# การควบคุมการรับส่งข้อมูลผ่านพอร์ตยูเอสบี

# DATA TRANSFER VIA USB PORT


โดย

นางสาว พัชราภรณ์ ชัยวณิชย์

นาย พิพัฒน์ ตั้งตรงสุนทร

นาย พิพัฒน์ นิเวศน์มรินทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

# การควบคุมการรับส่งข้อมูลผ่านพอร์ตยูเอสบี

# DATA TRANSFER VIA USB PORT


โดย

| | | | | |
|---|---|---|---|---|
| นางสาว | พัชราภรณ์ | ชัยวณิชย์ | รหัสนักศึกษา | 42010223 |
| นาย | พิพัฒน์ | ตั้งตรงสุนทร | รหัสนักศึกษา | 42010232 |
| นาย | พิพัฒน์ | นิเวศน์มรินทร์ | รหัสนักศึกษา | 42010233 |

อาจารย์ที่ปรึกษา

ผศ. พลผดุง ผดุงกุล


ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ ปีการศึกษา 2545

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมการรับส่งข้อมูลผ่านพอร์ตยูเอสบี


ผู้จัดทำ

| | | | |
|---|---|---|---|
| 1.นางสาว พัชราภรณ์ | ชัยวณิชย์ | รหัสประจำตัว | 42010223 |
| 2. นาย พิพัฒน์ | ตั้งตรงสุนทร | รหัสประจำตัว | 42010232 |
| 3. นาย พิพัฒน์ | นิเวศน์มรินทร์ | รหัสประจำตัว | 42010233 |


.....อาจารย์ที่ปรึกษา

( ผศ.พลผดุง ผดุงกุล )

โครงงานเรื่อง ( ภาษาไทย ) **การควบคุมการรับส่งข้อมูลผ่านพอร์ตยูเอสบี**

(ภาษาอังกฤษ) **DATA TRANSFER VIA USB PORT**

จัดทำโดย


| | | | | |
|---|---|---|---|---|
| นางสาว | พัชราภรณ์ | ชัยวณิชย์ | รหัสนักศึกษา | 42010223 |
| นาย | พิพัฒน์ | ตั้งตรงสุนทร | รหัสนักศึกษา | 42010232 |
| นาย | พิพัฒน | นิเวศน์มรินทร์ | รหัสนักศึกษา | 42010233 |


โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

( ผศ.พลผดุง ผดุงกุล )

อาจารย์ที่ปรึกษา

# การควบคุมการรับส่งข้อมูลผ่านพอร์ต USB

นางสาว พัชราภรณ์ ชัยวณิชย์
นาย พิพัฒน์ ตั้งตรงสุนทร
นาย พิพัฒน์ นิเวศน์มรินทร์
ผศ. พลพดุง พดุงกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

## บทคัดย่อ

โครงงานนี้นำเสนอการควบคุมการรับส่งข้อมูลผ่านพอร์ต USB โดยใช้ IC PDIUSBD11 Rev.1.1 ส่งผ่านข้อมูลด้วยความเร็วสูงแบบ $I^2C$ ซึ่งทำการควบคุมผ่าน MCS-51 มีการติดต่อรับส่งข้อมูลระหว่าง IC PDIUSBD11 และ IC ชนิดหน่วยความจำแบบ EEPROM 24LC256 เป็น DATA LOGGER ทำหน้าที่เก็บข้อมูลของอุณหภูมิ โดย IC DS1620 ตรวจจับค่าอุณหภูมิตัวอย่าง จากนั้นจะถูกนำมาเก็บไว้ใน EEPROM 24LC256 โดยการส่งข้อมูลผ่านทาง $I^2C$ ข้อมูลที่วัดได้จะถูกแสดงผ่านทางหน้าจอคอมพิวเตอร์

# DATA TRANSFER VIA USB PORT

Miss. Patcharaporn Chaivanich

Mr. Pipat Tangtongsuntorn

Mr. Pipat Nivatemarin

Asst. Prof. Polpadung Padungkul Advisor

Year 2002

Abstract

Universal serial Bus (USB) is a powerful standard for connecting all kinds of peripherals to PC's . This Project presents controlled data transfer via USB port for using PDIUSBD11 Rev.1.1 high speed $I^2C$ interface . This device is controlled by MCS-51 transmitted data between IC PDIUSBD11 and EEPROM memory , 24LC256 by $I^2C$ . IC DS1620 measures the temperature and converted data to store in 24LC256. Data detected show on the programming interfacing.

# กิตติกรรมประกาศ

โครงงานนี้สำเร็จได้ด้วยความช่วยเหลือและคำแนะนำต่างๆมากมาย ขอขอบพระคุณ อาจารย์ พลผดุง ผดุงกุล สำหรับคำแนะนำความรู้และช่วยหาหนังสือและอุปกรณ์มาให้ ขอขอบคุณ เข้ม และ ต้อง สำหรับการหาข้อมูลต่างๆและช่วยในการแก้ไข

.....

(นางสาว พัชราภรณ์ ชัยวณิชย์)

.....

(นาย พิพัฒน์ ตั้งตรงสุนทร)

.....

(นาย พิพัฒน์ นิเวศน์มรินทร์)

ผู้จัดทำ

# สารบัญ

# สารบัญรูป

# สารบัญตาราง

# บทที่ 1

# บทนำ

**USB (Universal Serial Bus)** คือ ระบบบัสอนุกรมที่ถูกออกแบบมาเพื่อการเชื่อมต่อและใช้งานอุปกรณ์ต่อพ่วงทั้งหลายที่มีแอดเดรสต่างๆกัน โดย USB มีความซับซ้อนของระบบการทำงานมาก แต่ก็มีการทำงานที่เร็วมาก และ มีการออกแบบให้ใช้ง่ายตรงกับแนวความคิดแบบ Plug & play เพื่อลดปัญหาเรื่อง อินเตอร์รัพต์ไม่ถูกต้อง และแอดเดรสผิดพลาด หรือ ไดรเวอร์ไม่มี รวมทั้งยังสามารถรองรับอุปกรณ์ต่อพ่วงได้ถึง 127 อุปกรณ์

ในโครงงานการควบคุมการรับส่งข้อมูลผ่านพอร์ต USB จะทำการควบคุมการทำงานของดาต้าลอคเกอร์ (data logger) ที่มีอุปกรณ์ส่งผ่านข้อมูลแบบ $I^2C$ และแสดงผลทางหน้าจอคอมพิวเตอร์

**ภาคส่งสัญญาณข้อมูล**

- วงจรแปลงสัญญาณ PDIUSBD11 แปลงสัญญาณ USB เป็น $I^2C$ เพื่อส่งข้อมูลไปยัง MCS51

**ภาคการเก็บข้อมูลและประมวลผล**

- MCS51 เบอร์ 89C51
- ไอซีวัดอุณหภูมิ DS1620 ส่งผ่านข้อมูลแบบ $I^2C$
- หน่วยความจำแบบ EEPROM ที่มีการส่งผ่านข้อมูลแบบ $I^2C$

**ภาคติดต่อกับผู้ใช้**

ในการติดต่อกับผู้ใช้ จะใช้โปรแกรม Visual Basic 6.0 (VB) ในการเขียนโปรแกรมติดต่อกับผู้ควบคุมและออกคำสั่งที่ได้จากการสั่งงานของผู้ควบคุมจะถูกส่งผ่าน USB port ไปยังวงจรแปลงสัญญาณ USB เป็น สัญญาณแบบ $I^2C$ และส่งต่อไปยัง MCS-51 เพื่อใช้ควบคุมการส่งอีกทีหนึ่ง

# บทที่ 2

# USB (universal serial bus)

**USB (universal serial bus)** คือ ระบบบัสอนุกรมที่ถูกออกแบบมาเพื่อการเชื่อมต่อและใช้งานอุปกรณ์ต่อพ่วงทั้งหลายที่มีแอดเดรสต่างๆกัน โดย USB มีความซับซ้อนของระบบการทำงานมาก แต่จะมีการทำงานที่เร็วมาก

## 2.1 ลักษณะทางกายภาพของ USB

### 2.1.1 คุณสมบัติของ USB

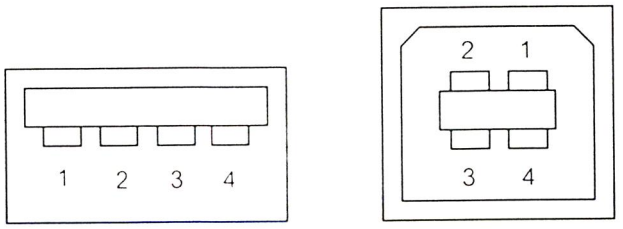
- ใช้คอนเน็กเตอร์เพียงชนิดเดียว(มีสองรูปแบบ)ในการเชื่อมต่ออุปกรณ์ทุกชนิดเข้ากับคอมพิวเตอร์
- ความสามารถในการเชื่อมต่ออุปกรณ์หลายๆชนิดรวมเข้าสู่บัสสัญญาณข้อมูลเดียวกัน
- ไม่เกิดการขัดแย้งกันสำหรับการเข้าใช้ทรัพยากรของระบบ
- ตรวจสอบการเชื่อมต่อและการตั้งค่าการทำงานต่างๆโดยอัตโนมัติระหว่างที่คอมพิวเตอร์ทำงานอยู่(Hot Attachment)
- ความเร็วในการส่งข้อมูลที่สูงถึง 12 เมกะบิตต่อวินาทีสำหรับอุปกรณ์ความเร็วสูง และ 1.5 เมกะบิตต่อวินาทีสำหรับอุปกรณ์ความเร็วต่ำ
- มีไฟเลี้ยงจ่ายให้ในระบบทำให้อุปกรณ์ที่ใช้ไฟเลี้ยงไม่มากนัก ไม่จำเป็นต้องอาศัยแหล่งจ่ายไฟภายนอก
- ค่าใช้จ่ายในการพัฒนาอุปกรณ์ต่ำเมื่อเทียบกับความเร็วที่ได้

### 2.1.2 คอนเน็กเตอร์

คอนเน็กเตอร์ USB นี้มีอยู่ 2 แบบ คือ แบบA และ B ดังแสดงในรูปที่ 2.1 โดยใช้แทนกันไม่ได้แต่มีลักษณะของขาต่างๆเหมือนกันดังตารางที่ 2.1

| หมายเลขขา | ชื่อสัญญาณ | สีของสายสัญญาณ |
|---|---|---|
| 1 | Vcc +5 V | แดง |
| 2 | ข้อมูล- (D-) | ขาว |
| 3 | ข้อมูล+ (D+) | เขียว |
| 4 | GND | ดำ |

ตารางที่ 2.1 หน้าที่ของขาคอนเน็กเตอร์




รูปที่ 2.1 รูปคอนเน็กเตอร์ซีรีส์ A และ B

- คอนเน็กเตอร์ซีรีส์ A เป็นคอนเน็กเตอร์ด้านฮับที่เชื่อมต่อระหว่าง USB พอร์ตของฮับกับสายเชื่อมต่อจากอุปกรณ์ โดยตัวเมียจะติดตั้งอยู่กับฮับ และคอนเน็กเตอร์ตัวผู้จะติดอยู่กับสายที่ต่อมาจากอุปกรณ์
- คอนเน็กเตอร์ซีรีส์ B เป็นคอนเน็กเตอร์ด้านอุปกรณ์ ที่เชื่อมต่อสายเข้ากับอุปกรณ์ USB โดยตัวเมียจะติดตั้งอยู่กับอุปกรณ์USB และคอนเน็กเตอร์ตัวผู้จะติดอยู่กับสายที่ต่อมาจากฮับ

สำหรับ คอนเน็กเตอร์ทั้งสองแบบนี้จะมีขาที่ใช้ส่งไฟเลี้ยงยื่นออกมายาวกว่าขาที่ใช้รับข้อมูล เพื่อที่จะทำให้อุปกรณ์ได้รับไฟเลี้ยง เข้าไปก่อนที่จะได้รับสัญญาณข้อมูลเมื่อมีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้าในระบบ ทำให้อุปกรณ์ที่เข้าไปใหม่สามารถทำงานได้ทันทีโดยไม่ได้รับความเสียหาย

**2.2 โครงสร้างของ USB**

USB เป็นมาตรฐานการรับส่งข้อมูลที่มีรูปแบบการเชื่อมต่อในระบบบัส คืออุปกรณ์ทุกๆตัวจะต้องส่งสัญญาณรวมกันไปในสายสัญญาณเพียงคู่เดียวดังนั้นอุปกรณ์ทุกๆตัวจะต้องส่งสัญญาณข้อมูลต่อๆกันไปเพื่อไม่ให้เกิดการชนกัน และเนื่องจาก USB เป็นระบบบัสที่ใช้สายส่งสัญญาณเพียงคู่เดียว และสัญญาณเป็นแบบผลต่างทำให้ในช่วงเวลาหนึ่งๆจะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียวเท่านั้น ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันได้(Half Duplex)

จังหวะการรับส่งข้อมูลของระบบทั้งหมดถูกควบคุมโดยโฮสต์ (Host) ซึ่งก็คือเครื่องคอมพิวเตอร์ที่เป็นจุดรวมของอุปกรณ์ทุกตัวที่เชื่อมต่ออยู่นั่นเอง การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรมโดยทุกๆ 1 มิลลิวินาที จะเกิดการรับส่งข้อมูลขึ้น 1 เฟรม แต่เนื่องจากเฟรมข้อมูลจะต้องรับส่งภายใน 1 มิลลิวินาที นั่นหมายความว่าข้อมูลของอุปกรณ์ทุกตัวที่ต่อเชื่อมกับบัสต้องถูกกำหนดขนาดไม่ให้ใหญ่เกินกว่า ที่จะส่งภายใน 1 มิลลิวินาที จึงจำเป็นต้องอาศัย ซอฟต์แวร์มาคอยจัดการ

และฮาร์ดแวร์ที่จะคอยกระจายการส่งและรวบรวมข้อมูลซึ่ง ซอฟต์แวร์และฮาร์ดแวร์ที่จำเป็นสำหรับระบบมีดังนี้

**2.3 ส่วนซอฟต์แวร์**

**2.3.1 ไดรเวอร์อุปกรณ์ USB**

ไดรเวอร์อุปกรณ์ เป็นโปรแกรมส่วนที่เก็บข้อมูลที่จำเป็นในการติดต่อไปยังตัวอุปกรณ์แต่ละตัว เมื่อมีความต้องการจะติดต่อกับอุปกรณ์ต่างๆจะต้องแจ้งมายังไดรเวอร์อุปกรณ์ เนื่องจากตัวไดรเวอร์นี้จะรู้ว่าถ้าต้องการติดต่อกับอุปกรณ์จะต้องติดต่อผ่านเอนด์พอยต์(End Point)ไหน (อุปกรณ์แต่ละตัวจะมีชนิดและจำนวนเอนด์พอยต์ต่างกัน) ดังนั้นอุปกรณ์แต่ละตัวจะต้องมีไดรเวอร์อุปกรณ์เป็นของตัวเอง

**2.3.2 ไดรเวอร์ USB**

จากที่กล่าวมาแล้วว่าการส่งข้อมูลต้องมีการแบ่งส่วนเวลากันไปอย่างพอเหมาะในเวลา 1 เฟรมเพื่อให้อุปกรณ์ทุกตัวสามารถทำงานไปได้พร้อมๆกันและซอฟต์แวร์ที่ทำหน้าที่นี้คือ ไดรเวอร์ USB นั่นเอง

ไดรเวอร์อุปกรณ์ USB ของอุปกรณ์แต่ละตัวจะส่งการร้องขอเพื่อการติดต่อมายังไดรเวอร์ USB และเมื่อไดรเวอร์ USB รับทราบมันจะพิจารณาว่าในรอบการรับส่งข้อมูลหนึ่งๆนั้น อุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้มากแค่ไหน หากปริมาณข้อมูลที่ต้องการรับส่งมีขนาดใหญ่มากจะต้องถูกแบ่งออกเป็นส่วนย่อยๆแล้วเก็บไว้เพื่อรอส่งในรอบถัดไป

**2.3.3 ไดรเวอร์โฮสคอนโทรลเลอร์**

หลังจากไดรเวอร์ USB พิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้างมันก็จะส่งข้อมูลของอุปกรณ์แต่ละตัวที่จะติดต่อในรอบการติดต่อนั้นๆลงมายังไดรเวอร์โฮสคอนโทรลเลอร์ ตัวไดรเวอร์โฮสคอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลของอุปกรณ์แต่ละชนิดลงเป็นเฟรมข้อมูลและเพิ่มเติมส่วนประกอบต่างๆให้ครบถูกต้องตามมาตรฐานการรับส่งข้อมูลแบบ USB แล้วส่งข้อมูลทั้งหมดนี้ไปยัง ฮาร์ดแวร์ USB โฮสคอนโทรลเลอร์ เพื่อส่งข้อมูลทั้งหมดออกไปยังอุปกรณ์ต่างๆ

