ปีการศึกษา 2533

ยูนิเวอร์แซล I/O พอร์ท

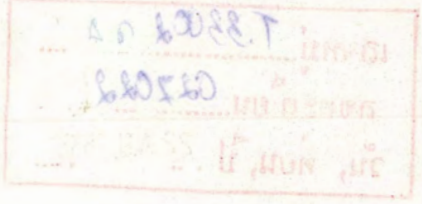
โดย

นายวรเดช คุณานพรัตน์ เลขประจำตัว 313312

นายวุฒพงศ์ จิรพรเจริญ เลขประจำตัว 313314

อาจารย์ที่ปรึกษา

ดร.ไพศาล นาคพิพัฒน์



ปริญญานิพนธ์ปีการศึกษา 2533
ภาควิชา เทคนิคอุตสาหกรรม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ยูนิเวอร์แซล I/O พอร์ท

ผู้จัดทำ

1 นายวรเดช คุณานพรัตน์ เลขประจำตัว 313312
2 นายวุฒพงศ์ จิรพรเจริญ เลขประจำตัว 313314

_____อาจารย์ที่ปรึกษา
(ดร.ไพศาล นาคพิพัฒน์)

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการเสนอวิธี การออกแบบและสร้างวงจรที่ใช้ในการควบคุมอุปกรณ์ภายนอก โดยใช้คอมพิวเตอร์พีซี ควบคุมการทำงานของระบบ โดยสามารถที่จะทำการควบคุมอุปกรณ์ภายนอกได้ 32 CH สามารถที่จะกำเนิดแรงดันคงที่ได้ 0 V ถึง 10 V และทำหน้าที่กำเนิดความถี่ซายน์เวฟได้ อีกทั้งยังมีตัว Timer/Counter เพื่อที่จะศึกษาการทำงาน

ABTRACT

THE OBJECTION OF THIS THESIS IS TO INTRODUCE COPUTER IS PROCESS , 'DESIGNING AND MAKING UP ELECTRONICS'S CIRCUITRY FOR EXTERNAL CONTROL BY USE PERSONEL COMPUTER . THIS SYSTEM IS ABLE TO CONTROL 32 CH EXTERNAL CHANNEL , GENERATING REGULATED VOLTAGE OV - 10V , AND SOURCE FOR SINUSOILDAL WAVE PLUS PROVIDING TIMER/COUNTER FOR OBSERVINSG SYSTEM 'S OPERATION

สารบัญ

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

จะพบว่าในทุกวันนี้ Computer นับได้ว่ามีบทบาทมากในชีวิตประจำวันของคนเราเกือบทุกคน ไม่ว่าจะเป็นทั้งโรงงานอุตสาหกรรมหรือบริษัทห้างร้านต่างๆ ต่างก็ใช้คอมพิวเตอร์ทั้งนั้น ซึ่งจะพบว่าคอมพิวเตอร์นั้นไม่ได้จำกัดอยู่แต่เพียง Software เท่านั้น โดยเราสามารถที่จะนำคอมพิวเตอร์นี้ไปประยุกต์ใช้งานทางด้าน Hardware ได้อีกมากมายจะทำให้คอมพิวเตอร์นี้มีประโยชน์มากขึ้นและ จะไม่ถูกจำกัดขอบเขตของตัวมันเองอีกต่อไป ในบริญญานิพนธ์นี้ก็ได้นำคอมพิวเตอร์ไปประยุกต์ใช้งานอีกด้านหนึ่ง ซึ่งได้เล็งเห็นว่าผู้ใช้งานนั้นสามารถที่จะทำความเข้าใจถึง การใช้คอมพิวเตอร์ร่วมกับอุปกรณ์ภายนอกได้ว่าทำกันอย่างไรซึ่งอาจจะเห็นได้ว่ามันง่าย แต่จะมีสักกี่คนที่จะเข้าใจเกี่ยวกับเรื่องนี้ ปริญญานิพนธ์นี้จึงได้ทำการออกแบบวงจร และใช้ Program ควบคุมการทำงานของระบบ (ใช้ภาษา Pascal) ซึ่งสามารถที่จะให้ผู้ใช้งานสามารถเรียนรู้การติดต่อกับอุปกรณ์ภายนอก และผู้ที่ต้องการนำเอาไปใช้งานในการควบคุมอุปกรณ์ภายนอกได้เช่นการ ปิด-เปิด อุปกรณ์ที่ใช้ไฟ AC ต่างๆ หรืออาจจะทำการตั้งเวลาการปิดเปิดได้แล้วแต่ความต้องการ และสามารถที่จะทำการเรียนรู้เกี่ยวกับการส่งข้อมูลและรับข้อมูลออกทาง Port8255 รวมไปถึงการศึกษา IC8253 ซึ่งทำหน้าที่เป็น Timer/Counter โดยอาจจะพบได้ว่ามีใช้ในพวก Single Board แล้วซึ่งมีขนาด 8 Bit แต่ในที่นี้เราจะเล่นกับเครื่อง 16 Bit บ้างว่าทำกันอย่างไร

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อที่จะสามารถควบคุมอุปกรณ์ภายนอกได้ 32 CH ขึ้นอยู่กับความต้องการ

1.2.2 เพื่อที่จะสามารถเรียนรู้เกี่ยวกับ IC 8255 เกี่ยวกับการติดต่อกับอุปกรณ์ภายนอก

1.2.3 เพื่อที่จะสามารถเรียนรู้การทำงานของ IC 8253 ใน Mode การทำงานต่างๆ เพื่อนำไปประยุกต์ใช้งาน

1.2.4 เพื่อที่จะเรียนรู้วิธีการออกแบบและสร้างวงจรในการติดต่อกับคอมพิวเตอร์พีซี และวงจรต่างๆ ภายในระบบ

1.2.5 เพื่อให้ผู้ใช้งานสามารถได้เข้าใจ และ เรียนรู้ถึงวิธีการติดต่อกับอุปกรณ์ภายนอกทำกันอย่างไร

1.2.6 สามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง

1.3 ขอบเขตของโครงงาน

1.3.1 ทำการใช้งาน ณ.อุณหภูมิห้อง

1.3.2 สามารถติดต่อกับอุปกรณ์ภายนอกได้ 32 CH

1.3.3 กำเนิดแรงดันอยู่ในช่วง 0-10 V

1.3.4 กำเนิดความถี่ Sine Wave ได้ 1Hz - 2MHz

1.3.5 สามารทำหน้าที่เป็น Timer/Counter

1.4 วิธีการดำเนินงาน

ในการดำเนินงานของปริญญานิพนธ์นี้ได้แบ่งออกเป็น 7 ส่วนด้วยกัน ซึ่งแต่ละส่วนจะมีรายละเอียดดังต่อไปนี้

1.4.1 ทำการศึกษาและออกแบบวงจร ในส่วนนี้เราจะต้องศึกษาถึงคุณสมบัติของวงจรและอุปกรณ์ที่จะนำมาใช้ในการทดลองและจะต้องจัดหา อุปกรณ์ที่มีแล้วนำมาใช้งาน อุปกรณ์ที่ได้เหล่านั้นจะนำมาใส่แทน Block ต่างๆ ที่ได้ทำการออกแบบไว้ให้เหมาะสมกับความต้องการ

1.4.2 สร้างวงจร เมื่อได้แบบมาแล้วก็จะต้องนำวงจรที่ได้นั้นมาทำการทดลองบนแผ่น Proto Board ว่าเป็นไปตามที่คาดหวังหรือไม่

1.4.3 เขียนโปรแกรมควบคุม โดยจะเอาทฤษฎีที่ได้ทำการเขียนโปรแกรมโดยจะนำแนวทฤษฎีมาเขียน โฟล์ชาร์ตก่อน แล้วจึงถอดมาเป็นโปรแกรม ซึ่งในโปรแกรมที่ใช้ในการควบคุมนั้นจะใช้ Pascal V5.5 ซึ่งเป็นเป็นของบริษัท Borland

1.4.4 ทดลองวงจรเมื่อผ่านขั้นตอนที่ 2,3 แล้วถึงขั้นตอนว่าด้วยการทดลองเป็นไปดังเป้าหมายหรือไม่ โดยจะตรวจสอบผลจากการทดลองว่ามีข้อผิดพลาดอะไรหรือไม่ และวงจรจะทำงานหรือไม่

1.4.5 แก้ไขปรับปรุงโปรแกรมและวงจรให้ดีขึ้น ในส่วนของโครงงานจะประกอบไปด้วยโปรแกรม และวงจรเราจะต้องแก้ไขข้อผิดพลาดที่เกิดขึ้นจากขั้นที่ 4 ให้ดีขึ้น เพื่อที่จะให้โครงงานนี้มีประสิทธิภาพที่ดีที่สุดเท่าที่จะทำได้ โดยการนำข้อผิดพลาดที่ได้จากขั้น 4 มาหาจุดบกพร่องและแก้ไข ก่อนที่จะสรุปผลต่างๆ ออกมา

1.4.6 สรุปผล ร่างปริญญานิพนธ์ และตรวจทาน ซึ่งในขั้นตอนนี้จะพบว่าเมื่อชิ้นงานที่ทำสำเร็จลุล่วงลงแล้วเราต้องทำการจัดทำ ปริญญานิพนธ์ขึ้นมาเพื่อที่จะทำให้รู้ลักษณะของขอบเขตของโครงงานที่ทำ อีกทั้งยังเป็นแนวทางให้รุ่นต่อๆ ไป

1.4.7 พิมพ์ ตรวจทานและแก้ไข พร้อมเสนอรายงานเราจะทำการจัดพิมพ์ปริญญานิพนธ์ที่ได้จัดทำขึ้นแล้วตรวจสอบความเรียบร้อยเป็นอันเสร็จภาระกิจที่ทำพร้อมที่จะเสนอรายงานนี้

บทที่ 2

ทฤษฎีและหลักการ

2.1 ทฤษฎีของระบบ

ในส่วนนี้จะเป็นการอธิบายทฤษฎีที่เกี่ยวข้องกับ PROJECT ชิ้นนี้ ซึ่งได้แบ่งออกเป็นหัวข้อต่างๆดังต่อไปนี้

2.1.1 SLOT บน IBM/PC
2.1.2 I/O MAP ของ IBM
2.1.3 การใช้งาน IC.8255
2.1.4 IC. PAL
2.1.5 DAC,ADC
2.1.6 การใช้งาน IC.8253

2.1.1 สัญญาณต่างๆ บน Slot IBM/PC

จะพบว่าภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเทอร์เฟสเข้าไปในระบบได้ในภายหลัง โดยผ่านทางสล็อตที่อยู่บน Main Board สำหรับสล็อตบนเมนบอร์ดนี้จะมีจำนวน 5 สล็อต (สำหรับใน IBM PC/XT) ซึ่งแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขาแบ่งออกเป็น 2 ข้างๆ ละ 31 ขาส่วนการเรียกตำแหน่งขาของสล็อตเหล่านี้ จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล็อตโดยขาที่อยู่ทางด้านซ้ายของสล็อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขาเช่นขา B16 ก็คือขาที่อยู่ทางด้านซ้ายของสล็อตขาที่ 16 (นับจากทางด้านท้ายของเครื่อง) ส่วนขาที่อยู่ทางด้านขวาของสล็อตจะเรียกโดยใช้อักษร "A" นำหน้าเลขตำแหน่งของขา เช่น ขา A24 ก็คือขาทางด้านขวาของสล็อตขาที่ 24 (นับจากทางด้านท้ายของเครื่อง) แต่ละขาของสล็อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่างๆบนเมนบอร์ด ทำให้การวางจรอินเทอร์เฟสกับ IBM/PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล็อตเหล่านี้จะประกอบไปด้วย เส้นสัญญาณของบัสแอดเดรส (Address Bus), บัสข้อมูล (Data Bus), บัสควบคุมสำหรับการเขียน/อ่านข้อมูลจากหน่วยความจำหรือพอร์ท I/O, เส้นสัญญาณสำหรับการขออินเทอร์รัพท์ของวงจรอินเทอร์เฟส, เส้นสัญญาณสำหรับการขอ DMA, สัญญาณฐานเวลา (Timing Signal) ต่างๆที่ใช้ในระบบ, เส้นสัญญาณแสดงการรีเฟรชหน่วยความจำและสัญญาณสำหรับการตรวจสอบความผิดพลาด (I/O CHCK)

การที่เราจะติดต่อกับวงจร InterFace ที่ต่อเพิ่มเข้าไปในระบบจะต้องรู้เกี่ยวกับตำแหน่งหน้าที่ของแต่ละขาบน Slot ที่จำเป็นในการใช้งานก่อน เพื่อที่จะให้เรียนรู้และเข้าใจได้ง่าย เพื่อที่จะสามารถต่อสัญญาณต่างๆ ของระบบกับ ชิ้นงานที่สร้างขึ้นได้ และสามารที่จะทำการดึงสัญญาณ Address มาใช้ในการ Decode เพื่อใช้ในการติดต่อกับ ชิ้นงานได้

1) รายละเอียดเกี่ยวกับขาสัญญาณต่างๆที่จำเป็น

Reset DRV (ขา B2) : ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งจะแอคทีฟ (ลอจิก "1") ในช่วงที่เราเริ่มจ่ายไฟให้กับระบบ และจะยังคงแอคทิฟไปจนกว่าระบบต่างๆ ภายใน IBM/PC จะพร้อมที่จะทำงานได้จากนั้นสัญญาณนี้ก็จะเปลี่ยนกลับเป็นลอจิก "0" นอกจากนี้ในระหว่างการทำงานของ IBM/PC ถ้าระดับแรงดันของแหล่งจ่ายไฟตกลง สัญญาณนี้ก็จะถูกทำให้แอคทีฟเช่นกัน โดยทั่วไปแล้วสัญญาณนี้จะถูกนำไปใช้ในการรีเซ็ทวงจรอินเทอร์เฟสหรืออุปกรณ์ I/O ต่างๆ ในช่วงที่เริ่มจ่ายไฟให้ กับระบบซึ่งจะเป็นการทำให้วงจรหรืออุปกรณ์เหล่านั้นถูกบรับ

ให้อยู่ในสภาวะที่แน่นอนก่อนที่จะเริ่มต้นการทำงานในระบบ (สภาวะนี้เป็นสภาวะที่เราทราบ และต้องการให้วงจรทำงานในขณะที่ระบบถูกรีเซ็ท)

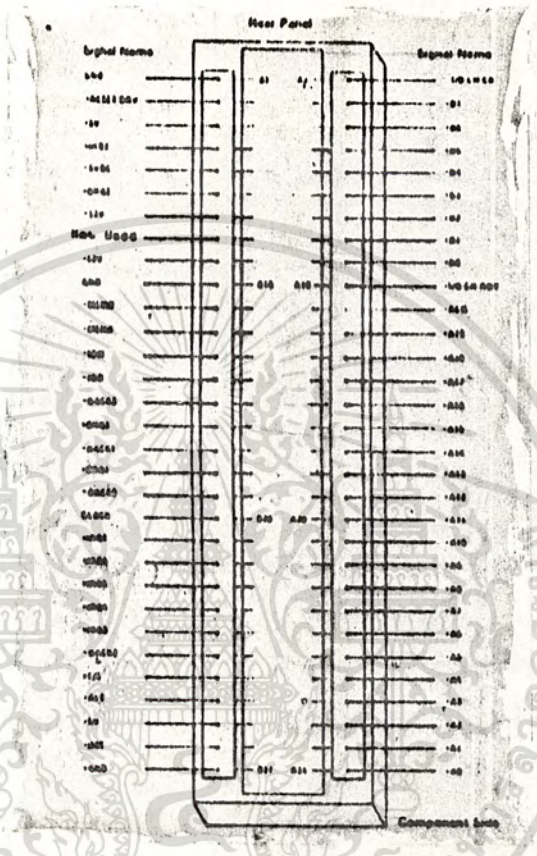
A0-A19 (Addresss Bus: ขา A31-A12) : ขาสัญญาณทั้ง 20 ขานี้เป็นเอาท์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำ หรืออุปกรณ์ I/O ที่ 8088 ต้องการติดต่อด้วย โดยที่สัญญาณ A0 จะมีนัยสำคัญต่ำสุด (Least Significant Bit) และ A19 จะมีนัยสำคัญสูงสุด (Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส A0-A19 นี้ จะถูกกำหนดโดย 8088 ในระหว่างขบวนการอ่าน/เขียนข้อมูลลงในหน่วยความจำหรืออุปกรณ์ I/O แต่ในช่วงของขบวนการ DMA นั้น DMA-Controller จะเป็นผู้กำหนดค่าแอดเดรสบนบัสแอดเดรสเอง (ในระหว่างนี้ 8088 จะถูกตัดออกจากระบบ) จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสของหน่วยความจำได้ถึง 1Mbyte แต่อย่างไรก็ตามจะมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว คือ แอดเดรสของหน่วยความจำ RAM บนเมนบอร์ดที่ถูกใช้โดยระบบจำนวน 64Kbyte(สำหรับ IBM PC/XT จะเป็นจำนวน 256 Kbyte) และแอดเดรสสำหรับหน่วยความจำ ROM อีก 48Kbyte ซึ่งถูกจัดในช่วงของแอดเดรสบนสุดใน 1Mbyte คือ OFC00H จนถึง OFFFFFH (สำหรับ IBM PC/XT จะเป็น 64Kbyte)

สำหรับการอ้างแอดเดรสของพอร์ท I/O นั้น จะใช้เส้นแอดเดรสเพียง 16 เส้น คือ A0-A15 ซึ่งจะทำให้อ้างแอดเดรสของพอร์ทได้ 64K พอร์ท โดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือคือ A16-A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสของพอร์ทเพียง 10 เส้น คือจาก A0-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง 0200H จนถึง 03FFH เท่านั้น

D0-D7 (Data Bus; ขา A9-A2) : ขาสัญญาณนี้จะเป็นแบบ Bi-Directional ซึ่งต่อกับบัสข้อมูลของระบบ เพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ท I/O กับ IBM/PC โดยบิท D0 จะมีนัยสำคัญต่ำสุดและบิท D7 จะมีนัยสำคัญสูงสุด สำหรับในบัสไซเคิลของการเขียนข้อมูลที่สร้างขึ้นโดย 8088 นั้น ข้อมูลจะถูกส่งออกมาบนบัสข้อมูล ก่อนที่สัญญาณ IOW (ในกรณีที่ต้องการส่งข้อมูลให้กับพอร์ท) หรือ MEMW (ในกรณีที่ต้องการส่งข้อมูลให้กับหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) ซึ่งโดยทั่วไปขอบขาขึ้นของสัญญาณ IOW หรือ MEMW นี้ จะถูกใช้เพื่อสั่งให้พอร์ท I/O หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

สำหรับในบัสไซเคิลของการอ่านข้อมูลที่สร้างขึ้นโดย 8088 นั้น พอร์ท I/O หรือหน่วยความจำที่ถูกอ้างถึงจะต้องส่งข้อมูลออกมาบนบัสข้อมูล ก่อนที่สัญญาณ IOR (ในกรณีที่ต้องการอ่านข้อมูลจากพอร์ท) หรือ MEMW(ในกรณีที่ต้อง

การอ่านข้อมูลจากหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น)



รูปที่ 2.1 แสดงตำแหน่งขาต่างๆ บน Slot IBM

IOR (I/O Read; ขา B14) : ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ที่สร้างขึ้นโดย 8288 Bus Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูล โดยข้อมูลจะต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ IOR เพื่อให้มั่นใจได้ว่า 8088 สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการ DMA 8237A-5 DMA Controller จะทำการสร้างสัญญาณ IOR เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำ (แทนที่จะเป็นแอดเดรสของพอร์ท I/O) ที่พอร์ท I/O ที่ขอ DMA ต้องการจะนำข้อมูลไปเก็บ การที่พอร์ทใดจะส่งข้อมูลออกมาบนบัสข้อมูลนั้น จะอาศัยสัญญาณ DACK จาก DMA Controller เป็นตัวกำหนด เช่นกรณีที่สัญญาณ DACK1 แอคทีฟก็จะแสดงว่าพอร์ท

I/O ที่จะต้องส่งข้อมูลออกมาบนบัสข้อมูลก็คือพอร์ท I/O ที่ขอ DMA ผ่านทางแชนแนลที่ 1 (DRQ1) เป็นต้น

IOW (I/O Write; ขา B13) : ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่งถูกสร้างขึ้นโดย 8288 Bus Controller เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลที่อยู่บนบัสข้อมูลข้อมูลอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ IOW แทนขอบขาลงในการทำให้พอร์ท I/O ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้ เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อน สำหรับในขบวนการ DMA นั้น DMA-Controller จะทำการสร้างสัญญาณ IOW เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ท I/O ที่ขอ DMA ต้องการจะอ่านข้อมูล

AEN สัญญาณนี้เป็นเอาท์พุทที่ใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ AEN แอคทีฟ (ลอจิก "1") นั้นเป็นบัสไซเคิลของขบวน การ DMA สำหรับบนเมนบอร์ดของ IBM/PC นั้นจะใช้สัญญาณนี้ในการดิสเอเบิล (Disable) 8288 Bus Controller และจะใช้ดิสเอเบิลพอร์ท I/O ต่างๆ ที่ไม่เกี่ยวข้องกับขบวนการ DMA ที่เกิดขึ้นนี้ ที่จำเป็นต้องทำเช่นนี้ก็เพราะในระหว่างขบวนการ DMA นั้น 8237A-5 จะส่งแอดเดรสของหน่วยความจำออกมาบนบัสแอดเดรส และจะทำให้สัญญาณ IOR หรือ IOW แอคทีฟด้วย ดังนั้นถ้าไม่ทำการดิสเอเบิลพอร์ท I/O ที่ไม่เกี่ยวข้องไว้ ก็อาจจะทำให้พอร์ท I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรส (ซึ่งเป็นแอดเดรสของหน่วยความจำ) นั้น ทำการอ่านหรือส่งข้อมูลออกมาบนบัสข้อมูลทำให้เกิดความผิดพลาดขึ้นได้

2) การจัดสัญญาณบนสล็อตของ IBM PC/XT

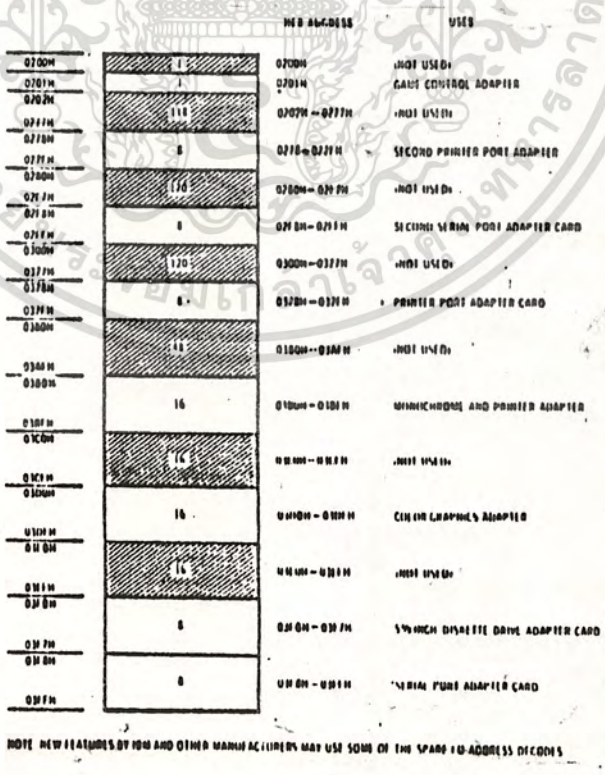
สำหรับใน IBM PC/XT นั้นจะมีสล็อตสำหรับเชื่อมต่อกับวงจรภายนอกได้มากขึ้นคือใน IBM PC/XT จะทำการเพิ่มจำนวนสล็อตบนเมนบอร์ดขึ้นเป็น 8 สล็อต จากเดิมที่มีอยู่เพียง 5 สล็อตบน IBM PC โดยการจัดสัญญาณต่างๆ ในทั้ง 8 สล็อตจะยังคงเหมือนกับใน IBM PC เพียงแต่สัญญาณต่างๆ ที่จะถูกส่งออกมายังขาของสล็อตที่ 8 นั้น จะถูกต่อผ่านวงจรขับกระแส (Buffer) ก่อน และในสล็อตที่ 8 นี้ขา B8 จะถูกใช้งานด้วย โดยจะถูกใช้เป็นขา CARD SLCTD (หรือ Card Selected) ซึ่งขาสัญญาณนี้จะเป็นสัญญาณอินพุทจากวงจรภายนอกที่เสียบอยู่บนสล็อตที่ 8 เพื่อให้วงจรบนเมนบอร์ดทราบว่าคาร์ดที่อยู่บนสล็อตนี้ถูกเลือกใช้งานอยู่ ซึ่งจะทำให้ Driver บนเมนบอร์ดทำการอ่านหรือส่งข้อมูลไปยังสล็อตที่ 8

2.1.2 การจัดแอดเดรสสำหรับพอร์ท I/O ใน IBM PC

ในหัวข้อนี้จะกล่าวถึงวิธี การอ้างและใช้งานแอดเดรสต่างๆของพอร์ท I/O ที่ใช้งานอยู่ใน IBM/PC ในการควบคุมและตรวจสอบสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิพซัพพอร์ทหรือคาร์ดต่างๆ ที่ใช้ระบบของ IBM/PC นั้นจะกระทำโดยผ่านทางพอร์ท I/O ของระบบ ดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกรณ์เหล่านี้ จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ท I/O ต่าง ๆ ของระบบด้วย และเนื่องจากการควบคุมหรือติดต่อกับพอร์ทเหล่านี้ต้องกระทำ โดยการอ้างถึงแอดเดรสของพอร์ท I/O เหล่านั้นโดยตรง เราจึงจำเป็นต้องศึกษาถึงหลักการอ้างแอดเดรสของ 8088 ใน IBM/PC ด้วยสำหรับแอดเดรสของพอร์ท I/O ต่างๆ นั้น จะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 ซึ่งแอดเดรสเหล่านี้เป็นแอดเดรสที่จัดไว้สำหรับพอร์ท I/O โดยเฉพาะคือแยกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ส่วนการส่งข้อมูลให้กับพอร์ทเหล่านี้จะทำได้โดยการใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลนั้นไปยังแอดเดรสของพอร์ทที่ต้องการ และ สำหรับการตรวจสอบหรือการอ่านข้อมูลจากพอร์ท ก็จะทำได้โดยการใช้คำสั่ง IN ของ 8088 อ่านข้อมูลจากแอดเดรสของพอร์ทที่ต้องการเช่นกัน ภายในไมโครโปรเซสเซอร์เบอร์ 8088 นี้จะมีแอดเดรสสำหรับใช้กับพอร์ท I/O อยู่ทั้งสิ้น 65,536 หรือ 64K แอดเดรส (ในขณะที่มีแอดเดรสสำหรับหน่วยความจำอยู่ 1Mbyte) ซึ่งทำให้การอ้างแอดเดรสของพอร์ท I/O ที่ทำงานร่วมกับ 8088 นั้น ต้องใช้จำนวนเส้นแอดเดรสในบัสแอดเดรสทั้งสิ้น 16 เส้นคือ A0-A15 แต่สำหรับใน IBM/PC นี้ ถูกออกแบบมาให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่าง คือ A0-A9 เท่านั้น ดังนั้นในการอ้างถึงแอดเดรสของพอร์ทของอุปกรณ์หรือชิพซัพพอร์ทใด ๆ ที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้นแอดเดรสเพียง 10 เส้นด้วย โดยเส้นแอดเดรสที่เหลือคือ A10-A15 นั้นจะไม่ถูกนำไปใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้นแอดเดรส A10-A15 นี้จะไม่ถูกนำไปใช้งาน แต่ค่าแอดเดรสบนเส้นแอดเดรสเหล่านี้ยังคงเปลี่ยนแปลงตามค่าแอดเดรสของพอร์ทที่กำหนดไว้ในคำสั่ง OUT หรือ IN อยู่ด้วยเพียงแต่ไม่ได้ถูกนำมาตีโค้ดร่วมกับแอดเดรส A0-A9 เท่านั้น ตัวอย่างเช่นในการใช้คำสั่ง OUT ส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0010H นั้น จะให้ผลเหมือนกับการส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0410H, 0810H, 0C10H, ทั้งนี้เนื่องจากแอดเดรส 6 บิทบนไม่ได้ถูกใช้งาน จึงทำให้การเปลี่ยนแปลงค่าแอดเดรสบนเส้นแอดเดรส A10-A15 นั้น ไม่ทำให้เกิดความแตกต่างใด ๆ ขึ้น

เนื่องจากใน IBM/PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้น (คือ A0-A9)ดังนั้นจึงสามารถที่จะอ้างแอดเดรสของพอร์ทได้สูงสุดเพียง 1024 พอร์ท

(จากจำนวน 64K พอร์ท) เท่านั้นนอกจากนี้ในกรณีที่เป็นการอ่านข้อมูลจากพอร์ทของ IBM/PC ข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ททั้ง 1024 พอร์ทออกเป็น 2 ส่วน (ส่วนละ 512 พอร์ท) อีกด้วย กล่าวคือถ้าข้อมูลในบิท A9 เป็น "0" แล้วเราจะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทของอุปกรณ์หรือชิพซัพพอร์ทต่างๆที่อยู่บนเมนบอร์ด (Main Board)ของ IBM/PC เช่น 8253-5, 8237-5 หรือ 8259A เท่านั้น แต่ถ้าข้อมูลในบิท A9 นี้เป็น "1" ก็จะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทที่อยู่บนคาร์ดต่างๆ เท่านั้นจากที่ได้กล่าวมานั้นจะสรุปได้ว่าพอร์ทบน IBM/PC ทั้ง 1024 พอร์ท เราสามารถที่จะเลือกส่งไปยังพอร์ทใดๆ ใน IBM/PC ได้ดังนั้นการเลือกแอดเดรสสำหรับพอร์ทที่อยู่บนคาร์ดจึงสามารถทำได้โดยสะดวกแต่อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงก็คือ ถ้าแอดเดรสที่เราเลือกให้กับพอร์ทนี้ตรงกับค่าแอดเดรสเดิมที่มีอยู่บนเมนบอร์ดแล้ว เมื่อเราทำการส่งข้อมูลให้กับพอร์ทที่อยู่บนคาร์ดด้วย ซึ่งในกรณีเช่นนี้อาจจะก่อให้เกิดความผิดพลาดขึ้นได้เช่นกัน ดังนั้นในการกำหนดค่าแอดเดรสให้กับพอร์ทที่ถูกสร้างขึ้นบนคาร์ดต่างๆ จึงควรจะใช้ค่าแอดเดรสที่แอดเดรสบิท A9 มีค่าเป็น "1" คือ แอดเดรส 0FE00H จนถึง 0FFFFH เท่านั้น (แอดเดรสบิท A10-A15 ไม่ถูกใช้ในการดีโค้ด แต่เพื่อความสะดวกจึงกำหนดให้มีค่าเป็น "1" ในฐานสองทั้งหมด แต่ในการใช้งานจริงอาจเปลี่ยนให้แอดเดรส A10-A15 แต่ละบิทมีค่าเป็น "1" หรือ "0" ก็ได้)



รูปที่ 2.2 การใช้งานแอดเดรสสำหรับพอร์ท I/O บนการ์ดต่างๆ

2.1.3 การใช้งาน 8255 PPI

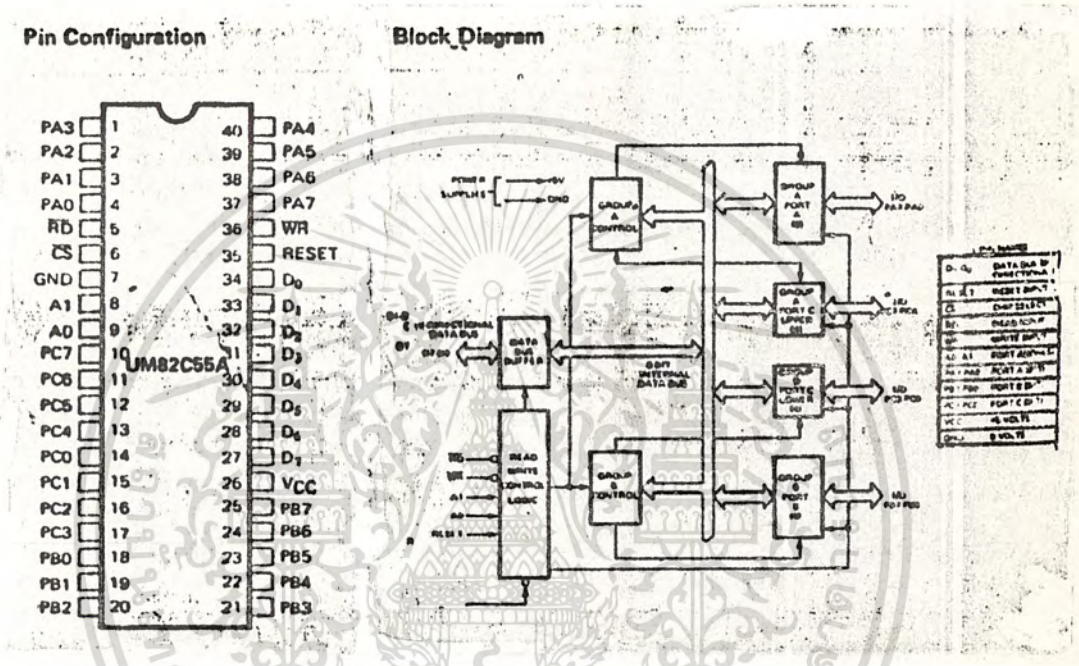
จะพบว่าถ้าต้องการให้เครื่อง Computer ทำการติดต่อกับอุปกรณ์ภายนอก (Peripheral Interface) นั้น เราจะต้องมี IC ที่ทำหน้าที่เป็น I/P และ O/P โดยทั่วไปแล้วมักจะใช้ IC จำพวก TTL (หรือจะเป็น CMOS ก็ได้) ในการ Interface กับ Computer ซึ่งจะพบว่าถ้าต้องการติดต่ออุปกรณ์ I/O ที่มากๆ เช่น 32 CH เท่ากับว่าเราจะต้องมี IC ที่หน้าที่เป็น I/P (เช่นใช้เบอร์ 74LS374 D FLIP FLOP ซึ่งมี I/P DATA 8 I/P) ใช้ 4 ตัว และตัวที่ทำหน้าที่เป็น O/P (เช่น ใช้เบอร์ 74LS244 Buffer 3 state)ใช้ 4 ตัวเช่นกัน รวมกันแล้วจะต้องใช้ถึง 8 ตัวถึงจะทำหน้าที่เป็น I/O 32 CH และจะยุ่งยากในการเลือก I/O แต่ละชุดอีก เราสามารถที่จะขจัดปัญหาในการติดต่อเหล่านั้นให้หมดไปได้โดยใช้ IC สำเร็จรูปที่ทำหน้าที่เป็นทั้ง I/P และ O/P อยู่ในตัวเดียวกันและสามารถที่จะควบคุมได้ด้วยว่าต้องการที่จะให้ Chip ตัวนี้เป็นอะไรก็ได้ (เป็นทั้ง I/P และ O/P,O/P อย่างเดียว, I/P อย่างเดียว) ขึ้นอยู่กับโปรแกรมที่ใช้ควบคุม Chip IC ตัวนี้จะมีประโยชน์มากต่อวงจร Electronic และใช้ได้ทั่วๆ ไปมีชื่อเรียกว่า 8255 PPI (Programmable Peripheral Interface) ใน Project นี้จะใช้ 82C55A เป็น C-MOS High speed ทำงานที่ความถี่ 8 MHz ซึ่งสามารถ Interface กับ IBM ได้ เป็นของบริษัท UMC

1) รายละเอียดเกี่ยวกับ 82C55A

82C55A นี้เป็นอุปกรณ์ประเภท Cmos ซึ่งมาตรฐานเดียวกับ 8255A ภายในจะเป็นเกทจำพวก Cmos ทั้งสิ้น มันเป็นอุปกรณ์จำพวก I/O โดยประกอบไปด้วย 24 I/O (มี 3 Prot A,B,C) และมีด้วยกัน 3 mode ซึ่งสามารถใช้งานร่วมกับ 8088 , 8086 , 8048 , 8051 ฯลฯ รายละเอียดและ Block diageam ภายใน IC เบอร์นี้ดังรูป 2.3

Block กลุ่มแรกที่จะกล่าวถึงนี้ ได้แก่ Block จำนวน 4 Block ที่อยู่ทางด้านขวาสุดของรูปซึ่งจะเป็นส่วนที่เชื่อมต่อกับอุปกรณ์ภายนอกอื่นๆ โดยมีสาย PA0-PA7 ,PB0-PB7 , PC0-PC7 เป็นทางผ่านของข้อมูล ระหว่างอุปกรณ์ภายนอกกับ 82C55A สายสัญญาณเหล่านี้จะถูกแบ่งออกเป็น 3 I/O Port ได้แก่ Port A (PA) ,Port B (PB) และ Port C (PC) Portเหล่านี้แต่ละ Port สามารถเป็นได้ทั้ง Port I/P และ O/P และแต่ละ Port จะมีสายสัญญาณเชื่อมเข้ากับบัสข้อมูลภายในของ 82C55A Block กลุ่มถัดมาได้แก่ Group A control และ Group B control ซึ่งจะเป็นตัว กำหนด

ลักษณะการทำงานของทั้ง 3 I/O Port(82C55A มีลักษณะการทำงานที่แตกต่างกัน 3 mode สามารถกำหนดได้โดยการส่ง control word ให้กับ 82C55A) จากรูปที่ 2.3 จะเห็นว่า Port C ซึ่งประกอบด้วยพอร์ทขนาด 4 bit 2 Prot กลุ่มหนึ่งจะถูกควบคุมโดย Group A Control และอีกกลุ่มหนึ่งจะประกอบไปด้วย Group B Control



