ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ ชุดสาธิตมาตรฐาน USB โดย PIC16F877A

USB Standard Demonstrator By PIC16F877A

ชื่อนักศึกษา 1. นายฉลองศักดิ์ คล่องการ รหัสประจำตัว 45035335

2. นางสาวสุจิตรา โสทอง รหัสประจำตัว 45035365

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ที่ปรึกษา อาจารย์สุรพงษ์ สิริพงศ์ดี

อาจารย์ที่ปรึกษาร่วม อาจารย์โกศล ตราชู

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์อำพล	ทองระอา	
2. อาจารย์สุรพงษ์	สิริพงศ์ดี	
3. อาจารย์โกศล	ตราชู	
4. อาจารย์ปิยะ	ศุภวราสุวัฒน์	
5. อาจารย์สุระชัย	พิมพ์สาลี	

วัน/เดือน/ปีที่สอบ วันพฤหัสบดีที่ 20 พฤศจิกายน พ.ศ. 2546 เวลา 10:00 น.

สถานที่สอบ ห้อง ค.311 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(นายสุรสิทธิ์ ราตรี)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 21 เดือน พ.ย. พ.ศ. 47

<BT4610162>

ชุดสาธิตมาตรฐาน USB โดย PIC16F877A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ปริญญานิพนธ์

ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

USB STANDARD DEMONSTRATOR BY PIC 16F877A



นายฉลองศักดิ์ คล่องการ

นางสาวสุจิตรา โสทอง

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เลขหมู่.....
เลขทะเบียน 51858
วัน,เดือน,ปี 3 ส.ค. 2547

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบ.....
เ.....

ปริญญานิพนธ์

เรื่อง ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

USB Standard Demonstrator By PIC 16F877A

วัตถุประสงค์

1. เพื่อศึกษาวิธีการทำงานของมาตรฐานที่ใช้ในระบบบัส USB
2. เพื่อออกแบบอุปกรณ์เชื่อมต่อกับพอร์ต USB โดยใช้ PIC 16F877A
3. เพื่อสร้างชุดสาธิตเชื่อมต่อกับพอร์ต USB โดยใช้ PIC 16F877A
4. เพื่อทดลองรับและส่งข้อมูลผ่านระบบบัส USB โดยใช้ PIC 16F877A
5. เพื่อนำไปใช้สอนการรับและส่งข้อมูลด้วยมาตรฐาน USB

ประโยชน์ที่คาดว่าจะได้รับ

1. รู้วิธีการทำงานของมาตรฐานที่ใช้ในระบบบัส USB
2. ได้วงจรการเชื่อมต่อพอร์ต USB โดยใช้ PIC 16F877A
3. ได้ชุดสาธิตเชื่อมต่อกับพอร์ต USB โดยใช้ PIC 16F877A
4. ได้โปรแกรมสำหรับการรับและส่งข้อมูลผ่านระบบบัสพอร์ต USB โดยใช้ PIC 16F877A
5. ได้ชุดสาธิตและใบงานการทดลองการรับและส่งข้อมูลด้วยมาตรฐาน USB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อหัวข้อ	ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A
นักศึกษา	นายฉลองศักดิ์ คล่องการ
	นางสาวสุจิตรา โสทอง
อาจารย์ที่ปรึกษา	อาจารย์สุรพงษ์ สิริพงศ์ดี
อาจารย์ที่ปรึกษาร่วม	อาจารย์โกศล ตราชู
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขา	อิเล็กทรอนิกส์และคอมพิวเตอร์
ปีการศึกษา	2546

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอ ชุดสาธิตมาตรฐาน USB ควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์เบอร์ PIC 16F877A ร่วมกับวงจรในส่วนที่ติดต่อกับพอร์ต USB ซึ่งในส่วนนี้จะมี ชิป FT245BM ทำหน้าที่ควบคุมเชื่อมต่อกับพอร์ต USB และมี EEPROM เก็บข้อมูลที่สำคัญต่างๆ ของอุปกรณ์ USB มีองค์ประกอบที่สำคัญอีกส่วนหนึ่งก็คือ ส่วนของวงจรแปลง RS-232, RS-422 และRS-485 โดยมี LED เป็นส่วนแสดงผล ในส่วนของการทดลองจะใช้โปรแกรม 2 โปรแกรมคือ โปรแกรม Serial Test ใช้ในการรับและส่งข้อมูลผ่านพอร์ตอนุกรมและโปรแกรม USB Test ใช้ในการรับและส่งข้อมูลผ่านพอร์ต USB ซึ่งการพัฒนาทำให้สามารถนำไปใช้ในการรับและส่งข้อมูลจากพอร์ต USB, RS-232, RS-422 และRS-485 เพื่อใช้ในการควบคุมต่างๆ ในงานอุตสาหกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	USB Standard Demonstrator By PIC 16F877A
Student	Mr.Chalongsak Klongkan
	Miss Sujittra Sothong
Advisor	Mr.Surapong Siripongdee
Co-Advisor	Mr.Koson Trachu
Education Level	Bachelor of Science in Industrial Education
Program in	Electronic and Computer
Academic Year	2003

ABSTRACT

This thesis presents the USB standard demonstrator. It has been implemented using the microcontroller PIC 16F877A .The circuit that connected to USB port, this circuit is based mainly on the internal resources of a chip FT245BM to control connecting USB port and EEPROM to store the important information of USB devices. Another important components are the circuit to adapt RS-232, RS-422 and RS-485 which its output section is displayed by LED. The experiment used two programs.The first serial test program that uses to send and receive data through serial port. The second USB test program that uses to send and receive data through USB port. The experimental result show that it can send and receive data through USB port, RS-232, RS-422 and RS-485. So it will be good to the controller used in the industry.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ลุล่วงไปด้วยดี เนื่องมาจากความร่วมมือของเพื่อนสมาชิกในกลุ่มทุกท่าน ขอขอบคุณ อาจารย์สุรพงษ์ สิริพงศ์ดี อาจารย์ที่ปรึกษา อาจารย์โกศล ตราชู อาจารย์ที่ปรึกษาร่วมและอาจารย์ในภาควิชาครุศาสตร์วิศวกรรมทุกท่าน ที่ช่วยเหลือในด้านแนวคิดและคอยให้คำปรึกษาเกี่ยวกับโครงงาน ทั้งทางด้านฮาร์ดแวร์และทางด้านซอฟต์แวร์ ตลอดจนอาจารย์ท่านอื่นๆ ในคณะที่ให้ข้อมูลเพิ่มเติมในเรื่องต่างๆ ในการทำโครงงาน ขอขอบคุณอาจารย์ที่คอยสั่งสอนมาตั้งแต่อดีตจนถึงปัจจุบันที่ทำให้เรามีวันนี้ได้และจะมีวันต่อๆ ไป ขอขอบคุณกำลังใจของเพื่อนอิเล็กทรอนิกส์และคอมพิวเตอร์และสาขาอื่นๆ ทุกคน ในยามท้อแท้เหนื่อยหน่าย และอยู่ด้วยกันจนนาทีสุดท้ายของการสอบโครงงาน และที่สำคัญที่สุด คือ ขอบพระคุณบิดามารดาที่ได้ให้ชีวิตนี้และเลี้ยงดูจนเติบโตมาอย่างดีมีประสิทธิภาพ คอยสนับสนุนในด้านกำลังใจ ทั้งทางด้านงบประมาณในการทำโครงงานและปริญญานิพนธ์ครั้งนี้ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เรื่อง	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VII
สารบัญรูป	IX
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญ	1
1.2 ขีดความสามารถของโครงงาน	1
1.3 เนื้อหาโดยสังเขป	1
บทที่ 2 ทฤษฎีและหลักการ	3
2.1 กล่าวนำ	3
2.2 พอร์ต USB	3
2.2.1 ภาพรวมการทำงานของพอร์ต USB	4
2.2.2 สายเชื่อมต่อและคอนเน็กเตอร์ของพอร์ต USB	15
2.2.3 สัญญาณต่างๆ ในระบบของพอร์ต USB	20
2.2.4 การจัดการด้านพลังงานของ USB	28
2.2.5 โครงสร้างพื้นฐานการส่งข้อมูลของพอร์ต USB	34
2.2.6 ความผิดพลาดและการแก้ไขในการสื่อสารข้อมูลผ่านพอร์ต USB	40
2.2.7 ดิสคริปเตอร์และกระบวนการตั้งค่า	44
2.3 ส่วนติดต่อพอร์ต USB	52
2.4 ส่วนควบคุม	52
2.5 การสื่อสารอนุกรม	53
2.5.1 RS-232	56
2.5.2 RS-422	60
2.5.3 RS-485	61
2.5.4 รีจิสเตอร์ของ PIC ที่เกี่ยวกับการสื่อสารอนุกรม	63

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
2.6 การสร้างชุดสาธิต	69
บทที่ 3 การออกแบบ การสร้าง และการทำงาน	70
3.1 ส่วนควบคุมการทำงานและแสดงผล	70
3.1.1 การออกแบบและการสร้าง	70
3.1.2 การทำงาน	71
3.2 ส่วนติดต่อพอร์ต USB	71
3.2.1 การออกแบบและการสร้าง	71
3.2.2 การทำงาน	72
3.3 ส่วนโปรแกรมการทำงาน	72
3.3.1 โปรแกรมรับและส่งข้อมูล	72
3.3.2 โปรแกรมควบคุมฮาร์ดแวร์	74
3.4 การสร้างชุดสาธิต	75
3.4.1 การออกแบบและการสร้าง	75
3.4.2 การทำงาน	79
บทที่ 4 การทดลองและผลการทดลอง	80
4.1 การทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-232	80
4.1.1 การทดลอง	80
4.1.2 ผลการทดลอง	81
4.2 การทดลองการแปลงข้อมูลจาก RS-232 เป็นพอร์ต USB	81
4.2.1 การทดลอง	81
4.2.2 ผลการทดลอง	82
4.3 การทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-422	82
4.3.1 การทดลอง	82
4.3.2 ผลการทดลอง	83
4.4 การทดลองการแปลงข้อมูลจาก RS-422 เป็นพอร์ต USB	83
4.4.1 การทดลอง	83

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
4.4.2 ผลการทดลอง	84
4.5 การทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-485	84
4.5.1 การทดลอง	84
4.5.2 ผลการทดลอง	86
4.6 การทดลองการแปลงข้อมูลจาก RS-485 เป็นพอร์ต USB	86
4.6.1 การทดลอง	86
4.6.2 ผลการทดลอง	87
บทที่ 5 บทสรุป	88
5.1 สรุป	88
5.2 ปัญหาและแนวทางแก้ไข	88
5.3 แนวทางการพัฒนา	88
บรรณานุกรม	89
ภาคผนวก ก เครื่องต้นแบบ	90
ภาคผนวก ข วงจรและแผ่นวงจรพิมพ์	93
ภาคผนวก ค รายการอุปกรณ์	103
ภาคผนวก ง แผนผังการทำงานและรหัสต้นฉบับของโปรแกรม	108
ภาคผนวก จ ใบงาน	157
ภาคผนวก ฉ คู่มือสำหรับผู้ทดลอง	211
ภาคผนวก ช คู่มือการใช้งาน	247
ภาคผนวก ซ รายละเอียดและคุณสมบัติของอุปกรณ์	258
ประวัติผู้แต่ง	270

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่	หน้า
2.1 หน้าที่ของขาคอนเน็กเตอร์แต่ละขา	17
2.2 ค่าอัตราการหน่วงเวลาของสาย USB เมื่อเทียบกับความยาวของสายสัญญาณ	19
2.3 สถานะของบัส USB	27
2.4 รายละเอียดข้อมูลแต่ละไบต์ของดีไวซ์ดิสคริปเตอร์	45
2.5 รายละเอียดข้อมูลแต่ละไบต์ของคอนฟิกิวเรชั่นดิสคริปเตอร์	46
2.6 รายละเอียดข้อมูลแต่ละไบต์ของอินเตอร์เฟสดิสคริปเตอร์	47
2.7 รายละเอียดข้อมูลแต่ละไบต์ของเอ็นด์พอยน์ดิสคริปเตอร์	48
2.8 รายละเอียดข้อมูลแต่ละไบต์ของฮับคลาสคริปเตอร์	49
2.8 (ต่อ) รายละเอียดข้อมูลแต่ละไบต์ของฮับคลาสคริปเตอร์	50
2.9 การจัดขาสัญญาณของพอร์ตอนุกรมในแบบต่างๆ และหน้าที่การทำงาน	57
2.9 (ต่อ) การจัดขาสัญญาณของพอร์ตอนุกรมในแบบต่างๆ และหน้าที่การทำงาน	58
2.10 ตารางเปรียบเทียบของรูปแบบ RS-232, RS-422 และRS-485	62
2.11 ตารางแสดง รีจิสเตอร์ที่เกี่ยวข้องกับการสื่อสารข้อมูลพอร์ตอนุกรม	63
2.12 ตารางแสดงการคำนวณหาค่า Baud rate	66
4.1 ข้อมูลในการส่งทดลอง	80
4.2 ข้อมูลในการส่งทดลอง	81
4.3 ข้อมูลในการส่งทดลอง	81
4.4 ข้อมูลในการส่งทดลอง	82
4.5 ข้อมูลในการส่งทดลอง	83
4.6 ข้อมูลในการส่งทดลอง	83
4.7 ข้อมูลในการส่งทดลอง	84
4.8 ข้อมูลในการส่งทดลอง	84
4.9 ข้อมูลในการส่งทดลอง	85
4.10 ข้อมูลในการส่งทดลอง	86
4.11 ข้อมูลในการส่งทดลอง	86
4.12 ข้อมูลในการส่งทดลอง	87
ค.1 รายการอุปกรณ์ของวงจรชุดสาธิตหลัก	104

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง (ต่อ)

ตารางที่	หน้า
ค.1 (ต่อ) รายการอุปกรณ์ของวงจรชุดสาธิตหลัก	105
ค.2 รายการอุปกรณ์ของวงจร USB โมดูล	105
ค.2 (ต่อ) รายการอุปกรณ์ของวงจร USB โมดูล	106
ค.3 รายการอุปกรณ์ของวงจรแปลง RS-422 to RS-232	106
ค.4 รายการอุปกรณ์ของวงจรแปลง RS-485 to RS-232	107

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่	หน้า
2.1 การจัดลำดับการรับส่งข้อมูลของอุปกรณ์แต่ละตัว	6
2.2 แผนผังการทำงานของ USB รูตฮับอย่างง่าย	8
2.3 โครงสร้างการเชื่อมต่อของ USB	10
2.4 ระดับการทำงานของดิสคริปเตอร์ในพอร์ต USB	12
2.5 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม A	15
2.6 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม B	16
2.7 การใช้งานสาย USB ที่ใช้คอนเน็กเตอร์ทั้งสองรูปแบบ	16
2.8 โครงสร้างภายในของสาย USB แบบความเร็วต่ำ	18
2.9 โครงสร้างภายในของสาย USB แบบความเร็วสูง	19
2.10 โครงสร้างภายในของสาย USB แบบความเร็วสูง	21
2.11 ระดับสัญญาณภายในสาย D+ และ D- เมื่อมีการเชื่อมต่อและปลดอุปกรณ์	22
2.12 การเข้ารหัส NRZI	23
2.13 การเติมบิตสตัฟ	24
2.14 วงจรเชื่อมต่อระหว่างภาคส่ง (ฮับ) และภาครับ (ตัวอุปกรณ์)	25
2.15 ช่วงเวลาขอบขาขึ้น (Rise time) และขอบขาลง (Fall time) ของสัญญาณ	27
2.16 ข้อจำกัดด้านแรงดันที่จุดเชื่อมต่อต่าง ๆ	29
2.17 การจัดการพลังงานของฮับที่มีแหล่งจ่ายไฟเลี้ยง	30
2.18 การใช้พลังงานของอุปกรณ์ไฮบริด	31
2.19 การ Resume จากอุปกรณ์ USB	34
2.20 ตัวอย่างการใช้งานกระบวนการกลับสถานะข้อมูลหรือดาต้าทอกเกิล	43
2.21 แผนผังการทำงานเบื้องต้นของส่วนติดต่อพอร์ต USB	52
2.22 แผนผังการทำงานของส่วนควบคุม	53
2.23 แผนผังเวลาของการสื่อสารข้อมูลแบบซิงโครนัส	54
2.24 รูปแบบของข้อมูลแบบอะซิงโครนัส	55
2.25 คอนเน็กเตอร์อนุกรม	57
2.26 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในรูปแบบต่างๆ	59

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่	หน้า
2.27 ระบบการส่งสัญญาณที่เป็นแบบผลต่าง	60
2.28 การต่อสายสัญญาณแบบ RS-422 ใช้สาย 4 เส้น	61
2.29 การต่อสายสัญญาณแบบ RS-485 ใช้สาย 2 เส้น	62
2.30 ข้อมูลภายในของรีจิสเตอร์ PIR1	63
2.31 ข้อมูลภายในของรีจิสเตอร์ RCSTA	64
2.32 ข้อมูลภายในของรีจิสเตอร์ TXSTA	65
2.33 ข้อมูลภายในของรีจิสเตอร์ PIE1	66
2.34 ข้อมูลภายในของรีจิสเตอร์ INTCON	67
2.35 ข้อมูลภายในของรีจิสเตอร์ PIE1	68
3.1 วงจรควบคุมการทำงานหลัก	70
3.2 วงจรUSB โมดูล	71
3.3 วงจรติดต่อ USB	72
3.4 โปรแกรม USB Test	73
3.5 โปรแกรม Serial Test	74
3.6 ผังงานโปรแกรมควบคุมฮาร์ดแวร์	75
3.7 วงจรแปลง RS-232	76
3.8 วงจรแปลง RS-422	76
3.9 วงจรแปลง RS-485	77
3.10 วงจรแหล่งจ่ายไฟฟ้า	77
3.11 จัมเปอร์เลือกใช้ RS-232, RS-422 และRS-485	77
3.12 วงจรแปลง RS-232 เป็น RS-422	78
3.13 วงจรแปลง RS-232 เป็น RS-485	78
4.1 การเชื่อมต่อชุดสาธิตเพื่อทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-232	80
4.2 การเชื่อมต่อชุดสาธิตเพื่อทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-422	82
4.3 การเชื่อมต่อชุดสาธิตเพื่อทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-485	85

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่	หน้า
ก.1 ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A	91
ก.2 ชุดวงจรแปลง RS-232 เป็น RS-485	91
ก.3 ชุดวงจรแปลง RS-232 เป็น RS-422	92
ข.1 วงจรควบคุมการทำงานหลัก	94
ข.2 วงจรติดต่อ USB	94
ข.3 วงจรแปลง RS-232	95
ข.4 วงจรแปลง RS-422	95
ข.5 วงจรแปลง RS-485	96
ข.6 วงจรแหล่งจ่ายไฟฟ้า	96
ข.7 จัมเปอร์เลือกใช้ RS-232, RS-422 และRS-485	96
ข.8 แผ่นวงจรพิมพ์ชุดวงจรหลัก ด้านบน (ขนาด 75%)	97
ข.9 แผ่นวงจรพิมพ์ชุดวงจรหลัก ด้านล่าง (ขนาด 75%)	97
ข.10 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์ชุดวงจรหลัก (ขนาด 75%)	98
ข.11 วงจร USB โมดูล	98
ข.12 แผ่นวงจรพิมพ์วงจร USB โมดูล ด้านบน	99
ข.13 แผ่นวงจรพิมพ์วงจร USB โมดูล ด้านล่าง	99
ข.14 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจร USB โมดูล	99
ข.15 วงจรแปลง RS-422 เป็น RS-232	100
ข.16 แผ่นวงจรพิมพ์วงจรแปลง RS-422 เป็น RS-232 ด้านบน	100
ข.17 แผ่นวงจรพิมพ์วงจรแปลง RS-422 เป็น RS-232 ด้านล่าง	100
ข.18 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรแปลง RS-422 เป็น RS-232	101
ข.19 วงจรแปลง RS-485 เป็น RS-232	101
ข.20 แผ่นวงจรพิมพ์วงจรแปลง RS-485 เป็น RS-232 ด้านบน	101
ข.21 แผ่นวงจรพิมพ์วงจรแปลง RS-485 เป็น RS-232 ด้านล่าง	102
ข.22 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรแปลง RS-485 เป็น RS-232	102
ง.1 โปรแกรม USB Test	109
ง.2 โปรแกรม Serial Test	129

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่	หน้า
ง.3 ผังงานโปรแกรม Work1	131
ง.4 ผังงานโปรแกรม Work2	136
ง.5 ผังงานโปรแกรม Work3	141
ง.6 ผังงานโปรแกรม Work4	146
ง.7 ผังงานโปรแกรม Work5	151
ช.1 ส่วนประกอบของชุดสาธิตมาตรฐาน USB โดย PIC 16F877A	249
ช.2 ตำแหน่งขาของจุดเชื่อมต่อภายนอก	250
ช.3 หน้าต่างแสดงเมื่อคอมพิวเตอร์ตรวจสอบพบชุดสาธิต	251
ช.4 หน้าต่างการติดตั้งไดรเวอร์	251
ช.5 รายชื่ออุปกรณ์เมื่อทำการติดตั้งเสร็จเรียบร้อย	252
ช.6 การเชื่อมใช้งานต่อ RS-232	253
ช.7 การเชื่อมใช้งานต่อ RS-422	253
ช.8 การเชื่อมใช้งานต่อ RS-485	254
ช.9 โปรแกรม USB Test	255
ช.10 โปรแกรม Serial Test	256

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ปัจจุบันเทคโนโลยีการรับและส่งข้อมูลของคอมพิวเตอร์มีความทันสมัยเป็นอย่างมากและพัฒนาอย่างรวดเร็ว ดังนั้นจึงได้เกิดแนวความคิดที่จะสร้างชุดสาธิตมาตรฐาน USBโดย PIC 16F877A ขึ้นเพื่อเป็นการศึกษาการออกแบบวงจรและหลักการทำงานของพอร์ต (Port) USB (Universal Serial Bus : USB) และสามารถสร้างชุดสาธิตมาตรฐาน USB โดย PIC 16F877A ที่สามารถใช้งานจริงได้

1.2 ขีดความสามารถของโครงงาน

1. สามารถแปลงข้อมูลจาก พอร์ต USB เป็น RS-232 ได้
2. สามารถแปลงข้อมูลจาก RS-232 เป็น พอร์ต USB ได้
3. สามารถแปลงข้อมูลจาก พอร์ต USB เป็น RS-422 ได้
4. สามารถแปลงข้อมูลจาก RS-422 เป็น พอร์ต USB ได้
5. สามารถแปลงข้อมูลจาก พอร์ต USB เป็น RS-485 ได้
6. สามารถแปลงข้อมูลจาก RS-485 เป็น พอร์ต USB ได้

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อสะดวกต่อการศึกษาและทำความเข้าใจ ในแต่ละบทจะประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปริญญานิพนธ์ ขอบเขตของโครงงานและเนื้อหาภายในปริญญานิพนธ์

บทที่ 2 ทฤษฎีและหลักการ กล่าวถึงทฤษฎีส่วนต่างๆ ของชุดสาธิตมาตรฐาน USB โดย PIC 16F877A และหลักการสร้างชุดสาธิต

บทที่ 3 การออกแบบ การสร้าง และการทำงาน กล่าวถึงแนวคิดเบื้องต้นในการออกแบบการสร้าง และการทำงานของส่วนที่สำคัญในการทำโครงงาน ขั้นตอนในการออกแบบวงจรในส่วนต่างๆ และวิธีการสร้าง การทำงานของวงจร การทำงานของโปรแกรมที่ใช้ในวงจร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงวิธีการทดลอง ผลการทดลองที่ได้ของวงจร วงจรในส่วนที่สำคัญของชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

บทที่ 5 บทสรุป ปัญหา แนวทางการแก้ไขและพัฒนา กล่าวสรุปการทำปริญญานิพนธ์ ปัญหาที่เกิดขึ้นตั้งแต่เริ่มทำปริญญานิพนธ์ วิธีการแก้ไขปัญหาที่เกิดขึ้น ข้อเสนอแนะและแนวทางในการทำปริญญานิพนธ์ไปพัฒนาโครงงานนี้ต่อไป เพื่อให้มีประสิทธิภาพมากยิ่งขึ้น

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข วงจรและแผ่นวงจรพิมพ์

ภาคผนวก ค รายการอุปกรณ์

ภาคผนวก ง แผนผังการทำงานและรหัสต้นฉบับของโปรแกรม

ภาคผนวก จ ใบงาน

ภาคผนวก ฉ คู่มือสำหรับผู้ทดลอง

ภาคผนวก ช คู่มือการใช้งาน

ภาคผนวก ซ รายละเอียดและคุณสมบัติของอุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

ทฤษฎีและหลักการต่างๆ ที่จะกล่าวถึงต่อจากนี้ไป เป็นหนึ่งในส่วนประกอบของการสร้างชุดสาธิตมาตรฐาน USB โดย PIC 16F877A โดยชุดสาธิตจะติดต่อรับและส่งข้อมูลจากพอร์ต USB, RS-232, RS-422 และRS-485 และมีไดโอดเปล่งแสง (Light Emitting Diode : LED) เพื่อแสดงผล และได้กล่าวถึง การสร้างชุดสาธิต

2.2 พอร์ต USB

นับจากวันแรกที่มีการคิดค้นเครื่องคอมพิวเตอร์ขึ้นมาจนถึงทุกวันนี้ก็เป็นเวลานานหลายทศวรรษแล้วจากจุดเริ่มต้นของเครื่องจักรขนาดใหญ่คับห้องถูกพัฒนาต่อเนื่องมาเรื่อยๆ จนในทุกวันนี้มีขนาดเล็กแน่นอนว่าระยะเวลาการพัฒนาที่ยาวนานบวกกับความก้าวหน้าอย่างรวดเร็วของเทคโนโลยีย่อมจะพัฒนาส่วนประกอบทั้งหมดไปพร้อมๆ กันและเมื่อคอมพิวเตอร์ทำงานได้เร็วขึ้นใช้งานได้สะดวกขึ้น ชีวิตของมนุษย์ก็จะเกี่ยวพันกับมันมากขึ้นเรื่อยๆ ทำให้เกิดอุปกรณ์ต่างๆ มากที่ถูกคิดค้นขึ้นมาเพื่อจะทำงานร่วมกับคอมพิวเตอร์ โดยอาศัยความสามารถในการประมวลผลที่รวดเร็วของมัน

เมื่อมีอุปกรณ์ที่ต้องนำมาเชื่อมต่อกับคอมพิวเตอร์ก็ต้องหาช่องทางสำหรับส่งและรับข้อมูลจากตัวเครื่อง ช่องทางการส่งหรือรับข้อมูลต่างๆ จากเครื่องคอมพิวเตอร์ถูกตั้งชื่อว่า พอร์ต โดยพอร์ตแต่ละชนิดก็จะถูกออกแบบมาเพื่อให้สามารถรับส่งข้อมูลเฉพาะได้อย่างเหมาะสม ไม่ว่าจะเป็นพอร์ตอนุกรม (Serial port) ซึ่งใช้รับส่งข้อมูลความเร็วต่ำแบบอนุกรมในระยะทางที่ไกล นำมาใช้กับเมาส์หรือโมเด็ม (MODEM) พอร์ตขนาน (Parallel port) ใช้รับส่งข้อมูลความเร็วปานกลางแบบขนานได้ในระยะที่ไม่ไกลมากนัก ถูกนำมาใช้กับเครื่องพิมพ์และสแกนเนอร์ เป็นต้น นอกจากนั้นยังมีพอร์ตเฉพาะพิเศษ อาทิ พอร์ตรับส่งข้อมูลผ่านแสงอินฟราเรดหรือที่เรียกว่า IrDA (Infrared Data Association port) และพอร์ต USB อันเป็นพอร์ตรับส่งข้อมูลอนุกรมแบบอเนกประสงค์ที่ได้รับความนิยมอย่างมากในการพัฒนาอุปกรณ์เชื่อมต่อกับคอมพิวเตอร์ถัดจากการใช้พอร์ตอนุกรมและขนาน ซึ่งจะได้ทำความรู้จักและทดลองเขียนโปรแกรมเพื่อติดต่อกับอุปกรณ์ภายนอกต่อไป

พอร์ตแต่ละชนิดของคอมพิวเตอร์ ได้รับการออกแบบมาเพื่องานเฉพาะทำให้อุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แต่ละตัวต้องเลือกใช้พอร์ตต่างชนิดกันไป ส่งผลให้การนำอุปกรณ์ต่างๆมาเชื่อมต่อกับเครื่องคอมพิวเตอร์เป็นเรื่องที่ต้องให้ความสนใจ เนื่องจากอุปกรณ์แต่ละชนิดก็จะมีคอนเน็กเตอร์ที่ใช้เชื่อมต่อแตกต่างกันไปตามชนิดของพอร์ต เช่น พอร์ตอนุกรมที่ใช้ต่อคีย์บอร์ดและเมาส์ (ทั้งแบบ PS/2 และพอร์ตอนุกรมมาตรฐาน) พอร์ตขนานสำหรับต่อเครื่องพิมพ์ พอร์ตเชื่อมต่อของจอภาพ ทำให้ผู้ใช้ที่ไม่มีความรู้ในการติดตั้งหรือที่เรียกว่า เอ็นด์ยูสเซอร์ (End user) พบกับความยากลำบากในการเรียนรู้เรื่องราวเหล่านี้

นอกจากนั้นการติดตั้งอุปกรณ์ต่างๆ เพิ่มเข้าไปในคอมพิวเตอร์จะเกิดปัญหาการแย่งกันใช้สัญญาณอินเตอร์รัพต์ (Interrupt Request : IRQ) ซึ่งเป็นตัวจำกัดจำนวนอุปกรณ์ที่จะมาต่อ ทำให้เกิดแนวความคิดที่จะกำหนดมาตรฐานเพื่อสร้างเป็นพอร์ตที่ทำให้การเชื่อมต่อทั้งหมดอยู่ในรูปแบบเดียวกัน ง่ายต่อการใช้งานของผู้ใช้ทั่วๆ ไป และไม่มีข้อจำกัดในการใช้ IRQ คำตอบของแนวคิดนี้คือ พอร์ต USB และมีคุณสมบัติต่างๆ ที่ทำให้สามารถกำหนดชื่อนี้ให้กับพอร์ตชนิดนี้ได้ดังนี้

1. ใช้คอนเน็กเตอร์เพียงชนิดเดียว ซึ่งมี 2 รูปแบบสำหรับเชื่อมต่ออุปกรณ์ทุกๆ ชนิดเข้ากับคอมพิวเตอร์

2. สามารถเชื่อมต่ออุปกรณ์หลายๆ ชนิดรวมเข้าสู่คอนเน็กเตอร์ตัวเดียว สูงสุด 127 ตัว

3. ไม่เกิดการขัดแย้งกันของการใช้ทรัพยากรของระบบ IRQ

4. ตรวจสอบการเชื่อมต่อและตั้งค่าการทำงานต่างๆ อัตโนมัติ ระหว่างที่เครื่องกำลังทำงานอยู่ (Hot attachment)

- มาตรฐาน USB 1.0/1.1 มีอัตราในการถ่ายทอดข้อมูลความเร็วต่ำ (Low speed) เท่ากับ 1.5 เมกะบิตต่อวินาที (Mbit/sec) และความเร็วเต็มที่ (Full speed) เท่ากับ 12 Mbit/sec

- มาตรฐาน USB 2.0 จะมีอัตราเร็วในการถ่ายทอดเพิ่มขึ้นอีก 1 ระดับคือ ความเร็วสูง (High speed) ซึ่งมีความเร็วสูงถึง 480 Mbit/sec

5. ที่ขาพอร์ต USB มีแรงดันไฟตรง +5 โวลต์ (Volts : V) จ่ายออกมาด้วย ทำให้อุปกรณ์ต่อพ่วงที่ใช้พลังงานไม่มากนัก สามารถใช้แรงดันจากพอร์ต USB นี้เป็นไฟเลี้ยงเพื่อทำงานได้ โดยไม่ต้องอาศัยแหล่งจ่ายไฟจากภายนอกเพิ่มเติมอีก

2.2.1 ภาพรวมการทำงานของพอร์ต USB

1. การส่งข้อมูลภายในบัส USB 1.0/1.1

USB เป็นการส่งข้อมูลที่มีรูปแบบการเชื่อมต่อในระบบบัสคือ อุปกรณ์ทุกๆ ตัวจะต้องส่งสัญญาณรวมกันไปในสายส่งสัญญาณเพียงคู่เดียว ดังนั้นอุปกรณ์ทุกๆ ตัวที่เชื่อมต่อกับบัสจะต้องส่งข้อมูลเรียงลำดับกันไปเพื่อไม่ให้เกิดการชนกันของข้อมูล และเนื่องจาก USB เป็นระบบบัสที่ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สายส่งสัญญาณเพียงคู่เดียวทำให้ในช่วงเวลาหนึ่งๆ จะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียว ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันได้หรือที่เรียกกันว่า การส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ (Half duplex)

จังหวะการรับส่งข้อมูลของระบบบัส USB ทั้งหมดจะถูกควบคุมจากโฮสต์ (Host) ซึ่งก็คือเครื่องคอมพิวเตอร์ที่เป็นจุดรวมของอุปกรณ์ทุกตัวที่เชื่อมต่ออยู่นั่นเอง ดังนั้นจึงไม่สามารถเชื่อมต่อเครื่องคอมพิวเตอร์ 2 เครื่องให้รับหรือส่งข้อมูลถึงกันได้โดยตรง เพราะถ้าคอมพิวเตอร์ทั้งสองเครื่องทำหน้าที่เป็นโฮสต์ทั้งคู่จะเกิดการชนกันของข้อมูลภายในบัส เนื่องจากแต่ละเครื่องก็จะพยายามกำหนดจังหวะในการรับส่งของตัวเองขึ้นมา ดังนั้นจะเชื่อมต่อคอมพิวเตอร์สองเครื่องเข้าด้วยกันผ่าน USB ต้องมีอุปกรณ์ที่เป็นตัวกลางเพื่อซิงโครไนซ์ตัวเองเข้ากับโฮสต์ทั้งสองให้ได้

การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรม โดยทุกๆ 1ms จะเกิดการรับส่งข้อมูลขึ้น 1 เฟรม ในแต่ละเฟรมจะแบ่งย่อยออกเป็นแพ็กเก็ต (Packet) เริ่มต้นการทำงานของแต่ละเฟรม โดยโฮสต์จะส่งสัญญาณเริ่มต้นเฟรม (Start Of Frame : SOF) ออกไปเพื่อให้อุปกรณ์ทุกตัวรู้จังหวะการเริ่มเฟรม หลังจากนั้นโฮสต์ก็จะเริ่มส่งหรือรับข้อมูลต่างๆ ตามที่ได้จัดลำดับความสำคัญไว้ อุปกรณ์ต่างๆ ที่อยู่ภายในบัสจะต้องทำงานตามจังหวะที่โฮสต์กำหนดไว้เท่านั้น การส่งข้อมูลกลับไปยังโฮสต์จะสามารถทำได้ก็ต่อเมื่อได้รับการถามหรือร้องขอจากโฮสต์

แต่เนื่องจากแต่ละเฟรมข้อมูลจะต้องรับส่งเสร็จภายใน 1ms นั่นหมายความว่าข้อมูลของอุปกรณ์ทุกๆ ตัวที่เชื่อมต่อกับบัสจะต้องถูกกำหนดขนาดไม่ให้ใหญ่เกินกว่าที่จะสามารถรับส่งได้ภายใน 1ms และเล็กพอที่จะทำให้อุปกรณ์ทุกๆ ตัวสามารถใช้งานบัสไปพร้อมๆ กันได้ดังนั้นในระบบบัส USB จึงจำเป็นต้องอาศัยซอฟต์แวร์ที่เข้ามาจัดการในด้านนี้ และยังต้องอาศัยฮาร์ดแวร์ที่จะคอยกระจายการส่งและรวบรวมการรับข้อมูลจากอุปกรณ์ทุกๆ ตัวในระบบ ซึ่งซอฟต์แวร์และฮาร์ดแวร์ที่จำเป็นสำหรับระบบ USB มีดังนี้

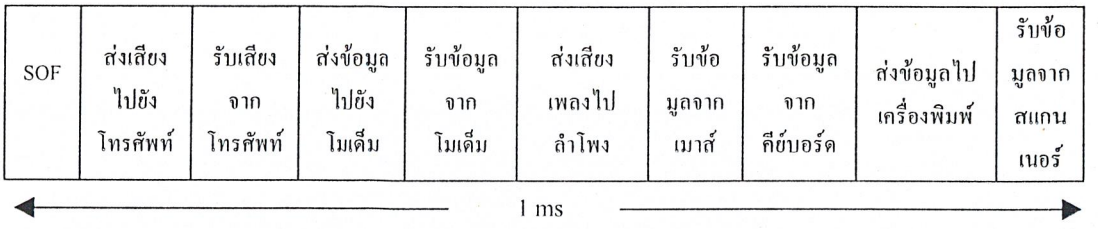
ส่วนซอฟต์แวร์

1. ไดรเวอร์อุปกรณ์ USB (USB device drivers)
2. ไดรเวอร์ USB (USB drivers)
3. ไดรเวอร์โฮสต์คอนโทรลเลอร์ (USB host controller drivers)

ส่วนฮาร์ดแวร์

1. USB โฮสต์คอนโทรลเลอร์ (USB host controller) / รูตฮับ (Root hub)
2. USB ฮับ (USB hub)
3. อุปกรณ์ USB (USB device)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 การจัดลำดับการรับส่งข้อมูลของอุปกรณ์แต่ละตัว

ส่วนประกอบทางซอฟต์แวร์

1. ไดรเวอร์อุปกรณ์ USB

ไดรเวอร์อุปกรณ์ USB คือโปรแกรมเก็บข้อมูลที่จำเป็นในการติดต่อไปยังอุปกรณ์แต่ละตัว เมื่อโปรแกรมใดมีความต้องการจะติดต่อกับอุปกรณ์ต่างๆ จะต้องแจ้งความต้องการนั้นๆ มายังไดรเวอร์อุปกรณ์ USB เนื่องจากตัวไดรเวอร์นี้จะรู้ว่าถ้าต้องการติดต่อกับอุปกรณ์จะต้อง ติดต่อผ่าน เอ็นด์พอยต์ (Endpoint) ไหนด้วยรูปแบบใด (การทำงานของอุปกรณ์ USB จะติดต่อ สั่งงานผ่าน เอ็นด์พอยต์ของตัวอุปกรณ์ ซึ่งอุปกรณ์ต่างๆ จะมีชนิดและจำนวนเอ็นด์พอยต์ที่ต่างกัน) ดังนั้นอุปกรณ์แต่ละตัวก็จะมีไดรเวอร์อุปกรณ์ USB เฉพาะตัวซึ่งเมื่อถึงคราวต้องนำอุปกรณ์นั้นมาต่อใช้งานกับเครื่องคอมพิวเตอร์จริงๆ ก็จะต้องนำไดรเวอร์ตัวเดียวกันมาติดตั้งเพิ่มเข้ากับระบบปฏิบัติการในคอมพิวเตอร์เพื่อให้ระบบรู้จักและติดต่อใช้งานอุปกรณ์ที่ติดตั้งเข้ามาใหม่นี้ได้ เช่น ถ้าต้องการติดต่อเพื่อรับข้อมูลจากคีย์บอร์ด ตัวไดรเวอร์อุปกรณ์ USB จะรู้ว่าต้องรับส่งข้อมูลด้วยอัตราเร็วต่ำ (Slow speed) โดยใช้รูปแบบการถ่ายทอดข้อมูลแบบอินเตอร์รัพต์ (Interrupt transfer type) ผ่านเอ็นด์พอยต์ตัวหนึ่งของคีย์บอร์ด และตรวจสอบข้อมูลการกดเป็นช่วงระยะห่างค่าหนึ่งแต่ในบางอุปกรณ์ที่เป็นอุปกรณ์พื้นฐานของเครื่องคอมพิวเตอร์ เช่น เมาส์และคีย์บอร์ด จะมีการบรรจุไดรเวอร์อุปกรณ์ USB ของอุปกรณ์เหล่านี้ไว้ภายในไบออสของเครื่องคอมพิวเตอร์เรียบร้อยแล้ว จึงไม่ต้องติดตั้งไดรเวอร์เพิ่มเติมสำหรับอุปกรณ์เหล่านี้ เพียงแต่เข้าไปเปิดการทำงาน ไบออสก็จะทำให้เครื่องคอมพิวเตอร์รู้จักอุปกรณ์เหล่านี้เอง

2. ไดรเวอร์ USB

การทำงานของ USB นั้นเป็นการต่อร่วมกันของอุปกรณ์หลายๆ ชนิดบนสายสัญญาณเพียงคู่เดียว ดังนั้นการส่งข้อมูลของอุปกรณ์แต่ละชนิดจะต้องมีการแบ่งสรรปันส่วนกันไปอย่างพอเหมาะพอดี เพื่อให้อุปกรณ์ทุกตัวสามารถทำงานไปได้พร้อมๆ กัน และแน่นอนว่าต้องมีซอฟต์แวร์ที่เข้ามาทำหน้าที่นี้ ซึ่งก็คือไดรเวอร์ USB นั่นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไดรเวอร์อุปกรณ์ USB ของอุปกรณ์แต่ละตัวจะส่งการร้องขอเพื่อการติดต่อ (Request) ลงมายังไดรเวอร์ USB และเมื่อไดรเวอร์ USB รับทราบความต้องการการติดต่อของอุปกรณ์ครบทุกๆ ตัวที่เชื่อมต่ออยู่กับบัสแล้ว ก็จะพิจารณาว่า ในรอบการรับส่งข้อมูลหนึ่งๆ นั้นอุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้มากเท่าใด หากปริมาณข้อมูลที่ต้องการรับส่งมีขนาดมากก็จะตัดแบ่งออกเป็นส่วนๆ แล้วเก็บไว้เพื่อสำรองในรอบถัดไป โดยปริมาณข้อมูลที่ส่งได้ของอุปกรณ์แต่ละตัวจะถูกพิจารณาจากชนิดของการถ่ายทอดข้อมูล (Transfer type) ว่าอุปกรณ์ใดใช้การถ่ายทอดข้อมูลแบบใด และการรับส่งข้อมูลชนิดนั้นมีลำดับความสำคัญมากน้อยเพียงใด

3. ไดรเวอร์โฮสต์คอนโทรลเลอร์

ไดรเวอร์อุปกรณ์ USB พิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้าง จะส่งข้อมูลของอุปกรณ์แต่ละตัวที่จะติดต่อในรอบการติดต่อนั้นๆ มายังไดรเวอร์โฮสต์คอนโทรลเลอร์จากนั้น ไดรเวอร์โฮสต์คอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลของอุปกรณ์แต่ละชนิดลงเป็นเฟรมข้อมูลเพิ่มเติมส่วนประกอบต่างๆ ของเฟรมข้อมูลให้ครบตามมาตรฐานการถ่ายทอดข้อมูลแบบ USB แล้วส่งข้อมูลทั้งหมดไปยังฮาร์ดแวร์ USB โฮสต์คอนโทรลเลอร์เพื่อส่งข้อมูลทั้งหมดไปยังอุปกรณ์ต่างๆ

ส่วนประกอบทางฮาร์ดแวร์

1. USB โฮสต์คอนโทรลเลอร์ / USB รูตฮับ

USB โฮสต์คอนโทรลเลอร์มีหน้าที่สร้างสัญญาณข้อมูลทางไฟฟ้า แล้วส่งต่อไปยังรูตฮับเพื่อกระจายออกไปยังอุปกรณ์ต่างๆ โดยมันจะสร้างสัญญาณข้อมูลการติดต่อรูปแบบต่างๆ ตามที่ไดรเวอร์โฮสต์คอนโทรลเลอร์กำหนดมาให้ จากนั้นแปลงข้อมูลที่จะส่งจากแบบขนานเป็นแบบอนุกรมเพื่อใช้ในการส่งต่อไป เมื่อสัญญาณที่ต้องการส่งมาถึงรูตฮับ รูตฮับจะส่งสัญญาณนั้นออกไปยังบัสเพื่อส่งต่อไปยังอุปกรณ์ต่างๆ นอกจากนั้นรูตฮับยังมีหน้าที่สำคัญอีก 4 อย่างคือ

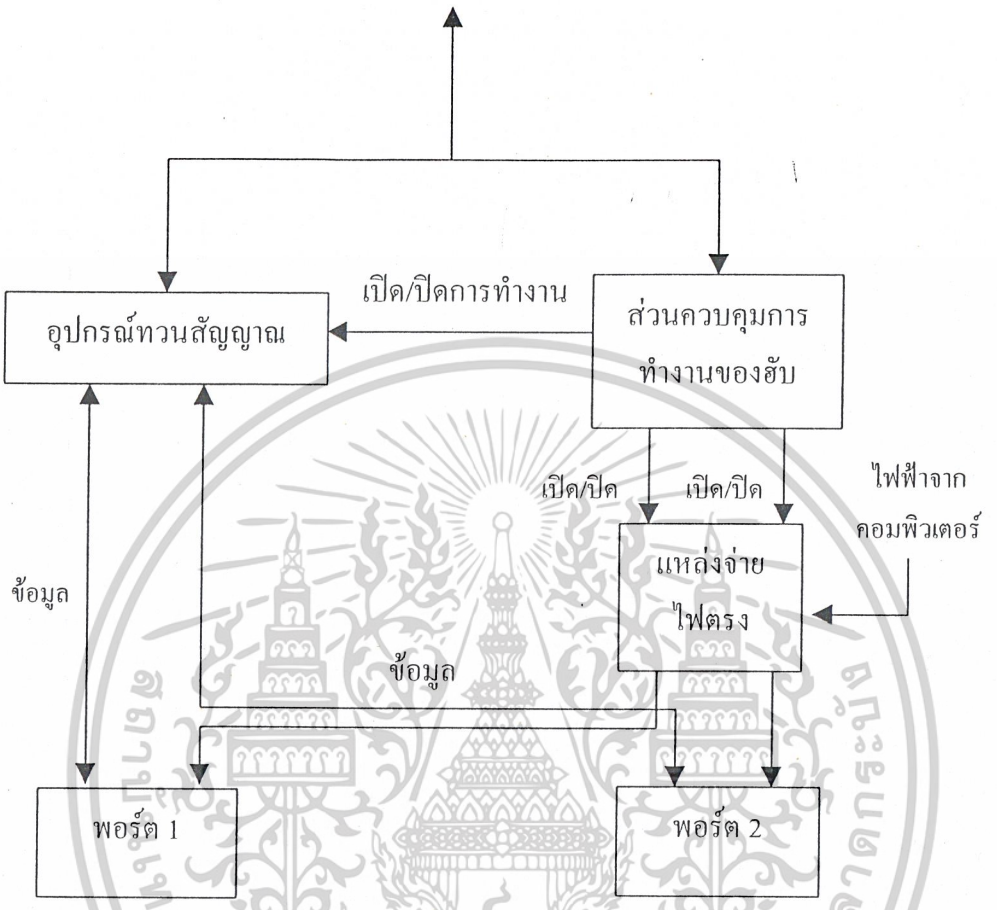
1. ควบคุมการใช้พลังงานของอุปกรณ์ที่มาต่อ

2. ตรวจสอบการเชื่อมต่อของอุปกรณ์ว่ามีอุปกรณ์ต่ออยู่หรือไม่

3. เปิดหรือเอ็นเอเบิลการใช้งานพอร์ตเมื่อมีอุปกรณ์ต่ออยู่ หรือปิดหรือดิสเอเบิลการใช้งานเมื่อปลดอุปกรณ์ออกไปแล้ว

4. รายงานสถานะของแต่ละพอร์ตเมื่อไดรเวอร์โฮสต์คอนโทรลเลอร์ร้องขอมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.2 แผนผังการทำงานของ USB รูตฮับอย่างง่าย

2. USB ฮับ

หน้าที่หลักๆ ของ USB ฮับ คือ ขยายการเชื่อมต่อให้อุปกรณ์จำนวนมากๆ สามารถเชื่อมต่อเข้ากับระบบบัสได้ โดยการทำงานหลักของ USB ฮับนั้นมีอยู่ 2 ส่วนคือ ทำหน้าที่เป็นตัวทวนสัญญาณ (Repeater) และตัวจัดการพลังงาน (Power management)

ในส่วนของการทวนสัญญาณ USB ฮับจะต้องรับสัญญาณจากโฮสต์มา แล้วส่งกระจายออกไปยังพอร์ตทุกๆ พอร์ต แล้วจับมารวมกันเพื่อส่งกลับไปที่โฮสต์ สำหรับส่วนของการจัดการพลังงานนั้นมีหน้าที่เหมือนกับรูตฮับก็คือ ตรวจสอบว่ามีการต่ออยู่ของอุปกรณ์ที่พอร์ตใดบ้าง หากมีอุปกรณ์ต่ออยู่ก็เปิดใช้งานพอร์ตนั้นๆ หากไม่มีอุปกรณ์ต่ออยู่ก็ปิดการใช้งาน ตรวจสอบการเชื่อมต่อหรือปลดออกของอุปกรณ์เพื่อรายงานผลเมื่อโฮสต์คอนโทรลเลอร์ร้องขอและป้องกันอุปกรณ์ที่ต่ออยู่ในแต่ละพอร์ตไม่ให้ดึงกระแสไฟฟ้าเกินกว่าที่กำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. อุปกรณ์ USB

ส่วนประกอบทางฮาร์ดแวร์ส่วนสุดท้ายที่ต้องรู้จักคือ อุปกรณ์ USB ซึ่งก็คืออุปกรณ์ต่างๆ ที่เชื่อมต่อกับคอมพิวเตอร์ด้วยพอร์ต USB สามารถแบ่งเป็น 3 ชนิดตามความเร็วในการถ่ายทอดข้อมูลคือ

1. อุปกรณ์ความเร็วต่ำ (Low-speed devices) ถ่ายทอดข้อมูลด้วยความเร็ว 1.5 Mbit/sec

2. อุปกรณ์ความเร็วเต็มที่ (Full-speed device) รับส่งข้อมูลด้วยความเร็ว 12 Mbit/sec

3.อุปกรณ์ความเร็วสูง (High-speed device) รับส่งข้อมูลด้วยความเร็ว 480 Mbit/sec

ปัจจุบันนี้อุปกรณ์ USB ที่มีจำหน่ายอยู่ตามท้องตลาดมีอยู่เป็นจำนวนมาก อาทิ จอยสติ๊ก เมาส์และคีย์บอร์ด เหล่านี้คืออุปกรณ์ USB ความเร็วต่ำ ส่วนจอมอนิเตอร์ กล้องถ่ายรูปดิจิตอล เครื่องพิมพ์ ลำโพง ซีดีรอมไดรฟและเครื่องเล่น MP3 จัดเป็นอุปกรณ์ USB ความเร็วสูงอุปกรณ์บางตัวจะบรรจุความสามารถของ USB ฮับเข้าไปด้วย ทำให้สามารถนำอุปกรณ์อื่นๆ มาเชื่อมต่อได้เหมือนกับการต่อเข้ากับฮับ อุปกรณ์ลักษณะนี้เรียกว่า Compound USB Device ตัวอย่างอุปกรณ์ที่มีฮับอยู่ภายในได้แก่ จอมอนิเตอร์หรือเครื่องพิมพ์ เป็นต้น

นอกจากการแบ่งชนิดของอุปกรณ์ตามความเร็วในการถ่ายทอดข้อมูลแล้ว อาจจะแบ่งกลุ่มตามการใช้งานของตัวอุปกรณ์เองก็ได้ ซึ่งสามารถแบ่งได้อีก 2 ชนิดคือ

1. อุปกรณ์ที่ใช้ไฟเลี้ยงจากบัส (Bus powered device) คือ อุปกรณ์ที่ใช้ไฟเลี้ยงจากบัสโดยตรงไม่ต้องมีแหล่งจ่ายไฟภายนอกเพิ่มเติม

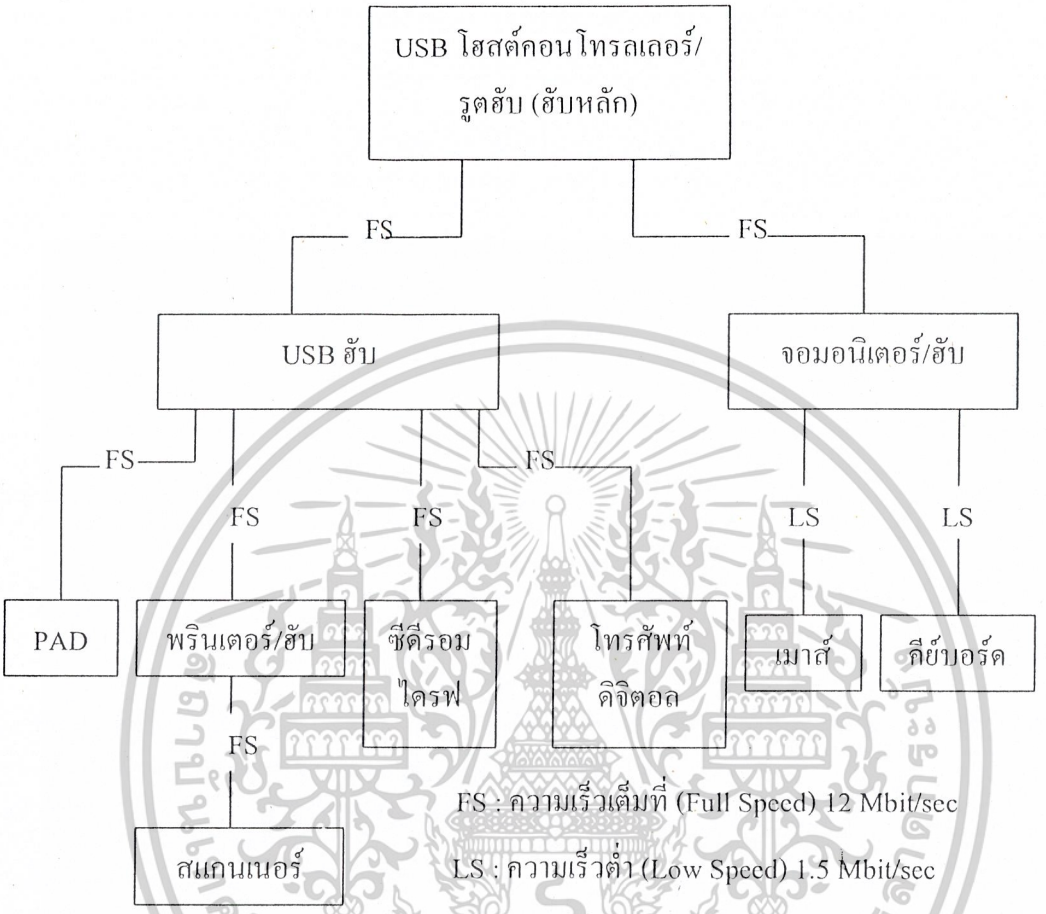
2. อุปกรณ์ที่ใช้ไฟเลี้ยงจากตัวเอง (Self powered device) คือ อุปกรณ์ที่มีแหล่งจ่ายไฟในตัวไม่ต้องอาศัยไฟเลี้ยงจากบัส

2. โครงสร้างการเชื่อมต่อ (Topology)

โครงสร้างการเชื่อมต่อของ USB นั้นเป็นแบบสตาร์ (Star) ดังรูป 2.3 โดยฮับทำหน้าที่เป็นศูนย์กลางจุดเชื่อมต่อไปยังอุปกรณ์ต่างๆ ในแต่ละระดับชั้น สายเชื่อมต่อแต่ละเส้นเป็นการต่อแบบจุดต่อจุด (Point to point)

เนื่องจากพอร์ต USB มีรูปแบบการเชื่อมต่อเป็นระบบบัส แต่มีโครงสร้างการเชื่อมต่อแบบสตาร์ ซึ่งอาจสร้างความสับสนในจุดนี้ได้ จึงขอขยายความหมายและเปรียบเทียบให้เห็นความแตกต่างตรงจุดนี้ จากคำว่า "โครงสร้างการเชื่อมต่อ" จะอธิบายถึงลักษณะการต่อสายสัญญาณของอุปกรณ์แต่ละตัวเข้าด้วยกัน ซึ่งก็คือ ลักษณะการต่อสายไฟระหว่างกันทำอย่างไร แต่คำว่า "รูปแบบการเชื่อมต่อ" นั้นจะอธิบายถึงการเดินทางของสัญญาณข้อมูลภายในสายไฟที่เชื่อมต่อกันอยู่จากอุปกรณ์ตัวหนึ่งไปยังอุปกรณ์แต่ละตัวที่ต่อเชื่อมอยู่ ความหมายโดยรวมก็คือ พอร์ต USB ต่อสายเข้าด้วยกันแบบสตาร์แต่รับส่งข้อมูลแบบบัส (โครงสร้างคล้ายกับระบบอีเทอร์เน็ต)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 แสดงโครงสร้างการเชื่อมต่อของ USB

3. การติดต่อระหว่างอุปกรณ์และโฮสต์

การถ่ายทอดสัญญาณ (Transfer type) ของบัส USB นั้นแบ่งออกเป็น 4 ชนิด ตามขนาดชนิดของข้อมูลและจังหวะการส่งข้อมูลดังนี้

1. การถ่ายทอดสัญญาณแบบไอโซโครนัส (Isochronous transfer)

2. การถ่ายทอดสัญญาณแบบบัลก์ (Bulk trasfer)

3. การถ่ายทอดสัญญาณแบบอินเตอร์รัพต์ (Interrupt transfer)

4. การถ่ายทอดสัญญาณควบคุม (Control transfer)

การถ่ายทอดสัญญาณ 3 ชนิดแรกใช้สำหรับข้อมูลทั่วไปที่ต้องการรับหรือส่งไปยังตัวอุปกรณ์ ส่วนการถ่ายทอดสัญญาณแบบที่ 4 การถ่ายทอดสัญญาณแบบไอโซโครนัสใช้ถ่ายทอด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลที่ต้องการความต่อเนื่องสูง เช่น ข้อมูลเสียงเพลง ส่วนการถ่ายทอดสัญญาณแบบบัลก์ใช้สำหรับการถ่ายทอดข้อมูลที่มีปริมาณมากๆ แต่ไม่ต้องการความต่อเนื่องของข้อมูล ในขณะที่การถ่ายทอดสัญญาณแบบอินเตอร์รัพต์ใช้สำหรับถ่ายทอดข้อมูลที่มีจำนวนน้อยครั้งและมีปริมาณของข้อมูลไม่มากในการสั่งงานอุปกรณ์แต่ละครั้งนั้น โฮสต์จะต้องระบุเป้าหมายปลายทางของข้อมูลที่ต้องการจะรับหรือส่ง เป้าหมายปลายทางที่ว่านี้เป็นกลุ่มของรีจิสเตอร์ที่อยู่ในอุปกรณ์แต่ละตัวซึ่งเรียกว่า เอ็นด์พอยต์ โดยแต่ละเอ็นด์พอยต์จะรองรับการถ่ายทอดสัญญาณชนิดต่างๆ 4 ชนิดข้างต้นแยกกันออกไป ดังนั้นอุปกรณ์แต่ละตัวจะมีจำนวนเอ็นด์พอยต์มากกว่า 1 เอ็นด์พอยต์ เพื่อรองรับการทำงานรูปแบบต่างๆ ยกตัวอย่าง ซีดีรอม USB แบบอ่านเขียนได้จะต้องมีเอ็นด์พอยต์ที่รองรับการถ่ายทอดสัญญาณควบคุมเพื่อรับคำสั่งจากโฮสต์ 1 เอ็นด์พอยต์ พร้อมกันนั้นยังต้องมีเอ็นด์พอยต์ที่รองรับการถ่ายทอดสัญญาณแบบบัลก์เพื่อส่งข้อมูลที่อ่านได้หรือรับข้อมูลเพื่อเขียนแผ่นซีดี 1 เอ็นด์พอยต์ และต้องมีเอ็นด์พอยต์ที่รองรับการถ่ายทอดสัญญาณไอโซโครนัสเพื่อส่งข้อมูลเพลงซึ่งมีความต่อเนื่องในกรณีที่เล่นแผ่นซีดีเพลงอีก 1 เอ็นด์พอยต์ เป็นต้น

4. ดิสคริปเตอร์ : ความหมาย, ชนิดและรูปแบบการทำงาน

อุปกรณ์แต่ละตัวมีคุณสมบัติและการทำงานที่แตกต่างกัน โฮสต์จำเป็นต้องรู้คุณสมบัติทั้งหมดของอุปกรณ์แต่ละตัวเพื่อให้การสั่งงานเป็นไปอย่างถูกต้อง และเนื่องจากบัสข้อมูลทั้งหมดจะถูกใช้งานรับส่งข้อมูลร่วมกันระหว่างอุปกรณ์ทุกๆ ตัว สิ่งที่โฮสต์จำเป็นต้องรู้อีกอย่างคือปริมาณข้อมูลที่ต้องการส่ง (Bandwidth) ของอุปกรณ์แต่ละตัวในบัส ดังนั้นเมื่อมีอุปกรณ์ตัวใหม่ต่อเข้ากับบัส โฮสต์ต้องอ่านข้อมูลต่างๆ ที่จำเป็นเข้ามาเพื่อใช้อ้างอิงในการสั่งงานอุปกรณ์ และใช้พิจารณาว่าระบบบัสสามารถรองรับอุปกรณ์ที่มาเชื่อมต่อใหม่ได้หรือไม่ ข้อมูลเหล่านี้รวมเรียกว่าดิสคริปเตอร์ของอุปกรณ์ (Device descripters)

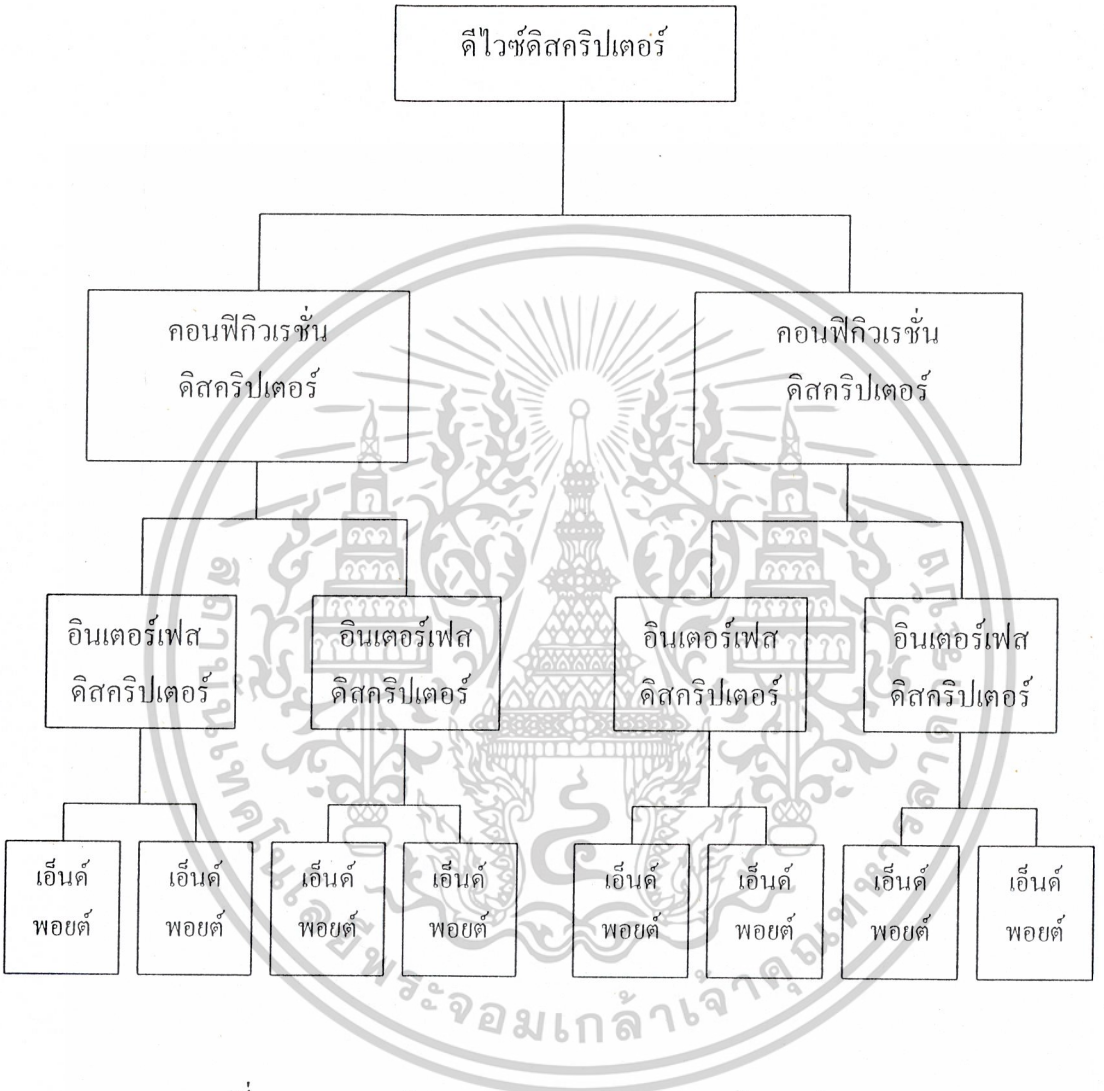
อุปกรณ์แต่ละตัวจะแจ้งรายละเอียดของตัวเองให้โฮสต์รู้ผ่านดิสคริปเตอร์ชนิดต่างๆ ซึ่งได้รับการแบ่งแยกเป็นชนิดตามข้อมูลที่จะแจ้งกลับไปยังโฮสต์ โดยการแบ่งแยกชนิดของดิสคริปเตอร์นั้นจะจัดเป็นลำดับชั้นตามรูปแบบ เช่น แบ่งการทำงานออกเป็น 2 โหมด แต่ละโหมดมีหน้าที่การทำงานแตกต่างกัน และแต่ละหน้าที่ก็จะใช้กลุ่มของเอ็นด์พอยต์ที่แตกต่างกัน สามารถสรุปได้ดังนี้

4.1 ดีไวซ์ดิสคริปเตอร์ (Device descripter)

ในอุปกรณ์แต่ละตัวจะมีดีไวซ์ดิสคริปเตอร์เพียง 1 ชุดเท่านั้น ภายในจะระบุข้อมูลที่ใช้ในการเชื่อมต่อขั้นแรก (Default communication pipe) เพื่อใช้ในการกำหนดข้อมูลสถานะของข้อมูลเข้ากับโฮสต์ นอกจากนั้นยังเก็บข้อมูลของข้อกำหนดในโหมดทำงานต่างๆ ของอุปกรณ์รวมถึงจำนวนคอนฟิกิวเรชั่นด้วยเพราะในครั้งแรกที่อุปกรณ์เชื่อมต่อเข้ากับบัสนั้นโฮสต์ไม่มีทางรู้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

เลยว่าต้องติดต่อกับอุปกรณ์ที่เอ็นด์พอยต์ใด จึงจำเป็นต้องขอข้อมูลจากส่วนนี้ก่อนที่จะติดต่อกับส่วนอื่น



รูปที่ 2.4 แสดงระดับการทำงานของดิสคริปเตอร์ในพอร์ต USB

4.2 คอนฟิกิวเรชั่นดิสคริปเตอร์ (Configuration descripter)

ใช้เก็บข้อมูลที่จำเป็นของการทำงานแต่ละโหมดการทำงานและเก็บจำนวนแต่ละเฟสที่ใช้งานในโหมดนั้นๆ เช่น อุปกรณ์บางตัวมีการทำงาน 2 โหมดคือ โหมดใช้พลังงานสูง และโหมดประหยัดพลังงาน ดิสคริปเตอร์ตัวนี้จะเก็บข้อมูลที่จำเป็นในการตั้งค่าต่างๆ ของโฮสต์เมื่อต้องการเลือกใช้โหมดการทำงานแต่ละโหมดของอุปกรณ์

4.3 อินเตอร์เฟสดิสคริปเตอร์ (Interface descripter)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในแต่ละโหมดการทำงานหรือคอนฟิกิวเรชั่นอาจจะมีการเชื่อมต่อหรืออินเตอร์เฟส 1 แบบหรือมากกว่าเพื่อใช้งานในหน้าที่ต่างๆ ตัวอย่างที่เห็นได้ชัดที่สุดก็คือซีดีรอม โดยในตัวซีดีรอมจะมีการรับส่งข้อมูลปริมาณมากๆ (Mass storage) การส่งข้อมูลเสียงออดิโอ และการส่งข้อมูลภาพจะเห็นได้ว่า มีอินเตอร์เฟสที่ใช้งานแยกกัน ภายในอินเตอร์เฟสดิสคริปเตอร์จะบรรจุข้อมูลการใช้งานอินเตอร์เฟสนั้นๆ โดยข้อมูลเหล่านี้จะระบุว่าอุปกรณ์ถูกจัดอยู่ในคลาส (Class) หรือคลาสย่อย (Subclass) ใด และมีเอ็นด์พอยต์จำนวนเท่าใดที่ใช้งานอินเตอร์เฟสนี้บ้าง

4.4 เอ็นด์พอยต์ดิสคริปเตอร์ (Endpoint descripter)

ใช้เก็บข้อมูลคุณสมบัติของแต่ละเอ็นด์พอยต์ เช่น ใช้การถ่ายทอดสัญญาณแบบใด (บัลก์ ไอโซโครนัส อินเตอร์รัพต์หรือสัญญาณควบคุม) และถ่ายทอดข้อมูลได้มากที่สุดครั้งละเท่าใด

4.5 สตริงดิสคริปเตอร์ (String descripter)

เป็นดริสคริปเตอร์ที่ไม่ได้อยู่ภายในโครงสร้างตามรูปที่ 2.4 เพราะดิสคริปเตอร์ชนิดนี้จะเก็บข้อมูลตัวอักษรที่สามารถอ่านเข้าใจได้ไว้อธิบายส่วนต่างๆ ของดิสคริปเตอร์ทั้งสี่ตัวข้างต้น เช่น เก็บชื่อบริษัทผู้ผลิต หรือหมายเลขประจำตัวของอุปกรณ์ เป็นต้น

4.6 คลาสสเปกซิฟิกดิสคริปเตอร์ (Class-specific descripter)

เป็นดิสคริปเตอร์พิเศษที่มีเฉพาะในอุปกรณ์บางตัวที่จัดอยู่ในบางคลาสเท่านั้น ภายในเก็บข้อมูลเฉพาะของคลาสนั้นๆ ที่อยู่นอกเหนือจากดิสคริปเตอร์พื้นฐาน 4 ชนิดแรก

ดังนั้นจึงพอสรุปการใช้งานของดิสคริปเตอร์พื้นฐานได้ดังนี้

1. ข้อมูลทั่วไปของตัวอุปกรณ์จะเก็บอยู่ในดิสคริปเตอร์ของอุปกรณ์หรือดีไวซ์ดิสคริปเตอร์
2. ข้อมูลการทำงานของแต่ละโหมดเก็บอยู่ในคอนฟิกิวเรชั่นดิสคริปเตอร์
3. ข้อมูลหน้าที่การทำงานเก็บอยู่ในดิสคริปเตอร์การเชื่อมต่อหรืออินเตอร์เฟสดิสคริปเตอร์
4. ข้อมูลการทำงานของแต่ละเอ็นด์พอยต์จะเก็บอยู่ในเอ็นด์พอยต์ดิสคริปเตอร์

4.7 การจัดการกับอุปกรณ์บนบัส USB ที่มีความเร็วต่างกัน

อุปกรณ์ USB แบ่งตามความเร็วของการถ่ายทอดข้อมูลได้ 3 ชนิดคือ อุปกรณ์ความเร็วสูงที่จะถ่ายทอดข้อมูลที่อัตรา 480 Mbit/sec อุปกรณ์ความเร็วเต็มที่จะถ่ายทอดข้อมูลที่อัตรา 12 Mbit/sec และอุปกรณ์ความเร็วต่ำถ่ายทอดข้อมูลที่อัตรา 1.5 Mbit/sec แต่เนื่องจากอุปกรณ์ทั้ง 3 ประเภทนี้ต่ออยู่บนบัสเดียวกันทั้งหมด ดังนั้นพอร์ตของฮับที่ให้อุปกรณ์ USB เข้ามาเชื่อมต่อจะต้องรองรับการทำงานได้ทั้ง 3 แบบ และด้วยความที่เป็นระบบบัส อุปกรณ์ทุกๆตัวจะได้รับข้อมูลทุกๆแพ็กเก็ตที่ส่งเข้ามา ไม่ว่าจะเป็นการส่งด้วยความเร็วสูงหรือต่ำ ดังนั้นจึงต้องมีการจัดการจราจรระหว่างข้อมูลที่ส่งด้วย โดยตั้งเป็นข้อกำหนดว่า พอร์ตของอุปกรณ์ความเร็วต่ำต้องไม่เปิดทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จนกว่าจะได้รับสัญญาณ “ปรีแอมเบิล” (Preamble) จากโฮสต์ โดยหลังจากได้รับสัญญาณปรีแอมเบิลแล้ว ฮับจะเปลี่ยนความเร็วในการถ่ายทอดข้อมูลไปสู่โหมดความเร็วต่ำ แล้วเปิดการทำงานของพอร์ตที่มีความเร็วต่ำ ทำให้อุปกรณ์ที่เชื่อมต่อด้วยความเร็วต่ำได้รับข้อมูล ในทางกลับกัน หลังจากได้รับสัญญาณปรีแอมเบิลแล้วอุปกรณ์ความเร็วเต็มที่ก็จะทราบทันทีว่าข้อมูลที่ตามหลังมาจะอยู่ในโหมดความเร็วต่ำ ไม่ต้องอ่านและตีความข้อมูลเหล่านั้น

5. การส่งสัญญาณในบัส USB

สัญญาณที่ปรากฏบนสายสัญญาณระหว่าง รูตฮับและอุปกรณ์จะส่งไปในแบบสัญญาณผลต่าง (Differential signaling) เพื่อลดการแพร่กระจายสนามแม่เหล็กไฟฟ้า (Electromagnetic Interference : EMI) เนื่องจากตามธรรมชาติของสัญญาณไฟฟ้า เมื่อมีการเปลี่ยนแปลงระดับสัญญาณด้วยความเร็วมากๆ (ก็คือการส่งข้อมูลที่ความเร็วมากๆ) จะทำให้เกิดการแพร่กระจายของสนามแม่เหล็กและสนามไฟฟ้าออกมารอบๆ สายส่งสัญญาณ ซึ่งอาจรบกวนการรบกวนการทำงานของอุปกรณ์ต่างๆ รอบข้างได้ด้วยการส่งสัญญาณแบบนี้จะทำให้การเปลี่ยนแปลงของสัญญาณในสายนำสัญญาณเกิดขึ้นพร้อมกันและเกิดในลักษณะตรงข้าม ทำให้สนามแม่เหล็กไฟฟ้าที่เกิดขึ้นหักล้างกัน ไม่แพร่ออกมาภายนอก

กระบวนการกำหนดการทำงานของอุปกรณ์เมื่อเริ่มการทำงานของ USB หรือเมื่อมีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้ามาในระบบ ลำดับขั้นตอนคร่าวๆ ในการทำงานจะเป็นดังนี้

1. ฮับตรวจสอบพบว่ามีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้าสู่ระบบแล้วแจ้งผลกลับไปยังโฮสต์คอนโทรลเลอร์

2. โฮสต์คอนโทรลเลอร์สั่งให้ฮับเปิดการทำงานของแหล่งจ่ายไฟในโหมดประหยัด เพื่อให้อุปกรณ์ที่อาศัยพลังงานจากบัสสามารถทำงานได้

3. โฮสต์คอนโทรลเลอร์สั่งให้ฮับรีเซตพอร์ตที่อุปกรณ์ตัวใหม่มาเชื่อมต่อเพื่อทำให้อุปกรณ์รีเซตค่าแอดเดรสและเอ็นด์พอยต์ของตนเองให้เป็นค่าเริ่มต้น (Default)

4. โฮสต์อ่านดิสคริปเตอร์ต่างๆ จากตัวอุปกรณ์และพิจารณาว่าทรัพยากรของระบบพอเพียงต่อความต้องการของตัวอุปกรณ์หรือไม่ ทรัพยากรในที่นี้คือ พลังงานไฟฟ้าและปริมาณข้อมูลที่จะส่งของตัวอุปกรณ์ หากพิจารณาแล้วว่าไม่สามารถทำงานได้ก็จะสั่งให้ฮับปิดการทำงานของพอร์ต

5. เมื่อโฮสต์พิจารณาแล้วว่าสามารถให้บริการแก่อุปกรณ์ตัวที่มาเชื่อมต่อได้ จะควบคุมให้แหล่งจ่ายไฟจ่ายพลังงานตามที่อุปกรณ์ต้องการ รวมไปถึงการตั้งค่าแอดเดรสและกำหนดค่าต่างๆ

6. หลังจากตั้งค่าเรียบร้อยแล้ว คอมพิวเตอร์ก็จะรู้จักกับอุปกรณ์ตัวใหม่และสามารถติดต่อกับอุปกรณ์ได้ทันทีโดยไม่ต้องปิดและเปิดคอมพิวเตอร์ใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

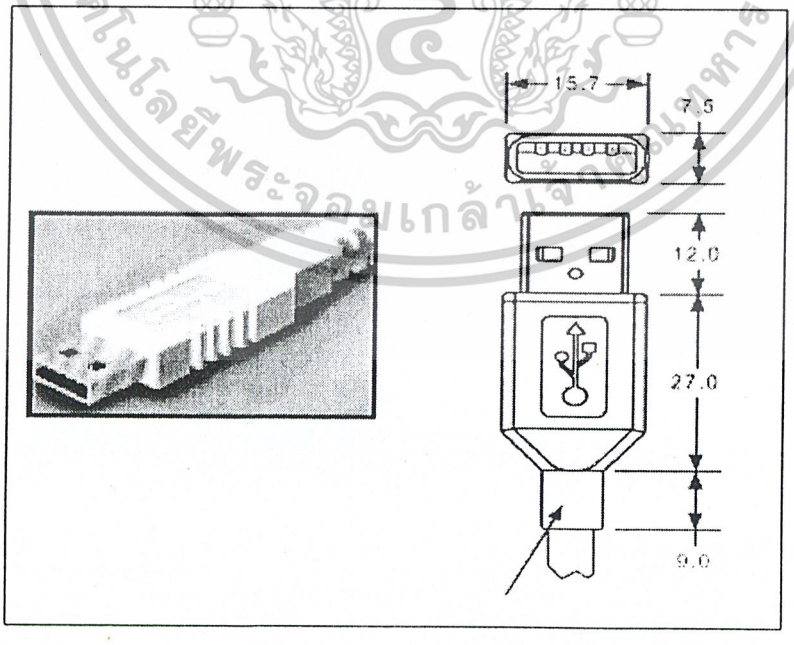
2.2.2 สายเชื่อมต่อและคอนเน็กเตอร์ของพอร์ต USB