**2.4 ส่วนฮาร์ดแวร์**

**2.4.1 USB โฮสคอนโทรลเลอร์/ USB รูตฮับ**

USB โฮสคอนโทรลเลอร์มีหน้าที่สร้างสัญญาณจริงๆ ขึ้นมาแล้วส่งต่อไปยังรูตฮับ เพื่อกระจายออกไปยังอุปกรณ์ต่างๆตามรูปแบบของ USB นอกเหนือจากการส่งสัญญาณออกไปภายนอกแล้วรูตฮับยังมีหน้าที่สำคัญอีก 4 อย่างคือ

- ควบคุมการใช้พลังงานของอุปกรณ์ที่มาต่อ
- ตรวจสอบการเชื่อมต่อของอุปกรณ์ว่าต่ออยู่หรือไม่
- เปิดการใช้งานพอร์ตเมื่อมีอุปกรณ์ต่ออยู่ และปิดการใช้งานพอร์ตเมื่อปลดปกรณ์ออก
- รายงานสถานะของแต่ละพอร์ต เมื่อไดรเวอร์โฮสคอนโทรลเลอร์ร้องขอมา

**2.4.2 USB ฮับ**

หน้าที่หลักของ USB ฮับคือการขยายการเชื่อมต่อให้อุปกรณ์จำนวนมากๆสามารถเชื่อมต่อเข้ากับระบบบัสได้ โดยการทำงานหลักของ USB ฮับ จะมีอยู่ 2 ส่วนคือ

- ทำหน้าที่เป็นตัวทวนสัญญาณ
- ตัวจัดการพลังงาน

ในส่วนของการทวนสัญญาณตัวฮับจะต้องรับสัญญาณจากโฮสมาแล้วส่งกระจายออกไปยังพอร์ตทุกๆพอร์ต และรับสัญญาณจากแต่ละพอร์ตแล้วนำมารวมกันแล้วส่งไปที่โฮส ในส่วนของการจัดการพลังงานนั้นจะทำหน้าที่เหมือนกับรูตฮับ

**2.4.3 อุปกรณ์ USB**

อุปกรณ์ USB เป็นฮาร์ดแวร์ส่วนสุดท้ายโดยสามารถแบ่งได้เป็น 2 ประเภทคือ

- อุปกรณ์ความเร็วต่ำ(Low Speed Device) รับส่งข้อมูลด้วยความเร็ว 1.5 เมกะบิตต่อวินาที
- อุปกรณ์ความเร็วสูง(High Speed Device) รับส่งข้อมูลด้วยความเร็ว 12 เมกะบิตต่อวินาที

**2.5 การติดต่อระหว่างอุปกรณ์กับโฮส**

อุปกรณ์แต่ละตัวมีคุณสมบัติและการทำงานที่แตกต่างกันโฮสจำเป็นต้องรู้คุณสมบัติทั้งหมดของอุปกรณ์แต่ละตัวเพื่อให้การสั่งงานเป็นไปอย่างถูกต้อง และเนื่องจากบัสข้อมูลทั้งหมดจะถูกใช้งานในการรับส่งข้อมูลร่วมกันระหว่างอุปกรณ์ทุกตัว สิ่งที่โฮสจำเป็นที่จะต้องรู้อีกอย่างก็คือ ปริมาณข้อมูลที่ต้องการส่งของอุปกรณ์แต่ละตัวในบัส ข้อมูลต่างๆเหล่านี้รวมเรียกว่าดิซคริปเตอร์ของอุปกรณ์(Device Descripter)

**ดิซคริปเตอร์ของอุปกรณ์(Device Descripter)**

อุปกรณ์แต่ละตัวจะบอกรายละเอียดของตัวเองให้โฮสต์รู้ผ่านดิซคริปเตอร์ชนิดต่างๆ ซึ่งถูกแบ่งแยกตามชนิดของข้อมูลที่จะแจ้งไปยังโฮสต์โดยแต่ละชนิดจะมีเอ็นด์พอยต์ต่างกัน โดยมีดิซคริปเตอร์หลักๆในการใช้งานอยู่ 4 ชนิดคือ

ดิไวซ์ดิซคริปเตอร์(Device Descripter) สำหรับข้อมูลทั่วไปของตัวอุปกรณ์

- คอนฟิกิวเรชันดิซคริปเตอร์(Configuration Descripter) สำหรับข้อมูลที่เกี่ยวกับการทำงานในแต่ละโหมดของตัวอุปกรณ์
- อินเตอร์เฟซดิซคริปเตอร์(Interface Descripter) สำหรับข้อมูลการทำงานแต่ละหน้าที่ของตัวอุปกรณ์
- เอ็นด์พอยต์ดิซคริปเตอร์(End Point Descripter) สำหรับข้อมูลการทำงานของแต่ละเอ็นพอยต์

**2.5.1 ดิไวซ์ดิซคริปเตอร์ (Device Descripter)**

ในอุปกรณ์แต่ละตัวจะมีดิไวซ์ดิซคริปเตอร์เพียงชุดเดียวเท่านั้น ภายในจะระบุข้อมูลที่ใช้ในการเชื่อมต่อขั้นแรกเพื่อใช้ในการตั้งค่าต่างๆของอุปกรณ์และโฮสต์ให้ทำงานเข้ากันได้ นอกจากนั้นยังเก็บข้อมูลของคอนฟิกิวเรชันโหมดต่างๆของอุปกรณ์ด้วย เพราะเมื่อครั้งแรกที่อุปกรณ์เชื่อมเข้ากับบัสนั้น โฮสต์ไม่มีทางรู้ได้เลยว่าต้องติดต่อกับอุปกรณ์นี้ที่เอ็นด์พอยต์ใดจึงจำเป็นต้องเรียกข้อมูลส่วนนี้จากอุปกรณ์ก่อน

**2.5.2 คอนฟิกิวเรชันดิซคริปเตอร์ (Configuration Descripter)**

ทำหน้าที่เก็บข้อมูลที่จำเป็นสำหรับการทำงานในแต่ละคอนฟิกิวเรชันโหมด และเก็บจำนวนอินเตอร์เฟซที่ใช้งานในคอนฟิกิวเรชันนั้นๆ

**2.5.3 อินเตอร์เฟซดิซคริปเตอร์ (Interface Descripter)**

ในแต่ละคอนฟิกิวเรชันอาจมีอินเตอร์เฟซได้มากกว่า 1 อินเตอร์เฟซ เพื่อใช้งานในหน้าที่ต่างๆ ภายในอินเตอร์เฟซดิซคริปเตอร์จะบรรจุข้อมูลการใช้งานอินเตอร์เฟซนั้นๆข้อมูลเหล่านี้จะระบุว่าอุปกรณ์ถูกจัดอยู่ในคลาส(Class)ใดและมีเอ็นด์พอยต์จำนวนเท่าใดที่ใช้งานในอินเตอร์เฟสนี้

**2.5.4 เอ็นพอยต์ดิซคริปเตอร์ (End Point Descripter)**

เป็นส่วนที่เก็บข้อมูลคุณสมบัติของแต่ละเอ็นด์พอยต์เช่น ใช้การส่งสัญญาณชนิดใด สามารถรับส่งข้อมูลได้มากที่สุดครั้งละเท่าใด

**2.5.5 สตริงดิซคริปเตอร์ (Sting Descripter)**

ดิซคริปเตอร์ชนิดนี้จะเก็บข้อมูลตัวอักษรที่สามารถอ่านเข้าใจได้ไว้อธิบายส่วนต่างๆ ของดิซคริปเตอร์ทั้ง 4 ชนิดข้างต้น เช่น เก็บชื่อบริษัทผู้ผลิต,ซีเรียลนัมเบอร์ของอุปกรณ์เป็นต้น

**2.5.6 คลาสสเป็กซิฟิกดิซคริปเตอร์(Class-Specific Descripter)**

เป็นดิซคริปเตอร์พิเศษที่มีเฉพาะในอุปกรณ์บางตัวที่จัดอยู่ในบางคลาสเท่านั้น ภายในเก็บข้อมูลเฉพาะของคลาสนั้นๆ ที่อยู่นอกเหนือจากดิซคริปเตอร์พื้นฐาน 4 ชนิดแรก

| ข้อมูลไบต์ที่ | ฟิลด์ | ขนาด (ไบต์) | ชนิดข้อมูล | คำอธิบาย |
|---|---|---|---|---|
| 0 | length | 1 | จำนวน | ความยาวของดิไวซ์ดิซคริปเตอร์ |
| 1 | Descriptor Type | 1 | ค่าคงที่ | ค่าระบุชนิดของดิซคริปเตอร์ Device Descriptor = 01h |
| 2 | USB | 2 | BCD | เวอร์ชันของ USB ที่อุปกรณ์นี้ใช้งาน เก็บในรูปของ BCD |
| 4 | Device Class | 1 | Class | บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสใด |
| 5 | Device Subclass | 1 | Sub Class | บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ซับคลาสใด |
| 6 | Device Protocol | 1 | Protocol | บอกโปรโตคอลโค้ดของอุปกรณ์ว่าใช้โปรโตคอลลักษณะใด |
| 7 | MaxPacket Size0 | 1 | จำนวน | บอกขนาดข้อมูลที่สามารถส่งได้ต่อแพ็กเก็ตของเอ็นพอยต์ 0 |
| 8 | Vendor | 2 | ID | ค่า ID ของ Vendor ที่ทาง USB กำหนดไว้ |
| 10 | Product | 2 | ID | ชนิดของอุปกรณ์ที่ผลิต กำหนดโดยตัวผู้ผลิตเอง |
| 12 | Device | 2 | BCD | เวอร์ชันของอุปกรณ์ เก็บในรูปแบบ BCD |
| 14 | Manufacture | 1 | Index | ค่าอินเด็กซ์ของสตริงดิซคริปเตอร์ที่แสดงข้อมูลผู้ผลิต |
| 15 | Product | 1 | Index | ค่าอินเด็กซ์ของสตริงดิซคริปเตอร์ที่แสดงข้อมูลชนิดของอุปกรณ์ |
| 16 | Serial Number | 1 | Index | ค่าอินเด็กซ์ของสตริงดิซคริปเตอร์ที่แสดงข้อมูลหมายเลขซีเรียลของอุปกรณ์ |
| 17 | Number Configuration | 1 | จำนวน | บอกจำนวนคอนฟิกิวเรชันทั้งหมดที่อุปกรณ์รองรับได้ |

ตารางที่ 2.2 รายละเอียดของข้อมูลแต่ละไบต์ของดิไวซ์ดิซคริปเตอร์

| ข้อมูลไบต์ที่ | ฟิลด์ | ขนาด (ไบต์) | ชนิดข้อมูล | คำอธิบาย |
|---|---|---|---|---|
| 0 | length | 1 | จำนวน | ความยาวของคอนฟิกิวเรชันดิซคริปเตอร์ |
| 1 | Descriptor Type | 1 | ค่าคงที่ | ค่าระบุชนิดของดิซคริปเตอร์<br>Configuration Descriptor = 02h |
| 2 | Total Length | 2 | จำนวน | ขนาดข้อมูลทั้งหมดที่ได้จากการอ่านดิซคริปเตอร์ |
| 4 | NumInterfaces | 1 | จำนวน | บอกจำนวนอินเตอร์เฟสทั้งหมดที่อุปกรณ์รองรับได้ |
| 5 | Configuration Value | 1 | จำนวน | ค่าหมายเลขคอนฟิกิวที่ใช้เลือกคอนฟิกิวเรชันนี้ |
| 6 | Configuration | 1 | Index | ค่าอินเด็กซ์ของสตริงดิซคริปเตอร์ที่อธิบายข้อมูลของคอนฟิกิวเรชันนี้ |
| 7 | Attributes | 1 | กำหนดเป็นบิต | แต่ละบิตบอกคุณสมบัติของคอนฟิกิวเรชันดังนี้<br>บิต D7 ใช้พลังงานจากบัส<br>บิต D6 ใช้พลังงานของตัวเอง<br>บิต D5 การรองรับ Remote Wakeup<br>บิต D4-D0 ไม่ใช้งาน<br>บิตใดเป็น 1 หมายถึงมีคุณสมบตินั้น |
| 8 | MaxPower | 1 | มิลลิแอมป์ (mA) | บอกค่ากระแสที่ใช้ แต่ละหน่วยมีค่า 2 มิลลิแอมป์ |

ตารางที่ 2.3 รายละเอียดของข้อมูลแต่ละไบต์ของคอนฟิกิวเรชันดิซคริปเตอร์

| ข้อมูลไบต์ที่ | ฟิลด์ | ขนาด (ไบต์) | ชนิดข้อมูล | คำอธิบาย |
|---|---|---|---|---|
| 0 | length | 1 | จำนวน | ความยาวของอินเตอร์เฟซดิซคริปเตอร์ |
| 1 | Descriptor Type | 1 | ค่าคงที่ | ค่าระบุชนิดของดิซคริปเตอร์ Device Descriptor = 01h |
| 2 | Interface Number | 1 | จำนวน | ค่าหมายเลขอินเตอร์เฟซที่ใช้เลือกอินเตอร์เฟซนี้ |
| 3 | Alternate Setting | 1 | จำนวน | ค่าหมายเลข Alternate Setting ที่ใช้เลือกในแต่ละอินเตอร์เฟซ |
| 4 | NumEndpoint | 1 | จำนวน | บอกจำนวนเอ็นด์พอยต์ที่ใช้งานในอินเตอร์เฟซนี้ |
| 5 | Interface Class | 1 | Class | บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในคลาสใด ในกรณีที่แต่ละอินเตอร์เฟซมีอยู่ในคลาสที่ต่างกัน |
| 6 | Interface SubClass | 1 | Sub Class | บอกการจัดหมวดหมู่ของอุปกรณ์ว่าอยู่ในซับคลาสใด ในกรณีที่แต่ละอินเตอร์เฟซมีอยู่ในซับคลาสที่ต่างกัน |
| 7 | Interface Protocol | 1 | Protocol | บอกโปรโตคอลโค้ดของอุปกรณ์ว่าใช้โปรโตคอลลักษณะใด ในกรณีที่แต่ละอินเอร์เฟซใช้โปรโตคอลที่ต่างกัน |
| 8 | Interface | 1 | Index | ค่าอินเด็กซ์ของสตริงดิซคริปเตอร์ที่อธิบายข้อมูลของอินเตอร์เฟซนี้ |

ตารางที่ 2.4 รายละเอียดของข้อมูลแต่ละไบต์ของอินเตอร์เฟซดิซคริปเตอร์

| ข้อมูลไบต์ที่ | ฟิลด์ | ขนาด (ไบต์) | ชนิดข้อมูล | คำอธิบาย |
|---|---|---|---|---|
| 0 | Length | 1 | จำนวน | ขนาดข้อมูลเฉพาะของเอ็นด์พอยต์ดิซคริปเตอร์ |
| 1 | Descriptor Type | 1 | ค่าคงที่ | ค่าระบุชนิดของดิซคริปเตอร์<br>Endpoint Descriptor = 05H |
| 2 | Endpoint Address | 1 | Endpoint | บอกค่าแอดเดรสของเอนต์พอยด์และทิศทางการส่งข้อมูล<br>บิต D0-D3 บอกหมายเลขแอดเดรสของเอ็นด์พอยต์<br>บิต D4-D6 ไม่ใช้งาน<br>บิต D7 บอกทิศทางการส่งข้อมูลของเอ็นด์พอยต์ |
| 3 | Attributes | 1 | กำหนดเป็นบิต | บอกชนิดการรับส่งข้อมูลที่เอ็นด์พอยต์ใช้ดังนี้<br>D1=0 D0=0 Control Transfer<br>D1=0 D0=1 Isochronous Transfer<br>D1=0 D0=0 Bulk Transfer<br>D1=0 D0=0 Interrupt Transfer<br>บิตที่เหลือไม่ใช้งาน |
| 4 | MaxPacket Size | 2 | จำนวน | ค่าขนาดของข้อมูลที่ใหญ่ที่สุดที่เอ็นด์พอยต์นี้ที่สามารถรองรับได้ |
| 6 | Interval | 1 | จำนวน | ค่าคาบเวลาในการโพลอ่านข้อมูลจากโฮสต์<br>สำหรับ Blukและ Control endpoint จะไม่สนใจบิตนี้ |

