จี พี ไอ บี การ์ดอินเตอร์เฟส

GPIB CARD INTERFACE

โดย

นายอนันต์ศักดิ์ แสงจันทร์

นายอนุกูล ชูทอง

นายอำนวย หน่อแดง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

จี พี ไอ บี การ์ดอินเตอร์เฟส

GPIB CARD INTERFACE

โดย

นายอนันต์ศักดิ์	แสงจันทร์	รหัส 38013218
นายอนุกูล	ชูทอง	รหัส 38013259
นายอำนวย	หน่อแดง	รหัส 38013262

อาจารย์ที่ปรึกษา

อาจารย์ ขนิษฐา แซ่ตั้ง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์เรื่อง จี พี ไอ บี การ์ดอินเตอร์เฟส

GPIB CARD INTERFACE

จัดทำโดย นายอนันต์ศักดิ์ แสงจันทร์

นายอนุกูล ชูทอง

นายอำนวย หน่อแดง

ปริญญานิพนธ์นี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....อาจารย์ที่ปรึกษา

(อ. ขนิษฐา แซ่ตั้ง)

วันที่...../...../.....

กิตติกรรมประกาศ

รายงานฉบับนี้สามารถที่จะทำสำเร็จลุล่วงไปได้ด้วยดี ต้องขอขอบพระคุณหลาย ๆ ฝ่ายที่คอยให้ความช่วยเหลือตลอดมาโดยเฉพาะอย่างยิ่งพระคุณของบิดามารดา ผู้ซึ่งคอยให้โอกาสและให้การสนับสนุน คณะผู้จัดทำตลอดมา และที่ขาดไม่ได้ต้องขอกราบขอบพระคุณท่านอาจารย์ที่ปรึกษา อาจารย์ขนิษฐา แซ่ตั้ง ที่คอยให้คำปรึกษา และคำแนะนำต่าง ๆ ตลอดจนอุปกรณ์เครื่องมือต่าง ๆ ที่ใช้ในการดำเนินงาน ทางคณะผู้จัดทำจึงขอกราบขอบพระคุณไว้ ณ โอกาสนี้ด้วย

นายอนันต์ศักดิ์ แสงจันทร์
นายอนุกูล ชูทอง
นายอำนวย หน่อแดง
คณะผู้จัดทำ

จี พี ไอ บี การ์ดอินเตอร์เฟส

นายอนันต์ศักดิ์ แสงจันทร์
นายอนุกูล ชูทอง
นายอำนวย หน่อแดง
อ. ขนิษฐา แซ่ตั้ง (อาจารย์ที่ปรึกษา)
ภาคการศึกษาที่ 2 ปีการศึกษา 2540

บทคัดย่อ

โครงงานนี้เป็นการอินเตอร์เฟสระหว่าง คอมพิวเตอร์กับแหล่งจ่ายไฟกระแสตรง โดยผ่านระบบมาตรฐาน GPIB IEEE - 488 (General Purpose Interface Bus) เป็นการใช้คอมพิวเตอร์ควบคุมแหล่งจ่ายไฟกระแสตรงให้สามารถจ่ายแรงดันได้ตามต้องการ ซึ่งในแหล่งจ่ายไฟกระแสตรงนี้จะมีชุดควบคุมการทำงานโดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม โดยไมโครคอนโทรลเลอร์จะทำหน้าที่ ควบคุมการจ่ายแรงดันโดยใช้สัญญาณข้อมูล 8 บิท ผ่านวงจรแปลงสัญญาณดิจิตอลเป็นอะนาล็อก เพื่อใช้ป้อนเป็นสัญญาณในการเปรียบเทียบอ้างอิงให้กับวงจรของแหล่งจายไฟกระแสตรง และยังควบคุมการจัดระบบและตรวจสอบของสัญญาณให้เข้ากับมาตรฐานของ IEEE-488 เพื่อที่จะทำให้แหล่งจ่ายไฟกระแสตรง ซึ่งทำหน้าที่เป็นตัวรับ (Listener) สามารถติดต่อกับคอมพิวเตอร์ซึ่งทำหน้าที่เป็นตัวควบคุม (Controller)ได้ ซึ่งสัญญาณที่ส่งมาจากคอมพิวเตอร์ถูกตรวจสอบโดยไมโครคอนโทรลเลอร์ สัญญาณที่ได้จากการตรวจสอบจะถูกทำการเปรียบเทียบกับคำสั่งที่อยู่ในหน่วยความจำของไมโครคอนโทรลเลอร์ แล้วไมโครคอนโทรลเลอร์ก็จะนำคำสั่งที่ได้มาทำการสั่งให้แหล่งจ่ายไฟปฏิบัติตาม

GPIB CARD INTERFACE

Mr. Anansak Sangchan
Mr. Anukool Chootong
Mr. Amnuay Nordang
Miss. Khanitta Seatang (Advisor)
2nd Semestor , Educational Year 1997

Abstract

This project is interfacing between the computer to be controller and dc power supply to be listener. The power supply has controlled and organized system to be complete standard IEEE- 488 (General Purpose Interface Bus) listener by 8031-8 bit micro-controller. The micro-controller use for checking signal and status of both computer controlled card and dc power supply.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

ความจำเป็นที่จะต้องมีระบบบัสมาตรฐานขึ้นมาใช้งานกัน ก็เนื่องมาจากเครื่องมือวัดที่ทันสมัยและมีความสามารถสูง อาศัยการทำงานในเชิงดิจิตอลเข้ามาเกี่ยวข้องด้วยและในการใช้งาน เมื่อจำเป็นต้องมีการต่อร่วมกันมากกว่า 1 เครื่อง แต่เดิมนั้นแต่ละผู้ผลิตจะมีระบบเชื่อมต่อของตนเอง ทำให้ไม่สะดวกต่อผู้ใช้งานที่มีเครื่องมือจากหลายบริษัท และเป็นผลเสียต่อผู้ผลิตเครื่องมือวัดเองด้วยในการพัฒนาเครื่องมือใหม่ๆ ออกมา

บริษัทต่างๆ ที่เป็นผู้ผลิตเครื่องมือวัดในอเมริกา จึงได้ตกลงร่วมกันที่จะจัดหาระบบอินเตอร์เฟสมาตรฐานสำหรับที่จะใช้ร่วมกันขึ้นมา และในประเทศเยอรมันก็มีการพัฒนาระบบบัสมาตรฐานขึ้นมาเช่นกันโดยความร่วมมือช่วยเหลือของ IEC (International Electrotechnical Commission) จนในปี 1972 อเมริกาโดย IEEE จึงได้มีการประชุมร่วมมือกันกับ IEC วางแผนพิจารณาระบบบัสมาตรฐานร่วมกัน

บริษัทฮิวเลตแพกการ์ด ผู้ผลิตเครื่องมือวัดรายใหญ่รายหนึ่งในอเมริกาได้ทำการพัฒนาระบบบัสมาตรฐานของตัวเองอยู่ก่อนแล้ว ชื่อว่า HPIB (Hewlett Packard Interface bus) จึงได้เสนอระบบบัสของตัวเองให้ IEEE พิจารณาและได้รับการยอมรับในปี 1975 เรียกว่ามาตรฐาน IEEE Std 488-1975 และมีการปรับปรุงต่อมาเป็น IEEE Std 488-1978 ระบบบัสนี้ก็คือ IEEE 488 ที่กล่าวถึงนั่นเอง สำหรับของ IEC มีการกำหนดมาตรฐานขึ้นมาอีกอันหนึ่งเรียกว่า IEC 625-1 ตั้งใจที่จะให้เป็นมาตรฐานสากลโดยมีรายละเอียดทางเทคนิคทุกประการเหมือนกับ IEEE Std 488-1978 เพียงแต่การวางตำแหน่งของสัญญาณต่างๆ ในขั้วต่อต่างกันเล็กน้อยเท่านั้นเอง คำว่า GPIB จึงหมายรวมถึงทั้ง 2 มาตรฐานดังกล่าว

จุดประสงค์ของบัสก็คือ การส่งข้อมูลข่าวสารถ่ายทอดกันระหว่างเครื่อง ข่าวสารที่ว่านี้ก็มีทั้งข้อมูลที่เป็นผลลัพธ์จากการวัดค่าหรือจากกระบวนการต่างๆ และรวมถึงคำสั่งที่ใช้กำหนดสภาวะทำงานของอุปกรณ์ต่างๆ ในระบบ

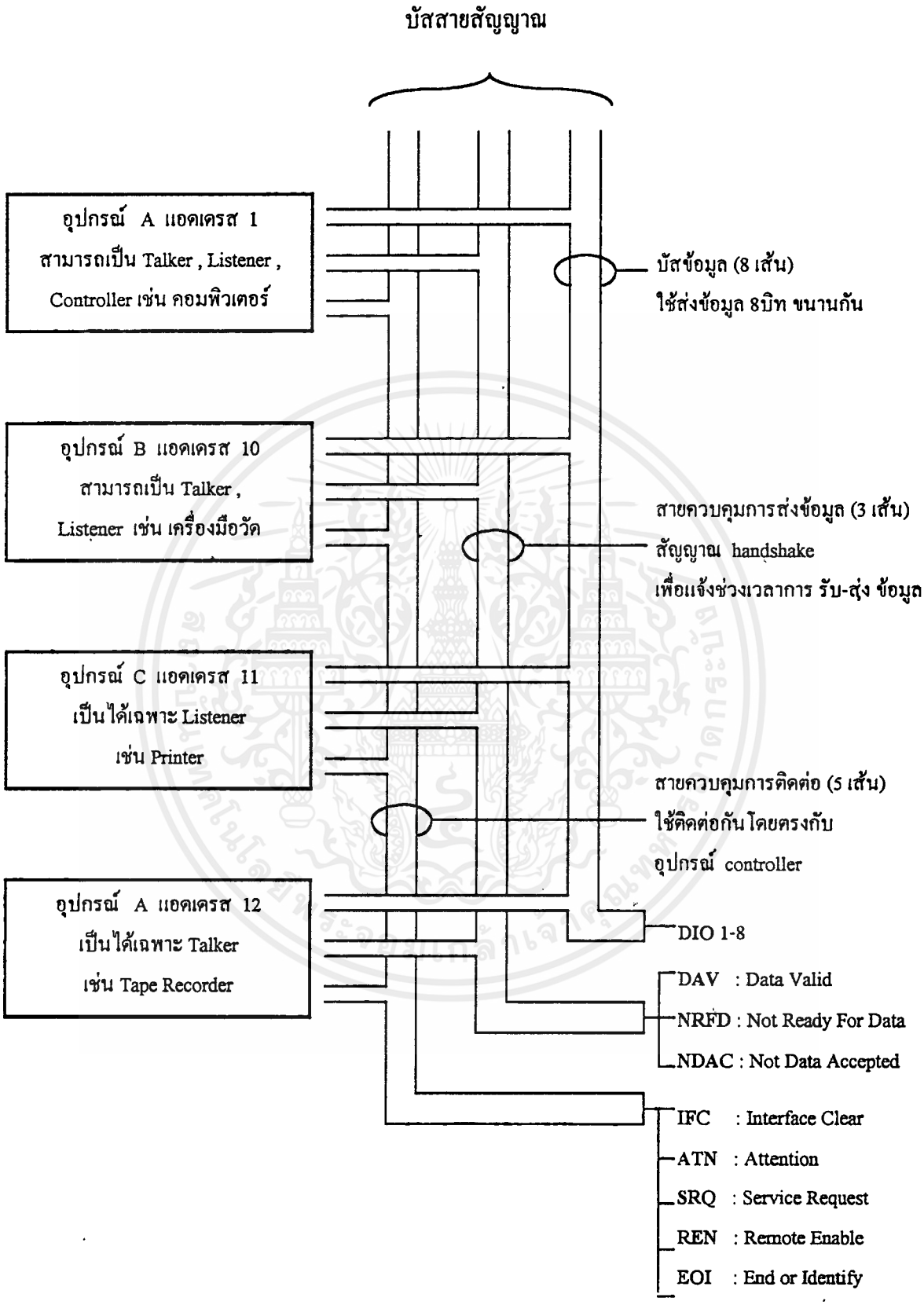
เมื่อมีการต่อเครื่องมือวัดหรืออุปกรณ์ที่ต้องการใช้งานร่วมกันเข้ากับระบบบัสแล้วจำเป็นต้องมีแบบแผนในการทำงานร่วมกันให้สอดคล้องกันจึงต้องมีสัญญาณสำหรับควบคุมระบบ นอกเหนือจากข่าวสารข้อมูลที่ต้องมีการติดต่อกัน (สัญญาณควบคุมนี้ใช้สำหรับควบคุมบัสและจัดระบบสำหรับการอินเตอร์เฟส อาจมองได้ว่าเป็นความเกี่ยวข้องทางฮาดแวร์ ไม่เกี่ยวข้องกับผู้ใช้งานภายนอก ส่วนคำสั่งสำหรับกำหนดการทำงานของอุปกรณ์ในระบบเป็นงานทางด้านซอฟแวร์ ที่เกี่ยวข้องกับการใช้งานโดยตรง)

บทที่ 2
คุณสมบัติและการใช้งานทั่วไป

GPIB มีชื่อเต็มว่า General Purpose Interface Bus เป็นบัสมาตรฐานที่ใช้ในการติดต่อรับส่งข้อมูลระหว่างอุปกรณ์ได้หลายเครื่อง โดยมีข้อพิเศษสำหรับผู้ใช้ก็คือในกรณีที่ต้องการขยายอุปกรณ์เพิ่มเติมเข้ามาในระบบ ผู้ใช้ไม่จำเป็นต้องเพิ่มเติมส่วนวงจรหรืออุปกรณ์อื่นๆ อีกเลย เพียงแต่ผู้ใช้เพิ่มสายเคเบิ้ลเชื่อมต่อเข้ามาขนานกับสายเคเบิ้ลหรือจะต่อเติมเท่านั้น โดยใช้การแก้ไขเฉพาะส่วนของซอฟแวร์

ในระบบที่ไม่ใหญ่โตนัก GPIB นั้นก็สามารถต่อเข้ากับอุปกรณ์หรือเครื่องมืออื่นๆ ได้สูงสุด 15 เครื่อง โดยใช้สายสัญญาณเพียง 1 เส้นต่อขนานกันไปเรื่อยๆ ดังนั้นบัสสัญญาณของ GPIB จึงเป็นบัสแบบขนานโดยมีสัญญาณควบคุมร่วมด้วย เพื่อกำหนดทิศทางและเลือกตัวอุปกรณ์ที่ต้องการจะติดต่อ

หากจะเปรียบเทียบกับมาตรฐานการรับส่งของ RS-232 หรือ Centronics Interface แล้ว GPIB มีข้อยุ่งยากและซับซ้อนกว่าในแง่การใช้งาน เพราะต้องมีการใช้คำสั่งควบคุมอุปกรณ์แต่ละตัวก่อนที่จะรับส่งข้อมูลกันได้



รูปที่ 2.1 แผนผังแสดงอุปกรณ์ GPIB และสายสัญญาณต่างๆ

2.1 โครงสร้างของ GPIB

ส่วนประกอบพื้นฐานของ GPIB แสดงไว้ในรูปที่ 2.1 กล่าวคือ GPIB ประกอบไปด้วยอุปกรณ์ที่ทำหน้าที่เป็นตัวส่ง (Talker), ตัวรับ (Listener) และตัวควบคุม (Controller)

Talker ทำหน้าที่ในการส่งข้อมูลโดยสามารถที่จะนำ Talker จำนวนหลายๆ ตัว มาใส่ไว้ในระบบ แต่จะมี Talker เพียงตัวเดียวเท่านั้นที่กำลังทำงานอยู่

Listener ทำหน้าที่รับข้อมูล Listener ก็เช่นเดียวกับ Talker คือ สามารถนำไปใส่ไว้ในระบบได้หลายๆ ตัว ดังนี้ในระบบหนึ่งๆ จึงประกอบด้วย Talker 1 ตัว และ Listener ตั้งแต่ 1 ตัวขึ้นไป

Controller เป็นตัวควบคุมสัญญาณต่างๆ บนบัส โดยรับความต้องการของอุปกรณ์ที่จะส่งข้อมูลหรือกำหนด Listener ที่จะทำการรับข้อมูล

2.2 อุปกรณ์ที่มีระบบ GPIB

อุปกรณ์หรือเครื่องมือที่มีระบบ GPIB นั้นแบ่งตามหน้าที่การทำงานได้ ดังนี้

1. ทำหน้าที่เป็น Talker เท่านั้น เช่น เครื่องมือวัด เป็นต้น
2. ทำหน้าที่เป็น Listener เท่านั้น เช่น เครื่องพิมพ์ เครื่องบันทึกเป็นต้น
3. ทำหน้าที่เป็นได้ทั้ง Talker และ Listener เช่น Computer เครื่องมือวัดที่สามารถควบคุมได้จากภายนอก เป็นต้น
4. ทำหน้าที่เป็นได้ทั้ง Listener, Talker และ Controller เช่น Computer ที่ทำหน้าที่ควบคุมระบบ

คุณสมบัติทางไฟฟ้าของ GPIB

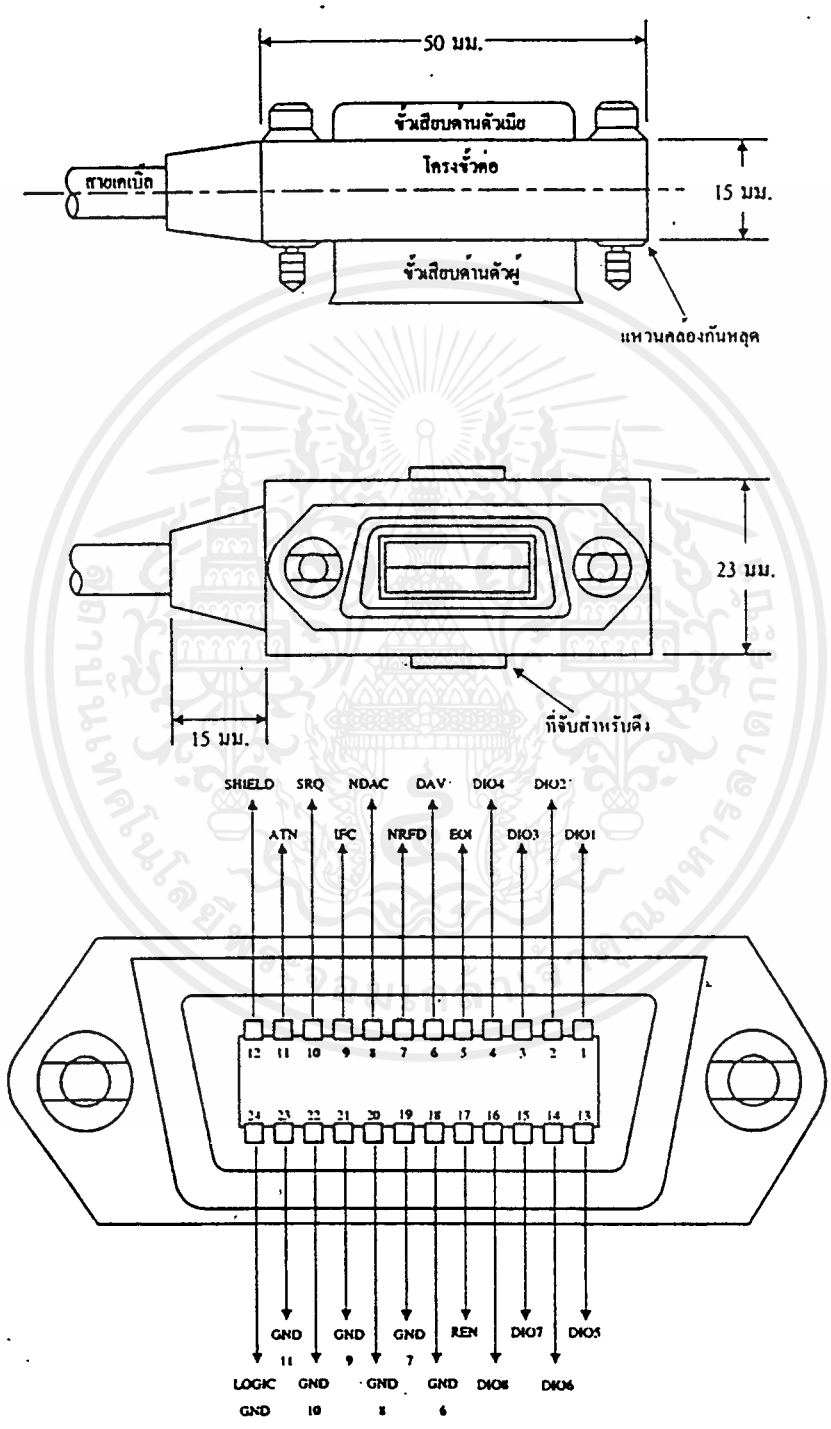
2.3 คุณสมบัติทางไฟฟ้าซึ่งจะเป็นตัวกำหนดขีดจำกัดของ GPIB

คุณสมบัติทางไฟฟ้าซึ่งจะเป็นตัวกำหนดขีดจำกัดของ GPIB นั้น มีดังนี้

1. จำนวนอุปกรณ์ (Talker, Listener, Controller) ที่ต่ออยู่กับสายสัญญาณ 1 เส้น จะต้องไม่เกิน 15 เครื่อง
2. สายเคเบิ้ลที่ใช้ต่ออยู่ระหว่างอุปกรณ์แต่ละตัวต้องยาวไม่เกิน 4 เมตร และความยาวรวมของสายเคเบิ้ลต้องไม่เกิน 20 เมตร
3. ความเร็วในการส่งข้อมูลต้องไม่เกิน 1 Mb/Sec
4. จำนวนของอุปกรณ์หรือเครื่องมือมากกว่าครึ่งหนึ่งต้องเปิดให้ทำงาน (จ่ายไฟ)

2.4 ขั้วต่อของ GPIB

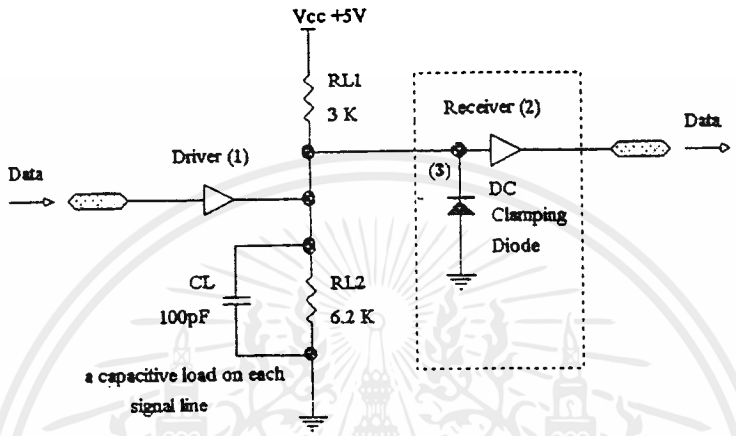
ขั้วต่อมาตรฐานของ GPIB มีทั้งหมด 24 ขา และมีการจัดตำแหน่งสัญญาณที่ขาต่างๆ ดังรูปที่ 2.2



รูปที่ 2.2 ขั้วต่อของ GPIB และการจัดขาของสัญญาณต่างๆ

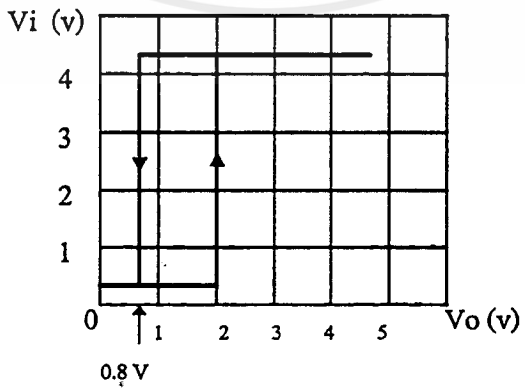
2.4.1 พิจารณาถึงวงจรทางไฟฟ้า

เมื่อพิจารณาถึงวงจรไฟฟ้าเมื่อมองย้อนเข้าไปยังจุดต่อสัญญาณบัสข้อมูล เราจะเห็นโหลดของ GPIB เป็นตัวต้านทานแบ่งแรงดัน 2 ค่า คือ 6.2 kΩ และ 3 kΩ ดังรูปที่ 2.3