1. คอนเน็กเตอร์ของพอร์ต USB

สายเชื่อมต่อ USB ถูกออกแบบมาสำหรับการเชื่อมต่อระหว่างตัวอุปกรณ์ USB กับฮับ ซึ่งอาจจะเป็นรูตฮับซึ่งอยู่หลังเครื่องคอมพิวเตอร์ ฮับที่รวมอยู่ในตัวอุปกรณ์ (Compound device) หรือฮับที่เป็นตัวเดี่ยวๆ (USB hub) อุปกรณ์หลายชนิดมีสายต่อด้านอุปกรณ์ไว้อย่างถาวรด้านหนึ่ง และมีคอนเน็กเตอร์ตัวผู้ไว้สำหรับต่อเข้ากับฮับอีกด้านหนึ่ง แต่มีอุปกรณ์อีกหลายชนิดที่ไม่มีสายต่อถาวรไว้โดยที่ตัวอุปกรณ์จะมีคอนเน็กเตอร์ USB ตัวเมียไว้ เพื่อใช้งานกับสาย USB ที่มีหัวคอนเน็กเตอร์ทั้งสองด้าน จากอุปกรณ์ที่ไม่มีสายถาวรต่อไว้จะเห็นได้ว่าต้องใช้สายเชื่อมต่อซึ่งมีคอนเน็กเตอร์ทั้งสองด้านซึ่งถ้าใช้คอนเน็กเตอร์แบบเดียวกันทั้งหมดอาจทำให้ผู้ใช้นำคอนเน็กเตอร์ด้านหนึ่งต่อเข้ากับฮับพอร์ตหนึ่ง แล้วต่อปลายอีกข้างเข้ากับฮับอีกพอร์ตหนึ่งได้ (เพราะฮับจะมีพอร์ตอยู่หลายพอร์ต) เพื่อป้องกันเหตุการณ์เช่นนี้จึงมีการกำหนดมาตรฐานของคอนเน็กเตอร์ USB ให้มี 2 รูปแบบดังนี้

1.1 คอนเน็กเตอร์อนุกรม A

เป็นคอนเน็กเตอร์ด้านฮับที่เชื่อมต่อระหว่าง USB พอร์ตของฮับ (ทั้งรูตฮับและ USB ฮับทั่วไป) กับสายเชื่อมต่อจากตัวอุปกรณ์ นั่นคือคอนเน็กเตอร์ตัวเมียจะติดตั้งอยู่กับฮับและคอนเน็กเตอร์ตัวผู้จะติดอยู่กับสายที่ต่อออกมาจากตัวอุปกรณ์ ดังในรูปที่ 2.5

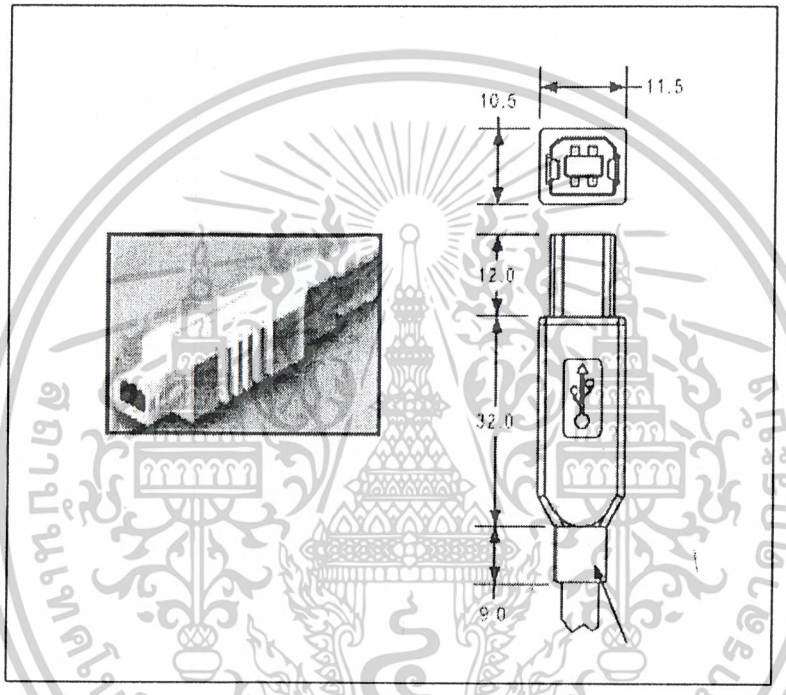


รูปที่ 2.5 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม A

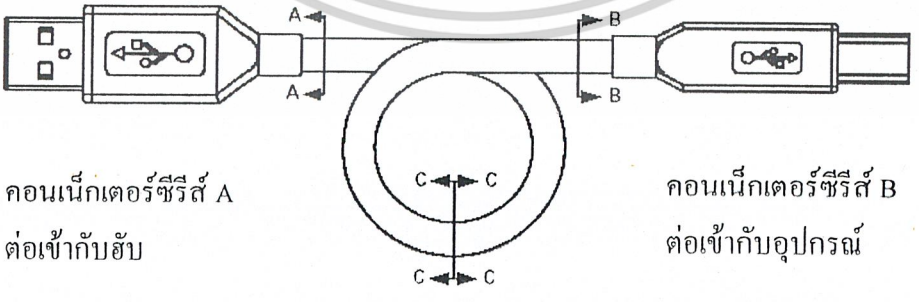
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 คอนเน็กเตอร์อนุกรม B

เป็นคอนเน็กเตอร์ด้านอุปกรณ์ที่เชื่อมต่อสายเข้ากับตัวอุปกรณ์ USB นั่นคือ คอนเน็กเตอร์ตัวเมียจะติดตั้งอยู่ในตัวอุปกรณ์ USB ส่วนคอนเน็กเตอร์ตัวผู้ก็จะอยู่ที่สายที่ต่อออกมาจากฮับ (ถ้าอุปกรณ์ใดที่มีสายต่อออกมาจากตัวอุปกรณ์อย่างถาวรจะไม่มีการใช้คอนเน็กเตอร์แบบนี้) ดังในรูปที่ 2.6



รูปที่ 2.6 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม B



รูปที่ 2.7 แสดงการใช้งานสาย USB ที่ใช้คอนเน็กเตอร์ทั้งสองรูปแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

2. หน้าสัมผัสคอนเน็กเตอร์ USB

ถึงแม้จะมีการแบ่งคอนเน็กเตอร์ออกเป็น 2 แบบ แต่ทั้งสองรูปแบบนั้นใช้สายนำสัญญาณเหมือนกัน โดยใช้สาย 2 เส้นในการส่งไฟเลี้ยงให้แก่อุปกรณ์ และสายอีก 2 เส้นสำหรับรับและส่งข้อมูล แต่เนื่องจากระบบ USB นั้นออกแบบมาเพื่อให้สามารถเชื่อมต่อหรือปลดอุปกรณ์ออกจากระบบได้ในระหว่างการใช้งาน ดังนั้นหน้าสัมผัสของคอนเน็กเตอร์จึงมีการออกแบบให้มีความพิเศษเพื่อให้รองรับคุณสมบัติ ดังกล่าวได้ นั่นคือหน้าสัมผัสของสาย 2 เส้นที่ใช้ส่งไฟเลี้ยงจะยื่นออกมายาวกว่าหน้าสัมผัสที่ใช้รับส่งข้อมูล หน้าสัมผัสสำหรับไฟเลี้ยงยาว 7.41 มิลลิเมตร (mm) หน้าสัมผัสสำหรับรับส่งข้อมูลยาว 6.41 mm การออกแบบคอนเน็กเตอร์เช่นนี้จะทำให้ตัวอุปกรณ์ได้รับไฟเลี้ยงก่อนที่จะได้รับสัญญาณข้อมูลเมื่อเชื่อมต่ออุปกรณ์ตัวใหม่เข้าระบบ ทำให้อุปกรณ์ที่เชื่อมต่อเข้าไปใหม่สามารถทำงานได้ทันทีโดยไม่ได้รับความเสียหาย เพราะถ้าหากอุปกรณ์ได้รับสัญญาณข้อมูลก่อนที่ได้รับไฟเลี้ยงจะเกิดความเสียหายแก่ไอซีที่อยู่ภายในได้ ขาของคอนเน็กเตอร์แต่ละตัวทำหน้าที่ตามที่แสดงในตารางที่ 2.1

ตารางที่ 2.1 หน้าที่ของขาคอนเน็กเตอร์แต่ละขา

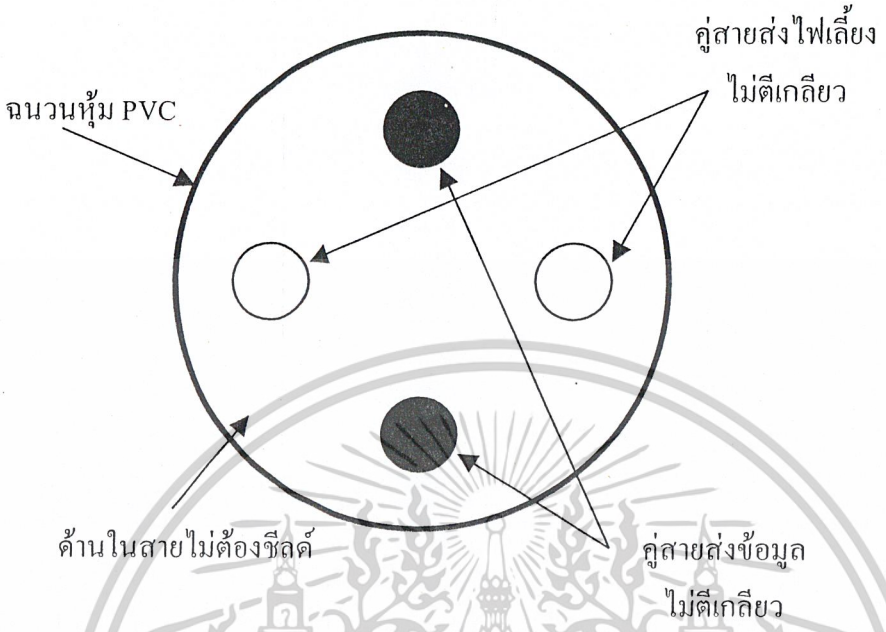
หมายเลขขาสัญญาณ	ชื่อขาสัญญาณ	สีของสายสัญญาณ
1	ไฟเลี้ยง +5V	แดง
2	D- (ข้อมูลลบ)	ขาว
3	D+ (ข้อมูลบวก)	เขียว
4	กราวด์	ดำ

3. สายนำสัญญาณ USB

การถ่ายทอดข้อมูลภายในสาย USB นั้นมีความเร็วสูงในระดับ 1 Mbit/sec ถึง 12 Mbit/sec การถ่ายทอดข้อมูลที่ความเร็วสูงขนาดนี้จะเกิดการแพร่กระจายสนามแม่เหล็กไฟฟ้าออกมาจากสายส่ง ทำให้ต้องมีการกำหนดคุณสมบัติของสายส่งในหลายๆ ด้านเพื่อไม่ให้เกิดการแพร่กระจายของสนามแม่เหล็กไฟฟ้าเหล่านี้ และเนื่องจาก USB มีการแบ่งความเร็วของการถ่ายทอดข้อมูลออกเป็น 2 ระดับคือ แบบความเร็วเต็มที่ ที่มีอัตราเร็ว 12 Mbit/sec และแบบความเร็วต่ำ ที่มีอัตราเร็ว 1.5 Mbit/sec ทำให้คุณสมบัติของสายนำสัญญาณของอุปกรณ์แต่ละชนิดจะถูกกำหนดแยกแตกต่างกันออกไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

51858

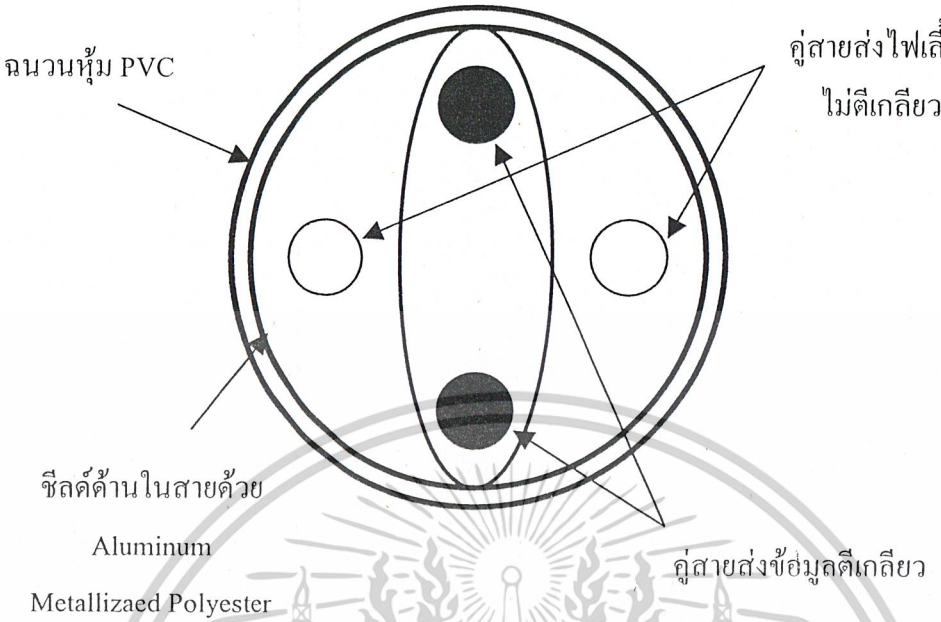


รูปที่ 2.8 โครงสร้างภายในของสาย USB แบบความเร็วต่ำ

3.1 สายนำสัญญาณความเร็วต่ำ

จากรูปที่ 2.8 แสดงให้เห็นภาพหน้าตัดของสายนำสัญญาณความเร็วต่ำ ชั้นนอกสุดของสายสัญญาณจะเป็นฉนวนหุ้มธรรมดา ภายในมีสายสัญญาณจะมีสายตัวนำ 4 เส้นโดย 2 เส้น ใช้ส่งไฟเลี้ยงกำหนดให้ใช้ลวดทองแดงเบอร์ 20-28 AWG ไม่ตีเกลียว ส่วนสายอีก 2 เส้นใช้ถ่ายทอดข้อมูลกำหนดให้ใช้ลวดทองแดงเบอร์ 28 AWG ไม่ตีเกลียว สายส่งข้อมูลความเร็วต่ำนี้ไม่จำเป็นต้องชีลด์ความยาวทั้งหมดของสายไม่เกิน 3 เมตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.9 โครงสร้างภายในของสาย USB แบบความเร็วสูง

3.2 สายนำสัญญาณความเร็วสูง

สายนำสัญญาณความเร็วสูงของ USB มีสาย 2 เส้นที่ใช้ถ่ายทอดข้อมูล กำหนดให้ใช้ลวดทองแดงเบอร์ 28 AWG เช่นเดียวกับสายนำสัญญาณความเร็วต่ำ แต่สาย 2 เส้นนี้จะต้องตีเกลียวและชั้นนอกสุดของสายสัญญาณ ถัดจากฉนวนจะต้องถูกหุ้มด้วยอลูมิเนียมฟอยล์ ดังรูปที่ 2.9 ความยาวของสายสัญญาณไม่เกิน 5 เมตร จะต้องมีค่าหน่วงเวลาสัญญาณ (Propagation delay) ไม่เกิน 30 นาโนวินาที (ns) ตลอดความยาวสาย 5 เมตร แต่ถ้าค่าหน่วงเวลามากกว่า 30ns ค่าความยาวสูงสุดของสายสัญญาณก็จะลดลงเป็นสัดส่วนดังตารางที่ 2.2

ตารางที่ 2.2 ค่าอัตราการหน่วงเวลาของสาย USB เมื่อเทียบกับความยาวของสายสัญญาณ

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.3 สัญญาณต่างๆ ในระบบของพอร์ต USB

1. การตรวจสอบการเชื่อมต่อและความเร็วของอุปกรณ์

คุณสมบัติเด่นข้อหนึ่งของ USB คือสามารถตรวจสอบการเชื่อมต่อของอุปกรณ์ตัวใหม่และตั้งค่าเริ่มต้นที่จำเป็นได้อัตโนมัติ แต่เนื่องจากอุปกรณ์ USB แบ่งออกเป็นอุปกรณ์ความเร็วสูงและต่ำ ถ้าส่งข้อมูลผิดความเร็วก็จะไม่สามารถสื่อสารกับอุปกรณ์นั้นๆ ได้ นั่นหมายความว่าจะต้องมีสัญญาณรูปแบบพิเศษที่ใช้แจ้งสถานะการเชื่อมต่อหรือปลดออกรวมถึงระบุความเร็วของอุปกรณ์นั้นๆ ได้ด้วย

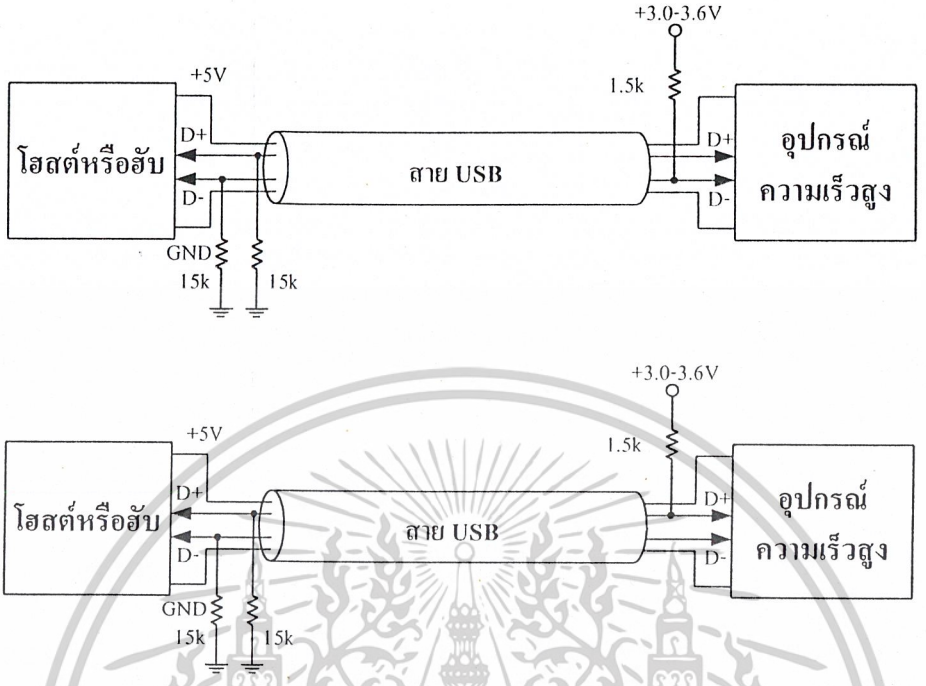
การตรวจสอบการเชื่อมต่อของ USB นั้นจะตรวจสอบจากกรเปลี่ยนแปลงระดับของสัญญาณในสายสัญญาณ D- และ D+ โดยสายข้อมูลทั้งสองที่ด้านฮับจะถูกต่อด้วยตัวต้านทาน 15 กิโลโอห์ม (KΩ) พูลดาวน์ไว้ทั้งสองเส้น (การพูลดาวน์ (Pull-down) : เป็นการต่อขาข้างหนึ่งของอุปกรณ์เข้ากับสายสัญญาณ อีกข้างหนึ่งต่อกราวด์ของระบบ) ซึ่งจะส่งผลให้สายสัญญาณทั้งสองมีระดับแรงดันเป็น 0V ในขณะที่ไม่มีอุปกรณ์ใดๆ ต่ออยู่เลย

ที่ด้านอุปกรณ์สำหรับอุปกรณ์ความเร็วต่ำสายสัญญาณ D- จะต่อตัวต้านทาน 1.5KΩ พูลอัปกับแรงดัน 3.0-3.6V (การพูลอัป (Pull-up) : เป็นการต่อขาข้างหนึ่งของอุปกรณ์เข้ากับสายสัญญาณอีกข้างหนึ่งต่อไฟเลี้ยง ทำให้สถานะที่จุดนั้นเกิดเป็นลอจิก “1”) ส่วนอุปกรณ์ความเร็วสูง สายสัญญาณที่ถูกพูลอัปคือ สายสัญญาณ D+ เมื่อมีอุปกรณ์ตัวใหม่ถูกต่อเข้ากับฮับจะเกิดการแบ่งแรงดันจากตัวต้านทานสองตัวก็คือตัวต้านทาน 1.5KΩ ที่พูลดาวน์อยู่ทางฮับ ทำให้แรงดันของสายสัญญาณเพิ่มจาก 0V ขึ้นไปที่ 90% ของไฟเลี้ยงที่พูลอัป คำนวณได้จากสูตรดังนี้

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

การเปลี่ยนแรงดันนี้จะทำให้ฮับรู้ว่ามีอุปกรณ์ตัวใหม่ต่อเข้ากับระบบ และจะทำให้รู้ว่าอุปกรณ์ถูกปลดออกจากระบบเช่นเดียวกัน โดยถ้าสัญญาณที่เกิดการเปลี่ยนแปลงแรงดันนี้เป็นสัญญาณ D- หมายความว่าอุปกรณ์ที่นำมาต่อเป็นอุปกรณ์ความเร็วต่ำ แต่ถ้าสัญญาณ D+ เปลี่ยนแปลงแรงดัน หมายความว่าอุปกรณ์ความเร็วสูงนำมาต่อเข้ากับระบบดังรูปที่ 2.10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.10 โครงสร้างภายในของสาย USB แบบความเร็วสูง

รูปที่ 2.11 แสดงให้เห็นถึงระดับแรงดันในสายสัญญาณเมื่อมีการเชื่อมต่ออุปกรณ์ใหม่ และเมื่อปลดอุปกรณ์ออกจากบัส ถ้าแรงดันของสัญญาณ D- หรือ D+ ตกลงต่ำกว่าค่า V_{SE}(min) หรือ 0.8V เป็นเวลานานกว่า 2.5 ไมโครวินาที (μs) ฮับจะถือว่าอุปกรณ์ได้ถูกปลดออกจากบัส ในทางกลับกันเมื่อแรงดันของสัญญาณ D- หรือ D+ เพิ่มขึ้นสูงกว่าค่า V_{SE}(max) หรือ 2.0V เป็นเวลานานกว่า 2.5μs จะถือว่ามีการต่ออุปกรณ์ตัวใหม่เข้ามาในระบบ ผลการตรวจสอบที่ได้ฮับจะเก็บไว้ในรีจิสเตอร์สถานะ ซึ่งโฮสต์จะคอยมาอ่านเป็นระยะๆ เพื่อการเชื่อมต่อของอุปกรณ์

2. เทคนิคการเข้ารหัสสัญญาณ

การเข้ารหัสสัญญาณข้อมูลก่อนจะส่งออกไปนั้นใช้เทคนิค 2 แบบประกอบกันคือ การเข้ารหัสแบบ NRZI (Non Return to Zero, Inverted) และการเติมบิตสตัฟ (Bit Stuffing)

2.1 การเข้ารหัส NRZI

ก่อนส่งข้อมูลไปในสาย ข้อมูลจะต้องถูกเข้ารหัสแบบ NRZI ประโยชน์ของการเข้ารหัสแบบนี้คือ สามารถแฝงสัญญาณนาฬิกาส่งไปกับข้อมูลได้ ทำให้อุปกรณ์ปลายทางสามารถซิงโครไนซ์สัญญาณนาฬิกาของตัวเองให้ตรงกับสัญญาณนาฬิกาของโฮสต์ได้ ทำให้สามารถอ่านข้อมูลที่ส่งมาได้อย่างถูกต้อง

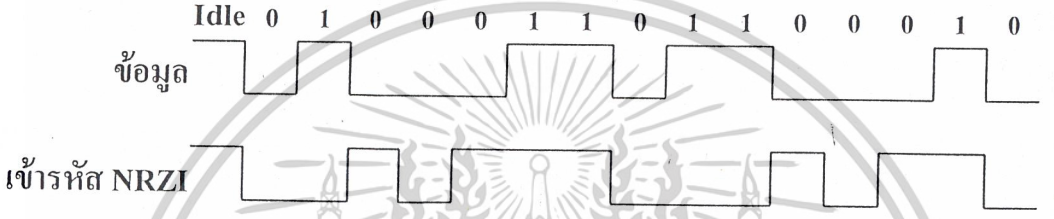
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.11 ระดับสัญญาณภายในสาย D+ และ D- เมื่อมีการเชื่อมต่อและปลดอุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในรูปที่ 2.12 แสดงให้เห็นการเข้ารหัสแบบ NRZI ของข้อมูล การเปลี่ยนแปลงของระดับสัญญาณจากระดับสูงไปต่ำหรือจากระดับต่ำไประดับสูงจะถือเป็นข้อมูล “0” แต่ถ้าไม่เกิดการเปลี่ยนแปลงของสัญญาณจะถือเป็นข้อมูล “1” แต่ด้วยการเข้ารหัสวิธีนี้ หากข้อมูลที่ส่งเป็น “1” เรียงกันหลายๆ บิต ก็จะไม่เกิดการเปลี่ยนแปลงระดับแรงดันในสายสัญญาณเลย อาจทำให้เกิดความผิดพลาดในการส่งข้อมูลได้ จึงต้องมีอีกเทคนิคหนึ่งในการส่งเข้ามาช่วยป้องกันคือ การเติมบิตสตัฟ



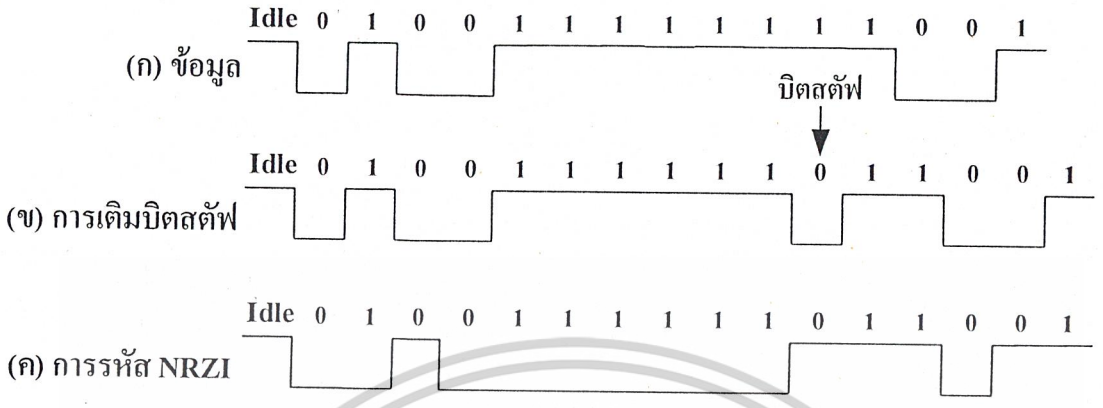
รูปที่ 2.12 การเข้ารหัส NRZI

2.2 การเติมบิตสตัฟ (Bit Stuffing)

การเติมบิตสตัฟจะเกิดขึ้นเมื่อข้อมูลที่ส่งมาเป็น “1” ต่อเนื่องกัน 6 บิต หลังจาก ข้อมูล “1” เรียงกันครบ 6ตัวจะเกิดการเติม “0” ลงในข้อมูลที่ส่งโดยอัตโนมัติ ซึ่งบิตข้อมูล “0” นี้เรียกว่า “บิตสตัฟ” ถึงแม้ว่าบิตที่ 7 ถัดไปจะเป็น “0” ก็ยังคงเติมบิตสตัฟนี้ด้วย การทำเช่นนี้ทำให้การส่งข้อมูลที่เข้ารหัสแบบ NRZI เกิดการเปลี่ยนแปลงอย่างน้อยทุกๆ 7 บิตข้อมูล เมื่อฝั่งรับได้รับข้อมูล “1” เรียงกัน 6 บิตก็จะตัดข้อมูล “0” ที่ตามมาทิ้งไปทำให้ข้อมูลที่ได้รับถูกต้อง

จากรูปที่ 2.13 เป็นตัวอย่างการเติมบิตสตัฟลงในข้อมูล โดยแถวบนสุดของรูปแสดงให้เห็นถึงข้อมูลที่จะส่งจริงๆ แต่เนื่องจากข้อมูลที่จะส่งเป็น “1” ติดกันถึง 8 ตัว ดังนั้นหลังจากส่งข้อมูล “1” ที่ติดกัน 6 บิตเสร็จแล้ว จะมีการเติมข้อมูล “0” ลงในข้อมูลหลัก ดังในรูปที่ 2.13(ข) เพื่อให้เกิดการเปลี่ยนแปลงระดับสัญญาณของการเข้ารหัสแบบ NRZI ดังในรูปที่ 2.13(ค) และหลังจากการเติมบิตสตัฟแล้ว ก็จะส่งข้อมูลหลักที่อยู่ต่อหลังไปเรื่อยๆ จนครบ ในขณะที่ฝั่งส่งมีการเติมบิตสตัฟ ทางฝั่งรับก็ต้องมีการตัดบิตสตัฟทิ้งเพื่อให้ได้รับข้อมูลที่ถูกต้องโดยการตัดก็จะใช้วิธีเดียวกันกับการแทรกคือเมื่อตรวจพบข้อมูล “1” เรียงต่อเนื่องกันครบ 6 ตัวจะตัดข้อมูล “0” ที่ตามหลังมาทิ้งโดยอัตโนมัติ แต่หากพบว่าหลังข้อมูล “1” ติดกัน 6 บิตไม่เป็น “0” ก็แสดงว่าเกิดความผิดพลาดในการส่งข้อมูลเกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



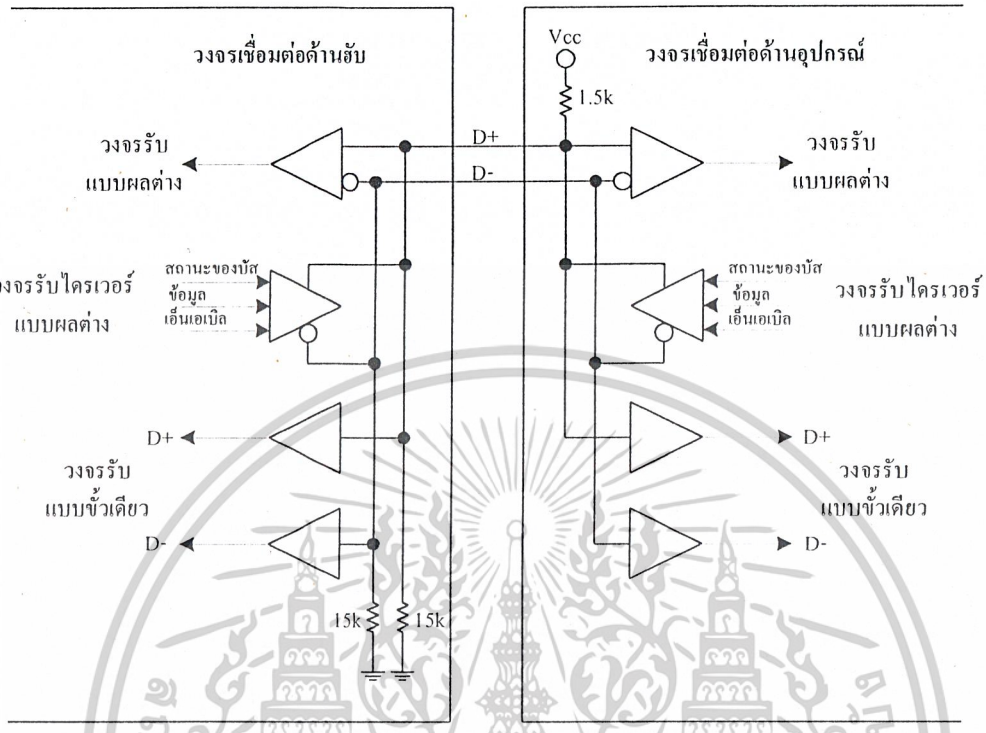
รูปที่ 2.13 การเติมบิตสตัฟ

3. การส่งข้อมูลในสายสัญญาณ

หลังจากข้อมูลถูกเข้ารหัสเรียบร้อยแล้ว ข้อมูลเหล่านั้นก็จะถูกส่งออกไปยังอุปกรณ์ต่างๆ ที่ต่ออยู่กับฮับ แต่เนื่องจาก USB ส่งข้อมูลด้วยความเร็วที่สูงมาก โดยใช้สายสัญญาณที่ยาว (เมื่อเทียบกับสัญญาณข้อมูลภายในคอมพิวเตอร์) ทำให้ไม่สามารถใช้การส่งสัญญาณแบบขั้วเดียว (Single end) ได้เพราะทำให้เกิดการแพร่กระจายของคลื่นแม่เหล็กไฟฟ้าไปกวนอุปกรณ์อื่น ๆ และยังทำให้สัญญาณรบกวนจากภายนอกเข้าไปกวนข้อมูลที่ต้องการส่งในสายสัญญาณได้ง่าย

การส่งสัญญาณที่แก้ไขปัญหา 2 ข้อที่กล่าวมาก็คือ การส่งสัญญาณผลต่าง เนื่องจากการส่งสัญญาณด้วยวิธีนี้จะทำให้สนามแม่เหล็กไฟฟ้าที่เกิดขึ้นจากสายสัญญาณทั้งสองเส้นหักล้างกัน เนื่องจากมีศักย์ตรงข้ามกัน จึงไม่เกิดการแพร่กระจายไปรบกวนอุปกรณ์อื่นๆ เมื่อมีสายสัญญาณรบกวนจากภายนอกเข้ามาในสาย ระดับของสัญญาณรบกวนจะเท่ากัน เพราะสายสัญญาณ 2 เส้นอยู่คู่กัน เมื่อสัญญาณรบกวนเหนี่ยวนำเข้าสู่สายเส้นหนึ่งก็จะเหนี่ยวนำเข้าสู่สายอีกเส้นในขนาดที่เท่ากันเมื่อถึงปลายทางในวงจรภาครับจะใช้วงจรขยายผลต่าง เป็นตัวรับสัญญาณ ซึ่งวงจรนี้จะมีคุณสมบัติขยายสัญญาณที่มีขนาดต่างกันและลดทอนสัญญาณที่มีขนาดเท่ากัน ดังนั้นสัญญาณข้อมูลซึ่งศักย์ตรงข้ามกันจะถูกขยายเพื่อส่งต่อไปยังส่วนถัดไป แต่สัญญาณรบกวนซึ่งมีขนาดเท่ากันในสายทั้ง 2 เส้นจะถูกลดทอนหรือตัดทิ้งไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.14 วงจรเชื่อมต่อระหว่างภาคส่ง (ฮับ) และภาครับ (ตัวอุปกรณ์)

จากรูปที่ 2.14 แสดงให้เห็นวงจรภาคส่งและภาครับของฮับและอุปกรณ์ จะเห็นได้ว่าทั้งสองด้านมีวงจรที่เหมือนกันเนื่องจาก USB เป็นบัสที่รับส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ นั่นคือทั้งสองด้านสามารถรับส่งข้อมูลได้ทั้งคู่ แต่คนละช่วงเวลากัน ทำให้สองด้านมีชุดวงจรที่เหมือนกัน และเนื่องจากการรับส่งข้อมูลแบบนี้จะมีอุปกรณ์เพียงตัวเดียวที่สามารถรับหรือส่งข้อมูลได้ในช่วงเวลาหนึ่งๆ อุปกรณ์ที่เหลือที่ไม่ได้ใช้งานบัสจำเป็นต้องปล่อยบัสสัญญาณให้มีสภาพอิมพิแดนซ์สูง ดังนั้นวงจรภาคส่งของอุปกรณ์แต่ละตัวจะต้องสามารถกำหนดสถานะของเอาต์พุตให้เป็นสภาพอิมพิแดนซ์สูงได้

ส่วนวงจรขับแบบผลต่าง ทำหน้าที่สร้างสัญญาณแบบผลต่างส่งไปในสายสัญญาณ D+ และ D- โดยสัญญาณในสายทั้ง 2 เส้นจะมีเฟสต่างกัน 180 องศา วงจรขับแบบผลต่างที่อยู่ในฮับจะต้องรองรับการถ่ายทอดข้อมูลได้ทั้งแบบความเร็วสูงและต่ำเพื่อให้สามารถถ่ายทอดข้อมูลไปยังอุปกรณ์ได้ทั้งสองชนิด แต่วงจรฝั่งตัวอุปกรณ์นั้นรองรับการส่งเพียงชนิดเดียวตามชนิดของอุปกรณ์นั้นๆ ก็เพียงพอ

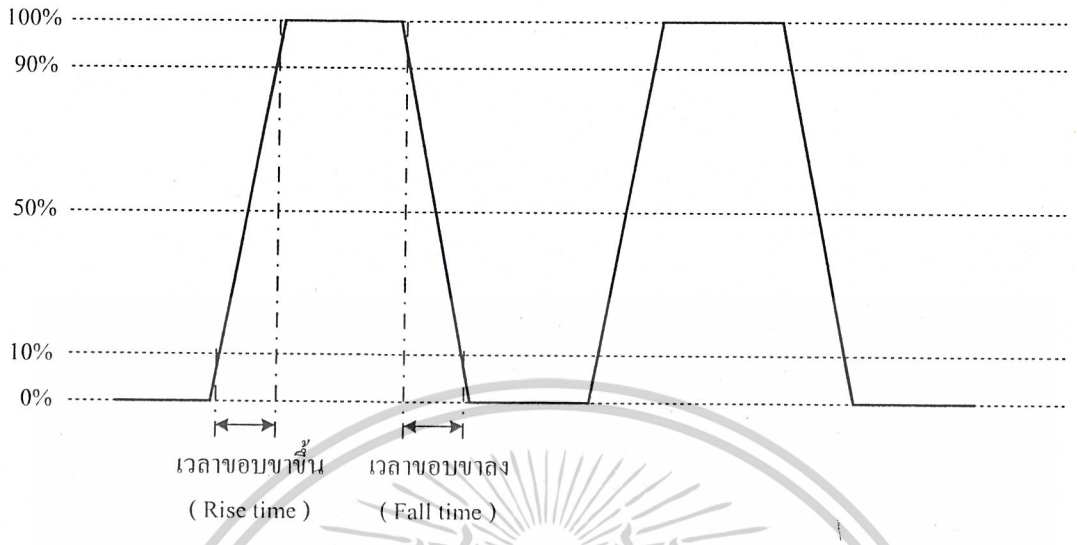
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คุณสมบัติที่สำคัญอีกประการหนึ่งที่จำเป็นในการออกแบบวงจรขับคือ ค่าเวลาขอบขาขึ้นและขอบขาลงของสัญญาณ หรือ Rise time และ Fall time ของสัญญาณข้อมูล โดยเวลาขอบขาขึ้นหรือ Rise time คือ เวลาที่ใช้ในการเปลี่ยนระดับแรงดันจาก 10% ไปถึง 90% ของแรงดันสูงสุด ในขณะที่เวลาขอบขาลงหรือ Fall time คือ เวลาที่ใช้ในการเปลี่ยนระดับแรงดันจาก 90% ไปถึง 10% ของแรงดันสูงสุด ดูรูปที่ 2.15 ประกอบ ค่าเวลาขอบขาขึ้นและลงของการถ่ายทอดข้อมูลความเร็วสูงจะต้องอยู่ในช่วง 4ns ถึง 20ns เพื่อไม่ให้เกิดผลกระทบต่อบิตข้อมูลส่วนการถ่ายทอดข้อมูลที่ความเร็วต่ำ ค่าเวลาขอบขาขึ้นและลงจะอยู่ในช่วง 75ns ถึง 300ns เพื่อให้เป็นไปตามมาตรฐาน FCC class B

ค่าข้อมูล "1" และ "0" ของการส่งแบบผลต่างนั้นจะดูจากความแตกต่างของระดับแรงดันในสายสัญญาณทั้ง 2 เส้น เช่น ถ้าแรงดันในสาย D+ มากกว่า D- จะถือเป็นข้อมูล "1" แต่ถ้าแรงดันในสาย D- มากกว่า D+ จะถือเป็นข้อมูล "0" จากตัวอย่างจะเห็นได้ว่า มีการตีความหมายของข้อมูลได้ 2 กรณี การถ่ายทอดข้อมูลแบบความเร็วสูงและต่ำจะตีความหมายสัญญาณทั้ง 2 แบบนี้ต่างกัน คือ ในโหมดความเร็วสูงข้อมูล "1" จะแทนด้วยระดับแรงดันในสาย D+ มากกว่า D- และข้อมูล "0" จะกลับตรงข้ามกัน ส่วนในโหมดความเร็วต่ำข้อมูล "1" จะแทนด้วยระดับแรงดันในสาย D- มากกว่า D+ และข้อมูล "0" จะกลับตรงข้ามเช่นเดียวกัน หรือกล่าวอย่างง่ายๆ ได้ว่า การถ่ายทอดข้อมูลในโหมดความเร็วสูงและความเร็วต่ำมีขั้วตรงข้ามกัน

ส่วนวงจรรับแบบผลต่าง ทำหน้าที่รับและขยายสัญญาณข้อมูลที่มี เฟสตรงกันข้าม และตัดสัญญาณที่รบกวนที่มีเฟสเหมือนกันทิ้งไป ส่วนวงจรรับแบบขั้วเดียว (Single-ended receiver) ทั้ง 2 ชุดนั้นใช้สำหรับตรวจสอบสถานะของบัส เพราะว่าสถานะต่างๆ ของบัสจะต้องพิจารณาจากระดับแรงดันในสายทั้ง 2 เส้นแยกจากกัน เช่นการเชื่อมต่อของอุปกรณ์ตัวใหม่จะเกิดการเปลี่ยนแปลงของสัญญาณเพียงเส้นเดียว ซึ่งวงจรรับแบบผลต่างไม่สามารถตรวจสอบการเปลี่ยนแปลงของสัญญาณในสายเพียงเส้นเดียวนี้ได้ เนื่องจากวงจรสามารถตรวจสอบได้ว่า เกิดการเปลี่ยนแปลงของสัญญาณ แต่ทราบว่าเกิดการเปลี่ยนแปลงที่สายเส้นใด ตารางที่ 2.3 แสดงให้เห็นสถานะต่าง ๆ ของบัส USB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.15 แสดงช่วงเวลาขอบขาขึ้น (Rise time) และขอบขาลง (Fall time) ของสัญญาณ

ตารางที่ 2.3 สถานะของบัส USB

สถานะของบัส	รายละเอียด
สถานะเตรียมพร้อม (Idle) กรณีความเร็วต่ำ (LS) กรณีความเร็วเต็มที่ (FS)	ระดับแรงดันที่ขา D- มากกว่า D+ เนื่องจากมีการต่อตัวต้านทานพูลอัปที่ขา D- ระดับแรงดันที่ขา D+ มากกว่า D- เนื่องจากมีการต่อตัวต้านทานพูลอัปที่ขา D+
สถานะติดต่อซ้ำ (Resume)	ข้อมูลในสายสัญญาณมีสถานะตรงข้ามกับสภาวะเตรียมพร้อม
เริ่มส่งข้อมูล (Start of Packet : SOP)	ข้อมูลในสายสัญญาณมีสถานะตรงข้ามกับสภาวะเตรียมพร้อม
สิ้นสุดข้อมูล (End of Packet : EOP)	แรงดันที่ขา D+ และ D- ต่ำกว่า 0.8V เป็นเวลา 2 บิตข้อมูล แล้วตามหลังด้วยสถานะเตรียมพร้อม
การเชื่อมต่ออุปกรณ์	แรงดันที่ขา D+ หรือ D- (ขึ้นอยู่กับชนิดของอุปกรณ์) สูงกว่า 2V นานกว่า 2.5μs
การปลดออกของอุปกรณ์	แรงดันที่ขา D+ หรือ D- (ขึ้นอยู่กับชนิดของอุปกรณ์) สูงกว่า 0.8V นานกว่า 2.5μs
รีเซต (Reset)	แรงดันที่ขา D+ หรือ D- ต่ำกว่า 0.8V เป็นเวลานาน 10 ms

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.4 การจัดการด้านพลังงานของ USB

เนื่องจากพอร์ต USB สามารถจ่ายพลังงานให้แก่อุปกรณ์ที่นำมาเชื่อมต่อได้ แต่ย่อมต้องมีข้อจำกัดว่า สามารถจ่ายพลังงานออกไปได้มากน้อยเพียงใด โดยเฉพาะกับพอร์ต USB ที่ยอมให้อุปกรณ์มาเชื่อมต่อได้มากถึง 127 ตัว โฮสต์จะต้องมีการจัดการด้านพลังงานที่ดีพอ มิฉะนั้นจะเกิดการดึงพลังงานจากบัส จนเครื่องคอมพิวเตอร์ไม่สามารถทำงานได้

1. การรายงานอัตราการใช้พลังงาน

โฮสต์จะรู้ข้อมูลเกี่ยวกับการใช้พลังงานของฮับและอุปกรณ์แต่ละตัวจากการอ่านคอนฟิกิวเรชั่นดิสคริปเตอร์ซึ่งจะถูกอ่านออกมาตั้งแต่ครั้งแรกที่มีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในบัส โดยข้อมูลที่เกี่ยวกับการใช้พลังงานจะมีอยู่ 2 ส่วน ส่วนแรกจะบอกโฮสต์ว่าตัวอุปกรณ์นี้ใช้พลังงานจากบัสหรือใช้พลังงานจากแหล่งพลังงานของตัวเองและส่วนที่สองบอกว่าจะขอใช้พลังงานจากบัสเป็นปริมาณเท่าใด มีหน่วยเป็นมิลลิแอมป์ (mA)

ในกรณีที่มีการเชื่อมต่อฮับเข้ากับบัสโฮสต์จะอ่านคอนฟิกิวเรชั่นดิสคริปเตอร์ไปเพื่อพิจารณาการใช้พลังงาน แต่เนื่องจากฮับจำเป็นจะต้องจ่ายพลังงานไปเลี้ยงอุปกรณ์ที่มาเชื่อมต่ออยู่กับตัวมันด้วย จึงต้องมีข้อกำหนดในด้านแรงดันของไฟเลี้ยงที่จะจ่ายออกจากตัวฮับเพื่อไม่ให้แรงดันปลายทางที่ตัวอุปกรณ์น้อยเกินไปจนไม่สามารถทำงานได้ โดยกำหนดไว้ว่า แรงดันที่จ่ายให้แก่ฮับเพื่อให้ฮับทำงานนั้นจะต้องไม่ต่ำกว่า 4.75V และแรงดันที่ฮับจ่ายออกไปยังตัวอุปกรณ์แต่ละตัวจะต้องไม่ต่ำกว่า 4.40V (ดูรูปที่ 2.16 ประกอบ) นั่นหมายความว่าอุปกรณ์ USB ทุกๆ ตัวจะต้องสามารถทำงานได้ที่แรงดันไฟเลี้ยงอย่างน้อย 4.40V

2. การจำกัดกระแส

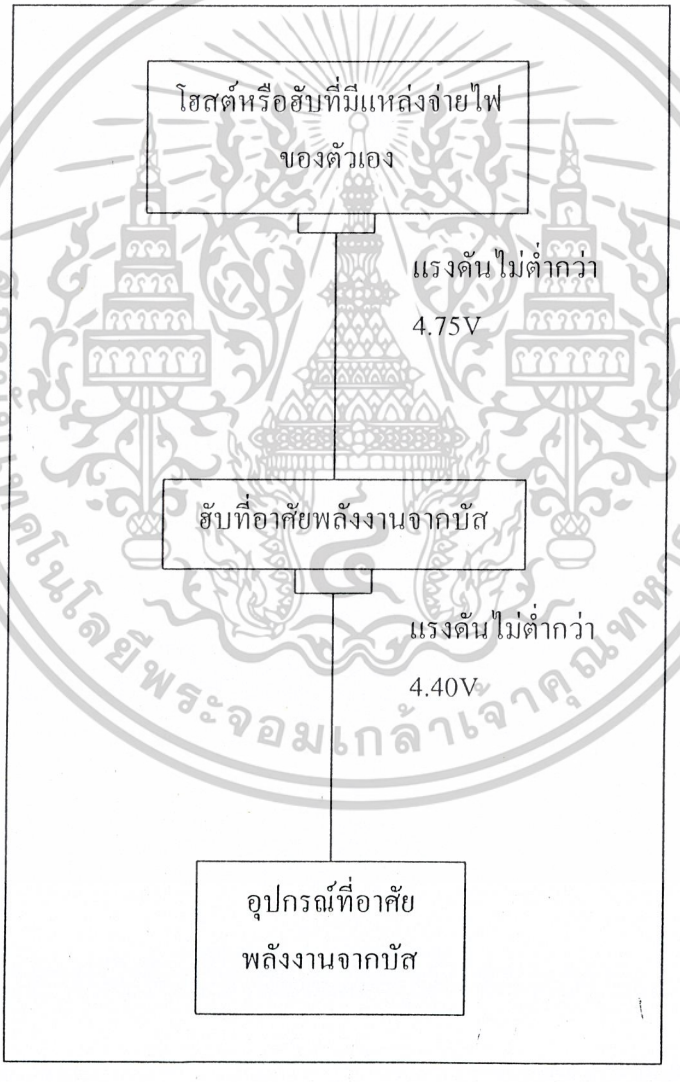
ที่กล่าวมาเป็นข้อกำหนดในเรื่องแรงดันของฮับ ข้อกำหนดอีกส่วนหนึ่งที่ต้องกล่าวถึงคือข้อกำหนดเรื่องกระแส ค่ากระแสต่ำสุดที่ฮับจะต้องจ่ายให้แต่ละพอร์ตจะต้องไม่น้อยกว่า 100mA และจ่ายได้มากที่สุด 500mA นั่นหมายความว่า อุปกรณ์ที่มาเชื่อมต่อกับบัส USB โดยใช้พลังงานจากบัสนั้นจะไม่สามารถดึงกระแสได้มากกว่า 500mA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

สำหรับฮับที่ใช้ไฟเลี้ยงจากบัส ตัวฮับเองจะใช้พลังงานจากบัสเพื่อควบคุมการทำงานภายในของตัวฮับเอง 1 หน่วยเท่ากับ 100mA และจากข้อกำหนดด้านกระแสที่ว่า บัสจะจ่ายกระแสได้ไม่เกิน 500mA นั่นหมายความว่า ฮับจะเหลือพลังงานที่สามารถจ่ายให้แก่พอร์ตต่างๆ ได้อีก 4 หน่วย ซึ่งเท่ากับว่า จะมีพอร์ตได้มากที่สุด 4 พอร์ต โดยแต่ละพอร์ตต้องดึงกระแสไม่เกิน 100mA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

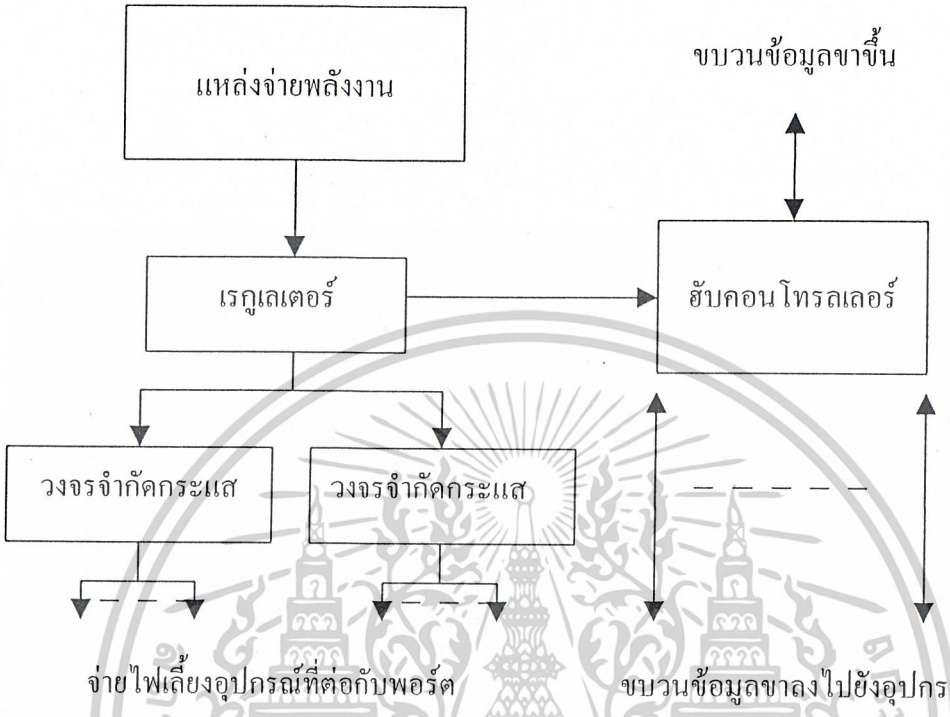
แต่ในกรณีของฮับที่มีแหล่งจ่ายไฟของตัวเองจะไม่พบกับปัญหานี้เพราะสามารถจ่ายกระแสให้แต่ละพอร์ตได้เต็ม 500mA

นอกจากข้อกำหนดในเรื่องการดึงกระแสของแต่ละพอร์ตแล้ว ในแต่ละพอร์ตจะต้องมีวงจรจำกัดกระแส (Current limit) ไม่ให้เกิน 5 แอมป์ (Amp : A) สำหรับพอร์ตแต่ละพอร์ตเพื่อป้องกันการลัดวงจรระหว่างไฟเลี้ยงและกราวด์ แต่ในกรณีของฮับที่อาศัยไฟเลี้ยงจากบัสไม่จำเป็นต้องมีวงจรป้องกันส่วนนี้เพราะรูตฮับหรือฮับตัวก่อนหน้า (ที่มีแหล่งจ่ายไฟของตัวเอง) จะมีวงจรป้องกันอยู่แล้ว



รูปที่ 2.16 ข้อจำกัดด้านแรงดันที่จุดเชื่อมต่อต่าง ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.17 การจัดการพลังงานของฮับที่มีแหล่งจ่ายไฟเลี้ยง

3. การเปิด-ปิดไฟเลี้ยงของฮับ

นอกจากฮับจะแบ่งออกเป็น 2 ชนิดตามการใช้พลังงานคือ ฮับที่ใช้พลังงานจากบัส และฮับที่ใช้พลังงานของตัวเองแล้ว ฮับยังแบ่งชนิดย่อยตามการเปิดการจ่ายพลังงานให้แก่อุปกรณ์ที่พอร์ตต่าง ๆ ด้วย โดยแบ่งออกเป็น 3 ชนิดคือ

1. จ่ายพลังงานโดยตรง ในแบบนี้จะจ่ายพลังงานไปยังพอร์ตต่าง ๆ ไว้ตลอดเวลา

2. เปิดปิดการจ่ายพลังงานแบบรวม ในแบบนี้ฮับสามารถสั่งปิดการจ่ายพลังงานแก่พอร์ตต่าง ๆ ได้ แต่จะเป็นการกระทำกับพอร์ตทุก ๆ พอร์ตพร้อมกัน นั่นคือ ถ้าเปิดก็เปิดทั้งหมดถ้าปิดก็ปิดทั้งหมด

3. เปิดปิดการจ่ายพลังงานแบบแยกพอร์ต ในแบบนี้ฮับสามารถสั่งปิดเปิดการจ่ายพลังงานแต่ละพอร์ตแยกกันอย่างอิสระ

4. การใช้พลังงานในช่วงตั้งค่าเริ่มต้นทำงานหรือคอนฟิก

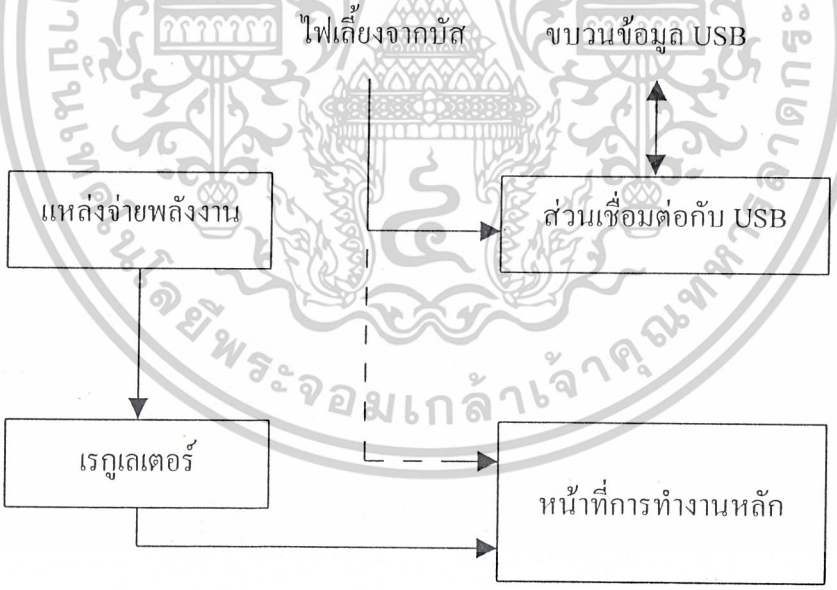
อุปกรณ์ USB เมื่อเชื่อมต่อเข้ามาในบัสต้องได้รับการกำหนดลักษณะการทำงานจากโฮสต์ก่อนเริ่มทำงานทุกครั้ง เพื่อให้โฮสต์ทราบถึงข้อมูลที่จำเป็นในการติดต่อกับอุปกรณ์นั้น ด้วยการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อ่านข้อมูลดิสคริปเตอร์ต่างๆ เพื่อตั้งค่าเริ่มต้นให้แก่ระบบและตัวอุปกรณ์ ในช่วงการตั้งค่าเริ่มต้นนี้จะมีข้อกำหนดว่า อุปกรณ์สามารถดึงพลังงานไปใช้ได้ไม่เกิน 1 หน่วยหรือ 100mA แต่หลังจากโฮสต์พิจารณาทรัพยากรที่ระบบมี ว่าเพียงพอต่อความต้องการของอุปกรณ์และตั้งค่าต่างๆ ให้จนเสร็จเรียบร้อยแล้ว อุปกรณ์หรือฮับจะสามารถดึงกระแสได้เต็มความต้องการที่แจ้งไว้ในดิสคริปเตอร์

5. การใช้พลังงานของอุปกรณ์ USB

อุปกรณ์ USB สามารถแบ่งชนิดจากลักษณะการใช้พลังงานได้ 2 กลุ่มคือ อุปกรณ์ที่ใช้พลังงานจากบัส (ใช้พลังงานต่ำ) เช่น เมาส์ คีย์บอร์ด และอุปกรณ์ที่ใช้พลังงานของตัวเอง (ใช้พลังงานสูง) เช่น เครื่องพิมพ์หรือซีดีรอมไดรฟ์ อุปกรณ์ที่ใช้พลังงานจากบัสสามารถดึงกระแสได้ตั้งแต่ 100-500mA ดังที่ได้กล่าวไปแล้วแต่ก็ต้องขึ้นอยู่กับความสามารถในการจ่ายพลังงานของฮับด้วยว่าฮับนั้นๆ มีอุปกรณ์ต่ออยู่แล้วกี่ตัว แต่ละตัวดึงพลังงานไปแล้วเท่าใด เหลือพลังงานให้เพียงพอกับที่อุปกรณ์ตัวใหม่ต้องการหรือไม่ หากโฮสต์พิจารณาแล้วว่าไม่สามารถจ่ายพลังงานให้อุปกรณ์ตัวใหม่นี้ได้เพียงพอก็จะไม่ตั้งค่าเริ่มต้นให้อุปกรณ์นั้น ๆ ทำให้อุปกรณ์ตัวนั้นไม่สามารถทำงานได้



รูปที่ 2.18 การใช้พลังงานของอุปกรณ์ไฮบริด

สำหรับอุปกรณ์ที่มีใช้พลังงานของตัวเองจะไม่เกิดเหตุการณ์ดังกล่าวขึ้น เพราะไม่ต้องใช้พลังงานจากบัสเลย แต่มีอุปกรณ์อีกประเภทหนึ่งที่มีรูปแบบการใช้พลังงานที่ต่างกันออกไป โดยจะใช้พลังงานจากบัสสำหรับวงจรส่วนติดต่อกับ USB แต่ใช้พลังงานของตัวเองสำหรับการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลักของตัวอุปกรณ์ อุปกรณ์ที่ใช้พลังงานลักษณะนี้เรียกว่า อุปกรณ์ไฮบริด (Hybrid) ข้อดีของอุปกรณ์นี้คือ โฮสต์สามารถรู้สถานะของตัวอุปกรณ์ได้ แม้อุปกรณ์ตัวนั้นจะยังไม่ได้จ่ายไฟเลี้ยง นอกจากนั้นยังทำให้สามารถออกแบบอุปกรณ์ที่มีฟังก์ชั่นการทำงาน 2 รูปแบบที่ต่างกันในจังหวะที่มีและไม่มีไฟเลี้ยงได้

6. การประหยัดพลังงานของพอร์ต USB

นอกจากข้อจำกัดด้านการใช้พลังงานในสภาวะปกติ USB ยังมีข้อกำหนดสำหรับการประหยัดพลังงานของอุปกรณ์ในบัสด้วย โดยการเข้าสู่โหมดประหยัดพลังงานเรียกว่า Suspend และการกลับมาทำงานในสภาวะปกติเรียกว่า Resume และเนื่องจาก USB เป็นระบบบัสทำให้การเข้าและออกจากโหมดประหยัดพลังงานจำเป็นต้องมีลำดับขั้นตอนในการส่งสัญญาณต่างๆ ที่สัมพันธ์กันตั้งแต่จุดเริ่มต้นคือ รูตฮับผ่านฮับไปยังตัวอุปกรณ์ต่าง ๆ

การเข้าสู่โหมดประหยัดพลังงานมีด้วยกัน 2 ชนิดคือ การ Suspend อุปกรณ์ทั้งบัส และการ Suspend อุปกรณ์เฉพาะตัว

7. กระบวนการเข้าสู่โหมดประหยัดพลังงานหรือ Suspend

USB เป็นบัสที่ควบคุมการทำงานจากโฮสต์ซึ่งก็คือเครื่องคอมพิวเตอร์เพียงจุดเดียวโดยข้อมูลทั้งหมดจะถูกส่งในลักษณะของเฟรมข้อมูล ในแต่ละเฟรมจะเริ่มต้นการส่งข้อมูลด้วย SOF (Start of Frame) แพ็กเก็ตแล้วจึงตามด้วยข้อมูลที่ต้องการส่งหรือรับ จังหวะการส่งข้อมูลกำหนดไว้ว่า ต้องเริ่มส่ง SOF แพ็กเก็ตเพื่อเริ่มต้นเฟรมทุกๆ 1ms นั่นหมายความว่า ทุกๆ 1ms อุปกรณ์ทุกๆ ตัวในบัสจะต้องได้รับข้อมูล SOF แพ็กเก็ตเพื่อเป็นสัญญาณบอกการเริ่มเฟรมข้อมูล ถ้าอุปกรณ์ไม่ได้รับข้อมูลใดๆ เลยจากบัสเป็นเวลามากกว่า 3ms อุปกรณ์นั้นๆ จะเข้าสู่โหมดประหยัดพลังงานหรือ Suspend โดยในช่วงที่ Suspend นี้อุปกรณ์จะดึงกระแสจากบัสได้ไม่เกินพอร์ตละ 500ไมโครแอมป์ (μA)

กระบวนการข้างต้นเป็นการทำงานของอุปกรณ์ความเร็วสูงเท่านั้น สำหรับอุปกรณ์ความเร็วต่ำจะแตกต่างออกไป เพราะอุปกรณ์ความเร็วต่ำจะไม่ได้รับข้อมูลความเร็วสูง และ SOF แพ็กเก็ตที่ถูกส่งออกมาทุกๆ 1ms เพื่อไม่ให้อุปกรณ์ Suspend ก็เป็นข้อมูลความเร็วสูง ในขณะที่อุปกรณ์ความเร็วต่ำเมื่อไม่ได้รับข้อมูล บัสจะอยู่ในสถานะสงบหรือไอเดิล (Idle) ดังนั้นจึงเป็นหน้าที่ของฮับที่จะต้องทำให้พอร์ตที่เชื่อมกับอุปกรณ์ความเร็วต่ำ มีการเปลี่ยนกลับขั้วสัญญาณจากสถานะไอเดิล เป็นตรงกันข้ามภายในทุกๆ 3ms เพื่อป้องกันการเข้าสู่โหมดประหยัดพลังงานของอุปกรณ์

กระบวนการ Suspend ของอุปกรณ์ทุกๆ ตัวในบัส เริ่มจากโฮสต์สั่งให้รูตฮับเข้าสู่ภาวะประหยัดพลังงาน เมื่อรูตฮับได้รับคำสั่งหยุดการส่งข้อมูลในบัส เมื่อรูตฮับหยุดส่งข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หมายความว่า ทั้งฮับและอุปกรณ์ทั้งหมดที่อยู่ภายในบัสก็จะไม่ได้รับข้อมูลไปด้วย เมื่อเวลาผ่านไป 3ms อุปกณ์ทุกตัวและฮับก็จะเข้าสู่โหมดประหยัดพลังงานพร้อมกัน แต่ถ้าโฮสต์ต้องการ Suspend อุปกรณ์เฉพาะตัว โฮสต์จะตรวจสอบว่าอุปกรณ์ตัวนั้นต่ออยู่กับพอร์ตใด หลังจากนั้นก็จะเข้าไปเขียนรีจิสเตอร์ของฮับพอร์ตนั้นๆ เพื่อสั่ง Suspend เมื่อฮับได้รับคำสั่งจากโฮสต์ก็จะปิดการส่งข้อมูลของพอร์ตนั้นทำให้อุปกรณ์ที่ต่อกับพอร์ตนั้นเข้าสู่โหมดประหยัดพลังงาน

ขณะที่อยู่ในโหมดประหยัดพลังงาน ถึงแม้ว่าอุปกรณ์จะดึงกระแสไม่เกิน 500μA แต่ตัวฮับจะต้องรองรับการใช้พลังงานระดับปกติของอุปกรณ์แต่ละตัวด้วย เพื่อให้ตัวอุปกรณ์สามารถออกจากโหมดประหยัดพลังงานเพื่อกลับเข้าสู่การทำงานปกติได้เมื่อต้องการส่งข้อมูลกลับ

8. กระบวนการออกจากโหมดประหยัดพลังงานหรือ Resume

การ Resume คือการกลับเข้าสู่สภาวะการใช้พลังงานปกติและมีการถ่ายทอดข้อมูลตามปกติ สัญญาณที่ใช้ Resume คือ การกลับขั้วของสัญญาณ D- และ D+ ให้ตรงข้ามกับสถานะไอเดิล กระบวนการ Resume สามารถเกิดขึ้นได้จากหลายเงื่อนไขดังนี้

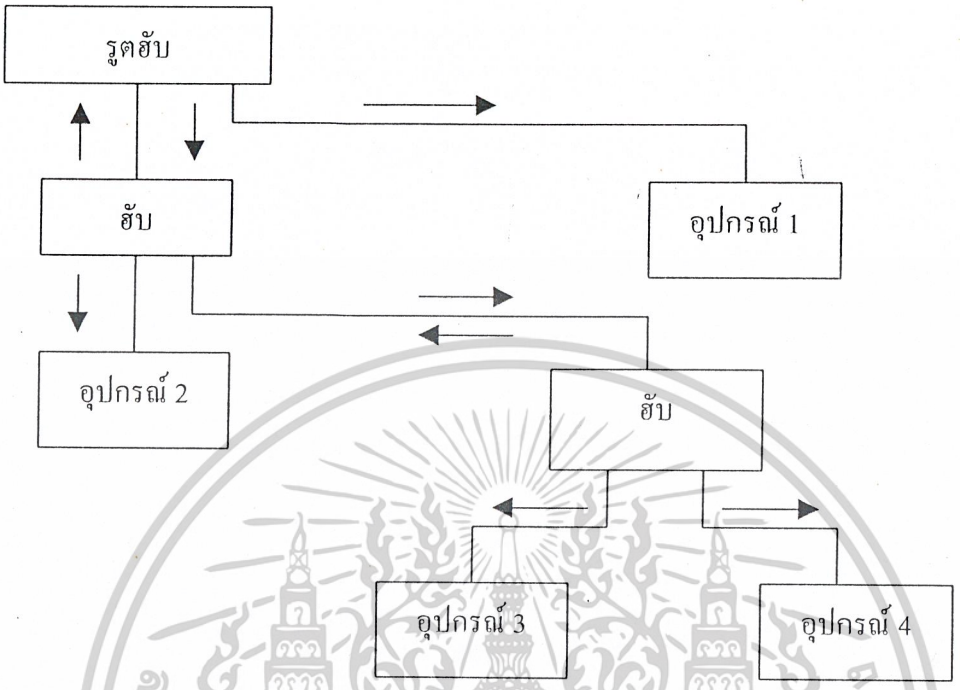
1. Resume อุปกรณ์ทั้งหมดโดยรูตฮับ
2. Resume อุปกรณ์ทั้งหมดจากตัวอุปกรณ์
3. Resume จากการเชื่อมต่อของอุปกรณ์ตัวใหม่
4. Resume จากการปลดอุปกรณ์ออกจากบัส
5. Resume จากการรีเซต
6. Resume อุปกรณ์เฉพาะตัวจากรูตฮับ

การ Resume อุปกรณ์ทั้งหมดจากรูตฮับจะเกิดจากรูตฮับส่งสถานะ Resume ออกมาในบัสเป็นเวลามากกว่า 20ms ซึ่งสถานะ Resume ก็จะถูกฮับกระจายออกไปยังอุปกรณ์ทุกๆ ตัว ทำให้อุปกรณ์ทุกตัวกลับเข้าสู่การทำงานตามปกติ

การ Resume จากอุปกรณ์มีกระบวนการที่แตกต่างและขั้นตอนที่มากขึ้น เริ่มจากอุปกรณ์ที่ต้องการ Resume ขับสถานะ Resume เข้าไปยังบัสเพื่อส่งขึ้นไปยังฮับที่ต่ออยู่ เมื่อฮับได้รับสถานะ Resume ก็จะส่งสถานะนี้ขึ้นไปด้านบน นอกจากนั้นยังส่งสถานะนี้ลงไปยังพอร์ตทุกพอร์ต (รวมพอร์ตของอุปกรณ์ที่สร้าง Resume ขึ้นมาด้วย) ภายในเวลา 100μs ถ้าด้านบนฮับมีฮับต่ออยู่อีกชั้นหนึ่งก็จะเกิดกระบวนการลักษณะเดียวกัน จนกระทั่งสถานะ Resume ถูกส่งขึ้นไปถึงรูตฮับเมื่อรูตฮับได้รับสถานะ Resume แล้วจะสร้างสถานะ Resume กลับลงมาอีกเป็นเวลามากกว่า 20ms

การจบสถานะ Resume ทำได้โดยสร้างสถานะ EOP เป็นเวลา 2 บิตของการส่งข้อมูลความเร็วต่ำ หรือ 1.33μs (การส่งข้อมูลความเร็วต่ำ 1.5 Mbit/sec เวลา 1 บิตเท่ากับ 6.67μs ดังนั้นเวลา 2 บิตจึงเท่ากับ 1.33μs)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.19 การ Resume จากอุปกรณ์ USB

2.2.5 โครงสร้างพื้นฐานการส่งข้อมูลของพอร์ต USB

การต่ออุปกรณ์ร่วมกันบนบัสเส้นเดียวในจำนวนมากๆ นั้นจำเป็นต้องมีการจัดการที่ดีพอ ทั้งในด้านขนาดของข้อมูลที่จะส่งระดับความสำคัญของแต่ละอุปกรณ์ และการจัดจังหวะการรับส่งข้อมูลเพราะอุปกรณ์ที่ต่อร่วมกันในบัสนั้นจะรับหรือส่งข้อมูลทั้งหมดร่วมกันภายในสายสัญญาณที่ต่ออยู่ในบทนี้จะกล่าวถึงการจัดข้อมูลจากอุปกรณ์ต่างๆ เข้าเป็นเฟรมข้อมูล อธิบายถึงการส่งข้อมูลชนิดต่างๆ และงานที่เหมาะสมกับการส่งข้อมูลแต่ละแบบ

1. ภาพรวมของการส่งสัญญาณ

ดังที่กล่าวไปแล้วว่าลักษณะการเชื่อมต่อของ USB อยู่ในรูปแบบของบัส คุณสมบัติด้านการส่งข้อมูลของการเชื่อมต่อแบบนี้คือ อุปกรณ์ทุกๆ ตัวจะต้องส่งข้อมูลร่วมกันไปในสายสัญญาณเพียงเส้นเดียวทำให้อุปกรณ์ทุกๆ ตัวได้รับข้อมูลเหมือนกันทั้งหมด แต่จะเลือกรับเฉพาะข้อมูลที่เป็นของตัวเองเท่านั้น ซึ่งจะทำให้เกิดปัญหาว่าจะทำอย่างไรให้อุปกรณ์แต่ละตัวรู้ว่าข้อมูลชุดใดเป็นของตนจะรับส่งข้อมูลได้เมื่อใด แล้วจะตรวจสอบอย่างไรว่าข้อมูลที่ได้นั้นถูกต้อง จึงต้องกำหนดรูปแบบการรับส่งข้อมูลแต่ละชุดให้มีความน่าเชื่อถือมากขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การส่งข้อมูลของอุปกรณ์หลายๆ ชนิดรวมกันไปในสายสัญญาณเส้นเดียวนั้นจะต้องมีการแบ่งช่วงข้อมูลแต่ละส่วนออกจากกันอย่างชัดเจน เพื่อไม่ให้เกิดความสับสนในการสื่อสาร ข้อมูลที่ถูกแบ่งออกเป็นส่วนๆ นี้เรียกว่า แพ็กเก็ต (Packet) โดยแต่ละแพ็กเก็ตนอกจากจะประกอบไปด้วยข้อมูลที่ต้องการจะรับหรือส่งจริงๆ แล้ว ยังต้องประกอบด้วยข้อมูลส่วนอื่นๆ รวมไปด้วยเพื่อควบคุมการส่งข้อมูลให้เกิดความถูกต้อง เช่น แอดเดรสของอุปกรณ์ หมายเลขเฟรมข้อมูล และส่วนตรวจสอบความผิดพลาด โดยข้อมูลเหล่านี้จะถูกนำไปรวมกับข้อมูลที่จะส่งจริงๆ ด้านหน้าและด้านหลัง ระบบแพ็กเก็ตจึงเป็นโครงสร้างพื้นฐานการถ่ายทอดข้อมูลของ USB

เนื่องจากข้อมูลที่ส่งไปภายในสาย USB นั้นมีอยู่หลายชนิด ทั้งข้อมูลจริงๆ ข้อมูลควบคุม ข้อมูลกำหนดจังหวะของเฟรม นอกจานนั้นทิศทางของข้อมูลยังมีทั้ง 2 ทิศทาง ดังนั้นแพ็กเก็ตข้อมูลจึงถูกแบ่งออกเป็นชนิดต่างๆ ตามจุดประสงค์ในการส่งและข้อมูลที่บรรจุอยู่ภายใน โดยแพ็กเก็ตข้อมูลถูกแบ่งออกเป็น 4 ชนิด คือ โทเค็นแพ็กเก็ต (Token packets) ดาต้าแพ็กเก็ต (Data packets) แฮนด์เช็กแพ็กเก็ต (Handshake packets) และ แพ็กเก็ตพิเศษ (Special packets)

เมื่อนำแพ็กเก็ตข้อมูลต่างชนิดกันมารวมกันเพื่อสร้างการติดต่อสื่อสารขึ้น การติดต่อที่สร้างขึ้นนี้เรียกว่า ทรานแซกชั่น (Transaction) ซึ่งก็จะถูกแบ่งตามลักษณะการติดต่อสื่อสารเป็น 3 ชนิด ตามทิศทางของการส่งข้อมูลคือ ทรานแซกชั่นขาเข้า (In tramsaction) ทรานแซกชั่นขาออก (Out tramsaction) และ เซตอัปทรานแซกชั่น (Setup tramsaction) แต่ละทรานแซกชั่นส่วนใหญ่จะประกอบด้วย 3 ส่วนข้อมูลหรือเรียกว่า “เฟส” ได้แก่ เฟสโทเค็น (ใช้โทเค็นแพ็กเก็ต) เฟสข้อมูล (ใช้ดาต้าแพ็กเก็ต) และเฟสแฮนด์เช็ก (ใช้แฮนด์เช็กแพ็กเก็ต) เมื่อสร้างทรานแซกชั่นขึ้นมาจากแพ็กเก็ตต่าง ๆ แล้วจะถูกนำไปจัดเรียงในเฟรมข้อมูลที่มีการส่งในทุก 1ms

การจัดรูปแบบของทรานแซกชั่นต่างๆ ของแต่ละเฟรมข้อมูลจะถูกกำหนดเป็นชนิดของการถ่ายทอดข้อมูลหรือเรียกทับศัพท์ว่า ทรานสเฟอร์ (Transfer) ซึ่งแบ่งออกเป็นชนิดต่างๆ ตามรูปแบบในการถ่ายทอดข้อมูลในแต่ละเฟรม (แต่ละ 1ms) ตามความต้องการในการใช้งานของตัวอุปกรณ์เป็น 4 ชนิดคือ การถ่ายทอดสัญญาณแบบไอโซโครนัส (Isochronous transfer) การถ่ายทอดสัญญาณแบบบัลก์ (Bulk transfer) การถ่ายทอดสัญญาณแบบอินเตอร์รัพต์ (Interrupt transfer) และการถ่ายทอดสัญญาณควบคุม (Control transfer)

2. ชนิดของการถ่ายทอดสัญญาณ (Transfer type)

เนื่องจาก USB ออกแบบไว้รองรับการทำงานของอุปกรณ์หลายๆ ประเภท ซึ่งแต่ละประเภทจะมีความต้องการในการรับหรือส่งข้อมูลที่แตกต่างกันออกไปตามชนิด หน้าที่ของอุปกรณ์นั้นๆ ดังนั้นการรับส่งสัญญาณจึงต้องออกแบบไว้ให้ครอบคลุมการทำงานทุกๆ ประเภทของอุปกรณ์ทุกๆ ชนิดทำให้ USB กำหนดชนิดของการรับส่งข้อมูลออกเป็น 4 ชนิดคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. การถ่ายทอดสัญญาณแบบไอโซโครนัส(Isochronous transfer)

