การออกแบบสร้างระบบควบคุมอุปกรณ์เครื่องมือวัดด้วยมาตรฐาน IEEE-488 (GPIB)

A DESIGN AND CONSTRUCTION OF MASUREMENT CONTROL SYSTEM BY IEEE-488 (GPIB) STANDARD

โดย

นาย บุญญฤทธิ์ ลักษณประณัย

นาย คทา จารุวงศ์รังสี

นาย นครินทร์ รัตนมณีศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถานบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การออกแบบสร้างระบบควบคุมอุปกรณ์เครื่องมือวัดด้วยมาตรฐาน IEEE-488 (GPIB)

A DESIGN AND CONSTRUCTION OF MASUREMENT CONTROL SYSTEM BY IEEE-488 (GPIB) STANDARD

โดย

นาย บุญญฤทธิ์ ลักษณประณัย รหัสประจำตัว 44015202

นาย คทา จารุวงศ์รังสี รหัสประจำตัว 44015187

นาย นครินทร์ รัตนมณีศิลป์ รหัสประจำตัว 44015195

อาจารย์ที่ปรึกษา

รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบสร้างระบบควบคุมอุปกรณ์เครื่องมือวัดด้วยมาตรฐาน IEEE-488 (GPIB)

ผู้จัดทำ

นาย บุญญฤทธิ์ ลักษณประณัย 44015202

นาย คทา จารุวงศ์รังสี 44015187

นาย นครินทร์ รัตนมณีศิลป์ 44015195

.....

(รศ.ดร.มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

การออกแบบสร้างระบบควบคุมอุปกรณ์เครื่องมือวัดด้วยมาตรฐาน IEEE-488 (GPIB)

A DESIGN AND CONSTRUCTION OF MASUREMENT CONTROL SYSTEM

BY IEEE-488 (GPIB) STANDARD

นาย บุญญฤทธิ์ ลักษณประณัย 44015202

นาย คทา จารุวงศ์รังสี 44015187

นาย นครินทร์ รัตนมณีศีลป์ 44015195

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ.ดร.มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

การออกแบบสร้างระบบควบคุมอุปกรณ์เครื่องมือวัดด้วยมาตรฐาน IEEE-488 (GPIB)

A DESIGN AND CONSTRUCTION OF MASUREMENT CONTROL SYSTEM BY IEEE-488 (GPIB) STANDARD

นาย บุญญฤทธิ์ ลักษณะประณัย 44015202

นาย คทา จารุวงศ์รังสี 44015187

นาย นครินทร์ รัตนมณีศีลป์ 44015195

รศ.ดร. มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา

ปีการศึกษา 2546

บทคัดย่อ

ระบบบัสแบบ GPIB (General Purpose Interface Bus) หรือเรียกตามรหัสมาตรฐานว่า IEEE-488 นั้นเป็นระบบบัสรูปแบบหนึ่งที่เป็นที่นิยมนำมาใช้งานในการควบคุมอุปกรณ์เครื่องมือวัดต่างๆ โดยมีการส่งข้อมูลในรูปแบบขนานจึงทำให้สามารถเชื่อมต่ออุปกรณ์หลายๆ ตัวร่วมไปในระบบบัสเดียวกันได้ โดยทั่วไปนั้นการอินเตอร์เฟสระบบบัส GPIB เพื่อที่จะส่งข้อมูลไปยังคอมพิวเตอร์นั้น มักพบเห็นในรูปแบบการ์ดอินเตอร์เฟสผ่านทางระบบบัส PCI หรืออาจจะเป็นระบบบัส ISA เป็นต้น ซึ่งการ์ดอินเตอร์เฟสเหล่านี้นั้นมีราคาสูง อีกทั้งการใช้งานก็มีข้อจำกัดทางด้านการติดตั้ง เนื่องจากต้องติดตั้งไว้ภายในเท่านั้น ดังนั้นในปริญญานิพนธ์นี้จึงเป็นการออกแบบในส่วนของการ์ดอินเตอร์เฟส ให้สามารถอินเตอร์เฟสผ่านทางพอร์ต USB และ RS-232 ซึ่งเป็นพอร์ตมาตรฐานของเครื่องคอมพิวเตอร์โดยทั่วไป ส่งผลให้มีความยืดหยุ่นในการใช้งานระบบบัส GPIB เป็นอย่างมาก

A DESIGN AND CONSTRUCTION OF MASUREMENT CONTROL SYSTEM BY IEEE-488 (GPIB) STANDARD

Mr.Boonyarit Luksanapranai 44015202

Mr.Kata Jaruwongrungsee 44015187

Mr.Nakarin Rattanamaneeslip 44015195

Assoc Prof. Manas Sangworasil Advisor

ABSTRACT

GPIB (General Purpose Interface Bus) or IEEE-488 is a type of bus which has been more used in each of measurement control. A parallel data transfer can be connected to many equipment in the same bus. Generally, we used GPIB interfacing in order to send a data to a computer. Interfacing by using PCI or ISA that high cost and hard to install because it is only internal installation so in this project we had designed an interfacing card that can be used with USB port or RS-232 port which is a common port of a personal computer and effect to flexible when using the GPIB.

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาของปริญญานิพนธ์

จากความต้องการที่จะรวมการควบคุมอุปกรณ์เครื่องวัดต่างๆ ไว้ที่ศูนย์กลางการควบคุมเพียงจุดเดียวซึ่งในการควบคุมนั้น นิยมที่จะใช้เครื่องคอมพิวเตอร์เป็นศูนย์กลางการควบคุมดังกล่าว ดังนั้นจึงต้องมีการสื่อสารข้อมูลระหว่างอุปกรณ์ต่างๆ กับเครื่องคอมพิวเตอร์ ซึ่งมาตรฐานหนึ่งที่เป็นที่นิยมในการใช้งานได้แก่ มาตรฐาน IEEE-488(GPIB) ซึ่งเป็นมาตรฐานพอร์ตขนานแบบหนึ่ง มีการส่งข้อมูลขนาน 8 บิต สามารถต่อพ่วงอุปกรณ์หลายๆ ตัวร่วมกันไปในขณะเดียวกันได้ แต่เครื่องคอมพิวเตอร์โดยทั่วไปนั้น จะไม่มีส่วนการอินเตอร์เฟสกับพอร์ต GPIB นี้ติดตั้งมาด้วย ซึ่งจำเป็นต้องซื้อการ์ดอินเตอร์เฟสมาติดตั้งเพื่อใช้งาน สำหรับการ์ดอินเตอร์เฟสนั้นโดยทั่วไปมีราคาสูง และซอฟแวร์ไดร์เวอร์ที่ให้มานั้นก็ยากต่อการพัฒนา เนื่องจากจำเป็นที่จะต้องเข้าใจในจุดที่ผู้ออกแบบได้ออกแบบไว้ ซึ่งนี่จึงเป็นจุดเริ่มต้นของปริญญานิพนธ์นี้

ในปริญญานิพนธ์นี้นั้น ได้เลือกใช้ไมโครคอนโทรลเลอร์ในการแปลงพอร์ตขนานมาตรฐาน GPIB ให้อยู่ในรูปของพอร์ตสื่อสาร USB และ RS-232 ซึ่งจากโครงงานนี้จะทำให้สามารถประหยัดงบประมาณ ในการจัดซื้อการ์ดอินเตอร์เฟส GPIB ได้อย่างมาก ส่วนควบคุมมีความยืดหยุ่นสูงในการพัฒนาโปรแกรมควบคุม สามารถใช้งานได้สะดวก อีกทั้งสามารถพัฒนาให้รองรับกับพอร์ตสื่อสารรูปแบบต่างๆ ต่อไปได้ในอนาคต

1.2 จุดมุ่งหมายของปริญญานิพนธ์

- เพื่อให้สามารถควบคุมอุปกรณ์เครื่องมือวัดที่ใช้มาตรฐานการส่งข้อมูลแบบ GPIB ด้วยคอมพิวเตอร์ผ่านทางพอร์ต USB และ RS-232 ได้
- เพื่อศึกษารูปแบบโปรโตคอลของ GPIB เพื่อเป็นพื้นฐานในการพัฒนาดัดแปลงให้ใช้งานร่วมกันพอร์ตสื่อสารอื่นๆ ได้ในอนาคต
- เพื่อใช้งานแทนการ์ดอินเตอร์เฟส GPIB ที่มีราคาสูง

1.3 ขอบเขตของโครงงาน

- ใช้การควบคุมสั่งการผ่านคอมพิวเตอร์โดยเขียนโปรแกรมควบคุมระบบโดยโปรแกรม Microsoft Visual Basic

- ในส่วนของโปรแกรมควบคุมที่ได้ออกแบบไว้นั้น ได้ออกแบบให้รองรับกับอุปกรณ์ดังต่อไปนี้

⟹ HAMEG Programmable Power Supply รุ่น HM8142

⟹ HAMEG Programmable Function Generator รุ่น HM8130

⟹ HAMEG Programmable Multimeter รุ่น HM8112-2

⟹ HAMEG Storage Analog/Digital Oscilloscope รุ่น HM 1007

หากเป็นการใช้งานกับอุปกรณ์อื่นๆ นอกเหนือจากนี้ สามารถทำได้โดยการเขียนโปรแกรมในส่วนควบคุมทางคอมพิวเตอร์ขึ้นใหม่ ให้รองรับกับคำสั่งของอุปกรณ์นั้นๆ ตามความต้องการ

- ติดต่อสื่อสารกับเครื่องคอมพิวเตอร์ผ่านทางพอร์ต USB และ RS-232

บทที่ 2
ระบบบัสมาตรฐาน IEEE-488 (GPIB)

2.1 ความเป็นมาของระบบบัส IEEE-488 (GPIB)

เครื่องมือวัดต่างๆ ที่สามารถเชื่อมต่อกับอุปกรณ์ภายนอกได้นั้น ในอดีตนั้นบริษัทผู้ผลิตแต่ละบริษัทจะทำการออกแบบระบบบัสเป็นของตนเอง ซึ่งทำให้ยุ่งยากในการที่จะนำอุปกรณ์เครื่องมือวัดจากบริษัทต่างๆ มาต่อรวมภายในระบบเดียวกันอีกทั้งเป็นอุปสรรคในการพัฒนาเครื่องมือวัดเพื่อให้รองรับกับผู้ใช้ที่หลากหลาย ดังนั้นจึงเกิดการรวมกลุ่มกันของผู้ผลิตเครื่องมือวัดในสหรัฐอเมริกาเพื่อพัฒนาระบบบัสขึ้นใช้งานร่วมกัน ซึ่งทางประเทศเยอรมณีก็ได้มีการร่วมมือกันออกแบบระบบบัสที่จะใช้งานร่วมกันขึ้นเช่นกัน โดยการช่วยเหลือของสถาบัน 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 เรียกกันว่า IEEE std 488-1978 ซึ่งก็คือระบบ IEEE 488 (GPIB) นั่นเอง ซึ่งระบบบัสนี้ได้ถูกนำไปใช้ในระบบของ IEC ด้วย เรียกว่าระบบบัส IEC625-1 โดยมีรายละเอียดเหมือน IEEE std 488-1978 ทุกประการ เพียงแต่แตกต่างกันในตำแหน่งของขั้วต่อสัญญาณเท่านั้น

2.2 คุณสมบัติและข้อจำกัดในการต่อพ่วงอุปกรณ์

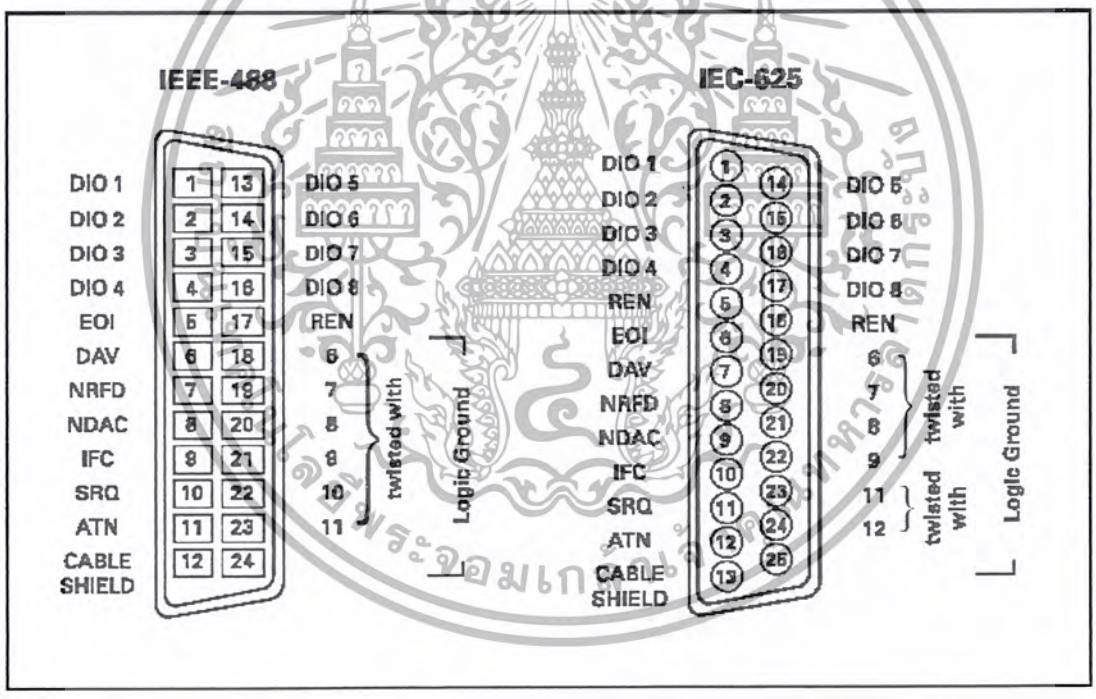
ในการระบุ Address ของอุปกรณ์ต่อพ่วงในระบบ IEEE-488 นี้นั้น มี Address ที่สามารถเป็นไปได้อยู่ระหว่าง 0 ถึง 30 ซึ่งในช่วงเวลาหนึ่งๆ นั้นตัวควบคุมจะสามารถติดต่ออุปกรณ์เครื่องมือวัดได้เพียง Address เดียวกันเท่านั้น โดยใน 1 ช่วงเวลาอุปกรณ์นั้นจะถูกควบคุมให้ทำงานเป็นตัวส่งหรือตัวรับเท่านั้น ไม่สามารถกระทำในเวลาเดียวกันได้ สำหรับส่วนของอุปกรณ์ต่อพ่วงต่างๆ นั้นจะมีอยู่ด้วยกัน 4 รูปแบบ ได้แก่

- อุปกรณ์ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว (Talker) เช่น เครื่องวัดแรงดัน (Volt Meter) เป็นต้น
- อุปกรณ์ที่ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว (Listener) เช่น เครื่องพิมพ์ (Printer) เป็นต้น

- อุปกรณ์ที่ทำหน้าที่เป็นทั้งตัวส่งและตัวรับ (Talker & Listener) เช่น เครื่องมือต่างๆ ที่ควบคุมได้จากภายนอกเครื่องคอมพิวเตอร์ เป็นต้น

- อุปกรณ์ที่ทำหน้าที่เป็นทั้งตัวส่ง, ตัวรับ และตัวควบคุม เช่นเครื่องคอมพิวเตอร์ เป็นต้น

สำหรับระบบบัส GPIB นั้น จะสามารถต่อพ่วงอุปกรณ์ได้ในขณะเดียวกันเพียง 16 เครื่องเท่านั้น ซึ่งเป็นข้อจำกัดทางด้านการขับกระแสผ่านบนบัสโดยมีการจำกัดปริมาณกระแส ตั้งแต่ 0 ถึงไม่เกิน 48 mA ซึ่งในอุปกรณ์ต่อพ่วงตัวหนึ่งๆ จะกินกระแสประมาณ มากกว่าหรือเท่ากับ 3 mA ดังนั้นเมื่อคิดอุปกรณ์ต่อพ่วง 15 ตัว กับ 1 ตัวควบคุม (15+1)x3 = 48 mA นั่นเอง *สำหรับเรื่องของสถานะ Logic ของ GPIB นั้นในขณะที่เรากล่าวถึง Logic 0 จะหมายถึงสถานะเอาท์พุตเป็น High ("1") และในขณะที่กล่าวถึง Logic 1 จะหมายถึงสถานะเอาท์พุตที่เป็น Low ("0") ซึ่งเป็นจุดที่ควรระวังในการทำความเข้าใจระบบด้วย*



รูปที่ 2.1 แสดงตำแหน่งขาสัญญาณของมาตราฐาน IEEE-488 และมาตราฐาน IEC625-1

2.3 สัญญาณต่างๆ ภายในระบบ GPIB

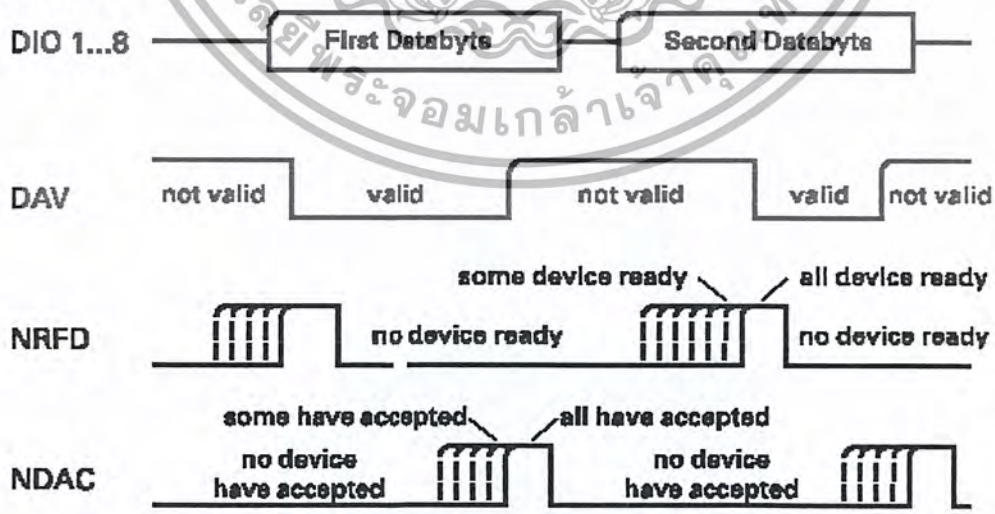
ในระบบ GPIB นั้นสายสัญญาณถูกแบ่งออกเป็น 3 ส่วนหลักๆ ได้แก่

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

2.3.2 สายสัญญาณควบคุมการรับส่งข้อมูล (Hand shake lines: DAV, NRFD and NDAC) ประกอบไปด้วยสายสัญญาณ 3 สายเพื่อใช้ในการควบคุมจังหวะของการรับและส่งข้อมูล ประกอบด้วย

- DAV (Data Valid) ในการส่งข้อมูลแต่ละชุดข้อมูลนั้นต้องมีการตรวจสอบความพร้อมของอุปกรณ์ต่างๆเป็นอันดับแรก 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 เมื่อได้รับคำสั่งใดๆ และเมื่อตัวรับได้รับชุดข้อมูลคำสั่ง มันจะทำการสั่งการเพื่อบ่งบอกว่า ขณะนี้อุปกรณ์ใดๆ พร้อมหรือไม่พร้อมที่จะรับชุดข้อมูล เมื่อมันอยู่ในสถานะ Logic 0 มันจะบ่งบอกว่าขณะนี้อุปกรณ์ที่เชื่อมต่อพร้อมที่จะรับคำสั่งซึ่งทั้งนี้การรับข้อมูลต้องขึ้นอยู่กับสถานะของ DAV ด้วย ซึ่งอุปกรณ์ตัวรับจะสั่งการ NDAC ให้เป็น Logic 0 และเริ่มทำการเก็บข้อมูลเก็บข้อมูล โดยหลังจากเก็บข้อมูลเสร็จเครื่องรับจะแจ้งกลับด้วยสัญญาณ NDAC เพื่อบ่งบอกให้ทราบว่า ข้อมูลถูกเก็บเรียบร้อยแล้ว (NDAC เป็น Logic 1) ซึ่งทั้งนี้ในขณะที่เริ่มมีการเก็บข้อมูลสัญญาณ DAV ต้องกลับมาอยู่ในสถานะ "Data not valid" (Logic 0) ด้วย เพราะเมื่อใดก็ตามที่อุปกรณ์ใดๆ ส่งสัญญาณ NRFD ให้เป็น Logic 1 สัญญาณ DAV ไม่สามารถที่จะเป็น Logic 1 ได้เพราะไม่เช่นนั้นจะไม่สามารถรับชุดข้อมูลชุดต่อไปได้



รูปที่ 2.2 แสดงความสัมพันธ์ของสัญญาณควบคุมการรับส่งข้อมูล

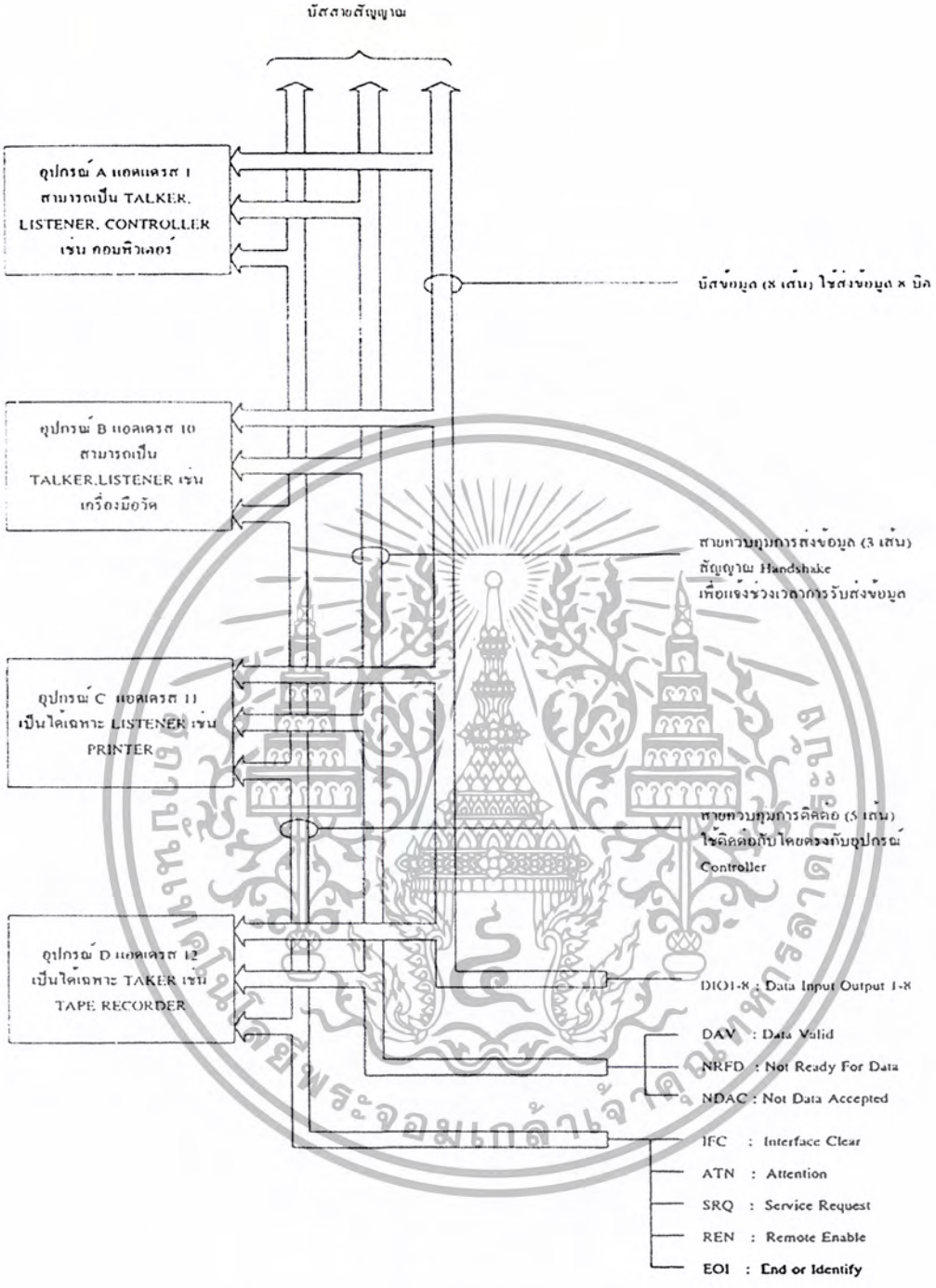
-NDAC (not data accepted) สายสัญญาณนี้จะถูกขับเมื่อมีการรับคำสั่งการใดๆ หรือชุดข้อมูลคำสั่ง โดยเป็นสายสัญญาณที่บ่งบอกว่าสัญญาณข้อมูลที่รออยู่บนบัสข้อมูลนั้นได้รับการเก็บไว้แล้วหรือไม่ โดยความสำคัญของสัญญาณ NDAC นี้คือบ่งบอกถึงว่าอุปกรณ์นั้นๆ พร้อมที่จะรับข้อมูลคำสั่งชุดใหม่แล้วหรือยัง ซึ่งมีผลเกี่ยวเนื่องกับสัญญาณ DAV ด้วยนั่นคือ ขณะที่รับข้อมูลไว้บนบัสนั้นสัญญาณ DAV จะอยู่ในสถานะ Logic 1 ซึ่งเมื่อข้อมูลถูกเก็บสู่ตัวรับ ตัวรับจะส่งสัญญาณ NDAC เป็น Logic 0 หรือสถานะ "Data Accepted" นั่นเอง ซึ่งทางด้านส่งก็จะรับรู้ และทำการเปลี่ยนสัญญาณ DAV เป็น Logic 0 เพื่อรอรับข้อมูลชุดใหม่ต่อไปและเมื่อตัวรับรับข้อมูลเรียบร้อยแล้วก็จะเปลี่ยนสถานะของ NDAC ให้เป็น Logic 1 อีกครั้งด้วยเช่นกัน

2.3.3 สายสัญญาณควบคุมเชื่อมต่อ (The Interface Management Line : ATN, IFC , REN , SRQ and EOI) ในส่วนของสายสัญญาณควบคุมการเชื่อมต่อนี้ จะเป็นกลุ่มของสายสัญญาณที่ใช้ควบคุมและจัดการการเชื่อมต่อต่างๆ โดยสายสัญญาณบางเส้นนั้นควบคุมจากส่วนควบคุมเท่านั้น ได้แก่สัญญาณ ATN , IFC และREN สำหรับสัญญาณ SQR นั้นจะถูกควบคุมโดยทางฝั่งของอุปกรณ์ที่มาเชื่อมต่อเท่านั้น และสำหรับสัญญาณ EOI นั้น เป็นสัญญาณที่จะถูกสั่งงานจากอุปกรณ์ใดก็ได้ที่ทำหน้าที่เป็นตัวส่งในขณะนั้นๆ

-ATN (Attention) สัญญาณ ATN นี้นั้น เป็นสัญญาณที่จะถูกควบคุมโดยทางภาคควบคุมเท่านั้น ซึ่งเป็นสัญญาณที่บ่งบอกสถานะของสัญญาณที่อยู่บนบัสข้อมูลว่าเป็นสัญญาณประเภทใด หากสัญญาณ ATN อยู่ในสถานะ logic 1 (Output low) จะเป็นการบ่งบอกว่าข้อมูลที่อยู่บนบัสข้อมูลนี้นั้นเป็นรหัสคำสั่งมาตราฐานของ GPIB ซึ่งเป็นข้อมูลที่มีบิตเดียว แต่เมื่อสัญญาณ ATN อยู่ในสถานะ logic 0 (Output High) จะเป็นการบ่งบอกว่าข้อมูลบนบัสข้อมูลนั้นๆ เป็นข้อมูลประเภทรหัส ASCII ซึ่งการส่งข้อมูลประเภทนี้นั้นจำเป็นต้องอาศัยสัญญาณ EOI ในการจัดชุดข้อมูลด้วย แต่อย่างไรก็ตามการส่งข้อมูลทั้งสองประเภทนี้นั้น จำเป็นที่จะต้องอาศัยการควบคุมการส่งข้อมูลจากสัญญาณควบคุมการรับส่งข้อมูล (DAV, NRFD และ NDAC) ด้วย

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

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



รูปที่ 2.3 แสดงองค์ประกอบของสัญญาณในระบบ

-SQR (Service Request) สัญญาณนี้เป็นสัญญาณจากอุปกรณ์ที่ต่อพ่วงกับตัวควบคุม เพื่อร้องขอการติดต่อจากตัวควบคุมซึ่งอาจเกิดข้อผิดพลาดในการสั่งการ จึงจำเป็นต้องร้องขอคำสั่งในการแก้ไขจากตัวควบคุม

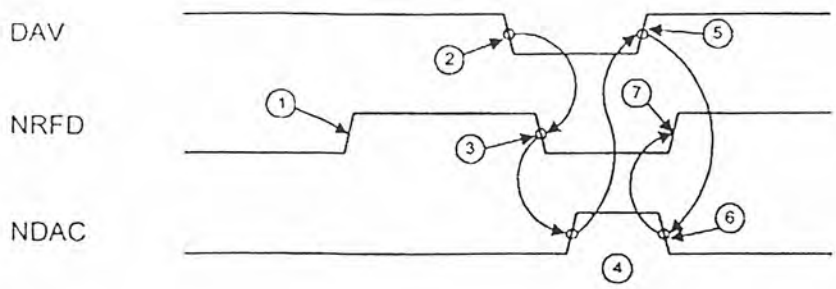
-EOI (End Of Data) เป็นสัญญาณที่ใช้ร่วมกับการส่งชุดข้อมูลคำสั่งที่เป็นประเภท String โดยในสภาวะปรกติสัญญาณนี้จะมี Logic 1 แต่เมื่อเริ่มมีการส่งข้อมูล String จากตัวส่ง สัญญาณนี้จะถูกตัวส่งกำหนดให้เป็น Logic 0 และเริ่มส่งข้อมูลออกไปทีละตัว ซึ่งทางภาครับก็จะทำการเก็บข้อมูลนั้นไว้แต่ยังไม่นำไปสั่งการจะรอขอมูลที่ตามมาในหลักต่อๆไป จนกระทั่งเมื่อสัญญาณ EOI ถูกเปลี่ยนให้กลับมาอยู่ในสถานะ Logic 1 อีกครั้ง จะเป็นการบ่งบอกว่าชุดข้อมูลที่ได้ทำการส่งนั้นจบลงแล้ว

2.4 ขบวนการแฮนด์เช็ค (Handshake Procedure)

ในการสื่อสารระหว่างภายในระบบ GPIB นั้นจะเป็นการสื่อสารแบบอะซิงโครนัส คือ เมื่อมีการรับส่งข้อมูลระหว่างตัวส่งและตัวรับ ตัวส่งจะต้องแจ้งให้ตัวรับทราบว่าตัวส่งได้ส่งข้อมูลลงไปบนบัสแล้ว และให้ตัวรับทำการเก็บข้อมูลได้ เมื่อตัวรับทำการเก็บข้อมูลเสร็จแล้ว ก็จะต้องแจ้งแก่ตัวส่งให้ทราบว่า ได้รับข้อมูลที่ส่งมาเรียบร้อยแล้ว เพื่อที่ตัวส่งจะได้ทำการหยุดส่งข้อมูลหรือทำการส่งข้อมูลชุดใหม่ลงไปบนบัส กระบวนการเหล่านี้จะเกิดทุกครั้งที่มีการรับส่งข้อมูลในระบบ ซึ่งกระบวนการเหล่านี้ถูกเรียกว่า ขบวนการแฮนด์เช็ค (Handshake Procedure)

รูปที่ 2.4 แสดงส่วนประกอบของระบบ

ในการพิจารณาถึงขบวนการแฮนด์เช็คนั้น จะทำการพิจารณาถึงระบบที่ไม่ซับซ้อนักเพื่อที่จะทำความเข้าใจได้โดยง่าย โดยกำหนดให้ในระบบมีตัวส่งและตัวรับอย่างละหนึ่งตัว ดังภาพที่ 4 ในการสื่อสารระหว่างตัวส่งและตัวรับนั้น จะมีสายสัญญาณควบคุมการรับส่งข้อมูลอยู่ 3 สัญญาณ คือ NRFD, NDAC, DAV โดยสัญญาณ DAV จะเป็นสัญญาณที่ถูกควบคุมโดยตัวส่ง ส่วนสัญญาณ NRFD, NDAC นั้นเป็นสัญญาณที่จะชี้ให้เห็นว่าตัวรับพร้อมที่จะรับข้อมูลที่ส่งลงมาบนบัสของระบบหรือไม่ สำหรับขั้นตอนของขบวนการแฮนด์เช็คสามารถแสดงได้ดังในภาพที่ 5



รูปที่ 2.5 แสดงแผนผังเวลาของขบวนการแฮนด์เช็ค

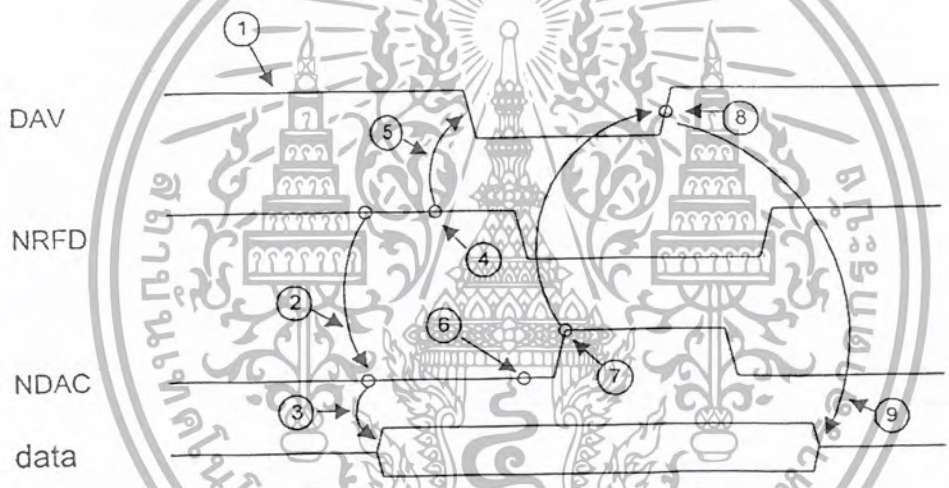
ขบวนการแฮนด์เช็คจะเริ่มขึ้นหลังจากที่ตัวควบคุมทำการบอกให้ระบบทราบว่าอุปกรณ์ตัวไหนทำหน้าที่เป็นตัวรับหรือตัวส่ง เมื่อตัวรับทราบแล้วก็จะส่งสัญญาณ NRFD ให้เป็น High (สถานะที่ 1) เพื่อบอกให้ตัวส่งทราบว่าตัวรับพร้อมที่จะรับข้อมูลแล้ว และตัวส่งก็จะทำการส่งข้อมูลลงไปบนบัสข้อมูล DIO1-DIO8 และจะทำการรออยู่ชั่วขณะหนึ่งแล้วตัวส่งส่งสัญญาณ DAV ให้เป็น Low เพื่อแจ้งให้ทราบว่าขณะนี้ตัวส่งได้ข้อมูลลงบนบัสข้อมูลแล้ว (สถานะที่ 2) เมื่อตัวรับทราบว่ามีข้อมูลอยู่บนบัสข้อมูลก็จะส่งสัญญาณ NRFD ให้มีค่าเป็น Low เมื่อตัวรับพร้อมที่จะรับข้อมูล (สถานะที่ 3) หลังจากตัวรับได้รับข้อมูลไปเก็บไว้ในบัฟเฟอร์เรียบร้อยแล้วก็จะส่งสัญญาณ NDAV ให้มีค่าเป็น High เพื่อแจ้งให้ทราบว่าตัวรับได้รับข้อมูลเรียบร้อยแล้ว (สถานะที่ 4) เมื่อตัวส่งได้รับสัญญาณ NDAC ที่เป็น High ตัวส่งก็จะทำการส่งสัญญาณ DAV ให้เป็น High เพื่อแจ้งให้ตัวรับไม่ต้องทำการเก็บข้อมูลนั้นอีก (สถานะที่ 5) เมื่อตัวรับได้รับสัญญาณ DAV ที่มีค่าเป็น High ก็จะส่งสัญญาณ NDAC ให้เป็น Low (สถานะที่ 6) ทำให้ข้อมูลในบัสถูกกำจัดออกไป หลังจากนั้นตัวรับก็จะส่งสัญญาณ NRFD ให้เป็น High (สถานะที่ 7) เพื่อบอกให้ทราบว่าตัวรับนั้นพร้อมที่จะรับข้อมูลชุดต่อไปที่จะถูกส่งเข้ามาในบัสเป็นอันเสร็จสิ้นขบวนการแฮนด์เช็ค

- ขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวส่ง

ขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวส่งสามารถแสดงด้วยแผงผังเวลา ดังภาพที่ 6 ซึ่งจะทำให้ง่ายต่อการเข้าใจในขั้นตอนการส่งข้อมูล โดยขบวนการดังกล่าวจะเกิดขึ้นหลังจากการกำหนดอุปกรณ์ในระบบแล้ว

ขบวนการแฮนด์เช็คจะเริ่มขึ้น เมื่อตัวควบคุมส่งสัญญาณ DAV ให้เป็น High (สถานะที่ 1) ซึ่งตัวควบคุมได้เซ็ทให้สัญญาณ DAV ให้มีค่าเป็น High อยู่ก่อนแล้ว หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ NRFD และ NDAC ว่ามีค่าเป็น High ทั้งคู่หรือไม่ (สถานะที่ 2) ถ้าสัญญาณทั้งสองเป็น High ทั้งคู่ แสดงว่าอุปกรณ์ไม่พร้อมที่จะทำงาน ขบวนการแฮนด์เช็คก็จะถูกยกเลิกไป แต่ถ้าที่สถานะที่ 2 หากสัญญาณใดสัญญาณหนึ่งเป็น Low ตัวควบคุมจะทำการส่งข้อมูล

ลงในบัสข้อมูล (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 จะต้องทำการเริ่มต้นใหม่ จากสถานะที่ 5 ตัวควบคุมจะรอเวลาให้ตัวรับทำการเก็บข้อมูล เมื่อถึงเวลาที่กำหนดตัวควบคุมจะทำการตรวจสอบสัญญาณว่าสัญญาณ 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 เพื่อบอกให้ทราบว่าตัวควบคุมยังไม่ได้รับข้อมูล (สถานะที่ 1) ต่อจากนั้นตัวควบคุมจะส่งสัญญาณ NRFD ให้เป็น High เพื่อบอกให้ตัวส่งทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลแล้ว (สถานะที่ 2) ตัวส่งจะทราบได้ทันทีว่าขณะนี้สามารถที่จะส่งข้อมูลลงบนบัสข้อมูลได้แล้ว (สถานะที่ 3) จากการที่สัญญาณ NRFD มีลอจิกเป็น High และสัญญาณ NDAC มีลอจิกเป็น Low ขั้นต่อไปตัวควบคุมจะทำการรอเวลาให้ตัวส่งทำการส่งข้อมูลลงบนบัสข้อมูลให้เสร็จ และจะทำการตรวจสอบด้วยว่าเกินเวลาที่กำหนดหรือไม่ หากเกินเวลาที่กำหนดก็จะทำการ

ออกจากขบวนการแฮนด์เช็ค หากยังไม่เกินก็จะรอจนหมดเวลาหรือจนกว่าสัญญาณ DAV จะเป็น Low (สถานะที่ 4) เมื่อสัญญาณ DAV มีลอจิกเป็น Low ตัวควบคุมก็จะตอบรับโดยการทำให้สัญญาณ NRFD มีค่าเป็น Low (สถานะที่ 5) เพื่อบอกให้ตัวส่งทราบว่าตัวควบคุมพร้อมที่จะเริ่มทำการเก็บข้อมูลแล้ว (สถานะที่ 6) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ EOI ว่าข้อมูลที่ตัวส่งทำการส่งมานั้นหมดหรือยัง โดยตัวส่งจะทำการเปลี่ยน สัญญาณ EOI ให้มีลอจิกเป็น Low เมื่อข้อมูลไบต์สุดท้ายถูกส่งลงไปบนบัสข้อมูล (สถานะที่ 7) หากสัญญาณ EOI ยังเป็น High ตัวควบคุมจะทำการเก็บข้อมูลในบัสข้อมูลต่อไป (สถานะที่ 8) และเมื่อตัวควบคุมได้ทำการเก็บข้อมูลเสร็จเรียบร้อยแล้ว ตัวควบคุมจะเปลี่ยนลอจิกของสัญญาณ NDAC ให้เป็น High (สถานะที่ 9) เมื่อตัวควบคุมทำให้สัญญาณ NDAC มีลอจิกเป็น High แล้ว ตัวส่งจะทำการลบข้อมูลบนบัสออกโดยการเปลี่ยนสัญญาณ DAV ให้มีค่าเป็น High (สถานะที่ 10) ตัวควบคุมก็จะทำการนำข้อมูลที่ได้ไปใช้งานและเปลี่ยนสัญญาณ NDAC ให้มีค่าเป็น Low (สถานะที่ 11) เป็นการสิ้นสุดขบวนการแฮนด์เช็ค ขบวนการแฮนด์เช็คแบบนี้สามารถเขียนแทนด้วยแผนผังเวลา ดังภาพที่ 7

รูปที่ 2.7 แสดงขบวนการแฮนด์เช็คเมื่อใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมและตัวรับ

2.5 คำสั่งใช้งานของ IEEE-488 (GPIB)

การควบคุมและกำหนดฟังก์ชันการทำงานให้แก่อุปกรณ์เครื่องมือวัดในระบบ GPIB นั้น ตัวควบคุมจะเป็นตัวกำหนด โดยการส่งรหัสคำสั่งไปยังตัวอุปกรณ์โดยผ่านทางบัสของระบบ คำสั่งสำหรับการกำหนดการทำงานต่างๆ ตามมาตรฐานของ IEEE-488 มีอยู่ 128 คำสั่ง แบ่งได้เป็น 2 กลุ่มคำสั่งใหญ่ๆ คือ กลุ่มคำสั่งหลัก (Primary Command Group) และกลุ่มคำสั่งรอง (Secondary Command Group) โดยกลุ่มคำสั่งหลักประกอบด้วย 4 กลุ่มคำสั่ง คือ กลุ่มคำสั่งเจาะจงจุดหมาย

(Addressed Command Group), กลุ่มคำสั่งครอบคลุม (Universal Command Group), กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (Listen Address Group) และกลุ่มคำสั่งกำหนดอุปกรณ์ตัวส่ง (Talk Addressed Group) ดังแสดงในตารางที่ 1 รหัสที่ใช้ในระบบ IEEE-488 นั้นสามารถที่จะใช้ร่วมกันได้ทั้งรหัสข้อมูลและรหัสคำสั่ง นั่นคือ ข้อมูลที่เหมือนกันมีความหมายได้ 2 อย่าง คือ เมื่อสัญญาณ ATN เป็น Low ข้อมูลที่อยู่ในบัสข้อมูลจะหมายถึงรหัสคำสั่ง แต่ถ้าสัญญาณ ATN เป็น High ข้อมูลที่อยู่ในบัสข้อมูลจะหมายถึงข้อมูลที่เป็นรหัส ASCI ดังแสดงในตารางที่ 1 เช่นกัน

1. กลุ่มคำสั่งเจาะจงจุดมุ่งหมาย (Address Command Group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ที่เป็นตัวส่งหรือตัวรับที่กำหนดไว้ล่วงหน้าแล้ว ประกอบด้วย

GTL (Go To Local) สั่งให้อุปกรณ์กลับไปสู่สถานะควบคุมด้วยปุ่มปรับที่หน้าปัทม์ตามปกติ

SDC (Selected Device Clear) สั่งให้อุปกรณ์กลับไปสู่สถานะเริ่มต้น

PPC (Parallel Poll Configure) เป็นคำสั่งสำหรับทำการจัดสรรสายสัญญาณของการกระทำกระบวนการตรวจสอบสภาพอุปกรณ์โดยวิธีขนาน โดยใช้งาน ร่วมกับกลุ่มคำสั่งรอง

GET (Group Execute Trigger) ใช้สั่งเริ่มต้นการทำงานของอุปกรณ์ทีละหลายๆ ตัว

TCT (Take Control) กำหนดให้อุปกรณ์ตัวส่งทำหน้าที่เป็นตัวควบคุม

2. กลุ่มคำสั่งครอบคลุม (Universal Command Group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ทุกตัวที่ต่ออยู่ในระบบ ประกอบด้วย

LLO (Local Lockout) สั่งให้อุปกรณ์ล็อคอยู่ในสถานะควบคุมด้วยปุ่มหน้าปัทม์ตามปกติ

DCL (Device Clear) สั่งให้อุปกรณ์ทุกตัวกลับไปสู่สถานะเริ่มต้น

PPU (Parallel Poll Configure) ใช้ยกเลิกกระบวนการตรวจสอบสภาพแบบขนานทั้งหมด

SPE (Serail Poll Enable) เป็นการเปลี่ยนโหมดการตรวจสอบสภาพเป็นแบบอนุกรม โดยในโหมดนี้จะเป็นการส่งสถานะของเครื่องแทนการส่งข้อมูล

SPD (Serail Poll Disable) ยกเลิกกระบวนการตรวจสอบสภาพแบบอนุกรม

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

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

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

key: ASCII character — octal 25 / hex 15 — NAK — PPU 21 — GPIB code decimal

ASCII & IEEE (GPIB) CODE CHART

B7 B6 B5 / BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0	20 DLE 16	40 LA 0 SP 32	60 LA 16 0 48	100 TA 0 @ 64	120 TA 16 P 80	140 SA 0 ` 96	160 SA 16 p 112
0 0 0 1	1 GTL SOH 1	21 LLO DC1 17	41 LA 1 ! 33	61 LA 17 1 49	101 TA 1 A 65	121 TA 17 Q 81	141 SA 1 a 97	161 SA 17 q 113
0 0 1 0	2 STX 2	22 DC2 18	42 LA 2 " 34	62 LA 18 2 50	102 TA 2 B 66	122 TA 18 R 82	142 SA 2 b 98	162 SA 18 r 114
0 0 1 1	3 ETX 3	23 DC3 19	43 LA 3 # 35	63 LA 19 3 51	103 TA 3 C 67	123 TA 19 S 83	143 SA 3 c 99	163 SA 19 s 115
0 1 0 0	4 SDC EOT 4	24 DCL DC4 20	44 LA 4 $ 36	64 LA 20 4 52	104 TA 4 D 68	124 TA 20 T 84	144 SA 4 d 100	164 SA 20 t 116
0 1 0 1	5 PPC ENQ 5	25 PPU NAK 21	45 LA 5 % 37	65 LA 21 5 53	105 TA 5 E 69	125 TA 21 U 85	145 SA 5 e 101	165 SA 21 u 117
0 1 1 0	6 ACK 6	26 SYN 22	46 LA 6 & 38	66 LA 22 6 54	106 TA 6 F 70	126 TA 22 V 86	146 SA 6 f 102	166 SA 22 v 118
0 1 1 1	7 BEL 7	27 ETB 23	47 LA 7 ' 39	67 LA 23 7 55	107 TA 7 G 71	127 TA 23 W 87	147 SA 7 g 103	167 SA 23 w 119
1 0 0 0	10 GET BS 8	30 SPE CAN 24	50 LA 8 (40	70 LA 24 8 56	110 TA 8 H 72	130 TA 24 X 88	150 SA 8 h 104	170 SA 24 x 120
1 0 0 1	11 TCT HT 9	31 SPD EM 25	51 LA 9) 41	71 LA 25 9 57	111 TA 9 I 73	131 TA 25 Y 89	151 SA 9 i 105	171 SA 25 y 121
1 0 1 0	12 LF 10	32 SUB 26	52 LA 10 * 42	72 LA 26 : 58	112 TA 10 J 74	132 TA 26 Z 90	152 SA 10 j 106	172 SA 26 z 122
1 0 1 1	13 VT 11	33 ESC 27	53 LA 11 + 43	73 LA 27 ; 59	113 TA 11 K 75	133 TA 27 [91	153 SA 11 k 107	173 SA 27 { 123
1 1 0 0	14 FF 12	34 FS 28	54 LA 12 , 44	74 LA 28 < 60	114 TA 12 L 76	134 TA 28 \ 92	154 SA 12 l 108	174 SA 28 | 124
1 1 0 1	15 CR 13	35 GS 29	55 LA 13 - 45	75 LA 29 = 61	115 TA 13 M 77	135 TA 29] 93	155 SA 13 m 109	175 SA 29 } 125
1 1 1 0	16 SO 14	36 RS 30	56 LA 14 . 46	76 LA 30 > 62	116 TA 14 N 78	136 TA 30 ^ 94	156 SA 14 n 110	176 SA 30 ~ 126
1 1 1 1	17 SI 15	37 US 31	57 LA 15 / 47	77 UNL ? 63	117 TA 15 O 79	137 UNT _ 95	157 SA 15 o 111	177 SA 31 RUBOUT DEL 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

ตารางที่ 1 แสดงคำสั่งมาตรฐาน IEEE-488 และรหัส ASCII

คำสั่งต่างๆ ซึ่งใช้ในการกำหนดสภาวะการทำงานของอุปกรณ์แต่ละสภาวะจะถูกกำหนดและมีจุดประสงค์ดังนี้ คือ

Device Clear ทำให้อุปกรณ์กลับคืนสู่สภาวะเริ่มต้น ซึ่งเป็นสภาวะที่ยังไม่มีการกำหนดฟังก์ชันใดๆ สภาวะเริ่มต้นนี้จะแตกต่างกันไป แล้วแต่ว่าอุปกรณ์นั้นได้ออกแบบมาไว้อย่างไร Device Clear แบ่งออกได้เป็น 2 ลักษณะ คือ

DCL (Device Clear) ทำการเคลียร์อุปกรณ์ทุกตัวที่ต่ออยู่

SDC (Select Device Clear) ทำการเคลียร์เฉพาะเจาะจงอุปกรณ์ตัวใดตัวหนึ่ง

แต่ว่าในการเคลียร์อุปกรณ์ให้อยู่ในสภาวะเริ่มต้นไม่ได้หมายความว่าอินเตอร์เฟสฟังก์ชันของระบบ GPIB จะถูกเคลียร์ให้กลับไปสู่สถานะเริ่มต้นด้วย เป็นเพียงแค่การเคลียร์ตัวอุปกรณ์เท่านั้น

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

Remote เป็นการกำหนดให้อุปกรณ์ที่ต่ออยู่ในระบบถูกควบคุมโดยอุปกรณ์ตัวอื่นหรือตัวควบคุมระบบ ทำให้ไม่สามารถที่จะควบคุมอุปกรณ์จากปุ่มหน้าปัทม์ของเครื่องได้

Local เป็นการกำหนดให้อุปกรณ์ที่ต่อยู่ในระบบสามารถควบคุมได้จากปุ่มหน้าปัทม์ของอุปกรณ์ตามปกติ

- การทำงานของ GPIB ในสถานะ Remote และ Local มี 4 ลักษณะ คือ

รูปที่ 2.8 แสดงการเปลี่ยนแปลงสภาวะของโหมด Remote และ Local ทั้ง 4 ลักษณะ

1. LOCS เป็นโหมด local ที่อยู่ในสภาพการควบคุมจากหน้าปัทม์ตามปกติ ซึ่งอุปกรณ์จะอยู่ในสภาวะนี้เมื่อเริ่มเปิดสวิทช์ของอุปกรณ์ หรือสัญญาณ REN มีลอจิกเป็น High หรือเมื่ออุปกรณ์ได้รับคำสั่ง GTL (Go to Local)

2. REMS เป็นโหมด Remote หมายถึงการตัดการควบคุมอุปกรณ์ออกจากการควบคุมด้วยปุ่มหน้าปัทม์ 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 เมื่อได้รับคำสั่งกำหนดให้เป็นอุปกรณ์ตัวรับ ก็จะทำการเปลี่ยนไปในสภาวะการ Remote แบบที่ถูกล็อคทันที การจะเข้าสู่สภาวะแบบ LWLS มีอยู่ 2 วิธี คือ

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

การเปลี่ยนแปลงสภาวะของการ Remote หรือ Local ทั้ง 4 ลักษณะแสดงดังภาพที่ 8 เมื่อสัญญาณ REN มีลอจิกเป็น High อุปกรณ์ก็จะอยู่ในสภาวะ Local ทันทีไม่ว่าสถานะเดิมจะเป็นอย่างไรก็ตาม แต่เมื่อสัญญาณ REN เป็น Low แล้ว หากไม่มีคำสั่งกำหนดอุปกรณ์ตัวรับ หรือ คำสั่ง LLO เข้ามาอุปกรณ์ก็ยังสถานะ Local อยู่เช่นเดิม

บทที่ 3
คุณสมบัติโดยทั่วไปของระบบบัส USB

3.1 คุณสมบัติเด่นของระบบบัส USB

ระบบบัส USB (Universal Serial Bus) นั้น เป็นระบบบัสที่มีความยืดหยุ่นสูง ปราศจากข้อจำกัดและการขัดขวางของการอินเตอร์เฟสทางด้านฮาร์ดแวร์ ซึ่งการตั้งค่าการทำงานต่างๆ จะถูกกระทำโดยระบบปฏิบัติการโดยอัตโนมัติ นอกจากนี้การเชื่อมต่ออุปกรณ์ USB เข้ากับระบบนั้นสามารถทำได้ทั้งในขณะที่เครื่องคอมพิวเตอร์ยังคงทำงานอยู่ ซึ่งกล่าวได้ว่าเป็นรูปแบบของปลั๊กแอนด์เพลย์ (Plug and Play) อย่างแท้จริง อีกทั้งความเร็วในการส่งถ่ายข้อมูลใน USB 2.0 นั้นก็มีความเร็วสูงกว่าการส่งถ่ายข้อมูลแบบขนานและอนุกรมแต่เดิมเป็นอย่างมาก และในการเพิ่มจำนวนของพอร์ตนั้นก็สามารถทำได้โดยง่ายเพียงนำ USB Hub มาต่อพ่วงเข้ากับระบบเท่านั้น

คุณสมบัติเด่นในระบบบัส USB ได้แก่

- สามารถนำอุปกรณ์ I/O มาต่อพ่วงเข้าสู่ระบบได้ในขณะที่เครื่องคอมพิวเตอร์ยังคงทำงานอยู่ได้ (Hot-Pluggable)
- ง่ายต่อการใช้งาน เนื่องจากเครื่องคอมพิวเตอร์จะมีความสามารถในการทำความรู้จักและจดจำอุปกรณ์ต่างๆที่นำมาต่อพ่วงในระบบโดยไดร์ฟเวอร์ที่เหมาะสม และในการตั้งค่าต่างๆ ก็เป็นไปโดยอัตโนมัติ
- ใช้คอนเน็คเตอร์ในการเชื่อมต่อเพียงชนิดเดียวจึงลดความสับสนในการเชื่อมต่อ
- มีประสิทธิภาพในการส่งถ่ายข้อมูลสูง โดยเป็นไปตามมาตรฐานดังนี้

  ⇒ *มาตรฐาน USB 1.0/1.1* มีระดับความเร็วในการส่งถ่ายข้อมูล 2 ระดับได้แก่ ที่ระดับความเร็วต่ำ (Low Speed) เท่ากับ 1.5 Mbit/Sec และที่ระดับความเร็วเต็มที่ (Full Speed) เท่ากับ 12 Mbit/Sec

  ⇒ *มาตรฐาน USB 2.0* จะมีการเพิ่มเติมระดับความเร็วในการส่งถ่ายขึ้นอีก 1 ระดับ ได้แก่ ที่ระดับความเร็วสูง (High Speed) มีความเร็วในการส่งถ่ายข้อมูลเท่ากับ 480 Mbit/sec

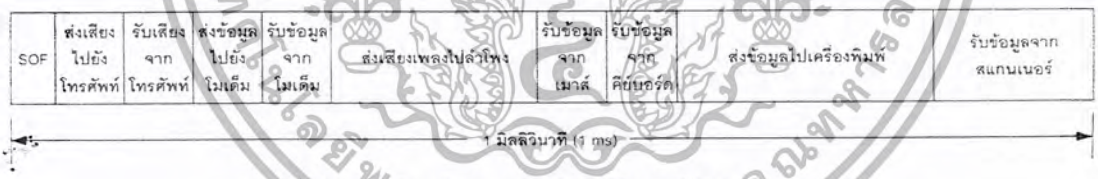
- ไม่เกิดการขัดแย้งกันของการเข้าใช้ทรัพยากรของระบบ (IRQ : Interrupt Request) ซึ่งเป็นการแก้ปัญหาทางด้านข้อจำกัดของจำนวนอุปกรณ์ที่มาเชื่อมต่อ
- สามารถต่ออุปกรณ์ภายในระบบได้สูงสุด 127 ตัว
- สายเคเบิลของ USB นั้น จะมีสายของแหล่งจ่ายกำลังงานรวมอยู่ภายใน และสามารถนำมาใช้งานได้

- มีการจัดการกับระบบพลังงานที่ชาญฉลาด ซึ่งกำลังงานบนบัสจะถูกลดระดับลงเมื่อไม่ได้มีการใช้งานเป็นระยะเวลาหนึ่ง

- มีความสามารถในการตรวจสอบและแก้ไขข้อผิดพลาดของข้อมูลโดยอัตโนมัติ

3.2 การส่งถ่ายข้อมูลบนระบบบัส USB

USB เป็นการสื่อสารข้อมูลในรูปแบบอนุกรมรูปแบบหนึ่งซึ่งอุปกรณ์ทุกๆ ตัวจะต้องส่งสัญญาณรวมกันไปในสายส่งสัญญาณเพียงคู่เดียว ดังนั้นอุปกรณ์ทุกๆ ตัวที่เชื่อมต่อกับบัสจะต้องส่งข้อมูลเรียงลำดับกันไปเพื่อไม่ให้เกิดการชนกันของข้อมูล และเนื่องจาก USB เป็นระบบบัสที่ใช้สายส่งสัญญาณเพียงคู่เดียว ทำให้ในช่วงเวลาหนึ่งๆ จะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียวเท่านั้น ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันได้หรือที่เรียกกันว่า การส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ (half duplex) โดยจังหวะการรับส่งข้อมูลของระบบบัส USB ทั้งหมดจะถูกควบคุมจากโฮสต์ (host) ซึ่งก็คือเครื่องคอมพิวเตอร์ที่เป็นจุดรวมของอุปกรณ์ทุกตัวที่เชื่อมต่ออยู่นั่นเอง ดังนั้นจึงไม่สามารถเชื่อมต่ออยู่นั่นเอง ดังนั้นจึงไม่สามารถเชื่อมต่อเครื่องคอมพิวเตอร์ 2 เครื่อง ให้รับหรือส่งข้อมูลถึงกันได้โดยตรง เพราะถ้าคอมพิวเตอร์ทั้งสองเครื่องทำหน้าที่เป็นโฮสต์ทั้งคู่จะเกิดการชนกันของข้อมูลภายในบัส เนื่องจากแต่ละเครื่องก็จะพยายามกำหนดจังหวะในการรับส่งของตัวเองขึ้นมา ดังนั้นจะเชื่อมต่อคอมพิวเตอร์ 2 เครื่องเข้าด้วยกันผ่าน USB จะต้องมีอุปกรณ์ที่เป็นตัวกลางเพื่อซิงโครไนซ์ตัวเองเข้ากันโฮสต์ทั้งสองให้ได้

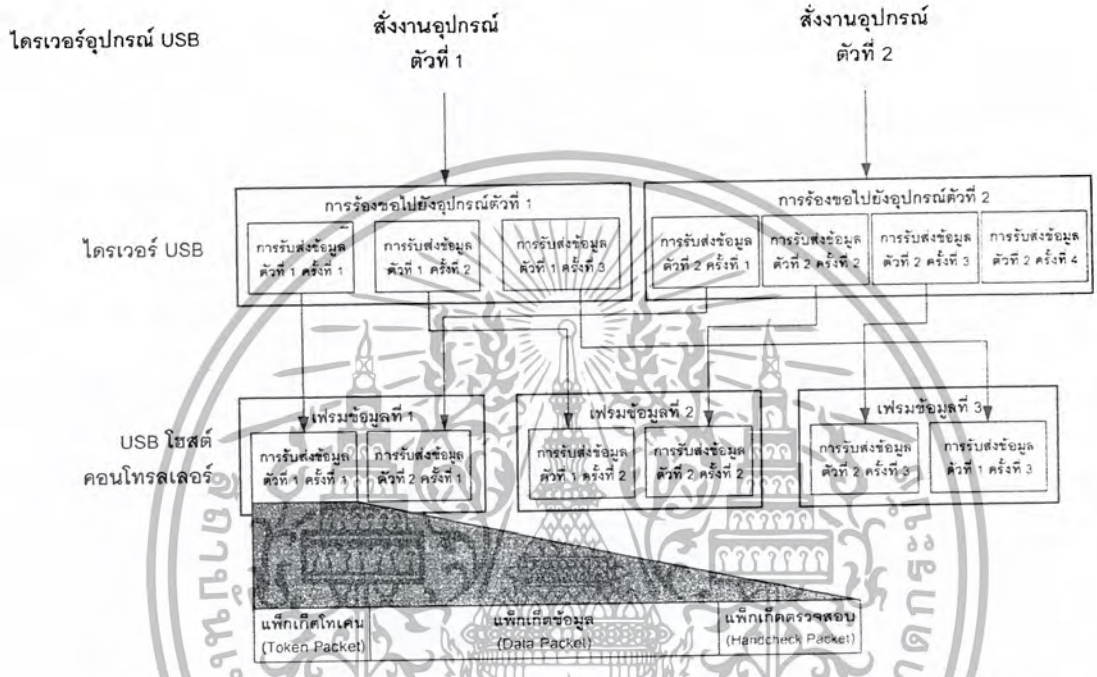


รูปที่ 3.1 แสดงรูปแบบการจัดลำดับการส่งข้อมูลบนบัส

การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรม โดยทุกๆ 1 มิลลิวินาที(ms) จะเกิดการรับส่งข้อมูลขึ้น 1 เฟรม ในแต่ละเฟรมจะแบ่งย่อยออกเป็นแพ็กเก็ต (packet) เริ่มต้นการทำงานของแต่ละเฟรมโดยโฮสต์จะต้องส่งสัญญาณเริ่มต้นเฟรมหรือ SOF (Start Of Frame) ออกไปเพื่อให้อุปกรณ์ทุกตัวรู้จังหวะการเริ่มเฟรม หลังจากนั้นโฮสต์ก็จะเริ่มส่งหรือรับข้อมูลต่างๆ ตามที่ได้จัดลำดับความสำเร็จไว้อุปกรณ์ต่าง ๆ ที่อยู่ภายในบัสจะต้องทำงานตามจังหวะที่โฮสต์กำหนดไว้เท่านั้น การส่งข้อมูลกลับไปยังโฮสต์จะสามารถทำได้ก็ต่อเมื่อได้รับการถามหรือร้องขอจากโฮสต์ แต่เนื่องจากแต่ละเฟรมข้อมูลจะต้องรับส่งเสร็จภายใน 1 มิลลิวินาที นั่นหมายความว่าข้อมูลของอุปกรณ์ทุกๆ ตัวที่เชื่อมตัวต่อกับบัสจะต้องถูกกำหนดขนาดไม่ให้ใหญ่เกินกว่าที่จะสามารถรับส่งได้ภายใน 1 มิลลิวินาที และเล็กพอที่จะทำให้อุปกรณ์ทุกๆ ตัวสามารถใช้งานบัสไปพร้อมๆ กันได้ ดังนั้นใน

ระบบบัส USB จึงจำเป็นต้องอาศัยซอฟต์แวร์ที่เข้ามาจัดการในด้านนี้ และยังต้องอาศัยฮาร์ดแวร์ที่จะคอยกระจายการส่งและรวบรวมการรับข้อมูลจากอุปกรณ์ทุกๆ ตัวในระบบ โดยแบ่งเป็นองค์ประกอบทางด้านซอฟท์แวร์และฮาร์ดแวร์ดังต่อไปนี้

3.3 องค์ประกอบทางด้านซอฟท์แวร์



รูปที่ 3.2 แสดงองค์ประกอบและการทำงานทางด้านซอฟต์แวร์

3.3.1 ไดรเวอร์อุปกรณ์ USB (USB Device Drivers)

ไดรเวอร์อุปกรณ์ USB คือโปรแกรมเก็บข้อมูลที่จำเป็นในการติดต่อไปยังอุปกรณ์แต่ละตัว เมื่อโปรแกรมใดมีความต้องการจะติดต่อกับอุปกรณ์ต่างๆ จะต้องแจ้งความต้องการนั้นๆ มายังไดรเวอร์อุปกรณ์ USB เนื่องจากตัวไดรเวอร์นี้จะรู้ว่าถ้าต้องการติดต่อกับอุปกรณ์จะต้องติดต่อผ่านเอ็นด์พอยต์ (Endpoint) ไหน ด้วยรูปแบบใด ดังนั้นอุปกรณ์แต่ละตัวก็จะมีไดรเวอร์อุปกรณ์ USB เฉพาะตัว ซึ่งเมื่อถึงคราวต้องต้องนำอุปกรณ์นั้นมาต่อใช้งานกับเครื่องคอมพิวเตอร์จริงๆ ก็จะต้องนำไดรเวอร์ตัวเดียวกันมาติดตั้งเพิ่มเข้ากับระบบปฏิบัติการในคอมพิวเตอร์เพื่อให้ระบบรู้จักและติดต่อใช้งานอุปกรณ์ที่ติดตั้งเข้ามาใหม่นี้ได้ เช่น ถ้าต้องการติดต่อเพื่อรับข้อมูลจากคีย์บอร์ด ตัวไดรเวอร์อุปกรณ์ USB จะรู้ว่าต้องรับส่งข้อมูลด้วยอัตราเร็วต่ำ(slow spped) โดยใช้รูปแบบการถ่ายทอดข้อมูลแบบอินเตอร์รัปต์(interrupt transfer type) ผ่านเอ็นด์พอยต์ตัวหนึ่งของคีย์บอร์ด และตรวจสอบข้อมูลการกดเป็นช่วงระยะห่างค่าหนึ่ง แต่ในบางอุปกรณ์ที่เป็นอุปกรณ์พื้นฐานของ

เครื่องคอมพิวเตอร์ เช่น เมาส์และคีย์บอร์ดจะมีการบรรจุไดรเวอร์อุปกรณ์USB ของอุปกรณ์เหล่านี้ไว้ภายในไบออสของเครื่องคอมพิวเตอร์เรียบร้อยแล้ว จึงไม่ต้องติดตั้งไดรเวอร์เพิ่มเติมสำหรับอุปกรณ์เหล่านี้ เพียงแต่เข้าไปเปิดการทำงาน ไบออสก็จะทำให้เครื่องคอมพิวเตอร์รู้จักอุปกรณ์เหล่านี้โดยอัตโนมัติ

3.3.2 ไดร์เวอร์ USB (USB Drivers)

ไดร์เวอร์ USB นั้นเป็นส่วนของซอฟต์แวร์ที่ทำหน้าที่ในการจัดการแบ่งสันปันส่วนช่องสัญญาณในแต่ละเฟรม ให้อุปกรณ์แต่ละตัวนั้นสามารถส่งข้อมูลร่วมกันไปในสายสัญญาณคู่เดียวได้ในเวลาเดียวกัน ไดรเวอร์อุปกรณ์ USB ของอุปกรณ์แต่ละตัวจะส่งการร้องรอเพื่อการติดต่อ (request) ลงมายังไดรเวอร์USB และเมื่อไดรเวอร์USB รับทราบความต้องการติดต่อของอุปกรณ์ครบทุกๆ ตัวที่เชื่อมต่ออยู่กับบัสแล้ว ก็จะพิจารณาว่า ในรอบการรับส่งข้อมูลหนึ่งๆ นั้นอุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้มากเท่าใด หากปริมาณข้อมูลที่ต้องการรับส่งมีขนาดมากก็จะตัดแบ่งออกเป็นส่วนๆ แล้วเก็บไว้เพื่อรอส่งในรอบถัดไป โดยปริมาณข้อมูลที่ส่งได้ของอุปกรณ์แต่ละตัวจะถูกพิจารณาจากชนิดของการถ่ายทอดข้อมูล (transfer type) ว่า อุปกรณ์ใดใช้การถ่ายทอดข้อมูลแบบใดและการรับส่งข้อมูลชนิดนั้นมีลำดับความสำคัญมากน้อยเพียงใด

3.3.3 ไดร์เวอร์โฮสต์คอนโทรลเลอร์ (USB Host Controller Driver)

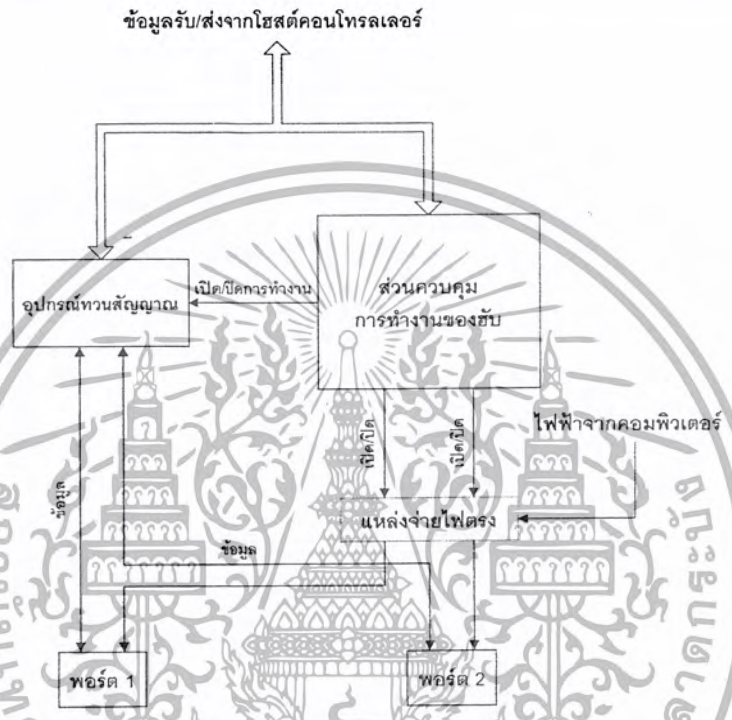
หลังจากไดรเวอร์ USB พิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้าง มันจะส่งข้อมูลของอุปกรณ์แต่ละตัวที่จะติดต่อในรอบการติดต่อนั้นๆ มายังไดรเวอร์โฮสต์คอนโทรลเลอร์ จากนั้นไดรเวอร์โฮสต์คอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลของอุปกรณ์แต่ละชนิดลงเป็นเฟรมข้อมูล เพิ่มเติมส่วนประกอบต่างๆ ของเฟรมข้อมูลให้ครบตามมาตรฐานการถ่ายทอดข้อมูลแบบ USB แล้วส่งข้อมูลทั้งหมดไปยังฮาร์ดแวร์ USB โฮสต์คอนโทรลเลอร์เพื่อส่งข้อมูลทั้งหมดออกไปยังอุปกรณ์ต่างๆ ในรูปที่ 3.2 แสดงลำดับและขั้นตอนการทำงานของซอฟต์แวร์ควบคุมการทำงานของพอร์ตUSB

3.4 องค์ประกอบทางด้านฮาร์ดแวร์

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

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

- ควบคุมการใช้พลังงานของอุปกรณ์ที่มาต่อร่วม
- ตรวจสอบการเชื่อมต่อของอุปกรณ์ว่ามีอุปกรณ์ต่อร่วมอยู่หรือไม่
- เปิดหรือเอ็นเอเบิลการใช้งานพอร์ตเมื่อมีอุปกรณ์ต่ออยู่ และปิดหรือดิสเอเบิลการใช้งานเมื่อปลดอุปกรณ์ออกไปแล้ว
- รายงานสถานะของแต่ละพอร์ตเมื่อไดรเวอร์โฮสต์คอนโทรลเลอร์ร้องขอมา



รูปที่ 3.3 บล็อคไดอะแกรมแสดงการทำงานของ USB รูทฮับอย่างง่าย

3.4.2 USB ฮับ (USB Hub)

หน้าที่หลักๆของ USBฮับคือ ขยายการเชื่อมต่อให้อุปกรณ์จำนวนมากๆ สามารถเชื่อมต่อเข้ากับระบบบัสได้ โดยการทำงานหลักของ USB ฮับนั้นมีอยู่ 2 ส่วนคือ ทำหน้าที่เป็นตัวทวนสัญญาณ (repeater) และตัวจัดการพลังงาน (power management) ในส่วนของการทวนสัญญาณ USB ฮับจะต้องรับสัญญาณจากโฮสต์มา แล้วส่งกระจายออกไปยังพอร์ตทุกๆ พอร์ต และรับสัญญาณจากแต่ละพอร์ต แล้วจับมารวมกันเพื่อส่งกลับไปที่โฮสต์สำหรับส่วนของการจัดการพลังงานนั้นมีหน้าที่เหมือนกับรูทฮับก็คือ ตรวจสอบว่ามีการต่ออยู่ของอุปกรณ์ที่พอร์ตใดบ้าง หากมีอุปกรณ์ต่ออยู่ก็เปิดการใช้งานพอร์ตนั้นๆ หากไม่มีอุปกรณ์ต่ออยู่ก็ปิดการใช้งาน ตรวจสอบการเชื่อมต่อหรือปลดออกของอุปกรณ์เพื่อรายงานผลเมื่อโฮสต์คอนโทรลเลอร์ร้องขอ และป้องกันอุปกรณ์ที่ต่ออยู่ในแต่ละพอร์ตไม่ให้ดึงกระแสไฟฟ้าเกินกว่าที่กำหนด

