การออกแบบระบบควบคุมของแขนกลอ่อนสองข้อต่อ

A CONTROL SCHEME FOR TWO-LINK FLEXIBLE ROBOT ARMS

โดย

นาย ธนา ภัทรเดช รหัสประจำตัว 40012014

นาย ภูดิท หมั่นตระกูล รหัสประจำตัว 40012024

อาจารย์ที่ปรึกษา

ดร. ปิติเขต สู้รักษา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

สาขา วิชาเทคโนโลยีอิเล็คทรอนิคส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ชื่อโครงงานภาษาไทย	การออกแบบระบบควบคุมของแขนกลอ่อนตัวสองข้อต่อ
ชื่อโครงงานภาษาอังกฤษ	A CONTROL SCHEME FOR TWO-LINK FLEXIBLE ROBOT ARMS
ผู้จัดทำ	นาย ธนา ภัทรเดช
	นาย ภูดิท หมั่นตระกูล
อาจารย์ที่ปรึกษา	ดร. ปีติเขต สู้รักษา
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2541

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

ลงชื่อ.....อาจารย์ที่ปรึกษา

(ดร. ปีติเขต สู้รักษา)

คณะกรรมการสอบปริญญานิพนธ์

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

ลิขสิทธิ์ของวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การออกแบบระบบควบคุมของแขนกลอ่อนตัวสองข้อต่อ

นาย ธนา ภัทรเดช
นาย ภูดิท หมั่นตระกูล
ดร. ปิติเขต สู้รักษา (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2541

บทคัดย่อ

ปริญญานิพนธ์เล่มนี้นำเสนอการควบคุมแขนกลอ่อนตัวสองข้อต่อ โดยใช้โปรแกรมเคลไฟล์ผ่านวงจรอินเตอร์เฟสที่ออกแบบ แขนกลอ่อนตัวสองข้อต่อประกอบด้วย ดีซีเซอร์โวมอเตอร์ (DC Servo Motor) ใช้ขับเคลื่อนแขนส่วนแรกและสเตปปิ้งมอเตอร์ (Stepping Motor) ขับเคลื่อนแขนส่วนที่สอง ทั้งสองส่วนนี้ควบคุมโดยโปรแกรมคอมพิวเตอร์

โครงงานนี้จะประยุกต์ใช้แขนกลอ่อนตัวสองข้อต่อทำการเล่นดนตรี ตามโปรแกรมโดยมีทั้งโหมดควบคุมโดยผู้ใช้และโหมดอัตโนมัติ โปรแกรมนี้สามารถควบคุมการเล่นดนตรีของแขนกลโดยที่เครื่องดนตรีที่เล่นเป็นแบบจำกัดโน้ตไม่เกินแปดตัว โดยมีเรียงกันของตัวโน้ตลักษณะเป็นแบบครึ่งวงกลม

A CONTROL SCHEME FOR TWO-LINK FLEXIBLE ROBOT ARMS

Mr. Thana Puttaradet 40012014
Mr. Phudit Mhuntrakool 40012024
Dr. Pitikhate Sooraksa (Adviser)
Academic Year 1998

Abstract

This thesis presents a control scheme for two-link flexible robot arms by sending control signals from the personal computer via the designed interface card to the robot. The two-link flexible robot arms used in this project consists of two important parts which are a DC control motor for driving the first link and a stepping motor for driving the second link. Both parts are controlled by computer program written in Delphi ™.

This project also shows an application of the robot by using it for playing a musical instrument. However, the music instrument must be the one having musical notes not exceeding eight notes and the notes on the instrument must be arranged in semicircle.

กิตติกรรมประกาศ

โครงงาน การออกแบบควบคุมระบบแขนกลอ่อนตัวสองข้อต่อ นี้ประสบความสำเร็จได้เนื่องจากได้รับปรึกษาและการแนะนำจาก ดร. ปิติเขต สู้รักษา (อาจารย์ที่ปรึกษา) จึงขอขอบคุณมา ณ โอกาสนี้

หากรายงานฉบับนี้เกิดข้อผิดพลาด หรือบกพร่องแต่ประการใด ทางคณะผู้จัดทำขอน้อมรับและขออภัยมา ณ โอกาสนี้ด้วย

ธนา ภัทรเดช

.....

(นาย ธนา ภัทรเดช)

.....

(นาย ภูดิท หมั่นตระกูล)

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

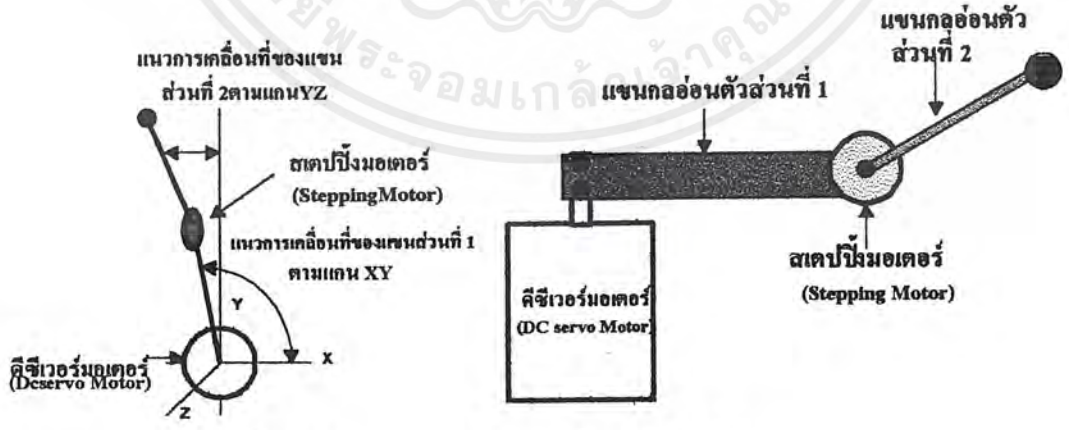
1.1 ความเป็นมาของโครงงาน

แขนกลหรือแขนหุ่นยนต์ก็เป็นเครื่องจักรกลอีกประเภทหนึ่งที่พบเห็นได้มากในโรงงานอุตสาหกรรมโดยทั่วไปสาเหตุที่ต้องเป็นแขนกลก็เพื่อ

1. งานที่กระทำซ้ำๆ บ่อยๆ
2. งานที่ทำแล้วอาจเกิดอันตรายได้ง่าย เช่น งานประเภทเชื่อมในที่สูงเหนือศีรษะ งานใต้ดิน หรืองานใต้ทะเลลึกๆ เป็นต้น
3. งานที่ใช้ความละเอียดค่อนข้างสูง
4. งานที่สิ่งแวดล้อมไม้เอื้ออำนวย เช่น งานในอวกาศ ในที่ที่มนุษย์ไม่สามารถอยู่ได้

แขนกลที่พบส่วนใหญ่แล้วจะเป็นแขนกลแบบที่มีน้ำหนักมากและมีขนาดใหญ่ แต่ในที่นี้จะกล่าวถึงแขนกลอ่อนตัวสองข้อต่อซึ่งมีข้อดีในการใช้งานคือ มีน้ำหนักเบาจึงทำให้ประหยัดพลังงานในการขับเคลื่อน และสามารถให้ผลตอบสนองที่รวดเร็ว

สำหรับแขนกลอ่อนตัวสองข้อต่อที่กล่าวถึงนี้ประกอบไปด้วยส่วนที่สำคัญสองส่วนคือ มอเตอร์กระแสตรงที่ทำให้แขนกลหมุนไปยังตำแหน่งที่ต้องการ และส่วนของแขนกลอ่อนตัว โดย



รูปที่ 1.1 รูปแขนกลอ่อนตัวสองข้อต่อ

ที่ปลายด้านหนึ่งของแขนกลจะยึดติดกับตัวดีซีมอเตอร์ (DC Server Motor) อีกด้านจะยึดติดกับสเตปปิ้งมอเตอร์ (Stepping Motor)ทำให้ตัวแขนกลมีการเคลื่อนที่ในแนวระนาบ XY และระนาบ YZ ตามลำดับดังแสดงในรูปที่ 1.1

1.2 ขีดความสามารถของโครงงาน

1. สามารถใช้โปรแกรมเดลไฟล์ (Delpli) ในการทำงานการควบคุมจากคอมพิวเตอร์ โดยผ่านวงจรอินเตอร์เฟสได้อย่างถูกต้อง
2. สามารถใช้โปรแกรมในการควบคุมตำแหน่งองศาในการเคลื่อนที่ได้ตรงตามค่าที่กำหนดได้อย่างถูกต้อง
3. สามารถใช้โปรแกรมในการควบคุมแขนกลนี้เล่นเครื่องดนตรีตามตัวโน้ตที่กำหนด โดยมีทั้งหมด 8 ตัวโน้ต
4. สามารถใช้โปรแกรมเดลไฟล์และวงจรอินเตอร์เฟสนี้เป็นการศึกษาถึงระบบควบคุมให้สูงยิ่งๆขึ้นไป

1.3 เนื้อหาพอสังเขป

บทที่ 2 จะเป็นทฤษฎีและหลักการ ที่จะกล่าวในเรื่องของการใช้งานโปรแกรมเดลไฟล์เบื้องต้น เพื่อไปควบคุมแขนกลและหลักการอินเตอร์เฟสโดยใช้สล็อต IBM/PC โดยมีการแปลงสัญญาณดิจิทอลเป็นอะนาล็อก (D/A Convertor)

บทที่ 3 การออกแบบและการสร้าง กล่าวถึงการสร้างโปรแกรมเดลไฟล์ที่ใช้ควบคุมแขนกลอ่อนตัวสองข้อต่อ และการออกแบบสร้างวงจรอินเตอร์เฟสที่เชื่อมระหว่างตัวแขนกลกับโปรแกรมควบคุม

บทที่ 4 ผลการทดลอง แสดงผลการรันโปรแกรม ค่าแรงดันต่างๆที่ออกจากวงจรอินเตอร์เฟส

บทที่ 5 ปัญหาและแนวทางการพัฒนา กล่าวถึงข้อสรุปการทำงาน ข้อบกพร่องต่างๆและวิธีการแก้ไข

บทที่2
ทฤษฎีและหลักการ

กล่าวนำ

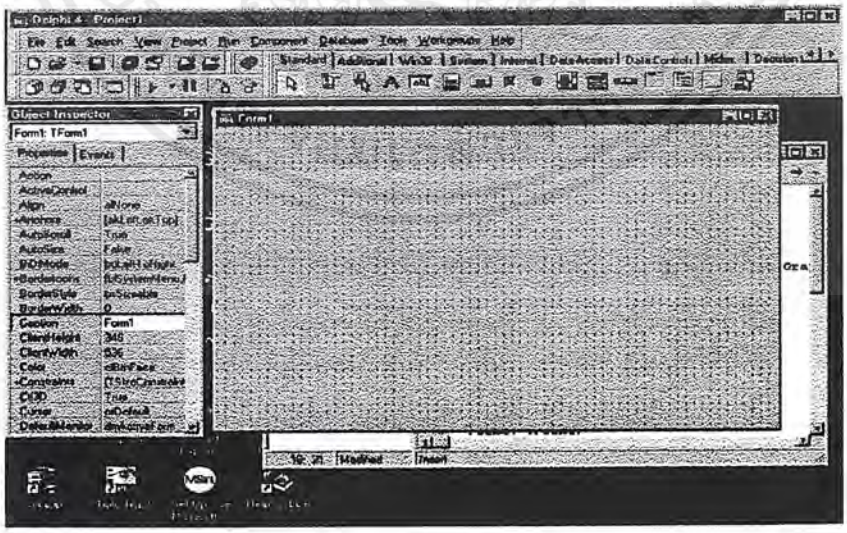
ในบทนี้จะกล่าวถึงทฤษฎีเบื้องต้น ที่เกี่ยวข้องกับการควบคุมแขนกลโดยแบ่งออกเป็น การใช้งานเดลไฟล์(Delphi) เบื้องต้น เพราะในโครงงานนี้จะใช้โปรแกรมเดลไฟล์ควบคุมแขนกล และการอินเตอร์เฟส โดยจะแบ่งออกเป็นส่วนของวงจรแปลงสัญญาณดิจิตอลเป็นอะนาล็อก (DAC CONVERTOR) และตำแหน่งขาต่างๆ บน สล็อต IBM PC(ISA SLOT) ซึ่งจะนำมาทำการ อินเตอร์เฟส และการดีโค้ดสัญญาณจากสล็อต IBM PC โดยมีลายละเอียดเรียงตามลำดับดังต่อไปนี้

2.1 การเขียนโปรแกรมด้วยโปรแกรมเดลไฟ

2.1.1 หน้าจอของโปรแกรมเดลไฟ

หน้าจอของเดลไฟจะแบ่งออกได้เป็น 4 ส่วนใหญ่ ๆ ดังรูปที่ 2.1 คือ

1. หน้าต่างหลัก
2. หน้าต่างฟอร์ม (Form)



รูปที่ 2.1 หน้าจอของเดลไฟโดยรวม

3. หน้าต่างยูนิต (Unit)
4. หน้าต่างออบเจ็กต์อินสเปกเตอร์(Object Inspector)

2.1.2 หน้าต่างหลัก

จากรูปที่ 2.1 หน้าต่างหลักประกอบด้วย

1. ไตเติลบาร์(Title bar) จะแสดงรายชื่อของชิ้นงาน (Application) แต่ละชิ้นงาน
2. เมนูบาร์ (Manu bar) แสดงรายการตั้งแต่ File จนถึง Help
3. กล่องคอมโพแนนต์ (Component) จะเป็นที่เก็บคอมโพแนนต์ต่าง ๆ ที่จะไปว่างไว้บนฟอร์ม
4. สปีดบาร์ (Speed Bar) คือ ส่วนของปุ่มที่ใช้ในการเรียกการทำงานดังรูปที่ 2.2 ประกอบไปด้วย

   4.1 New คือ ปุ่มที่ใช้เปิด Form ใหม่

   4.2 Open คือ ปุ่มที่ใช้เปิด File

   4.3Open Project คือ ปุ่มที่ใช้เปิดโปรเจกต์ใหม่ที่สร้างไว้แล้ว

   4.4 Save File คือ ปุ่มที่ใช้เปิดไฟล์ใหม่ที่สร้างไว้แล้ว

   4.5 Save All คือ ปุ่มที่ใช้เซฟโปรเจกของการทำงานที่กำลังทำงานอยู่

   4.6 Open Project คือ ปุ่มที่ใช้เปิดโปรเจกต์ใหม่ที่สร้างไว้แล้ว

   4.7 Add File To Project คือ ปุ่มที่ใช้ในการนำยูนิต (ไฟล์นามสกุล .PAS) ของโปรเจกต์อื่นเข้ามาทำงานร่วมกับโปรเจกต์ที่เรากำลังทำงานอยู่

   4.8 Remove File To Project คือ ปุ่มที่ใช้การนำยูนิตของโปรเจกต์ที่กำลังทำงานอยู่ออกไป

   4.9 ใช้เลือกยูนิตที่มีอยู่แล้วใน

   4.10 โปรเจกต์ขึ้นมาใช้งาน

   4.11 Select Form Form List คือ ปุ่มที่ใช้เลือกฟอร์มที่มีอยู่แล้วใน โปรเจกต์ขึ้นมาใช้งาน

   4.12 Toggle Form/Unit คือ ปุ่มที่ใช้ในการสลับการทำงานระหว่าง หน้าต่างฟอร์มกับหน้าต่างยูนิต

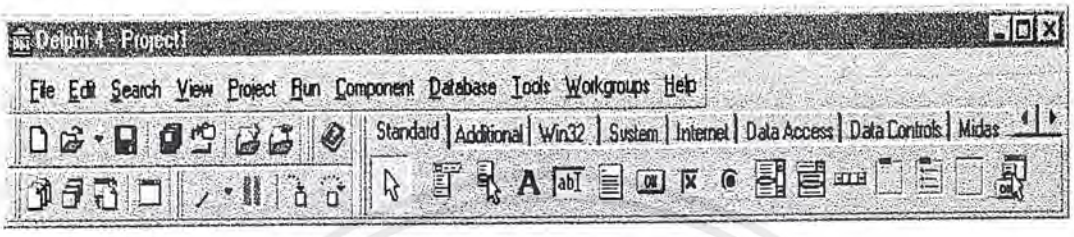
   4.13 New form คือ ปุ่มที่ใช้เรียกฟอร์มใหม่ขึ้นมาใช้งานร่วมกับโปรเจกต์กำลังใช้งานอยู่

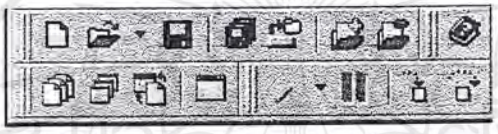
   4.14 Run คือ ปุ่มที่ทำให้ตัว Project Compile เริ่มทำงาน

4.15 Pausc คือ ปุ่มที่ทำให้การทำงานของโปรเจกต์หยุดชั่วคราว

4.16 Trace into คือ ปุ่มที่ใช้แสดงผลการทำงานของโปรเจกต์ทีละขั้นตอน

4.17 Step Over คือ ปุ่มที่ใช้ในการข้ามการแสดงผลบางช่วงไป





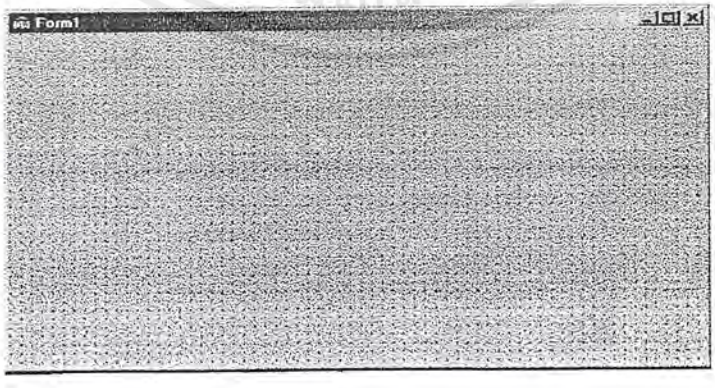
4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8

4.9 4.10 4.11 4.12 4.13 4.14 4.15 4.16

รูปที่ 2.2 หน้าต่างหลัก

2.1.3 หน้าต่างฟอร์ม

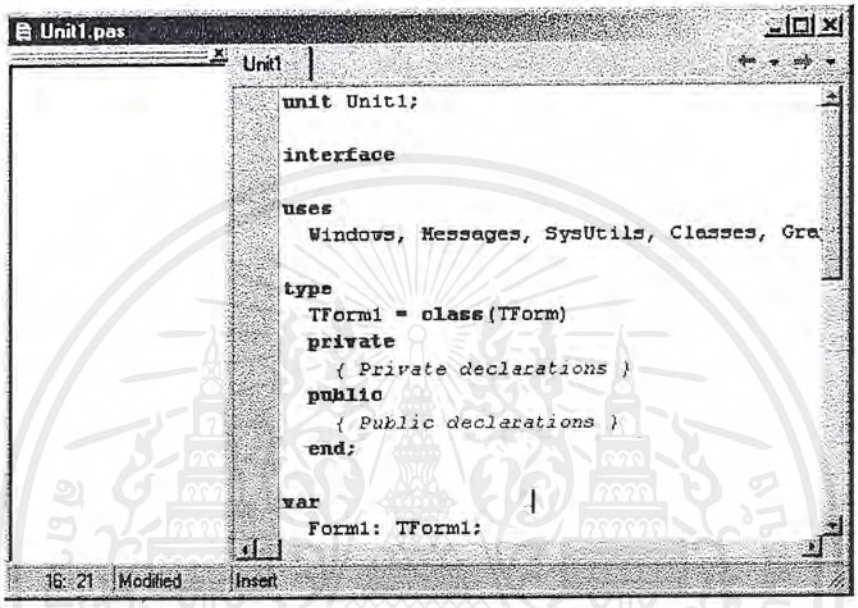
หน้าต่างฟอร์มแสดงดังรูปที่ 2.3 เป็นหน้าจ่างที่มีไว้สำหรับแสดงผลในการทำงานของตัวโปรเจกต์ เวลาใช้ให้นำคอมโปรแนนต์ที่ต้องการใช้ลงไปวาง แล้วกำหนด Properties กับ Events ใน ออบเจ็กต์อินสเปคเตอร์



รูปที่ 2.3 หน้าต่างฟอร์ม

2.1.4 หน้าต่างยูนิต

หน้าต่างยูนิตแสดงดังรูปที่ 2.4 เป็นหน้าต่างที่มีไว้สำหรับใส่เงื่อนไขในการทำงานของโปรแกรม โดยผ่านทาง Events ของออบเจ็กต์อินสเปคเตอร์ หรือจะใส่เงื่อนไขโดยตรงก็ได้



รูปที่ 2.4 หน้าต่างยูนิต

2.1.5 หน้าต่างออบเจ็กต์อินสเปคเตอร์

หน้าต่างออบเจ็กต์อินสเปตเตอร์ดังแสดงในรูปที่ 2.5 จะประกอบด้วย Properties จะเป็นตัวกำหนดการทำงานของคอมโพรแนนต์ต่าง ๆ

1. Events จะเป็นตัวผ่านจากคอมโพแนนต์ไปกำหนดเงื่อนไขต่าง ๆ ในยูนิตเช่น
   - OnClick จำทำงานก็ต่อเมื่อเราเลือกด้วยเมาส์หรือด้วยคีย์บอร์ด
   - OnMouseDown จะทำงานก็ต่อเมื่อเราคลิกเมาส์ลง
   - OnMouseUp จะทำงานก็ต่อเมื่อเราคลิกเมาส์ขึ้น
   - OnKeyDown จะทำงานโดยการกดคีย์ลงจะไปทำงานที่ Key code ที่กำหนดไว้ในเงื่อนไขของ Events นั้น ๆ ในโปรแกรม

รูปที่ 2.5 หน้าต่าง Object Inspector

2.2 เริ่มต้นการใช้งาน

การเริ่มต้นการใช้งานจริงนี้ จะเริ่มอธิบายตั้งแต่การเปิดใช้งานของ โปรแกรม เดลไฟ, การเปิดโปรเจกต์ใหม่,การวางคอมโพแนนต์,การกำหนดพร็อพเพอร์ตี้,การกำหนดอีเวนต์,การกำหนดเงื่อนไข,การเซฟไฟล์,การคอมไพล์,การเปิดโปรเจกต์เก่าขึ้นมาแก้ไข และการนำโปรเจกต์ไปใช้งานจริงหลังจากคอมไฟล์แล้ว

1. การเปิดโปรแกรมเดลไฟล์ขึ้นมาใช้งาน

เลือกไปที่ Start แล้วเลือกไปที่ Programs / Borland Delphi ก็จะแสดงหน้าจะของโปรแกรมเดลไฟ ออกมา

2. การเปิดโปรเจกต์ใหม่และการใช้งาน

1. ไปที่เมนูบาร์ เลือก File / New Application

2. กำหนดชื่อ Project ใหม่ให้เหมะสมกับการใช้งาน

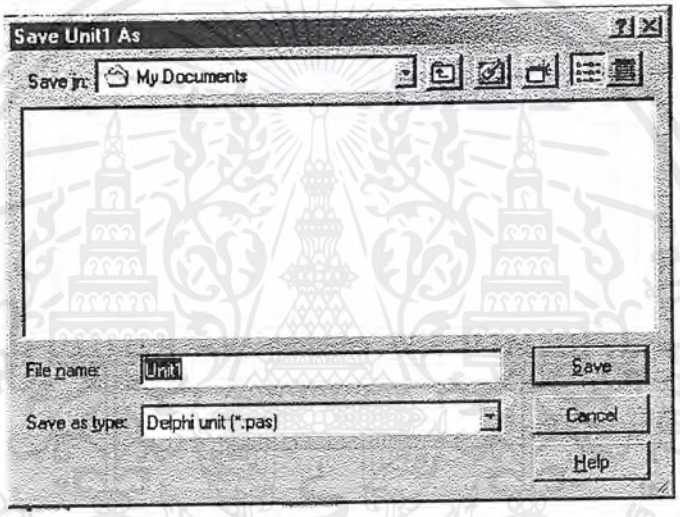
2.1 ไปที่เมนูบาร์ เลือก File/Save Project As หรือจะใช้เมาส์ไปคลิกที่ปุ่ม Save All บนสปีดบาร์

2.2 จะมีหน้าต่าง Save Unit As แสดงขึ้นมาดังรูปที่ 2.6 เป็นหน้าต่างที่มีไว้ให้เซฟไฟล์ยูนิต ให้พิมพ์ชื่อไฟล์ยูนิตที่เราต้องการลงในช่อง File name

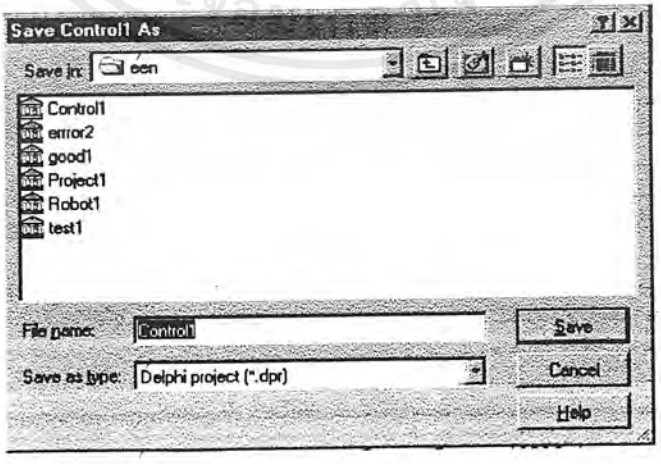
2.3 คลิกไปที่ Save

2.4 จะมีหน้าต่าง Save Project As แสดงขึ้นมาเป็นหน้าต่างที่มีไว้ให้เซฟไฟล์โปรเจกต์ ให้พิมพ์ชื่อไฟล์โปรเจกต์ที่เราต้องการลงในช่อง File name

2.5 คลิกไปที่ Save



รูปที่ 2.6 หน้าต่าง Save Unit A



รูปที่ 2.7 หน้าต่าง Save Project As

2.3 ตัวอย่างการเริ่มต้นใช้งาน

ตัวอย่างการกำหนดชื่อโปรเจกต์ใหม่

1.KPROGRAM PNEW (TEST1.dpr)

2.Unit PNEW1 (TEST.pas)

2.3.1 อธิบายตัวอย่างการกำหนดชื่อโปรเจกต์ใหม่

บรรทัดที่ 1 คือการตั้งชื่อโปรเจกต์ใหม่ โดยให้ชื่อว่า TEST1 แล้วเซฟเป็นสกุล dpr

บรรทัดที่ 2 คือการตั้งชื่อยูนิตใหม่ โดยให้ชื่อว่า TEST แล้วเซฟเป็นสกุล pas ทั้งสองบรรทัดข้างบนนี้จะทำได้ โดยการเลือกไปที่สปีดบาร์แล้งเลือกที่ Save All หรือเลือกที่ เมนู File/Save All จะแสดงหน้าต่าง Save Unit1 As ดังในรูปที่ 2.8 ให้พิมพ์ TEST ลงในช่อง File name แล้วกด Save หลังจากนั้นจะมีหน้าต่าง Save Project1 As แสดงขึ้นมา ดังในรูปที่ 2.7 ให้พิมพ์ TEST1 ลงในช่อง File name แล้วกด Save

2.3.2 จัดการกำหนดคอมโพเนนต์ที่ต้องการ

ตัวอย่างการวางคอมโพเนนต์

Components

1. Form=Form1
   - Caption = Pnew
2. Standard :: Button = Button1
   - Caption = OK
3. Standard :: Labol = Label1
   - Caption = Hi

