สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การศึกษาการส่งข้อมูลผ่านพอร์ตยูเอสบี

DATA TRANSFER VIA USB PORT



โดย

นายพรพจน์ ท่าอิฐ

นายเพิ่มสิทธิ์ ขามช่วง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปพ.
พ 246 ก
2545

เลขหมู่.....
เลขทะเบียน 50339
วัน,เดือน,ปี 29 เม.ย. 2547

.b.....
.i.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การศึกษาการส่งข้อมูลผ่านพอร์ตยูเอสบี

DATA TRANSFER VIA USB PORT

โดย

นายพรพจน์ ท่าอิฐ 42010218

นายเพิ่มสิทธิ์ ขามช่วง 42010246

อาจารย์ที่ปรึกษา

ผศ. พลผดุง ผดุงกุล

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อโครงงาน (ภาษาไทย) การศึกษาการส่งข้อมูลผ่านพอร์ตยูเอสบี

ชื่อโครงงาน (ภาษาอังกฤษ) DATA TRANSFER VIA USB PORT

นายพรพจน์ ท่าอิฐ 42010218

นายเพิ่มสิทธิ์ ขามช่วง 42010246

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(ผศ.พลผดุง ผดุงกุล)

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ ปีการศึกษา2545

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การศึกษาการส่งข้อมูลผ่านพอร์ตยูเอสบี

ผู้จัดทำ

1. นายพรพจน์ ท่าอิฐ 42010218
2. นายเพิ่มสิทธ์ ขามช่วง 42010246



..... อาจารย์ที่ปรึกษา

(ผศ.พลผดุง ผดุงกุล)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การศึกษาการส่งข้อมูลผ่านพอร์ตยูเอสบี

พรพจน์ ท่าอิฐ
เพิ่มสิทธิ์ ขามช่วง
ผศ.พลผดุง ผดุงกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

Universal Serial Bus (USB) เป็นรูปแบบหนึ่งของการสื่อสารแบบอนุกรมโดยถูกสร้างขึ้นมาเพื่อให้สามารถใช้งานได้เลยหลังการเชื่อมต่อโดยไม่ต้องเปิดเครื่องคอมพิวเตอร์ใหม่ และด้วยคุณสมบัติที่สามารถส่งข้อมูลโดยมีอัตราเร็วในการส่งข้อมูลสูงกว่าการสื่อสารแบบอนุกรมRS-232 มาก อีกทั้งลักษณะทางกายภาพที่มีขนาดเล็กและการใช้งานง่าย เหมาะสมกับเทคโนโลยีในปัจจุบัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Data transfer via USB port

Mr. Pornpoj Tha-it

Mr. Permsit Khamchuang

Advisor Assist. Prof. Polphadung Phadungkul

Educational Year 2002

ABSTRACT

Universal Serial Bus (USB) is a type of serial ports communication that is becomes a standard technology of communication between PC and peripheral devices. USB standard was invented in order that the users can connect and it will be detected immediately without restarting. Even though USB's operation is rather similar to RS-232 standard, but USB is able to send data much faster than RS-232 standard, moreover its physical size is proper and easy to use according to "PLUG&PLAY" purpose. Therefore the USB standard is very popular and well-known in nowadays

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

บทคัดย่อ (ภาษาไทย)	I
บทคัดย่อ (ภาษาอังกฤษ)	II
สารบัญ	III
สารบัญรูป	V
สารบัญตาราง	VII
บทที่ 1 อารัมภบท	1
1.1 กล่าวนำ	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์	1
1.3 รายละเอียดของปริญญานิพนธ์	1
บทที่ 2 ระบบยูเอสบี	2
2.1 คุณสมบัติของยูเอสบี	2
2.2 โครงสร้างของยูเอสบี	3
2.3 ซอฟต์แวร์ยูเอสบี	3
2.4 ฮาร์ดแวร์ยูเอสบี	4
2.5 การติดต่อระหว่างอุปกรณ์กับโฮส	5
2.6 ดิซคริปเตอร์ของอุปกรณ์	6
บทที่ 3 ชนิดของข้อมูลและการส่งผ่านบนระบบยูเอสบี	8
3.1 พื้นฐานข้อมูลแบบแพ็คเกจ	8
3.2 ชนิดของแพ็คเกจ	10
3.3 การประกอบแพ็คเกจเป็นทรานแซ็คชัน	13
3.4 คาต้าท็อกเกิล	18
3.5 เทคนิคการเข้ารหัสสัญญาณ NRZI	19
บทที่ 4 การคอนฟิกอุปกรณ์	21
4.1 ขั้นตอนการคอนฟิกอุปกรณ์	21
บทที่ 5 การออกแบบในส่วนของฮาร์ดแวร์	24
5.1 วงจรในส่วนติดต่อกับโฮส	24
5.1.1 คุณสมบัติของไอซี PDIUSBD11	24
5.2 วงจรในส่วนแสดงผล LED DOT MATRIX	26

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3 วงจรในส่วนของเมาส์ยูเอสบี 28
บทที่ 6 การออกแบบในส่วนของซอฟต์แวร์ 29
6.1 การเขียนโปรแกรมด้วยภาษา C เพื่อควบคุมอุปกรณ์ 29
6.1.1 I2C โปรโตคอลของไอซี PDIUSBD11 29
6.1.2 ฟังก์ชันต่างๆภายในไอซี PDIUSBD11 32
6.1.3 คอนโทรลโฟล์วของดีไวซ์ไดร์เวอร์ 41
6.1.4 ดีไวซ์ดิสคริปเตอร์ของเมาส์ยูเอสบี 45
6.2 การเขียนโปรแกรมส่วนติดต่อกับผู้ใช้งานด้วยภาษา Visual Basic 49
6.2.1 การเปิดภาพบิตแม็ป 51
บทที่ 7 ผลการทดลอง 52
7.1 ส่วนติดต่อกับผู้ใช้งานเขียนด้วยภาษา Visual Basic 52
7.2 ค่าของข้อมูลที่ส่งจากอุปกรณ์ไปยังโฮสต์ 54
บทที่ 8 บทสรุปและข้อเสนอแนะในการพัฒนา 55
8.1 บทสรุป 55
8.2 ข้อเสนอแนะในการพัฒนา 55
ภาคผนวก 56
ภาคผนวก ก ส่วนติดต่อกับผู้ใช้งาน 57
ภาคผนวก ข วงจรสมบูรณ์ของเมาส์ยูเอสบี 58
ภาคผนวก ค วงจรสมบูรณ์ของการควบคุมการแสดงผล LED DOT MATRIX 59
ภาคผนวก ง รูปแบบการเก็บข้อมูลของไฟล์ภาพบิตแม็ป 60
กิตติกรรมประกาศ 63
หนังสืออ้างอิง 64

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่ 2.5.1 การรับ-ส่งข้อมูลระหว่างโฮสต์ กับ อุปกรณ์ 5

รูปที่ 3.1.1 ระบบแพ็คเกจ 8

รูปที่ 3.2.1 แพ็กเกจเริ่มต้น 10

รูปที่ 3.2.2 แพ็คเกจขาเข้า ขาออก และ SETUP แพ็คเกจ 11

รูปที่ 3.2.3 แพ็คเกจข้อมูล 11

รูปที่ 3.2.4 แพ็คเกจตรวจสอบ 12

รูปที่ 3.3.1 การจัดแพ็คเกจข้อมูลแบบอินเตอร์รัพต์ 13

รูปที่ 3.3.2 การจัดแพ็คเกจข้อมูลแบบไอโซโครนัส 15

รูปที่ 3.3.3 เฟสของการจัดแพ็คเกจข้อมูลแบบคอนโทรล 16

รูปที่ 3.3.4 เฟส Setup ของการจัดแพ็คเกจข้อมูลแบบคอนโทรล 17

รูปที่ 3.3.5 เฟสการอ่านข้อมูลของการจัดแพ็คเกจข้อมูลแบบคอนโทรล 17

รูปที่ 3.3.6 เฟสการเขียนข้อมูลของการจัดแพ็คเกจข้อมูลแบบคอนโทรล 17

รูปที่ 3.3.7 เฟสแสดงสภาวะของการจัดแพ็คเกจข้อมูลแบบคอนโทรล 18

รูปที่ 3.5.1 การเข้ารหัสข้อมูลเป็น NRZI 20

รูปที่ 3.5.2 การเติมสตัฟบิต 20

รูปที่ 5.1.1 บล็อกไดอะแกรมของส่งผ่านข้อมูลระหว่างอุปกรณ์กับโฮสต์ 24

รูปที่ 5.1.2 วงจรในส่วนที่ติดต่อกับโฮสต์ 25

รูปที่ 5.2.1 วงจรในส่วนแสดงผล LED DOT MATRIX 26

รูปที่ 5.2.2 วงจรขับ LED 27

รูปที่ 5.2.3 ฟังก์ชันการทำงานและไทมิ่งไดอะแกรมของไอซี 74HC164N 27

รูปที่ 5.3.1 วงจรในส่วนของเมาส์ยูเอสบี 28

รูปที่ 6.1.1 รูปแบบของข้อมูลอนุกรมบนบัส I2C โดยใช้การอ้างถึงแบบ 7 บิต 29

รูปที่ 6.1.2 รูปแบบของข้อมูลอนุกรมบนบัส I2C ด้วยคำสั่งเขียนของ PDIUSBD11 30

รูปที่ 6.1.3 รูปแบบของข้อมูลอนุกรมบนบัส I2C ด้วยคำสั่งอ่านของ PDIUSBD11 31

รูปที่ 6.1.4 โฟล์วคอนโทรลของโปรแกรมหลัก 41

รูปที่ 6.1.5 โฟล์วคอนโทรลสำหรับโปรแกรมย่อย 42

รูปที่ 6.1.6 โฟล์วคอนโทรลสำหรับ StandardRequest และ ClassRequest 43

รูปที่ 6.1.7 โฟล์วคอนโทรลสำหรับสแกนการกดเมาส์ 44

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 6.2.1 โครงสร้างของโฮสต์ซอฟต์แวร์ 49
รูปที่ 6.2.2 โครงสร้างข้อมูลของภาพบิตแม็ป 51
รูปที่ 7.1.1 แสดงว่าพบโฮสต์คอนโทรลเลอร์ 52
รูปที่ 7.1.2 ไม่มีอุปกรณ์เชื่อมต่อที่โฮสต์ 52
รูปที่ 7.1.3 มีอุปกรณ์เชื่อมต่อโฮสต์ที่พอร์ต 1 53
รูปที่ 7.1.4 ดิซคริปเตอร์ของอุปกรณ์ที่เชื่อมต่อโฮสต์ที่พอร์ต 1 53
รูปที่ 7.1.5 ค่าของดิซคริปเตอร์ของอุปกรณ์ที่เชื่อมต่อโฮสต์ที่พอร์ต 1 54
รูปที่ 7.1.6 ค่าของข้อมูลที่ส่งจากอุปกรณ์ไปยังโฮสต์เพื่อควบคุมการเคลื่อนที่ของเมาส์ 54

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่ 3.1.1 ชนิดของแพ็คเกจที่ระบุโดยค่า PID 9
ตารางที่ 3.3.1 ชนิดของทรานแซ็คชัน 13
ตารางที่ 6.1.1 แอดเดรส I2C ของไอซี PDIUSBD11 30
ตารางที่ 6.1.2 เอ็นพอยต์ของไอซี PDIUSBD11 33
ตารางที่ 6.1.3 ความหมายของ No Lazy Clock และ Clock running 34
ตารางที่ 6.1.4 ความหมายของ Code Error 37

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
อารัมภบท

1.1 กล่าวนำ

นับตั้งแต่เครื่องคอมพิวเตอร์ส่วนบุคคลเข้ามามีบทบาทในชีวิตประจำวันมากขึ้น อุปกรณ์ต่างๆถูกนำมาต่อพ่วงมากขึ้น นั่นหมายถึงการใช้พลังงานก็มากขึ้นตามด้วย อีกทั้งอุปกรณ์ที่นำต่อพ่วงที่พอร์ตของคอมพิวเตอร์เช่น พอร์ตอนุกรม และพอร์ตขนานนั้นมีอยู่จำกัด และมีความเร็วไม่สูงนัก เพื่อลดปัญหาที่ได้กล่าวไปแล้วจึงได้เกิดมาตรฐานยูเอสบี (USB: Universal Serial Bus) เป็นการสื่อสารข้อมูลแบบอนุกรมอีกชนิดซึ่งสามารถต่ออุปกรณ์ได้ถึง 127 ชิ้นโดยผ่านทางฮับซึ่งเป็นตัวกระจายพลังงาน ซึ่งพอร์ตยูเอสบีมีจุดเด่นที่สุดคือ ความเร็วในการส่งข้อมูลที่ได้ถึง 12 เมกกะบิตต่อวินาทีใน USB1.1 และ 480 เมกกะบิตต่อวินาทีใน USB2.0

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

ในการทำปริญญานิพนธ์นี้เป็นการศึกษาการส่งผ่านข้อมูลผ่านพอร์ตยูเอสบี โดยจะทำการสร้างอุปกรณ์ขึ้นมา 2 ชิ้นคือ เมาส์ยูเอสบีเพื่อทดสอบการส่งข้อมูลมายังคอมพิวเตอร์ และการแสดงผล LED DOT MATRIX เพื่อทดสอบการส่งข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์

1.3 รายละเอียดของปริญญานิพนธ์

ในปริญญานิพนธ์นี้แบ่งเนื้อหาออกเป็น 8 บทโดยในแต่ละบทจะมีรายละเอียดดังต่อไปนี้

บทที่ 1 เป็นการอารัมภบทและกล่าวถึงวัตถุประสงค์ในการทำปริญญานิพนธ์

บทที่ 2 จะกล่าวถึงระบบยูเอสบีว่าถ้าจะสร้างอุปกรณ์ขึ้นมาชนิดหนึ่ง จะต้องทำอะไรบ้าง

บทที่ 3 เป็นการอธิบายถึงรูปแบบของข้อมูลที่สื่อสารบนบัสอนุกรมยูเอสบี

บทที่ 4 จะอธิบายขั้นตอนในการติดต่อกับโฮสเพื่อที่จะขอใช้งานพอร์ตยูเอสบี

บทที่ 5 เป็นการออกแบบอุปกรณ์ 2 ชนิดคือ เมาส์ และ LED Display

บทที่ 6 จะเป็นการออกแบบทางซอฟต์แวร์โดยแบ่งออกเป็น 2 ส่วนคือ ส่วนติดต่อกับผู้ใช้งานโดยใช้ภาษา Visual Basic และอีกส่วนหนึ่งคือส่วนควบคุมอุปกรณ์โดยใช้ภาษา C

บทที่ 7 ผลการทดลอง

บทที่ 8 เป็นบทสรุปการทำงานทั้งหมดและข้อเสนอแนะในการพัฒนาต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ระบบยูเอสบี

ยูเอสบี คือ ระบบบัสอนุกรมที่ถูกออกแบบมาเพื่อการเชื่อมต่อและใช้งานอุปกรณ์ต่อพ่วงทั้งหลายที่มีแอดเดรสต่างๆกัน โดยยูเอสบี มีความซับซ้อนของระบบการทำงานมาก แต่จะมีการทำงานที่เร็วมาก

2.1 คุณสมบัติของยูเอสบี

2.1.1 ใช้คอนเน็กเตอร์เพียงชนิดเดียว(มีสองรูปแบบ)ในการเชื่อมต่ออุปกรณ์ทุกชนิดเข้ากับคอมพิวเตอร์

2.1.2 สามารถเชื่อมต่ออุปกรณ์หลายๆชนิดรวมเข้าสู่บัสสัญญาณข้อมูลเดียวกัน

2.1.3 ไม่เกิดการขัดแย้งกันสำหรับการเข้าใช้ทรัพยากรของระบบ

2.1.4 สามารถตรวจสอบการเชื่อมต่อและการตั้งค่าการทำงานต่างๆโดยอัตโนมัติระหว่างที่คอมพิวเตอร์ทำงานอยู่ (Hot Attachment)

2.1.5 ความเร็วในการส่งข้อมูลที่สูงถึง 12 เมกกะบิตต่อวินาทีสำหรับอุปกรณ์ความเร็วสูง และ1.5 เมกกะบิตต่อวินาทีสำหรับอุปกรณ์ความเร็วต่ำ

2.1.6 มีไฟเลี้ยงจ่ายให้ในระบบทำให้อุปกรณ์ที่ใช้ไฟเลี้ยงไม่มากนัก ไม่จำเป็นต้องอาศัยแหล่งจ่ายไฟภายนอก

2.1.7 ค่าใช้จ่ายในการพัฒนาอุปกรณ์ต่ำเมื่อเทียบกับความเร็วที่ได้

2.1.8 ข้อมูลที่ส่งเป็นแบบสัญญาณผลต่างคู่ (Differential Pair Signaling) ซึ่งจะช่วยลดการรบกวนทางแม่เหล็กไฟฟ้า (EMI : Electromagnetic Interference)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 โครงสร้างของยูเอสบี

ยูเอสบี เป็นมาตรฐานการรับส่งข้อมูลที่มีรูปแบบการเชื่อมต่อในระบบบัสคืออุปกรณ์ทุกๆตัวจะต้องส่งสัญญาณรวมกันไปในสายสัญญาณเพียงคู่เดียวดังนั้นอุปกรณ์ทุกๆตัวจะต้องส่งสัญญาณข้อมูลต่อๆกันไปเพื่อไม่ให้เกิดการชนกัน และเนื่องจากยูเอสบี เป็นระบบบัสที่ใช้สายส่งสัญญาณเพียงคู่เดียว และสัญญาณเป็นแบบผลต่างทำให้ในช่วงเวลาหนึ่งๆจะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียวเท่านั้น ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันได้ (Half Duplex)

จังหวะในการรับส่งข้อมูลของระบบทั้งหมดถูกควบคุมโดยโฮสต์ (HOST) ซึ่งก็คือเครื่องคอมพิวเตอร์ที่เป็นจุดรวมของอุปกรณ์ทุกตัวที่เชื่อมต่ออยู่นั่นเอง การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรมโดยทุกๆ 1 มิลลิวินาที จะเกิดการรับส่งข้อมูลขึ้น 1 เฟรม แต่เนื่องจากเฟรมข้อมูลจะต้องรับส่งภายใน 1 มิลลิวินาที นั่นหมายความว่าข้อมูลของอุปกรณ์ทุกตัวที่ต่อเชื่อมกับบัสต้องถูกกำหนดขนาดไม่ให้ใหญ่เกินกว่า ที่จะส่งภายใน 1 มิลลิวินาที จึงจำเป็นต้องอาศัย ซอฟต์แวร์มาคอยจัดการและฮาร์ดแวร์ที่จะคอยกระจายการส่งและรวบรวมข้อมูลซึ่งรายละเอียดซอฟต์แวร์และฮาร์ดแวร์ที่จำเป็นสำหรับระบบมีดังนี้

2.3 ซอฟต์แวร์ยูเอสบี

2.3.1 ไดรเวอร์อุปกรณ์ยูเอสบี (USB Device Driver)

ไดรเวอร์อุปกรณ์ เป็นโปรแกรมส่วนที่เก็บข้อมูลที่จำเป็นในการติดต่อไปยังตัวอุปกรณ์แต่ละตัว เมื่อมีความต้องการจะติดต่อกับอุปกรณ์ต่างๆจะต้องแจ้งมายังไดรเวอร์อุปกรณ์ เนื่องจากตัวไดรเวอร์นี้จะรู้ว่าถ้าต้องการติดต่อกับอุปกรณ์จะต้องติดต่อผ่านเอ็นพอยต์(Endpoint)ไหน(อุปกรณ์แต่ละตัวจะมีชนิดและจำนวนเอ็นพอยต์ต่างกัน) ดังนั้นอุปกรณ์แต่ละตัวจะต้องมีไดรเวอร์อุปกรณ์เป็นของตัวเอง

2.3.2 ไดรเวอร์ยูเอสบี (USB Driver)

จากที่กล่าวมาแล้วว่าการส่งข้อมูลต้องมีการแบ่งส่วนเวลากันอย่างพอเหมาะในเวลา 1 เฟรม เพื่อให้อุปกรณ์ทุกตัวสามารถทำงานไปได้พร้อมๆกัน ซอฟต์แวร์ที่ทำหน้าที่นี้คือ ไดรเวอร์ยูเอสบีนั่นเอง

ไดรเวอร์อุปกรณ์ยูเอสบี ของอุปกรณ์แต่ละตัวจะส่งการร้องขอเพื่อการติดต่อมายังไดรเวอร์ยูเอสบี และเมื่อไดรเวอร์ยูเอสบี รับทราบมันจะพิจารณาว่าในรอบการรับส่งข้อมูลหนึ่งๆนั้นอุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้มากแค่ไหน หากปริมาณข้อมูลที่ต้องการรับส่งมีขนาดใหญ่มากจะต้องถูกแบ่งออกเป็นส่วนย่อยๆแล้วเก็บไว้เพื่อรอส่งในรอบถัดไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.3 ไดรเวอร์โฮสต์คอนโทรลเลอร์ (USB HOST Controller Driver)

หลังจากไดรเวอร์ยูเอสบี พิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้างมันก็จะส่งข้อมูลของอุปกรณ์แต่ละตัวที่จะติดต่อในรอบการติดต่อนั้นๆลงมายังไดรเวอร์โฮสต์คอนโทรลเลอร์ ตัวไดรเวอร์โฮสต์คอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลของอุปกรณ์แต่ละชนิดลงเป็นเฟรมข้อมูลและเพิ่มเติมส่วนประกอบต่างๆให้ครบถูกต้องตามมาตรฐานการรับส่งข้อมูลแบบยูเอสบี แล้วส่งข้อมูลทั้งหมดนี้ไปยัง ฮาร์ดแวร์ยูเอสบีโฮสต์คอนโทรลเลอร์ เพื่อส่งข้อมูลทั้งหมดออกไปยังอุปกรณ์ต่างๆ

2.4 ฮาร์ดแวร์ยูเอสบี

2.4.1 ยูเอสบีโฮสต์คอนโทรลเลอร์/ ยูเอสบีรูตฮับ (USB HOST Controller/Root Hub)