ตารางที่ 2.5 รายละเอียดของข้อมูลแต่ละไบต์ของเอ็นด์พอยต์ดิซคริปเตอร์

# บทที่ 3
# โครงสร้างพื้นฐานในการรับส่งข้อมูล ของ USB

## 3.1 ภาพรวมของการรับส่งข้อมูล ของ USB

จากที่ทราบกันมาแล้วว่าการเชื่อมต่อของ USB อยู่ในรูปแบบของบัส ดังนั้นการส่งข้อมูลของอุปกรณ์หลายๆชนิดรวมกันไปในสายส่งเส้นเดียวนั้นจะต้องมีการแบ่งช่วงข้อมูลแต่ละส่วนออกจากกันอย่างชัดเจน ข้อมูลที่ถูกแบ่งออกเป็นส่วนๆนี้เรียกว่า แพ็กเก็ต(Packet) โดยแต่ละแพ็กเก็ตนอกจากจะประกอบไปด้วยข้อมูลที่ต้องการจะรับหรือส่งจริงๆแล้วยังต้องประกอบไปด้วยข้อมูลส่วนอื่นๆรวมไปด้วย เพื่อควบคุมการส่งข้อมูลให้เกิดความถูกต้อง

เนื่องจากข้อมูลที่ถูกส่งไปในสาย USB นั้นมีอยู่หลายชนิด ดังนั้นแพ็กเก็ตข้อมูลจึงถูกแบ่งออกเป็นชนิดต่างๆตามจุดประสงค์ในการส่งและชนิดของข้อมูลที่บรรจุอยู่ภายใน โดยแพ็กเก็ตข้อมูลแบ่งออกเป็น 4 ชนิดคือ

- โทเคนแพ็กเก็ต(Token Packet)
- ดาต้าแพ็กเก็ต(Data Packet)
- แฮนด์เช็กแพ็กเก็ต(Handshake Packet)
- แพ็กเก็ตพิเศษ(Special Packet)

เมื่อนำแพ็กเก็ตข้อมูลต่างชนิดกันมารวมกันเพื่อสร้างการติดต่อสื่อสารขึ้นมา เราเรียกการติดต่อนี้ว่า ทรานแซ็กชัน(Transaction) ซึ่งถูกแบ่งออกตามลักษณะการติดต่อสื่อสารเป็น 3 ชนิดตามทิศทางการส่งข้อมูลคือ

- ทรานแซ็คชันขาเข้า(In Transaction)
- ทรานแซ็คชันขาออก(Out Transaction)
- เซ็ตอัพทรานแซ็คชัน(Setup Transaction)

แต่ละทรานแซ็คชัน ส่วนใหญ่จะประกอบไปด้วย 3 เฟสคือ

- เฟสโทเคนแพ็กเก็ต(Token Packet Phase)
- เฟสดาต้าแพ็กเก็ต(Data Packet Phase)
- เฟสแฮนด์เช็กแพ็กเก็ต(Handshake Packet Phase)

เมื่อสร้างทรานแซ็คชันขึ้นมาจากแพ็กเก็ตต่างๆแล้วก็จะถูกนำไปเรียงในเฟรมข้อมูลที่มีการส่งในทุกๆ 1 มิลลิวินาที การจัดรูปแบบของทรานแซ็คชันต่างๆของแต่ละเฟรมข้อมูลจะถูกกำหนดเป็นชนิดของการส่งข้อมูลหรือเรียกว่าทรานสเฟอร์(Transfer)ตามความต้องการของอุปกรณ์เป็น 4 ชนิด คือ

1. อินเตอร์รัพต์ทรานสเฟอร์ (InterruptTransfer)
2. ไอโซโครนัสทรานสเฟอร์ (Isochronous Transfer)
3. บัลก์ทรานสเฟอร์ (Bulk Transfer)
4. คอนโทรลทรานสเฟอร์ (Control Transfer)

**3.2 โครงสร้างการรับส่งข้อมูลระดับล่าง**

เราทราบกันมาแล้วว่าโครงสร้างระดับล่างสุดของ USB คือแพ็กเก็ต เมื่อนำมารวมกันจะประกอบขึ้นเป็นทรานแซ็คชัน โดยในแต่ละทรานแซ็คชันส่วนมากจะประกอบด้วย 3 เฟสหรือแพ็กเก็ต(เฟสจะสื่อให้เห็นถึงการแบ่งช่วงข้อมูลมากกว่า)

**3.2.1เฟสโทเคน (Token Packet Phase)**

เป็นเฟสเริ่มต้นของทุกๆทรานแซ็คชันภายในบรรจุข้อมูลที่บอกชนิด ของทรานแซ็คชัน รวมไปถึง แอดเดรสของอุปกรณ์ที่จะติดต่อด้วย

**3.2.2เฟสข้อมูล (Data Packet)**

เป็นเฟสที่ทำหน้าที่ในการส่งหรือรับข้อมูลปริมาณมากๆโดยข้อมูลที่สามารถส่งหรือรับได้มากที่สุดต่อครั้งเท่ากับ 1023 ไบต์ แต่ขนาดข้อมูลมากที่สุดที่สามารถจะส่งได้จะถูกกำหนดจากชนิดของการส่งข้อมูลอีกทีหนึ่ง

**3.2.3เฟสแฮนด์เช็ก (Handshake Packet Phase)**

เนื่องจากทุกๆครั้งที่มีการส่งข้อมูล(ยกเว้นการส่งข้อมูลแบบไอโซโครนัส)จะมีการรับประกันความถูกต้องของการส่งดังนั้นเฟสนี้จึงเป็นข้อมูลที่ฝ่ายรับส่งกลับไปยังฝ่ายส่ง เพื่อตรวจสอบความถูกต้องในการสื่อสารข้อมูล

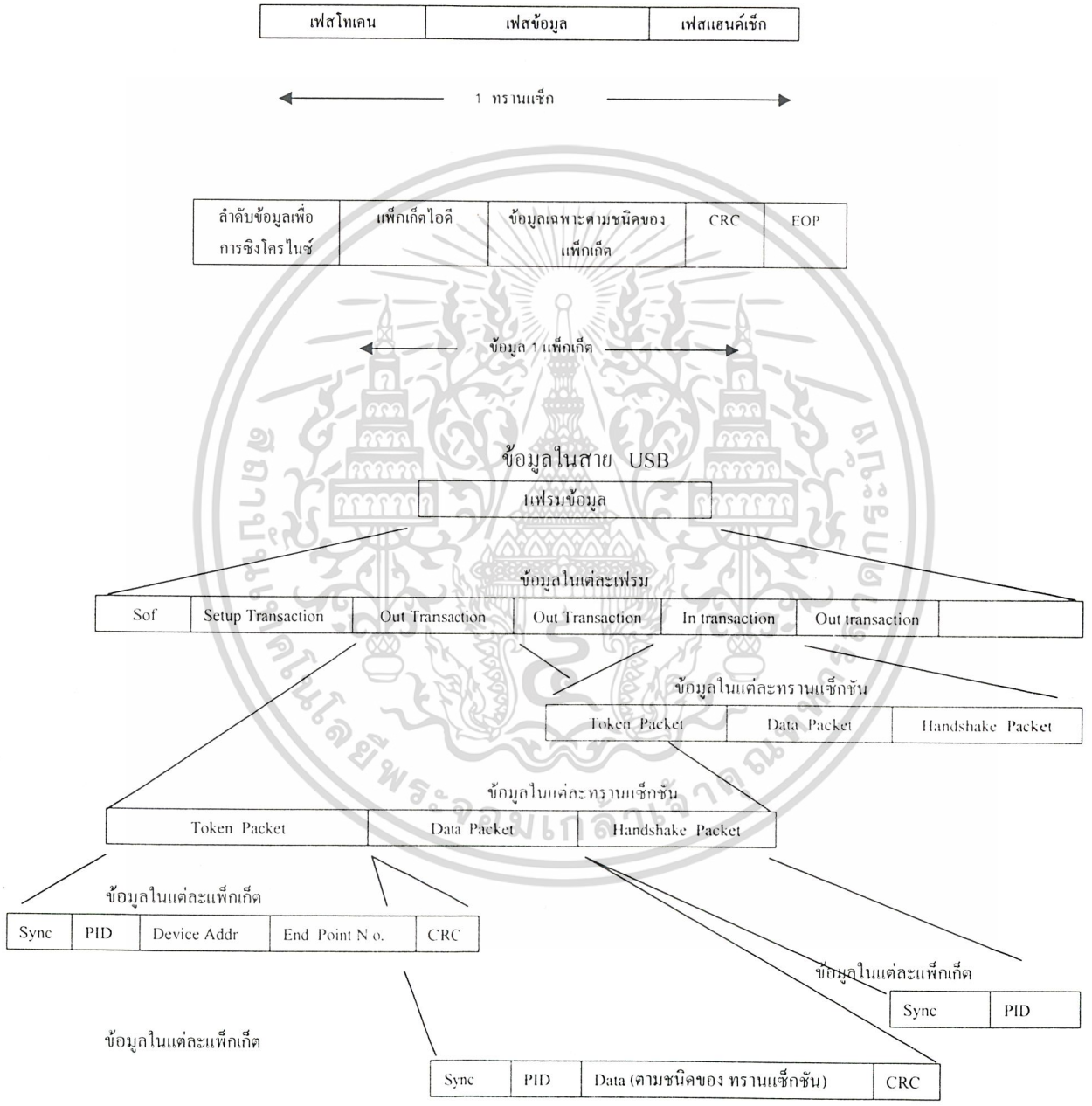
แต่ละ แพ็กเก็ตประกอบจากข้อมูลหลายๆส่วนดังรูปที่ 3.1 แต่ก่อนที่จะส่งแพ็กเก็ตข้อมูลทุกครั้งจะต้องเริ่มด้วยลำดับข้อมูลเพื่อการซิงโครไนซ์ เพื่อกระตุ้นให้อุปกรณ์ปลายทางรู้ว่าเกิดการส่งข้อมูลขึ้นแล้วและทำให้อุปกรณ์ปลายทางสามารถปรับสัญญาณนาฬิกาให้เข้ากับจังหวะสัญญาณข้อมูลที่รับได้

ต่อจากลำดับข้อมูลเพื่อการซิงโครไนซ์ จะเป็นส่วนเริ่มต้นของแพ็กเก็ตคือ แพ็กเก็ตไอดี (Packet ID:PID)ส่วนนี้จะเป็นส่วนที่บอกชนิดของแพ็กเก็ตว่าเป็นชนิดใดข้อมูลในส่วนนี้จะมีขนาด 8 บิตโดย 4 บิตแรกเป็นชนิดของแพ็กเก็ตและ 4 บิตหลังเป็นส่วนตรวจสอบซึ่งจะเป็นข้อมูลคอมพลีเมนต์(Complement)



หลังจากส่วนแพ็กเก็ตไอดีจะเป็นส่วนของข้อมูลเฉพาะตามชนิดของแพ็กเก็ตต่างๆ ซึ่งจะแตกต่างกันไปตามหน้าที่ของแต่ละแพ็กเก็ต หลังจากส่วนนี้ก็จะตามหลังด้วยส่วนตรวจสอบ

ความผิดพลาดแบบ CRC โดยจะใช้ CRC 16 บิต เมื่อแพ็กเก็ตนั้นเป็นแพ็กเก็ตข้อมูล และใช้ CRC 5 ในกรณีที่เป็นแพ็กเก็ตชนิดอื่นๆ หลังจากส่วนของ CRC ก็จะปิดท้ายด้วยสัญญาณ EOP(End of Packet)เพื่อเป็นการบอกว่าจบแพ็กเก็ต




รูปที่ 3.1 โครงสร้างการส่งข้อมูลของ USB

เมื่อนำแพ็กเก็ตทั้งหมดมาจัดเรียงจะได้ทรานแซ็คชันสำหรับส่งหรือรับข้อมูลขึ้นมาโดย ทรานแซ็คชัน มีอยู่ 3 ชนิดคือ

**3.2.4ทรานแซ็คชันขาเข้า(In Transaction)**

เริ่มต้นด้วยฝั่งโฮสต์ส่งโทเคนแพ็กเก็ต In ซึ่งภายในจะระบุแอดเดรสและเอ็นด์พอยน์ที่ต้องการรับข้อมูลเข้ามาหลังจากนั้นอุปกรณ์จะส่งดาต้าแพ็กเก็ต 0 หรือ 1 ที่บรรจุข้อมูลตามแอดเดรสและเอ็นด์พอยน์ที่ระบุกลับมายังโฮสต์เมื่อโฮสต์ได้รับข้อมูลครบและตรวจสอบแล้วว่าถูกต้องโฮสต์จะส่งแฮนด์เช็กแพ็กเก็ต ACK กลับไปยังอุปกรณ์ แต่ถ้าอุปกรณ์ไม่สามารถส่งแพ็กเก็ตข้อมูลกลับมายังโฮสต์ได้จะส่งแฮนด์เช็กแพ็กเก็ต NAK ขึ้นอยู่กับลักษณะความผิดพลาดที่เกิดขึ้น

**3.2.5ทรานแซ็คชันขาออก(Out Transaction)**

เริ่มต้นด้วยฝั่งโฮสต์ส่งโทเคนแพ็กเก็ต Out ซึ่งภายในจะระบุแอดเดรสและเอ็นด์พอยน์ปลายทางที่ต้องการจะส่งข้อมูลไปหลังจากนั้นอุปกรณ์จะส่งดาต้าแพ็กเก็ต 0 หรือ 1 ที่บรรจุข้อมูลที่ต้องการส่งตามออกไปเมื่ออุปกรณ์ได้รับข้อมูลครบและตรวจสอบแล้วว่าถูกต้อง ก็จะส่งแฮนด์เช็กแพ็กเก็ต ACK กลับมายังโฮสต์ แต่ถ้าอุปกรณ์ไม่สามารถรับแพ็กเก็ตข้อมูลกลับมายังโฮสต์ได้จะส่งแฮนด์เช็กแพ็กเก็ต NAK ขึ้นอยู่กับลักษณะความผิดพลาดที่เกิดขึ้น

**3.2.6ทรานแซ็คชันเซ็ตอัพ(Setup Transaction)**

ทรานแซ็กชันนี้ คือ สเตจที่ 1 ของคอนโทรลทรานสเฟอร์ เริ่มต้นจากฝั่งโฮสต์ส่งโทเคนแพ็กเก็ต Setup ซึ่งภายในจะระบุแอดเดรสและเอ็นด์พอยน์ที่จะคอยรับคำสั่งของอุปกรณ์ไปหลังจากนั้นก็จะตามด้วยแพ็กเก็ตข้อมูลที่บรรจุคำสั่งที่ต้องการส่งไปควบคุม เมื่ออุปกรณ์ได้รับข้อมูลแล้วก็จะส่งแฮนด์เช็คแพ็กเก็ตกลับมา

## 3.3 ชนิดของการรับส่งข้อมูล ของ USB

### 3.3.1 ไอโซโครนัสทรานสเฟอร์

การรับส่งข้อมูลชนิดนี้จะมีการส่งหรือรับข้อมูลในทุกๆเฟรมข้อมูล ทำให้เกิดการรับส่งข้อมูลด้วยความเร็วคงที่ แต่ก็หมายความว่าการรับส่งแบบนี้จะต้องจองการรับส่งข้อมูลในแต่ละเฟรมไว้ตลอดเวลาด้วย

ประเภทของการใช้งานที่ใช้รับส่งข้อมูลชนิดนี้ก็คือ งานที่ต้องการอัตราการส่งข้อมูลที่คงที่ตลอดเวลา แต่จากที่ว่าต้องรับและส่งข้อมูลในทุกๆเฟรมนั่นหมายความว่าหากเกิดความผิดพลาดของข้อมูลจะไม่มีการพยายามส่งข้อมูลที่ผิดนั้นกลับไปใหม่(ไม่มีการรับประกันความถูกต้อง)

ก่อนเริ่มการทำงานของอุปกรณ์จะต้องทำการซิงโครไนซ์การทำงานของตัวเองเข้ากับสัญญาณเริ่มต้นในแต่ละเฟรม(SOF) การซิงโครไนซ์มี 3 วิธีคือ

**3.3.1.1อะซิงโครนัส(Asynchronous)**

การซิงโครไนซ์วิธีนี้หมายความว่าสัญญาณนาฬิกาของอุปกรณ์ไม่จำเป็นต้องเข้าจังหวะกับสัญญาณข้อมูลของ USB แต่อัตราการส่งข้อมูลจะต้องถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่งตอนเริ่มต้นการเชื่อมต่อ