รูปที่ 2.3 แสดง Block Diagram และการวางตำแหน่งของ 82C55A

Block กลุ่มสุดท้ายได้แก่ DATA BUS BUFFER และ Read/write control Logic ซึ่ง Bolck เหล่านี้จะเป็นส่วนที่ติดต่อกับ CPU DATA BUS ของ CPU ส่วน Read/Write control Logic จะเป็นส่วนที่ควบคุมให้ข้อมูลเข้าออกจาก Register ภายในตัวที่ถูกต้องและในเวลาที่เหมาะสม

2) รายละเอียดของขาต่างๆ ภายใน 82C55A

ในส่วนนี้ เราจะพิจารณาหน้าที่ต่างๆของขา 82C55A ซึ่งจะได้สามารถใช้งานร่วมกับ CPU ได้ถูกต้อง

รายละเอียดของขาต่างๆมีดังนี้

ก) D0-D7เป็นสายข้อมูล INPUT/OUTPUT แบบสองทิศทาง (Bi-

directionnal Bus) ซึ่งจะเป็นทางผ่านของข้อมูลระหว่าง PORT ต่างๆของ 82C55A กับ CPU โดยสามารถที่จะต่อขนานกับ CPU ได้เลย

ข) CS(Chip Select Input):ขานี้จะทำงาน(Active) ที่สภาวะ Logic"0" เป็นการ Enable 82C55A เพื่อที่จะทำการอ่านหรือเขียนข้อมูล

ค) RD (Read Input): ขานี้จะทำงาน (Active) ที่สภาวะ Logic "0" และสัญญาณ CS ต้องเป็น Logic"0" ข้อมูลนั้นก็จะถูกอ่านเข้ามาสู่ระบบ Bus ข้อมูลของ CPU ได้

ง) WR (Write Input): ขานี้จะทำงาน (Active) ที่สภาวะ Logic "0" และสัญญาณ CS ต้องเป็น Logic"0" ข้อมูลนั้นก็จะถูกเขียนเข้าไปยัง 82C55A ได้

จ) A0-A1 (Address Input): จะเป็นตัวกำหนดการเลือกใช้ Register ภายในของ 82C55A

ฉ) RESET:จะทำงาน (Active) ที่สภาวะ Logic "1"จะทำให้ 82C55A อยู่ในสภาวะ Reset ซึ่งจะทำให้ทุกๆ Port ถูก Setให้อยู่ใน Mode I/P

ช) PA0-PA7,PB0-PB7 ขาสัญญาณเหล่านี้จะถูกใช้เป็นพอร์ท I/O ขนาด 8 บิท ใช้ต่อเข้ากับอุปกรณ์ภายนิกอื่นๆ

ซ) PC0-PC7 ขาสัญญาณนี้ถูกใช้เป็นพอร์ท I/o ขนาด 8 บิท เช่นเดียวกับ PA0-PA7 และ PB0-PB7 แต่กลุ่มของขาสัญญาณเหล่านี้สามารถแบ่งออกเป็น 2 กลุ่ม โดยแต่ละกลุ่มมีขนาด 4 บิท ได้แก่ กลุ่มแรก จะใช้ควบคุม PB0-PB7 และกลุ่มที่ 2 ใช้ควบคุม PA0-PA7

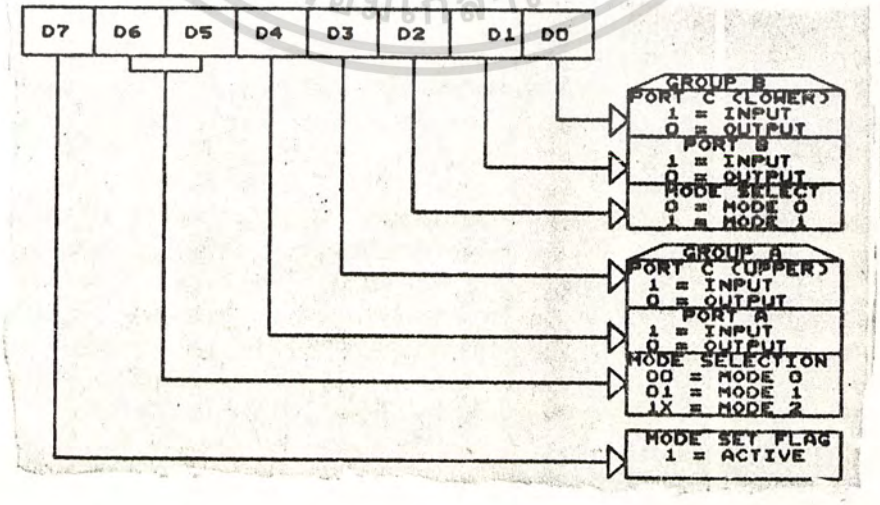
3) การ READ และ WRITE REGISTER ภายใน 82C55A

ขณะนี้เราได้ทำการต่อ 82C55A เข้ากับระบบของ 8088(8086) แล้วต่อไปเราจะศึกษาการโปรแกรมใช้งาน 82C55A เพื่อที่จะให้ทำงานตามที่เราต้องการจะเริ่มต้นพิจารณา Register ภายใน 4 ตัวของ 82C55A สำหรับในตัวอย่างการถอดรหัสของเรานี้ตำแหน่งของ Register จะอยู่ใน Address 300H,301H,302H,303H รายละเอียดต่างๆและขาสัญญาณที่ใช้ในการControl มีดังนี้

ตารางที่ 2.1 แสดงการควบคุม Register ภายในของ 82C55

A1	A0	RD	WR	CS	Input Operation (Read)
0	0	0	1	0	Port A ----- Data Bus
0	1	0	1	0	Port B ----- Data Bus
1	0	0	1	0	Port C ----- Data Bus
1	1	0	1	0	Control Word ---- Data Bus
					OutPut Operation (Write)
0	0	1	0	0	Data bus ---- Port A
0	1	1	0	0	Data Bus ---- Port B
1	0	1	0	0	Data Bus ---- Port C
1	1	1	0	0	Data Bus ---- Control
					Disable Function
X	X	X	X	1	Data Bus Tri-State
X	X	1	1	0	Data Bus Tri-State

หน้าที่ของ Register หมายเลข 0-2 จะถูกกำหนดลักษณะการทำงานจาก Register หมายเลข 3 (Control Register) รูปที่ 2.4 จะแสดงรายละเอียดของแต่ละ Bit ของ Register ควบคุมต่อไปจะกล่าวถึงลักษณะการทำงานของ 82C55A ใน Mode ต่างๆ (มี3 Mode) และการ Program ให้อยู่ในโหมดต่างๆดังต่อไปนี้



รูปที่ 2.4 แสดงรายละเอียดแต่ละบิทของ Register ควบคุมของ 82C55A

Mode ต่างๆ ภายใน 82C55A ซึ่งมีอยู่ด้วยกัน 3 Mode ดังนี้

Mode 0 : Basic Input/Output

Mode 1 : strobed Input/Output

Mode 2 : Bi-Directional Bus

เมื่อขา Resetของ 82C55A มีสภาวะเป็น Logic"1" Port ต่างๆ (Port A-C) ก็จะถูก Set ให้เป็น I/P Mode โดย Bus ภายในจะถูก Hold ไว้หลังจากนั้นถ้าขา Reset ถูกเอาออก (Logic"0") 82C55A ก็ยังคงเป็นสภาวะเดิมจนกว่าเราจะมีการ Initial (การกำหนดให้ก่อนเริ่มทำงาน) ก่อน ใน Project ชิ้นนี้จะกล่าวเฉพาะเพียง Mode เดียวเท่านั้น คือ Mode 0 เพราะเราเพียงแต่ต้องการให้มันเป็นแค่อินพุทและเอาท์พุทแบบธรรมดาเท่านั้น

Mode 0 : รายละเอียดต่างๆใน Mode นี้

1. 8-Bit ขนาด 2 Port และ 4-Bit ขนาด 2 Port
2. ทุกๆ Port สามารถเป็นได้ทั้ง Input และ Output Port
3. Output จะถูก Latch
4. Input จะไม่ถูก Latch
5. มีความแตกต่างระหว่าง Input กับ Output 16 ระดับที่เป็นไปได้

โดยในการ set 82C55A อยู่ใน Mode 0 นั้นเราจะต้องส่งคำสั่งควบคุม (Control Word) ให้แก่ Register ควบคุมก่อน คำสั่งควบคุมนี้จะกำหนดลักษณะการทำงานให้แก่แต่ละ Port ของ 82C55A ตัวอย่างคำสั่งควบคุมที่จะสั่งให้ 82C55A ทำงานใน Mode 0 นี้ได้แก่

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

จากรูปที่ 2.4 เราจะได้ว่า

Bit D_7 เป็นตัวกำหนดว่าเป็นคำสั่งควบคุม (Control Word)

Bit D_6 และ D_5 กำหนด Mode การทำงานของ Port A

D_6, D_5 มีค่าเป็น 0 แสดงว่าอยู่ใน Mode 0

Bit D_4 = "0" กำหนดให้ Port A เป็น Output Port

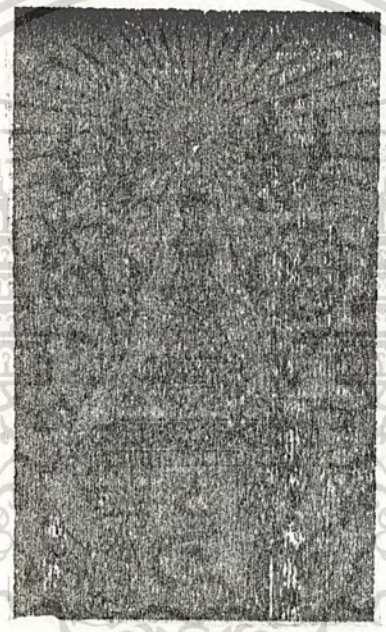
Bit D_3 = "0" กำหนดให้ Port C 4 Bit บนเป็น Output Port

Bit D_2 = "0" Set Mode ของ Port B ให้อยู่ใน Mode 0
Bit D_1 = "0" Set Port B เป็น Port Output
Bit D_0 = "0" Set Port C ให้ 4 Bit ล่างเป็น Port Output

คำสั่งควบคุมนี้จะกำหนดให้ Port ทั้ง3 ของ 82C55A ทำงานอยู่ใน Mode 0 และเป็น Port Output จะได้สายสัญญาณซึ่งสามารถติดต่อกับอุปกรณ์ภายนอกได้ถึง 24 สายในการที่จะทำให้ 82C55A ทำงานตามที่เราต้องการต่อไปก็ขึ้นอยู่กับการเขียน Soft-ware และส่ง Data (Out Port)ออกไปยังตำแหน่งที่ต้องการ เมื่อเราทำการ Initial Port(โดยการ Out ไปที่ Port 303H) แล้วต่อไปเราก็จะส่ง Data ไปยังแต่ละ Port(ในที่นี้เราทำให้ทุก Port ทำหน้าที่เป็น Output หมด)โดย Port A Out ไปที่ 300H, Port B Out ไปที่ 301H,Port C Out ไปที่ 302H นั่นเอง ซึ่งหลังจากทำการ Run แล้ว Port ต่างๆก็จะมีข้อมูลปรากฏตามที่เราใส่ไป

2.1.4 ความรู้เกี่ยวกับไอซี PAL

IC PAL นับได้ว่าเป็น อุปกรณ์ที่มีประโยชน์มากชนิดหนึ่ง ซึ่งลักษณะโดยทั่วๆไปแล้วมันมีลักษณะที่คล้ายอุปกรณ์จำพวก PROM ซึ่งก็เป็นที่รู้จักกันมานานแล้วแต่มีลักษณะแตกต่างกันที่โครงสร้างภายในเท่านั้น IC PAL มีลักษณะโครงสร้างภายในที่ตรงข้ามกับ PROM คืออินพุทของ AND เป็นแบบโปรแกรมได้ ส่วนอินพุทของ OR จะคงที่ดังตัวอย่างรูปที่ 2.5 ซึ่งแสดงโครงสร้างภายในของ PAL ขนาด 4-อินพุท-4 เอ๊าท์พุท จากตัวอย่างนี้สังเกตว่าสัญญาณอินพุต I_0-I_3



รูปที่ 2.5 แสดงโครงสร้างภายในของ PAL ขนาด 4-อินพุท 4-เอาท์พุท

จะส่งผ่านบัฟเฟอร์ และอินเวอร์เตอร์ออกมาเป็นสัญญาณ 8 เส้น (I_0-I_3, I_0-I_3) ป้อนเข้า AND ทุกตัว ซึ่ง AND ทุกตัวจะต้องมีอินพุตอยู่ 8 ขา โดยที่สามารถโปรแกรมได้ว่าจะต่อเข้ากับ I_0-I_3 เส้นใด ส่วนเอาต์พุตของ And จะถูกจัดแบ่งเป็น 4 ชุดป้อนเข้า OR อย่างตายตัวคือเอาท์พุต 4 ตัวบนจะป้อนเข้า OR ตัวแรก ถัดลงมา 4 ตัวจะป้อนเข้า OR ตัวที่สอง เป็นเช่นนี้ลงมาเรื่อยๆ ในกรณีนี้หมายถึงว่า OR จะต้องมีอินพุตอยู่ 4 ขาเท่านั้นและตามความหมายของเบอร์ IC PAL ที่กำหนดขึ้นจะมีดังนี้ ในที่นี้จะขอยกตัวอย่างเบอร์ที่ใช้งานใน PROJECT นี้

PAL 16 L 8

ชุดที่ 1 2 3 4

ชุดที่1 บอกว่าเป็น IC PAL
ชุดที่2 คือจำนวนอินพุท (มี 8,10,12,14,16,20)
ชุดที่3 คือชนิดของเอาท์พุท (H,L,C,R,X,A)
ชุดที่4 คือจำนวนของเอาท์พุท (2,4,6,8)
ซึ่งรายละเอียดมีดังต่อไปนี้

ก) จำนวนอินพุตเราสามารถเลือกได้หลายขนาดซึ่งขึ้นอยู่กับความต้องการ มี 8,10,12,14,16,18,20 ขา
ข) จำนวนเอาท์พุต มี 2,4,6,8 ขา
ค) บัฟเฟอร์ทางด้านเอาท์พุทสามารถจะป้อนกลับเข้ามาเป็นอินพุตได้
ง) โปรแกรมได้ทั้งทางอินพุตและเอาต์พุท
จ) ทำหน้าที่คำนวณทางคณิตศาสตร์ได้ขึ้นอยู่กับโปรแกรมที่ต้องการ

การกำหนดเบอร์ของ PAL จะแสดงถึงจำนวนอินพุตและเอาต์พุตและและชนิดของเอาต์พุท ดังแสดงความหมายของเบอร์ที่กล่าวมาแล้ว ตัวอักษรซึ่งบอกชนิดของเอาต์พุทมีรายละเอียดดังนี้

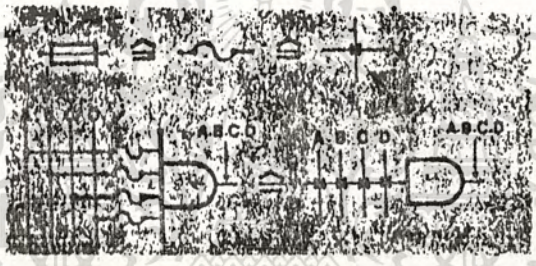
H แอคทีฟที่ Logic'1'
L แอคทีฟที่ Logic'0'
C มีให้เลือกทั้งแอคทีฟที่ Logic '1' และ Logic '0'
R เป็นรีจิสเตอร์ หมายถึงรักษาสถานะเอาต์พุตให้ค้างเอาไว้ได้ด้วย พลิบฟลอบและเอาต์พุทยังป้อนย้อนกลับมาที่อินพุตของ AND ได้ (โปรแกรมได้ด้วย)
X เป็นรีจิสเตอร์แบบ EX-OR
A เป็นรีจิสเตอร์คำนวณทางคณิตศาสตร์ได้

1) สัญญลักษณ์ของลอจิกใน IC PAL

เนื่องจากวงจรลอจิกภายในตัว PAL ค่อนข้างซับซ้อนและมีเป็นจำนวนมาก การเขียนสัญญลักษณ์แบบมาตรฐานทั่วไปเพื่อแสดงวงจรภายใน PAL จึงทำให้ดูสับสนวุ่นวายและต้องเปลืองเนื้อที่มาก โรงงานผู้ผลิตส่วนใหญ่จึงนิยมใช้สัญญลักษณ์ที่กำหนดขึ้นมาใหม่ เพื่อให้เขียนแสดงวงจรภายในได้สะดวกขึ้นและอ่านความหมายจากวงจรได้ง่ายขึ้น ดังนี้

รูปที่ 2.6 สัญญลักษณ์ของ Gate ในไอซี PAL

สัญญาณจากขาอินพุตมักจะต้องป้อนเข้าบัฟเฟอร์ 2 ตัวก่อนเพื่อให้ได้ 2 สัญญาณที่ตรงกันข้ามกัน คือบัฟเฟอร์ตัวหนึ่งจะไม่กลับลอจิกแต่อีกตัวหนึ่งจะกลับลอจิกเพื่อให้การเขียนวงจรลอจิกภายใน PAL ทำได้ง่ายขึ้นแทนที่จะต้องเขียนสัญญลักษณ์ของบัฟเฟอร์ทั้ง 2 ตัว จึงรวมมาเขียนเป็นตัวเดียวกันเลยโดยให้มีเอาต์พุต 2 สัญญาณดังรูปที่ 2.6



รูปที่ 2.7 แสดงสัญญลักษณ์ของอินพุท Gate แบบ Matrix

ขาอินพุตของเกตมักจะมีขาอินพุตอยู่หลายสัญญาณ และจะต่อเข้ากับฟิวส์ในลักษณะแมทริกซ์ เพื่อให้เขียนวงจรได้ง่ายขึ้นจึงใช้เครื่องหมายกากบาทแสดงถึงว่ามีฟิวส์ต่ออยู่ระหว่างขาอินพุตของเกต กับขาสัญญาณทางแนวตั้งตามตำแหน่งของเครื่องหมายกากบาทนั้นๆ จึงทำให้เขียนขาสัญญาณอินพุทของเกตเพียงเส้นเดียว แสดงจำนวนขาสัญญาณอินพุตของเกตหลายสัญญาณได้ (จำนวนอินพุทของเกต = จำนวนขาสัญญาณอินพุตทางแนวตั้ง) ดังรูปที่ 2.7

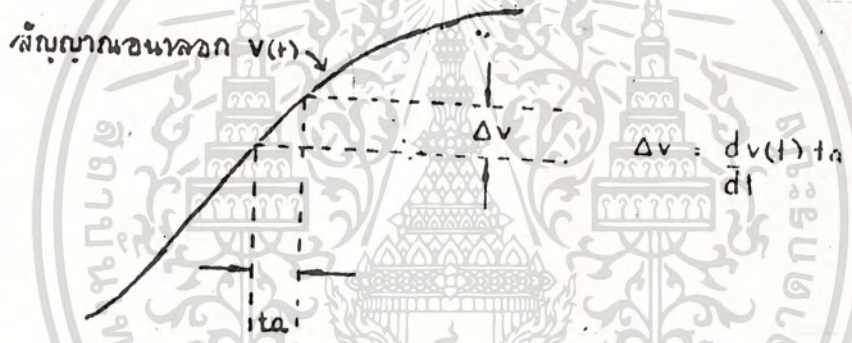
รูปที่ 2.8 แสดงสัญญลักษณ์แทนฟิวส์ยังคงอยู่

ในกรณีที่ฟิวส์ทุกตัวที่ต่ออยู่เข้ากับ อินพุตของเกตไม่ถูกระเบิดทิ้งไปเลยแม้แต่ตัวเดียว จะเขียนเครื่องหมายกากบาทเพียงตัวเดียวเข้าที่สัญลักษณ์ของเกตเลย ในกรณีเช่นนี้ยังแสดงถึงว่าอินพุทของเกตตัวนั้นจะเป็น '0' เสมอ ดังรูปที่ 2.8

2.1.5 การแปลงสัญญาณแอนะลอกเป็นดิจิตอล และ ดิจิตอลเป็นแอนะลอก

1) ทฤษฎีการสุ่มตัวอย่าง

ในการแปลงสัญญาณแอนะลอกเป็นรหัสดิจิตอลนั้น ADC จะต้องใช้เวลาช่วงหนึ่งในการจัดการ ซึ่งช่วงเวลาดังกล่าวนั้นขึ้นอยู่กับหลายแฟคเตอร์ เช่น ความละเอียดของการเปลี่ยนสัญญาณ เทคนิคของการแปลงสัญญาณ และความเร็วในการทำงานของอุปกรณ์รวมอื่นๆ ความเร็วในการแปลงสัญญาณนี้จำเป็นสำหรับการประยุกต์ใช้งานเฉพาะอย่าง และความแม่นยำที่ต้องการ Aperture Time เป็นช่วงเวลาในการแปลงสัญญาณ ซึ่งคำว่า Aperture Time โดยทั่วไปหมายถึงช่วงเวลาที่เกิดความไม่แน่นอนในการวัด และผลก็คือเกิดผิดพลาดต่อค่าที่วัดได้



รูปที่ 2.9 แสดงข้อผิดพลาดจากการวัดใน Aperture Time

ดังนั้นหากเวลาที่ ADC ใช้ในการเปลี่ยนสัญญาณในเวลา t_a นี้ รหัสดิจิตอลที่ได้อาจจะตรงกับขนาดของ สัญญาณแอนะลอกค่าใดค่าหนึ่งในช่วงนี้และส่วนอื่นๆ ที่เหลือคือความผิดพลาดที่เกิดขึ้น ซึ่งแน่นอนในบางครั้งเป็นไปได้ที่รหัสดิจิตอลจะตรงกับค่าของแอนะลอกที่ถูกต้อง

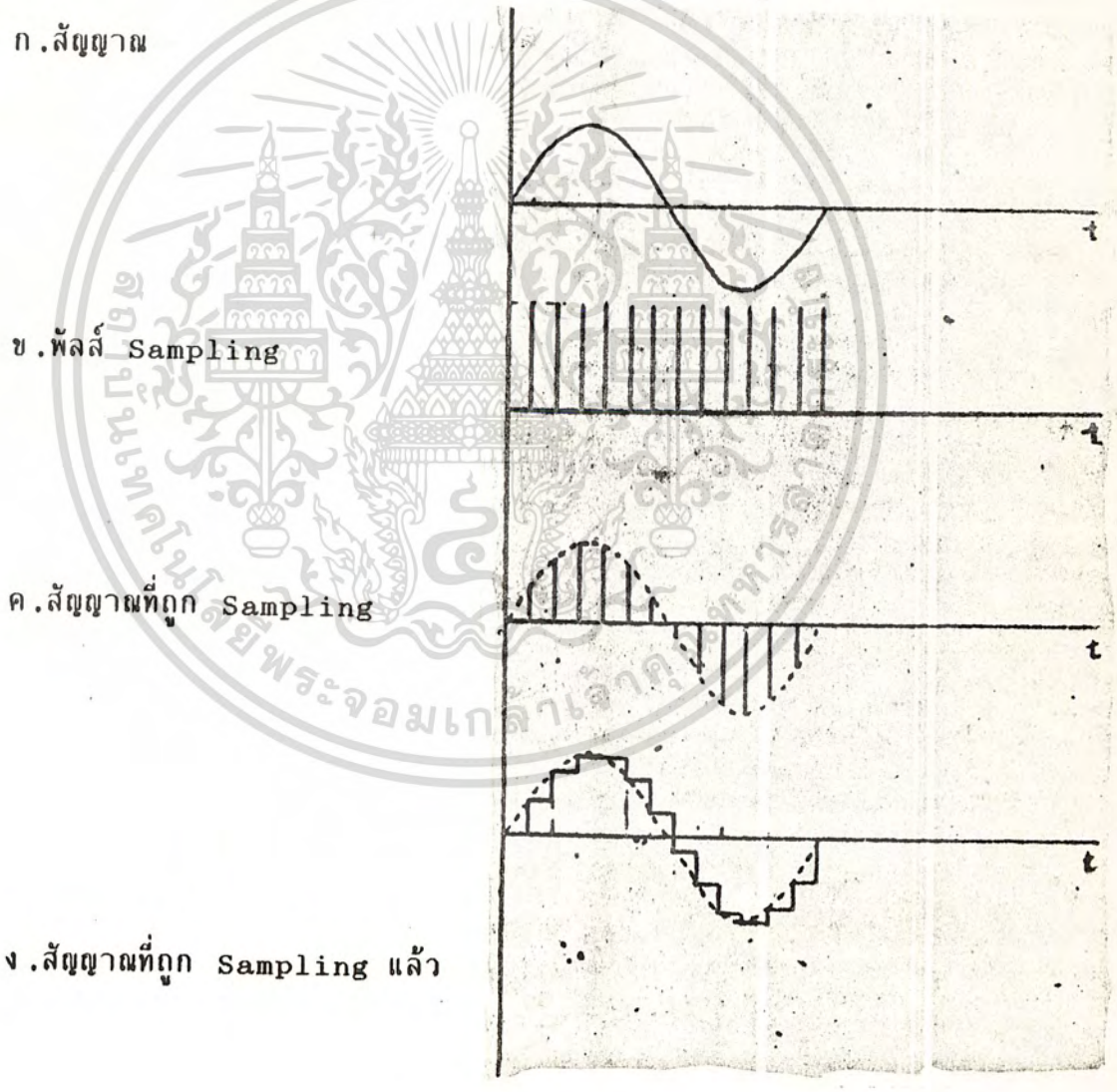
2) Sample and Hold และ Aperture Error

วงจร Sample And Hold จะทำการสุ่มสัญญาณอินพุท และนำสัญญาณที่สุ่มนั้นมาเก็บหรือโฮลด์ไว้ในเวลาหนึ่งได้ ซึ่งส่วนใหญ่จะใช้การประจุแรงดันนั้นไว้ในตัวเก็บประจุที่รั่วไหลต่ำ ดังนั้นเมื่อแรงดันอินพุทสามารถคงอยู่ได้นานพอ ADC จึงไม่จำเป็นต้องมีเวลาในการแปลง (Conversion Time) อย่างรวดเร็วนัก Aperture Time ของ Sample And Hold คือเวลาตั้งแต่เริ่มสุ่มสัญญาณ ตัวเก็บประจุค่าแรงดันจนถึงค่าที่สุ่ม ซึ่งสำหรับ Sample

And Hold แล้ว Aperture Time ขึ้นอยู่กับแบนด์วิดธ์ และ Switching Time ของอุปกรณ์แอคตีพที่ใช้ในวงจร ซึ่งหาและสร้างได้ง่ายตลอดจนราคาถูกกว่าการสร้าง ADC ความเร็วสูง

3) ระบบ Sample ข้อมูลและทฤษฎีการ Sample

ในระบบการ Sampling สัญญาณแอนะลอกจะถูกสุ่มเป็นระยะๆ คงที่ตาม รูปที่ 2.10 กลุ่มของสัญญาณสุ่มจะแทนสวิทช์ที่ทำงานด้วยความเร็วสูงซึ่งจะทำการตัดต่อสัญญาณแอนะลอกในช่วงเวลาอันสั้น



รูปที่ 2.10 การ Sampling สัญญาณ

ผลของการสุ่มสัญญาณด้วยความเร็ว จะเสมือนกับการคูณขบวนสัญญาณ

พัลส์กับสัญญาณแอนะลอก ซึ่งจะได้สัญญาณที่มอดดูเลท (Modulate) ระหว่างขบวนพัลส์ กับ สัญญาณแอนะลอกดังแสดงใน รูปที่ 2.10ค โดยสัญญาณแอนะลอกจะขี่มาบนพัลส์ ถ้าหากเอาตัวเก็บประจุแทนสวิทช์แล้วสัญญาณแอนะลอกที่ถูกสุ่มจะถูก Hold ไว้ในตัวเก็บประจุคอยจนกว่าสัญญาณค่าใหม่ถูกสุ่มเข้ามาใหม่ ลักษณะของเอาท์พุทที่ได้แสดงในรูปที่ 2.10ง

มีปัญหาที่ว่าอัตราการสุ่มสัญญาณนั้นควรมีค่าเท่าใดที่จะไม่ทำให้ ข้อมูลเสียไป เมื่อสัญญาณนั้นถูกเปลี่ยนกลับมาเช่นเดิม คำตอบคือขึ้นอยู่กับความถี่ของสัญญาณแอนะลอกและทฤษฎีของการสุ่มกล่าวไว้ว่า "ถ้าสัญญาณต่อเนื่องซึ่งมีความถี่และฮาร์โมนิคส์ไม่เกิน f_s แล้ว สัญญาณดังกล่าวจะสามารถเปลี่ยนกลับมาได้อย่างเดิมโดยไม่สูญเสียรายละเอียด หรือผิดเพี้ยนไป ถ้าอัตราการสุ่มไม่น้อยกว่า $2f_s$ ต่อวินาที"

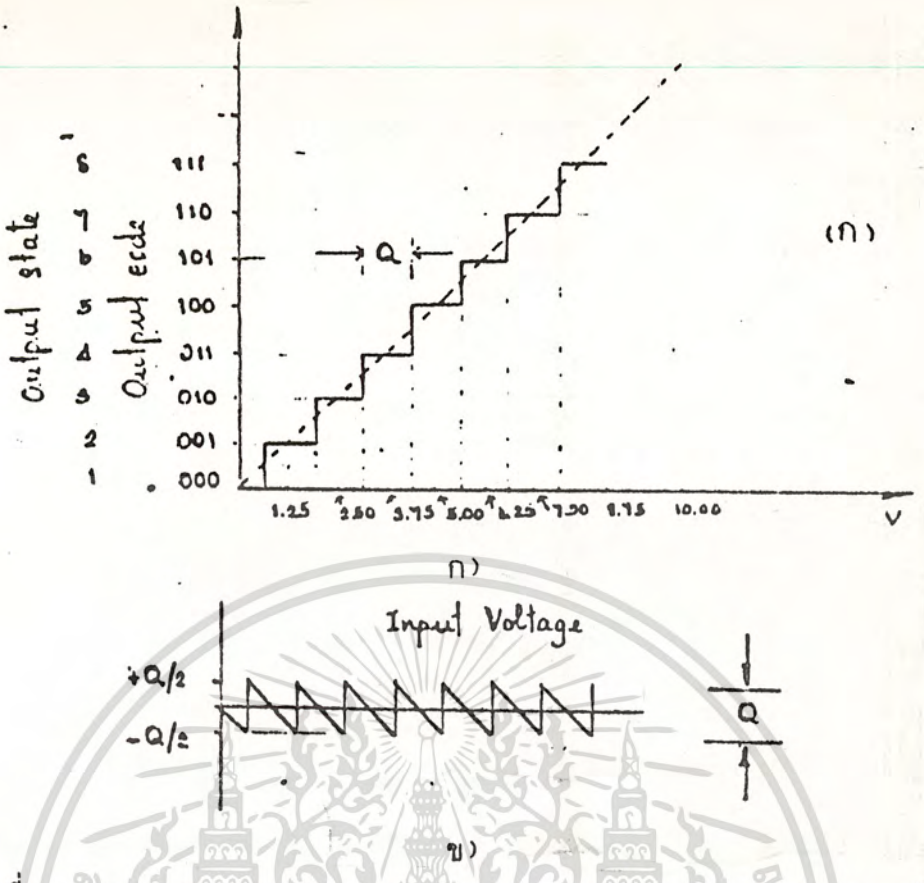
4) Quantizing Theory

Quantizingเป็นขบวนการที่แปลงสัญญาณแอนะลอกเป็นสัญญาณที่ไม่ต่อเนื่อง (Discrete Signal) ซึ่งจากนั้นก็จะผ่านขบวนการ Coding จัดให้สัญญาณที่ไม่ต่อเนื่องนั้นอยู่ในรูปที่ง่ายต่อการประมวลทำความเข้าใจ และเป็นสัดส่วนสัมพันธ์กับสัญญาณแอนะลอกเช่น อยู่ในรูปของรหัสไบนารี เป็นต้น หากนำเอาขนาดของสัญญาณแอนะลอกและสัญญาณดิจิตอลที่สัมพันธ์จากการ Quantize และการเข้ารหัส (Encode) แล้วมาเขียนกราฟก็จะได้กราฟแสดง Quantize Transfer Function ดังแสดงในรูปที่ 2.11

จุดสำคัญที่เกี่ยวข้องกับกราฟ Transfer Function ในรูปที่ 2.11 อันแรก ได้แก่ รีโซลูชัน (Resolution) ของ Quantizer ซึ่งกำหนดได้จากจำนวนบิทของรหัสดิจิตอล หรือจากกราฟ คือขนาดความกว้างของ Step ทางแกนแอนะลอกว่าเป็นสัดส่วนเท่าใดระหว่างเต็มสเกลแอนะลอกกับค่า 2^n โดยที่ nเป็นจำนวนบิทของรหัสดิจิตอล

5) Quantizer Resolution และ Error

ในแต่ละสถานะของ สัญญาณดิจิตอลเอาท์พุทจะแทนขนาดของสัญญาณแอนะลอกค่าใดค่าหนึ่งในช่วงเล็กๆ ระหว่างจุดแบ่งระดับสองจุด เรียกช่วงเล็กๆ นี้ว่า เป็นขนาดหนึ่ง Analog Quantization หรือหนึ่งควันตัม (Quantum) หรือ 1 LSB (Least Sinificant Bit) ของการแปลงสัญญาณ ตัวอย่างในรูปที่ 2.11ก ควันตัม คือ 1.25 โวลต์ ค่านี้ได้จากการคำนวณดังนี้



รูปที่ 2.11 Transfer Function ของ Quantize 3 บิท

$$Q = (FSR)/(2^n)$$

FSR คือ ช่วงเต็มสเกลของแรงดันแอนะลอก (Full Scale Range)

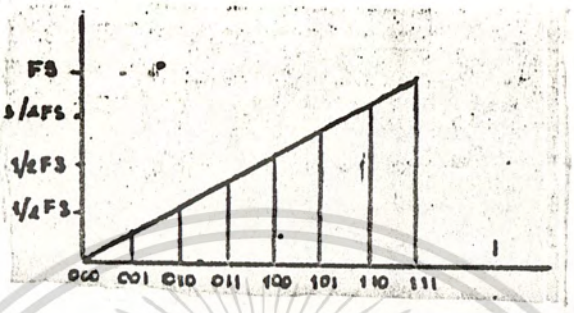
n คือ จำนวนบิทของรหัสดิจิตอล

จากสมการจะเห็นว่าหากจำนวนบิทมากขนาดของ Quantizing ก็จะลดลงและถ้าให้อินพุทของ Quantizer กวาดไปตลอดช่วงของสัญญาณแอนะลอกจะเห็นช่วงของผลต่างแอนะลอกอินพุทและดิจิตอลเอาท์พุทเป็นช่วงซึ่งพล็อตได้เป็นรูปฟันเลื่อย ดังรูปที่ 2.11ข เรียกว่า Quantizing Error ซึ่ง Error นั้นคือ 1 ช่วงสัญญาณแอนะลอกแปลงเป็นรหัสดิจิตอล 1 สถานะ Error นี้เป็นธรรมชาติของ Quantizing ซึ่งจะทำการแก้ไขไม่ได้นอกจากการเพิ่มจำนวนบิทของ Quantizer ให้มากขึ้น และเอาท์พุท Error จะอยู่ระหว่าง 0 ถึง Q/2

6) วงจร DAC (Digital To Analog Converter)

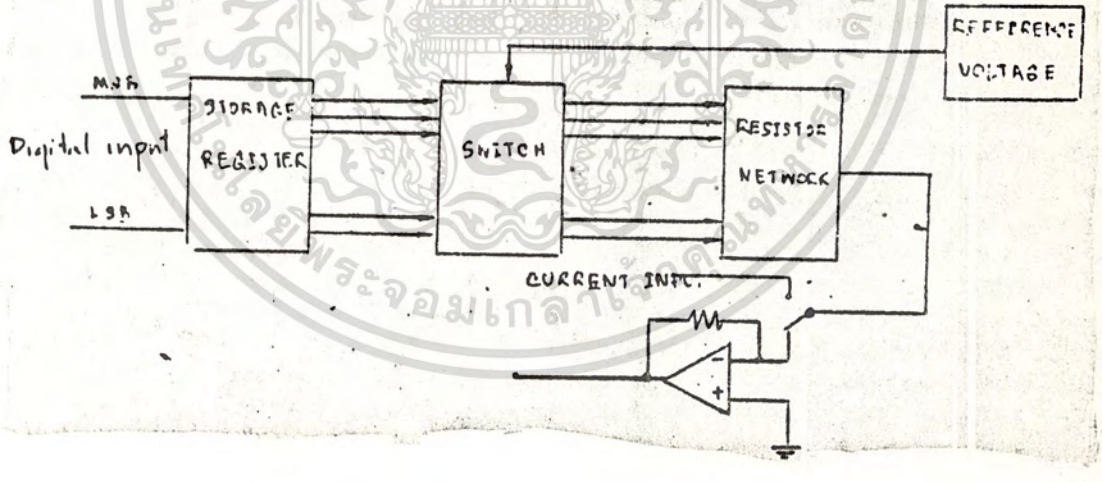
DAC นับเป็นอุปกรณ์สำคัญที่ทำให้ดิจิตอลคอมพิวเตอร์เชื่อมโยงกับอุปกรณ์หรือวงจรแอนะลอกอื่นๆ ตัวอย่างการใช้งาน DAC คือ ระบบแสดงผลบนจอภาพ ระบบสังเคราะห์เสียง เป็นต้น และที่สำคัญ DAC ยังเป็นส่วนประกอบที่

สำคัญในระบบ ADC ที่ใช้กันอยู่ในสมัยปัจจุบัน รูปที่ 2.12 แสดงทรานสเฟอร์ฟังชันของ DAC ขนาด 3 บิท จะเห็นว่ารหัสดิจิตอลอินพุท 1 Word จะแปลงเป็นแรงดันแอนะลอก 1 ค่า



