การตรวจจับเลนถนนโดยคอมพิวเตอร์

COMPUTER VISION (LANE DETECTION)

โดย

นายสุพจน์ วัชระดนัย

นายสุรชาติ เหล็กงาม

นายโอภาส อัครจันทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

การตรวจจับเลนถนนโดยคอมพิวเตอร์

COMPUTER VISION (LANE DETECTION)

โดย

นายสุพจน์ วัชระดนัย
นายสุรชาติ เหล็กงาม
นายโอภาส อัครจันทร์

อาจารย์ที่ปรึกษา
ดร.สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การตรวจจับเลนถนนโดยคอมพิวเตอร์

Computer Vision (Lane Detection)

จัดทำโดย นายสุพจน์ วัชระดนัย

นายสุรชาติ เหล็กงาม

นายโอภาส อัครจันทร์

ลงชื่อ..... อาจารย์ที่ปรึกษา

(ดร.สุรพันธ์ เอื้อไพบูลย์)

เรื่อง การตรวจจับเลนถนนโดยคอมพิวเตอร์

Computer Vision (Lane Detection)

จัดทำโดย นายสุพจน์ วัชระดนัย

นายสุรชาติ เหล็กงาม

นายโอภาส อัครจันทร์

อาจารย์ที่ปรึกษา ดร.สุรพันธ์ เอื้อไพบูลย์

โครงงานนี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ..... อาจารย์ที่ปรึกษา

(ดร.สุรพันธ์ เอื้อไพบูลย์)

วันที่ 26 มี.ค. 41

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้มิอาจจะสำเร็จลุล่วงไปได้โดยปราศจากความช่วยเหลือจากบุคคลหลาย ๆ ท่าน เพื่อน ๆ ภาคอิเล็กทรอนิกส์ ที่ให้คำแนะนำ อุปกรณ์และกำลังใจ เพื่อน ๆ ภาคคอนโทรลที่อนุเคราะห์อุปกรณ์ พี่ปริญญาโทที่ให้คำแนะนำ โดยเฉพาะอย่างยิ่ง ดร.สุรพันธ์ เอื้อไพบูลย์ ที่ได้ให้คำปรึกษาและความช่วยเหลือต่าง ๆ อย่างมากมาย ทางผู้ทำการวิจัยต้องขอขอบพระคุณทุกท่านอย่างสูงไว้ ณ ที่นี้ด้วย

สุพจน์ วัชระดนัย

(นายสุพจน์ วัชระดนัย)

(นายสุรชาติ เหล็กงาม)

โอภาส อัครจันทร์

(นายโอภาส อัครจันทร์)

ผู้จัดทำ

การตรวจจับเลนถนนโดยคอมพิวเตอร์

นายสุพจน์ วัชระดนัย
นายสุรชาติ เหล็กงาม
นายโอภาส อัครจันทร์
ดร.สุรพันธ์ เอื้อไพบูลย์ (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2540

บทคัดย่อ

จุดมุ่งหมายหลักสำหรับการพัฒนาระบบตรวจจับเลนถนน ซึ่งเป็นงานที่ยากเมื่อต้องประมวลผลในเวลาจริง (Real Time) ขณะที่พาหนะกำลังมีการเคลื่อนที่ จะต้องมีฮาร์ดแวร์เฉพาะที่สามารถทำงานร่วมกัน เพื่อที่จะทำให้การทำงานของระบบมีประสิทธิภาพ และเสียค่าใช้จ่ายน้อยที่สุด

โครงงานนี้จะจำลองการประมวลผลภาพอย่างหยาบสำหรับการควบคุม การนำภาพที่รับเข้ามาเพื่อที่จะนำมาปรับปรุงเปลี่ยนแปลง โดยใช้ความละเอียดของการประมวลผลที่ต่างกัน ฟังก์ชันในการประมวลผลที่ขนาดไม่ใหญ่นัก ทำให้ภาพที่ถูกเปลี่ยนแปลงได้เป็นภาพที่ต้องการออกมา โดยภาพที่จะทำการประมวลผลจะได้มาจากกล้องวิดีโอที่ติดอยู่ที่ตัวรถ หลังจากนั้นจะทำการนำภาพที่ได้ผ่านกระบวนการทินนิ่ง (Thinning) เพื่อให้ได้เส้นขอบของเลนถนน และทำการหาจุดกึ่งกลางของเลนถนน 3 จุด สำหรับใช้ในการเปรียบเทียบความแตกต่างของจุดทั้งสามและนำค่าที่ได้ไปใช้ในการควบคุมการเคลื่อนที่ของตัวรถ

การควบคุมการเคลื่อนที่ของรถจะใช้การควบคุมความเร็วของสเต็ปปิ้งมอเตอร์ (Stepping Motor) ที่ล้อทั้งสองข้าง ทำให้รถที่ใช้ในระบบนี้สามารถเคลื่อนที่ไปตามถนนได้โดยอัตโนมัติโดยใช้ระบบการตรวจจับเลนถนนนี้

Computer Vision (Lane Detection.)

Mr. Supot Watcharadanai
Mr. Surachat Lekngam
Mr. Opas Akkarajun
Dr. Suraphan Airphiboon (Advisor)
2rd Semester, Educational Year 1997

Abstract

The main aim of this thesis is the development of a vision - based road detection system fast enough to cope with the difficult real - time constraints imposed by moving vehicle applications. The hardware platform, a special - purpose massively parallel system, has been chosen to minimize system production and operational costs.

The input image is assumed to contain distorted version of giver template; a multiresolution stretching proceed is used to reshape the original template in accordance with the acquired image content, minimizing a potential function. The distorted template is the process output. The image from video camera on vehicle is used to process. And take the input image to thinning function to get the boundary of road. After that find three middle point of lane which use to compare the value of them. We use that value to control vehicle. We can control vehicle by stepping motor. The vehicle can travel using this lane detection system by automatically.

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

เนื่องจากการศึกษาการตรวจจับเลนหรือถนนต่าง ๆ เป็นเทคโนโลยีที่อาจจะเรียกได้ว่าใหม่สำหรับการขับเคลื่อนพาหนะโดยใช้ระบบอัตโนมัติ ซึ่งปัจจุบันได้มีนักค้นคว้าหลายท่านได้กำลังทำการวิจัยถึงแนวทางและหลักการต่าง ๆ เพื่อที่จะให้ได้วิธีที่ดีที่สุด ในทางคอมพิวเตอร์ก็คือการใช้เวลาในการประมวลผลน้อยที่สุด ซึ่งจะส่งผลให้ฮาร์ดแวร์ต่าง ๆ สามารถตอบสนองต่อการประมวลผลนั้นได้ทันท่วงที และการประยุกต์ใช้ในปัจจุบันได้มีการทดลองการควบคุมการวิ่งของพาหนะโดยใช้การประมวลผลโดยคอมพิวเตอร์ในการตรวจตราดูแลทัศนะวิสัยต่าง ๆ ซึ่งอาจจะเกินความสามารถที่มนุษย์จะมีได้ เช่น การประมวลผลหรือการคาดหมายลักษณะของถนนด้านหน้ารถในความมืดหรือที่ที่มีทัศนะวิสัยที่เลวร้ายต่อการขับขี่ ซึ่งหลักการโดยทั่วไปต้องอาศัยกล้องที่มีประสิทธิภาพในการตรวจจับภาพที่ดีพอ และคอมพิวเตอร์ที่ใช้ในการประมวลผลต้องประมวลผลด้วยความเร็วที่ค่อนข้างสูง หรือการใช้คอมพิวเตอร์หลายตัวมาประมวลผลร่วมกัน เพื่อที่จะทำให้ใช้เวลาในการประมวลผลน้อยลง

เราจะประสบกับปัญหาหลายอย่างในการนำทางยานพาหนะบนถนนโดยอัตโนมัติ เราจึงได้ศึกษาและพัฒนาระบบการนำทางยานพาหนะโดยอัตโนมัติที่มีประโยชน์และมีราคาถูก ในการทดลองของเราจะใช้คอมพิวเตอร์ในการประมวลผลภาพ

ในการประมวลผลภาพจะมีรายละเอียดขั้นตอนดังต่อไปนี้

1. การสร้างตัวกำหนดตำแหน่งของเลนถนน (lane marker detector) เพื่อใช้ในการกำหนดขอบเขตของถนน โดยปราศจากสัญญาณรบกวน หรือวัตถุอื่น ๆ ในถนน เช่น รถยนต์คันอื่น หรือเงาต่าง ๆ เป็นต้น
2. การแบ่งส่วนย่อย ๆ ของภาพที่ได้จากขอบของถนน (ส่วนเทคนิคการจัดกลุ่ม (clustering technique) จะใช้ในตอนสุดท้ายของการประมวลผล)
3. ขั้นตอนการปรับปรุงในส่วนของความเร็วและความน่าเชื่อถือของขั้นตอนต่าง ๆ ในเบื้องต้น โดยใช้คาลมานฟิลเตอริ่ง (Kalman filtering)
4. ขั้นตอนการประมาณรูปแบบของถนน ประกอบไปด้วยพิกัดของจุดกึ่งกลาง ความโค้งของถนน จำนวนของเลนถนน และตำแหน่งของรถยนต์บนถนน

1.2 การหาขอบเขตของขอบถนนในเวลาจริง (Real Time detection of lane Boundaries)

ในที่นี้เราจะวิเคราะห์ส่วนที่ใช้ในการหาขอบเขตของขอบถนนในเวลาจริง การหาทิศทางของถนน และ ตำแหน่งของยานพาหนะ เพื่อที่จะนำไปใช้เป็นข้อมูลในการขับมอเตอร์ ในการหาข้อมูลในเวลาจริงเราจะสนใจเฉพาะสิ่งแวดล้อมของถนน และจะประมวลผลโดยใช้ภาพขาว-ดำ (gray level images)

ในระบบที่เราใช้งานจะมีสถานะของสิ่งแวดล้อมต่าง ๆ กันไป เช่น มีแสงสว่างจ้า มีแสงน้อย (มืดทึบ) หรือทัศนวิสัยไม่ดี เราจึงได้พัฒนาโปรแกรมโดยใช้อัลกอริทึมฟาสเอจดีเทคชั่น (Fast edge detection)

อัลกอริทึมที่ได้นี้จะผลการทดสอบที่ครอบคลุมสถานการณ์ต่าง ๆ ได้เป็นอย่างดี และเป็นวิธีพื้นฐานในการใช้ขับมอเตอร์ของรถทดสอบ

1.3 ตัวกำหนดตำแหน่งของเลนถนน (Lane marker detector)

จากข้างต้นเราจะพบว่าจะมีสิ่งแวดล้อมต่าง ๆ เข้ามาเกี่ยวข้องในภาพของเลนถนน เช่น ปริมาณของแสงขนาดต่าง ๆ สภาพอากาศที่ต่างกันไป และ สิ่งก่อสร้างต่าง ๆ เช่น สะพาน หรือ อุโมงค์ เราจึงต้องมีการพัฒนาระบบที่จะหาตำแหน่งของถนนได้แม้ว่าจะมีสภาพแวดล้อมต่างกันได้

การทดลองจะแสดงให้เห็นเมื่อนำภาพผ่านตัวกรองผ่านความถี่ต่ำ (low pass filter) จะได้เซตของจุดต่าง ๆ ที่จะขึ้นอยู่กับขอบของถนน ตัวกำหนดตำแหน่งของเลนถนนจะปรากฏในรูปของจุดสองจุดที่มีเกรเดียนสูงมาก คือทั้งสองจุดจะแทนการเปลี่ยนจากบริเวณมืดไปยังบริเวณสว่าง

บทที่ 2
การทำงานของสเต็ปเปอร์มอเตอร์

2.1 การทำงานของสเต็ปเปอร์มอเตอร์

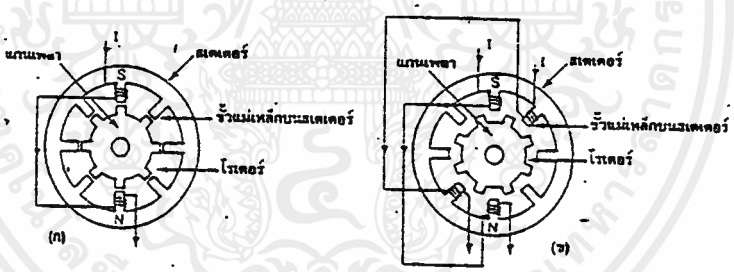
สเต็ปเปอร์มอเตอร์มีความแตกต่างจากมอเตอร์ทั่ว ๆ ไป โดยเมื่อป้อนกำลังไฟฟ้าให้กับมัน มันจะหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งแตกต่างจากมอเตอร์ทั่ว ๆ ไป ซึ่งจะหมุนทันที และตลอดเวลาเมื่อป้อนแรงดันไฟฟ้า สเต็ปเปอร์มอเตอร์สามารถกำหนดตำแหน่งของการหมุนด้วยตัวเลขได้อย่างละเอียดโดยการใช้คอมพิวเตอร์เป็นตัวกำหนดและจัดเก็บตัวเลขเหล่านั้นไว้

สเต็ปเปอร์มอเตอร์สามารถใช้งานในระบบเปิด (open loop system) นั่นก็คือมันสามารถทำงานได้โดยไม่ต้องการการป้อนกลับ (feedback) แต่ทุกวิธีที่ต้องการกำหนดตำแหน่งได้อย่างถูกต้องจำเป็นที่จะต้องการการป้อนกลับไปยังระบบให้รับรู้ และตัวบอกตำแหน่งว่าถูกต้องหรือเกิดการผิดพลาด

วิธีหนึ่งที่ใช้กันโดยทั่วไปกับสเต็ปเปอร์มอเตอร์ คือการใช้สวิทช์ติดตั้งไว้ที่ตำแหน่งที่ต้องการตรวจจับ (limit switch) เมื่อสเต็ปเปอร์มอเตอร์เริ่มหมุนและหมุนจนกระทั่งถึงตำแหน่งของสวิทช์ตรวจจับสัญญาณ ก็จะถูกป้อนกลับไปสู่ระบบและจะทำให้ทราบการทำงานของสเต็ปเปอร์มอเตอร์ได้ตลอดเวลาซึ่งโดยปกติในวงจรคอนโทรลเลอร์จะมีการกำหนดจุดอ้างอิง (reference point) ไว้ด้วย เพื่อให้เริ่มต้นทำงานและอ้างอิงตำแหน่งได้อย่างถูกต้อง ตัวอย่างเช่น ถ้าเริ่มจ่ายกำลังไฟฟ้าให้กับฟล็อปปี้ดิสก์ไดรฟ์ จะได้ยินมันก้าลังเคลื่อนที่เพื่อหาจุดอ้างอิงที่กำหนด หลังจากนั้นวงจรไดรฟ์คอนโทรลเลอร์จะเริ่มทำงานได้ โดยมันจะทราบถึงทุก ๆ สเต็ปที่กำลังขับเคลื่อนหัวอ่าน - เขียนไปยังแต่ละแทร็คบนดิสก์ เช่นเดียวกับมอเตอร์ทั่วไปการที่จะทำให้เกิดการหมุนของโรเตอร์ (rotor) ได้ต้องมีการกระทำของสนามแม่เหล็กที่เกิดขึ้นระหว่างโรเตอร์และสเตเตอร์ (stator) ซึ่งขึ้นอยู่กับการจัดวางขั้วแม่เหล็ก (pole) การหมุนทำได้ทั้งแบบต่อเนื่องและกลับทิศทางไปมา โดยกระบวนการทางไฟสลับหรือการจัดวางแปรงถ่าน และการจัดแยกคอมมิวเตอร์ และทำการสวิทชิ่งกำลังไฟฟ้าทำให้เกิดแรงดึงดูดของแม่เหล็ก (magnetic attraction) ที่ขั้วแม่เหล็กสร้างและหยุดสลับกัน ผลก็คือเกิดสนามแม่เหล็กหมุนขึ้นบนสเตเตอร์โดยการจ่ายกำลังไฟฟ้าทีละคู่ของขั้วแม่เหล็กในทิศทางตรงกันข้ามไปตลอดเวลา และเมื่อต้องการหยุดหมุนทำได้โดยหยุดการเกิดขั้วแม่เหล็กที่จุดหนึ่ง โดยหยุดการสวิชชิ่งในลำดับต่อไปเสีย การหมุนกลับทิศทางก็ทำได้เช่นเดียวกับที่

กล่าวมาแล้วเพียงแต่ทำการสวิชชิ่งกำลังไฟฟ้าให้เกิดสนามแม่เหล็กหมุนในทิศทางกลับกันหรือกลับลำดับการสวิทช์ของมัน

โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ประกอบขึ้นจากแผ่นเหล็กวงแหวนที่มีขี่ยื่นออกมาแต่ละขี่เหล่านั้นจะมีคอยล์พันสวมอยู่ ดังนั้นเมื่อป้อนกระแสผ่านคอยล์ทำให้เกิดสนามแม่เหล็กไฟฟ้า (electromagnetic) ขึ้นด้านตรงข้ามของแต่ละขั้วแม่เหล็กจะได้รับกระแสไฟฟ้าในขณะเดียวกัน แต่ว่าจะไหลวนในทิศทางตรงกันข้ามทำให้เกิดสนามแม่เหล็กไฟฟ้าในทิศตรงข้ามขึ้นดังแสดงในรูป ที่ 2.1 (ก) ดังนั้นถ้าเพิ่มจำนวนของขั้วแม่เหล็กให้มากขึ้นจะเพิ่มจำนวนของสเต็ปต่อวงรอบมากขึ้นตามไปด้วย อย่างไรก็ตามผู้ใช้สามารถเพิ่มจำนวนของสเต็ปได้อีกวิธีหนึ่งโดยไม่ต้องปรับเปลี่ยนโครงสร้างภายในโดยทำการจ่ายกำลังไฟฟ้าไปยังขั้วแม่เหล็ก 2 ขั้วที่อยู่ใกล้กันในเวลาเดียวกันซึ่งจะทำให้โรเตอร์หยุดหมุนอยู่ระหว่างกลางของ 2 ขั้วแม่เหล็กนั้นหรือเคลื่อนที่ไปครึ่งสเต็ปเท่านั้น และวิธีการนี้ยังช่วยให้เกิดแรงบิด (torque) มากขึ้นด้วยดังแสดงในรูปที่ 2.1 (ข)



รูปที่ 2.1 (ก) แสดงสเต็ปเปอร์มอเตอร์ที่มีการต่อวงจรขดลวดภายในเพื่อกระตุ้นให้เกิดขั้วแม่เหล็กขึ้นหนึ่งขั้วในทิศทางตรงกันข้าม ส่วนขดอื่น ๆ จะไม่ถูกกระตุ้นเลย

(ข) แสดงการต่อวงจรขดลวดแบบกระตุ้นให้เกิดขั้วแม่เหล็กพร้อมกัน 2 ขั้วที่อยู่โกล้กันทำให้โรเตอร์เคลื่อนที่มาหยุดอยู่ระหว่างขั้วแม่เหล็กทั้งสอง

สเต็ปปิ้งมอเตอร์(Stepping) เป็นมอเตอร์ชนิดที่หมุนทีละสเต็ป โดยที่ในการหมุนแต่ละสเต็ปมอเตอร์จะหมุนด้วยมุมคงที่ค่าหนึ่ง ซึ่งในการควบคุมการหมุนของ สเต็ปมอเตอร์นั้น จะอาศัยการควบคุมทางดิจิตอล โดยที่วงจรทางดิจิตอลนี้จะทำหน้าที่ในการจัดลำดับการกระตุ้นในแต่ละเฟส

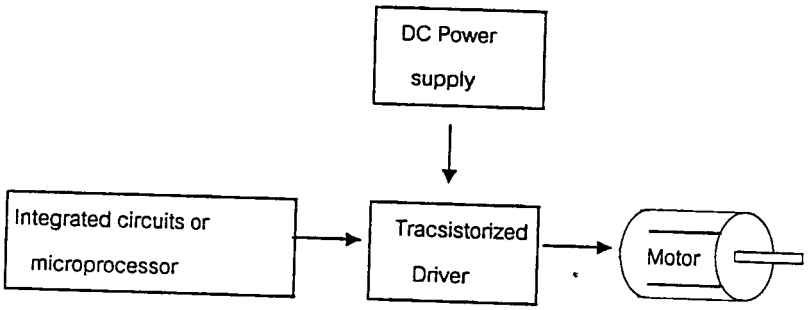
ของสเต็ปมอเตอร์ ซึ่งจะทำให้สามารถกำหนดทิศทางการหมุน ความเร็วในการหมุนและตำแหน่งที่ต้องการจะเลื่อนไปของสเต็ปมอเตอร์ได้อย่างถูกต้องและแม่นยำ

เนื่องจากวงจรทางดิจิตอลที่ใช้ในกาควบคุมการหมุนของสเต็ปมอเตอร์ สามารถกำหนดความเร็วในการหมุนและตำแหน่งที่ต้องการจะเลื่อนไปของสเต็ปมอเตอร์ได้อย่างถูกต้อง ดังนั้นจึงไม่จำเป็นตองมีการป้อนกลับ(Feedback Control) เพื่อควบคุมความเร็วและตำแหน่งในการหมุน เพราะฉะนั้นระบบควบคุมการหมุนของสเต็ปมอเตอร์เป็นแบบไม่มีการป้อนกลับ สำหรับการประยุกต์การใช้งานของสเต็ปมอเตอร์ มีดังนี้คือ

1) อุปกรณ์ประกอบการทำงานของคอมพิวเตอร์ เช่น
   - Serial Printer
   - X-Y Plotter
   - Floppy Disk Drives
2) ระบบ Numerical Control
   - X-Y Table And Index Table
   - Driller Machine
   - Automatic Drafting Machine
3) การประยุกต์ใช้งานอื่น
   - Facsimiles
   - Semiautomatic Wiring Unit
   - Application Inspec Science

2.2 ทฤษฎีและหลักการทำงานของสเต็ปมอเตอร์

สเต็ปมอเตอร์ เป็นอุปกรณ์จำพวกเชิงกลทางไฟฟ้า ที่มีอินพุทเป็นกลุ่มของไบนารีโวลท์เตจ และเอาท์พุทเป็นลักษณะของการเคลื่อนที่แบบเชิงมุม หรือหมุนไปทีละสเตป (แต่ละสเตปอยู่ในช่วง 0.1 ถึง 30 องศา ขึ้งอยู่กับโครงสร้างของสเต็ปมอเตอร์ ตามสัญญาณพัลซ์ที่ป้อนให้กับขดสเตเตอร์ (Stator) ซึ่งจะเกิดแรงผลักต่อโรเตอร์ (Rotor) หมุนไป แต่ลักษณะของสเต็ปมอเตอร์ จะมีขดของสเตเตอร์อยู่หลายขดซึ่งเรียกว่า "เฟส (Phase)" ฉะนั้นเมื่อป้อนสัญญาณที่เป็นพัลซ์ในลักษณะซีเค้วน (Sequence) ของเลขไบนารีโดยผ่านวงจรไดรเวอร์ (Driver) จะทำให้โรเตอร์หมุนได้อย่างต่อเนื่อง ดังบล็อคไดอะแกรมรูปที่ 2.2



รูปที่ 2.1 บล็อคไดอะแกรมแสดงการควบคุมสเตปปิ้งมอเตอร์

สเต็ปเปอร์มอเตอร์โดยทั่วไปมีจำนวนของขั้วแม่เหล็กหรือจำนวนสเต็ปต่อรอบเป็นจำนวนมากปกติอยู่ที่ประมาณ 100 - 400 สเต็ปต่อรอบ การมีจำนวนสเต็ปมาก ๆ นี้ไม่ได้เพิ่มจำนวนขั้วแม่เหล็กไฟฟ้าที่สเตเตอร์ แต่ทำได้โดยเพิ่มจำนวนซี่ขั้วแม่เหล็กที่โรเตอร์ จำนวนสเต็ปต่อรอบทั้งหมดจะได้จากการคูณจำนวนขั้วแม่เหล็กบนสเตเตอร์และจำนวนซี่ที่โรเตอร์ ดังเช่น ถ้ามีขั้วแม่เหล็ก 3 ขั้วบนสเตเตอร์ และ 8 ซี่ขั้วแม่เหล็กบนโรเตอร์ สเต็ปเปอร์มอเตอร์นี้จะทำงานที่ 24 สเต็ปต่อรอบ หรือหมุนไปเป็นมุม 15 องศาต่อสเต็ป

