การแลกเปลี่ยนข้อมูลผ่าน USB

Exchanging Information though USB

โดย

นาย ชาติชาย ไชยกุมาร

นาย โสภณ ด่วนแสวง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา2546

ปริญญานิพนธ์ปีการศึกษา2546

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การแลกเปลี่ยนข้อมูลผ่าน USB

Exchanging Information though USB

ผู้จัดทำ 1. นายชาติชาย ไชยคุมาร 44015280

2. นายโสภณ ด่านแสวง 44015312

.....อาจารย์ที่ปรึกษา

(ร.ศ. สุเธียร เกียรติสุนทร)

การแลกเปลี่ยนข้อมูลผ่าน USB

ชาติชาย ไชยกุมาร 44015280
โสภณ ด่วนแสวง 44015312
ร.ศ. สุธีธร เกียรติสุนทร อาจารย์ที่ปรึกษา
ปีการศึกษา2546

บทคัดย่อ

วัตถุประสงค์โครงงานนี้ เพื่อการแลกเปลี่ยนข้อมูลผ่าน USB ระหว่างคอมพิวเตอร์ 2 เครื่อง โดยใช้โปรโตคอล TCP/IP เพื่อช่วยลดความซับซ้อนในการเขียนโปรแกรมติดต่อกับฮาร์ดแวร์โดยตรง จากนั้นทำการจำลองระบบควบคุมระดับน้ำ ขึ้นมาโดยให้เครื่องคอมพิวเตอร์เครื่องหนึ่งเป็นแทงค์น้ำ และให้คอมพิวเตอร์อีกเครื่องหนึ่งเป็นตัวควบคุมระบบ โดยการจำลองนี้จะใช้การควบคุมแบบ PID เป็นตัวควบคุม

Exchanging Information though USB

Chatchai Chaikumarn

Sopon Duansawang

Associate Professor Sutean Keattisunton Advisor

2003

ABSTRACT

The objective of the project is to exchange information though USB port between two computers by using TCP/IP (Transmission Control Protocol / Internet Protocol). It aims to reduce complication of creating program by directly connect with hardware. After that we will stimulate tidewater control system by using two computers as the devices. One of them will act as tank and another will be the control system. This stimulation will use PID system as a controller.

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงการ

ปัจจุบันการรับและส่งข้อมูลระหว่างคอมพิวเตอร์ได้รับการพัฒนาอย่างรวดเร็ว เมื่อสมัยก่อนเมื่อต้องการส่งข้อมูลมักจะนิยมใช้ พอร์ตอนุกรม (Serial Port) หรือพอร์ตขนาน (Parallel Port) ในการรับส่งข้อมูล แต่เนื่องจากพอร์ตอนุกรม หรือพอร์ตขนานมีข้อจำกัดหลายอย่าง เช่น ความเร็ว จำนวนพอร์ตในการใช้งาน ความเข้ากันได้กับอุปกรณ์ต่อพ่วงสมัยใหม่

ดังนั้นจึงได้มีการพัฒนาพอร์ตที่สามารถแก้ปัญหาดังกล่าวข้างต้น และเพื่อให้ง่ายต่อการใช้งาน จึงได้มีการพัฒนาเป็น พอร์ต USB (Universal Serial Bus) ที่มีคุณสมบัติโดดเด่นหลายประการเช่น ความเร็ว ความเข้ากันได้กับอุปกรณ์สมัยใหม่ และสามารถขยายพอร์ตได้ถึง 127 พอร์ต

โครงงานนี้จึงต้องอาศัยความรู้เกี่ยวพอร์ตและมาตรฐานของ USB และโปรโตคอล TCP/IP ที่สนับสนุนการทำงานของสาย USB ด้วย รวมถึงการจำลองการทำงานของระดับน้ำที่ใช้การควบคุมแบบพีไอดี (PID) ด้วยการแลกเปลี่ยนข้อมูลผ่าน USB

1.2 วัตถุประสงค์ในการดำเนินงาน

1. เพื่อศึกษามาตรฐานและการทำงานของพอร์ต USB
2. เพื่อศึกษาและการใช้งานโปรโตคอล TCP/IP
3. เพื่อศึกษาการเขียนโปรแกรมและการใช้งานโปรแกรม Delphi
4. เพื่อศึกษาการทำงานของระบบควบคุมแบบพีไอดี

1.3 ขั้นตอนการศึกษาและจัดทำโครงงาน

การศึกษาและจัดทำโครงงานนี้เริ่มต้นจาก ศึกษาหาข้อมูลเกี่ยวกับมาตรฐานและการทำงานของพอร์ต USB เพื่อที่จะได้ทราบคุณสมบัติและข้อกำหนดของพอร์ต USB ร่วมถึงการทำงานของพอร์ต USB ด้วย หลังจากนั้นทำการศึกษาเกี่ยวกับโปรโตคอล TCP/IP ที่สนับสนุนการทำงานของสาย USB

โครงงานนี้ต้องอาศัยพื้นฐานการเขียนโปรแกรมเกี่ยวกับโปรโตคอล TCP/IP เพื่อใช้ในการแลกเปลี่ยนข้อมูลถึงกันระหว่างคอมพิวเตอร์ ร่วมถึงการนำทฤษฎีทางด้านระบบควบคุมมาจำลองการทำงานของระดับน้ำอีกด้วย

ในโครงงานนี้จะทำการจำลองการควบคุมระดับน้ำแบบพีไอดี โดยการแยกออกเป็นสองส่วน โดยส่วนแรกจะเป็นระบบควบคุมระดับน้ำ และส่วนที่สองจะเป็นระดับน้ำ โดยสองส่วนจะเชื่อมถึงกันด้วยสาย USB และจะทำการแลกเปลี่ยนข้อมูลระหว่างกันโดยใช้โปรโตคอล TCP/IP

1.4 เนื้อหาที่จะกล่าวในปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้คือ ในบทที่ 2 จะกล่าวถึงทฤษฎีและมาตรฐานของ USB ที่ใช้งานอยู่ในปัจจุบัน ในบทที่ 3 จะกล่าวถึงโปรโตคอล TCP/IP มาตรฐาน OSI Model และมาตรฐานของ TCP/IP ในบทที่ 5 จะกล่าวถึงทฤษฎีระบบควบคุม ระบบพี พีไอ พีดี และ พีไอดี ในบทที่6 เป็นส่วนของการวิเคราะห์และผลการทดลอง การทำงานของโปรแกรมการแลกเปลี่ยนข้อมูลเพื่อการควบคุมระดับน้ำ ในบทที่ 7 จะเป็นส่วนวิจารณ์และสรุปผลการดำเนินงานและปัญหาที่ประสบ และแนวทางการปรับปรุงและพัฒนาโครงงานชิ้นนี้ต่อไป

บทที่ 2

Universal Serial Bus

เมื่อมีอุปกรณ์ที่ต้องการนำมาเชื่อมต่อกับคอมพิวเตอร์ก็ต้องหาช่องทางสำหรับรับและส่งข้อมูลจากตัวเครื่อง ช่องทางการส่งหรือรับข้อมูลต่าง ๆ จากเครื่องคอมพิวเตอร์ถูกตั้งชื่อว่า พอร์ต (Port) โดยพอร์ตแต่ละชนิดก็ถูกออกแบบมาเพื่อให้สามารถรับส่งข้อมูล เฉพาะได้อย่างเหมาะสม ไม่ว่าจะเป็น พอร์ตอนุกรม (Serial port) ซึ่งใช้รับส่งข้อมูลความเร็วต่ำแบบอนุกรมในระยะทางที่ไกล มักนำมาใช้กับเมาส์หรือโมเด็ม พอร์ตขนาน (Parallel port) ใช้รับส่งข้อมูลความเร็วปานกลางแบบขนานได้ในระยะที่ไม่ไกลมากนัก ถูกนำมาใช้กับเครื่องพิมพ์และสแกนเนอร์ เป็นต้น นอกจากนั้นยังมีพอร์ตเฉพาะพิเศษ อาทิ พอร์ตรับส่งข้อมูลผ่านแสงอินฟราเรด (IrDA) และพอร์ต USB (Universal serial Bus) อันเป็นพอร์ตรับส่งข้อมูลอนุกรมแบบอเนกประสงค์ที่ได้รับความนิยมอย่างมากในการพัฒนาอุปกรณ์เชื่อมต่อกับคอมพิวเตอร์ตัดจากการใช้พอร์ตอนุกรมและขนาน

2.1 จุดกำเนิดของ USB

พอร์ตแต่ละชนิดของคอมพิวเตอร์ได้รับการออกแบบมาเพื่องานเฉพาะ ทำให้อุปกรณ์แต่ละตัวต้องเลือกใช้พอร์ตต่างชนิดกันไป ส่งผลให้การนำอุปกรณ์ต่าง ๆ มาเชื่อมต่อกับเครื่องคอมพิวเตอร์เป็นเรื่องที่ต้องใช้ความสนใจ เนื่องจากอุปกรณ์แต่ละชนิดก็จะมีคอนเน็กเตอร์ที่ใช้เชื่อมต่อแตกต่างกันไปตามชนิดของพอร์ต เช่น พอร์ตอนุกรมที่ใช้ต่อคีย์บอร์ดและเมาส์ (ทั้งแบบ PS/2 และพอร์ตอนุกรมมาตรฐาน) พอร์ตขนานสำหรับต่อเครื่องพิมพ์ พอร์ตเชื่อมต่อของจอ ฯลฯ ทำให้ผู้ใช้ที่ไม่มีความรู้ในการติดตั้งหรือเรียกกันว่า เอ็นยูสเซอร์ (End user) พบกับความยากลำบากในการเรียนรู้เรื่องราวเหล่านี้

นอกจากนั้นการติดตั้งอุปกรณ์ต่างๆ เพิ่มเข้าไปในเครื่องคอมพิวเตอร์จะเกิดปัญหาการแย่งกันใช้สัญญาณ IRQ (Interrupt Request) ซึ่งเป็นตัวจำกัดจำนวนอุปกรณ์ที่จะนำมาต่อ ทำให้เกิดแนวความคิดที่จะกำหนดมาตรฐาน เพื่อสร้างเป็นพอร์ตที่ทำให้การเชื่อมต่อทั้งหมดอยู่ในรูปแบบเดียวกัน ง่ายต่อการใช้งานของผู้ใช้ทั่ว ๆ ไปและไม่มีข้อจำกัดการใช้ IRQ คำตอบแนวคิดนี้คือ พอร์ต USB นั้นเอง

USB ย่อมาจาก Universal Serial Bus คือบัสอนุกรมเอนกประสงค์ คุณสมบัติต่างๆ ที่ทำให้สามารถกำหนดชื่อนี้ให้กับพอร์ตชนิดนี้ได้มีดังนี้

- ใช้คอนเน็กเตอร์เพียงชนิดเดียว ซึ่งมี 2 รูปแบบสำหรับเชื่อมต่ออุปกรณ์ทุก ๆ ชนิดเข้ากับคอมพิวเตอร์

- สามารถเชื่อมต่ออุปกรณ์หลาย ๆ ชนิดรวมเข้าสู่คอมเน็กเตอร์ตัวเดียว สูงสุด 127 ตัว
- ไม่เกิดการขัดแย้งกันของการใช้ทรัพยากรของระบบ (IRQ)
- ตรวจสอบการเชื่อมต่อและตั้งค่าการทำงานต่าง ๆ อัตโนมัติระหว่างที่เครื่องกำลังทำงานอยู่ (Hot Attachment)
- ความเร็วในการถ่ายทอดข้อมูลจะขึ้นอยู่กับมาตรฐาน อันมีรายละเอียดดังนี้
  1. มาตรฐาน USB 1.0/1.1 มีอัตราในการถ่ายทอดข้อมูลความเร็วต่ำ (low speed) เท่ากับ 1.5 เมกะบิตต่อวินาที (Mbit/sec) และความเร็วเต็มที่ (Full Speed) เท่ากับ 12 เมกะบิตต่อวินาที (Mbit/sec)
  2. มาตรฐาน USB 2.0 จะมีอัตราเร็วในการถ่ายทอดเพิ่มขึ้นอีก 1 ระดับคือ ความเร็วสูง (High Speed) ซึ่งมีความเร็วสูงสุดถึง 480 เมกะบิตต่อวินาที
- ที่ขาพอร์ต USB มีแรงดันไฟตรง +5V จ่ายออกมาด้วย ทำให้อุปกรณ์ต่อพ่วงที่ใช้พลังงานไม่มากนัก สามารถใช้แรงดันจากพอร์ต USB นี้เป็นไฟเลี้ยงเพื่อทำงานได้ โดยไม่ต้องอาศัยแหล่งจ่ายไฟภายนอกเพิ่มเติมอีก

2.2 ความเร็วการสื่อสารข้อมูลของอุปกรณ์ USB

1. อุปกรณ์แบบโลว์สปีด (Low Speed Device) มีความเร็วของการสื่อสารข้อมูลที่ 1.5 เมกะบิตต่อวินาที ตัวอย่างของอุปกรณ์ที่ใช้การสื่อสารในโหมดนี้ได้แก่ เมาส์และคีย์บอร์ด
2. อุปกรณ์แบบฟูลสปีด (Full Speed Device) มีความเร็วของการสื่อสารข้อมูลที่ 12 เมกะบิตต่อวินาที เหมาะสำหรับอุปกรณ์เกี่ยวกับเสียงและสัญญาณภาพที่มีการบีบอัดแล้ว
3. อุปกรณ์ไฮสปีด (High Speed Device) มีความเร็วของการสื่อสารข้อมูลที่ 480 เมกะบิตต่อวินาที เหมาะสำหรับอุปกรณ์เกี่ยวกับภาพและสื่อเก็บข้อมูล เช่นฮาร์ดดิสก์

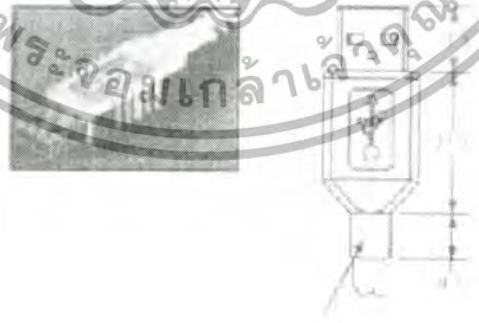
ความเร็วต่าง ๆ เหล่านี้คือค่าความเร็วของสัญญาณ หรืออัตราบิตข้อมูลซึ่งสนับสนุนโดยระบบบัส แต่ในทางปฏิบัตินั้นอัตราการส่งข้อมูลที่แท้จริงอาจมีค่าต่ำกว่านี้เนื่องจากระบบบัสต้องมีการส่งถ่ายสัญญาณข้อมูลเป็นจำนวนมาก และยังมีการแบ่งช่องสัญญาณ ในกรณีที่มีอุปกรณ์รอบข้างหลายตัวต่ออยู่ภายในระบบ ซึ่งตามทฤษฎีแล้วในอุปกรณ์แบบไฮสปีด จะมีความเร็วอยู่ในช่วง 25 - 400 เมกะบิตต่อวินาที ส่วนอุปกรณ์แบบฟูลสปีดจะมีความเร็วอยู่ที่ประมาณ 0.5 ถึง 10 เมกะบิตต่อวินาทีและอุปกรณ์แบบโลว์สปีดจะอยู่ที่ 10 – 100 กิโลบิตต่อวินาที

2.3 คอนเน็คเตอร์และสายสัญญาณของ USB

2.3.1 คอนเน็กเตอร์ USB

สายเชื่อมต่อ USB ถูกออกแบบมาสำหรับการเชื่อมต่อระหว่างตัวอุปกรณ์ USB กับฮับซึ่งอาจจะเป็นรูทฮับซึ่งอยู่หลังเครื่องคอมพิวเตอร์ ฮับที่รวมอยู่ในตัวอุปกรณ์ (compound device) หรือฮับที่เป็นตัวเดี่ยวๆ (USB hub) อุปกรณ์หลายชนิดมีสายต่อด้านอุปกรณ์ไว้อย่างถาวรด้านหนึ่งและมีคอนเน็กเตอร์ตัวผู้ไว้สำหรับต่อเข้ากับฮับอีกด้านหนึ่ง แต่มีอุปกรณ์อีกหลายชนิดที่ไม่มีสายต่อถาวรไว้ โดยที่ตัวอุปกรณ์จะมีคอนเน็กเตอร์ USB ตัวเมียไว้เพื่อใช้งานกับสาย USB ที่มีหัวคอนเน็กเตอร์ทั้งสองด้าน จากอุปกรณ์ที่ไม่มีสายถาวรต่อไว้จะเห็นได้ว่าต้องใช้สายเชื่อมต่อซึ่งมีคอนเน็กเตอร์ทั้งสองด้าน ซึ่งถ้าใช้คอนเน็กเตอร์แบบเดียวกันทั้งหมดอาจจะทำให้ผู้ใช้นำคอนเน็กเตอร์ด้านหนึ่งต่อเข้ากับฮับพอร์ตหนึ่ง แล้วต่อปลายอีกข้างเข้ากับฮับอีกพอร์ตหนึ่งได้ (เพราะฮับจะมีพอร์ตอยู่หลายพอร์ต) เพื่อป้องกันเหตุการณ์เช่นนี้จึงมีการกำหนดมาตรฐานของคอนเน็กเตอร์ USB ให้มี 2 รูปแบบ ดังนี้

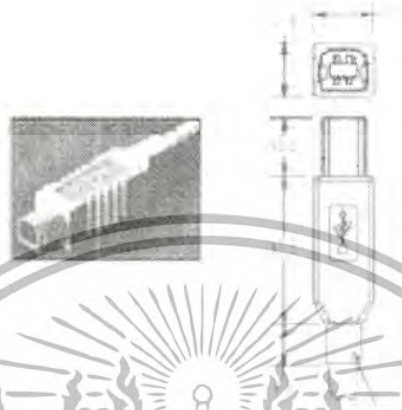
1. คอนเน็กเตอร์อนุกรม A เป็นคอนเน็กเตอร์ด้านฮับที่เชื่อมต่อระหว่าง USB พอร์ตของฮับ (ทั้งรูทฮับและ USB ฮับทั่วไป) กับสายเชื่อมต่อจากตัวอุปกรณ์ นั่นคือคอนเน็กเตอร์ตัวเมียจะติดตั้งอยู่กับฮับและคอนเน็กเตอร์ตัวผู้จะติดอยู่กับสายที่ต่อออกมาจากตัวอุปกรณ์ ดังในรูปที่ 2.1



รูปที่ 2.1 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม A

2. คอนเน็กเตอร์อนุกรม B เป็นคอนเน็กเตอร์ด้านอุปกรณ์ที่เชื่อมต่อสายเข้ากับตัวอุปกรณ์ USB นั้นคือคอนเน็กเตอร์ตัวเมียจะติดตั้งอยู่ในตัวอุปกรณ์ USB ส่วนคอนเน็กเตอร์ตัว

ผู้ก็จะอยู่ที่สายที่ต่อออกมาจากฮับ (ถ้าอุปกรณ์ใดที่มีสายต่อออกมาจากตัวอุปกรณ์อย่างถาวรจะไม่มีการใช้คอนเน็กเตอร์แบบนี้) มีรูปร่างแสดงในรูปที่ 2.2



รูปที่ 2.2 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม B

ถึงแม้จะมีการแบ่งคอนเน็กเตอร์ออกเป็น 2 แบบ แต่ทั้งสองรูปแบบนั้นใช้สายนำสัญญาณเหมือนกัน โดยใช้สาย 2 เส้นในการส่งไฟเลี้ยงให้แก่อุปกรณ์ และสายอีก 2 เส้นสำหรับรับและส่งข้อมูล แต่เนื่องจากระบบ USB นี้ถูกออกแบบมาเพื่อให้สามารถเชื่อมต่อหรือปลดอุปกรณ์ออกจากระบบได้ในระหว่างการใช้งาน ดังนั้นหน้าสัมผัสของคอนเน็กเตอร์จึงมีการออกแบบให้มีความพิเศษเพื่อให้รองรับคุณสมบัติดังกล่าวได้นั่นคือหน้าสัมผัสของสาย 2 เส้นที่ใช้ส่งไฟเลี้ยงจะยื่นออกมายาวกว่าหน้าสัมผัสที่ใช้รับส่งข้อมูล (หน้าสัมผัสสำหรับไฟเลี้ยงยาว 7.41 มิลลิเมตร หน้าสัมผัสสำหรับรับส่งข้อมูลยาว 6.41 มิลลิเมตร) การออกแบบคอนเน็กเตอร์เช่นนี้จะทำให้ตัวอุปกรณ์ได้รับไฟเลี้ยงก่อนที่จะได้รับสัญญาณข้อมูลเมื่อเชื่อมต่ออุปกรณ์ตัวใหม่เข้าในระบบ ทำให้อุปกรณ์ที่เชื่อมต่อเข้าไปใหม่สามารถทำงานได้ทันทีโดยไม่ได้รับความเสียหาย เพราะถ้าหากอุปกรณ์ได้รับสัญญาณข้อมูลก่อนที่ได้รับไฟเลี้ยงจะเกิดความเสียหายแก่ไอซีที่อยู่ภายในได้ขาของคอนเน็กเตอร์แต่ละตัวทำหน้าที่ตามที่แสดงในตารางที่ 2.1

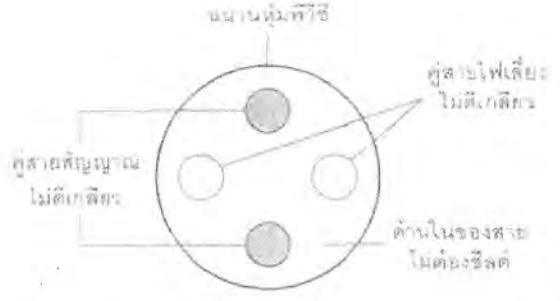
หมายเลขขาสัญญาณ	ชื่อขาสัญญาณ	สีของสายสัญญาณ
1	ไฟเลี้ยง +5V	แดง
2	D - (ข้อมูลลบ)	ขาว
3	D + (ข้อมูลบวก)	เขียว
4	กราวด์	ดำ

ตารางที่ 2.1 หน้าที่ของขาคอนเน็คเตอร์แต่ละขา

2.3.2 สายนำสัญญาณ USB

การถ่ายทอดข้อมูลภายในสาย USB นั้นมีความเร็วสูงในระดับ 1 Mb/s ถึง 12 Mb/s การถ่ายทอดข้อมูลที่ความเร็วสูงขนาดนี้จะเกิดการแพร่กระจายสนามแม่เหล็กไฟฟ้าออกมาจากสายส่งทำให้ต้องมีการกำหนดคุณสมบัติของสายส่งในหลายๆ ด้าน เพื่อไม่ให้เกิดการแพร่กระจายของสนามแม่เหล็กไฟฟ้าเหล่านี้ และเนื่องจาก USB มีการแบ่งความเร็วของการถ่ายทอดข้อมูลออกเป็น 2 ระดับ คือ แบบความเร็วเต็มที่ (full speed) ที่มีอัตราเร็ว 1 Mb/s และแบบความเร็วต่ำ (low speed) ที่มีอัตราเร็ว 1.5 Mb/s ทำให้คุณสมบัติของสายนำสัญญาณของอุปกรณ์แต่ละชนิดจะถูกกำหนดแยกแตกต่างกันไปดังนี้คือ

1. สายนำสัญญาณความเร็วต่ำ จากรูปที่ 2.3 แสดงให้เห็นภาพหน้าตัดของสายนำสัญญาณความเร็วต่ำ ชั้นนอกสุดของสายสัญญาณจะเป็นฉนวนหุ้มธรรมดา ภายในมีสายสัญญาณจะมีสายตัวนำ 4 เส้น โดย 2 เส้นใช้ส่งไฟเลี้ยงกำหนดให้ใช้ลวดทองแดงเบอร์ 20-28 AWG ไม่ตีเกลียวส่วนสายอีก 2 เส้นใช้ถ่ายทอดข้อมูลกำหนดให้ใช้ลวดทองแดงเบอร์ 28 AWG ไม่ตีเกลียวสายส่งข้อมูลความเร็วต่ำนี้ไม่จำเป็นต้องชีลด์ (Shield) ความยาวทั้งหมดของสายไม่เกิน 3 เมตร



รูปที่ 2.3 โครงสร้างภายในของสาย USB แบบความเร็วต่ำ

2. สายนำสัญญาณความเร็วสูง ข้อกำหนดของสายนำสัญญาณความเร็วสูงของ USB จะมีมากกว่าสายนำสัญญาณความเร็วต่ำอยู่หลายๆ ด้าน สาย 2 เส้นที่ใช้ถ่ายทอดข้อมูลกำหนดให้ใช้ลวดทองแดงเบอร์ 28 AWG เช่นเดียวกับสายนำสัญญาณความเร็วต่ำ แต่สาย 2 เส้นนี้จะต้องตีเกลียวและชั้นนอกสุดของสายสัญญาณถัดจากฉนวนจะต้องถูกหุ้มด้วยอลูมิเนียมฟอยล์ ดังในรูปที่ 10 ความยาวของสายสัญญาณไม่เกิน 5 เมตร จะต้องมีค่าหน่วงเวลาสัญญาณ (propagation delay) ไม่เกิน 30 นาโนวินาทีตลอดความยาวสาย 5 เมตร แต่ถ้าค่าหน่วงเวลามากกว่า 30 นาโนวินาที ค่าความยาวสูงสุดของสายสัญญาณก็จะลดลงเป็นสัดส่วนดังในตารางที่ 2.2

ค่าอัตราการหน่วงเวลาของสาย USB	ความยาวของสายสัญญาณ
9.0 นาโนวินาทีต่อเมตร	3.3 เมตร
8.0 นาโนวินาทีต่อเมตร	3.7 เมตร
7.0 นาโนวินาทีต่อเมตร	4.3 เมตร
6.5 นาโนวินาทีต่อเมตร	4.6 เมตร

ตารางที่ 2.2 อัตราการหน่วงเวลาของสาย USB เมื่อเทียบกับความยาวของสายสัญญาณ

2.3.3 การตรวจสอบการเชื่อมต่อของระบบ USB

คุณสมบัติเด่นข้อหนึ่งของ USB คือ สามารถตรวจสอบการเชื่อมต่อของอุปกรณ์ตัวใหม่และตั้งค่าเริ่มต้นที่จำเป็นได้อัตโนมัติ เนื่องจากอุปกรณ์ USB แบ่งออกเป็นอุปกรณ์ความเร็วสูงและต่ำ ถ้าส่งข้อมูลผิดความเร็ว ก็จะไม่สามารถสื่อสารกับอุปกรณ์นั้น ๆ ได้นั้นหมายความว่าจะต้องมีสัญญาณรูปแบบพิเศษที่ใช้แจ้งสถานะการเชื่อมต่อหรือปลดออก รวมถึงระบุความเร็วของอุปกรณ์นั้น ๆ ได้ด้วย

การตรวจสอบการเชื่อมต่อของพอร์ต USB นั้นจะตรวจสอบจากการเปลี่ยนแปลงของระดับสัญญาณในสายสัญญาณ D- และ D+ โดยสายข้อมูลทั้งสองที่ด้านฮับจะถูกต่อด้วยตัวต้านทาน 15K โอมห์ พูลดาวน์ไว้ทั้งสองเส้น (การพูลดาวน์ (Pull-Down): เป็นการต่อขาข้างหนึ่งของอุปกรณ์เข้ากับสายสัญญาณอีกข้างหนึ่งต่อกราวด์ของระบบ) ซึ่งจะส่งผลให้สายสัญญาณทั้งสองมีระดับแรงดันเป็น 0V ในขณะที่ไม่มีอุปกรณ์ใด ๆ ต่ออยู่

ที่ด้านอุปกรณ์สำหรับอุปกรณ์ความเร็วต่ำสายสัญญาณ D- จะต่อตัวต้านทาน 1.5Kโอมห์ พูลอัปกับแรงดัน 3.3 -3.6 V (การพูลอัป (Pull-Up): เป็นการต่อขาข้างหนึ่งของอุปกรณ์เข้ากับสายสัญญาณอีกข้างหนึ่งต่อไฟเลี้ยง ทำให้สถานะที่จุดนั้นเกิดเป็นลอจิก "1") ส่วนอุปกรณ์ความเร็วสูงสายสัญญาณที่ถูกพูลอัปคือ สายสัญญาณ D+ เมื่อมีอุปกรณ์ตัวใหม่ถูกต่อเข้ากับฮับจะเกิดการแบ่งแรงดันจากตัวต้านทานสองตัวก็คือตัวต้านทาน 1.5K โอมห์ ที่พูลอัปทางอุปกรณ์และตัวต้านทาน 15 Kโอมห์ ที่พูลดาวน์อยู่ทางฮับ ทำให้แรงดันของสายสัญญาณเพิ่มจาก 0V ขึ้นไปที่ 90% ของไฟเลี้ยงที่พูลอัป

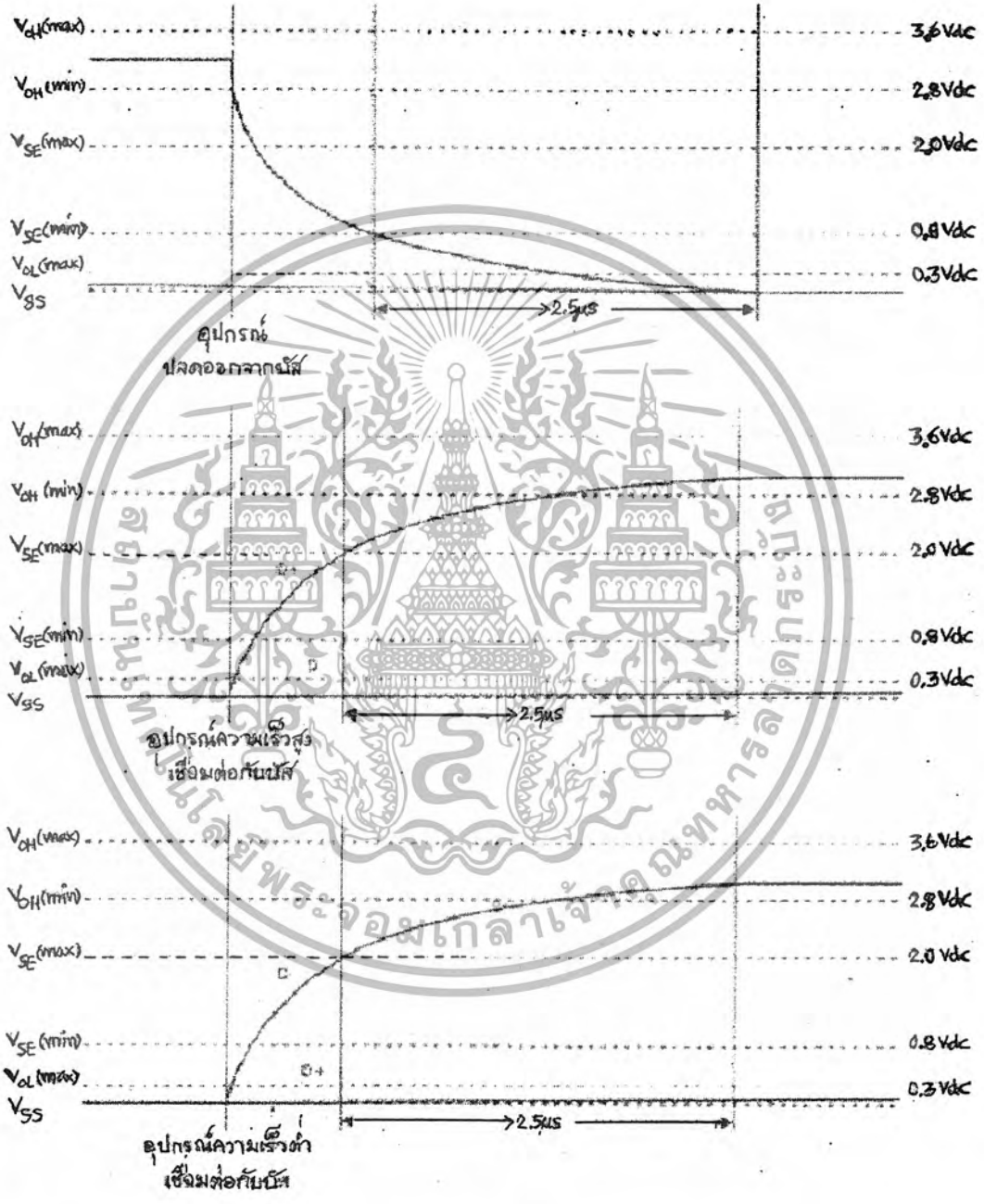
$$[\text{คำนวณได้จาก } \frac{15\times10^{3}}{(1.5+15)\times10^{3}\times Vcc}]$$

การเปลี่ยนแรงดันนี้จะทำให้ฮับรู้ว่ามีอุปกรณ์ตัวใหม่ต่อเข้ากับระบบและจะทำให้รู้ว่าอุปกรณ์ถูกปลดออกจากระบบเช่นเดียวกัน โดยถ้าสัญญาณที่เกิดการเปลี่ยนแปลงแรงดันนี้เป็นสัญญาณ D- หมายความว่าอุปกรณ์ที่นำมาต่อเป็นอุปกรณ์ความเร็วต่ำ แต่ถ้าสัญญาณ D+ เปลี่ยนแปลงแรงดันหมายความว่าอุปกรณ์ความเร็วสูงถูกนำมาต่อเข้ากับระบบดังรูปที่ 2.4

