อาคารจำลองที่จอดรถแบบอัตโนมัติ

AUTOMATIC CAR PARKING MODEL

นายประกาศิต มฤทุสาธร

นายสันติ ฤทธิสิงห์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

AUTOMATIC CAR PARKING MODEL

PRAKASIT MRUETUSATORN
SANTI RISTHISING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING
DEPARTMENT OF INSTRUMENTATION ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2002

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมการศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์ อาคารจำลองที่จอดรถแบบอัตโนมัติ

AUTOMATIC CAR PARKING MODEL

นักศึกษาผู้จัดทำ นายประกาศิต มฤทุสาธร รหัสประจำตัว 43015569

นายสันติ ฤทธิสิงห์ รหัสประจำตัว 43015593

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2545

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
อ. อัมพวัน ใจกล้า	

วัน/เดือน/ปี ที่สอบ วันอังคารที่ 25 มีนาคม พ.ศ 2546

สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(ผศ.ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	อาคารจำลองที่จอดรถแบบอัตโนมัติ
	AUTOMATIC CAR PARKING MODEL
นักศึกษาผู้จัดทำ	นายประกาศิต มฤทุสาธร
	นายสันติ ฤทธิสิงห์
อาจารย์ที่ปรึกษา	อ. อัมพวัน ใจกล้า
ปีการศึกษา	2545

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการนำเสนอแบบจำลองอาคารที่จอดรถ โดยการประยุกต์ใช้เครื่องคอมพิวเตอร์ส่วนบุคคล ในการควบคุมการเข้า – ออกของรถในแต่ละชั้น พร้อมทั้งตรวจสอบจำนวนรถที่จอดอยู่ โดยใช้ Visual Basic ในการเขียนโปรแกรม การควบคุมทำได้ 2 แบบ คือ แบบกำหนดชั้นในการจอดเอง และแบบอัตโนมัติ เมื่อมีรถวิ่งเข้า – ออก จากอาคารที่จอดรถ และวิ่งผ่านตัวตรวจจับแบบวัตถุตัดลำแสง ตัวตรวจจับจะส่งสัญญาณไปยังคอมพิวเตอร์ เพื่อประมวลผลตามโปรแกรม พร้อมทั้งควบคุมการเลื่อนขึ้น – ลงของลิฟท์ เพื่อนำรถเข้าออกจากชั้นที่กำหนด ซึ่งโครงงานนี้สามารถนำไปประยุกต์ใช้ได้กับงานจริงตามที่ต้องการที่ต้องการ

Thesis Title Automatic Car Parking Model

Authors Mr. Prakasit Mruetusatorn

Mr. Santi Risthising

Thesis Advisor Miss. Amphawan Chaikla

Year 2002

ABSTRACT

This project presents automatic car parking model using personal computer application. The computer control flows sequence for parking and control number the car in each floor by using the program written in Visual Basic . The car parking model can additionally control by manual mode. When the car comes in the sensors produce signal input and send to computer for control elevator in order take it to the parking. According to this project can be applied in real system.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความเมตตาจาก อาจารย์ อัมพวัน ใจกล้า ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่าง ๆ รวมถึงค่าใช้จ่ายต่าง ๆ ในการวิจัยในการทำปริญญานิพนธ์นี้ ผู้วิจัยรู้สึกทราบซึ้งและขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณอาจารย์ภาควิศวกรรมการวัดคุมทุกท่าน ที่ให้คำแนะนำและความช่วยเหลือทุก ๆ ด้านอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

และขอกราบขอบพระคุณคุณพ่อ คุณแม่ อันเป็นที่รักและเคารพยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้

คุณค่าและประโยชน์ที่พึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่1
บทนำ

1.1 แนวความคิดในการนำเสนอปริญญานิพนธ์

แนวความคิดในการนำเสนอปริญญานิพนธ์นี้มีจุดเริ่มต้นคือความต้องการของมนุษย์เพื่อต้องการใช้สิ่งอำนวยความสะดวกมีมากขึ้นทุกขณะ โดยสิ่งอำนวยความสะดวกที่มนุษย์ต้องการมากเป็นอันดับต้น ๆ ในการเดินทางนั้นก็คือ รถยนต์ แต่ในปัจจุบันพื้นที่ว่างที่จะใช้สำหรับจอดรถมีอยู่น้อยและประกอบกับราคาของที่ดินมีราคาสูงขึ้น และที่จอดรถโดยทั่วไปนั้นจะมีลักษณะเป็นอาคารที่จอดรถโดยการขับวนขึ้นหรือลงเพื่อไปยังที่จอดรถทำให้สิ้นเปลืองพื้นที่มากประกอบกับไม่มีระบบในการควบคุมที่จอดรถทำให้ผู้ที่จะนำรถยนต์ไปจอดไม่ทราบว่าชั้นไหนว่าง และบริเวณใดของอาคารว่างบ้าง จึงได้มีแนวคิดในการสร้างอาคารจอดรถที่มีการใช้สอยพื้นที่ให้เกิดประโยชน์สูงสุด โดยการใช้ลิฟท์ในการยกรถขึ้นและลงเพื่อนำรถยนต์ไปจอดยังชั้นต่างๆ ในการควบคุมที่จอดรถจะทำการตรวจสอบที่ว่างในการจอด เพื่อทำให้ผู้ที่จะนำรถไปจอดทราบว่าในแต่ละชั้นมีรถจอดอยู่ในชั้นนั้นๆ เท่าไรและต้องการจะนำรถไปจอดยังชั้นไหนได้บ้าง

อาคารจำลองที่จอดรถที่นำเสนอนี้จะทำให้ประหยัดพื้นที่ของอาคารที่จะใช้เป็นส่วนของการขับวนขึ้นหรือลงในระหว่างชั้นของอาคาร ประหยัดค่าใช้จ่ายและประหยัดทรัพยากรบุคคลรวมถึงทำให้เกิดความเชื่อมั่นในระบบรักษาความปลอดภัย โดยมีโครงสร้างเป็นอาคาร 3 ชั้น ในแต่ละชั้นสามารถจอดรถได้ 10 คัน และมีลิฟท์สำหรับขึ้น 1 ตัว ลิฟท์สำหรับลงอีก 1 ตัว โดยใช้สเต็ปมอเตอร์ (Stepping Motor) เป็นตัวควบคุมลิฟท์และใช้คอมพิวเตอร์ส่วนบุคคลเป็นตัวควบคุมโดยพัฒนาบนโปรแกรม Visual Basic การควบคุมอาคารจำลองทำได้โดยเมื่อรถผ่านเข้ามาจะมีตัวตรวจจับ ตรวจว่ามีรถเข้ามาและจะทำการสั่งลิฟท์ให้ลงมารอที่ชั้นล่าง และทำการหน่วงเวลาเพื่อรอให้รถเข้าไปในลิฟท์ให้เรียบร้อยและหลังจากนั้นลิฟท์จะเคลื่อนที่ขึ้นไปยังชั้นที่ยังมีพื้นที่ว่างเมื่อเคลื่อนไปชั้นที่ว่างแล้วจะทำการหน่วงเวลาเพื่อรอให้รถวิ่งออกจากลิฟท์เพื่อไปจอด เมื่อรถวิ่งผ่านตัวตรวจจับในแต่ละชั้นตัวตรวจจับนั้นก็จะเป็นเป็นตัวนับและส่งผลไปยังคอมพิวเตอร์ว่าได้มีรถเข้ามาจอดในชั้นนั้นแล้ว เมื่อรถต้องการจะลง ก็จะมีตัวตรวจจับในด้านลงแต่ละชั้นเพื่อสั่งการให้ลิฟท์ทางด้านลงเคลื่อนที่ขึ้นมารับรถลงไป โดยจะทำการตรวจนับและส่งผลไปยังคอมพิวเตอร์ เมื่อลิฟท์ขึ้นมารับรถก็จะหน่วงเวลาเพื่อรอให้รถเข้าไปจอดในลิฟท์ และลิฟท์ก็จะเคลื่อนที่ลงมายังชั้นล่างและหน่วงเวลาเพื่อรอให้รถออกไปจากลิฟท์และจะมีตัวตรวจจับว่ารถวิ่งออกจากลิฟท์ไปแล้วและเมื่อรถออกจากตัวอาคารก็จะเป็นการจบกระบวนการ

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาเกี่ยวกับการเชื่อมต่ออุปกรณ์ภายนอกโดยผ่านพอร์ตขนาน (Parallel Port)
2. สามารถนำความรู้ที่ได้จากการเชื่อมต่อผ่านพอร์ตขนานมาประยุกต์ใช้ในการควบคุมกระบวนการได้
3. ออกแบบและสร้างแบบจำลองของอาคารที่ใช้เป็นที่จอดรถ
4. ออกแบบและสร้างชุดขับสเต็ปมอเตอร์โดยใช้การเชื่อมต่อผ่านพอร์ตขนาน
5. ออกแบบและสร้างวงจรขยายพอร์ตสัญญาณ I^2C
6. ศึกษาโปรแกรมที่ใช้ในการควบคุมโดยใช้โปรแกรม Visual Basic
7. เขียนโปรแกรม Visual Basicในการควบคุมตัวตรวจจับและสเต็ปมอเตอร์เพื่อจัดการในการควบคุมการจอดรถอัติโนมัติ

1.3 ขอบเขตของปริญญานิพนธ์

1. ออกแบบและสร้างแบบจำลองของอาคารที่จะใช้เป็นที่จอดรถซึ่งจะมี 3 ชั้น และใช้ลิฟท์เพื่อทำการยกรถขึ้นเพื่อนำรถไปจอดในชั้นต่าง ๆ
2. ออกแบบและสร้างบอร์ด ที่ทำหน้าที่เป็นวงจรในการส่งสัญญาณโดยใช้การเชื่อมต่ออุปกรณ์ภายนอกผ่านพอร์ตขนาน
3. ออกแบบและสร้างบอร์ดที่ทำหน้าที่เป็นตัวขับสเต็ปมอเตอร์โดยใช้พอร์ตขนาน
4. ออกแบบและสร้างบอร์ดที่ทำหน้าที่ขยายพอร์ตโดย I^2C
5. ศึกษาโปรแกรม Visual Basic เพื่อใช้โปรแกรมนี้ในการควบคุม
6. เขียนโปรแกรม Visual Basic ในการควบคุมตัวตรวจจับและสเต็ปมอเตอร์เพื่อจัดการในการควบคุมการจอดรถอัติโนมัติ

1.4 รายละเอียดของปริญญานิพนธ์

ภายในปริญญานิพนธ์เล่มนี้แบ่งออกเป็น 7 บทด้วยกัน โดยมีรายละเอียดแต่ละบทดังนี้

บทที่ 1 กล่าวนำและวัตถุประสงค์ในการทำปริญญานิพนธ์ และขอบเขตของการศึกษาและรายละเอียดของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีเบื้องต้นเกี่ยวกับพอร์ตขนาน

บทที่ 3 การเขียนโปรแกรมติดต่อกับพอร์ตขนานพร้อมบอร์ดวงจรรวมถึงการขยายพอร์ตผ่าน I^2C

บทที่4 หลักการพื้นฐานของสเต็ปมอเตอร์

บทที่ 5 โปรแกรม Visual Basic 6.0

บทที่ 6 อาคารจำลองที่จอดรถแบบอัตโนมัติ

บทที่ 7 สรุปและวิจารณ์

บทที่ 2
ความรู้เบื้องต้นเกี่ยวกับพอร์ตขนาน

การประมวลผลข้อมูลเพื่อการควบคุมนั้น สิ่งแรกคือจะต้องมีส่วนของสัญญาณอินพุต ซึ่งอาจจะมาจากตัวตรวจจับต่างๆ ผ่านวงจรเพื่อเปลี่ยนรูปแบบสัญญาณอินพุตให้เหมาะสมกับการเชื่อมต่อคอมพิวเตอร์ เมื่อข้อมูลอินพุตถูกส่งเข้าคอมพิวเตอร์แล้ว คอมพิวเตอร์จะทำการประมวลผลข้อมูลที่ได้มาเหล่านั้นให้อยู่ในรูปแบบที่เหมาะสมก่อนที่จะส่งข้อมูลออกไปยังภายนอกผ่านอุปกรณ์เอาต์พุต ซึ่งจะเป็นการส่งออกไปยังจอภาพ หรือส่งออกไปยังจุดเชื่อมต่ออื่น ๆ เพื่อควบคุมอุปกรณ์เอาต์พุตต่อไป

การเชื่อมต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอกทั้งส่วนของภาคอินพุตและเอาต์พุตสามารถทำได้หลายวิธี ดังนี้

1. เชื่อมต่อผ่านทางการ์ดอินพุต/เอาต์พุต ซึ่งใช้การเสียบหรือติดตั้งการ์ดลงในสล็อตภายในเครื่องคอมพิวเตอร์
2. เชื่อมต่อผ่านทางพอร์ตอนุกรม
3. เชื่อมต่อผ่านทางพอร์ตขนาน
4. เชื่อมต่อผ่านทางระบบมาตรฐานอื่น ๆ เช่น พอร์ต USB (Universal Serial Bus) พอร์ต SCSI หรือพอร์ต GAME เป็นต้น

2.1 การเลือกใช้พอร์ตขนาน

เมื่อเทียบกับการใช้งานการ์ดอินพุต/เอาต์พุตที่ต้องติดตั้งภายในเครื่องคอมพิวเตอร์แล้ว พอร์ตขนานมีข้อดีอยู่หลายประการ ดังนี้

ในด้านความปลอดภัย การที่ต้องถอดฝาเครื่องคอมพิวเตอร์ออกมาเพื่อเสียบการ์ดเชื่อมต่อลงในสล็อตของคอมพิวเตอร์ อาจจะทำให้เกิดความเสียหายแก่ส่วนอื่นของคอมพิวเตอร์ได้ ถ้าไม่มีความชำนาญหรือเมื่อเกิดต่อวงจรผิดพลาด

ในด้านการเข้ากันได้กับคอมพิวเตอร์ส่วนใหญ่ การเชื่อมต่อโดยการใช้การ์ดเพื่อเสียบลงในสล็อต ไม่สามารถใช้กับคอมพิวเตอร์ในปัจจุบันได้ทุกรุ่น ยกตัวอย่างคอมพิวเตอร์โน๊ตบุ๊กจะไม่มีสล็อตเสียบการ์ดแต่จะมีที่เสียบการ์ด PCMCIA แทนในขณะที่พอร์ตขนานจะติดตั้งอยู่ในคอมพิวเตอร์ทุกเครื่องทั้งนี้เพื่อใช้ติดต่อกับเครื่องพิมพ์

ข้อจำกัดด้านพื้นที่ คอมพิวเตอร์บางเครื่องมีการเสียบการ์ดเชื่อมต่อตัวอื่นๆอยู่แล้ว อาทิ การ์ดเสียง การ์ดโมเด็ม เป็นต้น จะไม่มีสล็อตสำหรับเสียบการ์ดเพิ่มเติม

ความสะดวกในการใช้งาน การเชื่อมต่อทางพอร์ตขนานสามารถทำได้ง่าย เพียงต่อสายสำหรับใช้งานเข้ากับคอนเน็กเตอร์ DB-25 ของพอร์ตขนาน

จำนวนช่องสัญญาณอินพุต/เอาต์พุต พอร์ตขนานมีจำนวนพอร์ตอินพุต/เอาต์พุตมากเพียงพอที่จะนำไปใช้งานต่าง ๆ และยังสามารถขยายให้มีพอร์ตเพิ่มขึ้นได้ โดยพอร์ตขนานปกติมีจำนวนขาเอาต์พุต 12 ขา และขาอินพุต 5 ขา

ความเร็วในการสื่อสารข้อมูลกับพอร์ตขนาน มีความเร็วเท่ากับการติดต่อระบบบัสโดยตรง และมีความเร็วมากกว่าการติดต่อผ่านพอร์ตอนุกรม

อะไหล่และชิ้นส่วนประกอบ คอนเน็กเตอร์และสายเชื่อมต่อต่าง ๆ ของการเชื่อมต่อผ่านพอร์ตขนานหาได้ง่ายและราคาไม่แพง หรือจะสร้างขึ้นมาเองก็สามารถทำได้

จากคุณสมบัติดังที่กล่าวมานั้นทำให้พอร์ตขนานเหมาะสมอย่างยิ่งที่จะนำไปใช้ในการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกเพื่อควบคุมหรือรับสัญญาณข้อมูล นอกจากนั้นหากนำคุณสมบัติของการเขียนโปรแกรม ผ่านระบบปฏิบัติการวินโดวส์ด้วยโปรแกรม Visual Basic ก็สามารถสร้างระบบการเชื่อมต่อที่สมบูรณ์และใช้งานได้ไม่ยาก

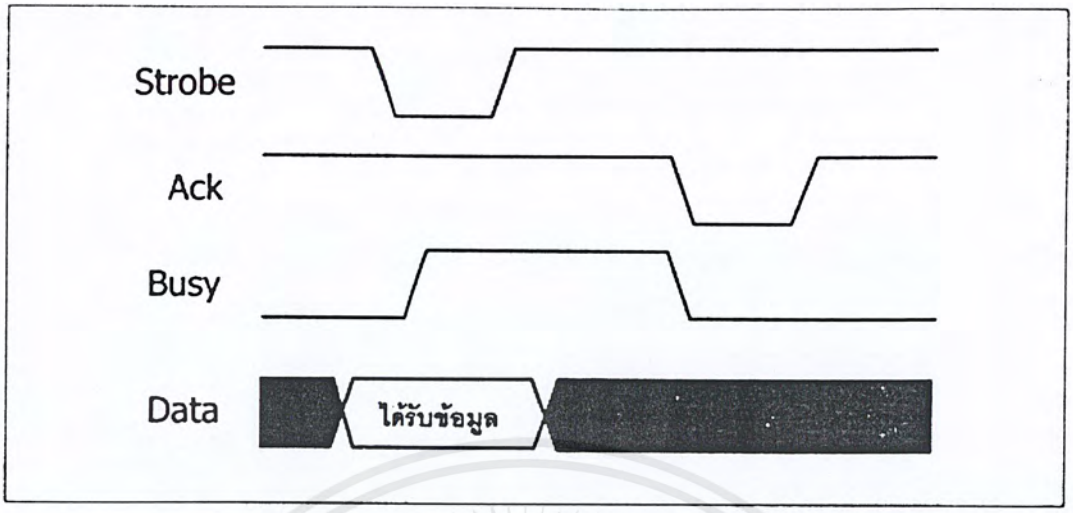
2.2 ความรู้เบื้องต้นเกี่ยวกับพอร์ตขนาน

พอร์ตขนาน สาเหตุที่เป็นชื่อนี้ เนื่องจากการถ่ายถอดข้อมูลของพอร์ตนี้เป็นแบบขนาน สำหรับชื่อเรียกอีกอย่างคือ พอร์ตเครื่องพิมพ์ (Printer port) เนื่องจากพอร์ตนี้ใช้เชื่อมต่อเครื่องพิมพ์นั้นเอง

ด้วยการถ่ายถอดข้อมูลแบบขนานนี้เอง ทำให้พอร์ตขนานมีอัตราถ่ายถอดข้อมูลสูงกว่าการถ่ายถอดข้อมูลแบบอนุกรมถึง 8 – 10 เท่า และการประมวลข้อมูลส่วนใหญ่จะเป็นแบบ 8 บิต ดังนั้นพอร์ตขนานจึงสมารถรองรับการถ่ายถอดข้อมูล 8 บิต ได้โดยไม่ต้องต่อส่วนอื่นเพิ่มเติม

2.3 ลักษณะทางกายภาพของพอร์ตขนาน

เพื่อให้เข้าใจถึงการนำพอร์ตขนานไปใช้งาน ก่อนอื่นต้องทำความเข้าใจก่อนว่าปกตินั้นการสั่งพิมพ์งานจากคอมพิวเตอร์ไปพอร์ตขนานนั้นมีรูปแบบการทำงานอย่างไร ดังรูปที่ 2-1 แสดงไดอะแกรมเวลาของการติดต่อระหว่างพอร์ตขนานกับเครื่องพิมพ์ ซึ่งจะเห็นได้ว่ามีสัญญาณที่ใช้งานจริง ๆ มีไม่มาก เริ่มจากสัญญาณพอร์ต Data ถูกส่งออกไปยังเครื่องพิมพ์ พร้อมทั้งส่งสัญญาณ Strobe ออกไปด้วย เพื่อให้เครื่องพิมพ์รู้ว่ามีการส่งข้อมูลใหม่มาที่ขา Data แล้ว จากนั้นคอมพิวเตอร์จะต้องรอการตอบกลับจากเครื่องพิมพ์ นั่นคือเครื่องพิมพ์จะสร้างสัญญาณ Busy หรือเพื่อบอกว่าเครื่องพิมพ์ยังไม่พร้อมที่จะรับข้อมูลใหม่ จนกระทั้งเครื่องพิมพ์พร้อม เครื่องพิมพ์จะสร้างสัญญาณ ACK ส่งไปยังคอมพิวเตอร์เพื่อแจ้งว่า พร้อมที่จะรับข้อมูลใหม่แล้ว



รูปที่ 2-1 ไดอะแกรมเวลาของการส่งข้อมูลไปยังเครื่องพิมพ์

สัญญาณข้อมูลชนิด 8 บิต สัญญาณ Strobe และสัญญาณ ACK (acknowledge) เป็นสัญญาณที่สำคัญในการส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ นอกจากสัญญาณทั้งสามแล้ว ส่วนใหญ่การติดต่อกับเครื่องพิมพ์ต้องมีสัญญาณอื่น ๆ ร่วมด้วย เนื่องจากเครื่องพิมพ์ต้องทำหน้าที่ถึง 3 อย่างด้วยกัน คือ รับข้อมูลจากคอมพิวเตอร์พิมพ์ข้อมูลที่รับเข้ามาและตอบสนองต่อการใช้งานของผู้ใช้ บางครั้งอาจเกิดเหตุการณ์ที่ไม่ปกติ เช่น บัฟเฟอร์สำหรับรับข้อมูลเต็ม เครื่องพิมพ์จะต้องแจ้งไปยังคอมพิวเตอร์ว่าให้หยุดส่งข้อมูลชั่วคราว เนื่องจากไม่สามารถรับข้อมูลมากว่านี้ได้แล้ว สัญญาณที่ส่งจากเครื่องพิมพ์ไปยังคอมพิวเตอร์คือ สัญญาณ Busy และเมื่อเครื่องพิมพ์เกิดข้อผิดพลาด เช่น กระดาษติด เครื่องพิมพ์จะต้องแจ้งไปยังเครื่องคอมพิวเตอร์เช่นกัน โดยสัญญาณที่แจ้งไปยังเครื่องคอมพิวเตอร์เรียกว่าสัญญาณ Error นอกจากนี้เมื่อคอมพิวเตอร์ต้องการ รีเซตเครื่องพิมพ์ คอมพิวเตอร์ต้องส่งสัญญาณ Reset ไปยังเครื่องพิมพ์เพื่อรีเซตเครื่องพิมพ์ด้วย สามารถสรุปขาสัญญาณที่จำเป็นสำหรับการติดต่อดังในตารางที่ 2-1

จากตารางที่ 2-1 จะเห็นได้ว่าพอร์ตขนานของคอมพิวเตอร์ยังแยกย่อยออกได้อีก 3 พอร์ต ได้แก่ พอร์ตเอาต์พุตที่ทำหน้าที่ส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ พอร์ตเอาต์พุตสำหรับสัญญาณ Srobe และ Reset พอร์ตอินพุตสำหรับการอ่านค่าสัญญาณ ACK สัญญาณ Busy และสัญญาณ Error จากเครื่องพิมพ์

ตารางที่ 2-1 สัญญาณสำคัญ ๆ ของพอร์ตขนานที่ใช้กับเครื่องพิมพ์

สัญญาณ	หน้าที่การทำงาน	ทิศทาง
ข้อมูล 8 บิต	ข้อมูลที่ส่งจากคอมพิวเตอร์ไปยังเครื่องพิมพ์	คอมพิวเตอร์
Strobe	แจ้งเครื่องพิมพ์ถึงข้อมูลที่ส่งมาใหม่	คอมพิวเตอร์
Acknowledge	เครื่องพิมพ์แจ้งมายังคอมพิวเตอร์ว่าได้รับข้อมูลแล้ว	คอมพิวเตอร์
Busy	แจ้งสถานะว่าเครื่องไม่ว่างที่จะรับข้อมูลใหม่	คอมพิวเตอร์
Error	แจ้งสถานะว่าเครื่องพิมพ์เกิดข้อผิดพลาด	คอมพิวเตอร์
Reset	รีเซตเครื่องพิมพ์	คอมพิวเตอร์

โดยปกติพอร์ตขนานออกแบบมาให้มีสายสัญญาณทั้งหมด 17 เส้น สายสัญญาณเหล่านั้นจะมีรีจิสเตอร์ 3 ตัวควบคุมการทำงานดังนี้

1. พอร์ตเอาต์พุตสำหรับสัญญาณข้อมูล 8 เส้น มีรีจิสเตอร์ Data ควบคุม
2. พอร์ตอินพุตสำหรับการอ่านสถานะต่าง ๆ จากภายนอก มีอยู่ด้วยกัน 5 เส้น โดยใช้รีจิสเตอร์ Status ในการควบคุม
3. พอร์ตเอาต์พุตสำหรับส่งสัญญาณควบคุมไปยังอุปกรณ์ภายนอก มีอยู่ด้วยกัน 4 เส้น ใช้รีจิสเตอร์ Control ในการควบคุม

บล็อกไดอะแกรมในรูปที่ 2-2 แสดงระบบบัสของคอมพิวเตอร์สำหรับการติดต่อกับพอร์ตขนาน สัญญาณเอาต์พุตจากพอร์ตขนานจะถูกส่งไปยังคอนเน็กเตอร์แบบ DB-25 สำหรับคอมพิวเตอร์ส่วนใหญ่ในปัจจุบัน พอร์ตขนานจะมาพร้อมกับเมนบอร์ด ไม่จำเป็นต้องใช้การ์ดเสียบเพิ่มเติมเหมือนในอดีต พร้อมทั้งมีฟังก์ชั่นที่ทำงานซับซ้อนมากขึ้น แต่ยังคงสนับสนุนการทำงานของพอร์ตขนานในรูปแบบมาตรฐานอยู่

เมื่อดูจากรูปที่ 2-1 เทียบการทำงานโดยทั่วไปกับการเชื่อมต่อผ่านการ์ดที่เสียบลงในสล็อตของคอมพิวเตอร์แล้ว พอร์ตขนานจะมีลักษณะใกล้เคียงกัน โดยการติดต่อกับพอร์ตขนานจะการอ้างแอดเดรส ตำแหน่งแอดเดรสที่ใช้อ้างถึงจะเป็นตำแหน่ง A0-A9 และใช้ขา IOR และ IOW สำหรับเป็นตัวเลือกว่า ต้องการอ่านหรือเขียนรีจิสเตอร์ใด จากการดีโค๊ดแอดเดรส A0-A9 นี้เองทำให้ได้สัญญาณออกมาเพื่อไปควบคุมหรือเอ็นเอเบิลวงจรบัฟเฟอร์ต่าง ๆ ดังนี้

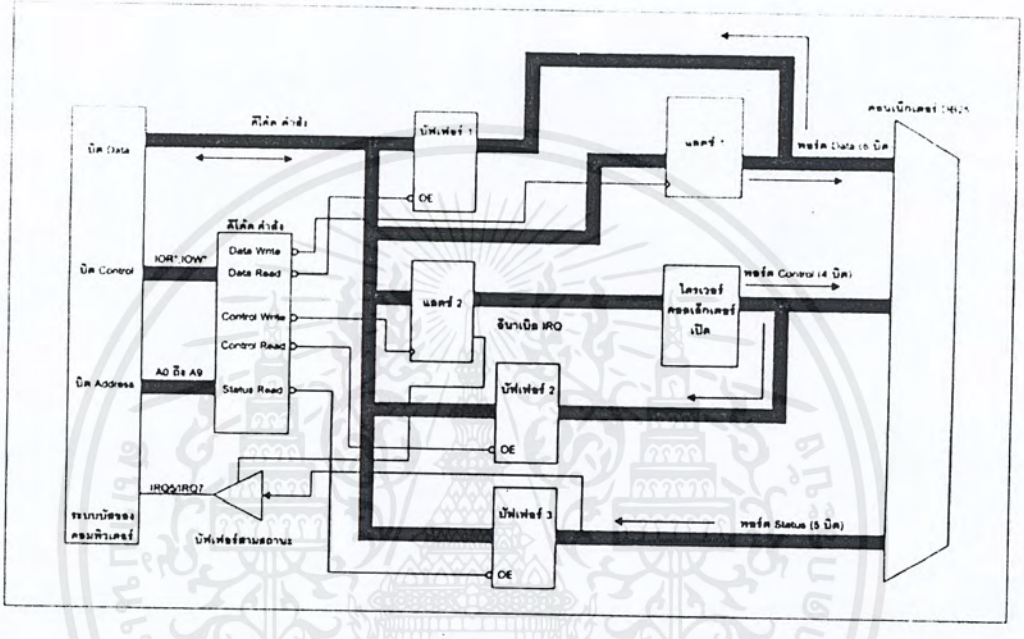
Data Write สัญญาณเอ็นเอเบิลสำหรับข้อมูลที่อยู่ในบัส Data ไปออกที่ขา Data ของพอร์ตขนาน

Data Read สัญญาณเอ็นเอเบิลสำหรับอ่านข้อมูลจากขา Data ของพอร์ตขนานมาเก็บไว้ในบัส Data

Control Write สัญญาณเอ็นเอเบิลสำหรับนำข้อมูลที่อยู่ในบัส Data ไปออกที่ขา Control ของพอร์ตขนานสำหรับพอร์ตนี้นอกจากจะส่งข้อมูลไปออกที่พอร์ตขนานแล้วยังทำหน้าที่อีนาเบิลอินเตอร์รัปต์ของการเปลี่ยนแปลงสัญญาณที่พอร์ต Status

Control Read สัญญาณเอ็นเอเบิลสำหรับอ่านข้อมูลจากขา Control มาเก็บไว้ในบัส Data

Status Read สัญญาณเอ็นเอเบิลสำหรับอ่านค่ามูลจากขาพอร์ต Status มาเก็บไว้ในบัส Data



รูปที่ 2- 2 ระบบบัสภายในของพอร์ตขนาน

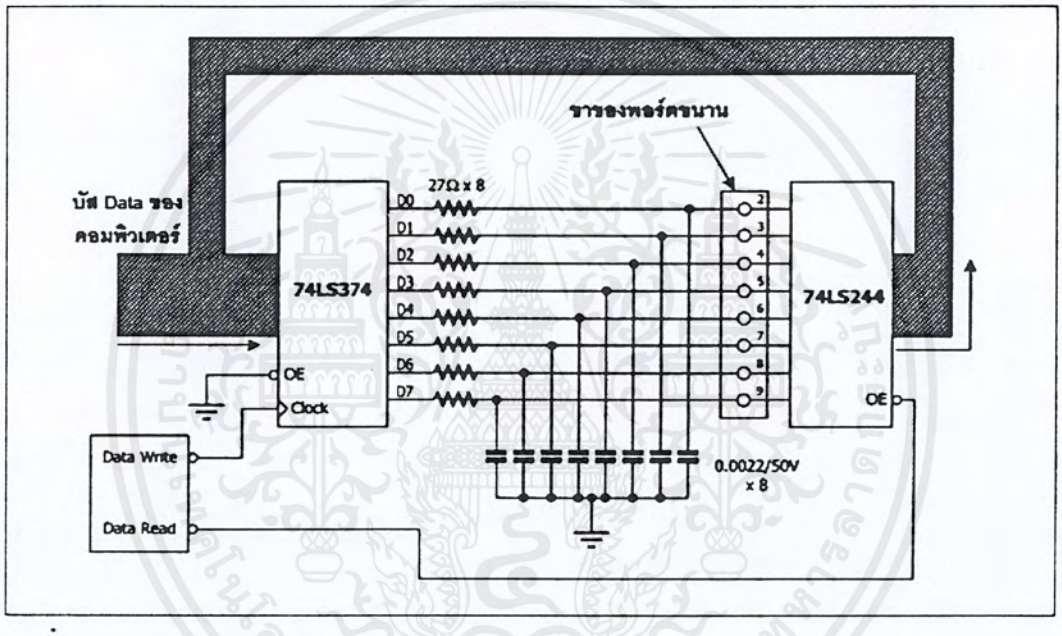
2.4 พอร์ตดาต้า (Data Port)