3.4.3 อุปกรณ์ USB (USB Device)

ส่วนประกอบนี้นั้นก็คือส่วนของอุปกรณ์ต่างๆที่นำมาต่อพ่วงกับพอร์ต USB นั่นเอง โดยการจัดประเภทของอุปกรณ์ USB นั้น สามารถจัดประเภทได้จากความเร็วในการส่งถ่ายข้อมูล และรูปแบบการใช้พลังงานของอุปกรณ์ โดยการจัดประเภทจากความเร็วในการส่งถ่ายข้อมูลนั้นแต่เดิมสามารถจัดได้เป็น 2 ประเภทได้แก่

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

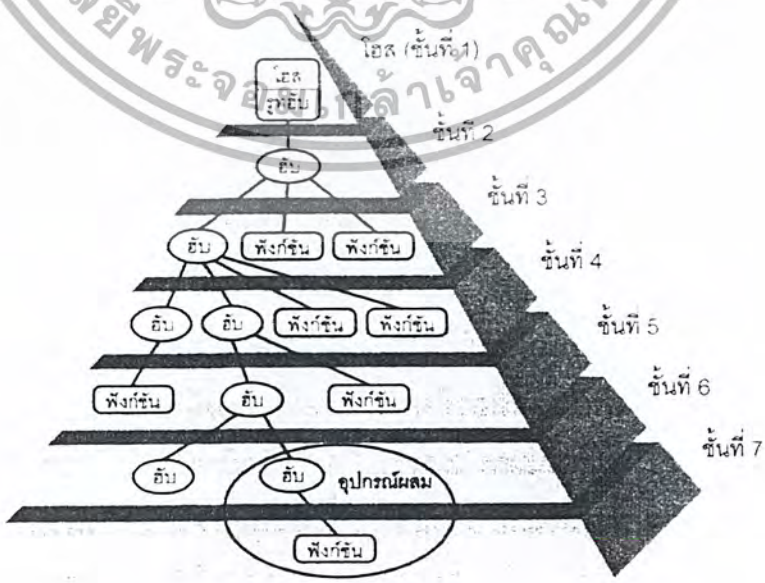
- อุปกรณ์ความเร็วเต็มที่ (Full-Speed Devices) เช่น เครื่องพิมพ์, กล้องดิจิตอล, ซีดีรอมไดร์ฟ, เครื่องเล่น MP3 เป็นต้น ซึ่งจะใช้ความเร็วในการส่งถ่ายข้อมูลเท่ากับ 12 Mbit/sec

แต่ในปัจจุบันนั้น ได้มีการเพิ่มเติมส่วนของมาตรฐาน USB2.0 ซึ่งมีความเร็วการส่งถ่ายข้อมูลในระดับความเร็วสูง (High Speed) เท่ากับ 480 Mbit/sec เพิ่มเติมขึ้นมา ซึ่งจะกล่าวถึงต่อไปในภายหลัง สำหรับการจัดประเภทของอุปกรณ์จากการใช้พลังงานของตัวมันนั้น จะแบ่งออกได้เป็น 2 ประเภทได้แก่

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

- อุปกรณ์ที่ใช้พลังงานจากตัวเอง (Self Powered Device) ได้แก่อุปกรณ์ที่ใช้แหล่งจ่ายพลังงานจากตัวเองโดยไม่พึ่งพลังงานจากระบบบัส

นอกจากนี้ยังมีอุปกรณ์ USB บางประเภทที่มีคุณสมบัติของ USB ฮับอยู่ด้วย นั่นคือสามารถนำอุปกรณ์อื่นๆ มาเชื่อมต่อเข้ากับตัวมันได้ เราเรียกอุปกรณ์ประเภทนี้ว่า Compound USB Device



รูปที่ 3.4 แสดงโครงสร้างการเชื่อมต่อบนระบบบัส USB

3.5 รูปแบบการส่งถ่ายข้อมูลบนระบบบัส USB

การถ่ายทอดสัญญาณ(tranfer type) ของบัส USB นั้นแบ่งออกเป็น 4 ชนิดตามขนาดชนิดของข้อมูลและจังหวะการส่งข้อมูลดังนี้

1.การถ่ายทอดสัญญาณแบบไอโซโครนัส(Isochronous tranfer)

2.การถ่ายทอดสัญญาณแบบบัลก์(Bulk transfer)

3.การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์(Interrupt transfer)

4.การถ่ายทอดสัญญาณควบคุม(Control tranfer)

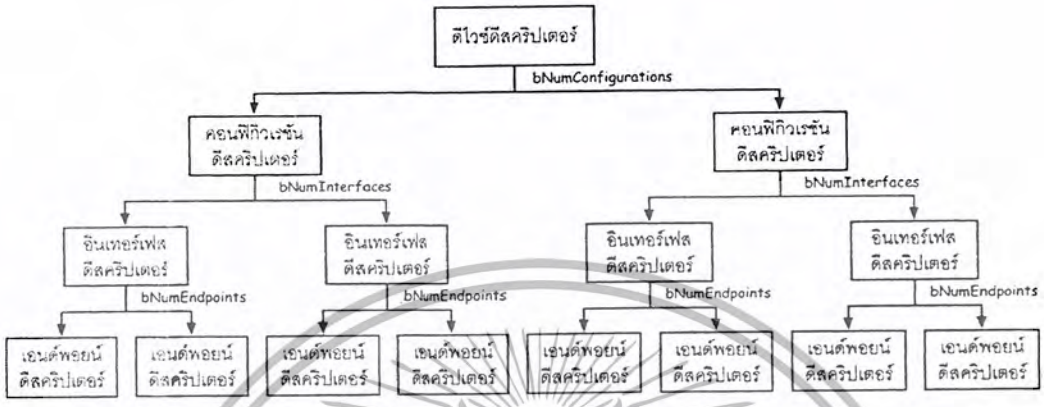
การถ่ายทอดสัญญาณ 3 ชนิดแรกใช้สำหรับข้อมูลทั่วไปที่ต้องการรับหรือส่งไปยังตัวอุปกรณ์ส่วนการถ่ายทอดสัญญาณแบบที่ 4 การถ่ายทอดสัญญาณแบบไอโซโครนัสใช้ถ่ายทอดข้อมูลที่ต้องการความต่อเนื่องสูง เช่น ข้อมูลสเสียงเพลง ส่วนการถ่ายทอดสัญญาณแบบบัลก์ใช้สำหรับถ่ายทอดข้อมูลที่มีปริมาณมากๆ แต่ไม่ต้องการความต่อเนื่องของข้อมูล ในขณะที่การถ่ายทอดสัญญาณแบบอินเตอร์รัปต์ใช้สำหรับถ่ายทอดข้อมูลที่มีจำนวนน้อยครั้งและมีปราณของข้อมูลไม่มาก

ในการสั่งงานอุปกรณ์แต่ละครั้งนั้น โฮสต์จะต้องระบุเป้าหมายปลายทางของข้อมูลที่ต้องการจะรับหรือส่ง เป้าหมายปลายทางที่ว่านี้เป็นกลุ่มของรีจิสเตอร์ที่อยู่ในอุปกรณ์ชนิดต่างๆ 4 ชนิดข้างต้นแยกกันออกไป ดังนั้นอุปกรณ์แต่ละตัวจะมีจำนวนเอ็นด์พอยต์มากกว่า 1 เอ็นด์พอยต์เพื่อรองรับการทำงานรูปแบบต่างๆ ยกตัวอย่าง ซีดีรอม USB แบบอ่านเขียนได้จะต้องมีเอ็นด์พอยต์ที่รองรับการถ่ายทอดสัญญาณควบคุมเพื่อรับคำสั่งจากโฮสต์ 1 เอ็นด์พอยต์ พร้อมกันนั้นยังต้องมีเอ็นด์พอยต์ที่รองรับการถ่ายทอดสัญญาณแบบบัลก์เพื่อส่งข้อมูลที่อ่านได้หรือรับข้อมูลเพื่อเขียนแผ่นซีดี 1 เอ็นด์พอยต์ และต้องมีเอ็นด์พอยต์ที่รองรับการถ่ายทอดสัญญาณไอโซโครนัส เพื่อส่งข้อมูลเพลงซึ่งมีความต่อเนื่องในกรณีที่เล่นแผ่นซีดีเพลงอีก 1 เอ็นด์พอยต์ เป็นต้น

3.6 ดิสคริปเตอร์ : ความหมาย , ชนิดและรูปแบบการทำงาน

อุปกรณ์แต่ละตัวมีคุณสมบัติและการทำงานที่แตกต่างกัน โฮสต์จำเป็นต้องรู้คุณสมบัติทั้งหมดของอุปกรณ์แต่ละตัวเพื่อให้การสั่งงานเป็นไปอย่างถูกต้อง และเนื่องจากบัสข้อมูลทั้งหมดจะถูกใช้งานรับส่งข้อมูลร่วมกันระหว่างอุปกรณ์ทุกๆ ตัว สิ่งที่โฮสต์จำเป็นต้องรู้อีกอย่างคือปริมาณข้อมูลที่ต้องการส่ง(bandwidth) ของอุปกรณ์แต่ละตัวในบัส ดังนั้นเมื่อมีอุปกรณ์ตัวใหม่ต่อเข้ากับบัสโฮสต์ต้องอ่านข้อมูลต่างๆ ที่จำเป็นเข้ามาเพื่อใช้อ้างอิงในการสั่งงานอุปกรณ์ และใช้พิจารณาว่าระบบบัสสามารถรองรับอุปกรณ์ที่มาเชื่อมต่อใหม่ได้หรือไม่ ข้อมูลเหล่านี้รวมเรียนว่าดิสคริปเตอร์ของอุปกรณ์ (device descripters) ซึ่งอุปกรณ์แต่ละตัวจะแจ้งรายละเอียดของตัวเองให้โฮสต์รู้ผ่านดิสคริปเตอร์ชนิดต่างๆ ซึ่งได้รับการแบ่งแยกเป็นชนิดตามข้อมูลที่จะแจ้งกลับไปยังโฮสต์ โดยการแบ่งแยกชนิดของดิสคริปเตอร์นั้นจะจัดเป็นลำดับชั้นดังรูปที่ 3.5 สาเหตุที่ต้อง

จัดเป็นระดับชั้นเป็นเพราะว่าอุปกรณ์แต่ละตัวนั้นอาจมีการทำงานที่หลากหลายรูปแบบ เช่น แบ่งการทำงานออกเป็น 2 โหมด แต่ละโหมดมีหน้าที่การทำงานแตกต่างกัน และแต่ละหน้าที่ก็จะใช้กลุ่มของเอ็นด์พอยต์ที่แตกต่างกันซึ่งสามารถสรุปได้ดังนี้



รูปที่ 3.5 แสดงระดับการทำงานของดิสคริปเตอร์ในระบบบัส USB

3.6.1 ดีไวซ์ดิสคริปเตอร์ (Device descripter)

ทำหน้าที่หลักในการเก็บข้อมูลโดยทั่วไปของตัวอุปกรณ์ ซึ่งในอุปกรณ์แต่ละตัวจะมีดีไวซ์ดิสคริปเตอร์เพียง 1 ชุดเท่านั้น ภายในจะระบุข้อมูลที่ใช้ในการเชื่อมต่อขั้นแรก (default communications pipe) เพื่อใช้ในการกำหนดข้อมูลสถานะของอุปกรณ์เข้า กับโฮสต์ นอกจากนั้นยังเก็บข้อมูลของข้อกำหนดในโหมดการทำงานต่างๆ ของตัวอุปกรณ์รวมถึงจำนวนคอนฟิกิวเรชั่นด้วย เพราะในครั้งแรกที่อุปกรณ์เชื่อมต่อเข้ากับบัสนั้น โฮสต์ไม่มีทางรู้ได้เลยว่า ต้องติดต่อกับอุปกรณ์ที่เอ็นด์พอยต์ใด จึงจำเป็นต้องขอข้อมูลส่วนนี้ก่อนที่จะติดต่อกับส่วนอื่น

3.6.2 คอนฟิกิวเรชั่นดิสคริปเตอร์ (Configuration descripter)

ใช้เก็บข้อมูลที่จำเป็นของการทำงานในแต่ละโหมดการทำงานและเก็บจำนวนอินเตอร์เฟสที่ใช้งานในโหมดนั้นๆ เช่น อุปกรณ์บางตัวมีการทำงาน 2 โหมดคือ โหมดใช้พลังงานสูง และโหมดประหยัดพลังงาน ดิสคริปเตอร์ตัวนี้จะเก็บข้อมูลที่จำเป็นในการตั้งค่าต่างๆ ของโฮสต์เมื่อต้องการเลือกใช้โหมดการทำงานแต่ละโหมดของอุปกรณ์

3.6.3 อินเตอร์เฟสดิสคริปเตอร์ (Interface descripter)

ใช้เก็บข้อมูลหน้าที่การทำงานภายใน ซึ่งในแต่ละโหมดการทำงานหรือคอนฟิกิวเรชั่นอาจจะมีการเชื่อมต่อหรืออินเตอร์เฟสเพียง 1 แบบหรือมากกว่าเพื่อใช้งานในหน้าที่ต่างๆ ตัวอย่างที่เห็นได้ชัดที่สุดก็คือซีดีรอม โดยในตัวซีดีรอมจะมีการรับส่งข้อมูลปริมาณมากๆ(mass storage), การ

ส่งข้อมูลเสียงออดิโอ และการส่งข้อมูลภาพจะเห็นได้ว่า มีอินเตอร์เฟสที่ใช้งานแยกกัน ภายในอินเตอร์เฟสดิสคริปเตอร์จะบรรจุข้อมูลการใช้งานอินเตอร์เฟสนั้นๆ โดยข้อมูลเหล่านี้จะระบุว่าอุปกรณ์ถูกจัดอยู่ในคลาส(class) หรือคลาสย่อย(subclass)ใด และมีเอ็นด์พอยต์จำนวนเท่าใดที่ใช้งานในอินเตอร์เฟสนี้บ้าง

3.6.4 เอ็นด์พอยต์ดิสคริปเตอร์ (Endpoint descripter)

ใช้เก็บข้อมูลคุณสมบัติของแต่ละเอ็นด์พอยต์ เช่น ใช้การถ่ายทอดสัญญาณแบบใด (ไอโซโครนัส, บัลก์, อินเตอร์รัปต์ หรือสัญญาณควบคุม) และถ่ายทอดข้อมูลได้มากที่สุดครั้งละเท่าใด

3.6.5 สตริงค์ดิสคริปเตอร์ (String descripter)

เป็นดิสคริปเตอร์ที่ไม่ได้อยู่ภายในโครงสร้างตามรูปที่ 2.5 เพราะดิสคริปเตอร์ชนิดนี้จะเก็บข้อมูลตัวอักษรที่สามารถอ่านเข้าใจได้ไว้อธิบายส่วนต่างๆ ของดิสคริปเตอร์ทั้งสี่ตัวข้างต้น เช่น เก็บชื่อบริษัทผู้ผลิต และ/หรือหมายเลขประจำตัวของอุปกรณ์ เป็นต้น

3.6.6 คลาสสเปกซิฟิกดิสคริปเตอร์(Class-specific descripter)

เป็นดิสคริปเตอร์พิเศษที่มีเฉพาะในอุปกรณ์บางตัวที่จัดอยู่ในบางคลาสเท่านั้น ภายในเก็บข้อมูลเฉพาะของคลาสนั้นๆ ที่อยู่นอกเหนือจากดิสคริปเตอร์พื้นฐาน 4 ชนิดแรก

3.7 การจัดการกับอุปกรณ์บนบัส USB ที่มีความเร็วต่างกัน

อุปกรณ์USB แบ่งตามความเร็วของการถ่ายทอดข้อมูลได้ 2 ชนิดคือ อุปกรณ์ความเร็วเต็มที่จะถ่ายทอดข้อมูลที่อัตรา 12Mb/s และอุปกรณ์ความเร็วต่ำถ่ายทอดข้อมูลที่อัตรา 1.5 Mb/s แต่เนื่องจากอุปกรณ์ทั้งสองประเภทนี้ต่ออยู่บนบัสเดียวกันทั้งหมด ดังนั้นพอร์ตของฮับที่ให้อุปกรณ์ USB เข้ามาเชื่อมต่อจะต้องรองรับการทำงานได้ทั้ง 2 แบบ และด้วยความที่เป็นระบบบัส อุปกรณ์ทุกๆ ตัวจะได้รับข้อมูลทุกๆ แพ็กเก็ตที่ส่งเข้ามา ไม่ว่าจะเป็นการส่งด้วยความเร็วสูงหรือต่ำ ดังนั้นจึงต้องมีการจัดการจราจรระหว่างข้อมูลที่ส่งด้วย โดยตั้งเป็นข้อกำหนดว่า พอร์ตของอุปกรณ์ความเร็วต่ำต้องไม่เปิดทำงานจนกว่าจะได้รับสัญญาณ “Preamble” จากโฮสต์ โดยหลังจากได้รับสัญญาณ Preamble แล้ว ฮับจะเปลี่ยนความเร็วในการถ่ายทอดข้อมูลไปสู่โหมดความเร็วต่ำ แล้วเปิดการทำงานของพอร์ตที่ความเร็วต่ำ ทำให้อุปกรณ์ที่เชื่อมต่อด้วยความเร็วต่ำได้รับข้อมูล ในทางกลับกัน หลังจากได้รับสัญญาณ Preamble แล้วอุปกรณ์ความเร็วเต็มที่ก็จะทราบทันทีว่าข้อมูลที่ตามหลังมาจะอยู่ในโหมดความเร็วต่ำ ไม่ต้องอ่านและตีความข้อมูลเหล่านั้น

3.8 การส่งสัญญาณในบัสUSB

สัญญาณที่ปรากฏบนสายสัญญาณระหว่าง รูตฮับและอุปกรณ์จะส่งไปในแบบสัญญาณผลต่าง (differential signaling) เพื่อลดการแพร่กระจายสนามแม่เหล็กไฟฟ้า (EMI:Electromagnetic Interference) เนื่องจากตามธรรมชาติของสัญญาณไฟฟ้า เมื่อมีการเปลี่ยนแปลงระดับสัญญาณด้วยความเร็วมากๆ จะทำให้เกิดการแพร่กระจายของสนามแม่เหล็กและสนามไฟฟ้าออกมารอบๆ สายส่งสัญญาณ ซึ่งอาจรบกวนการทำงานของอุปกรณ์ต่างๆ รอบข้างได้ด้วยการส่งสัญญาณแบบนี้จะทำให้การเปลี่ยนแปลงของสัญญาณในสายนำสัญญาณเกิดขึ้นมาพร้อมกันและเกิดในลักษณะตรงข้ามทำให้สนามแม่เหล็กไฟฟ้าที่เกิดขึ้นหักล้างกัน ไม่แพร่ออกมาภายนอก

3.9 กระบวนการกำหนดการทำงานของอุปกรณ์

ในส่วนนี้เป็นลำดับขั้นตอนในการเชื่อมต่ออุปกรณ์ USB ไปยังคอมพิวเตอร์ในเบื้องต้น โดยมีลำดับขั้นตอนดังต่อไปนี้

- ฮับตรวจสอบพบว่ามีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้าสู่ระบบ แล้วแจ้งผลกลับไปยังโฮสต์คอนโทรลเลอร์

- โฮสต์คอนโทรลเลอร์สั่งให้ฮับเปิดการทำงานของแหล่งจ่ายไฟในโหมดประหยัดเพื่อให้อุปกรณ์ที่อาศัยพลังงานจากบัสสามารถทำงานได้

- โฮสต์คอนโทรลเลอร์สั่งให้ฮับรีเซตพอร์ตที่อุปกรณ์รีเซตค่าแอดเดรสและเอ็นด์พอยต์ของตัวเองให้เป็นค่าเริ่มต้น(default)

- โฮสต์อ่านดิสคริปเตอร์ต่างๆ จากตัวอุปกรณ์และพิจารณาว่าทรัพยากรของระบบพอเพียงต่อความต้องการของตัวอุปกรณ์หรือไม่ ทรัพยากรในที่นี้คือ พลังงานไฟฟ้าและปริมาณข้อมูลที่จะส่งของตัวอุปกรณ์ หากพิจารณาแล้วว่าไม่สามารถทำงานได้ก็จะสั่งให้ฮับปิดการทำงานของพอร์ตนั้น

- 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 นั้นจำเป็นที่จะต้องต่อร่วมกับระบบ (โฮสต์และฮับ) ที่ใช้ระบบบัส USB 2.0 เช่นเดียวกันจึงจะสามารถส่งถ่ายข้อมูลที่ระดับความเร็วสูง

นี้ได้ แต่อย่างไรก็ตามระบบก็ยังสามารถใช้งานร่วมกับอุปกรณ์ที่เป็น USB 1.0/1.1 แต่เดิมได้ โดยส่วนของโฮสต์และฮับ จะทำหน้าที่ในการปรับระดับความเร็วในการส่งถ่ายข้อมูลให้ตรงกัน ซึ่งคุณสมบัตินี้เป็นการเพิ่มความซับซ้อนให้กับวงจรและรูปแบบโปรโตคอลของโฮสต์และฮับ แต่ก็เป็นการทำให้ไม่จำเป็นต้องใช้โฮสต์และฮับที่มีความเร็วแตกต่างกันหลายๆ ตัวในระบบ

สำหรับในส่วนการอินเตอร์เฟส USB ของโครงงานนี้นั้นได้เลือกใช้ IC ที่ทำหน้าที่ในการแปลงรูปแบบการสื่อสารจาก USB ไปยังการสื่อสารในรูปแบบอนุกรม ซึ่งทำให้ไม่ต้องเข้าจัดการกับกระบวนการอินเตอร์เฟสกับ USB โดยตรง โดยจะกล่าวถึงรายละเอียดต่อไป

บทที่ 4
คำสั่งในการควบคุมอุปกรณ์เครื่องมือวัด

ในการควบคุมอุปกรณ์เครื่องมือวัดต่างๆผ่านระบบบัส IEEE-488 นั้น จำเป็นจะต้องทราบถึงคำสั่งใช้งานของอุปกรณ์แต่ละตัว ซึ่งคำสั่งเหล่านี้ก็จะแตกต่างกันไปตามแต่ประเภทของอุปกรณ์ รวมถึงบริษัทผู้ผลิต โดยคำสั่งเหล่านี้จะถูกโปรแกรมไว้ในส่วนของโปรมแกรมควบคุมที่เขียนขึ้นทางฝั่งคอมพิวเตอร์

สำหรับอุปกรณ์เครื่องมือวัดที่ได้เลือกมาทดสอบในปริญญานิพนธ์นี้ประกอบด้วย

- HAMEG Programmable Power Supply รุ่น HM8142
- HAMEG Programmable Function Generator รุ่น HM8130
- HAMEG Programmable Multimeter รุ่น HM8112-2
- HAMEG Storage Analog/Digital Oscilloscope รุ่น HM 1007

โดยมีรูปแบบคำสั่งต่างๆ ดังต่อไปนี้

4.1 HAMEG Programmable Power Supply รุ่น HM8142

เป็นแหล่งจ่ายพลังงานไฟฟ้ากระแสตรงขนาด 1A 2 Channel สามารถควบคุมผ่านระบบบัส IEEE-488 โดยมีคำสั่งดังสั่งการดังต่อไปนี้

คำสั่ง : RM1 / RM0

รูปแบบ : RM1

ฟังก์ชั่น : สั่งให้ Power Supply ทำงานในโหมดรีโมท โดยไม่สามารถควบคุมผ่านหน้าปัทม์ได้ แต่สามารถควบคุมการทำงานได้โดยผ่านระบบบัสแบบ IEEE-488 เท่านั้น และสามารถออกจากโหมดรีโมทได้โดยคำสั่ง RM0 หรือการกดปุ่ม LOCAL ที่หน้าปัทม์ของเครื่อง

รูปแบบ :RM0

ฟังก์ชั่น : ทำการยกเลิกโหมดรีโมท ให้เครื่องกลับสู่สถานะ LOCAL (สามารถควบคุมการทำงานของเครื่องได้โดยผ่านปุ่มหน้าปัทม์ 1

ข้อสังเกต : คำสั่ง RM 0 จะมีผมต่อคำสั่ง LK 1

คำสั่ง : MX1 / MX0

รูปแบบ : MX1

ฟังก์ชั่น : เปลี่ยนโหมดรีโมทไปเป็นโหมดแบบผสม ในโหมดนี้สามารถที่จะควบคุมการทำงานของเครื่องได้ทั้งโดยการส่งคำสั่งผ่านระบบแบบ IEEE-488 และการควบคุมผ่านทางหน้าปัทม์ของเครื่อง

รูปแบบ : MX0

ฟังก์ชั่น : ยกเลิกโหมดแบบผสมและกลับเข้าสู่สถานะปกติที่ควบคุม โดยผ่านระบบบัสแบบ IEEE-488

คำสั่ง: LK1 / LK0

รูปแบบ : LK1

ฟังก์ชั่น : สั่งให้เครื่องเข้าสู่โหมดรีโมท โดยไม่สามารถกดปุ่ม LOCAL เพื่อที่จะให้เครื่องกลับสู่โหมด LOCAL ได้

รูปแบบ : LK0

ฟังก์ชั่น : สั่งให้เครื่องออกจากโหมดรีโมทที่ไม่สามารถกดปุ่ม LOCAL ทำให้สามารถกดปุ่ม LOCAL เพื่อที่จะสามารถควบคุมการทำงานผ่านหน้าปัทม์ของเครื่องได้

ข้อสังเกต : โหมดรีโมทที่ไม่สามารถกดปุ่ม LOCAL นี้สามารถถูกยกเลิกได้โดยคำสั่ง RM0

คำสั่ง : SU1 และ SU2

รูปแบบ : SU1 VV.mV.mV หรือ SU2 01.34

ฟังก์ชัน : ตั้งค่าศักดาไฟฟ้าของแหล่งจ่ายที่ 1 หรือ 2

คำสั่ง : SI1 และ SI2

รูปแบบ : SI1 A.mAmAmA หรือ SI2 0.123

ฟังก์ชัน : ตั้งค่ากระแสไฟฟ้าของแหล่งจ่ายที่ 1 หรือ 2 เพื่อจำกัดค่ากระแสไฟฟ้า

คำสั่ง : RU1 และ RU2

รูปแบบ : RU1 หรือ RU2

ฟังก์ชัน : ค่าศักดาไฟฟ้าที่ส่งกลับโดยเครื่อง จะเป็นค่าศักดาไฟฟ้าที่ถูกกำหนดค่าโดยคำสั่งกำหนดค่า (SET)

ข้อสังเกต : ใช้คำสั่ง MUX เพื่อที่จะถามค่ากระแสไฟฟ้าจริงที่วัดได้จากเอาท์พุท

คำสั่ง : RI1 และ RI2

รูปแบบ : RI1 หรือ RI2

ค่าที่ส่งกลับ : I1_1.000A หรือ I2_0.012A

ฟังก์ชัน : ค่ากระแสไฟฟ้าที่ส่งกลับโดยเครื่อง จะเป็นค่ากระแสไฟฟ้าที่ถูกกำหนดค่าโดยคำสั่งจำกัดค่ากระแสไฟฟ้า (SI1 หรือ SI2)

ข้อสังเกต : ใช้คำสั่ง MIX เพื่อถามค่ากระแสไฟฟ้าจริงที่วัดได้จากเอาท์พุทจริง

คำสั่ง : MU1 และ MU2

รูปแบบ : MU1 หรือ MU2

ค่าที่ส่งกลับ : U1:12.34V หรือ U2:12.24V

ฟังก์ชัน : ค่าศักดาไฟฟ้าที่ส่งกลับโดยเครื่อง ซึ่งเป็นค่าที่วัดได้จากเอาท์พุทจริง

ข้อสังเกต : คำสั่ง RUX ใช้เพื่อถามค่าศักดาไฟฟ้าที่ทำการกำหนด

คำสั่ง : MI1 และ MI2

รูปแบบ : MI1 หรือ MI2

ค่าที่ส่งกลับ : I1= +1.000A หรือ I2= -0.123A

ฟังก์ชัน : ค่ากระแสไฟฟ้าที่ส่งกลับโดยเครื่อง เป็นค่ากระแสไฟฟ้าที่วัดได้

ข้อสังเกต : คำสั่ง RIX ใช้เพื่อถามถึงค่ากระแสไฟฟ้าที่จำกัดเอาไว้ ถ้าเอาท์พุทถูกปิดไว้ค่าที่ส่งกลับจะเป็น I1=_1.000A

คำสั่ง : TRU

รูปแบบ : TRU

ฟังก์ชัน : กำหนดค่าศักดาไฟฟ้าของแหล่งจ่ายตัวที่ 1 และ 2 ให้มีค่าเท่ากันตามต้องการ

คำสั่ง : TRI

รูปแบบ : TRI

ฟังก์ชัน : กำหนดค่ากระแสไฟฟ้าของแหล่งจ่ายตัวที่ 1 และ 2 ให้มีค่าเท่ากันตามต้องการ

คำสั่ง : SR1 และ SR0

รูปแบบ : SR1

ฟังก์ชัน : ทำการ enable โหมด service request โดยไม่มีการเปลี่ยนแปลงใดๆ ในสถานะของอุปกรณ์ที่จะมีผลต่อสัญญาณ SRQ (service request)

รูปแบบ : SR0

ฟังก์ชัน : ทำการยกเลิกโหมด Service request

คำสั่ง : STA

รูปแบบ : STA

ค่าที่ส่งกลับ : OP1/0SR1/0ER0/1CV1/CC1/CV2/CC2/RM0/1

OP0 สวิทช์เอาท์พุทปิด

OP1 สวิทช์เอาท์พุทเปิด

SQ1 แสดงสถานะของอุปกรณ์ที่เปลี่ยนแปลง (CV ไปเป็น CC หรือ OP1 ไป เป็น OP0 เป็นต้น) (จะทำงานเฉพาะเมื่อ SRQ นั้น enable หรือมีค่าเป็น 1 , ดูคำสั่ง SR1)

SQ2 เมื่อสัญญาณ service request (SRQ) ถูกทำให้ enable แสดงว่าไม่มีการเปลี่ยนแปลงสถานะของเครื่องมือ

ER0 ไม่มีความผิดพลาด

ER1 ความร้อนสูงเกิน

CV1 แหล่งจ่ายตัวที่ 1 ทำการจ่ายศักดาไฟฟ้าคงที่

CC1 แหล่งจ่ายตัวที่ 1 ทำการจ่ายกระแสไฟฟ้าคงที่

CV2 แหล่งจ่ายตัวที่ 2 ทำการจ่ายศักดาไฟฟ้าคงที่

CC2 แหล่งจ่ายตัวที่ 2 ทำการจ่ายกระแสไฟฟ้าคงที่

RM1 อุปกรณ์อยู่ในโหมดรีโหมด

RM0 อุปกรณ์ไม่อยู่ในโหมดรีโหมด

ฟังก์ชัน : สั่งให้เครื่องส่งค่าสตริงที่เก็บสถานะปัจจุบันของเครื่องมาให้

คำสั่ง: OP1 และ OP0

รูปแบบ : OP1

ฟังก์ชัน : สวิทช์เอาท์พุทเปิด

รูปแบบ : OP0

ฟังก์ชัน : สวิทช์เอาท์พุทปิด

คำสั่ง : CLR (clear)

รูปแบบ : CLR

ฟังก์ชัน : ยกเลิกฟังก์ชั่นทั้งหมดของเครื่อง และทำการเริ่มต้นใหม่ที่สถานะศูนย์ (zero state) ในโหมดรีโมท คีย์บอร์ดไม่สามารถใช้งานได้และสวิทช์เอาท์พุทปิด รวมทั้งศักดาไฟฟ้า และกระแสไฟฟ้าจะถูกตั้งค่าให้เป็นศูนย์

คำสั่ง : VER

รูปแบบ : VER

ค่าที่ส่งกลับ : sw Vx.xhw Vx.xxxxxxx HAMEG/Paris KRP&VM

ฟังก์ชัน : แสดงเวอร์ชันของซอฟต์แวร์และฮาร์ดแวร์ของเครื่อง

คำสั่ง : ID?

รูปแบบ : ID?