ยูเอสบีโฮสต์คอนโทรลเลอร์มีหน้าที่สร้างสัญญาณจริงๆ ขึ้นมาแล้วส่งต่อไปยังรูตฮับ เพื่อกระจายออกไปยังอุปกรณ์ต่างๆตามรูปแบบของยูเอสบี นอกเหนือจากการส่งสัญญาณออกไปภายนอกแล้วรูตฮับยังมีหน้าที่สำคัญอีก 4 อย่างคือ

ควบคุมการใช้พลังงานของอุปกรณ์ที่มาต่อ
ตรวจสอบการเชื่อมต่อของอุปกรณ์ว่าต่ออยู่หรือไม่
เปิดการใช้งานพอร์ตเมื่อมีอุปกรณ์ต่ออยู่และปิดการใช้งานเมื่อปลดอุปกรณ์ออก
รายงานสถานะของแต่ละพอร์ต เมื่อไดรเวอร์โฮสต์คอนโทรลเลอร์ร้องขอมา

2.4.2 ฮับยูเอสบี(USB Hub)

หน้าที่หลักของ ฮับยูเอสบีคือการขยายการเชื่อมต่อให้อุปกรณ์จำนวนมากๆสามารถเชื่อมต่อเข้ากับระบบบัสได้ โดยการทำงานหลักของ ฮับยูเอสบี จะมีอยู่ 2 ส่วนคือ

ทำหน้าที่เป็นตัวทวนสัญญาณ
ตัวจัดการพลังงาน

ในส่วนของการทวนสัญญาณตัวฮับจะต้องรับสัญญาณจากโฮสต์มาแล้วส่งกระจายออกไปยังพอร์ตทุกๆพอร์ต และรับสัญญาณจากแต่ละพอร์ตแล้วนำมารวมกันแล้วส่งไปที่โฮสต์ ในส่วนของการจัดการพลังงานนั้นจะทำหน้าที่เหมือนกับรูตฮับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.3 อุปกรณ์ยูเอสบี (USB Device)

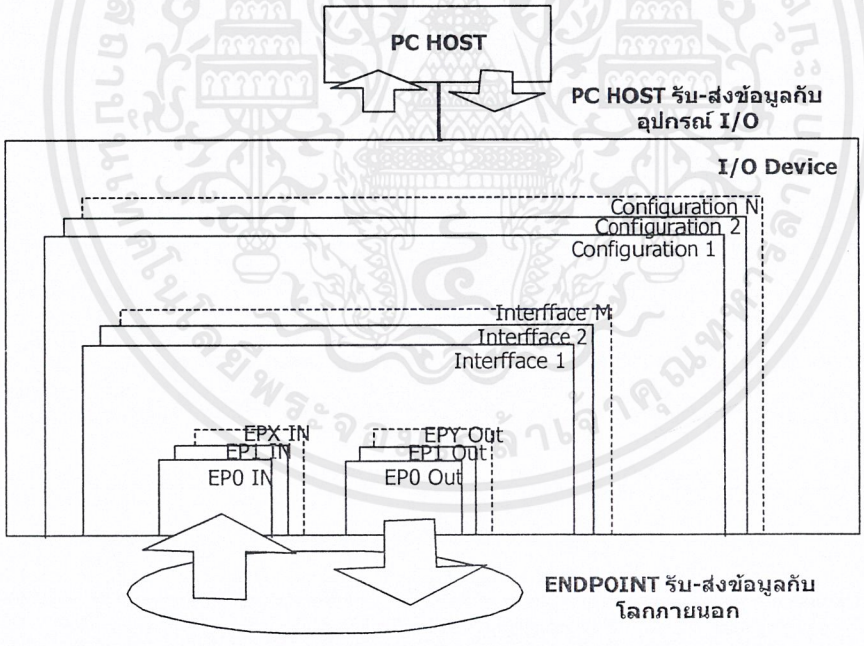
อุปกรณ์ยูเอสบี เป็นฮาร์ดแวร์ส่วนสุดท้ายโดยสามารถแบ่งได้เป็น 2 ประเภทคือ

อุปกรณ์ความเร็วต่ำ รับส่งข้อมูลด้วยความเร็ว 1.5 เมกะบิตต่อวินาที

อุปกรณ์ความเร็วสูง รับส่งข้อมูลด้วยความเร็ว 12 เมกะบิตต่อวินาที

2.5 การติดต่อระหว่างอุปกรณ์กับโฮสต์

อุปกรณ์แต่ละตัวมีคุณสมบัติและการทำงานที่แตกต่างกันโฮสต์จำเป็นต้องรู้คุณสมบัติทั้งหมดของอุปกรณ์แต่ละตัวเพื่อให้การสั่งงานเป็นไปอย่างถูกต้อง และเนื่องจากบัสข้อมูลทั้งหมดจะถูกใช้งานในการรับส่งข้อูลร่วมกันระหว่างอุปกรณ์ทุกตัว สิ่งที่โฮสต์จำเป็นที่จะต้องรู้อีกอย่างก็คือปริมาณข้อมูลที่ต้องการส่งของอุปกรณ์แต่ละตัวในบัส ข้อมูลต่างๆเหล่านี้รวมเรียกว่า ดิซคริปเตอร์ของอุปกรณ์ (Device Descriptor) ซึ่งข้อมูลนั้นจะถูกส่งผ่านเอ็นพอยต์ไปยังโฮสต์ โดยที่การติดต่อระหว่างอุปกรณ์กับโฮสต์แสดงได้ดังรูปที่ 2.5.1



รูปที่ 2.5.1 การรับ-ส่งข้อมูลระหว่างโฮสต์ กับ อุปกรณ์

โดยข้อมูลที่รับเข้ามาหรือส่งออกไปในระบบยูเอสบี นั้นจะกระทำผ่านทางเอ็นพอยต์โดยข้อมูลที่ส่งผ่านนั้นจะจัดเป็นทรานแซ็คชัน(Transaction)ซึ่งแต่ละทรานแซ็คชันประกอบด้วยหลายๆแพ็คเกจ (Packet)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 ดิซคริปเตอร์ของอุปกรณ์

อุปกรณ์แต่ละตัวจะบอกรายละเอียดของตัวเองให้โฮสต์รู้ผ่านดิซคริปเตอร์ชนิดต่างๆ ซึ่งถูกแบ่งแยกตามชนิดของข้อมูลที่จะแจ้งไปยังโฮสต์โดยแต่ละชนิดจะมีเอ็นพอยต์ต่างกัน โดยที่จะมีดิซคริปเตอร์หลักๆในการใช้งานอยู่ 4 ชนิดคือ

2.6.1 ดิไวสซ์ดิซคริปเตอร์ สำหรับข้อมูลทั่วไปของตัวอุปกรณ์

2.6.2 คอนฟิกิวเรชันดิซคริปเตอร์(Configuration Descriptor) สำหรับข้อมูลที่เกี่ยวกับการทำงานในแต่ละโหมดของตัวอุปกรณ์

2.6.3 อินเตอร์เฟซดิซคริปเตอร์(Interface Descriptor) สำหรับข้อมูลการทำงานแต่ละหน้าที่ของตัวอุปกรณ์

2.6.4 เอ็นพอยต์ดิซคริปเตอร์(EndPoint Descriptor) สำหรับข้อมูลการทำงานของแต่ละเอ็นพอยต์

2.6.1 ดิไวสซ์ดิซคริปเตอร์

ในอุปกรณ์แต่ละตัวจะมีดีไวส์ดิซคริปเตอร์เพียงชุดเดียวเท่านั้น ภายในจะระบุข้อมูลที่ใช้ในการเชื่อมต่อขั้นแรกเพื่อใช้ในการตั้งค่าต่างๆของอุปกรณ์และโฮสต์ให้ทำงานเข้ากันได้นอกจากนั้นยังเก็บข้อมูลของคอนฟิกิวเรชันโหมดต่างๆของอุปกรณ์ด้วย เพราะเมื่อครั้งแรกที่อุปกรณ์เชื่อมเข้ากับบัสนั้น โฮสต์ไม่มีทางรู้ได้เลยว่าต้องติดต่อกับอุปกรณ์นี้ที่เอ็นพอยต์ใดจึงจำเป็นต้องเรียกข้อมูลส่วนนี้จากอุปกรณ์ก่อน

2.6.2 คอนฟิกิวเรชันดิซคริปเตอร์

ทำหน้าที่เก็บข้อมูลที่จำเป็นสำหรับการทำงานในแต่ละคอนฟิกิวเรชันโหมดและเก็บจำนวนอินเตอร์เฟซที่ใช้งานในคอนฟิกิวเรชันนั้นๆ

2.6.3 อินเตอร์เฟซดิซคริปเตอร์

ในแต่ละคอนฟิกิวเรชันอาจมีอินเตอร์เฟซได้มากกว่า 1 อินเตอร์เฟซ เพื่อใช้งานในหน้าที่ต่างๆภายในอินเตอร์เฟซดิซคริปเตอร์จะบรรจุข้อมูลการใช้งานอินเตอร์เฟซนั้นๆข้อมูลเหล่านี้จะระบุว่าอุปกรณ์ถูกจัดอยู่ในคลาส(Class)ใดและมีเอ็นพอยต์จำนวนเท่าใดที่ใช้งานในอินเตอร์เฟสนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.4 เอ็นพอยต์ดิซคริปเตอร์

เป็นส่วนที่เก็บข้อมูลคุณสมบัติของแต่ละเอ็นพอยต์เช่นใช้การส่งสัญญาณชนิดใด สามารถรับส่งข้อมูลได้มากที่สุดได้ครั้งละเท่าใด

ส่วนดิซคริปเตอร์อื่นๆนอกเหนือจาก 4 ชนิดที่กล่าวไปแล้วคือ

2.6.5 สตริงดิซคริปเตอร์(Sting Descriptor)

ดิซคริปเตอร์ชนิดนี้จะเก็บข้อมูลตัวอักษรที่สามารถอ่านเข้าใจได้ไว้อธิบายส่วนต่างๆของดิซคริปเตอร์ทั้ง 4 ชนิดข้างต้น เช่น เก็บชื่อบริษัทผู้ผลิต,ซีเรียลนัมเบอร์ของอุปกรณ์ เป็นต้น

2.6.6 คลาสสเป็กซิฟิกดิซคริปเตอร์(Class-Specific Descriptor)

เป็นดิซคริปเตอร์พิเศษที่มีเฉพาะในอุปกรณ์บางตัวที่จัดอยู่ในบางคลาสเท่านั้น ภายในเก็บข้อมูลเฉพาะของคลาสนั้นๆ ที่อยู่นอกเหนือจากดิซคริปเตอร์พื้นฐาน 4 ชนิดแรก

2.6.7 HID ดิซคริปเตอร์(Human Interface Descriptor)

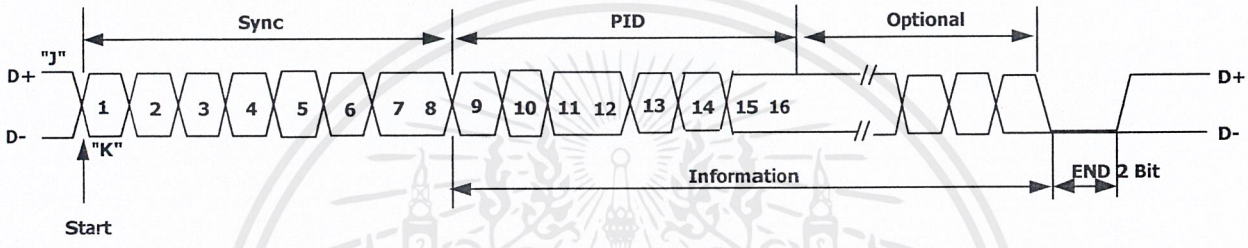
เป็นดิซคริปเตอร์ที่มีอยู่ในระบบปฏิบัติการ อุปกรณ์สามารถเรียกใช้ผ่านดิซคริปเตอร์ชนิดนี้ได้เพื่อลดความยุ่งยากในการเขียนไดร์เวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ชนิดของข้อมูลและการส่งผ่านบนระบบยูเอสบี

ข้อมูลที่ปรากฏบนระบบบัสแบบยูเอสบีนั้นมีพื้นฐานมาจากชุดของข้อมูลที่เรียกว่า แพ็คเกจ ซึ่งมีรายละเอียดดังนี้

3.1 พื้นฐานข้อมูลแบบแพ็คเกจ



รูปที่ 3.1.1 ระบบแพ็คเกจ

การสื่อสารข้อมูลบนระบัสยูเอสบี นั้นอยู่บนพื้นฐานของการจัดเรียงชุดของข้อมูลแพ็คเกจ ดังแสดงไว้ในรูปที่ 3.1.1

ในการใช้งานในโหมดความเร็วสูง(Full Speed: 12 เมกกะบิตต่อวินาที) ในสภาวะปกติบัสจะอยู่ในสภาวะ Idle นั่นคือ ระดับแรงดัน D+ จะมากกว่าแรงดัน D- ซึ่งจะกลับกันกับในโหมดความเร็วต่ำ (Low Speed: 1.5 เมกกะบิตต่อวินาที) ซึ่งจุดเริ่มต้นของแพ็กเกจนั้นจะเริ่มที่ การเปลี่ยนสภาวะจาก “J” ไปเป็น “K” ซึ่งข้อมูล 8 บิตแรกจะเป็นสัญญาณซิงค์ (SYNC) โดยมีวัตถุประสงค์คือเมื่อส่งผ่านข้อมูลไปแล้วทางฝั่งรับสามารถปรับสัญญาณนาฬิกาให้เข้าจังหวะกันได้

หลังจากบิตที่ 8 นั้นจะตามด้วยข้อมูล ซึ่งสามารถบรรจุข้อมูลได้ตั้งแต่ 0 ถึง 1023 ไบต์ โดยบิตที่ 9-16 เป็นแพ็คเก็ตบ่งชี้ (PID: Identifier Packet) โดยบิตที่ 9-12 จะมีสภาวะตรงกันข้าม กับบิตที่ 13-16 โดยแพ็กเก็ตบ่งชี้ จะบอกให้รู้ว่าข้อมูลที่ตามหลังมาเป็นข้อมูลชนิดไหนดังตารางที่ 3.1.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PID บิตที่ 9-12	ชนิดของแพ็คเกจ	ประเภทของแพ็คเกจ
0101	SOF	Token
1101	SETUP	Token
1001	IN	Token
0001	OUT	Token
0011	DATA0	Data
1011	DATA1	Data
0010	ACK	Handshake
1010	NAK	Handshake
1110	STALL	Handshake
1100	PRE	Special
Other	RESERVED	RESERVED

ตารางที่ 3.1.1 ชนิดของแพ็คเกจที่ระบุโดยค่า PID

และจุดสิ้นสุดของแพ็คเกจ (EOP: End Of Packet) คือสภาวะสัญญาณที่ขา D+ และ D- เป็นลอจิกต่ำ (low) ช่วงเวลา 2 บิต ซึ่งเป็นจุดเดียวในการส่งข้อมูลที่สัญญาณไม่เป็นสัญญาณ เรียกสภาวะนี้ว่า Single-Ended-Zero (SE0)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 ชนิดของแพ็คเกจ (Packet Type)

เมื่อจัดประเภทของแพ็คเกจเป็น 4 ชนิดตามตารางที่ 2.1) เราสามารถแบ่งประเภทของแพ็คเกจออกเป็น 4 ชนิดคือ

3.2.1 แพ็คเกจโทเคน (Token packet) เป็นแพ็คเกจสำหรับเตรียมความพร้อมก่อนที่จะส่งแพ็คเกจข้อมูล

3.2.2 แพ็คเกจข้อมูล (Data packet) เป็นข้อมูลที่จะส่ง

3.2.3 แพ็คเกจตรวจสอบ (Handshake packet) เป็นแพ็คเกจแจ้งสภาวะการส่งว่าเรียบร้อยดีหรือไม่

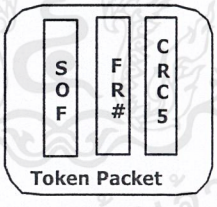
3.2.4 แพ็คเกจพิเศษ(Special packet) เป็นแพ็คเก็ตที่ใช้ในการส่งสำหรับอุปกรณ์ความเร็วต่ำ

เราสามารถแบ่งย่อยชนิดของแพ็คเกจตามประเภทได้ดังนี้

3.2.1 แพ็คเกจโทเคน

สามารถแบ่งย่อยออกได้ดังนี้

แพ็คเกจเริ่มต้น (SOF: Start Of Frame Token packet)



รูปที่ 3.2.1 แพ็กเกจเริ่มต้น

เมื่อเรานำออสซิลโลสโคป มาจับในขณะที่ไม่มีการรับส่งข้อมูลบนบัสยูเอสบี พบว่าในทุกๆ 1 มิลลิวินาทีโฮสต์ จะส่งแพ็คเกจเริ่มต้น ออกมาและช่วงระยะเวลาระหว่าง แพ็คเกจเริ่มต้นแรกกับ แพ็คเกจเริ่มต้นถัดไปเราจะเรียกว่าเฟรม เมื่อเราใช้คริสตัล 12 เมกกะเฮิร์ต ในการสร้างสัญญาณนาฬิกาจะได้

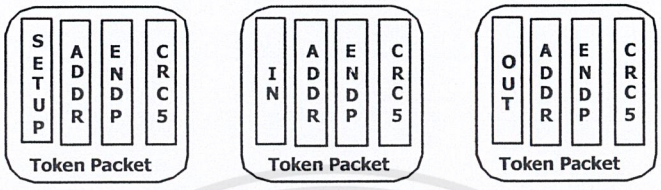
ระยะเวลา 1 บิตข้อมูลเท่ากับ $\frac{1}{12\ MHz} = 83.33\ ns$

ดังนั้นใน 1 มิลลิวินาที เราสามารถส่งข้อมูลได้ $\frac{1\ ms}{83.33\ ns} = 12{,}000\ bit$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หรือส่งได้สูงสุด 1,500 ไบต์ซึ่งเป็นค่าทางทฤษฎี ในทางปฏิบัติจะส่งด้วยอัตราการส่งข้อมูลไม่เกิน 1,200 ไบต์ต่อเฟรม

แพ็คเกจขาเข้า ขาออก และ SETUP แพ็คเกจ (IN, OUT, SETUP Token packet)



รูปที่ 3.2.2 แพ็คเกจขาเข้า ขาออก และ SETUP แพ็คเกจ

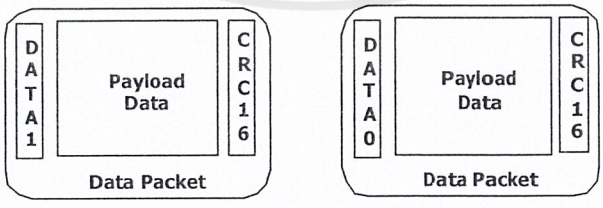
จากรูปที่ 3.2.2 ทั้ง 3 แพ็คเก็ตจะประกอบด้วยแอดเดรสของอุปกรณ์ขนาด 7 บิต (สามารถต่ออุปกรณ์ได้ 127 ตัว), แอดเดรสของเอ็นพอยต์4 บิต และ ส่วนตรวจสอบความผิดพลาดของข้อมูลอีก 5 บิต โดยหน้าที่ของแต่ละแพ็คเก็ตเป็นดังนี้

แพ็คเกจขาเข้า เป็นการเตรียมที่จะส่งแพ็คเกจข้อมูลจากอุปกรณ์ไปยังโฮสต์

แพ็คเกจขาออก เป็นการเตรียมที่จะส่งแพ็คเกจข้อมูลจากโฮสต์ไปยังอุปกรณ์ ซึ่งทั้งแพ็คเกจขาเข้า และแพ็คเกจขาออก นั้นสามารถส่งไปยังแอดเดรสใดๆก็ได้

SETUP แพ็คเกจ จัดอยู่ในประเภทแพ็คเกจขาออก แต่มีลำดับความสำคัญมากกว่า กล่าวคือถึงแม้ว่าอุปกรณ์จะปฏิเสธที่จะรับแพ็คเกจขาออก แต่ไม่สามารถปฏิเสธ SETUP แพ็คเกจ ได้ดังนั้นบ่อยครั้งเราจะใช้ SETUP แพ็คเกจ ในการปลุกอุปกรณ์จากการที่อยู่ในโหมดประหยัดพลังงาน

3.2.2 แพ็คเกจข้อมูล

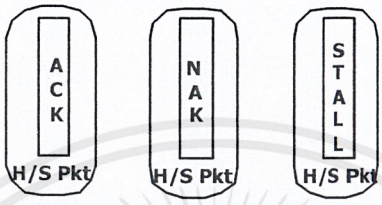


รูปที่ 3.2.3 แพ็คเกจข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากที่ส่ง แพ็คเกจโทเคน แล้วเราจะส่งแพ็คเกจข้อมูล ซึ่งคือข้อมูล (Payload Data) ที่เราต้องการจะส่ง ซึ่งสามารถส่งได้ตั้งแต่ 0-1023 ไบต์โดยจัดรูปแบบเป็น DATA1 กับ DATA0 เรียกว่าดาต้าท็อกเกิล (Data toggle) ซึ่งจะอธิบายทีหลัง

3.2.3 แพ็คเกจตรวจสอบ



รูปที่ 3.2.4 แพ็คเกจตรวจสอบ

เป็นแพ็คเกจสำหรับใช้ในการบอกสภาวะของข้อมูลด้านฝั่งรับที่ตอบกลับไปยังฝั่งที่ส่งซึ่งมีอยู่ 3 ชนิดคือ

แพ็คเกจรับรู้ (ACK : Acknowledge Handshake packet) ใช้สำหรับตอบกลับไปว่าได้รับข้อมูลเรียบร้อยแล้ว

แพ็คเกจปฏิเสธ (NAK: Negative Acknowledge Handshake packet) ใช้สำหรับตอบกลับไปว่ายังไม่สามารถรับข้อมูลตอนนี้ได้เนื่องจากเกิดปัญหา เช่น อาจมีสาเหตุมาจากบัฟเฟอร์ไม่พอเป็นต้น แพ็คเกจปฏิเสธ นี้เกิดกับทางฝั่งรับเท่านั้นและสามารถปฏิเสธการรับข้อมูลได้ทุกทรานแซ็คชัน ยกเว้น SETUP แพ็คเกจแต่โฮสต์ จะไม่มีทางตอบกลับเป็นแพ็คเกจปฏิเสธ เพราะว่าโฮสต์ ถูกทำให้มีเวลาและทรัพยากรมากพอเสมอ