รูปที่ 2.4 การตรวจสอบการเชื่อมต่อของระบบ USB

ในรูปที่ 2.5 แสดงให้เห็นถึงระดับแรงดันในสายสัญญาณเมื่อมีการเชื่อมต่ออุปกรณ์ใหม่และเมื่อปลดอุปกรณ์ออกจากบัส ถ้าแรงดันของสัญญาณ D- หรือ D+ ตกลงต่ำกว่าค่า $V_{SE}(min)$ หรือ 0.8 V เป็นเวลานานกว่า 2.5 ไมโครวินาที ฮับจะถือว่าอุปกรณ์ได้ถูกปลดออกจากบัส ในทางกลับกันเมื่อแรงดันของสัญญาณ D- หรือ D+ เพิ่มขึ้นสูงกว่าค่า $V_{SE}(max)$ หรือ 2.0 V เป็นเวลานานกว่า 2.5 ไมโครวินาที จะถือว่ามีการต่ออุปกรณ์ตัวใหม่เข้ามาในระบบ ผลการตรวจสอบที่ได้ฮับจะ

เก็บไว้ในรีจิสเตอร์สถานะ ซึ่งโฮสจะคอยมาอ่านเป็นระยะ ๆ เพื่อตรวจสอบการเชื่อมต่อของอุปกรณ์



รูปที่ 2.5 ระดับสัญญาณภายในสาย D+, D- เมื่อมีการเชื่อมต่อและปลดอุปกรณ์

2.4 การจัดการด้านพลังงานของ USB

เนื่องจากพอร์ต USB สามารถจ่ายพลังงานให้แก่อุปกรณ์ที่นำมาเชื่อมต่อได้ แต่ย่อมต้องมีข้อจำกัดว่า สามารถจ่ายพลังงานออกไปได้มากน้อยเพียงใด โดยเฉพาะกับพอร์ต USB ที่ยอมให้อุปกรณ์มาเชื่อมต่อได้มากถึง 127 ตัว โฮสจะต้องมีการจัดการด้านพลังงานที่ดีพอมิฉะนั้นจะเกิดการส่งพลังงานจากบัสมากจนเครื่องคอมพิวเตอร์ไม่สามารถทำงานได้

2.4.1 การรายงานอัตราการใช้พลังงาน

โฮสจะรู้ข้อมูลเกี่ยวกับการใช้พลังงาน ของฮับและอุปกรณ์แต่ละตัวจากการอ่านคอนฟิกิวเรชันดีสคริปเตอร์ ซึ่งจะถูกอ่านออกมาตั้งแต่ครั้งแรกที่มีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในบัส โดยข้อมูลที่เกี่ยวกับการใช้พลังงานจะมีอยู่ 2 ส่วน ส่วนแรกจะบอกโฮสว่าตัวอุปกรณ์นี้ใช้พลังงานจากบัสหรือใช้พลังงานจากแหล่งพลังงานของตัวเอง และส่วนที่สองจะบอกว่าจะขอใช้พลังงานจากบัสเป็นปริมาณเท่าใด มีหน่วยเป็นมิลลิแอมป์ (mA)

รูปที่ 2.6 ข้อจำกัดด้านแรงดันที่จุดเชื่อมต่อต่าง ๆ

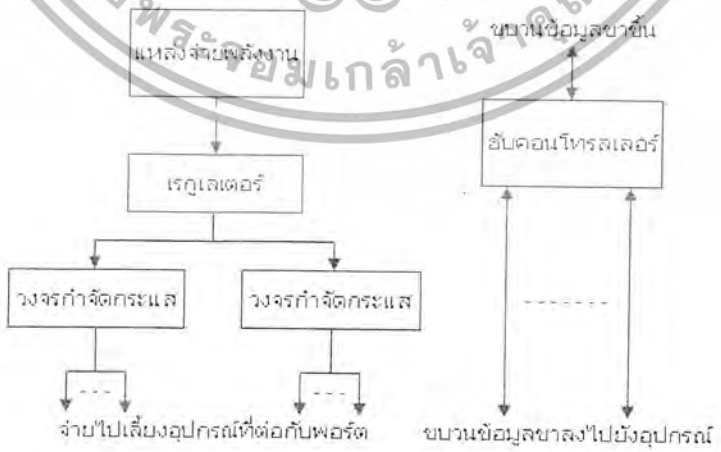
ในกรณีที่มีการเชื่อมต่อฮับเข้ากับบัส โฮสจะอ่านคอนฟิกิวเรชันดีสคริปเตอร์ ไปเพื่อพิจารณาการใช้พลังงาน แต่เนื่องจากฮับจำเป็นจะต้องจ่ายพลังงานไปเลี้ยงอุปกรณ์ที่มาเชื่อมต่ออยู่

กับตัวมันด้วย จึงต้องมีข้อกำหนดในด้านแรงดันของไฟเลี้ยงที่จะจ่ายออกจากตัวฮับเพื่อไม่ให้แรงดันปลายทางที่ตัวอุปกรณ์น้อยเกินไปจนไม่สามารถทำงานได้ โดยกำหนดไว้ว่า แรงดันที่จ่ายให้แก่ฮับเพื่อให้ฮับทำงานนั้นจะต้องไม่ต่ำกว่า 4.75V และแรงดันที่ฮับจ่ายออกไปยังตัวอุปกรณ์แต่ละตัวจะต้องไม่ต่ำกว่า 4.40V (ดูรูปที่ 2.6 ประกอบ) นั้นหมายความว่าอุปกรณ์ USB ทุกๆ ตัวจะต้องสามารถทำงานได้ที่แรงดันไฟเลี้ยงอย่างน้อย 4.40 V

2.4.2 การจำกัดกระแส

ที่กล่าวมาเป็นข้อกำหนดในเรื่องแรงดันของฮับ ข้อกำหนดอีกส่วนหนึ่งที่ต้องกล่าวถึงคือข้อกำหนดเรื่องกระแส ค่ากระแสต่ำสุดที่ฮับจะต้องจ่ายให้แต่ละพอร์ตจะต้องไม่น้อยกว่า 100 mA และจ่ายได้มากที่สุด 500 mA นั้นหมายความว่า อุปกรณ์ที่มาเชื่อมต่อกับบัส USB โดยใช้พลังงานจากบัสนั้นจะไม่สามารถดึงกระแสได้มากกว่า 500 mA การใช้พลังงานของตัวอุปกรณ์หรือฮับจะแบ่งย่อยออกเป็นหน่วย หน่วยละ 100 mA หมายความว่า อุปกรณ์ที่ใช้พลังงานน้อยที่สุดจะใช้กระแสไฟฟ้า 1 หน่วยหรือ 100 mA นั่นเอง

สำหรับฮับที่ใช้ไฟเลี้ยงจากบัส ตัวฮับเองจะใช้พลังงานจากบัสเพื่อควบคุมการทำงานภายในของตัวฮับเอง 1 หน่วยเท่ากับ 100 mA และจากข้อกำหนดด้านกระแสที่ว่า บัสจะจ่ายกระแสได้ไม่เกิน 500mA นั้นหมายความว่า ฮับจะเหลือพลังงานที่สามารถจ่ายให้แก่พอร์ตต่างๆ ได้อีก 4 หน่วย ซึ่งเท่ากับว่า จะมีพอร์ตได้มากที่สุด 4 พอร์ตโดยแต่ละพอร์ตต้องดึงกระแสไม่เกิน 100 mA แต่ในกรณีของฮับที่มีแหล่งจ่ายไฟของตัวเองจะไม่พบกับปัญหานี้เพราะสามารถจ่ายกระแสให้แต่ละพอร์ตได้เต็ม 500 mA



รูปที่ 2.7 การจัดการพลังงานของฮับที่มีแหล่งจ่ายไฟเลี้ยง

นอกจากข้อกำหนดในเรื่องการดึงกระแสของแต่ละพอร์ตแล้ว ในแต่ละพอร์ตจะต้องมีวงจรจำกัดกระแส (Current limit) ไม่ให้เกิน 5 A สำหรับพอร์ตแต่ละพอร์ต เพื่อป้องกันการลัดวงจรระหว่างไฟเลี้ยงและกราวด์ แต่ในกรณีของฮับที่อาศัยไฟเลี้ยงจากบัสไม่จำเป็นต้องมีวงจรป้องกันส่วนนี้เพราะรูทฮับหรือฮับตัวก่อนหน้า (ที่มีแหล่งจ่ายไฟของตัวเอง) จะมีวงจรป้องกันอยู่แล้ว

2.4.3 การเปิด-ปิดไฟเลี้ยงของฮับ

นอกจากฮับจะแบ่งออกเป็น 2 ชนิดตามการใช้พลังงานคือ ฮับที่ใช้พลังงานจากบัส และฮับที่ใช้พลังงานของตัวเองแล้ว ฮับยังแบ่งชนิดย่อยตามการเปิดการจ่ายพลังงานให้แก่อุปกรณ์ที่พอร์ตต่างๆ ด้วยโดยแบ่งออกเป็น 3 ชนิดคือ

1. จ่ายพลังงานโดยตรง ในแบบนี้จะจ่ายพลังงานไปยังพอร์ตต่างๆ ไว้ตลอดเวลา
2. เปิดปิดการจ่ายพลังงานแบบรวม ในแบบนี้ฮับสามารถสั่งปิดหรือเปิดการจ่ายพลังงานแก่พอร์ตต่างๆ ได้ แต่จะเป็นการกระทำกับพอร์ตทุกๆ พอร์ตพร้อมกัน นั้นคือถ้าเปิดก็เปิดทั้งหมด ถ้าปิดก็ปิดทั้งหมด
3. เปิดปิดการจ่ายพลังงานแบบแยกพอร์ต ในแบบนี้ฮับสามารถสั่งปิดเปิดการจ่ายพลังงานแต่ละพอร์ตแยกกันอย่างอิสระ

โฮสจะทราบถึงลักษณะการจ่ายพลังงานของฮับได้จากการอ่านฮับคลาสดีสคริปเตอร์ (เป็นคลาสสเปกซิฟิกดีสคริปเตอร์ชนิดหนึ่ง ดูรายละเอียดของดีสคริปเตอร์ชนิดต่างๆ ได้จากหัวข้อภาพรวมการทำงานของพอร์ต USB)

2.4.4 การใช้พลังงานในช่วงตั้งค่าเริ่มต้นทำงานหรือคอนฟิก

อุปกรณ์ USB เมื่อเชื่อมต่อเข้ามาในบัสต้องได้รับการกำหนดลักษณะการทำงานจากโฮสก่อนเริ่มทำงานทุกครั้ง เพื่อให้โฮสทราบถึงข้อมูลที่จำเป็นในการติดต่อกับอุปกรณ์นั้นๆ ด้วยการอ่านข้อมูลดีสคริปเตอร์ต่าง ๆ เพื่อตั้งค่าเริ่มต้นให้แก่ระบบและตัวอุปกรณ์ ในช่วงการตั้งค่าเริ่มต้นนี้จะมีข้อกำหนดว่า อุปกรณ์จะสามารถดึงพลังงานไปใช้ได้ไม่เกิน 1 หน่วยหรือ 100 mA แต่หลังจากโฮสพิจารณาทรัพยากรที่ระบบมี ว่าเพียงพอต่อความต้องการของอุปกรณ์และตั้งค่าต่าง ๆ ให้จนเสร็จเรียบร้อยแล้ว อุปกรณ์หรือฮับจะสามารถดึงกระแสได้เต็มความต้องการที่แจ้งไว้ในดีสคริปเตอร์

2.4.5 การใช้พลังงานของอุปกรณ์ USB

อุปกรณ์ USB สามารถแบ่งชนิดจากลักษณะการใช้พลังงานได้ 2 กลุ่มคือ อุปกรณ์ที่ใช้พลังงานจากบัส (ใช้พลังงานต่ำ) เช่น เมาส์ คีย์บอร์ด และอุปกรณ์ที่ใช้พลังงานของตัวเอง (ใช้พลังงานสูง) เช่น เครื่องพิมพ์ ซีดีรอมไดร์ฟ อุปกรณ์ที่ใช้พลังงานจากบัสสามารถดึงกระแสได้ตั้งแต่ 100 – 500 mA ดังที่ได้กล่าวไปแล้ว แต่ก็ต้องขึ้นอยู่กับความสามารถในการจ่ายพลังงานของฮับด้วยว่า ฮับนั้นๆ มีอุปกรณ์ต่ออยู่แล้วกี่ตัว แต่ละตัวดึงพลังงานไปแล้วเท่าใด เหลือพลังงานให้เพียงพอกับที่อุปกรณ์ตัวใหม่ต้องการหรือไม่ หากโฮสพิจารณาแล้วว่าไม่สามารถจ่ายพลังงานให้อุปกรณ์ตัวใหม่นี้ได้เพียงพอ ก็จะไม่ตั้งค่าเริ่มต้นให้อุปกรณ์นั้นๆ ทำให้อุปกรณ์ตัวนั้นไม่สามารถทำงานได้

สำหรับอุปกรณ์ที่มีใช้พลังงานของตัวเองจะไม่เกิดเหตุการณ์ดังกล่าวขึ้น เพราะไม่ต้องใช้พลังงานจากบัสเลย แต่มีอุปกรณ์อีกประเภทหนึ่งที่มีรูปแบบการใช้พลังงานที่ต่างออกไป โดยจะใช้พลังงานจากบัสสำหรับวงจรส่วนติดต่อกับ USB แต่ใช้พลังงานของตัวเองสำหรับการทำงานหลักของตัวอุปกรณ์ อุปกรณ์ที่ใช้พลังงานลักษณะนี้เรียกว่า อุปกรณ์ไฮบริด (Hybrid) ข้อดีของอุปกรณ์นี้คือ โฮสสามารถรู้สถานะของตัวอุปกรณ์ได้ แม้อุปกรณ์ตัวนั้นจะยังไม่ได้จ่ายไฟเลี้ยง นอกจากนั้นยังทำให้สามารถออกแบบอุปกรณ์ที่มีฟังก์ชันการทำงาน 2 รูปแบบที่ต่างกันในจังหวะที่มีและไม่มีไฟเลี้ยงได้

รูปที่ 2.8 การใช้พลังงานของอุปกรณ์ไฮบริด

2.4.6 การประหยัดพลังงานของพอรัต USB

นอกจากข้อจำกัดด้านการใช้พลังงานในสภาวะปกติ USB ยังมีข้อกำหนดสำหรับการประหยัดพลังงานของอุปกรณ์ในบัสด้วย โดยการเข้าสู่โหมดประหยัดพลังงานเรียกว่า Suspend และการกลับมาทำงานในสภาวะปกติเรียกว่า Resume และเนื่องจาก USB เป็นระบบบัสทำให้การเข้า

และออกจากโหมดประหยัดพลังงานจำเป็นต้องมีลำดับขั้นตอนในการส่งสัญญาณต่างๆ ที่สัมพันธ์กันตั้งแต่จุดเริ่มต้นคือ รูทฮับผ่านฮับไปยังตัวอุปกรณ์ต่างๆ

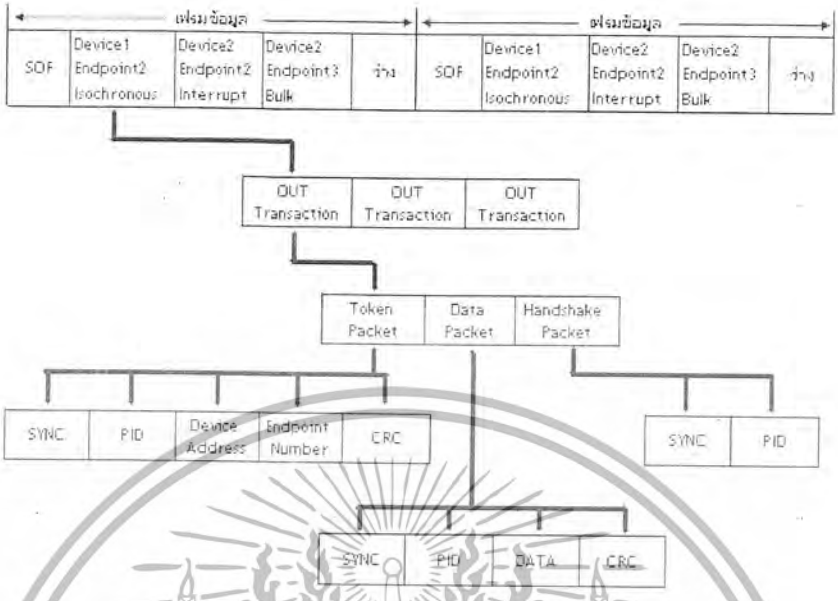
การเข้าสู่โหมดประหยัดพลังงานมีด้วยกัน 2 ชนิดคือ การ Suspend อุปกรณ์ทั้งบัส และ การ Suspend อุปกรณ์เฉพาะตัว

2.5 การส่งถ่ายข้อมูลบนระบบบัส USB

2.5.1 การจัดการข้อมูลบนระบบบัส USB

การสื่อสารข้อมูลบนระบบบัส USB เป็นการสื่อสารที่มีโฮสเป็นจุดศูนย์กลาง ซึ่งโฮสมีหน้าที่ในการทำให้การส่งถ่ายข้อมูลทั้งหมดที่อยู่บนบัสเกิดขึ้นเร็วที่สุดเท่าที่จะเป็นไปได้ โดยมันจะจัดการการขนส่งข้อมูลที่อยู่ภายในระบบบัสด้วยการแบ่งเวลาออกเป็นส่วนเล็ก ๆ เรียกว่าเฟรมและสำหรับโหมดไฮสปีดจะแบ่งเป็นไมโครเฟรม โฮสจะแบ่งส่วนของเฟรมหรือไมโครเฟรมให้แก่การส่งถ่ายข้อมูลแต่ละชุด (รูปที่ 2.9) โดยช่วงเวลาของเฟรมสำหรับข้อมูลโลว์สปีดและฟูลสปีดแต่ละเฟรมจะมีค่าเท่ากับ 1 มิลลิวินาที สำหรับการส่งถ่ายข้อมูลแบบไฮสปีดโฮสจะแบ่งเฟรมแต่ละเฟรมออกเป็น 8 ส่วน โดยให้ชื่อแต่ละส่วนว่าไมโครเฟรมและมีช่วงเวลาเท่ากัน 125 มิลลิวินาที แต่ละเฟรมหรือไมโครเฟรมจะเริ่มต้นด้วยตัวเริ่มเฟรมหรือเรียกสั้นๆ ว่าSOF (Start-of-Frame)

การส่งถ่ายข้อมูลแต่ละชนิดจะประกอบไปด้วยการสื่อสารหรือเรียกทับศัพท์ว่าทรานแซคชัน (Transaction) มากกว่า 1 ชุด สำหรับการส่งถ่ายข้อมูลแบบคอนโทรลจะมีทรานแซคชันมากกว่า 1 ชุดเสมอเนื่องจากพวกมันจะมีขั้นตอนหรือสเตจ (Stage) มากกว่า 1 สเตจ ซึ่งแต่ละสเตจอาจประกอบไปด้วยทรานแซคชันมากกว่า 1 ชุด ส่วนการส่งถ่ายข้อมูลแบบอื่นๆนั้น จะใช้ทรานแซคชันหลายชุดก็ต่อเมื่อมันมีข้อมูลมากเกินกว่าที่จะใส่ลงไปในทรานแซคชันเดียวได้ ทรานแซคชันของการส่งถ่ายข้อมูลอาจจะทำได้เสร็จภายใน 1 เฟรมหรือไมโครเฟรม หรืออาจต้องแบ่งข้อมูลออกเป็นส่วนๆ แล้วส่งไปกับเฟรมหรือไมโครเฟรมหลายๆ ชุด ขึ้นอยู่กับการจัดเวลาของโฮสให้ทรานแซคชันและความเร็วในการตอบสนองของอุปกรณ์



รูปที่ 2.9 การจัดการข้อมูลภายในเฟรมของ USB

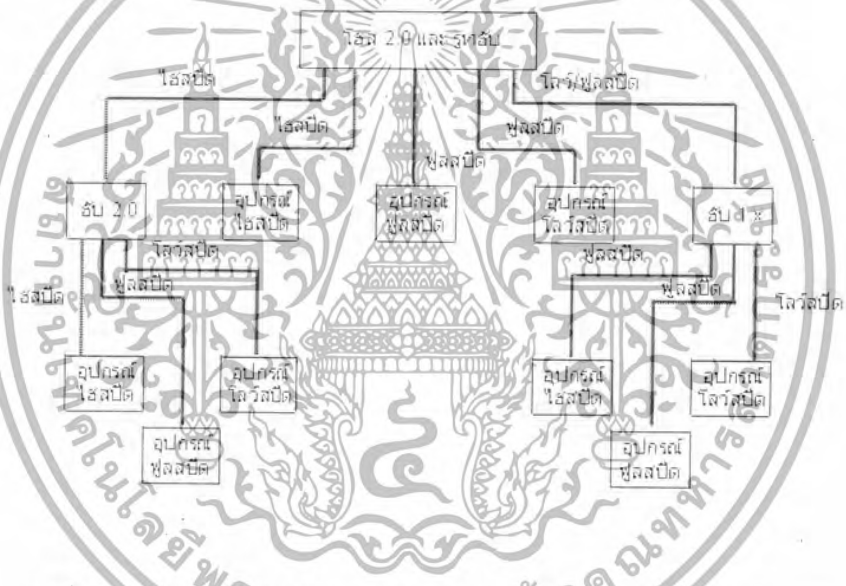
เนื่องจากการส่งถ่ายข้อมูลทุกครั้งจะต้องมีการใช้เส้นทางข้อมูลร่วมกัน ดังนั้นทรานแซคชันแต่ละชุดจะต้องมีแอดเดรสของอุปกรณ์โฮสอยู่ด้วย อุปกรณ์ทุกตัวจะมีแอดเดรสของมันเองที่ไม่ซ้ำกับตัวอื่นซึ่งถูกกำหนดมาจากโฮส ทรานแซคชันแต่ละชุดจะเริ่มต้นขึ้นเมื่อโฮสส่งบล็อคข้อมูลซึ่งมีแอดเดรสของอุปกรณ์และตำแหน่งเฉพาะหรือเอนด์พอยน์ภายในอุปกรณ์ที่ใช้รับข้อมูลรวมอยู่ด้วย ทุกสิ่งทุกอย่างที่ส่งออกมาจากตัวอุปกรณ์จะเป็นการตอบสนองต่อการร้องขอที่รับเข้ามาจากโฮสเพื่อเป็นการส่งข้อมูลหรือสถานะของอุปกรณ์กลับออกไป

2.5.2 ความเร็วการส่งถ่ายข้อมูลของโฮสและบัส

โฮสในเวอร์ชัน 1.x (เวอร์ชัน 1.0 และ 1.1) จะสนับสนุนโหมดการทำงานโลว์สปีดและฟูลสปีด ส่วนโฮส 2.0 จะต้องสนับสนุนโหมดการทำงานทั้งโลว์สปีด ฟูลสปีด และ ไฮสปีด

ฮับ 1.x จะไม่มีการแปลงความเร็วของการส่งข้อมูล แต่มันจะทำหน้าที่เพียงแค่เป็นตัวผ่านการส่งข้อมูลที่รับเข้ามา และทำการปรับอัตราขอบของสัญญาณ (Edge Rate) เพื่อให้สอดคล้องกับความเร็วของอุปกรณ์ปลายทางเท่านั้น ส่วนฮับ 2.0 จะทำหน้าที่แปลงความเร็วของโหมดไฮสปีดและโลว์สปีดหรือฟูลสปีดในกรณีที่จำเป็น นอกจากนี้ยังทำฟังก์ชันอื่นๆ ซึ่งช่วยให้การใช้เวลาของบัสมีประสิทธิภาพมากยิ่งขึ้นด้วย ความสามารถที่ถูกเพิ่มเข้าไปในฮับ 2.0 นี้เป็นเหตุผลที่ทำให้บัสแบบไฮสปีดยังคงเข้ากันได้กับฮาร์ดแวร์ 1.x อยู่

การขนถ่ายข้อมูลในส่วนใด ๆ ของบัสจะมีความเร็วเป็นไฮสปีดได้ก็ต่อเมื่อโฮสคอน-โทรลเลอร์และอัพสตรีมฮับทั้งหมดเป็น 2.0 เท่านั้น จากรูปที่ 2.10 ระบบบัสแบบไฮสปีดอาจจะมีฮับ 1.x ต่ออยู่และอาจมีบัสดาวน์สตรีมบางส่วนเป็นโลว์สปีดหรือฟูลสปีด การขนส่งข้อมูลที่มาจากอุปกรณ์โลว์สปีดและฟูลสปีดจะเดินทางที่โหมดไฮสปีดได้ในส่วนที่เป็นจุดเชื่อมโฮสและฮับ 2.0 เท่านั้น ซึ่งในส่วนนี้จะต้องไม่มีฮับ 1.x ต่อขั้นอยู่ ในการขนส่งข้อมูลระหว่างฮับ 2.0 และ 1.x หรืออุปกรณ์โลว์/ฟูลสปีดตัวอื่นๆ ข้อมูลจะเดินทางที่ความเร็วโลว์สปีดหรือฟูลสปีดเท่านั้น ส่วนในระบบบัสที่ใช้โฮสคอนโทรลเลอร์ 1.x เป็นศูนย์กลางในการสื่อสาร ระบบนั้นก็จะสนับสนุนเพียงแค่โหมดโลว์สปีดและฟูลสปีดเท่านั้นแม้ว่าจะมีอุปกรณ์หรือฮับไฮสปีดต่ออยู่ภายในระบบก็ตาม



รูปที่ 2.10 ความเร็วในการทำงานของอุปกรณ์ที่มีความเร็วในระบบบัส 2.0

2.6 องค์ประกอบของการส่งถ่ายข้อมูล

การส่งถ่ายข้อมูลแต่ละชนิดของระบบบัส USB ประกอบขึ้นจากทรานแซคชัน (Transaction) ซึ่งแต่ละทรานแซคชันจะถูกสร้างขึ้นมาจากแพ็กเก็ต โดยที่แต่ละแพ็กเก็ตจะบรรจุข้อมูลเอาไว้ แต่ก่อนจะกล่าวถึงรายละเอียดของทรานแซคชันนั้นเราจะต้องทำความเข้าใจกับองค์ประกอบพื้นฐานที่สำคัญของการสื่อสารก่อน นั่นคือ เอนด์พอยน์ และไปป์

2.6.1 ดีไวซ์เอนด์พอยน์ (Device Endpoint)

การสื่อสารทุกครั้งข้อมูลจะต้องเดินทางออกจากดีไวซ์เอนด์พอยน์ หรือถูกส่งเข้าไปในดีไวซ์เอนด์พอยน์ ซึ่งเอนด์พอยน์คือบัฟเฟอร์ซึ่งใช้สำหรับเก็บไบต์ข้อมูล ตามปกติแล้วจะเป็นกลุ่มของหน่วยความจำหรือเป็นรีจิสเตอร์ซึ่งอยู่ภายในตัวคอนโทรลเลอร์ชิป ข้อมูลที่เก็บอยู่ภายในดีไวซ์เอนด์พอยน์อาจเป็นข้อมูลที่รับเข้ามาหรืออาจจะเป็นข้อมูลที่รอจะส่งออกไป สำหรับทางด้านโฮสนั้นจะไม่มีเอนด์พอยน์แต่จะมีบัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาและสำหรับเก็บข้อมูลซึ่งพร้อมที่จะส่งออกไป โดยโฮสจะทำตัวเป็นเหมือนจุดเริ่มต้นของการสื่อสารกับดีไวซ์เอนด์พอยน์

ในข้อกำหนด USB ได้ให้คำจำกัดความของเอนด์พอยน์ไว้ว่า "เป็นส่วนหนึ่งของอุปกรณ์ซึ่งสามารถกำหนดตำแหน่งให้ไม่ซ้ำกับอุปกรณ์ตัวอื่นได้ ซึ่งมันสามารถเป็นได้ทั้งแหล่งจ่ายหรือรับข้อมูลของการสื่อสารที่อยู่ระหว่างโฮสและอุปกรณ์" คำจำกัดความนี้ชี้ให้เห็นได้ว่า การสื่อสารข้อมูลกับเอนด์พอยน์นั้นสามารถทำได้ในทางเดียวเท่านั้น แต่สำหรับคอนโทรลเอนด์พอยน์ จะเป็นกรณีพิเศษซึ่งมีการสื่อสารในลักษณะสองทิศทาง นั่นหมายความว่าถ้าเรามีคอนโทรลเอนด์พอยน์ IN เราก็จะมีคอนโทรลเอนด์พอยน์ OUT โดยอัตโนมัติ

เอนด์พอยน์แต่ละตัวจะต้องการแอดเดรสที่ไม่ซ้ำกับตัวอื่น ซึ่งแอดเดรสเหล่านี้จะประกอบไปด้วยหมายเลขเอนด์พอยน์และทิศทาง หมายเลขเอนด์พอยน์อาจมีค่าอยู่ในช่วง 0 ถึง 15 สำหรับทิศทางจะเป็นการมองจากมุมมองของโฮสซึ่ง IN จะมีทิศทางไปยังโฮส และ OUT จะเป็นทิศทางที่ออกจากโฮสเสมอ สำหรับเอนด์พอยน์ที่ถูกตั้งค่ามาให้ใช้การส่งถ่ายข้อมูลแบบคอนโทรลจะต้องส่งถ่ายข้อมูลทั้งสองทิศทาง ดังนั้นคอนโทรลเอนด์พอยน์จึงประกอบมาจากคู่ของเอนด์พอยน์ IN และ OUT ซึ่งใช้หมายเลขเอนด์พอยน์ร่วมกันนั่นคือ เอนด์พอยน์ศูนย์

อุปกรณ์ทุกตัวจะมีเอนด์พอยน์ศูนย์ซึ่งถูกตั้งมาให้เป็นคอนโทรลเอนด์พอยน์ ตามปกติเรามักใช้คอนโทรลเอนด์พอยน์เพียงแค่ตัวเดียวเท่านั้น แต่หากต้องการใช้คอนโทรลเอนด์พอยน์มากกว่านั้นก็สามารถทำได้และมีคอนโทรลเลอร์ชิปบางตัวที่สนับสนุนการทำงานในลักษณะนี้

การส่งถ่ายข้อมูลชนิดอื่นๆ จะสื่อสารข้อมูลในทิศทางเดียวกันเท่านั้น ซึ่งแอดเดรสของเอนด์พอยน์หนึ่งๆ สามารถกำหนดให้เป็น IN และ OUT แต่มันจะอยู่แยกกัน เช่น เอนด์พอยน์ 1 (EP1) บนตัวอุปกรณ์อาจกำหนดตำแหน่งให้เป็นเอนด์พอยน์ IN (EP1 IN) สำหรับส่งถ่ายข้อมูลไปยังโฮส และกำหนดตำแหน่งให้เป็นเอนด์พอยน์ OUT (EP1 OUT) สำหรับการส่งถ่ายข้อมูลจากโฮสได้ด้วย

นอกจากเอนด์พอยน์ศูนย์แล้ว อุปกรณ์ฟูลสปีดยังสามารถมีเอนด์พอยน์เพิ่มเติมได้อีกถึง 30 เอนด์พอยน์ (มีค่าแอดเดรส 1 ถึง 15 ซึ่งแต่ละแอดเดรสสนับสนุนทั้ง IN และ OUT) ส่วนอุปกรณ์โลว์สปีดจะมีจำนวนเอนด์พอยน์เพิ่มเติมได้อีกเพียง 2 เอนด์พอยน์เท่านั้นโดยการจับคู่กันด้านทิศทาง เช่น เอนด์พอยน์ 1 IN และเอนด์พอยน์ 1 OUT หรือเอนด์พอยน์ 1IN และเอนด์พอยน์ 2 IN

ทุกๆทรานแซคชันที่อยู่บนบัสจะมีการใส่หมายเลขเอนด์พอยน์และโค้ดซึ่งเป็นตัวบอกทิศทางการไหลของข้อมูลเอาไว้ หรืออาจมีโค้ดเพื่อบ่งบอกว่าทรานแซคชันนี้เป็นจุดเริ่มของการส่งถ่ายข้อมูลแบบคอนโทรล โค้ดดังกล่าวได้แก่ IN, OUT และ SETUP ดังตารางที่ 2.3

ชนิดของทรานแซคชัน	แหล่งของข้อมูล	ชนิดของการส่งถ่ายข้อมูลซึ่งใช้ทรานแซคชันชนิดนี้	สิ่งที่อยู่ภายในทรานแซคชัน
IN	อุปกรณ์	ทุกชนิด	ข้อมูลทั่วไป
OUT	โฮส	ทุกชนิด	ข้อมูลทั่วไป
SETUP	โฮส	คอนโทรล	คำร้องขอ

ตารางที่ 2.3 โค้ดบอกทิศทางการไหลข้อมูลของทรานแซคชัน

การให้ชื่อทิศทางของเอนด์พอยน์สำหรับทรานแซคชัน IN และ OUT จะยึดเอามุมมองของโฮสเป็นหลัก นั่นคือในทรานแซคชัน IN ข้อมูลจะเดินทางจากอุปกรณ์รอบข้างไปยังโฮส ส่วนทรานแซคชัน OUT ข้อมูลจะเดินทางไปยังอุปกรณ์รอบข้าง

