การเชื่อมต่อ USB กับพอร์ต IEEE-488 (GPIB) ของเครื่องมือวัด

USB port to interface IEEE-488 (GPIB) instruments

โดย

นายภัคพงศ์ สุขสนิท

นายอุดมศักดิ์ เสือสาวะถี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

การเชื่อมต่อ USB กับพอร์ต IEEE-488 (GPIB) ของเครื่องมือวัด

USB port to interface IEEE-488 (GPIB) instruments

โดย

นายภัคพงศ์ สุขสนิท 45015023

นายอุดมศักดิ์ เสือสาวะถี 45015044

อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

อาจารย์สมปอง วิเศษพานิชกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การเชื่อมต่อ USB กับพอร์ต IEEE-488 (GPIB) ของเครื่องมือวัด**

USB port to interface IEEE-488 (GPIB) instruments

ผู้จัดทำ

1 .นายภัคพงศ์ สุขสนิท 45015023

2. นายอุดมศักดิ์ เสือสาวะถี 45015044

..... อาจารย์ที่ปรึกษา

(รศ.ดร. กอบชัย เดชหาญ)

..... อาจารย์ที่ปรึกษา

(อาจารย์สมปอง วิเศษพานิชกิจ)

การเชื่อมต่อ USB กับพอร์ต IEEE-488 (GPIB) ของเครื่องมือวัด

USB port to interface IEEE-488 (GPIB) Instruments

โดย นายภัคพงศ์ สุขสนิท 45015023
นายอุดมศักดิ์ เสือสาวะถี 45015044

อาจารย์ที่ปรึกษา รศ.ดร. กอบชัย เดชหาญ
อาจารย์สมปอง วิเศษพานิชกิจ

บทคัดย่อ

เนื่องจากการเชื่อมต่ออุปกรณ์ในปัจจุบันโดยผ่านพอร์ต USB มีอย่างแพร่หลายและได้รับความนิยมเป็นอย่างมาก โครงงานนี้จึงได้ประยุกต์การใช้งานพอร์ต USB ที่มีอยู่ในเครื่องคอมพิวเตอร์เพื่อประโยชน์ทางวิศวกรรมโดยการนำมาใช้เชื่อมต่อกับเครื่องมือวัดผ่านที่พอร์ต IEEE-488 (GPIB) โดยจะมีฮาร์ดแวร์เป็นตัวแปลงสัญญาณทางด้านพอร์ต USB และด้าน GPIB ให้สามารถเชื่อมต่อกันได้ซึ่งจะสามารถควบคุมเครื่องมือวัดโดยใช้คอมพิวเตอร์ในการควบคุม เป็นผลให้การใช้งานเครื่องมือวัดสะดวกสบาย และสามารถประยุกต์ใช้งานได้มากขึ้น เพื่อประโยชน์ของผู้ใช้

Abstract

The connection of equipments via USB port is widely used , this project proposes an application of USB port which installed in computer board in order to be useful for engineering experiment . The connection of equipment via IEEE-488 port or GPIB will have the hardware part for converting the signal at USB port and GPIB port . It is able to control the equipment by using computer as controller . The advantage is the convenience to use the equipment and it depends on the computer and application .

สารบัญ

สารบัญรูป

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาของปริญญานิพนธ์

จากความต้องการที่จะรวมการควบคุมอุปกรณ์เครื่องมือวัดต่าง ๆ ไว้ที่ศูนย์กลางการควบคุมเพียงจุดเดียวซึ่งใช้คอมพิวเตอร์เป็นตัวควบคุมนั้น นิยมที่จะใช้เครื่องคอมพิวเตอร์เป็นศูนย์กลางการควบคุม ดังนั้นจึงต้องมีการสื่อสารข้อมูลระหว่างเครื่องมือวัดกับคอมพิวเตอร์ ซึ่งมาตรฐานหนึ่งที่นิยมใช้งานกันอย่างแพร่หลายในวงการเครื่องมือวัดคือ มาตรฐาน IEEE-488(GPIB) ซึ่งเป็นมาตรฐานพอร์ตขนานแบบหนึ่งซึ่งมีการส่งข้อมูลขนานแบบ 8 บิต และสามารถสามารถต่อพ่วงเครื่องมือวัดหลาย ๆ ตัวร่วมกันได้แต่เครื่องคอมพิวเตอร์ทั่วไปนั้น จะไม่มีส่วนการอินเตอร์เฟสกับพอร์ต GPIB นี้ติดตั้งมาด้วย ดังนั้นจึงจำเป็นต้องซื้อการ์ดอินเตอร์เฟสมาติดตั้งเพื่อใช้งาน สำหรับการ์ดอินเตอร์เฟสนั้นมีราคาสูงและซอร์ฟแวร์ไดร์เวอร์ที่ให้มานั้นก็ยากต่อการพัฒนา เนื่องจากจำเป็นต้องเข้าใจในจุดที่ผู้ออกแบบได้ออกแบบไว้ ซึ่งก็เป็นเหตุที่มาในการเริ่มต้นในการทำปริญญานิพนธ์นี้

ในปริญญานิพนธ์นี้นั้นได้เลือกใช้พอร์ต USB ของเครื่องคอมพิวเตอร์ในการเชื่อมต่อกับการ์ดอินเตอร์เฟส GPIB ที่สร้างขึ้น เนื่องจากว่าพอร์ต USB สามารถเชื่อมต่อกับพีซีคอมพิวเตอร์ทั่วไปและคอมพิวเตอร์แบบโน้ตบุค อีกทั้งคุณสมบัติในการเชื่อมต่อแบบ ปลั๊กแอนด์เพลย์ (Plug and Play) ของ USB ที่สามารถเชื่อมต่ออุปกรณ์เข้ากับระบบในขณะที่เครื่องคอมพิวเตอร์เปิดใช้งานอยู่ ซึ่งก็เป็นคุณสมบัติเด่นของ USB ที่ทำให้การใช้งานสะดวกและคล่องตัวมากขึ้น

1.2 วัตถุประสงค์ของปริญญานิพนธ์

- เพื่อให้สามารถควบคุมเครื่องมือวัดที่ใช้มาตรฐานการส่งข้อมูลแบบ GPIB ด้วยคอมพิวเตอร์ผ่านทางพอร์ต USB ได้
- เพื่อศึกษารูปแบบโปรโตคอลของ GPIB เพื่อเป็นพื้นฐานในการพัฒนาดัดแปลงให้ใช้งานร่วมกันกับพอร์ตสื่อสารอื่น ๆ ได้ในอนาคต
- เพื่อใช้งานแทนการ์ดอินเตอร์เฟส GPIB ที่มีราคาสูง

1.3 ขอบเขตของโครงงาน

- ใช้การควบคุมสั่งการผ่านคอมพิวเตอร์โดยเขียนโปรแกรมควบคุมโดยโปรแกรม Microsoft Visual C++
- ในส่วนของโปรแกรมควบคุมที่ได้ออกแบบไว้นั้นให้รองรับอุปกรณ์ดังนี้
    - Oscilloscope ยี่ห้อ Tektronix รุ่น TDS3032
    - Function Generator ยี่ห้อ Sony Tektronix รุ่น AFG310

หากเป็นการใช้งานกับเครื่องมือวัดอย่างอื่นนอกเหนือจากนี้ สามารถทำได้โดยการเขียนโปรแกรมในส่วนควบคุมทางคอมพิวเตอร์ขึ้นใหม่ให้รองรับคำสั่งของอุปกรณ์นั้นตามต้องการ

- ติดต่อสื่อสารกับเครื่องคอมพิวเตอร์ผ่านทางพอร์ต USB

1.4 เนื้อหาของปริญญานิพนธ์

ในบทที่ 2 จะกล่าวถึงทฤษฎีและหลักการพื้นฐานของมาตรฐาน IEEE-488 (GPIB)

ในบทที่ 3 จะกล่าวถึงทฤษฎีและหลักการพื้นฐานของ USB

ในบทที่ 4 จะอธิบายถึงการออกแบบส่วนอินเตอร์เฟสของระบบ

ในบทที่ 5 การทดลองและผลการทดลอง

ในบทที่ 6 บทสรุปและวิจารณ์

บทที่ 2

ระบบบัสมาตรฐาน IEEE – 488 (GPIB)

2.1 ความเป็นมาของระบบบัส IEEE – 488 (GPIB)

เครื่องมือวัดต่างๆ ที่สามารถเชื่อมต่อกับอุปกรณ์ภายนอกได้นั้น ในอดีตนั้นบริษัทผู้ผลิตแต่ละบริษัทจะทำการออกแบบระบบบัสเป็นของตนเอง ซึ่งทำให้ยุ่งยากในการที่จะนำอุปกรณ์เครื่องมือวัดจากบริษัทต่างๆ มาต่อรวมภายในระบบเดียวกันอีกทั้งเป็นอุปสรรคในการพัฒนาเครื่องมือวัดเพื่อให้รองรับกับผู้ใช้ที่หลากหลาย ดังนั้นจึงเกิดการรวมกลุ่มกันของผู้ผลิตเครื่องวัดในสหรัฐอเมริกาเพื่อพัฒนาระบบบัสขึ้นใช้งานร่วมกัน ซึ่งทางประเทศเยอรมณีก็ได้มีการร่วมมือกันออกแบบระบบบัสที่จะใช้งานร่วมกันขึ้นเช่นกัน โดยการช่วยการช่วยเหลือของสถาบัน IEC (International Electro technical Commission) จนกระทั่งปี ค.ศ. 1972 สถาบันวิศวกรไฟฟ้าและอิเล็กทรอนิกส์ของอเมริกา (Institute of Electrical and Electronics Engineering : IEEE) ได้ประชุมเพื่อวางแนวทางของระบบบัสข้อมูลดังกล่าว ซึ่งในที่สุดบริษัท Hewlett Packard ได้เสนอระบบบัสซึ่งทาง HP ได้พัฒนามาก่อนแล้วซึ่งมีชื่อว่า HPIB (Hewlett Packard Interface Bus) ซึ่งทาง IEEE ได้ยอมรับเป็นลำดับที่ 488 ในปี ค.ศ. 1975 เรียกว่า IEEE std 488-1975 และได้มีการปรับปรุงอีกครั้งในปี 1978 เรียกว่า IEEE std 488-1978 ซึ่งก็คือระบบ IEEE 488 (GPIB) นั่นเองซึ่งระบบบัสนี้ได้ถูกนำไปใช้ในระบบของ IEC ด้วย เรียกว่าระบบบัส IEC625-1 โดยรายละเอียดเหมือน IEEE std 488-1978 ทุกประการเพียงแต่ตากต่างกันในตำแหน่งของขั้วต่อสัญญาณเท่านั้น

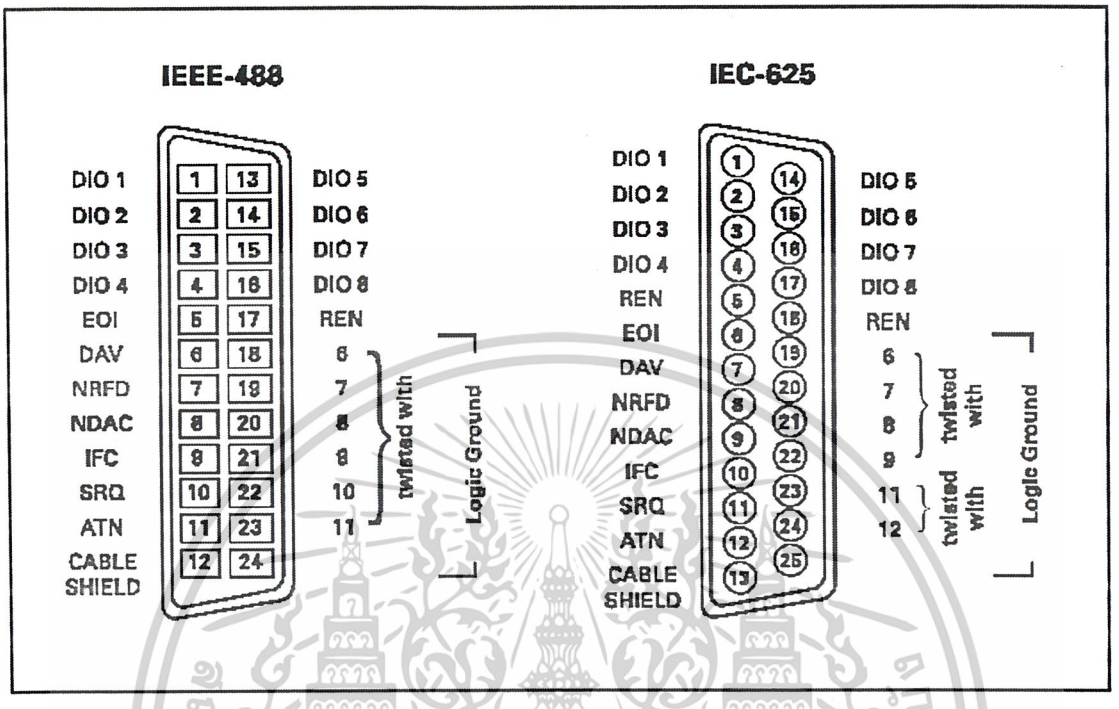
2.2 คุณสมบัติและข้อจำกัดในการต่อพ่วงอุปกรณ์

ในการระบุ Address ของอุปกรณ์ต่อพ่วงในระบบ IEEE-488 นี้นั้นมี Address ที่สามารถเป็นไปได้อยู่ระหว่าง 0 ถึง 30 ซึ่งในช่วงเวลาหนึ่ง ๆ นั้นตัวควบคุมจะสามารถติดต่ออุปกรณ์เครื่องมือวัดได้เพียง Address เดียวกันเท่านั้น โดยใน 1 ช่วงเวลาอุปกรณ์นั้นจะถูกควบคุมให้ทำงานเป็นตัวส่งหรือตัวรับเท่านั้นไม่สามารถกระทำในเวลาเดียวกันได้ สำหรับส่วนของอุปกรณ์ต่อพ่วงต่าง ๆ นั้นจะมีอยู่ด้วยกัน 4 รูปแบบได้แก่

- อุปกรณ์ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว (Talker) เช่น เครื่องวัดแรงดัน (Volt Meter) เป็นต้น
- อุปกรณ์ที่ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว (Listener) เช่น เครื่องพิมพ์ (Printer) เป็นต้น
- อุปกรณ์ที่ทำหน้าที่เป็นทั้งตัวส่งและตัวรับ (Talker & Listener) เช่นเครื่องมือต่าง ๆ ที่ควบคุมได้ภายนอกเครื่องคอมพิวเตอร์ เป็นต้น
- อุปกรณ์ที่ทำหน้าที่เป็นทั้งตัวส่ง, ตัวรับและตัวควบคุม เช่นคอมพิวเตอร์ เป็นต้น

สำหรับระบบบัส GPIB นั้น สามารถต่อพ่วงอุปกรณ์ได้ในขณะเดียวกันเพียง 16 เครื่องเท่านั้นซึ่งเป็นข้อจำกัดทางด้านการขับกระแสผ่านบนบัสโดยมีการจำกัดปริมาณกระแส ตั้งแต่ 0 ถึงไม่เกิน 48 mA ซึ่งในอุปกรณ์ต่อพ่วงตัวหนึ่ง ๆ จะกินกระแสประมาณ มากกว่าหรือเท่ากับ 3 mA ดังนั้นเมื่ออุปกรณ์ ต่อพ่วง 15 ตัว กับ 1 ตัวควบคุม (15+1)x3 = 48mA นั่นเอง สำหรับเรื่องของสถานะ Logic ของ GPIB นั้น

ในขณะที่เรากล่าวถึง Logic 0 จะหมายถึงสถานะเอาท์พุตเป็น High (“1”) และในขณะที่กล่าวถึง Logic 1 จะหมายถึงสถานะเอาท์พุตที่เป็น Low (“1”) ซึ่งเป็นจุดที่ควรระวังในการทำความเข้าใจระบบด้วย



รูปที่ 2.1 แสดงตำแหน่งขาสัญญาณของมาตรฐาน IEEE-488 และมาตรฐาน IEC625-1

2.3 สัญญาณต่างๆ ภายในระบบ GPIB

ในระบบ GPIB นั้นสายสัญญาณถูกแบ่งออกเป็น 3 ส่วนหลัก ๆ ได้แก่

2.3.1 สายสัญญาณข้อมูล (Data Input Output 1-8 : DIO 1-8) เป็นสัญญาณข้อมูลหลักในการส่งถ่ายข้อมูลระหว่างกันของอุปกรณ์เครื่องมือวัดและตัวควบคุม

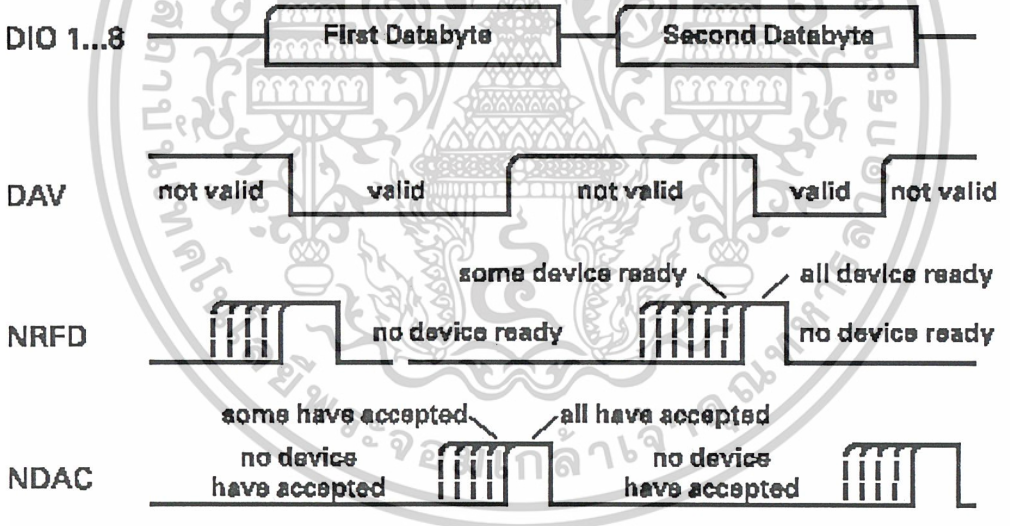
2.3.1 สายสัญญาณควบคุมการรับส่งข้อมูล (Hand shake lines: DAV, NRFD and NDAC) ประกอบไปด้วยสายสัญญาณ 3 สายเพื่อใช้ในการควบคุมจังหวะของการรับและส่งข้อมูลประกอบด้วย

- **DAV (Data Valid)** ในการส่งข้อมูลแต่และชุดข้อมูลนั้นต้องมีการตรวจสอบความพร้อมของอุปกรณ์ต่าง ๆ เป็นอันดับแรก ซึ่งสามารถตรวจสอบได้จากสัญญาณ NRFD โดยเมื่ออุปกรณ์ทุกตัวพร้อมที่จะรับคำสั่ง สัญญาณ NRFD จะเป็น Logic 0 (Output High) สัญญาณ DAV นี้จะเปลี่ยนเป็น Logic 1 (Output Low) เพื่อแจ้งให้เกิดการรับค่าของข้อมูลที่รออยู่บน Data Bus ซึ่งการส่งข้อมูลนี้จะขึ้นอยู่กับสัญญาณ NDAC โดยเมื่อใดก็ตามที่สัญญาณ NDAC เปลี่ยนเป็น Logic 0 นั่นหมายถึงการที่ข้อมูลถูกรับเข้าสู่อุปกรณ์ต่างๆ เรียบร้อยแล้ว สัญญาณ DAV จะถูกกำหนดให้เป็น Logic 0 เพื่อแจ้งว่าสามารถที่จะรับข้อมูลชุดใหม่เข้ามาได้แล้วนั่นเอง

- **NRFD (not ready for data)** สายสัญญาณนี้ถูกขับโดยอุปกรณ์ทุกตัว เมื่อได้รับคำสั่งใด ๆ เมื่อตัวรับได้รับชุดข้อมูลคำสั่ง มันจะทำการสั่งการเพื่อบ่งบอกว่า ขณะนี้อุปกรณ์ใด ๆ พร้อมหรือไม่พร้อมที่

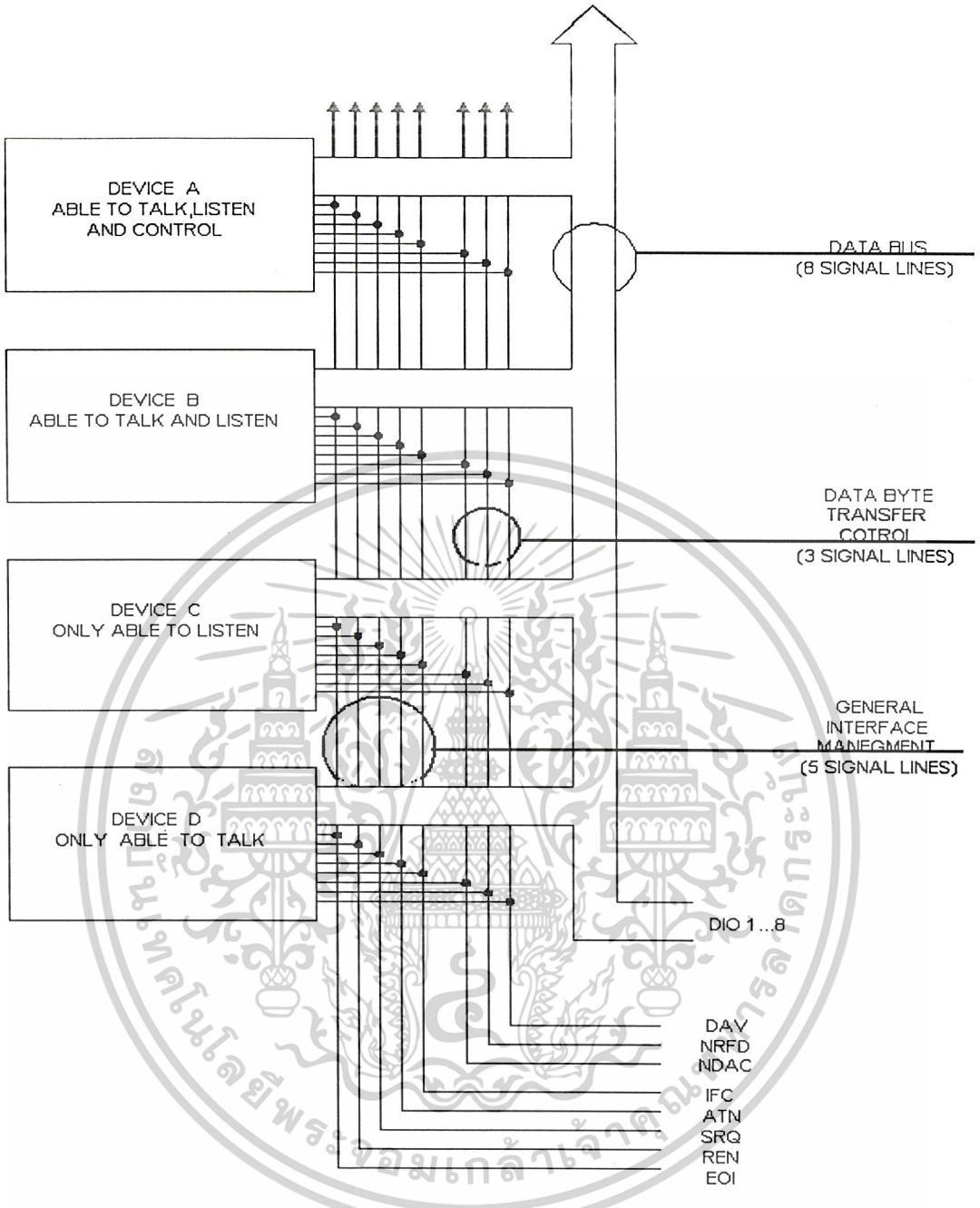
จะรับชุดข้อมูลเมื่อมันอยู่ในสถานะ Logic 0 มันจะบ่งบอกว่าขณะนี้อุปกรณ์ที่เชื่อมต่อพร้อมที่จะรับคำสั่ง ซึ่งทั้งนี้การรับข้อมูลต้องขึ้นอยู่กับสถานะของ DAV ด้วย ซึ่งอุปกรณ์ตัวรับจะสั่งการ NDAV ให้เป็น Logic 0 และเริ่มทำการเก็บข้อมูลโดยหลังจากการเก็บข้อมูลเสร็จเครื่องรับจะแจ้งกลับด้วยสัญญาณ NDAC เพื่อบ่งบอกให้ทราบว่าข้อมูลถูกเก็บเรียบร้อยแล้ว (NDAC เป็น Logic 1) ซึ่งทั้งนี้ในขณะที่เริ่มมีการเก็บข้อมูลสัญญาณ DAV ต้องกลับมาอยู่ในสถานะ "Data not valid" (Logic 0) ด้วยเพราะเมื่อใดก็ตามที่อุปกรณ์ใด ๆ ส่งสัญญาณ NRFD ให้เป็น Logic 1 สัญญาณ DAV ไม่สามารถที่จะเป็น Logic 1 ได้ เพราะไม่เช่นนั้นจะไม่สามารถรับชุดข้อมูลชุดต่อไปได้

- NDAC (not data accepted) สายสัญญาณนี้จะถูกขับเมื่อมีการรับคำสั่งการใด ๆ หรือชุดข้อมูลคำสั่ง โดยเป็นสายสัญญาณที่บ่งบอกว่าสัญญาณข้อมูลที่รอบนบัสข้อมูลนั้นได้รับการเก็บไว้แล้วหรือไม่ โดยความสำคัญของสัญญาณ NDAC นี้คือบ่งบอกถึงว่าอุปกรณ์นั้น ๆ พร้อมที่จะรับข้อมูลคำสั่งชุดใหม่แล้วหรือยังซึ่งมีผลเกี่ยวเนื่องกับสัญญาณ DAV ด้วยนั่นคือ ขณะที่รับข้อมูลไว้บนสัญญาณ DAV เป็น Logic 0 หรือสถานะ " Data Accepted " นั่นเอง ซึ่งทางด้านส่งก็จะรับรู้และทำการเปลี่ยนสัญญาณ DAV เป็น Logic 0 เพื่อรอรับข้อมูลชุดใหม่ต่อไปและเมื่อตัวรับข้อมูลเรียบร้อยแล้วก็จะเปลี่ยนสถานะของ NDAV ให้เป็น Logic 1 อีกครั้งด้วยเช่นกัน



รูปที่ 2.2 แสดงความสัมพันธ์ของสัญญาณควบคุมการรับส่งข้อมูล

2.3.3 สายสัญญาณควบคุมเชื่อมต่อ (The Interface Management Line : ATN, IFC, REN, SRQ หรือ EOI) ในส่วนของสายสัญญาณควบคุมการเชื่อมต่อนี้ จะเป็นกลุ่มของสายสัญญาณที่ใช้ควบคุมและจัดการการเชื่อมต่อต่าง ๆ โดยสายสัญญาณบางเส้นนั้นควบคุมจากส่วนควบคุมเท่านั้นได้แก่สัญญาณ ATN, IFC และ REN สำรับสัญญาณ SQR นั้นจะถูกควบคุมโดยทางฝั่งของอุปกรณ์ที่มาเชื่อมต่อเท่านั้น และสำหรับสัญญาณ EOI นั้นเป็นสัญญาณที่จะถูกสั่งงานจากอุปกรณ์ใดก็ที่ทำหน้าที่เป็นตัวส่งในขณะนั้นๆ



รูปที่ 2.3 แสดงองค์ประกอบของสัญญาณในระบบ

- **ATN (Attention)** สัญญาณ ATN นี้นั้นเป็นสัญญาณที่จะถูกควบคุมโดยทางภาคควบคุมเท่านั้น ซึ่งเป็นสัญญาณที่บ่งบอกสถานะของสัญญาณที่อยู่บนบัสข้อมูลว่าเป็นสัญญาณประเภทใดหากสัญญาณ ATN อยู่ในสถานะ logic 1 (Output low) จะเป็นการบ่งบอกว่าข้อมูลที่อยู่บนบัสข้อมูลนี้นั้นเป็นรหัสคำสั่งมาตรฐานของ GPIB ซึ่งเป็นข้อมูลที่มีบิตเดียว แต่เมื่อสัญญาณ ATN อยู่ในสถานะ logic 0 (Output High) จะเป็นการบ่งบอกว่าข้อมูลบนบัสข้อมูลนั้น ๆ เป็นข้อมูลประเภทรหัส ASCII ซึ่งการส่งข้อมูลทั้งนี้จำเป็นต้องอาศัยสัญญาณ EOI ในการจัดชุดข้อมูลด้วย แต่อย่างไรก็ตามการส่งข้อมูลทั้งสองประเภทนี้นั้นจำเป็นที่จะต้องอาศัยการควบคุมส่งข้อมูลจากสัญญาณควบคุมการรับส่งข้อมูล (DAV, NRFD และ NDAC) ด้วย

- IFC (Interface Clear) สัญญาณนี้นั้น เป็นสัญญาณที่ใช้สั่งการให้อุปกรณ์ทั้งหมดที่ต่ออยู่ร่วมกับภาคควบคุมกลับไปอยู่ในสถานะเริ่มต้นอีกครั้ง (สถานะที่ไม่ได้มีการเชื่อมต่อ Address ไปยังอุปกรณ์ใด ๆ) ในทางปฏิบัตินั้นเมื่อมีการเริ่มการใช้งาน ควรมีการสั่งการให้สัญญาณ IFC ทำงานเพื่อกำหนดสถานะเริ่มต้นให้แก่อุปกรณ์ที่เชื่อมต่ออยู่ทั้งหมดด้วย

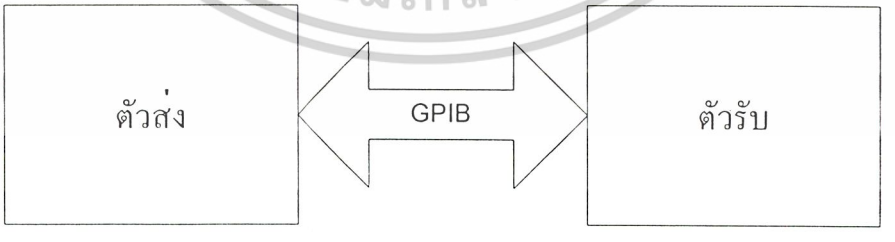
- REM (Remote enable) สัญญาณ REM นี้เป็นสัญญาณที่ใช้ควบคุมระบบการทำงานของอุปกรณ์ที่ต่อพ่วงอยู่ทั้งหมด โดยในขณะที่สัญญาณ REM นี้ถูกสั่งให้เป็น Logic 1 อุปกรณ์ที่ต่อพ่วงอยู่ทั้งหมดจะทำงานในโหมด Remote แต่หากสัญญาณ REM นี้ถูกกำหนดให้เป็น Logic 0 จะเป็นการกำหนดให้อุปกรณ์ที่ต่อพ่วงอยู่ทำงานในโหมด Local ตามปกติ

- SQR (Service Request) สัญญาณนี้เป็นสัญญาณจากอุปกรณ์จากอุปกรณ์ที่ต่อพ่วงกับตัวควบคุมเพื่อร้องขอการติดต่อจากตัวควบคุมซึ่งอาจเกิดข้อผิดพลาดในการสั่งการ จึงจำเป็นต้องร้องขอคำสั่งในการแก้จากตัวควบคุม

- EOI (End Of Data) เป็นสัญญาณที่ใช้ร่วมกับการส่งชุดข้อมูลคำสั่งที่ประเภท String โดยในสภาวะปกติสัญญาณนี้จะมี Logic 1 แต่เมื่อเริ่มมีการส่งข้อมูล String จากตัวส่งสัญญาณนี้จะถูกตัวส่งกำหนดให้เป็นเป็น Logic 0 และเริ่มส่งข้อมูลออกไปทีละตัว ซึ่งทางภาครับก็จะทำการเก็บข้อมูลนั้นไว้แต่ไม่นำไปสั่งการจะรอข้อมูลที่ตามมาในหลักต่อ ๆ ไป จนกระทั่งเมื่อสัญญาณ EOI ถูกเปลี่ยนให้กลับมาอยู่ในสถานะ Logic 1 อีกครั้ง จะเป็นการบ่งบอกว่าชุดข้อมูลที่ได้ทำการส่งนั้นจบลงแล้ว

2.4 ขบวนการแฮนด์เช็ค (Handshake Procedure)

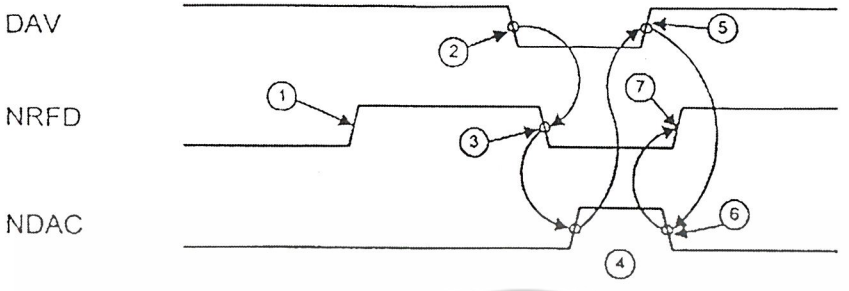
ในการสื่อสารระหว่างภายในระบบ GPIB นั้นจะเป็นการสื่อสารแบบอะซิงโครนัส คือเมื่อมีการรับส่งข้อมูลระหว่างตัวส่งและตัวรับ ตัวส่งจะต้องแจ้งให้ตัวรับทราบว่าตัวส่งได้ส่งข้อมูลลงไปบนบัสแล้ว และให้ตัวรับทำการเก็บข้อมูลได้ เมื่อตัวรับทำการเก็บข้อมูลเสร็จแล้ว ก็จะต้องแจ้งแก่ตัวส่งให้ทราบว่าได้รับข้อมูลที่ส่งมาเรียบร้อยแล้ว เพื่อที่ตัวส่งจะได้ทำการหยุดส่งข้อมูลในระบบซึ่งกระบวนการเหล่านี้ถูกเรียกว่า ขบวนการแฮนด์เช็ค (Handshake Procedure)



รูปที่ 2.4 แสดงส่วนประกอบของระบบ

ในการพิจารณาถึงขบวนการแฮนด์เช็คนั้น จะทำการพิจารณาถึงระบบที่ไม่ซับซ้อนนักเพื่อที่จะทำความเข้าใจได้โดยง่าย โดยกำหนดให้ในระบบมีตัวส่งและตัวรับอย่างละหนึ่งตัว ในการสื่อสารระหว่างตัวส่งและตัวรับนั้น จะมีสายสัญญาณควบคุมการรับส่งข้อมูลอยู่ 3 สัญญาณ คือ NRFD, NDAC, DAV โดยสัญญาณ DAV จะเป็นสัญญาณที่ถูกควบคุมโดยตัวส่ง ส่วนสัญญาณ NRFD, NDAC นั้นเป็นสัญญาณ

ที่ขณะชี้ให้เห็นว่าตัวรับพร้อมที่จะรับข้อมูลที่ส่งลงมาบนบัสของระบบหรือไม่ สำหรับขั้นตอนของขบวนการแฮนด์เช็คดังที่แสดงในรูปที่ 2.5