การใช้วงจรดิจิตอลคอนโทรลเลอร์กำหนดการจ่ายกำลังไฟฟ้าเข้าสู่ขดลวดบนสเตเตอร์แบบซีเควนเชียลทำให้สามารถควบคุมการเคลื่อนที่ทุกสเต็ปได้เช่นเดียวกับการควบคุมในวงจรดีซีเซอร์โว (DC servo) แต่การควบคุมด้วยดิจิตอลไม่จำเป็นต้องมีการป้อนกลับ การเคลื่อนที่ทุกสเต็ปได้จากการคำนวณจำนวนรอบหรือมุมในการหมุนที่ต้องการ แล้วจึงส่งข้อมูลที่ได้ไปควบคุมการหมุนของมอเตอร์ พิกัดในการทำงาน อาทิ ความเร็ว มุมในการเคลื่อนที่ ตำแหน่งของเพลา ถูกกำหนดจากข้อมูลที่ส่งมาควบคุม

2.3 ชนิดของสเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์แบ่งตามพื้นฐานได้เป็น 3 ชนิดคือ วาริเอเบิลรีลักแตนซ์ (variable reluctance : VR) ,เพอร์มาเนนต์แมกเน็ต (permanent magnet :PM) และแบบไฮบริดจ์ (hybrid)

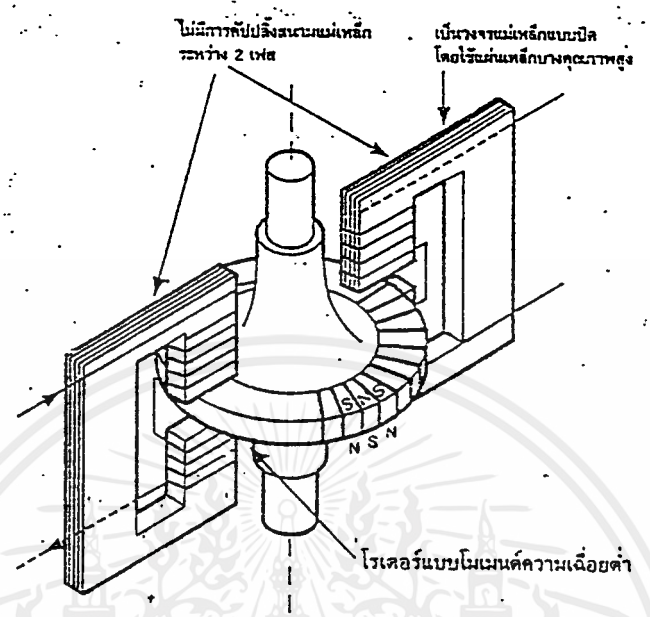
ชนิดวาริเอเบิลรีลักแตนซ์มีโครงสร้างของโรเตอร์แบบมัลติทูธ (multi - tooth) ทำจากเหล็กอ่อนเราจะทราบได้ว่าเป็นมอเตอร์ชนิดนี้โดยการทดสอบได้ง่ายมากคือใช้นิ้วหมุนเพลาของมอเตอร์และสังเกตมอเตอร์ชนิดนี้ที่โรเตอร์จะไม่เกิดปรากฏการณ์ทางแม่เหล็ก (magnetism) มันจึงหมุนได้ตลอดโดยไม่ติดขัด แตกต่างจากชนิด PM และชนิดไฮบริดจ์ซึ่งมีสนามแม่เหล็กที่โรเตอร์เมื่อหมุนจะรู้

สึกขัด ๆ เหมือนเป็นฟันเฟือง สเต็ปเปอร์มอเตอร์ชนิดนี้มีจุดด้อยในเรื่องของความถูกต้องของตำแหน่งและทำงานได้ไม่ดีนักเมื่อสเต็ปในการหมุนสูง

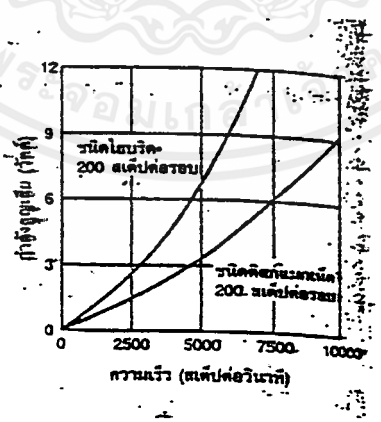
ชนิดเพอร์มาเนนต์แมกเน็ตมีโครงสร้างของโรเตอร์แบบเรียบไม่มีซี่ขั้วแม่เหล็กและบนโรเตอร์จะเป็นแบบแม่เหล็กถาวร การควบคุมทำได้โดยป้อนกระแสกระตุ้นที่ขดลวดบนสเตเตอร์ เช่น ถ้าเป็นสเตเตอร์แบบ 4 เฟส จะมีขั้วแม่เหล็กอยู่ 4 ขั้ว ซึ่งมีคอยล์พันอยู่แยกจากกัน ขั้วแม่เหล็กถาวรบนโรเตอร์จะถูกดึงดูดจากขั้วแม่เหล็กบนสเตเตอร์ เมื่อป้อนกระแสไฟฟ้าเข้าสู่ขดลวด และโรเตอร์จะอยู่คงที่ที่ขั้วแม่เหล็กบนสเตเตอร์นั้นถึงแม้ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป ทำให้เกิดเป็นแรงยึดหน่วงขึ้น สเต็ปเปอร์มอเตอร์ชนิดนี้มีข้อดีในเรื่องของความถูกต้องของตำแหน่ง และความเร็วมากขึ้นเมื่อเปรียบเทียบกับชนิดอื่น

ชนิดไฮบริดเป็นชนิดที่นิยมใช้งานกันมากที่สุด โดยเฉพาะนำมาใช้งานอย่างมากในอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ ชนิดไฮบริดมีโครงสร้างภายในซึ่งได้จากการรวมเอาโครงสร้างของสเตเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ และโครงสร้างของโรเตอร์จากชนิดเพอร์มาเนนต์แมกเน็ตมาประกอบเข้าด้วยกันจึงทำให้เป็นมอเตอร์ชนิดที่มีแรงยึดหน่วงสูง มีแรงบิดดึงและผลักได้ดีซึ่งมีความคงที่และทำงานได้ดีถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง

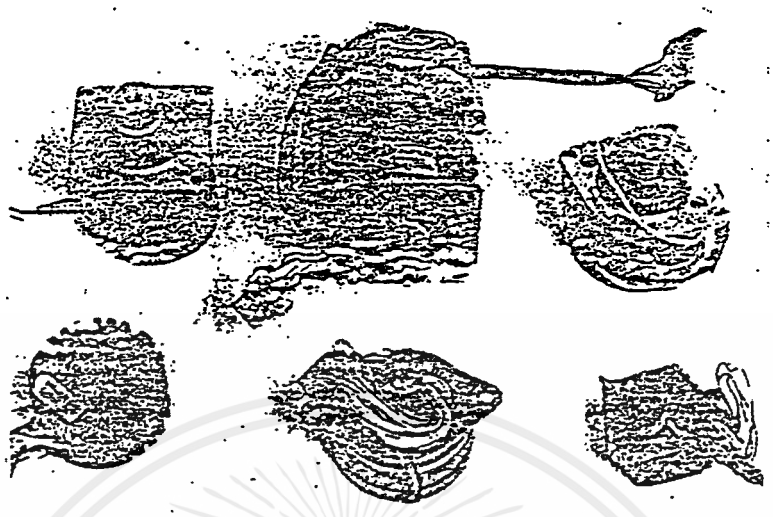
สเต็ปเปอร์มอเตอร์แบบใหม่อีกชนิดหนึ่งเป็นชนิดที่ปรับปรุงมาจากชนิดเพอมาเนนต์แมกเน็ต นั่นคือชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต (rare earth permanent magnet) ดังแสดงโครงสร้างภายในดังรูปที่ 2.3 หรือที่เรียกกันว่าชนิดดิสก์แมกเน็ตสเต็ปเปอร์มอเตอร์ (disk magnet steppers) โครงสร้างของโรเตอร์ของมอเตอร์ชนิดนี้มีลักษณะเป็นแผ่นซึ่งยึดกับเพลาของมอเตอร์ การทำงานของมอเตอร์ยังคงเป็นเช่นเดิม แต่ด้วยโครงสร้างแบบใหม่นี้ช่วยทำให้เกิดโมเมนต์ของความเฉื่อยต่ำมาก มีอัตราเร่งสูง มอเตอร์ชนิดนี้จึงจัดเป็นอีกชนิดหนึ่งที่มันก็มีประสิทธิภาพสูงอีกหลายด้าน เช่น แรงบิดดึง กำลังทางกลที่ได้ของมอเตอร์ ความถูกต้องของตำแหน่งสูงมาก และความเร็วในการเริ่มหมุนและหยุดหมุนสูง อีกทั้งยังมีความสูญเสียของกำลังงานต่ำดังแสดงในรูปที่ 2.4



รูปที่ 2.3 แสดงโครงสร้างภายในพื้นฐานของชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ตสเต็ปเปอร์มอเตอร์



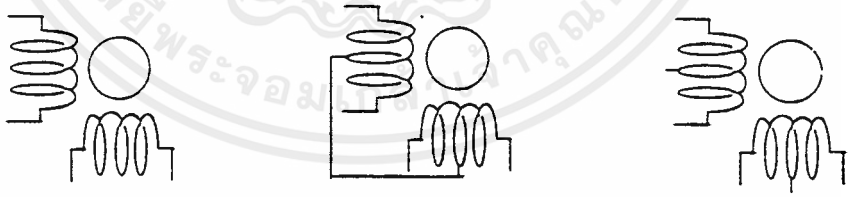
รูปที่ 2.4 แสดงกราฟข้อมูลการสูญเสียกำลังไฟฟ้า ความเร็วในการหมุนโดยเปรียบเทียบระหว่างสเต็ปเปอร์มอเตอร์ชนิดไฮบริดและชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต



รูปที่ 2.5 แสดงรูปร่างหน้าตาตัวจริงของสเต็ปเปอร์มอเตอร์ชนิดต่าง ๆ มอเตอร์เป็นชนิด - วาริเอเบิลรีลักแตนซ์ซึ่งมีตัวเข้ารหัสต่ออยู่ด้านท้ายของเพลา

2.4 การจำแนกชนิดของสเต็ปเปอร์มอเตอร์ด้วยการพันคอยล์

การพันขดลวดหรือคอยล์บนสเต็ปเปอร์มอเตอร์มีอยู่ 2 วิธีคือ แบบไบโพลาร์ (bipolar) และแบบยูนิโพลาร์ (unipolar) ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 แสดงการพันขดลวดบนสเตเตอร์ของสเต็ปเปอร์มอเตอร์ ด้านซ้ายเป็นแบบไบโพลาร์ และที่เหลือเป็นแบบยูนิโพลาร์ซึ่งมีทั้งแบบ 5 สาย และ 6 สาย 4 เฟส

สเต็ปเปอร์มอเตอร์แบบไบโพลาร์มีการพันขดลวดหนึ่งขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ ขั้วแม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้าและสามารถทำให้เกิดขั้วแม่

เหล็กในทิศทางตรงข้ามได้โดยการกลับทิศทางการไหลของกระแสไฟฟ้าซึ่งการกำหนดทิศการไหลและการกลับทิศทางของกระแสไฟฟ้าทำได้โดยการใช้วงจรสวิทชิ่งกลับขั้วไฟฟ้า

สำหรับยูนิโพลาร์จะมีการพันขดลวด 2 ขด บนแต่ละขั้วแม่เหล็กของสเตเตอร์ ซึ่งแต่ละขดจะทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามกัน การกลับขั้วแม่เหล็กเปลี่ยนไปมาทำได้โดยการสวิทชิ่งกระแสไฟฟ้าจากขดลวดขดหนึ่งไปยังอีกขดหนึ่งแทนเท่านั้น โดยปกติขดลวดทั้งสองจะมีการเชื่อมต่อกันหรือมีจุดร่วมเพื่อลดจำนวนของสายไฟที่ต่อจากมอเตอร์ วงจรจ่ายกำลังไฟฟ้าของมอเตอร์แบบยูนิโพลาร์ทำได้ง่ายกว่าชนิดไบโพลาร์ เพราะมันต้องการเพียงสวิทช์ธรรมดาในการเปิดและปิดกำลังไฟฟ้าให้กับขดลวดบนสเตเตอร์ในทิศทางที่ต้องการให้หมุนได้ทันที รูปที่ 2.7 แสดงวงจรจ่ายกำลังไฟฟ้าซึ่งใช้ทรานซิสเตอร์ทำหน้าที่เป็นตัวสวิทชิ่งให้กับสเต็ปเปอร์มอเตอร์ที่มีการพันขดลวดทั้ง 2 แบบ จะเห็นได้ว่าในแบบของยูนิโพลาร์ เป็นวงจรที่ง่ายและไม่มีความซับซ้อนเลย

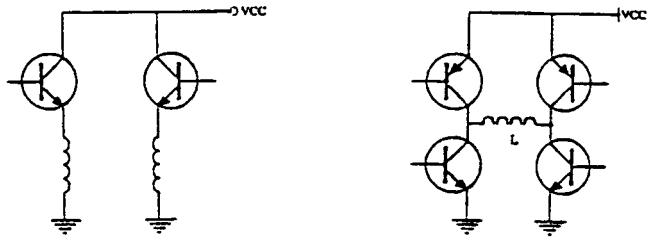
อย่างไรก็ตามการพันขดลวดแบบยูนิโพลาร์ก็มีจุดด้อยตรงที่การพันแบบนี้จะทำให้เกิดแรงบิดน้อยกว่าแบบไบโพลาร์เพราะจะมีเพียงครึ่งหนึ่งของขดลวดที่ถูกกระตุ้นให้ทำงานเท่านั้นในระยะเวลาหนึ่ง

การพิจารณาว่าสเต็ปเปอร์มอเตอร์ตัวใดมีการพันขดลวดแบบใดสังเกตได้ง่ายโดยถ้าเป็นแบบไบโพลาร์จะมีสายไฟต่อออกจากมอเตอร์เพียง 4 สาย และถ้าเป็นแบบยูนิโพลาร์จะมี 5 สาย หรือ 6 สาย หรือทราบได้โดยการอ่านจากป้าย (name plate) ที่ติดอยู่กับมอเตอร์ได้

2.5 การกระตุ้นและควบคุมการหมุนของสเต็ปเปอร์มอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้องแบ่งได้เป็น 3 รูปแบบ คือ แบบเวฟ (wave) แบบ 2 เฟส (two phase) และแบบครึ่งสเต็ป (half step) ทั้ง 3 แบบต่างก็มีข้อดีและข้อเสียแตกต่างกันออกไป

แบบเวฟเป็นการกระตุ้นรูปแบบที่ง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งและเรียงถัดกันไป ดังเช่นขดที่ 1,2,3,4,1 หรือ 1,4,3,2,1 ขึ้นอยู่กับทิศทางที่ต้องการให้หมุน ดังนั้นจึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่าง ๆ แสดงดังในตารางที่ 2.1



รูปที่ 2.7 แสดงวงจรจ่ายกำลังไฟฟ้าให้กับสเต็ปเปอร์ทั้ง 2 แบบ

(ก) สำหรับชนิดยูนิโพลาร์ซึ่งใช้ทรานซิสเตอร์สวิทช์เพียงตัวเดียวต่อ 1 คอยล์

(ข) สำหรับชนิดไบโพลาร์ซึ่งต้องใช้ทรานซิสเตอร์สวิทช์ 4 ตัวต่อ 1 คอยล์

ตารางที่ 2.1 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบเวฟ

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

แบบ 2 เฟสเป็นการกระตุ้นอีกรูปแบบหนึ่งซึ่งคล้ายกับแบบเวฟแต่การกระตุ้นแบบนี้จะทำโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกันและเรียงถัดกันไปเช่นเดียวกับแบบเวฟคือขดลวดที่ถูกกระตุ้น 12,23,34,41,12 หรือ 14,43,32,21,14 ขึ้นอยู่กับทิศทางการหมุน การเพิ่มจำนวนของขดลวดที่ถูกกระตุ้นนี้ทำให้เพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ได้ด้วยแรงดึงแบบเต็มแรง จาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกันและต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือ การกระตุ้นแบบนี้ต้องใช้แหล่งจ่ายกำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่าง ๆ แสดงดังในตารางที่ 2.2

ตารางที่ 2.2 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบ 2 เฟส

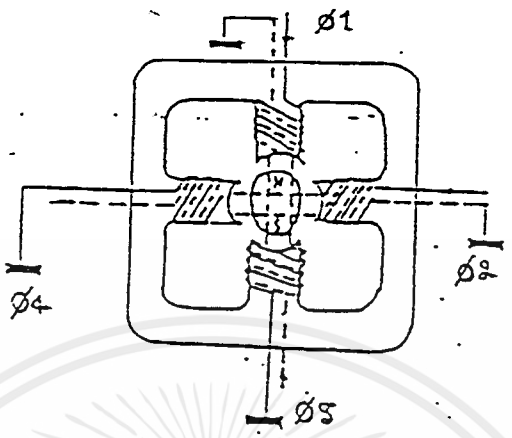
สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

แบบครึ่งสเต็ปเป็นรูปแบบที่เกิดจากการผสมผสานระหว่างการกระตุ้นแบบเวฟและแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ ขดลวดที่ถูกกระตุ้น 1,12,2,23,3,34,4,41,1 หรือในการหมุนอีกทิศทางหนึ่งจะได้เป็น 1,14,4,43,3,32,2,21,1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลงและแต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น

สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องมชเทียบเท่ากับแบบ 2 เฟสจึงจะเพียงพอ ขั้นตอนการทำงานต่าง ๆ ดังแสดงในตารางที่ 2.3

ตารางที่ 2.3 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบครึ่งสเต็ป

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



รูปที่ 2.8 แสดงโครงสร้างจำลองของ stepping motor ชนิด 4 เฟสและตำแหน่งของโรเตอร์

เฟสที่จ่ายกระแสไฟฟ้า	ϕ1	ϕ1ϕ2	ϕ2	ϕ2ϕ3	ϕ3	ϕ3ϕ4	ϕ4	ϕ4ϕ1
ตำแหน่งของโรเตอร์	↑	↗	→	↘	↓	↙	←	↖

ตารางที่ 2.4 แสดงมุมของโรเตอร์เทียบกับกระแสไฟฟ้าที่จ่ายแก่เฟสต่าง ๆ 8 ตำแหน่ง

จากลักษณะของมุมโรเตอร์หมุนกับกระแสไฟที่ป้อนแก่เฟสต่าง ๆ เราจะสามารถสั่งงานให้ stepping motor **หมุนได้ 3 อย่าง คือ**

1) **แบบจ่ายกระแสไฟให้เฟสเดียววนเวียนกันไปรียก** one - excitation **หรือ** half drive **คือ** 01,02,03,04 **การ** out - exitation **แบบนี้แรงบิดจะน้อย**
2) **แบบจ่ายกระแสไฟให้พร้อมกันทีเดียว** 2 **เฟส เรียก** two - exitation **หรือ** full step 0102,0203,0304,0401 **หมุนเวียนกันไปแบบนี้แรงบิดจะมาก**

3) แบบจ่ายกระแสไฟให้ทีละ 1 เฟสสลับกัน 2 เฟส เรียก one - two excitation หรือ half step เหมือนรูปแสดงมุมของโรเตอร์ แต่แบบนี้จำนวน step จะเพิ่มขึ้นเป็น 2 เท่าของ 2 แบบแรก แต่แรงบิดเฉลี่ยจะน้อย

จากการจ่ายกระแสให้เฟสทั้ง 3 อย่าง เราก็สามารถสั่งให้ stepping motor หมุนทวนเข็มได้ โดยมองการจ่ายกระแสให้เฟสย้อนกลับ เช่น ตามเข็มนาฬิกาแบบ 10 เป็น

04	03	02	01	
0	0	0	1	01
0	0	1	0	02
0	1	0	0	04
1	0	0	0	08

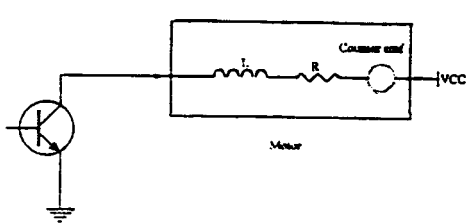
ทวนเข็มจะเป็น

08 04 02 01

เมื่อรู้ซีเควนซ์ของมันแล้ว ต่อไปเราก็ต้องมีวงจร driver ให้แก่ stepping motor วิธีง่ายที่สุดในการต่อวงจรซีเควนซ์เข้ากับวงจร driver คือ การต่อโดยตรง

2.6 ปัญหาเกี่ยวกับวงจร Driver

ขดลวดของ Stepping motor เป็น Inductive และมีค่าเปรียบเสมือนผลรวมของอินดัคแตนซ์อนุกรมกับความต้านทานดังรูป

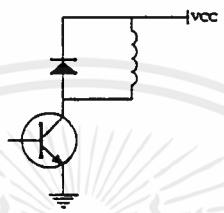


รูปที่ 2.9 วงจรสมบูรณ์ของขดลวด Stepping motor

2.6.1 **ซัพเพรสเซอร์**

เมื่อ transistor หยุดนำกระแส จะทำให้เกิด Voltage ค่าสูงจำนวนหนึ่ง เนื่องจากผลของการเปลี่ยนแปลงของกระแสในอินดัคแตนซ์ และ voltage นี้อาจจะเป็นอันตรายแก่ transistor ได้ วิธีป้องกันคือ

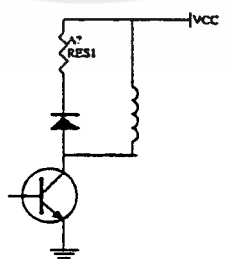
1) ไดโอดซัพเพรสเซอร์ ดังรูป



รูปที่ 2.10 แสดงการใช้ไดโอดซัพเพรสเซอร์

กระแสหมุนเวียน circulating current จะเริ่มไหลหลังจาก transistor หยุดกระแสและศักดา collector จะเท่ากับศักดาของแหล่งจ่าย ข้อเสียคือ กระแสจะหมุนเวียนอยู่นานและจะทำให้เกิดแรงบิดห้ามล้อ (breaking torque) พลังงานส่วนใหญ่สูญเสียในความต้านทานของขดลวด มีปัญหาเรื่องทำความเย็น

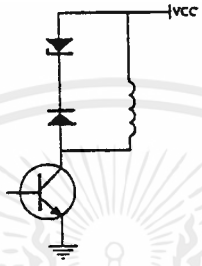
2) ไดโอดและรีจิสเตอร์ซัพเพรสเซอร์ ดังรูป



รูปที่ 2.11 แสดงการใช้ไดโอดและรีจิสเตอร์ซัพเพรสเซอร์

ถ้า Rs ยิ่งมากกระแสหมุนเวียนก็จะลดลงเร็วขึ้น แต่ศักดาของ collector จะมีค่าสูงขึ้น พลังงานส่วนใหญ่สูญเสียใน Rs

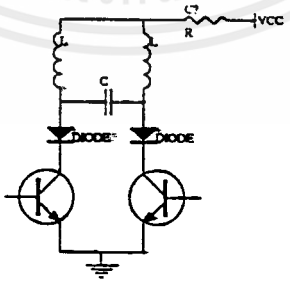
3) ซีเนอร์ไดโอดซัพเพรสเซอร์ ดังรูป



รูปที่ 2.12 แสดงการใช้ซีเนอร์ไดโอดซัพเพรสเซอร์

เมื่อ Transistor cutoff กระแสจะลดลงได้เร็วกว่า 2 แบบแรก และศักดาที่ collector จะเท่ากับศักดาของซีเนอร์บวกกับศักดาของแหล่งจ่าย ซึ่งเป็นอิสระต่อกระแส พลังงานส่วนใหญ่สูญเสียในซีเนอร์

4) คอนเดนเซอร์ซัพเพรสเซอร์ ดังรูป



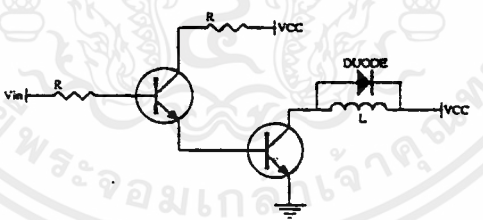
รูปที่ 2.13 แสดงการใช้คอนเดนเซอร์ซัพเพรสเซอร์

จะใส่ condenser 01 กับ 03 และ 02 กับ 04 เมื่อ transistor หยุดนำกระแส condenser จะต่อกับ transistor โดยผ่านทาง diode และจะดูดกลืนกระแสที่ค่อย ๆ ลดลงจากขดลวดของมอเตอร์เพื่อป้องกันทรานซิสเตอร์เสียและยังช่วยลดความร้อนที่เกิดขึ้นในขดลวดสเตเตอร์เนื่องจากการออสซิลเลท (oscillate) ของโรเตอร์ (rotor)