**3.3.1.2ซิงโครนัส(synchronous)**

การซิงโครไนซ์วิธีนี้หมายความว่าสัญญาณนาฬิกาของอุปกรณ์ต้องเข้าจังหวะกับสัญญาณ SOF ของ USB โดยก่อนเริ่มการทำงานอุปกรณ์จะต้องมีการปรับสัญญาณให้เข้ากับจังหวะของสัญญาณ SOF แต่หากอุปกรณ์ไม่สามารถปรับสัญญาณนาฬิกาของตัวเองได้ USB จะยอมให้จังหวะการส่ง SOF ของอุปกรณ์เป็นหลัก โดยอุปกรณ์นั้นจะถูกเรียกว่า อุปกรณ์มาสเตอร์(Master Device) อัตราการส่งข้อมูลจะต้องถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่งตอนเริ่มต้นการเชื่อมต่อเช่นเดียวกับแบบอะซิงโครนัส

**3.3.1.3อะแดปทีฟ(Adaptive)**

เป็นวิธีที่อุปกรณ์มีความสามารถมากที่สุดเพราะว่าอุปกรณ์จะสามารถปรับความเร็วในการส่งข้อมูลได้ในช่วงที่กว้างมาก ไม่ถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่ง สามารถเปลี่ยนแปลงความเร็วได้ในระหว่างการใช้งาน สัญญาณนาฬิกาภายในระบบของอุปกรณ์จะเข้าจังหวะกับสัญญาณข้อมูลที่ส่งในสาย

**3.3.2 อินเตอร์รัพต์ทรานสเฟอร์**

การรับส่งข้อมูลชนิดนี้สร้างขึ้นเพื่อเลียนแบบระบบการทำงานของอุปกรณ์ที่สร้างสัญญาณอินเตอร์รัพต์ให้แก่ระบบ วิธีการเลียนแบบอาศัยการโพล อ่านข้อมูลจากอุปกรณ์ต่างๆในระยะเวลาที่กำหนดอย่างสม่ำเสมอโดยอัตราการโพลข้อมูลนี้จะต้องไม่ช้าเกินไปเพราะจะทำให้เกิดการสูญเสียข้อมูลที่โพลอ่านไม่ทันได้ และก็จะต้องไม่เร็วเกินไปจนกินแบนด์วิดธ์ข้อมูลจนเกินความจำเป็น ถ้าหากอุปกรณ์ไม่มีข้อมูลที่ต้องการส่ง ก็จะส่งสัญญาณ NAK (No Acknowledge) ตอบกลับมาแต่หากมีข้อมูลที่ต้องการส่งก็จะส่งข้อมูลที่ต้องการนั้นกลับมาตามเวลาในการโพลจะขึ้นอยู่

กับความต้องการของอุปกรณ์แต่ละชนิด โฮสต์จะรู้คาบเวลาในการโพลของเอ็นต์พอยน์ที่ใช้อินเตอร์รัพต์ทรานสเฟอร์จากการอ่านเอนต์พอยน์ดิซคริปเตอร์

การส่งข้อมูลแต่ละครั้งกำหนดให้มีการส่งข้อมูลได้มากที่สุด 64 ไบต์โดยการส่งข้อมูลในแต่ละครั้งจะต้องส่งข้อมูลเต็มจำนวน(64ไบต์) แล้วการส่งครั้งสุดท้ายจะส่งไม่เต็มจำนวนสูงสุด (น้อยกว่า 64 ไบต์)ซึ่งจะเป็นตัวบอกให้โฮสต์รู้ว่าจบการส่งข้อมูลแล้ว การส่งข้อมูลแบบนี้จะมีการตรวจสอบความถูกต้องของข้อมูลและมีการส่งข้อมูลใหม่ในรอบการโพลถัดไปหากเกิดความผิดพลาด

**3.3.3 บัลก์ทรานสเฟอร์**

บัลก์ทรานสเฟอร์เป็นการรับส่งข้อมูลที่ใช้กับงานที่ไม่ต้องการอัตราเร็วในการส่งข้อมูลที่คงที่แต่ข้อมูลที่จะส่งต้องไม่เสียหาย การรับส่งข้อมูลชนิดนี้ไม่สามารถจองการใช้งานแบนด์วิดธ์ของบัสได้ ดังนั้นการรับส่งข้อมูลจึงเกิดขึ้นเมื่อบัสว่างเท่านั้น

เนื่องจากอุปกรณ์ที่ใช้ในการส่งข้อมูลชนิดนี้ส่วนใหญ่จะต้องการความถูกต้องของข้อมูลสูง ในการส่งข้อมูลแต่ละครั้งจึงมีการตรวจความถูกต้องในทุกๆครั้งและจะเกิดการส่งใหม่ถ้ามีความผิดพลาดเกิดขึ้น

**3.3.4 คอนโทรลทรานสเฟอร์**

การรับส่งข้อมูลชนิดนี้อาจจะถือได้ว่าเป็นการรับส่งข้อมูลที่สำคัญที่สุดก็ว่าได้ เพราะจะถูกใช้สำหรับการส่งคำสั่งควบคุมการทำงานของอุปกรณ์ทั้งหมด ตั้งแต่เริ่มแรกที่อุปกรณ์ถูกเชื่อมเข้ากับระบบก็จะเกิดการอ่านข้อมูลดิซคริปเตอร์ต่างๆเมื่อได้ข้อมูลมาโฮสต์ก็จะพิจารณาว่าสามารถรองรับการทำงานได้หรือไม่ ทั้งหมดนี้จะใช้คอนโทรลทรานสเฟอร์เพื่อติดต่อกับคอนโทรลเอ็นด์พอยน์ทั้งสิ้น

ทั้งหมดที่กล่าวไปเป็นโครงสร้างพื้นฐานของการส่งข้อมูลในระบบ USB แต่ในการใช้งานจริงจะมีการแบ่งแพ็กเก็ตข้อมูลเป็น 2 ชนิดคือ 0 กับ 1ที่เรียกว่า ดาต้าท็อกเกิล

**3.4 ดาต้าท็อกเกิล(Data toggle)**

สาเหตุที่ต้องมีการกำหนดแพ็กเก็ตข้อมูลออกเป็น 2 ชนิดก็เพื่อแก้ปัญหาการรับส่งข้อมูลไม่ถูกต้องในบางรูปแบบ ซึ่งเป็นช่องว่างในการตรวจสอบทั้งหมดก่อนหน้านี้เช่น

ในกรณีของทรานแซ็กชันขาเข้าเริ่มต้นจากโฮสต์ส่งแพ็กเก็ตโทเคนขาเข้าไปยังอุปกรณ์ การส่งสมบูรณ์อุปกรณ์ได้รับข้อมูลถูกต้องจึงส่งแพ็กเก็ตข้อมูลที่โฮสต์ต้องการกลับมา ซึ่งโฮสต์

ก็ได้รับอย่างถูกต้องอีกเช่นกันจึงส่งแพ็กเก็ตแฮนด์เช็กกลับไปยังอุปกรณ์ แต่โชคร้ายเกิดขึ้นขณะกำลังส่งแพ็กเก็ตแฮนด์เช็กทำให้เกิดความผิดพลาดขึ้นกับแพ็กเก็ตแฮนด์เช็ก เมื่ออุปกรณ์ได้รับข้อมูลที่ผิดก็จะไม่สนใจแพ็กเก็ตนั้นๆผลที่ตามมาก็คือ อุปกรณ์จะคิดว่าโฮสต์ยังไม่ได้รับข้อมูลที่ตัวเองส่งไป เมื่อถึงทรานแซ็กชันขาเข้าครั้งถัดไปอุปกรณ์ก็จะส่งข้อมูลชุดเดิมที่คิดว่าโฮสต์ยังไม่ได้รับในขณะที่โฮสต์เข้าใจว่าข้อมูลนี้คือข้อมูลชุดใหม่

วิธีแก้ไขข้อปัญหานี้ เรียกว่า ดาต้าท็อกเกิล(Data Toggle) ซึ่งวิธีนี้เป็นการจัดลำดับการใช้งานแพ็กเก็ตข้อมูล 0 และ 1 เพื่อให้โฮสต์หรือตัวอุปกรณ์ทราบถึงความผิดพลาดในการส่งข้อมูล โดยกระบวนการส่งข้อมูลจะเริ่มต้นที่การส่งข้อมูลในทรานแซ็กชันแรกซึ่งใช้แพ็กเก็ตข้อมูล 0 ในการส่งหรือรับข้อมูลและเมื่อมีการรับส่งข้อมูลครั้งต่อไปก็จะสลับไปใช้แพ็กเก็ตข้อมูล 1 และสลับกันไปเรื่อยๆเช่น

จากปัญหาที่แล้วเริ่มจากฮสต์ส่งแพ็กเก็ตโทเคนไปยังอุปกรณ์เรียบร้อยถูกต้อง ซึ่งอุปกรณ์ก็จะส่งแพ็กเก็ตข้อมูลกลับมายังโฮสต์ และเนื่องจากเป็นการส่งแพ็กเก็ตข้อมูลครั้งแรกจึงต้องใช้แพ็กเก็ตข้อมูล 0 เมื่อโฮสต์ได้รับข้อมูลถูกต้องก็จะส่งแพ็กเก็ตแฮนด์เช็กไปยังอุปกรณ์ และได้เกิดความผิดพลาดขึ้นกับแพ็กเก็ตแฮนด์เช็กนี้ทำให้อุปกรณ์คิดว่าโฮสต์ไม่ได้รับข้อมูล เมื่อถึงทรานแซ็กชันขาเข้าครั้งถัดไปอุปกรณ์ก็จะส่งข้อมูลชุดเดิมที่คิดว่าโฮสต์ยังไม่ได้รับกลับมาอีกครั้งโดยใช้แพ็กเก็ตข้อมูล 0 เหมือนเดิมเมื่อโฮสต์ได้รับแพ็กเก็ตข้อมูล 0 ซ้ำอีกครั้งก็จะรู้ได้ว่านี่เป็นข้อมูลชุดเดิมไม่ต้องเอาไปใช้งาน

### 3.4.1 เทคนิคการเข้ารหัสสัญญาณ

การเข้ารหัสสัญญาณข้อมูลก่อนจะส่งออกไปนั้นใช้เทคนิค 2 ส่วนประกอบกันคือ NRZI และการเติมสตัฟบิต

#### 3.4.1.1)การเข้ารหัส NRZI

ก่อนจะส่งข้อมูลไปในสายส่งนั้น ข้อมูลจะต้องถูกเข้ารหัสแบบ NRZI (Non Return to Zero,Inverted)ก่อน ประโยชน์ของการเข้ารหัสข้อมูลแบบนี้ก็คือการแฝงสัญญาณนาฬิกาส่งไปกับตัวข้อมูลด้วย ทำให้อุปกรณ์ปลายทางสามารถซิโครไนซ์ความถี่ของตัวเองให้ตรงกับความถี่ของฝั่งโฮสต์ได้ ซึ่งทำให้สามารถอ่านข้อมูลที่ส่งมาได้อย่างถูกต้อง

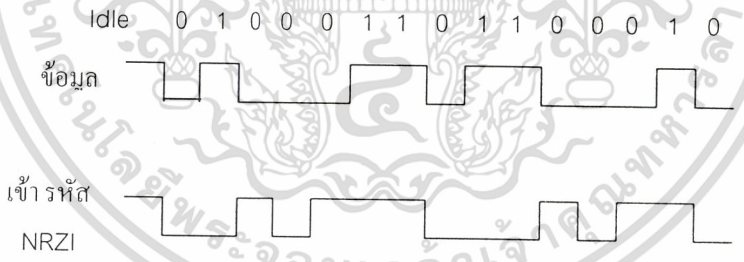
รูปที่ 3.2 แสดงให้เห็นการเข้ารหัสแบบ NRZI ของข้อมูลการเปลี่ยนแปลงของระดับสัญญาณจาก High เป็น Low หรือจาก Low เป็น High จะถือเป็นข้อมูล 0 แต่ถ้าไม่เกิดการเปลี่ยนแปลงของสัญญาณจะถือเป็นข้อมูล 1 แต่ด้วยการเข้ารหัสวิธีนี้หากข้อมูลที่ส่งเป็น 1 เรียงกันหลายๆบิตก็จะไม่เกิดการเปลี่ยนแปลงระดับแรงดันในสายส่งซึ่งอาจทำให้เกิดความผิดพลาดในการ

ส่งข้อมูลได้จึงต้องมีอีกเทคนิคหนึ่งในการส่งข้อมูลเพื่อช่วยป้องกันความผิดพลาดจากเหตุการณ์ดังกล่าวนั่นก็คือการเติมสตัฟบิต

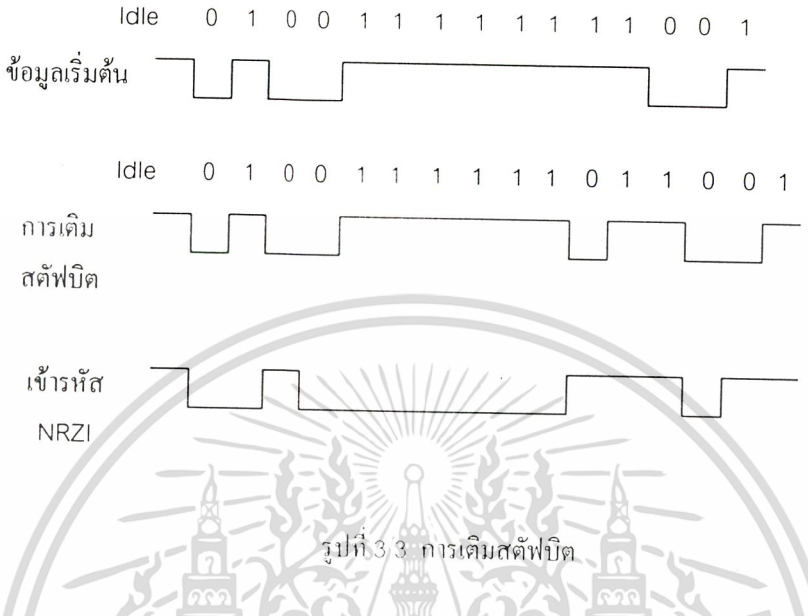
**3.4.1.2)การเติมสตัฟบิต(Bit Stuffing)**

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

จากรูปที่ 3.3 เป็นตัวอย่างของการเติมสตัฟบิตลงในข้อมูลโดยแถวบนสุดของรูปแสดงให้เห็นถึงข้อมูลที่จะส่งจริงๆแต่เนื่องจากข้อมูลที่จะส่งเป็นข้อมูล 1 ติดกันถึง 8 ตัวดังนั้นหลังจากส่งข้อมูล 1 ที่ติดกัน 6 บิตเสร็จแล้ว จะมีการเติมข้อมูล 0 ลงในข้อมูลหลัก และหลังจากการเติมสตัฟบิตเรียบร้อยแล้วก็จะส่งข้อมูลหลักที่อยู่ต่อหลังไปเรื่อยๆจนครบ




รูปที่ 3.2การเข้ารหัส NRZI




รูปที่ 3.3 การเติมสตัฟบิต

# บทที่ 4

# ENUMERATION

## 4.1 ขั้นตอนการคอนฟิกอุปกรณ์(Configuration Process)

จากเนื้อหาทั้งหมดได้กล่าวถึงส่วนประกอบต่างๆของUSB จึงสามารถสรุปการทำงานของระบบเริ่มตั้งแต่มีอุปกรณ์ตังใหม่เชื่อมต่อเข้ามาในระบบว่ามีลำดับขั้นตอนอย่างไรบ้าง การสื่อสารข้อมูลระหว่างโฮสต์และอุปกรณ์ในขั้นเริ่มต้นตั้งแต่อุปกรณ์ตังใหม่เชื่อมต่อเข้ามาในระบบไปจนถึงอุปกรณ์สามารถทำงานได้ตามหน้าที่ จัดเรียกโดยรวมว่า การคอนฟิกอุปกรณ์ โดยมีลำดับดังนี้

- **ขั้นที่ 1** ตรวจสอบสถานะ การเชื่อมต่อและชนิดของอุปกรณ์

  เมื่อมีอุปกรณ์ตัวใหม่เชื่อมต่อเข้ามาในระบบ ฮับจะทราบจากการตรวจสอบระดับแรงดันของสายส่งว่าอยู่ในสถานะใด ซึ่งจะรู้ด้วยว่า อุปกรณ์ที่เชื่อมต่อเข้ามานั้นเป็นอุปกรณ์ความเร็วสูงหรือต่ำ หลังจากนั้นจะเช็คแฟลกสถานะของพอร์ตนั้นๆ เพื่อรอการอ่านกลับจากโฮสต์ ในขั้นตอนนี้ยังไม่มีการจ่ายไฟเลี้ยงไปยังตัวอุปกรณ์
- **ขั้นที่ 2** จ่ายไฟเลี้ยงไปยังพอร์ต