2. การถ่ายทอดสัญญาณแบบอินเตอร์รัพต์ (Interrupt transfer)

3. การถ่ายทอดสัญญาณแบบบัลก์ (Bulk transfer)

4. การถ่ายทอดสัญญาณควบคุม (Control transfer)

เอ็นด์พอยต์เป็นชุดรีจิสเตอร์ที่ใช้ติดต่ออุปกรณ์โดยแต่ละเอ็นด์พอยต์ก็จะมีหน้าที่ในการทำงานที่ต่างกันไป นั่นหมายความว่า แต่ละเอ็นด์พอยต์ก็จะใช้การรับส่งข้อมูล 1 ใน 4 ชนิดที่กล่าวไปนี้ในการติดต่อกับโฮสต์ หรือกล่าวอย่างง่ายๆ ก็คือแต่ละเอ็นด์พอยต์จะใช้ชนิดการรับส่งข้อมูลเป็นของตัวเองเพียงชนิดเดียวแยกจากเอ็นด์พอยต์อื่นๆ อย่างอิสระ เช่น ในอุปกรณ์ตัวหนึ่งจะมีเอ็นด์พอยต์ควบคุมที่ใช้การถ่ายทอดสัญญาณควบคุมเพื่อรับส่งข้อมูลควบคุมการทำงานของอุปกรณ์ ในขณะที่อีกเอ็นด์พอยต์หนึ่งใช้การถ่ายทอดแบบไอโซโครนัสเพื่อส่งข้อมูลจากการทำงานหน้าที่หนึ่ง และใช้อีกเอ็นด์พอยต์หนึ่งที่ใช้การถ่ายทอดแบบบัลก์เพื่อรับส่งตัวข้อมูลจากการทำงานอีกหน้าที่หนึ่ง เป็นต้น ส่วนถัดไปจะกล่าวถึงรายละเอียดของการรับส่งข้อมูลทั้ง 4 ชนิดในด้านความเหมาะสมกับงานต่าง ๆ และจังหวะการรับส่งข้อมูล

2.1 การถ่ายทอดสัญญาณแบบไอโซโครนัส

การรับส่งข้อมูลชนิดนี้จะมีการส่งหรือรับข้อมูลในทุก ๆ เฟรมข้อมูล (ทุกๆ 1ms) ทำให้เกิดการรับส่งข้อมูลด้วยอัตราเร็วที่คงที่ แต่นั่นก็หมายความว่า การรับส่งแบบนี้จะต้องจองการรับส่งข้อมูลในแต่ละเฟรม 1ms ไว้ตลอดเวลาด้วยประเภทของงานที่ใช้การรับส่งข้อมูลชนิดนี้คือ งานที่ต้องการอัตราการส่งข้อมูลที่คงที่ตลอดเวลา เช่น การส่งข้อมูลเสียงเพลงไปยังลำโพง USB หรือการรับข้อมูลจากไมโครโฟน USB เพราะว่า ลักษณะงานประเภทนี้ทางปลายทางจะนำข้อมูลที่ได้รับแปลงกลับเป็นข้อมูลแอนะลอก (Analog) ซึ่งจะเกิดความผิดเพี้ยนทางความถี่ได้ หากข้อมูลที่ส่งเกิดการขาดช่วงไม่สม่ำเสมอ แต่จากข้อกำหนดที่ว่าต้องส่งหรือรับข้อมูลต่อเนื่องในทุก ๆ เฟรมข้อมูล นั่นหมายความว่า หากเกิดความผิดพลาดของข้อมูลจะไม่มีการพยามยามส่งข้อมูลที่ผิดนั้นกลับไปใหม่ (เพราะหากมีการส่งใหม่จะทำให้อัตราการส่งข้อมูลเปลี่ยนไป) จึงทำให้การถ่ายทอดข้อมูลแบบนี้จะไม่มีการรับประกันความถูกต้องของข้อมูล

ก่อนเริ่มการทำงาน อุปกรณ์จะต้องการซิงโครไนซ์การทำงานของตัวเองเข้ากับสัญญาณเริ่มต้นแต่ละเฟรมของ USB เพื่อให้อุปกรณ์สามารถรับหรือส่งข้อมูลในทุก ๆ 1ms ได้ทัน ทำให้เกิดการรับและส่งข้อมูลในจังหวะที่ถูกต้อง การซิงโครไนซ์เข้ากับเฟรมข้อมูลนั้นมีอยู่ 3 วิธีดังนี้

1. อะซิงโครนัส (Asynchronous) การซิงโครไนซ์วิธีนี้สัญญาณนาฬิกาของอุปกรณ์ไม่จำเป็นต้องเข้าจังหวะกับสัญญาณข้อมูลของ USB สัญญาณนาฬิกาที่ควบคุมการทำงานของอุปกรณ์จะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นอิสระไม่ขึ้นอยู่กับสัญญาณ SOF แต่อัตราการส่งข้อมูลจะต้องถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่ง ซึ่งกำหนดตอนเริ่มต้นการเชื่อมต่อ ไม่สามารถเปลี่ยนแปลงความเร็วได้

ตัวอย่างของอุปกรณ์ที่ซิงโครไนซ์ด้วยวิธีนี้คือ ซีดีรอมในโหมดการเล่นเพลง เพราะว่า ซีดีเพลงนั้นจะมีอัตราการสุ่มข้อมูลคงที่เท่ากับ 44.1 กิโลเฮิรตซ์ (kHz) ซึ่งอัตราการส่งข้อมูลนี้จะเป็นอิสระจากการส่งสัญญาณ SOF ของ USB นอกจากซีดีรอมแล้วอีกตัวอย่างหนึ่งที่เห็นได้ชัดก็คือ ไมโครโฟน USB ที่มีอัตราการสุ่มข้อมูล (Sampling rate) เป็นอิสระไม่ขึ้นอยู่กับสัญญาณ SOF ของ USB

2. ซิงโครนัส (Synchronous) ในวิธีสัญญาณนาฬิกาที่ใช้อุปกรณ์กับสัญญาณ SOF ของ USB จะต้องเข้าจังหวะกัน โดยก่อนเริ่มการทำงาน อุปกรณ์จะต้องมีการปรับสัญญาณนาฬิกาของตัวเองให้เข้ากับจังหวะ SOF แต่ถ้าหากอุปกรณ์ตัวนั้นไม่สามารถปรับสัญญาณนาฬิกาของตัวเองได้ USBจะยอมให้จังหวะการส่ง SOF ของตัวอุปกรณ์นั้นเป็นหลัก แล้วปรับจังหวะการส่ง SOF ของตัวโฮสต์เองให้เข้ากับการส่งข้อมูลของอุปกรณ์นั้นได้ โดยอุปกรณ์ตัวนั้นจะเรียกว่า อุปกรณ์มาสเตอร์ (Master device) แต่ USB อนุญาตให้มีอุปกรณ์มาสเตอร์ได้เพียงตัวเดียวเท่านั้นในระบบ ความเร็วในการถ่ายทอดข้อมูลวิธีนี้จะต้องกำหนดไว้ตายตัวตั้งแต่เริ่มต้นการเชื่อมต่อเช่นเดียวกับแบบอะซิงโครนัส

ตัวอย่างของอุปกรณ์ที่ทำงานลักษณะนี้คือ ไมโครโฟน USB ที่มีอัตราการสุ่มข้อมูลเข้าจังหวะกับสัญญาณ SOF นั่นเอง อีกตัวอย่างหนึ่งก็คือ โมเด็ม (MODEM) ISDN ความเร็ว 64 Kbit/sec จะต้องเกิดการซิงโครไนซ์สัญญาณ SOF ของโฮสต์เข้ากับสัญญาณนาฬิกาของระบบ ISFN ก่อนจึงจะเกิดการถ่ายทอดข้อมูลที่ความเร็วคงที่ 64 Kbit/sec ได้

3. อะแดปตีฟ (Adaptive) วิธีนี้เป็นวิธีที่ตัวอุปกรณ์มีความสามารถมากที่สุด เพราะอุปกรณ์สามารถปรับความเร็วในการถ่ายทอดข้อมูลได้ในช่วงที่กว้างมาก ไม่ถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่งสามารถเปลี่ยนแปลงความเร็วได้ในระหว่างการใช้งาน สัญญาณนาฬิกาภายในระบบของอุปกรณ์จะเข้าจังหวะกับสัญญาณข้อมูลที่ส่งในสาย (แตกต่างจากวิธีซิงโครนัสที่เข้าจังหวะกับสัญญาณ SOF)

ตัวอย่างของอุปกรณ์ที่ทำงานลักษณะนี้คือ ซีดีรอมเล่นเพลงที่มีอุปกรณ์พิเศษในการเปลี่ยนอัตราการสุ่มข้อมูล (Sample rate converts : SRC) ซึ่งด้วยอุปกรณ์ตัวนี้จะทำให้ความเร็วในการสุ่มข้อมูลไม่จำเป็นต้องคงที่เท่ากับ 44.1 kHz แต่จะสามารถปรับเปลี่ยนไปตามขอบเขตการทำงานของตัว SRC

จังหวะการถ่ายทอดข้อมูลแบบไอโซโครนัสจะถูกกำหนดให้ส่งข้อมูลในทุกๆ 1ms เพื่อให้เกิดอัตราการส่งข้อมูลที่คงที่ขึ้น นั่นหมายความว่า จะต้องจองแบนด์วิดธ์ของสายไว้คงที่ตลอดเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขนาดข้อมูลใหญ่ที่สุดที่สามารถส่งได้ในแต่ละเฟรมเท่ากับ 1,023 ไบต์ โดยอุปกรณ์จะแจ้งขนาดของการรับส่งข้อมูลที่ต้องการใช้ในแต่ละเฟรมแก่โฮสต์ทางเอ็นด์พอยต์ดิสคริปเตอร์ ซึ่งโฮสต์จะนำไปใช้พิจารณาความสามารถของระบบว่า รองรับการทำงานของอุปกรณ์ได้หรือไม่ หากพิจารณาแล้วว่าแบนด์วิทธ์ข้อมูลของระบบไม่เพียงพอต่อความต้องการของตัวอุปกรณ์ ก็จะไม่ตั้งค่าที่จำเป็นในการทำงานให้แก่ตัวอุปกรณ์ และปิดการทำงานของพอร์ตนั้นๆ ทำให้อุปกรณ์ตัวนั้นไม่สามารถทำงานได้

เนื่องจากในการถ่ายทอดข้อมูลแบบนี้ต้องการความเร็วที่คงที่ตลอดเวลา เมื่อเกิดความผิดพลาดของข้อมูลจึงไม่สามารถส่งใหม่ได้ เพราะจะทำให้เกิดการซิงโครไนซ์ของข้อมูลคลาดเคลื่อนไปส่งผลให้การรับข้อมูลเกิดความผิดพลาด ดังนั้นการถ่ายทอดข้อมูลแบบไอโซโครนัสจึงไม่มีการตรวจสอบความผิดพลาดของการส่งข้อมูลและไม่มีการส่งข้อมูลซ้ำ ทำให้วิธีการนี้ไม่เหมาะสมกับงานที่ต้องการความถูกต้องของข้อมูลเป็นสำคัญ

2.2 การถ่ายทอดสัญญาณแบบอินเตอร์รัพต์

การรับส่งข้อมูลชนิดนี้สร้างขึ้นเพื่อเลียนแบบการสร้างสัญญาณอินเตอร์รัพต์ของอุปกรณ์ให้แก่ระบบ วิธีการเลียนแบบอาศัยการวนอ่านข้อมูลจากตัวอุปกรณ์ต่างๆ ในระยะเวลาที่กำหนดอย่างสม่ำเสมอหรือเรียกว่า โพลลิ่ง (Polling) โดยอัตราการวนอ่านข้อมูลนี้จะต้องไม่ช้าเกินไปเพราะจะทำให้เกิดการสูญเสียข้อมูลที่อ่านไม่ทันได้ ในขณะเดียวกันก็จะต้องไม่เร็วเกินไป จนเกินเข้าไปในแบนด์วิดธ์ข้อมูลมากเกินจำเป็น ถ้าหากอุปกรณ์ไม่มีข้อมูลที่ต้องการส่ง ก็จะส่งสัญญาณ No Acknowledge (NAK) ตอบกลับ

คาบเวลาในการวนอ่านข้อมูลของอุปกรณ์แต่ละตัวจะขึ้นอยู่กับความต้องการของอุปกรณ์แต่ละชนิด คาบเวลานี้สามารถเปลี่ยนได้ในช่วงกว้างตั้งแต่วนอ่านทุกๆ 1 เฟรม (1ms) ไปจนถึงทุกๆ 255 เฟรม (255ms) แต่ถ้าเป็นอุปกรณ์ความเร็วต่ำคาบเวลาที่สั้นที่สุด (ความถี่สูงสุด) ที่สามารถวนอ่านได้คือ ทุกๆ 10 เฟรม (10ms) โฮสต์จะรู้คาบเวลาในการวนอ่านของเอ็นด์พอยต์ที่ใช้วิธีการถ่ายทอดสัญญาณแบบอินเตอร์รัพต์จากการอ่านเอ็นด์พอยต์ดิสคริปเตอร์

การส่งข้อมูลแต่ละครั้งสามารถกำหนดให้ส่งได้มากที่สุด 64 ไบต์ (ถ้าหากเป็นอุปกรณ์ความเร็วต่ำจะส่งได้สูงสุด 8 ไบต์) หากข้อมูลที่ต้องการส่งทั้งไม่มีมากกว่าก็จะต้องแบ่งเป็นส่วน ๆ โดยแต่ละส่วนจะต้องส่งให้เต็มจำนวนข้อมูลที่สามารถส่งได้สูงสุด (เพื่อให้โฮสต์รับทราบว่ายังส่งข้อมูลไม่ครบ) แล้วในการส่งครั้งสุดท้ายซึ่งจะส่งไม่เต็มจำนวนสูงสุดจะเป็นตัวบอกให้โฮสต์ทราบว่าจบการส่งข้อมูลแล้ว

อย่างไรก็ตามการถ่ายทอดสัญญาณแบบอินเตอร์รัพต์นี้ก็ต้องการอัตราการอ่านข้อมูลที่คงที่คล้ายกับการแบบไอโซโครนัส เพราะถ้าหากโฮสต์ไม่สามารถวนอ่านข้อมูลได้ตามที่แจ้งไว้ใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอ็นด์พอยต์ดิสคริปเตอร์ อาจทำให้เกิดการล้นของข้อมูลที่ต้อการจะส่งจากอุปกรณ์กลับมายังโฮสต์ได้หรือเกิดโอเวอร์โฟลว เนื่องจากโฮสต์ไม่อ่านข้อมูลออกไปจนกระทั่งข้อมูลที่เตรียมจะส่งล้นบัฟเฟอร์ ดังนั้นก่อนเริ่มการทำงานของอุปกรณ์โฮสต์จะต้องอ่านเอ็นด์พอยต์ดิสคริปเตอร์ออกมาแล้วพิจารณาว่าระบบสามารถรองรับการทำงานได้หรือไม่ ถ้าได้ก็จะตั้งค่าที่จำเป็นในการทำงานแล้วเริ่มการทำงาน แต่ถ้าไม่สามารถรองรับได้ก็จะไม่ตั้งค่าให้และปิดการทำงานของพอร์ตนั้น

การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์นี้ถึงแม้จะต้องการอัตราการวนอ่านข้อมูลที่คงที่แต่ก็ไม่จำเป็นต้องซิงโครไนซ์อัตราการส่งข้อมูลเข้ากับระบบ ดังนั้นเมื่อเกิดความผิดพลาดของการส่งข้อมูลขึ้นก็สามารถส่งใหม่ได้ ทำให้การถ่ายทอดสัญญาณแบบนี้จึงต้องมีการตรวจสอบความถูกต้องของข้อมูลและมีการส่งข้อมูลใหม่ในรอบการวนอ่านถัดไปหากเกิดความผิดพลาดในการส่ง ตัวอย่างของอุปกรณ์ที่ใช้การถ่ายทอดสัญญาณแบบนี้คือ คีย์บอร์ดและเมาส์ เป็นต้น

2.3 การถ่ายทอดสัญญาณแบบบัลก์

เป็นการรับส่งข้อมูลผ่านพอร์ต USB สำหรับงานที่ไม่ต้องการอัตราเร็วในการถ่ายทอดสัญญาณที่คงที่ อุปกรณ์ที่เห็นได้ชัดเจนที่สุดสำหรับการถ่ายทอดสัญญาณในลักษณะนี้คือ เครื่องพิมพ์ เพราะข้อมูลที่ผู้ใช้งานต้องการส่งไปพิมพ์นั้นไม่ต้องการความเร็วในการส่งที่สม่ำเสมอ การส่งช้าหรือเร็วก็จะส่งผลกระทบในด้านเวลาที่ต้องใช้มากขึ้นเท่านั้น แต่ข้อมูลที่ได้ก็ยังไม่เสียหาย เนื่องจากการถ่ายทอดสัญญาณแบบนี้ไม่สามารถจองการใช้งานแบนด์วิดธ์ของบัสได้ ดังนั้นการรับส่งข้อมูลจะเกิดขึ้นก็ต่อเมื่อบัสว่างจากการส่งข้อมูลอื่นๆ ทั้งหมดแล้วเท่านั้น หรือกล่าวอย่างง่ายๆ ว่าการถ่ายทอดสัญญาณแบบบัลก์จะเกิดขึ้นได้ก็ต่อเมื่อเหลือที่ว่างของข้อมูลในแต่ละเฟรมเท่านั้น หากไม่มีที่ว่างเหลือก็จะต้องรอไปเรื่อยๆ จนกว่าจะสามารถส่งได้ แต่ในทางกลับกันถ้าในบัสไม่มีอุปกรณ์ตัวใดใช้งานอยู่เลยก็จะสามารถส่งข้อมูลได้เต็มความสามารถที่บัสจะส่งได้ทันทีทำให้ความเร็วในการส่งข้อมูลสูงมาก อาจสูงถึง 1 เมกะไบต์ต่อวินาที (MByte/sec) เลยทีเดียว (ขนาดข้อมูล 1 ไบต์ = 8 บิต เพราะฉะนั้น 12 Mbit/sec จึงเท่ากับ 1.5 MByte/sec)

ถึงแม้ว่าบัสจะว่างมากเพียงใด แต่การถ่ายทอดสัญญาณแบบบัลก์ก็ไม่สามารถส่งข้อมูลขนาดใหญ่ไปในคราวเดียวได้ เพราะเสี่ยงต่อการเกิดความผิดพลาดของข้อมูลระหว่างส่ง ข้อมูลจะต้องถูกแบ่งออกเป็นส่วนๆ แล้วทยอยส่งต่อกันไปในเฟรมเดียวกัน โดยขนาดของชิ้นข้อมูลที่ใหญ่ที่สุดหลังจากแบ่งแล้วจะเท่ากับ 8, 16, 32 หรือ 64 ไบต์ การส่งข้อมูลที่ถูกแบ่งจะต้องส่งในขนาดใหญ่ที่สุดที่ตกลงกันไว้กับโฮสต์เพื่อให้โฮสต์ทราบว่า ยังไม่จบการส่งข้อมูล และโฮสต์จะทราบว่าจบการส่ง เมื่อพบว่าขนาดของข้อมูลที่ส่งไม่เต็มขนาดใหญ่สุดที่กำหนดไว้ (เหมือนกับแบบอินเตอร์รัพต์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เนื่องจากอุปกรณ์ที่ใช้การถ่ายทอดสัญญาณแบบนี้ส่วนใหญ่ต้องการความถูกต้องของข้อมูลสูง เช่น ซีดีรอม (ในส่วนของการอ่านข้อมูลที่ไม่ใช่เพลง) เครื่องพิมพ์ หรือสแกนเนอร์ ดังนั้นการตรวจสอบข้อมูลจึงเป็นเรื่องจำเป็น ในการส่งข้อมูลแต่ละครั้งจึงมีการตรวจสอบความถูกต้องในทุกๆ ครั้ง และจะเกิดการส่งใหม่ถ้ามีความผิดพลาดเกิดขึ้น

2.2.6 ความผิดพลาดและการแก้ไขในการสื่อสารข้อมูลผ่านพอร์ต USB

จากข้างต้นได้กล่าวถึงการถ่ายทอดสัญญาณข้อมูลชนิดต่างๆ ไปครบทุกชนิดแล้ว ซึ่งถ้านำข้อกำหนดดังกล่าวมาใช้ทั้งหมดโดยไม่เกิดความผิดพลาดขึ้นเลยก็คงจะไม่มีปัญหา แต่ในความเป็นจริง ข้อมูลที่เกิดการรับหรือการส่งภายในบัสนั้นมีโอกาสที่จะเกิดความผิดพลาดขึ้นได้ มาตรฐาน USB จึงกำหนดความผิดพลาดชนิดต่างๆ ขึ้นรวมไปถึงการตรวจสอบหาความผิดพลาดนั้นๆ และการแก้ไข ในระหว่างการส่งข้อมูลจากโฮสต์ไปยังอุปกรณ์หรือจากอุปกรณ์กลับมายังโฮสต์อาจเกิดความผิดพลาดขึ้น ณ เวลาใดก็ได้ ดังนั้นจึงต้องมีการกำหนดวิธีค้นหาความผิดพลาดดังกล่าวให้พบเพื่อไม่ให้เกิดการนำข้อมูลที่ผิดไปใช้งาน ความผิดพลาดที่ถูกกำหนดขึ้นตามมาตรฐาน USB มีดังนี้

1. ความผิดพลาดภายในแพ็กเก็ต
2. สถานะ EOP ผิดพลาด
3. บัสไทม์เอาต์ (ไม่ได้รับการตอบสนอง)
4. ส่งข้อมูลซ้ำเนื่องจากความผิดพลาดในการแพ็กเก็ตแฮนด์เชก
5. สถานะบัสค้าง (Babble)
6. การสูญเสียรูปแบบการส่งข้อมูล (Loss of Activity : LOA)

1. ความผิดพลาดภายในแพ็กเก็ต

การตรวจสอบความผิดพลาดภายในแพ็กเก็ตข้อมูลสามารถทำได้ 3 จุดนี้

1.1 PID

ในข้อมูลทุกๆ แพ็กเก็ตจะเริ่มต้นด้วยส่วน PID หรือ Packet ID เสมอ โดยจะเป็นข้อมูล PID ขนาด 4 บิตแล้วตามด้วยส่วนตรวจสอบอีก 4 บิตซึ่งเป็นข้อมูลคอมพลีเมนต์ของ 4 บิตแรก (คอมพลีเมนต์ (Complement) หมายถึง การกลับสถานะข้อมูลให้เป็นตรงข้าม) ถ้าพบว่าทั้งสองส่วนไม่เป็นคอมพลีเมนต์ของกันและกัน นั่นหมายความว่าข้อมูลที่ได้รับนี้ผิดพลาดแน่นอน และเมื่อเกิดความผิดพลาดของ PID อุปกรณ์เป้าหมายจะไม่สนใจข้อมูลในแพ็กเก็ตนั้นทั้งหมดเพราะไม่สามารถทราบได้ว่าเป็นทรานแซกชั่นชนิดใด

1.2 CRC

ดังที่กล่าวไปแล้วว่าในแต่ละแพ็กเก็ตจะมีส่วนตรวจสอบความผิดพลาดแบบ CRC ตามปิดท้ายเสมอ โดยจำนวนบิตของ CRC ที่ใช้มีขนาด 5 และ 16 บิตขึ้นอยู่กับชนิดของแพ็กเก็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อนำค่านี้ไปคำนวณตามกระบวนการ CRC แล้วพบว่าไม่ถูกต้อง หมายความว่า ข้อมูลแพ็กเก็ตนั้นเกิดความผิดพลาด

1.3 บิตสตัฟ

ในระหว่างการส่งข้อมูลถ้ามีข้อมูล "1" เรียงต่อเนื่องกัน 6 บิตจะมีการแทรกข้อมูล "0" ลงไป 1 บิต ซึ่งหากพบว่ามีข้อมูล "1" เรียงกัน 6 บิตแล้วถัดมาไม่เป็น "0" แสดงว่าเกิดความผิดพลาดของข้อมูล

การตรวจสอบทั้ง 3 จุดข้างต้นจะรายงานผลออกมาเหมือนกันคือ เกิดความผิดพลาดภายในแพ็กเก็ต และเมื่อความผิดพลาดชนิดนี้ อุปกรณ์หรือโฮสต์ที่เป็นผู้รับจะไม่สนใจข้อมูลแพ็กเก็ตนั้นและไม่ตอบแฮนด์เช็กกลับจนกระทั่งทางฝั่งส่งตรวจพบบัสไทม์เอาต์ทำให้อีกทางรู้ว่าเกิดความผิดพลาดขึ้นและจะเกิดการส่งหรือรับข้อมูลที่เสียไปนั้นใหม่ในรอบถัดไป

2. สถานะ EOP (End of frame) ผิดพลาด

ความผิดพลาดนี้เกิดขึ้นจากการตรวจสอบสถานะ EOP ในขณะที่ยังส่งข้อมูลไม่เสร็จสิ้น ซึ่งอาจทำให้เกิดการชนกันของข้อมูลถ้าอุปกรณ์ปลายทางพบสถานะ EOP ซึ่งถือว่าเป็นตัวจบแพ็กเก็ตหลังจากนั้นอุปกรณ์ก็จะพยายามส่งข้อมูลหรือแฮนด์เช็กกลับทำให้เกิดการชนกันของข้อมูล

เมื่อตรวจพบความผิดพลาดของสถานะ EOP กระบวนการตอบสนองจะเหมือนกับการผิดพลาดภายในแพ็กเก็ตคือ อุปกรณ์ที่พบความผิดพลาดจะไม่ตอบแฮนด์เช็กหรือข้อมูลกลับจนกระทั่งฝ่ายส่งข้อมูลตรวจพบบัสไทม์เอาต์จึงจะเริ่มส่งข้อมูลใหม่

3. บัสไทม์เอาต์

ทรานแซกชั่นต่างๆ ของ USB ล้วนมีการตอบกลับไปมาระหว่างโฮสต์และอุปกรณ์เสมอ เว้นแต่ทรานแซกชั่นขาออกของการถ่ายทอดสัญญาณแบบไอโซโครนัส ระยะเวลาที่รอการตอบกลับจากอีกฝ่ายนานที่สุดจะคำนวณจากค่าหน่วงเวลาของสัญญาณภายในสายส่งและภายในฮับ โดยค่าที่คำนวณโดยประมาณจะเท่ากับเวลา 18 บิตของการส่งข้อมูล ซึ่งตัวเลขนี้สามารถใช้ได้ทั้งการส่งข้อมูลความเร็วสูงและความเร็วต่ำ

4. ส่งข้อมูลซ้ำเนื่องจากความผิดพลาดในการแพ็กเก็ตแฮนด์เช็ก

สาเหตุที่ต้องมีการกำหนดแพ็กเก็ตข้อมูลออกเป็น 2 ชนิดก็เพื่อแก้ปัญหาการรับส่งข้อมูลไม่ถูกต้องในบางรูปแบบ ซึ่งเป็นช่องวางจากการตรวจสอบทั้งหมดที่ได้พูดถึง ดังตัวอย่างต่อไปนี้

ในกรณีของทรานแซกชั่นขาเข้า เริ่มต้นจากโฮสต์ส่งแพ็กเก็ตโทเค็น IN ไปยังอุปกรณ์ ถ้าหากการส่งข้อมูลเสร็จสมบูรณ์ ตัวอุปกรณ์ได้รับข้อมูลถูกต้อง อุปกรณ์จะส่งแพ็กเก็ตข้อมูลที่โฮสต์ต้องการกลับมา ถ้าหากโฮสต์ได้รับอย่างถูกต้อง ก็จะส่งแพ็กเก็ตแฮนด์เช็กกลับไปยังตัวอุปกรณ์ ใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

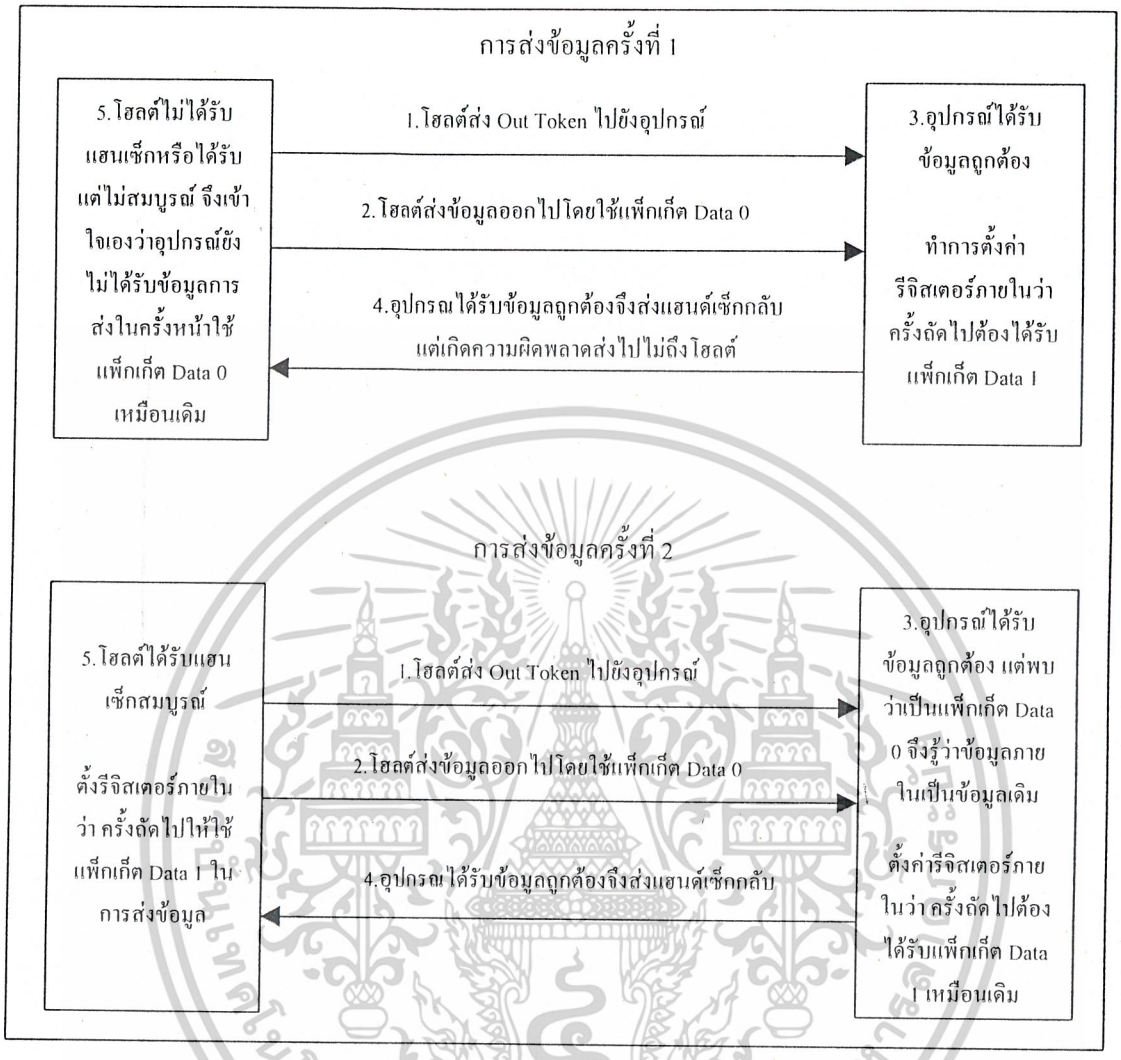
ขณะที่กำลังส่งแพ็กเก็ตแฮนด์เช็กเกิดความผิดพลาดขึ้นกับตัวแพ็กเก็ตแฮนเช็ก ตัวอุปกรณ์เมื่อได้รับข้อมูลที่ผิดก็จะไม่สนใจแพ็กเก็ตนั้นๆ ผลที่ตามมาคือ อุปกรณ์จะเข้าใจว่าโฮสต์ยังไม่ได้รับข้อมูลที่ส่งไป เมื่อถึงทรานแซกชั่นขาเข้าครั้งถัดไป อุปกรณ์ก็จะส่งข้อมูลชุดเดิมที่คิดว่าโฮสต์ยังไม่ได้รับไปอีกครั้ง ในขณะที่โฮสต์เข้าใจว่า ข้อมูลนี้คือข้อมูลชุดใหม่ ทำให้เกิดเป็นการรับข้อมูลซ้ำ จากปัญหานี้จึงมีการคิดหาวิธีแก้ไข ซึ่งแพ็กเก็ตข้อมูล “0” และ “1” ก็คือเครื่องมือที่เข้ามาช่วยในงานนี้

วิธีที่ใช้แก้ปัญหานี้เรียกว่า การกลับสถานะชุดส่งข้อมูล วิธีนี้เป็นการจัดลำดับ การใช้งานแพ็กเก็ตข้อมูล “0” และ “1” เพื่อให้โฮสต์หรือตัวอุปกรณ์ทราบถึงความผิดพลาดของการส่งข้อมูล โดยกระบวนการส่งข้อมูลจะเริ่มต้นที่การส่งข้อมูลในทรานแซกชั่นแรกซึ่งจะใช้แพ็กเก็ตข้อมูล “0” ในการส่งหรือรับข้อมูล และเมื่อมีการส่งข้อมูลครั้งต่อไปก็จะสลับไปใช้แพ็กเก็ตข้อมูล “1” และสลับกันไปเรื่อยๆ

จากตัวอย่างปัญหา เริ่มจากโฮสต์ส่งแพ็กเก็ตโทเค็นไปยังอุปกรณ์เรียบร้อยถูกต้อง อุปกรณ์ก็จะส่งแพ็ตเก็ตกลับมายังโฮสต์ และเนื่องจากเป็นการส่งแพ็กเก็ตข้อมูลครั้งแรกจึงต้องใช้แพ็กเก็ตข้อมูล “0” เมื่อโฮสต์ได้รับข้อมูลถูกต้องก็จะส่งแพ็กเก็ตแฮนด์เช็กไปยังตัวอุปกรณ์แล้วเกิดความผิดพลาดขึ้นกับแพ็กเก็ตข้อมูลนี้ทำให้ตัวอุปกรณ์คิดว่าโฮสต์ไม่ได้รับข้อมูล

เมื่อถึงทรานแซกชั่นขาเข้าครั้งถัดไป ตัวอุปกรณ์ก็จะส่งข้อมูลชุดเดิมที่คิดว่าโฮสต์ยังไม่ได้รับกลับมาอีกครั้งโดยใช้แพ็กเก็ตข้อมูล “0” เหมือนเดิม เพราะถือว่าการส่งครั้งที่แล้วไม่สมบูรณ์จึงไม่สลับเป็นแพ็กเก็ต “1” ซึ่งเมื่อโฮสต์ได้รับแพ็กเก็ตข้อมูล “0” ซ้ำอีกครั้งก็จะได้ว่านี่เป็นข้อมูลชุดเดิมไม่ต้องเอาไปใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.



รูปที่ 2.20 ตัวอย่างการใช้งานกระบวนการกลับสถานะข้อมูลหรือคาต้าทอกเกิล

5. สถานะบัสค้างและการสูญเสียรูปแบบการส่งข้อมูล

ความผิดพลาด 2 ชนิดนี้จะขอกล่าวรวมกันเพราะผลที่เกิดจากความผิดพลาดทั้งคู่จะทำให้บัสทั้งบัสไม่สามารถใช้งานต่อไปได้ นั่นคือ ไม่มีอุปกรณ์ตัวใดจะสามารถส่งหรือรับข้อมูลได้อีก (Deadlock)

สถานะบัสค้าง (Babble) คือ การตรวจพบว่าบัสไม่อยู่ในสถานะไอเดิล หลังจากบัสเข้าสู่สถานะสิ้นสุดเฟรมข้อมูล (End of frame : EOF) ตามปกติเมื่อบัสเข้าสู่สถานะ EOF แพ็กเก็ตเพียงชนิดเดียวที่สามารถเกิดขึ้นได้คือแพ็กเก็ต SOF เพื่อเริ่มต้นเฟรมใหม่ ซึ่งถ้าตรวจพบว่าสถานะของบัสไม่เป็นไอเดิล จึงทำให้อุปกรณ์ทุกตัวรวมถึงโฮสต์คิดว่ามีการใช้งานบัสอยู่จึงไม่สามารถส่งข้อมูลใดๆ ลงมาในบัสได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. การสูญเสียรูปแบบการส่งข้อมูล (Loss of Activity : LOA)

จะเกิดขึ้นจากการที่มีอุปกรณ์ตัวใดตัวหนึ่งส่งข้อมูลเข้ามาในบัสแต่ไม่มีสถานะ EOP ปิดท้ายแพ็กเก็ต ส่งผลให้อุปกรณ์ที่เหลือไม่สามารถทำงานต่อได้เพราะเข้าใจว่าบัสไม่ว่าง

ความผิดพลาดทั้งสองชนิดนี้ถือเป็นความผิดพลาดขั้นรายแรง โดยปกติหากโฮสต์ตรวจพบว่าอุปกรณ์ใดเป็นตัวสร้างปัญหาจะทำการปิดการทำงานของพอร์ตนั้นๆ เพื่อให้บัสยังคงสามารถทำงานต่อไปได้ แต่ถ้าหากโฮสต์เองไม่สามารถปิดการทำงานของพอร์ตนั้นได้ ซึ่งอาจเกิดจากการที่อุปกรณ์ไม่ยอมปล่อยบัสว่าง ทำให้โฮสต์ไม่สามารถส่งคำสั่งลงไปยังฮับเพื่อปิดการทำงานได้ โฮสต์จะต้องรีเซตบัสทั้งหมดเพื่อเริ่มงานทำงานใหม่

2.2.7 ดิสคริปเตอร์และกระบวนการตั้งค่า

1. ข้อมูลภายในดิสคริปเตอร์

จะกล่าวถึงรายละเอียดแบบเจาะลึกกว่าแต่ละไบต์ของข้อมูลข้างในเก็บค่าอะไรอยู่บ้าง

1. ดีไวซ์ดิสคริปเตอร์ มีรายละเอียดดังในตารางที่ 2.4
2. คอนฟิกิวเรชั่นดิสคริปเตอร์ มีรายละเอียดดังตารางที่ 2.5
3. อินเตอร์เฟสดิสดิสคริปเตอร์ มีรายละเอียดดังในตารางที่ 2.6
4. เอ็นพอยต์ดิสคริปเตอร์ มีรายละเอียดดังในตารางที่ 2.7
5. ฮับคลาสดิสคริปเตอร์เป็นดิสคริปเตอร์ใช้สำหรับรายงานข้อมูลของอุปกรณ์ที่มาเชื่อมต่อพอร์ตต่างๆ กลับไปยังโฮสต์ มีรายละเอียดดังในตารางที่ 2.8

2. ขั้นตอนการตั้งค่าอุปกรณ์ (Configuration process)

จากเนื้อหาที่ผ่านมาทั้งหมด ทำให้ทราบถึงส่วนประกอบต่างๆ ทั้งหมดของ USB ต่อไปจะเป็นขั้นตอนการทำงานของระบบ เริ่มตั้งแต่มีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในระบบว่ามีลำดับขั้นอย่างไรบ้าง การสื่อสารข้อมูลระหว่างโฮสต์และตัวอุปกรณ์ตั้งแต่อุปกรณ์ถูกเชื่อมต่อกับระบบไปจนถึงตัวอุปกรณ์สามารถทำงานตามหน้าที่ของมันได้ ขั้นตอนทั้งหมดนี้เรียกรวมว่า การตั้งค่า (Config) อุปกรณ์เข้ากับระบบ

เมื่อมีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในระบบ ก่อนที่ตัวอุปกรณ์จะทำงานได้ โฮสต์จำเป็นต้องทราบข้อมูลที่ใช้ควบคุมการทำงานของตัวอุปกรณ์นั้นๆ รวมไปจนถึงความต้องการทรัพยากรด้านต่างๆ ไม่ว่าจะเป็นไฟเลี้ยงหรือความเร็วในการรับส่งข้อมูลเพื่อพิจารณาว่า ระบบ ณ ขณะนั้นสามารถรองรับการทำงานของตัวอุปกรณ์ใหม่นี้ได้หรือไม่ ซึ่งข้อมูลต่างๆ เหล่านี้จะบรรจุอยู่ในดิสคริปเตอร์ต่างๆ ภายในตัวอุปกรณ์ กระบวนการในการอ่านดิสคริปเตอร์ต่างๆ นี้ เรียกว่า การร้องขอข้อมูล (Data request) ซึ่งจะกล่าวโดยละเอียดในหัวข้อถัดไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.4 รายละเอียดข้อมูลแต่ละไบต์ของดีไวซ์ดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ความยาวของดีไวซ์ดิสคริปเตอร์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น Device Descripter = 01H
2	USB	2	BCD	เวอร์ชันของ USB ที่อุปกรณ์นี้ใช้งาน เก็บในรูปแบบของ BCD
4	Device class	1	Class	บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสใด
5	Device subclass	1	Subclass	บอกการจัดหมวดหมู่ของอุปกรณ์ ว่าอยู่ในซับคลาสใด
6	Device Protocal	1	Protocal	บอกรูปแบบการติดต่อหรือรหัสโปรโตคอลของอุปกรณ์
7	MaxPacket Size0	1	จำนวน	บอกขนาดข้อมูลที่สามารถส่งได้ต่อแพ็กเก็ตของเอ็นด์พอยต์ 0 (ค่าที่ใช้งานคือ 8,16,32 หรือ 64 ไบต์เท่านั้น)
8	Vendor	2	ID	ค่า ID ของ Vendor
10	Product	2	ID	ชนิดของอุปกรณ์ที่ผลิต
12	Device	2	BCD	เวอร์ชันของตัวอุปกรณ์ เก็บในรูปแบบ BCD
14	Manufac-turer	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์ที่แสดงข้อมูลและชื่อผู้ผลิต
15	Product	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์แสดงชนิดของอุปกรณ์
16	Serial number	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์แสดงรุ่นของอุปกรณ์
17	Num Configuration	1	จำนวน	บอกจำนวนคอนฟิกิวเรชั่นทั้งหมดที่อุปกรณ์รองรับได้

ก่อนจะอธิบายขั้นตอนการตั้งค่าจะขอขยายความในเรื่องโครงสร้างบัสของ USB เพิ่มเติมอีกเล็กน้อย เพื่อไม่ให้เกิดความสับสนจากคำว่า "บัส" การเชื่อมต่ออุปกรณ์แบบบัสนั้นอุปกรณ์ทุกตัวในระบบจะได้รับข้อมูลทุกๆ แพ็กเก็ต แต่จะเลือกเก็บเฉพาะข้อมูลที่เป็นของตัวเองเท่านั้น โดยสังเกตจากแอดเดรสที่นำหน้าของแต่ล่ะแพ็กเก็ตซึ่งถ้าตีความจากข้อความข้างต้นฮับซึ่งเป็นอุปกรณ์ที่ขยายการเชื่อมต่อจะทำหน้าที่กระจายข้อมูลออกไปยังอุปกรณ์ทุกๆ ตัวที่มาต่ออยู่เท่านั้น เช่นเดียวกับฮับของระบบเน็ตเวิร์ก LAN (Local Area Network) แต่เนื่องจาก USB มีสถานะบางสถานะที่กำหนดจากระดับแรงดันภายในสายสัญญาณโดยตรง เช่น สถานะรีเซต สถานะประหยัดพลังงาน และสถานะเชื่อมต่อหรือปลดออกจากระบบ เป็นต้น ซึ่งสถานะเหล่านั้นไม่สามารถส่งผ่าน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

จากฮับรวมไปกับข้อมูลอื่นๆ ได้ ดังนั้นฮับของ USB จะต้องทำหน้าที่มากกว่าการกระจายข้อมูลตามปกติ นั่นคือต้องมีการตรวจสอบการเชื่อมต่อหรือปลดออกของอุปกรณ์ได้ รับคำสั่งรีเซตเพื่อรีเซตบัสเฉพาะพอร์ตที่กำหนด ดังนั้นถ้ากล่าวถึงการส่งสถานะพิเศษเหล่านี้

ตารางที่ 2.5 รายละเอียดข้อมูลแต่ละไบต์ของคอนฟิกิวเรชั่นดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของคอนฟิกิวเรชั่นดิสคริปเตอร์นี้ใน .หน่วยไบต์
1	Descripter type	1	ค่าคงที่	ระบุชนิดของดิสคริปเตอร์ถ้าเป็น Configuration descripter = 02H
2	TotalLength	2	จำนวน	ขนาดข้อมูลทั้งหมดที่ได้จากการอ่านดิสคริปเตอร์ (เมื่อส่งคำสั่งขออ่านคอนฟิกิวเรชั่นดิสคริปเตอร์ ข้อมูลที่ได้คือ ข้อมูลคอนฟิกิวเรชั่นดิสคริปเตอร์ ทั้งหมดเรียงกันรวมจาก คอนฟิกิวเรชั่นดิสคริปเตอร์ อินเตอร์เฟสดิสคริปเตอร์และเอ็นด์พอยต์ดิสคริปเตอร์)
4	Numinterface	1	จำนวน	บอกจำนวนอินเตอร์เฟสทั้งหมดที่อุปกรณ์รองรับได้
5	Configuration Value	1	จำนวน	ค่าหมายเลขคอนฟิกิวเรชั่นที่ใช้ในการเลือกคอนฟิกิวเรชั่น นี้
6	Configuration	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์ใช้อธิบายข้อมูลของคอนฟิกิวเรชั่น นี้
7	Attributes	1	กำหนดเป็นบิต	แต่ละบิตบอกคุณสมบัติของคอนฟิกิวเรชั่น ดังนี้ บิต D7 ใช้พลังงานจากบัส บิต D6 ใช้แหล่งงานของตัวเอง บิต D5 การรองรับ Remote wakeup บิต D4-D0 สงวนไว้ไม่ใช้งาน หากบิตใดมีค่าเป็น "1" คือมีคุณสมบัตินั้นๆ
8	MaxPower	2	มิลลิแอมป์	บอกค่ากระแสไฟฟ้าที่คอนฟิกิวเรชั่น นี้ต้องการเมื่อทำงานเต็มความสามารถของมันแต่ละหน่วยเท่ากับ 2 mA

ไปยังตัวอุปกรณ์จะหมายความว่าโฮสต์ส่งคำสั่งไปยังฮับที่ตัวอุปกรณ์เชื่อมต่ออยู่แล้วฮับจะนำคำสั่งไปแปลงเป็นระดับสัญญาณในสายสัญญาณเฉพาะพอร์ตเท่านั้น ไม่ใช่การส่งสถานะออกมาจากโฮสต์โดยตรงแล้วกระจายไปยังอุปกรณ์ทุกๆ ตัว ลำดับขั้นตอนที่จะกล่าวถึงนี้เป็นลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานพื้นฐาน ในการใช้งานจริงอาจจะมีการตัดหรือเพิ่มบางขั้นตอน หรืออาจจะมีการสลับบางขั้นตอน เพื่อให้เกิดความเหมาะสมกับการทำงานแต่ละอย่าง

ขั้นที่ 1 ตรวจสอบสถานะการเชื่อมต่อและชนิดของอุปกรณ์

เมื่อมีอุปกรณ์ตัวใหม่เชื่อต่อเข้ามาในระบบฮับจะทราบจากการตรวจสอบระดับแรงดันของสายสัญญาณว่าอยู่ในสถานะใดซึ่งจะทราบด้วยว่า อุปกรณ์ที่เชื่อมต่อเข้ามานั้นเป็นอุปกรณ์ความเร็วสูงหรือความเร็วต่ำ หลังจากนั้นจะเซตแฟลกของพอร์ตนั้นๆ เพื่อรอการอ่านกลับจากโฮสต์ ในขั้นตอนนี้ยังไม่มีการจ่ายไฟเลี้ยงไปยังตัวอุปกรณ์

ตารางที่ 2.6 รายละเอียดข้อมูลแต่ละไบต์ของอินเตอร์เฟสดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของอินเตอร์เฟสดิสคริปเตอร์นี้ในหน่วยไบต์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น Interface Descripter = 04H
2	Interface Number	1	จำนวน	ค่าหมายเลขอินเตอร์เฟสที่ใช้เลือกอินเตอร์เฟสนี้
3	Alternate Setting	1	จำนวน	ค่าหมายเลข Alternate Setting ในแต่ละอินเตอร์เฟส
4	NumEnd Points	1	จำนวน	บอกจำนวนเอ็นด์พอยต์ที่ใช้งานในอินเตอร์เฟสนี้ (ไม่รวมเอ็นด์พอยต์ 0)
5	Interface class	1	Class	บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสใดในกรณีที่แต่ละอินเตอร์เฟสมีอยู่ในคลาสที่ต่างกัน
6	Interface subclass	1	Subclass	บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสย่อยใดในกรณีที่แต่ละอินเตอร์เฟสมีอยู่ในคลาสย่อยที่ต่างกัน
7	Interface Protocal	1	Interface	บอกรหัสโปรโตคอลของอุปกรณ์ว่าใช้โปรโตคอลลักษณะใด ในกรณีที่แต่ละอินเตอร์เฟสใช้โปรโตคอลที่ต่างกัน
8	Interface	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์ที่ใช้อธิบายข้อมูลของอินเตอร์เฟสนี้

ขั้นที่ 2 จ่ายไฟเลี้ยงไปยังพอร์ต

หลังจากโฮสต์ทราบว่า พอร์ตใดมีอุปกรณ์ตัวใหม่ต่อเข้ามา โดยอ่านจากแฟลกสถานะของฮับ โฮสต์จะสั่งให้ฮับจ่ายไฟเลี้ยงแก่พอร์ตของตัวอุปกรณ์ใหม่นั้น (ในกรณีที่ฮับไม่สามารถควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจ่ายไฟเลี้ยงได้สามารถข้ามขั้นตอนไปได้) โดยในขั้นตอนนี้อุปกรณ์จะดึงกระแสได้ไม่เกิน 100mA อุปกรณ์จะดึงกระแสได้เต็มที่ก็ต่อเมื่อได้รับการตั้งค่าจากโฮสต์แล้วเท่านั้น

ขั้นที่ 3 รีเซตพอร์ตของอุปกรณ์ที่เชื่อมใหม่

เมื่ออุปกรณ์ตัวใหม่ได้รับไฟเลี้ยงแล้วโฮสต์จะส่งสถานะรีเซตไปยังพอร์ตนั้นๆ โดยส่งคำสั่งไปยังฮับที่ควบคุมการทำงานของพอร์ตนั้นเมื่ออุปกรณ์ได้รับสถานะนี้แล้วจะต้องรีเซตตัวเองและปรับค่าต่างๆ ไปยังสถานะเริ่มต้น ทำให้แอดเดรสของอุปกรณ์ที่ได้รับการรีเซตไปอยู่ที่ "0" ทำให้โฮสต์สามารถติดต่ออุปกรณ์ตัวใหม่นี้ได้เพราะถ้าไม่มีการตั้งแอดเดรสใหม่โฮสต์จะไม่ทราบว่าต้องส่งคำสั่งไปที่ใด

ตารางที่ 2.7 รายละเอียดข้อมูลแต่ละไบต์ของเอ็นด์พอยน์ดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของอินเตอร์เฟสดิสคริปเตอร์นี้ในหน่วยไบต์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น End point Descripter = 05H
2	Endpoint Address	1	เอ็นด์พอยต์	บอกค่าแอดเดรสของเอ็นด์พอยต์และทิศทางการส่งข้อมูลดังนี้ บิต D0-D3 บอกหมายเลขแอดเดรสของเอ็นด์พอยต์ บิต D4-D6 สงวนไว้ไม่ใช้งาน บิต D7 บอกทิศทางการส่งข้อมูลของเอ็นด์พอยต์ - เป็น "0" คือส่งข้อมูลกลับมายังโฮสต์ - เป็น "1" คือรับข้อมูลจากโฮสต์
3	Attributes	1	กำหนดเป็นบิต	บอกชนิดของการรับส่งข้อมูลที่เอ็นด์พอยต์นี้ใช้ D1=0, D0=0 Control Transfer D1=0,D0=1 Isochronous Transfer D1=1, D0=0 Bulk Transfer D1=1 , D0=1 Interrupt Transfer บิตที่เหลือนอกจากนี้สงวนไว้ไม่ใช้งาน
4	MaxPacket Size	2	จำนวน	ขนาดของข้อมูลใหญ่ที่สุดที่เอ็นด์พอยต์นี้สามารถรับส่งได้
6	**Interval**	1	**จำนวน**	เวลาในการวนอ่านข้อมูลจากโฮสต์ หน่วยเป็นมิลลิวินาที (ms)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นที่ 4 โฮส์ร้องขอข้อมูลดีไวซ์คริปเตอร์จากตัวอุปกรณ์

หลังจากแอดเดรสของอุปกรณ์ตัวใหม่ถูกรีเซตเป็นศูนย์โฮสต์จะเริ่มร้องขอข้อมูลดิสคริปเตอร์ต่างๆ จากตัวอุปกรณ์เพื่อนำมาพิจารณว่าสามารถรองรับอุปกรณ์ตัวใหม่ได้หรือไม่ โดยดิสคริปเตอร์ตัวแรกที่จะถูกอ่านออกมาก็คือ ดีไวซ์ดิสคริปเตอร์ การอ่านดิสคริปเตอร์จากตัวอุปกรณ์ก็คือ การส่งคำสั่งร้องขอเข้าไปยังตัวอุปกรณ์ ปลายทางที่จะคอยรับคำสั่งคือ เอ็นด์พอยต์ควบคุม ที่มีหมายเลขเอ็นด์พอยต์เป็น "0" ดังนั้นการอ่านดีไวซ์ดิสคริปเตอร์คือการส่งคำสั่งร้องขอดีไวซ์ดิสคริปเตอร์ไปยังอุปกรณ์แอดเดรส "0" เอ็นด์พอยต์ "0"

ตารางที่ 2.8 รายละเอียดข้อมูลแต่ละไบต์ของฮับคลาสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของอินเตอร์เฟสดิสคริปเตอร์นี้ ในหน่วยไบต์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น Hub class Descripter = 29H
2	NbrPorts	1	จำนวน	บอกจำนวนพอร์ตทั้งหมดที่ให้อุปกรณ์อื่นๆ มาเชื่อมต่อกับฮับได้
3	Hub Characteristic	2	กำหนดเป็นบิต	D1-D0 บอกข้อมูลการจ่ายพลังงานของฮับ 00= เปิดจ่ายพลังงานทุกพอร์ตในครั้งเดียว 01= แยกจ่ายพลังงานเป็พอร์ตๆ แยกกัน 10 และ 11= ไม่สามารถปิดการจ่ายพลังงานได้ D2 ระบุว่าเป็นคอมพาวน์ดีไวซ์หรือไม่ 0= ไม่เป็นคอมพาวน์ดีไวซ์ 1= เป็นคอมพาวน์ดีไวซ์ D4-D3 บอกข้อมูลการป้องกันกระแสเกิน 00= ป้องกันกระแสเกินแบบรวม 01= แยกป้องกันได้อิสระในแต่ละพอร์ต 10 และ 11= ไม่สามารถป้องกันการดึงกระแสเกินได้ใช้กับฮับที่อาศัยพลังงานจากบัสเท่านั้น
5	PowerOn2 PowerGood	1	จำนวน	บอกค่าเวลาที่ฮับใช้ในการตั้งค่าภายใน ตั้งแต่เริ่มจ่ายกระแสไฟฟ้าจนกระทั่งทำงานได้สมบูรณ์ สามารถรับคำสั่งได้ ใช้สำหรับกำหนดค่าหน่วงเวลาหลังจากจ่ายไฟเลี้ยง
6	HubContr Current	1	จำนวน	ค่ากระแสไฟฟ้าที่ใช้สำหรับฮับคอนโทรลเลอร์ หน่วยละ 2 mA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.8 (ต่อ) รายละเอียดข้อมูลแต่ละไบต์ของฮับคลาสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
7	Device Removable	1	กำหนดเป็นบิต	ขนาดข้อมูลของฟิลด์นี้ขึ้นอยู่กับจำนวนพอร์ตของฮับ เนื่องจากข้อมูล 1 บิต แทนพอร์ต 1 พอร์ต แต่ละบิตบอกชนิดของอุปกรณ์ว่าเป็นอุปกรณ์ที่สามารถถอดเปลี่ยนได้หรือต่อถาวร ถ้าบิตใดเป็น "0" แทนถึงอุปกรณ์ที่ถอดเปลี่ยนได้ แต่ถ้าเป็น "1" แทนถึงอุปกรณ์ที่ต่อถาวร
-	PortPwrCtr Mask	-	กำหนดเป็นบิต	ตำแหน่งของข้อมูลนี้ขึ้นอยู่กับขนาดของข้อมูล Device Removable ถ้าบิตใดเป็น "0" หมายความว่า พอร์ตนั้นควบคุมการจ่ายไฟแบบรวม แต่ถ้าเป็น "1" หมายความว่าพอร์ตนั้นแยกจ่ายไฟเลี้ยงเป็นอิสระ

ข้อมูลที่ต้องการเป็นค่าแรกคือ ขนาดแพ็กเก็ตข้อมูลใหญ่ที่สุดที่สามารถรับส่งได้ต่อครั้ง ซึ่งข้อมูลนี้จะอยู่ในฟิลด์ MaxPacketSize0 (ก่อนอ่านข้อมูลนี้ออกมากำหนดขนาดแพ็กเก็ตใหญ่ที่สุดเป็น 8 ไบต์)

ขั้นที่ 5 กำหนดแอดเดรสให้แก่ตัวอุปกรณ์

เมื่อทราบขนาดแพ็กเก็ตข้อมูลที่จะใช้ในการติดต่อจากการอ่านดีไวซ์ดิสคริปเตอร์แล้ว โฮสต์จะกำหนดแอดเดรสให้แก่ตัวอุปกรณ์ โดยใช้คำสั่ง Set_Address ซึ่งค่าแอดเดรสที่กำหนดขึ้นนี้จะเป็นแอดเดรสที่ไม่ซ้ำกับอุปกรณ์ตัวอื่นๆ ในระบบเลยเพื่อให้โฮสต์สามารถติดต่อสื่อสารกับอุปกรณ์แต่ละตัวได้

ขั้นที่ 6 โฮสต์อ่านดิสคริปเตอร์ต่างๆ เพื่อพิจารณาการรองรับได้ของระบบ

ก่อนที่โฮสต์จะตั้งค่าตัวอุปกรณ์โฮสต์จะต้องอ่านค่าดิสคริปเตอร์ต่างๆ ภายในตัวอุปกรณ์เพื่อพิจารณาว่าระบบมีทรัพยากรเพียงพอตามที่ตัวอุปกรณ์ต้องการหรือไม่ ข้อมูลหลักๆ ที่จำเป็นในการพิจารณาคือ กระแสไฟฟ้าและความเร็วในการรับส่งข้อมูลที่ต้องการ โดยข้อมูลด้านกระแสไฟฟ้าจะได้จากฟิลด์ MaxPower ภายในคอนฟิกิวเรชั่นดิสคริปเตอร์ ส่วนข้อมูลด้านความเร็วจะได้จากฟิลด์ MaxPacketSize ภายในแต่ละเอ็นด์พอยต์ดิสคริปเตอร์

การพิจารณาเริ่มจากตรวจสอบว่าอุปกรณ์ตัวนี้มีโหมดการทำงานกี่โหมด (กี่คอนฟิกิวเรชั่น) จากข้อมูลในฟิลด์ Num-Configurations ของดีไวซ์ดิสคริปเตอร์ หลังจากนั้นส่งคำร้องขอคอนฟิกิวเรชั่นดิสคริปเตอร์ไปยังตัวอุปกรณ์ด้วยคำสั่งนี้อุปกรณ์จะส่งดิสคริปเตอร์ที่เหลือทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

ออกมาว่าจะเป็นคอนฟิกิวเรชั่น อินเตอร์เฟส และเอ็นด์พอยต์ดิสคริปเตอร์ในครั้งเดียว เมื่อได้ข้อมูลทั้งหมดออกมาแล้วจะต้องนำมาตัดแยกเป็นแต่ละส่วน เมื่อแยกคอนฟิกิวเรชั่นดิสคริปเตอร์ออกมาได้แล้ว โฮสต์จะทราบว่า ถ้าตั้งค่าให้อุปกรณ์ทำงานในแต่ละโหมดจะต้องจ่ายกระแสออกไปเท่าใด จากข้อมูลในฟิลด์ MaxPower ทำให้โฮสต์ทราบว่า สามารถจ่ายไฟเลี้ยงให้แก่อุปกรณ์ในคอนฟิกิวเรชั่นใดได้บ้าง

ข้อมูลถัดมาคือ อินเตอร์เฟสดิสคริปเตอร์จะให้ข้อมูลรายละเอียดของแต่ละอินเตอร์เฟส (เปรียบเสมือนแต่ละหน้าที่การทำงาน) ว่าจัดอยู่ในคลาสใด ใช้โปรโตคอลพิเศษแบบใดในการติดต่อรวมไปถึงจำนวนเอ็นด์พอยต์ที่ต้องใช้ติดต่อในอินเตอร์เฟสนั้นๆ หลังจากการอินเตอร์เฟสดิสคริปเตอร์ข้อมูลที่ตามมาคือ เอ็นด์พอยต์ดิสคริปเตอร์ซึ่งจะแจ้งความต้องการด้านปริมาณการรับส่งข้อมูลว่าแต่ละเอ็นด์พอยต์รับส่งข้อมูลแบบใด และต้องการความเร็วในการรับส่งข้อมูลเท่าใด โดยชนิดของการรับส่งข้อมูลทราบได้จากฟิลด์ Attributes ส่วนความเร็วของการรับส่งข้อมูลทราบได้จากฟิลด์ MaxPacketSize

เมื่อได้ข้อมูลเหล่านี้มาโฮสต์จะนำไปเปรียบเทียบกับทรัพยากรที่ระบบเหลืออยู่ว่าสามารถรองรับการจ่ายไฟเลี้ยงและความเร็วในการรับส่งข้อมูลเพียงพอกับที่อุปกรณ์ต้องการหรือไม่หากพิจารณาแล้วว่าสามารถรองรับได้ก็จะข้ามไปสู่การตั้งค่าในขั้นถัดไปได้แต่ถ้าพิจารณาแล้วพบว่าระบบเหลือทรัพยากรไม่เพียงพอ โฮสต์จะพิจารณาคอนฟิกิวเรชั่นและอินเตอร์เฟสอื่นๆ ที่ต้องการทรัพยากรน้อยกว่าในกรณีที่อุปกรณ์นั้นมีคอนฟิกิวเรชั่นและอินเตอร์เฟสมากกว่าหนึ่ง แต่หากพิจารณาคอนฟิกิวเรชั่นหรืออินเตอร์เฟสทั้งหมดแล้วยังไม่สามารถรองรับความต้องการของอุปกรณ์ได้โฮสต์ก็จะหยุดการตั้งค่าอุปกรณ์ ซึ่งจะทำให้อุปกรณ์นั้นไม่สามารถทำงานได้

ขั้นที่ 7 โฮสต์กำหนดเลือกคอนฟิกิวเรชั่นและอินเตอร์เฟส

หลังจากพิจารณาแล้วว่าระบบสามารถรองรับความต้องการของอุปกรณ์ตัวใหม่ได้โฮสต์จะส่งคำสั่ง Set_Configuration และ Set_Interface โดยระบุหมายเลขคอนฟิกิวเรชั่นและหมายเลขอินเตอร์เฟสที่เลือกเข้าไปในตัวคำสั่งหลังจากสิ้นสุดขั้นตอนนี้ อุปกรณ์จะดึงไฟเลี้ยงจากฮับเต็มพิกัดที่ระบุไว้ในคอนฟิกิวเรชั่นดิสคริปเตอร์และเริ่มทำงานตามหน้าที่ของมัน ถือเป็นการจบกระบวนการตั้งค่าตัวอุปกรณ์

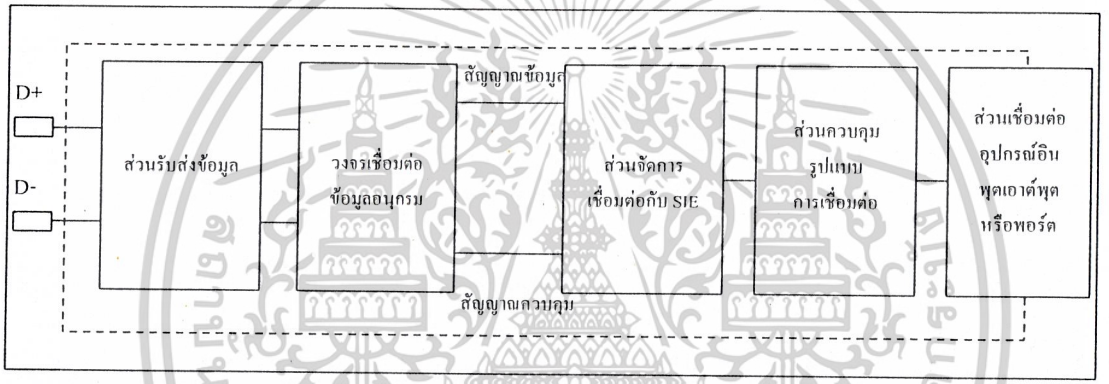
และสำหรับกรณีที่ต้องตั้งค่าอุปกรณ์หลายๆ ตัวพร้อมกัน เช่น การเปิดเครื่องคอมพิวเตอร์ขึ้นมาในขณะที่มีอุปกรณ์เชื่อมต่ออยู่มากกว่า 1ตัว โฮสต์จะทำการตั้งค่าอุปกรณ์ไปทีละตัวตามกระบวนการข้างต้นจนครบทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 ส่วนติดต่อพอร์ต USB

ในรูปที่ 2.21 แผนผังการทำงานเบื้องต้นของส่วนติดต่อพอร์ต USB ซึ่งมีส่วนประกอบ 5 ส่วน คือ

1. ส่วนรับส่งข้อมูล (Transceiver)
2. วงจรเชื่อมต่อข้อมูลอนุกรม (Serial Interface Engine : SIE)
3. ส่วนจัดการเชื่อมต่อกับ SIE (SIE Interface)
4. ส่วนควบคุมรูปแบบการเชื่อมต่อ (Protocal controller)
5. ส่วนเชื่อมต่ออุปกรณ์อินพุตเอาต์พุตหรือพอร์ต (Input/Output : I/O)



รูปที่ 2.21 แผนผังการทำงานเบื้องต้นของส่วนติดต่อพอร์ต USB

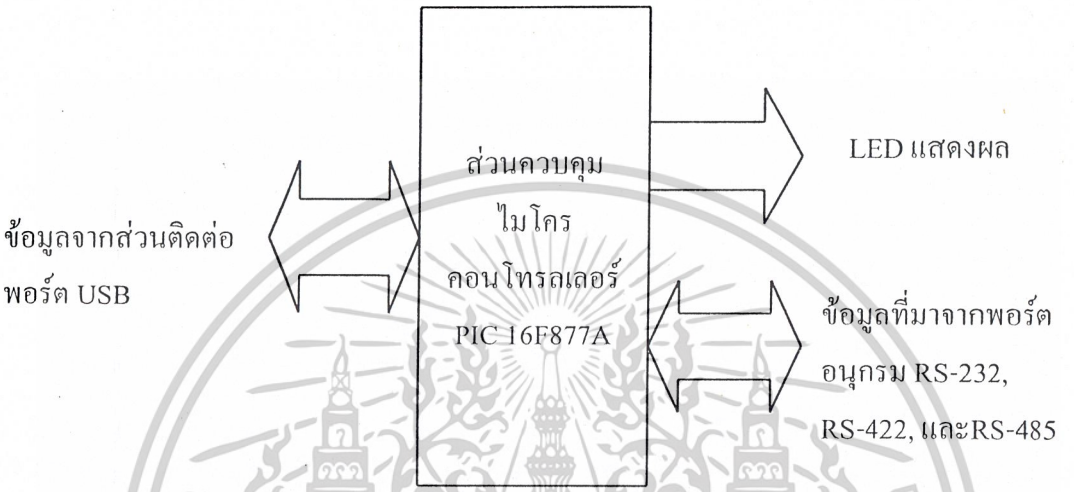
ส่วนติดต่อพอร์ต USB จะทำหน้าที่จัดรูปแบบข้อมูลที่รับมาจากคอนโทรลเลอร์ก่อนที่จะส่งไปยังพอร์ต USB และในทางกลับกันเมื่อรับข้อมูลมาจากพอร์ต USB ก็จะจัดข้อมูลนั้นให้อยู่ในรูปแบบที่สามารถติดต่อกับคอนโทรลเลอร์ได้ ซึ่งในส่วนติดต่อพอร์ต USB นี้จะสามารถที่จะเก็บข้อมูลต่างๆ ที่ผู้ใช้กำหนดได้อีกด้วย เช่น Manufacturer Name, Manufacturer ID, Vendor ID (VID), Product ID (PID), Strings Description, Serial Number, Isochronous Transfer Mode และ Power Descriptor

2.4 ส่วนควบคุม

ส่วนควบคุมซึ่งประกอบด้วยไมโครคอนโทรลเลอร์ PIC 16F877A เป็นอุปกรณ์ควบคุมหลักซึ่งจะทำหน้าที่คอยรับและส่งข้อมูลที่เข้ามาจากพอร์ตต่างๆ คือ พอร์ต USB, RS-232, RS-422 และRS-485 อีกทั้งยังมี LED เป็นส่วนแสดงผล และหน้าที่อีกอย่างหนึ่งของส่วนควบคุมคือ ทำการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จัดรูปแบบข้อมูลที่รับมาจากส่วนติดต่อพอร์ต USB ไปเป็นแบบพอร์ตอนุกรม RS-232, RS-422 และRS-485 และในทางกลับกันก็จะส่งข้อมูลที่รับจากพอร์ตอนุกรม RS-232, RS-422 และRS-485 ไปยังส่วนติดต่อพอร์ต USB ดังรูปที่ 2.22



รูปที่ 2.22 แผนผังการทำงานของส่วนควบคุม

2.5 การสื่อสารอนุกรม

การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงภายนอกหรือคอมพิวเตอร์ด้วยกันมี 2 รูปแบบคือ การรับส่งข้อมูลแบบขนานและการรับส่งข้อมูลแบบอนุกรม

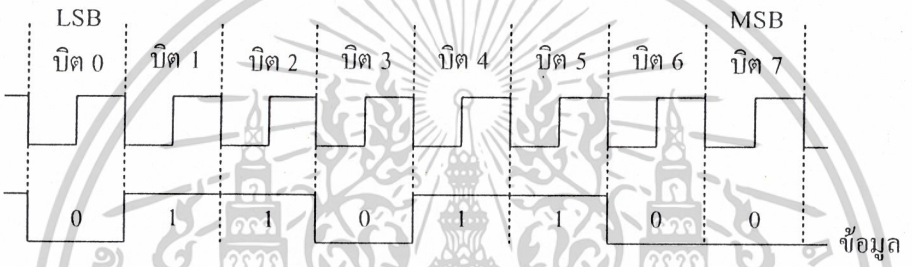
การรับส่งข้อมูลแบบขนานเป็นการรับและส่งข้อมูลคราวละ 4 ถึง 8 บิต (Bit) ในเวลาเดียวกัน ทำให้การรับและส่งข้อมูลมีความเร็วสูง แต่จำนวนสายที่ใช้ในการถ่ายทอดข้อมูลก็มีมากเท่ากับจำนวนบิตของข้อมูลที่ทำการถ่ายทอด นอกจากนั้นยังมีสายที่ใช้สำหรับควบคุมและตรวจสอบการรับส่งข้อมูลด้วย ซึ่งอาจต้องใช้สายมากเป็น 2 เท่าของจำนวนบิตข้อมูลก็ได้

ในขณะที่การรับส่งข้อมูลแบบอนุกรมจะเป็นการรับส่งข้อมูลครั้งละ 1 บิต โดยมีรูปแบบการรับส่งที่เป็นมาตรฐาน ต้องมีการตรวจสอบความพร้อมในการรับและส่งข้อมูลของตัวส่งและตัวรับ การรับส่งข้อมูลแบบอนุกรมมีข้อดีในเรื่องจำนวนสายสัญญาณที่น้อยมากและไม่แปรผันตามจำนวนบิตของข้อมูล ระยะทางในการรับส่งข้อมูลสูงกว่าแบบขนานมาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

การสื่อสารแบบอนุกรมแบ่งออกเป็น 2 แบบคือ การสื่อสารอนุกรมแบบซิงโครนัส (Synchronous) และการสื่อสารอนุกรมแบบอะซิงโครนัส (Asynchronous)

1. การสื่อสารอนุกรมแบบซิงโครนัส จะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้น คือ สัญญาณนาฬิกา ข้อมูลและกราวด์ รูปที่ 2.23 แสดงให้เห็นถึงแผนผังเวลาของการสื่อสารข้อมูลแบบซิงโครนัส



รูปที่ 2.23 แผนผังเวลาของการสื่อสารข้อมูลแบบซิงโครนัส

2. การสื่อสารอนุกรมแบบอะซิงโครนัส คือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราความเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอดหรือบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

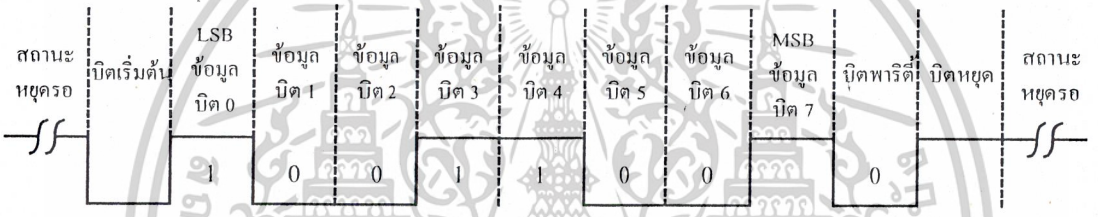
1. บิตเริ่มต้น
2. บิตข้อมูลแบบอนุกรม มีขนาด 5, 6, 7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) มีขนาด 1 บิต หรือไม่มีบิต
4. บิตปิดท้ายหรือบิตหยุด (Stop Bit) มีขนาด 1, 1.5 หรือ 2 บิต

รูปที่ 2.24 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูลขา DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (Waiting Stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่าบิตเริ่มต้น (Start Bit) จากนั้นบิตข้อมูลจะถูกส่งออกไปโดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งอาจมีจำนวน 5, 6, 7 หรือ 8 บิตก็ได้ จากนั้นตามด้วยบิตพาริตี้ ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งก็คือ บิตปิดท้ายหรือบิตหยุด โดยจะเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำให้ขา DATA มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต, 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่า ได้แก่ 110, 150, 300, 600, 1,200, 2,400, 4,800, 9,600 และ 19,200 บิตต่อวินาที (Bit/sec) โดยมีค่ามากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากบอดเรตคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมุติว่า ข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์ จะมีความยาวเท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9,600 Bit/sec ก็จะสามารถส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที (Byte/sec)



รูปที่ 2.24 รูปแบบของข้อมูลแบบอะซิงโครนัส

การตรวจสอบพาริตี้สามารถกำหนดเป็นแบบคี่ (Odd) แบบคู่ (Even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์ รวมพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99H หรือ 10011001B จะเห็นว่าข้อมูลในไบต์มีจำนวนลอจิก "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของพาริตี้บิตจะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asynchronous Receiver Transmitter) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คี่หรือพาริตี้คู่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าข้อมูลที่ทำการส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ไอซี UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ไอซี UART เบอร์ 8250 ไอซี UART เหล่านี้มีระดับแรงดันของลอจิกเป็นแบบทีทีแอล +5 โวลต์ (Volts : V) แต่เพื่อให้แรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะไกลมากยิ่งขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก "0" จะมีระดับแรงดัน –3V ถึง –12V และลอจิก "1" มีระดับแรงดัน +3V ถึง +12V

2.5.1 RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็ม (MODEM) เพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้ส่งผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งอยู่ห่างไกลโดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์ (Connector) เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3V จนถึง –12V แสดงว่ามีข้อมูล (Mark) และ +3V จนถึง +12V แสดงว่าเป็นช่องว่าง (Space) มีอัตราความเร็วการส่งข้อมูล 20 Kbit/sec

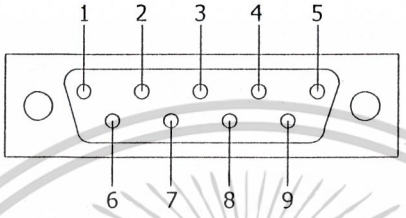
มาตรฐาน RS-232 ถูกใช้ในการกำหนดรูปแบบการสื่อสารข้อมูลกันระหว่างอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

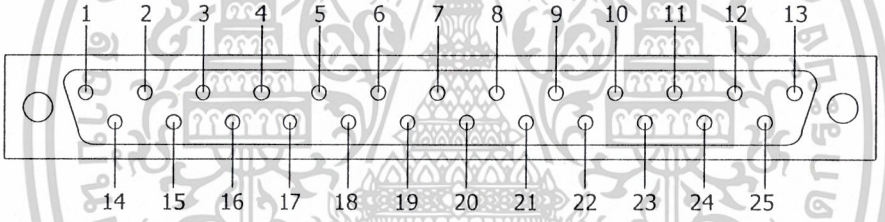
สำหรับการใช้งานคอมพิวเตอร์ พอร์ตอนุกรม RS-232 ถูกใช้เพื่อเชื่อมต่อกับโมเด็ม เมาส์ และเครื่องพิมพ์ที่สามารถติดต่อทางพอร์ตอนุกรมได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอเน็กเตอร์แบบ DB-25 ตัวผู้ หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยมีใช้งานมาในอดีตไม่ค่อยมีความสำคัญมากนักจึงถูกยกเลิกไป โดยแสดงตำแหน่งขาและรูปร่างในรูปที่ 2.25



(ก) คอนเน็กเตอร์อนุกรม 9 ขา หรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)



(ข) คอนเน็กเตอร์อนุกรม 25 ขา หรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์)