2.7 การแก้ไขการเพิ่มขึ้นของกระแส

เมื่อทรานซิสเตอร์นำกระแสเพื่อกระตุ้นเฟสแหล่งจ่ายไฟ (power supply) จะต้องขนะผลของอินดัคแตนซ์ของขดลวดก่อนที่กระแสจะเพิ่มขึ้นได้ถึงค่าที่กำหนด คือ อินดัคแตนซ์มีคุณสมบัติที่จะต้านการเพิ่มขึ้นของกระแสต่อไซเคิลจะมีค่ามากขึ้น และเป็นผลให้แรงบิด (torque) ลดลง และผลตอบสนองลดลง วิธีการลดเวลาการเพิ่มขึ้นของกระแส และปรับปรุงคุณลักษณะของแรงบิดที่มีความเร็วสูงให้ดีขึ้น โดยต่อความต้านทาน R อนุกรมกับขดลวดของมอเตอร์ซึ่งต่อกับแหล่งจ่ายไฟตรง จะต้องเลือกค่าความต้านทานที่เหมาะสม เพื่อให้ได้กระแสที่ต้องการไหลผ่านขดลวดที่มีสภาวะคงที่ Time constance ของวงจรจะลดลงเนื่องจากผลรวมของความต้านทานในขดลวดเพิ่มขึ้น

การไบอัส (Bias) วงจรภาคขยายกระแสไฟให้ได้กระแสแต่ละเฟสตามต้องการ



รูปที่ 2.14 แสดงการไบอัส (Bias) วงจรภาคขยายกระแสไฟให้ได้กระแสแต่ละเฟสตามต้องการ

สมมติให้ B2 คือค่าขยายกระแสต่ำสุดของ Q2 และ Ic2 คือกระแสขับสูงสุดต่อเฟส จะได้

$$R2 = \frac{(5 - Vbe2)}{Ib2} = \frac{(5 - Vbe2)*B}{Ic2}$$

สำหรับ R1 สมมติให้ B คือ ค่าขยายกระแสต่ำสุดของ Q1 และ Vin มีค่าประมาณ 4 โวลท์ สำหรับลอจิก (logic) "1" จะได้

$$R1 = \frac{(4 - Vbe1 - Vbe2)}{Ib1} = \frac{(4 - Vbe1 - Vbe2) * B1}{Ic1}$$

$$R1 = \frac{2.8 * B1}{Ib2}$$

บทที่ 3
ส่วนติดต่อระหว่างฮาร์ดแวร์ภายนอก กับคอมพิวเตอร์

3.1 CARD PORT 8255

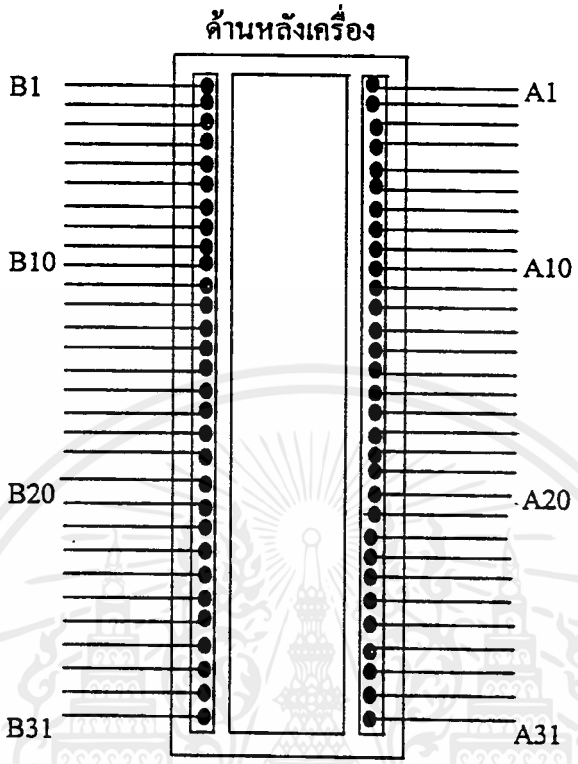
การควบคุมอุปกรณ์สนามไฟฟ้า โดยใช้เครื่องคอมพิวเตอร์นั้น จะต้องทำให้คอมพิวเตอร์ติดต่อกับอุปกรณ์ภายนอกให้ได้เสียก่อน แต่การที่จะทำเช่นนั้นได้ต้องผ่านอุปกรณ์ตัวหนึ่งเสียก่อน ซึ่งเรียกว่า พอร์ท (PORT) ซึ่งมีหลายลักษณะด้วยกัน แต่ในที่นี้จะใช้ IC 8255 ซึ่งเป็นพอร์ทขนานซึ่งถูกทำให้อยู่ในรูปของ การ์ด (CARD) ประกอบด้วยส่วนประกอบต่างๆรวมอยู่ใน CARDแผ่นเดียว ซึ่งจะกล่าวในรายละเอียดต่อไป

อุปกรณ์ที่ใช้บน CARD

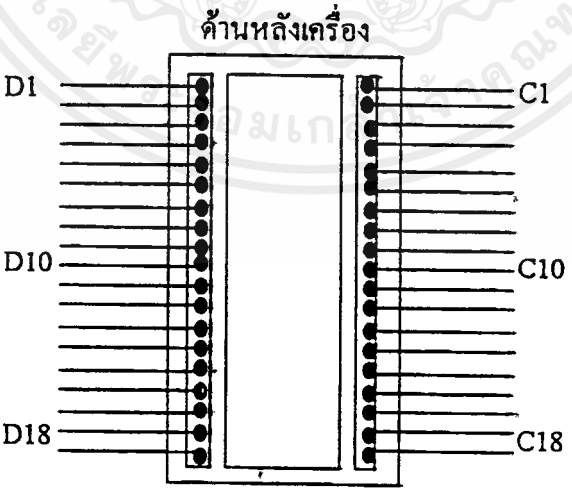
IC 8255	3	ตัว
IC 74LS139	1	ตัว
IC 74F245	1	ตัว
IC 74LS245	1	ตัว
IC 74LS688	1	ตัว
CONNECTOR 34 PIN	3	ตัว
DIP SW 8 จุด	1	ตัว
RESISTOR-PACK 10 K	11	ตัว
RESISTOR 330	1	ตัว
RESISTOR 4.7K	1	ตัว
CAPASESTOR 0.1 uF	1	ตัว
CAPASISTOR 33uF 16V	2	ตัว

ในCARD นี้จะมี 2 ส่วนใหญ่ด้วยกัน ซึ่งจะเป็นส่วนของวงจะ Decode และส่วนของ PORT 8255 ซึ่งจะอธิบายในรายละเอียดต่อไป

ที่เครื่องคอมพิวเตอร์นั้นจะมีช่องเสียบเรียกว่า SLOT PC โดยการเอาCARDมาเสียบที่ Slot นี้เพื่อเป็นการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์ภายนอก



การนับขาของสล็อตแบบ 62 ขา



การนับขาของสล็อตแบบ 36 ขา

รูปที่ 3.1 แสดงการนับของของสล็อตแบบ 62 และ 36 ขา

ตารางที่ 3.1 ชื่อของสัญญาณขาต่าง ๆ ของ SLOT

ขาอินพุต/เอาท์พุท	ชื่อสัญญาณ	อินพุต/เอาท์พุต
A1	- I/O CH CK	I
A2	SD7	I/O
A3	SD6	I/O
A4	SD5	I/O
A5	SD4	I/O
A6	SD3	I/O
A7	SD2	I/O
A8	SD1	I/O
A9	SD0	I/O
A10	- I/O CH RDY	I
A11	AEN	O
A12	SA19	I/O
A13	SA18	I/O
A14	SA17	I/O
A15	SA16	I/O
A16	SA15	I/O
A17	SA14	I/O
A18	SA13	I/O
A19	SA12	I/O
A20	SA11	I/O
A21	SA10	I/O
A22	SA9	I/O
A23	SA8	I/O
A24	SA7	I/O
A25	SA6	I/O
A26	SA5	I/O
A27	SA4	I/O

ขาอินพุต/เอาท์พุท	ชื่อสัญญาณ	อินพุต/เอาท์พุต
A28	SA3	I/O
A29	SA2	I/O
A31	SA1	I/O
A31	SA0	I/O
B1	GND	กราวน์
B2	RESET DRV	O
B3	+5 VDC	แหล่งจ่ายไฟเลี้ยง
B4	IRQ9	I
B5	-5 VDC	แหล่งจ่ายไฟเลี้ยง
B6	DRQ2	I
B7	-12 VDC	แหล่งจ่ายไฟเลี้ยง
B8	OWS	I
B9	+12 VDC	แหล่งจ่ายไฟเลี้ยง
B10	GND	กราวน์
B11	- SMEMW	O
B12	-SEMER	O
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	O
B16	DOQ3	I
B17	-DACK1	O
B18	DRQ1	I
B19	-REFRESH	I/O
B20	CLK	O
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I

ขาอินพุต/เอาท์พุท	ชื่อสัญญาณ	อินพุต/เอาท์พุต
B26	-DACK2	O
B27	T/C	O
B28	BALE	O
B29	+5 VDC	แหล่งจ่ายไฟเลี้ยง
B30	OSC	O
B31	GND	กราวน์
C1	SBHE	I/O
C2	LA23	I/O
C3	LA22	I/O
C4	LA21	I/O
C5	LA20	I/O
C6	LA19	I/O
C7	LA18	I/O
C8	LA17	I/O
C9	-MEMR	I/O
C10	-MEMW	I/O
C11	SDO8	I/O
C12	SD09	I/O
C13	SD10	I/O
C14	SD11	I/O
C15	SD12	I/O
C16	SD13	I/O
C17	SD14	I/O
C18	SD15	I/O
D1	- MEM CS16	I
D2	- I/O CS16	I
D3	IRQ10	I
D4	IRQ11	I
D5	IRQ12	I

ขาอินพุต/เอาท์พุท	ชื่อสัญญาณ	อินพุต/เอาท์พุต
D6	IRQ13	I
D7	IRQ14	I
D8	- DACK0	O
D9	DRQ0	I
D10	- DACK5	O
D11	DRQ5	I
D12	- DACK6	O
D13	DRQ6	I
D14	- DACK7	O
D15	DRQ7	I
D16	+ 5 VDC	แหล่งจ่ายไฟเลี้ยง
D17	- MASTER	I
D18	GND	กราวน์

สัญลักษณ์ในตาราง

I คือ INPUT

O คือ OUTPUT

I/O คือ INPUT/OUTPUT

สัญญาณที่ออกจาก SLOT PK ที่ใช้กับ CARD มีดังนี้

A0 - A11 เป็นแอดเดรสของระบบที่ใช้ติดต่อกับหน่วยความจำและอุปกรณ์ อินพุท/เอาท์พุท

D0 - D7 เป็นสัญญาณข้อมูลขนาด 8 บิท ที่ใช้ติดต่อกับหน่วยความจำของไมโครโปรเซสเซอร์

$\overline{\text{IORQ}}$ เป็นสัญญาณอ่าน อินพุท/เอาท์พุท เป็นสัญญาณที่ส่งมาจาก CPU จะแอกทีฟที่ " 0 "

$\overline{\text{IOW}}$ เป็นสัญญาณหเขียนข้อมูลลงบนอุปกรณ์ อินพุท/เอาท์พุทสัญญาณนี้ควบคุมจากไมโครโปรเซสเซอร์ เอกทีฟที่ " 0 "

RES	สัญญาณนี้ใช้สำหรับรีเซ็ทระบบในขณะเปิดเครื่องหรือขณะที่แหล่งจ่ายไฟเลี้ยงขาด
AEN	อีนาเบิลแอดเตรส (เป็นเอาท์พุท)

เมื่อรู้สัญญาณต่างๆที่ออกมาจากเครื่องคอมพิวเตอร์แล้ว ก็มาดูรายละเอียดของวงจรบน CARD บ้าง ซึ่งมี 2 ส่วนดังที่กล่าวมาแล้วในตอนต้นแต่ทั้งสองส่นจะทำงานร่วมกัน ส่วนของวงจร DECODE ซึ่งใน CARD นี้ประกอบด้วย IC 74LS688, 74LS139 และ DIP SW 3 PIN เพื่อที่สามารถปรับ SET DIP SW ในการตั้งตำแหน่งเบอร์พอร์ท PORT ของ CARD ได้โดยเพียงแค่ปรับที่ DIP SW เท่านั้น สิ่งที่ต้องระวังคือ ในการปรับ DIP SW นั้นจะต้องไม่ให้ตรงกับตำแหน่ง PORT ที่เครื่องคอมพิวเตอร์ใช้อยู่แล้ว ส่วน IC 74F245 ทำหน้าที่เป็น BUFFER 2 ทางคือ อยู่ในสถานะรับข้อมูลหรือส่งข้อมูลโดยการกำหนดที่เครื่องคอมพิวเตอร์ ส่วนเบอร์ 74LS245 ก็เป็นบัสบัฟเฟอร์แต่ถูกเซ็ทให้ข้อมูลจาก A ไป B เท่านั้น

3.2 การทำงานของวงจร

เริ่มต้นที่ SLOT PC โดยจากวงจรจะเห็นว่า D0-D7 จะต่อเข้าที่ขา A1-A8 ของ IC 74F245 และที่ ขา B1-B8 ของ IC 74F245 จะต่อเข้าที่ขา D0-D7 ของ PORT 8255 ทั้ง 3 ตัว โดยที่ขา DIR ของ IC 74LS245 จะต่อเข้ากับขา B0 ของ IC 74LS245 เหตุที่ต่อเช่นนี้เพราะขา IOR ของ SLOT PC ต่อกับขา A0 ของ IC 74LS245 ซึ่งจะเป็นตัวควบคุมว่าตอนนี้อยู่ในสภาวะรับหรือส่งข้อมูล ส่วนขา IOW ของ SLOT PC จะต่อเข้ากับขา A1 ของ IC 74LS245 ซึ่งขา B0,B1 ของ IC 74LS245 ก็จะไปต่อกับขา IOR , IOW ของพอร์ท 8255 ทุกตัวเช่นเดียวกัน เพื่อเป็นการบอกสถานะให้รู้ว่าจะรับหรือส่งข้อมูล ส่วนขา RESET A0,A1 ของ SLOT PC ก็จะต่อเข้าที่ขา A2,A3,A6 ของ IC 74LS245 โดยขา B7 จะต่อเข้าขา RESET ทุกตัวของพอร์ท8255 ส่วนขา B2,B3 จะต่อเข้าที่ขา A0,A1 ของ 8255 ทุกตัวเพื่อเป็นการเซ็ทว่าจะใช้งานพอร์ทไหน และให้เป็นอินพุทหรือเอาท์พุท โดยเซ็ทที่ A0,A1 ที่มาจาก SLOT PC จะเห็นว่าขา E ของ IC 74F245 จะมีไฟเลี้ยงตลอดทำให้ IC 74F245 ไม่ทำงานจนกว่า จะมีลอจิก "0" จากเอาท์พุทของ 74LS688 มาเท่านั้น ส่วน 74LS245 จะทำงานตลอดเพราะต่อขา E ลงกราวน์และที่ขา DIR ก็มีไฟเลี้ยงตลอดเวลา ซึ่งเป็นการเซ็ทให้ข้อมูลส่งจาก A ไป B เท่านั้น ส่วนขา A2,A3 ของ SLOT PC จะเข้าที่ขา A$,A5 ของ 74LS245 และออกที่ขา B4,B5 ไปเข้าที่ขา A,B ของ 74LS139 ซึ่งเป็นวงจร DECODE 2 LINE TO 4 LINE โดยขา A2,A3 ของ SLOT PC จะทำหน้าที่เลือกว่าจะให้ 8255 ตัวไหนทำงานซึ่งมี 3 ตัวด้วยกัน การที่จะส่งข้อมูลหรือรับข้อมูลนั้น เราต้องอ้างพอร์ทของ CARD ก่อน จากวงจรจะใช้ 74LS688 ซึ่งเป็นวงจร

COMPARATOR โดยขา A4-A11 ของ SLOT จะต่อเข้ากับ P0-P7 และที่ DIP SW ขา 1-8 จะต่อเข้ากับขา Q0-Q7 ของ 74LS688 ซึ่งถ้า ADDRESS ที่ส่งมาตรงกับที่ตั้ง DIP SW ไว้ ก็จะทำให้ขา P=Q แอกทีฟ "0" ซึ่งทำให้ขา E ของ 74F245 แอคทีฟทำให้รู้ว่า 8255 ตัวไหนทำงานที่ตำแหน่งพอร์ทเท่าไร และสามารถใช้พอร์ทไหนในการทำงานและเป็นการรับหรือส่งข้อมูล

ตัวอย่างเช่น

เราตั้ง DIP SW ไว้ที่ตำแหน่ง 300(00110000) เมื่อเราป้อน A0-A11 มาเป็น 300H คือ

A0-A1 เป็น 00 จะทำให้รู้ว่าเราใช้ PORT A

A2-A3 เป็น 00 จะทำให้ 74LS139 DECODE ออกที่ YO ที่ให้ 8255 ตัวที่ 1 ทำงาน

A4-A11 เป็น 00001100 ก็จะตรงกับที่ DIP SW ตั้งไว้ก็ทำให้วงจรทำงานสมบูรณ์

ในการอ้างพอร์ท (กรณี A4-A11 = 30XH)

8255 ตัวที่ 1
PORT A = 300H
PORT B = 301H
PORT C = 302H
CONTROL PORT1 = 303H

8255 ตัวที่ 2
PORT A = 304H
PORT B = 305H
PORT C = 306H
CONTROL PORT1 = 307H

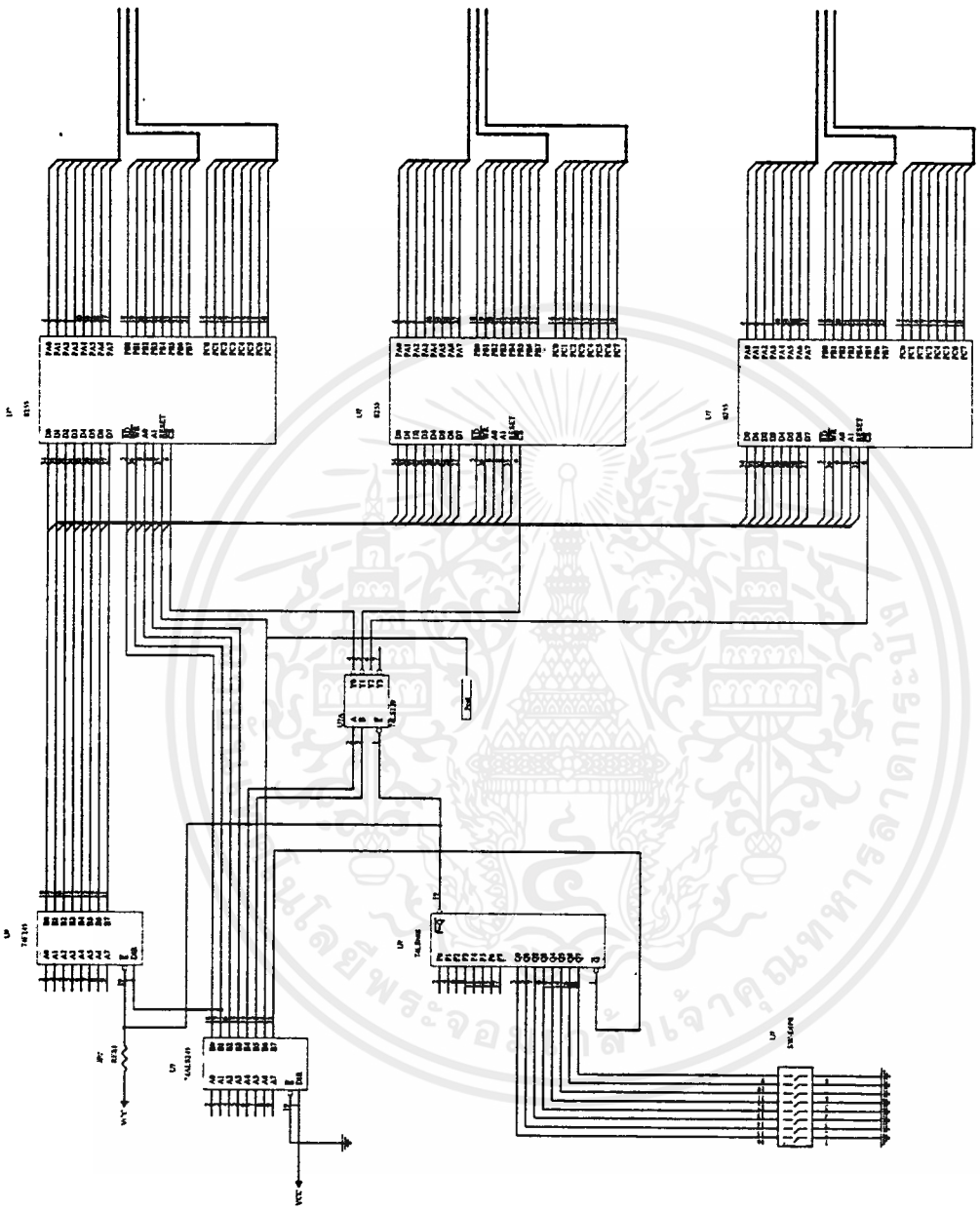
8255 ตัวที่ 3
PORT A = 308H
PORT B = 309H
PORT C = 30AH
CONTROL PORT1 = 30BH

หมายเลข PORT สามารถเปลี่ยนแปลงได้แต่ห้ามไม่ให้ตรงกับที่ใช้อยู่ในเครื่องคอมพิวเตอร์อยู่แล้ว ดูได้จากตารางที่ 3.2

ตารางที่ 3.2 แสดงหมายเลขพอร์ทของเครื่องคอมพิวเตอร์

หมายเลขพอร์ทรหัสฐานสิบหก	ชื่ออุปกรณ์
000-01F	DMA คอนโทรลเลอร์หมายเลข 1 8237A-5
020-03F	อินเตอร์รัพต์คอนโทรลเลอร์หมายเลข 1 8259A ตัวหลัก
040-05F	ไทเมอร์ 8254-2
060-06F	8042 คีบอร์ด
070-07F	นาฬิกาและ NMI และซีมอสแรม
080-09F	DMA เพจรีจิสเตอร์
0A0-0BF	อินเทอร์รัพติคอนโทรลเลอร์หมายเลข 2 8237A-5
0C0-0DF	DMA คอนโทรลเลอร์หมายเลข 2 8237A-5
0F0	เคลียร์โปรเซสเซอร์คณิตศาสตร์
0F1	รีเซ็ตโปรเซสเซอร์คณิตศาสตร์
0F8-0FF	โปรเซสเซอร์คณิตศาสตร์
1F0-1F8	ฮาร์ดดิสก์
200-207	เกมไอโอ
278-27F	พอร์ทเครื่องพิมพ์หมายเลข 2
2F8-2FF	พอร์ทอนุกรมหมายเลข 2
300-31F	โปรโตไทป์การ์ด
360-36F	สำรอง
378-37F	พอร์ทเครื่องพิมพ์หมายเลข 1
380-38F	SDLC, ไบซิงค์ 2
3A0-3AF	ไบซิงค์ 1
3B0-3BF	โมโนโครมและเครื่องพิมพ์
3C0-3CF	สำรอง

หมายเลขพอร์ทรหัสฐานสิบหก	ชื่ออุปกรณ์
3D0-3DF	จอภาพสี
3F0-3F7	ควบคุมดิสเก็ต
3F8-3FF	พอร์ทอนุกรมหมายเลข 1



รูปที่ 3.2 แสดงการเชื่อมต่ออุปกรณ์ของการ์ด 8255

บทที่ 4
ทฤษฎีของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยว (single chip microcontroller) คือไมโครคอมพิวเตอร์แบบที่มีขนาดเล็กโดยบรรจุไว้ในแผงวงจรรวม (integrated circuit) เพียงชิพเดี่ยวเหมาะสำหรับงานควบคุมอุปกรณ์อื่น ๆ แบบอัตโนมัติ เพราะผู้ใช้สามารถเขียนโปรแกรมควบคุมการทำงานได้ตามต้องการ ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล 51 หรือ MCS51 อันได้แก่ เบอร์ 8051 และ 8052 ซึ่งมีโครงสร้างและชุดคำสั่งแตกต่างกันเพียงเล็กน้อย

MCS-51 ผลิตโดยบริษัท Intel มีการทำงานเป็นแบบ 8 บิท หมายความว่าส่วนที่ทำหน้าที่ในการคำนวณ (Arithmetic Logic Unit , ALU) จะทำงานสูงสุดทีละ 8 บิท