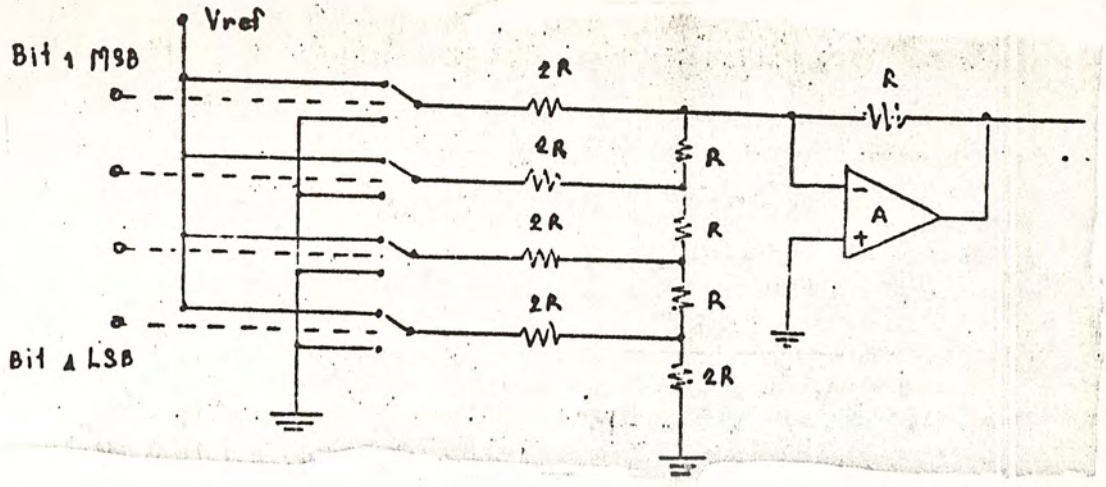
รูปที่ 2.12 ทรานสเฟอร์ฟังชันของ DAC 3 บิทตามทฤษฎี

ลักษณะการจัดวงจร DAC เป็นลักษณะดังรูปที่ 2.13



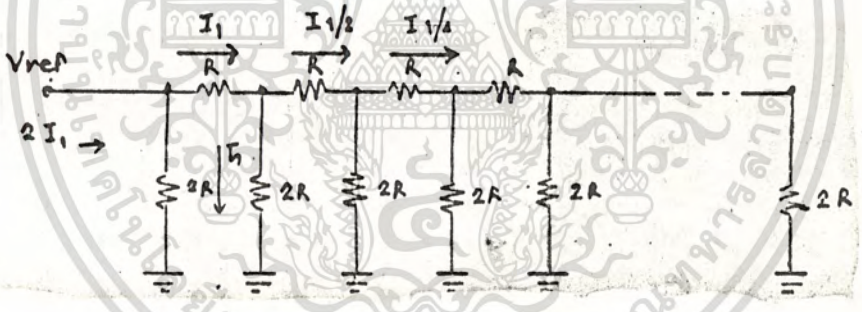
รูปที่ 2.13 บล็อกไดอะแกรมของ DAC

วงจร DAC ที่ใช้กันมีหลายแบบแต่ในที่นี้จะขอกล่าวถึงวงจร DAC แบบ R-2R Ladder ซึ่งเป็นแบบที่ใช้ในบริญญานิพนธ์นี้ ลักษณะของวงจรแสดงดังรูปที่ 2.14



รูปที่ 2.14 วงจร DAC แบบ R-2R Ladder ขนาด 4 บิท

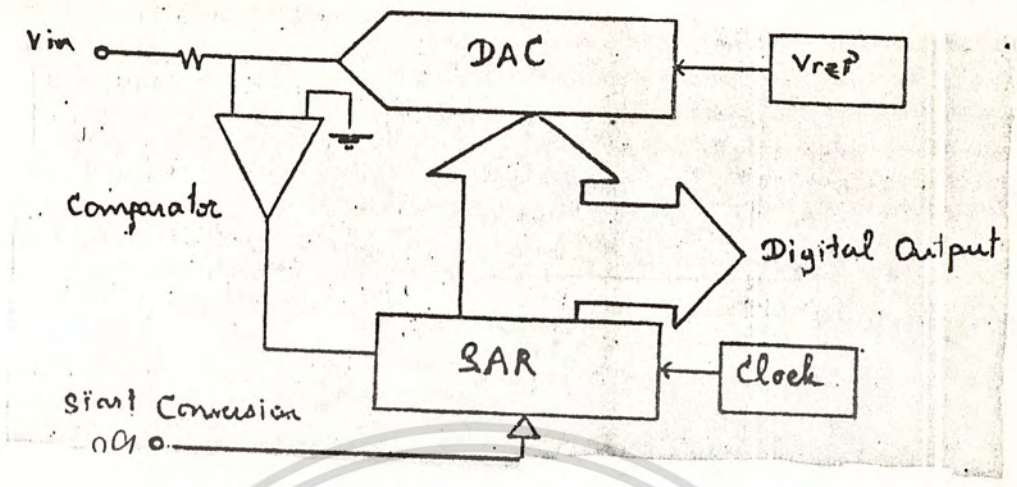
ในวงจรนี้สวิทช์จะตัดต่อให้แรงดันอ้านอิงต่อเข้ากับวงจรแลดเดอร์หรือต่อแลดเดอร์ลงกราวด์ที่ขา 2R จะเห็นได้ว่าสวิทช์อินพุทรีซีสเตอร์ (2R) มองเข้าไปจะเห็นคู่ของรีซีสเตอร์ระหว่างจุดต่อ R-2R ที่ติดกันกระแสจะถูกบั่นทอนไปในอัตรา 2/1 ซึ่งสอดคล้องกับรหัสไบนารี ดังแสดงในรูปที่ 2.15



รูปที่ 2.15 วงจรรีซีสทีพแลดเดอร์ DAC

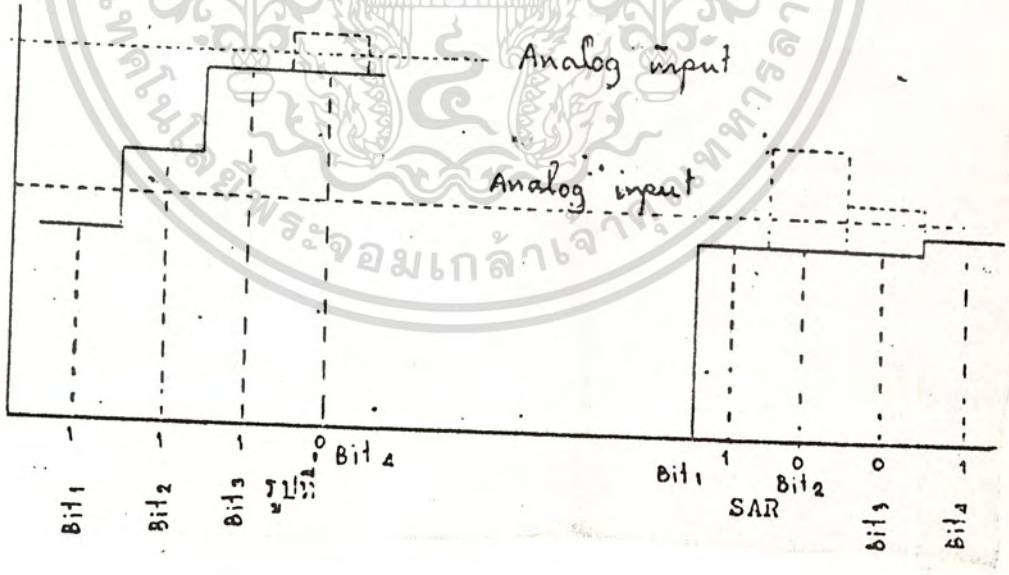
7) Analog To Digital Converter Circuit

ลักษณะการจัดวงจร ADC มีหลายแบบแต่ที่นิยมใช้มีอยู่ไม่กี่แบบ และส่วนใหญ่จะอยู่ในรูปของวงจรรวม ในที่นี้จะกล่าวเฉพาะ ADC แบบการประมาณซัคเซสซีฟ (Successive Approximation ADC) วงจร ADC ชนิดนี้เป็นเทคนิคที่ได้รับความนิยมในงานประยุกต์ที่ต้องการความเร็วสูงและปานกลาง การจัดวงจรจะคล้ายกันแบบเคาท์เตอร์ที่ทำงานในลักษณะป้อนกลับ บล็อกไดอะแกรมในรูปที่ 2.16 แสดงฟังก์ชัน



รูปที่ 2.16 บล็อกไดอะแกรมของ Successive Approximation Converter

ต่างๆใน ADC ชนิดนี้คอมพอเรเตอร์จะคอยเปรียบเทียบเอาท์พุทจาก DAC กับแอนะลอกอินพุท V_{in} เอาท์พุทจะไปควบคุม Successive Approximation Register (SAR) ซึ่งเป็นไอซี MSI ที่ได้รับการออกแบบเป็นพิเศษเพื่อทำหน้าที่นี้โดยเฉพาะการทำงานของ SAR เป็นดังนี้



รูปที่ 2.17 ไทมิ่งไดอะแกรมของ SAR

รูปที่ 2.17 แสดงไทมิ่งไดอะแกรมของ ADC ที่มีระดับแอนะลอกอินพุท 1 และ 2 เมื่อสัญญาณนาฬิกา (Clock) เข้าไป 1 ลูกจะทำให้ MSB (most significant Bit) เป็น1 บิทอื่นๆ ยังคงเป็นศูนย์ DAC จะเปลี่ยนเอาท์พุทของ SAR เป็นแอนะลอกเปรียบเทียบกับสัญญาณแอนะลอกอินพุท ถ้าผลการเปรียบเทียบ

ที่คอมพาเรเตอร์ว่าน้อยกว่าอินพุทให้คงบิทนั้นเป็น 1 แต่ถ้ามากกว่าจะให้บิทนั้นเป็น 0 จากนั้นจะทำการทดสอบบิทถัดไปโดยทำให้เป็น 1 หากผลของสองบิทหรือบิทหลังมากกว่าก็จะทำให้บิทนั้นเป็น 0 แต่ถ้าน้อยกว่าให้คง 1 ไว้แล้วทดสอบบิทถัดไปกรรมวิธีดังกล่าวจะกระทำต่อไปจนครบทุกบิท หรือจนกว่าเอาท์พุทจะต่างจาก V_{2n} ไม่เกิน 1 LSB (least significant Bit) ในตัวอย่างแสดงการทำงานเมื่อ V_{2n} ต่ำลงมาอีกระดับหนึ่ง

มีข้อจำกัดประการหนึ่งสำหรับการทำงาน คือสัญญาณแอนะลอกอินพุทจะต้องคงที่ในช่วงเวลาที่ทำการเปลี่ยนแปลงสัญญาณ โดยเปลี่ยนไม่เกิน 1/2 LSB ในช่วงสุดท้ายของการเปลี่ยนสัญญาณดิจิตอล เอาท์พุทจะออกมาขนานกันทุกบิท แต่บางแบบจะให้เอาท์พุทออกมาในลักษณะอนุกรม

วงจร ADC แบบนี้สามารถทำงานได้สองโหมดคือโหมดที่ทำงานอิสระ (Free Run) และโหมดที่รอคำสั่งเริ่มต้นการทำงาน (Start Conversion) จากภายนอก เวลาที่ใช้ในการเปลี่ยนสัญญาณจะใช้ N+1 ลูกของพัลส์ Clock โดย Clock ลูกแรกจะใช้ในการรีเซ็ทรีจีสเตอร์ภายใน ADC เทคนิคนี้จะเที่ยงตรงมากหากมีแรงดันอ้างอิงและ D/A รวมอยู่บนชิพเดียวกันแต่คุณภาพของระบบจะแย่ลงหาก DAC ไม่มีคุณภาพ

2.1.6 การใช้งาน IC 8253

ในการออกแบบวงจร Interface โดยทั่วไป มักจะมีความจำเป็นต้องใช้ฐานเวลาหรือวงจรนับ (Counter) ต่างๆ ในการทำงานของวงจรด้วย ซึ่งจะมี Chip ตัวหนึ่งที่ทำหน้าที่เป็นวงจรฐานเวลา และ Counter โดยสามารถควบคุมได้ Chip Support ตัวนี้เบอร์ 8253 (Programmable Interval Timer) สำหรับ 8253 นี้จะมี Channel ที่ใช้งานเป็นวงจรสร้างฐานเวลาหรือวงจรนับอยู่ 3 Channel คือ CH 0, CH 1, CH 2 โดยที่เราสามารถจะ Program และใช้งาน CH ทั้ง 3 นี้แยกกันได้อย่างอิสระ

1) สายสัญญาณ Clock ,Gate และ Out ของ Counter

Counter แต่ละตัวใน Block Diagram ของรูปที่ 2.18 จะมีสายสัญญาณต่อกันแต่ละ Block อยู่ 3 เส้น โดยสายสัญญาณนี้มีชื่อว่า Clock และ Gate ใช้เป็น I/P ส่วน Out ใช้เป็น O/P หน้าที่ในการทำงานของสายเหล่านี้เปลี่ยนแปลงได้ขึ้นอยู่กับว่า อุปกรณ์เหล่านี้ถูกกำหนดหน้าที่การทำงานเบื้องต้นไว้อย่างไรหรือถูก Program มาอย่างไร

Clock : เป็น I/P ที่ใช้ป้อนสัญญาณ Clock ให้แก่ Counter ซึ่ง Counter ในที่นี้มีขนาด 16 บิท ความถี่ของสัญญาณ CLK ที่มากที่สุดที่ป้อนให้แก่ Counter เป็น 2.6 MHz และความถี่ของ CLK ที่น้อยที่สุดเป็น 0 Hz (DC) หรือ Static Operation

Gate : เป็นสายสัญญาณ I/P ที่ทำตัวเสมือน Gate ที่จะยอมหรือไม่ยอมให้สัญญาณ CLK ผ่านเข้าไปยัง Counter และ Gate สามารถใช้เป็นสายสัญญาณที่ป้อน Pulse เพื่อกระตุ้นให้ Counter เริ่มนับซึ่งขึ้นอยู่กับ Mode ที่ Program ให้กับ Counter

Out : เป็นสายสัญญาณ เอ้าท์พุทของ Counter ซึ่งการทำงานขึ้นอยู่กับการ Program Internal Register ของ 8253

2) Internal Register ของ 8253

ในรูปที่ 2.18 นั้นเป็น Register ภายในของ 8253 ในขั้นต้นนี้จะพิจารณา Mode Word Register ก่อน Register นี้เป็นตัวกำหนดการทำงานทั้งหมดของ 8253 จากที่ทราบมาแล้วว่าการทำงานของแต่ละ Counter ภายใน 8253 เป็นอิสระต่อกันอย่างสมบูรณ์ ทำให้สามารถ Program การทำงานใน Counter แต่ละตัวได้โดยการให้ข้อมูลที่ถูกต้องกับ Mode Word Register ต่อไปนี้ เราจะพิจารณา Register ภายในทั้ง 4 ดังรูปที่ 2.18

	RD	WR	A0	A1	
COUNTER 0	1	0	0	0	LOAD COUNTER 0
	0	1	0	0	READ COUNTER 0
COUNTER 1	1	0	0	1	LOAD COUNTER 1
	0	1	0	1	READ COUNTER 1
COUNTER 2	1	0	1	0	LOAD COUNTER 2
	0	1	1	0	READ COUNTER 2
MODE WORD OR	1	0	1	1	WRITE MODE WORD
CONTROL WORD	0	1	1	1	NO-OPERATE

รูปที่ 2.18 แสดงลอจิกของ Register ภายใน 8253 ที่จะ Program ให้กับ Counter

Control Word Register: เป็น Register ที่ใช้ควบคุม Mode การทำงานและใช้เลือกวิธีการนับของ Counter ว่าจะให้นับแบบ Binary หรือ BCD (Binary Code Decimal) ก่อนที่จะใช้งานจะต้อง Program ข้อมูลให้กับ Register นี้เสียก่อน ซึ่งข้อมูลที่ Program ต่อไปนี้จะเป็นตัวกำหนดลักษณะการทำงานของ Counter Register นี้สามารถเขียนข้อมูลเข้าไปได้อย่างเดียวไม่สามารถอ่านออกมาได้ และจะติดต่อกับ Register นี้ได้เมื่อขา A0 ,A1 มี Logic "1"

Counter #0, #1, #2 :Counter ทั้ง 3 นี้มีลักษณะที่เหมือนกันและทำงานอย่างเป็นอิสระต่อกันและกัน แต่ละ Counter นี้มีขนาด 16 บิท Pre-Settable Down Counter และสามารถนับได้เป็น Binary หรือ BCD ก็ได้ข้อมูลที่อยู่ภายใน Counter เหล่านี้สามารถถูกอ่านโดย Microprocesser ได้โดยไม่ทำให้ข้อมูลภายใน Counter นั้นเสียหาย ซึ่งระบบสามารถจะแสดงค่าใน Counter ได้ตลอดเวลา โดยไม่กระทบกระเทือนการทำงานทั้งหมดของ Counter

3) การ Program 8253 (Control Word Format)

Mode การทำงานของ Counter ทั้งหมด สามารถเลือกได้โดยการเขียน ข้อมูลเข้าไปใน Register ควบคุมซึ่งมีรูปแบบของคำสั่งควบคุม (Control Word Format)ดังรูปที่ 2.19 ทำให้ Address ของคำสั่งควบคุมนี้เป็น Address ที่มีค่าของ A0 ,A1 เป็น Logic "1" ซึ่งในระบบที่เราพิจารณาอยู่นี้ให้ Address ของคำสั่งควบคุมเป็น 303H

เนื่องจาก Counter ของ 8253 มีจำนวน 3 ตัวด้วยกันฉะนั้นการ Program Counter นั้น จำเป็นต้องกำหนด Counter ที่ต้องการจะ Program เสียก่อนการกำหนดทำได้โดยให้ Logic ที่ถูกต้องกับ Bit D7 ,D6 ซึ่งมีชื่อว่า SC1 ,SC2 ตามลำดับ ของรูปที่ 2.19 เมื่อได้ Counter ที่ต้องการแล้ว Counter นั้นก็จะถูก Set และจะอยู่ในสภาพนั้น จนกว่าจะมีคำสั่งควบคุมอื่นๆ มาทำให้เปลี่ยนแปลง ส่วนการกำหนดค่า Logic ของ Bit D7 ,D6 สำหรับในการเลือก Counter มีดังนี้

Control Byte D7-D0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCP

รูปที่ 2.19 แสดงข้อกำหนดของแต่ละบิทของ Register ควบคุม

<u>หมายเหตุ</u> ใน 1 ไบท์ของ Counter จะประกอบด้วย 16 บิท โดยบิท D0-D7 เป็น Least-Signification Byte และ D8-D15 เป็น Most-significant Byte

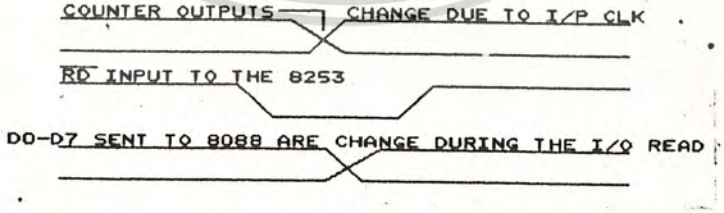
D7	D6	Counter Select
0	0	0
0	1	1
1	0	2
1	1	ไม่มีความหมาย

เมื่อเลือก Counter จากการใช้ Bit D7, D6 ได้แล้วต่อไป Bit D5,D4 จะเป็นตัวกำหนดว่า Counter นี้ (หรือ Register) จะใช้ในการ Read/Load Mode ซึ่ง MOde การอ่าน (Read Mode) เป็น Mode ที่ Microprocesser อ่านข้อมูลจาก Counter ส่วน Mode การ LOad (Load Mode) เป็น Mode ที่ Microprocesser เขียนข้อมูลเข้าไปใน Counter Bit D5 และ D4 ถูกกำหนดดังนี้

D5	D4	R/L Difination
0	0	ค่าใน Counter ถูก Latch หมายความว่าค่าที่มีอยู่ใน Counter ที่ถูกกำหนดนี้ จะนำเข้าไปเก็บไว้ใน F/F ซึ่ง CPU สามารถ อ่านออกไปได้
0	1	Read/Load เฉพาะไบท์ที่มีนัยสำคัญต่ำ (Least-Significant Byte)
1	0	Read/Load เฉพาะไบท์ที่มีนัยสำคัญสูง (Most-significant Byte)
1	1	Read/Load ไบท์ที่มีนัยสำคัญต่ำก่อนเสร็จแล้วตามด้วยไบท์ที่มีนัยสำคัญสูง

เมื่อ D5, D4 มีค่าเป็น 00H Counter จะถูกทำให้อยู่ในโหมดการ Latch ซึ่งเป็นโหมดที่ใช้สำหรับการอ่านค่าของ Counter ขณะที่ Counter ยังทำงานอยู่การเขียนโหมดนี้ให้กับ Register ควบคุม จะทำให้ค่าที่อยู่ใน Counter ถูก Latch ให้กับ Register ภายใน และเมื่อทำการอ่าน Counter ค่านี้จะถูกอ่านออกไป

ถ้าไม่อยู่ในโหมดการ Latch แล้วการอ่านข้อมูลจะเกิดข้อผิดพลาดขึ้นได้เพราะขณะที่ทำการอ่านข้อมูลนั้น ขบวนการที่เกิดขึ้นใน Counter จะทำให้ข้อมูลที่อยู่เดิมเปลี่ยนไป(ดังแสดงใน Timing Diagram ดังรูปที่ 2.20)เป็นผลทำให้ข้อมูลที่ป้อนเข้า CPU เกิดผิดพลาดขึ้นฉะนั้นเพื่อที่จะอ่านค่าของ Counter ให้ถูกต้องในขณะที่ Counter กำลังอยู่ในขบวนการนับอยู่นั้น สามารถทำได้โดยกำหนดคำสั่งควบคุมการ Latchก่อนแล้วจึงให้คำสั่งควบคุมอื่นที่เป็นคำสั่งการอ่านในไบท์ต่อไป Counter ของ 8253 ซึ่งสามารถแก้ไขได้โดยการที่ Micro-processer Latch ข้อมูลจาก เอ้าท์พุทของ Counter ก่อนที่จะทำการอ่าน



รูปที่ 2.20 เป็น Timing Diagram ที่แสดงการผิดพลาดระหว่างการอ่านข้อมูลจาก เคาน์เตอร์ของ 8253

ยังมีอีก 4บิทที่เหลือของคำสั่งควบคุมในรูปที่ 2.19 คือ D3, D2, D1, D0 แต่จะกล่าวถึง 3 บิทแรกก่อนคือ D3, D2, D1 บิทเหล่านี้เป็นบิทที่กำหนดโหมดการทำงานพื้นฐานของ Counter ซึ่งต่อไปจะได้อธิบายและแสดงตัวอย่างการใช้ Counter ในแต่ละโหมดทั้ง 5โหมด ในที่นี้เรามาดูลอจิกที่ให้กับ

D3, D2, D1 ในแต่ละโหมดดังนี้

D3	D2	D1	Mode Value
0	0	0	Mode 0 : Interrupt On Terminal Count
0	0	1	Mode 1 : Programable One-Shot
X	1	0	Mode 2 : Rate Generter
X	1	1	MOde 3 : Square Wave Generter
1	0	0	Mode 4 : Softwere Triggered Strobe
1	0	1	Mode 5 : Hardware Triggered Strobe

บิทสุดท้ายของคำสั่งควบคุมคือ D0 ใช้กำหนดลักษณะการนับของ Counter ว่ามกษณะการนับเป็นอย่างไร นั่นคือจะนับเป็น BCD หรือ Binary ถ้า D0 มีลอจิกเป็น "1" Counter จะนับแบบ BCD ถ้า D0 มีลอจิก เป็น "0" จะนับเป็นแบบ Binary ค่าที่มากที่สุดสำหรับการนับในโหมดการนับแบบ Binary มีค่าเท่ากับ 2^{16} และในโหมดการนับแบบ BCD เป็น 10^4

4) การทำงานในโหมดต่างๆของ 8253

<u>การทำงานใน Mode 0 :Interupt on Terminal control</u>

ลักษณะการทำงานในโหมด 0 ของ 8253 Counter จะนับแบบนับลงเมื่อค่าที่ Counter นับมีค่าเป็น 0 ขา Out ของ Counter จะมีลอจิกเป็น "1" ฉะนั้นในโหมด 0 นี้ Counter จะต้องถูกโปรแกรมค่าเริ่มแรกที่จะนับเสียก่อน จากนั้นก็ทำการนับลงด้วยอัตราเท่ากับความถี่ของ CLK ที่ป้อนเข้ามา เมื่อค่าที่นับมีค่าเป็น 0000H แล้วขา OUT ก็จะมีลอจิกเป็น "1" ซึ่งการใช้งานของขา OUT นี้สามารถนำไปใช้กับการอินเทอร์รัพท์ Microprocesser ได้ ขา OUT จะมีลอจิกเป็น "1" จนกว่าเคาน์เตอร์จะถูกป้อนค่าที่ต้องการนับเข้าไปอีก (อาจเป็นค่าเก่าหรือค่าใหม่ก็ได้) หรือเมื่อมีการเขียนคำสั่งเลือกโหมด เข้าไปให้กับ 8253

เมื่อ Counter เริ่มทำการนับ (นับลง) แล้วเราสามารถหยุดการนับของ Counter ได้ โดยการให้ลอจิก "0" ที่ขาอินพุท GATE ของ Counter ตัวอย่างในการเซ็ท Counter ใน Mode 0 โดยเราจะต้องทำการโปรแกรมให้กับคำสั่งควบคุมโดยให้บิทต่างๆ เป็นไปตามข้อกำหนดที่ได้แสดงดังรูป 5 สำหรับตัวอย่างนี้มีคำสั่งควบคุมเป็น 0011001B ซึ่งแต่ละบิทมีความหมายดังนี้

Bit D7 และ D6 = 00 เป็นการกำหนดให้ใช้ Counter หมายเลข 0

Bit D5 และ D4 = 11 เป็นการเซ็ทให้ Counter ทำการโหลด

	LSB เข้าไปก่อนแล้วตามด้วย MSB
Control Bit D3,D2	เป็นการกำหนดให้ Counter ใช้งานใน
และ D1 = 000	MOde 0
D0 = 1	เป็นการกำหนดให้การนับของ Counter
	เป็นแบบ BCD

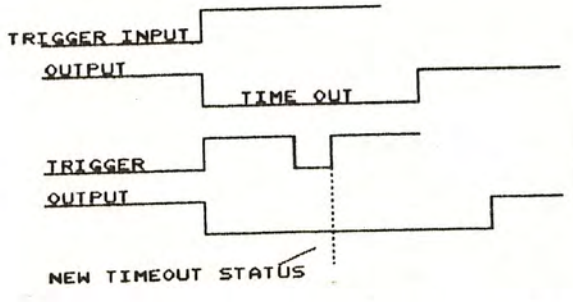
ฉะนั้นคำสั่งควบคุมซึ่งเท่ากับ 0011001B จะถูกเขียนเข้าไปที่พอร์ต 303H ซึ่งเป็นแอดเดรสของ Control Word Register

เมื่อทำการเซ็ทเคาน์เตอร์แล้ว เราจะต้องเขียนค่าเริ่มแรกของการนับให้กับรีจิสเตอร์ของเคาน์เตอร์ ให้มีลักษณะของการนับแบบ BCD โดยให้ LSB มีค่าเป็น 37 (เป็นเลขฐาน 10) และให้ MSB เป็น 01 ทั้งสองไบท์นี้จะถูกเขียนไปยังพอร์ทที่ 300H

ทันทีที่ไบท์ที่สอง (MSB) ได้ป้อนให้กับรีจิสเตอร์ของ Counters หมายเลข 0 การนับก็จะเริ่มขึ้นและเมื่อนับถึง 125 (ก็คือรีจิสเตอร์ของเคาน์เตอร์มีค่าเป็น 0) ขา OUT ของเคาน์เตอร์ก็จะมีลอจิกเป็น "1" ซึ่งเราอาจจะนำเอาขานี้ไปใช้ในการควบคุมอุปกรณ์ต่างๆ ได้

การทำงานใน Mode 1 :Programmable One-Shot

ในโหมด 1 นี้ 8253 จะทำงานในลักษณะของ One-Shot คือสามารถให้เอาท์พุทในรูปของพัลส์ และความกว้างของพัลส์นี้ มีค่าเป็นจำนวนเท่าที่เป็นเลขจำนวนเต็มของพัลส์ของสัญญาณนาฬิกาที่ป้อนเข้าที่ขา CLOCK การทำงานของ One-Shot จะเกิดขึ้นเมื่อมีสัญญาณขอบขาขึ้น (Rising Edge) เข้ามากระตุ้น (Trigger) ที่ขาอินพุท GATE ทำให้เกิดพัลส์ที่เอาท์พุท แต่ถ้ามีสัญญาณกระตุ้น เข้ามาที่ขา GATE ในขณะที่เอาท์พุทยังทำงานไม่เสร็จสิ้น สัญญาณที่มากระตุ้นใหม่นี้ก็จะทำให้เกิดพัลส์ลูกใหม่ดังแสดงในรูปที่ 2.21



รูปที่ 2.21 แสดงไดอะแกรมเวลาของ 8253 เมื่อโปรแกรมให้ทำงานในแบบ One-Shot

การทำงานใน Mode 2 : Rate Generator

ในโหมด 2 นี้ 8253 ถูกใช้เป็นเคาน์เตอร์ที่ทำหน้าที่หารด้วย "N" ทำให้ได้สัญญาณที่ขาเอาท์พุท เป็นลอจิก "0" และ "1" สลับกันไปด้วยอัตราความถี่เท่ากับสัญญาณอินพุทคล็อกหารด้วย "N" ซึ่งทำให้มีช่วงกว้างของลอจิก "1" มีค่าเท่ากับคาบของสัญญาณอินพุทคล็อกคูณอ้วยจำนวนค่า "N" ดังรูปที่ 2.22 ซึ่งค่า "N" นี้เป็นค่าที่ได้จากการโหลดให้กับเคาน์เตอร์ในโหมด 2 แต่ถ้าขณะที่เอาท์พุทของเคาน์เตอร์ยังทำงานอยู่ในช่วงของลอจิก "1" เกิดมีค่า "N" ค่าใหม่โหลดเข้ามาเอ๊าท์พุทของเคาน์เตอร์ก็ยังคงทำงานในลักษณะเดิมจนกระทั่งหมดคาบ แล้วจึงจะทำตามค่า "N" ที่โหลดเข้ามาใหม่

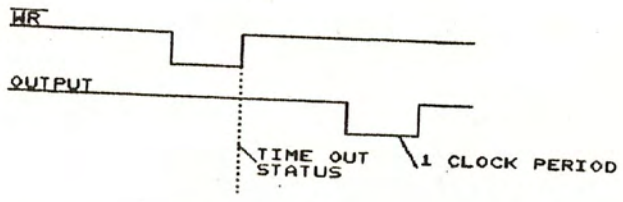
รูปที่ 2.22 แสดงไดอะแกรมเวลาที่เอ๊าท์พุทของ 8253 ในโหมด 2

การทำงานใน Mode 3: Square Wave Generator

การทำงานในโหมด 3 นี้มีลักษณะคล้ายกับโหมด 2 ที่กล่าวมาแล้ว เว้นแต่เอ๊าท์พุทที่ได้เป็น Square Wave ที่มีช่วงกว้างของลอจิก "1" และ "0" สมมาตรกัน (Duty Cycle = 50 %) แต่ถ้าค่า "N" เป็นเลขคี่ เอาท์พุทที่ได้จะมีช่วงกว้างของลอจิก "1" เท่ากับ (N+1)/2 * คาบเวลาของสัญญาณ CLK และช่วงกว้างของลอจิก "0" เท่ากับ (N-1)/2 * คาบเวลาของสัญญาณ CLK

การทำงานใน Mode 4 : Software Triggered Stobe

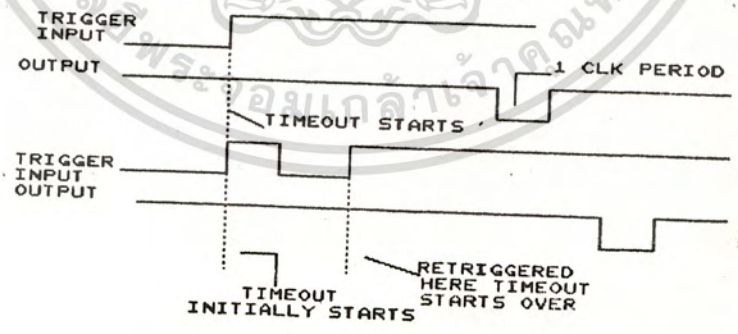
ในโหมดนี้ ผู้เขียนโปรแกรมสามารถเซ็ทเคาน์เตอร์ให้มีเอาท์พุทหน่วงเวลาออกไปหลังจากที่เคาน์เตอร์เริ่มทำงาน (โดยเริ่มนับค่าที่โหลดให้) เอาท์พุทที่ได้จะมีลักษณะเป็นลอจิก "0" มีคาบเวลาเท่ากับคาบเวลาของ CLK 1 ลูก และจะกลับไปเป็นลอจิก "1" อีก (เอ๊าท์พุทเป็นลอจิก "1" ทันทีเมื่อเราเซ็ทโหมด 4 ให้) เอ๊าท์พุทนี้จะเกิดเมื่อเคาน์เตอร์นับค่าที่เราโหลดให้ลดลง (นิยมนับลง) จนค่าที่ปรากฏที่เคาน์เตอร์เท่ากับ 0 ดังรูปที่ 2.23



รูปที่ 2. 23 แสดงไดอะแกรมเวลาที่เอาท์พุทของ 8253 ในโหมด 4

การทำงานใน Mode 5 : Hardwae Triggered Strobe

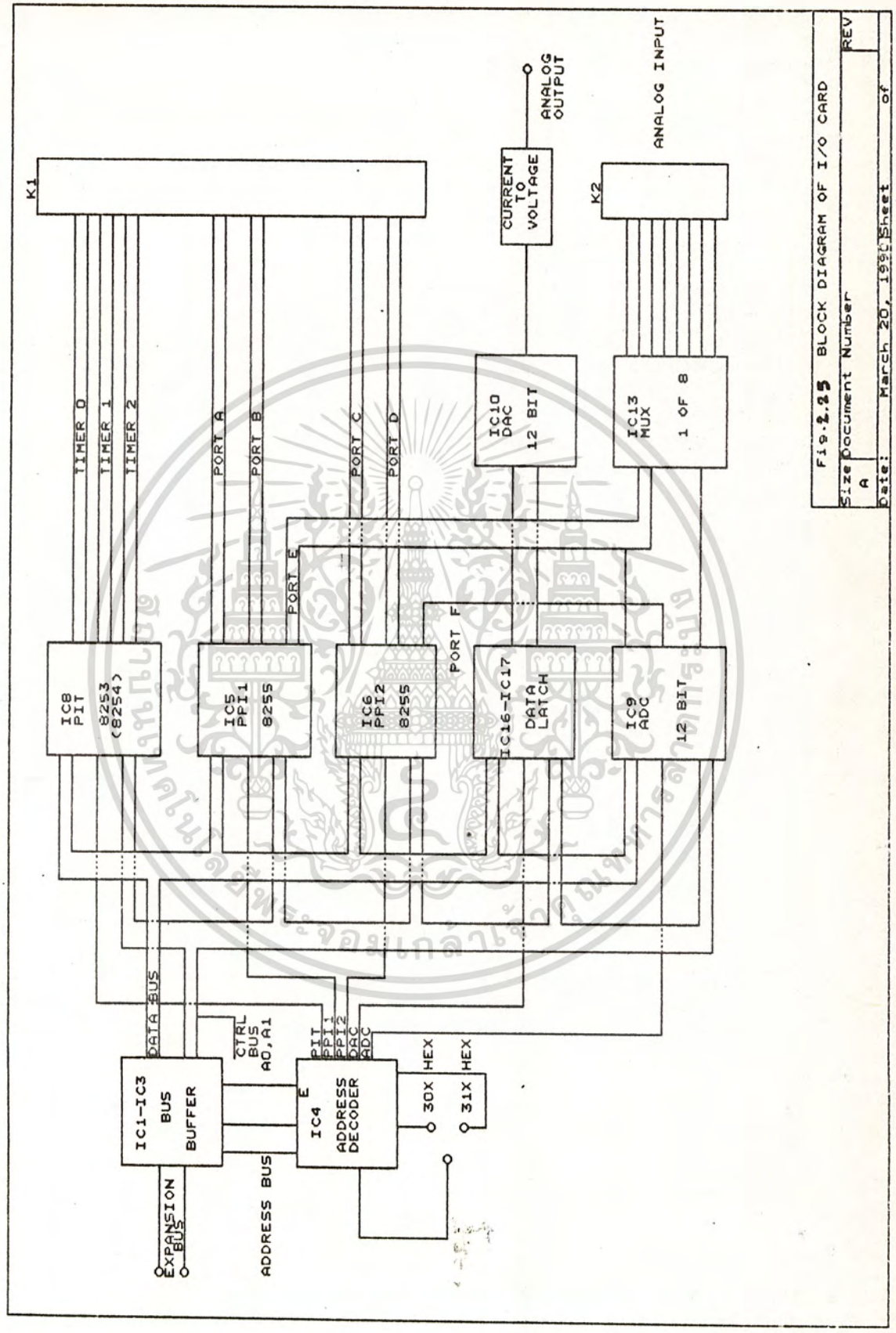
การทำงานในโหมด 5 นี้ เคาน์เตอร์จะให้เอ๊าท์พุทหน่วงเวลาออกไปหลังจากที่เคาน์เตอร์เริ่มนับเหมือนกับในโหมด 4 แต่เคาน์เตอร์จะเริ่มทำการนับก็ต่อเมื่อมีขอบขาขึ้น (Rising Edge) ของสัญญาณกระตุ้น (Trigger) เข้ามาให้กับเคาน์เตอร์ที่ขาอินพุท Gate เมื่อเคาน์เตอร์นับค่าเป็น 0 ก็จะให้เอ๊าท์พุทเป็นลอจิก "0" มีคาบเท่ากับคาบของ CLK 1 ลูกแล้วเอ๊าท์พุทก็กลับเป็นลอจิก "1" อีก แต่ถ้าในขณะที่เคาน์เตอร์กำลังนับอยู่นี้ เกิดมีสัญญาณกระตุ้นลูกใหม่เข้ามา เคาน์เตอร์จะถูกทำให้กลับไปเริ่มต้นนับใหม่ ดังแสดงในรูปที่ 2.24