รูปที่ 2.25 คอนเน็กเตอร์อนุกรม

ตารางที่ 2.9 การจัดขาสัญญาณของพอร์ตอนุกรมในแบบต่างๆ และหน้าที่การทำงาน

คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Detect : DCD	อินพุต
2	3	Received Data : RxD	อินพุต
3	2	Transmitted Data : TxD	เอาต์พุต
4	20	Data Terminal Ready : DTR	เอาต์พุต
5	7	Single Ground : GND	-

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.9 (ต่อ) การจัดขาสัญญาณของพอร์ตอนุกรมในแบบต่างๆ และหน้าที่การทำงาน

คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
6	6	Data Set Ready : DSR	อินพุต
7	4	Request ToSend : RTS	เอาต์พุต
8	5	Clear To Send : CTS	อินพุต
9	22	Ring Indicator : RI	อินพุต

ขา Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับใช้งานปกติ ขานี้จะไม่ถูกนำมาใช้งานมากนัก

ขา Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยจะนำข้อมูลที่อ่านได้ไปเก็บไว้ในรีจิสเตอร์บัฟเฟอร์ (Register Buffer)

ขา Transmitted Data : TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์ โดยการนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

ขา Data Terminal Ready : DTR เป็นขาเอาต์พุตที่ใช้สำหรับส่งสัญญาณออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อกับอุปกรณ์ปลายทาง โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์และถ้าใช้การเชื่อมต่อแบบ 3 สาย ต้องเชื่อมต่อขา DTR และ DSR ของพอร์ตอนุกรมเข้าด้วยกัน และจะต้องต่อเชื่อมเข้ากับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

ขา Signal Ground : GND เป็นขากราวด์ของสัญญาณ

ขา Data Set Ready : DSR ขานี้จะใช้ควบคู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอก

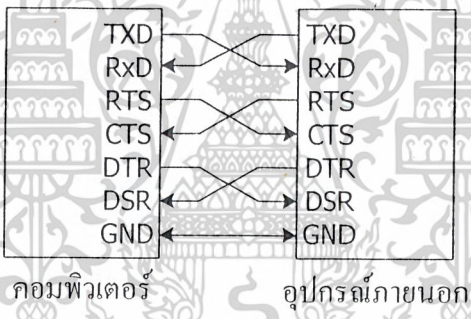
ขา Request To Send : RTS เป็นขาเอาต์พุตสำหรับส่งสัญญาณร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลมาให้คอมพิวเตอร์โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ซึ่งในกรณีที่มีการเชื่อมต่อแบบ 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS เข้าด้วยกัน เพื่อให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

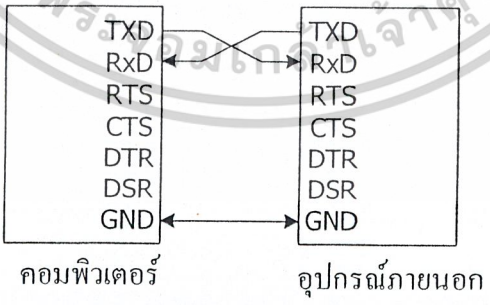
ขา Clear To Send : CTS เป็นขาอินพุตทำหน้าที่รอรับสัญญาณที่ส่งเข้ามา เมื่อมีการส่งสัญญาณเข้ามาที่ขานี้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ขานี้จะใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลแล้วหรือยัง

ขา Ring Indicator : RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มแล้วยังมีความต้องการตรวจสอบสัญญาณเรียกจากสายโทรศัพท์

สำหรับการเชื่อมต่อสายระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอกแสดงดังในรูปที่ 2.26 ซึ่งลูกศรในรูปจะแสดงถึงทิศทางของข้อมูล การเชื่อมต่อในรูปที่ 2.26 (ก) เป็นการเชื่อมต่อแบบ NULL MODEM หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็มส่วนการเชื่อมต่อในรูปที่ 2.26 (ข) เป็นการเชื่อมต่อโดยใช้สัญญาณน้อยที่สุดเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์



(ก) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ NULL MODEM



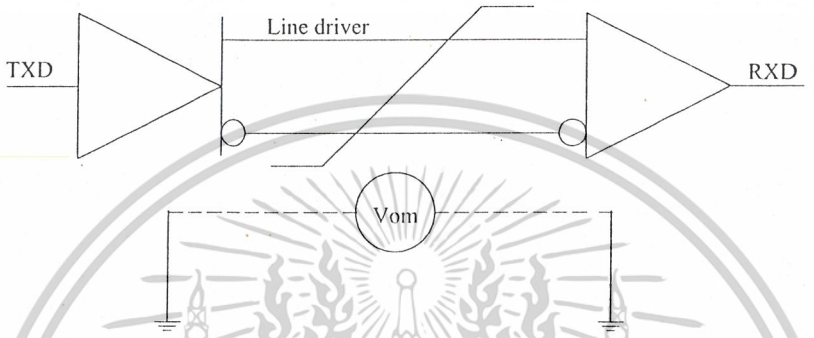
(ข) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ RS-232
ในลักษณะที่ใช้สายสัญญาณน้อยที่สุดเพียง 3 เส้น

รูปที่ 2.26 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในรูปแบบต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.2 RS-422

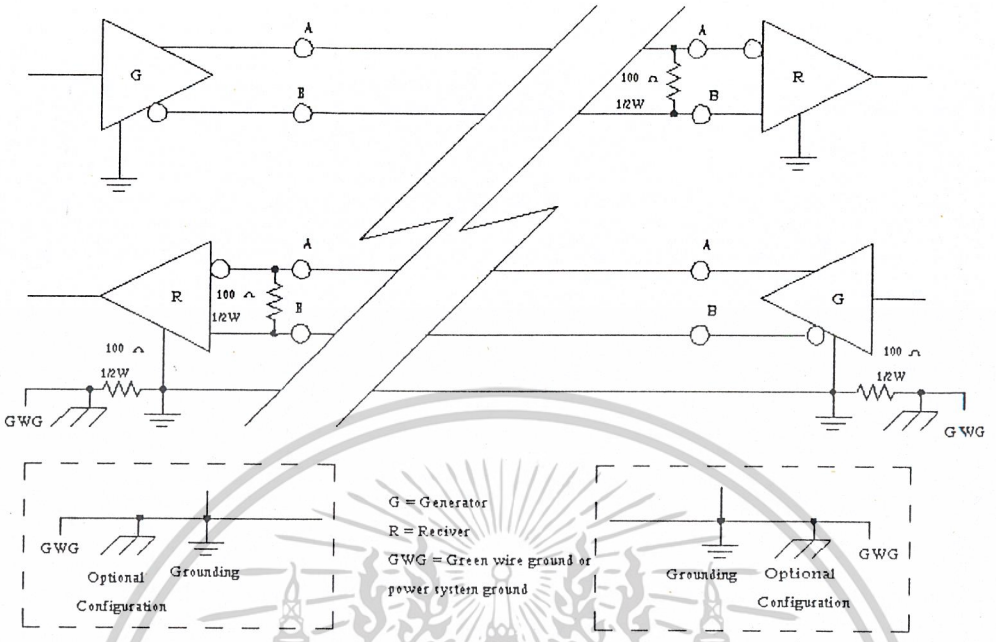
RS-422 ได้รับการพัฒนามาจากรูปแบบ RS-232 แต่ยังคงใช้งานลักษณะจุดต่อจุด (Point to point) เช่นเดิมอยู่โดยเป็นระบบการส่งสัญญาณที่เป็นแบบผลต่างระหว่างสาย 2 เส้น ที่บิดเกลียวซึ่งกันและกัน ดังรูปที่ 2.27



รูปที่ 2.27 ระบบการส่งสัญญาณที่เป็นแบบผลต่าง

การส่งสัญญาณแบบนี้จะไม่มีสายกราวด์ แบบ RS-232 แต่จะสลับแรงดันบวกและลบออกมาแทนค่า "1" และ "0" ยกตัวอย่าง ให้เห็นได้ชัดเจนดังนี้ คือ ถ้าสายพันเกลียว 1 คู่ ประกอบด้วยสาย A และ สาย B เมื่อมีการส่งสัญญาณ "1" ออกมา สาย A จะมีแรงดันบวกและสาย B จะมีแรงดันไฟลบออกมา ส่วนถ้าต้องการส่งสัญญาณ "0" สาย A จะมีค่าแรงดันไฟเป็นลบและสาย B จะมีค่าแรงดันเป็นบวก โดยการส่งสัญญาณในลักษณะนี้ จะคล้ายกับการส่งสัญญาณไฟฟ้ากระแสสลับแบบสมดุล ซึ่งจะทำให้สัญญาณรบกวนต่ำและสามารถส่งสัญญาณไปได้ไกลกว่าเดิมถึง 4,000 ฟุต และมีอัตราความเร็วการส่งข้อมูลสูงได้มากถึง 10 เมกะบิตต่อวินาที (Mbit/sec) ข้อดีที่เห็นได้ชัดคือ การส่งสัญญาณรูปแบบนี้ จะมีสัญญาณรบกวนต่ำมาก เนื่องจากสัญญาณรบกวนจะส่งเข้ามาในสายสัญญาณแบบเฟส (Phase) เดียวกัน เมื่อเข้ามายังอินพุตของวงจรขยายแบบผลต่าง แล้วก็หักล้างกันหมดไป ลักษณะการใช้งาน RS-422 จริงจะใช้สายคู่บิดเกลียว 2 ชุด โดยจะใช้การส่งสัญญาณชุดส่งไปและส่งกลับ แยกจากกันโดยเด็ดขาดซึ่งคล้ายกับรูปแบบ RS-232 เดิม การส่งแบบสองทิศทางสมบูรณ์สามารถต่อตัวพ่วงได้ถึง 10 ตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

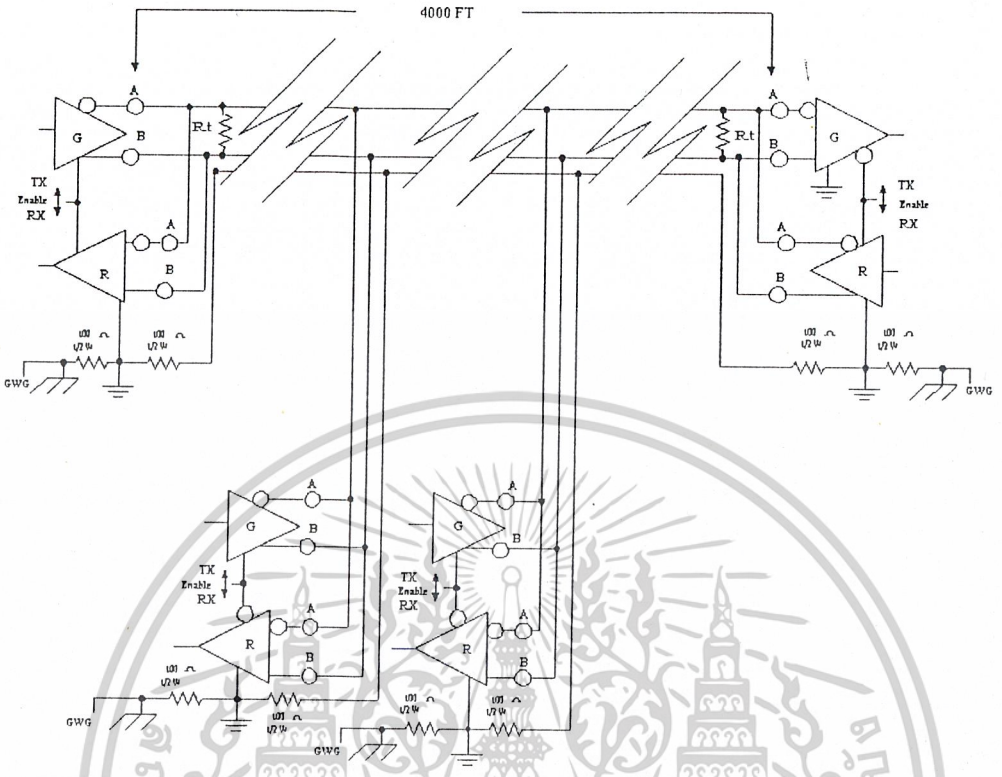


รูปที่ 2.28 การต่อสายสัญญาณแบบ RS-422 ใช้สาย 4 เส้น

2.5.3 RS-485

เป็นรูปแบบที่นำมาจากรูปแบบของ RS-422 ทุกประการ เพียงแต่มีการดัดแปลงให้ใช้สายสัญญาณเพียง 1 คู่สายบิดเกลียว โดยการใช้สายนี้เป็นทั้งสายส่งและสายรับในสายคู่เดียวกัน ซึ่งการที่จะทำงานในลักษณะนี้ได้นั้น ที่ขั้วของอุปกรณ์ทุกตัวจะต้องมีการต่อบัฟเฟอร์ที่สามารปรับให้เป็นความต้านสูงได้ คือตัดออกจากวงจรชั่วคราวได้นั่นเอง เพื่อใช้ในการบริหารสัญญาณ มักจะเป็นการสื่อสารในลักษณะ เครื่องลูกข่ายและเครื่องแม่ข่าย คือ มีเครื่องแม่ข่ายเป็นตัวหลักอยู่ 1 ตัว ซึ่งจะคอยควบคุมดูแลและสั่งการทั้งหมด ส่วนเครื่องลูกข่ายที่ต่ออยู่ทุกตัวก็จะรอรับการสั่งงานจากเครื่องแม่ข่ายเท่านั้นตลอดเวลา เครื่องลูกข่ายแต่ละตัวจะต้องมีชื่อเรียก เพื่อที่เครื่องแม่ข่ายจะใช้ในการสั่งงาน มีอีกวิธีการหนึ่ง ซึ่งได้ถูกพัฒนาขึ้นใหม่ คือ การสื่อสารกันโดยไม่มีเครื่องแม่ข่ายหลักคือทุกตัวในโครงข่ายเป็นเครื่องแม่ข่ายทั้งหมด แต่สามารถส่งข้อมูลด่วนออกมาควบคุมตัวอื่นได้โดยฉับพลัน จากการที่ได้ทำการทดลองได้ผลสถิติว่ารูปแบบหลังนี้สามารถลดการสื่อสารโดยไม่จำเป็นลงได้ถึง 30% ซึ่งขึ้นอยู่กับการพัฒนาซอฟต์แวร์ที่ซับซ้อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.



รูปที่ 2.29 การต่อสายสัญญาณแบบ RS-485 ใช้สาย 2 เส้น

ตารางที่ 2.10 ตารางเปรียบเทียบของรูปแบบ RS-232, RS-422 และRS-485

ข้อมูลของแต่ละรูปแบบ	RS-232	RS-422	RS-485
หมวดการทำงาน	Single Ended	Differential	Differential
จำนวนตัวลูกสูงสุด	1 Drive 1 Recp	1 Drive 10 Recp	1 Drive 32 Recp
ความยาวสูงสุดของสายเคเบิ้ล	50 ฟุต	4,000 ฟุต	4,000 ฟุต
อัตราความเร็วการส่งข้อมูล	20 Kbit/sec	10 Mbit/sec	10 Mbit/sec
แรงดันส่งออกสูงสุดเอาต์พุต	+/-25 V	-0.25V ถึง +6V	-7V ถึง +12V
ค่าความต้านทานโหลด (Load)	3 – 7 กิโลโอห์ม (KΩ)	100 Ω	54 Ω
ความไวในการรับ	+/-3 V	+/-200 มิลลิโวลต์ (mV)	+/-200 mV
ความต้านทานอินพุต	3 – 7 KΩ	มากกว่า 4 KΩ	มากว่า 12 KΩ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.4 รีจิสเตอร์ของ PIC ที่เกี่ยวกับการสื่อสารอนุกรม

เนื่องจากเราจะทำการส่งแบบอะซิงโครนัส (Asynchronous) ดังนั้นเราจะพิจารณา Register ต่างๆ ในกรณีแบบ Asynchronous

ตารางที่ 2.11 ตารางแสดง รีจิสเตอร์ที่เกี่ยวข้องกับการสื่อสารข้อมูลพอร์ตอนุกรม

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR BOR	Value on All other Reset
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000X	0000 000X
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

หมายเหตุ : x = ไม่ทราบสถานะ, - = ไม่ใช้งานแต่เมื่ออ่านค่ามาจะมีค่าเป็น “0”

1. PIR1 register บิตที่เกี่ยวข้องก็คือ RCIF และ TXIF

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
Bit 7							Bit 0

รูปที่ 2.30 ข้อมูลภายในของ รีจิสเตอร์ PIR1

เมื่อ RCIF = 1 หมายถึง บัฟเฟอร์ (Buffer) ที่ทำหน้าที่รับข้อมูลอนุกรมนั้นเต็ม

เมื่อ RCIF = 0 หมายถึง Buffer ที่ทำหน้าที่รับข้อมูลอนุกรมนั้นว่าง

เมื่อ TXIF = 1 หมายถึง Buffer ที่ทำหน้าที่ส่งข้อมูลอนุกรมนั้นเต็ม

เมื่อ TXIF = 0 หมายถึง Buffer ที่ทำหน้าที่ส่งข้อมูลอนุกรมนั้นว่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. RCSTA Register เป็น Register ที่เกี่ยวข้องกับการรับข้อมูล

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
Bit 7							Bit 0

รูปที่ 2.31 ข้อมูลภายในของรีจิสเตอร์ RCSTA

Bit 7 SPEN

1 = หมายถึง Serial port enable (อนุญาตให้ใช้งานพอร์ตอนุกรมได้)

0 = หมายถึง Serial port disable

Bit 6 RX9

1 = หมายถึง ขนาดข้อมูล 1 Bytes ที่รับมีขนาด 9 Bits

0 = หมายถึง ขนาดข้อมูล 1 Bytes ที่รับมีขนาด 8 Bits

Bit 5 SREN จะไม่สนใจ Bit นี้ในกรณีที่ใช้แบบ Asynchronous

Bit 4 CREN

1 = หมายถึง กำหนดให้มีการรับข้อมูลที่เข้ามาอย่างต่อเนื่อง

0 = หมายถึง ไม่ให้มีการรับข้อมูลที่เข้ามาอย่างต่อเนื่อง

Bit 3 ADDEN เนื่องจากส่งข้อมูลแบบ 8 Bit ดังนั้นจะไม่สนใจ บิตนี้

Bit 2 FERR

1 = หมายถึง มี Frame error เกิดขึ้นกับข้อมูลที่ได้รับ

0 = หมายถึง ไม่มี Frame error เกิดขึ้นกับข้อมูลที่ได้รับ

Bit 1 OERR

1 = หมายถึง มี Overrun error เกิดขึ้นกับข้อมูลที่ได้รับ

0 = หมายถึง ไม่มี Overrun error เกิดขึ้นกับข้อมูลที่ได้รับ

Bit 0 RX9D

เป็นบิตที่ 9 ของข้อมูลในกรณีที่ข้อมูลเป็นแบบ 9 Bits

3. RCREG Register คือข้อมูลขนาด 1 Byte ที่รับได้จากภายนอกในขณะนั้น

4. TXSTA Register เป็น Register ที่เกี่ยวข้องกับการส่งข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. SPBRG Register คือ Register ที่กำหนดค่าที่ Baud rate

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D
Bit 7							Bit 0

รูปที่ 2.32 ข้อมูลภายในของรีจิสเตอร์ TXSTA

Bit 7 CSRC จะไม่สนใจ Bit นี้ในกรณีที่ใช้แบบ Asynchronous

Bit 6 TX9

1 = หมายถึง กำหนดให้เป็นการส่งแบบ 9 Bit

0 = หมายถึง กำหนดให้เป็นการส่งแบบ 8 Bit

Bit 5 TXEN

1 = หมายถึง อนุญาตให้มีการส่งข้อมูล

0 = หมายถึง ไม่อนุญาตให้มีการส่งข้อมูล

Bit 4 SYNC

1 = หมายถึง เป็นการส่งแบบ Synchronous mode

0 = หมายถึง เป็นการส่งแบบ Asynchronous mode

Bit 3 ไม่มีการใช้งาน

Bit 2 BRGH ใน Asynchronous mode หมายถึง

1 = หมายถึง เป็นแบบ High speed

0 = หมายถึง เป็นแบบ Low speed

Bit1 TRMT หมายถึง Bit ที่แสดงสถานะของ Transmit Shift Register

1 = หมายถึง เป็น TSR ว่าง

0 = หมายถึง เป็น TSR เต็ม

Bit 0 TX9D

เป็นข้อมูล Bit ที่ 9 ถ้าจะทำการส่งแบบ 9 Bit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. PIE1 บิตที่เกี่ยวข้องคือ RCIE, TXIE

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
Bit 7							Bit 0

รูปที่ 2.33 ข้อมูลภายในของรีจิสเตอร์ PIE1

Bit 5 RCE

1 = หมายถึง กำหนดให้มีอินเตอร์รัพต์ (Interrupt) เกิดขึ้นได้เมื่อทำการรับข้อมูลทางพอร์ตอนุกรม เสร็จ 1 Byte

0 = หมายถึง กำหนดไม่ให้มีอินเตอร์รัพต์เกิดขึ้นได้ เมื่อทำการรับข้อมูลทางพอร์ตอนุกรม เสร็จ 1 Byte

Bit4 TXIE

1 = หมายถึง กำหนดให้มีอินเตอร์รัพต์เกิดขึ้นเมื่อทำการรับข้อมูลทางพอร์ตอนุกรม เสร็จ 1 Byte

0 = หมายถึง กำหนดไม่ให้มีอินเตอร์รัพต์เกิดขึ้นได้ เมื่อทำการรับข้อมูลทางพอร์ตอนุกรม เสร็จ 1 Byte

การคำนวณหาค่าบอดเรต (Baud Rate) สูตรที่ใช้คำนวณหา ค่า Baud Rate ขึ้นกับอยู่บิต 2 บิต คือ SYNC กับ BRGH ส่วนที่ว่าเลือกบิตให้ใช้สูตรคำนวณหา Baud Rate เป็นอย่างไร ก็ดูส่วนที่จะใช้คือ SYNC = 0, BRGH = 0 ดังนั้นจะใช้สูตร

Baud Rate = Fosc/(64(X+1))

ตารางที่ 2.12 ตารางแสดงการคำนวณหาค่า Baud rate

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64(X+1))	Baud Rate = Fosc/(16(X+1))
1	(Synchronous) Baud Rate = Fosc/(4(X+1))	NA

หมายเหตุ : X = value in SPBRG (0 to 255)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

การอินเตอร์รัพต์ คือการขัดจังหวะการทำงานปกติของซีพียู เมื่อเกิดการอินเตอร์รัพต์ขึ้น ซีพียูจะทำงานจนจบไซเกิลนั้น แล้วจึงมาตอบสนองการอินเตอร์รัพต์โดยกระโดดมาทำงานยังตำแหน่งที่กำหนดไว้ จนกระทั่งเมื่อตอบสนองการอินเตอร์รัพต์เรียบร้อยแล้ว ซีพียูจะกลับไปทำงานต่อจากเหตุการณ์ที่เกิดขึ้นก่อนหน้าการอินเตอร์รัพต์

7. INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IE	INTE	RBIF
Bit 7							Bit 0

รูปที่ 2.34 ข้อมูลภายในของรีจิสเตอร์ INTCON

Bit 7 GIE

1 = Enables การอินเตอร์รัพต์ทั้งหมด

0 = Disables การอินเตอร์รัพต์ทั้งหมด

Bit 6 PEIE

1 = Enables การอินเตอร์รัพต์ภายนอกทั้งหมด

0 = Disables การอินเตอร์รัพต์ภายนอกทั้งหมด

Bit 5 TMR0IE

1 = Enables การอินเตอร์รัพต์จาก TMR0

0 = Disables การอินเตอร์รัพต์จาก TMR0

Bit 4 INTE

1 = Enables การอินเตอร์รัพต์ภายจากRB0/INT

0 = Disables การอินเตอร์รัพต์ภายจากRB0/INT

Bit 3 RBIE

1 = Enables การอินเตอร์รัพต์ภายจาก พอร์ต (Port) B

0 = Disables การอินเตอร์รัพต์ภายจาก พอร์ต B

Bit 2 TMR0IF

1 = TMR0 เกิดการโอเวอร์โฟลว (Overflow)

0 = TMR0 ไม่เกิดการโอเวอร์โฟลว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Bit 1 INTF

1 = มีการอินเตอร์รัพต์จากภายนอกเกิดขึ้น

0 = ไม่มีการอินเตอร์รัพต์จากภายนอกเกิดขึ้น

Bit 0 RBIF

1 = การเปลี่ยนแปลงขึ้นที่ขาใดขาหนึ่งของบิต 4-7 ของพอร์ต B

0 = ไม่มีการเปลี่ยนแปลงขึ้นที่ขาใดขาหนึ่งของบิต 4-7 ของพอร์ต B

8. PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
Bit7							Bit0

รูปที่ 2.35 ข้อมูลภายในของรีจิสเตอร์ PIE1

Bit 7 PSPIE

1 = Enables การอินเตอร์รัพต์จากการอ่าน/เขียน PSP

0 = Disables การอินเตอร์รัพต์จากการอ่าน/เขียน PSP

Bit 6 ADIE

1 = Enables การอินเตอร์รัพต์จาก A/D converter

0 = Disables การอินเตอร์รัพต์จาก A/D converter

Bit 5 RCIE

1 = Enables การอินเตอร์รัพต์จากการรับข้อมูลจากพอร์ตอนุกรมแบบ Asynchronous

0 = Disables การอินเตอร์รัพต์จากการรับข้อมูลจากพอร์ตอนุกรมแบบ Asynchronous

Bit 4 TXIE

1 = Enables การอินเตอร์รัพต์จากการส่งข้อมูลจากพอร์ตอนุกรมแบบ Asynchronous

0 = Disables การอินเตอร์รัพต์จากการส่งข้อมูลจากพอร์ตอนุกรมแบบ Asynchronous

Bit 3 SSPIE

1 = Enables การอินเตอร์รัพต์จากพอร์ตอนุกรมแบบ Synchronous

0 = Disables การอินเตอร์รัพต์จากพอร์ตอนุกรมแบบ Synchronous

Bit 2 CCP1IE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1 = Enables การอินเตอร์รัพต์จาก CCP1

0 = Disables การอินเตอร์รัพต์จาก CCP1

Bit 1 TMR2IE

1 = Enables การอินเตอร์รัพต์จาก TMR2

0 = Disables การอินเตอร์รัพต์จาก TMR2

Bit 0 TMR1IE

1 = Enables การอินเตอร์รัพต์จาก TMR1

0 = Disables การอินเตอร์รัพต์จาก TMR1

2.6 การสร้างชุดสาธิต

ในการสร้างชุดสาธิตมาตรฐาน USB โดยPIC 16F877A ได้ดำเนินการสร้างดังนี้

1. ศึกษาทฤษฎีและหลักการทำงานของพอร์ต USB มาตรฐานของ RS-232, RS-422 และ RS-485 การเขียนโปรแกรมและการใช้งานไมโครคอลโทรลเลอร์ PIC 16F877A จากเอกสาร ตำรา และผลงานวิจัยที่เกี่ยวข้อง โดยขอคำแนะนำจากอาจารย์ที่ปรึกษา

2. ออกแบบชุดสาธิต ได้แบ่งออกเป็นส่วนต่างๆ ดังนี้

2.1 ส่วนติดต่อพอร์ต USB

2.2 ส่วนควบคุม

2.3 ส่วนแปลงข้อมูล RS-232, RS-422 และRS-485

2.4 ส่วนโปรแกรมการทำงาน

3. เมื่อได้รูปแบบวงจรและโปรแกรมของชุดสาธิตเรียบร้อยก็ดำเนินการสร้างชุดสาธิต

4. สร้างใบงานการทดลองและแบบฝึกหัด

5. ตรวจสอบความถูกต้องโดยส่งให้อาจารย์ที่ปรึกษาตรวจสอบอีกครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

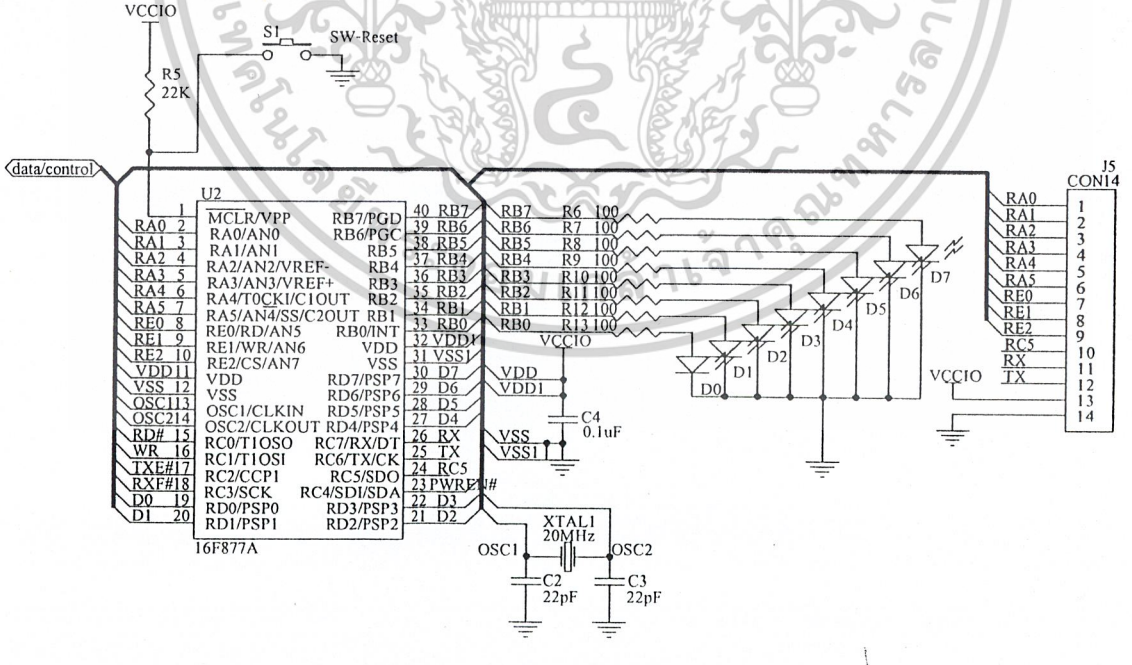
บทที่ 3
การออกแบบ การสร้าง และการทำงาน

จากหลักการพื้นฐาน และทฤษฎีต่างๆ ที่เกี่ยวข้องกับพอร์ตUSB, RS-232, RS-422, RS-485 และไมโครคอนโทรลเลอร์ PIC 16F877A สามารถทำการออกแบบ และสร้างวงจรการทำงานของชุดสาธิตมาตรฐาน USB โดย PIC 16F877A โดยมีขั้นตอนการออกแบบ การสร้าง และการทำงานได้ดังต่อไปนี้

3.1 ส่วนควบคุมการทำงานและแสดงผล

3.1.1 การออกแบบและการสร้าง

ส่วนควบคุมการทำงานและแสดงผล นั้นประกอบด้วยไมโครคอนโทรลเลอร์ PIC16F877A เป็นส่วนที่ทำหน้าควบคุมการทำงานของส่วนต่างๆในชุดสาธิต โดยจะต่ออยู่กับ LED เพื่อใช้ในการแสดงผล ตามวงจรดังรูปที่ 3.1



รูปที่ 3.1 วงจรควบคุมการทำงานหลัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.2 การทำงาน

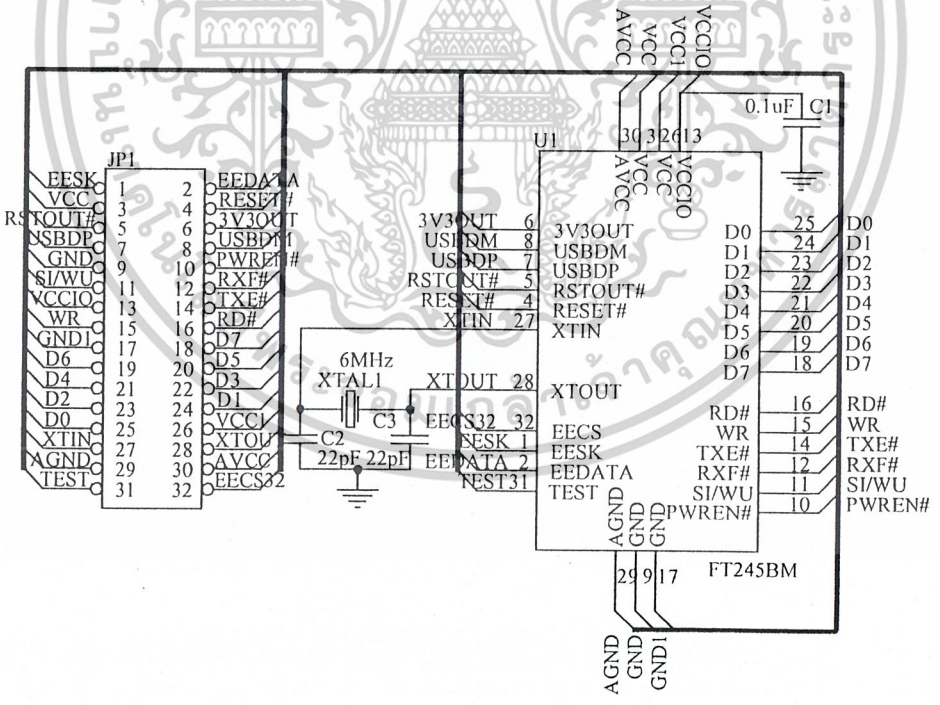
ในส่วนนี้จะมีไมโครคอนโทรลเลอร์ PIC 16F877A ซึ่งต่ออยู่กับส่วนต่างๆ ของชุดทดลอง คือ ส่วนติดต่อ USB ,ส่วน RS-232 ,ส่วน RS-422 และส่วน RS-485 เพื่อที่จะควบคุมการรับและส่งข้อมูลจากส่วนต่างๆ

3.2 ส่วนติดต่อ USB

3.2.1 การออกแบบและการสร้าง

ส่วนติดต่อ USB นั้นประกอบด้วยชิป FT245BM เป็นส่วนที่ทำหน้าติดกับพอร์ต USB และการรับหรือส่งข้อมูลนั้นจะถูกควบคุมด้วยส่วนควบคุมการทำงาน โดยในการออกแบบนั้นได้แบ่งในส่วนนี้เป็น 2 ส่วน คือ USB โมดูล และวงจรติดต่อ USB

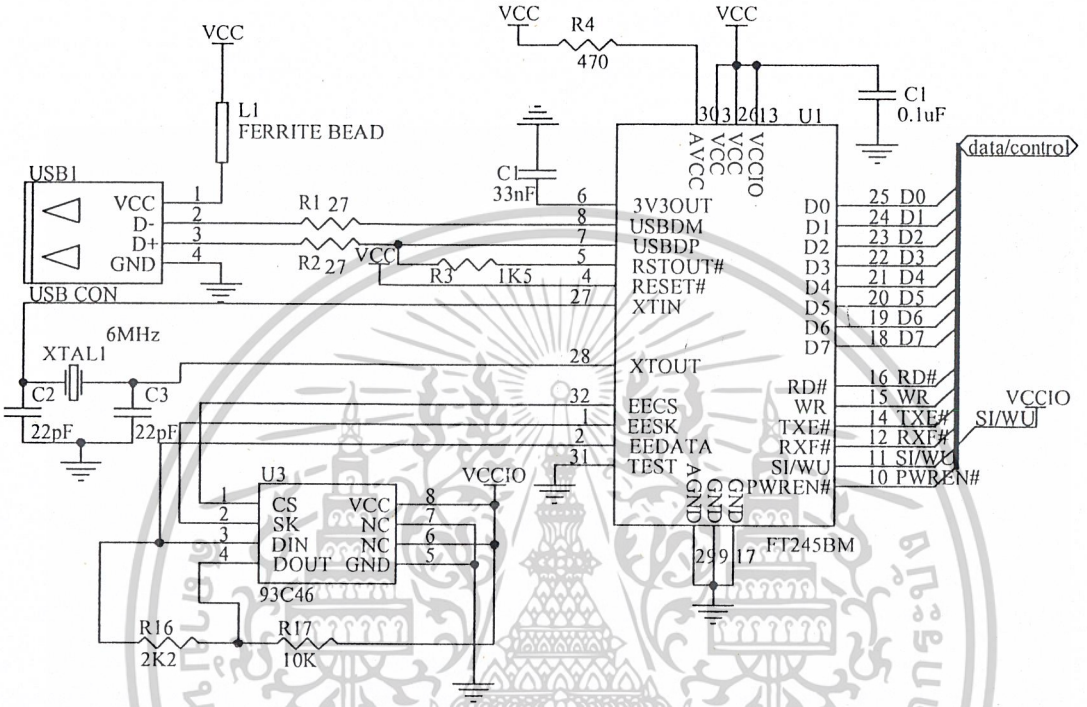
1. ส่วน USB Module ได้ออกแบบให้เป็นแบบโมดูลเพื่อสะดวกต่อการประยุกต์ใช้งาน ตามวงจรดังรูปที่ 3.2



รูปที่ 3.2 วงจรUSB โมดูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ส่วนวงจรติดต่อ USB ในส่วนออกแบบเพื่อเชื่อมต่อใช้งานกับส่วน USB โมดูล และมี EEPROM ซึ่งทำหน้าที่เก็บข้อมูลเกี่ยวกับอุปกรณ์ ดังรูปที่ 3.3



รูปที่ 3.3 วงจรติดต่อ USB

3.2.2 การทำงาน

เมื่อมีการเชื่อมต่อเข้ากับพอร์ต USB ชิป FT245BMจะทำการอ่านข้อมูลที่อยู่ใน EEPROM และส่งข้อมูลต่างๆของอุปกรณ์ไปยังคอมพิวเตอร์ เพื่อให้สามารถติดต่อรับและส่งข้อมูลกับคอมพิวเตอร์ได้

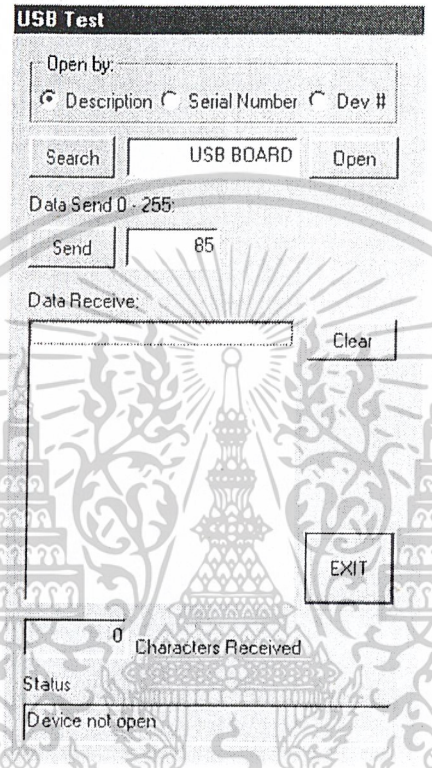
3.3 ส่วนโปรแกรมการทำงาน

3.3.1 โปรแกรมรับและส่งข้อมูล

โปรแกรมรับและส่งข้อมูล มีอยู่ 2 โปรแกรม ได้แก่ โปรแกรม USB Test และโปรแกรม Serial Test

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

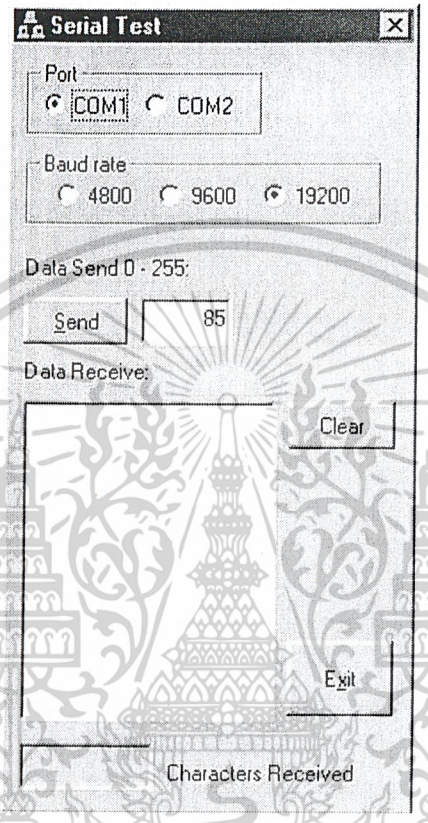
1. โปรแกรม USB Test เป็นโปรแกรมที่ใช้ในการรับและส่งข้อมูลผ่านพอร์ต USB ซึ่งเขียนด้วยโปรแกรม Visual C++ โดยมีการเรียกใช้โปรแกรมย่อยจากไฟล์ FTD2XX.dll เพื่อง่ายต่อการเขียนโปรแกรม



รูปที่ 3.4 โปรแกรม USB Test

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

2. โปรแกรม Serial Test เป็นโปรแกรมที่ใช้ในการรับและส่งข้อมูลผ่านพอร์ต RS-232 ซึ่งเขียนด้วยโปรแกรม Visual Basic

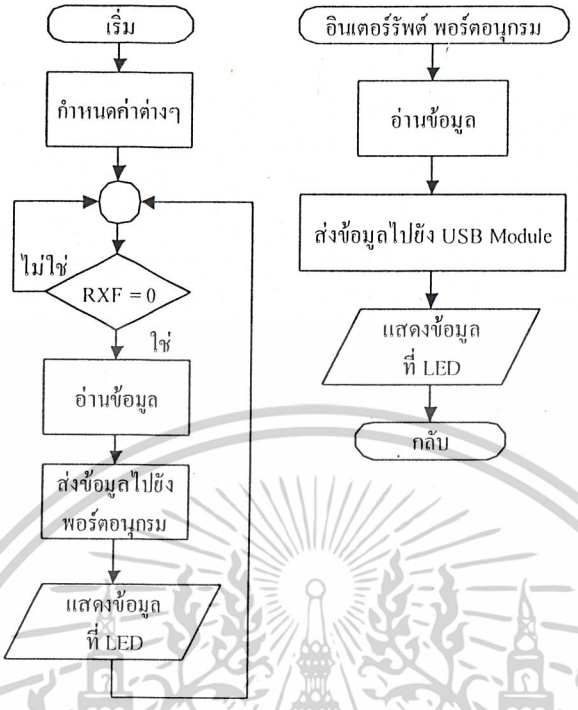


รูปที่ 3.5 โปรแกรม Serial Test

3.3.2 โปรแกรมควบคุมฮาร์ดแวร์

โปรแกรมควบคุมฮาร์ดแวร์ เขียนด้วยภาษา C for PIC โดยโปรแกรม PCW ซึ่งเป็นโปรแกรมที่ควบคุมให้ PIC16F877A ทำหน้าที่คอยรับและส่งข้อมูลจากพอร์ต USB และพอร์ตอนุกรม ซึ่งมีการทำงานดังรูปที่ 3.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .



รูปที่ 3.6 ผังงานโปรแกรมควบคุมฮาร์ดแวร์

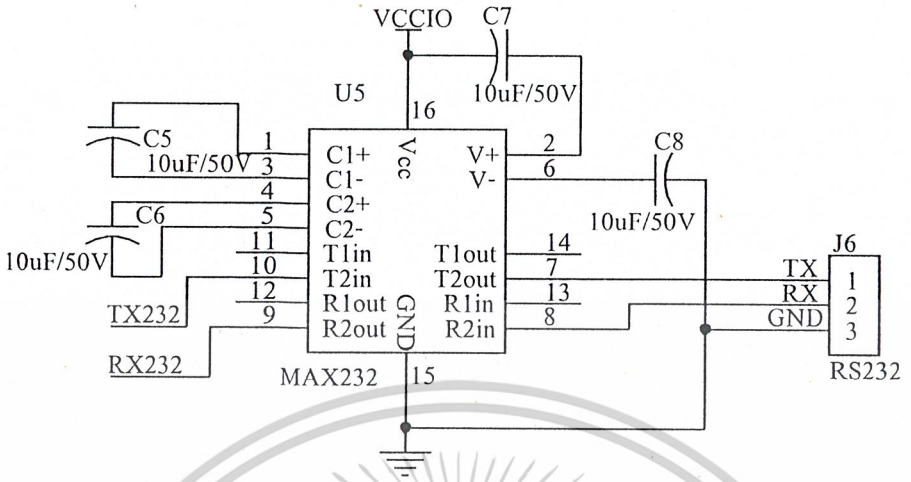
3.4 การสร้างชุดสาธิต

3.4.1 การออกแบบและการสร้าง

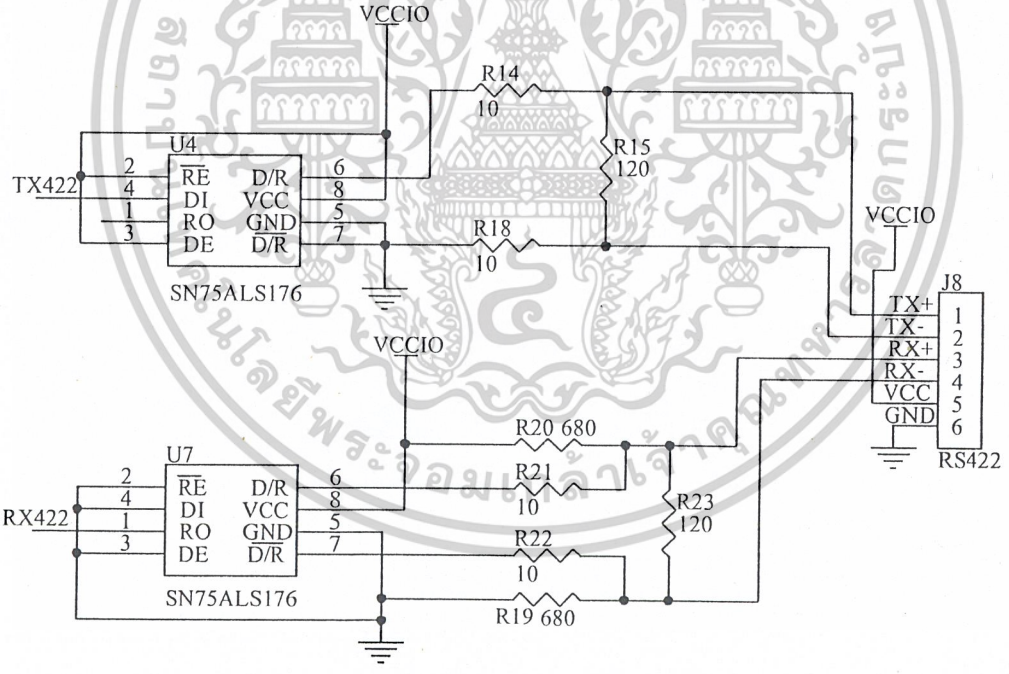
ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A นั้นประกอบด้วยชุดวงจรพิมพ์ทั้งหมด 3 วงจร ได้แก่ ชุดวงจรหลัก ชุดRS-422 และชุดRS-485

1. ชุดวงจรหลัก ประกอบด้วยวงจรต่างๆได้แก่ วงจรควบคุมการทำงานและแสดงผล วงจรติดต่อ USB วงจรแปลงRS-232 วงจรแปลงRS-422 วงจรแปลงRS-485 และวงจรแหล่งจ่ายไฟฟ้า ซึ่งวงจรควบคุมการทำงานและแสดงผล และวงจรติดต่อ USB ได้กล่าวมาแล้วในหัวข้อที่ 3.1 และ 3.2 ส่วนวงจรแปลงRS-232 วงจรแปลงRS-422 วงจรแปลงRS-485 และวงจรแหล่งจ่ายไฟฟ้า มีวงจรดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

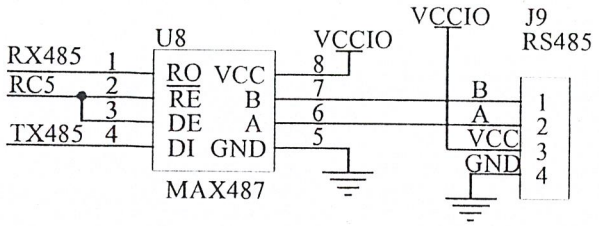


รูปที่ 3.7 วงจรแปลง RS-232

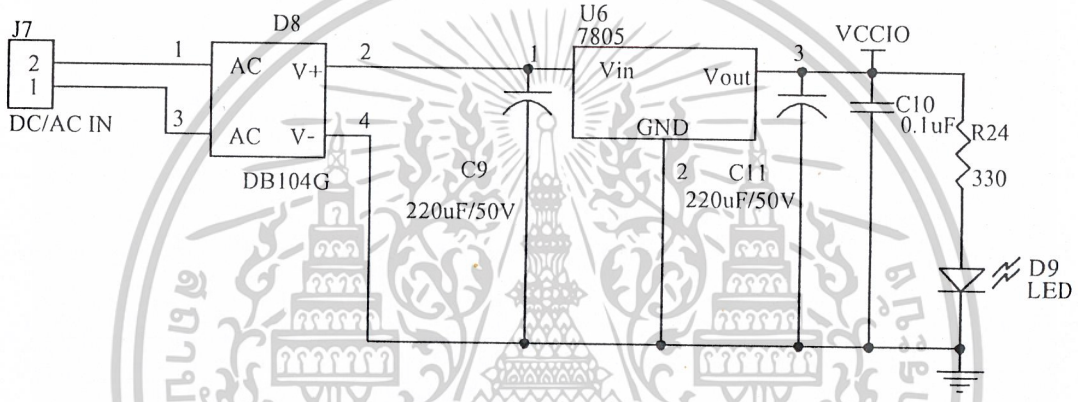


รูปที่ 3.8 วงจรแปลง RS-422

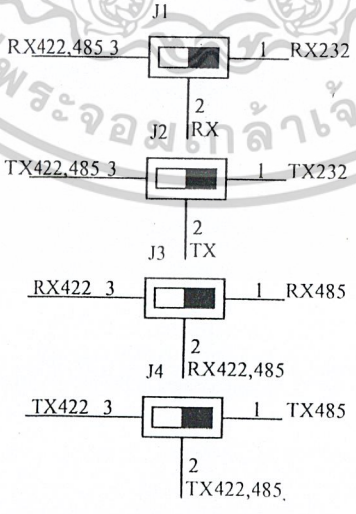
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.9 วงจรแปลง RS-485



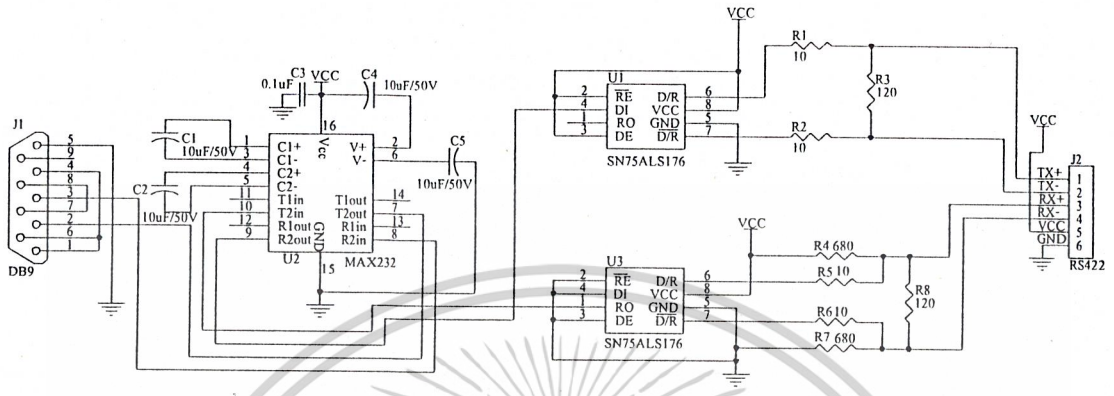
รูปที่ 3.10 วงจรแหล่งจ่ายไฟฟ้า



รูปที่ 3.11 จัมเปอร์เลือกใช้ RS-232, RS-422 และRS-485

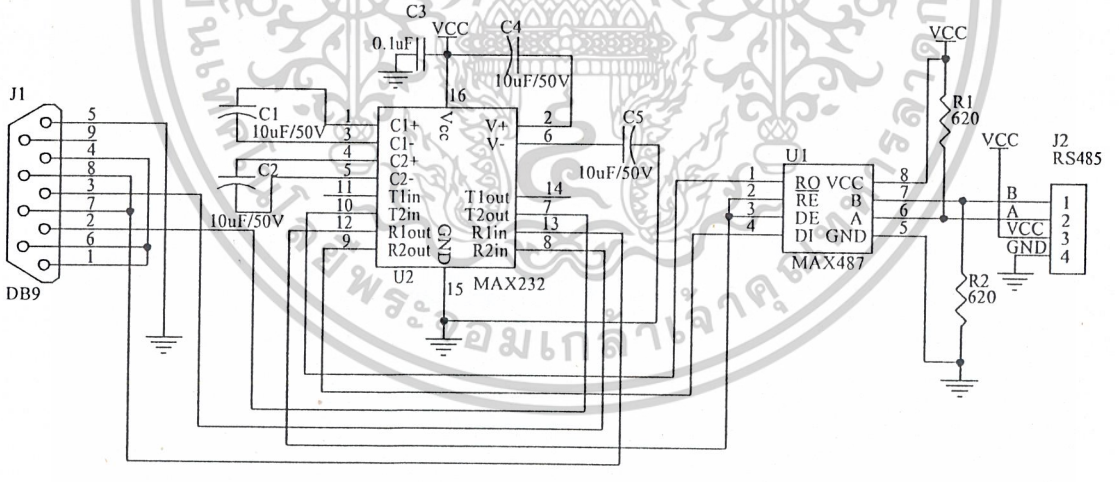
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ชุดRS-422 เป็นวงจรที่ทำหน้าที่แปลงข้อมูลจาก RS-232 เป็น RS-422



รูปที่ 3.12 วงจรแปลง RS-232 เป็น RS-422

3. ชุดRS-485 เป็นวงจรที่ทำหน้าที่แปลงข้อมูลจาก RS-232 เป็น RS-485



รูปที่ 3.13 วงจรแปลง RS-232 เป็น RS-485

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.2 การทำงาน

1. ชุดวงจรหลัก ซึ่งประกอบด้วย วงจรควบคุมการทำงานและแสดงผล วงจรติดต่อ USB วงจรแปลงRS-232 วงจรแปลงRS-422 วงจรแปลงRS-485 และจะมีจัมเปอร์เพื่อเลือกวงจรแปลง RS-232, RS-422 และRS-485 ตามแต่ละแบบการใช้งาน

2. ชุดRS-422 จากรูปที่ 3.18 วงจรจะแปลงจากRS-232 เป็นระดับ TTL โดย IC MAX232 แล้วจะทำการแปลงเป็นRS-422 ด้วย IC SN75176 2 ตัวโดยแยก RX และTX ออกจากกัน ซึ่งเป็นการใช้สาย 4 เส้น TX-, TX+, RX- และRX+

3. ชุดRS-485 จากรูปที่ 3.22 วงจรจะแปลงจาก RS-232 เป็นระดับ TTL โดย IC MAX232 แล้วจะทำการแปลงเป็น RS-485 ด้วย IC MAX487 ซึ่งเป็นการใช้สาย 2 เส้น A และB



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

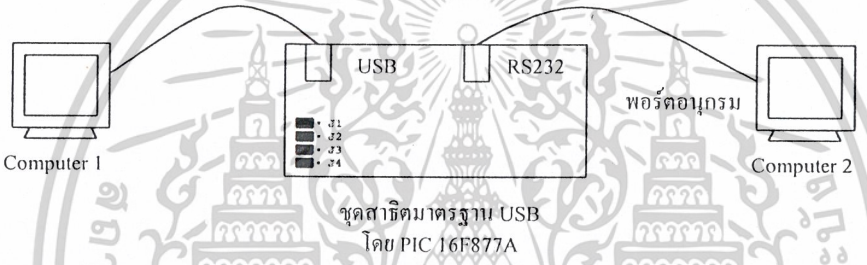
บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-232

4.1.1 การทดลอง

เป็นการทดลองส่งข้อมูลจากพอร์ต USB เป็น RS-232 โดยมีการเชื่อมต่อชุดสาธิต ดังรูปที่ 4.1



รูปที่ 4.1 การเชื่อมต่อชุดสาธิตเพื่อทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-232

ส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 1 โดยใช้โปรแกรม USB Test ทำการส่งข้อมูลตามตารางที่ 4.1 แล้วทำการอ่านข้อมูลจากโปรแกรม Serial Test ของคอมพิวเตอร์เครื่องที่ 2

ตารางที่ 4.1 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 1)
0
15
85
170
255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.2 ผลการทดลอง

ตารางที่ 4.2 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 1)	ข้อมูลที่รับ (จากเครื่องที่ 2)
0	0
15	15
85	85
170	170
255	255

4.2 การทดลองการแปลงข้อมูลจาก RS-232 เป็นพอร์ต USB

4.2.1 การทดลอง

เป็นการทดลองส่งข้อมูลจาก RS-232 เป็น พอร์ต USB โดยมีการเชื่อมต่อชุดสาธิตเช่นเดียวกับการทดลองในหัวข้อที่ 4.1

ส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 2 โดยใช้โปรแกรม Serial Test ทำการส่งข้อมูลตามตารางที่ 4.3 แล้วทำการอ่านข้อมูลจากโปรแกรม USB Test ของคอมพิวเตอร์เครื่องที่ 1

ตารางที่ 4.3 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 2)
0
15
85
170
255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2 ผลการทดลอง

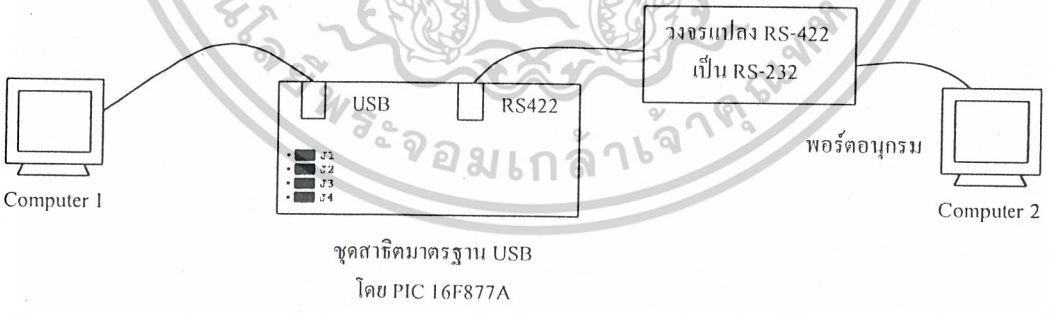
ตารางที่ 4.4 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 2)	ข้อมูลที่รับ (จากเครื่องที่ 1)
0	0
15	15
85	85
170	170
255	255

4.3 การทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-422

4.3.1 การทดลอง

เป็นการทดลองส่งข้อมูลจากพอร์ต USB เป็น RS-422 โดยมีการเชื่อมต่อชุดสาธิต ดังรูปที่ 4.2



รูปที่ 4.2 การเชื่อมต่อชุดสาธิตเพื่อทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-422

ส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 1 โดยใช้โปรแกรม USB Test ทำการส่งข้อมูลตาม ตารางที่ 4.5 แล้วทำการอ่านข้อมูลจากโปรแกรม Serial Test ของคอมพิวเตอร์เครื่องที่ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

ตารางที่ 4.5 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 1)
0
15
85
170
255

4.3.2 ผลการทดลอง

ตารางที่ 4.6 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 1)	ข้อมูลที่รับ (จากเครื่องที่ 2)
0	0
15	15
85	85
170	170
255	255

4.4 การทดลองการแปลงข้อมูลจาก RS-422 เป็นพอร์ต USB

4.4.1 การทดลอง

เป็นการทดลองส่งข้อมูลจาก RS-422 เป็น พอร์ต USB โดยมีการเชื่อมต่อชุดสาธิตเช่นเดียวกับการทดลองในหัวข้อที่ 4.3

ส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 2 โดยใช้โปรแกรม Serial Test ทำการส่งข้อมูลตามตารางที่ 4.7 แล้วทำการอ่านข้อมูลจากโปรแกรม USB Test ของคอมพิวเตอร์เครื่องที่ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.7 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 2)
0
15
85
170
255

4.4.2 ผลการทดลอง

ตารางที่ 4.8 ข้อมูลในการส่งทดลอง

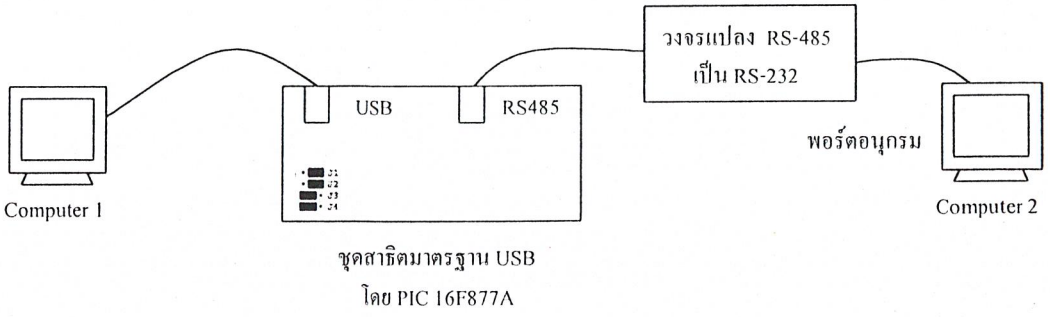
ข้อมูลที่ส่ง (จากเครื่องที่ 2)	ข้อมูลที่รับ (จากเครื่องที่ 1)
0	0
15	15
85	85
170	170
255	255

4.5 การทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-485

4.5.1 การทดลอง

เป็นการทดลองส่งข้อมูลจากพอร์ต USB เป็น RS-485 โดยมีการเชื่อมต่อชุดสาธิต ดังรูปที่ 4.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 การเชื่อมต่อชุดสาธิตเพื่อทดลองการแปลงข้อมูลจากพอร์ต USB เป็น RS-485

ส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 1 โดยใช้โปรแกรม USB Test ทำการส่งข้อมูลตามตารางที่ 4.9 แล้วทำการอ่านข้อมูลจากโปรแกรม Serial Test ของคอมพิวเตอร์เครื่องที่ 2

ตารางที่ 4.9 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 1)
0
15
85
170
255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5.2 ผลการทดลอง

ตารางที่ 4.10 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 1)	ข้อมูลที่รับ (จากเครื่องที่ 2)
0	0
15	15
85	85
170	170
255	255

4.6 การทดลองการแปลงข้อมูลจาก RS-485 เป็นพอร์ต USB

4.6.1 การทดลอง

เป็นการทดลองส่งข้อมูลจาก RS-485 เป็น พอร์ต USB โดยมีการเชื่อมต่อชุดสาธิตเช่นเดียวกับการทดลองในหัวข้อที่ 4.5

ส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 2 โดยใช้โปรแกรม Serial Test ทำการส่งข้อมูลตามตารางที่ 4.11 แล้วทำการอ่านข้อมูลจากโปรแกรม USB Test ของคอมพิวเตอร์เครื่องที่ 1

ตารางที่ 4.11 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 2)
0
15
85
170
255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.6.2 ผลการทดลอง

ตารางที่ 4.12 ข้อมูลในการส่งทดลอง

ข้อมูลที่ส่ง (จากเครื่องที่ 2)	ข้อมูลที่รับ (จากเครื่องที่ 1)
0	0
15	15
85	85
170	170
255	255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
บทสรุป

5.1 สรุป

ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A เป็นชุดสาธิตที่นำเสนอให้เห็นถึงหลักการใช้งานพอร์ต USB โดยสามารถที่จะรับหรือส่งข้อมูลจากพอร์ต USB เพื่อนำไปใช้ในการควบคุมหรือใช้งานต่างๆ โดยชุดสาธิตแยกออกเป็น 3 วงจรหลักได้แก่ ชุดวงจรหลัก ชุดRS-422 และชุดRS-485 ชุดวงจรหลักซึ่งประกอบด้วยไมโครคอนโทรลเลอร์ PIC16F877A เป็นตัวควบคุมการทำงานทั้งหมดทุกส่วนของชุดสาธิต ชุดRS-422 นั้นมีหน้าที่แปลงข้อมูลจากRS-422 เป็น RS-232 โดยใช้สายในการรับและส่งจำนวน 4 เส้น และชุดRS-485 มีหน้าที่แปลงข้อมูลจากRS-485 เป็น RS-232 โดยใช้สายในการรับและส่งจำนวน 2 เส้น

5.2 ปัญหา และแนวทางแก้ไข

จากการดำเนินการสร้างและทดสอบโครงงานพบว่ามีปัญหาเกิดขึ้นหลายประการ ซึ่งสรุปได้ดังนี้

1. ปัญหา อุปกรณ์ไม่ทำงานเนื่องจากการตอบสนองไม่ทันต่อโปรแกรมการทำงาน

แนวทางการแก้ไข ต้องเขียนโปรแกรมหน่วงเวลาเพื่อให้อุปกรณ์ทำงานได้ทัน

2. ปัญหา การออกแบบและสร้างชุดสาธิตในครั้งแรก ลายเส้นของแผ่นวงจรพิมพ์ที่ทำมีขนาดเล็กมากและอาจทำให้มีปัญหาด้านกระแสไฟฟ้า

แนวทางการแก้ไข ดำเนินการออกแบบแผ่นวงจรพิมพ์ใหม่ ให้มีขนาดใหญ่กว่าเดิม

5.3 แนวทางการพัฒนา

1. เพิ่มวงจรอินเตอร์เฟสให้กับไมโครคอนโทรลเลอร์มากขึ้นเพื่อใช้ในการศึกษาประยุกต์ใช้งานต่างๆ

2. สามารถเปลี่ยนจากไมโครคอนโทรลเลอร์ PIC16F877A เป็นไมโครคอนโทรลเลอร์เบอร์อื่นๆหรืออุปกรณ์ประเภท FPGA (Field Programmable Gate Array)

3. พัฒนาโปรแกรมรับและส่งข้อมูล ให้มีความสามารถมากขึ้นเพื่อใช้ในงานควบคุมหรืองานอื่นๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

บรรณานุกรม

กฤษดา ใจเย็น และชัยวัฒน์ ลิ้มพรจิตรวิไล. **PIC16F84 THEORY & PRACTICAL APPROACH.** กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์. 2543

ชัยวัฒน์ ลิ้มพรจิตรวิไล และวรพจน์ กรแก้ววัฒนกุล. **เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51.** กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์. 2543

ลภน สุภาพ และคณะ. **เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ต USB ขั้นพื้นฐาน.** กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์. ม.ป.ป.

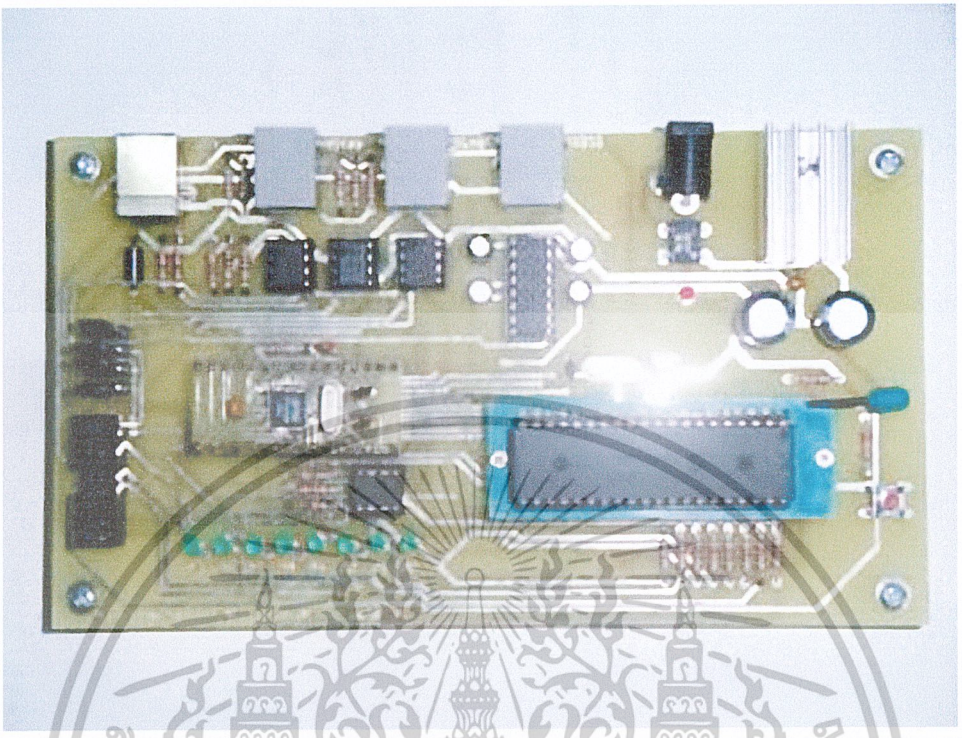
สมชาย รัตนเลิศนุสรณ์. **การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี.** พิมพ์ครั้งที่ 2. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี. 2545

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

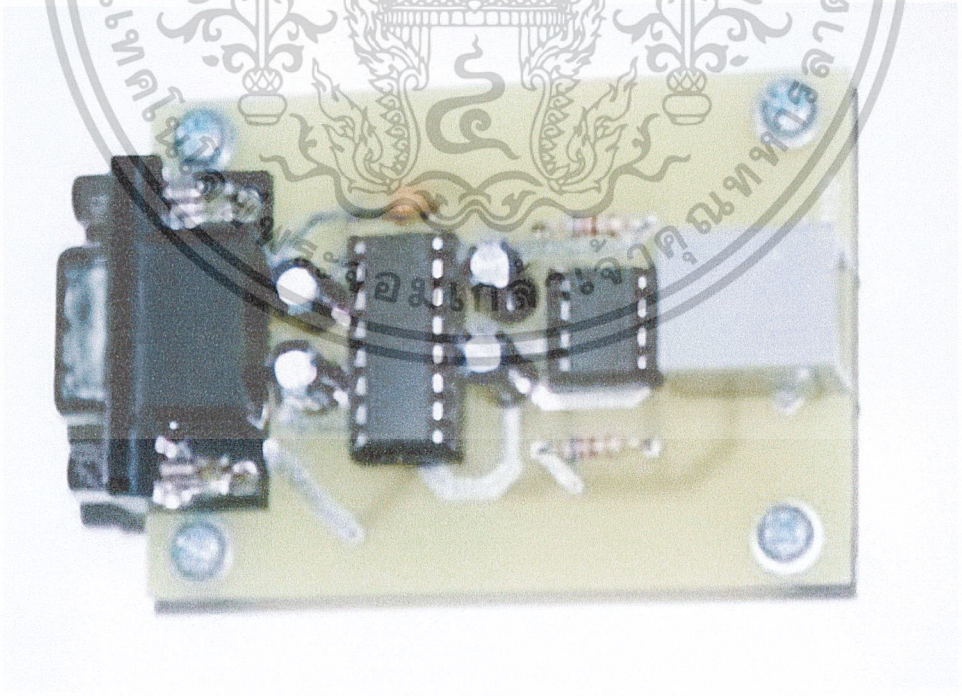


ภาคผนวก ก
เครื่องต้นแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

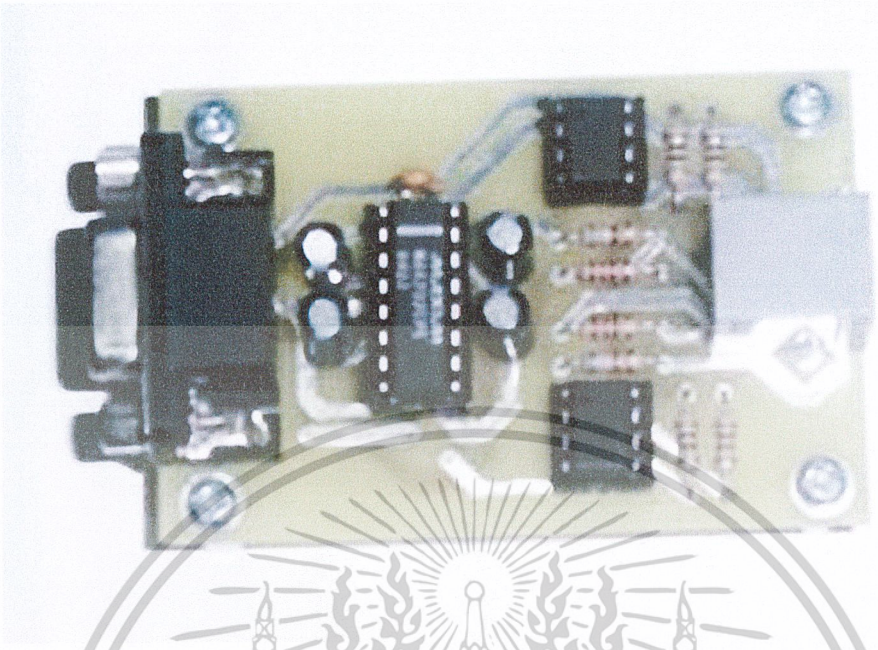


รูปที่ ก.1 ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A



รูปที่ ก.2 ชุดวงจรแปลง RS-232 เป็น RS-485

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



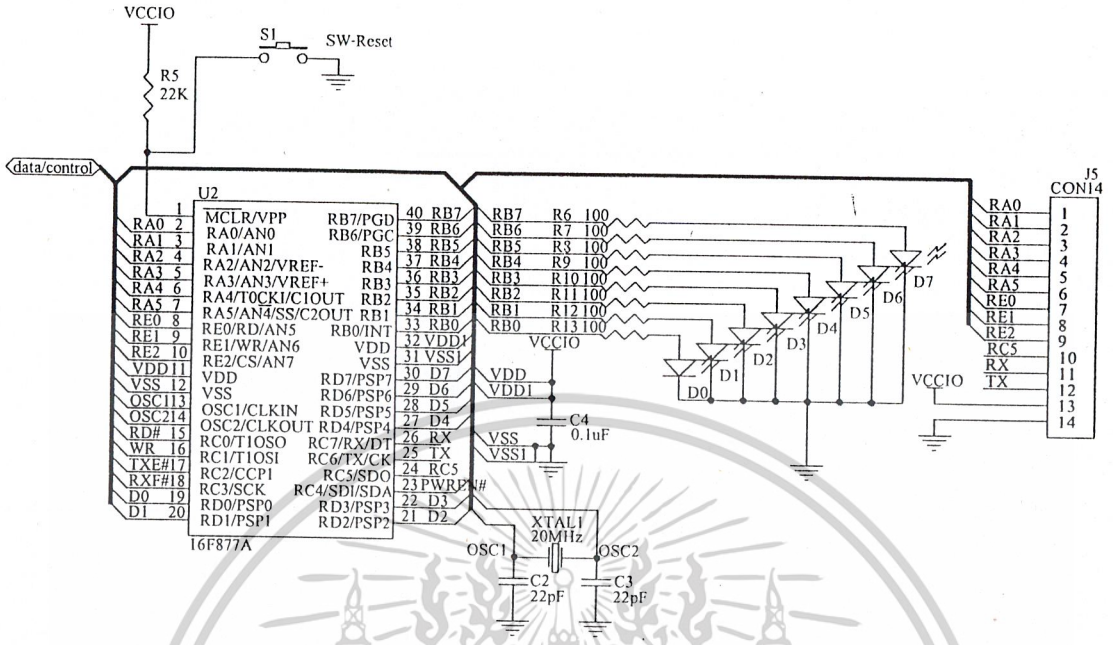
รูปที่ ก.3 ชุดวงจรแปลง RS-232 เป็น RS-422

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

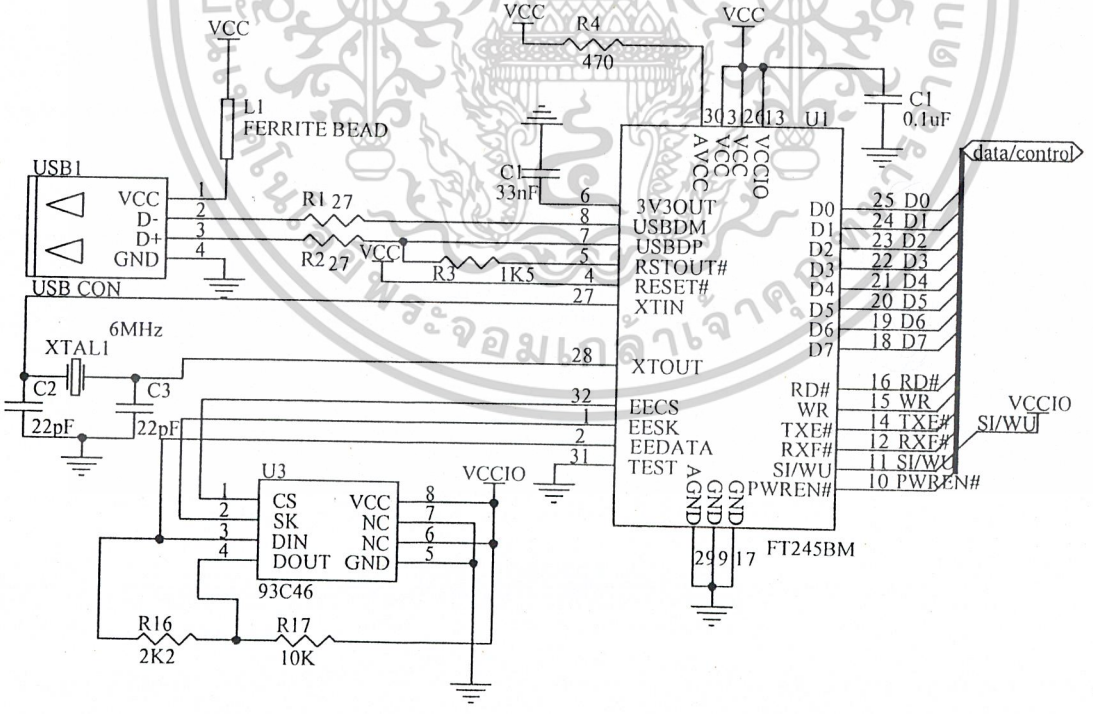
ภาคผนวก ข

วงจรและแผ่นวงจรพิมพ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

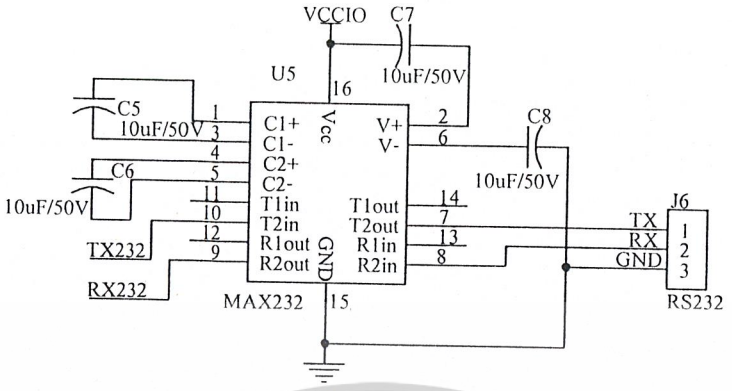


รูปที่ ข.1 วงจรควบคุมการทำงานหลัก

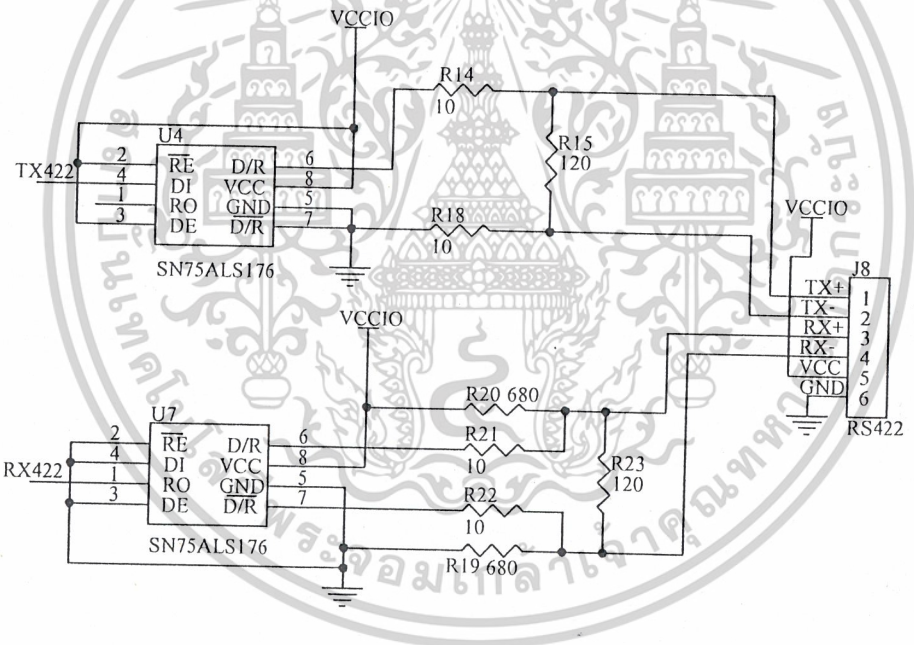


รูปที่ ข.2 วงจรติดต่อ USB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

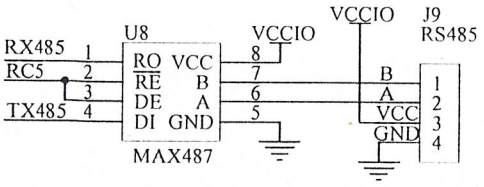


รูปที่ ข.3 วงจรแปลง RS-232

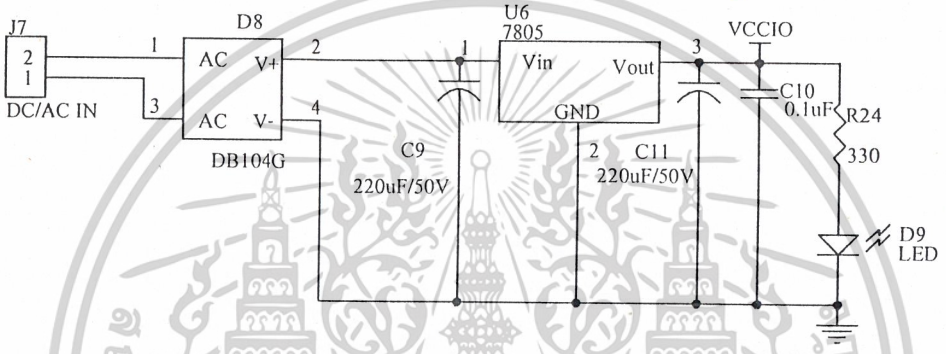


รูปที่ ข.4 วงจรแปลง RS-422

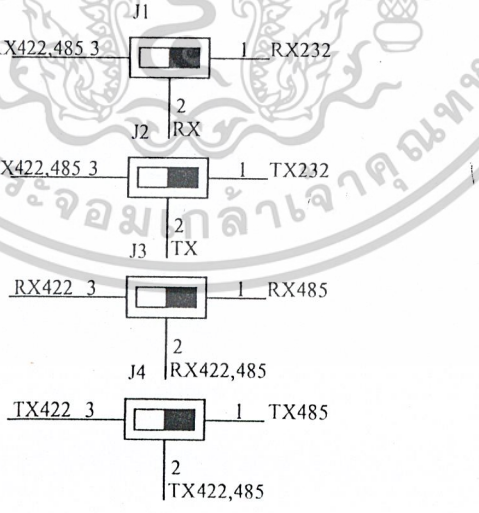
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.