Stall แพ็คเกจ (Stall Handshake packet) ใช้ในการตอบกลับว่าเกิดปัญหาขึ้นกับอุปกรณ์ เช่นอาจจะไม่เข้าใจคำสั่งที่โฮสต์ส่งมา เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 การประกอบแพ็คเกจเป็นทรานแซ็คชัน

ทรานแซ็คชัน คือ การจัดแพ็คเกจของข้อมูลที่จะส่งโดยสามารถแบ่งทรานแซ็คชันออกเป็น 4 ชนิดตามเวลาและคุณภาพ ดังตารางที่ 3.3.1

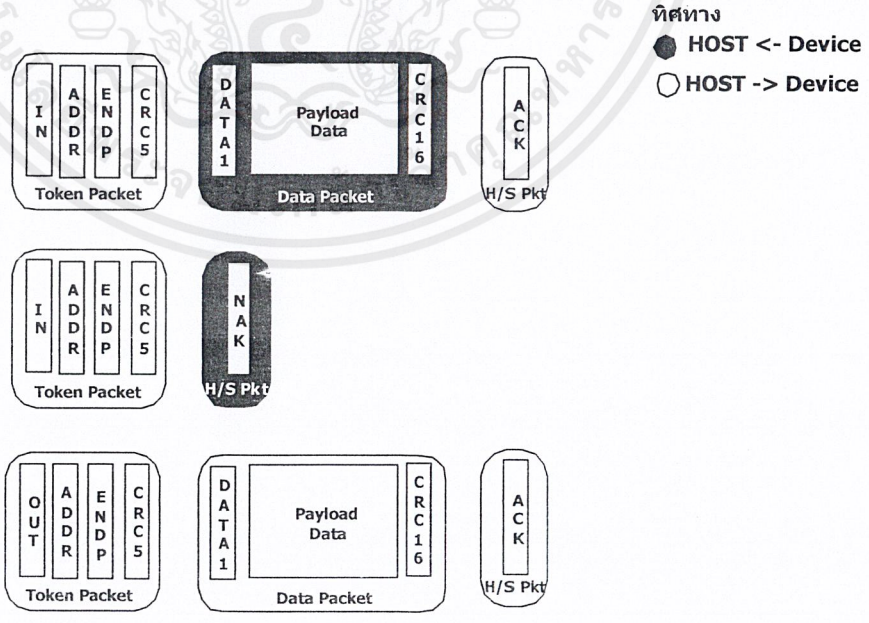
ชนิด	สิ่งที่ให้ความสำคัญ	ขนาด (ไบต์)	ตัวอย่าง
Interrupt	คุณภาพ	64 (8 สำหรับLS)	Mouse , Keyboard
Bulk	คุณภาพ	64	Printer , Scanner
Isochronous	เวลา	1024	Speakers , Video
Control	คุณภาพ และ เวลา	64 (8 สำหรับLS)	System Control

ตารางที่ 3.3.1 ชนิดของทรานแซ็คชัน

เมื่อแบ่งตามความสำคัญของเวลา ข้อมูลที่จะส่งจากโฮสต์ จะถูกจองสัดส่วนไว้ในทุกๆ เฟรม สำหรับการส่งข้อมูลแบบ Isochronous และ Control Transfer

และเมื่อแบ่งตามคุณภาพเรามั่นใจได้เลยว่าข้อมูลที่ได้จาก Interrupt, Bulk จะได้ข้อมูลที่ครบถ้วนแน่นอนแต่อาจจะใช้เวลามากกว่า

3.3.1 การจัดแพ็คเกจข้อมูลแบบอินเตอร์รัฟต์ (Interrupt Transfer)



รูปที่ 3.3.1 การจัดแพ็คเกจข้อมูลแบบอินเตอร์รัฟต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้การส่งผ่านข้อมูลแบบอินเตอร์รัฟต์นี้มี 2 แบบคือแบบที่หนึ่งเมื่อเกิดอินเตอร์รัฟต์ขึ้นก็ให้ไปสู่โปรแกรมสำหรับบริการอินเตอร์รัฟต์เลย หรือวิธีที่ 2 ใช้การโพลลิ่ง (Polling)นั่นคือการวนมาตรวจสอบตามเวลที่กำหนด เช่น โฮสต์ จะทำการโพลลิ่ง ทุกๆ 1 ms หรือ 1000 ครั้งต่อนาที เต่เนื่องจากอุปกรณ์ของเราบางอย่างอย่างตอบสนองไม่ทันดังนั้นอุปกรณ์อาจจะใช้เวลาในการโพลลิ่งที่น้อยกว่าโฮสต์

จากรูปที่ 3.3.1 เป็นการจัดแพ็กเกจเป็นทรานแซ็คชันขาเข้า (IN Transaction) (2 รูปบน) และทรานแซ็คชันขาออก (OUT Transaction) รูปล่างสุด จากรูปบนหลังจากโฮสต์ทำการโพลลิ่ง ไปที่อุปกรณ์แล้วพบว่าอุปกรณ์ต้องการส่งข้อมูลมาที่โฮสต์ ตัวโฮสต์จึงจัดแพ็กเก็ตเป็นทรานแซ็คชันขาเข้า เพื่อส่งไปที่อุปกรณ์ถ้าทางอุปกรณ์มีข้อมูลที่พร้อมจะส่งเรียบร้อยจะส่งข้อมูลกลับมาที่โฮสต์ หลังจากได้รับแพ็คเกจขาเข้า เมื่อโฮสต์ได้รับข้อมูลเรียบร้อยจะส่งแพ็คเกจรับรู้ ตอบกลับไป

จากรูปตรงกลางถ้าอุปกรณ์มีปัญหาเกี่ยวกับข้อมูลที่จะส่งหลังจากได้รับแพ็คเกจขาเข้า แล้วมันจะตอบกลับไปด้วยแพ็คเกจปฏิเสธ เมื่อโฮสต์ได้รับแพ็คเกจปฏิเสธแล้ว จะทำการพยายามอีกในเฟรมถัดไป

ส่วนรูปข้างล่างสุดเป็นการจัดเป็นทรานแซ็คชันขาออก ถ้าอุปกรณ์มีความพร้อมในการรับข้อมูล เช่น บัฟเฟอร์ว่างโฮสต์ จะทำการส่งข้อมูลไปต่อจากการส่งแพ็คเกจขาออก และเมื่ออุปกรณ์ได้รับข้อมูลเรียบร้อยมันจะตอบแพ็คเกจรับรู้ กลับมา

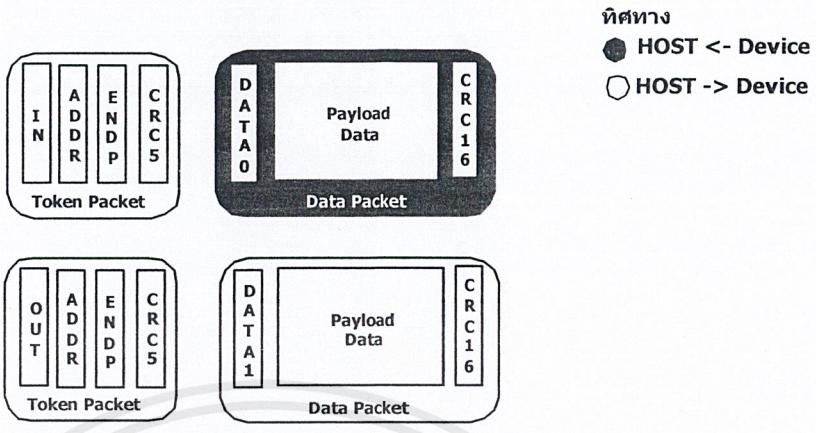
3.3.2 การจัดแพ็คเกจข้อมูลแบบบัลต์ (Bulk Transfer)

การส่งแบบนี้จะเหมือนกับการจัดแพ็คเกจข้อมูลแบบอินเตอร์รัฟต์ ต่างกันตรงที่ข้อมูลที่ส่งนั้นมีขนาดใหญ่กว่าและใช้ในการรับส่งข้อมูลที่ใช้กับงานที่ไม่ต้องการอัตราเร็วในการส่งข้อมูลที่คงที่แต่ข้อมูลที่จะส่งต้องไม่เสียหาย การรับส่งข้อมูลชนิดนี้ไม่สามารถจองการใช้งานแบนด์วิดธ์ของบัสได้ ดังนั้นการรับส่งข้อมูลจึงเกิดขึ้นเมื่อบัสว่างเท่านั้น

เนื่องจากอุปกรณ์ที่ใช้ในการส่งข้อมูลชนิดนี้ส่วนใหญ่จะต้องการความถูกต้องของข้อมูลสูงในการส่งข้อมูลแต่ละครั้งจึงมีการตรวจความถูกต้องในทุกๆครั้งและจะเกิดการส่งใหม่ถ้ามีความผิดพลาดเกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 การจัดแพ็คเกจข้อมูลแบบไอโซโครนัส (Isochronous Transfer)



รูปที่ 3.3.2 การจัดแพ็คเกจข้อมูลแบบไอโซโครนัส

การรับส่งข้อมูลชนิดนี้จะมีการส่งหรือรับข้อมูลในทุกๆเฟรมข้อมูล ทำให้เกิดการรับส่งข้อมูลด้วยความเร็วคงที่ แต่ก็หมายความว่าการรับส่งแบบนี้จะต้องจองการรับส่งข้อมูลในแต่ละเฟรมไว้ตลอดเวลาด้วย

ประเภทของการใช้งานที่ใช้รับส่งข้อมูลชนิดนี้ก็คือ งานที่ต้องการอัตราการส่งข้อมูลที่คงที่ตลอดเวลา แต่จากที่ว่าต้องรับและส่งข้อมูลในทุกๆเฟรมนั่นหมายความว่าหากเกิดความผิดพลาดของข้อมูลจะไม่มีการพยายามส่งข้อมูลที่ผิดนั้นกลับไปใหม่(ไม่มีการรับประกันความถูกต้อง)สังเกตุจากรูปที่ 3.3.2 จะไม่มีแพ็คเกจตรวจสอบ หากการส่งข้อมูลในเฟรมใดผิดพลาดก็จะละทิ้งข้อมูลในเฟรมนั้นไปเลยโดยเอาข้อมูลในเฟรมก่อนหน้ามาใช้

ก่อนเริ่มการทำงานของอุปกรณ์จะต้องทำการซิงโครไนซ์การทำงานของตัวเองเข้ากับแพ็คเกจเริ่มต้นในแต่ละเฟรม การซิงโครไนซ์มี 3 วิธีคือ

อะซิงโครนัส (Asynchronous)

การซิงโครไนซ์วิธีนี้หมายความว่าสัญญาณนาฬิกาของอุปกรณ์ไม่จำเป็นต้องเข้าจังหวะกับสัญญาณข้อมูลของยูเอสบี แต่อัตราการส่งข้อมูลจะต้องถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่งตอนเริ่มต้นการเชื่อมต่อ

ซิงโครนัส (synchronous)

การซิงโครไนซ์วิธีนี้หมายความว่าสัญญาณนาฬิกาของอุปกรณ์ต้องเข้าจังหวะกับสัญญาณของแพ็คเกจเริ่มต้นของยูเอสบี โดยก่อนเริ่มการทำงานอุปกรณ์จะต้องมีการปรับสัญญาณให้เข้ากับจังหวะของสัญญาณของแพ็คเกจเริ่มต้น แต่หากอุปกรณ์ไม่สามารถปรับสัญญาณนาฬิกาของตัวเองได้ยูเอสบี จะยอมให้จังหวะการส่งแพ็คเกจเริ่มต้นของอุปกรณ์เป็นหลัก โดยอุปกรณ์นั้นจะถูกเรียก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ว่า อุปกรณ์มาสเตอร์ (Master Device) อัตราการส่งข้อมูลจะต้องถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่งตอนเริ่มต้นการเชื่อมต่อเช่นเดียวกับ แบบอะซิงโครนัส

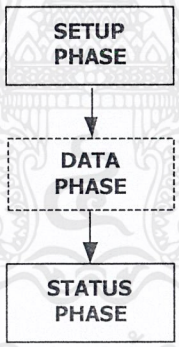
อะแดปทีฟ (Adaptive)

เป็นวิธีที่อุปกรณ์มีความสามารถมากที่สุดเพราะว่าอุปกรณ์จะสามารถปรับความเร็วในการส่งข้อมูลได้ในช่วงที่กว้างมาก ไม่ถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่ง สามารถเปลี่ยนแปลงความเร็วได้ในระหว่างการใช้งาน สัญญาณนาฬิกาภายในระบบของอุปกรณ์จะเข้าจังหวะกับสัญญาณข้อมูลที่ส่งในสาย

3.3.4 การจัดแพ็คเกจข้อมูลแบบคอนโทรล (Control Transfer)

การส่งข้อมูลแบบนี้มีความยุ่งยากมากกว่าแบบอื่นๆเพราะว่าต้องการทั้ง คุณภาพและเวลา โดยเราจะใช้ การจัดแพ็คเกจข้อมูลแบบคอนโทรล นี้ตอนกระทำกระบวนการคอนฟิกอุปกรณ์ (Enumeration Process) และตอนตั้งค่าเริ่มต้นระบบ

ซึ่งส่วนมากจะใช้ในการส่งเป็นคำสั่งควบคุมแต่ก็สามารถส่งข้อมูลได้ด้วยถ้าจำเป็นโดยจะแบ่งเป็นเฟสดังรูปที่ 3.3.3



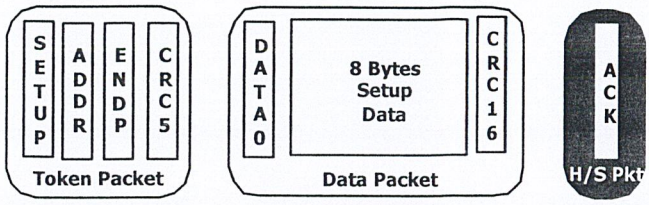
รูปที่ 3.3.3 เฟสของการจัดแพ็คเกจข้อมูลแบบคอนโทรล

การรับส่งข้อมูลชนิดนี้อาจจะถือได้ว่าเป็นการรับส่งข้อมูลที่สำคัญที่สุดก็ว่าได้ เพราะจะถูกใช้สำหรับการส่งคำสั่งควบคุมการทำงานของอุปกรณ์ทั้งหมด ตั้งแต่เริ่มแรกที่อุปกรณ์ถูกเชื่อมเข้ากับระบบก็จะเกิดการอ่านข้อมูลดิซคริปเตอร์ต่างๆเมื่อได้ข้อมูลมาโฮสต์ก็จะพิจารณาว่าสามารถรองรับการทำงานได้หรือไม่ ทั้งหมดนี้จะใช้คอนโทรลทรานสเฟอร์เพื่อติดต่อกับคอนโทรลเอ็นพอยต์ทั้งสิ้น

จะว่าไปแล้วการแบ่งเป็นเฟสก็คล้ายกับการจัดเป็นทรานแซ็คชัน แต่เพื่ออ้างถึงการใช้กับการจัดแพ็คเกจข้อมูลแบบคอนโทรลจึงใช้คำว่าเฟสโดยที่ เฟส Setup จะประกอบไปด้วย Setup โทเคน ดังรูปที่ 3.3.4

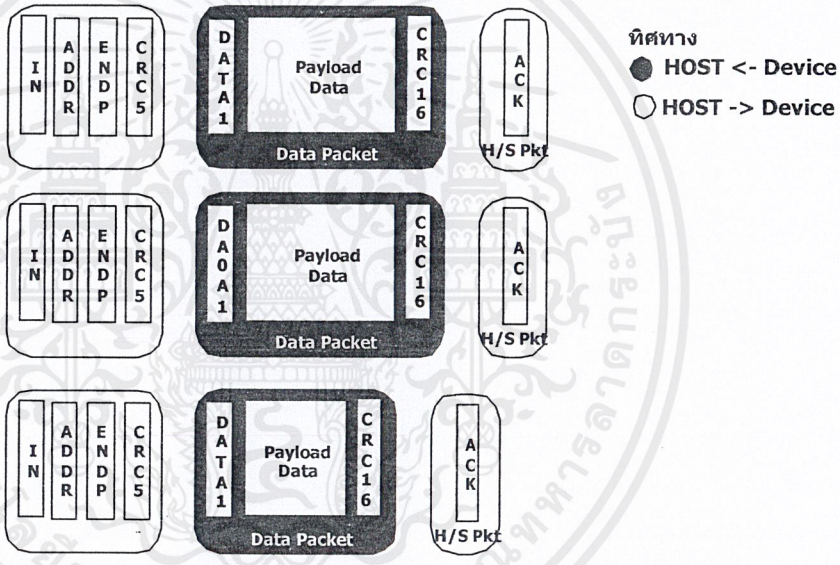
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

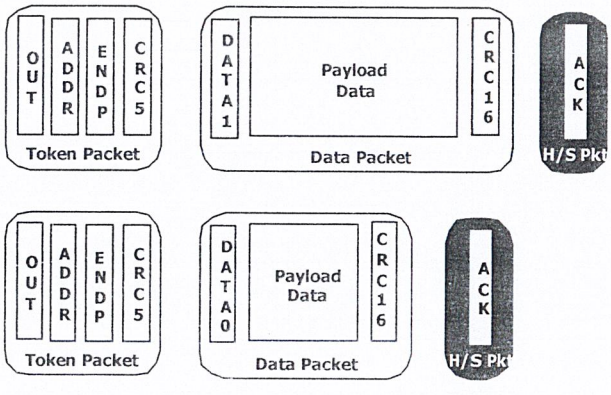


รูปที่ 3.3.4 เฟส Setup ของการจัดแพ็คเกจข้อมูลแบบคอนโทรล

หากต้องการใช้การจัดแพ็คเกจข้อมูลแบบคอนโทรลในการอ่านหรือเขียนข้อมูลจากอุปกรณ์ก็สามารถทำได้เช่นกัน โดยการจัดแพ็คเกจเป็นดังรูปที่ 3.3.5 สำหรับการอ่าน และจัดเป็นรูปที่ 3.3.6 สำหรับการเขียน



รูปที่ 3.3.5 เฟสการอ่านข้อมูลของการจัดแพ็คเกจข้อมูลแบบคอนโทรล



รูปที่ 3.3.6 เฟสการเขียนข้อมูลของการจัดแพ็คเกจข้อมูลแบบคอนโทรล

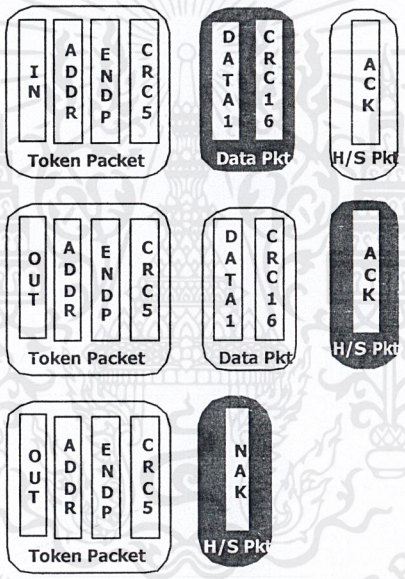
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
50339

การจัดแพ็คเกจข้อมูลแบบคอนโทรล นั้นจะต้องจบด้วยเฟสแสดงสภาวะ (Status Phase) โดยจะใช้แพ็คเกจขาเข้า เมื่ออุปกรณ์เป็นตัวกำหนดสภาวะ และใช้แพ็คเกจขาออกเมื่อโฮสต์เป็นตัวกำหนดสภาวะ

ในกรณีที่เฟสข้อมูล (Data Phase) เป็นการอ่านข้อมูลของการจัดแพ็คเกจข้อมูลแบบคอนโทรลแล้วโฮสต์ จะเป็นตัวที่ส่งแพ็คเกจตรวจสอบ กลับไปดังรูปที่ 3.3.7

ในกรณีเฟสข้อมูลเป็นการเขียนข้อมูลของการจัดแพ็คเกจข้อมูลแบบคอนโทรลแล้วอุปกรณ์ จะเป็นตัวที่ส่งแพ็คเกจตรวจสอบ กลับไป

และในกรณีที่ไม่มีเฟสข้อมูล อุปกรณ์ก็ต้องส่งแพ็คเกจตรวจสอบ กลับด้วยเช่นกัน



รูปที่ 3.3.7 เฟสแสดงสภาวะของการจัดแพ็คเกจข้อมูลแบบคอนโทรล

3.4 ดาต้าท็อกเกิล (Data toggle)

สาเหตุที่ต้องมีการกำหนดแพ็คเกจข้อมูลออกเป็น 2 ชนิดก็เพื่อแก้ปัญหาการรับส่งข้อมูลไม่ถูกต้องในบางรูปแบบ ซึ่งเป็นช่องว่างในการตรวจสอบทั้งหมดก่อนหน้านี้เช่น ในกรณีของทรานแซ็คชันขาเข้า เริ่มต้นจากโฮสต์ส่ง แพ็คเกจขาเข้าไปยังอุปกรณ์ การส่งสมบูรณ์อุปกรณ์ได้รับข้อมูลถูกต้องจึงส่งแพ็กเก็ตข้อมูลที่โฮสต์ต้องการกลับมา ซึ่งโฮสต์ก็ได้รับอย่างถูกต้องอีกเช่นกันจึงส่งแพ็คเกจตรวจสอบ กลับไปยังอุปกรณ์ แต่โชคร้ายเกิดขึ้นขณะกำลังส่งแพ็คเกจตรวจสอบ ทำให้เกิด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ความผิดพลาดขึ้นกับแพ็คเกจตรวจสอบ เมื่ออุปกรณ์ได้รับข้อมูลที่ผิดก็จะไม่สนใจแพ็คเกจนั้นๆผลที่ตามมาก็คือ อุปกรณ์จะคิดว่าโฮสต์ยังไม่ได้รับข้อมูลที่ตัวเองส่งไป เมื่อถึงทรานแซ็คชันขาเข้าครั้งถัดไปอุปกรณ์ก็จะส่งข้อมูลชุดเดิมที่คิดว่าโฮสต์ยังไม่ได้รับในขณะที่โฮสต์เข้าใจว่าข้อมูลนี้คือข้อมูลชุดใหม่