รูปที่ 2.24 ไดอะแกรมเวลาแสดงการทำงานของ 8253 ในโหมด 5

2.2 หลักการของระบบ

จะพบว่าความต้องการของเรานั้น คือการที่จะสามารถทำการควบคุมอุปกรณ์ I/O ได้ 32 CH และมี Counter/Timer ที่สามารถ Program ได้ซึ่งอาจจะใช้ในการตั้งเวลาหรือเป็นฐานเวลา และมีทั้งตัวแปลงสัญญาณ Digital เป็น Analog และ Analog เป็น Digital ซึ่งความต้องการต่างๆเหล่านี้ก็เพื่อที่จะนำไปประยุกต์ใช้งาน ในด้านต่างๆอาทิเช่นใช้ในงานอุตสาหกรรมซึ่งสามารถที่จะนำเอา Computer PC ติดต่อกับอุปกรณ์ภายนอกได้โดยการใช้ Board นี้ควบคุม และอาจจะให้ Board นี้ทำหน้าที่เป็นตัวสร้างความถี่ โดยสามารถที่จะเลือกย่านในการ Sweep ได้ ซึ่งมีประโยชน์มากในการทดลองงานอิเล็คทรอนิคส์ หลักการของระบบใน Board นี้ ได้แสดงดัง block diagram รูปที่ 2.25



การทำงานของระบบ

2.2.1. ส่วน decoder ภายในส่วนนี้จะมี Buffer ต่อกั้นไว้ก่อน เพื่อที่จะไม่ให้การ์ดนี้ไป load ต่อวงจรของ PC ซึ่งในส่วนของการ Decode Address เพื่อที่จะให้สามารถทำการติดต่อกับอุปกรณ์ต่างๆได้ในที่นี้เราจะใช้ IC.PAL เป็นตัวทำหน้าที่ในส่วนนี้ ซึ่งจะได้สัญญาณ O/P ต่างๆออกไปควบคุมอุปกรณ์ต่างๆ เช่น 8255*2, 8253, DAC และสัญญาณอื่นๆตามความต้องการ โดยจะต้องทำการเขียนสมการ Boolean แล้วทำการ MAP ลงบน PAL MAP แล้วทำการ Burn

2.2.2 ส่วนของ IC.8255 จะทำหน้าที่ควบคุมอุปกรณ์ภายนอกที่สามารถ Program ได้โดยการควบคุมให้เป็น input หรือoutput ได้ 32 CH ในการใช้งานจะอาศัย Port A ถึง Port C โดยการProgram ที่ Register ควบคุม ในการกำหนดให้เป็น Input หรือ Output ซึ่งในการใช้งานสามารถที่จะควบคุมการปิด-เปิดอุปกรณ์ไฟฟ้าต่างๆได้ หรือทำการควบคุม Steping Motor ได้ขึ้นอยู่กับ โปรแกรม สำหรับใน Project นี้จะใช้ 8255 จำนวน 2 ตัว ซึ่งจะพบว่ามีทั้งหมด 6 Port 48 CH แต่เราจะใช้เพียง 32 CH เนื่องจากจะนำเอา Port อีก 2 Port มาใช้เป็นตัวส่งสัญญาณควบคุมต่างๆ

2.2.3 ส่วนของ 8253 จะทำหน้าที่เป็นทั้ง Timer/Counter ที่สามารถ Program ได้ซึ่งเราอาจจะใช้เป็นตัวตั้งเวลาหรือเป็นตัวกำเนิดสัญญาณ Square โดยการ Program และสัญญาณ CLK ที่ป้อนให้กับ IC ตัวนี้สามารถเลือกได้โดยการใช้ Dip Switch มีให้เลือกโดยจะทำการหารอยู่ 4 ค่า คือ หาร 2 , 4 , 8 , 16 ของสัญญาณ Clock ของระบบ

2.2.4 ส่วนการแปลงสัญญาณ Digital เป็น Analog ในที่นี้จะใช้ IC.PM7541 ซึ่งเป็น 12 Bit DAC ที่เป็นแบบ Linear โดยจะรับข้อมูลที่ส่งมาจาก Data Bus บน Slot IBM PC ซึ่งจะต้องมีภาค Latch ข้อมูลก่อนส่งเข้า PM7541 เพื่อที่จะสามารถให้เราหยุดข้อมูลเป็นช่วงๆได้ซึ่งในส่วนการใช้งานเพื่อทำหน้าที่เป็น Sweep Gen. หรือทำหน้าที่เป็นตัวกำเนิดแรงดันคงที่ถ้าเราไม่มีการ Latch ข้อมูลไว้การจะเรียกดูที่ความถี่ไหนก็คงจะทำได้ยาก สำหรับการ Latch นี้จะใช้ IC.74LS373 เป็นตัวทำหน้าที่ในส่วนนี้ ซึ่งจะต้องมีการทำการเลือกค่าตัวเก็บประจุให้กับ XR-2206 จึงต้องเอาเอาท์พุทพอร์ทของ 8255 มาควบคุมอีก 3 บิท

2.2.5 ส่วนการแปลงสัญญาณ Analog เป็น Digital จะใช้

ICL.7109 ซึ่งเป็น ADC ขนาด 12 Bit แต่เนื่องจากมีสายสัญญาณ Input เพียง CH เดียว แต่เราต้องการ 8 CH ดังนั้นจึงต้องใช้ IC.4051 เป็น CMOS ทำหน้าที่เป็น Multiplexer ซึ่งจะมี Input 8 CH Output 1 CH โดยสามารถทำการเลือก CH ได้โดยส่งสัญญาณควบคุมออกมาทาง Port ของ 8255

บทที่3

การออกแบบวงจร

ในส่วนนี้จะเป็นวิธีการออกแบบวงจรที่ใช้งานในส่วนต่างๆของ Board ซึ่งจะแบ่งออกได้เป็นส่วนต่างๆ ดังนี้

3.1 การออกแบบส่วนควบคุมการกำเนิดความถี่ (Sweep Generater)

3.2 การออกแบบส่วนควบคุมการติดต่อกับ IBM

3.3 การออกแบบส่วนควบคุมการติดต่อกับอุปกรณ์ภายนอก

3.4 การออกแบบส่วนควบคุมการแปลงสัญญาณ Digital เป็น Analog

3.5 การออกแบบส่วนควบคุมอุปกรณ์ภายนอก

3.6 การออกแบบส่วน Power Supply

3.1 ส่วนควบคุมการสร้างความถี่

ในการสร้างความถี่ที่ใช้ในชิ้นงานนี้ได้ใช้ไอซี XR-2206 ซึ่งเป็นไอซีสำเร็จรูปที่ใช้ทำฟังก์ชั่นเจนเนอเรเตอร์ซึ่งสามารถจะสร้างสัญญาณต่างๆ ได้คือ Sine Wave , Square Wave , Trainangle Wave นอกจากนี้ยังนำไปประยุกต์ใช้เป็น FM, AM, FSK (Frequency Shift Keying) สร้างสัญญาณ Pulse และ Ramp ฯลฯ ซึ่งสำหรับ XR-2206 สามารถที่จะสร้างความถี่ได้ตั้งแต่ 0.01 Hz ถึง 1 MHz ความเพี้ยนต่ำและมีค่าความถี่ที่แปรเปลี่ยนกับความต่างศักย์ควบคุมภายนอกเป็นเชิงเส้น (Linear Sweep) ถึง 2000:1 ของย่านความถี่ ในไอซี XR-2206 จะประกอบด้วย Function Plot 4 Function คือ VCO (Voltage Control Oscillator) วงจรควบคุมแอนะลอกวงจรปรับแต่งสัญญาณ Sine (Analog Multipler + Sine Shaper) , วงจร Buffer (Unity Gain Buffer Amplifier) และ ชุดตัดต่อกระแส (Set of Current Switches) ซึ่งมีหลักการดังนี้ VCO จะสร้างความถี่ต่างๆ ตามกระแสอินพุท ซึ่งได้จากความต้านทานที่ใช้ Bias กระแสก็จะได้ความถี่ตามสมการ t = 1/RC ชุดตัดต่อกระแสไว้สำหรับทำ FSK สัญญาณ Pluse และสัญญาณ Ramp วงจรคูณแอนะลอก และวงจรปรับแต่งสัญญาณ Sine เนื่องจาก VCO จะให้สัญญาณรูปสี่เหลี่ยมออกมาจึงต้องมีวงจรปรับแต่งสัญญาณเพื่อจะได้สัญญาณรูป Sine มีความคม ความเพี้ยนต่ำและมีฮาร์โมนิคน้อย โดยมีส่วนปรับแต่ง 2 ชุด คือการปรับรูปสัญญาณ (Wave Form Adjust) และการปรับความสมมาตร (Symmetry Adjust) ซึ่งสามารถทำได้ถึง 0.5% ส่วนวงจรคูณจะเป็นวงจรเพื่อขยายสัญญาณ และในส่วนสุดท้ายคือวงจรบัฟเฟอร์เพื่อใช้ขับกระแสของสัญญาณที่ได้

3.1.1 หลักการพื้นฐานการใช้งาน XR-2206

ไอซี XR-2206 สามารถจะให้ค่าความถี่ต่างๆ ได้โดยการเลือกใช้ค่าตัวต้านทาน (Timing Resistor) และค่าตัวเก็บประจุ (Timing Capacitor) เป็นไปตามสมการ

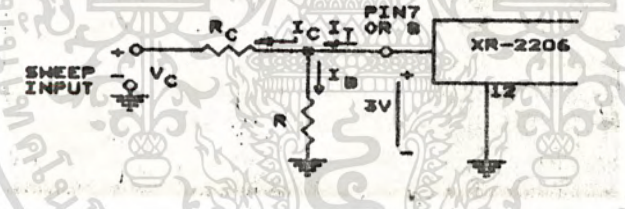
$$t = 1/RC \quad \text{Sec} \qquad \ldots.(3.1)$$

หมายเหตุ ค่าตัวต้านทานจะใช้ในช่วง 4 K จนกระทั่งถึง 200 K ค่าตัวเก็บประจุจะใช้ในช่วง 1000 PF จนกระทั่งถึง 100 PF สำหรับการสร้างความถี่เปลี่ยนแปลงเฉพาะช่วง (Frequency Sweep)

จากหลักการของความถี่ที่เกิดจากค่ากระแส Bias ที่ได้จากกระแสเวลารวม (Total Timing Current; I_t) เราจึงนำค่า I_t ,มาใช้ประโยชน์ในการควบคุมความถี่ โดยการ Bias กระแสที่ Timing Terminal ขา 7 หรือขา 8 โดยการเลือกที่ขา 9 ถ้าขา 9 เป็นบวกจะเลือกใช้ขา 7 และเป็นลบหรือกราวด์ จะเลือกใช้ขา 8) ซึ่งจะได้ผลเป็นไปตามสูตร

$$f = 320 * I_t(ma)/C(uF) \quad Hz \qquad ....(3.2)$$

เนื่องจากขา ไทม์มิ่งเทอร์มินอลมีค่าจุด Impadance ต่ำ (Low Impedance Point) และมีค่า แรงดันต่ำ และ Bias ภายในเป็น 3 โวลท์ จึงทำให้จะต้องมีขอบเขตของกระแสตั้งแต่ 1uA ถึง 3mA ดังนั้น เราจึงสามารถควบคุมความถี่ด้วยความต่างศักย์ V_c ดังรูปที่ 3.1 โดยการต่อตัวต้านทานที่ R Timing Terminal (Pin 7 or 8) ความถี่ที่ใช้ในการ Oscillate จะมีความสัมพันธ์กับ V_c เป็น



รูปที่ 3.1 แสดงการต่อวงจรเพื่อที่จะทำให้ XR-2206 ทำหน้าที่เป็น Sweep

$$f = 1/RC + R*(1-V_c/3)/RC \quad Hz \qquad ....(3.3)$$

เมื่อ V_c มีหน่วยเป็น โวลท์ ส่วน Voltage-To-Frequency Gain K จะมีค่าเป็น

$$K = f/\ V_c = -0.32/(R_cC) \quad Hz/V \qquad ....(3.4)$$

นั่นคือถ้าเราต้องการที่จะให้แรงดันควบคุม ให้เกิดการ Sweep นั้นเราจะต้องให้ $0 < V_c < 3$ V

จากสมการ 3.3 จะได้ความถี่ต่ำสุดของการ Sweep เมื่อ V_c=3V จะได้

$$f_L = 1/RC \quad \text{....(3.5)}$$

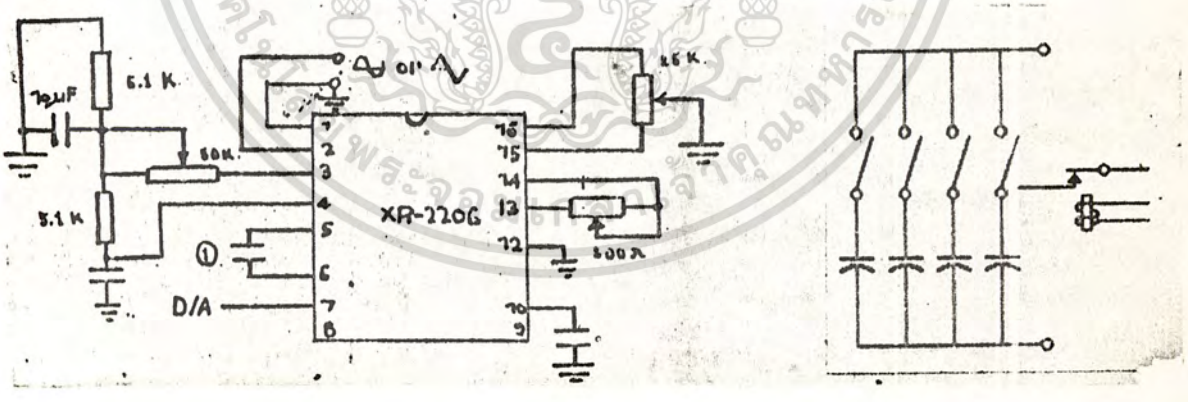
และจะได้ความถี่สูงสุดของการ Sweep เมื่อ V_c = 0 V จะได้

$$f_H = 1/RC + R/R_c \quad \text{....(3.6)}$$

สำหรับการควบคุม Amplitude ของสัญญาณ จะใช้ตัวต้านทานที่ปรับค่าได้ต่อเข้าที่ขา 3 กับไฟลบ จะทำให้สัญญาณมี Amplitude เปลี่ยนไป สำหรับสัญญาณรูป Sine จะเป็นค่า

$$A = 60*R \text{ (K-ohm)} \quad mV \quad \text{....(3.7)}$$

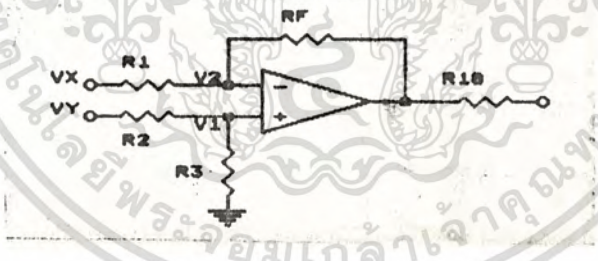
ซึ่งเราต้องการขนาดของ Amplitude 2-3 V_p เนื่องจาก XR-2206 สามารถควบคุมความถี่จากภายนอกด้วยการเลือกค่า C ที่คล่อมระหว่างขา 5 และขา 6 เดิมเราใช้ Analog Switch เบอร์ CD 4066 ทำหน้าที่ในการเลือกค่า C โดยแอนะลอกสวิทช์ตัวนี้ภายในมี 4 ตัวแต่ก็มีปัญหาว่ามันไม่สามารถที่จะทำการเลือก C ได้ เราจึงหันมาใช้ Reed Relay แทนจึงทำให้ทำงานได้ และสัญญาณที่จะมาควบคุมการ Select จะถูกส่งมาจาก 8255 จะเป็นตัวสร้างสัญญาณ MUXDAC0-3 ซึ่งจะทำการเลือกใช้ค่า C อยู่ 4 ค่า



รูปที่ 3.2 แสดงวงจร XR-2206 พื้นฐาน ที่ใช้เป็นวงจร Sweep

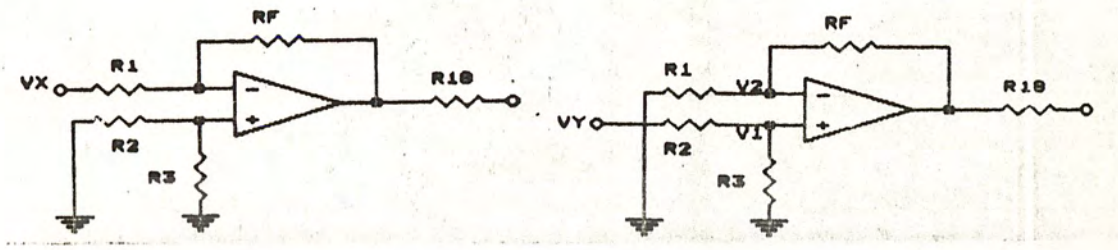
จุดประสงค์หลักของการนำเอาไอซี XR-2206 มาใช้งานก็เพื่อทำให้ตัวมันสามารถทำหน้าที่เป็น Sweep Gen ในที่นี้ใช้ขา 7 จึงปล่อยขา 9 ลอยไว้หรือต่อเข้าไฟบวกก็ได้ ซึ่งจะต้องมีการคำนวณหาค่าอุกรณ์และวงจรในการ

ทำให้เกิด ค่าแรงดันที่เหมาะสมมาควบคุมการ Sweep โดยสิ่งสำคัญในการที่จะทำให้ XR-2206 สามารถทำหน้าที่เป็น Sweep ให้ได้ค่าที่ถูกต้องและมีความไวพอเมื่อมีแรงดันควบคุมจากภาค DAC ที่ใช้ในการควบคุมการ Sweep โดยจะต้องทำการ Set ค่าความต้านทานในการ จำกัดกระแสที่ไหลออกจากขา 7 ของ XR-2206 ตัวต้านทานที่ใช้ควบคุมในที่นี้คือ R_{18} โดยจะต้องคำนวณหาค่าที่เหมาะสมและให้อยู่ในย่านที่คู่มือของ XR-2206 ได้บอกไว้ ในที่นี้ ไม่เกิน 3 mA (I_t) และ V_c = 3 Volts เราจึงต้องทำการออกแบบส่วนนี้ให้ดีๆ ในชิ้นงานนี้ก็ได้ทำการออกแบบโดยใช้วงจร Differential Amplifier ซึ่งสัญญาณอินพุท ทางขา Inverting ก็มาจากเอาท์พุทของ DAC และ อินพุททางขา Non-Inverting ก็เอามาจากแรงดันคงที่ที่ขา 10 ของ XR-2206 ซึ่งในภาวะปกติแล้วขา 10 นี้จะมีแรงดันคงที่ประมาณ 3 โวลท์ โดยมี C 10 uF ทำหน้าที่เป็น ดีคัปปลิ้ง เพื่อเป็นแรงดันอ้างอิงสำหรับนำมาปรับความกี่ และจะถูก Divider แรงดันลงครึ่งหนึ่งเพราะอยู่ที่ลักษณะของวงจร Diff-Amp ป้อนเข้าขา Non-Inverting ของ Op-Amp ดังรูปที่ 3.4



รูปที่ 3.4 แสดงวงจรควบคุมการ Sweep โดยอาศัยวงจร Diff-Amp

จากรูปที่ 3.4 แสดงวงจร Diff - Amp ใช้ Op-Amp เพียงตัวเดียว เราสามารถที่จะวิเคราะห์วงจรนี้ได้โดยการ Derive ค่า Voltage Gain ของวงจรดังต่อไปนี้ หา Voltage Gain ของวงจรที่แสดงในรูปที่ 3.4 เมื่อมี 2 อินพุทในที่นี้คือ V_x ,V_y เพราะฉะนั้นเราจะใช้ ทฤษฎี Supperposition ในการหาความสัมพันธ์ระหว่างแรงดันเอาท์พุท ต่อแรงดันอินพุทออกเป็นส่วนๆ โดยกรณีแรกคิดเมื่ออินพุท V_y =0 จะได้ดังรูป 3.5ก



รูปที่ 3.5ก แสดงวิธีคิดโดยอาศัยทฤษฎี Superposition
ก) เมื่ออินพุท V_y = 0 V ข) เมื่ออินพุท V_x = 0 V

จะพบว่าลักษณะของวงจรจะเป็น Inverting Amplifier ซึ่งเอาท์พุทที่ได้ก็เนื่องมาจากแรงดัน V_x เท่านั้น นั่นคือ

$$V_{ox} = -R_f/R_1 \ *(V_x) \qquad ....(3.8)$$

โดยที่ R_2 ,R_3 ทำหน้าที่เป็น Off Set
ในกรณีที่สอง เมื่อทำการหาอีกส่วนหนึ่งเราก็ให้ แรงดัน V_x = 0 V รูปก็จะกลายเป็น Non-Inverting Amplifier ดังรูปที่ 3.5ข ซึ่งแรงดันอินพุทจะถูก Divider ลงเนื่องจาก R_2 ,R_3 ที่ขา Non-inverting อินพุท (ขา3) เพราะฉะนั้นจะได้ว่า

$$V_1 = R_3*(V_y)/(R_2+R_3) \qquad ....(3.9)$$

และ เอาท์พุทเนื่องมาจาก V_y ก็จะเป็น

$$V_{oy} = (1+R_f/R_1)*V_1 \qquad ....(3.10)$$

แทนค่า V_1 ลงในสมการ 3.10 จะได้

$$V_{oy} = R_3*\{(R_1+R_f)/R_1\}*V_y(R2+R3) \qquad ....(3.11)$$

ในวงจร Diff-Amp เพื่อให้ง่ายต่อการคำนวณเรามักจะ Set ค่าให้ $R_1=R_2$ และ $R_f=R_3$ จะได้ว่า

$$V_{oy} = R_f * V_y / R1 \qquad ....(3.12)$$

ทำการรวมสมการ 3.8 และ 3.12 เข้าด้วยกันจะได้

$$V_o = V_{ox} + V_{oy}$$

$$V_o = -R_f/R1 * (V_x - V_y) \qquad ....(3.13)$$

หรือ Voltage Gain ของวงจร จะได้

$$A_D = V_o/(V_x - V_y) = -R_f/R_1 \qquad ....(3.14)$$

แต่ในการออกแบบของชิ้นงานนี้เราต้องการ Gain = 1 ดังนั้นเราจึง Set ค่าดังนี้ $R_1=R_f=R_2=R_3=1M$ ดังรูปที่ 3.4

ในการคำนวณค่า R_{1B} เพื่อที่จะให้ได้ค่าที่เหมาะสมโดยเอาเงื่อนไขของ XR-2206 ที่ว่าจะต้องให้แรงดัน <= 3 Volts และกระแส <= 3 mA เมื่อเราใช้สูตรของวงจร Diff-Amp จะได้

$$V_o = -(V_x - V_y) \qquad ....(3.15)$$

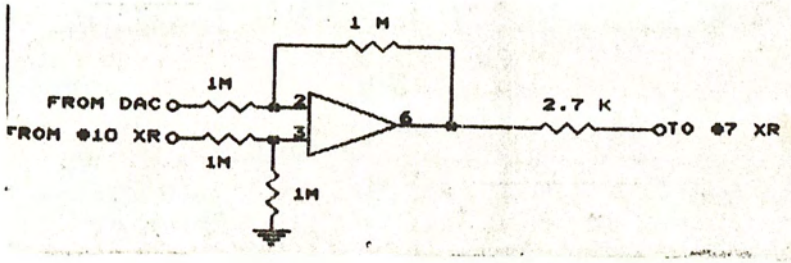
โดยที่แรงดัน V_x สูงที่สุดในการควบคุมในที่นี้มีค่าเท่ากับ 10 โวลท์ และ V_y มีค่าประมาณ -3 โวลท์ เมื่อถูก Divider ลงมาจะได้ V_y = -1.5V จะได้

$$V_{o1} = -(10-(-1.5)) = -11.5 \text{ V}$$

จาก Spect ของ XR-2206 กระแส I_t = 3 mA , V_c = 3V

$V_{R1B} = V_{o1} - v_c = 11.5-3 = 8.5$ V

$R_{1B} = V_{R1B}/I_t = 8.5V/3mA = 2.83$ K ในที่นี้เลือกใช้ 2.7K



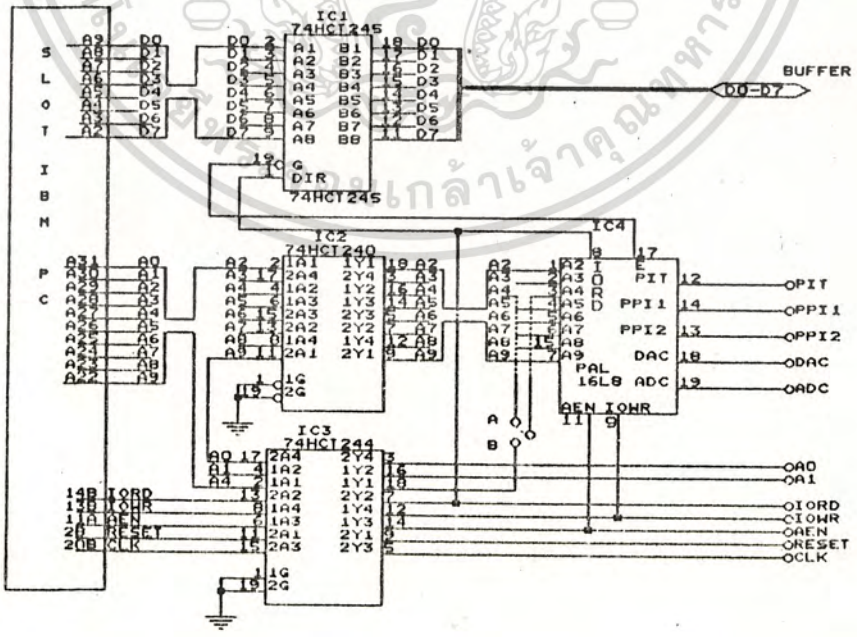
รูปที่ 3.6 แสดงวงจรสมบูรณ์ที่ใช้ควบคุมการ Sweep

3.2 ส่วนควบคุมการติดต่อกับ IBM PC

ในส่วนนี้ความหมายก็คือ การถอดรหัส Address ของ CPU เพื่อที่จะสามารถทำให้ชิ้นงานนี้ติดต่อกับ Computer ได้ ซึ่งอุปกรณ์ที่ทำหน้าที่นี้ก็คือ IC. PAL (Programmable Array Logic) ซึ่งจะใช้เบอร์ 16L8 ตามความหมายของเบอร์นี้คือจะมีจำนวน Input อยู่ 10 เส้น และ Output 8 เส้น และจำนวน Input ที่สามารถ Program ได้อีก 6 เส้น ภายในตัว IC. จะประกอบไปด้วย Gate AND,OR,INVERT โดยในการ Program มีลำดับขั้นดังต่อไปนี้

1) เขียนสมการ Boolean ที่ต้องการ
2) กำหนดตำแหน่งหน้าที่ของแต่ละขาลงใน Logic Diagram
3) ทำการ Mark จุดลงบน LogicDiagram ตามสมการ Boolean ที่เขียนไว้
4) ใช้เครื่อง Burn PAL ทำการบรรจุ Program ในการควบคุมลงไปไว้
5) ทดสอบโดยการป้อน Input ตามสมการ Boolean แล้ววัด Output ที่ได้ออกมาว่าตรงตามที่ต้องการหรือไม่

วงจรที่ใช้ควบคุมการติดต่อระหว่างตัวควบคุมกับ Computer แสดงดังรูปที่ 3.7



รูปที่ 3.7 วงจรที่ใช้ในการติดต่อกับ Computer

ในการเขียนสมการ Boolean นั้นจะต้องรู้ว่าสัญญาณ Output ที่ต้องการมีอะไรบ้างและ Output ต่างๆเหล่านั้นมี Input อะไรที่เกี่ยวข้อง ซึ่งใน Project นี้ต้องการสัญญาณที่จะควบคุมดังนี้คือ

1) สัญญาณ $\overline{CS}$ ของ 8255 ต้องการ Logic "0" จะสร้างสัญญาณเป็น $\overline{PPI1}$ และ $\overline{PPI2}$
2) สัญญาณ $\overline{CS}$ ของ 8253 ต้องการ Logic "0" จะสร้างสัญญาณเป็น $\overline{PIT}$
3) สัญญาณ $\overline{Write\ DAC}$ ต้องการ Logic "0" จะสร้างสัญญาณ DAC
4) สัญญาณ $\overline{CS}/\overline{LOAD}$ ของ ADC ต้องการ Logic "0" จะสร้างสัญญาณ ADC
5) สัญญาณ $\overline{E}$ ในการเปิด Gate Tri-State แบบ Bi-Directional

ซึ่งจะพบว่าสัญญาณที่ต้องการมีจำนวนมากถ้าเราใช้ Gate ธรรมดาก็คงจะสิ้นเปลืองและไม่สะดวก จึงใช้ IC.PAL มาทำหน้าที่นี้แทน โดยในการสร้างสมการ Boolean นั้นเราจะสร้างสมการของสัญญาณ E ก่อนเนื่องจากเป็นสัญญาณร่วมของสัญญาณอื่นๆ สัญญาณ E(enable)นี้จะใช้ในการเปิด Gate เพื่อทำการอ่านและเขียนบน Data Bus ในการถอดรหัส Address นั้นเราต้องดูว่า I/O Map จะว่างช่วงไหนซึ่งจะมีให้เลือกใช้ได้หลายช่วงแต่สำหรับ Project นี้จะใช้ Address ในช่วง 0300H-0377H โดยทำการ Decode Address ที่ 0300H-030FH และ ที่ 0310H-031FH ซึ่งในการเลือกทำได้โดยการ Jump สายบน Board สัญญาณ $\overline{E}$(enable)จะเกิดขึ้นจากการถอดรหัส Address ดังนี้คือ

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	0	0	0	X	X	X	X

ซึ่งในการสร้างสัญญาณ $\overline{E}$ เราจะนำเอา A4-A9 มาทำ โดยจะพบว่าในการที่จะทำการถอดรหัส Address นั้นจะต้องใช้สัญญาณ AEN มาร่วมด้วย เพราะเหตุว่าในกรณีที่ไม่ต้องการจะติดต่อกับ Board นี้สัญญาณ AEN ก็จะ Active ที่ Logic"1" ซึ่งเป็นขบวนการ DMA เพื่อที่จะป้องกันไม่ให้ทำการอ่านหรือเขียนข้อมูลออกมาที่ Data Bus ซึ่งจะทำให้เกิดความผิดพลาดขึ้นได้ ส่วนในการควบคุมของ Data Bus นั้นจะพบว่ามีทั้งการอ่านและเขียนข้อมูลฉนั้นในขบวนการนี้ถ้าต้องทำการเขียนข้อมูลลงใน Data Bus นั้นจะพบว่าสัญญาณของ $\overline{E}$ นี้ต้อง Active ตลอดไม่ว่าจะเป็นขบวนการอ่านหรือขบวนการเขียน ดังนั้นจะต้องนำเอาสมการของการอ่านมาทำการ OR กับสมการของการเขียนจะได้สมการคือ

$$E = A9.A8.A7.A6.A5.A4.AEN.IORD+A9.A8.A7.A6.A5.A4.AEN.IOWR$$

โดยสัญญาณนี้ปกติเราต้องการ Active ที่ Logic "0" ฉะนั้น เราจึงเขียนสมการของ $\overline{E}$ ใหม่ เพื่อให้เข้าใจได้โดยที่เอาสมการนี้ไป Mark ลงใน Fuse Map จะเป็นตำแหน่งที่ Fuse ยังคงอยู่ นอกนั้น Burn ทิ้งหมด

$$\overline{E} = \overline{A9.A8.A7.A6.A5.A4.AEN.IORD+A9.A8.A7.A6.A5.A4.AEN.IOWR}$$

เราจะเอาสัญญาณ $\overline{E}$ เป็นส่วนร่วมในการสร้างสัญญาณอื่นๆ ดังต่อไปนี้ในการสร้างสัญญาณ $\overline{CS}$ ของ 8255 สัญญาณ $\overline{Write}$, $\overline{Read}$ Adc และ DAC, สัญญาณ $\overline{CS}$ ของ 8253 ดังนั้น เราจะต้องทำการแบ่ง Address เพื่อที่จะใช้ในการ Decode ต่างๆ โดยในที่นี้ได้จัดขบวนการติดต่อ ดังตารางที่ 3.1

ซึ่งเมื่อดูตำแหน่ง Address ในการ Decode แล้วพบว่าในการที่จะทำการเลือกการใช้งานอุปกรณ์แต่ละตัวที่อยู่บน Board นั้นจะเปลี่ยนแปลงเฉพาะตำแหน่ง A3,A2 โดยเราต้องการตามตาราง 3.2

ตารางที่ 3.1 แสดงตำแหน่งการถอดรหัส Address

Address(HEX)	READ	WRITE
3X0	not allowed	not allowed
3X1	MS byte of ADC	MS byte of DAC
3X2	LS byte of ADC	LS byte of DAC
3X3	not allowed	not allowed
3X4	port A	port A
3X5	port E	port E
3X6	port B	port B
3X7	not allowed	control register IC5
3X8	port C	port C
3X9	port F	port F
3XA	port D	port D
3XB	not allowed	control register IC6
3XC	Timer 0	Timer 0
3XD	Timer 1	Timer 1
3XE	Timer 2	Timer 2
3XF	not allowed	control register IC8

หมายเหตุ X = 0 หรือ 1 ขึ้นอยู่กับการ set jump short บน Borad

ตาราง 3.2 ค่าของ A3,A2

A3	A2	O/P
0	0	ADC,DAC
0	1	PPI1
1	0	PPI2
1	1	PIT

จากตารางเราจะได้สมการ Boolean ของแต่ละสัญญาณเป็น

$$\overline{RDADC} = \overline{IORD.A2.A3.E}$$

$$\overline{WRDAC} = \overline{IOWR.A2.A3.E}$$

$$\overline{PPI1} = \overline{A2.A3.E}$$

$$\overline{PPI2} = \overline{A2.A3.E}$$

$$\overline{PIT} = \overline{A2.A3.E}$$

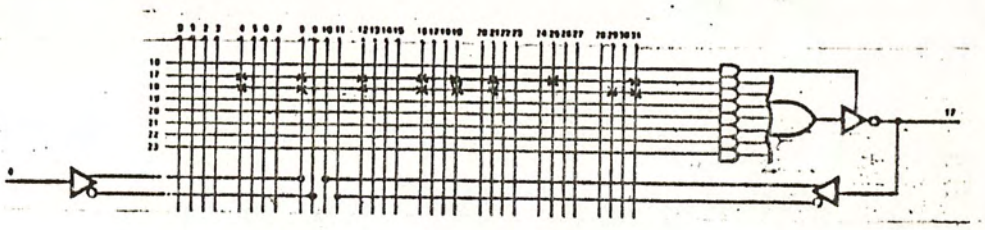
เมื่อเราได้สมการ Boolean มาเรียบร้อยขั้นต่อไปคือการกำหนดตำแหน่งลงบน Logic Diagram ของ PAL 16L8 ก่อนดังต่อไปนี้

ขา	ตำแหน่ง	ขา	ตำแหน่ง
1	A2	11	AEN
2	A3	12	PIT
3	A4	13	PPI1
4	A5	14	PPI2
5	A6	15	A8
6	A7	16	IOO
7	A9	17	E
8	IORD	18	WRDAC
9	IOWR	19	RDADC
10	GND	20	VCC

ขั้นต่อไปจะทำการ Mark ตำแหน่งลงใน Fuse Map โดยจะใช้คุณสมบัติของ IC.PAL ที่ได้กล่าวมาแล้วในหัวข้อทฤษฎี

โดยที่ IC.PAL 16L8 นี้ภายในถ้า Fuse ยังไม่ถูก Burn ณ.จุดนั้นจะมีสภาวะ Logic"0" ส่วนตำแหน่งใดที่ถูก Burn ที่จุดนั้นจะมีสภาวะ Logic"1" ซึ่งเมื่อนำเอาสมการ Boolean ที่ได้มา Mark ลงใน Fuse Map ดังนี้

ในกรณีของสมการ $\overline{E}$ โดยจะต้องมีหลักอยู่ว่าตามสมการที่ได้นั้นจะเป็นตำแหน่งที่ Fuse ไม่ถูก Burn(หรือคือจุด Mark) นอกเหนือจากนั้นจะ Burn ทั้งหมด(ใน Line เดียวกัน)จากการกำหนดตำแหน่งขาเราเลือกขา17เป็นตำแหน่งขา $\overline{E}$ วิธี Mark จะเป็นดังนี้ ณ.Row ที่ 17 และ colum ที่ 4จะเป็นตำแหน่งของ A4,colum ที่ 8 เป็นตำแหน่งของ A5,colum ที่ 12 เป็นตำแหน่งของขา A6,colum ที่ 16 เป็นตำแหน่งของ A7 ทำการไล่ไปทีละตัวก็จะได้ ดังรูป 3.8 จะพบว่าขาที่ไม่ได้ถูก Burn เลย เราจะแทนที่ด้วยเครื่องหมายกากบาทที่อยู่ใน AND Gate