รูปที่ ข.5 วงจรแปลง RS-485

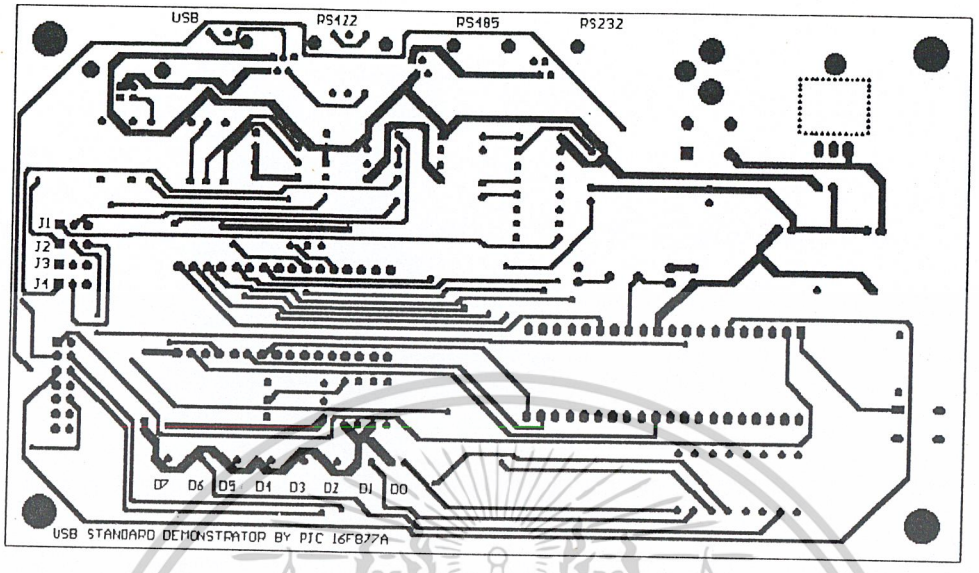


รูปที่ ข.6 วงจรแหล่งจ่ายไฟฟ้า

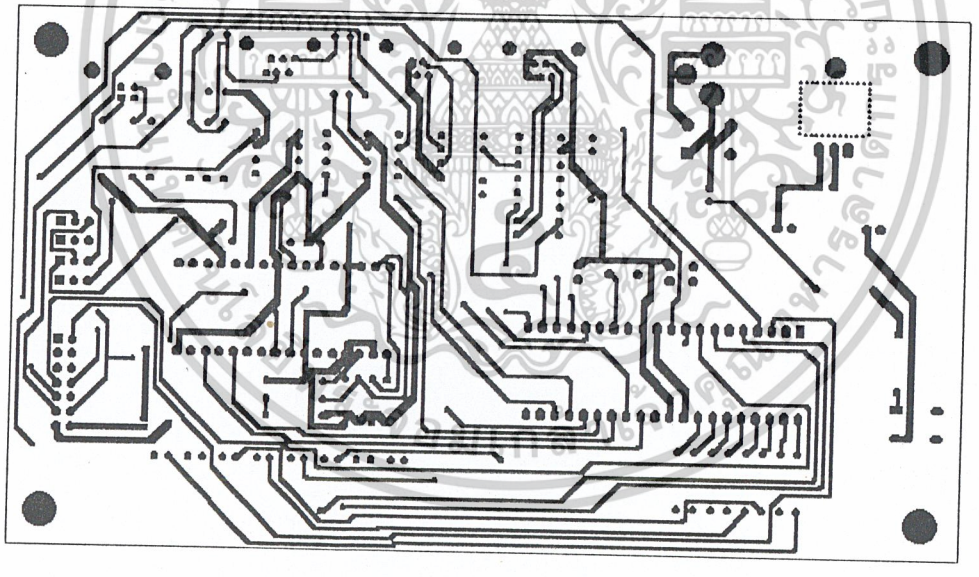


รูปที่ ข.7 จัมเปอร์เลือกใช้ RS-232, RS-422 และRS-485

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

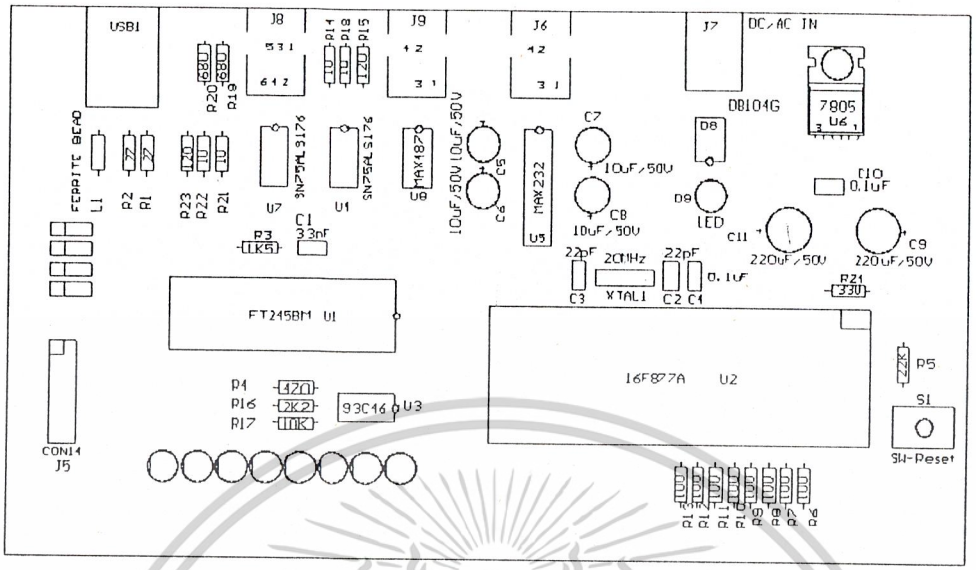


รูปที่ ข.8 แผ่นวงจรพิมพ์ชุดวงจรหลัก ด้านบน (ขนาด 75%)

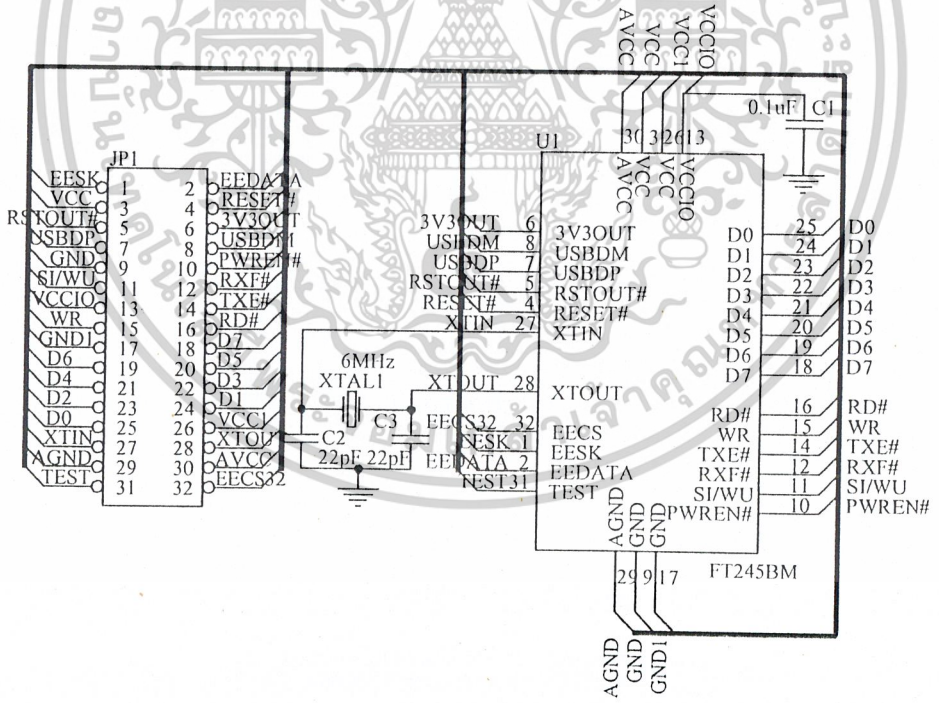


รูปที่ ข.9 แผ่นวงจรพิมพ์ชุดวงจรหลัก ด้านล่าง (ขนาด 75%)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

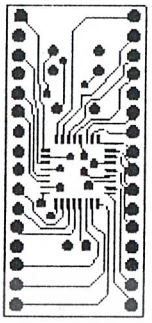


รูปที่ ข.10 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์ชุดวงจรหลัก (ขนาด 75%)

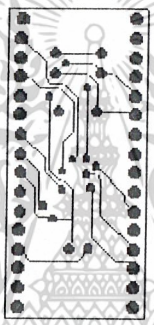


รูปที่ ข.11 วงจร USB โมดูล

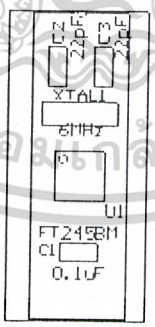
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .



รูปที่ ข.12 แผ่นวงจรพิมพ์วงจร USB โมดูล ด้านบน

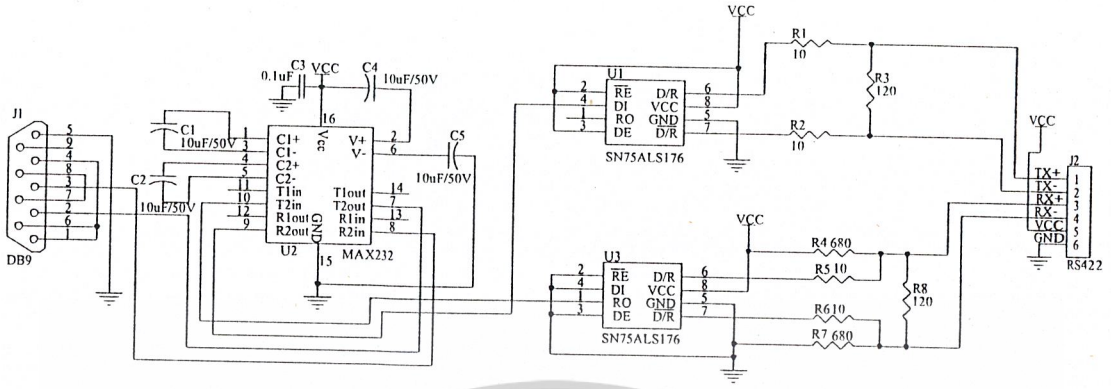


รูปที่ ข.13 แผ่นวงจรพิมพ์วงจร USB โมดูล ด้านล่าง

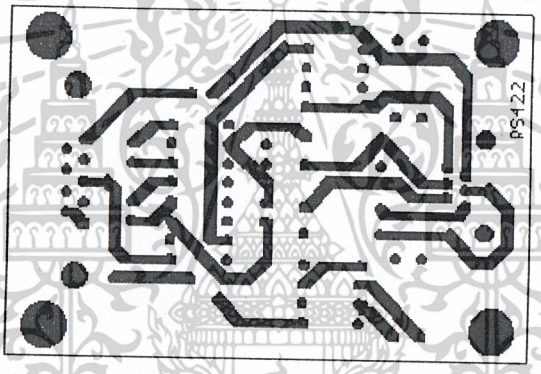


รูปที่ ข.14 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจร USB โมดูล

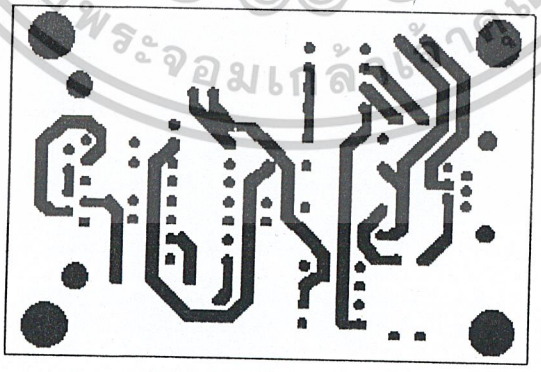
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข.15 วงจรแปลง RS-422 เป็น RS-232

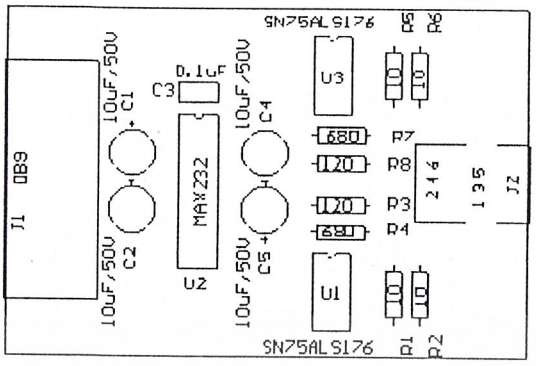


รูปที่ ข.16 แผ่นวงจรพิมพ์วงจรแปลง RS-422 เป็น RS-232 ด้านบน

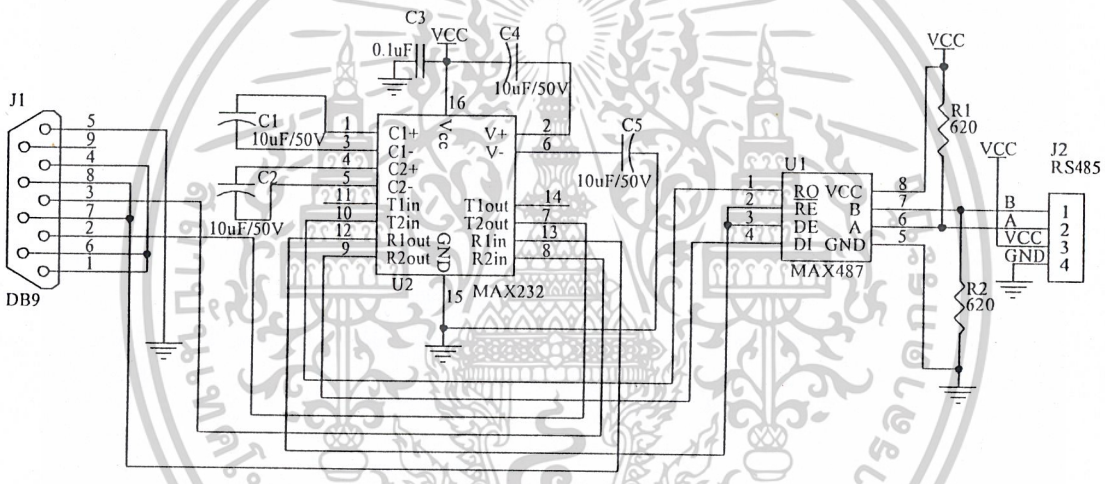


รูปที่ ข.17 แผ่นวงจรพิมพ์วงจรแปลง RS-422 เป็น RS-232 ด้านล่าง

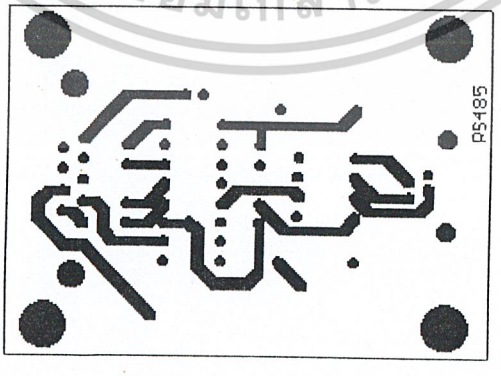
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข.18 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรแปลง RS-422 เป็น RS-232

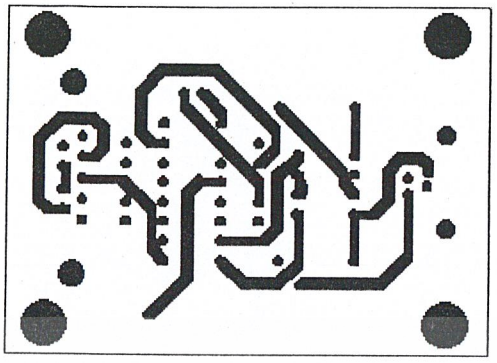


รูปที่ ข.19 วงจรแปลง RS-485 เป็น RS-232

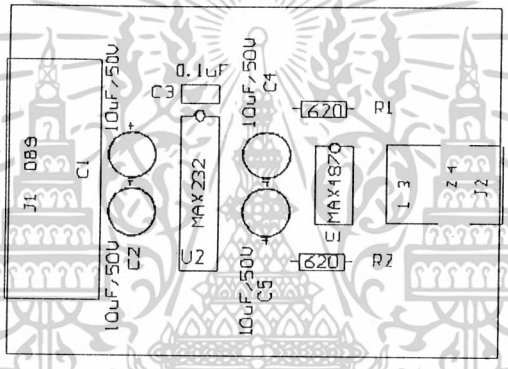


รูปที่ ข.20 แผ่นวงจรพิมพ์วงจรแปลง RS-485 เป็น RS-232 ด้านบน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข.21 แผ่นวงจรพิมพ์วงจรแปลง RS-485 เป็น RS-232 ด้านล่าง



รูปที่ ข.22 ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรแปลง RS-485 เป็น RS-232

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาคผนวก ค

รายการอุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ค.1 รายการอุปกรณ์ของวงจรชุดสาธิตหลัก

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
U2	PIC 16F877A	1
U3	93C46	1
U4, U7	SN75ALS176	2
U5	MAX232	1
U6	7805	1
U8	MAX487	1
ตัวเก็บประจุ		
C1	33nF	1
C2, C3	22pF	2
C4	0.1uF	1
C5-C8	10uF/50V	4
C9, C11	220uF/50V	2
C10	0.1uF	1
ตัวต้านทาน		
R1, R2	27Ω ¼ W	2
R3	1K5Ω ¼ W	1
R4	470Ω ¼ W	1
R5	22KΩ ¼ W	1
R6-R13	100Ω ¼ W	8
R14, R18, R21, R22	10Ω ¼ W	4
R15, R23	120Ω ¼ W	2
R16	2K2Ω ¼ W	1
R17	10KΩ ¼ W	1
R19, R20	680Ω ¼ W	2
R24	330Ω ¼ W	1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ค.2 (ต่อ) รายการอุปกรณ์ของวงจร USB โมดูล

ชื่ออุปกรณ์	รายละเอียด	จำนวน
อุปกรณ์อื่นๆ		
XTAL1	6MHz	1

ตารางที่ ค.3 รายการอุปกรณ์ของวงจรแปลง RS-422 to RS-232

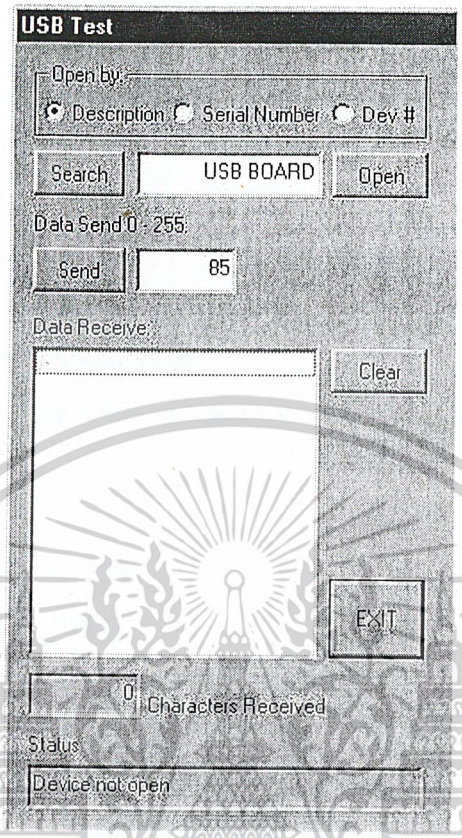
ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
U1, U3	SN75ALS176	2
U2	MAX232	1
ตัวเก็บประจุ		
C1, C2, C4, C5	10uF/50V	4
C3	0.1uF	1
ตัวต้านทาน		
R1, R2, R5, R6	10Ω ¼ W	4
R3	120Ω ¼ W	1
R4, R7	680Ω ¼ W	2
R8	120Ω ¼ W	1
อุปกรณ์อื่นๆ		
J1	DB-9RA/F	1
J2	TEL 6 PIN	1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ค.4 รายการอุปกรณ์ของวงจรแปลง RS-485 to RS-232

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
U1	MAX487	1
U2	MAX232	1
ตัวเก็บประจุ		
C1, C2, C4, C5	10uF/50V	4
C3	0.1uF	1
ตัวต้านทาน		
R1	620Ω ¼ W	1
R2	620Ω ¼ W	1
อุปกรณ์อื่นๆ		
J1	DB-9RA/F	1
J2	TEL 4 PIN	1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ง.1 โปรแกรม USB Test

โปรแกรม USB Test

USBTestDlg.cpp

```
// USBtestDlg.cpp : implementation file
#include "stdafx.h"
#include "USBtest.h"
#include "USBtestDlg.h"
#include "ftd2xx.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//globals
int running;
int board_present;
int bus_busy;

// CAboutDlg dialog used for App About
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void CUSBtestDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CUSBtestDlg)
	DDX_Control(pDX, IDC_LIST1, m_Received);
	DDX_Text(pDX, IDC_EDIT_PORT_STATUS, m_PortStatus);
	DDX_Text(pDX, IDC_EDIT_NUM_RECD, m_NumRecd);
	DDX_Text(pDX, IDC_EDIT_NAME_NUMBER, m_NameNmbr);
	DDX_Radio(pDX, IDC_RADIO_NAME_NUM, m_SerDesc);
	DDX_Text(pDX, IDC_EDIT_DATA, m_DATA);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CUSBtestDlg, Cdialog)
	//{{AFX_MSG_MAP(CUSBtestDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_BUTTON_CLEAR, OnButtonClear)
	ON_BN_CLICKED(IDC_BUTTON_HELLO, OnButtonHello)
	ON_BN_CLICKED(IDC_RADIO_NAME_NUM, OnRadioNameNum)
	ON_BN_CLICKED(IDC_RADIO_SERNUM, OnRadioSernum)
	ON_EN_CHANGE(IDC_EDIT_NAME_NUMBER, OnChangeEditNameNumber)
	ON_BN_CLICKED(IDC_RADIO_DEVNO, OnRadioDevno)
	ON_BN_CLICKED(IDC_BUTTON_SEARCH, OnButtonSearch)
	ON_LBN_SELCHANGE(IDC_LIST1, OnSelchangeList1)
	ON_BN_CLICKED(IDC_BUTTON_SEND, OnButtonSend)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CUSBtestDlg message handlers

void CUSBtestDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

void CUSBtestDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc
(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(&rect);
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;

            // Draw the icon
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            CDialog::OnPaint();
      }
}

// The system calls this to obtain the cursor to display while the
user drags
//  the minimized window.
HCURSOR CUSBtestDlg::OnQueryDragIcon()
{
      return (HCURSOR) m_hIcon;
}

//*****************************************************************************
BOOL CUSBtestDlg::OnInitDialog()
{
      CDialog::OnInitDialog();

      // Add "About..." menu item to system menu.

      // IDM_ABOUTBOX must be in the system command range.
      ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
      ASSERT(IDM_ABOUTBOX < 0xF000);

      CMenu* pSysMenu = GetSystemMenu(FALSE);
      if (pSysMenu != NULL)
      {
            CString strAboutMenu;
            strAboutMenu.LoadString(IDS_ABOUTBOX);
            if (!strAboutMenu.IsEmpty())
            {
                  pSysMenu->AppendMenu(MF_SEPARATOR);
                  pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
            }
      }

      // Set the icon for this dialog.  The framework does this
automatically
      //  when the application's main window is not a dialog
      SetIcon(m_hIcon, TRUE);            // Set big icon
      SetIcon(m_hIcon, FALSE);           // Set small icon

      // TODO: Add extra initialization here
      bus_busy=0;
      board_present=0;
      SetTimer(15, 100, NULL);//check for new USB data in the
buffer every 100mS
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            UpdateWindow();

            bytes_in_buf = 0;
      }

      if(bytes_in_buf)
      {
            status = Read(rx, bytes_in_buf<600 ? bytes_in_buf :
600, &ret_bytes);
            if((status == FT_OK) || ((status == FT_IO_ERROR) &&
(ret_bytes > 0)) /* Read timeout */)
            {
                  if(ret_bytes)
                  {
                        CString str;
                        UpdateData(TRUE);
                        for(DWORD x=0; x<ret_bytes; x++)
                        {
                              m_NumRecd++;
                              str.Format("%d  0x%.2X", rx[x], rx
[x]);
                              m_Received.AddString(str);
                        }
                  }
            }

            UpdateData(FALSE);
            UpdateWindow();
      }

      CDialog::OnTimer(nIDEvent);
}

//*****************************************************************
BOOL CUSBtestDlg::OnCommand(WPARAM wParam, LPARAM lParam) //disable
the ESC key
{
      if(wParam==2) return FALSE;
      return CDialog::OnCommand(wParam, lParam);
}


//*****************************************************************
void CUSBtestDlg::OnButtonClear()
{
            UpdateData(TRUE);
            m_NumRecd=0;
            m_Received.ResetContent();
            UpdateData(FALSE);
            UpdateWindow();
}


//*****************************************************************
void CUSBtestDlg::LoadDLL()
{
      m_hmodule = LoadLibrary("Ftd2xx.dll");
      if(m_hmodule == NULL)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
	{
		AfxMessageBox("Error: Can't Load ft8u245.dll");
		return;
	}

	m_pWrite = (PtrToWrite)GetProcAddress(m_hmodule, "FT_Write");
	if (m_pWrite == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_Write");
		return;
	}

	m_pRead = (PtrToRead)GetProcAddress(m_hmodule, "FT_Read");
	if (m_pRead == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_Read");
		return;
	}

	m_pOpen = (PtrToOpen)GetProcAddress(m_hmodule, "FT_Open");
	if (m_pOpen == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_Open");
		return;
	}

	m_pOpenEx = (PtrToOpenEx)GetProcAddress(m_hmodule,
"FT_OpenEx");
	if (m_pOpenEx == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_OpenEx");
		return;
	}

	m_pListDevices = (PtrToListDevices)GetProcAddress(m_hmodule,
"FT_ListDevices");
	if(m_pListDevices == NULL)
		{
			AfxMessageBox("Error: Can't Find
FT_ListDevices");
			return;
		}

	m_pClose = (PtrToClose)GetProcAddress(m_hmodule, "FT_Close");
	if (m_pClose == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_Close");
		return;
	}

	m_pResetDevice = (PtrToResetDevice)GetProcAddress(m_hmodule,
"FT_ResetDevice");
	if (m_pResetDevice == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_ResetDevice");
		return;
	}
	m_pPurge = (PtrToPurge)GetProcAddress(m_hmodule, "FT_Purge");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
	if (m_pPurge == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_Purge");
		return;
	}

	m_pSetTimeouts = (PtrToSetTimeouts)GetProcAddress(m_hmodule,
"FT_SetTimeouts");
	if (m_pSetTimeouts == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_SetTimeouts");
		return;
	}

	m_pGetQueueStatus = (PtrToGetQueueStatus)GetProcAddress
(m_hmodule, "FT_GetQueueStatus");
	if (m_pGetQueueStatus == NULL)
	{
		AfxMessageBox("Error: Can't Find FT_GetQueueStatus");
		return;
	}
}


//*****************************************************************
FT_STATUS CUSBtestDlg::Read(LPVOID lpvBuffer, DWORD dwBuffSize,
LPDWORD lpdwBytesRead)
{
	if (!m_pRead)
	{
		AfxMessageBox("FT_Read is not valid!");
		return FT_INVALID_HANDLE;
	}

	return (*m_pRead)(m_ftHandle, lpvBuffer, dwBuffSize,
lpdwBytesRead);
}


//*****************************************************************
FT_STATUS CUSBtestDlg::Write(LPVOID lpvBuffer, DWORD dwBuffSize,
LPDWORD lpdwBytes)
{
	if (!m_pWrite)
	{
		AfxMessageBox("FT_Write is not valid!");
		return FT_INVALID_HANDLE;
	}

	return (*m_pWrite)(m_ftHandle, lpvBuffer, dwBuffSize,
lpdwBytes);
}


//*****************************************************************
FT_STATUS CUSBtestDlg::Open(PVOID pvDevice)
{
	if (!m_pOpen)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        {
                AfxMessageBox("FT_Open is not valid!");
                return FT_INVALID_HANDLE;
        }

        return (*m_pOpen)(pvDevice, &m_ftHandle );
}

//*********************************************************************
FT_STATUS CUSBtestDlg::OpenEx(PVOID pArg1, DWORD dwFlags)
{
        if (!m_pOpenEx)
        {
                AfxMessageBox("FT_OpenEx is not valid!");
                return FT_INVALID_HANDLE;
        }

        return (*m_pOpenEx)(pArg1, dwFlags, &m_ftHandle);
}


//*********************************************************************
FT_STATUS CUSBtestDlg::ListDevices(PVOID pArg1, PVOID pArg2, DWORD
dwFlags)
{
        if (!m_pListDevices)
        {
                AfxMessageBox("FT_ListDevices is not valid!");
                return FT_INVALID_HANDLE;
        }

        return (*m_pListDevices)(pArg1, pArg2, dwFlags);
}

//*********************************************************************
FT_STATUS CUSBtestDlg::Close()
{
        if (!m_pClose)
        {
                AfxMessageBox("FT_Close is not valid!");
                return FT_INVALID_HANDLE;
        }

        return (*m_pClose)(m_ftHandle);
}

//*********************************************************************
FT_STATUS CUSBtestDlg::ResetDevice()
{
        if (!m_pResetDevice)
        {
                AfxMessageBox("FT_ResetDevice is not valid!");
                return FT_INVALID_HANDLE;
        }

        return (*m_pResetDevice)(m_ftHandle);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

```
//*****************************************************************
FT_STATUS CUSBtestDlg::Purge(ULONG dwMask)
{
	if (!m_pPurge)
	{
		AfxMessageBox("FT_Purge is not valid!");
		return FT_INVALID_HANDLE;
	}

	return (*m_pPurge)(m_ftHandle, dwMask);
}


//*****************************************************************
FT_STATUS CUSBtestDlg::SetTimeouts(ULONG dwReadTimeout, ULONG
dwWriteTimeout)
{
	if (!m_pSetTimeouts)
	{
		AfxMessageBox("FT_SetTimeouts is not valid!");
		return FT_INVALID_HANDLE;
	}

	return (*m_pSetTimeouts)(m_ftHandle, dwReadTimeout,
dwWriteTimeout);
}


//*****************************************************************
FT_STATUS CUSBtestDlg::GetQueueStatus(LPDWORD lpdwAmountInRxQueue)
{
	if (!m_pGetQueueStatus)
	{
		AfxMessageBox("FT_GetQueueStatus is not valid!");
		return FT_INVALID_HANDLE;
	}

	return (*m_pGetQueueStatus)(m_ftHandle, lpdwAmountInRxQueue);
}


//*****************************************************************
void CUSBtestDlg::OnRadioNameNum()
{
	CWinApp* pApp = AfxGetApp();
	//write the new setup to the registry
	pApp->WriteProfileInt("MyUSBTestAp", "SerDesc", 0);

	UpdateData(TRUE);
	m_NameNmbr = AfxGetApp()->GetProfileString("MyUSBTestAp",
"DescString", "Enter device Description or Serial Number here.");
	UpdateData(FALSE);
	UpdateWindow();
}

//*****************************************************************
void CUSBtestDlg::OnRadioSernum()
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
	CWinApp* pApp = AfxGetApp();
	//write the new setup to the registry
	pApp->WriteProfileInt("MyUSBTestAp", "SerDesc", 1);

	UpdateData(TRUE);
	m_NameNmbr = AfxGetApp()->GetProfileString("MyUSBTestAp",
"SerialString", "Enter device Description or Serial Number here.");
	UpdateData(FALSE);
	UpdateWindow();
}

//*****************************************************************
void CUSBtestDlg::OnRadioDevno()
{
	CWinApp* pApp = AfxGetApp();
	//write the new setup to the registry
	pApp->WriteProfileInt("MyUSBTestAp", "SerDesc", 2);

	UpdateData(TRUE);
	m_NameNmbr = AfxGetApp()->GetProfileString("MyUSBTestAp",
"DevNmbr", "Enter device Description or Serial Number here.");
	UpdateData(FALSE);
	UpdateWindow();
}


//*****************************************************************
void CUSBtestDlg::OnChangeEditNameNumber()
{
	UpdateData(TRUE);

	//write the new setup to the registry
	if(m_SerDesc==0)
		AfxGetApp()->WriteProfileString("MyUSBTestAp",
"DescString", m_NameNmbr);
	if(m_SerDesc==1)
		AfxGetApp()->WriteProfileString("MyUSBTestAp",
"SerialString", m_NameNmbr);
	if(m_SerDesc==2)
		AfxGetApp()->WriteProfileString("MyUSBTestAp",
"DevNmbr", m_NameNmbr);

	if(m_NameNmbr.GetLength() < 1)
		m_SerDesc = 2;
	UpdateData(FALSE);
	UpdateWindow();
}




//*****************************************************************
void CUSBtestDlg::Loadem()
{
//    unsigned char txbuf[25], rxbuf[25];
//    DWORD ret bytes;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

```
	LoadDLL();

//	open the device
	FT_STATUS status = OpenBy();
	if(status)
	{
		UpdateData(TRUE);
		m_PortStatus = _T("Device not open");
		UpdateData(FALSE);
		UpdateWindow();
		board_present=0;
		GetDlgItem(IDC_BUTTON_HELLO)->EnableWindow(TRUE);
	}
	else
	{
		UpdateData(TRUE);
		m_PortStatus = _T("Device ready.");
		board_present=1;
		UpdateData(FALSE);
		UpdateWindow();
	}
	UpdateData(TRUE);
	m_NumRecd=0;
	m_Received.ResetContent();
	UpdateData(FALSE);
	UpdateWindow();
}


//*****************************************************************
FT_STATUS CUSBtestDlg::OpenBy()
{
	UpdateData(TRUE);

	if(m_NameNmbr.Find("Enter device Description") > -1)
	{
		//highlight the descriptor/serial number to draw
attention...
		CEdit *pEdt;
		pEdt = (CEdit *)GetDlgItem(IDC_EDIT_NAME_NUMBER);
		pEdt->SetFocus();
		pEdt->SetSel(0, -1, FALSE);
		AfxMessageBox("You must enter a Device Description or
Serial Number.");
		return FT_DEVICE_NOT_OPENED;
	}

	FT_STATUS status;
	ULONG x=0;
	if(m_SerDesc==0)
		status = OpenEx((PVOID)(LPCTSTR)m_NameNmbr,
FT_OPEN_BY_DESCRIPTION);
	if(m_SerDesc==1)
		status = OpenEx((PVOID)(LPCTSTR)m_NameNmbr,
FT_OPEN_BY_SERIAL_NUMBER);
	if((m_SerDesc==2) || (m_SerDesc==-1))//if open by device OR
no method was selected
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      {
            if(m_NameNmbr.GetLength() < 1)//nothing entered - open
default device 0
            {
                  status = Open((PVOID)x);//load default device 0
            }
         else
         {
                  if(m_NameNmbr.GetLength() > 2)//no Open() method
selected...
                  {
                        AfxMessageBox("Select a method to open or
enter a valid device number (0-64).");
                        return FT_DEVICE_NOT_OPENED;
                  }

                  //convert string to decimal number and open(x)
                  char str[3];
                  strcpy(str, (LPCTSTR)m_NameNmbr);
                  x = atoi(str);
                  if((x<0) || (x>64))
                  {
                        AfxMessageBox("Select a method to open or
enter a valid device number (0-64).");
                        return FT_DEVICE_NOT_OPENED;
                  }
                  status = Open((PVOID)x);
         }
      }

      return status;
}


//*****************************************************************
void CUSBtestDlg::OnButtonHello() //code for the "Open" button
{
      UpdateData(TRUE);
      m_PortStatus = _T("-Open-");
      UpdateData(FALSE);
      UpdateWindow();
      Close();
      //open the device
      FT_STATUS status = OpenBy();
      if(status>0)
      {
            m_PortStatus = _T("Could not open Device");
            board_present=0;
      }
      else
      {
            UpdateData(TRUE);
            m_PortStatus = _T("-ready-");
            UpdateData(FALSE);
            UpdateWindow();
//          buttons(ON);//turn em off till a port is active
            board_present=1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}


//*********************************************************************
void CUSBtestDlg::OnSelchangeList1()
{
	UpdateData(TRUE);

	int pos = m_Received.GetCurSel();
	m_Received.GetText(pos, m_NameNmbr);

	UpdateData(FALSE);
	UpdateWindow();

	if(m_SerDesc==0)
		AfxGetApp()->WriteProfileString("MyUSBTestAp",
"DescString", m_NameNmbr);
	if(m_SerDesc==1)
		AfxGetApp()->WriteProfileString("MyUSBTestAp",
"SerialString", m_NameNmbr);
	if(m_SerDesc==2)
		AfxGetApp()->WriteProfileString("MyUSBTestAp",
"DevNmbr", m_NameNmbr);
}


//*********************************************************************
void CUSBtestDlg::OnButtonSearch()
{
	//search for Descriptions or Serial Numbers depending on the
current mode
	FT_STATUS ftStatus;
	DWORD numDevs;

	Close();//must be closed to perform the ListDevices()
function
	UpdateData(TRUE);
	m_PortStatus = _T("Closed");
	m_NumRecd=0;
	m_NameNmbr = _T("");
	m_Received.ResetContent();
	UpdateData(FALSE);
	UpdateWindow();

	ftStatus = ListDevices(&numDevs, NULL, FT_LIST_NUMBER_ONLY);
	if(ftStatus == FT_OK)
	{
		// FT_ListDevices OK, show number of devices connected
in list box
		CString str;
		str.Format("%d device(s) attached:", (int)numDevs);
		m_Received.AddString(str);

		//if current mode is open "by device #" then list
device numbers
		if((m_SerDesc==2) && (numDevs>0))
		{
			for(DWORD d=0; d<numDevs; d++)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                {
                        str.Format("%d", d);
                        m_Received.AddString(str);
                }

        }



        //if current mode is open "by description" then list
descriptions of all connected devices
        if((m_SerDesc==0) && (numDevs>0))
        {
                ftStatus = ListDevices(&numDevs, NULL,
FT_LIST_NUMBER_ONLY);
                if(ftStatus == FT_OK)
                {
                        char *BufPtrs[64]; // pointer to array of
64 pointers
                        for(DWORD d=0; d<numDevs; d++)
                                BufPtrs[d] = new char[64];
                        BufPtrs[d] = NULL;

                        FtStatus = ListDevices(BufPtrs, &numDevs,
FT_LIST_ALL|FT_OPEN_BY_DESCRIPTION);
                        if (FT_SUCCESS(ftStatus))
                        {
                                for(DWORD u=0; u<numDevs; u++)
                                {
                                        str.Format("%s", BufPtrs[u]);
                                        m_Received.AddString(str);

                                }
                        }
                        else
                        {
                                str.Format("ListDevices failed");
                                m_Received.AddString(str);
                        }

                        //free ram to avoid memory leaks
                        for(d=0; d<numDevs; d++)
                        {
                                delete BufPtrs[d];
                        }
                }
        }

        //if current mode is open "by serial number" the list
descriptions
        //of all connected devices
        if((m_SerDesc==1) && (numDevs>0))
        {
                //AfxMessageBox("by serial");
                ftStatus = ListDevices(&numDevs, NULL,
FT_LIST_NUMBER_ONLY);
                if(ftStatus == FT_OK)
                {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                        char *BufPtrs[64]; // pointer to array of
64 pointers
                        for(DWORD d=0; d<numDevs; d++)
                              BufPtrs[d] = new char[64];
                        BufPtrs[d] = NULL;

                        FtStatus = ListDevices(BufPtrs, &numDevs,
FT_LIST_ALL|FT_OPEN_BY_SERIAL_NUMBER);
                        if (FT_SUCCESS(ftStatus))
                        {
                              for(DWORD u=0; u<numDevs; u++)
                              {
                                    str.Format("%s", BufPtrs[u]);
                                    m_Received.AddString(str);

                              }
                        }
                        else
                        {
                              str.Format("ListDevices failed");
                              m_Received.AddString(str);
                        }

                        //free ram to avoid memory leaks
                        for(d=0; d<numDevs; d++)
                              delete BufPtrs[d];
                  }
            }
      }
      else
      {
            // FT_ListDevices failed
            AfxMessageBox("FT_ListDevices failed");
      }
}
//*****************************************************************
void CUSBtestDlg::OnButtonSend()
{
      unsigned char txbuf;
      DWORD ret_bytes;

      UpdateData(TRUE);
      m_PortStatus = _T("-Send-");
      UpdateData(FALSE);
      UpdateWindow();

      Purge(FT_PURGE_RX || FT_PURGE_TX);
      txbuf = m_DATA;
      Write(&txbuf, 1, &ret_bytes);

      UpdateData(TRUE);
      m_PortStatus = _T("-ready-");
      UpdateData(FALSE);
      UpdateWindow();
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

Stdafx.h

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//       are changed infrequently
//

#if
!defined(AFX_STDAFX_H__07909308_D75E_11D4_A644_93781B97CC41__INCLUD
ED_)
#define
AFX_STDAFX_H__07909308_D75E_11D4_A644_93781B97CC41__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN            // Exclude rarely-used stuff from
Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>           // MFC support for Internet Explorer
4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                 // MFC support for Windows
Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_STDAFX_H__07909308_D75E_11D4_A644_93781B97CC41__INCLUD
ED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

USBtest.h

```
// USBtest.h : main header file for the USBTEST application
//
#if
!defined(AFX_USBTEST_H__07909304_D75E_11D4_A644_93781B97CC41__INCLU
DED_)
#define
AFX_USBTEST_H__07909304_D75E_11D4_A644_93781B97CC41__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
      #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
/////////////////////////////////////////////////////////////////////////
// CUSBtestApp:
// See USBtest.cpp for the implementation of this class
//

class CUSBtestApp : public CWinApp
{
public:
      CUSBtestApp();

// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CUSBtestApp)
      public:
      virtual BOOL InitInstance();
      //}}AFX_VIRTUAL

// Implementation

      //{{AFX_MSG(CUSBtestApp)
            // NOTE - the ClassWizard will add and remove member
functions here.
            //    DO NOT EDIT what you see in these blocks of
generated code !
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.


#endif //
!defined(AFX_USBTEST_H__07909304_D75E_11D4_A644_93781B97CC41__INCLU
DED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

USBtestDlg.h

```
// USBtestDlg.h : header file
//

#if
!defined(AFX_USBTESTDLG_H__07909306_D75E_11D4_A644_93781B97CC41__IN
CLUDED_)
#define
AFX_USBTESTDLG_H__07909306_D75E_11D4_A644_93781B97CC41__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "ftd2xx.h"

UINT ThreadProc1(LPVOID param);
const int ON = 100;
const int OFF = 101;

// CUSBtestDlg dialog
class CUSBtestDlg : public CDialog
{
// Construction
public:
      CUSBtestDlg(CWnd* pParent = NULL);  // standard constructor
      void CUSBtestDlg::buttons(int onoff);
      void CUSBtestDlg::Loadem();
      FT_STATUS Write(LPVOID, DWORD, LPDWORD);
      FT_STATUS CUSBtestDlg::OpenBy();

// Dialog Data
      //{{AFX_DATA(CUSBtestDlg)
      enum { IDD = USBTEST };
      CSliderCtrl m_Slide;
      CListBox    m_Received;
      CString     m_PortStatus;
      long  m_NumRecd;
      CString     m_NameNmbr;
      int         m_SerDesc;
      BYTE  m_DATA;
      //}}AFX_DATA

      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CUSBtestDlg)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);// DDX/DDV
support
      virtual BOOL OnCommand(WPARAM wParam, LPARAM lParam);
      //}}AFX_VIRTUAL

// Implementation
protected:
      HICON m_hIcon;
      HMODULE m_hmodule;
      FT_HANDLE m_ftHandle;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        void LoadDLL();

        typedef FT_STATUS (WINAPI *PtrToOpen)(PVOID, FT_HANDLE *);
        PtrToOpen m_pOpen;
        FT_STATUS Open(PVOID);

        typedef FT_STATUS (WINAPI *PtrToOpenEx)(PVOID, DWORD,
FT_HANDLE *);
        PtrToOpenEx m_pOpenEx;
        FT_STATUS OpenEx(PVOID, DWORD);

        typedef FT_STATUS (WINAPI *PtrToListDevices)(PVOID, PVOID,
DWORD);
        PtrToListDevices m_pListDevices;
        FT_STATUS ListDevices(PVOID, PVOID, DWORD);

        typedef FT_STATUS (WINAPI *PtrToClose)(FT_HANDLE);
        PtrToClose m_pClose;
        FT_STATUS Close();

        typedef FT_STATUS (WINAPI *PtrToRead)(FT_HANDLE, LPVOID,
DWORD, LPDWORD);
        PtrToRead m_pRead;
        FT_STATUS Read(LPVOID, DWORD, LPDWORD);

        typedef FT_STATUS (WINAPI *PtrToWrite)(FT_HANDLE, LPVOID,
DWORD, LPDWORD);
        PtrToWrite m_pWrite;
        //FT_STATUS Write(LPVOID, DWORD, LPDWORD);

        typedef FT_STATUS (WINAPI *PtrToResetDevice)(FT_HANDLE);
        PtrToResetDevice m_pResetDevice;
        FT_STATUS ResetDevice();

        typedef FT_STATUS (WINAPI *PtrToPurge)(FT_HANDLE, ULONG);
        PtrToPurge m_pPurge;
        FT_STATUS Purge(ULONG);

        typedef FT_STATUS (WINAPI *PtrToSetTimeouts)(FT_HANDLE,
ULONG, ULONG);
        PtrToSetTimeouts m_pSetTimeouts;
        FT_STATUS SetTimeouts(ULONG, ULONG);

        typedef FT_STATUS (WINAPI *PtrToGetQueueStatus)(FT_HANDLE,
LPDWORD);
        PtrToGetQueueStatus m_pGetQueueStatus;
        FT_STATUS GetQueueStatus(LPDWORD);

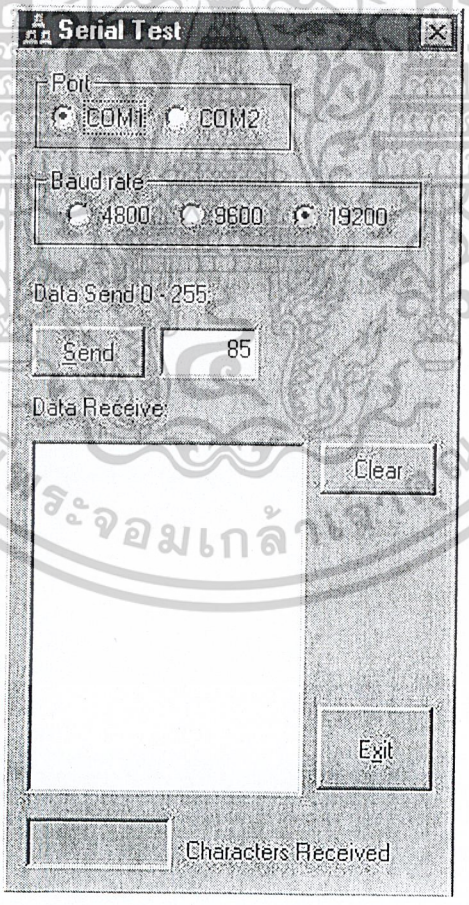
        // Generated message map functions
        //{{AFX_MSG(CUSBtestDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        virtual void OnOK();
        afx_msg void OnTimer(UINT nIDEvent);
        afx_msg void OnButtonClear();
        afx_msg void OnButtonHello();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        afx_msg void OnRadioNameNum();
        afx_msg void OnRadioSernum();
        afx_msg void OnChangeEditNameNumber();
        afx_msg void OnRadioDevno();
        afx_msg void OnButtonSearch();
        afx_msg void OnSelchangeList1();
        afx_msg void OnButtonSend();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_USBTESTDLG_H__07909306_D75E_11D4_A644_93781B97CC41__IN
CLUDED_)
```



รูปที่ ง.2 โปรแกรม Serial Test

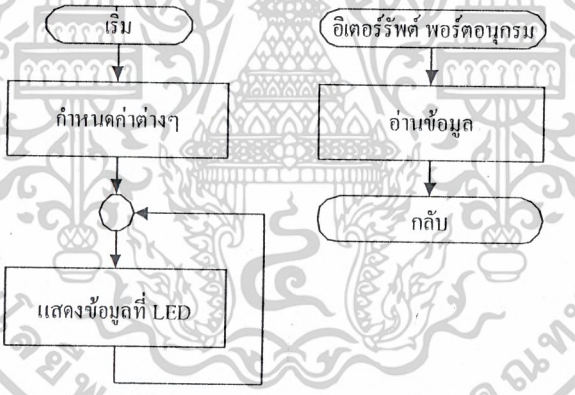
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม Serial Test

```
Private Sub Cmd_clr_Click()
List1.Clear
End Sub
'-------------------------------------------------------
Private Sub Command_exit_Click()
End
End Sub
'-------------------------------------------------------
Private Sub Command_send_Click()
    MSComm1.RTSEnable = False
If Val(Texttx.Text) < 256 Then
    MSComm1.Output = Chr(Val(Texttx.Text))
    For i = 0 To 100
    DoEvents
    Next I
Else
    MsgBox ("ป้อนข้อมูลระหว่าง 0-255")
End If
MSComm1.RTSEnable = True
End Sub
'-------------------------------------------------------
Private Sub Form_Load()
MSComm1.CommPort = 1
MSComm1.Settings = "19200,n,8,1"
MSComm1.InputLen = 1
MSComm1.RThreshold = 1
If Not MSComm1.PortOpen Then
MSComm1.PortOpen = True
End If
MSComm1.RTSEnable = True
End Sub
'-------------------------------------------------------
Private Sub Option_19200_Click()
MSComm1.PortOpen = False
MSComm1.Settings = "19200,n,8,1"
MSComm1.PortOpen = True
End Sub
'-------------------------------------------------------
Private Sub Option_4800_Click()
MSComm1.PortOpen = False
MSComm1.Settings = "4800,n,8,1"
MSComm1.PortOpen = True
End Sub
'-------------------------------------------------------
Private Sub Option_9600_Click()
MSComm1.PortOpen = False
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub
'-------------------------------------------------------
Private Sub Option_com1_Click()
MSComm1.PortOpen = False
MSComm1.CommPort = 1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
MSComm1.PortOpen = True
End Sub
'-------------------------------------------------------
Private Sub Option_com2_Click()
MSComm1.PortOpen = False
MSComm1.CommPort = 2
MSComm1.PortOpen = True
End Sub
'-------------------------------------------------------
Private Sub MSComm1_OnComm()
    Dim tmp, tmp1 As Variant
    If MSComm1.InBufferCount > 0 Then
        tmp = MSComm1.Input
        If Len(Hex(Asc(tmp))) = 1 Then
            tmp1 = (Asc(tmp) & "  0x0" & Hex(Asc(tmp)))
        Else
            tmp1 = (Asc(tmp) & "  0x" & Hex(Asc(tmp)))
        End If
        List1.AddItem tmp1
        Textcnt.Text = List1.ListCount
    End If
End Sub
```



รูปที่ ง.3 ผังงานโปรแกรม Work1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม Work1

Work1.c

```
#include <WORK1.H>
byte tmp;

#int_rda
rda_isr() {
   tmp=getc(); //Read_Data
}

main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
      enable_interrupts(INT_RDA);
      enable_interrupts(GLOBAL);
   set_tris_b(0x00);
   set_tris_c(0x9c);
   bit_clear(port_c,5);
while(1){
Port_b=tmp; //Show_Data
}
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Work1.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
```

เอกสารนี้สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
#define T1_EXTERNAL_SYNC    3
#define T1_CLK_OUT          8
#define T1_DIV_BY_1         0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte   TIMER_1_LOW=         0x0e
#byte   TIMER_1_HIGH=        0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte   TIMER_2=             0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                        0
#define CCP_CAPTURE_FE                 4
#define CCP_CAPTURE_RE                 5
#define CCP_CAPTURE_DIV_4              6
#define CCP_CAPTURE_DIV_16             7
#define CCP_COMPARE_SET_ON_MATCH       8
#define CCP_COMPARE_CLR_ON_MATCH       9
#define CCP_COMPARE_INT                0xA
#define CCP_COMPARE_RESET_TIMER        0xB
#define CCP_PWM                        0xC
#define CCP_PWM_PLUS_1                 0x1c
#define CCP_PWM_PLUS_2                 0x2c
#define CCP_PWM_PLUS_3                 0x3c
#byte   CCP_1_LOW=                     0x15
#byte   CCP_1_HIGH=                    0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP2()
#byte   CCP_2_LOW=                     0x1B
#byte   CCP_2_HIGH=                    0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                    0x10
#define PSP_DISABLED                   0
#byte   PSP_DATA=                      8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER           0x20
#define SPI_SLAVE            0x24
#define SPI_L_TO_H           0
#define SPI_H_TO_L           0x10
#define SPI_CLK_DIV_4        0
#define SPI_CLK_DIV_16       1
#define SPI_CLK_DIV_64       2
#define SPI_CLK_T2           3
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

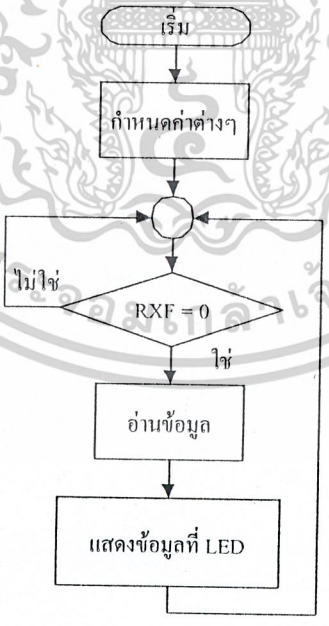
```
#define SPI_SS_DISABLED  1
#define INT_SSP                0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS

#define INT_RDA                0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG                0
#define ANALOG_RA3_REF            1
#define A_ANALOG                  2
#define A_ANALOG_RA3_REF          3
#define RA0_RA1_RA3_ANALOG        4
#define RA0_RA1_ANALOG_RA3_REF    5
#define NO_ANALOGS                6
#define ADC_OFF                   0
#define ADC_CLOCK_DIV_2           1
#define ADC_CLOCK_DIV_8           0x41
#define ADC_CLOCK_DIV_32          0x81
#define ADC_CLOCK_INTERNAL        0xc1

#define ADC_DONE               0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```



รูปที่ ง.4 ผังงานโปรแกรม Work2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

โปรแกรม Work2

Work2.c

```
#include <WORK2.H>
   byte tmp;
main() {
       setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
       setup_port_a(NO_ANALOGS);
       setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_c(0x9c);
   set_tris_b(0x00);
   set_tris_d(0xff);

   while(1){
     while((Port_c&0x08)==0x00)
     {
       bit_clear(port_c,0);
       tmp=port_d; //Read Data
       bit_set(port_c,0);
     }
     port_b=tmp; //Show data
   }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Work2.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
#define T1_EXTERNAL_SYNC    3
#define T1_CLK_OUT          8
#define T1_DIV_BY_1         0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte   TIMER_1_LOW=         0x0e
#byte   TIMER_1_HIGH=        0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte   TIMER_2=             0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                          0
#define CCP_CAPTURE_FE                   4
#define CCP_CAPTURE_RE                   5
#define CCP_CAPTURE_DIV_4                6
#define CCP_CAPTURE_DIV_16               7
#define CCP_COMPARE_SET_ON_MATCH         8
#define CCP_COMPARE_CLR_ON_MATCH         9
#define CCP_COMPARE_INT                  0xA
#define CCP_COMPARE_RESET_TIMER          0xB
#define CCP_PWM                          0xC
#define CCP_PWM_PLUS_1                   0x1c
#define CCP_PWM_PLUS_2                   0x2c
#define CCP_PWM_PLUS_3                   0x3c
#byte   CCP_1_LOW=                       0x15
#byte   CCP_1_HIGH=                      0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP2()
#byte   CCP_2_LOW=                       0x1B
#byte   CCP_2_HIGH=                      0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                      0x10
#define PSP_DISABLED                     0
#byte   PSP_DATA=                        8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER       0x20
#define SPI_SLAVE        0x24
#define SPI_L_TO_H       0
#define SPI_H_TO_L       0x10
#define SPI_CLK_DIV_4    0
#define SPI_CLK_DIV_16   1
#define SPI_CLK_DIV_64   2
#define SPI_CLK_T2       3
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

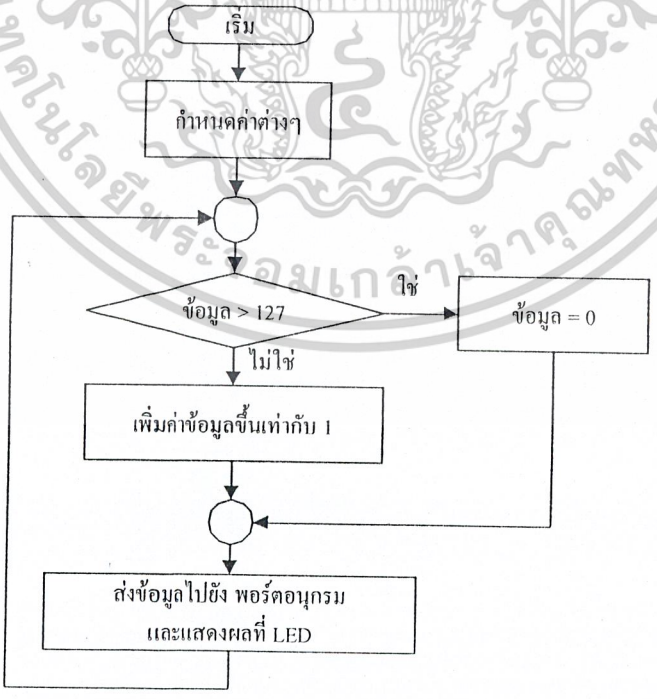
```
#define SPI_SS_DISABLED  1
#define INT_SSP                0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS

#define INT_RDA                0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////////// Constants used for SETUP_ADC
()
#define ALL_ANALOG                0
#define ANALOG_RA3_REF            1
#define A_ANALOG                  2
#define A_ANALOG_RA3_REF          3
#define RA0_RA1_RA3_ANALOG        4
#define RA0_RA1_ANALOG_RA3_REF    5
#define NO_ANALOGS                6
#define ADC_OFF                   0
#define ADC_CLOCK_DIV_2           1
#define ADC_CLOCK_DIV_8           0x41
#define ADC_CLOCK_DIV_32          0x81
#define ADC_CLOCK_INTERNAL        0xc1

#define ADC_DONE               0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```



รูปที่ ง.5 ผังงานโปรแกรม Work3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

โปรแกรม Work3

Work3.c

```
#include <WORK3.H>
   byte count;
main() {

      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_b(0x00);
   set_tris_c(0x9c);
   bit_set(port_c,5);
   while(1){
      for(count=0;count<128;count++){
      putc(count); //Send Data to Serial Port
      port_b=count; //Show Data
      delay_ms(1000);
      }
   }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

Work3.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
#define T1_EXTERNAL_SYNC    3
#define T1_CLK_OUT          8
#define T1_DIV_BY_1         0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte   TIMER_1_LOW=         0x0e
#byte   TIMER_1_HIGH=        0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte   TIMER_2=             0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                        0
#define CCP_CAPTURE_FE                 4
#define CCP_CAPTURE_RE                 5
#define CCP_CAPTURE_DIV_4              6
#define CCP_CAPTURE_DIV_16             7
#define CCP_COMPARE_SET_ON_MATCH       8
#define CCP_COMPARE_CLR_ON_MATCH       9
#define CCP_COMPARE_INT                0xA
#define CCP_COMPARE_RESET_TIMER        0xB
#define CCP_PWM                        0xC
#define CCP_PWM_PLUS_1                 0x1c
#define CCP_PWM_PLUS_2                 0x2c
#define CCP_PWM_PLUS_3                 0x3c
#byte   CCP_1_LOW=                     0x15
#byte   CCP_1_HIGH=                    0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP2()
#byte   CCP_2_LOW=                     0x1B
#byte   CCP_2_HIGH=                    0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                    0x10
#define PSP_DISABLED                   0
#byte   PSP_DATA=                      8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER       0x20
#define SPI_SLAVE        0x24
#define SPI_L_TO_H       0
#define SPI_H_TO_L       0x10
#define SPI_CLK_DIV_4    0
#define SPI_CLK_DIV_16   1
#define SPI_CLK_DIV_64   2
#define SPI_CLK_T2       3
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

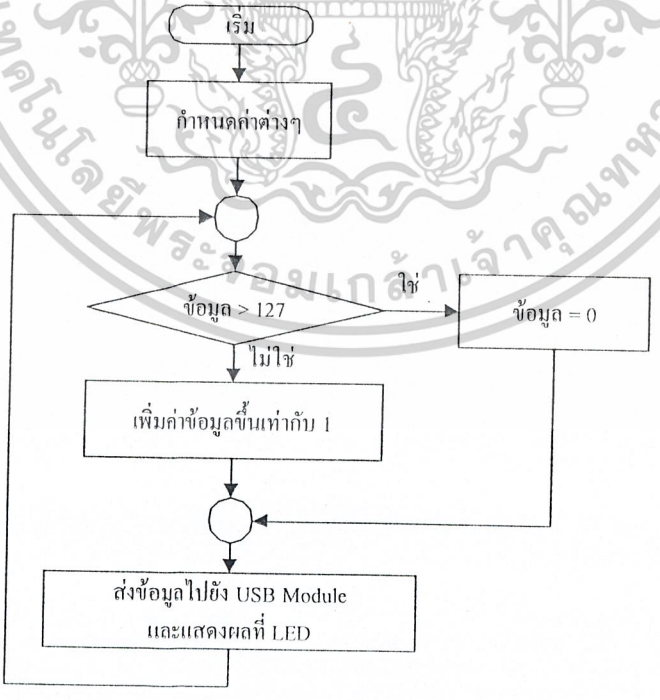
```
#define SPI_SS_DISABLED  1
#define INT_SSP                 0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS

#define INT_RDA                 0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                 0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////////// Constants used for SETUP_ADC
()
#define ALL_ANALOG              0
#define ANALOG_RA3_REF          1
#define A_ANALOG                2
#define A_ANALOG_RA3_REF        3
#define RA0_RA1_RA3_ANALOG      4
#define RA0_RA1_ANALOG_RA3_REF  5
#define NO_ANALOGS              6
#define ADC_OFF                 0
#define ADC_CLOCK_DIV_2         1
#define ADC_CLOCK_DIV_8         0x41
#define ADC_CLOCK_DIV_32        0x81
#define ADC_CLOCK_INTERNAL      0xc1

#define ADC_DONE                0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```



รูปที่ ง.6 ผังงานโปรแกรม Work4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม Work4

Work4.c

```
#include <WORK4.H>
byte count;
main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_c(0x9c);
   set_tris_d(0x00);
   set_tris_b(0x00);
   while(1){
      for(count=0;count<128;count++){
         while((Port_c&0x04)==0x04);
            bit_set(port_c,1);
            port_d=count; //Send_Data to USB Module
            bit_clear(port_c,1);
            port_b=count; //Show_data
            delay_ms(500);
      }
   }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Work4.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
#define T1_EXTERNAL_SYNC    3
#define T1_CLK_OUT          8
#define T1_DIV_BY_1         0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte   TIMER_1_LOW=         0x0e
#byte   TIMER_1_HIGH=        0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte   TIMER_2=             0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                          0
#define CCP_CAPTURE_FE                   4
#define CCP_CAPTURE_RE                   5
#define CCP_CAPTURE_DIV_4                6
#define CCP_CAPTURE_DIV_16               7
#define CCP_COMPARE_SET_ON_MATCH         8
#define CCP_COMPARE_CLR_ON_MATCH         9
#define CCP_COMPARE_INT                  0xA
#define CCP_COMPARE_RESET_TIMER          0xB
#define CCP_PWM                          0xC
#define CCP_PWM_PLUS_1                   0x1c
#define CCP_PWM_PLUS_2                   0x2c
#define CCP_PWM_PLUS_3                   0x3c
#byte   CCP_1_LOW=                       0x15
#byte   CCP_1_HIGH=                      0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP2()
#byte   CCP_2_LOW=                       0x1B
#byte   CCP_2_HIGH=                      0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                      0x10
#define PSP_DISABLED                     0
#byte   PSP_DATA=                        8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER       0x20
#define SPI_SLAVE        0x24
#define SPI_L_TO_H       0
#define SPI_H_TO_L       0x10
#define SPI_CLK_DIV_4    0
#define SPI_CLK_DIV_16   1
#define SPI_CLK_DIV_64   2
#define SPI_CLK_T2       3
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

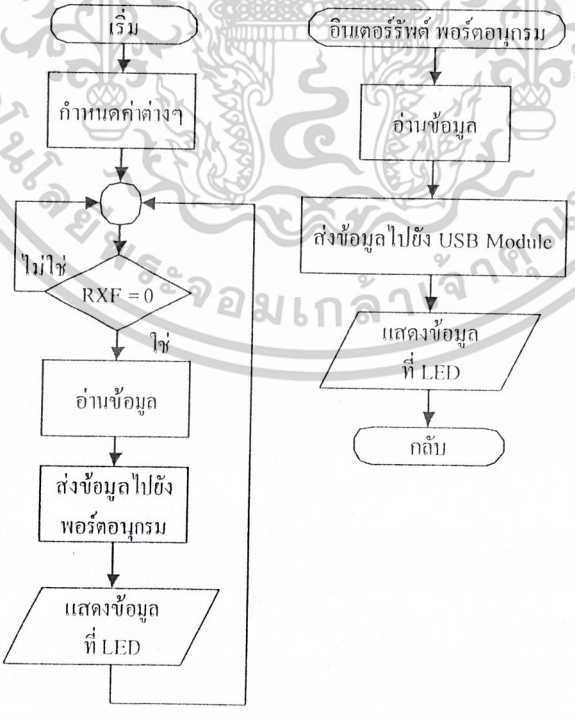
```
#define SPI_SS_DISABLED  1
#define INT_SSP                0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS

#define INT_RDA                0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG              0
#define ANALOG_RA3_REF          1
#define A_ANALOG                2
#define A_ANALOG_RA3_REF        3
#define RA0_RA1_RA3_ANALOG      4
#define RA0_RA1_ANALOG_RA3_REF  5
#define NO_ANALOGS              6
#define ADC_OFF                 0
#define ADC_CLOCK_DIV_2         1
#define ADC_CLOCK_DIV_8         0x41
#define ADC_CLOCK_DIV_32        0x81
#define ADC_CLOCK_INTERNAL      0xc1

#define ADC_DONE               0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```



รูปที่ ง.7 ผังงานโปรแกรม Work5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม Work5

Work5.c

```
#include <WORK5.H>
unsigned byte tmp,tmp1;
#int_rda
   rda_isr() {
                                    // Async Serial Data In
   set_tris_d(0x00);                //Portd = output
   tmp=getc();                      //Read_data
   port_d=tmp;                       //Read_data
   while((Port_c&0x04)==0x04);      //chack TXE#
   bit_set(port_c,1);               // WR = 1
   delay_us(1);
   port_b=tmp;                      //show_data
   bit_clear(port_c,1);             // WR = 0
}

main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
      enable_interrupts(INT_RDA);
      enable_interrupts(GLOBAL);
   set_tris_b(0x00);                //Portb = output
   set_tris_c(0x9c);                //Portc = in/output
   set_tris_d(0xff);                //Portd = input
   bit_clear(port_c,5);            //rs485 rx
   bit_clear(port_c,1);            //WR = 0
   bit_set(port_c,0);              //RD# = 1

   while(1){
      while((Port_c&0x08)==0x00) //chack RXF#
      {
         set_tris_d(0xff);          //Portd = input
         bit_set(port_c,5);         //rs485 tx
         bit_clear(port_c,0);       //RD# = 0
         tmp1=port_d;
         putc(tmp1);                // Send_data to Serial
         delay_ms(4);
         port_b=tmp1;               //show_data
         bit_clear(port_c,5);       //rs485 rx
         bit_set(port_c,0);         //RD# = 1
      }
   }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Work5.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO               0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE               0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT                 0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL                  0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED             0
#define T1_INTERNAL             5
#define T1_EXTERNAL             7
#define T1_EXTERNAL_SYNC        3
#define T1_CLK_OUT              8
#define T1_DIV_BY_1             0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

```
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte   TIMER_1_LOW=         0x0e
#byte   TIMER_1_HIGH=        0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte   TIMER_2=             0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                          0
#define CCP_CAPTURE_FE                   4
#define CCP_CAPTURE_RE                   5
#define CCP_CAPTURE_DIV_4                6
#define CCP_CAPTURE_DIV_16               7
#define CCP_COMPARE_SET_ON_MATCH         8
#define CCP_COMPARE_CLR_ON_MATCH         9
#define CCP_COMPARE_INT                  0xA
#define CCP_COMPARE_RESET_TIMER          0xB
#define CCP_PWM                          0xC
#define CCP_PWM_PLUS_1                   0x1c
#define CCP_PWM_PLUS_2                   0x2c
#define CCP_PWM_PLUS_3                   0x3c
#byte   CCP_1_LOW=                       0x15
#byte   CCP_1_HIGH=                      0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_CCP2()
#byte   CCP_2_LOW=                       0x1B
#byte   CCP_2_HIGH=                      0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                      0x10
#define PSP_DISABLED                     0
#byte   PSP_DATA=                        8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER           0x20
#define SPI_SLAVE            0x24
#define SPI_L_TO_H           0
#define SPI_H_TO_L           0x10
#define SPI_CLK_DIV_4        0
#define SPI_CLK_DIV_16       1
#define SPI_CLK_DIV_64       2
#define SPI_CLK_T2           3
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

```
#define SPI_SS_DISABLED  1
#define INT_SSP                 0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS

#define INT_RDA                 0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                 0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG                0
#define ANALOG_RA3_REF            1
#define A_ANALOG                  2
#define A_ANALOG_RA3_REF          3
#define RA0_RA1_RA3_ANALOG        4
#define RA0_RA1_ANALOG_RA3_REF    5
#define NO_ANALOGS                6
#define ADC_OFF                   0
#define ADC_CLOCK_DIV_2           1
#define ADC_CLOCK_DIV_8           0x41
#define ADC_CLOCK_DIV_32          0x81
#define ADC_CLOCK_INTERNAL        0xc1

#define ADC_DONE                0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก จ

ใบงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 1
การรับข้อมูลจากพอร์ตอนุกรม

วัตถุประสงค์เชิงพฤติกรรม

เพื่อให้นักศึกษาสามารถ

1. อธิบายหลักการการรับข้อมูลจากพอร์ตอนุกรมของ PIC 16F877A ได้
2. เขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับข้อมูลจากพอร์ตอนุกรมได้
3. สรุปและวิจารณ์ผลการทดลองได้

เครื่องมือและอุปกรณ์

1. ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A 1 ชุด
2. คอมพิวเตอร์ที่ติดตั้งโปรแกรม Serial Test 1 เครื่อง

ทฤษฎีเบื้องต้น

1. คำสั่งภาษา C for Pic

1.1 คำสั่งการกำหนดสถานะของพอร์ต

```
set_tris_b(0xXX);
```

จากคำสั่งเป็นการกำหนดสถานะของพอร์ต B ว่าต้องการให้เป็นอินพุตหรือเอาท์พุต โดยการกำหนดค่าใน (0xXX) หากต้องการให้ทั้ง 8 บิตเป็นเอาท์พุต จะกำหนดเท่ากับ (0x00) หรือถ้าต้องการให้เป็นเฉพาะบิตใดบิตหนึ่งก็ให้บิตนั้นเป็น 0

ถ้าหากต้องการให้เป็นอินพุตจะกำหนดเท่ากับ (0xFF) หรือถ้าต้องการให้เป็นเฉพาะบิตใดบิตหนึ่งก็ให้บิตนั้นเป็น 1

1.2 คำสั่งการเคลื่อนย้ายข้อมูล

```
Port_b=0xff;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากคำสั่งเป็นการนำข้อมูล ffh ไปยังพอร์ต B

1.3 คำสั่งการรับและส่งข้อมูลจากพอร์ตอนุกรม

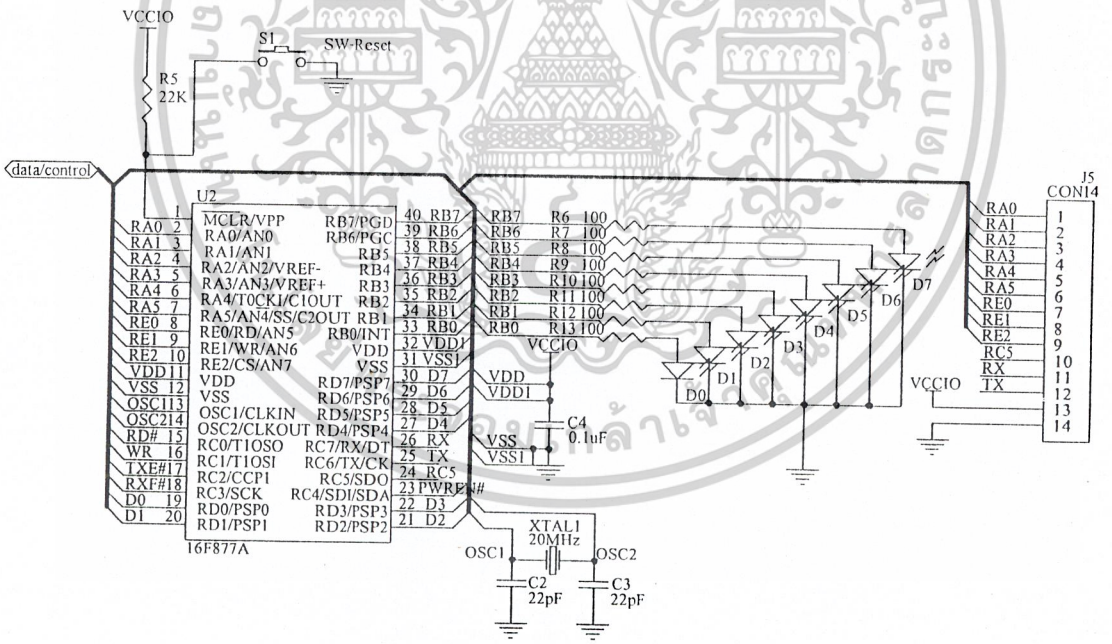
```
tmp=getc();
```

จากคำสั่งเป็นการรับข้อมูลจากพอร์ตอนุกรมไปเก็บไว้ยังตัวแปร tmp