วิธีแก้ไขข้อปัญหานี้ เรียกว่า ดาต้าท็อกเกิล ซึ่งวิธีนี้เป็นการจัดลำดับการใช้งานแพ็คเกจข้อมูล 0 และ 1 เพื่อให้โฮสต์หรือตัวอุปกรณ์ทราบถึงความผิดพลาดในการส่งข้อมูล โดยกระบวนการส่งข้อมูลจะเริ่มต้นที่การส่งข้อมูลในทรานแซ็คชันแรกซึ่งใช้แพ็คเกจข้อมูล 0 ในการส่งหรือรับข้อมูลและเมื่อมีการรับส่งข้อมูลครั้งต่อไปก็จะสลับไปใช้แพ็คเกจข้อมูล 1 และสลับกันไปเรื่อยๆ เช่น จากปัญหาที่แล้วเริ่มจากโฮสต์ส่ง แพ็คเกจโทเคน ไปยังอุปกรณ์เรียบร้อยถูกต้อง ซึ่งอุปกรณ์ก็จะส่งแพ็คเกจข้อมูลกลับมายังโฮสต์ และเนื่องจากเป็นการส่งแพ็คเกจข้อมูลครั้งแรกจึงต้องใช้แพ็คเกจข้อมูล 0 เมื่อโฮสต์ได้รับข้อมูลถูกต้องก็จะส่งแพ็คเกจตรวจสอบไปยังอุปกรณ์ และได้เกิดความผิดพลาดขึ้นกับแพ็คเกจตรวจสอบนี้ทำให้อุปกรณ์คิดว่าโฮสต์ไม่ได้รับข้อมูล เมื่อถึงทรานแซ็คชันขาเข้าครั้งถัดไปอุปกรณ์ก็จะส่งข้อมูลชุดเดิมที่คิดว่าโฮสต์ยังไม่ได้รับกลับมาอีกครั้งโดยใช้แพ็คเกจข้อมูล 0 เหมือนเดิมเมื่อโฮสต์ได้รับแพ็คเกจข้อมูล 0 ซ้ำอีกครั้งก็จะรู้ได้ว่านี่เป็นข้อมูลชุดเดิมไม่ต้องเอาไปใช้งาน

3.5 เทคนิคการเข้ารหัสสัญญาณ NRZI (Non Return to Zero Inverted)

การเข้ารหัสสัญญาณข้อมูลก่อนจะส่งออกไปนั้นใช้เทคนิค 2 ส่วนประกอบกันคือ NRZI และการเติมสตัฟบิต

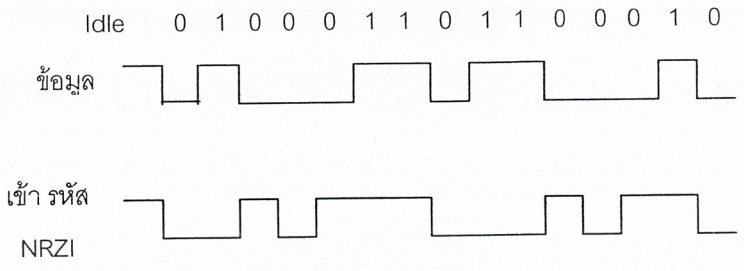
3.5.1 การเข้ารหัส NRZI

ก่อนจะส่งข้อมูลไปในสายส่งนั้น ข้อมูลจะต้องถูกเข้ารหัสแบบ NRZI ก่อน ประโยชน์ของการเข้ารหัสข้อมูลแบบนี้ก็คือการแฝงสัญญาณนาฬิกาส่งไปกับตัวข้อมูลด้วย ทำให้อุปกรณ์ปลายทางสามารถซิงโครไนซ์ความถี่ของตัวเองให้ตรงกับความถี่ของฝั่งโฮสต์ได้ ซึ่งทำให้สามารถอ่านข้อมูลที่ส่งมาได้อย่างถูกต้อง

รูปที่ 3.5.1 แสดงให้เห็นการเข้ารหัสแบบ NRZI ของข้อมูลการเปลี่ยนแปลงของระดับสัญญาณจากระดับลอจิกสูง (High) เป็นระดับลอจิกต่ำ (Low) หรือจากระดับลอจิกต่ำ เป็นระดับลอจิกสูง จะถือเป็นข้อมูล 0 แต่ถ้าไม่เกิดการเปลี่ยนแปลงของสัญญาณจะถือเป็นข้อมูล 1 แต่ด้วยการเข้ารหัสวิธีนี้หากข้อมูลที่ส่งเป็น 1 เรียงกันหลายๆบิตก็จะไม่เกิดการเปลี่ยนแปลงระดับแรงดันใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สายส่งเลยซึ่งอาจทำให้เกิดความผิดพลาดในการส่งข้อมูลได้จึงต้องมีอีกเทคนิคหนึ่งในการส่งข้อมูล เพื่อช่วยป้องกันความผิดพลาดจากเหตุการณ์ดังกล่าวนั่นก็คือการเติมสตัฟบิต

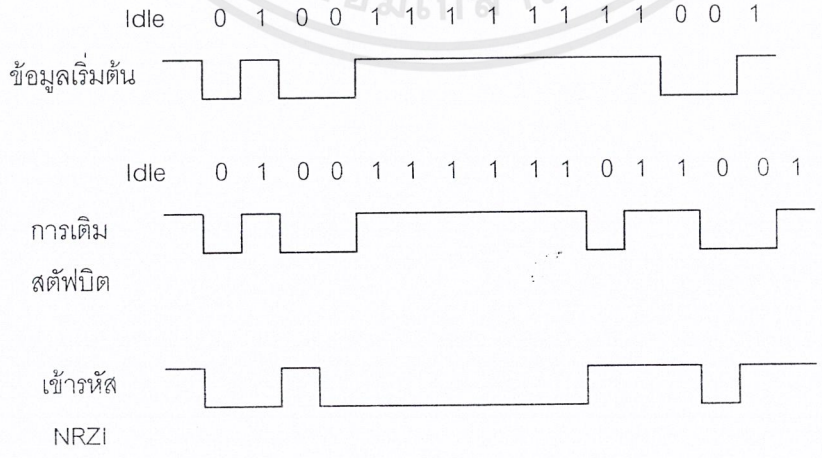


รูปที่ 3.5.1 การเข้ารหัสข้อมูลเป็น NRZI

3.5.2 การเติมสตัฟบิต (Bit Stuffing)

การเติมสตัฟบิตจะเกิดขึ้นเมื่อข้อมูลที่จะส่งเป็น 1 เรียงติดกัน 6 ตัว คือหลังจากข้อมูล 1 เรียงกันครบ 6 ตัวจะเกิดการเติม 0 เข้าในตัวข้อมูลที่จะส่งโดยอัตโนมัติ (แม้ว่าข้อมูลที่ 7 จะเป็น 0 ก็ตาม) การทำเช่นนี้ทำให้การส่งข้อมูลที่เข้ารหัสแบบ NRZI เกิดการเปลี่ยนแปลงสัญญาณทุกๆ 7 บิต ข้อมูล และเมื่อทางฝั่งรับได้รับข้อมูล 1 เรียงกันครบ 6 บิตก็จะตัดข้อมูล 0 ที่ตามมาทิ้งไปเพื่อทำให้ข้อมูลที่ได้รับถูกต้อง

จากรูปที่ 3.5.2 เป็นตัวอย่างของการเติมสตัฟบิตลงในข้อมูลโดยแถวบนสุดของรูปแสดงให้เห็นถึงข้อมูลที่จะส่งจริงๆแต่เนื่องจากข้อมูลที่จะส่งเป็นข้อมูล 1 ติดกันถึง 8 ตัวดังนั้นหลังจากส่งข้อมูล 1 ที่ติดกัน 6 บิตเสร็จแล้ว จะมีการเติมข้อมูล 0 ลงในข้อมูลหลัก และหลังจากการเติมสตัฟบิตเรียบร้อยแล้วก็จะส่งข้อมูลหลักที่อยู่ต่อหลังไปเรื่อยๆจนครบ



รูปที่ 3.5.2 การเติมสตัฟบิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การคอนฟิกอุปกรณ์ (Device Configuration)

จากเนื้อหาทั้งหมดได้กล่าวถึงส่วนประกอบต่างๆของยูเอสบี จึงสามารถสรุปการทำงานของระบบเริ่มตั้งแต่มีอุปกรณ์เชื่อมต่อเข้ามาในระบบว่ามีลำดับขั้นตอนอย่างไรบ้าง การสื่อสารข้อมูลระหว่างโฮสต์และอุปกรณ์ในขั้นเริ่มต้นตั้งแต่อุปกรณ์ตังใหม่เชื่อมต่อเข้ามาในระบบไปจนถึงอุปกรณ์สามารถทำงานได้ตามหน้าที่ จัดเรียกโดยรวมว่า การคอนฟิกอุปกรณ์ โดยมีลำดับดังนี้

4.1 ขั้นตอนการคอนฟิกอุปกรณ์ (Configuration Peocess)

ขั้นที่ 1 ตรวจสอบสถานะการเชื่อมต่อและชนิดของอุปกรณ์

เมื่อมีอุปกรณ์ตังใหม่เชื่อมต่อเข้ามาในระบบ ฮับจะทราบจากการตรวจสอบระดับแรงดันของสายส่งว่าอยู่ในสถานะใด ซึ่งจะรู้ด้วยว่าอุปกรณ์ที่เชื่อมต่อเข้ามานั้นเป็นอุปกรณ์ความเร็วสูงหรือต่ำ หลังจากนั้นจะเช็คแฟลกสถานะของพอร์ตนั้นๆ เพื่อรอการอ่านกลับจากโฮสต์ ในขั้นตอนนี้ยังไม่มีการจ่ายไฟเลี้ยงไปยังตัวอุปกรณ์

ขั้นที่ 2 จ่ายไฟเลี้ยงไปยังพอร์ต

หลังจากโฮสต์รู้ว่าพอร์ตใดมีอุปกรณ์ตัวใหม่ต่อเข้ามาโดยการอ่านแฟลกสถานะจากฮับแล้ว โฮสต์จะสั่งให้ฮับจ่ายไฟเลี้ยงให้แก่พอร์ตที่ต่อกับอุปกรณ์ตัวใหม่นั้น โดยในขั้นตอนนี้อุปกรณ์จะดึงกระแสได้ไม่เกิน 100 มิลลิแอมป์ อุปกรณ์จะดึงกระแสได้เต็มที่ก็ต่อเมื่อได้รับการคอนฟิกจากโฮสต์แล้วเท่านั้น

ขั้นที่ 3 รีเซ็ตพอร์ตของอุปกรณ์ที่เชื่อมต่อใหม่

เมื่ออุปกรณ์ตัวใหม่ได้รับไฟเลี้ยงเรียบร้อยแล้วโฮสต์จะส่งสถานะรีเซ็ตไปยังพอร์ตนั้น โดยส่งคำสั่งไปยังฮับที่ควบคุมการทำงานของพอร์ตนั้น ซึ่งเมื่ออุปกรณ์ได้รับสถานะนี้แล้วอุปกรณ์นั้นจะต้องรีเซ็ตตัวเองและปรับค่าต่างๆกลับไปยังสถานะเริ่มต้น ทำให้แอดเดรสของอุปกรณ์ที่โดนรีเซ็ตกลับไปอยู่ที่ค่า 0 ซึ่งทำให้โฮสต์สามารถติดต่อกับอุปกรณ์ตัวใหม่นี้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นที่ 4 โฮสต์ร้องขอข้อมูลดิไวซ์ดิซคริปเตอร์จากอุปกรณ์

หลังจากที่แอดเดรสของอุปกรณ์ตัวใหม่ถูกรีเซ็ตเป็น 0 แล้ว โฮสต์จะเริ่มร้องขอข้อมูลดิซคริปเตอร์ต่างๆจากอุปกรณ์เพื่อนำค่าที่ได้มาพิจารณาว่าระบบสามารถรองรับอุปกรณ์ตัวใหม่นี้ได้หรือไม่ โดยดิซคริปเตอร์ตัวแรกที่จะถูกอ่านออกมาคือดีไวซ์ดิซคริปเตอร์ การอ่านดิซคริปเตอร์จากอุปกรณ์ก็คือการส่งคำสั่งร้องขอ เข้าไปยังตัวอุปกรณ์ ซึ่งตำแหน่งปลายทางที่จะคอยรับคำสั่งก็คือเอ็นพอยต์ควบคุม

ขั้นที่ 5 กำหนดแอดเดรสของอุปกรณ์

เมื่อทราบขนาดแพ็คเกจข้อมูลที่จะใช้ในการติดต่อจากการอ่านดีไวซ์ดิซคริปเตอร์แล้ว โฮสต์จะกำหนดแอดเดรสให้แก่ตัวอุปกรณ์ ซึ่งค่าแอดเดรสที่กำหนดขึ้นนี้จะไม่ซ้ำกับอุปกรณ์ตัวอื่นๆในระบบเลย

ขั้นที่ 6 โฮสต์อ่านดิซคริปเตอร์ต่างๆเพื่อพิจารณาการรองรับได้ของระบบ

ก่อนที่โฮสต์จะคอนฟิกกับอุปกรณ์ได้โฮสต์จะต้องอ่านดิซคริปเตอร์ต่างๆภายในตัวอุปกรณ์เพื่อพิจารณาว่าระบบมีทรัพยากรเพียงพอตามที่อุปกรณ์ต้องการหรือไม่ การพิจารณาจะเริ่มจากการที่โฮสต์ตรวจสอบดูว่าอุปกรณ์ตัวนั้นๆ มีโหมดการทำงานกี่โหมด หลังจากนั้นจะส่งคำสั่งร้องขอคอนฟิกิวเรชันดิซคริปเตอร์ไปยังอุปกรณ์ ด้วยคำสั่งนี้อุปกรณ์จะส่งคำสั่งดิซคริปเตอร์ที่เหลือทั้งหมดออกมา

เมื่อได้ข้อมูลทั้งหมดออกมาแล้วจะต้องนำมาตัดแยกออกเป็นแต่ละส่วนเองซึ่งเมื่อแยก คอนฟิกิวเรชันดิซคริปเตอร์ออกมาได้แล้วโฮสต์จะทราบว่าถ้าคอนฟิกให้อุปกรณ์ทำงานในแต่ละโหมดจะต้องจ่ายกระแสออกไปทั้งหมดเท่าใด และจะมีดิซคริปเตอร์ต่างๆที่บอกรายละเอียดของอุปกรณ์ออกมาทั้งหมด จากนั้นโฮสต์จะพิจารณาเปรียบเทียบกับทรัพยากรที่เหลือว่าสามารถรองรับการจ่ายไฟเลี้ยงและความเร็วในการรับส่งข้อมูลเพียงพอกับที่อุปกรณ์ต้องการหรือไม่หากสามารถรองรับได้ก็จะเข้าสู่การคอนฟิกขั้นต่อไป แต่ถ้าไม่สามารถรองรับได้โฮสต์จะพิจารณาอุปกรณ์ตัวอื่น และจะทำการหยุดการคอนฟิกเมื่อไม่สามารถรองรับได้อีกต่อไปแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นที่ 7 โฮสต์กำหนดเลือกคอนฟิกิวเรชันและอินเตอร์เฟซ

หลังจากพิจารณาแล้วว่าระบบสามารถรองรับความต้องการของอุปกรณ์ตัวใหม่ได้ โฮสต์จะส่งคำสั่งเพื่อระบุหมายเลขคอนฟิกิวเรชันและหมายเลขอินเตอร์เฟซหลังจากสิ้นสุดขั้นตอนนี้ อุปกรณ์จะดึงไฟเลี้ยงจากฮับเต็มพิกัดที่อุปกรณ์ระบุไว้ในคอนฟิกิวเรชันดิซคริปเตอร์และเริ่มทำงานตามหน้าที่ซึ่งถือเป็นการจบกระบวนการคอนฟิกอุปกรณ์

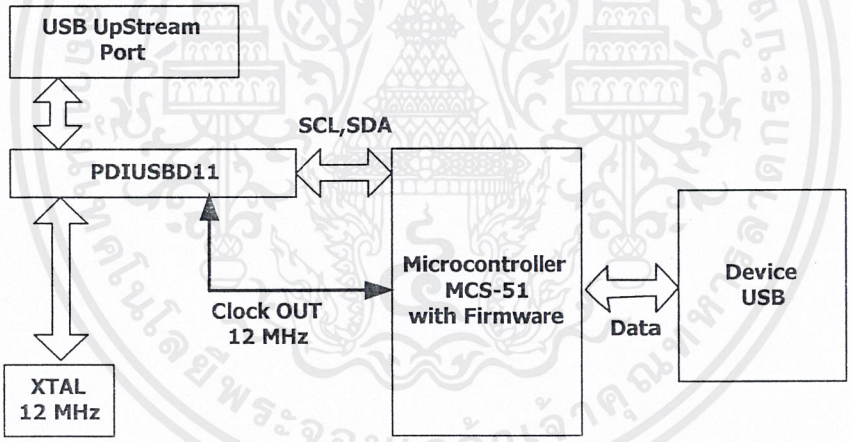
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การออกแบบทางฮาร์ดแวร์

ในการศึกษาการส่งผ่านข้อมูลผ่านพอร์ตยูเอสบีนั้น จะทำการศึกษาโดยการส่งข้อมูลไปมาระหว่างอุปกรณ์กับโฮสต์ โดยได้ทำการสร้างอุปกรณ์ออกมา 2 ชนิดคือ เมาส์ยูเอสบี เพื่อศึกษาการส่ง ข้อมูลจากอุปกรณ์ไปยังโฮสต์ และอีกอุปกรณ์หนึ่งคือ LED Display เพื่อศึกษาการส่งข้อมูลจากโฮสต์ไปยังอุปกรณ์เพื่อควบคุมการแสดงผล

5.1 วงจรในส่วนติดต่อกับโฮสต์ (HOST)

ในการติดต่อกับระบบยูเอสบี จำเป็นจะต้องมีอุปกรณ์สำหรับรับส่ง (transceiver) ในที่นี้เลือกใช้ไอซี PDIUSBD11 เนื่องจากมีราคาถูกและใช้งานได้ง่ายโดยมีคุณสมบัติดังนี้



รูปที่ 5.1.1 บล็อกไดอะแกรมของส่งผ่านข้อมูลระหว่างอุปกรณ์กับโฮสต์

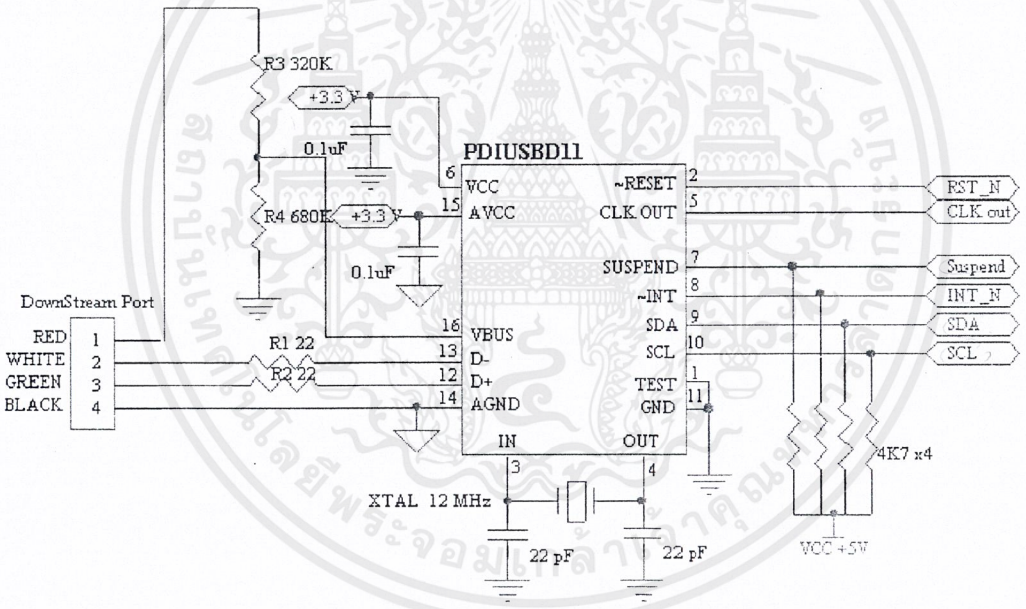
5.1.1 คุณสมบัติของไอซี PDIUSBD11

PDIUSBD11 เป็นอุปกรณ์ยูเอสบีความเร็วสูง เนื่องจากมี R พูลอัพ ที่ขา D+ อยู่ภายในตัวไอซีสามารถเซ็ตได้โดยกระบวนการทางซอฟต์แวร์ (SoftConnect™) ซึ่งเป็นเทคโนโลยีของบริษัทฟิลลิปป์ โดยมีการส่งข้อมูลแบบอนุกรมด้วยระบบบัส I2C ที่มีความเร็วสูงสุดถึง 1 เมกกะบิตต่อวินาที ซึ่งเป็นค่าตามทฤษฎีแต่ในทางปฏิบัติแล้วสามารถส่งได้สูงสุดเพียงแค่ 568 กิโลบิตต่อวินาที[5]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยสนับสนุนการส่งข้อมูลแบบ คอนโทรล, บัลต์ และอินเตอร์รัฟต์ แต่ไม่สนับสนุนการส่งข้อมูลแบบไอโซโครนัส

ถึงแม้ว่าไอซี PDIUSBD11 จะใช้ไฟเลี้ยง 3.3 โวท์ล แต่พอร์ตอินพุทและเอาท์พุทสามารถเชื่อมต่อกับระดับแรงดัน 5 โวท์ล ผ่าน R พูลอัพ (จะได้ระดับแรงดัน 0-5 โวท์ลซึ่งเป็นระดับแรงดันของไมโครคอนโทรลเลอร์)ได้เนื่องจากพอร์ตอินพุทและเอาท์พุทเป็นแบบ Open Drain ข้อดีอีกข้อหนึ่งของไอซีตัวนี้คือสามารถโปรแกรมสัญญาณนาฬิกาได้ถึง 48 เมกกะเฮิร์ต โดยใช้คริสตัล 12 เมกกะเฮิร์ต เนื่องจากมีวงจรเฟสล็อกลูปภายใน ซึ่งคริสตัล 12 เมกกะเฮิร์ต มีราคาที่ถูกกว่าคริสตัล 48 เมกกะเฮิร์ต และยังช่วยลดการแพร่กระจายสนามแม่เหล็กไฟฟ้า เนื่องจากธรรมชาติของสัญญาณไฟฟ้า เมื่อมีการเปลี่ยนแปลงระดับสัญญาณด้วยความเร็วมากๆ (การส่งสัญญาณที่ความเร็วสูงๆ) จะทำให้เกิดการแพร่กระจายของสนามแม่เหล็กและสนามไฟฟ้าออกมารอบๆสายส่งออกไปรบกวนระบบ อีกด้วย



