 2 IN	bEndpointAddress 0x82 EP 2 IN
bmAttributes 3	bmAttributes 3
Transfer Type Interrupt	Transfer Type Interrupt
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x0008 1x 8 bytes	wMaxPacketSize 0x0008 1x 8 bytes
bInterval 10	bInterval 10
11. Interface Descriptor:	**14. Interface Descriptor:**
bLength 9	bLength 9
bDescriptorType 4	bDescriptorType 4
bInterfaceNumber 0	bInterfaceNumber 0
bAlternateSetting 3	bAlternateSetting 4
bNumEndpoints 2	bNumEndpoints 2
bInterfaceClass 255 Vendor Specific Class	bInterfaceClass 255 Vendor Specific Class
bInterfaceSubClass 255 Vendor Specific Subclass	bInterfaceSubClass 255 Vendor Specific Subclass
bInterfaceProtocol 255 Vendor Specific Protocol	bInterfaceProtocol 255 Vendor Specific Protocol
iInterface 0	iInterface 0

15. Endpoint Descriptor:	**19. Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x81 EP 1 IN	bEndpointAddress 0x81 EP 1 IN
bmAttributes 1	bmAttributes 1
Transfer Type Isochronous	Transfer Type Isochronous
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x0180 1x 384 bytes	wMaxPacketSize 0x0200 1x 512 bytes
bInterval 1	bInterval 1
17. Endpoint Descriptor:	**20. Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x82 EP 2 IN	bEndpointAddress 0x82 EP 2 IN
bmAttributes 3	bmAttributes 3
Transfer Type Interrupt	Transfer Type Interrupt
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x0008 1x 8 bytes	wMaxPacketSize 0x0008 1x 8 bytes
bInterval 10	bInterval 10
18. Interface Descriptor:	**21. Interface Descriptor:**
bLength 9	bLength 9
bDescriptorType 4	bDescriptorType 4
bInterfaceNumber 0	bInterfaceNumber 0
bAlternateSetting 5	bAlternateSetting 6
bNumEndpoints 2	bNumEndpoints 2
bInterfaceClass 255 Vendor Specific Class	bInterfaceClass 255 Vendor Specific Class
bInterfaceSubClass 255 Vendor Specific Subclass	bInterfaceSubClass 255 Vendor Specific Subclass
bInterfaceProtocol 255 Vendor Specific Protocol	bInterfaceProtocol 255 Vendor Specific Protocol
iInterface 0	iInterface 0

22. Endpoint Descriptor:	**25. Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x81 EP 1 IN	bEndpointAddress 0x81 EP 1 IN
bmAttributes 1	bmAttributes 1
Transfer Type Isochronous	Transfer Type Isochronous
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x0300 1x 768 bytes	wMaxPacketSize 0x0380 1x 896 bytes
bInterval 1	bInterval 1
23. Endpoint Descriptor:	**26. Endpoint Descriptor:**
bLength 7	bLength 7
bDescriptorType 5	bDescriptorType 5
bEndpointAddress 0x82 EP 2 IN	bEndpointAddress 0x82 EP 2 IN
bmAttributes 3	bmAttributes 3
Transfer Type Interrupt	Transfer Type Interrupt
Synch Type None	Synch Type None
Usage Type Data	Usage Type Data
wMaxPacketSize 0x0008 1x 8 bytes	wMaxPacketSize 0x0008 1x 8 bytes
bInterval 10	bInterval 10
24. Interface Descriptor:	
bLength 9	
bDescriptorType 4	
bInterfaceNumber 0	
bAlternateSetting 7	
bNumEndpoints 2	
bInterfaceClass 255 Vendor Specific Class	
bInterfaceSubClass 255 Vendor Specific Subclass	
bInterfaceProtocol 255 Vendor Specific Protocol	
iInterface 0	

4.6 การส่งข้อมูลไปยังระบบ LAN

ส่วนนี้เป็นส่วนของข้อมูลที่รับมาจากโฮสต์ยูเอสบีที่ใช้ควบคุมกล้องยูเอสบีโดยใช้โปรแกรม Ethercal ในการดูข้อมูลจาก LAN โดยส่งจากMAC Address Embedded Device (00-13-f7-0a-14-d3) ไปยัง MAC Address Microcomputer Top (00-13-8F-E3-24-1A)ซึ่งค่าที่รับเข้ามาคือ Chip Revision: 20 โดยค่า 20 คือค่าที่บ่งบอกถึงเวอร์ชันของโฮสต์ยูเอสบีดังแสดงรูปด้านล่าง