จากรูปที่ 2-3 แสดงให้เห็นว่าพอร์ต Data ประกอบด้วยบัฟเฟอร์ 1 ตัว และไอซีแลตช์อีก 1 ตัว เมื่อคอมพิวเตอร์ต้องการส่งข้อมูลไปยังเครื่องพิมพ์ คอมพิวเตอร์จะเขียนข้อมูลไปยังไอซีแลตช์ 1 ทั้ง 8 บิต เอาต์พุตของไอซีแลตช์ 1 คือ D0-D7 ซึ่งเอาต์พุตนี้จะไปปรากฏที่พอร์ตขนานในตำแหน่งขา 2 ถึงขา 9 และที่ขาเอาต์พุตนี้สัญญาณ Data จะส่งกลับไปเป็นอินพุตของบัฟเฟอร์ 1 ด้วย ทำให้คอมพิวเตอร์สามารถอ่านค่าสถานะปัจจุบันที่เกิดขึ้นกับพอร์ต Data ได้

เมื่อคอมพิวเตอร์ส่งข้อมูล ข้อมูลจะถูกส่งมาจากบัสข้อมูลของคอมพิวเตอร์ผ่านไปให้กับไอซี 74LS374 ซึ่งเป็นไอซีแลตช์ข้อมูล และเมื่อต้องการให้ข้อมูลปรากฏที่เอาต์พุต คอมพิวเตอร์จะส่งสัญญาณ Data Write ออกไปที่ขา CLK ของ 74LS374 เอาต์พุตจาก 74LS374 จะถูกกรองด้วยวงจร RC ซึ่งประกอบด้วยตัวต้านทานค่า 27 Ω และตัวเก็บประจุ 0.0022 μF เพื่อให้ช่วงเวลาที่เปลี่ยนแปลงจากลอจิก “0” เป็นลอจิก “1” หรือจากลอจิก “1” เป็นลอจิก “0” เป็นไปอย่างช้า ๆ

เนื่องจากการเปลี่ยนแปลงแรงดันที่รวดเร็วทำให้เกิดสัญญาณรบกวนเหนี่ยวนำข้ามไปยังข้อมูลบิตอื่น ๆ ได้ ทำให้ข้อมูลที่ส่งออกไปมีข้อมูลผิดพลาด จากค่าตัวต้านทานและตัวเก็บประจุในวงจรทำให้เกิดการหน่วงเวลา ไปประมาณ 60 นาโนวินาที จากวงจรในรูปที่ 2-3 ทำให้เอาต์พุตของพอร์ต Data มีคุณสมบัติดังนี้

- กระแสซิงก์สูงสุด 24 mA
- กระแส Soure สูงสุด 2. 6 mA
- ระดับแรงดันของลอจิก 1 ต่ำสุดเท่ากับ 2.4 v
- ระดับแรงดันสูงสุดลอจิก 0 เท่ากับ 0.5 v



รูปที่ 2-3 วงจรภายในของพอร์ต Data

สำหรับบัฟเฟอร์สำหรับการอ่านข้อมูลกลับ ได้แก่เบอร์ 74LS244 ซึ่งเมื่อต้องการอ่านค่าคอมพิวเตอร์จะส่งสัญญาณ Data Read ออกกมาเพื่อเอ็นเอเบิลไอซี 74LS244 สำหรับพอร์ตขนานแบบมาตรฐาน (Standard Parallel Port :SPP) พอร์ต Data จะต้องใช้เพื่อการส่งค่าออกเอาต์พุตเท่านั้น แต่สำหรับพอร์ตขนานที่มีการสื่อสารสองทิศทาง สามารถอ่านค่าจากพอร์ต Data ได้ด้วย แต่ก่อนที่จะอ่านค่าต้องจำไว้เสมอว่าจะต้องป้อนค่าเอาต์พุตให้มีค่าลอจิก 1 ทั้งหมดก่อน

ตารางที่ 2-2 สัญญาณทั้งหมดที่อยู่บนพอร์ตขนาน

ขาของพอร์ตขนาน	รีจิสเตอร์	ทิศทาง	ตำแหน่งบิต	ชื่อขาสัญญาณ	หน้าที่การทำงาน
1	Control	Out	CO	STROBE	แอกตีฟ 0 ส่งค่าออกไปเพื่อบอกว่าที่ขาดาต้ามีข้อมูลแล้ว
2-9	Data	Out	DO-D7	DATAO-DATA7	สำหรับพอร์ตขนานมาตราฐานเดิมขานี้ทำหน้าที่เป็นขาส่งข้อมูลเอาต์พุตเท่านั้นสำหรับปัจจุบันขานี้รับข้อมูลอินพุตได้ด้วย
10	Status	In	S6	nACK	เป็นพัลส์ลอจิก “0” ที่ส่งมาจากเครื่องพิมพ์เพื่อบอกว่าได้รับข้อมูลที่ส่งไปแล้ว
11	tatus	In	S7	BUSY	เป็นสัญญาณแจ้งมาจากเครื่องพิมพ์ว่ายังไม่พร้อมรับข้อมูล
12	Status	In	S5	PE	แจ้งกระดาษหมด
13	Status	In	S4	SELECT	แจ้งว่าเครื่องต่ออยู่
14	Control	Out	C1	AUTO FEED	สั่งเครื่องพิมพ์ให้เลื่อนบรรทัด
15	Status	In	S3	ERROR	สัญญาณจากเครื่องมายังคอมพิวเตอร์เพื่อแสดงข้อผิดพลาดจากการพิมพ์
16	Control	Out	C2	INIT	รีเซตเครื่องโดยให้ลอจิก “0”
17	Control	Out	C3	SELECT-IN	ส่งสัญญาณไปยังเครื่องพิมพ์เพื่อแจ้งว่าต้องการเครื่องพิมพ์เครื่องนี้
18-25				GND	กราวน์ด

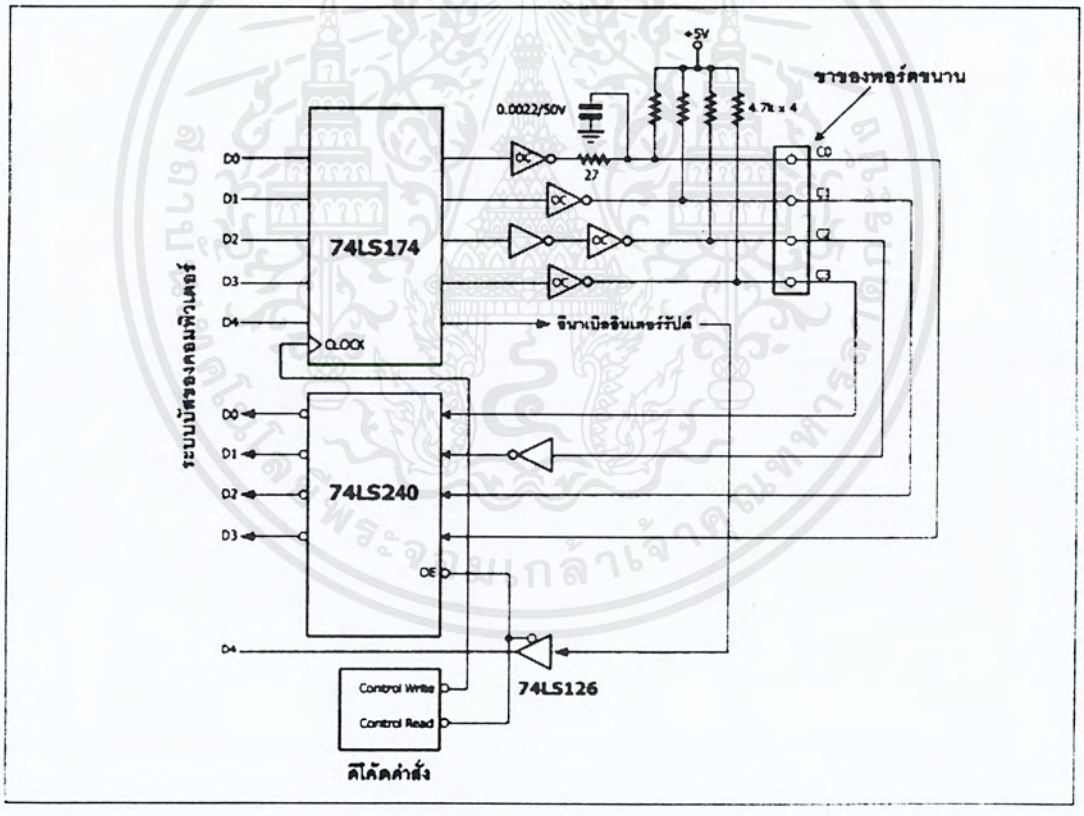
2.5 พอร์ตควบคุม (Control Port)

พอร์ต Control ใช้สำหรับคอมพิวเตอร์ควบคุมเครื่องพิมพ์ จากตารางที่ 2-2 จะเห็นว่าพอร์ต Control ประกอบไปด้วยบิตเอาต์พุต 4 บิต ที่ต่อไปยังเครื่องพิมพ์ ส่วนบิตเอ็นเอเบิลอินเตอร์รัปต์

ไม่ได้ถูกต่อออกไป รูปที่ 2-4 แสดงวงจรภายในของพอร์ต Control จะเห็นว่าเอาต์พุตของพอร์ต Control มีอินเวอร์เตอร์แบบคอลเล็กเตอร์ต่อร่วมอยู่ โดยเอาต์พุตเหล่านี้จะถูกพูลอัปไว้ด้วยตัวต้านทานค่า 4.7 kΩ สำหรับบิต C2 จะผ่านอินเวอร์เตอร์ถึงสองตัวทำให้ที่เอาต์พุตของบิต C2 ไม่มีการกลับสถานะลอจิก

สถานะของพอร์ต Control สามารถอ่านกลับได้และด้วยการใช้บัฟเฟอร์เบอร์ 74 LS240 ซึ่งที่เอาต์พุตมีอินเวอร์เตอร์ต่ออยู่ภายใน ทำให้ค่าที่อ่านได้ตรงกับค่าที่ส่งออกไป การควบคุมการอ่านและเขียนข้อมูลกับพอร์ต Control คอมพิวเตอร์จะส่งข้อมูลมาที่ขา Control Write และ Control Read

เนื่องจากเอาต์พุตของพอร์ต Control เป็นแบบคอลเล็กเตอร์เปิด ดังนั้นผู้ใช้งานสามารถใช้พอร์ตนี้ ในการอ่านค่าสัญญาณอินพุตจากภายนอกได้ โดยก่อนที่จะอ่านค่าจะต้องทำให้ขาพอร์ตที่ต้องการอ่านค่ามีลอจิก 1 เสียก่อน

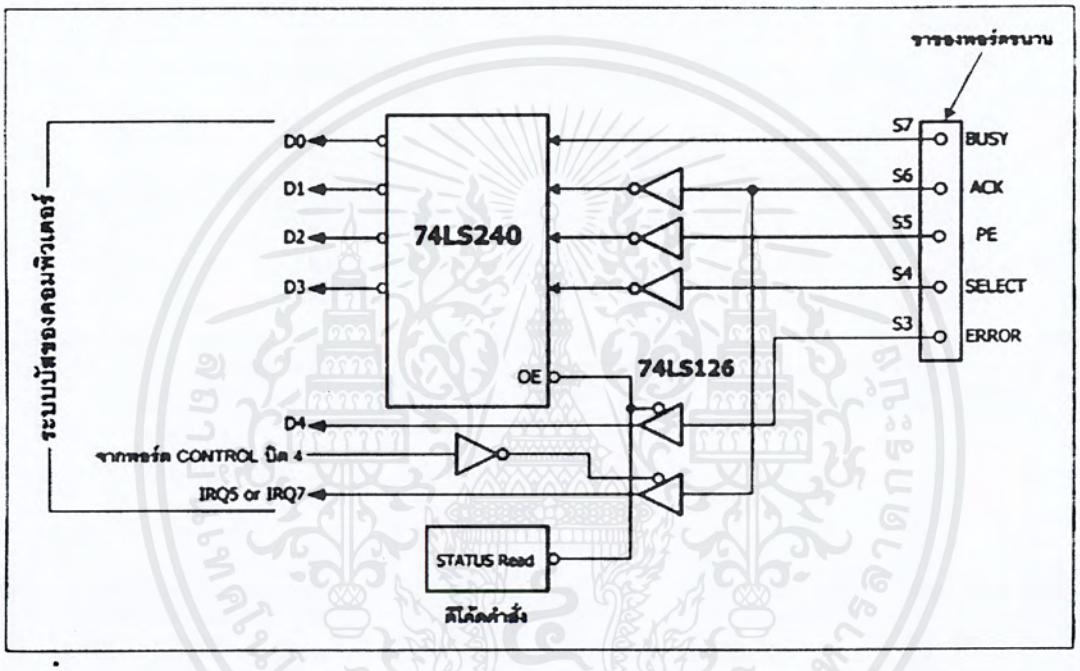


รูปที่ 2-4 วงจรภายในของพอร์ต Control

2.6 พอร์ตแสดงสถานะ (Status Port)

พอร์ต Status เป็นพอร์ตที่คอมพิวเตอร์ใช้สำหรับการอ่านค่าสถานะจากเครื่องพิมพ์ รูปที่

2-5 แสดงรายละเอียดภายในพอร์ต Status จะสังเกตได้ว่ามีขาสัญญาณทั้งหมด 5 สัญญาณด้วยกัน และจะเรียกชื่อเป็น S3, S4, S5, S6 และ S7 ซึ่งตัวเลขหมายถึงตำแหน่งบิตของขาเหล่านี้ ภายในรีจิสเตอร์ Status นั้นเอง สำหรับบิต S7 จะมีข้อแตกต่างจากบิตอื่น ๆ ที่เมื่อสัญญาณจากภายนอกส่งเข้ามาแล้วจะไม่ผ่านอินเวอร์เตอร์ ในขณะที่ขาอื่น ๆ ผ่านอินเวอร์เตอร์ทั้งหมด ดังนั้นเมื่อข้อมูลผ่านจากขาอินพุตไปยัง 74LS240 ซึ่งเอาต์พุตมีการกลับสถานะทำให้บิต S7 เป็นบิตเดียวที่มีการกลับสถานะ นอกจากนี้ในการใช้งานถ้าต้องการให้มีการสร้างสัญญาณอินเวอร์เตอร์รัปต์จากขาขอบขึ้นของขา S6 สามารถกำหนดค่าได้จากพอร์ต Control บิต 4



รูปที่ 2-5 แสดงวงจรภายในของพอร์ต Status

2.7 การนำพอร์ตขนานไปใช้งาน

สำหรับพอร์ตขนานแบบมาตรฐาน ผู้ใช้งานสามารถนำพอร์ตอินพุต 5 บิต (พอร์ต Status) พอร์ตเอาต์พุต 4 บิต (พอร์ต Control) และพอร์ตเอาต์พุตอีก 8 บิต (พอร์ต Data) ไปใช้งานได้โดยตรง โดยที่ 4 บิตของพอร์ตเอาต์พุตหรือพอร์ต Control นั้นสามรถดัดแปลงให้ใช้งานเป็นพอร์ตอินพุตขนาน 4 บิตได้ด้วย ดังนั้นผู้ใช้งานสามารถนำสัญญาณจากพอร์ตขนานที่มีมากถึง 17 เส้นไปใช้งานในการควบคุมได้โดยใช้ระดับสัญญาณ TTL

บทที่ 3
การเขียนโปรแกรมติดต่อกับพอร์ตขนาน

พอร์ตขนานของคอมพิวเตอร์จะมีลักษณะเช่นเดียวกับอุปกรณ์อินพุต/เอาต์พุตตัวอื่น ๆ คือ เมื่อต้องการติดต่อจะต้องกำหนดแอดเดรสที่ต้องการติดต่อด้วย ตารางที่ 3-1 แสดงแอดเดรสของพอร์ตรีจิสเตอร์ Control โดยแอดเดรสนี้จะมีทั้งหมด 3 ชุด สำหรับพอร์ตขนาน 3 ชุดคือ LPT1, LPT2 และ LPT3

ตารางที่ 3-1 แอดเดรสของพอร์ตขนาน

ชื่อพอร์ต	LPT1		LPT2		LPT3	
	ฐานสิบ	ฐานสิบหก	ฐานสิบ	ฐานสิบหก	ฐานสิบ	ฐานสิบหก
Data	888	378H	956	3BCH	632	278H
Status	889	379H	957	3BDH	633	279H
Control	890	37AH	958	3BEH	634	27AH

เมื่อต้องการติดต่อกับพอร์ตขนานในตำแหน่งใดก็ให้ส่งค่าข้อมูลออกไปที่พอร์ตขนานในตำแหน่งนั้น ๆ ยกตัวอย่างการเขียนโปรมแกรมด้วย QBASIC เพื่อส่งค่าลอจิก 1 ออกไปทุกบิตของพอร์ต Data ของ LPT1 จะต้องเขียนโปรแกรมดังนี้

```
Out    &H378,HFF
```

โดยที่เครื่องหมาย &H ที่แสดงนั้นหมายถึงเลขฐานสิบหก

คำสั่ง Out เป็นการส่งค่าข้อมูลจากเอาต์พุตของพอร์ตอินพุตเอาต์พุต

ค่า 378 เป็นแอดเดรสของรีจิสเตอร์ Data สำหรับ LPT1

ค่าข้อมูล FF เป็นข้อมูลเลขฐานสิบหก ซึ่งหมายถึงการใช้บิตทุกบิตของรีจิสเตอร์ Data จะมีลอจิกเป็น "1" นั่นเอง

ส่วนการอ่านค่าจากพอร์ตขนานมายังคอมพิวเตอร์ผ่านทางพอร์ต Status ของ LPT1 สามารถเขียนโปรแกรมด้วย QBASIC ได้ดังนี้

```
Temp = INP(&H379)
```

โดยที่ คำสั่ง INP(_) เป็นคำสั่งสำหรับอ่านค่าข้อมูล

ค่า 379ในตำแหน่งแอดเดรสของรีจิสเตอร์ Status สำหรับ LPT1 ในตัวเลขฐานสิบหก

ตัวแปร Temp เป็นตัวแปรที่ใช้เก็บข้อมูลที่อ่านได้จากพอร์ตขนาน

สำหรับโปรแกรมอื่น ๆ เช่น แอสแซมบลี้ เทอร์โบปาสคาล หรือเทอร์โบซี จะมีรูปแบบการเขียนโปรแกรมที่แตกต่างกันบ้าง เช่น

แอสแซมบลี้

การส่งค่าข้อมูลออกไปยังพอร์ต

```
mov dx, 378h
mov al, ffh
out dx, al
```

การอ่านข้อมูลจากพอร์ตขนาน

```
mov dx,379h
in al,dx
```

เทอร์โบบาสคาล

การส่งค่าข้อมูลออกไปยังพอร์ตขนาน

```
Port[378H] := FFH
```

การอ่านค่าข้อมูลจากพอร์ตขนาน

```
Temp := Port[379H]
```

เทอร์โบซี

การส่งค่าข้อมูลออกไปยังพอร์ตขนาน

```
Outportb (0x378,0xff)
```

การอ่านค่าข้อมูลจากพอร์ตขนาน

```
Temp = inportb (0x379)
```

อย่างไรก็ตามโปรแกรมทุกตัวต่างก็ใช้วิธีการเดียวกันคือ กำหนดแอดเดรสที่จะทำการติดต่อจากนั้นจึงติดต่อกับแอดเดรสเหล่านั้นด้วยคำสั่งสำหรับการอ่านหรือเขียน

3.1 การเขียนโปรแกรมติดต่อกับพอร์ตขนานด้วย Visual Basic

การเขียนโปรแกรมด้วย Visual Basic ชุดคำส่งส่วนใหญ่จะมีรูปแบบใกล้เคียงกับ QBASIC แต่ Visual Basic จะไม่มีคำสั่งสำหรับการติดต่อกับพอร์ตโดยตรงคือ คำสั่ง Inp () และคำสั่ง Out เหมือนกับ QBASIC ดังนั้นเพื่อให้สามารถติดต่อกับพอร์ตขนานได้จึงจำเป็นต้องเพิ่มโปรแกรมบางตัวเข้าไป โดยโปรแกรมที่เพิ่มเข้าไปนี้จะอยู่ในรูปของ DLL (Dynamic Linked Library)

ไฟล์ dll ที่ใช้งานในที่นี้คือ io.dll โดยสามารถใช้ได้กับระบบปฏิบัติการที่เป็น 32 บิตซึ่งก็คือวินโดวส์ 95 ขึ้นไป นั่นคือวินโดวส์ 95/98/ME/2000/NT/XP

สำหรับตำแหน่งที่ใช้เก็บไฟล์ io.dll นั้นจะต้องเก็บไว้ที่ไดเรกทอรี่ SYSTEM ซึ่งอยู่ภายในไดเรกทอรี่เก็บโปรแกรมวินโดวส์ โดยส่วนใหญ่จะมีชื่อเป็น Windows

สาเหตุที่โปรแกรม Visual Basic จึงไม่รวมคำสั่ง Inp และคำสั่ง Out ไว้ในโปรแกรม Visual Basic เนื่องจากว่าการเขียนและอ่านข้อมูลไปยังพอร์ตหรือหน่วยความจำโดยตรงนั้นอาจทำให้เกิดมีปัญหาแฮงก์หรือทำงานผิดพลาดได้และ Visual Basic เป็นระบบปฏิบัติการที่ทำงานบนวินโดว์ซึ่งมีการทำงานแบบมัลติทาร์สกิ้ง (multitasking) มีโปรแกรมหลาย ๆ ตัวทำงานอยู่พร้อมกัน ดังนั้นเมื่อเกิดความเสียหายแก่โปรแกรมตัวหนึ่งก็อาจส่งผลให้โปรแกรมที่ทำงานอยู่ทั้งหมดเกิดความเสียหายได้ นอกจากนี้การเขียนข้อมูลโดยตรงไปยังพอร์ต อาจจะไปทับซ้อนกับโปรแกรมอื่น ๆ ที่มีการเขียนข้อมูลไปยังพอร์ตเช่นเดียวกัน ส่งผลให้โปรแกรมทำงานผิดพลาด

สำหรับระบบปฏิบัติการวินโดว์ 95 ขึ้นไป นอกจากจะสามารถใช้งาน DLL ในการติดต่อกับพอร์ตโดยตรงแล้ว ยังสามารถใช้งานโปรแกรมประเภท Visual Device Drive (Vxd) ในการติดต่อกับอุปกรณ์อินพุต/เอาต์พุต โดย Vxd จะตัดปัญหาเรื่องการเข้าถึงพอร์ตพร้อมกันของโปรแกรมหลาย ๆ ตัวได้ แต่สำหรับโปรแกรมสั้น ๆ เช่น โปรแกรมอ่านค่าอุณหภูมิ โปรแกรมควบคุมอุปกรณ์อินพุต/เอาต์พุต ปกติไม่มีการติดต่อกับพอร์ตอยู่ตลอดเวลา คำสั่ง Inp และ OUT ใน dll ก็ยังทำงานได้ดีและมีรูปแบบการทำงานที่ง่ายกว่า

3.2 รายละเอียดเกี่ยวกับ io .dll

io.dll พัฒนาขึ้นโดยโปรแกรมเมอร์นิรนามที่ใช้ชื่อว่า Fred ซึ่งมีความชำนาญด้านการพัฒนาระบบเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก โดยไฟล์ io.dll มีผู้เขียนโปรแกรมเผยแพร่โดยไม่คิดมูลค่า ซึ่งผู้สนใจสามารถดาวน์โหลดเวอร์ชั่นใหม่ที่อาจมีได้ที่ http://www.geekhideout.com

io.dll สามารถใช้งานได้กับระบบปฏิบัติการวินโดวส์ 95 /98 2000/NT หรือกระทั่ง XP จึงช่วยลดเวลาและขั้นตอนในการพัฒนาโปรแกรมติดต่อกับพอร์ตที่รันบนวินโดวส์ได้อย่างมาก

3.3 ฟังก์ชั่นที่มีอยู่ใน io.dll

ProtOut	ใช้ส่งข้อมูลขนาน 1 ไบต์ไปยังพอร์ตที่กำหนด
ProtWordOut	ใช้ส่งข้อมูล 1 เวิร์ด (16 บิต)ไปยังพอร์ตที่กำหนด
ProtDWordOut	ใช้ส่งข้อมูลดับเบิลเวิร์ด (32 บิต) ไปยังพอร์ตที่กำหนด
PortIn	ใช้อ่านข้อมูลขนาน 1 ไบต์ จากพอร์ตที่กำหนด

PortWordIn	ใช้อ่านข้อมูล 1 เวิร์ด (16 บิต)จากพอร์ตที่กำหนด
ProtDWordIn	ใช้อ่านข้อมูลดับเบิลเวิร์ด (32 บิต) จากพอร์ตที่กำหนด
SetPortBit	ใช้เซตข้อมูลในระดับบิตของพอร์ตที่กำหนด
ClrPortBit	ใช้เคลียร์ข้อมูลในระดับบิตของพอร์ตที่กำหนด
NotPortBit	ใช้กลับข้อมูลในระดับบิตของพอร์ตที่กำหนด
GetPortBit	ใช้อ่านสถานะของบิตที่กำหนด
RightPortShift	ใช้เลื่อนข้อมูลของพอร์ตไปทางขวา จะได้ค่าของบิต LSB กลับมา
LeftPortShift	ใช้เลื่อนข้อมูลของพอร์ตไปทางซ้าย จะได้ค่าของบิต MSB กลับมา
IsDriverInstalled	ใช้ตรวจสอบการติดตั้งไฟล์ io.dll โดยถ้าหากมีการติดตั้งไฟล์จะได้ค่า "1" หรือ "non-zero" กลับมา

3.4 ตัวอย่างการใช้ไฟล์ io.dll ในการเขียนโปรแกรมด้วย Visual Basic

```
Private Declar Sub PortOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Byte)
Private Declare Sub PortWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Integer)
Private Declare Sub PortWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Long)
Private Declare Function PortIn Lib "IO.DLL" (ByVal Port As Integer) As Byte
Private Declare Function PortWordIn Lib "IO.DLL" (ByVal Port As Integer) As Integer
Private Declare Function PortWordIn Lib "IO.DLL" (ByVal Port As Integer) As Long
Private Declare Sub SetPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Private Declare Sub ClrPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Private Declare Sub NotPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Private Declare Function GetPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte) As Boolean
Private Declare Function RightPortShift Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Boolean) As Boolean
Private Declare Function LeftPortShift Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Boolean) As Boolean
```

Private Declare Function IsDriverInstalled Lib "IO.Dll" As Boolean

3.5 การติดต่อระหว่างพอร์ตขนานกับอุปกรณ์อินพุต/เอาต์พุตอย่างง่าย

รีจิสเตอร์ของพอร์ขนานสามารถแบ่งออกได้เป็น 3 รีจิสเตอร์ คือ

1. รีจิสเตอร์ Data ทำหน้าที่เป็นเอาต์พุต
2. รีจิสเตอร์ Status ทำหน้าที่เป็นอินพุต
3. รีจิสเตอร์ Control ทำหน้าที่เป็นเอาต์พุต

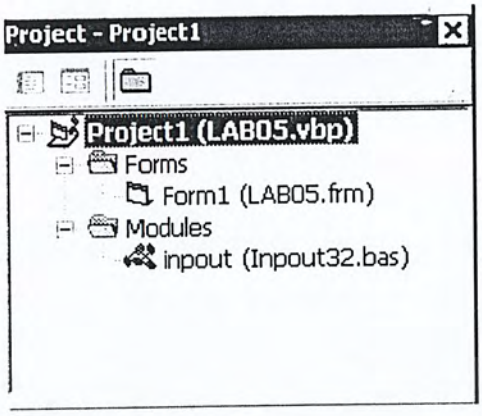
ดังนั้นถ้าผู้ใช้งานต้องการส่งค่าออกเอาต์พุตก็จะต้องใช้รีจิสเตอร์ Data หรือรีจิสเตอร์ Control ส่วนถ้าต้องการรับค่าจากอินพุต ผู้ใช้งานต้องใช้รีจิสเตอร์ Status ในการอ่านค่าอินพุต การจะอ้างถึงรีจิสเตอร์แต่ละตัวนั้น ผู้ใช้งานจะต้องใช้ตำแหน่งแอดเดรสเป็นตัวอ้าง ดังมีรายละเอียดแสดงในตารางที่ 3-2

ตารางที่ 3-2 แอดเดรสรีจิสเตอร์ของพอร์ตขนาน

รีจิสเตอร์	LPT1	LPT2	LPT3
DATA	378H	3BCH	278H
STATUS	379H	3BDH	
CONTROL	37AH	3BEH	27AH

โดยปกติแล้ว Visual Basic ไม่มีคำสั่ง Out กับ QBASIC ดังนั้นเพื่อให้ Visual Basic สามารถใช้คำสั่งนี้ได้จำเป็นต้องเพิ่มไฟล์ inpout32.bas เข้าไปในผังงานหรือโปรเจ็กต์ (Project) ของ Visual BASIC ที่กำลังใช้งานอยู่ โดยต้องทำตามขั้นตอนดังนี้

1. ไปที่เมนู Project เรียกคำสั่ง Add File แล้วเพิ่มไฟล์ inpout32.bas ลงไปใน Project ที่หน้าต่าง Project จะปรากฏไฟล์ inpout32.bas ดังแสดงในรูปที่ 3.1
2. เมื่อเลือกชี้ที่ไฟล์ inpout32.bas แล้วใช้คำสั่ง View Code เพื่อดูรายละเอียดภายในของไฟล์ inpout32.bas ดังในรูปที่ 3.2
3. สำหรับไฟล์ inpout32.bas นั้นจะไปกำหนดคำสั่ง Inp และ Out ให้กับ Visual Basic โดยจะต้องมีไฟล์ io.dll บรรจะอยู่ในไดเรกตอรี่ System อยู่ก่อน ซึ่งได้กล่าวไว้แล้ว
4. เมื่อถึงขั้นตอนนี้ผู้ใช้งานสามารถใช้คำสั่ง inp และคำสั่ง Out ในโปรแกรมเพื่อรับและส่งค่ากับพอร์ตขนานได้แล้ว



รูปที่ 3-1 การเรียกไฟล์ inpout32.bas

3.6 บอร์ดเชื่อมต่อพอร์ตขนาน

วงจรของบอร์ดพอร์ตขนานแสดงในรูปที่ 3.2 โดยเริ่มจากคอนเน็กเตอร์ k002 แบบ DB-25 ตัวเมียอันเป็นจุดที่ใช้ต่อเชื่อมกับพอร์ตขนานของคอมพิวเตอร์ ซึ่งได้รับการจัดสรรออกเป็น 3 ส่วน คือ

1. พอร์ต Data มีตำแหน่งอยู่ที่ขา 2 ถึงขา 9 ใช้ทำหน้าที่เป็นขาเอาต์พุต สัญญาณข้อมูลจะถูกส่งเข้าสู่ไอซีบัฟเฟอร์เบอร์ 74HC541 เพื่อขยายกระแสให้กับขาเอาต์พุต D0-D7 ทั้ง 8 ขา นอกจากนั้นยังทำหน้าที่เป็นตัวป้องกันความเสียหายที่อาจจะเกิดกับพอร์ตขนานอีกด้วย เอาต์พุตจากไอซี 74HC541 จะส่งออกไปยังคอนเน็กเตอร์ Data Bus ซึ่งมีการจัดขาตามมาตรฐาน UIC-10 และส่งออกไปรวมกับคอนเน็กเตอร์ P-BUS โดยคอนเน็กเตอร์ Data Bus และ P-BUS จะใช้ในการเชื่อมต่อกับบอร์ด EX-series ซึ่งเป็นบอร์ดสำหรับทดลองการเชื่อมต่อกับอุปกรณ์ภายนอก