ค่าที่ส่งกลับ : HM8142-1

ฟังก์ชัน : แสดงถึงชื่อรุ่นของเครื่อง

4.2 HAMEG Programmable Function Generator รุ่น HM8130

ทำหน้าที่สร้างสัญญาณรูปแบบต่างๆ ป้อนให้แก่วงจรที่ทำการทดลอง และมีคำสั่งควบคุมฟังก์ชันในการทำงานผ่านระบบบัสแบบ IEEE-488 โดยสามารถแบ่งได้เป็นกลุ่มคำสั่งดังต่อไปนี้

หมวดคำสั่งที่ไม่ต้องมีข้อมูลประกอบ

คำสั่ง: SIN

รูปแบบ: SIN

ฟังก์ชัน : ฟังก์ชันการสร้างสัญญาณคลื่นรูปซายน์

คำสั่ง: TRI

รูปแบบ: TRI

ฟังก์ชัน: ฟังก์ชันการสร้างสัญญาณคลื่นรูปสามเหลี่ยม

คำสั่ง: SQR

รูปแบบ: SQR

ฟังก์ชัน : ฟังก์ชันการสร้างสัญญาณคลื่นรูปสี่เหลี่ยม

คำสั่ง: PLS

รูปแบบ: PLS

ฟังก์ชัน: ฟังก์ชันการสร้างสัญญาณคลื่นเป็นพัลซ์

คำสั่ง: RMS

รูปแบบ: RMS

ฟังก์ชัน : ฟังก์ชันการสร้างสัญญาณคลื่นรูปฟันเลื่อย โดยมีสัญญาณขาขึ้นเป็นบวก

คำสั่ง: RMN

รูปแบบ: RMN

ฟังก์ชัน : ฟังก์ชันการสร้างสัญญาณคลื่นรูปฟันเลื่อย โดยมีสัญญาณขาขึ้นเป็นลง

คำสั่ง: ARB

รูปแบบ: ARB

ฟังก์ชัน : ฟังก์ชันการสร้างสัญญาณคลื่นรูปแบบ arbitary

คำสั่ง: SW1/0

รูปแบบ: SW1/0

ฟังก์ชัน : การเลือกการกวาดสัญญาณเปิด/ปิด

คำสั่ง: CTM

รูปแบบ: CTM

ฟังก์ชัน : การเลือกโหมดการสร้างสัญญาณแบบต่อเนื่อง

คำสั่ง: GTM

รูปแบบ: GTM

ฟังก์ชัน : การเลือกโหมดการสร้างสัญญาณแบบGATED

คำสั่ง: TRM

รูปแบบ: TRM

ฟังก์ชัน : การเลือกโหมดการสร้างสัญญาณแบบTRIGGER

คำสั่ง: OT1/0

รูปแบบ: OT1/0

ฟังก์ชัน : การเลือกให้เอาท์พุทเปิด/ปิด

คำสั่ง: OF1/0

รูปแบบ: OF1/0

ฟังก์ชัน : การเลือกให้ออฟเซ็ทเปิด/ปิด

คำสั่ง: DFR

รูปแบบ: DFR

ฟังก์ชัน : ให้แสดงความถี่ของสัญญาณ

คำสั่ง: DST

รูปแบบ: DST

ฟังก์ชัน : ให้แสดงความถี่เริ่มต้นของสัญญาณ

คำสั่ง: DSP

รูปแบบ: DSP

ฟังก์ชัน : ให้แสดงความถี่สุดท้ายของสัญญาณ

คำสั่ง: DWT

รูปแบบ: DWT

ฟังก์ชัน : ให้แสดงความกว้างของสัญญาณพัลซ์

คำสั่ง: DSW

รูปแบบ: DSW

ฟังก์ชัน : ให้แสดงเวลาในการกวาดสัญญาณ

คำสั่ง: DAM

รูปแบบ: DAM

ฟังก์ชัน : ให้แสดงค่าแอมปลิจูดของเอาท์พุท

คำสั่ง: DOF

รูปแบบ: DOF

ฟังก์ชัน : ให้แสดงค่าออฟเซ็ท

คำสั่ง: RM0

รูปแบบ: RM0

ฟังก์ชัน : ยกเลิกโหมดรีโมท แล้วกลับเข้าสู่การควบคุมการทำงานผ่านหน้าปัทม์ของเครื่อง เงื่อนไขนี้สามารถทำได้โดยการกดปุ่ม LOCAL เช่นเดียวกัน

ข้อสังเกต: คำสั่งRM0จะยกเลิกคำสั่ง LK1

คำสั่ง: LK1

รูปแบบ: LK1

ฟังก์ชัน : ยกเลิก LOCAL ชั่วคราว โดยที่สถานะนี้การควบคุมการทำงานของเครื่องทำได้โดยผ่านทางระบบบัสเท่านั้น และการกลับสู่สถานะ LOCAL ไม่สามารถทำได้โดยการปุ่ม LOCAL

คำสั่ง: LK0

รูปแบบ: LK0

ฟังก์ชัน : ยกเลิกสถานการณ์ยกเลิกปุ่ม LOCAL ชั่วคราว ทำให้สามารถกดปุ่ม LOCAL เพื่อที่จะกลับสู่สถานะ LOCAL ได้ ทำให้สามารถควบคุมการทำงานของเครื่องผ่านปุ่มหน้าปัทม์ได้อีกครั้ง

คำสั่ง: TRG

รูปแบบ: TRG

ฟังก์ชัน : TRIG คาบสัญญาณ (ปิดการกวาดสัญญาณ) หรือการกวาดสัญญาณที่สมบูรณ์ (การกวาดสัญญาณเปิด)

คำสั่ง: CLR

รูปแบบ: CLR

ฟังก์ชัน : รีเซ็ทเครื่องและกลับสู่สถานะเริ่มต้น คำสั่ง CLR เหมือนกับคำสั่ง SDC ตามมาตรฐานของ IEEE-488

คำสั่ง: ARC

รูปแบบ: ARC

ฟังก์ชัน : ลบข้อมูลที่เป็นการสร้างสัญญาณแบบใดๆ และรีเซ็ทตัวนับภายในให้มีค่าเป็นศูนย์

คำสั่ง: ARE

รูปแบบ: ARE

ฟังก์ชัน : ยกเลิกการเขียนข้อมูลที่ใช้สร้างแบบ arbitary

หมวดคำสั่งที่ประกอบด้วยข้อมูลที่เป็นเลขทศนิยม

คำสั่งทั้งหมดที่ประกอบด้วยข้อมูลแบบทศนิยมจะประกอบด้วยตัวอักษร 3 ตัวแล้วตามด้วย ":" ความยาวของข้อมูลที่มากที่สุดจะเป็นเลขจำนวน 5 หลักรวมกับจุดทศนิยมด้วย โดยรูปแบบของคำสั่ง นั้นจะมีค่าเลขยกกำลังหรือไม่มีก็ได้ จะมีจุดทศนิยมหรือไม่มีก็ได้ แต่จะมีหน่วยเป็นโวลท์,เฮริทซ์ และวินาที แต่จะ ไม่มีการส่งค่าหน่อยลงไปบนบัส ข้อมูลสามารถมีเครื่องหมายนำหน้าที่ได้ถ้าจำเป็น แต่สัญญาณที่เป็นบวกไม่ต้องมีเครื่องหมายนำหน้าก็ได้ ระหว่างเครื่องหมายกับค่าที่กำหนดจะต้องติดกันห้ามเว้นวรรค

ตัวอย่าง : FRQ:1000.0 มีค่าเท่ากับ FRQ:1000.0

FRQ:1E3 มีค่าเท่ากับ FRQ:1E+3

FRQ:1.0000E+3 มีค่าเท่ากับ FRQ:1.0000E+2

FRQ:0.0001E7 มีค่าเท่ากับ FRQ:0.0001E-1

คำสั่ง:FRQ

รูปแบบ:FRQ:<ข้อมูล>

ฟังก์ชัน: กำหนดค่าความถี่ที่ต้องการ < >Hz

คำสั่ง:STT

รูปแบบ:STT:<ข้อมูล>

ฟังก์ชัน: กำหนดค่าความถี่เริ่มต้น < >Hz

คำสั่ง:STP

รูปแบบ:STP:<ข้อมูล>

ฟังก์ชัน: กำหนดค่าความถี่สุดท้าย < >Hz

คำสั่ง:SWT

รูปแบบ:SWT:<ข้อมูล>

ฟังก์ชัน: กำหนดค่าเวลาในการกวาดสัญญาณ <> S

คำสั่ง:WDT

รูปแบบ:WDT:<ข้อมูล>

ฟังก์ชัน: กำหนดค่าความกว้างของสัญญาณพัลซ์ <> S

คำสั่ง:AMP

รูปแบบ:AMP:<ข้อมูล>

ฟังก์ชัน: กำหนดค่าแอมปลิจูดของสัญญาณ <> V

คำสั่ง:OFS

รูปแบบ:OFS:<ข้อมูล>

ฟังก์ชัน: กำหนดค่าออฟเซ็ทของสัญญาณ <> V

หมายเหตุ : ค่าความถี่และเวลากำหนดได้มากที่สุด 5 หลัก และค่าศักดาไฟฟ้ากำหนดได้มากที่สุด 3 หลัก

ค่าแอมปลิจูดสามารถกำหนดได้ 2 วิธี ถ้าเป็นค่าที่ไม่มีเครื่องหมายจะถือว่าเป็นค่าศักดาไฟฟ้าแบบ peak to peak (ในกรณีของสัญญาณรูปพัลซ์นั้น ศักดาไฟฟ้าสูงสุดจะมีค่าเท่ากับครึ่งหนึ่งของค่านี้) ถ้าเป็นคู่ที่มีเครื่องหมายนำหน้าจะถือว่าเป็นค่าศักดาไฟฟ้าสูงสุด

หมวดคำสั่งร้องขอ

คำสั่งเหล่านี้จะทำการเก็บค่าสตริงของข้อมูลที่สามารถอ่านออกมาได้ทันทีที่เครื่องถูกกำหนดให้เป็นตัวส่ง โดยมีคำสั่งต่างๆ และรูปแบบ คือ

คำสั่ง: FRQ?

รูปแบบ: FRQ?

ฟังก์ชัน: แสดงความถี่ของสัญญาณ

คำสั่ง:STT?

รูปแบบ:STT?

ฟังก์ชัน: แสดงความถี่เริ่มต้นของสัญญาณ

คำสั่ง:STP?

รูปแบบ:STP?

ฟังก์ชัน: แสดงความถี่สุดท้ายของสัญญาณ

คำสั่ง:SWT?

รูปแบบ:SWT?

ฟังก์ชัน: แสดงเวลาในการกวาดสัญญาณ

คำสั่ง:WDT?

รูปแบบ:WDT?

ฟังก์ชัน: แสดงความกว้างของสัญญาณพัลซ์

คำสั่ง:AMP?

รูปแบบ:AMP?

ฟังก์ชัน: แสดงค่าแอมปลิจูดของศักดาไฟฟ้าเอาท์พุท

คำสั่ง:OFS?

รูปแบบ:OFS?

ฟังก์ชัน: แสดงค่าออฟเซ็ท

คำสั่ง:ARD?

รูปแบบ:ARD?

ฟังก์ชัน: แสดงข้อมูลของสัญญาณแบบ arbitary

คำสั่ง:ID?

รูปแบบ:ID?

ฟังก์ชัน: แสดงหมายเลยรุ่นของเครื่อง

คำสั่ง:VER?

รูปแบบ:VER?

ฟังก์ชัน: แสดงเวอร์ชันของซอฟต์แวร์และฮาร์ดแวร์ของเครื่อง

คำสั่ง:STA?

รูปแบบ:STA?

ฟังก์ชัน: แสดงสถานะของเครื่อง โดยสตริงข้อมูลที่อ่านได้จะมีทั้งหมด 21 ตัวอักษร ซึ่งจะบอกถึงสถานะของเครื่อง โดยมีความหมาย คือ

OT0OF0SW0SINCTMDFRDAM

1.OT0 สวิทช์เอาท์พุทเปิด (ปิด)

2. OF0 การเซ็ทค่าออฟเซ็ทปิด (เปิด)

3. SW0 การกวาดสัญญาณปิด (เปิด)

4. SIN รูปแบบของสัญญาณ (ซายน์)

5. CTM โหมดของการทำงาน (โหมดการสร้างสัญญาณแบบต่อเนื่อง)

6.DFR แสดงรายละเอียด (ความถี่)

7.DAM แสดงรายละเอียด (แอมปลิจูด)

ค่าเอาท์พุทจะอยู่ในรูปแบบเลขทศนิยมกับเลขยกกำลัง สตริงข้อมูลแต่ละค่าจะขึ้นต้นด้วยคำสั่งของตัวมันเอง เช่น

"FRQ:1.2345E+3"

"OFS:-3.0E+0"

"WDT:45.6E-6"

สำหรับการส่งคำสั่งควบคุมเครื่องสามารถที่จะส่งคำสั่งไปควบคุมพร้อมกันหลายฟังก์ชันโดยสามารถส่งคำสั่งเหล่านั้นไปพร้อมกันเป็นข้อมูลชุดเดียวกันได้ เช่น

"FRQ:12.34E+3 TRI OT1 AMP:10"

Frequency 12.3 K Hz; Triangle; Output on; Voltage amplitude 10V

4.3 HAMEG Programmable Multimeter รุ่น HM8112-2

ใช้สำหรับวัดค่าพารามิเตอร์ต่างๆ ทางไฟฟ้า โดยมีคำสั่งควบคุมฟังก์ชันการทำงานผ่านระบบบัสแบบ IEEE-488 ดังนี้

คำสั่ง:VD

รูปแบบ:VD

ฟังก์ชัน: การวัดศักดาไฟฟ้ากระแสตรง

คำสั่ง:VA

รูปแบบ:VA

ฟังก์ชัน: การวัดศักดาไฟฟ้ากระแสสลับ

คำสั่ง:O2

รูปแบบ:O2

ฟังก์ชัน: การวัดค่าความต้านทาน

คำสั่ง:ID

รูปแบบ:ID

ฟังก์ชัน: การวัดไฟฟ้ากระแสตรง

คำสั่ง:IA

รูปแบบ:IA

ฟังก์ชัน: การวัดไฟฟ้ากระแสสลับ

คำสั่ง:TC

รูปแบบ:TC

ฟังก์ชัน: การวัดอุณหภูมิแบบเซลเซียส

คำสั่ง:TK

รูปแบบ:TK

ฟังก์ชัน: การวัดอุณหภูมิแบบเคลวิน

คำสั่ง:TF

รูปแบบ:TF

ฟังก์ชัน: การวัดอุณหภูมิแบบฟาเรนไฮต์

คำสั่ง:RX

รูปแบบ:RX

ฟังก์ชัน: เลือกช่วงการวัดช่วงต่างๆ โดย "X" คือช่วงการวัดที่ต้องการ คือ

R1ช่วงเวลา 0.2Vdc, Vac, KΩ

R2ช่วงเวลา	2Vdc,	Vac,	KΩ	2 mAdc	2mAac
R3ช่วงเวลา	20Vdc,	Vac,	KΩ	----	----
R4ช่วงเวลา	200Vdc,	Vac,	KΩ	----	----
R5ช่วงเวลา	1000Vdc,	Vac,	2000KΩ	2000 mAdc	2000 mAac
R6ช่วงเวลา	10000Vdc,	Vac,	12000KΩ	----	----

คำสั่ง:A0/A1

รูปแบบ:A0/A1

ฟังก์ชัน: ไม่เลือกฟังก์ชันการปรับช่วงการวัดอัตโนมัติ, เลือกฟังก์ชันการปรับช่วงการวัดอัตโนมัต

คำสั่ง:TX

รูปแบบ:TX

ฟังก์ชัน:เลือกช่วงเวลาและจำนวนหลักในการแสดงผลที่ได้จากการวัด

T1 ช่วงเวลา	100ms	การแสดงผล	5 ½ หลัก
T2 ช่วงเวลา	1s	การแสดงผล	5 ½ หลัก
T3 ช่วงเวลา	1s	การแสดงผล	6 ½ หลัก
T4 ช่วงเวลา	10s	การแสดงผล	5 ½ หลัก

คำสั่ง:ZO

รูปแบบ:ZO

ฟังก์ชัน:เลือกการปรับค่าออฟเซ็ท

คำสั่ง:S0

รูปแบบ:S0

ฟังก์ชัน:เริ่มการวัดแบบต่อเนื่อง

คำสั่ง:S1

รูปแบบ:S1

ฟังก์ชัน:หยุดการวัดแบบต่อเนื่อง

คำสั่ง:MO

รูปแบบ:MO

ฟังก์ชัน:เลือกช่องมัลติเพลกเซอร์

คำสั่ง:MO-M9

รูปแบบ:MO-M9

ฟังก์ชัน:เลือกช่อง 0-9

คำสั่ง:L0

รูปแบบ:L0

ฟังก์ชัน:ส่งข้อมูลเฉพาะข้อมูลบล็อกแรก

คำสั่ง:L1

รูปแบบ:L1

ฟังก์ชัน:ส่งข้อมูลทั้งสองบล็อกข้อมูล (ข้อมูลที่วัดและตัวอักษรจะอยู่ในข้อมูลบล๊อกแรก ส่วนข้อมูลที่เป็นการ โปรแกรมจะอยู่ในข้อมูลบล๊อกที่ 2

รูปแบบข้อมูลที่ส่งนั้น จะประกอบด้วยข้อมูล 2 บล็อกข้อมูล โดยข้อมูลบล็อกแรกจะประกอบด้วยตัวอักษร 12 ตัว และข้อมูลบล็อกที่ 2 จะประกอบด้วย ตัวอักษร 20 ตัวอักษรบวกกับตัวอักษรที่ปิดท้ายข้อมูล โดยรูปแบบข้อมูลจะมีลักษณะดังนี้ คือ

+X.XXXXXXXE+XVDR1A0T1S0Q0M0X0P0B0

โดยข้อมูลบล็อกแรกจะเป็นข้อมูลที่ได้จากการวัด (+X.XXXXXXX+X) และข้อมูลบล็อกที่ 2 จะเป็นฟังก์ชันในการวัดของเครื่องหรือสถานะของเครื่องนั้นเอง

คำสั่ง:Q0

รูปแบบ:Q0

ฟังก์ชัน: ไม่ส่งสัญญาณ SRQ (scrvice request)

คำสั่ง:Q1

รูปแบบ:Q1

ฟังก์ชัน: ส่งสัญญาณ SRQ (scrvice request)

คำสั่ง:NVXXXXXXXX

รูปแบบ: NVXXXXXXXX

ฟังก์ชัน: ทำการปรับแต่งเครื่องผ่านระบบบัสแบบ IEEE-488 โดยหลังคำสั่ง NV นั้น เครื่องจะถือว่าเลขจำนวนเต็ม 6 หลักเป็นค่าที่ใช้สำหรับการปรับแต่ง

คำสั่ง:P1

รูปแบบ:P1

ฟังก์ชัน: แสดงค่าออฟเซ็ท คำนวนจาก R=X-C

คำสั่ง:P2

รูปแบบ:P2

ฟังก์ชัน: แสดงเปอร์เซ้นต์ความเบี่ยงเบน คำนวนจาก R = 100(X-C)/C

คำสั่ง:P3

รูปแบบ:P3

ฟังก์ชัน: แสดง dB คำนวนจาก R = 20 LOG(X/C)

คำสั่ง:P4

รูปแบบ:P4

ฟังก์ชัน: แสดง dBm คำนวนจาก R=20 LOG(X-C)

เมื่อ C=0.775V สำหรับค่าศักดาไฟฟ้า และ C=1.29 mA สำหรับค่ากระแสไฟฟ้า

คำสั่ง:PXEN

รูปแบบ:PXEN

ฟังก์ชัน:เลือกค่าการวัดที่ X เป็นค่าคงที่ ถ้าP2-4 ; X มีค่าเท่ากับ 1,2,3 ตามลำดับ

คำสั่ง:ID?

รูปแบบ:ID?

ฟังก์ชัน:ทำการส่งหมายเลขรุ่นของเครื่องลงบนหน่อยความจำของเครื่อง

คำสั่ง:STA?

รูปแบบ:STA?

ฟังก์ชัน:ทำการส่งสถานะของเครื่อง (ข้อมูลบล็อกที่2)

คำสั่ง:D0/D1

รูปแบบ:D0/D1

ฟังก์ชัน:ปิด/เปิด หน้าจอ

คำสั่ง:BX

รูปแบบ:BX

ฟังก์ชัน:ส่งค่าของการสวิทซ์ที่เลือกครั้งสุดท้าย โดยที่ x=1,…,9,A,…,F

คำสั่ง:EOI

รูปแบบ:EOI

ฟังก์ชัน:เลือกให้มีการส่งสัญญาณ EOI

คำสั่ง:EOS1/EOI2

รูปแบบ: EOS1/EOI2

ฟังก์ชัน:เลือกให้ข้อมูลที่ส่งมีการส่งค่า EOS1 หรือ EOS2 ปิดท้ายข้อมูล

คำสั่ง:END

รูปแบบ:END

ฟังก์ชัน:ยกเลิกตัวอักษรที่ปิดท้ายสตริงข้อมูล

4.4 HAMEG Storage Analog/Digital Oscilloscope รุ่น HM 1007

เป็น Osilloscope ที่สามารถวัดสัญญาณที่มีค่าความถี่สูงสุด 100 เมกกะเฮิร์ทซ์ โดย Oscilloscope นี้สามารถที่จะทำการบันทึกสัญญาณที่ได้จากการวัดมาเก็บไว้ในหน่วยความจำของเครื่อง และสามารถที่จะอ่านค่าของสัญญาณที่บันทึกเก็บไว้ผ่านระบบบัสแบบ IEEE 488 ได้ แต่ไม่สามารถที่จะควบคุมช่วงของเวลาต่อช่องและจำนวนช่องต่อโวล์ทได้ สำหรับคำสั่งใช้ควบคุมการทำงานผ่านระบบบัส IEEE 488 มีดังนี้ คือ

คำสั่ง :ID?

รูปแบบ:ID?

ฟังก์ชัน:ทำการส่งหมายเลขรุ่นของเครื่องลงบนหน่วยความจำของเครื่อง

คำสั่ง:DIG

รูปแบบ:DIG[channel code]

ฟังก์ชัน: 1.ส่งข้อมูลเก็บไว้ในหน่วยความจำของเครื่องเฉพาะช่องการวัดที่ 1

2.ส่งข้อมูลเก็บไว้ในหน่วยความจำของเครื่องเฉพาะช่องการวัดที่ 2

3.ส่งข้อมูลเก็บไว้ในหน่วยความจำของเครื่องเฉพาะช่องการวัด

คำสั่ง:GET [Channel code]

รูปแบบ:GET

ฟังก์ชัน: 1.ส่งสัญญาณ TRIG โดยไม่กำหนดช่องการวัด

2.ส่งสัญญาณ TRIG โดยกำหนดช่องการวัด

คำสั่ง:STA

รูปแบบ:STA

ฟังก์ชัน: ตรวจสอบสถานะเซ็ทช่องการวัดของ Oscilloscope

คำสั่ง:TXT

รูปแบบ:TXT <สตริง@>

ฟังก์ชัน: ส่งสตริงไปยังการอินเตอร์เฟส

คำสั่ง:OFS

รูปแบบ:OFS

ฟังก์ชัน: ตรวจสอบการเลื่อนตำแหน่งของ Y จากหน่อยความจำอ้างอิง หลังจากสัญญาณอ้างอิงถูกเก็บไว้ในหน่อยความจำ

คำสั่ง:FRM

รูปแบบ:FRM [format]

ฟังก์ชัน: กำหนดรูปแบบข้อมูลที่ทำการส่ง (0 ข้อมูลเป็บแบบไบนารี)

คำสั่ง:V24

รูปแบบ:V24

ฟังก์ชัน: ส่งข้อมูลผ่านระบบบัสแบบ IEEE 488 ไปยังอุปกรณ์ภายนอกที่ต่อผ่านระบบบัสแบบ RS232C

คำสั่ง:PRN

รูปแบบ:PRN

ฟังก์ชัน: เริ่มส่งข้อมูลผ่านพอร์ตใดพอร์ตหนึ่งในจำนวน 3 พอร์ตของเครื่อง

คำสั่ง:XYZ

รูปแบบ:XYZ

ฟังก์ชัน: เปลี่ยนโหมดการอ่านค่าสัญญาณเป็นการอ่านค่าสัญญาณในโหมด XY

สิ่งสำคัญสิ่งหนึ่งซึ่งขาดไม่ได้คือในการจบชุดคำสั่งของแต่ละข้อมูลจะต้องตามด้วยรหัส ASCII อักษรตัวที่ 13 (ในทางการเขียนโปรแกรม) หรือการกด Enter นั่นเอง ในส่วนของการส่งค่าของชุดข้อมูลคำสั่งเหล่านี้นั้น จำเป็นที่จะต้องอาศัยสัญญาณควบคุมการรับส่งข้อมูล (handshake lines) และสัญญาณ EOI ในการจัดเรียงชุดข้อมูลด้วย เนื่องจากข้อมูลนี้นั้นถูกส่งออกไปครั้งละตัวอักษรดังนั้นจึงต้องอาศัยสัญญาณ EOI เพื่อจัดเรียงข้อมูลแต่ละตัวอักษรนั้นเป็นชุดคำสั่งนั่นเอง

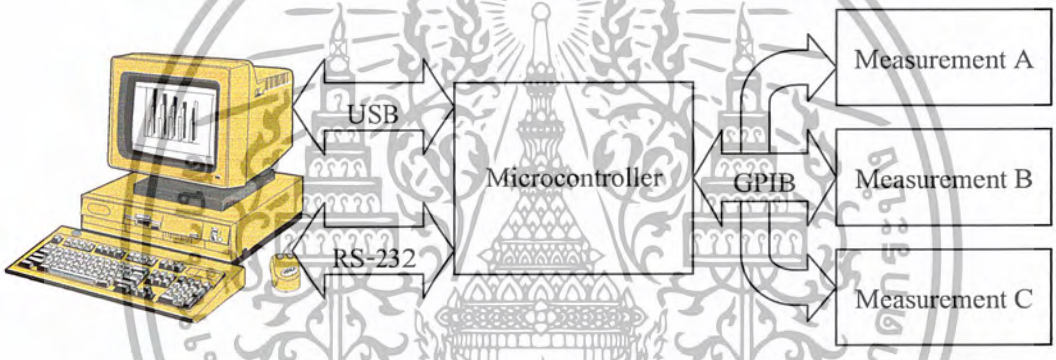
นอกเหนือจากคำสั่งของอุปกรณ์เหล่านี้ หากเป็นการใช้งานร่วมกับอุปกรณ์อื่นๆ ก็สามารถทำได้โดยการเปลี่ยนแปลงคำสั่งของโปรแกรมที่เขียนขึ้นควบคุมให้เหมาะสมกับอุปกรณ์เครื่องมือวัดเหล่านั้นนั่นเอง

บทที่ 5
การออกแบบส่วนการอินเตอร์เฟสของระบบ

5.1 องค์ประกอบของระบบ

ภายในระบบควบคุมอุปกรณ์เครื่องมือวัดนั้น ประกอบไปด้วยส่วนประกอบหลัก 3 ส่วน ได้แก่

- โปรแกรมสั่งการทางคอมพิวเตอร์
- ส่วนการอินเตอร์เฟสของระบบ
- ส่วนของอุปกรณ์เครื่องมือวัดที่ต้องการควบคุม



รูปที่ 5.1 แสดงองค์ประกอบโดยรวมของระบบ

5.2 หน้าที่ของส่วนการอินเตอร์เฟสของระบบ

ในส่วนนี้ถือได้ว่าเป็นหัวใจของระบบ ซึ่งทำหน้าที่เสมือนตัวแปลภาษาระหว่างระบบบัส GPIB ไปยัง USB และ RS-232 ซึ่งหน้าที่หลักของส่วนการอินเตอร์เฟสนี้คือจัดเรียงข้อมูลให้ถูกต้องตามมาตรฐานของแต่ละพอร์ต โดยในที่นี้นั้นได้เลือกใช้ไมโครคอนโทรลเลอร์ PIC16F877 มาทำหน้าที่ในจัดเรียงข้อมูลเพื่ออินเตอร์เฟสไปยังระบบบัส GPIB โดยการส่งข้อมูลระหว่างระบบบัส GPIB และ ไมโครคอนโทรลเลอร์นั้นจำเป็นจะต้องกระทำให้ถูกต้องตามมาตรฐานของ GPIB สำหรับในส่วนของข้อมูลที่จะติดต่อไปยังคอมพิวเตอร์นั้น ข้อมูลจะถูกสร้างขึ้นใหม่โดยไมโครคอนโทรลเลอร์ และส่งออกไปในรูปแบบข้อมูลอนุกรม ซึ่งข้อมูลที่ได้มานี้จะถูกนำไปเปลี่ยนแปลงรูปแบบในการติดต่อกับคอมพิวเตอร์ตามที่ต้องการ

5.3 ขบวนการติดต่อไปยังอุปกรณ์เครื่องมือวัด

ในการเชื่อมต่อระบบไปยังอุปกรณ์เครื่องมือวัดต่างๆนั้น เนื่องจากอุปกรณ์เครื่องมือวัดที่ต่อร่วมอยู่ภายในระบบนั้นล้วนแล้วแต่ใช้สายส่งข้อมูล (Data) ชุดเดียวกัน ดังนั้นในช่วงเวลาหนึ่งจะมีอุปกรณ์ที่ได้รับอนุญาตให้รับหรือส่งข้อมูลเพียงอุปกรณ์เดียวเท่านั้น ดังนั้นส่วนที่จะทำหน้าที่ในการระบุอุปกรณ์ที่ได้รับอนุญาตก็คือ Address นั่นเอง ซึ่งในการระบุ Addresss นั้นจะเป็นจะต้องกระทำก่อนการรับหรือส่งข้อมูลไปยังอุปกรณ์ใดๆ 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 นั้นข้อมูลที่อยู่บนบัสข้อมูลจะหมายถึงชุดข้อมูลคำสั่ง ซึ่งชุดข้อมูลนี้นั้นจะแตกต่างกันไปสำหรับอุปกรณ์ที่แตกต่างกัน โดยการส่งข้อมูลเหล่านี้จะถูกส่งออกไปทีละตัวอักษร (ตามรหัส ASCII) และถูกควบคุมโดยสัญญาณ EOI เพื่อระบุชุดของคำสั่งนั่นเอง

ในการกำหนดการเริ่มติดต่อสื่อสารข้อมูลระหว่างกันนั้นจำเป็นจะต้องระบุ Address ของส่วนที่ทำการติดต่อ โดยในการกำหนดส่วนการเริ่มต้นนี้นั้นจะแบ่งออกได้ 2 กรณีได้แก่ ระหว่างตัวควบคุมกับตัวส่ง หรือตัวควบคุมกับตัวรับ ซึ่งในขณะที่ทางด้านหนึ่งทำหน้าที่เป็นผู้ส่ง อีกด้านหนึ่งก็จะทำหน้าที่เป็นผู้รับนั่นเอง

- ขณะที่ตัวควบคุมติดต่อกับภาคส่ง

ในขณะนี้นั้นตัวควบคุมจะทำหน้าที่เสมือนเป็นทางด้านรับ (Listener) โดยในขณะนี้นั้น ATN จะถูกกำหนดให้เป็น Low เป็นอันดับแรกเพื่อกำหนดให้เข้าสู่โหมดของการสั่งงานคำสั่งมาตรฐาน ดังแผนผังเวลาในรูปที่ 5.2 ซึ่งในส่วนของการกำหนดค่าคำสั่งต่างๆ นั้นมีรายละเอียดดังต่อไปนี้

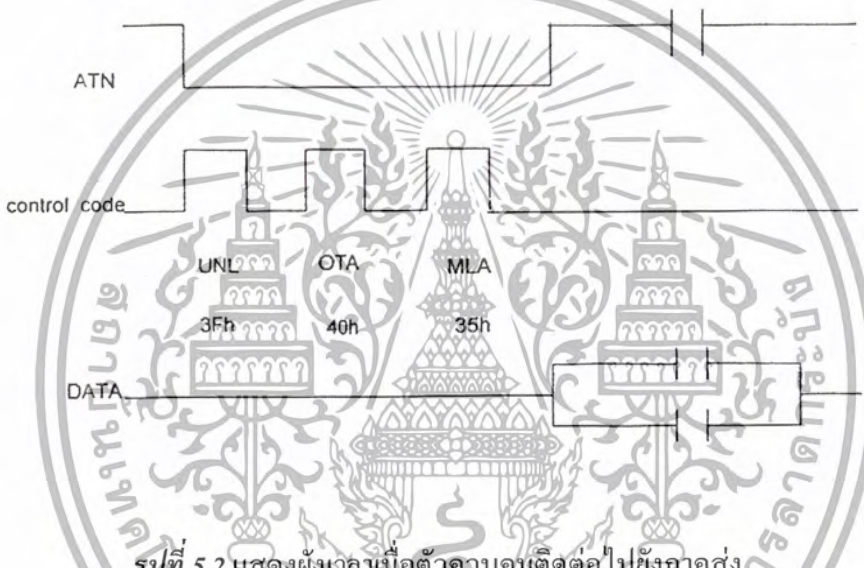
UNL (3Fh) (Unlistener) คำสั่งนี้นั้นเป็นคำสั่งเคลียร์อุปกรณ์ที่ทำหน้าที่เป็นตัวรับที่ต่อพ่วงอยู่ภายในระบบทุกตัว ซึ่งจะทำให้ไม่มีอุปกรณ์ใดในระบบทำหน้าที่เป็นตัวรับอีก

OTA (40h+Address) (Our Talker Address) ในส่วนนี้นั้นเป็นการแจ้งค่า Address ของตัวควบคุมให้อุปกรณ์ที่ต่อพ่วงอยู่บนบัสทุกตัวรับรู้ ซึ่ง Address ของตัวควบคุมนั้นจะเป็น 0 โดยใน

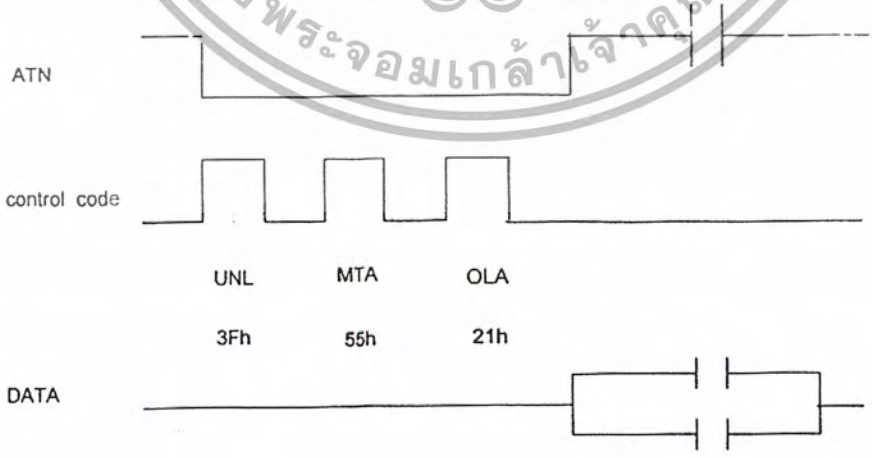
การส่งนั้นจะเป็นการส่งค่า 40h บวกกับค่า Address ของตัวควบคุมไปได้แก่ 0 จึงเป็นการส่งค่า 40h ออกไปนั่นเอง

MLA (20h+Address) (My Listener Address) คำสั่งนี้เป็นการกำหนดค่าของ Addess ของอุปกรณ์ตัวส่งซึ่งเป็นไปในลักษณะเดียวกับกำหนด Address ของตัวควบคุม แต่จะเป็นการส่งค่า 20h บวกกับค่า Address อุปกรณ์ที่ต้องการให้ทำหน้าที่เป็นตัวส่ง ซึ่งในตัวอย่างนี้นั้นได้แก่ 21 (15h) ดังนั้นค่าที่ส่งออกไปจึงเป็น 20h+15h = 35h ดังตัวอย่าง

หลังจากที่คำสั่งเหล่านี้ถูกสั่งงานแล้ว ตัวควบคุมจะทำการสั่งให้สัญญาณ ATN เป็น High เพื่อที่จะเริ่มทำการรับค่าข้อมูลที่จะส่งมาจากตัวส่งต่อไป



รูปที่ 5.2 แสดงผังเวลาเมื่อตัวควบคุมติดต่อไปยังภาคส่ง



รูปที่ 5.3 แสดงผังเวลาเมื่อตัวควบคุมติดต่อไปยังภาครับ

- *ขณะที่ตัวควบคุมติดต่อกับภาครับ*

ในขณะที่ตัวควบคุมทำหน้าที่เป็นภาคส่งเพื่อติดต่อกับภาครับนี้นั้นจะเป็นไปในลักษณะเดียวกับขณะที่ตัวควบคุมทำหน้าที่เป็นภาครับ โดยการสั่งการจะเริ่มต้นจากการที่กำหนดให้สัญญาณ ATN เป็น Low แล้วจึงเริ่มส่งชุดข้อมูลออกไป

UNL (3Fh) (Unlistener) เป็นคำสั่งยกเลิกอุปกรณ์ที่ทำหน้าที่เป็นตัวรับอยู่ทุกตัว เพื่อรอรับคำสั่งใหม่

MTA (40h+Address) (My Talker Address) เป็นคำสั่งที่จะแจ้ง Address ของตัวควบคุมที่จะให้ตัวรับรับรู้ว่าส่งมาจาก Address ใดโดยค่าในการส่งจะเป็น 40h+Address ของตัวคัวควบคุม ซึ่งดังตัวอย่างรูปที่ 5.3 นั้น Address ของตัวควบคุมจะเป็น 21 (15h) ดังนั้นค่าที่ส่งไปจึงเป็น 40h+15h = 55h ดังรูป

OLA (20h+Address) (Our Listener Address) เป็นคำสั่งที่ใช้กำหนด Address ของอุปกรณ์ที่ทำหน้าที่เป็นตัวรับโดยค่าที่ส่งออกไปจะเป็น 20h+Address อุปกรณ์ตัวรับ ซึ่งดังรูปนั้นอุปกรณ์ตัวรับมี Address เป็น 1 ดังนั้นค่าที่ส่งออกไปจึงเป็น 21h นั่นเอง

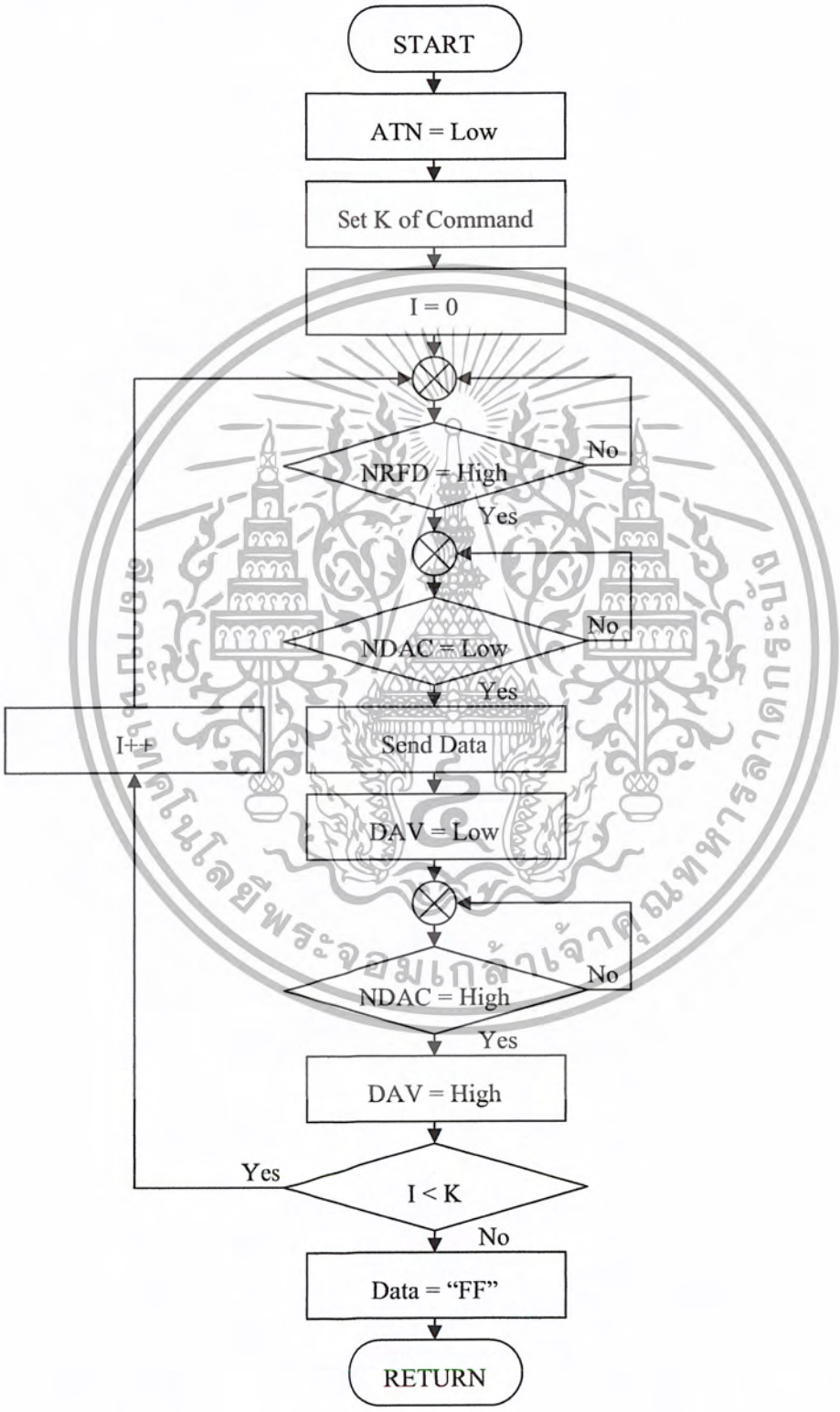
5.4 การออกแบบโปรแกรมขบวนการแฮนด์เช็คในการรับส่งข้อมูล