สำหรับทรานแซคชัน SETUP ข้อมูลจะเดินทางจากโฮสไปยังอุปกรณ์รอบข้างเช่นกัน แต่ทรานแซคชัน SETUP จะเป็นกรณีพิเศษเนื่องจากทรานแซคชันนี้เป็นตัวริเริ่มการส่งถ่ายข้อมูลคอนโทรล อุปกรณ์ USB ทุกตัวต้องรู้จักและเข้าใจทรานแซคชัน SETUP เพื่อให้มันทราบถึงวิธีแปลความหมายของข้อมูลซึ่งบรรจุอยู่ภายใน นอกจากนี้ทรานแซคชัน SETUP ยังเป็นเพียงชนิดเดียวที่อุปกรณ์จะต้องเป็นฝ่ายรับเสมอ ซึ่งการส่งถ่ายข้อมูลชนิดใดๆ อาจใช้ทรานแซคชัน IN หรือ OUT ก็ได้ แต่การส่งถ่ายข้อมูลคอนโทรลจะต้องใช้ทรานแซคชัน SETUP เท่านั้น

ในแต่ละทรานแซคชันจะมีแอดเดรสของอุปกรณ์และแอดเดรสของเอนด์พอยน์บรรจุเอาไว้เมื่ออุปกรณ์ได้รับทรานแซคชัน OUT หรือ SETUP ซึ่งบรรจุแอดเดรสของอุปกรณ์ตัวนั้นเอาไว้ ฮาร์ดแวร์ของมันจะเก็บข้อมูลที่รับเข้ามาไว้ในเอนด์พอยน์ที่เหมาะสม และทำการกระตุ้นอินเทอร์รัพท์ จากนั้นโปรแกรมบริการอินเทอร์รัพท์ ในตัวอุปกรณ์จะประมวลผลข้อมูลที่รับเข้ามาและทำงานต่างๆ ตามที่ทรานแซคชันนั้นๆ ต้องการ ในกรณีที่อุปกรณ์ได้รับทรานแซคชัน IN ซึ่งบรรจุแอดเดรสที่ตรงกับอุปกรณ์ตัวนั้นและถ้ามันมีข้อมูลที่พร้อมจะส่งไปยังโฮส ฮาร์ดแวร์จะส่งข้อมูลจากเอนด์พอยน์ที่ถูกระบุมาลงไปบนบัสและทำการกระตุ้นอินเทอร์รัพท์ จากนั้นโปรแกรมบริการอินเทอร์รัพท์ที่อยู่ในอุปกรณ์จะทำงานต่างๆ ที่จำเป็นเพื่อให้พร้อมสำหรับทรานแซคชัน IN ถัดไป

2.6.2 ไปป์ (Pipe)

ก่อนที่การส่งถ่ายข้อมูลจะเกิดขึ้น โฮสและอุปกรณ์ต้องสร้างไปป์ขึ้นมาก่อน ไปป์ในระบบบัส USB นั้นไม่ใช่วัตถุที่สามารถจับต้องได้ทางกายภาพแต่มันจะเป็นการเชื่อมต่อทางลอจิกระหว่างโฮสและเอนด์พอยน์ ซึ่งใช้ทำงานร่วมกันระหว่างเอนด์พอยน์ของอุปกรณ์และซอฟต์แวร์ของโฮสคอนโทรลเลอร์

โฮสจะสร้างไปป์ขึ้นในกรณีที่มีการจ่ายไฟให้แก่ระบบ หรือการเสียบอุปกรณ์เข้ากับระบบบัสและในกรณีที่มีการร้องขอข้อมูลคอนฟิกิวเรชันจากอุปกรณ์ และโฮสจะทำลายไปป์ที่ไม่ได้ใช้แล้วออกไป เมื่ออุปกรณ์ถูกถอดออกจากระบบบัสนอกจากนี้โฮสจะร้องขอไปป์ชุดใหม่ หรือทำลายไปป์ที่ไม่ได้ใช้แล้วออกไปที่เวลาใดๆ ได้เมื่อมีการร้องขอคอนฟิกิวเรชันหรืออินเทอร์เฟสชุดใหม่ของอุปกรณ์ อุปกรณ์ USB ทุกตัวจะมีไปป์ที่เรียกว่าดีฟอลต์คอนโทรลไปป์ (Default Control Pipe) ซึ่งประกอบขึ้นจากเอนด์พอยน์ 2 ชุดนั่นคือ เอนด์พอยน์ศูนย์ IN และเอนด์พอยน์ศูนย์ OUT โดยที่ไปป์นี้จะถูกใช้สำหรับการตั้งค่าการทำงานให้แก่อุปกรณ์

ข้อมูลคอนฟิกิวเรชันที่โฮสรับเข้ามาจะมีดีสคริปเตอร์ของเอนด์พอยน์แต่ละชุดซึ่งอุปกรณ์ต้องการใช้ โดยเอนด์พอยน์ดีสคริปเตอร์แต่ละชุดจะใช้แจ้งถึงข้อมูลของเอนด์พอยน์นั้นๆ ให้แก่โฮสเพื่อใช้ในการสื่อสารกับมัน ซึ่งข้อมูลนี้ประกอบด้วยแอดเดรสของเอนด์พอยน์, ชนิดของการส่งถ่ายข้อมูลที่ใช้, ขนาดสูงสุดของคาต้าเพย์โหลด และช่วงเวลาที่ต้องการสำหรับการส่งถ่ายข้อมูล

ในบางกรณีโฮสจะรับคำร้องขอคอนฟิกิวเรชันมาส่งจากแม่ใจแล้วว่าบัสมีแบนด์วิดธ์ว่างเพียงพอที่จะทำการส่งถ่ายข้อมูลในอัตราที่ต้องการ ซึ่งกรณีนี้จะเกิดขึ้นเมื่อคอนฟิกิวเรชันนั้นร้องขอไปป์สำหรับเป็นสื่อในการส่งถ่ายข้อมูลแบบไอโซโครนัสซึ่งมีการรับประกันอัตราการส่งข้อมูลและการส่งถ่ายข้อมูลแบบอินเทอร์รัพท์ซึ่งมีการรับประกันค่าเวลาสูงสุดระหว่างทรานแซคชัน ซึ่งในกรณีนี้โฮสจะทำการตรวจสอบแบนด์วิดธ์ก่อนที่จะสร้างไปป์ขึ้นมา ถ้าแบนด์วิดธ์ของระบบบัสมีเพียงพอโฮสจะรับคำร้องขอคอนฟิกิวเรชัน ซึ่งวิธีการนี้ทำให้มั่นใจว่าการส่งถ่ายข้อมูลจะใช้เวลาตามที่มันต้องการแต่ถ้าแบนด์วิดธ์มีไม่เพียงพอโฮสจะปฏิเสธคำร้องขอคอนฟิกิวเรชันนั้นไป และซอฟต์แวร์ที่ทำการร้องขอนั้นจะต้องพยายามทำการร้องขอใหม่อีกครั้ง ซึ่งอาจต้องรอจนกระทั่งแบนด์วิดธ์ของระบบมีพอเพียง หรืออาจเป็นการเลือกคอนฟิกิวเรชันใหม่ซึ่งใช้แบนด์วิดธ์ที่น้อยกว่าสำหรับไปป์ซึ่งเป็นสื่อสำหรับการร้องขอการส่งถ่ายข้อมูลที่ไม่ต้องการรับประกันทางด้านเวลา โฮสจะไม่ทำการตรวจสอบแบนด์วิดธ์ที่เหลืออยู่แต่มันเพียงตอบรับว่าจะจัดการส่งถ่ายข้อมูลที่ร้องขอมานั้นลงไปในแบนด์วิดธ์ที่เหลืออยู่ให้ดีที่สุดเท่าที่มันสามารถทำได้

ในข้อกำหนด USB ได้กำหนดชนิดของไปป์ 2 ชนิด ได้แก่ เมสเสจไปป์ และสตรีมไปป์ ซึ่งในการใช้งานจะสอดคล้องกับทิศทางการเดินของข้อมูลว่าเป็นหนึ่งหรือสองทิศทาง โดยเมสเสจ

ไปป์เป็นไปป์สองทิศทางซึ่งใช้กับการส่งถ่ายข้อมูลแบบคอนโทรลเพียงชนิดเดียวเท่านั้น ส่วนสตรีมไปป์จะเป็นไปป์ที่มีทิศทางเดียวซึ่งใช้กับการส่งถ่ายข้อมูลแบบอื่นๆ นอกเหนือจากการส่งถ่ายข้อมูลแบบคอนโทรล

2.6.3 เมสเสจไปป์ (Message Pipe)

ในข้อกำหนด USB ได้มีการกำหนดรูปแบบของข้อมูลที่ถูกส่งบนไปป์ชนิดนี้เอาไว้ โดยมันจะถูกควบคุมจากโฮสซึ่งเริ่มต้นด้วยการส่งคำร้องขอจากโฮส จากนั้นข้อมูลจะถูกส่งถ่ายไปในทิศทางที่ต้องการตามที่ได้กำหนดไว้ในคำร้องขอ เมสเสจไปป์จะอนุญาตให้ข้อมูลไหลได้ทั้งสองทิศทางแต่ไปป์ชนิดนี้จะสนับสนุนการถ่ายโอนข้อมูลแบบคอนโทรลเท่านั้น และดีฟอลต์คอนโทรลไปป์จะเป็นเมสเสจไปป์เสมอ

2.6.4 สตรีมไปป์ (Stream Pipe)

ข้อกำหนด USB ไม่ได้กำหนดรูปแบบของข้อมูลที่ส่งไปบนไปป์ชนิดนี้เอาไว้ ดังนั้นเราจึงสามารถส่งข้อมูลชนิดใดก็ได้ลงบนสตรีมไปป์ อุปกรณ์ที่เป็นตัวรับจะรับสิ่งที่เดินทางเข้ามา จากนั้นเฟิร์มแวร์ของอุปกรณ์หรือซอฟต์แวร์ของโฮสจะประมวลผลข้อมูลด้วยวิธีที่เหมาะสมกับงาน

แม้ว่าจะเป็นการสื่อสารด้วยข้อมูลแบบสตรีมก็ตาม แต่อุปกรณ์ซึ่งเป็นตัวรับหรือส่งจะต้องมีการกำหนดรูปแบบเอาไว้ก่อน เช่น แอพพลิเคชันซอฟต์แวร์ของโฮสอาจจะกำหนดโค้ดซึ่งร้องขอให้อุปกรณ์ส่งชุดของไบต์ข้อมูลที่เป็นตัวแทนถึงการอ่านอุณหภูมิและเวลาที่อ่านค่าออกมา แม้ว่าโฮสจะสามารถใช้การส่งถ่ายข้อมูลคอนโทรลกับคำร้องขอที่ผู้ผลิตเป็นผู้กำหนดขึ้นมาเช่น Get_Temperature ได้ก็ตาม แต่กรณีนี้การใช้การส่งถ่ายข้อมูลแบบอินเทอร์รัพท์จะเหมาะสมกว่า ทั้งนี้เพื่อเป็นการรับประกันว่าโฮสจะร้องขอการรอบการถ่ายข้อมูลตามช่วงเวลาที่กำหนด

2.7 การเริ่มส่งถ่ายข้อมูล

เมื่อดีไวซ์ไดร์ฟเวอร์ซึ่งอยู่ที่โฮสต้องการติดต่อสื่อสารกับอุปกรณ์ มันจะเริ่มเข้าสู่กระบวนการส่งถ่ายข้อมูล ในข้อกำหนดได้กำหนดให้การส่งถ่ายข้อมูลเป็นกระบวนการของการสร้างและการกระทำตามคำร้องขอที่เกิดจากการสื่อสาร ในกรณีที่การส่งไบต์ข้อมูลที่มีจำนวนเล็กน้อยการส่งถ่ายข้อมูลนี้อาจจะสั้นมาก และถ้าเป็นการส่งข้อมูลซึ่งเป็นไฟล์ขนาดใหญ่การส่งถ่ายข้อมูลนั้นจะยาวนานขึ้น

ตามปกติแล้วแอพพลิเคชันซอฟต์แวร์ของวินโดว์จะเปิดการติดต่อสื่อสารกับอุปกรณ์ด้วยการใช้ฟังก์ชัน API มาตรฐาน ในการเริ่มต้นการสื่อสารนั้นแอพพลิเคชันซอฟต์แวร์จะเรียกฟังก์ชัน

API เพื่อร้องขอการส่งถ่ายข้อมูลจากดีไวซ์ไดร์ฟเวอร์ แอพพลิเคชันซอฟต์แวร์สามารถร้องขอข้อมูลจากอุปกรณ์หรือเป็นตัวจ่ายข้อมูลให้แก่อุปกรณ์ก็ได้ เมื่อแอพพลิเคชันซอฟต์แวร์ร้องขอการส่งถ่ายข้อมูลระบบปฏิบัติการจะผ่านคำร้องขอไปยังดีไวซ์ไดร์ฟเวอร์ที่เหมาะสม ซึ่งจะส่งคำร้องขอเหล่านี้ไปยังไดร์ฟเวอร์อื่นๆ ที่อยู่ในระดับระบบและส่งต่อไปยังโฮสคอนโทรลเลอร์ จากนั้นโฮสคอนโทรลเลอร์จึงเริ่มทำการส่งถ่ายข้อมูลบนระบบบัส

ในบางกรณีไดร์ฟเวอร์จะถูกตั้งค่าไว้เพื่อให้ร้องขอการส่งถ่ายข้อมูลทุกช่วงเวลา และแอพพลิเคชันซอฟต์แวร์จะอ่านข้อมูลที่รับเข้ามาหรือจ่ายข้อมูลออกไปกับการส่งถ่ายข้อมูลเหล่านี้ส่วนการส่งถ่ายข้อมูลชนิดอื่นๆ เช่น การส่งถ่ายข้อมูลที่ใช้ในกระบวนการอินิวเมอร์เรท จะเริ่มต้นขึ้นโดยระบบปฏิบัติการเมื่อมันตรวจพบอุปกรณ์ที่ถูกเพิ่มเข้ามาในระบบ

2.8 รูปแบบของแพ็กเก็ต

ภายใน USB แพ็กเก็ตจะประกอบไปด้วยฟิลด์ต่างๆ ซึ่งแต่ละฟิลด์จะมีข้อมูลเฉพาะตัวตามชนิดของมัน ชนิดของฟิลด์นั้นได้แก่ SYNC, PID, แอดเดรส, เอนด์พอยน์, หมายเลขเฟรม, ดาต้า หรือข้อมูล และ CRC

ชื่อ	ขนาด (บิต)	ชนิดของแพ็กเก็ต	จุดประสงค์การใช้งาน
SYNC	8	ทุกชนิด	ใช้เริ่มแพ็กเก็ตและการเข้าจังหวะ
PID	8	ทุกชนิด	แสดงชนิดของแพ็กเก็ต
แอดเดรส	7	IN, OUT, SETUP	แสดงแอดเดรสของฟังก์ชัน
เอนด์พอยน์	4	IN, OUT, SETUP	แสดงเอนด์พอยน์
หมายเลขเฟรม	11	SOF	แสดงเฟรม
ดาต้า	0 ถึง 8192 (1024 ไบต์) สำหรับ ฮาร์ดแวร์ 2.0 และ 0 ถึง 8184 (1023 ไบต์) สำหรับ ฮาร์ดแวร์ 1.x	DATA0, DATA1	ข้อมูล
CRC	5 หรือ 16	IN, OUT, SETUP, DATA0, DATA1	ตรวจจับความผิดพลาด

ตาราง 2.4 ฟิลด์ต่างๆ ภายในแพ็กเก็ต

2.8.1 ฟิลด์ SYNC

แพ็กเก็ตทุกชุดต้องเริ่มต้นด้วยฟิลด์ SYNC เสมอ ฟิลด์นี้มีขนาด 8 บิต ใช้สำหรับเข้าจังหวะสัญญาณนาฬิกาของตัวรับและตัวส่ง

2.8.2 ฟิลด์ PID (Packet Identifier Field)

ฟิลด์ PID มีขนาด 8 บิต โดยบิต 0 ถึงบิต 3 เป็นตัวบอกชนิดของแพ็กเก็ต และบิต 4 ถึงบิต 7 จะเป็นคอมพลิเมนต์ที่ 1 หรืออินเวอร์สของบิต 0 ถึงบิต 3 ซึ่งใช้สำหรับตรวจสอบความผิดพลาดดังรูปที่ 23 ในข้อกำหนด USB ได้มีการกำหนดโค้ดของ PID ไว้ 16 ตัวแยกเป็นกลุ่มของโทเค็น, ดาต้า, แฮนด์เช็ก และแพ็กเก็ตพิเศษ

ชนิดของ PID	ค่า PID	คำอธิบาย
โทเค็น	0001	โทเค็น OUT
	1001	โทเค็น IN
	0101	โทเค็น SOF
	1101	โทเค็น SETUP
ดาต้า	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
แฮนด์เช็ก	0010	แฮนด์เช็ก ACK
	1010	แฮนด์เช็ก NAK
	1110	แฮนด์เช็ก STALL
	0110	NYET (No Response Yet)
พิเศษ	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping

ตาราง 2.5 ชนิดของ PID

PID_0	PID_1	PID_2	PID_3	$nPID_0$	$nPID_1$	$nPID_2$	$nPID_3$

รูปที่ 2.11 รูปแบบของ PID

2.8.3 ฟิลด์ ADDR (Address Field)

ฟิลด์ ADDR หรือ ฟิลด์แอดเดรสใช้สำหรับระบุว่าแพ็กเก็ตนั้นๆ ถูกกำหนดให้ใช้สำหรับอุปกรณ์ตัวไหน เนื่องจากความยาวของฟิลด์นี้มีค่าเท่ากับ 7 บิต จึงทำให้มันสามารถสนับสนุนอุปกรณ์ได้ 127 ตัว สำหรับแอดเดรส 0 จะไม่มีการใช้งาน อุปกรณ์บางตัวที่ยังไม่ได้รับการกำหนดแอดเดรสจะต้องตอบสนองกับต่อแพ็กเก็ตที่ถูกส่งไปยังแอดเดรส 0

2.8.4 ฟิลด์ ENDP (Endpoint Field)

ฟิลด์ ENDP หรือฟิลด์เอนด์พอยน์มีขนาด 4 บิตใช้สำหรับบอกถึงหมายเลขเอนด์พอยน์ที่อยู่ภายในอุปกรณ์ ซึ่งอุปกรณ์โลว์สปีดสามารถมีหมายเลขเอนด์พอยน์ได้ไม่เกิน 3 หมายเลข ส่วนอุปกรณ์ฟูลสปีดและไฮสปีดสามารถมีหมายเลขเอนด์พอยน์ได้16 หมายเลข

2.8.5 ฟิลด์หมายเลขเฟรม (Frame Number Field)

ฟิลด์หมายเลขเฟรมมีขนาด 11 บิต ทำหน้าที่เป็นตัวระบุหมายเลขเฟรม โฮสจะส่งฟิลด์นี้ไปในแพ็กเก็ต SOF ซึ่งเป็นตัวเริ่มต้นเฟรมหรือไมโครเฟรม ตัวเลขมีค่าอยู่ระหว่าง 0 ถึง 7FFh (ถ้ามีค่าเกิน 7FFh ตัวนับจะวนกลับมาที่ 0 ใหม่อีกครั้ง) โฮสแบบฟูลสปีดจะใช้ตัวนับ 11 บิตซึ่งมีค่าเพิ่มขึ้น 1 ครั้งต่อเฟรม ส่วนโฮสแบบไฮสปีดจะใช้ตัวนับ 14 บิตและมีค่าเพิ่มขึ้น 1 ต่อไมโครเฟรม แต่จะมีเพียงบิต 3-13 ของตัวนับไมโครเฟรมเท่านั้นที่ถูกส่งลงไปภายในฟิลด์หมายเลขเฟรมนี้ ดังนั้นหมายเลขเฟรมจะเพิ่มขึ้น 1 ครั้งต่อ 1 เฟรม หรือ 8 ไมโครเฟรม

2.8.6 ฟิลด์ดาต้า (Data Field)

ฟิลด์ดาต้านี้อาจมีขนาดอยู่ระหว่าง 0 ถึง 1024 ไบต์ ขึ้นอยู่กับชนิดของการส่งถ่ายข้อมูล ปริมาณของข้อมูลในทรานแซคชัน และเวอร์ชันของ USB ซึ่งอุปกรณ์ตัวนั้นสนับสนุน

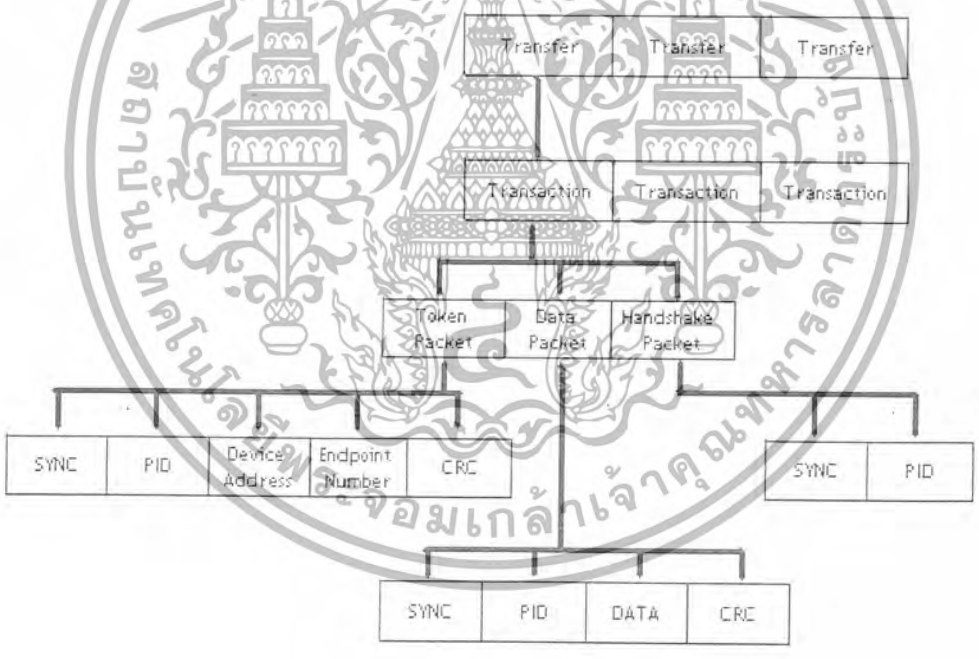
2.8.7 ฟิลด์ CRC (CRC Field)

ฟิลด์ CRC (Cyclic Redundancy Check) ใช้สำหรับตรวจสอบความผิดพลาดของการส่งข้อมูล โดยฮาร์ดแวร์ของตัวส่งจะแทรกบิต CRC และทางด้านรับจะมีฮาร์ดแวร์สำหรับการคำนวณใน

ลักษณะเดียวกับตัวส่ง . สำหรับโทเค็นแพ็กเก็ตซึ่งครอบคลุมฟิลด์แอดเดรสและเอนด์พอยน์จะมีขนาด 5 บิต ส่วนในกรณีที่ใช้กับดาต้าแพ็กเก็ตจะมีขนาด 16 บิต

2.9 ทรานแซคชัน (Transactions)

องค์ประกอบของการสื่อสารข้อมูล หรือ ทรานแซคชัน แสดงดังรูปที่ 24 ส่วนในตารางที่ 2.6 เป็นรายการขององค์ประกอบซึ่งรวมตัวกันเป็นการส่งถ่ายข้อมูลทั้งสี่แบบ ที่ผ่านมาเราอาจจะพบกับคำศัพท์ที่มีความหมายคล้ายๆ กันมาบ้างแล้ว เช่น การส่งถ่ายข้อมูล หรือ การทรานสเฟอร์ (Transfer) กับ ทรานแซคชัน, ขั้นตอนหรือสเตจ (Stage) กับ เฟส (Phase), ดาต้าทรานแซคชัน (Data Transaction) กับ ดาต้าแพ็กเก็ต (Data Packet), สเตตัสสเตจ (Status Stage) กับ แฮนด์เช็กเฟส (Handshake Phase) เป็นต้น ซึ่งตารางที่ 7 จะช่วยลดความสับสนของคำเหล่านี้ลงไปได้



รูปที่ 2.12 องค์ประกอบของทรานแซคชัน

การส่งถ่ายข้อมูลแต่ละชนิดจะสร้างขึ้นจากทรานแซคชัน 1 ทรานแซคชันหรือมากกว่านั้น และแต่ละทรานแซคชันจะประกอบไปด้วยแพ็กเก็ต 1, 2 หรือ 3 แพ็กเก็ต

ทรานแซคชันทั้ง 3 ชนิดนี้กำหนดขึ้นจากจุดประสงค์ของการใช้งานและทิศทางการไหลของข้อมูล สำหรับเซ็ตอัพทรานแซคชันจะใช้สำหรับส่งคำร้องขอของการส่งถ่ายข้อมูลคอนโทรล

ไปยังอุปกรณ์ ส่วนทรานแซคชัน IN เป็นการรับข้อมูลจากอุปกรณ์ และทรานแซคชัน OUT จะใช้สำหรับส่งข้อมูลต่างๆ ไปยังอุปกรณ์ ในข้อกำหนด USB ได้กำหนดให้ทรานแซคชันเป็นการส่งการบริการหรือเซอร์วิซ (Service) ไปยังเอนด์พอยน์ ซึ่งเซอร์วิซในที่นี้สามารถหมายความถึงการส่งก้อนข้อมูลของโฮสไปยังอุปกรณ์ หรือการร้องขอของอุปกรณ์และการรับก้อนข้อมูลจากอุปกรณ์

ชนิดของการส่งถ่ายข้อมูล	สเตจ (0 ทรานแซคชัน หรือมากกว่านั้น)	เฟส (แพ็กเก็ต) สำหรับแพ็กเก็ตโลว์สปีดจะตามด้วยแพ็กเก็ต PRE ในแต่ละดาวน์สตรีม
คอนโทรล	เซ็ตอัพ	โทเค็น
		ดาต้า
		แฮนด์เชค
	ดาต้า (IN หรือ OUT) (อาจมีหรือไม่มีก็ได้)	โทเค็น
		ดาต้า
		แฮนด์เชค
	สเตตัส (IN หรือ OUT)	โทเค็น
		ดาต้า
		แฮนด์เชค
บัลค์	ดาต้า (IN หรือ OUT)	โทเค็น
		ดาต้า
		แฮนด์เชค
อินเทอร์รัพท์	ดาต้า (IN หรือ OUT)	โทเค็น
		ดาต้า
		แฮนด์เชค
ไอโซโครนัส	ดาต้า (IN หรือ OUT)	โทเค็น
		ดาต้า

ตารางที่ 2.6 องค์ประกอบของการส่งถ่ายข้อมูลแต่ละชนิด

ในแต่ละทรานแซคชันจะประกอบไปด้วย การแสดงชนิดของทรานแซคชัน (Identify), การตรวจสอบความผิดพลาด, สถานะหรือสเตตัส และข้อมูลคอนโทรล รวมถึงข้อมูลใดๆ ก็ตามที่ต้องการส่งถ่ายกัน การส่งถ่ายข้อมูลจนครบสมบูรณ์อาจต้องใช้เฟรมหลายเฟรมแต่เนื่องจากทราน

แซคชันเป็นการสื่อสารเดี่ยวซึ่งจะต้องต่อเนื่องอย่างสมบูรณ์ ดังนั้นจึงไม่มีการสื่อสารใดๆ บนระบบบัสที่สามารถหยุดทรานแซคชันไว้กลางทางได้

อุปกรณ์ต้องสามารถตอบสนองด้วยข้อมูลหรืออาจเป็นข้อมูลทางสถานะที่ถูกร้องขอมาในทรานแซคชันให้เร็วที่สุด เฟิร์มแวร์ที่อยู่ภายในตัวอุปกรณ์อาจมีการจัดเตรียมเอนด์พอยน์สำหรับตอบสนองต่อการร้องขอทรานแซคชันเอาไว้แล้ว แต่ฮาร์ดแวร์จะเป็นตัวจัดการการตอบสนองต่อคำร้องขอเมื่อมันถูกส่งมาถึง

การส่งถ่ายข้อมูลด้วยข้อมูลจำนวนน้อยๆ อาจใช้ทรานแซคชันเพียงแค่ครั้งเดียว แต่ถ้าข้อมูลมีขนาดใหญ่ การส่งถ่ายข้อมูลอาจต้องแบ่งข้อมูลออกเป็นส่วนๆ แล้วส่งไปกับทรานแซคชันหลายครั้ง

2.9.1 ทรานแซคชันเฟส (Transaction Phase)

ภายในทรานแซคชันแต่ละครั้งอาจประกอบไปด้วยส่วนซึ่งเกิดขึ้นเป็นลำดับ หรือเรียกว่าเฟส (Phase) ได้มากถึง 3 เฟส ได้แก่ โทเค็น, ดาต้า และแฮนด์เช็ก โดยแต่ละเฟสจะประกอบไปด้วยแพ็กเก็ตหนึ่งหรือสองแพ็กเก็ต ซึ่งแพ็กเก็ตคือกลุ่มของข้อมูลที่มีการกำหนดรูปแบบเอาไว้ แพ็กเก็ตทุกชุดจะเริ่มต้นด้วย หมายเลขแพ็กเก็ต (Packet ID: PID) สามารถอยู่ถัดจาก PID จะขึ้นอยู่กับชนิดของทรานแซคชัน ซึ่งอาจเป็นแอนด์พอยน์แอดเดรส, ดาต้า, ข้อมูลสเตตัส หรือหมายเลขเฟรมร่วมกับบิตตรวจสอบความผิดพลาด

ในส่วนของโทเค็นแฟสนั้นโฮสจะส่งคำร้องขอการสื่อสารลงไปในโทเค็นแพ็กเก็ต โดยที่ส่วน PID จะเป็นตัวบอกชนิดของทรานแซคชัน เช่น Setup, IN, OUT หรือ SOF (Start of Frame)

ในดาต้าเฟส โฮสหรืออุปกรณ์อาจส่งถ่ายข้อมูลชนิดใดๆ ก็ได้ลงในดาต้าแพ็กเก็ต โดยที่ส่วน PID จะเป็นตัวบอกถึงค่าดาต้าทือกเกิล (Data Toggle) ซึ่งใช้ในการบอกตำแหน่งของข้อมูลในกรณีที่มีดาต้าแพ็กเก็ตหลายชุด

ในแฮนด์เช็กเฟส โฮสหรืออุปกรณ์จะส่งข้อมูลสถานะ หรือแฮนด์เช็ก ลงในแฮนด์เช็กแพ็กเก็ต โดย PID จะเก็บโค้ดของสถานะเอาไว้ (ACK, NAK, STALL, NYET) บางครั้งในข้อกำหนด USB จะใช้คำว่าสเตตัสเฟส และสเตตัสแพ็กเก็ตเพื่ออ้างถึงแฮนด์เช็กเฟส และแฮนด์เช็กแพ็กเก็ต

นอกจากนี้โทเค็นเฟสอาจใช้ในการส่งแพ็กเก็ตเริ่มเฟรมหรือ SOF ซึ่งเป็นตัวอ้างอิงทางด้านเวลาโดยในระบบบัสฟูลสปีดโฮสจะส่งมันออกมาทุกๆ 1 มิลลิวินาที และในระบบบัสไฮสปีดโฮสจะส่งออกมาทุกๆ 125 ไมโครวินาที ภายในแพ็กเก็ตนี้ยังบรรจุหมายเลขเฟรมซึ่งเป็นตัวบอกถึงการนับเฟรม ซึ่งไมโครเฟรม 8 ชุดที่อยู่ภายในเฟรมเดียวกันจะมีหมายเลขเดียวกันทั้งหมด นอกจากนี้

SOF ยังเป็นตัวที่ช่วยรักษาอุปกรณ์จากการเข้าสู่สภาวะซัซเพนต์เมื่อไม่มีการเคลื่อนไหวของข้อมูลในระบบบัสอีกด้วย