2. พอร์ต Control ใช้ตำแหน่งขา 1, 14, 16 และ 17 โดยต่อเข้ากับไอซี 74HC541 เพื่อขยายกระแสและป้องกันความเสียหายที่จะเกิดกับพอร์ตขนาน แล้วต่อไปเข้าคอนเน็กเตอร์ CONTROL จะเห็นว่าใช้เฉพาะขา C0-C3 พร้อมกันนั้นยังไปรวมกันที่คอนเน็กเตอร์ P-BUS ด้วย

นอกจากนี้พอร์ต Control ยังถูกใช้งานเพื่อเป็นขาเอาต์พุตสำหรับการติดต่อสื่อสารด้วยระบบบัสแบบ I^2C ด้วย โดยจะใช้ขา C1 ในการสร้างสัญญาณ SCL (สัญญาณนาฬิกา) และขา C0 ในการสร้างสัญญาณ SDA (ส่งข้อมูล) โดยใช้ขา S7 ซึ่งอยู่ในส่วนของพอร์ต Status รับข้อมูลจาก SDA การใช้งานระบบบัส I^2C จะเลือกผ่านดิปสวิตช์ เนื่องจากต้องการให้สามารถใช้งานขาพอร์ต Control และพอร์ต Status ในงานปกติได้ เมื่อไม่มีการใช้งานระบบบัส I^2C

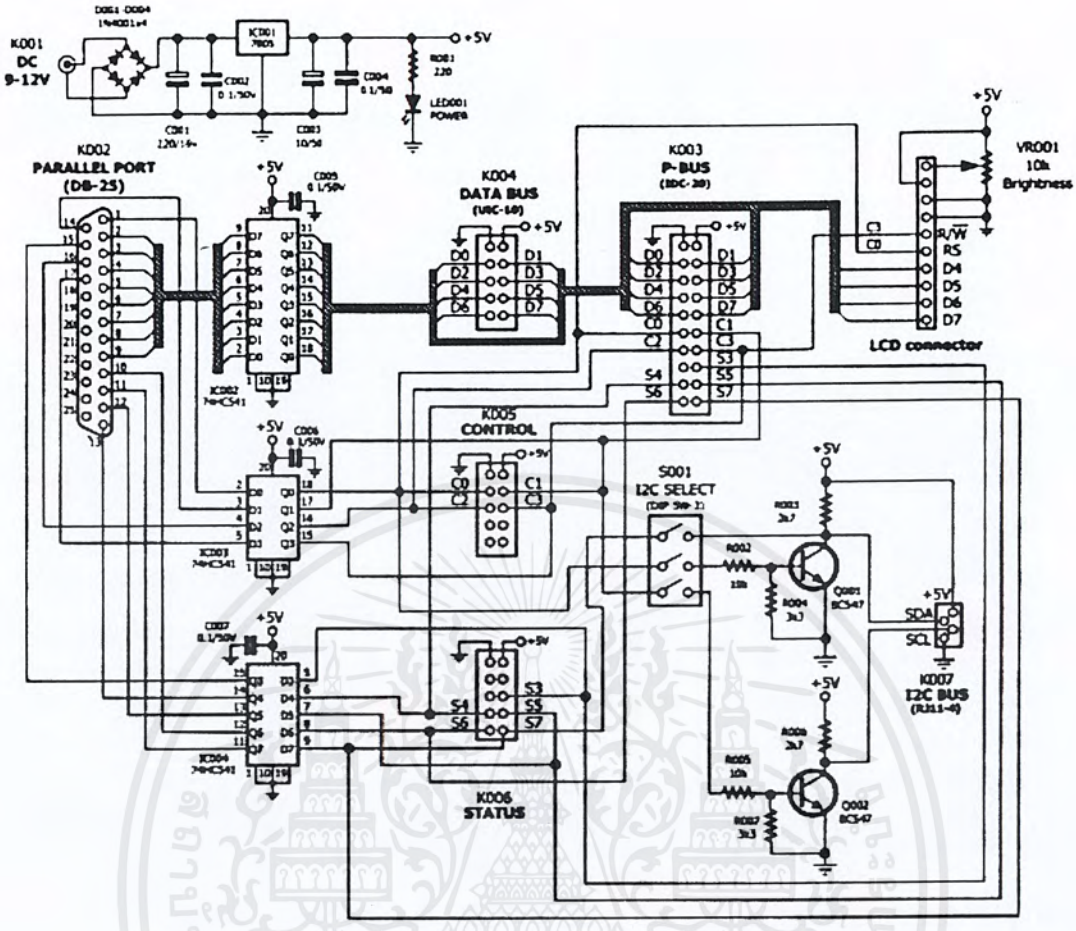
วงจรของส่วนเชื่อมต่อระบบบัส I^2C ของบอร์ดพอร์ตขนานจะใช้ทรานซิสเตอร์ 2 ตัวต่อในลักษณะคอลเล็กเตอร์เปิด ซึ่งในช่วงที่ไม่ได้ป้อนลอจิก "1" ให้ทรานซิสเตอร์จะไม่ทำงานและทาง

ด้านเอาต์พุตของขา SDA และ SCL จะมีลอจิก "1" จากตัวต้านทานพูลอัปค่า 2.1 kΩ ที่ต่อเอาไว้ ซึ่งจะเรียกสภาวะนี้ว่า บัสว่าง และเมื่อต้องการติดต่อกับบัส I^2C จะต้องป้อนลอจิก "1" ให้ทรานซิสเตอร์ทำงาน และให้เอาต์พุตของบัส I^2C ออกมาเป็น "0" เนื่องจากทรานซิสเตอร์จะทำการลัดวงจรขาเอาต์พุตลงกราวด์ สำหรับรายละเอียดการใช้งานระบบ บัส I^2C จะกล่าวถึงอีกครั้งในหัวข้อต่อไป

3. พอร์ต Status ใช้ตำแหน่งขา 15, 13, 12, 10 และ 11 ของพอร์ตขนาน โดยขาเล่านี้เป็นขาอินพุต ดังนั้นไอซีบัฟเฟอร์ที่นำมาต่อด้วยจะเป็นการรับสัญญาณอินพุตจากภายนอกและส่งสัญญาณไปให้กับพอร์ตขนานซึ่งตรงข้ามกับ 2 พอร์ตแรก ไอซีบัฟเฟอร์ที่ใช้ยังเป็นเบอร์ 74HC541 ส่วนอินพุตของพอร์ต Status จะต่อเชื่อมกับคอนเน็กเตอร์ Status ที่จัดขาตามมาตรฐาน UIC-10 เช่นกัน โดยใช้งานเพียง 5 ตำแหน่งเท่านั้นคือ S3, S4, S5, S6 และ S7 โดยขา S7 เป็นขาอินพุตให้กับบัส I^2C ด้วย ขาพอร์ต Status ทั้งหมดหลังจากผ่านบัฟเฟอร์จะไปรวมกันที่คอนเน็กเตอร์ P-BUS

นอกจากนี้บอร์ดพอร์ตขนานได้เตรียมคอนเน็กเตอร์สำหรับเชื่อมต่อกับโมดูล LCD แบบอักขระขนาด 16 ตัวอักษร 1 หรือ 2 บรรทัดเอาไว้ โดยกำหนดโหมดการติดต่อกับโมดูล LCD เป็นแบบ 4 บิต ซึ่งใช้สายสัญญาณ D4-D7 จากพอร์ต Data จะต่อเข้ากับขา D4-D7 ของโมดูล LCD ส่วนขา C0 ของพอร์ต Control จะต่อเข้ากับขา E ของโมดูล LCD ขา C3 ของพอร์ต Control จะต่อเข้ากับขา RS ของโมดูล LCD และเนื่องจากการติดต่อกับโมดูล LCD จะเป็นการเขียนข้อมูลไปอย่างเดียว ดังนั้นขา R/W ของโมดูล LCD จึงไม่ต้องใช้งานขานี้ ให้ต่อลงกราวด์

บอร์ดพอร์ตขนานใช้ไฟเลี้ยงจากภายนอกป้อนเข้ามาทางแจ็กอะแดปเตอร์ ผ่านไดโอดซึ่งต่อกันในลักษณ์บริดจ์เพื่อจัดขั้วของไฟเลี้ยงบบบอร์ดพอร์ตขนานใหม่ จากนั้นจะส่งผ่านไปยังวงจรเรกูเลเตอร์เพื่อแปลงแรงดันให้เท่ากับ +5V สำหรับเป็นไฟเลี้ยงอุปกรณ์บนบอร์ด และเป็นไฟเลี้ยงให้กับวงจรต่อพ่วงต่างๆ ที่เชื่อมต่อกันผ่านคอนเน็กเตอร์ P-Bus , Data Bus และบัส I^2C



รูปที่ 3-2 วงจรสมบูรณ์ของบอร์ดเชื่อมต่อพอร์ตขนาน

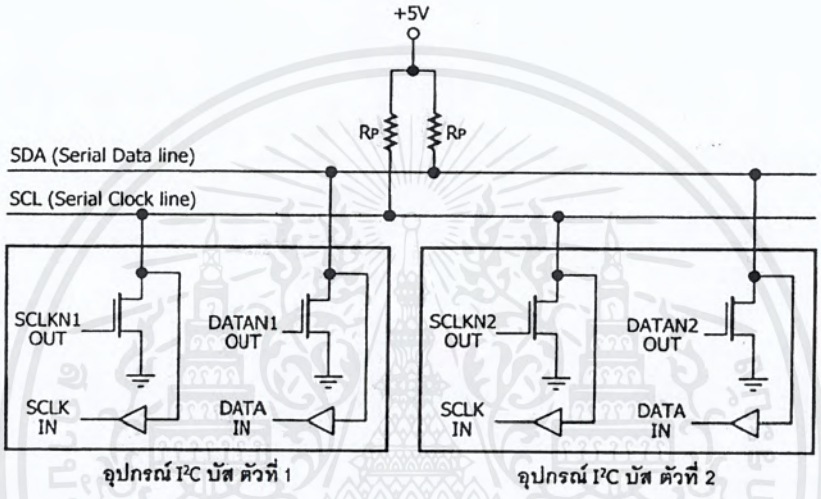
3.7 การขยายความสามารถของพอร์ตขนานผ่านระบบบัส I²C

I²C ย่อมาจาก Inter IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I²C พัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลักก็คือ ต้องการให้ไอซีหรือโมดูลติดต่อสั่งงานและควบคุมภายใต้สัญญาณเพียง 2 เส้น เส้นหนึ่งคือสายข้อมูล อีกเส้นหนึ่งคือสายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I²C ทำได้ง่ายมาก เพียงต่อสายสัญญาณและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกัน ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสแต่ละตัว

สายข้อมูลบนบัส I²C มีชื่อเรียกอย่างเป็นทางการว่าสายข้อมูลอนุกรมหรือ SDA (Serial Data Line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรมหรือ SCL (Serial Clock Line)

3.8 คุณสมบัติโดยทั่วไปของบัส I²C

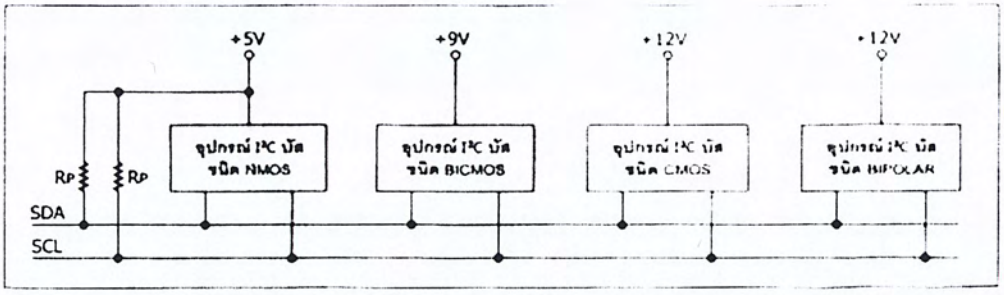
สาย SDA และสาย SCL เป็นสายสัญญาณ 2 ทิศทาง (Bi-directional Line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลาเพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I²C ต้องมีสถานะเปิด (Open Circuit) หรือคอลเล็กเตอร์เปิด (Open Collector) ดังแสดงในรูปที่ 3-3



รูปที่ 3-3 วงจรเอาต์พุตของอุปกรณ์บนระบบบัส I²C

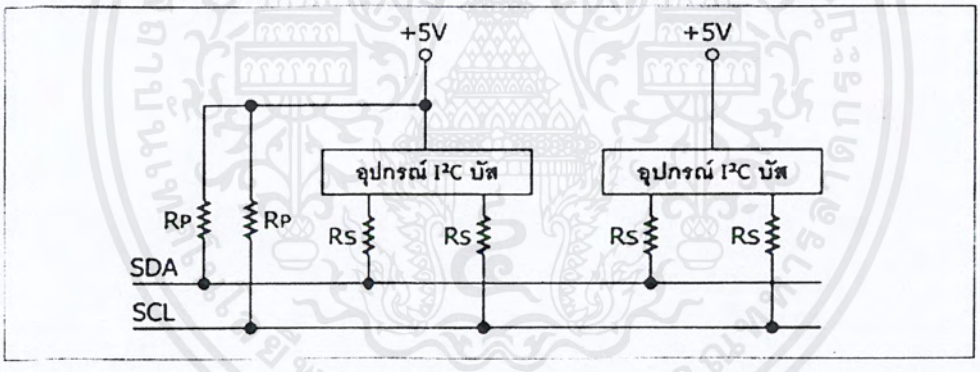
อัตราการถ่ายทอดข้อมูลบนบัส I²C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (Standard Mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (Fast Mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส I²C จะต้องมีความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400pF การเข้าถึงอุปกรณ์บนบัส I²C ทำได้ 2 แบบคือ แบบ 7 บิต (7-bit addressing) 10 บิต (10-bit addressing)

ข้อเด่นอีกประการหนึ่งของบัส I²C คือสามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้ โดยอุปกรณ์บนบัส I²C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12V การต่อร่วมกันบนบัส I²C สามารถกระทำได้ในลักษณะเดียวกันกับกรณีที่อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเดียวกัน กล่าวคือการต่อสาย SDA และ SCL ของอุปกรณ์แต่ละตัวเข้าด้วยกันต้องต่อตัวต้านทานพูลอัป (R_p) เข้ากับแรงดัน +5V ไว้ด้วยเสมอ ดังแสดงในรูปที่ 3-4



รูปที่ 3-4 การต่อพ่วงอุปกรณ์ระบบบัส I²R ที่ใช้ไฟเลี้ยงไม่เท่ากัน

ในกรณีที่อาจมีแรงดันไฟกระชากขนาดใหญ่ปะปนเข้ามาในบัส I²R ที่ขา SDA และ SCL ของอุปกรณ์แต่ละตัวต้องต่อความต้านทานอนุกรมกับขา SDA และ SCL ที่เรียกว่า RS ก่อนต่อเข้าสู่บัส I²C ดังแสดงในรูปที่ 3- 5



รูปที่ 3-5 การต่อตัวต้านทานอนุกรมกับขาสัญญาณของอุปกรณ์ บนระบบบัส I²C เพื่อลดสัญญาณรบกวน

3.9 หลักการของบัส I²C

บัส I²C ประกอบด้วยสายสัญญาณสองเส้น ดังที่ได้กล่าวมาแล้วคือสาย SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่าโปรโตคอล (Protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง โดยจะได้อธิบายลักษณะ หน้าที่และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I²C เพื่อเป็นข้อตกลงพื้นฐานในการอธิบายการทำงานของบัส I²C

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่าตัวส่ง (Transmitter) ส่วนอุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่าตัวรับ (Receiver) อุปกรณ์บนบัส I²C สามารถเป็นได้ทั้งตัวรับและตัวส่ง อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I²C เรียกว่า มาสเตอร์ (Master) อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I²C เรียกว่า สเลฟ (Slave)

ข้อกำหนด 2 ประการที่สำคัญของการติดต่อบนบัส I²C คือ

1. การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

2. ในระหว่างการถ่ายทอดข้อมูลเมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

สภาวะที่เกิดขึ้นบนบัส I²C มีด้วยกัน 5 สภาวะ ดังนี้

1. บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั้นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

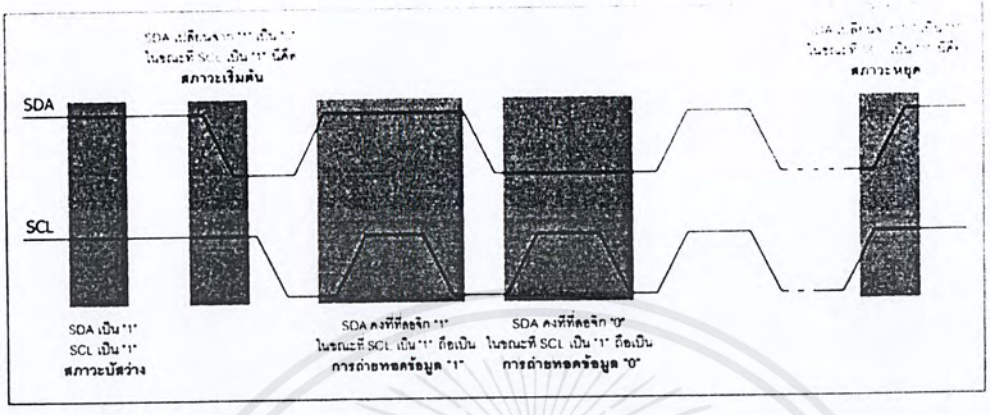
2. เริ่มต้นถ่ายทอดข้อมูล (Start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะนี้ว่า สภาวะเริ่มต้น (START)

3. หยุดการถ่ายทอดข้อมูล (Stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP)

4. ข้อมูลดำรงอยู่บนบัส (Data valid) สภาวะนี้เกิดขึ้นต่อจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอดเมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่เพื่อให้อุปกรณ์รับข้อมูลในจังหวะนั้นว่า เป็น 0 หรือ 1 ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกคู่แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูงหากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ถ่ายทอดเกิดความผิดพลาดขึ้น

5. รับรู้ข้อมูล (Acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตเรียกว่าบิตรับรู้ (Acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกาเพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟที่ถูกอ้างอิงถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

ในรูปที่ 3-6 เป็นไดอะแกรมเวลาที่แสดงถึงการเกิดสภาวะต่าง ๆ บนบัส I²C ไม่ว่าจะเป็นสภาวะบัสว่าง สภาวะเริ่มต้น การถ่ายทอดข้อมูล การรับรู้และสภาวะการหยุดการถ่ายทอดข้อมูล



รูปที่ 3-6 ไดอะแกรมแสดงสถานะต่าง ๆ บนระบบบัส I²C

3.10 การทำงานบนบัส I²C

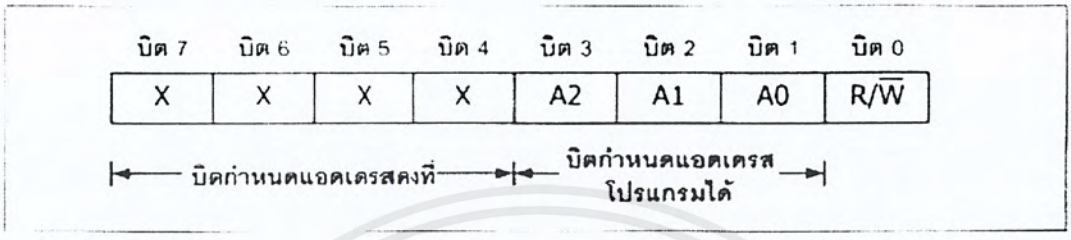
ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่าง ๆ ที่ต่อยู่บนบัส ต้องมีการอ้างถึงเสียก่อนโดยการอ้างถึงอุปกรณ์บนบัส I²C จะต้องใช้การอ้างถึงแบบ 7 บิตหหรือ10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มากนักใช้การอ้างถึงแบบ 7 บิตก็เพียงพอแต่ถ้ามีอุปกรณ์ต่ออยู่บนบัส มากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิตหลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้แล้ว ก็จะเริ่มต้นถ่ายทอดข้อมูลกันต่อไป

ดังนั้นหลักสำคัญในอันดับแรกของการทำงานบนบัส I²R ก็คือการอ้างถึงอุปกรณ์แต่ละตัว

3.10.1 การอ้างถึงแบบ 7 บิต (7 bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อหรือข้อมูลกำหนดแอดเดรส โดยมีรูปแบบดังแสดงในรูปที่ 3-7 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์ที่ต้องการติดต่อโดยแบ่งเป็นบิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสภาวะลอจิกให้แก่ขา A0 – A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I²C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้น ๆ หากบิต LSB เป็น “0” หมายถึงการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น “1” จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ

ข้อมูลในไบต์ต่อมาคือข้อมูลควบคุม (Control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันออกไปเช่น ไอซีขยายพอร์ตที่มีข่อมูลควบคุมที่ใช้กำหนดว่าบิตใดเป็นบิตอินพุตบิตใดเป็นบิตเอต์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น



รูปที่ 3-7 รูปแบบของข้อมูลกำหนดแอดเดรสของอุปกรณ์บนระบบบัส I²C

หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณตอบรับกลับมาด้วยทุกครั้ง เพื่อให้ขบวนการถ่ายทอดข้อมูลยังสามารถดำเนินต่อไปได้ ในรูปที่ 3-8 แสดงรูปแบบข้อมูลอนุกรมที่เกิดขึ้นในการติดต่อบนบัส I²C ของการอ้างถึงแบบ 7 บิต

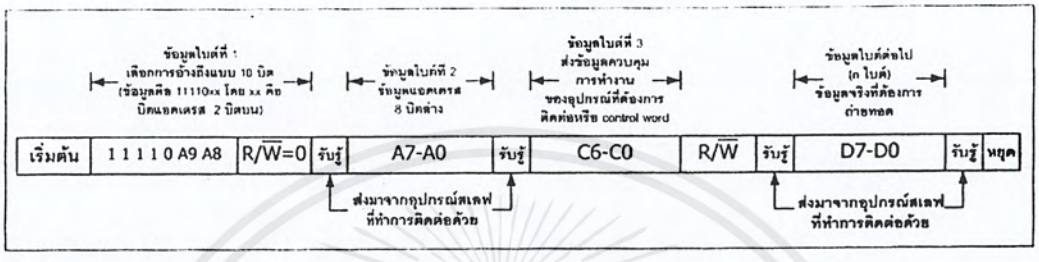
รูปที่ 3- 8 รูปแบบของข้อมูลที่ใช้ในการอ้างถึงแบบ 7 บิตของระบบบัส I²C

3.10.2 การอ้างถึงแบบ 10 บิต (10 bit addressing)

ในการอ้างแบบนี้ยังคงใช้รูปแบบอนุกรมที่เหมือนกับแบบ 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบร์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่า ต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการ

ติดต่อด้วยข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงจะเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นจะเป็นข้อมูลจริงที่ใช้ในการติดต่อ

เช่นเดียวกับการอ้างถึงแบบ 7 บิตหลังจากถ่ายทอดข้อมูลครบทุกไบต์ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้ขบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ ในรูปที่ 3-9 แสดงรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ 10 บิต

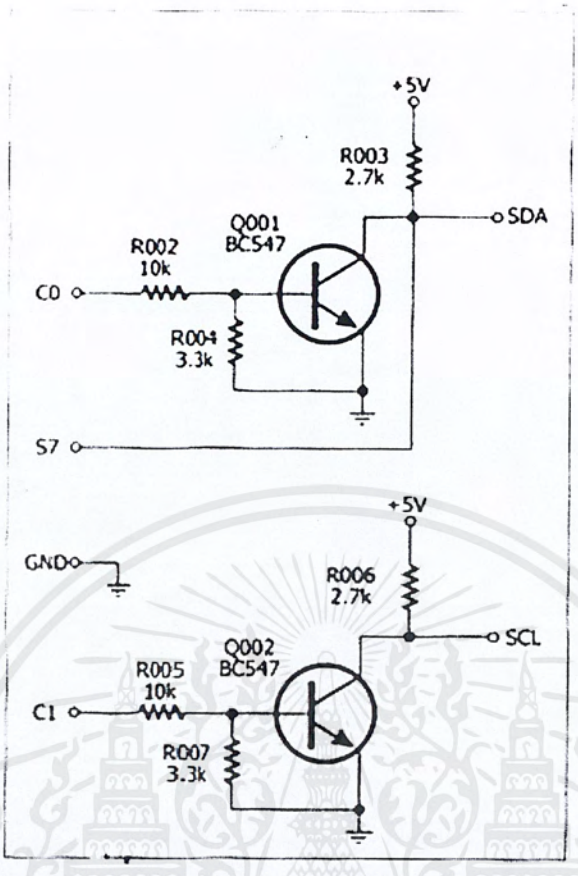


รูปที่ 3-9 รูปแบบของข้อมูลที่ใช้ในการอ้างถึงแบบ 10 บิต ของระบบบัส I²C

3.11 การต่อระบบบัส I²C กับบอร์ดพอร์ตขนาน

เพื่อให้พอร์ตขนานสามารถติดต่อกับอุปกรณ์ที่มีการเชื่อมต่อเป็นบัส I²C จะต้องสร้างวงจรเชื่อมต่อขึ้นมา ดังแสดงในรูปที่ 3-10 ทรานซิสเตอร์ Q001 และ Q002 ได้รับการจัดวงจรให้มีลักษณะเป็นวงจรบัฟเฟอร์แบบคอลเล็กเตอร์เปิดตามข้อกำหนดของวงจรเอาต์พุตของบัส I²C โดย Q001 ใช้ถ่ายทอดสัญญาณของสาย SDA ในขณะที่Q002 ทำหน้าที่ถ่ายทอดสัญญาณของสาย SCL ไฟเลี้ยงของวงจรคือ +5V จึงสามารถใช้ทรานซิสเตอร์แบบ NPN ตามมาตรฐานเบอร์ใดก็ได้ที่สามารถตอบสนองความถี่สูงถึง 100 kHz

สำหรับบอร์ดพอร์ตขนานที่ใช้ในการทดลองนี้ ได้กำหนดให้ข้อมูลถ่ายทอดสู่สาย SDA ของบัส I²C ผ่านทางขา C0 และรับข้อมูลผ่านขา S7 ส่วนสายสัญญาณนาฬิกาจะถูกส่งออกมาทางขา C1 เพื่อเป็นสาย SCL สำหรับระบบบัส I²C



รูปที่ 3-10 วงจรเชื่อมต่อระบบบัส I²C ของบอร์ดเชื่อต่อพอร์ตขนาน

3.12 การเขียนโปรแกรมเพื่อสร้างสัญญาณต่าง ๆ สำหรับบัส I²C

เพื่อให้ง่ายต่อการเรียกใช้งาน ดั้งนั้นการติดต่อกับบัส I²C โดยพอร์ตขนานจึงต้องเขียนโปรแกรมย่อยเพื่อสร้างสภาวะต่าง ๆ ของระบบบัส I²C เพื่อสร้างเป็นโมดูลสำหรับติดต่อกับบัส I²C ซึ่งมีรายละเอียดดังต่อไปนี้

สัญญาณ SDA "0"

```
Private Sub SDA_L()
        Out &H37A,  Inp(&H37A)  And &HFE
End Sub
```

สัญญาณ SCL "1"

```
Private Sub SDA_H()
        Out &H37A,  Inp(&H37A)  or 1
```

```
End Sub
```

สัญญาณ SCL"0"

```
Private Sub SCL_L()
        Out &H37A,  Inp(&H37A)  And  &HFD
End Sub
```

สัญญาณ SCL"1"

```
Private Sub SCL_H()
        Out &H37A,  Inp(&H37A)  or 2
End Sub
```

สัญญาณ SDA

```
Private Function Rd_SDA()  As  Boolean
        SDA_H
        Rd_SDA = (Inp(&H379)  And  &H80 < > &H80
End  Sub
```

สัญญาณ Start

```
Public  Sub  I2Cstart90
        SDA_H
        SCL_H
        SDA_L
        SCL_L
End  Sub
```

สัญญาณ Stop

```
Public  Sub  I2Cstop()
        SDA_L
        SCL_H
        SDA_H
End  Sub
```

สัญญาณ "0"

```
Public Sub Send0()
        SDA_L
        SCL_H
        SCL_L
End Sub
```

สัญญาณ "1"

```
Public Sub Send()
        SDA_H
        SCL_H
        SCL_L
End Sub
```

สัญญาณ Acknowledge

```
Public Function Ack() As Boolean
        Ack = Not Rd_SDA
        SCL_H
        SCL_L
End Function
```

การอ่านสัญญาณ Acknowledge จากอุปกรณ์สเลฟจะช่วยให้ตรวจสอบได้ว่าอุปกรณ์ที่ติดต่อนั้น ยังทำงานอยู่หรือไม่

ส่งสัญญาณ Master Acknowledge

```
public Sub Mack()
        SDA_L
        SCL_H
        SCL_L
End Sub
```

ส่งสัญญาณ Master Not Acknowledge

```
Public  Sub Mack()
        SDA_L
        SCL_H
        SCL_L
End  Sub
```

ส่งสัญญาณ Master Not Acknowledge

```
Public  Sub  MNAck()
        SCL_H
        SCL_L
End  Sub
```

3.12.1 โปรแกรมย่อยรับส่งหรืออ่านเขียนข้อมูล 8 บิต

ในการส่งและรับข้อมูล 8 บิตบนระบบบัส I^2C สามารถเขียนโปรแกรมย่อยได้ดังนี้

ส่งข้อมูล 8 บิต

```
Public Sub Send8BIT(Data As Byte)
Dim I As Integer
        For I = 7 To 0 Step –1
                If (Data And 2 ^ I) = 2 ^ i  Then
                        Call Send1
                Else
                        Call Send0
                End If
        Next I
End Sub
```

อ่านข้อมูล 8 บิต

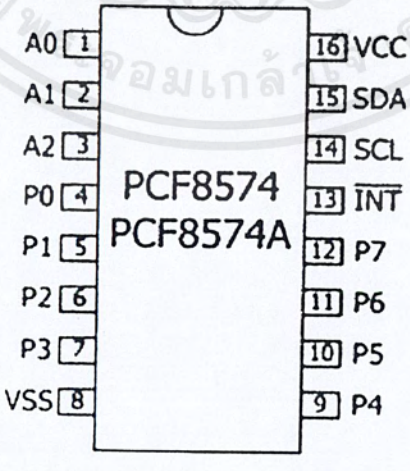
```
Public Function Read8Bit() As Byte
Dim Dat1 As Integer
Dim i As Integer
```

```
        For i = 7 To 0 Step –1
                If Rd_SDA Then
                        Dat1 = (2 ^ i) or Dat1
        End If
        SCL_H
        SCL_L
  Next i
  Read8Bit = Dat1
End Function
```

อนึ่งสำหรับโปรแกรมย่อยทั้งหมดที่สร้างขึ้นเป็นโมดูลสำหรับติดต่อบนระบบบัส I^2C สามารถศึกษาเพิ่มเติมได้ที่หัวข้อที่ 3

3.13 การขยายจำนวนพอร์ตอินพุตเอาต์พุตด้วยไอซี PCF8574A

สำหรับการใช้งานอินพุตเอาต์พุตจำนวนมากนั้น ขาของพอร์ตขนานอาจจะไม่มากพอที่จะนำไปใช้ในงานได้โดยตรง จึงต้องต่ออุปกรณ์เพิ่มเติมเพื่อขยายจำนวนพอร์ตอินพุตเอาต์พุตไอซีขยายพอร์ตผ่านระบบบัส I^2C ที่เลือกใช้นี้คือเบอร์ PCF8574A ที่สามารถขยายพอร์ตอินพุตเอาต์พุตได้ตัวละ 8 ช่อง และต่อพ่วงกันได้ 8 ตัว ทำให้สามารถขยายพอร์ตได้มากถึง 64 ช่อง