  หลังจากโฮสต์รู้ว่าพอร์ตใดมีอุปกรณ์ตัวใหม่ต่อเข้ามาโดยการอ่านแฟลกสถานะจากฮับแล้ว โฮสต์จะสั่งให้ฮับจ่ายไฟเลี้ยงให้แก่พอร์ตที่ต่อกับอุปกรณ์ตัวใหม่นั้น โดยในขั้นตอนนี้อุปกรณ์จะดึงกระแสได้ไม่เกิน 100 มิลลิแอมป์ อุปกรณ์จะดึงกระแสได้เต็มที่ก็ต่อเมื่อได้รับการคอนฟิกจากโฮสต์แล้วเท่านั้น
- **ขั้นที่ 3** รีเซ็ตพอร์ตของอุปกรณ์ที่เชื่อมต่อใหม่

  เมื่ออุปกรณ์ตัวใหม่ได้รับไฟเลี้ยงเรียบร้อยแล้วโฮสต์จะส่งสถานะรีเซ็ตไปยังพอร์ตนั้น โดยส่งคำสั่งไปยังฮับที่ควบคุมการทำงานของพอร์ตนั้น ซึ่งเมื่ออุปกรณ์ได้รับสถานะนี้แล้วอุปกรณ์นั้นจะต้องรีเซ็ตตัวเองและปรับค่าต่างๆกลับไปยังสถานะเริ่มต้น ทำให้แอดเดรสของอุปกรณ์ที่โดนรีเซ็ตกลับไปอยู่ที่ค่า 0 ซึ่งทำให้โฮสต์สามารถติดต่อกับอุปกรณ์ตัวใหม่นี้ได้
- **ขั้นที่ 4** โฮสต์ร้องขอข้อมูลดิไวซ์ดิซคริปเตอร์จากอุปกรณ์

  หลังจากแอดเดรสของอุปกรณ์ตัวใหม่ถูกรีเซ็ตเป็น 0 แล้ว โฮสต์จะเริ่มร้องขอข้อมูลดิซตริปเตอร์ต่างๆจากอุปกรณ์เพื่อนำค่าที่ได้มาพิจารณาว่าระบบสามารถรองรับอุปกรณ์ตัวใหม่นี้ได้หรือไม่ โดยดิซคริปเตอร์ตัวแรกที่จะถูกอ่านออกมาคือดิไวซ์ดิซคริปเตอร์ การอ่านดิซคริปเตอร์จากอุปกรณ์ก็คือการส่งคำสั่งร้องขอ เข้าไปยังตัวอุปกรณ์ ซึ่งตำแหน่งปลายทางที่จะคอยรับคำสั่งคือเอ็นด์พอยต์ควบคุม

- **ขั้นที่ 5** กำหนดแอดเดรสของอุปกรณ์

เมื่อทราบขนาดแพ็กเก็ตข้อมูลที่จะใช้ในการติดต่อจากการอ่านดิไวซ์ดิซคริปเตอร์แล้ว โฮสต์จะกำหนดแอดเดรสให้แก่ตัวอุปกรณ์ ซึ่งค่าแอดเดรสที่กำหนดขึ้นนี้จะไม่ซ้ำกับอุปกรณ์ตัวอื่นๆในระบบเลย

- **ขั้นที่ 6** โฮสต์อ่านดิซคริปเตอร์ต่างๆเพื่อพิจารณาการรองรับได้ของระบบ

ก่อนที่โฮสต์จะคอนฟิกกับอุปกรณ์ได้ โฮสต์จะต้องอ่านดิซคริปเตอร์ต่างๆภายในตัวอุปกรณ์เพื่อพิจารณาว่าระบบมีทรัพยากรเพียงพอตามที่อุปกรณ์ต้องการหรือไม่ การพิจารณาจะเริ่มจากการตรวจสอบดูว่าอุปกรณ์ตัวนั้นๆมีโหมดการทำงานกี่โหมด หลังจากนั้นจะส่งคำสั่งร้องขอคอนฟิกิวเรชันดิซคริปเตอร์ไปยังอุปกรณ์ ด้วยคำสั่งนี้อุปกรณ์จะส่งคำสั่งดิซคริปเตอร์ที่เหลือทั้งหมดออกมา

เมื่อได้ข้อมูลทั้งหมดออกมาแล้วจะต้องนำมาตัดแยกออกเป็นแต่ละส่วนเองซึ่งเมื่อแยกคอนฟิกิวเรชันดิซคริปเตอร์ออกมาได้แล้วโฮสต์จะทราบว่าถ้าคอนฟิกให้อุปกรณ์ทำงานในแต่ละโหมดจะต้องจ่ายกระแสออกไปทั้งหมดเท่าใด และจะมีดิซคริปเตอร์ต่างๆที่บอกรายละเอียดของอุปกรณ์ออกมาทั้งหมด จากนั้นโฮสต์จะพิจารณาเปรียบเทียบกับทรัพยากรที่เหลือว่าสามารถรองรับการจ่ายไฟเลี้ยงและความเร็วในการรับส่งข้อมูลเพียงพอกับที่อุปกรณ์ต้องการหรือไม่หากสามารถรองรับได้ก็จะเข้าสู่การคอนฟิกขั้นต่อไป แต่ถ้าไม่สามารถรองรับได้โฮสต์จะพิจารณาอุปกรณ์ตัวอื่น และจะทำการหยุดการคอนฟิกเมื่อไม่สามารถรองรับได้อีกต่อไปแล้ว

- **ขั้นที่ 7** โฮสต์กำหนดเลือกคอนฟิกิวเรชันและอินเตอร์เฟซ

หลังจากพิจารณาแล้วว่าระบบสามารถรองรับความต้องการของอุปกรณ์ตัวใหม่ได้ โฮสต์จะส่งคำสั่งเพื่อระบุหมายเลขคอนฟิกิวเรชันและหมายเลขอินเตอร์เฟซ หลังจากสิ้นสุดขั้นตอนนี้อุปกรณ์จะดึงไฟเลี้ยงจากฮับเต็มพิกัดที่อุปกรณ์ระบุไว้ในคอนฟิกิวเรชันดิซคริปเตอร์และเริ่มทำงานตามหน้าที่ซึ่งถือเป็นการจบกระบวนการคอนฟิกอุปกรณ์

**4.2 ENUMERATION STEPS**

กำหนดให้ ToHub เป็นแอดเดรสของฮับ ToIO เป็นแอดเดรสของอุปกรณ์ที่นำมาต่อใหม่

**4.2.1 ToHub :**

Get _Port _Status โฮสต์ค้นหาอุปกรณ์ที่นำมาต่อใหม่

**4.2.2.ToHub :**

Clear_Port_Feature(C_PORT_CONNECTION)เคลียร์ค่าแฟลกใน STATUS_CHANGE register

**4.2.3. ToHub :**

Set _ Port _Feature(PORT_RESET) ฮับจะรักษาการติดต่อไว้อย่างน้อย 10 ms ตรวจสอบ RESET_CHANGE bit ที่ PORT_CHANGE register และ เซท PORT_ENABLE bit ใน PORT_STATUS register

**4.2.4. ToHub :**

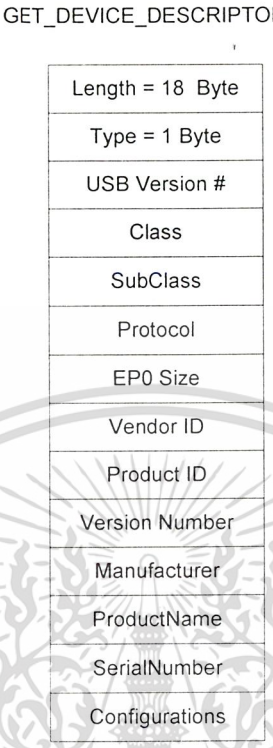
Get _Port _Status : โฮสต์ติดตั้งอุปกรณ์ได้แล้ว

**4.2.5. ToHub :**

Clear _Port _Feature ( C_PORT_RESET) : เคลียร์แฟลกที่ STATUS_CHANGE I/O device ถูกติดตั้งและมีไฟเข้าสู่ตัวอุปกรณ์แล้ว อุปกรณ์จะส่งสัญญาณตอบกลับไปให้ PC Host request โดยกำหนดแอดเดรส 0 เป็น default address และถ้ามี อุปกรณ์ตัวอื่นๆมาต่อ ก็จะต้องรอให้ตัวที่ติดตั้งอยู่ก่อน ติดตั้งเสร็จเรียบร้อยก่อนเพราะจะมีอุปกรณ์ตัวเดียวเท่านั้นที่สามารถส่งสัญญาณ ตอบกลับไปสู่ PC Host request ได้

**4.2.6. ToIO :**

Get _Device _Descriptor : PC Host จะตรวจสอบว่าเป็นอุปกรณ์ชนิดไหนที่ได้ติดตั้งไว้ โดยอุปกรณ์จะส่งสัญญาณไปที่ Device Descriptor ดังรูปที่ 4.1




รูปที่ 4.1 Device response to GET_DEVICE_DESCRIPTOR

**4.2.7. ToIO :**

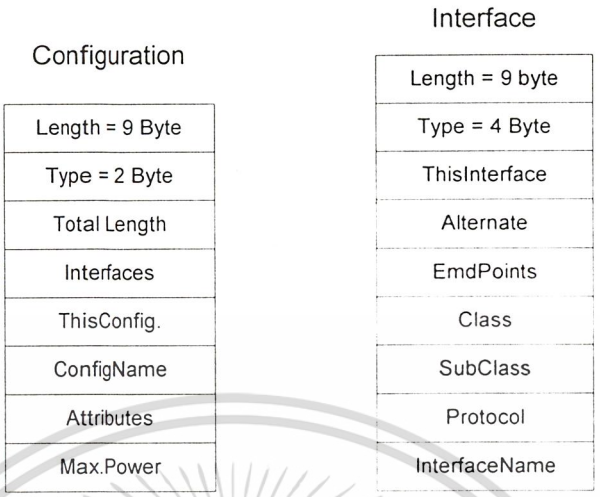
Set _Address : PC Host จะกำหนดค่า address สำหรับอุปกรณ์ที่ติดตั้งใหม่นี้

**4.2.8. ToIO :**

Get _Device _Descriptor : PC Host ส่งสัญญาณไปให้อุปกรณ์ที่แอดเดรสใหม่นี้โดยใช้ Device Descriptor ดังรูปที่ 4.2

**4.2.9.ToIO :**

Get _Configuration _Descriptor : device driver เริ่มเก็บค่า information คือ interfaces , endpoints ดังรูปที่ 4.2




รูปที่ 4.2 Configuration และ Interface Descriptor

**4.2.10. Select Device Driver :**

เลือกใช้driver ให้ตรงกับตัวอุปกรณ์ที่จะติดตั้ง

**4.2.11. ToIO :**

Set _Configuration : อุปกรณ์สามารถใช้งานได้

เมื่อมีการต่ออุปกรณ์ USB เข้ากับคอมพิวเตอร์ Hub driver จะทำการตรวจสอบ Device Descriptor ของ vendor และ product ของ idVendor idProduct คือ

USB\VID_v(4)&PID_d(4)&REV_r(4)

เมื่อ ค่า V(4) คือ ค่าของ 4 bit vendor code

d(4) คือ ค่าของ 4 bit product code

r(4) คือ ค่า revision code

โดยทั่วไป ในdriver ของอุปกรณ์ ซึ่งใช้ไฟล์ .inf มักกำหนดค่า Device Descriptor เป็น

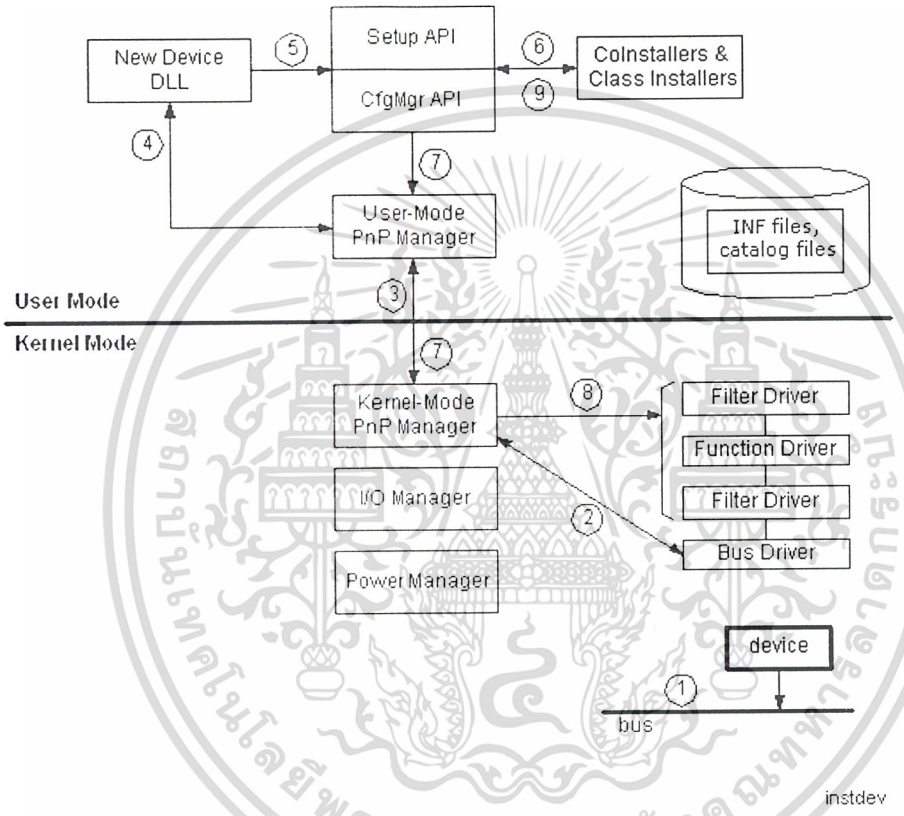
USB\VID_v(4)&PID_d(4)

ในกรณีของ Multiple-Interface USB Devices มี Device Description คือ

USB\VID_v(4)&PID_d(4)&MI_z(2)

ค่าของ z(2) คือ ค่า multiple interface number ซึ่งเป็นค่าที่เกิดจาก interface descriptor

**4.3 PnP Device Installation ( Plug and Play )**




รูปที่ 4.3 แสดงการติดตั้ง Plug and play

**การติดตั้งการทำงานของส่วน Plug and Play**

4.3.1 User ต่ออุปกรณ์เข้ากับคอมพิวเตอร์

4.3.2 อุปกรณ์และเส้นทางการส่งผ่าข้อมูลได้ถูกกำหนดไว้ จากการต่อของอุปกรณ์ใหม่

4.3.3 อุปกรณ์เริ่มขบวนการ Enumeration อาศัย Driver โดยอาศัย kernel-mode PnP



4.3.4 ใช้ kernel – mode เพื่อการติดตั้ง โดยใช้ rundll32.exe และ new

4.3.5 อุปกรณ์ ตัวใหม่ที่ได้ติดตั้ง จะเรียกใช้ DLL calls (Setup API)and PnP Configuration Manager functions ( CfgMgr API )

4.3.6 เรียกใช้ SetupDiCallClassInstaller เพื่อส่งข้อมูล Dif Code

4.3.7 ติดตั้งชุดควบคุมการทำงาน ซึ่งติดต่อกับ kernel mode และเริ่มใช้งานอุปกรณ์ได้

4.3.8 ระบบของ PnP Manager เรียกใช้ฟังก์ชันการทำงานของ driver คือ AddDevice และส่วนประกอบอื่นๆ เช่น filter drivers ของอุปกรณ์นี้ด้วย

**4.4 การร้องขอข้อมูล (data request)**

จากกระบวนการคอนฟิกอุปกรณ์ที่ผ่านมาจะเห็นได้ว่าโฮสต์จะต้องส่งคำสั่งหลายๆชนิดไปที่อุปกรณ์ไม่ว่าจะเป็นการอ่านค่าดิซสคริปเตอร์เพื่อหาข้อมูลต่างๆที่จำเป็นในการเชื่อมต่ออุปกรณ์ รวมไปจนถึงการเลือกค่าคอนฟิวกิวเรชันและอินเตอร์เฟซ การส่งคำสั่งควบคุมทั้งหมดนี้เรียกว่าการร้องขอข้อมูล (data request) ซึ่งแบ่งออกเป็น 2 ชนิด คือการร้องขอข้อมูลจากอูปกรณ์ (device request) และการร้องขอข้อมูลพิเศษจากฮับ (hub request)