รูปที่ 2.5 แสดงแผนผังเวลาของขบวนการแฮนด์เช็ค

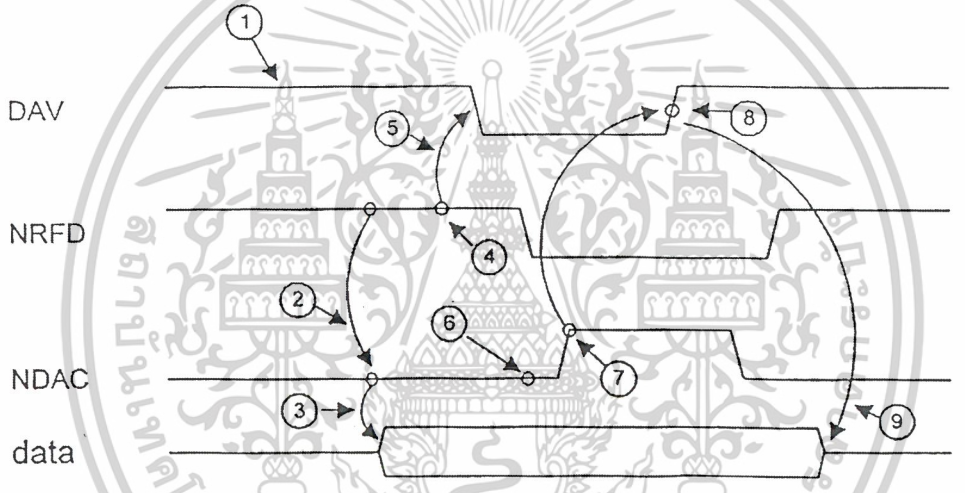
ขบวนการแฮนด์เช็คจะเริ่มขึ้นหลังจากที่ตัวควบคุมทำการบอกให้ระบบทราบว่าอุปกรณ์ตัวไหนทำหน้าที่เป็นตัวรับหรือตัวส่ง เมื่อตัวรับทราบแล้วก็จะส่งสัญญาณ NRFD ให้เป็น High (สถานะที่ 1) เพื่อบอกให้ตัวส่งทราบว่าตัวรับพร้อมที่จะรับข้อมูลแล้ว และตัวส่งก็จะทำการส่งข้อมูลลงไปบนบัสข้อมูล DIO1-DIO8 และจะทำการรออยู่ชั่วขณะหนึ่งแล้วส่งสัญญาณ DAV ให้เป็น Low เพื่อแจ้งให้ทราบว่าขณะนี้ตัวส่งได้ข้อมูลลงบนบัสข้อมูลแล้ว (สถานะที่ 2) เมื่อตัวรับทราบว่ามีข้อมูลอยู่บนบัสข้อมูลก็จะส่งสัญญาณ NRFD ให้ค่าเป็น Low เมื่อตัวรับพร้อมที่จะรับข้อมูล (สถานะที่3) หลังจากตัวรับได้รับข้อมูลไปเก็บไว้ในบัฟเฟอร์เรียบร้อยแล้วก็จะส่งสัญญาณ NDAV ให้ค่าเป็น High เพื่อแจ้งให้ทราบว่าตัวรับได้รับข้อมูลเรียบร้อยแล้ว (สถานะที่ 4) เมื่อตัวส่งได้รับสัญญาณ NDAC ให้เป็น High ตัวส่งก็จะทำการส่งสัญญาณ DAV ให้เป็น High เพื่อแจ้งให้ตัวรับไม่ต้องทำการเก็บข้อมูลนั้นอีก (สถานะที่ 5) เมื่อตัวรับได้รับสัญญาณ DAV ที่มีค่าเป็น High ก็จะส่งสัญญาณ NDAC ให้เป็น Low (สถานที่ 6) ทำให้ข้อมูลในบัสถูกออกไป หลังจากนั้นตัวรับก็จะส่งสัญญาณ NRFD ให้เป็น High (สถานะที่ 6) ทำให้ข้อมูลในบัสถูกกำจัดออกไป หลังจากนั้นตัวรับก็จะส่งสัญญาณ NRFD ให้เป็น High (สถานะที่ 7) เพื่อบอกให้ทราบว่าตัวรับนั้นพร้อมที่จะรับข้อมูลชุดต่อไปที่จะถูกส่งเข้ามาในบัส เป็นอันเสร็จสิ้นขบวนการแฮนด์เช็ค

- ขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวส่ง

ขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวส่งสามารถแสดงด้วยแผนผังเวลา ซึ่งจะทำให้ง่ายต่อการเข้าใจในขั้นตอนการส่งข้อมูล โดยขบวนการดังกล่าวจะเกิดขึ้นหลังจากการกำหนดอุปกรณ์ในระบบแล้ว

ขบวนการแฮนด์เช็คจะเริ่มขึ้น เมื่อตัวควบคุมส่งสัญญาณ DAV จะให้เป็น High (สถานะที่ 1) ซึ่งตัวควบคุมได้เซทให้สัญญาณ DAV ให้มีค่าเป็น High อยู่ก่อนแล้ว หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ NRFD และ NDAC ว่ามีค่าเป็น High ทั้งคู่หรือไม่ (สภาวะที่ 2) ถ้าสัญญาณทั้งสองเป็น High ทั้งคู่ แสดงว่าอุปกรณ์ไม่พร้อมที่จะทำงาน ขบวนการแฮนด์เช็คก็จะถูกยกเลิกไป แต่ถ้าสถานะที่ 2

หากสัญญาณใดสัญญาณหนึ่งเป็น Low ตัวควบคุมจะทำการส่งข้อมูลลงในบัสข้อมูล(สถานะที่ 3) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ NRFD ว่าเป็น High หรือไม่ (สถานะที่ 4) ถ้าสัญญาณ NRFD เป็น High ตัวควบคุมก็จะส่งสัญญาณ DAV ให้มีค่าเป็น Low เพื่อบอกให้ตัวรับทราบว่ามีข้อมูลอยู่ในบัสข้อมูล (สถานะที่ 5) แต่ถ้าที่สถานะที่ 4 สัญญาณ NRFD มีค่าเป็น Low แสดงว่าขบวนการแฮนด์เช็คเกิดความผิดพลาดขึ้น จะต้องการทำการเริ่มต้นใหม่ จากสถานะที่ 5 ตัวควบคุมจะรอเวลาให้ตัวรับทำการเก็บข้อมูลเมื่อถึงเวลาที่กำหนดตัวควบคุมจะทำการตรวจสอบสัญญาณว่าสัญญาณ NDAC ถูกเปลี่ยนให้เป็น High ในเวลาที่กำหนดหรือไม่ (สถานะที่ 6) ถ้าตัวรับไม่ได้รับข้อมูลในเวลาที่กำหนดขบวนการแฮนด์เช็คจะถูกยกเลิก แต่ถ้าตัวรับได้รับข้อมูลภายในเวลาที่กำหนดตัวรับจะทำการเปลี่ยนสัญญาณ NDAV ให้เป็น High (สถานะที่ 7) ตัวควบคุมก็จะทำการตอบสนองโดยการเปลี่ยนสัญญาณ DAV ให้เป็น High (สถานะที่ 8) และตัวควบคุมก็จะทำการลบข้อมูลที่อยู่ในบัสข้อมูลออกไป(สถานะที่ 9) เป็นการเสร็จสิ้นขบวนการแฮนด์เช็คดังกล่าว

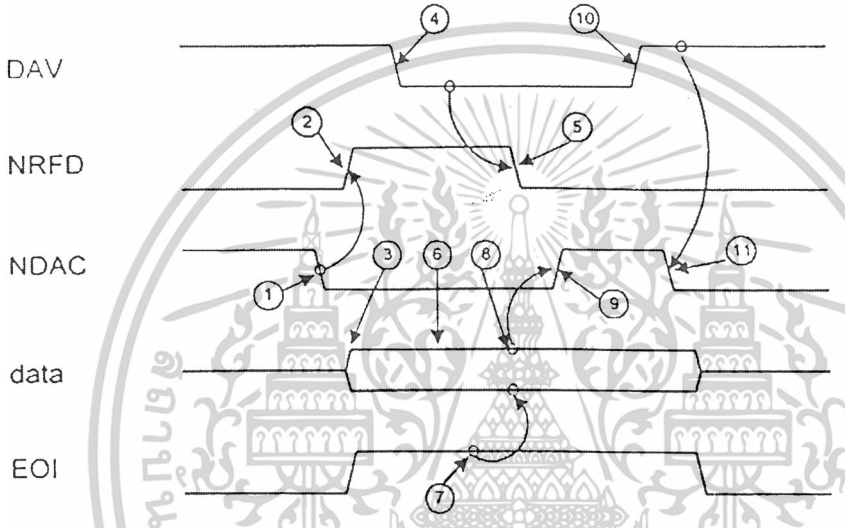


รูปที่ 2.6 แสดงขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวส่ง

- ขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวรับ

ขบวนการแฮนด์เช็คเริ่มขึ้นโดยตัวควบคุมรับรู้ว่าตัวส่งจะทำการส่งข้อมูล ตัวควบคุมจะส่งสัญญาณ NDAV ให้มีค่าเป็น Low เพื่อบอกให้ตัวควบคุมยังไม่ได้รับข้อมูล (สถานะที่ 1) ต่อจากนั้นตัวควบคุมจะส่งสัญญาณ NRFD ให้เป็น High เพื่อบอกให้ตัวทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลแล้ว (สถานะที่ 2) ตัวส่งจะทราบได้ทันทีว่าขณะนี้สามารถที่จะส่งข้อมูลลงบนบัสข้อมูลได้แล้ว (สถานะที่ 3) จากการที่สัญญาณ NRFD มีลอจิกเป็น High และสัญญาณ NDAC มีลอจิกเป็น Low ขั้นต่อไปตัวควบคุมจะทำการรอเวลาให้ตัวส่งทำการส่งข้อมูลลงบนบัสข้อมูลให้เสร็จและจะทำการตรวจสอบด้วยว่าเกินเวลาที่กำหนดหรือไม่ หากเกินเวลาที่กำหนดก็จะทำการออกจากขบวนการแฮนด์เช็ค หากยังไม่เกินก็จะรอจนหมดเวลาหรือจนกว่าสัญญาณ DAV จะเป็น Low (สถานะที่ 4) เมื่อสัญญาณ DAV มีลอจิกเป็น Low ตัวควบคุมก็จะตอบรับโดยการทำให้สัญญาณ NRFD มีค่าเป็น Low (สถานะที่ 5) เพื่อบอกให้ตัวส่งทราบว่า

ตัวควบคุมพร้อมที่จะเริ่มทำการเก็บข้อมูลแล้ว (สถานะที่ 6) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ EOI ให้มีลอจิกเป็น Low เมื่อข้อมูลไบต์สุดท้ายถูกส่งลงไปบนบัสข้อมูล (สถานะที่ 7) หากสัญญาณ EOI ยังเป็น High ตัวควบคุมจะการเก็บข้อมูลในบัสข้อมูลต่อไป (สถานที่ 8) และเมื่อตัวควบคุมได้ทำการเก็บข้อมูลเสร็จเรียบร้อยแล้ว ตัวควบคุมจะเปลี่ยนลอจิกของสัญญาณ NDAC ให้เป็น High (สถานะที่ 9) เมื่อตัวควบคุมทำให้สัญญาณ DAV ให้มีค่าเป็น High (สถานะที่ 10) ตัวควบคุมก็จะทำการนำข้อมูลที่ได้ไปใช้งานและเปลี่ยนสัญญาณ NDAC ให้มีค่าเป็น Low (สถานะที่ 11) เป็นการสิ้นสุดขบวนการแฮนด์เช็ค ขบวนการแฮนด์เช็คแบบนี้สามารถเขียนแทนด้วยแผนผังเวลาดังรูปที่ 2.7



รูปที่ 2.7 แสดงขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวรับ

2.5 คำสั่งใช้งานของ IEEE-488(GPIB)

การควบคุมและกำหนดฟังก์ชั่นการทำงานให้แก่อุปกรณ์เครื่องมือวัดในระบบ GPIB นั้นตัวควบคุมจะเป็นตัวกำหนด โดยการส่งรหัสคำสั่งไปยังตัวอุปกรณ์โดยผ่านทางบัสของระบบ คำสั่งสำหรับการกำหนดการทำงานต่างๆ ตามมาตรฐานของ IEEE-488(GPIB) มีอยู่ 128 คำสั่ง แบ่งเป็น 2 กลุ่มคำสั่งใหญ่ ๆ คือ กลุ่มคำสั่งหลัก (Primary Command Group) และกลุ่มคำสั่งรอง (Secondary Command Group) โดยกลุ่มคำสั่งหลักประกอบด้วย 4 กลุ่มคำสั่งคือ กลุ่มคำสั่งเจาะจงจุดหมาย (Address Command Group), กลุ่มคำสั่งครอบคลุม (Universal Command Group), กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (Listen Address Group) และกลุ่มคำสั่งกำหนดอุปกรณ์ตัวส่ง (Talk Addressed Group) รหัสที่ใช้ระบบ IEEE-488 นั้นสามารถที่จะใช้ร่วมกันได้ทั้งรหัสข้อมูลและรหัสคำสั่ง นั่นคือ ข้อมูลที่เหมือนกันมีความหมายได้ 2 อย่าง คือเมื่อสัญญาณ ATN เป็น Low ข้อมูลที่อยู่ในบัสข้อมูลจะหมายถึงรหัสคำสั่ง แต่ถ้าสัญญาณ ATN เป็น High ข้อมูลที่อยู่ในบัสข้อมูลจะหมายถึงข้อมูลที่เป็นรหัส ASCII ดังแสดงในตารางที่ 1 เช่นกัน

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ตารางที่ 2.1 รหัส ASCII

1. กลุ่มคำสั่งเจาะจงจุดมุ่งหมาย (Address Command Group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ที่ตัวส่งหรือตัวรับที่กำหนดไว้ล่วงหน้าแล้ว ประกอบด้วย

GTL (Go To Local) สั่งให้อุปกรณ์กลับไปสู่สถานะควบคุมด้วยปุ่มปรับที่หน้าปัดตามปกติ

SDC (Selected Device Clear) สั่งให้อุปกรณ์กลับไปสู่สถานะเริ่มต้น

PPC (Parallel Poll Configure) เป็นคำสั่งสำหรับทำการจัดสรรสายสัญญาณของการกระทำกระบวนการตรวจสอบสภาพอุปกรณ์โดยวิธีขนาน โดยใช้งานร่วมกับกลุ่มคำสั่งรอง

GET (Group Execute Trigger) ใช้สั่งเริ่มต้นการทำงานของอุปกรณ์ที่ละหลาย ๆ ตัว

TCT (Take Control) กำหนดให้อุปกรณ์ตัวส่งทำหน้าที่เป็นตัวควบคุม

2. กลุ่มคำสั่งครอบคลุม (Universal Command Group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ทุกตัวที่อยู่ในระบบประกอบด้วย

LLO (Local Lockout) สั่งให้อุปกรณ์ล็อคอยู่ในสถานะควบคุมด้วยปุ่มหน้าปัดตามปกติ

DCL (Device Clear) สั่งให้อุปกรณ์ทุกตัวกลับไปสู่สถานะเริ่มต้น

PPU (Parallel Poll Configure) ใช้ยกเลิกกระบวนการตรวจสอบสภาพแบบขนานทั้งหมด

SPE (Serial Poll Enable) เป็นการเปลี่ยนโหมดการตรวจสอบสภาพเป็นแบบอนุกรม โดยในโหมดนี้จะเป็นการส่งสถานะของเครื่องแทนการส่งข้อมูล

SPD (Serial Poll Disable) ยกเลิกกระบวนการตรวจสอบสภาพแบบอนุกรม

3. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (Listen Address Group) เป็นคำสั่งสำหรับกำหนดอุปกรณ์เป็นตัวรับตามรหัสหมายเลข 0 ถึง 30 และมีคำสั่ง UNL (Unlistener) สำหรับใช้ยกเลิกเช่นกัน

4. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวส่ง (Talk Addressed Group) ใช้คำสั่งที่กำหนดอุปกรณ์เป็นตัวส่งตามรหัสหมายเลข 0 ถึง 30 และมีคำสั่ง UNT (Untalker) ใช้สำหรับยกเลิกเช่นกัน

5. กลุ่มคำสั่งรอง (Secondary Command Group) เป็นคำสั่งที่กำหนดรายละเอียดย่อยของอุปกรณ์แต่ละตัวที่ต่ออยู่ในระบบ ให้มีการทำงานตามจุดประสงค์การใช้งานของอุปกรณ์นั้น คำสั่งรองนี้จะใช้ตามหลังคำสั่งหลัก คือ จะใช้หลังจากที่อุปกรณ์ต่าง ๆ ถูกกำหนดไว้ในระบบเรียบร้อยแล้ว

คำสั่งต่าง ๆ ซึ่งใช้ในการกำหนดสภาวะการทำงานของอุปกรณ์แต่ละสภาวะจะถูกกำหนดและมีจุดประสงค์ดังนี้ คือ

Device Clear ทำให้อุปกรณ์กลับคืนสู่สภาวะเริ่มต้น ซึ่งเป็นสภาวะที่ยังไม่มีการกำหนดฟังก์ชันใด ๆ สภาวะเริ่มต้นนี้แตกต่างกันไป แล้วแต่ว่าอุปกรณ์นั้นได้ออกแบบมาไว้อย่างไร Device Clear แบ่งออกได้เป็น 2 ลักษณะ คือ

DCL (Device Clear) ทำการเคลียร์อุปกรณ์ทุกตัวที่ต่ออยู่

SDC (Select Device Clear) ทำการเคลียร์เฉพาะเจาะจงอุปกรณ์ตัวใดตัวหนึ่ง

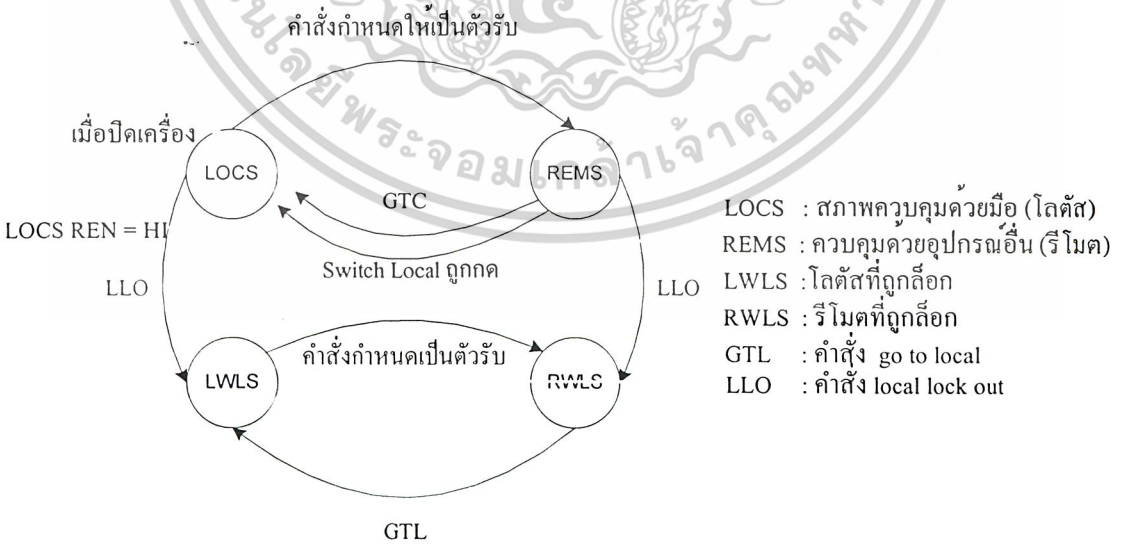
แต่ว่าในการเคลียร์อุปกรณ์ให้อยู่ในสภาวะเริ่มต้นไม่ได้หมายความว่าอินเตอร์เฟสฟังก์ชันของระบบ GPIB จะถูกเคลียร์ให้กลับไปสู่สภาวะเริ่มต้นด้วย เป็นเพียงแค่การเคลียร์ตัวอุปกรณ์เท่านั้น

Interface Clear จะใช้ในการเคลียร์สภาพการอินเตอร์เฟสให้อยู่ในสภาวะเริ่มต้น ซึ่งจะทำให้ทุกฟังก์ชันถูกยกเลิกไป ยกเว้น SR (Service Request), RL (Remote/Local) และ PP (Parallel Poll)

Remote เป็นการกำหนดให้อุปกรณ์ที่ต่ออยู่ในระบบถูกควบคุมโดยอุปกรณ์ตัวอื่นหรือตัวควบคุมระบบ ทำให้ไม่สามารถที่จะควบคุมอุปกรณ์จากปุ่มหน้าปัดของเครื่องได้

Local เป็นการกำหนดให้อุปกรณ์ที่ต่ออยู่ในระบบสามารถควบคุมได้จากปุ่มหน้าปัดของอุปกรณ์ตามปกติ

- การทำงานของ GPIB ในสถานะ Remote และLocal มี 4 ลักษณะ คือ



รูปที่ 2.8 แสดงการเปลี่ยนแปลงสภาวะของโหมด Remote และ Local ทั้ง 4 ลักษณะ

1. LOCS เป็นโหมด local ที่อยู่ในสภาพการควบคุมจากหน้าปัดตามปกติซึ่งอุปกรณ์จะอยู่ในสภาวะนี้เมื่อเริ่มเปิดสวิทซ์ของอุปกรณ์หรือสัญญาณ REN มีลอจิกเป็น High หรือเมื่ออุปกรณ์ได้รับคำสั่ง GTL (Go to Local)

2. REMS เป็นโหมด Remote หมายถึงการตัดการควบคุมอุปกรณ์ออกจากการควบคุมด้วยปุ่มหน้าปัด โดยถูกควบคุมจากอุปกรณ์ตัวอื่นที่ทำหน้าที่เป็นตัวควบคุม สภาวะการ Remote จะเกิดขึ้นเมื่อสัญญาณ REN (Remote Enable) มีลอจิกเป็น low และจะถูกล็อคไว้ที่สถานะนี้จนกว่าสวิทช์ Local ที่ตัวอุปกรณ์จะถูกกด เพื่อที่จะทำให้อุปกรณ์กลับสู่สถานะ Local

3. RWLS เป็นสภาวะการ Remote ที่ถูกล็อคไว้เช่นกัน แต่จะตัดการควบคุมของสวิทช์ Local ที่ตัวอุปกรณ์ออกไป สภาวะการ Remote แบบ RWLS นี้มีความสำคัญสูงกว่าสภาวะการ Remote แบบ REMS และสามารถที่จะยกเลิกสภาวะดังกล่าวนี้ด้วยคำสั่ง LLO (Local Lock Out)

4. LWLS มีสภาวะเช่นเดียวกับ Local แต่ต่างกันที่สภาวะ LWLS นี้ แบบที่ถูกล็อคทันที การจะเข้าสู่สภาวะแบบ LWLS มีอยู่ 2 วิธี คือ

- เมื่ออยู่ในสภาวะ Local แบบธรรมดา (LOCS) แล้วได้รับคำสั่ง LLO (Local Lock Out)
- เมื่ออยู่ในสภาวะ REMS แล้วรับคำสั่ง GTL

การเปลี่ยนแปลงสภาวะของการ Remote หรือ Local ทั้ง 4 ลักษณะ เมื่อสัญญาณ REN มีลอจิกเป็น High อุปกรณ์ก็จะอยู่ในสภาวะ Local ทันทีไม่ว่าสถานะเดิมจะเป็นอย่างไรก็ตาม แต่เมื่อสัญญาณ REN เป็น Low แล้ว หากไม่มีคำสั่งกำหนดอุปกรณ์ตัวรับหรือคำสั่ง LLO เข้ามาอุปกรณ์ก็ยังมีสถานะ Local

บทที่ 3
คุณสมบัติโดยทั่วไปของระบบบัส USB

3.1 คุณสมบัติเด่นของระบบบัส USB

ระบบบัส USB (Universal Serial Bus) นั้นเป็นระบบที่มีความยืดหยุ่นสูง ปราศจากข้อกำจัดและการขัดขวางของการอินเตอร์เฟสทางด้านฮาร์ดแวร์ ซึ่งการตั้งค่าการทำงานต่างๆ จะถูกกระทำโดยระบบปฏิบัติการโดยอัตโนมัติ นอกจากนี้การเชื่อมต่ออุปกรณ์ USB เข้ากับระบบนั้นสามารถทำได้ทั้งในขณะที่เครื่องคอมพิวเตอร์ยังคงทำงานอยู่ซึ่งกล่าวได้ว่าเป็นรูปแบบของปลั๊กแอนด์เพลย์ (Plug and Play) อย่างแท้จริง อีกทั้งความเร็วในการส่งถ่ายข้อมูลในUSB 2.0 นั้นก็มีความเร็วสูงกว่าการส่งถ่ายข้อมูลแบบขนานและอนุกรมแต่เดิมเป็นอย่างมาก และในการเพิ่มจำนวนของพอร์ตนั้นก็สามารถทำได้โดยง่ายเพียงนำ USB Hub มาต่อพ่วงเข้ากับระบบเท่านั้น

คุณสมบัติเด่นในระบบบัส USB ได้แก่

- สามารถนำอุปกรณ์ I/O มาต่อพ่วงเข้าสู่ระบบได้ในขณะที่เครื่องคอมพิวเตอร์ยังคงทำงานอยู่ได้ (Hot-Pluggable)
- ง่ายต่อการใช้งาน เนื่องจากเครื่องคอมพิวเตอร์จะมีความสามารถในการทำความรู้จักและจดจำอุปกรณ์ต่างๆ ที่นำมาต่อพ่วงในระบบโดยไดร์ฟเวอร์ที่เหมาะสมและในการตั้งค่าต่างๆ ก็เป็นไปโดยอัตโนมัติ
- ใช้คอนเน็คเตอร์ในการเชื่อมต่อเพียงชนิดเดียวจึงลดความสับสนในการเชื่อมต่อ
- มีประสิทธิภาพในการส่งถ่ายข้อมูลสูง โดยเป็นไปตามมาตรฐานดังนี้
  - ⟹ มาตรฐาน USB 1.0/1.1 มีระดับความเร็วในการส่งถ่ายข้อมูล 2 ระดับได้แก่ที่ระดับความเร็วต่ำ (Low Speed) เท่ากับ 1.5 Mbit/Sec และที่ระดับความเร็วเต็มที่ (Full Speed) เท่ากับ 12 Mbit/Sec
  - ⟹ มาตรฐาน USB 2.0 จะมีการเพิ่มเติมระดับความเร็วในการส่งถ่ายขึ้นอีก 1 ระดับได้แก่ ที่ระดับความสูง (High Speed) มีความเร็วในการส่งถ่ายข้อมูลเท่ากับ 480 Mbit/Sec
- ไม่เกิดการขัดแย้งกันของการเข้าใช้ทรัพยากรของระบบ (IRQ : Interrupt Request) ซึ่งเป็นการแก้ปัญหาทางด้านข้อจำกัดของจำนวนอุปกรณ์ที่มาเชื่อมต่อ
- สามารถต่ออุปกรณ์ภายในระบบได้สูงสุด 127 ตัว
- สายเคเบิลของ USB นั้น จะมีสายของแหล่งจ่ายกำลังงานรวมอยู่ภายในและสามารถนำมาใช้งานได้
- มีการจัดการกับระบบพลังงานที่ชาญฉลาด ซึ่งกำลังงานบนบัสจะถูกลดระดับลงเมื่อไม่ได้มีการใช้งานเป็นระยะหนึ่ง
- มีความสามารถในการตรวจสอบและแก้ไขข้อผิดพลาดของข้อมูลโดยอัตโนมัติ

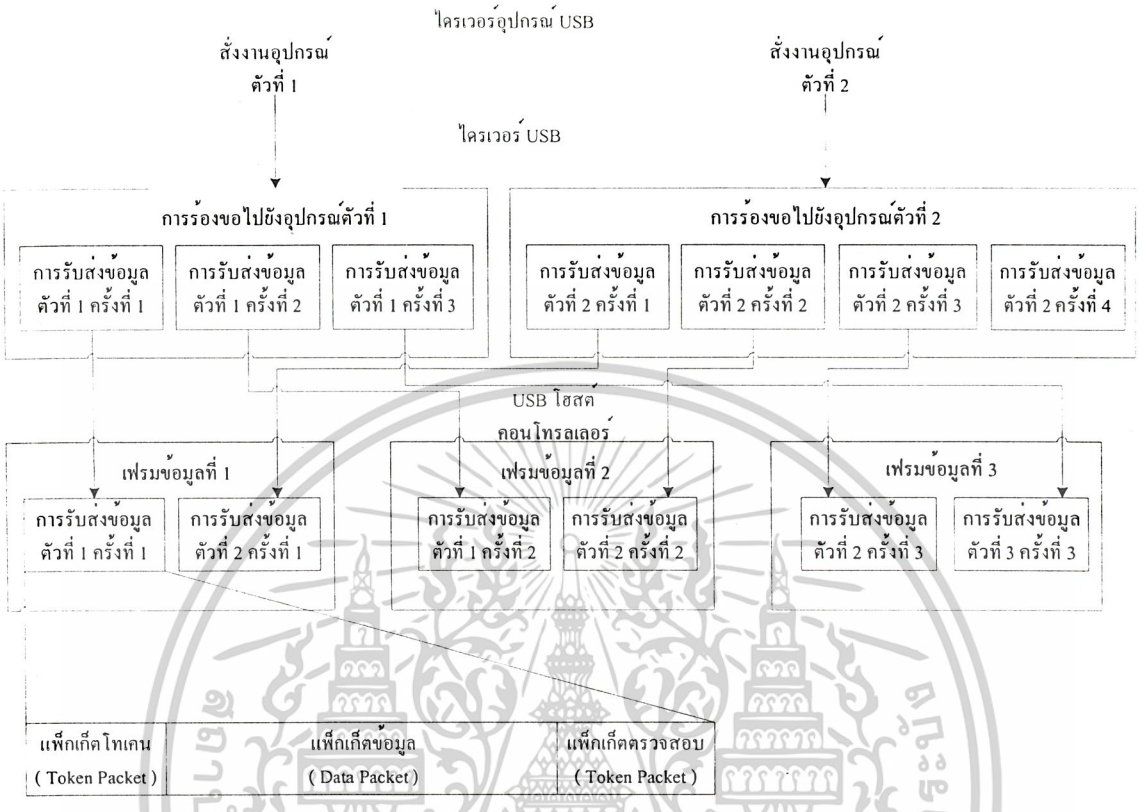
3.2 การส่งถ่ายข้อมูลบนระบบบัส USB

USB เป็นการสื่อสารข้อมูลในรูปแบบอนุกรมรูปแบบหนึ่งซึ่งอุปกรณ์ทุกๆ ตัวจะต้องส่งสัญญาณรวมกันไปในสายส่งสัญญาณเพียงคู่เดียว ดังนั้นอุปกรณ์ทุก ๆ ตัวที่เชื่อมต่อกับบัสจะต้องส่งข้อมูลเรียงลำดับกันไปเพื่อไม่ให้เกิดการชนกันของข้อมูลและเนื่องจาก USB เป็นระบบบัสที่ใช้สายส่งสัญญาณเพียงคู่เดียวทำให้ในช่วงเวลาหนึ่งๆ จะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียวเท่านั้น ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันไปหรือที่เรียกว่า การส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ (half duplex) โดยจังหวะการรับส่งข้อมูลของระบบบัส USB ทั้งหมดจะถูกควบคุมจากโฮสต์ (host) ซึ่งก็คือเครื่องคอมพิวเตอร์ที่เป็นจุดของอุปกรณ์ทุกตัวที่เชื่อมต่ออยู่นั่นเอง ดังนั้นจึงไม่สามารถเชื่อมต่ออยู่นั่นเอง ดังนั้นจึงไม่สามารถเชื่อมต่อเครื่องคอมพิวเตอร์ 2 เครื่อง ให้รับหรือส่งข้อมูลถึงกันได้โดยตรง เพราะถ้าคอมพิวเตอร์ทั้งสองเครื่องทำหน้าที่เป็นโฮสต์ทั้งคู่จะเกิดการชนกันของข้อมูลภายในบัส เนื่องจากแต่ละเครื่องก็จะพยายามกำหนดจังหวะในการรับส่งของตัวเองขึ้นมา ดังนั้นจะเชื่อมต่อคอมพิวเตอร์ 2 เครื่องเข้าด้วยกันผ่าน USB จะต้องมีอุปกรณ์ที่เป็นตัวกลางเพื่อซิงโครไนซ์ตัวเองเข้ากันโฮสต์ทั้งสองให้ได้

รูปที่ 3.1 แสดงรูปแบบการจัดข้อมูลการส่งข้อมูลบนบัส

การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรม โดยทุกๆ 1 มิลลิวินาที (ms) จะเกิดการรับส่งข้อมูลขึ้น 1 เฟรม ในแต่ละเฟรมจะแบ่งย่อยออกเป็นแพ็กเก็ต (packet) เริ่มต้นการทำงานของแต่ละเฟรมโดยโฮสต์จะต้องส่งสัญญาณเริ่มต้นเฟรมหรือ SOF (Start Of Frame) ออกไปเพื่อให้อุปกรณ์ทุกตัวรู้จังหวะการเริ่มเฟรม หลังจากนั้นโฮสต์ก็จะเริ่มส่งหรือรับข้อมูลต่างๆ ตามที่ได้จัดลำดับความสำเร็จไว้อุปกรณ์ต่างๆ ที่อยู่ภายในบัสจะต้องทำงานตามจังหวะที่โฮสต์กำหนดไว้เท่านั้น การส่งข้อมูลกลับไปยังโฮสต์จะสามารถทำได้ก็ต่อเมื่อได้รับการถามหรือร้องขอจากโฮสต์ แต่เนื่องจากแต่ละเฟรมข้อมูลกลับไปยังโฮสต์จะสามารถทำได้ก็ต่อเมื่อได้รับการถามหรือร้องขอจากโฮสต์ แต่เนื่องจากแต่ละเฟรมข้อมูลจะต้องรับส่งเสร็จภายใน 1 มิลลิวินาที นั่นหมายความว่าข้อมูลของอุปกรณ์ทุกๆ ตัวที่เชื่อมตัวต่อกับบัสจะต้องถูกกำหนดขนาดไม่ให้ใหญ่เกินกว่าที่จะสามารถรับส่งภายใน 1 มิลลิวินาที และเล็กพอที่จะทำให้อุปกรณ์ทุกๆ ตัวสามารถใช้งานบัสไปพร้อม ๆ กันได้ ดังนั้นในระบบบัส USB จึงจำเป็นต้องอาศัยซอร์ฟแวร์ที่เข้ามาจัดการในด้านนี้ และยังต้องอาศัยฮาร์ดแวร์ที่จะคอยกระจายการส่งและรวบรวมการรับข้อมูลจากอุปกรณ์ทุกๆ ตัวในระบบโดยแบ่งเป็นองค์ประกอบทางด้านซอร์ฟแวร์และฮาร์ดแวร์ดังต่อไปนี้

3.3 องค์ประกอบทางด้านซอฟท์แวร์



รูปที่ 3.2 แสดงองค์ประกอบและการทำงานทางด้านซอร์ฟแวร์

3.3.1 ไดรเวอร์อุปกรณ์ USB (USB Device Drivers)