อธิบายตัวอย่างการวางคอมโพเนนต์

ข้อ 1 เป็นการให้ Caption ของ Form1 เป็น Pnew โดยผ่านทาง Properties

ข้อ 2 เป็นการนำคอมโพเนนต์ Button จากกล่องคอมโพเนนต์ Standard มาวางบน Form1 จะได้ Button1 แล้วให้ค่า Properties ของ Button1 ให้ Caption เป็น OK

ข้อ 3 เป็นการนำคอมโพเนนต์ Label จากกล่องคอมโพเนนต์ Standard มาวางบน Form1 จะได้ Label1 แล้วให้ค่า Properties ของ Label1 ให้ Caption เป็น Hi

เมื่อกำหนดคอมโพเนนต์ตามตัวอย่างได้แล้วก็จะได้ฟอร์มออกมาจัดการใส่เงื่อนไขโดยผ่าน Events

ตัวอย่างใส่เงื่อนไขโดยผ่าน Events

Events OnClick ของ Button1

```
1 : procedure Tform1.Button1Click(Sender: Tobject)
2 : begin
3 : Label1.Caption:=Hello;
4 : end;
```

อธิบายตัวอย่างการใส่เงื่อนไขโดยผ่าน Events

การที่เราจะเข้าถึง Events ตามในตัวอย่างนี้ได้ เราต้องเข้าไปกำหนด Events (Object) ที่ชื่อว่า Button1 โดยทำดังต่อไปนี้

1. ใช้เมาส์เลือกที่วัตถุ Button1
2. ใช้เมาส์เลือกไปที่ Events ในออบเจ็กต์อินสเปคเตอร์
3. ใช้เมาต์คัลเบิลคลิกไปที่ OnClick
4. และหน้าที่ยูนิตก็จะแสดงขึ้นมา
5. แล้วใส่เงื่อนไขตามบรรทัดที่ 3 ในตัวอย่างลงไป (Label.Caption := Hello;)

2.3.3 ทำการคอมไพล์ (COMPILE) ให้อยู่ในสกุล exe

1 ขั้นตอนในการคอมไพล์มี 2 วิธี คือ

1.1 เลือกไปที่เมนู Project/Compile หรือกด Ctrl+F9

1.2 เลือกไปที่เมนู Run/Run หรือกด F9 แต่วิธีง่ายที่สุด คือใช้เมาส์เลือกไปที่ปุ่ม Run

2 เมื่อเข้าใจขั้นตอนการคอมไพล์แล้วก็ทดลองนำโปรแกรม PNEW มาทำดู จะได้ผล

3 เมื่อต้องการจะปิดผลการรัน ก็ให้ใช้แมาส์เลือกที่ปุ่ม X ตรงมุมขวาบนของหน้าต่างที่กำลังรันอยู่

2.3.4 ทดลองปิดโปรเจกต์ PNEW

1 เลือกไปที่เมนู File/Open หรือเลือกไปที่ปุ่ม Open Project

2 ให้เมาส์เลือกไปที่ชื่อโปรแกรมที่ต้องการจะเปิด

3 แล้วเลือกปุ่ม Open โปรแกรมก็จะเปิดออกมาพร้อมที่จะรับการแก้ไข

2.3.5 การนำโปรแกรมที่คอมไพล์แล้วมาทำเป็น SHORTCUT

ขั้นตอนดังต่อไปนี้

1 ใช้เมาส์คลิดขวา แล้วเลือกไปที่ New / Shortcut ก็จะแสดงหน้าต่าง Create Shortcut ขึ้นมา

2 ปุ่ม Browse ไฟล์ที่เราต้องการที่จะทำเป็น Shortcut

3 เมื่อเลือกได้แล้วให้ไปเลือกที่ปุ่ม Open จะกลับมาที่หน้าต่าง Create Shortcut

4 เลือกไปที่ Next และ Finish ก็จะได้ Shortcut

2.4 วงจรดีเอซี

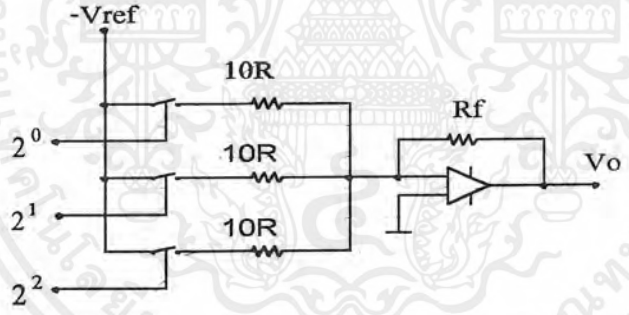
วงจรดีเอซี คือ วงจรที่ใช้สำหรับเชื่อมต่อระหว่างวงจรดิจิตอลกับวงจรอะนาลอก สัญญาณอะนาลอกเป็นสัญญาณที่เกิดขึ้นจากธรรมชาติ ตัวตรวจจับสัญญาณจะเป็นหน่วยเปลี่ยนกระบวนการทางฟิสิกส์ เช่น อุณหภูมิ แรงดัน ความชื้นหรืออื่นๆ ให้เป็นสัญญาณทางไฟฟ้าในรูปของแรงดัน กระแส หรือความต้านทานก็ตาม แต่จะมีความยุ่งยากมากขึ้นหากต้องการเก็บสัญญาณอะนาลอกไว้ตลอดในช่วงเวลานานๆ เพื่อนำมาใช้ในการเปรียบเทียบหรือคำนวณในภายหลัง ตรงกันข้ามคอมพิวเตอร์สามารถทำงานดังกล่าวนี้ได้ดีกว่ามากด้วยสัญญาณดิจิตอลหากเมื่อใดต้องการที่จะนำผลที่ได้จากการประมวลผลด้วยคอมพิวเตอร์ออกไปควบคุมอุปกรณ์หรือเครื่องจักรกลใดๆ ที่ใช้สัญญาณอะนาลอก จำเป็นต้องมีวงจรดีเอซีต่อร่วมด้วยเสมอ

หลักการเบื้องต้นของวงจรดีเอซี หากนำข้อมูลดิจิตอลขนาด 3 บิตจาก 000 ถึง 111 มาแปลงให้เป็นเลขฐานสิบจะได้ดังแสดงในตาราง2.1

จากตารางจะเห็นว่าเลขฐานสิบที่ได้จะเป็นการรวมค่าของเลขฐานสองในแต่ละหลักที่มีค่าตาม 2 ยกกำลัง จาก 0 + 0 + 0 = 0 ถึง 4 + 2 1 = 7 สถานะลอจิกของเลขฐานสองแต่ละหลักสามารถนำไปควบคุมสวิตช์อิเลกทรอนิกส์ทางด้านอินพุตของวงจรออปแอมป์ซึ่งทำหน้าที่เป็นวงจรขยายผลรวม โดยการกำหนดค่าความต้านทานอินพุตของวงจรเป็นแบบสัดส่วนดังแสดงในรูป2.8

เลขฐานสอง			เลขฐานสิบ
2^2	2^1	2^0	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

ตารางที่ 2.1 แสดงการแปลงรหัสดิจิตอล 3 บิท เป็นเลขฐานสิบ



รูปที่ 2.8 แสดงวงจรดีเอซีแบบสัดส่วนความต้านทาน

จากรูปที่ 2.8 เราหาค่า Vo ได้ดังนี้

$$Vo = -(-Vref)\, Rf\, (\, 1/10R + 1/5R + 1/2.5R)$$

$$= \left(\frac{1}{10} + \frac{1}{5} + \frac{1}{2.5}\right) Rf/R$$

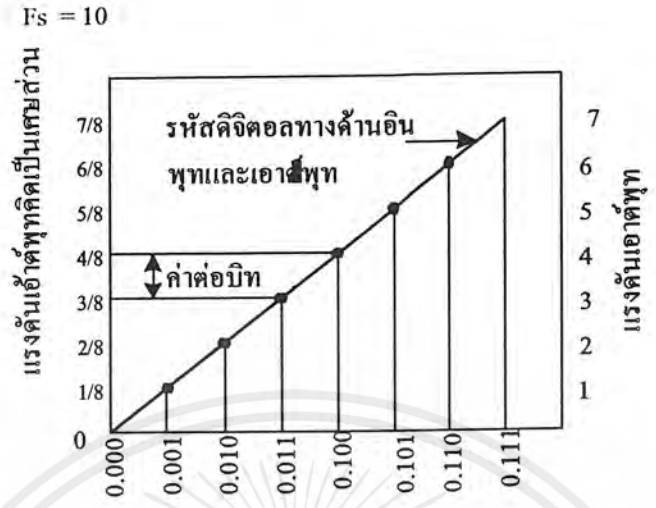
$$= 0.1\, Vref\, (\, 1 + 2 + 4\,)\, Rf/R$$

จากสมการแสดงให้เห็นว่า Vo จะมีค่าแตกต่างกันได้หากสวิตช์ควบคุมอินพุท 2^0 ถึง 2^2 ปิดและเปิดตามตารางที่ 2.2ในที่นี้สมมุติว่า Vref มีค่า -10 โวลต์ Rf/R มีค่าเป็น 1 ลอจิก "0" ควบคุมให้สวิตช์เปิด ลอจิก "1" ควบคุมให้สวิตช์ปิด

คุณสมบัติเชิงอุดมคติของวงจรดีเอซี จากรูปวงจร 2.8 และตาราง 2.2 หากนำค่ามาเขียนเป็นกราฟคุณสมบัติระหว่างอินพุตและเอ้าต์พุตได้ดังแสดงในรูป2.9โดยอินพุตคิดเป็นลักษณะเศษส่วน เช่น 0.111 เท่ากับ 7/8 และเอ้าต์พุต V_{FS} คือค่าเต็มสเกลคิดเป็น 1.0 จะเป็นค่าที่ไม่เกิดขึ้นเลยในทางปฏิบัติ เพราะอินพุตสูงสุดจะมีค่าเพียง 0.111 = 7/8 เท่านั้น จากตารางที่2.2ค่าสูงสุดของสัญญาณดิจิตอล 3บิต คือ 111 และค่าแรงดันเอ้าต์พุตสูงสุดเท่ากับ 7 โวลต์ ซึ่งสามารถนำมาเขียนกราฟคุณสมบัติของดีเอซีขึ้นใหม่ดังแสดงในรูป 2.10 เพื่อใช้หาค่าความละเอียดของดีเอซี

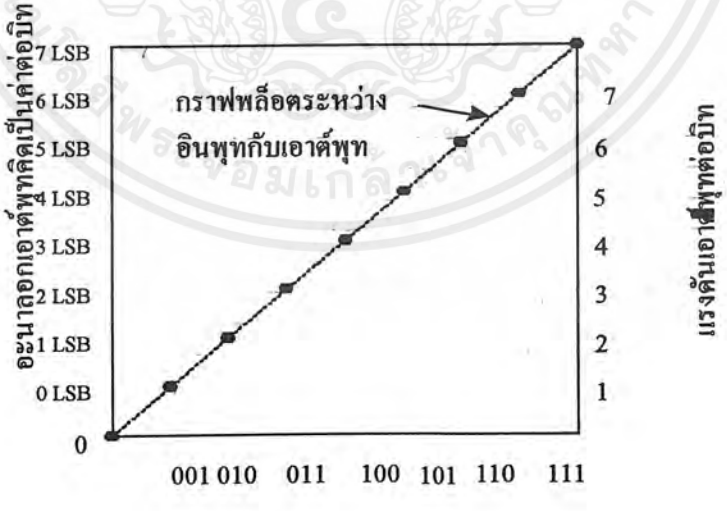
สวิตช์			Vo
2^2	2^1	2^0	ขณะ Vref = -10 V
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

ตารางที่ 2.2 แสดงลอจิกอินพุทและเอาต์พุทของวงจรรูปที่ 2.8



รูปที่ 2.9 แสดงกราฟระหว่างอินพุทกับเอาต์พุทของดีเอซี 3 บิต

2.4.1 ความละเอียดของดีเอซี



รูปที่ 2.10 แสดงกราฟอะนาลอกเอาต์พุทกับดิจิตอลอินพุทของดีเอซี

จากรูป 2.10 จะเห็นว่ามี 8 อินพุตของลอจิกที่ไม่ซ้ำกันจาก 000 ถึง 111 ดังนั้นเอ้าต์พุตจะแบ่งได้เป็น 8 ส่วน ดีเอซีขนาดอินพุต 3 บิต จะมีค่าเอ้าต์พุตเท่ากับ 8 หรือ 2^3 ดังนั้นถ้าเป็นดีเอซี n บิต ค่าความละเอียดหาได้จาก

ความละเอียด = 2 กำลัง n

หรืออาจจะนิยามความละเอียดเป็นค่าของบิตต่ำสุดดังนี้

ความละเอียดของแรงดัน = ค่าแรงดันเอ้าต์พุตขณะบิตต่ำสุดเป็นลอจิก "1"

สมการแรงดันเอ้าต์พุตของดีเอซี หาได้จากสูตร

V_0 = ความละเอียดของแรงดันคูณด้วย D

กำหนดให้ V_0 คือ ค่าแรงดันเอ้าต์พุต หน่วยเป็นโวลต์

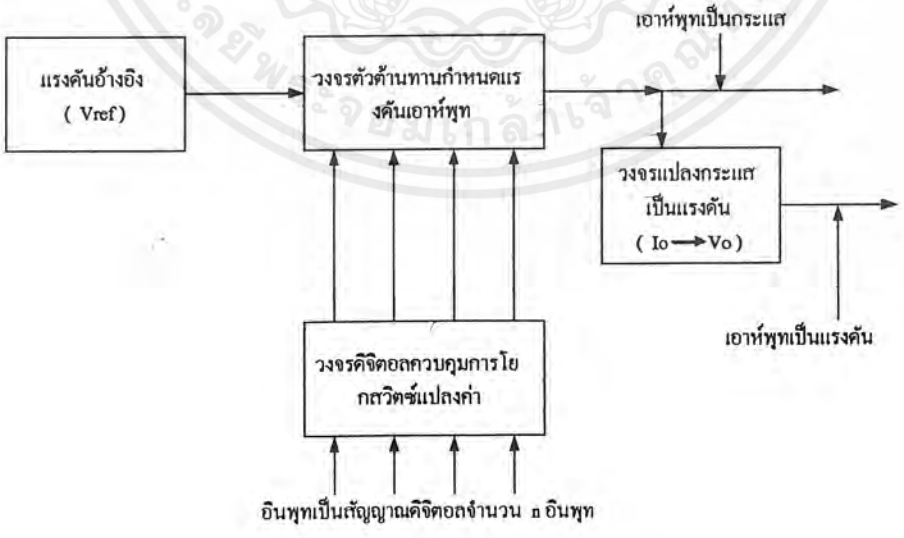
D คือ ค่าเลขฐานสิบที่แปลงมาจากเลขฐานสองจากลอจิกอินพุต

ค่าเต็มสเกลของดีเอซีขนาด n บิต หาได้จากสูตร

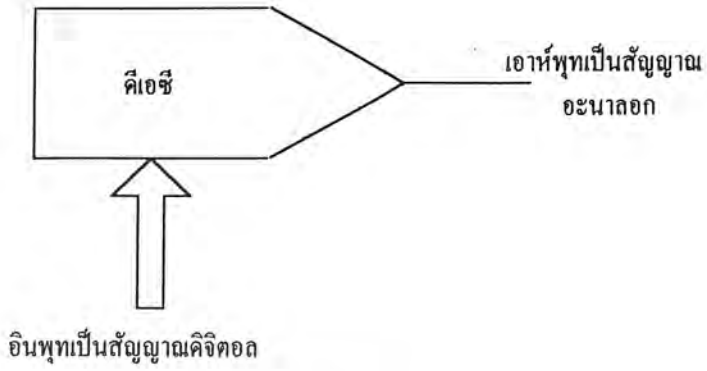
แรงดันเอ้าต์พุตเต็มสเกล = ความละเอียดของแรงดันคูณด้วย $(2^n - 1)$

กำหนดให้ n คือค่าจำนวนบิตทางด้านอินพุตของดีเอซี

2.4.2หลักการทำงานของดีเอซี

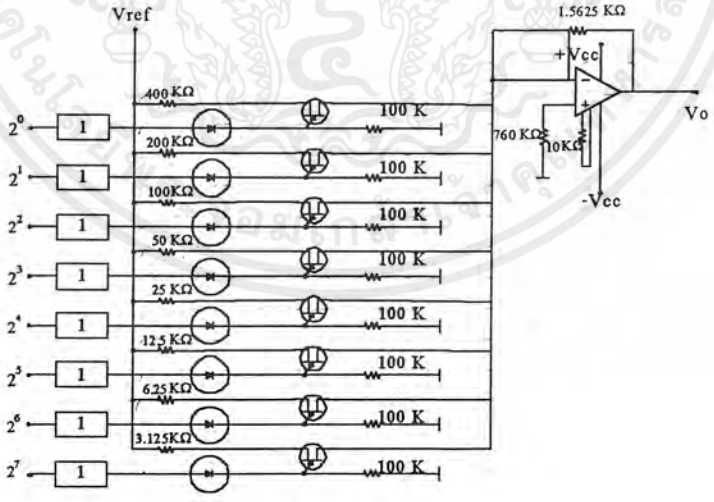


รูปที่ 2.11 แสดงแผนผังของวงจรดีเอซีและสัญลักษณ์



รูปที่ 2.12 แสดงแผนผังของวงจรดีเอซีและสัญลักษณ์

เพื่ออำนวยความสะดวกในการรับส่งข้อมูลจากสายข้อมูลของระบบคอมพิวเตอร์ขนาด 8 บิตหรือมากกว่านั้น จึงจำเป็นต้องให้ ดีเอซี มีจำนวนอินพุตมากขึ้นดังแสดงตัวอย่างวงจรพื้นฐานในรูป 2.12 โดยใช้ตัวต้านทานของบิตต่ำสุดเป็น 400 กิโลโอห์ม ส่วนบิตสูงสุดใช้ตัวต้านทานขนาด 3.125 กิโลโอห์ม และใช้ทรานซิสเตอร์เฟตเป็นอะนาลอกสวิตช์ ร่วมกับวงจรขยายออปแอมป์ เบอร์ 741



รูปที่ 2.13 แสดงวงจรดีเอซีขนาด 8 บิต

ขา 5 ถึงขา 12 เป็นขาอินพุท ต่อกับสัญญาณดิจิตอลชนิดทีทีแอลหรือซีมอส ขา 5 เป็นบิตนัยสำคัญสูงสุด ขา 12 เป็นบิตนัยสำคัญต่ำสุด ลอจิก "0" จะต้องมีค่าสูงไม่เกิน 0.8 โวลต์ "1" จะต้องมีค่าไม่ต่ำกว่า +0.2 โวลต์

ขา 1 เป็นขาควบคุมการทริก (V_{TH} = threshold voltage) กำหนดให้ $V_{TH} = V_{LC} + 14$ โวลต์ V_{LC} คือค่าแรงดันที่ขา 1

ขา 2 และขา 4 เป็นขาเอาห์พุท ขา 4 เป็นขาที่ I_{out} ไหลเมื่อบิตใดๆ ทางด้านอินพุทถูกควบคุมด้วยลอจิก "1" ขา 2 เป็นขาที่ $I^{\wedge}_{out}$ ไหล ขณะที่บิตใดๆ ทางด้านอินพุทถูกควบคุมด้วยลอจิก "0"

ความละเอียดของกระแส $= (V_{ref}/R_{fef})\,(1/2^{n})$

I_{out} = ความละเอียดของกระแสคูรด้วยค่า D

I_{FS} = ความละเอียดของกระแสคูณด้วย 255

I_{FS} หมายถึงกระแสเต็มสเกลเมื่ออินพุทเป็น 11111111 = 255 = D

$I^{\wedge}_{out} = I_{FS} - I_{out}$

จากรูปวงจรที่2.13หาความละเอียดของแรงดันได้จากสูตร

ความละเอียดของแรงดัน $= (V_{ref} / R_{ref}) \times R_f (1/2^{n})$

V_{out} = ความละเอียดของแรงดันคูรด้วยค่า D

$= I_{out} \times R_f$

แรงดันเอาห์พุทขณะใช้เอาห์พุทแบบคู่

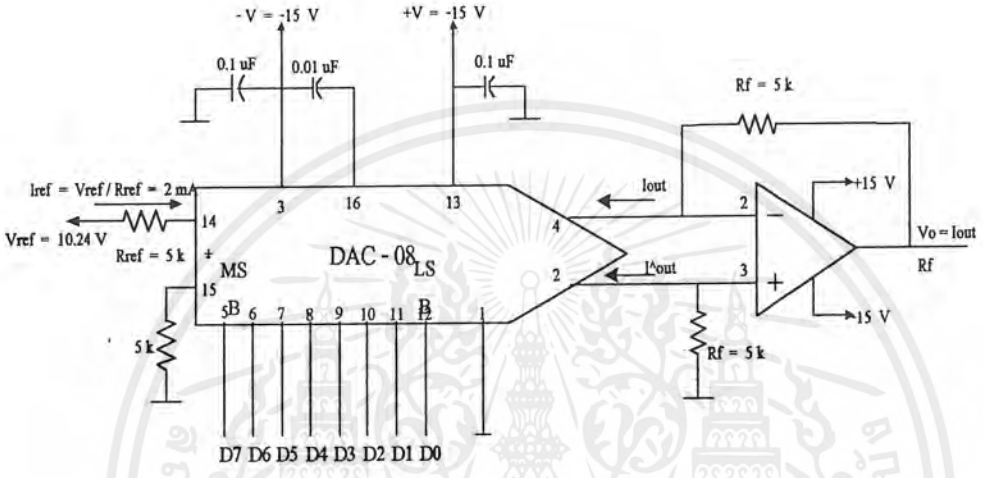
ไอซี DAC –08 ต่อเอาห์พุท ดังแสดงในรูปที่ 2.14 เขียนสมการเอาห์พุทได้ดังนี้

$Vo = (I_{out} - I^{\wedge}_{out})\, R_f$

I_{out} = ทำให้แรงดันเอาห์พุทมีศักย์เป็นบวก

$I^{\wedge}_{out}$ = ทำให้แรงดันเอาห์พุทมีศักย์เป็นลบ

V_{ref} มีค่าเป็น 10.24 โวลต์ มีค่า 5 กิโลโอห์ม ทำให้ได้ค่ากระแส I_{ref} เท่ากับ 2.048 มิลลิแอมแปร์ ค่ากระแสที่เปลี่ยนไปต่อบิตเท่ากับ 8 ไมโครแอมป์แปร์ ได้แรงดันเอาท์พุท 40 มิลลิโวลต์ต่อบิต ซึ่งค่าต่างๆนี้แสดงไว้ในรูปที่2.14



รูปที่ 2.14 แสดงการต่อ DAC –08 แบบเอ้าต์พุทคู่

ดีเอซีที่ใช้กับไมโครโปรเซสเซอร์ เนื่องจากปัจจุบันได้มีผู้นำเอาดีเอซีมาใช้ร่วมกับระบบไมโครโปรเซสเซอร์หรือระบบคอมพิวเตอร์แพร่หลายมากขึ้น จึงมีบริษัทผู้ผลิตทำการผลิตไอซีที่สามารถใช้ต่อร่วมกับระบบได้โดยตรง ดังจะได้กล่าวถึงวงจรพื้นฐานต่อไปนี้

ดีเอซีที่สามารถนำมาต่อร่วมกับระบบไมโครโปรเซสเซอร์ได้ จะต้องมีคุณสมบัติดังนี้

1. เขียนข้อมูลจากสายข้อมูล (data bus) เข้าไปเก็บไว้ในรีจิสเตอร์ภายในอินพุตของดีเอซีได้ เมื่อไอซีดีเอนั้น ๆ ถูกเรียกใช้งาน
2. สามารถปลดตัวเองออกจากระบบได้ และยังคงมีข้อมูลเดิมครั้งหลังสุดค้างอยู่ที่บัฟเฟอร์ทางอินพุตของดีเอซีภายหลังจากการรับข้อมูลจากซีพียูของระบบ

2.5 สัญญาณต่าง ๆ บนสล๊อตของ IBM/PC

ภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มวงจรอินเทอร์เฟสเข้าในภายหลังได้ โดยผ่านทางสล๊อตที่อยู่บนเมนบอร์ด (Main Board) สำหรับสล็อตบนเมนบอร์ดนี้จะมีจำนวน 5

สล๊อต(สำหรับใน IBM PC/XT จะมี 3 สล๊อต : จะกล่าวในภายหลัง) ซึ่งแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขาแบ่งออกเป็น 2ข้าง ๆ ละ31 ขา ส่วนการเรียกตำแหน่งขาของสล็อตเหล่านี้จะขึ้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล๊อต โดยขาที่อยู่ทางด้านซ้ายสล็อตจะเรียกโดยใช้อักษร 'B' นำหน้าเลขตำแหน่งของขา เช่น B16 ก็คือขาทางด้านซ้ายของสล็อต ขาที่ 16 นับจากทางด้านท้ายของเครื่อง) ส่วนขาที่อยู่ทางคด้านขวาของสล๊อตจะเรียกโดยใช้อักษร 'A' นำหน้าเลขตำแหน่งของขา เช่น ขา A24 ก็คือขาทางด้านขวาของสล๊อตขาที่ 24 (นับจากทางด่านท้ายของเครื่อง)

แต่ละขาของสล๊อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่าง ๆ บนเมนบอร์ด ทำให้การสร้างวงจรอินเตอร์เฟสกับ IBM/PC สามารถทำได้สะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล๊อตเหล่านี้จะประกอบไปด้วย เส้นสัญญาณของบัสแอดเดรส (Address Bus), บัสข้อมูล (Data Bus), บัสควบคุมสำหรับการเขียน/อ่านข้อมูลจากหน่วยความจำ หรือพอร์ตI/O, เส้นสัญญาณสำหรับการขออินเตอร์เฟส, เส้นสัญญาณสำหรับขอ DMA

นอกจากเส้นสัญญาณเหล่านี้แล้ว สล๊อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่าง ๆ ที่ใช้ในระบบอีกด้วย คือ +5Vdc, -5Vdc, +12Vdc และ –12Vdc

2.5.1รายละเอียดเกี่ยวกับสัญญาณต่าง ๆ

OSC (Oscillator ; ขา B30)

ขานี้เป็นเอาต์พุทที่เชื่อมต่อกับสัญญาณคล็อกที่มีความถี่สูงสุดบนเมนบอร์ด คือ 14.31818 MHz ซึ่งมีคาบเวลาประมาณ 70 nanosec และมี Duty Cycle (ช่วงเวลาใน 1 คาบที่สัญญาณคล็อกมีลอจิกเป็น " 1 " หารด้วยคาบเวลาทั้งหมด) ประมาณ 50% สัญญาณคล็อกอื่น ๆ ของระบบ เช่น คล็อกที่ป้อนให้กับ 8088 หรือ ชิพซัพพอร์ตต่าง ๆ นั้นถูกสร้างขึ้นโดยการหารสัญญาณคล็อกนี้ อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงในการใช้งานสัญญาณ OSC ก็คือ สัญญาณนี้จะไม่ Synchronize กับสัญญาณอื่น ๆ บนบัสของระบบ ดังนั้นจึงไม่ควรที่จะนำสัญญาณจากขา OSC นี้ไปใช้เป็นสัญญาณคล็อกสำหรับวงจรภายนอกอื่น ๆ ที่ทำงานร่วมกับระบบ

CLK (Clock ; ขา B20)