รูปที่ 3.8 แสดงการ Mark ลงบน Fuse Map

และในที่นี้ ณ.ที่ขา 15 อยู่ตรง Output ของ Tri-State ซึ่งจะพบว่า IC. ตัวนี้จะมีขาที่ทำหน้าที่เป็นได้ทั้ง Input และ Output โดยเราต้องการขา 15 ทำหน้าที่เป็น Inputดังนั้นสัญญาณที่ไปเปิด Gate Tri-State จะต้องเป็น Logic"0" เพราะเหตุว่าจะไม่ให้ Input ของ Tri-State มากวนกันได้จึงต้องทำให้ Fuse ยังคงอยู่

ขั้นต่อไปเมื่อทำการ Mark จุดลงบน Logic Diagram(หรือ Fuse Map)แล้วก็นำไปเข้าเครื่อง Burn PAL ก็จะสามารถบรรจุสมการ Boolean แล้วทำการทดลองเป็นใช้ได้

3.3 ส่วนการติดต่อกับอุปกรณ์ภายนอก

ในส่วนนี้จะเป็นการออกแบบการติดต่อกับอุปกรณ์ภายนอกได้ 32 I/O และอีกทั้งต้องการสัญญาณที่ใช้ในการควบคุมภายในระบบอีกซึ่งเมื่อรวมกันแล้วจะต้องใช้ Gate จำนวนมาก เราจึงเลือกใช้ Chip IC.สำเร็จรูปทำหน้าที่ในส่วนนี้คือ IC.8255 ซึ่งภายในมี 3 Port สามารถ Program ให้เป็น Input หรือ Output ได้ สำหรับสัญญาณต่างๆที่เราต้องการมีดังนี้คือ

1) สามารถติดต่อกับอุปกรณ์ภายนอก ไม่ว่าจะเป็น Input หรือ Output ได้ทั้งหมด 32 CH

2) สัญญาณควบคุมการเลือก CH ของภาคแปลงสัญญาณ Analog เป็น Digital

3) สัญญาณควบคุมในการเริ่มแปลงสัญญาณ Analog เป็น Digital

4) สัญญาณที่ใช้ในการตรวจสอบ Status ในภาค ADC

ในส่วนแรกนั้นเราจะใช้เป็นตัวทำหน้าที่ในการติดต่อกับอุปกรณ์ภายนอกได้ 32 CHเราจึงใช้ทั้งหมด 4 Port โดยกำหนดเป็น Port A,B,C, D ซึ่งสามารถที่จะควบคุมหรือ Program ให้เป็นได้ทั้ง Input และ Output ด้วย SoftWare

ส่วนที่2 นั้นจะเป็นสัญญาณในการควบคุมการเลือก CH ในที่นี้สามารถที่จะรับ Input เข้ามาได้ 8 CH ซึ่งจะสร้างเป็นสัญญาณ MUXADC 0-2 โดยเราสามารถที่จะทำการผ่านสัญญาณในแต่ละ CH ได้ด้วย Program ที่ใช้ควบคุม

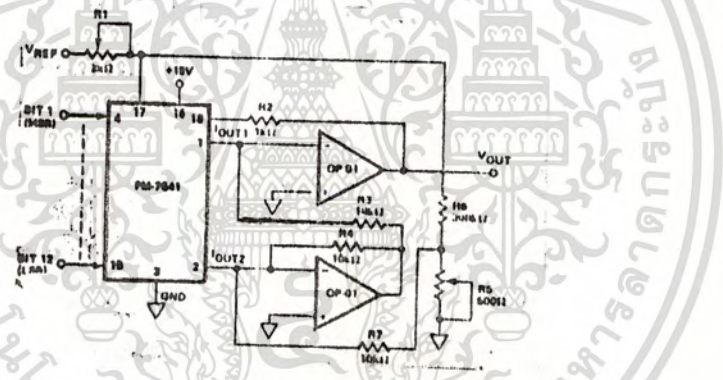
ส่วนที่3 เป็นส่วนที่ควบคุมในภาค ADC ที่จะเริ่มทำการแปลงสัญญาณ Input เมื่อไหร่โดยถ้าเราส่ง Logic"0" ก็เป็นสภาวะที่ยังไม่แปลง แต่ถ้าเป็นเป็น Logic"1" แสดงว่าเริ่มทำการแปลงสัญญาณ ในที่นี้จะสร้างสัญญาณ RUN

ส่วนที่4 จะเป็น Input ใช้ในการตรวจสอบสัญญาณจากภาค ADC "0" แสดงว่าทำการแปลงสัญญาณเรียบร้อยแล้วจึงจะส่งสัญญาณไปทำการอ่านจากภาค ADC ถ้าเป็น"1"ก็จะทำการตรวจสอบจนกว่าจะได้เป็น"0"

3.4 ส่วนการแปลงสัญญาณ DAC

3.4.1 การแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก (DAC)

ในการแปลงสัญญาณ Digital เป็นสัญญาณ Analog ในส่วนนี้เราจะใช้ IC PM 7541 ซึ่งมีความละเอียดถึง 12 บิท หรือเท่ากับ $2^{12} = 4096$ ระดับ เป็นแบบเชิงเส้น ภายในตัวมันจะอาศัยหลักการของ R,2R Ladder โดยที่สัญญาณอินพุท จะเอามาจากเอาท์พุทของ 8255 PortA และ PortC บน เหตุที่ต้องใช้ขนาดถึง 12 บิท ก็เพราะว่าในการส่งแรงดันไปควบคุมในภาคกำเนิดความถี่นั้นจะต้องมีความละเอียดพอสมควร เพื่อที่จะสามารถกำเนิดความถี่ได้ตามต้องการ วงจรที่ใช้ดังรูปที่ 3.9



รูปที่ 3.9 แสดงลักษณะวงจรแปลงสัญญาณดิจิตอลเป็นแอนะลอก

ซึ่งลักษณะของตัวมันมีการทำงานเป็นแบบ Bipolar Binary นั่นก็คือ อินพุทที่เป็น Binary นั้นสามารถที่จะควบคุมให้ DAC ทำการแปลงรหัสสัญญาณให้เอาท์พุทออกมาได้ทั้งค่าบวกและค่าลบ หรืออย่างใดอย่างหนึ่งก็ได้ ซึ่งอยู่กับลักษณะการจัดวงจร ในที่นี้เราจะอาศัยวงจรในคู่มือของบริษัทผู้ผลิต จะทำการจัดวงจรให้ทำงานแบบ Bipolar Binary ซึ่งแรงดันเอาท์พุทที่ได้จะมีค่าทั้งบวกและลบ โดยการป้อนสัญญาณดิจิตอล เข้าทางด้านอินพุทของ PM 7541 ดังแสดงในตารางที่ 3.2 เป็นการแสดงความสัมพันธ์ระหว่างสัญญาณดิจิตอล กับแรงดันเอาท์ที่ได้

ตารางที่ 3.2 แสดงความสัมพันธ์ระหว่างข้อมูลดิจิตอลกับแรงดันเอาท์พุท

Digital Input	Nominal Analog Output
111111111111	-0.99951 Vref
100000000001	-0.00049 Vref
100000000000	0
010000000000	+0.50000 Vref
000000000000	+1.00000 Vref

ซึ่งจากตารางความต้องการของเรา ต้องการที่จะให้ได้แรงดันเอาท์พุทสูงสุดที่ใช้ในการควบคุมการกำเนิดความถี่มีค่าเท่ากับ 10 โวลท์ เป็นแรงดันไฟบวก ดังนั้นจากตารางที่ 3.2 ทำให้เรารู้ว่า สัญญาณดิจิตอลที่ต้องการเพื่อให้แรงดันเอาท์พุทที่ต้องการจะอยู่ในช่วง 000H จนถึง 800H ซึ่งก็จะทำให้เราได้แรงดันเอาท์พุทที่มีค่าระหว่าง 0 ถึง 10 โวลท์ ไปใช้ควบคุมการกำเนิดความถี่ได้ โดยในการ Set ค่าของ PM 7541 เราจะต้องทำการป้อนสัญญาณดิจิตอลเข้าทางด้านอินพุทของ PM 7541 มีค่าเท่ากับ 000H แล้วทำการจับที่เอาท์พุทของวงจร อ่านค่าทำการปรับตัวต้านทาน R1 จนกว่าที่จะอ่านค่าแรงดันเอาท์พุทที่ได้ตรงตามที่ต้องการ (ในที่นี้คือ 10 โวลท์) ความละเอียดของวงจรขึ้นอยู่กับการ Set ค่าตัวต้านทาน 10K ที่ต่ออยู่ทางด้านเอาท์พุทของ PM 7541 ซึ่งจะต้องให้มีความละเอียดพอควรและค่าความต้านทานนี้จะต้องมีค่าที่ไม่ต่างกันมากนัก ในที่นี้จึงใช้ตัวต้านทานที่มีความผิดพลาด 1 เปอร์เซนต์ ในการส่งข้อมูลออกมาทาง Data Bus นั้นจะพบว่าเราจะต้องทำการ Latch ข้อมูลก่อนป้อนเข้า PM 7541 ก่อน เพื่อที่จะสามารถหยุดข้อมูลหรือต้องการดูได้ในแต่ละช่วงในที่นี้เราจึงได้ใช้ 74LS373 ซึ่งเป็น D F/F เราจึงใช้ 2 ตัวเพราะเหตุว่า DAC นั้นมีขนาด 12 Bit ดังนั้นเราจึงต้องทำการ Decode สัญญาณเพื่อที่จะส่งข้อมูล High Byte และ Low Byte ไปให้กับ PM 7541 ทำการแปลง ซึ่งการ Decoder จะแสดงได้ดังต่อไปนี้

ตารางที่ 3.2 แสดงความสัมพันธ์ระหว่างข้อมูลดิจิตอลกับแรงดันเอาท์พุท

Digital Input	Nominal Analog Output
111111111111	-0.99951 Vref
100000000001	-0.00049 Vref
100000000000	0
010000000000	+0.50000 Vref
000000000000	+1.00000 Vref

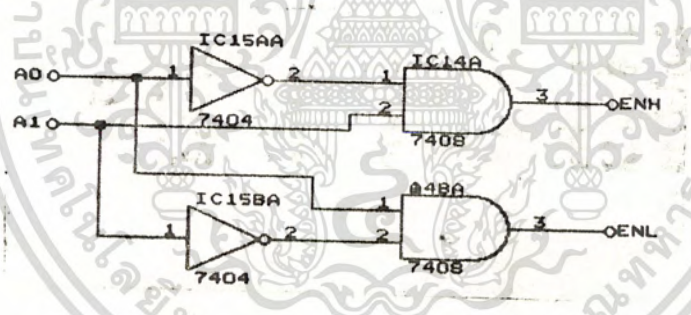
ซึ่งจากตารางความต้องการของเรา ต้องการที่จะให้ได้แรงดันเอาท์พุทสูงสุดที่ใช้ในการควบคุมการกำเนิดความถี่มีค่าเท่ากับ 10 โวลท์ เป็นแรงดันไฟบวก ดังนั้นจากตารางที่ 3.2 ทำให้เรารู้ว่า สัญญาณดิจิตอลที่ต้องการเพื่อให้แรงดันเอาท์พุทที่ต้องการจะอยู่ในช่วง 000H จนถึง 800H ซึ่งก็จะทำให้เราได้แรงดันเอาท์พุทที่มีค่าระหว่าง 0 ถึง 10 โวลท์ ไปใช้ควบคุมการกำเนิดความถี่ได้ โดยในการ Set ค่าของ PM 7541 เราจะต้องทำการป้อนสัญญาณดิจิตอลเข้าทางด้านอินพุทของ PM 7541 มีค่าเท่ากับ 000H แล้วทำการจับที่เอาท์พุทของวงจร อ่านค่าทำการปรับตัวต้านทาน R1 จนกว่าที่จะอ่านค่าแรงดันเอาท์พุทที่ได้ตรงตามที่ต้องการ (ในที่นี้คือ 10 โวลท์) ความละเอียดของวงจรขึ้นอยู่กับการ Set ค่าตัวต้านทาน 10K ที่ต่ออยู่ทางด้านเอาท์พุทของ PM 7541 ซึ่งจะต้องให้มีความละเอียดพอควรและค่าความต้านทานนี้จะต้องมีค่าที่ไม่ต่างกันมากนัก ในที่นี้จึงใช้ตัวต้านทานที่มีความผิดพลาด 1 เปอร์เซนต์ ในการส่งข้อมูลออกมาทาง Data Bus นั้นจะพบว่าเราจะต้องทำการ Latch ข้อมูลก่อนป้อนเข้า PM 7541 ก่อน เพื่อที่จะสามารถหยุดข้อมูลหรือต้องการดูได้ในแต่ละช่วงในที่นี้เราจึงได้ใช้ 74LS373 ซึ่งเป็น D F/F เราจึงใช้ 2 ตัวเพราะเหตุว่า DAC นั้นมีขนาด 12 Bit ดังนั้นเราจึงต้องทำการ Decode สัญญาณเพื่อที่จะส่งข้อมูล High Byte และ Low Byte ไปให้กับ PM 7541 ทำการแปลง ซึ่งการ Decoder จะแสดงได้ดังต่อไปนี้

ตารางที่ 3.3 แสดงการถอดรหัสสัญญาณในการส่งข้อมูลให้กับ PM7541

A1	A0	ENH	ENL
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

จากตารางที่ 3.3 เราจะมาทำการออกแบบ วงจร โดยเราจะต้องดูตรงสภาวะที่ให้เอาท์พุทเป็น Logic '1' ของสัญญาณ ENH ก่อนจะได้ A1.A0 และเอาท์พุทที่เป็น Logic '1' ของสัญญาณ ENL จะได้ A1.A0 แล้วนำไปสร้างวงจร

จากสมการ Boolean ของ A1 และ A0 จะได้



รูปที่ 3.10 แสดงวงจรในการส่งข้อมูลให้กับ DAC

ซึ่งในที่นี้เอาท์พุท ของ PM 7541 นั้นเราจะให้มันทำหน้าที่เป็นตัวควบคุมการกำเนิดความถี่ โดยจะนำแรงดันจากเอาท์พุทของ DAC ไปเข้าวงจรเปรียบเทียบ เพื่อสร้างสัญญาณที่ใช้ควบคุมการกำเนิดความถี่ในส่วนของ Sweep Gen อีกทีหนึ่ง

3.5 การออกแบบส่วนควบคุมอุปกรณ์ภายนอก

ในส่วนนี้จะเป็นการออกแบบวงจรเพื่อที่จะนำไปควบคุมอุปกรณ์ภายนอก ซึ่งเราจะพบว่าถ้านำเอาเอ๊าท์พุทของ Port 8255 ไปควบคุมอุปกรณ์ภายนอกนั้นพบว่าไม่สามารถทำได้แน่นอนเนื่องจากว่าแรงดันต่ำและถ้ายิ่งนำไป ควบคุมอุปกรณ์จำพวกใช้แรงดันเอซีแล้วย่อมเกิดปัญหาแน่นอน ซึ่งมีไฟสูงและเป็นไฟสลับอาจจะเป็นอันตรายต่ออุปกรณ์ภายใน Board และ Computer ได้ เราจึงทำการแยกระหว่างไฟสูงกับไฟต่ำออกจากกันโดยเอาไฟต่ำไป ควบคุมไฟสูงอีกทีหนึ่ง เราจึงใช้อุปกรณ์จำพวก Opto-Isolator หรือ Solid State Relay แต่ในที่นี้เราจะใช้ Solid State Relay ลักษณะวงจรที่ใช้จะเป็นดังรูปที่ในภาคผนวก

จากวงจรที่ในภาคผนวก เราจะแสดงวงจรในการขับอุปกรณ์ภายนอกมาให้ดูเพียง 4 CH เท่านั้นถ้าเราต้องการมากกว่านี้เราก็ทำการเพิ่มขึ้นได้แล้วแต่ความต้องการ ซึ่งในโปรแกรมที่ได้เขียนขึ้นมานี้เราสามารถที่จะทำการควบคุมอุปกรณ์ภายนอกได้ถึง 16 CH โดยต่อออกมาจากขา 1-16 ของ K1 ซึ่งจากวงจรเรายังแสดงให้ดูถึงการควบคุมอุปกรณ์ โดยการใช้สวิทช์ควบคุมภายนอกในส่วนนี้ก็จะมีโปรแกรมในการควบคุม ซึ่งจะแสดงสภาวะของสวิทช์ในแต่ละตำแหน่งบนจอ Monitor

3.6 ส่วนของแหล่งจ่ายไฟ

ในส่วนนี้จะพบว่าเราควรจะเลือกใช้ แหล่งจ่ายไฟเท่าไหร่จึงจะเหมาะสมต่องานของเรา ซึ่งจะต้องทำการกำหนดความต้องการของแต่ละส่วนเพื่อให้ระบบทำงานได้อย่างมีประสิทธิภาพ เราจึงแบ่งออกเป็นส่วนๆ ดังต่อไปนี้

แหล่งจ่ายไฟ	+12 V, -12 V	ต้องการที่จะจ่ายให้กับ Op-Amp ทุกตัวและ IC PM 7541 (จะใช้เฉพาะไฟบวก)
	+5 V	ต้องการที่จะจ่ายแรงดันให้กับอุปกรณ์จำพวก TTL
	+5 V, -5 V	ต้องการที่จะจ่ายให้กับ ICL 7109,4051

เมื่อทำการออกแบบเราพบว่ามีไอซี Regulate ที่สามารถจะจ่ายแรงดันไฟบวก และ ลบ 5V ได้ เราจึงเลือกเบอร์ 7805 ซึ่งจะได้เอาท์พุทเป็นไฟบวก และ 7905 จะได้เอาท์พุทเป็นไฟลบมาใช้งาน

ส่วนในการออกแบบแหล่งจ่ายไฟขนาด 10V ซึ่งจะเป็นแรงดันไฟอ้างอิงคงที่ให้กับ PM 7541 นั้น จะพบว่าทำไมเราไม่ใช้ ICRegulate ที่ให้เอาท์พุท 10V เลย เพราะเหตุว่าเอาท์พุทที่ออกมาจาก IC เหล่านี้จะไม่ได้ค่าตามที่บอกไว้ จะผิดพลาดเล็กน้อยเพราะเหตุว่าไอซีเหล่านี้ที่มีขายตามท้องตลาดไม่ดีเท่าที่ควร เราจึงมาเลือกใช้ ไอซีที่สามารถปรับแรงดันได้ตามความต้องการและมีความละเอียดกว่า จึงเลือกใช้เบอร์ LM-317เหตุที่ต้องเลือกใช้เพราะเหตุว่ามันจะให้เอาท์พุทที่ละเอียดและปรับแรงดันได้ตามความต้องการเพื่อที่จะสามารจ่ายแรงดันคงที่จริงๆให้กับ PM 7541 ซึ่งในการคำนวณหาค่าของอุปกรณ์ต่างๆ เฉพาะแหล่งจ่ายไฟอ้างอิง 10 V จะได้

จะพบว่าความต้องการเราต้องการแรงดันเอาท์พุท เท่ากับ 10 โวลท์ เราจึงใช้แรงดันอินพุทที่ให้กับไอซีตัวนี้ เท่ากับ 12 โวลท์ และจากคู่มือของไอซีจะได้ $V_{ref} = 1.25V$, $I_{adj} = 50\ uA$ และถ้าเราเลือกใช้ ตัวต้านทาน R1 เท่ากับ 240 โอห์มเราจะต้องเลือกค่าตัวต้านทานที่ปรับค่าได้ R2 เท่าไรจึงจะเหมาะสม คำนวณ หาแรงดันที่เอาท์พุทจะได้ว่า

$$V_o = V_{ref} + (I_1 + I_{adj})*R_2 \qquad ....(3.28)$$

$$I_1 = V_{ref}/R_1 \qquad ....(3.29)$$

แทนค่าสมการที่ 3.29 ลงในสมการที่ 3.28 จะได้ว่า

$$V_o = V_{ref} + [(V_{ref}/R_1) + I_{adj}]*R_2$$

$$V_o = V_{ref} + V_{ref}*R_2/R_1 + I_{adj}*R_2$$

$$V_o = V_{ref}*(1 + R_2/R_1) + I_{adj}*R_2 \quad ....(3.30)$$

จะพบว่าจากสมการ 3.30 นั้น ค่าของ I_{adj} นั้นมีค่าน้อยมากๆ จากการหาค่าประมาณ ทำให้เทอมสุดท้ายมีค่าเป็นศูนย์ จะได้ว่า

$$V_o = V_{ref}*(1 + R_2/R_1) \quad ....(3.31)$$

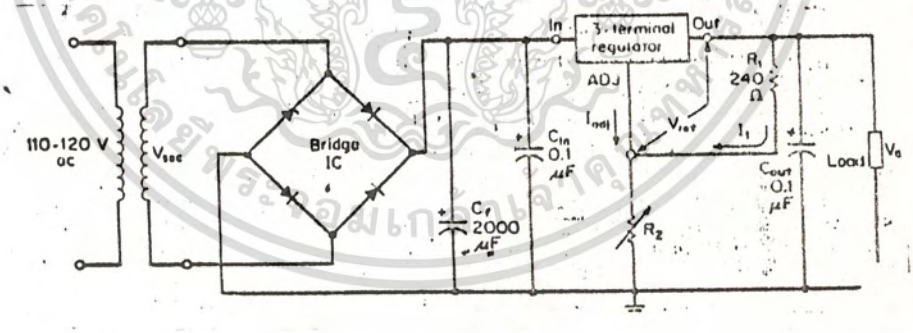
ทำการแทนค่าที่กำหนดไว้ในตอนต้น ลงในสมการ 3.31 จะได้ว่า

$$10\ V = 1.25*(1 + R_2/240) \quad ....(3.32)$$

ย้ายข้างสมการที่ 3.32 หาค่า R_2 จะได้ว่า

$10/1.25 = 1 + R_2/240$

$R_2 = (8-1)*240 = 1680$ โอห์ม เลือกใช้ 2K



รูปที่ 3.11 แสดงวงจรจ่ายแรงดันไฟบวก 5 โวลท์

บทที่ 4

การทดลองและผลการทดลอง

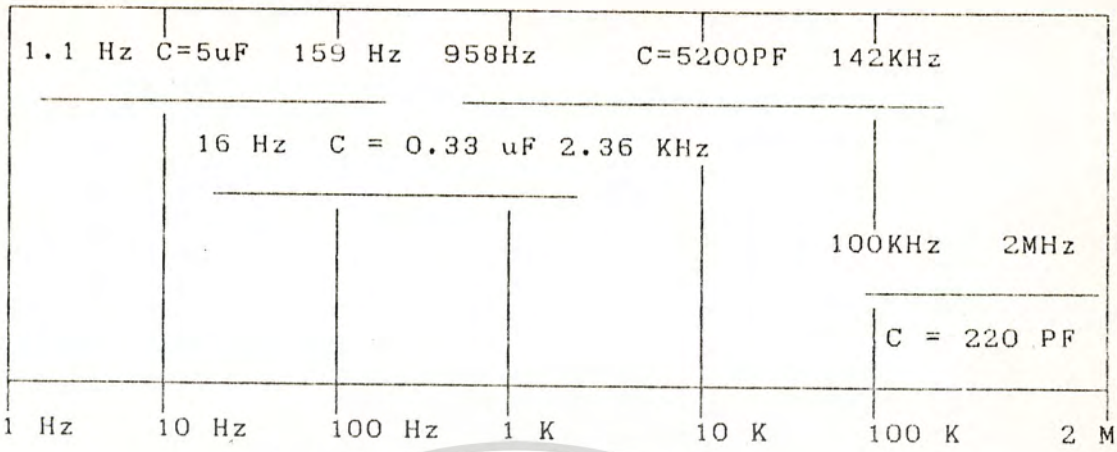
ในส่วนนี้จะเป็นการทดลองเกี่ยวกับวงจรที่มีส่วนเกี่ยวข้องกับ Board นี้ ซึ่งจากการทดลองและผลการทดลองที่ได้จะเป็นดังต่อไปนี้

4.1 การทดลองในส่วนของการติดต่อกับ I/O ในส่วนนี้เราใช้ IC 8255 ซึ่งในการทดลองนี้เราทำการเขียนโปรแกรม ควบคุมการเอ๊าท์ข้อมูลออกไปยัง Port 8255 และลองรับข้อมูลเข้าทาง Port 8255 ผลปรากฏว่าไม่สามารถที่จะทำการ เอ๊าท์ และ อิน ข้อมูลเข้าทาง Port 8255 ได้ จึงได้ทำการตรวจสอบผลปรากฏว่า GND ของ IC ไม่ได้ถูกต่อร่วมกัน เกิดสภาวะลอยจึงแก้ไขแล้วทำการทดลองส่งข้อมูลออกไปควบคุมการ ปิด-เปิด LED แล้วทำงานได้

4.2 การทดลองในส่วนของการ Decoder Address Portโดยในการทดลองในส่วนนี้เราทำการทดลอง IC PAL เมื่อเราทำการเขียนสมการ Boolean แล้วทำการ Mark จุดลงบน Fuse Map ตอนแรกคิดว่าสมการที่เขียนขึ้นมาแล้วทำการ Mark จุดลงบน Fuse Map เป็นจุดที่ Burn ทิ้งแล้วทำการทดลองผลปรากฏว่าไม่ทำงาน จึงคิดว่าสมการ Boolean ที่เขียนเป็นจุดที่ Fuse ยังคงอยู่นอกเหนือจากนั้นจะ Burn ทิ้งไปหมดใน Line เดียวกัน แล้วทำการทดลองใหม่ปรากฏว่าใช้งานได้

4.3 การทดลองในส่วนของ DAC เมื่อทำการทดลองแล้วทำการส่งข้อมูลออกไปผลปรากฏว่าข้อมูลออกมาเพียงแวบเดียว ซึ่งตามความเป็นจริงเราต้องการ Latch ข้อมูล แต่ทำไมจึงไม่เกิดการ Latch เมื่อทำการตรวจสอบแล้วพบว่าขา OE ของเราแทนที่จะต่อลง GND แต่เราไปต่อที่ขา Decoder Address ที่ออกจาก O/P ของ PAL จึงทำให้เกิดปัญหาได้ทำการแก้ไขแล้วทำการทดลองส่งข้อมูลออกไป ผลปรากฏว่าได้ตรงตามความต้องการ

4.4 การทดลองส่วนควบคุมการกำเนิดความถี่ ในส่วนการสร้างความถี่นี้พบว่าเราจะต้องมีการเลือกค่าตัวเก็บประจุที่ต่อคล่อมขา 5 และขา 6 ของ XR-2206 ให้มีค่าที่เหมาสมเพื่อที่จะสามารถทำให้กำเนิดความถี่ได้อยู่ในช่วงที่ต้องการ ซึ่งในการทดลองนี้เราได้ทำการเลือกค่าตัวเก็บประจุโดยนำค่าตัวเก็บประจุมาชุดหนึ่งแล้วทำ การทดลองผลปรากฏว่ามีตัวเก็บประจุอยู่ 4 ค่าที่ได้ผลตามต้องการ ซึ่งมีค่าดังนี้ 0.5 uF 0.33 uF , 5200 PF , 220 PF โดยค่าตัวเก็บประจุที่ได้ในแต่ละตัวนั้นสามารถแสดงให้เห็นช่วงของการกำเนิดความถี่ได้ดังแสดงดังรูปที่ 4.1

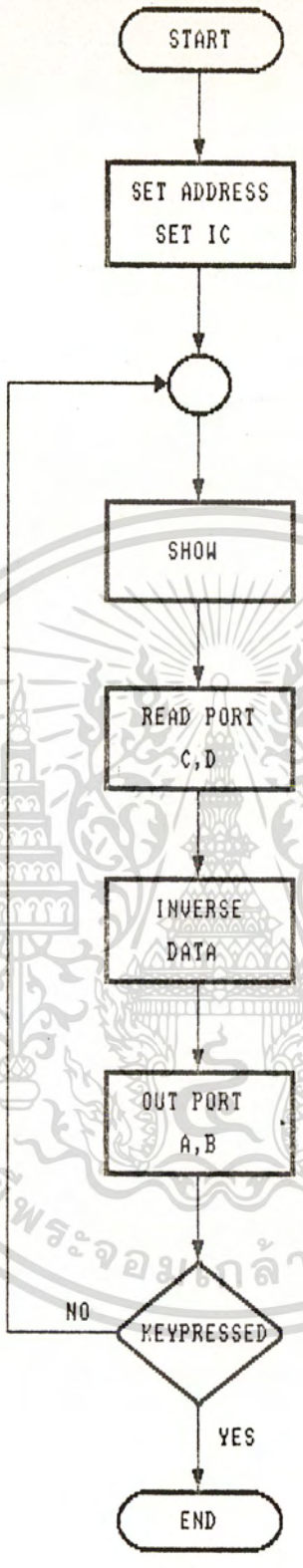


รูปที่ 4.1 แสดงค่าตัวเก็บประจุที่สามารถนำมาใช้งานได้ในแต่ละช่วง

โดยสัญญาณควบคุมในการเลือกตัวเก็บประจุจะถูกส่งออกมาทาง Port 8255 แล้วนำไปควบคุม แอนะลอกสวิทช์ให้ทำการเลือกตัวเก็บประจุผลปรากฏว่า วัดสัญญาณ O/P จาก XR-2206 แล้วไม่มีสัญญาณออกมา จึงคิดว่าไม่มีตัวเก็บประจุต่อคล่อมระหว่างขา 5,6 ของ XR-2206 จึงลองทดลองต่อ C คล่อมขา 5,6 ผลปรากฏใช้ได้ จึงทำให้เราเปลี่ยนแอนะลอก SW เป็นการใช้ Reed Relay แทนผลปรากฏใช้ได้

4.5 การทดลองในส่วนของ Timer/Counter เราทำการทดลองโดยความคุมการทำงานใน Mode ต่างๆ แล้วผลปรากฏว่า ไม่สามารถทำงานได้ จึงทำการตรวจสอบโปรแกรมดูผลปรากฏว่า ต้องแก้ไข Program เมื่อทำการแก้ไขแล้วทดลองผลปรากฏว่าใช้งานได้

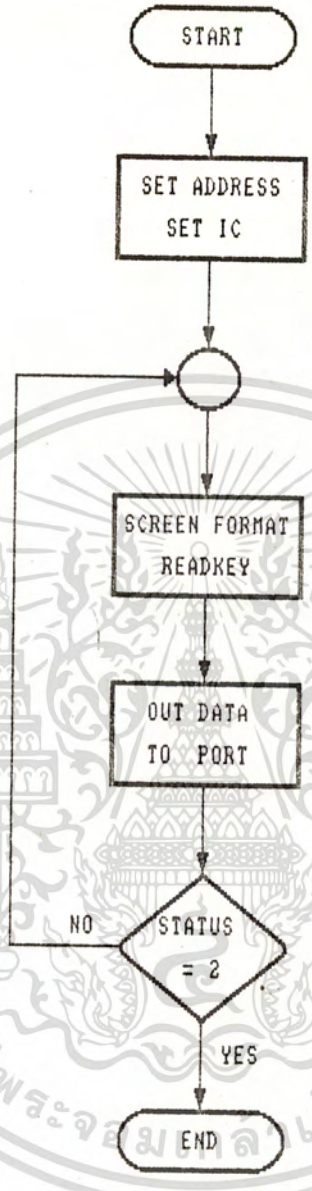
4.6 การทดลองในส่วนของการประยุกต์ใช้งาน ซึ่งในส่วนนี้เราได้ทำการเขียน Program ที่ใช้ควบคุมระบบภายนอกมาแสดงไว้ในส่วนของ Demo ดังต่อไปนี้



DEMO

การทำงานของ Program Demo

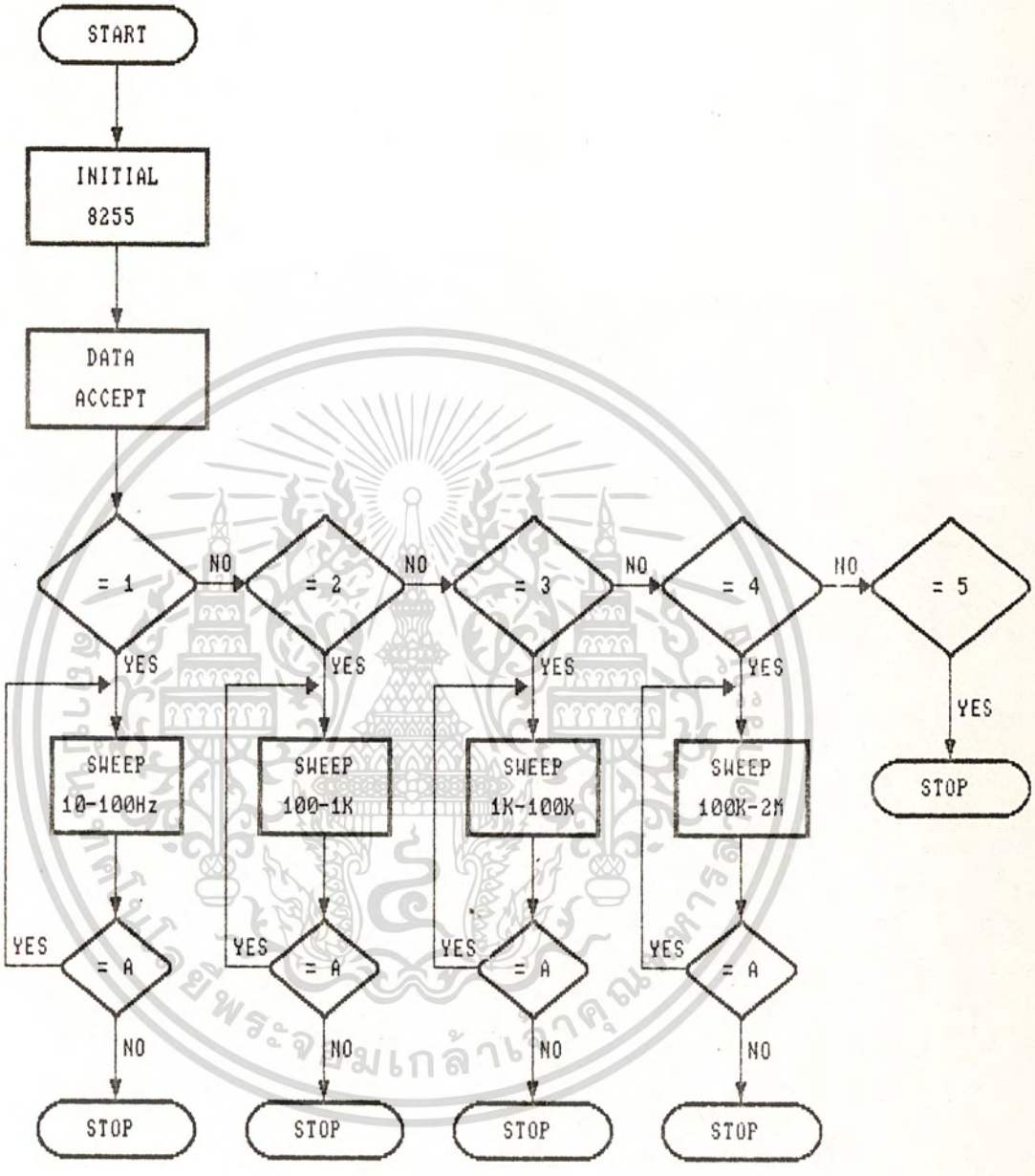
Program จะเริ่มจากการตั้งค่า Address ของการ์ด จากนั้นก็จะทำการ set IC โดยทำการส่ง control word ไปยัง Port C1 และ Port C2 โดยที่ Port C1 ซึ่งเป็นของ IC 8255 ตัวที่ 1 นั้นเราจะ set ให้เป็น output หมดดังนั้นจึงส่งค่า $80 ไปส่วน Port C2 ซึ่งเป็นของ IC 8255 ตัวที่ 2 เราจะ setให้เป็น input หมดดังนั้นจึงส่ง $9B ไป หลังจาก set IC แล้วก็จะทำการอ่านจากPort C กับ Port D จากนั้นก็จะทำการ inverse แล้วส่งให้Port A กับ Port B ค่าที่อ่านได้จะมาทำการเก็บไว้ในตัวแปรซึ่งจะเอาไว้ทำการเปรียบเทียบกับค่าที่จะอ่านเข้ามาครั้งต่อไป ซึ่งเมื่อเปรียบเทียบกันแล้วมีการเปลี่ยนแปลงเกิดขึ้นก็จะทำการส่งค่าใหม่ที่เปลี่ยนแปลงนี้ออกไป พร้อมทั้งแสดงผลการเปลี่ยนแปลงขึ้นที่หน้าจอด้วยประโยชน์ของ Program นี้ ก็เช่นนำมาทำเป็น remote control