ไดรเวอร์อุปกรณ์ USB คือโปรแกรมเก็บข้อมูลที่จำเป็นในการติดต่อไปยังอุปกรณ์แต่ละตัวเมื่อโปรแกรมใดมีความต้องการจะติดต่อกับอุปกรณ์ต่างๆ จะต้องแจ้งความต้องการนั้นๆ มายังไดรเวอร์อุปกรณ์ USB เนื่องจากตัวไดรเวอร์นี้จะรู้ว่าถ้าต้องการติดต่อกับอุปกรณ์จะต้องติดต่อผ่านเอ็นด์พอยต์ (Endpoint) ไหน ด้วยรูปแบบใด ดั้งนั้นอุปกรณ์แต่ละตัวก็จะมีไดรเวอร์อุปกรณ์ USB เฉพาะตัว ซึ่งเมื่อถึงคราวต้องนำอุปกรณ์นั้นมาต่อใช้งานกับเครื่องคอมพิวเตอร์จริง ๆ ก็จะต้องนำไดรเวอร์ตัวเดียวกันมาติดตั้งเข้ามาใหม่นี้ได้ เช่น ถ้าต้องการติดต่อเพื่อรับข้อมูลจากคีย์บอร์ด ตัวไดรเวอร์อุปกรณ์ USB จะรู้ว่าต้องการรับส่งข้อมูลด้วยอัตราเร็วต่ำ (slow speed) โดยใช้รูปแบบการถ่ายทอดข้อมูลแบบอินเตอร์รัปต์ (interrupt transfer type) ผ่านเอ็นพอยต์ตัวหนึ่งของคีย์บอร์ด และตรวจสอบข้อมูลการกดเป็นช่วงระยะห่างค่าหนึ่ง แต่ในบางอุปกรณ์ที่เป็นอุปกรณ์พื้นฐานของเครื่องคอมพิวเตอร์ เช่นเมาส์และคีย์บอร์ดจะมีการบรรจุไดรเวอร์อุปกรณ์ USB ของอุปกรณ์เหล่านี้ไว้ภายในไบออสของเครื่องคอมพิวเตอร์เรียบร้อยแล้ว จึงไม่ต้องติดไดรเวอร์เพิ่มเติมสำหรับอุปกรณ์เหล่านี้ เพียงแต่เข้าไปเปิดการทำงาน ไบออสก็จะทำให้เครื่องคอมพิวเตอร์รู้จักอุปกรณ์เหล่านี้โดยอัตโนมัติ

3.3.2 ไดรเวอร์อุปกรณ์ USB (USB Drivers)

ไดรเวอร์อุปกรณ์ USB นั้นเป็นส่วนของซอร์ฟแวร์ที่ทำหน้าที่ในการจัดการแบ่งสันปันส่วนช่องสัญญาณในแต่ละเฟรม ใช้อุปกรณ์แต่ละตัวนั้นสามารถส่งข้อมูลร่วมกันไปในสายสัญญาณคู่เดียวได้ในเวลาเดียวกัน ไดรเวอร์อุปกรณ์ USB ของอุปกรณ์แต่ละตัวจะส่งการร้องขอเพื่อการติดต่อ (request) ลงมาไดรเวอร์ USB และเมื่อไดรเวอร์ USB รับทราบความต้องการติดต่อของอุปกรณ์ครบทุกๆ ตัวที่เชื่อมต่ออยู่กับบัสแล้ว ก็จะพิจารณาว่าในรอบการรับส่งข้อมูลหนึ่งๆ นั้นอุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้มากเท่าใด หากปริมาณข้อมูลที่ส่งได้ของอุปกรณ์แต่ละตัวจะถูกพิจารณาจากชนิดของการถ่ายทอดข้อมูล (Transfer type) ว่า อุปกรณ์ใดใช้การถ่ายทอดข้อมูลแบบใดและการรับส่งข้อมูลชนิดนั้นมีลำดับความสำคัญมากน้อยเพียงใด

3.3.3 ไดร์เวอร์โฮสต์คอนโทรลเลอร์ (USB Host Controller Driver)

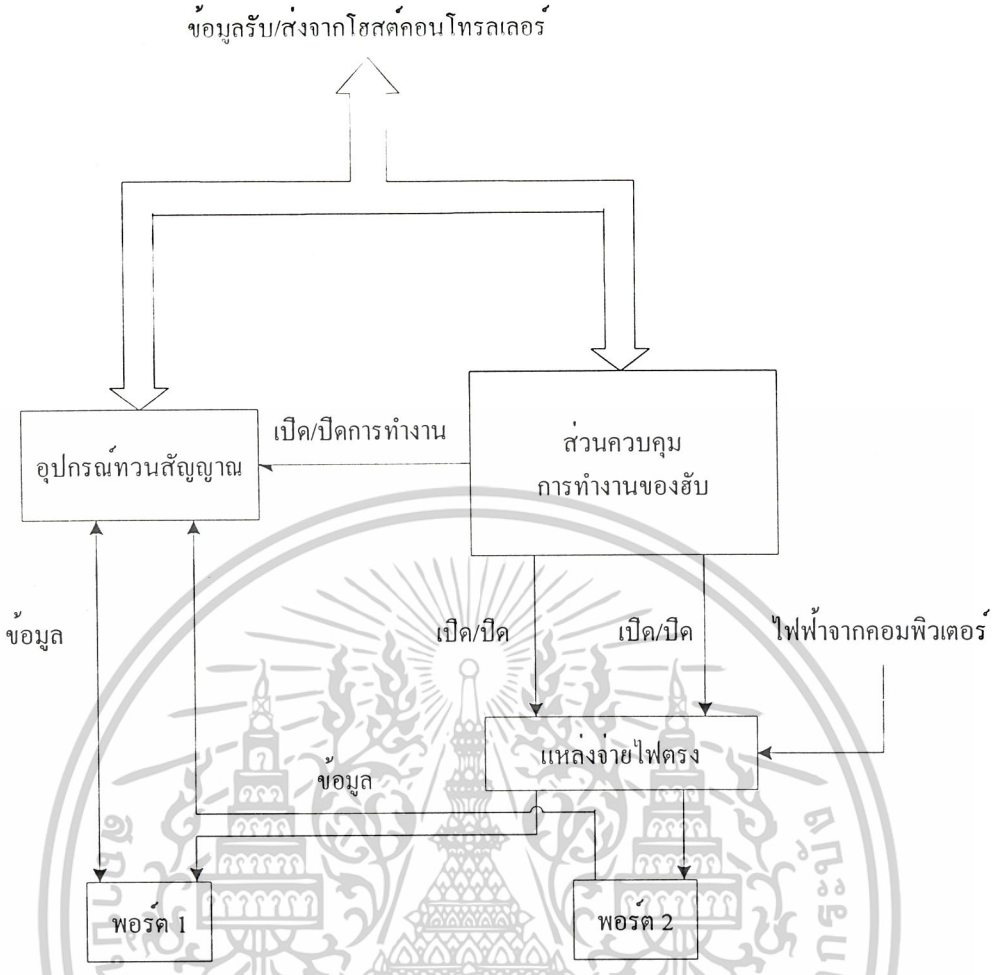
หลังจากไดรเวอร์ USB พิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้างมันจะส่งข้อมูลของอุปกรณ์แต่ละตัวที่จะติดต่อในรอบการติดต่อนั้นๆ มายังไดรเวอร์โฮลต์คอนโทรลเลอร์จากนั้นไดรเวอร์โฮสต์คอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลของอุปกรณ์แต่ละชนิดลงเป็นเฟรมข้อมูล เพิ่มเติมส่วนประกอบต่างๆ ของเฟรมข้อมูลให้ครบตามมาตรฐานการถ่ายทอดข้อมูลแบบ USB แล้วส่งข้อมูลทั้งหมดไปยังฮาร์ดแวร์ USB โฮสต์คอนโทรลเลอร์เพื่อส่งข้อมูลทั้งหมดออกไปยังอุปกรณ์ต่างๆ ในรูปที่ 3.2 แสดงลำดับและขั้นตอนการทำงานของซอร์ฟแวร์ควบคุมการทำงานของพอร์ต USB

3.4 องค์ประกอบทางด้านฮาร์ดแวร์

3.4.1 USB โฮสต์คอนโทรลเลอร์ & USB รูตฮับ (USB Host Controller Diver & Root Hub)

ในส่วนของ USB โฮสต์คอนโทรลเลอร์นั้นมีหน้าที่ในการสร้างสัญญาณไฟฟ้าเพื่อใช้ในการติดต่อสื่อสาร โดยสัญญาณที่ได้จากตัวมันจะถูกส่งไปยัง USB รูตฮับ โดยมันจะทำหน้าที่ในการแปลงข้อมูลรูปแบบขนานที่รับมาจาก USB โฮสต์คอนโทรลเลอร์ ให้เป็นข้อมูลรูปแบบอนุกรมที่ใช้ในการส่งต่อไป นอกเหนือจากนี้นั้น USB รูตฮับยังมีหน้าที่สำคัญอีก 4 ประการได้แก่

- ควบคุมการใช้พลังงานของอุปกรณ์ที่มาต่อร่วม
- ตรวจสอบการเชื่อมต่อของอุปกรณ์ที่ต่อร่วมอยู่หรือไม่
- เปิดหรือเอ็นเอเบิลการใช้งานพอร์ตเมื่อมีอุปกรณ์ต่ออยู่และปิดหรือดิสเอเบิลการใช้งานเมื่อปลดอุปกรณ์ออกไปแล้ว
- รายงานสถานะของแต่ละพอร์ตเมื่อไดรเวอร์โฮสต์คอนโทรลเลอร์ร้องขอมา



รูปที่ 3.3 บล๊อคไดอะแกรมแสดงการทำงานของ USB รูทฮับอย่างง่าย

3.4.2 USB ฮับ (USB Hub)

หน้าที่หลักๆ ของ USB คือการขยายเชื่อมต่อให้อุปกรณ์จำนวนมากๆ สามารถเชื่อมต่อกับระบบบัสได้ โดยการทำงานหลักของ USB ฮับนั้นมีอยู่ 2 ส่วน คือ ทำหน้าที่เป็นตัวทวนสัญญาณ (repeater) และตัวจัดการพลังงาน (Power management) ในส่วนของการทวนสัญญาณ USB ฮับจะต้องรับสัญญาณจากโฮสต์มา แล้วส่งกระจายออกไปยังพอร์ตทุก ๆ พอร์ต และรับสัญญาณจากแตะละพอร์ต แล้วรับมารวมกันเพื่อส่งกลับไปที่โฮสต์สำหรับส่วนของการจัดการพลังงานนั้นๆ มีหน้าที่เหมือนกับรูตฮับก็คือตรวจสอบว่ามีการต่ออยู่ของอุปกรณ์ที่พอร์ตใดบ้างหากมีอุปกรณ์ต่ออยู่ก็เปิดการใช้งานพอร์ตนั้นๆ หากไม่มีอุปกรณ์อยู่ก็ปิดการใช้งาน ตรวจสอบการเชื่อมต่อหรือปลดออกของอุปกรณ์เพื่อรายงานผลเมื่อโฮลต์คอนโทรลเลอร์ร้องขอ และป้องกันอุปกรณ์ที่ต่ออยู่แต่ละพอร์ตไม่ให้ดึงกระแสไฟฟ้าเกินกว่าที่กำหนด

3.4.3 อุปกรณ์ USB (USB Device)

ส่วนประกอบนี้นั้นก็คืออุปกรณ์ต่างๆ ที่นำมาต่อพ่วงกับพอร์ต USB นั้นเอง โดยการจัดประเภทของอุปกรณ์ USB นั้น สามารถจัดประเภทได้จากความเร็วในการส่งถ่ายข้อมูล และรูปแบบการใช้พลังงาน

ของอุปกรณ์ โดยการจัดประเภทจากความเร็วในการถ่ายส่งข้อมูลนั้นแต่เดิมสามารถจัดได้เป็น 2 ประเภท ได้แก่

- อุปกรณ์ความเร็วต่ำ (Low-Speed Devices) เช่น เมาส์, จอยสติ๊ก, คีย์บอร์ด เป็นต้น ซึ่งจะใช้ความเร็วในการส่งถ่ายเท่ากับ 1.5 Mbit/sec

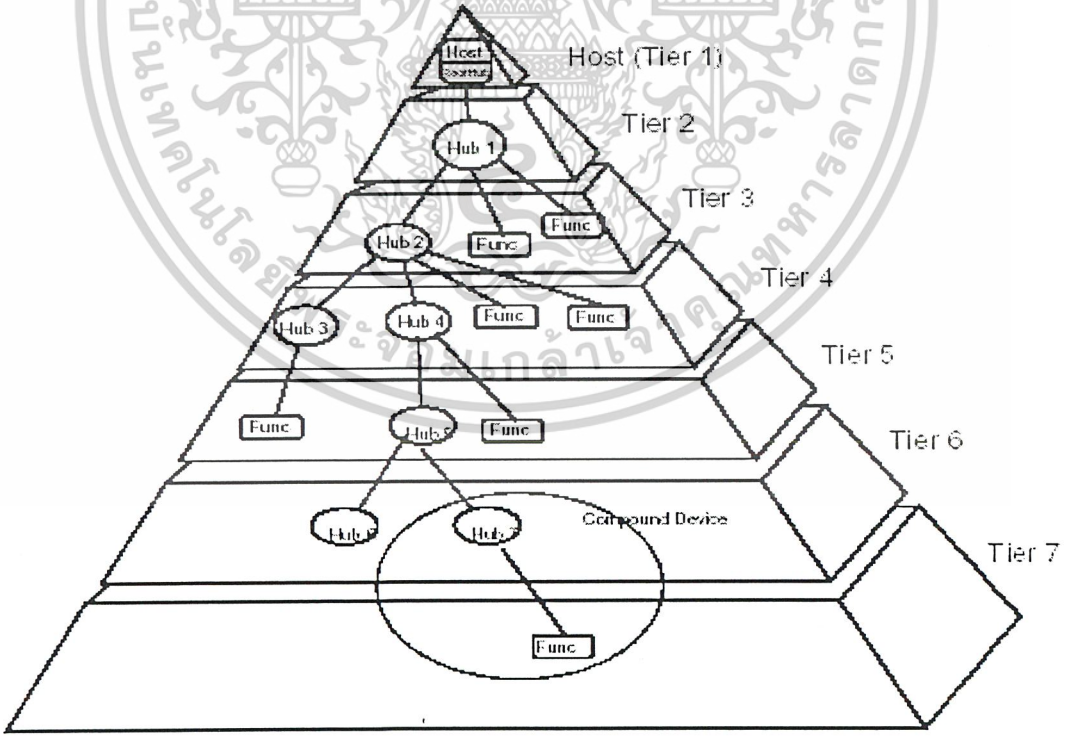
- อุปกรณ์ความเร็วเต็มที่ (Full-Speed Devices) เช่นเครื่องพิมพ์, กล้องดิจิตอล, ซีดีรอมไดร์ฟ, เครื่องเล่น MP3 เป็นต้น ซึ่งใช้ความเร็วในการส่งถ่ายข้อมูลเท่ากับ 12 Mbit/sec

แต่ในปัจจุบันนั้น ได้มีการเพิ่มเติมส่วนของมาตรฐาน USB 2.0 ซึ่งมีความเร็วการส่งถ่ายข้อมูลในระดับความเร็วสูง (High speed) เท่ากับ 480 Mbit/sec เพิ่มเติมขึ้นมา ซึ่งจะกล่าวถึงต่อไปในภายหลัง สำหรับการจัดประเภทของอุปกรณ์จากการใช้พลังงานของตัวมันนั้น จะแบ่งออกได้เป็น 2 ประเภท ได้แก่

- อุปกรณ์ที่ใช้พลังงานจากระบบบัส (Bus Powered Device) ได้แก่อุปกรณ์ที่ใช้พลังงานจากระบบบัสโดยตรง เช่น Flash Drive เป็นต้น

- อุปกรณ์ที่ใช้พลังงานจากตัวเอง (Self Powered Device) ได้แก่อุปกรณ์ที่ใช้แหล่งจ่ายพลังงานจากตัวเองโดยไม่พึ่งพลังงานจากระบบบัส

นอกจากนี้ยังมีอุปกรณ์ USB บางประเภทที่มีคุณสมบัติของ USB ฮับอยู่ด้วย นั่นคือสามารถนำอุปกรณ์อื่นๆ มาเชื่อมต่อเข้ากับตัวมันไปได้เราเรียกอุปกรณ์ประเภทนี้ว่า Compound USB Device



รูปที่ 3.4 แสดงโครงสร้างการเชื่อมต่อบนระบบบัส USB

3.5 รูปแบบการส่งถ่ายข้อมูลบนระบบบัส USB

การถ่ายทอดสัญญาณ (transfer type) ของบัส USB นั้นแบ่งออกเป็น 4 ชนิดตามขนาดชนิดของข้อมูลและจังหวะการส่งข้อมูลดังนี้

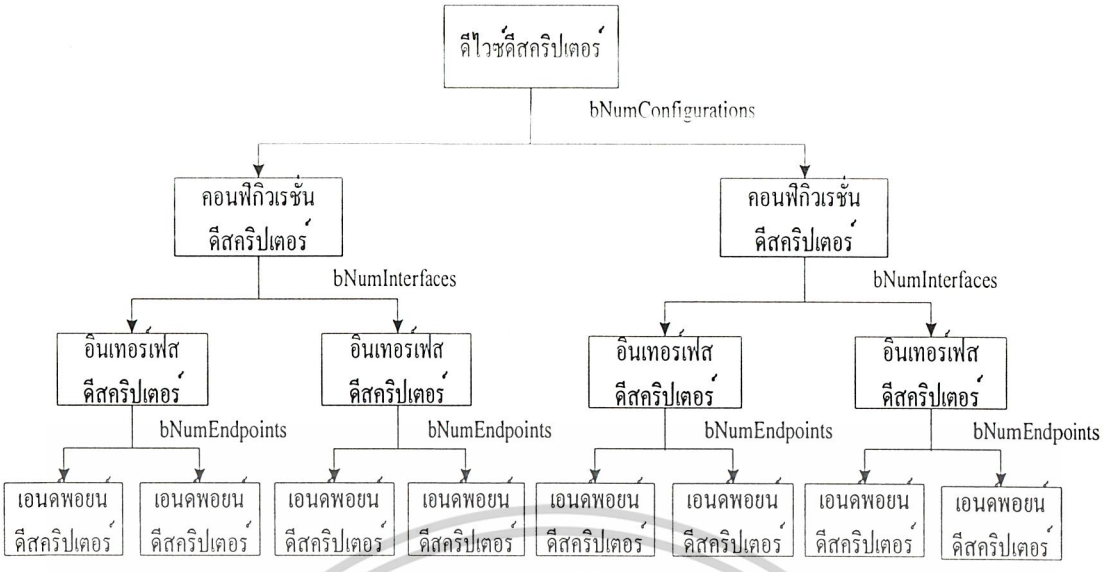
1. การถ่ายทอดสัญญาณแบบไอโซโครนัส (Isochronous transfer)
2. การถ่ายทอดสัญญาณแบบบัลก์ (Bulk transfer)
3. การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์ (Interrupt transfer)
4. การถ่ายทอดสัญญาณแบบควบคุม (Control transfer)

การถ่ายทอดสัญญาณ 3 ชนิดแรกใช้สำหรับข้อมูลทั่วไปที่ต้องการรับหรือส่งไปยังตัวอุปกรณ์ ส่วนการถ่ายทอดสัญญาณแบบที่ 4 การถ่ายทอดสัญญาณแบบไอโซโครนัสใช้ถ่ายทอดข้อมูลที่ต้องการความต่อเนื่องสูง เช่น ข้อมูลเสียงเพลง ส่วนการถ่ายทอดสัญญาณแบบบัลก์ใช้สำหรับถ่ายทอดข้อมูลที่มีปริมาณมากๆ แต่ไม่ต้องการความต่อเนื่องของข้อมูล ในขณะที่การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์ใช้สำหรับถ่ายทอดข้อมูลที่มีจำนวนน้อยครั้งและมีปริมาณของข้อมูลไม่มาก

ในการสั่งงานแต่ละครั้งนั้น โฮสต์จะต้องระบุเป้าหมายปลายทางของข้อมูลที่ต้องการจะรับหรือส่ง เป้าหมายปลายทางที่ว่านี้เป็นกลุ่มของรีจิสเตอร์ที่อยู่ในอุปกรณ์ชนิดต่าง ๆ 4 ชนิดข้างต้นแยกกันออกไป ดังนั้นอุปกรณ์แต่ละตัวจะมีจำนวนเอ็นพอยต์มากกว่า 1 เอ็นพอยต์เพื่อรองรับการทำงานรูปแบบต่าง ๆ ยกตัวอย่าง ซีดีรอม USB แบบอ่านเขียนได้จะต้องมีเอ็นพอยต์ที่รองรับการถ่ายทอดสัญญาณควบคุมเพื่อรับคำสั่งจากโฮสต์ 1 เอ็นพอยต์ พร้อมกันนั้นยังต้องมีเอ็นพอยต์ที่รองรับถ่ายทอดสัญญาณแบบบัลก์เพื่อส่งข้อมูลที่อ่านได้หรือรับข้อมูลเพื่อเขียนแผ่นซีดี 1 เอ็นพอยต์ และต้องมีเอ็นพอยต์ที่รองรับการถ่ายทอดสัญญาณไอโซโครนัส เพื่อส่งข้อมูลเพลงซึ่งมีความต่อเนื่องในการที่เล่นแผ่นซีดีเพลง 1 เอ็นพอยต์ เป็นต้น

3.6 ความหมาย ชนิด และรูปแบบการทำงานของดิสคริปเตอร์

อุปกรณ์แต่ละตัวมีคุณสมบัติและการทำงานที่แตกต่างกัน โฮสต์จำเป็นต้องรู้คุณสมบัติทั้งหมดของอุปกรณ์แต่ละตัวเพื่อให้การสั่งงานเป็นไปอย่างถูกต้อง และเนื่องจากบัสข้อมูลทั้งหมดจะถูกใช้งานรับส่งข้อมูลร่วมกันระหว่างอุปกรณ์ทุก ๆ ตัว สิ่งที่โฮสต์จำเป็นต้องรู้อีกอย่างคือปริมาณข้อมูลที่ต้องการส่ง (bandwidth) ของอุปกรณ์แต่ละตัวในบัส ดังนั้นเมื่อมีอุปกรณ์ตัวใหม่ต่อเข้ากับบัสโฮสต์ต้องอ่านข้อมูลต่าง ๆ ที่จำเป็นเข้ามาเพื่อใช้อ้างอิงในการสั่งงานอุปกรณ์ (device descripters) ซึ่งอุปกรณ์แต่ละตัวจะแจ้งรายละเอียดของตัวเองให้โฮสล์รู้ผ่านดริสคริปเตอร์ชนิดต่างๆ ซึ่งได้รับการแบ่งแยกเป็นชนิดตามข้อมูลที่จะแจ้งกลับไปยังโฮสต์ โดยการแบ่งแยกชนิดของคริปเตอร์นั้นจะจัดเป็นลำดับชั้น สาเหตุที่ต้องจัดเป็นระดับชั้นเพราะว่าอุปกรณ์แต่ละตัวนั้นอาจมีการทำงานที่หลากหลายรูปแบบ เช่น แบ่งการทำงานออกเป็น 2 โหมด แต่ละโหมดมีหน้าที่การทำงานแตกต่างกัน และแต่ละหน้าที่ก็จะใช้กลุ่มของเอ็นพอยต์ที่แตกต่างกันซึ่งสามารถสรุปได้ดังนี้



รูปที่ 3.5 แสดงระดับการทำงานของดิสคริปเตอร์ในระบบบัส USB

3.6.1 ดีไวซ์ดิสคริปเตอร์ (Device descriptor)

ทำหน้าที่หลักในการเก็บข้อมูลโดยทั่วไปของตัวอุปกรณ์ ซึ่งในอุปกรณ์ซึ่งในอุปกรณ์แต่ละตัวจะมีดีไวซ์ดิสคริปเตอร์เพียง 1 ชุดเท่านั้น ภายในจะระบุข้อมูลที่ใช้ในการเชื่อมต่อขั้นแรก (default communications pipe) เพื่อใช้ในการกำหนดข้อมูลสถานะของอุปกรณ์เข้ากับโฮสต์ นอกจากนั้นยังเก็บข้อมูลของข้อกำหนดในโหมดการทำงานต่าง ๆ ของตัวอุปกรณ์รวมถึงจำนวนคอนฟิกิวเรชั่นด้วย เพราะในครั้งแรกที่อุปกรณ์เชื่อมต่อเข้ากับบัสนั้น โฮสต์ไม่มีทางรู้ได้เลยว่าต้องติดต่อกับอุปกรณ์ที่เอ็นพอยต์ใด จึงจำเป็นต้องขอข้อมูลส่วนนี้ก่อนที่จะติดต่อกับส่วนอื่น

3.6.2 คอนฟิกิวเรชั่นดิสคริปเตอร์ (Configuration descriptor)

ใช้เก็บข้อมูลที่จำเป็นของการทำงานในแต่ละโหมดการทำงานและเก็บจำนวนอินเตอร์เฟสที่ใช้งานในโหมดนั้นๆ เช่น อุปกรณ์บางตัวมีการทำงาน 2 โหมด คือ โหมดใช้พลังงานสูง และโหมดประหยัดพลังงาน ดิสคริปเตอร์ตัวนี้จะเก็บข้อมูลที่จำเป็นในการตั้งค่าต่างๆ ของโฮสต์เมื่อต้องการเลือกใช้โหมดการทำงานแต่ละโหมดของอุปกรณ์

3.6.3 อินเตอร์เฟสดิสคริปเตอร์ (Interface descriptor)

ใช้เก็บข้อมูลหน้าที่การทำงานภายใน ซึ่งในแต่ละโหมดการทำงานหรือคอนฟิกิวเรชั่นอาจจะมีการเชื่อมต่อหรืออินเตอร์เฟสเพียง 1 แบบหรือมากกว่าเพื่อใช้งานในหน้าที่ต่างๆ ตัวอย่างที่เห็นได้ชัดที่สุดก็คือซีดีรอม โดยในตัวซีดีรอมจะมีการรับส่งข้อมูลปริมาณมาก (mass storage) การส่งข้อมูลเสียงออดิโอและการส่งข้อมูลภาพจะเห็นได้ว่ามีอินเตอร์เฟสที่ใช้งานแยกกัน ภายในอินเตอร์เฟสดิสคริปเตอร์จะบรรจุข้อมูลการใช้งานอินเตอร์เฟสนั้นๆ โดยข้อมูลเหล่านี้จะระบุว่าอุปกรณ์ถูกจัดอยู่ในคลาส (class) หรือคลาสย่อย (subclass) ใด และมีเอ็นด์พอยต์จำนวนเท่าใดที่ใช้งานในอินเตอร์เฟสนี้บ้าง

3.6.4 เอ็นด์พอยต์ดิสคริปเตอร์ (Endpoint descriptor)

ใช้เก็บข้อมูลคุณสมบัติของแต่ละเอ็นด์พอยต์ เช่น ใช้การถ่ายทอดสัญญาณแบบใด(ไอโซโครนัส, บัลก์, อินเตอร์รัปต์ หรือ สัญญาณควบคุม) และถ่ายทอดข้อมูลได้มากที่สุดครั้งละเท่าใด

3.6.5 สตริงดิสคริปเตอร์ (String descriptor)

เป็นดิสคริปเตอร์ที่ไม่ได้อยู่ภายในโครงสร้างตามรูปที่ 2.5 เพราะดิสคริปเตอร์ชนิดนี้จะเก็บข้อมูลตัวอักษรที่สามารถอ่านเข้าใจได้ไว้อธิบายส่วนต่างๆ ของดิสคริปเตอร์ทั้งสี่ตัวข้างต้น เช่น เก็บชื่อบริษัทผู้ผลิต หรือหมายเลขประจำตัวของอุปกรณ์ เป็นต้น

3.7 การจัดการกับอุปกรณ์บนบัส USB ที่มีความเร็วต่างกัน

อุปกรณ์ USB แบ่งตามความเร็วของการถ่ายทอดข้อมูลได้ 2 ชนิด คือ อุปกรณ์ความเร็วเต็มที่จะถ่ายทอดข้อมูลที่อัตรา 12 Mb/s และอุปกรณ์ความเร็วต่ำถ่ายทอดข้อมูลที่อัตรา 1.5 Mb/s แต่เนื่องจากอุปกรณ์ทั้งสองประเภทนี้ต่ออยู่บนบัสเดียวกันทั้งหมด ดังนั้นพอร์ตของฮับที่ให้อุปกรณ์ USB เข้ามาเชื่อมต่อจะต้องรองรับการทำงานได้ทั้ง 2 แบบ และด้วยความที่เป็นระบบบัส อุปกรณ์ทุกๆ ตัวจะได้รับข้อมูลทุกๆ แพ็กเก็ตที่ส่งเข้ามาไม่ว่าจะเป็นด้วยการความเร็วสูงหรือต่ำ ดังนั้น จึงต้องมีการจัดการจราจรระหว่างข้อมูลที่ส่งด้วย โดยตั้งเป็นข้อกำหนดว่า พอร์ตของอุปกรณ์ความเร็วต่ำต้องไม่เปิดทำงานจนกว่าจะได้รับสัญญาณ “Preamble” จาก โฮสต์ โดยหลังจากได้รับสัญญาณ Preamble แล้ว ฮับจะเปลี่ยนความเร็วในการถ่ายทอดข้อมูลไปสู่โหมดความเร็วต่ำ แล้วเปิดการทำงานของพอร์ตที่ความเร็วต่ำทำให้อุปกรณ์ที่เชื่อมต่อด้วยความเร็วต่ำได้รับข้อมูล ในทางกลับกันหลังจากได้รับสัญญาณ Preamble แล้ว อุปกรณ์ความเร็วเต็มที่ก็จะทราบทันทีว่าข้อมูลที่ตามหลังมาจะอยู่ในโหมดความเร็วต่ำ ไม่ต้องอ่านและตีความข้อมูลเหล่านั้น

3.8 การส่งสัญญาณในบัส USB

สัญญาณที่ปรากฏบนสายสัญญาณระหว่าง รูตฮับและอุปกรณ์จะส่งไปในแบบสัญญาณผลต่าง (differential signaling) เพื่อลดการแพร่กระจายสนามแม่เหล็กไฟฟ้า (EMI : Electromagnetic Interference) เนื่องจากจากตามธรรมชาติของสัญญาณไฟฟ้า เมื่อมีการเปลี่ยนแปลงระดับสัญญาณด้วยความเร็วมากๆ จะทำให้เกิดการแพร่กระจายของสนามแม่เหล็กและสนามไฟฟ้าออกมารอบๆสายส่งสัญญาณ ซึ่งอาจรบกวนการทำงานของอุปกรณ์ต่างๆ รอบข้างได้ด้วยการส่งสัญญาณแบบนี้จะทำให้การเปลี่ยนแปลงของสัญญาณในสายนำสัญญาณเกิดขึ้นมาพร้อมกันและเกิดในลักษณะตรงข้ามทำให้สนามแม่เหล็กไฟฟ้าที่เกิดขึ้นหักล้างกัน ไม่แพร่ออกมาภายนอก

3.9 กระบวนการกำหนดการทำงานของอุปกรณ์

ในส่วนนี้เป็นลำดับขั้นตอนในการเชื่อมต่ออุปกรณ์ USB ไปยังคอมพิวเตอร์ในเบื้องต้นโดยมีลำดับขั้นตอนดังต่อไปนี้

- ฮับตรวจสอบพบว่ามีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้าสู่ระบบ แล้วแจ้งผลกลับไปยังโฮสต์คอนโทรลเลอร์

- โฮสต์คอนโทรลเลอร์สั่งให้ฮับเปิดการทำงานของแหล่งจ่ายไฟในโหมดประหยัดเพื่อให้อุปกรณ์ที่อาศัยพลังงานจากบัสสามารถทำงานได้

- โฮสต์คอนโทรลเลอร์สั่งให้ฮับรีเซตพอร์ตที่อุปกรณ์รีเซตค่าแอดเดรสและเอ็นด์พอยด์ของตัวเองให้เป็นค่าเริ่มต้น (default)

-โฮสต์อ่านดิสคริปเตอร์ต่างๆ จากตัวอุปกรณ์และพิจาณาว่าทรัพยากรของระบบพอเพียงต่อความต้องการของตัวอุปกรณ์หรือไม่ ทรัพยากรในที่นี้คือพลังงานไฟฟ้าและปริมาณข้อมูลที่จะส่งของตัวอุปกรณ์ หากพิจารณาแล้วว่าไม่สามารถทำงานได้ก็สั่งให้ฮับปิดการทำงานของพอร์ตนั้น

- เมื่อโฮสต์พิจารณาแล้วว่าสามารถให้บริการแก่อุปกรณ์ตัวที่มาเชื่อมต่อได้ จะควบคุมให้แหล่งจ่ายไฟจ่ายพลังงานตามที่อุปกรณ์ต้องการ รวมถึงการตั้งค่าแอดเดรสและกำหนดค่าต่างๆ

- หลังจากตั้งค่าเรียบร้อย คอมพิวเตอร์ก็จะรู้จักกับอุปกรณ์ตัวใหม่และสามารถติดต่อกับอุปกรณ์ได้ทันทีโดยไม่จำเป็นต้องทำการ Restart เครื่องคอมพิวเตอร์

3.10 ระบบบัส USB 2.0

เป็นระบบบัสที่ได้รับการพัฒนาขึ้นอย่างมากจากระบบบัส USB 1.1 ซึ่งมีคุณสมบัติเด่นด้านความเร็วในการส่งถ่ายข้อมูลที่สูงถึง 480 Mbit/sec ทำให้ขอบเขตในการใช้งานระบบบัส USB กว้างขวางมากยิ่งขึ้น โดยอุปกรณ์ที่ใช้ระบบบัส USB 2.0 นั้นจำเป็นที่จะต้องต่อร่วมกันระบบ (โฮสต์และฮับ) ที่ใช้ระบบบัส USB 2.0 เช่นเดียวกันจึงสามารถส่งถ่ายข้อมูลที่ระดับความเร็วสูงนี้ได้แต่อย่างไรก็ตามระบบก็ยังสามารถใช้งานร่วมกับอุปกรณ์ที่เป็น USB 1.0/1.1 แต่เดิมได้ โดยส่วนของโฮสต์และฮับ จะทำหน้าที่ในการปรับระดับความเร็วในการส่งถ่ายข้อมูลให้ตรงกัน ซึ่งคุณสมบัตินี้เป็นการเพิ่มความซับซ้อนให้กับวงจรและรูปแบบโปรโตคอลของโฮสต์และฮับ แต่ถ้าเป็นการทำให้ไม่จำเป็นต้องใช้โฮสต์และฮับ ที่มีความเร็วแตกต่างกันหลาย ๆ ตัวในระบบ

สำหรับในส่วนการอินเตอร์เฟส USB ของโครงงานนี้นั้นได้เลือกใช้ IC ที่ทำหน้าที่ในการแปลงรูปแบบการสื่อสารจาก USB ไปยังการสื่อสารในรูปแบบอนุกรม ซึ่งทำให้ไม่ต้องเข้าจัดการกับกระบวนการอินเตอร์เฟสกับ USB โดยตรง