รูปที่ 3-11 (ก) การจัดขาของไอซี PCF8547A

ชื่อ	ตำแหน่งขา	หน้าที่
A0	1	อินพุตแอดเดรสตัวที่ 1
A1	2	อินพุตแอดเดรสตัวที่ 2
A2	3	อินพุตแอดเดรสตัวที่ 3
P0	4	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 0
P1	5	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 1
P2	6	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 2
P3	7	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 3
VSS	8	กราวด์
P4	9	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 4
P5	10	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 5
P6	11	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 6
P7	12	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 7
INT	13	ขาเอาต์พุตอินเตอร์รัปต์(ทำงานที่ลอจิก 0)
SCL	14	ขาสัญญาณนาฬิกาสำหรับ I^2C บัส
SDA	15	ขาข้อมูลสำหรับ I^2C บัส
VDD	16	ไฟเลี้ยง

รูปที่ 3-11(ข) ชื่อขาใช้งานของไอซี PCF8547A

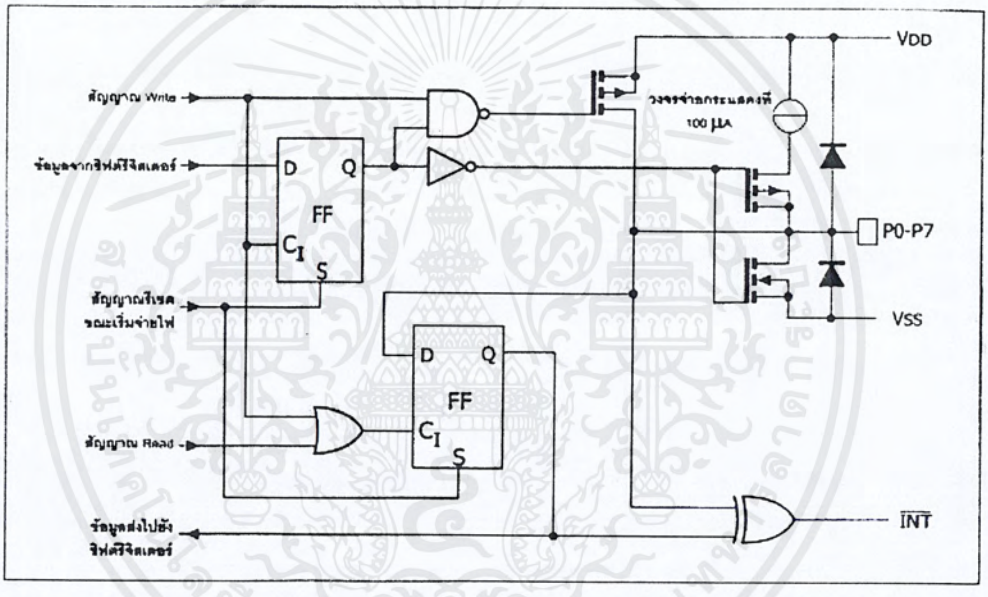
ข้อมูลเบื้องต้นของ PCF8574A มีดังนี้

- ทำงานที่ระดับแรงดันตั้งแต่ 2.5V ถึง 6V
- กินกระแสในสภาวะสแตนด์บายต่ำเพียง 10μA
- ใช้ในการเชื่อมต่อแบบบัส I^2C
- มีเอาต์พุตอินเตอร์รัปต์แบบเดรนเปิด
- เอาต์พุตสามารถแลตช์ค่าได้ขับกระแสได้ 100μA ถ้าต้องการขับ LED โดยตรงต้องต่อตัวต้านทานพูลอัปค่า 4.7kΩ เข้าที่ขาพอร์ตที่กำหนดให้เป็นเอาต์พุต
- สามารถกำหนดตำแหน่งแอดเดรสของไอซีทางฮาร์ดแวร์ด้วยขา A0-A2 ทำให้สามารถต่อพ่วงกันได้ถึง 8 ตัว

การจัดขาและหน้าที่การทำงานไอซี PCf8574A แสดงในรูปที่ 3-11 ขาพอร์ตทั้ง 8 ขาของ PCF8574A สามารถกำหนดให้เป็นอินพุตหรือเอาต์พุตได้โดยอิสระ ลักษณะวงจรภายในของพอร์ตอินพุต/เอาต์พุต แสดงในรูปที่ 3-12 เมื่อจ่ายไฟให้กับ PCF8574A ครั้งแรกขาพอร์ตทั้ง 8 ขาจะมีลอจิกเป็น “1” ซึ่งจะเป็นการจ่ายกระแสมาจากแหล่งจ่ายกระแสคงที่ภายในตัวไอซี ทำให้มี

กระแสในลอจิก “1” นี้เพียง 100μA เท่านั้น ในกรณีที่ต้องการให้มีการจ่ายกระแสสูง ๆ จำเป็นต้องต่อตัวต้านทานพูลอัปเอาไว้ที่ขาพอร์ตเหล่านี้ด้วย

เมื่อต้องการให้ขาพอร์ตเหล่านี้ทำหน้าที่เป็นอินพุตจะต้องส่งสัญญาณให้ขาเหล่านี้มีลอจิก “1” เสียก่อน เมื่อขาอินพุตได้รับสัญญาณจากภายนอกป้อนเข้ามาไอซี PCF8574A จะสร้างสัญญาณอินเตอร์รัปต์ (INT) ป้อนให้คอมพิวเตอร์รับรู้แทนการต้องคอยตรวจสอบขาอินพุตอยู่ตลอดเวลาสัญญาณอินเตอร์รัปต์นี้จะถูกรีเซตเมื่อมีการอ่านค่าข้อมูลหรือมีการเปลี่ยนค่าของอินพุตไปสู่ค่าเดิม



รูปที่ 3-12 รายละเอียดวงจรขาพอร์ตของไอซี PCF8547A

การขยายจำนวนพอร์ตอินพุตเอาต์พุตด้วยไอซี PCF8574A

เนื่องจาก PCF8574A มีการเชื่อมต่อเป็นแบบบัส I^2C ดังนั้นการติดต่อจึงต้องอ้างถึงโปรแกรมย่อยดังที่กล่าวไปแล้วในตอนต้น โดยจะต้องส่งข้อมูลแอดเดรสเพื่อติดต่อกับ PCF8574A ดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	1	1	1	A2	A1	A0	R/$\overline{W}$

บิต A0, A1, A2 ใช้ในการระบุ PCF8574A ที่ใช้ในบอร์ดในกรณีที่มีการต่อ PCF8574A มากกว่า 1 ตัว โดยค่าของ A0-A2 จะมีความแตกต่างกันไปในแต่ละตัวบิต R/$\overline{W}$ ใช้กำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซี PCF8574A จากการกำหนดแอดเดรส A0-A2 จะให้สามารถขยายพอร์ตอินพุต/เอาต์พุตได้มากถึง 64 จุดจากการต่อพ่วง PCF8574A ร่วมกันครบ 8 ตัว

ตัวอย่างโปรแกรมย่อย Sendout ต่อไปนี้แสดงให้เห็นถึงการส่งค่าไปยังเอาต์พุตของ PCF8574A โดยมีลำดับขั้นตอน ดังนี้

โปรแกรมย่อย Sendout เพื่อส่งข้อมูล 8 บิต

```
Privat Sub Sndout (B As Byte)
   Call  I2Cstart
   Call  Send8Bit(&H70)
   Call  Ack
   Call  I2Cstop
End Sub
```

1. ส่งสัญญาณ START
2. ส่งข้อมูลกำหนดแอดเดรสโดยในที่นี้จะกำหนดแอดเดรสของ PCF8574A ไว้ที่ "000" (ขา A0, A1, A2 ต่อลงกราวด์ทั้งหมดและให้ทำงานในโหมดเขียนข้อมูล โดยการป้อนลอจิก "0" ให้แก่บิต R/$\overline{W}$)
3. รอรับสัญญาณ ACK หรือรอรับการตอบกลับจาก PCF8574A
4. ส่งข้อมูลไปยังเอาต์พุตของ PCF8574A โดยการกำหนดค่าไปยังตัวแปร B
5. รอรับสัญญาณ ACK อีกครั้ง
6. ส่งสัญญาณ STOP

3.14 บอร์ดขยายอินพุต/เอาต์พุตขนาด 16 บิตผ่านระบบบัส I^2C

บอร์ดจะมีไอซี PCF8574A อยู่บนบอร์ดทั้งหมด 2 ตัว ทำให้สามารถนำไปใช้ในการขับอุปกรณ์เอาต์พุตหรืออ่านค่าจากอุปกรณ์อินพุตได้ตามต้องการมากถึง 16 ช่อง โดยมีรายละเอียด

ของวงจรดังแสดงในรูปที่ 3-12 บนบอร์ดจะมีจั๊มเปอร์สำหรับเลือกแอดเดรสของ PCF8574A แต่ละตัวในการใช้งานจะต้องกำหนดแอดเดรสไว้คนละตำแหน่งกัน ไม่เช่นนั้นจะทำให้เกิดการทำงานที่ทับซ้อนกันซึ่งอาจส่งผลถึงขั้นไอซีเสียหายได้

เนื่องจากใช้การติดต่อเป็นระบบบัส I^2C ทำให้สามารถต่อบอร์ดพ่วงกันได้หลายๆบอร์ด เพื่อขยายพอร์ตให้มีจำนวนมากขึ้น โดยสามารถต่อพ่วงได้สูงสุด 8 บอร์ด (รวมบอร์ดเริ่มต้นด้วย) และเพื่ออำนวยความสะดวกในการต่อพ่วงบนบอร์ดได้จัดเตรียมแจ็กสำหรับเชื่อมต่อเป็นแบบโมดูล่าร์หรือที่เรียกกันทั่วไปว่าแจ๊กโทรศัพท์ขนาด 4 ขาจำนวน 2 ชุดพร้อมสายสำหรับเชื่อมต่อ

วงจรเอาต์พุตของ PCF8574A เป็นแบบคอลเล็กเตอร์เปิด ดังนั้นที่ขาพอร์ตของ PCF8574A จะต้องต่อตัวต้านทานพูลอัปเพื่อกำหนดให้ในภาวะปกติ ขาพอร์ตมีสถานะลอจิก "1" เอาต์พุตของ PCF8574A จะต่อกับคอนเน็กเตอร์ Data Bus ซึ่งสามารถนำไปต่อกับบอร์ดตัวอื่น ๆ ได้ เช่น ต่อกับวงจรเพื่อขับรีเลย์ ต่อกับบอร์ดเพื่อขับ LED แสดงค่าข้อมูลต่อกับบอร์ดเพื่อรับค่าจากสวิตช์ หรือต่อกับบอร์ดวงจรเพื่อขับสเต็ปเปอร์มอเตอร์ เป็นต้น ไฟเลี้ยงที่ใช้สำหรับบอร์ดมาจากบอร์ดพอร์ตขนานผ่านทางแจ็ก I^2C ดังนั้นจึงไม่จำเป็นต้องหาแหล่งจ่ายไฟเพิ่มเติม

3.15 การเชื่อมต่อสัญญาณอะนาลอกของพอร์ตขนานผ่านระบบบัส I^2C

ปกติแล้วข้อมูลในการติดต่อกับพอร์ตขนานของคอมพิวเตอร์นั้นจะเป็นสัญญาณดิจิตอลทั้งสิ้น แต่เมื่อนำมาต่อกับอุปกรณ์ภายนอกแล้วย่อมต้องเชื่อมต่อและประมวลผลกับสัญญาณอะนาลอกด้วย ในการเชื่อมต่อกับสัญญาณอะนาลอกต้องใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่เรียกว่าไอซี ADC (Analog to Digital Converter) เพื่อให้พอร์ตขนานสามารถอ่านค่าสัญญาณอะนาลอกจากภายนอกเข้ามาประมวลผลได้ โดยไอซี ADC ที่ใช้เป็นเบอร์ PCF8591 ซึ่งมีรูปแบบในการเชื่อมต่อแบบบัส I^2C

3.15.1 ข้อมูลเบื้องต้นของ PCF8591

PCF8591 เป็นไอซีทำหน้าที่แปลงสัญญาณอะนาลอกเป็นดิจิตอลและแปลงสัญญาณดิจิตอลเป็นอะนาลอกในตัวเดียวกันด้วยความสมารถที่รวมเอาวงจร ADC และ DAC เข้าไว้ในไอซีเพียงตัวเดียวทำให้สามารถนำไอซีประเภทนี้ไปใช้งานได้กว้างขวาง โดยคุณสมบัติทางเทคนิคที่สำคัญของ PCF8591 มีดังนี้

1. ทำงานด้วยความละเอียดของข้อมูลดิจิตอลขนาน 8 บิต
2. มีวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล (ADC) ขนาด 8 บิต 4 ช่อง
3. มีวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก (DAC) ขนาด 8 บิต 1 ช่อง
4. ทำงานโดยใช้แหล่งจ่ายไฟเพียงชุดเดียว ตั้งแต่ 2.5 – 6 V
5. กินกระแสไฟฟ้าขณะอยู่ในสภาวะสแตนบายต์ต่ำ

6. ติดต่อกับไมโครคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์ผ่านระบบบัส I^2C

7. สามารถเลือกตำแหน่งแอดเดรสทางอาร์ดแวร์จากขา A0, A1, A2 ทำให้สามารถต่อพ่วงกันได้สูงสุดถึง 8 ตัว จึงขยายจำนวนช่องอินพุตของสัญญาณอะนาลอกได้สูงถึง 32 ช่อง และช่องเอาต์พุตของสัญญาณอะนาลอกจากวงจร DAC สูงถึง 8 ช่อง

8. อัตราการสุ่มข้อมูล ขึ้นอยู่กับความเร็วของสัญญาณนาฬิกาบนบัส I^2C

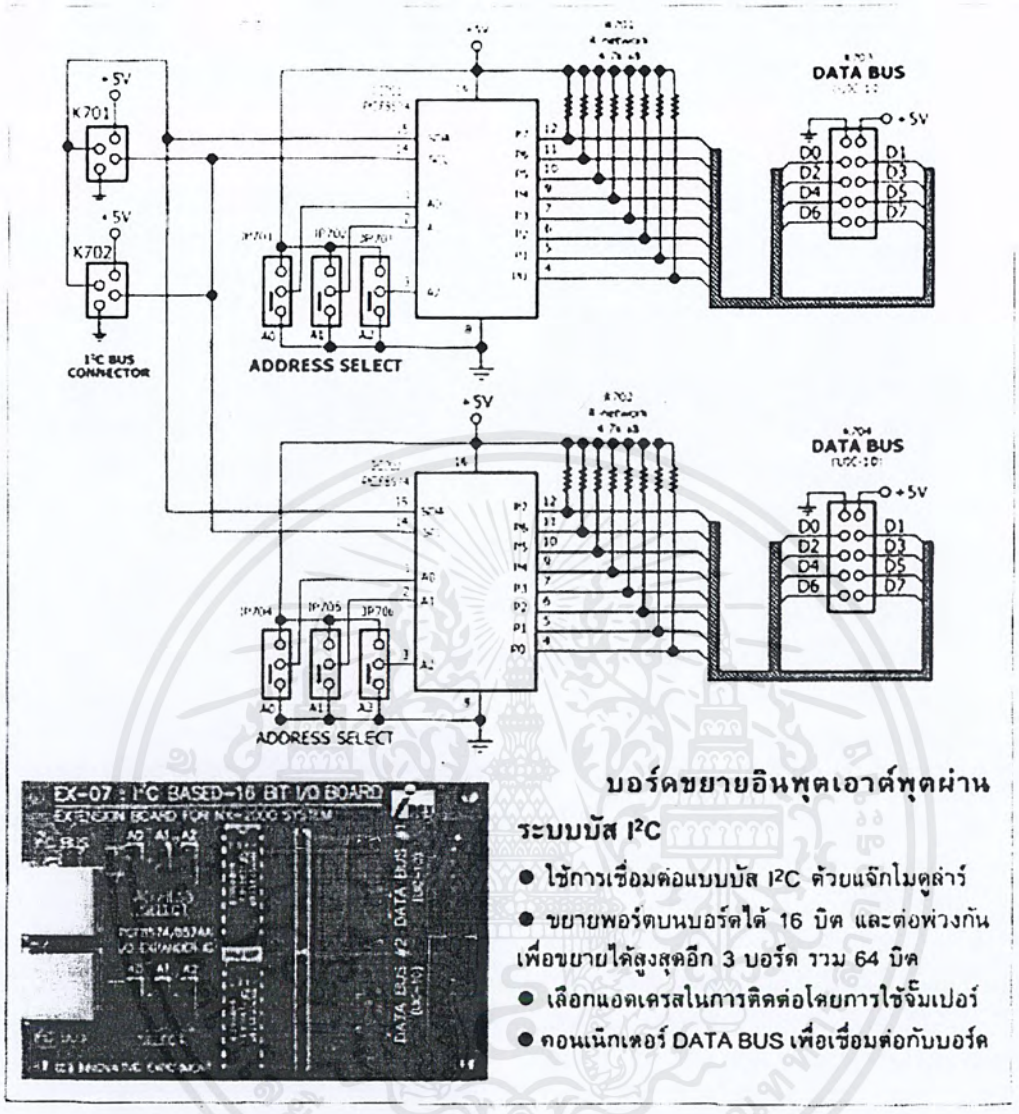
9. วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล สามารถเลือกการทำงานเป็นแบบแยกช่องหรือทำงานเป็นแบบดิฟเฟอเรนเชียลก็ได้

10. การอ่านค่าสามารถกำหนดให้เลื่อนช่องอินพุตโดยอัตโนมัติได้

11. สามารถรับสัญญาณอะนาลอกระดับแรงดันตั้งแต่ V_{SS} ไปจนถึง V_{DD}

12. วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลเป็นแบบ Successive Approximation

PCF8591 สามารถทำงานเป็นไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 8 บิต 4 ช่อง และเป็นไอซีแปลงสัญญาณดิจิตอลเป็นอะนาลอกได้ในคราวเดียวกัน ด้วยการควบคุมผ่านระบบบัส I^2C ทำให้สามารถต่อพ่วงไอซี PCF8591 ได้สูงสุด 8 ตัว รองรับการอ่านค่าสัญญาณอะนาลอกอินพุตได้สูงถึง 32 ช่องและส่งสัญญาณอะนาลอกเอาต์พุตสูงสุด 8 ช่องด้วยการกำหนดแอดเดรสจากขา A0, A1 และ A2 การจัดขาและรายละเอียดขาของ PCF8491 ดังแสดงในรูปที่ 3-13



รูปที่ 3-13 วงจรสมบูรณ์ของบอร์ดขยายอินพุตเอาต์พุตผ่านระบบบัส I^2C

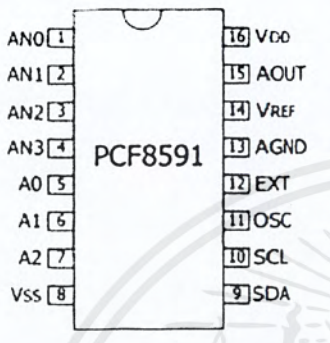
3.15.2 รายละเอียดฟังก์ชั่นต่าง ๆ ของ PCF8591

3.15.2.1 ตำแหน่งแอดเดรส

ในระบบบัส I^2C การติดต่อกับอุปกรณ์แต่ละตัวต้องระบุแอดเดรสของอุปกรณ์เหล่านั้นอย่างชัดเจนถ้าเป็นการอ้างถึงแบบ 7 บิต ข้อมูลกำหนดแอดเดรส 4 บิตบนจะเป็นแอดเดรสเฉพาะของอุปกรณ์ตัวนั้น ๆ ที่กำหนดจากผู้ผลิต ผู้ใช้งานไม่สามารถเปลี่ยนแปลงได้สำหรับไอซี PCF8591 จะมีค่าเท่ากับ 1001 (ฐานสอง) ข้อมูล 3 บิตถัดมาจะเป็นค่าแอดเดรสที่ผู้ใช้งานสามารถกำหนดได้ทางฮาร์ดแวร์เพื่อเลือกไอซีเบอร์ PCF8591 ที่ต้องการติดต่อด้วยในกรณีที่มีการต่อใช้งาน

มากกว่า 1 ตัว ส่วนบิต LSB ใช้ในการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซีตัวนั้น ๆ โดยมีรูปแบบการกำหนดค่าดังต่อไปนี้

ตัวอย่างเช่น ถ้าต้องการอ่านข้อมูลจากชิปที่กำหนดแอดเดรสไว้เป็น 000 จะต้องป้อนข้อมูลแอดเดรสเท่ากับ &H91 เป็นต้น



ขา	ตำแหน่งขา	หน้าที่
AN0	1	[illegible]
AN1	2	[illegible]
AN2	3	[illegible]
AN3	4	[illegible]
A0	5	[illegible]
A1	6	[illegible]
A2	7	[illegible]
VSS	8	กราวด์
SDA	9	ขาข้อมูลสำหรับบัส I²C
SCL	10	ขาสัญญาณนาฬิกาสำหรับบัส I²C
OSC	11	ขาอินพุตเอาต์พุตออสซิลเลเตอร์
EXT	12	ขาเลือกออสซิลเลเตอร์ภายในหรือภายนอก
AGND	13	กราวด์อะนาลอก
VREF	14	ขาอินพุตสำหรับแรงดันอ้างอิง
AOUT	15	ขาอะนาลอกเอาต์พุตของ DAC
VDD	16	ไฟเลี้ยง

รูปที่ 3-14 การจัดขาและตารางแสดงชื่อขาสัญญาณของ PCF8591

3.15.2.2 ข้อมูลควบคุม

หลังจากส่งข้อมูลกำหนดแอดเดรสให้แก่ PCF8591 แล้วต้องส่งข้อมูลควบคุมตามไปด้วยเพื่อกำหนดคุณสมบัติของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลและวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกภายใน PCF8591 โดยมีรายละเอียดของข้อมูลในแต่ละบิต ดังแสดงในรูปที่ 3-14

บิต 6 ของข้อมูลควบคุมใช้สำหรับการเอ็นนาเบิลขาอะนาลอกเอาต์พุต เมื่อต้องการเอ็นนาเบิลต้องกำหนดให้ขานี้เป็น 1

บิต 4 และบิต 5 ของข้อมูลควบคุมใช้สำหรับกำหนดรูปแบบของสัญญาณอะนาลอกอินพุตที่ป้อนให้แก่ PCF8591

บิต 2 ใช้สำหรับเลือกรูปแบบอ่านข้อมูลจากขาอินพุตอะนาลอกว่าจะเป็นอ่านจากเพียงอินพุตเดียวหรืออ่านแบบเรียงลำดับทุกอินพุต ถ้าต้องการเลือกให้อ่านแบบเรียงลำดับต้องกำหนดให้บิตนี้เป็น “1”

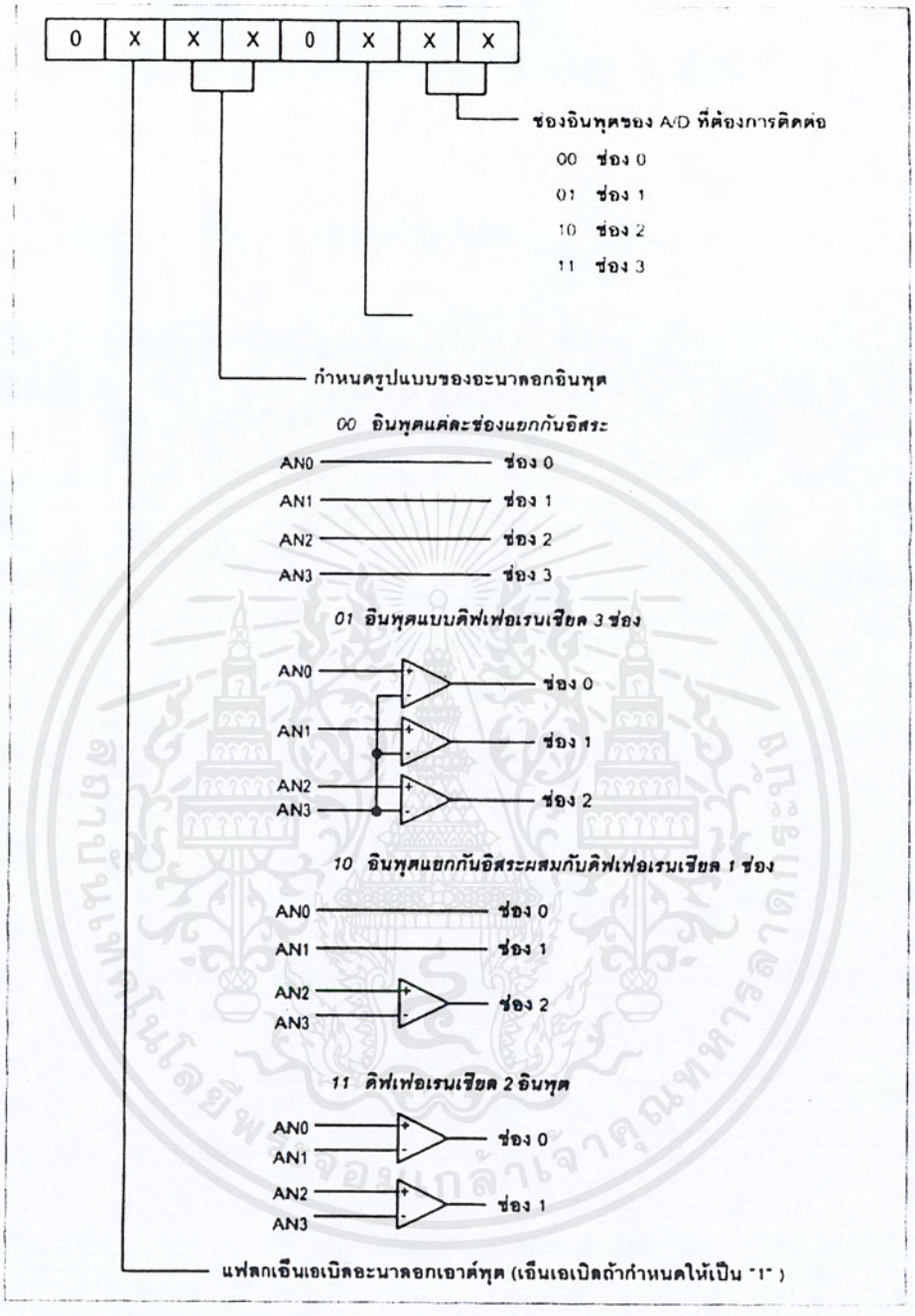
บิต 0 และบิต 1 ใช้สำหรับกำหนดช่องของอินพุตอะนาลอกที่ต้องการอ่าน ถ้ากำหนดให้บิต 2 เป็น “1” หลังจากอ่านค่าของบิต 0 และบิต 1 แล้ว ในการอ่านครั้งต่อไปจะเป็นการอ่านค่า อินพุตจากช่องที่ 1

ข้อมูลทั้งหมดจะถูกจัดเก็บที่รีจิสเตอร์ควบคุมภายใน PCF8591

เมื่อจ่ายไฟให้แก่ PCF8591 ครั้งแรกบิตต่าง ๆ ข้อมูลภายในรีจิสเตอร์ควบคุมจะถูกกำหนดให้เป็น “0”

3.15.2.3 ออสซิเลเตอร์

วงจรออสซิเลเตอร์ภายใน PCF8591 จะสร้างสัญญาณนาฬิกาสำหรับแปลงสัญญาณ อะนาลอกเป็นดิจิตอล เมื่อต้องการใช้วงจรออสซิเลเตอร์ภายใน ขา EXT ต้องต่อลงกราวด์ ถ้าต้องการใช้ออสซิเลเตอร์จากภายนอก ขา EXT ต้องต่อเข้ากับไฟบวกแล้วป้อนสัญญาณนาฬิกาเข้าที่ขา OSC ของ PCF8591 โดยความถี่ของสัญญาณนาฬิกาสูงสุดที่ป้อนให้กับออสซิเลเตอร์เท่ากับ 1.25 MHz

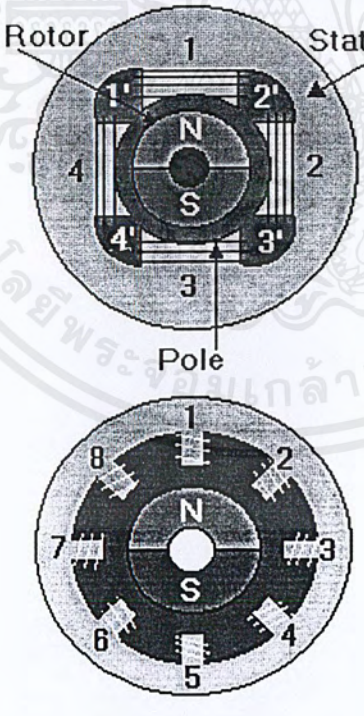


รูปที่ 3-15 รายละเอียดข้อมูลควบคุมการทำงานของ PCF8591

บทที่ 4
หลักการพื้นฐานของสเต็ปมอเตอร์

สเต็ปมอเตอร์เป็นมอเตอร์ที่มีลักษณะการป้อนไฟฟ้าให้กับมอเตอร์ทำให้หมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งต่างจากมอเตอร์ทั่วไปที่จะหมุนทันทีและตลอดเวลาเมื่อป้อนแรงดันไฟฟ้าข้อดีของสเต็ปมอเตอร์คือสามารถกำหนดตำแหน่งของการหมุนด้วยตัวเลข (องศาหรือระยะทาง) ได้อย่างละเอียดโดยใช้คอมพิวเตอร์หรือไมโครคอนโทรลเลอร์เป็นเครื่องกำหนดและจัดเก็บตัวเลข

โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ทำมาจากแผ่นเหล็กวงแหวนที่มีซี่ยื่นออกมาประกอบกันเป็นชั้นๆ โดยที่แต่ละซี่นั้นจะมีคอยล์ (ขดลวด) พันสวมอยู่เมื่อมีการป้อนกระแสผ่านคอยล์ทำให้เกิดสนามแม่เหล็กไฟฟ้า (Electromagnetic) ดังแสดงในรูปที่ 4-1 เป็นการแสดงถึงองค์ประกอบที่กล่าวมา



รูปที่ 4-1 โครงสร้างภายในสเต็ปมอเตอร์

ในที่นี้ถ้ามีการเพิ่มจำนวนของขั้วแม่เหล็กมากขึ้นจะเป็นการเพิ่มจำนวนของสเต็ปต่อวงจรรอบมากขึ้นตามด้วย พิจารณาตามรูปที่ 4-1 ลักษณะการนำไปใช้งานสเต็ปมอเตอร์จะนำไปใช้งาน

ในลักษณะระบบเปิด (Open Loop System) คือสเต็ปมอเตอร์สามารถทำงานได้โดยไม่ต้องมีการป้อนค่าพารามิเตอร์กลับมา (Feedback) แต่ทุกวิธีที่ต้องการกำหนดตำแหน่งที่แน่นอนจะต้องการป้อนกลับไปยังระบบและตัวบอกตำแหน่งว่าถูกต้องหรือผิดพลาดให้รับทราบ ดังเช่นวิธีที่ใช้กับสเต็ปมอเตอร์ คือเรานำลิมิทสวิตช์ ติดตามตำแหน่งที่จะตรวจจับ เมื่อสเต็ปมอเตอร์เริ่มหมุนแล้วหมุนไปจนถึงตำแหน่งของสวิตช์ตรวจจับสัญญาณ สวิตช์ทำงานก็จะป้อนกลับไปสู่ระบบ ซึ่งก็จะทำให้รู้การทำงานของสเต็ปมอเตอร์ตลอด ตัววงจรไมโครคอนโทรลเลอร์เองจะมีจุดอ้างอิงไว้ให้เริ่มต้นการทำงานและอ้างอิงตำแหน่งได้ถูกต้อง

โดยแนวทางสเต็ปมอเตอร์เป็นอุปกรณ์จำพวกเชิงกลทางไฟฟ้า โดยมีกรุ๊ปของไบนารี่โวลท์เตทเป็นอินพุตและการเคลื่อนที่แบบเชิงมุมเป็นเอาท์พุต หรือว่าหมุนทีละสเต็ปซึ่งอยู่ระหว่าง 0.1 - 30 องศา อยู่ที่โครงสร้างของสเต็ปมอเตอร์ โดยตามสัญญาณพัลซ์ที่จ่ายให้กับขดสเตเตอร์ทำให้เกิดแรงผลักแก่โรเตอร์หมุนไป สเต็ปมอเตอร์มีขดลวดหลายชุดในที่นี้เราเรียกว่า เฟส (Phase) ดังนั้นสัญญาณที่ต่อเนื่องเป็น ซีเควน (Sequence) ลักษณะของไบนารี่ (Binary) ซึ่งจะต้องไปผ่านวงจรไดรเวอร์ (Driver) ก็จะทำให้โรเตอร์หมุนไปอย่างต่อเนื่อง โดยแสดงดังรูปที่ 4-2 (ก)

