การตรวจจับเลนถนนโดยคอมพิวเตอร์

LANE BOUNDARY DETECTION

โดย

นาย เกศ บัวลำใย

นางสาว ปรียาภรณ์ ภักดี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

การตรวจจับเลนถนนโดยคอมพิวเตอร์

LANE BOUNDARY DETECTION

โดย

นาย เกศ บัวลำใย รหัส 38014035

นางสาว ปรียาภรณ์ ภักดี รหัส 38014284

อาจารย์ที่ปรึกษา

ผศ. ดร. สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ ปีการศึกษา 2541

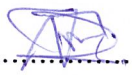
ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การตรวจจับเสนถนนโดยคอมพิวเตอร์

ผู้จัดทำ

1. นาย เกศ บัวลำใย
2. นางสาว ปรียาภรณ์ ภักดี



.....อาจารย์ที่ปรึกษา

(ผศ. ดร. สุรพันธ์ เอื้อไพบูลย์)

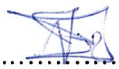
การตรวจจับเลนถนนโดยคอมพิวเตอร์

LANE BOUNDARY DETECTION

นาย เกศ บัวลำใย รหัส 38014035

นางสาว ปรียาภรณ์ ภักดี รหัส 38014284

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้



.....

(ผศ. ดร. สุรพันธ์ เอื้อไพบูลย์)

อาจารย์ที่ปรึกษา

การตรวจจับเลนถนนโดยคอมพิวเตอร์

นาย เกศ บัวลำใย
นางสาว ปรียาภรณ์ ภักดี
ผศ. ดร. สุรพันธ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ในการพัฒนาระบบการตรวจจับเลนถนนของระบบขับเคลื่อนใดๆ นับเป็นงานที่น่าสนใจแต่มีความยากยิ่งในการประมวลผลภาพ โครงงานนี้จึงเป็นเสมือนแบบจำลองการทำงานของระบบ โดยจะรับภาพจากกล้องดิจิตอลที่ติดตั้งอยู่บนรถ และนำภาพอย่างหยาบๆ ที่ได้มาทำการประมวลผลเพื่อให้คอมพิวเตอร์ได้ทราบว่าขณะนั้นระบบขับเคลื่อนของเราอยู่ในสถานการณ์เช่นไร จากนั้นส่งผลการประมวลผลที่ได้ไปยังไมโครคอนโทรลเลอร์ ซึ่งการขับเคลื่อนของรถจะอาศัยการควบคุมความเร็วดีซีมอเตอร์ ทำให้รถสามารถเคลื่อนที่ไปได้อัตโนมัติ พร้อมทั้งตรวจจับเลนของถนนไปในขณะเดียวกัน

LANE BOUNDARY DETECTION

Kate Bualamyai

Preeyaporn Pakdee

Dr. Surapan Aopaiboon Advisor

1998

ABSTRACT

In the developing of a vision – based on road detection system is very interesting to cope with but we are facing the difficulty in a process to give the picture out.This project use as a model to display system's function by taking the image from digital camera on vehicle.Then take the image that assumed to contain distorted version of giver template to a process system so that a computer acknowledge situation.The vehicle movement is controlled by DC motor.So the vehicle can move automatically and can detect the lane simultaneously.

สารบัญ

หน้า

สารบัญรูปภาพ

บทที่ 4 **หน้า**

บทที่ 5

หน้า

สารบัญตาราง

บทที่ 1

บทนำ

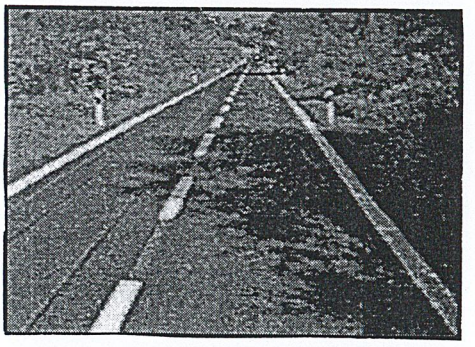
1.1 ความเป็นมาและความสำคัญของโครงงาน

เนื่องจากการศึกษาการตรวจขับเลนถนน เป็นเทคโนโลยีที่อาจจะเรียกได้ว่าใหม่สำหรับการขับเคลื่อนยานพาหนะระบบอัตโนมัติ ซึ่งปัจจุบันมีนักวิจัยหลายท่านกำลังศึกษาถึงแนวทางที่ดีที่สุด ซึ่งในทางคอมพิวเตอร์ก็คือการใช้เวลาในการประมวลผลให้น้อยที่สุด ซึ่งส่งผลให้สามารถตอบสนองต่อสภาวะต่างๆ ได้ทันท่วงที ในปัจจุบันได้มีการทดลองควบคุมการขับเคลื่อนยานพาหนะโดยใช้การประมวลผลของคอมพิวเตอร์ เพื่อตรวจดูทัศนะวิสัยต่างๆ ซึ่งเกินกว่าความสามารถของคอมพิวเตอร์จะทำได้ โดยต้องอาศัยกล้องที่มีประสิทธิภาพสูงและคอมพิวเตอร์ต้องมีความเร็วในการประมวลผลสูง ซึ่งในการประมวลผลภาพจะมีรายละเอียดขั้นตอนดังต่อไปนี้

1.) การสร้างตัวกำหนดตำแหน่งของเลนถนน เพื่อใช้ในการกำหนดขอบเขตของเลนถนนโดยปราศจากสัญญาณรบกวนหรือวัตถุอื่นบนถนน

2.) การแบ่งส่วนย่อยต่างๆ ของภาพที่ได้จากเลนถนน

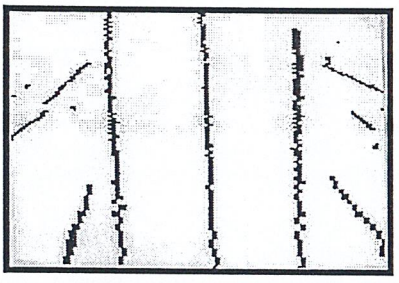
3.) ขั้นตอนการประมาณรูปแบบของเลนถนน ประกอบด้วย พิกัดของจุดกึ่งกลาง ความโค้งของถนน จำนวนของเลนถนน และตำแหน่งของรถยนต์บนเลนถนน



รูปที่ 1.1 เลนถนนที่จะทำการตรวจจับ

1.2 การหาขอบเขตของถนนในเวลาจริง

เราจะวิเคราะห์ส่วนที่ใช้ในการหาขอบเขตของถนนในเวลาจริง การหาทิศทางของถนน และตำแหน่งของยานพาหนะ เพื่อนำไปใช้เป็นข้อมูลในการขับมอเตอร์ ในการหาข้อมูลในเวลาจริงเราจะสนใจเฉพาะสภาพแวดล้อมของถนน และจะประมวลผลโดยใช้ภาพขาวดำ



รูปที่ 1.2 เลนถนนที่ผ่านการดีเทคและจะนำไปประมวลผล

1.3 ตัวกำหนดตำแหน่งของเลนถนน

จากข้างต้นเราจะพบว่ามีสิ่งแวดล้อมต่างๆ เข้ามาเกี่ยวข้องในภาพของเลนถนน เช่น ปริมาณของแสง และสิ่งก่อสร้างต่างๆ เราจึงต้องพัฒนาระบบที่จะหาตำแหน่งของถนนในสภาวะที่ต่างกันได้ ซึ่งเมื่อเรานำภาพผ่านตัวกรองความถี่ต่ำผ่าน จะได้เซ็ทของจุดต่างๆ ที่ขึ้นกับขอบถนน ตัวกำหนดตำแหน่งของเลนถนนจะปรากฏในรูปจุดสองจุดที่มีเกรเดียนท์สูงมาก คือ จุดทั้งสองแทนการเปลี่ยนจากบริเวณมืดเป็นบริเวณสว่าง

รูปที่ 1.3 ภาพที่นำเข้าไปเป็นอินพุทและเอาท์พุทที่ได้จากการประมวลผล

บทที่2
ระบบขับเคลื่อน (Driving System)

ระบบขับเคลื่อนในโครงงานนี้จะใช้มอเตอร์กระสตรง ซึ่งจะเปลี่ยนความเร็วโดยใช้การเปลี่ยนความกว้างพัลส์ของกระแสที่ใช้ขับมอเตอร์ โดยค่าความเร็วที่ต้องการนั้นจะควบคุมโดยไมโครคอนโทรลเลอร์ ซึ่งจะส่งค่าความเร็วที่ต้องการไปยังวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก แล้วส่งค่าแรงดันอะนาลอกที่ได้ไปยังพัลส์วิธมอดูเลเตอร์เพื่อควบคุมความกว้างพัลส์ โดยหลักการทำงานของมอเตอร์กระแสตรงและวงจรควบคุมความเร็วเป็นดังนี้

2.1 หลักการทำงานของมอเตอร์กระแสตรง

มอเตอร์กระแสตรงเป็นเครื่องจักรกลไฟฟ้าที่ทำหน้าที่แปลงพลังงานไฟฟ้าเป็นแรงบิดทางกล โดยแรงบิดของแกนมอเตอร์จะขึ้นอยู่กับค่ากระแสในขดลวดอาเมเจอร์และเส้นแรงแม่เหล็กที่ตัดผ่านขดลวดอาเมเจอร์ดังแสดงในสมการ 2.1 โดยทิศทางของกระแส แรงบิด และสนามแม่เหล็กแสดงดังรูปที่ 2.1

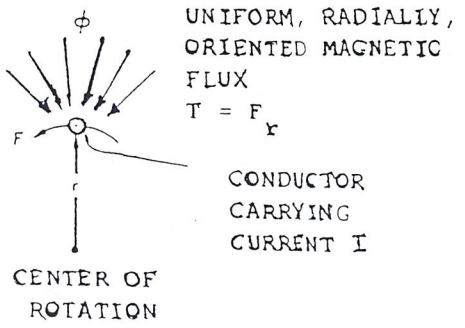
$$T = K \varnothing I \qquad (2.1)$$

เมื่อ T คือ แรงบิดที่แกนของมอเตอร์มีหน่วยเป็นนิวตันเมตร (N-m)

K คือ ค่าคงที่ทางกายภาพของมอเตอร์ (Sec^2/m)

$\varnothing$ คือ เส้นแรงแม่เหล็กที่ตัดผ่านขดลวดอาเมเจอร์มีหน่วยเป็นเวเบอร์ (Weber)

I คือ กระแสในขดลวดอาเมเจอร์มีหน่วงเป็นแอมแปร์ (Ampere)



รูปที่ 2.1 แสดงการเกิดแรงบิดในมอเตอร์กระแสตรง

และเมื่อขดลวดตัวนำเคลื่อนที่ในสนามแม่เหล็กจะทำให้เกิดแรงดันตกคร่อมขดลวด ซึ่งมีค่าขึ้นอยู่กับความเร็วของมอเตอร์และมีทิศทางต้านการไหลของกระแส โดยมีความสัมพันธ์ดังแสดงในสมการ 2.2

$$E = K \phi W \qquad (2.2)$$

เมื่อ E คือ แรงดันย้อนกลับมีหน่วยเป็นโวลท์ (Volt)

W คือ ความเร็วของมอเตอร์มีหน่วยเป็นเรเดียนต่อวินาที (rad/sec)

2.1.1 การแยกประเภทของมอเตอร์กระแสตรง

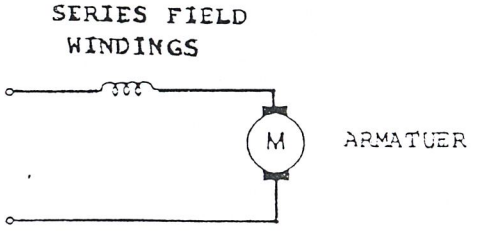
มอเตอร์กระแสตรงสามารถแบ่งได้หลายประเภทตามลักษณะการสร้างสนามแม่เหล็กและโครงสร้างของอาเมเจอร์ โดยเมื่อแบ่งตามลักษณะการสร้างสนามแม่เหล็กสามารถแบ่งได้สองประเภท คือ

1) มอเตอร์กระแสตรงแบบปรับเส้นแรงแม่เหล็กได้

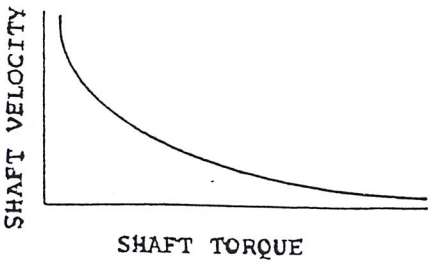
มอเตอร์กระแสตรงแบบปรับเส้นแรงแม่เหล็กได้สามารถแบ่งได้สองประเภท คือ

- แบบขดลวดสนามแม่เหล็กต่ออนุกรมกับขดลวดอาเมเจอร์

มีโครงสร้างดังรูปที่ 2.2 มอเตอร์แบบนี้มีเส้นแรงแม่เหล็กเป็นสัดส่วนกับกระแส จึงสามารถปรับค่าเส้นแรงแม่เหล็กได้ โดยมีความสัมพันธ์ในลักษณะที่ไม่เป็นเชิงเส้นดังรูปที่ 2.3 มอเตอร์ชนิดนี้เหมาะสำหรับงานที่ต้องการแรงบิดต่ำความเร็วสูงหรือความเร็วต่ำแรงบิดสูง



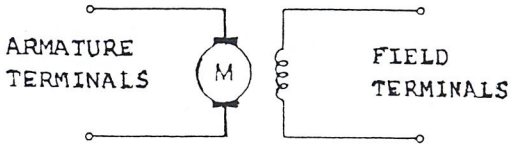
รูปที่ 2.2 มอเตอร์กระแสตรงแบบอาเมเจอร์ต่ออนุกรมกับสนามแม่เหล็ก



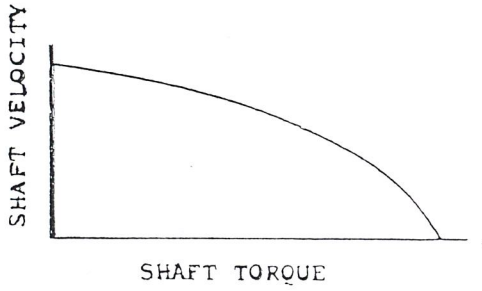
รูปที่ 2.3 ความสัมพันธ์ระหว่างความเร็วและแรงบิดของมอเตอร์อนุกรมที่แรงดันคงที่

- มอเตอร์กระแสตรงแบบสนามแม่เหล็กแยกกระตุ้น

นิยมเรียกอีกชื่อว่าชันท์มอเตอร์(shunt motor) มอเตอร์แบบนี้สามารถสามารถปรับเส้นแรงแม่เหล็กได้โดยอิสระไม่ขึ้นกับค่ากระแสในอาเมเจอร์ มักจะใช้ในงานที่ต้องการแรงบิดสูง โครงสร้างของมอเตอร์ชนิดนี้แสดงดังรูปที่ 2.4 ความสัมพันธ์ระหว่างแรงบิดกับความเร็วของชันท์มอเตอร์ภายใต้สนามแม่เหล็กและแรงดันคงที่แสดงดังรูปที่ 2.5



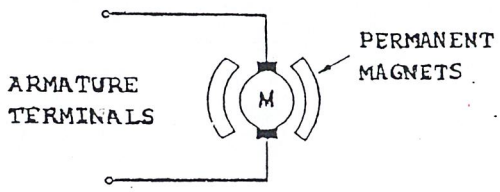
รูปที่ 2.4 โครงสร้างของมอเตอร์กระแสตรงแบบสนามแม่เหล็กแยกกระตุ้น



รูปที่ 2.5 ความสัมพันธ์ระหว่างความเร็วและแรงบิดของชันท์มอเตอร์ที่สนามแม่เหล็กและแรงดันคงที่

2) มอเตอร์กระแสตรงแบบสนามแม่เหล็กคงที่

ระบบการกระตุ้นฟิลด์ของสนามแม่เหล็กโดยทั่วไป มักใช้แม่เหล็กถาวรดังรูปที่ 2.6



รูปที่ 2.6 มอเตอร์กระแสตรงแบบฟิลด์แม่เหล็กถาวร

ในมอเตอร์กระแสตรงแบบนี้สนามแม่เหล็กจะมีค่าคงที่ ดังนั้นสามารถเขียนความสัมพันธ์ของแรงบิดและแรงดันย้อนกลับได้ดังนี้

$$T = K_t I \qquad (2.3)$$

$$E = K_e W \qquad (2.4)$$

โดยสมการทางไฟฟ้าของมอเตอร์กระแสตรงแบบสนามแม่เหล็กคงที่เป็นดังนี้

$$V = K_e W + L\, di/dt + iR \qquad (2.5)$$

เมื่อ V คือ แรงดันที่ป้อนให้มอเตอร์มีหน่วยเป็นโวลท์ (Volt)

K_e คือ ค่าคงที่แรงดันย้อนกลับ

L คือ ค่าความเหนี่ยวนำของมอเตอร์มีหน่วยเป็นเฮนรี่ (Henry)

R คือ ความต้านทานของขั้วมอเตอร์มีหน่วยเป็นโอห์ม (Ohm)

$$T_g = J\, dw/dt + BW + T_f + T_l \qquad (2.6)$$

เมื่อ T_g คือ แรงบิดที่เกิดจากอาร์เมเจอร์มีหน่วยเป็นนิวตัน-เมตร (N-m)

J คือ ผลรวมของโมเมนต์ความเฉื่อยของมอเตอร์และโหลด

W คือ ความเร็วการหมุนของมอเตอร์มีหน่วยเป็นเรเดียนต่อวินาที (Rad/Sec)

B คือ สัมประสิทธิ์ของวิสคอสแดมพ์ปิ้ง

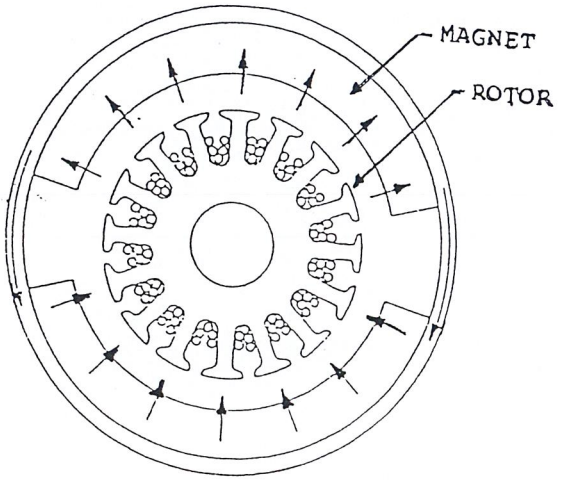
T_f คือ แรงบิดเสียดทานมีหน่วยเป็นนิวตัน-เมตร (N-m)

T_l คือ แรงบิดที่โหลดมีหน่วยเป็นนิวตัน-เมตร (N-m)

และเมื่อเราแบ่งมอเตอร์กระแสตรงออกตามลักษณะโครงสร้างของอาเมเจอร์ เราจะสามารถแบ่งได้ดังนี้

1) มอเตอร์กระแสตรงแบบอาเมเจอร์มีขดลวดพันอยู่บนพื้นผิว

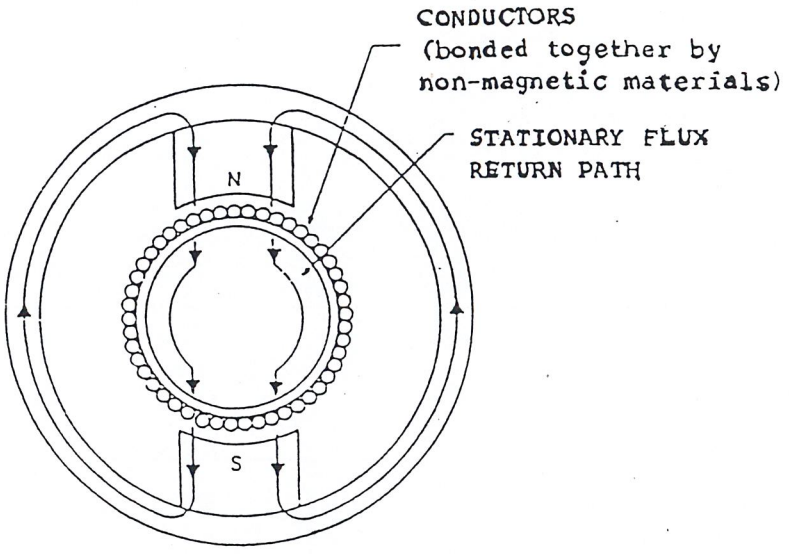
มีโครงสร้างดังรูปที่ 2.7 ซึ่งโครงสร้างแบบนี้มีค่าความเหนี่ยวนำต่ำ ทำให้มอเตอร์แบบนี้มีขนาดใหญ่และราคาแพง



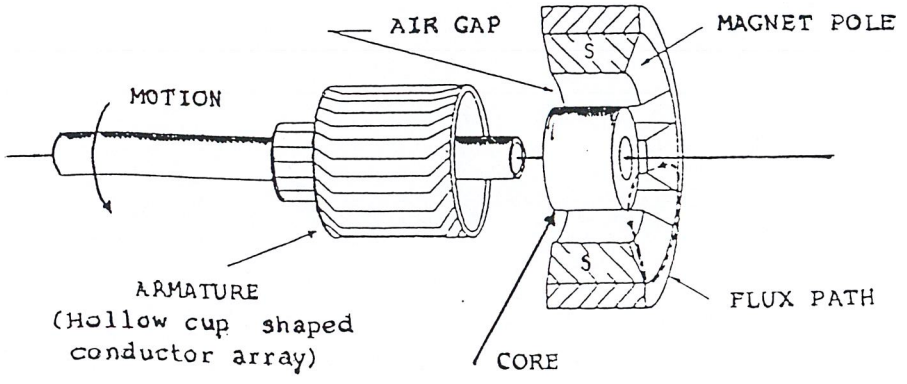
รูปที่ 2.7 แสดงรูปหน้าตัดของมอเตอร์กระแสตรงแบบอาเมเจอร์มีขดลวดพันอยู่บนพื้นผิวและฟิลด์เป็นแม่เหล็กถาวร

2) มอเตอร์กระแสตรงแบบอาเมเจอร์เป็นขดลวดหมุน

มอเตอร์กระแสตรงแบบนี้ได้รับการออกแบบให้โมเมนต์ของแรงเฉื่อยต่ำ มีโครงสร้างแสดงดังรูปที่ 2.8 และ 2.9 มอเตอร์กระแสตรงแบบนี้มีช่องว่างอากาศมากกว่ามอเตอร์กระแสตรงแบบอื่นๆ ดังนั้นจึงต้องออกแบบให้โครงสร้างของแม่เหล็กใหญ่ขึ้น จึงมีราคาแพงมาก นอกจากนี้ยังมีความจุความร้อนต่ำ ทำให้มอเตอร์เสียได้ง่ายและยังมีค่าความเหนี่ยวนำต่ำมากอีกด้วย



รูปที่ 2.8 หน้าตัดของมอเตอร์กระแสตรงแบบอาเมเจอร์เป็นขดลวดหมุน



รูปที่ 2.9 รูปหน้าตัดด้านข้างของมอเตอร์กระแสตรงแบบอาเมเจอร์เป็นขดลวดหมุน

นอกจากนี้ยังมีมอเตอร์กระแสตรงที่มีโครงสร้างแบบอื่นๆ อีกซึ่งไม่ขอกล่าวถึงในที่นี้

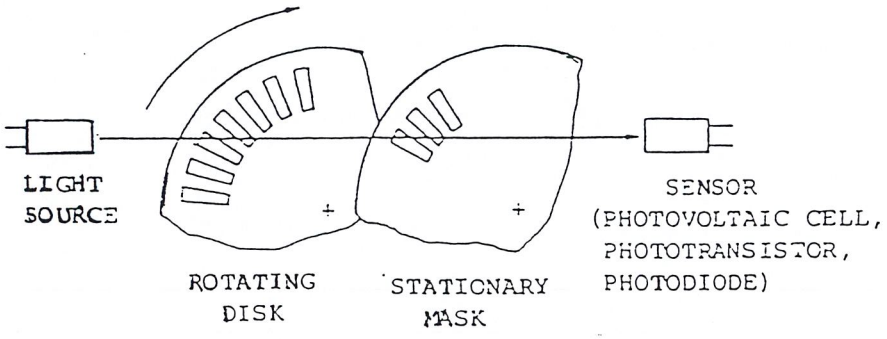
2.2 พัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์ (Pulse Width Modulation Amplifier)

ระบบที่ใช้ควบคุมความเร็วโดยทั่วไป ได้แก่ ระบบดีซีลิเนียร์เซอร์โวแอมปลิฟายเออร์ สร้างขึ้นเพื่อใช้เป็นอุปกรณ์ขยายแบบลิเนียร์ มีหน้าที่ควบคุมกระแสและแรงดันที่ใช้ในการขับเคลื่อนมอเตอร์เพื่อให้ได้ค่าความเร็วที่ต้องการ และเนื่องจากการควบคุมนี้ทำโดยการเปลี่ยนความกว้างพัลส์ของแรงดันที่ใช้ขับมอเตอร์ ดังนั้นส่วนแอมปลิฟายเออร์จะทำการลดแรงดัน ซึ่งเท่ากับผลต่างของแรงดันและแรงดันที่มอเตอร์ใช้งานจริง

2.2.1 การทำงานของพัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์

พัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์สามารถแบ่งได้เป็นสามชนิดตามลักษณะของการทำงาน คือ ไบโพล่าร์ ยูนิโพล่าร์ และลิมิตยูนิโพล่าร์ ซึ่งมีโครงสร้างดังรูปที่ 2.10 โดยมีความถี่ในการสวิทช์เป็น fs และ ton ที่เกิดในช่วงแรก ส่วน toff เกิดในช่วงหลัง โดย

ton เมื่อ 0 < t < t1 , toff เมื่อ t1 < t < tf



รูปที่ 2.10 พัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์และมอเตอร์กระแสตรง

แบบไบโพล่าร์จะมี T1 และ T4 นำกระแสระหว่างเฟสออน ส่วน T2 และ T3 นำกระแสระหว่างเฟสออฟ ซึ่งจะได้แรงดันตกคร่อมมอเตอร์เป็น

$$Vm = Vab \begin{cases} Vg : 0 \le t \le t1 \\ -Vg : t1 \le t \le tf \end{cases} \qquad (2.7)$$

เมื่อ Vm คือ แรงดันตกคร่อมมอเตอร์มีหน่วยเป็นโวลท์ (Volt)

แบบยูนิโพลาร์จะลดจำนวนทรานซิสเตอร์ในการสวิทช์ลง การสวิทช์จะขึ้นอยู่กับว่า Vin มีค่าเป็นบวกหรือลบ เมื่อ Vin เป็นบวก T4 จะนำกระแสตลอดคาบ ในขณะที่ T1 นำกระแสในช่วง

เฟสออน และ T2 นำกระแสในช่วงเฟสออฟ เมื่อ Vin เป็นลบ T2 จะนำกระแสตลอดคาบ โดยมี T3 และ T4 สลับกันทำงาน เมื่อ Vin เป็นบวกได้

$$\text{Vin} \left\{ \begin{array}{ll} \text{Vg} : 0 < t < t1 \\ 0 \quad : t1 < t < tf \end{array} \right. \qquad (2.8)$$

และเมื่อ Vin ปันลบจะต่างกันที่เครื่องหมายของ Vin เท่านั้น

จากลักษณะที่กล่าวมาทั้งสองแบบนั้นมีประโยชน์เหมือนกัน ซึ่งในแต่ละกรณีจะมีทรานซิสเตอร์คู่หนึ่งนำกระแสในขณะที่อีกคู่หนึ่งหยุดนำกระแส ซึ่งจะมีเวลาสะสมและปล่อยออกของทรานซิสเตอร์เกิดขึ้น และอาจทำให้ทรานซิสเตอร์ทั้งหมดนำกระแสในเวลาเดียวกัน ซึ่งทำให้แหล่งจ่ายไฟลัดวงจร เราจำเป็นต้องหลีกเลี่ยงสภาวะดังกล่าว โดยสร้างช่วงดีเลย์ระหว่างการหยุดและนำกระแสของทรานซิสเตอร์

แบบลิมิตยูนิโพล่าร์มีความจำเป็นต้องมีช่วงดีเลย์ ซึ่งการสวิทช์ขึ้นกับค่า Vin เป็นบวกหรือลบ เมื่อ Vin เป็นบวก T4 จะนำกระแสตลอดคาบ ในขณะที่ T1 นำกระแสในช่วงเฟสออน ทั้ง T1 และ T4 จะนำกระแสส่งผลให้แรงดันตกคร่อมมอเตอร์ คือ

$$\text{Vm} = \text{Vg} \text{ เมื่อ } 0 \leq t \leq t1$$

ระหว่างเฟสออฟจะมีกระแสเพียงตัวเดียวส่งผลให้ Vin ขึ้นอยู่กับ Iab ตราบใดที่ Iab > 0 ซึ่งเป็นสภาวะปกติ เมื่อ Vab > 0 กระแส Iab จะไหลผ่าน D2 และ T4 เป็นผลให้ Va = 0 และ

$$\text{Vm} = \text{Vab} = 0 \left\{ \begin{array}{ll} t1 \leq t \leq tf \\ \text{Iab} > 0 \end{array} \right. \qquad (2.9)$$

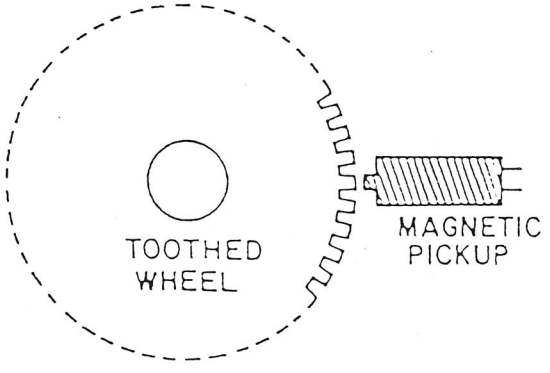
ซึ่งจะเกิดขึ้นภายหลังเปลี่ยนขั้ว Vin ในที่สุดถ้าเราสามารถทำให้ Iab = 0 ส่งผลให้ทั้ง D1 และ D4 ไม่นำกระแสและแรงดัน Vin จะอยู่ระหว่าง 0 และ Vg ดังนี้

$$\left.\begin{array}{l} 0 < Vin < Vg \\ \\ \\ \end{array}\right\} \begin{array}{l} t1 \le t < tf \\ \\ Iab = 0 \end{array} \qquad (2.10)$$

อย่างไรก็ตามถ้า Iab > 0 เป็นสภาวะปกติ เมื่อ Vin > 0 แบบยูนิโพล่าร์และแบบลิมิตยูนิโพล่าร์จะแสดงคุณสมบัติคล้ายคลึงกันมาก

2.2.2 กำลังงานสูญเสียที่ตัวมอเตอร์

ธรรมชาติของพัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์ซึ่งถูกใช้กับมอเตอร์กระแสตรงจะมีคุณสมบัติดังรูปที่ 2.11



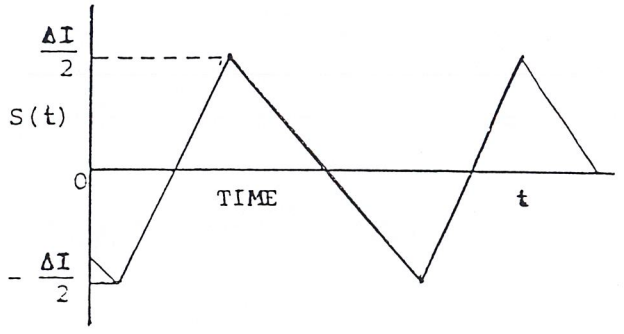
รูปที่ 2.11 ความสัมพันธ์ระหว่างความเร็วและแรงบิด

ปัญหาแรกในการพิจารณาที่เกี่ยวข้องกับการสูญเสียกำลังงานของแอมปลิฟายเออร์ จะแสดงโดยเปรียบเทียบระหว่างพัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์กับลิเนียร์มอดูเลชั่นแอมปลิฟายเออร์ การสูญเสียกำลังงานในตัวมอเตอร์ที่มีค่าความต้านทานของขดลวดอาเมเจอร์ (R) แรงบิดเสียดทานภายในมอเตอร์ (Tf) และวิสคอสแดมพ์ปิ้งแฟคเตอร์ (D)

$$P = I^2(t)R + \frac{K_e T_f W}{K_t} + \frac{K_e D W^2}{K_t} \qquad (2.11)$$

ถ้าเราให้มอเตอร์ถูกขับที่ความเร็ว W ทำโดยใช้ลิเนียร์แอมปลิฟายเออร์ซึ่งมีกระแสไหลในขดลวดอาเมเจอร์ I (t) และการสญเสียกำลังงาน P1 ถ้าใช้พัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์ขับมอเตอร์ให้มีความเร็วเดียวกันนี้ กระแสที่ต้องการ คือ Ip(t) ซึ่งเป็นผลรวมระหว่าง I(t) และค่ายอดของลูกคลื่นสามเหลี่ยม S(t) ซึ่งแสดงดังรูปที่ 2.12 คือ

$$Ip(t) = I(t) + S(t) \qquad (2.12)$$



รูปที่ 2.12 ลูกคลื่นสามเหลี่ยมของกระแส

ค่าการสูญเสียกำลังงานของลิเนียร์แอมปลิฟายเออร์ คือ

$$P_L = I^2(t)R + \frac{K_e T_f W}{K_t} + \frac{K_e D W^2}{K_t} \qquad (2.13)$$

Plp หาจากการแทนสมการ 2.13 ลงในสมการ 2.12 สังเกตว่าเทอมแรกเท่านั้นที่ขึ้นอยู่กับ $RI_p(t)$ คือ

$$I^2_p(t) = R[\, I(t) + S(t)\,]^2 = RI^2(t) + 2RI(t)S(t) + RS^2(t) \qquad (2.14)$$

เทอมที่สองทางขวามือของสมการ 2.17 เป็นผลคูณของ S(t) ซึ่งมีค่าเฉลี่ยเป็นศูนย์กับ

I(t) ถ้าค่าทั้งสองไม่สัมพันธ์กันดังปรากฏในกรณีส่วนใหญ่ ผลคูณที่ได้จะมีค่าเฉลี่ยเป็นศูนย์เช่นกัน เทอมสุดท้ายของสมการ 2.17 สามารถหาค่า S(t) ที่ให้มาในรูปที่ 2.12 และจะสามารถหาค่าเฉลี่ยได้เป็น

$$RS^2(t) = RI^2(t) \qquad (2.15)$$

ต่อไปนี้เมื่อพิจารณาค่าความแตกต่างเฉลี่ยในการสูญเสียพลังงานจากสมการ 2.15 ,2.16 ,2.17 และ2.18 คือ

$$(PLp - PL)ave = R[I^2P(t) - I^2(t)]ave \qquad (2.16)$$

ซึ่งถ้าทำการทดลองดูจะพบว่าพัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์มีการสูญเสียกำลังงานในมอเตอร์น้อยมาก

2.2.3 การเลือกความถี่ของการสวิทช์

การเลือกความถี่ของการสวิทช์ fs สามารถทำได้โดยใช้หลักการดังต่อไปนี้

1) ความถี่ของการสวิทช์ต้องมีค่าสูงพอที่จะทำให้แอดมิดแตนซ์ของอาเมเจอร์มีค่าสูงกว่าความต้านทานของความถี่ที่ถูกเลือก เพื่อเป็นการกำจัดการเปลี่ยนแปลงของกระแส

$$2\pi f_s L >> R \qquad (2.17)$$

เมื่อ fs คือ ความถี่ในการสวิทช์มีหน่วยเป็นเฮิร์ทซ (Hz)

L คือ ความเหนี่ยวนำของขดลวดอาเมเจอร์มีหน่วยเป็นเฮนรี่ (Henry)

R คือ ความต้านทานของขดลวดอาเมเจอร์มีหน่วยเป็นโอห์ม (Ohm)