DEMO1

การทำงานของ Program Demo1

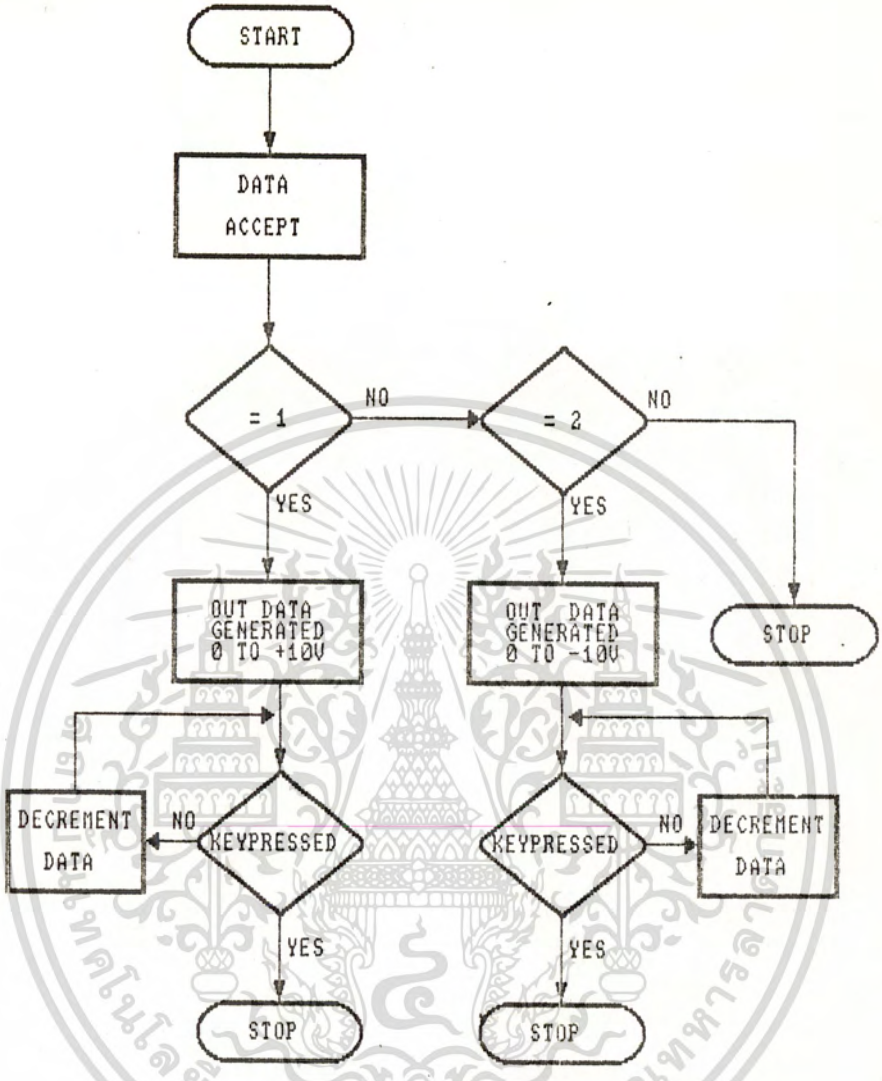
อันดับแรก Program จะให้เราตั้งค่า Address จากนั้นก็จะทำการเซ็ต IC 8255 โดยส่ง control word ไปที่ Register ควบคุมซึ่งมีรายละเอียดอยู่ในทฤษฎีแล้ว สำหรับ Program นี้ต้องการให้ Port A และ Port B เป็น output ดังนั้นจึงส่ง $80 เมื่อทำการ set เรียบร้อยแล้วก็จะเป็นการสร้างรูปแบบบนจอภาพเพื่อแสดงสถานะของ bit ต่างๆของPort A และ Port B ซึ่งในการ run Program ครั้งแรกจะ set ให้เป็น OFF หมด จากนั้นก็จะเป็นขบวนการรับค่าจาก Keyboardซึ่งแบ่งออกเป็นการเลือกGroup การเลือก bit และการเลือกสภาวะโดยที่ให้ Port A เป็น Group 1 และ Port B เป็น Group 2 สำหรับการเลือก bit นั้นจะให้ 1bit เป็น SW 1 ตัว ดังนั้นจึงสามารถเลือกได้ตั้งแต่ 1 ถึง 8 และสุดท้ายคือการเลือกสภาวะว่าจะให้ ON หรือ OFF ซึ่งจะใช้การป้อนเลขคือ 1=ON และ 0=OFF เมื่อทำการเลือกเสร็จแล้ว ส่วนที่เลือกไว้ก็จะเปลี่ยนแปลงตามที่เราต้องการ จากนั้นก็จะกลับมารอรับคีย์ต่อไปสำหรับการออกจาก Program เราจะใช้การป้อนเลข 2 ที่การเลือกสภาวะ เราสามารถที่จะนำ Program นี้ไปใช้ควบคุมการปิดเปิดอุปกรณ์ไฟฟ้าเช่นหลอดไฟแสงสว่าง พัดลม เป็นต้น



DEMO2

DEMO2

Demo 2 นี้จะเป็นการแสดงเกี่ยวกับการควบคุมการกำเนิดความถี่รูป Sine Wave โดยจะทำให้เกิดการ Sweep ตามความต้องการขึ้นอยู่กับการเลือกหมายเลข ขบวนการทำงานเป็นดัง Flow Chart ที่ได้แสดงไว้ โดยเริ่มต้นจากการ Initial 8255 ให้ PortE เป็น Port เอาท์พุทก็เพื่อที่จะส่งสัญญาณควบคุมไปทำการเลือกตัวเก็บประจุให้กับ XR-2206 ในส่วนของวงจร Sweep หลังจากนั้นก็จะทำการรอรับ Key หมายเลขที่ต้องการขึ้นอยู่กับว่าต้องการความถี่เท่าไรก็ทำการเลือก เมื่อขบวนการเสร็จสิ้นแล้วก็จะมีการถามว่าต้องการที่จะทำงานอีกครั้งหรือไม่ถ้าไม่ต้องการก็กด Key อะไรก็ได้ถ้าต้องการก็ให้กด A และถ้าทำการเลือกหมายเลขที่เกินจาก 4 ก็จะกับไปยัง Program

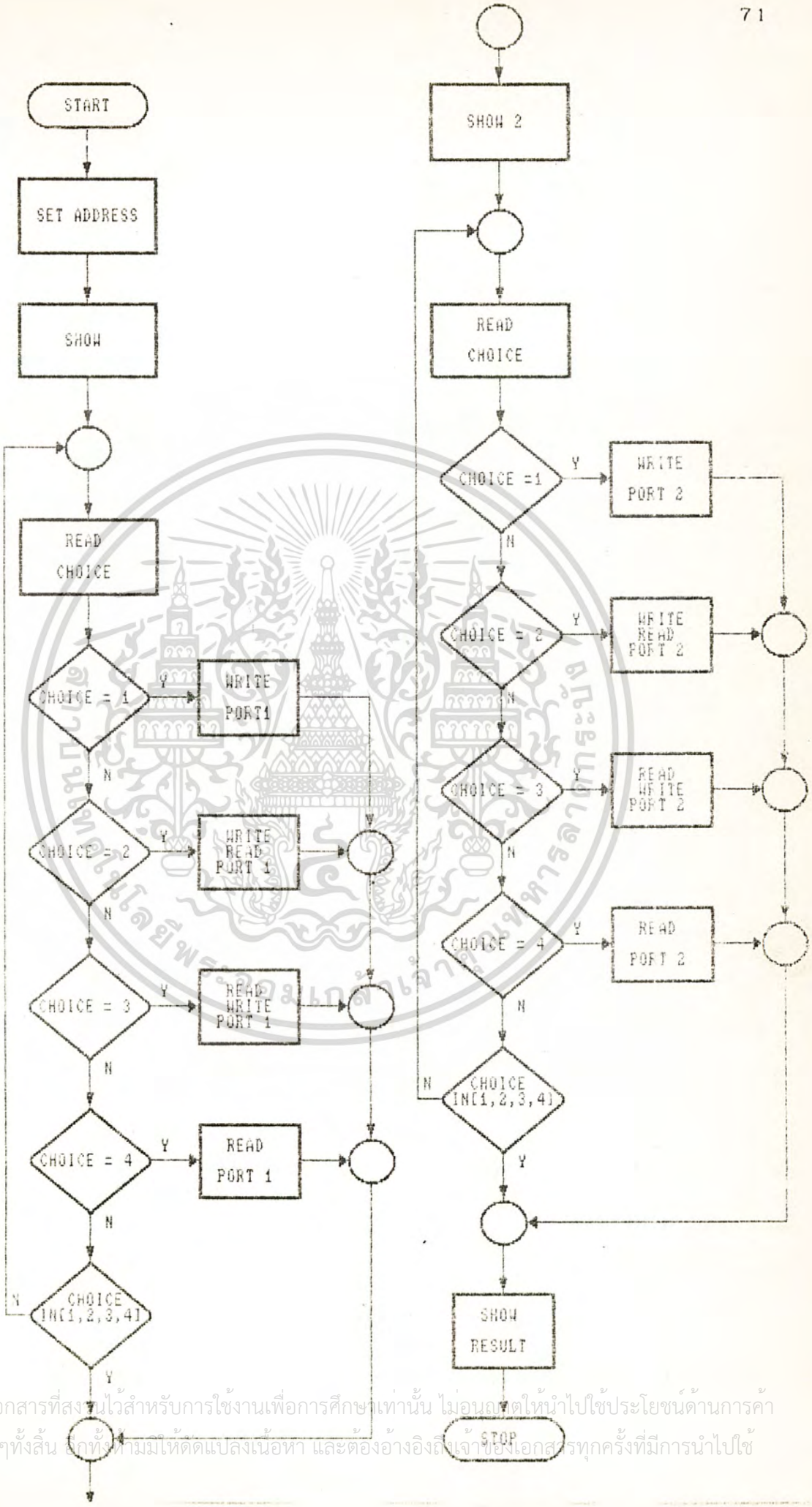


DEMO3

DEMO3

ในโปรแกรม Demo3 นี้จะแสดงการทำงานเป็น Voltage Recgulator ซึ่งได้แสดงใน Flow Chart โดยเราสามารถอธิบายได้ดังต่อไปนี้ เริ่มต้นด้วยการรอรับข้อมูล โดยมีการให้เลือกว่าต้องการแรงดันเอ๊าท์พุทที่เป็นบวกหรือลบตั้งแต่ 0-10v โดยถ้ากดหมายเลข 1 ก็จะทำการส่งข้อมูลออกไปควบคุม ตัวแปลงสัญญาณดิจิตอลเป็นแอนะลอก ทำการแปลงให้เป็นแรงดัน โดยจะ Generate แรงดัน 0-10 โวลท์ เราจะส่งข้อมูลตั้งแต่ 800H-00H แล้วในแต่ละข้อมูลเราจะมีการหน่วงเพื่อให้ผู้ใช้งานดูว่าถึงแรงดันที่ต้องการหรือไม่ ถ้าใช่ให้ทำการกด Key อะไรก็ได้ก็จะได้แรงดันที่ต้องการ

ส่วนถ้าเลือกหมายเลข 2 ก็จะทำการส่งข้อมูลที่ ตั้งแต่ 800H-FFFH ซึ่งจะทำให้ DAC Generate แรงดันในช่วงลบและขบวนขั้นต่อไปก็จะเหมือนกับช่วงบวก



การทำงานของ Progam Demo4

จาก Flow Chart เราจะเริ่มโดยการ Set Address ของการ์ดก่อนจากนั้นก็จะเป็นการแสดงหัวข้อให้เลือกซึ่งเป็นของ Port A และPort B โดยมีด้วยกัน 4 หัวข้อ เมื่อเราเลือกหัวข้อแล้วโปรแกรมก็จะทำการเช็คว่าเราเลือกข้อไหน แล้วก็จะไปทำตามโปรแกรมย่อยของข้อนั้นซึ่งแต่ละข้อจะมีโปรแกรมย่อยคือ

ข้อ1 : จะทำการรับข้อมูลจาก KeyBoard มาไว้ในตัวแปรซึ่งเป็นแบบ Array โดยที่การรับจะอยู่ในรูปของเลขฐาน 2 จากนั้นก็จะแปลงเลขฐาน 2 ให้เป็นเลขฐาน 10 เก็บไว้ในตัวแปร Data A และ Data B เสร็จแล้วก็จะทำการ Set IC ให้ Port A และ Port B เป็น Outputและทำการส่ง Data A,Data B ออกไปที่ Port A,Port B ตามลำดับ

ข้อ2 : จะรับข้อมูลเข้ามาในรูปของเลขฐาน 2 แล้วแปลงเป็นเลขฐาน 10 เก็บไว้ใน Data A ทำการ Set IC ให้ Port A เป็น Outputและ Port B เป็น Input ส่งค่า Data A ออกไปที่ Port A และอ่านค่าจาก port B มาไว้ใน Data B ทำการแปลงเป็นเลขฐาน 2 แสดงผลออกทางจอภาพ

ข้อ3 : จะรับข้อมูลมาเก็บไว้ในตัวแปร Data B ทำการ Set ICให้ Port A เป็น Input และ Port B เป็น Output อ่านค่าจาก Port A เก็บไว้ใน Data A ทำการแปลงเป็นเลขฐาน 2 และแสดงผลออกจอภาพแล้วส่งค่า Data B ออกไปที่ Port B

ข้อ4 : จะ Set IC ให้ทั้ง Port A และ Port B เป็น Input จากนั้นก็จะทำการอ่านค่าจาก Port A และ Port B มาเก็บไว้ในตัวแปรData A และ Data B ตามลำดับ ทำการแปลงเป็นเลขฐาน 2 แล้วแสดงผลออกทางจอภาพ

หลังจากเสร็จแล้วก็จะเป็นการทำงานของ Show2 คือการแสดงหัวข้อให้เลือกแต่จะเป็นของ Port C และ Port Dโดยจะมี 4 หัวข้อเหมือนกันและการทำงานในแต่ละหัวข้อก็จะเหมือนกับกรณีของ Port A และ Port Bเพียงแต่จะเปลี่ยนจาก Port A เป็น Port C และ Port B เป็น port D

FLOWCHART OF PROGRAM DEMONSTARTE IC8253 PROGRAMMABLE TIMER

การทำงานของ Program Demo5

Program นี้เป็นการแสดงการทำงานของ IC8253ใน Mode ต่างๆซึ่งมีการทำงานของ Program ดังนี้คือ

จาก Flow Chart เราจะเริ่มโดยการ Set Address ของการ์ดซึ่งการ์ดสามารถที่จะเลือก Address โดยการเปลี่ยนจุด Jump สายบนการ์ดโดยมี Address อยู่ 2 Address คือที่ 30XH และ 31XH แต่ในที่นี้จะใช้ Address ที่ 30XH ดังนั้นจึงเลือกข้อ 1 จากนั้นก็จะทำการเลือก Counter ที่มีอยู่ 3 Counter คือ Counter0, Counter1, และ Counter2 เมื่อเลือก Counterc แล้วก็จะเป็นการเลือก Mode การทำงานของ Counter ซึ่งมีอยู่ 6 Mode คือ

Mode0 : Interrupt on terminal count
Mode1 : Programable one-shot
Mode2 : Rate generater
Mode3 : Square wave generater
Mode4 : Software triggered strobe
Mode5 : Hardware triggered strobe

สำหรับรายละเอียดของแต่ละ Mode นั้นจะมีอยู่ในภาคทฤษฎี จากนั้นก็จะเป็นการเลือกลักษณะการนับของ Counter ว่าจะให้นับเป็น BCD(Binary Code Decimal) หรือเป็น Binary โดยที่ถ้าเป็นการนับแบบ BCD จะนับได้สูงสุดเท่ากับ 10^4และถ้าเป็นการนับแบบ Binary จะนับได้สูงสุดเท่ากับ 2^{16} เมื่อเลือกเสร็จแล้วก็จะเป็นการป้อนข้อมูลที่จะให้ Counter ทำการนับจากนั้น Program ก็จะบอกให้ต่อขา Output ของ Counter ที่ได้ทำการเลือกไว้เข้ากับ Osciloscope และกด Enter เพื่อทำการ Start Counter หลังจาก Start แล้วเราสามารถที่จะมาทำการเลือก Counter ตัวอื่นต่อไปได้โดยไม่มีผลต่อ Counter ที่กำลังนับอยู่และเมื่อต้องการออกจาก Program ก็ทำได้โดยการกดปุ่ม Q เมื่อมีอักษรขึ้นบนจอว่า Press Q to quit

บทที่ 5

สรุปผลของโครงงานและข้อเสนอแนะ

5.1 สรุปผลที่ได้จากโครงงาน

จะพบว่าในการควบคุมอุปกรณ์ภายนอกโดยใช้ Board นี้ต้องการเพียง 32 CH ซึ่งไม่ได้ถูกจำกัดโดยเราสามารถที่จะทำให้ได้มากกว่านี้ก็ได้ โดยใน Board นี้ได้ทำสัญญาณในการการควบคุมให้สามารใช้ได้อีกตัวหนึ่งโดยถ้าต้องการจะต้องทำการต่อภายนอก Board เนื่องจากว่าไม่ได้เผื่อเนื้อที่ไว้ให้ จากการทดลองการทำงานในส่วนนี้ก็เป็นไปตามที่คาดหวังไว้ ส่วนในการใช้งานใน Board นี้ในส่วนของ DAC เราทำหน้าที่ 2 อย่าง คือ กำเนิดแรงดันคงที่ และสร้างความที่ Sine Wave ซึ่งจากการทดลองการกำเนิดความถี่พบว่าส่วนที่สำคัญที่สุดก็คือส่วนที่ควบคุมการ Sweep ในการทดลองนั้นพบว่าย่านความถี่ที่ใช้งานโดยมีการให้เลือกใช้ เมื่อทำการส่งข้อมูลไปควบคุมการกำเนิดความถี่นั้นตาม Spec อัตราการกำเนิดความถี่สูงจะเป็น 2000 เท่าของการกำเนิดความถี่ต่ำ ซึ่งจากการทดลองทำจะพบว่า Range ของการ Sweep แคบมาก ซึ่งอาจจะเป็นสาเหตุมาจาก ตัวเก็บประจุ หรือ การ Set ค่าในภาคควบคุมการ Sweep

5.2 ข้อเสนอแนะในการพัฒนาโครงงาน

ในโครงงานที่ได้ทำขึ้นนี้ก็เพื่อที่ จะสามารถที่จะทำให้อุปกรณ์ภายนอกนั้นสามารถที่จะทำการติดต่อกับ IBM PC ได้ ซึ่งจากการประยุกต์ใช้งานไม่งานไม่ได้เพียงแต่จำกัดอยู่ที่ การควบคุมการปิด-เปิดอุปกรณ์ชนิดเอซีอย่างเดียว ซึ่งสามารถที่จะประยุกต์ใช้งานได้อย่างมากมาย ขึ้นอยู่กับความต้องการของแต่ละคน ซึ่งในส่วนสำคัญของการที่จะใช้ประโยชน์ของ Board นี้ให้ได้มากที่สุดนั้น ก็อยู่กับ Soft Ware โดยถ้าเป็นไปได้ควรจะสามารถควบคุมอุปกรณ์ทั้งระบบภายในโรงงานไม่ได้เพียงแต่ควบคุมการปิด-เปิดอุปกรณ์เท่านั้น และในส่วนของการสร้างสัญญาณรป Sine จะพบว่ามันมีประโยชน์มากซึ่งถ้าเรานำสัญญาณนี้ได้ใช้งานในด้านอื่นๆก็สามารถทำได้ เช่น สามารถที่จะทำการหาผลการตอบสนองของวงจร Filter ต่างๆทำได้ ซึ่งในการหาผลการตอบสนองของวงจรเหล่านั้น จากการทดลองเราทำการป้อนความถี่ค่าหนึ่งแล้วทำการวัดแรงดันเอ๊าท์พุทได้ค่าหนึ่งแล้วนำมาเข้าสมการ Log เพื่อจะนำไป Plot โดยใน Board นี้นั้นทำให้เกิดการ Sweep สัญญาณ Sine Wave

เราก็หาส่วนเพื่อเติมภายนอกอีกเล็กน้อยก็สามารถทำการหาผลการตอบสนองของวงจรเหล่านั้น ย่อมทำได้

เอกสารอ้างอิง

1. กองบรรณาธิการ. PAL ลอจิกเกตสารพัดนึก. เซมิคอนดัคเตอร์อิเลค-ทรอนิคส์. ฉบับที่ 83 (ธันวาคม 2530) 150-158

2. ธานินทร์ ถาวรศาสนวงศ์. ทินกร ดุ๊ก. การอินเตอร์เฟส IBM PC. กรุงเทพมหานคร : ฟิสิกส์เซนเตอร์, 2532

3. Lewis, Eggebrecht C. Interface To The IBM Personal Computer. 1st edition. Indiana : Howard W. Sams & Co., Inc., 1983

4. Katsuhiko Ogata. Modern Control Engineering. 13th Edition. New Delhi : Prentice Hall, 1970

5. Sergio Franco. Design With Operational Amplifier And Analog Integrated circuit. 1st Edition. U.S.A. : McGraw-Hill, 1988

6. Kennedy E.J. (Eldredge Johnson). Operation Amplifier Circuit Theory and Applications. 1st Edition. New York : Holt, Rinehart and Winston, Inc., 1988

7. Ramakant Gayakwad A. OP-Amp and Linear Integrated Circuit. 1st Edition. U.S.A. : Prentice-Hall, 1988

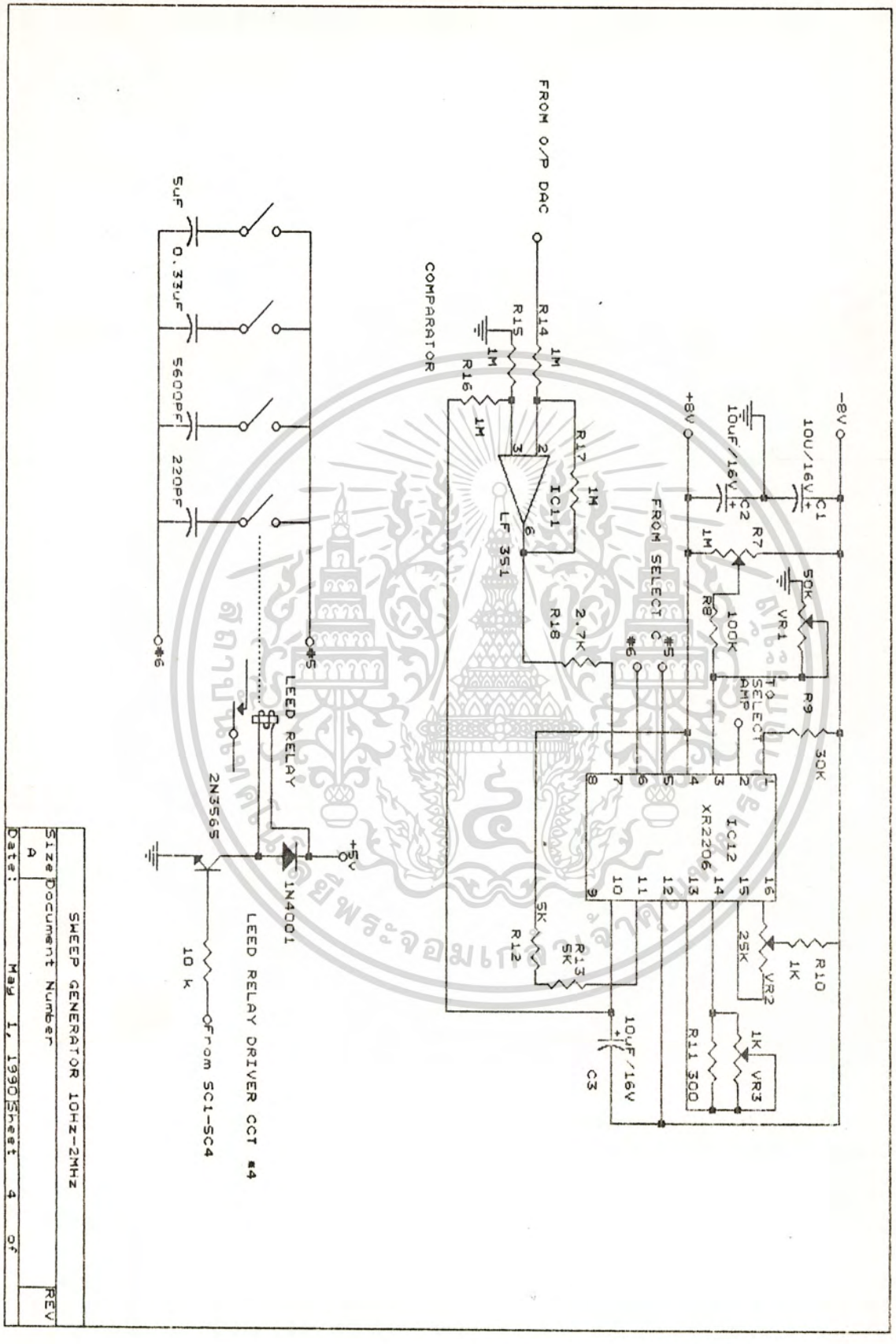
8. Harry Helms. Linear IC Device 1987 Source Book. 1st Edition. U.S.A. : Technipubs, Inc, 1987

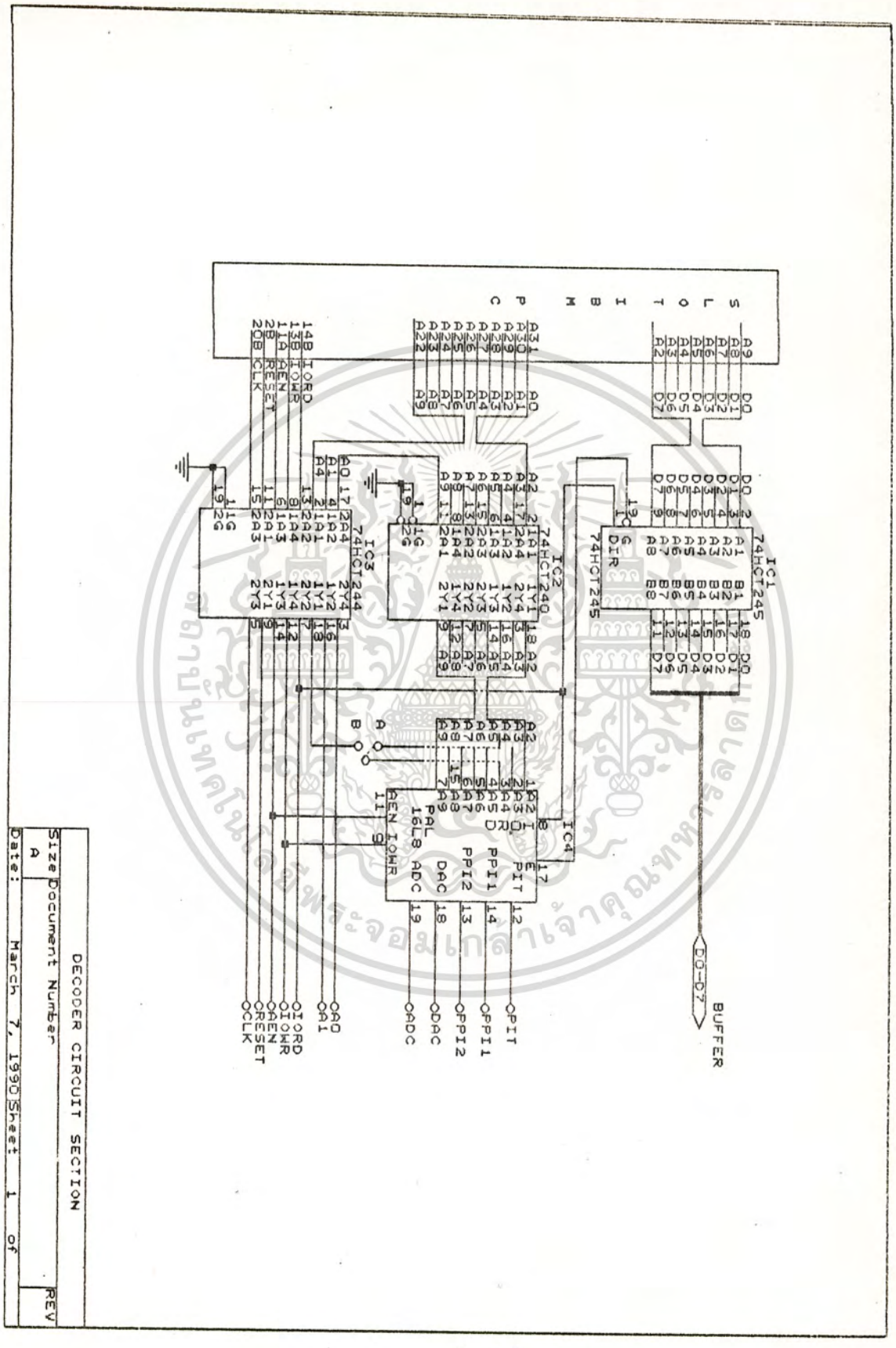
9. Walter Jung G. IC OP-AMP Cookbook. 1st edition. U.S.A. Howard W.Sam & Co., Inc., 1977

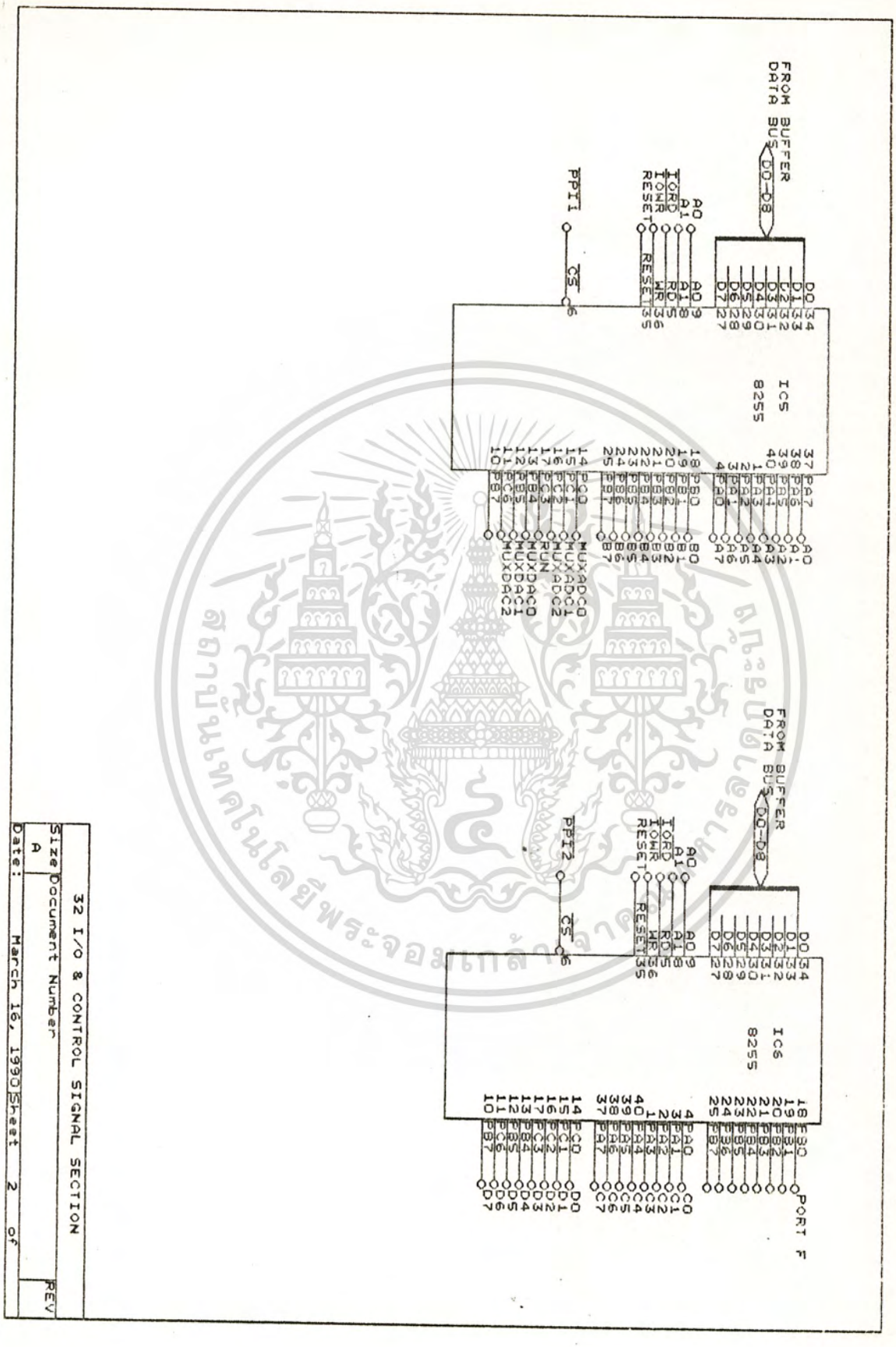
กิตติกรรมประกาศ

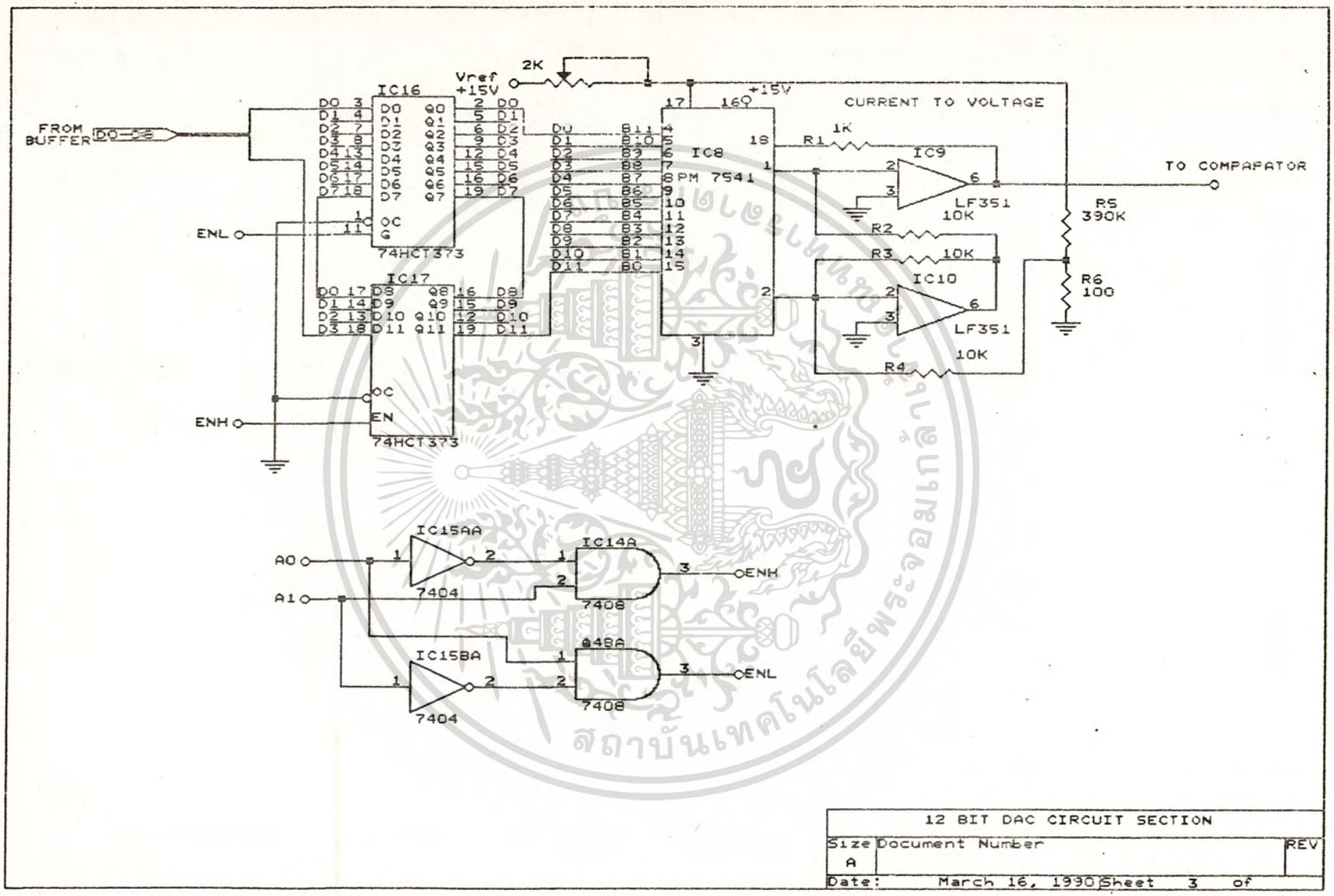
โครงงาน Universal I/O Port นี้ได้เป็นผลสำเร็จเรียบร้อยลงได้ ผู้จัดทำจึงขอขอบพระคุณท่านอาจารย์ที่ปรึกษาโครงงาน คือ ดร.ไพศาล นาคพิพัฒน์ รวมทั้งอาจารย์ในภาควิชาเทคนิคอุตสาหกรรมที่ได้ให้ความรู้ทางด้านวิชาการ และเพื่อนๆ ที่ได้ให้การสนับสนุนมาด้วยดี