MCS-51 มีข้อดีดังนี้

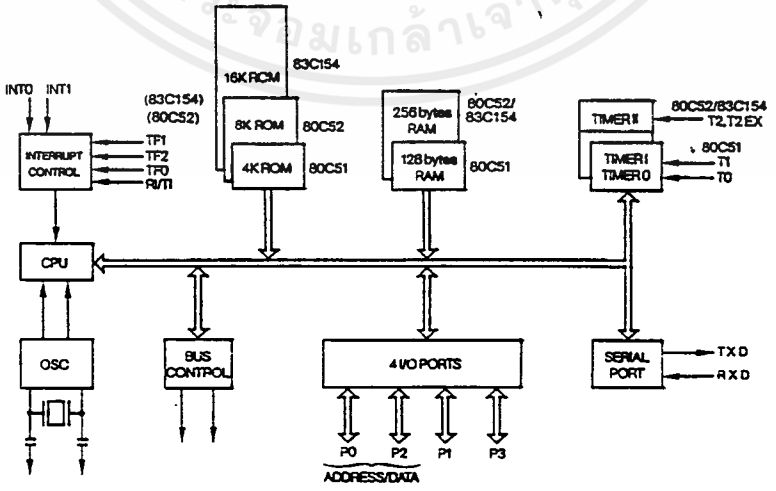
- สามารถนำเอาข้อมูลมา AND , OR หรือทำ Complement ทั้งแบบทีละ 8 บิท และ 1 บิท
- สามารถใช้กับหน่วยความจำสำหรับโปรแกรม (Program Memory) ซึ่งเป็นหน่วยความจำที่ใช้สำหรับเก็บชุดคำสั่งที่จะให้ MCS-51 ทำงาน ได้สูงสุด 64 กิโลไบท์ (Kilobyte) (64 x 1024 ไบท์) ทำให้เขียนโปรแกรมควบคุมการทำงานได้มาก
- สามารถต่อกับหน่วยความจำสำหรับข้อมูล (Data Memory) ซึ่งเป็นหน่วยความจำสำหรับเก็บข้อมูลในระหว่างการทำงานของโปรแกรมได้สูงสุด 64 กิโลไบท์
- ใน 8051 และ 8751 มีหน่วยความจำสำหรับโปรแกรมจำนวน 4 กิโลไบท์ (ใน 8052 และ 8752 มีหน่วยความจำสำหรับโปรแกรมจำนวน 8 กิโลไบท์) อยู่ภายในวงจรรวม ทำให้ไม่ต้องต่อหน่วยความจำสำหรับโปรแกรมอยู่ภายนอก ระบบรวมทั้งหมดจึงมีขนาดเล็กและสัญญาณรบกวนจากภายนอกจะทำให้ MCS-51 ทำงานผิดพลาดได้ยาก
- มีพอร์ทแบบขนาน (Parallel Port) สำหรับข้อมูลเข้าและออกจำนวน 32 บิท ที่ข้อมูลแต่ละบิทเป็นอิสระต่อกัน
- มีวงจร Timer / Counter ขนาด 16 บิท 2 ชุด (8052 มี 3 ชุด) ที่ทำงานในโหมดต่าง ๆ ได้ถึง 4 โหมด
- มี Universal Asynchronous Receiver Transmitter (UART) สำหรับรับ - ส่งข้อมูลอนุกรม (Serial) แบบ Full duplex ที่สามารถเลือกรูปแบบการรับ - ส่งข้อมูลได้ 4 แบบ

- มีแหล่งกำเนิดสัญญาณขอขัดจังหวะการทำงานของโปรแกรม (Interrupt Request Signal) 6 แหล่ง ซึ่งสามารถทำกระโดดไปทำงานตอบสนองการขัดจังหวะ (Interrupt Service Routine) ได้ต่าง ๆ กัน 5 ตำแหน่ง
- สามารถเลือกการทำงานให้อยู่โหมดของ Idle และ Power Down ซึ่งจะประหยัดการใช้กำลังไฟในการทำงาน

ซึ่งจากข้อดีดังกล่าว จึงทำให้ MCS-51 เป็นที่นิยมนำมาใช้ในการควบคุมระบบอัตโนมัติมาก คุณสมบัติดังกล่าวบรรจุไว้ในวงจรรวมเดียว (Single Chip) ขนาด 40 ขา ดังนั้นจึงสามารถออกแบบให้ระบบทั้งหมดมีขนาดเล็ก และการที่ทั้งหมดบรรจุอยู่ภายในวงจรรวมเดียวจึงทำให้การตรวจสอบหาข้อผิดพลาดในระบบง่ายไม่สลับซับซ้อน รวมทั้งลดปัญหาเรื่องการที่มีสัญญาณรบกวนในระบบจนทำให้การทำงานผิดพลาดไป แต่การที่จะนำเอา MCS-51 มาใช้งานได้จำเป็นที่จะต้องศึกษาและทำความเข้าใจถึงโครงสร้างและองค์ประกอบของ MCS-51 เสียก่อน แล้วถึงเขียนโปรแกรมเพื่อควบคุมการทำงานของ MCS-51 ให้เป็นไปตามต้องการ

4.1 โครงสร้างของ 8051

ภายใน 8051 จะประกอบขึ้นด้วย GATE ต่าง ๆ เช่น AND , OR , NOT ซึ่ง GATE เหล่านี้จะถูกนำเอามาออกแบบให้มีหน้าที่การทำงานต่าง ๆ เช่นวงจรถอดรหัสคำสั่ง (Instruction Decoder) , วงจรสร้างสัญญาณนาฬิกา (Clock Signal Generator) โครงสร้างภายในของ 8051 จะประกอบด้วยส่วนย่อย ๆ ดังไดอะแกรมในรูปที่ 4.1



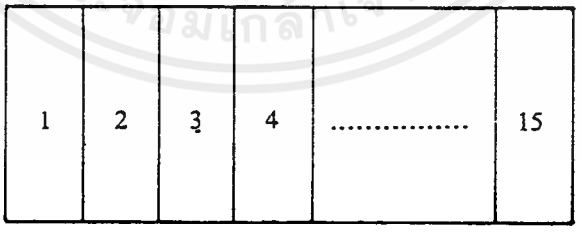
รูปที่ 4.1 ไดอะแกรมโครงสร้างของ 8051

ไดอะแกรมในรูปที่ 4.1 เป็นโครงสร้างใหญ่ ๆ ของ 8051 เนื่องจากลักษณะของ 8051 เป็นคอมพิวเตอร์จึงประกอบด้วย 3 ส่วนหลัก ๆ คือ

ส่วนที่ 1 คือ CPU (Central Processing Unit) หรือตัวประมวลผล ส่วนนี้จะมีวงจรที่ทำหน้าที่สร้างสัญญาณควบคุมในการติดต่อกับส่วนอื่น ๆ เรียกว่าวงจรศวบคุม (Control Unit) สัญญาณที่สร้างจากวงจรควบคุมได้แก่สัญญาณสำหรับการติดต่อกับหน่วยความจำ , อุปกรณ์รับข้อมูลเข้าหรือส่งข้อมูลออกจากตัว 8051 ซึ่งส่วนควบคุมการขัดจังหวะ (Interrupt Control) และส่วนควบคุมบัส (Bus Control) ก็เป็นส่วนหนึ่งของวงจรควบคุมด้วย การสร้างสัญญาณควบคุมจากส่วน CPU นี้จะทำการสร้างสัญญาณโดยการถอดรหัสจากคำสั่ง (Instruction) ตามที่มีการกำหนดไว้ และสัญญาณที่สร้างขึ้นมาจะอ้างอิงกับสัญญาณนาฬิกาที่สร้างจากวงจรออสซิลเลเตอร์เพื่อให้ทุก ๆ ส่วนในวงจรทำงานประสานกัน (Synchronize) อย่างถูกต้อง

ใน CPU นี้ยังประกอบด้วยส่วนย่อยอีกส่วนที่เรียกว่าส่วนประมวลผล (Arithmetic Logic Unit) ส่วนนี้จะทำหน้าที่ประมวลผลข้อมูลเช่น การบวก , ลบ , คูณ , หรือหารข้อมูลแล้วนำผลลัพธ์ไปเก็บไว้ในรีจิสเตอร์หรือหน่วยความจำที่ต้องการ

ส่วนที่ 2 คือ หน่วยความจำ (Memory) มีไว้สำหรับจดจำข้อมูล ถ้าจะให้เห็นภาพพจน์ของหน่วยความจำได้ดีก็คือ หน่วยความจำเปรียบเหมือนกล่องเก็บเอกสารจำนวนมากที่นำมาต่อเรียงกันไว้ แต่ละกล่องก็มีเอกสาร 1 แผ่น ดังในรูปที่ 4.2 มีกล่องเอกสารทั้งหมด 15 กล่อง



รูปที่ 4.2 ภาพเสมือนของหน่วยความจำ

ถ้าต้องการเอาเอกสารจากกล่องใด หรือเอาเอกสารไปเก็บที่กล่องใด จะต้องรู้หมายเลขของกล่องข้อมูลเสียก่อน ซึ่งถ้าเป็นหน่วยความจำแล้วหมายเลขของกล่องก็คือตำแหน่งของหน่วยความจำหรือแอดเดรส (Address) นั่นเอง การเอาข้อมูลไปเก็บในหน่วยความจำเรียกว่าการเขียน (

Write) ข้อมูล และการเอาข้อมูลออกจากหน่วยความจำจะเรียกว่าการอ่าน (Read) ข้อมูล ซึ่งแต่ละตำแหน่งของหน่วยความจำจะเก็บข้อมูลได้เพียงค่าเดียวเท่านั้น ในไมโครโปรเซสเซอร์ทั่วไปรวมทั้ง 8051 นั้นข้อมูลในแต่ละตำแหน่งของหน่วยความจำจะมีค่าได้เพียง 8 หลักของเลขฐาน 2 (8 บิท เท่ากับ 1 ไบท์) ดังนั้นแต่ละตำแหน่งของหน่วยความจำจะเก็บข้อมูลมีค่าได้ระหว่าง 0 ถึง 255 (00000000 ถึง 11111111 ในเลขฐาน 2) แต่จำนวนตำแหน่งที่จะเก็บข้อมูลได้ขึ้นกับไมโครโปรเซสเซอร์แต่ละเบอร์ การติดต่อกับหน่วยความจำจะต้องมีสัญญาณ 3 กลุ่มคือ

1. แอดเดรสหรือค่าตำแหน่งที่ต้องการติดต่อกับหน่วยความจำใน 8051 จะติดต่อกับหน่วยความจำประเภท Program Memory หรือ Data Memory ได้สูงสุดชนิดละ 65536 ตำแหน่ง ดังนั้นการอ้างอิงแต่ละตำแหน่งของหน่วยความจำจะต้องใช้เส้นแสดงตำแหน่งในเลขฐาน 2 ทั้งหมด 16 เส้น (2^{16} เท่ากับ 64 x 1024 = 65536)
2. ข้อมูลที่จะอ่านหรือเขียนกับหน่วยความจำที่ตำแหน่งในข้อ 1
3. สัญญาณควบคุมที่จะส่งไปยังหน่วยความจำ เพื่อบอกกับหน่วยความจำว่าต้องการอ่านหรือเขียนข้อมูล

สัญญาณเหล่านี้จะถูกวงจรควบคุมภายใน 8051 สร้างมาจากวงจรถอดรหัสของคำสั่งที่ 8051 อ่านจากหน่วยความจำ Program Memory เข้าไปทำงานนั่นเอง ในรูปที่ 4.1 หน่วยความจำได้แก่ 4K ROM และ 128 Byte RAM ซึ่งขนาดของหน่วยความจำนี้มีขนาดต่าง ๆ กันตามเบอร์ของไมโครคอนโทรลเลอร์ และจะอธิบายโดยละเอียดในหัวข้อ 4.2

ส่วนที่ 3 อุปกรณ์อินพุทและเอาท์พุท (Input / Output Device) เป็นส่วนที่จะใช้ส่งข้อมูลเข้าหรือออกจาก 8051 ทำให้ 8051 ติดต่อกับภายนอกได้ ดังในไดอะแกรมรูปที่ 4.1 อุปกรณ์อินพุทและเอาท์พุทได้แก่ 4 I / O Port , Time 0 , Time 1 , Serial Port การทำงานของแต่ละส่วนมีดังนี้

1. 4 I/O Port คำว่าพอร์ทหมายถึงจุดที่จะติดต่อกับส่วนที่อยู่ภายนอก 4 I/O Port ของ 8051 เป็นที่ใช้สำหรับรับ - ส่งข้อมูลซึ่งเป็นสัญญาณดิจิตอลเข้าหรือออกจากตัว MCS-51 พอร์ทมีทั้งหมด 4 พอร์ท โดยแต่ละพอร์ทจะรับ - ส่งข้อมูลได้ 8 บิท มีพอร์ท P0 , P1 , P2 และ P3 บางพอร์ทจะใช้ทำงานมากกว่า 1 อย่างก็ได้ เช่น พอร์ท P0 และ P2 จะใช้สำหรับการส่งค่าตำแหน่ง (Address) ของหน่วยความจำที่ต้องการติดต่อและพอร์ท P0 จะใช้รับส่งข้อมูลเมื่อติดต่อกับหน่วยความจำได้ด้วยแต่สิ่งเหล่านี้ไม่ได้เกิดที่เวลาเดียวกัน แต่จะใช้วิธีการทำงานตามลำดับ โดยควบคุมจากสัญญาณควบคุม (Control) ที่ถอดรหัสมาจากแต่ละคำสั่งที่ให้คอมพิวเตอร์ทำงานนั่นเอง และสัญญาณทั้งหมดจะอ้างอิงกับสัญญาณนาฬิกา

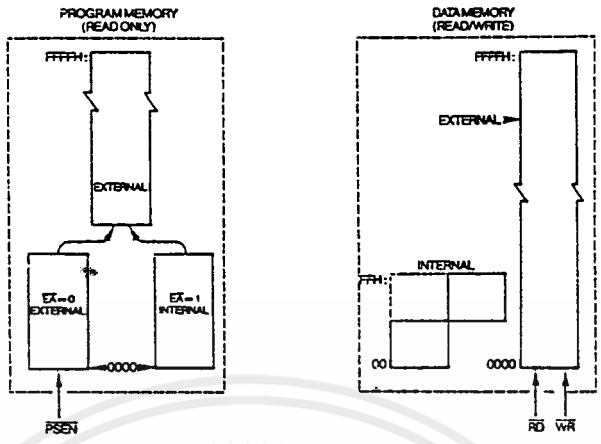
2. Timer 0 และ Timer 1 เป็นวงจรนับที่สามารถกำหนดให้ทำการนับจำนวนไซเคิลของสัญญาณที่ต่อจากภายนอก 8051 หรือจำนวนไซเคิลของสัญญาณนาฬิกาภายใน 8051 ก็ได้ค่าจากการนับจะถูกอ่านหรือตั้งค่าเริ่มต้นของการนับได้โดย CPU
3. Serial Port หรือพอร์ทอนุกรม CPU จะอ่านและเขียนข้อมูลกับ Serial Port เป็นแบบ 8 บิท แต่ข้อมูลจะถูกส่งออกจาก 8051 เรียงไปทีละบิทออกจากขา TXD และในการรับข้อมูลเข้าก็จะรับเข้ามาทีละบิททางขา RXD แล้วจัดเรียงใหม่เป็น 8 บิทเพื่อให้ CPU อ่านไปใช้งานต่อไป

8051 มีพอร์ทให้ใช้งานได้หลายแบบทำให้สะดวกแก่การนำไปใช้งานต่าง ๆ มากมาย การจะนำพอร์ทเหล่านี้ไปใช้งานได้จะต้องเขียนโปรแกรมขึ้นมาควบคุมที่จะได้กล่าวต่อไป

4.2 การจัดการหน่วยความจำของ 8051

หน่วยความจำของ 8051 แบ่งออกไว้เป็น 2 แบบตามลักษณะของการใช้งานคือ

1. Program Memory เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปรหัสภาษาเครื่อง (Machine Language) ซึ่งต้องการให้ 8051 ทำงาน เมื่อ 8051 ทำงานก็จะอ่านข้อมูลที่เก็บในหน่วยความจำประเภทนี้เข้าไปถอดรหัสแล้วสร้างสัญญาณควบคุมส่วนอื่น ๆ ตามการทำงานของแต่ละคำสั่งนั้น หน่วยความจำแบบนี้จะต้องเป็นแบบ Read Only Memory (ROM) และผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของหน่วยความจำเป็นรหัสภาษาเครื่องของ 8051 ตามลำดับการทำงานที่ต้องการ (หน่วยความจำแบบ ROM เป็นแบบ Non - volatile ซึ่งเมื่อปิดไฟแล้วข้อมูลก็ไม่มีการสูญหาย) การเขียนข้อมูลลงไปบน ROM จะต้องใช้เครื่องมือพิเศษ ในระหว่างการทำงานของ 8051 ผู้ใช้จะไม่สามารถใช้คำสั่งทำการเขียนข้อมูลลงในหน่วยความจำแบบนี้ได้ จำนวนตำแหน่งสูงสุดของหน่วยความจำแบบนี้ที่ 8051 จะใช้งานได้คือ 65536 ตำแหน่ง ค่าของตำแหน่ง (Address) จะเขียนเป็นเลขฐาน 16 ได้ตั้งแต่ 0000H ถึง FFFFH หน่วยความจำตำแหน่ง 0000H ถึง 0FFFH จำนวน 4 กิโลไบท์ นั้นผู้ใช้จะเลือกได้ว่าเป็นตำแหน่งของ ROM ที่อยู่ภายในหรือภายนอก 8051 (ไมโครคอนโทรลเลอร์เบอร์อื่น ๆ เช่น 8052 จะมีขนาดของ ROM ส่วนนี้ได้ถึง 8 กิโลไบท์ ตำแหน่ง 0000H ถึง 1FFFH) ถ้าต้องการให้ 8051 ทำงานตามคำสั่งที่เก็บไว้ใน ROM ภายใน 8051 ก็ให้ป้อนสัญญาณสภาวะลอจิก High (1) เข้าที่ขา EA ของ 8051 แต่ถ้าต้องการให้ทำงานในโปรแกรมที่เก็บไว้ใน ROM ภายนอก 8051 ก็ให้ต่อลอจิก Low (0) เข้าที่ขา EA ของ 8051 ส่วนหน่วยความจำที่ตำแหน่ง 1FFFH ถึง FFFFH จะต้องต่ออยู่ภายนอก 8051 เสมอดังแสดงในแผนภูมิหน่วยความจำ (Memory Map) ในรูปที่ 4.3



รูปที่ 4.3 แผนภูมิหน่วยความจำของ 8051

Internal Memory หมายถึงหน่วยความจำนั้นอยู่ภายใน 8051 ส่วน External Memory หมายถึง หน่วยความจำนั้นอยู่ภายนอก 8051

ไมโครคอนโทรลเลอร์เบอร์ 8031 , 8051 และ 8751 นั้น โดยโครงสร้างและรหัสคำสั่งจะเหมือนกันทุกประการแตกต่างกันที่

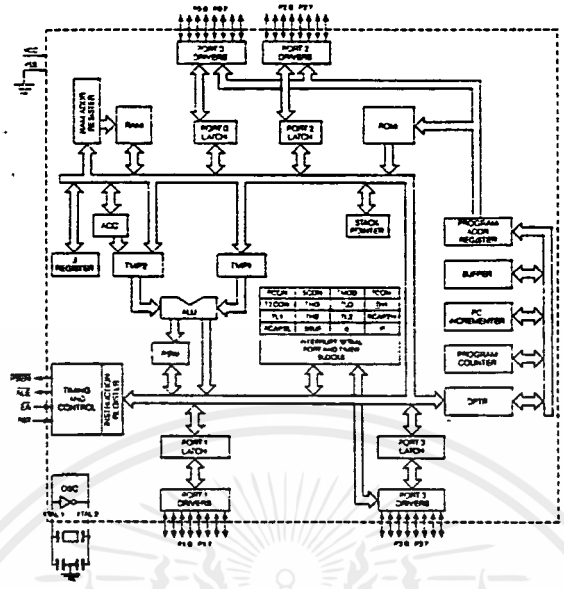
- 8031 จะไม่มี ROM ขนาด 4 กิโลไบท์อยู่ภายใน ผู้ใช้จะต้องเลือกการใช้งาน Program Memory อยู่ภายนอกวงจรรวมทั้งหมด 64 กิโลไบท์
- 8051 จะมี ROM ขนาด 4 กิโลไบท์ อยู่ภายใน ถ้าต้องการเก็บคำสั่งควบคุมการทำงานไว้ในหน่วยความจำส่วนนี้ จะต้องส่งโปรแกรมคำสั่งไปให้โรงงานผู้ผลิตทำการเขียนใส่ใน ROM ให้ตั้งแต่ในขั้นตอนของการผลิตวงจรรวม ผู้ใช้ไม่สามารถแก้ไขโปรแกรมได้เอง ถ้าจะนำมาใช้งานโดยเก็บโปรแกรมไว้ในหน่วยความจำช่วง 4 กิโลไบท์แรกอยู่ภายนอกก็สามารถทำได้ โดยการต่อ ROM ไว้ภายนอก แล้วต่อขา EA ของ 8051 ไว้กับสัญญาณที่มีสภาวะลอจิกเป็น 0
- 8751 จะมี หน่วยความจำขนาด 4 กิโลไบท์เป็นแบบ EPROM (Erasable Program Read Only Memory) อยู่ภายในวงจรรวมเอาไว้ ใช้เก็บโปรแกรมคำสั่งที่จะให้ 8751 ทำงาน ผู้ใช้สามารถเขียนคำสั่งลงไปใน EPROM ได้เองโดยใช้เครื่องมือที่เรียกว่าเครื่องโปรแกรม EPROM (EPROM Programmer) และผู้ใช้สามารถแก้ไขโปรแกรมที่อยู่ใน EPROM ได้โดยการล้างข้อมูลในทุกตำแหน่งของ EPROM ออกด้วยการฉายแสงอุลตร้าไวโอเลต (Ultraviolet) ผ่านกระจกใสบนวงจรรวมเข้าไปในวงจรภายใน

ตามเวลาที่กำหนดในคู่มือเฉพาะ (Data sheet) ของ 8751 จากนั้นก็ใช้เครื่องโปรแกรม EPROM เขียนโปรแกรมลงไปใหม่ 8751 นี้จะสะดวกมากสำหรับการพัฒนาโปรแกรม

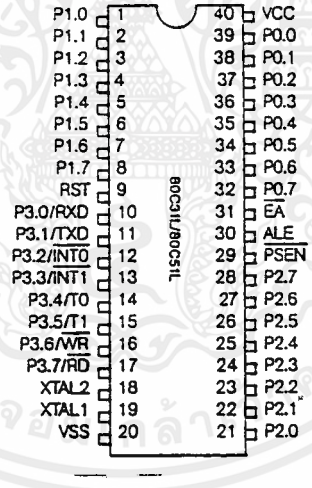
2. Data Memory เป็นหน่วยความจำที่ 8051 จะใช้สำหรับพัก , เก็บข้อมูล แล้วเรียกมาใช้ใหม่ในระหว่างการทำงานของ 8051 การอ่านหรือเขียนข้อมูลจากหน่วยความจำจะกระทำโดยคำสั่งที่เก็บไว้ใน Program Memory หน่วยความจำแบบนี้เป็นประเภท Random Access Memory (RAM) ถ้ามีไฟเลี้ยงอยู่ข้อมูลที่เก็บไว้จะไม่สูญหาย แต่ถ้าปิดเครื่องหรือไม่จ่ายไฟให้แก่ RAM แล้วข้อมูลใน RAM ก็จะสูญหายไป การสูญหายของข้อมูลไม่ได้หมายความว่าไม่มีอะไรอยู่เลยแต่เป็นการที่มีข้อมูลใหม่ซึ่งไม่ใช่ข้อมูลที่เก็บไว้เดิมเข้ามาอยู่แทนที่ เช่นเดิมเก็บข้อมูล 18H ไว้ที่ตำแหน่ง 1900H เมื่อปิดไฟแล้วเปิดใหม่ ข้อมูลที่ตำแหน่ง 1900H จะไม่ใช่ 18H อาจเป็นค่าอะไรก็ได้ ซึ่งเรียกการเกิดลักษณะแบบนี้ว่าข้อมูลสูญหายไป หน่วยความจำแบบ Data Memory ของ 8051 จะมีอยู่ 2 ชุด ชุดหนึ่งอยู่ภายใน 8051 จำนวน 128 ไบท์ที่ตำแหน่ง 00H ถึง 7FH (เบอร์ 8052 จะมี 256 ไบท์อยู่ที่ตำแหน่ง 00H ถึง FFH) และอีกชุดหนึ่งจะต้องต่ออยู่ภายนอกของวงจร รวม 8051 มีได้สูงสุด 65536 ไบท์ (64 กิโลไบท์) อยู่ที่ตำแหน่ง 0000H ถึง FFFFH ดังแสดงในรูปที่ 4.3 หน่วยความจำแบบ Data Memory ภายใน 8051 ที่ตำแหน่ง 80H ถึง FFH นั้นไม่ได้มีอยู่ทุกตำแหน่ง จะมีเฉพาะในบางตำแหน่งซึ่งเรียกหน่วยความจำบางตำแหน่งนี้ว่า Special Function Register (SFR) เพราะจะใช้หน่วยความจำเหล่านี้สำหรับงานพิเศษเท่านั้นเอง แต่ละตำแหน่งของหน่วยความจำแบบ SFR นี้อาจเป็น RAM หรือวงจรนับ (Counter) วงจรตั้งเวลา (Timer) ก็ได้เช่นเป็น Timer 0 , Timer 1 ดังนั้นใน 8051 จึงไม่ถือว่า SFR เป็น Data Memory ถ้าเป็น 8052 ซึ่งมี Data Memory ขนาด 256 ไบท์ จะใช้บางตำแหน่งของหน่วยความจำช่วงตำแหน่ง 80H ถึง FFH เป็น SFR ส่วนตำแหน่งอื่นที่เหลือก็เป็น RAM เหมือนกับหน่วยความจำช่วง 00H ถึง 7FH นั่นเอง