ในการรับส่งข้อมูลบนระบบบัส GPIB นั้นต้องกระทำให้เป็นไปตามขบวนการแฮนด์เช็คของระบบบัส GPIB (กล่าวไว้โดยละเอียดในบทที่ 2) โดยในการออกแบบให้อยู่ในรูปของโปรแกรมไมโครคอนโทรลเลอร์นั้น แสดงดังรูปที่ 5.4 ซึ่งเป็นโฟล์วชาร์ตแสดงขบวนการแฮนด์เช็คในการส่งรหัสคำสั่งมาตรฐาน และรูปที่ 5.5 เป็นโฟล์วชาร์ตแสดงขบวนการแฮนด์เช็คในการส่งชุดข้อมูลคำสั่ง

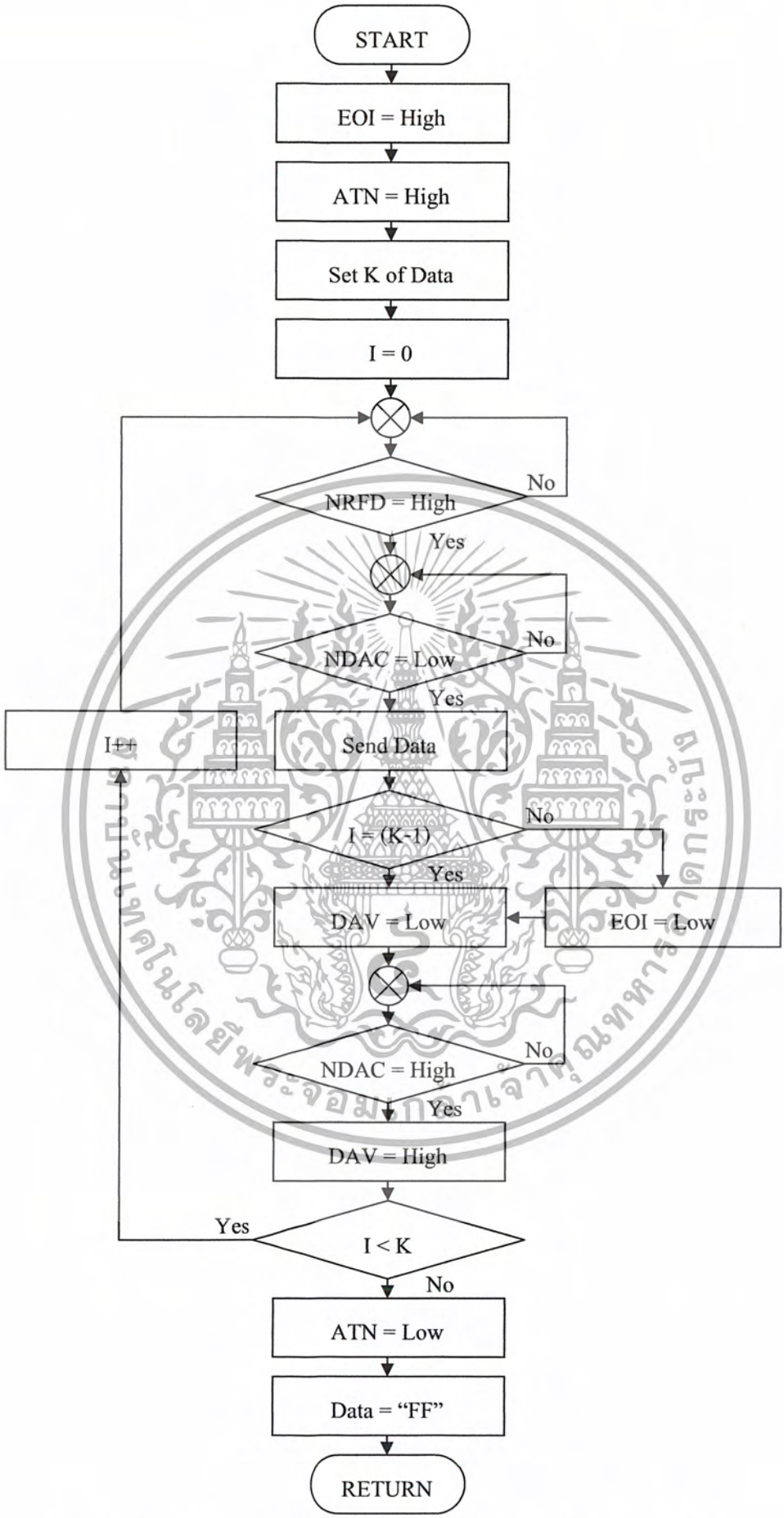
- *ขบวนการแฮนด์เช็คในการส่งรหัสคำสั่งมาตรฐาน*

ในขั้นแรกนั้นเป็นการกำหนดให้สัญญาณ ATN เป็น Low เพื่อบ่งบอกว่าเป็นการส่งรหัสคำสั่งมาตรฐาน (ในสภาวะปรกตินั้นสัญญาณ ATN ก็จะคงอยู่ในสภาวะ Output Low) ต่อมาจะเป็นการกำหนดจำนวดชุดของคำสั่งที่ต้องการจะทำการส่ง 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 ซึ่งเป็นการบ่งบอกว่าข้อมูลที่ส่งไปนั้นทางภาครับได้รับข้อมูลแล้ว ทางภาคส่งก็จะกำหนดให้สัญญาณ DAV เป็น High เพื่อเป็นการบ่งบอกการสิ้นสุดของการส่งข้อมูลใน 1 ชุดข้อมูล หลังจากนั้นจะเป็นการตรวจสอบว่าชุดรหัสคำสั่งนั้นสิ้นสุดแล้ว

หรือไม่ หากยังไม่จบชุดของรหัสก็จะทำการเข้าสู่กระบวนการส่งอีกครั้งเพื่อส่งรหัสคำสั่งที่เหลืออยู่ หลังจากขบวนการส่งจบสิ้นแล้ว ก็จะทำการเคลียร์พอร์ต Data โดยกำหนดข้อมูลให้เป็น FF เป็นอันเสร็จสิ้นขบวนการส่งรหัสคำสั่ง



รูปที่ 5.4 โฟล์วชาร์ตแสดงขบวนการแฮนด์เช็คในการส่งรหัสคำสั่งมาตรฐาน



รูปที่ 5.5 โฟล์วชาร์ตแสดงขบวนการแฮนด์เช็คในการส่งชุดข้อมูลคำสั่ง

- ขบวนการแฮนด์เช็คในการส่งชุดข้อมูลคำสั่ง

ในส่วนของการส่งชุดข้อมูลคำสั่งนั้นส่วนที่แตกต่างจากการส่งรหัสคำสั่งมาตรฐานคือการกำหนดสัญญาณ EOI ซึ่งจะเป็นสัญญาณที่กำหนดแต่ละชุดข้อมูลคำสั่ง โดยขบวนการส่งนั้นในขั้นแรกเป็นการกำหนดสัญญาณ EOI เป็น High เพื่อบ่งบอกถึงการเริ่มต้นของชุดข้อมูล และกำหนดสัญญาณ ATN เป็น High เพื่อบ่งบอกว่าเป็นข้อมูลในรูปแบบชุดข้อมูลคำสั่งหลังจากนั้นจึงกำหนดจำนวนของตัวอักษรที่ต้องการจะส่ง (เช่นคำสั่ง ID? ตัวแปร K จะเท่ากับ 2) และกำหนดตัวแปรนับชุดข้อมูล (I) เป็น 0 หลังจากนี้ หลังจากนั้นจะเริ่มเข้าสู่ขบวนการส่ง โดยตรวจสอบสัญญาณ NRFD ว่าเป็น High หรือไม่ และตรวจสอบสัญญาณ NDAC ว่าเป็น Low หรือไม่ เมื่อเป็นไปตามเงื่อนไขแล้วข้อมูลจะถูกส่งออกไปยังพอร์ต Data หลังจากนั้นจะเป็นการตรวจสอบข้อมูลว่าเป็นข้อมูลชุดสุดท้ายหรือไม่ซึ่งหากเป็นข้อมูลชุดสุดท้าย สัญญาณ EOI จะถูกกำหนดให้เป็น Low แต่หากยังไม่จบชุดข้อมูลสัญญาณ DAV จะถูกกำหนดให้เป็น Low หลังจากนั้นจะเป็นการหลังจากนั้นทางภาคส่งจะตรวจสอบสัญญาณ NDAC อีกครั้งว่าเป็น High หรือไม่ ซึ่งเป็นการบ่งบอกว่าข้อมูลที่ส่งไปนั้นทางภาครับได้รับข้อมูลแล้ว ทางภาคส่งก็จะกำหนดให้สัญญาณ DAV เป็น High เพื่อเป็นการบ่งบอกการสิ้นสุดของการส่งข้อมูลใน 1 ชุดข้อมูล หลังจากนั้นจะเป็นการตรวจสอบว่าชุดข้อมูลคำสั่งนั้นสิ้นสุดแล้วหรือไม่ หากยังไม่จบชุดข้อมูลคำสั่งก็จะทำการเข้าสู่กระบวนการส่งอีกครั้งเพื่อส่งรหัสคำสั่งที่เหลืออยู่ หลังจากขบวนการส่งจบสิ้นแล้ว ก็จะทำการเคลียร์พอร์ต Data โดยกำหนดข้อมูลให้เป็น FF เป็นอันเสร็จสิ้นขบวนการส่งชุดข้อมูลคำสั่ง

5.5 สัญลักษณ์ที่ใช้บ่งบอกความหมายของชุดข้อมูล

เนื่องจากชุดข้อมูลที่จะส่งไปควบคุมอุปกรณ์เครื่องมือวัดต่างๆนั้น จำเป็นจะต้องระบุถึง Address ของอุปกรณ์เครื่องมือวัดที่ต้องการควบคุมด้วย ดังนั้นจึงต้องมีการออกแบบรูปแบบของข้อมูลที่ส่งมาจากทางคอมพิวเตอร์ เพื่อที่จะให้สามารถสื่อสารกันได้เข้าใจ โดยในที่นี้ได้ออกแบบรูปแบบของชุดข้อมูลไว้ดังนี้

@[Address][XXX…..X][#]

โดยมีความหมายของรูปแบบข้อมูลดังต่อไปนี้

@	เครื่องหมาย @ นั้นใช้บ่งบอกถึงจุดเริ่มต้นของชุดคำสั่ง
[Address]	ใช้ระบุ Address ของอุปกรณ์เครื่องมือวัดที่ต้องการติดต่อ
[XXX…..X]	เป็นส่วนของชุดคำสั่งที่ต้องการส่งไปยังไมโครคอนโทรลเลอร์
[#]	เครื่องหมาย # ใช้ในการบ่งบอกให้ไมโครคอนโทรลเลอร์รับทราบว่าคำสั่งนั้นๆ มีข้อมูลส่งกลับมาหรือไม่ ซึ่งหากใน

ชุดคำสั่งมีสัญลักษณ์ # ต่อท้าย จะหมายถึงหลังจากที่ไมโครคอนโทรลเลอร์ส่งชุดคำสั่งนั้นๆไปแล้ว ให้ทำการรอรับค่าที่ส่งกลับมาจากทางอุปกรณ์เครื่องมือวัดด้วย

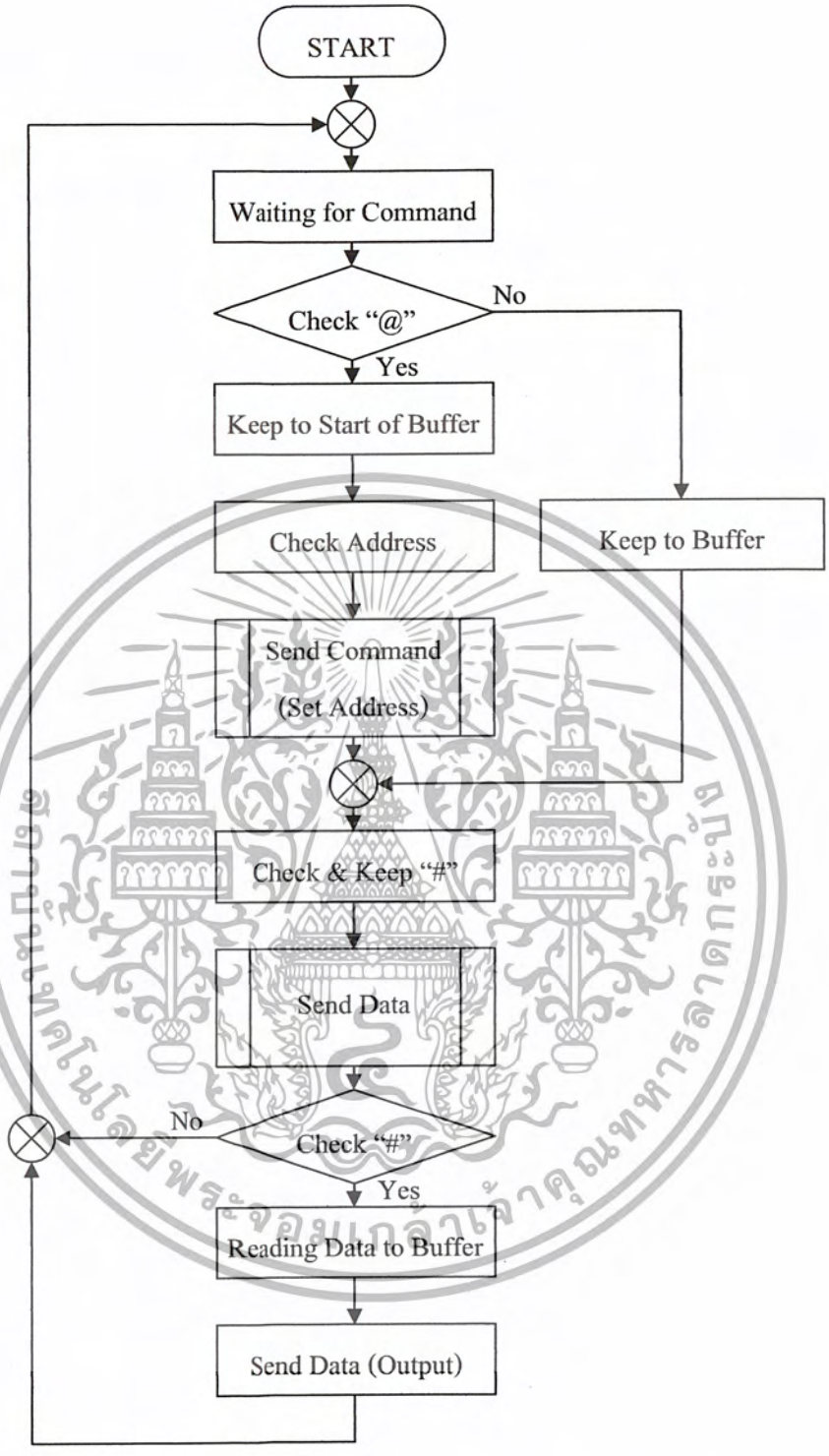
ตัวอย่างชุดข้อมูล

@1SU1 13.13 หมายถึง ให้ทำการติดต่อไปยังอุปกรณ์ที่ต่ออยู่ใน Address ที่ 1 โดยคำสั่งที่ต้องการจะส่งไปคือ "SU1 13.13" ซึ่งหลังจากการส่งคำสั่งแล้วจะไม่มีข้อมูลส่งกลับสามารถรอรับคำสั่งต่อไปได้ทันที

@7FRQ?# หมายถึง ให้ทำการติดต่อไปยังอุปกรณ์ที่ต่ออยู่ใน Address ที่ 7 โดยคำสั่งที่ต้องการจะส่งไปคือ "FRQ?" ซึ่งหลังจากการส่งคำสั่งแล้ว ให้ทำการรอเพื่อรับค่าที่จะตอบรับกลับมาด้วย

5.6 โปรแกรมการทำงานของไมโครคอนโทรลเลอร์

ในส่วนการทำงานของไมโครคอนโทรลเลอร์นั้นแสดงตามรูปที่ 5.6 ซึ่งเป็นโฟลว์ชาร์ตแสดงการทำงานโดยรวมของไมโครคอนโทรลเลอร์ โดยการทำงานจะเริ่มต้นจากการรอรับคำสั่งที่จะส่งมาจากทางคอมพิวเตอร์ ซึ่งคำสั่งที่ได้มานั้นจะต้องทำการตรวจสอบว่าเป็นจุดเริ่มต้นของชุดคำสั่งหรือไม่โดยตรวจสอบจากเครื่องหมาย "@" โดยถ้าไม่พบเครื่องหมาย "@" ข้อมูลจะถูกนำไปเขียนต่อจากข้อมูลเดิมใน Buffer และจะนำข้อมูลนี้ไปรอที่จะทำการส่งต่อไป แต่ถ้าหากตรวจพบเครื่องหมาย "@" ข้อมูลจะถูกนำไปเขียนลงในจุดเริ่มต้นของ Buffer หลังจากนั้นจะเป็นการตรวจสอบ Address ว่าต้องการติดต่อไปยังอุปกรณ์ใด ซึ่ง Address จะถูกส่งไปเพื่อกำหนดอุปกรณ์เครื่องมือวัดที่จะได้รับคำสั่งต่อไป โดยลำดับขั้นตอนการส่งจะเป็นไปตามโฟลว์ชาร์ตที่ 5.4 (Send Data) หลังจากนั้น ข้อมูลใน Buffer จะถูกทำการตรวจสอบเครื่องหมาย "#" ซึ่ง หากพบเครื่องหมายนี้เงื่อนไข Check "#" ที่จะกล่าวถึงต่อไปจะเป็นจริง และเครื่องหมายนี้ก็จะถูกนำออกจากชุดข้อมูลด้วย หลังจากนี้จะเป็นการส่งชุดข้อมูลออกไปยัง Address ที่ได้กำหนดไว้เบื้องต้นแล้ว ซึ่งการส่งนั้นจะเป็นไปตามโฟลว์ชาร์ตที่ 5.5 (Send Command) สำหรับค่า "#" ซึ่งก่อนหน้านี้ได้มีการตรวจสอบและเก็บค่าไว้นั้น หากไม่พบเครื่องหมาย "#" นั่นหมายถึงคำสั่งนี้ ไม่ต้องการข้อมูลตอบกลับ แต่ถ้ามีการตรวจสอบพบ หลังจากการส่งชุดคำสั่งก็จะทำการรอรับค่าที่จะส่งกลับมา เพื่อเก็บไว้ใน Buffer สำหรับค่าที่ได้รับมานั้นจะถูกส่งออกไปในรูปแบบข้อมูลอนุกรม เพื่อส่งให้แก่ทางคอมพิวเตอร์ต่อไป



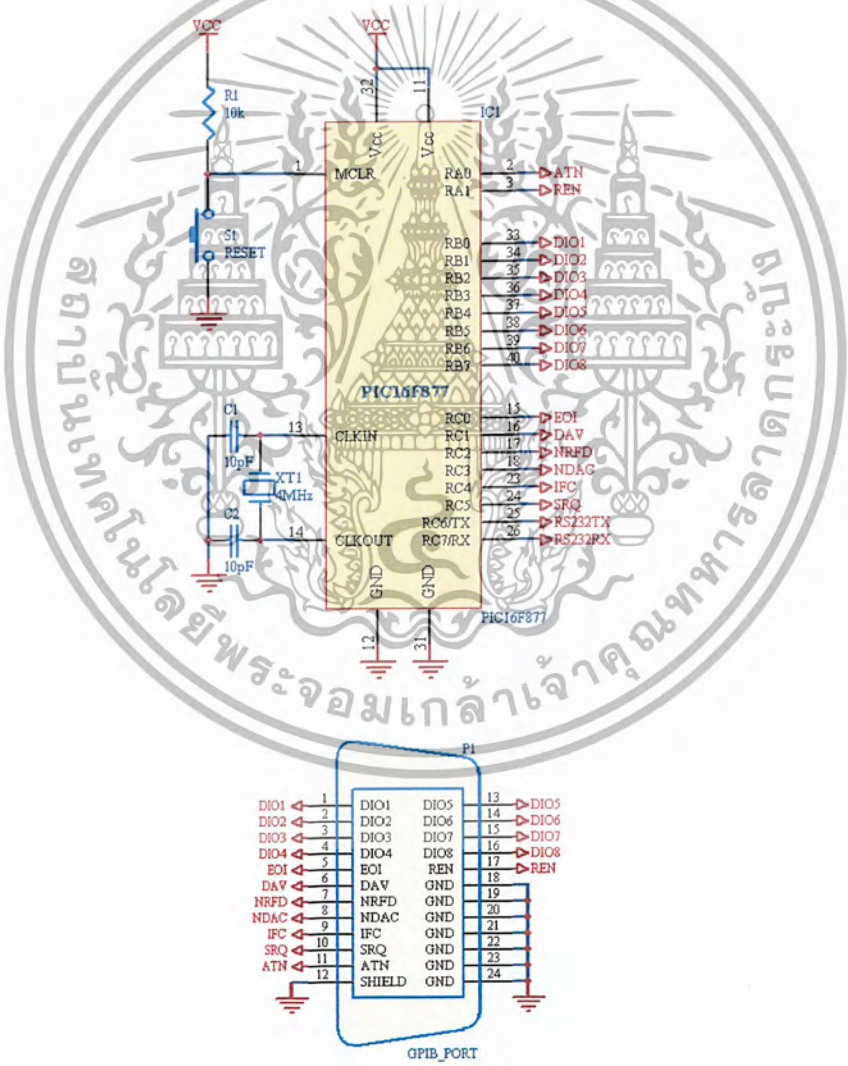
รูปที่ 5.6 โฟล์วชาร์ตแสดงการทำงานของไมโครคอนโทรลเลอร์

5.7 วงจรส่วนการอินเตอร์เฟส และหน้าที่การทำงานของแต่ละส่วนประกอบ

ในส่วนนี้นั้นจะกล่าวถึงวงจรในส่วนต่างๆ ของส่วนอินเตอร์เฟส ซึ่งแบ่งออกเป็นส่วนประกอบหลักๆ ได้ดังนี้

5.8.1 Microcontroller

ส่วนนี้ถือเป็นหัวใจสำคัญของระบบ เนื่องจากทำหน้าที่ในการจัดเรียงข้อมูลที่รับเข้ามาในรูปแบบข้อมูลอนุกรมเพื่อส่งออกไปในมาตรฐาน GPIB หรือในทางกลับกันคือรับข้อมูลจากอุปกรณ์เครื่องมือวัดทางพอร์ต GPIB เพื่อส่งออกไปในรูปแบบข้อมูลอนุกรม ซึ่งในที่นี้ได้เลือกใช้ Microcontroller เบอร์ PIC16F877 โดยมีคุณสมบัติในการจ่ายกระแสของแต่ละพอร์ตมากถึง 25mA (Sink/Source 25mA) ซึ่งเพียงพอที่จะจ่ายไปสั่งการอุปกรณ์เครื่องมือวัดที่ต่อร่วมกันอยู่หลายๆ ตัวได้ อีกทั้งคุณสมบัติทางด้านความเร็วในการทำงาน โดยไมโครคอนโทรลเลอร์ในตระกูลนี้จะใช้สัญญาณนาฬิกาเพียง 1 หรือ 2 สัญญาณในการทำงาน 1 คำสั่ง ทำให้สามารถทำงานได้เร็วกว่าไมโครคอนโทรลเลอร์บางตระกูลที่ความถี่เดียวกัน

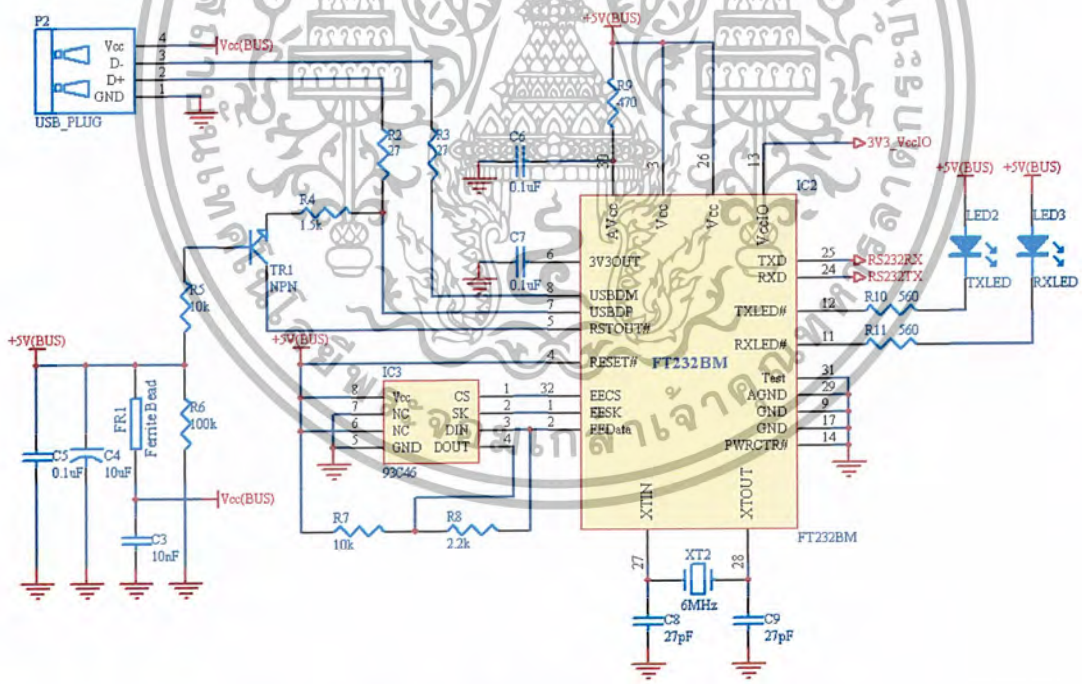


รูปที่ 5.7 แสดงการเชื่อมต่อวงจรไมโครคอนโทรลเลอร์และพอร์ต GPIB

จากวงจรจะเห็นได้ว่าพอร์ต B จะถูกใช้งานเป็นพอร์ต Data ในการส่งข้อมูล ซึ่งค่า Data ต่างๆ ที่จะส่งไปยัง GPIB นั้น ต้องรำลึกไว้เสมอว่าข้อมูลในมาตรฐาน GPIB นั้นจะเป็นในลักษณะตรงกันข้าม (Invert) กับข้อมูลที่คอมพิวเตอร์รับรู้ ดังนั้นค่าที่จะส่งหรือรับมานั้น ต้องทำการ Invert ค่าก่อนเสมอ สำหรับสัญญาณอื่นๆนั้น ต่อร่วมกับพอร์ตต่างๆ ตามรูปวงจร และสำหรับในส่วนของการสื่อสารไปยังคอมพิวเตอร์ จะเป็นการสื่อสารผ่านทาง USART (Universal Synchronous Asynchronous Receiver Transmitter) ซึ่งเป็นพอร์ตสื่อสารอนุกรมในตัวไมโครคอนโทรลเลอร์เอง โดยการติดต่อสื่อสารไปยังคอมพิวเตอร์โดย USB และ RS-232 นั้นจะกล่าวถึงต่อไป

5.8.2 USB Converter

สำหรับส่วนนี้นั้นเป็นส่วนที่ทำหน้าที่ในการเปลี่ยนแปลงการสื่อสารอนุกรมจากทางไมโครคอนโทรลเลอร์ไปสู่การสื่อสารผ่านระบบบัส USB ซึ่งในที่นี้นั้นได้เลือกใช้ IC เบอร์ FT232BM ของบริษัท Future Technology Devices International Limited : FTDI ซึ่งมีลักษณะการส่งถ่ายข้อมูลภายในแบบ FIFO โดย IC นี้มีความสามารถในการจัดการกับโปรโตคอล USB ได้อย่างสมบูรณ์ โดยไม่ต้องอาศัยเฟิร์มแวร์ในการติดต่อกับระบบบัส USB เพิ่มเติม



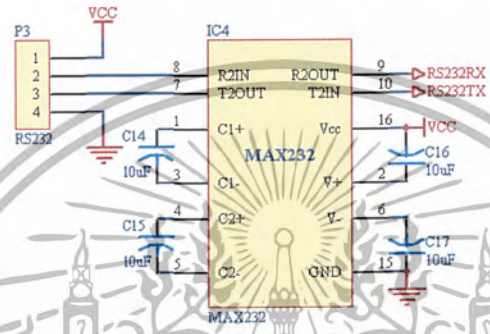
รูปที่ 5.8 แสดงวงจรในส่วนของ USB Convertor

สำหรับในการติดต่อกับโปรแกรมที่เขียนขึ้นทางคอมพิวเตอร์นั้น จะใช้ไดร์เวอร์พอร์ตอนุกรมเสมือน (Vitual COM Port : VCP) ซึ่งทำให้การเชื่อมต่อผ่านทาง USB ถูกมองเป็นการ

เชื่อมต่อในลักษณะของพอร์ตอนุกรม (COM) ทำให้มีความสะดวกในการเขียนโปรแกรมติดต่อไปยังพอร์ต USB เป็นอย่างมาก

5.8.3 RS-232

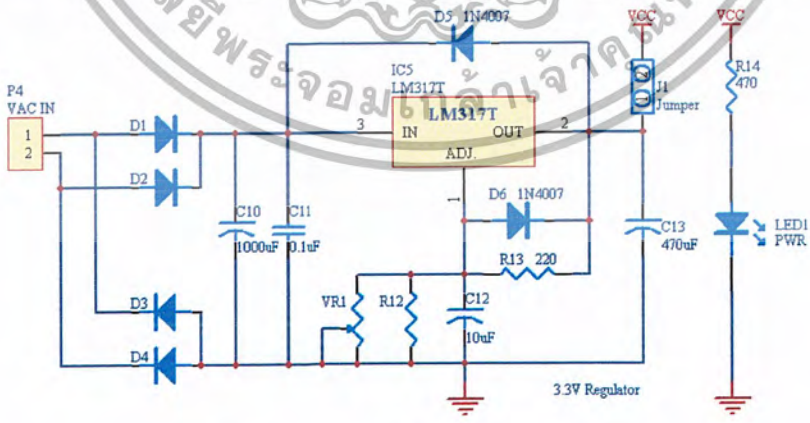
วงจรในส่วนนี้จะทำหน้าที่เปลี่ยนแปลงการสื่อสารอนุกรมของไมโครคอนโทรลเลอร์ให้เป็นไปตามมาตรฐานของการสื่อสารทางคอมพิวเตอร์ ซึ่งมีค่าลอจิกที่สูงกว่า โดยวงจรที่ใช้เป็นวงจรมาตรฐานซึ่งใช้ IC MAX232 ทำหน้าที่เป็นหลัก



รูปที่ 5.9 แสดงวงจรในส่วน RS-232

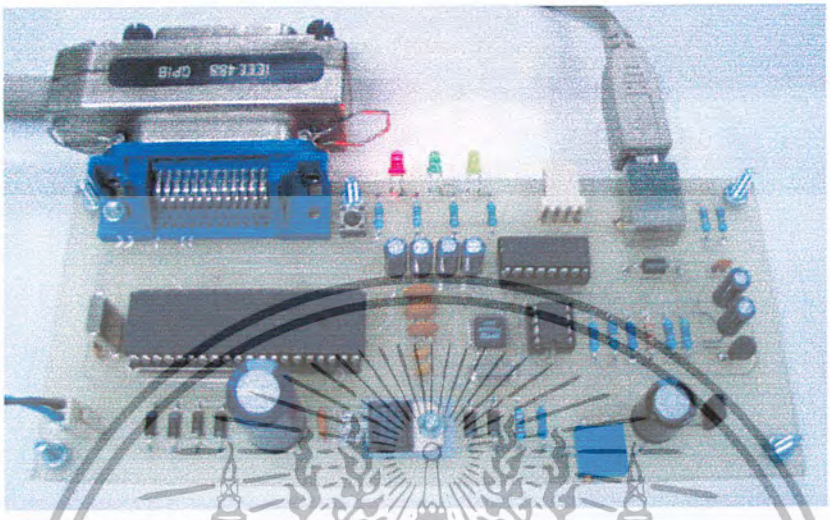
5.8.4 Regulator

ในส่วนนี้ทำหน้าที่ในการสร้างแรงดันไฟฟ้า 3.3V ซึ่งเป็นระดับแรงดันที่ใช้เป็น Output High ของ GPIB โดยในที่นี้นั้นใช้ IC LM317T ซึ่งเป็น IC Regulator 3 ขาที่พบเห็นได้โดยทั่วไป โดยวงจรจะตั้งค่าแรงดันได้จาก VR1

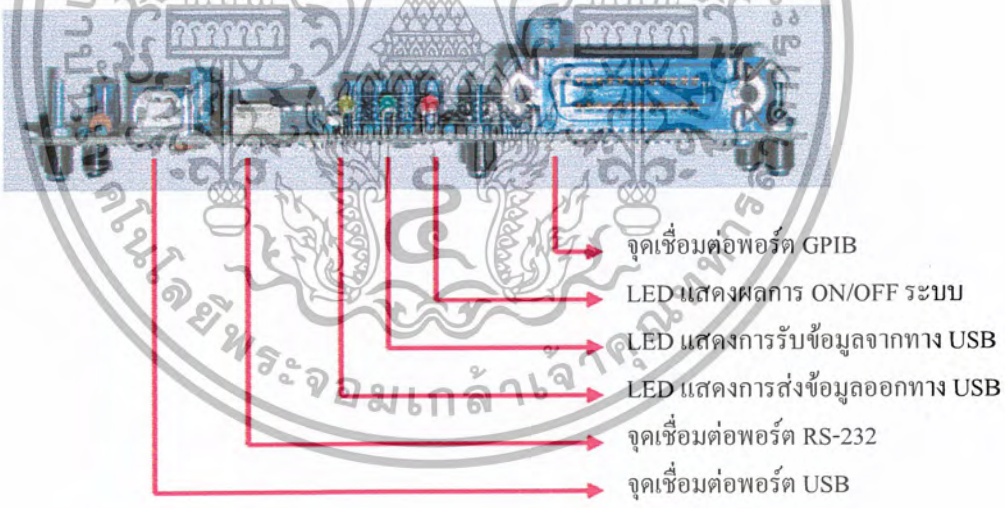


รูปที่ 5.10 แสดงวงจรในส่วนของ Regulator

จากส่วนประกอบต่างๆที่ได้กล่าวมานั้น ได้ถูกออกแบบไว้บนแผ่น PCB เดียวกันเพื่อทำหน้าที่เป็นการ์ดอินเตอร์เฟสดังรูปที่ 5.11 และรูปที่ 5.12 สำหรับผลการทดลองใช้งานการ์ดอินเตอร์เฟสนี้นั้นจะกล่าวถึงในบทต่อไป