รูปที่ 2.3 วงจรภายในจุดต่อบัสข้อมูลที่ใช้รับส่งข้อมูล

ทางด้านเกตตัวส่ง (Driver) เป็นแบบ Open collector หรือ แบบเกต 3 สถานะ (3 State Driver) โดยขับด้วยกระแสขนาด 48 mA ส่วนทางด้านเกตตัวรับ (Receiver) ใช้แบบ Schmitt trigger เพื่อกำจัดสัญญาณรบกวน โดยมีคุณสมบัติการให้สัญญาณเอาท์พุท ดังรูปที่ 2.4



รูปที่ 2.4 คุณสมบัติฮิสเตอริซิสของตัว Receiver แบบ Schmitt trigger

จากรูปที่ 2.4 ที่ระดับแรงดันอินพุทมีค่าต่ำหรือน้อยกว่า 0.8 V เอาท์พุทจะมีค่าต่ำด้วย แต่ถ้าเพิ่มแรงดันอินพุทไปเรื่อยๆ จนมีค่ามากกว่า 2 V เอาท์พุทจะมีค่าสูง หลังจากนั้นถ้าแรงดันมีค่าลดลงเรื่อยๆ จนต่ำกว่า 2 V เอาท์พุทก็ยังค่าสูงอยู่ แรงดันอินพุทต้องลดลงจนกระทั่งต่ำกว่า 0.8 V เอาท์พุทจึงกลับมามีค่าต่ำ ลักษณะเช่นนี้เป็นลักษณะของ ฮิสเตอรีซิส

แต่อย่างไรก็ตามตัวเกต Driver/Receiver ที่ใช้กับ GPIB และมีจำหน่ายอยู่นั้นอาจมีคุณสมบัติแตกต่างไปจากที่กล่าวมาก็ได้ เช่น กรณีที่ใช้เป็นแบบอุปกรณ์แยกชิ้น (Discrete เช่น ทรานซิสเตอร์ทำหน้าที่เป็นสวิตช์) แล้วต่อปลาย (โหลด) ด้วยตัวต้านทานเมื่อปิดแหล่งจ่ายไฟจะทำให้แหล่งจ่ายไฟมีค่าเป็น 0 V (ลัดวงจร) แต่ที่สัญญาณบัสข้อมูลยังมีตัวต้านทาน 6.2 kΩ และ 3 kΩ ต่อขนานกันอยู่ ดังนั้นจึงทำให้มีค่าโหลดประมาณ 2 kΩ ต่อสัญญาณบัสข้อมูลกับกราวด์ซึ่งจะเป็นผลเสียทางช่วงของสัญญาณรบกวน (Noise margin)

กรณีที่ใช้เกต Driver/Receiver เฉพาะของ GPIB เช่น IC เบอร์ SN75160, MC6488A เป็นต้น ซึ่งเป็นโหลดประเภทแอคทีฟ เมื่อปิดแปล่งจ่ายไฟจะทำให้ไม่มีโหลดเข้าไปเกี่ยวพันด้วย ทำให้เกิดผลดีทางด้าน Noise margin คือข้อมูลไม่ผิดพลาดได้ง่าย

2.5 ความหมายของสัญญาณต่างๆ ในบัส

GPIB เป็นระบบบัสแบบขนาน ดังนั้นอุปกรณ์ทุกตัวที่ต่อร่วมบัสกันอยู่จึงต่อขนานกันหมด สัญญาณต่างๆ จากระบบบัสจึงปรากฏต่ออุปกรณ์ทุกตัวในจำนวนสายต่อทั้ง 24 เส้น ของบัสสามารถแบ่งออกได้เป็น 3 กลุ่มสัญญาณ คือ

2.5.1 กลุ่มสัญญาณควบคุมการรับส่งข้อมูล

DAV(Data Valid) เมื่อถูกดึงให้มีระดับแรงดันเป็น "LOW" โดยอุปกรณ์ที่ทำหน้าที่เป็นตัวส่ง (talker) เป็นการแจ้งแก่ระบบบัสว่าตอนนี้ตัวส่งได้ทำการส่งข้อมูลลงไปที่สายสัญญาณข้อมูลเรียบร้อยแล้ว

NRFD(Not Ready For Data) เมื่อถูกดึงให้มีระดับแรงดันเป็น "LOW" เป็นการแสดงว่าตอนนี้ระบบบัสยังไม่พร้อมที่จะรับข้อมูล เนื่องจากอุปกรณ์ในระบบยังพร้อมไม่หมดทุกตัวซึ่งสัญญาณเส้นนี้จะไม่เป็น HIGH จนกว่าอุปกรณ์ทุกตัวให้ระดับแรงดัน HIGH ครบถ้วนแล้วสัญญาณนี้ใช้ประโยชน์สำหรับกรณีที่อุปกรณ์ที่ใช้ร่วมกันมีความเร็วในการทำงานต่างกัน

NDAC (Not Data Accepted) สัญญาณเส้นนี้ควบคุมโดยอุปกรณ์ตัวรับ (Listener) จะให้ระดับแรงดันเป็น "LOW" ในขณะที่ตัวรับกำลังเก็บข้อมูลจากสายข้อมูล และเป็น HIGH เมื่ออ่านข้อมูลเรียบร้อยแล้ว

2.5.2 กลุ่มสัญญาณควบคุมการอินเตอร์เฟส

ประกอบด้วยสัญญาณดังต่อไปนี้

ATN (Attention) เป็นสัญญาณจากอุปกรณ์ที่เป็นตัวควบคุม (controller) ใช้ในการสั่งให้อุปกรณ์ทุกตัวในระบบเตรียมพร้อม เพื่อรอรับคำสั่งต่อไป

IFC (Interface Clear) เป็นสัญญาณรีเซตหรือเคลียร์ระบบ ซึ่งกำเนิดโดยตัวควบคุมเท่านั้น เมื่ออุปกรณ์ในบัสได้รับสัญญาณเคลียร์นี้ จะกลับคืนสู่สภาวะเริ่มต้นใหม่เป็นสภาวะแรกก่อนการกำหนดฟังก์ชั่นเหมือนเมื่อแรกเปิดสวิตช์

REN (Remote Enable) สัญญาณเส้นนี้ควบคุมโดยอุปกรณ์ตัวควบคุมตัวเดียวเท่านั้นเช่นกัน ใช้สั่งให้อุปกรณ์เปลี่ยนจากโหมดที่ใช้งานปกติด้วยมือ มาเป็นการควบคุมโดยตัวควบคุมแทน

SRQ (Service Request) เป็นสายสัญญาณอินเตอร์รัพต์ เพื่อเป็นการบอกแก่ระบบว่า ขณะนี้อุปกรณ์ต้องการการติดต่อจากตัวควบคุม ยกตัวอย่างเช่น โวลต์มิเตอร์ที่มีค่าที่อ่านได้พร้อม แล้วที่จะส่งให้แก่ระบบเพื่อไปยังอุปกรณ์ที่ต้องการข้อมูลนี้ หรือเมื่อเกิดข้อผิดพลาดบางอย่าง

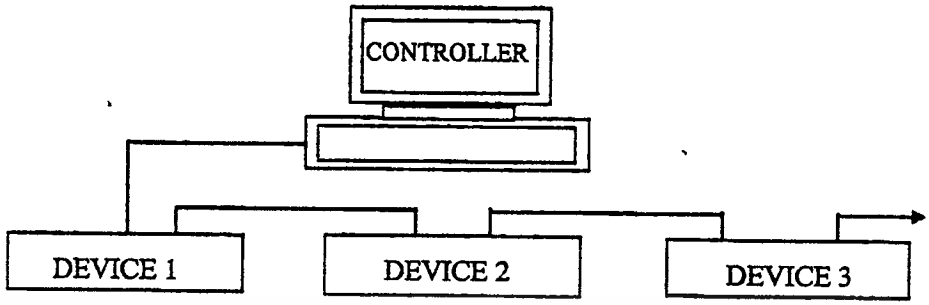
EOI (End Or Identify) สัญญาณที่ควบคุมได้ทั้งโดยอุปกรณ์ที่เป็นตัวควบคุมหรือตัวส่งก็ได้ ใช้แสดงว่าข่าวสารข้อมูลที่ส่งเป็นชุดนั้นสิ้นสุดลงแล้ว

2.5.3 กลุ่มสัญญาณข้อมูล

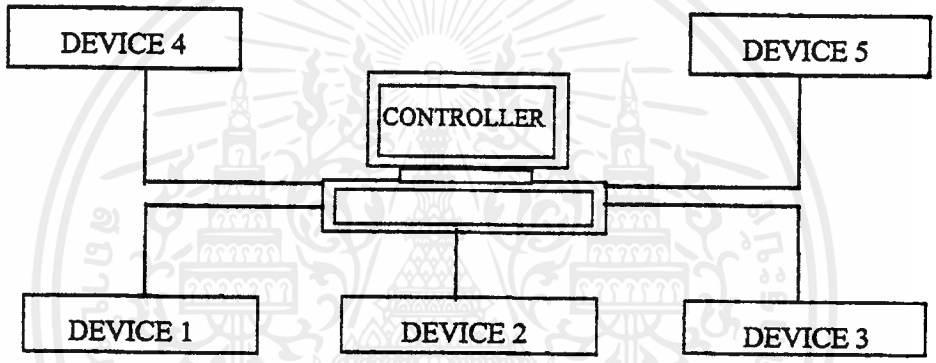
ประกอบด้วยสายสัญญาณจำนวน 8 เส้น สำหรับเป็นทางผ่านของข่าวสารข้อมูลของระบบ สัญญาณลอจิกที่ใช้ใน GPIB บัสนี้มีลักษณะเป็น คอมพลีเมนต์ทั้งหมด คือ "1" เท่ากับ "LOW" และ "0" เท่ากับ "HI" ซึ่งตรงกันข้ามกับที่เราคุ้นเคย

สัญญาณควบคุมที่มีอยู่อาจไม่ได้ใช้ทั้งหมดพร้อมกัน บางสายอาจไม่ได้ใช้งานในอุปกรณ์บางเครื่องก็ได้ แล้วแต่ความจำเป็นในการติดต่อ สำหรับสายที่เหลืออีก 7 เส้นนั้น เป็นสายกราวด์สำหรับการต่ออุปกรณ์ต่างๆ ในระบบ GPIB อินเตอร์เฟสบัส มีอยู่ 2 วิธี คือ แบบต่อเนื่องกันไปเรื่อยๆ จากเครื่องหนึ่งไปยังอีกเครื่องหนึ่งและอีกวิธีหนึ่ง คือ การต่อแบบกระจายดังแสดงในรูปที่ 2.5

A) Daisy Chain



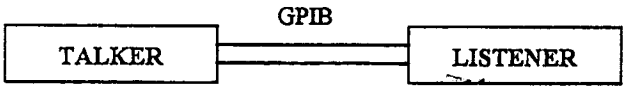
B) Star Configuration



รูปที่ 2.5 ลักษณะการต่อเข้ากับระบบบัส GPIB ทั้ง 2 วิธี

2.6 ขบวนการแฮนด์เชค (Handshake Procedure)

เมื่อมีการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ในระบบ ดังนั้นเมื่อระบบจะทำงานจึงต้องมีการตรวจสอบเสียก่อน ซึ่งการตรวจสอบนี้เราเรียกว่าขบวนการแฮนด์เชค (Handshake Procedure)

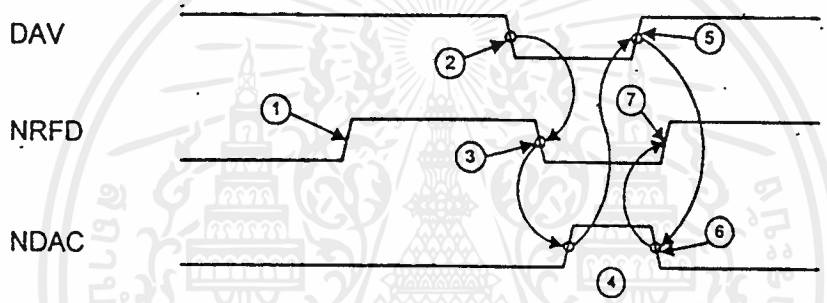


รูปที่2.6 ระบบการอินเตอร์เฟสอย่างง่าย

พิจารณาระบบที่ไม่ซับซ้อนนักโดยกำหนดให้มีตัวส่ง (Talker) และตัวรับ (Listener) อย่างละ 1 ตัว การที่จะมีการสื่อสารระหว่างตัวส่งและตัวรับ จะกระทำได้โดยการที่ตัวส่งจะทำหน้าที่ส่งสัญญาณออกมาเพื่อให้ตัวรับ ทราบว่ามีข้อมูลเสร็จในสายสัญญาณและตัวรับก็จะทำการส่งสัญญาณเพื่อให้ตัวส่งทราบว่าได้ทำการรับข้อมูลเป็นที่เรียบร้อยแล้ว

การส่งสัญญาณในการสื่อสารระหว่างตัวส่งและตัวรับนั้น จะส่งมาทางสายสัญญาณ 3 สาย ซึ่งสัญญาณดังกล่าวนั้นเป็นสายสัญญาณในกลุ่มควบคุมการรับ-ส่งข้อมูล คือ NRFD (Not Ready For Data), DAV (Data Valid) และ NDAC (Not Data Accepted)

DAV เป็นสัญญาณที่ถูกส่งโดยตัวส่ง NRFD และ NDAC เป็นสัญญาณที่ถูกส่งโดยตัวรับโดยตัวรับจะใช้สัญญาณ NDAC เพื่อชี้ให้เห็นว่าพร้อม หรือไม่พร้อมที่จะรับข้อมูล

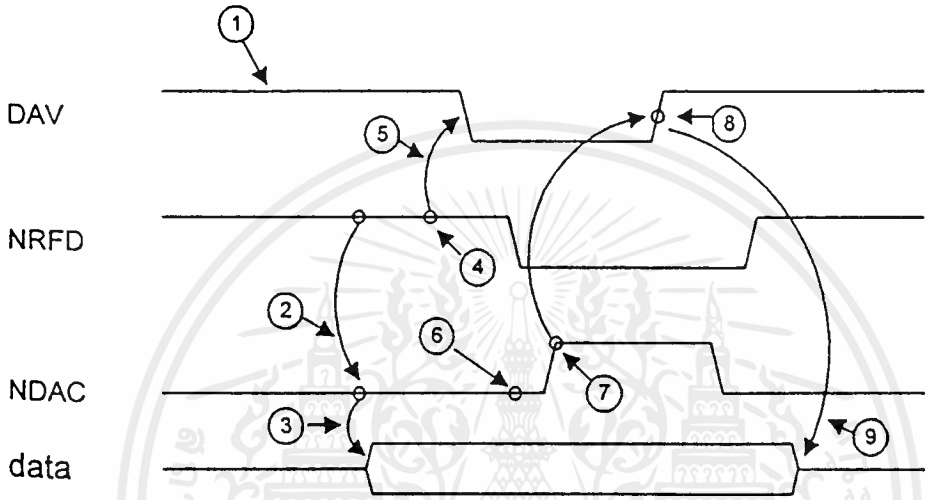


รูปที่ 2.7 แสดงขบวนการแฮนด์เชค

ขบวนการแฮนด์เชคเริ่มขึ้นเมื่อ ตัวส่งมีข้อมูลที่จะถ่ายทอดลงสายสัญญาณไปยังตัวรับ เริ่มด้วยการที่ตัวรับจะทำให้สัญญาณ NRFD เป็น "Hi" (มีค่าเป็น 0) นั่นคือเป็นการบอกให้ทราบว่าตัวรับพร้อมที่จะให้ตัวส่งทำการถ่ายข้อมูลลงในสายสัญญาณแล้ว (ในจุดที่ 1) ตัวส่งจะทำการถ่ายข้อมูลลงในสายสัญญาณ *p-3X DIO1-DI08 หลังจากทำการรอเวลา เพื่อให้ตัวส่งถ่ายข้อมูลแล้ว ตัวส่งจะทำให้สัญญาณ DAV มีลอจิกเป็น "Low" (มีค่าเป็น 1) เพื่อที่จะบอกให้ทราบว่าขณะนี้ได้มีข้อมูลในสายสัญญาณทั้ง 8 เส้นแล้ว (ในจุดที่ 2) ต่อมาเมื่อผู้รับทราบว่ามีข้อมูลอยู่ในบัสก็จะทำให้สัญญาณ NRFD ให้มีค่าเป็น "Low" เพื่อบอกให้ตัวส่งทราบว่ายังไม่พร้อมที่จะให้ตัวส่งทำการส่งข้อมูลชุดต่อไป และตัวรับพร้อมที่จะเก็บข้อมูลจากสายสัญญาณ (ในจุดที่ 3) หลังจากตัวรับทำการเก็บข้อมูลจากสายสัญญาณเสร็จแล้ว ตัวรับจะทำให้สัญญาณ NDAC ให้มีสถานะเป็น "Hi" เพื่อบอกให้ทราบว่าได้ทำการเก็บข้อมูลเสร็จแล้ว (ในจุดที่ 4) เมื่อตัวส่งตรวจพบว่าสัญญาณ NDAC มีลอจิกเป็น "Hi" ก็จะทำให้สัญญาณ DAV ให้มีลอจิกเป็น "Hi" ด้วยเพื่อไม่ให้ตัวรับทำการรับข้อมูลในบัสอีก (ในจุดที่ 5) เมื่อตัวรับทราบว่า สัญญาณ DAV มีลอจิกเป็น "Hi" ก็จะทำให้สัญญาณ NDAC ให้เป็น "Low" ทำให้

ข้อมูลในบัสถูกกำจัดออกไป หลังจากนั้นก็จะทำให้สัญญาณ NRFD ให้เป็น "Hi" เพื่อบอกให้ทราบว่าพร้อมที่จะรับข้อมูลชุดต่อไปแล้วขบวนการแฮนด์เชคจึงเสร็จสิ้นลง และจะเริ่มขึ้นใหม่เมื่อมีสัญญาณชุดใหม่เข้ามา

2.6.1 ขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวส่ง



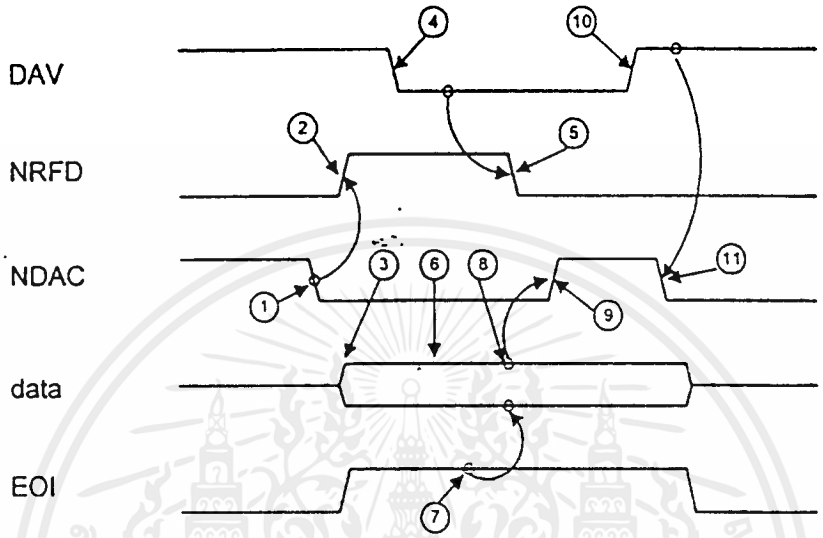
รูปที่ 2.8 ขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวส่ง

ขบวนการแฮนด์เชคเริ่มขึ้นโดยการที่ตัวควบคุม ทำการเซตค่าให้สัญญาณ DAV มีค่าเป็น "Hi" (ในจุดที่ 1) ต่อมาเป็นการตรวจสอบว่าสัญญาณ NDAC และ NRFD มีลอจิก "Hi" ทั้งคู่หรือไม่ (ในจุดที่ 2) ถ้าทั้งคู่เป็น "Hi" นั่นคือเกิดการผิดปกติ คืออุปกรณ์นั้นไม่พร้อมที่จะทำงาน จึงทำการออกจากขบวนการแฮนด์เชค

กลับมาดูในจุดที่ 2 หากสัญญาณใดสัญญาณหนึ่งมีลอจิกเป็น "Low" ตัวควบคุมจะทำการส่งข้อมูลลงใน GPIB Bus (ในจุดที่ 3) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบว่าสัญญาณ NRFD มีลอจิกเป็น "Hi" (ในจุดที่ 4) หลังจากนั้นตัวควบคุมจะทำให้สัญญาณ DAV ให้มีลอจิกเป็น "Low" เพื่อให้ตัวรับทราบว่ามีข้อมูลอยู่ในบัสข้อมูล (ในจุดที่ 5) ตัวควบคุมจะรอเวลาเพื่อให้ตัวรับทำการเก็บข้อมูล เมื่อครบกำหนดเวลาตัวควบคุมจะทำการเชคว่าตัวรับทำงานเกินเวลาที่กำหนดหรือไม่ หากเกินเวลาก็จะทำการตรวจสอบต่อไปว่าสัญญาณ NDAC ทำลอจิกเป็น "Hi" ใช่หรือไม่ก็จะทำการตรวจสอบอีกครั้งว่าเกินเวลาหรือไม่ กลับไปดูที่จุดที่ 6 หากสัญญาณ NDAC เป็น "Hi" (ในจุดที่ 7) ตัวควบคุมจะทำการตอบสนองโดยการทำสัญญาณ DAV ให้มีลอจิกเป็น "Hi" เพื่อบอกให้ตัวรับ

ทราบว่าให้หยุดการเก็บข้อมูลจากบัส (ในจุดที่ 8) หลังจากนั้นข้อมูลในบัสข้อมูลจะถูกนำออกไป (จุดที่ 9) จึงเป็นการเสร็จสิ้นขบวนการแฮนด์เชค

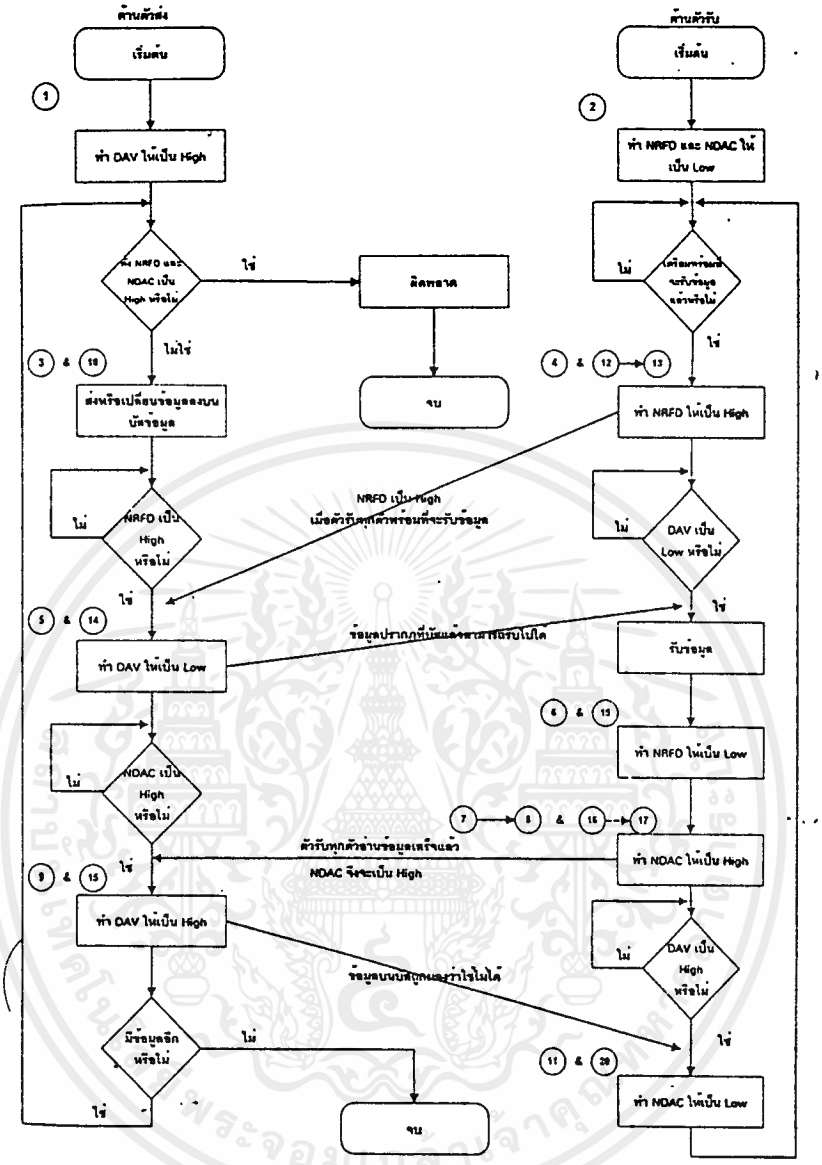
2.6.2 ขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวรับ



รูปที่ 2.9 ขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวรับ

ขบวนการแฮนด์เชคเริ่มขึ้น หลังจากการที่ตัวควบคุมรับรู้ว่าตัวส่งจะทำการส่งข้อมูลเมื่อตัวควบคุมทราบว่าตัวส่งจะทำการส่งข้อมูล ตัวควบคุมรับรู้ว่าตัวส่งจะทำให้สัญญาณ NDAC มีลอจิกเป็น "LOW" เพื่อบอกให้ทราบว่าตัวควบคุมยังไม่ได้ทำการเก็บข้อมูล (ข้อมูลในที่นี้คือ แอดเดรส หรือข้อมูลทั่วไปซึ่งตัวส่งทำการส่งมาในบัสข้อมูล) นั่นคือในจุดที่ 1

ต่อมาตัวควบคุมจะทำการเซ็ตค่าสัญญาณ NRFD ให้เป็น "HIGH" เพื่อที่จะให้ตัวส่งทราบว่าขณะนี้ตัวควบคุมพร้อมที่จะรับข้อมูลแล้ว (ในจุดที่ 2) เมื่อสัญญาณ NRFD มีค่า "HIGH" และสัญญาณ NDAC มีค่าเป็น "LOW" แล้วตัวส่งจะทราบทันทีว่าทำการส่งข้อมูลลงมาในบัสข้อมูลได้แล้ว (ในจุดที่ 3) หลังจากนั้นตัวควบคุมจะรอเวลาให้ตัวส่งทำการส่งข้อมูลลงในบัสข้อมูลให้เสร็จ โดยจะทำการตรวจสอบไปด้วยว่าเกินเวลาที่กำหนดหรือยัง หากเกินเวลาที่กำหนดก็จะออกจากขบวนการแฮนด์เชค หากไม่เกินกำหนดเวลาจะทำการตรวจสอบเวลาว่าเกินเวลาที่ต้องคอยแล้วหรือยัง หากยังไม่เกินเวลาก็จะทำการเชคต่อไปจนกระทั่งหมดเวลาหรือจนกว่าสัญญาณ DAV เป็น "Low" หรือไม่หากไม่ก็จะตรวจสอบเวลาว่าเกินเวลาที่ต้องรอแล้วหรือยัง หากยังไม่เกินเวลาก็จะทำการเช็คต่อไปจนกระทั่งหมดเวลาหรือจนกว่าสัญญาณ DAV จะมีค่าเป็น "Low"



รูปที่ 2.10 แสดงผังงานของการรับส่งข้อมูล

เมื่อสัญญาณ DAV เป็น "Low" แล้วตัวควบคุมจะตอบรับ โดยการทำให้สัญญาณ NRFD มีค่าเป็น "Low" (ในจุดที่ 5) นั่นเป็นการบอกว่าตัวควบคุมพร้อมที่จะเริ่มเก็บข้อมูล (ในจุดที่ 6) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ EOI เพื่อตรวจสอบว่าข้อมูลที่ตัวส่งทำการส่งลงมานั้นหมดแล้วหรือยัง (ถ้าตัวส่งทำการส่งข้อมูลลงในบัสหมดแล้วจะทำการตั้งค่าสัญญาณ EOI ให้เป็น "Low") (ในจุดที่ 7) หากสัญญาณ EOI เป็น "Low" ตัวควบคุมจะเซตสถานะว่า ขณะนี้ตัวส่งได้ทำการส่งข้อมูลลงมาในบัสหมดแล้ว หากสัญญาณ EOI เป็น "Hi" ก็ไม่ทำการเซตสถานะ หลังจากนั้นตัวควบคุมจะทำการเก็บข้อมูลในบัส (ในจุดที่ 8) แล้วทำให้สัญญาณ NDAC มีค่าเป็น "Hi" เพื่อบอกให้

ทราบว่าตัวควบคุมได้ทำการอ่านข้อมูลเสร็จเรียบร้อยแล้ว (ในจุดที่ 9) เมื่อตัวควบคุมทำให้สัญญาณ NDAC เป็น "Hi" (ในจุดที่ 10) ซึ่งก่อนหน้านี้ตัวควบคุมจะทำการตรวจสอบอยู่แล้วว่าสัญญาณ DAV เป็น "Hi" หรือยัง

เมื่อสัญญาณ DAV-เป็น "Hi" แล้วตัวควบคุมจะทำให้สัญญาณ NDAC มีค่าเป็น "Low" (ในจุดที่ 11) แล้วจึงออกจากขบวนการแฮนด์เชค

2.7 คำสั่งใช้งานของ GPIB

การสั่งการต่างๆ เพื่อกำหนดหน้าที่การทำงานและกำหนดฟังก์ชัน เช่น กำหนดช่วงการวัด โหมดการวัด หรืออื่นๆ แก่เครื่องวัดที่ต่ออยู่ เหล่านี้นั้นตัวควบคุมจะเป็นตัวกำหนดโดยการส่งรหัสคำสั่งไปที่อุปกรณ์โดยผ่าน DI1-DI6 รหัสคำสั่งนี้จะถูกส่งไปในช่วงสายสัญญาณ ATN เป็น LOW

คำสั่งสำหรับกำหนดหน้าที่การทำงานต่างๆ ตามมาตรฐานของ GPIB มีอยู่ด้วยกัน 128 คำสั่ง ดังแสดงในตารางที่ 2.1 ต่อไปนี้ โดยแบ่งเป็น 5 กลุ่มคำสั่ง

รหัสที่ใช้ในระบบ GPIB บัสนั้นใช้ร่วมกัน ทั้งรหัสข้อมูล และรหัสคำสั่ง นั่นคือรหัสเดียวกันมีความหมายได้ 2 อย่าง คือ เมื่อ ATN เป็น LOW จะหมายถึงรหัสคำสั่ง แต่ถ้า ATN เป็น HIGH รหัสนี้จะแทนข้อมูลที่เป็น ASCII แทน ซึ่งในตารางก็ได้แบ่งความหมายออกเป็น 2 คอลัมน์

ตารางที่ 2.1

ASCII-IEEE 488 BUS MESSAGES (COMMANDS AND ADDRESSES) HEX CODES

MSD / LSD	0		1		2		3		4		5		6		7	
	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG
0	NUL		DLE		SP	00	0	16	@	00	P	16	.		p	
1	SOH	GTL	DC1	LLO	!	01	1	17	A	01	Q	17	a		q	
2	STX		DC2		..	02	2	18	B	02	R	18	b		r	
3	ETX		DC3		“	03	3	19	C	03	S	19	c		s	
4	EOT	SDC	DC4	DCL	S	04	4	20	D	04	T	20	d		t	
5	ENO	PPC	NAK	PPU	1/2	05	5	21	E	05	U	21	e		u	
6	ACX		SYN		&	06	6	22	F	06	V	22	f		v	
7	BEL		ETB		‘	07	7	23	G	07	W	23	g		w	
8	BS	GET	CAN	SPE	(	08	8	24	H	08	X	24	h		x	
9	HT	TCT	EM	SPO	)	09	9	25	I	09	Y	25	i		y	
A	LF		SUB		.	10	:	26	J	10	Z	26	j		z	
B	VT		ESC		+	11	:	27	K	11	[	27	k		}	
C	FF		FS		.	12	<	28	L	12	\	28	l		.	
D	CR		GS		-	13	=	29	M	13	]	29	m		{	
E	SO		RS		.	14	>	30	N	14	-	30	n		-	
F	SI		US		/	15	?	UNL	O	15	_	UNT	o		DEL	

ADDRESSED COMMAND GROUP

UNIVERSAL COMMAND GROUP

LISTEN ADDRESS GROUP

TALK ADDRESS GROUP

XECONDARY COMMAND GROUP

PRIMARY COMMAND GROUP (PCG)

2.7.1 กลุ่มคำสั่งเจาะจงจุดหมาย (addressed command group)

เป็นคำสั่งที่สั่งไปยังอุปกรณ์ที่เป็นตัวส่งหรือตัวรับที่กำหนดไว้ล่วงหน้าแล้ว คำสั่งนี้ประกอบด้วย

GTL (got to local) สั่งให้อุปกรณ์กลับสู่สภาพการควบคุมปกติด้วยมือ

SCT (selected device clear) สั่งให้อุปกรณ์เคลียร์ตัวเองสู่สภาพเริ่มต้นใหม่

PPC (paralled poll configure) เป็นคำสั่งสำหรับทำการจัดสายสัญญาณของการทำการจัดสรรสายสัญญาณของการทำกระบวนการตรวจสอบสภาพอุปกรณ์โดยวิธีขนานโดยใช้กับกลุ่มคำสั่งรอง

GET (group excute trigger) ใช้สั่งเริ่มต้นการทำงานของอุปกรณ์ทีละหลายตัว

TCT (take control) เป็นการกำหนดให้อุปกรณ์ตัวส่งทำหน้าที่เป็นตัวควบคุม

2.7.2 กลุ่มคำสั่งครอบคลุม (universal command group)

เป็นคำสั่งที่ส่งไปยังอุปกรณ์ทุกตัวที่ต่ออยู่ในบัส ประกอบด้วย

LLO (local lockout) เป็นการสั่งให้อุปกรณ์ล็อกอยู่ที่สภาวะควบคุม โดยปุ่มปรับที่หน้าปัทม์ตามปกติ

DCL (device clear) สั่งให้อุปกรณ์ทุกตัวกลับไปสู่สภาวะเริ่มต้น

PPU (parallel poll unconfigure) ใช้ยกเลิกกระบวนการตรวจสอบสภาพแบบขนานทั้งหมด

SPE (serial poll enable) เปลี่ยนโหมดของการตรวจสอบสภาพเป็นแบบอนุกรมในโหมดนี้จะเป็นการส่งสถานะของเครื่องแทนการส่งข้อมูล

SPD (serial poll disable) ยกเลิกโหมดการตรวจสอบแบบอนุกรม

2.7.3 กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (listener address group)

เป็นคำสั่งสำหรับกำหนดให้อุปกรณ์เป็นตัวรับ ตามรหัสหมายเลขจาก 0-30 และมีคำสั่ง UNT (untalker) สำหรับยกเลิก

2.7.4 กลุ่มคำสังกำหนดอุปกรณ์ตัวส่ง (talker address group)

สำหรับกำหนดให้อุปกรณ์เป็นตัวส่ง ตามรหัสหมายเลขจาก 0-30 และมีคำสั่ง UNL (unlisten) สำหรับยกเลิกเช่นกัน

คำสั่งกลุ่มที่ 1 ถึง 4 นั้น จัดเป็นกลุ่มคำสั่งหลัก ที่มีความหมายตายตัวยังมีคำสั่งอีกกลุ่มที่ขึ้นอยู่กับการกำหนดภายหลังนั่นคือกลุ่มคำสั่งรอง

2.7.5 กลุ่มคำสั่งรอง (secondary command group)

เป็นคำสั่งที่กำหนดรายละเอียดย่อยของอุปกรณ์แต่ละตัวที่ต่ออยู่ในระบบ ให้มีการทำงานอย่างไร ตามจุดประสงค์ใช้งานของเครื่องมือนั้น เช่นเดียวกับการปรับปุ่มต่างๆ ด้วยมือนั่นเอง คำสั่งรองนี้จะตามหลังคำสั่งหลักคือจะใช้หลังจาก อุปกรณ์ต่างๆ ถูกกำหนดวางตัวในระบบเรียบร้อยแล้ว

คำสั่งต่างๆ ที่กล่าวไป ซึ่งใช้ในการกำหนดสภาวะการทำงาน ของอุปกรณ์ แต่ละสภาวะที่กำหนดไปนั้นเป็นอย่างไร และมีจุดประสงค์เพื่ออะไร ดังต่อไปนี้

2.7.5.1 Device clear/ Interface Clear

Device clear ใช้ในการทำให้อุปกรณ์ที่ต่ออยู่ในบัสกลับไปสู่สภาวะเริ่มต้น ยังไม่มีการกำหนดฟังก์ชันใดๆ สภาวะเริ่มต้นนี้จะแตกต่างกันไป แล้วแต่ว่าอุปกรณ์นั้นออกแบบไว้อย่างไร Device clear มีอยู่ 2 ลักษณะ คือ เคลียร์หมดทุกตัวที่ต่ออยู่ (DCL) กับเคลียร์เจาะจงอุปกรณ์ตัวใดตัวหนึ่ง (SDC)

แต่ว่าในการเคลียร์อุปกรณ์ให้อยู่ในสภาวะเริ่มต้นนั้นไม่ได้หมายความว่า Interface function ของ GPIB จะถูกเคลียร์อุปกรณ์ให้ไปอยู่ในสภาวะเริ่มต้นด้วยแต่อย่างใด interface function คือสภาพการ interface ที่ได้กำหนดไว้ในระบบประกอบด้วยฟังก์ชันต่างๆ ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 แสดงฟังก์ชันต่างๆ ของการอินเตอร์เฟส

ฟังก์ชัน	สัญลักษณ์	การกลับสู่จุดเริ่มต้นโดย IFC
source handshake	SH	ได้
acceptor handshake	AH	ได้
talker or enlarge talker	T or TE	ได้
listener or enlarge listener	L or LT	ได้
service request	SR	ไม่ได้
reomte/local	RL	ไม่ได้
parallel poll	PP	ไม่ได้
device clear	DC	ได้
device trigger	DT	ได้
controller	C	ได้

2.7.5.2 Remote / Local

remote เป็นการกำหนดให้อุปกรณ์ที่อยู่ในระบบ เช่น เครื่องมือวัดให้อยู่ในการควบคุมของอุปกรณ์ตัวอื่นแทน ซึ่งปุ่มปรับต่างๆ บนหน้าปัทม์เครื่องจะไม่มีผลต่อการทำงานส่วน local เป็นการควบคุมการทำงานของเครื่องมือวัดด้วยปุ่มปรับบนหน้าปัทม์ตามปกติ

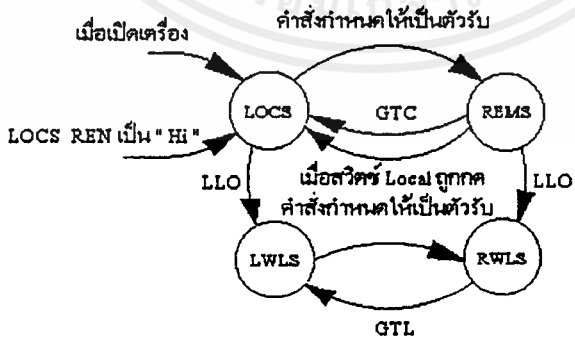
การใช้ Remote มีประโยชน์ในแง่ที่ขณะที่ตัวการควบคุมเช่น คอมพิวเตอร์กำลังติดต่ออุปกรณ์ตัวนั้นอยู่ หากไม่มีการตัดการควบคุมโดยปุ่มปรับบนหน้าปัทม์ออก ถ้ามีใครมาปรับแต่งก็จะทำให้การทำงานผิดพลาดไปได้ การทำงานของ GPIB ใน remote and local มี 4 ลักษณะดังนี้

1. LOCS ก็คือ local นั่นเอง เป็นสภาพการควบคุมที่ปุ่มตามปกติ จะอยู่ในสภาพนี้ ตอนเปิดเครื่อง หรือ REN เป็น HIGH หรือเมื่อได้รับคำสั่ง GTL

2. REMS คือ remote หมายถึงการตัดการควบคุมโดยปุ่มหน้าปัทม์ออก จะเกิดขึ้นเมื่อ REN เป็น LOW และจะถูกล็อกไว้ เว้นแต่ว่าสวิตช์ local ที่ตัวอุปกรณ์จะถูกเปลี่ยนไปตำแหน่ง local

3. RWLS เป็นสภาพ remote ที่ถูกล็อกเอาไว้เช่นกัน แต่ว่าจะตัดการควบคุมตรงสวิตช์ local ที่ตัวอุปกรณ์ออกไป สภาพ remote โดย RWLS จึงมีความสำคัญสูงกว่า REMS อย่างไรก็ตามยังถูกยกเลิกได้ด้วยคำสั่ง LLO

4. LWLS มีสภาพเช่นเดียวกับ local แต่จะแตกต่างกันตรงที่สภาพ local โดย LWLS นี้ เมื่อได้รับคำสั่งกำหนดอุปกรณ์ตัวรับจะเปลี่ยนไปอยู่ในสภาพแบบล็อกหรือ RWLS ทันทีในการที่จะมาที่สภาพ LWLS นี้ได้ก็มี 2 กรณีคือ เมื่ออยู่ในสภาพ local ธรรมดา (LOCS) แล้วได้รับคำสั่ง LLO หรืออยู่ใน RWLS แล้วได้รับคำสั่ง GTL



รูปที่ 2.11 สภาพโลคัลและรีโมตลักษณะต่างๆ และแผนภูมิการเปลี่ยนสภาวะ

2.8 การขอบริการและการตรวจสอบ (Service Request and Polling)

เมื่อตัวควบคุมได้รับ SRQ เป็น LOW จะให้อุปกรณ์ส่งข้อมูลแสดงสถานะการทำงานซึ่งมีอยู่ 2 วิธี คือ

1. การตรวจสอบแบบอนุกรม ซึ่งมีขั้นตอนดังนี้

1.1 ATN ถูกดึงเป็น LOW หลังจากได้รับ LOW จากสายสัญญาณ SRQ

1.2 คำสั่ง UNL ถูกสั่งไปยังอุปกรณ์

1.3 ตัวควบคุมจะแจ้งรหัสตัวรับของตน และกำหนดรหัสตัวส่งของอุปกรณ์ที่จะตรวจสอบไปที่บัส

1.4 ตามด้วยคำสั่ง SPE และสาย ATN กลายเป็น HI ซึ่งอุปกรณ์ที่ถูกเรียกจะส่งข้อมูลแสดงสถานะออกมา 1 ไบต์ โดยบิตที่ 7 จะเป็นตัวชี้ว่าอุปกรณ์ไหนเป็นตัวขอบริการ ถ้าใช่จะเป็น LOW ส่วนบิตอื่นๆ ก็ใช้บอกข้อมูลอื่นๆ ซึ่งมิได้กำหนดเฉพาะ

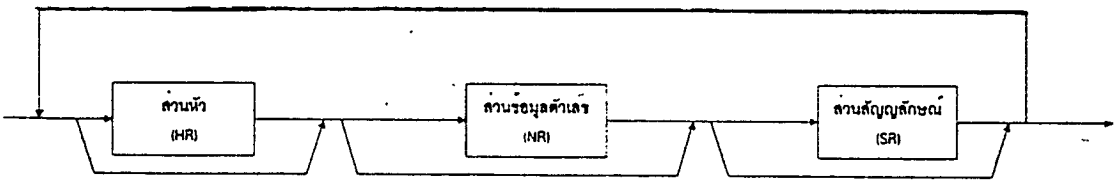
1.5 สาย ATN ถูกดึงเป็น LOW อีกที เพื่อส่งคำสั่งยกเลิกการตรวจสอบคือ SPD

1.6 จากนั้นคำสั่ง UNT ก็ถูกส่งไปยังอุปกรณ์เพื่อยกเลิกการเป็นตัวส่งซึ่งถ้าหาก SRQ ยังคงเป็น LOW อยู่ ก็จะมีการตรวจสอบไปยังอุปกรณ์ตัวอื่นๆ ต่อไปตามขั้นตอนเดิม

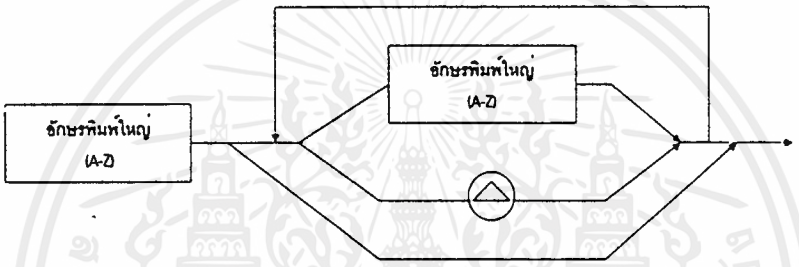
2. การตรวจสอบแบบขนาน สามารถทำได้เร็วกว่าแบบอนุกรมทั้งนี้เพราะสามารถอ่านข้อมูลเพราะสามารถอ่านข้อมูลเพียงไบต์เดียวก็สามารถรู้ได้ทันทีว่าอุปกรณ์ตัวใดเป็นผู้ขอบริการ

2.9 รูปแบบข้อมูล

โดยทั่วไปข้อมูลจากอุปกรณ์ (device message) แบ่งได้ออกเป็น 3 ส่วนดังแสดงในรูปที่ 2.12 อันได้แก่ส่วนหัว (HR) ซึ่งจะอยู่ส่วนหน้าสุด เป็นตัวบอกชนิดข้อมูล ส่วนประกอบ HR แสดงในรูปที่ 2.13จะเห็นว่าประกอบด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ช่องว่าที่เป็นเว้นวรรค ( ) ปกติจะมีอักษรประมาณ 1-3 ตัว

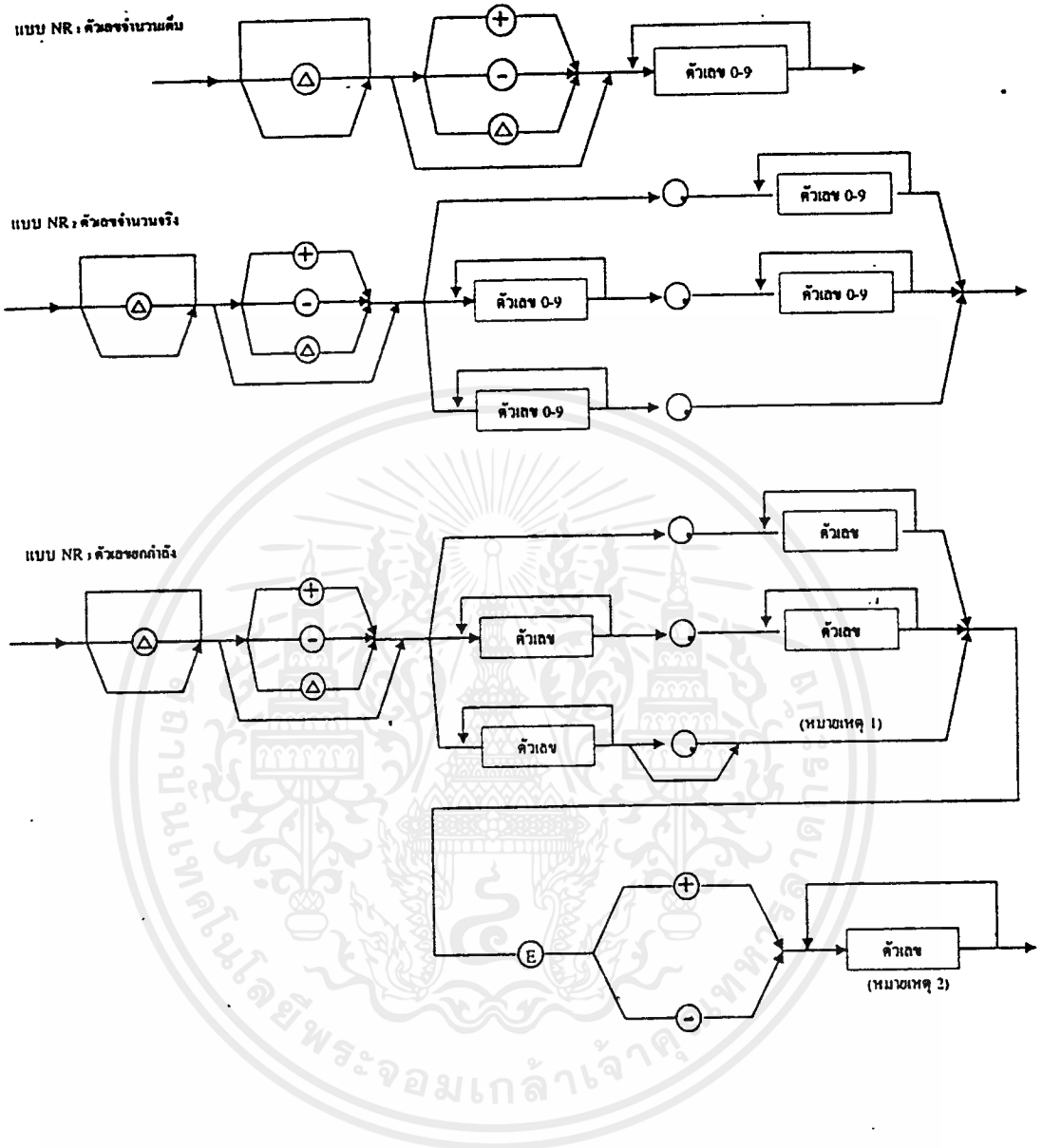


รูปที่ 2.12 แสดงส่วนประกอบของข้อมูลอุปกรณ์



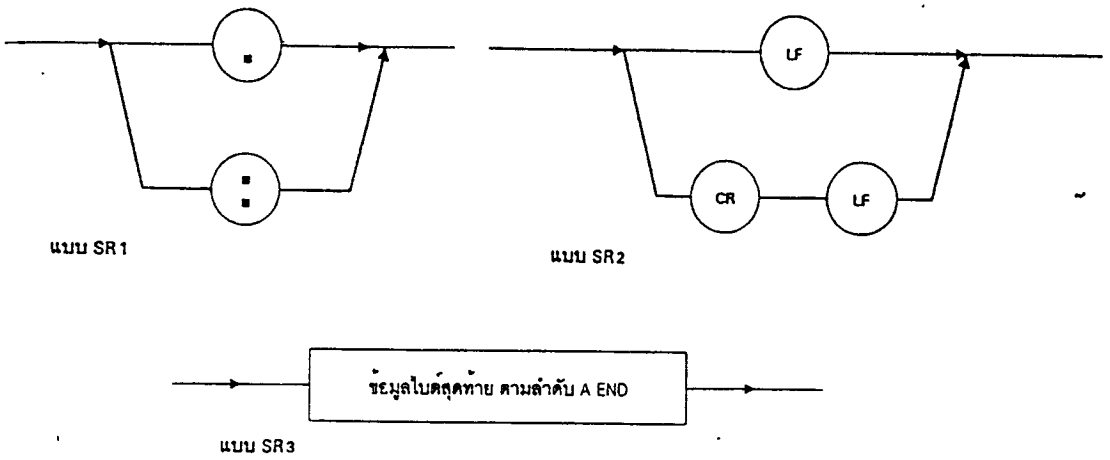
รูปที่ 2.13 แสดงรายละเอียดของส่วนตัว (HR)