2) ความถี่ของการสวิทช์ต้องสูงพอที่จะทำให้ระบบเซอร์โวไม่มีผลตอบสนอง ดังนั้น

$$f_s > 10\, f_{BW} \qquad (2.18)$$

3) fs ต้องมากกว่าความถี่เรโซแนนต์ของลูป

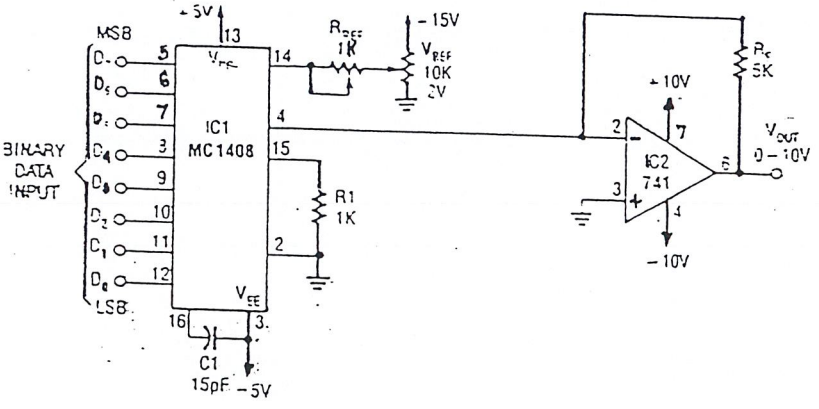
$$fs > (fres)max \qquad (2.19)$$

ตามภาวะข้างต้นพบว่าต้องเพิ่มค่า fs แต่การเพิ่มค่า fs ก็ทำให้เกิดผลเสียบางประการ

4) การเพิ่มค่าความถี่การสวิทช์ทำให้เกิดกำลังงานสูญเสียในเอาท์พุททรานซิสเตอร์มากขึ้น ดังนั้นต้องทำให้ fs ต่ำลงโดยยังสอดคล้องกับสภาวะที่ผ่านมาด้วย

5) ดีเลย์เฟสที่ต้องการในไบโพล่าร์และยูนิโพล่าร์แอมปลิฟายเออร์เพื่อป้องกันการลัดกระแสของทรานซิสเตอร์ ส่งผลให้เกิดขีดจำกัดในการเลือกความถี่

2.3 การใช้งาน Digital to Analog Converter (DAC)



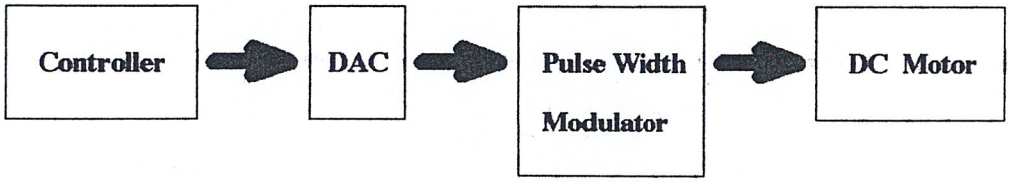
รูปที่ 2.13 วงจร Unipolar D/A

รูปที่ 2.13 แสดงการประยุกต์ใช้งาน DAC อย่างง่ายๆ โดยใช้ DAC เบอร์ MC1408 ของบริษัทโมโตโรล่า การทำงานของวงจรตรงไปตรงมาไม่ซับซ้อน ระดับสัญญาณ TTL ขนาด 8 บิทซึ่งเป็นเลขฐานสองป้อนให้กับอินพุท D0 – D7 ของ DAC ใช้เวลาในการแปลงสัญญาณประมาณ 300 นาโนวินาที ซึ่งเป็นช่วงเวลา settle time ของ MC1408 ภายใน MC1408 ใช้ไบนารี่แลดเดอร์ในการสวิทช์และแปลงสัญญาณให้เป็นกระแสไฟฟ้าออกไปทางเอาท์พุท ซึ่งถูกต่อไว้ด้วยออปแอมป์เพื่อแปลงกระแสให้เป็นแรงดันเอาท์พุท

ความละเอียดของวงจรนี้มีขนาดแปดบิท หมายถึง แรงดันเอาท์พุทสามารถเปลี่ยนได้ตั้งแต่ 0 ถึง 10 โวลท์ โดยมีการเปลี่ยนแปลงได้ 256 ขั้นๆ ละประมาณ 0.039 โวลท์ รหัสฐานสิบหก 00H จะให้แรงดันเอาท์พุท 0 โวลท์ ครึ่งหนึ่งของอินพุทสูงสุด คือ 7FH ให้แรงดันเอาท์พุท 5 โวลท์ และเมื่ออินพุทเป็น FFH จะให้แรงดันเอาท์พุท 10 โวลท์

วงจรในรูปที่ 2.13 สามารถนำไปใช้ได้กับทุกวงจรที่ต้องการ DAC เพื่ออินเตอร์เฟสวงจรอนาลอกกับดิจิตอลเข้าด้วยกัน โดยอินพุทจะนำมาจากวงจรดิจิตอลแล้วนำเอาท์พุทที่ได้ไปใช้กับวงจรอนาลอกต่อไป

2.4 หลักการออกแบบและสร้างระบบขับเคลื่อน



รูปที่ 2.14 บล๊อคไดอะแกรมของระบบขับเคลื่อน

ในระบบขับเคลื่อนได้ใช้พัลส์วิธมอดูเลชั่นแอมปลิฟายเออร์ในการขับเคลื่อนมอเตอร์กระแสตรง โดยมีหลักการทำงาน คือ ไมโครคอนโทรลเลอร์จะส่งค่าความเร็วเป็นเลขฐานสองขนาดสี่บิทออกมายังดิจิทอลทูอนาลอกคอนเวอร์เตอร์ (MC1408) ซึ่งทำหน้าที่แปลงค่าความเร็วซึ่งเป็นเลขฐานสองให้เป็นแรงดันไฟตรงผ่านทางเอาท์พุทของออปแอมป์ และนำไปป้อนให้กับขา Dead time control ของพัลส์วิธมอดูเลเตอร์ (TL494) โดยความกว้างของเอาท์พุทพัลส์จะเปลี่ยนไปตามค่าแรงดันที่ขา Dead time control ซึ่งเอาท์พุทพัลส์นี้จะนำไปใช้ขับทรานซิสเตอร์ซึ่งต่อแบบยูนิโพล่าร์ เพื่อขับกระแสให้มอเตอร์กระแสตรง

บทที่ 3

ส่วนติดต่อระหว่างฮาร์ดแวร์ภายนอกกับคอมพิวเตอร์

ในการควบคุมอุปกรณ์สนามไฟฟ้าโดยใช้เครื่องคอมพิวเตอร์นั้น เราจะควบคุมผ่านทางพอร์ท ซึ่งในโครงงานนี้เราจะใช้การ์ดพอร์ท8255ในการติดต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอก เพื่อใช้ในการควบคุมอุปกรณ์สนามไฟฟ้าต่อไป ซึ่งการเชื่อมต่อและการทำงานของการ์ดพอร์ท 8255มีรายละเอียดดังนี้

3.1 การเชื่อมต่อการ์ดพอร์ท8255กับคอมพิวเตอร์

ในการเชื่อมต่อการ์ดพอร์ท8255เข้ากับเครื่องคอมพิวเตอร์นั้น เราจะต่อผ่านทางสล๊อท62 ขา ซึ่งมีตำแหน่งของขาและการนับขาดังตารางที่ 3.1 และรูปที่ 3.1 ตามลำดับ

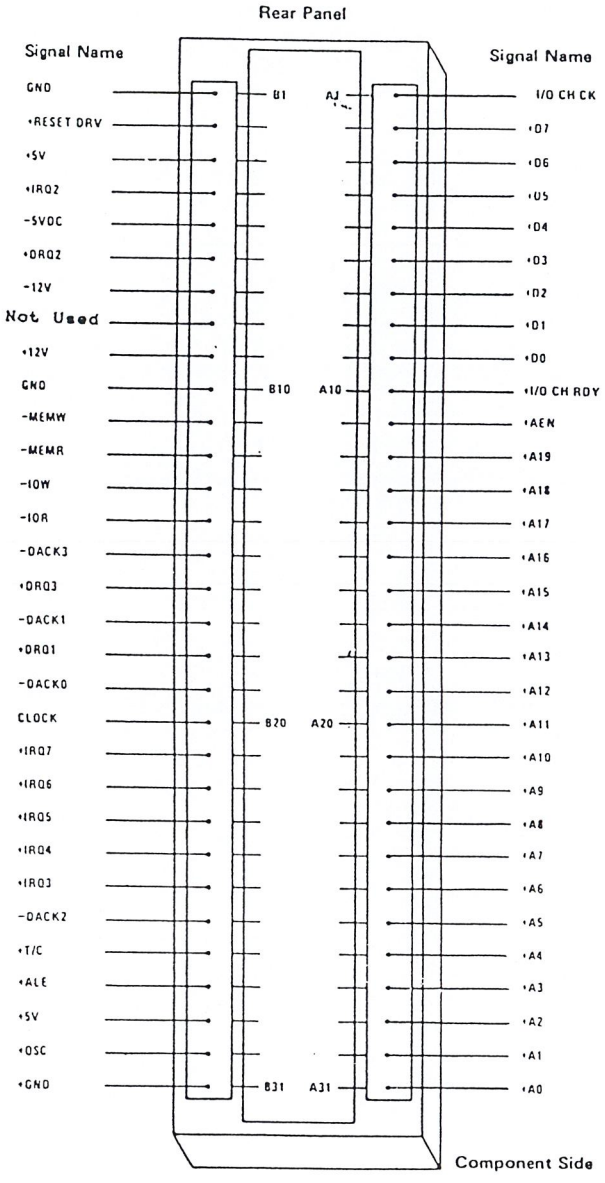
ตำแหน่ง	สัญญาณ	อินพุท/เอาท์พุท
A1	I/O CH CK	อินพุท
A2	D7	อินพุท/เอาท์พุท
A3	D6	อินพุท/เอาท์พุท
A4	D5	อินพุท/เอาท์พุท
A5	D4	อินพุท/เอาท์พุท
A6	D3	อินพุท/เอาท์พุท
A7	D2	อินพุท/เอาท์พุท
A8	D1	อินพุท/เอาท์พุท
A9	D0	อินพุท/เอาท์พุท
A10	I/O CH RDY	อินพุท
A11	AEN	เอาท์พุท
A12	A19	อินพุท/เอาท์พุท
A13	A18	อินพุท/เอาท์พุท
A14	A17	อินพุท/เอาท์พุท

ตำแหน่ง	สัญญาณ	อินพุท/เอาท์พุท
A15	A16	อินพุท
A16	A15	อินพุท/เอาท์พุท
A17	A14	อินพุท/เอาท์พุท
A18	A13	อินพุท/เอาท์พุท
A19	A12	อินพุท/เอาท์พุท
A20	A11	อินพุท/เอาท์พุท
A21	A10	อินพุท/เอาท์พุท
A22	A9	อินพุท/เอาท์พุท
A23	A8	อินพุท/เอาท์พุท
A24	A7	อินพุท/เอาท์พุท
A25	A6	อินพุท/เอาท์พุท
A26	A5	อินพุท/เอาท์พุท
A27	A4	อินพุท/เอาท์พุท
A28	A3	อินพุท/เอาท์พุท
A29	A2	อินพุท/เอาท์พุท
A30	A1	อินพุท/เอาท์พุท
A31	A0	อินพุท/เอาท์พุท
B1	GND	กราวนด์
B2	RESET DRV	เอาท์พุท
B3	+5V	แหล่งจ่ายไฟเลี้ยง
B4	IRQ2	อินพุท
B5	-5VDC	แหล่งจ่ายไฟเลี้ยง
B6	DRQ2	อินพุท
B7	-12V	แหล่งจ่ายไฟเลี้ยง
B8	NOT USED	อินพุท
B9	+12V	แหล่งจ่ายไฟเลี้ยง

ตำแหน่ง	สัญญาณ	อินพุท/เอาท์พุท
B10	GND	กราวนด์
B11	-MEMW	เอาท์พุท
B12	-MEMR	เอาท์พุท
B13	-IOW	อินพุท/เอาท์พุท
B14	-IOR	อินพุท/เอาท์พุท
B15	-DACK3	เอาท์พุท
B16	DRQ3	อินพุท
B17	-DACK1	เอาท์พุท
B18	DRQ1	อินพุท
B19	-DACK0	อินพุท/เอาท์พุท
B20	CLOCK	เอาท์พุท
B21	IRQ7	อินพุท
B22	IRQ6	อินพุท
B23	IRQ5	อินพุท
B24	IRQ4	อินพุท
B25	IRQ3	อินพุท
B26	-DACK2	เอาท์พุท
B27	T/C	เอาท์พุท
B28	ALE	เอาท์พุท
B29	+5V	แหล่งจ่ายไฟเลี้ยง
B30	OSC	เอาท์พุท
B31	GND	กราวนด์

ตารางที่ 3.1 แสดงหน้าที่ของแต่ละขาบนสล๊อท62ขา

ด้านหลังเครื่อง



รูปที่ 3.1 แสดงการนับขาของสล๊อทแบบ62ขา

โดยสัญญาณที่เราจะนำมาใช้กับการ์ดพอร์ท8255 ได้แก่ A0 – A11 เป็นแอดเดรสของระบบที่ใช้ติดต่อกับหน่วยความจำและอุปกรณ์อินพุท/เอาท์พุท D0 – D7 เป็นสัญญาณข้อมูลขนาด8 บิทที่ใช้ติดต่อกับหน่วยความจำของไมโครโปรเซสเซอร์

$\overline{\text{IORQ}}$ เป็นสัญญาณอ่านอินพุท/เอาท์พุท เป็นสัญญาณที่ส่งมาจากCPU แอกทีฟที่0

$\overline{\text{IOW}}$ เป็นสัญญาณเขียนข้อมูลลงบนอุปกรณ์อินพุท/เอาท์พุท สัญญาณนี้ถูกควบคุมจากอุปกรณ์อินพุท/เอาท์พุท แอกทิฟที่ 0

RES สัญญาณนี้ใช้สำหรับรีเซทระบบในขณะเปิดเครื่องหรือขณะที่ขาดไฟเลี้ยง

AEN อีนาเบิลแอดเดรส

นอกจากนี้เรายังต้องรู้แอดเดรสของพอร์ทซึ่งเครื่องคอมพิวเตอร์มีการใช้งานอยู่ เพื่อไม่ให้เกิดความผิดพลาดจากการใช้พอร์ทซ้ำกัน ซึ่งตำแหน่งของพอร์ทต่างๆ แสดงไว้ดังตารางที่ 3.2

แอดเดรส	หน้าที่
000 - 01F	ดีเอ็มเอ คอนโทรลหมายเลข 1 (8237A – 5)
020 - 03F	อินเตอร์รัพท์ คอนโทรลหมายเลข 1 (8259A)
040 - 05F	ไทม์เมอร์ (8254)
060 - 06F	คีย์บอร์ด (8042)
070 - 07F	นอนมาสเคเบิลอินเตอร์รัพท์และซีมอสแรม
080 - 09F	ดีเอ็มเอ เพจรีจิสเตอร์ (74LS612)
0A0 - 0BF	อินเตอร์รัพท์ คอนโทรลหมายเลข 2 (8259A)
0C0 - 0DF	ดีเอ็มเอ คอนโทรลหมายเลข 2 (8237A)
0F0 - 0FF	80287 ตัวช่วยประมวลผลทางคณิตศาสตร์
1F0 - 1F8	ฮาร์ดดิสค์
200 - 20F	พอร์ทควบคุมจอยสติ๊ก
258 - 25F	อินเทล อะโบฟ พอร์ท
278 - 27F	พอร์ทขนานเครื่องพิมพ์หมายเลข 2

แอดเดรส	หน้าที่
2E8 - 2EF	พอร์ทอนุกรมหมายเลข 4
2F8 - 2FF	พอร์ทอนุกรมหมายเลข 2
300 - 31F	โพรโตไทพ์ การ์ด
378 - 37F	พอร์ทขนานเครื่องพิมพ์หมายเลข 1
380 - 38F	พอร์ทไบซิงโครนัสหมายเลข 2
3A0 - 3AF	พอร์ทไบซิงโครนัสหมายเลข 1
3B0 - 3BF	โมโนโครมอะแดปเตอร์
3BC - 3BE	พอร์ทขนานเครื่องพิมพ์บนโมโนโครมอะแดปเตอร์
3C0 - 3CF	สำรองสำหรับ อีจีเอ
3D0 - 3D7	คัลเลอร์กราฟฟิคอะแดปเตอร์
3E8 - 3EF	พอร์ทอนุกรมหมายเลข 3
3F0 - 3F7	ควบคุมฟลอปปี้ดิสค์
3F8 - 3FF	พอร์ทอนุกรมหมายเลข 1

ตารางที่ 3.2 แสดงหมายเลขพอร์ทของเครื่องคอมพิวเตอร์

3.2 การทำงานของการ์ดพอร์ท8255

เริ่มจากสล๊อทของคอมพิวเตอร์จะพบว่า D0-D7 จะถูกต่อเข้ากับขา A1-A8 ของไอซี 74F245 และขาB1-B8 ของไอซี 74F245 จะต่อเข้ากับขา D0-D7 ของพอร์ท8255ทั้งสามตัว โดยขาDIR ของไอซี 74LS245 จะต่อเข้ากับขา B0 ของไอซี 74LS245 เนื่องจากขา IOR ของสล๊อทต่อเข้ากับขา A0 ของไอซี 74LS245 ซึ่งเป็นตัวควบคุมสภาวะรับหรือส่งข้อมูล ส่วนขา IOW ของสล๊อทต่อเข้ากับขา A1 ของไอซี 74LS245 และขา B0,B1 ของไอซี 74LS245 ต่อเข้ากับขา IOR,IOW ของพอร์ท8255ทุกตัวเช่นเดียวกัน เพื่อเป็นการบอกว่าจะรับหรือส่งข้อมูล ส่วนขารีเซท A0,A1 ของสล๊อทต่อเข้ากับขา A2,A3,A6 ของไอซี 74LS245 โดยขา B7 ต่อเข้ากับขารีเซทของพอร์ท8255ทุกตัว ส่วนขาB2,B3 ต่อเข้ากับขา A0,A1 ของพอร์ท8255ทุกตัวเพื่อเซทว่าจะใช้งานพอร์ทใด และให้เป็นอินพุทหรือเอาท์พุท โดย

เซ็ทที่ A0,A1 ของสล๊อท ขา E ของไอซี 74F245 จะมีไฟเลี้ยงตลอดทำให้ไอซี 74F245 ไม่ทำงานจนกว่าจะมีลอจิก0จากเอาท์พุทของไอซี 74LS688 เท่านั้น ส่วนไอซี 74LS245 ทำงานตลอดเพราะต่อขา E ลงกราวนด์และขา DIR มีไฟเลี้ยงตลอดเวลา ซึ่งเป็นการเซ็ทให้ข้อมูลส่งจาก A ไป B เท่านั้น ส่วนขา A2,A3 ของสล๊อทต่อเข้ากับขา A4,A5 ของไอซี 74LS245 และออกที่ขา B4,B5 ไปเข้าที่ขา A,B ของไอซี 74LS139 ซึ่งเป็นวงจรดีโค้ด 2 ไลน์ทู 4 ไลน์ โดยขาA2,A3 ของสล๊อททำหน้าที่เลือกว่าจะให้พอร์ท 8255ตัวใดทำงาน การจะส่งหรือรับข้อมูลนั้นต้องอ้างพอร์ทของการ์ดก่อนโดยใช้ไอซี 74LS688 ซึ่งเป็นคอมพาราเตอร์ โดยขา A4-A11 ของสล๊อทต่อเข้ากับ P0-P7 และขา1-8ของดิปสวิทช์ต่อเข้ากับขา Q0-Q7 ของไอซี 74LS688 ซึ่งถ้าแอดเดรสที่ส่งมาตรงกับที่ตั้งดิปสวิทช์ไว้จะทำให้ P เท่ากับ Q แอกทีฟ 0 ทำให้ขา E ของไอซี 74F245 แอกทีฟทำให้รู้ว่าพอร์ท8255ตัวใดทำงานที่ตำแหน่งพอร์ทเท่าใด และใช้พอร์ทใดรับหรือส่งข้อมูลตัวอย่างเช่นเราตั้งดิปสวิทช์ไว้ที่ 00110000 (300) เมื่อเราป้อน A0-A11 เป็น 300H คือ

A0-A1 เป็น 00 ทำให้ทราบว่าเราใช้พอร์ท A

A2-A3 เป็น 00 ทำให้ 74LS139 ดีโค้ดออกที่ Y0 ทำให้พอร์ท 8255 ตัวที่หนึ่งทำงาน

A4-A11 เป็น 00110000 ตรงกับที่ตั้งดิปสวิทช์ไว้ทำให้การ์ดพอร์ททำงานโดย

พอร์ท 8255 ตัวที่หนึ่ง

PORT A = 300 H

PORT B = 301 H

PORT C = 302 H

CONTROL PORT 1 = 303 H

พอร์ท 8255 ตัวที่สอง

PORT A = 304 H

PORT B = 305 H

PORT C = 306 H

CONTROL PORT 1 = 307 H

พอร์ท 8255 ตัวที่สาม

PORT A = 308 H

PORT B = 309 H

PORT C = 30A H

CONTROL PORT 1 = 30B H

บทที่ 4
ทฤษฎีบทของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยว (Single Ship Microcontroller)คือไมโครคอมพิวเตอร์ที่มีขนาดเล็กโดยบรรจุไว้ในแผงวงจรรวม(integrated circuit) เพียงชิพ เดี่ยวเหมาะสมสำหรับงานงานควบคุมอุปกรณ์อื่นๆโดยอัตโนมัติ เพราะผู้ใช้สามารถเขียนโปรแกรมควบคุมการทำงานได้ตามต้องการ ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล 51 หรือ MCS51 อันได้แก่ เบอร์ 8051 และ 8052 จะมีชุดคำสั่งต่างกันเล็กน้อย MCS-51มีหลายรุ่นซึ่งมีสถาปัตยกรรมภายในที่เหมือนกัน เพียงแต่ขนาดหรือจำนวนหน่วยการทำงานภายในต่างกันออกไป ในที่นี้รวมเรียกว่า 8051

คุณลักษณะพื้นฐานของ MCS - 51

* หน่วยประมวลผลกลาง 8 bit
* หน่วยประมวลผลสำหรับข้อมูลแบบบิต (Boolean Processor)
* ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบท์
* ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบท์
* หน่วยความจำภายในขนาด 4 กิโลไบท์ แบบ EPROM (8751) หรือแบบ ROM (เบอร์ 8051)
* หน่วยความจำแบบ RAM ภายในจำนวน 128 กิโลไบท์
* พอร์ตอินพุท/เอาท์พุทแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกการทำงานได้อย่างอิสระ
* วงจรนับ/จับเวลาขนาด 16 บิต จำนวน 2 วงจร
* วงจรสื่อสารแบบอนุกรมแบบฟลูดูเพล็กซ์(Full Duplex)
* วงจรควบคุมการอินเตอร์รับต์จากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนดลำดับความสำคัญไว้ 2 ระดับ
* วงจรออสซิลเลเตอร์ภายใน

จากข้อดีดังกล่าวทำให้ MCS-51 เป็นที่นิยมนำมาใช้ในการควบคุมระบบอัตโนมัติมาก คุณสมบัติดังกล่าวบรรจุไว้ในวงจรรวมเดี่ยว (Single Chip) ขนาด 40 ขา ดังนั้นจำสามารถออกแบบให้ระบบทั้งหมดมีขนาดเล็ก และการที่ทั้งหมดบรรจุในวงจรรวมเดี่ยวทำให้ตรวจสอบหาข้อผิดพลาดง่ายและลดปัญหาเรื่องการที่มีสัญญาณรบกวนในระบบ แต่การที่จะใช้งาน MCS-51 จำเป็นต้องศึกษาและ

ทำความเข้าใจถึงองค์ประกอบและโครงสร้างของ MCS-51 เสียก่อนแล้วจึงเขียนโปรแกรมควบคุมการทำงานให้เป็นไปตามต้องการ

4.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมของ 8051 เป็นหน่วยความจำสำหรับเก็บข้อมูลและคำสั่งต่างๆ ซึ่งแม้ไม่มีการจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์ ข้อมูลก็ไม่สูญหาย โครสร้างหน่วยความจำมีลักษณะ เช่น ROM (Read Only Memory) หรือ EPROM (Erasable Programable Read Only Memory) เป็นต้น

4.1.1 หน่วยความจำโปรแกรมภายใน

ไมโครคอนโทรลเลอร์ตระกูล 8051 นี้ มีขนาดหน่วยความจำภายในแตกต่างกัน ดังนี้

8051 และ 8052 มีหน่วยความจำแบบ ROM ขนาด 4 กิโลไบต์ และ 8 กิโลไบต์ ตามลำดับ ประกอบภายในไอซี เหมาะกับงานอุตสาหกรรมที่มีกำลังการผลิตมาก

8751 มีหน่วยความจำ EPROM ขนาด 4กิโลไบต์อยู่ภายในไอซี ข้อมูลที่จัดเก็บภายในนี้สามารถลบได้โดยใช้แสงอัลตราไวโอเลต แล้วนำไปโปรแกรมใหม่ได้อีก คล้าย EEPROM

8031 และ 8032 ไม่มีหน่วยความจำภายใน จึงต้องมีหน่วยความจำภายนอกในการใช้งานเสมอ

4.1.2 หน่วยความจำโปรแกรมภายนอก

หน่วยความจำโปรแกรมภายนอกเป็นการใช้ EEPROM หรือ ROM เชื่อมต่อเข้ากับระบบของ 8051 โดยมีความจำเป็นต้องใช้เนื่องจากต้องการเก็บข้อมูลมากกว่า 128 หรือ 256 ไบต์เท่านั้น อย่างไรก็ตามไม่ว่าจะนำหน่วยความจำภายนอกนี้ไปใช้งานอะไรก็ตามจะต้องใช้พอร์ทอินพุท/เอาท์พุทของ 8051 ในการติดต่อกับหน่วยความจำภายนอก

4.2 การจัดการหน่วยความจำของ 8051

หน่วยความจำของ 8051 แบ่งออกไว้ 2 แบบตามลักษณะการใช้งาน คือ

4.2.1 Program Memory เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปรหัสภาษาเครื่อง (Machine Language) ซึ่งต้องการให้ 8051 ทำงาน เมื่อ 8051 ทำงานก็จะอ่านข้อมูลที่เก็บไว้ในหน่วยความจำประเภทนี้ไปถอดรหัสแล้วสร้างสัญญาณควบคุมส่วนอื่นๆ ให้ทำงานตามรหัสคำสั่งนั้น หน่วยความจำนี้ต้องเป็นแบบ ROM และผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของหน่วยความจำเป็นรหัสภาษาเครื่องของ 8051 ตามลำดับการทำงานที่ต้องการ

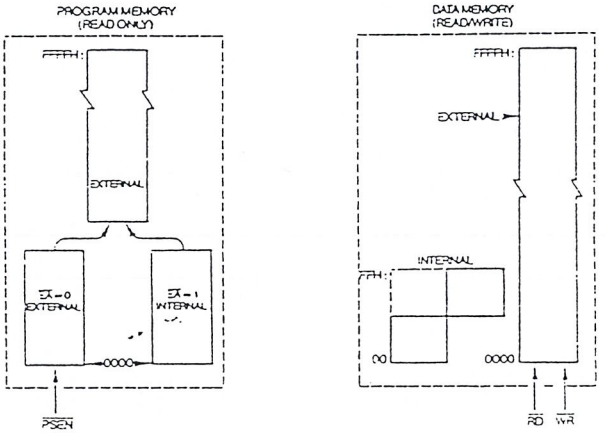
Internal Memory หมายถึง หน่วยความจำที่อยู่ภายใน 8051 ส่วน External Memory หมายถึง หน่วยความจำที่อยู่ภายนอก 8051

ไมโครคอนโทรลเลอร์เบอร์ 8031 ,8051 และ8751 นั้น โดยโครงสร้างและรหัสคำสั่งจะเหมือนกันทุกประการ ต่างกันที่

- 8031 ไม่มีหน่วยความจำภายใน ผู้ใช้ต้องทำการต่อ Program Memory ไว้เองภายนอก 8051
- 8051 มี ROM ภายในขนาด 4 กิโลไบต์ ถ้าต้องการเก็บคำสั่งควบคุมการทำงานไว้ในหน่วยความจำส่วนนี้ ต้องทำการโปรแกรมไว้ตั้งแต่ขั้นตอนการผลิตไอซี ซึ่งผู้ใช้จะไม่สามารถแก้โปรแกรมได้ ถ้าต้องการนำมาใช้งานกับหน่วยความจำโปรแกรมภายนอกก็สามารถทำได้โดยการต่อ ROM ไว้ภายนอก แล้วต่อขา EA/ ของ 8051 ลงกราวนด์ไว้
- 8751 มีหน่วยความจำภายใน EPROM ขนาด 4 กิโลไบต์ ไว้เก็บโปรแกรมคำสั่งที่จะให้ 8751 ทำงาน ผู้ใช้สามารถเขียนโปรแกรมลงไปใน EPROM ได้โดยใช้เครื่องโปรแกรม EPROM (EPROM Programmer) และผู้ใช้สามารถทำการแก้ไขโปรแกรมภายใน EPROM ได้โดยการล้างข้อมูลในทุกตำแหน่งของ EPROM ออกโดยใช้แสงอัลตราไวโอเลตผ่านกระจกใสบนวงจรรวมเข้าไปในวงจรรวมตามเวลาที่กำหนด แล้วจึงใช้เครื่องโปรแกรม EPROM ทำการเขียนโปรแกรมใหม่ลงไป

4.2.2 Data Memory เป็นหน่วยความจำที่ 8051 ใช้สำหรับเก็บข้อมูลแล้วนำมาใช้อีก ในระหว่างการทำงานของ 8051 การอ่านหรือเขียนข้อมูลจากหน่วยความจำทำโดยคำสั่งที่เก็บไว้ใน Program Memory หน่วยความจำของ Data Memoryนี้จะเป็น Random Access Memory (RAM) ซึ่งถ้ามีไฟเลี้ยงอยู่ ข้อมูลก็จะไม่สูญหาย แต่ถ้าไม่มีไฟเลี้ยง ข้อมูลก็จะสูญหายไป การสูญหายของข้อมูลไม่ได้

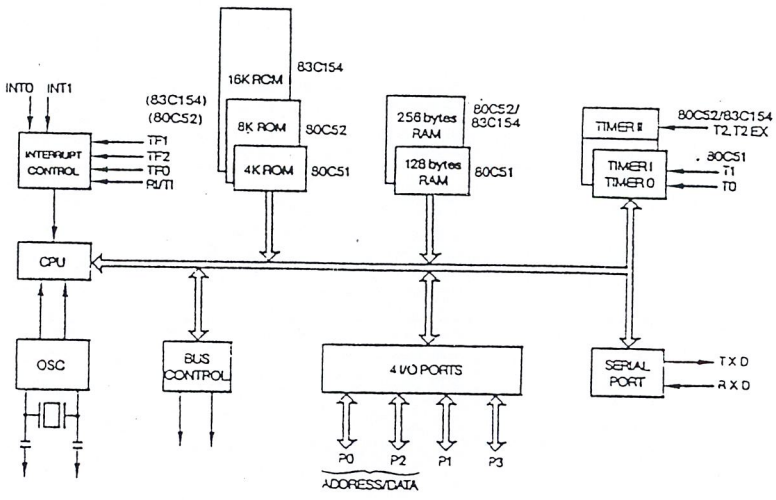
หมายความว่าไม่มีอะไรอยู่เลยแต่เป็นการที่มีข้อมูลใหม่ที่ไม่ใช้ข้อมูลเดิมเข้าเกี่ยวข้อง หน่วยความจำแบบ Data Memory ของ 8051 มีอยู่ 2 ชุด ชุดหนึ่งอยู่ภายใน 8051 จำนวน 128 ไบท์ที่ตำแหน่ง 00H ถึง 7FH และอีกชุดอยู่ภายนอกของวงจรรวม 8051 มีได้สูงสุด 65535 ไบท์ (64 กิโลไบท์) อยู่ที่ตำแหน่ง 0000H ถึง FFFFH จะต้องต่ออยู่ภายนอก 8051 เสมอ ดังแสดงในแผนภูมิหน่วยความจำ (Memory Map) ในรูปที่ 4.1



รูปที่ 4.1 แผนภูมิหน่วยความจำของ 8051

4.3 โครงสร้างของ 8051

ภายใน 8051 จะประกอบด้วย gate ต่างๆ เช่น AND OR NOT ซึ่งGATE เหล่านี้จะถูกนำมาออกแบบให้มีหน้าที่การทำงานต่างๆ เช่น วงจรถอดรหัสคำสั่ง (Instrument Decoder) , วงจรสร้างสัญญาณนาฬิกา (Clock Signal Generator) โครงสร้างภายในของ 8051 ประกอบด้วยส่วนย่อยๆ ดังในไดอะแกรมรูปที่ 4.2



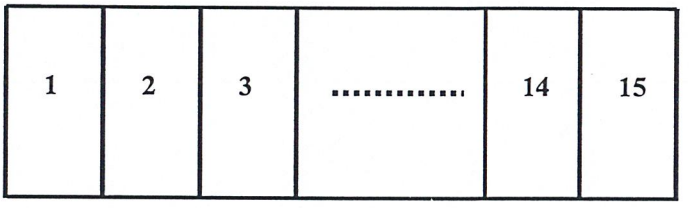
รูปที่ 4.2 ไดอะแกรมโครงสร้างของ 8051

ไดอะแกรมในรูปที่ 4.2 เป็นโครงสร้างหลักๆ ของ 8051 เนื่องจากลักษณะของ 8051 คล้ายคอมพิวเตอร์จึงประกอบด้วย 3 ส่วนหลักๆ คือ

4.3.1 CPU (Central Processing Unit) หรือ ตัวประมวลผลมีหน้าที่สร้างสัญญาณควบคุมในการติดต่อกับส่วนอื่นๆ เรียกว่าส่วนควบคุม (Control Unit) สัญญาณที่สร้างจากวงจรควบคุม ได้แก่ สัญญาณติดต่อกับหน่วยความจำ ,อุปกรณ์รับหรือส่งข้อมูลกับ 8051 ซึ่งส่วนควบคุมการอินเตอร์รัพท์ (Interrupt Control) และส่วนควบคุมบัส (Bus Control) ก็เป็นส่วนหนึ่งของวงจรควบคุมด้วย การสร้างสัญญาณควบคุมจาก CPU นี้ จะทำการสร้างสัญญาณโดยการถอดรหัสคำสั่งตามที่มีการโปรแกรมไว้ และสัญญาณที่สร้างขึ้นจะมีการอิงกับสัญญาณนาฬิกาที่สร้างจากวงจรออสซิลเลเตอร์เพื่อให้ทุกส่วนมีการทำงานประสานกันได้อย่างถูกต้อง

ใน CPU นี้ยังประกอบไปด้วยส่วนย่อยอีกส่วนที่เรียกว่า ALU (Arithmatic Logic Unit) ซึ่งทำหน้าที่ประมวลผลทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร แล้วนำผลลัพธ์ไปเก็บยังรีจิสเตอร์ที่ต้องการ

4.3.2 หน่วยความจำ (Memory) มีไว้สำหรับเก็บข้อมูล มีลักษณะคล้ายกล่องเอกสารที่เรียงต่อกันไว้จำนวนมาก แต่ละกล่องมีเอกสารหนึ่งแผ่น ดังรูปที่ 4.3 มีกล่องเอกสารทั้งหมด 15 กล่อง