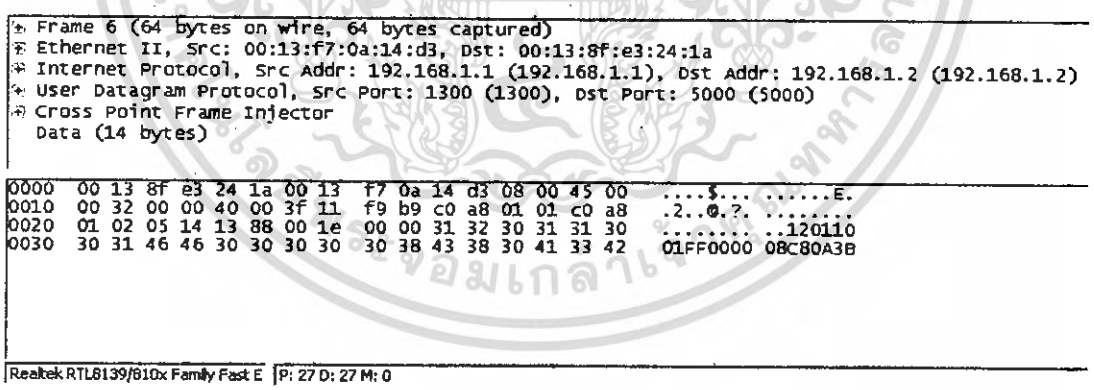


รูปที่ 4.7 ส่วนของข้อมูลที่รับมาจากโฮสต์ยูเอสบี

ส่วนนี้เป็นส่วนของข้อมูลที่รับมาจากกล้องยูเอสบีแล้วส่งต่อมายังระบบLANซึ่งข้อมูลนี้เป็นข้อมูลของดีไวซ์ดีสคริปเตอร์ 18 ไบต์ ดังนี้

Device descriptor

12 01 10 01 FF 00 00 08 C8 0A 3B 30 00 01 01 02 00 01



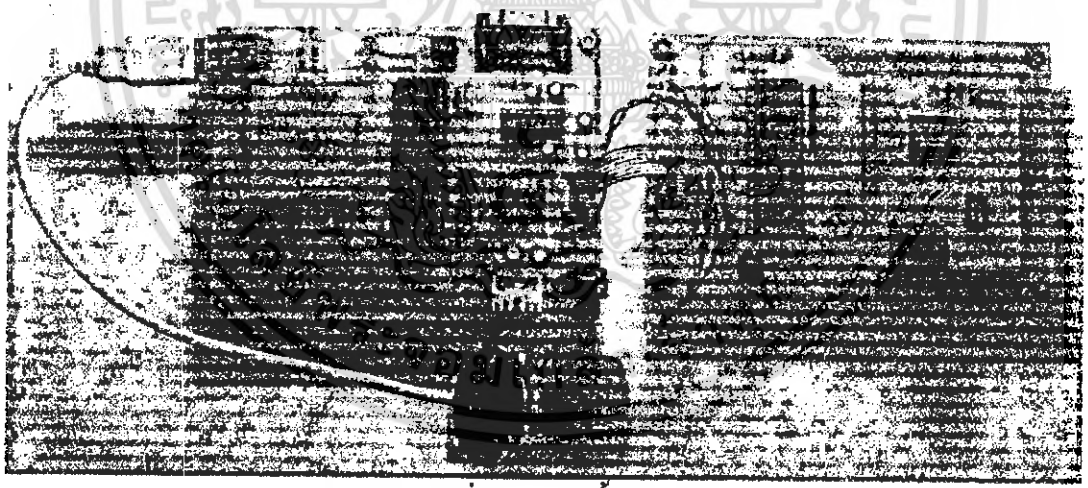
รูปที่ 4.8 ส่วนของข้อมูลที่รับมาจากกล้องยูเอสบี

```
Frame 7 (64 bytes on wire, 64 bytes captured)
Ethernet II, Src: 00:13:f7:0a:14:d3, Dst: 00:13:8f:e3:24:1a
Internet Protocol, Src Addr: 192.168.1.1 (192.168.1.1), Dst Addr: 192.168.1.2 (192.168.1.2)
User Datagram Protocol, Src Port: 1300 (1300), Dst Port: 5000 (5000)
Cross Point Frame Injector
Data (14 bytes)

0000  00 13 8f e3 24 1a 00 13  f7 0a 14 d3 08 00 45 00   ....$... ......E.
0010  00 32 00 00 40 00 3f 11  f9 b9 c0 a8 01 01 c0 a8   .2..@.?. ........
0020  01 02 05 14 13 88 00 1e  00 00 33 30 30 30 30 31   ........ ..300001
0030  30 31 30 32 30 30 30 31  20 20 20 20 20 20 20 20   01020001

Realtek RTL8139/810x Family Fast E  P: 28 D: 28 M: 0
```

รูปที่ 4.8 ส่วนของข้อมูลที่รับมาจากกล้องยูเอสบี(ต่อ)