ส่วนที่สองคือเนื้อหาข้อมูล (NR) ซึ่งใช้แสดงค่าตัวเลข มีอยู่ 3 แบบ คือ NR1, NR2 และ NR3 ดังแสดงในรูปที่ 2.14 ส่วนท้ายของ NR ยังอาจมีตัวอักษรแสดงหน่วยตามมา

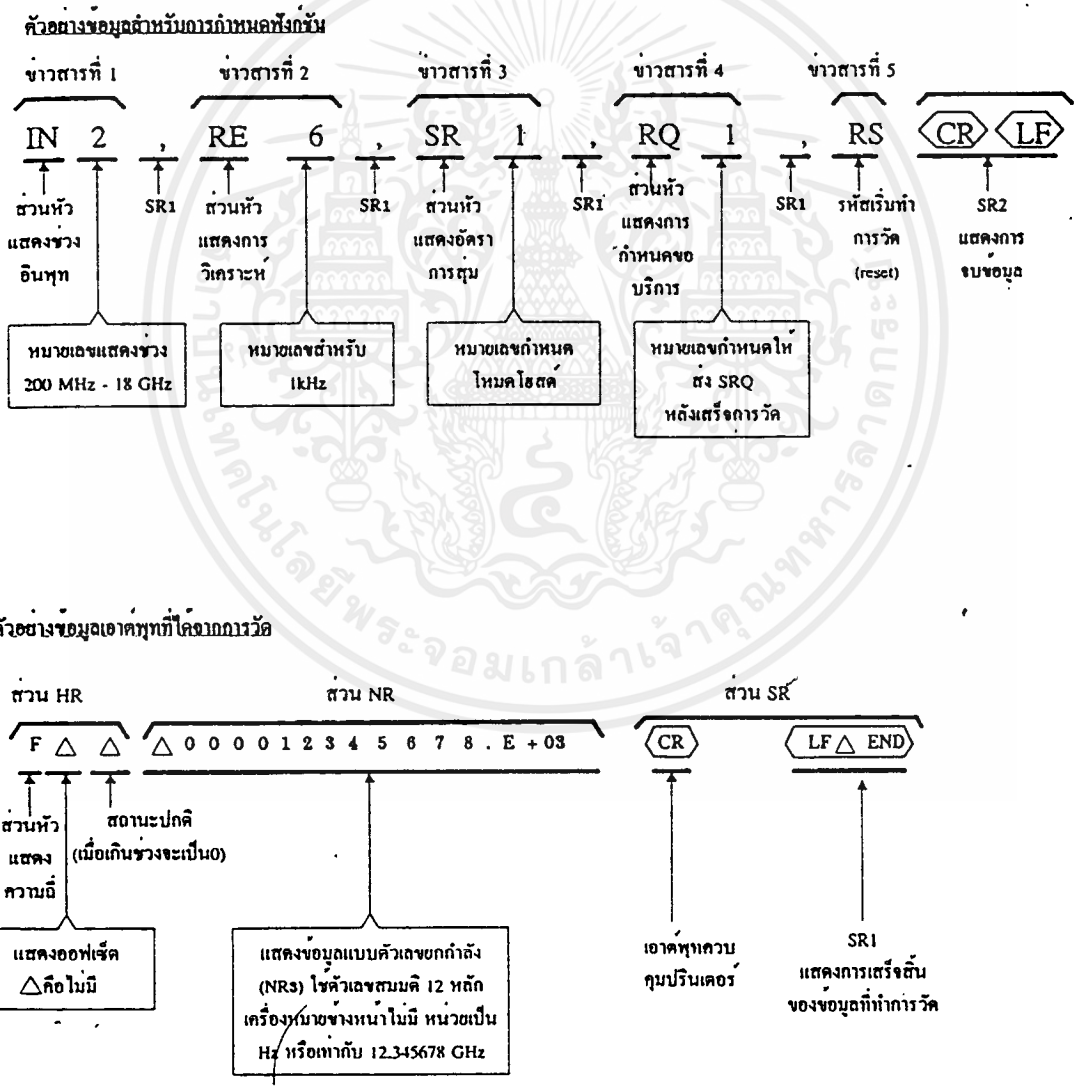


รูปที่ 2.14 แสดงรายละเอียดของส่วนข้อมูลตัวเลข (NR)

ส่วนที่ 3 คือสัญญาณแบ่งข้อมูลแต่ละชุด (SR) ดังแสดงในรูปที่ 2.15 โดน SR1 ใช้แสดงการต่อเนื่องของข้อมูล (ข้อมูลยังมีต่อ) SR2 และ SR3 แสดงการสิ้นสุดของข้อมูล แต่ SR3 เป็นการบอกการเสร็จสิ้นข้อมูลทั้งหมดจากการวัด และรูปที่ 2.16 เป็นตัวอย่างข้อมูลสำหรับการกำหนดฟังก์ชั่นให้เครื่องวัดความถี่และข้อมูลความถี่ที่วัดได้



รูปที่ 2.15 แสดงรายละเอียดของส่วนสัญลักษณ์แบ่งย่อย (SR)



รูปที่ 2.16 แสดงตัวอย่างข้อมูลจริงจากอุปกรณ์

บทที่ 3
การอินเตอร์เฟส

3.1 การเชื่อมต่อกับคอมพิวเตอร์

คอมพิวเตอร์สามารถเชื่อมต่อกับอุปกรณ์ภายนอกต่าง ๆ เพื่อทำการวัดปริมาณทางกายภาพและส่งผลกลับ ในการควบคุมปริมาณทางกายภาพที่แวดล้อมตัวเราอาจจะเป็นภายในบ้าน ห้องทดลอง โรงงานอุตสาหกรรมก็ได้ ปริมาณกายภาพเหล่านี้ได้แก่ อุณหภูมิ ความชื้น ระดับ เป็นต้น โดยคอมพิวเตอร์จะมีเส้นทางในการเชื่อมต่อกับเครื่องมือและอุปกรณ์ภายนอกโดยมีการเรียกข้อมูลเข้ามาและเมื่อต้องการควบคุมก็ส่งข้อมูลออกไป ซึ่งจำเป็นจะต้องกระทำผ่านทางตัวแปลงสัญญาณก็คือ interface card นั่นเอง

3.1.1 การเชื่อมต่อระหว่างอุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์

ประกอบด้วยอุปกรณ์ 2 ส่วนคือ

1) ส่วนของ hard ware (Interface Card) โดยส่วนนี้จะทำหน้าที่ในการแปลงสัญญาณอนาล็อกจากอุปกรณ์ภายนอกมาเป็นสัญญาณดิจิตอลเพื่อส่งให้กับคอมพิวเตอร์ประมวลผลต่อไปแล้วทำหน้าที่ในการแปลงสัญญาณดิจิตอลที่ประมวลผลแล้วนั้นเป็นสัญญาณอนาล็อกเพื่อส่งไปควบคุมอุปกรณ์เอาท์พุทต่อไป นอกจากนี้ยังมีหน้าที่อีกหลายอย่างซึ่งแล้วแต่ผู้ออกแบบว่าจะกำหนดให้การ์ดนี้ทำหน้าที่อะไรบ้าง

2) ส่วนของ sorf ware ส่วนนี้ทำหน้าที่ในการควบคุมการทำงานของ interface card และนำสัญญาณจากภายนอกที่ผ่านการแปลงเรียบร้อยแล้วมาแสดงผลทางจอภาพโดยในส่วนนี้ประกอบด้วยโปรแกรมต่างๆที่ทำหน้าที่ในการสั่งการให้คอมพิวเตอร์ทำงาน ตามที่ผู้เขียนโปรแกรมได้กำหนดไว้ เช่นให้รับค่าที่อินพุทเข้ามาประมวลผล การแสดงผลหน้าจอในโหมดกราฟฟิก การกำหนดพารามิเตอร์ต่างๆ เกี่ยวกับการควบคุม เป็นต้น จากที่กล่าวมาเป็นรายละเอียดของการ interface โดยทั่วไป ต่อไปนี้จะกล่าวถึง interface card ของโครงงานนี้ต่อไปคือ

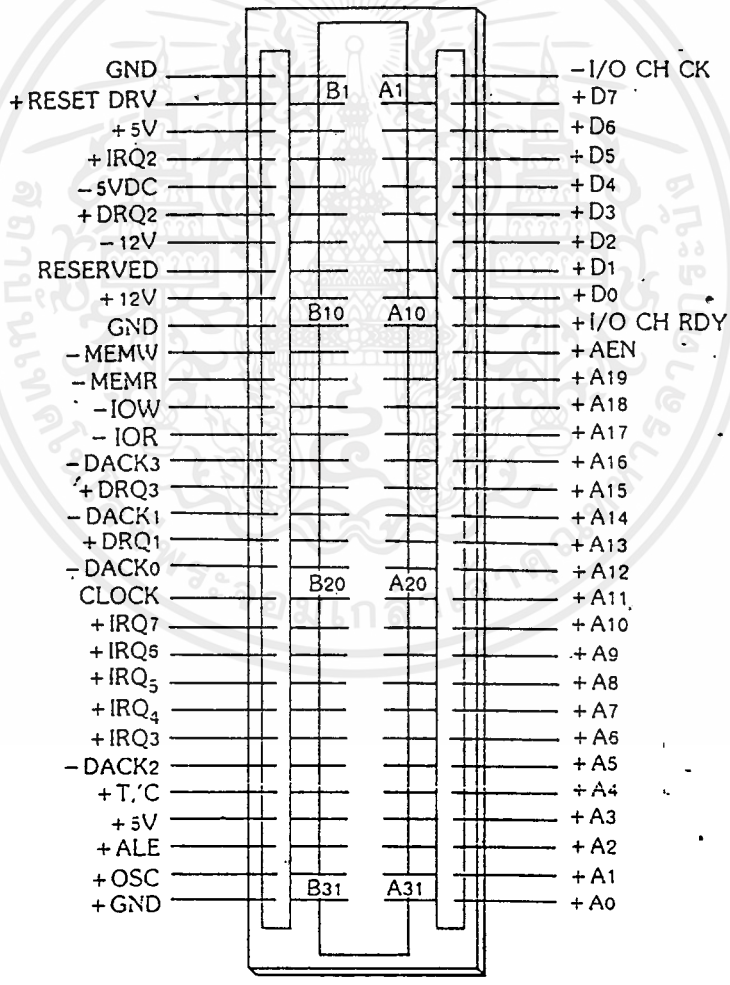
3.1.2 ส่วนของ interface card

ในการนำข้อมูลเข้า (input data) และการส่งข้อมูลออก (output data) ต้องกระทำผ่านทาง interface card ซึ่งเสียบลงบน I/O slot ของคอมพิวเตอร์ ดังนั้นก่อนอื่นเราควรทราบรายละเอียดเกี่ยวกับ I/O slot ก่อน

3.3 สัญญาณต่าง ๆ บน slot ของ IBM PC

ภายใน IBM PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเตอร์เฟสเข้าไปในในภายหลังได้ โดยผ่านสล๊อตที่อยู่บนเมนบอร์ด (Main Board) สำหรับสล๊อตบนเมนบอร์ดจะมีจำนวน 5 สล๊อตซึ่งแต่ละสล๊อตจะมีจำนวน 62 ขา แบ่งเป็น 2 ข้าง ๆ ละ 31 ขา ส่วนการเรียกตำแหน่งขาของสล๊อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใดของสล๊อคโดยขาที่อยู่ทางด้านซ้ายของสล๊อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขา ส่วนทางด้านขวาของสล๊อตเรียกโดยใช้อักษร " A" นำหน้าเลขตำแหน่งของขา

สำหรับขาต่าง ๆ บนสล๊อตที่ต่อเชื่อมสัญญาณต่าง ๆ บนเมนบอร์ด แสดงดังรูปที่ 3.1 และจะแบ่งสัญญาณต่าง ๆที่ IBM PC จัดมาให้เป็นกลุ่มดังนี้



รูปที่ 3.1 ขาสัญญาณต่าง ๆ ใน I/O slot บน IBM PC

ลักษณะของการจัดแบ่งกลุ่มขาสัญญาณ

1.สัญญาณนาฬิกา ประกอบด้วย	- ขา OSC (Osillator)
	- ขา CLK (Clock)
2.Adress Bus ประกอบด้วย	- ขา A0-A19
3.Data Bus ประกอบด้วย	- ขา D0-D7
4.สัญญาณขบวนการอินเตอร์รัพท์	- ขา IRQ2-IRQ7
(Interrupt Request 2-7)	
5.สัญญาณขบวนการ DMA ประกอบด้วย	- ขา DRQ1-DRQ3
	(DMA Request 1-3)
	- ขา DACK0-DACK3
	(DMA Acknowledge0-3)
	- ขา AEN (Adress Enable)
	- ขา T/C (Terminal Count)
6.สัญญาณการรีเซ็ต ประกอบด้วย	- ขา Reset DRV
7.สัญญาณควบคุมการใช้บัส ประกอบด้วย	- ขา IOR(I/O Read)
	- ขา IOW(I/O Write)
	-ขาMEMW(Memory Write)
	- ขา MEMR (Memory Read)
8.สัญญาณการติดต่อ I/O ประกอบด้วย	- ขา I/O CHCK
	(I/O Channel Check)
	- ขา I/O CHRDY
	(I/O Channel Ready)
9.สัญญาณไฟเลี้ยงของระบบ ประกอบด้วย	- ขา +5 ,-5 Vdc
	- ขา +12 ,-12 Vdc
	- ขา GND

สำหรับการจัดสัญญาณบนสล๊อตของ IBM PC /XT นั้นจะมีจำนวนสล๊อตบนเมนบอร์ดเพิ่มขึ้นเป็น 8 สล๊อตแต่การจัดสัญญาณต่าง ๆ ในทั้ง 8 สล๊อตจะยังคงเหมือนกับ IBM PC เพียงแต่สัญญาณต่าง ๆ ที่ส่งออกมายังขาของสล๊อตที่ 8 นั้นจะถูกต่อผ่านวงจรขับกระแส (Buffer) ก่อนและในสล๊อตที่ 8 นี้ขา B8 จะถูกใช้งานด้วย โดยจะถูกใช้เป็นขา CARD SLCTD (Card Selected) ซึ่งขาสัญญาณนี้จะเป็นขาสัญญาณอินพุทจากวงจรภายนอกที่เสียบอยู่บนสล๊อตที่ 8 เพื่อบอกให้วงจรบนเมนบอร์ดทราบ

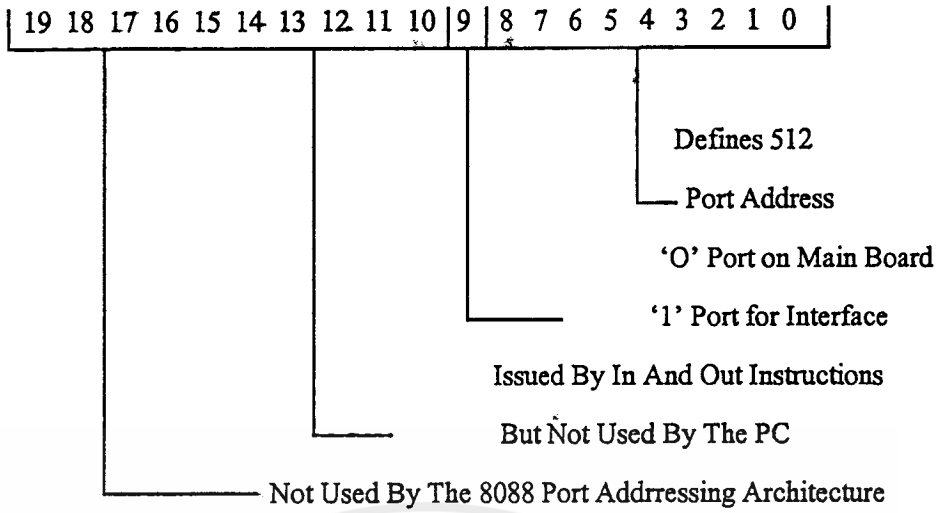
ว่าการ์ดที่อยู่บนสล๊อตนี้ถูกใช้งานอยู่ ซึ่งจะทำให้ไดรเวอร์(Driver) บนเมนบอร์ดทำการอ่านหรือส่งข้อมูลไปยังสล๊อตที่ 8

3.4 การจัดแอดเดรสสำหรับ I/O

ในการควบคุมและตรวจสอบสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิพซัพพอร์ท หรือการ์ดต่าง ๆที่ใช้ในระบบของ IBM PC นั้น จะกระทำโดยผ่านพอร์ท I/O ของระบบดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกรณ์เหล่านี้ จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ท I/O ต่างๆ ของระบบและศึกษาถึงการอ้างแอดเดรสด้วย สำหรับแอดเดรสของพอร์ท I/O ต่าง ๆ นั้นจะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 ซึ่งแอดเดรสเหล่านี้เป็นแอดเดรสที่จัดไว้สำหรับพอร์ท I/O โดยเฉพาะคือแยกออกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ส่วนการส่งข้อมูลให้พอร์ทเหล่านี้ จะทำโดยการใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลไปยังพอร์ทที่ต้องการ และสำหรับการตรวจสอบหรือการอ่านข้อมูลจากพอร์ทก็จะทำได้โดยใช้คำสั่ง IN ของ 8088 อ่านข้อมูลจากแอดเดรสของพอร์ทที่ต้องการ

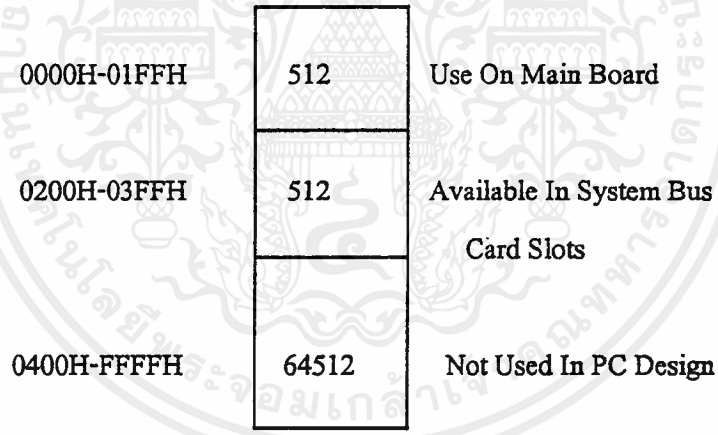
ภายใน 8088 นี้จะมีแอดเดรสสำหรับใช้กับพอร์ท I/O อยู่ทั้งหมด 65,536หรือ 64 K แอดเดรส (ในขณะที่มีแอดเดรสสำหรับหน่วยความจำอยู่ 1 Mbyte) ซึ่งทำให้การอ้างแอดเดรสของพอร์ท I/O ที่ทำงานร่วมกับ 8088 นั้น ต้องใช้จำนวนแอดเดรส 16 เส้น คือ A0-A15 แต่สำหรับใน IBM PC นี้จะถูกออกแบบมาให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่างคือ A0-A9 เท่านั้น ดังนั้นในการดีโค๊ดแอดเดรสในการใช้พอร์ทบน IBM PC โดยใช้แอดเดรสจึงใช้เส้นแอดเดรส A0-A9 เท่านั้นเองดังนั้นเมื่อมีการอ้างพอร์ทต่างๆในทางซอฟแวร์ เส้นแอดเดรส 8 บิทบนจึงไม่มีผลต่อการอ้างพอร์ท คือถือว่าเป็นพอร์ทเดียวกัน

เนื่องจากใน IBM PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้น ดังนั้นจึงสามารถอ้างแอดเดรสของพอร์ทได้สูงสุดเพียง 1024 พอร์ทเท่านั้น นอกจากนี้ใน IBM PC ยังได้แบ่งการทำงานของพอร์ท 1024 พอร์ทนี้ออกเป็น 2 ส่วน คือถ้าข้อมูลในบิท A9 เป็น "0" แล้วจะเป็นพอร์ทที่ใช้งานสำหรับบนเมนบอร์ดของ IBM PC และถ้าข้อมูลในบิท A9 เป็น "1" จะเป็นพอร์ทที่สามารถถูกใช้จากการ์ดต่างๆ ได้



รูปที่ 3.2 การใช้แอดเดรสบิทต่าง ๆ ในการอ้างแอดเดรสของพอร์ทใน IBM PC

3.5 การใช้งานแอดเดรสสำหรับพอร์ท I/O ใน IBM PC



รูปที่ 3.3 การใช้งานแอดเดรสของพอร์ทบน IBM PC

3.5.1 กลุ่มที่หนึ่ง กลุ่มนี้เป็นกลุ่มของพอร์ท I/O ที่อยู่บนเมนบอร์ดของ IBM PC ซึ่งจะมีแอดเดรสอยู่ในตำแหน่ง 0000H จนถึง 01FFH (โดยใช้ A10-A15 เป็น "0") หรือ แอดเดรสที่บิท A9 เป็น "0" นั่นเอง

สำหรับแอดเดรสพอร์ทในกลุ่มนี้จะถูกใช้ในการอ้างแอดเดรสพอร์ทของชิพชัพพอร์ท และอุปกรณ์ที่เป็น I/O ต่าง ๆ ที่อยู่บน Main Board ของ IBM PC

0000H-001FH	32	0000H-000FH (16)	DMA CHIP
0020H-003FH	32	0020H-0021H (2)	INTERRUPT CHIP
0040H-005FH	32	0040H-0043H (4)	TIMER COUNTER CHIP
0060H-007FH	32	0060H-0063H (4)	PPI CHIP
0080H-009FH	32	0080H-0083H (4)	DMA PAGE REISTER
00A0H-00BFH	32	00A0H (1)	NMI MASK BIT
00C0H-01FFH	320	NOT DECODED OR USED ON THE BASEBOARD	

รูปที่ 3.4 การใช้งานแอดเดรสต่างๆ สำหรับพอร์ท I/O ของ IBM PC

จากรูปที่ 3.4 จะพบว่าแอดเดรส 00C0H จนถึงแอดเดรส 01FFH นั้นไม่ได้ถูกใช้งานบนเมนบอร์ดของ IBM PC ดังนั้นในกรณีนี้เราก็สามารถที่จะใช้งานแอดเดรสต่าง ๆ เหล่านี้ได้ แต่อย่างไรก็ตามแอดเดรสเหล่านี้ยังคงถูกดีโค้ดให้เป็นแอดเดรสที่ใช้ในการอ่านข้อมูลจากพอร์ท I/O บนเมนบอร์ดเท่านั้นดังนั้นการใช้ค่าแอดเดรส 00C0H-01FFH กับพอร์ท I/O บนการ์ดหรือวงจรอินเตอร์เฟสที่เราสร้างขึ้นนั้นต้องเป็นพอร์ทเอาท์พุทเพียงชนิดเดียวเท่านั้น