ขาสัญญาณนีเป็นเอาต์พุท ซึ่งต่อกับสัญญาณคล็อกที่ถูกสร้างขึ้นโดยการหารสัญญาณ OSC ด้วย 3 ทำให้ได้ความถี่ประมาณ 4.77 MHz (14.31818 MHz / 3) หรือ มีช่วงเวลาใน 1 คาบ เท่ากับ 210 naonsec (1/4.77 MHz) สำหรับค่า Duty Cycle ของสัญญาณนี้จะมีค่าประมาณ 1 / 3 คือ ใน 1 คาบจะมีช่วงเวลาที่เป็นลอจิก ' 1 ' เท่ากับ 1 / 3 ของคาบเวลาทั้งหมด หรือประมาณ 70 nanosec และช่วงเวลาที่เป็นลอจิก ' 0 ' เท่ากับ

2 / 3 ของคาบเวลาทั้งหมด หรือประมาณ 140 nanosec

RESET DRV (ขา B2)

ขาสัญญาณนี้เป็นเอาร์พุท ซึ่งแอคทีฟ (ลอจิก ' 1 ') ในช่วงที่เราเริ่มจ่ายไฟให้กับระบบ และจะยังคงแอคทีฟไปจนกว่าระบบต่าง ๆ ภายใน IBM/PC จะพร้อมที่จะทำงานได้จากนั้นสัญญาณนี้จะเปลี่ยนกลับเป็นลอจิก ' 0 ' นอกจากนี้ในระหว่างการทำงานของ IBM/PC ถ้าระดับแรงดันของอหล่งจ่ายไฟตกลง สัญญาณนี้ก็จะถูกทำให้แอกทีฟเช่นกัน โดยทั่วไปแล้วสัญญาณนี้จะถูกนำไปใช้ในการรีเซ็ทวงจรอินเตอร์เฟสหรืออุปกรณ์ I/O ต่าง ๆ ในช่วงที่เริ่มจ่ายไฟให้กับระบบ ซึ่งจะเป็นการทำให้วงจรหรืออุปกรณ์เหล่านี้ถูกปรับให้อยู่ในสถาวะที่แน่นอน ก่อนที่จะเริ่มต้นการทำงานในระบบ (สภาวะนี้เป็นสภาวะที่เราทราบ และต้องการให้วงจรทำงานในขณะที่ระบบถูกรีเซ็ต)

A0-A19 Address Bus :(ขา A31-A12):

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาต์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำหรืออุปกรณ์ 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 8088จะถูกตัดออกจากระบบ)

จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสของหน่วยความจำได้ถึง 1 Mbyte แต่อย่างไรก็ตามจะมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว คือแอดเดรสของหน่วยความจำ RAM

สำหรับการอ้างแอดเดรสของพอร์ต I/O นั้น จะใช้เส้นแอดเดรสเพียง 16 เส้น คือ A0-A15 ซึ่งจะทำให้อ้างแอดเดรสของพอร์ตได้ 64K พอร์ต โดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือ A16-A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสของพอร์ตเพียง 10 เส้น คือจาก A0-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ใช้ช่วง 0200H จนถึง 03FFH เท่านั้น

D0-D7 (Data Bus : ขา A9-A2):

ขาสัญญาณนี้จะเป็นแบบ Bi-Directional ซึ่งต่อกับบัสข้อมูลของระบบ เพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ต I / O กับ IBM/PC โดยบิท D0 จะมีนัยสำคัญต่ำสุดและบิท D7 จะมีนัยสำคัญสูงสุด

สำหรับในบัสไซเคิลของการเขียนข้อมูลที่สร้างขึ้นโดย 8088 นั้น ข้อมูลจะถูกส่งออกมาบนบัสข้อมูล ก่อนที่สัญญาณ IOW (ในกรณีที่ต้องการส่งข้อมูลให้กับพอร์ต) หรือ MEMW (ในกรณีที่ต้องส่งข้อมูลให้กับหน่วยความจำ) จะเปลี่ยนจากลอจิก ' 0 ' เป็นลอจิก ' 1 ' (ขอบขาขึ้น) ซึ่งโดนทั้ง

ไปขอบขาขึ้นของสัญญาณ IOW หรือ MEMW นี้ จะถูกใช้เพื่อสั่งให้พอร์ต I/O หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

สำหรับในบัสไซเคิลของการอ่านข้อมูลที่สร้างขึ้นโดย 8088 นั้นพอร์ต I/O หรือหน่วยความจำที่ถูกอ้างถึงจะต้องส่งข้อมูลออกมาบนบัสข้อมูล ก่อนที่สัญญาณ IOR (ในกรณีที่ต้องการอ่านข้อมูลจากพอร์ต)

ALE (Address Latch Enable : ขา B28) :

ขาสัญญาณนี้เป็นสัญญาณเอาต์พุทที่ 8288 Bus Controller สร้างขึ้นเพื่อใช้สำหรับแสดงการเริ่มต้นของไซเคิล และแสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ 8088 ต้องการจะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดราแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยน
จากลอจิก ' 1 ' เป็น ' 0 ' เมื่อค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสเรียบร้อยแล้ว ดังนั้นขอบขาลงของสัญญาณ ALE นี้จะถูกใช้ในการแลทช์ค่าแอดเดรสจากบัสแอดเดรส/ข้อมูล (Address/Data Bus ; AD0AD7) ของ 8088 ทำให้สามารถแยกค่าแอดเดรส (a0-A19) และข้อมูล (A0-A7) ออกจากกันได้ อย่างไรก็ตามสัญญาณ ALE จะแอคทีฟเฉพาะในบัสไซเคิลที่สร้างขึ้นโดย 8088 เท่านั้นโดยจะไม่แอคทีฟในระหว่างขบวนการ DMA

I/O Chck(I/O Channel Check ; ขา A1) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้ในการแสดงความมิดพลาดเกี่ยวกับพาริตี้ ที่เกิดขึ้นในการทำงานของวงจรอินเทอร์เฟสหรืออุปกรณ์ I/O เมื่อขาสัญญาณนี้ได้รับลอจิก ' 0 ' จะทำให้ 8088 ถูกอินเทอร์รัพท์แบบ Non-Maskable(NMI) อย่างไรก็ตามเราสามารถที่จะกำหนดให้วงจรภายในของ IBM/PC ทำการอินเทอร์รัพท์ (เมื่อได้รับสัญญาณ I/O ChCK) หรือไม่ก็ได้ โดยการกำหนดลอจิกของบิทข้อมูลของพอร์ตที่ควบคุมการขออินเทอร์รัพท์แบบ NMI คือบิท D7 ของพอร์ต 00A0H ในกรณีที่บิท D7 ของพอร์ต 00A0H ถูกเซ็ทเป็น ' 1 ' ก็จะทำให้วงจรภายนอกของอินเทอร์รัพท์แบบ NMI ได้ (Enable) แต่ถ้าบิท D7 ของพอร์ต 00A0H ถูกเซ็ทเป็น ' 0 ' ดิสเอเบิล (Disable) การขออินเทอร์รัพท์แบบ NMI ดังนี้

Enable : ใช้คำสั่ง OUT ส่งข้อมูล 80H ไปยังพอร์ต 00A0H

Disable : ใช้คำสั่ง OUT ส่งข้อมูล 00H ไปยังพอร์ต 00A0H

และเนื่องจากยังมีอุปกรณ์อื่นที่สามารถขออินเทอร์รัพท์แบบ NMI ได้อีก ดังนั้นซอฟท์แวร์ที่ใช้งานจะต้องสามารถสอบว่าการขออินเทอร์รัพท์นั้นเกิดขึ้นจากแหล่งใดได้ด้วย

I/O CHRDY (I/O Channel Ready : ขา A10) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้เพิ่มช่วงเวลาในบัสไซเคิลในกรณีที่อุปกรณ์ I/O หรือหน่วยความจำที่เกี่ยวข้องกับขบวนการในบัสไซเคิลที่เกิดขึ้นนั้น ไม่สามารถทำงานทันตามช่วงเวลาปกติ

ของบัสไซเคิลนั้น ๆ ได้ ช่วงเวลาของบัสไซเคิลที่เกี่ยวกับหน่วยความจำใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 4 ลูกหรือ 840 nanosec ในขณะที่บัสไซเคิลที่เกี่ยวกับ I/O จะใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 5 ลูกหรือ 1.05 usec

เมื่ออุปกรณ์ I/O หรือหน่วยความจำต้องการที่จะเพิ่มช่วงเวลาในบัสไซเคิลให้นานขึ้นอีกนั้น จะสามารถทำได้โดยการป้อนลอจิก ' 0 ' ให้กับขา I/O CHRDY ในช่วงเวลาที่ I/O หรือหน่วยความจำที่ถูกกำหนดนั้น ได้รับสัญญาณจากการดีโค้ดแอดเดรสและสัญญาณMEMR,MEMW,IOR หรือ IOW แอคทีฟ สำหรับรายละเอียดเกี่ยวกับการเพิ่มช่วงเวลาในบัส

IRQ2-IRQ7 Interrupt Request 2-7 : ขา B4 และ B25 –B21) :

ขาสัญญาณทั้ง 6 นี้เป็นขาอินพุทที่ใช้สำหรับการขออินเทอร์รัพท์จาก 8088 โดยสัญญาณเหล่านี้จะต่อเข้ากับ 8259A บนเมนบอร์ดโดยตรง โปรแกรมในส่วน BIOS ของ IBM/PC จะทำการโปรแกรม 8259A ให้ IRQ2 มีลำดับความสำคัญสูงสุด (Highest Priority) และ IRQ7 มีลำดับความสำคัญที่ต่ำสุด ในกรณีที่มีการขออินเทอร์รัพท์เกิดขึ้น คือระดับลอจิกที่ขา IRQ ขาใดขาหนึ่งถูกเปลี่ยนจากลอจิก ' 0 ' เป็นลอจิก ' 1 ' (ขอบขาขึ้น) 8259A ก็จะทำการส่งสัญญาณ INT ให้กับ 8088 เพื่อทำการขออินเทอร์รัพท์

สิ่งสำคัญในการอินเทอร์รัพท์โดยผ่านทาง IRQ2-IRQ7 นี้ ก็คืออุปกรณ์ที่ทำการขออินเทอร์รัพท์โดยผ่านทาง IRQ ขาใดก็จะต้องรักษาระดับสัญญาณที่ขา IRQ นั้น ให้แอคทีฟ (ลอจิก ' 1 ') อยู่จนกว่าจะได้รับสัญญาณ INTA (Interrupt Acknowledge) จาก 8088 เสียก่อนถ้าไม่เช่นนั้นการขออินเทอร์รัพท์จะถูกยกเลิก และอินเทอร์รัพท์ Level 7 (IRQ7) ก็จะถูกสร้างขึ้นโดยอัตโนมัต ไม่ว่าการขออินเทอร์รพรท์ที่ถูกยกเลิกนั้นจะเป็นการขออินเทอร์รัพท์ใน Level หรือขาใด

แต่อย่างไรก็ตามสัญญาณ INTA นี้จะไม่ถูกต่อออกมาที่ขาของสล็อตด้วย ดังนั้นโปรแกรมที่ทำการตอบสนองต่อการขออินเทอร์รพท์ (Interrupt Service Routine) จะต้องทำการรีเซ็ทสัญญาณ IRQ เองโดยใช้คำสั่ง OUT ไปยังพอร์ต I/O ที่เกี่ยวข้อง

IOR (I/O Read ; ขา B14) :

ขาสัญญาณนี้เป็นเอาต์พุทแอคทีฟที่ลอจิก ' 0 ' ที่สร้างขึ้นโดย 8288 Bus Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ต I/O เพื่อให้พอร์ต I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูล โดยข้อมูลจะต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ IOR ประมาณ 30 nanosec เพื่อให้มั่นใจได้ว่า 8088 สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการ DMA 8237A-5 DMA Controller จะทำการสร้างสัญญาณ IOR เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำ (แทนที่จะเป็นแอดเดรสของพอร์ต I/O) ที่พอร์ต I/O ที่ขอ DMA ต้องการจะนำข้อมูลไป

เก็บ การที่พอร์ตใดจะส่งข้อมูลออกมาบนบัสข้อมูลนั้น อาศัยสัญญาณ DACK จาก DMA Controller เป็นตัวกำหนด เช่นกรณีที่สัญญาณ DACK1 แอคทีฟก็จะแสดงว่าพอร์ต I/O ที่จะต้องส่งข้อมูลออกมาบนบัสข้อมูลก็คือพอร์ต I/O ที่ขอ DMA ผ่านทางแชนแนลที่ 1 (DRQ1) เป็นต้น

IOW (I/O Write : ขา B13) :

ขาสัญญาณนี้เป็นเอาต์พุทแอคทีฟที่ลอจิก ' 0 ' ซึ่งถูกสร้างขึ้นโดย 8288 Bus Controller เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ต I/O เพื่อให้พอร์ต I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมมูลไปเก็บไว้อย่างไรก็ตามเนื่องจากในช่วงเวลาที่สัญญาณ IOW นี้แอคทีฟ (ลอจิก ' 0 ') นั้นข้อมูลบัสอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ IOW แทนขอบขาลงในการทำขบวนการ DMA นั้น DMA-Controller จะทำการสร้างสัญญาณ IOW เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ต I/O ที่ขอ DMA ต้องกหารจะอ่านข้อมูล

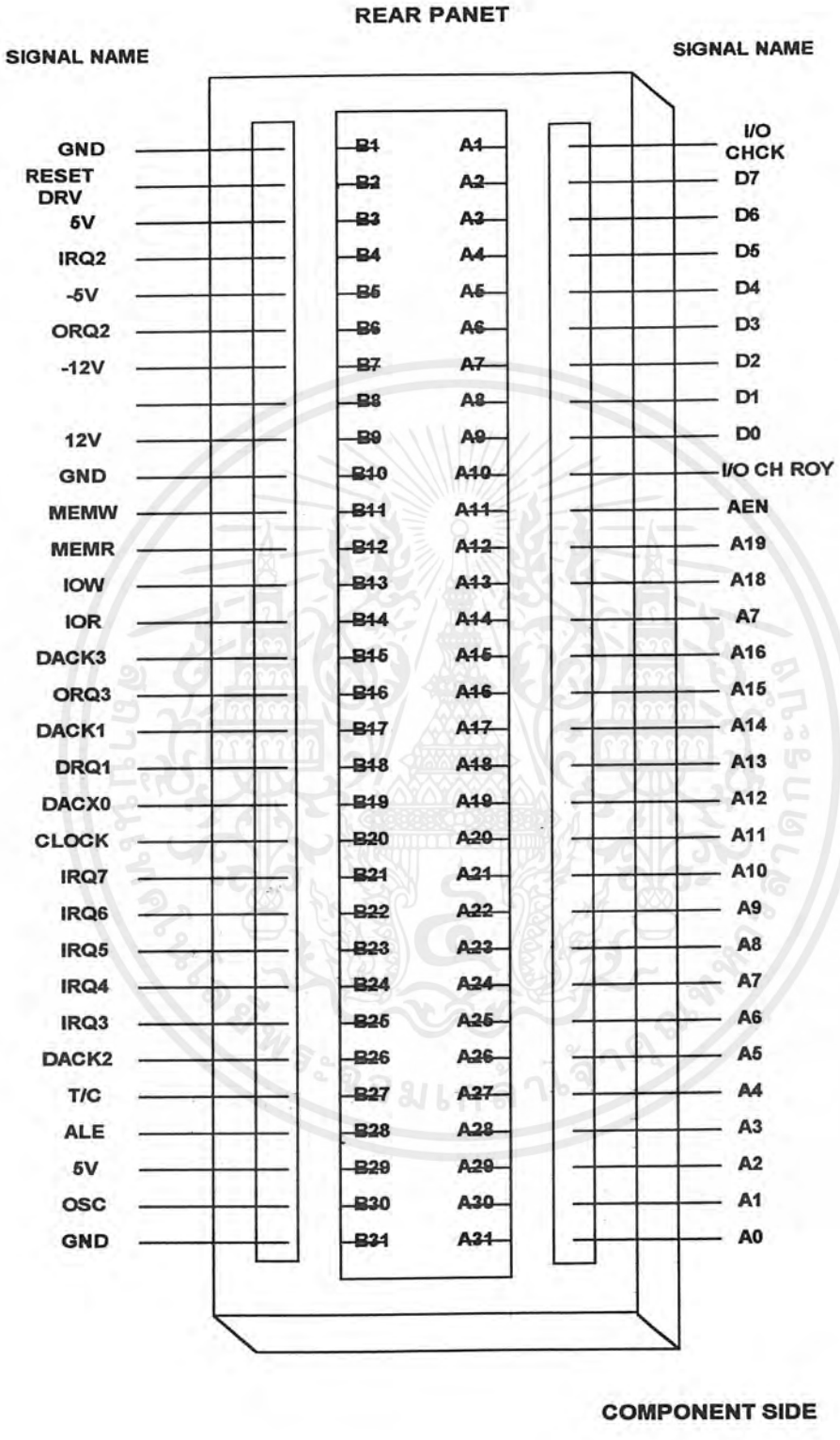
MEMW (Memory Write ; ขา B11) :

ขานี้เป็นเอาต์พุทแอคทีฟที่ลอจิก ' 0 ' ซึ่ง 8288 Bus Controller สร้างขึ้นในระหว่างบัสไซเคิลในการเขียนข้อมูลลงในหน่วยความจำของ 8088 สัญญาณ MEMW นี้จะถูกส่งออกมาเพื่อให้หน่วยความจำที่แอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการรับข้มูล้อยู่บนบัสข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ โดยทั่วไปหน่วยความจำจะรับข้อมูลในช่วงลงขอบขาขึ้นของสัญญาณ MEMW สำหรับในระหว่างขบวนการ DMA นั้น 8237A-5 DMA-Controller จะทำการควบคุมบัสต่าง ๆ ของระบบแทน 8088 และสัญญาณ MEMQ จะถูกใช้ในบัสไซเคิลของการเขียนข้อมูลลงในหน่วยความจำ (ข้อมูลถูกส่งจากอุปกรณ์ I/O ไปให้กับหน่วยความจำ)

MEMR (Memory Read ; ขา B12) :

ขานี้เนเอาต์พุทจาก 8288 ซึ่งสัญญาณนี้จะแอคทีฟ (ลอจิก ' 0 ' ในระหว่างบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำของ8088เพื่อให้หน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการส่งข้อมูลออกมาบนบัสข้อมูล โดยหน่วยความจำนั้นจะต้องส่งข้อมูลออกมาในช่วงเวลา 30 nanosec ก่อนที่สัญญาณ MEMW จะกลับเป็นลอจิก ' 1 ' ทั้งนี้ก็เพื่อให้ 8088 ได้รับข้อมูลที่ถูกต้อง

สำหรับในระหว่างขบวนการ DMA นั้น DMA-Controller จะควบคุมบัสต่าง ๆ ของระบบแทน 8088 และสัญญาณ MEMW จะถูกใช้ในบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำ (ข้อมูลถูกส่งหน่วยความจำไปให้กับอุปกรณ์ I/O)



รูปที่ 2.15 แสดง SLOT ISA

DRQ1- DRQ3 (DMQ Request 1-3 ; ขา B18 ,B6 และ ขา B16) :

ขาสัญญาณทั้งสามนี้เป็นสัญญาณอินพุทแอคทีฟที่ลอจิก ' 1 ' ซึ่งอุปกรณ์ภายนอกสามารถใช้ในการขอ DMA จากระบบ โดยการป้อนระดับสัญญาณลอจิก ' 1 ' ให้กับขา DRQ ขาใดขาหนึ่ง (ขา DRQ ทั้งสามนี้จะต่อเข้ากับ DRQ1-DRQ3 ของ 8237A-5)

เมื่อ 8237A-5 ได้รับสัญญาณนี้แล้วก็จะตรวจสอบว่ามีการขอ DMA ในแชนแนลที่มีลำดับความสำคัญ (Priority) สูงกว่าหรือไม่ ถ้าไม่มีก็จะทำการขอ DMA จาก8088 และตอบรับการขอ DMA จากอุปกรณ์ภายนอก (สัญญาณ DACK ของแชนแนลที่ขอ DMAจะแอคทีฟ) แต่ถ้ามี 8237A-5 ก็จะทำการขอ DMA ให้กับแชนแนลที่มีลำดับความสำคัญสูงกว่าก่อนแล้วจึงทำการขอ DMA ให้กับแชนแนลที่มีลำดับต่ำกว่า ภายใน ROM BIOS ของ IBM/PC จะโปรแกรม
8237A-5 ให้ DRQ1 มีลำดับความสำคัญสูงสุดและDRQ3 มีลำดับความสำคัญต่ำสุด ดังนั้นถ้ามีการของ DMA ของอุปกรณ์ภายนอกผ่านทางแชนแนลที่ (DRQ1) และแชนแนลที่2 (DRQ2) 8237A-5 ก็จะทำการขอ DMA ให้กับแชนแนลที่ 1 ก่อน จากนั้นเมื่อเสร็จจากขบวนการ DMA ของแชลแนลที่ 1 แล้ว จึงจะทำการของ DMA ให้กับแชนแนลที่ 2

อย่างไรก็ตาม 8237A-5 ยังมีแชนแนลสำหรับการขอ DMA อยู่อีก 1 แชนแนลคือ แชนแนลที่ 0 (DMA0) ซึ่งในความเป็นจริงแล้วแชนแนลนี้จะมีลำดับความสำคัญที่สูงกว่าแชนแนลที่ 1 แต่จะไม่ถูกต่อออกมายังขาของสล็อต เนื่องจาก IBM/PC จะใช้แชนแนลที่ 0 นี้ในการรีเฟรชหน่วยความจำที่เป็น dynamic RAM

ในการขอ DMA นั้นสัญญาณ DRQ นี้จะต้องแอคทีฟอยู่ในช่วงระยะเวลาหนึ่งเท่านั้น ถ้าสัญญาณนี้แอคทีฟอยู่นานเกินไป จะทำให้เกิดขบวนการ DMA ขึ้นมากกว่า 1 ขบวนการได้สำหรับวงจรที่ขอ DMA โดยทั่งไปแล้วจะใช้สัญญาณตอบรับการขอ DMA หรือสัญญาณ DACK ของแชนแนลที่ขอ DMA นั้น ในการเซ็ทสัญญาณ DRQ เช่นอุปกรณ์ภายนอกที่ขอ DMA ผ่านทางแชนแนลที่ 1 (DRQ1) ก็จะคอยตรวจสอบการตอบรับการขอ DMA จากสัญญาณ DACK แชนแนลที่ 1 (DACK1) เมื่อได้รับสัญญาณจาก DACK1 แล้ว ก็จะรีเซ็ทสัญญาณ DRQ1 (เปลี่ยนจากลอจิก ' 1 ' เป็น ' 0 ')

DACK0-DACK3 (DMA Acknowledge 0-3 : ขา B19,B17,B26 และ B15) :

สัญญาณทั้ง 4 นี้เป็นเอาต์พุทแอคทีฟที่ลอจิก ' 0 ' ซึ่ง 8237A-5 สร้างขึ้นเพื่อแสดงให้วงจรภายนอกที่ขอ DMA ทราบว่าการขอ DMA นั้นได้รับการตอบสนองแล้ว และ
8237A-5 จะเข้าสู่ขบวนการ DMA เพื่อให้การส่งผ่านข้อมูลระหว่างอุปกรณ์ I/O ที่ขอ DMA กับหน่วยความจำเกิดขึ้นได้โดยตรง (คือไม่ต้องผ่าน 8088) โดยสัญญาณ DACK นี้จะแอคทีฟในแชน

แนลใดก็ขึ้นอยู่กับว่าขบวนการ DAM ที่จะเกิดขึ้นนั้น เป็นการตอบสนองต่อการขอ DMA ในแชนแนลที่ 2 (DRQ2) สัญญาณ DACK2 ก็จะแอคทีฟ เป็นต้น

ดังที่ได้กล่าวแล้วว่าสัญญาณ DRQ0 นั้น จะไม่ถูกต่อออกมายังขาของสล็อต ดังนั้นวงจรอินเทอร์เฟสจึงไม่สามารถขอ DMA ผ่านทางแชนแนล 0 ได้ แต่สัญญาณ DACK0 จะถูกต่อออกมาบังสล็อตด้วย (ขา B19) ทั้งนี้ก็เพื่อที่จะแสดงให้วงจรอินเทอร์เฟสต่าง ๆ ทราบว่าขบวนการ DMA ที่เกิดขึ้นในช่วงเวลาที่ DACK แอคทีฟนั้น เป็นขบวนการที่ใช้สำหรับการรีเฟรชหน่วยความจำที่เป็น Dynamic RAM ซึ่งวงจรอินเทอร์เฟสที่ใช้หน่วยความจำประเภทนี้สามารถจะนำไปใช้ในการรีเฟรช Dynamic RAM ที่อยู่ในวงจรได้

โดยที่การรีเฟรชหน่วยความจำนั้นจะเกิดขึ้นในทุก ๆ 15.12 usec หรือ ทุก ๆ 72 คล็อก ดังนั้นสัญญาณ DACK0 นี้ก็จะแอคทีฟในทุก ๆ 15.12 usec ด้วย

AEN (Address Enable ; ขา A11) :

สัญญาณนี้เป็นเอาต์พุทที่ใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ 8255 Bus Controller และจะใช้ดิสเอเบิลพอร์ต I/O ต่าง ๆ ที่ไม่เกี่ยวข้องกับขบวนการ DMA ที่เกิดขึ้น ที่จำเป็นต้องทำเช่นนี้ก็เพราะในระหว่างขบวนการ DMA นั้น 8237A-5 จะส่งแอดเดรสของหน่วยความจำออกมาบนบัสแอดเดรส และจะทำให้สัญญาณ IOR หรือ IOW แอคทีฟด้วย ดังนั้นถ้าไม่ทำการดิสเอเบิลพอร์ต I/O ที่ไม่เกี่ยวข้องไว้ ก็อาจจะทำให้พอร์ต I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรส (ซึ่งเป็นแอดเดรสของหน่วยความจำ) นั้น ทำการอ่านหรือส่งข้อมูลออกมาบนบัสข้อมูลทำให้เกิดความผิดพลาดขึ้นได้

T/C (Terminal Count : ขา B27) :

สัญญาณนี้ถูกสร้างขึ้นจากการนำเอาสัญญาณเอาต์พุทที่ขา EOP ของ 8237A-5 มากลับลอจิก (โดยใช้เกท Inverter) ทำให้สัญญาณ T/C นี้แอคทีฟที่ลอจิก ' 1 '

สำหรับสัญญาณนี้จะแอคทีฟเมื่อจำนวนไบท์ในการส่งข้อมูลของขบวนการ DMA ในแชนแนลใดแชนแนลหนึ่ง ครบตามจำนวนที่กำหนดไว้ โดยทั่วไปแล้วสัญญาณที่จะถูกใช้ในการสิ้นสุดขบวนการ DMA ที่ทำการส่งผ่านข้อมูลเป็นบล็อก เนื่องจากสัญญาณนี้จะแอคทีฟโดยไม่แสดงว่าเป็นสัญญาณของแชนแนลใด ดังนั้นจึงต้องทำการสัญญาณ T/C นี้ผ่านเกท Inverter แล้วนำไป OR กับสัญญาณ DACK เพื่อให้สามารถทราบได้ว่า สัญญาณ T/C ที่เกิดขึ้นนั้นเป็นสัญญาณของแชนแนลใด

บัสของแหล่งจ่ายไฟของระบบ

+5 Vdc (ขา B3 และ B29) :

ขาทั้งสองนี้ต่อกับแหล่างจ่ายไฟ DC +5V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) 5% คืออยู่ในช่วง +4.75 ถึง +5.25 Vdc

+ 12 Vdc (ขา B9) :