**4.4.1 Device Request**

- **Set/clear Feature** ใช้เลือกหรือยกเลิกโหมดการทำงาน Device Remote Wakeup คือ การเรียกอุปกรณ์ให้กลับมาทำงานจากโฮสต์ และ Endpoint Stall คือการ เคลียร์สถานะ Stall ของเอ็นพอยต์ที่ในฟิลด์ Index
- **Set/get Configuration** ใช้สำหรับาคอนฟิวกิวเรชั่นที่ต้องการและขอดูค่าคอนฟิวกิวเรชันที่ใช้งานอยู่
- **Set/get Descriptor** ใช้ในการอ่านค่าดิซสคริปเตอร์ต่างๆออกมาจากอุปกรณ์ โดยสามารถอ่านดิซสคริปเตอร์ได้ 3 ชนิด คือ ดิไวซ์ดิซสคริปเตอร์ คอนฟิวกิวเรชันดิซสคริปเตอร์ และสตริงดิซสคริปเตอร์ ขึ้นอยู่กับการกำหนดค่าในฟิลด์ Value ดังตารางที่ 4.1
- **Set/get Interface** ทำงานคล้ายกับ set/get configuration คือใช้สำหรับกำหนดเลือกอินเตอร์เฟซที่ต้องการและขอดูอินเตอร์เฟซที่ใช้งานอยู่ แต่จะต้องกำหนดค่า Alternatesetting ซึ่งเปรียบเสมือนหัวข้อย่อยในการเข้าไปในคำสั่งด้วยและเมื่ออ่านค่าข้อมูลกลับจะได้ค่า AlternateSetting ที่กำหนดไว้
- **Get status** ใช้อ่านสถานะของอุปกรณ์ แบ่งออกเป็น 3 ระดับคือสถานะของอุปกรณ์ (Device status) สถานะของอินเตอร์เฟซ (Interface status) และสถานะของเอ็นด์พอยต์ (Endpoint Status) โดยข้อมลที่ได้รับกลับมาจะมีขนาด 2 ไบต์เท่ากันทั้ง 3 ระดับ

**4.4.2 Hub Request**

- **Get Status Request** ใช้สำหรับอ่านสถานะของฮับและสถานะของพอร์ตต่างๆ โดยแบ่งออกเป็น 2 ชนิดคือ Hub Status และ Port Status ซึ่งจะมีโค้ดคำสั่งที่แตกต่างกัน ข้อมูลที่ได้กลับมาจะเป็นข้อมูลขนาด 4 ไบต์ โดย 2 บิตแรกจะบอกจะบอกสถานะของฮับหรือพอร์ตปัจจุบัน ในขณะที่ 2 ไบต์หลัง จะบอกการเปลี่ยนแปลงของสถานะทำให้โฮสต์ทราบการเปลี่ยนแปลง
- **Set/Clear Feature Request** ใช้สำหรับเซ็ตหรือเคลียร์ค่าบิตสถานะต่างๆที่กล่าวมาข้างต้น เป็นการสั่งให้ฮับทำงานในบางหน้าที่เช่น การสั่งรีเซ็ต ในขณะที่การทำงานบางบิตเป็นเพียงการแสดงให้ฮับรู้ว่าโฮสต์รับทราบสถานะนั้นๆแล้วให้เคลียร์บิตนั้นทิ้งได้ เช่นการรับทราบสถานการณ์ใช้กระแสเกิน
- **Get Bus State** เป็นคำสั่งที่ใช้ดูสถานะของสายส่ง D+ และ D- เพื่อใช้พิจารณาบางสถานะของสายส่งซึ่งจะได้ค่าสัญญาณของสายส่ง D+ มากับข้อมูลบิต D1 และสัญญาณสายส่ง D- จากข้อมูลบิต D0 โดยค่าสัญญาณที่อ่านกลับมานี้จะอยู่ในช่วงจังหวะ EOF2 สุดท้ายของจังหวะการส่งข้อมูลในแต่ละเฟรม ( 1มิลลิวินาที )

| ข้อมูลไบต์ที่ | ฟิลด์ | ขนาด (ไบต์) | ชนิดข้อมูล | คำอธิบาย |
|---|---|---|---|---|
| 0 | Request Type | 1 | กำหนดเป็นบิต | แสดงคุณสมบัติของชุดคำสั่ง<br>D7 แสดงทิศทางของคำสั่ง<br>0 = ส่งจากโฮสต์ไปยังอุปกรณ์<br>1 = ส่งจากอุปกรณ์ไปยังโฮสต์<br>D6 - D7 ชนิดของคำสั่ง<br>00 = คำสั่งมาตรฐาน<br>01 = คำสั่งเฉพาะของคลาส<br>10 = คำสั่งพาะของผู้ผลิต<br>11 = สงวนไว้ไม่ใช้งาน<br>D0 –D1 ระบุผู้รับ<br>00 = อุปกรณ์ ในที่นี้หมายถึงเอ็นด์พอยด์ 0 ของอุปกรณ์<br>01 = อินเตอร์เฟซ<br>10 = เอ็นด์พอยด์ ( คือเอ็นด์พอยด์อื่นๆ ) |
| 1 | Request | 1 | ค่าที่กำหนด | ข้อมูลไบต์นี้คือคำสั่งที่ต้องการส่งไปยังปลายทาง |
| 2 | Value | 2 | ค่าที่กำหนด | ข้อมูล 2 ไบต์เป็นส่วนขยายของชุดคำสั่งซึ่งขึ้นอยู่กับแต่ละคำสั่งจะใช้ส่งข้อมูลดะไร |
| 4 | Index | 2 | อินเด็กซ์ | อินเด็กซ์อ้างอิงสำหรับบางคำสั่งข้อมูล |
| 6 | Length | 2 | จำนวน | ขนาดของข้อมูลที่จะเกิดการรับหรือการส่งในช่วงดาต้าเสตจ |

ตารางที่ 4.1 ไบต์ที่บรรจุเฟสข้อมูลของเซ็ตอัพเสตจ

| Request Type | คำสั่ง(Request) | Value | Index | Length ดาต้าสเตจ | ข้อมูลในส่วน |
|---|---|---|---|---|---|
| 00000000 b 00000001 b 00000010 b | Clear_Feature (01h) | ค่าfeature ที่ต้องการกำหนด | 0 /interface /endpoint | 0 | ไม่มี |
| 10000000 b | Get_Configuration (08h) | 0 | 0 | 1 | คอนฟิวกิว เรชั่น ที่ใช้งานอยู่ |
| 10000000 b | Get_Description (06h) | ชนิดของ ดิซสคริปเตอร์ ที่จะอ่าน | 0 | ความยาว ดิซสคริปเตอร์ | ข้อมูล ดิซสคริปเตอร์ ที่อ่าน |
| 10000001 b | Get_Interface (0Ah) | 0 | หมายเลข อินเตอร์ เฟซ | 0 | ค่า Alternate Setting ที่ใช้ |
| 10000000 b 10000000 b 10000010 b | Get_Status (00h) | 0 | 0 /interface /endpoint | 2 | ค่าสถานะ |
| 00000000 b | Set_Address (05h) | ค่าแอดเดรส ที่จะกำหนด | 0 | 0 | ไม่มี |
| 00000000 b | Set_Configuration (09h) | ค่าหมายเลข คอนฟิวกิเรชั่น ที่จะเลือก | 0 | 0 | ไม่มี |
| 00000000 b | Set_Descriptor (07h) | ชนิดของดิซสคริปเตอร์ ที่กำหนดค่า | 0 | ความยาว ดิซสคริปเตอร์ | ข้อมูล ดิซสคริปเตอร์ ที่เขียน |

| Request Type | คำสั่ง Requset | Value | Index | Length ดาต้าสเตจ | ข้อมูลในส่วน |
|---|---|---|---|---|---|
| 00000000 b<br>00000001 b<br>00000010 b | Set_Feature (03h) | ค่าของ feature ที่ต้องการกำหนด | 0 /interface /endpoint | 0 | ไม่มี |
| 00000001 b | Set_Interface (0Bh) | ค่า AlternateSetting | หมายเลขอินเตอร์เฟซ | 0 | ไม่มี |
| 10000010 b | Synch_Frame (0Ch) | 0 | หมายเลขเอ็นด์พอยด์ | 2 | หมายเลขเฟรม |

ตารางที่ 4.2 Device Request

| Request Type | คำสั่ง(Request) | Value | Index | Length ดาต้าเสตจ | ข้อมูลในส่วน |
|---|---|---|---|---|---|
| 00100000b | Clear_Fub_Feature (01h) | ค่า feature ที่ต้องการกำหนด | 0 | 0 | ไม่มี |
| 00100011b | Clear_Port_Feature (01h) | ค่า feature ที่ต้องการกำหนด | หมายเลขพอร์ต | 0 | ไม่มี |
| 10100011b | Get_Bus_State (02h) | 0 | หมายเลขพอร์ต | 1 | สถานะของแต่ละพอร์ต |
| 10100000b | Get_Descriptor (06h) | ชนิดของดิซสกริปเตอร์ที่จะอ่าน | 0 | ความยาวของดิซสกริปเตอร์ | ข้อมูลดิซสกริปเตอร์ที่อ่าน |
| 10100000b | Get_Hub_Status (00h) | 0 | 0 | 4 | สถานะของฮับ |
| 10100011b | Get_Port_Status (00h) | 0 | | 4 | สถานะของพอร์ต |
| 00100000b | Set_Hub_Feature (03h) | ค่า feature ที่ต้องการกำหนด | 0 | 0 | ไม่มี |
| 00100011b | Set_Port_Feature (03h) | ค่า feature ที่ต้องการกำหนด | หมายเลขพอร์ต | 0 | ไม่มี |

ตารางที่ 4.3 Hub Request

| ชนิดของ feature | ปลายทาง | ค่าที่นำไปใส่ในชุดคำสั่ง |
|---|---|---|
| Device_Remote_wakeup | อุปกรณ์ | 1 |
| Endpoint_Stall | เอ็นด์พอยด์ | 0 |

ตารางที่ 4.4 Set/Clear Feature

| ชนิดของดิซสกริปเตอร์ที่จะอ่าน | ค่าที่ต้องนำไปใส่ในฟิลด์ Value |
|---|---|
| ดิไวซ์ดอซสกริปเตอร์ | 1 |
| คอนฟิวกิวเรชั่นดิซสกริปเตอร์ | 2 |
| สตริงดิซสกริปเตอร์ | 3 |
| อินเตอร์เฟซดิซสกริปเตอร์ | 4 |
| เอ็นด์พอยด์ดิวสกริปเตอร์ | 5 |

ตารางที่ 4.5 ค่า ฟิลด์ Value ในชุดคำสั่งสำหรับดิซสกริปเตอร์ต่างๆ

# บทที่ 5

# Hardware

**การทำงานในส่วนของฮาร์ดแวร์**

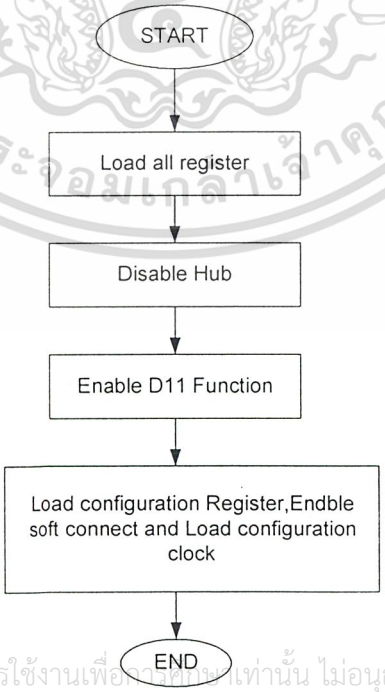
สำหรับเครื่องคาต้าลอคเกอร์ นี้มีการติดต่อกับคอมพิวเตอร์ทาง USB พอร์ต ซึ่งชิ้นงานนี้ใช้ PDIUSBD11 เพื่อช่วยในการติดต่อสื่อสารซึ่งการติดต่อกับ PDIUSBD11 นี้ใช้การสื่อสารในระบบบัส I2C โดยการทำงานในส่วนของโปรแกรมการติดต่อสื่อสารสามารถแบ่งได้ 2 ส่วน คือ

**5.1.) การติดต่อกับ PDIUSBD11**

เป็นส่วนการสื่อสารกับ PDIUSBD11 เพื่อทำการรับส่งข้อมูลและเซ็ตค่าต่างๆ ดังนี้

**5.1.1การเซ็ตค่าอุปกรณ์ (Initialisation Routine)**

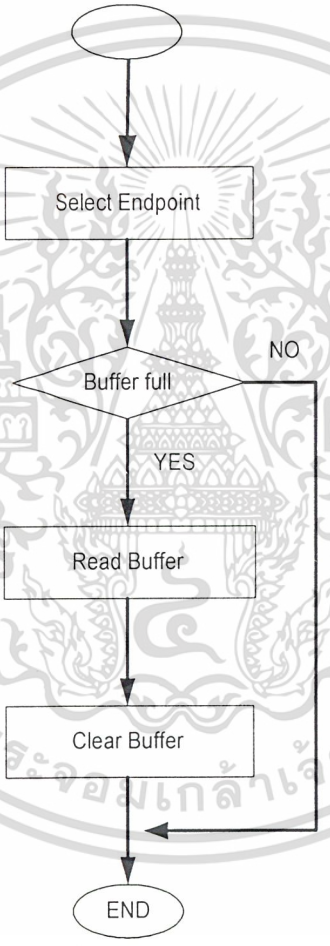
ดังรูปที่ 5.1 การเซ็ตค่าเริ่มต้นด้วยการ อ่านต่ารีจีสเตอร์ที่จะทำการคอนฟิก(Load all register) จากนั้นจะทำการปิดการใช้ยับ(Disable Hub)เนื่องจากอุปกรณ์ตัวนี้เป็นอุปกรณ์ที่ไม่มีอุปกรณ์ต่อพ่วง และจะทำการเปิดโหมดการใช้งานอุปกรณ์(Enable D11 Function) หลังจากทำการเซ็ตให้ PDIUSBD11 ทำงานเสร็จแล้วจะทำการเซ็ตค่าของ สัญญาณ clock และการใช้โหมดการทำงานแบบ soft connect




รูปที่ 5.1 ขั้นตอนการเซ็ต PDIUSBD11

**5.1.2 ) การอ่านข้อมูลจาก PDIUSBD11**

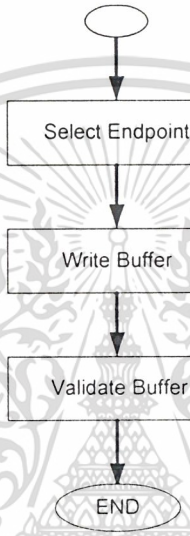
รูปที่5. 2 เป็นการอ่านข้อมูลโดยเริ่มจากการที่ตัวอุปกรณ์(PDIUSBD11)แจ้งว่ามีการส่งข้อมูลมา จากนั้นจะเป็นการเลือกอ่านข้อมูลที่ต้องการ(Select Endpoint) โดยคอนโทรลเลอร์จะทำการตรวจสอบ PDIUSBD11 ว่าบัฟเฟอร์เต็มรึเปล่า(เมื่อมีข้อมูลส่ง D11 จะเซ็ตให้บัฟเฟอร์เต็ม) ถ้าเต็มคอนโทรลเลอร์จะทำการอ่านข้อมูล(Read Buffer) เมื่ออ่านเสร็จจะทำการเคลียร์บัฟเฟอร์(Clear Buffer)เพื่อรองรับข้อมูลถัดไป




รูปที่ 5. 2 การอ่านข้อมูลจาก PDIUSBD11

**5.1.3.) การเขียนข้อมูลจาก PDIUSBD11**

รูปที่ 5.3 เป็นการเขียนข้อมูลโดยเริ่มจากการที่คอนโทรลเลอร์เลือกเอ็นด์พอยต์ที่ต้องการจะส่งจากนั้นจะเป็นการส่งข้อมูลเข้าไปยังตัว D11 (Write Buffer)และจะทำการส่งคำสั่ง Validate Buffer เพื่อเซ็ตให้ D11 ส่งข้อมูลออกไปเมื่อมีสัญญาณ control in ครั้งหน้า