ชนิดของแพ็กเก็ต	ชื่อ PID	ค่า	ใช้กับชนิดของการส่งถ่ายข้อมูล	แหล่งกำเนิด	ความเร็วบัส	คำอธิบาย
โทเค็น (แสดงชนิดของทรานแซคชัน)	OUT	0001	ทุกชนิด	โฮส	ทุกความเร็ว	เอนด์พอยน์แอดเดรสสำหรับทรานแซคชัน OUT (โฮสไปอุปกรณ์)
	IN	1001	ทุกชนิด	โฮส	ทุกความเร็ว	เอนด์พอยน์แอดเดรสสำหรับทรานแซคชัน IN (อุปกรณ์ไปโฮส)
	SOF	0101	SOF	โฮส	ทุกความเร็ว	ตัวบอกตำแหน่งเริ่มเฟรมและหมายเลขเฟรม
	SETUP	1101	คอนโทรล	โฮส	ทุกความเร็ว	เอนด์พอยน์แอดเดรสสำหรับทรานแซคชัน SETUP
ดาต้า (มีข้อมูลและโค้ดสถานะ)	DATA0	0011	ทุกชนิด	โฮส, อุปกรณ์	ทุกความเร็ว	ดาต้าที่จะเกิด, ลำดับข้อมูล
	DATA1	1011	ทุกชนิด	โฮส, อุปกรณ์	ทุกความเร็ว	ดาต้าที่จะเกิด, ลำดับข้อมูล
	DATA2	0111	ไอโซโครนัส	โฮส, อุปกรณ์	โฮสปิด	ลำดับข้อมูล
	MDATA	1111	ไอโซโครนัส, อินเทอร์รัพท์	โฮส, อุปกรณ์	โฮสปิด	ลำดับข้อมูล
แฮนด์เช็ก (มีโค้ดสถานะ)	ACK	0010	ทุกชนิด	โฮส, อุปกรณ์	ทุกความเร็ว	ผู้รับได้รับข้อมูลที่ปราศจากความผิดพลาด
	NAK	1010	คอนโทรล, บัลก์, อินเทอร์รัพท์	อุปกรณ์	ทุกความเร็ว	ผู้รับไม่สามารถรับข้อมูล หรือผู้ส่งไม่สามารถส่งข้อมูลหรือไม่มีข้อมูลที่จะส่ง
	STALL	1110	คอนโทรล, บัลก์, อินเทอร์รัพท์	อุปกรณ์	ทุกความเร็ว	ไม่มีการสนับสนุนคำร้องขอคอนโทรลหรือเอนด์พอยน์ถูกทำให้ฮอล์ท (Halt)

	NYET	0110	คอนโทรล Write, บัลก์ OUT, สปลิตทรานแซคชัน	อุปกรณ์	โฮสปีด	อุปกรณ์ได้รับดาต้าแพ็กเก็ตที่ไม่มีข้อผิดพลาด แต่ขณะนั้นมันยังไม่พร้อม หรือขณะนั้นฮับยังทำ สปลิตดาต้าไม่สมบูรณ์
ชนิดพิเศษ	PRE	1100	คอนโทรล, บัลก์, อินเทอร์รัพท์	โฮส	ฟูลสปีด	ตัวบอกล่วงหน้าซึ่งโฮสส่งออกมาเพื่อบอกว่าแพ็กเก็ตต่อไปจะเป็นโลว์สปีด
	ERR	1100	ทุกชนิด	ฮับ	ไฮสปีด	ถูกส่งกลับมาจากฮับเพื่อเป็นการรายงานความผิดพลาดของโลว์ หรือฟูลสปีดในการทำสปลิตทรานแซคชัน
	SPLIT	1000	ทุกชนิด	โฮส	ไฮสปีด	นำหน้าโทเค็นแพ็กเก็ตเพื่อเป็นการบอกว่ามันเป็นสปลิตทรานแซคชัน
	PING	0100	คอนโทรล Write, บัลก์ OUT	โฮส	ไฮสปีด	เช็กการไม่ว่างสำหรับทรานแซคชันบัลก์ OUT และคอนโทรล Write หลังจาก NYET
	สงวน	0000	-	-	-	สำหรับใช้ในอนาคต

ตารางที่ 2.7 ข้อมูลเกี่ยวกับทรานแซคชันที่ได้จาก PID

อุปกรณ์โลว์สปีดจะไม่มองแพ็กเก็ต SOF แต่ฮับที่ต่อกับอุปกรณ์จะใช้สัญญาณจบแพ็กเก็ต (End of Packet: EOP) ส่งไปเฟรมละหนึ่งครั้งแทน นอกจากนี้อุปกรณ์โลว์สปีดยังใช้ EOP เป็นตัวที่ช่วยรักษาอุปกรณ์จากการเข้าสู่สภาวะซัซเพนต์อีกด้วย

สำหรับ PID ชนิดพิเศษ 4 ตัวนั้น ตัวหนึ่งจะใช้อุปกรณ์โลว์สปีดเท่านั้น อีกตัวหนึ่งจะใช้สำหรับไฮสปีดอย่างเดียว และอีกสองตัวที่เหลือจะถูกใช้เมื่อฮับ 2.0 ของอุปกรณ์โลว์สปีด หรือฟูลสปีดต้องการติดต่อสื่อสารกับโฮสที่ไฮสปีด

PID ชนิดพิเศษสำหรับโลว์สปีดคือ PRE ซึ่งจะมีโค้ดที่เป็นตัวบอกล่วงหน้าให้แก่ฮับได้ทราบว่าแพ็กเก็ตถัดไปที่จะเข้ามาเป็นโลว์สปีดและฮับจะเปิดการสื่อสารกับอุปกรณ์โลว์สปีดที่ต่ออยู่ภายในระบบ บนระบบบัสโลว์สปีดและฟูสสปีดนั้น PRE จะเป็นส่วนที่ถูกส่งออกมาก่อนแพ็ก

เก็ตทุกชุด (โทเค็น, ดาต้า และแฮนด์เช็ก) ที่ส่งไปยังอุปกรณ์โลว์สปีด ส่วนระบบบัสไฮสปีดจะทำการเข้ารหัส PRE เข้าไปในแพ็กเก็ต SPLIT ดังนั้นมันจึงไม่สามารถส่งแยกออกมาต่างหากได้ สำหรับแพ็กเก็ตโลว์สปีดที่ส่งโดยอุปกรณ์นั้นไม่จำเป็นต้องใช้ PRE

ส่วน PID ที่มีใช้เฉพาะอุปกรณ์ไฮสปีด คือ PING โฮสจะส่ง PING ออกมาเพื่อหาว่าดีไวซ์เอนด์พอยน์แบบไฮสปีดอยู่ในภาวะที่พร้อมหรือไม่ก่อนที่จะมีการส่งดาต้าแพ็กเก็ตถัดไปซึ่งเป็นการส่งถ่ายข้อมูลแบบบัลก์หรือคอนโทรลที่มีดาต้าแพ็กเก็ตหลายชุด

สำหรับ SPLIT เป็นตัวที่ใช้บอกว่าโทเค็นแพ็กเก็ตนั้นๆ เป็นส่วนหนึ่งของทรานแซคชันที่เกิดจากการแบ่งออกเป็นส่วนๆ หรือเรียกว่าสปลิตทรานแซคชัน (Split Transaction) ทางด้านโฮสและฮับ 2.0 จะขนส่งข้อมูลที่เป็นโลว์สปีดและฟูลสปีดที่ความเร็วไฮสปีดเพื่อให้การใช้เวลาของบัสมีประสิทธิภาพมากขึ้นเมื่อโฮสเริ่มทรานแซคชันซึ่งมีเป้าหมายของการส่งไปที่อุปกรณ์โลว์สปีดและฟูลสปีด ฮับ 2.0 ที่อยู่ใกล้กับอุปกรณ์ตัวเป้าหมายมากที่สุดจะรับผิดชอบในการทำทรานแซคชันกับอุปกรณ์ให้สมบูรณ์รวมถึงการเก็บบัฟเฟอร์ข้อมูลสถานะใดๆ ที่ถูกส่งออกกลับมา และรายงานกลับไปในทรานแซคชัน 1 ครั้งหรือมากกว่านั้น ด้วยวิธีการนี้ระบบบัสทั้งระบบจะไม่ต้องรอให้ทรานแซคชันที่ความเร็วต่ำกว่าสำเร็จสมบูรณ์ ซึ่งทรานแซคชันพิเศษระหว่างฮับและโฮสนี้จะเรียกว่าสปลิตทรานแซคชัน หรือ การแบ่งทรานแซคชัน

ในส่วนของ ERR จะมีใช้เฉพาะในสปลิตทรานแซคชันเท่านั้น ฮับ 2.0 จะใช้ PID นี้เพื่อรายงานความผิดพลาดให้แก่โฮสในทรานแซคชันโลว์สปีดและฟูลสปีด ซึ่ง ERR และ PRE จะมีค่าเหมือนกันแต่จะการสับสนกันเนื่องจากฮับไม่มีการส่ง PRE ไปยังโฮส หรือส่ง ERR ไปยังอุปกรณ์

2.9.2 ลำดับของแพ็กเก็ต

ทรานแซคชันทุกๆ ครั้งจะมีโทเค็นแพ็กเก็ตบรรจุอยู่ โดยโฮสจะเป็นตัวสร้างแพ็กเก็ตชนิดนี้เสมอ ซึ่งทรานแซคชันที่สร้างขึ้นนี้จะประกอบด้วยการแสดงชนิดของแพ็กเก็ต, อุปกรณ์เอนด์พอยน์ที่เป็นตัวรับ และทิศทางของข้อมูลใดๆ ซึ่งทรานแซคชันจะทำการส่งถ่ายข้อมูล ในกรณีที่ทรานแซคชันนั้นๆ เป็นแบบโลว์สปีดที่ถูกส่งไปบนระบบบัสแบบฟูลสปีดจะต้องมีการส่งแพ็กเก็ต PRE นำหน้าโทเค็นแพ็กเก็ตออกมา และถ้าทรานแซคชันนั้นๆ เป็นสปลิตทรานแซคชัน ทรานแซคชันนั้นจะต้องมีแพ็กเก็ต SPLIT ถูกส่งนำหน้าโทเค็นแพ็กเก็ตออกมาเสมอ

แพ็กเก็ตถัดไปที่จะถูกส่งตามโทเค็นแพ็กเก็ตออกมาได้แก่ ดาต้าแพ็กเก็ต แต่แพ็กเก็ตนี้อาจมีหรือไม่มีก็ได้ขึ้นอยู่กับชนิดของการส่งถ่ายข้อมูล และในขณะนั้นอุปกรณ์มีข้อมูลที่ต้องการส่งออกไปหรือไม่ โดยทิศทางที่กำหนดอยู่ในโทเค็นแพ็กเก็ตจะเป็นตัวที่ใช้หาว่าโฮสหรืออุปกรณ์จะเป็นตัวส่งดาต้าแพ็กเก็ต

ในการส่งถ่ายข้อมูลทุกชนิดยกเว้นไอโซโครนัส อุปกรณ์ที่ได้รับดาต้าแพ็กเก็ตจะส่งแฮนด์เช็กแพ็กเก็ตกลับออกมา ซึ่งมันจะบรรจุโค้ดที่แสดงถึงความสำเร็จหรือความผิดพลาดของทรานแซคชัน

2.9.3 การจัดเวลาในทรานแซคชัน

ภายในทรานแซคชันจะยอมให้เกิดการหน่วงเวลาระหว่างโทเค็นแพ็กเก็ต, ดาต้าแพ็กเก็ต และแฮนด์เช็กแพ็กเก็ตได้เล็กน้อย ซึ่งค่าความหน่วงเวลาที่ยอมให้เกิดขึ้นนี้ถูกออกแบบไว้ในกรณีที่เกิดความหน่วงเวลาจากสายเคเบิลและค่าเวลาสวิตชิ่ง (Switching Time) รวมกับค่าเวลาอีกเล็กน้อยที่ฮาร์ดแวร์ใช้สำหรับเตรียมการตอบสนองแพ็กเก็ตที่ได้รับเข้ามา

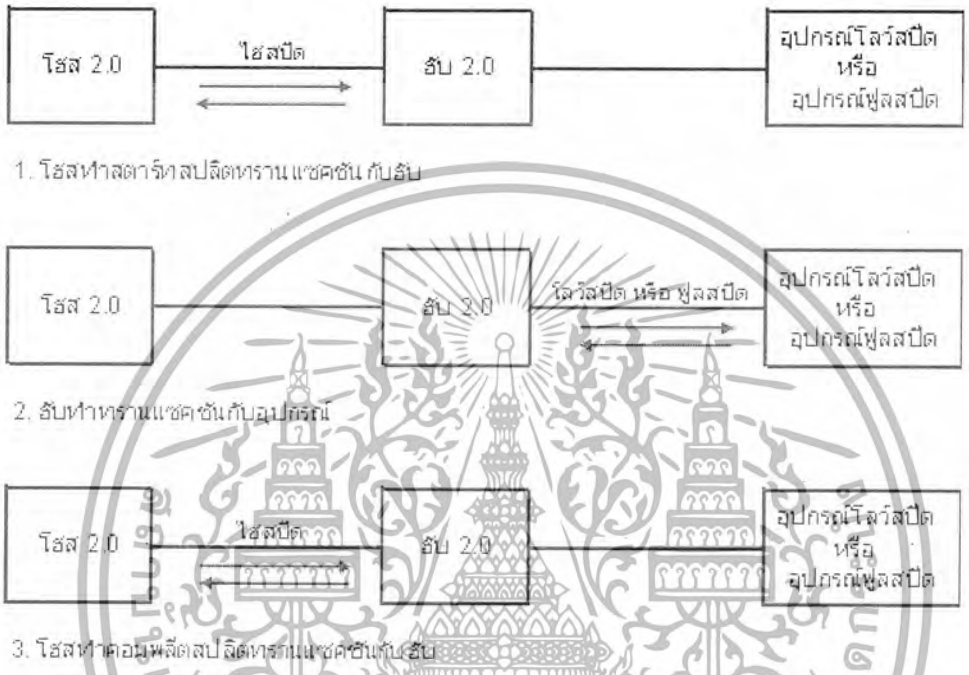
ขนาดสูงสุดของแพ็กเก็ตในการส่งถ่ายข้อมูลและเอนด์พอยน์แต่ละชนิด จะเป็นตัวจำกัดปริมาณของข้อมูลซึ่งทรานแซคชันสามารถบรรจุเข้าไปได้ การส่งถ่ายข้อมูลที่มีหลายทรานแซคชันนั้นอาจต้องใช้จำนวนเฟรมหลายเฟรม แต่ไม่จำเป็นต้องเป็นเฟรมที่ต่อเนื่องกัน เช่น ในการส่งถ่ายข้อมูลบัลก์ฟูลสปีดขนาด 512 ไบต์ ซึ่งในการส่งถ่ายข้อมูลชนิดนี้จะมีจำนวนไบต์สูงสุดสำหรับหนึ่งทรานแซคชันเท่ากับ 64 ไบต์ ดังนั้นการส่งถ่ายข้อมูลทั้งหมดจึงจำเป็นต้องใช้อย่างน้อย 8 ทรานแซคชัน

2.9.4 การแบ่งทรานแซคชัน (Split Transaction)

ฮับ 2.0 จะติดต่อสื่อสารกับโฮส 2.0 ที่ไฮสปีด ถ้าไม่มีฮับ 1.x ต่อคั่นอยู่ระหว่างมันเมื่ออุปกรณ์แบบโลว์สปีดและฟูลสปีดถูกต่อเข้ากับฮับ 2.0 ฮับจะแปลงความเร็วของการสื่อสารให้เหมาะสมกับความต้องการของอุปกรณ์ ซึ่งการแปลงความเร็วนี้เป็นเพียงหน้าที่หนึ่งที่ฮับจะต้องทำในการจัดการกับการสื่อสารที่ความเร็วหลายๆ แบบ เนื่องจากบัสไฮสปีดจะมีความเร็วมากกว่าบัสฟูลสปีดอยู่ประมาณ 40 เท่า และเร็วกว่าโลว์สปีดถึง 320 เท่า ด้วยความเร็วที่แตกต่างกันอย่างมากนี้จึงอาจทำให้เกิดปัญหาแก่ระบบขึ้น โดยทั้งระบบจะต้องรอการส่งข้อมูลในขณะที่ฮับมีการแลกเปลี่ยนข้อมูลระหว่างข้อมูลโลว์สปีดหรือฟูลสปีดกับอุปกรณ์รอบข้าง

ปัญหาดังกล่าวสามารถแก้ไขได้ด้วยการแบ่งทรานแซคชัน หรือสปลิตทรานแซคชัน (Split Transaction) (รูปที่ 25) โฮส 2.0 จะใช้สปลิตทรานแซคชันเมื่อมันทำการติดต่อสื่อสารกับอุปกรณ์โลว์สปีดหรือฟูลสปีดบนบัสแบบไฮสปีด ตามปกติในทรานแซคชันหนึ่งๆ ที่สื่อสารระหว่างบัสไฮสปีดกับบัสโลว์สปีดหรือฟูลสปีดจะต้องสปลิตทรานแซคชัน 2 ชนิด นั่นคือจะมี 1 ทรานแซคชันหรือมากกว่านั้นที่เป็นสตาร์ทสปลิตทรานแซคชัน (Start Split Transaction) เพื่อใช้ในการส่งข้อมูลไปยังอุปกรณ์ และจะมีอีก 1 ทรานแซคชันหรือมากกว่านั้นที่เป็นคอมพลีตสปลิตทรานแซคชัน

(Complete Split Transaction) เพื่อใช้ในการรองรับข้อมูลจากอุปกรณ์ ยกเว้นทรานแซคชันแบบไอโซโครนัส OUT ซึ่งไม่จำเป็นต้องใช้คอมพลีตสปลิตทรานแซคชันเนื่องจากในการส่งถ่ายข้อมูลแบบนี้จะไม่มีการส่งค่ากลับมา



รูปที่ 2.13 สปลิตทรานแซคชัน

ถึงแม้ว่าจะมีการทำสปลิตทรานแซคชันจะทำให้เกิดทรานแซคชันขึ้นหลายชุดในระบบบัสก็ตาม แต่สปลิตทรานแซคชันก็ทำให้การใช้งานบนบัสดีขึ้น เนื่องจากมันช่วยลดปริมาณการใช้เวลาของบัสในการรอให้อุปกรณ์แบบโลว์สปีดหรือฟูลสปีดตอบสนองกลับมา ตารางที่ 9 เป็นการเปรียบเทียบโครงสร้างและเนื้อหาภายในของทรานแซคชันกับอุปกรณ์แบบโลว์สปีดและฟูลสปีดที่ความเร็วบัสที่แตกต่างกัน

ความเร็วบัส	ชนิดของทรานแซกชัน	ทรานแซกชันเฟส		
		โทเค็น	ดาต้า	แฮนด์เช็ก
การสื่อสารโลว์/ฟูลสปีดกับอุปกรณ์	SETUP, OUT	PRE ถ้าเป็นโลว์สปีด, โทเค็น LS/FS	PRE ถ้าเป็นโลว์สปีด, ดาต้า	สเตตัส (ยกเว้นสำหรับไอโซโครนัส)
	IN	PRE ถ้าเป็นโลว์สปีด, โทเค็น LS/FS	ดาต้าหรือสเตตัส	PRE ถ้าเป็นโลว์สปีด, สเตตัส (ยกเว้นสำหรับไอโซโครนัส)
การสื่อสารไฮสปีดระหว่างฮับ 2.0 และโฮสในการทรานแซคชันกับอุปกรณ์โลว์สปีดหรือฟูลสปีด	SETUP, OUT (ไอโซโครนัส OUT จะไม่มีทรานแซคชัน CSPLIT)	SSPLIT, โทเค็น LS/FS	ดาต้า	สเตตัส
		CSPLIT, โทเค็น LS/FS	-	สเตตัส (เฉพาะบัลก์และคอนโทรล)
	IN	SSPLIT, โทเค็น LS/FS	-	-
		CSPLIT, โทเค็น LS/FS	ดาต้าหรือสเตตัส	-

ตาราง 2.8 ทรานแซคชันไฮสเมื่อมีการสื่อสารบนบัสที่มีความเร็วแตกต่างกัน

สำหรับการทำงานของสปลิตทรานแซคชันในการส่งถ่ายข้อมูลแบบบัลก์และคอนโทรลซึ่งไม่มีการบังคับทางด้านเวลาอย่างเข้มงวดเหมือนกับการส่งถ่ายข้อมูลแบบอินเทอร์รัพท์และไอโซโครนัสสามารถอธิบายได้ดังนี้

ในส่วนของสตาร์ทสปลิตทรานแซคชันจะเริ่มจากโฮส 2.0 ทำการส่งสตาร์ทสปลิตโทเค็นแพ็กเก็ต (Start Split Token Packet: SSPLIT) ออกมาแล้วตามด้วยโทเค็นแพ็กเก็ตตามปกติของโลว์สปีดหรือฟูลสปีด และดาต้าแพ็กเก็ตใดๆ ที่มีจุดหมายไปที่อุปกรณ์ ฮับ 2.0 ที่อุปกรณ์ต่ออยู่จะส่งค่ากลับมาเป็น ACK หรืออาจจะเป็น NAK ในกรณีที่อุปกรณ์อยู่ในภาวะไม่ว่างหรือไม่มีข้อมูลที่ต้องการส่ง จากนั้นโฮสจะอยู่ในสภาวะว่างเพื่อให้สามารถใช้บัสกับทรานแซคชันอื่นๆ ได้ แต่ในขณะนี้อุปกรณ์จะยังไม่รับรู้อะไรเกี่ยวกับทรานแซคชันเลย

เมื่อมีการส่ง ACK เพื่อเป็นการตอบสตาร์ทสปลิตทรานแซคชันกลับมา ฮับจะทำหน้าที่ 2 อย่างด้วยกัน อย่างแรกคือมันต้องทำทรานแซคชันกับอุปกรณ์ให้เสร็จสมบูรณ์ และอย่างที่สองคือมันยังต้องจัดการกับการขนส่งอื่นๆ ที่อยู่บนบัสที่รับมาจากโฮสหรืออุปกรณ์ตัวอื่นๆ ที่ต่ออยู่ในระบบ

ในการทำทรานแซคชันให้เสร็จสมบูรณ์ ฮับจะแปลงแพ็กเก็ตที่รับเข้ามาจากโฮสให้อยู่ในความเร็วที่เหมาะสมแล้วส่งไปยังอุปกรณ์ และบางครั้งอาจต้องเก็บการตอบสนองของอุปกรณ์เอาไว้ด้วย อุปกรณ์อาจส่งข้อมูลหรือแฮนด์เช็กกลับมา หรือไม่ส่งอะไรออกมาก็ได้ทั้งนี้ขึ้นอยู่กับชนิดของทรานแซคชัน ทรานแซคชันที่มายังอุปกรณ์จะถูกส่งต่อไปที่โลว์สปีดหรือฟูลสปีดตามแต่ชนิดของอุปกรณ์ซึ่งในขณะนี้ถือว่าทรานแซคชันเสร็จสมบูรณ์แล้ว แต่อุปกรณ์จะไม่ทราบว่าสิ่งที่รับเข้ามาเป็นสปลิตทรานแซคชัน และในขณะนี้โฮสจะยังไม่ได้รับการตอบสนองจากอุปกรณ์

ในขณะที่ฮับกำลังทำทรานแซคชันกับอุปกรณ์ให้เสร็จสมบูรณ์ โฮสอาจต้องเริ่มทำการขนส่งข้อมูลอื่นๆ บนบัส ซึ่งฮับจะต้องเป็นตัวที่ช่วยจัดการด้วยเช่นกัน หน้าที่การทำงานทั้งสองอย่างของฮับจะถูกจัดการโดยโมดูลของฮาร์ดแวร์ที่แยกจากกันภายในตัวฮับ

สำหรับทรานแซคชันทั้งหมดยกเว้นทรานแซคชันแบบไอโซโครนัส OUT เมื่อโฮสเห็นว่าฮับมีเวลาที่เพียงพอในการทำทรานแซคชันกับอุปกรณ์ให้สมบูรณ์ มันก็จะเริ่มทำคอมพลีตสปลิตทรานแซคชันกับฮับ

ในคอมพลีตสปลิตทรานแซคชันนั้น โฮสจะส่งคอมพลีตสปลิตโทเคินแพ็กเก็ต (Complete Split Token Packet ; CSPLIT) ออกมาก่อนตามด้วยโทเคินแพ็กเก็ตของโลว์สปีดหรือฟูลสปีดตามปกติเพื่อร้องขอข้อมูลหรือข้อมูลสถานะ ซึ่งฮับจะได้รับจากอุปกรณ์อีกต่อหนึ่ง ฮับจะส่งข้อมูลหรือโค้ดสถานะที่โฮสร้องขอมากลับออกไป ซึ่งขณะนี้ถือว่าทรานแซคชันเสร็จสมบูรณ์แล้ว และทางโฮสไม่จำเป็นต้องส่ง ACK กลับออกมา แต่ถ้าฮับไม่มีแพ็กเก็ตซึ่งพร้อมจะส่งออกมาฮับจะส่งโค้ดสถานะ NYET กลับออกมาและโฮสจะพยายามใหม่อีกครั้งในภายหลัง ทางด้านของอุปกรณ์จะไม่ทราบได้เลยว่าการสื่อสารนี้เป็นคอมพลีตสปลิตทรานแซคชัน เนื่องจากอุปกรณ์ได้ทำทรานแซคชันกับฮับเสร็จสมบูรณ์ไปก่อนหน้านี้แล้ว

สตาร์ทสปลิตทรานแซคชันในการส่งถ่ายข้อมูลแบบอินเทอร์รัพท์และไอโซโครนัสจะไม่มีเฟสของแฮนด์เช็ก ไม่เหมือนกับที่ใช้ในการส่งข้อมูลแบบบัลก์และคอนโทรล ซึ่งในกรณีนี้จะมีเพียงสตาร์ทสปลิตโทเค็นแล้วตามมาด้วยโทเค็น IN, OUT หรือ SETUP และข้อมูลในกรณีที่มันเป็นทรานแซคชัน OUT หรือเซ็ตอัพทรานแซคชัน

ในส่วนของอินเทอร์รัพท์ทรานแซคชันนั้น ฮับจะทำการจัดตารางเวลาของสตาร์ทสปลิตในไมโครเฟรมไว้ก่อนเวลาซึ่งฮับคาดว่าจะเริ่มทรานแซคชันกับอุปกรณ์ ตัวอย่างเช่น สมมติว่าไมโคร

เฟรมที่อยู่ในเฟรมถูกกำหนดชื่อให้เป็น Y0 ถึง Y7 ถ้าสตาร์ทสปลิตอยู่ใน Y0 ทรานแซคชันกับอุปกรณ์อาจเกิดขึ้นก่อน Y1 อุปกรณ์อาจมีข้อมูลหรือแฮนด์เช็กที่ใช้ในการตอบสนองกลับไปยังโฮสก่อน Y2 ผลของทรานแซคชันที่ผ่านมาและการทำบิตสตัฟ (Bit Stuff) จะสามารถมีผลต่อมาได้เมื่อทรานแซคชันกับอุปกรณ์เกิดขึ้นจริง ดังนั้นโฮสจะจัดเวลาคอมพลีตสปลิตทรานแซคชันใน Y2, Y3 และ Y4 ถ้าฮับยังไม่มีข้อมูลที่จะส่งกลับในคอมพลีตสปลิตมันก็จะส่ง NYET กลับไป และโฮสจะพยายามใหม่อีกครั้ง

ไอโซโครนัสทรานแซคชันแบบฟูลสปีดจะสามารถส่งถ่ายข้อมูลได้ถึง 1023 ไบต์ การทำทรานแซคชันด้วยแพ็กเก็ตขนาดใหญ่จะใช้สตาร์ทสปลิต หรือ คอมพลีตสปลิตหลายตัว โดยในแต่ละตัวจะมีถึง 188 ไบต์ เพื่อให้มั่นใจได้ว่าการส่งถ่ายข้อมูลสามารถทำได้ทันเวลาหรือทันทีทันใดที่อุปกรณ์มีการส่งหรือพร้อมที่จะรับข้อมูล ซึ่งค่านี้คือปริมาณสูงสุดของข้อมูลฟูลสปีดซึ่งสามารถส่งถ่ายได้ในไมโครเฟรม ข้อมูลของทรานแซคชันหนึ่งๆ อาจต้องการสตาร์ทสปลิต หรือ คอมพลีตสปลิตทรานแซคชันได้มากถึง 8 ทรานแซคชัน

ในไอโซโครนัสทรานแซคชัน IN โฮสจะจัดเวลาสำหรับคอมพลีตสปลิตทรานแซคชันในทุกๆไมโครเฟรมซึ่งมันคาดหมายว่าอุปกรณ์จะมีส่วนของข้อมูลอย่างน้อยส่วนหนึ่งกลับออกมา การร้องขอข้อมูลในส่วนที่เล็กกว่าจะทำให้มั่นใจได้ว่าโฮสรับข้อมูลได้อย่างเร็วที่สุดเท่าที่จะเป็นไปได้ โดยโฮสไม่ต้องรอข้อมูลทั้งหมดที่จะส่งถ่ายจากอุปกรณ์ที่ฟูลสปีดก่อนจะมีการเริ่มรับมันเข้ามา

ในไอโซโครนัสทรานแซคชัน OUT โฮสจะส่งข้อมูลไปในสตาร์ทสปลิตทรานแซคชัน 1 ครั้งหรือมากกว่านั้น โดยโฮสจะจัดเวลาของทรานแซคชันเพื่อไม่ให้บัฟเฟอร์ของฮับว่างเปล่าลงแต่จะบรรจุด้วยไบต์เล็กเพียงเท่าที่จะเป็นไปได้ลงไปแทน ในแต่ละแพ็กเก็ต SPLIT จะมีบิตซึ่งใช้บอกตำแหน่งของข้อมูลในดาต้าแพ็กเก็ตของโลว์สปีดหรือฟูลสปีด (เริ่มต้น, กึ่งกลาง, ตอนท้าย หรือทั้งหมด) แต่ในที่นี้จะไม่มีคอมพลีตสปลิตทรานแซคชัน

2.10 ชนิดของการส่งถ่ายข้อมูล

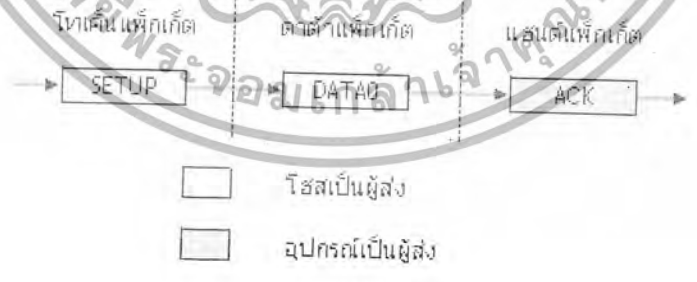
ข้อกำหนด USB มีการกำหนดรูปแบบการส่งถ่ายข้อมูลไว้ 4 ชนิด ได้แก่ การส่งถ่ายข้อมูลแบบคอนโทรล, บัลก์, อินเทอร์รัพท์ และไอโซโครนัส ซึ่งแต่ละชนิดจะมีจุดประสงค์การใช้งานที่แตกต่างกัน

2.10.1 การส่งถ่ายข้อมูลคอนโทรล (Control Transfer)

การส่งถ่ายข้อมูลชนิดนี้เป็นการสื่อสารข้อมูลแบบสองทิศทาง ซึ่งใช้สำหรับการตั้งค่าการทำงาน, การส่งคำสั่งหรือข้อมูลสถานะ การส่งถ่ายข้อมูลชนิดนี้จะเริ่มต้นขึ้นจากโฮส และใช้ความ

พยายามในการส่งมากที่สุด นอกจากนี้ยังมีการตรวจสอบความผิดพลาดด้วยการใช้ CRC (Cyclic Redundancy Check) เพื่อไม่ให้ข้อมูลเกิดการผิดพลาดได้เนื่องจากข้อมูลที่ใช้ในการส่งถ่ายข้อมูลชนิดนี้มีความสำคัญมากที่สุด ขนาดของข้อมูลซึ่งถูกใส่ลงไปในฟิลด์ข้อมูลของดาต้าแพ็กเก็ตหรือเรียกทับศัพท์ว่าดาต้าเพย์โหลด (Data Payload) สำหรับการส่งถ่ายข้อมูลคอนโทรลในอุปกรณ์โลว์สปีดจะมีขนาดเท่ากับ 8 ไบต์. อุปกรณ์ฟูลสปีดมีขนาดเท่ากับ 64 ไบต์ และอุปกรณ์ไฮสปีดสามารถมีขนาดของดาต้าเพย์โหลดเท่ากับ 8. 16. 32 หรือ 64 ไบต์ ส่วนขั้นตอนการทำงานหรือสเตจ (Stage) ของมันอาจประกอบไปด้วย 2 ถึง 3 สเตจ ได้แก่ เซ็ตอัพสเตจ. ดาต้าสเตจ (อาจจะมีหรือไม่มีก็ได้) และสเตตัสสเตจ

- เซ็ตอัพสเตจ (Setup Stage) เป็นขั้นตอนที่คำร้องขอถูกส่งออกไป ประกอบด้วยแพ็กเก็ต 3 ชุด โดยโทเค็นแพ็กเก็ตจะถูกส่งออกไปเป็นลำดับแรก ซึ่งภายในประกอบด้วย PID ที่มีค่าเป็น SETUP จากนั้นดาต้าแพ็กเก็ตจะถูกส่งออกมาเป็นลำดับถัดไป ซึ่ง PID ของแพ็กเก็ตนี้จะเป็น DATA0 เสมอ ภายในแพ็กเก็ตนี้ประกอบไปด้วยเซ็ตอัพแพ็กเก็ตซึ่งมีข้อมูลเกี่ยวกับคำร้องขอ รวมถึงหมายเลขของคำร้องขอด้วย ส่วนแพ็คเก็ตสุดท้ายคือแฮนด์เช็กแพ็กเก็ตซึ่งใช้สำหรับการบอกสถานะของการรับส่งข้อมูลว่าสำเร็จหรือไม่ ถ้าอุปกรณ์รับข้อมูลสำเร็จ (CRC และ PID ถูกต้อง) มันจะตอบกลับมาด้วย ACK เพียงอย่างเดียว มิฉะนั้นมันจะละทิ้งข้อมูลนั้นไปและไม่ส่งแฮนด์เช็กแพ็กเก็ตกลับไปยังโฮส ทั้งนี้อุปกรณ์จะไม่มีการส่ง STALL หรือ NAK ในการตอบสนองต่อเซ็ตอัพแพ็กเก็ต