ขาทั้งสองนี้ต่อกับแหล่างจ่ายไฟ DC +12V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) 15% คืออยู่ในช่วง +11.4 ถึง +12.6 Vdc

- 5 Vdc (ขา B5) :

ขาทั้งสองนี้ต่อกับแหล่างจ่ายไฟ DC -5V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) 10% คืออยู่ในช่วง –5.5 ถึง 4.5 Vdc

- 12 Vdc (ขา B7) :

ขาทั้งสองนี้ต่อกับแหล่างจ่ายไฟ DC -12V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) 10% คืออยู่ในช่วง –13.2 ถึง –10.8 Vdc

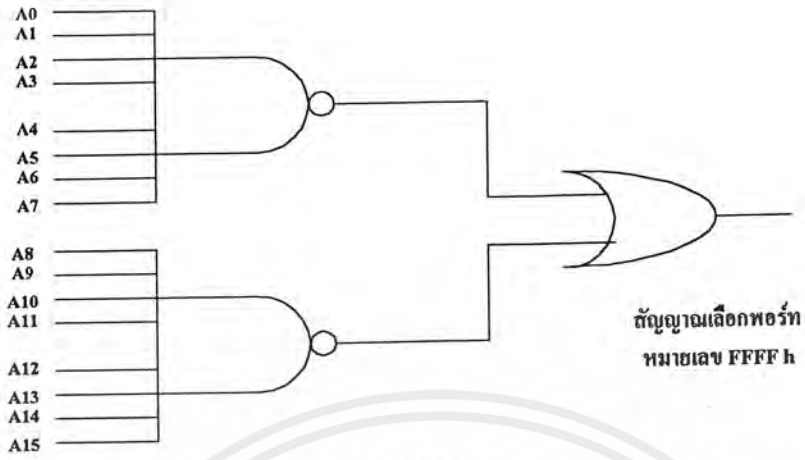
GND (ขา B1,B10 และ B31) :

ขาทั้งสามนี้จะต่อเข้ากับกราวด์ (Ground) ของระบบ

2.6 ตำแหน่งของพอร์ต

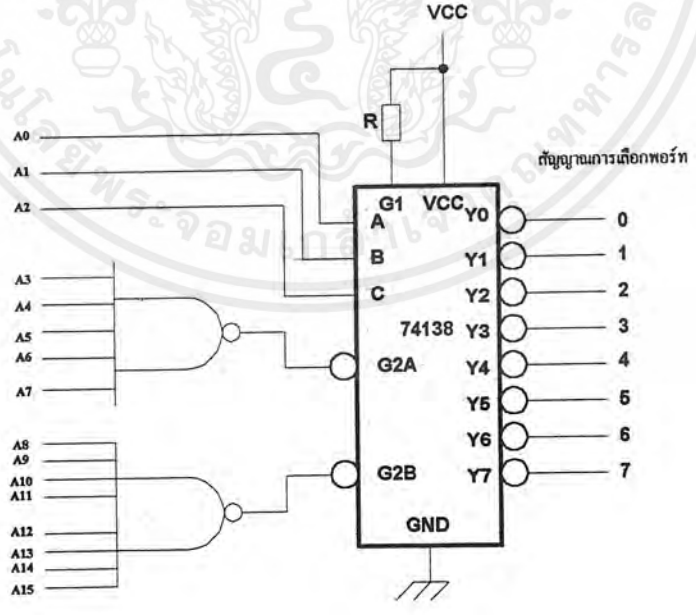
สล็อต IBM/PC ใช้บัสตำแหน่งทางด้านต่ำคือ A15 ถึง A0 เพื่อกำหนดตำแหน่งของพอร์ต ดังนั้นหมายความว่า สล็อตสามารถติดต่อกับพอร์ตขนาด 16 บิทได้ถึง 65536 พอร์ต และเนื่องจากในการอ่านและเขียนข้อมูล สัญญาณควบคุม RD และ WR จะไม่มีโอกาสแอคตีฟพร้อมกัน ดังนั้นพอร์ต หมายเลขเดียวกัน สามารถกำหนดให้เป็นพอร์ตอินพุตหรือพอร์ตเอ้าต์พุตก็ได้ คำสั่งทุกคำสั่งในกลุ่มอินพุต/เอ้าต์พุตจะใช้บัสตำแหน่ง A15-A0 นี้ เพื่อกำหนดตำแหน่งของพอร์ตทั้งสิ้น ดังนั้นในการถอดรหัสตำแหน่งของพอร์ตโดยทั่วไป แต่ถ้าเราต้องการใช้ข้อมูลที่บัสตำแหน่ง A15-A0 เพื่อเป็นตำแหน่งของพอร์ต ก็สามารถทำได้เช่น แต่ในการออกแบบวงจร ฮาร์ดแวร์โดยทั่ว ๆ ไป เราจะทำการถอดรหัสตำแหน่งของพอร์ตที่บัสตำแหน่งที่ A15-A0 วงจรในการถอดรหัสตำแหน่งของพอร์ตอย่างง่าย ๆ ดังรูป 2.16

จากรูป 2.16 เมื่อสล็อตทำคำสั่งอินพุตหรือเอ้าต์พุต มันจะส่วนสัญญาณในการเลือกพอร์ตมาบนบัสตำแหน่ง A15-A0 ถ้าข้อมูลที่ส่งมาเป็น FFFFH ก็จะทำให้เอ้าต์พุตของเกตแนนด์เป็นลอจิก 0 แต่ถ้าข้อมูลที่ส่งมานี้ไม่ใช้ FFFFH เอ้าต์พุตของเกตแนนด์ก็จะเป็น 1



รูปที่ 2.16 วงจรเลือกพอร์ตเบื้องต้น

แต่ในการใช้งานโดยทั่วไปนั้นจะมีการใช้พอร์ตมากกว่า 1 ดังนั้นการถอดรหัสจะใช้วงจรถอดรหัสหรือวงจรดีมัลติเพล็กซ์เพื่อทำการกำเนิดสัญญาณการเลือกพอร์ต รูปที่ 2.17 แสดงการกำเนิดสัญญาณการเลือกพอร์ตที่นิยมใช้แบบหนึ่ง

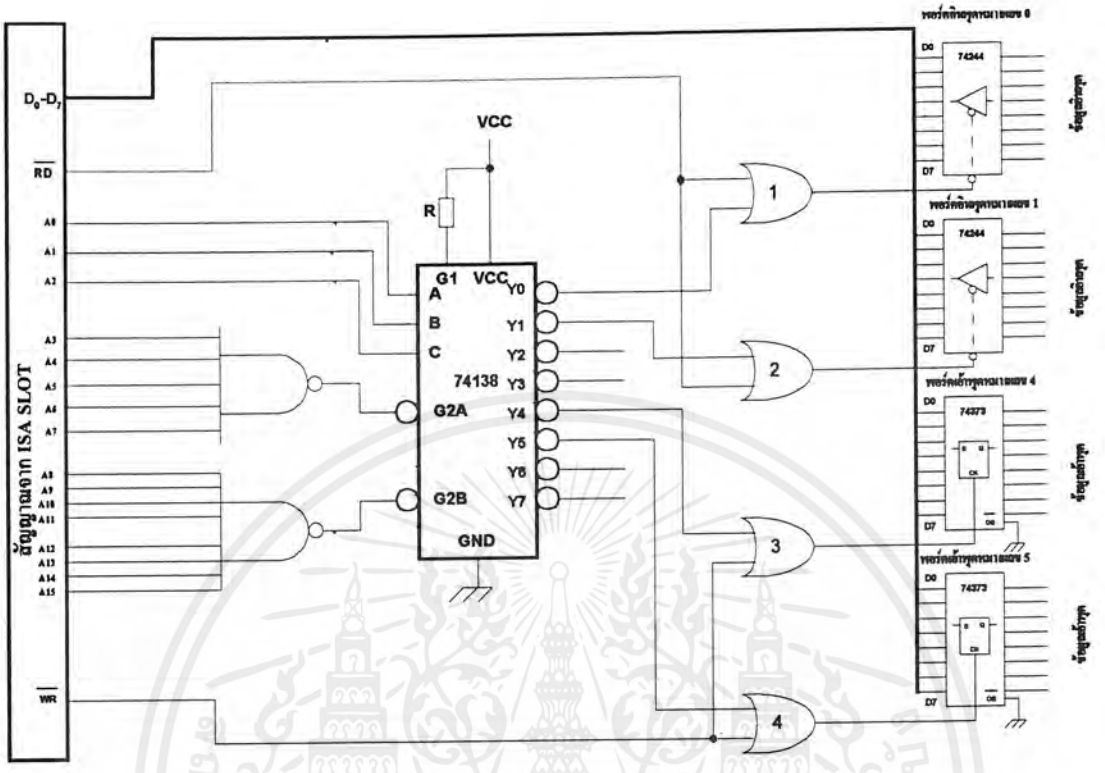


รูปที่ 2.17 วงจรสร้างสัญญาณการเลือกพอร์ต

จากวงจรรูปที่ 2.17 วงจรดีมัลติเพล็กซ์จะทำงานก็ต่อเมื่อสล็อตทำคำสั่งเกี่ยวกับการอินพุต หรือเอ้าต์พุต เพราะจะทำให้ขา IORQ แอคตีฟแต่ถ้าทำคำสั่งที่เกียวกับหน่วยความจำวงจรดีมัลติเพล็กซ์นี้จะไม่ทำงานเนื่องจากขา IORQ ไม่แอคตีฟดังนั้นเอ้าต์พุตของวงจรดีมัลติเพล็กซ์จะใช้สำหรับการเลือกพอร์ตหมายเลข 0 ถึงหมายเลข 7 ซึ่งพอร์ตต่าง ๆ เหล่านี้อาจกำหนดให้เป็น พอร์ตอินพุต หรือ พอร์ตเอาร์พุตก็ได้โดยขึ้นอยู่กับสัญญาณการควบคุมการอ่านและเขียนอีกสัญญาณหนึ่ง

วงจรการเชื่อมต่อพอร์ตแบบพื้นฐานที่แสดงถึง พอร์ตเอ้าต์พุต ซึ่งพอร์ตอินพุตจะใช้วงจรลอจิกแบบ 3 สถานะส่วนพอร์ตเอ้าต์พุต จะเป็นฟลิปฟลอปแบบ D วงจรการเชื่อมต่อ พอร์ตอินพุต และพอร์ตเอ้าต์พุต แบบพื้นฐานแสดงได้ดังรูปที่ 2.18

จากวงจรรูปที่ 2.18 เป็นการต่อพอร์ตอินพุต 2 พอร์ต และพอร์ตเอ้าต์พุต 2 พอร์ตโดยที่ใช้ไอซีทีทีแอลเป็นพอร์ตคือ พอร์ตอินพุตใช้ไอซี 3 สถานะเบอร์ 74244 เป็นอินพุตพอร์ตหมายเลข 0 และหมายเลข 1 ส่วนพอร์ตเอ้าต์พุต ใช้ไอซีฟลิปฟลอปแบบ D เบอร์ 74373 และกำหนดให้เป็นพอร์ตเอ้าต์พุตหมายเลข 4 และหมายเลข 5 จากรูปเอ้าต์พุตของวงจรดีมัลติเพล็กซ์ที่ Y0 จะแอคทีฟเมื่อเป็นการอ้างถึงพอร์ตหมายเลข 0 และเอ้าต์พุต Y1และแอคตีฟเมื่อเป็นการอ้างถึงพอร์ตหมายเลข 1 และเรียงไปตามลำดับจนถึงเอ้าต์พุต Y7 สำหรับการอ้างถึงพอร์ตหมายเลข 7 เมื่อนำสัญญาณจาก Y0 และ Y1 ไปทำการ OR กับสัญญาณ RD จากสล็อต จะได้สัญญาณควบคุมการอ่าน โดยที่เอ้าต์พุตของเกตออร์ 1 และเอ้าต์พุตของเกตออร์ 2 จะเป็นสัญญาณควบคุมการอ่านของอินพุตพอร์ตหมายเลข 0 และหมายเลข 1 ตามลำดับ ส่วนสัญญาณ Y4 และ Y5 ทำการออร์กับสัญญาณ WR ดังนั้นเอ้าต์พุตของเกตออร์ 3 และเกตออร์4จะเป็นสัญญาณควบคุมการเขียนข้อมูลออกที่พอร์ตเอ้าต์พุต ดังนั้นจากวงจรรูปที่ 2.18 นี้ จะเป็นวงจรที่ประกอบด้วยอินพุตพอร์ต 2 พอร์ตคือพอร์ตหมายเลข 0 และหมายเลข 1 และมีเอ้าต์พุตพอร์ต 2 พอร์ตคือ พอร์ตหมายเลข 4 และ หมายเลข 5 ซึ่งลำดับขั้นของการอ่านและเขียนข้อมูลเป็นดังนี้



รูปที่ 2.18 วงจรการต่อพอร์ตอินพุท/พอร์ตเอ้าพุท แบบขนาน 8 บิท

2.6.1 การอ่านข้อมูลจากพอร์ตอินพุต

ในการอ่านข้อมูลจากพอร์ต ทำได้โดยคำสั่งอินพุต เช่น IN A,(n) สมมุติให้หมายเลขพอร์ตเป็น 0 ดังนั้นลำดับขั้นในการอ่านข้อมูลมีดังนี้

1. เมื่อสล็อตทำคำสั่งอินพุต มันจะทำการกำหนดข้อมูลที่ใช้ในการระบุตำแหน่งลงมาบนบัสตำแหน่งที่ A7-A0 ในขณะนี้เอ้าต์พุตของวงจรดีมัลติเพล็กจะยังไม่แอคตีฟ เนื่องจากสัญญาณสโตรบที่ G2B ยังไม่แอคตีฟ
2. สัญญาณ IORQ แอคตีฟ เป็นระดับลอจิก 0 พร้อมกับสัญญาณ RD
3. สัญญาณเอ้าต์พุตของวงจรดีมัลติเพล็กที่ Y0 จะมีระดับลอจิก 0 และในขณะนี้สัญญาณ RD จะมีระกับ 0 ซึ่งจะทำให้ขาอีนาเบิลของ 74244 แอคตีฟ นั่นคือสัญญาณอินพุตของ 74244 จะต่อเข้ากับบัสข้อมูลของระบบ
4. ในช่วงเวลาต่อมา สล็อตจะทำการอ่านข้อมูลที่บัสข้อมูลนี้เข้าสู่รีจิสเตอร์ภายในของสล็อต

5. เมื่อสัญญาณ RD และ IORQ กลับสู่สภาวะลอจิก 1 จะทำให้พอร์ตอินพุตอยู่ในสภาวะอิมพีแดนซ์สูง หรือตัดสัญญาณอินพุตออกจากบัสของระบบ ซึ่งเป็นการสิ้นสุดขบวนการอ่านข้อมูลจากพอร์ตอินพุต

เห็นได้ว่าการอ่านข้อมูลจากพอร์ตอินพุตนี้ เมื่อทำการอ่านเสร็จแล้วจะทำให้สัญญาณอินพุตถูกตัดออกทันทีเพื่อไม่ให้รบกวนกับบัสข้อมูลของระบบที่จะมีการทำงานต่อไป

2.6.2 การเขียนข้อมูลออกพอร์ตเอ้าต์พุต

ในการเขียนข้อมูลออกยังงพอร์ตเอ้าต์พุต ทำได้โดยคำสั่งเอ้าต์พุต เช่น OUT (n),A ในที่นี้สมมุติให้พอร์ตเอ้าต์พุตเป็นพอร์ตหมายเลข 4 ดังนั้นลำดับขั้นในการส่งข้อมูลออกยังพอร์ตเอาต์พุตจะเป็นดังนี้

1. เมื่อสล็อตทำคำสั่งเอ้าต์พุต สล็อตจะทำการกำหนดข้อมูลที่ใช้ระบุตำแน่งของพอร์ตลงมาบนบัสตำแหน่ง
2. สล็อตส่งข้อมูลที่ต้องการส่งออกมาบนบัสข้อมูล
3. จากนั้น สล็อตส่งสัญญาณ IORQ ออกมาให้อยู่ในสภาวะแอคตีฟ คือระดับลอจิก 0 ดังนั้นในขณะนี้ เอ้าต์พุตของวงจรดีมัลติเพล็กที่ตำแหน่ง Y4 จะอยู่ในสถาวะแอคตีฟ
4. ต่อไปนี้สัญญาณ WR จะแอคตีฟ เป็นระดับ 0 ดังนั้นเมื่อสัญญาณ WR ตกมาเป็นระดับ 0 เอ้าต์พุตของเกตออร์ ก็จะตกมาเป็นระดับ 0 ด้วย แต่ในขณะนี้จะยังไม่มีผลต่อฟลิปฟลอปเนื่องจากขา ck ของฟลิปฟลอปแอคตีฟที่ของพัลส์ขาขึ้น ดังนั้นเมื่อสัญญาณ WR กลับสู่ระดับ 1 และจะทำให้เอ้าต์พุตของเกตออร์เปลี่ยนไปเป็นระดับ 1 ซึ่งเป็นขอบขาพัลส์ขาขึ้นและจำทำให้ฟลิปฟลอปนำสัญญาณที่บัสข้อมูลที่ต่ออยู่ที่อินพุตขอขฟลิปฟลอป ออกสู่เอ้าต์พุตของฟลิปฟลอป ซึ่งเป็นอันสิ้นสุดขบวนการเขียนข้อมูลออกสู่เอ้าต์พุตพอร์ต

บทที่ 3
การออกแบบสร้างโปรแกรมควบคุม
และวงจรอินเตอร์เฟสของแขนกลอ่อนตัวสองข้อต่อ

กล่าวนำ

ในบทนี้จะกล่าวถึงการออกแบบ ระบบโปรแกรมควบคุมแขนกลทั้งสองข้อต่อ โดยใช้โปรแกรม เดลไฟล์(Delphi) เขียนควบคุมโดยมีทั้งโหมดควบคุมโดยผู้ใช้และโหมดควบคุมอัตโนมัติ มีการปรับค่าตัวควบคุมพี (K_p) จากโปรแกรม การออกแบบส่วนอินเตอร์เฟส ในการออกแบบและการสร้างโปรแกรมนั้นจะพยายามแบ่งโปรแกรมออกเป็นส่วน ๆ เพื่อที่จะให้มีความง่ายต่อการเขียนโปรแกรมและง่ายต่อการที่จะพัฒนาโรแกรมในอนาคตด้วย ซึ่งโปรแกรมจะแสดงการควบคุมบนหน้าจอและจะส่งผลออกไปควบคุมที่อุปกรณ์ภายนอก โดยผ่านอินเตอร์เฟส ซึ่งสามารถอธิบายรายละเอียดต่าง ๆ ของการออกแบบและการสร้างได้ดังนี้

3.1 โครงสร้างของโปรแกรม

โครงสร้างของโปรแกรมสามารถแบ่งออกเป็นส่วนต่าง ๆ ได้ดังนี้

ฟอร์มหลัก ของโปรแกรม ทำหน้าที่ ควบคุมการทำงานทั้งหมดของโปรแกรม เช่น ควบคุมการเปิด ปิด ของ ฟอร์ม ต่าง ๆ , ทำการควบคุมอุปกรณ์ภายนอกโดยการสั่งจากหน้าจอ เป็นต้น

3.2 การออกแบบและการสร้างฟอร์มหลักของโปรแกรม

3.2.1 การออกแบบฟอร์มหลัก

ในการออกแบบของโปรแกรม เราจะต้องหาก่อนว่า เราต้องการให้โปรแกรมทำงานอะไรบ้าง ในส่วนของโปรแกรมนี้ เราต้องการให้ทำงาน โดยแบ่งออกเป็น 2 โหมดด้วยกัน ซึ่งสามารถแบ่งได้ดังนี้ คือ

- โหมดควบคุมโดยผู้ใช้
- โหมดอัตโนมัติ

เมื่อทราบถึง โหมด ต่าง ๆ แล้วตอนนี้ เราก็จะกล่าวถึงรายละเอียดต่าง ๆ และทำงานของ โหมด ที่กล่าวข้างบนอย่างละเอียดดังต่อไปนี้

การออกแบบฟอร์มหลักใน โหมดควบคุมโดยผู้ใช้ในโหมดควบคุมโดยผู้ใช้นั้นจะใช้อุปกรณ์ในการควบคุมบนฟอร์มหลักเกือบทั้งหมด โดยที่มีส่วนของ โหมดอัตโนมัติ เท่านั้นที่ไปใช้ไม่ได้

3.2.2 โหมดควบคุมโดยผู้ใช้

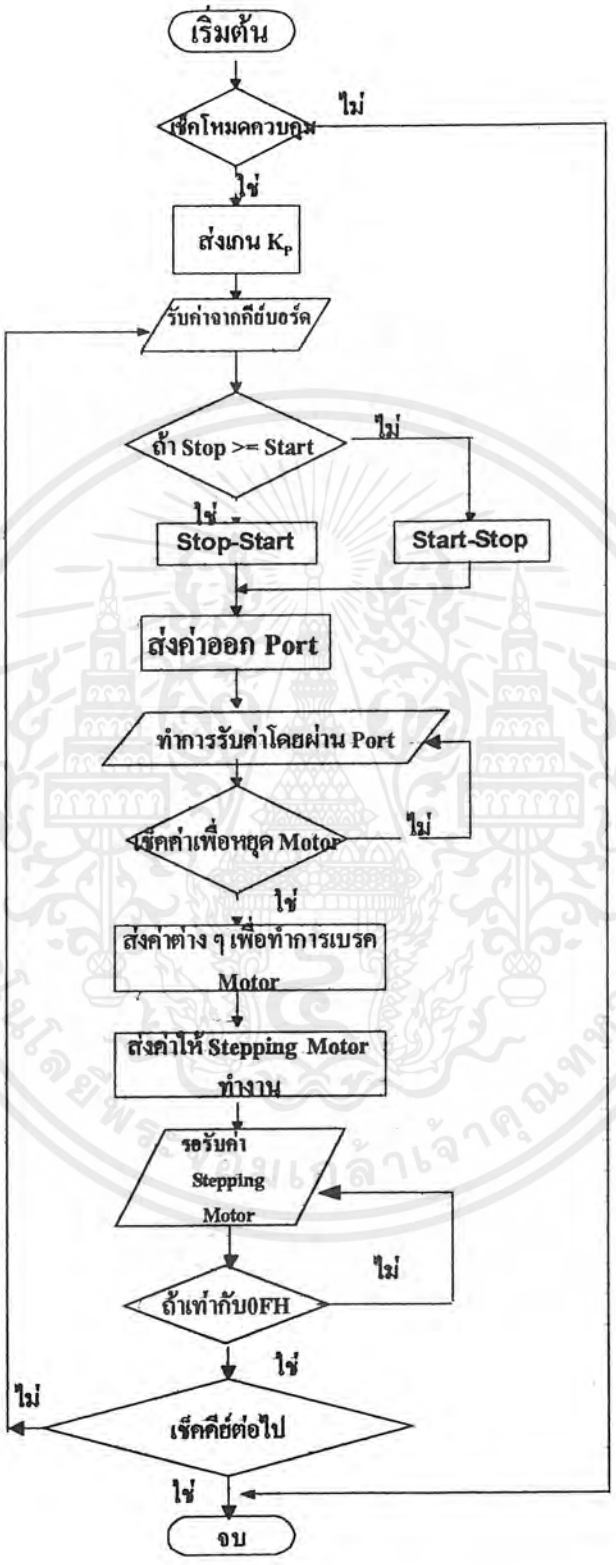
จากรูปที่ 3.1 จะแสดงการทำงานของโปรแกรมใน โหมดควบคุมโดยผู้ใช้เพื่อที่จะเข้าใจได้ง่ายขึ้น

เมื่อโปรแกรมเริ่มการทำงาน โปรแกรมจะทำการเช็ค โหมดการทำงานก่อน โดยในโปรแกรมนี้จะใช้ค่าของ ดัชนี เป็นตัวเช็ค โดยที่ค่าดัชนีนี้จะได้มาจากการคลิกใน ฟอร์มโหมดควบคุม เมื่อได้ค่า ดัชนี มาแล้วจะทำการเปรียบเทียบ ถ้าค่า ดัชนี มีค่า เท่ากับศูนย์โปรแกรมก็จะเข้าสู่ระบบของ โหมดควบคุมโดยผู้ใช้ แต่ถ้าค่า ดัชนี มีค่าไม่เท่ากับ ศูนย์ ก็จะไม่ได้เข้าสู่ โหมดควบคุมโดยผู้ใช้ ก็จะไม่สามารถทำงานได้

เมื่อโปรแกรมเข้าสู่ โหมดควบคุมโดยผู้ใช้แล้วโปรแกรมก็จะทำการรอที่จะส่งค่าของ K_p ในโหมดนี้ค่า K_p จะต้องตั้งค่าเอง โดยการตั้งค่าจาก ฟอร์ม เมื่อตั้งค่าเสร็จแล้วทำการส่ง

เมื่อโปรแกรมส่งค่า K_p แล้วเราจะมาที่ตัวคีย์ของโปรแกรม โดยที่โปรแกรมจะทำการรอค่าจาก คีย์ เพื่อที่จะทำงานต่อไป โดยที่คีย์แต่ละตัวจะมีค่าประจำคีย์อยู่แล้ว ค่าเล่านี้จะถูกส่งไปทำการเปรียบเทียบกัน เพื่อที่จะส่งไปควบคุมอุปกรณ์ภายนอกได้อย่างแม่นยำ การเปรียบเทียบนี้ทำได้โดย เมื่อกด คีย์ ค่าประจำตัวของ คีย์ จะถูกส่งไปเก็บไว้ที่ ตัวที่สร้างขึ้นเพื่อทำการเก็บค่า มีชื่อว่า Start และเมื่อทำการกด คีย์ อีกครั้งค่าที่ได้ก็จะไปเก็บไว้ที่ตัวเก็บค่าที่ชื่อว่า Stop และเมื่อได้ค่าเหล่านี้แล้วก็จะทำการเปรียบเทียบ โดยการ ถ้า Start มากว่า Stop ก็จะทำการให้ Start – Stop ค่าที่ออกมาจะทำการส่งไปเพื่อทำการควบคุม แต่ถ้า Stop มากกว่า Start ก็จะให้ Stop – Start แล้วส่งค่าออกไปเหมือนกัน

เมื่อส่งค่าออกไปควบคุมแล้วก็จะทำการ ให้ค่าของ Stop เท่ากับ Start และวนกลับไปเพื่อไปรอรับค่า เพื่อที่จะทำงานต่อไป



รูปที่ 3.1 โหมดควบคุมโดยผู้ใช้

3.3 การออกแบบของโหมดอัตโนมัติ

การออกแบบของ โหมดอัตโนมัติ การใกล้เคียงกับ โหมดควบคุมโดยผู้ใช้ อยู่แต่ต่างกับที่การรับค่า แล้วเราจะกล่าวให้อย่างละเอียดต่อไปนี้