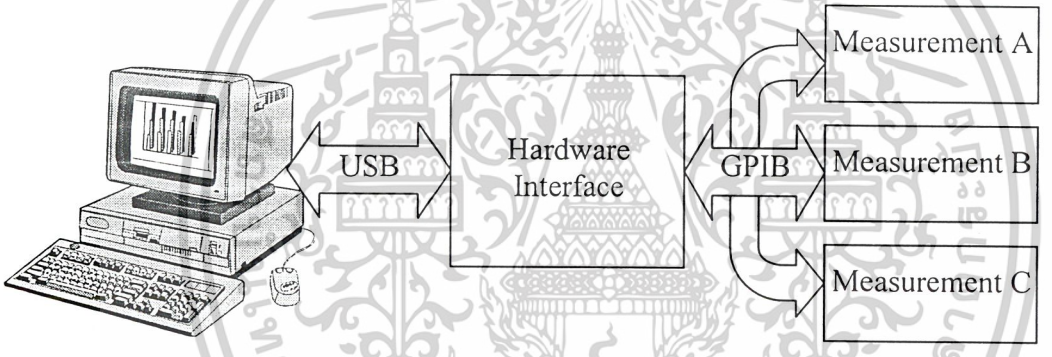
บทที่ 4

การออกแบบส่วนอินเตอร์เฟสของระบบ

4.1 องค์ประกอบของระบบ

ภายในระบบควบคุมอุปกรณ์เครื่องมือวัดนั้น ประกอบไปด้วยส่วนประกอบหลัก 3 ส่วน ได้แก่

- โปรแกรมสั่งการทางคอมพิวเตอร์
- ส่วนการอินเตอร์เฟสของระบบ
- ส่วนของอุปกรณ์เครื่องมือวัดที่ต้องการควบคุม



รูปที่ 4.1 แสดงองค์ประกอบโดยรวมของระบบ

4.2 หน้าที่ของส่วนการอินเตอร์เฟสของระบบ

ในส่วนนี้ถือได้ว่าเป็นหัวใจของระบบ ซึ่งทำหน้าที่เสมือนตัวแปลภาษาระหว่างระบบบัส GPIB ไปยัง USB และ RS-232 ซึ่งหน้าที่หลักของส่วนการอินเตอร์เฟสนี้ คือจัดเรียงข้อมูลให้ถูกต้องตามมาตรฐานของแต่ละ พอร์ต โดยในที่นี้นั้นได้เลือกใช้ ไมโครคอนโทรลเลอร์ PIC16F877 มาทำหน้าที่ในการจัดเรียงข้อมูลเพื่ออินเตอร์เฟสไปยังระบบบัส GPIB โดยการส่งข้อมูลระหว่างระบบบัส GPIB และไมโครคอนโทรลเลอร์นั้นจำเป็นจะต้องกระทำให้ถูกต้องตามมาตรฐานของ GPIB สำหรับในส่วนของข้อมูล ที่จะติดต่อไปยังคอมพิวเตอร์นั้น ข้อมูลจะถูกสร้างขึ้นใหม่โดยไมโครคอนโทรลเลอร์ และส่งออกไปในรูปแบบข้อมูลอนุกรม ซึ่งข้อมูลที่ได้มานี้จะถูกนำไปเปลี่ยนแปลงรูปแบบในการติดต่อกับคอมพิวเตอร์ตามต้องการ

4.3 ขบวนการติดต่อไปยังอุปกรณ์เครื่องมือวัด

ในการเชื่อมต่อระบบไปยังอุปกรณ์เครื่องมือวัดต่าง ๆ นั้น เนื่องจากอุปกรณ์เครื่องวัดที่ต่อร่วมอยู่ภายในระบบนั้นล้วนแล้วแต่ใช้สายส่งข้อมูล (Data) ชุดเดียวกัน ดังนั้นในช่วงเวลาหนึ่งจะมีอุปกรณ์ที่ได้รับอนุญาตให้รับหรือส่งข้องมูลเพียงอุปกรณ์เดียวเท่านั้น ดังนั้นส่วนที่จะทำหน้าที่ในการระบุอุปกรณ์ที่ได้รับอนุญาตก็คือ Address นั้นจำเป็นจะต้องกระทำก่อนการรับหรือส่งข้อมูลไปยังอุปกรณ์ใด ๆ โดยลำดับขั้นตอนของการระบุ Address นั้นเป็นไปตามลำดับขั้นตอนดังนี้

ในขั้นแรกนั้นสัญญาณที่มีความสำคัญในการกำหนดรูปแบบของข้อมูลที่สั่งการออกไปนั่นคือสัญญาณ ATN ซึ่งจะมีความสำคัญต่อระบบดังต่อไปนี้

- ในขณะที่สัญญาณ ATN เป็น Low

ในขณะที่สัญญาณ ATN เป็น Low นั้นข้อมูลบนบัสข้อมูลของมันนั้นทางอุปกรณ์จะรับรู้ว่าข้อมูลเหล่านี้เป็นรหัสคำสั่งมาตรฐานของ GPIB ดังตารางที่ 1 ซึ่งในระหว่างที่ ATN เป็น Low นี้รหัสคำสั่งจะถูกส่งออกไปกี่คำสั่งก็ได้

- ในขณะที่สัญญาณ ATN เป็น High

เมื่อสัญญาณ ATN อยู่ในสถานะ High นั้น ข้อมูลที่อยู่บนบัสข้อมูลจะหมายถึงชุดข้อมูลคำสั่ง ซึ่งชุดข้อมูลนี้จะแตกต่างกันไปสำหรับอุปกรณ์ที่แตกต่างกัน โดยการส่งข้อมูลเหล่านี้จะถูกส่งออกไปทีละตัวอักษร (ตามรหัส ASCII) และถูกควบคุมโดยสัญญาณ EOI เพื่อระบุชุดของคำสั่งนั่นเอง

ในการกำหนดการเริ่มติดต่อสื่อสารข้อมูลระหว่างกันนั้นจำเป็นจะต้องระบุ Address ของส่วนที่ทำการติดต่อ โดยในการกำหนดส่วนการเริ่มต้นนี้จะแบ่งออกได้ 2 กรณี ได้แก่ ระหว่างตัวควบคุมกับตัวส่ง หรือตัวควบคุมกับตัวรับ ซึ่งในขณะที่ทางด้านหนึ่งทำหน้าที่เป็นผู้ส่ง อีกด้านหนึ่งก็จะทำหน้าที่เป็นผู้รับนั่นเอง

- ขณะที่ตัวควบคุมติดต่อกับภาคส่ง

ในขณะนี้นั้นตัวควบคุมจะทำหน้าที่เสมือนเป็นทางด้านรับ (Listener) โดยในขณะนี้นั้น ATN จะถูกกำหนดให้เป็น Low เป็นอันดับแรกเพื่อกำหนดให้เข้าสู่โหมดของการสั่งงานคำสั่งมาตรฐาน ดังแผนผังเวลาในรูปที่ 4.2 ซึ่งในส่วนของการกำหนดค่าคำสั่งต่างๆ นั้นมีรายละเอียดดังต่อไปนี้

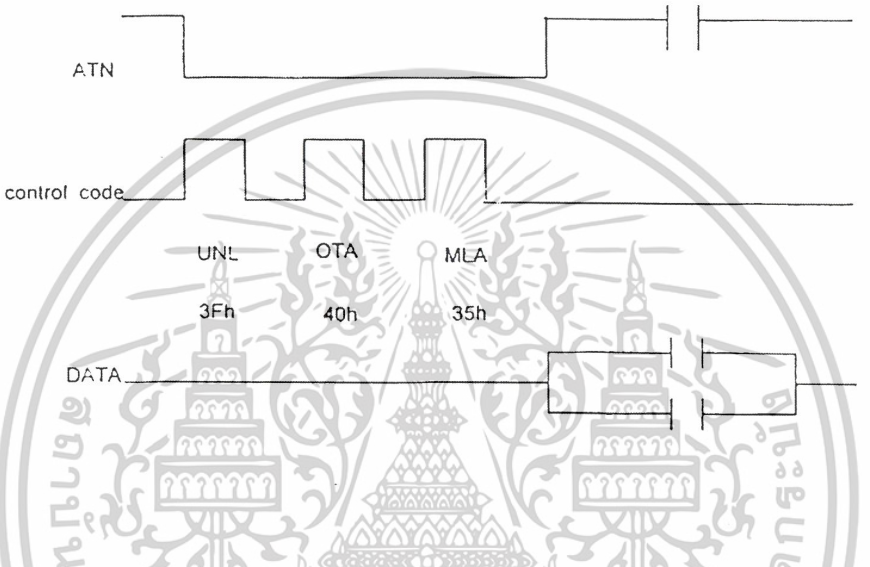
UNL (3Fh) (Unlistener) คำสั่งนี้เป็นคำสั่งเคลียร์อุปกรณ์ที่ทำหน้าที่เป็นตัวรับที่ต่อพ่วงอยู่ภายในระบบทุกตัว ซึ่งจะทำให้ไม่มีอุปกรณ์ใดในระบบทำหน้าที่เป็นตัวรับอีก

OTA (40h + Address) (Our Talker Address) ในส่วนนี้เป็นการแจ้งค่า Address ของตัวควบคุมให้อุปกรณ์ที่ต่อพ่วงอยู่บนบัสทุกตัวรับรู้ซึ่ง Address ของตัวควบคุมนั้นจะเป็น 0 โดยในการส่งนั้นจะเป็นการส่งค่า 40h บวกกับค่า Address ของตัวควบคุมไปได้แก่ 0 จึงเป็นการส่งค่า 40h ออกไปนั่นเอง

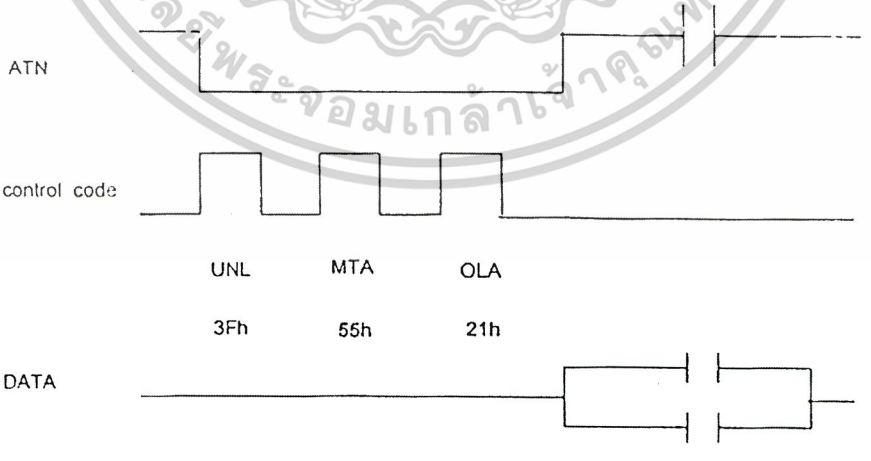
MLA (20h+Address) (My Listener Address) คำสั่งนี้เป็นการกำหนดค่าของ Address ของอุปกรณ์ตัวส่งซึ่งเป็นไปในลักษณะเดียวกับการกำหนด Address ของตัวควบคุม แต่จะเป็นการส่งค่า 20h บวกกับค่า

Address อุปกรณ์ที่ต้องการให้ทำหน้าที่เป็นตัวส่ง ซึ่งในตัวอย่างนี้นั้นได้แก่ 21 (15h) ดังนั้นค่าที่ส่งออกไปจึงเป็น 20h+15h = 35h ดังตัวอย่าง

หลังจากคำสั่งเหล่านี้ถูกสั่งงานแล้ว ตัวควบคุมจะทำการส่งสัญญาณ ATN เป็น High เพื่อที่จะเริ่มทำการรับค่าข้อมูลที่จะส่งมาจากตัวส่งต่อไป



รูปที่ 4.2 แสดงผังเวลาเมื่อตัวควบคุมติดต่อไปยังภาคส่ง



รูปที่ 4.3 แสดงผังเวลาเมื่อตัวควบคุมติดต่อไปยังภาครับ

- ขณะที่ตัวควบคุมติดต่อกับภาครับ

ในขณะที่ตัวควบคุมทำหน้าที่เป็นภาคส่งเพื่อติดต่อกับภาครับนี้นั้นจะเป็นไปในลักษณะเดียวกับขณะที่ตัวควบคุมทำหน้าที่เป็นภาครับ โดยการสั่งการจะเริ่มต้นจากการที่กำหนดให้สัญญาณ ATN เป็น Low แล้วจึงเริ่มส่งชุดข้อมูลออกไป

UNL (3Fh) (Unlistener) เป็นคำสั่งยกเลิกอุปกรณ์ที่ทำหน้าที่เป็นตัวรับอยู่ทุกตัวเพื่อรอรับคำสั่งใหม่

MTA (20h+Address) (My Talker Address) เป็นคำสั่งที่จะแจ้ง Address ของตัวควบคุมที่จะให้ตัวรับรับรู้ว่าส่งมาจาก Address ใด โดยค่าในการส่งจะเป็นค่า 40h+Address ของตัวควบคุมซึ่งดังตัวอย่างรูปที่ 4.3 นั้น Address ของตัวควบคุมจะเป็น 21 (15h) ดังนั้นค่าที่ส่งไปจึงเป็น 40h+15h = 55h ดังรูป

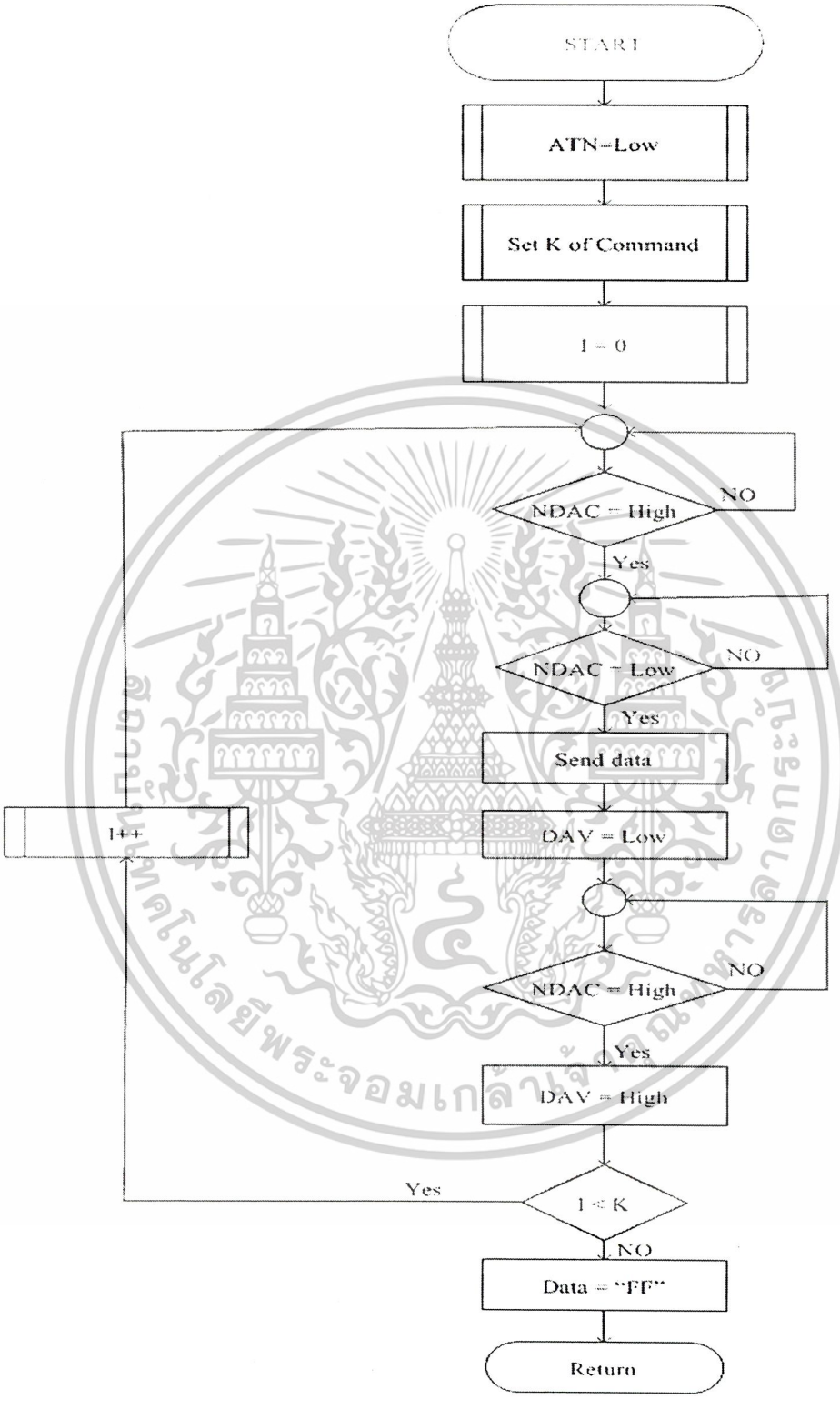
OLA (20h+Address) (Our Listener Address) เป็นคำสั่งที่ใช้กำหนด Address ของตัวอุปกรณ์ที่ทำหน้าที่เป็นตัวรับโดยค่าที่ส่งออกไปจะเป็น 20h+Address อุปกรณ์ตัวรับซึ่งดังรูปนั้นอุปกรณ์ตัวรับมี Address เป็น 1 ดังนั้นค่าที่ส่งออกไปจึงเป็น 21h นั่นเอง

4.4 การออกแบบขบวนการแฮนด์เช็คในการรับส่งข้อมูล

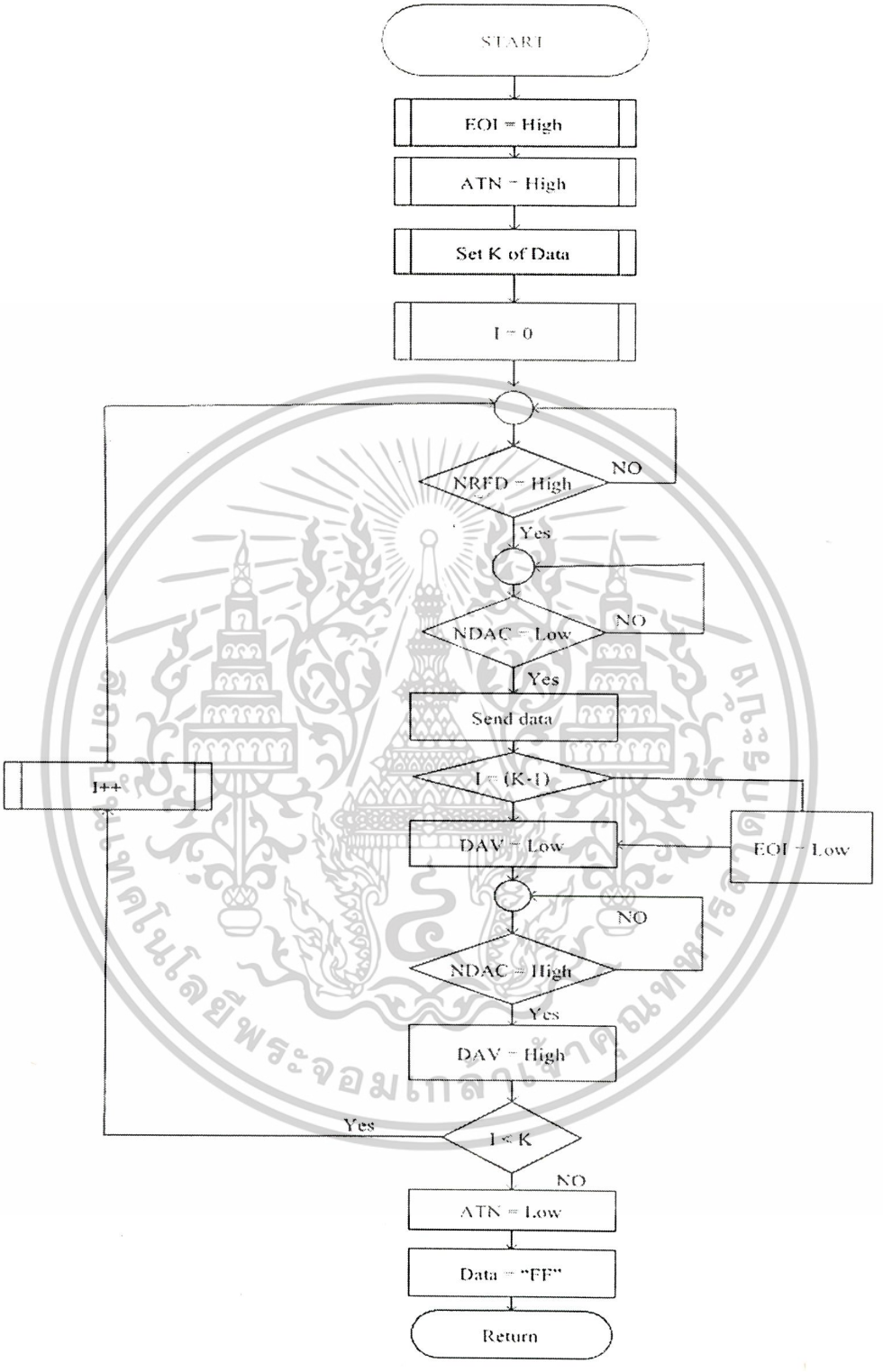
ในการรับส่งข้อมูลบนระบบบัส GPIB นั้นต้องกระทำให้เป็นไปตามขบวนการแฮนด์เช็คของระบบบัส GPIB โดยการออกแบบให้อยู่ในรูปของโปรแกรมไมโครคอนโทรลเลอร์นั้น แสดงดังรูป 4.4 ซึ่งเป็นโฟล์วชาร์ตแสดงขบวนการแฮนด์เช็คในการส่งคำสั่งรหัสมาตรฐาน และรูปที่ 4.5 เป็นโฟล์วชาร์ตแสดงขบวนการแฮนด์เช็คในการส่งชุดข้อมูลคำสั่ง

- ขบวนการแฮนด์เช็คในการส่งรหัสคำสั่งมาตรฐาน

ในขั้นแรกนั้นเป็นการกำหนดให้สัญญาณ ATN เป็น Low เพื่อเป็นการบ่งบอกว่าเป็นการส่งรหัสคำสั่งมาตรฐาน (ในสภาวะปรกตินั้นสัญญาณ ATN ก็จะคงอยู่ในสภาวะ Output Low) ต่อมาก็จะเป็นการกำหนดจำนวนชุดของคำสั่งที่ต้องการจะทำการส่ง โดยเก็บค่าในตัวแปร K และกำหนดตัวแปรนับชุดข้อมูล (I) เป็น 0 หลังจากนั้นจะเริ่มเข้าสู่ขบวนการส่ง โดยตรวจสอบสัญญาณ NRFD ว่าเป็น High หรือไม่ และตรวจสอบสัญญาณ NDAC ว่าเป็น Low หรือไม่ เมื่อเป็นไปตามเงื่อนไขแล้วข้อมูลจะถูกส่งออกไปยังพอร์ต Data (ชุดข้อมูลจากไมโครคอนโทรลเลอร์ ต้องทำการ Invert ก่อนที่จะทำการส่งไปยังพอร์ต Data) หลังจากนั้นชั่วขณะหนึ่งสัญญาณ DAV จะถูกกำหนดให้เป็น Low เพื่อบ่งบอกไปยังทางภาคส่งว่าข้อมูลได้ถูกส่งออกไปแล้ว หลังจากนั้นทางภาคส่งจะตรวจสอบสัญญาณ NDAC อีกครั้งว่าเป็น High หรือไม่ ซึ่งเป็นการบ่งบอกว่าข้อมูลที่ส่งไปนั้นทางภาครับได้รับข้อมูลแล้วทางภาคส่งก็จะกำหนดให้สัญญาณ DAV เป็น High เพื่อเป็นการบ่งบอกการสิ้นสุดของการส่งข้อมูลใน 1 ชุดข้อมูลหลังจากนั้นจะเป็นการตรวจสอบว่าชุดรหัสคำสั่งนั้นสิ้นสุดแล้วหรือไม่ หากยังไม่จบชุดของรหัสก็จะทำการเข้าสู่กระบวนการส่งอีกครั้งเพื่อส่งรหัสคำสั่งที่เหลืออยู่ หลังจากขบวนการส่งจบสิ้นแล้วก็จะทำการเคลียร์พอร์ต Data โดยกำหนดข้อมูลให้เป็น FF เป็นอันเสร็จสิ้นขบวนการส่งรหัสคำสั่ง



รูปที่ 4.4 โฟล์วชาร์ตแสดงขบวนการแฮนด์เช็ค ในการส่งรหัสคำสั่งมาตรฐาน



รูปที่ 4.5 โฟล์วชาร์ตแสดงขบวนการแฮนด์เช็คในการส่งชุดข้อมูลคำสั่ง

4.5 สัญลักษณ์ที่ใช้บ่งบอกความหมายของชุดข้อมูล

เนื่องจากชุดข้อมูลที่จะส่งไปควบคุมอุปกรณ์เครื่องมือวัดต่างๆ นั้นจำเป็นจะต้องระบุถึง Address ของอุปกรณ์เครื่องมือวัดที่ต้องการควบคุมด้วย ดังนั้นจึงต้องมีการออกแบบรูปแบบของข้อมูลที่ส่งมาจากทางคอมพิวเตอร์ เพื่อที่จะให้สามารถสื่อสารกันได้เข้าใจ โดยในที่นี้ได้ออกแบบรูปแบบของชุดข้อมูลดังนี้

@[Address][XXX…..X][#]

โดยมีความหมายของรูปแบบข้อมูลดังนี้

@	เครื่องหมาย @ นั้นใช้บ่งบอกถึงจุดเริ่มต้นของชุดคำสั่ง
[Address]	ใช้ระบุ Address ของอุปกรณ์เครื่องมือวัดที่ต้องการติดต่อ
[XXX…..X]	เป็นส่วนของชุดคำสั่งที่ต้องการส่งไปยังไมโครคอนโทรลเลอร์
[#]	เครื่องหมาย # ใช้ในการบ่งบอกให้ไมโครคอนโทรลเลอร์ รับทราบว่าคำสั่งนั้นๆมีข้อมูลส่งกลับมาหรือไม่ ซึ่งหากในชุดคำสั่งมีสัญลักษณ์ # ต่อท้ายจะหมายถึงหลังจากที่ไมโครคอนโทรลเลอร์ส่งชุดคำสั่งนั้นๆไปแล้ว ให้ทำการรอรับคำสั่งที่ส่งกลับมาจากทางอุปกรณ์เครื่องมือวัดด้วย

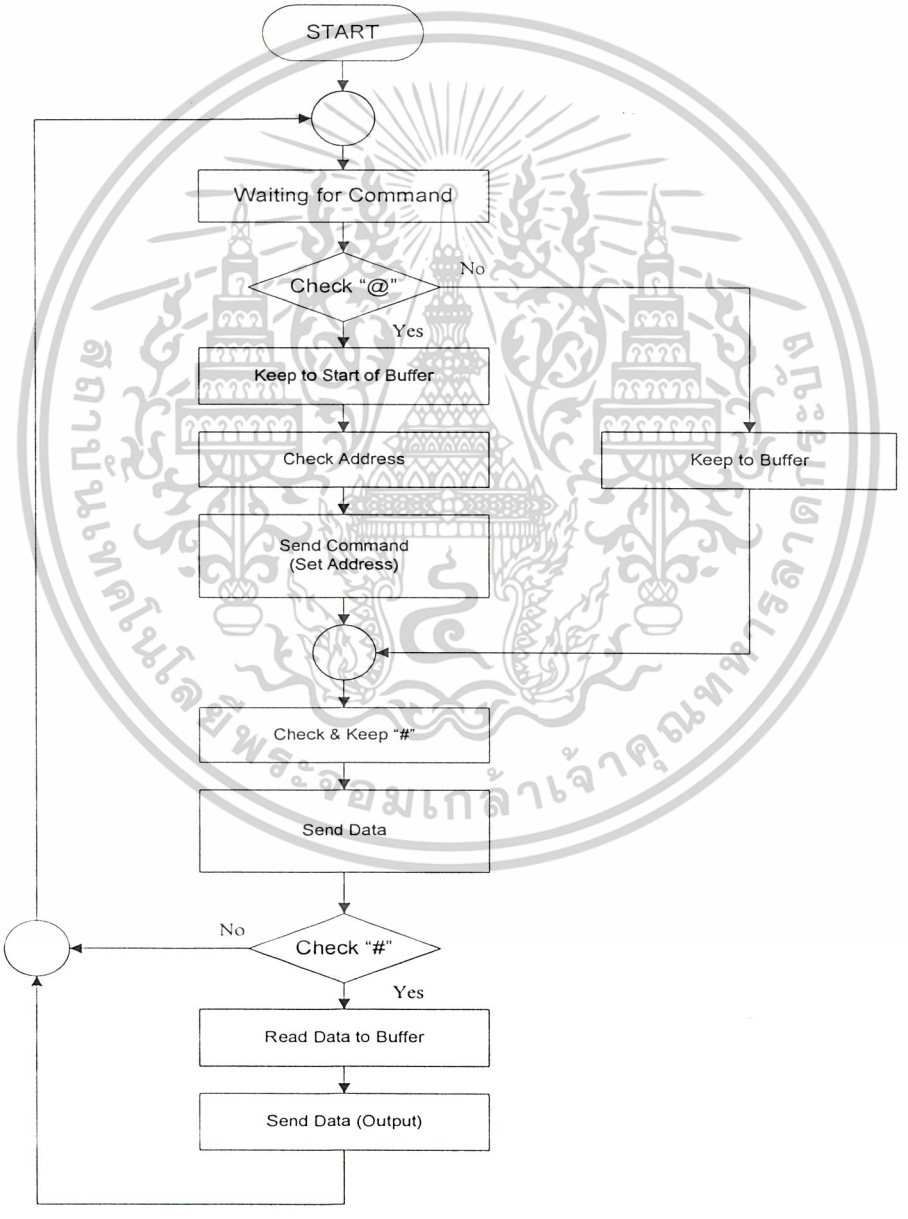
ตัวอย่างชุดข้อมูล

@1SU1 13.13	หมายถึงให้ทำการติดต่อไปยังอุปกรณ์ที่อยู่ใน Address ที่ 1 โดยคำสั่งที่ต้องการจะส่งไปคือ “ SU1 13.13 ” ซึ่งหลังจากการส่งคำสั่งแล้วจะไม่มีข้อมูลส่งกลับสามารถรอคำสั่งต่อไปได้ทันที
@7FRQ?#	หมายถึง ให้ทำการติดต่อไปยังอุปกรณ์ที่ต่ออยู่ใน Address ที่7 โดยคำสั่งที่ต้องการจะส่งไปคือ “FRQ?” ซึ่งหลังจากการส่งคำสั่งแล้ว ให้ทำการรอเพื่อรับค่าที่จะตอบรับกลับมาด้วย

4.6 โปรแกรมการทำงานของไมโครคอนโทรลเลอร์

ในส่วนการทำงานของไมโครคอนโทรลเลอร์นั้นแสดงตามรูปที่ 4.6 ซึ่งเป็นโฟลว์ชาร์ตแสดงการทำงานโดยรวมของไมโครคอนโทรลเลอร์ โดยการทำงานจะเริ่มต้นจากกการรอรับคำสั่งที่จะส่งมาจากทางคอมพิวเตอร์ ซึ่งคำสั่งที่ได้มานั้นจะทำการตรวจสอบว่าเป็นจุดเริ่มต้นของชุดคำสั่งหรือไม่โดยตรวจสอบจากเครื่องหมาย “@” โดยถ้าไม่พบเครื่องหมาย “@” ข้อมูลจะถูกนำไปเขียนต่อจากข้อมูลเดิมใน Buffer และจะนำข้อมูลนี้ไปรอที่จะทำการส่งต่อไป แต่ถ้าหากตรวจพบเครื่องหมาย “@” ข้อมูลจะถูกนำไปเขียนในจุดเริ่มต้นของ Buffer หลังจากนั้นจะเป็นการตรวจสอบ Address ว่าต้องการติดต่อไปยังอุปกรณ์ใด ซึ่ง Address จะส่งไปเพื่อกำหนดอุปกรณ์เครื่องมือวัดที่จะได้รับคำสั่งต่อไปนี้ โดยลำดับขั้นการส่งจะเป็นไปตามมาตรฐานการส่ง

ข้อมูลของ GPIB หลังจากนั้นข้อมูลใน Buffer จะทำการตรวจสอบเครื่องหมาย "# " ซึ่งหากพบเครื่องหมายนี้เงื่อนไข Check "# " ที่กล่าวถึงต่อไปนี้จะเป็นจริง และเครื่องหมายนี้จะนำออกจากชุดข้อมูลด้วย หลังจากนี้จะเป็นการส่งข้อมูลออกไปยัง Address ที่กำหนดไว้เบื้องต้นแล้ว ซึ่งการส่งนั้นจะเป็นไปตามมาตรฐานการส่งคำสั่งของ GPIB สำหรับค่า "# " ซึ่งก่อนหน้านี้ได้มีการตรวจสอบและเก็บค่าไว้นั้นหากไม่พบเครื่องหมาย "#" นั้นหมายถึงคำสั่งนี้ไม่ต้องการข้อมูลตอบกลับ แต่ถ้ามีการตรวจพบหลังจากการตรวจสอบชุดคำสั่งก็จะทำการรอรับค่าที่ส่งกลับมา เพื่อเก็บไว้ใน Buffer สำหรับค่าที่ได้รับมานั้นจะถูกส่งออกไปในรูปแบบข้อมูลอนุกรมเพื่อส่งให้แก่ทางคอมพิวเตอร์ต่อไป



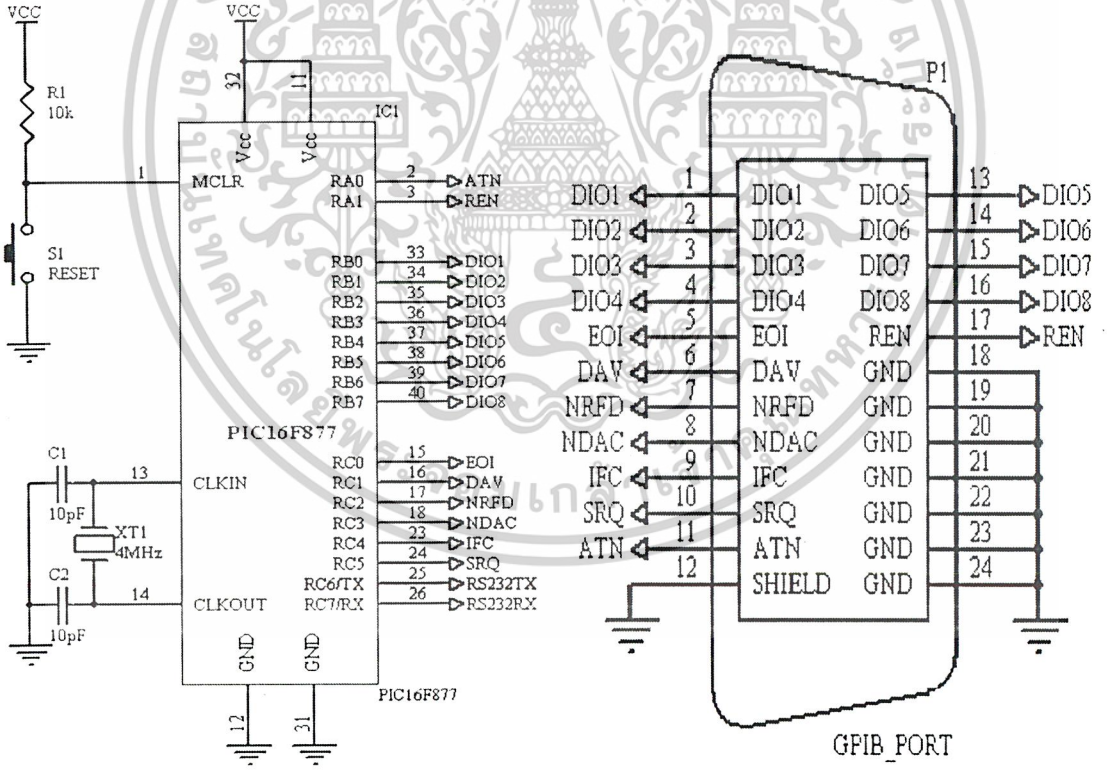
รูปที่ 4.6 โฟล์วชาร์ตแสดงการทำงานของไมโครคอนโทรลเลอร์