```
putc(tmp1);
```

จากคำสั่งเป็นการส่งข้อมูลในตัวแปร tmp1 ไปยังพอร์ตอนุกรม

ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A มีส่วนของการแสดงผลซึ่งเป็น LED ดังรูป

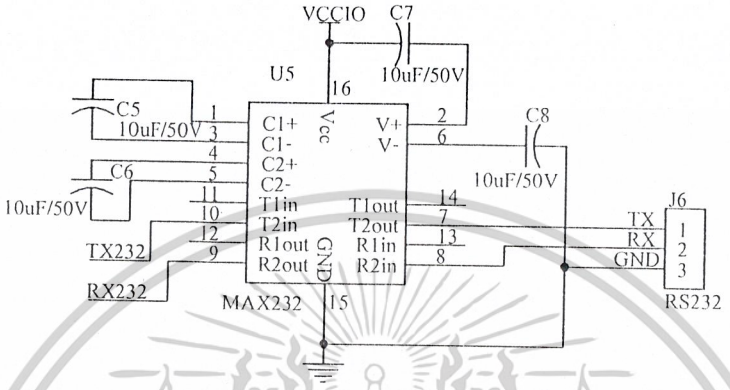


รูปที่ 1.1 วงจรการต่อ LED แสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

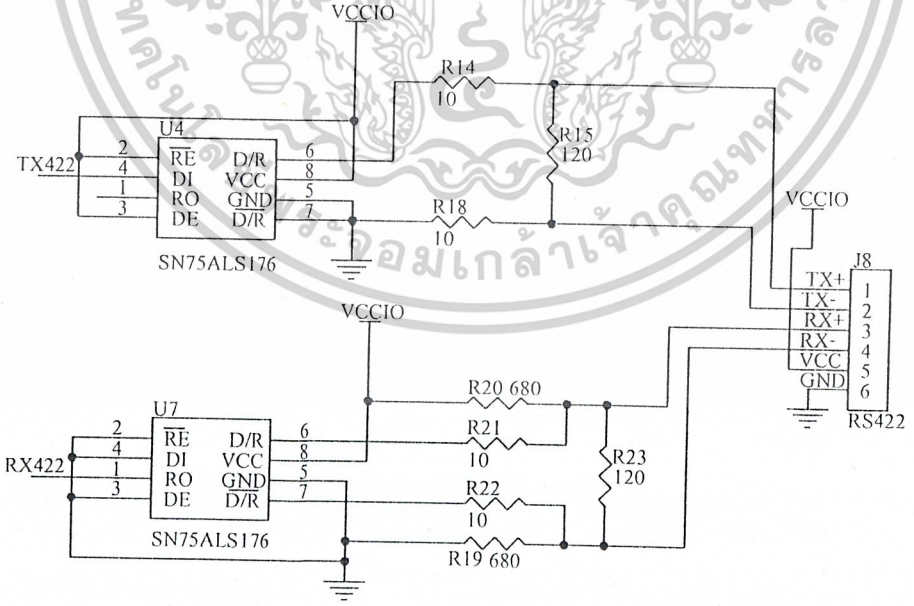
ในส่วนของวงจรสื่อสารอนุกรมนั้นได้แบ่งออกเป็น 3 วงจร ได้แก่

1. RS-232



รูปที่ 1.2 วงจรแปลงข้อมูล RS-232

2. RS-422

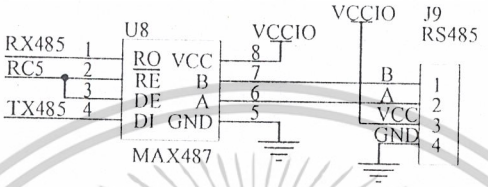


รูปที่ 1.3 วงจรแปลงข้อมูล RS-422

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. RS-485

ในการเขียนโปรแกรมเพื่อรับและส่งโดยผ่านวงจร RS-485 จะต้องมีการกำหนดสภาวะในการรับหรือส่งข้อมูลของวงจรโดยพอร์ต RC5 เมื่อต้องการรับข้อมูล กำหนดให้เป็น 0 และถ้าหากต้องการส่งข้อมูล กำหนดให้เป็น 1



รูปที่ 1.4 วงจรแปลงข้อมูล RS-485

ลำดับขั้นการทดลอง

1. เชื่อมต่อชุดสาธิตมาตรฐาน USB โดย PIC 16F877A เข้ากับคอมพิวเตอร์โดยใช้ชุด RS-232 และเซต (Set) ตัวจัมเปอร์ ดังรูปที่ 1.5

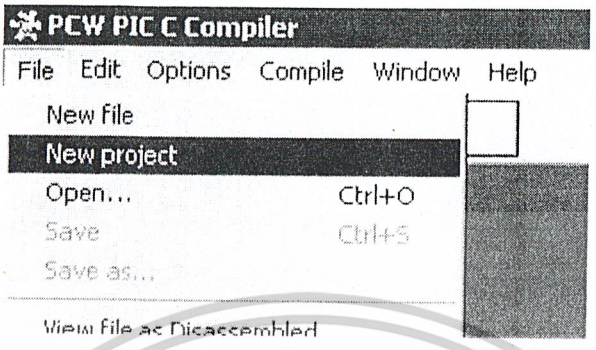


รูปที่ 1.5 การเซตจัมเปอร์เมื่อใช้วงจร RS-232

2. เปิดโปรแกรม PCW

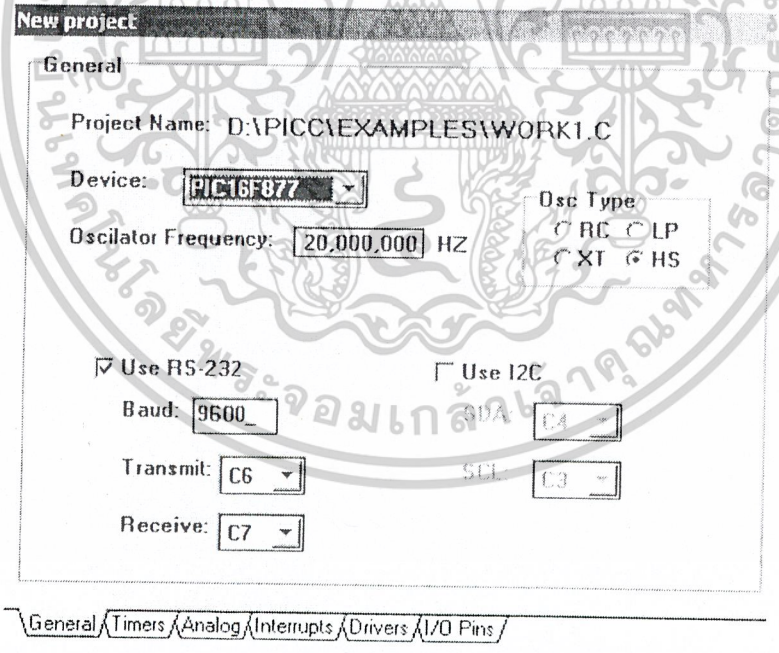
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

3. สร้าง Project โดยคลิกที่ File > New project



รูปที่ 1.6 เมนูสร้าง project ใหม่

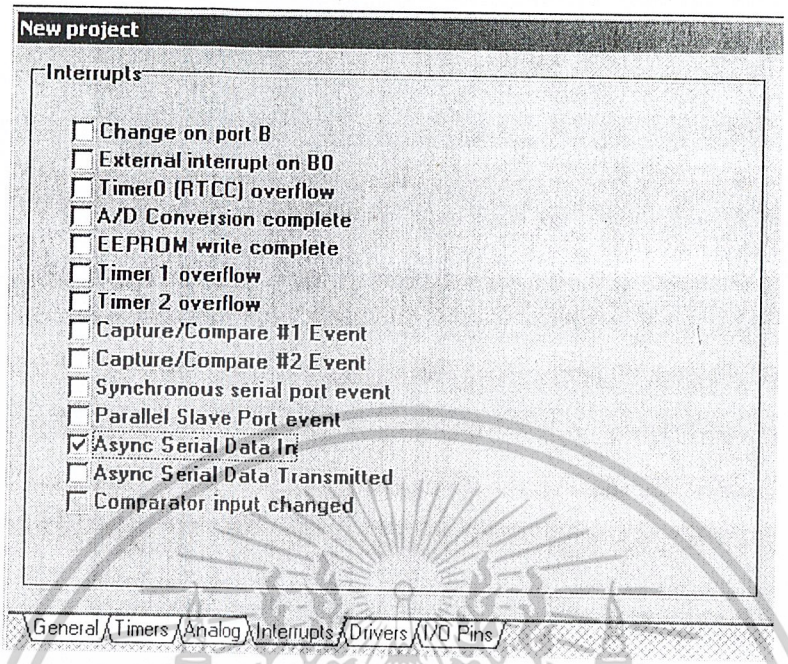
4. ตั้งชื่อ Project เป็น Work1.c ที่ แท็ป General เลือก Device : 16F877, Oscilator Frequency : 20,000,000 Hz (เฮิร์ต), Osc Type : HS, คลิกที่ Use RS-232 และBaud : 9,600



รูปที่ 1.7 การกำหนดค่าในแท็ป General

5. คลิกที่แท็ป Interrupts เลือก Async Serial Data In คือ ให้มีการอินเตอร์รัพต์เมื่อมีข้อมูลเข้ามายังพอร์ตอนุกรม คลิกปุ่ม OK

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.8 การกำหนดค่าในแท็ป Interrupts

6. จากนั้นจะได้โปรแกรมตามด้านล่างนี้

```
#include <WORK1.H>
#int_rda
rda_isr() {
            // Async Serial Data In
}
main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
      enable_interrupts(INT_RDA);
      enable_interrupts(GLOBAL);
}
```

7. ให้ทำการเขียนโปรแกรมเพิ่มเติมตามดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <WORK1.H>
byte tmp;

#int_rda
rda_isr() {
   tmp=getc(); //Read_Data
}

main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
      enable_interrupts(INT_RDA);
      enable_interrupts(GLOBAL);
   set_tris_b(0x00);
   set_tris_c(0x9c);
   bit_clear(port_c,5);
while(1){
Port_b=tmp; //Show_Data
}
}
```

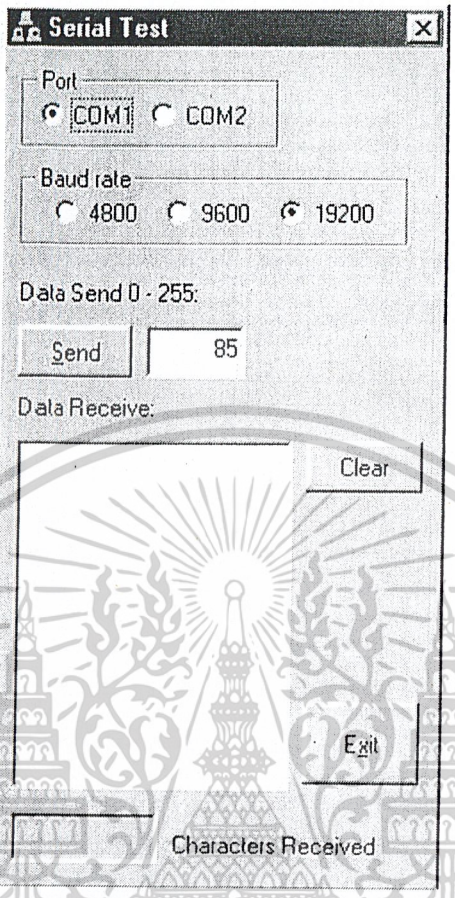
8. ทำการ Compile โปรแกรม โดยคลิกที่ Compile > PCM

9. นำ File Work1.hex ที่ได้เขียนลงไปยัง IC PIC 16F877A

10. ทำการต่อแหล่งจ่ายไฟเข้ากับ ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

11. ทำการเปิดโปรแกรม Serial Test โดยเลือกที่ Port เชื่อมต่อและเลือกอัตรา Baud Rate = 9,600 b/s (บิตต่อวินาที)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.9 โปรแกรม Serial Test

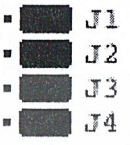
12. ทำการป้อนค่าในช่อง Send Data แล้วคลิกที่ปุ่ม Send ตามตารางข้างล่างและบันทึกผลการทดลองที่แสดงจาก LED ลงในตารางที่ 1.1

ตารางที่ 1.1 ตารางบันทึกผลการทดลอง RS-232

ลำดับที่	ข้อมูลที่ส่ง	ข้อมูลที่อ่านได้จาก LED
1	85	
2	170	
3	15	
4	240	
5	60	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

13. ทำการป้อนค่าในช่อง Send Data แล้วคลิกที่ปุ่ม Send ตามตารางข้างล่างและบันทึกผลการทดลองที่แสดงจาก LED ลงในตารางที่ 1.2 โดยเปลี่ยนไปใช้ชุดรับและส่ง RS-422 และมีการเซตตัวจัมเปอร์ ดังรูปที่ 1.10

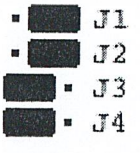


รูปที่ 1.10 การเซตจัมเปอร์เมื่อใช้วงจร RS-422

ตารางที่ 1.2 ตารางบันทึกผลการทดลอง RS-422

ลำดับที่	ข้อมูลที่ส่ง	ข้อมูลที่อ่านได้จาก LED
1	85	
2	170	
3	15	
4	240	
5	60	

14. ทำการป้อนค่าในช่อง Send Data แล้วคลิกที่ปุ่ม Send ตามตารางข้างล่างและบันทึกผลการทดลองที่แสดงจาก LED ลงในตารางที่ 1.3โดยเปลี่ยนไปใช้ชุดรับและส่ง RS-485 โดยมีการเซตตัวจัมเปอร์ ดังรูปที่ 1.11



รูปที่ 1.11 การเซตจัมเปอร์เมื่อใช้วงจร RS-485

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 1.3 ตารางบันทึกผลการทดลอง RS-485

ลำดับที่	ข้อมูลที่ส่ง	ข้อมูลที่อ่านได้จาก LED
1	85	
2	170	
3	15	
4	240	
5	60	

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการการรับข้อมูลจากพอร์ตอนุกรมของ PIC 16F877A มาพอสังเขป

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับข้อมูลจากพอร์ตอนุกรม โดยใช้ Baud rate = 19,200 b/s หากข้อมูลที่รับเข้ามาเท่ากับ 1 ให้ LED วิ่งไปทางซ้าย และหากข้อมูลที่รับเข้ามาเท่ากับ 2 ให้ LED วิ่งไปทางขวา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 2
การรับข้อมูลจากพอร์ต USB

วัตถุประสงค์เชิงพฤติกรรม

เพื่อให้นักศึกษาสามารถ

1. อธิบายหลักการการรับข้อมูลจากพอร์ต USB โดยผ่านชิป FT245BM ได้
2. เขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับข้อมูลจากพอร์ต USB ได้
3. สรุปและวิจารณ์ผลการทดลองได้

เครื่องมือและอุปกรณ์

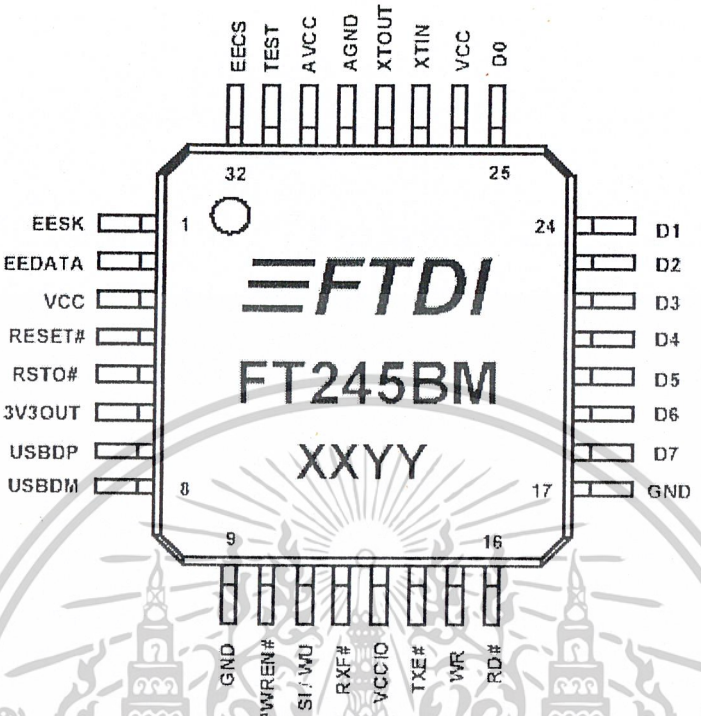
1. ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A 1 ชุด
2. คอมพิวเตอร์ที่ติดตั้งโปรแกรม Usb test 1 เครื่อง

ทฤษฎีเบื้องต้น

1. วงจรติดต่อพอร์ต USB

วงจรติดต่อพอร์ต USB ของชุดสาธิตมาตรฐาน USB โดย PIC 16F877A นั้นใช้ชิปสำเร็จรูปซึ่งเรียกว่า USB โมดูล ซึ่งหัวใจหลักที่สำคัญก็คือชิป FT245BM ดังรูป 1.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



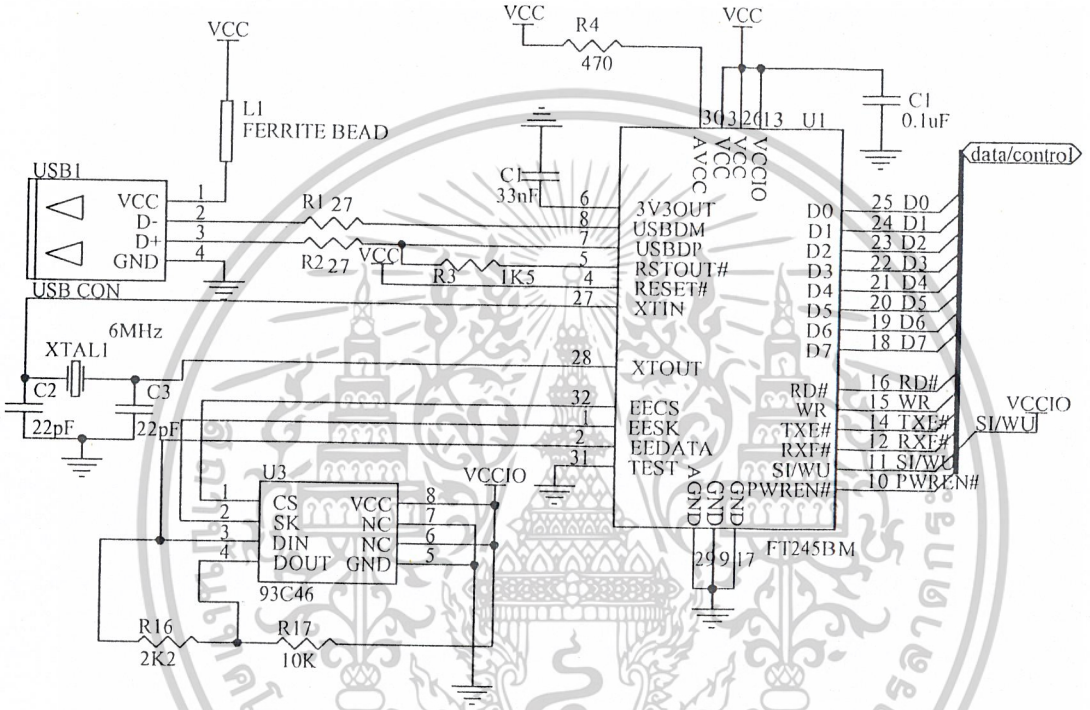
รูปที่ 1.12 ชิป FT245BM

รายละเอียดทางด้านฮาร์ดแวร์

1. มีอัตราการรับและส่งข้อมูล 1MByte / Sec เมื่อใช้ไดเวอร์ D2XX Drivers
2. มีอัตราการรับและส่ง 3,000 Baud เมื่อใช้ไดเวอร์ VCP Drivers
3. มีส่วนจัดการ USB protocol อยู่ภายในชิปโดยไม่ต้องมีการเขียนโปรแกรมเพิ่มเติม
4. มี FIFO Tx buffer 384 Byte / FIFO Rx Buffer 128 Byte
5. สนับสนุน USB Suspend / Resume ผ่านขา PWREN# และ WAKEUP
6. สนับสนุนการควบคุมอุปกรณ์พลังงานสูงโดยขา PWREN#
7. มีวงจร Regulator 3.3V อยู่ภายในสำหรับ USB IO และมีวงจร Power-On-Reset อยู่ภายใน
8. มีส่วนของ clock multiplier PLL 6MHz – 48MHz
9. สามารถใช้โหมด USB Bulk หรือ Isocronous
10. ในโหมด Bit-Bang สามารถใช้เป็น Data bus 8 bit โดยไม่ต้องต่อกับ MCU หรือวงจรลอจิกอื่นๆ
11. ใช้แหล่งจ่ายไฟ 4.4V - 5.25V ในการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

12. สนับสนุนการเชื่อมต่อ USB 1.1 และ USB 2.0 (เฉพาะโหมด Full Speed)
13. ส่วนของUSB VID, PID , Serial Number และProduct Description strings สามารถเก็บไว้ใน EEPROM ภายนอก
14. เป็นแบบ Compact 32LD PQFP package



รูปที่ 1.13 วงจรส่วนติดต่อ USB

จากรูปที่ 2.2 เป็นวงจรการต่อใช้งานชิปFT245BM ซึ่งมีรายละเอียดขาที่สำคัญดังนี้

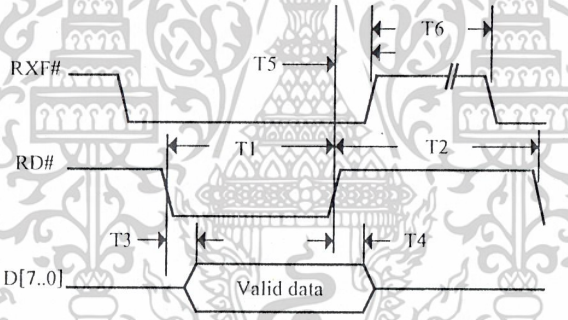
1. D0-D7 เป็นขาข้อมูลมีทั้งหมด 8 บิต
2. RD# เป็นขาที่ใช้ในการอ่านข้อมูลจากชิป FT245BM
3. WR เป็นขาที่ใช้ในการเขียนข้อมูลจากชิป FT245BM
4. TXE# เมื่อขานี้เป็นลอจิก 0 แสดงว่าชิป FT245BM พร้อมที่จะรับข้อมูลเข้ามา
5. RXF# เมื่อขานี้เป็นลอจิก 0 แสดงว่าชิป FT245BM ได้ข้อมูลจากพอร์ต USB
6. PWREN# ขานี้จะเป็นลอจิก 0 ก็ต่อเมื่อมีการ Configured เสร็จ ดังนั้นสามารถที่ต่อ เข้ากับ MOSFET แบบ P-Channel เพื่อควบคุมเปิดการใช้พลังงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.1 ตารางแสดงการเชื่อมต่อระหว่าง FT245BM กับ 16F877A

FT245BM	PIC 16F877A
D0-D7	Port_D (RD0-RD7)
RD#	RC0
WR	RC1
TXE#	RC2
RXF#	RC3
PWREN#	RC4

แผนผังเวลาการอ่านข้อมูลของ FT245BM



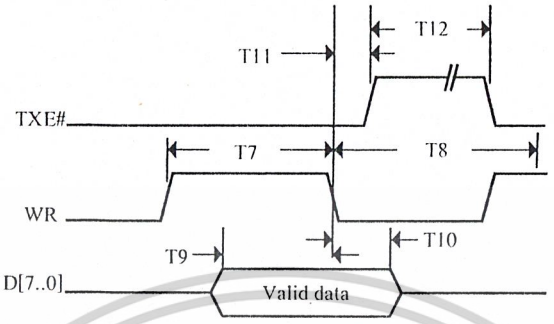
รูปที่ 1.14 แผนผังเวลาการอ่านข้อมูล

ตารางที่ 2.2 ตารางแสดงเวลาในช่วงต่างๆ

Time	Description	Min	Max	Unit
T1	RD Active Pule Wide	50		ns
T2	RD to RD Pre-Charge Time	50		ns
T3	RD Active to Valid Data		30	ns
T4	Valid Data Hold Time from RD Inactive	10		ns
T5	RD Inactive to RXF#	5	25	ns
T6	RXF inactive after RD cycle	80		ns

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แผนผังเวลาการเขียนข้อมูลของ FT245BM



รูปที่ 1.15 แผนผังเวลาการอ่านข้อมูล

ตารางที่ 2.3 ตารางแสดงเวลาในช่วงต่างๆ

Time	Description	Min	Max	Unit
T7	WR Active Pule Wide	50		ns
T8	WR to WR Pre-Charge Time	50		ns
T9	Data Setup Time before WR inactive		20	ns
T10	Data Hold Time from WR inactive	10		ns
T11	WR Inactive to TXE#	5	25	ns
T12	TXE inactive after RD cycle	80		ns

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลำดับขั้นการทดลอง

1. เชื่อมต่อชุดสาธิตมาตรฐาน USB โดย PIC 16F877A เข้ากับคอมพิวเตอร์โดยผ่านพอร์ต USB

2. เขียนโปรแกรมภาษา C for Picโดยใช้โปรแกรม PCW โดยสร้าง Project ชื่อ Work2.c

```
#include <WORK2.H>
   byte tmp;
main() {
       setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
       setup_port_a(NO_ANALOGS);
       setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_c(0x9c);
   set_tris_b(0x00);
   set_tris_d(0xff);

   while(1){
     while((Port_c&0x08)==0x00)
     {
       bit_clear(port_c,0);
       tmp=port_d; //Read Data
       bit_set(port_c,0);
     }
     port_b=tmp; //Show data
   }
}
```

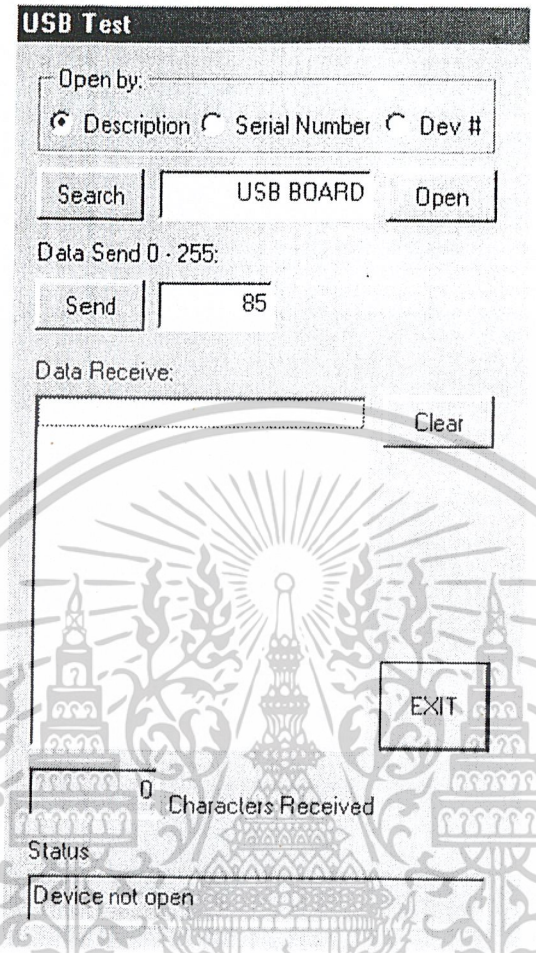
3. เปิดโปรแกรม USB Test

4. ให้ทำการเซตค่าดังนี้

   - ที่ Open by: ให้เลือก Description
   - คลิก Search เลือก USB BOARD
   - คลิกปุ่ม Open เพื่อทำการเชื่อมต่อกับอุปกรณ์

5. ป้อนข้อมูลที่ต้องการส่งในช่อง Data Send 0 – 255 : ดังรูปที่ 1.16

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.16 โปรแกรม USB Test

6. ทำการส่งข้อมูลโดยคลิกปุ่ม Send พร้อมบันทึกผลการทดลองจาก LED ในตารางที่ 2.4

ตารางที่ 2.4 ตารางบันทึกผลการทดลองรับข้อมูลจากพอร์ต USB

ลำดับที่	ข้อมูลที่ส่ง	ข้อมูลที่อ่านได้จาก LED
1	85	
2	170	
3	15	
4	240	
5	60	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการการรับข้อมูลจากพอร์ต USB โดยผ่านชิป FT245BM มาพอสังเขป

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับข้อมูลจากพอร์ต USB หากข้อมูลที่รับเข้ามาเท่ากับ 1 ให้ LED วิ่งไปทางซ้าย และหากข้อมูลที่รับเข้ามาเท่ากับ 2 ให้ LED วิ่งไปทางขวา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 3
การส่งข้อมูลไปยังพอร์ตอนุกรม

วัตถุประสงค์เชิงพฤติกรรม

เพื่อให้นักศึกษาสามารถ

1. อธิบายหลักการการส่งข้อมูลไปยังพอร์ตอนุกรมของ PIC 16F877A ได้
2. เขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อส่งข้อมูลไปยังพอร์ตอนุกรมได้
3. สรุปและวิจารณ์ผลการทดลองได้

เครื่องมือและอุปกรณ์

1. ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A	1 ชุด
2. คอมพิวเตอร์ที่ติดตั้งโปรแกรม Serial Test	1 เครื่อง

ทฤษฎีเบื้องต้น

คำสั่งที่ใช้สำหรับการส่งข้อมูลคือ

```
putc(argument list);
```

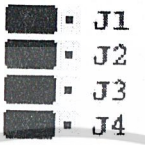
argument list คือ ค่าคงที่ หรือตัวแปร หรือนิพจน์ เช่น tmp (tmp เป็นชื่อตัวแปร) ตัวอย่างเช่น

putc(tmp); ทำการส่งข้อมูลที่อยู่ในตัวแปร tmp

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลำดับขั้นการทดลอง

1. เชื่อมต่อชุดสาธิตมาตรฐาน USB โดย PIC 16F877A เข้ากับคอมพิวเตอร์โดยใช้ชุด RS-232 และเซตจัมเปอร์ ดังรูปที่ 1.17



รูปที่ 1.17 การเซตจัมเปอร์เมื่อใช้วงจร RS-232

2. เขียนโปรแกรมภาษา C for Pic โดยใช้โปรแกรม PCW โดยสร้างมีอัตรา Baud Rate = 9,600 b/s

```
#include <WORK3.H>
   byte count;
main() {

       setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
       setup_port_a(NO_ANALOGS);
       setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_b(0x00);
   set_tris_c(0x9c);
   bit_set(port_c,5);
   while(1){
      for(count=0;count<128;count++){
      putc(count); //Send Data to Serial Port
      port_b=count; //Show Data
      delay_ms(1000);
      }
   }
}
```

3. ทำการเปิดโปรแกรม Serial Test และเลือก Port ที่เชื่อมต่อ อัตรา Baud Rate = 9,600 b/s

4. คลิกปุ่ม Receive

5. ทำการต่อแหล่งจ่ายไฟเข้ากับชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. สังเกตในช่อง Data Receive และบันทึกผลการทดลองที่ได้

.....

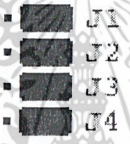
.....

.....

.....

.....

7. ทำการทดลองซ้ำอีกครั้งในข้อที่ 3-5 โดยเปลี่ยนไปใช้ชุดรับและส่ง RS-422 โดยมีการเซตจัมเปอร์ ดังรูปที่ 1.18



รูปที่ 1.18 การเซตจัมเปอร์เมื่อใช้วงจร RS-422

8. สังเกตในช่อง Data Receive และบันทึกผลการทดลองที่ได้

.....

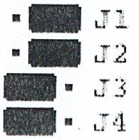
.....

.....

.....

.....

9. ทำการทดลองซ้ำอีกครั้งในข้อที่ 3-5 โดยเปลี่ยนไปใช้ชุดรับและส่ง RS-485 โดยมีการเซตจัมเปอร์ ดังรูปที่ 1.19



รูปที่ 1.19 การเซตจัมเปอร์เมื่อใช้วงจร RS-485

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

10. สังเกตในช่อง Data Receive และบันทึกผลการทดลองที่ได้

.....

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการการส่งข้อมูลไปยังพอร์ตอนุกรมของ PIC 16F877A มาพอสังเขป

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อส่งข้อมูล 127 โดยลดค่าลงที่ละ 1 จนถึงค่า 1 ไปยังพอร์ตอนุกรมโดยใช้อัตราบอดเรต (Baud Rate) 19,200 b/s

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 4
การส่งข้อมูลไปยังพอร์ต USB

วัตถุประสงค์เชิงพฤติกรรม

เพื่อให้นักศึกษาสามารถ

1. อธิบายหลักการการส่งข้อมูลไปยังพอร์ต USB โดยผ่านชิป FT245BM ได้
2. เขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อส่งข้อมูลไปยังพอร์ต USB ได้
3. สรุปและวิจารณ์ผลการทดลองได้

เครื่องมือและอุปกรณ์

1. ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A	1 ชุด
2. คอมพิวเตอร์ที่ติดตั้งโปรแกรม USB Test	1 เครื่อง

ทฤษฎีเบื้องต้น

จากทฤษฎีเบื้องต้นในใบงานที่ 2 เมื่อต้องการที่จะส่งข้อมูลไปยังคอมพิวเตอร์โดยผ่านพอร์ต USB จะต้องเช็คดูที่ขา TXE# ของชิป FT248BM มีลอจิก “0” หรือไม่ถ้าหากเป็นลอจิก “0” ก็สามารถทำการส่งข้อมูลไปยังพอร์ต USB ได้

ลำดับขั้นการทดลอง

1. เชื่อมต่อชุดสาธิตมาตรฐาน USB โดย PIC 16F877A เข้ากับคอมพิวเตอร์โดยผ่านพอร์ต USB
2. เขียนโปรแกรมภาษา C for Pic โดยใช้โปรแกรม PCW โดยสร้าง Project ชื่อ Work4.c

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <WORK4.H>
byte count;
main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_c(0x9c);
   set_tris_d(0x00);
   set_tris_b(0x00);
   while(1){
      for(count=0;count<128;count++){
         while((Port_c&0x04)==0x04);
            bit_set(port_c,1);
            port_d=count; //Send_Data to USB Module
            bit_clear(port_c,1);
            port_b=count; //Show_data
            delay_ms(500);
       }
   }
}
```

3. เปิดโปรแกรม USB Test

4. ให้ทำการเซตค่าดังนี้

   - ที่ Open by: ให้เลือก Description

   - คลิก Search เลือก USB BOARD

   - คลิกปุ่ม Open เพื่อทำการเชื่อมต่อกับอุปกรณ์

6. สังเกตในช่อง Data Receive และบันทึกผลการทดลองที่ได้

.....

.....

.....

.....

.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการการส่งข้อมูลไปยังพอร์ต USB โดยผ่านชิป FT248BM มาพอสังเขป

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อส่งข้อมูล 127 โดยลดค่าลงที่ละ 1 จนถึงค่า 1 ไปยังพอร์ต USB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 5

การรับและส่งข้อมูลระหว่างพอร์ต USB กับ พอร์ตอนุกรม

วัตถุประสงค์เชิงพฤติกรรม

เพื่อให้นักศึกษาสามารถ

1. อธิบายหลักการการรับและส่งข้อมูลระหว่างพอร์ต USB กับ พอร์ตอนุกรมได้

2. เขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับและส่งข้อมูลระหว่างพอร์ต USB กับ พอร์ตอนุกรมได้

3. สรุปและวิจารณ์ผลการทดลองได้

เครื่องมือและอุปกรณ์

1. ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A 1 ชุด
2. คอมพิวเตอร์ที่ติดตั้งโปรแกรม USB Test และ Serial Test 2 เครื่อง

ทฤษฎีเบื้องต้น

จากที่ได้ทำการทดลองในใบงานที่ 1-4 มาแล้ว ก็ได้ทราบถึงการเขียนโปรแกรมเพื่อที่จะควบคุม PIC 16F877A ให้ส่งหรือรับข้อมูล ผ่านพอร์ตอนุกรมและพอร์ต USB มาแล้ว ในใบงานที่ 5 นี้จะเป็นการประยุกต์ใช้ความรู้ที่ได้จากการทำใบงานที่ผ่านมา เพื่อเขียนโปรแกรมควบคุม PIC 16F877A ให้ทำหน้าที่รับและส่งข้อมูลระหว่างพอร์ตอนุกรมกับพอร์ต USB

ลำดับขั้นการทดลอง

1. เชื่อมต่อชุดสาธิตมาตรฐาน USB โดย PIC 16F877A เข้ากับคอมพิวเตอร์โดยเครื่องที่ 1 ต่อกับพอร์ต USB และเครื่องที่ 2 ต่อกับพอร์ต RS-232

2. เขียนโปรแกรมภาษา C for Pic โดยใช้โปรแกรม PCW โดยสร้าง Project ชื่อ Work5.c

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <WORK5.H>
unsigned byte tmp,tmp1;
#int_rda
   rda_isr() {
                                    // Async Serial Data In
   set_tris_d(0x00);                //Portd = output
   tmp=getc();                      //Read_data
   port_d=tmp;                       //Read_data
   while((Port_c&0x04)==0x04);      //chack TXE#
   bit_set(port_c,1);               // WR = 1
   delay_us(1);
   port_b=tmp;                      //show_data
   bit_clear(port_c,1);             // WR = 0
}

main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
      enable_interrupts(INT_RDA);
      enable_interrupts(GLOBAL);
   set_tris_b(0x00);                //Portb = output
   set_tris_c(0x9c);                //Portc = in/output
   set_tris_d(0xff);                //Portd = input
   bit_clear(port_c,5);          //rs485 rx
   bit_clear(port_c,1);          //WR = 0
   bit_set(port_c,0);            //RD# = 1

  while(1){
     while((Port_c&0x08)==0x00) //chack RXF#
     {
        set_tris_d(0xff);          //Portd = input
        bit_set(port_c,5);         //rs485 tx
        bit_clear(port_c,0);       //RD# = 0
        tmp1=port_d;
        putc(tmp1);                // Send_data to Serial
        delay_ms(4);
        port_b=tmp1;              //show_data
        bit_clear(port_c,5);      //rs485 rx
        bit_set(port_c,0);        //RD# = 1
     }
  }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. เปิดโปรแกรม USB Test

4. ให้ทำการเซตค่าดังนี้

- ที่ Open by: ให้เลือก Description
- คลิก Search เลือก USB BOARD
- คลิกปุ่ม Open เพื่อทำการเชื่อมต่อกับอุปกรณ์

5. คอมพิวเตอร์เครื่องที่ 2 เปิดโปรแกรม Serial Test

6. ให้ทำการเซตค่าดังนี้

- เลือก Port ที่เชื่อมต่อ
- อัตรา Baud Rate = 9,600 b/s
- คลิกปุ่ม Receive

7. ทำการต่อแหล่งจ่ายไฟเข้ากับชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

8. ทดลองส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 1 ผ่านพอร์ต USB ไปยังเครื่องที่ 2 ซึ่งต่ออยู่กับพอร์ต RS-232 และบันทึกผลที่ได้ลงตารางที่ 5.1

ตารางที่ 5.1 ตารางบันทึกผลการส่งข้อมูลผ่านพอร์ต USB

ลำดับที่	ข้อมูลที่ส่ง (เครื่องที่ 1)	ข้อมูลที่รับ (เครื่องที่ 2)
1	85	
2	170	
3	15	
4	240	
5	60	

9. ทดลองส่งข้อมูลจากคอมพิวเตอร์เครื่องที่ 2 ผ่าน พอร์ต RS-232 ไปยังเครื่องที่ 1 ซึ่งต่ออยู่กับพอร์ต USB และบันทึกผลการทดลองที่ได้ลงตารางที่ 5.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.2 ตารางบันทึกผลการส่งข้อมูลผ่านพอร์ต RS-232

ลำดับที่	ข้อมูลที่ส่ง (เครื่องที่ 2)	ข้อมูลที่รับ (เครื่องที่ 1)
1	85	
2	170	
3	15	
4	240	
5	60	

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการการรับและส่งข้อมูลระหว่างพอร์ต USB กับ พอร์ตอนุกรมมาพอสังเขป

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับและส่งข้อมูลระหว่างพอร์ต USB กับ พอร์ตอนุกรม โดยให้มีอัตราบอดเรต (Baud Rate) 19,200 b/s

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

เฉลยใบงาน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เฉลยใบงานที่ 1

1. จงอธิบายหลักการการรับข้อมูลจากพอร์ตอนุกรมของ PIC 16F877A มาพอสังเขป

ตอบ เมื่อมีข้อมูลเข้ามายังพอร์ตอนุกรม ก็จะทำให้เกิด Interrupt ซึ่งในโปรแกรมบริการ Interrupt จะเป็นการอ่านข้อมูลที่รับเข้า ด้วยคำสั่ง getc(); แล้วแสดงผลไปยัง LED ต่อไป

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับข้อมูลจากพอร์ตอนุกรม โดยใช้ Baud rate = 19,200 b/s หากข้อมูลที่รับเข้ามาเท่ากับ 1 ให้ LED วิ่งไปทางซ้าย และหากข้อมูลที่รับเข้ามาเท่ากับ 2 ให้ LED วิ่งไปทางขวา

ตอบ

โปรแกรม work1_1.c

```
#include <WORK1_1.H>
byte tmp,tmp1;

#int_rda
rda_isr() {
   tmp=getc(); //Read Data
}

main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
      enable_interrupts(INT_RDA);
      enable_interrupts(GLOBAL);
   set_tris_b(0x00);
   set_tris_c(0x9c);
   bit_clear(port_c,5);

   tmp1=0x01;
while(1){
   Port_b=tmp1;
   switch(tmp)
   {
      case 1:rotate_left(&tmp1, 1);
             break;
      case 2:rotate_right(&tmp1, 1);
   }
   delay_ms(500);
}
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม work1_1.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#use rs232(baud=19200,xmit=PIN_C6,rcv=PIN_C7)

//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
#define T1_EXTERNAL_SYNC    3
#define T1_CLK_OUT          8
```

เอกสารนี้จัดทำขึ้นสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define T1_DIV_BY_1          0
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte   TIMER_1_LOW=         0x0e
#byte   TIMER_1_HIGH=        0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte   TIMER_2=             0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                          0
#define CCP_CAPTURE_FE                   4
#define CCP_CAPTURE_RE                   5
#define CCP_CAPTURE_DIV_4                6
#define CCP_CAPTURE_DIV_16               7
#define CCP_COMPARE_SET_ON_MATCH         8
#define CCP_COMPARE_CLR_ON_MATCH         9
#define CCP_COMPARE_INT                  0xA
#define CCP_COMPARE_RESET_TIMER          0xB
#define CCP_PWM                          0xC
#define CCP_PWM_PLUS_1                   0x1c
#define CCP_PWM_PLUS_2                   0x2c
#define CCP_PWM_PLUS_3                   0x3c
#byte   CCP_1_LOW=                       0x15
#byte   CCP_1_HIGH=                      0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP2()
#byte   CCP_2_LOW=                       0x1B
#byte   CCP_2_HIGH=                      0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                      0x10
#define PSP_DISABLED                     0
#byte   PSP_DATA=                        8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER       0x20
#define SPI_SLAVE        0x24
#define SPI_L_TO_H       0
#define SPI_H_TO_L       0x10
#define SPI_CLK_DIV_4    0
#define SPI_CLK_DIV_16   1
#define SPI_CLK_DIV_64   2
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define SPI_CLK_T2        3
#define SPI_SS_DISABLED   1
#define INT_SSP                0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS


#define INT_RDA                0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG              0
#define ANALOG_RA3_REF          1
#define A_ANALOG                2
#define A_ANALOG_RA3_REF        3
#define RA0_RA1_RA3_ANALOG      4
#define RA0_RA1_ANALOG_RA3_REF  5
#define NO_ANALOGS              6
#define ADC_OFF                 0
#define ADC_CLOCK_DIV_2         1
#define ADC_CLOCK_DIV_8         0x41
#define ADC_CLOCK_DIV_32        0x81
#define ADC_CLOCK_INTERNAL      0xc1

#define ADC_DONE               0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```

เฉลยใบงานที่ 2

1. จงอธิบายหลักการการรับข้อมูลจากพอร์ต USB โดยผ่านชิป FT245BM มาพอสังเขป

ตอบ ทำการตรวจสอบที่ขา RXF# ว่าเป็น 0 หรือไม่ หากเป็น 0 ก็จะทำการอ่านข้อมูลเข้ามาแล้วทำการแสดงผลไปยัง LED

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับข้อมูลจากพอร์ต USB หากข้อมูลที่รับเข้ามาเท่ากับ 1 ให้ LED วิ่งไปทางซ้าย และหากข้อมูลที่รับเข้ามาเท่ากับ 2 ให้ LED วิ่งไปทางขวา

ตอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม work2_1.c

```
#include <WORK2_1.H>
   byte tmp,tmp1;
main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_c(0x9c);
   set_tris_b(0x00);
   set_tris_d(0xff);
   tmp1=0x01;

   while(1){
      if((Port_c&0x08)==0x00)
      {
          bit_clear(port_c,0);
          tmp=port_d;
          bit_set(port_c,0);
      }
      Port_b=tmp1;
      switch(tmp)
      {
         case 1:rotate_left(&tmp1, 1);
            break;
         case 2:rotate_right(&tmp1, 1);
      }
      delay_ms(500);
   }
}
```

โปรแกรม work2_1.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

```
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define RTCC_DIV_16      3
#define RTCC_DIV_32      4
#define RTCC_DIV_64      5
#define RTCC_DIV_128     6
#define RTCC_DIV_256     7
#define WDT_18MS         8
#define WDT_36MS         9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H              0x40
#define H_TO_L                0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
#define T1_EXTERNAL_SYNC    3
#define T1_CLK_OUT          8
#define T1_DIV_BY_1         0
#define T1_DIV_BY_2         0x10
#define T1_DIV_BY_4         0x20
#define T1_DIV_BY_8         0x30
#byte   TIMER_1_LOW=        0x0e
#byte   TIMER_1_HIGH=       0x0f
#define T2_DISABLED         0
#define T2_DIV_BY_1         4
#define T2_DIV_BY_4         5
#define T2_DIV_BY_16        6
#byte   TIMER_2=            0x11

#define INT_TIMER1          0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2          0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                         0
#define CCP_CAPTURE_FE                  4
#define CCP_CAPTURE_RE                  5
#define CCP_CAPTURE_DIV_4               6
#define CCP_CAPTURE_DIV_16              7
#define CCP_COMPARE_SET_ON_MATCH        8
#define CCP_COMPARE_CLR_ON_MATCH        9
#define CCP_COMPARE_INT                 0xA
```

เอกสารนี้จัดทำขึ้นเพื่อการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define CCP_COMPARE_RESET_TIMER          0xB
#define CCP_PWM                          0xC
#define CCP_PWM_PLUS_1                   0x1c
#define CCP_PWM_PLUS_2                   0x2c
#define CCP_PWM_PLUS_3                   0x3c
#byte    CCP_1_LOW=                      0x15
#byte    CCP_1_HIGH=                     0x16

#define INT_CCP1              0x8C04     // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP2()
#byte    CCP_2_LOW=                      0x1B
#byte    CCP_2_HIGH=                     0x1C
#define INT_CCP2              0x8D01     // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                      0x10
#define PSP_DISABLED                     0
#byte    PSP_DATA=                       8
#define INT_PSP               0x8C80     // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER       0x20
#define SPI_SLAVE        0x24
#define SPI_L_TO_H       0
#define SPI_H_TO_L       0x10
#define SPI_CLK_DIV_4    0
#define SPI_CLK_DIV_16   1
#define SPI_CLK_DIV_64   2
#define SPI_CLK_T2       3
#define SPI_SS_DISABLED  1
#define INT_SSP               0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS


#define INT_RDA               0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE               0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG                0
#define ANALOG_RA3_REF            1
#define A_ANALOG                  2
#define A_ANALOG_RA3_REF          3
#define RA0_RA1_RA3_ANALOG        4
#define RA0_RA1_ANALOG_RA3_REF    5
#define NO_ANALOGS                6
#define ADC_OFF                   0
#define ADC_CLOCK_DIV_2           1
#define ADC_CLOCK_DIV_8       0x41
#define ADC_CLOCK_DIV_32      0x81
#define ADC_CLOCK_INTERNAL    0xc1

#define ADC_DONE              0x8C40     // Used for ENABLE/DISABLE
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
INTERRUPTS

#list
```

เฉลยใบงานที่ 3

1. จงอธิบายหลักการการส่งข้อมูลไปยังพอร์ตอนุกรมของ PIC 16F877A มาพอสังเขป

ตอบ ทำการส่งข้อมูลด้วยคำสั่ง putc(); โดยต้องมีการกำหนดอัตราบอด ที่ใช้ในการส่งด้วย

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อส่งข้อมูล 127 โดยลดค่าลงที่ละ 1 จนถึงค่า 1 ไปยังพอร์ตอนุกรมโดยใช้อัตราบอดเรต (Baud Rate) 19,200 b/s

ตอบ

โปรแกรม work3_1.c

```
#include <WORK3_1.H>
   byte count;
main() {

       setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
       setup_port_a(NO_ANALOGS);
       setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_b(0x00);
   set_tris_c(0x9c);
   bit_set(port_c,5);
   while(1){
       for(count=127;count>0;count--){
       putc(count);
       port_b=count;
       delay_ms(500);
       }
   }
}
```

โปรแกรม work3_1.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

#use rs232(baud=19200,xmit=PIN_C6,rcv=PIN_C7)

////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO       0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE       0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT         0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL          0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED     0
#define T1_INTERNAL     5
#define T1_EXTERNAL     7
#define T1_EXTERNAL_SYNC 3
#define T1_CLK_OUT      8
#define T1_DIV_BY_1     0
#define T1_DIV_BY_2     0x10
#define T1_DIV_BY_4     0x20
#define T1_DIV_BY_8     0x30
#byte   TIMER_1_LOW=    0x0e
#byte   TIMER_1_HIGH=   0x0f
#define T2_DISABLED     0
#define T2_DIV_BY_1     4
#define T2_DIV_BY_4     5
#define T2_DIV_BY_16    6
#byte   TIMER_2=        0x11

#define INT_TIMER1      0x8C01    // Used for ENABLE/DISABLE
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
INTERRUPTS
#define INT_TIMER2              0x8C02     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                                 0
#define CCP_CAPTURE_FE                          4
#define CCP_CAPTURE_RE                          5
#define CCP_CAPTURE_DIV_4                       6
#define CCP_CAPTURE_DIV_16                      7
#define CCP_COMPARE_SET_ON_MATCH                8
#define CCP_COMPARE_CLR_ON_MATCH                9
#define CCP_COMPARE_INT                         0xA
#define CCP_COMPARE_RESET_TIMER                 0xB
#define CCP_PWM                                 0xC
#define CCP_PWM_PLUS_1                          0x1c
#define CCP_PWM_PLUS_2                          0x2c
#define CCP_PWM_PLUS_3                          0x3c
#byte    CCP_1_LOW=                             0x15
#byte    CCP_1_HIGH=                            0x16

#define INT_CCP1                0x8C04     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used for SETUP_CCP2()
#byte    CCP_2_LOW=                             0x1B
#byte    CCP_2_HIGH=                            0x1C
#define INT_CCP2                0x8D01     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                             0x10
#define PSP_DISABLED                            0
#byte    PSP_DATA=                              8
#define INT_PSP                 0x8C80     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER      0x20
#define SPI_SLAVE       0x24
#define SPI_L_TO_H      0
#define SPI_H_TO_L      0x10
#define SPI_CLK_DIV_4   0
#define SPI_CLK_DIV_16  1
#define SPI_CLK_DIV_64  2
#define SPI_CLK_T2      3
#define SPI_SS_DISABLED 1
#define INT_SSP                 0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS


#define INT_RDA                 0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                 0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG              0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define ANALOG_RA3_REF          1
#define A_ANALOG                2
#define A_ANALOG_RA3_REF        3
#define RA0_RA1_RA3_ANALOG      4
#define RA0_RA1_ANALOG_RA3_REF  5
#define NO_ANALOGS              6
#define ADC_OFF                 0
#define ADC_CLOCK_DIV_2         1
#define ADC_CLOCK_DIV_8         0x41
#define ADC_CLOCK_DIV_32        0x81
#define ADC_CLOCK_INTERNAL      0xc1

#define ADC_DONE                0x8C40    // Used for ENABLE/DISABLE
INTERRUPTS

#list
```

เฉลยใบงานที่ 4

1. จงอธิบายหลักการการส่งข้อมูลไปยังพอร์ต USB โดยผ่านชิป FT248BM มาพอสังเขป

ตอบ ทำการตรวจสอบที่ขา TXE# ว่าเป็น 0 หรือไม่ หากเป็น 0 ก็จะทำการส่งข้อมูลเข้ามาแล้วทำการแสดงผลไปยัง LED

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อส่งข้อมูล 127 โดยลดค่าลงที่ละ 1 จนถึงค่า 1 ไปยังพอร์ต USB

ตอบ

โปรแกรม work4_1.c

```
#include <WORK4_1.H>
byte count;
main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
   set_tris_c(0x9c);
   set_tris_d(0x00);
   set_tris_b(0x00);
   while(1){
      for(count=127;count>0;count--){
         while((Port_c&0x04)==0x04);
            bit_set(port_c,1);
            port_d=count;
            bit_clear(port_c,1);
            port_b=count;
            delay_ms(500);
      }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
}
```

โปรแกรม work4_1.h

```
#device PIC16F877
#use delay(clock=20000000)
#nolist
////////////////////////////// I/O definitions for INPUT() and
OUTPUT_xxx()
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45
#use fixed_io(A_OUTPUTS=)
#byte port_A=5

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
#use fixed_io(B_OUTPUTS=)
#byte port_B=6

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63
#use fixed_io(C_OUTPUTS=PIN_C6)
#byte port_C=7

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71
#use fixed_io(D_OUTPUTS=)
#byte port_D=8

#define PIN_E0  72
#define PIN_E1  73
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define PIN_E2  74
#use fixed_io(E_OUTPUTS=)
#byte port_E=9

////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H              0x40
#define H_TO_L                 0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
////////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define T1_EXTERNAL_SYNC     3
#define T1_CLK_OUT           8
#define T1_DIV_BY_1          0
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte    TIMER_1_LOW=        0x0e
#byte    TIMER_1_HIGH=       0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte    TIMER_2=            0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                          0
#define CCP_CAPTURE_FE                   4
#define CCP_CAPTURE_RE                   5
#define CCP_CAPTURE_DIV_4                6
#define CCP_CAPTURE_DIV_16               7
#define CCP_COMPARE_SET_ON_MATCH         8
#define CCP_COMPARE_CLR_ON_MATCH         9
#define CCP_COMPARE_INT                  0xA
#define CCP_COMPARE_RESET_TIMER          0xB
#define CCP_PWM                          0xC
#define CCP_PWM_PLUS_1                   0x1c
#define CCP_PWM_PLUS_2                   0x2c
#define CCP_PWM_PLUS_3                   0x3c
#byte    CCP_1_LOW=                      0x15
#byte    CCP_1_HIGH=                     0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP2()
#byte    CCP_2_LOW=                      0x1B
#byte    CCP_2_HIGH=                     0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                        0x10
#define PSP_DISABLED                       0
#byte    PSP_DATA=                         8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER       0x20
#define SPI_SLAVE        0x24
#define SPI_L_TO_H       0
#define SPI_H_TO_L       0x10
#define SPI_CLK_DIV_4    0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define SPI_CLK_DIV_16    1
#define SPI_CLK_DIV_64    2
#define SPI_CLK_T2        3
#define SPI_SS_DISABLED   1
#define INT_SSP                0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS


#define INT_RDA                0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG                0
#define ANALOG_RA3_REF            1
#define A_ANALOG                  2
#define A_ANALOG_RA3_REF          3
#define RA0_RA1_RA3_ANALOG        4
#define RA0_RA1_ANALOG_RA3_REF    5
#define NO_ANALOGS                6
#define ADC_OFF                   0
#define ADC_CLOCK_DIV_2           1
#define ADC_CLOCK_DIV_8           0x41
#define ADC_CLOCK_DIV_32          0x81
#define ADC_CLOCK_INTERNAL        0xc1

#define ADC_DONE               0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```

เฉลยใบงานที่ 5

1. จงอธิบายหลักการการรับและส่งข้อมูลระหว่างพอร์ต USB กับ พอร์ตอนุกรมมาพอสังเขป

ตอบ เมื่อมีข้อมูลเข้ามายังพอร์ตอนุกรมทำให้เกิดการ Interrupt แล้วทำการอ่านข้อมูลเข้ามาแล้วส่งไปยัง USB Module เพื่อส่งออกพอร์ต USB และขณะเดียวกันทำการตรวจสอบที่ขา RXF# ว่าเป็น 0 หรือไม่ หากเป็น 0 ก็จะทำการอ่านข้อมูลจาก USB Module เข้ามาแล้วทำการส่งข้อมูลที่ได้ไปยังพอร์ตอนุกรม

2. จงเขียนโปรแกรมภาษา C for PIC ควบคุม PIC 16F877A เพื่อรับและส่งข้อมูลระหว่างพอร์ต USB กับ พอร์ตอนุกรม โดยให้มีอัตราบอดเรต (Baud Rate) 19,200 b/s

ตอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม work5_1.c

```
#include <WORK5_1.H>
unsigned byte tmp,tmp1;
#int_rda
   rda_isr() {
                                    // Async Serial Data In
   set_tris_d(0x00);                //Portd = output
   tmp=getc();                      //Read_data
   port_d=tmp;                       //Read_data
   while((Port_c&0x04)==0x04);      //chack TXE#
   bit_set(port_c,1);               // WR = 1
   delay_us(1);
   port_b=tmp;                      //show_data
   bit_clear(port_c,1);             // WR = 0
}

main() {
      setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
      setup_port_a(NO_ANALOGS);
      setup_adc(ADC_CLOCK_INTERNAL);
      enable_interrupts(INT_RDA);
      enable_interrupts(GLOBAL);
   set_tris_b(0x00);                //Portb = output
   set_tris_c(0x9c);                //Portc = in/output
   set_tris_d(0xff);                //Portd = input
   bit_clear(port_c,5);            //rs485 rx
   bit_clear(port_c,1);            //WR = 0
   bit_set(port_c,0);              //RD# = 1

   while(1){
      while((Port_c&0x08)==0x00) //chack RXF#
      {
         set_tris_d(0xff);          //Portd = input
         bit_set(port_c,5);         //rs485 tx
         bit_clear(port_c,0);       //RD# = 0
         tmp1=port_d;
         putc(tmp1);                // Send_data to Serial
         delay_ms(4);
         port_b=tmp1;               //show_data
         bit_clear(port_c,5);       //rs485 rx
         bit_set(port_c,0);         //RD# = 1
      }
   }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#use rs232(baud=19200,xmit=PIN_C6,rcv=PIN_C7)

//////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

//////////////////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP  0
#define WDT_TIMEOUT     8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
//////////////////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS        10
#define WDT_144MS       11
#define WDT_288MS       12
#define WDT_576MS       13
#define WDT_1152MS      14
#define WDT_2304MS      15
#define L_TO_H          0x40
#define H_TO_L          0

#define RTCC_ZERO           0x0B20    // Used for ENABLE/DISABLE
INTERRUPTS
#define RB_CHANGE           0x0B08    // Used for ENABLE/DISABLE
INTERRUPTS
#define EXT_INT             0x0B10    // Used for ENABLE/DISABLE
INTERRUPTS

#define GLOBAL              0x0BC0    // Used for ENABLE/DISABLE
INTERRUPTS
//////////////////////////////////////// Constants used for Timer1 and
Timer2
#define T1_DISABLED         0
#define T1_INTERNAL         5
#define T1_EXTERNAL         7
#define T1_EXTERNAL_SYNC    3
#define T1_CLK_OUT          8
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define T1_DIV_BY_1          0
#define T1_DIV_BY_2          0x10
#define T1_DIV_BY_4          0x20
#define T1_DIV_BY_8          0x30
#byte   TIMER_1_LOW=         0x0e
#byte   TIMER_1_HIGH=        0x0f
#define T2_DISABLED          0
#define T2_DIV_BY_1          4
#define T2_DIV_BY_4          5
#define T2_DIV_BY_16         6
#byte   TIMER_2=             0x11

#define INT_TIMER1           0x8C01    // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TIMER2           0x8C02    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP1()
#define CCP_OFF                          0
#define CCP_CAPTURE_FE                   4
#define CCP_CAPTURE_RE                   5
#define CCP_CAPTURE_DIV_4                6
#define CCP_CAPTURE_DIV_16               7
#define CCP_COMPARE_SET_ON_MATCH         8
#define CCP_COMPARE_CLR_ON_MATCH         9
#define CCP_COMPARE_INT                  0xA
#define CCP_COMPARE_RESET_TIMER          0xB
#define CCP_PWM                          0xC
#define CCP_PWM_PLUS_1                   0x1c
#define CCP_PWM_PLUS_2                   0x2c
#define CCP_PWM_PLUS_3                   0x3c
#byte   CCP_1_LOW=                       0x15
#byte   CCP_1_HIGH=                      0x16

#define INT_CCP1             0x8C04    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////// Constants used for SETUP_CCP2()
#byte   CCP_2_LOW=                       0x1B
#byte   CCP_2_HIGH=                      0x1C
#define INT_CCP2             0x8D01    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_PSP()
#define PSP_ENABLED                        0x10
#define PSP_DISABLED                       0
#byte   PSP_DATA=                          8
#define INT_PSP              0x8C80    // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used in SETUP_SSP()
#define SPI_MASTER       0x20
#define SPI_SLAVE        0x24
#define SPI_L_TO_H       0
#define SPI_H_TO_L       0x10
#define SPI_CLK_DIV_4    0
#define SPI_CLK_DIV_16   1
#define SPI_CLK_DIV_64   2
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define SPI_CLK_T2        3
#define SPI_SS_DISABLED   1
#define INT_SSP                0x8C08     // Used for ENABLE/DISABLE
INTERRUPTS


#define INT_RDA                0x8C20     // Used for ENABLE/DISABLE
INTERRUPTS
#define INT_TBE                0x8C10     // Used for ENABLE/DISABLE
INTERRUPTS

//////////////////////////////////// Constants used for SETUP_ADC()
#define ALL_ANALOG              0
#define ANALOG_RA3_REF          1
#define A_ANALOG                2
#define A_ANALOG_RA3_REF        3
#define RA0_RA1_RA3_ANALOG      4
#define RA0_RA1_ANALOG_RA3_REF  5
#define NO_ANALOGS              6
#define ADC_OFF                 0
#define ADC_CLOCK_DIV_2         1
#define ADC_CLOCK_DIV_8         0x41
#define ADC_CLOCK_DIV_32        0x81
#define ADC_CLOCK_INTERNAL      0xc1

#define ADC_DONE               0x8C40     // Used for ENABLE/DISABLE
INTERRUPTS

#list
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาคผนวก ฉ
คู่มือสำหรับผู้ทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คู่มือสำหรับผู้ทดลอง

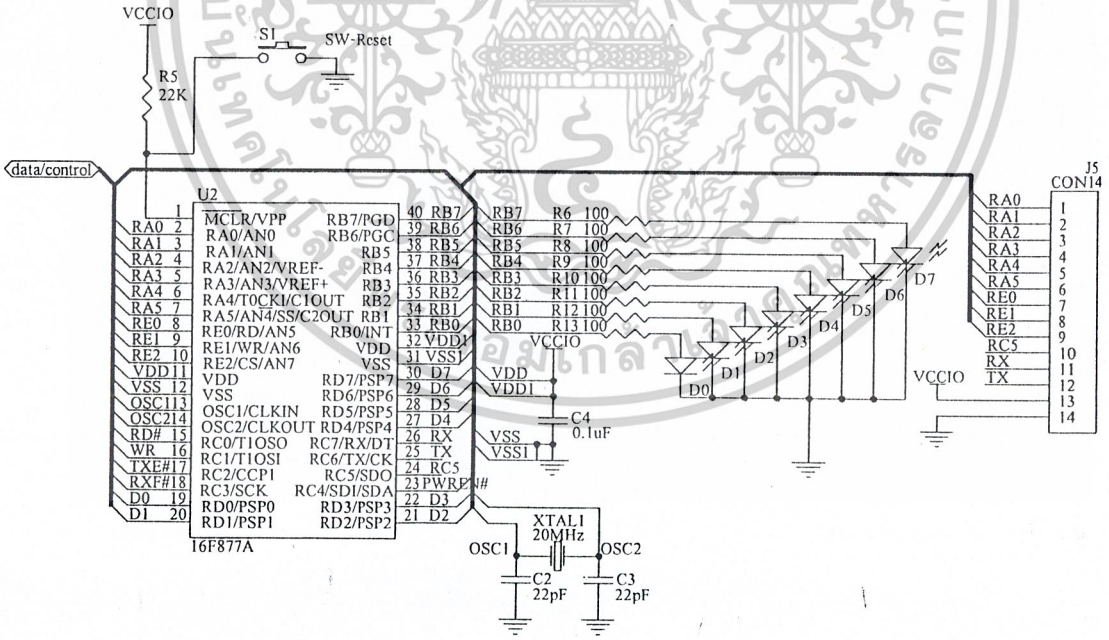
ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

1. วงจรชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

ชุดสาธิตมาตรฐาน USB โดย PIC 16F877A จะมีวงจรส่วนต่างๆ ที่สำคัญได้แก่ ส่วนควบคุม, ส่วนติดต่อพอร์ต USB, พอร์ตอนุกรมRS-232, RS-422 และRS-485 และมีไดโอดเปล่งแสงเชื่อมต่ออยู่กับส่วนควบคุมเพื่อใช้ในการแสดงผล

1.1 วงจรควบคุม

วงจรส่วนนี้เป็นส่วนที่ทำหน้าที่ควบคุมการทำงานทั้งหมดของชุดสาธิต โดยองค์ประกอบที่สำคัญที่สุดของส่วนนี้ก็คือ ไมโครคอนโทรลเลอร์ PIC 16F877A เป็นคอนโทรลเลอร์ขนาด 40 ขา มีพอร์ตใช้งานทั้งหมดอยู่ 5 พอร์ต ได้แก่ พอร์ต A, พอร์ต B, พอร์ต C, พอร์ต D และพอร์ต E จากรูปที่ 1.1 เป็นวงจรส่วนควบคุมซึ่งมีไดโอดเปล่งแสงเชื่อมต่ออยู่กับ พอร์ต B



รูปที่ 1.1 วงจรควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 วงจรติดต่อพอร์ต USB

คุณสมบัติต่างๆของพอร์ต USB มีดังนี้

1. ใช้คอนเน็กเตอร์เพียงชนิดเดียว ซึ่งมี 2 รูปแบบสำหรับเชื่อมต่ออุปกรณ์ทุกๆ ชนิดเข้ากับคอมพิวเตอร์

2. สามารถเชื่อมต่ออุปกรณ์หลายๆ ชนิดรวมเข้าสู่คอนเน็กเตอร์ตัวเดียว สูงสุด 127 ตัว

3. ไม่เกิดการขัดแย้งกันของการใช้ทรัพยากรของระบบ IRQ

4. ตรวจสอบการเชื่อมต่อและตั้งค่าการทำงานต่างๆ อัตโนมัติ ระหว่างที่เครื่องกำลังทำงานอยู่ (Hot attachment)

- มาตรฐาน USB 1.0/1.1 มีอัตราในการถ่ายทอดข้อมูลความเร็วต่ำ (Low speed) เท่ากับ 1.5 เมกะบิตต่อวินาที (Mbit/sec) และความเร็วเต็มที่ (Full speed) เท่ากับ 12 Mbit/sec

- มาตรฐาน USB 2.0 จะมีอัตราเร็วในการถ่ายทอดเพิ่มขึ้นอีก 1 ระดับคือ ความเร็วสูง (High speed) ซึ่งมีความเร็วสูงถึง 480 Mbit/sec

5. ที่ขาพอร์ต USB มีแรงดันไฟตรง +5 โวลต์ (Volts : V) จ่ายออกมาด้วย ทำให้อุปกรณ์ต่อพ่วงที่ใช้พลังงานไม่มากนัก สามารถใช้แรงดันจากพอร์ต USB นี้เป็นไฟเลี้ยงเพื่อทำงานได้โดยไม่ต้องอาศัยแหล่งจ่ายไฟจากภายนอกเพิ่มเติมอีก

ดิสคริปเตอร์ : ความหมาย, ชนิดและรูปแบบการทำงาน

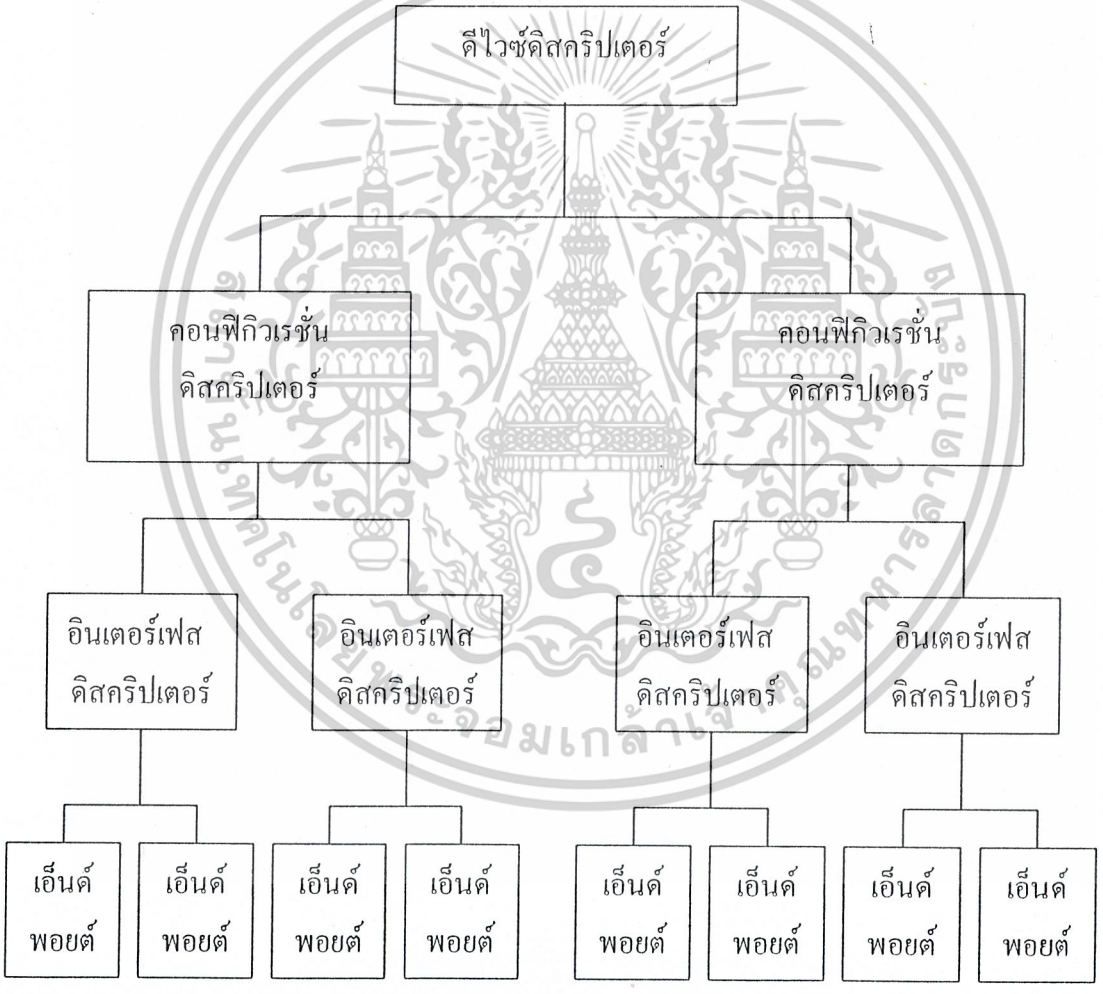
อุปกรณ์แต่ละตัวมีคุณสมบัติและการทำงานที่แตกต่างกัน โฮสต์จำเป็นต้องรู้คุณสมบัติทั้งหมดของอุปกรณ์แต่ละตัวเพื่อให้การสั่งงานเป็นไปอย่างถูกต้อง และเนื่องจากบัสข้อมูลทั้งหมดจะถูกใช้งานรับส่งข้อมูลร่วมกันระหว่างอุปกรณ์ทุกๆ ตัว สิ่งที่โฮสต์จำเป็นต้องรู้อีกอย่างคือปริมาณข้อมูลที่ต้องการส่ง (Bandwidth) ของอุปกรณ์แต่ละตัวในบัส ดังนั้นเมื่อมีอุปกรณ์ตัวใหม่ต่อเข้ากับบัส โฮสต์ต้องอ่านข้อมูลต่างๆ ที่จำเป็นเข้ามาเพื่อใช้อ้างอิงในการสั่งงานอุปกรณ์ และใช้พิจารณาว่าระบบบัสสามารถรองรับอุปกรณ์ที่มาเชื่อมต่อใหม่ได้หรือไม่ ข้อมูลเหล่านี้รวมเรียกว่าดิสคริปเตอร์ของอุปกรณ์ (Device descripters)

อุปกรณ์แต่ละตัวจะแจ้งรายละเอียดของตัวเองให้โฮสต์รู้ผ่านดิสคริปเตอร์ชนิดต่างๆ ซึ่งได้รับการแบ่งแยกเป็นชนิดตามข้อมูลที่จะแจ้งกลับไปยังโฮสต์ โดยการแบ่งแยกชนิดของดิสคริปเตอร์นั้นจะจัดเป็นลำดับชั้นตามรูปแบบ เช่น แบ่งการทำงานออกเป็น 2 โหมด แต่ละโหมดมีหน้าที่การทำงานแตกต่างกันและแต่ละหน้าที่ก็จะใช้กลุ่มของเอ็นด์พอยต์ที่แตกต่างกัน สามารถสรุปได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. ดีไวซ์ดิสคริปเตอร์ (Device descripter)

ในอุปกรณ์แต่ละตัวจะมีดีไวซ์ดิสคริปเตอร์เพียง 1 ชุดเท่านั้น ภายในจะระบุข้อมูลที่ใช้ในการเชื่อมต่อขั้นแรก (Default communication pipe) เพื่อใช้ในการกำหนดข้อมูลสถานะของ ข้อมูลเข้ากับโฮสต์ นอกจากนั้นยังเก็บข้อมูลของข้อกำหนดในโหมดทำงานต่างๆ ของอุปกรณ์รวมถึงจำนวนคอนฟิกิวเรชั่นด้วยเพราะในครั้งแรกที่อุปกรณ์เชื่อมต่อเข้ากับบัสนั้นโฮสต์ไม่มีทางรู้ได้เลยว่าต้องติดต่อกับอุปกรณ์ที่เอ็นด์พอยต์ใด จึงจำเป็นต้องขอข้อมูลจากส่วนนี้ก่อนที่จะติดต่อกับส่วนอื่น



รูปที่ 1.2 ระดับการทำงานของดิสคริปเตอร์ในพอร์ต USB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. คอนฟิกิวเรชั่นดิสคริปเตอร์ (Configuration descripter)

ใช้เก็บข้อมูลที่จำเป็นของการทำงานแต่ละโหมดการทำงานและเก็บจำนวนแต่ละเฟสที่ใช้งานในโหมดนั้นๆ เช่น อุปกรณ์บางตัวมีการทำงาน 2 โหมดคือ โหมดใช้พลังงานสูงและโหมดประหยัดพลังงาน ดิสคริปเตอร์ตัวนี้จะเก็บข้อมูลที่จำเป็นในการตั้งค่าต่างๆ ของโฮสต์เมื่อต้องการเลือกใช้โหมดการทำงานแต่ละโหมดของอุปกรณ์

3. อินเตอร์เฟสดิสคริปเตอร์ (Interface descripter)

ในแต่ละโหมดการทำงานหรือคอนฟิกิวเรชั่นอาจจะมีการเชื่อมต่อหรืออินเตอร์เฟส 1 แบบหรือมากกว่าเพื่อใช้งานในหน้าที่ต่างๆ ตัวอย่างที่เห็นได้ชัดที่สุดก็คือซีดีรอม โดยในตัวซีดีรอมจะมีการรับส่งข้อมูลปริมาณมากๆ (Mass storage) การส่งข้อมูลเสียงออดิโอและการส่งข้อมูลภาพจะเห็นได้ว่า มีอินเตอร์เฟสที่ใช้งานแยกกัน ภายในอินเตอร์เฟสดิสคริปเตอร์จะบรรจุข้อมูลการใช้งานอินเตอร์เฟสนั้นๆ โดยข้อมูลเหล่านี้จะระบุว่าอุปกรณ์ถูกจัดอยู่ในคลาส (Class) หรือคลาสย่อย (Subclass) ใด และมีเอ็นด์พอยต์จำนวนเท่าใดที่ใช้งานอินเตอร์เฟสนี้บ้าง

4. เอ็นด์พอยต์ดิสคริปเตอร์ (Endpoint descripter)

ใช้เก็บข้อมูลคุณสมบัติของแต่ละเอ็นด์พอยต์ เช่น ใช้การถ่ายทอดสัญญาณแบบใด (บัลก์ ไอโซโครนัส อินเตอร์รัพต์หรือสัญญาณควบคุม) และถ่ายทอดข้อมูลได้มากที่สุดครั้งละเท่าใด

5. สตริงดิสคริปเตอร์ (String descripter)

เป็นดิสคริปเตอร์ที่ไม่ได้อยู่ภายในโครงสร้างตามรูปที่ 1.2 เพราะดิสคริปเตอร์ชนิดนี้จะเก็บข้อมูลตัวอักษรที่สามารถอ่านเข้าใจได้ไว้อธิบายส่วนต่างๆ ของดิสคริปเตอร์ทั้งสี่ตัวข้างต้น เช่น เก็บชื่อบริษัทผู้ผลิต หรือหมายเลขประจำตัวของอุปกรณ์ เป็นต้น

6. คลาสสเปกซิฟิกดิสคริปเตอร์ (Class-specific descripter)

เป็นดิสคริปเตอร์พิเศษที่มีเฉพาะในอุปกรณ์บางตัวที่จัดอยู่ในบางคลาสเท่านั้น ภายในเก็บข้อมูลเฉพาะของคลาสนั้นๆ ที่อยู่นอกเหนือจากดิสคริปเตอร์พื้นฐาน 4 ชนิดแรก

ดังนั้นจึงพอสรุปการใช้งานของดิสคริปเตอร์พื้นฐานได้ดังนี้

1. ข้อมูลทั่วไปของตัวอุปกรณ์จะเก็บอยู่ในดิสคริปเตอร์ของอุปกรณ์หรือดีไวซ์ดิสคริปเตอร์
2. ข้อมูลการทำงานของแต่ละโหมดเก็บอยู่ในคอนฟิกิวเรชั่นดิสคริปเตอร์
3. ข้อมูลหน้าที่การทำงานเก็บอยู่ในดิสคริปเตอร์การเชื่อมต่อหรืออินเตอร์เฟสดิสคริปเตอร์
4. ข้อมูลการทำงานของแต่ละเอ็นด์พอยต์จะเก็บอยู่ในเอ็นด์พอยต์ดิสคริปเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กระบวนการกำหนดการทำงานของอุปกรณ์เมื่อเริ่มการทำงานของ USB หรือเมื่อมีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้ามาในระบบ ลำดับขั้นตอนคร่าวๆ ในการทำงานจะเป็นดังนี้

1. ฮับตรวจสอบพบว่ามีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้าสู่ระบบแล้วแจ้งผลกลับไปยังโฮสต์คอนโทรลเลอร์

2. โฮสต์คอนโทรลเลอร์สั่งให้ฮับเปิดการทำงานของแหล่งจ่ายไฟในโหมดประหยัด เพื่อให้อุปกรณ์ที่อาศัยพลังงานจากบัสสามารถทำงานได้

3. โฮสต์คอนโทรลเลอร์สั่งให้ฮับรีเซตพอร์ตที่อุปกรณ์ตัวใหม่มาเชื่อมต่อเพื่อทำให้อุปกรณ์รีเซตค่าแอดเดรสและเอ็นด์พอยต์ของตนเองให้เป็นค่าเริ่มต้น (Default)

4. โฮสต์อ่านดิสคริปเตอร์ต่างๆ จากตัวอุปกรณ์และพิจารณาว่าทรัพยากรของระบบพอเพียงต่อความต้องการของตัวอุปกรณ์หรือไม่ ทรัพยากรในที่นี้คือ พลังงานไฟฟ้าและปริมาณข้อมูลที่จะส่งของตัวอุปกรณ์ หากพิจารณาแล้วว่าไม่สามารถทำงานได้ก็จะสั่งให้ฮับปิดการทำงานของพอร์ต

5. เมื่อโฮสต์พิจารณาแล้วว่าสามารถให้บริการแก่อุปกรณ์ตัวที่มาเชื่อมต่อได้ จะควบคุมให้แหล่งจ่ายไฟจ่ายพลังงานตามที่อุปกรณ์ต้องการ รวมไปถึงการตั้งค่าแอดเดรสและกำหนดค่าต่างๆ

6. หลังจากตั้งค่าเรียบร้อยแล้ว คอมพิวเตอร์ก็จะรู้จักกับอุปกรณ์ตัวใหม่และสามารถติดต่อกับอุปกรณ์ได้ทันทีโดยไม่ต้องปิดและเปิดคอมพิวเตอร์ใหม่

สายเชื่อมต่อและคอนเน็กเตอร์ของพอร์ต USB

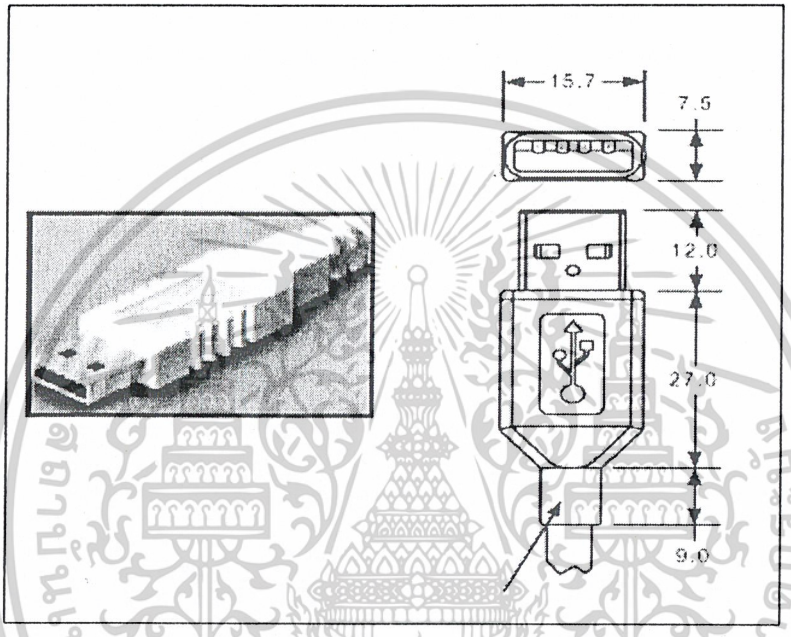
1. คอนเน็กเตอร์ของพอร์ต USB

สายเชื่อมต่อ USB ถูกออกแบบมาสำหรับการเชื่อมต่อระหว่างตัวอุปกรณ์ USB กับฮับ ซึ่งอาจจะเป็นรูตฮับซึ่งอยู่หลังเครื่องคอมพิวเตอร์ ฮับที่รวมอยู่ในตัวอุปกรณ์ (Compound device) หรือฮับที่เป็นตัวเดี่ยวๆ (USB hub) อุปกรณ์หลายชนิดมีสายต่อด้านอุปกรณ์ไว้อย่างถาวรด้านหนึ่ง และมีคอนเน็กเตอร์ตัวผู้ไว้สำหรับต่อเข้ากับฮับอีกด้านหนึ่ง แต่มีอุปกรณ์อีกหลายชนิดที่ไม่มีสายต่อถาวรไว้โดยที่ตัวอุปกรณ์จะมีคอนเน็กเตอร์ USB ตัวเมียไว้ เพื่อใช้งานกับสาย USB ที่มีหัวคอนเน็กเตอร์ทั้งสองด้าน จากอุปกรณ์ที่ไม่มีสายถาวรต่อไว้จะเห็นได้ว่าต้องใช้สายเชื่อมต่อซึ่งมีคอนเน็กเตอร์ทั้งสองด้านซึ่งถ้าใช้คอนเน็กเตอร์แบบเดียวกันทั้งหมดอาจทำให้ผู้ใช้นำคอนเน็กเตอร์ด้านหนึ่งต่อเข้ากับฮับพอร์ตหนึ่ง แล้วต่อปลายอีกข้างเข้ากับฮับอีกพอร์ตหนึ่งได้ (เพราะฮับจะมีพอร์ตอยู่หลายพอร์ต) เพื่อป้องกันเหตุการณ์เช่นนี้จึงมีการกำหนดมาตรฐานของคอนเน็กเตอร์ USB ให้มี 2 รูปแบบดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.1 คอนเน็กเตอร์อนุกรม A

เป็นคอนเน็กเตอร์ด้านฮับที่เชื่อมต่อระหว่าง USB พอร์ตของฮับ (ทั้งรูตฮับและ USB ฮับทั่วไป) กับสายเชื่อมต่อจากตัวอุปกรณ์นั่นคือคอนเน็กเตอร์ตัวเมียจะติดตั้งอยู่กับฮับและคอนเน็กเตอร์ตัวผู้จะติดอยู่กับสายที่ต่อออกมาจากตัวอุปกรณ์ ดังในรูปที่ 1.3

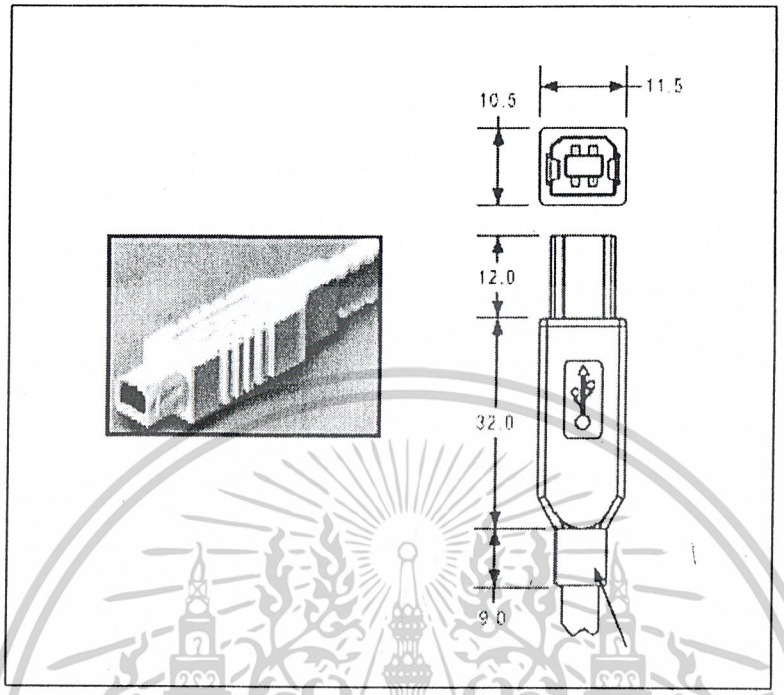


รูปที่ 1.3 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม A

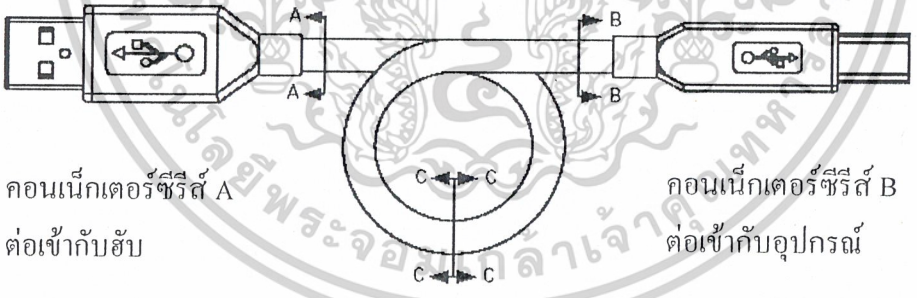
1.2 คอนเน็กเตอร์อนุกรม B

เป็นคอนเน็กเตอร์ด้านอุปกรณ์ที่เชื่อมต่อสายเข้ากับตัวอุปกรณ์ USB นั่นคือ คอนเน็กเตอร์ตัวเมียจะติดตั้งอยู่ในตัวอุปกรณ์ USB ส่วนคอนเน็กเตอร์ตัวผู้ก็จะอยู่ที่สายที่ต่อออกมาจากฮับ (ถ้าอุปกรณ์ใดที่มีสายต่อออกมาจากตัวอุปกรณ์อย่างถาวรจะไม่มีการใช้คอนเน็กเตอร์แบบนี้) ดังในรูปที่ 1.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.4 รูปร่างและขนาดของคอนเน็กเตอร์ USB ในอนุกรม B



รูปที่ 1.5 การใช้งานสาย USB ที่ใช้คอนเน็กเตอร์ทั้งสองรูปแบบ

2. หน้าสัมผัสคอนเน็กเตอร์ USB

ถึงแม้จะมีการแบ่งคอนเน็กเตอร์ออกเป็น 2 แบบ แต่ทั้งสองรูปแบบนั้นใช้สายนำสัญญาณเหมือนกัน โดยใช้สาย 2 เส้นในการส่งไฟเลี้ยงให้แก่อุปกรณ์ และสายอีก 2 เส้นสำหรับรับและส่งข้อมูล แต่เนื่องจากระบบ USB นั้นออกแบบมาเพื่อให้สามารถเชื่อมต่อหรือปลดอุปกรณ์ออกจากระบบได้ในระหว่างการใช้งานดังนั้นหน้าสัมผัสของคอนเน็กเตอร์จึงมีการออกแบบให้มีความพิเศษ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เพื่อให้รองรับคุณสมบัติ ดังกล่าวได้ นั่นคือหน้าสัมผัสของสาย 2 เส้นที่ใช้ส่งไฟเลี้ยงจะยื่นออกมายาวกว่าหน้าสัมผัสที่ใช้รับส่งข้อมูล หน้าสัมผัสสำหรับไฟเลี้ยงยาว 7.41 มิลลิเมตร (mm) หน้าสัมผัสสำหรับรับส่งข้อมูลยาว 6.41mm การออกแบบคอนเน็กเตอร์เช่นนี้จะทำให้ตัวอุปกรณ์ได้รับไฟเลี้ยงก่อนที่จะได้รับสัญญาณข้อมูลเมื่อเชื่อมต่ออุปกรณ์ตัวใหม่เข้าระบบ ทำให้อุปกรณ์ที่เชื่อมต่อเข้าไปใหม่สามารถทำงานได้ทันทีโดยไม่ได้รับความเสียหาย เพราะถ้าหากอุปกรณ์ได้รับสัญญาณข้อมูลก่อนที่ได้รับไฟเลี้ยงจะเกิดความเสียหายแก่ไอซีที่อยู่ภายในได้ ขาของคอนเน็กเตอร์แต่ละตัวทำหน้าที่ตามที่แสดงในตารางที่ 1.1

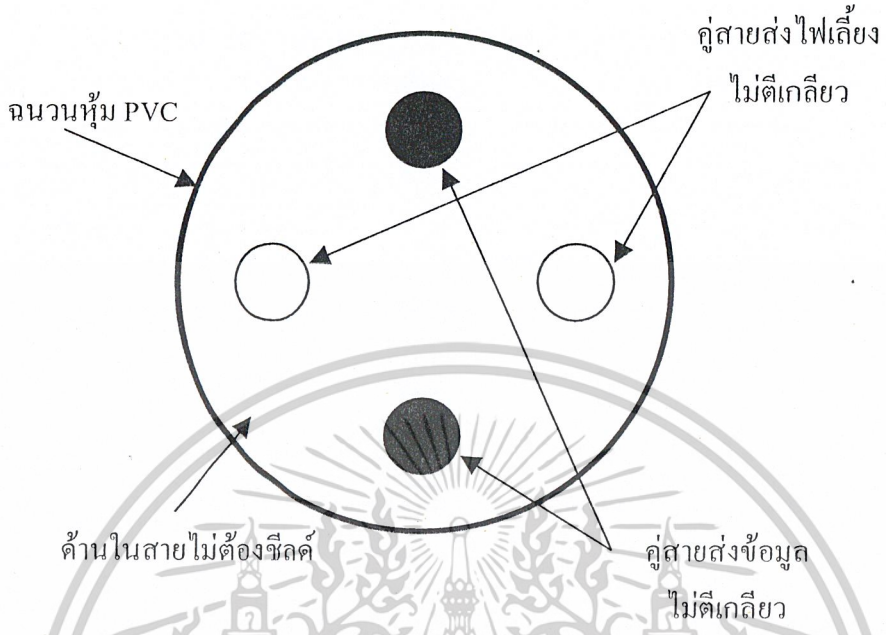
ตารางที่ 1.1 หน้าที่ของขาคอนเน็กเตอร์แต่ละขา

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

3. สายนำสัญญาณ USB

การถ่ายทอดข้อมูลภายในสาย USB นั้นมีความเร็วสูงในระดับ 1 Mbit/sec ถึง 12 Mbit/sec การถ่ายทอดข้อมูลที่ความเร็วสูงขนาดนี้จะเกิดการแพร่กระจายสนามแม่เหล็กไฟฟ้าออกมาจากสายส่ง ทำให้ต้องมีการกำหนดคุณสมบัติของสายส่งในหลายๆ ด้านเพื่อไม่ให้เกิดการแพร่กระจายของสนามแม่เหล็กไฟฟ้าเหล่านี้ และเนื่องจาก USB มีการแบ่งความเร็วของการถ่ายทอดข้อมูลออกเป็น 2 ระดับคือ แบบความเร็วเต็มที่ ที่มีอัตราเร็ว 12 Mbit/sec และแบบความเร็วต่ำ ที่มีอัตราเร็ว 1.5 Mbit/sec ทำให้คุณสมบัติของสายนำสัญญาณของอุปกรณ์แต่ละชนิดจะถูกกำหนดแยกแตกต่างกันออกไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

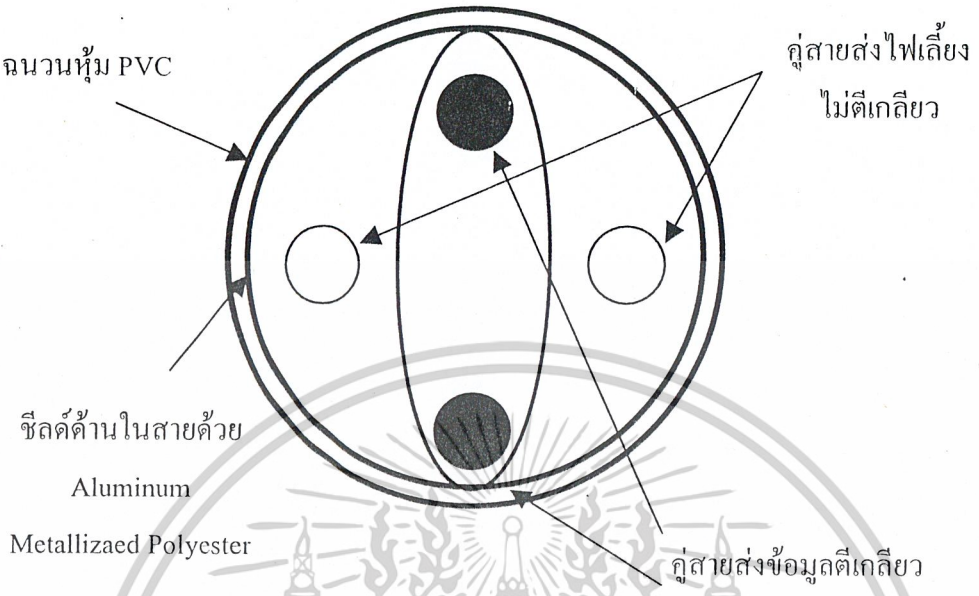


รูปที่ 1.6 โครงสร้างภายในของสาย USB แบบความเร็วต่ำ

3.1 สายนำสัญญาณความเร็วต่ำ

จากรูปที่ 1.6 แสดงให้เห็นภาพหน้าตัดของสายนำสัญญาณความเร็วต่ำ ชั้นนอกสุดของสายสัญญาณจะเป็นฉนวนหุ้มธรรมดา ภายในมีสายสัญญาณจะมีสายตัวนำ 4 เส้นโดย 2 เส้น ใช้ส่งไฟเลี้ยงกำหนดให้ใช้ลวดทองแดงเบอร์ 20-28 AWG ไม่ตีเกลียว ส่วนสายอีก 2 เส้นใช้ถ่ายทอดข้อมูลกำหนดให้ใช้ลวดทองแดงเบอร์ 28 AWG ไม่ตีเกลียว สายส่งข้อมูลความเร็วต่ำนี้ไม่จำเป็นต้องชีลด์ความยาวทั้งหมดของสายไม่เกิน 3 เมตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.7 โครงสร้างภายในของสาย USB แบบความเร็วสูง

3.2 สายนำสัญญาณความเร็วสูง

สายนำสัญญาณความเร็วสูงของ USB มีสาย 2 เส้นที่ใช้ถ่ายทอดข้อมูล กำหนดให้ใช้ลวดทองแดงเบอร์ 28 AWG เช่นเดียวกับสายนำสัญญาณความเร็วต่ำ แต่สาย 2 เส้นนี้จะต้องตีเกลียวและชั้นนอกสุดของสายสัญญาณ ถัดจากฉนวนจะต้องถูกหุ้มด้วยอลูมิเนียมฟอยล์ ดังรูปที่ 1.7 ความยาวของสายสัญญาณไม่เกิน 5 เมตร จะต้องมีค่าหน่วงเวลาสัญญาณ (Propagation delay) ไม่เกิน 30 นาโนวินาที (ns) ตลอดความยาวสาย 5 เมตร แต่ถ้าค่าหน่วงเวลามากกว่า 30ns ค่าความยาวสูงสุดของสายสัญญาณก็จะลดลงเป็นสัดส่วนดังตารางที่ 1.2

ตารางที่ 1.2 แสดงค่าอัตราการหน่วงเวลาของสาย USB เมื่อเทียบกับความยาวของสายสัญญาณ

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การตรวจสอบการเชื่อมต่อและความเร็วของอุปกรณ์

คุณสมบัติเด่นข้อหนึ่งของ USB คือสามารถตรวจสอบการเชื่อมต่อของอุปกรณ์ตัวใหม่และตั้งค่าเริ่มต้นที่จำเป็นได้อัตโนมัติ แต่เนื่องจากอุปกรณ์ USB แบ่งออกเป็นอุปกรณ์ความเร็วสูงและต่ำ ถ้าส่งข้อมูลผิดความเร็วก็จะไม่สามารถสื่อสารกับอุปกรณ์นั้นๆ ได้ นั่นหมายความว่าจะต้องมีสัญญาณรูปแบบพิเศษที่ใช้แจ้งสถานะการเชื่อมต่อหรือปลดออกรวมถึงระบุความเร็วของอุปกรณ์นั้นๆ ได้ด้วย

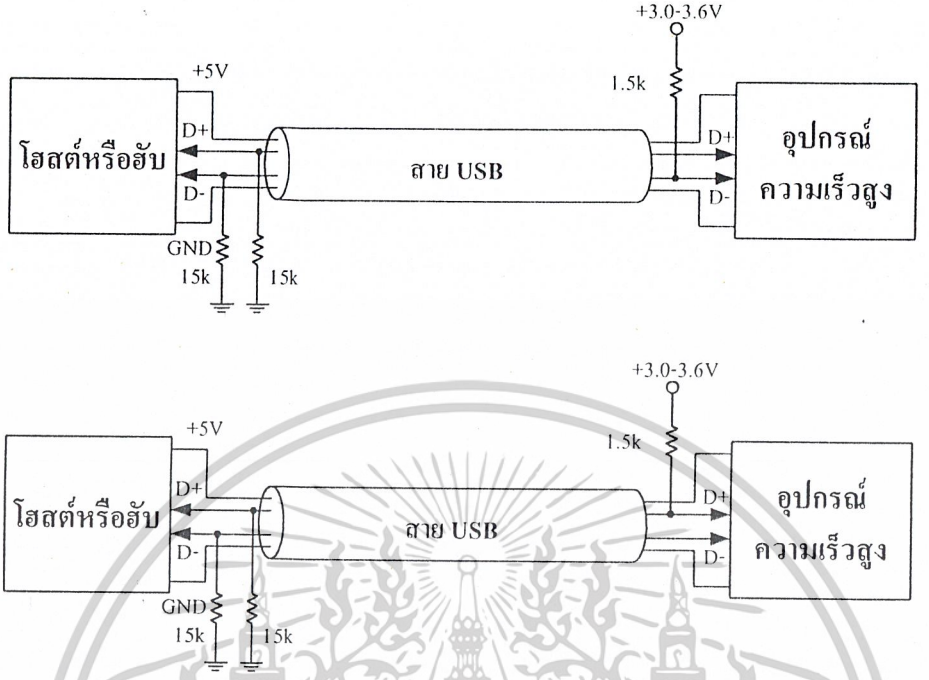
การตรวจสอบการเชื่อมต่อของ USB นั้นจะตรวจสอบจากกรเปลี่ยนแปลงระดับของสัญญาณในสายสัญญาณ D- และ D+ โดยสายข้อมูลทั้งสองที่ด้านฮับจะถูกต่อด้วยตัวต้านทาน 15 กิโลโอห์ม (KΩ) พูลดาวน์ไว้ทั้งสองเส้น (การพูลดาวน์ (Pull-down) : เป็นการต่อขาข้างหนึ่งของอุปกรณ์เข้ากับสายสัญญาณ อีกข้างหนึ่งต่อกราวด์ของระบบ) ซึ่งจะส่งผลให้สายสัญญาณทั้งสองมีระดับแรงดันเป็น 0V ในขณะที่ไม่มีอุปกรณ์ใดๆ ต่ออยู่เลย

ที่ด้านอุปกรณ์สำหรับอุปกรณ์ความเร็วต่ำสายสัญญาณ D- จะต่อตัวต้านทาน 1.5KΩ พูลอัปกับแรงดัน 3.0-3.6V (การพูลอัป (Pull-up) : เป็นการต่อขาข้างหนึ่งของอุปกรณ์เข้ากับสายสัญญาณอีกข้างหนึ่งต่อไฟเลี้ยง ทำให้สถานะที่จุดนั้นเกิดเป็นลอจิก “1”) ส่วนอุปกรณ์ความเร็วสูง สายสัญญาณที่ถูกพูลอัปคือ สายสัญญาณ D+ เมื่อมีอุปกรณ์ตัวใหม่ถูกต่อเข้ากับฮับจะเกิดการแบ่งแรงดันจากตัวต้านทานสองตัวก็คือตัวต้านทาน 1.5KΩ ที่พูลดาวน์อยู่ทางฮับ ทำให้แรงดันของสายสัญญาณเพิ่มจาก 0V ขึ้นไปที่ 90% ของไฟเลี้ยงที่พูลอัป คำนวณได้จากสูตรดังนี้

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

การเปลี่ยนแรงดันนี้จะทำให้ฮับรู้ว่ามีอุปกรณ์ตัวใหม่ต่อเข้ากับระบบ และจะทำให้รู้ว่าอุปกรณ์ถูกปลดออกจากระบบเช่นเดียวกัน โดยถ้าสัญญาณที่เกิดการเปลี่ยนแปลงแรงดันนี้เป็นสัญญาณ D- หมายความว่าอุปกรณ์ที่นำมาต่อเป็นอุปกรณ์ความเร็วต่ำ แต่ถ้าสัญญาณ D+ เปลี่ยนแปลงแรงดัน หมายความว่าอุปกรณ์ความเร็วสูงนำมาต่อเข้ากับระบบดังรูปที่ 1.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.8โครงสร้างภายในของสาย USB แบบความเร็วสูง

การจัดการด้านพลังงานของ USB

เนื่องจากพอร์ต USB สามารถจ่ายพลังงานให้แก่อุปกรณ์ที่นำมาเชื่อมต่อได้ แต่ย่อมต้องมีข้อจำกัดว่า สามารถจ่ายพลังงานออกไปได้มากน้อยเพียงใด โดยเฉพาะกับพอร์ต USB ที่ยอมให้อุปกรณ์มาเชื่อมต่อได้มากถึง 127 ตัว โฮสต์จะต้องมีการจัดการด้านพลังงานที่ดีพอ มิฉะนั้นจะเกิดการดึงพลังงานจากบัส จนเครื่องคอมพิวเตอร์ไม่สามารถทำงานได้

1. การรายงานอัตราการใช้พลังงาน

โฮสต์จะรู้ข้อมูลเกี่ยวกับการใช้พลังงานของฮับและอุปกรณ์แต่ละตัวจากการอ่านคอนฟิกิวเรชั่นดิสคริปเตอร์ซึ่งจะถูกอ่านออกมาตั้งแต่ครั้งแรกที่มีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในบัส โดยข้อมูลที่เกี่ยวกับการใช้พลังงานจะมีอยู่ 2 ส่วน ส่วนแรกจะบอกโฮสต์ว่าตัวอุปกรณ์นี้ใช้พลังงานจากบัสหรือใช้พลังงานจากแหล่งพลังงานของตัวเองและส่วนที่สองบอกว่าจะขอใช้พลังงานจากบัสเป็นปริมาณเท่าใด มีหน่วยเป็นมิลลิแอมป์ (mA)

ในกรณีที่มีการเชื่อมต่อฮับเข้ากับบัสโฮสต์จะอ่านคอนฟิกิวเรชั่นดิสคริปเตอร์ไปเพื่อพิจารณาการใช้พลังงาน แต่เนื่องจากฮับจำเป็นจะต้องจ่ายพลังงานไปเลี้ยงอุปกรณ์ที่มาเชื่อมต่ออยู่กับตัวมันด้วย จึงต้องมีข้อกำหนดในด้านแรงดันของไฟเลี้ยงที่จะจ่ายออกจากตัวฮับเพื่อไม่ให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แรงดันปลายทางที่ตัวอุปกรณ์น้อยเกินไปจนไม่สามารถทำงานได้ โดยกำหนดไว้ว่า แรงดันที่จ่ายให้แก่ฮับเพื่อให้ฮับทำงานนั้นจะต้องไม่ต่ำกว่า 4.75V และแรงดันที่ฮับจ่ายออกไปยังตัวอุปกรณ์แต่ละตัวจะต้องไม่ต่ำกว่า 4.40V (ดูรูปที่ 1.9 ประกอบ) นั่นหมายความว่าอุปกรณ์ USB ทุกๆ ตัวจะต้องสามารถทำงานได้ที่แรงดันไฟเลี้ยงอย่างน้อย 4.40V

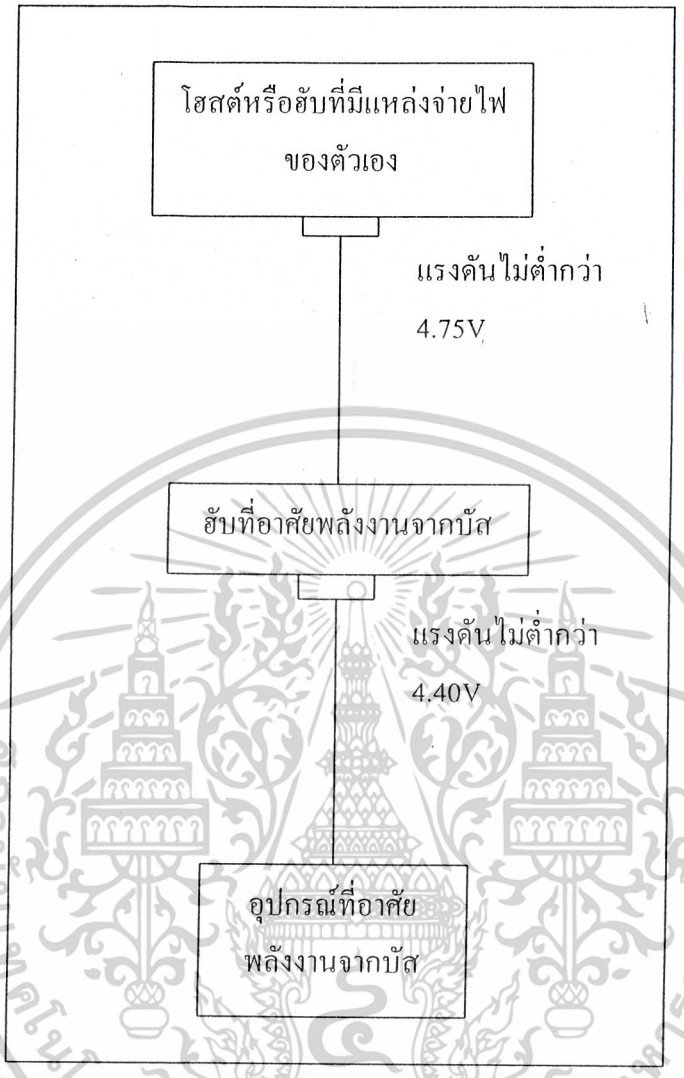
2. การจำกัดกระแส

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

สำหรับฮับที่ใช้ไฟเลี้ยงจากบัส ตัวฮับเองจะใช้พลังงานจากบัสเพื่อควบคุมการทำงานภายในของตัวฮับเอง 1 หน่วยเท่ากับ 100mA และจากข้อกำหนดด้านกระแสที่ว่า บัสจะจ่ายกระแสได้ ไม่เกิน 500mA นั่นหมายความว่า ฮับจะเหลือพลังงานที่สามารถจ่ายให้แก่พอร์ตต่างๆ ได้อีก 4 หน่วย ซึ่งเท่ากับว่า จะมีพอร์ตได้มากที่สุด 4 พอร์ต โดยแต่ละพอร์ตต้องดึงกระแสไม่เกิน 100mA แต่ในกรณีของฮับที่มีแหล่งจ่ายไฟของตัวเองจะไม่พบกับปัญหานี้เพราะสามารถจ่ายกระแสให้แต่ละพอร์ตได้เต็ม 500mA

นอกจากข้อกำหนดในเรื่องการดึงกระแสของแต่ละพอร์ตแล้ว ในแต่ละพอร์ตจะต้องมีวงจรจำกัดกระแส (Current limit) ไม่ให้เกิน 5 แอมป์ (Amp : A) สำหรับพอร์ตแต่ละพอร์ตเพื่อป้องกันการลัดวงจรระหว่างไฟเลี้ยงและกราวด์ แต่ในกรณีของฮับที่อาศัยไฟเลี้ยงจากบัสไม่จำเป็นต้องมีวงจรป้องกันส่วนนี้เพราะรูตฮับหรือฮับตัวก่อนหน้า (ที่มีแหล่งจ่ายไฟของตัวเอง) จะมีวงจรป้องกันอยู่แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.



รูปที่ 1.9 ข้อจำกัดด้านแรงดันที่จุดเชื่อมต่อต่าง ๆ

ชนิดของการถ่ายทอดสัญญาณ (Transfer type)

เนื่องจาก USB ออกแบบไว้รองรับการทำงานของอุปกรณ์หลายๆ ประเภท ซึ่งแต่ละประเภทจะมีความต้องการในการรับหรือส่งข้อมูลที่แตกต่างกันออกไปตามชนิด หน้าที่ของอุปกรณ์นั้นๆ ดังนั้นการรับส่งสัญญาณจึงต้องออกแบบไว้ให้ครอบคลุมการทำงานทุกๆ ประเภทของอุปกรณ์ทุกๆ ชนิดทำให้ USB กำหนดชนิดของการรับส่งข้อมูลออกเป็น 4 ชนิดคือ

1. การถ่ายทอดสัญญาณแบบไอโซโครนัส(Isochronous transfer)
2. การถ่ายทอดสัญญาณแบบอินเตอร์รัพต์ (Interrupt transfer)
3. การถ่ายทอดสัญญาณแบบบัลก์ (Bulk transfer)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

4. การถ่ายทอดสัญญาณควบคุม (Control transfer)

เอ็นด์พอยต์เป็นชุดรีจิสเตอร์ที่ใช้ติดต่ออุปกรณ์โดยแต่ละเอ็นด์พอยต์ก็จะมีหน้าที่ในการทำงานที่ต่างกันไป นั่นหมายความว่า แต่ละเอ็นด์พอยต์ก็จะใช้การรับส่งข้อมูล 1 ใน 4 ชนิดที่กล่าวไปนี้ในการติดต่อกับโฮสต์หรือกล่าวอย่างง่ายๆก็คือแต่ละเอ็นด์พอยต์จะใช้ชนิดการรับส่งข้อมูลเป็นของตัวเองเพียงชนิดเดียวแยกจากเอ็นด์พอยต์อื่นๆ อย่างอิสระ เช่น ในอุปกรณ์ตัวหนึ่งจะมีเอ็นด์พอยต์ควบคุมที่ใช้การถ่ายทอดสัญญาณควบคุมเพื่อรับส่งข้อมูลควบคุมการทำงานของอุปกรณ์ ในขณะที่อีกเอ็นด์พอยต์หนึ่งใช้การถ่ายทอดแบบไอโซโครนัสเพื่อส่งข้อมูลจากการทำงานหน้าที่หนึ่ง และใช้อีกเอ็นด์พอยต์หนึ่งที่ใช้การถ่ายทอดแบบบัลก์เพื่อรับส่งตัวข้อมูลจากการทำงานอีกหน้าที่หนึ่ง เป็นต้น ส่วนถัดไปจะกล่าวถึงรายละเอียดของการรับส่งข้อมูลทั้ง 4 ชนิดในด้านความเหมาะสมกับงานต่าง ๆ และจังหวะการรับส่งข้อมูล

1. การถ่ายทอดสัญญาณแบบไอโซโครนัส

การรับส่งข้อมูลชนิดนี้จะมีการส่งหรือรับข้อมูลในทุก ๆ เฟรมข้อมูล (ทุกๆ 1ms) ทำให้เกิดการรับส่งข้อมูลด้วยอัตราเร็วที่คงที่ แต่นั่นก็หมายความว่า การรับส่งแบบนี้จะต้องจองการรับส่งข้อมูลในแต่ละเฟรม 1ms ไว้ตลอดเวลาด้วยประเภทของงานที่ใช้การรับส่งข้อมูลชนิดนี้คือ งานที่ต้องการอัตราการส่งข้อมูลที่คงที่ตลอดเวลา เช่น การส่งข้อมูลเสียงเพลงไปยังลำโพง USB หรือการรับข้อมูลจากไมโครโฟน USB เพราะว่า ลักษณะงานประเภทนี้ทางปลายทางจะนำข้อมูลที่ได้รับแปลงกลับเป็นข้อมูลแอนะลอก (Analog) ซึ่งจะเกิดความผิดเพี้ยนทางความถี่ได้ หากข้อมูลที่ส่งเกิดการขาดช่วงไม่สม่ำเสมอ แต่จากข้อกำหนดที่ว่าต้องส่งหรือรับข้อมูลต่อเนื่องในทุก ๆ เฟรมข้อมูล นั่นหมายความว่า หากเกิดความผิดพลาดของข้อมูลจะไม่มีการพยามยามส่งข้อมูลที่ผิดนั้นกลับไปใหม่ (เพราะหากมีการส่งใหม่จะทำให้อัตราการส่งข้อมูลเปลี่ยนไป) จึงทำให้การถ่ายทอดข้อมูลแบบนี้จะไม่มีการรับประกันความถูกต้องของข้อมูล

ก่อนเริ่มการทำงาน อุปกรณ์จะต้องการซิงโครไนซ์การทำงานของตัวเองเข้ากับสัญญาณเริ่มต้นแต่ละเฟรมของ USB เพื่อให้อุปกรณ์สามารถรับหรือส่งข้อมูลในทุก ๆ 1ms ได้ทัน ทำให้เกิดการรับและส่งข้อมูลในจังหวะที่ถูกต้อง การซิงโครไนซ์เข้ากับเฟรมข้อมูลนั้นมีอยู่ 3 วิธีดังนี้

1. อะซิงโครนัส (Asynchronous) การซิงโครไนซ์วิธีนี้สัญญาณนาฬิกาของอุปกรณ์ไม่จำเป็นต้องเข้าจังหวะกับสัญญาณข้อมูลของ USB สัญญาณนาฬิกาที่ควบคุมการทำงานของอุปกรณ์จะเป็นอิสระไม่ขึ้นอยู่กับสัญญาณ SOF แต่อัตราการส่งข้อมูลจะต้องถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่ง ซึ่งกำหนดตอนเริ่มต้นการเชื่อมต่อ ไม่สามารถเปลี่ยนแปลงความเร็วได้

ตัวอย่างของอุปกรณ์ที่ซิงโครไนซ์ด้วยวิธีนี้คือ ซีดีรอมในโหมดการเล่นเพลง เพราะว่า ซีดีเพลงนั้นจะมีอัตราการสุ่มข้อมูลคงที่เท่ากับ 44.1 กิโลเฮิรตซ์ (kHz) ซึ่งอัตราการส่งข้อมูลนี้จะเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อิสระจากการส่งสัญญาณ SOF ของ USB นอกจากซีดีรอมแล้วอีกตัวอย่างหนึ่งที่เห็นได้ชัดก็คือไมโครโฟน USB ที่มีอัตราการสุ่มข้อมูล (Sampling rate) เป็นอิสระไม่ขึ้นอยู่กับสัญญาณ SOF ของ USB

2. ซิงโครนัส (Synchronous) ในวิธีสัญญาณนาฬิกาที่ใช้อุปกรณ์กับสัญญาณ SOF ของ USB จะต้องเข้าจังหวะกัน โดยก่อนเริ่มการทำงาน อุปกรณ์จะต้องมีการปรับสัญญาณนาฬิกาของตัวเองให้เข้ากับจังหวะ SOF แต่ถ้าหากอุปกรณ์ตัวนั้นไม่สามารถปรับสัญญาณนาฬิกาของตัวเองได้ USBจะยอมให้จังหวะการส่ง SOF ของตัวอุปกรณ์นั้นเป็นหลัก แล้วปรับจังหวะการส่ง SOF ของตัวโฮสต์เองให้เข้ากับการส่งข้อมูลของอุปกรณ์นั้นได้ โดยอุปกรณ์ตัวนั้นจะเรียกว่า อุปกรณ์มาสเตอร์ (Master device) แต่ USB อนุญาตให้มีอุปกรณ์มาสเตอร์ได้เพียงตัวเดียวเท่านั้นในระบบ ความเร็วในการถ่ายทอดข้อมูลวิธีนี้จะต้องกำหนดไว้ตายตัวตั้งแต่เริ่มต้นการเชื่อมต่อเช่นเดียวกับแบบอะซิงโครนัส

ตัวอย่างของอุปกรณ์ที่ทำงานลักษณะนี้คือ ไมโครโฟน USB ที่มีอัตราการสุ่มข้อมูลเข้าจังหวะกับสัญญาณ SOF นั่นเองอีกตัวอย่างหนึ่งก็คือโมเด็ม (MODEM) ISDN ความเร็ว 64 Kbit/sec จะต้องเกิดการซิงโครไนซ์สัญญาณ SOF ของโฮสต์เข้ากับสัญญาณนาฬิกาของระบบ ISFN ก่อนจึงจะเกิดการถ่ายทอดข้อมูลที่ความเร็วคงที่ 64 Kbit/sec ได้

3. อะแดปตีฟ (Adaptive) วิธีนี้เป็นวิธีที่ตัวอุปกรณ์มีความสามารถมากที่สุด เพราะอุปกรณ์สามารถปรับความเร็วในการถ่ายทอดข้อมูลได้ในช่วงที่กว้างมาก ไม่ถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่งสามารถเปลี่ยนแปลงความเร็วได้ในระหว่างการใช้งาน สัญญาณนาฬิกาภายในระบบของอุปกรณ์จะเข้าจังหวะกับสัญญาณข้อมูลที่ส่งในสาย (แตกต่างจากวิธีซิงโครนัสที่เข้าจังหวะกับสัญญาณ SOF)

ตัวอย่างของอุปกรณ์ที่ทำงานลักษณะนี้คือ ซีดีรอมเล่นเพลงที่มีอุปกรณ์พิเศษในการเปลี่ยนอัตราการสุ่มข้อมูล (Sample rate converts : SRC) ซึ่งด้วยอุปกรณ์ตัวนี้จะทำให้ความเร็วในการสุ่มข้อมูลไม่จำเป็นต้องคงที่เท่ากับ 44.1kHz แต่จะสามารถปรับเปลี่ยนไปตามขอบเขตการทำงานของตัว SRC

จังหวะการถ่ายทอดข้อมูลแบบไอโซโครนัสจะถูกกำหนดให้ส่งข้อมูลในทุกๆ 1ms เพื่อให้เกิดอัตราการส่งข้อมูลที่คงที่ขึ้น นั่นหมายความว่า จะต้องจองแบนด์วิดธ์ของสายไว้คงที่ตลอดเวลา ขนาดข้อมูลใหญ่ที่สุดที่สามารถส่งได้ในแต่ละเฟรมเท่ากับ 1,023 ไบต์ โดยอุปกรณ์จะแจ้งขนาดของการรับส่งข้อมูลที่ต้องการใช้ในแต่ละเฟรมแก่โฮสต์ทางเอ็นด์พอยต์ดิสคริปเตอร์ ซึ่งโฮสต์จะนำไปใช้พิจารณาความสามารถของระบบว่า รองรับการทำงานของอุปกรณ์ได้หรือไม่ หากพิจารณาแล้วว่าแบนด์วิทธ์ข้อมูลของระบบไม่เพียงพอต่อความต้องการของตัวอุปกรณ์ ก็จะไม่ตั้งค่าที่จำเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

ในการทำงานให้แก่ตัวอุปกรณ์ และปิดการทำงานของพอร์ตนั้นๆ ทำให้อุปกรณ์ตัวนั้นไม่สามารถทำงานได้

เนื่องจากในการถ่ายทอดข้อมูลแบบนี้ต้องการความเร็วที่คงที่ตลอดเวลา เมื่อเกิดความผิดพลาดของข้อมูลจึงไม่สามารถส่งใหม่ได้ เพราะจะทำให้เกิดการซิงโครไนซ์ของข้อมูลคลาดเคลื่อนไปส่งผลให้การรับข้อมูลเกิดความผิดพลาด ดังนั้นการถ่ายทอดข้อมูลแบบไอโซโครนัสจึงไม่มีการตรวจสอบความผิดพลาดของการส่งข้อมูลและไม่มีการส่งข้อมูลซ้ำ ทำให้วิธีการนี้ไม่เหมาะสมกับงานที่ต้องการความถูกต้องของข้อมูลเป็นสำคัญ

2. การถ่ายทอดสัญญาณแบบอินเตอร์รัพต์

การรับส่งข้อมูลชนิดนี้สร้างขึ้นเพื่อเลียนแบบการสร้างสัญญาณอินเตอร์รัพต์ของอุปกรณ์ให้แก่ระบบ วิธีการเลียนแบบอาศัยการวนอ่านข้อมูลจากตัวอุปกรณ์ต่างๆ ในระยะเวลาที่กำหนดอย่างสม่ำเสมอหรือเรียกว่า โพลลิ่ง (Polling) โดยอัตราการวนอ่านข้อมูลนี้จะต้องไม่ช้าเกินไปเพราะจะทำให้เกิดการสูญเสียข้อมูลที่อ่านไม่ทันได้ ในขณะเดียวกันก็จะต้องไม่เร็วเกินไป จนเกินเข้าไปในแบนด์วิดธ์ข้อมูลมากเกินจำเป็น ถ้าหากอุปกรณ์ไม่มีข้อมูลที่ต้องการส่ง ก็จะส่งสัญญาณ No Acknowledge (NAK) ตอบกลับ

คาบเวลาในการวนอ่านข้อมูลของอุปกรณ์แต่ละตัวจะขึ้นอยู่กับความต้องการของอุปกรณ์แต่ละชนิด คาบเวลานี้สามารถเปลี่ยนได้ในช่วงกว้างตั้งแต่วนอ่านทุกๆ 1 เฟรม (1ms) ไปจนถึงทุกๆ 255 เฟรม (255ms) แต่ถ้าเป็นอุปกรณ์ความเร็วต่ำคาบเวลาที่สั้นที่สุด (ความถี่สูงสุด) ที่สามารถวนอ่านได้คือ ทุกๆ 10 เฟรม (10ms) โฮสต์จะรู้คาบเวลาในการวนอ่านของเอ็นด์พอยต์ที่ใช้วิธีการถ่ายทอดสัญญาณแบบอินเตอร์รัพต์จากการอ่านเอ็นด์พอยต์ดิสคริปเตอร์

การส่งข้อมูลแต่ละครั้งสามารถกำหนดให้ส่งได้มากที่สุด 64 ไบต์ (ถ้าหากเป็นอุปกรณ์ความเร็วต่ำจะส่งได้สูงสุด 8 ไบต์) หากข้อมูลที่ต้องการส่งทั้งไม่มีมากกว่าก็จะต้องแบ่งเป็นส่วนๆ โดยแต่ละส่วนจะต้องส่งให้เต็มจำนวนข้อมูลที่สามารถส่งได้สูงสุด (เพื่อให้โฮสต์รับทราบว่ายังส่งข้อมูลไม่ครบ) แล้วในการส่งครั้งสุดท้ายซึ่งจะส่งไม่เต็มจำนวนสูงสุดจะเป็นตัวบอกให้โฮสต์ทราบว่าจบการส่งข้อมูลแล้ว

อย่างไรก็ตามการถ่ายทอดสัญญาณแบบอินเตอร์รัพต์นี้ก็ต้องการอัตราการอ่านข้อมูลที่คงที่คล้ายกับการแบบไอโซโครนัส เพราะถ้าหากโฮสต์ไม่สามารถวนอ่านข้อมูลได้ตามที่แจ้งไว้ในเอ็นด์พอยต์ดิสคริปเตอร์ อาจทำให้เกิดการล้นของข้อมูลที่ต้อการจะส่งจากอุปกรณ์กลับมายังโฮสต์ได้หรือเกิดโอเวอร์โฟลว เนื่องจากโฮสต์ไม่อ่านข้อมูลออกไปจนกระทั่งข้อมูลที่เตรียมจะส่งล้นบัฟเฟอร์ ดังนั้นก่อนเริ่มการทำงานของอุปกรณ์โฮสต์จะต้องอ่านเอ็นด์พอยต์ดิสคริปเตอร์ออกมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

แล้วพิจารณาว่าระบบสามารถรองรับการทำงานได้หรือไม่ ถ้าได้ก็จะตั้งค่าที่จำเป็นในการทำงานแล้วเริ่มการทำงาน แต่ถ้าไม่สามารถรองรับได้ก็จะไม่ตั้งค่าให้และปิดการทำงานของพอร์ตนั้น

การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์นี้ถึงแม้จะต้องการอัตราการวนอ่านข้อมูลที่คงที่แต่ก็ไม่จำเป็นต้องซิงโครไนซ์อัตราการส่งข้อมูลเข้ากับระบบ ดังนั้นเมื่อเกิดความผิดพลาดของการส่งข้อมูลขึ้นก็สามารถส่งใหม่ได้ ทำให้การถ่ายทอดสัญญาณแบบนี้จึงต้องมีการตรวจสอบความถูกต้องของข้อมูลและมีการส่งข้อมูลใหม่ในรอบการวนอ่านถัดไปหากเกิดความผิดพลาดในการส่ง ตัวอย่างของอุปกรณ์ที่ใช้การถ่ายทอดสัญญาณแบบนี้คือ คีย์บอร์ดและเมาส์ เป็นต้น

3. การถ่ายทอดสัญญาณแบบบัลก์

เป็นการรับส่งข้อมูลผ่านพอร์ต USB สำหรับงานที่ไม่ต้องการอัตราเร็วในการถ่ายทอดสัญญาณที่คงที่ อุปกรณ์ที่เห็นได้ชัดเจนที่สุดสำหรับการถ่ายทอดสัญญาณในลักษณะนี้คือ เครื่องพิมพ์ เพราะข้อมูลที่ผู้ใช้งานต้องการส่งไปพิมพ์นั้นไม่ต้องการความเร็วในการส่งที่สม่ำเสมอ การส่งช้าหรือเร็วก็จะส่งผลกระทบในด้านเวลาที่ต้องใช้มากขึ้นเท่านั้น แต่ข้อมูลที่ได้ก็ยังไม่เสียหาย เนื่องจากการถ่ายทอดสัญญาณแบบนี้ไม่สามารถจองการใช้งานแบนด์วิดธ์ของบัสได้ ดังนั้นการรับส่งข้อมูลจะเกิดขึ้นก็ต่อเมื่อบัสว่างจากการส่งข้อมูลอื่นๆ ทั้งหมดแล้วเท่านั้น หรือกล่าวอย่างง่ายๆ ว่าการถ่ายทอดสัญญาณแบบบัลก์จะเกิดขึ้นได้ก็ต่อเมื่อเหลือที่ว่างของข้อมูลในแต่ละเฟรมเท่านั้น หากไม่มีที่ว่างเหลือก็จะต้องรอไปเรื่อยๆ จนกว่าจะสามารถส่งได้ แต่ในทางกลับกันถ้าในบัสไม่มีอุปกรณ์ตัวใดใช้งานอยู่เลยก็จะสามารถส่งข้อมูลได้เต็มความสามารถที่บัสจะส่งได้ทันทีทำให้ความเร็วในการส่งข้อมูลสูงมาก อาจสูงถึง 1 เมกะไบต์ต่อวินาที (MByte/sec) เลยทีเดียว (ขนาดข้อมูล 1 ไบต์ = 8 บิต เพราะฉะนั้น 12 Mbit/sec จึงเท่ากับ 1.5 MByte/sec)

ถึงแม้ว่าบัสจะว่างมากเพียงใด แต่การถ่ายทอดสัญญาณแบบบัลก์ก็ไม่สามารถส่งข้อมูลขนาดใหญ่ไปในคราวเดียวได้ เพราะเสี่ยงต่อการเกิดความผิดพลาดของข้อมูลระหว่างส่ง ข้อมูลจะต้องถูกแบ่งออกเป็นส่วนๆ แล้วทยอยส่งต่อกันไปในเฟรมเดียวกัน โดยขนาดของชิ้นข้อมูลที่ใหญ่ที่สุดหลังจากแบ่งแล้วจะเท่ากับ 8, 16, 32 หรือ 64 ไบต์ การส่งข้อมูลที่ถูกแบ่งจะต้องส่งในขนาดใหญ่ที่สุดที่ตกลงกันไว้กับโฮสต์เพื่อให้โฮสต์ทราบว่า ยังไม่จบการส่งข้อมูลและโฮสต์จะทราบว่าจบการส่ง เมื่อพบว่าขนาดของข้อมูลที่ส่งไม่เต็มขนาดใหญ่สุดที่กำหนดไว้ (เหมือนกับแบบอินเตอร์รัพต์)

เนื่องจากอุปกรณ์ที่ใช้การถ่ายทอดสัญญาณแบบนี้ส่วนใหญ่ต้องการความถูกต้องของข้อมูลสูง เช่น ซีดีรอม (ในส่วนของการอ่านข้อมูลที่ไม่ใช่เพลง) เครื่องพิมพ์ หรือสแกนเนอร์ ดังนั้นการตรวจสอบข้อมูลจึงเป็นเรื่องจำเป็น ในการส่งข้อมูลแต่ละครั้งจึงมีการตรวจสอบความถูกต้องในทุกๆ ครั้ง และจะเกิดการส่งใหม่ถ้ามีความผิดพลาดเกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

ดิสคริปเตอร์และกระบวนการตั้งค่า

1. ข้อมูลภายในดิสคริปเตอร์

จะกล่าวถึงรายละเอียดแบบเจาะลึกกว่าแต่ละไบต์ของข้อมูลข้างในเก็บค่าอะไรอยู่บ้าง

1. ดีไวซ์ดิสคริปเตอร์ มีรายละเอียดดังในตารางที่ 1.3

2. คอนฟิกิวเรชั่นดิสคริปเตอร์ มีรายละเอียดดังตารางที่ 1.4

3. อินเตอร์เฟสดิสดิสคริปเตอร์ มีรายละเอียดดังในตารางที่ 1.5

4. เอ็นพอยต์ดิสคริปเตอร์ มีรายละเอียดดังในตารางที่ 1.6

5. ฮับคลาสดิสคริปเตอร์เป็นดิสคริปเตอร์ใช้สำหรับรายงานข้อมูลของอุปกรณ์ที่มาเชื่อมต่อพอร์ตต่างๆ กลับไปยังโฮสต์ มีรายละเอียดดังในตารางที่ 1.7

2. ขั้นตอนการตั้งค่าอุปกรณ์ (Configuration process)

จากเนื้อหาที่ผ่านมาทั้งหมด ทำให้ทราบถึงส่วนประกอบต่างๆ ทั้งหมดของ USB ต่อไปจะเป็นขั้นตอนการทำงานของระบบ เริ่มตั้งแต่มีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในระบบว่ามีลำดับขั้นอย่างไรบ้าง การสื่อสารข้อมูลระหว่างโฮสต์และตัวอุปกรณ์ตั้งแต่อุปกรณ์ถูกเชื่อมต่อกับระบบไปจนถึงตัวอุปกรณ์สามารถทำงานตามหน้าที่ของมันได้ ขั้นตอนทั้งหมดนี้เรียกรวมว่า การตั้งค่า (Config) อุปกรณ์เข้ากับระบบ

เมื่อมีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในระบบ ก่อนที่ตัวอุปกรณ์จะทำงานได้ โฮสต์จำเป็นต้องทราบข้อมูลที่ใช้ควบคุมการทำงานของตัวอุปกรณ์นั้นๆ รวมไปจนถึงความต้องการทรัพยากรด้านต่างๆ ไม่ว่าจะเป็นไฟเลี้ยงหรือความเร็วในการรับส่งข้อมูลเพื่อพิจารณาว่า ระบบ ณ ขณะนั้นสามารถรองรับการทำงานของตัวอุปกรณ์ใหม่นี้ได้หรือไม่ ซึ่งข้อมูลต่างๆ เหล่านี้จะบรรจุอยู่ในดิสคริปเตอร์ต่างๆ ภายในตัวอุปกรณ์ กระบวนการในการอ่านดิสคริปเตอร์ต่างๆ นี้ เรียกว่า การร้องขอข้อมูล (Data request) ซึ่งจะกล่าวโดยละเอียดในหัวข้อถัดไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 1.3 รายละเอียดข้อมูลแต่ละไบต์ของดีไวซ์ดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ความยาวของดีไวซ์ดิสคริปเตอร์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น Device Descripter = 01H
2	USB	2	BCD	เวอร์ชันของ USB ที่อุปกรณ์นี้ใช้งาน เก็บในรูปแบบของ BCD
4	Device class	1	Class	บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสใด
5	Device subclass	1	Subclass	บอกการจัดหมวดหมู่ของอุปกรณ์ ว่าอยู่ในซับคลาสใด
6	Device Protocal	1	Protocal	บอกรูปแบบการติดต่อหรือรหัสโปรโตคอลของอุปกรณ์
7	MaxPacket Size0	1	จำนวน	บอกขนาดข้อมูลที่สามารถส่งได้ต่อแพ็กเก็ตของเอ็นด์พอยต์ 0 (ค่าที่ใช้งานคือ 8,16,32 หรือ 64 ไบต์เท่านั้น)
8	Vendor	2	ID	ค่า ID ของ Vendor
10	Product	2	ID	ชนิดของอุปกรณ์ที่ผลิต
12	Device	2	BCD	เวอร์ชันของตัวอุปกรณ์ เก็บในรูปแบบ BCD
14	Manufac-Turer	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์ที่แสดงข้อมูลและชื่อผู้ผลิต
15	Product	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์แสดงชนิดของอุปกรณ์
16	Serial number	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์แสดงรุ่นของอุปกรณ์
17	Num Configuration	1	จำนวน	บอกจำนวนคอนฟิกิวเรชั่นทั้งหมดที่อุปกรณ์รองรับได้

ก่อนจะอธิบายขั้นตอนการตั้งค่าจะขอขยายความในเรื่องโครงสร้างบัสของ USB เพิ่มเติมอีกเล็กน้อย เพื่อไม่ให้เกิดความสับสนจากคำว่า "บัส" การเชื่อมต่ออุปกรณ์แบบบัสนั้นอุปกรณ์ทุกตัวในระบบจะได้รับข้อมูลทุกๆ แพ็กเก็ต แต่จะเลือกเก็บเฉพาะข้อมูลที่เป็นของตัวเองเท่านั้น โดยสังเกตจากแอดเดรสที่นำหน้าของแต่ละแพ็กเก็ตซึ่งถ้าตีความจากข้อความข้างต้นฮับซึ่งเป็นอุปกรณ์ที่ขยายการเชื่อมต่อจะทำหน้าที่กระจายข้อมูลออกไปยังอุปกรณ์ทุกๆ ตัวที่มาต่ออยู่เท่านั้น เช่นเดียวกับฮับของระบบเน็ตเวิร์ก LAN (Local Area Network) แต่เนื่องจาก USB มีสถานะบางสถานะที่กำหนดจากระดับแรงดันภายในสายสัญญาณโดยตรง เช่น สถานะรีเซต สถานะประหยัดพลังงาน และสถานะเชื่อมต่อหรือปลดออกจากระบบ เป็นต้น ซึ่งสถานะเหล่านั้นไม่สามารถส่งผ่าน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

จากฮับรวมไปกับข้อมูลอื่นๆ ได้ ดังนั้นฮับของ USB จะต้องทำหน้าที่มากกว่าการกระจายข้อมูลตามปกติ นั่นคือต้องมีการตรวจสอบการเชื่อมต่อหรือปลดออกของอุปกรณ์ได้ รับคำสั่งรีเซตเพื่อรีเซตบัสเฉพาะพอร์ตที่กำหนด ดังนั้นถ้ากล่าวถึงการส่งสถานะพิเศษเหล่านี้

ตารางที่ 1.4 รายละเอียดข้อมูลแต่ละไบต์ของคอนฟิกิวเรชั่นดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของคอนฟิกิวเรชั่นดิสคริปเตอร์นี้ในหน่วยไบต์
1	Descripter type	1	ค่าคงที่	ระบุชนิดของดิสคริปเตอร์ถ้าเป็น Configuration descripter = 02H
2	TotalLength	2	จำนวน	ขนาดข้อมูลทั้งหมดที่ได้จากการอ่านดิสคริปเตอร์ (เมื่อส่งคำสั่งขออ่านคอนฟิกิวเรชั่นดิสคริปเตอร์ ข้อมูลที่ได้คือ ข้อมูลคอนฟิกิวเรชั่นดิสคริปเตอร์ทั้งหมดเรียงกันริ่มจาก คอนฟิกิวเรชั่นดิสคริปเตอร์ อินเตอร์เฟสดิสคริปเตอร์และเอ็นด์พอยต์ดิสคริปเตอร์)
4	Numinterface	1	จำนวน	บอกจำนวนอินเตอร์เฟสทั้งหมดที่อุปกรณ์รองรับได้
5	Configuration Value	1	จำนวน	ค่าหมายเลขคอนฟิกิวเรชั่นที่ใช้ในการเลือกคอนฟิกิวเรชั่น นี้
6	Configuration	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์ใช้อธิบายข้อมูลของคอนฟิกิวเรชั่น นี้
7	Attributes	1	กำหนดเป็นบิต	แต่ละบิตบอกคุณสมบัติของคอนฟิกิวเรชั่น ดังนี้ บิต D7 ใช้พลังงานจากบัส บิต D6 ใช้แหล่งงานของตัวเอง บิต D5 การรองรับ Remote wakeup บิต D4-D0 สงวนไว้ไม่ใช้งาน หากบิตใดมีค่าเป็น "1" คือมีคุณสมบัตินั้นๆ
8	MaxPower	2	มิลลิแอมป์	บอกค่ากระแสไฟฟ้าที่คอนฟิกิวเรชั่น นี้ต้องการเมื่อทำงานเต็มความสามารถของมันแต่ละหน่วยเท่ากับ 2 mA

ไปยังตัวอุปกรณ์จะหมายความว่าโฮสต์ส่งคำสั่งไปยังฮับที่ตัวอุปกรณ์เชื่อมต่ออยู่แล้วฮับ จะนำคำสั่งไปแปลงเป็นระดับสัญญาณในสายสัญญาณเฉพาะพอร์ตเท่านั้น ไม่ใช่การส่งสถานะออกมาจากโฮสต์โดยตรงแล้วกระจายไปยังอุปกรณ์ทุกๆ ตัว ลำดับขั้นตอนที่จะกล่าวถึงนี้เป็นลำดับการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

พื้นฐาน ในการใช้งานจริงอาจจะมีการตัดหรือเพิ่มบางขั้นตอน หรืออาจจะมีการสลับบางขั้นตอน เพื่อให้เกิดความเหมาะสมกับการทำงานแต่ละอย่าง

ขั้นที่ 1 ตรวจสอบสถานะการเชื่อมต่อและชนิดของอุปกรณ์

เมื่อมีอุปกรณ์ตัวใหม่เชื่อต่อเข้ามาในระบบฮับจะทราบจากการตรวจสอบระดับแรงดันของสายสัญญาณว่าอยู่ในสถานะใดซึ่งจะทราบด้วยว่า อุปกรณ์ที่เชื่อมต่อเข้ามานั้นเป็นอุปกรณ์ความเร็วสูงหรือความเร็วต่ำ หลังจากนั้นจะเซตแฟลกของพอร์ตนั้นๆ เพื่อรอการอ่านกลับจากโฮสต์ในขั้น ตอนนี้ยังไม่มีการจ่ายไฟเลี้ยงไปยังตัวอุปกรณ์

ตารางที่ 1.5 รายละเอียดข้อมูลแต่ละไบต์ของอินเตอร์เฟสดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของอินเตอร์เฟสดิสคริปเตอร์นี้ในหน่วยไบต์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น Interface Descripter = 04H
2	Interface Number	1	จำนวน	ค่าหมายเลขอินเตอร์เฟสที่ใช้เลือกอินเตอร์เฟสนี้
3	Alternate Setting	1	จำนวน	ค่าหมายเลข Alternate Setting ในแต่ละอินเตอร์เฟส
4	NumEnd Points	1	จำนวน	บอกจำนวนเอ็นด์พอยต์ที่ใช้งานในอินเตอร์เฟสนี้ (ไม่รวมเอ็นด์พอยต์ 0)
5	Interface class	1	Class	บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสใดในกรณีที่แต่ละอินเตอร์เฟสมีอยู่ในคลาสที่ต่างกัน
6	Interface subclass	1	Subclass	บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสย่อยใดในกรณีที่แต่ละอินเตอร์เฟสมีอยู่ในคลาสย่อยที่ต่างกัน
7	Interface Protocal	1	Interface	บอกรหัสโปรโตคอลของอุปกรณ์ว่าใช้โปรโตคอลลักษณะใด ในกรณีที่แต่ละอินเตอร์เฟสใช้โปรโตคอลที่ต่างกัน
8	Interface	1	อินเด็กซ์	ค่าอินเด็กซ์ของสตริงดิสคริปเตอร์ที่ใช้อธิบายข้อมูลของอินเตอร์เฟสนี้

ขั้นที่ 2 จ่ายไฟเลี้ยงไปยังพอร์ต

หลังจากโฮสต์ทราบว่า พอร์ตใดมีอุปกรณ์ตัวใหม่ต่อเข้ามา โดยอ่านจากแฟลกสถานะของฮับ โฮสต์จะสั่งให้ฮับจ่ายไฟเลี้ยงแก่พอร์ตของตัวอุปกรณ์ใหม่นั้น (ในกรณีที่ฮับไม่สามารถควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจ่ายไฟเลี้ยงได้สามารถข้ามขั้นตอนไปได้) โดยในขั้นตอนนี้อุปกรณ์จะดึงกระแสได้ไม่เกิน 100mA อุปกรณ์จะดึงกระแสได้เต็มที่ก็ต่อเมื่อได้รับการตั้งค่าจากโฮสต์แล้วเท่านั้น

ขั้นที่ 3 รีเซตพอร์ตของอุปกรณ์ที่เชื่อมใหม่

เมื่ออุปกรณ์ตัวใหม่ได้รับไฟเลี้ยงแล้วโฮสต์จะส่งสถานะรีเซตไปยังพอร์ตนั้นๆ โดยส่งคำสั่งไปยังฮับที่ควบคุมการทำงานของพอร์ตนั้นเมื่ออุปกรณ์ได้รับสถานะนี้แล้วจะต้องรีเซตตัวเองและปรับค่าต่างๆ ไปยังสถานะเริ่มต้น ทำให้แอดเดรสของอุปกรณ์ที่ได้รับการรีเซตไปอยู่ที่ "0" ทำให้โฮสต์สามารถติดต่ออุปกรณ์ตัวใหม่นี้ได้เพราะถ้าไม่มีการตั้งแอดเดรสใหม่โฮสต์จะไม่ทราบว่าต้องส่งคำสั่งไปที่ใด

ตารางที่ 1.6 รายละเอียดข้อมูลแต่ละไบต์ของเอ็นด์พอยน์ดิสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของอินเตอร์เฟสดิสคริปเตอร์นี้ในหน่วยไบต์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น End point Descripter = 05H
2	Endpoint Address	1	เอ็นด์พอยต์	บอกค่าแอดเดรสของเอ็นด์พอยต์และทิศทางการส่งข้อมูลดังนี้ บิต D0-D3 บอกหมายเลขแอดเดรสของเอ็นด์พอยต์ บิต D4-D6 สงวนไว้ไม่ใช้งาน บิต D7 บอกทิศทางการส่งข้อมูลของเอ็นด์พอยต์ - เป็น "0" คือส่งข้อมูลกลับมายังโฮสต์ - เป็น "1" คือรับข้อมูลจากโฮสต์
3	Attributes	1	กำหนดเป็นบิต	บอกชนิดของการรับส่งข้อมูลที่เอ็นด์พอยต์นี้ใช้ D1=0, D0=0 Control Transfer D1=0,D0=1 Isochronous Transfer D1=1, D0=0 Bulk Transfer D1=1 , D0=1 Interrupt Transfer บิตที่เหลือนอกจากนี้สงวนไว้ไม่ใช้งาน
4	MaxPacket Size	2	จำนวน	ขนาดของข้อมูลใหญ่ที่สุดที่เอ็นด์พอยต์นี้สามารถรับส่งได้
6	Interval	1	จำนวน	เวลาในการวนอ่านข้อมูลจากโฮสต์ หน่วยเป็นมิลลิวินาที (ms)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นที่ 4 โฮต์ร้องขอข้อมูลดีไวซ์คริปเตอร์จากตัวอุปกรณ์

หลังจากแอดเดรสของอุปกรณ์ตัวใหม่ถูกรีเซตเป็นศูนย์โฮสต์จะเริ่มร้องขอข้อมูลดิสคริปเตอร์ต่างๆ จากตัวอุปกรณ์เพื่อนำมาพิจารณว่าสามารถรองรับอุปกรณ์ตัวใหม่ได้หรือไม่ โดยดิสคริปเตอร์ตัวแรกที่จะถูกอ่านออกมาก็คือ ดีไวซ์ดิสคริปเตอร์ การอ่านดิสคริปเตอร์จากตัวอุปกรณ์ก็คือ การส่งคำสั่งร้องขอเข้าไปยังตัวอุปกรณ์ ปลายทางที่จะคอยรับคำสั่งคือ เอ็นด์พอยต์ควบคุม ที่มีหมายเลขเอ็นด์พอยต์เป็น “0” ดังนั้นการอ่านดีไวซ์ดิสคริปเตอร์คือการส่งคำสั่งร้องขอดีไวซ์ดิสคริปเตอร์ไปยังอุปกรณ์แอดเดรส “0” เอ็นด์พอยต์ “0”

ตารางที่ 1.7 รายละเอียดข้อมูลแต่ละไบต์ของฮับคลาสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
0	Length	1	จำนวน	ขนาดข้อมูลของอินเตอร์เฟสดิสคริปเตอร์นี้ ในหน่วยไบต์
1	Descripter type	1	ค่าคงที่	ค่าระบุชนิดของดิสคริปเตอร์ ถ้าเป็น Hub class Descripter = 29H
2	NbrPorts	1	จำนวน	บอกจำนวนพอร์ตทั้งหมดที่ให้อุปกรณ์อื่นๆ มาเชื่อมต่อกับฮับได้
3	Hub Characteristic	2	กำหนดเป็นบิต	D1-D0 บอกข้อมูลการจ่ายพลังงานของฮับ 00= เปิดจ่ายพลังงานทุกพอร์ตในครั้งเดียว 01= แยกจ่ายพลังงานเป็นพอร์ตๆ แยกกัน 10 และ 11= ไม่สามารถปิดการจ่ายพลังงานได้ D2 ระบุว่าเป็นคอมพาวน์ดีไวซ์หรือไม่ 0= ไม่เป็นคอมพาวน์ดีไวซ์ 1= เป็นคอมพาวน์ดีไวซ์ D4-D3 บอกข้อมูลการป้องกันกระแสเกิน 00= ป้องกันกระแสเกินแบบรวม 01= แยกป้องกันได้อิสระในแต่ละพอร์ต 10 และ 11= ไม่สามารถป้องกันการดึงกระแสเกินได้ใช้กับฮับที่อาศัยพลังงานจากบัสเท่านั้น