4.7 การติดตั้งและการนำไปใช้งาน

โดยเริ่มจาก อีเดอร์เน็ตคาเมล่าโมดูล (ซ้ายมือ)ทำหน้าที่ควบคุมการทำงานของกล้องยูเอสบี โดยโมดูลตัวนี้จะมีส่วนที่ทำหน้าที่เป็น ยูเอสบีโฮสต์คอนโทรลเลอร์เพื่อเป็นตัวจัดการให้รูปแบบการรับส่งข้อมูลเป็นไปตามมาตรฐานการติดต่อแบบ ยูเอสบี เพื่อที่จะสามารถติดต่อกับอุปกรณ์ ยูเอสบี ได้ จากนั้นทำการส่งข้อมูลผ่านพอร์ตอนุกรมของไมโครคอนโทรลเลอร์เพื่อส่งไปยังโมดูลของLAN (ซ้ายมือ)เพื่อจัดการข้อมูลให้เป็นไปตามมาตรฐานของระบบเครือข่ายท้องถิ่นเพื่อส่งข้อมูลภาพผ่านระบบเครือข่ายท้องถิ่นตามที่เราต้องการ



รูปที่ 4.9 การติดตั้งอุปกรณ์

บทที่ 5
บทวิจารณ์และสรุปผล

5.1 สรุปผลของการดำเนินโครงงาน

ในส่วนของโฮสต์ยูเอสบีได้นำชิปเบอร์ SL811HST มาต่อร่วมกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 เพื่อทำเป็นหน้าที่เป็นโฮสต์ยูเอสบี ควบคุมการทำงานของกล้องยูเอสบีโดยการทำงานจะต้องมีซอฟต์แวร์เพื่อบรรจุลงในไมโครคอนโทรลเลอร์ที่เรียกว่าเฟิร์มแวร์ ในการควบคุมการทำงานซึ่งตอนนี้สามารถร้องขอข้อมูลดีไวซ์ดีสคริปเตอร์ของ กล้องยูเอสบี และข้อมูลคอนฟิกิวเรชันดีสคริปเตอร์ เพื่อใช้ในการเขียนไดร์ฟเวอร์ให้กับอุปกรณ์ เพื่อร้องขอข้อมูลภาพต่อไป

ในส่วนของเครือข่าย LAN สามารถทำงานรับส่งข้อมูลได้แล้วพร้อมที่จะรองรับข้อมูลภาพจากโฮสต์ยูเอสบีที่ส่งข้อมูลภาพที่มาจากกล้องยูเอสบี

5.2 ปัญหาที่พบในการทดลอง

กล้องยูเอสบีจะทำงานได้จะต้องอาศัยไดร์ฟเวอร์ในการควบคุมการทำงานซึ่งโครงงานนี้จะเขียนอยู่ในรูปของเฟิร์มแวร์ ซึ่งยังเป็นปัญหามากเพราะเราต้องศึกษาข้อมูลในส่วนฮาร์ดแวร์ของกล้องยูเอสบีด้วยเพื่อใช้ในการเขียนไดร์ฟเวอร์กล้องยูเอสบี

หนังสืออ้างอิง

[1] ร.ศ.สมยศ จุณณะปิยะ "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์"ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2546

[2] ลภน สุภาพ, อรรถพล บุญยะโภคา, วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์และอุปกรณ์ภายนอกผ่านพอร์ต USB ขั้นพื้นฐาน", 2546

[3] ลัญจกร วุฒิสิทธิกุลกิจ "โครงข่าย Internet และโปรโตคอล TCP/IP" สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2545