4.7 วงจรส่วนการอินเตอร์เฟส และหน้าที่การทำงานแต่ละส่วนประกอบ

ในส่วนนี้นั้นจะกล่าวถึงวงจรในส่วนต่าง ๆ ของส่วนอินเตอร์เฟสซึ่งแบ่งออกเป็นส่วนประกอบหลักๆ ดังต่อไปนี้

4.7.1 Microcontroller

ส่วนนี้ถือเป็นหัวใจสำคัญของระบบ เนื่องจากทำหน้าที่ในการจัดเรียงข้อมูลที่รับเข้ามาในรูปแบบข้อมูลอนุกรมเพื่อส่งออกไปให้แก่เครื่องมือวัดในมาตรฐาน GPIB หรือในทางกลับกันคือรับข้อมูลจากอุปกรณ์เครื่องมือวัดทางพอร์ต GPIB เพื่อส่งให้กับคอมพิวเตอร์ในรูปแบบข้อมูลอนุกรม ซึ่งในที่นี้ได้เลือกใช้ Microcontrollerเบอร์PIC16F877โดยมีคุณสมบัติในการจ่ายกระแสของแต่ละพอร์ตมากถึง 25mA (Sink/Source 25mA) ซึ่งเพียงพอที่จะจ่ายไปสั่งการอุปกรณ์เครื่องมือวัด ที่ต่อร่วมกันอยู่หลายๆ ตัวได้ อีกทั้งคุณสมบัติทางด้านความเร็วในการทำงาน โดยไมโครคอนโทรลเลอร์ในตระกูลนี้จะใช้สัญญาณนาฬิกาเพียง 1 หรือ 2 สัญญาณในการทำงาน 1 คำสั่ง ทำให้สามารถทำงานได้เร็วกว่าไมโครคอนโทรลเลอร์บางตระกูลที่ความถี่เดียวกัน



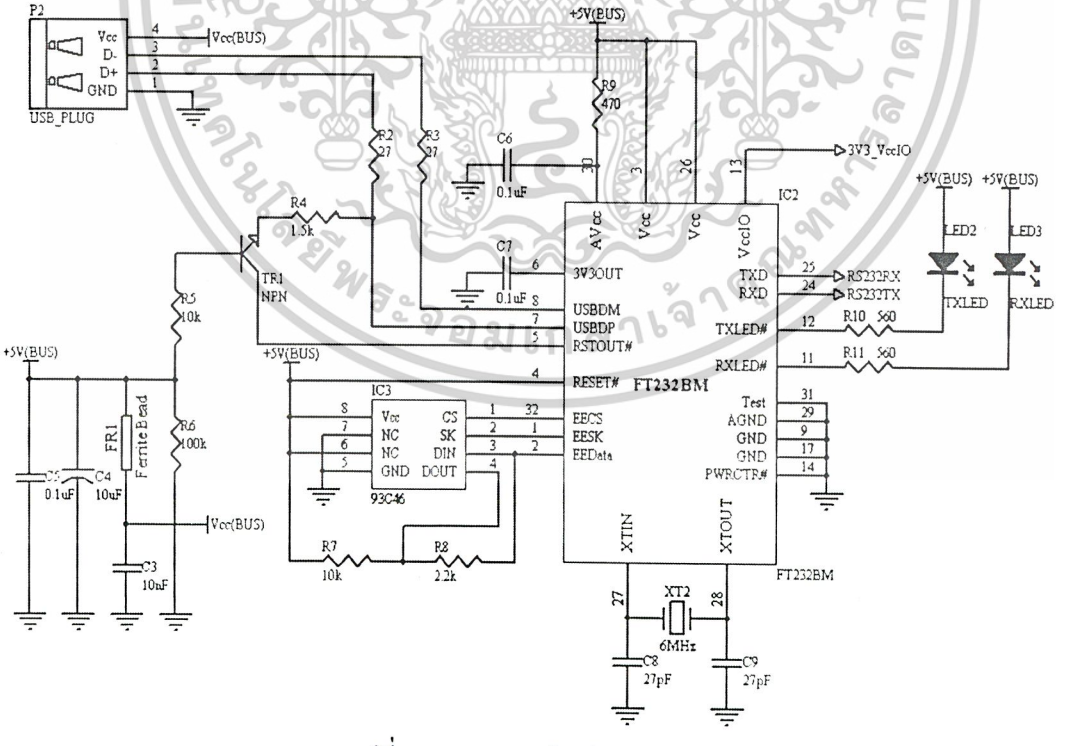
รูปที่ 4.7 แสดงการเชื่อมต่อระหว่างวงจรส่วนไมโครคอนโทรลเลอร์ และ พอร์ต GPIB

จากรูปที่ 4.7 จะเห็นได้ว่าพอร์ต B จะถูกใช้งานเป็นพอร์ต Data ในการส่งข้อมูล ซึ่งค่า Data ต่างๆ ที่จะส่งไปยัง GPIB นั้น ต้องรำลึกไว้เสมอว่าข้อมูลในมาตรฐาน GPIB นั้น จะเป็นในลักษณะตรงกันข้ามกับข้อมูลที่คอมพิวเตอร์รับรู้ ดังนั้นค่าที่ส่งและรับเข้ามานั้นต้องมีการกลับค่าก่อนเสมอ สำหรับสัญญาณอื่นๆ นั้นต่อร่วมกับพอร์ตต่างๆ ตามรูปวงจร และสำหรับในส่วนของการสื่อสารไปยังคอมพิวเตอร์ จะเป็นการสื่อสารผ่านทาง USART (Universal Synchronous Asynchronous Receiver Transmitter) ซึ่งเป็นพอร์ตสื่อสารอนุกรมในตัวไมโครคอนโทรลเลอร์เองโดยการติดต่อสื่อสารไปยังคอมพิวเตอร์โดย USB นั้นจะกล่าวถึงต่อไป

4.7.2 USB Converter

สำหรับส่วนนี้นั้นเป็นส่วนที่ทำหน้าที่ในการเปลี่ยนแปลงการสื่อสารอนุกรมจากไมโครคอนโทรลเลอร์ไปสู่การทางสื่อสารผ่านระบบบัส USB ซึ่งในที่นี้นั้นได้เลือกใช้ IC เบอร์ FT232BM ของบริษัท Future Technology Devices International Limited :FTDI ซึ่งมีลักษณะการส่งถ่ายข้อมูลภายในแบบ FIFO โดย IC เบอร์นี้มีความสามารถในการจัดการกับโปรโตคอล USB ได้อย่างสมบูรณ์ โดยไม่ต้องอาศัยเฟิร์มแวร์(Firmware) ในการติดต่อกับระบบบัส USB เพิ่มเติม

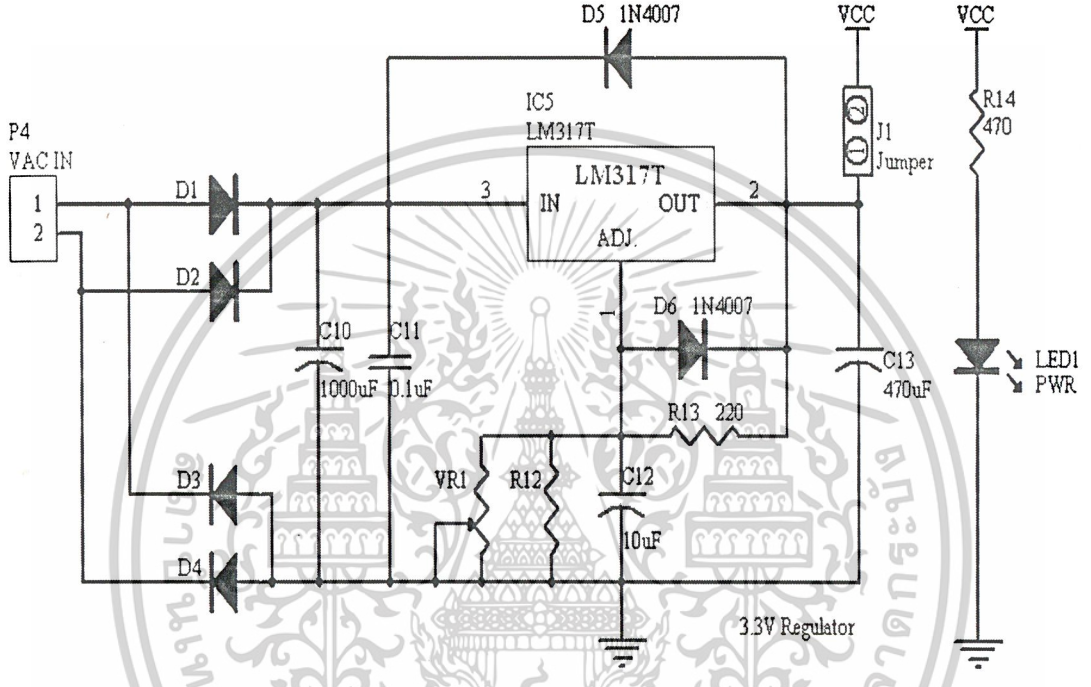
สำหรับในการติดต่อกับโปรแกรมที่เขียนขึ้นนั้น จะใช้ไดร์เวอร์พอร์ตอนุกรมเสมือน (Virtual COM Port : VCP) ซึ่งทำให้การเชื่อมต่อผ่านทาง USB ถูกมองเป็นการเชื่อมต่อในลักษณะของพอร์ตอนุกรม (COM) ทำให้มีความสะดวกในการเขียนโปรแกรมติดต่อไปยังพอร์ต USB เป็น อย่างมาก



รูปที่ 4.8 แสดงวงจรในส่วนของ USB Convertor

4.7.3 Regulator

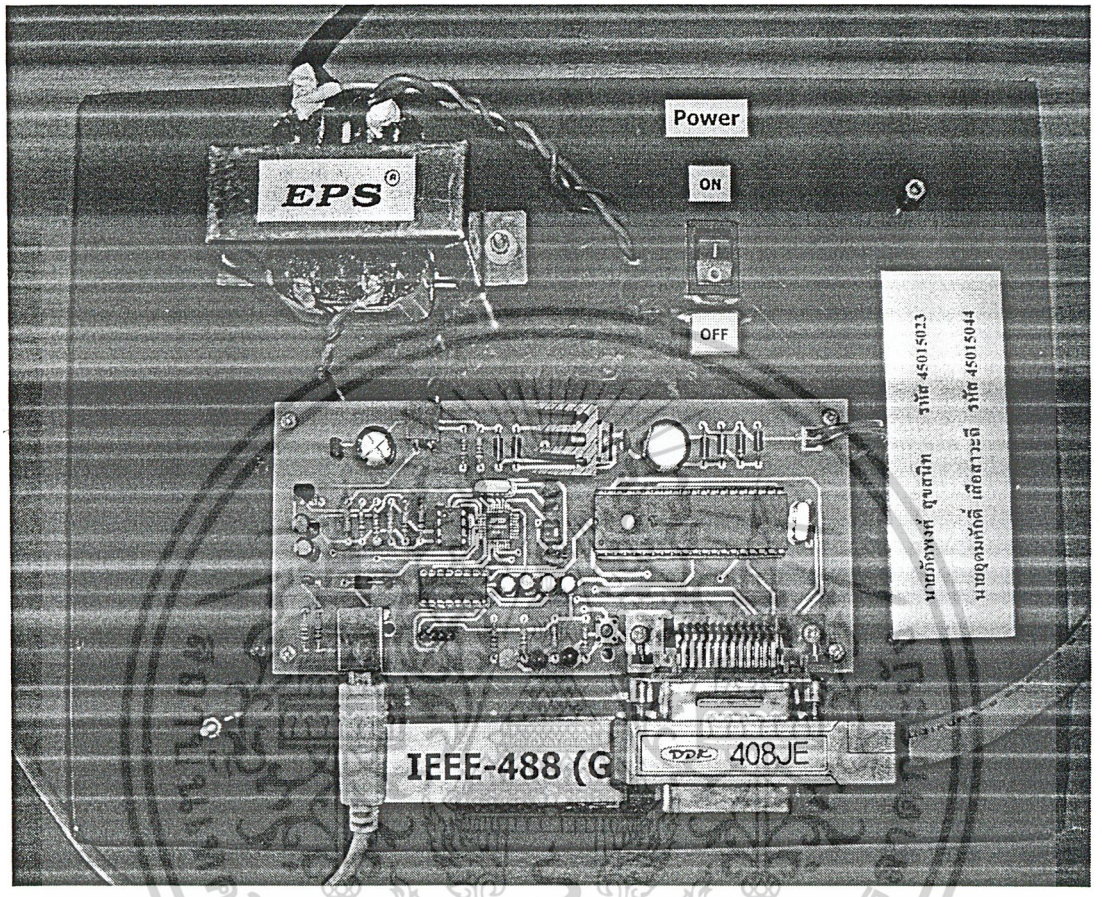
ในส่วนนี้ทำหน้าที่ในการสร้างแรงดัน 3.3 V ซึ่งในที่นี้เราใช้ IC เบอร์ LM317T เป็นตัวสร้างแรงดันเพื่อนำไปใช้ในส่วนของ USB Converter



รูปที่ 4.9 วงจรในส่วนของ Regulator

จากส่วนประกอบทั้งหมดเมื่อนำมาเขียนเรียงกันในแผ่น PCB เดียวกันก็จะสามารถนำมาเป็นส่วนของฮาร์ดแวร์ (Hardware) ที่ใช้ในโครงงานชิ้นนี้

จากส่วนประกอบต่างๆ ที่ได้กล่าวมานั้น ได้ถูกออกแบบไว้บนแผ่น PCB เดียวกันเพื่อทำหน้าที่เป็นการ์ดอินเตอร์เฟสดังรูปที่ 4.10



รูปที่ 4.10 แสดงการ์ดอินเตอร์เฟสที่ได้ทำการออกแบบ

บทที่ 5
การทดลองและผลการทดลอง

ในการทดสอบส่วนควบคุมการอินเตอร์เฟสนั้น ใช้คำสั่งที่เขียนขึ้นจากโปรแกรม Microsoft Visual C++ ซึ่งเขียนขึ้นเพื่ออินเตอร์เฟสกับฮาร์ดแวร์โดยผ่านพอร์ต USB

5.1 Address ของอุปกรณ์เครื่องมือวัดที่นำมาทดสอบ

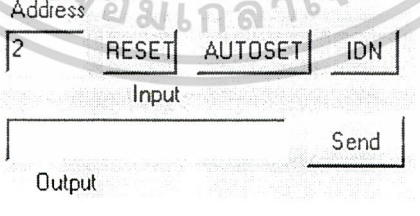
ดังที่ได้กล่าวไว้แล้วว่าในการสั่งการคำสั่งใด ๆ จำเป็นจะต้องระบุ Address ของอุปกรณ์ของเครื่องมือวัดที่ต้องการสั่งการ ซึ่งการส่งคำสั่งแต่ละคำสั่งก็จะมี Address เหล่านี้กำหนดอยู่ด้วยดังนั้นในการเขียนโปรแกรมเพื่อรองรับกับอุปกรณ์ใด ๆ จึงต้องทราบถึง Address ของอุปกรณ์เหล่านั้นเสียก่อน โดยในการทดลองนี้นั้นจะกำหนด Address ของอุปกรณ์ ดังนี้

อุปกรณ์	Address
Oscilloscope ยี่ห้อ Tektronix รุ่น TDS3032	2
Function Generator ยี่ห้อ Sony Tektronix รุ่น AFG310	3

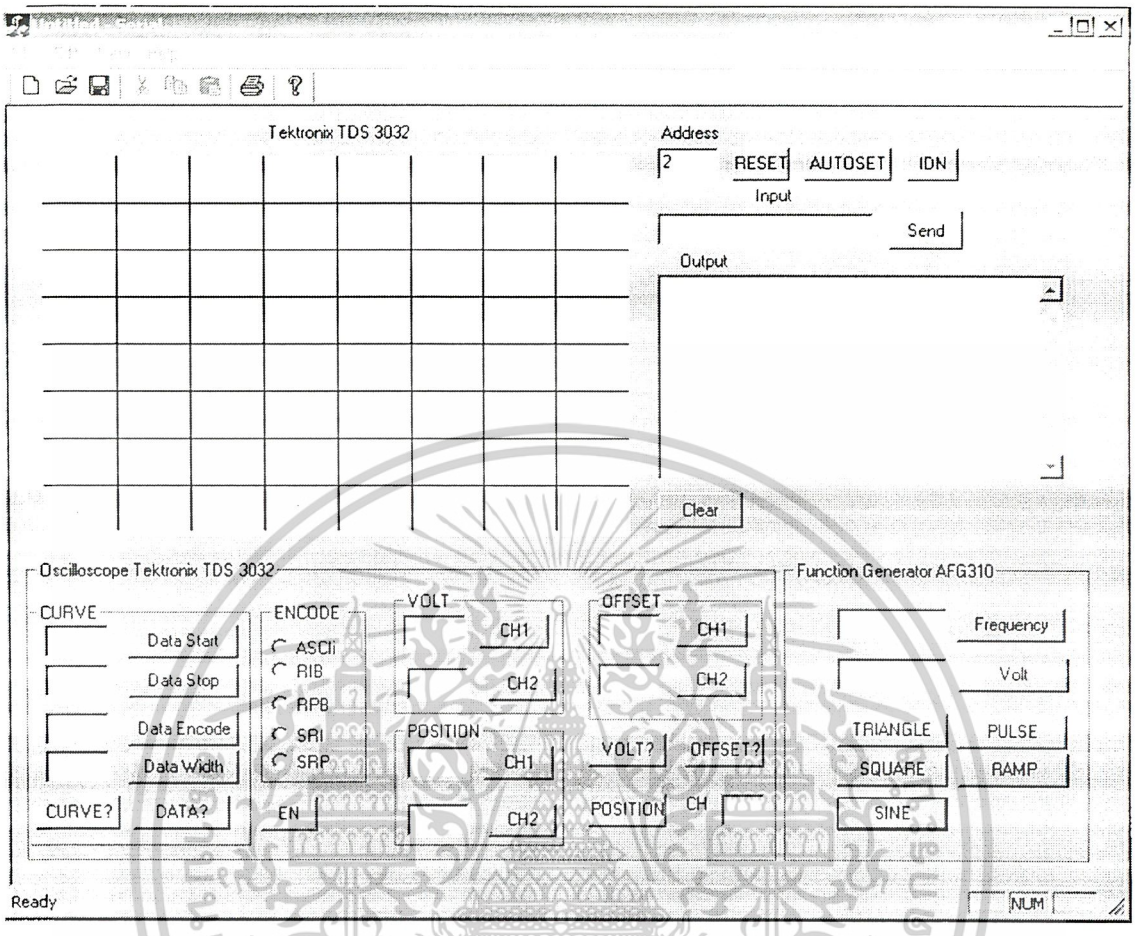
5.2 ทำการทดลองโดยใช้โปรแกรมสั่งการที่เขียนขึ้นจากโปรแกรม Microsoft Visual C++

ขั้นตอนการใช้งาน

1. ต้องทำการกำหนด Address ของอุปกรณ์ที่จะทำการสั่งงานก่อนโดยปรับไปที่ Address 2 เมื่อต้องการใช้ Oscilloscope



รูปที่ 5.1 การกำหนด Address ของอุปกรณ์เพื่อสั่งงาน Oscilloscope

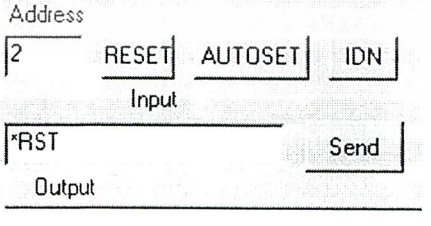


รูปที่ 5.2 ซอร์ฟแวร์ที่เขียนขึ้นเพื่อใช้ในการสั่งงานเครื่องมือวัด

2. ป้อนคำสั่งเพื่อทำการสั่งงาน Oscilloscope ยี่ห้อ Tektronix รุ่น TDS3032

คำสั่ง Reset

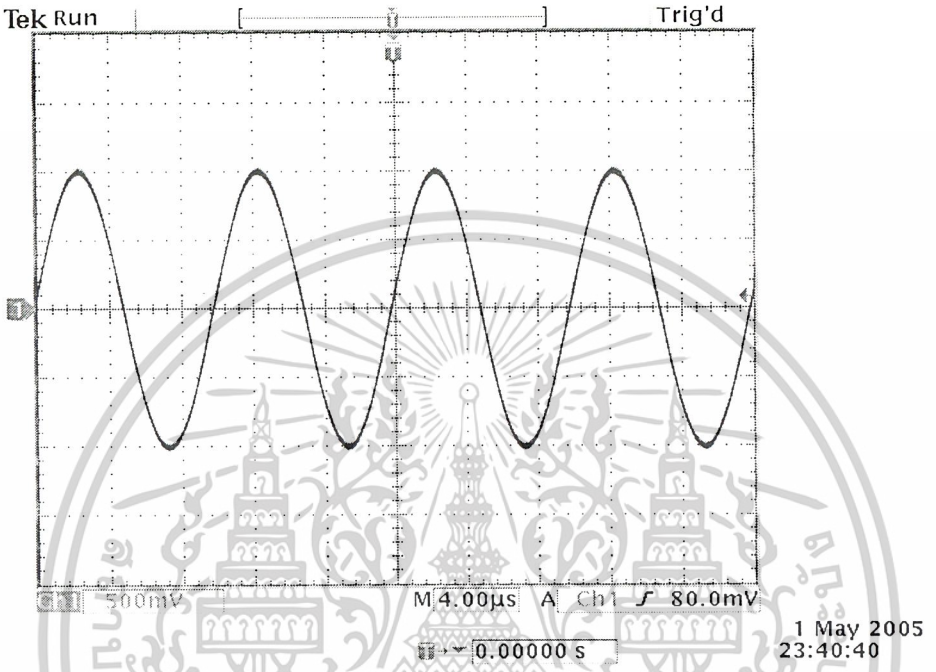
รูปแบบคำสั่ง : *RST



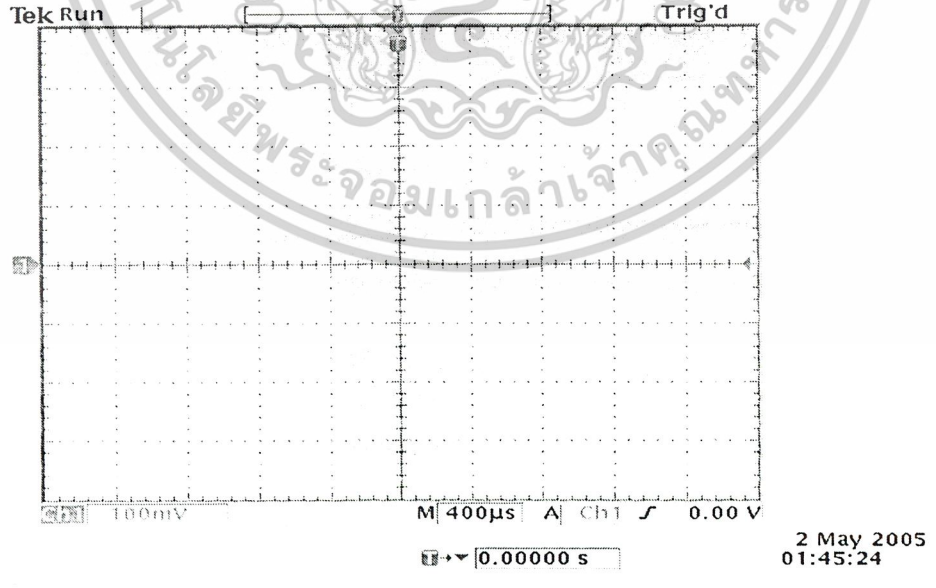
รูปที่ 5.3 ทำการสั่ง คำสั่ง Reset

จากโปรแกรมที่เขียนขึ้นทำการ Reset ได้ 2 วิธีคือ

- คลิกเมาส์ปุ่ม RESET
- พิมพ์คำว่า * RST แล้วคลิกที่ปุ่ม Send



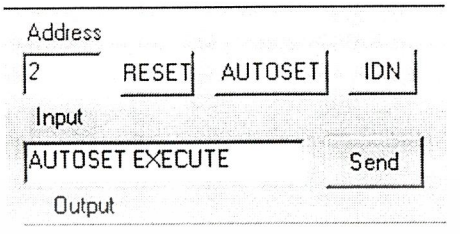
รูปที่ 5.4 รูปคลื่นก่อนสั่งคำสั่ง * RST



รูปที่ 5.5 รูปคลื่นหลังสั่งคำสั่ง *RST

คำสั่ง Auto Set

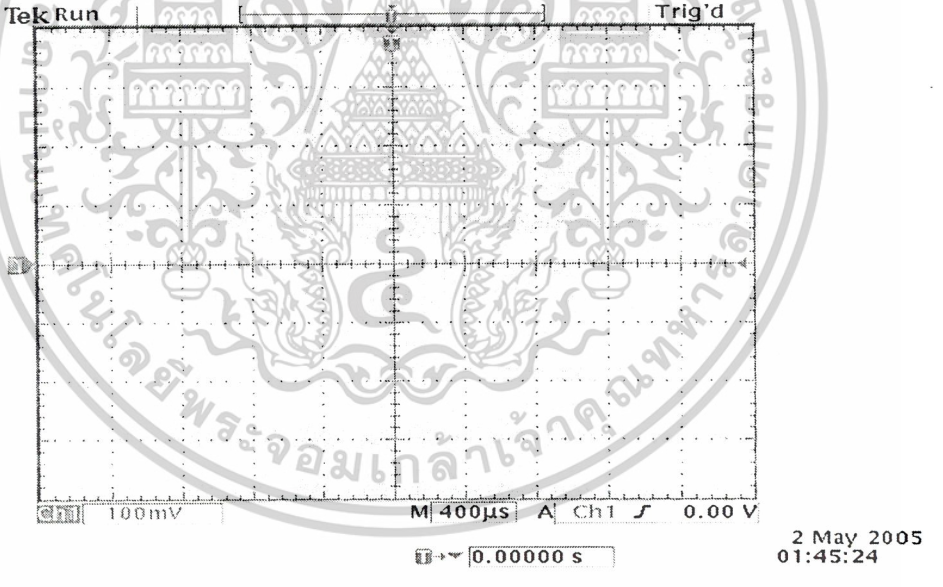
รูปแบบคำสั่ง : AUTOSET EXECUTE



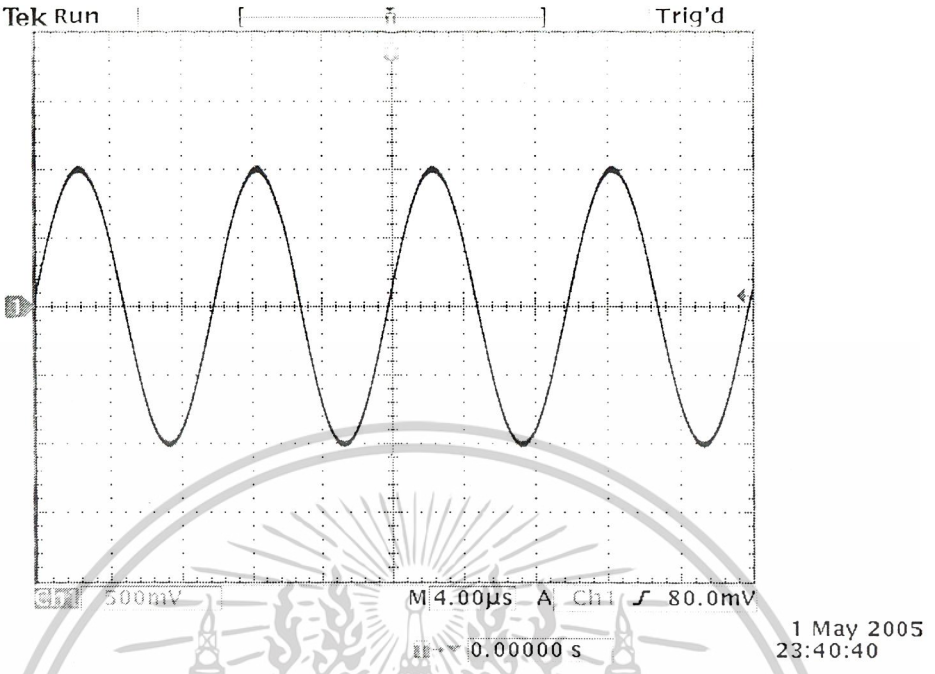
รูปที่ 5.6 ทำการสั่ง คำสั่ง Reset

จากโปรแกรมที่เขียนขึ้นทำการ Auto Set ได้ 2 วิธีคือ

- คลิกเมาส์ปุ่ม AUTOSET
- พิมพ์คำว่า AUTOSET EXECUTE แล้วคลิกที่ปุ่ม Send



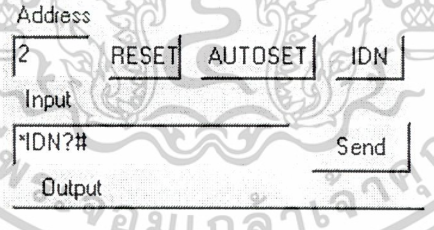
รูปที่ 5.7 รูปคลื่นก่อนสั่งคำสั่ง AUTOSET EXECUTE



รูปที่ 5.8 รูปคลื่นหลังสั่งคำสั่ง AUTOSET EXECUTE

คำสั่ง ถามรายละเอียดเบื้องต้นของเครื่องมือวัด

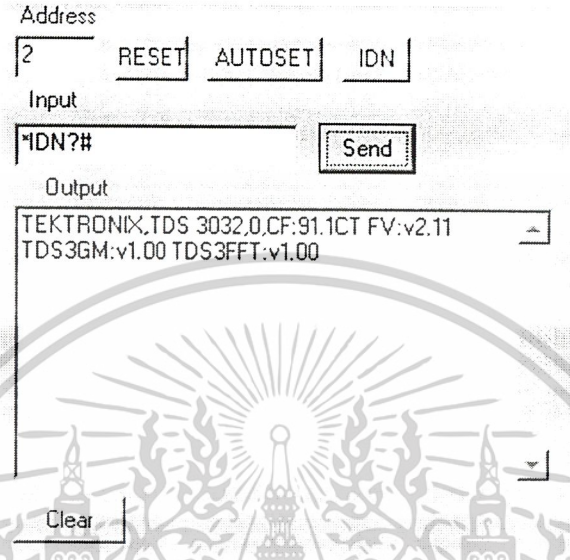
รูปแบบคำสั่ง : *IDN?



รูปที่ 5.9 ทำการสั่งคำสั่งถามรายละเอียดเบื้องต้นของเครื่องมือวัด

จากโปรแกรมที่เขียนขึ้นทำการถามรายละเอียดเบื้องต้นของเครื่องมือวัดได้ 2 วิธีคือ

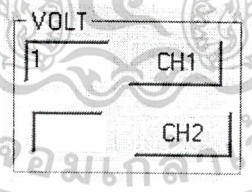
- คลิกเมาส์ปุ่ม IDN
- พิมพ์คำว่า *IDN?# แล้วคลิกที่ปุ่ม Send เนื่องจากว่าต้องมีการส่งค่ากลับจากเครื่องมือวัดจึงต้องใส่เครื่องหมาย # ตามหลังด้วย



รูปที่ 5.10 ผลที่ได้จากคำสั่งถามรายละเอียดเบื้องต้นของเครื่องมือวัด

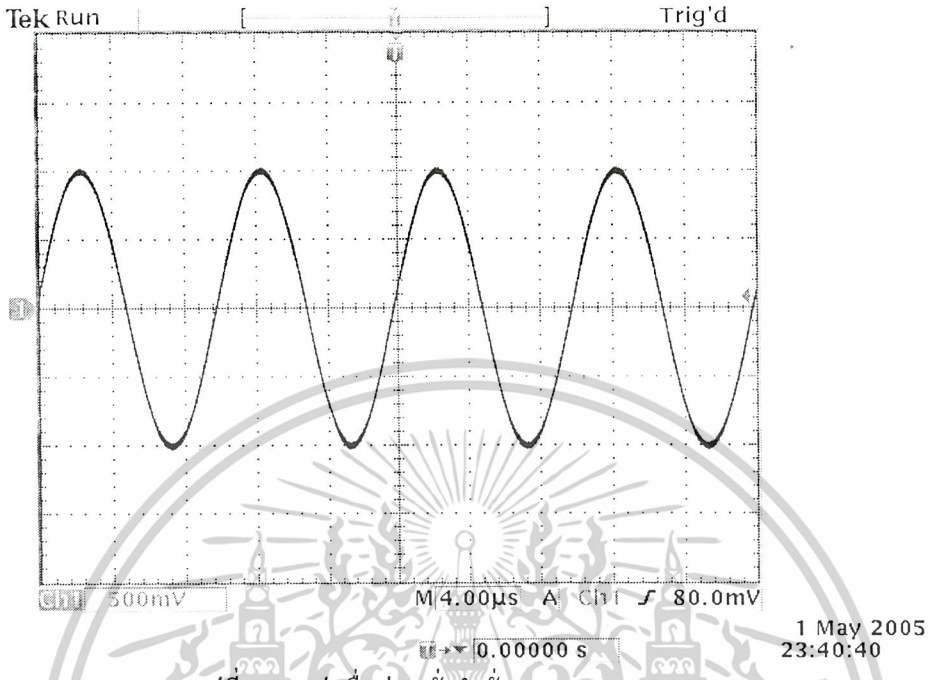
คำสั่ง ปรับ **Channel Volt Div**

รูปแบบคำสั่ง : CH<x> :VOLTS <Number>

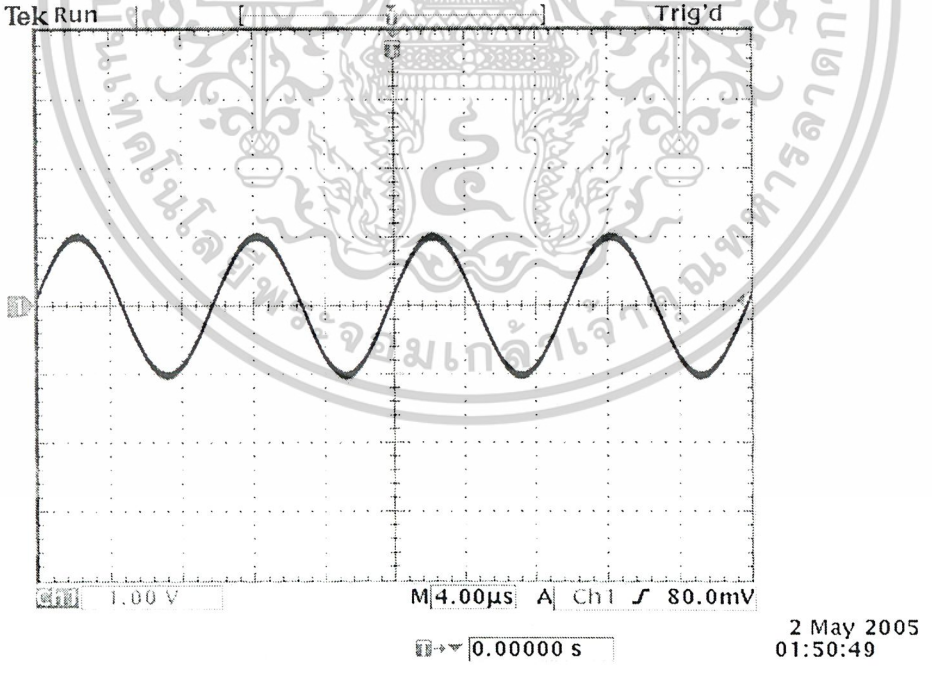


รูปที่ 5.11 ทำการสั่ง คำสั่ง Channel Volt Div

- ตามรูปที่ 5.11 ทำการใส่ตัวเลขที่ช่องด้านบนแล้วคลิกที่ปุ่ม CH1 เพื่อทำการปรับ Volt Div ของ Channel 1 ให้เป็น 1V



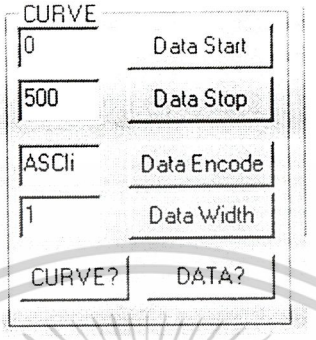
รูปที่ 5.12 รูปคลื่นก่อนสั่งคำสั่ง Channel Volt Div 1V



รูปที่ 5.13 รูปคลื่นหลังสั่งคำสั่ง Channel Volt Div 1V

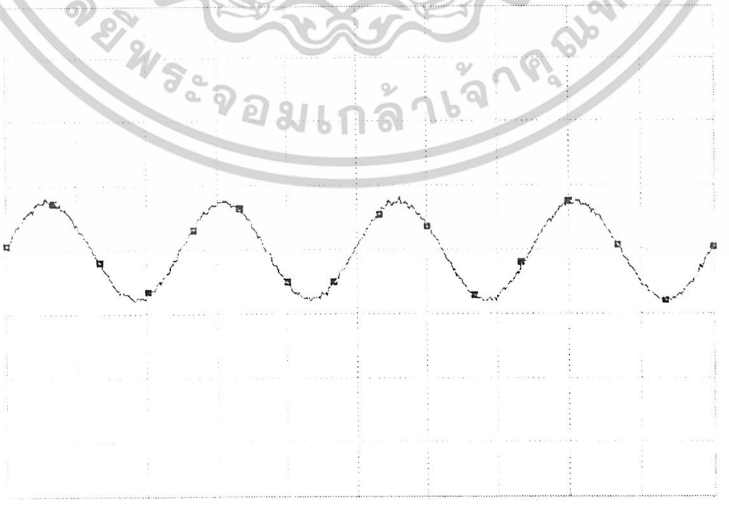
คำสั่ง Plot Curve

รูปแบบคำสั่ง : CURVE?