เมื่อพิจารณาจากรูปที่ 4-2 (ข) จะเห็นว่าการพันมีด้วยกัน 2 วิธี คือ แบบไบโพล่าร์ (Bipolar) กับแบบยูนิโพล่าร์ (Unipolar)

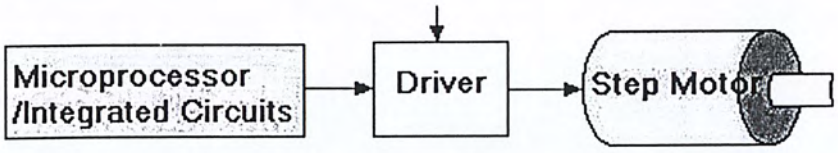
แบบไบโพล่าร์

จะมีการพันขดลวดหนึ่งขด (จะกี่รอบก็แล้วแต่รูปแบบการใช้งาน) ในแต่ขั้วแม่เหล็กของสเตเตอร์โดยขั้วแม่เหล็กที่เกิดขึ้นที่สเตเตอร์จะถูกกำหนดโดยทิศทางของการไหลของกระแสไฟฟ้า ซึ่งสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงกันข้ามได้เพียงการกลับทิศทางของการไหลกระแสไฟฟ้า โดยมาจากการควบคุมของวงจรสวิทช์ชิ่งให้กลับขั้วไฟฟ้า

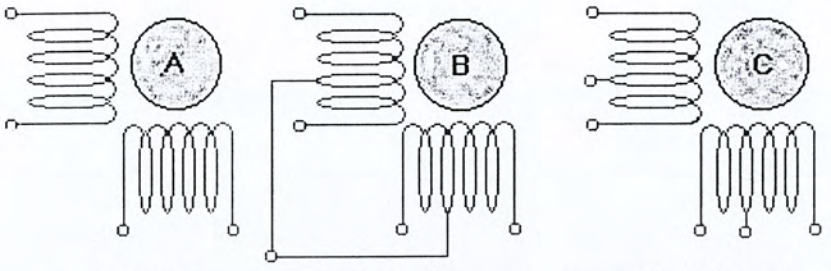
แบบยูนิโพล่าร์

แบบนี้มี 2 ขด บนแต่ละขั้วแม่เหล็กของสเตเตอร์ ทำให้แต่ละขดลวดเกิดขั้วแม่เหล็กในทิศทางตรงกันข้าม ส่วนการกลับทิศทางขั้วแม่เหล็กสามารถทำได้โดยใช้วงจรสวิทช์ชิ่งให้สลับขั่วหนึ่งไปยังอีกขั้วหนึ่งแทนกัน

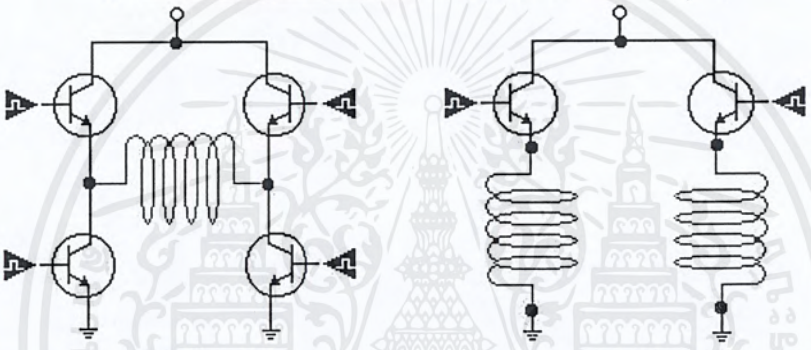
พื้นฐานการสวิทช์ชิ่งดังแสดงในรูปที่ 4-2 (ค) แบบยูนิโพล่าร์จะทำให้เกิดแรงบิดน้อยกว่าแบบไบโพล่าร์ โดยทั้งสองแบบมีข้อแตกต่างกันที่สายไฟที่ต่อมาจากตัวสเต็ปมอเตอร์ซึ่งแบบไบโพลาร์จะมี 4 สาย ส่วนเป็นแบบยูนิโพล่าร์จะมี 5 สายหรือ 6 สาย



ง. การควบคุมระบบสเต็ปมอเตอร์



จ. การพันขดลวดบนสเตเตอร์ของสเต็ปมอเตอร์



คือ ต่อเข้ากับแหล่งจ่ายไฟหรือจากพอร์ตพีซีเพื่อทริกทรานซิสเตอร์ให้ทำงาน

ก. วงจรการจ่ายไฟให้กับสเต็ปมอเตอร์

รูปที่ 4-2 รูปประกอบการทำงานของสเต็ปมอเตอร์

4.1 การสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์

การควบคุมและสั่งงานให้สเต็ปมอเตอร์ทำงานไปทีละสเต็ป สามารถทำได้โดยการจ่ายกำลังไฟไปยังขดลวด ในแต่ละเส้นบนสเตเตอร์ โดยการป้อนจะทำในลักษณะเป็นลำดับหรือเรียกว่า ซีเควนเวียลในลูปที่ถูกต้อง ซึ่งจะแบ่งได้เป็น 3 รูปแบบ คือ แบบเวฟ (Wave) แบบ 2 เฟส (2 Phase) และแบบครึ่งสเต็ป (Half Step) ทั้ง 3 แบบนี้ก็จะมีข้อดีและข้อเสียต่างกันออกไป ดังนี้

4.1.1 แบบเวฟ (Wave)

จะเป็นการกระตุ้นแบบที่ง่ายที่สุด ซึ่งจะทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่ง ๆ เรียงกันไป ตัวอย่างเช่น ขดที่ 1, 2, 3, 4, 1, 2, 3, และ 4 เป็นลำดับอย่างนี้ หรือ ขด 1, 4, 3, 2, 1, 4, 3, และ 2 เป็นลำดับกันไป ทั้งนี้ขึ้นอยู่กับทิศทางที่เราต้องให้มอเตอร์หมุนไป วงจรที่นำมากระตุ้นนั้นจะมี

ราคาค่อนข้างจะถูกกว่าและง่ายกว่า ดังแสดงในรูปที่ 4-2 (ค) สามารถเขียนขั้นตอนการทำงานเป็นตารางออกมาได้ดังตารางที่ 4-1

ตารางที่ 4-1 การกระตุ้นสเต็ปมอเตอร์แบบเวฟ

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	NO			
2		NO		
3			NO	
4				NO
5	NO			
6		NO		

4.1.2 แบบ 2 เฟส (2 Phase)

แบบนี้ก็จะคล้ายกับการกระตุ้นในแบบเวฟแต่จะต่างกันตรงที่ แบบ 2 เฟส จะกระตุ้นทีละ 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกันและจะเรียงลำดับกันไป ดังเช่นแบบเดียวกับแบบเวฟ ดังตัวอย่างการกระตุ้นขดลวดในลักษณะซีเควนดังนี้ 12, 23, 34, 41, 12, 23, 34, 41 เรียงลำดับกันไปเรื่อยๆ หรือจะเป็น 14, 43, 32, 21, 14, 43, 32, 21 เรียงกันไปเรื่อย ๆ เช่นกัน สำหรับข้อดีข้อเสียของแบบ 2 เฟส มีดังนี้

ข้อดี การเพิ่มจำนวนขดลวดที่ถูกกระตุ้นจะทำให้แรงบิดได้มากกว่าแบบเวฟ ซึ่งโรเตอร์จะหมุนด้วยแรงดึงแบบเต็ม ๆ แรงจากทั้ง 2 ขดลวดที่กระตุ้นพร้อมกัน

ข้อเสีย แบบ 2 เฟส จะกระตุ้นขดลวดนั้นต้องใช้กำลังไฟมากขึ้นเป็น 2 เท่าของแบบเวฟ สำหรับลำดับการกระตุ้นของขดลวดแบบ 2 เฟส ดังแสดงในตารางที่ 4-2

ตารางที่ 4-2 การกระตุ้นสเต็ปมอเตอร์แบบ 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	NO	NO		
2		NO	NO	
3			NO	NO
4	NO			NO
5	NO	NO		
6		NO	NO	

4.1.3 แบบครึ่งสเต็ป (Half Step)

แบบนี้เป็นรูปแบบการผสมผสานของการกระตุ้นระหว่างแบบเวฟกับแบบ 2 เฟส เพื่อทำให้จำนวนรอบของสเต็ปมากขึ้นเป็น 2 เท่า ซึ่งในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเรื่อย ๆ เป็นลำดับ ดังตัวอย่างต่อไปนี้ 1, 12, 2, 23, 3, 34, 4, 41, 1, 12, 2, 23, 3, 34, 4, 41, และ1 เป็นลำดับอยู่อย่างนี้เรื่อยไป ถ้าทำการกลับทิศทางการหมุนก็จะได้ 1, 41, 4, 43, 3, 32, 2, 21, 1, 41, 4, 43 3, 32, 2, 21, และ1 เป็นลำดับกันไป ส่วนข้อดีและข้อเสียของการกระตุ้นแบบครึ่งสเต็ปคือ

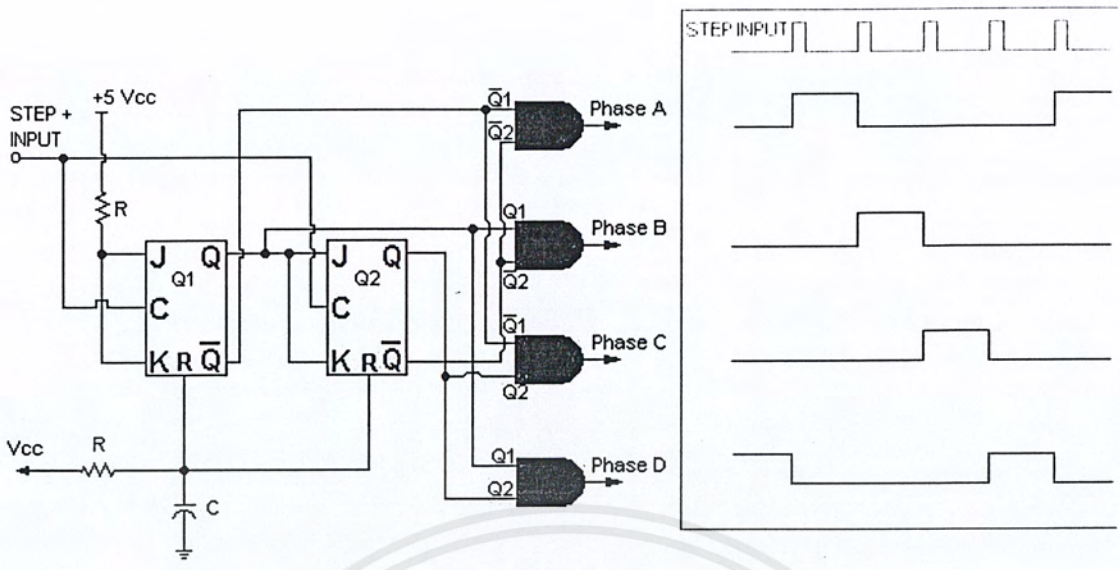
ข้อดี การกระตุ้นแบบนี้จะให้แรงบิดที่เพิ่มมากขึ้น เนื่องจากช่วงสเต็ปที่มีระยะสั้นลงอีกประการหนึ่งแต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกันเป็นผลให้ค่าตำแหน่งความถูกต้องมากขึ้นไปด้วย

ข้อเสีย เช่นเดียวกับแบบ 2 เฟส ที่ต้องจ่ายกำลังไฟเป็น 2 เท่าของแบบเวฟหรือจะใช้เท่ากับแบบ 2 เฟสนั้นเอง ดังนั้นสามารถแสดงลำดับการทำงานของแบบครึ่งเฟส ได้ดังตารางที่ 4-3

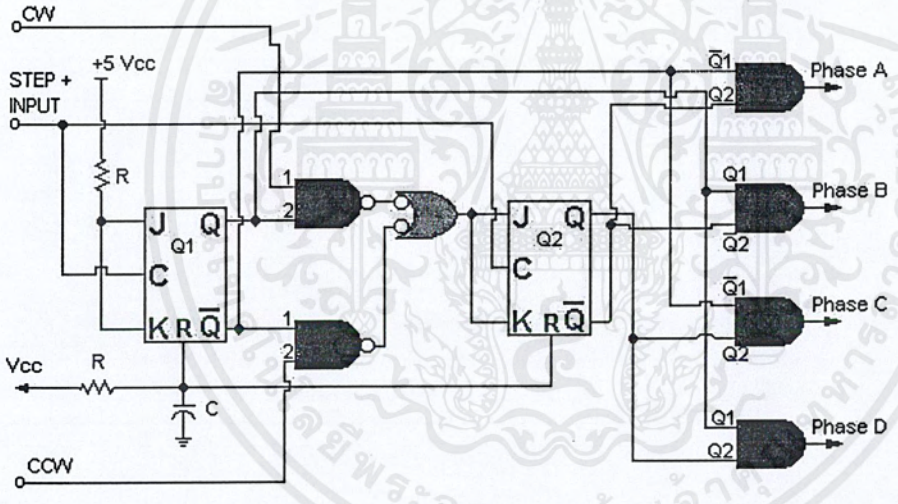
ตารางที่ 4-3 การกระตุ้นสเต็ปมอเตอร์แบบครึ่งเฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	NO			
2	NO	NO		
3		NO		
4		NO	NO	
5			NO	
6			NO	NO
7				NO
8	NO			NO
9	NO			
10	NO	NO		

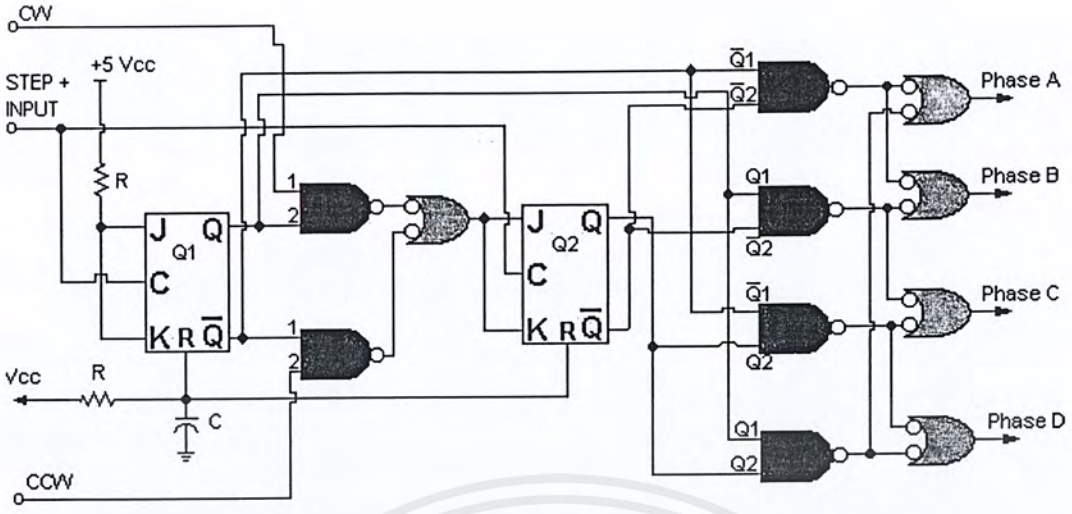
ทั้งหมดนี้ก็เป็นพื้นฐานของความรู้ด้านสเต็ปมอเตอร์เพื่อที่จะสามารถนำไปประกอบกับการใช้ทำโครงงานต่าง ๆ ในงานอินเทอร์เฟสโดยการเขียนโปรแกรมไปควบคุมสเต็ปมอเตอร์ได้ต่อไป



รูปที่ 4-3 วงจรสร้างสเต็ปลำดับแบบ 4 เฟสแบบเวฟ



รูปที่ 4-4 วงจรสร้างสเต็ปลำดับแบบแบบ 4 เฟสแบบบริดจ์



รูปที่ 4-5 วงจรสร้างสเต็ปลำดับ 2 เฟสเป็น 4 เฟส แบบลอจิก

4.2 การขับสเต็ปมอเตอร์ด้วยพอร์ตขนาน

สเต็ปมอเตอร์ เป็นมอเตอร์ที่มีลักษณะการทำงานแตกต่างจากมอเตอร์ทั่วไป เพราะจะต้องป้อนสัญญาณเป็นพัลส์ให้แก่ขดลวดของมอเตอร์เป็นจังหวะอย่างเหมาะสม และการหมุนของมอเตอร์ชนิดนี้จะหมุนเป็นจังหวะตามพัลส์ที่ป้อนเข้ามา ไม่หมุนต่อเนื่องเหมือนกับมอเตอร์ธรรมดา ทำให้ผู้ควบคุมสามารถเลือกตำแหน่งที่ต้องการให้มอเตอร์หยุดหมุนได้ จังหวะการหมุนของสเต็ปมอเตอร์เรียกว่า สเต็ป (step) ความละเอียดของมอเตอร์กำหนดเป็นองศาที่หมุนไปในหนึ่งสเต็ป หากมอเตอร์มีจำนวนองศาต่อสเต็ปมาก หมายความว่า มอเตอร์ตัวนี้มีความละเอียดของการหมุนต่ำ ยกตัวอย่าง การหมุนครบ 1 รอบเท่ากับ 360 องศา หากมอเตอร์มีสเต็ปการหมุนเท่ากับ 7.5 องศาต่อสเต็ป มอเตอร์ตัวนี้มีความละเอียดของการหมุนเท่ากับ 48 ตำแหน่ง แต่ถ้าหากมีสเต็ปการหมุนกับ 1.8 องศาต่อสเต็ป ความละเอียดของการหมุนเท่ากับ 200 จะเห็นได้ว่ามอเตอร์ตัวหลังมีความละเอียดสูงกว่าตัวแรกมาก ทำให้นำมาใช้ในงานที่ต้องการกำหนดตำแหน่งได้ดีกว่าแม่นยำกว่า ผนวกเข้ากับวงจรขับแบบครึ่งสเต็ป ความละเอียดของการหมุนจะเพิ่มขึ้นอีก 2 เท่า ทำให้มีความละเอียดของการหมุนกลายเป็น 400 ตำแหน่ง

ขนาดของสเต็ปมอเตอร์ที่มีการผลิตและจำหน่ายในท้องตลาด มีตั้งแต่ขนาดแรงดันต่ำ 3 V ไปจนถึง 24 V ส่วนขนาดของกระแสมีตั้งแต่ไม่กี่สิบมิลลิแอมป์อันเป็นสต์ปเปอร์มอเตอร์ตัวเล็กไปจนถึงเป็นสิบแอมป์ ซึ่งแน่นอนขนาดของมอเตอร์ย่อมต้องใหญ่โตขึ้นตามลำดับ ราคาอยู่ในหลักเป็นร้อยบาทขึ้นไปสำหรับของใหม่

4.3 สเต็ปมอเตอร์แบบยูนิโพล่าร์

สเต็ปมอเตอร์ได้รับการพัฒนามาอย่างต่อเนื่อง จนในปัจจุบันสเต็ปเปอร์มอเตอร์ที่นิยมใช้อย่างแพร่หลายมากที่สุดและหาได้ง่ายคือ สเต็ปมอเตอร์แบบยูนิโพล่าร์ มีลักษณะการพันขดลวดของมอเตอร์ดังแสดงในรูปที่ 4-4

สเต็ปมอเตอร์แบบนี้มีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ แต่ละขดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวจะมี 4 เฟส คือเฟส 1, 2, 3 และ 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเต็ปมอเตอร์แบบนี้มีทั้งแบบ 5 สาย และ 6 สาย ถ้าเป็นแบบ 5 สาย จะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสองมาต่อรวมกันเป็นสายเดียว

4.4 การกระตุ้นเพื่อขับสเต็ปมอเตอร์

สามารถทำได้โดยการจ่ายพลังงานไฟฟ้าไปยังขดลวดแต่ละขดบนแกนแม่เหล็กคงที่ ซึ่งต้องป้อนเป็นลำดับตามรูปแบบที่ถูกต้อง โดยสามารถแบ่งได้ 3 รูปแบบ คือ ฟูลสเต็ป 1 เฟส (Full-step 1 Phase) ฟูลสเต็ป 2 เฟส (Full-step 2 Phase) และฮาล์ฟสเต็ป (Half Step)

4.4.1 แบบฟูลสเต็ป 1 เฟส

เป็นการกระตุ้นที่ง่ายที่สุด โดยกระตุ้นขดลวดทีละขดไล่เรียงกันไป เช่น เริ่มต้นที่ขดที่ 1, 2, 3 และ 4 แล้ววนกลับมาขดที่ 1 หรือเริ่มที่ขดที่ 1 แล้วย้อนไปยังขดที่ 4, 3 และ 2 แล้วกลับมายังขดที่ 1 อีกครั้ง ทำให้ทิศทางการหมุนสวนกัน ในการกระตุ้นแบบนี้จึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น การกระตุ้นแบบนี้มีราคาถูกและง่าย สรุปขั้นตอนการทำงานแสดงดังในตารางที่ 4-4

ตารางที่ 4 -4 รูปแบบการขับสเต็ปมอเตอร์

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน			
2		ทำงาน		
3			ทำงาน	
4				ทำงาน

4.4.2 แบบฟูลสเต็ป 2 เฟส

จะกระตุ้นโดยจ่ายแรงดันไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกันและเรียงถัดกันไป ขดลวดชุดแรกที่ถูกกระตุ้นจะเป็นขดที่ 1 และ 2 ตามด้วยการกระตุ้นขดที่ 2 และ 3 ต่อไปเป็นขดที่ 3 และ 4 ถัดไปเป็นขดที่ 4 และ 1 แล้วกลับมาที่ขดที่ 1 และ 2 วนไปตามลำดับเช่นนี้ หรือเริ่มที่ขด 1 และ 4 ตามด้วยขดที่ 4 และ 3 ถัดไปเป็นขดที่ 3 และ 2 ต่อไปเป็นขดที่ 2 และ 1 แล้ววนกลับมาที่ขดที่ 1 และ 4 ทิศทางการหมุนจะสวนกัน ดังแสดงขั้นตอนการกระตุ้นใน
ตารางที่ 4-5 การกระตุ้นแบบนี้สามารถเพิ่มแรงบิดได้มากกว่าแบบ 1 เฟส แกนแม่เหล็กเคลื่อนที่ภายในมอเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน และหมุนต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือ ต้องใช้กำลังไฟฟ้ามากขึ้น

ตารางที่ 4-5 รูปแบบการขับสเต็ปมอเตอร์แบบฟูลสเต็ป 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน		
2		ทำงาน	ทำงาน	
3			ทำงาน	ทำงาน
4	ทำงาน			ทำงาน

4.4.3 แบบฮาล์ฟสเต็ป

เป็นรูปแบบที่ผสมผสานระหว่างการกระตุ้นแบบที่ 1 และแบบที่ 2 เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ เริ่มจากขดลวดที่ 1, 1 และ 2, 2, 2 และ 3, 3, 3 และ 4, 4, 4 และ 1 แล้ววนกลับมายังขดลวดที่ 1 ดังแสดงในตารางที่ 4-6 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้น เนื่องจากช่วงของการเคลื่อนที่ในแต่ละสเต็ปมีระยะสั้นลง แต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังว่า เมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องหมุนถึง 2 สเต็ป จึงจะได้เท่ากับระยะเท่ากับการกระตุ้นแบบฟูลสเต็ป 1 สเต็ป สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้ขนาดเท่ากับแบบ 2 เฟสเป็นอย่างน้อยจึงจะเพียงพอ

ตารางที่ 4-6 รูปแบบการขับสเต็ปมอเตอร์แบบฮาล์ฟสเต็ป

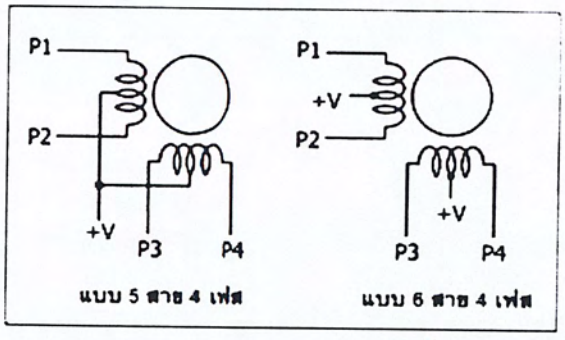
สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน			
2	ทำงาน	ทำงาน		
3		ทำงาน		
4		ทำงาน	ทำงาน	
5			ทำงาน	
6			ทำงาน	ทำงาน
7				ทำงาน
8	ทำงาน			ทำงาน

4.5 วงจรขับสเต็ปมอเตอร์

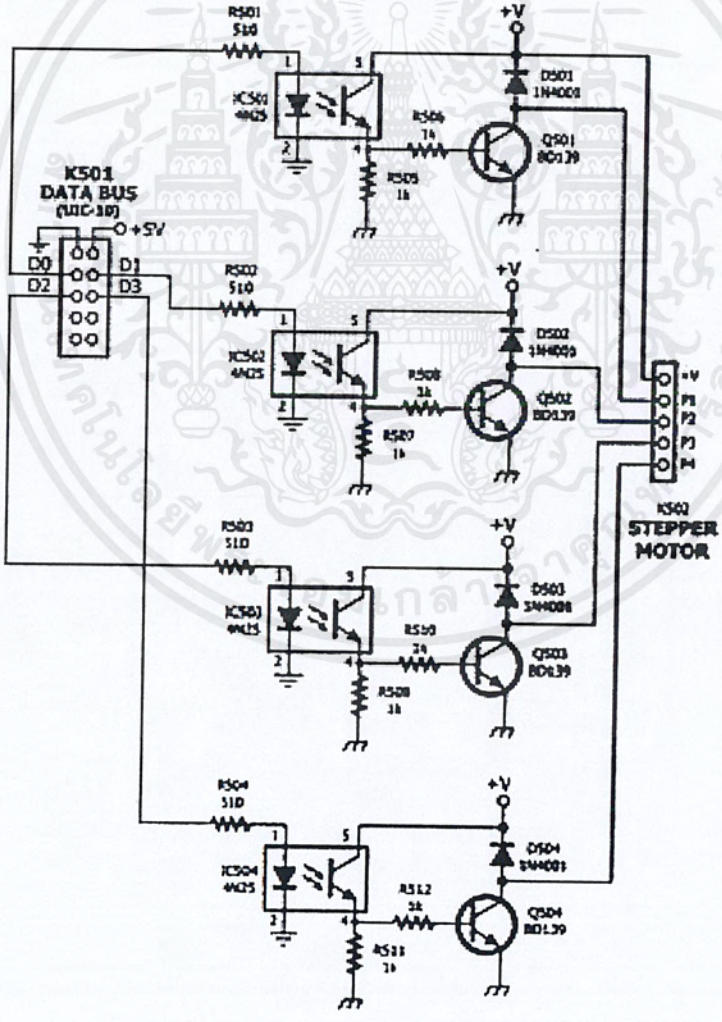
วงจรขับสเต็ปมอเตอร์แบบยูนิโพล่าร์ เพื่อขับสเต็ปเปอร์มอเตอร์ผ่านทางพอร์ตขนานของคอมพิวเตอร์โดยบอร์ดมีการแยกกราวน์ดทางไฟฟ้าของส่วนอินพุตที่เชื่อมต่อกับพอร์ตขนานของคอมพิวเตอร์และเอาต์พุตที่เชื่อมต่อกับสเต็ปเปอร์มอเตอร์ออกจากกัน แล้วใช้การเชื่อมต่อทางแสงโดยอุปกรณ์ที่เรียกว่า ออปโต้คัปเปอร์ในการถ่ายทอดสัญญาณควบคุม วงจรของบอร์ดแสดงในรูปที่ 4-5 สัญญาณอินพุตมาจากคอนเน็กเตอร์ DATA BUS ซึ่งเชื่อมต่อเข้ากับบอร์ดพอร์ตขนาน แล้วส่งไปขับ LED อินฟราเรดซึ่งอยู่ภายในออปโต้คัปเปอร์ IC501–IC504 โดยผ่านตัวต้านทาน R501-R504 เพื่อจำกัดกระแสให้กับ LED ในขณะที่เอาต์พุตของออปโต้คัปเปอร์จะต่อเข้ากับทรานซิสเตอร์ Q501-Q504 เบอร์ BD139 เพื่อขับมอเตอร์ต่อไป เมื่อป้อนลอจิก “1” ให้กับออปโต้คัปเปอร์จะทำให้ LED ภายในออปโต้คัปเปอร์ทำงาน กระตุ้นให้โฟโต้ทรานซิสเตอร์ภายในออปโต้คัปเปอร์ทำงาน เกิดแรงดันผ่านตัวต้านทานป้อนเข้าที่ขาเบสของทรานซิสเตอร์ ทำให้ทรานซิสเตอร์ BD139 นำกระแสเสมือนว่าขดลวดด้านหนึ่งถูกต่อลงกราวน์ดจึงเกิดกระแสไหลผ่านขดลวด ทำให้เกิดการหมุนขึ้น การหมุนของสเต็ปเปอร์มอเตอร์จะเป็นอย่างไร ขึ้นอยู่กับข้อมูลที่ป้อนให้แก่วงจรขับบนบอร์ดนี้

สำหรับไฟเลี้ยงของสเต็ปเปอร์มอเตอร์จะแยกออกจากวงจรอินพุตที่เชื่อมต่อกับบอร์ดพอร์ตขนาน โดยต้องป้อนแรงดันตามพิกัดของมอเตอร์ที่ต้องการขับ แต่มีข้อจำกัดว่าต้องมีค่าไม่เกิน 30V กระแสไฟฟ้าประมาณ 1A

ถ้าผู้ใช้งานต้องการขับสเต็ปมอเตอร์หลาย ๆ ตัว ในคราวเดียวก็สามารถทำได้โดยนำขาพอร์ต Data ทั้ง 8 บิตมาใช้ทำให้สามารถขับสเต็ปมอเตอร์ได้ 2 ตัว และถ้าใช้พอร์ต Control มาช่วยก็จะสามารถขับสเต็ปมอเตอร์ได้ในคราวเดียวถึง 3 ตัว



รูปที่ 4-6 โครงสร้างพื้นฐานสเต็ปมอเตอร์ชนิดยูนิโพล่าร์ทั้ง 5 และ 6 สาย



รูปที่ 4-7 วงจรขับสเต็ปมอเตอร์

บทที่ 5
การใช้งานโปรแกรม Visual Basic

Visual Basic เป็นเครื่องมือที่ช่วยพัฒนาแอพพลิเคชั่นสำหรับวินโดว์ตัวแรกที่ประสบความสำเร็จเป็นอย่างมาก ทั้งนี้เนื่องจากแนวความคิดที่จะนำเอาความสามารถของคอนโทรลมาใช้ในการออกแบบโปรแกรมนี้เอง เพราะคอนโทรลเป็นเครื่องมือที่ช่วยลดความซับซ้อนในการเขียนโค้ดลงไปได้มาก และคอนโทรลยังมีส่วนที่แสดงผลเพื่อสื่อความหมายของการทำงานระหว่างคอนโทรลและผู้เขียน และนอกจากนี้คอนโทรลยังมีส่วนที่แสดงผลเพื่อสื่อความหมายของการทำงานระหว่างคอนโทรลและผู้ใช้ได้อีกด้วย ส่วนการทำงานก็ไม่มีความซับซ้อนเพียงแต่ผู้ใช้ทำการเชื่อมต่อคอนโทรลเข้ากับสภาพแวดล้อมของ Visual Basic จากนั้นก็สามารถที่จะนำมาเพิ่มลงในฟอร์มได้ทันที สำหรับ Visual Basic ได้มีการแบ่งคอนโทรลออกเป็น 4 ส่วนหลัก ดังนี้