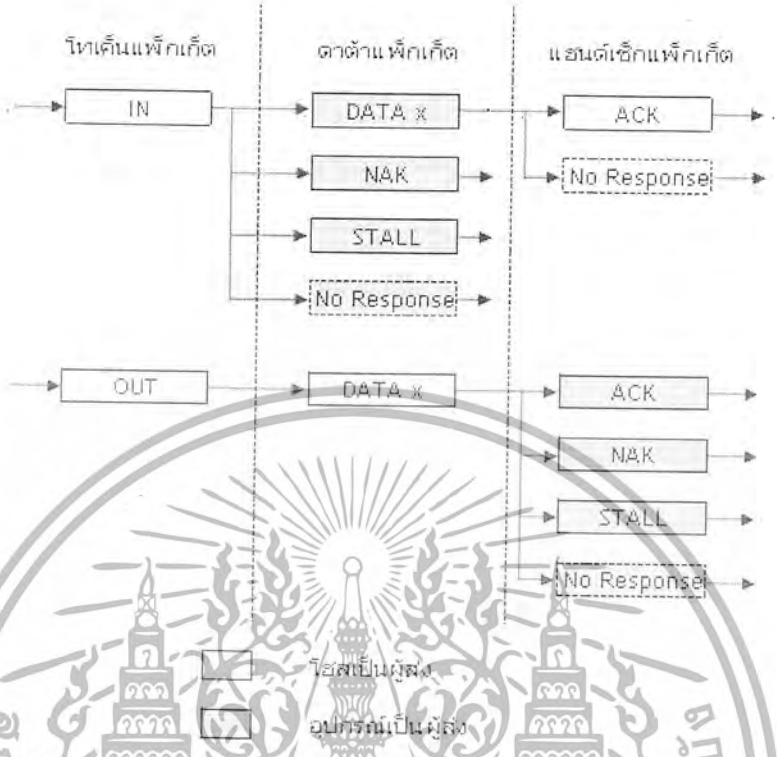


รูปที่ 2.14 เซ็ตอัพสเตจของการส่งถ่ายข้อมูลคอนโทรล

- ดาต้าสเตจ (Data Stage) สเตจนี้อาจมีหรือไม่มีก็ได้ โดยมันจะประกอบไปด้วยการส่งถ่ายข้อมูลแบบ IN หรือ OUT อย่างใดอย่างหนึ่งเป็นจำนวน 1 ครั้งหรือมากกว่านั้น ซึ่งจำนวนของข้อมูลที่ถูกส่งไปในสเตจนี้จะถูกกำหนดไว้ในเซ็ตอัพแพ็กเก็ต ถ้าหากข้อมูลมีขนาด

เกินกว่าขนาดสูงสุดของดาต้าเพย์โหลด ข้อมูลนั้นจะถูกส่งด้วยการส่งถ่ายข้อมูลหลายครั้ง โดยในแต่ละครั้งจะมีขนาดเท่ากับขนาดสูงสุดของดาต้าเพย์โหลดยกเว้นการส่งในครั้งสุดท้าย ดาต้าสเตจจะมีบทบาทที่แตกต่างกัน 2 อย่างโดยขึ้นอยู่กับทิศทางของการส่งถ่ายข้อมูล ได้แก่

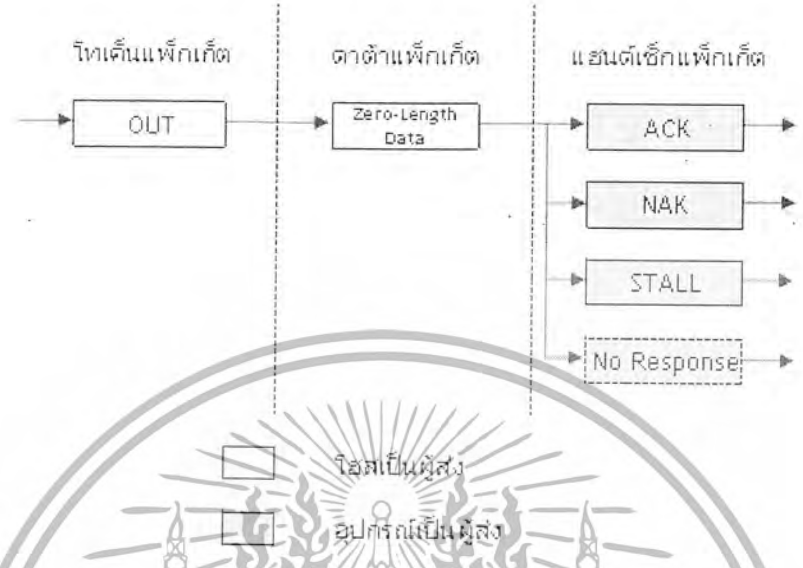
- IN : เกิดขึ้นในการส่งถ่ายข้อมูลแบบคอนโทรล Read โดยเมื่อโฮสพร้อมรับข้อมูลคอนโทรล โฮสจะส่งโทเค็น IN ออกมา ถ้าอุปกรณ์ได้นับโทเค็น IN ที่มีความผิดพลาด เช่น PID กับ ส่วนอินเวอร์สของมันไม่สอดคล้องกัน อุปกรณ์จะละทิ้งแพ็กเก็ตนี้ไป แต่ถ้าโทเค็นที่ได้รับมาถูกต้องอุปกรณ์อาจตอบกลับมาด้วยดาต้าแพ็กเก็ตซึ่งบรรจุข้อมูลคอนโทรลที่จะส่ง หรืออาจเป็นแพ็กเก็ต STALL เพื่อเป็นการบอกว่าเกิดการผิดพลาดขึ้นที่เอนด์พอยน์ หรืออาจเป็นแพ็กเก็ต NAK เพื่อเป็นการบอกโฮสว่าเอนด์พอยน์สามารถทำงานได้ตามปกติแต่ในขณะนั้นไม่มีข้อมูลที่จะทำการส่ง
- OUT : เกิดขึ้นในการส่งถ่ายข้อมูลแบบคอนโทรล Write โดยเมื่อโฮสต้องการส่งคอนโทรลแพ็กเก็ตไปยังอุปกรณ์ มันจะส่งโทเค็น OUT ออกมาเก้วตามด้วยดาต้าแพ็กเก็ตซึ่งบรรจุดาต้าเพย์โหลดที่เป็นข้อมูลคอนโทรลเอาไว้ ถ้าบางส่วนของโทเค็น OUT หรือดาต้าแพ็กเก็ตไม่ถูกต้อง อุปกรณ์จะละทิ้งแพ็กเก็ตนี้ไป ถ้าเอนด์พอยน์บัฟเฟอร์ของอุปกรณ์ว่างและสามารถป้อนข้อมูลเข้าไปในเอนด์พอยน์บัฟเฟอร์ได้ อุปกรณ์จะส่ง ACK ออกมาเพื่อแจ้งให้โฮสทราบว่ามันรับข้อมูลสำเร็จแล้ว ถ้าเอนด์พอยน์บัฟเฟอร์ไม่ว่างเนื่องจากการประมวลผลแพ็กเก็ตก่อนหน้านี้ยังไม่เสร็จสมบูรณ์ อุปกรณ์จะส่งส่ง NAK กลับไป แต่อย่างไรก็ตามมันอาจส่งค่า STALL กลับออกมาถ้าเอนด์พอยน์มีข้อผิดพลาดและบิต HALT ถูกเซ็ตอยู่



รูปที่ 2.15 ดาต้าสเตจของการส่งถ่ายข้อมูลคอนโทรล

- สเตตัสสเตจ (Status Stage) เป็นสเตจที่ใช้รายงานสถานะของการร้องขอทั้งหมด และมีทิศทางแปรเปลี่ยนไปตามการเปลี่ยนแปลงทิศทางของการส่งถ่ายข้อมูล ซึ่งการรายงานสถานะจะกระทำจากอุปกรณ์เสมอ

  - ถ้าการส่งถ่ายข้อมูลคอนโทรลครั้งนั้นๆ ไม่มีดาต้าสเตจ อุปกรณ์จะตอบกลับไปด้วย ACK แต่ถ้าข้อมูลมีการผิดพลาด มันจะละทิ้งข้อมูลนั้นไปและไม่ส่งแฮนด์เช็กแพ็กเก็ตกลับไปยังโฮส
  - กรณีที่เป็นคอนโทรล Read โฮสจะต้องตอบรับ (Acknowledge) ว่าการรับข้อมูลนี้ทำได้สำเร็จ โดยโฮสต้องทำการส่งโทเค็น OUT แล้วตามด้วยดาต้าแพ็กเก็ตที่มีความยาวเป็นศูนย์ในขณะนี้อุปกรณ์จะสามารถรายงานสถานะของมันลงมาในแฮนด์เช็กสเตจด้วย ACK ซึ่งเป็นการบอกว่าอุปกรณ์ได้ทำตามคำสั่งโดยสมบูรณ์แล้ว และพร้อมที่จะรับคำสั่งถัดไป ถ้าเกิดข้อผิดพลาดขึ้นในขณะประมวลคำสั่งนี้อุปกรณ์จะส่ง STALL ออกมา อย่างไรก็ตามถ้าอุปกรณ์ยังคงประมวลผลต่อไปมันก็จะส่ง NAK กลับออกมาเพื่อแจ้งให้โฮสทำสเตตัสสเตจอีกครั้งในภายหลัง



รูปที่ 2.16 สเตตัสสเตจของการส่งถ่ายข้อมูลคอนโทรล Read

- กรณีนี้เป็นคอนโทรล Write อุปกรณ์จะตอบรับว่าการรับข้อมูลทำได้สำเร็จโดยการส่งแพ็กเก็ตที่มีความยาวเป็นศูนย์ในการตอบสนองโทเค็น IN อย่างไรก็ตามถ้ามีข้อผิดพลาดเกิดขึ้น มันจะส่ง STALL ออกมา หรือถ้าอุปกรณ์ยังคงอยู่ในระหว่างการประมวลผลข้อมูลอยู่ มันจะส่ง NAK ออกไปเพื่อบอกโฮสให้ข้ามสเตตัสสเตจอีกครั้งในภายหลัง

รูปที่ 2.17 สเตตัสสเตจของการส่งถ่ายข้อมูลคอนโทรล Write

2.10.2 การส่งถ่ายข้อมูลอินเทอร์รัพท์ (Interrupt Transfer)

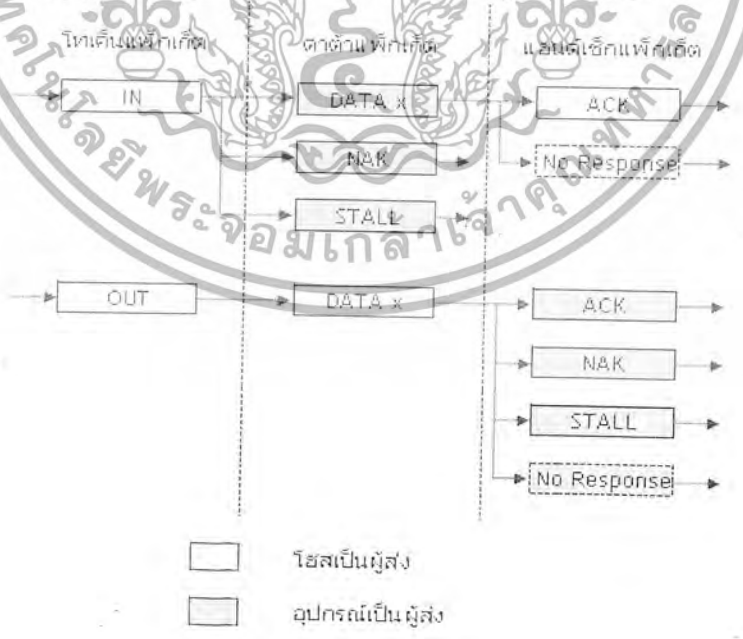
การส่งถ่ายข้อมูลอินเทอร์รัพท์ในระบบบัส USB มีความหมายไม่เหมือนกับอินเทอร์รัพท์ที่ใช้กันโดยทั่วไปในระบบไมโครคอนโทลเลอร์ซึ่งเป็นสิ่งที่อุปกรณ์รอบข้างสร้างขึ้น แต่ในระบบบัส USB จะใช้สำหรับส่งสัญญาณสอบถามหรือโพล์ต่างๆ เพื่อทำการหาว่าอุปกรณ์ตัวไหนต้องการรับส่งข้อมูลกับโฮส ซึ่งถ้าอุปกรณ์ต้องการติดต่อกับโฮส มันจะต้องรอจนกระทั่งโฮสส่งสัญญาณโพล์ไปที่มันก่อน จากนั้นจึงบอกความต้องการในการรับส่งข้อมูลกับโฮส

คุณสมบัติของการส่งถ่ายข้อมูลอินเทอร์รัพท์

- มีการรับประกันความถูกต้องของข้อมูล
- ใช้สตรีมไปป์ (ทิศทางเดียว)
- มีการตรวจสอบความผิดพลาดของข้อมูล

ตามปกติการส่งถ่ายข้อมูลอินเทอร์รัพท์จะไม่เป็นช่วงนาน คำร้องขออินเทอร์รัพท์จะถูกจัดอยู่ในอุปกรณ์จนกระทั่งโฮสทำการโพล์ไปยังอุปกรณ์ USB ตัวนั้นเพื่อสอบถามข้อมูลสำหรับขนาดสูงสุดของดาต้าเพย์โหลดของโหมดการสื่อสารต่างๆมีดังนี้

- ขนาดสูงสุดของข้อมูลภายในดาต้าเพย์โหลดสำหรับอุปกรณ์โลว์สปีดมีค่าเท่ากับ 8 ไบต์
- ขนาดสูงสุดของข้อมูลภายในดาต้าเพย์โหลดสำหรับอุปกรณ์โลว์สปีดมีค่าเท่ากับ 64 ไบต์
- ขนาดสูงสุดของข้อมูลภายในดาต้าเพย์โหลดสำหรับอุปกรณ์โลว์สปีดมีค่าเท่ากับ1024 ไบต์



รูปที่ 2.18 เฟสของการส่งถ่ายข้อมูลอินเทอร์รัพท์

ทรานแซคชันของอินเทอร์รัพท์อาจประกอบไปด้วยการส่งถ่ายข้อมูล IN และ OUT ดังนี้

- IN : โฮสส่งโฟล์ไปที่อินเทอร์รัพท์อย่างสม่ำเสมอซึ่งอัตราการโพล์นี้จะระบุไว้ในเอนด์พอยน์ดีสคริปเตอร์ การโพล์แต่ละครั้งจะทำให้โฮสส่งโทเค็น IN ออกไป ถ้าโทเค็น IN เกิดความผิดพลาด อุปกรณ์จะละทิ้งแพ็กเก็ตนั้นและคอยตรวจสอบบัสเพื่อหาโทเค็นตัวใหม่ ถ้าคำร้องขออินเทอร์รัพท์ถูกจัดเตรียมไว้แล้วในตัวอุปกรณ์ เมื่อมันได้รับโทเค็น IN มันจะส่งดาต้าแพ็กเก็ตซึ่งบรรจุข้อมูลที่เกี่ยวข้องกับการอินเทอร์รัพท์ออกไป หลังจากโฮสได้รับข้อมูลสำเร็จแล้วมันจะส่ง ACK กลับมา แต่ถ้าข้อมูลเกิดความผิดพลาดมันจะไม่ส่งสถานะใดๆ กลับออกมา อีกกรณีหนึ่งคือถ้าอุปกรณ์ไม่ต้องการอินเทอร์รัพท์ เมื่อโฮสโพล์ไปที่อินเทอร์รัพท์เอนด์พอยน์ด้วยโทเค็น IN อุปกรณ์จะส่งสัญญาณ NAK ออกมา แต่ถ้าเกิดข้อผิดพลาดขึ้นที่เอนด์พอยน์ อุปกรณ์จะส่ง STALL ตอบโทเค็น IN กลับมา
- OUT : เมื่อโฮสต้องการส่งข้อมูลสำหรับอินเทอร์รัพท์อุปกรณ์ มันจะส่งโทเค็น OUT ออกมาแล้วตามด้วยดาต้าแพ็กเก็ตซึ่งบรรจุข้อมูลอินเทอร์รัพท์ไว้ ถ้าบางส่วนของโทเค็น OUT หรือดาต้าแพ็กเก็ตเกิดความผิดพลาดขึ้น อุปกรณ์จะละทิ้งแพ็กเก็ตนี้ไป ถ้าเอนด์พอยน์บัฟเฟอร์ของอุปกรณ์มีพื้นที่ว่างมันจะมีการป้อนข้อมูลเข้าไปในเอนด์พอยน์บัฟเฟอร์ อุปกรณ์จะส่ง ACK ออกไปเพื่อแจ้งให้โฮสทราบว่าได้ทำการรับข้อมูลสำเร็จแล้ว ถ้าเอนด์พอยน์บัฟเฟอร์ไม่มีพื้นที่ว่างเนื่องจากการประมวลผลแพ็กเก็ตก่อนหน้านี้ อุปกรณ์จะส่ง NAK กลับออกมา แต่ถ้าเกิดความผิดพลาดขึ้นที่เอนด์พอยน์และบิต HALT ถูกเซ็ตให้เป็น 1 อุปกรณ์ก็จะส่ง STALL กลับออกมา

2.10.3 การส่งถ่ายข้อมูลไอโซโครนัส (Isochronous Transfer)

การส่งถ่ายข้อมูลแบบไอโซโครนัสเป็นการส่งถ่ายข้อมูลที่เกิดขึ้นอย่างต่อเนื่องและเป็นช่วงเวลาแน่นอน ตามปกติจะใช้กับข้อมูลที่มีการเปลี่ยนแปลงอย่างต่อเนื่องตามเวลา เช่น สตรีมของข้อมูลเสียงหรือภาพ สำหรับคุณสมบัติของการส่งถ่ายข้อมูลแบบไอโซโครนัสมีดังนี้

- มีการรับประกันการเข้าถึงแบนด์วิดธ์ของบัส
- ใช้สตรีมไปป์ (ทิศทางเดียว)
- มีการตรวจจับข้อผิดพลาดโดยใช้ CRC แต่ไม่มีการส่งซ้ำหรือรับประกันการส่ง
- ใช้กับโหมดฟูลสปีดและไฮสปีดเท่านั้น
- ไม่มีการทำดาต้าท็อกเกิล

ขนาดสูงสุดของข้อมูลในดาต้าดาต้าเพย์โหลดจะถูกระบุเอาไว้ในเอนด์พอยน์ดีสคริปเตอร์ของไอโซโครนัสเอนด์พอยน์ ซึ่งสามารถมีค่าได้สูงสุดถึง 1023 ไบต์ในกรณีของอุปกรณ์ฟูลสปีด

และ 1024 ไบต์ในกรณีของอุปกรณ์ไฮสปีด เนื่องจากขนาดสูงสุดของดาต้าเพย์โหลดอาจส่งผลกระทบต่อความต้องการทางแบนด์วิดธ์ของระบบบัสเป็นอย่างมาก ดังนั้นการจำกัดขนาดของดาต้าเพย์โหลดจึงเป็นสิ่งที่สมควรกระทำ นั่นคือถ้าหากมีความต้องการใช้ดาต้าเพย์โหลดขนาดใหญ่ในการสื่อสารข้อมูล เราควรสร้างทางเลือกของอินเทอร์เฟสที่มีขนาดของดาต้าเพย์โหลดที่มีขนาดแตกต่างกันไป ซึ่งเป็นผลดีถ้าในขณะที่ทำการอินิวเมอร์เรทนั้น โฮสไม่สามารถจัดหาแบนด์วิดธ์ให้แก่ไอโซโครนัสที่ใช้ดาต้าเพย์โหลดขนาดใหญ่ได้ โฮสจะมีทางเลือกในการใช้อินเทอร์เฟสทางอื่นซึ่งใช้แบนด์วิดธ์ของไอโซโครนัสที่น้อยกว่าได้

รูปแบบของทรานแซคชันแบบไอโซโครนัส IN และ OUT แสดงดังรูปที่ 31 ซึ่งทรานแซคชันแบบนี้ไม่มีสเตจสำหรับการทำแฮนด์เช็กและไม่สามารถรายงานความผิดพลาดได้ (STALL และ HALT)

รูปที่ 2.19 เซ็ตอัพสเตจของการส่งถ่ายข้อมูลไอโซโครนัส

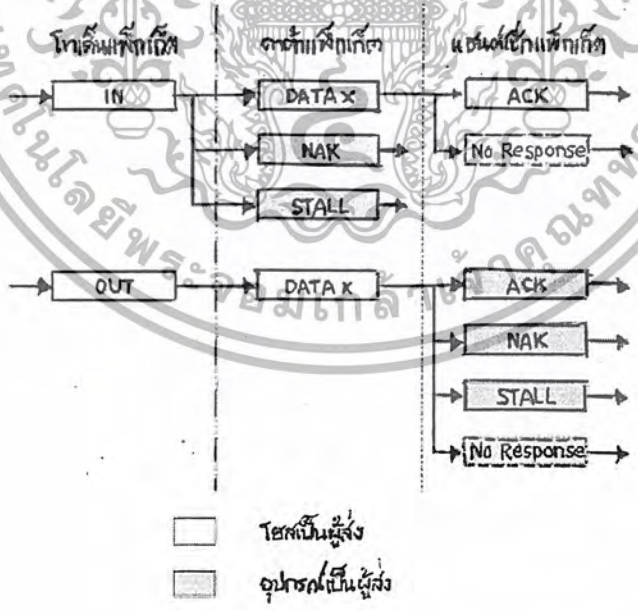
2.10.4 การส่งถ่ายข้อมูลแบบบัลก์ (Bulk Transfer)

การส่งถ่ายข้อมูลบัลก์ถูกออกแบบมาให้ใช้สำหรับการส่งข้อมูลที่มีปริมาณมากๆ ซึ่งต้องมีการรับประกันความถูกต้องของข้อมูล แต่เวลาที่ใช้สำหรับการส่งนั้นไม่ใช่สิ่งสำคัญ ตัวอย่างข้อมูลที่ใช้การรับส่งแบบบัลก์ได้แก่ ข้อมูลจากพรินเตอร์ หรือข้อมูลเครื่องสแกนเนอร์ การส่งถ่ายข้อมูลแบบบัลก์จะมีการแก้ไขความผิดพลาดด้วยการใช้ CRC16 และด้วยกลไกการตรวจจับความผิดพลาดรวมถึงการส่งข้อมูลซ้ำ ช่วยทำให้มั่นใจได้ว่าข้อมูลที่รับและส่งนั้นจะไม่มีความผิดพลาดเกิดขึ้นเลย

การส่งถ่ายข้อมูลแบบบัลก์จะใช้แบนด์วิดธ์ของบัสที่เหลือภายหลังจากที่ทรานแซคชันอื่นได้ทำการจองเรียบร้อยแล้ว ถ้าระบบบัสอยู่ในสภาวะที่ไม่ว่างเนื่องจากมีการส่งถ่ายข้อมูลแบบไอโซโครนัสหรืออินเทอร์รัพท์อยู่อาจส่งผลให้ข้อมูลบัลก์มีความล่าช้าลงไปบ้าง ดังนั้นการส่งถ่ายข้อมูลแบบนี้จึงเหมาะกับข้อมูลที่ไม่ขึ้นเปลี่ยนแปลงตามเวลา สำหรับคุณสมบัติของการส่งถ่ายข้อมูลบัลก์มีดังนี้

- สามารถใช้กับข้อมูลที่มีขนาดใหญ่มากๆ ได้
- มีการตรวจจับความผิดพลาดโดยผ่านทาง CRC และมีการรับประกันว่าการส่งสามารถส่งข้อมูลได้อย่างครบถ้วนถูกต้อง
- ไม่มีการรับประกันทางด้านแบนด์วิดธ์
- ใช้สตรีมไปป์(ทิศทางเดียว)
- มีใช้เฉพาะในโหมดฟูลสปีดและไฮสปีด

การส่งถ่ายข้อมูลแบบบัลก์สนับสนุนเฉพาะกับอุปกรณ์แบบฟูลสปีดและไฮสปีดเท่านั้น สำหรับขนาดสูงสุดของแพ็กเก็ตในโหมดฟูลสปีดจะมีขนาดเท่ากับ 8, 16, 32 หรือ 64 ไบต์ และสำหรับสูงสุดของแพ็กเก็ตในโหมดไฮสปีดสามารถมีค่าเท่ากับ 512 ไบต์



รูปที่ 2.20 เฟสของการส่งถ่ายข้อมูลบัลก์

ทรานแซคชันของบัลก์อาจประกอบไปด้วยการส่งถ่ายข้อมูล IN และ OUT ดังนี้

- IN : เมื่อโฮสพร้อมรับข้อมูลบัลก์มันจะส่งโทเค็น IN ออกมาถ้าอุปกรณ์ได้รับโทเค็น IN แล้วเกิดความผิดพลาดมันจะละทิ้งแพ็กเก็ตนี้ไป แต่ถ้าอุปกรณ์ได้รับโทเค็นอย่างถูกต้อง มันอาจจะทำการตอบกลับด้วยแพ็กเก็ต DATA ซึ่งบรรจุข้อมูลบัลก์ที่ต้องการส่ง หรืออาจจะส่งแพ็กเก็ต STALL เพื่อเป็นการบอกว่ามีความผิดพลาดเกิดขึ้นที่เอนด์พอยน์นั้นๆ สามารถทำงานได้แต่ไม่มีข้อมูลที่จะส่งไปในขณะนั้น
- OUT : เมื่อโฮสต้องการส่งข้อมูลบัลก์ มันจะส่งโทเค็น OUT ออกมาแล้วตามด้วยดาต้าแพ็กเก็ตซึ่งบรรจุข้อมูลบัลก์เอาไว้ ถ้าบางส่วนของโทเค็น OUT หรือดาต้าแพ็กเก็ตเกิดการผิดพลาดขึ้นมาอุปกรณ์จะละทิ้งแพ็กเก็ตนี้ไป ถ้าเอนด์พอยน์บัฟเฟอร์ของอุปกรณ์มีพื้นที่ว่างและมีการป้อนข้อมูลเข้าไปในเอนด์พอยน์บัฟเฟอร์ อุปกรณ์ก็จะส่ง ACK กลับมาเพื่อแจ้งแก่โฮสว่าได้ทำการรับข้อมูลสำเร็จแล้ว ถ้าเอนด์พอยน์บัฟเฟอร์ไม่มีพื้นที่ว่างเนื่องจากมีการประมวลผลแพ็กเก็ตก่อนหน้านี้ อุปกรณ์ก็จะส่ง NAK กลับออกมา แต่ถ้าเอนด์พอยน์เกิดการผิดพลาดขึ้นและบิต HALT ของมันถูกเซ็ต อุปกรณ์จะส่ง STALL กลับออกมา

ชนิดของการส่งถ่ายข้อมูล	คอนโทรล	บัลก์	อินเทอร์รัพท์	ไอโซโครนัส
การใช้งาน	การตั้งค่า	พรินเตอร์, สแกนเนอร์	เมาส์, คีย์บอร์ด	เสียง
ความจำเป็น	จำเป็น	ไม่จำเป็น	ไม่จำเป็น	ไม่จำเป็น
การสนับสนุนของอุปกรณ์โลว์สปีด	สนับสนุน	ไม่สนับสนุน	สนับสนุน	ไม่สนับสนุน
ทิศทางการไหลของข้อมูล	IN และ OUT	IN หรือ OUT	IN หรือ OUT (เวอร์ชัน 1.0 จะสนับสนุนเฉพาะ IN)	IN หรือ OUT
ปริมาณการจองแบนด์วิดธ์สูงสุด	10% ของเฟรมสำหรับโลว์สปีดและฟูลสปีด 20% ของไมโครเฟรมสำหรับไฮสปีด	ไม่มี	90% ของเฟรมสำหรับโลว์สปีด/ฟูลสปีด, 80% ของไมโครเฟรมสำหรับไฮสปีด (ไอโซโครนัสและอินเทอร์รัพท์รวมกัน)	
การแก้ไขความผิดพลาดของข้อมูล	มี	มี	มี	ไม่มี
ชนิดของข้อมูล (เมสเสจ สตรีม)	เมสเสจ	สตรีม	สตรีม	สตรีม
การรับประกันอัตราการส่ง	ไม่มี	ไม่มี	ไม่มี	มี
การรับประกันค่าเวลาสูงสุดระหว่างการส่งถ่ายข้อมูล	ไม่มี	ไม่มี	มี	มี

ตารางที่ 2.9 การส่งถ่ายข้อมูลของ USB แต่ละชนิด

บทที่ 3
โปรโตคอล TCP/IP

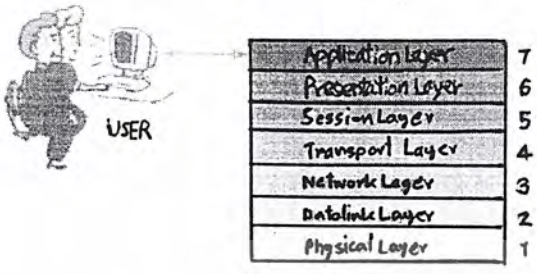
TCP/IP (Transmission Control Protocol / Internet Protocol) เป็นชุดโปรโตคอลที่มีการพัฒนามาตั้งแต่ปี 1960 และมีการปรับปรุงแก้ไขเรื่อยมาเพื่อให้สามารถใช้งานได้หลากหลายและมีประสิทธิภาพมากขึ้น โดยมีวัตถุประสงค์ให้สามารถใช้สื่อสารจากต้นทางข้ามเน็ตเวิร์กไปยังปลายทางได้ จึงจำเป็นต้องมีมาตรฐานในการจัดส่งข้อมูลไปยังปลายทางเพื่อให้ได้รับข้อมูลที่ถูกต้อง

3.1 OSI Model มาตรฐานที่ใช้ในการสื่อสารข้อมูล

เมื่อคอมพิวเตอร์เครื่องหนึ่งมีความต้องการรับส่งข้อมูลกับคอมพิวเตอร์เครื่องอื่น ๆ การเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์เข้าด้วยกันเป็นระบบเครือข่ายก็เกิดขึ้น ปัญหาที่ตามมาก็คือการเชื่อมต่อคอมพิวเตอร์ระหว่างระบบที่แตกต่างกันหรือคนละผู้ผลิตเป็นสิ่งที่ทำได้ยากในยุคต้นของการสื่อสารข้อมูล เนื่องจากขาดมาตรฐานส่วนกลางที่จำเป็นต้องใช้ในการรับส่งข้อมูล โดยมากและผู้ผลิตแต่ละรายก็จะมีมาตรฐานของตนเองซึ่งเข้ากันไม่ได้กับของผู้ผลิตรายอื่น ๆ ทำให้ผู้ใช้ต้องผูกติดอยู่กับผู้ผลิตรายใดรายหนึ่ง นับเป็นข้อจำกัดในการเชื่อมต่อคอมพิวเตอร์กันคนละชนิดไม่ให้รับส่งข้อมูลกันถึงกันได้

ปัญหานี้ได้ทำให้หน่วยงานกำหนดมาตรฐานสากล คือ ISO (International Standards Organization) ทำการกำหนดโครงสร้างทั้งหมดที่จำเป็นต้องใช้ในการสื่อสารข้อมูลจากคอมพิวเตอร์ระบบหนึ่งไปยังคอมพิวเตอร์อีกระบบหนึ่ง จุดมุ่งหมายนี้เพื่อเปิดช่องทางให้ข้อมูลที่เก็บอยู่ในระบบคอมพิวเตอร์หนึ่ง ๆ รับส่งไปยังคอมพิวเตอร์ที่เป็นระบบเดียวกันหรือต่างระบบได้อย่างอิสระ โดยไม่ขึ้นกับผู้ผลิตอย่างที่เป็นอยู่ในอดีต ซึ่งจะเรียกการทำงานในลักษณะนี้ว่า ระบบเปิด (Open System) เราเรียกโครงสร้างของมาตรฐานการรับส่งข้อมูลนี้ว่า OSI Model (Open System Interconnection) ซึ่งจัดทำขึ้นในปี ค.ศ. 1977 และทำการเผยแพร่ในปี ค.ศ. 1984

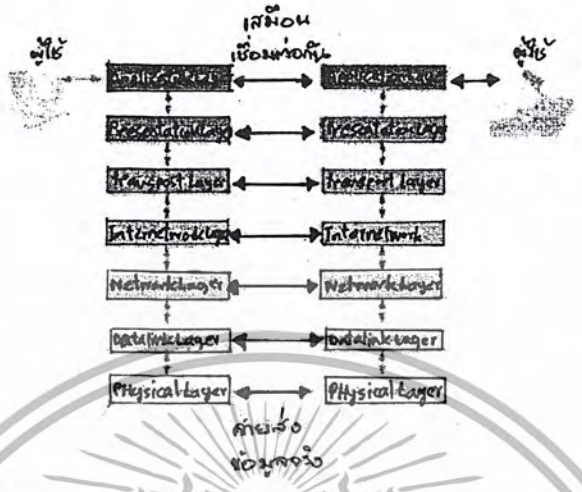
OSI Model กำหนดให้การสื่อสารข้อมูลจากระบบคอมพิวเตอร์หนึ่งไปยังอีกระบบหนึ่ง แบ่งออกเป็น 7 ชั้นย่อย ๆ ซึ่งคอมพิวเตอร์ทั้งสองระบบจะมีขั้นตอนทั้ง 7 ชั้นนี้เหมือนกันทั้งสองฝั่ง เราเรียกชื่อเต็ม ๆ ของแบบการสื่อสารข้อมูลนี้ว่า OSI 7-Layer Reference Model ดังแสดงในรูป 3.1