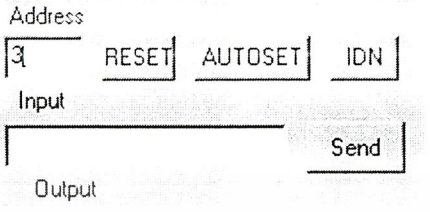
รูปที่ 5.14 ค่าพารามิเตอร์ของรูปคลื่น

- ทำการตั้งค่าพารามิเตอร์ของค่าที่ต้องการจะนำมาพล็อตกราฟ
  - ตั้งค่าเริ่มต้นของข้อมูลที่ Data Start
  - ตั้งค่าสิ้นสุดของข้อมูลที่ Data Stop
  - ตั้งค่ารูปแบบของข้อมูลที่ Data Encode
  - ตั้งค่าจำนวนความกว้างของข้อมูลที่ Data Width
- คลิกที่ปุ่ม CURVE? เพื่อนำค่าที่ได้จากหน้าจอของออสซิลโลสโคปกลับมาพล็อตกราฟค่าที่นำกลับมาพล็อตกราฟเป็นค่าที่ได้จากรูปคลื่นในรูปที่ 5.13 โดยนำมาพล็อตที่ซอร์ฟแวร์



รูปที่ 5.15 ผลที่ได้จากการนำค่าจากออสซิลโลสโคปกลับมาพล็อตกราฟ

3. ต้องทำการกำหนด Address ของอุปกรณ์ที่จะทำการสั่งงานก่อนโดยปรับไปที่ Address 3 เมื่อต้องการใช้ Function Generator ยี่ห้อ Sony Tektronix รุ่น AFG310

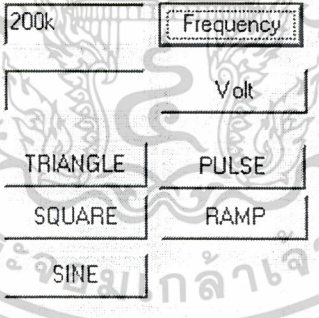


รูปที่ 5.16 การกำหนด Address ของอุปกรณ์เพื่อสั่งงาน Function Generator

4. ป้อนคำสั่งเพื่อทำการสั่งงาน Function Generator ยี่ห้อ Sony Tektronix รุ่น AFG310

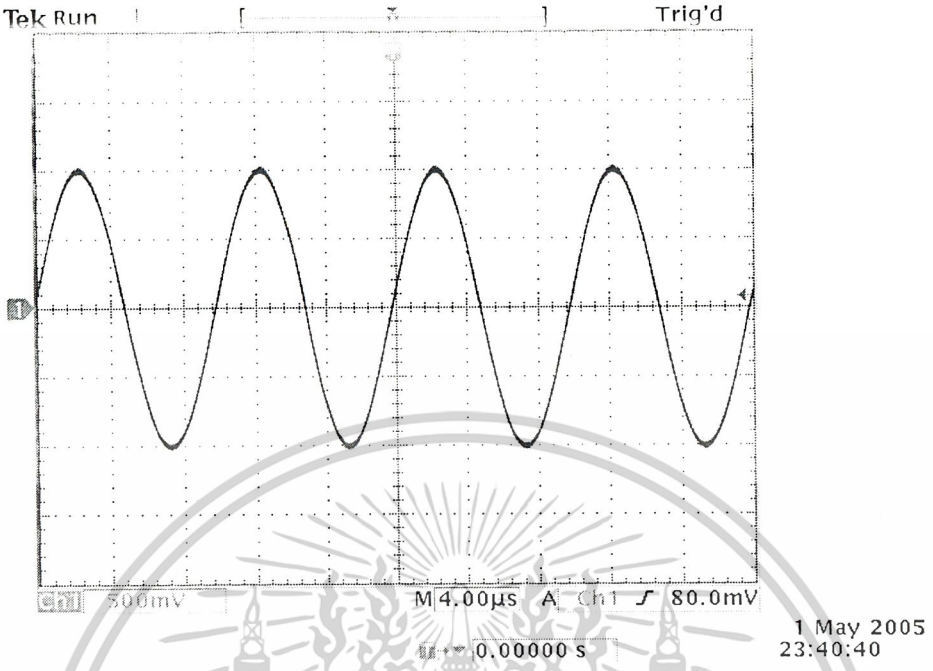
คำสั่ง ปรับความถี่

รูปแบบคำสั่ง : SOURCE<x>:FREQUENCY:FIXED <Number>Hz

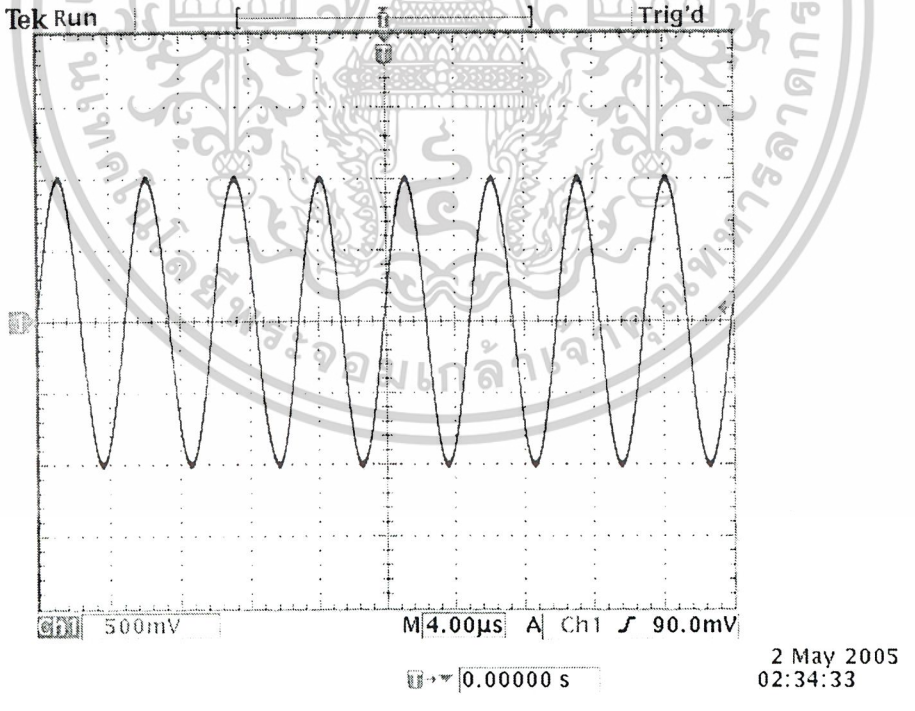


รูปที่ 5.17 ทำการสั่ง คำสั่งปรับความถี่

- พิมพ์ค่าความถี่ที่ต้องการลงในช่องว่างแล้วคลิกที่ปุ่ม Frequency



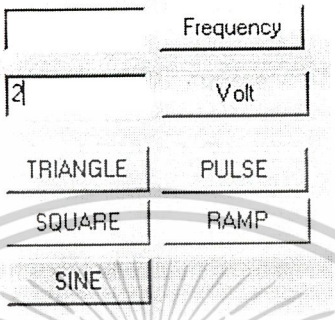
รูปที่ 5.18 ผลที่ได้ก่อนสั่งคำสั่งปรับความถี่



รูปที่ 5.19 ผลที่ได้หลังสั่งคำสั่งปรับความถี่

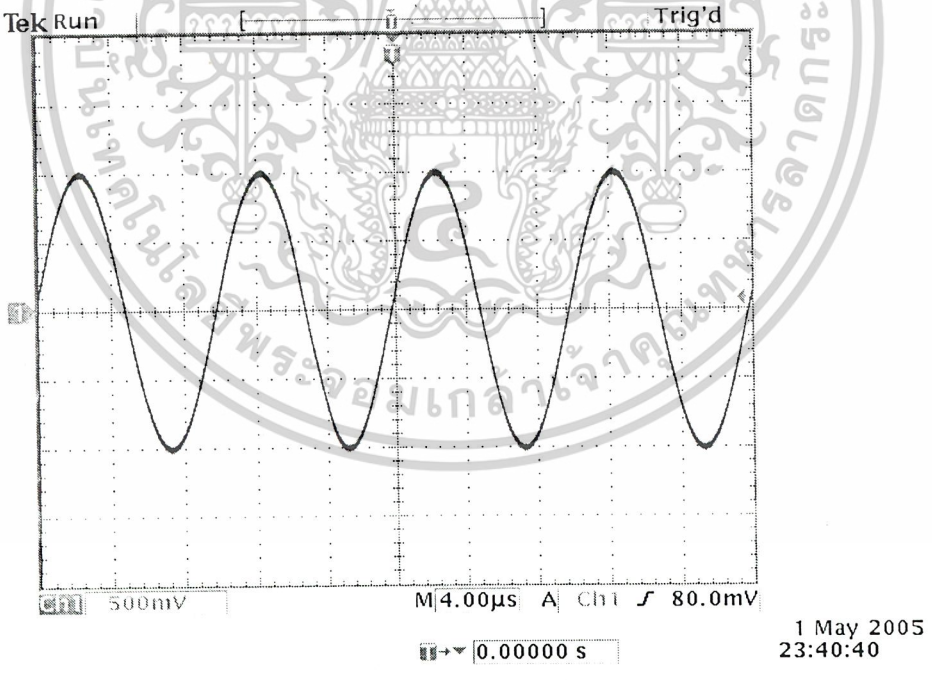
คำสั่ง โวลต์

รูปแบบคำสั่ง : SOURCE<x>:VOLTAGE:LEVEL:IMMEDIATE:OFFSET <Number>V

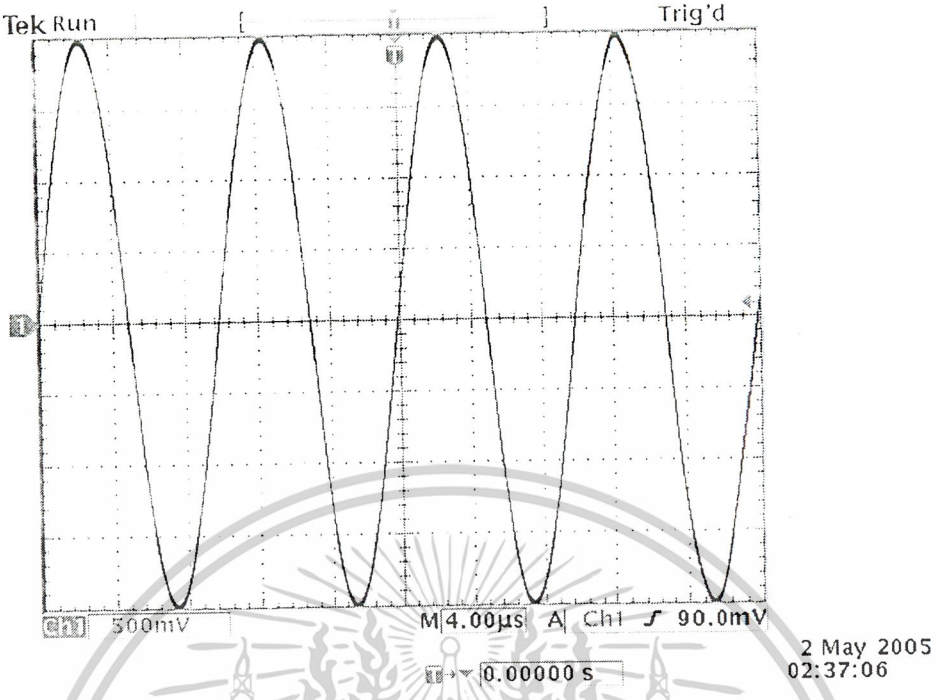


รูปที่ 5.20 ทำการสั่ง คำสั่งปรับโวลต์

- พิมพ์ค่าโวลต์ที่ต้องการลงในช่องว่างแล้วคลิกที่ปุ่ม Volt



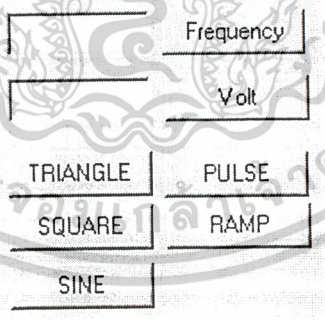
รูปที่ 5.21 ผลที่ได้ก่อนสั่งคำสั่งปรับโวลต์



รูปที่ 5.22 ผลที่ได้หลังสั่งคำสั่งปรับโวลต์

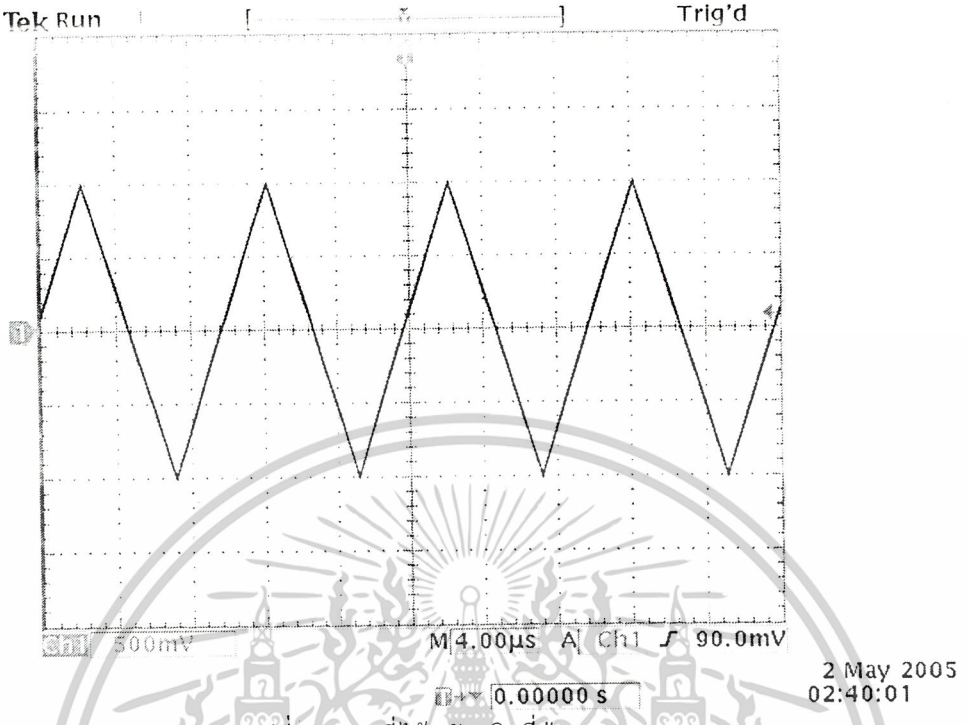
คำสั่ง ปรับรูปคลื่น

รูปแบบคำสั่ง : SOURCE<x>:FUNCTION:SHAPE <Type of shape>



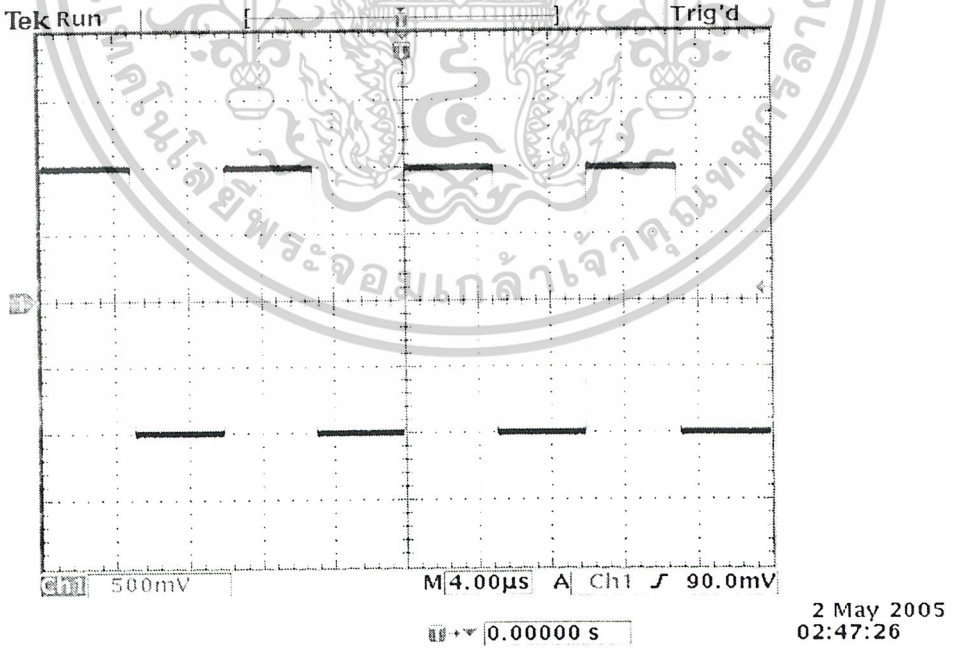
รูปที่ 5.23 ทำการสั่ง คำสั่งปรับโวลต์

- คลิกที่ปุ่ม TRIANGLE เพื่อทำการปรับรูปคลื่นให้เป็นแบบ 3 เหลี่ยม



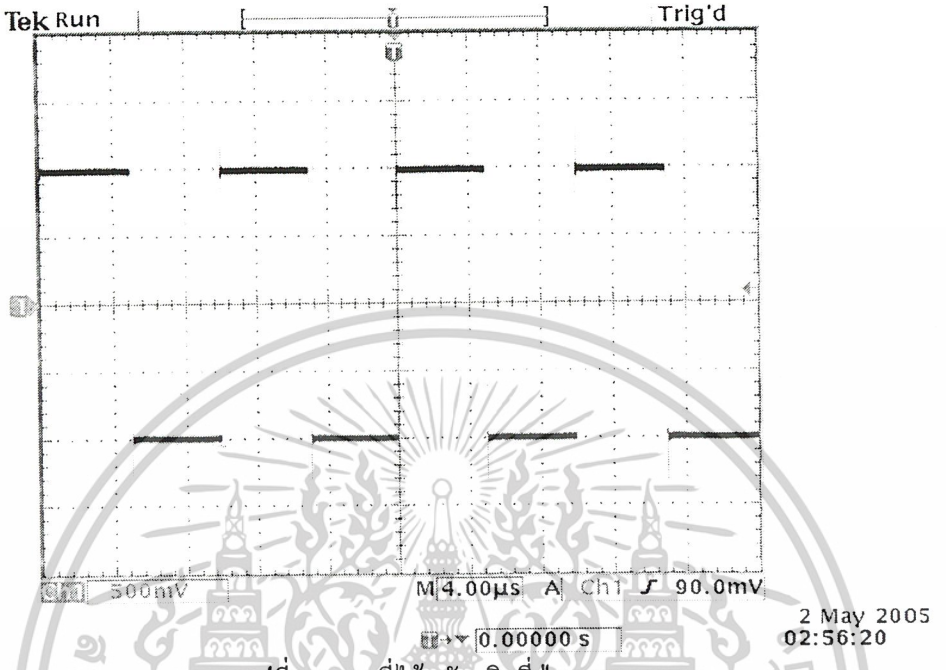
รูปที่ 5.24 ผลที่ได้หลังคลิกที่ปุ่ม TRIANGLE

- คลิกที่ปุ่ม PULSE เพื่อทำการปรับรูปคลื่นให้เป็นแบบพัลส์



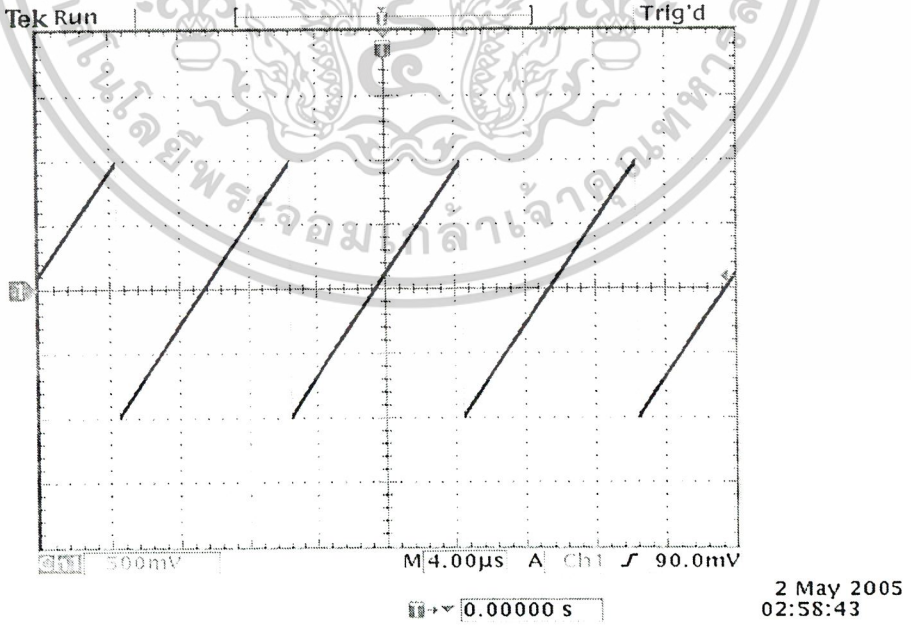
รูปที่ 5.25 ผลที่ได้หลังคลิกที่ปุ่ม PULSE

- คลิกที่ปุ่ม SQUARE เพื่อทำการปรับรูปคลื่นให้เป็นแบบสี่เหลี่ยม



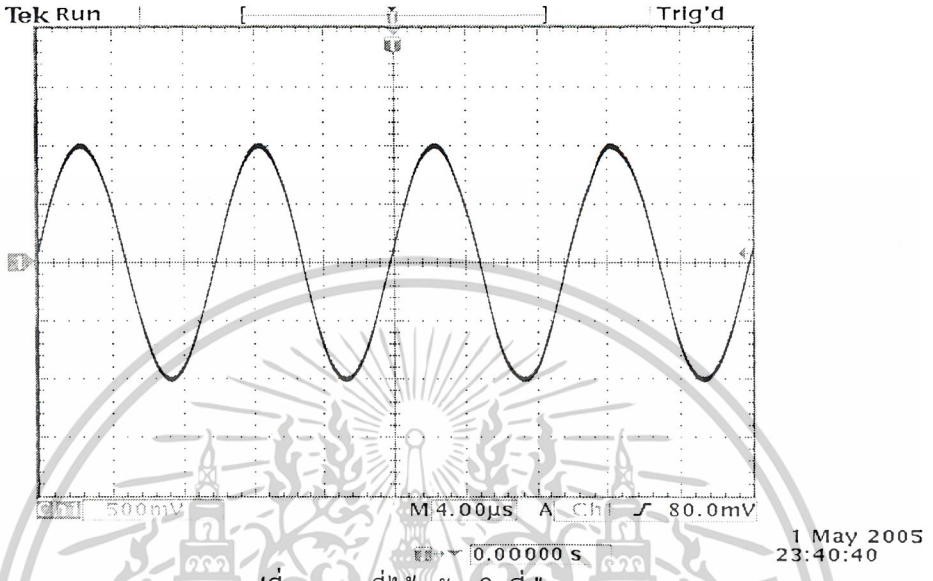
รูปที่ 5.26 ผลที่ได้หลังคลิกที่ปุ่ม SQUARE

- คลิกที่ปุ่ม RAMP เพื่อทำการปรับรูปคลื่นให้เป็นแบบฟันเลื่อย

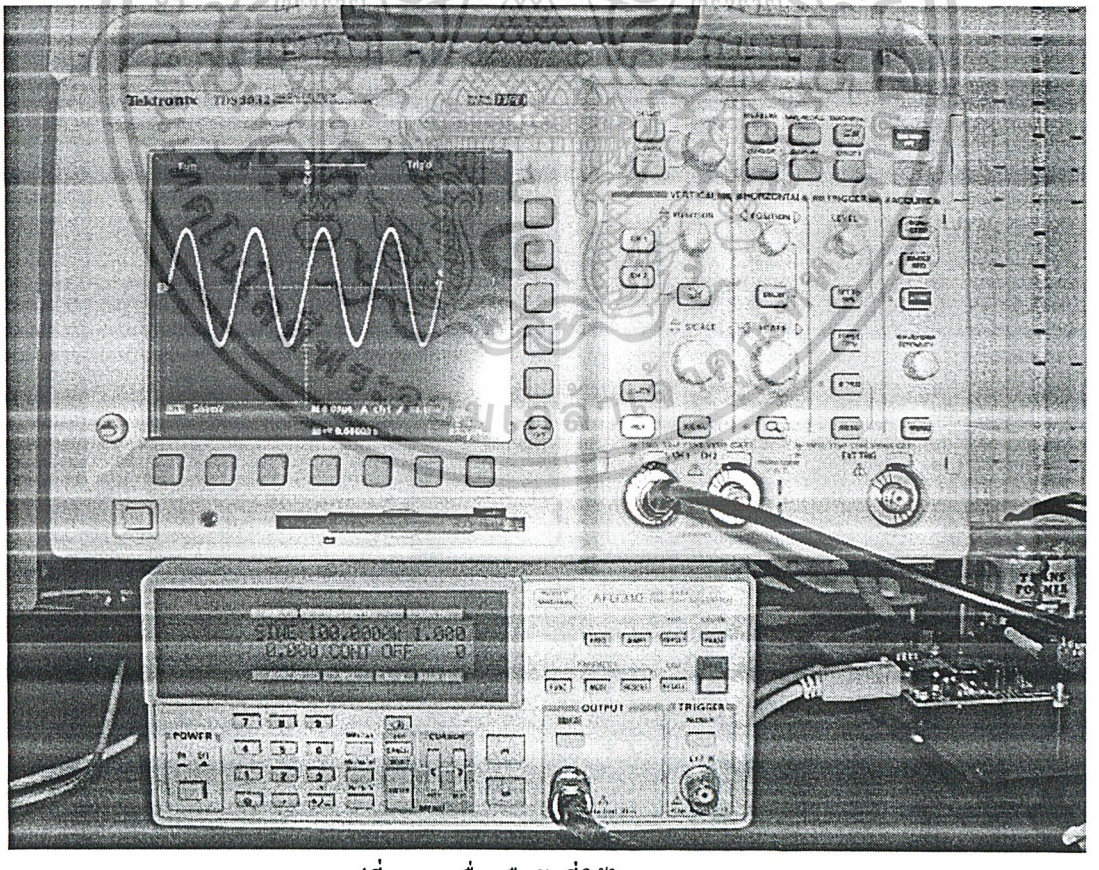


รูปที่ 5.27 ผลที่ได้หลังคลิกที่ปุ่ม RAMP

- คลิกที่ปุ่ม SINE เพื่อทำการปรับรูปคลื่นให้เป็นแบบซายน์



รูปที่ 5.28 ผลที่ได้หลังคลิกที่ปุ่ม SINE



รูปที่ 5.29 เครื่องมือวัดที่ใช้ในการทดลอง

บทที่ 6
บทสรุปและวิจารณ์

จากการทดลองควบคุมเครื่องมือวัดโดยใช้ซอร์ฟแวร์จากคอมพิวเตอร์เป็นตัวควบคุมนั้นจะเห็นว่าสามารถประยุกต์ใช้งานได้อย่างหลากหลาย ซึ่งโปรแกรมที่เขียนขึ้นเพื่อทดสอบการทำงานนี้เป็นเพียงส่วนหนึ่งของการใช้งานเท่านั้น ซึ่งในทางปฏิบัติจะสามารถออกแบบโปรแกรมให้รองรับกับรูปแบบของงานตามต้องการ รวมถึงการออกแบบให้เป็นระบบควบคุมอัตโนมัติด้วย

ผลการทดลองได้ทดสอบกับเครื่องมือวัด Oscilloscope ยี่ห้อ Tektronix รุ่น TDS3032 และ Function Generator ยี่ห้อ Sony Tektronix รุ่น AFG310 จะเห็นว่าสามารถใช้งานได้เป็นอย่างดี และนอกจากนี้ยังสามารถใช้โปรแกรม Labview ซึ่งใช้งานร่วมกับเครื่องมือวัดได้เป็นอย่างดีมาใช้ในการควบคุมเครื่องมือวัดด้วย

ข้อควรระวังในการใช้งานการ์ดอินเตอร์เฟส ในขณะที่ทำการติดตั้งสาย GPIB เข้ากับการ์ดอินเตอร์เฟสนั้น ควรพึงระวังไว้เสมอว่าอุปกรณ์เครื่องมือวัดต่าง ๆ รวมถึงส่วนของการ์ดอินเตอร์เฟสนั้น ควรจะอยู่ในสภาวะ OFF ทั้งหมด เนื่องจากการ์ดอินเตอร์เฟสนี้ได้ใช้พอร์ตของไมโครคอนโทรลเลอร์ต่อเพื่อควบคุมอุปกรณ์เครื่องมือวัดโดยตรง ซึ่งพอร์ตของไมโครคอนโทรลเลอร์นี้มีข้อจำกัดทางด้านกระแสที่ไม่สูงนัก (25mA) ดังนั้นในการเชื่อมต่อพอร์ตของไมโครคอนโทรลเลอร์ เข้ากับอุปกรณ์เครื่องมือวัดโดยตรงในขณะที่อุปกรณ์เครื่องมือวัดทำงานอยู่นั้น สภาวะลอจิกที่คงค้างอยู่ของแต่ละอุปกรณ์อาจก่อให้เกิดกระแสเฉียบพลันซึ่งอาจมีค่าสูงเพียงพอที่จะทำให้พอร์ตของไมโครคอนโทรลเลอร์เสียหายได้

กิตติกรรมประกาศ

ขอขอบพระคุณ รศ.ดร. กอบชัย เดชหาญ ที่คอยช่วยสนับสนุนและให้การดูแลเป็นอย่างดี และอาจารย์สมปอง วิเศษพานิชกิจ ที่ช่วยให้ความรู้คำแนะนำ และสนับสนุนในทุก ๆ ด้าน ขอขอบพระคุณอาจารย์และเจ้าหน้าที่ภาควิชาวิศวกรรมโทรคมนาคมที่มีส่วนช่วยเหลือในปริญญานิพนธ์นี้ให้สำเร็จลุล่วงด้วยดี ขอขอบพระคุณพ่อ แม่ และเพื่อน ๆ ที่คอยเป็นกำลังใจและช่วยเหลือในการทดลองการทำงานของโครงงานนี้

บรรณานุกรม

[1] วรเทพ ไพบูลย์รัตนากร, บุญอนันต์ เกียงเอีย , "สัมผัสโลก USB," บริษัท แอสทรอน ลอจิกรีเสิร์ช แอนด์ดิวิลอปเมนต์ จำกัด, 2537

[2] National Instrument Corporate Headquarters, "GPIB-232/485CT-A User Manual," 1995

[3] HAMEG Instruments, "HO88 Manual," 2002

[4] Tektronix Instruments, "Tektronix TDS 3032 Manual," 2001

[5] นภดล มณีรัตน์ "การออกแบบและสร้างระบบการส่งข้อมูลแบบขนานสำหรับการควบคุมเครื่องมือวัดผ่านระบบบัสมาตรฐาน IEEE-488 (GPIB)," วิทยานิพนธ์บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2540

[6] บุญญฤทธิ์ ลักษณประณัย, ดร. จารุวงศ์รังสี, นครินทร์ รัตนมณีศิลป์, "การออกแบบและสร้างระบบควบคุมเครื่องมือวัดด้วยมาตรฐาน IEEE-488 (GPIB)," ปริญญานิพนธ์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2546

[7] นิรุธ อำนวยศิลป์, "คู่มือการเขียนโปรแกรม Visual C++ 6.0," บริษัท ซัคเซส มีเดีย จำกัด, 2521

ภาคผนวก

The FT232BM is the 2nd generation of FTDI's popular USB UART i.c. This device not only adds extra functionality to it's FT8U232AM 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 ⇔Asynchronous Serial Data Transfer
- Full Handshaking & Modem Interface Signals
- UART I/F Supports 7 / 8 Bit Data, 1 / 2 Stop Bits and Odd/Even/Mark/Space/No Parity
- Data rate 300 => 3M Baud (TTL)
- Data rate 300 => 1M Baud (RS232)
- Data rate 300 => 3M Baud (RS422/RS485)
- 384 Byte Receive Buffer / 128 Byte Transmit Buffer for high data throughput
- Adjustable RX buffer timeout
- Full hardware assisted hardware or X-On / X-Off handshaking
- In-built support for event characters and line break condition
- Auto Transmit Buffer control for RS485
- Support for USB Suspend / Resume through SLEEP# and RI# pins
- Support for high power USB Bus powered devices through PWREN# pin
- Integrated level converter on UART 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 Isochronous data transfer modes
- 4.35V 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 32-LD LQFP 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

- USB ⇔RS232 Converters
- USB ⇔ RS422 / RS485 Converters
- Upgrading RS232 Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA ⇔ USB data transfer
- USB Smart Card Readers
- Set Top Box (S.T.B.) PC - USB interface
- USB Hardware Modems
- USB Wireless Modems
- USB Instrumentation
- USB Bar Code Readers

[** = In planning or under development]

2.0 Enhancements

This section summarises the enhancements of the 2nd generation device compared to it's FT8U232AM 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 simply be hard wired to VCC. In addition, a new reset output pin (RSTOUT#) is provided in order to allow the new POR circuit to provide a stable reset to external MCU and other devices. RSTOUT# 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 UART interface and control signals**
  The previous devices would drive the UART 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

- **Improved Power Management control for USB Bus Powered, high current devices**
  The previous devices had a USBEN pin, which became active when the device was enumerated by USB. To provide power control, this signal had to be externally gated with SLEEP# and RESET#. This gating is now done on-chip - USBEN has now been replaced with the new PWREN# signal 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 UART 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 circuitry controlled by PWREN# resets reliably when power is restored.