รูปที่ 4.3 ภาพเสมือนของหน่วยความจำ

ถ้าต้องการเอาเอกสารจากกล่องใด หรือเอาเอกสารไปเก็บไว้ในกล่องใด ต้องรู้หมายเลขของกล่องเสียก่อน ซึ่งถ้าเป็นหน่วยความจำแล้วหมายเลขของกล่องก็คือ ตำแหน่งของหน่วยความจำ หรือแอดเดรสนั่นเอง การนำข้อมูลไปเก็บไว้ในกล่องเรียกการเขียนข้อมูล (Write) และการนำข้อมูลออกจากหน่วยความจำเรียกการอ่านข้อมูล (Read) ซึ่งแต่ละตำแหน่งของหน่วยความจำจะสามารถเก็บข้อมูลได้เพียงแปดหลักของเลขฐานสอง ดังนั้นแต่ละตำแหน่งของหน่วยความจำสามารถเก็บข้อมูลได้ระหว่าง 0 ถึง 255 (00000000 ถึง 11111111 ในเลขฐานสอง) แต่จำนวนตำแหน่งที่จะเก็บข้อมูลได้ขึ้นกับไมโครโปรเซสเซอร์แต่ละเบอร์ การติดต่อกับหน่วยความจำมีสัญญาณสามกลุ่ม คือ

1) แอดเดรสหรือค่าตำแหน่งที่ต้องการติดต่อกับหน่วยความจำใน 8051 จะติดต่อกับหน่วยความจำประเภท Program Memory หรือ Data Memory ได้สูงสุดชนิดละ 65536 ตำแหน่ง ดังนั้นการอ้างอิงกับหน่วยความจำต้องใช้เลขฐานสองทั้งหมด 16 ตำแหน่ง

2) ข้อมูลที่จะทำการอ่านหรือเขียน

3) สัญญาณควบคุมที่ส่งไปยังหน่วยความจำ เพื่อบอกว่าจะทำการอ่านหรือเขียนข้อมูล

สัญญานเหล่านี้จะถูกควบคุมใน 8051 สร้างมาจากวงจรถอดรหัสของคำสั่งที่ 8051 อ่าน จากหน่วยความจำ Program Memory เข้าไปทำงานนั่นเอง ในรูปที่ 4.2 หน่วยความจำนี้มีขนาดต่างกัน ตามเบอร์ของไมโครคอนโทรลเลอร์ดังอธิบายรายละเอียดในหัวข้อที่ 4.1

4.3.3 อุปกรณ์อินพุทและเอาท์พุท (Input/Output Device) เป็นส่วนที่จะใช้ส่งข้อมูลเข้าหรือออกจาก 8051 ทำให้ 8051 ติดต่อกับภายนอกได้ ดังในไดอะแกรมรูปที่ 4.2 อุปกรณ์อินพุทเอาท์พุทได้แก่ 4 I/O Port ,Time 0, Time 1, Serial Port การทำงานแต่ละส่วนมีดังนี้

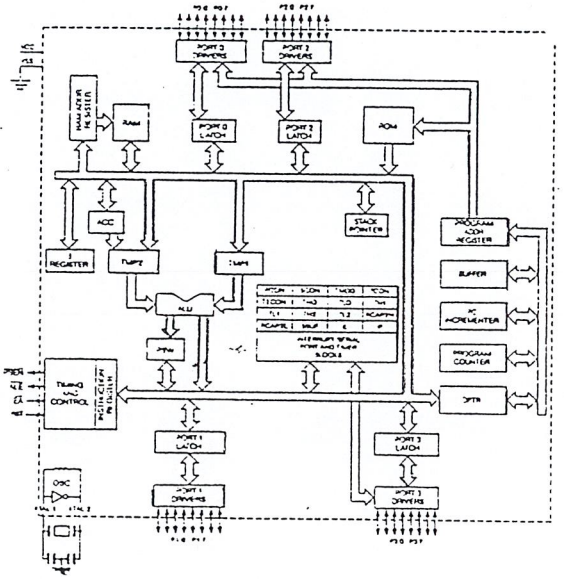
1) I/O Port คำว่า พอร์ทหมายถึงจุดที่ติดต่อกับส่วนที่อยู่นอก 4 I/O Port ของ 8051 เป็นที่ใช้สำหรับรับ-ส่งข้อมูลซึ่งเป็นสัญญานดิจิตอลเข้าหรือออกจากตัว MSC-51 พอร์ทมีทั้งหมด4พอร์ท โดยแต่ละพอร์ทจะรับ-ส่งข้อมูลได้8บิต มีพอร์ท PO,P1,P2,P3 บางพอร์ทใช้ทำงานมากกว่าหนึ่งอย่างได้ เช่น พอร์ทP0 และP2 จะใช้สำหรับส่งค่าตำแหน่ง(Address) ของหน่วยความจำที่ต้องการติดต่อและพอร์ท P0 จะใช้สำหรับส่งข้อมูลเมื่อต้องการติดต่อกับหน่วยความจำได้ด้วยแต่ไม่ได้เกิดที่เวลาเดียวกัน แต่จะใช้วิธีการทำงานตามลำดับ โดยควบคุมจากสัญญานควบคุม (Control) ที่ถอดรหัสมาจากแต่ละคำสั่งที่ทำให้คอมพิวเตอร์ทำงานนั่นเอง และสัญญาณทั้งหมดจะอ้างอิงกับสัญญาณนาฬิกา

2) Timer 0 และ Timer 1 เป็นวงจรนับที่สามารถกำหนดให้ทำการนับจำนวนไซเคิลของสัญญาณที่ต่อจากสัญญาณภานนอก 8051 หรือไซเคิลของสัญญาณนาฬิกาใน8051 ก็ได้ ค่าจากการนับจะถูกอ่านหรือตั้งค่าเริ่มต้นของการนับได้โดย CPU

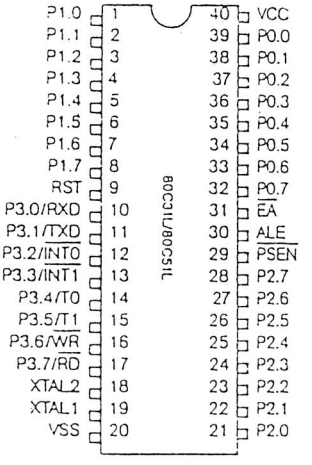
3) Serial Port หรือพอร์ทอนุกรม CPUจะอ่านและเขียนข้อมูลกับ Serial Port เป็นแบบ 8 บิท แต่ข้อมูลจะถูกส่งออกจาก 8051 เรียงไปทีละบิทออกจากขา TXD และในการรับข้อมูลก็จะรับเข้ามาทีละบิททางขา RXD และจัดเรียงใหม่เป็น 8 บิท เพื่อให้CPUอ่านใช้งานต่อไป

4.4 สถาปัตยกรรมของ 8051

ในหัวข้อที่ 4.2 ได้กล่าวถึง ไดอะแกรมของ 8051 อย่างกว้างๆ ซึ่งพอจะบอกได้โดยสังเขปว่าประกอบด้วยส่วยใหญ่ๆอะไรบ้าง ในรูปแบบที่ 4.4 เป็นสถาปัตยกรรมภายใน 8051 เพียงชิพเดียว และสัญญาณจากภายในจะต่อออกสู่ภายนอกขา (Pin) ของ8051 ที่มีอยู่ 40 ขา ดังรูปที่ 4.5



รูปที่ 4.4 สถาปัตยกรรมภายใน 8051



รูปที่ 4.5 ไดอะแกรมของ 8051 แบบ DIP

8051 ไมโครคอนโทรลเลอร์ที่บรรจุอยู่ในวงจรรวมแบบ Dual Inline Package (DIP) ซึ่งแต่ละข้างของ 8051 มีขาอยู่ข้างละ 20 ขารวมทั้งหมด 40 ขานั้นจะใช้งานต่างๆ ดังนี้

Vcc

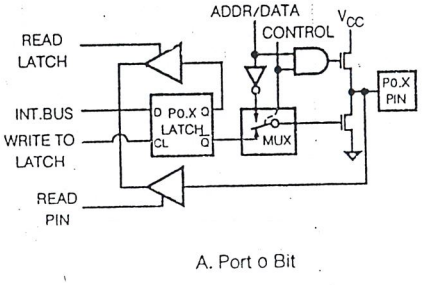
ขา 40 ที่ต้องป้อนไฟเลี้ยง +5 โวลท์เข้าไปเพื่อให้วงจรรวมทำงานได้ ระดับโวลท์เตทของลอจิก 0 และ 1 ของ 8051 จึงต่อเข้ากับอุปกรณ์ลอจิกแบบ TTL ได้โดยตรง

Vss

ขาที่ 20 เป็นขาที่ต้องต่อกับกราวด์ (Ground)ของแหล่งจ่ายไฟ การต่ออุปกรณ์ทั้งหมดจะต้องมีกราวด์ของอุปกรณ์ต่อเข้าด้วยกัน

4.4.1 Port 0

เป็นพอร์ตขนานขนาด 8 บิท อยู่ที่ขา 39 ถึง 32 เริ่มจากบิท 0 ถึงบิท 7 ตามลำดับดังในรูปที่ 4.5 แต่ละขาจะเรียกว่า P0.0 , P0.1 , ..., P0.7นั้น P0.7หมายถึงบิทที่ 7 ของพอร์ท 0 เป็นบิทที่มีนัยสำคัญสูงสุด (Most Significant) และ P0.0คือบิท 0 ของพอร์ท 0 ซึ่งเป็นบิทที่มีนัยสำคัญต่ำสุด (Least Significant) พอร์ท 0 นี้ใช้ได้ทั้งการรับ - ส่งตำแหน่งและข้อมูลกับหน่วยความจำหรือใช้เป็นพอร์ทรับ - ส่งข้อมูลก็ได้ ข้อมูลที่ส่งออกทางพอร์ท 0 จะถูก Latch ไว้ที่ขาของพอร์ท 0 เป็นแบบ Open Drain Bidirectional ดังรูปที่ 4.6



รูปที่ 4.6 โครงสร้างของพอร์ท 0

ในรูปที่ 4.6 เมื่อเปรียบเทียบกับรูปที่ 4.4 ส่วนหนึ่งของรูปที่ 4.6 ก็คือ Port 0 Latch ในรูปที่ 4.4 และส่วนที่ 2 ของรูปที่ 4.6 ก็คือ Port 0 Driver ของรูปที่4.4 นั่นเอง

จากโครงสร้างในรูปที่ 4.6 เมื่อมีคำสั่งการเขียนข้อมูลมายังพอร์ท 0 ข้อมูลจากInternal Data Bus จะถูก Latch ไว้ที่ D-FFโดยสัญญาณ "Write To Latch" ที่ถูกสร้างมาจากส่วน Timing and control และสัญญาณในการอ่านข้อมูลจากพอร์ท 0 จะอ่านได้ 2 แบบ คือการอ่านข้อมูลที่ส่งไปเก็บไว้ที่พอร์ทก็จะมีสัญญาณ Read Latch มาเพื่ออ่านข้อมูลจาก D-FF กลับไปยัง Internal Data Bus การอ่านข้อมูลอีกแบบหนึ่งก็คืออ่านสภาวะของสัญญาณที่เข้ามาทางพอร์ท 0 ก็จะมีสัญญาณ Read Pin มาควบคุมการอ่านพอร์ท 0 จะใช้งานหลายอย่างดังนี้

1) ใช้สำหรับส่งค่าตำแหน่งหน่วยความจำภายนอกที่ต้องการติดต่อด้วย ตำแหน่งหน่วยความจำสูงสุดที่จะติดต่อได้ก็คือ 64 Kbyte จึงมีค่าตำแหน่งหน่วยความจำ 16 บิทของเลขฐานสอง ค่าตำแหน่งหน่วยความจำ 8 บิทล่างจะถูกส่งออกไปทางพอร์ท 0และ 8 บิทบน จะถูกส่งออกไปทางพอร์ท 2

2) ใช้รับ-ส่งข้อมูลกับ Data Memory หรือใช้รับข้อมูลจาก Program Memory

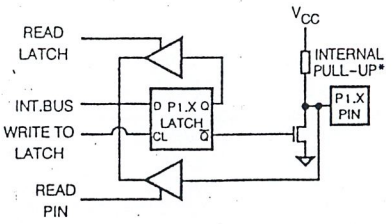
3) ใช้รับส่งข้อมูลผ่านทางพอร์ทโดยตรง ในกรณีที่ไม่มีการใช้หน่วยความจำของ Program Memory หรือ Data Memory ภายนอก

วงจรภายในส่วน Timing and Control จะใช้เป็นตัวสร้างสัญญาณมาควบคุมวงจรในรูปที่ 4.6 เพื่อให้การทำงานแต่ละอย่างข้างต้นเมื่อแต่ละบิทของพอร์ท 0 ทำงานตามข้อ 1 และ 2 ข้างต้น วงจร Timing and Control จะทำให้สภาวะลอจิกของขา Control เป็น 1 ซึ่งทำให้ สวิทซ์ MUX อยู่ในตำแหน่งข้างบน เมื่อพอร์ท 0 จะส่งข้อมูลซึ่งเป็นค่าตำแหน่งหน่วยความจำหรือข้อมูลที่จะเขียนออกไปยังหน่วยความจำภายนอกก็ส่งค่าดังกล่าวมายัง ADDR/DATA ถ้าข้อมูลที่ส่งมาเป็น 1 จะทำให้สัญญาณออกจาก AND GATE เป็น 1 และสัญญาณที่ออกจากInverter เป็น 0 ดังนั้น FET ตัวบน ON (สภาวะ ON ของ FET ก็คือความต้านทานขา D และ S มีค่าต่ำมากเสมือนกับเป็นวงจรปิด) ส่วน FET ตัวล่าง OFF ON (สภาวะ ON ของ FET ก็คือความต้านทานขา D และ S มีค่าต่ำมากเสมือนวงจรเปิด) สภาวะลอจิกที่ขา P0.X PIN จะเป็น 1 แต่ถ้าข้อมูลที่ส่งออกมายัง ADDR/DATA เป็น0ก็จะทำให้สัญญาณจาก AND GATE เป็น 0 และสัญญาณที่ออกจาก Interver เป็น 1 ดังนั้น FET ตัวบนจะ OFF ส่วน FET ตัวล่างจะ

ON ทำให้สภาวะลอจิกที่ขา P0.X PIN เป็น 0 เมื่อ 8051 ต้องการใช้พอร์ท 0สำหรับการอ่านข้อมูลจากหน่วยความจำภายนอก หรือใช้ทำงานในข้อ 3 ข้างบนก็จะได้โดย วงจรTimeing and Control ทำให้สภาวะของสัญญาณ Control ทำให้สภาวะลอจิกของสัญญาณ Control เป็น 0 ทำให้เอาท์พุทจาก AND GATE เป็น 0 FET ตัวบนเป็น OFF และสวิทช์ MUX จะอยู่ในตำแหน่งข้างล่าง ดังนั้น FET ตัวล่าง จะ ON หรือ OFF ก็แล้วแต่ข้อมูลที่ขา Q ของ D-FF ก็จะมีสัญญาณ Write to Latch มายัง D-FF เมื่อมีการเขียนข้อมูลจาก Internal Data Bus มายัง D-FF ก็จะมีสัญญาณ Write to Latch มายัง D-FF ด้วย ถ้าข้อมูลที่เขียนมาเป็น 1 ก็จะทำให้ขา Q มีสภาวะลอจิกเป็น0 ทำให้ FET ตัวล่าง OFF ดังนั้นขา P0.X จะอยู่เป็นสภาวะอิมพีแดนซ์สูง (High Impedance) เพราะ FET ทั้งสองตัวเป็น OFF แต่ถ้าข้อมูลที่เขียนมายัง D-FF เป็น 0 ทำให้ FET ตัวล่าง ON แต่ตัวบนOFF ทำให้สภาวะลอจิกที่ขา P0.X เป็น1 ดังนั้น PORT 0 เมื่อให้ทำงานเป็น พอร์ท 0 เป็น พอร์ทส่งข้อมูล (ไม่ใช่ตำแหน่งส่งความจำ) จะไม่สามารถแสดงภาวะลอจิก 1ได้จึงต้องต่อตัวต้านทาน Pull Up ไว้ภายนอก ระหว่างขา P0.X กับไฟเลี้ยงวงจร ถ้าใช้พอร์ท 0 สำหรับรับข้อมูลเข้าจะต้องเขียน 1 มาเก็บไว้ยัง D-FF เสียก่อนเพื่อให้ขา P0.X อยู่ในสภาวะ High Impedence แล้วใช้คำสั่งอ่านสภาวะลอจิกเข้าไปยัง Internal Data Bus ต่อไป โดยคำสั่งอ่านสภาวะลอจิกทางพอร์ท 0 ก็จะทำให้วงจร Timing and Control สร้างสัญญาณ Read Pin สำหรับการอ่านสภาวะลอจิกข้างต้น ถ้าไม่เขียน 1มาเก็บไว้ยัง D-FF ก่อนที่จะอ่านข้อมูลแล้วอาจมีข้อมูลค้างอยู่ที่ D-FF ทำให้ Q เป็น 0 และ Q/ เป็น 1 ซึ่งทำให้ FET ตัวล่าง ON สัญญาณที่ต่อเข้ามาที่ขา PO.X ไม่ว่าจะมีสภาวะลอจิกใดจึงถูกดึงลงกราวด์ ดังนั้นเมื่ออ่านข้อมูลเข้าไปก็จะพบว่าเป็น 0 เสมอ ในการข้อมูลจากหน่วยความจำภายนอกนั้น วงจรTiming and Controlก็จะเขียนข้อมูลมายัง D-FF ให้เป็น 1 และสร้างสัญญาณ Control ให้มีลอจิกเป็น 0 ก่อนจะอ่านข้อมูลเข้าไปด้วย

4.4.1 Port 1

เป็นพอร์ทขนานขนาด 8 บิท ในรูปที่ 4.5 คือ ขาP1.0 ถึง P1.7 (ขา 1-8) P1.0 หมายถึงบิทศูนย์ของพอร์ท 1 ซึ่งเป็น Least Signuficant Bit และ P1.7 หมายถึงบิทเจ็ดของพอร์ท 1 ซึ่งเป็น Most Signuficant Bit แต่ละบิทเป็นดังรูปที่ 4.7



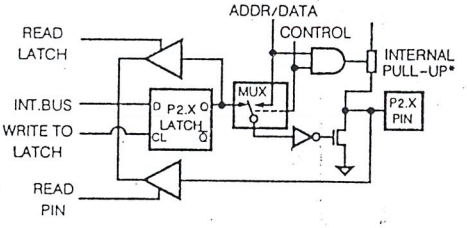
B. Port 1 Bit

รูปที่ 4.7 โครงสร้างของพอร์ท 1

ส่วนที่ 1 คือ Port 1 Latch ในรูปที่ 4.4 ซึ่งจะมีการทำงานเหมือนส่วนที่ 1 ของพอร์ท 0 ในรูปที่ 4.6 ส่วนที่ 2 คือ Port 1 Driver ในรูปที่ 4.4 Port 1 Driver นี้จะมีตัวต้านทานต่ออยู่เป็น Internal Pull up พอร์ท 1 นี้จะใช้ทำหน้าที่เป็นตัวรับ - ส่งข้อมูลที่ส่งออกไปทางแต่ละขา ก่อนที่จะอ่านข้อมูลเข้าไปทางพอร์ท 1 จะต้องเขียน 1 ไปยังทุกบิทของพอร์ทของส่งข้อมูลที่ส่งออกไปทางแต่ละขา ก่อนที่จะอ่านข้อมูลเข้าไปทางพอร์ท 1 จะต้องเขียน 1 ไปยังทุกบิทของพอร์ท 1 เสียก่อนเพื่อให้ FET อยู่ในสภาวะ OFF ก่อนมิฉะนั้นแล้วถ้ามีข้อมูล 0 ส่งออกมาค้างอยู่ที่ D-FF จะทำให้ FET อยู่ในสภาวะ ON ดังนั้นถ้าสัญญาณภายนอกส่งเข้ามาที่จานี้ก็จะถูกลัดวงจรบงกราวด์ โดยไม่สนใจว่าสภาวะลอจิกของสัญญาณที่เข้ามาจะเป็นอะไร ข้อมูลที่อ่านเข้าไปจึงจะเป็น 0 เสมอ

4.4.2 Port 2

พอร์ทขนานขนาด 8 บิทคือขา P2.0 ถึง P2.7 (บิท 0 ถึงบิท 7 ของพอร์ท 2) ในรูปที่ 4.5 โครงสร้างของพอร์ท 2 แต่ละบิทจะมีดังรูปที่ 4.8



C. Port 2 Bit

รูปที่ 4.8 โครงสร้างของพอร์ท 2

ลักษณะโครงสร้างจะเหมือนกับ Port 0 แตกต่างกันที่ใน Port 2 นั้นภาค Driver ขะใช้งานเพียง 2 ลักษณะคือ

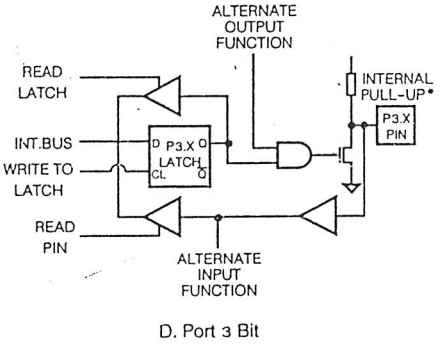
1. ใช้ส่งค่าตำแหน่งหน่วยความจะภายนอกที่ต้องการติดต่อ ค่าตำแหน่งนี้เป็น 8 บิท บนของค่าตำแหน่ง

2. ใช้เป็นพอร์ทรับและส่งข้อมูลกับภายนอก

ดังนั้นภาค Driver ของพอร์ท 2 จึงแตกต่างจาก Driver ของพอร์ท 0 โดยที่ในพอร์ท 2 นั้นจะมีเฉพาะ ADDR (ตำแหน่งหน่วยความจำ) เข้ามาที่ MUX (Multiplexer) เท่านั้น นอกนั้นแล้วการทำงานจะเหมือนกันและที่เอาท์พุทของพอร์ท 2 จะมี Internal pull-up ซึ่งเป็นตัวต้านทานและจะทำให้เอาท์พุทของพอร์ท 2 แสดงสถานะลอจิกเป็น 1 ได้ถ้า FET อยู่ในสถานะ OFF บางครั้งเรียกว่า " Quasi - bidirectional" เมื่อใช้เป็นพอร์ท อินพุทก็สามารถทำได้โดยการต่อสัญญาณภายนอกเข้ามาโดยตรง ถ้าสัญญาณภายนอกเป็น 0 ก็จะมีกระแสไหลออกจากพอร์ท (Source Current) ในการที่จะใช้พอร์ทนี้เป็นพอร์ทรับข้อมูลเข้า จะต้องเขียน 1 ไปยังแต่ละบิทของพอร์ทเสียก่อน ดังได้อธิบายในเรื่อง Port 0 และ Port 1

4.4.3 Port 3

คือขา P3.0 ถึง P3.7 หรือขา 10 - 17 ตามลำดับในรูปที่ 4.5 พอร์ทนี้มีโครงสร้างดังรูปที่ 4.9



รูปที่ 4.9 โครงสร้างของพอร์ท 3

ส่วนที่ 1 ในรูปที่ 4.9 เป็น Latch ข้อมูลที่เขียนมายังพอร์ท 3 ทาง Internal Bus เหมือนกับพอร์ทอื่น ๆ และพอร์ท 3 จะมี Internal pull - up อยู่ทุกบิท แต่พอร์ท 3 นี้แต่ละบิทจะใช้ในการ

REGISTER	CONTENT
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	00H
DPTR	0000H
P0-P3	0FFH
IP	00H
IE	0X000000B
TMOD	00H
TCON	00H
T2CON	00H
THO	00H
TLO	00H
TH1	00H
TL1	00H
TH2	00H
TL2	00H
RCAP2H	00H
RCAP2L	00H
SCON	00H
SBUF	Indeterminate
IOCON	00H

รูปที่ 4.10 ค่าของรีจิสเตอร์เมื่อเกิดการรีเซ็ต 8051

ในตารางรูปที่ 4.10 ช่องทางขวาเป็นค่ารีจิสเตอร์ที่อยู่ทาง.ซ้ายเมื่อสิ้นสุดการรีเซ็ท ในรีจิสเตอร์ SBUF เมื่อสิ้นสุดการีเซ็ท จะมีการีเซ็ตจะมีค่าที่ไม่แน่นอน และพอร์ทจะอยู่สภาวะลอจิก 1ทุกบิทตลอดเวลาที่สัญญาณของขา RST เป็น HIGH อยู่

เมื่อสัญญาณที่ขา RSTกลับเป็น 0 ก็จะออกจากการรีเซ็ต 8051 จะเริ่มทำงานจากคำสั่งที่อยู่ใน Program memory ตำแหน่ง 0000H เพราะค่าของรีซิสเตอร์ PC (Program Counter) ซึ่งใช้ชี้ตำแหน่งโปรแกรมที่จะทำงานถูกเปลี่ยนให้เป็น 0000H ดังนั้นผู้ใช้จะต้องเขียนโปรแกรมโปรแกรมมาเก็บไว้ที่ตำแหน่ง 0000H ในเครื่องไมโครคอมพิวเตอร์แบบบอร์ดเดี่ยว (Single Board Microcomputer) จะมีโปรแกรมที่เขียนเก็บไว้เริ่มจากตำแหน่ง 0000H นี้เรียกว่า มอนิเตอร์โปรแกรม (Monitor Program)ที่จะคอยรับการกดแป้นพิมพ์ (Keyboard) และแสดงผล (Display) แบบ 7 Segment

ALE

Address Latch Enable ขานี้จะส่งสัญญาณที่มีความถี่ 1/6 เท่าของสัญญาณนาฬิกาจากออสซิลเลเตอร์สัญญาณนี้จะส่งออกมาตลอดเวลายกเว้นบางครั้งของการติดต่อกับหน่วยความจำสำหรับข้อมูลภายนอก 8051 สัญญาณนี้จะใช้บอกกับอุปกรณ์ภายนอก 8051 ว่าขณะนี้สัญญาณนี้ Active (เป็นลอจิก1) จะมีการส่งข้อมูลที่เป็น 8 บิตล่างของตำแหน่งหน่วยความจำภายนอก 8051 ที่ต้องการติดต่อออกไปทางพอร์ท 0 อุปกรณ์ภายนอกจะใช้สัญญาณนี้ในการ Latch ข้อมูลไว้เพราะพอร์ท 0 จะส่งค่าตำแหน่งหน่วยความจำออกมาเพียงชั่วขณะเท่านั้น ซึ่งในเวลาต่อมาพอร์ท 0 จะใช้รับ - ส่งข้อมูลกับหน่วยความจำภายนอก สัญญาณ ALE จะสามารถต่อเข้ากับอุปกรณ์ TTL ชนิด LS ได้ถึง 8 อินพุท

$\overline{\text{PSEN}}$

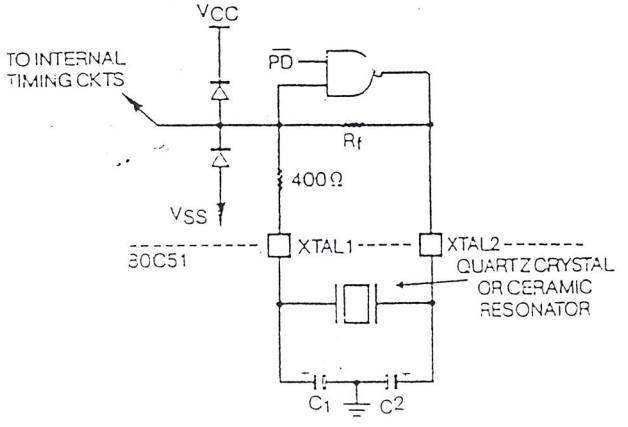
Program Store Enable เป็นขาที่ 29 ในรูปที่ 4.5 ขานี้ปกติจะให้ลอจิก 1 แต่จะส่งลอจิก 0 เมื่อต้องการอ่านคำสั่ง (Latch Instruction) ที่จะนำไปทำงานมาหน่วยความจำสำหรับโปรแกรมภายนอก 8051 ในกรณีที่อ่านคำสั่งซึ่งเก็บอยู่ในหน่วยความจำสำหรับโปรแกรมภายใน 8051 แล้วสัญญาณนี้จะไม่เปลี่ยนลอจิกเป็น 0 ขา PSEN นี้สามารถต่อไปยังขาอินพุทของ TTL ชนิด LS ได้ถึง 8 อินพุท

$\overline{\text{EA}}$

External Access ขา 31 ของรูปที่ 4.5 ขานี้เป็นขาอินพุทที่ต่อเข้าไปยังวงจร Timing and Control ในรูปที่ 4.4 เพื่อควบคุมการสร้างสัญญาณ PSEN ถ้าป้อนสัญญาณลอจิก 0 เข้าไปที่ขา EA นี้แสดงว่าโปรแกรมในตำแหน่ง 0000H ถึง 0FFFH ที่ต้องการให้ทำงานถูกเก็บไว้ภายนอก เพื่อทำการ FETCH คำสั่งเข้ามาในการทำงานแต่ถ้าสัญญาณที่ป้อนให้ขา EA เป็น 1 หมายความว่าโปรแกรมในตำแหน่ง 0000H ถึง 0FFFH ถูกเก็บไว้ภายใน 8051 การทำงานในตำแหน่งหน่วยความจำช่วงนี้จะอ่านคำสั่งต่างๆ จาก ROM ภายใน 8051

XTAL 1

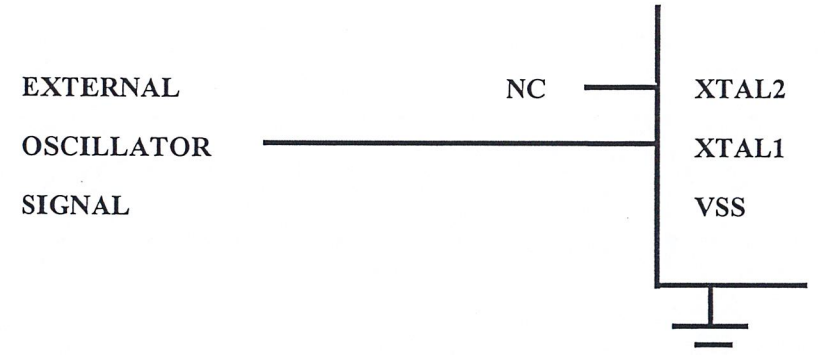
ขาที่ 19 ของรูปที่ 4.5 ขานี้จะต่อกับขาของ Inverting Amplifier (วงจรขยายแบบกับเฟสของสัญญาณ) ที่ประกอบเป็นวงจรออสซิลเลเตอร์ NAND Gate จะทำหน้าที่เป็นวงจรขยายแบบกลับเฟสของสัญญาณที่จะควบคุมให้มีการออสซิลเลตหรือไม่ก็ขึ้นกับสัญญาณ PD ซึ่งต่อมาจากบิท PD ของรีจิสเตอร์ PCON ถ้าในหน่วยความจำต้องการใช้สัญญาณนาฬิกา (Clock Signal) จากภายนอกมาเป็นสัญญาณนาฬิกา ควบคุมการทำงานของ 8051 ก็ให้ป้อนสัญญาณเข้ามาที่จุดนี้แต่ถ้าต้องการใช้วงจรออสซิลเลเตอร์ภายในก็ให้ต่อ Crystal หรือ เซรามิคเรโซเนเตอร์ดังรูปที่ 4.11 คาปาซิเตอร์ในวงจรควรมีค่าประมาณ 20 PF



รูปที่ 4.11 วงจรออสซิลเลเตอร์ภายใน 8051

XTAL 2

ขาที่ 18 ของรูปที่ 4.5 ขานี้เป็นจุดเอาท์พุทของวงจรขยายแบบกลับเฟสสัญญาณที่ประกอบเป็นวงจรออสซิลเลเตอร์ (อินพุทคือขา XTAL 1)ถ้าจะใช้สัญญาณที่สร้างมาจากภายนอกมาเป็นสัญญาณนาฬิกาของ 8051 แล้ว ให้ปล่อยขานี้ลอยไว้แล้วป้อนสัญญาณนาฬิกาจากภายนอกเข้ามาที่ขา XTAL 1 ดังรูปที่ 4.12



รูปที่ 4.12 8051 ที่ทำงานโดยสัญญาณที่มาจากภายนอก

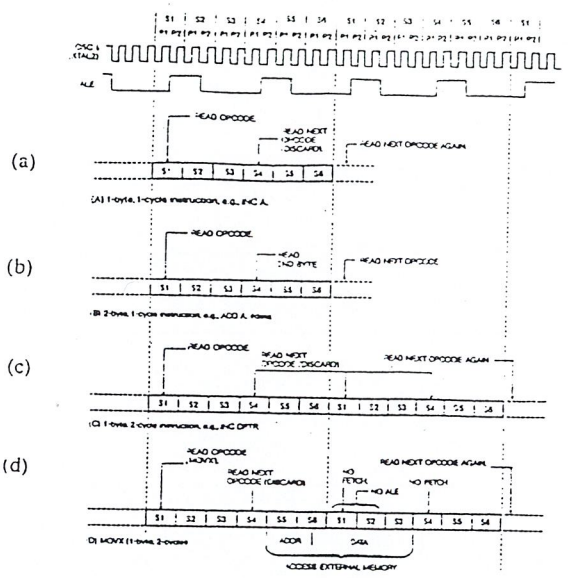
4.5 การทำงานของ 8051

คอมพิวเตอร์จะทำงานด้วยวงจรที่เรียกว่าฮาร์ดแวร์ประกอบขึ้นมาเพียงอย่างเดียวไม่ได้ ต้องมีโปรแกรมหรือคำสั่งที่จัดเรียงกันไว้เพื่อให้คอมพิวเตอร์ทำงานตามลำดับ ใน 8051 ก็เช่นเดียวกัน ผู้ใช้จะต้องเขียนโปรแกรมเป็นภาษาเครื่องซึ่งอยู่ในรูปเลขฐานสองเก็บไว้ในหน่วยความจำ ซึ่งแต่ละคำสั่งจะมีความยาวกี่ไบท์นั้นขึ้นอยู่กับคำสั่ง

จากรูปที่ 4.4 เมื่อเริ่มป้อนไฟเลี้ยงให้กับ 8051 ซึ่งมีวงจร Power on reset ต่ออยู่จะมีการรีเซทเกิดขึ้นการทำงานภายใน 8051 จะเริ่มจากโปรแกรมเคานเตอร์ซึ่งเป็นวงจรนับชนิดหนึ่ง ส่งตำแหน่งของหน่วยความจำลงบนบัสหมายเลขหนึ่ง บัสนี้มีขนาด 16 บิท ค่าหน่วยความจำนี้จะถูกส่งไปเก็บไว้ที่ Program address register ซึ่งเป็นวงจรแลทช์ข้อมูลซึ่งเป็นตำแหน่งหน่วยความจำ ซึ่งจะปรากฏบนบัส 16 บิทหมายเลขสอง ถ้าเป็นตำแหน่งหน่วยความจำแรกหลังจากการรีเซท ตำแหน่งหน่วยความจำจะเป็น 0000H หน่วยความจำโปรแกรมจะเลือกได้ว่าเป็น ROM ภายนอกหรือภายใน 8051 โดยการป้อนลอจิกเข้าไปทางขา **EA** ซึ่งต่ออยู่กับส่วน Timing and control ซึ่งทำหน้าที่เป็นวงจรถอดหัส แล้วสร้างสัญญาณควบคุมต่อไป ถ้าป้อนสัญญาณลอจิก 1 เข้าไปที่ขา **EA** จะเป็นการเลือก