3.3.1 โหมดอัตโนมัติ

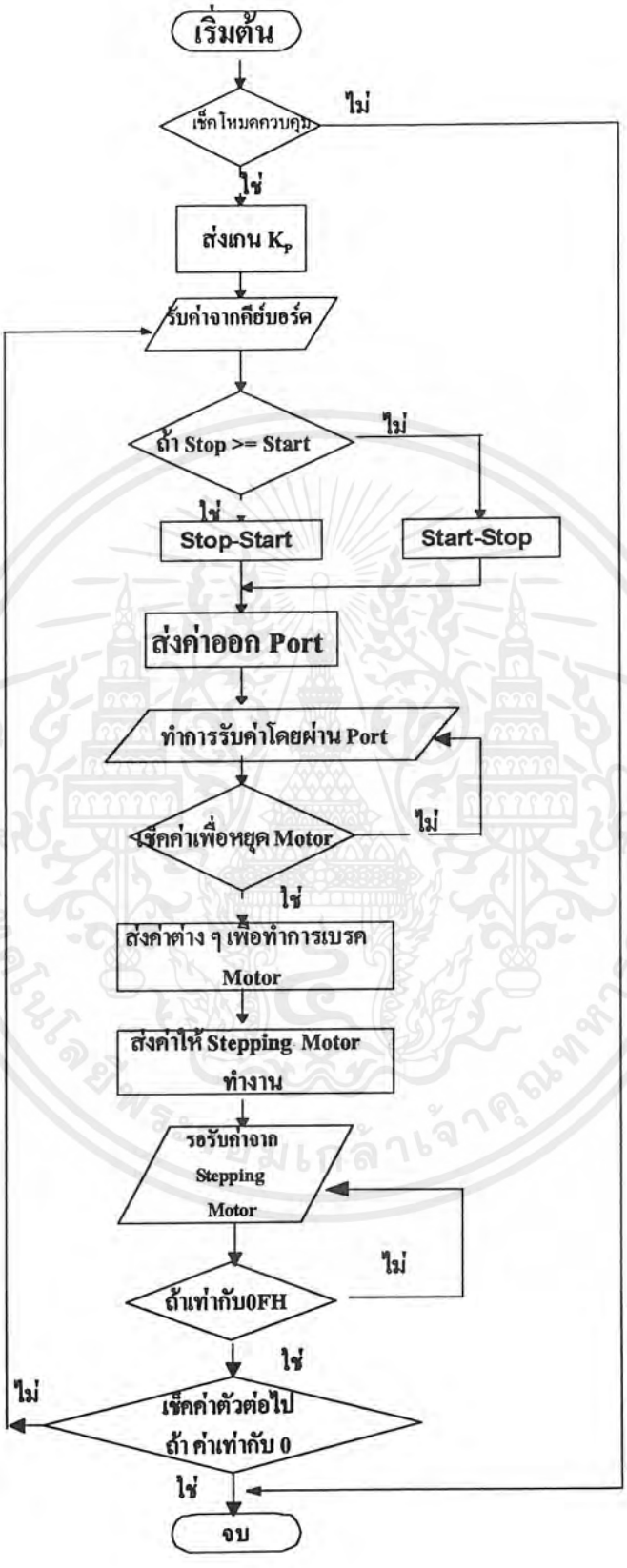
จากรูปที่ 3.2 การทำงานใน โหมดอัตโนมัติ เมื่อเริ่มต้นโปรแกรม โปรแกรมก็จะทำการเช็คค่าของ ดัชนี ก่อนเหมือนกับแบบ โหมดควบคุมโดยผู้ใช้ แต่โดยที่ว่าค่าของ ดัชนี จะเป็น 0 แต่ของโหมดอัตโนมัติ นั้นค่า ดัชนี จะมีค่าเป็น 1 เมื่อโปรแกรมได้ค่าดัชนีเป็น 1 โปรแกรมจะให้ทำงานในโหมดอัตโนมัติ

เมื่อเข้าสู่ในโหมดอัตโนมัติโปรแกรมจะเริ่มทำการส่งค่า K_p ทั้ง 3 ค่าออกไป โดยโปรแกรมในโหมดนี้จะไม่สามารถตั้งค่า K_p ได้เอง เพราะจะถูกตั้งไว้เรียบร้อยแล้ว โดยค่า K_p ที่ถูกตั้งไว้นี้จะเป็นค่าที่ดีที่สุด สำหรับอุปกรณ์ภายนอกที่จะทำงาน

เมื่อส่งค่า K_p เรียบร้อยแล้ว โปรแกรมจะมาทำการรับค่าที่ป้อนเข้ามา โดยการป้อนเข้ามานั้นค่าที่เข้ามาจะเป็นตัวเลข และตัวเลขเหล่านี้จะถูกนำ โดยการให้ตัวเลขเหล่านี้เป็น Sting และใช้ค่าประจำตำแหน่งที่เรียกว่าดัชนี กำหนดตำแหน่งของตัวเลขที่จะดึงเข้ามาใช้ทีละตัว แล้วนำค่าตัวเลข ไปเปรียบเทียบว่าตัวของมีค่าตรงกับค่าระจำตัวเท่าไร ตัวอย่างเช่น เลข 1 จะมีค่าประจำตัวเท่ากับ 10 เป็นต้น และโปรแกรมจะนำค่าประจำตัวของแต่ละตัวมาทำการเปรียบเทียบ

การเปรียบเทียบ นี้จะเป็นการเปปรียบเทียบที่จะเอาผลที่ได้นำไปทำการควบคุมอุปกรณ์ภายนอกทำโดยการนำค่าประจำตัวของตัวเลขตัวที่หนึ่งมาให้เป็นตัวที่เรื่อกว่าตัว Start และให้ค่าระจำตัวเลขตัวที่สองเป็นตัวที่เรียกว่า Stop และมีเงื่อนไขว่า ถ้าตัว Start มากกว่า Stop ก็จะให้ ค่า Start –Stop และนำผลที่ได้ส่งไปควบคุมและ ถ้า Stop มากกว่า Start ก็จะให้ Stop – Start และนำผลที่ได้ส่งออกไป

เมื่อส่งผลออกไปแล้ว โปรแกรมจะทำการรอรับค่าที่เข้า โดยที่ค่าที่เข้า ถ้ามีค่าเป็นค่า 0F h โปรแกรมก็จะทำงาน โดยการให้ค่าประจำตัวเลขที่อยู่ใน Stop ก็จะถูกส่งมาเก็บยัง ตัว Start และค่า Index ที่เป็นตัวบอกตำแหน่งของตัวเลขก็จะเพิ่มขึ้น เพื่อที่จะได้นำค่าตัวเลขเข้าไปในโปรแกรมเพื่อที่จะได้ทำงานต่อไปเรื่อย ๆ จนกว่าตัวเลขที่ป้อนให้หมดไป แต่ถ้าโปรแกรมรับค่าที่เข้ามาค่าที่ไม่ใช้ค่า 0F โปรแกรมจะไม่ยอมเพิ่มค่าดัชนี เพื่อเลื่อนให้เป็นค่าตัวเลขตัวต่อไป



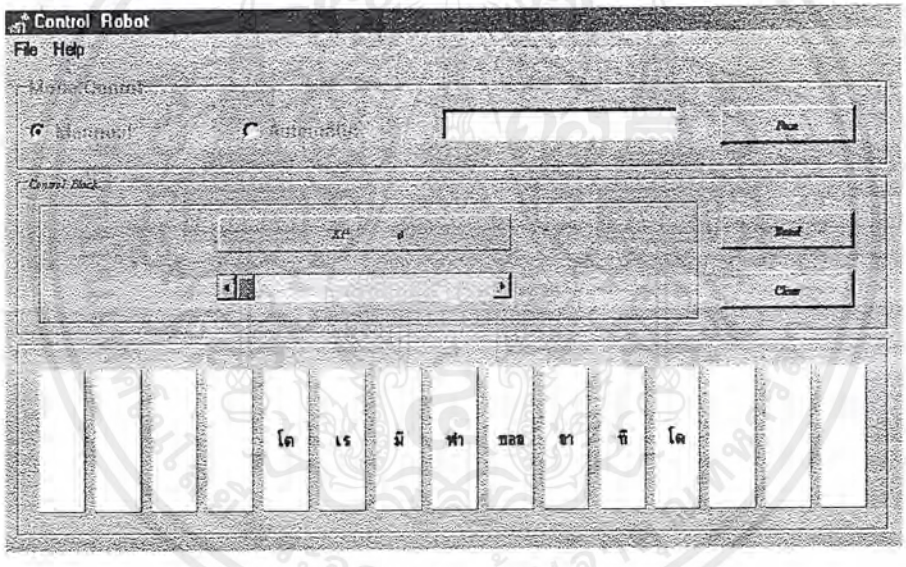
รูปที่ 3.2 โหมดแบบอัตโนมัติ

3.4 การสร้างในส่วนของฟอร์มหลัก

ในฟอร์มของฟอร์มหลัก ทำหน้าที่ในการรับคำสั่งการทำงานต่าง ๆ ทั้งหมด ของโปรแกรม ในการสร้างนั้นเราจะแบ่งออกเป็นส่วนย่อย ๆ โดยในส่วนย่อย ๆ นี้จะประกอบไปด้วย

- ส่วนของเมนู
- ส่วนของ โหมดควบคุม
- ส่วนของ
- ส่วนของคีย์

โดยฟอร์มที่สร้างนี้เป็นฟอร์มแบบWin95 Form จึงแสดงเป็นดังรูปที่ 3.3



รูปที่ 3.3 แสดงฟอร์มหลัก

เมื่อเราได้เห็นฟอร์มแล้ว เรามาดูถึงรายละเอียดต่าง ๆ จึงขออธิบายในการสร้างในแต่ละส่วนดังต่อไปนี้

1. การสร้างส่วนของ Menu โดยเราจะใช้การกำหนด TMainMenu

ขั้นตอนแรก เราจะทำการกำหนดส่วนที่จะต้องใช้ก่อนที่จะทำการสร้าง โดยในโปรแกรมนี้จะมี Menu ในคำสั่งเหล่านี้ คือ

- File ประกอบไปด้วย Exit ทำหน้าที่ปิดฟอร์มหลัก
- Help ประกอบด้วย Content , About ทำหน้าที่ไว้เป็นตัวช่วยเหลือ

ขั้นตอนที่สอง เมื่อได้ส่วนที่ต้องการแล้ว ก็ทำการ Component ที่จะต้องใช้ คือ Component ที่มีชื่อว่า TMainMenu ในตัว Component แต่ละตัวจะต้องทำการกำหนดค่าของตัวมันด้วย เพื่อที่จะได้การทำงานของ Component ตามโปรแกรมที่เราต้องการ

เมื่อเราได้ Component เรียบร้อยแล้ว เรามาดูค่าต่าง ๆ ของ Component จะแสดงให้เห็นในข้างล่าง

```
object Mainmenu : TMainMenu
    Left = 86
    Top = 4
  object File : TMainMenu
    Caption = ' File '
    ShortCut = 0
  object Exit : TMainItem
    Caption = ' Exit '
    ShortCut = 0
    OnClicK = ExitClicK
  end
  object Graph1 : TMainItem
    Caption = ' Graph '
    ShortCut = 0
  end
  object Help1 :TMenuItem
    Caption = ' Help '
    ShortCut = 0
  object Content1 :TMenuItem
    Caption = ' Content '
    ShortCut = 0
  end
  object About1 :TMenuItem
    Caption = ' About '
    ShortCut = 0
```

end

2. การสร้างส่วนของโหมดควบคุม

ขั้นตอนแรก การทำคล้ายกับแบบแรก โดยการเลือก Component ที่ต้องการจะใช้งาน โดยเราจะต้องรู้ว่า Component แต่ละตัวทำงานอะไรบ้าง แต่ในโปรแกรมนี้จะใช้ Component ดังนี้ คือ

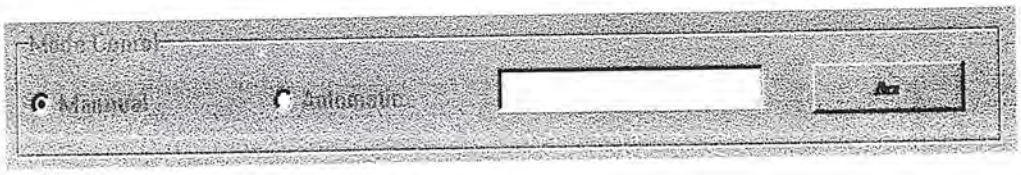
- Groupbox ทำหน้าที่ ใส่ไว้เพื่อแยก ส่วนต่าง ๆ ออกจากกัน
- Radio Group ทำหน้าที่ เลือก โหมด
- Edit ทำหน้าที่ เป็นส่วนรับข้อมูล ใน โหมดอัตโนมัติ
- Button ทำหน้าที่ สั่งให้โปรแกรม RUN ใน โหมดอัตโนมัติ

ขั้นตอนที่สอง เมื่อเราได้ Component ที่ต้องการแล้ว เราก็จะทำการกำหนด ค่าของ Component ในแต่ละตัวลงไป เพื่อที่จะให้ได้ตำแหน่ง และหน้าที่ของ Component ตามที่เราต้องการ การกำหนดค่าจะมาค่าเป็นดังต่อไปนี้

```
object RadioGroup : TRadioGroup
        Left = -238
        Top = 8
        Width = 625
        Height = 65
        Caption = 'Mode Control'
        Columns = 4
        Font.Color = clFuchsia
        Font.Height = -13
        Font.Name = 'Century SchoolbooK]
        Font.Style = [fsBold]
        ItemIndex = 0
        Ithems.Strings = ('Mannual' Automatic')
        ParentFont = False
        TabOrder = 2
end
```

```
object Editl : TEdit
        Left = 66
        Top = 32
        Width = 169
        Height = 22
        TabOrder = 3
end
object Button3 : TButton
        Left = 266
        Top =32
        Width = 97
        Height = 25
        Caption = 'Run'
        Font.Color = clBlacK
        Font.Height = -13
        Font.Name = 'Century SchoolbooK'
        Font.Style = [fsBold, fsItalic]
        ParentFont = False
        TabOrder = 4
        OnClicK = Button3 ClicK
end
```

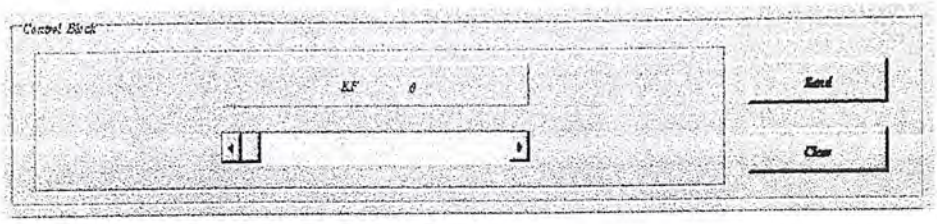
เมื่อทำการกำหนด Component เสร็จแล้ว เราก็จะได้ดังรูปที่ 3.4



รูปที่ 3.4 ส่วนของโหมดควบคุม

```
        OnChange = ScrollBarlChange
end
object Labell : TLabel
        Left = 80
        Top = 4
        Width = 9
        Height = 16
        Caption = '0'
end
end
object Panel25 : TPanel
        Left = 176
        Top = 16
        Width = 121
        Height = 25
        Caption = 'KP '
        TabOrder = 4
object Buttonl : TButton
        Left = 504
        Top = 32
        Width = 96
        Height = 25
        Caption = 'Send'
        TabOrder = 1
        OnClicK = ButtonlChicK
end
```

เมื่อทำการกำหนด Component เสร็จแล้ว เราก็จะได้ดังรูปที่ 3.5



รูปที่ 3.5 แสดงส่วนของบล็อกควบคุม

ขั้นตอนที่สาม ทำการเขียนโปรแกรมลงใน Component ตามที่เราต้องการ

4. การสร้างส่วนของคีย์

ขั้นตอนแรก ทำการกำหนด Component ที่ต้องการ ในโปรแกรมในส่วนนี้จะใช้ Component ดังต่อไปนี้

- Group Box ในส่วนนี้จะใช้ 1 ตัว ทำหน้าที่ แบ่งส่วนออกจากส่วนอื่น
- Panel ในส่วนนี้จะใช้ 15 ตัว ทำหน้าที่ ไว้กดเพื่อส่งค่า

ขั้นตอนที่สอง เมื่อได้ Component ที่ต้องการ จะต้องทำการกำหนดค่าของ Component ตัวต่าง ๆ โดยค่าที่กำหนดจะเป็นดังต่อไปนี้

```
object GroupBox1:TGroupBox
        Left = -238
        Top = 184
        Width = 625
        Height = 137
        TabOrder = 0
object Panel1:TPanel
        Left = 16
        Top = 24
        Width = 33
        Heigth = 97
```

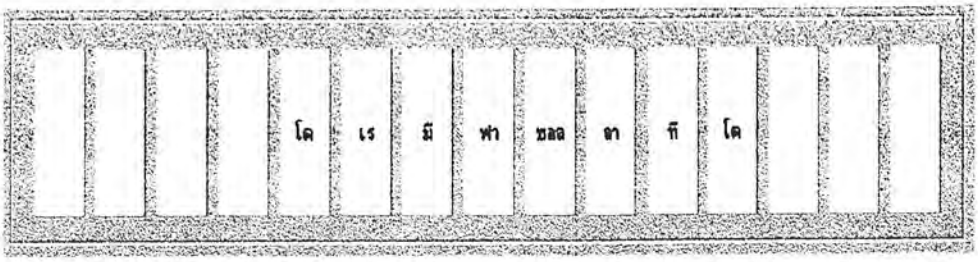
```
        Color = clWhile
        TabOrder = 0
end
object Panel2:TPanel
        Left = 56
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel3:TPanel
        Left = 96
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
        end
object Panel4:TPanel
        Left = 136
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel5:TPanel
        Left = 176
        Top = 24
```

```
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel6:TPanel
        Left = 216
        Top = 24
        Width = 33
        Heigth  = 97
        Color = clWhile
        TabOrder = 0
end
object Panel7:TPanel
        Left = 256
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel8:TPanel
        Left = 296
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel9:TPanel
```

```
        Left = 336
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel10:TPanel
        Left = 376
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel11:TPanel
        Left = 417
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel12:TPanel
        Left = 456
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
```

```
end
object Panel13:TPanel
        Left = 496
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel14:TPanel
        Left = 536
        Top = 24
        Width = 33
        Heigth = 97
        Color = clWhile
        TabOrder = 0
end
object Panel15:TPanel
        Left = 576
        Top = 24
        Width = 33
        Heigth .= 97
        Color = clWhile
        TabOrder = 0
end
```

เมื่อกำหนดค่าใน Component เสร็จแล้วเราก็จะได้ รูปแบบ เป็นดังรูปที่ 3.6



รูปที่ 3.6 แสดงรูปแบบส่วนของคีย์

ขั้นตอนที่สาม เมื่อทำการกำหนดรูปแบบเสร็จแล้วก็จะทำการเขียนโปรแกรมลงไปในคอมโพแนนต์ต่าง ๆ ตามที่เราต้องการ

3.5 รายละเอียดการทำงานของ function และ Procedure ในกิจกรรมต่าง ๆ

1. **function InPort (Port : Word) : Word;**

หน้าที่

ตัวนี้ไว้สำหรับรับค่าข้อมูลที่ส่งเข้ามา โดยผ่านพอร์ทที่กำหนดไว้ โดยค่าที่เข้ามาจะเป็นค่าที่อยู่ในชนิด Word

หลักการทำงาน

เมื่อเรียกใช้ฟังก์ชันนี้ ฟังก์ชันนี้จะทำการนำค่า Port ที่ป้อนให้ เข้ามาทำการกำหนด Port และทำการนำข้อมูลเข้ามา โดยข้อมูลที่เข้ามานั้นจะถูกส่งให้ไปเก็บไว้ใน ตัว ตัวแปร ที่มีชื่อว่า Result เพื่อที่จะไว้ให้ในส่วนอื่น ๆ ของโปรแกรมเรียกใช้ได้ง่าย

```
function Inport ( Port : Word ) : Word;
  begin
        asm
        mov dx , Port
```

```
        in ax , dx
        mov result , ax
        end
end;
```

2. procedure OutPort (Address , Data : Word) ; Assembler

หน้าที่

หน้าที่มีไว้สำหรับทำการส่งข้อมูลที่อยู่ใน ตัว ตัวแปร ที่มีชื่อว่า Data ไปสู่ Port ที่กำหนดเข้ามา โดยค่าของ Port จะอยู่ใน ตัวแปร ที่มีชื่อว่า Address

หลักการทำงาน

ในการทำงานใน procedure นี้ จะทำการ Clear ค่าของ register ที่จะเก็บค่าของ Address และ Data ออกไปก่อน หลังจากนั้นจะนำค่าของ Address , Data ที่กำหนดให้มาใส่ลงไป เพื่อที่จะทำการเข้าสู่การส่งข้อมูล เมื่อส่งข้อมูลเสร็จแล้ว ก็จะนำค่าที่เก็บไว้ใน register ก่อนที่จะทำงานในการส่งข้อมูลกลับมาเก็บไว้ที่เดิม ก็เป็นการเสร็จของ procedure นี้

```
procedure OutPort ( Address , Data : Word ) ;Assembler
begin
        asm
        push dx
        push ax
        mov dx , Address
        mov ax , Data
        out  dx , ax
        pop ax
        pop dx
        end
end;
```

3. procedure Move (Start , Stop : Integer) ;

หน้าที่

หน้าที่ในส่วนของ procedure นี้คือ เป็นตัวเปรียบเทียบ ค่าที่ได้รับมาจากการกดคีย์ ของระบบ โหมดควบคุมโดยตัวผู้ใช้และการรับค่ามาจากการป้อนตัวเลขของระบบ อัตโนมัติ เพื่อที่จะให้ได้ค่าที่จะส่งออกไปควบคุมอุปกรณ์ภายนอก

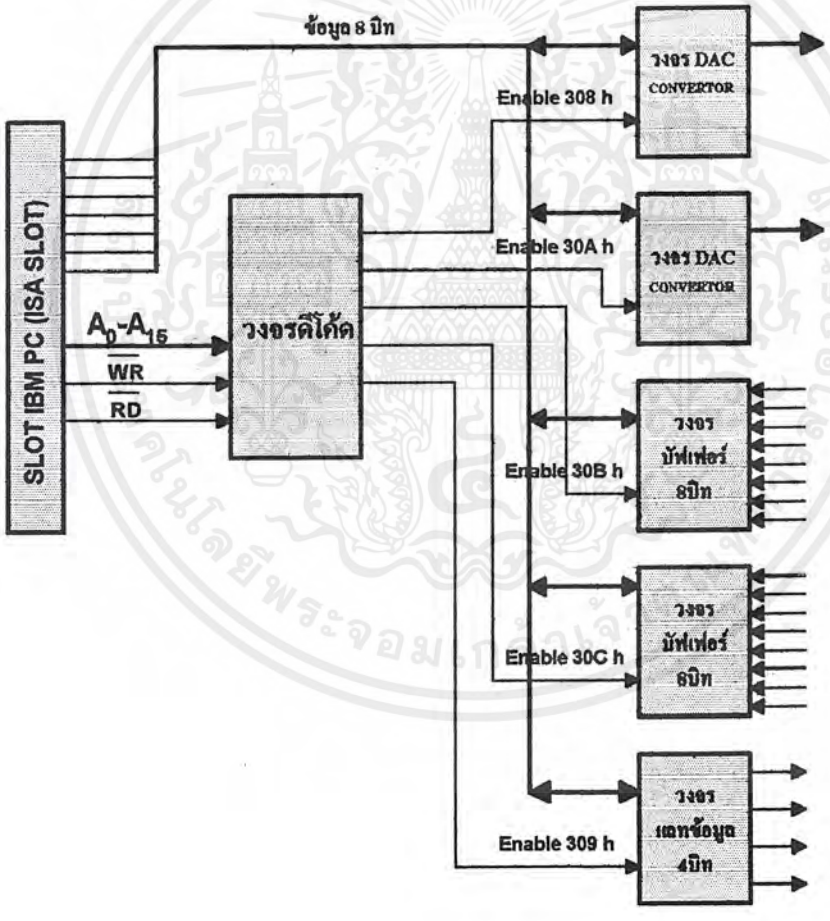
หลักการทำงาน

หลักการทำงานของ Procedure นี้คือ ใน procedure นี้ จะมีการกำหนดตัว ตัวแปร หลัก ๆ อยู่ 2 ตัว คือ Start , Stop โดย procedure นี้จะทำการรับค่าของ Start และ Stop เข้ามาและทำการเปรียบเทียบ ถ้า Start มากกว่า Stop ก็จะให้ Start ลบด้วย Stop แต่ถ้าค่า Stop มากกว่า Start ก็จะให้ Stop ลบด้วย Start เพื่อที่จะให้ค่าที่ออกมานั้นมีค่าไม่เป็นลบ นั้นเอง และเมื่อได้ค่ามาแล้วก็จะทำการส่งค่าออกไป โดยใช้ procedure ที่ได้กล่าวมาแล้ว

```
precedure Move ( Start , Stop : Integer );
begin
        if Stop >= Start then
         begin
          Mov := Stop – Start ;
          OutPort ( $0378 , Mov ) ;
         end
        else
         begin
          Mov := Start – Stop ;
          OutPort ( $0378 , Mov ) ;
         end;
end;
```

3.6 การดีโค้ดแอดเดรสและวงจรแล็ชท์

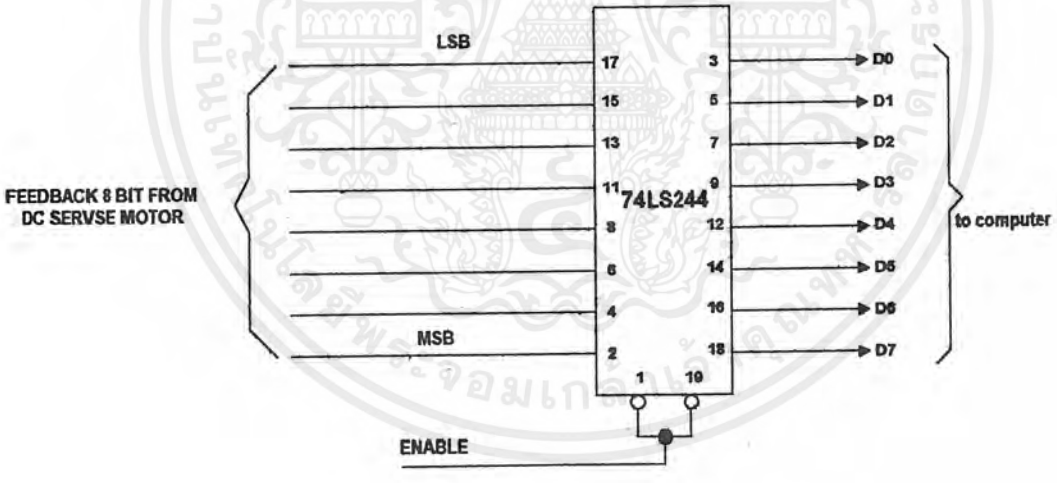
ในการเขียนข้อมูลออกไปยังบัสข้อมูลข้าไปยังอุปกรณ์รอบนอก เราต้องกำหนดแอดเดรสของอุปกรณ์รอบนอกเหล่านั้นด้วย อุปกรณ์รอบนอกเหล่านั้นในที่นี้ได้แก่ DAC 1408 เนื่องจาก DAC 1408 ไม่มีส่วนติดต่อแอดเดรสโดยตรงจากคอมพิวเตอร์ได้ เราจำเป็นต้องใช้วงจรดีโค้ดแอดเดรสแยกต่างหาก ในที่นี้เราจะใช้ไอซีเบอร์ 74LS138 (three to eight-line decoder) เป็นตัวดีโค้ดแอดเดรสโครงสร้างของขาไอซี 74LS138 แสดงดังในรูปที่ 3.7