รูปที่ 5.1.2 วงจรในส่วนที่ติดต่อกับโฮสต์

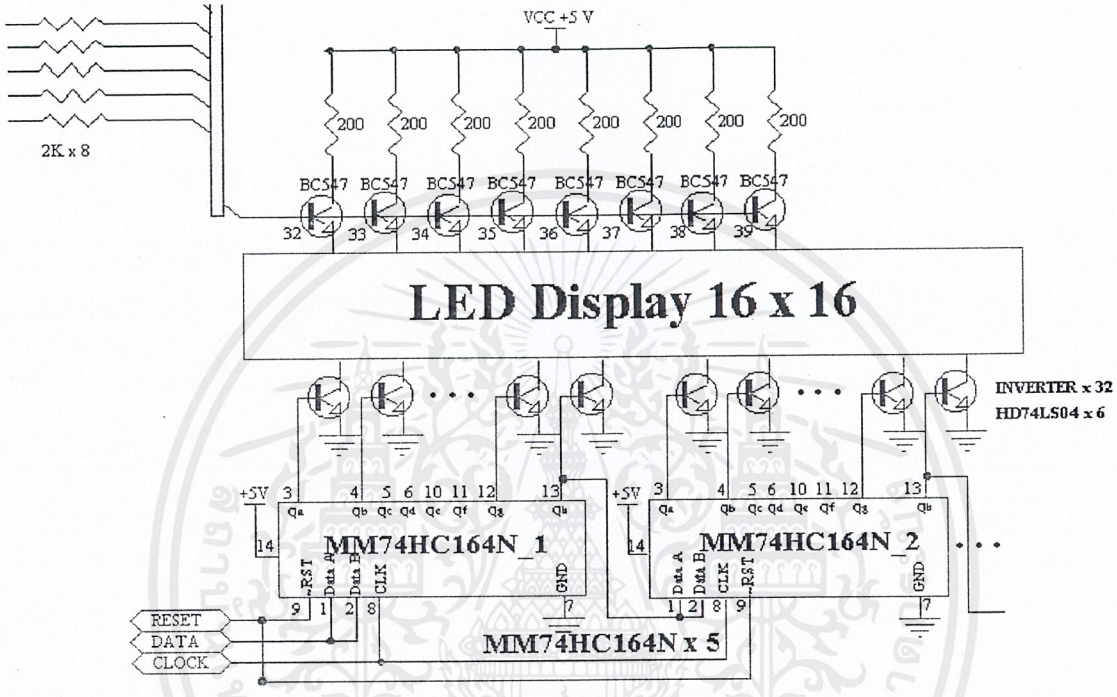
จากวงจรในรูปที่ 5.1.2 R1, R2 มีค่าอยู่ระหว่าง 22 – 44 โอห์มซึ่งมีค่าใกล้เคียงกับอิมพีแดนซ์ของสายยูเอสบี จะทำให้สามารถส่งผ่านกำลังงานได้สูงสุด

ถึงแม้ว่าขา VBUS ซึ่งเป็นขาที่โฮสต์ สามารถตรวจจับว่ามีอุปกรณ์มาเชื่อมต่อนั้นสามารถเชื่อมต่อกับแรงดัน 5 โวท์ลได้โดยตรงแต่เนื่องจาก PDIUSBD11 ใช้ไฟเลี้ยง 3.3 โวท์ล แต่เป็นการดีกว่าที่จะต่อ R3 และ R4 ซึ่งเป็นวงจรแบ่งแรงดันให้เหลือ 3.3 โวท์ล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

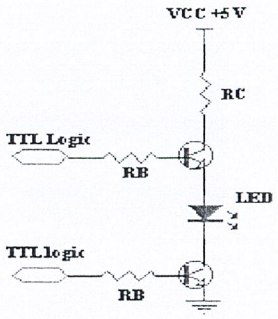
การใช้กำลังงานในสภาวะปกติของไอซี PDIUSBD11 จะดึงกระแสประมาณ 25 มิลลิแอมป์ และเมื่ออยู่ในโหมดประหยัดพลังงาน (Suspend) ดึงกระแสไม่เกิน 500 ไมโครแอมป์

5.2 วงจรในส่วนแสดงผล LED DOT MATRIX



รูปที่ 5.2.1 วงจรในส่วนแสดงผล LED DOT MATRIX

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



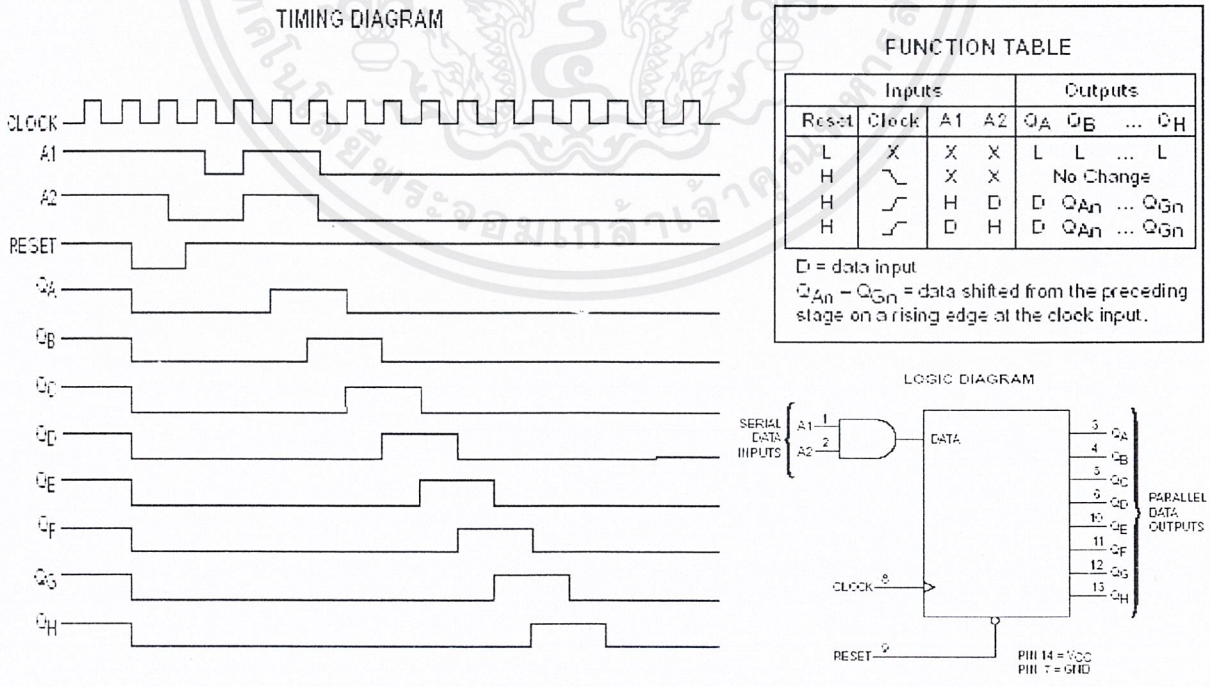
รูปที่ 5.2.2 วงจรขับ LED

ในการออกแบบวงจรแสดงผลตามรูปที่ 5.2.2 นั้นวงจรขับ LED นั้นจะออกแบบให้กระแสคอลเล็กเตอร์ไม่เกิน $20\ mA, \beta \le 200$

$$R_{CMIN} = \frac{V_{CC} - 2V_{CESAT} - V_{BE}}{I_{CMAX}}$$

$$= \frac{5 - 2(0.3) - 0.7}{20 \times 10^{-3}} = 185\ \Omega$$

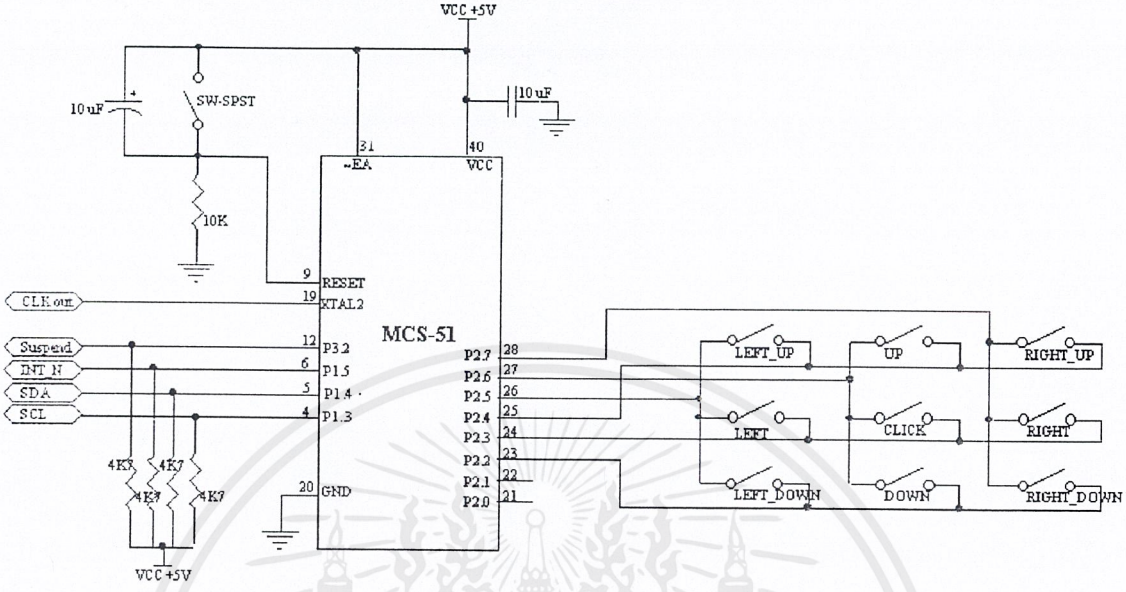
จากวงจรในรูปที่ 5.2.1 จะใช้ไอซี MM74HC164_N เป็นไอซี 8-bit Serial-Input/Parallel-Output Shift Register จะใช้ในการสแกน LED ในแต่ละหลักโดยไอซีตัวนี้สามารถสแกนได้สูงถึง 20 MHz ที่ไฟเลี้ยง 5 โวท์ล อุณหภูมิ <125 องศาเซลเซียส ซึ่งมี Timing Diagram ดังรูปที่ 5.2.3



รูปที่ 5.2.3 ฟังก์ชันการทำงานและไทมิ่งไดอะแกรมของไอซี 74HC164N

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3 วงจรในส่วนของเมาส์ยูเอสบี



รูปที่ 5.3.1 วงจรในส่วนของเมาส์ยูเอสบี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

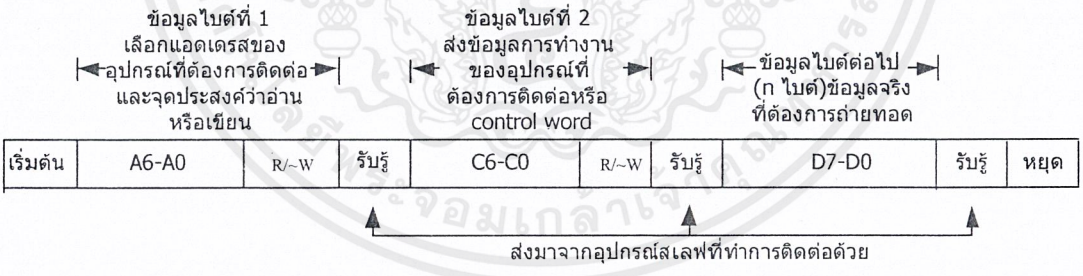
บทที่ 6
การออกแบบทางซอฟต์แวร์

การออกแบบทางซอฟต์แวร์นี้จะใช้ภาษา Visual Basic ซึ่งเป็นการเขียนโปรแกรมสำหรับเปิดภาพบิตแม็ป และโปรแกรมในส่วนติดต่อกับผู้ใช้งานเพื่อทำการแสดงผลของภาพบิตแม็ปขาวดำขนาดคงที่ 16x16 พิกเซลไปยัง LED DOT MATRIX ขนาด 16x16 โดยผ่านทางพอร์ตยูเอสบี และส่วนของภาษาแอสเซมบลีเป็นการเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ยูเอสบีทั้ง 2 อุปกรณ์ผ่านทางเอ็นพอยต์ขาเข้า (IN Endpoint) สำหรับอุปกรณ์เมาส์ และเอ็นพอยต์ขาออก (OUT Endpoint) สำหรับอุปกรณ์ LED Display

6.1 การเขียนโปรแกรมด้วยภาษา C

6.1.1 I2C โปรโตคอลของ PDIUSBD11

อุปกรณ์ที่ใช้บัส I2C โดยทั่วๆไปนั้นจะต้องมีแอดเดรสที่ไม่เหมือนใครอยู่แอดเดรสหนึ่ง เช่น ไอซี PCF8574A จะมีแอดเดรสคงที่ 0111 xxxy (x : ไดนามิกแอดเดรส, y : คำสั่งอ่านหรือเขียน) หมายความว่า ต่อไอซี PCF8574A บน I2C บัสได้ 8 ตัว (000 – 111)



รูปที่ 6.1.1 รูปแบบของข้อมูลอนุกรมบนบัส I2C โดยใช้การอ้างถึงแบบ 7 บิต

แต่แอดเดรส I2C ของไอซี PDIUSBD11 จะแตกต่างออกไปกล่าวคือ จะมีอยู่ 2 แอดเดรสในไอซีตัวเดียวกันนั่นคือ แอดเดรสสำหรับคำสั่งหนึ่งแอดเดรส และ แอดเดรสสำหรับอ่านหรือเขียนข้อมูลอีกหนึ่งแอดเดรส ดังตารางที่ 6.1.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังก์ชั่น	I2C แอดเดรส	LSBit	แอดเดรสจริง
Command Write	0011 011	0	0x36
Data Write	0011 010	0	0x34
Data Read	0011 010	1	0x35

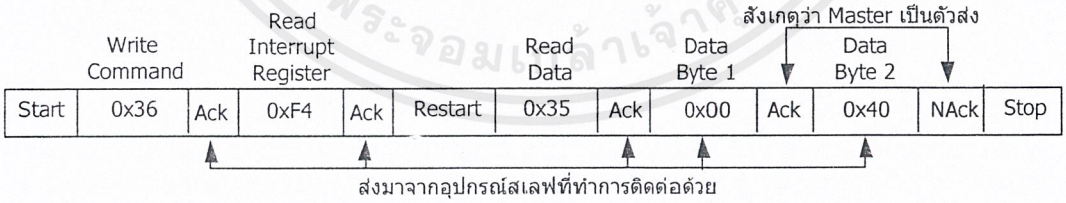
ตารางที่ 6.1.1 แอดเดรส I2C ของไอซี PDIUSBD11

ยกตัวอย่างเช่น ถ้าอยากเรียกใช้ฟังก์ชันภายในไอซี (Embedded function) ที่เกี่ยวกับฮับ (0xD0) และเขียนข้อมูลเพื่อทำการปิดการทำงานของฮับ หมายความว่าไม่ได้ใช้อุปกรณ์ตัวนี้เป็น ฮับ ด้วยค่า (0x00) ดังนั้นสามารถจัดเรียงข้อมูลได้ดังรูปที่ 6.1.2



รูปที่ 6.1.2 รูปแบบของข้อมูลอนุกรมบนบัส I2C ด้วยคำสั่งเขียนของ PDIUSBD11

เปรียบเทียบรูปที่ 6.1.1 กับรูปที่ 6.1.2 แตกต่างกันตรงที่ตัวอุปกรณ์ที่เป็นมาสเตอร์ สามารถเขียนแอดเดรสคำสั่งแทรกลงไปได้เลยโดยไม่ต้องรอให้ถึงสภาวะหยุด (สังเกตตรง Restart) จะเห็นว่าคำสั่งเขียนไปยังตัวไอซี PDIUSBD11 นั้นยังไม่ยุ่งยากมากนักเมื่อเทียบกับคำสั่งในการอ่าน ดังรูปที่ 6.1.3



รูปที่ 6.1.3 รูปแบบของข้อมูลอนุกรมบนบัส I2C ด้วยคำสั่งอ่านของ PDIUSBD11

จากรูปที่ 6.1.3 ความยุ่งยากจากคำสั่งอยู่ตรงที่เมื่อต้องการอ่านค่าหลายๆไบต์เราไม่สามารถจะบอกได้ว่าเมื่อไหร่ที่จะปล่อยให้บัสว่าง ดังนั้นทุกๆไบต์ข้อมูลที่จะรับข้อมูลมานั้นอุปกรณ์ที่เป็นมาสเตอร์ จะต้องส่งบิตรับรู้ (Acknowledge bit) ทุกๆไบต์จนกระทั่งถึงไบต์สุดท้ายอุปกรณ์ที่เป็นมาสเตอร์ จึงส่งบิตปฏิเสธที่จะรับข้อมูล (Negative Acknowledge bit) เพื่อเป็นการบอกอุปกรณ์ที่เป็นสเลฟ (Slave) ว่าสามารถอ่านข้อมูลได้ครบแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เนื่องจากไอซี PDIUSBD11 นั้นมีฟังก์ชันที่เกี่ยวกับฮับถ้าเราจะใช้ไอซีตัวนี้เป็นอุปกรณ์ที่ไม่ใช่ฮับ เราจะต้องทำการปิดการทำโดยของฟังก์ชันนี้โดยการ โดยการเรียกใช้ฟังก์ชัน SET_HUB_ADDRESS(0xD0) จากนั้นทำการปิดการทำงานโดยเขียนข้อมูล (0x00) ลงไปที่ไอซี PDIUSBD11 แต่ทุกครั้งที่เกิดบัสรีเซ็ต (BUS RESET) ฟังก์ชั่นของฮับจะเปิดการทำงานทุกครั้ง ดังนั้นเพื่อที่จะทำการปิดการทำงานของฮับและเปิดการทำงานของฟังก์ชันต่างๆภายในไอซี PDIUSBD11 เราจึงต้องทำตามขั้นตอนดังนี้

/* Disable Hub Function in PDIUSBD11 */

SET_HUB_ADDRESS(0xD0) to 0x00

/* Set Address to zero (default) and enable function */

SET_ADDRESS_ENABLE(0xD1) to 0x80

/* Enable function generic endpoints */

SET_ENDPOINT_ENABLE(0xD8) to 0x02

/* Set Mode - Enable SoftConnect */

SET_MODE(0xF3) to 0x97

/*Embedded Function, SoftConnect, Clk Run, No LazyClk, Remote Wakeup */

BYTE 2 to 0x03

/* CLKOut = 12 MHz */

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.2 ฟังก์ชันต่างๆภายในไอซี PDIUSBD11

จากที่กล่าวไปแล้วข้างต้นได้อ้างถึงคำสั่งบางคำสั่งไปพอสมควรตอนนี้เราจะมาดูกันว่าฟังก์ชั่นทั้งหมดภายในไอซี PDIUSBD11 นั้นมีอะไรบ้าง

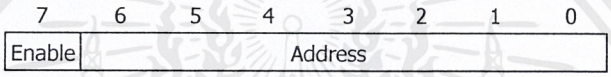
6.1.2ก ฟังก์ชัน Set Address Enable

Command : *0xD0 (Set Hub Address)*

0xD1 (Set Embedded Function)

Data : เขียนข้อมูล *1 ไบต์* จะเขียนตามคำสั่งCommand ไปยัง PDIUSBD11 เพื่อทำการเปิดการใช้งานที่แอดเดรสที่ต้องการ

Data Byte1



คำสั่งนี้เป็นการเปิดการใช้งานฟังก์ชันภายในตัวไอซี PDIUSBD11 โดยสามารถเปิดการทำงานได้โดยเซ็ตบิตที่ 7 เป็น “1” , ส่วนบิตที่ 6-0 นั้นจะใช้ในการเซ็ตแอดเดรสของฟังก์ชัน

ตัวอย่างการใช้งาน

```
/* Disable Hub */
void DisOtherAddress(void)
{
        unsigned char dummy;
        dummy = 0;
        Wr_command(0x0D0); //SetAddr Command
        Wr_Ndata(&dummy, 1); //SetAddr Byte
}
```

จากตัวอย่างเป็นการปิดการทำงานของฮับ โดยที่ *void DisOtherAddress(void)* เป็นโปรแกรมย่อยที่จะบรรจุคำสั่งต่างๆลงในบัส I2C ตามรูปที่ 6.1.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.2ข ฟังก์ชัน Set Endpoint Enable

Command : *0xD8 (Set Endpoint Enable)*

Data : เขียน *1 ไบต์* จะเขียนตามคำสั่งCommand ไปยัง PDIUSBD11 โดยข้อมูลที่เขียนไปมีรูปแบบดังนี้

Data Byte1

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Generic Endpoint Enable	Reserved

หลังจากเราเปิดการทำงานของฟังก์ชันโดยการเรียกใช้ *ฟังก์ชัน Set Address Enable* จะทำให้เราสามารถใช้งาน เอ็นพอยต์อินเด็กซ์ 2 และ เอ็นพอยต์อินเด็กซ์ 3 ซึ่งเป็นคอนโทรลเอ็นพอยต์

ในขณะที่ทำกระบวนการคอนฟิกอุปกรณ์นั้นเองตัวอุปกรณ์ยูเอสบี ของเราจะบอกกับโฮสต์ว่าจะใช้เอ็นพอยต์ ไหนในการสื่อสารระหว่างโฮสต์กับอุปกรณ์ โดยการเซ็ตบิตที่ 1 เป็น "1" ส่วนบิตอื่นๆ เซ็ตเป็น "0" เพื่อบอกกับโฮสต์ ผ่านทางเอ็นพอยต์ดิซคริปเตอร์ ซึ่งไอซี PDIUSBD11 ตัวนี้มีให้เลือกใช้ 3 เอ็นพอยต์ ซึ่งมีอินเด็กซ์แตกต่างกันไปดังตารางที่ 6.1.2

Endpoint	Endpoint Index	Endpoint Type	Direction
0	2	Control/Default	OUT
	3		IN
1	5	Generic	OUT
	4		IN
2	6	Generic	OUT
	7		IN
3	8	Generic	OUT
	9		IN