ใช้ ROM ภายใน 8051 โดยที่วงจร Timing and control จะส้างสัญญาณไปยัง ROM ภายในให้เป็นข้อมูลที่เป็นคำสั่งจากตำแหน่งที่ถูกชี้ด้วยตำแหน่งที่ส่งมาทางบัสหมายเลขสอง ข้อมูลจาก ROM จะถูกส่งลงไปยังบัสหมายเลขสามที่เรียกว่า Internal data bus แล้วนำไปเก็บไว้ที่ Instruction register (เป็นวงจร Latch) เพื่อส่งต่อไปให้วงจร Timing and control ทำการถอดรหัสแล้วควบคุมการทำงานส่วนอื่นๆ ต่อไปแล้วแต่ว่าเป็นคำสั่งอะไร ในกรณีที่เลือกROM ภายนอก 8051 โดยป้อนลอจิก 0 เข้าที่ขา **EA** ทำให้วงจร Timing and control ส่งสัญญาณไปยังพอร์ท 0 และพอร์ท 2 เพื่อส่งค่าตำแหน่งหน่วยความจำบนบัสหมายเลขสองออกไปชี้หน่วยความจำภายนอก จากนั้นจะอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาทางพอร์ท 0 ไปยัง Internal data bus แล้วไปเก็บที่ Instruction register เพื่อทำงานต่อไปเหมือนกับตอนที่อ่านคำสั่งจาก ROM ภายใน การทำงานในช่วงส่งตำแหน่งหน่วยความจำไปยังหน่วยความจำแล้วอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาเก็บไว้ใน Instruction register เรียกว่าเป็นช่วงของการ Fetch ช่วงต่อไปจะเป็นช่วงของการทำงานตามคำสั่งเรียกว่าช่วง Execute เช่นถ้าเป็นคำสั่งให้บวกข้อมูลใน Accumulator กับข้อมูลจากหน่วยความจำ RAM ภายในตำแหน่ง 23H วงจร Timing and control ก็จะส่งสัญญาณให้ Instruction register ส่งตำแหน่งหน่วยความจำ 23H ลงไปยัง Internal data bus แล้วนำข้อมูลไปเก็บไว้ที่ RAM address register เพื่อใช้ชี้ตำแหน่งหน่วยความจำ RAM จากนั้นวงจร Timing and control จะสั่งให้ RAM ส่งข้อมูลที่เก็บอยู่ในหน่วยความจำตำแหน่ง 23H ลงมายัง Internal data bus แล้วนำข้อมูลไปเก็บไว้ที่ TMP1 (เป็นวงจร latch) ขณะเดียวกันวงจร Timing and control ก็จะส่งสัญญาณไปยัง Accumulator ให้ส่งข้อมูลมายัง TMP2 (เป็นวงจร latch) วงจร Arithmatic logic unit ซึ่งโครงสร้างเป็นวงจรทำการคำนวณทางคณิตศาสตร์ (บวก ลบ คูณ หาร) และยังสามารถประมวลผลทางลอจิก (AND,OR,NOT,XOR) จะทำการบวกเลขจาก TMP1 และ TMP2 เข้าด้วยกัน ผลลัพธ์ที่ได้จะส่งผ่าน Internal data bus กลับไปยัง Program status word ซึ่งจะทำหน้าที่เก็บสถานะผลลัพธ์ของการทำงานใน Arithmatic logic unit เช่น ผลลัพธ์ในการบวกมีค่าเกิน 8 บิท ก็จะทำให้บิทที่หนึ่งใน Program status word ถูกเซ็ท

การทำงานที่กล่าวมาข้างต้นจะขึ้นกับสัญญาณควบคุมที่สร้างมาจากวงจร Timing and control และสัญญาณที่สร้างขึ้นอ้างอิงกับสัญญาณนาฬิกาที่สร้างมาจากวงจร Oscillator ทำให้การทำงานต่างๆ เป็นดังรูปที่4.13



รูปที่ 4.13 ลำดับสถานะการทำงานใน MCS-51

คำสั่งแต่ละคำสั่งของ 8051 จะใช้เวลาทำงาน 1-3 แมชชีนไซเคิล แล้วแต่ว่าเป็นคำสั่งใด โดยหนึ่งแมชชีนไซเคิลจะใช้เวลา 12 ไซเคิลของสัญญาณนาฬิกา ดังนั้นแต่ละคำสั่งของ 8051 จะใช้เวลาทำงาน 12-36 ไซเคิลของสัญญาณนาฬิกา แต่ละแมชชีนไซเคิลจะแบ่งออกเป็น 6 เสตท คือ S1 ,S2 ,S3 ,S4 ,S5 และS6 แต่ละเสตทจะประกอบด้วยสองไซเคิลของสัญญาณนาฬิกา เรียกไซเคิลแรกว่า เฟส 1 (P1) และเรียกไซเคิลที่สองว่า เฟส 2 (P2) ใน
แต่ละเฟสจะนับตั้งแต่ขอบขาลงของสัญญาณนาฬิกาถึงขอบขาลงของสัญญาณนาฬิกาถัดไปดังรูปที่ 4.13 เมื่อ8051 ทำงานเสร็จหนึ่งแมชชีนไซเคิลก็จะเริ่มทำงานเสตท 1 เฟส 1 ของไซเคิลถัดไป ในหนึ่งแมชชีนไซเคิลวงจร Timing and control จะสร้างสัญญาณ ALE ออกมาสองไซเคิลเพื่อ Fetch คำสั่งเข้าไปสองครั้งเสมอ ที่บริเวณขอบขาขึ้นของสัญญาณ ALE คำสั่งใดมีกี่ไบท์สามารถดูได้จากตารางชุดคำสั่ง 8051

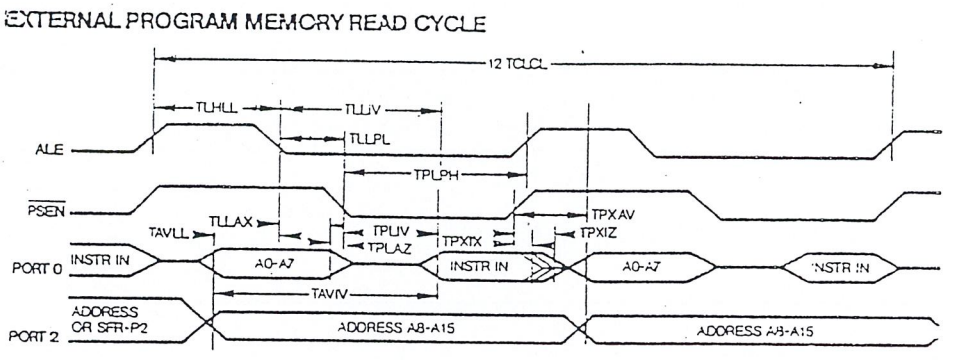
คำสั่งประเภทหนึ่งไบท์หนึ่งแมชชีนไซเคิล ได้แก่ คำสั่ง INC A จะมีการอ่านคำสั่งจากหน่วยความจำโปรแกรมสองครั้ง ที่เวลาประมาณขอบขาขึ้นของสัญญาณ ALE แรก แล้วนำไปเก็บที่ Instruction register เพื่อให้วงจร Timing and control ถอดรหัส แล้วเข้าสู่การ Execute ขณะเดียวกันก็จะเริ่มต้นการ Fetch คำสั่งที่อยู่ในหน่วยความจำตำแหน่งถัดไปเข้ามา และคำสั่งที่สองจะถูกอ่านเข้ามาที่เวลาขอบขาขึ้นของสัญญาณ ALEถัดไป วงจร Timing and control เมื่อถอดรหัสคำสั่งแรกก็ทราบว่าการทำงานคำสั่งนี้ให้สิ้นสุดจะใช้คำสั่งเพียงหนึ่งไบท์ ดังนั้นคำสั่งที่ถูกอ่านมาไบท์ที่สองจะไม่ถูกนำมาทำงาน เพียงแต่อ่านเข้ามาแล้วทิ้งไปดังรูปที่ 4.13 a

คำสั่งประเภทสองไบท์หนึ่งแมชชีนไซเคิล ได้แก่ คำสั่ง ADD A,#DATA ในหนึ่งแมชชีนไซเคิลจะมีการอ่านคำสั่งเข้ามาสองไบท์เหมือนกับคำสั่งประเภทหนึ่งไบท์หนึ่งแมชชีนไซเคิล แตกต่างกันที่ไบท์ที่สองจะถูกนำมาใช้งานด้วยไม่ได้ทิ้งไปดังรูปที่ 4.13 b ตัวอย่างเช่น คำสั่ง ADD A,#33H สามารถเขียนเป็นภาษาเครื่องได้สองไบท์ คือ
24 และ 33 เมื่ออ่านคำสั่งไบท์แรก คือ 24 เข้าไปไว้ที่ Instruction register แล้ววงจร Timing and control จะถอดรหัสพบว่าเป็นคำสั่งบวกเลข ก็จะส่งสัญญาณไปยัง Accumulator ให้เอาข้อมูลไปไว้ที่ TMP1 เมื่อคำสั่งที่สองถูกอ่านเข้ามาที่ Instruction register แล้ววงจร Timing and control จะสั่งให้เอาข้อมูลไบท์ที่สองส่งลงไปยัง Internal data bus ไปเก็บยัง TMP1 จากนั้นวงจร Arithmatic logic unit จะนำเอาข้อมูลใน TMP1 และ TMP2 มาบวกกัน ผลลัพธ์ที่ได้จะส่งออกจากวงจร Arithmatic logic unit ไปยัง Internal data bus แล้วไปเก็บไว้ที่ Accumulator

คำสั่งประเภท 1,2 หรือ3 ไบท์ที่ใช้เวลาทำงานสองแมชชีนไซเคิล เช่น คำสั่ง INC DPTR จะมีการอ่านคำสั่งเข้าไปสี่ครั้งทุกๆ ขอบขาขึ้นของสัญญาณ ALE ที่มีสองครั้งต่อหนึ่งแมชชีนไซเคิล ถ้าเป็นคำสั่งประเภท 1,2 หรือ3 ไบท์ วงจร Timing and control จะเอาคำสั่ง 1,2 หรือ3 ไบท์แรกเท่านั้นไปทำงานส่วนคำสั่งที่เหลือทิ้งไปดังรูปที่ 4.13 C คำสั่งหนึ่งไบท์ที่ใช้เวลาทำงานสองแมชชีนไซเคิลที่กล่าวมาแล้วจะไม่รวมถึงคำสั่ง MOVX ซึ่งใช้ในการอ่านหรือเขียนข้อมูลกับหน่วยความจำข้อมูลภายนอก การทำงานของคำสั่งนี้จะมีการ Fetch คำสั่งเข้าไปสองไบท์ในแมชชีนไซเคิลแรก ในแมชชีนไซเคิลที่สองจะไม่มีการ Fetch คำสั่งเข้าไปแต่จะเป็นช่วงเวลาของการอ่านหรือเขียนข้อมูลกับหน่วยความจำข้อมูลภายนอก สัญญาณ ALE ซึ่งปกติจะเปลี่ยนเป็น 1 ที่เสตท 1 เฟส 2 ก็จะไม่เปลี่ยนเป็น 1 ในแมชชีนไซเคิลที่สอง โดยจะเป็น 0 อยู่จนกว่าจะถึงเสตท 4 เฟส 2 ของแมชชีนไซเคิลที่สอง สัญญาณ ALE จะเปลี่ยนเป็น 1 เพื่อทำการอ่านหรือเขียนข้อมูลกับหน่วยความจำข้อมูลภายนอก

4.6 ไดอะแกรมเวลาของการติดต่อกับหน่วยความจำ

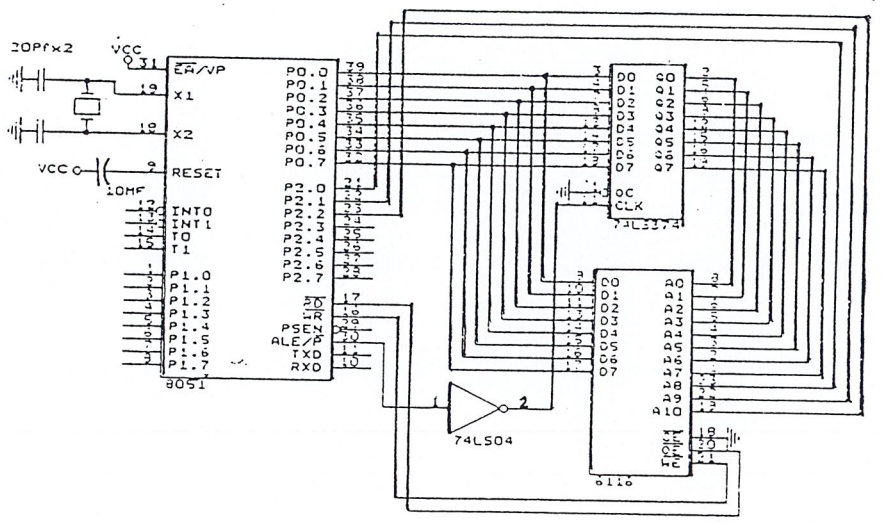
การอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกสำหรับ 8051 นั้น ลำดับสัญญาณตามเวลา (Timing diagram) ของการอ่านคำสั่งดังรูปที่ 4.14



รูปที่ 4.14 Timing diagram ของการอ่านโปรแกรมจากหน่วยความจำภายนอก

การอ่านคำสั่ง (Fetch) จาก Program area ภายนอกจะเริ่มจาก 8051 ส่งสัญญาณลอจิก 1 ออกมา

ทางขา ALE ขณะนี้ขาสัญญาณ **PSEN** จะเป็น 1 จากนั้นพอร์ท 0 จะส่งตำแหน่งหน่วยความจำ 8 บิทล่างและพอร์ท 2 จะส่งตำแหน่งหน่วยความจำ 8 บิทบนออกมาแล้วสัญญาณ ALE จะกลับเป็น 0 อุปกรณ์ภายนอกจะสามารถใช้ขอบขาลงของสัญญาณ ALE เพื่อแลทช์ตำแหน่งของหน่วยความจำที่พอร์ท 0 ไว้ จากนั้นพอร์ท 0 จะยกเลิกการส่งตำแหน่งหน่วยความจำเข้าสู่สภาวะ High impedance และสัญญาณ **PSEN** จะเป็น 0 เพื่อเตรียมรับคำสั่งที่ส่งออกจากหน่วยความจำภายนอกเข้าไปยัง 8051 เพื่อทำงานต่อไป เมื่อคำสั่งถูกอ่านเข้าไปเก็บใน Instruction register แล้วสัญญาณ **PSEN** จะกลับเป็น 1 พร้อมกับสัญญาณ ALE ก็จะกลับเป็น 1 เพื่อการอ่านคำสั่งต่อไป ข้อมูลในพอร์ท 2 จะคงที่ตลอดเวลาตั้งแต่สัญญาณ ALE เป็น 1 จนกระทั่งสัญญาณ ALE เปลี่ยนเป็น 0 และกลับเป็น 1 อีกครั้งหนึ่งจากนั้นจะเริ่มลำดับการ Fetch ข้อมูลไบท์ที่ 2 จากหน่วยความจำโปรแกรม ซึ่งจะมีการเปลี่ยนแปลงของสัญญาณตามเวลาเหมือนกับการ Fetch ไบท์แรกนั่นเอง จาก Timing diagram ดังกล่าวจะออกแบบวงจรที่มีหน่วยความจำโปรแกรมอยู่ภายนอก 8051 ดังตัวอย่างในรูปที่ 4.15



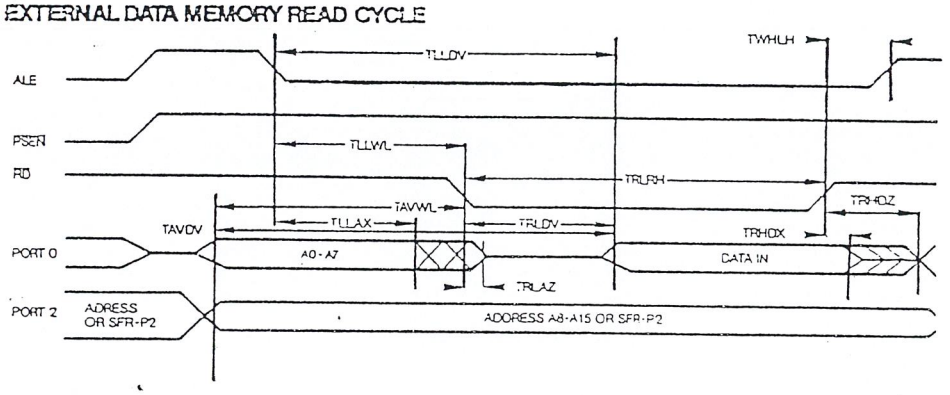
รูปที่ 4.15 วงจรที่มีหน่วยความจำโปรแกรมอยู่ภายนอก 8051

74LS374 ในรูปที่ 4.15 จะทำหน้าที่แลทช์ตำแหน่งในหน่วยความจำ 8 บิทล่างที่เวลาขอบขาลงของสัญญาณ ALE ซึ่งสัญญาณ ALE จะถูกกลับให้เป็นตรงข้ามโดยอินเวอร์เตอร์ 74LS04 ก่อนที่จะป้อนให้ขา CK ของ74LS374 และที่ขอบขาขึ้นของสัญญาณที่ออกจาก 74LS04 จะแลทช์ตำแหน่งหน่วยความจำ ข้อมูลที่ออกจาก74LS374 จะเป็นค่า 8 บิทล่างของตำแหน่งหน่วยความจำที่ต้องการติดต่อ ในวงจรได้ต่อตำแหน่งหน่วยความจำ 8 บิทเข้ากับ A0-A7 ของ EPROM และข้อมูลจากพอร์ท 2 บิท P2.0-P2.3 จะต่อเข้ากับ A8-A11 ของ EPROM โดยตรงเพราะตำแหน่งหน่วยความจำ 8 บิทบนจากพอร์ท 2 จะคงที่ตลอดเวลา ขา **PSEN** ของ8051จะถูกต่อเข้ากับขา OE ของ EPROM 2716 ดังนั้นเมื่อ

สัญญาณ **PSEN** มีลอจิกเป็น 0 ก็จะส่งคำสั่งที่เก็บใน EPROM ตำแหน่งที่ชี้ข้อมูลโดยA0-A11 ออกมายังพอร์ท 0 และถูก 8051 นำไปทำงานต่อไป

การอ่าน – เขียนข้อมูลกับหน่วยความจำข้อมูลภายนอก 8051

การอ่าน – เขียนข้อมูลกับหน่วยความจำข้อมูลภายใน 8051 นั้นจะมีสัญญาณสร้างมาจากส่วนTiming and control โดยที่ผู้ใช้ไม่จำเป็นต้องทำความเข้าใจ แต่การอ่าน – เขียนข้อมูลกับหน่วยความจำข้อมูลภายนอก อันเนื่องมาจากคำสั่ง MOVX นั้น เมื่อคำสั่งดังกล่าวถูกอ่านมายัง Instruction register แล้ว Timing and control จะทำการถอดรหัสแล้วสร้างสัญญาณควบคุม โดยการอ่านข้อมูลกับหน่วยความจำข้อมูลภายนอกจะมี Timing diagram ดังรูปที่ 4.16

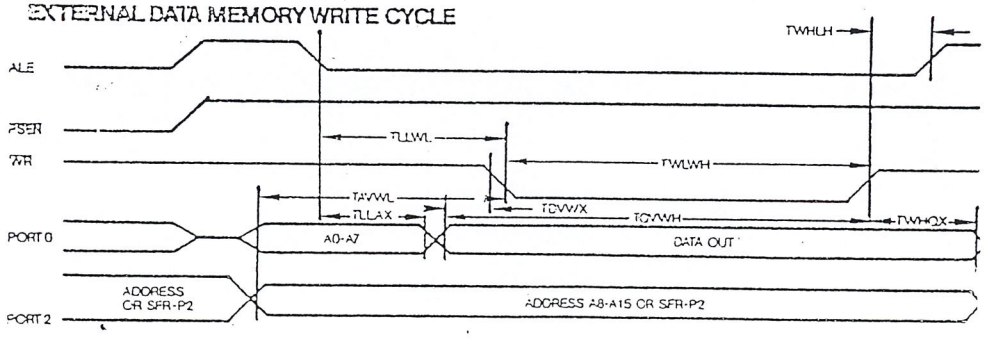


รูปที่ 4.16 Timing diagram สำหรับการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอก 8051

การทำงานจะเริ่มจากการส่งตำแหน่งหน่วยความจำภายนอก 8 บิทล่างออกมาทางพอร์ท 0 และ 8 บิทบนออกมาทางพอร์ท 2 เมื่อส่งตำแหน่งแล้วสัญญาณ ALE ซึ่งเดิมมีลอจิก 1 จะกลับมาเป็น 0 เพื่อให้อุปกรณ์ภายนอกสามารถแลทช์ตำแหน่งหน่วยความจำไว้เหมือนกับการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก 8051 เพื่อส่งไปยังหน่วยความจำ แม้ว่าข้อมูลบนพอร์ท 0 จะเปลี่ยนแปลงไปก็จะยังมีตำแหน่งหน่วยความจำส่งไปยังหน่วยความจำ ในระหว่างการติดต่อกับหน่วยความจำข้อมูล

นี้สัญญาณ **PSEN** จะเป็น 1 ตลอดเวลา สัญญาณ **PSEN** จะแอกทีฟ (เป็น 0) ก็ต่อเมื่อเป็นการติดต่อกับหน่วยความจำโปรแกรมภายนอก 8051 เท่านั้น 8051 จะส่งสัญญาณออกมาทาง **RD** (P3.7) เพื่อบอกกับหน่วยความจำภายนอกว่าต้องการอ่านข้อมูลเข้าไป เมื่อ 8051 ส่งสัญญาณออกมาทาง **RD** เป็นลอจิก 0 จะทำให้พอร์ท 0 เข้าสู่สถานะ High impedance พร้อมที่จะให้หน่วยความจำภายนอกส่งข้อมูลภายนอกมาบนพอร์ท 0 ข้อมูลบนพอร์ท 0 ซึ่งส่งมาจากหน่วยความจำภายนอกจะถูกอ่านเข้าไปเก็บที่ขอบขาขึ้นของสัญญาณ **RD** จากนั้น ALE ก็จะสามารถกลับเป็น 1 เพื่อเริ่มการทำงานในคำสั่งต่อไป ในระหว่างการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอกนี้ พอร์ท 2 จะส่งตำแหน่งหน่วยความจำ 8 บิทบนออกมาตลอดเวลา

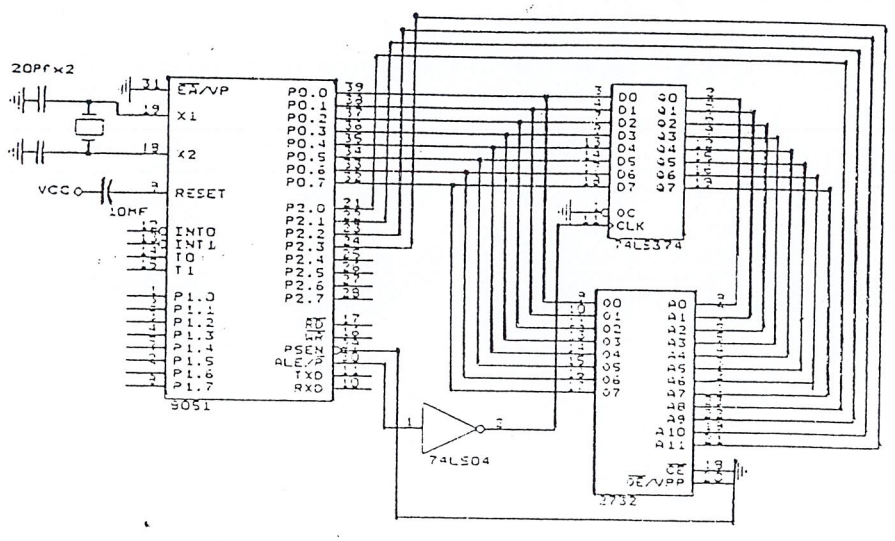
การเขียนข้อมูลไปยังหน่วยความจำข้อมูลภายนอก 8051 จะมี Timing diagram ดังรูปที่ 4.17



รูปที่ 4.17 Timing diagram ของการเขียนข้อมูลไปยังหน่วยความจำข้อมูลภายนอก 8051

เมื่อ 8051 ส่งค่าตำแหน่งหน่วยความจำ 8 บิทล่างออกมาทางพอร์ท 0 และ8บิท บนออกมาทางพอร์ท 2 แล้วสัญญาณ ALE จะกลับเป็น 0 อุปกรณ์ภายนอกจะสามารถใช้สัญญาณนี้ในการแลทช์ตำแหน่งหน่วยความจำบนพอร์ท 0 เหมือนกับการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอก เมื่อ

สัญญาณ ALE เป็น 0 แล้ว 8051 จะส่งข้อมูลที่ต้องการเขียนไปยังพอร์ท 0 แล้วจะให้สัญญาณ **WR** เปลี่ยนสภาวะเป็นลอจิก 0 ขณะนี้หน่วยความจำภายนอกจะต้องเขียนข้อมูลไปเก็บยังตำแหน่งที่กำหนด จากนั้นสัญญาณ **WR** จะกลับเป็น 1 เพื่อเป็นการบอกสิ้นสุดการเขียนข้อมูล แล้วสัญญาณ ALE ก็จะกลับเป็น 1 เพื่อ Fetch คำสั่งต่อไปมาทำงาน ซึ่งหน่วยความจำข้อมูลภายนอกที่สามารถอ่านและเขียนข้อมูลได้ จะสามารถเขียนเป็นวงจรได้ดังรูปที่ 4.18



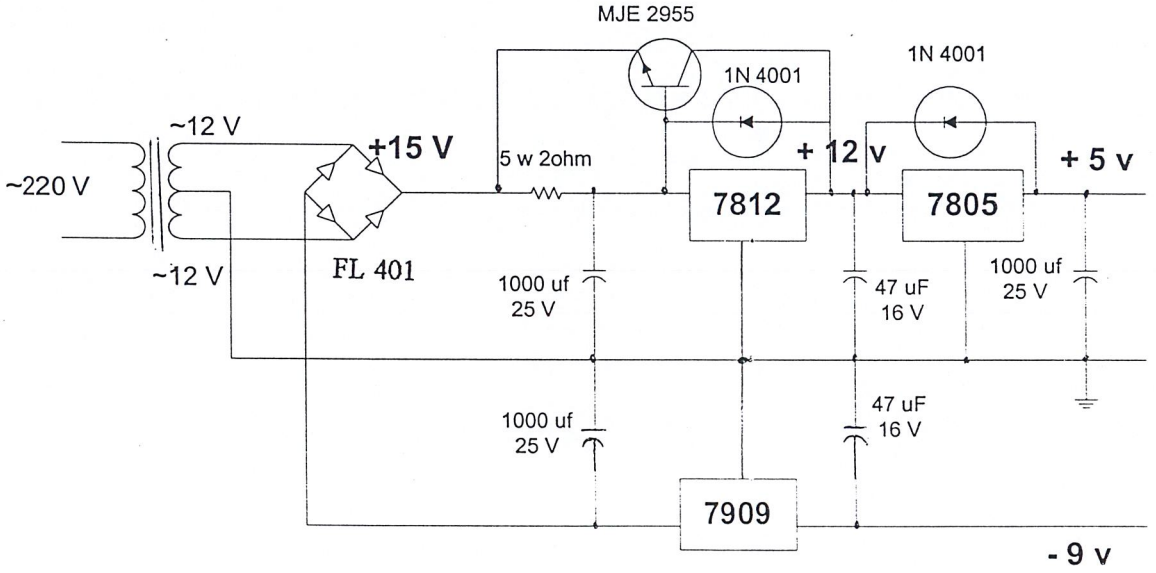
รูปที่ 4.18 วงจรซึ่งมีหน่วยความจำข้อมูลภายนอก 8051

74LS374 ในรูปจะใช้สำหรับแลทช์ตำแหน่งหน่วยความจำ 8 บิทล่างไว้ แม้ว่าข้อมูลบนพอร์ท 2 จะเปลี่ยนไป สัญญาณ **RD** และ **WR** จะอ่านและเขียนข้อมูลจากหน่วยความจำภายนอก 6116 เป็นหน่วยความจำแบบ RAM ที่สามารถจะอ่านและเขียนข้อมูลได้

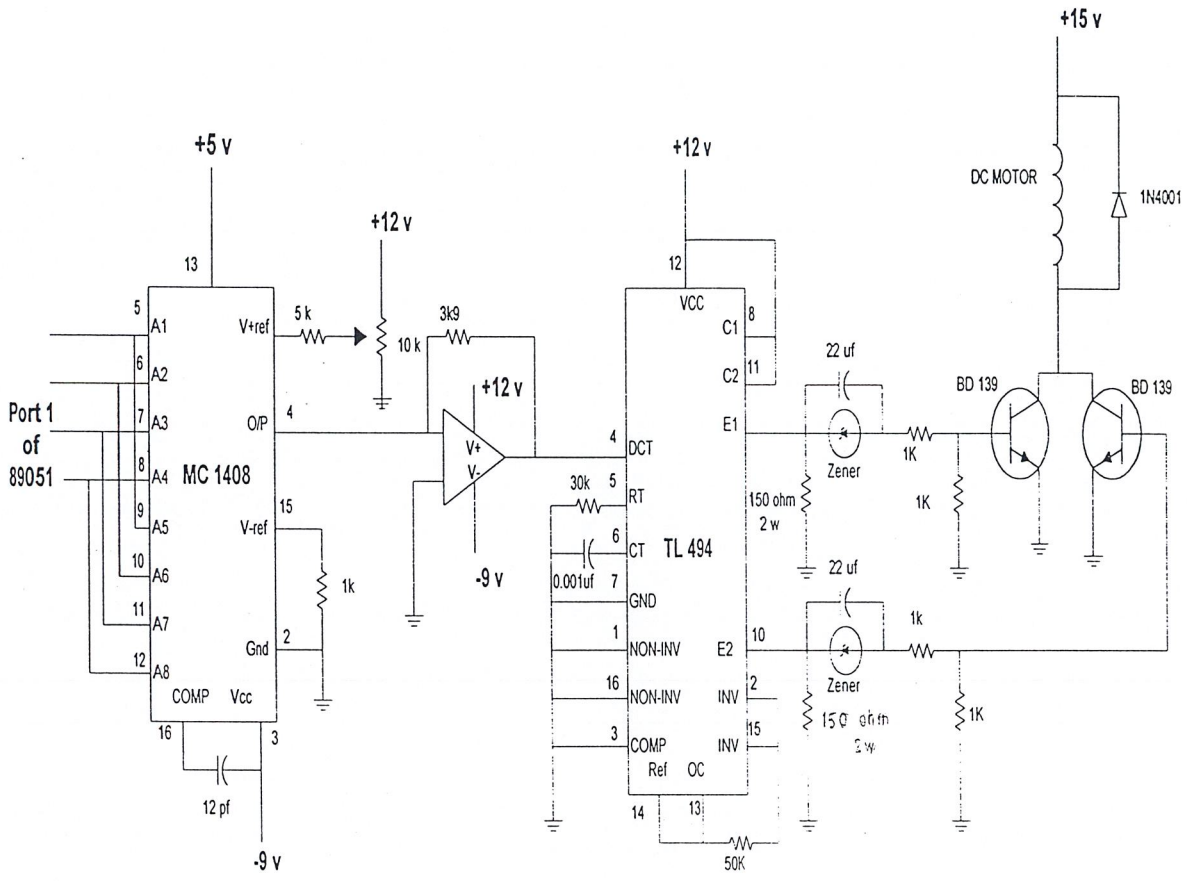
บทที่ 5
การออกแบบควบคุมรถและอัลกอริทึมการทำงานของโปรแกรม

5.1 การทำงานของวงจรควบคุมการขับเคลื่อนรถ (Driver Control)

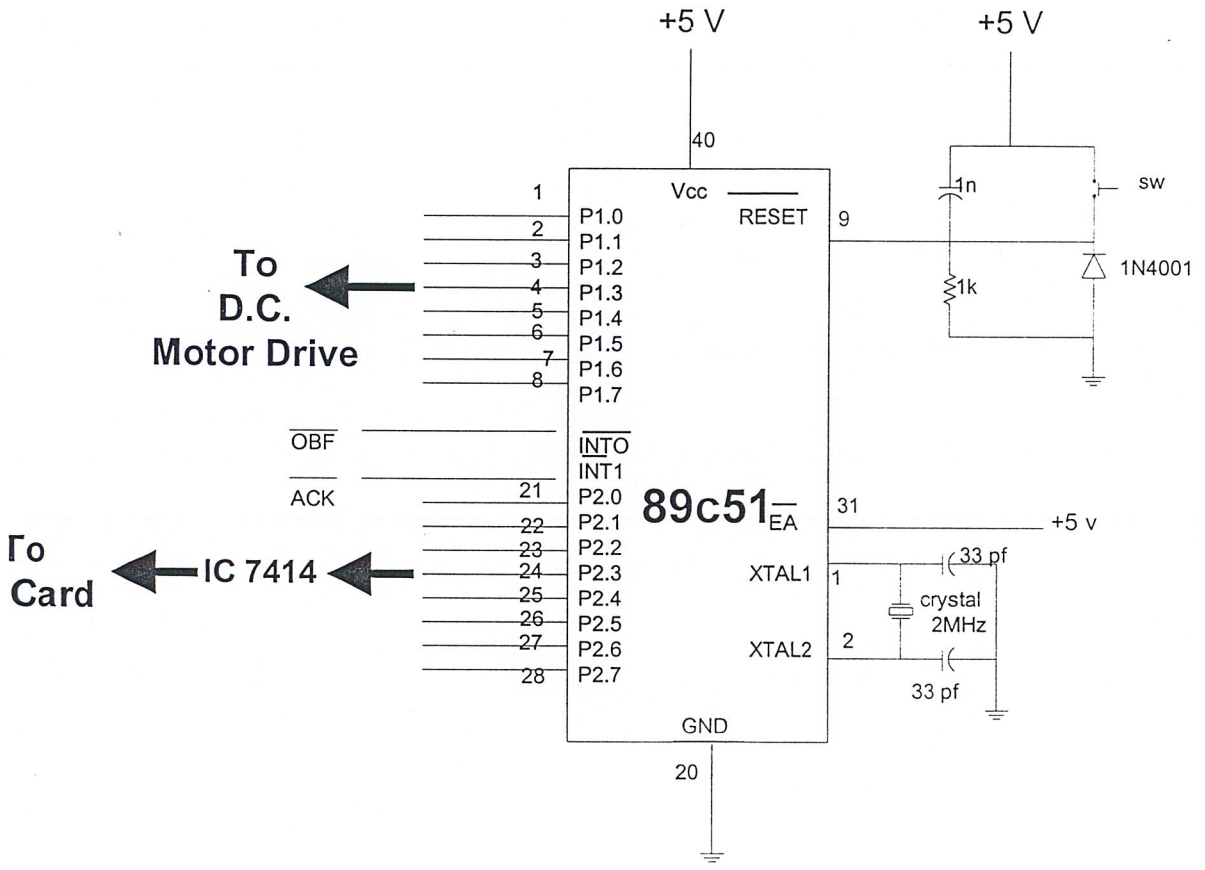
การทำงานของวงจรขับเคลื่อนมอเตอร์กระแสตรงเริ่มจากไมโครคอนโทรลเลอร์ 89C51 ส่งค่าความเร็วของมอเตอร์ทั้งสองผ่านทางพอร์ทหนึ่งไปยังไอซี MC1408 ซึ่งทำหน้าที่แปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก แล้วส่งสัญญาณอนาลอกที่ได้ไปยังไอซี TL494 ซึ่งเป็นวงจรพัลส์วิธมอดูเลเตอร์ ซึ่งเอาท์พุทที่ได้จะนำไปขับทรานซิสเตอร์ BD139 เพื่อขับกระแสให้มอเตอร์กระแสตรงทั้งสองต่อไป โดยเราจะใช้โปรแกรม Visual C++ เขียนโปรแกรมเพื่อควบคุมการ์ดพอร์ท 8255 ในการติดต่อกับไมโครคอนโทรลเลอร์ โดยวงจรที่ใช้ในการควบคุมการขับเคลื่อนมอเตอร์กระแสตรงนั้น ดังแสดงให้เห็นในรูปหน้าถัดๆ ไป



รูปที่ 5.1 วงจรแหล่งจ่ายไฟกระแสตรง +15 V., +12 V., +5 V. และ –9 V.