รูปที่ 3.7 บล็อกไดอะแกรมการทำงานของวงจรดีโค้ดและการนำข้อมูลเข้าออก

การดีโค้ดเลขฐานสิบหก (Hex) 308H กับ 309H ให้กับ DAC 1408 จากวงจรดีโค้ด 74LS138 และแอดเดรส 30AH ไปอิเนเบิล 74LS373 และแอดเดรส 30BH กับ 30CH จะนำข้อมูล 8 บิท ไปอิเนเบิลให้ไอซี 74LS244 ซึ่งเป็น BUFFER ดังนั้นเอ้าท์พุทจากวงจรดีโค้ด คือ 308H-30CH จากวงจรสัญญาณ A_0-A_2 เป็นอินพุท " select " เข้าที่ขา A,B,C ของ 74LS138 ตามลำดับ A_{15} , A_{14} , A_{13}, A_{12}, A_{11} นำมาทำการ OR กัน เพื่อเข้าที่ขา G_2A ส่วน A_{10}, A_7, A_6, A_5, A_4 ก็ทำการ OR กันเพื่อเข้าที่ขา G_2B เพราะขา G_2A และ G_2B ต้องการลอจิก " 0 " ในการดีโค้ด สำหรับ A_9, A_8, A_3 ทำการ AND เพื่อเข้าที่ขา G_1 ต้องการลอจิก " 1 "

ตามฟังก์ชั่นของวงจรดีโค้ด 74LS138 เอ้าท์พุทแอดเดรส 308H ออกที่ขา Y_0 จะเป็นลิจิก " 0 " พร้อมทั้งขา IOWR เป็นลิจิก " 0 " เพื่อไปเลือกอิเนเบิลขา 11 ของ 74LS373 ทำงาน ส่วนเอ้าท์พุทแอดเดรส 309H-30CH ออกที่ขา Y_1-Y_4 ตามลำดับ แต่แอดเดรส 30BH และ 30CH จะใช้ขา IORD ไปอิเนเบิล 74LS244 เพื่อนำข้อมูลเข้าดังรูป 3.8



รูปที่ 3.8 วงจรนำข้อมูลเข้า

คุณลักษณะทางไฟฟ้าที่สำคัญของ 74LS373 (Octal D-type latches) มีดังต่อไปนี้

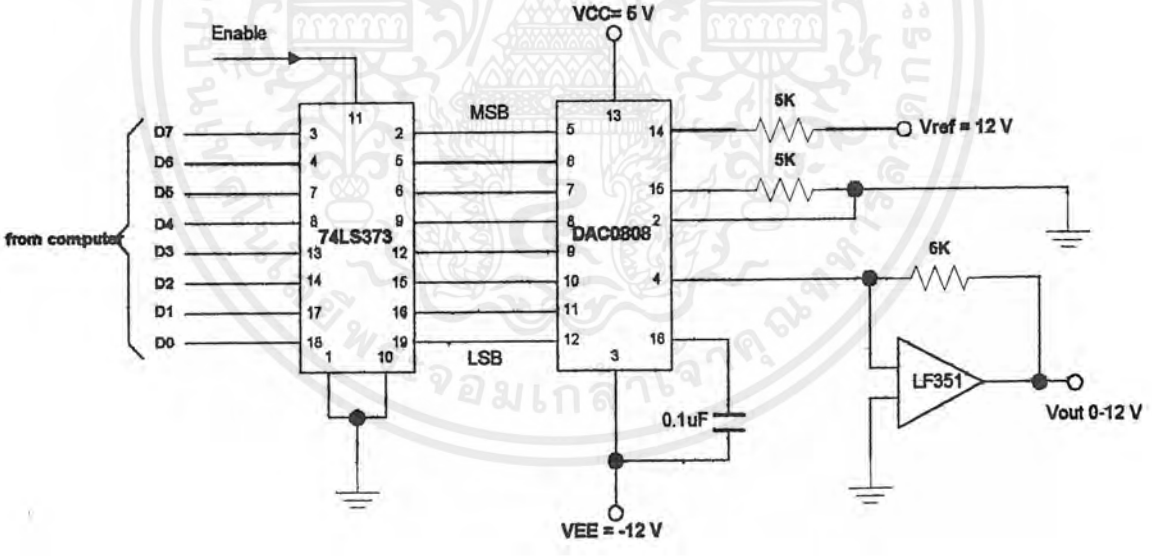
- ไอซีนี้มี 20 ขา แต่มีการแลชท์ข้อมูลได้ 8 บิท
- การโหลดข้อมูลกระทำได้แบบขนาน
- มีบัฟเฟอร์คอนโทรลอินพุท

- มีบัสโทร-สเตทสำหรับขับเอ้าท์พุท
- อินเบิลอินพุทมีฮีสเทเรซีสเพื่อแก้ไขการกำจัดน้อยส์ให้ดีขึ้น

ไอซี 74LS373 มีการทำงานดังนี้ เมื่ออินเบิล (G) อินพุทมีค่า HIGH เอ้าท์พุท Q จะมีค่าเป็นไปตามข้อมูลอินพุท (D) ถ้าหากอินพุทอินเบิล (G) มีค่าเป็น LOW เอ้าท์พุท Q จะแลชท์ข้อมูลที่ได้เซ็ตไว้แล้ว โครงสร้างของขาไอซี 74LS373 และตารางฟังก์ชั่นแสดงได้ดังในรูปที่ 3.9

เราจะต่อบัสข้อมูลของ คอมพิวเตอร์ เข้ากับอินพุทข้อมูล 74LS373 แลชท์ เอ้าท์พุทของแลชท์จะถูกต่อเข้ากับอินพุทของ DAC 1408 และเราจะต่ออินพุท OC เข้ากับกราวนด์ เพื่อว่าเอ้าพุทของแลชท์สามารถมีค่าเป็นได้ทั้ง HIGH หรือ LOW ในที่สุดอินพุท G ของ 74LS373 จะได้จากการ NOR สัญญาณเอ้าท์พุท Y_0 ของ 74LLS373 กับสัญญาณ R/W ของคอมพิวเตอร์ วงจรดีโค้ดแอดเดรสและแลชท์ที่สมบูรณ์แสดงได้ในรูปที่ 3.9

การดีโค้ดจะนำสัญญาต่างๆจาก slot ISA ดังนี้



รูปที่ 3.9 วงจรนำข้อมูลออก

บทที่ 4
ผลการทดลอง

4.1 การทดลองโปรแกรม

4.1.1 โหมดควบคุมโดยผู้ใช้

ในการทดลอง เมื่อเลือกโหมดควบคุมโดยผู้ใช้และเลือกค่าต่าง ๆ เสร็จเรียบร้อยแล้ว เราจึงทำการสั่งไปที่คีย์ขั้นตอนผลที่ได้คือ

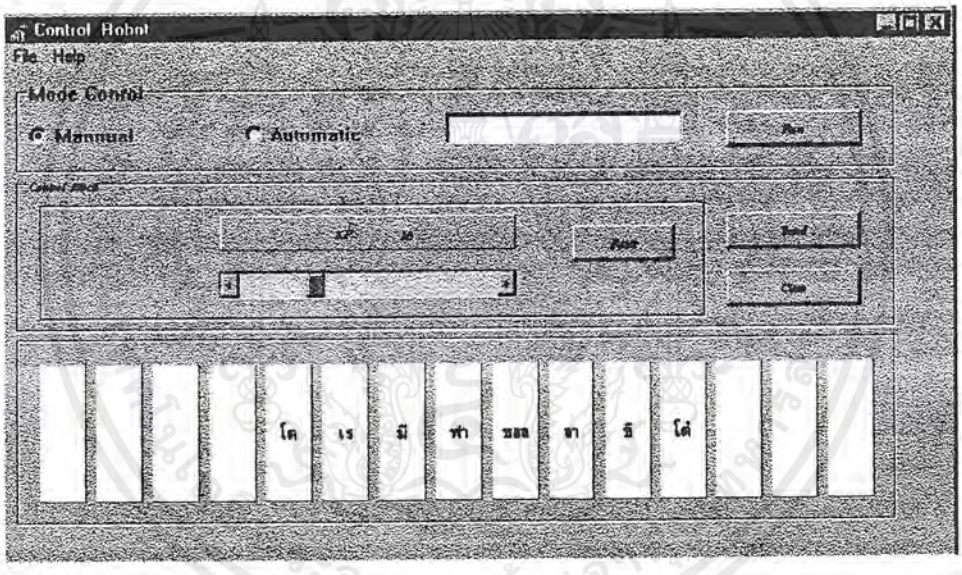
เมื่อกดคีย์ค่าที่ประจำคีย์จะเข้าไปทำการเปรียบเทียบและทำการส่งค่าของความแตกต่างออกมายังพอร์ท 0309h โดยค่าที่ออกเป็น 8 บิท โดยค่า 0 บิทนี้ บิทที่ 8 จะเป็นตัวกำหนดทิศทางของมอเตอร์โดยถ้าบิทที่ 8 มาค่าเป็น 1 จะทำให้มอเตอร์หมุนไปทางขวา แต่ถ้าบิทที่ 8 มาค่าเป็น 0 จะทำให้มอเตอร์หมุนไปทางซ้าย

เมื่อโปรแกรมทำการส่งค่าออกไปแล้ว โปรแกรมจะทำการป้องกันการกดคีย์จนว่าจะทำงานเสร็จ เมื่อโปรแกรมทำการป้องกันคีย์เสร็จแล้วก็จะทำการเช็คค่าที่จะเข้ามา เมื่อได้ตาม เราสั่งออกไปแล้ว โปรแกรมจะทำการเช็คค่าที่เข้ามาตลอดเวลา โดยจะมีค่าที่เข้ามานี้จะมาจากมอเตอร์ ส่งเข้ามา เมื่อได้ค่าตรงกับที่เราส่งออกไปแล้ว โปรแกรมจะทำการหยุดมอเตอร์โดยการส่งค่า 00h ออกไปยัง พอร์ท 0308h (เลขฐาน 16) และส่งค่า C0h,40h,00h (เลขฐาน 16) ไปยังพอร์ท 0309h (เลขฐาน 16) ตามลำดับ เมื่อส่งค่าเหล่านี้เสร็จเรียบร้อยแล้วก็จะส่งค่าไปให้ยังสเตปปิ้งมอเตอร์ โดยค่าที่ส่งนี้จะมีค่า 20 (เลขฐาน 16) เพื่อทำการสั่งให้สเตปปิ้งมอเตอร์ทำงาน

เมื่อโปรแกรมทำการส่งค่าไปยังสเตปปิ้งมอเตอร์แล้วโปรแกรมจะทำการเช็ค เพื่อหาค่าโดยค่าที่เช็คนี้จะต้องมีค่า 0F (เลขฐาน 2) เพื่อที่จะบอกโปรแกรมว่าทำงานเสร็จเรียบร้อยแล้ว ก็เป็นการเสร็จการทำงานในการส่งหนึ่งครั้ง

4.1.2 โหมดอัตโนมัติ

เมื่อทำการเลือกโหมดเรียบร้อยแล้ว ทำการใส่ข้อมูลที่เราต้องการให้โปรแกรมทำงาน เมื่อใส่ข้อมูลแล้วให้กดปุ่ม Run โปรแกรมก็จะเริ่มทำงาน เมื่อโปรแกรมเริ่มทำงาน โปรแกรมจะทำการส่งข้อมูลเข้าไปเก็บไว้ใน String และทำการเรียกค่าออกมาทำงานทีละตัว เพื่อทำการเปรียบเทียบเพื่อให้ได้ค่าที่จะทำการส่งไปยังมอเตอร์ โดยค่านี้จะทำการส่งออกไปยังพอร์ท 0308h (เลขฐาน 16) โดยค่าที่ออกเป็น 8 บิท โดยค่า 0 บิทนี้ บิทที่ 8 จะเป็นตัวกำหนดทิศทางของมอเตอร์โดยถ้าบิทที่ 8 มาค่าเป็น 1 จะทำให้มอเตอร์หมุนไปทางขวา แต่ถ้าบิทที่ 8 มาค่าเป็น 0 จะทำให้มอเตอร์หมุนไปทางซ้าย



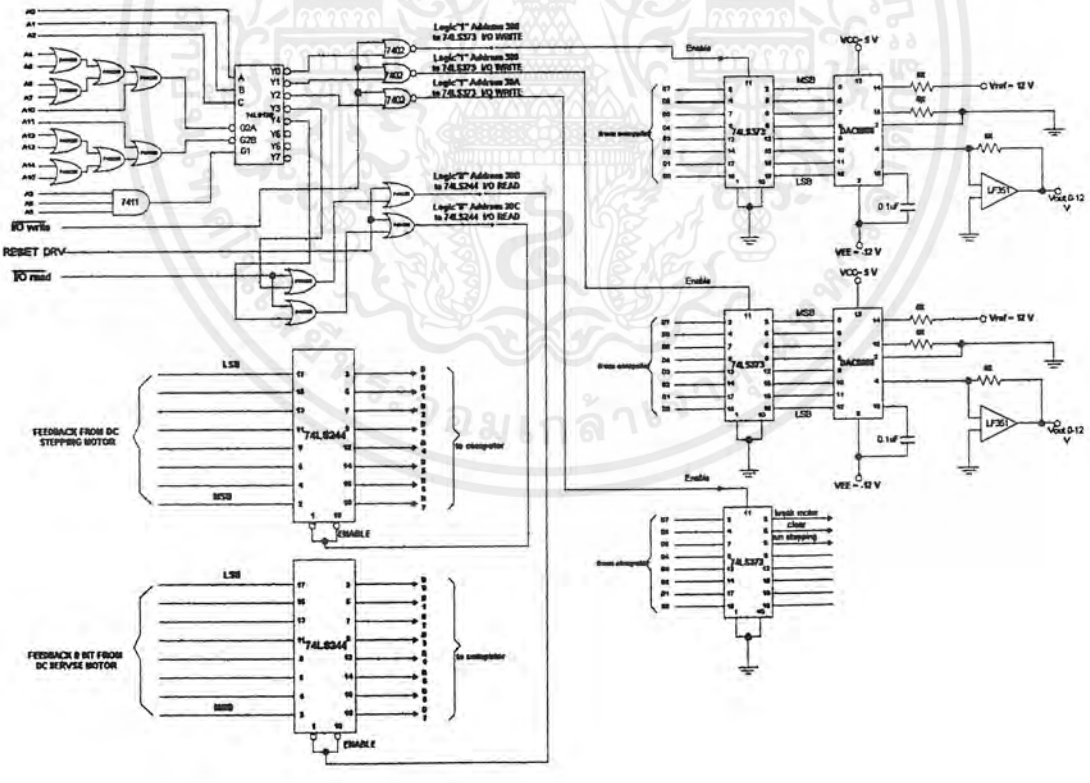
รูปที่ 4.1 รูปขณะที่โปรแกรมทำงาน

เมื่อโปรแกรมส่งค่าออกไปแล้วก็จะทำการเช็คค่า โดยจะทำการนำค่าเข้ามา โดยจะมีค่าที่เข้ามานี้จะมาจากมอเตอร์ส่งเข้ามา เมื่อได้ค่าตรงกับที่เราส่งออกไปแล้ว โปรแกรมจะทำการ หยุดมอเตอร์ โดยการส่งค่า 00 ออกไปยัง พอร์ท 0308h (เลขฐาน 16) และส่งค่า C0h,40h,00h (เลขฐาน 16) ไปยังพอร์ท 0309h (เลขฐาน 16) ตามลำดับ เมื่อส่งค่าเหล่านี้เสร็จเรียบร้อยแล้วก็จะส่งค่าไปให้ยัง สเตปปิ้งมอเตอร์ โดยค่าที่ส่งนี้จะมีค่า 20 (เลขฐาน 16) เพื่อทำการสั่งให้สเตปปิ้งมอเตอร์ทำงาน

เมื่อโปรแกรมทำการส่งค่าไปยังสเตปปิ้งมอเตอร์แล้วโปรแกรมจะทำการเช็ค เพื่อหาค่า โดยค่าที่เช็คนี้จะต้องมีค่า 0Fh (เลขฐาน 16) เมื่อได้ค่า 0Fh (เลขฐาน 16) โปรแกรมจะทำการเพิ่มค่าของตำแหน่งข้อมูล เพื่อที่จะนำข้อมูลตัวต่อไปเข้ามาทำงานอีก

4.2 การทดลองวงจรอินเตอร์เฟส

การทดลองโดยใช้โปรแกรมทดสอบในภาคผนวกโดยการส่งข้อมูลออกที่ แอดแดรส 308 h จะผ่านวงจรแปลงดิจิทอลเป็นอนาล็อกโดยทำการวัดไฟกระแสตรงได้ตั้งแต่ 0-12 โวลท์ โดยทำการเปลี่ยนข้อมูลตั้งแต่ 00h – FFh ไฟส่วนนี้จะส่งไปวงจรขับดีซีมอเตอร์ และส่งข้อมูลออกที่แอดแดรส 309 h ไฟกระแสตรงได้ตั้งแต่ 0-5 โวลท์ จะเปลี่ยนเกณฑ์ของตัวควบคุมพีแอดแดรส 30b h จะนำข้อมูลจากมอเตอร์ผ่านวงจรนับ 8บิท มาเปรียบเทียบ กับค่าที่ส่งออกไป ถ้าเท่ากันจะส่ง 00h ออกที่แอดเดรส 309h ทันที พร้อมส่งค่าออกไปตัดไฟเบรคของมอเตอร์ และลบข้อมูลเดิมของวงจร



รูปที่ 4.2 วงจรอินเตอร์เฟส

ภายนอกทั้งนี้จะส่งค่าไปให้สเตปปิ้งทำงาน เมื่อสเตปปิ้งทำงานเสร็จจะส่งค่า 0Fh มาเข้าทาง พอร์ทแอดเดรส 30C h เพื่อให้โปรแกรมรันค่าต่อไปได้

ค่าที่ส่งออกไปที่พอร์ทแอดเดรส 308 h จะเป็นการปรับมุมของดีซีมอเตอร์เพื่อไปยังตำแหน่งตัวโน้ตโดยการเปลี่ยนตำแหน่งควบคุมมอเตอร์ไปทางซ้ายหรือขวาจะควบคุมโดยบิทที่ 8 คือ D7 จะให้ค่า logic 1 หมุนทางซ้าย logic 0 หมุนทางขวา โดยรูปที่ 4.2 เป็นวงจรอินเตอร์เฟสในการควบคุมแขนกล

4.2.1 การทดสอบค่าแรงดันจุดต่างๆ

การทดสอบวัดแรงดันที่ออกจากวงจรอินเตอร์เฟส โดยที่พอร์ท 308h จะมีค่าแรงดันออกต่างกันเนื่องจากการให้ข้อมูลที่ต่างกันโดยการกดตัวโน้ตต่างๆ โปรแกรมจะนำค่าแรงดันออกดังตารางต่อไปนี้

สภาวะที่แล้ว	สภาวะปัจจุบัน	ค่าแรงดัน	แรงดันกำหนดทิศทาง
โด	เร	1 โวลท์	0 โวลท์
เร	มี	1 โวลท์	0 โวลท์
มี	ที	3 โวลท์	0 โวลท์
ที	โด	4 โวลย์	5 โวลท์
โด	โด๋	4.5 โวลท์	0 โวลท์
ลา	ฟา	2 โวลท์	5 โวลท์
ซอล	มี	1.5 โวลท์	5 โวลท์

ตารางที่ 4.1 แสดงระดับแรงดันที่ระยะห่างของตัวโน้ต

ค่าแรงดันที่แสดงในตารางที่ 4.1 เป็นค่าแรงดันที่ส่งไปยังวงจรที่ควบคุมการทำงานของดีซีเซอร์โวมอเตอร์ ลักษณะแรงดันที่ต่างกันเพื่อเวลาการเล่นตัวโน้ตของแขนกล จะมีเวลาที่ใกล้เคียงกัน สรุปคือแรงดันจะมีค่ามากตามระยะที่แขนกลเคลื่อนที่

ส่วนพอร์ท 30A h จะเป็นค่าแรงดันที่สามารถปรับได้ตั้งแต่ 0.5 โวลท์ ถึง 4 โวลท์ เพื่อนำไปปรับค่าของตัวควบคุมพี ซึ่งเป็นอุปกรณ์ภายนอกที่เปลี่ยนแปลงค่าความต้านทานตามแรงดัน ที่พอร์ท 309 h จะส่งค่าเป็นค่าต่างๆดังนี้

ส่งค่า (เลขฐานสิบหก)	หน้าที่
C0 h	หยุดดีซีมอเตอร์และ เคลียร์ค่าต่างๆ
A0 h	หยุดดีซีมอเตอร์และ สั่งสเตปปิ้งมอเตอร์ทำงาน
40 h	เคลียร์ค่าต่างๆ พร้อมที่จะทำงานต่อไป

ตารางที่ 4.2 ค่าที่ออกจากพอร์ท 309 h

จากตารางที่ 4.2 ค่าที่ส่งออก 40 h ก่อนส่งค่านี้จะทำการรับค่าจากพอร์ท 30C h ที่เป็นค่าเท่ากับ 0F h ค่านี้จะบอกว่า สเตปปิ้งทำงานเสร็จแล้ว

บทที่5
สรุปผลและวิจารณ์ผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดลอง ในส่วนของโปรแกรม การนำข้อมูลออกที่ได้กล่าวแล้วว่าจะคงค่าข้อมูลเอาไว้ และระดับแรงดันที่ออกไปให้ดีซีมอเตอร์จะไม่เท่ากัน ซึ่งแรงดันจะแปรผันไปตามระยะทางที่มอเตอร์จะเคลื่อนตัวไป (คือระยะทางมากแรงดันจะมากตาม) ในขณะเดียวกันที่มอเตอร์ทำงานจะรับค่าขนาด 8 บิท เข้ามาเพื่อนำไปเปรียบเทียบกับตำแหน่งต่างๆ ว่าสั่งไปทางซ้ายหรือทางขวาและไปไกลกี่ตัวโน้ต ถ้าเท่ากันกับระดับที่อ้างอิงกับระยะทางจริงที่เคลื่อนที่ โปรแกรมจะสั่งให้ออกค่าเป็น 0 โวลท์ ที่ไปขับมอเตอร์

เมื่อระยะทางที่มอเตอร์เคลื่อนที่ไปได้เท่ากับระยะทางที่กำหนดจะมีพอร์ต สั่งไปตัดไฟเบรคของมอเตอร์ เพื่อลดแรงเฉื่อยในการที่มอเตอร์จะหยุดเลย ขณะเดียวกัน จะสั่งค่าออกไปลบข้อมูลของวงจรภายนอกและสั่งงานให้ สเตปปิ้งมอเตอร์ ซึ่งเป็นตัวเคาะเพื่อทำให้เกิดเสียง 1 ครั้ง สเตปปิ้งทำงานครบ 1 ครั้ง คือ ทำการตีลงและยกขึ้น จะมีค่าเข้าพอร์ต คือ ค่า 0Fh โปรแกรมจะรับค่าเข้าไปเพื่อรับรู้ว่าทำงานเสร็จโปรแกรมจะสั่งลบค่าทุดค่าที่ส่งออกไป และรอรับคำสั่งต่อไป(ในโหมดควบคุมโดยผู้ใช้) โปรแกรมจะทำงานส่งอย่างต่อเนื่อง (ในโหมดอัตโนมัติ)

ในการกำหนดค่าเป็นตัวโน้ต ผู้จัดทำได้ให้การกำหนดดังนี้เฉพาะโหมดอัตโนมัติ

โด	=	1
เร	=	2
มี	=	3
ฟาร์	=	4
ซอล	=	5
ลา	=	6
ซี	=	7
โด	=	8
จบการทำงาน	=	0

ในโหมดอัตโนมัตินี้ จะสามารถใส่โน้ตเพลงได้ทันที ทีละหลายตัวโน้ตโดยโปรแกรมจะสั่งทำงานจนจบตามข้อมูลที่ใส่ ต่างกับโหมดธรรมดาซึ่งต้องคลิกด้วย เม้าส์เองขณะที่ต้องการโน้ตตัวใดในเพลงนั้น

การทำงานของวงจรอินเตอร์เฟส จากบทที่ 3 ที่ออกแบบไว้ใช้ แอดเดรสตั้งแต่ 308h – 30Ch โดยแบ่งเป็นแอดเดรส 302h – 30Ah จะนำข้อมูลเข้า คณะผู้จัดทำได้มีการปรับเกนของ ตัวควบคุมพี(K_p)จากโปรแกรมในที่นี้ไว้ปรับเกนเพื่อเร่งความเร็วของการเล่นเพลง ถ้าค่าเกนมากจะเล่นไวกว่าการปรับเกนน้อย

5.2 วิจารณ์ผลการทดลอง

การทดลองจะติดปัญหาเนื่องจากอุปกรณ์ที่ติดต่อนำข้อมูลเข้าออก ผู้จัดทำใช้อุปกรณ์ไอซี TTL จะมีปัญหากับอุปกรณ์ที่มีหน้าสัมผัส เช่น รีเลย์ และการตัดไฟเบรคของมอเตอร์ เพราะการทำงานดังกล่าว จะทำให้บิทผิดพลาดในการส่งและรับ ดังนั้นผู้จัดทำจึงต้องแก้โปรแกรม ให้ส่งค่าวนออกขณะอุปกรณ์ที่มีหน้าสัมผัสทำงาน เพื่อรักษาการค้างข้อมูลไว้ให้คงเดิมตลอด

ภาคผนวก ก

คู่มือการใช้ Program

1. ก่อนการใช้งาน

การต่ออินเตอร์เฟสเข้ากับ Main Broad

การต่ออินเตอร์เฟส เข้ากับ Main Broad นั้นจะต้องต่ออินเตอร์เฟสเข้าที่สล็อตของ IBM/PC

- ข้อสังเกต สล็อตของ IBM/PC จะมีขนาดที่ยาวกว่าสล็อตอื่น ๆ โดยจะมี 64 ขา

2. การใช้งาน

2.1 การเปิดโปรแกรม

การเปิดโปรแกรมนั้นจะต้องเลื่อนเมาส์ไปที่ Icon ของโปรแกรมดังรูป และทำการ ดับเบิลคลิกที่ Icon นั้น ก็จะทำให้โปรแกรมเปิดออกมาให้ใช้งาน

Robot

รูปแบบ Icon ที่ใช้งาน

2.2 การปิดโปรแกรม

การปิดโปรแกรม จะต้องเลื่อน เมาส์ ไปที่เมนูที่ชื่อ File แล้วคลิก จะทำให้มีเมนูย่อย ที่ชื่อ Exit ออกมา เราก็จะต้องเลื่อนเมาส์ ไป และทำการคลิกที่ Exit โปรแกรมก็จะทำการเปิดฟอร์ม ดังรูป โดยในฟอร์มนั้นจะถามคุณว่า " คุณต้องการจะปิดโปรแกรมหรือไม่ " ถ้าคุณต้องการปิดโปรแกรมให้คุณเลื่อน เมาส์ มา คลิก ที่คำว่า " ใช่ " แต่ถ้าคุณยังไม่ต้องการที่จะปิดให้คุณเลื่อนเมาส์ มา คลิก ที่คำว่า "ไม่ "

2.3 การใช้ Help

การใช้ Help เราจะเลื่อนเมาส์ไปที่ เมนู ที่ชื่อว่า Help และที่ Help จะมีตัวให้เลือก 2 ตัวคือ Content และ About โดยให้เลือกไปที่ Content ก็จะเปิด ฟอร์ม help ออกมา ส่วน About จะเป็นตัวรายละเอียด