ตารางที่ 6.1.2 เอ็นพอยต์ของไอซี PDIUSBD11

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.2ค ฟังก์ชัน Set Mode

Command : *0xF3 (Set Mode)*

Data : เขียน *2 ไบต์* จะเขียนตามคำสั่งCommand ไปยัง PDIUSBD11

Data Byte1

7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	1
Embedded Function Mode	Reserved	Reserved	Soft Connect	Debug Mode	Clock Running	No Lazy Clock	Remote Wakeup

Remote Wakeup : เมื่อเซ็ตเป็น "1" จะเป็นการปลุกให้ไอซีทำงานหลังจากอยู่ในโหมดประหยัดพลังงาน และเมื่อเกิดสภาวะบัสรีเซ็ต จะเป็นการเปิดการใช้งานฟังก์ชันนี้อัตโนมัติ

No Lazy Clock & Clock Running : ความหมายของสองบิตนี้เป็นดังตารางที่ 6.1.3

Debug Mode : เมื่อเซ็ตเป็น "1" สามารถจะทราบว่าความผิดพลาดที่เกิดขึ้นนั้เกิดจากอะไร

Soft Connet : เมื่อเซ็ตเป็น "1" จะเป็นการเชื่อมต่อ R-pullup ที่ขา D+ เข้ากับไฟเลี้ยงโดยจะทำงานในโหมดความเร็วสูง

Embedded Function Mode : เซ็ตเป็น "1" ในสภาวะปกติ

บิตที่ 6-5 : ไม่ใช้งานเซ็ตเป็น "0"

No Lazy Clock	Clock Running	การทำงาน	กระแสที่ใช้
0	0	อยู่ในโหมด LazyClock คือจะมี Clk out ที่ขา 5 ออกมา 30 KHz หลังจากขา suspend มีค่า "High"คริสตัลและ PLL หยุดทำงาน	~2.5 mA
0	1	อยู่ในโหมด LazyClock คือจะมี Clk out ที่ขา 5 ออกมา 30 KHz หลังจากขา suspend มีค่า "High" แต่คริสตัลและ PLL ยังคงทำงาน	~25 mA
1	0	คริสตัลและ PLL หยุดทำงาน และไม่มี Clk out ที่ขา 5	~210 uA
1	1	มี Clk out ที่ขา 5 ออกมาตามที่ได้โปรแกรมไว้ คริสตัลและ PLL ยังคงทำงาน	~25 mA

ตารางที่ 6.1.3 ความหมายของ No Lazy Clock และ Clock running

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Data Byte2

7	6	5	4	3	2	1	0
x	x	0	0	0	0	1	1
x	x	Clock Divisor					

ไบต์ที่สองนี้จะใช้ในการโปรแกรม Clk out ที่ขา 5 โดย Clk out ที่ได้เท่ากับ

$$Clk\ out = \frac{48\ MHz}{N+1}$$

โดยที่ N กำหนดจาก บิตที่ 5-0 เช่นต้องการ Clk out 12 MHz จะต้องเซ็ตเป็น " 00000003 "

ตัวอย่างการใช้งาน

```
/* Set Mode Function */
void SetMode(unsigned char D11_MDATA, unsigned char D11_CLKDIV)
{
    Wr_command(0x0F3); //SetMode Command
    temp_buffer[0] = D11_MDATA; //SetMode, Configuration Byte
    temp_buffer[1] = D11_CLKDIV; //SetMode, Configuration Byte
    Wr_Ndata(temp_buffer, 2);
}
```

จากตัวอย่างเป็นเรียกใช้ฟังก์ชัน Set Mode ในการเรียกใช้ SoftConnet™ และใช้ ClkOut 12 เมกกะเฮิร์ต ส่วน *void SetMode(unsigned char D11_MDATA, unsigned char D11_CLKDIV)* เป็นการจัดรูปแบบข้อมูลลงบนบัส I2C ตามรูปที่ 6.1.2 แต่ใช้ Data 2 ไบต์

6.1.2ง ฟังก์ชัน Read Interrupt Register

Command : *0xF4 (Read Interrupt Register)*

Data : อ่าน *2 ไบต์* หลังจากที่เขียนคำสั่ง *0xF4* ไปยังไอซี PDIUSBD11 แล้วจะได้ข้อมูลมาเก็บที่บัฟเฟอร์ของ PDIUSBD11 จำนวน 2 ไบต์ซึ่งมีความหมายดังนี้

Data Byte1

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
Endpoint Index 7	Endpoint Index 6	Endpoint Index 5	Endpoint Index 4	Control IN Endpoint	Control OUT Endpoint	Reserved (HUB)	Reserved (HUB)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Data Byte2

0	0	0	0	0	0	0	0
Reserved	Bus Reset	Reserved	Reserved	Reserved	Reserved	Endpoint Index 8	Endpoint Index 9

เมื่อเกิดอินเทอร์รัฟต์ขึ้นเราสามารถจะรู้ได้ว่าที่มาของการอินเทอร์รัฟต์นั้นมาจากที่ไหนโดยเขียนคำสั่ง Read Interrupt Register ไปที่ไอซี PDIUSBD11 โดยที่สามารถเคลียร์ค่าในแต่ละบิตได้โดย เรียกใช้ *ฟังก์ชัน Read Last Transaction Status*

6.1.2จ ฟังก์ชัน Select Endpoint

Command : *0x02 (Select Control Out Endpoint), 0x03(Select Control In Endpoint)*
0x04 (Select Generic Endpoint1 IN), 0x05 (Select Generic Endpoint1 OUT)
0x06 (Select Generic Endpoint2 OUT), 0x07 (Select Generic Endpoint2 IN)
0x08 (Select Generic Endpoint3 OUT , 0x09 (Select Generic Endpoint3 IN)

Data : อ่าน *1 ไบต์*

Data Byte 1

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Buffer Full

ฟังก์ชันนี้ใช้ในการเลือกเอ็นพอยต์ ที่ต้องการใช้งานโดยสามารถดูว่าเอ็นพอยต์ ที่เลือกนั้นพร้อมใช้งานหรือไม่โดยดูที่บิตที่ 0 จากค่าที่อ่านได้ถ้าเป็น "1" หมายความว่าเอ็นพอยต์ ที่เลือกนั้นยังไม่พร้อมใช้งานและเราสามารถดูข้อมูลได้มากกว่านี้โดยเรียก *ฟังก์ชัน Read Endpoint Status*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.3ฉ ฟังก์ชัน Read Last Transaction Status

Command : *0x42 (Read Last Transaction Status for the Control Out Endpoint)*

0x43(Read Last Transaction Status for the Control In Endpoint)

0x44 (Read Last Transaction Status for the Generic Endpoint1 IN)

0x45 (Read Last Transaction Status for the Generic Endpoint1 OUT)

*0x46 (*Read Last Transaction Status for the *Generic Endpoint2 OUT)*

*0x47 (*Read Last Transaction Status for the *Generic Endpoint2 IN)*

*0x48 (*Read Last Transaction Status for the *Generic Endpoint3 OUT)*

*0x49 (*Read Last Transaction Status for the *Generic Endpoint3 IN)*

Data : อ่าน *1 ไบต์*

Data Byte1

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
Previous Status not Read	Data 0/1 Packet	Setup Packet	Error Code				Data Receive Transmit Success

โดยฟังก์ชันนี้ใช้ในการตรวจสอบสภาวะต่างๆบนบัส โดยที่หากมีข้อผิดพลาดเกิดขึ้น สามารถดูได้ว่ามีสาเหตุมาจากอะไรซึ่ง Error Code เป็นไปตามตารางที่ 6.1.4

Code	Error
0000	No Error
0001	PID Encoding Error
0010	PID Unknown
0011	Unexpected Packet
0100	Token CRC Error
0101	Data CRC Error
0110	Time Out Error
0111	Babble Error
1000	Unexpected End of Packet
1001	Sent or Received NAK
1010	Sent Stall
1011	Overflow Error
1101	BitStuff Error
1111	Wrong Data PID

ตารางที่ 6.1.4 ความหมายของ Code Error

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.2ช ฟังก์ชัน Set Endpoint Status

Command : *0x42 (set Control Out Endpoint Status)*

0x43(set Control In Endpoint Status)

0x44 (set Generic Endpoint1 IN Status)

0x45 (set Generic Endpoint1 OUT Status)

*0x46 (*set *Generic Endpoint2 OUT Status)*

*0x47 (*set *Generic Endpoint2 IN Status)*

*0x48 (*set *Generic Endpoint3 OUT Status)*

*0x49 (*set *Generic Endpoint3 IN Status)*

Data : เขียน *1 Byte*

Data Byte1

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Stalled

สังเกตุเห็นว่าคำสั่ง Command จะเหมือนกับ *ฟังก์ชัน Read Last Transaction Status* แตกต่างกันเพียงแต่เป็น การเขียนข้อมูล 1 ไบต์ โดยคำสั่งนี้ใช้สำหรับทำให้เอ็นพอยต์อยู่ในสภาวะ Stall เพื่อที่จะไม่รับ แพ็คเกจข้อมูลใดๆ ยกเว้น แพ็คเกจ Setup นั่นหมายความว่าเราจะให้เอ็นพอยต์ อยู่ในสภาวะ Un-Stall ได้โดยใช้ แพ็คเกจ Setup

6.1.2ซ ฟังก์ชัน Read Endpoint Status

Command : *0x82 (Read Control OUT Endpoint Status)*

0x83(Read Control IN Endpoint Status)

0x84 (Read Generic Endpoint1 IN Status)

0x85 (Read Generic Endpoint1 OUT Status)

*0x86 (*Read *Generic Endpoint2 OUT* Status*)*

*0x87 (*Read *Generic Endpoint2 IN* Status*)*

*0x88 (*Read *Generic Endpoint3 OUT* Status*)*

0x89 (Read Generic Endpoint3 IN Status)

Data : อ่าน *1 Byte*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Data Byte1

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
Reserved	Reserved	Buffer Full	Data 0/1 Packet	Stalled	Setup Packet	Reserved	Reserved

ฟังก์ชันนี้ใช้สำหรับการดีบัก (Debug) เช่นกัน

6.1.2ฌ ฟังก์ชัน Read/Write Buffer

Command : *0xF0 (Read/Write Buffer)*

Data : *อ่าน 1 – 8 ไบต์*

เขียน 1 – 8 ไบต์

ข้อมูลที่อยู่ในบัฟเฟอร์จะมีรูปแบบดังนี้

Offset 0	Offset 1	Offset 2	Offset 3	...	Offset 9
Reserved Undefined Value	Number of bytes to follow	Byte 1	Byte 2	...	Byte 8

สมมติว่าใช้คำสั่งอ่าน 8 ไบต์ (บัฟเฟอร์ของ PDIUSBD11 มีขนาด 8 ไบต์) ข้อมูลจริงๆจะอยู่ที่ Offset 2 เป็นต้นไป

6.1.2ญ ฟังก์ชัน Clear Buffer

Command : *0xF2*

Data : *ไม่มี*

เมื่อการรับแพ็คเกจ ข้อมูลใดๆ เรียบร้อยแล้ว แฟลกที่แจ้งสภาวะบัฟเฟอร์เต็มจะถูกเซ็ต ด้วยเหตุนี้เองจึงต้องสั่งเคลียร์บัฟเฟอร์เพื่อที่จะสามารถรับแพ็คเกจ ข้อมูลถัดไปได้

6.1.2ฎ ฟังก์ชัน Validate Buffer

Command : *0xFA*

Data : *ไม่มี*

ในครั้งแรกเลยที่เราเขียนข้อมูลไปที่บัฟเฟอร์ เราควรจะใช้ *ฟังก์ชัน Validate Buffer* เพื่อบอกให้ PDIUSBD11 รู้ว่ามีข้อมูลอยู่ในบัฟเฟอร์เรียบร้อยแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.2ฎ ฟังก์ชัน Acknowledge Setup

Command : *0xF1*

Data : *ไม่มี*

เมื่อ PDIUSBD11 ได้รับแพ็คเกจ Setup เรียบร้อยแล้วมันจะทำการเคลียร์บัฟเฟอร์ของคอนโทรลเอ็นพอยต์ขาเข้า (Control In Endpoint Buffer) และปิดการทำงานของคำสั่ง *Validate Buffer และ Clear Buffer* จนกว่าแพ็คเกจ นั้นจะได้รับการรับรู้ โดยการส่งคำสั่ง *Acknowledge Setup* ไปยังเอ็นพอยต์ขาเข้าและเอ็นพอยต์ขาออก

6.1.2ฐ ฟังก์ชัน Send Resume

Command : *0xF6*

Data : *ไม่มี*

เมื่อโฮสต์ อยู่ในโหมดประหยัดพลังงานแล้วเราต้องการให้โฮสต์ กลับมาทำงานในโหมดปกติเราจะต้องทำการส่งคำสั่ง *Send Resume* ไปยังโฮสต์ เพื่อทำการปลุกโฮสต์

6.1.2ฑ ฟังก์ชัน Read Current Frame Number

Command : *0xF5*

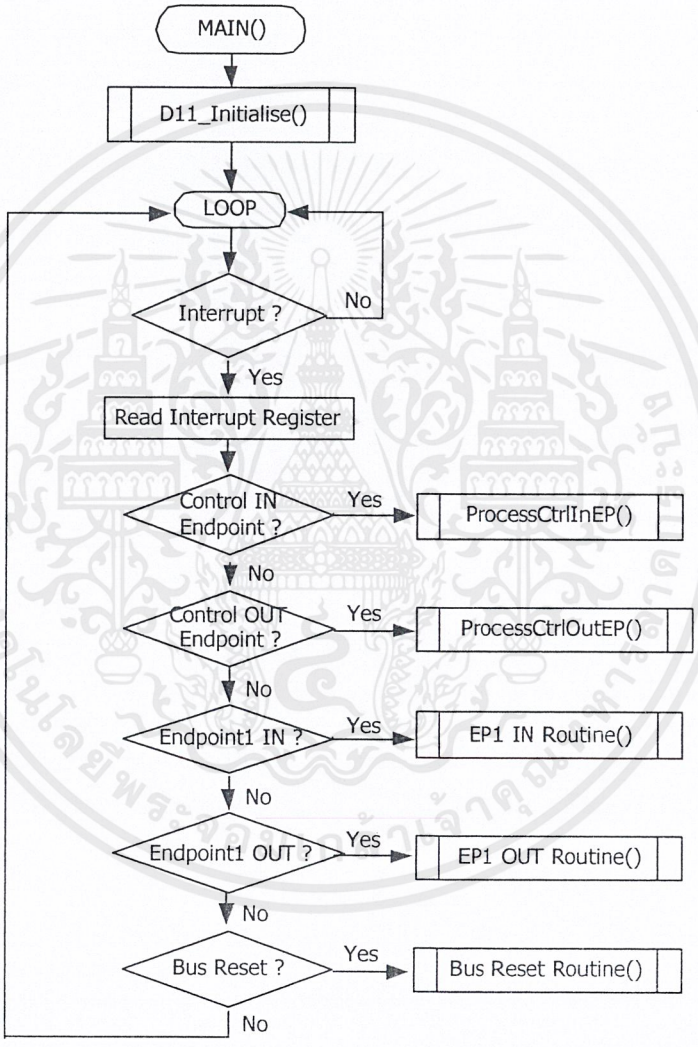
Data : *อ่าน 1-2 ไบต์*

ฟังก์ชันนี้ใช้สำหรับส่งหมายเลขเฟรมปัจจุบันขนาด 16 บิตกลับไปหลังจากได้รับแพ็คเกจเริ่มต้น แล้วโดยจะส่ง LSByte ก่อน แล้วตามด้วย MSByte

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

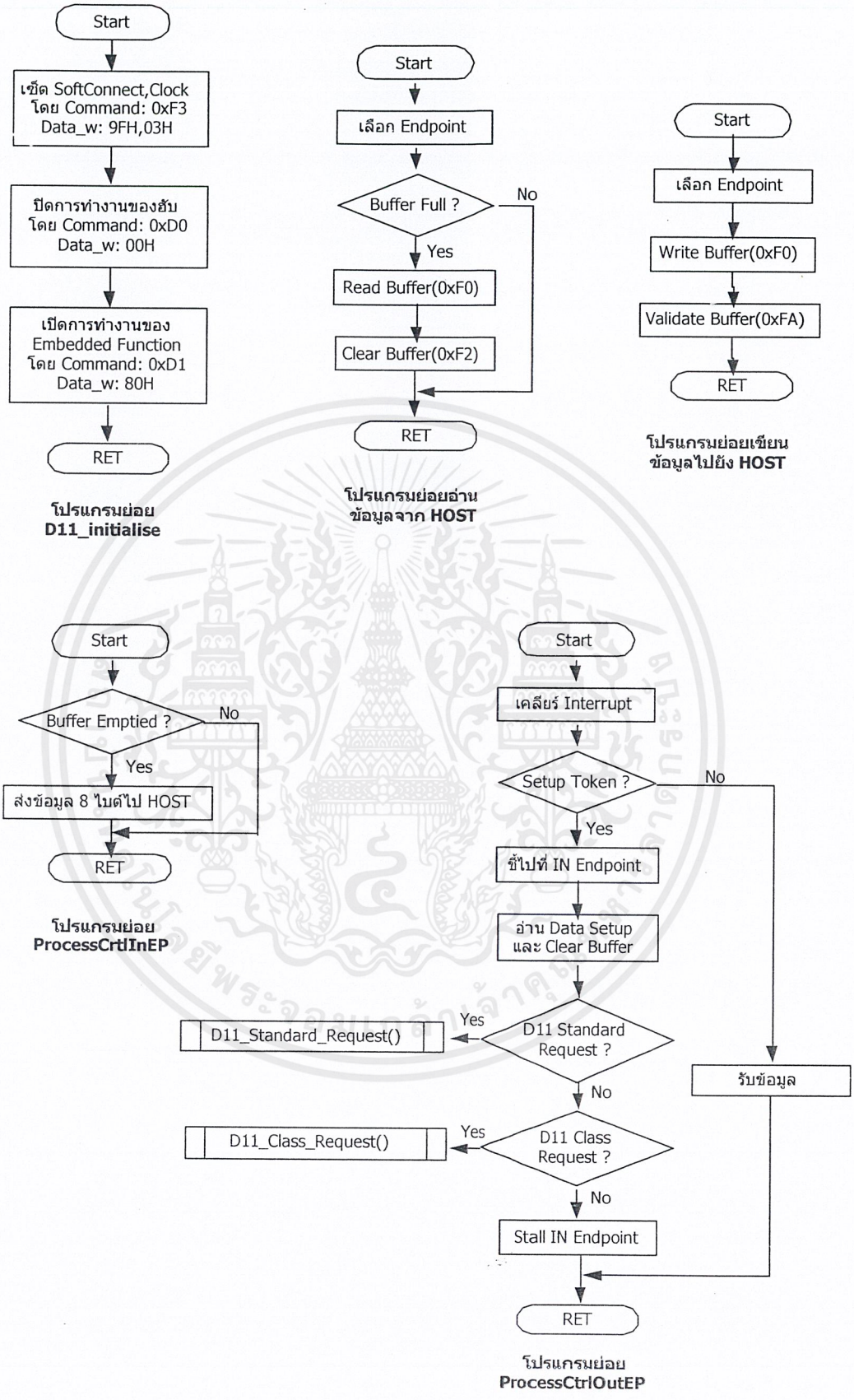
6.1.3 คอนโทรลโฟล์วของดีไวซ์ไดร์เวอร์

โปรแกรมหลักจะคอยตรวจสอบว่าเกิดอินเตอร์รัฟต์หรือไม่ ถ้าเกิดจะตรวจสอบว่าอินเตอร์ที่เกิดขึ้นนั้นมีสาเหตุมาจากเอ็นพอยต์ขาเข้า หรือเอ็นพอยต์ขาออก จากนั้นก็จะทำการบริการอินเตอร์รัฟต์ หากไม่เกิดอินเตอร์รัฟต์หรือไม่สามารถคอนฟิกอุปกรณ์ได้ก็จะทำการรีเซ็ตบัส แล้วกลับมาวนตรวจสอบการอินเตอร์รัฟต์ไปเรื่อยๆ



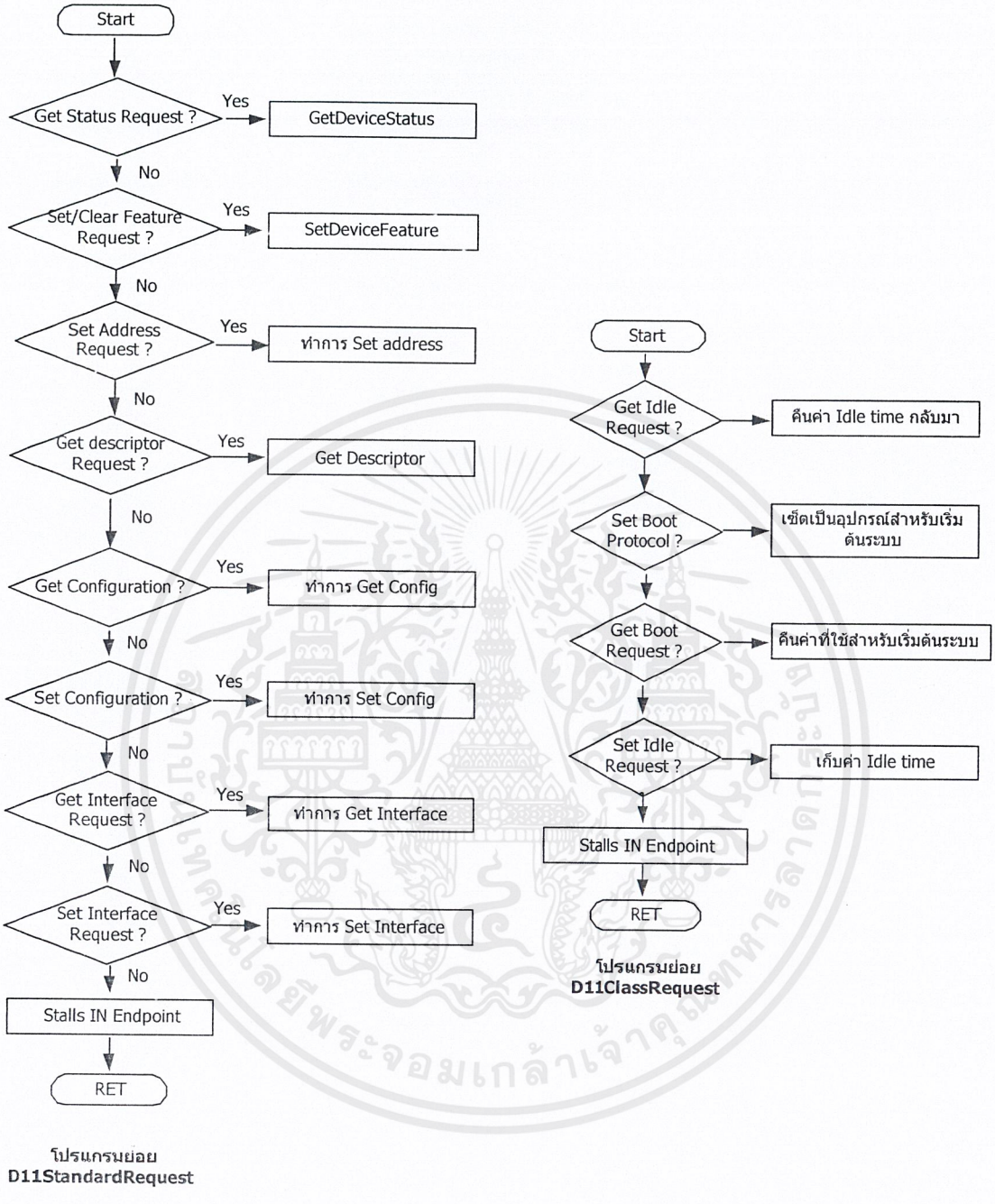
รูปที่ 6.1.4 โฟล์วคอนโทรลของโปรแกรมหลัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