1. **คอนโทรลภายใน (Intrinsic Control)** เช่น ComboBox, commandBox, หรือ PictureBox เป็นต้น ซึ่งเป็นคอนโทรลที่ถูกสร้างลงในสภาพแวดล้อมของ vb.exe ดังนั้นทุกครั้งที่ผู้ใช้โหลด Visual Basic คอนโทรลเหล่านี้จะออกจากกล่องเครื่องมือได้เลยดังนั้นจึงได้ว่าเป็นคอนโทรล มาตรฐาน (Stand Control) กลุ่มหนึ่งของ Visual Basic

2. **คอนโทรลมาตรฐาน (Standard Control)** เป็นคอนโทรล ActiveX ที่ถูกสร้างเป็นไฟล์ .ocx ที่แยกออกมาต่างหาก เช่น Dbgrid (Apex data-bound grid) MSFlexGrid หรือ CommonDialog เป็นต้น ดังนั้นก่อนที่จะสามารถใช้งานคอนโทรลในกลุ่มนี้ได้ เราต้องทำการเชื่อมต่อไฟล์ .ocx เหล่านี้เข้ากับสภาแวดล้อมของ Visual Basic เสียก่อน โดยใช้คำสั่ง Components ในเมนู Project เช่นเดียวกัน

3. **คอนโทรลร่วมวินโดว์ (Windows Common Control)** เป็นคอนโทรล ActiveX ที่ถูกสร้างเป็นไฟล์ .ocx ที่ต้องใช้ร่วมกับไฟล์ .dll ของวินโดว์ เช่น Rich TextBox ,Slider หรือ Statusbar เป็นต้น เช่นเดียวกับคอนโทรลมาตรฐาน เพียงแต่คอนโทรล ในกลุ่มนี้ได้ถูกจัดเป็นไฟล์พื้นฐานของวินโดว์ 95 โดยที่คอนโทรลร่วมกับวินโดว์จะถูกจัดเก็บลงในไฟล์ conctl32.ocx และ conct232.ocx

4. **คอนโทรล ActiveX รุ่นมืออาชีพ** (Professional ActiveX Control) เป็นคอนโทรล ActiveX ที่ถูกสร้างเป็นไฟล์ .ocx เช่นเดียวกับคอนโทรลมาตรฐาน เช่น MSComm (Communication) , MapiMessage (MAPI message) หรือ MMControl (Multimedia MCI) เป็นต้น แต่คอนโทรลกลุ่มนี้ได้ถูกสร้างและแจกจ่ายมากับ Visual Basic รุ่น Professional และ Enterprise เท่านั้น

5.1 คุณสมบัติแสดงค่าของคอนโทรลที่สำคัญ

คอนโทรลทั้งหมดที่มากับ Visual Basic ไม่ว่าจะเป็นคอนโทรลภายในหรือ ActiveX จะมีคุณสมบัติตัวหนึ่งที่ถูกใช้สำหรับการกำหนดค่า (Value) หรืออ่านจากค่าคอนโทรลและคุณสมบัตินี้ได้ถูกกำหนดให้เป็นคุณสมบัติปกติ (Default) ของคอนโทรล โดยในการเขียนโค้ดเราสามารถใช้เพียงชื่อของคอนโทรล (Control Name) โดยไม่ต้องกำหนดคุณสมบัติปกติของทุก ๆ คอนโทรลได้โดยไม่เกิดข้อผิดพลาด

5.2 การแบ่งกลุ่มของคอนโทรลภายใน

เราสามารถแยกตามวัตถุประสงค์ของการใช้งานได้ทั้งหมด 4 กลุ่ม

1. คอนโทรลภายในทั่วไป ประกอบด้วยคอนโทรลที่แสดงผลในลักษณะของการเลือกตอบหรือเลือกรายการ เช่น CheckBox ,OptionBox หรือ ListBox เป็นต้น

2. คอนโทรลภายในด้านระบบไฟล์ ประกอบด้วยคอนโทรลที่ทำหน้าที่ติดต่อ หรือแสดงผลระบบไฟล์ (รวมทั้งไดรฟ์ และไดเร็กทรอรี่ด้วย) ของวินโดว์ เช่น FileListBox หรือ DirListBox

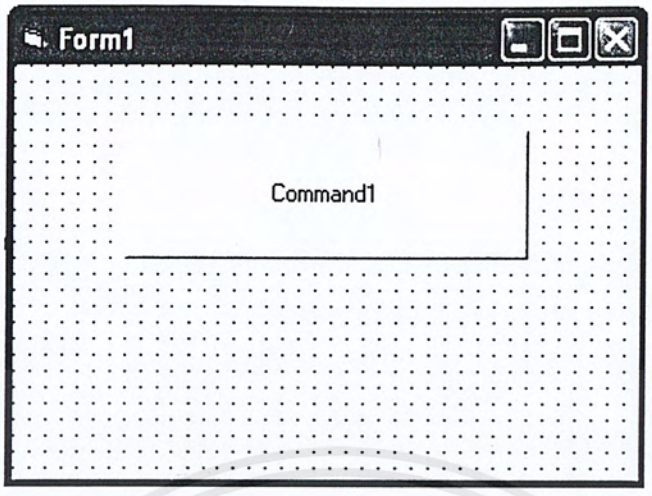
3. คอนโทรลภายในด้านกราฟิก ประกอบด้วยคอนโทรลที่ทำหน้าที่ด้านการแสดงผลกราฟิกด้วยวิธีกราฟิกคอนโทรล หรือฟังก์ชั่นวินโดว์ API หรือ ไฟล์กราฟิกในรูปแบบต่าง ๆ เช่น PictureBox,Shape, และ Image เป็นต้น

4. คอนโทรลภายในด้านเวลา ซึ่งจะมีคอนโทรลเดียวได้แก่ Timer ซึ่งมีหน้าที่สร้างเหตุการณ์ที่ตอบสนองเป็นครั้ง ๆ ตามช่วงเวลาที่ถูกกำหนด

5.2.1 คอนโทรลภายใน

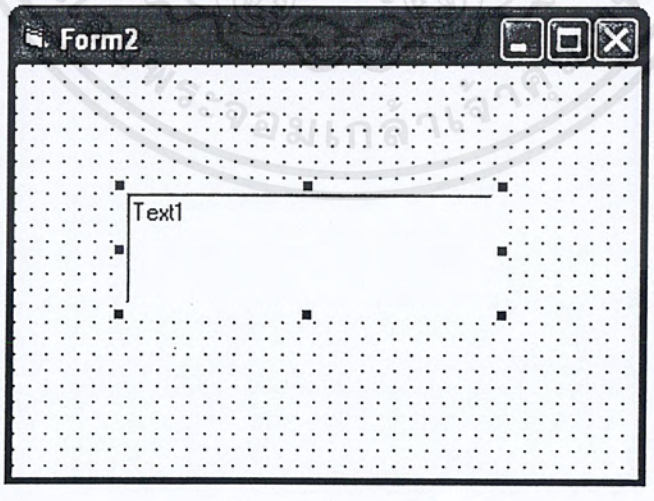
คอนโทรลภายในเป็นคอนโทรลพื้นฐานที่ถูกนำไปใช้งานมากที่สุด เพราะจะเป็นกลุ่มของคอนโทรลที่ช่วยในการสื่อสารแบบสองทางหรือรับเงื่อนไขจากผู้ใช้ ดังนั้นทุก ๆ แอพลิเคชั่นจะใช้คอนโทรล CommandButton สำหรับผู้ใช้เลือกที่ยอมรับ (OK) ยกเลิก (Cancle) หรืออื่น ๆ ตามข้อกำหนดของแต่ละแอพลิเคชั่น เป็นต้น ซึ่งคอนโทรลภายในส่วนใหญ่จะประกออบด้วย คอนโทรลต่าง ๆ ต่อไปนี้

1. คอนโทรล CommandButton จะเป็นคอนโทรลที่ถูกนำมาใช้งานมากที่สุด เพราะในการกำหนดให้ผู้ใช้เลือก OK หรือ Cancle นั้นมักจะใช้คอนโทรล CommandButton เป็นส่วนใหญ่ ดังนั้นจึงถือเป็นคอนโทรลที่พื้นฐานที่สุดของ Visual Basic เนื่องจากคอนโทรลนี้เป็นปุ่มสำคัญที่ใช้งานในรูปแบบของการคลิกเพื่อยืนยัน ดังนั้นจึงอาจจะเรียกคอนโทรล CommandButton อีกอย่างหนึ่งได้ว่า PushButton ในขณะออกแบบคอนโทรล CommandButton ที่วางลงบนฟอร์มจะมีลักษณะดังแสดงในรูปที่ 5-1



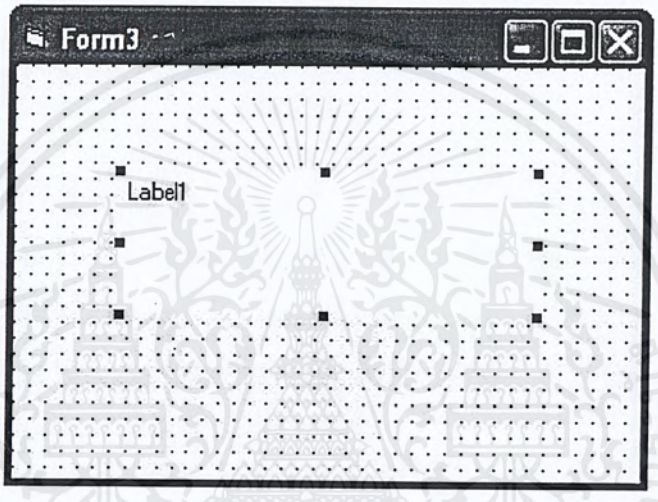
รูปที่ 5-1 คอนโทรล ControlButton ในขณะออกแบบ

2. คอนโทรล TextBox มักจะถูกนำไปใช้ทุก ๆ ฟอร์มที่มีการรับการกรอกข้อมูลจากผู้ใช้ เนื่องจากคอนโทรลนี้มีหน้าที่แสดงข้อมูลในคอนโทรลและยังอนุญาตให้ผู้ใช้สามารถแก้ไขอักษรต่าง ๆ นอกจากนี้แล้วคอนโทรล TextBox ยังได้รวบรวมเอาความสามารถหลาย ๆ ด้านของคอนโทรล Lable มาใช้ เช่น สามารถแสดงข้อความได้มากกว่า 1 บรรทัด ความสามารถด้าน DDE (Dynamic Data Exchange) และนอกจากนี้ยังสามารถนำไปใช้ในลักษณะของการกรอกรหัสผ่าน (Password) ได้อีกด้วย ดังแสดงในรูปที่ 5-2



รูปที่ 5-2 คอนโทรล TextBox

3 คอนโทรล Label เป็นคอนโทรลในลักษณะของกราฟิกที่ถูกใช้งานด้านการแสดงผลข้อความบนฟอร์ม เหมือนกับผู้ใช้ได้นำป้ายข้อความไปวางไว้บนฟอร์ม เพื่อใช้ในการสื่อข้อความกับผู้ใช้ และคอนโทรลนี้ผู้ใช้สามารถแก้ไขได้โดยตรงด้วยวิธีการคีย์หรือใช้เมาส์ในขณะนั้นแอพลิเคชั่น นอกเสียจากในแอพลิเคชั่นจะมีการเขียนโค้ดสำหรับแก้ไขข้อความในคอนโทรลโดยการแก้ไขค่าคุณสมบัติ Caption เท่านั้น และนอกจากนี้ Label ยังเป็นคอนโทรลที่มีความสามารถด้าน DDE อีกด้วย ในขณะออกแบบสามารถเพิ่มคอนโทรลลงในฟอร์ม หรือตัวบรรจุอื่น ๆ ของคอนโทรล ดังแสดงในรูปที่ 5-3

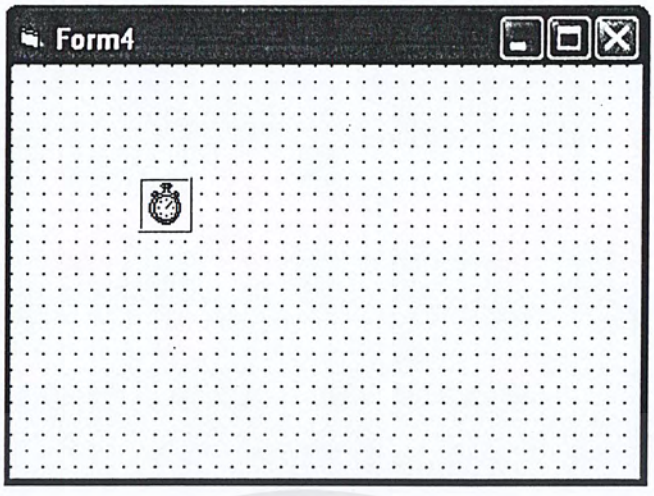


รูปที่ 5-3 คอนโทรล Label

ในการควบคุมพฤติกรรมของคอนโทรลสามารถกระทำได้โดยการกำหนดค่าต่าง ๆ ให้กับคุณสมบัติของคอนโทรล

4. คอนโทรล Timer เป็นคอนโทรลที่ใช้ควบคุมจัดการด้านเวลา ซึ่งสามารถเขียนโค้ดเพื่อทำงานใด ๆ เมื่อช่วงเวลาผ่านไปตามค่าที่กำหนด เช่น ทำการปรับการแสดงผลของฟอร์มทุก ๆ 1 นาที เป็นต้น โดยคอนโทรลนี้จะตอบสนองเหตุการณ์เพียงเหตุการณ์เดียวเท่านั้น แต่เราสามารถกำหนดให้แต่ละฟอร์มมีคอนโทรล Timer มากกว่า 1 คอนโทรล เนื่องจากคอนโทรล Timer เป็นคอนโทรลที่ทำงานตามนาฬิกาของระบบ ดังนั้นมันจึงถูกควบคุมโดยตัวของมันเอง

ในขณะที่ออกแบบคอนโทรล Timer ที่วางลงเป็นฟอร์มให้กับฟอร์มก็จะมีลักษณะดังแสดงในรูปที่ 5-4 และเมื่อรันแอพลิเคชั่น คอนโทรลนี้จะไม่ถูกแสดงผล แต่จะมีการทำให้เกิดเหตุการณ์ Time ทุกครั้งที่ช่วงเวลาครบตามที่กำหนดให้กับคุณสมบัติ Interval ของคอนโทรล Timer



รูปที่ 5-4 คอนโทรล Timer

5.2.2 คอนโทรลด้านฐานข้อมูล

แอพลิเคชั่นที่ใช้กับโครงงานนี้จะมีการเข้าถึงฐานข้อมูล ดังนั้นตัวแปรภาษาที่เหมาะสมกับการสร้างแอพลิเคชั่นเหล่านี้จึงต้องมีเครื่องมือที่สนับสนุนการจัดข้อมูลอย่างง่ายและมีประสิทธิภาพ เราจึงเลือกใช้ Visual Basic เป็นตัวแปรภาษาที่มีการสนับสนุนระบบจัดการฐานข้อมูลในรูปแบบ Microsoft Access โดยอาศัย JET Datbase Engine ซึ่งเป็นเครื่องมือที่ให้โปรแกรมสามารถจัดการเก็บข้อมูลได้

คอนโทรลด้านข้อมูลเป็นคอนโทรลที่ใช้ในการเข้าถึงฐานข้อมูล ซึ่งหลัก ๆ สามารถแบ่งออกได้เป็น 3 กลุ่ม ดังนี้

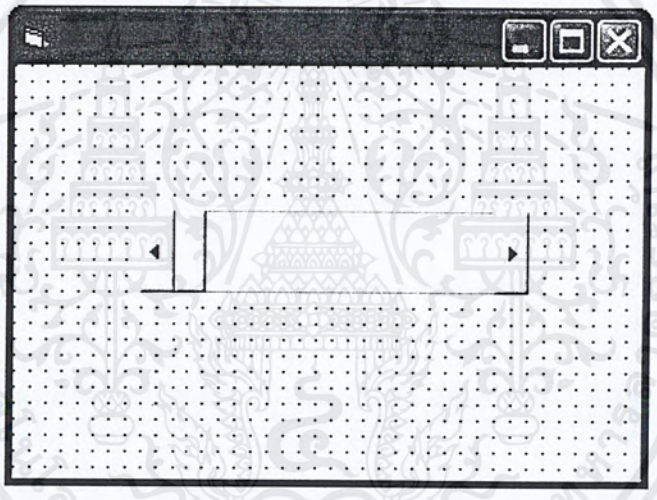
1. คอนโทรล Data เป็นคอนโทรลหลักที่ใช้ในการควบคุมการติดต่อระหว่างคอนโทรลด้านฐานข้อมูลกับฐานข้อมูล โดยที่คอนโทรล Data จะทำหน้าที่ควบคุมการเข้าถึงฐานข้อมูล เช่น การเคลื่อนที่ไปยังเรคคอร์ด การเปิดปิด การเก็บฐานข้อมูล เป็นต้น

2. คอนโทรลภายใน Data-Aware เป็นคอนโทรลภายในของ Visual Basic ที่สนับสนุนคุณสมบัติการแสดงผลข้อมูลของฟิลด์หนึ่งของฐานข้อมูล เช่น CheckBox, PictureBox หรือ TextBox เป็นต้น ซึ่งคอนโทรลภายใน Data-Aware จะแตกต่างกับคอนโทรล Data-Bound ตรงที่คอนโทรลภายใน Data-Aware จะสามารถเชื่อมต่อเข้ากับฟิลด์ของฐานข้อมูลคอนโทรลละ 1 ฟิลด์เท่านั้น

3. คอนโทรล Data-Bound เป็นคอนโทรล ActiveX ที่ถูกออกแบบพิเศษเพื่อให้สามารถเชื่อมต่อกับ Record ของฐานข้อมูล เช่น Dblist, DbCombo, DbGrid

5.2.2.1 คอนโทรล Data

เป็นคอนโทรลที่ใช้เข้าถึงฐานข้อมูล และทำหน้าที่เชื่อมต่อการแสดงผลข้อมูลแต่ละฟิลด์ในฐานข้อมูลเข้ากับคอนโทรลด้านฐานข้อมูล โดยเมื่อมีการเคลื่อนที่ไปยังเรคคอร์ดใด ๆ ในฐานข้อมูลด้วยคอนโทรล Data ข้อมูลที่ถูกแก้ไขในคอนโทรลด้านฐานข้อมูลที่เชื่อมต่อเข้ากับคอนโทรล Data ซึ่งเป็นเรคคอร์ดปัจจุบันในขณะนั้น ก็จะถูกจัดเก็บลงในฐานข้อมูลโดยอัตโนมัติแล้วจึงเคลื่อนที่ไปยังเรคคอร์ดถัดไปทันที แต่เนื่องจากคอนโทรล Data สามารถที่จะจัดเก็บข้อมูลที่แก้ไขให้โดยอัตโนมัติ ดังนั้นถ้าหากเราต้องการตรวจสอบความถูกต้องของข้อมูลก่อนที่จะจัดเก็บโดยคอนโทรล Data ก็สามารถกระทำได้โดยการเขียนโค้ดเพื่อตรวจสอบความถูกต้องของข้อมูลในโพซีเยอร์ (Procedure) ของเหตุการณ์ Validate ในขณะออกแบบคอนโทรล Data ที่วางลงแบบฟอร์ม ดังแสดงในรูปที่ 5-5



รูปที่ 5-5 คอนโทรล Data

5.2.2.2 คอนโทรล DBGrid (Apex Data-Bound Grid)

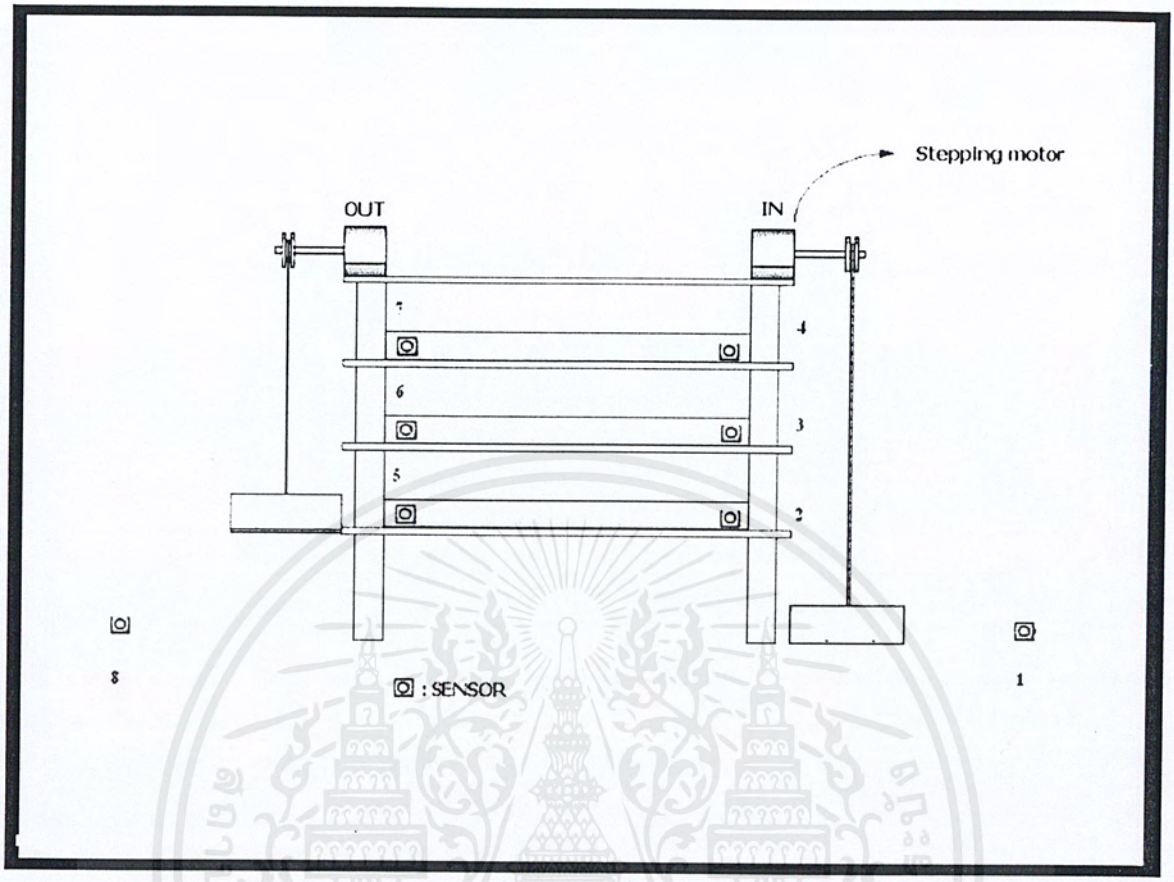
ทำหน้าที่แสดงผลเรคคอร์ดในรูปแบบของตาราง (Grid) โดยที่เราต้องกำหนดชื่อของคอนโทรล Data ให้กับคุณสมบัติ DataSource ของคอนโทรล DBGrid ซึ่งข้อมูลทั้งหมดของเรคคอร์ดจะถูกแสดงผลภายในคอนโทรล DBGrid โดยอัตโนมัติ โดยปกติคอนโทรล DBGrid จะสามารถแสดงจำนวนคอลัมน์ได้มากถึง 1700 คอลัมน์ ส่วนจำนวนแถวก็สามารถแสดงผลได้มากที่สุดเท่าที่ทรัพยากรจะอำนวย และในการเคลื่อนที่ไปยัง ตำแหน่งใด ๆ ของคอนโทรล DBGrid ด้วยวิธี Move เราก็ควรจะเขียนโค้ดเพื่อเรียกใช้วิธี Refresh ให้ทำการวาดคอนโทรลใหม่อีกครั้ง

บทที่ 6
อาคารจำลองที่จอดรถแบบอัตโนมัติ

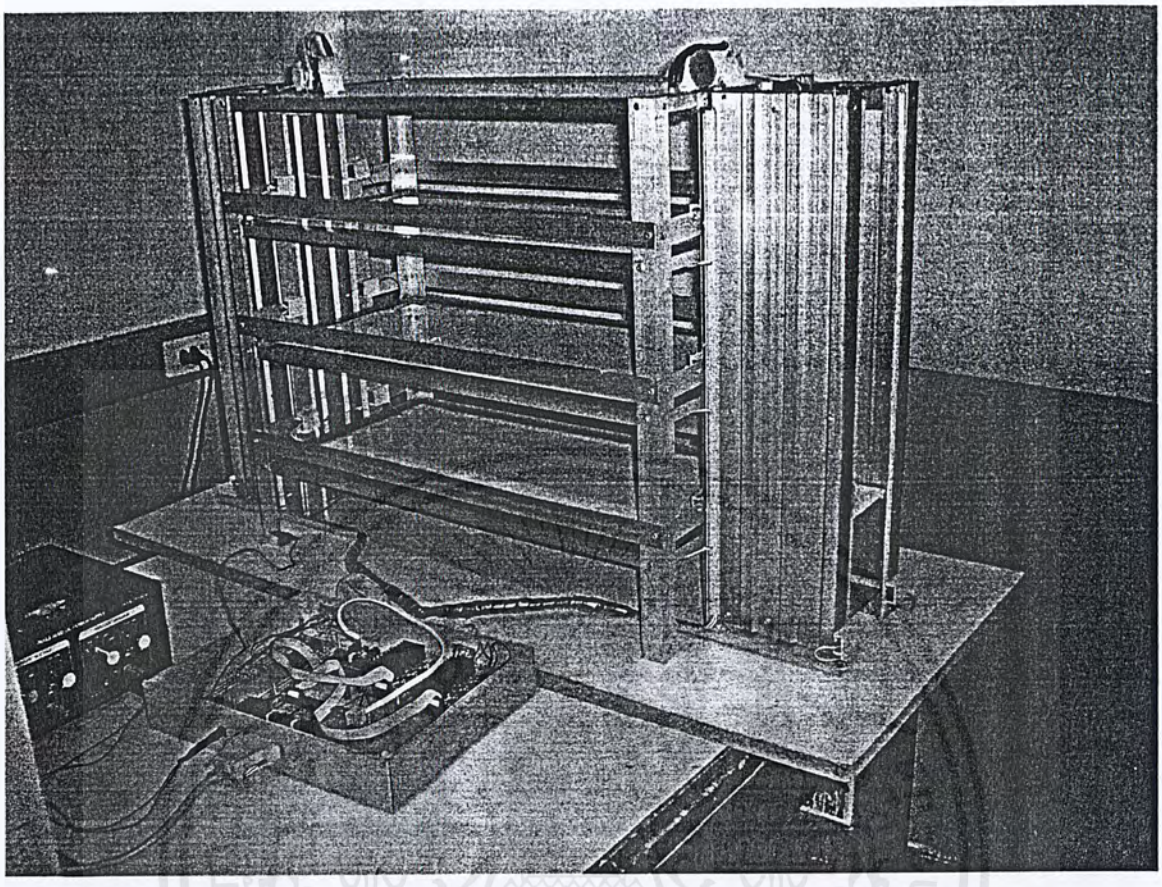
6.1 ลักษณะของอาคารจำลองที่จอดรถ

การออกแบบโครงสร้างของอาคารจำลองที่จอดรถ โดยออกแบบให้มีขนาดความยาว 2.5 ฟุต กว้าง 1.5 ฟุต และความสูงแต่ละชั้น 0.5 ฟุต จำนวน 3 ชั้น โดยกำหนดให้ในแต่ละชั้น จอดรถได้ 10 คัน พร้อมทั้งโครงสร้างด้านข้างทั้งสองของตัวอาคารจำลองเพื่อติดตั้งลิฟท์ในการขึ้นลงรับส่งรถ

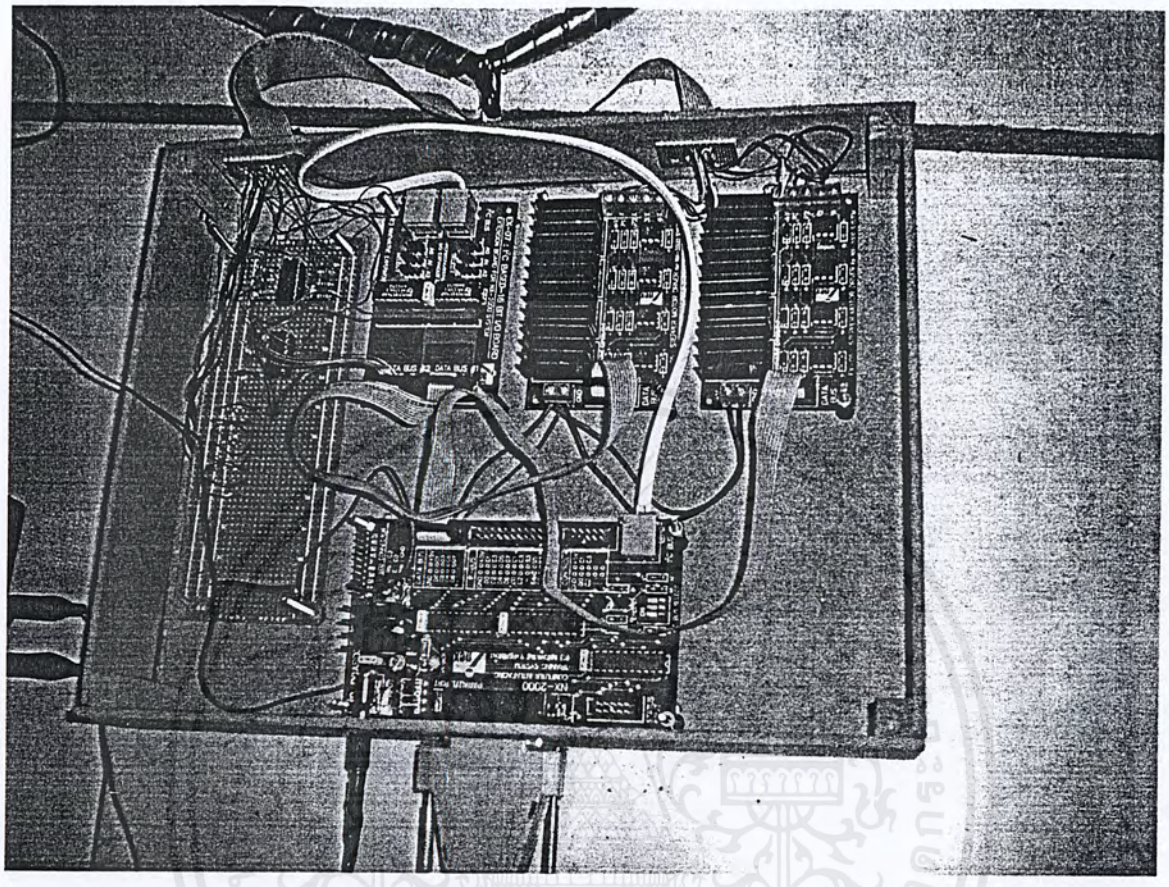
การติดตั้งตัวตรวจจับติดตั้งไว้จำนวน 8 จุดเพื่อทำการตรวจสอบเมื่อมีรถเข้า – ออก จากตัวอาคาร โดยติดตั้งบริเวณก่อนทางเข้าลิฟท์ 1 ตัว เพื่อตรวจสอบว่ามีรถเข้ามาและสั่งงานให้ลิฟท์ลงมารับรถยังชั้นล่างสุด และติดตั้งในตัวอาคารทางเข้าและทางออกในแต่ละชั้น 3 ตัว ทั้งหมดรวม 6 ตัว โดยตัวตรวจจับทางด้านเข้าจะทำหน้าที่ตรวจสอบว่ามีรถเข้ามาจอดในชั้นนั้น ๆ และเก็บค่าจำนวนรถที่เข้ามาจอดไว้ ส่วนตัวตรวจจับทางด้านรถออกเมื่อรถวิ่งผ่านตัวตรวจจับตัวตรวจจับนั้นจะทำหน้าที่ตรวจสอบว่าในชั้นนั้น ๆ มีรถต้องการออกและจะสั่งให้ลิฟท์ขึ้นมารับรถลงไปยังชั้นล่างสุด พร้อมทั้งทำการลบค่าจำนวนรถที่จอดอยู่ในชั้นนั้น ๆ ออก 1 คัน และมีตัวตรวจจับอีก 1 ตัวติดตั้งไว้ที่บริเวณทางออกชั้นล่าง เมื่อรถวิ่งออกจากลิฟท์ผ่านตัวตรวจจับก็จะตรวจสอบได้ว่ารถได้ออกจากลิฟท์ไปแล้ว แล้วทำการรีเซ็ตระบบเพื่อให้ระบบพร้อมทำงานอีกครั้งทางด้านรถออก ดังแสดงในรูปที่ 6-1 และรูปที่ 6-2 ตามลำดับ โดยมีแบบ (Drawing) และแบบจำลอง (Model)