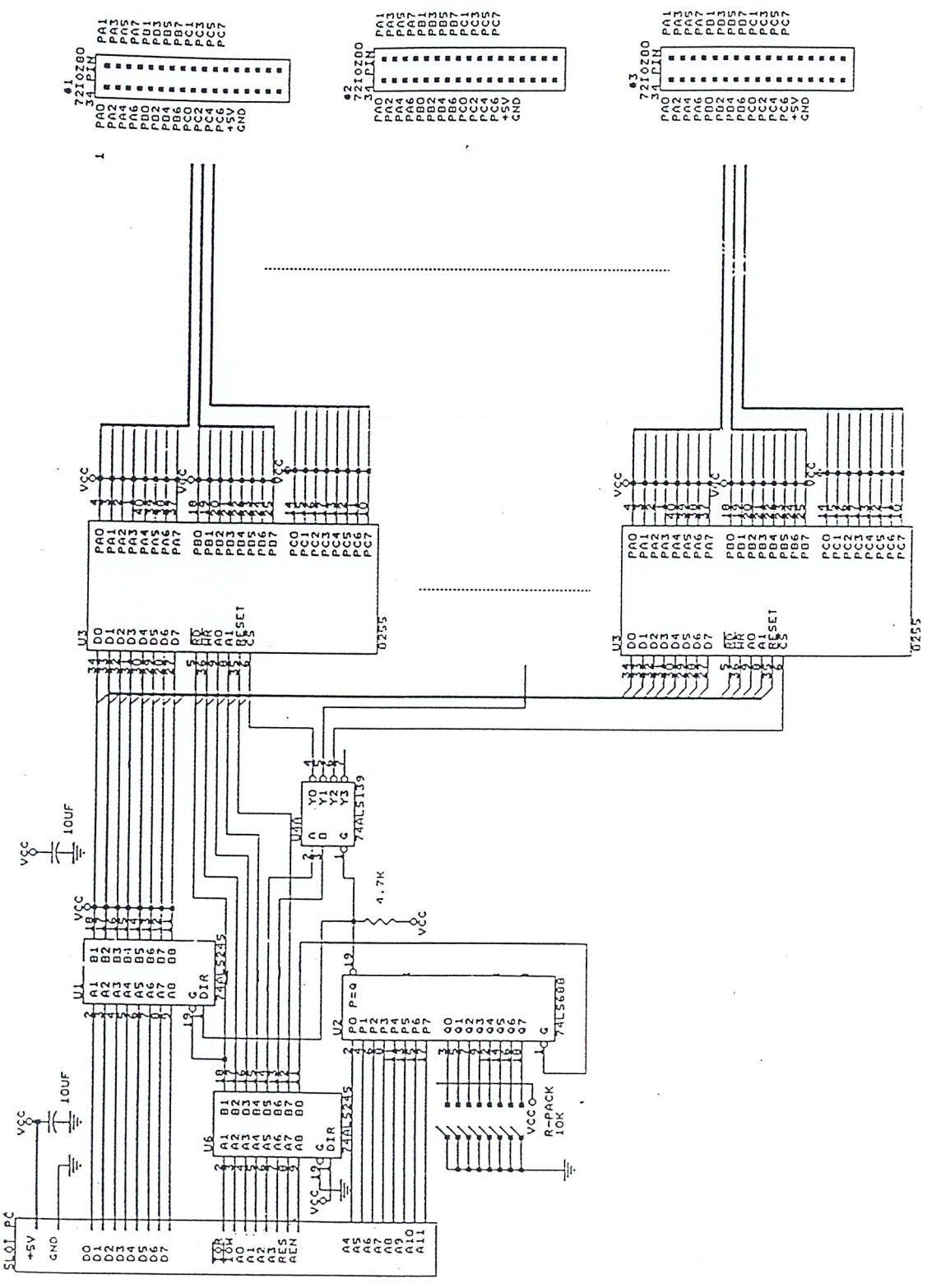


รูปที่ 5.2 แสดงส่วนวงจรขับเคลื่อนมอเตอร์กระแสตรง



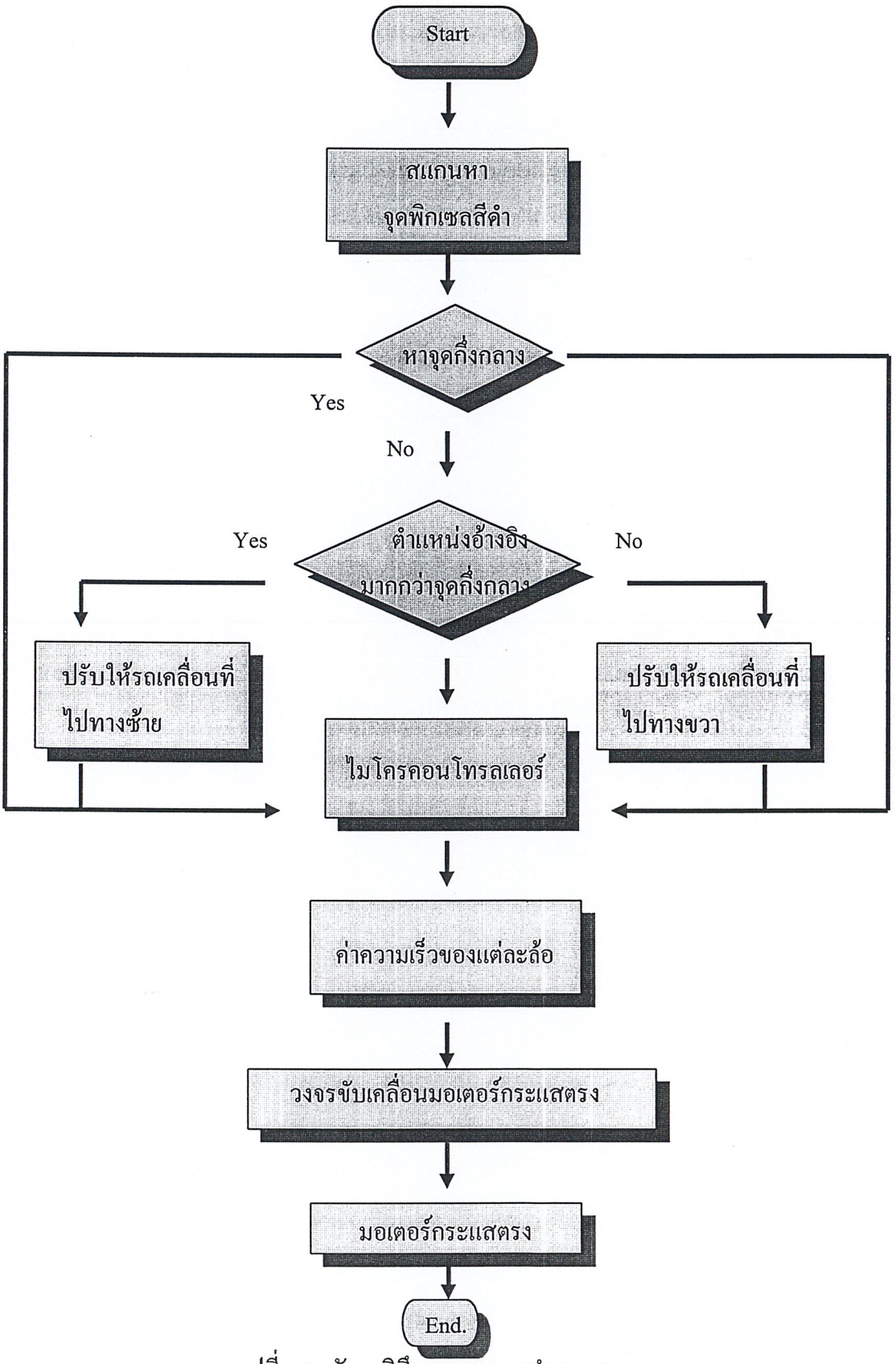
รูปที่ 5.3 แสดงวงจรส่วนไมโครคอนโทรลเลอร์ 89C51

รูปที่ 5.4 แสดงการเชื่อมต่ออุปกรณ์ของการ์ดพอร์ท 8255



5.2 อัลกอริทึมการทำงานในส่วนต่างๆ ของโปรแกรม

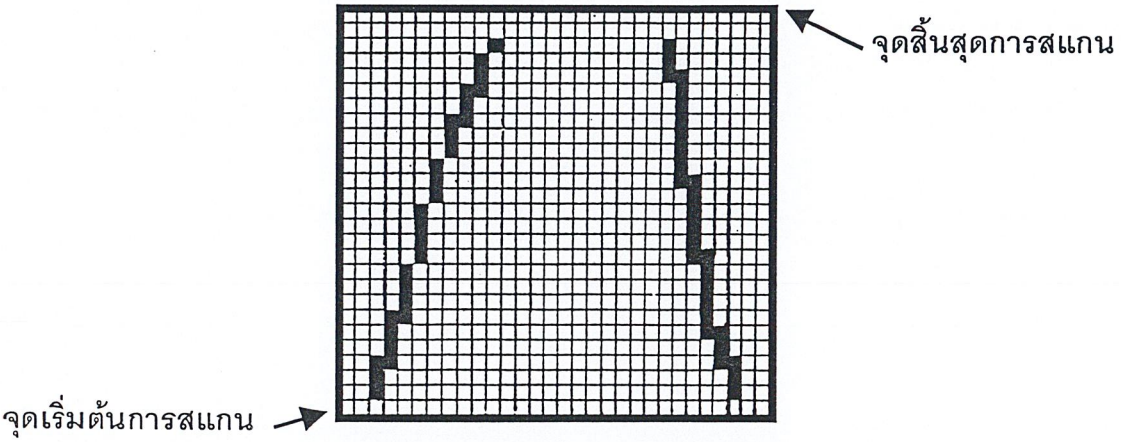
ในการควบคุมการขับเคลื่อนรถนั้น เราจะใช้โปรแกรมซึ่งเขียนขึ้นมาด้วยโปรแกรม Visual C++ เพื่อทำการประมวลผลสัญญาณภาพเพื่อหาจุดกึ่งกลางเลน แล้วนำไปวิเคราะห์หาระยะที่เบี่ยงเบนจากจุดกึ่งกลางเลนเพื่อส่งต่อไปให้ไมโครคอนโทรลเลอร์ และนำมาประมวลผลเพื่อควบคุมไมโครคอนโทรลเลอร์ต่อไป



รูปที่ 5.5 อัลกอริทึมควบคุมการทำงานของรถ

5.2.1 การหาจุดกึ่งกลางเลน (Find Navigatior Point)

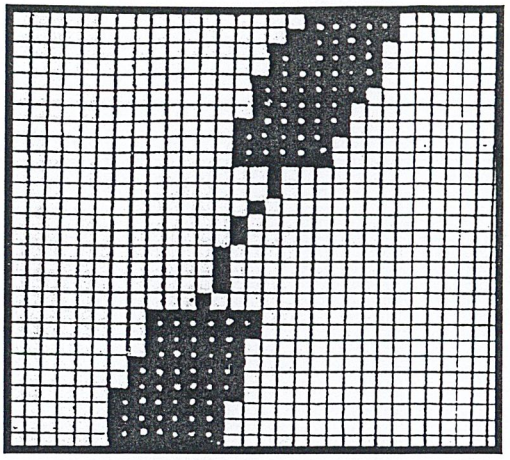
ภาพที่ได้จากการผ่านกระบวนการธินนิ่ง(Thinning)แล้วนั้น จะเป็นภาพที่ได้ของขอบถนนซึ่งมีความบางมากเท่ากับ 1 พิกเซล (pixel) หรืออาจมากกว่า 1 พิกเซลก็ได้ โดยจากภาพที่ได้จะนำมาหาจุดกึ่งกลางเลน โดยโปรแกรมจะทำการสแกนแต่ละพิกเซล หากพบพิกเซลสีขาวมันจะไม่สนใจ และข้ามพิกเซลนั้นไป แต่ถ้าหากพบพิกเซลสีดำ โปรแกรมจะทำการเก็บค่าตำแหน่งของพิกเซลนั้นไว้ แล้วทำการเพิ่มค่าตัวนับของพิกเซลนั้น ทั้งนี้ขึ้นอยู่กับความชัดเจนของภาพ โดยในแถวเดียวกันอาจมีพิกเซลมากกว่า 2 พิกเซลก็ได้ เพราะถ้าหากมีความผิดพลาดของภาพมาก ทำให้พิกเซลที่ได้มีมากกว่า 1 พิกเซลก็ได้



รูปที่ 5.6 ภาพที่ได้จากการทำธินนิ่งทั้งเส้น

เมื่อผ่านกระบวนการธินนิ่งแล้ว โปรแกรมจะทำการตรวจสอบพิกเซลโดยจะทำการตรวจสอบเฉพาะพิกเซลที่เป็นสีดำ โดยแบ่งตามจำนวนตัวนับได้ 3 กรณี ดังนี้

- **กรณีที่ตัวนับพิกเซลเท่ากับ 1** กรณีนี้บริเวณที่ทำการธินนิ่งจะเป็นเส้นตรง โดยที่เส้นของถนนเป็นเส้นตรงและที่บริเวณเส้นแบ่งเลนเป็นเส้นประซึ่งทำให้มีการขาดเป็นช่วงๆ หรืออาจเกิดจากการที่ขอบของเลนถนนได้หลุดออกจากสโคปที่กล้องจะจับภาพได้
- **กรณีที่ตัวนับพิกเซลเท่ากับ 2** อาจเกิดจากเส้นแบ่งเลนส์ของถนนทั้งสองเส้นปรากกฏอยู่ในขอบเขตของกล้อง หรืออาจเกิดจากมีเส้นแบ่งเลนมีเพียงเส้นเดียว ซึ่งจะได้ 1 พิกเซล ส่วนอีกพิกเซลเป็นเกิดจากความผิดพลาดของข้อมูลที่รับเข้ามา
- **กรณีที่ตัวนับพิกเซลมากกว่า 2** เกิดมาจากความผิดพลาดของข้อมูลโดยอาจเกิดจากแลงที่สะท้อนเข้ามาหรือเกิดจากสิ่งแวดล้อมที่ไม่เหมาะสม โดยทำให้เกิดพิกเซลมากกว่า 2 พิกเซลได้ ซึ่งทำให้เกิดความผิดพลาดของข้อมูลได้



รูปที่ 5.7 แสดงการทำธินนิ่งเพื่อให้ได้เส้นสีดำบางๆเพียง 1 พิกเซล

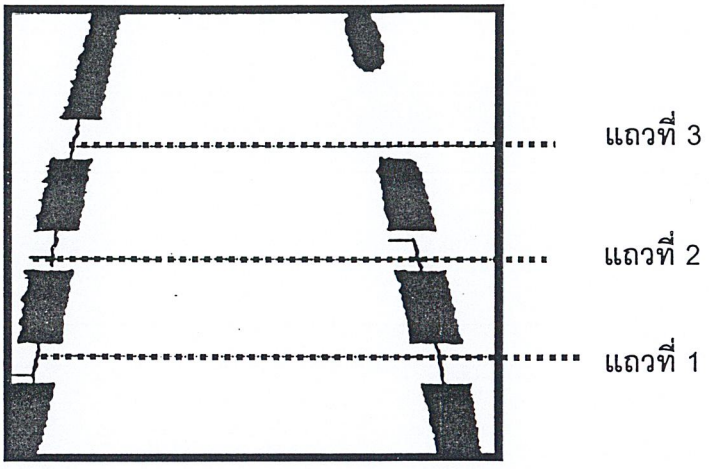
จากภาพที่ แสดงการทำธินนิ่งในกรณีต่างๆ คือ กรณีหาพิกเซลสีดำได้ 2 จุด , มากกว่า 2 จุด และกรณีที่พบเพียง 1 พิกเซล (เมื่อพบเส้นแบ่งเลนเป็นเส้นประ)

อธิบายได้ดังนี้

◆ กรณีที่มี 1 พิกเซล โปรแกรมจะทำการนับรวมเฉพาะแถวที่ตัวนำพิกเซลสีดำมีค่าเท่ากับหนึ่งเพื่อนำไปหาค่าเฉลี่ยของตำแหน่งพิกเซลสีดำของเส้นแบ่งเลน แล้วให้ค่าตำแหน่งพิกเซลบนเส้นแบ่งเลนที่เหลือเท่ากับ 0

◆ กรณีที่มี 2 พิกเซล โปรแกรมจะทำการคำนวณระยะห่างระหว่างตำแหน่งของพิกเซลทั้งสองว่าใกล้เคียงกับระยะห่างจริงของเส้นแบ่งเลนทั้งสองหรือไม่ ถ้าหากมีขนาดใกล้เคียง โปรแกรมก็จะทำการเก็บค่าทั้งสอง และระยะห่างระหว่างพิกเซลไว้ แล้วทำการเพิ่มค่าตัวนับพิกเซลสีดำที่มีค่าเท่ากับสอง เพื่อทำการประมวลผลข้อมูลต่อไป

◆ กรณีที่มีมากกว่า 2 พิกเซล โปรแกรมจะถือว่าข้อมูลที่ป้อนเข้ามามีความผิดพลาด จะไม่นำค่าต่างๆ มาทำการประมวลผล



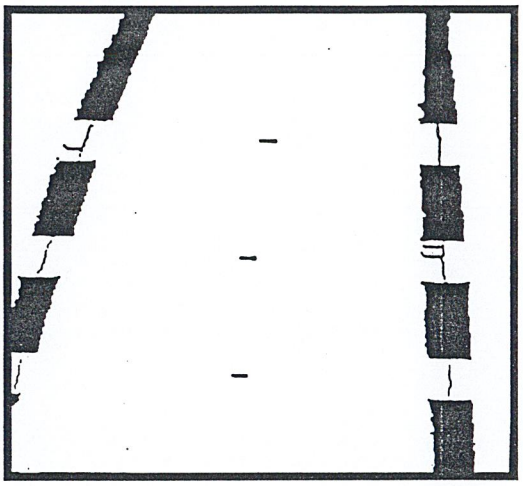
รูปที่ 5.8 แสดงกรณีทั้งสามของการทำธินนิ่ง

5.2.2 การหาค่าเฉลี่ยของจุดกึ่งกลางเส้นแบ่งเลน

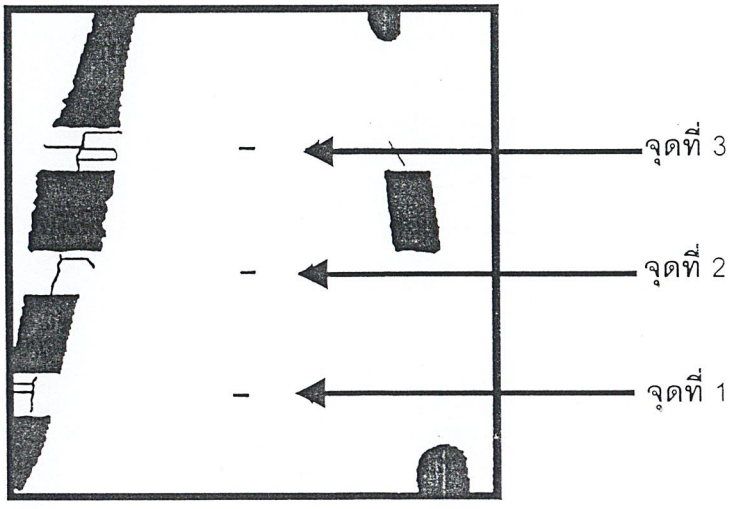
5.2.2.1 กรณีที่มีจุดบนเส้นแบ่งเลนเพียงเส้นเดียว โปรแกรมจะใช้ระยะห่างระหว่างจุดบนเส้นแบ่งเลนที่โปรแกรมสามารถคำนวณได้จากภาพที่สมบูรณ์ล่าสุด นำมาคำนวณร่วมกับค่าตำแหน่งของจุดใหม่ที่โปรแกรมสามารถหาได้ โดยการเปรียบเทียบกับจุดบนเส้นแบ่งเลนทั้งสองว่าจุดใหม่ที่ได้นี้มีตำแหน่งใกล้เคียงกับจุดใดมากที่สุดก็จะเป็นตำแหน่งของจุดบนเส้นแบ่งเลนนั้น โดยแบ่งเป็น 2 กรณี คือ

- ถ้าหากเป็นจุดที่อยู่บนเส้นแบ่งเลนเส้นซ้ายของเลนถนนทำการคำนวณโดยอาศัยระยะห่างระหว่างจุดบนเส้นแบ่งเลนทั้งสองนั้นรวมกับจุดบนเส้นแบ่งเลนใหม่ที่ได้
- กรณีที่เป็นจุดที่อยู่บนเส้นแบ่งเลนเส้นขวา จะทำการลบออกจากตำแหน่งของจุดบนเส้นแบ่งเลนส์

ทั้งสองกรณีนี้จะทำให้โปรแกรมสามารถหาตำแหน่งของจุดบนเส้นแบ่งเลน ที่โปรแกรมไม่สามารถหาได้จากภาพที่ปรากฏ ซึ่งอาจมีค่าความคลาดเคลื่อนเล็กน้อย



รูปที่ 5.9 แสดงการหาจุดกึ่งกลางของเลนโดยสมบูรณ์ทั้ง 3 จุดของโครงงาน



รูปที่ 5.10 แสดงการหาจุดกึ่งกลางของเลนในกรณีต่าง ๆ

5.2.2.2 กรณีที่มีจุดบนเส้นแบ่งเลนได้ถูกต้อง ดังจุดที่ 2 และ 3 ของรูปที่ 5.10 โดยโปรแกรมจะทำการคำนวณระยะห่างระหว่างจุดทั้งสองจุดไว้ เพื่อจะได้นำมาใช้ต่อไปในกรณีที่ไม่สามารถหาจุดทั้งสองจุดได้ และโปรแกรมจะทำการเปลี่ยนแปลงระยะห่างนี้ เฉพาะในกรณีที่สามารถหาจุดสองจุดได้อย่างถูกต้องเท่านั้น

การนำเอาตำแหน่งและจำนวนของพิกเซลที่ได้จากกระบวนการข้างต้นมาหาค่าเฉลี่ย ดังนั้นโปรแกรมจะสามารถรู้ตำแหน่งโดยประมาณของเส้นแบ่งได้ หลังจากนั้น โปรแกรมสามารถรู้ตำแหน่งของจุดกึ่งกลางของเลนขณะใดขณะหนึ่งได้จากการหาจุดกึ่งกลางระหวางจุดที่อยู่บนเส้นแบ่งเลนทั้งสองได้

โปรแกรมสามารถหาจุดกึ่งกลางของเลน ณ ตำแหน่งใดก็ได้ แต่ในการทดลองนี้ถูกเซ็ตให้หาจุดกึ่งกลางของเลนเพียง 3 ตำแหน่งดังที่ได้แสดงในรูปที่ แล้วนำทั้ง 3 ตำแหน่งที่ได้มาทำการประมวลผลเพื่อควบคุมทิศทางรถต่อไป

5.2.3 อัลกอริทึมการสร้างรหัสในการควบคุมทิศทาง

อัลกอริทึมนี้จะทำการรับค่าทิศทางและความเร็วในการควบคุมทิศทางรถและการเลี้ยวจากอัลกอริทึมส่วนอื่น

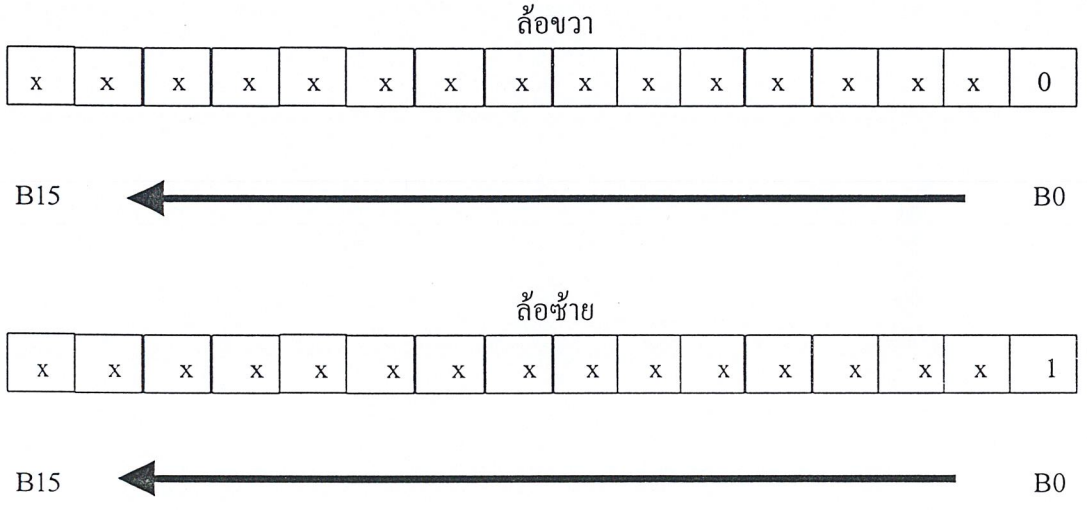
อัลกอริทึมการควบคุมทิศทางรถถูกแบ่งออกเป็น

- เริ่มการทำงาน (Start) จะเป็นการสั่งให้ไมโครคอนโทรลเลอร์อยู่ในสภาวะพร้อมที่จะรับข้อมูลต่างๆที่โปรแกรมจะทำการส่งให้ ซึ่งคำสั่งสตาร์ทนี้จะส่งค่าความเร็วที่ต่ำสุดให้ไมโครคอนโทรลเลอร์ ส่วนไมโครคอนโทรลเลอร์เองก็จะทำการขับเคลื่อนรถไปด้วยความเร็วต่ำสุดเช่นเดียวกัน
- หยุดการทำงาน (Stop) จะสั่งให้รถหยุดการเคลื่อนที่ของรถ
- เพิ่มความเร็ว (Speed) เป็นการสั่งให้ไมโครคอนโทรลเลอร์เพิ่มความเร็วในการหมุนดีซีมอเตอร์ โดยสั่งให้มีความเร็วเพิ่มขึ้นพร้อมกันทั้งสองข้างซึ่งจะเป็นความเร็วของตัวรถ
- ลดความเร็ว (Slow) จะสั่งการให้ไมโครคอนโทรลเลอร์ลดความเร็วของดีซีมอเตอร์ทั้งสองข้างลง โดยจะลดความเร็วพร้อมกันทั้งสองข้าง และเป็นความเร็วของตัวรถ
- เลี้ยวซ้าย (Turn Left) จะสั่งให้รถทำการเลี้ยวซ้ายโดยสั่งให้ไมโครคอนโทรลเลอร์ป้อนความเร็วให้กับดีซีมอเตอร์แต่ละตัวด้วยความเร็วที่ไม่เท่ากัน โดยที่ดีซีมอเตอร์ด้านขวามีความเร็วเพิ่มขึ้น แต่ความเร็วด้านซ้ายมีความเร็วเท่ากับความเร็วของรถ และมีความเร็วมากกว่าน้อยกว่าความเร็วด้านขวา โดยจะทำให้รถเลี้ยวขวาได้

- เลี้ยวขวา (Turn Right)จะสั่งให้รถทำการเลี้ยวขวาได้โดยที่ไมโครคอนโทรลเลอร์จะป้อนความเร็วให้กับดีซีมอเตอร์ด้วยความเร็วที่ไม่เท่ากัน ซึ่งจะป้อนความเร็วให้กับล้อด้านซ้ายเพิ่มขึ้นในขณะที่ความเร็วของล้อขวาจะเท่ากับความเร็วของตัวรถ ทำให้รถเลี้ยวขวาได้

5.2.4 อัลกอริทึมการส่งข้อมูลให้ไมโครคอนโทรลเลอร์

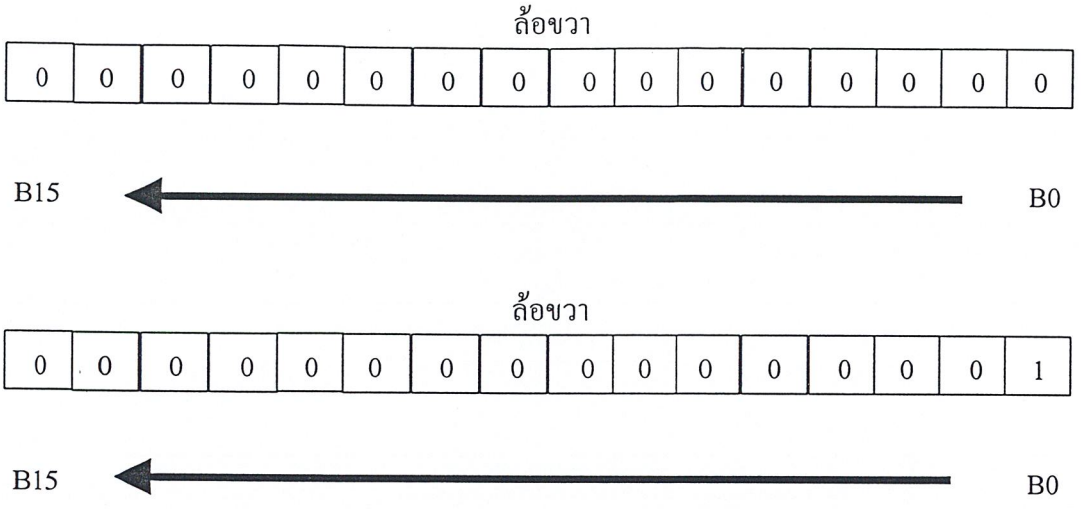
จะเริ่มจากการรับข้อมูลจากอัลกอริทึมการสร้างรหัสในการควบคุมทิศทางของรถ ในการรับข้อมูลนั้น จะรับข้อมูลเข้ามาทีละ 4 ไบต์โดยแบ่งเป็นดีซีมอเตอร์ล้อซ้ายและขวาอย่างละ 2 ไบต์ โดยบิตสุดท้าย (บิต 0) จะเป็นตัวบอกว่าเป็นของล้อซ้ายหรือล้อขวาและอีก 15 บิตที่เหลือจะเป็นข้อมูลความเร็วในการหมุนของมอเตอร์



รูปที่ 5.11 แสดงข้อมูลที่ใช้ในการควบคุมดีซีมอเตอร์ล้อซ้ายและล้อขวา

5.2.5 อัลกอริทึมการทำงานของไมโครคอนโทรลเลอร์

การทำงานของไมโครคอนโทรลเลอร์เริ่มจากการเลือกโหมดการอินเตอรรัพท์ของอินเตอร์รัพท์0โดยจะกำหนดให้ทำงานที่ขอบขาลงของสัญญาณอินเตอร์รัพท์ โดยจะใช้เฉพาะอินเตอร์รัพท์ 0 เท่านั้น จากนั้นก็ทำการเซ็ทค่าให้แก่รีจิสเตอร์ต่างๆ เพื่อให้สามารถใช้ในการทำงานได้ อันได้แก่ แสตคพอยน์เตอร์ รีจิสเตอร์R0,R1และR3 จากนั้นก็ทำการตั้งค่าความเร็วเริ่มต้นให้แก่ล้อทั้งสอง(ความเร็วเริ่มต้นเป็นศูนย์ทั้งสองข้าง) และเซ็ทพอร์ทสำหรับการส่งสัญญาณแฮนด์เช็ค แล้วจึงหยุดเพื่อเฝ้ารอการอินเตอร์รัพท์เพื่อรับค่าความเร็วจากการ์ดพอร์ทต่อไป

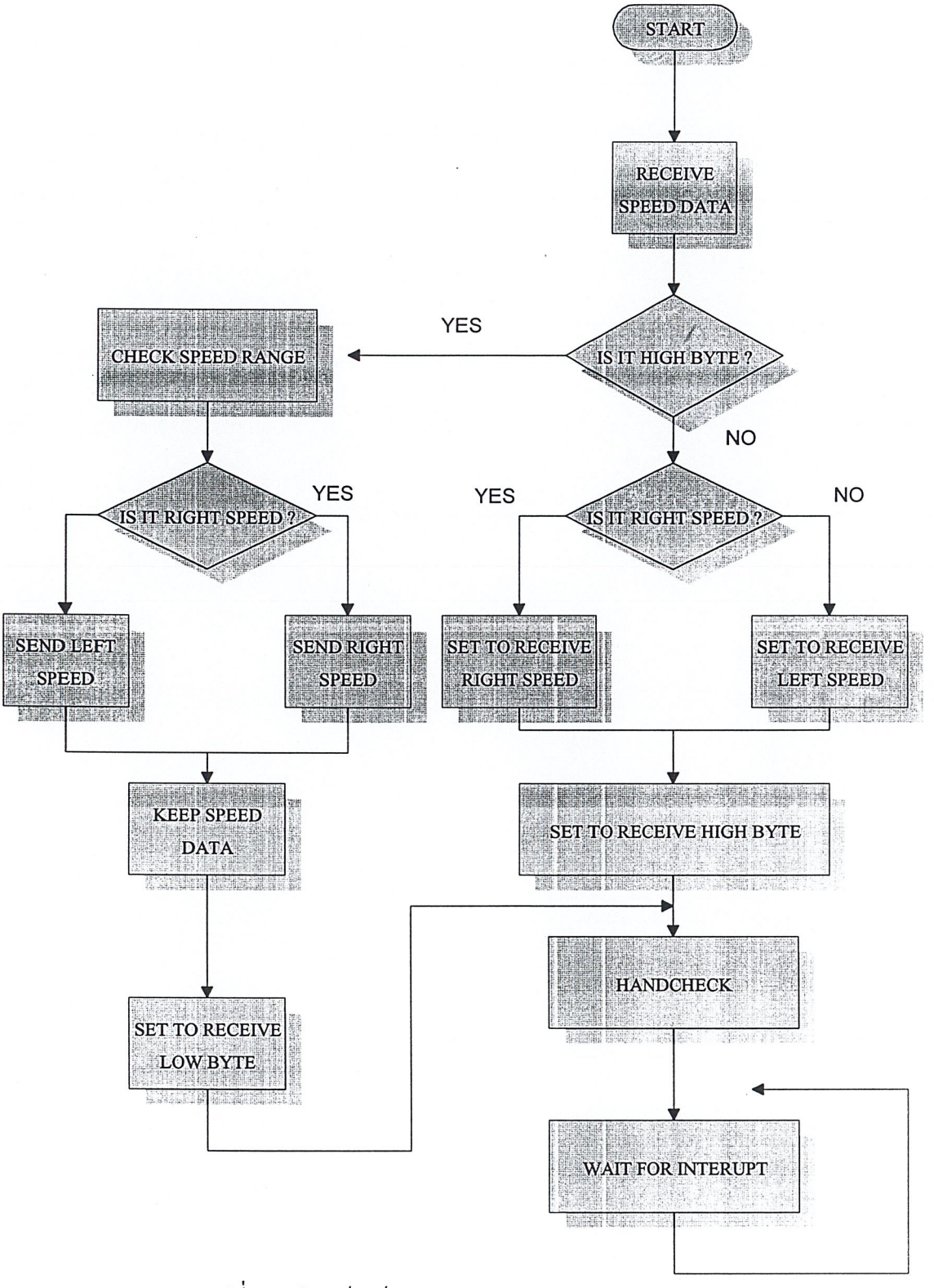


รูปที่ 5.12 แสดงข้อมูลที่ใช้ในการควบคุมมอเตอร์ล้อซ้ายและล้อขวา

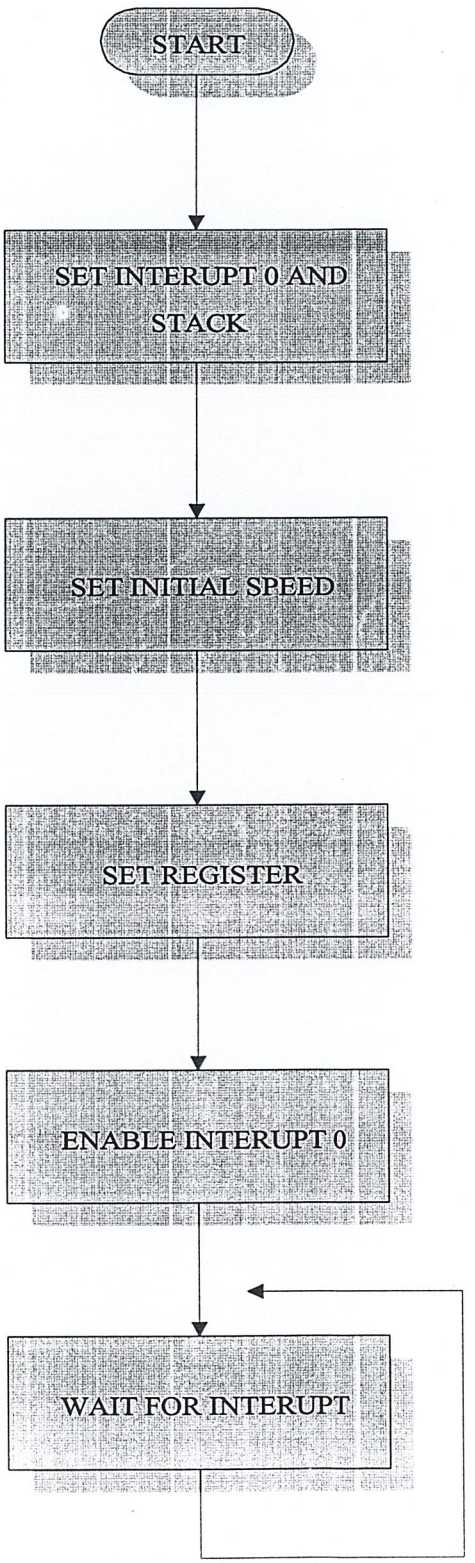
5.2.6 อัลกอริทึมการอินเตอร์รัพท์ภายนอก

เมื่อมีการอินเตอร์รัพท์เข้ามาเพื่อส่งค่าความเร็วแล้ว ไมโครคอนโทรลเลอร์จะทำการดิสเอเบิ้ลอินเตอร์รัพท์ นำข้อมูลเดิมไปเก็บไว้ในแสตค แล้วจึงอ่านค่าเข้ามาจากพอร์ทสอง แล้วจึงทำการเช็คว่าค่าที่ส่งมาเป็นไบท์ต่ำหรือไบท์สูง(เนื่องจากในการส่งค่าความเร็วนั้นจะส่งมาสองไบท์) ถ้าเป็นไบท์ต่ำก็จะทำการเช็คว่าเป็นค่าความเร็วของล้อซ้ายหรือขวา แล้วจึงเซ็ทค่าในรีจิสเตอร์เพื่อให้สามารถรับไบท์สูงของล้อแต่ละข้างได้อย่างถูกต้อง เสร็จแล้วจึงดึงค่าที่เก็บไว้ในแสตคกลับมา แล้วจึงเซ็ทค่าในรีจิสเตอร์เพื่อรับค่าความเร็วไบท์สูง ส่งสัญญาณแฮนด์เช็คไปยังการ์ดพอร์ทเพื่อให้การ์ดพอร์ทส่งค่าไบท์สูงมา ทำการอีนาเบิ้ลอินเตอร์รัพท์ แล้วจึงหยุดเพื่อเฝ้ารอการอินเตอร์รัพท์เพื่อรับค่าความเร็วจากการ์ดพอร์ทต่อไป