ภาคผนวก

Source Code Ethernet Module

```
//File main.c
#include <reg51.h>
#include <io.h>
#include <initial.h>
#include <arp.h>
#include <UART.h>
void txudp();
unsigned char Current_Data,p,n;
xdata unsigned char  Uart_Buffer[128];
void main()
{           Red = on;
            int_sio_interrupt();
             n= 0;
             p= 0;
            initial8900A();
            txarp();
            while(1)
                  {
                        Red = off;
                        if(p==1)
                        {
                              Uart_Buffer[n]=Current_Data;
                              n++;
                              p=0;
                        }
                        if(n==22)
                        {     Green = on;
                              txudp();
                              n=0;
                              Green = off;
                        }
                  }

      while(1);
}

void serial_interrupt(void)interrupt 4

{

  if (RI == 1)
  {
            RI=0;
            p=1;
            Red = on;
            Current_Data= SBUF;
  }

  if (TI == 1);

      {
            TI=0;

      }
}
```

```
void txudp()
{
      ////////////////////////
      // Transmit  Datagram //
      ////////////////////////


      // Send Transmit Command (Data Sheet P.70 Setbit 7,6)
            ioWrite(TxCmd,          0xc0);
            ioWrite(TxCmd + 1,      0x00);

      // 64 bytes to be sent
      // Ethernet Header(14)+IP Header(20) + UDP Header(8) + Data(22)
            ioWrite(TxLength,   0x40);
            ioWrite(TxLength+1, 0x00);

      // Check Bus status (Data Sheet P.67)
            ioWrite(PPPtr,     0x38);
            ioWrite(PPPtr + 1, 0x01);

      // Check Bus status,Is ready for transmit (Check bit 7)
            do {
                  BusST0 = ioRead(PPData);
                  BusST1 = ioRead(PPData+1);
            }while(!(BusST1==0x01));

      //--Ready to Transmit,Transmit Datagram in RxTxData Port--//

      ////////////////////////
      //  Ethernet Header  // (14 bytes)
      ////////////////////////

    // Send Destination (6 bytes)
      // MAC Address Microcomputer Top (00-13-8F-E3-24-1A)
            ioWrite(RxTxData,       0x00);
            ioWrite(RxTxData+1, 0x13);
            ioWrite(RxTxData, 0x8F);
            ioWrite(RxTxData+1, 0xE3);
            ioWrite(RxTxData,       0x24);
            ioWrite(RxTxData+1,     0x1A);
      // Send Source (6 bytes)
      // MAC Address Embedded Device (00-13-f7-0a-14-d3)
            ioWrite(RxTxData,       0x00);
            ioWrite(RxTxData+1,     0x13);
            ioWrite(RxTxData,       0xf7);
            ioWrite(RxTxData+1, 0x0a);
            ioWrite(RxTxData,       0x14);
            ioWrite(RxTxData+1,     0xd3);


      // Send Protocol Type (2 bytes)
      // IP=0x0800
            ioWrite(RxTxData,       0x08);
            ioWrite(RxTxData+1, 0x00);
```

```
//////////////////
//  IP Header  // (20 bytes)
//////////////////

// Version Header Length (1 byte)
// Using IPv4
      ioWrite(RxTxData,       0x45);

// Service (1 byte)
// Normally set to zero because not used
      ioWrite(RxTxData+1, 0x00);

// Length (2 bytes)
// 92 byte Length (IP Header(20) + UDP Header(8) + Data(22))
      ioWrite(RxTxData,       0x00);
      ioWrite(RxTxData+1, 0x32);

// Identifier (2 bytes)
// 0x0000
      ioWrite(RxTxData,       0x00);
      ioWrite(RxTxData+1, 0x00);

// Flags Fragment offset (2 bytes)(no flag)
// 0x4000
      ioWrite(RxTxData, 0x40);
      ioWrite(RxTxData+1, 0x00);

// Time to Live (1 byte)
// 63
      ioWrite(RxTxData,       0x3f);

// Protocol (1 byte)
// UDP 17 = 0x11
      ioWrite(RxTxData+1, 0x11);

// Checksum IP Header (2 bytes)
// 0x7431
      ioWrite(RxTxData,       0xF9);
      ioWrite(RxTxData+1, 0xB9);

      // Source Address (4 bytes)
// IP address 192.168.1.1 (Embedded Device)
      ioWrite(RxTxData,       0xc0);      // 192
      ioWrite(RxTxData+1, 0xa8);          // 168
      ioWrite(RxTxData,       0x01);      // 1
      ioWrite(RxTxData+1, 0x01);          // 1

// Destination Address (4 bytes)
// IP address 192.168.1.2 (Labtop)
      ioWrite(RxTxData,       0xc0);      // 192
      ioWrite(RxTxData+1, 0xa8);          // 168
      ioWrite(RxTxData,       0x01);      // 1
      ioWrite(RxTxData+1, 0x02);          // 2
```

```
        //////////////////
        //  UDP Header // (8 bytes)
        //////////////////

        // Source Port (2 bytes)
        // Port 1300 (0x0514)
              ioWrite(RxTxData,       0x05);
              ioWrite(RxTxData+1, 0x14);

        // Destination Port (2 bytes)
        // Port 5000 (0x1388)
              ioWrite(RxTxData,       0x13);
              ioWrite(RxTxData+1, 0x88);

        // Message Length (2 bytes)
        // 30 bytes (UDP Header(8) + Data(22))
              ioWrite(RxTxData,       0x00);
              ioWrite(RxTxData+1, 0x1e);

        // Checksum (2 bytes)
        // 0x0000 (no checksum used)
              ioWrite(RxTxData,       0x00);
              ioWrite(RxTxData+1, 0x00);

        ////////////
        //  Data  // (22 bytes)
        ////////////

        // Write Message (22 bytes)

        k=0;
        for (i=1;i<=22;i++)
              {
                    ioWrite(RxTxData + k,Uart_Buffer[i-1]);
                    if(i%2==1)
                          k=1;

                    else{
                          k=0;
                    }
              }
}
```

```
//File io.h
#define addr P2
#define data P0
#define on  0