รูปที่ 5.11 แสดงการ์ดอินเตอร์เฟสที่ได้ทำการออกแบบ



รูปที่ 5.12 แสดงตำแหน่งของจุดเชื่อมต่อพอร์ตและ LED แสดงผล

บทที่ 6
ผลการทดลอง

ในการทดสอบส่วนควบคุมการอินเตอร์เฟสนั้น จะแบ่งการทดสอบออกเป็นสองส่วน ได้แก่ การทดสอบโดยใช้โปรแกรม Hyper Terminal และการทดสอบโดยโปรแกรมสั่งการซึ่งเขียนขึ้นโดยใช้โปรแกรม Microsoft Visual Basic โดยข้อมูลที่ได้รับมานั้นจะเป็นไปในรูปแบบเดียวกัน แต่ข้อมูลที่ได้รับมาจากอุปกรณ์บางตัวนั้นไม่สามารถตรวจสอบค่าได้โดยตรง เช่น ข้อมูลจากหน้าจอของ Scope ซึ่งจะส่งกลับมาในรูปแบบของชุดข้อมูลปริมาณมาก จึงต้องนำข้อมูลเหล่านั้นมาทำการพล๊อตค่าเพื่อตรวจสอบรูปสัญญาณ

6.1 Address ของอุปกรณ์เครื่องมือวัดที่นำมาทดสอบ

ดังที่ได้กล่าวไว้แล้วว่าในการสั่งการคำสั่งใดๆ จำเป็นจะต้องระบุ Address ของอุปกรณ์เครื่องมือวัดที่ต้องการสั่งการ ซึ่งการส่งคำสั่งแต่ละคำสั่งก็จะมี Address เหล่านี้กำหนดอยู่ดังนั้นในการเขียนโปรแกรมเพื่อรองรับกับอุปกรณ์ใดๆ จึงต้องทราบถึง Address ของอุปกรณ์เหล่านั้นเสียก่อน โดยในการทดลองนี้นั้นจะกำหนด Address ของอุปกรณ์ ดังนี้

อุปกรณ์	*Address*
HAMEG Programmable Power Supply รุ่น HM8142	1
HAMEG Programmable Function Generator รุ่น HM8130	3
HAMEG Programmable Multimeter รุ่น HM8112-2	7
HAMEG Storage Analog/Digital Oscilloscope รุ่น HM 1007	9

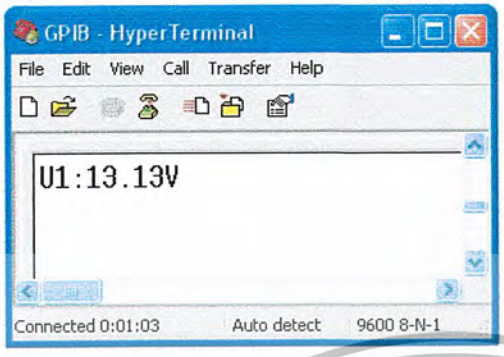
6.2 การทดลองโดยใช้ Hyper Terminal

การทดลองในส่วนนี้จะใช้โปรแกรม Hyper Terminal ซึ่งเป็นโปรแกรมมาตรฐานที่ติดตั้งอยู่บนระบบปฏิบัติการ Windows โดยทั่วไป ซึ่งเป็นโปรแกรมที่จะติดต่อสื่อสารข้อมูลจากผู้ใช้ไปสู่พอร์ตสื่อสารต่างๆได้โดยการคีย์ค่าซึ่งค่าที่ผู้ใช้คีย์ลงไปนั้นจะไม่ปรากฏบนหน้าจอ แต่ค่าที่ปรากฏบนหน้าจอคือค่าที่ได้รับมาจากพอร์ตสื่อสารนั้น

การทดลองส่งคำสั่งไปยัง Power Supply (Address 1) เพื่ออ่านค่าแรงดันของ Channel 1

คำสั่งที่ส่งไป "@1MU1#"

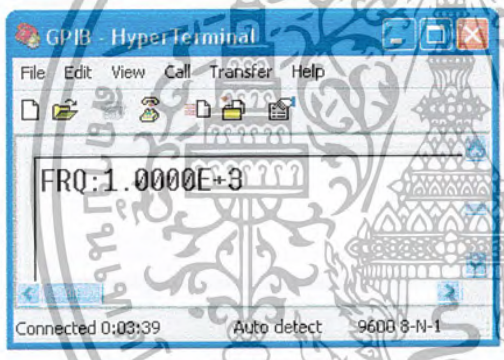
ผลที่ได้รับ



การทดลองส่งคำสั่งไปยัง Generator (Address 3) เพื่ออ่านค่าความถี่ของ Generator

คำสั่งที่ส่งไป "@3FRQ?#"

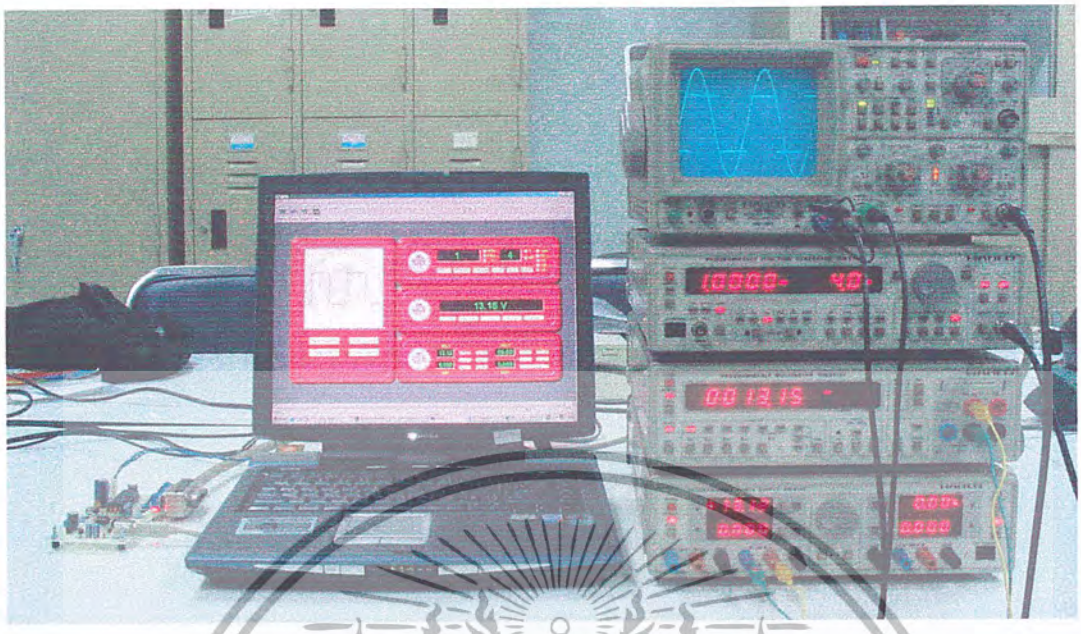
ผลที่ได้รับ



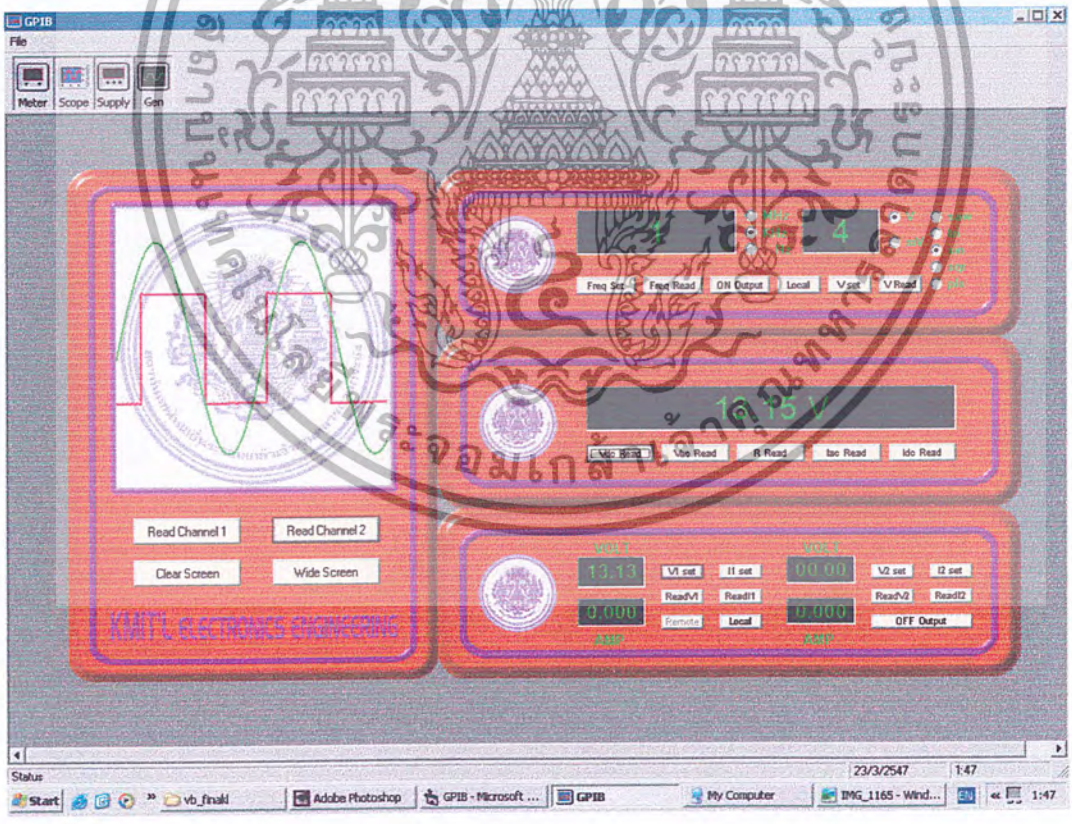
ในการทดลองส่วนนี้จะไม่กล่าวถึงมากนัก เนื่องจากไม่สามารถแสดงความสัมพันธ์ของคำสั่งที่ส่งไปและค่าที่ได้รับกลับมาได้โดยง่าย ดังนั้นจะกล่าวถึงการทดลองโดยละเอียดในส่วนของการทดลองโดยใช้โปรแกรมสั่งการที่เขียนขึ้นต่อไป

6.3 การทดลองโดยใช้โปรแกรมสั่งการที่เขียนขึ้น

โปรแกรมที่เขียนขึ้นนี้พัฒนาขึ้นโดยโปรแกรม Microsoft Visual Basic ซึ่งในการทดลองนั้นจะเป็นเสมือนการนำเอาคำสั่งต่างๆ ที่ต้องการจะสั่งการมาแทนด้วยการคลิ๊กปุ่มสั่งการต่างๆ แทน โดยคำสั่งที่บรรจุอยู่ภายในโปรแกรมนั้นสามารถตรวจสอบได้จาก บทที่ 4 สำหรับหน้าต่างของโปรแกรมนั้น แสดงไว้ในรูปที่ 6.2 สำหรับการทดลองนั้น จะแบ่งออกเป็น 2 การทดลองหลัก ดังนี้



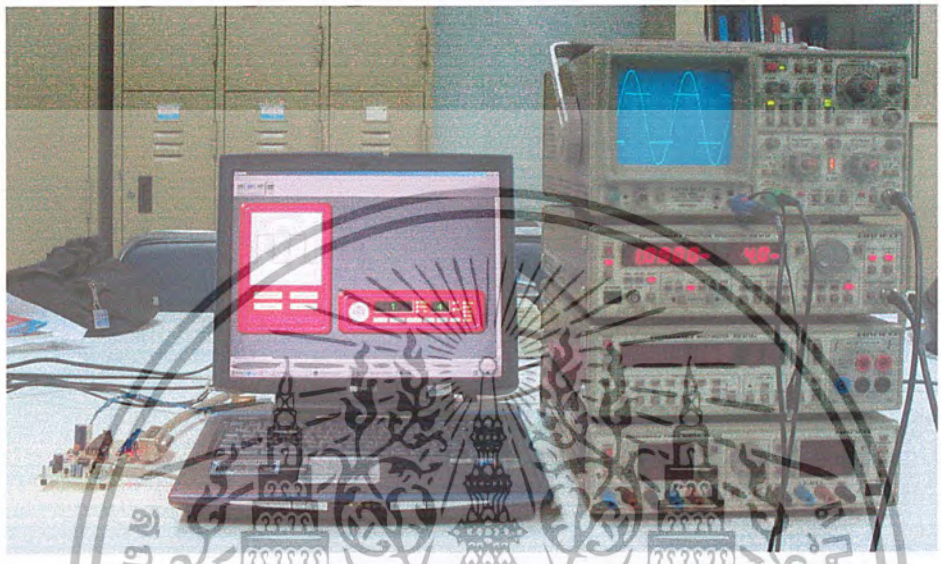
รูปที่ 6.1 องค์ประกอบโดยรวมในการทดลอง



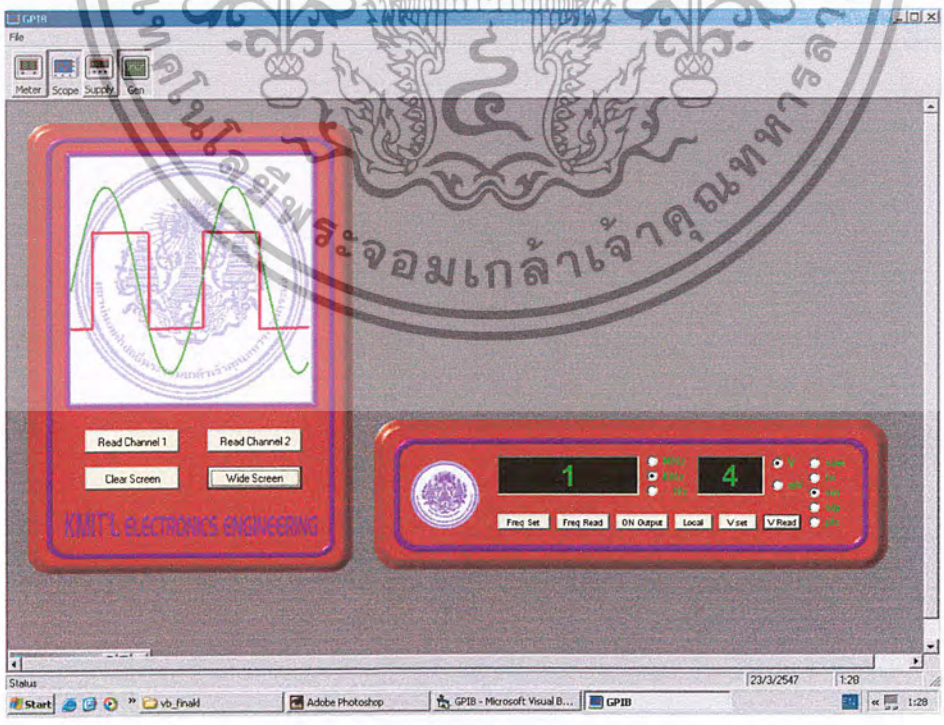
รูปที่ 6.2 หน้าต่างโปรแกรมสั่งการที่เขียนขึ้น

6.3.1 การทดลองที่ 1

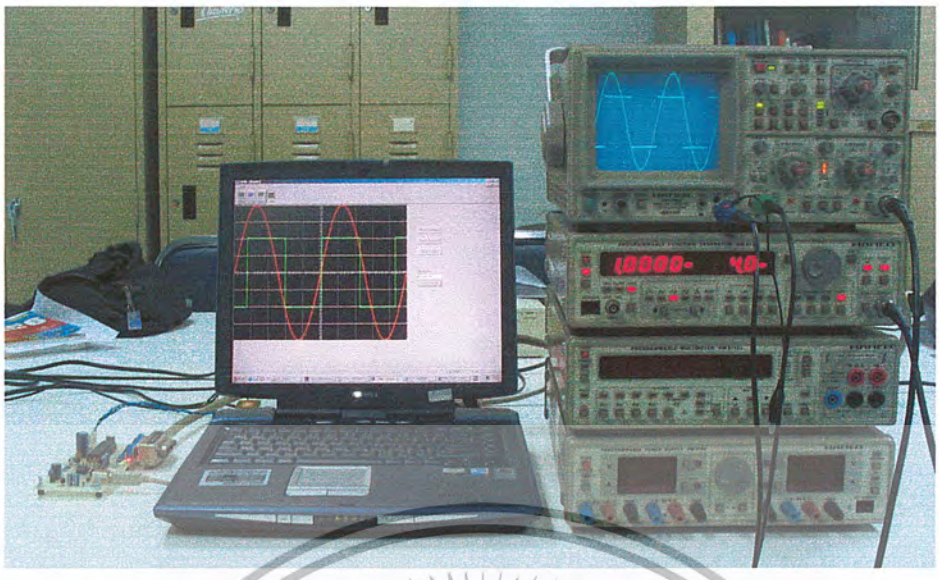
ในการทดลองนี้จะใช้ Ocilloscope ต่อร่วมกับ Function Generator โดยทำการตั้งค่าความถี่และแรงดันเอาท์พุตจากทางโปรแกรม (ในที่นี้ทดลองโดยการตั้งสัญญาณ Sine 1kHz 4V) โดยสัญญาณ Ch1 ของ Oscilloscope นำมาจากสัญญาณพัลซ์ของตัว Oscilloscope เองและสัญญาณ Ch2 ต่อเข้ากับ Function Generator ซึ่งผลการทดลองนั้นเป็นไปดังภาพที่ 6.3 – 6.6



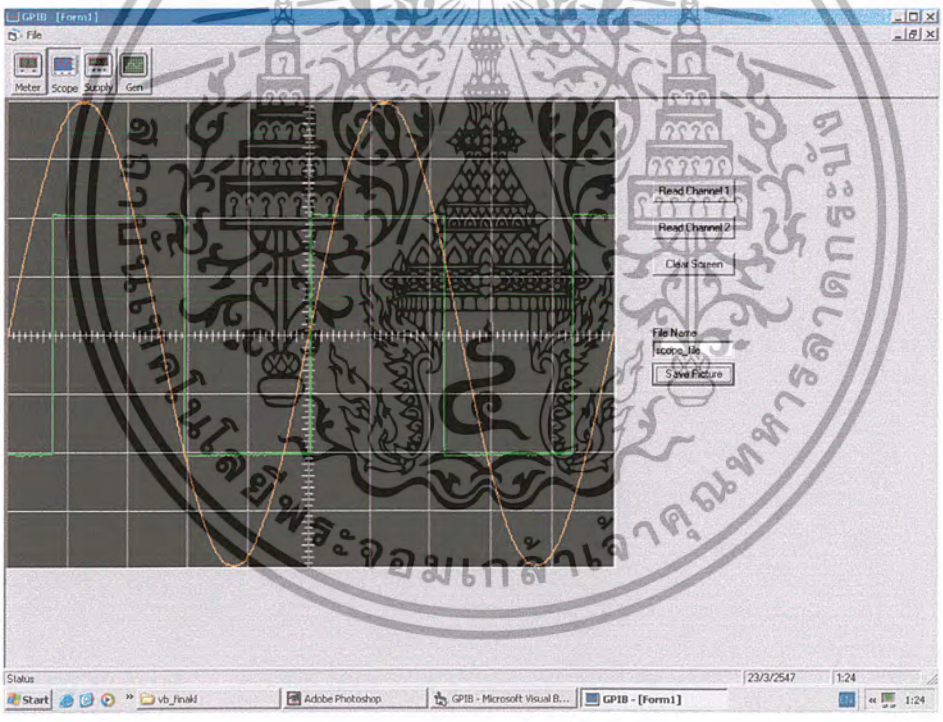
รูปที่ 6.3 แสดงผลการทดลองที่ 1 (ทางด้านอุปกรณ์เครื่องมือวัด)



รูปที่ 6.4 แสดงผลการทดลองที่ 1 (ทางด้านโปรแกรม)



รูปที่ 6.5 แสดงผลการทดลองที่ 1 ในโหมด Wide Screen (ทางด้านอุปกรณ์เครื่องมือวัด)



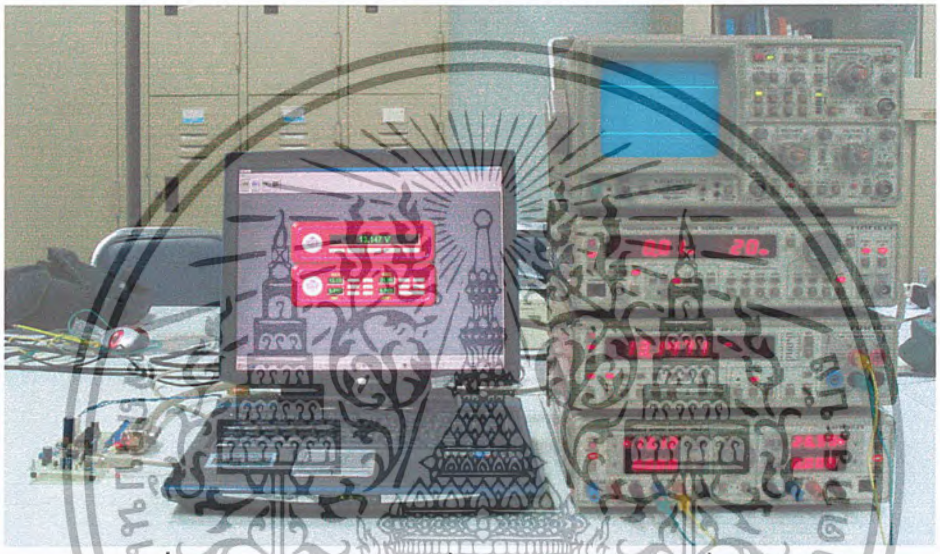
รูปที่ 6.6 แสดงผลการทดลองที่ 1 ในโหมด Wide Screen (ทางด้านโปรแกรม)

ในส่วนของ Ocilloscope นั้นโดยปรกติค่าที่แสดงอยู่บนหน้าต่างนั้นจะไม่มี Scale ในการอ่านค่าใดๆ ซึ่งหากต้องการดูรูปสัญญาณโดยละเอียดก็สามารถทำได้โดยการตรวจสอบสัญญาณในโหมด Wide Screen ซึ่งผลการทดลองแสดงอยู่ในรูปที่ 6.5 และ 6.6 นอกจากนี้สัญญาณที่ได้มายังสามารถเก็บไว้ในรูปแบบของไฟล์ BMP อีกด้วย

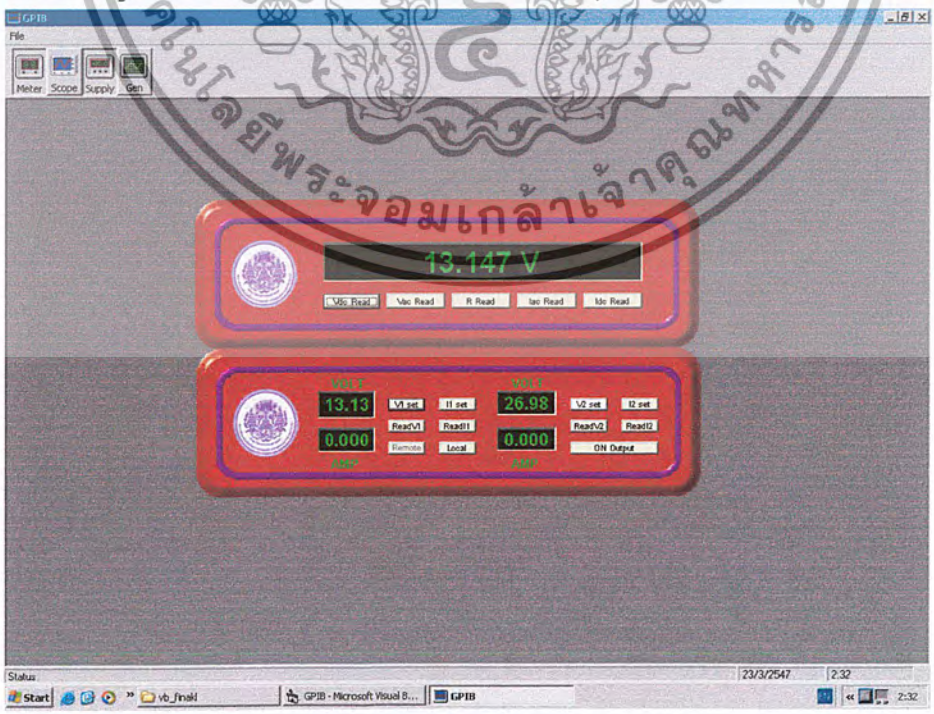
ทั้งนี้ค่าที่ได้มาจาก HAMEG Storage Analog/Digital Oscilloscope รุ่น HM 1007 นี้นั้นมีจุดด้อยที่สำคัญคือ ไม่สามารถกำหนดย่านของแรงดันต่อช่อง (V/Div) และค่าของเวลาต่อช่อง (Time/Div) ในการวัดได้ ทำให้ขาดความยืดหยุ่นในการวัดสัญญาณที่มีความถี่แตกต่างกันไป

6.3.1 การทดลองที่ 2

ในส่วนของการทดลองนี้จะเป็นการทดลองในส่วนของ DC Power Supply และ Digital Multimeter ซึ่ง Power Supply จะถูกต่อร่วมอยู่กับ Digital Multimeter เพื่อวัดค่าแรงดันเอาท์พุทที่กำหนดจากทางโปรแกรม และอ่านค่าที่ได้มาแสดงบนหน้าจอโปรแกรม ดังรูปที่ 6.7 และ 6.8



รูปที่ 6.7 แสดงผลการทดลองที่ 2 (ทางด้านอุปกรณ์เครื่องมือวัด)



รูปที่ 6.8 แสดงผลการทดลองที่ 2 (ทางด้านโปรแกรม)

บทที่ 7
บทสรุปและแนวทางการพัฒนา

หากกล่าวถึงการประยุกต์การใช้งานระบบควบคุมอุปกรณ์เครื่องมือวัดระยะไกลนั้นจะเห็นได้ว่าสามารถประยุกต์ใช้งานได้อย่างหลากหลาย ซึ่งโปรแกรมที่เขียนขึ้นเพื่อทดสอบการทำงานในปริญญานิพนธ์นี้นั้น เป็นเพียงตัวอย่างหนึ่งของการใช้งานเท่านั้น ซึ่งในทางปฏิบัติสามารถออกแบบโปรแกรมให้รองรับกับรูปแบบของงานตามต้องการ รวมถึงการออกแบบให้เป็นระบบควบคุมอัตโนมัติอีกด้วย

สำหรับแนวทางในการพัฒนานั้น จะเห็นได้ชัดเจนอย่างหนึ่งว่าส่วนของการ์ดอินเตอร์เฟสที่ออกแบบขึ้นนั้น ใช้รูปแบบการสั่งคำสั่งเฉพาะที่ผู้ออกแบบได้ออกแบบไว้ ซึ่งหากพัฒนาให้ให้การสั่งการเป็นไปตามมาตรฐานสากลที่ใช้ในการควบคุมอุปกรณ์เครื่องวัดโดยทั่วไป ก็จะทำให้ผู้ใช้งานที่คุ้นเคยกับมาตรฐานสากลสามารถใช้งานได้สะดวกขึ้น สำหรับในส่วนที่ทำหน้าที่จัดเรียงข้อมูลนั้น ในปริญญานิพนธ์นี้ได้ใช้ไมโครคอนโทรลเลอร์ในการทำหน้าที่ดังกล่าว ซึ่งหากกล่าวถึงเสถียรภาพ และการผลิตในจำนวนมากนั้น ยังด้อยกว่าการออกแบบโดย FPGA ซึ่งมีเสถียรภาพสูงเนื่องจากมีโครงสร้างการทำงานที่ใกล้เคียงการทำงานในระดับลอจิกเกท อีกทั้งสามารถส่งผลิตในเชิงพาณิชย์ได้เป็นอย่างดี

ข้อควรระวังอย่างยิ่งในการใช้งานการ์ดอินเตอร์เฟสนี้คือ ในขณะที่ทำการติดตั้งสาย GPIB CABLE เข้ากับการ์ดอินเตอร์เฟสนั้น ควรพึงระวังไว้เสมอว่าอุปกรณ์เครื่องมือวัดต่างๆ รวมถึงส่วนของการ์ดอินเตอร์เฟสนั้น ควรจะอยู่ในสภาวะ OFF ทั้งหมด เนื่องจากการ์ดอินเตอร์เฟสนี้ได้ใช้พอร์ตของไมโครคอนโทรลเลอร์ต่อเพื่อควบคุมอุปกรณ์เครื่องมือวัดโดยตรง ซึ่งพอร์ตของไมโครคอนโทรลเลอร์นี้มีข้อจำกัดทางด้านกระแสที่ไม่สูงนัก (~25mA) ดังนั้นในการเชื่อมต่อพอร์ตของไมโครคอนโทรลเลอร์ เข้ากับอุปกรณ์เครื่องมือวัดโดยตรงในขณะที่อุปกรณ์เครื่องมือวัดทำงานอยู่นั้น สภาวะลอจิกที่คงค้างอยู่ของแต่ละอุปกรณ์ อาจก่อให้เกิดกระแสเฉียบพลันซึ่งอาจมีค่าสูงเพียงพอที่จะทำให้พอร์ตของไมโครคอนโทรลเลอร์เสียหายได้

บรรณานุกรม

[1] HAMEG Instruments. "HO88 Manual" www.hameg.com

[2] นพดล มณีรัตน์. "การออกแบบและสร้างระบบการส่งข้อมูลแบบขนานสำหรับการควบคุมเครื่องมือวัดผ่านระบบบัสมาตรฐาน IEEE-488 (GPIB)" วิทยานิพนธ์บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2540.

[3] ลภณ สุภาพ และคณะ. "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ต USB ขั้นพื้นฐาน" Innovative Experiment Co., Ltd.

[4] วรเทพ ไพบูลย์รัตนากร, บุญอนันต์ เกียงเอีย. "สัมผัสโลก USB ด้วย Ezy USB Module" Astron Logic Research & Development. 2537.

[5] วัชรินทร์ เคารพ. "เรียนรู้และเข้าใจสถาปัตยกรรมไมโครคอนโทรลเลอร์ PIC16F877" ETT Co., Ltd. 2537.

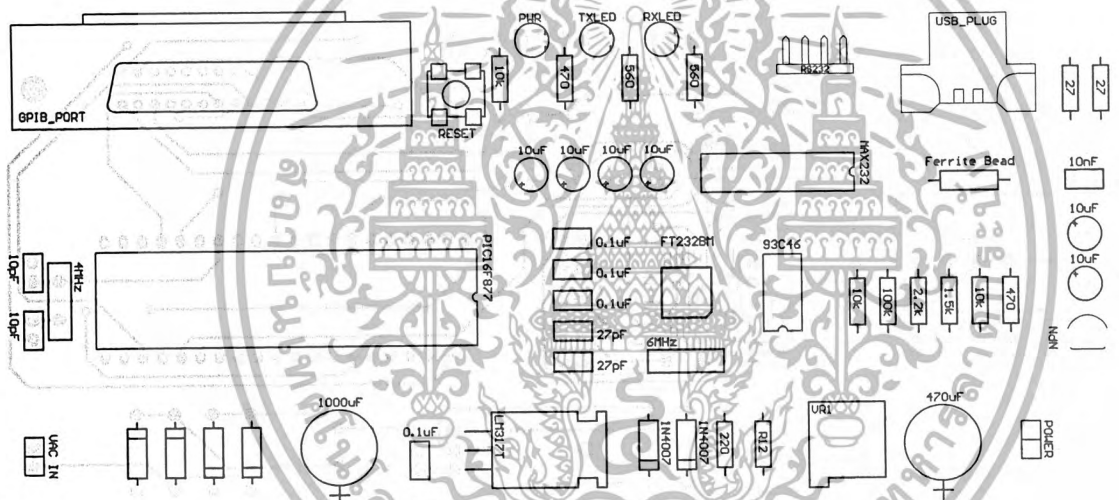
[6] ฉัททวุฒิ พืชผล, พิชิต สันติกุลานนท์. "คู่มือเรียน Visual Basic 6" Provision Co. Ltd. 2544.

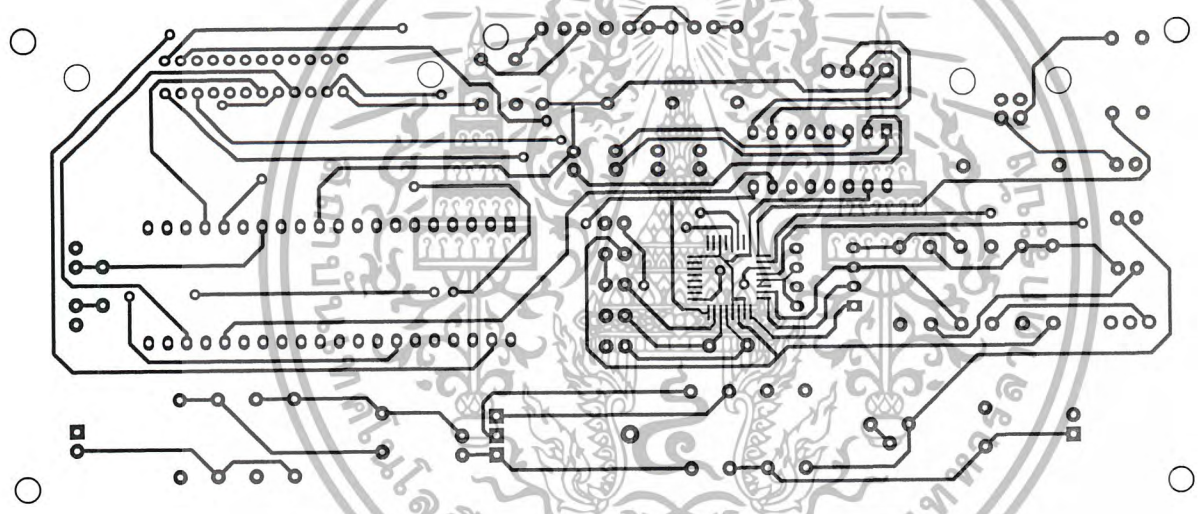
[7] สัจจะ จรัสรุ่งรวีวร. "คู่มือการเขียนโปรแกรมและใช้งาน Visual Basic .NET ฉบับสมบูรณ์" Inforpress Developer Book. 2537.

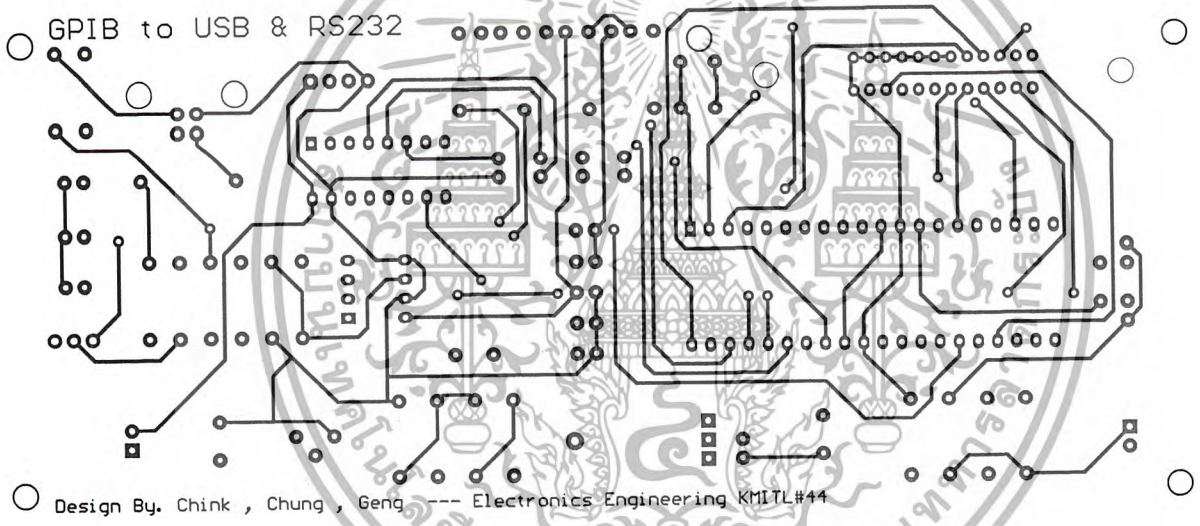
[8] อรรถพล บุญยะโภคา และคณะ. "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม" Innovative Experiment Co., Ltd.

[9] อภิชาติ ภู่พลับ. "เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic" Inforpress Developer Book. 2546.