แต่หากค่าที่ส่งมาเป็นไบท์สูง ไมโครคอนโทรลเลอร์ก็จะทำการเช็คว่าเป็นค่าของล้อซ้ายหรือขวาแล้วจึงทำการส่งค่าความเร็วผ่านทางพอร์ท1 ไปยังวงจรควบคุมความเร็วต่อไป แล้วจึงเซ็ทค่าในรีจิสเตอร์เพื่อรับค่าไบท์ต่ำในครั้งต่อไป แสดงให้เห็นดังในรูปที่ 5.13 และ5.14



รูปที่ 5.13 โฟลว์ชาร์ตแสดงการควบคุมความเร็วของรถ

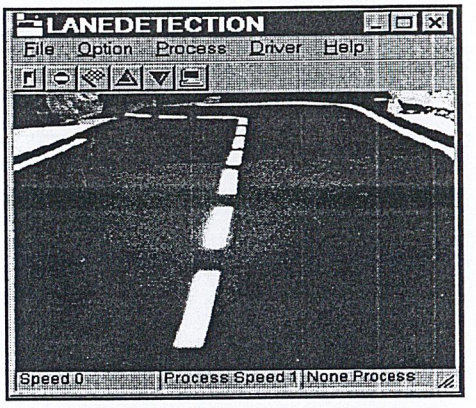


รูปที่ 5.14 แสดงการเซ็ตการอินเตอร์รัปท์

บทที่ 6

การทดลองและผลการทดลอง

ในขั้นแรกเมื่อเราเข้าสู่การทำงานของโปรแกรม โดยเมื่อทำการเปิดโปรแกรมจะได้ดังรูปที่ 6.1



รูปที่ 6.1 แสดงหน้าจอเมื่อทำการเปิดโปรแกรม

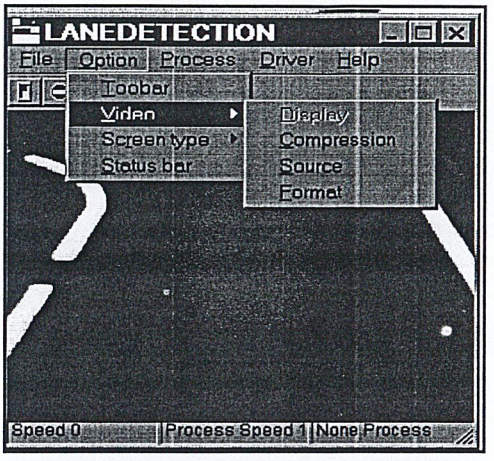
6.1 รายละเอียดต่างๆ ของโปรแกรม

6.1.1 เมนูอ๊อปชั่น (Option Menu) จะแบ่งเป็น

1) **ทูลบาร์ (Tool Bar)** เป็นส่วนที่ใช้แสดงแถบของเครื่องมือที่ใช้ในการทำงาน

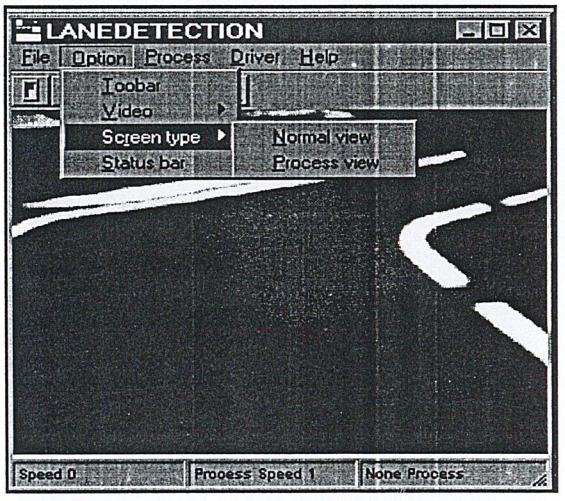
2) **วีดีโอ (Video)** เป็นส่วนที่ทำหน้าที่รับภาพจากกล้องดิจิตอล โดยจะทำการกำหนดค่าพารามิเตอร์ต่างๆ ได้แก่

- **ค่าในการบีบอัด (Compression)** มีหน้าที่ในการบีบอัดข้อมูลที่ได้จากกล้องดิจิตอลเพื่อใช้ในการแสดงผล
- **ซอร์ส (Source)** ทำหน้าที่ในการปรับค่าพารามิเตอร์ต่างๆ ของภาพ เช่น ความสว่าง(Brightness) ,แซททูเรชั่น (Saturation) และระดับความเข้ม (Contrast) เป็นต้น
- **ฟอร์แมท (Format)** ใช้ในการกำหนดขนาดของภาพที่แสดง ความทึบของสีที่แสดง และความคมชัดของภาพ



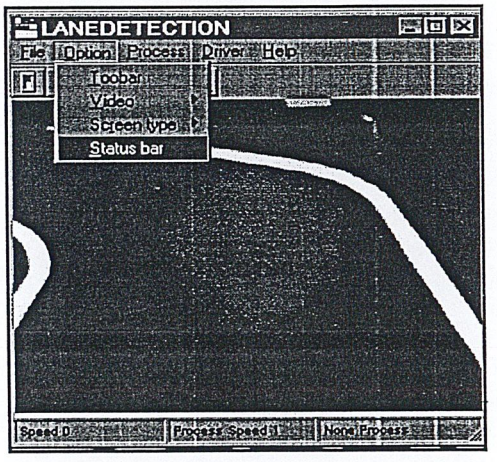
รูปที่ 6.2 แสดง อ๊อปชั่นเมนูส่วนวีดีโอ

3. **สกรีนไทป์ (Screen Type)** ใช้กำหนดรูปแบบการแสดงผลของโปรแกรม โดยภาพที่แสดงบนจอภาพมีการประมวลผลได้เป็น 2 รูปแบบคือ
   - **ภาพจริง (Normal View)** จะแสดงภาพที่ได้จากกล้องวีดิโอในขณะนั้นจริงๆ ไม่ผ่านกระบวนการใดๆ
   - **ภาพจากการประมวลผล (Process view)** ภาพที่ได้จะผ่านการประมวลผล



รูปที่ 6.3 แสดง อ๊อปชั่นเมนู ส่วนสกรีนไทป์

4. **แถบแสดงสถานะ (Status Bar)** โดยจะใช้กำหนดว่าจะให้แสดงแถบสถานะของโปรแกรมในขณะนั้นหรือไม่ ซึ่งจะบอกถึง
    - ค่าความเร็วของรถ (Vehicle Speed)
    - ความเร็วในการประมวลผลภาพ (Process Speed)
    - สถานะการทำงานของโปรแกรม (Program Status)



รูปที่ 6.4 แสดงอ๊อปชั่นเมนูส่วนแถบสถานะ

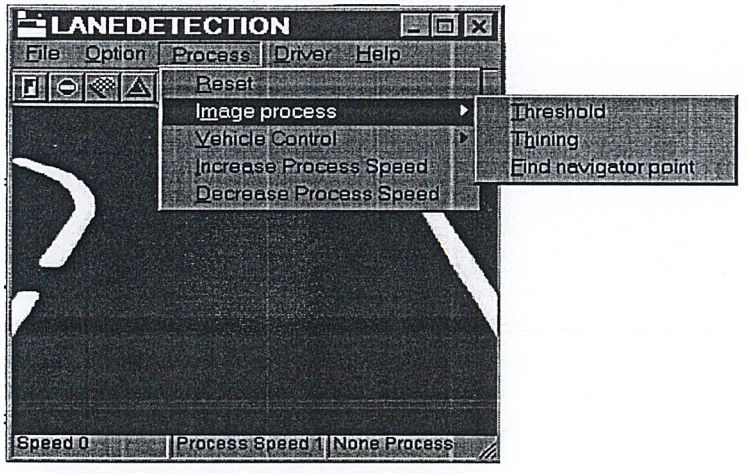
6.1.2 เมนูแสดงการประมวลผลภาพ (Process Menu)

มีชุดคำสั่งดังนี้

1. **รีเซ็ต (Reset)** ใช้ในการรีเซ็ตโปรแกรมหรือเริ่มต้นการทำงานของโปรแกรมใหม่ทั้งหมด
2. **อิมเมจโปรเซส (Image Process)** ใช้ในการประมวลผลภาพในรูปแบบต่างๆ ซึ่งมีการทำงานของฟังก์ชั่นต่างๆ ดังต่อไปนี้

- **เทรชโชลด์(Threshold)** เมื่อใช้ฟังก์ชั่นนี้จำทำให้ภาพที่ได้มีพิกเซล (pixel) เป็นภาพขาว - ดำ ทั้งนี้การที่จะเป็นเช่นนี้ได้ขึ้น กับว่าค่าที่ได้จากการเปรียบเทียบ (Threshold Compare) ที่เราเซ็ตไว้ในตัวโปรแกรมเป็นเช่นไร

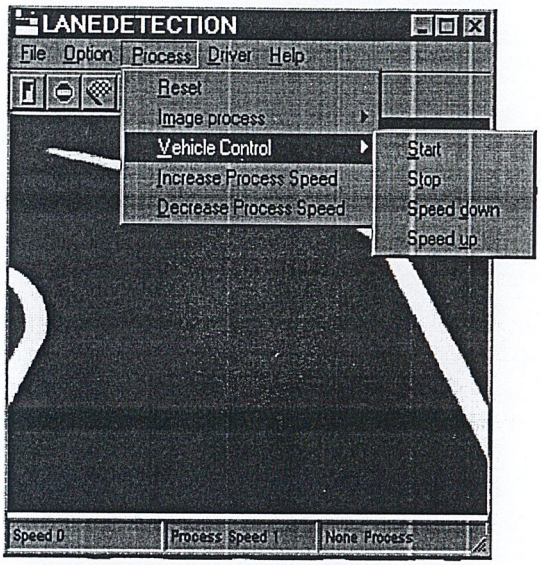
- **ธินนิ่ง (Thinning)** จะทำให้ภาพที่ได้จากการประมวลผลเป็นภาพของขอบเลนถนน แต่ทั้งนี้ภาพที่ได้จะไม่ได้ทำการประมวลผลทั้งหมด แต่จะทำเพียงบางส่วนเท่านั้น
- **การหาจุดกึ่งกลางของเลนถนน (Find Navigator Point)** ภาพจากกระบวนการนี้ จะเป็นภาพที่ได้จากการธินนิ่งด้วยและจะทำการหาจุดกึ่งกลางของเลนด้วย ซึ่งจะได้มา 3 จุด โดยคอมพิวเตอร์จะนำทั้ง 3 จุดที่ได้ในการตรวจจับเลนของถนน



รูปที่ 6.5 แสดงเมนูโปรเซส

3. **ส่วนควบคุมตัวรถ (Vehicle control)**

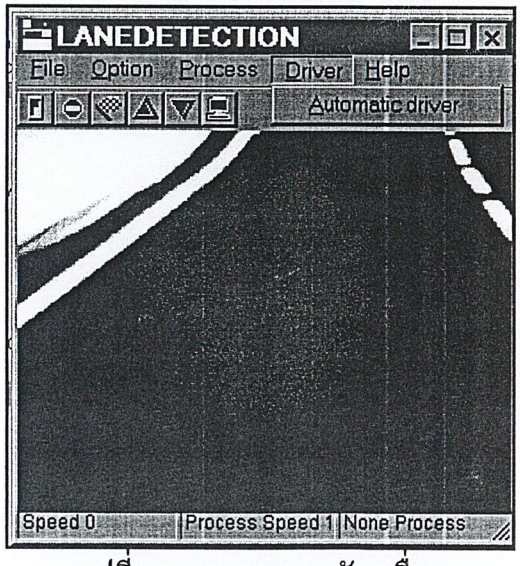
- เริ่มต้นการทำงาน(Start)โดยเริ่มจากเมื่อรถหยุดนิ่งอยู่
- หยุดการเคลื่อนที่ (Stop)เมื่อรถขับเคลื่อนอยู่
- เพิ่มความเร็ว (Speed Up)
- ลดความเร็ว (Speed Down)



รูปที่ 6.6 แสดงฟังก์ชั่นควบคุมการขับเคลื่อน

4. **ส่วนเพิ่มความเร็วในการประมวลผลภาพ (Increase Process Speed)**
5. **ส่วนลดความเร็วในการประมวลผลภาพ (Decrease Process Speed)**

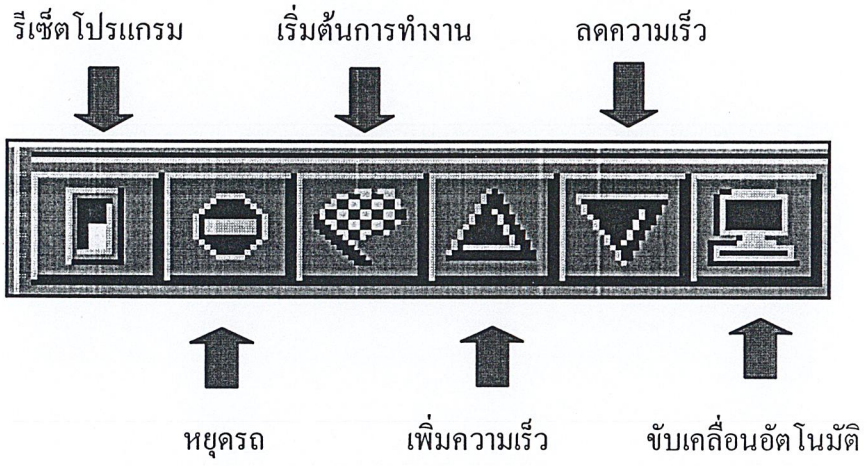
6.1.3เมนูการขับเคลื่อน (Driver Menu) ประกอบด้วย



รูปที่ 6.7 แสดงเมนูการขับเคลื่อน

- **ส่วนการขับเคลื่อนรถโดยอัตโนมัติ (Automatic Driver)** จะเป็นฟังก์ชั่นการทำงานโดยให้รถขับเคลื่อนโดยคำสั่งของคอมพิวเตอร์

6.1.4 แถบเครื่องมือ (Tool Bar) จะแสดงได้ดังรูป 6.8

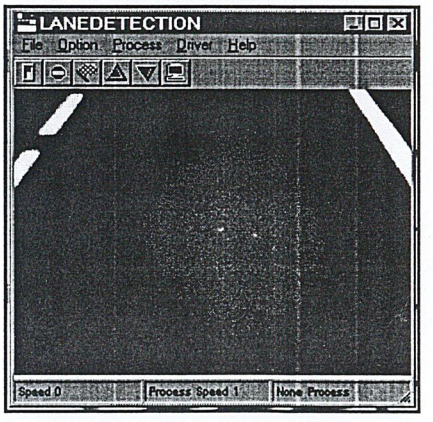


รูปที่ 6.8 แสดงแถบสถานะ (Tool Bar)

6.2 การใช้งานโปรแกรมส่วนการขับเคลื่อนรถ (Lane Detection Program)

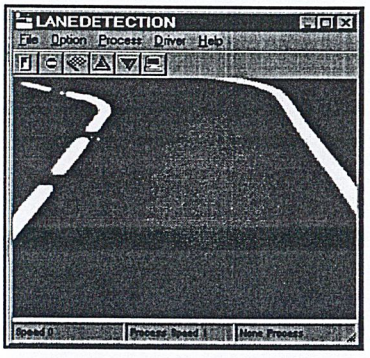
6.2.1 การตรวจจับเลนโดยคอมพิวเตอร์

เมื่อเริ่มต้นการทำงานของโปรแกรมจะปรากฏส่วนการทำงานของโปรแกรมดังรูป 6.9



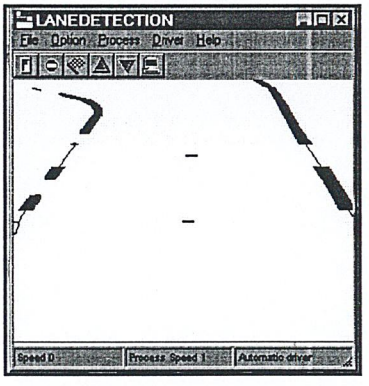
รูปที่ 6.9 แสดงผลเมื่อเริ่มโปรแกรม

ภาพที่รับเข้ามาจะเป็นภาพที่ปรากฏจริงๆบนถนนโดยหากทำการเลือกที่เมนูออปชัน (Option Menu)จะปรากฏภาพ 2 ส่วนคือ ส่วนภาพจริง(normal view) และส่วนของภาพจากการประมวลผล(process view) ดังรูป 6.10



ก.)

ข.)



ค.)

รูปที่ 6.10 แสดงผลการทำงานส่วน สกรีนไทป์

ก.) การแสดงผลแบบนอร์มัลวิว

ข.) การแสดงผลแบบโปรเซสวิว

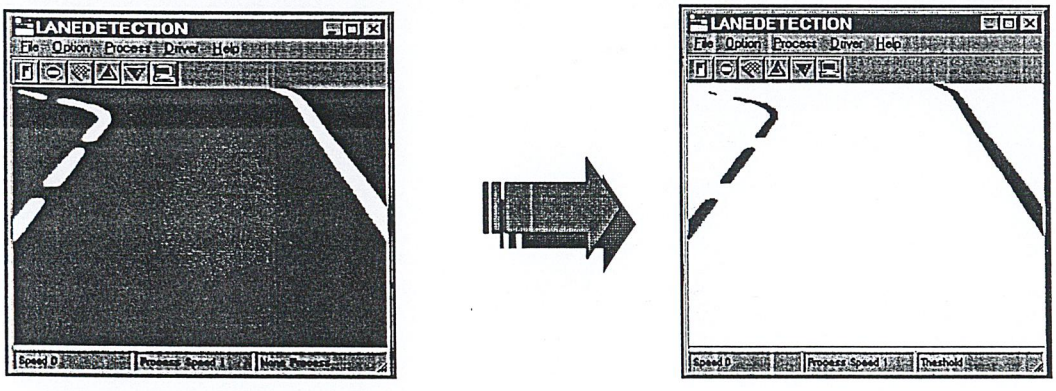
ค.) การแสดงผลเมื่อรถขับเคลื่อนอัตโนมัติ

จากนั้นทำการขับเคลื่อนรถโดยก่อนที่รถจะเคลื่อนที่ให้เราเลือกที่ เมนูการขับเคลื่อน (Driver Menu) แล้วเลือก Automatic driver ซึ่งโปรแกรมจะทำการตรวจจับเลนถนน แล้วเข้าไปที่ Process Menu เพื่อที่จะทำการประมวลผลภาพ แล้วเลือกที่ Vehicle Menu เพื่อเลือกที่ฟังก์ชั่น Start เพื่อเริ่มต้นการทำงานให้กับรถหรืออาจเลือกคำสั่งเริ่มการทำงานได้ที่แถบเครื่องมือเลยก็ได้ เราสามารถเพิ่มหรือลดความเร็วของรถได้โดยใช้คำสั่ง Speed Up หรือ Speed Down ได้จากแถบเครื่องมือหรือจาก Process Menu ก็ได้ โดยที่ในขณะที่รถเคลื่อนที่เราสามารถที่จะให้ภาพแสดงภาพจริงหรือภาพจากการประมวลผลก็ได้โดยใช้คำสั่ง normal view หรือ process view ได้จากเมนู Screen Type

6.2.2 การประมวลผลภาพ

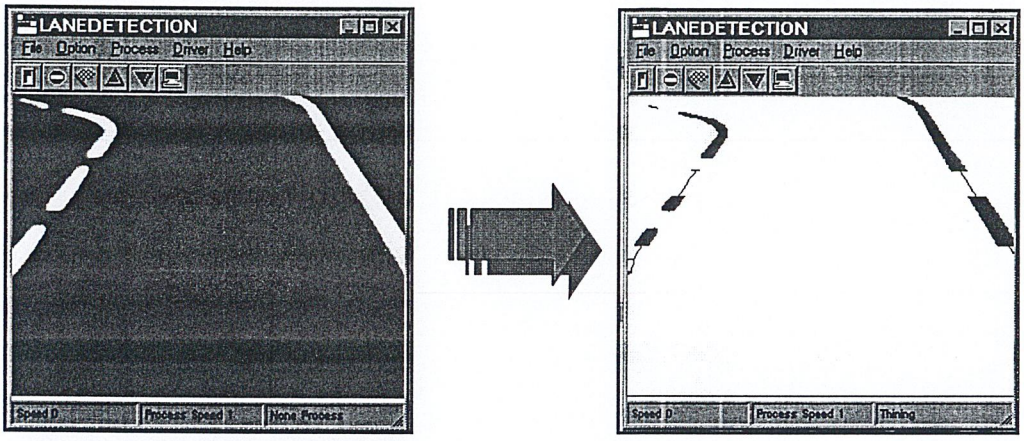
ภาพที่ได้จากการประมวลผลภายหลังจากที่ได้รับจากกล้องดิจิตอลเรียบร้อยแล้วนั้น จะนำผลลัพธ์ที่ได้แสดงออกทางส่วนของจอภาพ โดยเราจะทำกระบวนการนี้ได้ก็ต่อเมื่อรถไม่ขับเคลื่อนแล้วเท่านั้น กระบวนการต่างๆมีดังนี้

- **เทรชโชลด์(Threshold)** ภาพที่ได้จากฟังก์ชั่นนี้เป็นภาพที่ได้เพียง 2 ระดับขั้นเท่านั้น คือ เป็นพิกเซล (pixel)ที่แสดงระดับสีขาวกับดำ ดังรูป 6.11



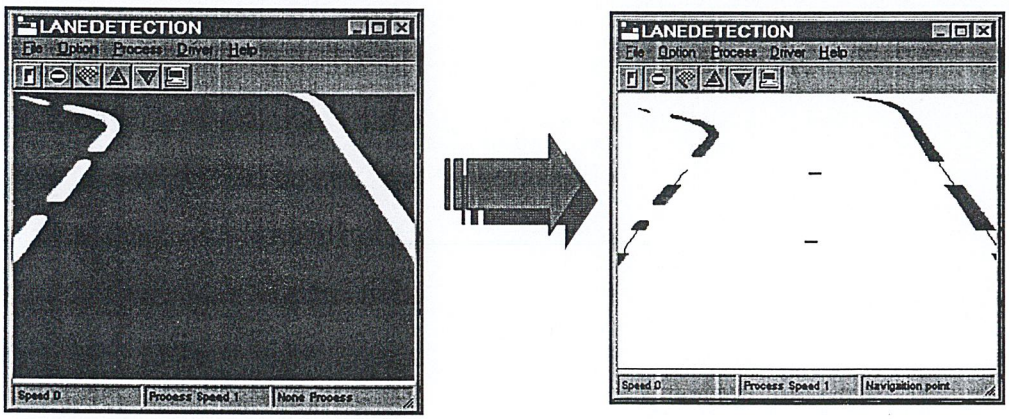
รูปที่ 6.11 แสดงฟังก์ชั่นเทรชโชลด์

- **ธินนิ่ง (Thinning)** ภาพที่ได้รับจากกล้องดิจิตอลจะถูกทำให้เหลือเพียงส่วนของขอบถนนเท่านั้นโดยที่ส่วนภาพที่แสดงนั้นยังเป็นส่วนที่ผ่านการเทรสโฮลด้วยแต่ถูกทำให้เหลือเพียงส่วนของขอบถนนเท่านั้น



รูปที่ 6.12 แสดงฟังก์ชั่นธินนิ่ง

- การหาจุดกึ่งกลางเลน (Find Navigation Point) เมื่อผ่านคำสั่งนี้ เราจะได้จุดกึ่งกลางของเลนถนนมา 3 จุด โดยจากทั้ง 3 จุดนี้จะถูกคอมพิวเตอร์นำไปใช้ในการประมวลผลภาพเพื่อบังคับรถให้แล่นไปได้ถูกทิศทางได้ถูกต้องดังรูป 6.13



รูปที่ 6.13 แสดงฟังก์ชั่นการหาจุดกึ่งกลางเลน

บทที่7
บทสรุป

จากโครงงานนนี้จะเห็นว่าประกอบด้วยส่วนหลักๆ สองส่วน คือ ส่วนของฮาร์ดแวร์ และส่วนของซอฟท์แวร์ ซึ่งส่วนของฮาร์ดแวร์เป็นส่วนของตัวรถ ส่วนซอฟท์แวร์จะเป็นส่วนของการประมวลผลภาพเลนถนน โดยทั้งสองส่วนจะทำงานร่วมกันเพื่อให้รถเคลื่อนไปตามเลนถนน

ในการขับเคลื่อนรถนั้นจะอาศัยการประสานงานกันระหว่างกล้องดิจิตอล วงจรไดรฟ์มอเตอร์กระแสตรง และไมโครคอนโทรลเลอร์ โดยจะแสดงผลทางจอคอมพิวเตอร์ พร้อมทั้งควบคุมการเคลื่อนที่ของรถ

เมื่อเราสั่งให้รถเคลื่อนที่ กล้องดิจิตอลจะรับภาพเลนถนนแล้วประมวลผลโดยผ่าน3 กระบวนการ ได้แก่ เทรชโชลด์ ธินนิ่ง และหาจุดกึ่งกลางเลน โดยการเทรชโชลด์จะแปลงจากภาพปกติ (normal view) เป็นภาพที่ประกอบด้วยพิกเซลสีขาวและดำ แล้วจึงหาขอบถนนด้วยการธินนิ่ง เสร็จแล้วจึงหาจุดกึ่งกลางเลน (find navigator) จากจุดกึ่งกลางเลนที่ได้จึงนำมาประมวลผลเพื่อควบคุมทิศทางการเคลื่อนที่ของรถต่อไป

ในส่วนวงจรควบคุมการเคลื่อนที่ของรถนั้น ในโครงงานนี้เราใช้มอเตอร์กระแสตรงซึ่งสามารถควบคุมความเร็วและตำแหน่งการเคลื่อนที่ได้ยาก จึงเป็นการยากที่จะควบคุมให้รถแล่นไปตามเลนได้อย่างถูกต้อง ดังจะสังเกตได้ว่าการเคลื่อนที่ของรถจะมีลักษณะซิกแซก ทั้งยังมีปัญหาในส่วนของกล้องที่ใช้จับภาพ กล่าวคือ กล้องที่ใช้นั้นสามารถจับภาพได้มุมแคบ ภาพเลนที่จับได้จึงมีระยะห่างจากตัวรถมาก ทำให้การเคลื่อนที่ของรถเบี่ยงเบนไปจากเลนมาก ทั้งกล้องยังมีความไวต่ำซึ่งเมื่อรวมกับความเร็วในการประมวลผลของคอมพิวเตอร์ซึ่งค่อนข้างต่ำแล้ว จะพบว่าบ่อยครั้งที่ไม่สามารถประมวลผลได้เร็วพอ ทำให้เกิดความผิดพลาดขึ้น

ดังนั้นแนวทางในการพัฒนาและปรับปรุงก็คือ ใช้กล้องดิจิตอลและอุปกรณ์ในการประมวลผลที่มีความเร็วและประสิทธิภาพสูง ซึ่งในขั้นนี้นั้นสามารถนำไปใช้ได้ในการทำงานที่มีสภาพแวดล้อมซึ่งไม่มีสิ่งรบกวนมาก เช่น ในโรงงานที่มีการจัดสภาพแวดล้อมให้เหมาะกับการประมวลผล เป็นต้น

ภาคผนวก

โปรแกรมภาษาแอสแซมบลี

```
        ORG     0000H
        LJMP    MAIN

        ORG     0030H
MAIN:   SETB    TCON.0
        SETB    IP.0
        MOV     SP,#10H
        MOV     A,#0EEH
        MOV     P1,A
        CPL     A
        MOV     R3,A
        MOV     R0,#00H
        MOV     R1,#00H
        SETB    P3.3
        MOV     IE,#81H
        SJMP    $

        ORG     0003H
        LJMP    INT_0

        ORG     0050H
INT_0:  MOV     A,P2
        CPL     A
        CJNE    R0,#00H,HI
        RRC     A
        JNC     LEFT
        MOV     R1,#0FFH
        JMP     HAND
LEFT:   MOV     R1,#00H
```

```
        JMP     HAND
HAND:   MOV     R0,#0FFH
        LJMP    WHAT
HI:     CJNE    A,#0FEH,NSTOP
        INC     A
        MOV     P1,A
        CPL     A
        MOV     R3,A
        LJMP    OK
NSTOP:  RLC     A
        JNC     LOWS
        RLC     A
        JNC     HIS1
        RLC     A
        RLC     A
        JNC     HIS2
FIFTEEN: MOV    R6,#0FH
        LJMP    MOVE
HIS2:   RLC     A
        JC      FIFTEEN
        MOV     R6,#0EH
        LJMP    MOVE
HIS1:   RLC     A
        JNC     HIS4
        RLC     A
        JNC     HIS3
        MOV     R6,#0DH
        LJMP    MOVE
HIS3:   MOV     R6,#0CH
        LJMP    MOVE
HIS4:   RLC     A
        JNC     HIS5
        MOV     R6,#0BH
        LJMP    MOVE
```

```
HIS5:   MOV   R6,#0AH
        LJMP  MOVE
LOWS:   RLC   A
        JNC   LOW1
        RLC   A
        JNC   LOW2
        RLC   A
        JNC   SEVEN
        RLC   A
        JNC   LOW4
NINE:   MOV   R6,#09H
        LJMP  MOVE
LOW4:   RLC   A
        JC    NINE
        MOV   R6,#08H
        LJMP  MOVE
SEVEN:  MOV   R6,#07H
        LJMP  MOVE
LOW2:   RLC   A
        JNC   FIVE
        MOV   R6,#06H
        LJMP  MOVE
FIVE:   MOV   R6,#05H
        LJMP  MOVE
LOW1:   RLC   A
        JNC   LOW3
        RLC   A
        JNC   THREE
        MOV   R6,#04H
        LJMP  MOVE
THREE:  MOV   R6,#03H
        LJMP  MOVE
LOW3:   RLC   A
        JNC   ONE
```

```
        MOV     R6,#02H
        LJMP    MOVE
ONE:    MOV     R6,#01H
MOVE:   CJNE    R1,#00H,RH
        MOV     A,R6
        SWAP    A
        MOV     R6,A
        MOV     A,R3
        ANL     A,#0FH
        JMP     OH
RH:     MOV     A,R3
        ANL     A,#0F0H
OH:     ORL     A,R6
        MOV     R3,A
        CPL     A
        MOV     P1,A
OK:     MOV     R0,#00H
WHAT:   CLR     P3.3
        SETB    P3.3
        RETI
END
```