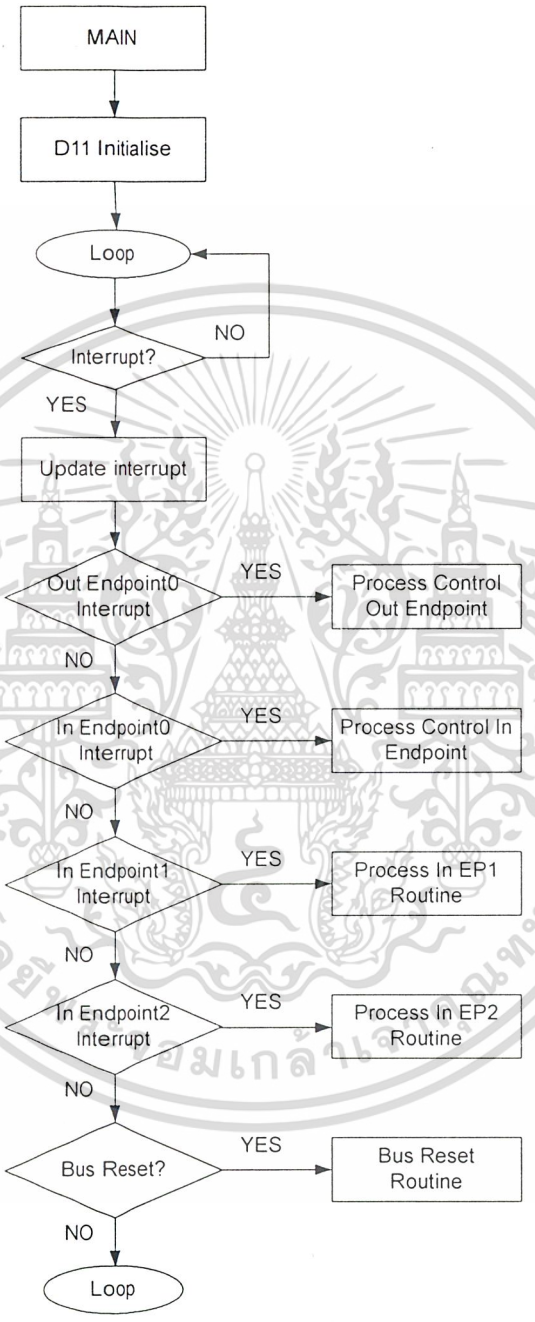



รูปที่ 5.3 การเขียนข้อมูลไปยังโฮสต์

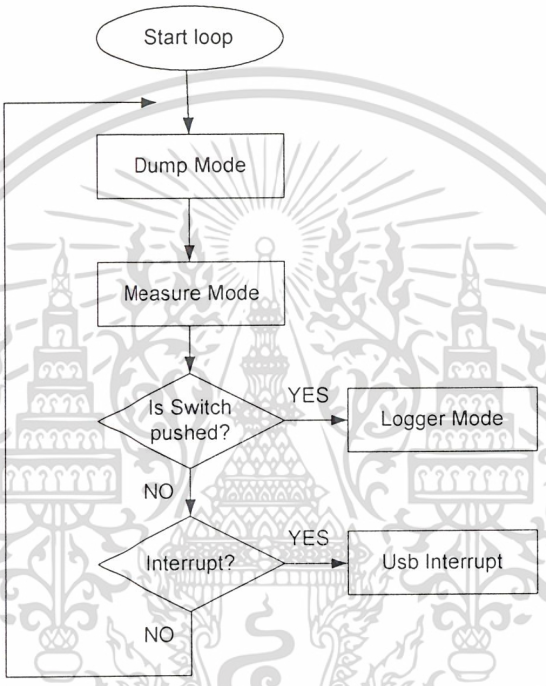
**5.2.) การทำงานของอุปกรณ์**

**5.2.1) ระบบการทำงานของอุปกรณ์ โดยมีขั้นตอนดังนี้**

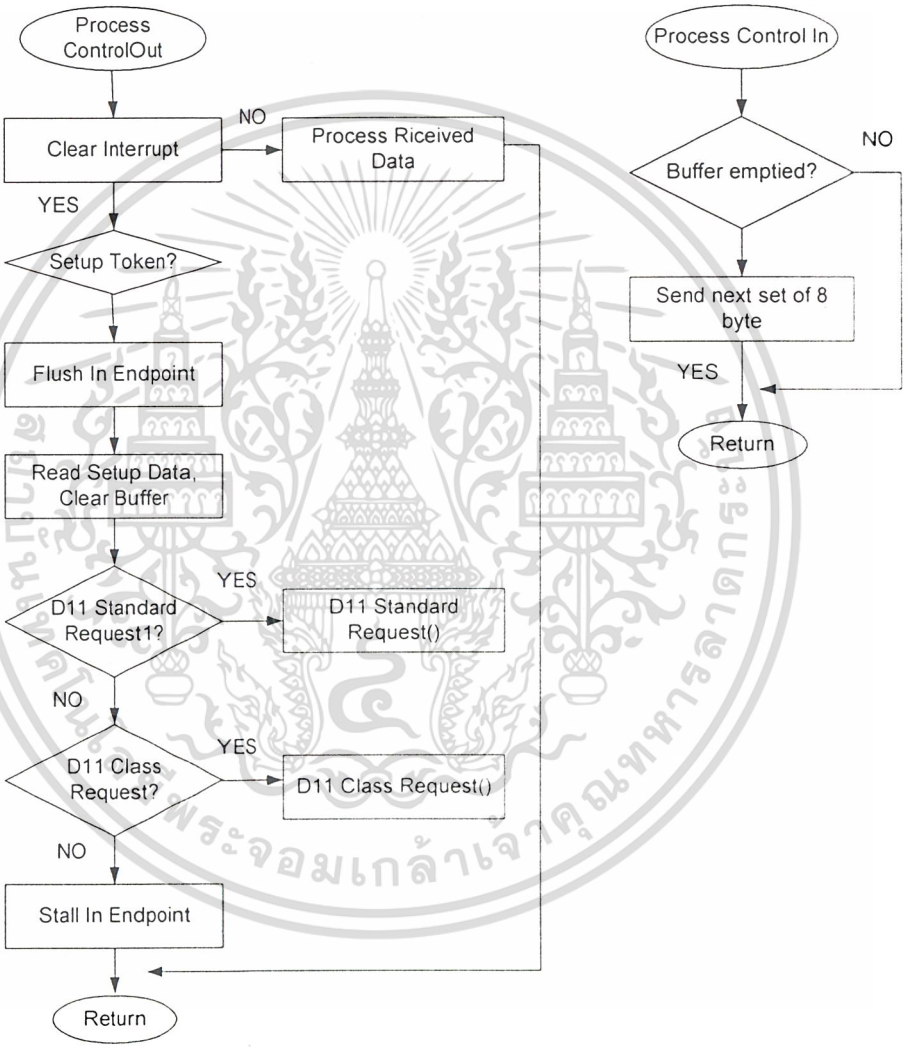
- เซ็ตค่าของตัว PDIUSBD11(D11 Initialise)
- เข้าสู่โปรแกรม Loop ซึ่งเป็นส่วนในการเซ็ตค่าหลักของอุปกรณ์ดังนี้
  - เข้าสู่โปรแกรมในส่วน Dump Mode
  - เข้าสู่โปรแกรมในส่วน Measure Mode
  - ตรวจสอบว่าต้องการให้เก็บค่าอุณหภูมิรีเปล่า(สวิตซ์ถูกกดรีเปล่า)ถ้ามีก็จะทำโปรแกรมในส่วนของ Logger Mode
- ตรวจสอบการอินเตอร์รัพต์ถ้ามีก็ไปทำงานในส่วนของการรับส่งข้อมูลต่างๆแต่ถ้าไม่มีก็จะกลับไปสู่การทำงานในส่วนของ Loop และวนไปเรื่อยๆ
- ในส่วนของการรับส่งข้อมูลต่างๆนั้นเป็นดังรูป 5.4 และ 5.5



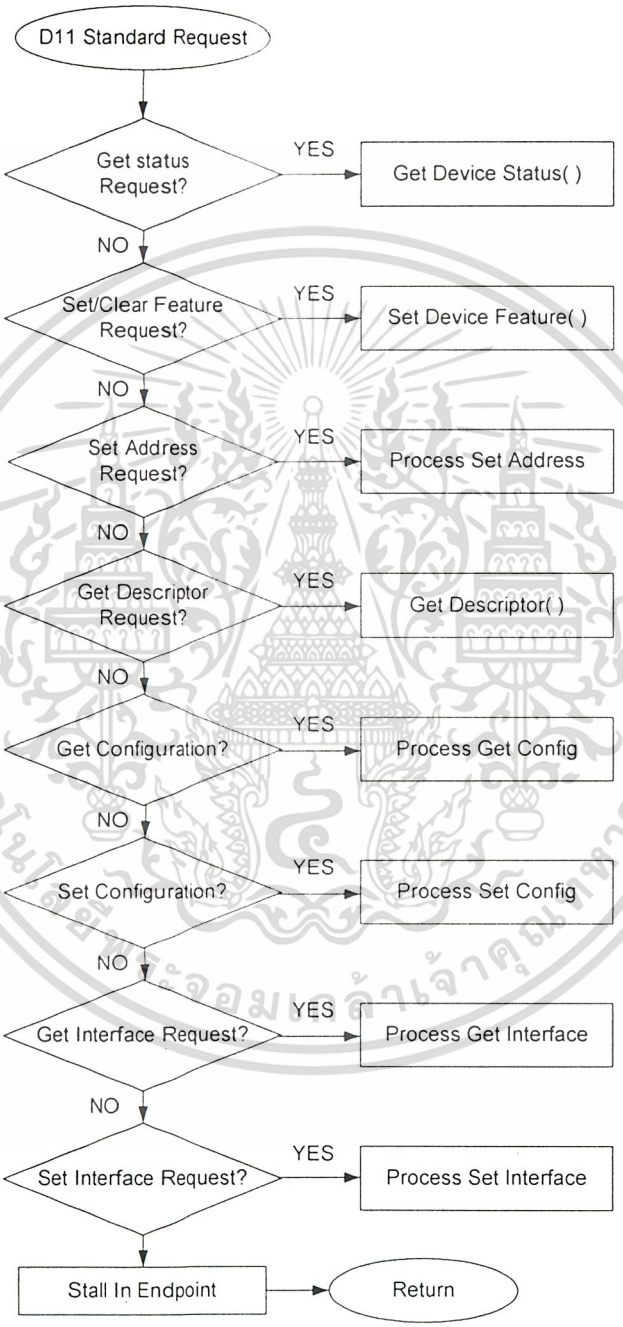

รูปที่ 5.4 การทำงานของฮาร์ดแวร์



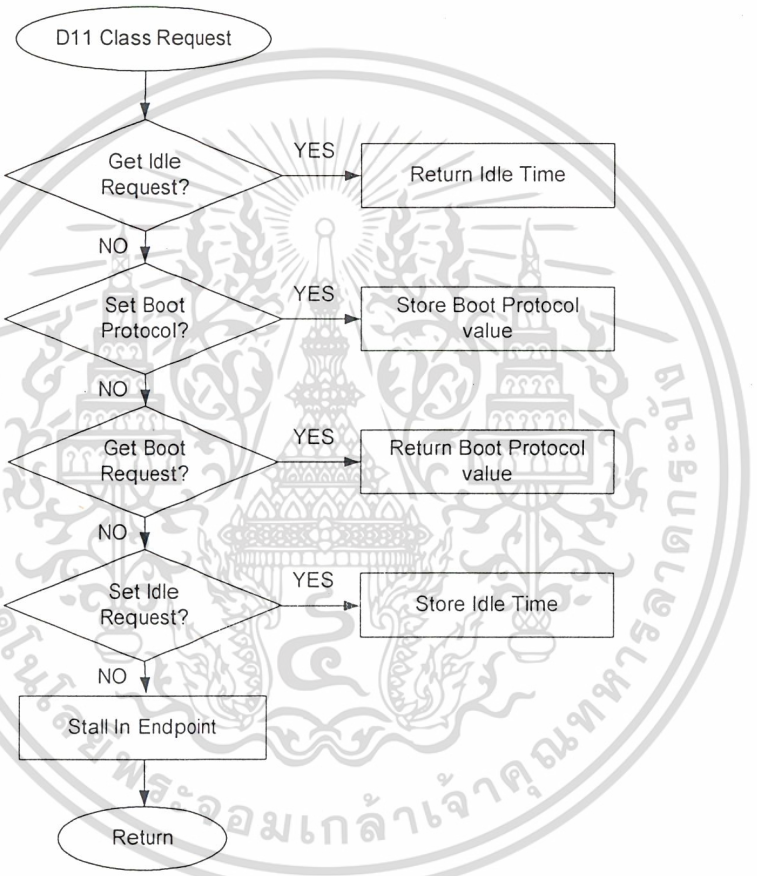

รูปที่ 5.5 การทำงานในส่วนของ Main Loop




รูปที่ 5.6 โปรแกรมในส่วนของการติดต่อกับโฮสต์




รูปที่ 5.7 การทำงานในส่วนของ D11 Standard Request




รูปที่ 5.8 การทำงานในส่วนของ D11 Class Request

**5.2.2) โหมดการทำงานของอุปกรณ์**

- **Logger Mode** วัดและบันทึกค่าข้อมูลลงหน่วยความจำ
- **Measure Mode** วัดและแสดงผลข้อมูลทางหน้าจอคอมพิวเตอร์โดยในโหมดนี้ทำการวัดและแสดงในขณะเวลาปัจจุบัน ไม่บันทึกลงในหน่วยความจำ
- **Dump Mode**

  เป็นการอ่านข้อมูลจากหน่วยความจำมาแสดงผลที่คอมพิวเตอร์
- **Update Interrupt**

  เป็นการอ่านข้อมูลอินเตอร์รัพต์จาก D11 เพื่อดูว่าเป็นอินเตอร์รัพต์ชนิดไหน

- **Process Control Out Endpoint**

เป็นกระบวนการตรวจสอบคำสั่งจากคอมพิวเตอร์ ว่าให้ทำอะไรแล้วไปทำงานที่คำสั่งนั้น โดย out endpoint เป็นเหตุการณ์ที่คอมพิวเตอร์ส่งข้อมูลมายังอุปกรณ์

- **Process Control In Endpoint(0,1,2)**

เป็นกระบวนการที่เกิดขึ้นเมื่อคอมพิวเตอร์มีการร้องขอข้อมูลของเอ็นด์พอยต์ต่างๆ โดยโปรแกรมส่วนนี้จะทำการส่งข้อมูลที่คอมพิวเตอร์ต้องการออกไป

- **Update Interrupt**

  เป็นการอ่านข้อมูลอินเตอร์รัพต์จาก D11 เพื่อดูว่าเป็นอินเตอร์รัพต์ชนิดไหน

- **D11 Standard Request**

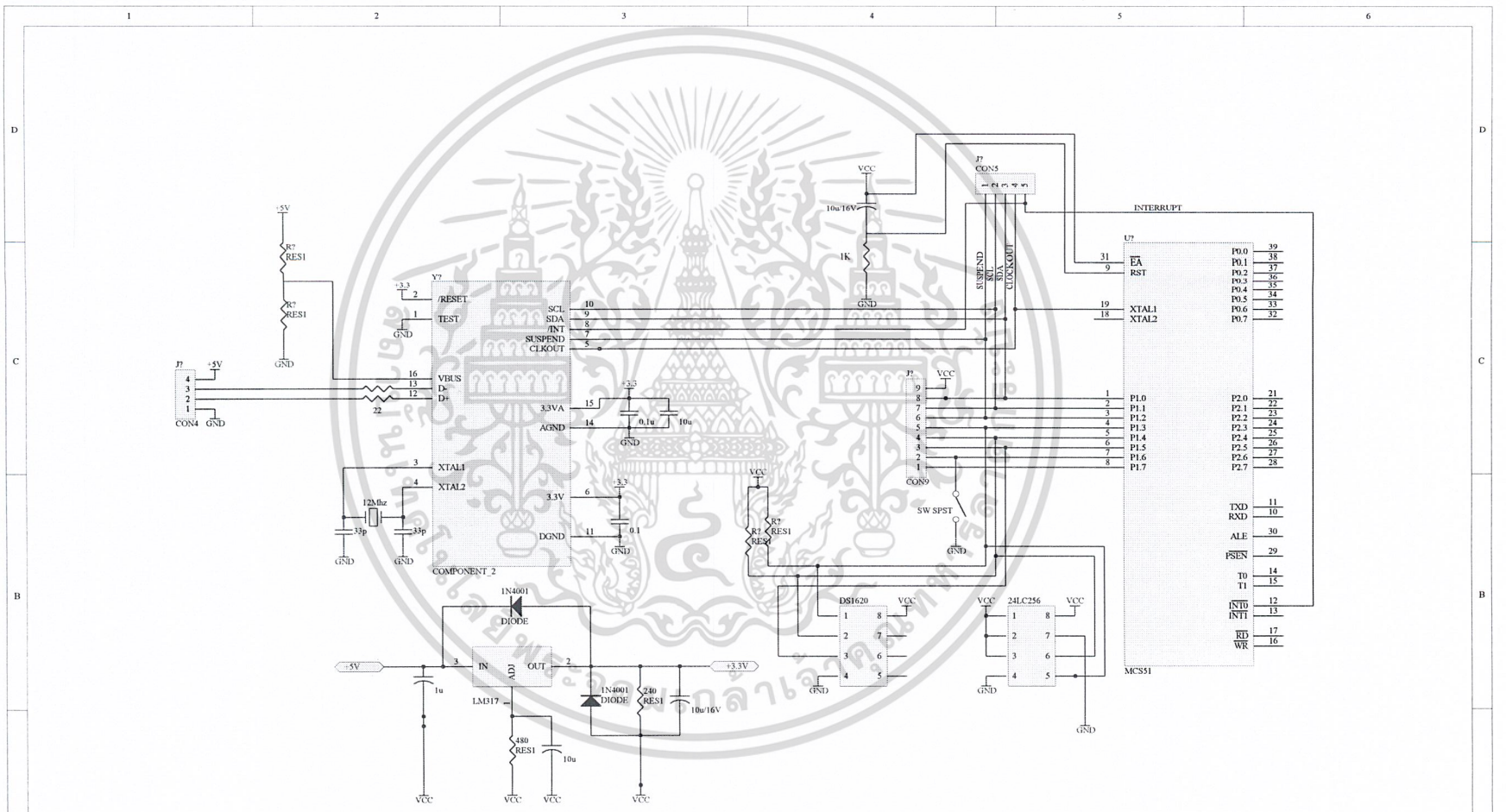
เป็นการเซ็ตค่าข้อมูลของตัวอุปกรณ์ในการทำงาน ในตัวโปรแกรมเป็นการเซ็ตค่าเพื่อรับรู้ว่าคอมพิวเตอร์ต้องการข้อมูลชนิดไหน