3.5.2 กลุ่มที่สอง กลุ่มนี้จะเป็นกลุ่มของพอร์ท I/O ที่ถูกใช้งานอยู่บนการ์ดที่ใช้เสียบบนสล็อตต่าง ๆ ของ IBM PC สำหรับแอดเดรสของพอร์ทเหล่านี้จะเริ่มต้นจากแอดเดรส 0200H จนถึง 03FFH ซึ่งก็คือแอดเดรสที่มีบิท A9 เป็น "1" นั่นเอง สำหรับการใช้งานแอดเดรสของพอร์ท I/O

0200H	1	NOT USED
0201H	1	COLTROL ADAPTER
0202H-0277H	118	NOT USED
0278H-027FH	8	SECOND PRINTER PORT ADAPTER
0280H-0287H	120	NOT USED
0288H-028FH	8	SECOND SERIAL PORT ADAPTER CARD
0300H-0377H	120	NOT USED
0378H-037FH	8	PRINTER PORT ADAPTER CARD
0380H-03AFH	48	NOT USED
03B0H-03BFH	16	MONOCROME AND PRINTER ADAPTER
03C0H-03CFH	16	NOT USED
03D0H-03DFH	16	COLOUR GRAPHICS ADAPTER
03E0H-03EFH	16	NOT USED
03F0H-03F7H	8	DISKDRIVE ADAPTER CARD
03F8H-03FFH	8	SERIAL PORT ADAPTER CARD

รูปที่ 3.5 การใช้งานแอดเดรสสำหรับพอร์ท I/O

บทที่ 4

การออกแบบวงจร

4.1 คุณสมบัติของเครื่องจ่ายไฟ

ในการออกแบบแหล่งจ่ายไฟกระแสตรงนั้นในขั้นแรกเราจะต้องรู้ถึงคุณสมบัติของแหล่งจ่ายไฟกระแสตรงนั้นเสียก่อน จากโครงงานนี้ เราต้องการออกแบบแหล่งจ่ายไฟกระแสตรงที่มีคุณสมบัติดังต่อไปนี้

1. จ่ายศักดาเอาท์พุทไฟบวก 0 ถึง +25 โวลท์
2. ควบคุมแรงดันด้วยสัญญาณดิจิตอล 8 บิท
3. แสดงค่าแรงดันเป็นระบบตัวเลข Segment Display
4. มีแรงดันริปเปิลต่ำกว่า 2 โวลท์

4.2 หลักการออกแบบเครื่องจ่ายไฟกระแสตรง

1. เลือกว่าจะใช้วงจรเรียงกระแสแบบใด
2. คำนวณขนาดของหม้อแปลง
3. คำนวณขนาดของไดโอดที่ใช้ในวงจรเรียงกระแส
4. คำนวณขนาดของตัวเก็บประจุในวงจรกรองกระแส
5. ออกแบบวงจร D/A Convertor

4.3 การออกแบบส่วนจ่ายไฟ 0 -25 โวลท์

4.3.1 คำนวณขนาดของหม้อแปลง

คุณสมบัติที่ต้องการคือ $V_{DC} = 25\ V$

$I_{DC} = 2\ A$

$V_r = 2\ V$

จาก $V_{AC(rms)} = \dfrac{V_{DC} + 0.5V_r}{\sqrt{2}}$

$= \dfrac{25 + (0.5)(2)}{\sqrt{2}}$

$$=18.38 \text{ V}$$

จาก $$I_{AC\,(RMS)} = \frac{V_{DC} \times I_{DC}}{V_{AC(rms))}}$$

$$= \frac{(25)(2)}{22}$$

$$=2.72 \text{ A}$$

ในการใช้งานจริง จะเผื่อค่าแรงดันและกระแสให้สูงขึ้นอีกเล็กน้อย ดังนั้นเราจะใช้หม้อแปลงที่มีขนาด 0 - 25 โวลท์ 3 แอมป์

4.3.2 คำนวณขนาดของไดโอดที่ใช้ในวงจรเรียงกระแส

ในกรณีนี้จะใช้วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์

กระแสสูงสุดที่ไดโอดจะทนได้ $= \frac{\pi}{2} \times I_{DC}$

$= 1.57 \times 2\text{A}$

$= 3.14\text{A}$

ค่า PIV ของไดโอด $\geq 2\ V_p$

$\geq 2 \times \frac{\pi}{2} \times V_{DC}$

$\geq 3.14 \times 25 \text{ V}$

$\geq 78.5 \text{ V}$

ดังนั้นจึงเลือกใช้บริดจ์ไดโอดที่ทนแรงดันได้ 200 โวลท์ 4 แอมป์

4.3.3 คำนวณค่าของตัวเก็บประจุที่ใช้ในวงจรกรองกระแส

กำหนดค่า ripple $V_r = 2 \text{ V}$

ความถี่ $f_r = 100 \text{ H}_z$ (Full wave)

กระแส $I_{DC} = 2 \text{ A}$

จาก $V_r = \frac{I_{DC}}{frC}$

ดังนั้น $C = \frac{I_{DC}}{frVr}$

$= \frac{2}{100 \times 2}$

= 10,000 μF

อัตราทนแรงดันของตัวเก็บประจุ =1.25 Vp

= $1.25 \times \sqrt{2} \times V_{AC(rms)}$

= $1.25 \times \sqrt{2} \times 18.38$

= 32.49 V

ดังนั้นจึงเลือกใช้ตัวเก็บประจุขนาด 4,700 μF, 50 V จำนวน 2 ตัวต่อขนานกัน

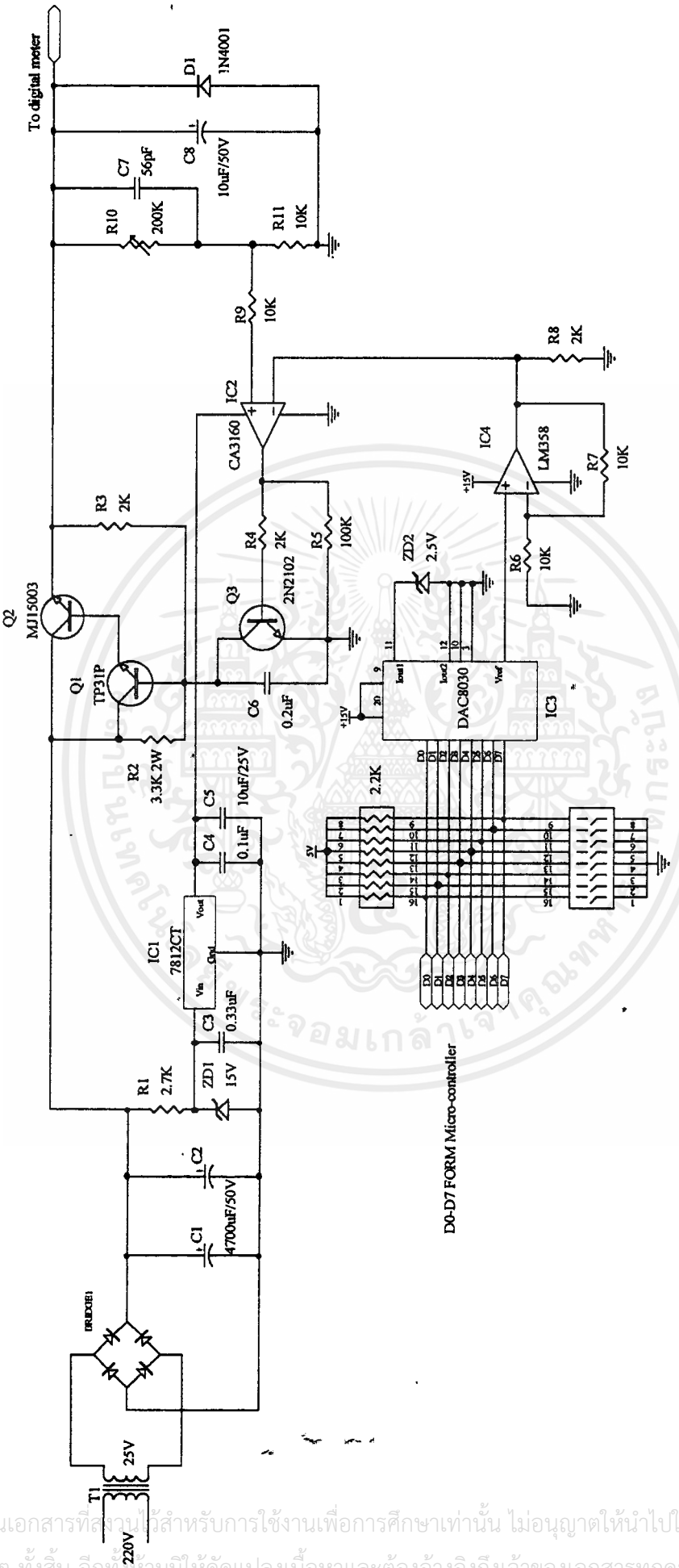
4.4 อธิบายการทำงานของวงจรรักษาค่าระดับแรงดัน 0 ถึง 25 โวลท์

จากวงจรดังรูปที่ 4.1 จะขออธิบายการทำงานดังต่อไปนี้

ทรานซิสเตอร์ Q_1 และ Q_2 ต่อกันแบบดาร์ลิงตัน ซึ่งจะทำหน้าที่เป็นแหล่งจ่ายกระแสของวงจร ตัวเก็บประจุ C_6 เป็นตัวป้องกันไฟกระชากในขณะที่เปิดเครื่อง ความต้านทาน R_2 ทำหน้าที่จำกัดกระแสเบสที่ไปไบอัสให้กับ Q_1 โดยมี R_3 ทำหน้าที่แบ่งกระแส เพื่อป้องกันไม่ให้ Q_2 เสียหาย ทรานซิสเตอร์ Q_2 จะทำงานก็ต่อเมื่อมีกระแสจากขาอิมิตเตอร์ของ Q_1 มาไบอัส ค่าอัตราการขยายรวมของแหล่งจ่ายกระแสมีค่าเท่ากับค่า β ของ Q_1 และ Q_2

ที่ออปแอมป์ IC_4 ต่อในลักษณะวงจรโวลท์เตจฟอลโลเวอร์แบบมีเกน จากวงจรออปแอมป์จะพยายามทำให้แรงดันที่ขาอินเวอร์ติ้ง และที่ขานอนอินเวอร์ติ้งมีแรงดันเท่ากัน โดยที่ขา 2 จะมี แรงดันเปรียบเทียบขนาดสูงสุด 2 โวลท์ ที่ได้จากวงจรแปลงสัญญาณ Digital to Analog และที่จาก ขา 3 จะเป็นขาที่นำแรงดันเอาท์พุทเข้ามาเปรียบเทียบ โดยผ่านวงจรลดทอนแรงดันให้มีขนาดที่ เหมาะสม โดยผ่าน R_{10} และ R_{11} ต่ออยู่ในลักษณะโวลท์เตจดิไวร์เดอร์ ตัวเก็บประจุ C_7 จะช่วยทำให้แรงดันที่ตกคร่อม R_{10} มีค่าคงที่

ที่ขา 3 ของออปแอมป์ จะมีการเอาแรงดันทางเอาท์พุทมาเปรียบเทียบกับระดับแรงดันอ้างอิงที่ขา 2 เมื่อระดับแรงดันที่ขา 3 ไม่เท่ากับที่ขา 2 มีผลทำให้ออปแอมป์พยายามที่จะให้แรงดันที่ขา 3 และขา 2 มีค่าเท่ากันทำให้เกิดการเปลี่ยนแปลงของกระแสที่ไหลออกจากขา 6 ของออปแอมป์กระแสดังกล่าวนี้จะไปไบอัสให้กับทรานซิสเตอร์ Q3 ทำงาน ทำให้แรงดันที่ไบอัสให้กับ Q2 เกิดการเปลี่ยนแปลงจนกว่าแรงดันที่ขา 3 ของออปแอมป์ จะมีค่าเท่ากับระดับแรงดันอ้างอิงที่ขา 2 ซึ่งเราสามารถที่จะหาค่าความต้านทาน ที่ต่อเป็นวงจรโวลท์เตจดิไวเดอร์ ได้ดังนี้



รูปที่ 4.1 แสดงแหล่งจ่ายไฟกระแสตรง 0-25 โวลท์

$$V_o = -V_{reff}\left(\frac{R_f + R_i}{R_i}\right)$$

เมื่อ

V_o = ค่าระดับแรงดันเอ้าท์พุท

Ri = R_{10}

Rf = R_{11}

ตัวเก็บประจุ C_8 ที่เอาท์พุทนั้นมีไว้เพื่อลดแรงดันทรานเซียนต์ และไดโอด D_1 มีไว้เพื่อป้องกันแรงดันย้อนกลับมาที่เอาท์พุท ที่ขา 7 ของออปแอมป์ต่อเข้าไฟเลี้ยง 12 โวลท์ซึ่งได้มาจากวงจรเรคกูแรคเตอร์ที่ใช้ IC 7812 ตัวเก็บประจุ C_5 จะทำหน้าที่ปรับปรุงระดับแรงดันทางเอาท์พุทให้ราบเรียบยิ่งขึ้น

4.5 การทำงานของวงจร DIGITAL METER

หัวใจในการทำงานของวงจรนี้คือไอซี CA3162E ซึ่งทำหน้าที่เป็นตัวแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอล ในแบบ DUAL-SLOPE A/D Conversion

เอาท์พุทที่ได้ของไอซีจะเป็นแบบมัลติเพล็กซ์ไบนารี่โค้ดดิสเพลย์ ทำให้ง่ายต่อการประกอบเป็นอย่างมากเนื่องจากใช้สายต่อไปยังดิสเพลย์น้อยลง

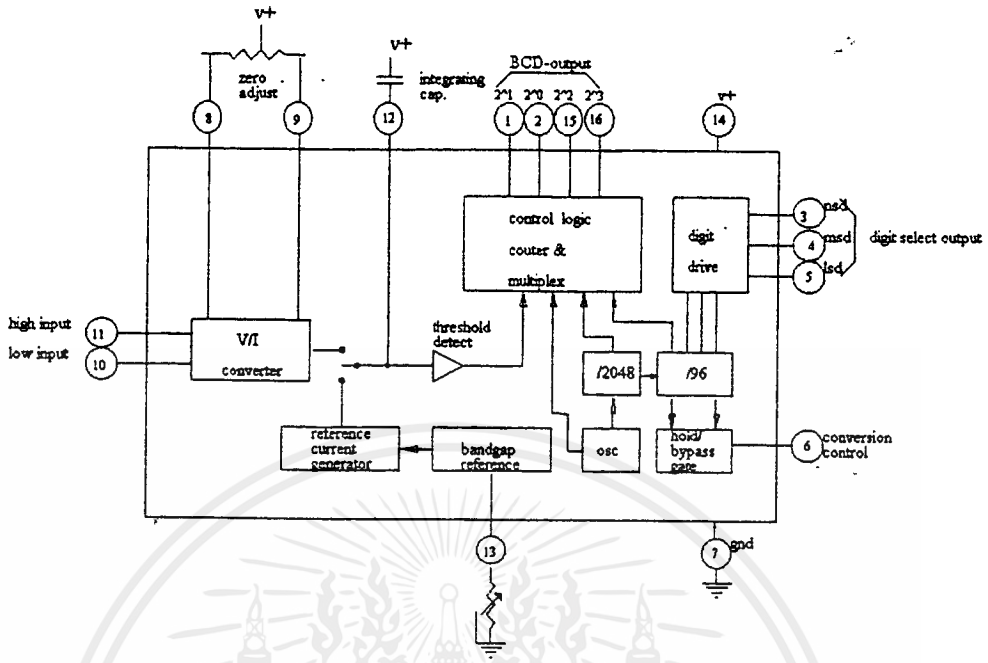
วงจรนี้มีคุณสมบัติคือ

- แสดงผลเป็นตัวเลข 3 หลัก
- แสดงผลสูงสุด + 999 mV และค้านลบได้สูงสุด - 99 mV
- แรงดันไฟเลี้ยงวงจร + 5 V เพียงชุดเดียว

ในรูปที่ 4.2 เป็นบล็อคไดอะแกรมของไอซี CA3162E หัวใจในการทำงานของวงจรคือวงจร V/I Converter And Reference - Current Generator

วงจร V/I Converter จะแปลงแรงไฟที่ป้อนเข้ามาระหว่างขา 10 และขา 11 ให้เป็นกระแสเพื่อชาร์จอินติเกรติ้งคาปาซิเตอร์ที่ขา 12 เป็นเวลาตามที่กำหนด

ในตอนท้ายของช่วงเวลาที่ชาร์จวงจร V/I Converter จะถูกตัดออกจากอินติเกรติ้งคาปาซิเตอร์ โดยวงจรจะต่อวงจรจ่ายกระแสคงที่ที่มีขั้วตรงข้ามเข้ามาแทนที่ จำนวนของคล็อคที่นับได้ในช่วงเวลาก่อนที่ประจุในคาปาซิเตอร์จะกลับคืนสภาพเดิมนั้นจะเป็นค่าที่สัมพันธ์กับสัญญาณอินพุท



รูปที่ 4.2 แสดงบล็อคไดอะแกรมของไอซี CA3162E

การคืนสภาพจะถูกตรวจสอบโดยวงจรคอมพาราเตอร์ซึ่งทำหน้าที่ Latch วงจรเคาเตอร์เอาท์พุทของวงจรเคาท์เตอร์จะถูกป้อนให้กับวงจรไบนารี่โค้ดในแบบมัลติเพล็กซ์

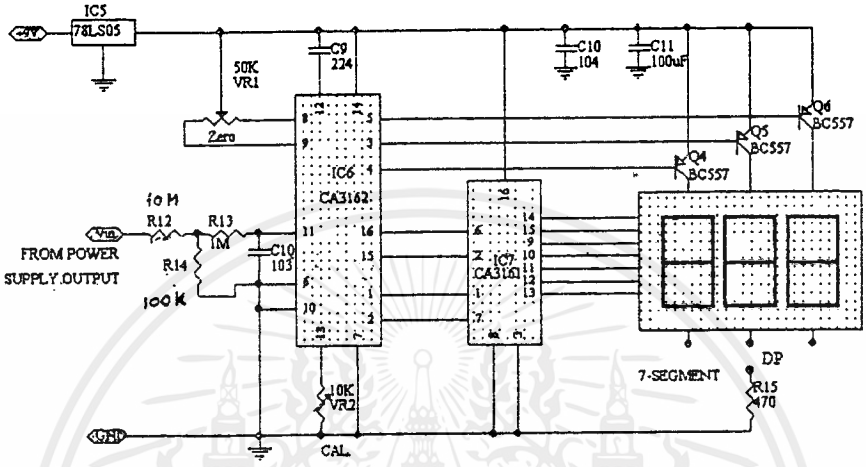
วงจรคล็อคจะเกิดจากวงจรออสซิลเลเตอร์ที่ทำความถี่ 786 เฮิร์ท อินพุทที่ขา 6 จะเป็นตัวกำหนดอัตราการ Sampling ถ้าแรงไฟที่ขา 6 เป็น +5 Vจะทำให้ Sampling Rate สูง (96 Hz) ถ้าแรงไฟที่ขา 6 เป็นกราวด์หรือปล่อยลอยไว้จะทำให้ Sampling Rate ต่ำ (4 Hz) ถ้าแรงไฟที่ขา 6 เท่ากับ 1.2 V โดยการต่อรีซิสเตอร์ 12KΩ ระหว่างขา 6 กับ ไฟ +5 V วงจรจะโฮลด์ (Hold) ในขณะที่วงจรโฮลด์ การ Sampling จะมีอัตรา 4 เฮิร์ท แต่ข้อมูลที่แสดงจะถูกโฮลด์ให้แสดงที่ค่าสุดท้าย

สำหรับการวัดสัญญาณอินพุทสามารถวัดแรงดันไฟตรงที่มีแรงดันไม่เกิน 1 โวลท์ แต่เราต้องการวัดแรงดันที่สูงกว่า 1 โวลท์ ให้ป้อนอินพุทเข้าที่จุด Vin และเปลี่ยนแปลงค่า R_{12} และ R_{14} ตามสูตรคือ

$$R14 = \frac{10{,}000{,}000}{Ei - 1}$$

ในที่นี้เราต้องการย่านวัดสูงสุด 99.9 โวลท์ ก็ใช้ค่า R_{13} =10 MΩ และ R_{14} =100KΩ เมื่อเราให้ค่า R_{13} คงที่

C_9 จะทำหน้าที่ อินติเกรติ้งคาปาซิเตอร์ ส่วนไอซี CA3161 ทำหน้าที่เป็นตัวไบนารี่ดีโคดเดอร์ ส่งค่าเอาท์พุทให้กับ 7- segment



รูปที่ 4.3 แสดงวงจรของดิจิตอลมิเตอร์

4.6 การออกแบบการ์ดอินเตอร์เฟส

การ์ดอินเตอร์เฟส GPIB นี้เป็นการ์ดที่ใช้เสียบในสล๊อตคอมพิวเตอร์ทั่วๆไป ซึ่งในโครงงานนี้ได้นำเอาไอซีเบอร์ 8255 มาใช้ประยุกต์ใช้ในการออกแบบวงจรร่วมกับไอซีเบอร์ SN75160 และ SN75161 ซึ่งเป็นไอซีที่ออกแบบมาให้ใช้กับระบบบัสของ GPIB โดยเฉพาะซึ่งจะทำหน้าที่เป็นส่วนของวงจรบัฟเฟอร์ของวงจร

4.6.1 การออกแบบส่วนของพอร์ทควบคุม

เราได้ทราบมาแล้วว่าแบบมาตรฐานของ IEEE 488 นั้นประกอบด้วยบัสต่างๆจำนวน 16 เส้น ซึ่งสามารถแบ่งได้ 3 กลุ่มคือ

1. กลุ่มสัญญาณควบคุมการรับส่งข้อมูลประกอบด้วยสัญญาณ DAC , NRFD และ NDAC ซึ่งถูกควบคุมโดยสัญญาณ TE

2. กลุ่มสัญญาณควบคุมการอินเตอร์เฟสประกอบด้วยสัญญาณ ATN,IFC,RENและ SRQ ถูกควบคุมโดยสัญญาณ DC และ EOI ควบคุมได้ทั้งสัญญาณ DC และ TE

3. กลุ่มสัญญาณข้อมูลเป็นเส้นทางผ่านของข้อมูลของระบบจำนวน 8 เส้น

นอกจากนั้นเรายังต้องกำหนดสัญญาณควบคุมของไอซี SN75160 และ SN75161 ด้วย สัญญาณนั้นคือสัญญาณ PE,TE และ DC ตามลำดับ ดังนั้นเราจึงต้องกำหนดพอร์ทสัญญาณทั้งสิ้น จำนวน 4 พอร์ทด้วยกัน และพอร์ทนี้จะต้องเป็นได้ทั้งอินพุทและเอาท์พุทพอร์ทซึ่งตรงกับคุณสมบัติของไอซี 8255 พอดี ซึ่งต้องใช้ไอซี 8255 จำนวน 2 ตัว โดยจะกำหนดการทำงานของ 8255 ดังต่อไปนี้

กำหนดให้ 8255 ทำงานในโหมดศูนย์ กำหนดให้พอร์ท A ของไอซี 8255 ตัวที่ 1 เป็นพอร์ทของบัสข้อมูล 8 บิทโดยกำหนดให้เป็นทั้งอินพุทและเอาท์พุท พอร์ท B ของไอซี 8255 ตัวที่ 1 เป็นพอร์ทที่ใช้สำหรับกำหนดทิศทางของสัญญาณข้อมูลและสัญญาณควบคุมโดยกำหนดให้เป็นเอาท์พุทพอร์ทเท่านั้น สำหรับพอร์ท C ซึ่งจะใช้ทั้งพอร์ท C บนและพอร์ท C ล่าง ซึ่งจะกำหนดให้เป็นทั้งอินพุทและเอาท์พุทพอร์ท โดยใช้พอร์ท C ของ 8255 ทั้ง 2 ตัวทำงานร่วมกันเพื่อความสะดวกในการควบคุมการทำงาน

4.6.2 การจัดกลุ่มของสัญญาณ

การจัดกลุ่มสัญญาณจะพิจารณาจากทิศทางของสัญญาณที่กำหนดไว้ในไอซีเบอร์ SN75160 และ SN75161 ซึ่งทำหน้าที่เป็นบัฟเฟอร์สำหรับการรับส่งข้อมูลและสัญญาณควบคุมสำหรับระบบ GPIB โดยเฉพาะ และพิจารณาถึงความสะดวกในการควบคุมการทำงานของ 8255