- **Lower Suspend Current**
  Integration of RCCLK within the device and internal design improvements reduce the suspend current of the FT232BM to under 200uA (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 Isochronous 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 Isochronous transfer via an option bit in the EEPROM.

- **Programmable Receive Buffer Timeout**
  In the previous device, the receive buffer timeout used to flush remaining data from the receive 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.

- **TXDEN Timing fix**
  TXDEN timing has now been fixed to remove the external delay that was previously required for RS485 applications at high baud rates. TXDEN now works correctly during a transmit send-break condition.

- **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.

- **Improved PreScaler Granularity**
  The previous version of the Prescaler supported division by (n + 0), (n + 0.125), (n + 0.25) and (n + 0.5) where n is an integer between 2 and 16,384 (2^{14}). To this we have added (n + 0.375), (n + 0.625), (n + 0.75) and (n+ 0.875) which can be used to improve the accuracy of some baud rates and generate new baud rates which were previously impossible (especially with higher baud rates).

- **Bit Bang Mode**
  The 2nd generation device has a new option referred to as "Bit Bang" mode. In Bit Bang mode, the eight UART interface control lines can be switched between UART 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 the prescaler setting. 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 FT232BM can switch back into UART 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 parties.

- **PreScaler Divide By 1 Fix**
  The previous device had a problem when the integer part of the divisor was set to 1. In the 2nd generation device setting the prescaler value to 1 gives a baud rate of 2 million baud and setting it to zero gives a baud rate of 3 million baud. Non-integer division is not supported with divisor values of 0 and 1.

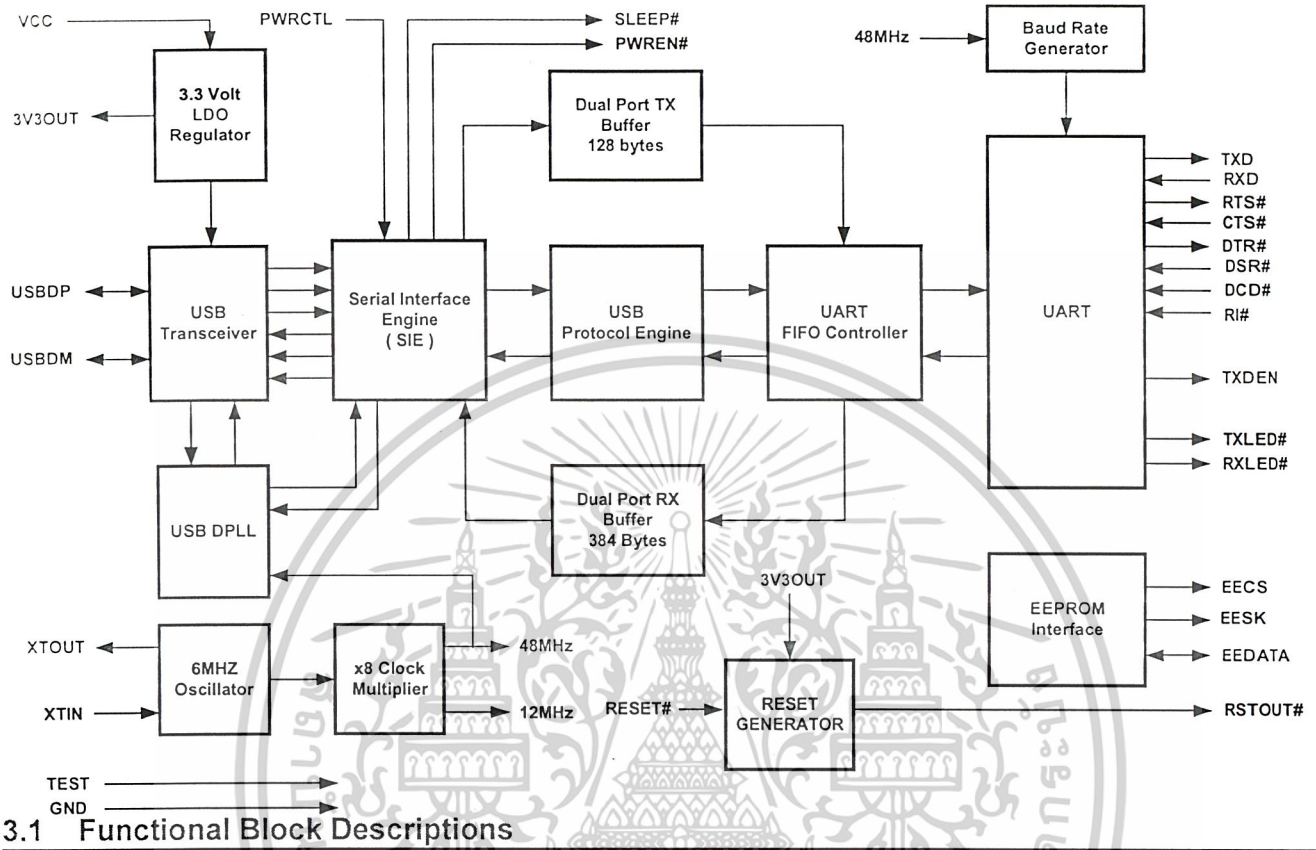
- **Less External Support Components**
  As well as eliminating the RCCLK RC network, and for most applications the need for an external reset circuit, we have also eliminated the requirement for a 100k pull-up on EECS to select 6MHz operation. When the FT232BM is being used without the configuration EEPROM, EECS, EESK and EEDATA can now be left n/c. For circuits requiring a long reset time (where the device is reset externally using a reset generator i.c., or

reset is controlled by the IO port of a MCU, FPGA or ASIC device) an external transistor circuit is no longer required as the 1.5k pull-up resistor on USB DP can be wired to the RSTOUT# pin instead of to 3.3V. Note : RSTOUT# drives out at 3.3V level, not at 5V VCC level. This is the preferred configuration for new designs.

- **Extended EEPROM Support**
  The previous generation of devices only supported EEPROM of type 93C46 (164 x 16 bit). The new devices will also work with EEPROM type 93C56 (128 x 16 bit) and 93C66 (256 x 16 bit). The extra space is not used by the device, however it is available for use by other external MCU / logic whilst the FT232BM is being held in reset.
- **USB 2.0 (full speed option)**
  A new EEPROM based option allows the FT232BM to return a USB 2.0 device descriptor as opposed to USB 1.1. Note : The device would be a USB 2.0 Full Speed device (12Mb/s) as opposed to a USB 2.0 High Speed device (480Mb/s).

- **Multiple Device Support without EEPROM**
  When no EEPROM (or a blank or invalid EEPROM) is attached to the device, the FT232BM no longer gives a serial number as part of it's USB descriptor. This allows multiple devices to be simultaneously connected to the same PC. However, we still highly recommend that EEPROM is used, as without serial numbers a device can only be identified by which hub port in the USB tree it is connected to which can change if the end user re-plugs the device into a different port.

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 UART FIFO controller blocks. It also generates a 48MHz reference clock for the USB DPPL and the Baud Rate Generator blocks.

- **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 USB 2.0 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 UART.

- **Dual Port TX Buffer (128 bytes)**

  Data from the USB data out endpoint is stored in the Dual Port TX buffer and removed from the buffer to the UART transmit register under control of the UART FIFO controller.

- **Dual Port RX Buffer (384 bytes)**

  Data from the UART receive register is stored in the Dual Port RX buffer prior to being removed by the SIE on a USB request for data from the device data in endpoint.

- **UART FIFO Controller**

  The UART FIFO controller handles the transfer of data between the Dual Port RX and TX buffers and the UART transmit and receive registers.

- **UART**

  The UART performs asynchronous 7 / 8 bit Parallel to Serial and Serial to Parallel conversion of the data on the RS232 (RS422 and RS485) interface. Control signals supported by the UART include RTS, CTS, DSR , DTR, DCD and RI.
  The UART provides a transmitter enable control signal (TXDEN) to assist with interfacing to RS485 transceivers. The UART supports RTS/ CTS, DSR/DTR and X-On/X-Off handshaking options. Handshaking, where required, is handled in hardware to ensure fast response times. The UART also supports the RS232 BREAK setting and detection conditions.

- **Baud Rate Generator**

  The Baud Rate Generator provides a x16 clock input to the UART from the 48MHz reference clock and consists of a 14 bit prescaler and 3 register bits which provide fine tuning of the baud rate (used to divide by a number plus a fraction). This determines the Baud Rate of the UART which is programmable from 183 baud to 3 million baud.

- **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 FT232BM or the FT232BM to reset other devices respectively. During reset, RSTOUT# is driven low, otherwise it drives out at the 3.3V provided by the onboard regulator. RSTOUT# can be used to control the 1.5k 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 FT232BM 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 FT232BM for OEM applications. 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 = 4.35V 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 FT232BM 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.

4.0 Device Pin-Out

Figure 1
Pin-Out
(LQFP-32 Package)

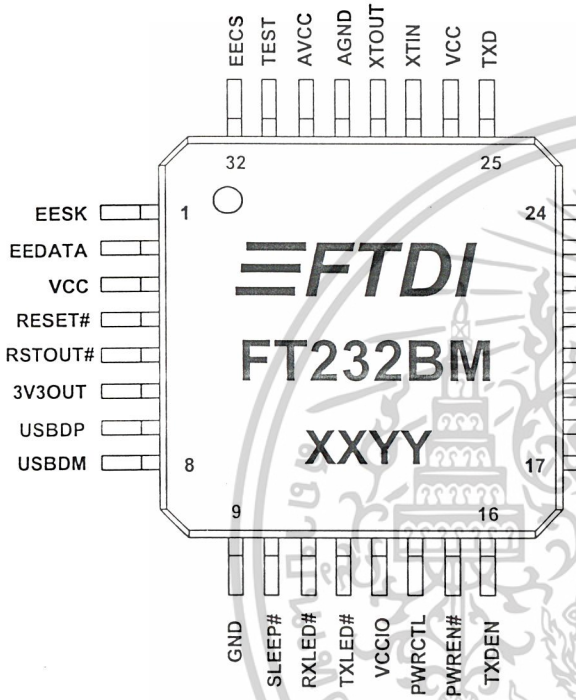
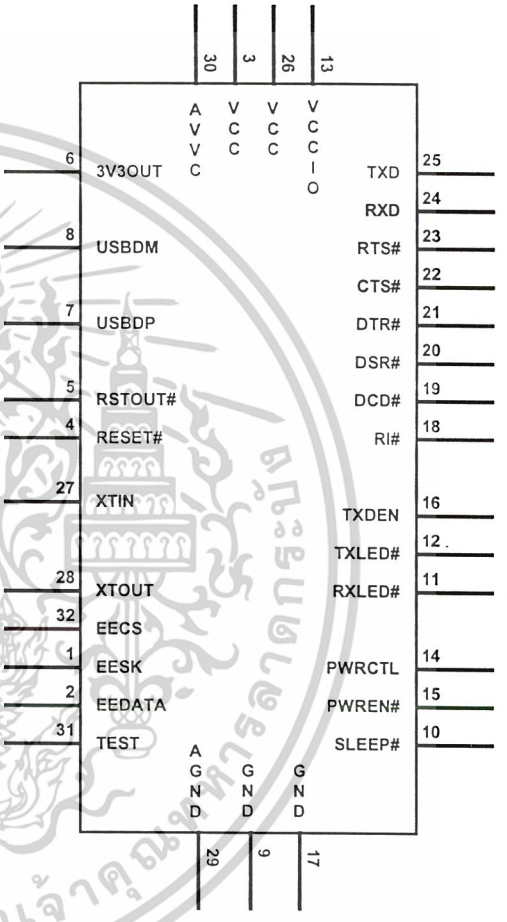


Figure 2
Pin-Out
(Schematic Symbol)



4.1 Signal Descriptions

Table 1 - FT232BM - PINOUT DESCRIPTION

UART INTERFACE GROUP

Pin#	Signal	Type	Description
25	TXD	OUT	Transmit Asynchronous Data Output
24	RXD	IN	Receive Asynchronous Data Input
23	RTS#	OUT	Request To Send Control Output / Handshake signal
22	CTS#	IN	Clear To Send Control Input / Handshake signal
21	DTR#	OUT	Data Terminal Ready Control Output / Handshake signal
20	DSR#	IN	Data Set Ready Control Input / Handshake signal
19	DCD#	IN	Data Carrier Detect Control Input
18	RI#	IN	Ring Indicator Control Input. When the Remote Wakeup option is enabled in the EEPROM, taking RI# low can be used to resume the PC USB Host controller from suspend.
16	TXDEN	OUT	Enable Transmit Data for RS485

USB INTERFACE GROUP

Pin#	Signal	Type	Description
7	USBDP	I/O	USB Data Signal Plus (Requires 1.5k pull-up to 3V3OUT or RSTOUT#)
8	USBDM	I/O	USB Data Signal Minus

EEPROM INTERFACE GROUP

Pin#	Signal	Type	Description
32	EECS	I/O	EEPROM – Chip Select. For 48MHz operation pull EECS to GND using a 10k resistor. For 6MHz operation no resistor is required. Tri-State during device reset. **Note 1
1	EESK	OUT	Clock signal to EEPROM. Tri-State during device reset, else drives out. **Note 1
2	EEDATA	I/O	EEPROM – Data I/O Connect directly to Data-In of the EEPROM and to Data-Out of the EEPROM via a 2.2k resistor. Also, pull Data-Out of the EEPROM to VCC via a 10k resistor for correct operation. Tri-State during device reset. **Note 1

POWER CONTROL GROUP

Pin#	Signal	Type	Description
10	SLEEP#	OUT	Goes Low during USB Suspend Mode. Typically used to power-down an external TTL to RS232 level converter i.c. in USB -> RS232 converter designs.
15	PWREN#	OUT	Goes Low after the device is configured via USB, then high during USB suspend. Can be used to control power to external logic using a P-Channel Logic Level MOSFET switch. Enable the Interface Pull-Down Option in EEPROM when using the PWREN# pin in this way.
14	PWRCTL	IN	Bus Powered – Tie Low / Self Powered – Tie High (to VCCIO)

MISCELLANEOUS SIGNAL GROUP

Pin#	Signal	Type	Description
4	RESET#	IN	Can be used by an external device to reset the FT232BM. If not required, tie to VCC.
5	RSTOUT#	OUT	Output of the internal Reset Generator. Stays high impedance for ~ 5ms after VCC > 3.5V and the internal clock starts up, then clamps it's output to the 3.3v output of the internal regulator. Taking RESET# low will also force RSTOUT# to drive low. RSTOUT# is NOT affected by a USB Bus Reset.
12	TXLED#	O.C.	LED Drive - Pulses Low when Transmitting Data via USB
11	RXLED#	O.C.	LED Drive - Pulses Low when Receiving Data via USB
27	XTIN	IN	Input to 6MHz Crystal Oscillator Cell. This pin can also be driven by an external 6MHz clock if required. Note : Switching threshold of this pin is VCC/2, so if driving from an external source, the source must be driving at 5V CMOS level or a.c. coupled to centre around VCC/2.
28	XTOUT	OUT	Output from 6MHz Crystal Oscillator Cell. XTOUT stops oscillating during USB suspend, so take care if using this signal to clock external logic.
31	TEST	IN	Puts device in i.c. test mode – must be tied to GND for normal operation.

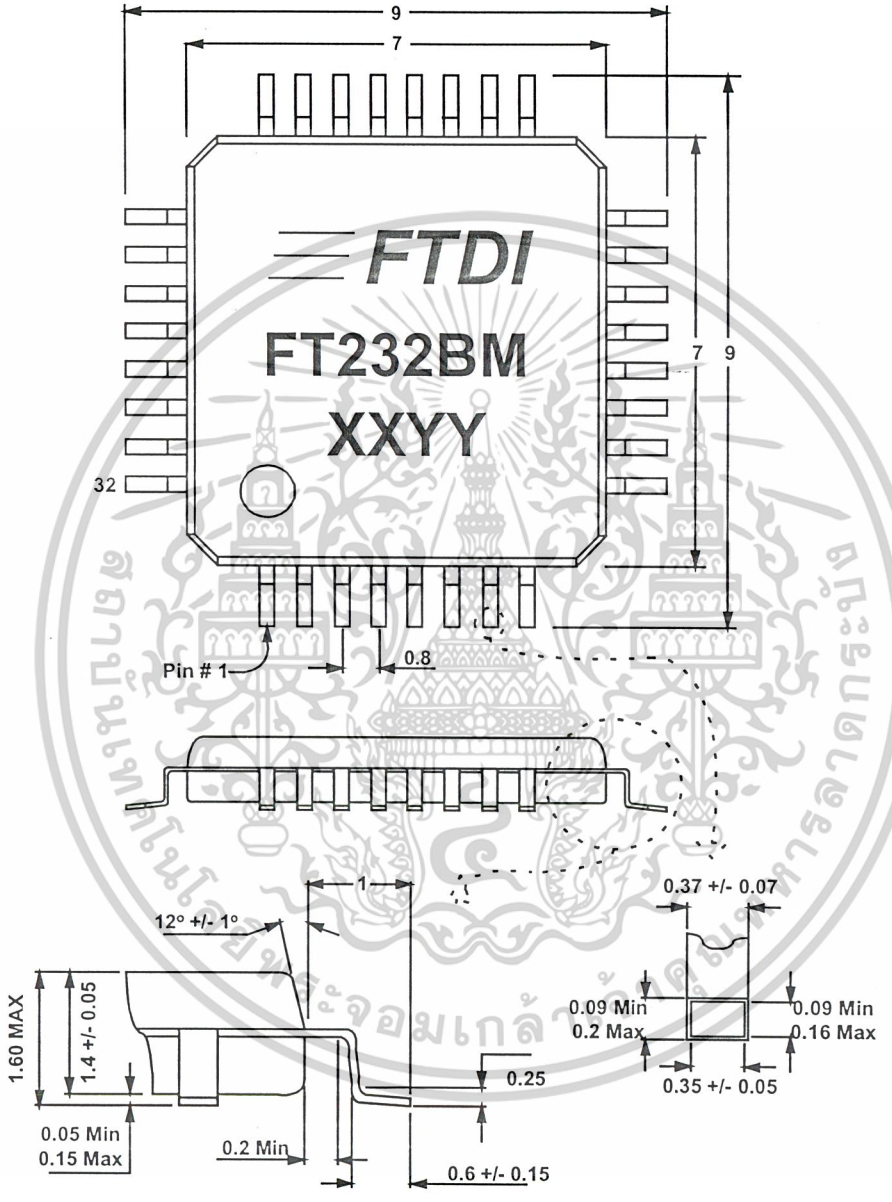
POWER AND GND GROUP

Pin#	Signal	Type	Description
6	3V3OUT	OUT	3.3 volt Output from the integrated L.D.O. regulator This pin should be decoupled to GND using a 33nF ceramic capacitor in close proximity to the device pin. It's prime purpose is to provide the internal 3.3V supply to the USB transceiver cell and the RSTOUT# pin. A small amount of current (<= 5mA) can be drawn from this pin to power external 3.3v logic if required.
3,26	VCC	PWR	+4.35 volt to +5.25 volt VCC to the device core, LDO and non-UART interface pins.
13	VCCIO	PWR	+3.0 volt to +5.25 volt VCC to the UART interface pins 10..12, 14..16 and 18..25. When interfacing with 3.3V external logic connect VCCIO to the 3.3V supply of the external logic, otherwise connect to VCC to drive out at 5V CMOS level.
9,17	GND	PWR	Device– Ground Supply Pins
30	AVCC	PWR	Device - Analog Power Supply for the internal x8 clock multiplier
29	AGND	PWR	Device - Analog Ground Supply for the internal x8 clock multiplier

**Note 1 - During device reset, these pins are tri-state but pulled up to VCC via internal 200k resistors.

5.0 Package Outline

Figure 3 – 32 LD LQFP Package Dimensions



The FT232BM is supplied in a 32 LD LQFP package as standard. This package has a 7mm x 7mm body (9mm x 9mm including leads) with leads on a 0.8mm pitch. A lead free version is available on special request.

An alternative 5mm x 5mm leadless chip scale package (C.S.P.) is also available on special request for projects where package area is critical.

Contact support@ftdichip.com for package details and availability.

The above drawing shows the LQFP-32 package – all dimensions are in millimetres.

XXYY = Date Code (XX = 1 or 2 digit year number, YY = 2 digit week number.

6.0 Absolute Maximum Ratings

These are the absolute maximum ratings for the FT232BM device in accordance with the Absolute Maximum Rating System (IEC 60134). Exceeding these may cause permanent damage to the device.

- Storage Temperature ..... –65°C to + 150°C
- Ambient Temperature (Power Applied) ..... 0°C to + 70°C
- VCC Supply Voltage ..... -0.5V to +6.00V
- DC Input Voltage - Inputs ..... -0.5V to VCC + 0.5V
- DC Input Voltage - High Impedance Bidirectionals ..... -0.5V to VCC + 0.5V
- DC Output Current – Outputs 24mA
- DC Output Current – Low Impedance Bidirectionals 24mA
- Power Dissipation (VCC = 5.25V) 500mW
- Electrostatic Discharge Voltage (I < 1uA) ..... +/- 2000V
- Latch Up Current (Vi < 0 or Vi > Vcc) 100mA

6.1 D.C. Characteristics

DC Characteristics (Ambient Temperature = 0 .. 70°C)

Operating Voltage and Current

Parameter	Description	Min	Typ	Max	Units	Conditions
Vcc1	VCC Operating Supply Voltage	4.35	5.0	5.25	V	
Vcc2	VCCIO Operating Supply Voltage	3.0	-	5.25	V	
Icc1	Operating Supply Current	-	25	-	mA	Normal Operation
Icc2	Operating Supply Current	-	180	200	uA	USB Suspend **Note 1

**Note 1 – *Supply current excludes the 200uA nominal drawn by the external pull-up resistor on USB DP.*

UART IO Pin Characteristics (VCCIO = 5.0V)

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
Vol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2mA
Vin	Input Switching Threshold	1.3	1.6	1.9	V	**Note 2
VHys	Input Switching Hysteresis	50	55	60	mV	

UART IO Pin Characteristics (VCCIO = 3.0 - 3.6V)

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	2.2	2.7	3.2	V	I source = 1mA
Vol	Output Voltage Low	0.3	0.4	0.5	V	I sink = 2 mA
Vin	Input Switching Threshold	1.0	1.2	1.5	V	**Note 2
VHys	Input Switching Hysteresis	20	25	30	mV	

**Note 2 – *Inputs have an internal 200k pull-up resistor to VCCIO.*

XTIN / XTOUT Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	4.0	-	5.0	V	Fosc = 6MHz
Vol	Output Voltage Low	0.1	-	1.0	V	Fosc = 6MHz
Vin	Input Switching Threshold	1.8	2.5	3.2	V	

RESET#, TEST, EECS, EESK, EEDATA Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
Vol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2 mA
Vin	Input Switching Threshold	1.3	1.6	1.9	V	**Note 3
VHys	Input Switching Hysteresis	50	55	60	mV	

**Note 3 – *EECS, EESK and EEDATA pins have an internal 200k pull-up resistor to VCC*

RSTOUT Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.0	-	3.6	V	I source = 2mA
Vol	Output Voltage Low	0.3	-	0.6	V	I sink = 2mA

USB IO Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
UVoh	IO Pins Static Output (High)	2.8		3.6	V	RI = 1.5k to 3V3Out (D+) RI = 15k to GND (D-)
UVol	IO Pins Static Output (Low)	0		0.3	V	RI = 1.5k to 3V3Out (D+) RI = 15k to GND (D-)
UVse	Single Ended Rx Threshold	0.8		2.0	V	
UCom	Differential Common Mode	0.8		2.5	V	
UVDif	Differential Input Sensitivity	0.2			V	
UDrvZ	Driver Output Impedance	29		44	Ohm	**Note 4

**Note 4 – *Driver Output Impedance includes the external 27R series resistors on USBDP and USBDM pins.*

7.0 Device Configuration Examples

7.1 Oscillator Configurations

Figure 4
3-Pin Ceramic Resonator Configuration

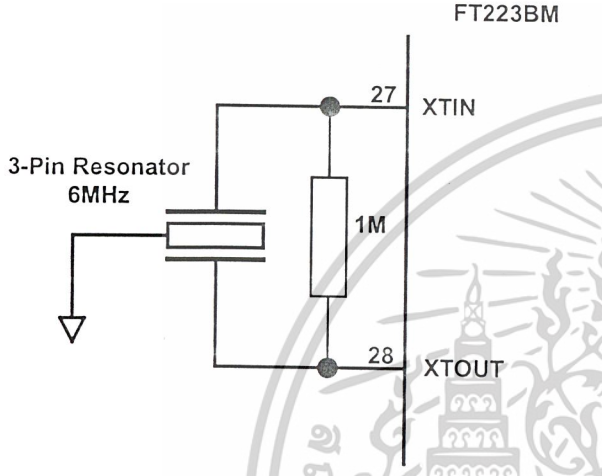


Figure 5
Crystal or 2-Pin Ceramic Resonator Configuration

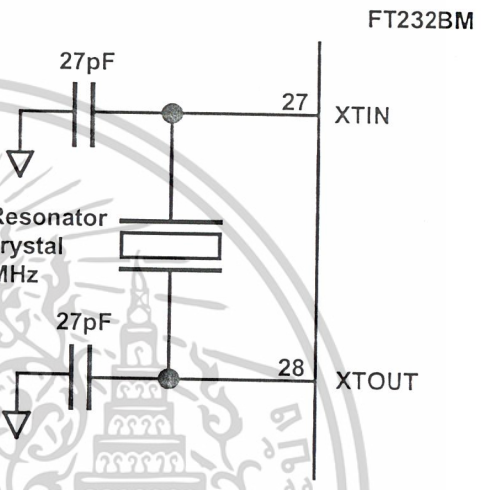


Figure 4 illustrates how to use the FT232BM with a 3-Pin Ceramic Resonator. A suitable part would be a ceramic resonator from Murata's CERALOCK range. (Murata Part Number CSTCR6M00G15), or equivalent. 3-Pin ceramic resonators have the load capacitors built into the resonator so no external loading capacitors are required. This makes for an economical configuration. The accuracy of this Murata ceramic resonator is +/- 0.1% and it is specifically designed for USB full speed applications. A 1 MOhm loading resistor across XTIN and XTOUT is recommended in order to guarantee this level of accuracy.

Other ceramic resonators with a lesser degree of accuracy (typically +/- 5%) are technically out-with the USB specification, but it has been calculated that using such a device will work satisfactorily in practice with a FT232BM design.

Figure 5 illustrates how to use the FT232BM with a 6MHz Crystal or 2-Pin Ceramic Resonator. In this case, these devices do not have in-built loading capacitors so these have to be added between XTIN, XTOUT and GND as shown. A value of 27pF is shown as the capacitor in the example – this will be good for many crystals and some resonators but do select the value based on the manufacturers recommendations wherever possible. If using a crystal, use a parallel cut type. If using a resonator, see the previous note on frequency accuracy.

7.2 EEPROM Configuration

Figure 6
EEProm Configuration

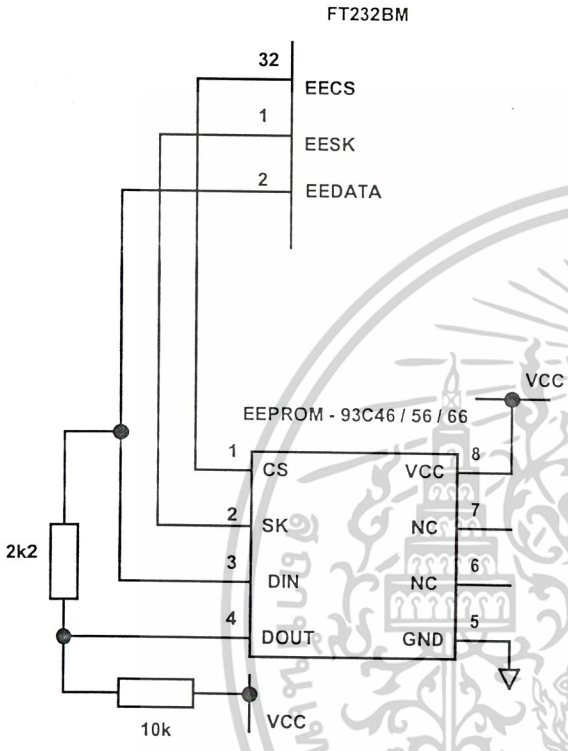


Figure 6 illustrates how to connect the FT232BM to the 93C46 (93C56 or 93C66) EEPROM. EECS (pin 32) is directly connected to the chip select (CS) pin of the EEPROM. EESK (pin 1) is directly connected to the clock (SK) pin of the EEPROM. EEDATA (pin 2) is directly connected to the Data In (Din) pin of the EEPROM. There is a potential condition whereby both the Data Output (Dout) of the EEPROM can drive out at the same time as the EEDATA pin of the FT232BM. To prevent potential data clash in this situation, the Dout of the EEPROM is connected to EEDATA of the FT232BM via a 2.2k resistor.

Following a power-on reset or a USB reset, the FT232BM will scan the EEPROM to find out (a) if an EEPROM is attached to the Device and (b) if the data in the device is valid. If both of these are the case, then the FT232BM will use the data in the EEPROM, otherwise it will use it's built-in default values. When a valid command is issued to the EEPROM from the FT232BM, the EEPROM will acknowledge the command by pulling it's Dout pin low. In order to check for this condition, it is necessary to pull Dout high using a 10k resistor. If the command acknowledge doesn't happen then EEDATA will be pulled high by the 10k resistor during this part of the cycle and the device will detect an invalid command or no EEPROM present.

There are two varieties of these EEPROM's on the market – one is configured as being 16 bits wide, the other is configured as being 8 bits wide. These are available from many sources such as Microchip, STMicro, ISSI etc. The FT232BM requires EEPROM's with a 16-bit wide configuration such as the Microchip 93LC46B device. The EEPROM must be capable of reading data at a 1Mb clock rate at a supply voltage of 4.35V to 5.25V. Most available parts are capable of this.

Check the manufacturers data sheet to find out how to connect pins 6 and 7 of the EEPROM. Some devices specify these as no-connect, others use them for selecting 8 / 16 bit mode or for test functions. Some other parts have their pinout rotated by 90° so please select the required part and it's options carefully.

It is possible to "share" the EEPROM between the FT232BM and another external device such as an MCU. However, this can only be done when the FT232BM is in it's reset condition as it tri-states it's EEPROM interface at that time. A typical configuration would use four bit's of an MCU IO Port. One bit would be used to hold the FT232BM reset (using RESET#) on power-up, the other three would connect to the EECS, EESK and EEDATA pins of the FT232BM in order to read / write data to the EEPROM at this time. Once the MCU has read / written the EEPROM, it would take RESET# high to allow the FT232BM to configure itself and enumerate over USB.

7.3 USB Bus Powered and Self Powered Configuration

Figure 7
USB Bus Powered Configuration

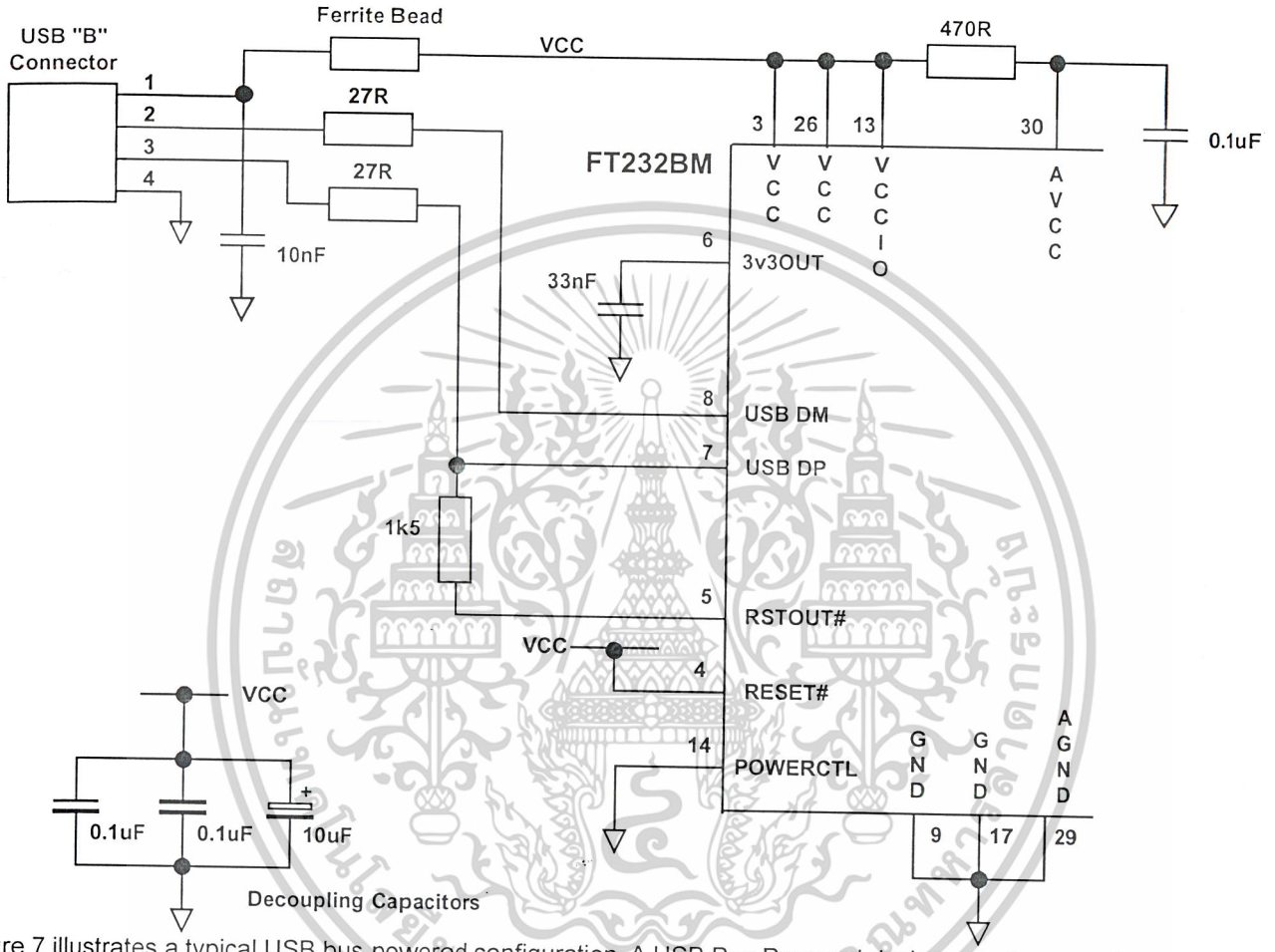


Figure 7 illustrates a typical USB bus powered configuration. A USB Bus Powered device gets its power from the USB bus. Basic rules for USB Bus power devices are as follows –

a) On plug-in, the device must draw no more than 100mA
b) On USB Suspend the device must draw no more than 500uA.
c) A Bus Powered High Power Device (one that draws more than 100mA) should use the PWREN# pin to keep the current below 100mA on plug-in and 500uA on USB suspend.
d) A device that consumes more than 100mA can not be plugged into a USB Bus Powered Hub
e) No device can draw more that 500mA from the USB Bus.