รูปที่ 3.1 โครงสร้างของ OSI 7-Layer Reference Model

แต่ละชั้นของแบบการสื่อสารข้อมูลเราจะเรียกว่า Layer หรือ " ชั้น " ของแบบการสื่อสารข้อมูลนั้นเอง ประกอบด้วยชั้นย่อย ๆ 7 ชั้น ในแต่ละชั้นหรือแต่ละ Layer จะเสมือนเชื่อมต่อเพื่อส่งข้อมูลกับชั้นเดียวกันในคอมพิวเตอร์อีกด้านหนึ่ง แต่ในการเชื่อมต่อกันจริง ๆ แล้วจะมีเพียง Layer ที่ 1 ซึ่งเป็นชั้นล่างสุดเท่านั้นที่มีการรับส่งข้อมูลเกิดขึ้นผ่านสายส่งข้อมูลระหว่างคอมพิวเตอร์ทั้งสองเครื่อง สำหรับใน Layer อื่น ๆ จะไม่ได้เชื่อมต่อกันจริง เพียงแต่ทำงานเสมือนกับว่ามีการติดต่อรับส่งข้อมูลกับชั้นเดียวกันของคอมพิวเตอร์อีกด้านหนึ่งเท่านั้น

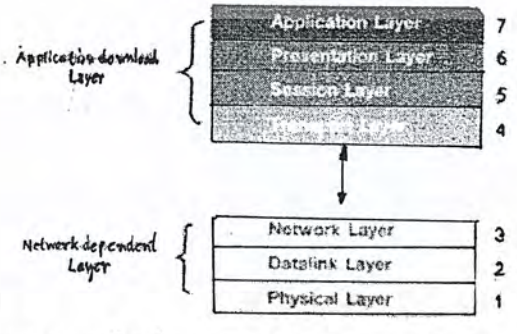
คุณสมบัติข้อที่สองของ OSI Model คือ แต่ละชั้นที่ทำหน้าที่รับส่งข้อมูลจะมีการติดต่อรับส่งข้อมูลกับชั้นที่อยู่ติดกับตัวเองเท่านั้น จะติดต่อรับส่งข้อมูลข้ามกระโดดไปชั้นอื่น ๆ ในคอมพิวเตอร์ของตัวเองไม่ได้ เช่น คอมพิวเตอร์ด้านที่ส่งข้อมูลออกไปให้ผู้รับใน Layer ที่ 7 ซึ่งอยู่บนสุดของด้านส่งข้อมูล จะเชื่อมต่อเข้ากับ Layer ที่ 6 เท่านั้น ซึ่ง Layer ที่ 6 นี้ก็จะมีการเชื่อมต่อรับส่งข้อมูลกับ Layer ที่ 7 และ Layer ที่ 5 จะไม่มีเหตุการณ์ที่ Layer ที่ 7 จะกระโดดไปทำการรับส่งข้อมูลกับ Layer ที่ 4 หรือ 5 การส่งข้อมูลจะต้องไล่ลำดับชั้นลงมาจนถึง Layer ที่ 1 ซึ่งจะเป็นชั้นเดียวที่เชื่อมต่อจริงเข้ากับคอมพิวเตอร์ด้านรับข้อมูลผ่านสายส่งข้อมูล และจะทำการรับข้อมูลจาก Layer ที่ 1 ไล่ขึ้นไปจนถึง Layer ที่ 7 ตามลำดับ แต่ละชั้นสื่อสารในด้านส่งจะทำงานเสมือนว่าเชื่อมต่อเพื่อรับส่งข้อมูลกับชั้นที่ตรงกันในคอมพิวเตอร์ด้านรับดังแสดงในรูปที่ 3.2



รูปที่ 3.2 การรับส่งข้อมูลของ OSI 7-Layer Reference Model

ผู้ใช้จะติดต่อรับส่งข้อมูลผ่านทาง Layer ที่ 7 ซึ่งอยู่ด้านบนสุดของ OSI Model เท่านั้น ในทางทฤษฎีแล้วแต่ละชั้นของการรับส่งข้อมูลจะมีฟังก์ชั่นการทำงานที่แน่นอน และแยกเด็ดขาดออกจากกัน ดังนั้นบริษัทผู้ผลิตอุปกรณ์หรือโปรโตคอลสามารถสร้างผลิตภัณฑ์ในชั้นสื่อสารที่ตนถนัด และนำแต่ละชั้นของผู้ผลิตแต่ละรายมาเชื่อมต่อกันได้อย่างไม่มีขีดจำกัด แต่ในทางปฏิบัติ OSI Model ได้แบ่งตามลักษณะออกเป็น 2 กลุ่มใหญ่ กลุ่มแรกได้แก่ 4 ชั้นสื่อสารด้านบน คือ Layer ที่ 7,6,5 และ 4 ทำหน้าที่เชื่อมต่อรับส่งข้อมูลระหว่างผู้ใช้โปรแกรมประยุกต์ เพื่อให้รับส่งข้อมูลกับฮาร์ดแวร์ที่อยู่ชั้นล่างได้อย่างถูกต้อง เรียกว่า Application-oriented Layer ซึ่งจะเกี่ยวข้องกับซอฟต์แวร์เป็นหลัก

กลุ่มที่สองจะเป็นชั้นล่าง ได้แก่ Layer ที่ 3,2 และ 1 ทำหน้าที่เกี่ยวกับการรับส่งข้อมูลผ่านสายส่ง และควบคุมการรับส่งข้อมูล ตรวจสอบข้อผิดพลาด รวมทั้งเลือกเส้นทางที่ใช้ในการรับส่งข้อมูล ซึ่งจะเกี่ยวข้องกับฮาร์ดแวร์เป็นหลักเรียกว่า Network-dependent Layer ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 การแบ่งกลุ่มของ OSI 7-Layer Reference Model

ซึ่งในส่วนของ 3 ชั้นล่างสุด หรือ Layer ที่ 1,2 และ 3 นั้นมักจะเกี่ยวข้องกับฮาร์ดแวร์และโปรแกรมควบคุมฮาร์ดแวร์เป็นหลัก ทำให้สามารถแยกแต่ละชั้นออกจากกันได้ง่าย และใช้ผลิตภัณฑ์ของต่างบริษัทกันได้อย่างไม่มีปัญหา

OSI Model ที่แบ่งการรับส่งข้อมูลระหว่างคอมพิวเตอร์ออกเป็น 7 ชั้น ในแต่ละชั้นมีชื่อเรียกและหน้าที่การทำงานดังนี้

3.1.1 Layer ที่ 7 Application Layer

เป็นชั้นที่อยู่บนสุดของขบวนการรับส่งข้อมูล ทำหน้าที่ติดต่อกับผู้ใช้ โดยจะรับคำสั่งต่าง ๆ จากผู้ใช้ส่งให้คอมพิวเตอร์แปลความหมาย และทำงานตามคำสั่งที่ได้รับในระดับโปรแกรมประยุกต์ เช่น แปลความหมายของการกดปุ่มบนเมาส์ให้เป็นคำสั่งในการก๊อปปี้ไฟล์ หรือดึงข้อมูลมาแสดงบนจอภาพ เป็นต้น ซึ่งการแปลคำสั่งจากผู้ใช้ส่งให้คอมพิวเตอร์รับไปทำงานนี้ จะต้องแปลออกมาถูกต้องตามกฎ ที่ใช้ในระเบียบปฏิบัติการของคอมพิวเตอร์นั้น ๆ ตัวอย่างเช่น ถ้ามีการก๊อปปี้ไฟล์เกิดขึ้นในระบบ คำสั่งที่ใช้จะต้องสร้างไฟล์ได้ถูกต้อง มีชื่อไฟล์ยาวไม่เกินจำนวนที่ระบบปฏิบัติการใช้อยู่ และชื่อไฟล์ต้องประกอบด้วยตัวอักษรที่กำหนด สิ่งต่าง ๆ เหล่านี้จะเกิดขึ้นใน Layer ที่ 7 ของการสื่อสารข้อมูล รวมทั้งฟังก์ชั่นในการเชื่อมต่อรับส่งข้อมูลระหว่าง Layer ที่ 7 กับ Layer ที่ 6 ด้วย

3.1.2 Layer ที่ 6 Presentation Layer

เป็นชั้นที่ทำหน้าที่ตกลงกับคอมพิวเตอร์อีกด้านหนึ่งในชั้นเดียวกันว่า การรับส่งข้อมูลในระดับโปรแกรมประยุกต์จะมีขั้นตอนและข้อบังคับอย่างไร ข้อมูลที่ทำการรับส่งกันใน Layer ที่ 6 นี้จะอยู่ในรูปแบบของข้อมูลชั้นสูง ซึ่งอยู่ในรูปแบบของคำสั่งที่มีกฎ (Syntax) บังคับอย่างแน่นอน เช่น ในการก๊อปปี้ไฟล์ก็จะมีขั้นตอนย่อยประกอบกัน คือสร้างไฟล์ที่กำหนดขึ้นมาเสียก่อน จากนั้นจึงเปิดไฟล์ แล้วทำการรับข้อมูลจากปลายทางมาเก็บลงในไฟล์ที่สร้างขึ้นใหม่นี้ โดยเนื้อหาของข้อมูลที่ทำการรับส่งระหว่างกัน ก็คือคำสั่งของขั้นตอนย่อย ๆ ข้างต้นนั่นเอง คำสั่งเหล่านี้จะต้องหมายถึงว่าจะให้ทำอะไรบ้างและถูกต้องตามกฎด้วย นอกจากนี้ใน Layer ที่ 6 ยังทำหน้าที่แปลความหมายของคำสั่งที่ได้รับจาก Layer ที่ 7 ให้เป็นคำสั่งระดับปฏิบัติการส่งให้ Layer ที่ 5 ต่อไปอีกด้วย

3.1.3 Layer ที่ 5 Session Layer

ทำหน้าที่ควบคุม "จังหวะ" ในการรับส่งข้อมูลของคอมพิวเตอร์ทั้งสองด้าน ที่รับส่งแลกเปลี่ยนข้อมูลกันให้มีความสอดคล้องกัน (Synchronization) และกำหนดวิธีที่ใช้รับส่งข้อมูล เช่น อาจจะเป็นในลักษณะสลับกันส่ง (Half Duplex) หรือรับส่งข้อมูลพร้อมกันทั้งสองด้าน (Full Duplex) ซึ่งใน Layer ที่ 5 นี้จะเป็นชั้นที่ใช้ควบคุมการรับส่งข้อมูลในลักษณะดังกล่าว ข้อมูลที่รับส่งกันใน Layer ที่ 5 นี้จะอยู่ในรูปของ dialog หรือประโยคของข้อมูลที่สนทนาโต้ตอบกันระหว่างด้านรับและด้านที่ส่งข้อมูล ไม่ได้มองเป็นคำสั่งอย่างใน Layer ที่ 6 เช่น เมื่อผู้รับได้รับข้อมูลส่วนแรกจากผู้ส่ง ก็จะโต้ตอบกลับไปให้ผู้ส่งรู้ว่าได้รับข้อมูลส่วนแรกเรียบร้อยแล้ว และพร้อมที่จะรับข้อมูลส่วนที่สองต่อไป คล้ายกับการสนทนาโต้ตอบกันระหว่างผู้รับและผู้ส่งนั่นเอง

3.1.4 Layer ที่ 4 Transport Layer

ทำหน้าที่เชื่อมต่อการรับส่งข้อมูลระดับสูงของ Layer ที่ 5 มาเป็นประโยคของข้อมูลที่รับส่งในระดับฮาร์ดแวร์ เช่น แปลงที่หรือชื่อคอมพิวเตอร์ในเครือข่ายให้เป็น Network address พร้อมทั้งเป็นชั้นที่ควบคุมการรับส่งข้อมูลจากปลายด้านส่งถึงปลายด้านรับข้อมูล ให้ข้อมูลมีการไหลลื่นตลอดเส้นทางตามจังหวะที่ควบคุมจาก Layer ที่ 5 โดยใน Layer ที่ 4 นี้จะเป็นรอยต่อระหว่างการรับส่งข้อมูลของซอฟต์แวร์กับฮาร์ดแวร์การรับส่งข้อมูลของระดับสูงจะถูกแยกจากฮาร์ดแวร์ที่ใช้รับส่งข้อมูลที่ Layer ที่ 4 นี้ และจะไม่มีส่วนใดผูกติดกับฮาร์ดแวร์ที่ใช้รับส่งข้อมูลในระดับล่าง ดังนั้นฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ควบคุมการรับส่งข้อมูลในระดับต่างส่งไปจาก Layer ที่ 4 จึงสามารถสับเปลี่ยน และใช้ข้ามไปมากับซอฟต์แวร์รับส่งข้อมูลในระดับสูงที่อยู่ข้างบน (ตั้งแต่ Layer ที่ 4 ขึ้นไปถึง Layer ที่ 7) ได้ง่าย หน้าที่อีกประการหนึ่งของ Layer ที่ 4 คือ การควบคุมคุณภาพของการรับส่งข้อมูลให้มีมาตรฐานในระดับที่ตกลงกันของทั้งสองฝ่าย และการตัดข้อมูลออกเป็นส่วนย่อย ๆ ให้เหมาะสมกับลักษณะการทำงานของฮาร์ดแวร์ที่ใช้ในเครือข่าย เช่น หาก Layer ที่ 5 ต้องการรับส่งข้อมูลที่มีความยาวเกินกว่าที่ระบบเครือข่ายจะส่งได้ Layer ที่ 4 ก็จะทำหน้าที่ตัดข้อมูลออกเป็นส่วนย่อย ๆ แล้วส่งไปให้ผู้รับ ข้อมูลที่ได้รับปลายทางก็จะถูกนำมาต่อกันที่ Layer ที่ 4 ของด้านผู้รับ และส่งให้ Layer ที่ 5 ต่อไป

3.1.5 Layer ที่ 3 Network Layer

ทำหน้าที่เชื่อมต่อคอมพิวเตอร์ของด้านรับและด้านส่งเข้าหากันผ่านระบบเครือข่าย พร้อมทั้งเลือกหรือกำหนดเส้นทางที่จะใช้ในการรับส่งข้อมูลระหว่างกัน และส่งผ่านข้อมูลที่ได้รับไปยังอุปกรณ์ในเครือข่ายต่าง ๆ จนกระทั่งถึงปลายทางใน Layer ที่ 3 นี้ข้อมูลที่รับส่งกันจะอยู่ในรูปแบบของกลุ่มข้อมูลที่เรียกว่า Packet หรือ Frame ข้อมูล Layer ที่ 4,5,6 และ 7 มองเห็นเป็นคำสั่งและ dialog ต่าง ๆ นั้นจะถูกแปลงและผนึกรวมอยู่ในรูปของ Packet หรือ Frame ที่มีเพียงแอดเดรสของผู้รับ, ผู้ส่ง, ลำดับการรับส่ง และส่วนของข้อมูลเท่านั้น ตัวเนื้อหาของข้อมูลจะไม่มีผลใด ๆ ในการรับส่งข้อมูลเลย ไม่ว่าข้อมูลในระดับสูงจะเป็น วิดีโอ, ภาพ, เสียง หรือข้อมูลอื่นใดก็ตาม แต่ใน Layer ที่ 3 จะมองข้อมูลทั้งหมดเป็น Packet หรือ Frame เท่านั้น หน้าที่อีกประการหนึ่งของ Layer ที่ 3 นี้คือการทำ Call Setup หรือเรียกติดต่อคอมพิวเตอร์ปลายทางก่อนการรับส่งข้อมูล และการทำ Call Clearing หรือยกเลิกการติดต่อเมื่อการรับส่งข้อมูลจบลงแล้ว ในกรณีที่การรับส่งข้อมูลนั้นต้องมีการติดต่อกันก่อน

3.1.6 Layer ที่ 2 Datalink Layer

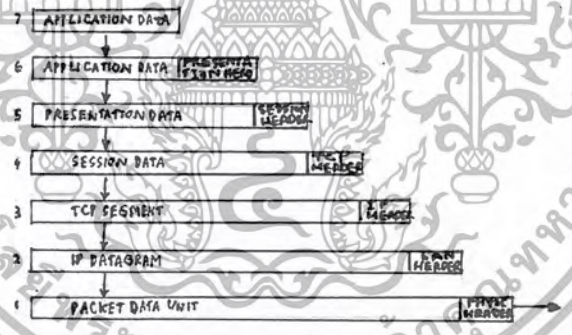
เป็นชั้นที่ทำหน้าที่เชื่อมต่อการรับส่งข้อมูลในระดับฮาร์ดแวร์ โดยเมื่อมีการสั่งให้รับข้อมูลจากใน Layer ที่ 3 ลงมา Layer ที่ 2 จะทำหน้าที่แปลคำสั่งนั้นให้เป็นคำสั่งควบคุมฮาร์ดแวร์ที่ใช้รับส่งข้อมูล ทำการตรวจสอบข้อผิดพลาดในการรับส่งข้อมูลของระดับฮาร์ดแวร์ และแก้ไขข้อผิดพลาดที่ตรวจพบนั้น ข้อมูลที่อยู่ใน Layer ที่ 2 นี้จะอยู่ในรูปของ Frame คือกลุ่มของข้อมูลที่มีรูปร่างตามข้อบังคับฮาร์ดแวร์ที่ใช้ในการรับส่งข้อมูล เช่น ถ้าฮาร์ดแวร์ที่ใช้นั้น Ethernet Lan ข้อมูลก็จะมีรูปร่างของ Frame ตามที่ระบุไว้ในมาตรฐานของ Ethernet หากว่าฮาร์ดแวร์ที่ใช้รับส่งข้อมูลเป็นชนิดอื่น เช่น Token Ring Lan การรับส่งข้อมูลก็จะเปลี่ยนไปตามมาตรฐานนั้น ๆ

3.1.7 Layer ที่ 1 Physical Layer

เป็นชั้นล่างสุดของขั้นตอนในการรับส่งข้อมูลของ OSI Model ซึ่งเป็นชั้นเดียวที่มีการเชื่อมต่อกันทางกายภาพระหว่างคอมพิวเตอร์สองระบบที่ทำการรับส่งข้อมูลกัน ใน Layer ที่ 1 นี้จะกำหนดคุณสมบัติทางกายภาพของฮาร์ดแวร์ที่ใช้เชื่อมต่อระหว่างคอมพิวเตอร์ทั้งสองระบบ เช่น สายที่ใช้รับส่งข้อมูลเป็นแบบไหน ข้อต่อหรือปลั๊กที่ใช้ในการรับส่งข้อมูลมีมาตรฐานอย่างไร ใช้ไฟกี่โวลต์ ความเร็วในการรับส่งข้อมูลเป็นเท่าใด สัญญาณที่ใช้รับส่งข้อมูลมีรูปร่างอย่างไร ข้อมูลใน Layer ที่ 1 นี้จะมองเห็น

เป็นการรับส่งข้อมูลทีละบิตเรียงต่อกันไป โดยไม่มีการพิจารณาเรื่องความหมายของข้อมูลเลย การรับส่งจะส่งข้อมูล "0" หรือ "1" ไปให้คอมพิวเตอร์ด้านรับข้อมูลในระดับฮาร์ดแวร์เท่านั้น หากการรับส่งข้อมูลมีปัญหาเนื่องจากฮาร์ดแวร์ เช่น สายสัญญาณที่ใช้รับส่งข้อมูลขาด, อุปกรณ์เสียหาย ก็จะเป็นหน้าที่ของ Layer ที่ 1 นี้เช่นกันที่จะตรวจสอบและแจ้งข้อผิดพลาดนั้นให้ชั้นอื่น ๆ ที่อยู่เหนือขึ้นไปทราบ

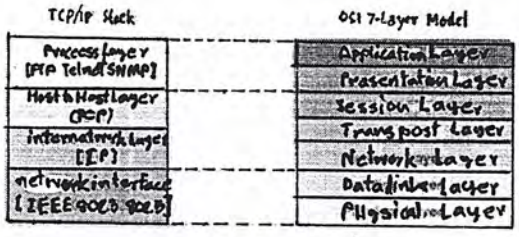
ในการรับส่งข้อมูลใน OSI Model นั้น ข้อมูลจากชั้นบนสุดคือ Layer ที่ 7 เมื่อถูกส่งลงไปในชั้นถัดไป ข้อมูลเดิมก็จะถูกผนึกรวมกับข้อมูลที่ใช้ควบคุมของแต่ละชั้นซ้อน ๆ กันเป็นลำดับเท่ากับจำนวนชั้นที่ผ่านลงไป ตัวอย่างเช่น Application Data เมื่อถูกส่งลงไปยังชั้นถัดไปก็จะถูกผนึกด้วย Application Header และทั้ง Application Header และ Application Data จะรวมกันเป็นข้อมูลของชั้นที่อยู่ถัดลงไปอีก ซึ่งชั้นที่อยู่ถัดลงไปอีกก็จะผนึกข้อมูลด้วย Header ของมันเองอีกครั้งหนึ่ง และทั้ง Header และข้อมูลเดิมนี้ก็จะกลายเป็นข้อมูลในชั้นถัดไปถึงปลายทาง ข้อมูลที่ได้รับจะถูกแยก Header ที่เพิ่มเข้ามานี้ออกทีละชั้น ซึ่งเป็นขบวนการย้อนกลับด้านส่ง จนกระทั่งถึงชั้นบนสุด จึงจะเป็นข้อมูลของ Application Data ให้ผู้รับตามต้องการ ดังแสดงในรูป 3.4



รูปที่ 3.4 การรับส่งข้อมูลแต่ละชั้นของ TCP/IP

3.2 การแบ่งชั้นเลเยอร์ (Layering) ของโปรโตคอล TCP/IP

เนื่องจาก OSI Model(Open System Interconnection) เป็นมาตรฐานที่เกิดหลังจากมีการใช้โปรโตคอล TCP/IP แล้ว แต่โปรโตคอล TCP/IP ก็มีการแบ่งโปรโตคอลสื่อสารออกเป็นชั้น ๆ เช่นกัน โดยมีจำนวนชั้นอยู่ 4 ชั้น โดยสามารถเปรียบเทียบกับ OSI Model ได้ดังรูปที่ 3.5



รูปที่ 3.5 โปรโตคอล TCP/IP เมื่อเทียบกับ OSI 7-Layer Reference Model

- ชั้นบนเรียกว่า Process Layer จะเป็น Application Protocol ที่ทำหน้าที่เชื่อมต่อกับผู้ใช้งาน
- ชั้นถัดมาเรียกว่า Host-to-Host Layer จะเป็น TCP หรือ UDP ที่ทำหน้าที่คล้ายกับ Layer ที่ 4 ของ OSI Model คือควบคุมการรับส่งข้อมูลจากปลายด้านส่งถึงปลายด้านรับข้อมูล
- ชั้นถัดลงมาคือ Internetwork Layer ได้แก่ส่วนของโปรโตคอล IP ซึ่งทำหน้าที่คล้ายกับ Layer ที่ 3 ของ OSI Model คือเชื่อมต่อคอมพิวเตอร์เข้ากับระบบเครือข่ายที่อยู่ชั้นล่างลงไป
- ชั้นสุดท้ายคือ Network Interface คือชั้นที่ควบคุมฮาร์ดแวร์การรับส่งข้อมูลผ่านเครือข่ายซึ่งเทียบได้กับ Layer ที่ 1 และ 2 ของ OSI Model

แนวความคิดหลักของระบบเครือข่ายคอมพิวเตอร์ก็คือ การเชื่อมโยงอุปกรณ์เข้าด้วยกัน ไม่ว่าจะเป็นเครื่องเซิร์ฟเวอร์, ไคลเอน หรืออุปกรณ์ในเครือข่ายอื่น ๆ เพื่อให้สามารถแชร์การใช้อุปกรณ์ร่วมกันได้ หรือสามารถส่งผ่านข้อมูล ไปมาระหว่างกัน ได้ถูกต้อง เมื่อมีการเชื่อมต่อกันแล้วก็จำเป็นต้องมีการกำหนดหรือระบุเลขหมายของอุปกรณ์ทุกชิ้นทุกชนิด ในเครือข่าย เพื่อให้อ้างอิงได้โดยไม่ซ้ำกัน เพราะถ้าซ้ำกันแล้วการรับส่งข้อมูลอาจจะไม่ถึงมือผู้รับปลายทางได้อย่างถูกต้อง เลขหมายดังกล่าวจะเรียกว่า แอดเดรส (Address) หรือเลขหมายประจำตัวที่มีข้อกำหนดเป็นมาตรฐาน ซึ่งในการใช้งานโปรโตคอล TCP/IP ที่เชื่อมโยงในเครือข่าย หมายเลขที่ใช้อ้างอิงถึงกันจะเรียกว่า IP Address (Internet Protocol Address)

3.3 IP Address

IP Address ถูกกำหนดขึ้นมาให้เป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออยู่ในระบบเครือข่าย โดยการกำหนด IP Address ให้แต่ละเครื่องหรือแต่ละอุปกรณ์นี้จะต้องไม่ซ้ำกันซึ่ง IP Address นี้จะไม่ถูกผูกติดกับฮาร์ดแวร์แต่อย่างใด จึงสามารถกำหนดใหม่หรือแก้ไขเปลี่ยนแปลงได้

เมื่อมีการเปลี่ยนตัวฮาร์ดแวร์ ทั้งนี้เนื่องจากเป็นการกำหนดด้วยซอฟต์แวร์แตกต่างกันกับหมายเลข MAC Address (Media Access Control address) ซึ่งเป็นหมายเลขประจำตัวของอุปกรณ์ที่ต่ออยู่ในเครือข่าย ค่า MAC Address จะถูกกำหนดจากบริษัทผู้ผลิตอุปกรณ์ตั้งแต่เริ่มผลิต เช่น อุปกรณ์ Network Interface Card (NIC) จะมีค่า MAC Address ประจำตัวที่ไม่ซ้ำกันและไม่สามารถแก้ไขได้ค่า MAC Address เป็นการระบุค่าอ้างอิงของอุปกรณ์ฮาร์ดแวร์ในระดับล่างสุด (Physical Layer) ของกลไกรับส่งข้อมูลภายในเครือข่าย ถ้าจะใช้หมายเลข MAC Address สำหรับระบุอ้างอิงกันในเครือข่ายแล้วจะเกิดปัญหามาก เมื่อมีการเปลี่ยนหรือย้ายเครื่องต้องทำการกำหนดระบบเครือข่ายใหม่ นอกจากนี้ยังจดจำได้ยากกว่า ตัวอย่างของหมายเลข MAC Address คือ 08:0a:0c:12:b5:05 การที่ IP Address ถูกใช้อ้างอิงในการติดต่อกันด้วยโปรโตคอล TCP/IP เพราะการใช้ IP Address จะยืดหยุ่นและคล่องตัวกว่า

การทำงานของโปรโตคอล IP จำเป็นต้องกำหนด IP Address เพื่อระบุและอ้างอิงอุปกรณ์ต่าง ๆ ที่ต่ออยู่ในเครือข่ายไม่ว่าจะเป็นเครื่องเซิร์ฟเวอร์ หรือ โฮสเอน IP Address จะเป็นค่าตัวเลขขนาด 32 บิต ถูกแบ่งออกเป็นส่วนละ 8 บิต รวมเป็น 4 ส่วน และคั่นแต่ละส่วนด้วยเครื่องหมายจุด ดังนั้นค่าตัวเลขในแต่ละส่วนจะมีได้ตั้งแต่ 0 ถึง 255 (2^8) ตัวอย่างเช่น 205.144.78.1 นอกจากนี้ IP Address ยังถูกแบ่งออกระดับชั้นของเครือข่าย เรียกว่า network class ซึ่งการกำหนดให้มี network class ก็เพื่อให้สามารถแจกจ่าย IP Address ให้กับเครือข่ายต่าง ๆ ได้อย่างเหมาะสมเพราะในแต่ละเครือข่ายมักจะแตกต่างกัน บางเครือข่ายมีจำนวนเครื่องลูกข่ายมาก บางเครือข่ายมีน้อย ฉะนั้นถ้าไม่มีการจัดลำดับของเครือข่ายให้ดี IP Address ก็จะถูกใช้งานอย่างสิ้นเปลืองและใช้งานไม่ได้เต็มจำนวนที่มี ลำดับชั้นของเครือข่ายแบ่งออกได้ดังตารางที่ 3.1

Class A	IP Address บิตแรกของไบต์แรกสุดจะเป็น 0 เสมอ
Class B	IP Address 2 บิตแรกของไบต์แรกสุดจะเป็น 1 และ 0 เสมอ
Class C	IP Address 3 บิตแรกของไบต์แรกสุดจะเป็น 1, 1 และ 0 เสมอ
Class D	กำหนดเป็น IP Address สำรองไว้สำหรับส่งข้อมูลแบบ multicast จะไม่มีการแจกจ่ายให้ใช้งานทั่วไป
Class E	กำหนดเป็น IP Address พิเศษที่ใช้สำหรับทดสอบและพัฒนา ไม่มีการกำหนดให้ใช้งานทั่วไป

ตารางที่ 3.1 แสดงการแบ่ง Class ของ IP Addrss

บทที่ 4
ทฤษฎีและหลักการระบบควบคุม

4.1 ระบบควบคุม

ในระบบควบคุมที่มีอยู่เราสามารถที่จะแบ่งแยกระบบควบคุมนั้นๆ ออกตามคุณลักษณะของการทำงาน ซึ่งเมื่อพิจารณาแล้วเราจะสามารถจำแนกได้เป็น 2 ประเภท คือ ระบบควบคุมแบบวงเปิด (Open-loop control system) และระบบควบคุมแบบวงปิดหรือป้อนกลับ (Closed-loop control system or Feedback control system)

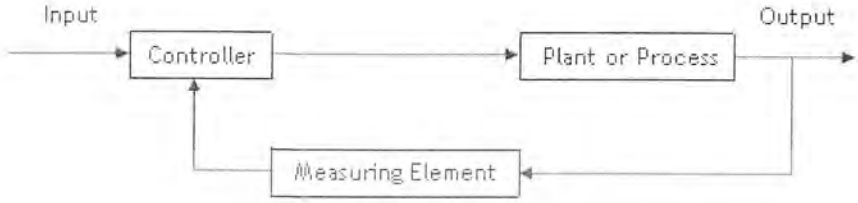
4.1.1ระบบควบคุมแบบวงเปิด

ระบบควบคุมแบบวงเปิดเป็นระบบควบคุมที่เอาท์พุทของระบบจะไม่มีผลต่อการควบคุมเลย นั่นคือ ในกรณีของระบบควบคุมแบบนี้นั้น เอาท์พุทของระบบจะไม่ถูกวัดหรือป้อนกลับเพื่อนำมาเปรียบเทียบกับอินพุท รูปที่ 4.1 เป็นบล็อคไดอะแกรมแสดงถึงความสัมพันธ์ระหว่างอินพุทและเอาท์พุทของระบบควบคุมแบบวงเปิด

รูปที่ 4.1ระบบควบคุมแบบวงเปิด

4.1.2 ระบบควบคุมแบบวงปิด

ระบบควบคุมแบบนี้ เป็นระบบควบคุมที่นำสัญญาณเอาท์พุทที่ได้จากกระบวนการนั้นป้อนกลับมาเปรียบเทียบกับสัญญาณอินพุทที่ต้องการ ซึ่งจะได้สัญญาณความคลาดเคลื่อนโดยเป็นสัญญาณความแตกต่างระหว่างสัญญาณอินพุทและสัญญาณป้อนกลับและจะถูกป้อนให้กับตัวควบคุมอีกครั้งเพื่อที่จะเป็นการลดค่าความคาดเคลื่อนให้น้อยลง แสดงบล็อคไดอะแกรมระบบควบคุมแบบวงปิดดังรูปที่ 4.2



รูปที่ 4.2 ระบบควบคุมแบบวงปิด

การควบคุมที่นิยมใช้ในกระบวนการ (Process) ได้แก่ การควบคุมแบบป้อนกลับ ซึ่งเป็นการควบคุมแบบใช้มนุษย์ปรับค่าควบคุมหรือแบบอัตโนมัติก็ได้ ตัวอย่างเช่น ในการควบคุมอุณหภูมิของน้ำในอ่างอาบน้ำ ผู้ควบคุมอาจจะใช้มือข้างหนึ่งจุ่มลงในอ่างเพื่อวัดอุณหภูมิ และใช้มืออีกข้างปรับน้ำร้อนที่ไหลเข้าให้มากหรือน้อย เพื่อให้น้ำในอ่างมีอุณหภูมิตามต้องการ การควบคุมแบบนี้จะเรียกว่าเป็นการควบคุมป้อนกลับแบบใช้มนุษย์ปรับค่าควบคุม (Manual) ถ้าผู้ควบคุมใช้เทอร์โมมิเตอร์ในการวัดอุณหภูมิแล้ว เขาก็จะสามารถวัดอุณหภูมิของน้ำในอ่างได้เที่ยงตรงมากขึ้น ซึ่งจะเป็นว่าถ้าทำการวัดได้อย่างเที่ยงตรงเท่าใดก็จะสามารถทำให้การควบคุมได้ดีขึ้น สำหรับในกรณีการควบคุมป้อนกลับแบบอัตโนมัตินั้นจะต้องใช้อุปกรณ์วัดอุณหภูมิของน้ำในอ่างแทนคนควบคุม จากนั้นจึงส่งค่าของอุณหภูมิของน้ำร้อนที่วัดได้ไปยังเครื่องควบคุมด้วยเครื่องส่งสัญญาณเพื่อไปทำการเปรียบเทียบกับค่าของอุณหภูมิของน้ำที่ต้องการ และเครื่องควบคุมก็จะสร้างสัญญาณควบคุมเพื่อไปควบคุมการปิดเปิดของวาล์วควบคุมที่ทำหน้าที่ควบคุมปริมาณของน้ำร้อนที่ไหลเข้าอ่างให้มากขึ้นหรือน้อยลง เพื่อทำให้น้ำในอ่างมีอุณหภูมิตามต้องการ