กลุ่มสัญญาณต่างๆ

1. DIO_1-DIO_8 ต่อกับพอร์ท A ของ 8255 ตัวที่ 1
2. NDAC, NRFD ต่อกับพอร์ท C บนของ 8255 ตัวที่ 1
3. DAV, EOI ต่อกับพอร์ท C ล่างของ 8255 ตัวที่ 1
4. SRQ ต่อกับพอร์ท C บนของ 8255 ตัวที่ 2
5. REN, IFC, ATN ต่อกับพอร์ท C ล่างของ 8255 ที่ 2

และกลุ่มสัญญาณควบคุมต่างๆ บน 8255 ซึ่งประกอบไปด้วย RD, WR และ RESET สามารถเข้าโดยตรงกับ IBM Slot ของคอมพิวเตอร์ได้เลย เพื่อใช้ในขบวนการอ่านเขียนและรีเซตตามลำดับ

4.6.3 การออกแบบวงจรถอดรหัส

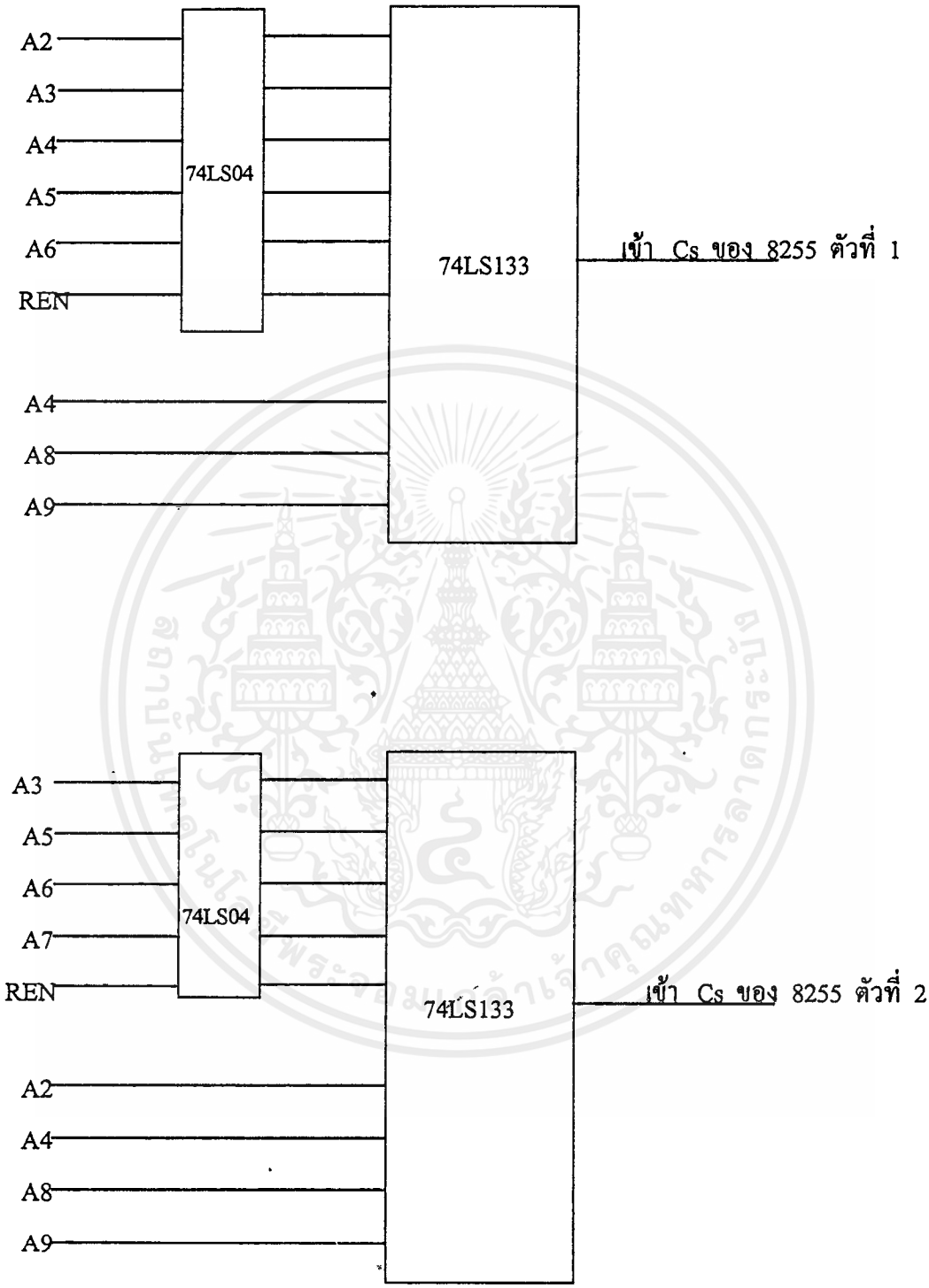
ในการทำโครงงานนี้ได้เลือกใช้พอร์ทแอดเดรสที่ 310H-317H สำหรับการกำหนดตำแหน่งของไอซี 8225 ทั้งสองตัวโดยเรากำหนดให้ แอดเดรสที่ 0310H-0313H เป็นของไอซี 8255 ตัวที่ 1 และแอดเดรสที่ 0314H-0317H เป็นของไอซี 8255 ตัวที่ 2 ซึ่งในการเลือกอุปกรณ์ Gate มา

มาสร้างวงจร Decoder เพื่อ Decode แอดเดรสของไอซี 8255 ทั้งสองตัว สามารถพิจารณาได้จากการแจกแจงค่าในแต่ละบิทของแอดเดรสต่างๆ จากช่องสล๊อตของคอมพิวเตอร์ ดังตารางที่ 4.1

ตารางที่ 4.1 การจัดแอดเดรสที่ใช้ติดต่อกับอุปกรณ์ภายนอกบน IBM PC

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Port Address	Port Name	8255 ตัวที่
1	1	0	0	0	1	0	0	0	0	310H	Port A	1
1	1	0	0	0	1	0	0	0	1	311H	Port B	1
1	1	0	0	0	1	0	0	1	0	312H	Port C	1
1	1	0	0	0	1	0	0	1	1	313H	Control	1
1	1	0	0	0	1	0	1	0	0	314H	Port A	2
1	1	0	0	0	1	0	1	0	1	315H	Port B	2
1	1	0	0	0	1	0	1	1	0	316H	Port C	2
1	1	0	0	0	1	0	1	1	1	317H	Control	2

จากตารางจะเห็นว่าสำหรับ 8255 ทั้งสองตัวนั้นมีเพียง A1 และ A0 เท่านั้นที่เปลี่ยนเมื่อแอดเดรสเปลี่ยน ดังนั้นเราจึงสามารถสร้างวงจร Decede แอดเดรสของ 8255 ทั้งสองตัวได้โดยการนำสัญญาณ A1 และ A0 ไปเข้าไอซีทั้งสองตัว โดยมีสัญญาณเลือกให้ไอซีตัวใดทำงานจากการนำเอาบิทที่เหลือของสัญญาณแอดเดรสนี้ไปเข้าวงจรเกตดังรูปที่ 4.4



รูปที่ 4.4 แสดงการเลือก Decode แอดเดรสของ 8255 ทั้งสองตัว

สำหรับ A0-A1 นั้นต่อเข้าโดยตรงกับ 8255 ทั้ง 2 ตัวเลย เพื่อใช้ถอดรหัสพอร์ทภายในซึ่งหมายเลขพอร์ทแสดงไว้ในตารางด้านบนแล้ว นอกจากนั้นเราจะต้องนำสัญญาณ AEN (Address Enable output) ซึ่งเป็นสัญญาณที่ใช้สำหรับแยกบัสแอดเดรสในการทำขบวนการ DMA เมื่อสัญญาณนี้แอคทีฟจะทำให้ DMA Controller สามารถควบคุมการทำงานแทนการควบคุม CPU มาต่อร่วมอยู่ด้วยเพื่อไม่ให้เกิดการชนกันของข้อมูลเมื่อ CPU ทำขบวนการ DMA นั่นเอง ขบวนการ DMA เป็นการเพิ่มความเร็วของการส่งผ่านข้อมูลระหว่างหน่วยความจำกับอุปกรณ์ I/O ได้ดียิ่งขึ้น

ในส่วนของ 8255 ยังไม่จบเพียงเท่านี้ เรายังต้องส่งรหัสควบคุม (Control byte) เข้าไปยังพอร์ทควบคุมข้อมูล (port สุดท้ายใน 4 พอร์ท คือ ที่ A1 กับ A0 เป็น 1 ซึ่งได้แก่ พอร์ทหมายเลข 313H และ 317H ของ 8255 ทั้งสองตัว) ให้ทำงานในโหมดไหนและให้เป็นอินพุทหรือเอาต์พุทพอร์ท ความหมายของบิทต่างๆของรหัสควบคุม (Control byte) หรือรหัสสั่งงานของ 8255 เป็นดังนี้ คือ

D7 - แสดงถึงรหัสควบคุมให้เริ่มทำงาน (1 คือ งาน) คือจะมีผลทำให้ 8255 รับรู้สิ่งต่อไปนี้ บิทต่างๆ ที่จะกำหนดให้เพราะ ฉะนั้นเวลาสั่งงานหรือสั่งหน้าที่ให้กับ 8255 บิทนี้จะเป็นลอจิก 1 เสมอ

D6, D5 - เป็นการเลือกโหมดการทำงานของพอร์ท A ซึ่งมีอยู่ 3 โหมด ใน 8255 จะได้กล่าวต่อไป

D4 - กำหนดให้พอร์ท A เป็นอินพุทหรือเอาท์พุทโดยที่

0 = Output port

1 = Input port

D3 - กำหนดให้ port C บนเป็นอินพุทหรือเอาท์พุทโดย

0 = Output port

1 = Input port

D2 - เป็นการเลือกโหมดให้กับพอร์ท B

0 = Mode 0

1 = Mode 1

D1 - กำหนดให้พอร์ท B เป็นอินพุทหรือเอาท์พุทโดย

0 = Output port

1 = Input port

D0 - กำหนดให้พอร์ท C ล่างเป็นอินพุทหรือเอาท์พุทพอร์ทโดย

0 = Output port

ซึ่งแสดงได้ดังต่อไปนี้

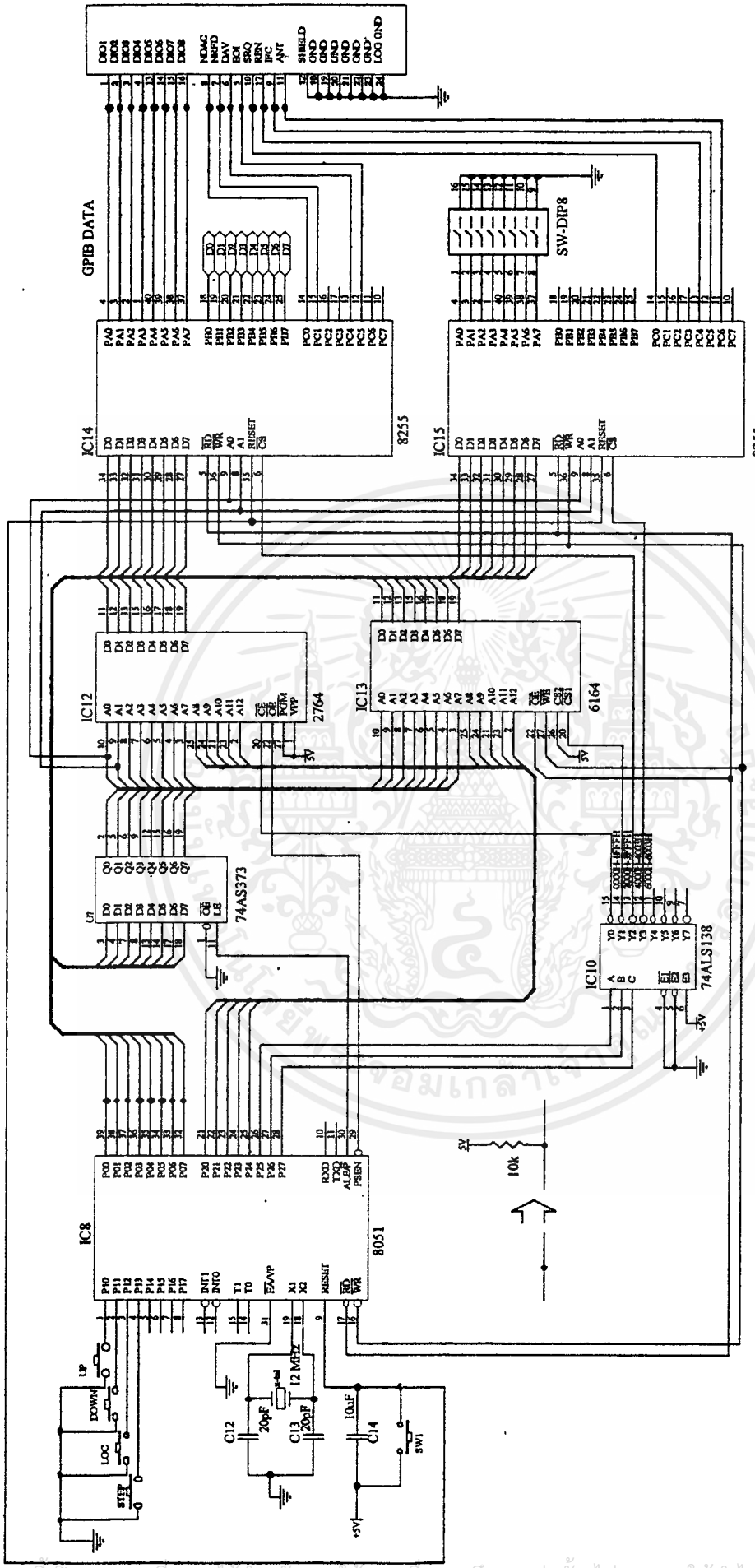
D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	X	X	0	0	X

จะเห็นได้ว่าเรายังไม่สามารถระบุได้แน่ชัดว่ารหัสคำสั่งเป็นอะไรทั้งนี้เพราะพอร์ท A พอร์ท C บนและพอร์ท C ล่าง สามารถเปลี่ยนแปลงได้ 2^3=8 ค่าดังนี้

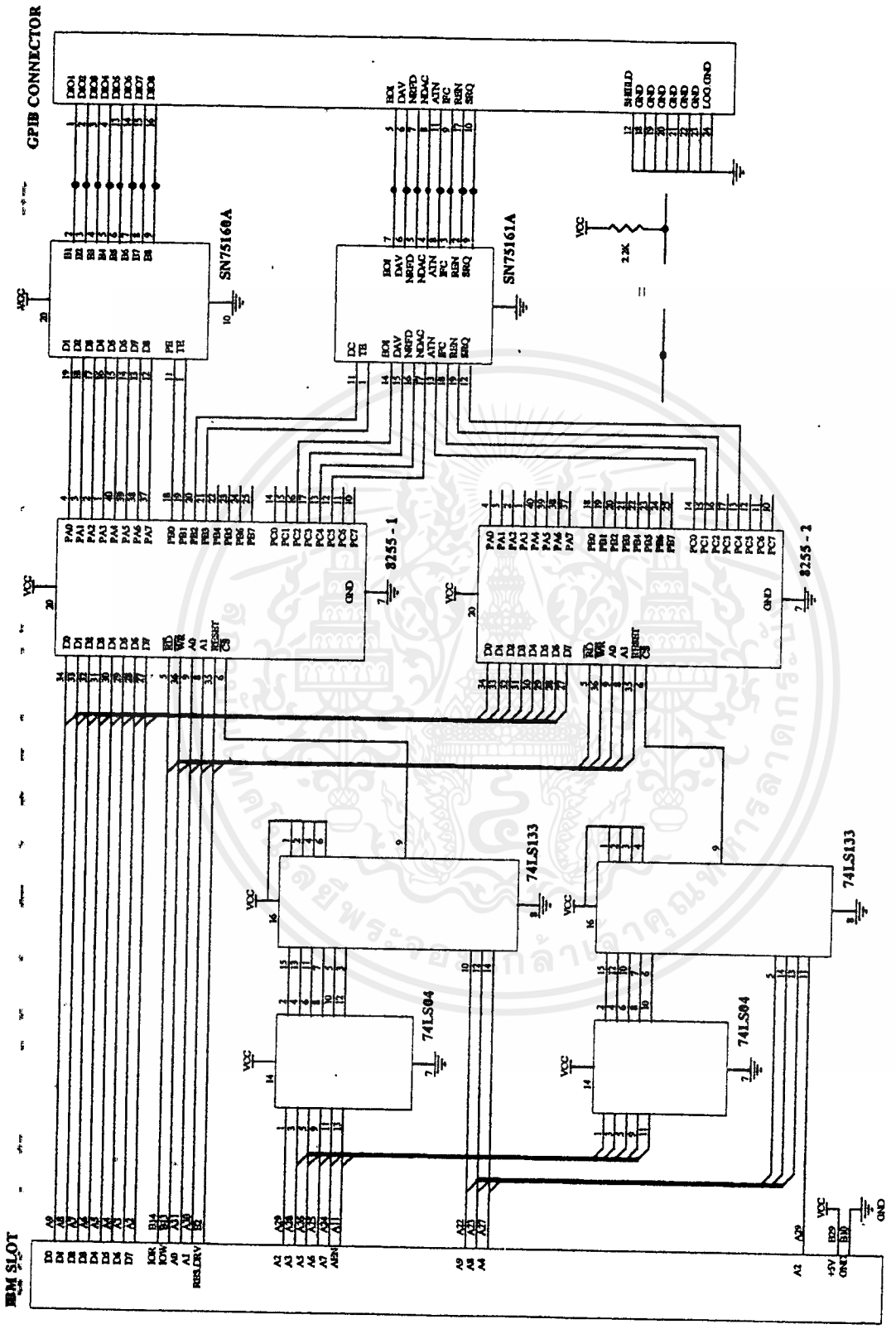
ตารางที่ 4.2 การเซตค่าอินนิเชียลโหมด 0 ของ 8255

A7	A6	A5	A4	A3	A2	A1	A0
1	0	0	x	x	0	0	x
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	0	1	0	0	0	1
1	0	0	1	1	0	0	0
1	0	0	1	1	0	0	1

โดยเราจะได้รหัสควบคุมคือ 80H, 81H, 88H, 89H, 90H, 91H,98H และ 99H ตามลำดับ จากที่ผ่านมาเราสามารถออกแบบ Hardware ของระบบบัสของ GPIB ได้ตามกระบวนการที่ได้กล่าวมาแล้วในตอนต้นทั้งหมด ในตอนต่อไปนี้ก็จะเป็นการรวมวงจรทั้งหมดที่กล่าวถึงมารวมกันเป็นวงจรใช้งานจริงดังรูปที่ 4.5 และ 4.6

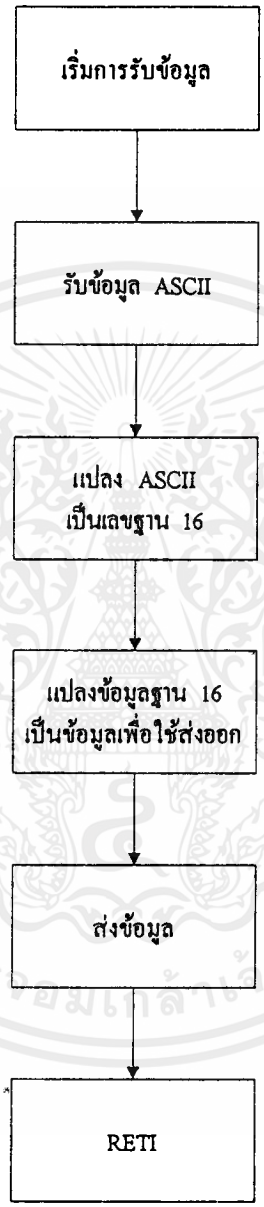


รูปที่ 4.5 แสดงส่วนของวงจร MICRO CONTROLLER

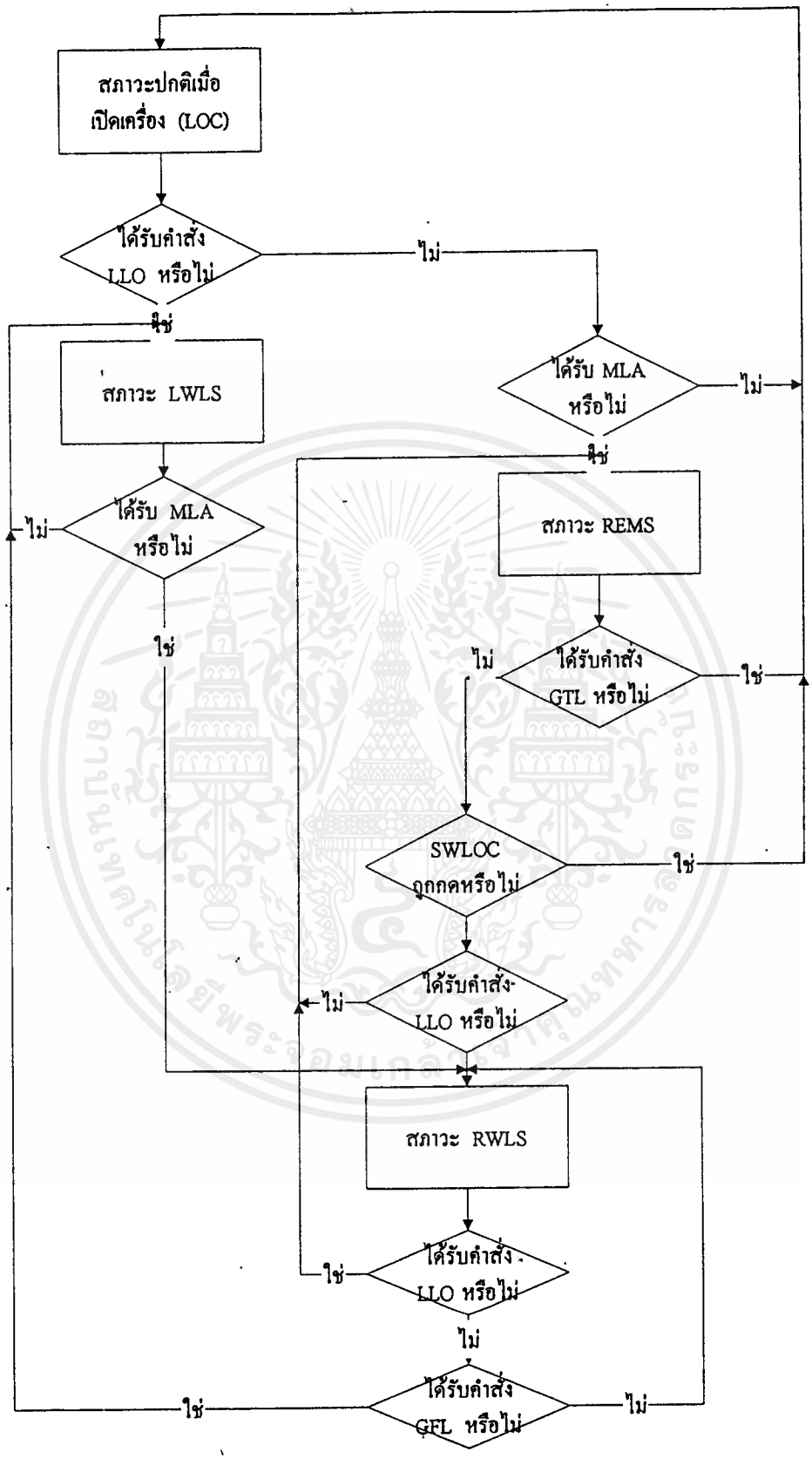


รูปที่ 4.6 แสดงส่วนของวงจร INTERFACE CARD

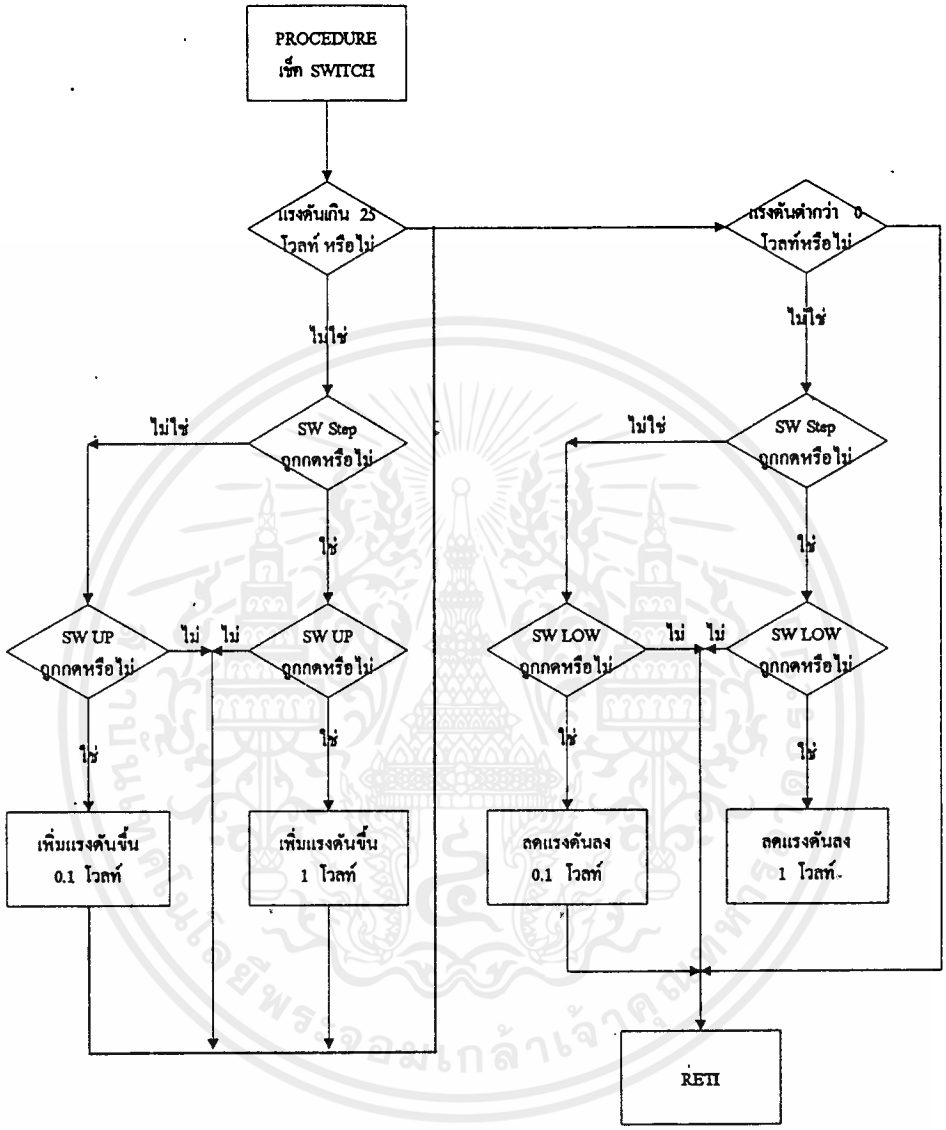
FLOW CHART แสดงการทำงานของ MICRO CONTROLLER