PWRCTL (pin 14) is pulled low to tell the device to use a USB Bus Power descriptor. The power descriptor in the EEPROM should be programmed to match the current draw of the device.

A Ferrite Bead is connected in series with USB power to prevent noise from the device and associated circuitry (EMI) being radiated down the USB cable to the Host. The value of the Ferrite Bead depends on the total current required by the circuit – a suitable range of Ferrite Beads is available from Steward (www.steward.com) for example Steward Part # **MI0805K400R-00 also available as DigiKey Part #** 240-1035-1.

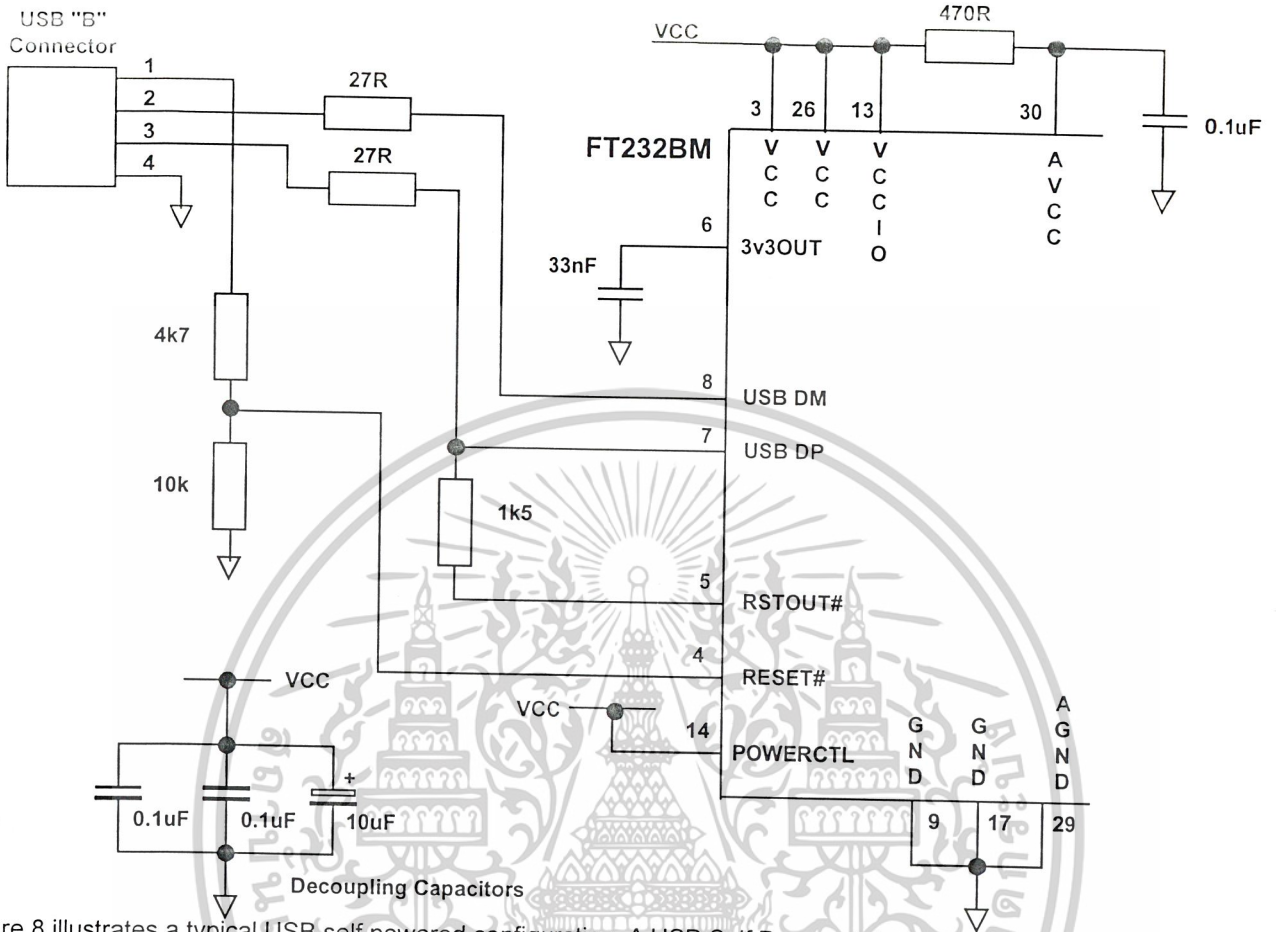


Figure 8 illustrates a typical USB self powered configuration. A USB Self Powered device gets its power from its own POWER SUPPLY and does not draw current from the USB bus. The basic rules for USB Self power devices are as follows –

a) A Self-Powered device should not force current down the USB bus when the USB Host or Hub Controller is powered down.

b) A Self Powered Device can take as much current as it likes during normal operation and USB suspend as it has its own POWER SUPPLY.

c) A Self Powered Device can be used with any USB Host and both Bus and Self Powered USB Hubs

PWRCTL (pin 14) is pulled high to tell the device to use a USB Bus Power descriptor. The power descriptor in the EEPROM should be programmed to a value of zero. The USB power descriptor option in the EEPROM should be programmed to a value of zero (self powered).

To meet requirement a) the 1.5k pull-up resistor on USB DP is connected to RSTOUT# as per the bus-power circuit. However, the USB Bus Power is used to control the RESET# Pin of the FT232BM device. When the USB Host or Hub is powered up RSTOUT# will pull the 1.5k resistor on USB DP to 3.3V, thus identifying the device as a full speed device to USB. When the USB Host or Hub power is off, RESET# will go low and the device will be held in reset. As RESET# is low, RSTOUT# will also be low, so no current will be forced down USB DP via the 1.5k pull-up resistor when the host or hub is powered down. Failure to do this may cause some USB host or hub controllers to power up erratically.

Note : When the FT232BM is in reset, the UART interface pins all go tri-state. These pins have internal 200k pull-up resistors to VCCIO, so they will gently pull high unless driven by some external logic.

7.4 UART Interface Configuration

Figure 9
USB <=> RS232 Converter Configuration

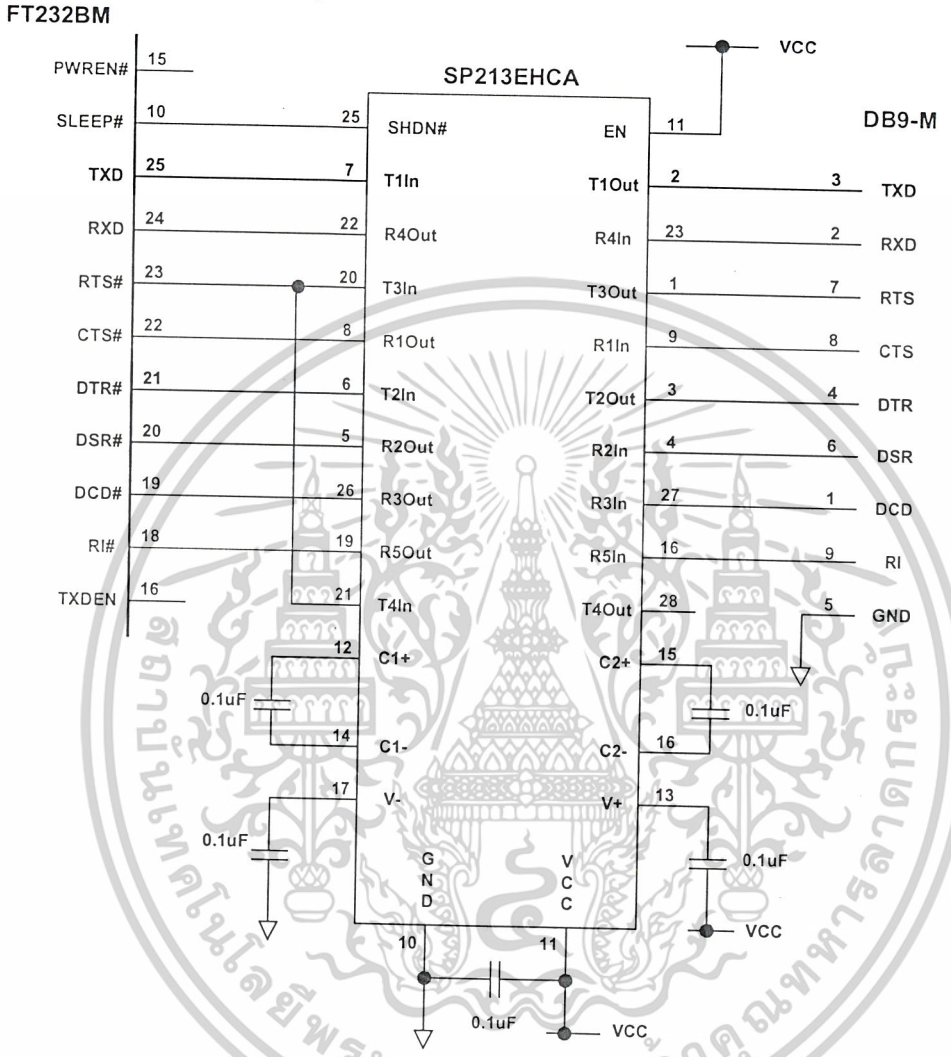


Figure 9 illustrates how to connect the UART interface of the FT232BM to a TTL – RS232 Level Converter I.C. to make a USB <=> RS232 converter using the popular "213" series of TTL to RS232 level converters. These devices have 4 transmitters and 5 receivers in a 28 LD SSOP package and feature an in-built voltage converter to convert the 5v (nominal) VCC to the +/- 9 volts required by RS232. An important feature of these devices is the SHDN# pin which can power down the device to a low quiescent current during USB suspend mode

The device used in the example is a Sipex SP213EHCA which is capable of RS232 communication at up to 500k baud. If a lower baud rate is acceptable, then several pin compatible alternatives are available such as Sipex SP213ECA , Maxim MAX213CAI and Analog Devices ADM213E which are good for communication at up to 115,200 baud. If a higher baud rate is desired, use a Maxim MAX3245CAI part which is capable of RS232 communication at rates of up to 1M baud. The MAX3245 is not pin compatible with the 213 series devices, also it's SHDN pin is active high so connect this to PWREN# instead of SLEEP#.

Figure 10
USB <=> RS422 Converter Configuration

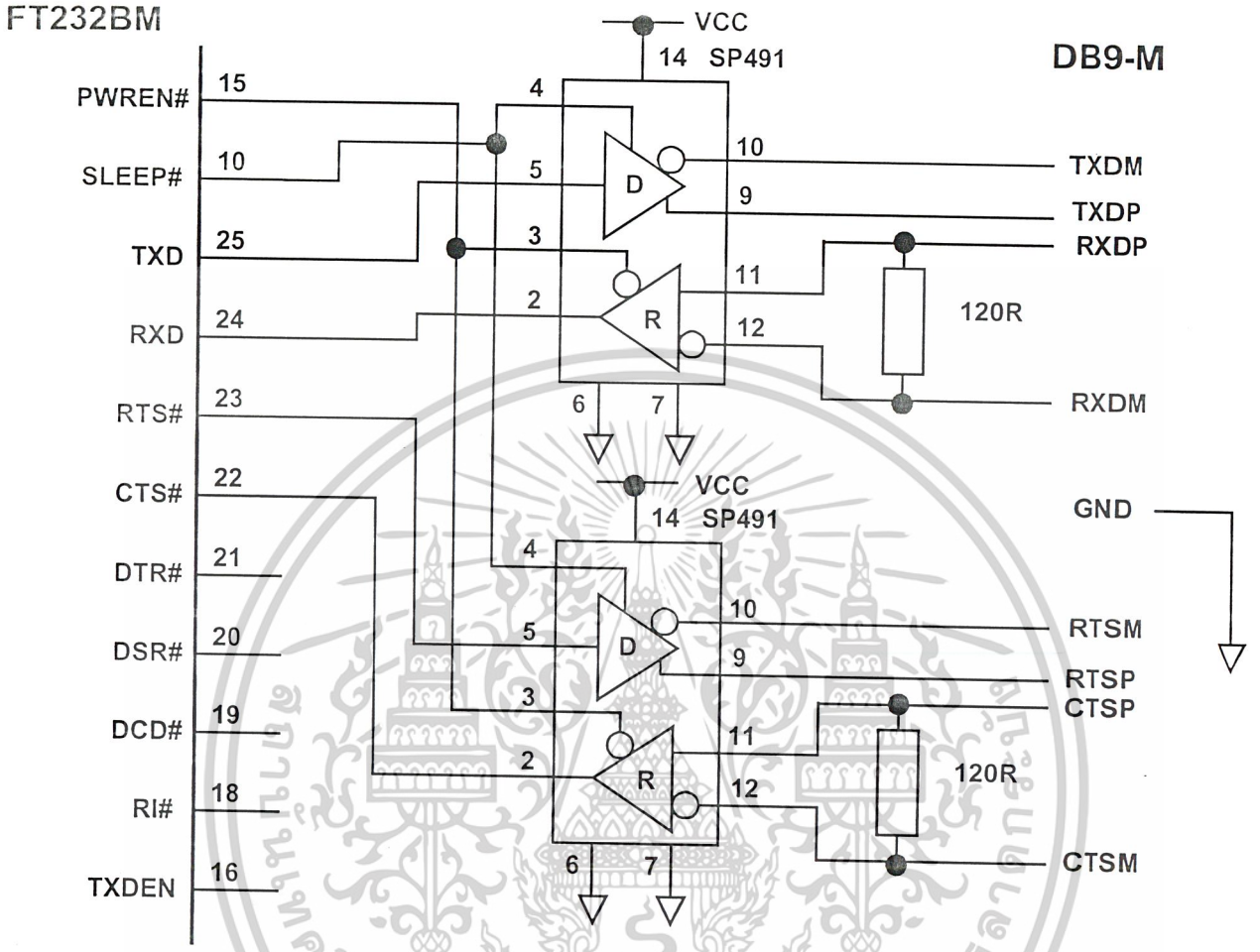


Figure 10 illustrates how to connect the UART interface of the FT232BM to a TTL – RS422 Level Converter I.C. to make a USB <=> RS422 converter. There are many such level converter devices available – this example uses Sipex SP491 devices which have enables on both the transmitter and receiver. Because the transmitter enable is active high, it is connected to the SLEEP# pin. The receiver enable is active low and is connected to the PWREN# pin. This ensures that both the transmitters and receivers are enabled when the device is active, and disabled when the device is in USB suspend mode. If the design is USB BUS powered, it may be necessary to use a P-Channel logic level MOSFET (controlled by PWREN#) in the VCC line of the SP491 devices to ensure that the USB standby current of 500uA is met.

The SP491 is good for sending and receiving data at a rate of up to 5M Baud – in this case the maximum rate is limited to 3M Baud by the FT232BM.

Figure 11
USB <=> RS485 Converter Configuration

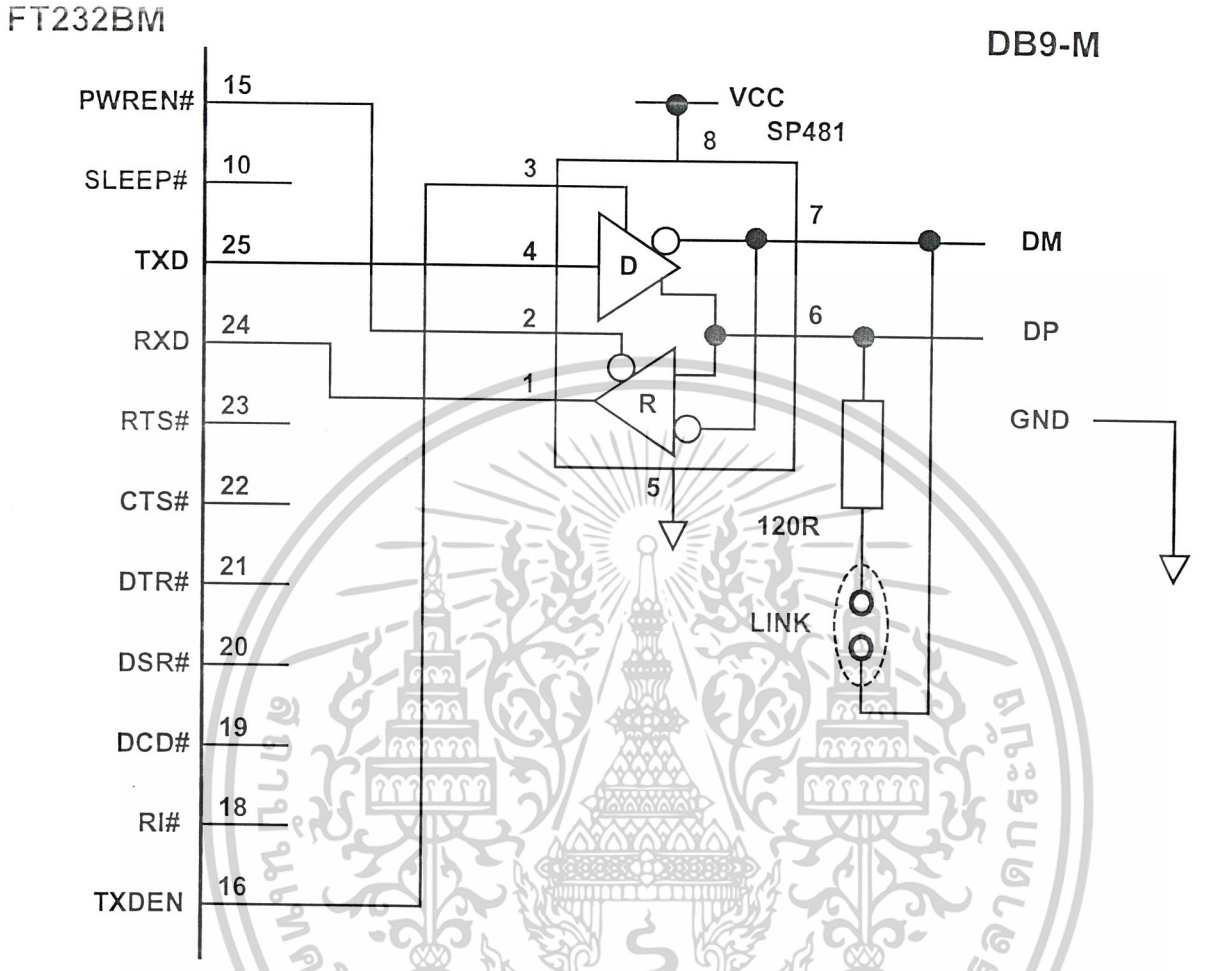


Figure 11 illustrates how to connect the UART interface of the FT232BM to a TTL – RS485 Level Converter I.C. to make a USB => RS485 converter. This example uses the Sipex SP491 device but there are similar parts available from Maxim and Analog Devices amongst others. The SP491 is a RS485 device in a compact 8 pin SOP package. It has separate enables on both the transmitter and receiver. With RS485, the transmitter is only enabled when a character is being transmitted from the UART. The TXDEN pin on the FT232BM is provided for exactly that purpose and so the transmitter enable is wired to TXDEN. The receiver enable is active low, so it is wired to the PWREN# pin to disable the receiver when in USB suspend mode.

RS485 is a multi-drop network – i.e. many devices can communicate with each other over a single two wire cable connection. The RS485 cable requires to be terminated at each end of the cable. A link is provided to allow the cable to be terminated if the device is physically positioned at either end of the cable.

In this example the data transmitted by the FT232BM is also received by the device that is transmitting. This is a common feature of RS485 and requires the application software to remove the transmitted data from the received data stream. With the FT232BM it is possible to do this entirely in hardware – simply modify the schematic so that RXD of the FT232BM is the logical OR of the SP481 receiver output with TXDEN using an HC32 or similar logic gate.

7.5 LED Interface

Figure 12
Dual LED Configuration

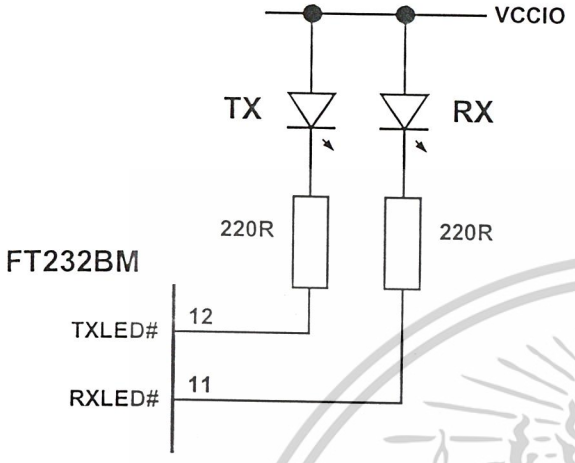
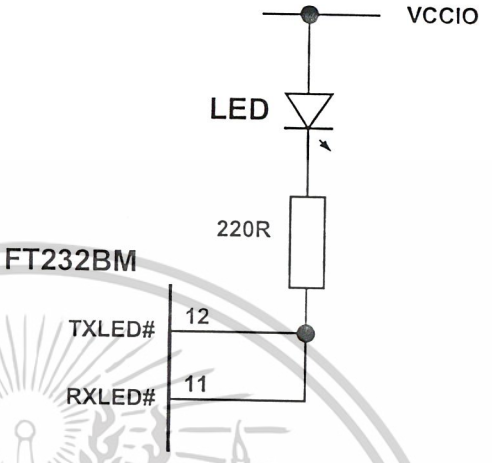


Figure 13
Single LED Configuration



The FT232BM has two IO pins dedicated to controlling LED status indicators, one for transmitted data the other for received data. When data is being transmitted / received the respective pins drive from tri-state to low in order to provide indication on the LEDs of data transfer. A digital one-shot timer is used so that even a small percentage of data transfer is visible to the end user. Figure 12 shows a configuration using two individual LED's – one for transmitted data the other for received data. In Figure 13, the transmit and receive LED indicators are wire-or'd together to give a single LED indicator which indicates any transmit or receive data activity.
Another possibility (not shown here) is to use a 3 pin common anode tri-color LED based on the circuit in Figure 13 to have a single LED that can display activity in a variety of colors depending on the ratio of transmit activity compared to receive activity.

Note that the LED's are connected to VCCIO.

7.6 Interfacing to 3.3v Logic

Figure 14
Bus Powered Circuit with 3.3V logic drive / supply voltage

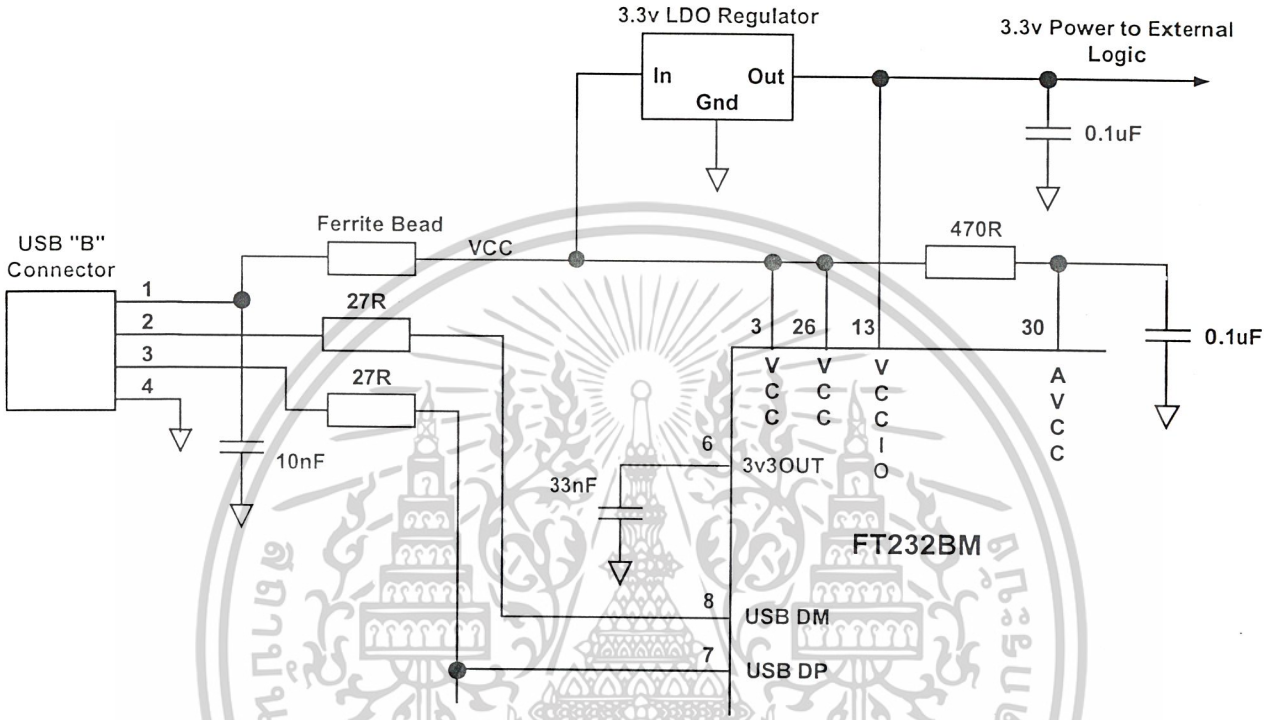


Figure 14 shows how to configure the FT232BM to interface with a 3.3V logic device. In this example, a discrete 3.3V regulator is used to supply the 3.3V logic from the USB supply. VCCIO is connected to the output of the 3.3V regulator, which in turn will cause the UART interface IO pins to drive out at 3.3V level. For USB bus powered circuits some considerations have to be taken into account when selecting the regulator –

a) The regulator must be capable of sustaining its output voltage with an input voltage of 4.35 volts. A Low Drop Out (LDO) regulator must be selected.
b) The quiescent current of the regulator must be low in order to meet the USB suspend total current requirement of <= 500uA during USB suspend.

An example of a regulator family that meets these requirements is the MicroChip (Telcom) TC55 Series. These devices can supply up to 250mA current and have a quiescent current of under 1uA.

In some cases, where only a small amount of current is required (< 5mA) , it may be possible to use the in-built regulator of the FT232BM to supply the 3.3v without any other components being required. In this case, connect VCCIO to the 3v3OUT pin of the FT232BM.

Note : It should be emphasised that the 3.3V supply for VCCIO in a bus powered design with a 3.3V logic interface should come from an LDO which is supplied by the USB bus, or from the 3V3OUT pin of the FT232BM, and not from any other source.

Figure 15
Self Powered Circuit with 3.3V logic drive / supply voltage

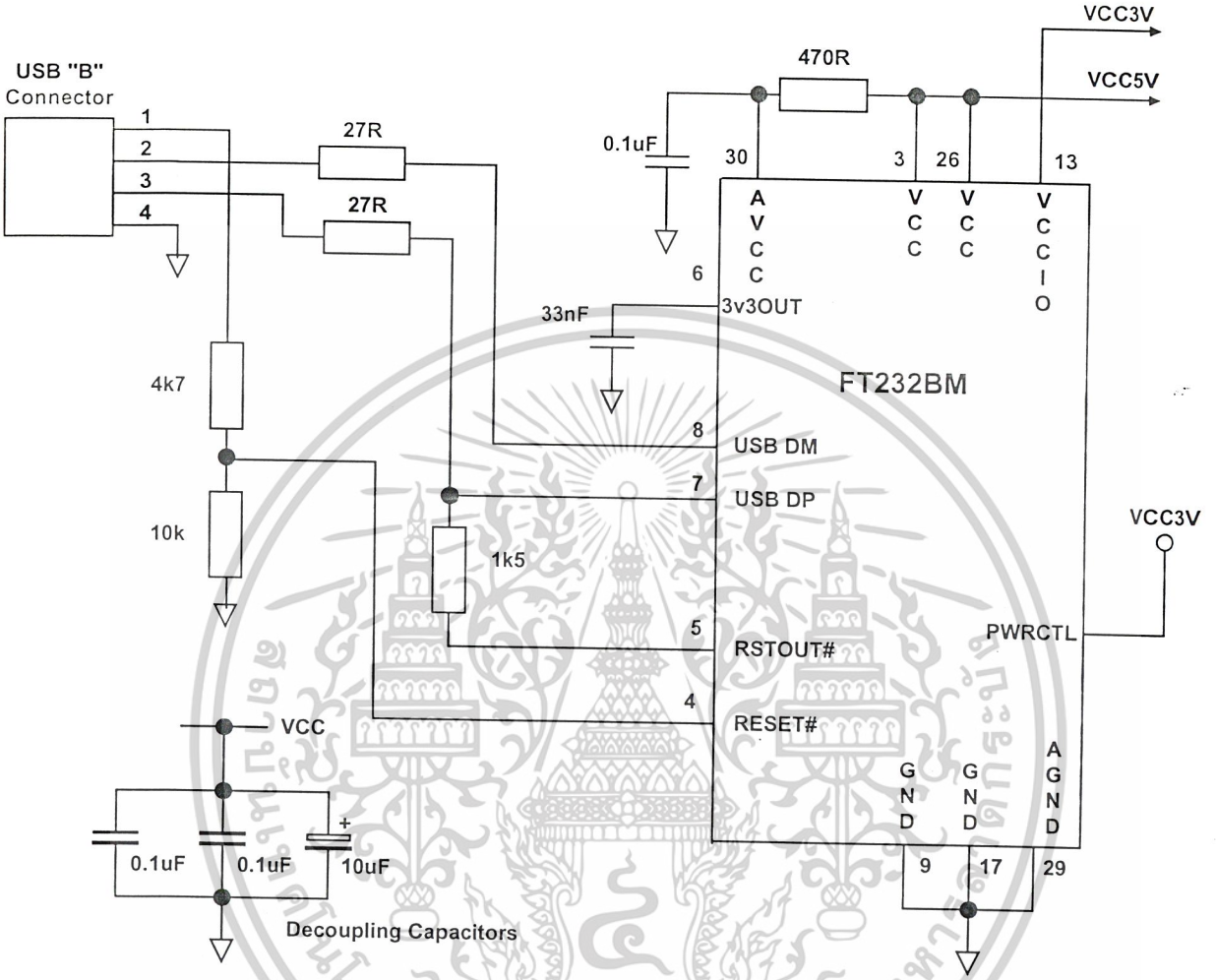


Figure 15 is an example of a USB self powered design with 3.3V interface. In this case VCCIO is supplied by an external 3.3V supply in order to make the device IO pins drive out at 3.3V logic level, thus allowing it to be connected to a 3.3V MCU or other external logic. A USB self powered design uses its own power supplies, and does not draw any of it's power from the USB bus. In such cases, no special care need be taken to meet the USB suspend current (0.5 mA) as the device does not get it's power from the USB port.

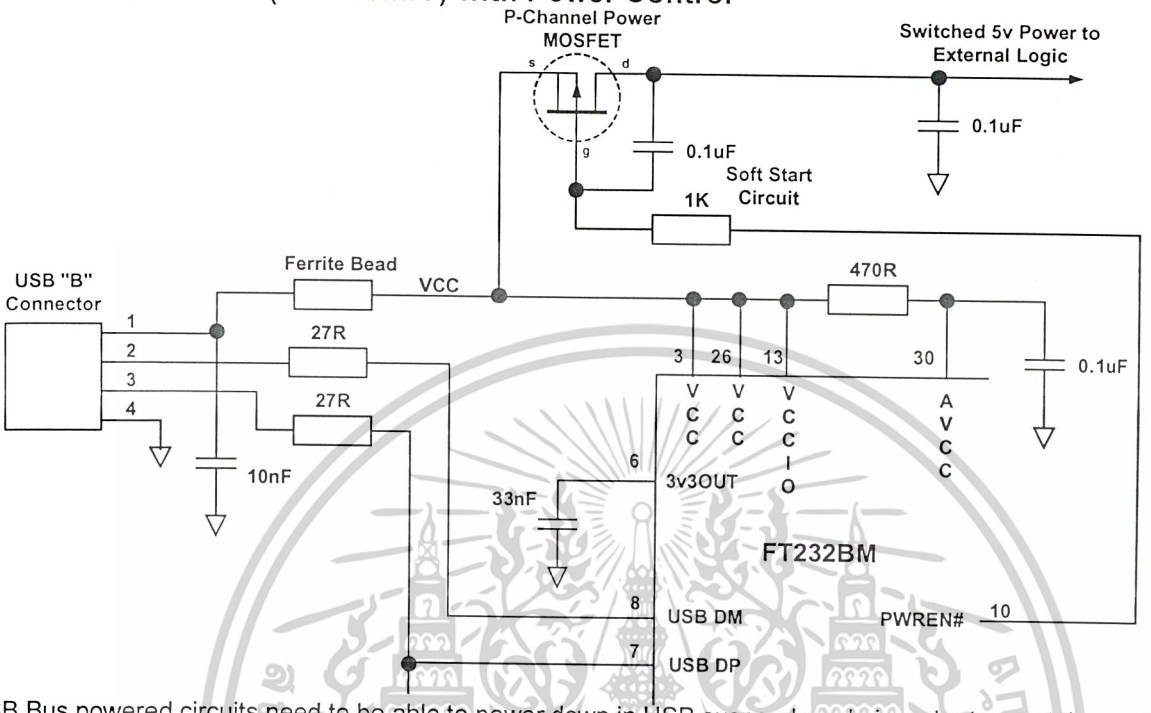
As with bus powered 3.3V interface designs, in some cases, where only a small amount of current is required (<5mA), it may be possible to use the in-built regulator of the FT232BM to supply the 3.3V without any other components being required. In this case, connect VCCIO to the 3v3OUT pin of the FT232BM.

Note that in this case PWRCTL is pulled up to VCCIO, not VCC.

7.7 Power Switching

Figure 16

Bus Powered Circuit (<= 100mA) with Power Control



USB Bus powered circuits need to be able to power down in USB suspend mode in order to meet the <= 500uA total suspend current requirement (including external logic). Some external logic can power itself down into a low current state by monitoring the PWREN# pin. For external logic that cannot power itself down in that way, the FT232BM provides a simple but effective way of turning off power to external circuitry during USB suspend.

Figure 16 shows how to use a discrete P-Channel Logic Level MOSFET to control the power to external logic circuits. A suitable device could be a Fairchild NDT456P, or International Rectifier IRLML6402, or equivalent. It is recommended that a "soft start" circuit consisting of a 1K series resistor and a 0.1 uF capacitor are used to limit the current surge when the MOSFET turns on. Without the soft start circuit there is a danger that the transient power surge of the MOSFET turning on will reset the FT232BM, or the USB host / hub controller. The values used here allow attached circuitry to power up with a slew rate of ~12.5 V per millisecond, in other words the output voltage will transition from GND to 5 V in approximately 400 microseconds.

Alternatively, a dedicated power switch i.c. with inbuilt "soft-start" can be used instead of a MOSFET. A suitable power switch i.c. for such an application would be a Micrel (www.micrel.com) MIC2025-2BM or equivalent.

Please note the following points in connection with power controlled designs –

a) The logic to be controlled must have it's own reset circuitry so that it will automatically reset itself when power is re-applied on coming out of suspend.

b) Set the Pull-down on Suspend option in the FT232BM's EEPROM.

c) For USB high-power bus powered device (one that consumes greater than 100 mA, and up to 500 mA of current from the USB bus), the power consumption of the device should be set in the max power field in the EEPROM. A high-power bus powered device must use this descriptor in the EEPROM to inform the system of it's power requirements.

d) For 3.3V power controlled circuits VCCIO must not be powered down with the external circuitry (PWREN# gets it's VCC supply from VCCIO). Either connect the power switch between the output of the 3.3V regulator and the external 3.3V logic OR if appropriate power VCCIO from the 3V3OUT pin of the FT232BM.