4.2 ตัวควบคุมแบบป้อนกลับ (Feedback Controller)

ตัวควบคุมทำหน้าที่ตรวจสอบสภาพของกระบวนการ โดยรับสัญญาณวัดจากเครื่องวัดเปรียบเทียบกับค่าเป้าหมาย และสร้างสัญญาณควบคุมที่เหมาะสมเพื่อปรับสภาพของกระบวนการให้เข้าสู่เป้าหมายตามที่ต้องการ

ชนิดของตัวควบคุมอัตโนมัติที่ใช้ในกระบวนการอุตสาหกรรม จำแนกออกได้ตามลักษณะการควบคุม ดังรายละเอียดดังต่อไปนี้

4.2.1 การควบคุมแบบ 2 ตำแหน่ง

ในการควบคุมแบบ 2 ตำแหน่งการควบคุมจะทำงานในตำแหน่งที่คงที่เพียง 2 ตำแหน่งเท่านั้น ในบางครั้งจึงมีชื่อเรียกว่า การควบคุมแบบเปิด-ปิด การควบคุม 2 ตำแหน่งนั้นจะเป็นการควบคุมแบบง่ายๆและราคาไม่แพง ดังนั้นจึงนิยมใช้กันอย่างกว้างขวางในงานควบคุมอุตสาหกรรมและในกรณีที่ผลจากการแกว่ง (Oscillate) นั้นเป็นที่ยอมรับได้

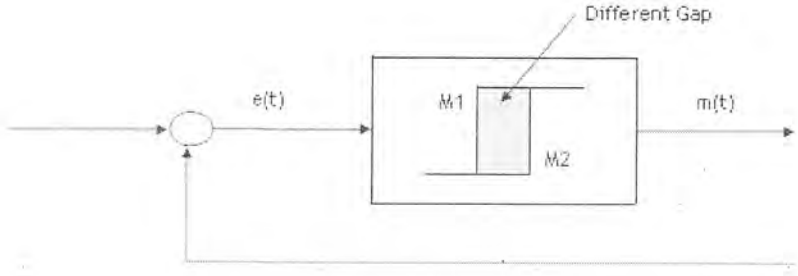
กำหนดให้สัญญาณเอาท์พุทของตัวควบคุมเป็น m(t) และสัญญาณค่าความคลาดเคลื่อนเป็น e(t) ฉะนั้นในการควบคุมแบบ 2 ตำแหน่งนั้น สัญญาณ จะมีค่าเพียงค่าสูงสุดและค่าต่ำสุดเท่านั้น โดยจะขึ้นอยู่กับว่าสัญญาณค่าความคาดเคลื่อนมีค่าเป็นบวกหรือลบ นั่นคือ

$M(t) = M1$ สำหรับ $e(t) > 0$; โดยที่ M1 และ M2 เป็นค่าคงที่

$M(t) = M2$ สำหรับ $e(t) < 0$

รูปที่ 4.3 แสดงถึงบล็อกไดอะแกรม ของตัวควบคุมแบบ 2 ตำแหน่ง และสำหรับช่วงซึ่งสัญญาณค่าความคลาดเคลื่อนเปลี่ยนแปลงไปก่อนเกิดการเปลี่ยนตำแหน่ง (Switching) ของการควบคุมนั้นเรียกว่า Difference Gap ดังแสดงดังรูปที่ 4.4 ช่วง Difference Gap มีบางครั้งเป็นการทำให้เกิดขึ้นเพื่อป้องกันการเปิด-ปิดบ่อยครั้งเกินไป

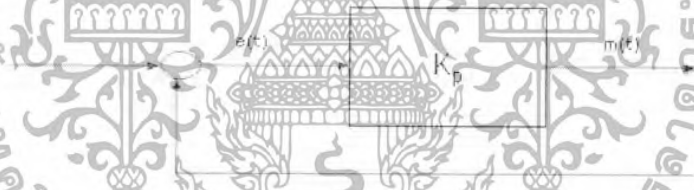
รูปที่ 4.3 บล็อกไดอะแกรมของตัวควบคุม 2 ตำแหน่ง



รูปที่ 4.4 ช่วง Difference Gap

4.2.2 การควบคุมแบบสัดส่วน (Proportional Control Action)

เป็นกิริยาการควบคุมซึ่งค่อนข้างจะสม่ำเสมอและเป็นเชิงเส้น ระหว่างการเปลี่ยนแปลงของอินพุทและเอาท์พุท สามารถแสดงความสัมพันธ์ได้ในรูปของบล็อกไดอะแกรม ดังรูปที่ 4.5



รูปที่ 4.5 การควบคุมแบบสัดส่วน

ความสัมพันธ์ระหว่างสัญญาณควบคุม(หรือเอาท์พุทของตัวควบคุม) m(t) กับสัญญาณค่าความคาดเคลื่อน e(t) คือ

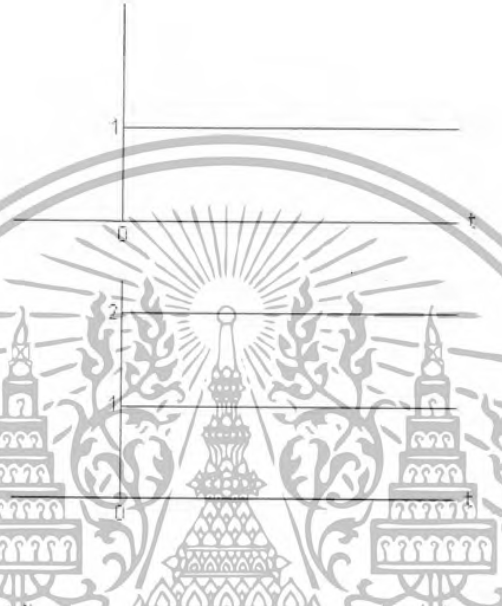
$$m(t) = K_p \times e(t)$$

หรือ
$$\frac{M(s)}{E(s)} = K_p$$

$$PB = \left(\frac{1}{K_p}\right) \times 100\%$$

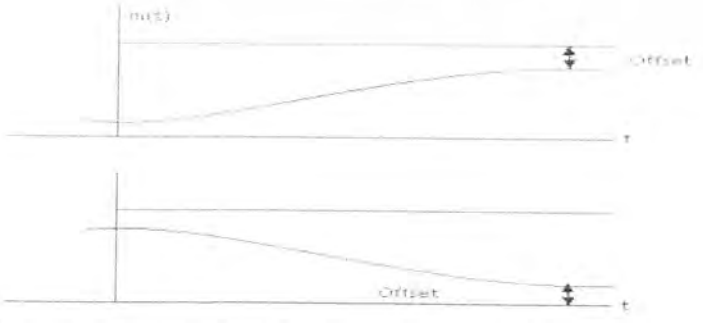
โดยที่ K_p จะอยู่ในเทอมของ Proportional Sensitivity หรือเกน (Gain)

Proportional Band (PB) เป็นการเปลี่ยนแปลงของอินพุทเพื่อที่จะทำให้เกิดการเปลี่ยนแปลงของเอาท์พุทมากที่สุดในการควบคุมแบบสัดส่วน ดังรูปที่ 4.6



รูปที่ 4.6ผลตอบสนองของการควบคุมแบบสัดส่วน

การเกิด Offset เป็นคุณลักษณะของระบบควบคุมแบบสัดส่วน ทั้งนี้เนื่องจากการทำงานของระบบควบคุมแบบสัดส่วนนั้น ไม่สามารถควบคุมระบบที่มีโหลด (Load) เปลี่ยนแปลงได้ดีเท่าที่ควร และในกรณีที่โหลดคงที่แต่เปลี่ยนค่าของระดับค่าเป้าหมาย (Set point) ที่ควบคุมไปก็จะเกิด Offset ขึ้นเช่นเดียวกัน โดยที่ Offset ก็คือค่าความแตกต่างของอินพุทและเอาท์พุทที่สภาวะคงที่เมื่อเป้าหมายคงที่นั่นเอง ดังรูปที่ 4.7



รูปที่ 4.7 แสดงความสัมพันธ์ระหว่างตัวแปรควบคุมกับอัตราการเปิดวาล์ว

เราสามารถลดค่า Offset ได้โดย

1. เพิ่มอัตราการขยายแบบสัดส่วน
2. เพิ่มค่าสัญญาณจัดการที่สภาวะเริ่มต้น (m_0)

$$M_1 = (K_p \times e) + m_0$$

3. เปลี่ยนค่าเป้าหมาย

4.2.3 การควบคุมแบบอินทิกรัล (Integral Control Action)

เป็นการควบคุมซึ่งค่าเอาท์พุทเป็นสัดส่วนโดยตรงกับค่าอินทิกรัลเชิงเวลาของอินพุท โดยจะมีความสัมพันธ์ระหว่างเอาท์พุทของตัวควบคุม m(t) และค่าความคาดเคลื่อน e(t) ดังนี้

$$m(t) = K_I \times \int_0^t e(t)dt$$

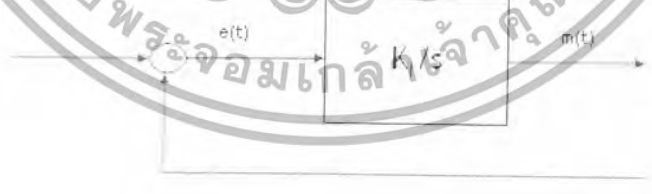
หรือ
$$\frac{dm(t)}{dt} = K_I \times e(t)$$

โดยที่ K_I เป็นค่าคงที่ที่สามารถปรับค่าได้

ทรานสฟอร์มฟังก์ชัน(Transfer Function) ของตัวแปรควบคุมแบบอินทิกรัลคือ

$$\frac{M(s)}{E(s)} = \frac{K_I}{s}$$

โดยสามารถแสดงในรูปของบล็อกไดอะแกรมได้ดังรูปที่ 4.8

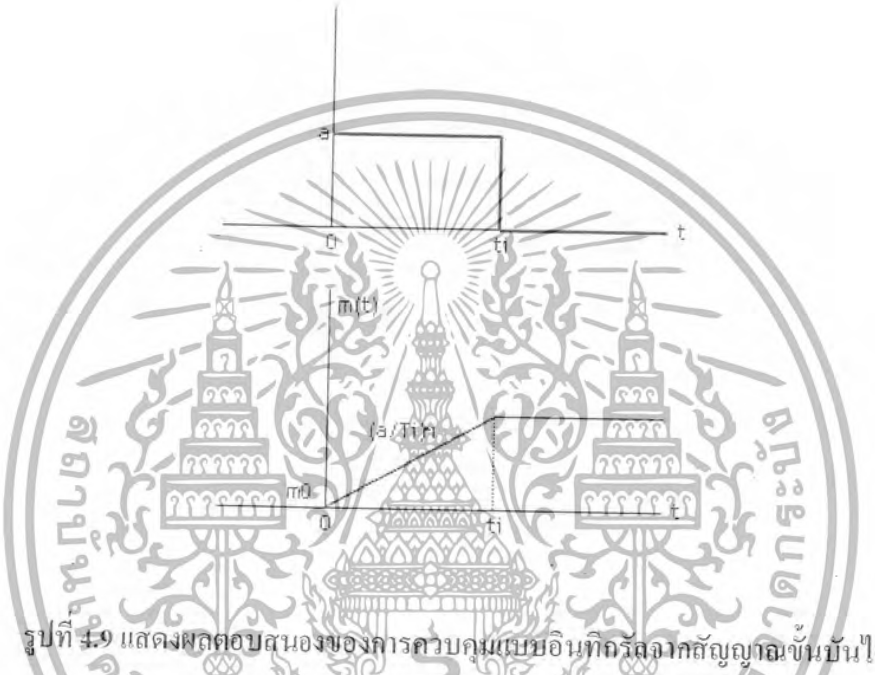


รูปที่ 4.8 บล็อกไดอะแกรมของการควบคุมแบบ Integral

ในการควบคุมแบบอินทิกรัลนั้นค่าเอาท์พุทของตัวควบคุม m(t) จะเปลี่ยนแปลงตามค่าความผิดพลาด e(t) ดังนั้น ถ้าความผิดพลาดซึ่งได้เกิดขึ้น ทำให้ระบบได้ค่าที่ผิดไป จากค่าที่ต้องการแล้ว อุปกรณ์ควบคุมจะจัดการกับค่าความผิดพลาดโดยเร็ว (โดยลดค่าความผิดพลาดนี้หมดไป) เมื่อตัวแปร

ควบคุมอยู่ที่ค่าเป้าหมายแล้วอุปกรณ์ควบคุมสุดท้ายจะยังไม่ทำงาน ซึ่งแสดงให้เห็นว่า ระบบอยู่ในสภาวะคงที่แล้วนั่นเอง

ดังนั้นในการควบคุมแบบอินทิกรัลจะไม่ทำให้เกิดค่า Offset ขึ้นมา ดังรูปที่ 4.9



รูปที่ 4.9 แสดงผลตอบสนองของการควบคุมแบบอินทิกรัลจากสัญญาณขั้นบันได

4.2.4 การควบคุมแบบเดริเวทีฟ (Derivative Control Action)

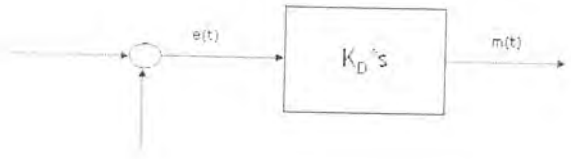
เป็นการควบคุมที่ค่าเอาท์พุทเป็นสัดส่วนกับอัตราการเปลี่ยนแปลงของอินพุท โดยมีความสัมพันธ์ดังนี้

$$m(t) = K_D \times \frac{de(t)}{dt}$$

หรือ $$\frac{M(s)}{E(s)} = K_D S$$

โดยที่ K_D เป็นค่าคงที่ที่สามารถปรับค่าได้

บล็อกไดอะแกรมของตัวควบคุมแบบเดริเวทีฟแสดงได้ดังรูปที่ 4.10



รูปที่ 4.10 บล็อกไดอะแกรมของตัวควบคุมแบบเดริเวทีฟ

รูปที่ 4.11 แสดงผลตอบสนองแบบขั้นบันไดของกิริยาการควบคุมแบบเดริเวทีฟ

เมื่อเพิ่มการควบคุมแบบเดริเวทีฟ ไปในเครื่องควบคุมจะเป็นการบวกมุมนำ(Phase lead) ในเครื่องควบคุมเพื่อชดเชยมุมตาม (Phase lag) ในทรานสเฟอร์ฟังก์ชัน (Loop Transfer Function) ซึ่งขบวนการส่วนใหญ่จะมีค่ามุมเป็นแบบมุมตาม

4.2.5 การควบคุมแบบพีไอ (Proportional and Integral Control Active)

เป็นการควบคุมที่ค่าเอาท์พุท เป็นสัดส่วนแบบเชิงเส้นกับผลรวมของค่าอินพุทและค่าอินทิกรัลเชิงเวลาของอินพุท โดยสามารถแสดงความสัมพันธ์ได้ดังสมการต่อไปนี้

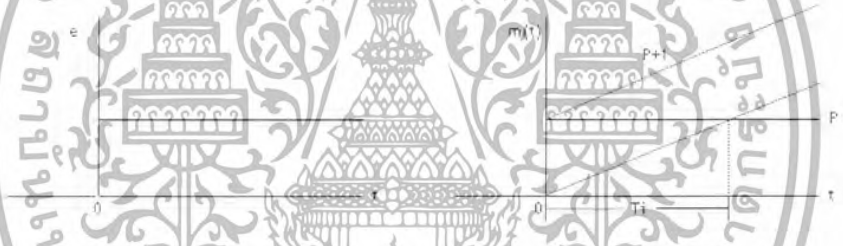
$$m(t) = (K_p \times e(t)) + \left(\frac{K_p}{T_i}\right)\int_0^t e(t)dt$$

หรือ $$\frac{M(s)}{E(s)} = K_p \times \left(1 + \frac{1}{Ts}\right)$$

โดยที่ K_p เป็นค่าของ Proportional Sensitivity หรือเกน (gain)

T_i เป็นค่าของ Integral Time

ดังรูปที่ 4.12



รูปที่ 4.12 การควบคุมแบบพีไอ

ข้อได้เปรียบของการรวมชุดอินทิกรัลและชุดควบคุมแบบสัดส่วน คือชุดอินทิกรัลจะกำจัด Offset ของชุดควบคุมแบบสัดส่วนให้หมดไป

4.2.6 การควบคุมแบบพีดี (Proportional and Derivative Control Action)

เป็นการควบคุมซึ่งค่าเอาท์พุทเป็นสัดส่วนโดยตรงกับผลรวมของค่าอินพุทกับผลคูณคาบเวลากับอัตราการเปลี่ยนแปลงอินพุท โดยสามารถแสดงด้วยสมการดังต่อไปนี้

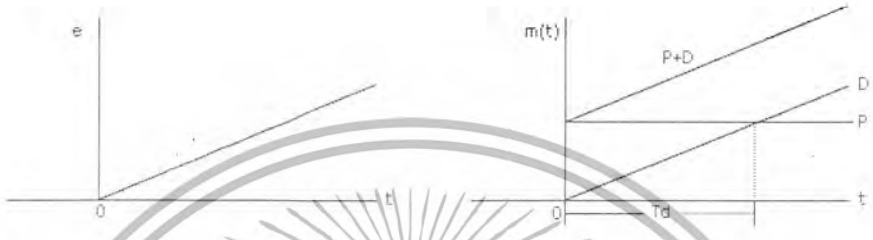
$$m(t) = (K_p \times e(t)) + \left(K_p \times T_D \times \frac{de(t)}{dt}\right)$$

หรือ $$\frac{M(s)}{E(s)} = K_p(1 + T_D S)$$

โดยที่ K_p เป็นค่าของ Proportional Sensitivity หรือเกน (gain)

T_D เป็นค่าของ Derivative time

ดังรูปที่ 4.13



รูปที่ 4.13 การควบคุมแบบพีดี

ข้อได้เปรียบในการควบคุมแบบพีดีนี้คือ เมื่อสัญญาณเข้าเป็นแบบเชิงเส้น (Ramp) จะมีผลตอบสนองทางเวลาได้เปรียบกว่าการควบคุมแบบสัดส่วนอย่างเดียว

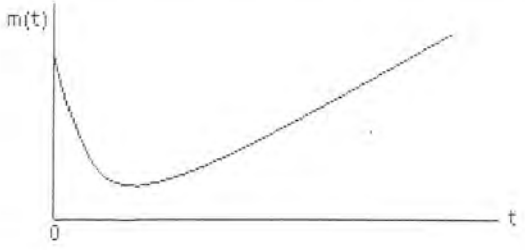
4.2.7 การควบคุมแบบพีไอดี (Proportional Integral and Derivative Control Active)

เป็นการควบคุมที่ค่าเอาท์พุทเป็นสัดส่วนกับค่าอินทิกรัลเชิงเวลาของอินพุทและค่าผลคูณของเวลากับอัตราการเปลี่ยนแปลงของอินพุต โดยสามารถแสดงได้ตามสมการ ต่อไปนี้

$$m(t) = (K_p \times e(t)) + \left(K_p \times T_D \times \frac{de(t)}{dt} \right) + \left(\frac{K_p}{T_i} \right) \int_0^t e(t)dt$$

หรือ

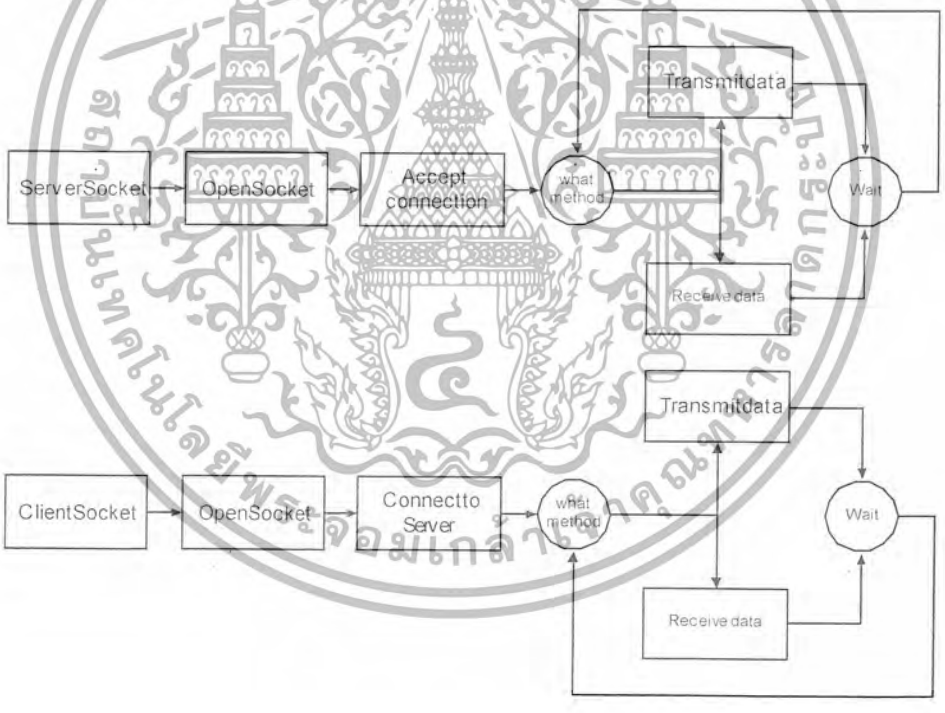
$$\frac{M(s)}{E(s)} = K_p \times \left(1 + T_D S + \frac{1}{T_i S} \right)$$



รูปที่ 4.14 ผลตอบสนองการควบคุมแบบพีไอดีกับสัญญาณเข้าแบบขั้นบันได

บทที่ 5
หลักการออกแบบ

การแลกเปลี่ยนข้อมูลถึงกันระหว่างคอมพิวเตอร์โดยใช้สาย USB TO USB Network Cable ในการเชื่อมต่อและส่งข้อมูลถึงกัน โดยผ่านโปรโตคอล TCP/IP (Transmission Control Protocol / Internet Protocol) เป็นการติดต่อกับระบบสื่อสารบนชั้น Application Layer ระบบจะทำหน้าที่ดูแลและควบคุมการส่งข้อมูลในระดับเลเยอร์อื่น ๆ ให้เองอัตโนมัติ จึงทำให้สะดวกในการพัฒนา โดยมีแนวความคิดในการทำงานของระบบโดยรวมดังรูปที่ 5.1

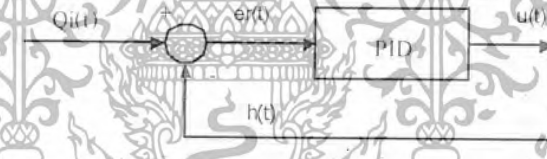


รูปที่ 5.1 แสดงบล็อกไดอะแกรมแสดงการทำงานของระบบ

5.1 ส่วนประกอบของโครงงานนี้แบ่งออกเป็นส่วนต่าง ๆ ดังนี้

1. สายที่ใช้ในการเชื่อมต่อระหว่างคอมพิวเตอร์จะใช้สายแบบ USB To USB Network Cable เพื่อเชื่อมต่อและส่งข้อมูลระหว่างคอมพิวเตอร์ โดยมีคุณสมบัติสามารถทำงานบนโปรโตคอล TCP/IP
2. โปรแกรมจำลองระดับน้ำ โดยใช้โปรแกรม Delphi ในการพัฒนาโดยแบ่งการทำงานออกเป็นสองส่วนได้แก่ โปรแกรมเซิร์ฟเวอร์ (Server) และโปรแกรมไคลเอน (Client) โดยโปรแกรมเซิร์ฟเวอร์จะจำลองการทำงานเป็นระบบควบคุมเพื่อไปควบคุมโปรแกรมไคลเอน ส่วนโปรแกรมไคลเอน จะจำลองเป็นระดับน้ำและมีการส่งค่ากลับมาเพื่อตรวจสอบความถูกต้องที่โปรแกรมตัวเซิร์ฟเวอร์
3. คอมพิวเตอร์จำนวน 2 เครื่อง เพื่อแบ่งการทำงานออกเป็นสองส่วน โดยโปรแกรมแม่จะติดตั้งที่คอมพิวเตอร์เครื่องที่หนึ่ง และโปรแกรมลูกจะถูกติดตั้งที่คอมพิวเตอร์เครื่องที่สองโดยมีการเชื่อมต่อด้วยสาย USB To USB Network Cable ผ่านทางพอร์ต USB

5.2 การทำงานของโปรแกรมเซิร์ฟเวอร์



รูปที่ 5.2 บล็อกไดอะแกรมโปรแกรมแม่

จากบล็อกไดอะแกรม การทำงานของโปรแกรมเซิร์ฟเวอร์ดังรูปที่ 5.2 จะเริ่มต้นการทำงานด้วยการรับค่าอินพุท (Qi(t)) จากคีย์บอร์ด หลังจากนั้นจะนำค่าที่ได้มาหาค่า Error จากสมาการที่ 5.1

$$er(t) = Qi(t) - h(t) \qquad \text{สมการที่ 5.1}$$

โดยที่ Qi(t) คือค่าที่ที่รับจากคีย์บอร์ด

h(t) คือค่าที่รับจากโปรแกรมเครื่องไคลเอน

หลังจากได้ค่า Error แล้วนำค่าที่ได้ไปเข้าสมการ PID เพื่อหาค่า u(t) เพื่อทำการส่งไปควบคุมระดับน้ำที่โปรแกรมไคลเอน โดยสมการ PID จะหาได้จากสมการที่ 5.2

$$\text{PID}: \quad \frac{u(s)}{er(s)} = \text{Kp}\left(1+\frac{Ti}{s}+\text{Td}\,s\right) \qquad \text{สมการที่ 5.2}$$

เมื่อทำการแปลงลาปาสทรานฟอร์มเพื่อให้อยู่ในรูปของ Time domain

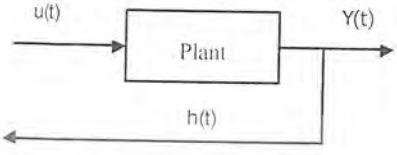
แล้วจะได้ดังนี้ $u(t) = Kp.e(t) + \frac{Kp}{Ti}[\int_{t0}^{t1} e(t)dt] + Td.Kp\frac{de(t)}{dt}$

$$u(t) = Kp.e(t) + [Ui(k-1) + Ki.\frac{Ts}{2}(e(k)+e(k+1))] + Kd(\frac{e(k)-e(k-1)}{Ts})$$

หลังจากนั้นก็ให้ส่งค่า u(t) ไปให้โปรแกรมเครื่องไคลเอนคำนวณ เพื่อจำลองหาระดับน้ำและมีการส่งค่ากลับมาตรวจสอบความถูกต้องที่โปรแกรมเซิร์ฟเวอร์ภายหลัง โปรแกรมเซิร์ฟเวอร์จะรอรับการส่งข้อมูลกลับจากโปรแกรมไคลเอน เพื่อนำไปแสดงผลและนำค่าไปตรวจสอบและเปรียบเทียบกับค่าอินพุท ดังแสดงใน Flow chart การทำงานของโปรแกรมเซิร์ฟเวอร์ในรูปที่ 5.3

รูปที่ 5.3 Flow Chart การทำงานของโปรแกรมเซิร์ฟเวอร์

5.3 การทำงานของโปรแกรมไคลเอน



รูปที่ 5.4 บล็อกไดอะแกรมของโปรแกรมไคลเอน

จากบล็อกไดอะแกรมของโปรแกรมไคลเอน ดังรูปที่ 3.4 จะเริ่มต้นการทำงานด้วยการรอรับค่า u(t) จากโปรแกรมเซิร์ฟเวอร์หลังจากนั้นจะเข้ากระบวนการที่เป็น First Order เพื่อหาค่าเอ้าพุทออกมา โดยสมการเอ้าพุทจะหาได้จากสมการที่ 3.3

$$y(s) = u(s)\frac{1}{s+1} \qquad \text{สมการที่ 5.3}$$

เมื่อทำการแปลงลาปาสมารถมฟอร์มเพื่อให้อยู่ในรูปของ Time domain แล้วจะได้ดังนี้

$$\frac{dy(t)}{dt} + y(t) = u(t)$$

$$\frac{y(t) - y(t-1)}{Ts} + y(t) = u(t)$$

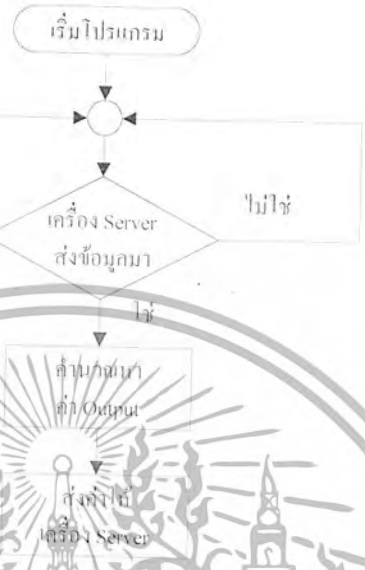
$$y(t) = \frac{Ts * u(t) - y(t-1)}{(Ts+1)}$$

เมื่อ u(t) คือค่าที่ได้รับจากโปรแกรมเซิร์ฟเวอร์

y(t) คือค่าเอ้าพุท และจะต้องทำการส่งกลับไปยังโปรแกรมเซิร์ฟเวอร์

Ts คือค่า period Time

หลังจากหาค่า y(t) ที่เป็นค่าเอ้าพุทได้แล้วจึงทำการส่งข้อมูลกลับไปยังโปรแกรมเซิร์ฟเวอร์ เพื่อทำนำไปแสดงผลออกสู่หน้าจอ และเปรียบเทียบกับค่าอินพุทเพื่อหาค่า Error แล้วทำการแก้ไขค่า y(t) ต่อไป การทำงานของโปรแกรมไคลเอนจะแสดงใน Flow chart การทำงานของโปรแกรมไคลเอน ในรูปที่ 5.5 ดังนี้



รูปที่ 5.5 Flow chart การทำงานของโปรแกรมไคลเอน

บทที่ 6
การทดลองและผลการทดลอง

เมื่อเราได้ทำการติดตั้งโปรแกรมเซิร์ฟเวอร์ (Server) และโปรแกรมไคลเอน (Client) ลงในคอมพิวเตอร์ทั้ง 2 เครื่องแล้ว ให้ทำการเชื่อมต่อคอมพิวเตอร์ทั้งสองด้วยสาย USB TO USB Network Cable ผ่านทางพอร์ต USB และทำการติดตั้งไดร์เวอร์ของสาย USB TO USB Network Cable แล้วจึงเริ่มทำการทดลองดังนี้

6.1 การเชื่อมต่อระหว่างโปรแกรมเซิร์ฟเวอร์และโปรแกรมไคลเอน

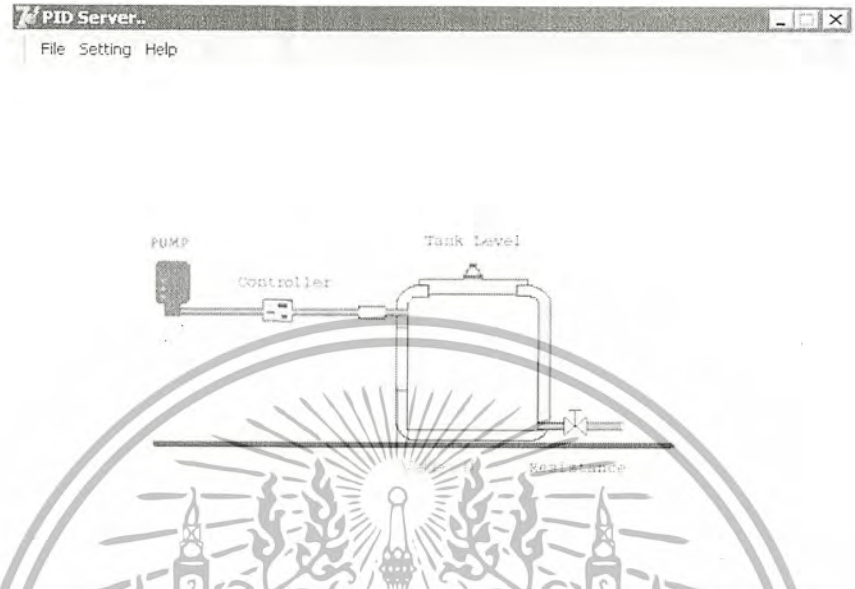
การเชื่อมต่อระหว่างคอมพิวเตอร์ทั้งสองผ่านทางสาย USB เพื่อให้คอมพิวเตอร์ทั้งสองเครื่องสามารถส่งข้อมูลถึงกันได้โดยการใช้งานโปรโตคอล TCP/IP โดยการระบุชื่อ หรือ IP Address ของเครื่องไคลเอน