4.3 สถาปัตยกรรมของ 8051

ในหัวข้อที่ 4.2 ได้กล่าวถึงไดอะแกรมภายในของ 8051 อย่างกว้าง ๆ ซึ่งพอจะบอกได้โดยสังเขปว่าประกอบด้วยส่วนใหญ่ ๆ อะไรบ้าง ในรูปที่ 4.4 เป็นสถาปัตยกรรมภายในของ 8051 ซึ่งจะอธิบายถึงส่วนย่อย ๆ ของภายใน 8051 เพียงชิพเดียว และสัญญาณจากภายในจะต่อออกสู่ภายนอกทางขา (Pin) ของ 8051 ที่มีอยู่ 40 ขา ดังรูปที่ 4.5



รูปที่ 4.4 สถาปัตยกรรมภายในของ 8051



รูปที่ 4.5 ไดอะแกรมขาของ 8051 แบบ DIP

8051 ไมโครคอนโทรลเลอร์ที่บรรจุอยู่ในวงจรรวมแบบ Dual Inline Package (DIP) ซึ่งแต่ละข้างของ 8051 มีขาอยู่ข้างละ 20 ขารวมทั้งหมด 40 ขานั้นจะใช้งานต่าง ๆ กันดังนี้คือ

Vcc

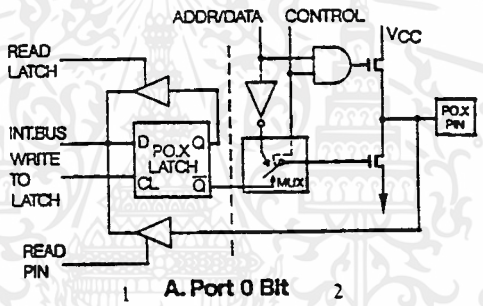
ขา 40 เป็นขาที่ต้องป้อนไฟเลี้ยง +5 โวลท์เข้าไปเพื่อให้วงจรรวมทำงานได้ ระดับโวลเตจของลอจิก 0 และ 1 ของ 8051 จึงต่อเข้ากับอุปกรณ์ลอจิกแบบ TTL ได้โดยตรง

Vss

ขา 20 เป็นขาที่ต้องต่อกับกราวด์ (Ground) ของแหล่งจ่ายไฟ การต่ออุปกรณ์ทั้งหมดจะต้องมีกราวด์ของอุปกรณ์ต่อเข้าด้วยกัน

Port 0

เป็นพอร์ทขนานขนาด 8 บิท อยู่ที่ขา 39 ถึง 32 เริ่มจากบิท 0 ถึง บิท 7 ตามลำดับดังในรูปที่ 4.5 แต่ละขาจะเขียนว่า P0.0 , P0.1 , ..., P0.7 นั้น P0.7 หมายถึงบิท 7 ของพอร์ท 0 ซึ่งเป็นบิทที่มีนัยสำคัญสูงสุด (Most Significant) และ P0.0 คือบิท 0 ของพอร์ท 0 เป็นบิทที่มีนัยสำคัญต่ำสุด (Least Significant) พอร์ท 0 นี้ใช้ได้ทั้งการรับ.- ส่งตำแหน่งและข้อมูลกับหน่วยความจำหรือใช้เป็นพอร์ทรับ - ส่งข้อมูลก็ได้ ข้อมูลที่ส่งออกทางพอร์ท 0 จะถูก Latch ไว้ที่ขาของพอร์ท โครงสร้างแต่ละบิทของพอร์ท 0 เป็นแบบ Open Drain Bidirectional ดังรูปที่ 4.6



รูปที่ 4.6 โครงสร้างของพอร์ท 0

ในรูปที่ 4.6 เมื่อเปรียบเทียบกับรูปที่ 4.4 ส่วนที่ 1 ของรูปที่ 4.6 ก็คือ Port 0 Latch ในรูปที่ 4.4 และส่วนที่ 2 ของรูปที่ 4.6 ก็คือ Port 0 Driver ของรูปที่ 4.4 นั่นเอง

จากโครงสร้างในรูปที่ 4.6 เมื่อมีคำสั่งการเขียนข้อมูลมายังพอร์ท 0 ข้อมูลจาก Internal Data Bus จะถูก Latch ไว้ที่ D-FF โดยสัญญาณ " Write to Latch " ที่ถูกสร้างมาจากส่วน Timing and Control และในการอ่านข้อมูลจากพอร์ท 0 จะอ่านได้ 2 แบบคือการอ่านข้อมูลที่ส่งไปเก็บไว้ที่พอร์ทก็จะมีสัญญาณ Read Latch มาเพื่ออ่านข้อมูลจาก D-FF กลับเข้าไปยัง Internal Data Bus การอ่านข้อมูลอีกแบบก็คือการอ่านสภาวะของสัญญาณที่เข้ามาทางพอร์ท 0 ก็จะมีสัญญาณ Read Pin มาควบคุมการอ่าน พอร์ท 0 จะใช้งานหลายอย่างดังนี้

1. ใช้สำหรับส่งค่าตำแหน่งหน่วยความจำภายนอกที่ต้องการติดต่อด้วย ตำแหน่งหน่วยความจำสูงสุดที่จะติดต่อได้ก็คือ 64 Kbyte จึงมีค่าตำแหน่งหน่วยความจำ 16 บิทของเลขฐาน

สอง ค่าตำแหน่งหน่วยความจำ 8 บิทล่างจะถูกส่งออกไปทางพอร์ท 0 และ 8 บิทบนจะส่งออกไปทางพอร์ท 2

2. ใช้รับ - ส่งข้อมูลกับ Data Memory หรือใช้รับข้อมูลจาก Program Memory

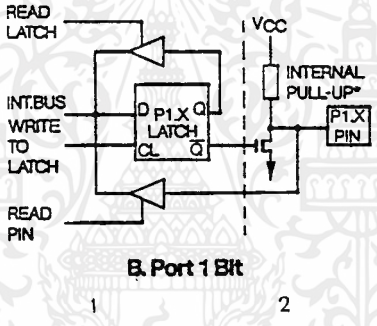
3. ใช้รับ - ส่งข้อมูลผ่านทางพอร์ทโดยตรง ในกรณีที่ไม่มีการใช้หน่วยความจำของ Program Memory หรือ Data Memory ภายนอก

วงจรภายในส่วน Timing and Control จะเป็นตัวสร้างสัญญาณมาควบคุมวงจรในรูปที่ 4.6 เพื่อให้การทำงานแต่ละอย่างข้างต้น เมื่อแต่ละบิทของพอร์ท 0 ทำงานตามข้อ 1 และ 2 ข้างต้น วงจร Timing and Control จะทำให้สภาวะลอจิกของขา Control เป็น 1 ซึ่งทำให้สวิทช์ MUX อยู่ในตำแหน่งข้างบน เมื่อพอร์ท 0 จะส่งข้อมูลซึ่งเป็นค่าตำแหน่งหน่วยความจำหรือข้อมูลที่จะเขียนออกไปยังหน่วยความจำภายนอกก็จะส่งค่าดังกล่าวมายัง ADDR / DATA ถ้าข้อมูลที่ส่งมาเป็น 1 จะทำให้สัญญาณออกจาก AND GATE เป็น 1 และสัญญาณที่ออกจาก Inverter เป็น 0 ดังนั้น FET ตัวบน ON (สภาวะ ON ของ FET คือความต้านทานระหว่างขา D กับ S มีค่าต่ำมากเสมือนกับเป็นวงจรปิด) ส่วน FET ตัวล่าง OFF (สภาวะ OFF ของ FET คือความต้านทานระหว่างขา D กับ S มีค่าสูงมากเสมือนกับเป็นวงจรเปิด) สภาวะลอจิกที่ขา P0.X PIN จะเป็น 1 แต่ถ้าข้อมูลที่ส่งออกมายัง ADDR / DATA เป็น 0 ก็จะทำให้สัญญาณจาก AND GATE เป็น 0 และสัญญาณที่ออกจาก Inverter เป็น 1 ดังนั้น FET ตัวบนจะ OFF ส่วน FET ตัวล่างจะ ON ทำให้สภาวะลอจิกที่ขา P0.X PIN เป็น 0 เมื่อ 8051 ต้องการใช้พอร์ท 0 สำหรับการอ่านข้อมูลจากหน่วยความจำภายนอก หรือใช้ทำงานในข้อ 3 ข้างบน ก็จะทำได้โดย วงจร Timing and Control ทำให้สภาวะลอจิกของสัญญาณ Control ในรูปเป็น 0 ทำให้เอาท์พุทจาก AND GATE เป็น 0 FET ตัวบนจะ OFF และสวิตช์ MUX จะอยู่ในตำแหน่งข้างล่าง ดังนั้น FET ตัวล่างจะ ON หรือ OFF ก็แล้วแต่ข้อมูลที่ขา Q ของ D-FF เมื่อมีการเขียนข้อมูลจาก Internal Data Bus มายัง D-FF ก็จะมีสัญญาณ Write to Latch มายัง D-FF ด้วย ถ้าข้อมูลที่เขียนมาเป็น 1 ก็จะทำให้ขา Q มีสภาวะลอจิกเป็น 0 ทำให้ FET ตัวล่าง OFF ดังนั้นขา P0.X จะอยู่ในสภาวะอิมพีแดนซ์สูง (High Impedancd) เพราะ FET ทั้ง 2 ตัว OFF แต่ถ้าข้อมูลที่เขียนมายัง D-FF เป็น 0 จะทำให้ FET ตัวล่าง ON แต่ตัวบน OFF ทำให้สภาวะลอจิกที่ขา P0.X เป็น 1 ดังนั้น PORT 0 เมื่อให้ทำงานเป็น พอร์ทส่งข้อมูล (ไม่ใช่ส่งตำแหน่งหน่วยความจำ) จะไม่สามารถแสดงสภาวะลอจิก 1 ได้จึงต้องต่อตัวต้านทาน Pull Up ไว้ภายนอก ระหว่างขา P0.X กับไฟเลี้ยงวงจร ถ้าจะใช้พอร์ท 0 สำหรับรับข้อมูลเข้าจะต้องเขียน 1 มาเก็บไว้ยัง D-FF เสียก่อนเพื่อให้ขา P0.X อยู่ในสภาวะ High Impedance แล้วจึงใช้คำสั่งอ่านสภาวะลอจิกเข้าไปยัง Internal Data Bus ต่อไป โดยคำสั่งอ่านสภาวะลอจิกทางพอร์ท 0 ก็จะทำให้วงจร Timing and Control สร้าง

สัญญาณ Read Pin สำหรับการอ่านสภาวะลอจิกข้างต้น ถ้าไม่เขียน 1 มาเก็บไว้ยัง D-FF ก่อนที่จะอ่านข้อมูลแล้วอาจมีข้อมูลค้างอยู่ที่ D-FF ทำให้ Q เป็น 0 และ Q เป็น 1 ซึ่งทำให้ FET ตัวล่าง ON สัญญาณที่ต่อเข้ามาที่ขา P0.X ไม่ว่าจะมีสภาวะลอจิกใดจะถูกดึงลงกราวด์ ดังนั้นเมื่ออ่านข้อมูลเข้าไปก็จะพบว่าเป็น 0 เสมอ ในการอ่านข้อมูลจากหน่วยความจำภายนอกนั้นวงจร Timing and Control ก็จะเขียนข้อมูลมายัง D-FF ให้เป็น 1 และสร้างสัญญาณ Control ให้มีลอจิกเป็น 0 ก่อนจะอ่านข้อมูลเข้าไปด้วย

Port 1

เป็นพอร์ทขนานขนาด 8 บิท ในรูปที่ 4.5 คือขา P1.0 ถึง P1.7 (ขา 1-8) P1.0 หมายถึงบิท 0 ของพอร์ท 1 ซึ่งเป็นบิท Least Significant Bit และบิท P1.7 หมายถึงบิทที่ 7 ของพอร์ท 1 ซึ่งเป็นบิท Most significant bit โครงสร้างของพอร์ท 1 แต่ละบิทมีดังรูปที่ 4.7

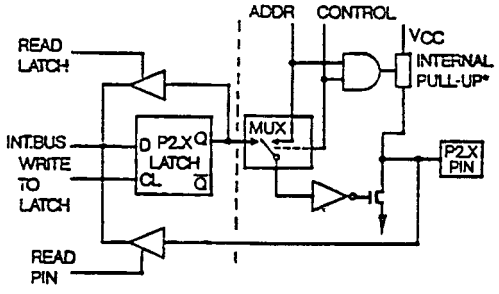


รูปที่ 4.7 โครงสร้างของพอร์ท 1

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-ส่งข้อมูลที่ส่งออกมาทางพอร์ท 1 จะถูก Latch ไว้แล้วส่งออกไปทางแต่ละขา ก่อนที่จะอ่านข้อมูลเข้าไปทางพอร์ท 1 จะต้องเขียน 1 ไปยังทุกบิทของพอร์ท 1 เสียก่อนเพื่อให้ FET อยู่ในสภาวะ OFF ก่อน มิฉะนั้นแล้วถ้ามีข้อมูล 0 ส่งออกมาค้างอยู่ที่ D-FF จะทำให้ FET อยู่ในสภาวะ ON ดังนั้นถ้าสัญญาณภายนอกส่งเข้ามาที่ขานี้ก็จะถูกลัดวงจรลงกราวด์ โดยไม่สนใจว่าสภาวะลอจิกของสัญญาณที่เข้ามาจะเป็นอะไร ข้อมูลที่อ่านเข้าไปจึงจะเป็น 0 เสมอ

Port 2

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



รูปที่ 4.8 โครงสร้างของพอร์ท 2

ลักษณะโครงสร้างจะเหมือนกับ Port 0 แตกต่างกันที่ใน Port 2 นั้นภาค Driver จะใช้งานเพียง 2 ลักษณะคือ

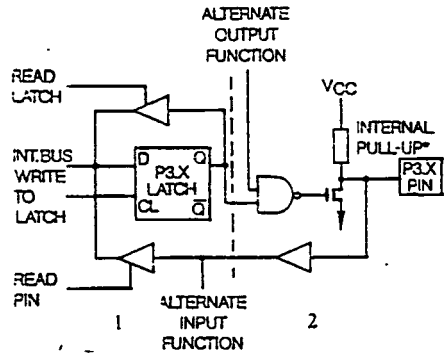
1. ใช้ส่งค่าตำแหน่งหน่วยความจำภายนอกที่ต้องการติดต่อ ค่าตำแหน่งนี้เป็น 8 บิท บนของค่าตำแหน่ง

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 ได้ ถ้า FET อยู่ในสภาวะ OFF บางครั้งเรียกว่า " Quasi - bidirectional " เมื่อใช้เป็นพอร์ทอินพุทก็สามารถทำได้โดยการต่อสัญญาณภายนอกเข้ามาโดยตรง ถ้าสัญญาณภายนอกเป็น 0 ก็จะมีกระแสไหลออกจากพอร์ท (Source Current) ในการที่จะใช้พอร์ทนี้เป็นพอร์ทรับข้อมูลเข้า จะต้องเขียน 1 ไปยังแต่ละบิทของพอร์ทเสียก่อน ดังได้อธิบายในเรื่อง Port 0 และ Port 1

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 เป็นส่วน Latch ข้อมูลที่เขียนมายังพอร์ท 3 ทาง Internal Bus เหมือนกับพอร์ทอื่น ๆ และพอร์ท 3 จะมี Internal pull-up อยู่ทุกบิท แต่พอร์ท 3 นี้แต่ละบิทจะใช้ในการทำงานอื่นได้โดยใช้คำสั่งควบคุมการทำงาน ในส่วนที่ 2 จะมีสัญญาณ Alternative Output Function ที่สร้างมาจากส่วน Timing and Control สัญญาณ Alternative Output Function เป็นสัญญาณที่ส่งออกในกรณีที่ใช้พอร์ท 3 ทำงานในฟังก์ชั่นอื่น และจุด Alternative Input Function เป็นจุดที่จะเอาสัญญาณไปเข้าส่วนอื่นตามการทำงานของบิทนั้น แต่ละบิทของพอร์ท 3 จะมีฟังก์ชั่นอื่นดังนี้

P3.0 / RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม

P3.1 / TXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม

P3.2 / $\overline{\text{INT0}}$ (External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.3 / $\overline{\text{INT1}}$ (External Interrupt) ใช้รับ สัญญาณขัดจังหวะจากภายนอก

P3.4 / T0 (Timer / Counter 0 External Input) ขารับสัญญาณเข้าไปยังวงจร Timer / Counter 0 ที่ทำหน้าที่นับจำนวนไซเคิลของสัญญาณ T0 นี้หรือสัญญาณนาฬิกาก็ได้

P3.5 / T1 (Timer / Counter 1 External Input) ขารับสัญญาณเข้าไปยังวงจร Timer / Counter 1 ซึ่งมีการทำงานเหมือนกับ T0

P3.6 / $\overline{\text{WR}}$ (External Data Memory Write Strobe) ขาสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก 8051

P3.7 / $\overline{\text{RD}}$ (External Data Memory Read Strobe) ขาสัญญาณควบคุมการอ่านข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก 8051

RST

ขารีเซ็ทขานี้จะใช้ทำการรีเซ็ทการทำงานของ 8051 ที่ขา RST ภายใน 8051 จะมีตัวต้านทานต่อระหว่างขานี้กับกราวด์ (Ground) ถ้าป้อนสัญญาณที่มีสภาวะลอจิก 1 เข้าไปที่ขานี้จะเป็นการรีเซ็ทการทำงานของ 8051 ดังนั้นจึงสามารถต่อตัวเก็บประจุ (Capacitor) ภายนอกระหว่างขา RST กับไฟเลี้ยง +5 โวลท์ เพื่อให้เกิดการรีเซ็ท เมื่อเริ่มป้อนไฟเลี้ยงให้กับ 8051 ซึ่งเรียกว่า Power on reset การรีเซ็ทจะทำให้ค่าในรีจิสเตอร์ต่าง ๆ เปลี่ยนไปเป็นค่าหนึ่งดังในตารางรูปที่ 4.10

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 ค่าของรีจิสเตอร์เมื่อเกิดการรีเซ็ท 8051

ในตารางรูปที่ 4.10 ช่องทางขวาเป็นค่าของรีจิสเตอร์ที่อยู่ทางซ้ายเมื่อสิ้นสุดการรีเซ็ท ในรีจิสเตอร์ SBUF เมื่อสิ้นสุดการรีเซ็ทจะมีค่าที่ไม่แน่นอน และพอร์ทจะอยู่ในสภาวะลอจิก 1 ทุกบิทตลอดเวลาที่สัญญาณของขา RST เป็น HIGH อยู่

เมื่อสัญญาณที่ขา RST กลับเป็น 0 ก็จะออกจากการรีเซ็ท 8051 จะเริ่มทำงานจากคำสั่งที่อยู่ใน Program memory ตำแหน่ง 0000H เพราะค่าของริจิสเตอร์ PC (Program Counter) ซึ่งใช้ชี้ตำแหน่งโปรแกรมที่จะทำงานถูกเปลี่ยนให้เป็น 0000H ดังนั้นผู้ใช้จะต้องเขียนโปรแกรมมาเก็บไว้ที่ตำแหน่ง 0000H ในเครื่องไมโครคอมพิวเตอร์แบบบอร์ดเดี่ยว (Single Board Microcomputer) จะมีโปรแกรมที่เขียนเก็บไว้เริ่มจากตำแหน่ง 0000H นี้เรียกว่า มอนิเตอร์

โปรแกรม (Monitor Program) ที่จะคอยรับการกดแป้นพิมพ์ (Keyboard) และแสดงผลทางตัวแสดงผล (Display) แบบ 7 Segment

ALE

Address Latch Enable ขานี้จะส่งสัญญาณที่มีความถี่ 1/6 เท่าของสัญญาณนาฬิกาจากออสซิลเลเตอร์สัญญาณนี้จะส่งออกมาตลอดเวลายกเว้นบางครั้งของการติดต่อกับหน่วยความจำสำหรับข้อมูลภายนอก 8051 สัญญาณนี้จะใช้บอกกับอุปกรณ์ภายนอก 8051 ว่าขณะนี้สัญญาณนี้ Active (เป็นลอจิก 1) จะมีการส่งข้อมูลที่เป็น 8 บิทล่างของตำแหน่งหน่วยความจำภายนอก 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 (Fetch Instruction) ที่จะนำไปทำงานมาจากหน่วยความจำสำหรับโปรแกรมภายนอก 8051 ในกรณีที่อ่านคำสั่งซึ่งเก็บอยู่ในหน่วยความจำสำหรับโปรแกรมภายใน 8051 แล้วสัญญาณนี้จะไม่เปลี่ยนลอจิกเป็น 0 ขา PSEN นี้สามารถต่อไปยังขาอินพุทของ TTL ชนิด LS ได้ถึง 8 อินพุท

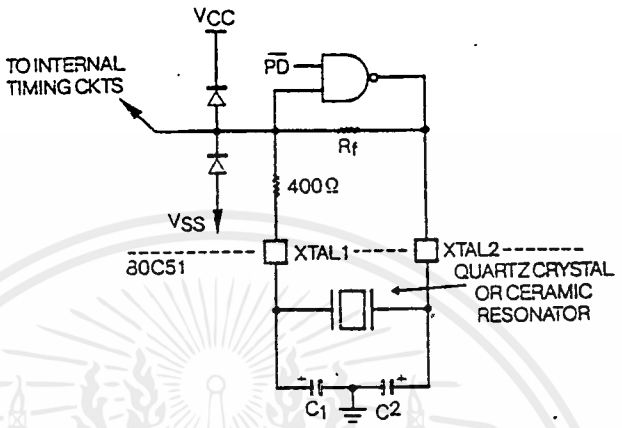
$\overline{\text{EA}}$

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 8051 จะต้องสร้างสัญญาณ PSEN ออกไปยังภายนอก เพื่อทำการ FETCH คำสั่งเข้ามาทำงาน แต่ถ้าสัญญาณที่ป้อนให้ขา EA เป็น 1 หมายความว่าโปรแกรมในตำแหน่ง 0000H ถึง 0FFFH ถูกเก็บไว้ภายใน 8051 การทำงานในตำแหน่งหน่วยความจำช่วงนี้จะอ่านคำสั่งต่าง ๆ จาก ROM ภายใน 8051

XTAL 1

ขาที่ 19 ของรูปที่ 4.5 ขานี้จะต่อเข้ากับขาของ Inverting Amplifier (วงจรขยายแบบป้อนกลับเฟสสัญญาณ) ที่ประกอบเป็นวงจรออสซิลเลเตอร์ ในรูปที่ 4.11 จะเห็นวงจรภายในของออสซิลเลเตอร์ NAND Gate จะทำหน้าที่เป็นวงจรขยายแบบกลับเฟสของสัญญาณที่จะควบคุมให้มีการออสซิลเลตหรือไม่ก็ขึ้นกับสัญญาณ PD ซึ่งต่อมาจากบิท PD ของรีจิสเตอร์ PCON ถ้า