5	PowerOn2 PowerGood	1	จำนวน	บอกค่าเวลาที่ฮับใช้ในการตั้งค่าภายใน ตั้งแต่เริ่มจ่ายกระแสไฟฟ้าจนกระทั่งทำงานได้สมบูรณ์ สามารถรับคำสั่งได้ ใช้สำหรับกำหนดค่าหน่วงเวลาหลังจากจ่ายไฟเลี้ยง
6	HubContr Current	1	จำนวน	ค่ากระแสไฟฟ้าที่ใช้สำหรับฮับคอนโทรลเลอร์ หน่วยละ 2 mA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 1.7 (ต่อ) รายละเอียดข้อมูลแต่ละไบต์ของฮับคลาสคริปเตอร์

ข้อมูลไบต์	ชื่อ	ขนาด (ไบต์)	ชนิดข้อมูล	คำอธิบาย
7	Device Removable	1	กำหนดเป็นบิต	ขนาดข้อมูลของฟิลด์นี้ขึ้นอยู่กับจำนวนพอร์ตของฮับ เนื่องจากข้อมูล 1 บิต แทนพอร์ต 1 พอร์ต แต่ละบิตบอกชนิดของอุปกรณ์ว่าเป็นอุปกรณ์ที่สามารถถอดเปลี่ยนได้หรือต่อถาวร ถ้าบิตใดเป็น "0" แทนถึงอุปกรณ์ที่ถอดเปลี่ยนได้ แต่ถ้าเป็น "1" แทนถึงอุปกรณ์ที่ต่อถาวร
-	PortPwrCtr Mask	-	กำหนดเป็นบิต	ตำแหน่งของข้อมูลนี้ขึ้นอยู่กับขนาดของข้อมูล Device Removable ถ้าบิตใดเป็น "0" หมายความว่า พอร์ตนั้นควบคุมการจ่ายไฟแบบรวม แต่ถ้าเป็น "1" หมายความว่าพอร์ตนั้นแยกจ่ายไฟเลี้ยงเป็นอิสระ

ข้อมูลที่ต้องการเป็นค่าแรกคือ ขนาดแพ็กเก็ตข้อมูลใหญ่ที่สุดที่สามารถรับส่งได้ต่อครั้ง ซึ่งข้อมูลนี้จะอยู่ในฟิลด์ MaxPacketSize0 (ก่อนอ่านข้อมูลนี้ออกมากำหนดขนาดแพ็กเก็ตใหญ่ที่สุดเป็น 8 ไบต์)

ขั้นที่ 5 กำหนดแอดเดรสให้แก่ตัวอุปกรณ์

เมื่อทราบขนาดแพ็กเก็ตข้อมูลที่จะใช้ในการติดต่อจากการอ่านดีไวซ์ดิสคริปเตอร์แล้ว โฮสต์จะกำหนดแอดเดรสให้แก่ตัวอุปกรณ์ โดยใช้คำสั่ง Set_Address ซึ่งค่าแอดเดรสที่กำหนดขึ้นนี้จะเป็นแอดเดรสที่ไม่ซ้ำกับอุปกรณ์ตัวอื่นๆ ในระบบเลยเพื่อให้โฮสต์สามารถติดต่อสื่อสารกับอุปกรณ์แต่ละตัวได้

ขั้นที่ 6 โฮสต์อ่านดิสคริปเตอร์ต่างๆ เพื่อพิจารณาการรองรับได้ของระบบ

ก่อนที่โฮสต์จะตั้งค่าตัวอุปกรณ์โฮสต์จะต้องอ่านค่าดิสคริปเตอร์ต่างๆ ภายในตัวอุป่กรณ์เพื่อพิจารณาว่าระบบมีทรัพยากรเพียงพอตามที่ตัวอุปกรณ์ต้องการหรือไม่ ข้อมูลหลักๆ ที่จำเป็นในการพิจารณาคือ กระแสไฟฟ้าและความเร็วในการรับส่งข้อมูลที่ต้องการ โดยข้อมูลด้านกระแสไฟฟ้าจะได้จากฟิลด์ MaxPower ภายในคอนฟิกิวเรชั่นดิสคริปเตอร์ ส่วนข้อมูลด้านความเร็วจะได้จากฟิลด์ MaxPacketSize ภายในแต่ละเอ็นด์พอยต์ดิสคริปเตอร์

การพิจารณาเริ่มจากตรวจสอบว่าอุปกรณ์ตัวนี้มีโหมดการทำงานกี่โหมด (กี่คอนฟิกิวเรชั่น) จากข้อมูลในฟิลด์ Num-Configurations ของดีไวซ์ดิสคริปเตอร์ หลังจากนั้นส่งคำร้องขอคอนฟิกิวเรชั่นดิสคริปเตอร์ไปยังตัวอุปกรณ์ด้วยคำสั่งนี้อุปกรณ์จะส่งดิสคริปเตอร์ที่เหลือทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ออกมาว่าจะเป็นคอนฟิกิวเรชั่น อินเตอร์เฟส และเอ็นด์พอยต์ดิสคริปเตอร์ในครั้งเดียว เมื่อได้ข้อมูลทั้งหมดออกมาแล้วจะต้องนำมาตัดแยกเป็นแต่ละส่วน เมื่อแยกคอนฟิกิวเรชั่นดิสคริปเตอร์ออกมาได้แล้ว โฮสต์จะทราบว่า ถ้าตั้งค่าให้อุปกรณ์ทำงานในแต่ละโหมดจะต้องจ่ายกระแสออกไปเท่าใด จากข้อมูลในฟิลด์ MaxPower ทำให้โฮสต์ทราบว่า สามารถจ่ายไฟเลี้ยงให้แก่อุปกรณ์ในคอนฟิกิวเรชั่นใดได้บ้าง

ข้อมูลถัดมาคือ อินเตอร์เฟสดิสคริปเตอร์จะให้ข้อมูลรายละเอียดของแต่ละอินเตอร์เฟส (เปรียบเสมือนแต่ละหน้าที่การทำงาน) ว่าจัดอยู่ในคลาสใด ใช้โปรโตคอลพิเศษแบบใดในการติดต่อรวมไปถึงจำนวนเอ็นด์พอยต์ที่ต้องใช้ติดต่อในอินเตอร์เฟสนั้นๆ หลังจากการอินเตอร์เฟสดิสคริปเตอร์ข้อมูลที่ตามมาคือ เอ็นด์พอยต์ดิสคริปเตอร์ซึ่งจะแจ้งความต้องการด้านปริมาณการรับส่งข้อมูลว่าแต่ละเอ็นด์พอยต์รับส่งข้อมูลแบบใด และต้องการความเร็วในการรับส่งข้อมูลเท่าใด โดยชนิดของการรับส่งข้อมูลทราบได้จากฟิลด์ Attributes ส่วนความเร็วของการรับส่งข้อมูลทราบได้จากฟิลด์ MaxPacketSize

เมื่อได้ข้อมูลเหล่านี้มาโฮสต์จะนำไปเปรียบเทียบกับทรัพยากรที่ระบบเหลืออยู่ว่าสามารถรองรับการจ่ายไฟเลี้ยงและความเร็วในการรับส่งข้อมูลเพียงพอกับที่อุปกรณ์ต้องการหรือไม่หากพิจารณาแล้วว่าสามารถรองรับได้ก็จะข้ามไปสู่การตั้งค่าในขั้นถัดไปได้แต่ถ้าพิจารณาแล้วพบว่าระบบเหลือทรัพยากรไม่เพียงพอ โฮสต์จะพิจารณาคอนฟิกิวเรชั่นและอินเตอร์เฟสอื่นๆ ที่ต้องการทรัพยากรน้อยกว่าในกรณีที่อุปกรณ์นั้นมีคอนฟิกิวเรชั่นและอินเตอร์เฟสมากกว่าหนึ่ง แต่หากพิจารณาคอนฟิกิวเรชั่นหรืออินเตอร์เฟสทั้งหมดแล้วยังไม่สามารถรองรับความต้องการของอุปกรณ์ได้โฮสต์ก็จะหยุดการตั้งค่าอุปกรณ์ ซึ่งจะทำให้อุปกรณ์นั้นไม่สามารถทำงานได้

ขั้นที่ 7 โฮสต์กำหนดเลือกคอนฟิกิวเรชั่นและอินเตอร์เฟส

หลังจากพิจารณาแล้วว่าระบบสามารถรองรับความต้องการของอุปกรณ์ตัวใหม่ได้โฮสต์จะส่งคำสั่ง Set_Configuration และ Set_Interface โดยระบุหมายเลขคอนฟิกิวเรชันและหมายเลขอินเตอร์เฟสที่เลือกเข้าไปในตัวคำสั่งหลังจากสิ้นสุดขั้นตอนนี้ อุปกรณ์จะดึงไฟเลี้ยงจากฮับเต็มพิกัดที่ระบุไว้ในคอนฟิกิวเรชั่นดิสคริปเตอร์และเริ่มทำงานตามหน้าที่ของมัน ถือเป็นการจบกระบวนการตั้งค่าตัวอุปกรณ์

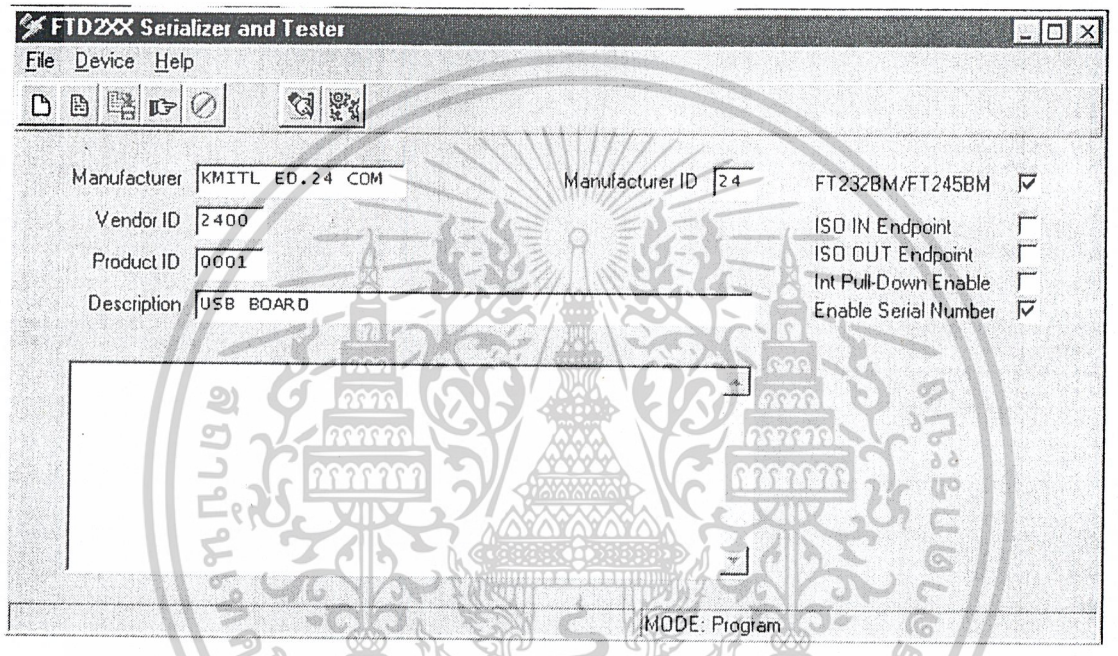
และสำหรับกรณีที่ต้องตั้งค่าอุปกรณ์หลายๆ ตัวพร้อมกัน เช่น การเปิดเครื่องคอมพิวเตอร์ขึ้นมาในขณะที่มีอุปกรณ์เชื่อมต่ออยู่มากกว่า 1ตัว โฮสต์จะทำการตั้งค่าอุปกรณ์ไปทีละตัวตามกระบวนการข้างต้นจนครบทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2.1 การเขียนข้อมูลของอุปกรณ์ลงใน EEPROM

ขั้นตอนการเขียนข้อมูลของอุปกรณ์ลงใน EEPROM นั้นสามารถกระทำได้โดยใช้โปรแกรม FTD2XX Serializer and Tester ซึ่งมีขั้นตอนดังนี้

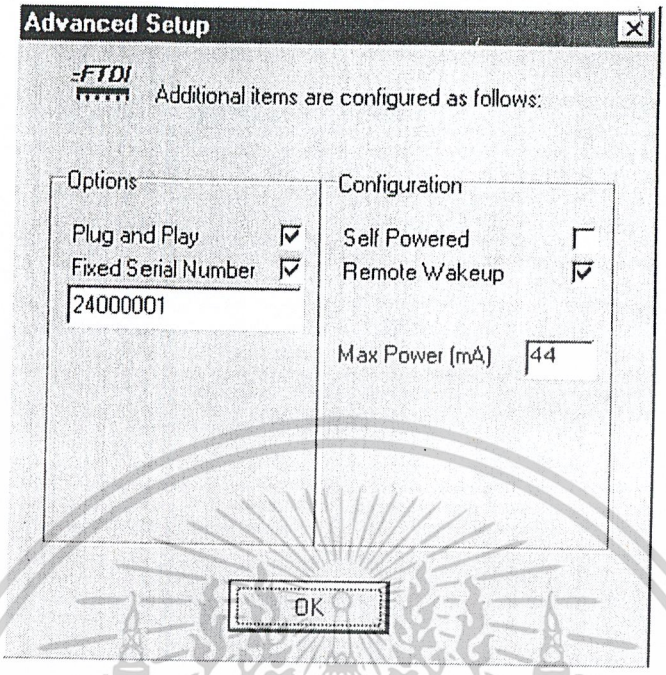
1. เปิดโปรแกรม FTD2XX Serializer and Tester
2. ใส่รายละเอียดต่างๆลงในช่อง เช่น ดังรูปที่ 1.10



รูปที่ 1.10 โปรแกรม FTD2XX Serializer and Tester

3. คลิกที่เมนู Device > Advanced Setup
4. ในส่วนนี้ให้ใส่รายละเอียดต่างๆดังรูปที่ 1.11

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.11 การใส่รายละเอียดในส่วนของ Advanced Setup

5. คลิกปุ่ม OK

6. คลิกปุ่ม Save

7. คลิกปุ่ม Program

8. ในส่วนของ Vendor ID และProduct ID นั้นหากมีการเปลี่ยนแปลงต่างจากเดิมต้องเข้าไปแก้ไขในไฟล์ ftd2xx.inf คือ ในส่วนที่เป็นหมายเลข VID และPID ต้องเปลี่ยนให้เหมือนกับที่ใส่ในช่อง Vendor ID และProduct ID

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 วงจรติดต่อพอร์ตอนุกรม

1. พอร์ตอนุกรม RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็ม (MODEM) เพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้ส่งผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งอยู่ห่างไกลโดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์ (Connector) เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3V จนถึง –12V แสดงว่ามีข้อมูล (Mark) และ +3V จนถึง +12V แสดงว่าเป็นช่องว่าง (Space) มีอัตราความเร็วการส่งข้อมูล 20 Kbit/sec

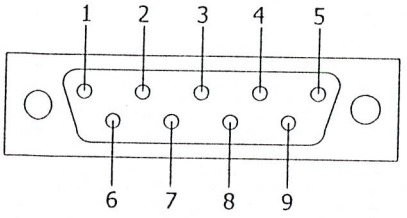
มาตรฐาน RS-232 ถูกใช้ในการกำหนดรูปแบบการสื่อสารข้อมูลกันระหว่างอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

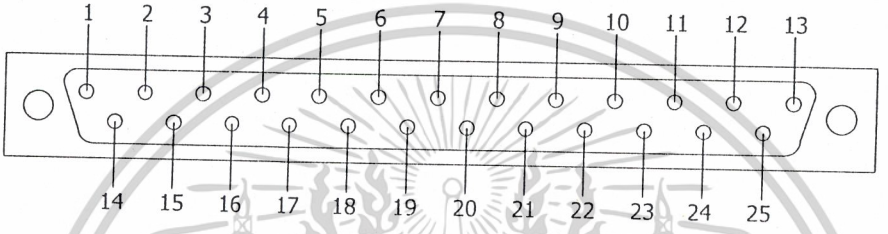
สำหรับการใช้งานคอมพิวเตอร์ พอร์ตอนุกรม RS-232 ถูกใช้เพื่อเชื่อมต่อกับโมเด็ม เมาส์ และเครื่องพิมพ์ที่สามารถติดต่อทางพอร์ตอนุกรมได้

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอเน็กเตอร์แบบ DB-25 ตัวผู้ หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยมีใช้งานมาในอดีตไม่ค่อยมีความสำคัญมากนักจึงถูกยกเลิกไป โดยแสดงตำแหน่งขาและรูปร่างในรูปที่ 1.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก) คอนเน็กเตอร์อนุกรม 9 ขา หรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)



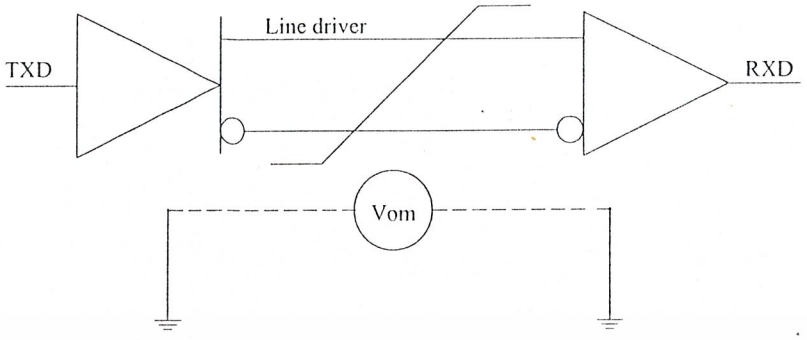
(ข) คอนเน็กเตอร์อนุกรม 25 ขา หรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์)

รูปที่ 1.12 คอนเน็กเตอร์อนุกรม

2. พอร์ตอนุกรม RS-422

RS-422 ได้รับการพัฒนามาจากรูปแบบ RS-232 แต่ยังคงใช้งานลักษณะจุดต่อจุด (Point to point) เช่นเดิมอยู่โดยเป็นระบบการส่งสัญญาณที่เป็นแบบผลต่างระหว่างสาย 2 เส้น ที่บิดเกลียวซึ่งกันและกัน ดังรูปที่ 1.13

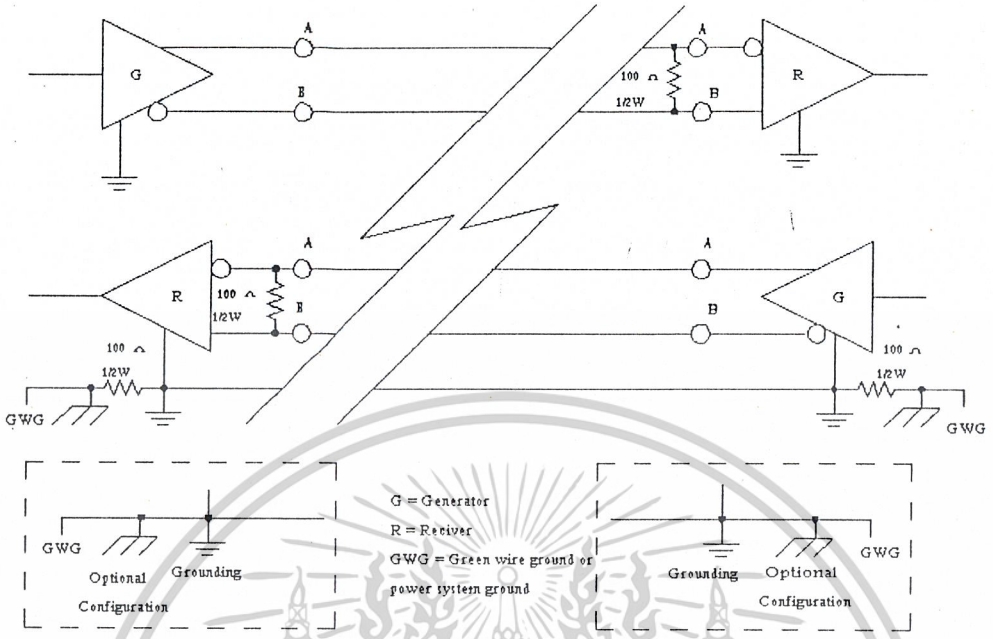
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.13 ระบบการส่งสัญญาณที่เป็นแบบผลต่าง

การส่งสัญญาณแบบนี้จะไม่มีสายกราวด์ แบบ RS-232 แต่จะสลับแรงดันบวกและลบออกมาแทนค่า "1" และ "0" ยกตัวอย่าง ให้เห็นได้ชัดเจนดังนี้ คือ ถ้าสายพันเกลียว 1 คู่ ประกอบด้วยสาย A และ สาย B เมื่อมีการส่งสัญญาณ "1" ออกมา สาย A จะมีแรงดันบวกและสาย B จะมีแรงดันไฟลบออกมา ส่วนถ้าต้องการส่งสัญญาณ "0" สาย A จะมีค่าแรงดันไฟเป็นลบและสาย B จะมีค่าแรงดันเป็นบวก โดยการส่งสัญญาณในลักษณะนี้ จะคล้ายกับการส่งสัญญาณไฟฟ้ากระแสสลับแบบสมดุล ซึ่งจะทำให้สัญญาณรบกวนต่ำและสามารถส่งสัญญาณไปได้ไกลกว่าเดิมถึง 4,000 ฟุต และมีอัตราความเร็วการส่งข้อมูลสูงได้มากถึง 10 เมกะบิตต่อวินาที (Mbit/sec) ข้อดีที่เห็นได้ชัด คือ การส่งสัญญาณรูปแบบนี้ จะมีสัญญาณรบกวนต่ำมาก เนื่องจากสัญญาณรบกวนจะส่งเข้ามาในสายสัญญาณแบบเฟส (Phase) เดียวกัน เมื่อเข้ามายังอินพุตของวงจรขยายแบบผลต่าง แล้วก็หักล้างกันหมดไป ลักษณะการใช้งาน RS-422 จริงจะใช้สายคู่บิดเกลียว 2 ชุด โดยจะใช้การส่งสัญญาณชุดส่งไปและส่งกลับ แยกจากกันโดยเด็ดขาดซึ่งคล้ายกับรูปแบบ RS-232 เดิม การส่งแบบสองทิศทางสมบูรณ์สามารถต่อตัวพ่วงได้ถึง 10 ตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

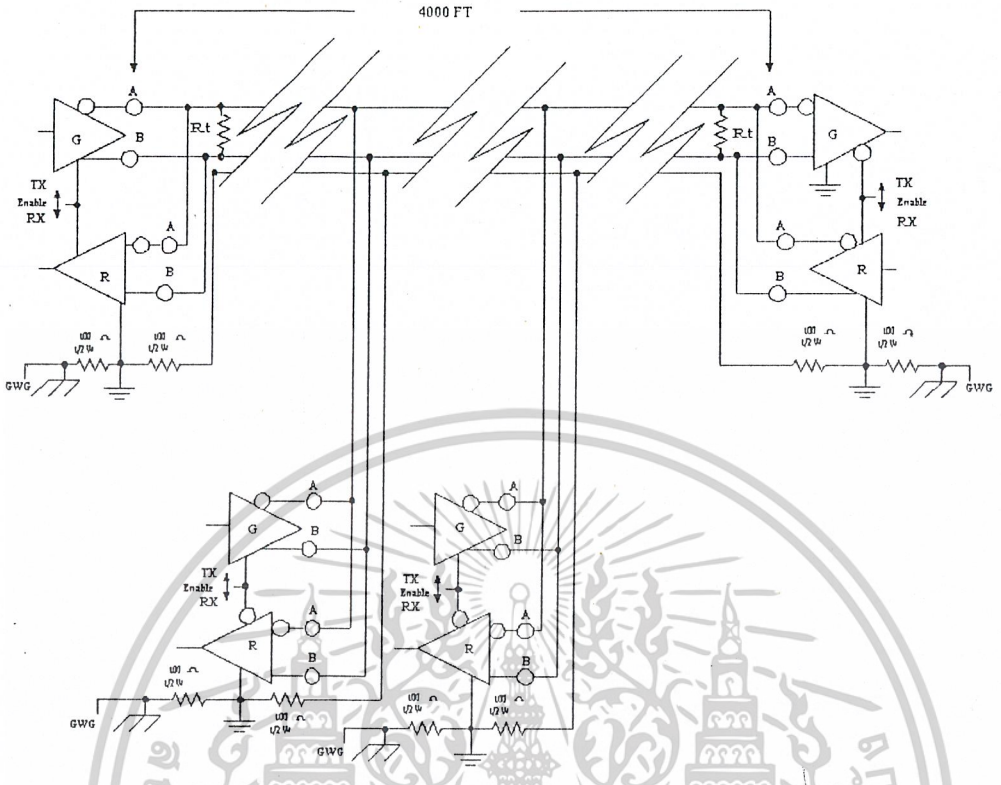


รูปที่ 1.14 การต่อสายสัญญาณแบบ RS-422 ใช้สาย 4 เส้น

3. พอร์ตอนุกรม RS-485

เป็นรูปแบบที่นำมาจากรูปแบบของ RS-422 ทุกประการ เพียงแต่มีการดัดแปลงให้ใช้สายสัญญาณเพียง 1 คู่สายบิดเกลียว โดยการใช้สายนี้เป็นทั้งสายส่งและสายรับในสายคู่เดียวกัน ซึ่งการที่จะทำงานในลักษณะนี้ได้นั้น ที่ขั้วของอุปกรณ์ทุกตัวจะต้องมีการต่อบัฟเฟอร์ที่สามารปรับให้เป็นความต้านสูงได้ คือตัดออกจากวงจรชั่วคราวได้นั่นเอง เพื่อใช้ในการบริหารสัญญาณ มักจะเป็นการสื่อสารในลักษณะ เครื่องลูกข่ายและเครื่องแม่ข่าย คือ มีเครื่องแม่ข่ายเป็นตัวหลักอยู่ 1 ตัว ซึ่งจะคอยควบคุมดูแลและสั่งการทั้งหมด ส่วนเครื่องลูกข่ายที่ต่ออยู่ทุกตัวก็จะรอรับการสั่งงานจากเครื่องแม่ข่ายเท่านั้นตลอดเวลา เครื่องลูกข่ายแต่ละตัวจะต้องมีชื่อเรียก เพื่อที่เครื่องแม่ข่ายจะใช้ในการสั่งงาน มีอีกวิธีการหนึ่ง ซึ่งได้ถูกพัฒนาขึ้นใหม่ คือ การสื่อสารกันโดยไม่มีเครื่องแม่ข่ายหลักคือทุกตัวในโครงข่ายเป็นเครื่องแม่ข่ายทั้งหมด แต่สามารถส่งข้อมูลด่วนออกมาควบคุมตัวอื่นได้โดยฉับพลัน จากการที่ได้ทำการทดลองได้ผลสถิติว่ารูปแบบหลังนี้สามารถลดการสื่อสารโดยไม่จำเป็นลงได้ถึง 30% ซึ่งขึ้นอยู่กับการพัฒนาซอฟต์แวร์ที่ซับซ้อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.15 การต่อสายสัญญาณแบบ RS-485 ใช้สาย 2 เส้น

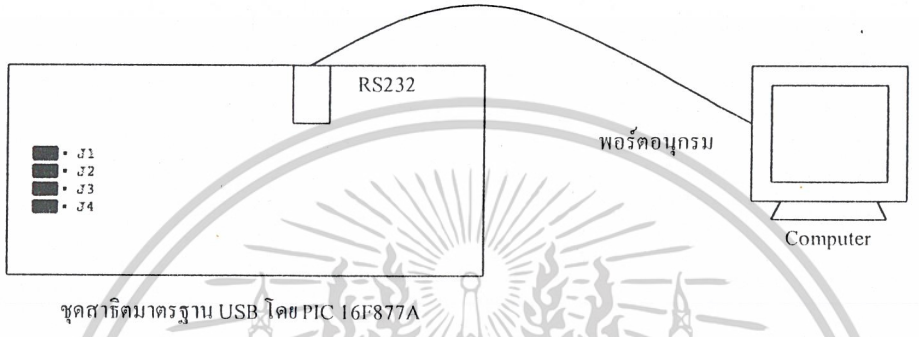
ตารางที่ 1.8 ตารางเปรียบเทียบของรูปแบบ RS-232, RS-422 และRS-485

ข้อมูลของแต่ละรูปแบบ	RS-232	RS-422	RS-485
หมวดการทำงาน	Single Ended	Differential	Differential
จำนวนตัวลูกสูงสุด	1 Drive 1 Recp	1 Drive 10 Recp	1 Drive 32 Recp
ความยาวสูงสุดของสายเคเบิ้ล	50 ฟุต	4,000 ฟุต	4,000 ฟุต
อัตราความเร็วการส่งข้อมูล	20 Kbit/sec	10 Mbit/sec	10 Mbit/sec
แรงดันส่งออกสูงสุดเอาต์พุต	+/-25 V	-0.25V ถึง +6V	-7V ถึง +12V
ค่าความต้านทานโหลด (Load)	3 – 7 กิโลโอห์ม (KΩ)	100 Ω	54 Ω
ความไวในการรับ	+/-3 V	+/-200 มิลลิโวลต์ (mV)	+/-200 mV
ความต้านทานอินพุต	3 – 7 KΩ	มากกว่า 4 KΩ	มากว่า 12 KΩ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเชื่อมต่อใช้งานชุด RS-232

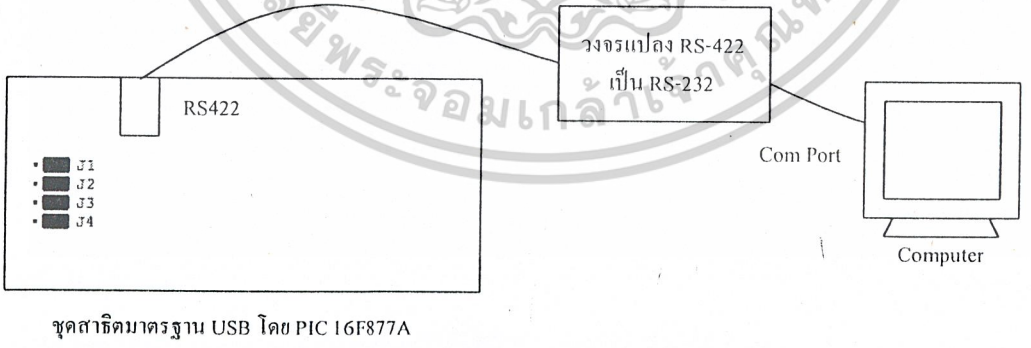
การต่อใช้งานนั้นสามารถต่อจากชุดสาธิตไปยังพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง โดยใช้สายที่ทำขึ้นมาเพื่อต่อกับชุด RS-232 ดังรูปที่ 1.16



รูปที่ 1.16 การเชื่อมใช้งานต่อ RS-232

การเชื่อมต่อใช้งานชุด RS-422

การต่อใช้งานนั้นต้องต่อจากชุดสาธิตไปยังชุดแปลง RS-422 เป็น RS-232 แล้วจึงค่อยต่อเข้าไปพอร์ตอนุกรมของคอมพิวเตอร์ดังรูปที่ 1.17

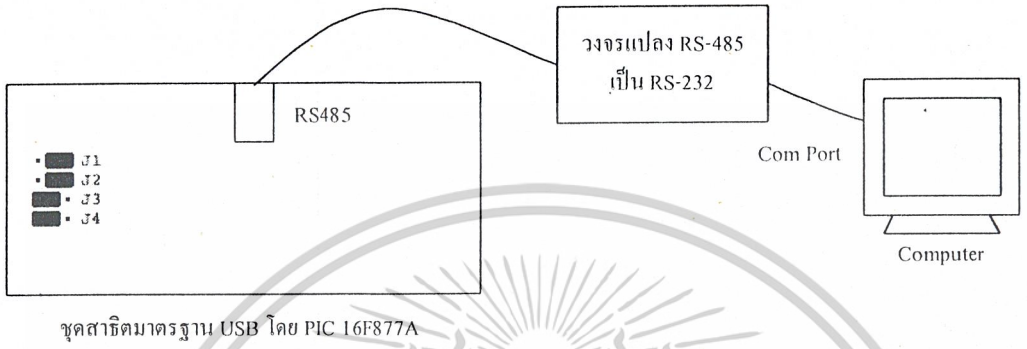


รูปที่ 1.17 การเชื่อมใช้งานต่อ RS-422

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเชื่อมต่อใช้งานชุด RS-485

การต่อใช้งานนั้นต้องต่อจากชุดสาธิตไปยังชุดแปลง RS-485 เป็น RS-232 แล้วจึงค่อยต่อเข้าไปพอร์ตอนุกรมของคอมพิวเตอร์ดังรูปที่ 1.18



รูปที่ 1.18 การเชื่อมใช้งานต่อ RS-485

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



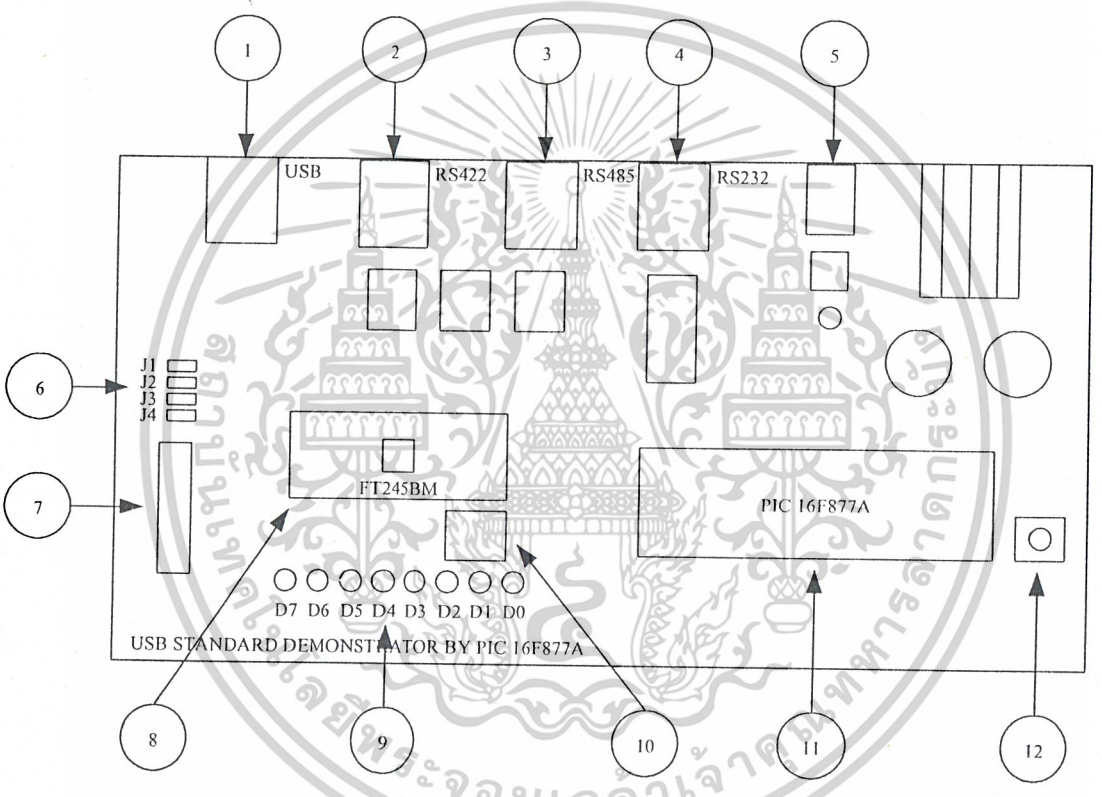
ภาคผนวก ช
คู่มือการใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. คำแนะนำเบื้องต้น

ก่อนที่จะใช้งานชุดสาธิตมาตรฐาน USB โดย PIC 16F877A ควรทำการศึกษาการใช้งานจากคู่มือให้เข้าใจเพื่อผลการทำงานที่ถูกต้อง และป้องกันการเสียหายที่อาจเกิดขึ้นกับชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

2. ส่วนประกอบ



รูปที่ ช.1 ส่วนประกอบของชุดสาธิตมาตรฐาน USB โดย PIC 16F877A

จากรูปที่ ช.1 มีรายละเอียดต่างๆดังนี้

① พอร์ต USB

② พอร์ต RS-422

③ พอร์ต RS-485

④ พอร์ต RS-232

⑤ จุดต่อแหล่งจ่ายไฟฟ้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

⑥ จัมเปอร์เลือกการใช้งาน RS-232, RS-422 และRS-485

⑦ จุดเชื่อมต่อภายนอก

⑧ USB โมดูล

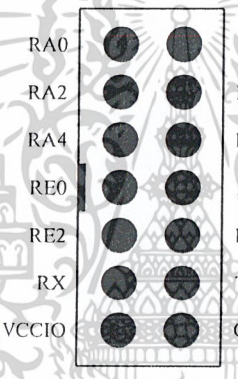
⑨ LED D0-D7

⑩ EEPROM

⑪ ไมโครคอนโทรลเลอร์ PIC 16F877A

⑫ สวิทช์รีเซต

จุดเชื่อมต่อภายนอกหมายเลข ⑪ มีตำแหน่งขาดังรูปที่ ช.2



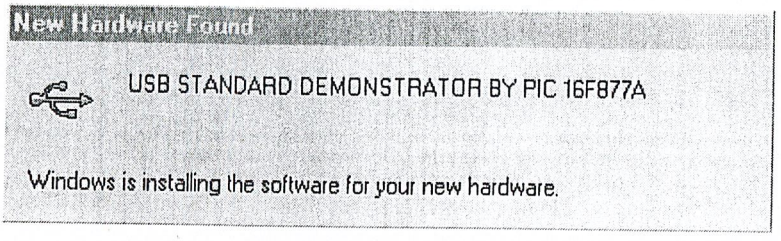
รูปที่ ช.2 ตำแหน่งขาของจุดเชื่อมต่อภายนอก

3. การติดตั้งและใช้งาน

3.1 การติดตั้งชุดสาธิตมาตรฐาน USB โดย PIC 16F877Aกับพอร์ต USB

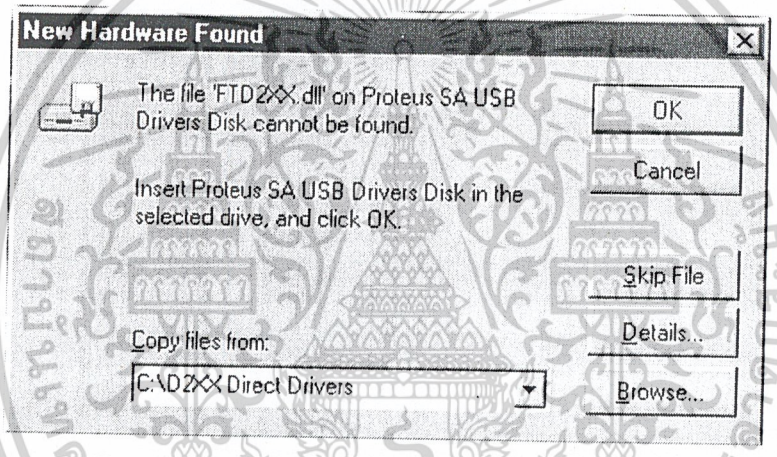
1. ต่อแหล่งจ่ายไฟฟ้าให้แก่ชุดสาธิต
2. เชื่อมชุดสาธิตเข้ากับคอมพิวเตอร์โดยผ่านพอร์ต USB
3. เมื่อคอมพิวเตอร์ตรวจสอบพบชุดสาธิตก็จะแสดงหน้าต่างดังรูปที่ ช.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ช.3 หน้าต่างแสดงเมื่อคอมพิวเตอร์ตรวจสอบพบชุดสาธิต

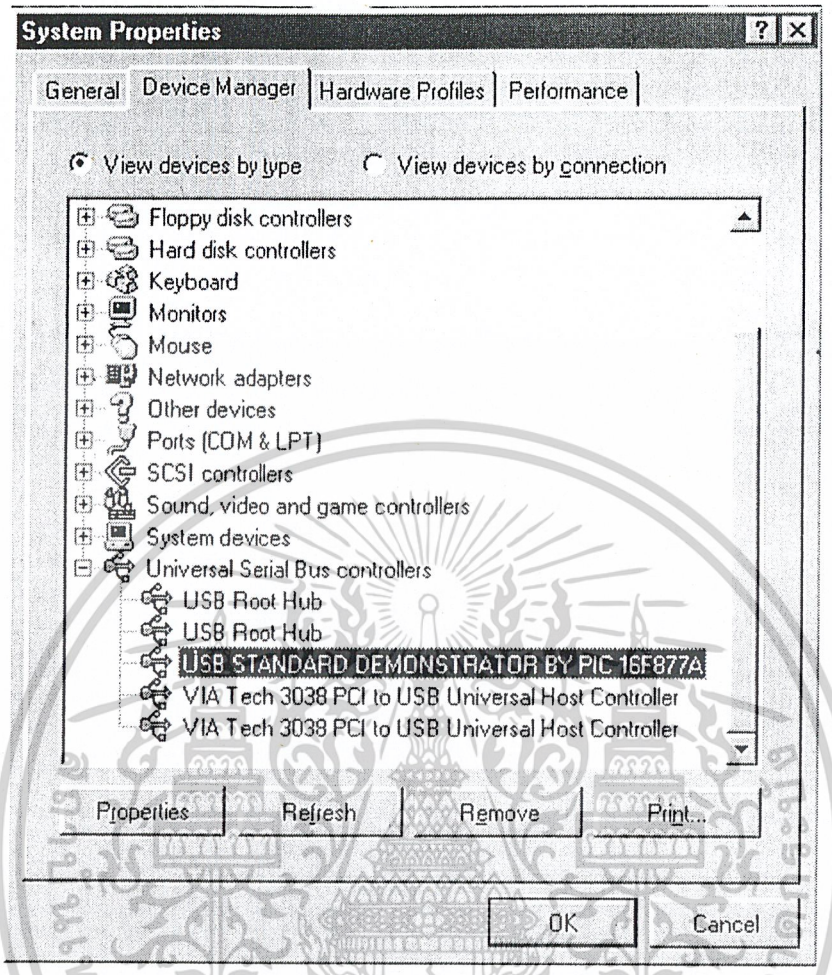
4. ทำการติดตั้งไดรเวอร์ดังรูปที่ ช.4



รูปที่ ช.4 หน้าต่างการติดตั้งไดรเวอร์

5. เมื่อทำการติดตั้งไดรเวอร์เสร็จ ก็จะปรากฏอุปกรณ์ดังรูปที่ ช.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



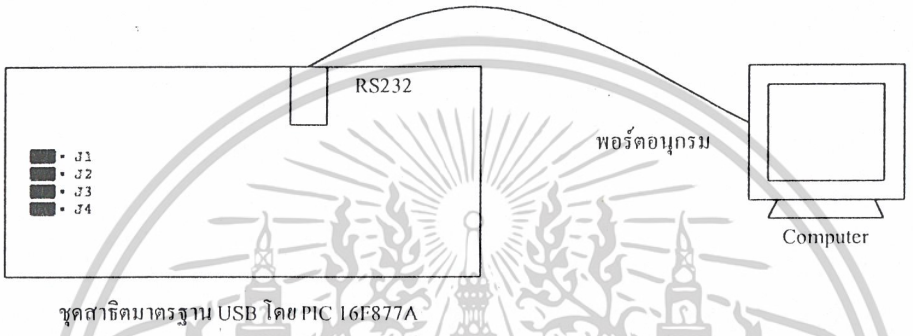
รูปที่ ช.5 รายชื่ออุปกรณ์เมื่อทำการติดตั้งเสร็จเรียบร้อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 การเชื่อมต่อชุดสาธิตมาตรฐาน USB โดย PIC 16F877A เข้ากับคอมพิวเตอร์โดยพอร์ต RS-232, RS-422 และRS-485

3.2.1 การเชื่อมต่อใช้งานชุด RS-232

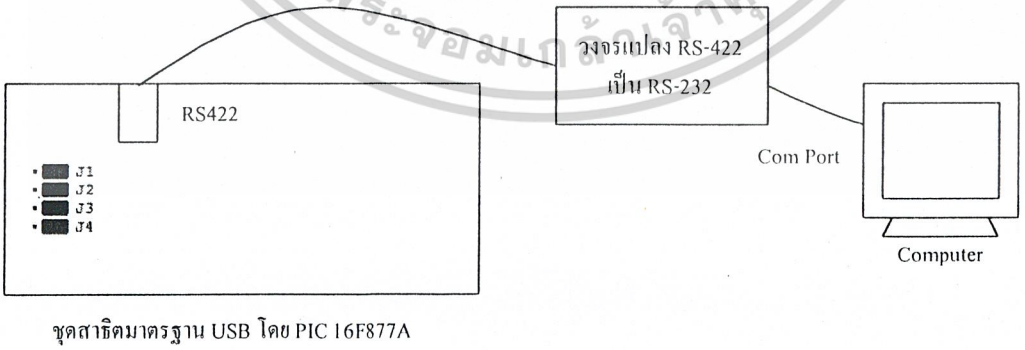
การต่อใช้งานนั้นสามารถต่อจากชุดสาธิตไปยังพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยใช้สายที่ทำขึ้นมาเพื่อต่อกับชุด RS-232 และต้องมีการเซตจัมเปอร์ดังรูปที่ ช.6



รูปที่ ช.6 การเชื่อมใช้งานต่อ RS-232

3.2.2 การเชื่อมต่อใช้งานชุด RS-422

การต่อใช้งานนั้นต้องต่อจากชุดสาธิตไปยังชุดแปลง RS-422 เป็น RS-232 แล้วจึงค่อยต่อเข้าไปพอร์ตอนุกรมของคอมพิวเตอร์และต้องมีการเซ็ตจัมเปอร์ดังรูปที่ ช.7

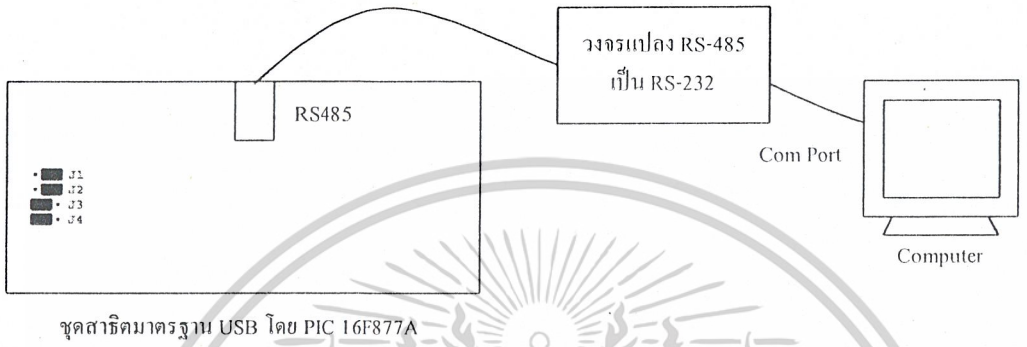


รูปที่ ช.7 การเชื่อมใช้งานต่อ RS-422

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 การเชื่อมต่อใช้งานชุด RS-485

การต่อใช้งานนั้นต้องต่อจากชุดสาธิตไปยังชุดแปลง RS-485 เป็น RS-232 แล้วจึงค่อยต่อเข้าไปพอร์ตอนุกรมของคอมพิวเตอร์และต้องมีการเซ็ตจัมเปอร์ดังรูปที่ ช.8



รูปที่ ช.8 การเชื่อมใช้งานต่อ RS-485

3.3 การใช้งานโปรแกรมเพื่อรับและส่งข้อมูล

มี 2 โปรแกรม ได้แก่ USB Test และSerial Test

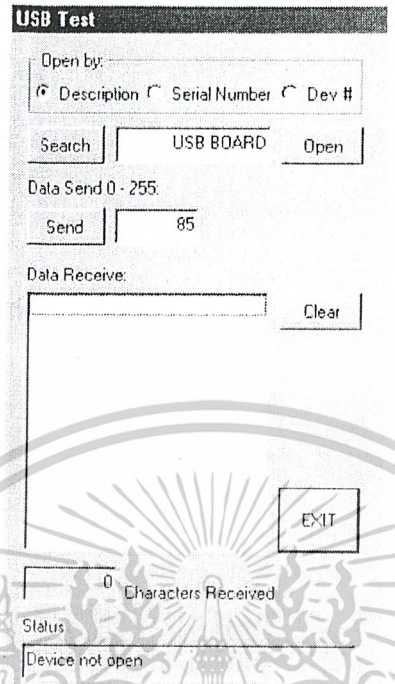
1. โปรแกรม USB Test เป็นโปรแกรมใช้ในการรับและส่งข้อมูลผ่านพอร์ต USB ซึ่งมีหน้าต่างดังรูปที่ ช.9

การติดตั้งโปรแกรม USB Test

1. ติดตั้งไดรเวอร์ของชุดสาธิต (หากไม่ติดตั้งเมื่อเปิดโปรแกรมจะเกิด Error ดูวิธีติดตั้งข้อที่ 3.1)

2. คัดลอกโปรแกรม USB Test ลงในเครื่องคอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



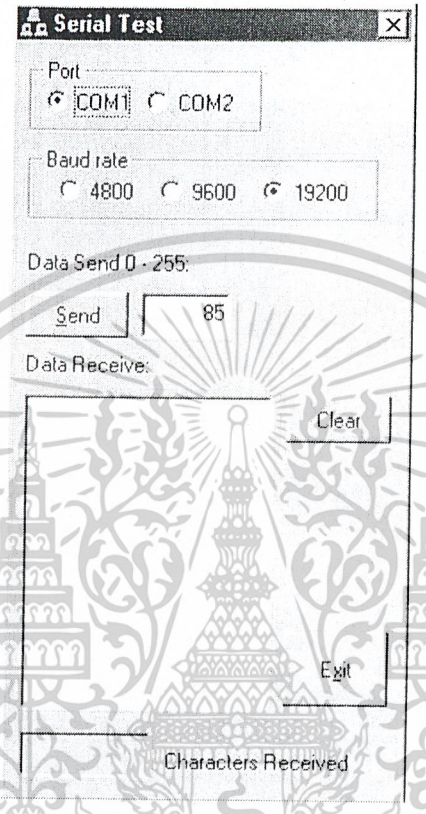
รูปที่ ช.9 โปรแกรม USB Test

การใช้งานโปรแกรม USB Test

1. ที่ Open by : จะมี Description, Serial Number และDev# ให้คลิกเลือก Description
2. คลิก Search เลือก USB BOARD
3. คลิกปุ่ม Open เพื่อทำการเชื่อมต่อกับอุปกรณ์ จะมีข้อความ Ready ที่แถบ Status
4. เมื่อต้องการส่งข้อมูล ให้ป้อนข้อมูลโดยมีค่าตั้งแต่ 0-255 ในช่อง Data Send 0-255 แล้วคลิกปุ่ม Send
5. ในการรับข้อมูล ข้อมูลที่ได้รับจะแสดงในช่อง Data Receive
6. หากต้องการลบข้อความที่อยู่ในช่อง Data Receive ให้คลิกปุ่ม Clear

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. โปรแกรม Serial Test เป็นโปรแกรมใช้ในการรับและส่งข้อมูลผ่านพอร์ตRS-232 ซึ่งมีหน้าต่างดังรูปที่ ช.10



รูปที่ ช.10 โปรแกรม Serial Test

การติดตั้งโปรแกรม Serial Test

1. คัดลอกไฟล์ Mscomm32.ocx ไว้ใน C:\windows\system
2. คัดลอกโปรแกรม Serial Test ลงในเครื่องคอมพิวเตอร์

การใช้งานโปรแกรม Serial Test

1. ที่ Port ให้เลือกพอร์ต COM ที่ต่ออยู่
2. ที่ Baud rate ให้เลือกอัตรา Baud rateที่ต้องการ
3. เมื่อต้องการส่งข้อมูล ให้ป้อนข้อมูลโดยมีค่าตั้งแต่ 0-255 ในช่อง Data Send 0-255 แล้วคลิกปุ่ม Send
4. ในการรับข้อมูลให้คลิกปุ่ม Receive ข้อมูลที่ได้รับจะแสดงในช่อง Data Receive

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. หากต้องการลบข้อความที่อยู่ในช่อง Data Receive ให้คลิกปุ่ม Clear

4. การแก้ปัญหาเบื้องต้น

เมื่อท่านประสบปัญหาในการใช้งานชุดสาธิตมาตรฐาน USB โดย PIC 16F877A สามารถตรวจสอบแนวทางการแก้ไขปัญหาเบื้องต้นได้จากตารางข้างล่างนี้

อาการ	สาเหตุและ/หรือวิธีแก้ไข
เกิดError "You must enter a Device Description or Serial Number"	ติดตั้งไดรเวอร์ของชุดสาธิตดูวิธีการติดตั้งข้อที่ 3.1
เปิดโปรแกรม Serial Test ไม่ได้ เกิด Error	คัดลอกไฟล์ MSCOMM32.OCX ไว้ใน C:\windows\system

5. การดูแลรักษาและข้อควรระวัง

5.1 การดูแลรักษา

- เก็บชุดสาธิตและสายให้เรียบร้อยหลังการใช้งาน

5.2 ข้อควรระวัง

- อย่าต่อแหล่งจ่ายไฟฟ้าเมื่อไม่ได้ใช้งาน
- อย่าให้วงจรหรือสายเกิดการช็อต

6. ข้อมูลจำเพาะ

คุณสมบัติ	รายละเอียด
ความเร็วสูงสุดในการส่งถ่ายข้อมูลกับพอร์ต USB	1 Mbytes/sec
ความเร็วสูงสุดในการส่งถ่ายข้อมูลกับพอร์ตอนุกรม	19,200 bit/sec
จำนวนพอร์ตเชื่อมต่อ	พอร์ต USB, พอร์ต RS-232, พอร์ต RS-422 และพอร์ต RS-485
ส่วนแสดงผล	ไดโอดแปลงแสงสีเขียว 8 ดวง
แหล่งจ่ายพลังงาน	ไฟฟ้ากระแสตรง 9 – 12 โวลต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.



ภาคผนวก ซ

รายละเอียดและคุณสมบัติของอุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

MICROCHIP

PIC16F87XA

FLASH Memory Programming Specification

This document includes programming specifications for the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

1.0 PROGRAMMING THE PIC16F87XA

The PIC16F87XA is programmed using a serial method. The Serial mode will allow the PIC16F87XA to be programmed while in the user's system. This allows for increased design flexibility. This programming specification applies to PIC16F87XA devices in all packages.

1.1 Programming Algorithm Requirements

The programming algorithm used depends on the operating voltage (V_{DD}) of the PIC16F87XA device, or whether internal or external timing is desired.

Algorithm #	V_{DD} Range	Timing
1	$2.0V \le V_{DD} < 5.5V$	Internal; 4 ms/op
2	$4.5V \le V_{DD} \le 5.5V$	External; 1 ms/op

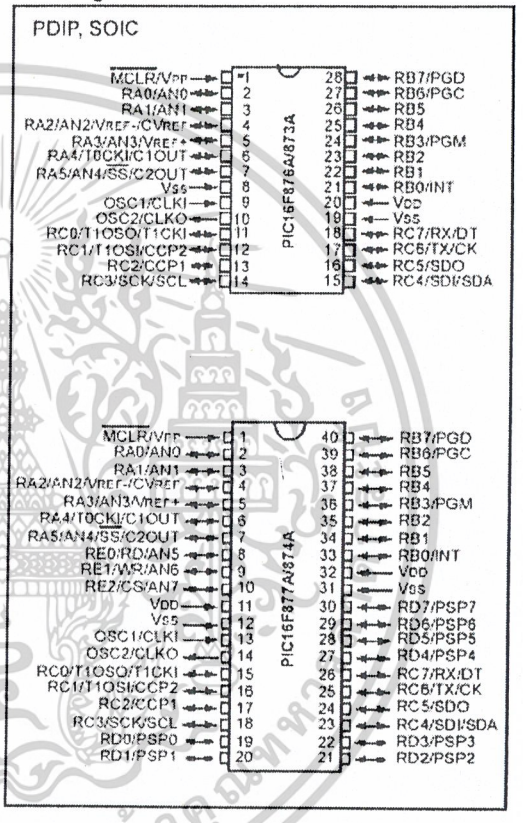
Both algorithms can be used with the two available programming entry methods. The first method follows the normal Microchip Programming mode entry of holding pins RB6 and RB7 low, while raising $\overline{MCLR}$ pin from V_{IL} to V_{IHH} (13V ± 0.5V). The second method, called Low Voltage ICSP™ or LVP for short, applies V_{DD} to $\overline{MCLR}$ and uses the I/O pin RB3 to enter Programming mode. When RB3 is driven to V_{DD} from ground, the PIC16F87XA device enters Programming mode.

1.2 Programming Mode

The Programming mode for the PIC16F87XA allows programming of user program memory, data memory, special locations used for ID, and the configuration word.

Pin Diagrams

PDIP, SOIC



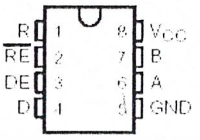
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SN65176B, SN75176B
DIFFERENTIAL BUS TRANSCEIVERS

SLLS101B – JULY 1985 – REVISED JUNE 1999

- Bidirectional Transceivers
- Meet or Exceed the Requirements of ANSI Standards TIA/EIA-422-B and TIA/EIA-485-A and ITU Recommendations V.11 and X.27
- Designed for Multipoint Transmission on Long Bus Lines in Noisy Environments
- 3-State Driver and Receiver Outputs
- Individual Driver and Receiver Enables
- Wide Positive and Negative Input/Output Bus Voltage Ranges
- Driver Output Capability . . . ±60 mA Max
- Thermal Shutdown Protection
- Driver Positive and Negative Current Limiting
- Receiver Input Impedance . . . 12 kΩ Min
- Receiver Input Sensitivity . . . ±200 mV
- Receiver Input Hysteresis . . . 50 mV Typ
- Operate From Single 5-V Supply

D OR P PACKAGE
(TOP VIEW)



description

The SN65176B and SN75176B differential bus transceivers are monolithic integrated circuits designed for bidirectional data communication on multipoint bus transmission lines. They are designed for balanced transmission lines and meet ANSI Standards TIA/EIA-422-B and TIA/EIA-485-A and ITU Recommendations V.11 and X.27.

The SN65176B and SN75176B combine a 3-state differential line driver and a differential input line receiver, both of which operate from a single 5-V power supply. The driver and receiver have active-high and active-low enables, respectively, that can be connected together externally to function as a direction control. The driver differential outputs and the receiver differential inputs are connected internally to form differential input/output (I/O) bus ports that are designed to offer minimum loading to the bus when the driver is disabled or V_{CC} = 0. These ports feature wide positive and negative common-mode voltage ranges, making the device suitable for party-line applications.

The driver is designed for up to 60 mA of sink or source current. The driver features positive and negative current limiting and thermal shutdown for protection from line-fault conditions. Thermal shutdown is designed to occur at a junction temperature of approximately 150°C. The receiver features a minimum input impedance of 12 kΩ, an input sensitivity of ±200 mV, and a typical input hysteresis of 50 mV.

The SN65176B and SN75176B can be used in transmission-line applications employing the SN75172 and SN75174 quadruple differential line drivers and SN75173 and SN75175 quadruple differential line receivers.

The SN65176B is characterized for operation from −40°C to 105°C and the SN75176B is characterized for operation from 0°C to 70°C.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 1999, Texas Instruments Incorporated

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FTDI Chip

FT245BM USB FIFO (USB - Parallel) I.C.

The FT245BM is the 2nd generation of FTDI's popular USB FIFO i.c. This device not only adds extra functionality to it's FT8U245AM predecessor and reduces external component count, but also maintains a high degree of pin compatibility with the original, making it easy to upgrade or cost reduce existing designs as well as increasing the potential for using the device in new application areas.

1.0 Features

HARDWARE FEATURES

- Single Chip USB ⇔ Parallel FIFO bi-directional Data Transfer
- Transfer Data rate to 1M Byte / Sec - D2XX Drivers
- Transfer Data rate to 3,000 Baud - VCP Drivers
- Simple to interface to MCU/ PLD / FPGA logic with a 4 wire handshake interface
- Entire USB protocol handled on-chip ... no USB-specific firmware programming required
- FTDI's royalty-free VCP and D2XX drivers eliminate the requirement for USB driver development in most cases.
- 384 Byte FIFO Tx buffer / 128 Byte FIFO Rx Buffer for high data throughput.
- New Send Immediate support via SENDI Pin for optimised data throughput
- Support for USB Suspend / Resume through PWREN# and WAKEUP pins.
- Support for high power USB Bus powered devices through PWREN# pin
- Adjustable RX buffer timeout
- In-built support for event characters
- Integrated level converter on FIFO and control signals for interfacing to 5v and 3.3v logic
- Integrated 3.3v regulator for USB IO
- Integrated Power-On-Reset circuit
- Integrated 6MHz – 48Mhz clock multiplier PLL
- USB Bulk or Isocronous data transfer modes
- New Bit-Bang Mode allows the data bus to be used as an 8 bit general purpose IO Port without the need for MCU or other support logic.
- 4.4v to 5.25v single supply operation
- UHCI / OHCI / EHCI host controller compatible
- USB 1.1 and USB 2.0 compatible
- USB VID, PID , Serial Number and Product Description strings in external EEPROM
- EEPROM programmable on-board via USB
- Compact 32LD PQFP package

VIRTUAL COM PORT (VCP) DRIVERS for

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP
- Windows CE **
- MAC OS-8 and OS-9
- MAC OS-X **
- Linux 2.40 and greater

D2XX (USB Direct Drivers + DLL S/W Interface)

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP

APPLICATION AREAS

- Easy MCU / PLD / FPGA interface to USB
- Upgrading Legacy Peripheral Designs to USB
- USB Instrumentation
- USB Industrial Control
- USB Audio and Low Bandwidth Video data transfer
- PDA ⇔ USB data transfer
- USB MP3 Player Interface
- USB FLASH Card Reader / Writers
- Set Top Box (S.T.B.) PC - USB interface
- USB Digital Camera Interface
- USB Hardware Modems
- USB Wireless Modems
-

[** = In planning or under development]

 © Future Technology Devices Intl. Ltd. 2002

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

1.1 General Description

The FT245BM provides an easy cost-effective method of transferring data to / from a peripheral and a host P.C. at up to 8 Million bits (1 Megabyte) per second. Its simple, FIFO-like design makes it easy to interface to any microcontroller or microprocessor via IO ports.

To send data from the peripheral to the host computer, simply write the byte-wide data into the module when TXE# is low. If the (384-byte) transmit buffer fills up or is busy storing the previously written byte, the device takes TXE# high in order to stop further data from being written until some of the FIFO data has been transferred over USB to the host.

When the host sends data to the peripheral over USB, the device will take RXF# low to let the peripheral know that at least one byte of data is available. The peripheral then reads the data until RXF# goes high indicating that no more data is available to read.

By using FTDI's virtual COM port drivers, the peripheral looks like a standard COM port to the application software. Commands to set the baud rate are ignored--the device always transfers data at its fastest rate regardless of the application's baud-rate setting. Alternatively, FTDI's D2XX drivers allow application software to access the device "directly" through a published DLL based API. Details of the current VCP and D2XX driver can be found on FTDI's web site (http://www.ftdichip.com)

2.0 Enhancements

This section summarises the enhancements of the 2nd generation device compared to it's FT8U245AM predecessor. For further details, consult the device pin-out description and functional descriptions.

- **Integrated Power-On-Reset (POR) Circuit**
  The device now incorporates an internal POR function. The existing RESET# pin is maintained in order to allow external logic to reset the device where required, however for many applications this pin can now be either left N/C or hard wired to VCC. In addition, a new reset output pin (RSTO#) is provided in order to allow the new POR circuit to provide a stable reset to external MCU and other devices. RSTO# was the TEST pin on the previous generation of devices.
- **Integrated RCCLK Circuit**
  In the previous devices, an external RC circuit was required to ensure that the oscillator and clock multiplier PLL frequency was stable prior to enabling the clock internal to the device. This circuit is now embedded on-chip – the pin assigned to this function is now designated as the TEST pin and should be tied to GND for normal operation.
- **Integrated Level Converter on FIFO Interface and control signals**
  The previous devices would drive the FIFO and control signals at 5v CMOS logic levels. The new device has a separate VCC-IO pin allowing the device to directly interface to 3.3v and other logic families without the need for external level converter i.c.'s
- **Power Management control for USB Bus Powered, high current devices**
  A new PWREN# signal is provided which can be used to directly drive a transistor or P-Channel MOSFET in applications where power switching of external circuitry is required. A new EEPROM based option makes the device pull gently down it's FIFO interface lines when the power is shut off (PWREN# is High). In this mode, any residual voltage on external circuitry is bled to GND when power is removed thus ensuring that external

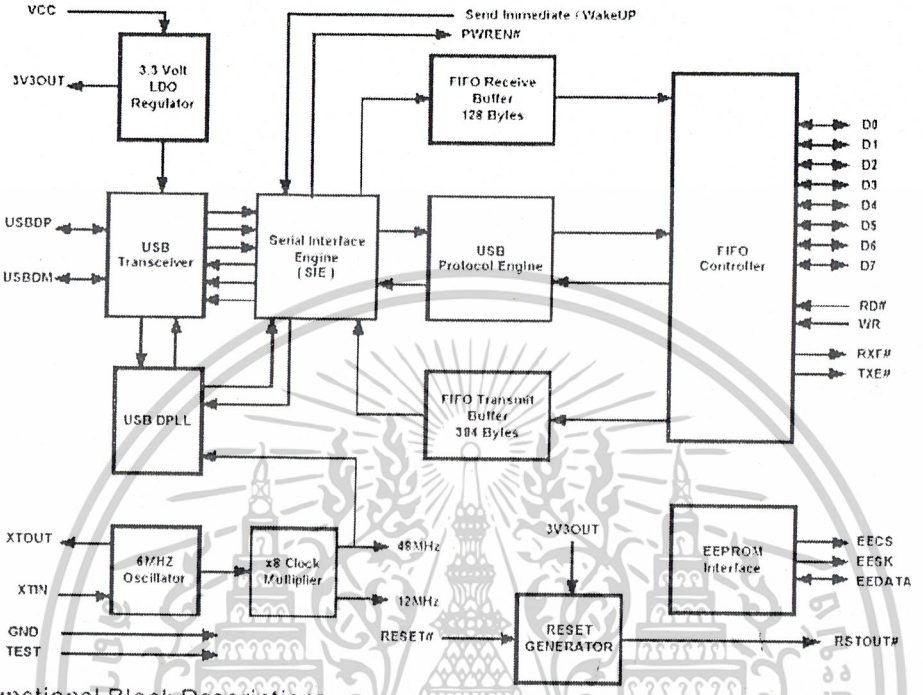
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

circuitry controlled by PWREN# resets reliably when power is restored. PWREN# can also be used by external circuitry to determine when USB is in suspend mode (PWREN# goes high).

- **Send Immediate / WakeUp (SI / WU) signal**
  The new Send Immediate / WakeUp signal combines two functions on a single pin. If USB is in suspend mode (and remote wakeup is enabled in the EEPROM), strobing this pin low will cause the device to request a resume from suspend (WakeUp) on the USB Bus. Normally, this can be used to wake up the Host PC. During normal operation, if this pin is strobed low any data in the device RX buffer will be sent out over USB on the next Bulk-IN request from the drivers regardless of the packet size. This can be used to optimise USB transfer speed for some applications.
- **Lower Suspend Current**
  Integration of RCCLK within the device and internal design improvements reduce the suspend current of the FT245BM to under 100uA typical (excluding the 1.5k pull-up on USB DP) in USB suspend mode. This allows greater margin for peripherals to meet the USB Suspend current limit of 500uA.
- **Support for USB Isocronous Transfers**
  Whilst USB Bulk transfer is usually the best choice for data transfer, the scheduling time of the data is not guaranteed. For applications where scheduling latency takes priority over data integrity such as transferring audio and low bandwidth video data, the new device now offers an option of USB Isocronous transfer via an option bit in the EEPROM.
- **Programmable FIFO TX Buffer Timeout**
  In the previous device, the TX buffer timeout used to flush remaining data from the TX buffer was fixed at 16ms timeout. This timeout is now programmable over USB in 1ms increments from 1ms to 255ms, thus allowing the device to be better optimised for protocols requiring faster response times from short data packets.
- **Relaxed VCC Decoupling**
  The 2nd generation devices now incorporate a level of on-chip VCC decoupling. Though this does not eliminate the need for external decoupling capacitors, it significantly improves the ease of pcb design requirements to meet FCC,CE and other EMI related specifications.
- **Bit Bang Mode**
  The 2nd generation device has a new option referred to as "Bit Bang" mode. In Bit Bang mode, the eight FIFO data lines can be switched between FIFO interface mode and an 8-bit Parallel IO port. Data packets can be sent to the device and they will be sequentially sent to the interface at a rate controlled by an internal timer (equivalent to the prescaler of the FT232BM device). As well as allowing the device to be used stand-alone as a general purpose IO controller for example controlling lights, relays and switches, some other interesting possibilities exist. For instance, it may be possible to connect the device to an SRAM configurable FPGA as supplied by vendors such as Altera and Xilinx. The FPGA device would normally be un-configured (i.e. have no defined function) at power-up. Application software on the PC could use Bit Bang Mode to download configuration data to the FPGA which would define it's hardware function, then after the FPGA device is configured the FT245BM can switch back into FIFO interface mode to allow the programmed FPGA device to communicate with the PC over USB. This approach allows a customer to create a "generic" USB peripheral who's hardware function can be defined under control of the application software. The FPGA based hardware can be easily upgraded or totally changed simply by changing the FPGA configuration data file. Application notes, software and development modules for this application area will be available from FTDI and other 3rd party developers.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ .

3.0 Block Diagram (simplified)



3.1 Functional Block Descriptions

- **3.3V LDO Regulator**

  The 3.3V LDO Regulator generates the 3.3 volt reference voltage for driving the USB transceiver cell output buffers. It requires an external decoupling capacitor to be attached to the 3V3OUT regulator output pin. It also provides 3.3v power to the RSTOUT# pin. The main function of this block is to power the USB Transceiver and the Reset Generator Cells rather than to power external logic. However, external circuitry requiring 3.3v nominal at a current of not greater than 5mA could also draw it's power from the 3V3OUT pin if required.

- **USB Transceiver**

  The USB Transceiver Cell provides the USB 1.1 / USB 2.0 full-speed physical interface to the USB cable. The output drivers provide 3.3 volt level slew rate control signalling, whilst a differential receiver and two single ended receivers provide USB data in, SEO and USB Reset condition detection.

- **USB DPLL**

  The USB DPLL cell locks on to the incoming NRZI USB data and provides separate recovered clock and data signals to the SIE block.

- **6MHz Oscillator**

  The 6MHz Oscillator cell generates a 6MHz reference clock input to the X8 Clock multiplier from an external 6MHz crystal or ceramic resonator.

- **x8 Clock Multiplier**

  The x8 Clock Multiplier takes the 6MHz input from the Oscillator cell and generates a 12MHz reference clock for the SIE, USB Protocol Engine and FIFO FIFO controller blocks. It also generates a 48MHz reference clock for the USB DPLL.

- **Serial Interface Engine (SIE)**

  The Serial Interface Engine (SIE) block performs the Parallel to Serial and Serial to Parallel conversion of the USB data. In accordance to the

 © Future Technology Devices Intl. Ltd. 2002

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

USB 1.1 specification. It performs bit stuffing / un-stuffing and CRC5 / CRC16 generation / checking on the USB data stream.

- **USB Protocol Engine**

  The USB Protocol Engine manages the data stream from the device USB control endpoint. It handles the low level USB protocol (Chapter 9) requests generated by the USB host controller and the commands for controlling the functional parameters of the FIFO.

- **FIFO Receive Buffer (128 bytes)**

  Data sent from the USB Host to the FIFO via the USB data out endpoint is stored in the FIFO Receive Buffer and is removed from the buffer by reading the FIFO contents using RD#

- **FIFO Transmit Buffer (384 bytes)**

  Data written into the FIFO using WR# is stored in the FIFO Transmit Buffer. The Host removes Data from the FIFO Transmit Data by sending a USB request for data from the device data in endpoint.

- **FIFO Controller**

  The FIFO Controller handles the transfer of data between the external FIFO interface pins and the FIFO Transmit and Receive buffers.

- **RESET Generator**

  The Reset Generator Cell provides a reliable power-on reset to the device internal circuitry on power up. An additional RESET# input and RSTOUT# output are provided to allow other devices to reset the FT245BM or the FT245BM to reset other devices respectively. During reset, RSTOUT# is high-impedance otherwise it drives out at the 3.3v provided by the onboard regulator. RSTOUT# can be used to control the 1k5 pull-up on USB DP directly where delayed USB enumeration is required. It can also be used to reset other devices. RSTOUT# will stay high-impedance for approximately 5ms after VCC has risen above 3.5v AND the device oscillator is running AND RESET# is high. RESET# should be tied to VCC unless it is a requirement to reset the device from external logic or an external reset generator i.c.

- **EEPROM Interface**

  Though the FT245BM will work without the optional EEPROM, an external 93C46 (93C56 or 93C66) EEPROM can be used to customise the USB VID, PID, Serial Number, Product Description Strings and Power Descriptor value of the FT245BM for OEM applications. The EEPROM is also required for applications where multiple FT245BM's are connected to a single PC as the drivers rely on a unique serial number for each device to bind a unique virtual COM port to each individual device. Other parameters controlled by the EEPROM include Remote Wake Up, Isochronous Transfer Mode, Soft Pull Down on Power-Off and USB 2.0 descriptor modes.

  The EEPROM should be a 16 bit wide configuration such as a MicroChip 93LC46B or equivalent capable of a 1Mb/s clock rate at VCC = 4.4v to 5.25v. The EEPROM is programmable-on board over USB using a utility available from FTDI's web site (http://www.ftdichip.com). This allows a blank part to be soldered onto the PCB and programmed as part of the manufacturing and test process.

  If no EEPROM is connected (or the EEPROM is blank), the FT245BM will use it's built-in default VID, PID Product Description and Power Descriptor Value. In this case, the device will not have a serial number as part of the USB descriptor.

 © Future Technology Devices Intl. Ltd. 2002

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

4.0 Device Pin-Out

Figure 1
Pin-Out
(LQFP-32 Package)

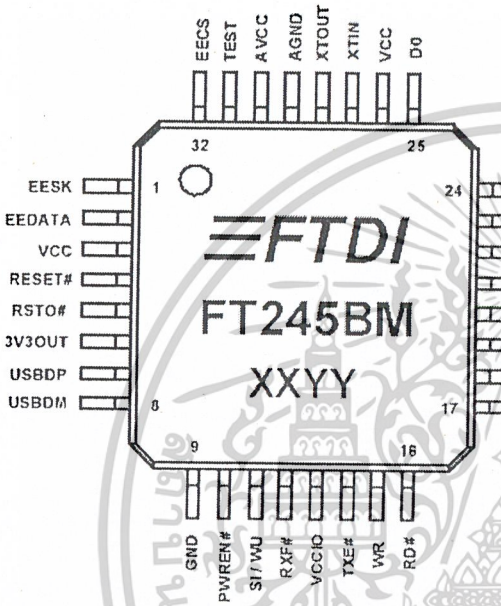
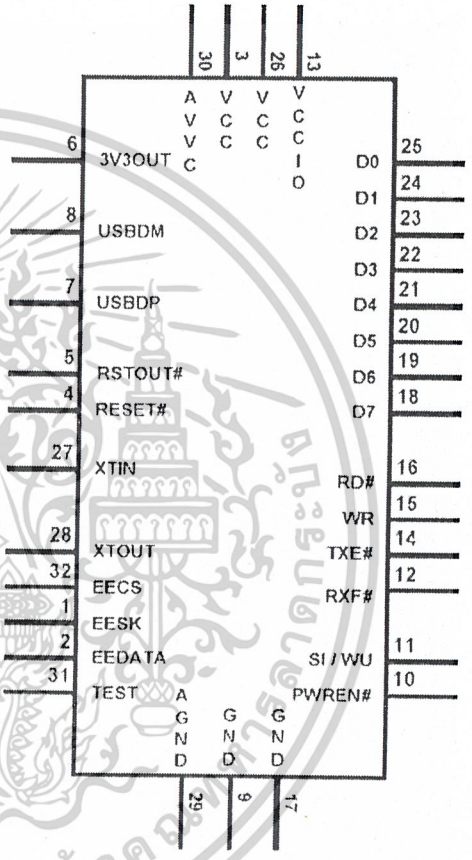


Figure 2
Pin-Out
(Schematic Symbol)



 © Future Technology Devices Intl. Ltd. 2002

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้.

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Pin Description

PIN					NAME	FUNCTION
MAX481/MAX483/MAX485/MAX487/MAX1487		MAX488/MAX490		MAX489/MAX491		
DIP/SO	µMAX	DIP/SO	µMAX	DIP/SO		
1	3	2	4	2	RO	Receiver Output: If A > B by 200mV, RO will be high; If A < B by 200mV, RO will be low.
2	4	—	—	3	$\overline{RE}$	Receiver Output Enable. RO is enabled when $\overline{RE}$ is low; RO is high impedance when RE is high.
3	5	—	—	4	DE	Driver Output Enable. The driver outputs, Y and Z, are enabled by bringing DE high. They are high impedance when DE is low. If the driver outputs are enabled, the parts function as line drivers. While they are high impedance, they function as line receivers if $\overline{RE}$ is low.
4	6	3	5	5	DI	Driver Input. A low on DI forces output Y low and output Z high. Similarly, a high on DI forces output Y high and output Z low.
5	7	4	6	6, 7	GND	Ground
—	—	5	7	9	Y	Noninverting Driver Output
—	—	6	8	10	Z	Inverting Driver Output
6	8	—	—	—	A	Noninverting Receiver Input and Noninverting Driver Output
—	—	8	2	12	A	Noninverting Receiver Input
7	1	—	—	—	B	Inverting Receiver Input and Inverting Driver Output
—	—	7	1	11	B	Inverting Receiver Input
8	2	1	3	14	Vcc	Positive Supply: 4.75V < Vcc < 5.25V
—	—	—	—	1, 8, 13	N.C.	No Connect—not internally connected

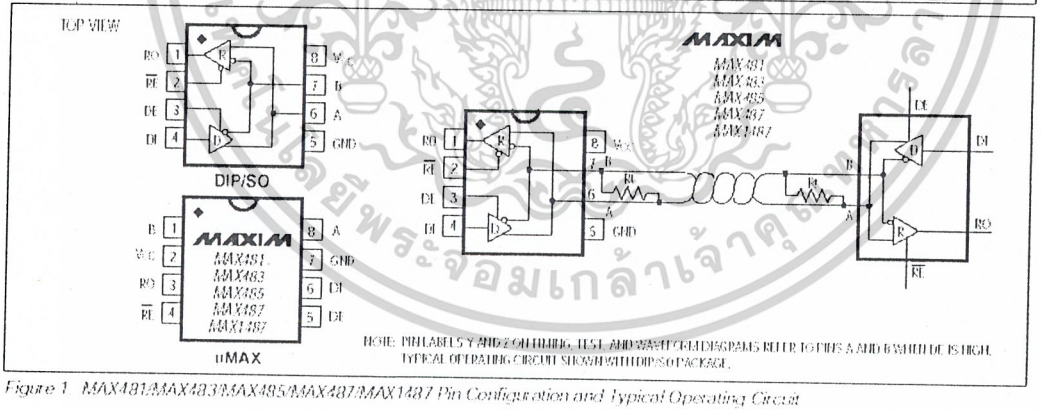


Figure 1. MAX481/MAX483/MAX485/MAX487/MAX1487 Pin Configuration and Typical Operating Circuit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

19-0175; Rev 3; 5/96

MAXIM

±15kV ESD-Protected, +5V RS-232 Transceivers

General Description

The MAX202E–MAX213E, MAX232E/MAX241E line drivers/receivers are designed for RS-232 and V.28 communications in harsh environments. Each transmitter output and receiver input is protected against ±15kV electrostatic discharge (ESD) shocks, without latchup. The various combinations of features are outlined in the *Selection Guide*. The drivers and receivers for all ten devices meet all EIA/TIA-232E and CCITT V.28 specifications at data rates up to 120kbps, when loaded in accordance with the EIA/TIA-232E specification.

The MAX211E/MAX213E/MAX241E are available in 28-pin SO packages, as well as a 28-pin SSOP that uses 60% less board space. The MAX202E/MAX232E come in 16-pin narrow SO, wide SO, and DIP packages. The MAX203E comes in a 20-pin DIP/SO package, and needs no external charge-pump capacitors. The MAX205E comes in a 24-pin wide DIP package, and also eliminates external charge-pump capacitors. The MAX206E/MAX207E/MAX208E come in 24-pin SO, SSOP, and narrow DIP packages. The MAX232E/MAX241E operate with four 1µF capacitors, while the MAX202E/MAX206E/MAX207E/MAX208E/MAX211E/MAX213E operate with four 0.1µF capacitors, further reducing cost and board space.

Applications

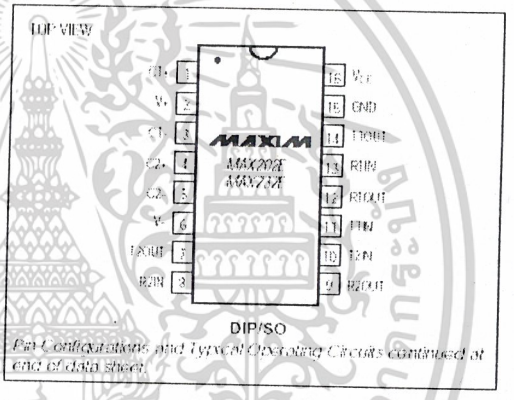
Notebook, Subnotebook, and Palmtop Computers
Battery-Powered Equipment
Hand-Held Equipment

Ordering Information appears at end of data sheet.

Features

- ESD Protection for RS-232 I/O Pins:
  ±15kV—Human Body Model
  ±8kV—IEC1000-4-2, Contact Discharge
  ±15kV—IEC1000-4-2, Air-Gap Discharge
- Latchup Free (unlike bipolar equivalents)
- Guaranteed 120kbps Data Rate—LapLink™ Compatible
- Guaranteed 3V/µs Min Slew Rate
- Operate from a Single +5V Power Supply

Pin Configurations



Pin Configurations and Typical Operating Circuits continued at end of data sheet.

Selection Guide

PART	No. of RS-232 DRIVERS	No. of RS-232 RECEIVERS	RECEIVERS ACTIVE IN SHUTDOWN	No. of EXTERNAL CAPACITORS	LOW-POWER SHUTDOWN	TTL THREE-STATE
MAX202E	2	2	0	4 (0.1µF)	No	No
MAX203E	2	2	0	None	No	No
MAX205E	5	5	0	None	Yes	Yes
MAX206E	4	3	0	4 (0.1µF)	Yes	Yes
MAX207E	5	3	0	4 (0.1µF)	No	No
MAX208E	4	4	0	4 (0.1µF)	No	No
MAX211E	4	5	0	4 (0.1µF)	Yes	Yes
MAX213E	4	5	2	4 (0.1µF)	Yes	Yes
MAX232E	2	2	0	4 (1µF)	No	No
MAX241E	4	5	0	4 (1µF)	Yes	Yes

LapLink is a registered trademark of Traveling Software, Inc.

For free samples & the latest literature: http://www.maxim-ic.com, or phone 1-800-998-8800

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อ-สกุล	นายฉลองศักดิ์ คล่องการ
วัน เดือน ปีเกิด	13 ตุลาคม พ.ศ. 2524
ภูมิลำเนา	2/41 หมู่ 1 ตำบลทับปุด อำเภอทับปุด จังหวัดพังงา 82180
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนวัดนิโครธาราม
มัธยมศึกษาตอนต้น	โรงเรียนทับปุดวิทยา
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคพังงา
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคหาดใหญ่
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สจล.
ผลงานที่ได้รับรางวัล	รองอันดับ 1 การแข่งขันทำ CD Multimedia ครบรอบ 60ปี กรมอาชีวะศึกษา
คติพจน์	อย่ากังวลกับอดีตทำวันนี้ให้เต็มที่แล้วมันจะดีเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อ-สกุล นางสาวสุจิตรา โสทอง

วัน เดือน ปีเกิด 17 เมษายน 2524

ภูมิลำเนา 239 หมู่ 7 ตำบลบ้านหว้า อำเภอเมือง
จังหวัดขอนแก่น 40000

ประวัติการศึกษา

ประถมศึกษา โรงเรียนบ้านหว้าเหล่าโพนทองประชานุกูล

มัธยมศึกษาตอนต้น โรงเรียนบ้านหว้าเหล่าโพนทองประชานุกูล

ประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคขอนแก่น

ประกาศนียบัตรวิชาชีพชั้นสูง สถาบันเทคโนโลยีราชมงคลวิทยาเขตขอนแก่น

ปริญญาตรี สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์
ภาควิชาครุศาสตร์วิศวกรรม
คณะครุศาสตร์อุตสาหกรรม สจล.

คติพจน์ อย่าปล่อยเวลาให้เสียเปล่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้