การทดลอง

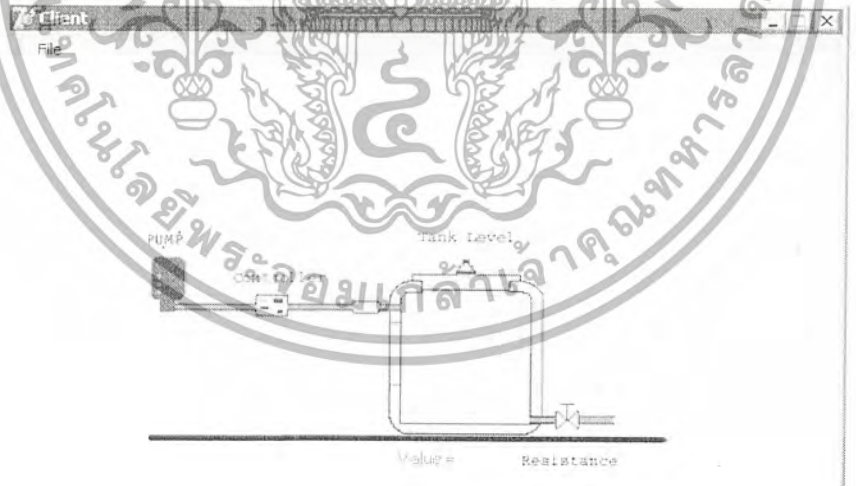
เปิดโปรแกรมเซิร์ฟเวอร์ (Server) ที่คอมพิวเตอร์เครื่องที่หนึ่งดังรูปที่ 6.1 และโปรแกรมไคลเอน (Client) ที่คอมพิวเตอร์เครื่องที่สองดังรูปที่ 6.2 หลังจากนั้นให้โปรแกรมเซิร์ฟเวอร์ทำการติดต่อกับโปรแกรมไคลเอน โดยการเลือกเมนู File แล้วเลือก Connect หรือใช้คีย์ลัด Ctrl+C หลังจากนั้นให้ทำการใส่ชื่อ หรือ IP Address ของเครื่องไคลเอนดังรูปที่ 6.3

ผลการทดลอง

สามารถทำการเชื่อมต่อระหว่างโปรแกรมเซิร์ฟเวอร์กับโปรแกรมไคลเอนได้สำเร็จ เมื่อการเชื่อมต่อสำเร็จจะเห็นว่าที่ไทเทิลบาร์ของโปรแกรมเซิร์ฟเวอร์ และโปรแกรมไคลเอนมีข้อความเปลี่ยนเป็นคำว่า "Connect Success" แสดงถึงการเชื่อมต่อสำเร็จนั่นเอง ดังแสดงในรูปที่ 6.4 ก และ 6.4 ข



รูปที่ 6.1 แสดงหน้าต่างโปรแกรมเซิร์ฟเวอร์ (Server)



รูปที่ 6.2 แสดงหน้าต่างโปรแกรมไคลเอน (Client)

รูปที่ 6.3 แสดงการใส่ชื่อ หรือ IP Address

รูปที่ 6.4 ก แสดงการเชื่อมต่อสำเร็จของเครื่องเซิร์ฟเวอร์(Server)

รูปที่ 6.4 ข แสดงการเชื่อมต่อสำเร็จของเครื่องไคลเอน (Client)

6.2 การรับและส่งข้อมูลไปยังโปรแกรมลูก

เมื่อการเชื่อมต่อสำเร็จโปรแกรมเซิร์ฟเวอร์จะทำงานทันที ทำให้มีการส่งค่าไปให้โปรแกรมไคลเอนเพื่อไปควบคุมระดับน้ำทางฝั่งไคลเอนทันที

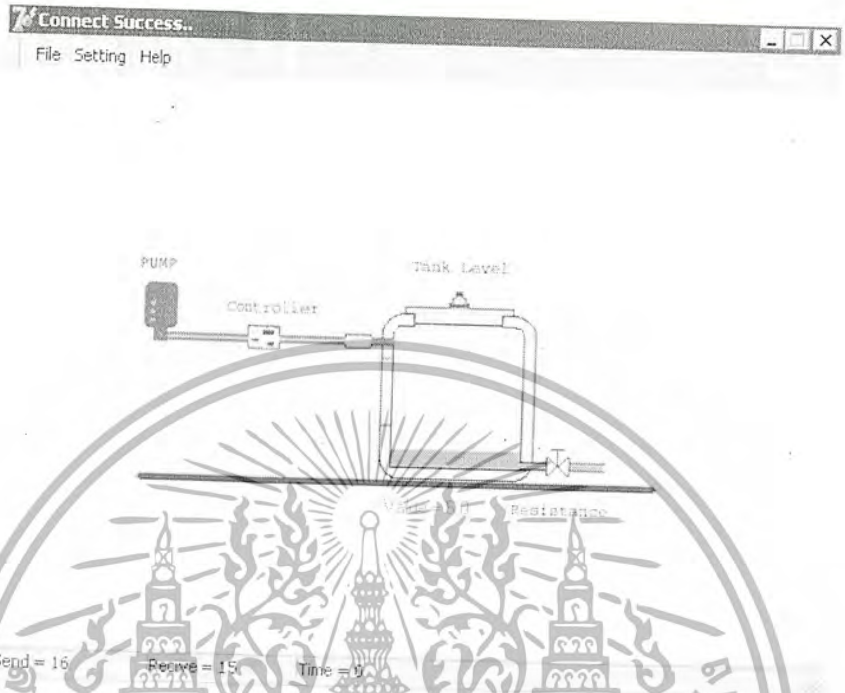
การทดลอง

เปลี่ยนค่าอินพุทโดยเลือกเมนู Setting แล้วเลือก Set Input\PID หรือกดเมนูลัด Ctrl+I ให้ทำการป้อนค่าที่ต้องการลงไปในช่องต่าง ๆ ดังแสดงในรูปที่ 6.5 หลังจากนั้นตอบตกลง คอมพิวเตอร์ทั้งสองจะมีการส่งข้อมูลถึงกัน และรอดูการแสดงผลที่หน้าจอคอมพิวเตอร์

รูปที่ 6.5 แสดงหน้าต่างในการใส่ค่าต่าง ๆ

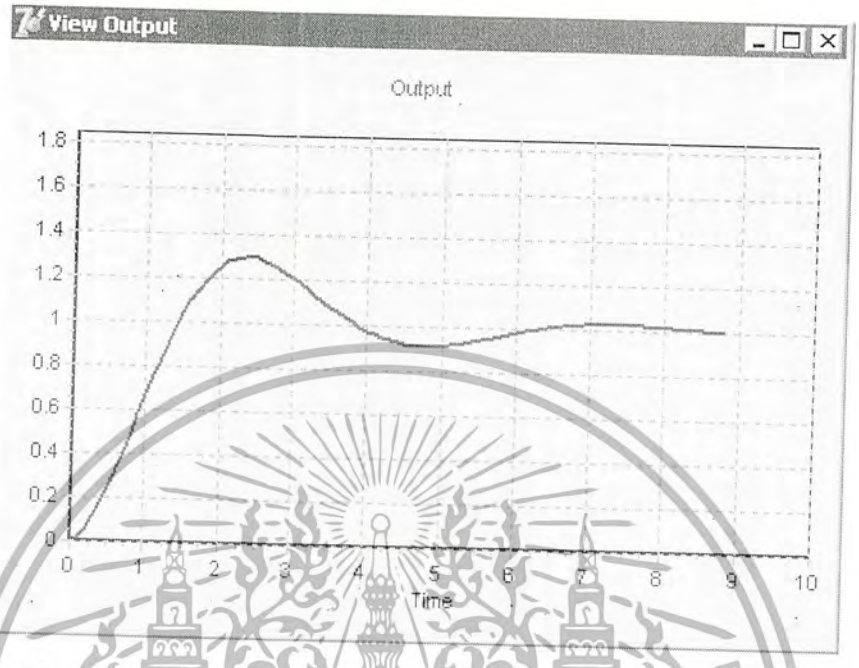
ผลการทดลอง

จากผลการทดลองเมื่อเปลี่ยนแปลงค่าอินพุท โปรแกรมจะทำการคำนวณและทำการส่งค่าไปยังโปรแกรมลูก ข้อมูลในการรับและส่งสามารถสังเกตจากทาสบาร์จะแสดงจำนวนไบต์ในการส่ง และการรับ ที่ใช้ในการส่งไปและการส่งกลับจากโปรแกรมไคลเอน ดังรูปที่ 6.6 ส่วนค่าระดับน้ำจะแสดงผลได้รูปถังน้ำ จะแสดงผลเหมือนกันทั้งโปรแกรมแม่และโปรแกรมลูก



รูปที่ 6.6 แสดงการแลกเปลี่ยนข้อมูลกับเครื่องไกเอน

เราสามารถดูการเปลี่ยนแปลงของระดับน้ำได้จากกราฟ โดยการเลือกเมนู Setting แล้วเลือก View Output หรือคีย์ลัด Ctrl+V จะแสดงหน้าต่างโดยที่แกน X คือ แกนเวลา ส่วนแกน Y คือ ค่าเอ้าพุท ของระบบ ดังแสดงในรูปที่ 6.7

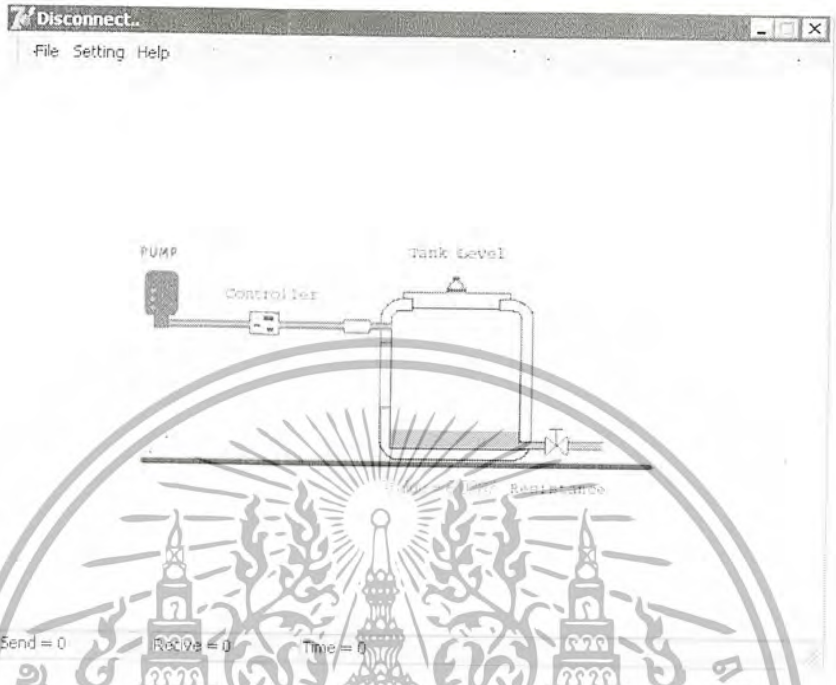


รูปที่ 6.7 แสดงหน้าต่างการแสดงกราฟ

6.3 การยกเลิกการติดต่อรับส่งข้อมูล

การทดลอง

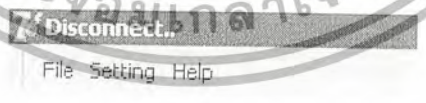
เมื่อต้องการยกเลิกการติดการรับส่งข้อมูลระหว่างคอมพิวเตอร์ สามารถทำได้โดยการเลือกเมนู File แล้วเลือก Disconnect หรือใช้คีย์ลัดคือ Crtl+D จะทำให้การเชื่อมต่อระหว่างคอมพิวเตอร์ทั้งสองหยุดลง การรับส่งข้อมูลก็จะหยุดลงด้วยดังแสดงในรูปที่ 6.8



รูปที่ 6.8 แสดงการยกเลิกการเชื่อมต่อ

ผลการทดลอง

เมื่อหยุดการเชื่อมต่อระหว่างคอมพิวเตอร์ทั้งสองเครื่องแล้ว จะทำให้การส่งข้อมูลหยุดลงและการแสดงผลหน้าจอจะหยุดลงด้วย การหยุดการติดต่อรับส่งข้อมูลจะสังเกตเห็นได้จากไทเทิลบาร์ จะแสดงข้อความว่า "Disconnect" ดังแสดงในรูปที่ 6.9



รูปที่ 6.9 แสดงข้อความการยกเลิกการเชื่อมต่อ

6.4 สรุปผลการทดลอง

จากการทดลอง การเชื่อมต่อเพื่อรับส่งข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง โดยใช้สาย USB Network Cable ในการเชื่อมต่อคอมพิวเตอร์ทั้งสองเครื่อง และเขียนโปรแกรมผ่านโปรโตคอล TCP/IP ในการแลกเปลี่ยนข้อมูลระหว่างกัน โดยการจำลองการควบคุมระดับน้ำ ซึ่งจากการทดลองสามารถเชื่อมต่อเพื่อทำการส่งข้อมูลได้เป็นผลสำเร็จ การส่งข้อมูลสามารถทำได้และสามารถเปลี่ยนแปลงค่าระดับน้ำหรือพารามิเตอร์ต่าง ๆ ของระบบควบคุมได้เป็นอย่างที่น่าพอใจ

เมื่อต้องการยกเลิกการติดต่อระหว่างคอมพิวเตอร์ทั้งสอง สามารถเลือกเมนูการยกเลิกการเชื่อมต่อ เพื่อให้คอมพิวเตอร์ทั้งสองหยุดการแลกเปลี่ยนข้อมูลระหว่างกัน ซึ่งจากการทดลองการยกเลิกการเชื่อมต่อสามารถทำงานได้อยู่ในเกณฑ์ที่น่าพอใจ

บทที่7
บทวิจารณ์และสรุป

7.1 สรุปผลการทดลอง

จากการทดลองส่งข้อมูลระหว่างคอมพิวเตอร์ โดยการจำลองการควบคุมระดับน้ำแบบ PID และแสดงผลออกสู่หน้าจอคอมพิวเตอร์ รวมถึงการแสดงกราฟ เพื่อแสดงสถานะการเปลี่ยนแปลงของระบบ พบว่าสามารถส่งข้อมูลระหว่างคอมพิวเตอร์ได้ดี และสามารถจำลองการควบคุมระดับน้ำได้เป็นที่น่าพอใจ

7.2 วิจารณ์การทดลอง

การทำโครงการนี้จำเป็นจะต้องทำการศึกษาความรู้พื้นฐานต่าง ๆ หลายแขนง ทั้งทางส่วนของการส่งข้อมูลโดยใช้โปรโตคอล TCP/IP การเชื่อมต่อผ่านพอร์ต USB การเขียนโปรแกรม Delphi และการจำลองระดับน้ำโดยการใช้ประดับน้ำแบบPIDเพื่อแสดงการส่งข้อมูลถึงกันระหว่างคอมพิวเตอร์

7.3 ประโยชน์ที่ได้รับจากโครงการนี้

ในการทำโครงงานนี้ต้องอาศัยความรู้ในหลาย ๆ ด้าน ทั้งความรู้ในด้านการเขียนโปรแกรมเพื่อทำการส่งข้อมูลถึงกันระหว่างคอมพิวเตอร์ ความรู้ทางด้านทฤษฎีของระบบควบคุมมาใช้งาน และการศึกษาและใช้งานโปรโตคอล TCP/IP รวมถึงมาตรฐานและข้อกำหนดต่าง ๆ ของพอร์ตและสาย USB ประโยชน์อีกด้านของการทำงานนี้จะได้ฝึกการแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้น ทั้งปัญหาที่เกิดจากการเขียนโปรแกรม การหาข้อมูลต่าง ๆ และประโยชน์ทางด้านการประสานงานในการทำงานร่วมกัน ทำให้มีการฝึกการวางแผนอย่างมีระบบและคิดแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นร่วมกัน

7.4 ปัญหาที่พบในการทำโครงงานนี้

ปัญหาที่พบในการทำงานโครงงานนี้อย่างแรกคือ ขาดความรู้และประสบการณ์ในด้านการสื่อสารข้อมูลบนโปรโตคอล TCP/IP และมาตรฐานและข้อกำหนดการใช้งานพอร์ตและสาย USB ทำให้ต้องใช้เวลาในการศึกษาหาข้อมูลเกี่ยวกับเรื่องดังกล่าวไปพอสมควร

นอกจากนั้นยังมีปัญหาทางด้านการพัฒนาโปรแกรมเพื่อให้สามารถรับและส่งข้อมูลระหว่างคอมพิวเตอร์ได้ ต้องทำการทดสอบและแก้ไขหลายครั้งกว่าจะได้ผลลัพธ์ที่ต้องการ รวมถึงการเลือกใช้ภาษาที่จะนำมาพัฒนาโปรแกรม

7.5 แนวทางการพัฒนา

ในการทำโครงงานนี้มีข้อผิดพลาดที่ควรแก้ไขคือ การเชื่อมต่อเพื่อรับและส่งข้อมูลระหว่างคอมพิวเตอร์ จำเป็นต้องทราบชื่อ หรือ IP Address ของเครื่องที่ต้องการเชื่อมต่อด้วย ถ้าไม่ทราบว่าชื่อหรือ IP Address นั้นก็ไม่สามารถเชื่อมต่อเพื่อทำการส่งข้อมูลถึงกันได้ ควรทำการแก้ไขและพัฒนาโปรแกรมเพื่อค้นหาชื่อ หรือ IP Address ที่มีการเชื่อมต่ออยู่ในเครือข่ายแล้วแสดงให้ผู้ใช้ได้ทราบอัตโนมัติ เพื่อทราบว่าจะติดต่อกับชื่อใดหรือ IP Address ใด และการแสดงของโครงงานนี้ยังเป็นภาพ 2 มิติควรพัฒนาเป็นภาพ 3 มิติ เพื่อความสมจริงและสวยงามมากยิ่งขึ้น

ภาคผนวก

ภาคผนวก ก

ก-1 โปรแกรมหลักเครื่อง Server

```
unit UPIDserver;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,Dialogs, ActnList,
StdActns, ToolWin, ActnMan, ActnCtrls, ActnMenus,XPStyleActnCtrls,ScktComp, StdCtrls,
ExtCtrls, ImgList, jpeg, ComCtrls;

type
 TForm1 = class(TForm)
  ActionManager1: TActionManager;
  ActionMainMenuBar1: TActionMainMenuBar;
  FileExit1: TFileExit;
  ACon: TAction;
  ADis: TAction;
  Actinput: TAction;
  ActView: TAction;
  HelpContents1: THelpContents;
  Timer1: TTimer;
  ImageList1: TImageList;
  Image1: TImage;
  water: TImage;
```

```
    Label1: TLabel;
    StatusBar1: TStatusBar;
    procedure AConExecute(Sender: TObject);
    procedure HelpContents1Execute(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ADisExecute(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ActInputExecute(Sender: TObject);
    procedure ActViewExecute(Sender: TObject);
  private
    client:Tclientsocket;
    procedure onread(Sender: TObject; Socket: TCustomWinSocket);
    procedure onconnect(Sender: TObject; Socket: TCustomWinSocket);
    { Private declarations }
  public
    Qi,Kp,Ti,Td,ek,uk,T,Yt : Real;
    ei,ui, ed:real ;
    level_i:extended;level_j :integer;
    function Kain:real;
    function integreat:real;
    function Differ:real;
    procedure initail;
    procedure levelwater;
    { Public declarations }
  end;

var
```

```
  Form1: TForm1;

implementation

uses UAbout, Uinput, Uview;

{$R *.dfm}

procedure TForm1.ACOnExecute(Sender: TObject);
var ip:shortstring;
begin
 ip:=InputBox('Remote IP Server','IP Address or Computer name','127.0.0.1');
 client.Host := ip;
 client.Open ;
end;

procedure TForm1.HelpContents1Execute(Sender: TObject);
begin
 AboutBox := TAboutBox.Create(nil);
 AboutBox.ShowModal ;
 AboutBox.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 client:=Tclientsocket.Create(self);
 client.Port := 4000;
 client.OnRead :=onread;
```

```
  client.OnConnect := onconnect;
  ADis.Enabled :=false;
  initail ;
  if forminput=nil then
  forminput:=TForminput.Create(self);
end;

procedure TForm1.onconnect(Sender: TObject; Socket: TCustomWinSocket);
begin
  self.Caption := 'Connect Success..';
  Timer1.Enabled :=true ;
  ACon.Enabled :=false;  ADis.Enabled :=true;
  ActView.Enabled :=true;
  FormView:=TFormView.Create(self);
end;

procedure TForm1.ouread(Sender: TObject; Socket: TCustomWinSocket);
begin
  Timer1.Enabled :=false;
  Yt:=StrToFloat(socket.ReceiveText);
  StatusBar1.Panels[1].Text :='Receive = '+inttostr(length(floattostr(Yt)));
  Label1.Caption :='Value = '+ FormatFloat('##0.0###',Yt);
  Timer1.Enabled :=true;
end;

procedure TForm1.ADisExecute(Sender: TObject);
begin
  client.Close ;
```

```
    Timer1.Enabled :=false;
    self.Caption := 'Disconnect..';
    ACon.Enabled :=true;   ADis.Enabled :=false;
    ActView.Enabled :=false;
    initail ;
    with Forminput do
    begin
     Edit1.Text :='1';
     Edit2.Text :='0.1';
     Edit3.Text :='0.1';
     Edit4.Text :='0.1';
    end;
    with StatusBar1 do
    begin
     Panels[0].Text :='Send = 0';
     Panels[1].Text:='Recive = 0';
    end;
  end;

function TForm1.Kain: real;
begin
  result:=Kp*ek ;
end;

function TForm1.integreat: real;
var tmp:real;
begin
 tmp:=ui+((Kp/Ti)*(T/2)*(ek+ei));
```

```
 ei:=ek;
 result:=tmp;
end;

function TForm1.Differ: real;
var tmp:real;
begin
 tmp:=(Td*Kp)*((ek-ed)/T);
 ed:=ek;
 result:=tmp;
end;

procedure TForm1.initail;
begin
 Qi:=1; Kp:=0.1; Td:=0.1; Ti:=0.1;
 ek:=0; ei:=0; uk:=0; ui:=0; ed:=0; T:=0.1; Yt:=0;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var P,I,D:real;
begin
   ek:=Qi-Yt ;
   P:=Kain ;
   I:=integreat ;
   D:=Differ ;
   uk:=P+I+D;
   ui:=uk;
   StatusBar1.Panels[0].Text :='Send = '+inttostr(length(floattostr(uk)));
```

```
    client.Socket.SendText(FloatToStr(uk));
    Timer1.Enabled := false;
    FormView.drawchart ;
    levelwater ;

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  ADisExecute(Sender);
end;

procedure TForm1.ActInputExecute(Sender: TObject);
begin
 Forminput.ShowModal ;
 if forminput.ModalResult = mrOK then
 begin
  Qi:=forminput.tmpqi ;
  Kp:=Forminput.tmpkp ;
  Ti:=Forminput.tmpti ;
  Td:=Forminput.tmptd ;
 end;
end;

procedure TForm1.ActViewExecute(Sender: TObject);
begin
 formview.ShowModal ;
end;
```

```
procedure TForm1.levelwater;
begin
  level_i:=int(Yt );
  level_j:=trunc(level_i*2);
  level_i:=Yt  - level_i;
  if (level_i >=0.5) then
  level_j :=level_j+1;
  if level_j<=0 then
  level_j:=0;
  if level_j>=85 then
  level_j:=85;
  water.Height:=level_j;
  water.Top :=271-level_j;
end;
end.
```

ก-2 โปรแกรมย่อยการรับค่าอินพุท

```
unit Uinput;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls,
ExtCtrls, Buttons, ComCtrls;
```

```
type
  TForminput = class(TForm)
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    Edit1: TEdit;
    UpDown1: TUpDown;
    Label1: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure BitBtn2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    { Private declarations }
  public
    tmpqi,tmpkp,tmpti,tmptd:real; { Public declarations }
  end;

var
  Forminput: TForminput;
```

```
implementation


{$R *.dfm}

procedure TForminput.BitBtn2Click(Sender: TObject);
begin
 self.Close ;
end;

procedure TForminput.FormActivate(Sender: TObject);
begin
 tmpqi := StrToFloat(edit1.Text ) ;
 tmpkp := StrToFloat(edit2.Text ) ;
 tmpti := StrToFloat(edit3.Text ) ;
 tmptd := StrToFloat(edit4.Text ) ;
end;

procedure TForminput.BitBtn1Click(Sender: TObject);
begin
 tmpqi := StrToFloat(edit1.Text );
 tmpkp := StrToFloat(edit2.Text );
 tmpTi := StrToFloat(edit3.Text );
 tmpTd := StrToFloat(edit4.Text );
end;

procedure TForminput.FormCloseQuery(Sender: TObject;
var CanClose: Boolean);
```

```
 begin
  edit1.text:= FloatToStr(tmpqi) ;
  edit2.text:= FloatToStr(tmpkp);
  edit3.text:= FloatToStr(tmpti) ;
  edit4.text:= FloatToStr(tmptd) ;
 end;
 end.
```

ก-3 โปรแกรมย่อยแสดงกราฟ

```
unit Uview;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,Dialogs, TeEngine,
Series, ExtCtrls, TeeProcs, Chart, StdCtrls;

type
 TFormView = class(TForm)
  Chart1: TChart;
  Series1: TLineSeries;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  T,tmpYt:real; i:integer;
```

```
  procedure drawchart;
  { Public declarations }
end;

var
 FormView: TFormView;

implementation

uses UPIDserver;

{$R *.dfm}

procedure TFormView.drawchart;
begin
tmpYt := form1.Yt;
 if tmpYt >= chart1.LeftAxis.Maximum then
 begin
  chart1.LeftAxis.Maximum := tmpYt+(tmpYt/2);
  chart1.LeftAxis.AdjustMaxMin;
 end;
 if (chart1.LeftAxis.Maximum-tmpYt)>10 then
 begin
  chart1.LeftAxis.Maximum :=tmpYt+2;
  chart1.LeftAxis.AdjustMaxMin;
 end;

 if i > 100 then
```

```
  begin
    chart1.BottomAxis.SetMinMax((T-1),(T+10));
    chart1.BottomAxis.AdjustMaxMin;
    i:=0;
  end;
  series1.AddXY(T,tmpYt,'',clHotlight);
  T:=T+0.1;
  i:=i+1;
end;

procedure TFormView.FormCreate(Sender: TObject);
begin
  With chart1.BottomAxis do
  begin
    AutomaticMaximum := false; AutomaticMinimum := false;
    Grid.Visible := true;
    SetMinMax(0,10);
  end;
  with chart1.LeftAxis do
  begin
    Automatic := false;  Grid.Visible := true;
    AutomaticMaximum :=false; AutomaticMinimum := false;
    SetMinMax(0,1.2);   Increment := 0.1;
  end;
  T:=0;  i:=0;
end;
end.
```

ก-4 โปรแกรมย่อยแสดงผู้จัดทำ

```
unit UAbout;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, jpeg;

type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
    Label1: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;
```

```
implementation

{$R *.dfm}
end.
```

ก-5 โปรแกรมหลักเครื่อง Client

```
unit UPIDclient;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,Dialogs, ActnList,
StdActns, ToolWin, ActnMan, ActnCtrls, ActnMenus,XPStyleActnCtrls,SckComp, StdCtrls,
Buttons, ExtCtrls, jpeg;
type
  TForm1 = class(TForm)
    ActionManager1: TActionManager;
    ActionMainMenuBar1: TActionMainMenuBar;
    FileExit1: TFileExit;
    Image1: TImage;
    water: TImage;
    Label1: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    server:Tserversocket;
    data:TCustomWinSocket;
    procedure onAccept(Sender: TObject; Socket: TCustomWinSocket);
```

```
    procedure onclientconnect(Sender: TObject; Socket: TCustomWinSocket);
    procedure onread(Sender: TObject; Socket: TCustomWinSocket);
    procedure ondiscon(Sender: TObject; Socket: TCustomWinSocket);
  public
    T,uk,Yt,Yt1 : Real;
    tmpstr:shortstring;
    level_i:extended; level_j:integer;
    procedure initail;
    procedure levelwater;
    function Output:real;
  end;

var
  Form1: TForm1;

implementation

uses Math;

{$R *.dfm}

function TForm1.Output: real;
begin
   Output :=((Ts*us)+yt1)/(1+Ts);
end;

procedure TForm1.initail;
begin
```

```
  uk:=0; T:=0.5; Yt:=0; Yt1:=0;
  FillChar(tmpstr.sizeof(shortstring),0);
 end;

 procedure TForm1.onAccept(Sender: TObject; Socket: TCustomWinSocket);
 begin
  data:=socket;
 end;

 procedure TForm1.onclientconneet(Sender: TObject;
  Socket: TCustomWinSocket);
 begin
  self.Caption := 'Connect Sucess';
 end;

 procedure TForm1.onread(Sender: TObject; Socket: TCustomWinSocket);
 begin
  tmpstr:=socket.ReceiveText ;
  uk:=strtofloat(tmpstr);
  tmpstr:='' ;
  Yt:=Output ;
  tmpstr:=FloatToStr(Yt);
  data.SendText(tmpstr);
  levelwater :
  Label1.Caption := 'Value = '+FormatFloat('##0.0###',Yt);
end;

procedure TForm1.FormCreate(Sender: TObject);
```

```
 begin
  server:=Tserversocket.Create(self);
  server.Port := 4000;
  server.OnAccept := onaccept;
  server.OnClientConnect  := onclientconnect;
  server.OnClientRead :=onread;
  server.OnClientDisconnect := ondiscon ;
  server.Open ;
  initail;
 end;

 procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
 begin
  If (server.Socket.Handle <> 0) then
  server.Close ;
  canclose:=true;
 end;

 procedure TForm1.ondiscon(Sender: TObject; Socket: TCustomWinSocket);
 begin
   initial;
   self.Caption := 'Disconnect..';
 end;

 procedure TForm1.levelwater;
 begin
  level_i:=int(Yt );
  level_j:=trunc(level_i*2);
```

```
    level_i:=Yt  - level_i;
    if (level_i >=0.5) then
    level_j :=level_j+1;
    if level_j<=0 then
    level_j:=0;
    if level_j>=85 then
    level_j:=85;
    water.Height :=level_j;
    water.Top := 271-level_j;
end;
end.
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ร่วมถึงโครงงาน สามารถดำเนินการได้สำเร็จลุล่วงไปด้วยดี ทั้งนี้เนื่องมาจากการได้รับความกรุณาเป็นอย่างสูงจาก ร.ศ. สุเธียร เกียรติสุนทร ซึ่งเป็นอาจารย์ที่ปรึกษา และคอยให้คำชี้แนะ เสนอแนวทางอันเป็นประโยชน์ในการจัดทำโครงงาน และทั้งนี้ต้องขอขอบพระคุณอาจารย์ทุกท่านที่ได้ประสิทธิประสาทวิชาความรู้ทั้งหลายแก่ผู้จัดทำ ซึ่งผู้จัดทำขอกราบขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้ด้วย

ผู้จัดทำขอขอบพระคุณทุก ๆ ท่าน ที่ให้การสนับสนุนและช่วยเหลือในการจัดทำปริญญานิพนธ์ชิ้นนี้ขึ้นมาได้อย่างสมบูรณ์

ผู้จัดทำ
นาย ชาติชาย ไชยกุมาร
นาย โสภณ ด่านแสวง

หนังสืออ้างอิง

1. ลภน สุภาพ, อรรถพล บุญยะโภคา, วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ต USB ขั้นพื้นฐาน" บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด
2. วรเทพ ไพบูลย์รัตนากร, บุญอนันต์ เกียงเอี่ย "สัมผัสโลก USB ด้วย Easy USB Module" บริษัท แอสทรอน ลอจิก รีเสร์ชแอนด์ดิวิลอปเมนต์ จำกัด
3. สุวัฒน์ ปุณณชัยยะ, ต้น ตันตาสุทธิวงศ์, สุพจน์ ปุณณชัยยะ "เปิดโลก TCP/IP และโปรโตคอลของอินเตอร์เน็ต" บริษัท โปรวิชั่น จำกัด
4. เรืองไกร รังศิพล "เจาะระบบ TCP/IP จุดอ่อนของโปรโตคอลและการป้องกัน" บริษัท โปรวิชั่น จำกัด
5. สัจจะ จรัสรุ่งรวีวาร, จตุรพงษ์ สุขประเสริฐ "เริ่มต้นอย่างมืออาชีพด้วย Delphi5" สำนักพิมพ์ อินโฟเพรส
6. ประยงค์ อู่ประสิทธิวงศ์ "การเขียนภาษา Object Pascal" บริษัท ซัคเซส มีเดีย จำกัด
7. ประพนธ์ อัศวภาณุวัฒน์ "Delphi Episode II เทคนิคและการพัฒนาโปรแกรมด้วยเดลไฟ" บริษัท ซีเอ็ดยูเคชั่น จำกัด