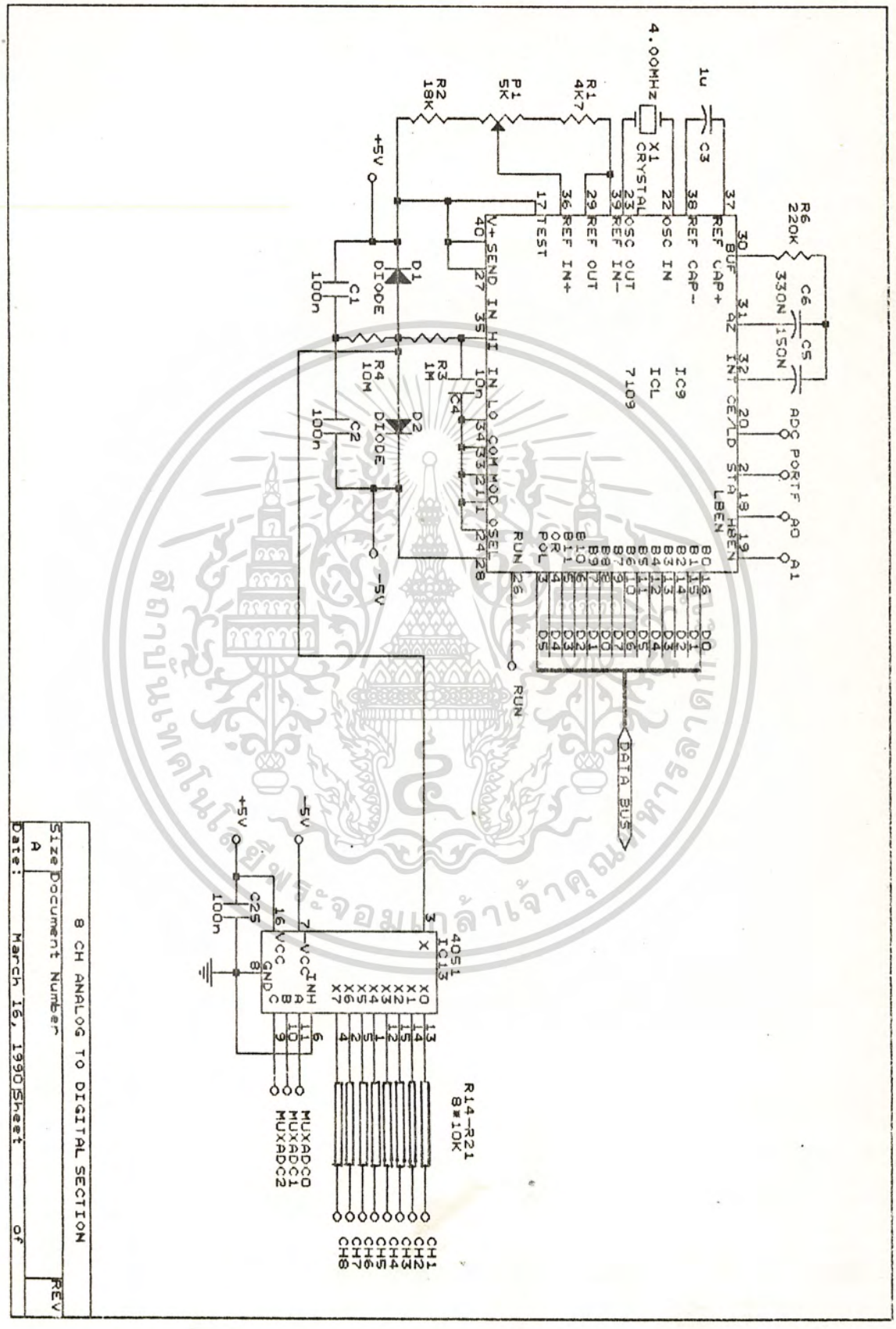
ภาคผนวก ก
Circuit

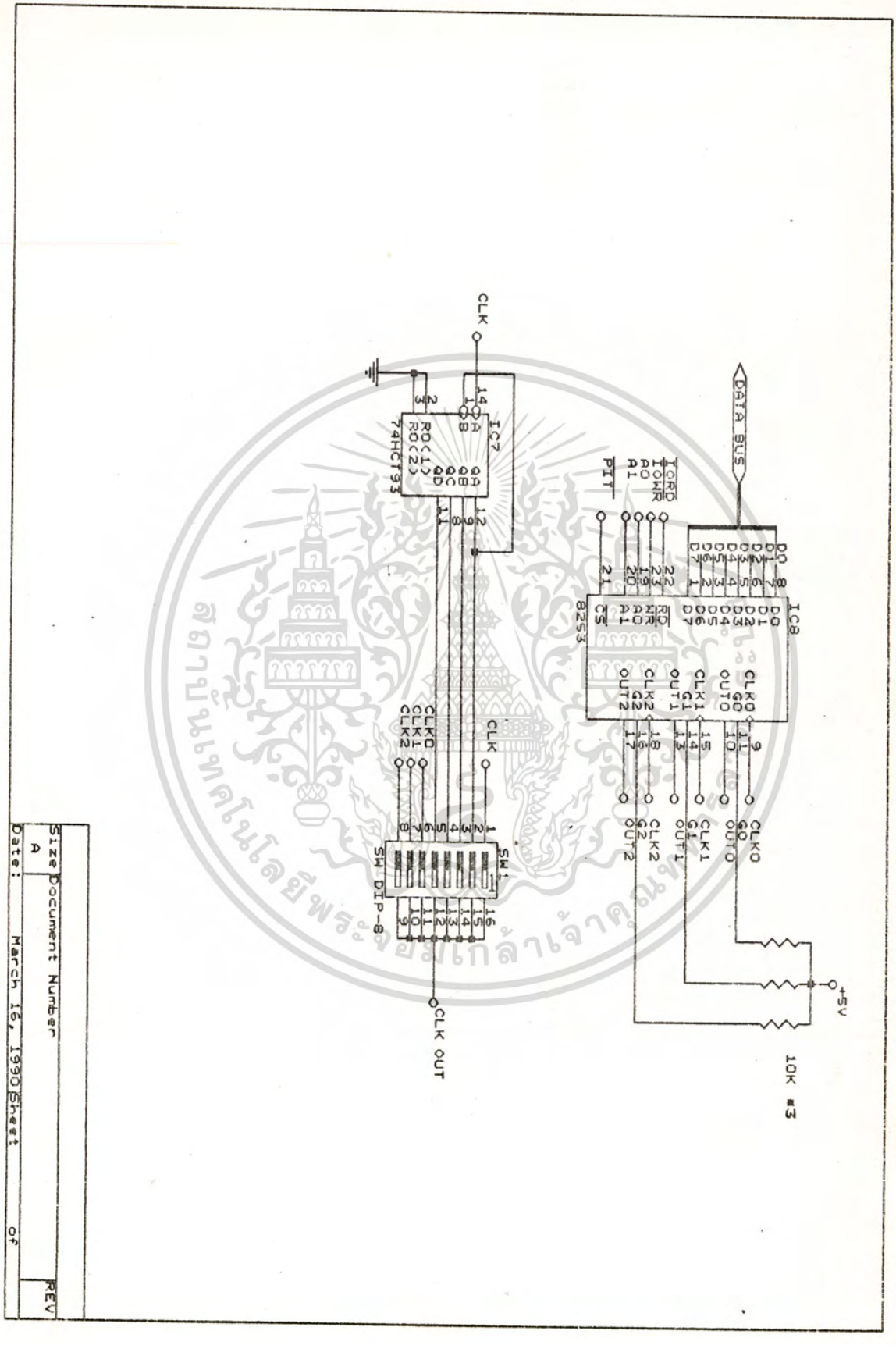


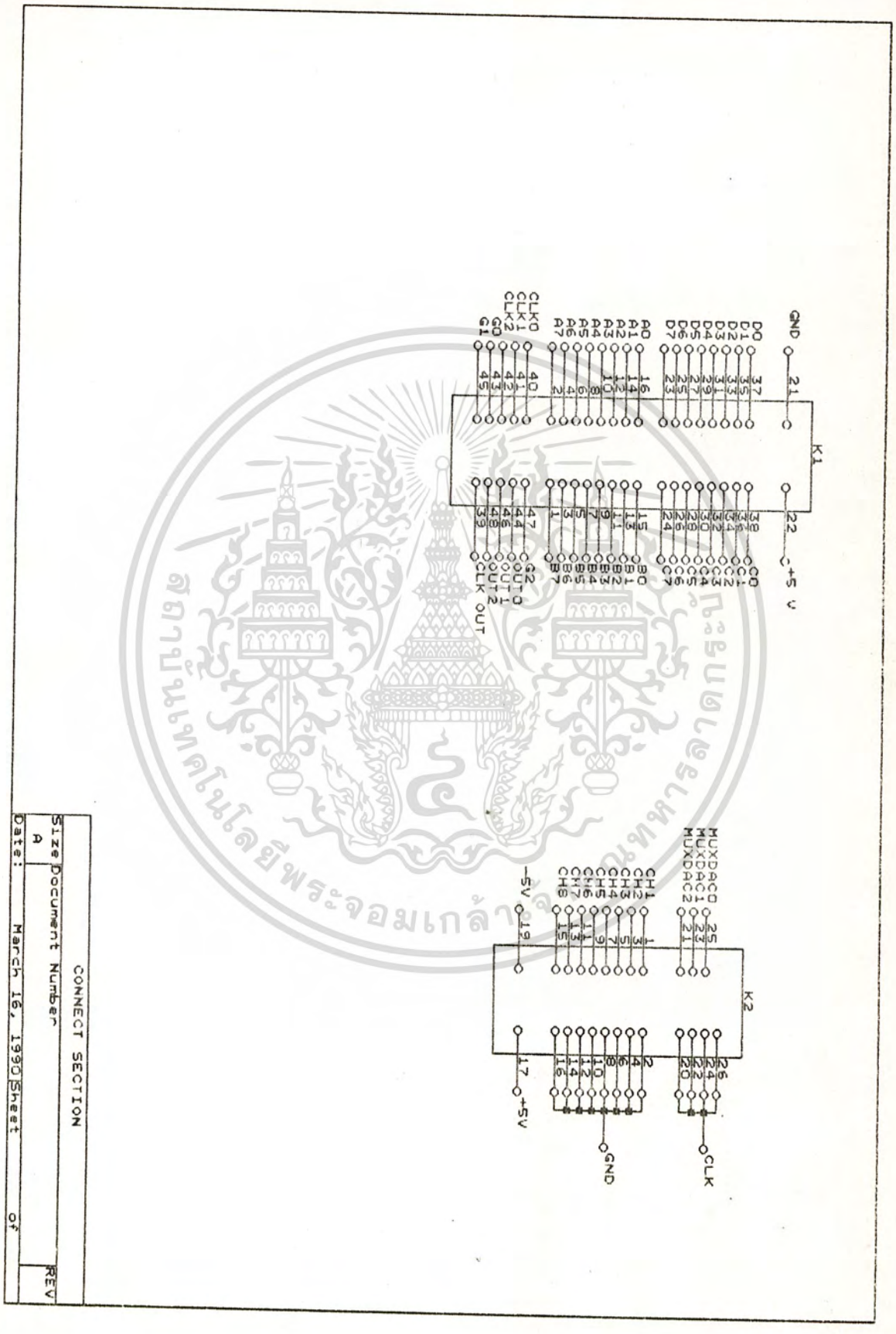


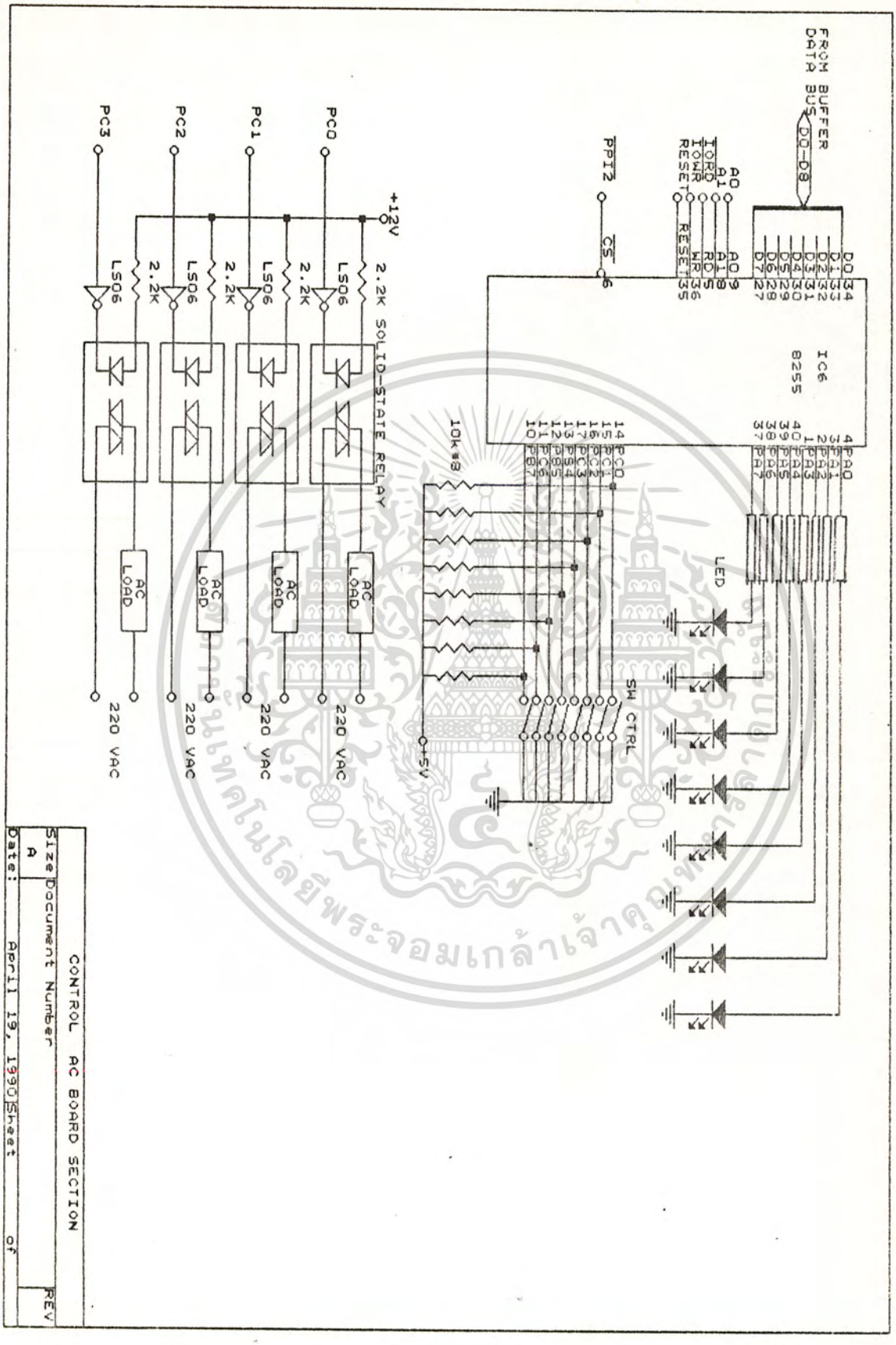


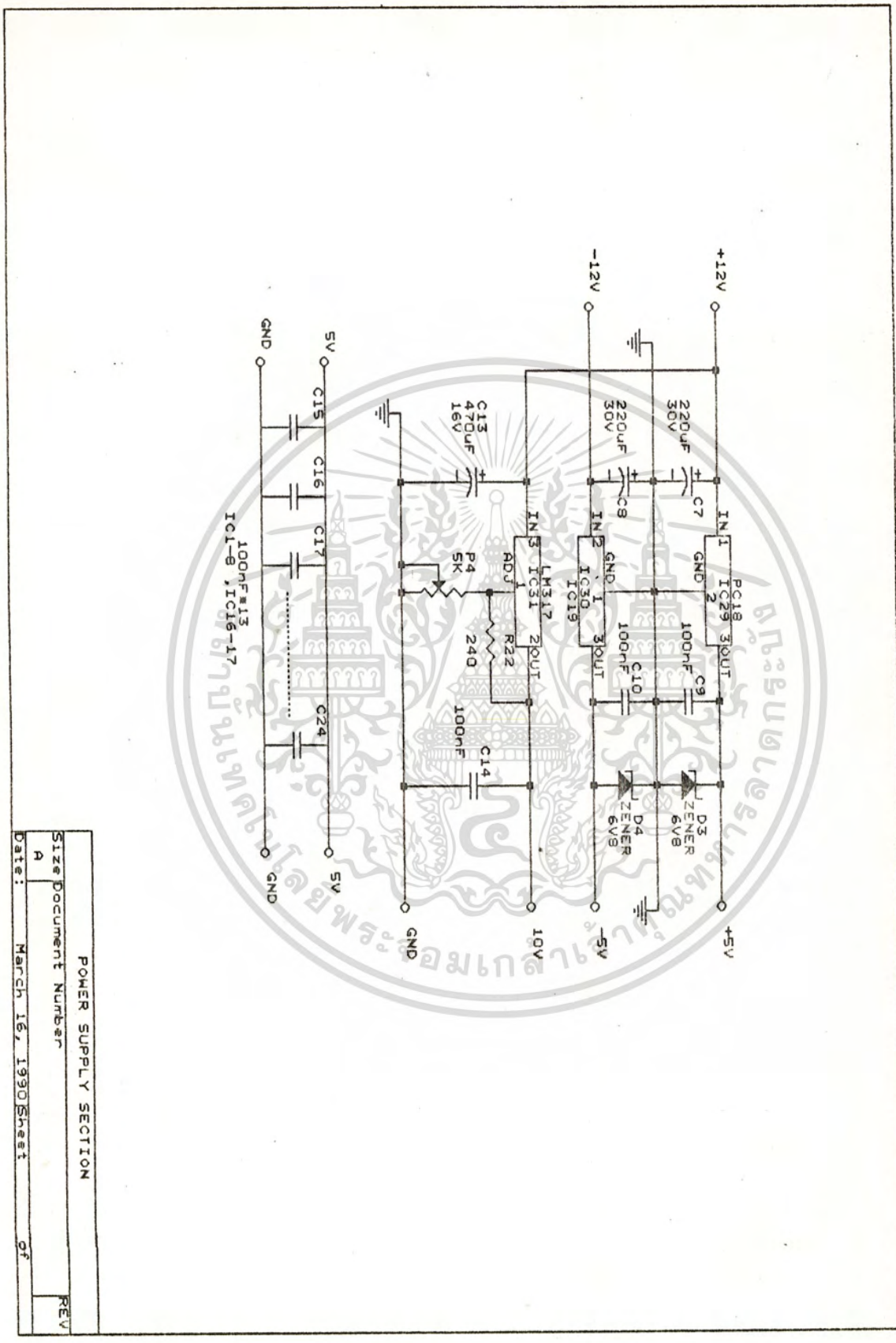

FROM BUFFER D0-D11
IC16
74HCT373
IC17
74HCT373
ENL
ENH
Vref +15V
2K
+15V
IC8
PM 7541
CURRENT TO VOLTAGE
R1 1K
IC9
LF351
10K
R2
R3 10K
IC10
LF351
10K
R4
R5 390K
R6 100
TO COMPAPATOR
A0
A1
IC15AA
7404
IC14A
7408
ENH
IC15BA
7404
IC48A
7408
ENL
12 BIT DAC CIRCUIT SECTION
Size A
Document Number
REV
Date: March 16, 1990 Sheet 3 of











ภาคผนวก ข

Program

```
program demo;    { IN-OUT 8255 }
uses    crt,screen,win,main;
const   msg1 :string[6] = 'ON';
        msg2 :string[6] = 'OFF' ;

var     datastatus : array[1..2,1..8]of byte;
        ch         : char;

procedure setIC;
begin
     port[c1] := $80;
     port[c2] := $9B;
end;

procedure clear;
var i,j : byte;
begin
    for i := 1 to 2 do
        for j := 1 to 8 do
            datastatus[i,j] := 0;
end;

procedure Transform(i : byte);
var DataX : integer;
    j     : byte;
    x     : byte;
    y     : real;
begin
     datax := 0;
     case i of
       1 : dataX := dataa;
       2 : DataX := Datab;
     end;
     for x := 7 downto 1 do
     begin
```

```
        j := x+1;
        if datax <= 0 then
           datastatus[i,j] := 0
        else
        begin
        y := datax mod 2;
        datax := datax div 2;
        if y = 1.0 then
        DataStatus[i,j] := 1
        else
        DataStatus[i,j] := 0;
        end;
      end;
        datax:= datax-1;
        if datax=0 then
         datastatus[i,1] := 1
        else
         datastatus[i,1] := 0;
end;

procedure  int;

begin
      DataC    := port[C];
      DataD    := port[D];
      DataA    := not(DAtaC);
      DAtaB    := not(DataD);
end;

procedure out;
begin

      port[A] := DataA;
      port[B] := DataB;
end;
```

```
procedure  line1(X,Y : byte);
var
     i,j : byte;
begin
     i := 0 ;
     gotoxy(x,y); write(#218);
     for i := 1 to 63 do
       write(#196);
     write(#191);
     gotoxy(x,y+1); write(#179);
     gotoxy(x+64,y+1); write(#179);
     gotoxy(x-8,y+2); write(#218);
end;
procedure  line2;
var i,j : byte;
begin
    for i := 1 to 7 do
     write(#196);
     write(#197);
     for j := 1 to 7 do
     begin
       for i := 1 to 7 do
       write(#196);
       write(#194);
     end;
     for i := 1to 7 do
     write(#196);
     write(#180)
end;
procedure line3(x,y : byte);
var  i,j : byte;
begin
     gotoxy(x-8,y+3); write(#179);
     gotoxy(x,y+3);   write(#179);
```

```
      for i := 1 to 8 do
      begin
      gotoxy(x+(8*i),y+3); write(#179);
      end;
end;

procedure line4(x,y : byte);
var i,j : byte;
begin
      gotoxy(x-8,y+4); write(#195);
      for j := 1 to 8 do
      begin
        for i := 1to 7 do
        write(#196);
        write(#197);
      end;
      for i := 1 to 7 do
      write(#196);
      write(#180);
end;

procedure line5(x,y : byte);
var i,j : byte;
begin
      gotoxy(x-8,y+6); write(#192);
      for j := 1 to 8 do
      begin
            for i := 1 to 7 do
            write(#196);
            write(#193);
      end;
      for i := 1 to 7 do
      write(#196);
      write(#217);
end;
```

```
procedure  pattern;
begin
      line1(12,6);
      line2;
      line3(12,6);
      line4(12,6);
      line3(12,8);
      line5(12,6);
      line1(12,15);
      line2;
      line3(12,15);
      line4(12,15);
      line3(12,17);
      line5(12,15);
end;

procedure  holdon(x,y,i : byte);
begin
      window((x+(8*i)-5),y,(x+(8*i)-2),y);
      clrscr;
      setattr(revblinkhigh);
      write(msg1);
      windowclose;
      setattr(highdisplay);
end;

procedure  holdoff(x,y,i : byte);
begin
      window((x+(8*i)-5),y,(x+(8*i)-2),y);
      clrscr;
      setattr(highdisplay);
      write(msg2);
      windowclose;
end;
```

```
procedure  Show;
var i,j : byte;
begin
     gotoxy(20,3);
     setattr(highdisplay);
     write('THIS PROGRAM IS DEMONSTRATE OF IC8255');
     gotoxy(20,4);
     for i := 1 to 37 do
     write(#178);
     setattr(lowdisplay);
     gotoxy(41,7); write('GROUP A');
     gotoxy(5,9);  write('SW. No.');
     for i := 1 to 8 do
     begin
          gotoxy(8+(8*i),9);
          write(i);
     end;
     gotoxy(5,11); write('STATUS');
     gotoxy(41,16); write('GROUP B');
     gotoxy(5,18);  write('SW. No.');
     for i := 1 to 8 do
     begin
          gotoxy(8+(8*i),18);
          write(i);
     end;
     gotoxy(5,20); write('STATUS');
end;

procedure  result;
var i : byte;
begin
     for i := 1 to 8 do
     begin
          gotoxy(6+(8*i),11);
```

```
            if datastatus[1,i] = 1 then
              holdon(12,11,i)
            else
              holdoff(12,11,i);
      end;
      for i := 1 to 8 do
      begin
            gotoxy(6+(8*i),20);
            if datastatus[2,i] = 1 then
              holdon(12,20,i)
            else
              holdoff(12,20,i);
      end;
end;
procedure  replay;
var   datax,datay : byte;
begin
      clear;
      cursoroff;
      int;
      datax := dataA;
      datay := dataB;
      out;
      transform(1);
      transform(2);
      pattern;
      show;
      result;
      window(14,23,50,23);
      write('PRESS ANY KEY TO QUIT');
      windowclose;
      repeat
         cursoroff;
         int;
         if (dataA <> dataX) then
```

```
        begin
             datax := dataA;
             out;
             transform(1);
             clrscr;
             pattern;
             result;
             window(14,23,50,23);
             write('PRESS ANY KEY TO QUIT');
             windowclose;
        end;
        if (dataB <> datay) then
        begin
          datay := dataB;
          out;
          transform(2);
          result;
          window(14,23,50,23);
          write('PRESS ANY KEY TO QUIT');
          windowclose;
        end;

        delay(100);
    until keypressed;
end;


begin
     clrscr;
     setaddress;
     setIC;
     replay;
end.
```

```
program demo1;    { ON-OFF 8255 }
uses    crt,win,keyboard,screen,main;
const   msg1 : string[6] = 'ON';
        msg2 : string[6] = 'OFF';

var     ch      : char;
        group   : integer;
        switch  : integer;
        data    : array[1..2] of byte;
        state   : array[1..2,1..8] of integer;
        code    : integer;

procedure setIC;
begin
     port[c1] := $80;
end;

procedure clear;
var i,j : byte;
begin
    for i := 1 to 2 do
      begin
        data[i]  := 0;
        for j := 1 to 8 do
         state[i,j]  := 0;
      end;
end;

procedure  line1(X,Y : byte);
var
     i,j : byte;
begin
     i := 0 ;
     gotoxy(x,y); write(#218);
     for i := 1 to 63 do
```

```
        write(#196);
      write(#191);
      gotoxy(x,y+1); write(#179);
      gotoxy(x+64,y+1); write(#179);
      gotoxy(x-8,y+2); write(#218);
end;
procedure  line2;
var i,j : byte;
begin
    for i := 1 to 7 do
      write(#196);
      write(#197);
      for j := 1 to 7 do
      begin
        for i := 1 to 7 do
      write(#196);
       write(#194);
      end;
      for i := 1to 7 do
      write(#196);
      write(#180)
end;
procedure line3(x,y : byte);
var  i,j : byte;
begin
      gotoxy(x-8,y+3); write(#179);
      gotoxy(x,y+3);   write(#179);
      for i := 1 to 8 do
      begin
      gotoxy(x+(8*i),y+3); write(#179);
      end;
end;

procedure line4(x,y : byte);
var i,j : byte;
```

```
begin
      gotoxy(x-8,y+4); write(#195);
      for j := 1 to 8 do
      begin
        for i := 1to 7 do
        write(#196);
        write(#197);
      end;
      for i := 1 to 7 do
      write(#196);
      write(#180);
end;

procedure line5(x,y : byte);
var i,j : byte;
begin
      gotoxy(x-8,y+6); write(#192);
      for j := 1 to 8 do
      begin
          for i := 1 to 7 do
          write(#196);
          write(#193);
      end;
      for i := 1 to 7 do
      write(#196);
      write(#217);
end;

procedure  pattern;
begin
      line1(12,6);
      line2;
      line3(12,6);
      line4(12,6);
      line3(12,8);
```

```
        line5(12,6);
        line1(12,15);
        line2;
        line3(12,15);
        line4(12,15);
        line3(12,17);
        line5(12,15);
end;

procedure  holdon(x,y,i : byte);
begin
        window((x+(8*i)-5),y,(x+(8*i)-2),y);
        clrscr;
        setattr(revblinkhigh);
        write(msg1);
        windowclose;
        setattr(highdisplay);
end;

procedure  holdoff(x,y,i : byte);
begin
        window((x+(8*i)-5),y,(x+(8*i)-2),y);
        clrscr;
        setattr(highdisplay);
        write(msg2);
        windowclose;
end;

procedure  Show;
var i,j : byte;
begin
        gotoxy(20,3);
        setattr(highdisplay);
        write('  THIS PROGRAM IS ON-OFF EQUIPMENT');
        gotoxy(20,4);
```

```
        for i := 1 to 36 do
        write(#178);
        setattr(lowdisplay);
        gotoxy(41,7); write('GROUP 1');
        gotoxy(5,9);  write('SW. No.');
        for i := 1 to 8 do
        begin
             gotoxy(8+(8*i),9);
             write(i);
        end;
        gotoxy(5,11); write('STATUS');
        gotoxy(41,16); write('GROUP 2');
        gotoxy(5,18);  write('SW. No.');
        for i := 1 to 8 do
        begin
             gotoxy(8+(8*i),18);
             write(i);
        end;
        gotoxy(5,20); write('STATUS');

end;

procedure  result;
var i : byte;
begin
        for i := 1 to 8 do
        begin
             gotoxy(6+(8*i),11);
             if state[1,i] = 1 then
               holdon(12,11,i)
             else
               holdoff(12,11,i);
        end;
        for i := 1 to 8 do
        begin
```

```
            gotoxy(6+(8*i),20);
            if state[2,i] = 1 then
              holdon(12,20,i)
            else
              holdoff(12,20,i);
      end;
end;

procedure  replay;
begin
      if state[group,switch] = 1 then
       begin
         case group of
          1 : holdon(12,11,switch);
          2 : holdon(12,20,switch);
         end;
       end
      else
       begin
         case group of
          1 : holdoff(12,11,switch);
          2 : holdoff(12,20,switch);
         end;
       end;
end;

procedure  group1;
begin
      if state[1,switch] = 1 then
      begin
          case switch of
           1 : data[1] := (data[1] or $01);
           2 : data[1] := (data[1] or $02);
           3 : data[1] := (data[1] or $04);
           4 : data[1] := (data[1] or $08);
```

```
            5 : data[1] := (data[1] or $10);
            6 : data[1] := (data[1] or $20);
            7 : data[1] := (data[1] or $40);
            8 : data[1] := (data[1] or $80);
          end;
      end
      else
      begin
          case switch of
            1 : data[1] := (data[1] and $FE);
            2 : data[1] := (data[1] and $FD);
            3 : data[1] := (data[1] and $FB);
            4 : data[1] := (data[1] and $F7);
            5 : data[1] := (data[1] and $EF);
            6 : data[1] := (data[1] and $DF);
            7 : data[1] := (data[1] and $BF);
            8 : data[1] := (data[1] and $7F);
          end;
      end;
      port[A] := data[1];
end;

procedure  group2;
begin
      if state[2,switch] = 1 then
      begin
          case switch of
            1 : data[2] := (data[2] or $01);
            2 : data[2] := (data[2] or $02);
            3 : data[2] := (data[2] or $04);
            4 : data[2] := (data[2] or $08);
            5 : data[2] := (data[2] or $10);
            6 : data[2] := (data[2] or $20);
            7 : data[2] := (data[2] or $40);
            8 : data[2] := (data[2] or $80);
```

```
            end;
        end
        else
        begin
            case switch of
             1 : data[2] := (data[2] and $FE);
             2 : data[2] := (data[2] and $FD);
             3 : data[2] := (data[2] and $FB);
             4 : data[2] := (data[2] and $F7);
             5 : data[2] := (data[2] and $EF);
             6 : data[2] := (data[2] and $DF);
             7 : data[2] := (data[2] and $BF);
             8 : data[2] := (data[2] and $7F);
            end;
        end;
        port[B] := data[2];
end;

procedure  select;
begin
        case group of
         1 : group1;
         2 : group2;
        end;
end;

procedure  enterdata;
begin
        window(20,23,60,25);
        cursoroff;
        clrscr;
        repeat
        delline;
        gotoxy(1,wherey);
        write('GROUP : ');
```

```
        readintnum(group,code,1);
        until (group in[1..2]) and (code =0);
        writeln;
        repeat
        delline;
        gotoxy(1,wherey);
        write('SWITCH No. : ');
        readintnum(switch,code,1);
        until (switch in [1..8]) and (code =0);
        writeln;
        repeat
        delline;
        gotoxy(1,wherey);
        write('STATUS (0:OFF ¦ 1:ON ¦ 2:EXIT) : ');
        readintnum(state[group,switch],code,1);
        until(state[group,switch] in[0..2]) and (code=0);
        windowclose;
end;

begin { main }
        setaddress;
        setic;
        clear;
        clrscr;
        pattern;
        show;
        result;
        repeat
        enterdata;
        replay;
        select;
        until state[group,switch] = 2;
end.
```

```
        clrscr;
        gotoxy(0,0);
        writeln('                    Frequency  Sweep Generator');
        writeln('               Generate BETWEEN  100Hz - 1KHz');
        Gotoxy(1,5) ;
        writeln('                      Press Any Key Will stop');
        port[$307] := $80;
        port[$305] := $20;
        OUT_DAC;
        RUN_OK;
        write('DATA..!');
        ch := readkey;
        if (ch = 'a') or (ch = 'A') then  sweep2
         else
         Show;
end;
procedure Sweep3;
var i : integer;
begin
        clrscr;
        gotoxy(0,0);
        writeln('                    Frequency  Sweep Generator');
        writeln('               Generate BETWEEN  1KHz - 10KHz');
        Gotoxy(1,5) ;
        writeln('                      Press Any Key Will stop');
        port[$307] := $80;
        port[$305] := $40;
        OUT_DAC;
        RUN_OK;
        write('DATA..!');
        ch := readkey;
        if (ch = 'a') or (ch = 'A') then  sweep3
         else
         Show;
end;
```

```
procedure Sweep4;
var i : integer;
begin
     clrscr;
     gotoxy(0,0);
     writeln('                    Frequency  Sweep Generator');
     writeln('            Generate BETWEEN  10KHz - 100KHz');
     Gotoxy(1,5) ;
     writeln('                          Press Any Key Will stop');
     port[$307] := $80;
     port[$305] := $80;
     OUT_DAC;
     RUN_OK;
     write('DATA..!d');
     ch :=readkey;
     if (ch = 'a') or (ch = 'A') then  sweep4
      else
      Show;
end;

{MAIN}
begin
   repeat
    SHOW;
    write('Choose choice NO...'); readln(choice);
     case choice of
       '1' : Sweep1;
       '2' : Sweep2;
       '3' : Sweep3;
       '4' : Sweep4;
      end;
   until choice = '5';
    port[$305] := $00;
end.
```

```
program  Voltage_Generator;  { Demo3 }
uses crt,dos;
var
  ch,choice : char;
  n  : integer;
procedure RUN_OK;
begin
    gotoxy(1,10);
    writeln(' !PROCESS RUN OK!');
    writeln;
    writeln('If you want run against');
    writeln('PLS you enter  (A)  and');
    writeln;
    writeln('If you want return to main MENU');
    writeln('PLS you enter (ANY KEY)!');
end;
procedure  Show;
begin
   clrscr;
   writeln('                                    THIS');
   writeln('                                  PROCESS');
   writeln('                                    IS');
   writeln('                              Voltage  Regulator');
   writeln;writeln;
   writeln('          MENU');writeln;writeln;
   writeln('     1. Range   0V  to  +10V  Plus (+) Supply');
   writeln('     2. Range  -10V to   0V   Minus(-) Supply');
   writeln('     3. EXIT                 ');
   writeln;writeln;writeln;
   writeln('PLS you enter DATA Range of  Voltage Regulator');

end;

procedure  OUT_DAC_Plus;
var i : integer;ch : char;
```

```
program IC8255;  { demo4 }
uses crt,main;
type
      string8 = string[8];
      data  =  record
                 col     : array[0..8] of byte;
                 row     : array[0..8] of byte;
                 msg1    : array[0..8] of string8;
                 msg2    : array[0..8] of string8;
               end;
      ArrayOfData = array[1..4,1..8]Of byte;
      byterec1    = record
                      lo,hi : byte;
                    end;

const
      bell        = #$07;

var
      portselect : byte;
      DataStatus : ArrayOfData;

function  Power(n:real):real;
begin
     Power := Exp(ln(2)*n)
end;

procedure  ReadData(j:byte);
var
    i             : byte;
    ch            : char;
begin
       i := 0;
       for i := 1 to 8 do
         begin
```

```
            Repeat
              ch := readkey;
              case ch of
               '0' : begin datastatus[j,i] := 0; write(ch); end;
               '1' : begin datastatus[j,i] := 1; write(ch); end;
              else
               write(bell);
              end;
            until (ch = '0')or(ch = '1');
          end;
        writeln;
end;

procedure  cleararray;
var i,j :byte;
begin
     dataa:=0;
     datab:=0;
     datac:=0;
     datad:=0;
     i    :=0;
     j    :=0;
     for i:= 1 to 8 do
         for j:=1 to 4 do
         datastatus[j,i] :=0;
end;

procedure AssignData(j : byte);
var DataX : byte;
    x     : longint;
    i     : byte;
    y     : real;
begin
     Datax := 0;
     x     := 0;
```

```
      y      := 0;
      i      := 0;
      for i := 1 to 8 do
        begin
           y  := (8-i);
           x := trunc(Power(y));
           DataX := DataX + (DataStatus[j,i]*x);
        end;
      case j of
        1 : DataA := Datax;
        2 : DataB := DataX;
        3 : DataC := DataX;
        4 : DataD := DataX;
      end;
end;

procedure Transform(j : byte);
var DataX : integer;
    i     : byte;
    y     : real;
    x     : byte;
begin
     datax := 0;
     case j of
       1 : DataX := DataA;
       2 : DataX := DataB;
       3 : DataX := DataC;
       4 : DataX := DataD;
     end;
     y := 0;
     i := 0;
     x := 0;
     for x := 7 downto 1 do
     begin
          i := x+1;
```

```
        y      := datax mod 2;
        datax := (datax div 2);
        if y=1.0 then
        DataStatus[j,i] := 0
        else
        DataStatus[j,i] := 1;
      end;
        if y = 1.0  then
           datastatus[j,1] := 0
        else
            datastatus[j,1] := 1;
end;

procedure  Bishow(j : byte);
var  i : byte;
begin
      case j of
        1 : write('dataA(BIN) = ');
        2 : write('dataB(BIN) = ');
        3 : write('dataC(BIN) = ');
        4 : write('dataD(BIN) = ');
      end;
      for i := 1 to 8 do
        write(DataStatus[j,i]);
      writeln;
end;

procedure ReadPort1;
begin
      port[$307] := $99;
      dataA := port[$304];
      transform(1);
      bishow(1);
      dataB := port[$306];
      transform(2);
```

```
        bishow(2);
end;

procedure WritePort1;
begin
        write('dataA(BIN) = ');
        ReadData(1);
        AssignData(1);
        write('dataB(BIN) = ');
        ReadData(2);
        AssignData(2);
        port[$307] := $80;
        port[$304] := dataA;
        port[$306] := dataB;
end;

procedure ReadWritePort1;
begin
        port[$307] := $90;
        dataA := port[$304];
        transform(1);
        bishow(1);
        write('dataB(BIN) = ');
        ReadData(2);
        AssignData(2);
        port[$306] := dataB;
end;

procedure WriteReadPort1;
begin
        write('dataA(BIN) = ');
        ReadData(1);
        AssignData(1);
        port[$307] := $89;
        port[$304] := dataA;
```

```
        dataB := port[$306];
        transform(2);
        bishow(2);
end;

procedure ReadPort2;
begin
        port[$30B] := $9B;
        dataC := port[$308];
        transform(3);
        bishow(3);
        dataD := port[$30A];
        transform(4);
        bishow(4);
end;

procedure WritePort2;
begin
        write('dataC(BIN) = ');
        ReadData(3);
        AssignData(3);
        write('dataD(BIN) = ');
        ReadData(4);
        AssignData(4);
        port[$30B] := $82;
        port[$308] := dataC;
        port[$30A] := dataD;
end;

procedure ReadWritePort2;
begin
        port[$30B] := $92;
        dataC := port[$308];
        transform(3);
        bishow(3);
```

```
end;

begin {main}
 clrscr;
 selectshow(1);
 readln;
 selectshow(2);
 readln;
end.
```

```
        ctshow;
        delay (1000);
        writeln;
        write('press  Q  to quit');
        ch := readkey;
        until ch in ['Q','q'];
end.
```

ภาคผนวก ค
DATA IC

EXAR

XR-2206

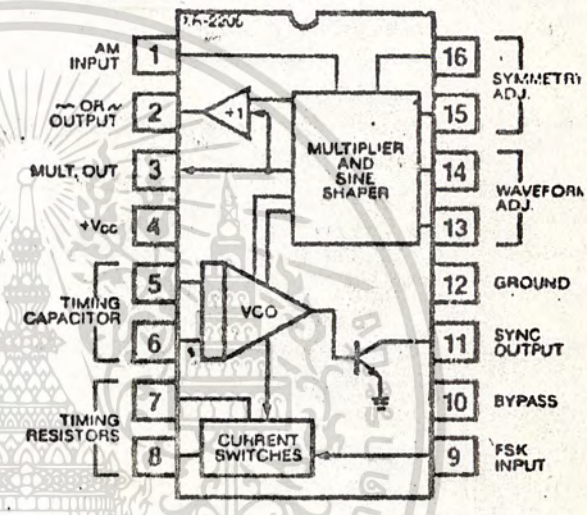
Monolithic Function Generator

GENERAL DESCRIPTION

The XR-2206 is a monolithic function generator integrated circuit capable of producing high quality sine, square, triangle, ramp, and pulse waveforms of high stability and accuracy. The output waveforms can be both amplitude and frequency modulated by an external voltage. Frequency of operation can be selected externally over a range of 0.01 Hz to more than 1 MHz.