#define off 1

#define RxTxData  0x00        // Receive/Transmit data (port 0)
#define RxTxData1 0x02        // Receive/Transmit data (port 1)
#define TxCmd           0x04        // Transmit Command
#define TxLength  0x06        // Transmit Length
#define ISQ           0x08        // Interrupt status queue
#define PPPtr           0x0A        // PacketPage pointer
#define PPData          0x0C        // PacketPage data (port 0)
#define PPData1         0x0E        // PacketPage data (port 1)
void delay_ms(unsigned char round);
sbit write  = P3^7;           // IOW active low
sbit read   = P3^6;           // IOR active low

sbit Green  = P1^0;
sbit Red    = P1^1;


//------DELAY1ms------//
void delay_ms(unsigned char round){
      unsigned char X;
            TMOD = (TMOD|0x01);
            IE     = (IE|0x80);

            for (X=0; X<round; X++){
                  TH0 = 0xFC;
                  TL0 = 0x66;
                  TF0 = 0;
                  TR0 = 1;
                  while(TF0 == 0);
                  TR0 = 0;
            }
      }

//--------------------//
//--Function ioRead Used to Read from the Data Bus--//
      unsigned char ioRead(unsigned char address){
            unsigned char value;
            addr  = (address&0x0F);
            read = 0;
            value = data;
            read = 1;
            return value;     // Return to ioread
      }
//--Funtcion ioWrite Used to write to the Data Bus--//
      void ioWrite(unsigned char address, unsigned char value){
            data  = value;
            addr  = (address&0x0F);
            write = 0;
            write = 1;
      }
```

```
//File initial.h
void initial();
      ////////////////////////
      // Initialize Registers //
      ////////////////////////
void initial8900A(){
    //------LED Active------//
            Green  = off;
            Red    = off;

      //------Reset Chip------//
      // (1) Write 0x0114 to PacketPage Pointer (Data Sheet P.64)
      // (2) Write 0x0040 to PacketPage Data Port (Set bit 6)
            ioWrite(PPPtr,      0x14);
            ioWrite(PPPtr + 1,  0x01);
            ioWrite(PPData,     0x40);
            ioWrite(PPData + 1, 0x00);

      // Delay time about 125 ms for chip resetting in progress
            delay_ms(126);

      //---Configure Receiver Control fo Promiscious mode, RxOK---//
      // (1) Write 0x0104 to PacketPage Pointer (Data Sheet P.54)
      // (2) Write 0x0180 to PacketPage Data Port (Set bit 7,8)
            ioWrite(PPPtr,      0x04);
            ioWrite(PPPtr + 1,  0x01);
            ioWrite(PPData,     0x00);
            ioWrite(PPData + 1, 0x2d);

      //--Set 10BaseT, SerRxOn, SerTxOn in Line Control--//
      // (1) Write 0x0112 to PacketPage Pointer (Data Sheet P.62)
      // (2) Write 0x00c0 to PacketPage Data Port (Set bit 6,7)
            ioWrite(PPPtr,      0x12);
            ioWrite(PPPtr + 1,  0x01);
            ioWrite(PPData,     0xc0);
            ioWrite(PPData + 1, 0x00);

      //Module Embedded Ethernet MAC Adrress (00-13-f7-0a-14-d3)//
      //Write MAC(IEEE) Address (Data Sheet P.71)

            ioWrite(PPPtr,      0x58);
            ioWrite(PPPtr + 1,  0x01);
            ioWrite(PPData,     0x00);
            ioWrite(PPData+1 ,  0x13);

            ioWrite(PPPtr,     0x5A);
            ioWrite(PPPtr + 1,0x01);
            ioWrite(PPData,    0xf7);
            ioWrite(PPData+1 ,0x0a);

            ioWrite(PPPtr,     0x5C);
            ioWrite(PPPtr + 1,0x01);
            ioWrite(PPData,    0x14);
            ioWrite(PPData+1 ,0xd3);
}
```

```
//File arp.h
unsigned char BusST0,BusST1;
unsigned int Status,Length;
unsigned char idata RxBuffer[100];
unsigned char DataL,DataH,i,j,k,l,buff,temp;
unsigned int x,y,z;
unsigned char Rs232Buffer(unsigned char key);

void txarp()
{
      ////////////////////////
      // Transmit  Datagram //
      ////////////////////////

      // Send Transmit Command (Data Sheet P.70 Setbit 7,6)
            ioWrite(TxCmd,          0xc0);
            ioWrite(TxCmd + 1,      0x00);

      // 42 bytes to be sent
      // Ethernet Header(14) +  ARP Header(28)
            ioWrite(TxLength,   0x2a);
            ioWrite(TxLength+1, 0x00);

      // Check Bus status (Data Sheet P.67)
            ioWrite(PPPtr,     0x38);
            ioWrite(PPPtr + 1, 0x01);

      // Check Bus status,Is ready for transmit (Check bit 7)
            do {
                  BusST0 = ioRead(PPData);
                  BusST1 = ioRead(PPData+1);
            }while(!(BusST1==0x01));

      //--Ready to Transmit,Transmit Datagram in RxTxData Port--//

      ////////////////////////
      //  Ethernet Header  // (14 bytes)
      ////////////////////////

      // Send Destination (6 bytes)
      // MAC Address Microcomputer Top (00-13-8F-E3-24-1A)
            ioWrite(RxTxData,       0x00);
            ioWrite(RxTxData+1, 0x13);
            ioWrite(RxTxData, 0x8F);
            ioWrite(RxTxData+1, 0xE3);
            ioWrite(RxTxData,       0x24);
            ioWrite(RxTxData+1,     0x1A);
      // Send Source (6 bytes)
      // MAC Address Embedded Device (00-13-f7-0a-14-d3)
            ioWrite(RxTxData,       0x00);
            ioWrite(RxTxData+1,     0x13);
            ioWrite(RxTxData,       0xf7);
            ioWrite(RxTxData+1, 0x0a);
            ioWrite(RxTxData,       0x14);
            ioWrite(RxTxData+1,     0xd3);

      // Send Protocol Type (2 bytes)
      // ARP=0x0806
            ioWrite(RxTxData,       0x08);
            ioWrite(RxTxData+1, 0x06);  //byte 14
```