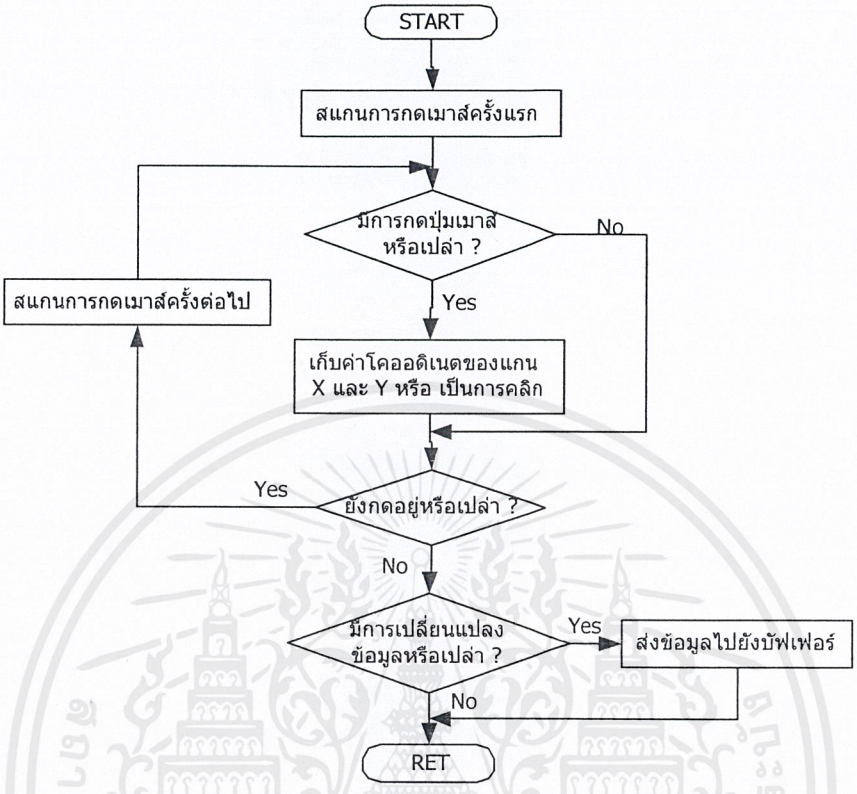
รูปที่ 6.1.5 โฟล์วคอนโทรลสำหรับโปรแกรมย่อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.1.6 โฟล์วคอนโทรลสำหรับ StandardRequest และ ClassRequest