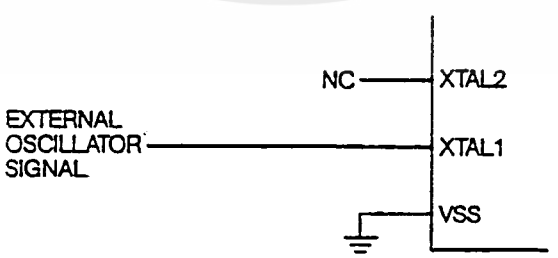
ต้องการใช้สัญญาณนาฬิกา (Clock Signal) จากภายนอกมาเป็นสัญญาณนาฬิกา ควบคุมการทำงานของ 8051 ก็ให้ป้อนสัญญาณเข้ามาที่จุดนี้แต่ถ้าต้องการใช้วงจรออสซิลเลเตอร์ภายในก็ให้ต่อ Crystal หรือเซรามิคเรโซเนเตอร์ดังรูปที่ 4.11 คาปาซิเตอร์ในวงจรควรมีค่าประมาณ 20 PF



รูปที่ 4.11 วงจรออสซิลเลเตอร์ภายใน 8051

XTAL 2

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



รูปที่ 4.12 8051 ที่ทำงานโดยสัญญาณที่มาจากภายนอก

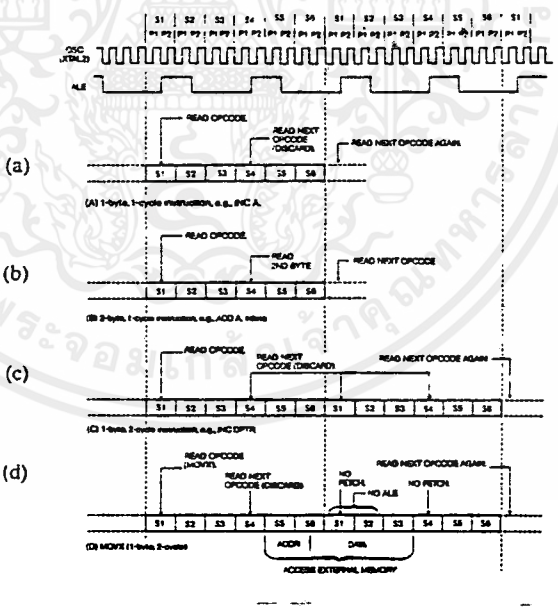
4.4 การทำงานของ 8051

คอมพิวเตอร์จะทำงานด้วยวงจรที่เรียกว่าฮาร์ดแวร์ (Hardware) ประกอบขึ้นมาเพียงอย่างเดียวไม่ได้ จะต้องมีโปรแกรมหรือคำสั่งที่จัดเรียงกันไว้ให้คอมพิวเตอร์ทำงานตามลำดับใน 8051 ก็เช่นกัน ผู้ใช้จะต้องเขียนโปรแกรมเป็นภาษาเครื่อง ซึ่งอยู่ในรูปของเลขฐานสอง เก็บไว้ในหน่วยความจำประเภท Program Memory แต่ละคำสั่งของ 8051 อาจประกอบด้วย 1,2 หรือ 3 ไบท์แล้วแต่ว่าจะเป็นคำสั่งให้ทำงานอะไร คอมพิวเตอร์ก็จะเหมือนกับคนที่จะต้องทำงานตามคำสั่ง เมื่อรับคำสั่งแล้วก็จะไปทำตามคำสั่งนั้นเสร็จสิ้นแล้วก็กลับมารับค่าสั่งต่อไป

จากรูปที่ 4.4 เมื่อเริ่มป้อนไฟเลี้ยงให้กับ 8051 ซึ่งมีวงจร Power on reset ต่ออยู่จะมีการรีเซ็ทเกิดขึ้น การทำงานภายใน 8051 จะเริ่มจากบล็อก Program Counter ซึ่งเป็นวงจรนับ (Counter Circuit) ชนิดหนึ่ง ส่งค่าตำแหน่งหน่วยความจำสำหรับโปรแกรมลงไปบนบัส (Bus) หมายเลข 1 บัสนี้มีขนาด 16 บิท ค่าตำแหน่งหน่วยความจำนี้จะถูกส่งไปเก็บไว้ที่ Program ADDR Register ที่เป็นวงจร Latch ข้อมูลซึ่งเป็นค่าตำแหน่งหน่วยความจำ จะปรากฏที่บนบัส 16 บิทหมายเลข 2 ถ้าเป็นค่าตำแหน่งหน่วยความจำแรกหลังจากรีเซ็ท ค่าตำแหน่งหน่วยความจำจะเป็น 0000H หน่วยความจำสำหรับโปรแกรมจะเลือกได้ว่าเป็น ROM ภายในหรือภายนอก 8051 โดยการป้อนสภาวะลอจิกเข้าไปที่ 8051 ทางขา $\overline{EA}$ ซึ่งต่ออยู่กับส่วน Timing and Control ทำหน้าที่เป็นวงจรถอดรหัส (Decoder) แล้วสร้างสัญญาณควบคุมต่อไปถ้าป้อนสัญญาณลอจิก 0 เข้าไปที่ขา $\overline{EA}$ จะเป็นการเลือกใช้ ROM ภายใน 8051 โดยที่วงจร Timing and Control จะสร้างสัญญาณไปยัง ROM ภายในให้ส่งข้อมูลที่เป็นคำสั่งจากตำแหน่งที่ถูกชี้ด้วยค่าตำแหน่งที่ส่งมาทางบัสหมายเลข 2 ข้อมูลจาก ROM จะถูกส่งลงไปยังบัสหมายเลข 3 ที่เรียกว่า Internal Data Bus แล้วนำไปเก็บไว้ที่ Instruction Register (เป็นวงจร Latch) เพื่อส่งต่อไปให้กับวงจร Timing and Control ทำการถอดรหัสแล้วควบคุมการทำงานส่วนอื่น ๆ ต่อไปแล้วแต่ว่าเป็นคำสั่งให้ทำงานอะไร ในกรณีที่เลือก ROM ภายนอก 8051 โดยป้อนสัญญาณลอจิก 1 เข้าไปที่ขา $\overline{EA}$ จะทำให้วงจร Timing and Control ส่งสัญญาณไปยังพอร์ท 0 และพอร์ท 2 เพื่อส่งค่าตำแหน่งหน่วยความจำบนบัสหมายเลข 2 ออกไปชี้หน่วยความจำภายนอก จากนั้นจะอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาทางพอร์ท 0 ไปยัง Internal Data Bus แล้วไปเก็บที่ Instruction Register เพื่อทำงานต่อไปเหมือนกับตอนอ่านคำสั่งจาก ROM ภายใน การทำงานในช่วงส่งค่าตำแหน่งหน่วยความจำไปยังหน่วยความจำแล้วอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาเก็บไว้ใน Instruction Register เรียกว่าเป็นช่วงของการ Fetch (Fetch Cycle) ช่วงต่อไปจะเป็นช่วงของการทำงานตามคำสั่งเรียกว่า Execute Cycle เช่นถ้าเป็นคำสั่งให้บวกข้อมูลในรีจิสเตอร์ Accumulator กับข้อมูลจากหน่วยความจำ Data Memory ภายใน RAM ตำแหน่ง 23H วงจร Timing and Control ก็จะ

ส่งสัญญาณให้ Instruction Register ส่งค่าตำแหน่งหน่วยความจำ 23H ลงไปยัง Internal Data Bus แล้วนำข้อมูลไปเก็บไว้ที่ RAM ADDR Register เพื่อใช้ชี้ตำแหน่งหน่วยความจำ RAM จากนั้น Timing and Control จะสั่งให้ RAM ส่งข้อมูลที่เก็บอยู่ในหน่วยความจำตำแหน่ง 23H ลงมายัง Internal Data Bus แล้วนำข้อมูลไปเก็บไว้ที่ TMP1 (วงจร Latch) ขณะเดียวกันวงจร Timing and Control ก็จะส่งสัญญาณไปยัง ACC ให้ส่งข้อมูลมายัง TMP2 (วงจร Latch) วงจร ALU ซึ่งโครงสร้างเป็นวงจรทำการคำนวณทางคณิตศาสตร์ (บวก, ลบ, คูณ, หาร) และยังสามารถทำงานทางลอจิก (AND, OR, NOT, XOR) จะทำการบวกเลขจาก TMP1 และ TMP2 เข้าด้วยกันผลลัพธ์ที่ได้จะส่งผ่าน Internal Data Bus กลับไปเก็บยัง ACC PSW (Program Status Word) ซึ่งจะทำหน้าที่เก็บสถานะผลลัพธ์ของการทำงานใน ALU เช่นผลลัพธ์การบวกมีค่าเกิน 8 บิท ก็จะทำให้บิทหนึ่งใน PSW ถูก SET เป็น 1

การทำงานที่กล่าวมาข้างต้นจะขึ้นกับสัญญาณควบคุมที่สร้างมาจากวงจร Timing and Control และสัญญาณที่สร้างขึ้นนี้จะอ้างอิงกับสัญญาณนาฬิกาที่สร้างมาจากวงจร Oscillator ทำให้การทำงานต่าง ๆ เป็นไปตามลำดับที่ผู้ผลิตได้ออกแบบไว้ดังรูปที่ 4.13



รูปที่ 4.13 ลำดับสถานะการทำงานใน MCS-51

คำสั่งแต่ละคำสั่งของ 8051 จะใช้เวลาทำงาน 1, 2 หรือ 3 ไซเคิลของเครื่อง (Machine Cycle) แล้วแต่ว่าเป็นคำสั่งประเภทใด 1 ไซเคิลของเครื่องจะใช้เวลา 12 ไซเคิลของสัญญาณนาฬิกา ดังนั้นแต่ละคำสั่งของ 8051 จะใช้เวลาการทำงาน 12 , 24 หรือ 36 ไซเคิลของสัญญาณ

นาฬิกานั่นเอง แต่ละไซเคิลของเครื่องถูกแบ่งออกเป็น 6 State คือ S1, S2, S3, S4, S5 และ S6 แต่ละ State จะประกอบด้วย 2 ไซเคิลของสัญญาณนาฬิกา ในไซเคิลแรกจะเรียกว่าเฟส 1 (P1) และไซเคิลที่ 2 เรียกเฟส 2 (P2) ในแต่ละเฟสจะนับตั้งแต่ขอบขาลงของสัญญาณนาฬิกาถึงขอบขาลงของสัญญาณนาฬิกาที่อยู่ถัดไปดังรูปที่ 4.13 เมื่อ 8051 ทำงานเสร็จ 1 ไซเคิลของเครื่องก็จะเริ่มทำงาน State 1 Phase (S1P1) ของไซเคิลต่อไป ใน 1 ไซเคิลของเครื่องวงจร Timing and Control จะสร้างสัญญาณ ALE ออกมา 2 ไซเคิลเพื่อ Fetch คำสั่งเข้าไป 2 ครั้งเสมอ ที่บริเวณขอบขาขึ้นของสัญญาณ ALE คำสั่งใดจะมีกี่ไบท์หรือใช้เวลาทำงานกี่ไซเคิลจะดูได้จากตารางชุดคำสั่ง 8051

คำสั่งประเภท 1 ไบท์ 1 ไซเคิลของเครื่องได้แก่คำสั่ง INC A จะมีการอ่านคำสั่งจากหน่วยความจำสำหรับโปรแกรม 2 ครั้ง ที่เวลาประมาณขอบขาขึ้นของสัญญาณ ALE เมื่อคำสั่งแรกถูกอ่านเข้าไปที่เวลาขอบขาขึ้นของสัญญาณ ALE แรก แล้วนำไปเก็บที่ Instruction Register เพื่อให้วงจร Timing and Control ถอดรหัส แล้วเข้าสู่การ Execute ขณะเดียวกันก็จะเริ่มต้นการ Fetch คำสั่งที่อยู่ในหน่วยความจำตำแหน่งถัดไปเข้ามาและคำสั่งที่ 2 จะถูกอ่านเข้ามาที่เวลาขอบขาขึ้นของสัญญาณ ALE ถัดไป วงจร Timing and Control เมื่อถอดรหัสคำสั่งแรกก็ทราบว่าการทำงานคำสั่งนี้ให้สิ้นสุดจะใช้คำสั่ง เพียง 1 ไบท์ ดังนั้นคำสั่งที่ถูกอ่านมาไบท์ที่ 2 จะไม่ถูกนำมาทำงาน เพียงแต่อ่านเข้ามาแล้วทิ้งไป (Discard) ดังในรูปที่ 4.13 (a)

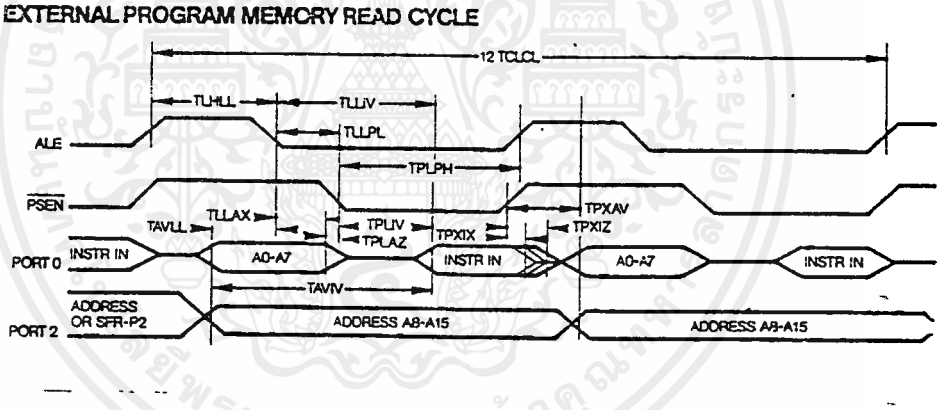
คำสั่งประเภท 2 ไบท์ และใช้เวลา 1 ไซเคิลของเครื่องได้แก่คำสั่ง ADD A , #data ในหนึ่งไซเคิลของเครื่องนี้จะมีการอ่านคำสั่งเข้ามา 2 ไบท์ เหมือนกับคำสั่งประเภท 1 ไบท์ 1 ไซเคิลของเครื่อง แตกต่างกันที่ไบท์ 2 จะถูกนำมาใช้งานด้วยไม่ได้ถูกทิ้งไปดังในรูปที่ 4.13(b) ตัวอย่างของคำสั่ง ADD A , #33H จะเขียนเป็นภาษาเครื่องได้ 2 ไบท์ คือ 24 33 เมื่ออ่านคำสั่งไบท์แรกคือ 24 เข้าไปไว้ที่ Instruction Register แล้ว Timing and Control จะถอดรหัสพบว่าเป็นคำสั่งบวกเลขก็จะส่งสัญญาณไปยัง Accumulator ให้เอาข้อมูลไปไว้ที่ TMP1 เมื่อคำสั่งที่ 2 ถูกอ่านเข้ามาที่ Instruction Register แล้ว Timing and Control จะสั่งให้เอาข้อมูลไบท์ที่ 2 ส่งลงไปยัง Internal Data Bus ไปเก็บยัง TMP1 จากนั้นวงจร ALU จะนำเอาข้อมูล TMP1 และ TMP2 มาบวกกัน ผลลัพธ์ที่ได้จะส่งออกจาก ALU ไปยัง Internal Data Bus แล้วไปเก็บไว้ที่ Accumulator

คำสั่งประเภท 1, 2 หรือ 3 ไบท์ ที่ใช้เวลาทำงาน 2 ไซเคิลของเครื่องเช่นคำสั่ง INC DPTR จะมีการอ่านคำสั่งเข้าไป 4 ครั้งทุก ๆ ขอบขาขึ้นของสัญญาณ ALE ที่มี 2 ครั้งต่อ 1 ไซเคิลของเครื่อง ถ้าเป็นคำสั่งประเภท 1, 2 หรือ 3 ไบท์ วงจร Timing and Control จะเอาคำสั่ง 1, 2 หรือ 3 ไบท์แรกเท่านั้นไปทำงานส่วนคำสั่งที่เหลือทิ้งไปดังในรูปที่ 4.13(c.) คำสั่ง 1 ไบท์

ที่ใช้เวลาทำงาน 2 ไซเคิลของเครื่องที่กล่าวมาแล้วจะไม่รวมถึงคำสั่ง MOVX ซึ่งใช้ในการอ่านหรือเขียนข้อมูลกับหน่วยความจำ Data Memory ภายนอก การทำงานของคำสั่งนี้จะมีการ Fetch คำสั่งเข้าไป 2 ไบท์ในไซเคิลของเครื่องแรก ในไซเคิลของเครื่องที่ 2 จะไม่มีการ Fetch คำสั่งเข้าไปแต่จะเป็นช่วงเวลาของการอ่านหรือเขียนข้อมูลกับ Data Memory ภายนอก สัญญาณ ALE ซึ่งปกติจะเปลี่ยนเป็น 1 ที่ S1P2 ก็จะไม่เปลี่ยนเป็น 1 ในไซเคิลของเครื่องที่ 2 โดยจะเป็น 0 อยู่จนกว่าจะถึงเวลา S4P2 ของไซเคิลของเครื่องที่ 2 สัญญาณ ALE จะเปลี่ยนเป็น 1 เพื่อทำการอ่านหรือเขียนข้อมูลกับ Data Memory ภายนอก

4.5 ไดอะแกรมเวลาของการติดต่อกับหน่วยความจำ

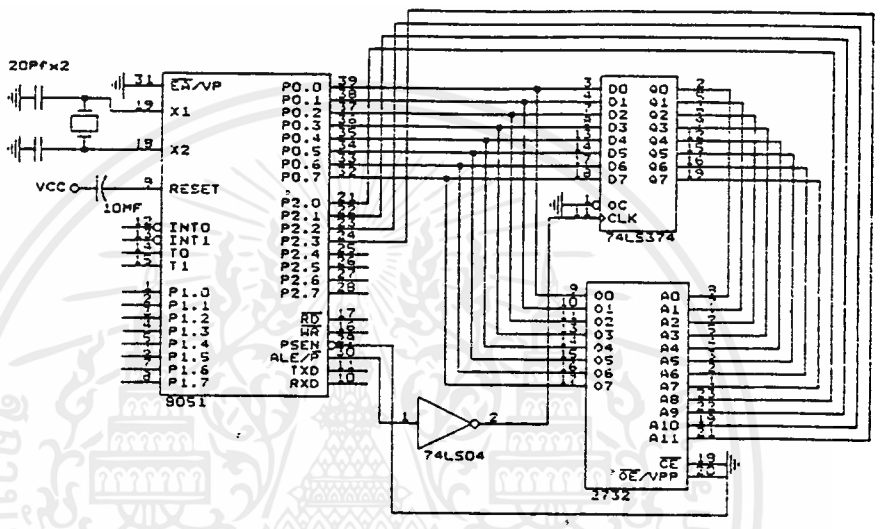
การอ่านข้อมูลจากหน่วยความจำสำหรับโปรแกรมภายนอก 8051 นั้น ลำดับสัญญาณตามเวลา (Timing Diagram) ของสัญญาณที่ทำการอ่านคำสั่งมีดังรูปที่ 4.14



รูปที่ 4.14 Timing Diagram ของการอ่านโปรแกรมจากหน่วยความจำภายนอก

การอ่านคำสั่ง (Fetch) จาก Program area ภายนอกจะเริ่มจาก 8051 ส่งสัญญาณลอจิก 1 ออกมาทางขา ALE ขณะนี้สัญญาณที่ขา PSEN จะเป็น 1 จากนั้น Port 0 จะส่งค่าตำแหน่งหน่วยความจำ 8 บิทล่างและพอร์ท 2 จะส่งตำแหน่งหน่วยความจำ 8 บิทบนออกมาแล้ว สัญญาณ ALE จะกลับเป็น 0 อุปกรณ์ภายนอกจะสามารถใช้ขอบขาลงของสัญญาณ ALE เพื่อ Latch ตำแหน่งหน่วยความจำที่พอร์ท 0 ไว้ จากนั้นพอร์ท 0 ก็จะยกเลิกการส่งค่าตำแหน่งหน่วยความจำเข้าสู่สภาวะ High Impedance และสัญญาณ $\overline{\text{PSEN}}$ จะเป็น 0 เพื่อเตรียมรับคำสั่งที่ส่งออกจากหน่วยความจำภายนอกเข้าไปยัง 8051 เพื่อทำงานต่อไป เมื่อคำสั่งถูกอ่านเข้าไปเก็บใน Instruction Register (ดูรูป 4.4) แล้วสัญญาณ $\overline{\text{PSEN}}$ จะกลับเป็น 1 พร้อมกับสัญญาณ ALE

ก็จะกลับเป็น High เพื่อการอ่านคำสั่งต่อไปทำงาน ข้อมูลในพอร์ท 2 จะคงที่ตลอดเวลา ตั้งแต่สัญญาณ ALE เป็น 1 จนกระทั่งสัญญาณ ALE เปลี่ยนเป็น 0 และกลับเป็น 1 อีกครั้งหนึ่งจากนั้นจะเริ่มลำดับการ Fetch ข้อมูลไบท์ที่ 2 จากหน่วยความจำสำหรับโปรแกรม ซึ่งจะมีการเปลี่ยนแปลงของสัญญาณตามเวลาเหมือนกับการ Fetch ไบท์แรกนั่นเอง จาก Timing Diagram ดังกล่าวจะออกแบบวงจรที่มี Program Memory อยู่ภายนอก 8051 ได้ดังตัวอย่างในรูปที่ 4.15



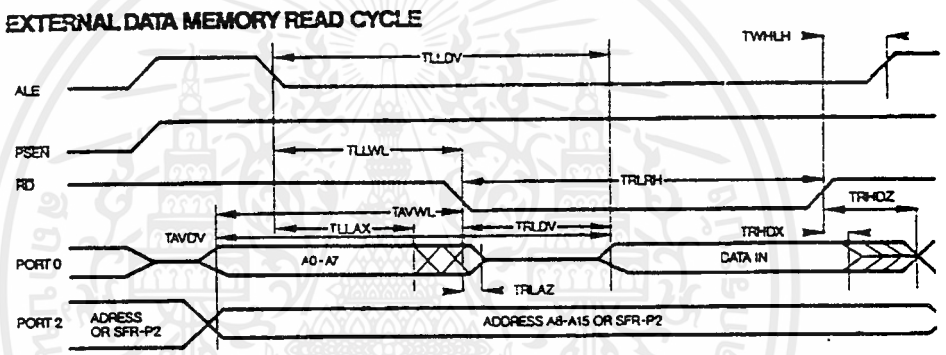
รูปที่ 4.15 วงจรที่มี Program Memory อยู่ภายนอก 8051

74LS374 ในรูปที่ 4.15 จะทำหน้าที่ Latch ตำแหน่งในหน่วยความจำ 8 บิทล่างที่เวลาขอบขาลงของสัญญาณ ALE ซึ่งสัญญาณ ALE จะถูกกลับให้เป็นตรงข้ามโดย Inverter 74LS04 ก่อนที่จะป้อนให้กับขา CK ของ 74LS374 และที่ขอบขาขึ้นของสัญญาณที่ออกจาก 74LS04 จะ Latch ตำแหน่งหน่วยความจำ ข้อมูลที่ออกจาก 74LS374 จะเป็นค่า 8 บิทล่างของตำแหน่งหน่วยความจำที่ต้องการติดต่อ ในวงจรได้ต่อค่าตำแหน่งหน่วยความจำ 8 บิทเข้ากับ A0 ถึง A7 ของ EPROM และข้อมูลจากพอร์ท 2 บิท P2.0 ถึง P2.3 จะต่อเข้ากับ A8 - A11 ของ EPROM โดยตรงเพราะค่าตำแหน่งหน่วยความจำ 8 บิทบนที่ออกมาจากพอร์ท 2 จะคงที่ตลอดเวลา ขา $\overline{\text{PSEN}}$ ของ 8051 จะถูกต่อเข้ากับขา OE ของ EPROM 2716 ดังนั้นเมื่อสัญญาณ $\overline{\text{PSEN}}$ มีสภาวะลอจิกเป็น 0 ก็จะส่งคำสั่งที่เก็บใน EPROM ณ ตำแหน่งที่ชี้โดยข้อมูลที่ขา A0 ถึง A11 ออกมายังพอร์ท 0 และถูก 8051 เก็บไปทำงานต่อไป

การอ่าน - เขียนข้อมูลกับหน่วยความจำสำหรับข้อมูลภายนอก 8051

การอ่าน - เขียนข้อมูลกับ Data memory ภายใน 8051 นั้นจะมีสัญญาณสร้างมาจากส่วน Timing and Control โดยที่ผู้ใช้ไม่จำเป็นที่จะต้องทำความเข้าใจ แต่การอ่าน - เขียนข้อมูลกับ Data Memory ภายนอก อันเนื่องมาจากคำสั่ง MOVX นั้น เมื่อคำสั่งดังกล่าวถูกอ่านเข้ามายัง Instruction Register แล้ว Timing and Control จะทำการถอดรหัสแล้วสร้างสัญญาณควบคุมดังนี้