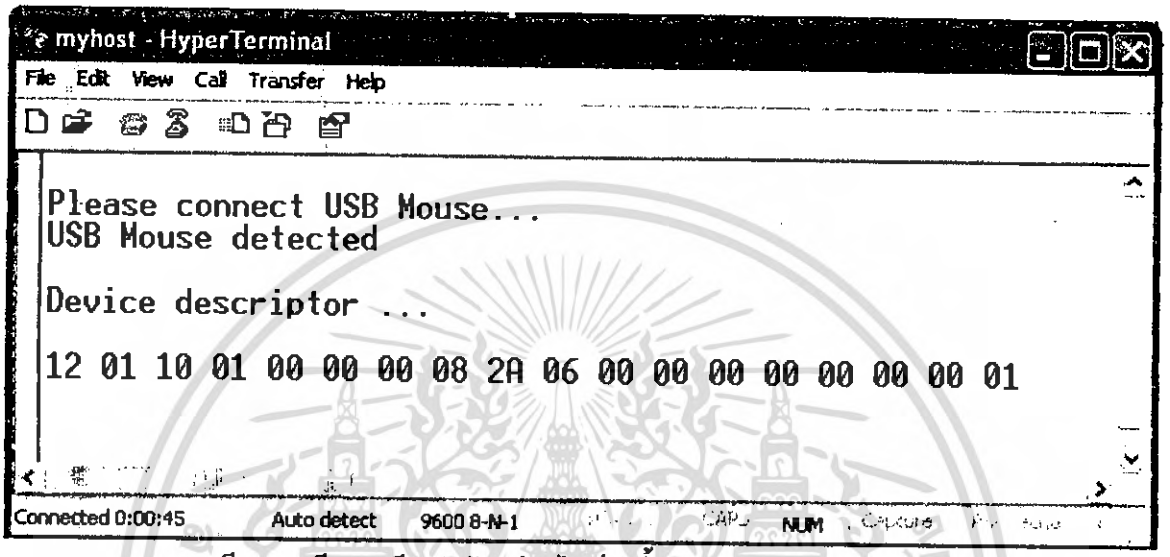
```
        /////////////////
        //  ARP Header  // (28 bytes)
        /////////////////

        // Hardware Type (2 byte)
        // Ethernet
              ioWrite(RxTxData,       0x00);
              ioWrite(RxTxData+1, 0x01);

        // Protocol type (2 bytes)
              ioWrite(RxTxData,       0x08);
              ioWrite(RxTxData+1, 0x00);

        // Hardware size (1 bytes)
              ioWrite(RxTxData,       0x06);
        //Protocol size (1 byte)
              ioWrite(RxTxData+1, 0x04);

        // Opcode (2 bytes)
        // reply 02
              ioWrite(RxTxData, 0x00);
              ioWrite(RxTxData+1, 0x02);  //byte 22
        //sender MAC hardware (6bytes) (Embedded Device (00-13-f7-0a-
14-d3))
              ioWrite(RxTxData,       0x00);
              ioWrite(RxTxData+1,     0x13);
              ioWrite(RxTxData,       0xf7);
              ioWrite(RxTxData+1, 0x0a);
              ioWrite(RxTxData,       0x14);
              ioWrite(RxTxData+1,     0xd3);
        // Source Address (4 bytes)
        // IP address 192.168.1.1 (Embedded Device)
              ioWrite(RxTxData,       0xc0);      // 192
              ioWrite(RxTxData+1, 0xa8);          // 168
              ioWrite(RxTxData,       0x01);      // 1
              ioWrite(RxTxData+1, 0x01);          // 1
        //Dest MAC computer (6bytes) Top (00-13-8F-E3-24-1A)
              ioWrite(RxTxData,       0x00);
              ioWrite(RxTxData+1, 0x13);
              ioWrite(RxTxData, 0x8F);
              ioWrite(RxTxData+1, 0xE3);
              ioWrite(RxTxData,       0x24);
              ioWrite(RxTxData+1,     0x1A);
        // Destination Address (4 bytes)
        // IP address 162.254.254.2 (Labtop)
              ioWrite(RxTxData,       0xc0);      // 192
              ioWrite(RxTxData+1, 0xa8);          // 168
              ioWrite(RxTxData,       0x01);      // 1
              ioWrite(RxTxData+1, 0x02);          // 2

}
```

```
//File uart.h
void int_sio_interrupt(void);
unsigned char Current_Data;
void init_232(void);
void int_sio_interrupt(void)

{
      TR1 =0;
      TMOD=0x21;
      TH1=0xFF;
      TR1=1;
      SCON=0x50;
      PCON|= 0x80;
      RI = 0;
      TI = 0;
      EA = 1;
      ES = 1;
}

//=============================================================
//                  RS232 UART FUNCTIONS
//=============================================================
void init_232(void)
{
      TR1 = 0;
      ET1 = 0;
      TMOD= 0X21;//T1 mode2
      TH1   = 0xFF;//115200
      TR1   = 1;//TH1=256-((f*k)/(384*baud))
      SCON= 0X50;
      PCON|= 0x80;
      TI    = 1;
}
```