รูปที่ 4.7 แสดง Flow chart การรับส่งข้อมูลของ Micro controller

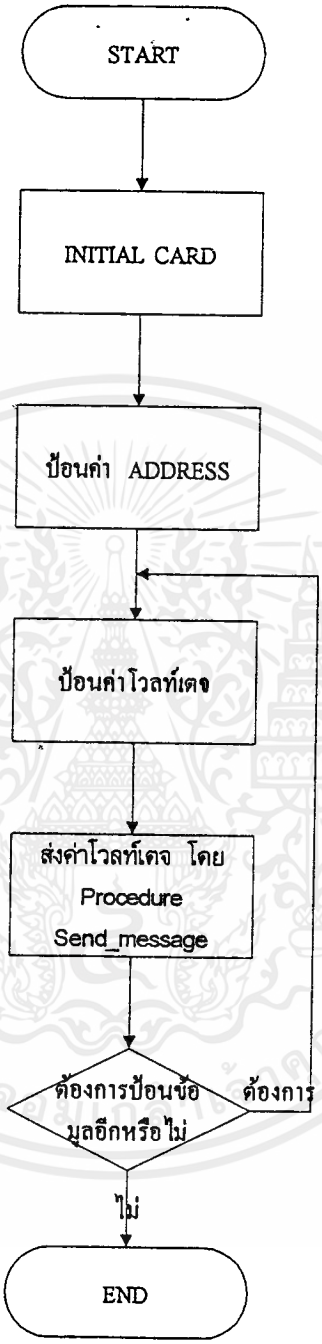


รูปที่ 4.8 แสดง Flow chart สภาวะต่างๆของ Power Supply

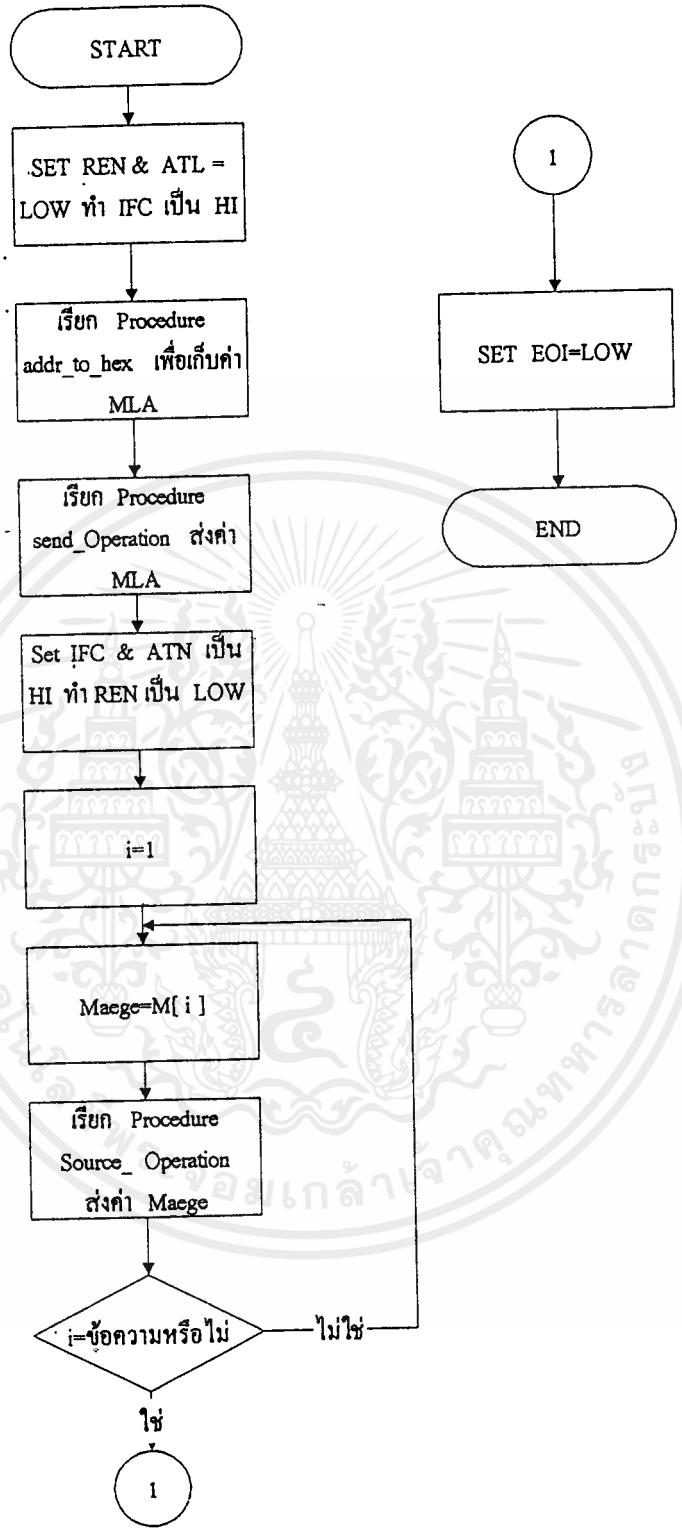


รูปที่ 4.9 แสดง Flow chart การเช็คการกดสวิทช์

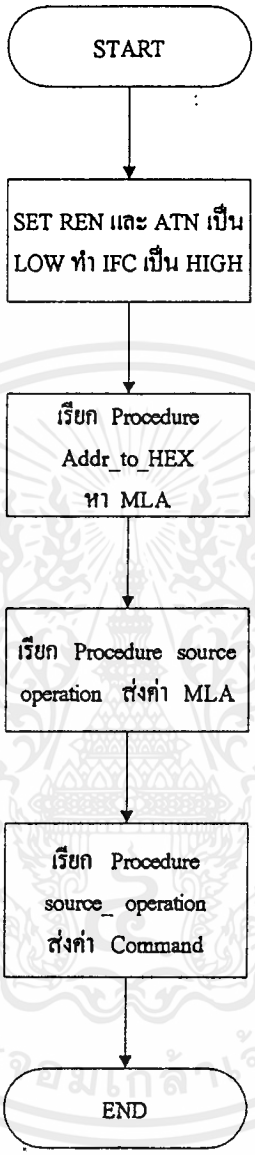
FLOW CHART การทำงานของโปรแกรมการควบคุมการส่งข้อมูล



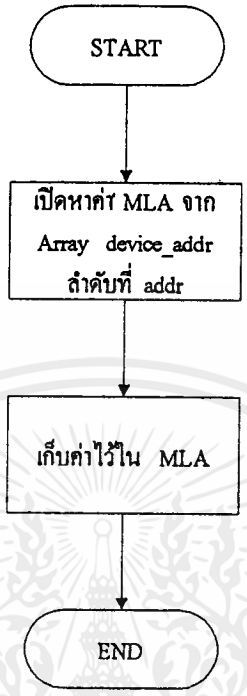
รูปที่ 4.10 แสดง Flow chart Main Program



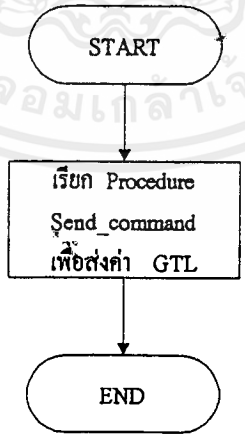
รูปที่ 4.11 แสดง Flow chart Procedure Send_Message



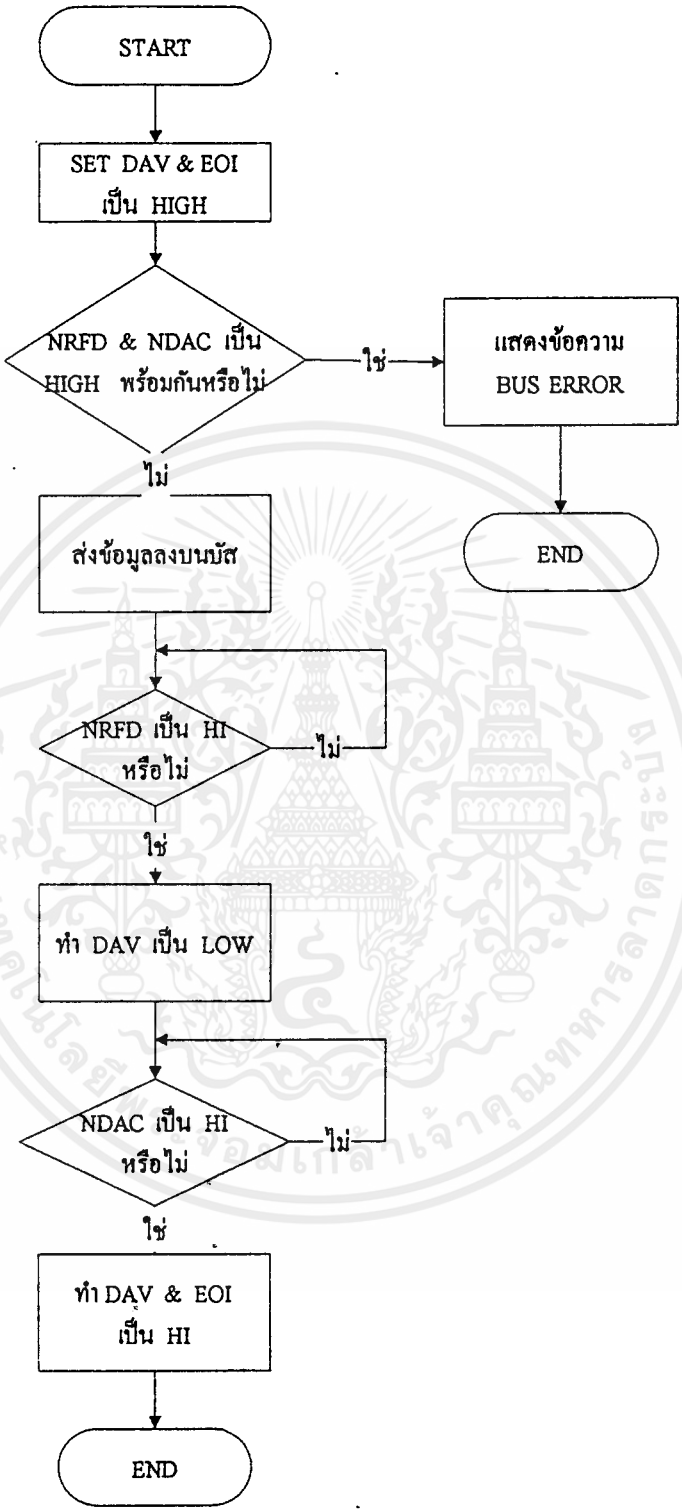
รูปที่ 4.12 แสดง Flow chart Procedure Send_Command



รูปที่ 4.13 แสดง Flow chart Procedure addr_to_Hex



รูปที่ 4.14 แสดง Flow chart ของ GTL



รูปที่ 4.15 แสดง Flow chart Source_Operation

บทที่ 5

การทดลองและผลการทดลอง

การทดสอบแหล่งจ่ายไฟกระแสตรงที่สร้างขึ้นมา โดยการควบคุมผ่านระบบบัสมาตรฐาน IEEE - 488 (GPIB) ผ่านการ์ดอินเตอร์เฟสที่สร้างขึ้นมา โดยใช้โปรแกรมควบคุมผลการทดลองที่ได้สามารถควบคุม Power supply ให้สามารถติดต่อสื่อสารกับคอมพิวเตอร์และสามารถจ่ายแรงดันให้ออก Output ได้โดยการส่งคำสั่งควบคุมจากคอมพิวเตอร์ และเมื่อมีการส่งข้อมูลรหัสคำสั่งโดยคอมพิวเตอร์อยู่นั้น จะไม่สามารถปรับการจ่ายแรงดันได้โดยปุ่มที่อยู่บนหน้าปัทม์ของเครื่อง จะปรับปุ่มนี้ได้จนกว่า Power supply นี้จะได้รับคำสั่งให้กลับไปอยู่ในโหมดของการควบคุมด้วยมือ โดยการส่งมาจากคอมพิวเตอร์

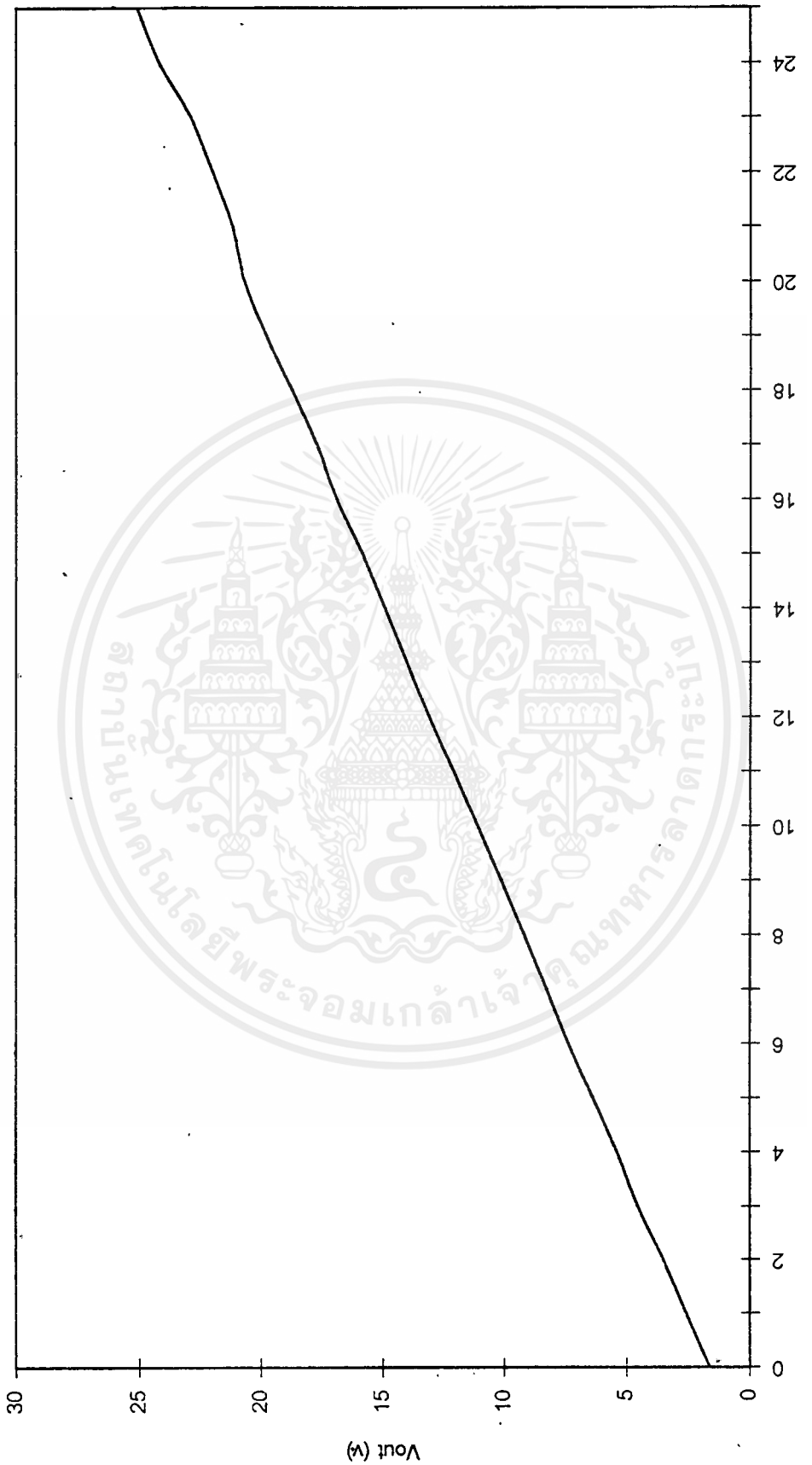
ตารางที่ 5.1 แสดงผลการทดลองการส่งค่ารหัสคำสั่ง

รหัสคำสั่ง	Input code D/A	Vref (V)	output (V)
0.0	0000 0000	0.02	1.63
1.0	0000 1010	0.075	2.58
2.0	0001 0100	0.156	3.51
3.0	0001 1110	0.239	4.53
4.0	0010 1000	0.322	5.39
5.0	0011 0010	0.403	6.36
6.0	0011 1100	0.489	7.36
7.0	0100 0110	0.571	8.26
8.0	0101 0000	0.657	9.20
9.0	0101 1010	0.733	10.14
10.0	0110 0100	0.821	11.09
11.0	0110 1110	0.901	12.09
12.0	0111 1000	0.993	13.09
13.0	1000 0010	1.084	14.03
14.0	1000 1100	1.157	14.92
15.0	1001 0110	1.235	15.84

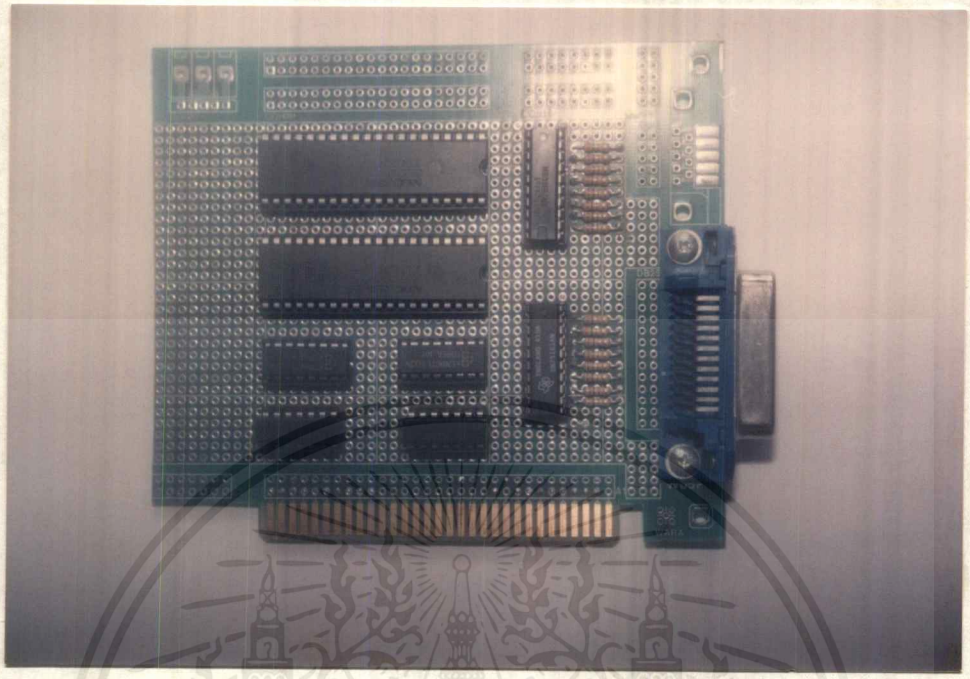
ตารางที่ 5.1 (ต่อ)

รหัสคำสั่ง	Input code D/A	Vref (V)	Voltage output (V)
16.0	1010 0000	1.334	16.90
17.0	1010 1010	1.400	17.73
18.0	1011 0100	1.489	18.75
19.0	1011 1110	1.580	19.80
20.0	1100 1000	1.666	20.7
21.0	1101 0010	1.743	21.2
22.0	1101 1100	1.831	22.0
23.0	1110 0110	1.914	22.9
24.0	1111 0000	2.02	24.2
25.0	1111 1010	2.09	25.1

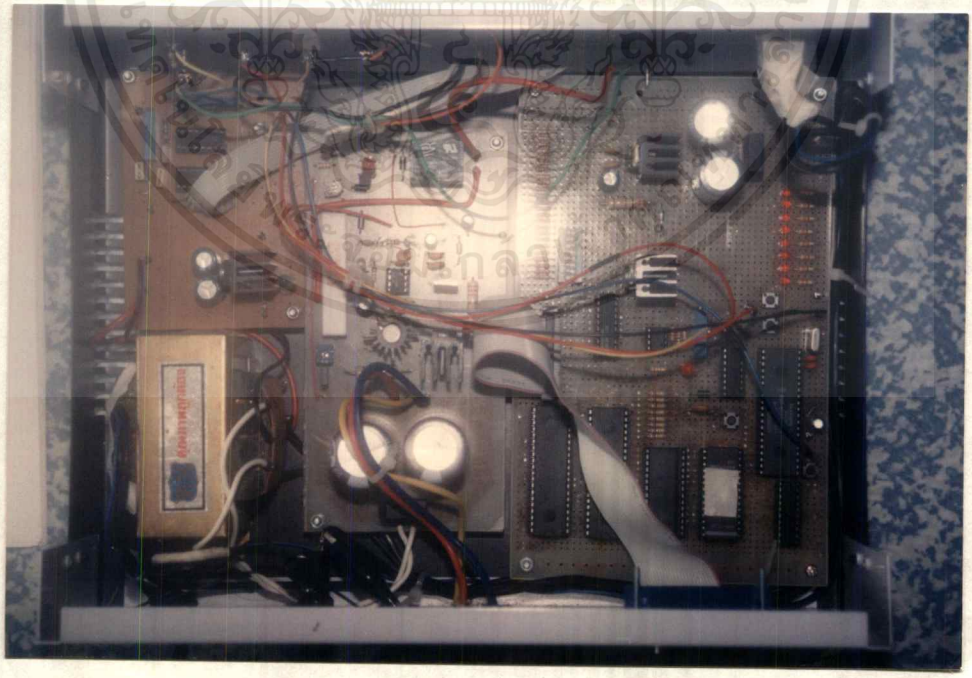
จากตารางผลการทดลอง จะเห็นว่าผลจากการส่งรหัสคำสั่ง เมื่อส่งผ่านระบบบัสแล้ว วงจร Microcontroller ก็ทำการแปลรหัสคำสั่งที่ส่งมาจากคอมพิวเตอร์ให้เป็นข้อมูลเลขฐานสองเพื่อที่จะส่งต่อไปยัง D/A จะถูกต้องตาม Step ที่ได้ทำการออกแบบไว้ แต่ผล Output Voltage จะมีการผิดพลาดอยู่ เนื่องมาจากการที่ตัว Power supply ไม่สามารถปรับค่าได้ถึงค่าศูนย์ คือจะมีแรงดันส่วนหนึ่งปรากฏที่เอาท์พุทอันเนื่องมาจาก การเปรียบเทียบแรงดันของ IC4 ซึ่งทำหน้าที่เป็นตัวเปรียบเทียบแรงดัน โดยที่ขาอินพุททั้งสองของ IC4 จะต้องมีแรงดันเป็นศูนย์ ซึ่งนั่นก็คือการนำเอาแรงดันเอาท์พุท มาเปรียบเทียบกับแรงดันที่ควบคุมโดยคอมพิวเตอร์ ดังนั้นค่าแรงดันที่ได้จากวงจร D/A มีค่าใกล้เคียงกับที่กำหนดไว้ แต่แรงดันที่ถูกแบ่งมาจากเอาท์พุทนั้นมีค่าไม่คงที่เท่าที่ควร จึงทำให้ค่าแรงดันเอาท์พุทที่ได้คลาดเคลื่อนอยู่