โปรแกรมควบคุมการทำงานของเครื่อง (Visual C++)

```
//Lane.c
#include <windows.h>
#include <windowsx.h>
#include <commctrl.h>
#include <stdio.h>
#include <vfw.h>
#include "globals.h"
#include "resource.h"

#define NUM_IMAGES          7 //number of images on tool bar
#define IMAGE_WIDTH              16//image width in pixel
#define IMAGE_HEIGHT        16//image height in pixel
#define BUTTON_WIDTH        0//button width in pixel (use default)
#define BUTTON_HEIGHT       0//button height in pixel (use default)

HWND hToolBar;
HWND hStatusBar;
HMENU hMenu;
BOOL fToolbar;
UINT ButtonID;
char cBuff[20];

TBBUTTON tbb[] =
{
        {0,ID_PROCESS_RESET,      TBSTATE_ENABLED,TBSTYLE_BUTTON,0,0},
        {1,ID_PROCESS_STOP,       TBSTATE_ENABLED,TBSTYLE_BUTTON,0,0},
        {2,ID_PROCESS_START,      TBSTATE_ENABLED,TBSTYLE_BUTTON,0,0},
        {3,ID_PROCESS_SPEEDUP,    TBSTATE_ENABLED,TBSTYLE_BUTTON,0,0},
```

```
        {4,ID_PROCESS_SPEEDDOWN,
        TBSTATE_ENABLED,TBSTYLE_BUTTON,0,0},
        {5,ID_DRIVER_AUTOMATICDRIVER,
        TBSTATE_ENABLED,TBSTYLE_BUTTON,0,0}
};

LRESULT CALLBACK WndProc(HWND hWnd,
                                        UINT uMessage,
                                        WPARAM wParam,
                                        LPARAM lParam)
{

        switch (uMessage)
        {
                case WM_CREATE:
                        //load common controle DLL
                        InitCommonControls();

                        //create a tool bar
                        CreateToolBar(hWnd);
                        hStatusBar=CreateStatusBar(hWnd);
                        SendMessage(hStatusBar,SB_SETTEXT,
                                        (WPARAM)0|0,(LPARAM)(LPSTR)"Speed 0");
                        SendMessage(hStatusBar,SB_SETTEXT,
                                        (WPARAM)1|0,(LPARAM)(LPSTR)"Process Speed 1");
                        SendMessage(hStatusBar,SB_SETTEXT,
                                        (WPARAM)2|0,(LPARAM)(LPSTR)"None Process");
                        CreateVideo(hWnd);
                        break;
```

```
/*      case WM_CHAR:
        {
                chCharCode = (TCHAR) wParam;    // character code
                switch(chCharCode)
                {
                        //DownArrow
                        case 53:Direction(_slow,20);
                                break;
                        //UpArrow
                        case 56:Direction(_fast,20);
                                break;
                        //Right Arrow
                        case 54:Direction(_right,10);
                                  Direction(_left,-10);
                                break;
                                //Left Arrow
                        case 52:Direction(_left,10);
                                        Direction(_right,-10);
                                break;
                }


        }

        break;*/
        case WM_COMMAND:
                switch (GET_WM_COMMAND_ID(wParam,lParam))
                {
                        case ID_FILE_EXIT:
```

```
                    if (DialogBox(hInst,MAKEINTRESOURCE
(EXITDLG),
                            hWnd, (DLGPROC)ExitProgram))
                    {
                            DestroyWindow(hWndV1);
                            DestroyWindow(hWnd);
                    }
                    break;

            case IDM_TOOLBAR:
                    hMenu=GetMenu(hWnd);
                    if(!fToolbar)
                    {
                            //show tool bar
                            fToolbar=true;
                            ShowWindow(hToolBar,SW_SHOW);
                            CheckMenuItem(hMenu,IDM_TOOLBAR,
MF_CHECKED);
                    }
                    else
                    {
                            //hide tool bar
                            fToolbar=false;
                            ShowWindow(hToolBar,SW_HIDE);
                            CheckMenuItem(hMenu,IDM_TOOLBAR,
MF_UNCHECKED);
                    }
                    break;

            case ID_OPTION_VIDEO_DISPLAY:
```

```
                    capDlgVideoDisplay(hWndV1);

//capSetCallbackOnFrame(hWndV1,VideoFrameCallbackFirst);
                    break;

                case ID_OPTION_VIDEO_COMPRESSION:
                    capDlgVideoCompression(hWndV1);

//capSetCallbackOnFrame(hWndV1,VideoFrameCallbackFirst);
                    break;

                case ID_OPTION_VIDEO_SOURCE:
                    capDlgVideoSource(hWndV1);

capSetCallbackOnFrame(hWndV1,VideoFrameCallbackFirst);
                    break;

                case ID_OPTION_VIDEO_FORMAT:
                    capDlgVideoFormat(hWndV1);
                    ResizeCaptureWindow(hWndV1);

capSetCallbackOnFrame(hWndV1,VideoFrameCallbackFirst);
                    break;
                case ID_OPTION_SCREENTYPE_PROCESSVIEW:

    EnableMenuItem(GetMenu(hWnd),ID_OPTION_SCREENTYPE_NORMALVIEW,
MF_ENABLED);

    EnableMenuItem(GetMenu(hWnd),ID_OPTION_SCREENTYPE_PROCESSVIEW,
MF_GRAYED);
```

```
                    if(mProcess==1)capSetCallbackOnFrame
(hWndV1,VideoFrameCallbackLaneDetection1);
                    else if(mProcess==2)capSetCallbackOnFrame
(hWndV1,VideoFrameCallbackLaneDetection2);
                    mProcess=2;
                    fReal=false;
                    break;
                case ID_OPTION_SCREENTYPE_NORMALVIEW:

    EnableMenuItem(GetMenu(hWnd),ID_OPTION_SCREENTYPE_NORMALVIEW,
MF_GRAYED);

    EnableMenuItem(GetMenu(hWnd),ID_OPTION_SCREENTYPE_PROCESSVIEW,
MF_ENABLED);
                    if(mProcess==1)capSetCallbackOnFrame
(hWndV1,VideoFrameCallbackLaneDetection1);
                    else if(mProcess==2)capSetCallbackOnFrame
(hWndV1,VideoFrameCallbackLaneDetection2);
                    mProcess=1;
                    fReal=true;
                    break;
                case ID_PROCESS_THRESHOLE:

    capSetCallbackOnFrame(hWndV1,VideoFrameCallbackThreshole);
                    mProcess=0;
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)2|0,(LPARAM)(LPSTR)"Theshold");
                    break;

                case ID_PROCESS_RESET:
```

```
capSetCallbackOnFrame(hWndV1,VideoFrameCallbackNOP);
                    mProcess=0;
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)2|0,(LPARAM)(LPSTR)"None Process");
                    break;
                case ID_PROCESS_IMAGEPROCESS_THINING:

capSetCallbackOnFrame(hWndV1,VideoFrameCallbackThining);
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)2|0,(LPARAM)(LPSTR)"Thining");
                    break;
                case
ID_PROCESS_IMAGEPROCESS_FINDNAVIGATORPOINT:

capSetCallbackOnFrame(hWndV1,VideoFrameCallbackFNP);
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)2|0,(LPARAM)(LPSTR)"Navigaition point");
                    break;

                case ID_PROCESS_INCREASEPROCESSSPEED:
                        if(mProcessSpeed<10) mProcessSpeed++;
                        sprintf(cBuff,"Process Speed
%d",mProcessSpeed);
                        SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)1|0,(LPARAM)(LPSTR)cBuff);
                    break;

                case ID_PROCESS_DECREASEPROCESSSPEED:
                        if(mProcessSpeed>1) mProcessSpeed--;
```

```
                    sprintf(cBuff,"Process Speed
%d",mProcessSpeed);
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)1|0,(LPARAM)(LPSTR)cBuff);
                    break;

            case ID_PROCESS_START:
                    Direction(_start,0);
                    break;

            case ID_PROCESS_STOP:
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)0|0,(LPARAM)(LPSTR)"Speed 0");
                    Direction(_stop,0);
                    break;

            case ID_PROCESS_SPEEDUP:
                    Direction(_fast,50);
                    sprintf(cBuff,"Speed %d",frequency);
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)0|0,(LPARAM)(LPSTR)cBuff);
                    break;

            case ID_PROCESS_SPEEDDOWN:
                    Direction(_slow,50);
                    sprintf(cBuff,"Speed %d",frequency);
                    SendMessage(hStatusBar,SB_SETTEXT,
                (WPARAM)0|0,(LPARAM)(LPSTR)cBuff);
                    break;
```

```
                    case ID_DRIVER_AUTOMATICDRIVER:
                        SendMessage(hStatusBar,SB_SETTEXT,
                    (WPARAM)2|0,(LPARAM)(LPSTR)"Automatic driver");
                        if(fReal)
                        {

capSetCallbackOnFrame(hWndV1,VideoFrameCallbackLaneDetection2);
                                mProcess=1;

                        }
                        else
                        {

capSetCallbackOnFrame(hWndV1,VideoFrameCallbackLaneDetection1);
                                mProcess=2;
                        }
                        break;
                    case ID_DRIVER_MANUALDRIVER:

capSetCallbackOnFrame(hWndV1,VideoFrameCallbackNOP);

                        break;

                    case IDM_ABOUT:
                        //creat an about dialog box
                        DialogBox(hInst,MAKEINTRESOURCE
(ABOUTDLG),
                            hWnd, (DLGPROC)About);
                        break;
```

```
                default:
                        return DefWindowProc (hWnd,uMessage,
wParam,lParam);
                }
                break;

        case WM_NOTIFY:
                switch(((LPNMHDR)lParam)-> code)
                {
                        //process tool tip control message
                        case TTN_NEEDTEXT:
                                {
                                LPTOOLTIPTEXT lpttt;

                                lpttt = (LPTOOLTIPTEXT)lParam;
                                lpttt -> hinst = hInst;

                                ButtonID = lpttt -> hdr.idFrom;
                                switch(ButtonID)
                                {
                                        case ID_PROCESS_RESET:
                                                lpttt->lpszText =
                                                        MAKEINTRESOURCE
(IDS_TIPS1);
                                                break;

                                        case ID_PROCESS_STOP:
                                                lpttt->lpszText =
```

```
                                                    MAKEINTRESOURCE
(IDS_TIPS2);
                                                    break;
                                                case ID_PROCESS_START:
                                                    lpttt->lpszText =
                                                        MAKEINTRESOURCE
(IDS_TIPS3);
                                                    break;
                                                case ID_PROCESS_SPEEDUP:
                                                    lpttt->lpszText =
                                                        MAKEINTRESOURCE
(IDS_TIPS4);
                                                    break;
                                                case ID_PROCESS_SPEEDDOWN:
                                                    lpttt->lpszText =
                                                        MAKEINTRESOURCE
(IDS_TIPS5);
                                                    break;
                                                case ID_DRIVER_AUTOMATICDRIVER:
                                                    lpttt->lpszText =
                                                        MAKEINTRESOURCE
(IDS_TIPS6);
                                                    break;
                                            }
                                            break;
                                            }
                                        default:break;
                                    }
                                    break;
```

```
                case WM_DESTROY:
                        free(bmi);
                        PostQuitMessage(0);
                        break;

                default:
                        return DefWindowProc(hWnd,uMessage,
                                                        wParam,lParam);
        }
        return 0;
}

//function for creating a tool bar
BOOL CreateToolBar (HWND hWnd)
{
        hToolBar = CreateToolbarEx(
                                hWnd,
                                WS_CHILD | WS_VISIBLE |
                                TBSTYLE_TOOLTIPS,
                                IDR_TOOLBAR1,
                                NUM_IMAGES,
                                hInst,
                                IDR_TOOLBAR1,  //toolbar image bitmap file
                                tbb,
                                sizeof (tbb)/sizeof (TBBUTTON),
                                16,
                                16,
                                0,
                                0,
                                sizeof(TBBUTTON));
```

```
        return (hToolBar != NULL);
}

//Function for create status bar
HWND CreateStatusBar(HWND hWnd)
{
        HWND hStatusBar;
        int textWidth[]={100,200,320};

        hStatusBar = CreateWindowEx(
                0,                                      //no extennded style
                STATUSCLASSNAME,//Class name for status bar
                LPSTR(NULL),            //No caption (title)
                WS_CHILD|                       //Child window and
                SBS_SIZEGRIP,                   //include a single grip
                0, 0,                           //ignore
                0, 0,                           //ignore
                hWnd,                           //parent window
                (HMENU)ID_OPTION_STATUSBAR,//child window identifier
                hInst,                          //instance
                NULL );                                 //no WM_CREATE parameter
        SendMessage(hStatusBar,SB_SETPARTS,(WPARAM)3,
                                        (LPARAM)(LPINT)textWidth);
        ShowWindow(hStatusBar,SW_SHOW);
        return hStatusBar;
}
```

TL494

SWITCHMODE™ Pulse Width Modulation Control Circuit

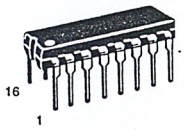
The TL494 is a fixed frequency, pulse width modulation control circuit designed primarily for SWITCHMODE power supply control.

- Complete Pulse Width Modulation Control Circuitry
- On–Chip Oscillator with Master or Slave Operation
- On–Chip Error Amplifiers
- On–Chip 5.0 V Reference
- Adjustable Deadtime Control
- Uncommitted Output Transistors Rated to 500 mA Source or Sink
- Output Control for Push–Pull or Single–Ended Operation
- Undervoltage Lockout

SWITCHMODE
PULSE WIDTH MODULATION
CONTROL CIRCUIT

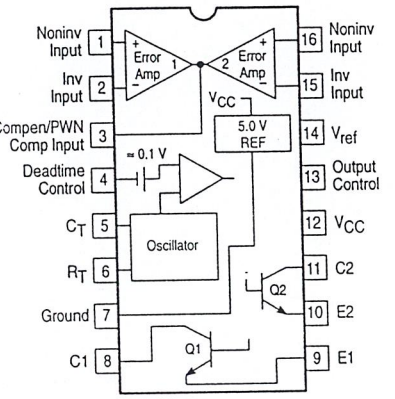
SEMICONDUCTOR
TECHNICAL DATA

D SUFFIX
PLASTIC PACKAGE
CASE 751B
(SO–16)



N SUFFIX
PLASTIC PACKAGE
CASE 648

PIN CONNECTIONS



(Top View)

MAXIMUM RATINGS (Full operating ambient temperature range applies, unless otherwise noted.)

Rating	Symbol	TL494C	TL494I	Unit
Power Supply Voltage	V_{CC}	42		V
Collector Output Voltage	V_{C1}, V_{C2}	42		V
Collector Output Current (Each transistor) (Note 1)	I_{C1}, I_{C2}	500		mA
Amplifier Input Voltage Range	V_{IR}	–0.3 to +42		V
Power Dissipation @ $T_A \leq 45°C$	P_D	1000		mW
Thermal Resistance, Junction–to–Ambient	$R_{\theta JA}$	80		°C/W
Operating Junction Temperature	T_J	125		°C
Storage Temperature Range	T_{stg}	–55 to +125		°C
Operating Ambient Temperature Range TL494C TL494I	T_A	 0 to +70 –25 to +85		°C
Derating Ambient Temperature	T_A	45		°C

NOTE: 1. Maximum thermal limits must be observed.

ORDERING INFORMATION

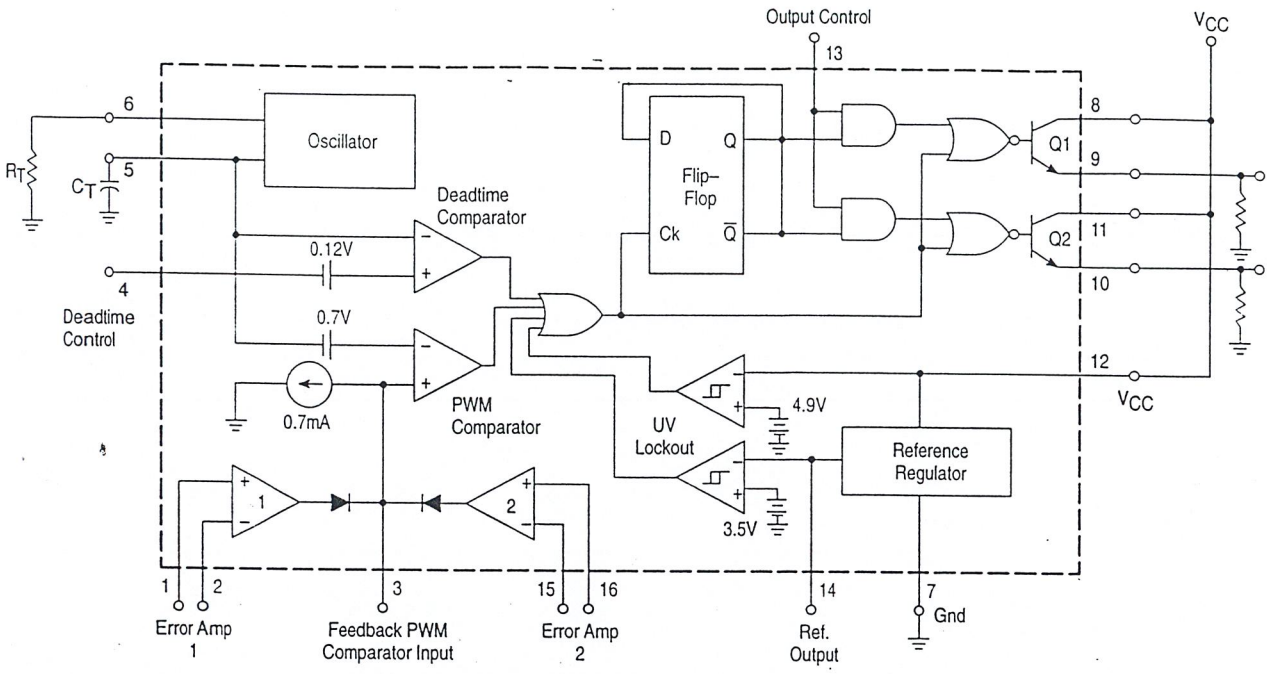
Device	Operating Temperature Range	Package
TL494CD	T_A = 0° to +70°C	SO–16
TL494CN		Plastic
TL494IN	T_A = – 25° to +85°C	Plastic

ELECTRICAL CHARACTERISTICS (V_{CC} = 15 V, C_T = 0.01 µF, R_T = 12 kΩ, unless otherwise noted.)
For typical values T_A = 25°C, for min/max values T_A is the operating ambient temperature range that applies, unless otherwise noted.

Characteristics	Symbol	Min	Typ	Max	Unit
ERROR AMPLIFIER SECTION					
Input Offset Voltage (V_O (Pin 3) = 2.5 V)	V_{IO}	–	2.0	10	mV
Input Offset Current (V_O (Pin 3) = 2.5 V)	I_{IO}	–	5.0	250	nA
Input Bias Current (V_O (Pin 3) = 2.5 V)	I_{IB}	–	–0.1	–1.0	µA
Input Common Mode Voltage Range (V_{CC} = 40 V, T_A = 25°C)	V_{ICR}	–0.3 to V_{CC}–2.0			V
Open Loop Voltage Gain (ΔV_O = 3.0 V, V_O = 0.5 V to 3.5 V, R_L = 2.0 kΩ)	A_{VOL}	70	95	–	dB
Unity–Gain Crossover Frequency (V_O = 0.5 V to 3.5 V, R_L = 2.0 kΩ)	f_{C-}	–	350	–	kHz
Phase Margin at Unity–Gain (V_O = 0.5 V to 3.5 V, R_L = 2.0 kΩ)	ϕ_m	–	65	–	deg.
Common Mode Rejection Ratio (V_{CC} = 40 V)	CMRR	65	90	–	dB
Power Supply Rejection Ratio (ΔV_{CC} = 33 V, V_O = 2.5 V, R_L = 2.0 kΩ)	PSRR	–	100	–	dB
Output Sink Current (V_O (Pin 3) = 0.7 V)	I_{O-}	0.3	0.7	–	mA
Output Source Current (V_O (Pin 3) = 3.5 V)	I_{O+}	2.0	–4.0	–	mA
PWM COMPARATOR SECTION (Test Circuit Figure 11)					
Input Threshold Voltage (Zero Duty Cycle)	V_{TH}	–	2.5	4.5	V
Input Sink Current ($V_{(Pin 3)}$ = 0.7 V)	I_{I-}	0.3	0.7	–	mA
DEADTIME CONTROL SECTION (Test Circuit Figure 11)					
Input Bias Current (Pin 4) ($V_{Pin 4}$ = 0 V to 5.25 V)	I_{IB} (DT)	–	–2.0	–10	µA
Maximum Duty Cycle, Each Output, Push–Pull Mode ($V_{Pin 4}$ = 0 V, C_T = 0.01 µF, R_T = 12 kΩ) ($V_{Pin 4}$ = 0 V, C_T = 0.001 µF, R_T = 30 kΩ)	DC_{max}	45 –	48 45	50 50	%
Input Threshold Voltage (Pin 4) (Zero Duty Cycle) (Maximum Duty Cycle)	V_{th}	– 0	2.8 –	3.3 –	V
OSCILLATOR SECTION					
Frequency (C_T = 0.001 µF, R_T = 30 kΩ)	f_{osc}	–	40	–	kHz
Standard Deviation of Frequency* (C_T = 0.001 µF, R_T = 30 kΩ)	σf_{osc}	–	3.0	–	%
Frequency Change with Voltage (V_{CC} = 7.0 V to 40 V, T_A = 25°C)	Δf_{osc} (ΔV)	–	0.1	–	%
Frequency Change with Temperature (ΔT_A = T_{low} to T_{high}) (C_T = 0.01 µF, R_T = 12 kΩ)	Δf_{osc} (ΔT)	–	–	12	%
UNDERVOLTAGE LOCKOUT SECTION					
Turn–On Threshold (V_{CC} increasing, I_{ref} = 1.0 mA)	V_{th}	5.5	6.43	7.0	V
TOTAL DEVICE					
Standby Supply Current (Pin 6 at V_{ref}, All other inputs and outputs open) (V_{CC} = 15 V) (V_{CC} = 40 V)	I_{CC}	– –	5.5 7.0	10 15	mA
Average Supply Current (C_T = 0.01 µF, R_T = 12 kΩ, $V_{(Pin 4)}$ = 2.0 V) (V_{CC} = 15 V) (See Figure 12)		–	7.0	–	mA

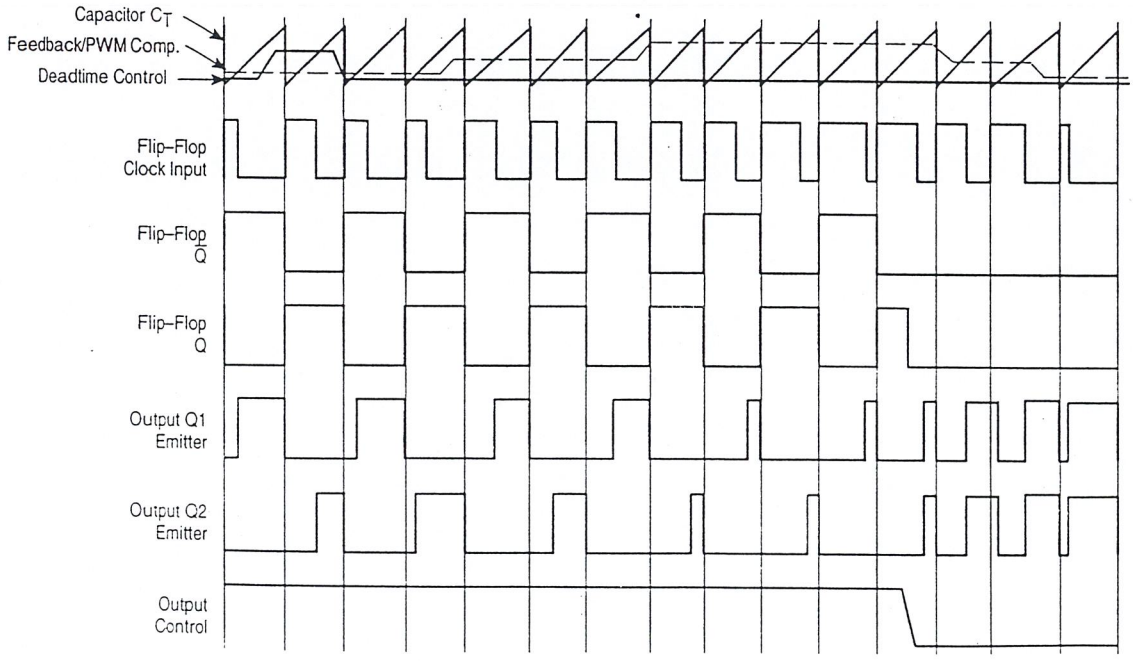
* Standard deviation is a measure of the statistical distribution about the mean as derived from the formula, $\sigma = \sqrt{\frac{\sum_{n=1}^{N} (X_n - \overline{X})^2}{N-1}}$

Figure 1. Representative Block Diagram



This device contains 46 active transistors.

Figure 2. Timing Diagram



APPLICATIONS INFORMATION

Description

The TL494 is a fixed–frequency pulse width modulation control circuit, incorporating the primary building blocks required for the control of a switching power supply. (See Figure 1.) An internal–linear sawtooth oscillator is frequency–programmable by two external components, R_T and C_T. The approximate oscillator frequency is determined by:

$$f_{osc} \approx \frac{1.1}{R_T \bullet C_T}$$

For more information refer to Figure 3.

Output pulse width modulation is accomplished by comparison of the positive sawtooth waveform across capacitor C_T to either of two control signals. The NOR gates, which drive output transistors Q1 and Q2, are enabled only when the flip–flop clock–input line is in its low state. This occurs only during that portion of time when the sawtooth voltage is greater than the control signals. Therefore, an increase in control–signal amplitude causes a corresponding linear decrease of output pulse width. (Refer to the Timing Diagram shown in Figure 2.)

The control signals are external inputs that can be fed into the deadtime control, the error amplifier inputs, or the feedback input. The deadtime control comparator has an effective 120 mV input offset which limits the minimum output deadtime to approximately the first 4% of the sawtooth–cycle time. This would result in a maximum duty cycle on a given output of 96% with the output control grounded, and 48% with it connected to the reference line. Additional deadtime may be imposed on the output by setting the deadtime–control input to a fixed voltage, ranging between 0 V to 3.3 V.

Functional Table

Input/Output Controls	Output Function	$\frac{f_{out}}{f_{osc}}$ =
Grounded	Single–ended PWM @ Q1 and Q2	1.0
@ V_{ref}	Push–pull Operation	0.5

The pulse width modulator comparator provides a means for the error amplifiers to adjust the output pulse width from the maximum percent on–time, established by the deadtime control input, down to zero, as the voltage at the feedback pin varies from 0.5 V to 3.5 V. Both error amplifiers have a common mode input range from –0.3 V to (V_{CC} – 2V), and may be used to sense power–supply output voltage and current. The error–amplifier outputs are active high and are ORed together at the noninverting input of the pulse–width modulator comparator. With this configuration, the amplifier that demands minimum output on time, dominates control of the loop.

When capacitor C_T is discharged, a positive pulse is generated on the output of the deadtime comparator, which clocks the pulse–steering flip–flop and inhibits the output transistors, Q1 and Q2. With the output–control connected to the reference line, the pulse–steering flip–flop directs the modulated pulses to each of the two output transistors alternately for push–pull operation. The output frequency is equal to half that of the oscillator. Output drive can also be taken from Q1 or Q2, when single–ended operation with a maximum on–time of less than 50% is required. This is desirable when the output transformer has a ringback winding with a catch diode used for snubbing. When higher output–drive currents are required for single–ended operation, Q1 and Q2 may be connected in parallel, and the output–mode pin must be tied to ground to disable the flip–flop. The output frequency will now be equal to that of the oscillator.

The TL494 has an internal 5.0 V reference capable of sourcing up to 10 mA of load current for external bias circuits. The reference has an internal accuracy of ±5.0% with a typical thermal drift of less than 50 mV over an operating temperature range of 0° to 70°C.

Figure 3. Oscillator Frequency versus Timing Resistance

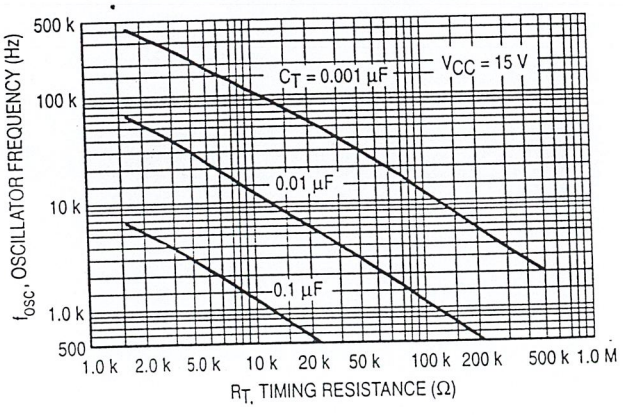


Figure 4. Open Loop Voltage Gain and Phase versus Frequency

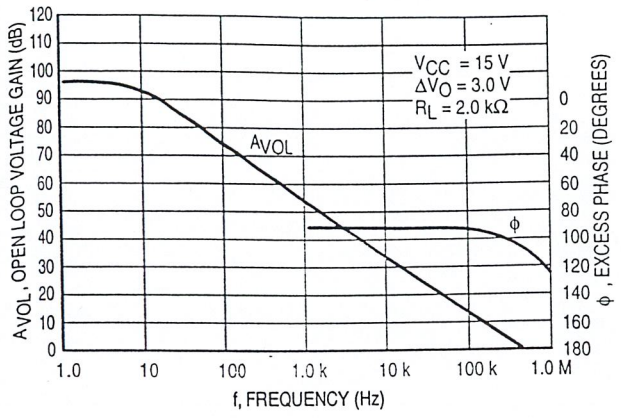


Figure 5. Percent Deadtime versus Oscillator Frequency

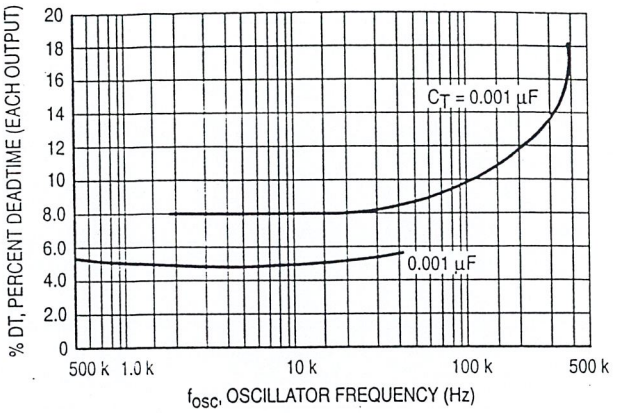


Figure 6. Percent Duty Cycle versus Deadtime Control Voltage

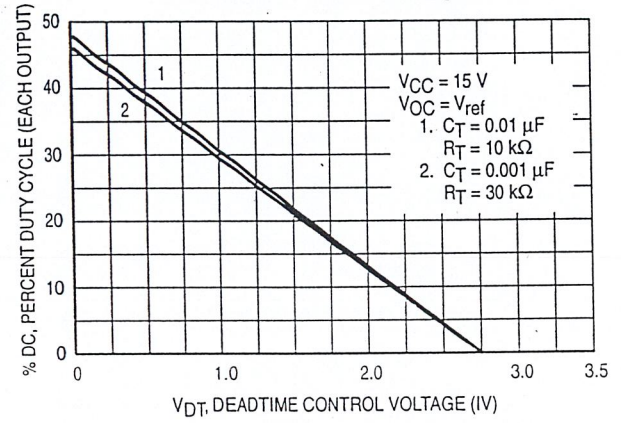


Figure 7. Emitter–Follower Configuration Output Saturation Voltage versus Emitter Current

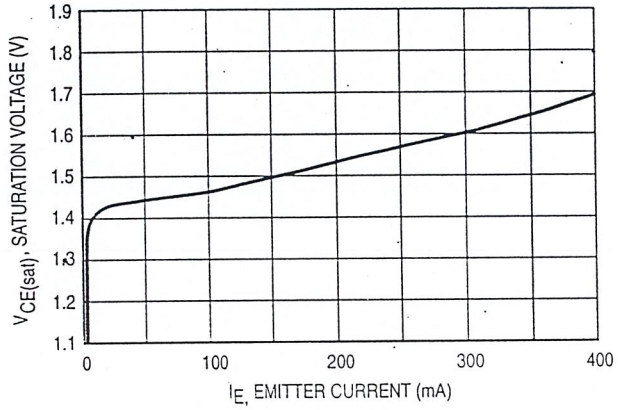


Figure 8. Common–Emitter Configuration Output Saturation Voltage versus Collector Current

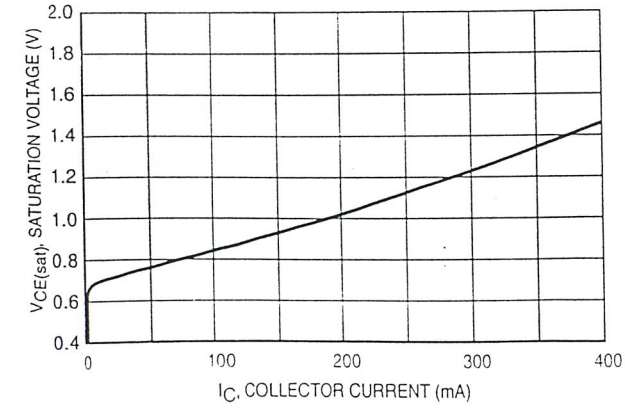
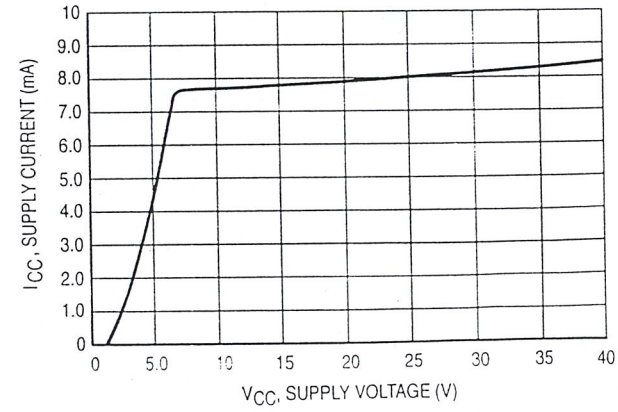


Figure 9. Standby Supply Current versus Supply Voltage



RECOMMENDED OPERATING CONDITIONS

Characteristics	Symbol	Min	Typ	Max	Unit
Power Supply Voltage	V_{CC}	7.0	15	40	V
Collector Output Voltage	V_{C1}, V_{C2}	–	30	40	V
Collector Output Current (Each transistor)	I_{C1}, I_{C2}	–	–	200	mA
Amplified Input Voltage	V_{in}	–0.3	–	V_{CC} – 2.0	V
Current Into Feedback Terminal	I_{fb}	–	–	0.3	mA
Reference Output Current	I_{ref}	–	–	10	mA
Timing Resistor	R_T	1.8	30	500	kΩ
Timing Capacitor	C_T	0.0047	0.001	10	µF
Oscillator Frequency	f_{osc}	1.0	40	200	kHz

ELECTRICAL CHARACTERISTICS (V_{CC} = 15 V, C_T = 0.01 µF, R_T = 12 kΩ, unless otherwise noted.)

For typical values T_A = 25°C, for min/max values T_A is the operating ambient temperature range that applies, unless otherwise noted.

Characteristics	Symbol	Min	Typ	Max	Unit
REFERENCE SECTION					
Reference Voltage (I_O = 1.0 mA)	V_{ref}	4.75	5.0	5.25	V
Line Regulation (V_{CC} = 7.0 V to 40 V)	Reg_{line}	–	2.0	25	mV
Load Regulation (I_O = 1.0 mA to 10 mA)	Reg_{load}	–	3.0	15	mV
Short Circuit Output Current (V_{ref} = 0 V)	I_{SC}	15	35	75	mA
OUTPUT SECTION					
Collector Off–State Current (V_{CC} = 40 V, V_{CE} = 40 V)	$I_{C(off)}$	–	2.0	100	µA
Emitter Off–State Current V_{CC} = 40 V, V_C = 40 V, V_E = 0 V)	$I_{E(off)}$	–	–	–100	µA
Collector–Emitter Saturation Voltage (Note 2)					V
Common–Emitter (V_E = 0 V, I_C = 200 mA)	$V_{sat(C)}$	–	1.1	1.3	
Emitter–Follower (V_C = 15 V, I_E = –200 mA)	$V_{sat(E)}$	–	1.5	2.5	
Output Control Pin Current					µA
Low State (V_{OC} ≤ 0.4 V)	I_{OCL}	–	10	–	
High State (V_{OC} = V_{ref})	I_{OCH}	–	0.2	3.5	mA
Output Voltage Rise Time	t_r				ns
Common–Emitter (See Figure 12)		–	100	200	
Emitter–Follower (See Figure 13)		–	100	200	
Output Voltage Fall Time	t_f				ns
Common–Emitter (See Figure 12)		–	25	100	
Emitter–Follower (See Figure 13)		–	40	100	

NOTE: 2. Low duty cycle pulse techniques are used during test to maintain junction temperature as close to ambient temperature as possible.