ผลการทดลองเพิ่มเติม

การร้องขอข้อมูลดีไวซ์ดีสกริปเตอร์ของเมาส์

เมื่อเราต้องการร้องขอข้อมูลดีไวซ์ดีสกริปเตอร์ของ เมาส์โดยจะใช้คำร้องขอ GET_DESCRIPTOR(06h) โดยมีทิศทาง อุปกรณ์ไปยังโฮสต์ซึ่งค่า ดีไวซ์ดีสกริปเตอร์เมาส์ส่งกลับมามีด้วยกัน 18 ไบต์ดังนี้



Device Descriptor: มีรายละเอียดจะเรียงลำดับแต่ละไบต์ดังนี้

bLength 18

bDescriptorType 1

bcdUSB 1.10

bDeviceClass 0

bDeviceSubClass 0

bDeviceProtocol 0

bMaxPacketSize0 8

idVendor 0x062A

idProduct 0x0000

bcdDevice 0

iManufacturer0

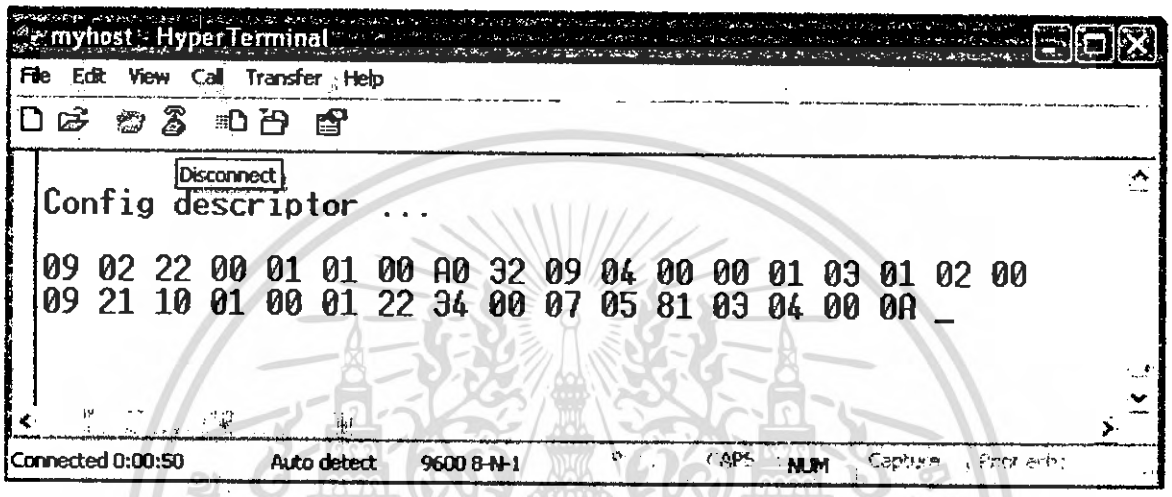
iProduct 0

iSerial 0

bNumConfigurations 1

การร้องขอข้อมูลคอนฟิกิวเรชันดีสคริปเตอร์

เป็นดีสคริปเตอร์ที่อธิบายถึงความสามารถ และวิธีการใช้งานอุปกรณ์ยูเอสบี โดยอุปกรณ์ยูเอสบีทุกตัวจะมีลำดับชั้นของดีสคริปเตอร์ซึ่งเป็นตัวอธิบายข้อมูลให้แก่โฮสต์ เช่น อุปกรณ์ตัวนี้คืออุปกรณ์อะไร ใครเป็นผู้ผลิต และเวอร์ชั่นของอุปกรณ์ยูเอสบี ที่อุปกรณ์ตัวนี้สนับสนุน และการตั้งค่าต่าง ๆ ตั้งได้กี่ทาง จำนวนและชนิดของเอนด์พอยต์ที่ใช้ และรวมทั้ง HID ดีสคริปเตอร์โดยมีข้อมูลและรายละเอียดดังนี้



Interface Descriptor (Mouse)

Part	Offset/Size (Bytes)	Description	Sample Value
bLength	0/1	Size of this descriptor in bytes.	0x09
bDescriptorType	1/1	Interface descriptor type	0x04
bInterfaceNumber	2/1	Number of interface.	0x00
bAlternateSetting	3/1	Value used to select alternate setting.	0x00
bNumEndpoints	4/1	Number of endpoints.	0x01
bInterfaceClass	5/1	Class code .	0x03
bInterfaceSubClass	6/1	1 = Boot Interface subclass.	0x01
bInterfaceProtocol	7/1	2 = Mouse.	0x02
iInterface	8/1	Index of string descriptor.	0x00