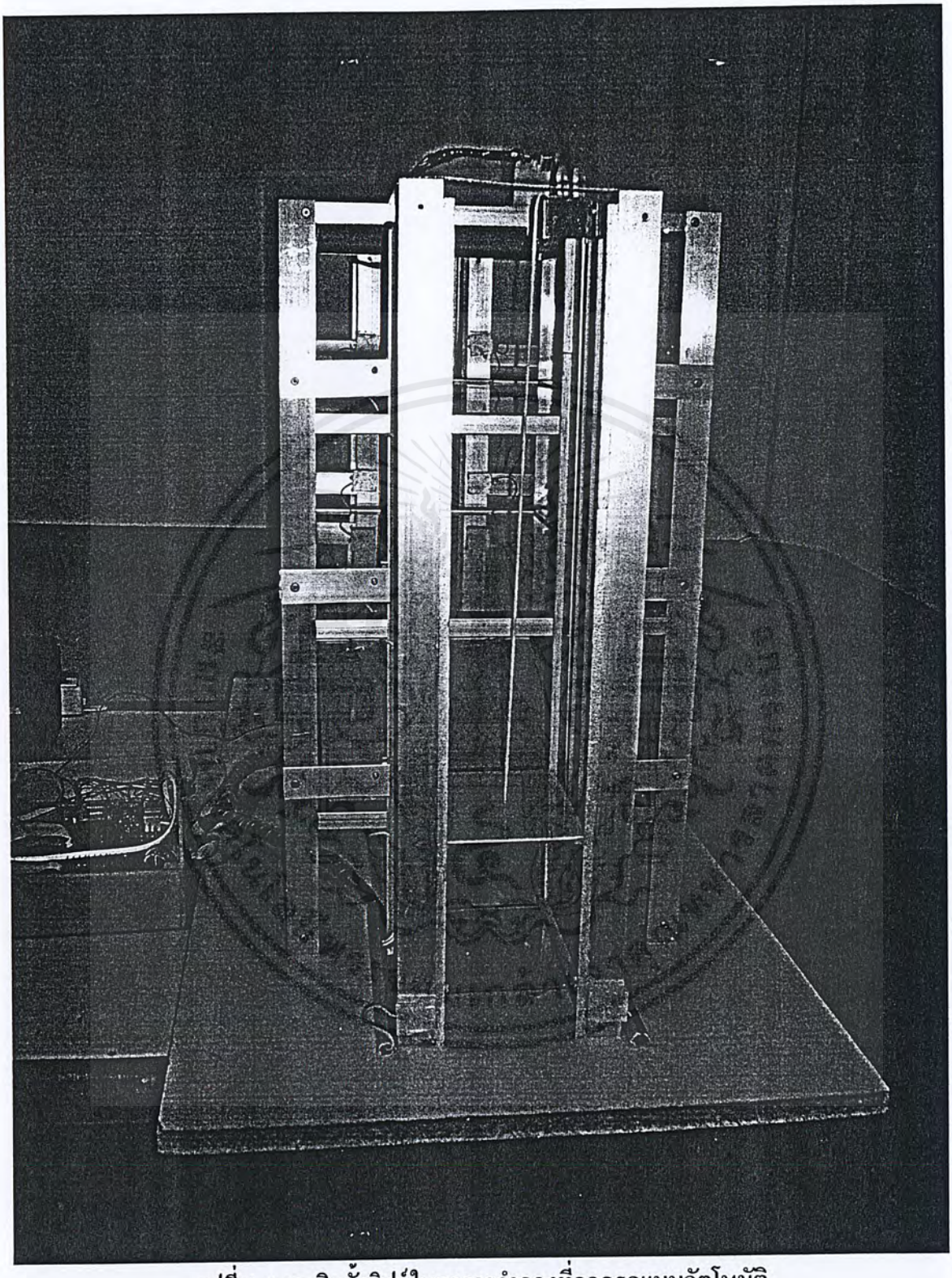
รูปที่ 6-1 แบบอาคารจำลองที่จอดรถ



รูปที่ 6-2 อาคารจำลองที่จอดรถแบบอัตโนมัติ



รูปที่ 6-3 บอร์ดวงจรการทำงานของระบบ

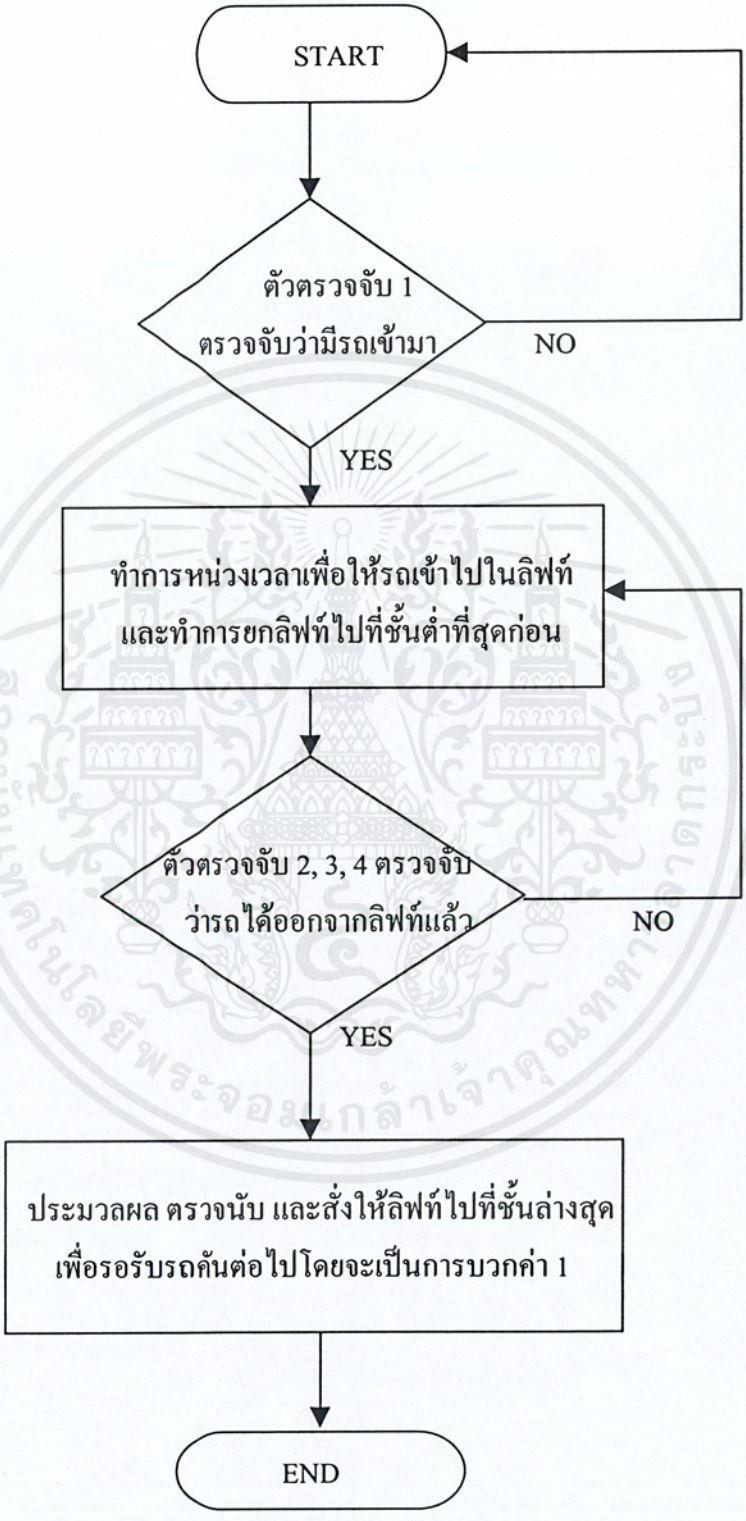


รูปที่ 6-4 การติดตั้งลิฟท์ในอาคารจำลองที่จอดรถแบบอัตโนมัติ

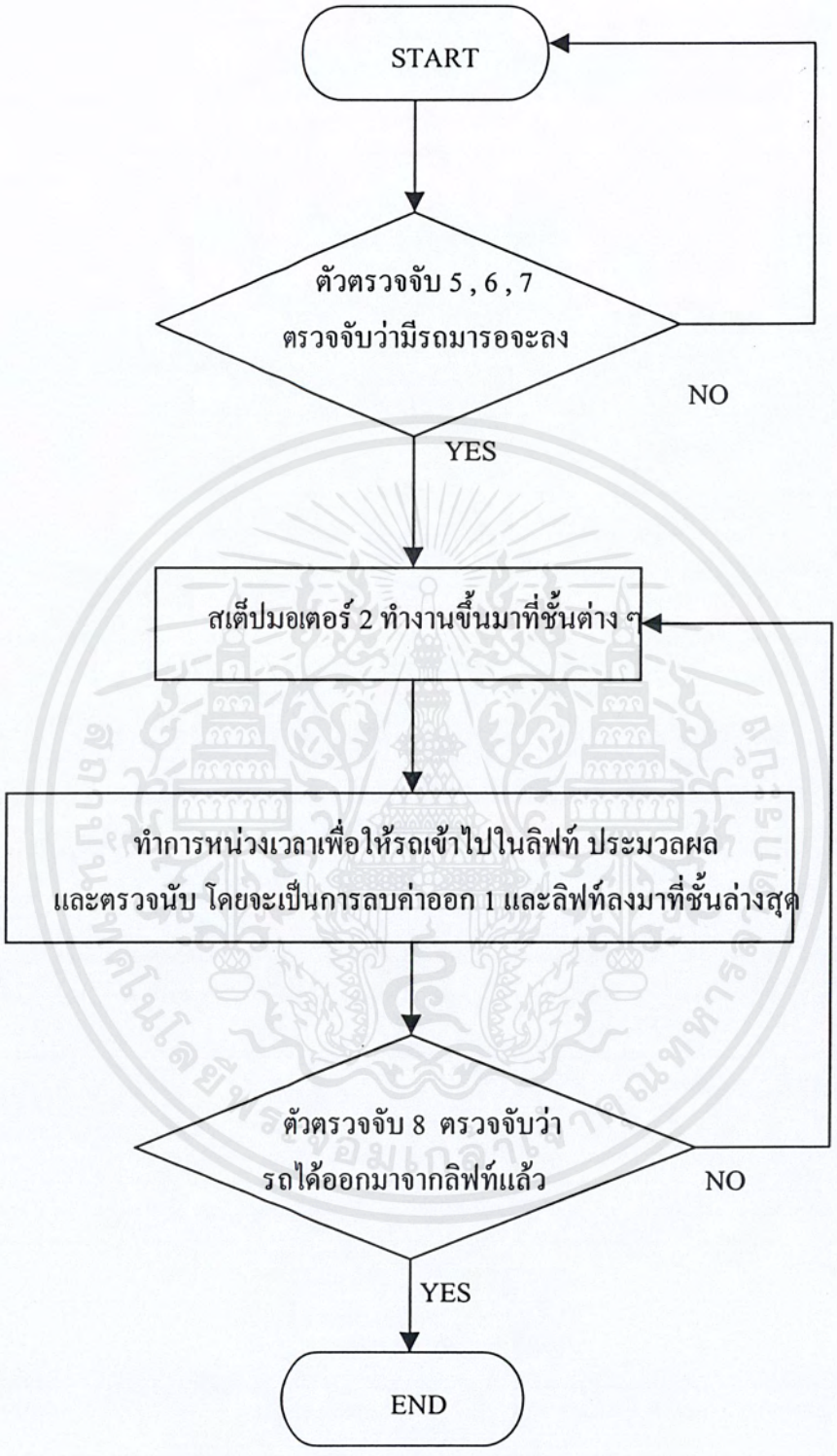
6.2 แผนขั้นตอนการทำงานของโปรแกรม

การควบคุมการทำงานของอาคารจำลองที่จอดรถนั้น แบ่งการควบคุมออกเป็น 2 แบบ คือ แบบที่กำหนดเองในการจอดรถเอง (Manual) และ แบบอัตโนมัติ (Automatic)

ในการควบคุมแบบอัตโมัติ จะให้ความสำคัญในการจอดรถโดยเรียงจากชั้น 1, 2 และ 3 ตามลำดับ การควบคุมที่จอดรถทั้ง 2 แบบ จะมีผังขั้นตอนในส่วนที่ควบคุมทางลงของลิฟท์ที่เหมือนกัน โดยอ้างหมายเลขของตัวตรวจจับ จากรูปที่ 6-1 ดังแสดงขั้นตอนการทำงานในรูปที่ 6-5 และรูปที่ 6-6 ตามลำดับ



รูปที่ 6-5 ผังขั้นตอนการทำงานทางด้านขาขึ้น



รูปที่ 6-6 ผังขั้นตอนการทำงานทางด้านขาลง

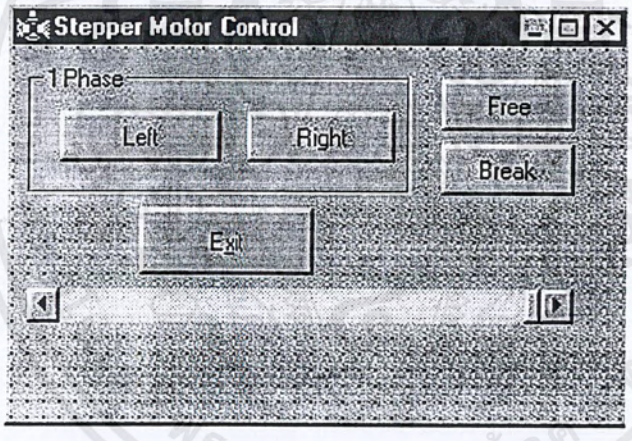
6.3 การทดลอง

การทดลองได้แบ่งออกเป็น 3 ขั้นตอน ดังนี้

1. การทดลองขับสเต็ปมอเตอร์โดยใช้วงจรขับสเต็ปมอเตอร์
2. การทดลองส่งข้อมูลผ่านไอซีขยายพอร์ต เบอร์ PCF8574 โดยเขียนโปรแกรมคำสั่ง
3. การทดลองโดยการเขียนโปรแกรมควบคุมโ I²C โดยรับค่าจากตัวตรวจจับเพื่อขับสเต็ปมอเตอร์ผ่านบอร์ดวงจรขนานและผ่านพอร์ตขยาย I²C เพื่อควบคุมลิฟท์

6.3.1 แสดงผลการทดลองที่ 1

โดยการเชื่อมต่อบอร์ดวงจรขนานกับวงจรขับสเต็ปมอเตอร์เข้าด้วยกัน และเขียนโปรแกรมคำสั่งเพื่อทำการทดสอบการหมุนของสเต็ปมอเตอร์ว่าสามารถควบคุมการหมุนได้ โดยสามารถควบคุมให้หมุนไปทางซ้ายหรือหมุนไปทางขวาหรือการหมุนแบบเป็นอิสระได้



รูปที่ 6-7 รูปแบบการหมุนของสเต็ปมอเตอร์

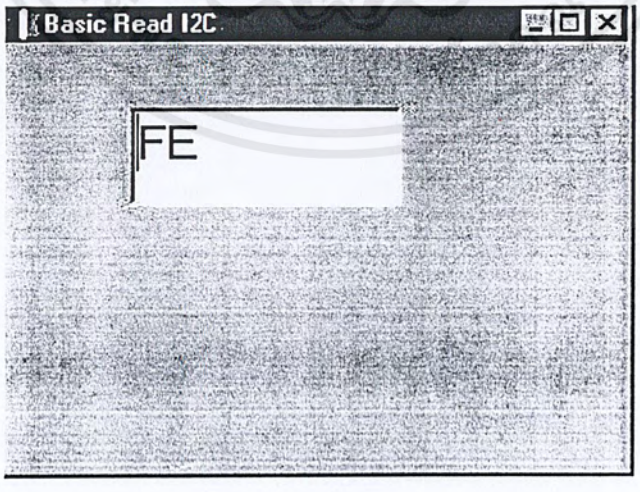
- Left คือ ปุ่มสั่งให้มอเตอร์หมุนไปทางซ้าย
- Right คือ ปุ่มสั่งให้มอเตอร์หมุนขวา
- Free คือ ปุ่มสั่งให้มอเตอร์หมุนอิสระ
- Break คือ ปุ่มสั่งมอเตอร์หยุดหมุน
- Exit คือ ปุ่มสั่งออกจากการรันโปรแกรม

6.3.2 แสดงการทดลองที่ 2

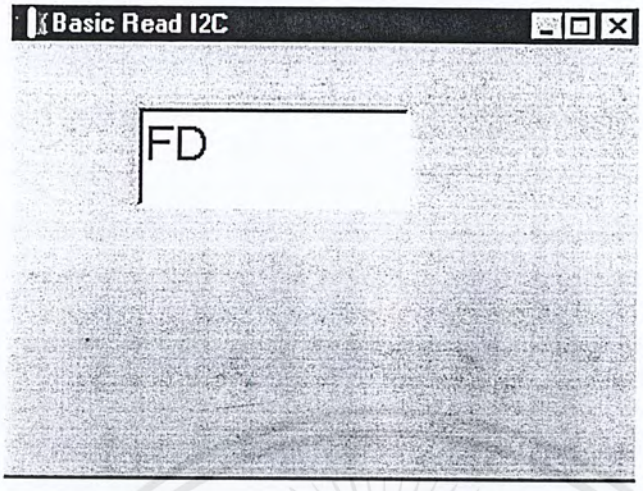
โดยการเชื่อมบอร์ดวงจรขนานกับวงจรขยายพอร์ต I²C เพื่อทำการเช็คว่าวงจรสามารถรับค่าจาก ตัวตรวจจับได้หรือไม่ โดยการทดสอบตัวตรวจจับทีละตัวว่ามีการเปลี่ยนค่าหรือไม่เมื่อมีวัตถุผ่านตัวตรวจจับไปแล้วค่าสัญญาณจะกลับมาเป็น FF ตัวอย่างเช่นเมื่อวัตถุวิ่งผ่านตัวตรวจจับตัวที่ 1 ด้านทางเข้าค่าสัญญาณที่ได้จะเป็น FE เมื่อวัตถุผ่านไปแล้วค่าสัญญาณก็จะกลับมาเป็น FF เช่นเดิม โดยค่าสัญญาณต่าง ๆ ในตัวตรวจจับแต่ละตัวดังแสดงในตารางที่ 6-1

ตารางที่ 6-1 ค่าสถานะของสัญญาณในตัวตรวจจับต่าง ๆ

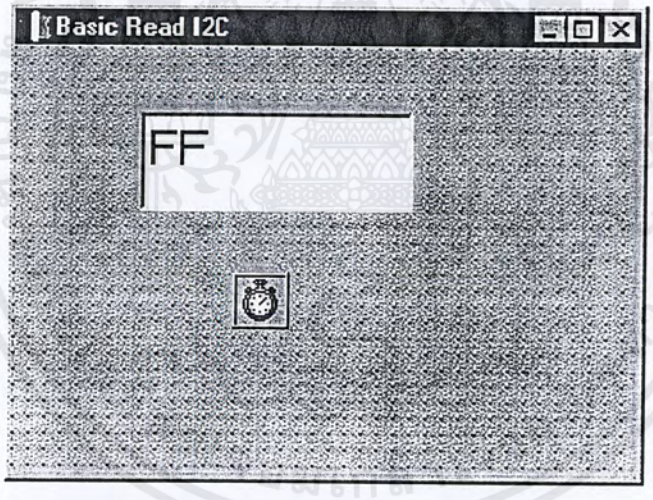
ตัวตรวจจับ	ค่าสถานะของสัญญาณ
1	FE
2	FD
3	FB
4	E7
5	DF
6	BF
7	7F
8	EF



รูปที่ 6-8 ค่าสถานะของสัญญาณในตัวตรวจจับตัวที่ 1



รูปที่ 6-9 ค่าสถานะของสัญญาณในตัวตรวจจับตัวที่ 2



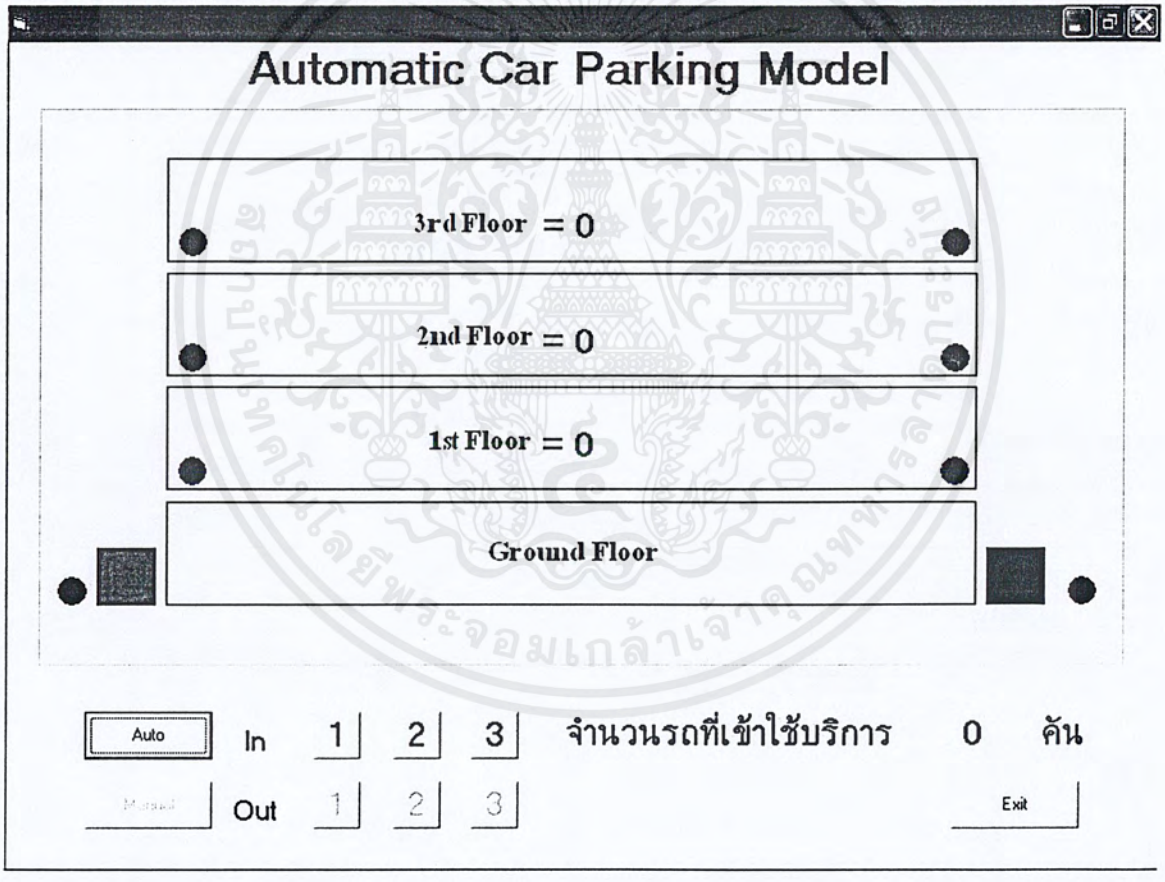
รูปที่ 6-10 ค่าเมื่อตัวตรวจจับเข้าสู่สภาวะปกติ

6.3.3 แสดงผลการทดลองที่ 3

โดยการเชื่อมต่อวงจรทั้งหมดเข้าด้วยกันและทำงานโดยรับค่าสัญญาณจากตัวตรวจจับและทำงานตามโปรแกรมที่เขียนและออกแบบไว้ หลักการทำงานคือ เลือกสวิตซ์แบบอัตโนมัติ (auto) หรือ manual และ เมื่อเลือกแบบอัตโนมัติ เมื่อมีรถเข้ามาตัวตรวจจับจะตรวจจับและหน่วงเวลาว่าได้มีรถเข้ามาในลิฟท์แล้วและสั่งให้ลิฟท์มาที่ชั้นหนึ่งและจะมีตัวตรวจจับตรวจในแต่ละชั้นเพื่อทำการเช็ค

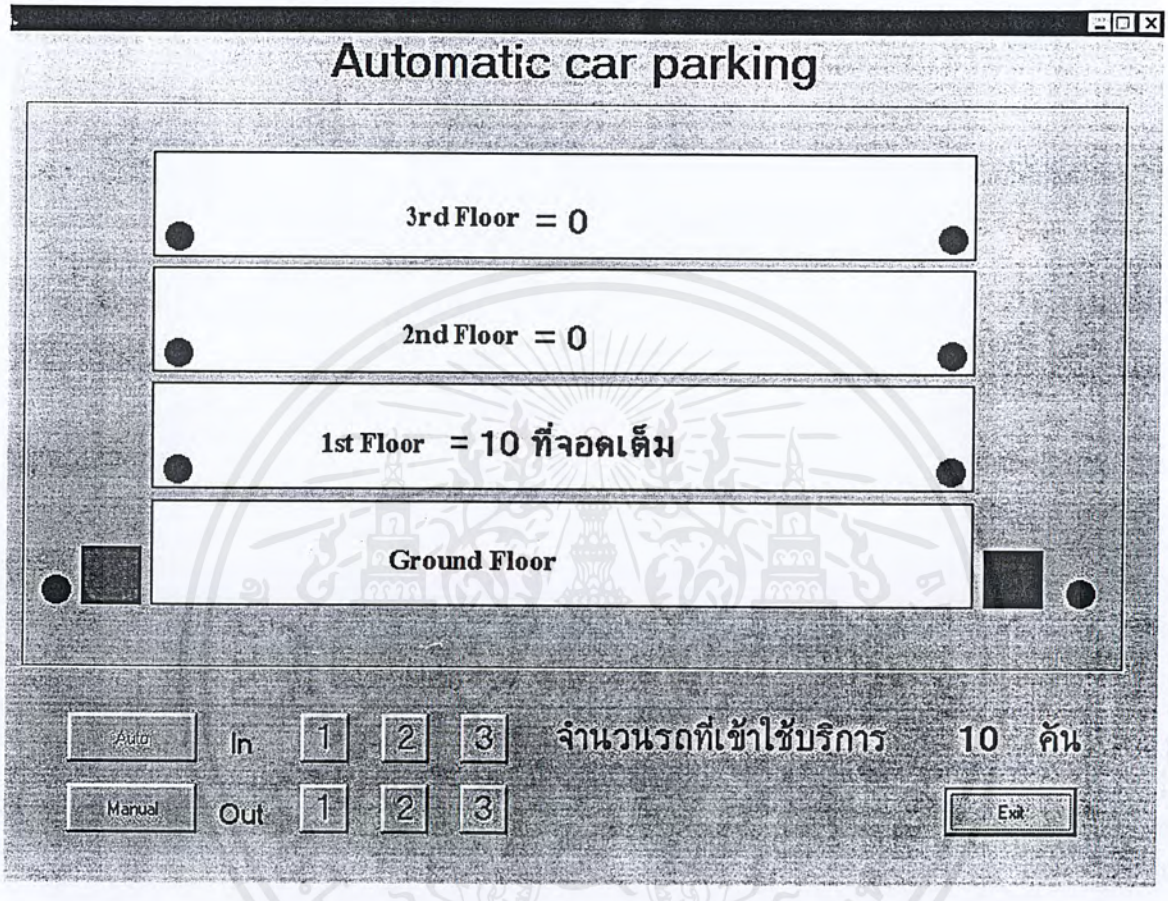
จำนวนรถที่เข้ามาจอดและเพื่อเป็นการตรวจว่ารถได้ออกมาจากลิฟท์แล้วและทำการบวกจำนวนรถที่เข้ามาจอดในชั้นนั้น ๆ เมื่อจอดครบ 10 คัน จะขึ้นว่าเต็ม แล้วต่อมาจะมาที่ชั้น 2 และ 3 เรื่อยๆ ในขณะเดียวกันทางลิฟท์ด้านนำรถลง เมื่อมีรถมาที่ตัวตรวจจับในแต่ละชั้น ลิฟท์จะมาที่ชั้นนั้นและทำการหน่วงเวลาเพื่อให้รถเข้าไปในลิฟท์ แล้วลงมาส่งรถที่ชั้นล่างสุดและจะมีตัวตรวจจับตรวจว่ารถได้ออกจากลิฟท์ทางลงไปแล้ว และทำกระบวนการต่อไปเรื่อยๆ ส่วนการเลือกแบบ manual คือเราสามารถเลือกหรือกำหนดได้ว่าต้องการนำรถไปจอดที่ชั้นใดก็ได้โดยการเลือกก่อนนำรถเข้าไปจอด

ดังแสดงในรูปที่ 6-11 เป็นแบบแสดงผลการทำงานในสภาวะเริ่มต้น คือ ในชั้นที่ 1, 2 และ 3 ยังไม่มีรถขึ้นไปจอดจึงยังไม่มีการแสดงจำนวนรถในแต่ละชั้น



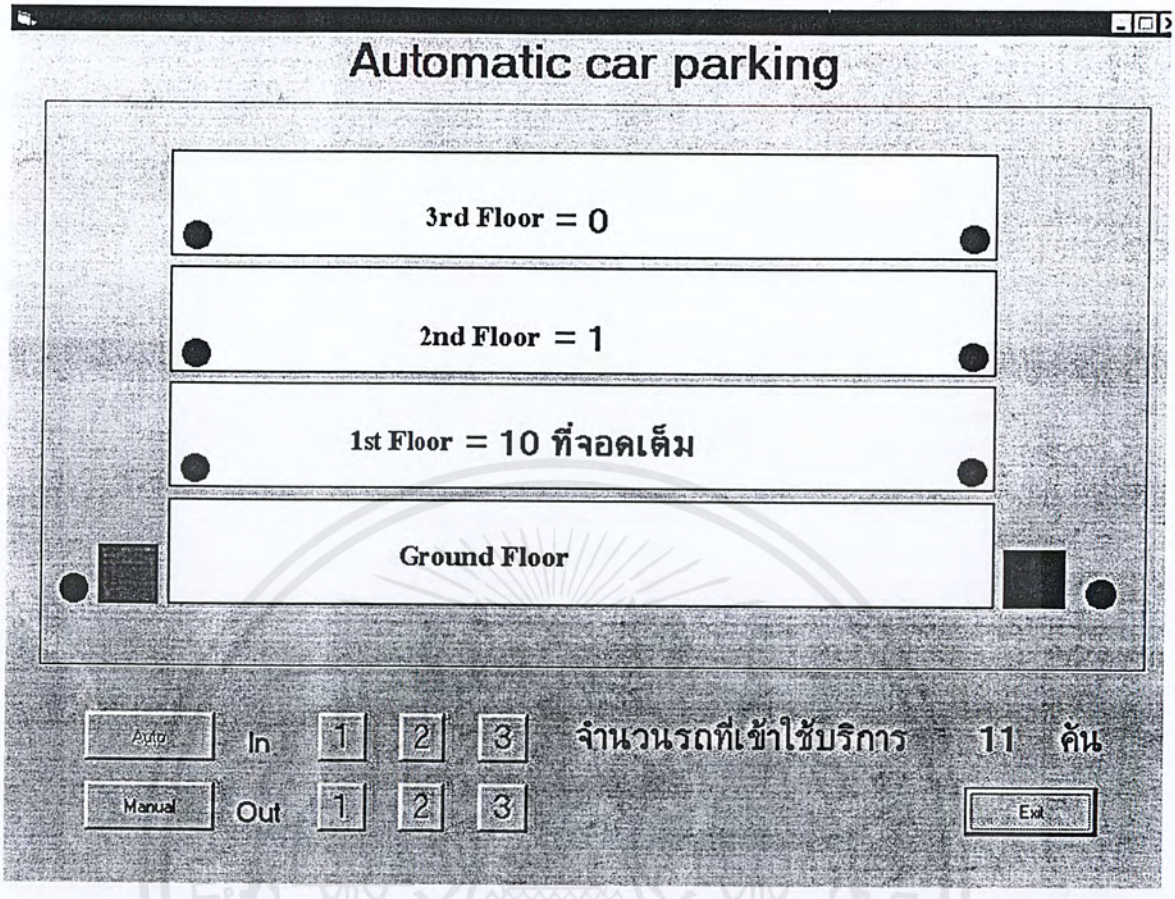
รูปที่ 6-11 รูปแบบการแสดงผลของอาคารจำลองที่จอดรถแบบอัตโนมัติ