Figure 10. Error–Amplifier Characteristics

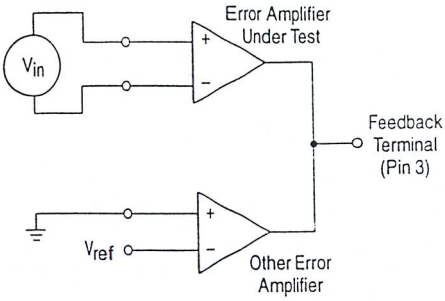


Figure 11. Deadtime and Feedback Control Circuit

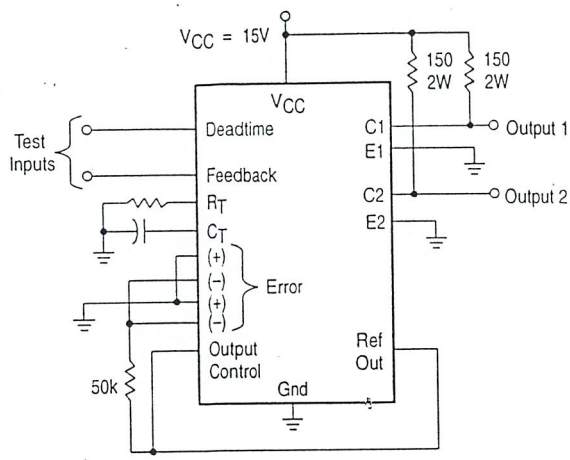


Figure 12. Common–Emitter Configuration Test Circuit and Waveform

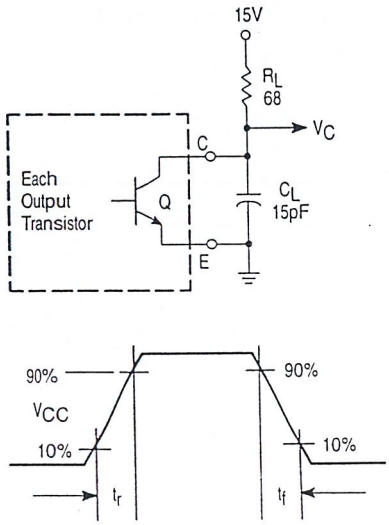


Figure 13. Emitter–Follower Configuration Test Circuit and Waveform

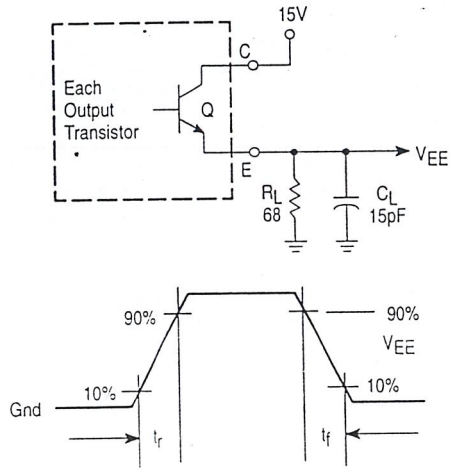
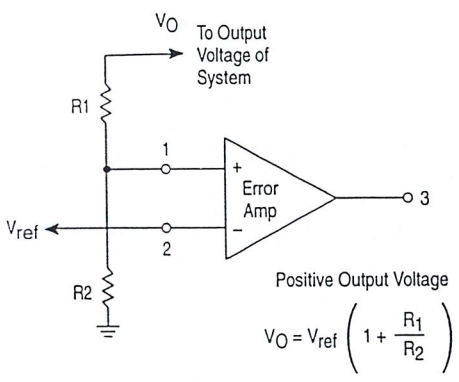


Figure 14. Error–Amplifier Sensing Techniques



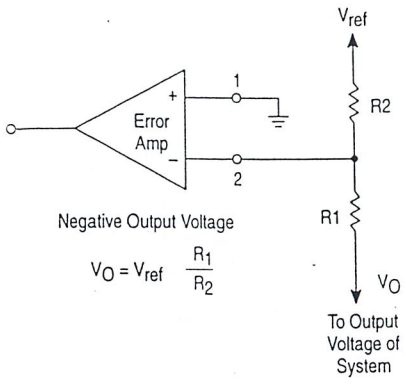


Figure 15. Deadtime Control Circuit

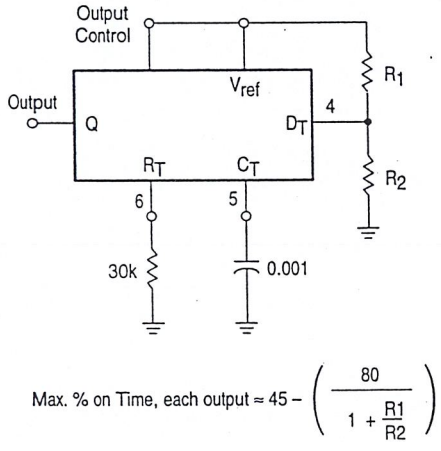


Figure 16. Soft–Start Circuit

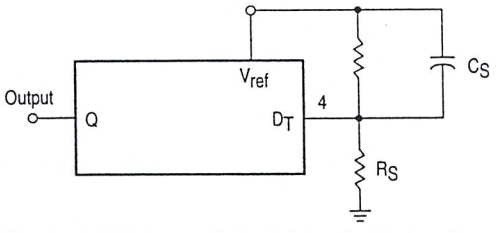
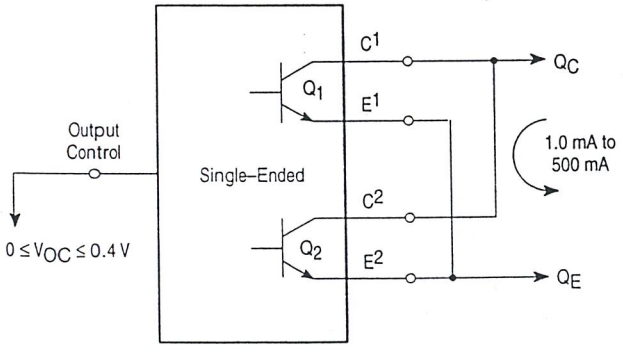


Figure 17. Output Connections for Single–Ended and Push–Pull Configurations



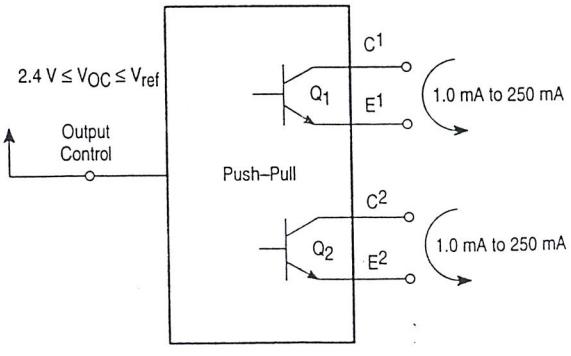


Figure 18. Slaving Two or More Control Circuits

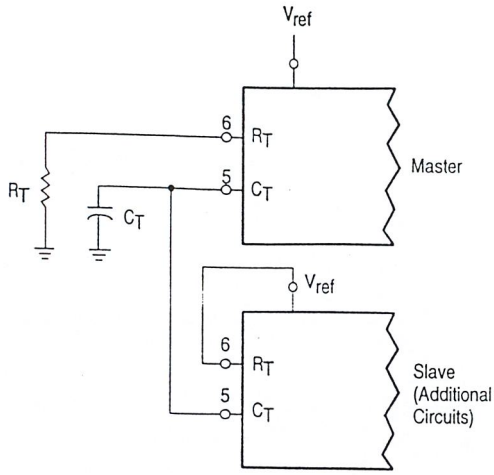


Figure 19. Operation with V_{in} > 40 V Using External Zener

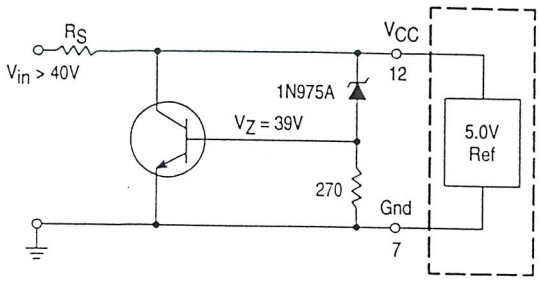
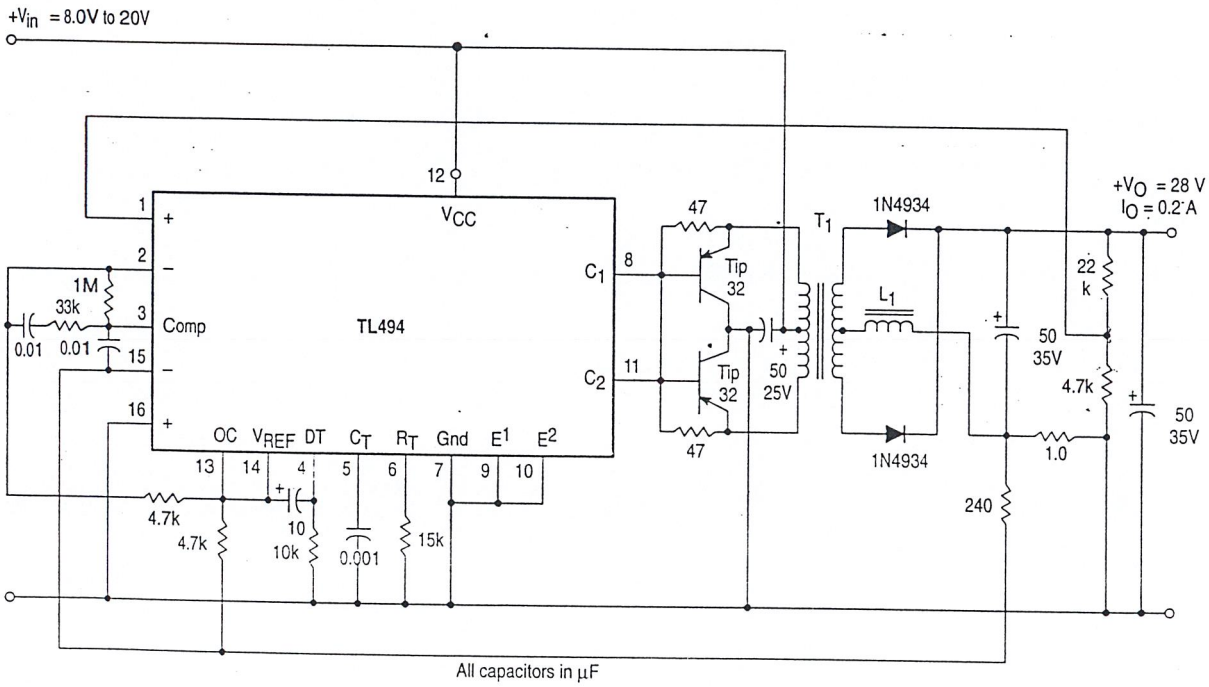


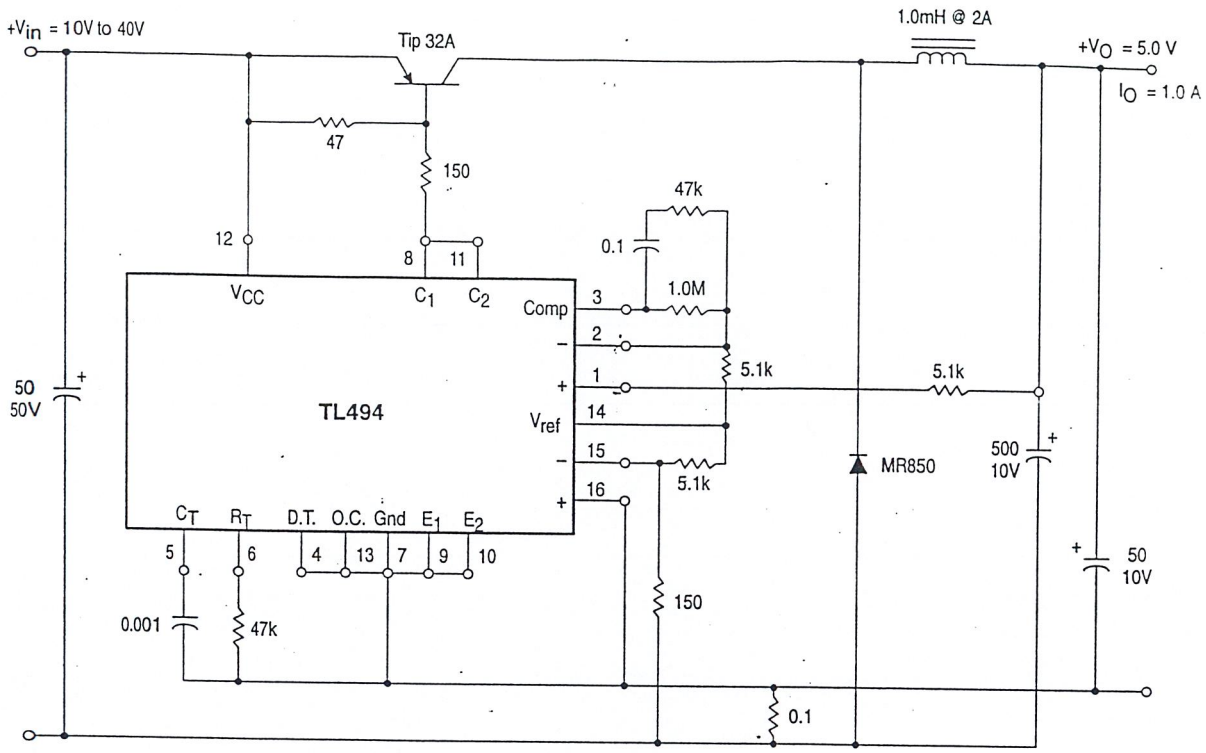
Figure 20. Pulse Width Modulated Push–Pull Converter



Test	Conditions	Results
Line Regulation	V_{in} = 10 V to 40 V	14 mV 0.28%
Load Regulation	V_{in} = 28 V, I_O = 1.0 mA to 1.0 A	3.0 mV 0.06%
Output Ripple	V_{in} = 28 V, I_O = 1.0 A	65 mV pp P.A.R.D.
Short Circuit Current	V_{in} = 28 V, R_L = 0.1 Ω	1.6 A
Efficiency	V_{in} = 28 V, I_O = 1.0 A	71%

L1 – 3.5 mH @ 0.3 A
T1 – Primary: 20T C.T. #28 AWG
Secondary: 12OT C.T. #36 AWG
Core: Ferroxcube 1408P–L00–3CB

Figure 21. Pulse Width Modulated Step–Down Converter



Test	Conditions	Results
Line Regulation	V_{in} = 8.0 V to 40 V	3.0 mV 0.01%
Load Regulation	V_{in} = 12.6 V, I_O = 0.2 mA to 200 mA	5.0 mV 0.02%
Output Ripple	V_{in} = 12.6 V, I_O = 200 mA	40 mV pp P.A.R.D.
Short Circuit Current	V_{in} = 12.6 V, R_L = 0.1 Ω	250 mA
Efficiency	V_{in} = 12.6 V, I_O = 200 mA	72%

DAC0808/DAC0807/DAC0806 8-Bit D/A Converters

General Description

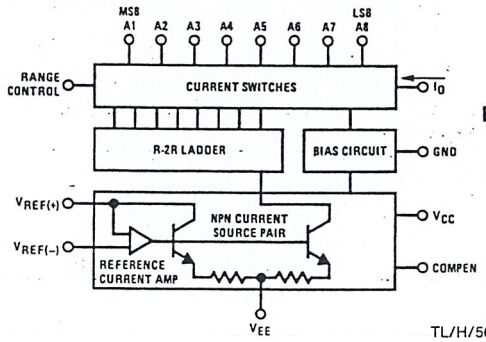
The DAC0808 series is an 8-bit monolithic digital-to-analog converter (DAC) featuring a full scale output current settling time of 150 ns while dissipating only 33 mW with ±5V supplies. No reference current (I_{REF}) trimming is required for most applications since the full scale output current is typically ±1 LSB of 255 I_{REF}/ 256. Relative accuracies of better than ±0.19% assure 8-bit monotonicity and linearity while zero level output current of less than 4 μA provides 8-bit zero accuracy for $I_{REF} \geq 2$ mA. The power supply currents of the DAC0808 series are independent of bit codes, and exhibits essentially constant device characteristics over the entire supply voltage range.

The DAC0808 will interface directly with popular TTL, DTL or CMOS logic levels, and is a direct replacement for the MC1508/MC1408. For higher speed applications, see DAC0800 data sheet.

Features

- Relative accuracy: ±0.19% error maximum (DAC0808)
- Full scale current match: ±1 LSB typ
- 7 and 6-bit accuracy available (DAC0807, DAC0806)
- Fast settling time: 150 ns typ
- Noninverting digital inputs are TTL and CMOS compatible
- High speed multiplying input slew rate: 8 mA/μs
- Power supply voltage range: ±4.5V to ±18V
- Low power consumption: 33 mW @ ±5V

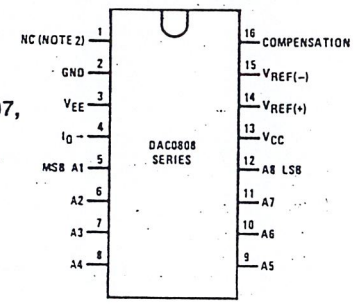
Block and Connection Diagrams



TL/H/5687–1

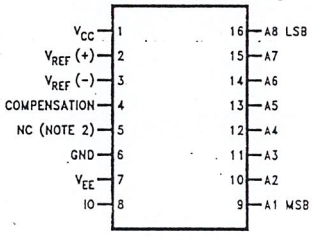
Order Number DAC0808, DAC0807, or DAC0806
See NS Package Number J16A, M16A or N16A

Dual-In-Line Package



TL/H/5687–2

Small-Outline Package



TL/H/5687–13

Top View

Ordering Information

ACCURACY	OPERATING TEMPERATURE RANGE	ORDER NUMBERS				
		J PACKAGE (J16A)*		N PACKAGE (N16A)*		SO PACKAGE (M16A)
7-bit	$0°C \leq T_A \leq +75°C$	DAC0807LCJ	MC1408L7	DAC0808LCN	MC1408P8	DAC0808LCM
				DAC0807LCN	MC1408P7	DAC0807LCM
6-bit	$0°C \leq T_A \leq +75°C$	DAC0806LCJ	MC1408L6	DAC0806LCN	MC1408P6	DAC0806LCM

*Note. Devices may be ordered by using either order number.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Power Supply Voltage	
V_{CC}	$+18\ V_{DC}$
V_{EE}	$-18\ V_{DC}$
Digital Input Voltage, V5–V12	$-10\ V_{DC}$ to $+18\ V_{DC}$
Applied Output Voltage, V_O	$-11\ V_{DC}$ to $+18\ V_{DC}$
Reference Current, I_{14}	5 mA
Reference Amplifier Inputs, V14, V15	V_{CC}, V_{EE}
Power Dissipation (Note 3)	1000 mW
ESD Susceptibility (Note 4)	TBD
Storage Temperature Range	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (Plastic)	260°C
Dual-In-Line Package (Ceramic)	300°C
Surface Mount Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C

Operating Ratings

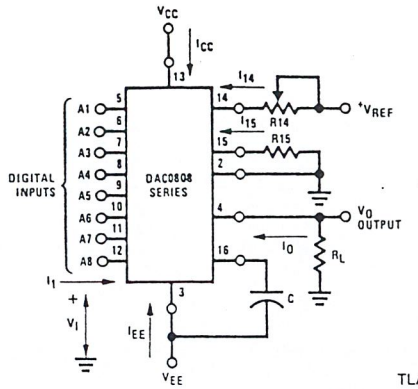
Temperature Range	$T_{MIN} \le T_A \le T_{MAX}$
DAC0808LC Series	$0 \le T_A \le +75°C$

Electrical Characteristics

(V_{CC} = 5V, V_{EE} = −15 V_{DC}, $V_{REF}/R14$ = 2 mA, DAC0808: T_A = −55°C to +125°C, DAC0808C, DAC0807C, DAC0806C, T_A = 0°C to +75°C, and all digital inputs at high logic level unless otherwise noted.)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
E_r	Relative Accuracy (Error Relative to Full Scale I_O)	*(Figure 4)*				%
	DAC0808LC (LM1408-8)				**±0.19**	%
	DAC0807LC (LM1408-7), (Note 5)				±0.39	%
	DAC0806LC (LM1408-6), (Note 5)				±0.78	%
	Settling Time to Within ½ LSB (Includes t_{PLH})	T_A = 25°C (Note 6), *(Figure 5)*		150		ns
t_{PLH}, t_{PHL}	Propagation Delay Time	T_A = 25°C, *(Figure 5)*		30	100	ns
TCI_O	Output Full Scale Current Drift			±20		ppm/°C
MSB	Digital Input Logic Levels	*(Figure 3)*				
V_{IH}	High Level, Logic "1"		2			V_{DC}
V_{IL}	Low Level, Logic "0"				0.8	V_{DC}
MSB	Digital Input Current	*(Figure 3)*				
	High Level	V_{IH} = 5V		0	0.040	mA
	Low Level	V_{IL} = 0.8V		−0.003	−0.8	mA
I_{15}	Reference Input Bias Current	*(Figure 3)*		−1	−3	μA
	Output Current Range	*(Figure 3)*				
		V_{EE} = −5V	0	2.0	2.1	mA
		V_{EE} = −15V, T_A = 25°C	0	2.0	4.2	mA
I_O	Output Current	V_{REF} = 2.000V, R14 = 1000Ω, *(Figure 3)*	1.9	1.99	2.1	mA
	Output Current, All Bits Low	*(Figure 3)*		0	4	μA
	Output Voltage Compliance (Note 2)	$E_r \le 0.19\%$, T_A = 25°C			−0.55, +0.4	V_{DC}
	V_{EE} = −5V, I_{REF} = 1 mA				−5.0, +0.4	V_{DC}
	V_{EE} Below −10V					

Test Circuits



TL/H/5687–6

FIGURE 3. Notation Definitions Test Circuit (Note 7)

V_I and I_1 apply to inputs A1–A8.

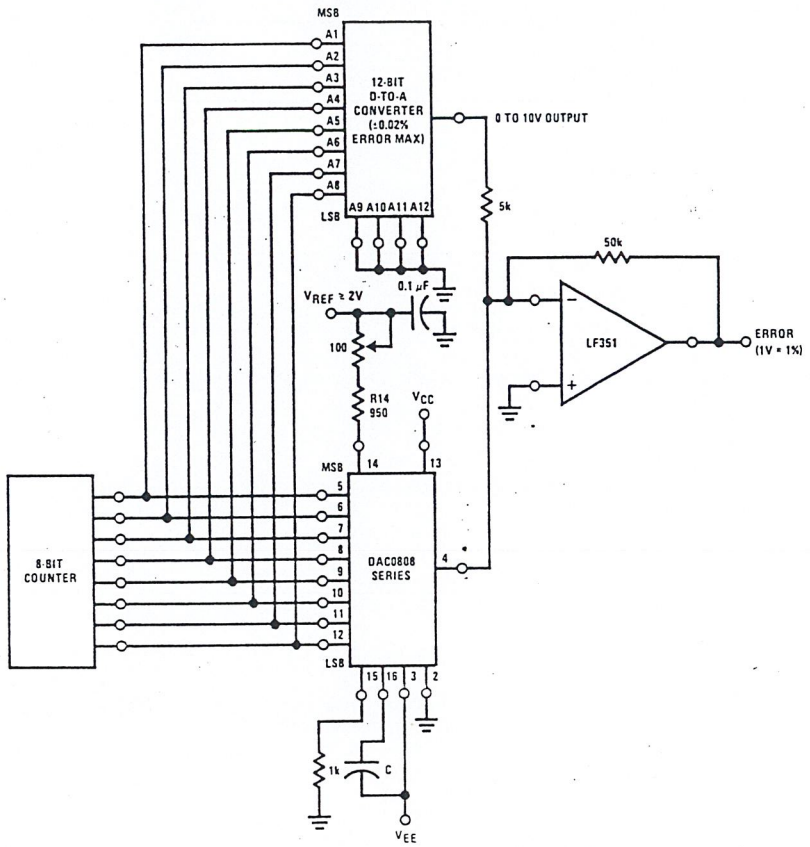
The resistor tied to pin 15 is to temperature compensate the bias current and may not be necessary for all applications.

$$I_O = K\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}\right)$$

where $K \cong \frac{V_{REF}}{R14}$

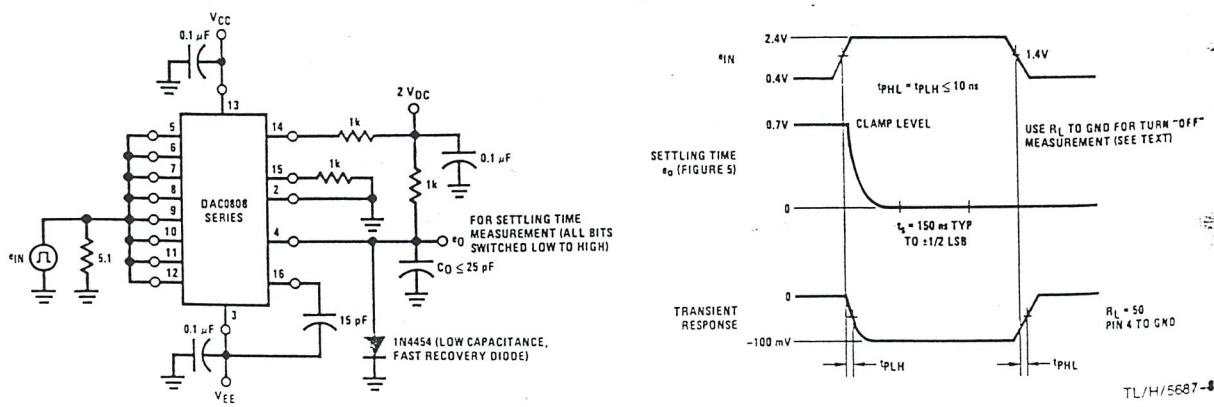
and A_N = "1" if A_N is at high level

A_N = "0" if A_N is at low level



TL/H/5687–7

FIGURE 4. Relative Accuracy Test Circuit (Note 7)



TL/H/5687–8

FIGURE 5. Transient Response and Settling Time (Note 7)

Electrical Characteristics (Continued)

(V_{CC} = 5V, V_{EE} = −15 V_{DC}, $V_{REF}/R14$ = 2 mA, DAC0808: T_A = −55°C to +125°C, DAC0808C, DAC0807C, DAC0806C, T_A = 0°C to +75°C, and all digital inputs at high logic level unless otherwise noted.)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
SRI_{REF}	Reference Current Slew Rate	*(Figure 6)*	4	8		mA/μs
	Output Current Power Supply Sensitivity	$-5V \le V_{EE} \le -16.5V$		0.05	2.7	μA/V
	Power Supply Current (All Bits Low)	*(Figure 3)*				
I_{CC}				2.3	22	mA
I_{EE}				−4.3	−13	mA
	Power Supply Voltage Range	T_A = 25°C, *(Figure 3)*				
V_{CC}			4.5	5.0	5.5	V_{DC}
V_{EE}			−4.5	−15	−16.5	V_{DC}
	Power Dissipation					
	All Bits Low	V_{CC} = 5V, V_{EE} = −5V		33	170	mW
		V_{CC} = 5V, V_{EE} = −15V		106	305	mW
	All Bits High	V_{CC} = 15V, V_{EE} = −5V		90		mW
		V_{CC} = 15V, V_{EE} = −15V		160		mW

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

Note 2: Range control is not required.

Note 3: The maximum power dissipation must be derated at elevated temperatures and is dictated by T_{JMAX}, θ_{JA}, and the ambient temperature, T_A. The maximum allowable power dissipation at any temperature is $P_D = (T_{JMAX} - T_A)/\theta_{JA}$ or the number given in the Absolute Maixmum Ratings, whichever is lower. For this device, T_{JMAX} = 125°C, and the typical junction-to-ambient thermal resistance of the dual-in-line J package when the board mounted is 100°C/W. For the dual-in-line N package, this number increases to 175°C/W and for the small outline M package this number is 100°C/W.

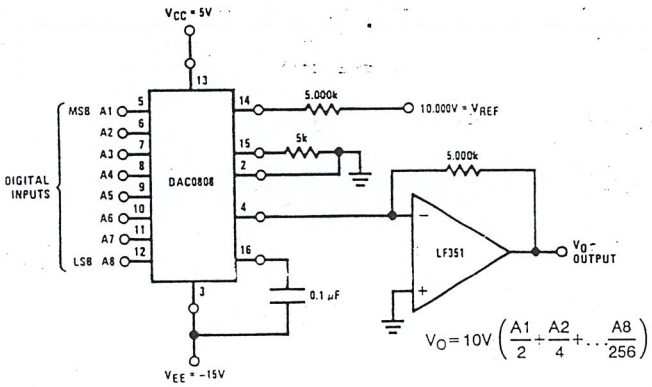
Note 4: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 5: All current switches are tested to guarantee at least 50% of rated current.

Note 6: All bits switched.

Note 7: Pin-out numbers for the DAL080X represent the dual-in-line package. The small outline package pinout differs from the dual-in-line package.

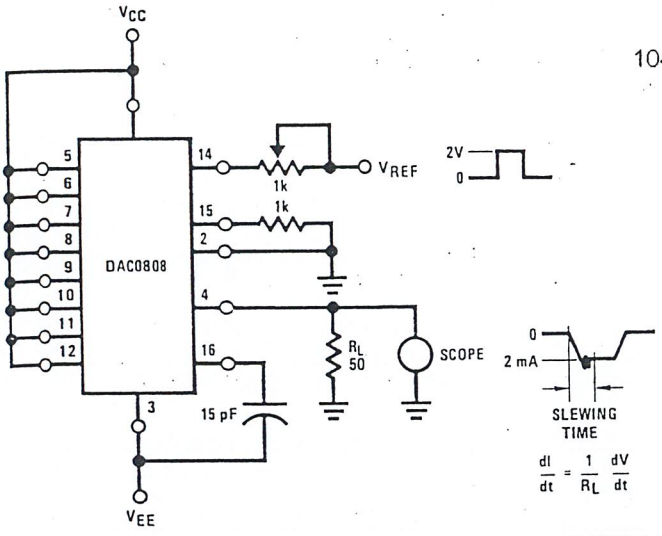
Typical Application



TL/H/5687–3

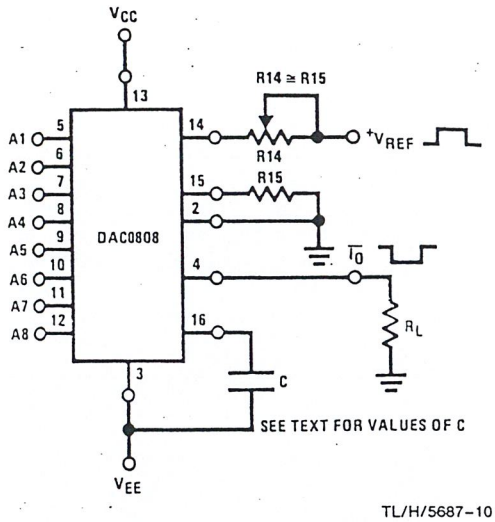
FIGURE 1. + 10V Output Digital to Analog Converter (Note 7)

Test Circuits (Continued)



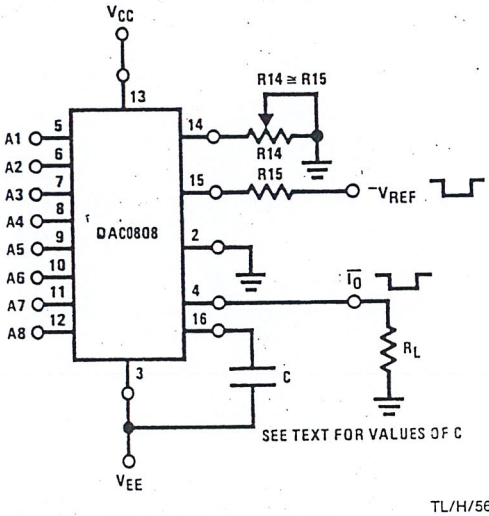
TL/H/5687–9

FIGURE 6. Reference Current Slew Rate Measurement (Note 7)



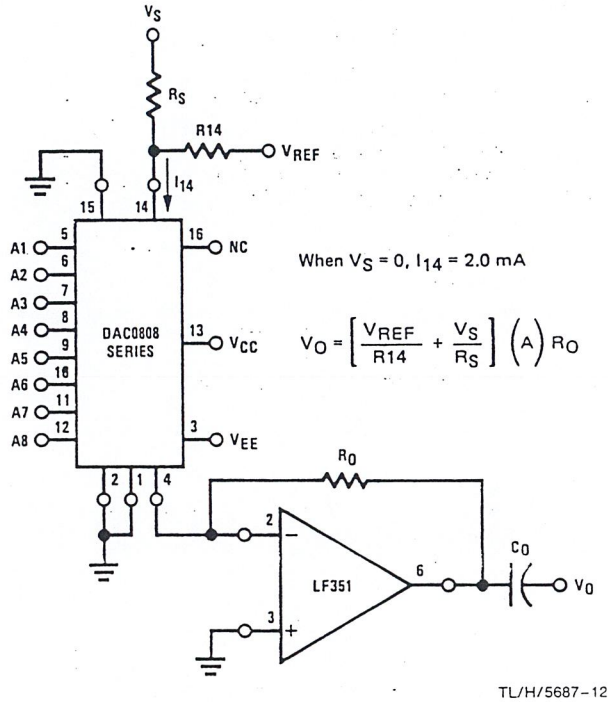
TL/H/5687–10

FIGURE 7. Positive V_{REF} (Note 7)



TL/H/5687–11

FIGURE 8. Negative V_{REF} (Note 7)



TL/H/5687–12

FIGURE 9. Programmable Gain Amplifier or Digital Attenuator Circuit (Note 7)

Application Hints

REFERENCE AMPLIFIER DRIVE AND COMPENSATION

The reference amplifier provides a voltage at pin 14 for converting the reference voltage to a current, and a turn-around circuit or current mirror for feeding the ladder. The reference amplifier input currrent, I_{14}, must always flow into pin 14, regardless of the set-up method or reference voltage polarity.

Connections for a positive voltage are shown in *Figure 7*. The reference voltage source supplies the full current I_{14}. For bipolar reference signals, as in the multiplying mode, R15 can be tied to a negative voltage corresponding to the minimum input level. It is possible to eliminate R15 with only a small sacrifice in accuracy and temperature drift.

The compensation capacitor value must be increased with increases in R14 to maintain proper phase margin; for R14 values of 1, 2.5 and 5 kΩ, minimum capacitor values are 15, 37 and 75 pF. The capacitor may be tied to either V_{EE} or ground, but using V_{EE} increases negative supply rejection.

กิตติกรรมประกาศ

โครงงานนี้จะไม่อาจสำเร็จลุล่วงไปได้ด้วยดี ถ้าหากไม่ได้รับความช่วยเหลือจากบุคคลหลาย ๆ ท่าน ไม่ว่าจะเป็น เพื่อน ๆ พี่ปริญญาโท อาจารย์หลาย ๆ ท่าน ร่วมทั้ง ภาควิชาเครื่องกลที่ได้ให้คำแนะนำในการปฏิบัติงานด้านฮาร์ดแวร์ และโดยเฉพาะอย่างยิ่ง ดร. สุรพันธ์ เอื้อไพบูลย์ ที่ให้คำแนะนำ และ คำปรึกษาต่าง ๆ มาโดยตลอด ขอขอบคุณคุณพ่อ คุณแม่ ที่ได้ให้กำลังใจที่ดีเสมอมา และผู้มีพระคุณหลาย ๆ ท่านที่มิอาจกล่าวได้หมด ณ ที่นี้

เกศ บัวลำใย

(นายเกศ บัวลำใย)

(นางสาว ปรียาภรณ์ ภักดี)

ผู้จัดทำ

เอกสารอ้างอิง

1.) อนิรุต ลิ่วหาทอง , " การเขียนโปรแกรมบนวินโดวส์ ด้วย Microsoft C++ " ซีเอ็ดยูเคชั่น

2.) สุนทร วิทูสุรพจน์ , " การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051 " บริษัท ซีเอ็ดยูเคชั่น จำกัดมหาชน , หน้า 29 - 46

3.) อ.ธีรวัฒน์ ประกอบผล , " การใช้งานไมโครคอนเลอร์ " บริษัท ดวงกมลสมัย จำกัด , พ.ศ. 2540 , หน้า 19 – 43

4.) ดร.ดวงแก้ว สวามิภักดิ์ , " การใช้โปรแกรมภาษา C " บริษัท ซีเอ็ดยูเคชัน , พ.ศ. 2539 ,

5.) Scott Stanfield with Ralph Arverson , " VISUAL C++ 4 How – To " , waite Droup Press TM A Division of Same Publishing Corte Madera , CA , 1996

เวปไซต์ ที่เข้าไปค้นหาข้อมูลทางอินเตอร์เน็ท

1. http://www.national.com
2. http://www.expert-exchange.com
3. http://www.microsolf.com