HID Descriptor (Mouse)

Part	Offset/Size(Bytes)	Description	Sample Value
bLength	0/1	Size of this descriptor in bytes.	0x09
bDescriptorType	1/1	HID descriptor type.	0x21
bcdHID	2/2	HID Class Specification release number.	0x101
bCountryCode	4/1	Hardware target country.	0x00
bNumDescriptors	5/1	Number of HID class descriptors to follow.	0x01
bDescriptorType	6/1	Report descriptor type.	0x22
wItemLength	7/2	Total length of Report descriptor.	0x34

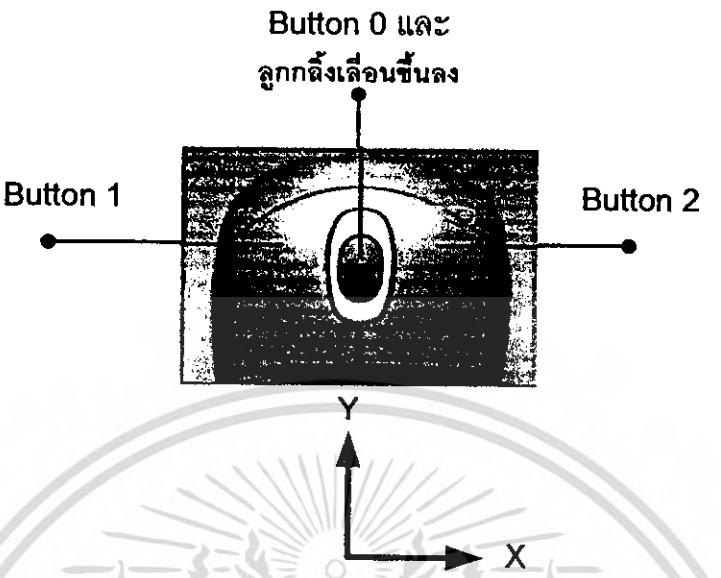
Endpoint Descriptor (Mouse)

Part	Offset/Size(Bytes)	Description	Sample Value
bLength	0/1	Size of this descriptor in bytes.	0x07
bDescriptorType	1/1	Endpoint descriptor type.	0x05
bEndpointAddress	2/1	The address of the endpoint.	10000001B
bmAttributes	3/1	This field describes the endpoint's attributes.	00000011B
wMaxPackctSize	4/2	Maximum packet size.	0x0004
bInterval	6/1	Interval for polling endpoint for data transfers.	0x0A

Report Descriptor (Mouse)

Item		Value (Hex)
Usage Page (Generic Desktop),		05 01
Usage (Mouse),		09 02
Collection (Application),		A1 01
Usage (Pointer),		09 01
Collection (Physical),		A1 00
Usage Page (Buttons),		05 09
Usage Minimum (01),		19 01
Usage Maximun (03),		29 03
Logical Minimum (0),		15 00
Logical Maximum (1),		25 01
Report Count (3),		95 03
Report Size (1),		75 01
Input (Data, Variable, Absolute), ;3 button bits		81 02
Report Count (1),		95 01
Report Size (5),		75 05
Input (Constant), ;5 bit padding		81 01
Usage Page (Generic Desktop),		05 01
Usage (X),		09 30
Usage (Y),		09 31
Usage (Y),		09 38
Logical Minimum (-127),		15 81
Logical Maximum (127),		25 7F
Report Size (8),		75 08
Report Count (3),		95 03
Input (Data, Variable, Relative), ;	2 position bytes (X & Y)	81 06
End Collection,		C0
End Collection		C0

ทดลองการใช้เมาส์



- เมื่อกด Button 1 , Button 2 และเลื่อนลูกกลิ้งขึ้นหรือลง

```
01 00 00 00
00 00 00 00
02 00 00 00
00 00 00 00
02 00 00 00
00 00 00 00
00 00 00 01
00 00 00 01
00 00 00 01
00 00 00 FF
00 00 00 FF
00 00 00 FF
```

โดยคอลัมน์ แรกแสดงการกด Button 1, Button 2 ถ้ากดButton 1 จะส่งค่า 01 ออกมา แต่ถ้ากด Button 2 จะส่งค่า 02 ออกมา และถ้าเลื่อนลูกกลิ้งขึ้นลง ตั้งแต่ 1-FF ดังแสดงตามคอลัมที่ 4

- เมื่อเลื่อนเมาส์ตามแกน X และแกน Y

```
00 00 FF 00
00 00 01 00
00 00 01 00
00 00 01 00
00 00 01 00
00 FF 00 00
00 FF 00 00
00 00 01 00
00 01 00 00
00 01 01 00
00 00 01 00
00 00 01 00
```

คอลัมน์ที่ 2 แสดงการเลื่อนเมาส์ตามแกน X โดยจะส่งค่าระหว่าง 1-FF ออกมา และคอลัมน์ ที่ 3 แสดงการเลื่อนเมาส์ตามแกน Y โดยจะส่งค่าระหว่าง 1-FF ออกมา