3. การใช้โปรแกรม

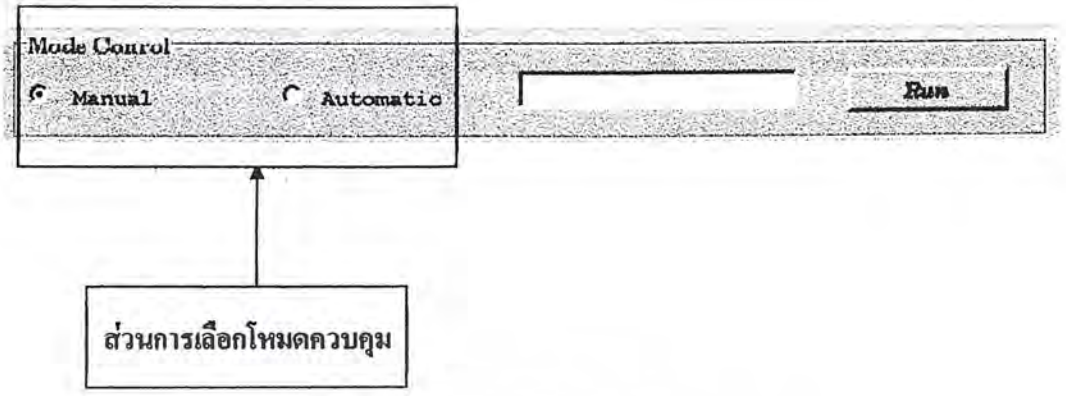
ในโปรแกรมตัวนี้ถูกออกแบบไว้ให้ทำงานได้ 2 โหมดการทำงาน โดยโหมดการทำงานจะแบ่งได้เป็น

1. โหมดควบคุมด้วยตัวผู้ใช้
2. โหมดอัตโนมัติ

เราจะมากล่าวถึงการใช้งานทีละ โหมด ดังนี้ คือ

3.1 การใช้งานของ โหมดควบคุมโดยผู้ใช้

1. ก่อนที่จะใช้ใน โหมดควบคุมด้วยตัวผู้ใช้เอง เราจะต้องดูว่าตอนนี้โปรแกรมอยู่ใน โหมด ใด โดยการดูจากตำแหน่งของช่องที่อยู่หน้าคำว่า Manual ถ้าเป็นสีดำ นั้นก็แสดงว่าอยู่ในโหมดควบคุมด้วยตัวผู้ใช้เอง แต่ถ้าไม่มี ก็ให้เลื่อน เมาส์ ไปที่ช่องด้านหน้าคำว่า Manual ที่อยู่ในส่วนของ โหมดควบคุม แล้วให้คลิกที่ช่องนั้นให้เป็นสีดำ เพื่อแสดงว่าโปรแกรมได้เข้าสู่ระบบโหมดควบคุมด้วยตัวผู้ใช้

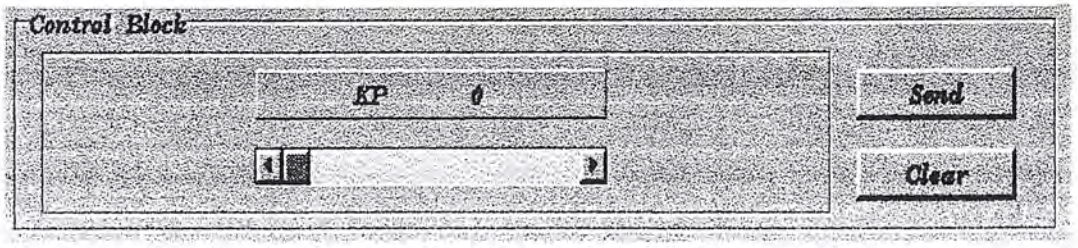


2. เมื่อเข้าสู่ระบบ โหมดควบคุมด้วยตัวผู้ใช้แล้ว เราก็จะต้องทำการส่งค่าของ K_p ในค่า โดย ทำการปรับค่า K_p นั้นจะแบ่งได้ออกเป็น 2 วิธี

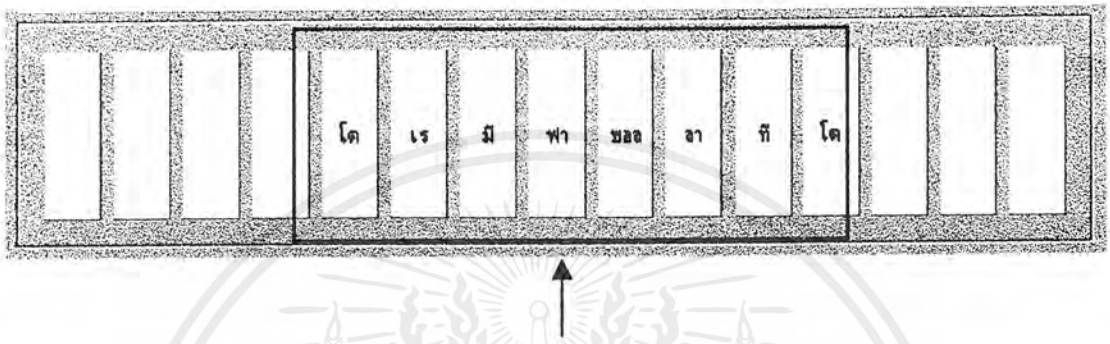
1. การปรับค่าทีละมาก ๆ โดยการรับจะทำได้โดยการเลื่อน เมาส์ ไปที่บล็อก ระหว่าง Scroll Bar และทำการ คลิก เมาส์ ค้างไว้ และทำการเลื่อน เมาส์ ให้ได้ตามค่าที่ต้องการ โดยการดูค่าให้ดูจาก ตัวเลขข้าง ๆ ตัว K_p ที่ทำการปรับ

2. การปรับค่าทีละหนึ่ง โดยการรับค่าจะทำได้โดยการเลื่อน เมาส์ ไปที่ Scroll Bar โดยที่หัวและท้ายของ Scroll Bar จะมีตัวที่เป็นรูป และ เมื่อต้องการจะเพิ่มค่าก็ให้เลื่อน เมาส์ ไปที่รูป แล้วทำการ คลิก ก็จะทำให้ค่าเพิ่มไป 1 หรือถ้าต้องการลดค่า ให้ไปที่ และทำการ คลิก ก็จะทำให้ค่าลดลง 1 เหมือนกัน

เมื่อทำการรับค่าของ K_p เสร็จเรียบร้อยแล้ว เราก็จะต้องทำการส่ง การส่งนั้นทำได้โดยการเลื่อน เมาส์ ไปที่ปุ่มที่มีชื่อว่า Send และทำการ คลิก ที่ปุ่มนั้น ค่า K_p ที่กำหนดไว้ก็จะถูกส่งออกไปยัง Port ที่กำหนดไว้



3. เมื่อทำข้อ 1, 2 เสร็จ เราก็จะทำการกด คีย์ โดยการเลื่อน เมาส์ มาที่คีย์ที่เราต้องการจะกด โดยที่ คีย์ จะแทนตัวโน้ตแต่ละตัว แต่ในโปรแกรมนี้ทำการกำหนด คีย์ ที่ทำงานได้ไว้ 8 คีย์ เท่านั้น ถ้ากดนอกเหนือที่กำหนดโปรแกรมก็จะไม่ทำงาน



ส่วนที่สามารถทำงานเป็นตัวโน๊ตได้

3.2 การใช้งานของ โหมดอัตโนมัติ

1 การใช้ โหมดอัตโนมัติ เราจะต้องทำการให้โปรแกรมเข้าสู่ โหมดอัตโนมัติ ก่อน โดยการเลื่อนเมาส์ ไปคลิก ที่ช่องสีขาวที่อยู่ด้านหน้าคำว่า Automatic ที่อยู่ในส่วนของ โหมดควบคุม ให้ปรากฏจุดสีดำอยู่ในช่องนั้น เพื่อที่จะแสดงว่าโปรแกรมเข้าสู่โหมดอัตโนมัติแล้ว

2. ทำการป้อนค่าที่ต้องการ โดยการป้อนค่าจะป้อนค่าลงที่ ช่องสี่เหลี่ยมผืนผ้าสีขาวที่อยู่ด้านหลังของคำว่า Automatic โดยการป้อนค่านั้น จะทำได้โดยป้อนค่าเป็นตัวเลข โดยที่ค่าตัวเลขที่จะป้อนนั้นจะมีค่าตั้งแต่ 1 – 8 โดยค่าตัวเลขเหล่านี้จะแทนค่าตัวโน้ต เช่นตัวเลข 1 จะแทนค่าตัวโน้ต โด เป็นต้น ตัวเลขจะแทนค่าตัวโน้ตไล่ไปเรื่อย ๆ การใส่ตัวเลขนั้นจะต้องใส่ค่าตัวเลขหนึ่งตัวแล้วเว้นหนึ่งช่องโดยการเคาะ space bar แล้วจึงใส่ตัวเลขอีกหนึ่งตัวได้ ตัวอย่างเช่น ใส่ 1 2 เป็นต้น

3. เมื่อทำการใส่ค่าที่ต้องการเรียบร้อยแล้ว เราก็จะสั่งให้โปรแกรมทำงาน ทำโดยการเลื่อน เมาส์ ไปที่ปุ่มที่มีชื่อว่า Run ที่อยู่ข้าง ๆ ช่องใส่ค่าตัวโน้ตโปรแกรมจะทำงาน

ภาคผนวก ข

โปรแกรมหลัก

```
unit good;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, ExtCtrls;

const
  Slow = 100000000;
  Sec = 70000000;
  ControlBreak1 = 192;
  ControlBreak2 = 64;
  ControlBreak3 = 0;
  Step = 1;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
```

```
Panel10: TPanel;
Panel11: TPanel;
Panel12: TPanel;
Panel13: TPanel;
Panel14: TPanel;
Panel15: TPanel;
GroupBox2: TGroupBox;
GroupBox3: TGroupBox;
Button1: TButton;
Button2: TButton;
MainMenu1: TMainMenu;
File1: TMenuItem;
ScrollBar1: TScrollBar;
Panel25: TPanel;
Label1: TLabel;
Exit1: TMenuItem;
RadioGroup1: TRadioGroup;
Edit1: TEdit;
Button3: TButton;
Help1: TMenuItem;
Content1: TMenuItem;
About1: TMenuItem;
Timer1: TTimer;
Timer2: TTimer;
Timer3: TTimer;
Timer4: TTimer;
procedure ScrollBar1Change(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Panel5Click(Sender: TObject);
procedure Panel6Click(Sender: TObject);
```

```
    procedure Panel7Click(Sender: TObject);
    procedure Panel8Click(Sender: TObject);
    procedure Panel9Click(Sender: TObject);
    procedure Panel10Click(Sender: TObject);
    procedure Panel11Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Panel12Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure Timer3Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer4Timer(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  KI,KP,K:Integer;
  Stop,Start,Index,Index1 :Integer;
  Chack,StepStop,ChackStep,ChackKey,Mov:Word;
  StartKey:Integer;
  S:String;
implementation

{$R *.DFM}
```

```
function Inport (port:Word):byte;
begin
  asm
  mov dx,port
  in al,dx
  mov result,al
  end
end;

procedure OutPort(Address,Data:Word);Assembler
begin
   asm
   push dx
   push ax
   mov dx,Address
   mov ax,Data
   out dx,ax
   pop ax
   pop dx
   end
 end;

procedure Delay(D:Integer);
var De:integer;
begin
 De:=0;
 While De<D do
 De := De+1;
end;
```

```
procedure Move(Start,Stop:Integer);
begin
  if Stop >= Start then
   begin
     Mov := Stop - Start;
     OutPort ($0378,Mov);
   end
  else
   begin
     Mov := Start - Stop;
     OutPort ($0378,Mov);
   end;
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
 KP:=ScrollBar1.Position;
 Label1.Caption :=InttoStr(ScrollBar1.Position);
end;


procedure TForm1.Button1Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   OutPort($0378,KP);
   end;
end;
```

```
procedure TForm1.Panel5Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 10;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;

procedure TForm1.Panel6Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 20;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;

procedure TForm1.Panel7Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 30;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;
```

```
procedure TForm1.Panel8Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 40;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;

procedure TForm1.Panel9Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 50;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;

procedure TForm1.Panel10Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 60;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;
```

```
procedure TForm1.Panel11Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 70;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;

procedure TForm1.Panel12Click(Sender: TObject);
begin
  if RadioGroup1.ItemIndex = 0 then
  begin
   Stop := 80;
   Move(Start,Stop);
   Start:=Stop;
  end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
 if ChackKey = 10 then
  begin
  Chack:=InPort($030a);
  if Chack = Mov then
  begin
   OutPort($0308,00);
   OutPort($0309,ControlBreak1);
   Delay(Sec);
```

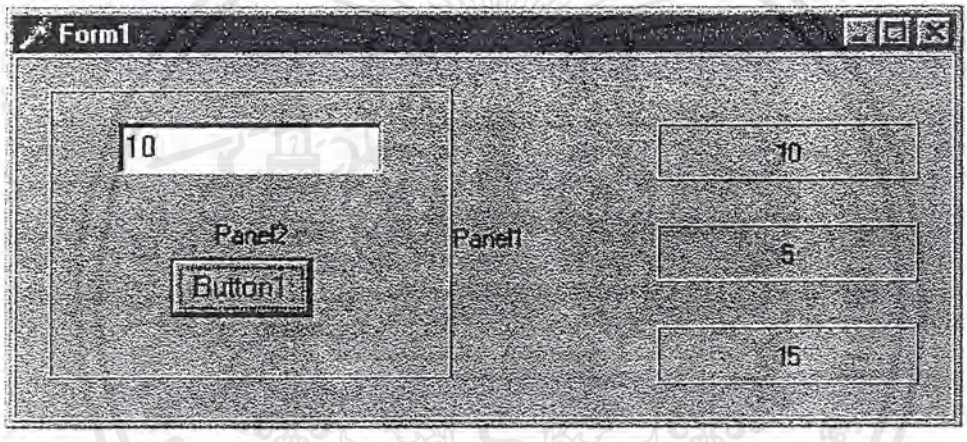
```
    OutPort($0309,ControlBreak2);
    Delay(Sec);
    OutPort($0309,ControlBreak3);
    Delay(Sec);
    OutPort($0309,Step);
    Mov := 00;
   end;
  end;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
var Num :Integer;
begin
 if Index < 11 then
  begin
  if Index <> Index1  then
   begin
     Num :=StrToInt(S[Index]);
     case Num of
      1:Stop:=10;
      2:Stop:=20;
      3:Stop:=30;
      4:Stop:=40;
      5:Stop:=50;
      6:Stop:=60;
      7:Stop:=70;
      8:Stop:=80;
     end;
    Move(Start,Stop);
   end;
```

```
  Chack :=InPort($030a);
  if Chack = Mov then
  begin
   OutPort($0308,00);
   OutPort($0309,ControlBreak1);
   Delay(Sec);
   OutPort($0309,ControlBreak2);
   Delay(Sec);
   OutPort($0309,ControlBreak3);
   Delay(Sec);
   OutPort($0309,Step);
   Mov := 00;
   Index1:=Index;
  end;
  end;
 end;

procedure TForm1.Button3Click(Sender: TObject);
begin
 if RadioGroup1.ItemIndex = 1 then
   begin
    OutPort($0378,KP);          {send KP}
    Index := 1;
    S:=Edit1.Text;
    Index := 1;
    Index1 := 11;
   end;
end;
```

```
procedure TForm1.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.Timer3Timer(Sender: TObject);
begin
  ChackStep := InPort($0378);
  if ChackStep = 1 Then
  begin
    Index := Index + 2;
    Start:=Stop;
  end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  ChackKey := 10;
end;

procedure TForm1.Timer4Timer(Sender: TObject);
begin
  ChackKey := InPort($0378);
  if ChackKey = $0F then
    begin
      StartKey := 10;
    end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
```

```
  KP:= 0 ;
  KP:=ScrollBar1.Position;
  Label1.Caption :=InttoStr(ScrollBar1.Position);
  OutPort($0378,KP);
end;

end.
```

โปรแกรมทดสอบ



รูปโปรแกรมทดสอบ

```
program nook1;

uses
  Forms,
  nook in 'nook.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
```

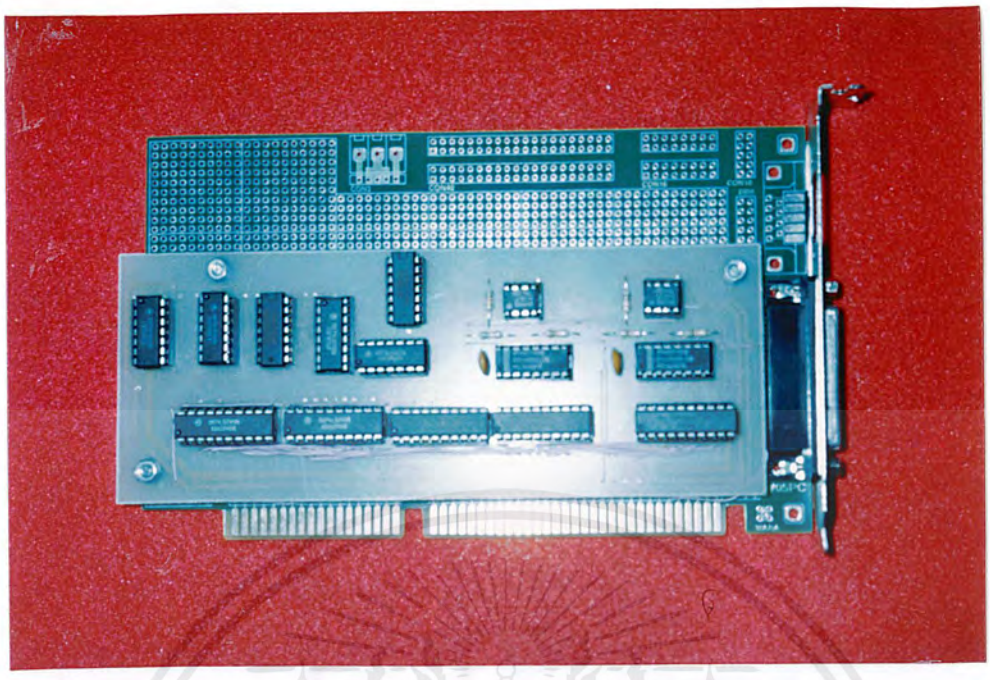
```
end.
unit nook;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Edit1: TEdit;
    Button1: TButton;
    Panel3: TPanel;
    Panel4: TPanel;
    Timer1: TTimer;
    Timer2: TTimer;
    Panel5: TPanel;
    procedure Button1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  { procedure Timer2Timer(Sender: TObject);  }
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1 : TForm1;
  dataout1,dataout2 : word   ;
  datain1,datain2   : byte   ;
```

```
const
  port1 : word = $308 ;
  port2 : word = $30A ;
  port3 : word = $309 ;
  port4 : word = $30B ;
  dstop : word = $00  ;
  dinit : word = $0F  ;
  ddelay1:word = $c0  ;
  ddelay2:word = $40  ;
  ddelay3:word = $20  ;

implementation

{$R *.DFM}


function InPort( port:word ): byte;
begin
 asm
  mov  dx,port
  in   al,dx
  mov  result,al
end;
end;


procedure OutPort(addr,data:word);Assembler;
begin
 Asm
  Push  dx;
  Push  ax;
  Mov   dx,addr;
  Mov   ax,data;
  Out   dx,ax;
```

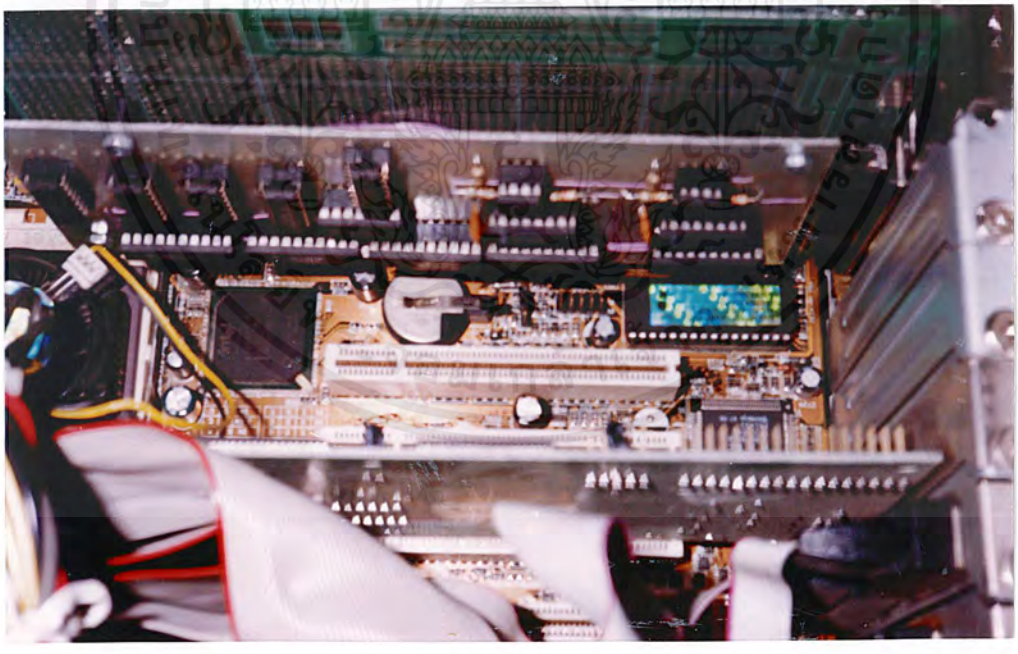
```
    Pop   ax;
    Pop   dx;
  end;
end;
{Ex.  Give output port is address 378h and data is 0ffh
     Send data to output port
Sol.
      var add,da:word;
      begin
        add:=$378;
        da:=$0ff;
        OutPort(add,da);
      end;
}

function Delay(dtime : integer):integer;
begin
    WHILE dtime > 0 DO
      dtime := dtime - 1 ;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
    dataout1 := StrToInt(Edit1.Text);
    OutPort(port1,dataout1);
    Panel3.Caption := IntToStr(dataout1);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
     datain1 := InPort(port2);
     Panel4.Caption := IntToStr(datain1);
     IF datain1 = dataout1 THEN
       begin
```

```
        Panel2.Enabled := False ;
        OutPort(port1,dstop);
        OutPort(port3,ddelay1);
        delay(100000000);
        OutPort(port3,ddelay2);
        delay(100000000);
        OutPort(port3,ddelay3);
        Timer2.Enabled := True  ;
      end
      ELSE
         Timer2.Enabled := False ;
         Panel5.Caption := 'Panel5';
    datain2 := InPort(port4);
    Panel5.Caption := IntToStr(datain2);
    IF datain2 = dinit THEN
      begin
        Panel2.Enabled := True  ;
      end;
   dataout1 := 260 ;
end;

{procedure TForm1.Timer2Timer(Sender: TObject);
begin

    datain2 := InPort(port4);
    Panel5.Caption := IntToStr(datain2);
    IF datain2 = dinit THEN
      begin
        Panel2.Enabled := True  ;
      end;
end; }

end.
```

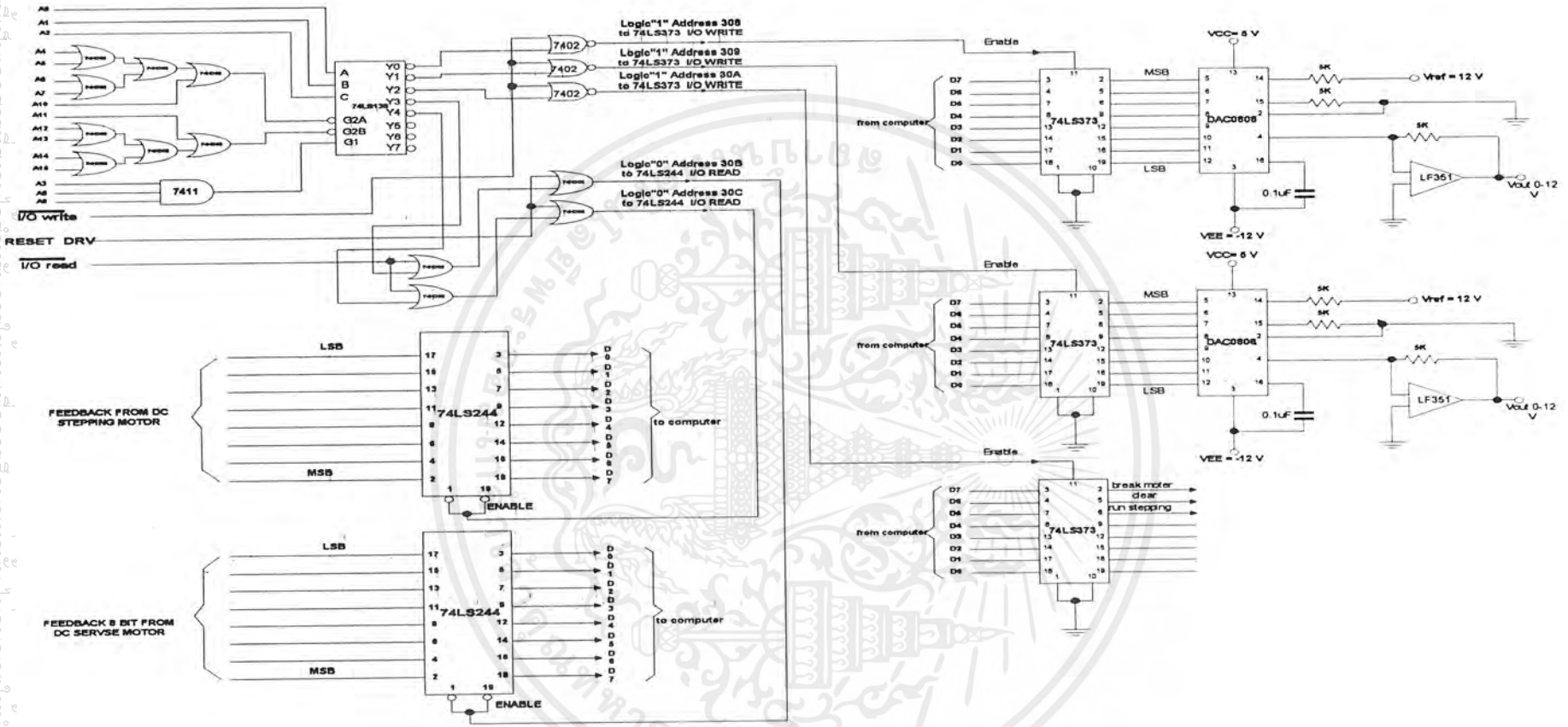
ภาคผนวก ค



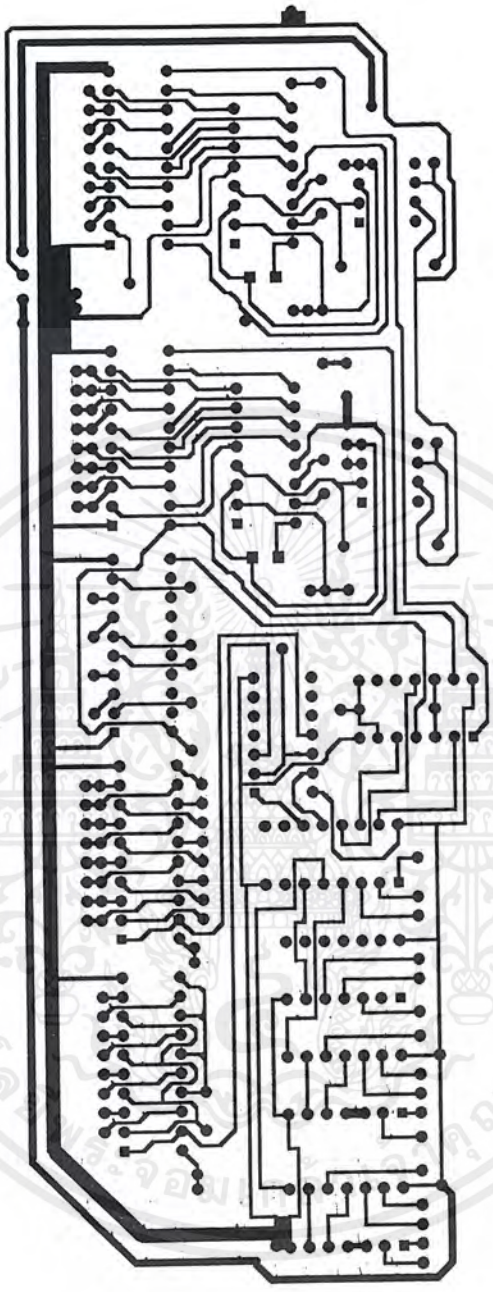
รูปการ์ดอินเตอร์เฟสที่ใช้งานจริง



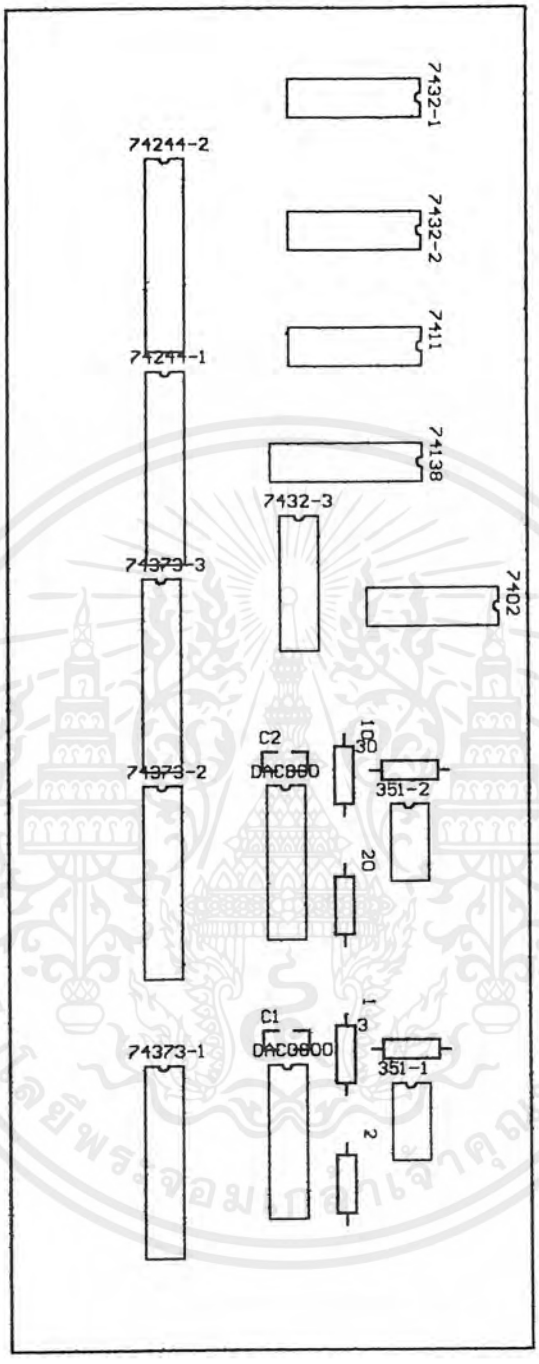
รูปการ์ดอินเตอร์เฟสที่ใช้งานบน SLOT IBM/PC (ISA SLOT)