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.1.7 โฟล์วคอนโทรลสำหรับสแกนการกดเมาส์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.4 ดีไวซ์ดิสคริปเตอร์ของเมาส์ยูเอสบี

```
struct DEVICE code sDevice= { sizeof( struct DEVICE),//BYTE bLength
                              0x01,  //BYTE bDescriptorType
                              0x0a01,//WORD bcdUSB
                              0xff,  //Byte bDeviceClass
                              0xff,  //BYTE bDeviceSubClass
                              0xff,  //BYTE bDeviceProtocol
                              0x08,  //BYTE bMaxPacketSize of 8
                              0x7104,//WORD idVendor
                              0x0211,//WORD idProduct, For Philips Hub mouse
                              0x0a01,//WORD bcdDevice
                              0x01,  //BYTE iManufacturer
                              0x02,  //BYTE iProduct
                              0x00,  //BYTE iSerialNumber
                              0x01 };//BYTE bNumConfigurations

struct CONFIG code sConfiguration= {sizeof(struct CONFIGURATION), //BYTE bLength
                              0x02, //BYTE bDescriptorType //Assigned by USB
                              sizeof( struct CONFIG), //BYTE wTotalLength
                              0x00, //BYTE wZero, always 00
                              0x01, //BYTE bNumInterfaces
                              0x01, //BYTE bConfigurationValue
                              0x00, //BYTE iConfiguration
                              0x0a0, //BYTE bmAttributes, Bus powered and remote wakeup
                              0x05, //BYTE MaxPower
                        sizeof(struct INTERFACE), //BYTE bLength
                              0x04, //BYTE bDescriptionType, assigned by USB
                              0x00, //BYTE bInterfaceNumber
                              0x00, //BYTE bAlternateSetting
                              0x01, //BYTE bNumEndpoints, uses 1 endpoints
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                0x03, //BYTE bInterfaceClass, HID Class - 0x03
                0x01, //BYTE bInterfaceSubClass
                0x01, //BYTE bInterfaceProtocol
                0x00 ,//BYTE iInterface
            sizeof(struct HIDDESC),//BYTE bLength;
                0x21, //BYTE bDescriptorType;
                0x0001,//WORD bcdHID;
                0x00,//BYTE bCountryCode;
                0x01,//BYTE bNumDescriptors;
                0x22,//BYTE bReportDescriptorType;
                0x3200,//WORD wItemLength;
          sizeof(struct ENDPOINT), //BYTE bLength
                0x05, //BYTE bDescriptorType, assigned by USB
                0x81, //BYTE bEndpointAddress, IN endpoint, endpoint 1
                0x03, //BYTE bmAttributes, Interrupt endpoint
                0x0800, //WORD wMaxPacketSize
                0x0A, //Polling Time
                }; //Interval

unsigned char code REPORT_DESCRIPTOR[50] =  /* 50 Bytes*/
                        {0x05, 0x01, 0x09, 0x02, 0xA1, 0x01, 0x09, 0x01,
                        0xA1, 0x00, 0x05, 0x09, 0x19, 0x01, 0x29, 0x03,
                        0x15, 0x00, 0x25, 0x01, 0x95, 0x03, 0x75, 0x01,
                        0x81, 0x02, 0x95, 0x01, 0x75, 0x05, 0x81, 0x01,
                        0x05, 0x01, 0x09, 0x30, 0x09, 0x31, 0x15, 0x81,
                        0x25, 0x7F, 0x75, 0x08, 0x95, 0x02, 0x81, 0x06,
                        0xC0, 0xC0,};
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unsigned char code LANG_ID[4] = { 0x04,  //4 bytes
                                  0x03,  //bDescriptorType = String Descriptor
                                  0x09,  //Lang ID for English = 0x0409
                                  0x04,};

#define SIZE_STR1 98
#define SIZE_STR2 18
#define SIZE_STR3 6

struct STRING1 { BYTE bLength;
                 BYTE bDescriptorType;
                 char string1[SIZE_STR1];
                 };

struct STRING1 code sSTRING1 = {SIZE_STR1+2, //Size of String
                                0x03, //STRING Descriptor Type
                                {'K', 00, 'i', 00, 'n', 00, 'g', 00, ' ', 00,
                                 'M', 00, 'o', 00, 'n', 00, 'g', 00, 'k', 00,
                                 'u', 00, 't', 00, '-', 00, 's', 00, ' ', 00,
                                 'I', 00, 'n', 00, 's', 00, 't', 00, 'i', 00,
                                 't', 00, 'u', 00, 't', 00, 'e', 00, ' ', 00,
                                 'o', 00, 'f', 00, ' ', 00, 'T', 00, 'e', 00,
                                 'c', 00, 'h', 00, 'n', 00, 'o', 00, 'l', 00,
                                 'o', 00, 'g', 00, 'y', 00, ' ', 00, 'L', 00,
                                 'a', 00, 'd', 00, 'k', 00, 'r', 00, 'a', 00,
                                 'b', 00, 'a', 00, 'n', 00, 'g', 00}};

struct STRING2 { BYTE bLength;
                 BYTE bDescriptorType;
                 char string2[SIZE_STR2];
                 };
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
struct STRING2 code sSTRING2 = {SIZE_STR2+2, //Size of String
                                0x03, //STRING Descriptor Type
                                {'M', 00, 'o', 00, 'u', 00, 's', 00, 'e', 00,
                                ' ', 00, 'U', 00, 'S', 00, 'B', 00}};

struct STRING3 { BYTE bLength;
                 BYTE bDescriptorType;
                 char string3[SIZE_STR3];
                 };

struct STRING3 code sSTRING3 = {SIZE_STR3+2, //Size of String
                                0x03, //STRING Descriptor Type
                                {'X', 00, 'W', 00, 'L', 00}};
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2 การเขียนโปรแกรมด้วยภาษา Visual Basic

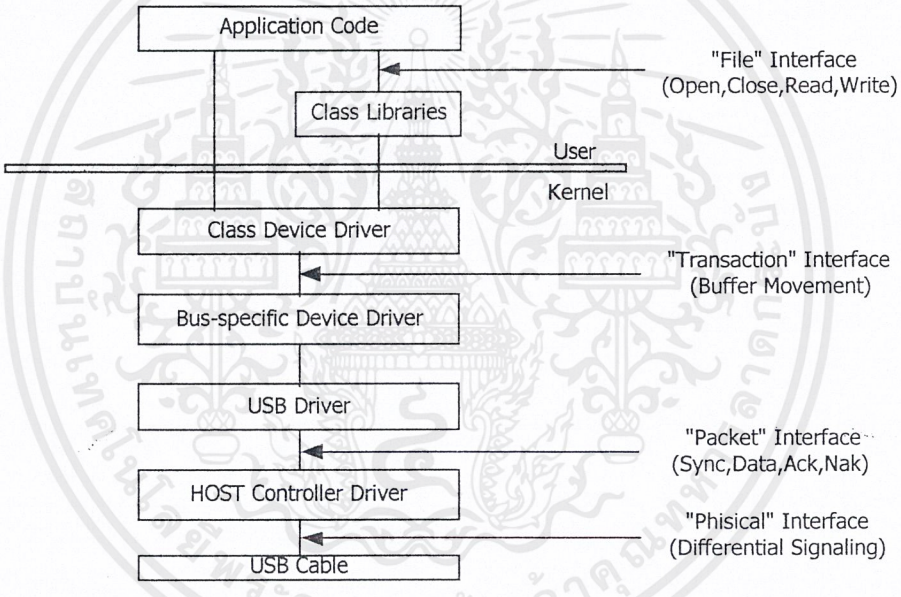
ตามปกติแล้วเมื่อเราสร้างอุปกรณ์ขึ้นมาหนึ่งชิ้นเราจะต้องเขียนซอฟต์แวร์เพื่อควบคุมการทำงานของอุปกรณ์ชิ้นนั้นเรียกว่าไดร์เวอร์ ซึ่งในอุปกรณ์ยูเอสบี นั้นจะประกอบไปด้วย

ไดรเวอร์อุปกรณ์ยูเอสบี

ไดรเวอร์ยูเอสบี

ไดรเวอร์โฮสต์คอนโทรลเลอร์

ซึ่งรายละเอียดได้อธิบายไปแล้วอยู่ในบทที่ 2 เพื่อให้การเขียนโปรแกรมง่ายขึ้น เราจะเรียกใช้ Class HID ที่มีอยู่ในระบบปฏิบัติการ (Windows98) โดยซอฟต์แวร์ของโฮสต์ มีโครงสร้างดังรูปที่ 6.2.1



รูปที่ 6.2.1 โครงสร้างของโฮสต์ซอฟต์แวร์

โดยในส่วนที่จะเขียนติดต่อผู้ใช้งานนั้นคือส่วนที่เป็น Application Code ซึ่งจะใช้ Win32 API ในการเรียกใช้ฟังก์ชันที่อยู่ในไฟล์ HIDCLASS.SYS และ HIDUSB.SYS โดยโปรแกรมที่เขียนนั้นจะออกแบบให้สามารถเปิด อ่าน เขียน และปิดอุปกรณ์ยูเอสบี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเปิดอุปกรณ์ยูเอสบี นั้นจะใช้ฟังก์ชัน

Public Function OpenUSBdevice(NameOfDevice$) As Boolean

โดยจะประกาศฟังก์ชันนี้ไว้ในโมดูลที่ชื่อ HIDInterface.bas โดยฟังก์ชันนี้จะเข้าไปค้นหาชื่อของอุปกรณ์ซึ่งในที่นี้ก็คือ Mouse USB ซึ่งเป็นสตริง ดิซคริปเตอร์ ถ้าหาพบจะคืนค่าเป็น True และถ้าไม่พบจะคืนค่าเป็น False

การอ่านข้อมูลจากอุปกรณ์ยูเอสบี นั้นจะใช้ฟังก์ชัน

Public Sub ReadUSBdevice(BufferPtr&, ByteCount&)

โดยจะประกาศฟังก์ชันนี้ไว้ในโมดูลที่ชื่อ HIDInterface.bas โดยที่ BufferPtr& นั้นคือตัวชี้ (pointer) ที่ชี้ไปยังบัฟเฟอร์ และ ByteCount& คือจำนวนไบต์ที่ต้องการอ่านค่า ซึ่งข้อมูลที่จะอ่านนั้นบรรจุอยู่ในอินพุทรีพอร์ต (Input Report) ของอุปกรณ์ยูเอสบี โดยที่ไบต์แรกจะเป็น ReportID ซึ่งมีค่าเป็น 0

การเขียนข้อมูลไปยังอุปกรณ์ยูเอสบี นั้นจะใช้ฟังก์ชัน

Public Sub WriteUSBdevice(BufferPtr&, ByteCount&)

การเขียนข้อมูลนั้นจะคล้ายๆกับการอ่านข้อมูลจากอุปกรณ์ยูเอสบี

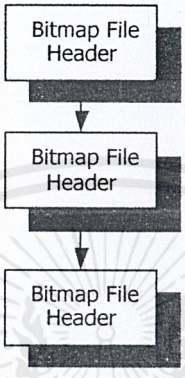
การปิดอุปกรณ์ยูเอสบี นั้นจะใช้ฟังก์ชัน

Public Sub CloseUSBdevice()

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2.1 การเปิดภาพบิตแม็ป

การเปิดภาพเพื่อที่จะดึงเอาข้อมูลที่อยู่ในภาพออกมาใช้งานเช่น ความกว้าง-ยาวของภาพ, ขนาดของภาพ, หรือค่าสีของแต่ละพิกเซลนั้น เราจำเป็นต้องรู้รูปแบบการเก็บข้อมูลของภาพแต่ละชนิด โดยรูปแบบการเก็บข้อมูลของภาพแบบบิตแม็ปแสดงไว้ในภาคผนวก



รูปที่ 6.2.2 โครงสร้างข้อมูลของภาพบิตแม็ป

ภาพบิตแม็ปนั้นประกอบไปด้วย 3 ส่วนคือ

Bitmap File Header

Bitmap Info

Bitmap Data

โดยที่ 2 ไบต์แรกนั้นจะเก็บตัวอักษร "BM" ในรหัส ASCII นั่นคือค่า (42 4D) โดยที่ลักษณะการเก็บข้อมูลนั้นจะเก็บทีละแถว ไล่จากซ้ายไปขวา และการเก็บข้อมูลจะเก็บจากบนลงล่าง นั่นคือ ไบต์ข้อมูลไบต์แรกจะอยู่ที่ตำแหน่งบนทางด้านซ้าย

ในภาษา Visual Basic นั้นได้จัดเตรียมเครื่องมือสำหรับกราฟฟิคซึ่งสามารถใช้งานได้ง่าย นั้นคือเครื่องมือที่ชื่อว่า "PictureBox" และ "Image" โดยพร็อพเพอร์ตี้ที่น่าสนใจที่นำมาใช้ของ "Image" คือ พร็อพเพอร์ตี้ Stretch ซึ่งเมื่อเปิดภาพไปที่ "Image" มันจะทำให้ภาพนั้นเติมเต็มขนาดของเครื่องมือ

เมื่อเปิดภาพมาเรียบร้อยแล้วเราจะดึงค่าสีของแต่ละพิกเซลโดยใช้ฟังก์ชัน GetPixel

Public Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long

โดยค่าที่ดึงออกมานั้นจะเป็นชนิด Long (32-bit Integer)คือค่าของ RGB ซึ่งเป็นค่าความสว่าง (Brightness) ดังนั้น

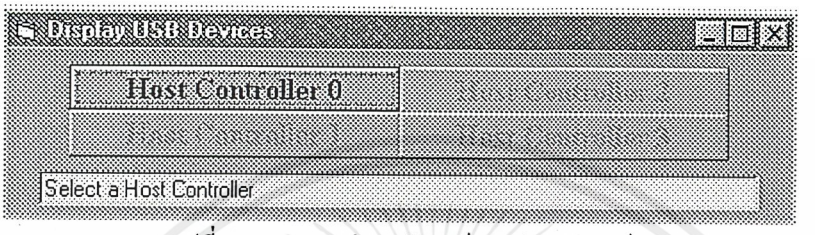
ค่าของสีขาวคือ 0x00FFFFFFH ค่าของสีดำคือ 0x00000000H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

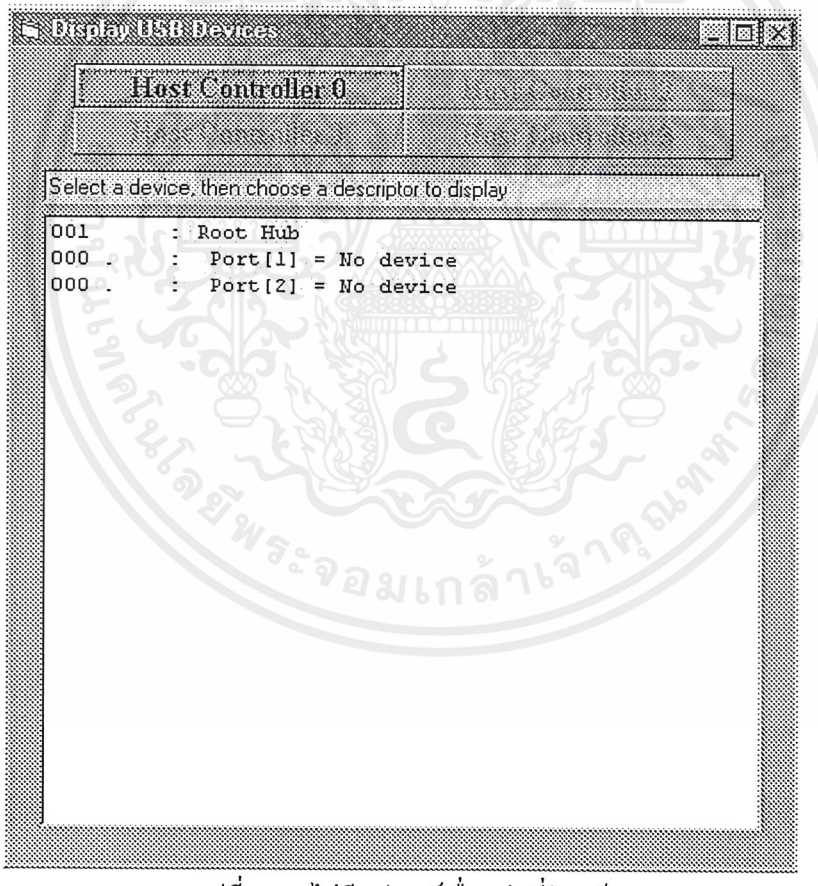
บทที่ 7

ผลการทดลอง

7.1 ส่วนติดต่อกับผู้ใช้งานเขียนด้วยภาษา Visual Basic

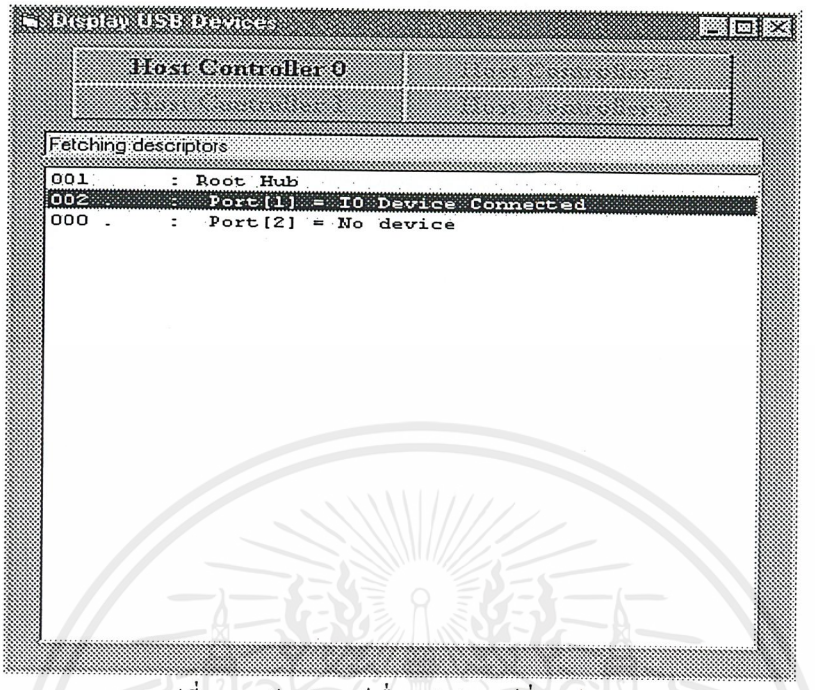


รูปที่ 7.1.1 แสดงว่าพบโฮสต์คอนโทรลเลอร์



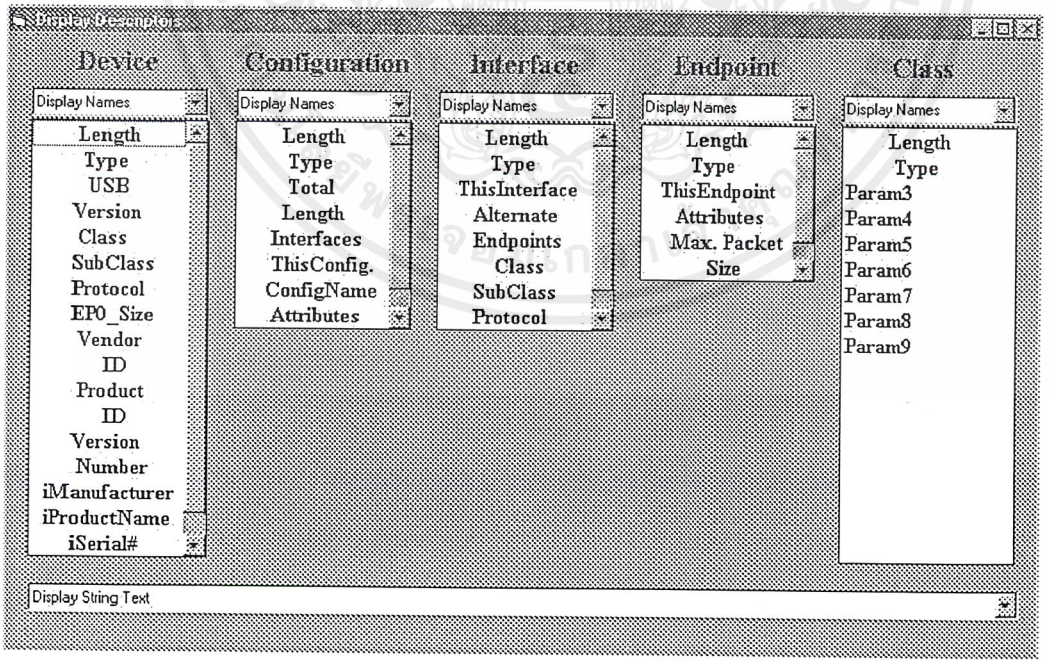
รูปที่ 7.1.2 ไม่มีอุปกรณ์เชื่อมต่อที่โฮสต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

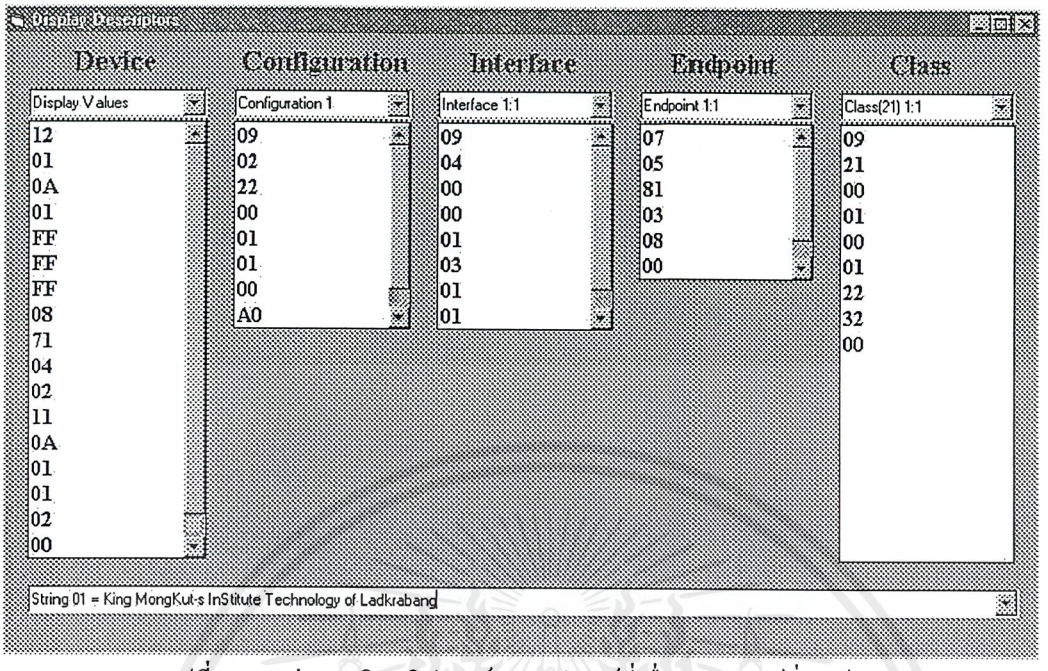


รูปที่ 7.1.3 มีอุปกรณ์เชื่อมต่อโฮสต์ที่พอร์ต 1

เมื่อคลิกเมาส์สองครั้งที่แถบสีในรูปที่ 7.1.3 จะสามารถดูดิซคริปเตอร์ต่างๆของอุปกรณ์ที่มาเชื่อมต่อดังแสดงในรูปที่ 7.1.4 และ รูปที่ 7.1.5

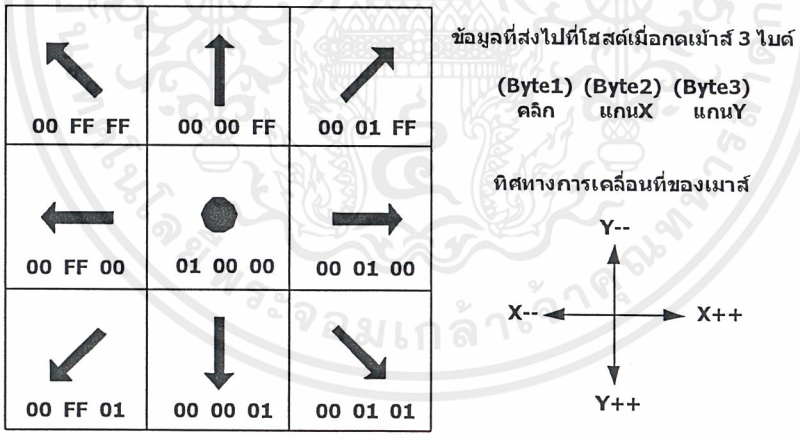


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.1.5 ค่าของดิซคริปเตอร์ของอุปกรณ์ที่เชื่อมต่อโฮสต์ที่พอร์ต 1

7.2 ค่าของข้อมูลที่ส่งจากอุปกรณ์ไปยังโฮสต์



รูปที่ 7.1.6 ค่าของข้อมูลที่ส่งจากอุปกรณ์ไปยังโฮสต์เพื่อควบคุมการเคลื่อนที่ของเมาส์

เมื่อไดร์เวอร์โฮสต์คอนโทรลเลอร์ได้รับข้อมูลที่อุปกรณ์ส่งมาจะส่งผ่านไปยังคลาส HID ที่มีอยู่แล้วในระบบปฏิบัติการวินโดว์เพื่อควบคุมการเคลื่อนที่ของเมาส์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
บทสรุปและข้อเสนอแนะในการพัฒนา

8.1 บทสรุป

ในการศึกษาการส่งผ่านข้อมูลโดยใช้อุปกรณ์เมาส์ในการทดสอบการส่งข้อมูลจากอุปกรณ์มายังโฮสต์สามารถใช้เป็นเมาส์ได้ตรงตามวัตถุประสงค์ แต่ในส่วนการทดสอบการส่งข้อมูลจากโฮสต์ไปยังอุปกรณ์แสดงผล LED DOT MATRIX นั้นยังไม่สามารถทำได้ตามวัตถุประสงค์เนื่องจากไม่สามารถเขียนโปรแกรมเพื่อส่งข้อมูลจากโปรแกรมประยุกต์ผ่านไปยังไดร์เวอร์ยูเอสบีได้

8.2 ข้อเสนอแนะในการพัฒนา

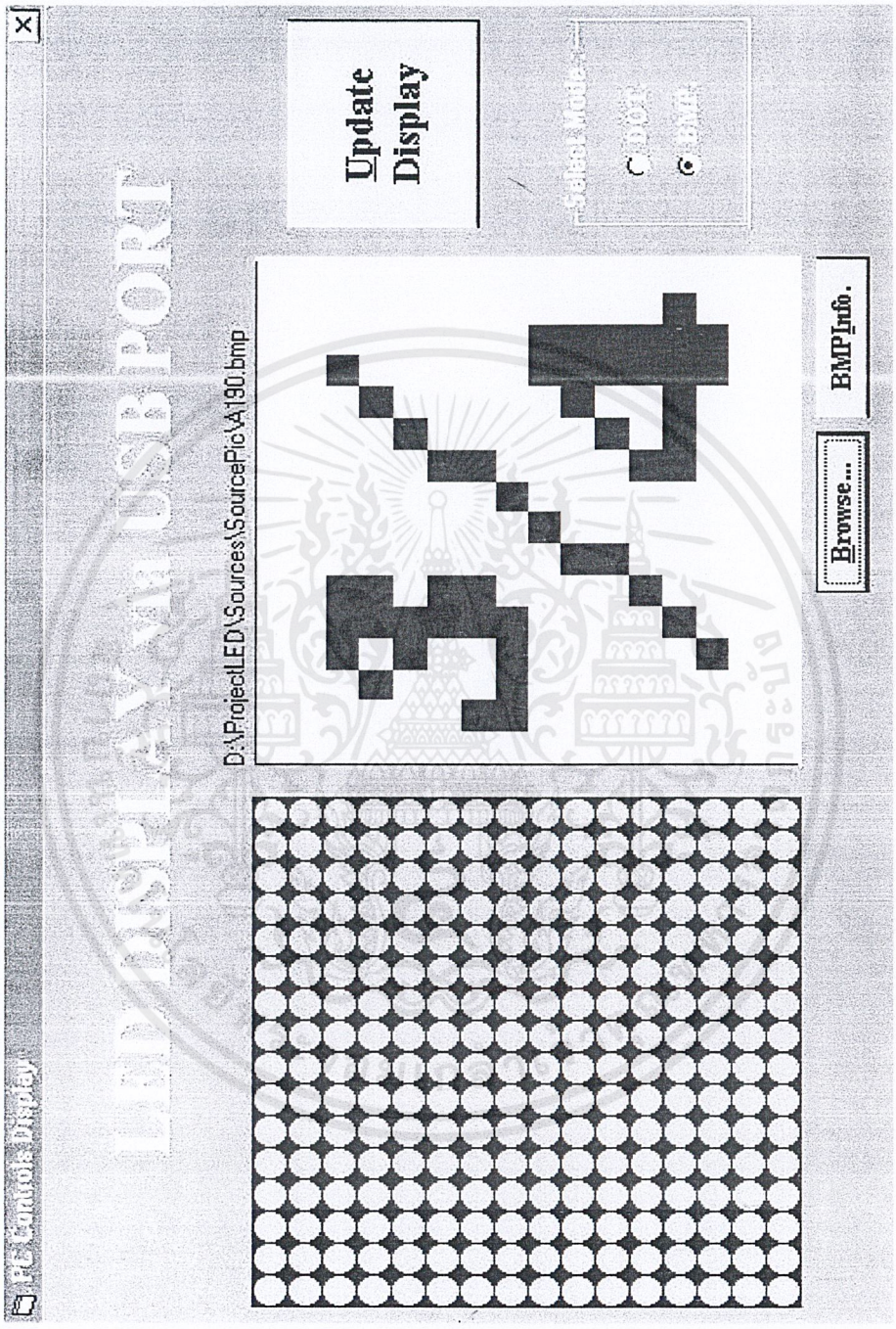
ในปัจจุบันเทคโนโลยีการสื่อสารที่เกี่ยวข้องกับการส่งข้อมูลด้วยความเร็วสูง ได้พัฒนาไปอย่างรวดเร็ว ซึ่งอุปกรณ์ยูเอสบีเป็นตัวอย่างที่ดีในการศึกษาการส่งข้อมูลด้วยความเร็วสูง ในการศึกษานี้พบปัญหาหลายประการเช่น ไอซีที่ใช้ในการทดลองไม่มีขายในประเทศไทยและ ไม่มีอุปกรณ์ที่ใช้ในการวัดผลคือลอจิกอนาไลเซอร์ (Logic Analyzer) ที่มีการตอบสนองได้ทันนั้นมีราคาแพงมาก นอกจากในส่วนของฮาร์ดแวร์แล้วในส่วนของซอฟต์แวร์นั้นควรจะศึกษาการเขียนไดร์เวอร์อุปกรณ์โดยในการศึกษานี้ใช้คอมไพเลอร์ที่ชื่อ KEIL ซึ่งใช้ภาษา C ในการเขียนซึ่งเขียนง่ายและผู้ที่มาศึกษาต่อก็สามารถทำความเข้าใจได้ไม่ยาก ส่วนโปรแกรมประยุกต์นั้นหากใช้ภาษา Visual Basic เขียนควรจะศึกษาไปถึงการเรียกใช้ฟังก์ชันของ Win32 API ซึ่งจะทำให้ง่ายขึ้นกว่าการเขียนโปรแกรมด้วยตนเองทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



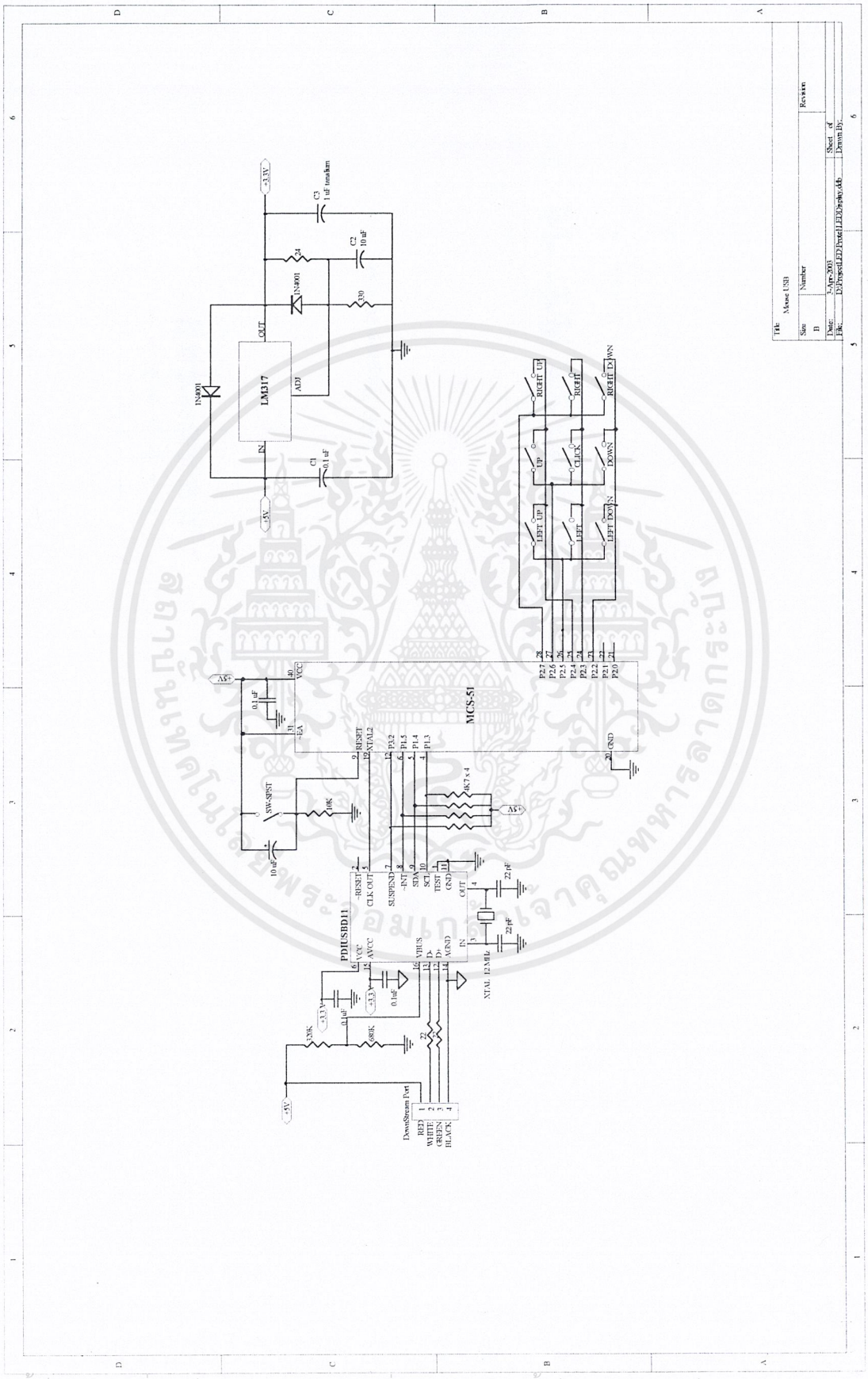
ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ส่วนติดต่อกับผู้ใช้งานของการแสดงผลภาพผ่านทางพอร์ตยูเอสบี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BMP

Acronym BitMaP format

Origin
Windows Marketing Group
Microsoft Corporation
16011 NE 36th Way
Box 97017
Redmond, WA 98073-9717
Tel. (206) 882-8080

Motivation To storage raster image data in a format that is independent of the color specification scheme used on any single hardware device

Users Microsoft Windows 3.0 (or higher) users and application programs running under the Windows 3.0 (or higher) environment

Control

Copyrighted by	Microsoft Corporation
Trademarks	Microsoft, Windows, Windows/286, Windows/386
Royalty fees	None

Versions Version 1

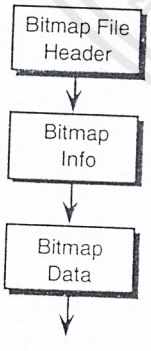
Overview

Major Type of Data	2D raster
Color Representation	Monochrome, Color lookup table, RGB
Data Organization	Sequential: 2D array of pixel values
Data Encoding	Binary
Data Compression	None, run-length

Data Types

Image size	Width and height
Bits per pixel	
Type of compression	
Resolution	Pixels per meter
Color palette	
Raster data	

Data Organization General syntax of the data stream



Signature
First two bytes, must be BM, in ASCII, Hex(42 4D)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BMP *(Continued)*

Bitmap File Header

All data values are in LSB first format

Byte#	Data	Details
1–2	File type	Must be ASCII text "BM"
3–6	Size of the file	In double words (32-bit integers)
7–10	Reserved for future use	Must be zero
11–14	Byte offset to bitmap data	Offset from the Bitmap File Header (i.e., the start of the file)

Bitmap Information

Byte#	Data	Details
1–4	Number of bytes in header	Currently 40 bytes
5–8	Width of bitmap	In pixels
9–12	Height of bitmap	In pixels
13–14	Number of color planes	Must be set to 1
15–16	Number of bits per pixel	Valid choices are 1, 4, 8, 24; if not 24, determines the size of the palette
17–20	Type of compression	0: No compression; 1: run length (8 bits per pixel); 2: run length (4 bits per pixel)
21–24	Size of image	In bytes
25–28	Horizontal resolution	In pixels/meter
29–32	Vertical resolution	In pixels/meter
33–36	Number of color indexes used by the bitmap	Zero indicates all colors are important
37–40	Number of color indexes important for displaying bitmap	A value of zero indicates all colors are important
41	Blue color value	Beginning color palette (entry 0) (Notice RGB values are reversed from typical order)
42	Green color value	
43	Red color value	
44	Reserved for future use	Must be zero
...	... Remaining color palette entries	4 bytes per palette entry, the number of entries is based on the bits per pixel value above. Colors should be listed in order of importance

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BMP *(Continued)*

Bitmap Data
The pixels are stored by rows, left to right within each row. *The rows are stored bottom to top;* the origin of the bitmap is the lower-left corner. The bits of each pixel are packed into bytes and each scan line is padded with zeros, if necessary, to be aligned with a 32-bit word boundary

File Name Suffix DOS, OS/2 .BMP

References

Microsoft Windows Programmer's Reference, New for Version 3, edited by Microsoft Corporation, Microsoft Press, Redmond, WA, 1990, ISBN 1-55615-309-0

Microsoft Windows Software Development Kit
Call the Microsoft Information Center at (800) 426-9400 for further information

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ดีด้วยความช่วยเหลือจากอาจารย์และบุคคลหลายท่านดังนี้

ผศ. พลผดุง ผดุงกุล อาจารย์ที่ปรึกษาที่ได้กรุณาให้คำปรึกษา พร้อมทั้งหาข้อมูล และชี้แนะแนวทางในการแก้ไขปัญหาต่างๆ ตลอดจนการฝึกฝนให้ผู้เขียนสามารถแก้ไขปัญหาและได้เรียนรู้สิ่งต่างๆ ผู้เขียนขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณบริษัท ฟิลลิปป์ ประเทศไทย จำกัด สำหรับไอซี PDIUSBD11 ที่ใช้ในการทำปริญญานิพนธ์

ขอขอบคุณภาควิชาอิเล็กทรอนิกส์ สำหรับเครื่องมือและอุปกรณ์ที่ใช้ในการทดลอง

สุดท้ายนี้ขอกราบขอบพระคุณ คุณพ่อคุณแม่ที่คอยเอาใจแส่ดูแลและให้กำลังใจ ตลอดจนให้ความช่วยเหลือในด้านต่างๆ แก่ผู้เขียนตลอดเวลาในการปริญญานิพนธ์ฉบับนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

1. ลภน สุภาพ, "เจาะลึกให้เข้าใจใน USB", วารสารเซมิคอนดักเตอร์อิเล็กทรอนิกส์, ฉบับที่ 225, ฉบับที่ 226, ฉบับที่ 227, ฉบับที่ 229, ฉบับที่ 230, 2544
2. วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล, "เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ฉบับ AT89C5x ของ Atmel", บริษัท อินโนเวตีฟ จำกัด
2. กิตติ ภักดีวัฒนะกุล, จำลอง ครูอุตสาหะ, "Visual Basic 6 ฉบับโปรแกรมเมอร์", สำนักพิมพ์ กรุงเทพฯ เคทีพี คอมพ์ แอนด์ คอนซัลท์, 2544
3. John Hyde, "USB Design by Example A practical Guide to Building I/O Devices", John Wiley & Sons, 1999
4. Wayne C. Brown, Barry J. Shepherd, "Graphics file formats : reference and guide", Greenwich, CT : Manning, 1995
5. http://www.semiconductors.philips.com/buses/usb/products/device/pdiusbd11/faq/index.html
6. http://www.usb.org
7. http://www.beyondlogic.org

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้