การอ่านข้อมูลจาก External Data Memory จะมีไดอะแกรมสัญญาณตามเวลาดังรูปที่ 4.16

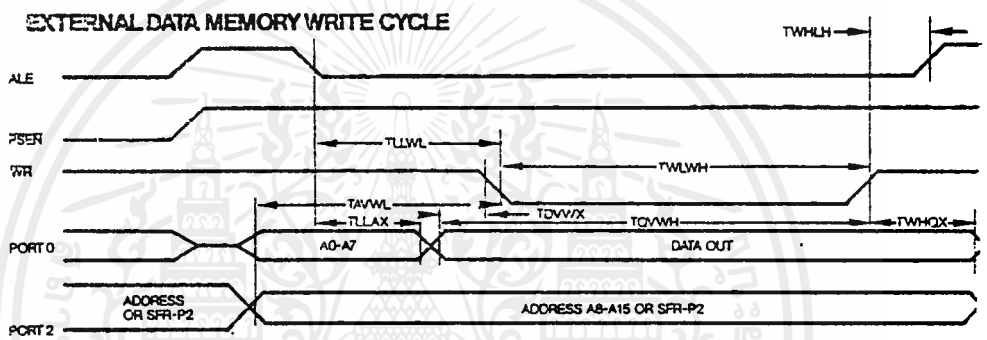


รูปที่ 4.16 Timing diagram ของการอ่านข้อมูลจากหน่วยความจำสำหรับข้อมูลภายนอก 8051

การทำงานจะเริ่มจากการส่งค่าตำแหน่งหน่วยความจำภายนอก 8 บิทล่างออกทางพอร์ท 0 และ 8 บิทบนออกทางพอร์ท 2 เมื่อส่งค่าตำแหน่งแล้ว สัญญาณ ALE ซึ่งเดิมมีลอจิกเป็น 1 จะกลับมาเป็น 0 เพื่อให้อุปกรณ์ภายนอกสามารถ Latch ตำแหน่งหน่วยความจำไว้เหมือนกับในการอ่านข้อมูลจากหน่วยความจำสำหรับโปรแกรมภายนอก 8051 เพื่อส่งไปยังหน่วยความจำ แม้ว่าข้อมูลบนพอร์ท 0 จะเปลี่ยนแปลงไปก็จะยังมีค่าตำแหน่งหน่วยความจำส่งไปยังหน่วยความจำในระหว่างการติดต่อกับ Data Memory นี้สัญญาณ $\overline{\text{PSEN}}$ จะเป็น 1 ตลอดเวลา สัญญาณ $\overline{\text{PSEN}}$ จะ Active (เป็น 0) ก็ต่อเมื่อเป็นการติดต่อกับหน่วยความจำสำหรับโปรแกรมภายนอก 8051 เท่านั้น 8051 จะส่งสัญญาณลอจิก 0 ออกมาทางขา $\overline{\text{RD}}$ (P3.7) เพื่อบอกกับหน่วยความจำภายนอกว่าต้องการอ่านข้อมูลเข้าไปเมื่อ 8051 ส่งสัญญาณ $\overline{\text{RD}}$ เป็นลอจิก 0 จะทำให้พอร์ท 0 เข้าสู่สถานะ High Impedance พร้อมที่จะให้หน่วยความจำภายนอกส่งข้อมูลมาบนพอร์ท 0 ข้อมูลบน

พอร์ท 0 ซึ่งส่งมาจากหน่วยความจำภายนอกจะถูกอ่านเข้าไปเก็บที่เวลาขอบขาขึ้นของสัญญาณ $\overline{RD}$ จากนั้นสามารถ ALE ก็จะกลับเป็น 1 เพื่อเริ่มการทำงานในคำสั่งต่อไปในระหว่างการอ่านข้อมูลจากหน่วยความจำสำหรับข้อมูลภายนอกนี้ พอร์ท 2 จะส่งค่าตำแหน่งหน่วยความจำ 8 บิทบนออกมาตลอดเวลา

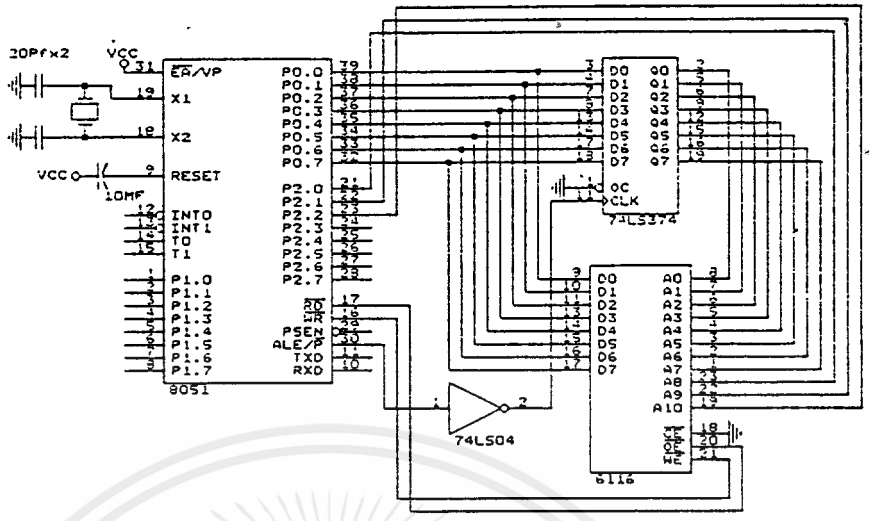
การเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก 8051 จะมีไดอะแกรมสัญญาณตามเวลาดังรูปที่ 4.17



รูปที่ 4.17 Timing diagram ของการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก 8051

เมื่อ 8051 ส่งค่าตำแหน่งหน่วยความจำ 8 บิทล่างไปทางพอร์ท 0 และ 8 บิทบนลงไปทางพอร์ท 2 แล้ว สัญญาณ ALE จะกลับเป็น 0 อุปกรณ์ภายนอกจะสามารถใช้สัญญาณนี้ในการ Latch ค่าตำแหน่งหน่วยความจำบนพอร์ท 0 เหมือนกับในการอ่านข้อมูลจากหน่วยความจำสำหรับข้อมูลภายนอก เมื่อสัญญาณ ALE เป็น 0 แล้ว 8051 จะส่งข้อมูลที่ต้องการเขียนไปยังพอร์ท 0 แล้วจะให้สัญญาณ $\overline{WR}$ เปลี่ยนสภาวะลอจิกเป็น 0 ขณะนี้หน่วยความจำภายนอกจะต้องเขียนข้อมูลไปเก็บยังตำแหน่งที่กำหนด จากนั้นสัญญาณ $\overline{WR}$ จะกลับเป็น 1 เพื่อเป็นการบอกสิ้นสุดการเขียนข้อมูลแล้วสัญญาณ ALE ก็จะกลับเป็น 1 เพื่อ Fetch คำสั่งต่อไปมาทำงาน

หน่วยความจำสำหรับข้อมูลภายนอกที่สามารถอ่านและเขียนข้อมูลได้ จะสามารถเขียนเป็นวงจรได้ดังรูปที่ 4.18

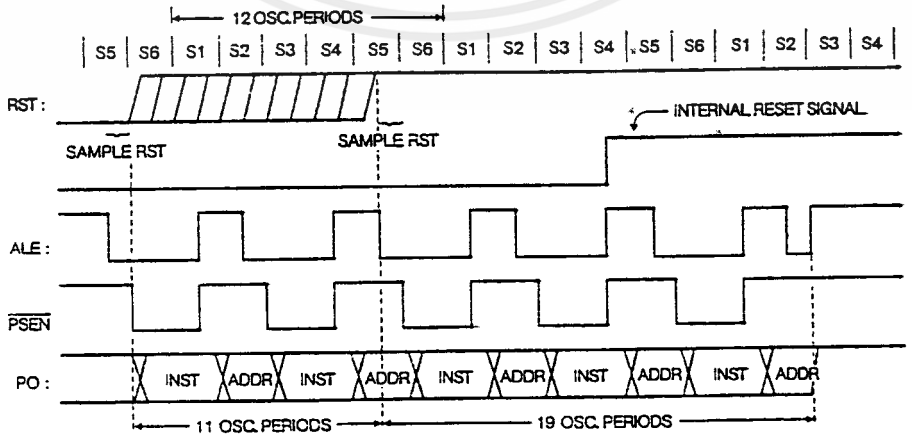


รูปที่ 4.18 วงจรที่มีหน่วยความจำสำหรับข้อมูลที่อยู่ภายนอก 8051

74LS374 ในรูปจะใช้สำหรับ Latch ค่าตำแหน่งหน่วยความจำ 8 บิทล่างไว้ แม้ว่าข้อมูลบนพอร์ท 2 จะเปลี่ยนไป สัญญาณ $\overline{RD}$ และ $\overline{WR}$ จะอ่านหรือเขียนข้อมูลจากหน่วยความจำภายนอก 6166 เป็นหน่วยความจำแบบ RAM ที่สามารถจะอ่านและเขียนข้อมูลได้

4.6 การรีเซ็ท

เมื่อป้อนสัญญาณที่มีสภาวะลอจิก 1 เข้าไปทางขา RST จะไม่ได้เกิดการรีเซ็ทขึ้นทันทีทันใด แต่ลำดับการเกิดรีเซ็ทจะแสดงได้ดังไดอะแกรมตามเวลาในรูปที่ 4.19



รูปที่ 4.19 ไดอะแกรมตามเวลาของการรีเซ็ท

ในรูป 4.19 เป็น Timing Diagram ของการรีเซ็ท สภาวะลอจิกของสัญญาณที่ขา RST จะถูกอ่านเข้ามาที่เวลา S5P2 (เฟส 2 State 5) ของทุก ๆ ไซเคิลของเครื่อง ในกรณีที่เป็นคำสั่งซึ่งมีการทำงานเสร็จสิ้นใน 2 ไซเคิลของเครื่องก็จะตรวจสอบเฉพาะสัญญาณที่อ่านเข้ามาในไซเคิลที่ 2 ของการทำงาน ดังนั้นในการรีเซ็ทจะต้องป้อนสัญญาณที่มีสภาวะลอจิก 1 เข้าไปที่ขานี้เป็นเวลาอย่างน้อย 2 ไซเคิลของเครื่องหรือ 24 ไซเคิลของสัญญาณนาฬิกาที่สร้างจากวงจรออสซิลเลเตอร์ภายใน 8051 เพื่อให้แน่ใจว่าสัญญาณรีเซ็ทจะถูกอ่านเข้าไปตรวจสอบและทำงาน ขณะที่ทำการรีเซ็ท 8051 ออสซิลเลเตอร์จึงจะต้องทำงานอยู่ด้วย เมื่อ 8051 สุ่มข้อมูลที่ขา RST แล้วตรวจพบว่าเป็นสภาวะลอจิก 1 ก็จะสร้างสัญญาณรีเซ็ทขึ้นภายใน ที่เวลา S2P4 ของไซเคิลเครื่องถัดไป ข้อมูลที่แต่ละพอร์ทส่งออกมาจะยังคงปรากฏที่พอร์ทจนกว่าจะเกิดการรีเซ็ทขึ้นซึ่งต้องใช้เวลา 19 ไซเคิลของสัญญาณจากออสซิลเลเตอร์นับตั้งแต่เวลา S5P2 ในไซเคิลของเครื่องที่พบสัญญาณรีเซ็ท ในระหว่างเวลา 19 ไซเคิลนี้จะยังคงมีการ Fetch คำสั่งเข้าไปทำงานได้อยู่

สภาวะของสัญญาณลอจิกที่ขา RST จะถูกอ่านเข้าไปตรวจสอบที่เวลา S5P2 ของทุก ๆ ไซเคิลของเครื่อง ดังนั้นถึงแม้ว่าสัญญาณที่ขา RST จะมีลอจิกเป็น 1 มาก่อนก็จะยังไม่เกิดการตรวจสอบสัญญาณรีเซ็ท ดังในรูปที่ 4.19 สัญญาณที่ขา RST อาจเป็น 1 มาตั้งแต่ State ที่ 6 ก็จะไม่เกิดอะไรขึ้นจนกระทั่ง 1 ไซเคิลของออสซิลเลเตอร์ต่อมาซึ่งเป็นเวลา S5P2 จึงจะเกิดการตรวจสอบสัญญาณที่ขา RST ถ้าคำสั่งนั้นมีการทำงานมากกว่า 1 ไซเคิลของเครื่อง 8051 ก็จะต้องทำงานในคำสั่งนั้นให้เสร็จสิ้นเสียก่อนจึงจะเริ่มการรีเซ็ทได้ โดย 8051 จะดูสภาวะของสัญญาณที่ขา RST ของ S5P2 ในไซเคิลของเครื่องสุดท้ายเท่านั้น ดังนั้นใน S5P2 ของไซเคิลเครื่องแรก ๆ ในคำสั่งอาจมีสภาวะลอจิกที่ขา RST เป็น 1 แต่ที่ S5P2 ของไซเคิลของเครื่องสุดท้าย มีสภาวะลอจิกที่ขา RST เป็น 0 ก็จะไม่เกิดการรีเซ็ท

ที่เวลา S5P2 เมื่อตรวจสอบสภาวะสัญญาณที่ขา RST แล้วพบว่าเป็น 1 จะต้องรอไปจนถึงเวลา S4P2 ของไซเคิลของเครื่องถัดไปจึงจะทำให้สัญญาณรีเซ็ทภายในเปลี่ยนสภาวะลอจิกจาก 0 เป็น 1 ในระหว่างเวลา S5P2 ที่ตรวจพบสัญญาณ RST มีลอจิกเป็น 1 จนถึง S4P2 ของไซเคิลของเครื่องถัดไปจะยังคงมีการ Fetch คำสั่งเข้าไปทำงานอีก 2 คำสั่ง เมื่อสัญญาณรีเซ็ทภายในเปลี่ยนเป็น 1 ก็จะเริ่มการรีเซ็ท โดยการเขียนข้อมูล 0 ไปยัง Special Function Register ทุกตัวยกเว้นพอร์ท 0 ถึงพอร์ท 3 Stack Pointer และรีจิสเตอร์ SBUF ดังตารางในรูปที่ 1.11 ระหว่างนี้ข้อมูลใน RAM ภายใน 8051 จะไม่เปลี่ยนแปลงข้อมูลในระหว่างการเขียนข้อมูลลงไปยัง SFR จะยังมีการ Fetch คำสั่งเข้ามาทำงานอีก 1 คำสั่งจนกว่าจะถึง S3P1 ของไซเคิลของเครื่องที่ 2 (นับแต่ไซเคิลของเครื่องที่ตรวจพบลอจิก 1 ที่ขา RST) ก็จะทำให้สภาวะ

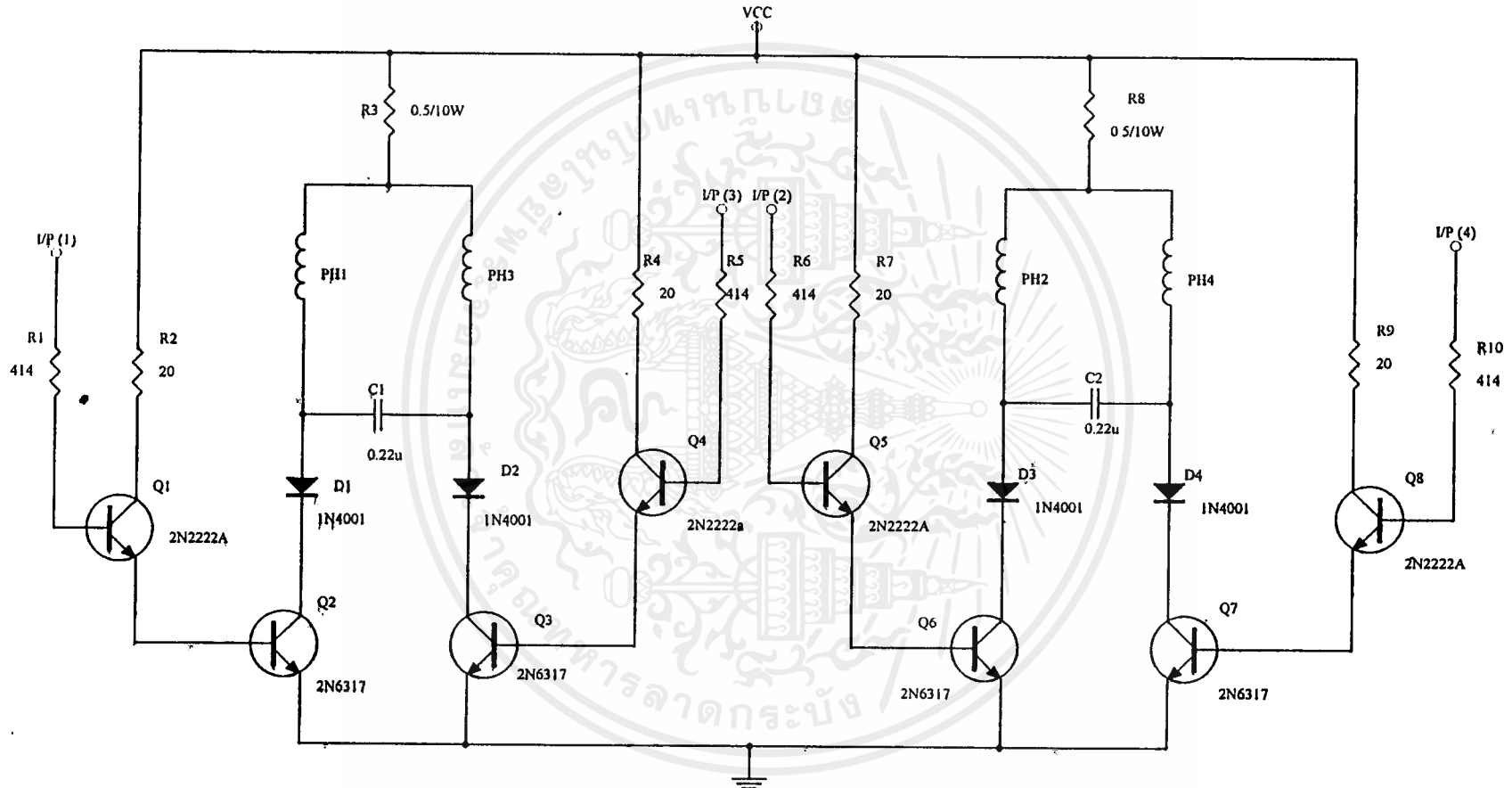
ลอจิกที่ขา ALE และ $\overline{\text{PSEN}}$ ค้างอยู่ที่สภาวะลอจิก 1 และจะเป็นอย่างนี้ไปจนกว่าสภาวะลอจิกที่ขา RST เป็น 0 เวลานับตั้งแต่พบสัญญาณลอจิก 1 ที่ขา RST ที่เวลา S5P2 จนถึงเวลาที่ ALE และ $\overline{\text{PSEN}}$ ค้างอยู่ที่ 1 จะเท่ากับ 19 ไซเคิลของออสซิลเลเตอร์เมื่อสัญญาณที่ขา RST ถูกเปลี่ยนกลับเป็นลอจิก 0 8051 จะรออีก 1 ถึง 2 ไซเคิลของเครื่องสัญญาณ ALE และ $\overline{\text{PSEN}}$ จะเริ่มเปลี่ยนแปลงเพื่อเริ่มกระบวนการ Fetch คำสั่งเข้าไปทำงานเริ่มจากคำสั่งในหน่วยความจำสำหรับโปรแกรมตำแหน่ง 0000H

บทที่ 5
การออกแบบวงจรควบคุมรถและ อัลกอริทึมของโปรแกรม Lane Detection

5.1 การทำงานของวงจร Driver

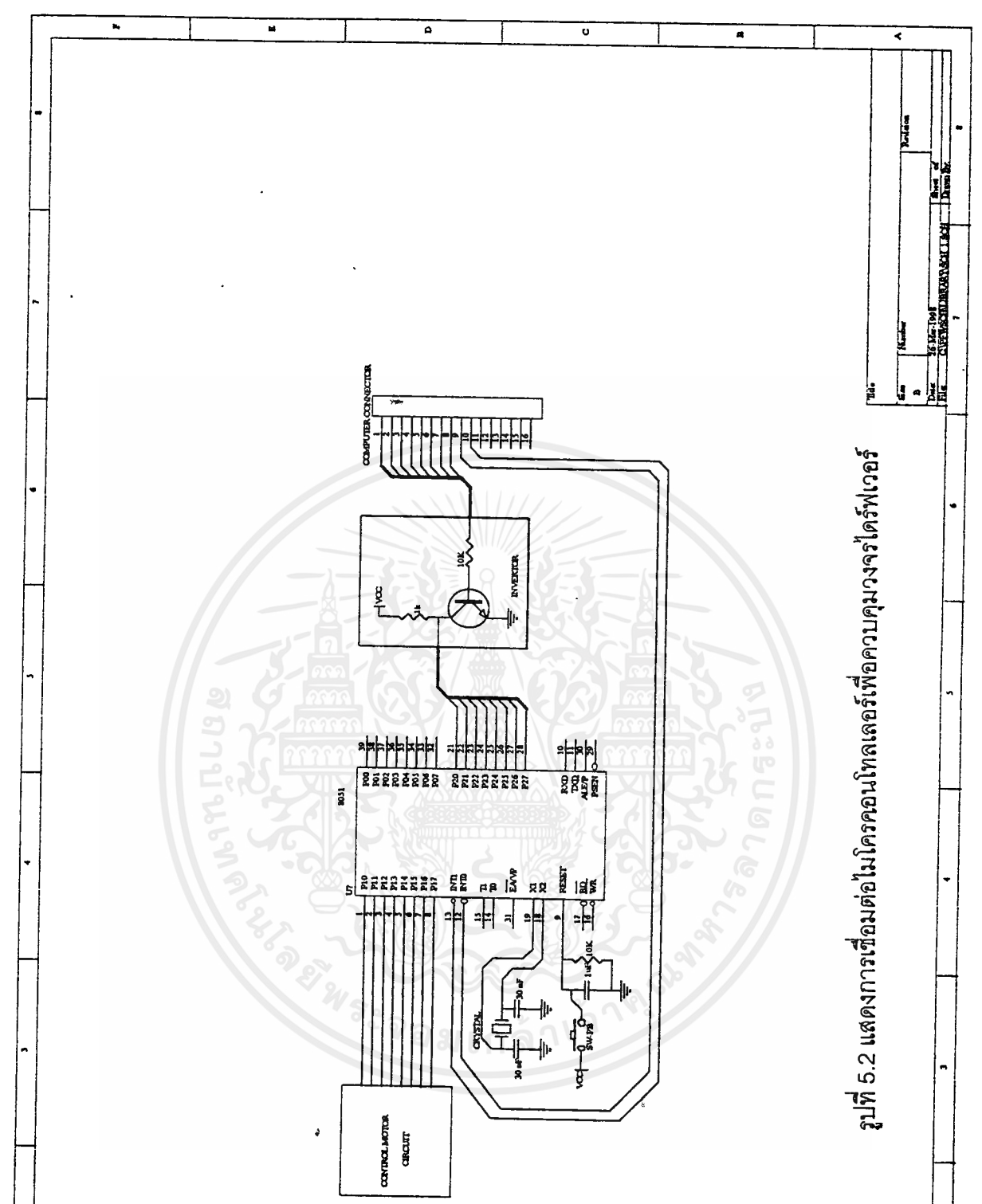
การทำงานของวงจร Driver จะเริ่มจากไมโครคอนโทรลเลอร์ 89C51 จะสัญญาณไปยังส่วนวงจร Driver เพื่อทำการขับสเต็ปเปอร์มอเตอร์ โดยในภาคไดรฟ์จะใช้ทรานซิสเตอร์ เบอร์ 2N2222A ต่อแบบดาร์ลิงตันกับทรานซิสเตอร์ เบอร์ 2N6317 ทรานซิสเตอร์จะต่อกันแบบดาร์ลิงตันและมีไดโอดเบอร์ 1N4001 ต่อคร่อมขาคอลเลคเตอร์กับขาอีมิตเตอร์เพื่อให้ไฟลบในขดลวดถูกบายพาส (Bypass) ลงกราวด์และเพื่อป้องกันแรงดันย้อนกลับในขดลวดของมอเตอร์ ซึ่งการกระตุ้นการทำงานของสเต็ปเปอร์มอเตอร์จะต้องเป็นไปตามลำดับ (Sequence) ทั้งในทิศทางตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา โดยเราจะใช้โปรแกรม Visual C++ เขียนโปรแกรมเพื่อควบคุมการติดต่อกับ 8255 Card Port เพื่อส่งข้อมูลควบคุมให้กับไมโครคอนโทรลเลอร์ โดยการใช้สายแพเป็นสายนำสัญญาณข้อมูล (Bus) ซึ่งต้องใช้สายแพ 15 PIN จำนวน 1 เส้น เนื่องจากในโครงงานนี้จะใช้พอร์ทของการ์ดพอร์ท 8255 เพียงพอร์ทเดียว คือ พอร์ท C เป็นเอาท์พุทพอร์ท