The circuit is ideally suited for communications, instrumentation, and function generator applications requiring sinusoidal tone, AM, FM, or FSK generation. It has a typical drift specification of 20 ppm/°C. The oscillator frequency can be linearly swept over a 2000:1 frequency range, with an external control voltage, having a very small affect on distortion.

FUNCTIONAL BLOCK DIAGRAM



FEATURES

Low-Sine Wave Distortion	0.5%, Typical
Excellent Temperature Stability	20 ppm/°C, Typical
Wide Sweep Range	2000:1, Typical
Low-Supply Sensitivity	0.01%V, Typical
Linear Amplitude Modulation	
TTL Compatible FSK Controls	
Wide Supply Range	10V to 26V
Adjustable Duty Cycle	1% to 99%

APPLICATIONS

Waveform Generation
Sweep Generation
AM/FM Generation
V/F Conversion
FSK Generation
Phase-Locked Loops (VCO)

ABSOLUTE MAXIMUM RATINGS

Power Supply	26V
Power Dissipation	750 mW
Derate Above 25°C	5 mW/°C
Total Timing Current	6 mA
Storage Temperature	−65°C to +150°C

ORDERING INFORMATION

Part Number	Package	Operating Temperature
XR-2206M	Ceramic	−55°C to +125°C
XR-2206N	Ceramic	0°C to +70°C
XR-2206P	Plastic	0°C to +70°C
XR-2206CN	Ceramic	0°C to +70°C
XR-2206CP	Plastic	0°C to +70°C

SYSTEM DESCRIPTION

The XR-2206 is comprised of four functional blocks; a voltage-controlled oscillator (VCO), an analog multiplier and sine-shaper; a unity gain buffer amplifier; and a set of current switches.

The VCO actually produces an output frequency proportional to an input current, which is produced by a resistor from the timing terminals to ground. The current switches route one of the timing pins current to the VCO controlled by an FSK input pin, to produce an output frequency. With two timing pins, two discrete output frequencies can be independently produced for FSK Generation Applications.

ELECTRICAL CHARACTERISTICS

Test Conditions: Test Circuit of Figure 1, $V^+ = 12V$, $T_A = 25°$, $C = 0.01\ \mu F$, $R_1 = 100\ k\Omega$, $R_2 = 10\ k\Omega$, $R_3 = 25\ k\Omega$ unless otherwise specified. S_1 open for triangle, closed for sine wave.

PARAMETERS	XR-2206M			XR-2206C			UNITS	CONDITIONS
	MIN	TYP	MAX	MIN	TYP	MAX		
GENERAL CHARACTERISTICS								
Single Supply Voltage	10		26	10		26	V	
Split-Supply Voltage	±5		±13	±5		±13	V	
Supply Current		12	17		14	20	mA	$R_1 \geq 10\ k\Omega$
OSCILLATOR SECTION								
Max. Operating Frequency	0.5	1		0.5	1		MHz	$C = 1000\ pF$, $R_1 = 1\ k\Omega$
Lowest Practical Frequency		0.01			0.01		Hz	$C = 50\ \mu F$, $R_1 = 2\ M\Omega$
Frequency Accuracy		±1	±4		±2		% of f_o	$f_o = 1/R_1C$
Temperature Stability		±10	±50		±20		ppm/°C	$0°C \leq T_A \leq 70°C$,
Supply Sensitivity		0.01	0.1		0.01		%/V	$R_1 = R_2 = 20\ k\Omega$ $V_{LOW} = 10V$, $V_{HIGH} = 20V$,
Sweep Range	1000:1	2000:1			2000:1		$f_H = f_L$	$R_1 = R_2 = 20\ k\Omega$ f_H @ $R_1 = 1\ k\Omega$ f_L @ $R_1 = 2\ M\Omega$
Sweep Linearity								
10:1 Sweep		2			2		%	$f_L = 1\ kHz$, $f_H = 10\ kHz$
1000:1 Sweep		8			8		%	$f_L = 100\ kHz$, $f_H = 100\ kHz$
FM Distortion		0.1			0.1		%	±10% Deviation
Recommended Timing Components								
Timing Capacitor: C	0.001		100	0.001		100	µF	See Figure 4.
Timing Resistors: R_1 & R_2	1		2000	1		2000	kΩ	
Triangle Sine Wave Output								See Note 1, Figure 2.
Triangle Amplitude		160			160		mV/kΩ	Figure 1, S_1 Open
Sine Wave Amplitude	40	60	80		60		mV/kΩ	Figure 1, S_1 Closed
Max. Output Swing		6			6		V p-p	
Output Impedance		600			600		Ω	
Triangle Linearity		1			1		%	
Amplitude Stability		0.5			0.5		dB	For 1000:1 Sweep
Sine Wave Amplitude Stability		4800			4800		ppm/°C	See Note 2.
Sine Wave Distortion								
Without Adjustment		2.5			2.5		%	$R_1 = 30\ k\Omega$
With Adjustment		0.4	1.0		0.5	1.5	%	See Figures 6 and 7.
Amplitude Modulation								
Input Impedance	50	100		50	100		kΩ	
Modulation Range		100			100		%	
Carrier Suppression		55			55		dB	
Linearity		2			2		%	For 95% modulation
Square-Wave Output								
Amplitude		12			12		V p-p	Measured at Pin 11.
Rise Time		250.			250		nsec	$C_L = 10\ pF$
Fall Time		50			50		nsec	$C_L = 10\ pF$
Saturation Voltage		0.2	0.4		0.2	0.6	V	$I_L = 2\ mA$
Leakage Current		0.1	20		0.1	100	µA	$V_{11} = 26V$
FSK Keying Level (Pin 9)	0.8	1.4	2.4	0.8	1.4	2.4	V	See section on circuit controls
Reference Bypass Voltage	2.9	3.1	3.3	2.5	3	3.5	V	Measured at Pin 10.

Note 1: Output amplitude is directly proportional to the resistance, R_3, on Pin 3. See Figure 2.
Note 2: For maximum amplitude stability, R_3 should be a positive temperature coefficient resistor.

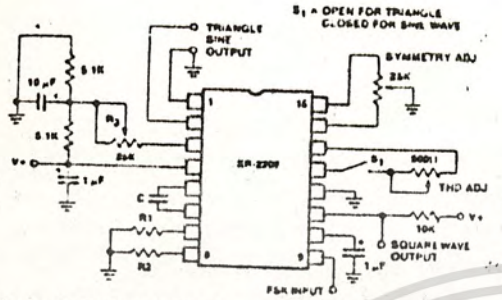


Figure 1. Basic Test Circuit.

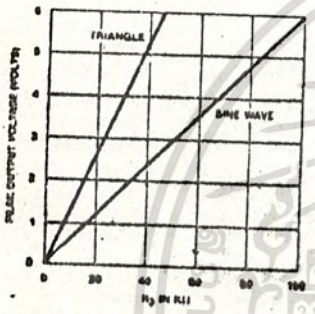


Figure 2. Output Amplitude as a Function of the Resistor, R_3, at Pin 3.

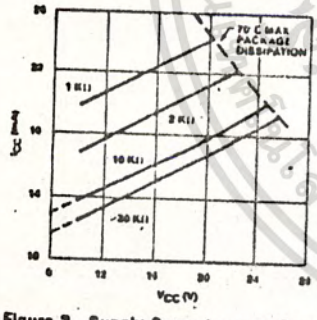


Figure 3. Supply Current versus Supply Voltage, Timing, R.

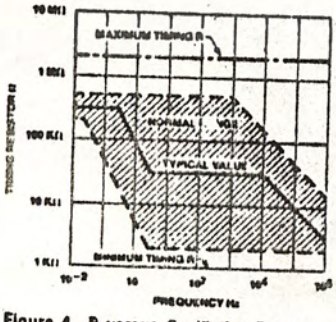


Figure 4. R versus Oscillation Frequency.

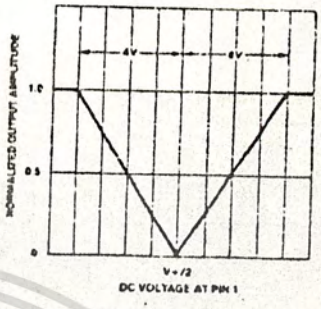


Figure 5. Normalized Output Amplitude versus DC Bias at AM Input (Pin 1).

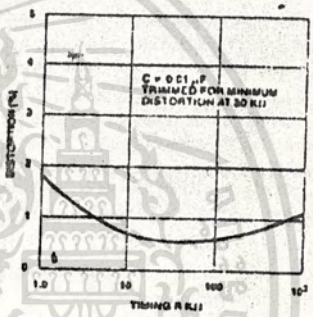


Figure 6. Trimmed Distortion versus Timing Resistor.

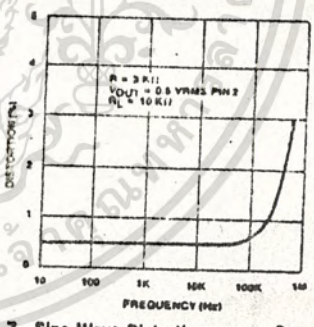


Figure 7. Sine Wave Distortion versus Operating Frequency with Timing Capacitors Varied.

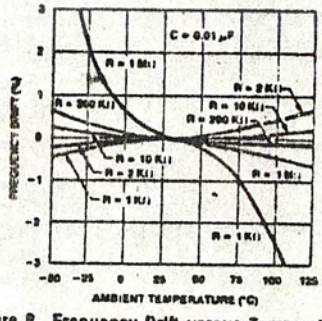


Figure 8. Frequency Drift versus Temperature.

PMI

PM-7541

CMOS 12-BIT MONOLITHIC MULTIPLYING D/A CONVERTER

Precision Monolithics Inc.

FEATURES

- Full Four-Quadrant Multiplication
- 12-Bit Endpoint Linearity (±1/2 LSB)
- Pretrimmed Gain
- TTL/CMOS Compatible
- Low Power Consumption
- Low Feedthrough Error
- Direct Replacement for AD7521 and AD7541
- Superior Power Supply Rejection from +5V to +15V
- Low Gain and Linearity Tempcos (TYP 2ppm of FSR/°C)
- Latch-Up Resistant

APPLICATIONS

- Digital/Synchro Conversion
- Programmable Amplifiers
- Ratiometric A/D Conversion
- Function Generator
- CRT Graphics Generator
- Digitally-Controlled Attenuator
- Digitally-Controlled Power Supplies
- Digital Filters

CROSS REFERENCE

PMI	ADI	TEMPERATURE RANGE
PM7541AX	AD7541D	MILITARY
PM7541BX	AD7541SD	
PM7541EX	AD7541BD	INDUSTRIAL
PM7541FX	AD7541AD	
PM7541GP	AD7541KN	COMMERCIAL
PM7541HP	AD7541JN	

GENERAL DESCRIPTION

The PMI PM-7541 is a 12-bit, 4-quadrant multiplying digital-to-analog converter. It is manufactured using an advanced oxide-isolated, silicon-gate, monolithic CMOS technology.

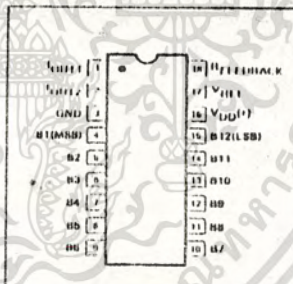
Laser-trimmed thin-film resistors on CMOS circuitry provide true 12-bit linearity and excellent absolute accuracy. The low power dissipation, together with NMOS temperature-compensating switches, assures the performance over the full temperature range. It is a pin-compatible replacement for Analog Devices AD7521 and AD7541 with equal or better performance.

ORDERING INFORMATION†

	PACKAGE: 18-PIN**		
NONLINEARITY	MILITARY* TEMPERATURE -55°C TO +125°C	INDUSTRIAL TEMPERATURE -25°C TO +85°C	COMMERCIAL TEMPERATURE 0°C TO +70°C
1 LSB	PM7541BX	PM7541FX	PM7541HP
1/2 LSB	PM7541AX	PM7541EX	PM7541GP

*For devices processed in total compliance to MIL-STD-883, add /883 after part number. Consult factory for 883 data sheet.
(also available in Side Braze—XB)
†All commercial and industrial temperature range parts are available with burn-in. For ordering information see 1986 Data Book, Section 2.
**Package Designation:
Suffix X: Hermetic DIP (XB - Side Braze)
Suffix P: Epoxy DIP

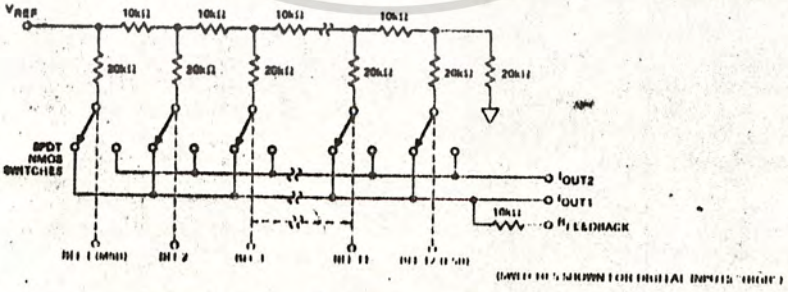
PIN CONNECTIONS



18-PIN EPOXY DIP (P-Suffix)

18-PIN HERMETIC DIP (X-Suffix)

FUNCTIONAL DIAGRAM



ABSOLUTE MAXIMUM RATINGS

(T_A = +25°C, unless otherwise noted.)

V_{DD} (to GND) ..... +17V
V_{REF} (to GND) ..... ±25V
Digital Input Voltage Range ..... V_{DD} to GND
Output Voltage (Pin 1, Pin 2) ..... 0.3V to V_{DD}
Power Dissipation (Package) 450mW
  Derate Above +75°C ..... 6mW/°C
Operating Temperature Range
  AX/BX Versions ..... -55°C to +125°C
  EX/FX Versions ..... -25°C to +85°C
  GP/HP Versions ..... 0°C to +70°C
Dice Junction Temperature ..... +150°C
Storage Temperature ..... -65°C to +150°C
Lead Temperature (Soldering, 60 sec) ..... 300°C

CAUTION:

1. Do not apply voltages higher than V_{DD} or less than GND potential on any terminal except V_{REF} (Pin 17) and R_{FB} (Pin 18).
2. The digital control inputs are zener protected; however, permanent damage may occur on unprotected units from high-energy electrostatic fields. Keep units in conductive foam at all times until ready to use.
3. Use proper anti-static handling procedures.
4. Absolute Maximum Ratings apply to both packaged devices and DICE. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device.

ELECTRICAL CHARACTERISTICS at V_{DD} = +15V, V_{REF} = +10V, AGND = DGND = 0V, $V_{OUT\,1}$ = $V_{OUT\,2}$ = 0V; and T_A = -55°C to +125°C, apply for PM-7541AX/BX; T_A = -25°C to +85°C apply for PM-7541EX/FX; and T_A = 0°C to +70°C apply for PM-7541GP/HP, unless otherwise noted.

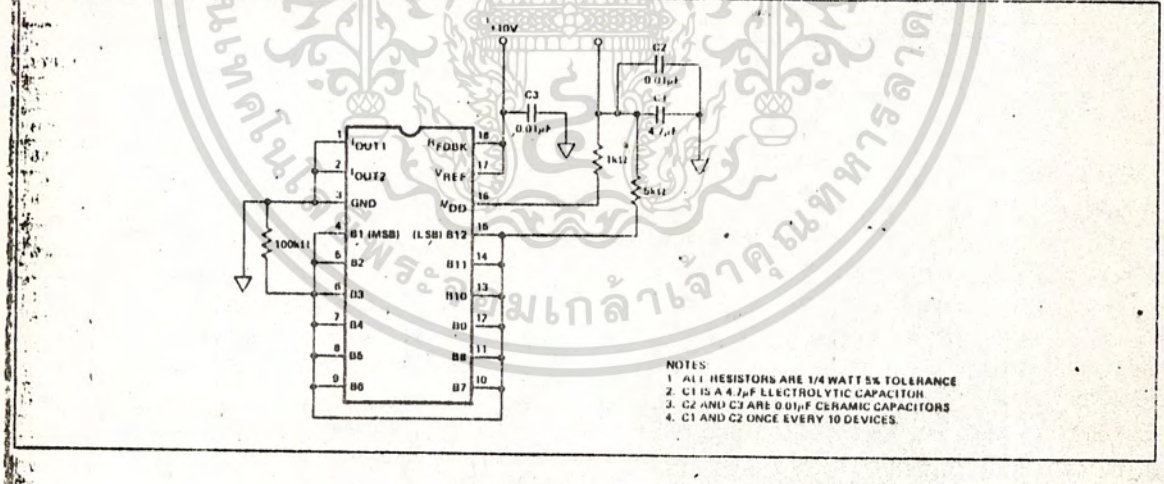
			PM-7541A/E/G			PM-7541B/F/H			
PARAMETER	**SYMBOL**	**CONDITIONS**	**MIN**	**TYP**	**MAX**	**MIN**	**TYP**	**MAX**	**UNITS**
STATIC ACCURACY									
Resolution	N		12	—	—	12	—	—	Bits
Nonlinearity (Notes 1, 2)	INL		—	—	±1/2	—	—	±1	LSB
Gain Error (Notes 3, 4)	G_{ER}	T_A = +25°C T_A = Full Temp. Range	— —	— —	±12.5 ±16.7	— —	— —	±12.5 ±16.7	LSB
Power Supply Rejection $\Delta Gain/\Delta V_{DD}$	PSRR	V_{REF} = +14.7V to +15.3V T_A = +25°C T_A = Full Temp. Range	— —	— —	±0.01 ±0.02	— —	— —	±0.01 ±0.02	%/%
Output Leakage Current (I_{OUT1}) (Notes 5, 6)	I_{LKG}	T_A = +25°C T_A = Full Temp. Range	— —	— —	±50 ±200	— —	— —	±50 ±200	nA
DYNAMIC PERFORMANCE									
Output Current Settling Time (Note 7)	t_S	To 1/2 LSB of FSR	—	—	1.0	—	—	1.0	µs
Feedthrough Error (Note 7)		V_{REF} = 20V_{p-p} @ f = 10kHz All digital inputs low	—	—	1.0	—	—	1.0	mV_{p-p}
REFERENCE INPUT									
Input Resistance (Note 8)	R_{REF}		5	—	20	5	—	20	kΩ
DIGITAL INPUTS									
Digital Input High	V_{IH}		2.4	—	—	2.4	—	—	V
Digital Input Low	V_{IL}		—	—	0.8	—	—	0.8	V
Input Leakage Current	I_{IL}	V_{IN} = 0 to 15V	—	—	±1	—	—	±1	µA
Input Capacitance (Note 7)	C_{IN}		—	—	10	—	—	10	pF
Input Coding		(Tables 1, 2)	Binary or Offset			Binary or Offset			

ELECTRICAL CHARACTERISTICS at V_{DD} = +15V, V_{REF} = +10V, AGND = DGND = 0V, $V_{OUT\,1}$ = $V_{OUT\,2}$ = 0V; and T_A = -55°C to +125°C apply for PM-7541AX/BX; T_A = -25°C to +85°C apply for PM-7541EX/FX; and T_A = 0°C to +70°C apply for PM-7541GP/HF, unless otherwise noted. (Continued)

PARAMETER	SYMBOL	CONDITIONS	PM-7541A/E/G MIN	PM-7541A/E/G TYP	PM-7541A/E/G MAX	PM-7541B/F/H MIN	PM-7541B/F/H TYP	PM-7541B/F/H MAX	UNITS
ANALOG OUTPUTS									
Output Capacitance (Note 7)	$C_{OUT\,1}$	Digital Inputs = V_{IH}	—	189	220	—	189	220	pF
	$C_{OUT\,2}$		—	36	60	—	36	60	
Output Capacitance (Note 7)	$C_{OUT\,1}$	Digital Inputs = V_{IL}	—	95	120	—	95	120	pF
	$C_{OUT\,2}$		—	134	165	—	134	165	
POWER SUPPLY									
V_{DD} Range	V_{DD}	Accuracy is not guaranteed over this range.	+5		+16	+5	—	+16	V
Supply Current	I_{DD}	Digital Inputs = V_{IH} or V_{IL}	—	—	2	—	—	2	mA

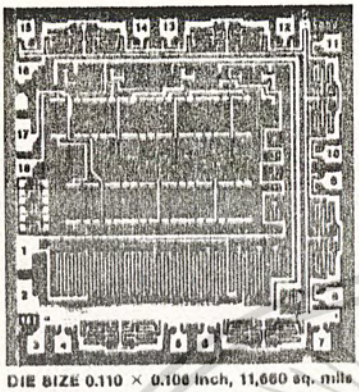
NOTES:
1. A/E/G versions are monotonic to 12-bits.
2. B/F/H versions are monotonic to 11-bits.
3. Using internal feedback resistor.
4. Maximum gain change from +25°C to T_{MAX} or T_{MIN} is ±4.2 LSB maximum.
5. Digital Inputs = V_{IL}.
6. Specification also applies for I_{OUT2} with all digital inputs = V_{IH}.
7. Guaranteed and not tested.
8. Absolute temperature coefficient is approximately ± 300 ppm/°C.

BURN-IN CIRCUIT



NOTES:
1. ALL RESISTORS ARE 1/4 WATT 5% TOLERANCE
2. C1 IS A 4.7µF ELECTROLYTIC CAPACITOR
3. C2 AND C3 ARE 0.01µF CERAMIC CAPACITORS
4. C1 AND C2 ONCE EVERY 10 DEVICES.

DICE CHARACTERISTICS



DIE SIZE 0.110 × 0.108 inch, 11,660 sq. mils
(2.692 × 2.794 mm, 7.52 sq. mm)

For additional DICE information refer to 1986 Data Book, Section 2.

1. CURRENT OUTPUT 1
2. CURRENT OUTPUT 2
3. GROUND
4. DIGITAL INPUT (BIT 1) (MOST SIGNIFICANT BIT)
5. DIGITAL INPUT (BIT 2)
6. DIGITAL INPUT (BIT 3)
7. DIGITAL INPUT (BIT 4)
8. DIGITAL INPUT (BIT 5)
9. DIGITAL INPUT (BIT 6)
10. DIGITAL INPUT (BIT 7)
11. DIGITAL INPUT (BIT 8)
12. DIGITAL INPUT (BIT 9)
13. DIGITAL INPUT (BIT 10)
14. DIGITAL INPUT (BIT 11)
15. DIGITAL INPUT (BIT 12) (LEAST SIGNIFICANT BIT)
16. POSITIVE POWER SUPPLY
17. REFERENCE INPUT VOLTAGE
18. INTERNAL FEEDBACK RESISTOR

WAFER TEST LIMITS at V_{DD} = +15V, V_{REF} = +10V, AGND = DGND = 0V, V_{OUT1} = V_{OUT2} = 0V, T_A = +25°C

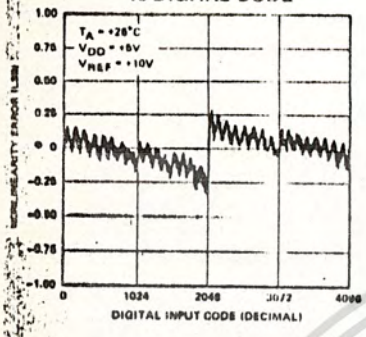
PARAMETER	SYMBOL	CONDITIONS	PM-7541G MIN	PM-7541G TYP	PM-7541G MAX	UNITS
STATIC ACCURACY						
Resolution	N		12	—	—	Bits
Nonlinearity	INL		—	—	±1	LSB
Gain Error (Note 1)	G_{ER}		—	—	±12.5	LSB
Power Supply Rejection	PSRR	V_{DD} = +14.5V to +15.5V	—	—	±0.01	%/%
Output Leakage Current (I_{OUT1}) (Note 2)	I_{LKG}	Digital Inputs = V_{IL}	—	—	±50	nA
REFERENCE INPUT						
Input Resistance	R_{REF}		5	—	20	kΩ
DIGITAL INPUTS						
Digital Input High	V_{IH}		2.4	—	—	V
Digital Input Low	V_{IL}		—	—	0.8	V
Input Leakage Current	I_{IL}	V_{IN} = 0 to 15V	—	—	±1	μA
POWER SUPPLY						
Supply Current	I_{DD}	Digital Inputs = V_{IH} or V_{IL}	—	—	2	mA

NOTES:
1. Using internal feedback resistor.
2. Specification also applies for I_{OUT2} but all Digital Inputs = V_{IH}.

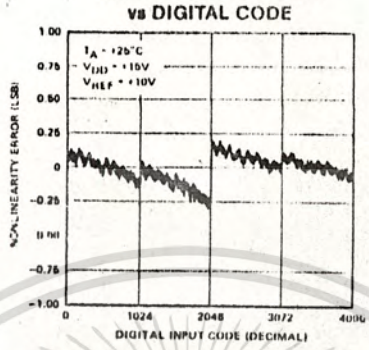
Electrical tests are performed at wafer probe to the limits shown. Due to variations in assembly methods and normal yield loss, yield after packaging is not guaranteed for standard product dice. Consult factory to negotiate specifications based on dice lot qualification through sample lot assembly and testing.

TYPICAL PERFORMANCE CHARACTERISTICS

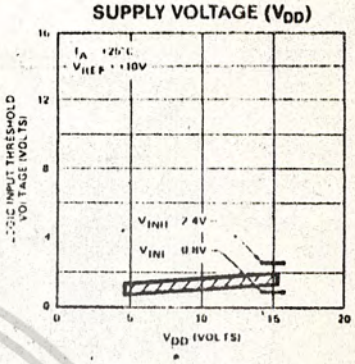
NONLINEARITY ERROR vs DIGITAL CODE



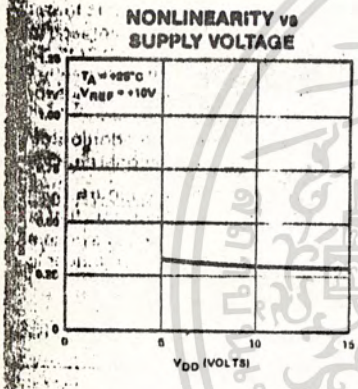
NONLINEARITY ERROR vs DIGITAL CODE



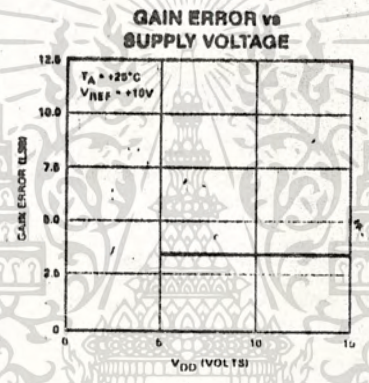
LOGIC INPUT THRESHOLD VOLTAGE vs SUPPLY VOLTAGE (V_{DD})



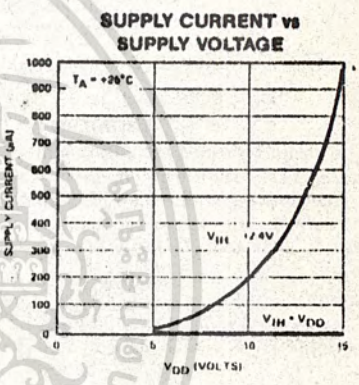
NONLINEARITY vs SUPPLY VOLTAGE



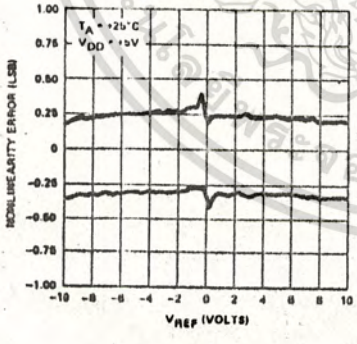
GAIN ERROR vs SUPPLY VOLTAGE



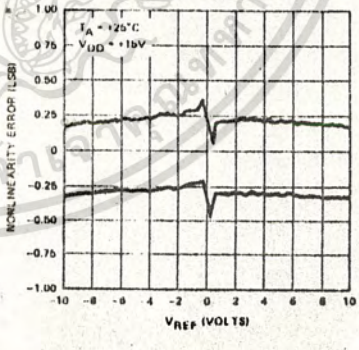
SUPPLY CURRENT vs SUPPLY VOLTAGE



NONLINEARITY ERROR vs REFERENCE VOLTAGE

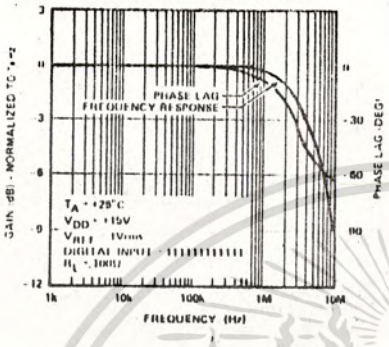


NONLINEARITY ERROR vs REFERENCE VOLTAGE

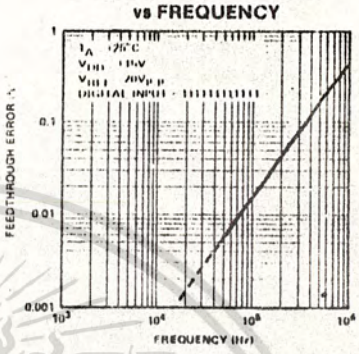


TYPICAL PERFORMANCE CHARACTERISTICS

V_{REF} PHASE LAG AND FREQUENCY RESPONSE



FEEDTHROUGH ERROR vs FREQUENCY



SPECIFICATION DEFINITIONS

RESOLUTION

The resolution of a DAC is the number of states (2^n) that the full-scale range (FSR) is divided (or resolved) into, where "n" is equal to the number of bits.

SETTLING TIME

Time required for the output function of the DAC to settle to within 1/2 LSB for a given digital input stimulus; i.e., zero to full scale.

GAIN

Ratio of the DAC's external-operational-amplifier output voltage to the V_{REF} input voltage.

FEEDTHROUGH ERROR

Error caused by capacitive coupling from V_{REF} to output with all switches OFF.

OUTPUT CAPACITANCE

Capacitance from I_{OUT1} or I_{OUT2} terminals to ground.

OUTPUT LEAKAGE CURRENT

Current which appears on I_{OUT1} terminal with all digital inputs LOW, or on I_{OUT2} terminal when all inputs are HIGH.

CIRCUIT DESCRIPTION

GENERAL CIRCUIT INFORMATION

The PM-7541 is a 12-bit multiplying D/A converter consisting of a highly-stable, silicon-chrome thin film R-2R ladder network and twelve pairs of NMOS current steering switches on a monolithic chip. Most applications require the addition of a voltage or current reference and an output operational amplifier.

A simplified circuit of the PM-7541 is shown in Figure 1. The R-2R inverted ladder binarily divides the input currents that are switched between I_{OUT1} and I_{OUT2} bus lines. This switching allows a constant current to be maintained in each ladder leg independent of the input code.

FIGURE 1: SIMPLIFIED DAC CIRCUIT

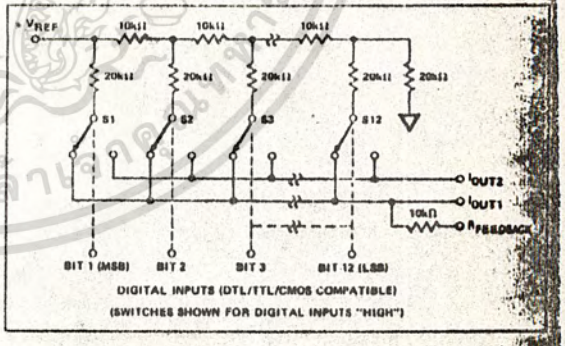
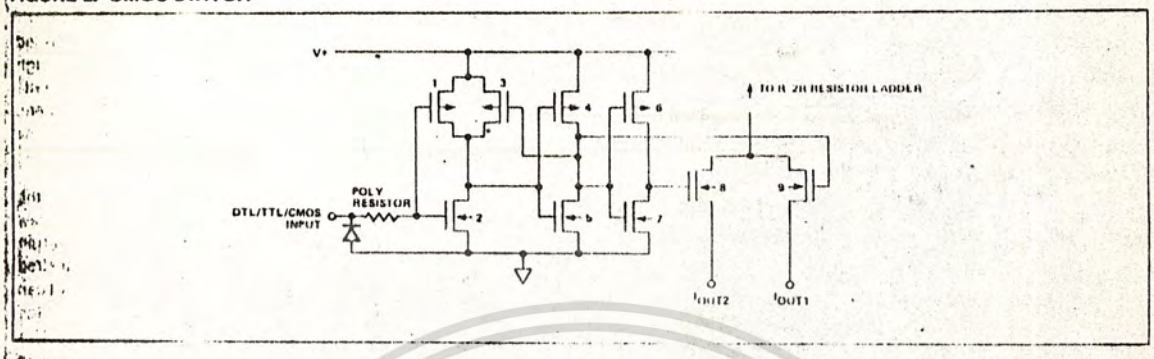


FIGURE 2: CMOS SWITCH



One of the twelve CMOS switches is shown in Figure 2. The digital input stage, devices 1, 2, and 3, drives the two inverters, devices 4, 5, 6, and 7; these inverters in turn drive the two output current steering switches, devices 8 and 9. Devices 1, 2, and 3 are designed such that the digital control inputs are DTL, TTL, and CMOS compatible over the full military temperature range.

The twelve output current-steering switches are in series with the R-2R resistor ladder, and therefore, can introduce bit errors. It is essential then, that the switch "ON" resistance be binarily scaled so that the voltage drop across each switch remains constant. If, for example, switch 1 of Figure 1 was designed with an "ON" resistance of 10 ohms, switch 2 for 20 ohms, etc., then with a 10 volt reference input, the current through switch 1 is 0.5mA, switch 2 is 0.25mA, etc., a constant 5mV drop will then be maintained across each switch.

EQUIVALENT CIRCUIT ANALYSIS

Figures 3 and 4 show the equivalent circuits for all digital inputs LOW and HIGH respectively. The reference current is switched to I_{OUT2} when all inputs are LOW and I_{OUT1} when inputs are HIGH. The $I_{LEAKAGE}$ current source is the combination of surface and junction leakages to the substrate; the 1/4096 current source represents the constant 1-bit current drain through the ladder terminating resistor. The output capacitance is dependent upon the digital input code, and is therefore modulated between the low and high values.

FIGURE 3: PM-7541 EQUIVALENT CIRCUIT (ALL INPUTS LOW)

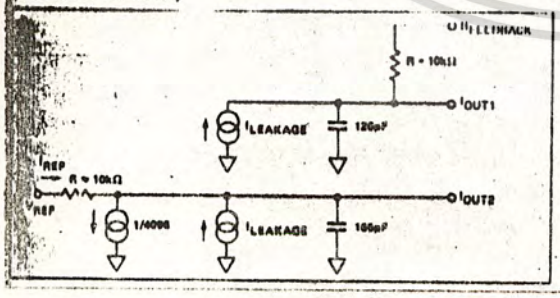
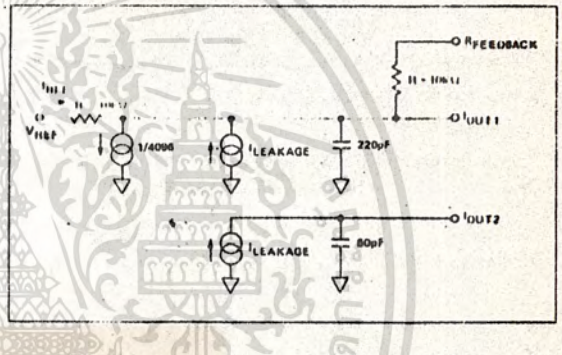


FIGURE 4: PM-7541 EQUIVALENT CIRCUIT (ALL DIGITAL INPUTS HIGH)



DYNAMIC PERFORMANCE

OUTPUT IMPEDANCE

The output resistance, as in the case of the output capacitance, is also modulated by the digital input code. The resistance looking back into the I_{OUT1} terminal, may be anywhere between 10kΩ (the feedback resistor alone when all digital inputs are low) and 7.5kΩ (the feedback resistor in parallel with approximately 30kΩ of the R-2R ladder network resistance when any single bit logic is high). The static accuracy and dynamic performance will be affected by this modulation. The gain and phase stability of the output amplifier, board layout, and power supply decoupling will all affect the dynamic performance of the PM-7541. The use of a compensation capacitor may be required when high-speed operational amplifiers are used. It may be connected across the amplifiers feedback resistor to provide the necessary phase compensation to critically damp the output.

The considerations when using high-speed amplifiers are:

1. Phase Compensation (See Figures 5 and 6).
2. Power supply decoupling at the device socket and use of proper grounding techniques.