ภาคผนวก

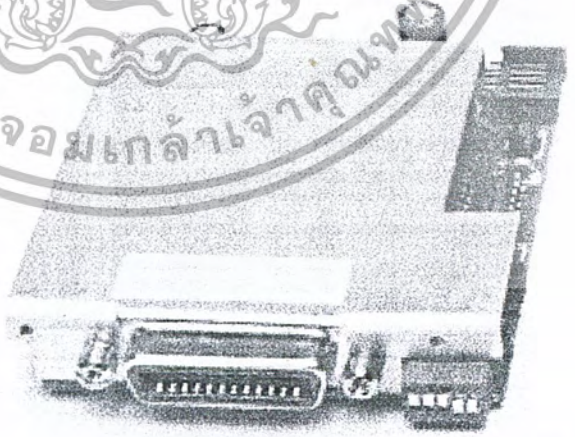

GPIB_PORT
RESET
PWR
TXLED
RXLED
10k
470
560
560
RS232
USB_PLUG
27
27
10uF
10uF
10uF
10uF
MAX232
Ferrite Bead
10nF
10uF
10uF
10pF
10pF
4MHz
PIC16F872
0.1uF
0.1uF
0.1uF
27pF
27pF
6MHz
FT232BM
93C46
10k
100k
2.2k
1.5k
10k
470
NPN
1000uF
0.1uF
LM317T
1N4007
1N4007
220
R12
VR1
470uF
POWER
VAC IN




GPIB to USB & RS232
Design By. Chink , Chung , Geng --- Electronics Engineering KMITL#44

HAMEG®
Instruments

HO88



MANUAL • HANDBUCH • MANUEL

Operation of test instruments using the IEEE-488 bus

The IEEE-488 bus, also known as the General Purpose Interface Bus (GPIB), is a standardized interface system through which interconnected test instruments (e.g. multimeters, power supplies, etc.) or peripherals (e.g. printers, plotters, etc.) and a controller (which can be a computer) can communicate. There are two types of data which can be transferred over the bus: socalled interface messages and device-dependent messages.
The Controller can instruct a device to receive data or transmit data. Devices able to receive data are designated as Listeners. Devices that can transmit data are called Talkers. A given device can be a Listener, a Talker, or both. The designations „Talker" and „Listener" are defined in the IEEE-488 standard, and will be used here as well to indicate the different device states.

History of the IEEE-488 bus

Originally designed by Hewlett-Packard as the Hewlett-Packard Interface-Bus (HP-IB) and marketed from 1965 on for interconnection and control of programmable instruments, the HPIB standard was adopted in 1975 by the IEEE committee in the United States as the IEEE-488 standard, being officially labelled the General Purpose Interface Bus, and in 1977 in Europe as the IEC 625-1 standard. The European and American standards use different connectors:

- IEC 625-1: 25-pin connector
- IEEE-488: 24-pin connector

However, in spite of their different names and choice of connector types, the European IEC 625-1 bus, the American IEEE-488 or GPIB bus, and the HP-IB bus are fully compatible with one another as regards electrical levels and connections, and where programming is concerned. Adapters are available for mating IEEE-488 connectors with IEC-625 connectors. In the following, the term IEEE-488 bus will be used, since the 24-pin connector specified by the American standard is used by HAMEG's test instruments.

Capabilities of the IEEE-488 bus

For operation of a number of devices connected to a bus, each device is assigned a unique address called the primary address between 0 and 30. On HAMEG equipment, the primary address is selected using DIP switches. The System Controller uses these primary addresses to route information and commands over the bus to the proper instrument or device. In order to instruct a device to begin sending data, the System Controller addresses it as a Talker. To enable an instrument's receiver function, it is addressed as a Listener.

Not every instrument has both Talker and Listener functions, like multimeters, for instance. Some devices can only act as Listeners (e.g. printers), and some only have Talker capabilities (such as a voltmeter that only sends measurements). Only one device at a time on an IEEE-488 bus may be enabled as a Talker, while any number can be Listeners at once. If two or more computers are connected to a bus, only one of them may act as the System Controller at any given time. It then has active controller status, and is referred to as the Controller-in-Charge (CIC). It can also address the other computers and instruct them to send or receive data, or transfer active control to another computer and become an idle controller, i.e. it can then be instructed to send or receive data as a Talker or Listener by the new CIC.
It is also possible for data to be sent over the bus even without the mediation of a controller. This works if the transmitting device has „Talk-only" status, and if the receiving device (or devices) has „ Listen-only" status. Data may then be passed from the former to the latter without their being addressed first.

Physical configuration of the IEEE-488 bus

The IEEE-488 bus uses 16 signal lines, 7 ground return lines and one shield drain line. Both of the valid standards - IEC- 625 of the International Electrotechnical Commission

and IEEE-488 of the Institute of Electrical and Electronic Engineers - contain binding stipulations for assignment of the lines to the device connectors (Figure 1). Either ribbon cables or round cables with single or double shielding and twisted conductors are used.

Up to a total of 16 devices - including instruments, controllers, and peripherals - can be connected to the IEEE-488 bus at any one time. This restriction to 16 devices is a consequence of the limited driving capability of the bus. It is defined at ≤ 48 mA. The current consumption of each device driver is ≤ 3 mA (15 devices + 1 controller with 3 mA each = 48 mA). The IEEE-488 bus uses negative logic with standard TTL logic levels between 0V and 5V. Logic 0 is a -TTL high level (≥ 2.0V), and logic 1 is a TTL low level ≤ 0.8V).

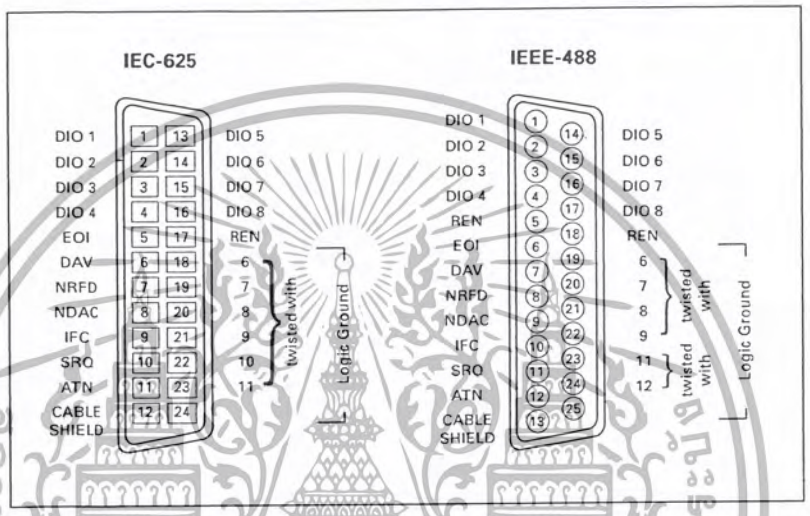


Figure 1: IEEE-488 bus connector configuration

Attention: The 25-pin connector specified by the IEC-625 standard is easy to confuse with the 25-pin connector normally used for RS-232C-serial interfaces. Use of the wrong connector can result in serious damage to the interface electronics!

Figure 2: Shielded IEEE-488 round multiconductor

No other interface for communications with peripheral devices is so precisely defined as the IEEE488 bus, from the bus protocol all the way to connector pin assignments. The only limitations for connection of devices are that one device should be added for every two meters of cable (although they need not be equally spaced along the length

of the cable), and the maximum cable length for any single run is limited to 20 meters. The devices are connected in parallel, with a cable assembly consisting of cables with both plug and receptacle connectors at each end, ideally as „piggy-back" connectors, i.e. with male and female connectors molded one above the other, the latter taking the plug connector of the next cable **(Figure 2)**. This design allows devices to be linked in either a daisy-chain or star configuration, or a combination of the two.

Function of the bus lines

Data transfer is performed by consecutive transmission of 8-bit bytes, each of which is sent in parallel over the data lines DIO1 through DIO8. The handshake lines designated DAV, NRFD and NDAC are used to control data transfer. The 5 interface management lines ATN, IFC, REN, SRQ and EOI are separate from the handshake lines, and are used by the System Controller to maintain order and initiate proceedings within the bus. 6 of the 7 ground return lines are twisted together with the lines DAV, NRFD, NDAC, ATN, SRQ and IFC (EOI in the IEC-625 bus). The shield drain line SHIELD is connected to ground.

Data lines DIO1 through DIO8

Each data line (DIG stands for „DATA IN OUT") is used for transfer of one bit of each 8-bit data word. Most data use the 7bit ASCII or ISO code set. Each letter or character is represented as a 7-bit word **(Figure 4)**.The 8th bit (DIO8) is either unused or is used for parity.

Binary, decimal, octal and hexadecimal coding systems are also used in practice for representing bit combinations. The ASCII code is now in general use by all manufacturers of test and computer equipment; consequently, the IEEE-488 and computers utilize the same character set. As a rule. It is therefore no longer necessary to convert character sets between computers and the IEEE-488 bus.

The handshake lines DAV, NRFD and NDAC

The handshake lines control the transfer of message bytes among devices over the data lines according to a scheme defined in the standard **(Figure 3)**.

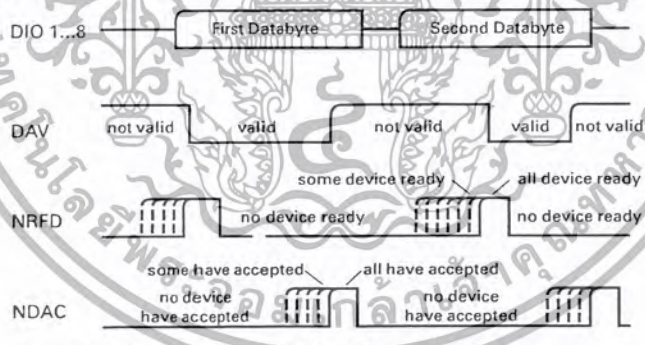


Figure 3: Handshake lines DAV, NRFD and NDAC

The user of an IEEE-488 bus-controlled system normally has no way of influencing the sequence of signal sent over the handshake lines. Even when this is possible, a trough understanding of the workings of the IEEE-488 bus is essential. The following explanations of the DAV, NRFD and NDAC lines are provided by way of information; they are not, however ,required for a general understandinf of the functions of the IEEE-488 bus.

DAV (data valid)
A device with data or commands to send first checks the NRFD line to verify whether all devices are ready to receive (NRFD must be in its logic 0 state). It then transfers

the data to the bus and uses the DAV line to indicate that the signals on the data lines are stable (valid) and can be safely accepted (DAV at logic 1). It then checks the NDAC line to determine whether the data have been accepted by all devices (logic 0), resets DAV to a logic 0 state (= not valid), and outputs a new message byte to the bus.

NRFD (not ready for data)
This- line is driven by all devices when receiving commands and by Listeners when receiving data messages. It indicates when a device is ready or not ready to receive a message byte. A logic 0 denotes readiness to receive messages. As soon as valid data are signalized by the DAV line, the receiving device puts the NDAC into its logic 0 state and strobes in the data. NDAC goes back to logic 1 when the device sending data indicates „data not valid" on the DAV line, and so on. Even if only one device is sending a logic 1 signal on the NRFD line, the DAV line cannot be pulled low (logic 1)to prompt acceptance of new data.

MDAC (not data accepted)
This line is driven by all devices when receiving commands and by Listeners when receiving data messages. It is used to communicate to devices with data or commands to send that the data on the signal lines have been accepted (NDAC logic 0). The device that is sending the message byte cannot put the DAV back into its logic 0 state (= data not valid) until this is done. After the DAV line carries the signal denoting that the data states are no longer valid, the NDAC goes into a logic 1 state again, and the device is ready for the next message byte. Even if just one device is transmitting a logic 1 on the NDAC line, the data on the signal lines cannot be taken off the bus.

The interface management lines ATN, IFC, REN, SRQ and EOI

The interface management lines are used to manage the flow of information across the IEEE-488 bus. They can be influenced by the user. ATN, IFC and REN can only be driven by the System Controller. SRQ can only be driven by a peripheral device. EOI can be driven by either the CIC or a peripheral device acting as Talker.

ATN (attention)
ATN can only be driven by the Controller-in-Charge. The state of this line indicates to the devices connected to the bus whether the states on the data lines should be interpreted as commands (logic 1) or device-dependent messages (logic 0). Acceptance of the data is controlled by the handshake lines DAV, NRFD and NDAC.

IFC (interface clear)
IFC can likewise only be driven by the System Controller. It is used to initialize the bus interfaces of all devices connected to the IEEE-488 bus (returning them to the same state as when they a re powered up p). IFC should be the first instruction issued when control of a device via the bus is begun;

REM (remote enable)
The System Controller drives the REN line. If REN is pulled to a-logic 1, this places all devices enables as Listeners in remote control mode. A logic 0 puts all devices back into local (manual) mode. The REN line can also be set globally to logic 1 by connecting it with the IEEE-488 bus ground, or within devices on the bus by fixed wiring. As long as REN stays at logic 0, none of the Listeners on the bus - unless they are internally wired for a logic 1 - will accept remote commands!

SRQ (service request)
Any device connected to the IEEE-488 bus can drive the SRQ line to request service from the System Controller, for instance if an abnormal condition exists or in order to transfer measurement data to the System Controller. The System Controller can only influence the SRQ line by indirect means, namely by performing a serial poll until it finds the device which is the perpetrator of the SRO interrupt. That device is then automatically cleared of SRQ data, thus resetting the SRQ line to logic 0.

EOI (end or identify)
The EOI line serves two purposes. A Talker (which can be the CIC) uses it to mark the end of a message string. When a Listener detects EOI. it terminates the conversation

and dumps any data bytes that follows. The EOI line is also used by the System Controller to perform a parallel poll of up to 8 different instruments on the bus. This involves setting both EOI and ATN to their logic 1 states.

IEEE-488 bus communications

Both uniline and multiline messages can be sent over the bus. The interface management lines ATN, IFC, REN. SRO and EOI are used uniline messages, and the data lines D101 through D108 areused in conjunction with the ATN line and the EOI line for multiline messages. The ATN line is used to distinguish between device-dependent messages and interface messages.

Uniline messages
These messages have the highest:priority, and are detected by devices on the bus regardless of the states on the data lines. They are sent using the following signal lines:

Line	Function
DAV	Handshake: data valid
NRFD	Handshake: not ready for data
NDAC	Handshake: data not yet accepted (notdataaccepted)
IFC	Initialize interface (interface clear)
REN	Remote/manual selection (remote enable)
SRQ	Interrupt service request (service request)
ATN	Device-dependent message/interface message (attention)
EOI	End of message (end or identify)
EOI	with Parallel poll (status byte query) ATN

Multiline messages
This category comprises both device-dependent messages and interface messages. Device-dependent messages. The ATN line is at logic 0 during transfer of this type of message. The data are transferred with aid of the handshake lines DAV. NRFD and NDAC. The end of a transmission is identified by a line feed and/or asserting EOI. Device-dependent messages contain device-specific information such as programming instructions, measurement results, machine status, and data files. Coded in the format stipulated by the instrument maker.

Interface messages
The ATN line is at logic 1 during transfer of this type of message. These messages contain commands issued by the System Controller to maintain order and initiate proceedings within the bus system, and are sent with the aid of the handshake lines DAV, NRFD and NDAC. If ATN is asserted, this tells the receiving devices that interface message units are being sent over the bus.

Interface messages are used for:
- Enabling the Talker function of devices (TAG) by sending their primary talker addresses, and disabling them.
- Enabling the Listener function of devices (LAG) by sending their primary listener addresses, and disabling them.
- Transfer of the addressed commands GTL, SDC, PPC, GET and TCT.
- Transfer of the unaddressed (universal) commands LLO, DCL, PPU, SPE and SPD.
- Transfer of secondary addresses (SCG: Secondary Command Group).

The ASCII character table **(Figure 3)** also contains the interface messages that can be sent while ATN is asserted.

These are summarized in the following:

Function	Description
Talk and Listen addresses	
These are used to address instruments as Talkers or Listeners.	
TAG	For addressing an Talker Address Group instrument as aTalker
LAG	For addressing an Listener Address Group instrument as a Listener

Addressed messages
These commands are received by only a single device, which is addressed.

GTL Go To Local	Switches to local (manual) operation
SDC Selected Device Clear	Initializes an instrument
PPC Parallel Poll Configure	Transmits the parallel poll byte (followed by PPE)
GET Group Execute Trigger	Activates a device function
TCT Take Control	Passes control to another controller

Universal (unaddressed) messages
These commands are received by all devices on the bus that have Listener capability.

LLO Local Lockout	Prevents manual operation
DCL Device Clear	Initializes instruments
PPU Parallel Poll Unconfig.	Removes parallel pollstatus
SPE Serial Poll Enable	Initiates serial poll
SPD Serial Poll Disable	Terminates serial poll

Secondary commands
These are used to transfer the parallel poll byte following PPC, for clearing the parallel poll status bits returned by polled instruments, and for transmitting secondary addresses following the talk or listen primary address of a device.

PPE Parallel Poll Enable	Defines parallel poll byte
PPD Parallel Poll Disable	Clears parallel poll byte
SCG Sec. Comm. Group	For transfer of a secondary address after TAG or LAG

The IEEE-488 bus interface H088

The test instruments of the HAMEG Series HM8100 have all been designed to permit their use in automated testing environments. In order to connect them to an IEEE488 bus, the interface HO88 (option) is required. Series HM8100 instruments equipped with the IEEE488 bus interface comply with the stipulations of the IEC-625-1 and IEEE-488 standards. If the HO88 interface is ordered together with the a HM 81 .. instrument, it is installed within it at the factory. The HO 88 interface is also available as a separate option for retrofitting purposes at a later time.

Software service

For operation of the Series 8100 instruments, HAMEG will be issuing software at irregular intervals. This software will be supplied free of charge-to the owners of these instruments. In addition, updates of the firmware contained in the instruments will be provided in the form of new EPROMs at cost price. In order to- benefit from this software service, it is sufficient to register by sending in your name and address and the serial number of-your HAMEG instrument.

Installation

An HM 81 .. instrument can be easily and unproblematically retrofitted by the user with the H088 Interface. The first step is to remove the instrument enclosure. To do

so, unscrew the 6 screws on the rear panel of the instrument and take off the plastic back cover; the enclosure can then be pulled off towards the back. Looking at the instrument from the back, the interface card is installed in the upper right corner of the instrument, with its component side facing down. The interface card is attached using the 3 supplied self-tapping screws.

First, however, the two cables for power supply and data transfer are inserted into the corresponding flat socket connectors on the circuit board of the instrument. The short cable is the power supply for the interface. The longer cable connects the interface card with the electrically isolated serial interface of the instrument itself. The enclosure is then slid back on. When doing so make sure that the edges of the metal enclosure slide exactly into the grooves of the plastic covers on the front and back. After the rear plastic cover has been replaced and fastened, the unit is again ready for operation.

Specifications

Connector:	Standard 24-pin IEEE-488 connector: Amphenol Series 57 MICRORIBBON
Output:	Open collector
Output voltage levels:	High: 2.5V
Low:	0.4 V at 48 mA
Input voltage levels:	Typical hysteresis: 0.8V
Input, high:	2.0V
Input, low:	0.6V
Terminations:	3.3 kΩ + 5% (+5 V)
	6.2 kΩ +5% (ground)
Capacitance :	100pF
Supply voltage:	9.36 V AC
Current consumption:	250 mA

All data and signal lines are electrically isolated from ground (even after installation in a Series HM8100 instrument)! When the IEEE-488 cable is unconnected-there is no electrically conductive path to the instrument chassis or the grounded wire of the power supply!

Address selection

All instruments connected to an IEEE488 bus must receive unique device addresses. This is done using the 5 DIP switches to the left of the IEEE488 bus connector on the instruments rear panel. The switches are binary-coded. All addresses are allowed except decimal 31 (binary 11111). If the switches are set to 11111, this has the effect of defining Talk-only mode. This mode is selected if it is wished for measurement data to be output directly, i.e. without the use of a controller. It may be wished, for example, to directly connect an instrument (Talk-only) with a printer (listen-only). The printer then continuously prints out the measurement data received from the counter. The Talk-only mode must never be used in bus configurations incorporating a Controller, since the instrument is then unable to „listen to" the Controller and consequently cannot be given instructions. If an instrument is accidentally set to Talk-only mode it will monopolize the bus and interfere with commands issued by the Controller and/or data on the bus coming from other instruments.

Interface attributes

Not all messages must be decoded by every device. Interface messages are only decoded If an instrument is properly equipped to do so. Devices that only have Listener capabilities, for example, do not decode Talker addresses. So-called subset ratings are used to indicate the capabilities of a given instrument for decoding interface messages. The IEEE488 standard divides the interface into twelve basic functions, each of which has a set of options that can be used to implement different subsets of these capabilities. For example, the specifications for an IEEE-488 interfaceable device might list the subset functions it supports as SH7, AM1, T5, L3, RL1, DC1, SR1 and C0. The letter or letters stand for the basic interface functions, and the following digit for the subset. In each case, a zero indicates that the corresponding capability is not given.

Here is a list of the basic interface functions and their subsets:

Function	Subset	designations
Service Request Whether or not an instrument is allowed to request service from the controller with the SRQ line)	SR	SR0,1
Remote-Local (Switching capabilities between manual (local) control and programmable (remote) operation)	RL	RL1,...,2
Parallel Poll	PP	PP0,...,2
Device Clear	DC	DC0,...,2
Device Trigger (Whether or not an instrument or group of instruments can be triggered or some action started upon receipt of the group executive trigger (GET) message)	DT	DT0,1
Source Handshake (Whether or not a device is allowed to generate the handshake cycle fortransmitting data)	SH	SH0,1
Acceptor Handshake (Whether or not a device is allowed to generate the handshakecycle for receiving data)	AH	AH0,1
Talker (Capabilities for transmission of data)	T	T0,...,1
Extended Talker (Like Talker, but using secondary addresses as well)	TE	TE0,...,8
Listener (Capabilities for receipt of data)	L	L0,...,4
Extended Listener (Like Listener, but using secondary addresses as well)	LE	LE0,...,4

The HM81 .. is equipped with the following functions:

Function	Code	Description
Source Handshake	SH1	Full capability
Acceptor Handshake	AH1	Full capability
Control function	C0	Cannot function as Controller over other devices
Talker	T5	Full capability
Listener	L4	Full capability (except Listen-only mode)
Service Request	SR1	Full capability
Remote-Local	RL1	Fullcapability
Parallel Poll	PP0	No capability
Device Clear	DC1	Full capability
DeviceTrigger	DT1	Full capability

Talker T5:
- BasicTalker
- Talk Only
- Serialpoll
- „Unaddressed if My Listen Address" (prevents instrument from being a talker and a listener at the same time)

Listener L4:
- Basic Listener
- „Unaddressed if My Talk Address" (prevents instrument from being a talker and a listener at the same time)

Source Handshake and Acceptor Handshake (SH1, AH1)

SH1 and AH1 denote, quite simply, that the HM81.. is able to generate the handshake cycle (using lines DAV, N RFD and NDAC) to exchange data with other instruments and/or a Controller.

Talker(T5)
The HM81.. can transmit measurements to other devices or the Controller (it „talks").
T5 also means that the instrument can reply to a serial poll by the Controller by transmitting a status byte.

Listener (L4)
The HM81.. can receive instructions from a Controller on the bus (it „listens").
Service Request (SR1) The HM81.. can interrupt the Controller, e.g. after each measurement, to indicate that it has additional data to transmit.
Remote-Local (RL1) The HM81.. can be operated either manually (local control) or by remote control (programmable mode).

Device Clear (DC1)
The HM81.. can be initialized to a predefined cleared state, either selectively or together with all other instruments on the bus.

Device Trigger (DT1)
A new measurement can be triggered by remote control.

Command codes
For device-cfependent messages that are understood by the HM 81 .. instruments see manuals of the HM8100 series.

Data format
Separators in the data field

Within a data transmission, the following characters can be used as separators:

Semicolon (;) or (3Bh)
Comma (,) or (2Ch)
Space ( ) or (20h)

These characters can also be used for multiline command codes within a data transfer. Two or more device-dependent or interface commands can be contained within a given data string; these are then executed sequentially. The only requirement is that they be separated by the characters listed above.

Input and output delimitation

The standardly accepted means of identifying the end of a message on the IEEE-488 bus is by a carriage return (CR) or (0Dh), with or without EOI.
However, any other character will also be accepted as a terminator, provided that it is send together with EOI. The last character of a message is marked by a carriage return (CR) or (0Dh) together with EOI.

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.

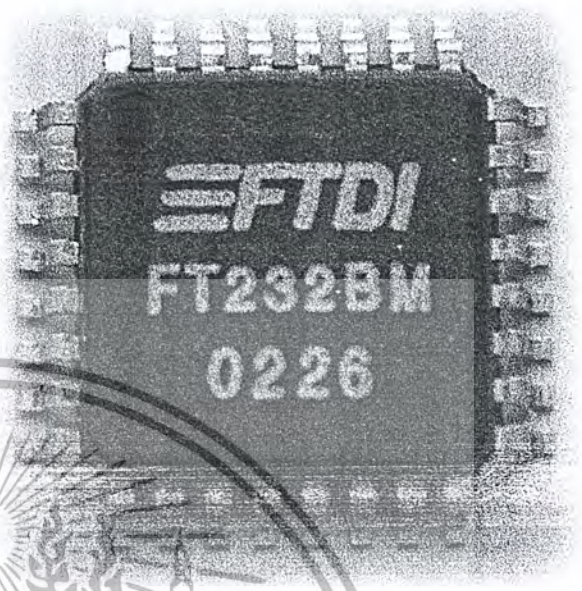
- **Improved PreScaler Granularity**
  The previous version of the Prescaler supported division by (n + 0), (n + 0.125), (n + 0.25) and (n + 0.5) where n is an integer between 2 and 16,384 (2^{14}). To this we have added (n + 0.375), (n + 0.625), (n + 0.75) and (n+ 0.875) which can be used to improve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

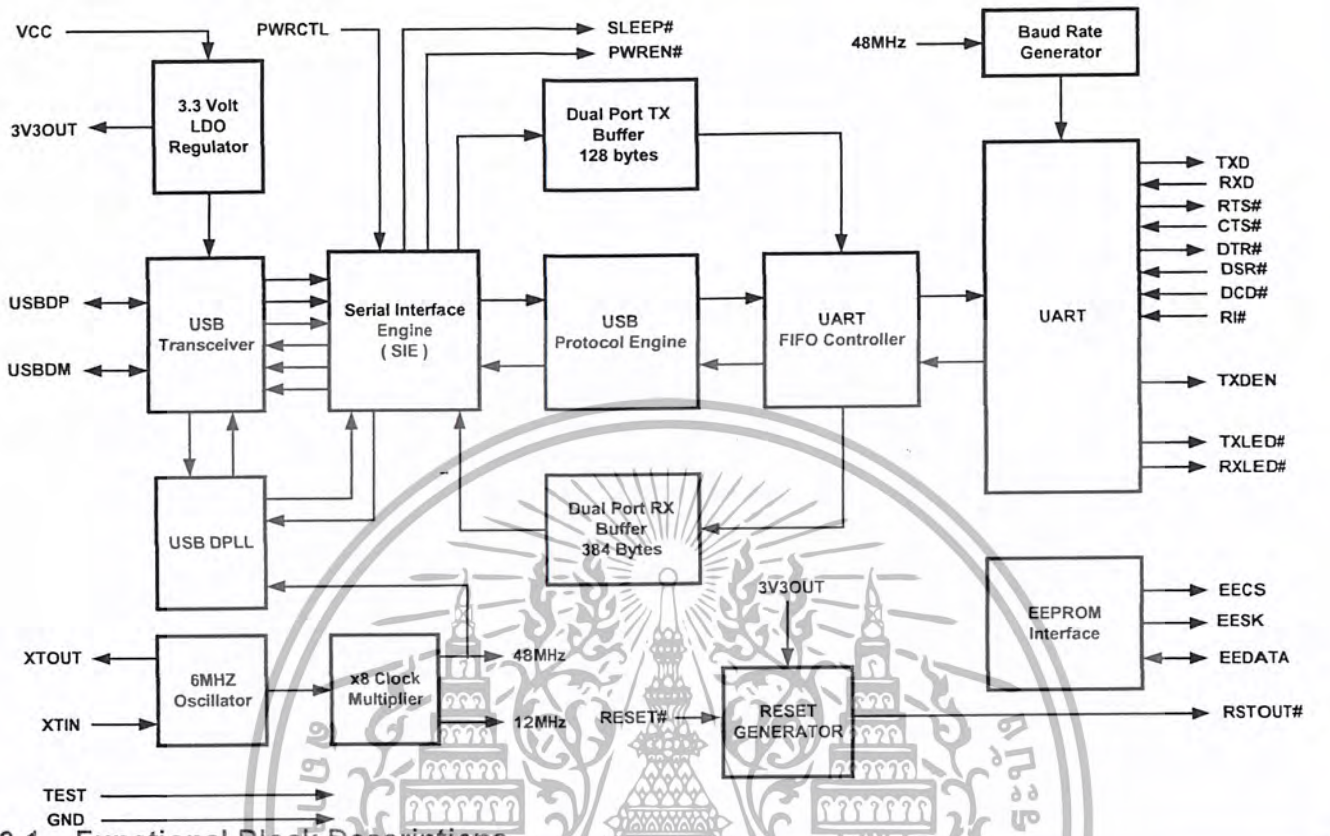
reset is controlled by the IO port of a MCU, FPGA or ASIC device) an external transistor circuit is no longer required as the 1.5k pull-up resistor on USB DP can be wired to the RSTOUT# pin instead of to 3.3V. Note : RSTOUT# drives out at 3.3V level, not at 5V VCC level. This is the preferred configuration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

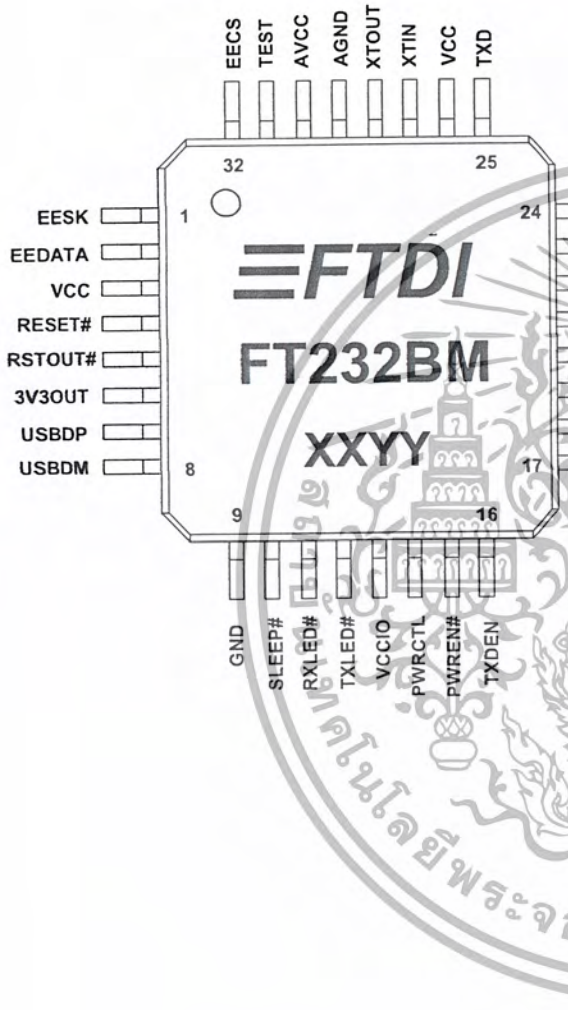
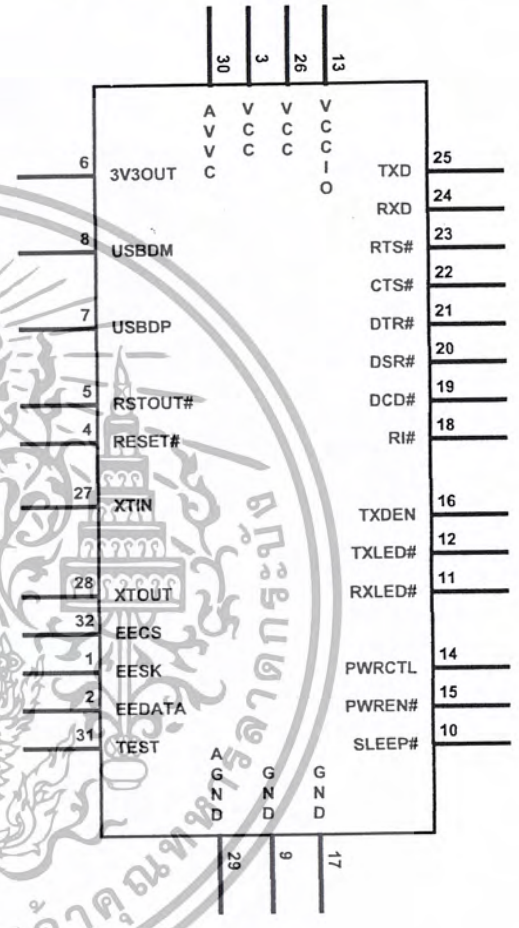


Figure 2
Pin-Out
(Schematic Symbol)



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.