- **D11 Class Request**

เป็นการบอกค่าของตัวอุปกรณ์ว่าเป็นอุปกรณ์อะไร ในตัวโปรแกรมเป็นการเซ็ตค่าเพื่อรับรู้ว่าคอมพิวเตอร์ต้องการค่าข้อมูลใด

- **Stall In Endpoint**

  เป็นการส่งสัญญาณตอบว่าได้รับข้อมูลเรียบร้อยแล้ว(ACK)




รูปที่ 5.9 วงจรรวมของอุปกรณ์

# บทที่ 6

# ผลการทดลอง

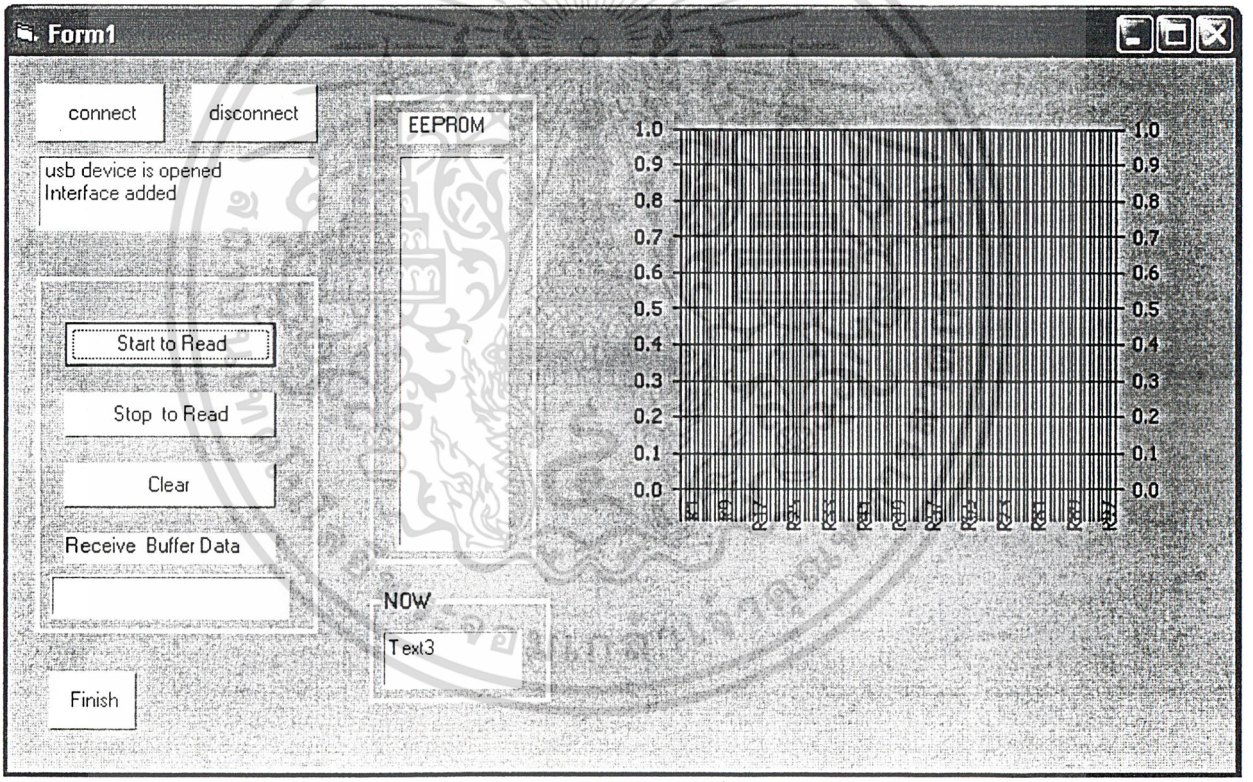
อุปกรณ์ชิ้นนี้เป็นอุปกรณ์ที่ใช้สำหรับตรวจวัดอุณหภูมิโดยแบ่งการทำงานออกเป็น 3 ส่วน ด้วยกันคือ ส่วนวัดอุณหภูมิและแสดงผลในเวลาปัจจุบัน ส่วนที่ใช้อ่านค่าข้อมูลที่บันทึกไว้ในหน่วยความจำ และส่วนที่ใช้ในการเก็บค่าอุณหภูมิไปไว้ในหน่วยความจำ

ในการทดลองการทำงานของอุปกรณ์ชิ้นนี้สามารถแบ่งได้ดังนี้

## 6.1) ส่วนของโปรแกรมแสดงผล

โปรแกรมการแสดงผลต่างๆมีขั้นตอนการใช้งานดังนี้

### 6.1.1)การใช้งานโปรแกรม

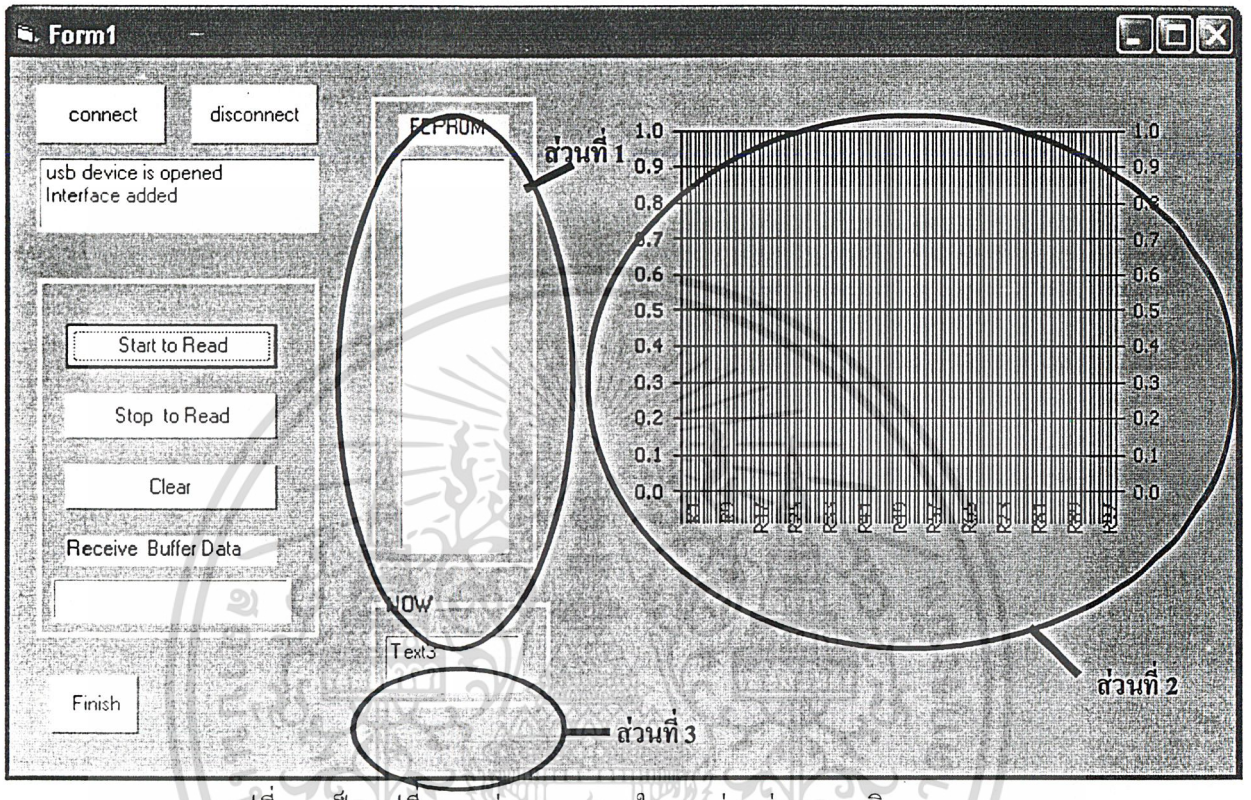



รูปที่ 6.1 หน้าจอแสดงค่าของอุณหภูมิที่อ่านได้

- ในการใช้งานโปรแกรมนี้เริ่มแรกต้องกดปุ่ม “connect” เพื่อทำการติดต่อกับพอร์ต ยูเอสบีเสียก่อน จากนั้นเมื่อต้องการอ่านค่าต่างๆ(ทั้งข้อมูลจากหน่วยความจำและค่าอุณหภูมิในเวลาในปัจจุบัน)
- จากนั้นเมื่อต้องการอ่านค่าต่างๆ(ทั้งข้อมูลจากหน่วยความจำและค่าอุณหภูมิในเวลาในปัจจุบัน)ให้คลิกที่ปุ่ม ”Start to Read”
- เมื่อต้องการหยุดการทำงานของโปรแกรมให้คลิกที่ ”Stop of Read”

- สำหรับปุ่ม “Clear” นั้นเมื่อกดแล้วจะทำการอ่านค่าใหม่ซึ่งปุ่มนี้เปรียบเสมือนปุ่มรีเซ็ตเมื่อการอ่านค่าของอุปกรณ์ผิดพลาด

**6.1.2)การแสดงผลของโปรแกรม**




รูปที่ 6.2 เป็นรูปที่แสดงส่วนแสดงผลในการอ่านค่าอุณหภูมิ

ในส่วนที่ 1 เป็นส่วนที่ใช้แสดงค่าของข้อมูลที่อ่านจากหน่วยความจำโดยจะอ่านค่าจากหน่วยความจำมาทั้งหมด 150 จำนวน โดยในส่วนที่ 2 เป็นส่วนกราฟแสดงผลของค่าที่เก็บไว้ในหน่วยความจำ (ค่าแต่ละค่าในหน่วยความจำเป็นการเก็บข้อมูลทุกๆ 3 วินาที) และในส่วนที่ 3 เป็นส่วนที่ใช้แสดงค่าอุณหภูมิที่วัดได้ในขณะเวลาปัจจุบัน

**6.2) ส่วนของการทำงานของอุปกรณ์**

การทำงานของอุปกรณนั้นแบ่งเป็น 2 ส่วนหลักๆคือ

**6.2.1) การอ่านค่าอุณหภูมิในเวลาปัจจุบันแล้วนำไปแสดงผลทางคอมพิวเตอร์**

ซึ่งทางด้านอุปกรณ์จะทำงานในส่วนนี้เองโดยอัตโนมัติ

**6.2.2) การเก็บค่าของอุณหภูมิเข้าไปไว้ในหน่วยความจำ**

เมื่อต้องการเก็บค่าของอุณหภูมิจะต้องทำการกดสวิตซ์ในตัวอุปกรณ์เสียก่อน ในขณะที่ทำการเก็บค่าอุณหภูมิอยู่นั้น(สวิตซ์ถูกกด) จะไม่มีการทำงานในส่วนอื่นๆของอุปกรณ์ และถ้าต้องการหยุดการเก็บค่าอุณหภูมิก็กดสวิตซ์อีกครั้ง โปรแกรมในส่วนอื่นๆก็จะสามารถทำงานได้โดยปกติ (ปกติแล้วจะทำการเก็บค่าจากภายนอกจึงไม่มีความจำเป็นที่จะต้องทำงานในส่วนอื่นๆ)

# บทที่ 7
# สรุปการทำงาน

จุดประสงค์หลักในการทำโปรเจคนี้ คือการศึกษาการใช้งาน พอร์ต USB ซึ่งในการใช้งานพอร์ต USB นี้จะใช้ไอซีเบอร์ PDIUSBD11 ซึ่งทำหน้าที่เป็นตัวกลางในการติดต่อระหว่างไมโครคอนโทรลเลอร์ กับ พอร์ตยูเอส บี ซึ่ง ไอซีเบอร์นี้มีการต่อต่อ กับ MCS51 ด้วยระบบบัส I2C

ในการทำงานของอุปกรณ์ชิ้นนี้นั้นสามารถที่จะวัดและเก็บอุณหภูมิไว้ในหน่วยความจำได้ตามที่ต้องการ โดยการแสดงผลทางคอมพิวเตอร์นั้นใช้โปรแกรมVisual Basicในการออกแบบการแสดงผลค่าที่อ่านได้ สำหรับอุปกรณ์ชิ้นนี้ก็ยังมีส่วนที่น่าจะปรับปรุงอยู่บางประการเพื่อให้มีรูปแบบที่ดีขึ้น เช่น การจัดส่งค่าของอุณหภูมิที่ได้ทำการเก็บไว้ในหน่วยความจำควรจะมีการเก็บค่าของเวลาที่ได้ทำการเก็บไว้ด้วย ในส่วนของการแสดงผลก็ควรมีการจัดรูปแบบที่สามารถใช้งานได้ง่ายโดยมีการรีเซ็ตค่าหรือเปลี่ยนฟังค์ชันโดยอัตโนมัติ

ในขั้นตอนการสร้างอุปกรณ์ชิ้นนี้ กลุ่มผู้ทำได้รับความรู้ต่างๆอย่างมากในเรื่องของระบบบัส I2C และขั้นตอนในการติดต่อกับ ยูเอสบี พอร์ตที่สำคัญยังได้รู้เรื่องของวงจรของระบบ I2C และ ยูเอสบี อีกด้วยซึ่งวงจรประเภทที่เกี่ยวกับความถี่สูงๆนั้นค่าของอุปกรณ์ที่ต่างกันเพียงเล็กน้อยอาจส่งผลอย่างมากกับชิ้นงานก็ได้

# ภาคผนวก

# หนังสืออ้างอิง


กิตติ ภักดีวัฒนะกุล, " Visual Basic 6 ฉบับโปรแกรมเมอร์" , หจก.ไทยเจริญการพิมพ์ , 621 หน้า,2543

ลภน สุภาพ ," เจาะลึกให้เข้าใจในUSB" , วารสารเซมิคอนดักเตอร์ , ฉบับที่ 225 ,2544 , หน้า 159-166

ลภน สุภาพ ," เจาะลึกให้เข้าใจในUSB" , วารสารเซมิคอนดักเตอร์ , ฉบับที่ 226 ,2544 , หน้า 174-181

ลภน สุภาพ ," เจาะลึกให้เข้าใจในUSB" , วารสารเซมิคอนดักเตอร์ , ฉบับที่ 227 , 2544 , หน้า 181-188

ลภน สุภาพ ," เจาะลึกให้เข้าใจในUSB" , วารสารเซมิคอนดักเตอร์ , ฉบับที่ 229 , 2544 , หน้า 158-169

ลภน สุภาพ ," เจาะลึกให้เข้าใจในUSB" , วารสารเซมิคอนดักเตอร์ , ฉบับที่ 230 , 2544 , หน้า 159-168

สมพล วรวุฒิยานนท์ , " USB บัสอนุกรมครอบจักรวาล" , วารสารเซมิคอนดักเตอร์ , ฉบับที่ 215, 2543 , หน้า 169-173

John Hyde, "USB Design by Example" ,Willey Computer Publishing,John Wiley & Sons,Inc. , 368 p.,1999 Intel Corporation.