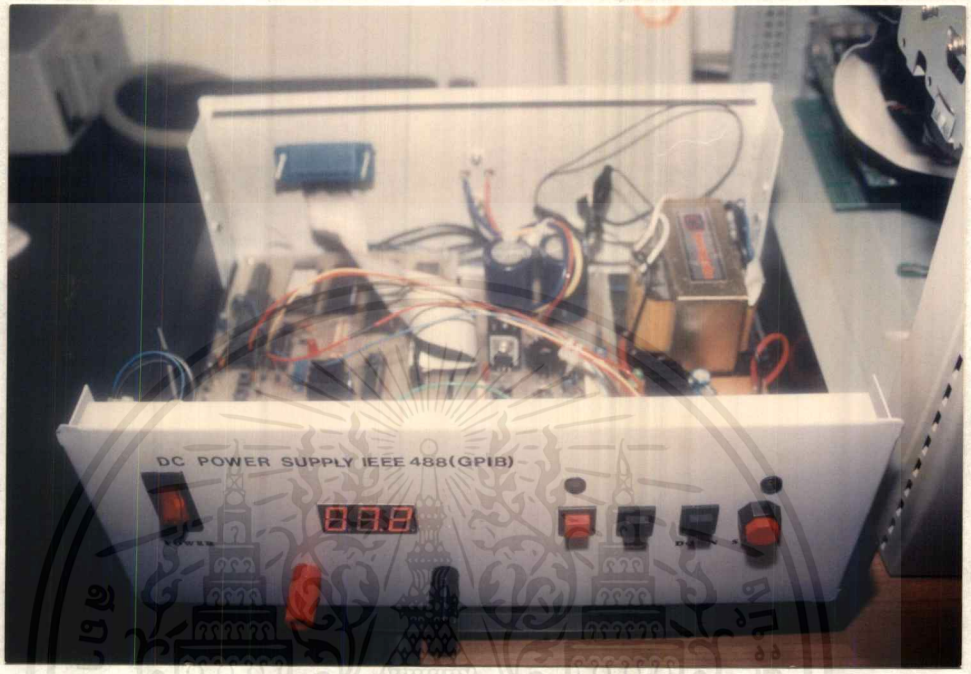
รูปที่ 5.1 แสดงกราฟความสัมพันธ์ระหว่างรหัสคำสั่งและแรงดันเอาท์พุท



รูปที่ 5.2 แสดงส่วนของการ์ดอินเตอร์เฟส



รูปที่ 5.3 แสดงการจัดวางแผ่นวงจรทั้งหมด



รูปที่ 5.4 แสดงภาพด้านหน้าของ Power Supply และ Computer

บทที่ 6
สรุปและวิจารณ์

ปัจจุบันการนำคอมพิวเตอร์มาประยุกต์ใช้งานทำได้อย่างกว้างขวาง เช่นเดียวกับการนำเอาคอมพิวเตอร์มาใช้ในการติดต่อสื่อสารกับเครื่องมือวัดต่างๆ ในทางอุตสาหกรรมส่วนใหญ่จะนิยมใช้ระบบบัสมาตรฐาน IEEE-488 (GPIB) เนื่องจากคุณสมบัติที่สำคัญของระบบบัสที่มีความเร็วสูง และสามารถเชื่อมต่อกับอุปกรณ์ได้หลายตัวในการติดต่อสื่อสาร ทำให้สะดวกในการใช้งาน โดยเครื่องมือวัดจะติดต่อคอมพิวเตอร์โดยผ่านการ์ดอินเตอร์เฟสเพื่อที่จะติดต่อสื่อสารกันได้

ปริญญานิพนธ์ฉบับนี้จะเป็นการศึกษาและออกแบบสร้าง Power Supply และการ์ดอินเตอร์เฟสให้สามารถนำมาใช้งานโดยการควบคุมผ่านระบบบัสมาตรฐาน IEEE-488 (GPIB) โดยที่ Power Supply ที่สร้างขึ้นมาจะประกอบด้วยส่วนของการควบคุมแปลงรหัสคำสั่งและส่วนของชุดจ่ายแรงดัน ในส่วนของชุดควบคุมจะมีไมโครคอนโทรลเลอร์ซึ่งจะทำงานตามโปรแกรมที่เขียนขึ้นมาและในส่วนของการอินเตอร์เฟสจะมีการ์ดอินเตอร์เฟสที่สร้างขึ้นมาและโปรแกรมการควบคุมที่ใช้ส่งรหัสคำสั่ง โดยในส่วนของการ์ดอินเตอร์เฟสจะประกอบด้วยวงจร 2 ส่วนคือ ส่วนของวงจรถอดรหัสแอดเดรสจากคอมพิวเตอร์และวงจรส่วนควบคุมการอินเตอร์เฟสกับอุปกรณ์

สำหรับในส่วนของชุดควบคุมด้วยไมโครคอนโทรลเลอร์ที่สร้างขึ้นนี้จะทำการสร้างรหัสคำสั่งที่ทำให้อุปกรณ์ที่เป็นตัวควบคุมเข้าใจถึงคำสั่งนั้นๆเพื่อให้สามารถติดต่อสื่อสารกันได้ โดยการสร้างคำสั่งในการ Hand sheck และคำสั่งที่ใช้สำหรับการอินเตอร์เฟสเพื่อให้การส่งข้อมูลหรือแอดเดรสเป็นไปตามขั้นตอนของการติดต่อของสัญญาณตามกระบวนการ Hand Sheck มาตรฐาน IEEE-488 นอกจากนี้ชุดควบคุมจะต้องทำการแปลงรหัสที่จะใช้สำหรับการควบคุมใน Local Mode และ Remote Mode ได้

จะเห็นว่าโครงงานนี้ได้ใช้บัฟเฟอร์มาตรฐาน IEEE-488 เบอร์ SN75160 และ SN75161 ในเฉพาะส่วนที่เป็นการ์ดอินเตอร์เฟสเพียงชุดเดียว เนื่องจากการควบคุมอุปกรณ์ จะควบคุม Power Supply เพียงตัวเดียว แต่ถ้ามีการควบคุมอุปกรณ์ที่มากขึ้นจะต้องใช้บัฟเฟอร์ของแต่ละอุปกรณ์นั้นด้วย

จากการทดลองเมื่อนำเอา Power Supply ที่สร้างขึ้นมาซึ่งทำหน้าที่เป็น Listener ทำการต่อเข้ากับระบบบัสมาตรฐาน IEEE-488 (GPIB) แล้วส่งรหัสคำสั่งจากคอมพิวเตอร์สามารถติดต่อสื่อสารกันได้ โดยสามารถปรับเปลี่ยนค่าแรงดันของ Power Supply ได้ทั้งการควบคุมด้วยมือ (Local Mode) และควบคุมได้ด้วยคอมพิวเตอร์ (Remote Mode) แต่ก็ยังมีปัญหาอยู่บ้าง กล่าวคือ ในส่วนของการป้อนค่าแรงดันเพื่อเป็นแรงดันอ้างอิงให้กับ Power Supply เมื่อทำการเปรียบเทียบ

ค่าแรงดันแล้วยังไม่สามารถที่จะทำให้แรงดันเอาท์พุทมีค่าต่ำสุดเป็นศูนย์ได้ คือ ยังมีแรงดันส่วนหนึ่งปรากฏที่เอาท์พุท แต่เมื่อทำการเพิ่มค่าแรงดันตาม step ที่ตั้งเอาไว้ก็สามารถที่จะได้ค่า แรงดันตามต้องการ โดยสามารถที่จะปรับค่าแรงดันขึ้นลงทีละ 1 โวลท์หรือ 0.1 โวลท์ก็ได้ ซึ่งจะต้องมีการพัฒนากันต่อไป โดยเฉพาะในส่วนของการสร้าง Power Supply ให้มีเสถียรภาพมากขึ้น เพื่อให้การทำงานเป็นไปได้ด้วยดี สำหรับโครงงานนี้ยังไม่ได้ทดลองใช้ร่วมกับระบบบัสมาตรฐานที่มีอยู่ทั่วไป และตัว Power Supply ยังไม่ได้ทดลองใช้ร่วมกับการ์ดมาตรฐานที่มีขายตามท้องตลาดทั่วๆไป แต่สามารถใช้ได้กับอุปกรณ์ควบคุมที่แตกต่างกันได้ เนื่องจากในการเขียนโปรแกรมได้ใช้ขบวนการต่างๆของ IEEE-488 เดียวกัน

บรรณานุกรม

1. Eugene Fisher - C.W.Jensen. PET/CBM and the IEEE 488 Bus (GPIB). 2nd Edition. Osborne/McGraw - Hill Berkeley, California 94710 U.S.A.

2. สมคิด วิริยะประสิทธิ์ชัย, สมบูรณ์ มาลานนท์ ทฤษฎีและการออกแบบแหล่งจ่ายไฟกระแสตรงแบบเชิงเส้น กรุงเทพมหานคร : สำนักพิมพ์ฟิสิกส์เซ็นเตอร์ , 237 หน้า

3. ธานินทร์ ถาวรศาสนวงศ์, ทินกร ดุ๊ก . การอินเตอร์เฟส IBM PC . กรุงเทพฯ. ฟิสิกส์เซ็นเตอร์ การพิมพ์. ม.ป.ป.

4. สุนทร วิทูสุรพจน์. การโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล 8051. กรุงเทพฯ. บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน).

5. ผศ.สมยศ จุณณะปิยะ. การใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS -51 . พิมพ์ครั้งที่ 1. สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ.

ภาคผนวก

```
;SOFTWARE MCS-51 FOR POWER SUPPLY
;IN STANDARD IEEE 488 (GPIB)

        ORG     0000H
DAV     EQU     0E4H
NRFD    EQU     0E1H
NDAC    EQU     0E0H
RENE    EQU     0E4H
ATN     EQU     0E6H
EOI     EQU     0E5H
DATA    EQU     080H
GTL     EQU     01H
UP      EQU     P1.0
DOWN    EQU     P1.1
SWLOC   EQU     P1.2
STEP    EQU     P1.3
MLA     EQU     030H
LLO     EQU     011H
COUNT   EQU     031H
;INITIAL 8255
        MOV     P2,#60H
        MOV     R0,#03H
        MOV     A,#098H
        MOVX    @R0,A
        MOV     P2,#40H
        MOVX    @R0,A
        MOV     R4,#00H
        MOV     R0,#00H

LOCAL:  ACALL   START
        MOV     P2,#60H
```

```
        MOV     R1,#02H
        MOVX    A,@R1
        JNB     RENE,LOCAL
        ACALL   START
        MOV     P2,#60H
        MOV     R1,#00H
        MOVX    A,@R1
        MOV     MLA,A           ;SET ADDRESS
        MOV     R1,#02H
        ACALL   DAVA
        JC      LOCAL           ;ATN=HI
        MOV     A,B
        CJNE    A,MLA,L1
        SJMP    REMS            ;DATA=MLA GO REMS
L1:     CJNE    A,#LLO,LOCAL
LWLS:   ACAL L  START           ;DATA=LLO GO LWLS
        ACAL L  DAVA
        JC      LWLS            ;ATN=HI
        MOV     A,B
        CJNE    A,MLA,LWLS
        SJMP    RWLS            ;DATA=MLA GO RWLS
REMS:   JB      SWLOC,RE        ;PUSH SWLOC GO LOCAL
        MOV     B,#GTL
        SJMP    CMP
RE:     MOV     COUNT,#00H
        ACALL   DAVA
        JNC     CMD1            ;ANT=HI GO INFO1
        ACALL   INFO1
        MOV     R4,A
CMD1:   MOV     A,B
        CJNE    A,#GTL,REMS1
```

```
LOCA:   SJMP    LOCAL       ;DATA=01 GO START
REMS1:  CJNE    A,#LLO,REMS
RWLS:   MOV     COUNT,#00H
        ACALL   DAVA        ;DATA=11 GO RWLS
        JNC     CMD2        ;ANT=HI GO INFO1
        ACALL   INFO1
        MOVX    @R1,A
        MOV     R4,A
        AJMP    RWLS
CMD2:   MOV     A,B
        CJNE    A,#GTL,RWLS1
        SJMP    LWLS        ;DATA=01 GO LWLS
RWLS1:  CJNE    A,#LLO,RWLS
        SJMP    REMS
INFO1:  MOV     R5,B        ;KEEP CHAR1
        ACALL   CHEOI
        JNB     EOI,ERROR   ;DATA = 1 CHAR ERROR
INFO2:  ACALL   DAVA
        MOV     R6,B        ;KEEP CHAR2
        ACALL   CHEOI
        JNB     EOI,ERROR   ;DATA = 1 CHAR ERROR
INFO3:  ACALL   DAVA
        MOV     R2,B        ;KEEP CHAR3
        ACALL   CHEOI
        JNB     EOI,COMP    ;DATA = 3 CHA GO COMP
INFO4:  ACALL   DAVA
        MOV     R7,B        ;KEEP CHAR4
COMP:   ACALL   ASCII
        MOV     P2,#40H
        MOV     R1,#01H
        MOVX    @R1,A       ;SHOW DATA
```

```
                JNB     SWLOC,LOCA
ERROR:          RETI
CHEOI:          MOV     P2,#40H
                MOV     R1,#02H
                MOVX    A,@R1
                RETI

START:          MOV     P2,#40H
                MOV     R1,#01H
                CLR     C
                MOV     A,#06H
                ADD     A,R4
                JC      SAK             ;R1>250 GO SAK
                JB      UP,SAK
                MOV     A,R4
                JNB     STEP,NOSTEPH    ;CHECK STEP
                CJNE    R4,#0F0H,OVSTEP
                SJMP    SAK
OVSTEP:         JNC     SAK
                ADD     A,#09H
NOSTEPH:        INC     A
                MOVX    @R1,A
                MOV     R4,A
SAK:            CLR     C
                MOV     A,R4
                ADD     A,#0FFH
                JNC     END             ;R1<0 GO END
                MOV     R2,#00H
                MOV     R3,#00H
                ACALL   DELAY
                JB      DOWN,END
```

```
                MOV     A,R4
                JNB     STEP,NOSTEPL    ;CHECK STEP
                CJNE    R4,#0AH,OVSTEP1
                SJMP    L
OVSTEP1:        JC      END
L:              SUBB    A,#09H
NOSTEPL:        DEC     A
                MOVX    @R1,A
                MOV     R4,A
END:            RETI
DAVA:           SETB    C               ;ATN HI
                MOV     P2,#60H
                MOV     R1,#02H
                MOVX    A,@R1
                JNB     RENE,READY      ;REN HI
                MOV     B,#00H
                RETI
READY:          MOV     P2,#40H
                MOV     R1,#02H
                SETB    NRFD            ;NRFD HI
                CLR     NDAC            ;NDAC LOW
                MOVX    @R1,A
DLOW:           MOVX    A,@R1
                JB      DAV,DLOW        ;DAV LOW
                MOV     A,#00H          ;NRFD LOW
                MOVX    @R1,A
                MOV     P2,#60H
                MOVX    A,@R1           ;CHECK ANT
                JNB     ATN;CNT
                INC     COUNT
CNT:            MOV     C,ATN
```

```
                MOV     P2,#40H
                MOVX    A,@R0
                MOV     B,A             ;DATA IN
                SETB    NDAC            ;NDAC HI
                MOVX    @R1,A
DHIGH:          MOVX    A,@R1
                JNB     DAV,DHIGH       ;DAV HI
                CLR     NDAC            ;NDAC LOW
                MOVX    @R1,A
                RETI
ASCII:
                MOV     A,COUNT
                CJNE    A,#04H,CHA3
                CJNE    R6,#2EH,CHA4
                SETB    C
CHA3:           JNC     BIT3
                MOV     A,R5
                MOV     R6,A
                MOV     A,R2
                MOV     R7,A
                MOV     A,#00H
                SJMP    COLUMN2
CHA4:           MOV     A,#00H
                CJNE    R5,#30H,BIT1
                ADD     A,#00H          ;R5=0 GET A ADD 0
                SJMP    COLUMN2
BIT1:           CJNE    R5,#31H,BIT2
                ADD     A,#64H          ;R5=1 GET A ADD 100
                SJMP    COLUMN2
BIT2:           CJNE    R5,#32H,BIT3
```

```
                ADD     A,#0C8H         ;R5=0 GET A ADD 200
                SJMP    COLUMN2
BIT3:           RETI
COLUMN2:        CJNE    R6,#30H,BIT11
                ADD     A,#00H          ;R6=0 GET A ADD 0
                SJMP    COLUMN3
BIT11:          CJNE    R6,#31H,BIT22
                ADD     A,#0AH          ;R6=1 GET A ADD 10
                SJMP    COLUMN3
BIT22:          CJNE    R6,#32H,BIT33
                ADD     A,#14H          ;R6=2 GET A ADD 20
                SJMP    COLUMN3
BIT33:          CJNE    R6,#33H,BIT44
                ADD     A,#1EH          ;R6=3 GET A ADD 30
                SJMP    COLUMN3
BIT44:          CJNE    R6,#34H,BIT55
                ADD     A,#28H          ;R6=4 GET A ADD 40
                SJMP    COLUMN3
BIT55:          CJNE    R6,#35H,BIT66
                ADD     A,#32H          ;R6=5 GET A ADD 50
                SJMP    COLUMN3
BIT66:          CJNE    R6,#36H,BIT77
                ADD     A,#3CH          ;R6=6 GET A ADD 60
                SJMP    COLUMN3
BIT77:          CJNE    R6,#37H,BIT88
                ADD     A,#46H          ;R6=7 GET A ADD 70
                SJMP    COLUMN3
BIT88:          CJNE    R6,#38H,BIT99
                ADD     A,#50H          ;R6=8 GET A ADD 80
                SJMP    COLUMN3
BIT99:          CJNE    R6,#39H,BITAA
```

```
                ADD     A,#5AH          ;R6=9 GET A ADD 90
                SJMP    COLUMN3
BITAA:          RETI
COLUMN3:        CJNE    R7,#30H,BIT01
                ADD     A,#00H          ;R7=0 GET A ADD 0
                RETI
BIT01:          CJNE    R7,#31H,BIT02
                ADD     A,#01H          ;R7=1 GET A ADD 1
                RETI
BIT02:          CJNE    R7,#32H,BIT03
                ADD     A,#02H          ;R7=2 GET A ADD 2
                RETI
BIT03:          CJNE    R7,#33H,BIT04
                ADD     A,#03H          ;R7=3 GET A ADD 3
                RETI
BIT04:          CJNE    R7,#34H,BIT05
                ADD     A,#04H          ;R7=4 GET A ADD 4
                RETI
BIT05:          CJNE    R7,#35H,BIT06
                ADD     A,#05H          ;R7=5 GET A ADD 5
                RETI
BIT06:          CJNE    R7,#36H,BIT07
                ADD     A,#06H          ;R7=6 GET A ADD 6
                RETI
BIT07:          CJNE    R7,#37H,BIT08
                ADD     A,#07H          ;R7=7 GET A ADD 7
                RETI
BIT08:          CJNE    R7,#38H,BIT09
                ADD     A,#08H          ;R7=8 GET A ADD 8
                RETI
BIT09:          CJNE    R7,#39H,BIT0A
```

```
                ADD     A,#09H          ;R7=9 GET A ADD 9
IT0A:           RETI
DELAY:          DJNZ    R2,DELAY
                DJNZ    R3,DELAY
                RETI
                END
```

โปรแกรมควบคุมการส่งข้อมูล

```
program IEEE_488;
uses dos,crt;
const
   device_addr_MLA: array[0..30]of byte=  ,
   ($20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$2A,$2B,$2C,$2D,$2E,$2F,
   $30,$31,$32,$33,$34,$35,$36,$37,$38,$39,$3A,$3B,$3C,$3D,$3E);

   GTL = $01;  { Go To Local }
   SDC = $04;  { Select Device Clear }
   PPC = $05;  { Parallel Poll Configure }
   GET = $08;  { Group Execute Trigger }
   TCT = $09;  { Take Control }
   LLO = $11;  { Local Lock Out }
   PPU = $15;  { Parallel Poll Unconfigure }
   SPE = $18;  { Serial Poll Enable }
   SPD = $19;  { Serial Poll Disable }

var     bus_error:boolean;  data:string[4];
        finish,addr,code,i: integer;
        ch:char;  a:real;   MLA:byte;

procedure Init_card;
begin
 port[$313]:=$88;  { A1 = O/P , B1 = O/P , Cup = I/P , Clow = O/P }
 port[$317]:=$88;  { CONTROL UP 2 = I/P , CONTROL LOW 2 = O/P }
 port[$311]:=$0B;
 port[$310]:=$FF;  { CLEAR DATA ON DIO LINE }
 port[$316]:=$F4;  { REN HI}
```

```
end;
procedure addr_to_Hex(addr:integer;var MLA :byte);
begin
        MLA := device_addr_MLA[addr];
end;

procedure source_operation(data:byte);
begin
        port[$312] := (port[$312] or $0C); { make DAV = H ,EOI = H }
        if ((port[$312] or $CF ) = $FF ) then { check NRFD = NDAC = H }
    begin
        bus_error := true;
        writeln;textcolor(12);
        writeln(' Bus error !!!!!');
    end
 else
    begin
        bus_error := false;
        port[$310] := data; { put data on DIO line }
   repeat
   until ((port[$312] or $EF) = $FF) or keypressed; { check NRFD = H }
        port[$312] := (port[$312] and $F7); {make DAV = L }
   repeat
   until ((port[$312] or $DF) = $FF) or keypressed; { check NDAV = H }
        port[$312] := (port[$312] or $0C) ; { make DAV = H ,EOI = H}
   end;
end;

procedure send_command(addr:integer; command:byte);
begin
        port[$316] := $02; { make REN = L , IFC = H , ATN = L }
```

```
        addr_to_Hex(addr,MLA);
        source_operation(MLA);
        if bus_error then
          begin
              writeln;textcolor(1);
              writeln(' Bus error !!!!!');
          end
      else source_operation(command);
       port[$312]:=port[$312]or$03;
end;

procedure Go_To_Local(addr:integer);
begin
          send_command(addr,GTL);
end;

procedure Local_Lock_Out;
begin
        source_operation(LLO);
end;

procedure send_message(addr:integer;m:string);
var msage : char;
  i:integer;
begin
        port[$316] := $02; { make REN = L , IFC = H , ATN = L }
         addr_to_Hex(addr,MLA);
         source_operation(MLA);
        if bus_error then
        { bus_error := true;}
  begin
```

```
            writeln;textcolor(1);
                    writeln(' Bus error !!!!!');
                end
            else
        begin
            bus_error:=false;
            port[$316] := $03; { make REN = L , IFC = H , ATN = H }
            for i := 1 to length(m) do
         begin
             msage := m[i];
             source_operation(integer(msage));
        end;
       port[$312]:=$0B; {  set EOI low }
    end;
    end;

    begin
             clrscr;
            init_card;
             textcolor(11);
             write(' Address of Device :=>> ');
             readln(addr);
                repeat
                    textcolor(11);
                    write(' Enter Your Number Output Voltate :=>> ');
                    readln(data);
                    val(data,a,code);
                     if (code=00)then
            begin
                    send_message(addr,data);
             end
```

```
        else
         begin
             writeln;
             textcolor(12);
             writeln(' Now Data is Error !.Try again with data');
             writeln;
         end;
             writeln;textcolor(10);
             write(' Do you want quit program [Y/N] .... ');
             readln(ch);writeln;
             if (ch = 'Y')or(ch='y') then finish:=1;
    until  (finish=1 );
    Go_To_Local(addr);
    writeln(' Quit Program ');
end.
```