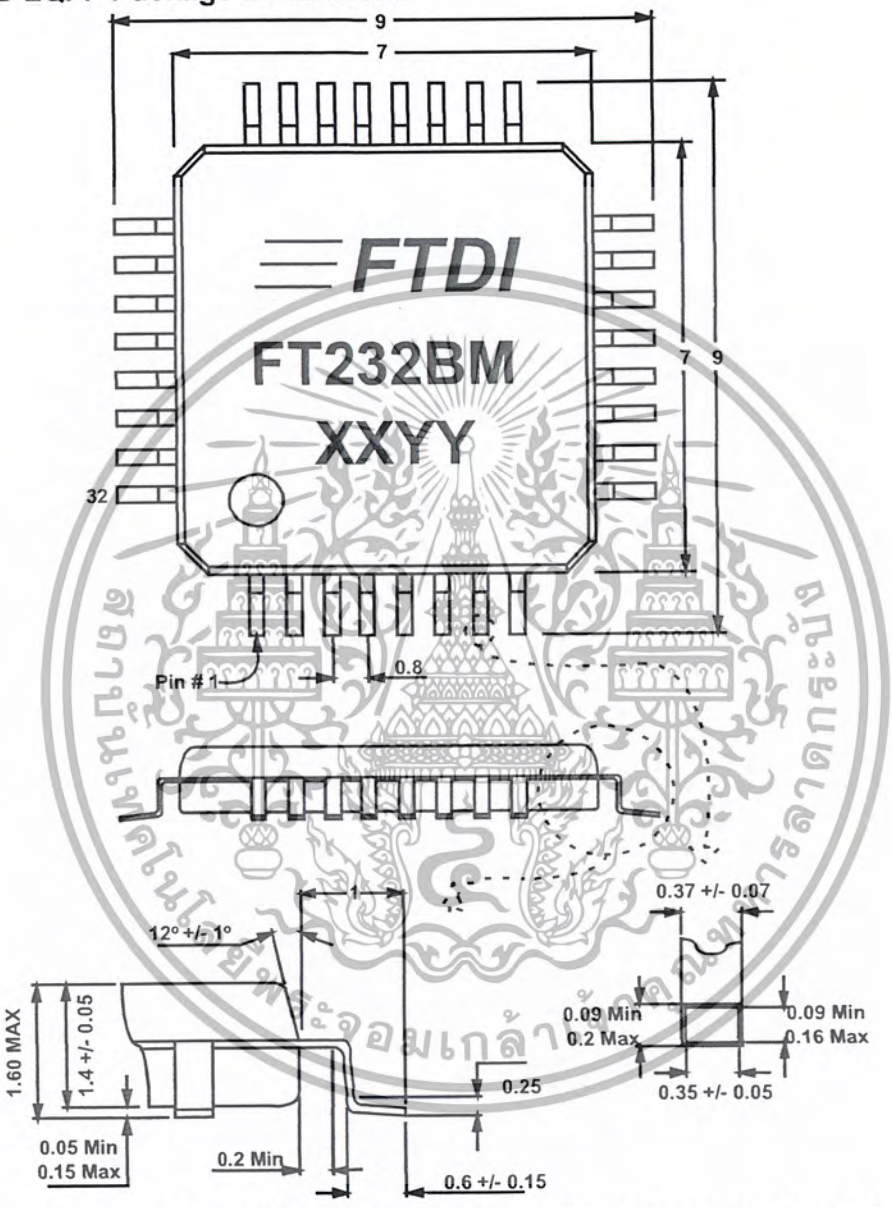
POWER AND GND GROUP

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

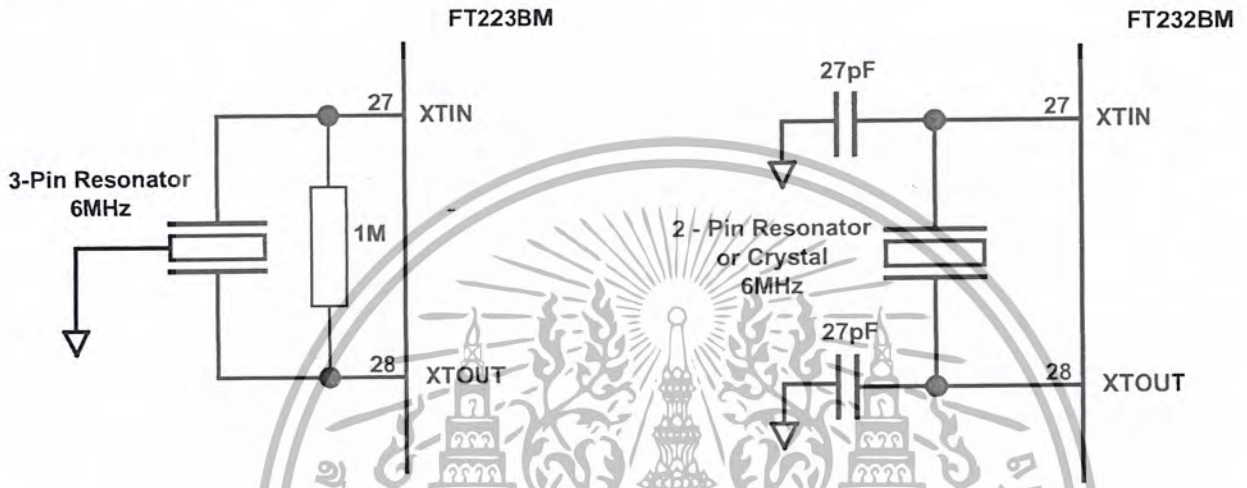


Figure 4 illustrates how to use the FT232BM with a 3-Pin Ceramic Resonator. A suitable part would be a ceramic resonator from Murata's CERALOCK range. (Murata Part Number CSTCR6M00G15), or equivalent. 3-Pin ceramic resonators have the load capacitors built into the resonator so no external loading capacitors 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

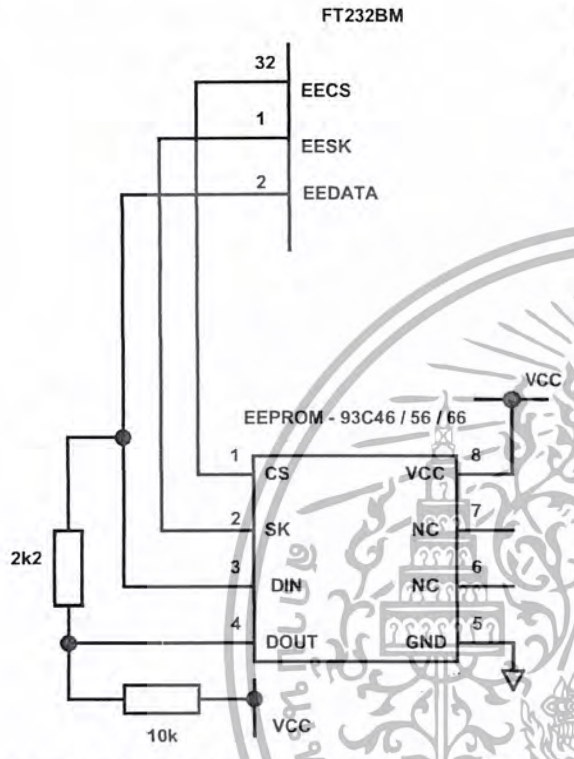


Figure 6 illustrates how to connect the FT232BM to the 93C46 (93C56 or 93C66) EEPROM. EECS (pin 32) is directly connected to the chip select (CS) pin of the EEPROM. EESK (pin 1) is directly connected to the clock (SK) pin of the EEPROM. EEDATA (pin 2) is directly connected to the Data In (Din) pin of 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

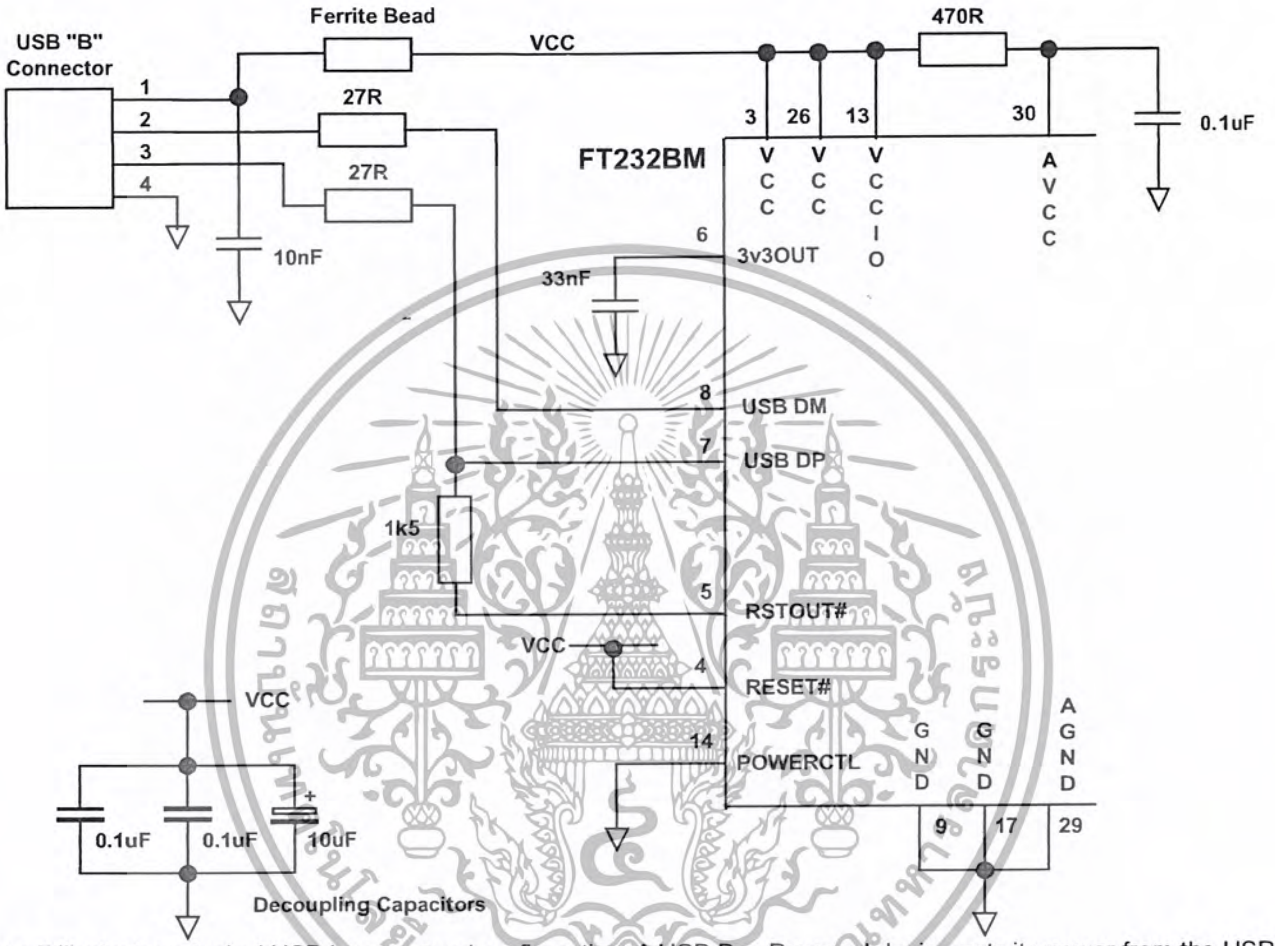


Figure 7 illustrates a typical USB bus powered configuration. A USB Bus Powered device gets its power from the USB bus. Basic rules for USB Bus power devices are as follows –

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
USB Self Powered Configuration

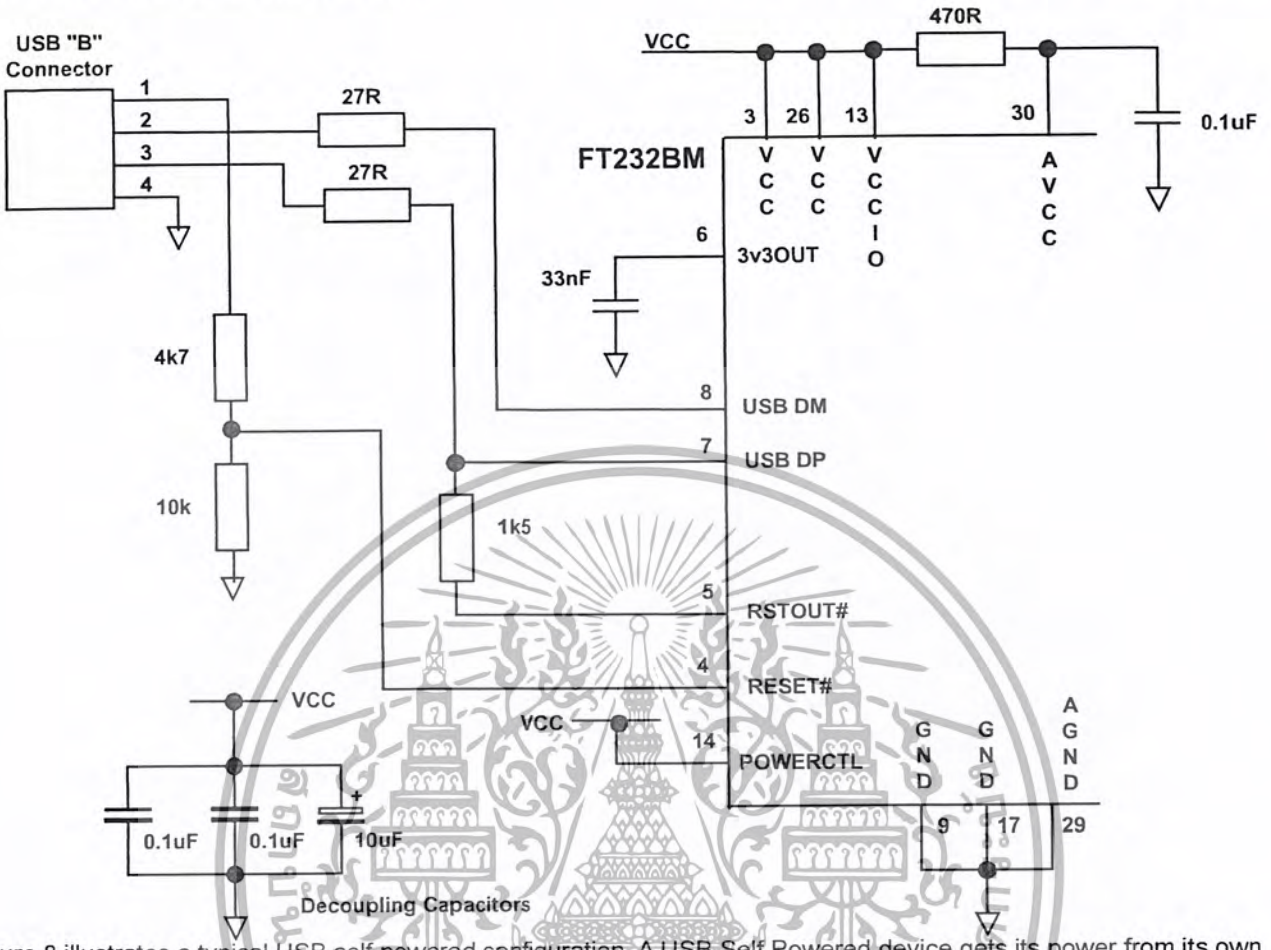


Figure 8 illustrates a typical USB self powered configuration. A USB Self Powered device gets its power from its own POWER SUPPLY and does not draw current from the USB bus. The basic rules for USB Self power devices are as follows –

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

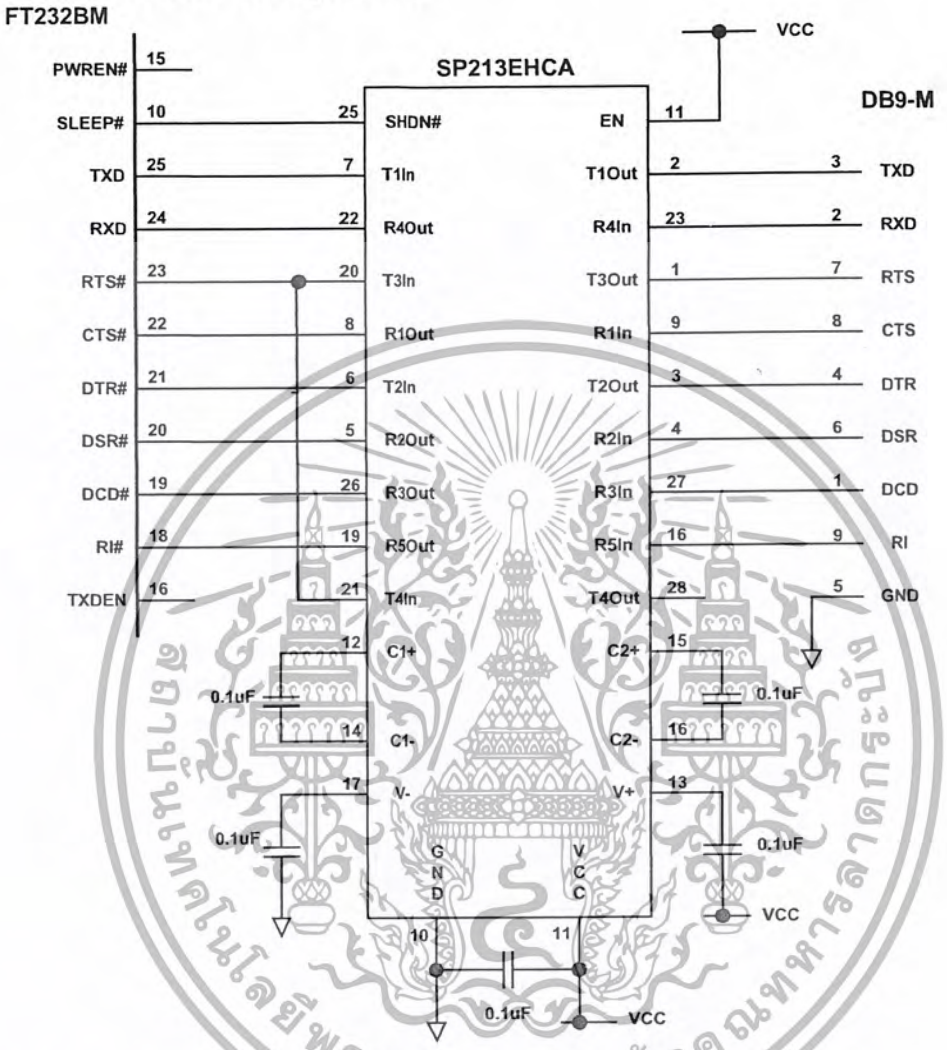


Figure 9 illustrates how to connect the UART interface of the FT232BM to a TTL – RS232 Level Converter I.C. to make a USB <=> RS232 converter using the popular "213" series of TTL to RS232 level converters. These devices have 4 transmitters and 5 receivers in a 28 LD SSOP package and feature an in-built 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

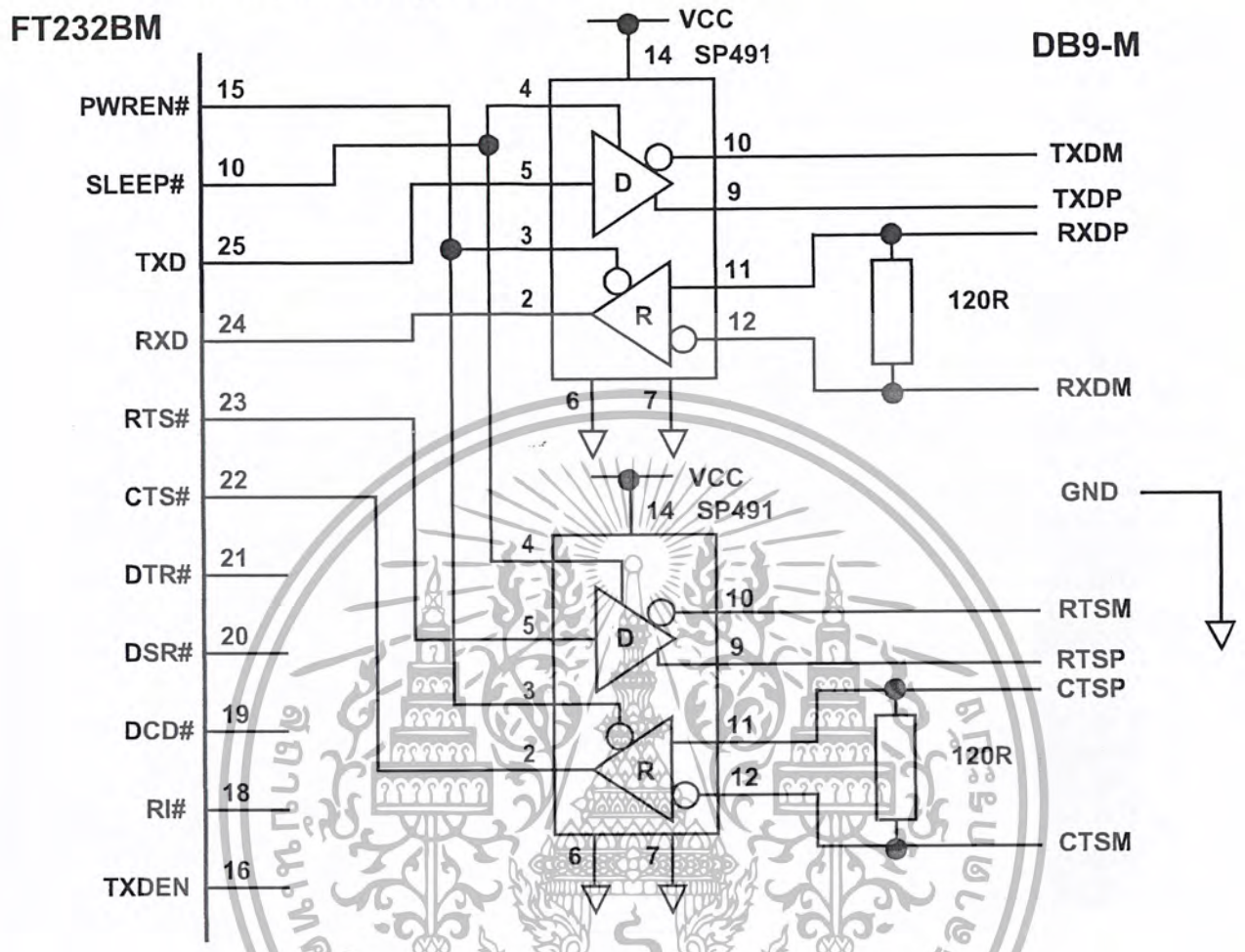


Figure 10 illustrates how to connect the UART interface of the FT232BM to a TTL – RS422 Level Converter I.C. to make a USB <=> RS422 converter. There are many such level converter devices available – this example uses Sipex SP491 devices which have enables on both the transmitter and receiver. Because 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

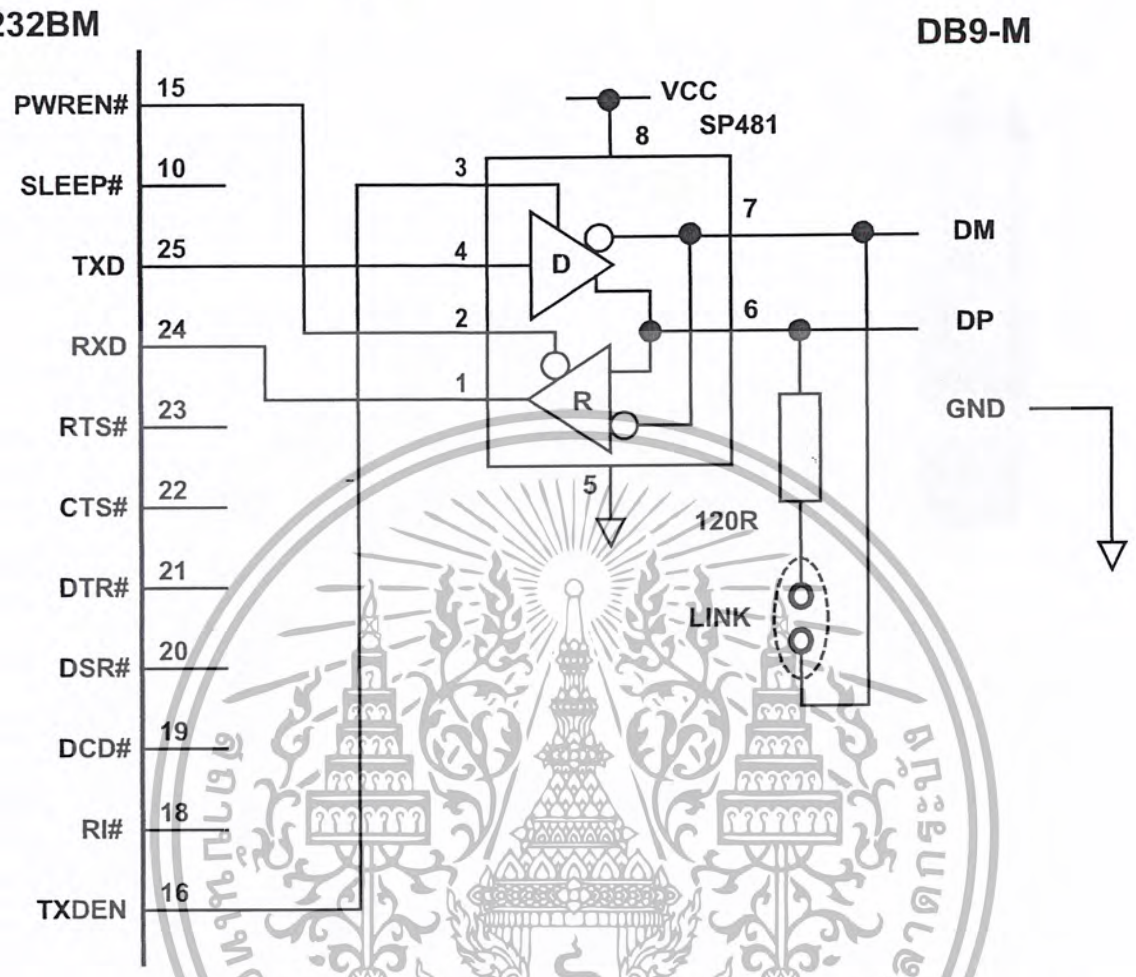


Figure 11 illustrates how to connect the UART interface of the FT232BM to a TTL – RS485 Level Converter I.C. to make a USB => RS485 converter. This example uses the Sipex SP491 device but there are similar parts available from Maxim and Analog Devices amongst others. The SP491 is a RS485 device in a 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

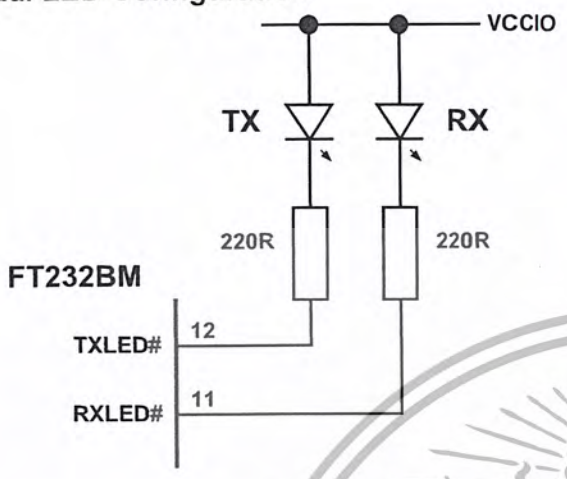
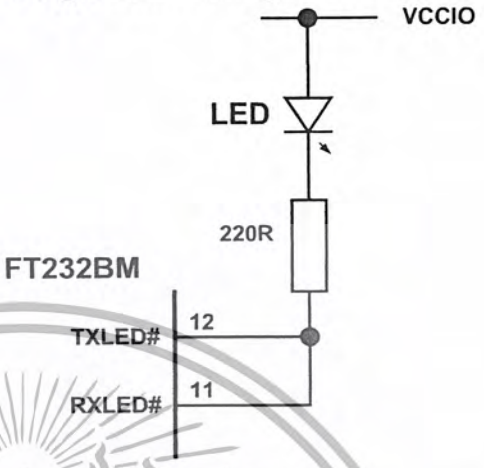


Figure 13
Single LED Configuration



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.

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

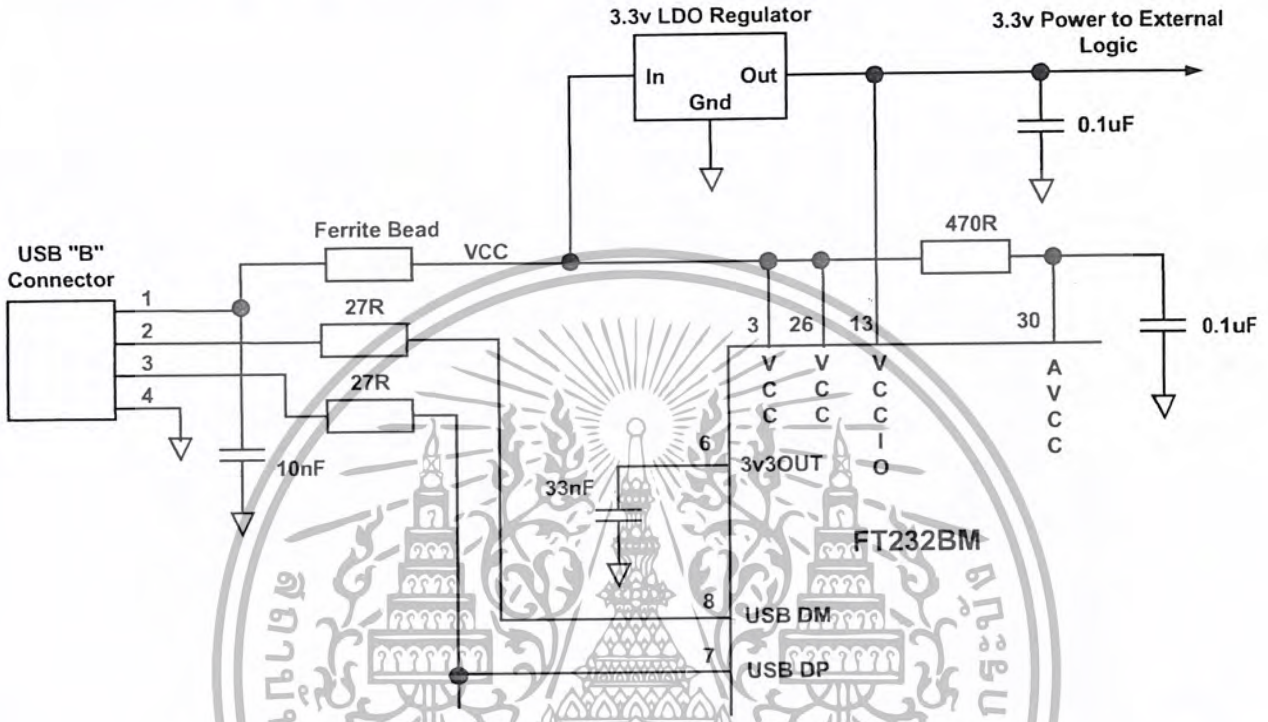


Figure 14 shows how to configure the FT232BM to interface with a 3.3V logic device. In this example, a discrete 3.3V regulator is used to supply the 3.3V logic from the USB supply. VCCIO is connected to the output of the 3.3V regulator, which in turn will cause the UART interface IO pins to drive out 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

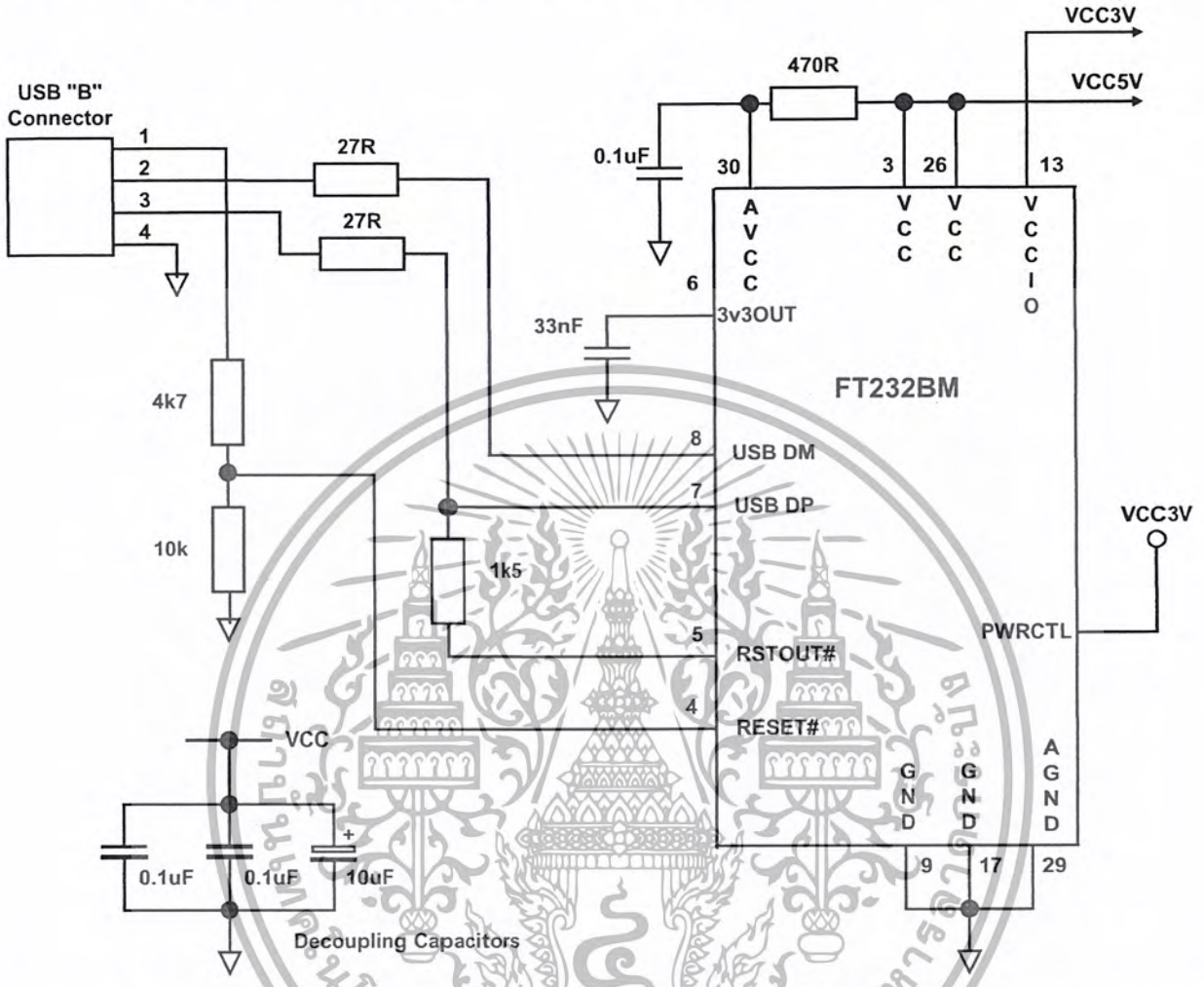


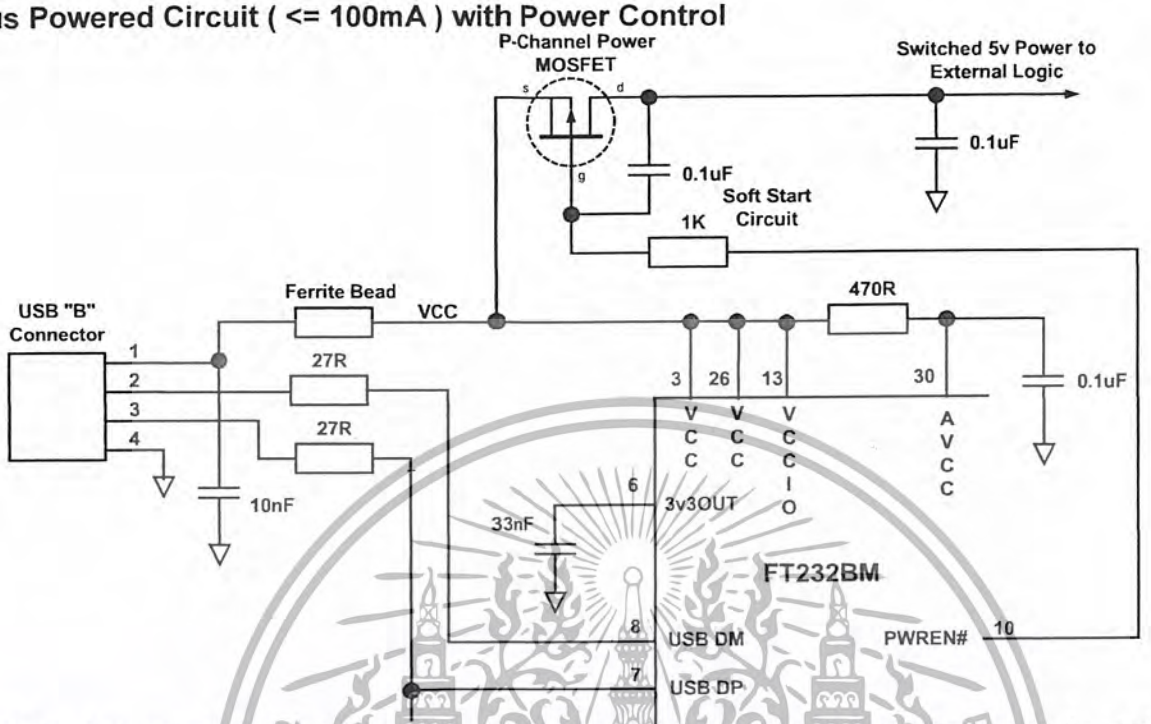
Figure 15 is an example of a USB self powered design with 3.3V interface. In this case VCCIO is supplied by an external 3.3V supply in order to make the device IO pins drive out at 3.3V logic level, thus allowing it to be connected to a 3.3V MCU or other external logic. A USB self powered design uses 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.

8.0 Document Revision History

DS232B Version 1.0 – Initial document created 30 April 2002.

DS232B Version 1.1 – Updated 04 August 2002

- Section 4.1 RESET# Pin description corrected (RESET# does not have an internal 200k pull-up to VCC as previously stated).
- Figure 2 pin-out corrected (EECS = Pin 32).

DS232B Version 1.2 – Updated 27 October 2002

- Pin and package naming made consistent throughout data sheet.
- Section 1.0 Updated to reflect availability of Mac OS X driver.
- Section 2.0 Minor corrections.
- Section 3.1 Minor changes to functional block descriptions of SIE, RESET Generator, and EEPROM interface.
- Section 4.1 Note added to EEPROM interface group.
- Section 4.1 RSTOUT# Pin description amended.
- Section 6.1 Minimum operating supply voltage adjusted.
- Section 6.1 EESK added to Note 3.
- Section 6.1 UART IO pin characteristics amended.
- Section 6.1 RESET#, TEST, EECS, EESK, and EEDATA pin characteristics amended.
- Section 6.1 RSTOUT pin characteristics amended.
- Section 7.1 Updated recommended ceramic resonator part number and circuit configuration.
- Section 7.3 "USB Self Powered Configuration (1)" (original figure 8) removed. Recommended circuit for USB self powered designs updated. Subsequent figure numbers have changed as a result.
- Section 7.6 Note added to description of Bus powered circuit with 3.3V logic drive / supply voltage.
- Section 7.6 Self Powered Circuit with 3.3V logic drive / supply voltage added (new figure 16).

9.0 Disclaimer

10.0 Contact Information

Future Technology Devices Intl. Limited
St. George's Studios
93/97 St. George's Road,
Glasgow G3 6JA,
United Kingdom.
Tel : +44 (0)141 353 2565
Fax : +44 (0)141 353 2656

E-Mail (Sales) : sales@ftdichip.com
E-Mail (Support) : support@ftdichip.com
E-Mail (General Enquiries) : admin@ftdichip.com
Web Site URL : http://www.ftdichip.com

Agents and Sales Representatives

At the time of writing our Sales Network covers over 40 different countries world-wide. Please visit the Sales Network page of our Web site for the contact details our distributor(s) in your country.