แสดงผลการทำงานเมื่อมีรถขึ้นมาจอดในชั้นที่ 1 จนกระทั้งเต็ม 10 คัน โดยจะแสดงจำนวนรถที่มาจอดในชั้นที่ 1 และแสดงข้อความว่าที่จอดเต็ม และแสดงจำนวนรถที่เข้าใช้บริการทั้งหมด 10 คัน



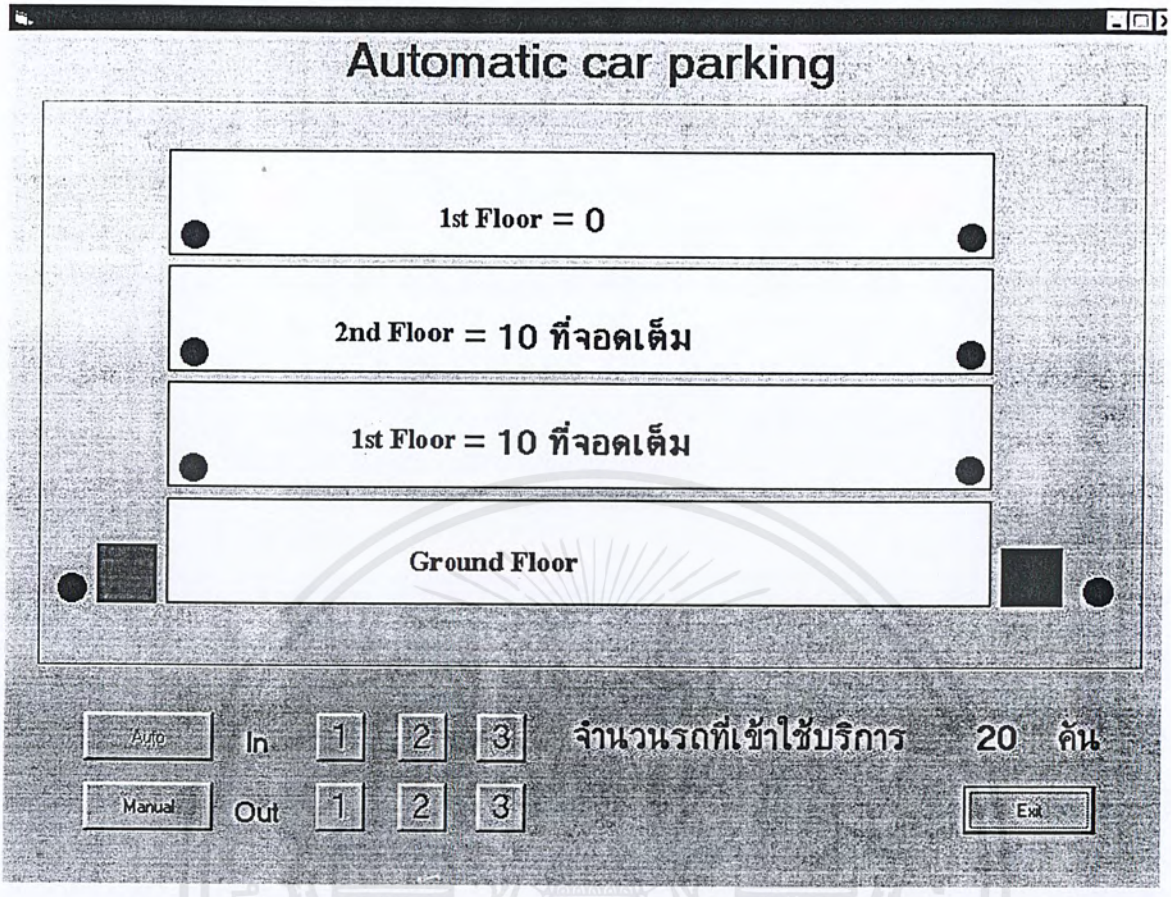
รูปที่ 6-12 รูปแบบแสดงผลเมื่อมีรถจอดเต็มในชั้นที่ 1

เมื่อชั้นที่ 1 เต็ม ลิฟท์ก็จะยกรถขึ้นไปจอดในชั้นที่ 2 และแสดงค่าของรถที่ขึ้นไปจอดในชั้นที่ 2 และแสดงจำนวนรถทั้งหมดที่เข้าใช้บริการ ในที่นี้แสดงจำนวนรถ 1 คันในชั้นที่ 2 และจำนวนรถทั้งหมดที่เข้าใช้บริการ 11 คัน ดังแสดงในรูปที่ 6-13

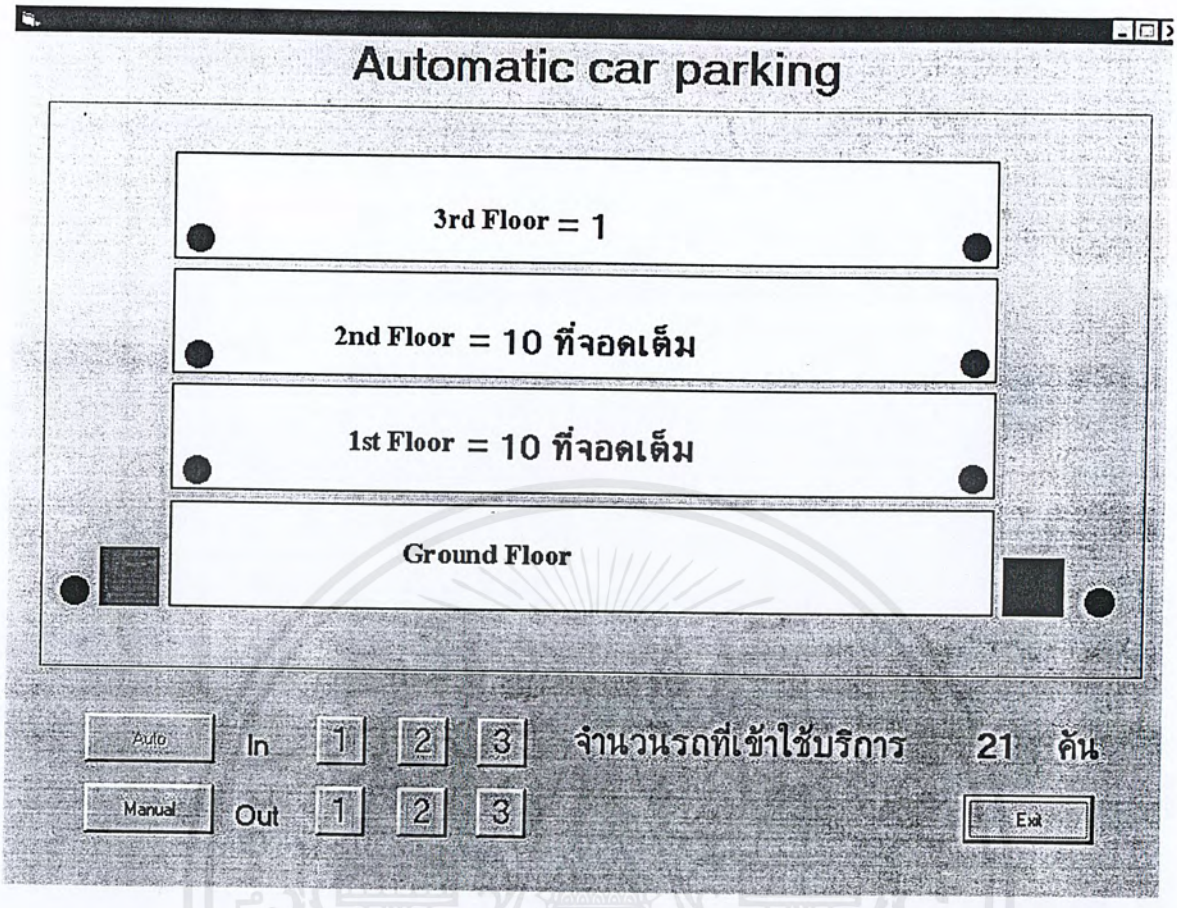


รูปที่ 6-13 รูปแบบแสดงผลจำนวนรถในชั้นที่ 2

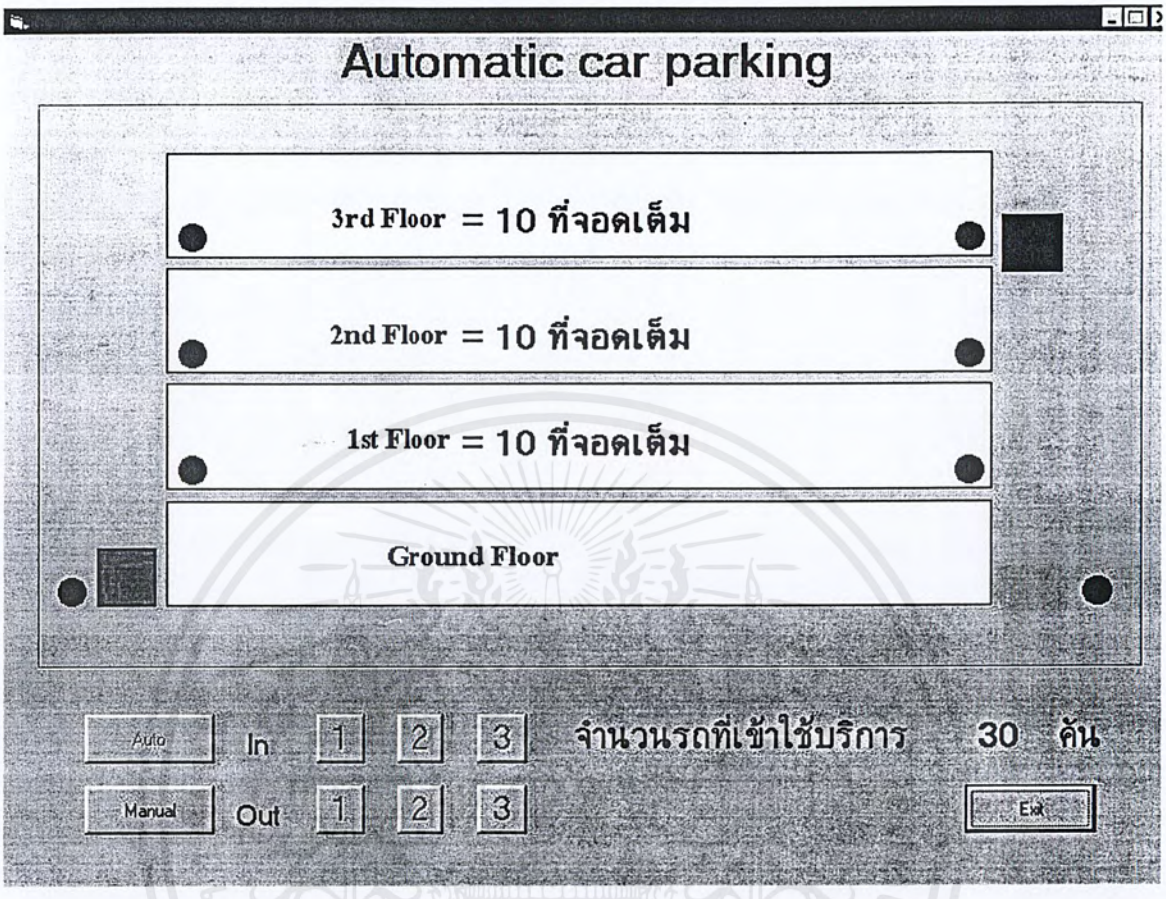
และเมื่อเมื่อชั้นที่ 2 เต็มก็จะแสดงข้อความว่าที่จอดเต็ม ดังแสดงในรูปที่ 6-14 และลิฟท์ก็จะยกรถคันต่อไปขึ้นไปยังชั้นที่ 3 และเมื่อชั้น 3 เต็มก็จะแสดงข้อความที่จอดรถเต็ม ดังแสดงในรูปที่ 6-15 และ 6-16



รูปที่ 6-14 รูปแบบแสดงผลเมื่อที่จอดรถชั้น 2 เต็ม



รูปที่ 6-15 รูปแบบแสดงผลเมื่อรถเริ่มยกขึ้นไปในชั้นที่ 3



รูปที่ 6-16 รูปแบบแสดงผลเมื่อที่จอดรถทั้ง 3 ชั้นเต็ม

บทที่ 7
บทสรุปและวิจารณ์

7.1 บทสรุปและวิจารณ์

การควบคุมการจอดรถ สำหรับแบบจำลองอาคารจอดรถที่ได้นำเสนอในโครงงานนี้จะแบ่งการควบคุมออกเป็น 2 แบบ คือ การควบคุมแบบ Manual ซึ่งจะสามารถกำหนดชั้นในการจอดรถได้เอง และแบบอัตโนมัติ ในการควบคุมแบบอัตโนมัติจะให้ความสำคัญในการจอดรถในชั้นที่ 1, 2 และ 3 เรียงตามลำดับ พร้อมทั้งตรวจสอบจำนวนรถที่จอดอยู่ในแต่ละชั้น และจำนวนรถทั้งหมดด้วย

7.2 ข้อเสนอแนะและแนวทางในการวิจัยและพัฒนาต่อ

สิ่งที่นำเสนอไปเป็นเพียงแนวความคิดต้นแบบ ที่ยังไม่สามารถนำไปใช้งานจริงได้ทั้งหมด แต่สามารถนำไปเป็นต้นแบบในการพัฒนาต่อไปได้ ถ้าเราต้องการที่จะให้ใช้งานจริงได้ สิ่งที่เราต้องปรับปรุงพัฒนาขึ้นมาใหม่ก็คือ

- โปรแกรมที่ใช้ในการควบคุม เราต้องกำหนดว่าต้องการการควบคุมแบบไหน ให้ลำดับความสำคัญที่ชั้นไหนก่อนหลัง โดยต้องคำนึงถึงจำนวนรถที่จะจอดรวมถึงจำนวนชั้นของอาคาร
- มอเตอร์ที่จะใช้งานในการขับลิฟท์ขึ้นลง จากแบบจำลองเราใช้เพียงสเต็ปมอเตอร์ ถ้าจะนำไปใช้งานจริงต้องใช้มอเตอร์ที่มีทอร์กมากกว่านี้
- ในการใช้งานจริงปัจจัยที่มีผลต่อการทำงานของมอเตอร์ที่ใช้ในการยกลิฟท์นั้นคือน้ำหนัก ซึ่งเป็นสิ่งที่สำคัญมากในการคำนวณของระบบทั้งน้ำหนักของลิฟท์และน้ำหนักของรถ
- วงจรไฟฟ้าและกระแสไฟฟ้าที่จ่ายให้ระบบ เพราะในแบบจำลองกับที่ใช้งานจริงจะใช้พลังงานที่ที่แตกต่างกัน
- อุปกรณ์ไฟฟ้าต่าง ๆ รวมถึงระบบที่อำนวยความสะดวกและอุปกรณ์ประเภทตัวตรวจจับต่าง ๆ ก็ต้องมีประสิทธิภาพที่สูงกว่าอุปกรณ์ต้นแบบ

นอกจากนี้การพัฒนาโครงงานยังสามารถเพิ่มขีดความสามารถในการทำงานเพิ่มขึ้นได้อีก ดังเช่น

1.สามารถเพิ่มจำนวนชั้นและขนาดความกว้างของอาคารที่จอดรถเพื่อให้สามารถจอดรถได้จำนวนมากขึ้น

2.จัดระบบการจัดการของตัวตรวจจับเพื่อให้มีประสิทธิภาพในการใช้งานมากที่สุดเมื่อนำโครงงานนี้ไปใช้จริง

3.สามารถประยุกต์การทำงานโดยใช้การควบคุมประเภทอื่นได้อีก เช่นใช้ไมโครคอนโทรเลอร์ PLC หรืออุปกรณ์ควบคุมประเภทอื่นได้

บรรณานุกรม

[1] http:// www.thaiio.com

[2] http:// www.thaibit.hypermart.net

[3] ชัยวัฒน์ ลิ้มพรจิตรวิไล , อรรถพล บุญยะโภคา , **เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตขนาน** , บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด , กรุงเทพฯ : 2544 .

[4] มงคล ทองสงคราม , **มอเตอร์ไฟฟ้ากระแสตรง** , บริษัท รามาการพิมพ์ จำกัด , กรุงเทพ ฯ .

[5] A.K. Sawhney, **Electrical and Electronic Measurements and Instrumentation ,** Dhanpat Rai & Sons ,Educational and Technical publishers 1682,Nai Sarak ,Delhi-110006, Phone 265367

ภาคผนวก

โปรแกรมการทำงานของระบบ

```
Private Sub Form_Load()
    Time_Delay_M = 500
    Time_Delay_S = 50000
    DIL = 29
    DB(0) = 0
    DB(1) = 0
    DB(2) = 0
    Status_Lift01 = -1
    Status_Lift02 = -1
    Para_Lift = 1180
    Mode = True
    SOut = True
    Temp = False
    MoveX = True
    MoveY = True
    showc(0) = 1 & " = " & DB(0)
    showc(1) = 2 & " = " & DB(1)
    showc(2) = 3 & " = " & DB(2)
    InCome = 0
End Sub

Private Sub auto_Click()
    manual.Enabled = True
    auto.Enabled = False
    Mode = False
    For i = 0 To 2
        status(i).Enabled = False
        status0(i).Enabled = False
    Next i
    Call Auto_Act
```

```
End Sub

Private Sub manual_Click()
   manual.Enabled = False
   auto.Enabled = True
   Mode = True
   For i = 0 To 2
      status(i).Enabled = True
      status0(i).Enabled = True
   Next i
End Sub

Private Sub Command1_Click()
   Call MoveDownLIN(Lin, 3)
   Call MoveDownLOUT(Lout, 3)
   End
End Sub

Private Sub showc_Click(Index As Integer)
   InCome = InCome + 9 - DB(Index)
   DB(Index) = 9
   showc(Index).Caption = (Index + 1) & " = " & DB(Index)
   OutAll.Caption = InCome
End Sub

Private Sub status_Click(Index As Integer)
   If DB(Index) < 10 Then
      If Mode = True And MoveX = True Then
         If (Index <> Status_Lift01) Then
            If Index < Status_Lift01 Then
               Call MoveDownLIN(Lin, 2 - Index)
               Status_Lift01 = Index
```

```
        Else
            Call MoveUpLIN(Lin, Index + 1)
            Status_Lift01 = Index
        End If
    End If
    Do
        InData = Recv_In()
        If (Index = 0 And InData = &HFD) Then
            SH(1).FillColor = &HFF&
            Temp = True
        End If
        If (Index = 1 And InData = &HFB) Then
            SH(2).FillColor = &HFF&
            Temp = True
        End If
        If (Index = 2 And InData = &HF7) Then
            SH(3).FillColor = &HFF&
            Temp = True
        End If
        OutPut = InData
        Call Delay
    Loop Until Temp = True
    InCome = InCome + 1
    OutAll.Caption = InCome
    DB(Index) = DB(Index) + 1
    showc(Index).Caption = (Index + 1) & " = " & DB(Index)
    Call Delay_Lift(Time_Delay_S)
    SH(Index + 1).FillColor = &H0&
    Call MoveDownLIN(Lin, 3)
    Temp = False
    Status_Lift01 = -1
End If
```

```
    End If
    If (DB(Index) > 0) Then
        status0(Index).Enabled = True
    End If
    If DB(Index) = 10 Then
        status(Index).Enabled = False
        showc(Index).Caption = (Index + 1) & " = " & DB(Index) & "ที่จอดเต็ม"
    End If

End Sub

Private Sub status0_Click(Index As Integer)
    If (DB(Index) > 0) Then
        If Mode = True And MoveY = True And SOut = True Then
            MoveY = False
            SOut = False
            status0(0).Enabled = False
            status0(1).Enabled = False
            status0(2).Enabled = False
            Call MoveUpLOUT(Lout, Index + 1)
            Do
                InData = Recv_In()
                If (Index = 0 And InData = &HDF) Then
                    SH(5).FillColor = &HFF&
                    Temp = True
                End If
                If (Index = 1 And InData = &HBF) Then
                    SH(6).FillColor = &HFF&
                    Temp = True
                End If
                If (Index = 2 And InData = &H7F) Then
                    SH(7).FillColor = &HFF&
```

```
            Temp = True
          End If
          OutPut = InData
          Call Delay

        Loop Until Temp = True
        DB(Index) = DB(Index) - 1

        showc(Index).Caption = (Index + 1) & " = " & DB(Index)
        Call Delay_Lift(Time_Delay_S)
        SH(Index + 5).FillColor = &H0&
        Call MoveDownLOUT(Lout, 3)
        status(Index).Enabled = True
        Temp = False
        Do
          InData = Recv_In()
          If (InData = &HEF) Then
            SH(4).FillColor = &HFF&
            Temp = True
            SOut = True
          End If
          Call Delay_Lift(Time_Delay_S)
          OutPut = InData
          SH(4).FillColor = &H0&
        Loop Until Temp = True
      End If
      For i = 0 To 2
        If (DB(i) > 0) Then
          status0(i).Enabled = True
        End If
      Next i
    Else
```

```
        showc(Index) = Index & " = " & DB(Index)
    End If

        MoveY = True
        Temp = False
End Sub

Public Sub Auto_Act()
Do
    InData = Recv_In()
    Select Case (InData)
        Case &HFE
            SH(0).FillColor = &HFF&
            If MoveX = True Then
                Call Delay_Lift(Time_Delay_S)
                SH(0).FillColor = &H0&
                If (DB(0) < 10) Then
                    Call MoveUpLIN(Lin, 1)
                    MoveX = False
                    Status_Lift01 = 1
                Else
                    If (DB(1) < 10) Then
                        Call MoveUpLIN(Lin, 2)
                        MoveX = False
                        Status_Lift01 = 2
                    Else
                        If (DB(2) < 10) Then
                            Call MoveUpLIN(Lin, 3)
                            MoveX = False
                            Status_Lift01 = 3
                        End If
                    End If
```

```
        End If
    End If
    SH(0).FillColor = &H0&

Case &HFD   'in 1
    SH(1).FillColor = &HFF&
    If (Status_Lift01 = 1) Then
        Call Delay_Lift(Time_Delay_S)
        SH(1).FillColor = &H0&
        DB(0) = DB(0) + 1
        If DB(0) < 10 Then
            showc(0).Caption = "1 =" & DB(0)
        Else
            showc(0).Caption = "1 =" & DB(0) & " ที่จอดเต็ม"
        End If
        MoveX = True
        InCome = InCome + 1
        OutAll.Caption = InCome
        Call MoveDownLIN(Lin, 3)
        Status_Lift01 = 0
    End If
    SH(1).FillColor = &H0&

Case &HFB   'IN 2
    SH(2).FillColor = &HFF&
    If (Status_Lift01 = 2) Then
        Call Delay_Lift(Time_Delay_S)
        SH(2).FillColor = &H0&
        DB(1) = DB(1) + 1
        If DB(1) < 10 Then
            showc(1).Caption = "2 =" & DB(1)
        Else
```

```
            showc(1).Caption = "2 = " & DB(1) & " ที่จอดเต็ม"
        End If
        MoveX = True
        InCome = InCome + 1
        OutAll.Caption = InCome
        Call MoveDownLIN(Lin, 3)
        Status_Lift01 = 0
    End If
    SH(2).FillColor = &H0&

Case &HF7   'In 3
    SH(3).FillColor = &HFF&
    If (Status_Lift01 = 3) Then
        Call Delay_Lift(Time_Delay_S)
        SH(3).FillColor = &H0&
        DB(2) = DB(2) + 1
        If DB(2) < 10 Then
            showc(2).Caption = "3 = " & DB(2)
        Else
            showc(2).Caption = "3 = " & DB(2) & " ที่จอดเต็ม"
        End If
        MoveX = True
        InCome = InCome + 1
        OutAll.Caption = InCome
        Call MoveDownLIN(Lin, 3)
        Status_Lift01 = 0
    End If
    SH(3).FillColor = &H0&


Case &HEF   'Out
    SH(4).FillColor = &HFF&
```

```
        Call Delay_Lift(Time_Delay_S)
        MoveY = True
        SH(4).FillColor = &H0&

    Case &HDF   'Out 1
        SH(5).FillColor = &HFF&
        If (MoveY = True And DB(0) <> 0) Then
            MoveY = False
            Call Delay_Lift(Time_Delay_S)
            Call MoveUpLOUT(Lout, 1)
            Call Delay_Lift(Time_Delay_S)
            DB(0) = DB(0) - 1
            If DB(0) < 10 Then
                showc(0).Caption = "1 = " & DB(0)
            Else
                showc(0).Caption = "1 = " & DB(0) & " ที่จอดเต็ม"
            End If
            SH(5).FillColor = &H0&
            Call MoveDownLOUT(Lout, 3)
        End If
        SH(5).FillColor = &H0&

Case &HBF   'Out 2
        SH(6).FillColor = &HFF&
        If (MoveY = True And DB(1) <> 0) Then
            MoveY = False
            Call Delay_Lift(Time_Delay_S)
            Call MoveUpLOUT(Lout, 2)
            Call Delay_Lift(Time_Delay_S)
            DB(1) = DB(1) - 1
            If DB(1) < 10 Then
                showc(1).Caption = "2 = " & DB(1)
```

```
            Else
               showc(1).Caption = "2 = " & DB(1) & " ที่จอดเต็ม"
            End If
            SH(6).FillColor = &H0&
            Call MoveDownLOUT(Lout, 3)
         End If
         SH(6).FillColor = &H0&

      Case &H7F   'Out 3
         SH(7).FillColor = &HFF&
         If (MoveY = True And DB(2) <> 0) Then
            MoveY = False
            Call Delay_Lift(Time_Delay_S)
            Call MoveUpLOUT(Lout, 3)
            Call Delay_Lift(Time_Delay_S)
            DB(2) = DB(2) - 1
            If DB(2) < 10 Then
               showc(2).Caption = "3 = " & DB(2)
            Else
               showc(2).Caption = "3 = " & DB(2) & " ที่จอดเต็ม"
            End If
            SH(7).FillColor = &H0&
            Call MoveDownLOUT(Lout, 3)
         End If
         SH(7).FillColor = &H0&
   End Select
   Call Delay_Lift(3000)
Loop Until Mode = True
End Sub


Public DB(2) As Integer
```

```
Public SOut As Boolean
Public InData As Byte
Public Status_Lift01 As Integer
Public Status_Lift02 As Integer
Public Para_Lift As Integer
Public Mode As Boolean
Public DIL As Integer
Public Time_Delay_S As Long
Public Time_Delay_M As Integer
Public Temp As Boolean
Public MoveX As Boolean
Public MoveY As Boolean
Public InCome As Integer

Public Sub MoveUpLIN(L As Shape, F As Integer)
   Call I2CStart
   Call Send8BIT(&H70)
   Call Ack
   Do
        DoEvents
          L.Top = L.Top - DIL
        Call Send8BIT(9)
        Call Ack
        Call Delay
        Call Send8BIT(3)
        Call Ack
        Call Delay
        Call Send8BIT(6)
        Call Ack
        Call Delay
        Call Send8BIT(12)
        Call Ack
```

```
        Call Delay
    Loop Until L.Top <= (4652 - (F * Para_Lift))
    Call I2CStop
End Sub

Public Sub MoveDownLIN(L As Shape, F As Integer)
    Call I2CStart
    Call Send8BIT(&H70)
    Call Ack
    Do
        DoEvents
          L.Top = L.Top + DIL
        Call Send8BIT(12)
        Call Ack
        Call Delay
        Call Send8BIT(6)
        Call Ack
        Call Delay
        Call Send8BIT(3)
        Call Ack
        Call Delay
        Call Send8BIT(9)
        Call Ack
        Call Delay
    Loop Until L.Top >= ((F * Para_Lift) + 1080)
End Sub

Public Sub MoveUpLOUT(L As Shape, F As Integer)
    Call I2CStart
    Call Send8BIT(&H70)
    Call Ack
    Do
```

```
        DoEvents
          L.Top = L.Top - DIL
        Out &H378, 9
        Call Delay
        Out &H378, 3
        Call Delay
        Out &H378, 6
        Call Delay
        Out &H378, 12
        Call Delay
    Loop Until L.Top <= (4652 - (F * Para_Lift))
    Call I2CStop
End Sub

Public Sub MoveDownLOUT(L As Shape, F As Integer)
    Call I2CStart
    Call Send8BIT(&H70)
    Call Ack
    Do
        DoEvents
          L.Top = L.Top + DIL
        Out &H378, 12
        Call Delay
        Out &H378, 6
        Call Delay
        Out &H378, 3
        Call Delay
        Out &H378, 9
        Call Delay
    Loop Until L.Top >= ((F * Para_Lift) + 1080)
End Sub
```

```
Public Function Recv_In() As Byte
   Dim Data As Byte
   Call I2CStart
   Call Send8BIT(&H73)
   Call Ack
   Data = Read8Bit
   OutPut = inout
   Call I2CStop
   Recv_In = Data
End Function

Public Sub Delay_Lift(ByRef j As Long)
   Dim x As Long
   For i = 0 To j
      DoEvents
   Next i
End Sub

Public Sub Delay()
   For i = 1 To Time_Delay_M
      DoEvents
   Next i
End Sub

Option Explicit
Public Declare Sub Out Lib "io.dll" Alias "PortOut" (ByVal Addr As Integer, ByVal Data As
Byte)
Public Declare Function Inp Lib "io.dll" Alias "PortIn" (ByVal Addr As Integer) As Byte

Public Sub I2CStart()
   SDA_H
   SCL_H
```

```
    SDA_L
    SCL_L
End Sub

Public Sub I2CStop()
    SDA_L
    SCL_H
    SDA_H
End Sub

Public Sub Send0()
    SDA_L
    SCL_H
    SCL_L
End Sub

Public Sub Send1()
    SDA_H
    SCL_H
    SCL_L
End Sub

Public Function Ack() As Boolean
    Ack = Not Rd_SDA
    SCL_H
    SCL_L
End Function

Public Sub MAck()
    SDA_L
    SCL_H
    SCL_L
```

```
End Sub

Public Sub MNAck()
   SCL_H
   SCL_L
End Sub

Public Function Read8Bit() As Byte
Dim Dat1 As Integer
Dim i As Integer
   For i = 7 To 0 Step -1
      If Rd_SDA Then    'Read SDA
         Dat1 = (2 ^ i) Or Dat1
      End If
      SCL_H
      SCL_L
   Next i
   Read8Bit = Dat1                  'Data 8 Bit
End Function

Public Sub Send8BIT(Data As Byte)
Dim i As Integer
   For i = 7 To 0 Step -1           ' Loop 7 Cycle
      If (Data And 2 ^ i) = 2 ^ i Then   'Test Bit 0 OR 1
         Call Send1
      Else
         Call Send0
      End If
   Next i
End Sub

Private Sub SDA_L()
```

```
  Out &H37A, Inp(&H37A) And &HFE   'SDA=0
End Sub

Private Sub SDA_H()
  Out &H37A, Inp(&H37A) Or 1    'SDA=1
End Sub

Private Sub SCL_L()
  Out &H37A, Inp(&H37A) And &HFD  'SCL=0
End Sub

Private Sub SCL_H()
  Out &H37A, Inp(&H37A) Or 2   'SCL=1
End Sub

Private Function Rd_SDA() As Boolean
  SDA_H
  Rd_SDA = (Inp(&H379) And &H80) <> &H80
End Function
```