วงจรอินเตอร์เฟส



รูปลายทองแดงวงจรอินเตอร์เฟส



รูปแสดงรายการอุปกรณ์และการวางอุปกรณ์ด้านบนวงจรอินเตอร์เฟส

8-bit multiplying D/A converter

MC1508-8/1408-8

DESCRIPTION

The MC1508/MC1408 series of 8-bit monolithic digital-to-analog converters provide high-speed performance with low cost. They are designed for use where the output current is a linear product of an 8-bit digital word and an analog reference voltage

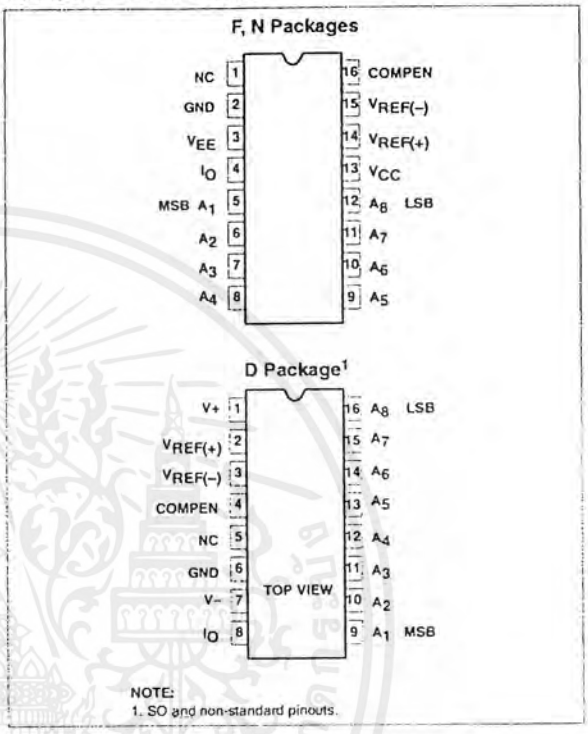
FEATURES

- Fast settling time — 70ns (typ)
- Relative accuracy ±0.19% (max error)
- Non-inverting digital inputs are TTL and CMOS compatible
- High-speed multiplying rate 4.0mA/µs (input slew)
- Output voltage swing +0.5V to –5.0V
- Standard supply voltages +5.0V and –5.0V to –15V
- Military qualifications pending

APPLICATIONS

- Tracking A-to-D converters
- 2 1/2-digit panel meters and DVMs
- Waveform synthesis
- Sample-and-Hold
- Peak detector
- Programmable gain and attenuation
- CRT character generation
- Audio digitizing and decoding
- Programmable power supplies
- Analog-digital multiplication
- Digital-digital multiplication
- Analog-digital division
- Digital addition and subtraction
- Speech compression and expansion
- Stepping motor drive modems
- Servo motor and pen drivers

PIN CONFIGURATIONS

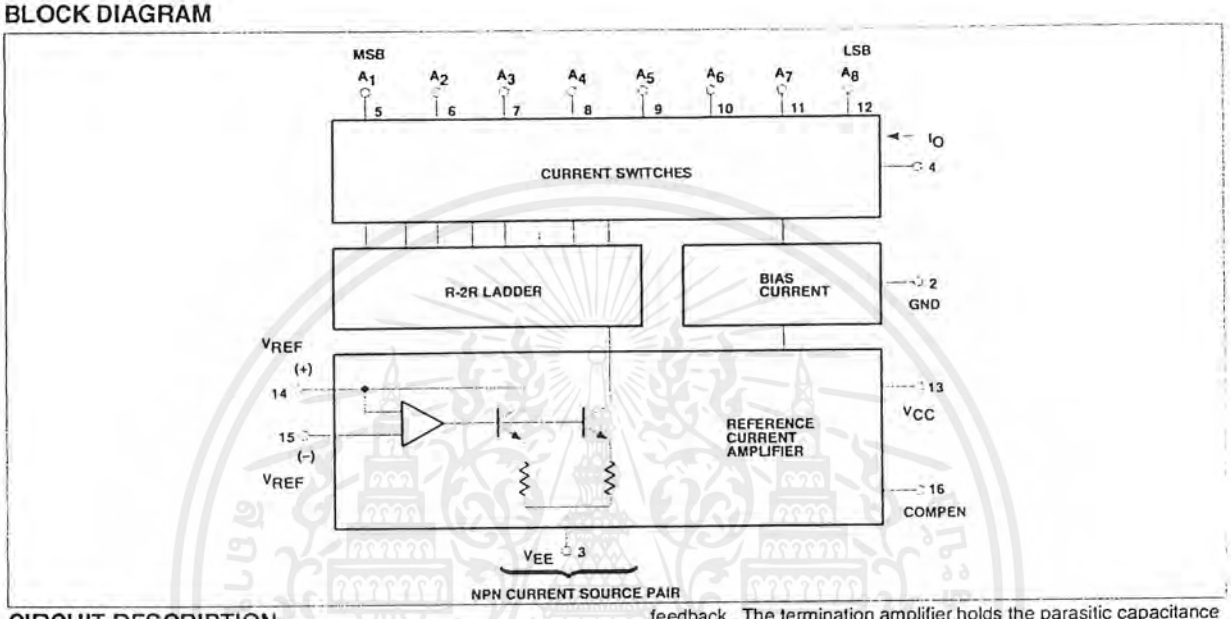


NOTE:
1. SO and non-standard pinouts.

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
16-Pin Ceramic Dual In-Line Package (CERDIP)	-55 to +125°C	MC1508-8F	0582B
16-Pin Ceramic Dual In-Line Package (CERDIP)	0 to +70°C	MC1408-8F	0582B
16-Pin Plastic Dual In-Line Package (DIP)	0 to +70°C	MC1408-8N	0406C
16-Pin Small Outline (SO) Package	0 to +70°C	MC1408-8D	0005D

BLOCK DIAGRAM



CIRCUIT DESCRIPTION

The MC1508/MC1408 consists of a reference current amplifier, an R-2R ladder, and 8 high-speed current switches. For many applications, only a reference resistor and reference voltage need be added.

The switches are non-inverting in operation; therefore, a high state on the input turns on the specified output current component.

The switch uses current steering for high speed, and a termination amplifier consisting of an active load gain stage with unity gain feedback. The termination amplifier holds the parasitic capacitance of the ladder at a constant voltage during switching, and provides a low impedance termination of equal voltage for all legs of the ladder.

The R-2R ladder divides the reference amplifier current into binarily-related components, which are fed to the remainder current which is equal to the least significant bit. This current is shunted to ground, and the maximum output current is 255/256 of the reference amplifier current, or 1.992mA for a 2.0mA reference amplifier current if the NPN current source pair is perfectly matched.

ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER	RATING	UNIT
V_{CC}	Positive power supply voltage	+5.5	V
V_{EE}	Negative power supply voltage	−16.5	V
$V_5 - V_{12}$	Digital input voltage	0 to V_{CC}	V
V_O	Applied output voltage	−5.2 to +18	V
I_{14}	Reference current	5.0	mA
V_{14}, V_{15}	Reference amplifier inputs	V_{EE} to V_{CC}	
P_D	Maximum power dissipation, T_A = 25°C (still-air)[1]		
	F package	1190	mW
	N package	1450	mW
	D package	1080	mW
T_{SOLD}	Lead soldering temperature (10 sec)	300	°C
T_A	Operating temperature range	300	°C
	MC1508	−55 to +125	°C
	MC1408	0 to +75	°C
T_{STG}	Storage temperature range	−65 to +150	°C

NOTES:
1. Derate above 25°C, at the following rates: F package at 9.5mW/°C, N package at 11.6mW/°C, D package at 8.6mW/°C.

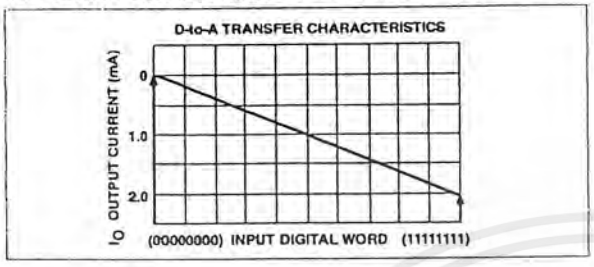
DC ELECTRICAL CHARACTERISTICS

Pin 3 must be 3V more negative than the potential to which R_{15} is returned. V_{CC} = +5.0V_{DC}, V_{EE} = −15V_{DC}, V_{REF}/R_{14} = 2.0mA unless otherwise specified. MC1508: T_A = −55°C to 125°C. MC1408: T_A = 0°C to 75°C, unless otherwise noted.

SYMBOL	PARAMETER	TEST CONDITIONS	MC1508-8			MC1408-8			UNIT
			Min	Typ	Max	Min	Typ	Max	
E_r	Relative accuracy	Error relative to full-scale I_O, Figure 3			±0.19			±0.19	%
t_S	Settling time[1]	To within 1/2 LSB, includes t_{PLH}, T_A = +25°C, Figure 4		70			70		ns
t_{PLH} t_{PHL}	Propagation delay time Low-to-High High-to-Low	T_A = +25°C, Figure 4		35	100		35	100	ns
TCI_O	Output full-scale current drift			−20			−20		ppm/°C
V_{IH} V_{IL}	Digital input logic level (MSB) High Low	Figure 5	2.0		0.8	2.0		0.8	V_{DC}
I_{IH} I_{IL}	Digital input current (MSB) High Low	Figure 5 V_{IH} = 5.0V V_{IL} = 0.8V		0 −0.4	0.04 −0.8		0 −0.4	0.04 −0.8	mA
I_{15}	Reference input bias current	Pin 15, Figure 5		−1.0	−5.0		−1.0	−5.0	µA
I_{OR}	Output current range	Figure 5 V_{EE} = −5.0V V_{EE} = −7.0V to −15V	0 0	2.0 2.0	2.1 4.2	0 0	2.0 2.0	2.1 4.2	mA
I_O $I_{O(min)}$	Output current Off-state	Figure 5 V_{REF} = 2.000V, R14 = 1000Ω All bits low	1.9	1.99 0	2.1 4.0	1.9	1.99 0	2.1 4.0	mA µA
V_O	Output voltage compliance	E_r ≤ 0.19% at T_A = +25°C, Figure 5 V_{EE} = −5V V_{EE} below −10V		−0.6 +10 −5.5, +10	−0.55, +0.5 −5.0, +0.5		−0.6 +10 −5.5, +10	−0.55, +0.5 −5.0, +0.5	V_{DC}
SRI_{REF}	Reference current slew rate	Figure 6		8.0			8.0		mA/µs
PSRR(−)	Output current power supply sensitivity	I_{REF} = 1mA		0.5	2.7		0.5	2.7	µA/V
I_{CC} I_{EE}	Power supply current Positive Negative	All bits low, Figure 5		+2.5 −6.5	+22 −13		+2.5 −6.5	+22 −13	mA
V_{CCR} V_{EER}	Power supply voltage range Positive Negative	T_A = +25°C, Figure 5	+4.5 −4.5	+5.0 −15	+5.5 −16.5	+4.5 −4.5	+5.0 −15	+5.5 −16.5	V_{DC}
P_D	Power dissipation	All bits low, Figure 5 V_{EE} = −5.0V_{DC} V_{EE} = −15.0V_{DC}		34 110	170 305		34 110	170 305	mW

NOTES:
1. All bits switched.

TYPICAL PERFORMANCE CHARACTERISTICS



FUNCTIONAL DESCRIPTION

Reference Amplifier Drive and Compensation

The reference amplifier input current must always flow into Pin 14, regardless of the setup method or reference supply voltage polarity.

Connections for a positive reference voltage are shown in Figure 1. The reference voltage source supplies the full reference current. For bipolar reference signals, as in the multiplying mode, R_{15} can be tied to a negative voltage corresponding to the minimum input level. R_{15} may be eliminated and Pin 15 grounded, with only a small sacrifice in accuracy and temperature drift.

The compensation capacitor value must be increased with increasing values of R_{14} to maintain proper phase margin. For R_{14} values of 1.0, 2.5, and 5.0kΩ, minimum capacitor values are 15, 37, and 75pF. The capacitor may be tied to either V_{EE} or ground, but using V_{EE} increases negative supply rejection. (Fluctuations in the negative supply have more effect on accuracy than do any changes in the positive supply.)

A negative reference voltage may be used if R_{14} is grounded and the reference voltage is applied to R_{15}, as shown in Figure 2. A high input impedance is the main advantage of this method. The negative reference voltage must be at least 3.0V above the V_{EE} supply. Bipolar input signals may be handled by connecting R_{14} to a positive reference voltage equal to the peak positive input level at Pin 15.

Capacitive bypass to ground is recommended when a DC reference voltage is used. The 5.0V logic supply is not recommended as a reference voltage, but if a well regulated 5.0V supply which drives logic is to be used as the reference, R_{14} should be formed of two series resistors and the junction of the two resistors bypassed with 0.1µF to ground. For reference voltages greater than 5.0V, a clamp diode is recommended between Pin 14 and ground.

If Pin 14 is driven by a high impedance such as a transistor current source, none of the above compensation methods apply and the amplifier must be heavily compensated, decreasing the overall bandwidth.

Output Voltage Range

The voltage at Pin 4 must always be at least 4.5V more positive than the voltage of the negative supply (Pin 3) when the reference current is 2mA or less, and at least 8V more positive than the negative supply when the reference current is between 2mA and 4mA. This is necessary to avoid saturation of the output transistors, which would cause serious degradation of accuracy.

Philips Semiconductors MC1508/MC1408 does not need a range control because the design extends the compliance range down to 4.5V (or 8V — see above) above the negative supply voltage without significant degradation of accuracy. Philips Semiconductors MC1508/MC1408 can be used in sockets designed for other manufacturers' MC1508/MC1408 without circuit modification.

Output Current Range

Any time the full-scale current exceeds 2mA, the negative supply must be at least 8V more negative than the output voltage. This is due to the increased internal voltage drops between the negative supply and the outputs with higher reference currents.

Accuracy

Absolute accuracy is the measure of each output current level with respect to its intended value, and is dependent upon relative accuracy, full-scale accuracy and full-scale current drift. Relative accuracy is the measure of each output current level as a fraction of the full-scale current after zero-scale current has been nulled out. The relative accuracy of the MC1508/MC1408 is essentially constant over the operating temperature range because of the excellent temperature tracking of the monolithic resistor ladder. The reference current may drift with temperature, causing a change in the absolute accuracy of output current; however, the MC1508/MC1408 has a very low full-scale current drift over the operating temperature range.

The MC1508/MC1408 series is guaranteed accurate to within ±1/2LSB at +25°C at a full-scale output current of 1.99mA. The relative accuracy test circuit is shown in Figure 3. The 12-bit converter is calibrated to a full-scale output current of 1.99219mA; then the MC1508/MC1408's full-scale current is trimmed to the same value with R_{14} so that a zero value appears at the error amplifier output. The counter is activated and the error band may be displayed on the oscilloscope, detected by comparators, or stored in a peak detector.

Two 8-bit D-to-A converters may not be used to construct a 16-bit accurate D-to-A converter. 16-bit accuracy implies a total of ±1/2 part in 65,536, or ±0.00076%, which is much more accurate than the ±0.19% specification of the MC1508/MC1408.

Monotonicity

A monotonic converter is one which always provides an analog output greater than or equal to the preceding value for a corresponding increment in the digital input code. The MC1508/MC1408 is monotonic for all values of reference current above 0.5mA. The recommended range for operation is a DC reference current between 0.5mA and 4.0mA.

Settling Time

The worst case switching condition occurs when all bits are switched on, which corresponds to a low-to-high transition for all input bits. This time is typically 70ns for settling to within 1/2LSB for 8-bit accuracy. This time applies when $R_L < 500\Omega$ and $C_O < 25pF$. The slowest single switch is the least significant bit, which typically turns on and settles in 65ns. In applications where the D-to-A converter functions in a positive going ramp mode, the worst-case condition does not occur and settling times less than 70ns may be realized.

Extra care must be taken in board layout since this usually is the dominant factor in satisfactory test results when measuring settling time. Short leads, 100µF supply bypassing for low frequencies, minimum scope lead length, good ground planes, and avoidance of ground loops are all mandatory.

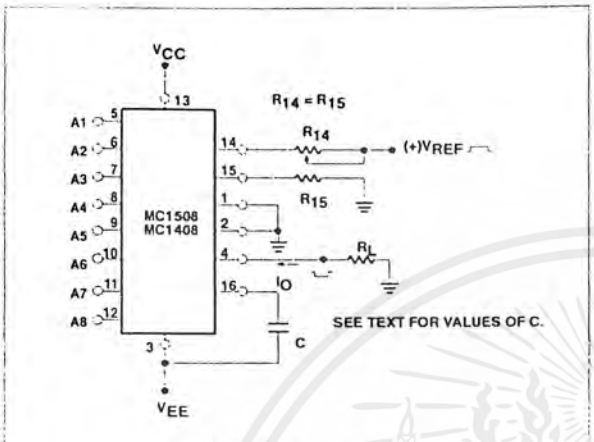


Figure 1. Positive V_{REF}

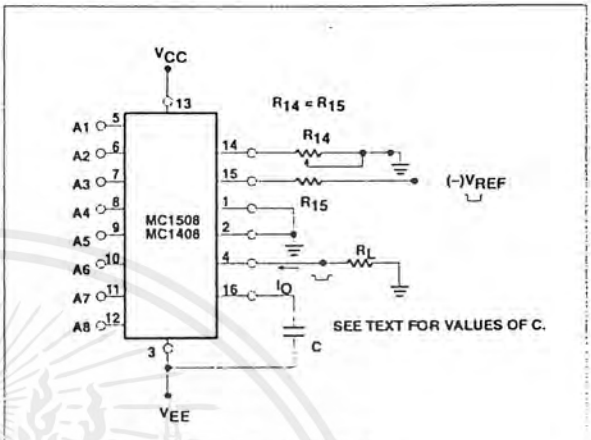


Figure 2. Negative V_{REF}

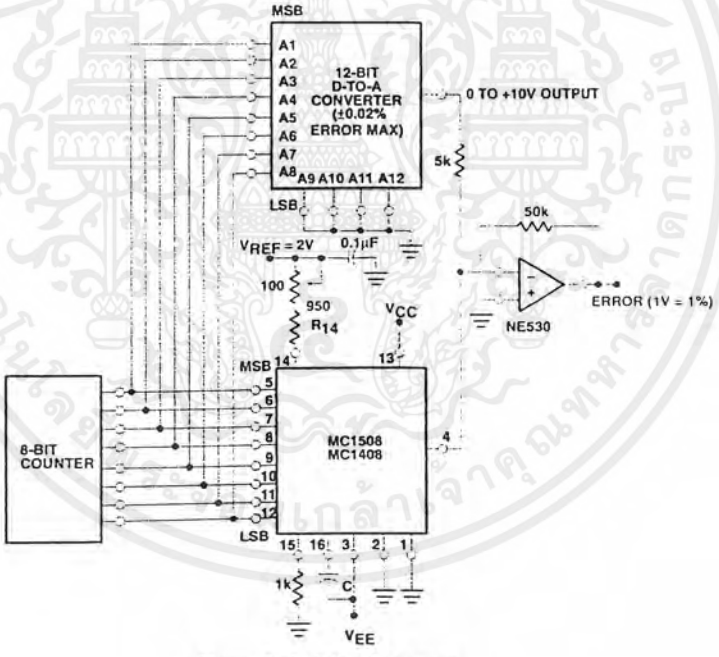


Figure 3. Relative Accuracy

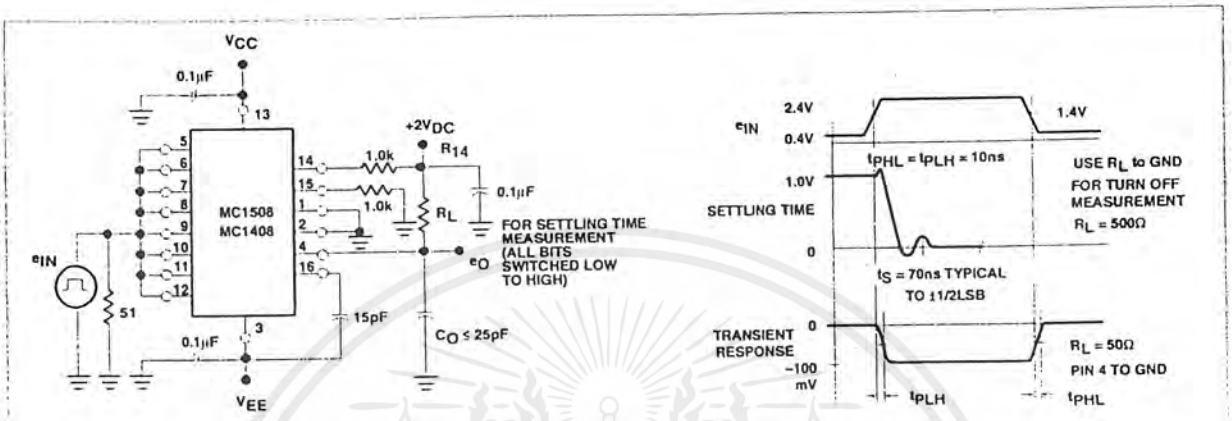


Figure 4. Transient Response and Settling Time

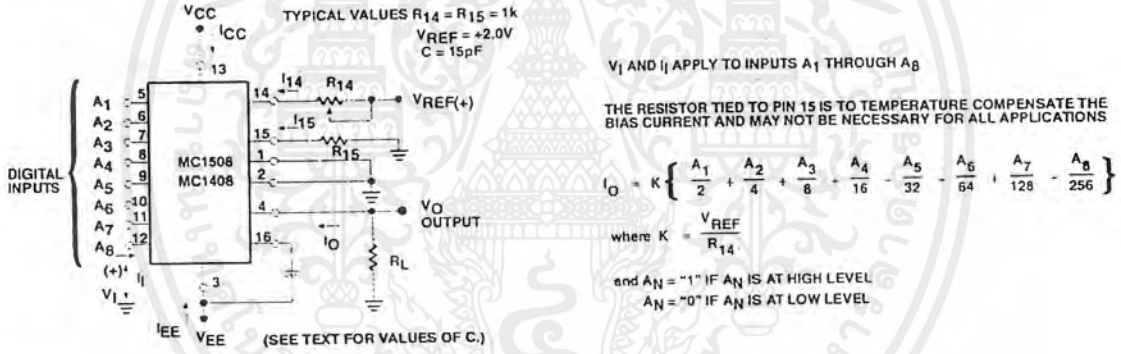


Figure 5. Notation Definitions

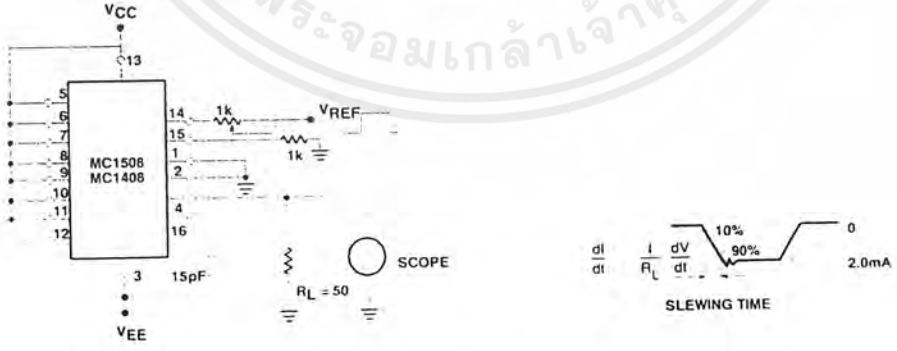


Figure 6. Reference Current Slew Rate Measurement

บรรณานุกรม

Hemanshu R.Pota, A Prototype flexible robot arm-an Interdisciplinary Undergraduate project , *IEEE transactions on education*. Vol 35 Feb 1992.

Fumitoshi Masuno,Toshino Asano,Yoshiyuki Sakawa. Modeling and Quasi-Static Hybrid Potision/Force control of constained Planar Two link flexible Manipulators , *IEEE transactions on robot and automation*, Vol 10.June 1994.

Muhammad Ali Mazidi and Jaince Gillispie Mazidi *THE 80x86 IBM PC &COMPATIBLE COMPUTERS* , Prentice-Hall International, Inc. 1995.

ธนัท ชัยยุทธ และ กณพ แก้วพิชัย" *ดิจิทอลพื้นฐาน* ", บริษัทซีเอ็ดยูเคชั่น. 2540

ปิยะ อำนวยพร " *Delphi 3* " , บริษัท เฟิสท์แปซิฟิก มีเดีย (ไทยแลนด์) จำกัด, 2541

วิบูลย์ ชื่นแขก "*ไมโครโปรเซสเซอร์*" สำนักพิมพ์สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ, 2532