ฮาร์ดแวร์ควบคุมการทำงานของสเต็ปเปอร์มอเตอร์ ประกอบด้วย การ์ดพอร์ท 8255 ซึ่งเสียบกับสล็อตของคอมพิวเตอร์ และต่อสายแพ 15 PIN จำนวน 1 เส้นมายังส่วนของไมโครคอนโทรลเลอร์และใช้ไมโครคอนโทรลเลอร์ควบคุมสเต็ปเปอร์มอเตอร์อีกทีหนึ่ง โดยพอร์ท C จะถูกแบ่งออกเป็น 2 ส่วน ส่วนละ 4 PIN โดยที่แต่ละส่วนจะแบ่งการทำงานในการส่งสัญญาณให้กับไมโครคอนโทรลเลอร์ โดยอัลกอริทึมการควบคุมสเต็ปเปอร์มอเตอร์โดยไมโครคอนโทรลเลอร์จะอธิบายในหัวข้อต่อไป เราสามารถแสดงรูปการต่อวงจร Driver ของสเต็ปเปอร์มอเตอร์ได้ดังรูปที่ 5.1 และรูปแสดงการเชื่อมต่อไมโครคอนโทรลเลอร์เพื่อควบคุมวงจร Driver ดังรูปที่ 5.2



รูปที่ 5.1 แสดงวงจรไดรฟ์สเต็ปเปอร์มอเตอร์

Title		
Size A4	Number	Revision
Date: 16-Oct-1997		Sheet of
File: A:\STEP.SCH		Drawn By:



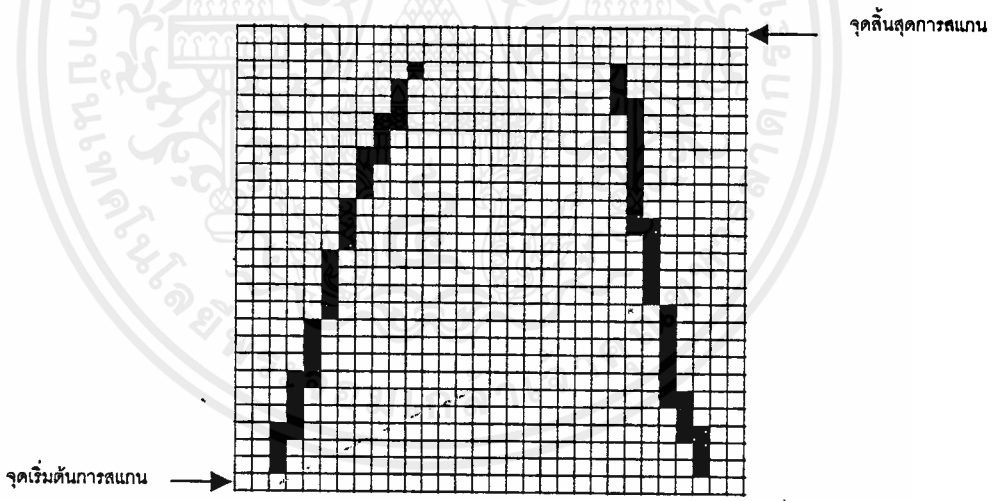
รูปที่ 5.2 แสดงการเชื่อมต่อไมโครคอนโทรเลอร์เพื่อควบคุมวงจรไดร์ฟเวอร์

5.2 อัลกอริทึมการทำงานในส่วนต่าง ๆ ของโปรแกรม

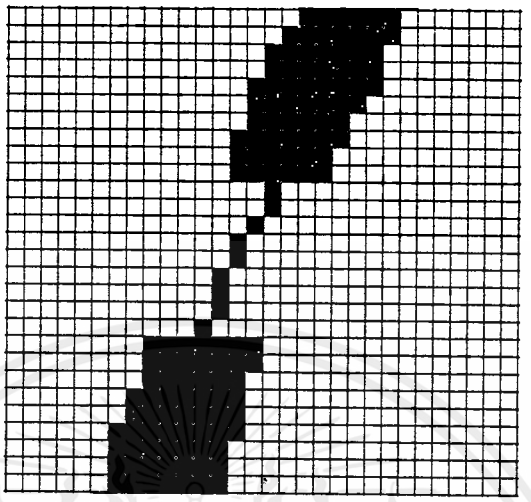
5.2.1 การหาจุดกึ่งกลางของเลน

หลังจากที่ภาพผ่านกระบวนการ Thinning แล้ว โปรแกรมจะได้เส้นขอบถนนที่มีความบางเท่ากับ 1 pixel ซึ่งความบางนี้อาจจะมีค่ามากกว่า 1 pixel ก็ได้ และหลังจากนั้นก็จะมาถึงกระบวนการในการหาจุดกึ่งกลางของเลนส์ ซึ่งจะมีกระบวนการดังต่อไปนี้

อัลกอลิทึมของโปรแกรมแสดงให้ดูดังรูปที่ 5.3 โปรแกรมจะเริ่มจากการสแกน pixel ทุก pixel ในเฉพาะแถวที่ผ่านการทำ Thinning แล้ว จากการสแกนโปรแกรมจะตรวบสอบ แต่ละ pixel เมื่อพบ pixel สีขาว โปรแกรมจะข้าม pixel นั้นไปแล้วไปสแกน pixel ใหม่ จนกว่าจะพบ pixel สีดำ เมื่อโปรแกรมพบ pixel สีดำแล้ว โปรแกรมจะทำการเก็บค่าตำแหน่งของ pixel สีดำนั้นไว้ แล้วทำการเพิ่มค่าตัวนับจำนวน pixel สีดำที่ถูกพบในแถวเดียวกัน ดังนั้นในแถวเดียวกันอาจจะสามารถมี pixel สีดำมากกว่า 2 pixel ก็ได้ ขึ้นอยู่กับความชัดเจนของภาพ ความผิดพลาดที่เกิดขึ้น และประสิทธิภาพของวิธีการ Thinning



รูปที่ 5.3 แสดงการทำ Thinning ทั้งเส้น



รูปที่ 5.4 แสดงการทำ Thinning เพื่อให้ได้เส้นสีดำบางลงเหลือขนาดเพียง 1 pixel

หลังจากทำ Thinning แล้วโปรแกรมจะทำการตรวจสอบตัวนับ pixel สีดำ โดยสามารถแบ่งออกได้เป็น 3 กรณีดังนี้

1. ในกรณีที่ตัวนับเท่ากับหนึ่ง อาจจะเกิดได้จากบริเวณที่ถูกทำ Thinning นั้น อยู่ในบริเวณถนนที่เป็นทางตรง ซึ่งเส้นแบ่งเลนส์กลางจะเป็นเส้นที่ไม่ต่อเนื่อง ทำให้มีการ ขาดหายไปเป็นช่วงๆหรืออาจเกิดจากเส้นแบ่งเลนส์ของถนน ได้หลุดออกจากนอกขอบเขตของกล้องที่ จะสามารถจับภาพได้

2. ในกรณีที่ตัวนับจำนวน pixel เท่ากับสอง อาจเกิดจากเส้นแบ่งเลนส์ของถนนทั้งสองเส้นปรากฏอยู่ ในขอบเขตของกล้อง หรืออาจเกิดจากมีเส้นแบ่งเลนส์เพียงเส้นเดียว และอีก pixel เกิดจากความผิดพลาดของข้อมูลภาพที่รับเข้ามา

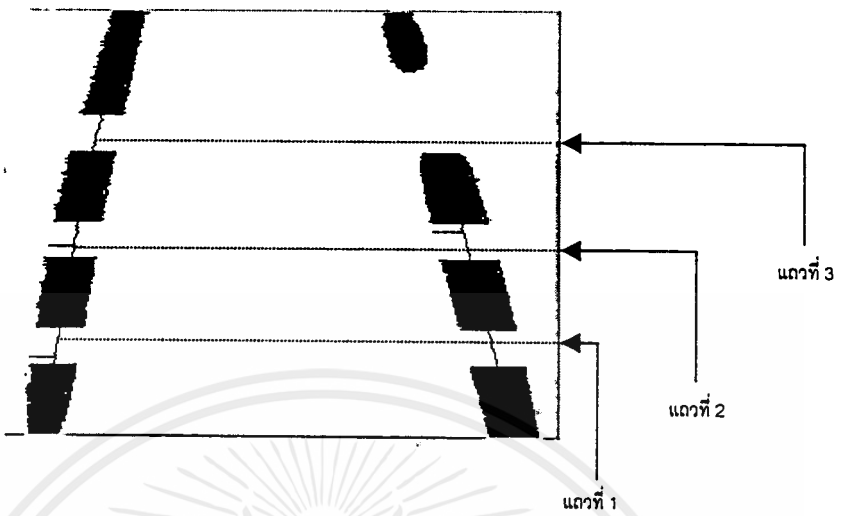
3. ในกรณีที่ตัวนับมีค่ามากกว่าสอง เกิดเนื่องมาจากความผิดพลาดของข้อมูล ซึ่งเกิดจากแสงสะท้อนจากภายนอก หรือความผิดพลาดอื่นๆ ซึ่งทำให้การประมวลผลเกิดความผิดพลาดได้

กรณีทั้ง 3 ของการทำ Thinning ถูกแสดงไว้ในรูปที่ 5.5

แถวที่ 1 สามารถหาจุด pixel สีดำได้ 2 จุด

แถวที่ 2 สามารถตรวจพบ pixel สีดำได้มากกว่า 2 จุด

แถวที่ 3 สามารถตรวจพบ pixel สีดำได้เพียงจุดเดียวเนื่องจากเส้นแบ่งเลนส์เป็นเส้นประ.



รูปที่ 5.5 แสดงกรณีทั้ง 3 ของการทำ Thinning

ในกรณีที่ตัวนับจำนวนจุดสีดำของแต่ละแถวมีค่าเท่ากับหนึ่งดูจากแถวที่ 3 ในรูปที่ 5.5 โปรแกรมจะทำการนับรวมเฉพาะแถวที่ตัวนับ pixel สีดำมีค่าเท่ากับหนึ่งเพื่อนำไปหาค่าเฉลี่ยของตำแหน่ง pixel สีดำของเส้นแบ่งเลนส์ แล้วจะให้ค่าตำแหน่ง pixel บนเส้นแบ่งเลนส์ที่เหลือเท่ากับ 0

กรณีที่ตัวนับของแต่ละแถวเท่ากับสอง ดูจากแถวที่ 1 ในรูปที่ 5.5 โปรแกรมจะคำนวณระยะห่างระหว่าง ตำแหน่งของ pixel ทั้งสอง ว่าใกล้เคียงกับระยะห่างจริงของเส้นแบ่งเลนส์ทั้งสองข้างหรือไม่ ถ้ามีขนาดใกล้เคียงโปรแกรมก็จะเก็บค่าทั้งสอง และค่าระยะห่างระหว่าง pixel ทั้งสองนี้ไว้ เสร็จแล้วจะเพิ่มค่าตัวนับ pixel สีดำที่มีค่าเท่ากับสอง เพื่อที่จะนำค่าต่างๆไปทำการประมวลผลข้อมูลต่อไป

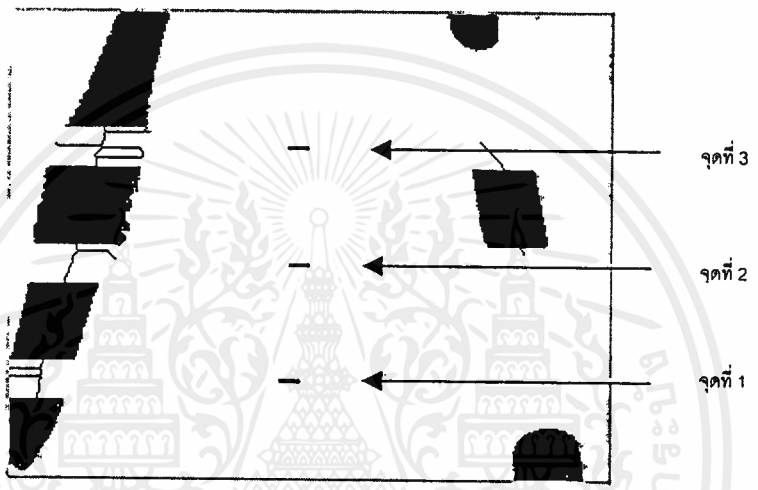
ส่วนในกรณีสุดท้าย คือตัวนับของแต่ละแถวมีค่ามากกว่าสองดูจากแถวที่ 2 ในรูปที่ 5.5 โปรแกรมจะถือว่าแถวนั้นเกิดความผิดพลาดขึ้นและโปรแกรมจะไม่นำค่าเหล่านั้นมาใช้ในการประมวลผลต่อไป

5.2.2 การหาค่าเฉลี่ยของจุดกึ่งกลางของเส้นแบ่งเลน

สามารถหาได้จาก

ในกรณีแรกเมื่อได้จุดบนเส้นแบ่งเลนส์เพียงเส้นเดียว จุดดังจุดที่ 1 ของรูปที่ 5.5 โปรแกรมจะใช้ระยะห่างระหว่างจุดบนเส้นแบ่งเลนส์ที่โปรแกรมสามารถคำนวณได้จากภาพที่สมบูรณ์ล่าสุด นำมาคำนวณร่วมกับค่าตำแหน่งของจุดใหม่ที่โปรแกรมสามารถหาได้ โดยการเปรียบเทียบจุดบนเส้นแบ่งเลนส์เดิมทั้งสอง ว่าจุดที่ได้ใหม่นี้มีตำแหน่งใกล้เคียงกับจุดใดมากที่สุดก็จะเป็นตำแหน่งของจุด

แบ่งเลนนั้น และทำการคำนวณโดยอาศัยค่าระยะห่างระหว่างจุดบนเส้นแบ่งเลนล่าสุดมาบวกกับจุดบนเส้นแบ่งเลนใหม่ที่ได้ ถ้าหากเป็นจุดที่อยู่บนเส้นแบ่งเลนเส้นซ้ายของเลน หรือลบออกจากตำแหน่งของจุดบนเส้นแบ่งเลนถ้าหากว่าจุดบนเส้นแบ่งเลนนั้นอยู่บนเส้นขวา ดังนั้นจะทำให้โปรแกรมสามารถหาตำแหน่งของจุดบนเส้นแบ่งเลน ที่โปรแกรมไม่สามารถหาได้จากภาพที่ปรากฏ ซึ่งวิธีนี้อาจจะมีความคลาดเคลื่อนบ้างเล็กน้อย



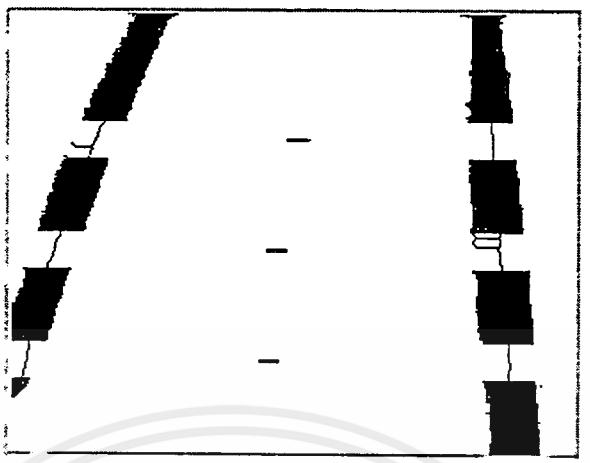
รูปที่ 5.6 รูปแสดงการหาจุดกึ่งกลางของเลนในกรณีต่างๆ

ในกรณีที่โปรแกรมสามารถหาจุดบนเส้นแบ่งเลนได้ถูกต้องทั้งสองจุดดังจุดที่ 2 และจุดที่ 3 ของรูปที่ 5.6 โปรแกรมจะทำการคำนวณระยะห่างระหว่างจุดทั้งสองจุดไว้ เพื่อที่จะนำมาใช้ในกรณีที่ไม่สามารถหาจุดทั้งสองจุดได้ และโปรแกรมจะทำการเปลี่ยนแปลงค่าระยะห่างนี้ เฉพาะในกรณีที่สามารถหาจุดสองจุดได้อย่างถูกต้องเท่านั้น

การนำเอาตำแหน่งและจำนวนของ pixel ที่หาได้จากกระบวนการข้างต้นมาหาค่าเฉลี่ย ดั้งนั้นโปรแกรมจะสามารถรู้ตำแหน่งโดยประมาณของเส้นแบ่งได้

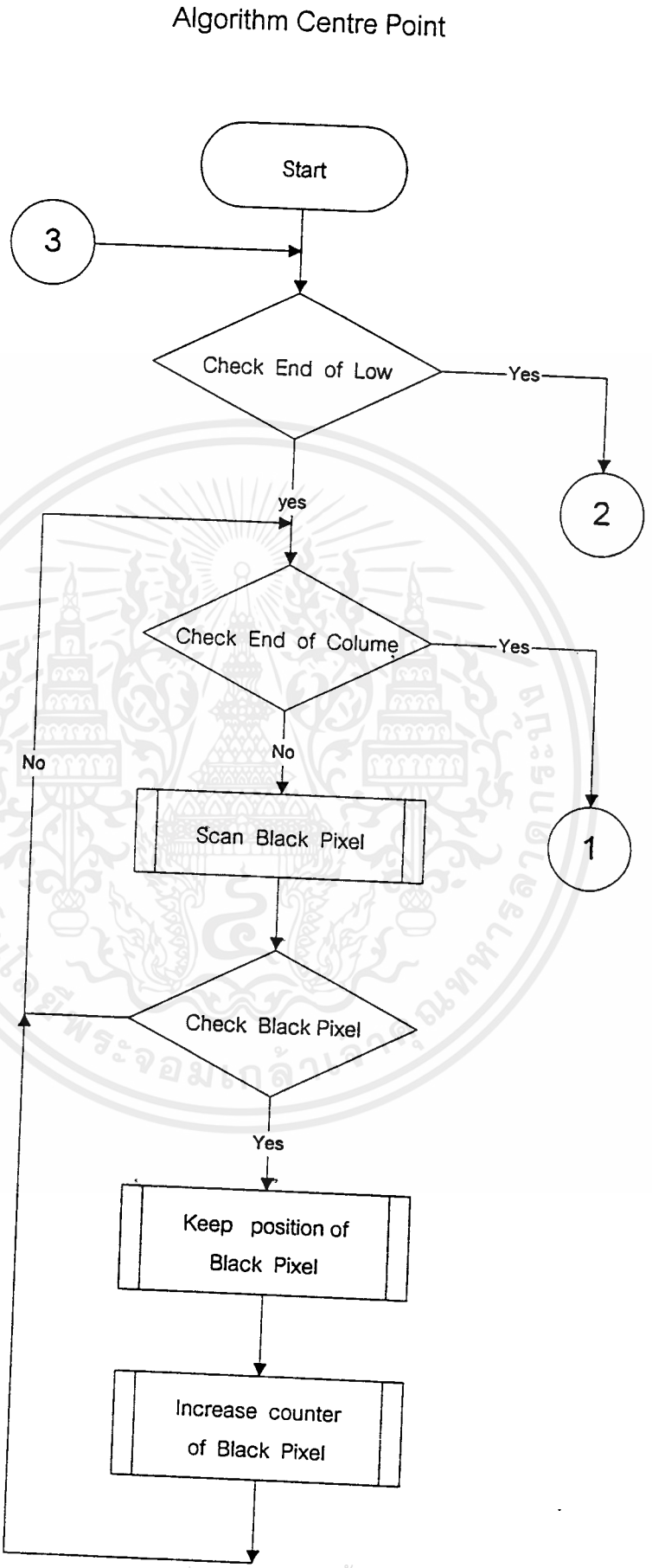
หลังจากนั้น โปรแกรมสามารถรู้ตำแหน่งของจุดกึ่งกลางของเลนขณะใดขณะหนึ่งได้จากการหาจุดกึ่งกลางระหว่างจุดที่อยู่บนเส้นแบ่งเลนทั้งสองข้าง

โปรแกรมสามารถหาจุดกึ่งกลางของเลน ณ ตำแหน่งใดก็ได้ แต่ในการทดลองนี้ถูกเซ็ตให้หาจุดกึ่งกลางของเลนเพียง 3 ตำแหน่งดังที่แสดงให้เห็นดังรูปที่ 5.7 แล้วนำทั้ง 3 ตำแหน่งที่ได้นี้มาทำการประมวลผลเพื่อที่จะนำไปใช้ในการควบคุมทิศทางของรถต่อไป

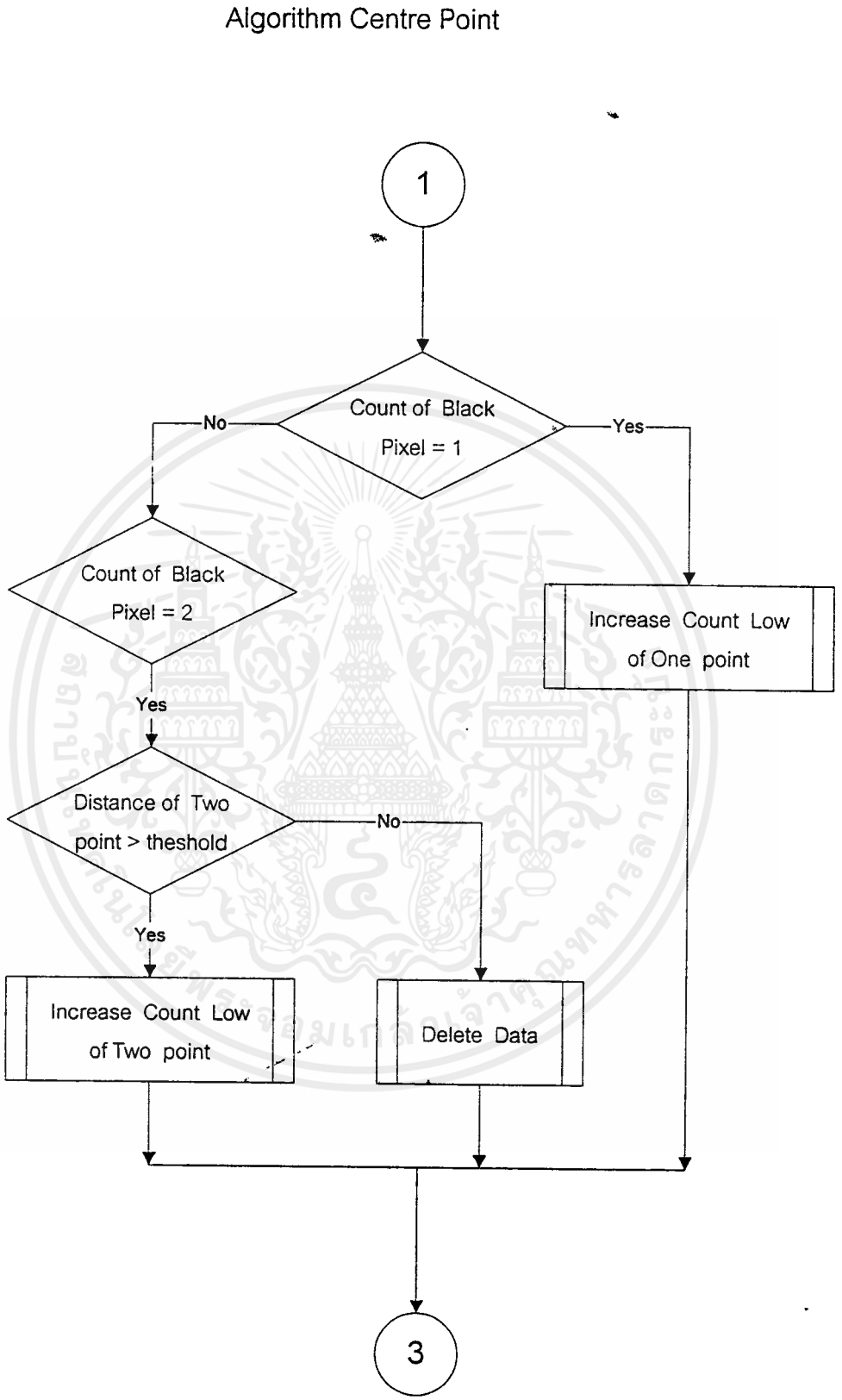


รูปที่ 5.7 แสดงการหาจุดกึ่งกลางของเลนโดยสมบูรณ์ทั้ง 3 จุดของโครงงาน

โฟล์วชาร์ต (Flow chart) ของอัลกอริทึมการหาจุดกึ่งกลางของเลนแสดงดังรูปที 5.8, 5.9, 5.10 และ 5.11

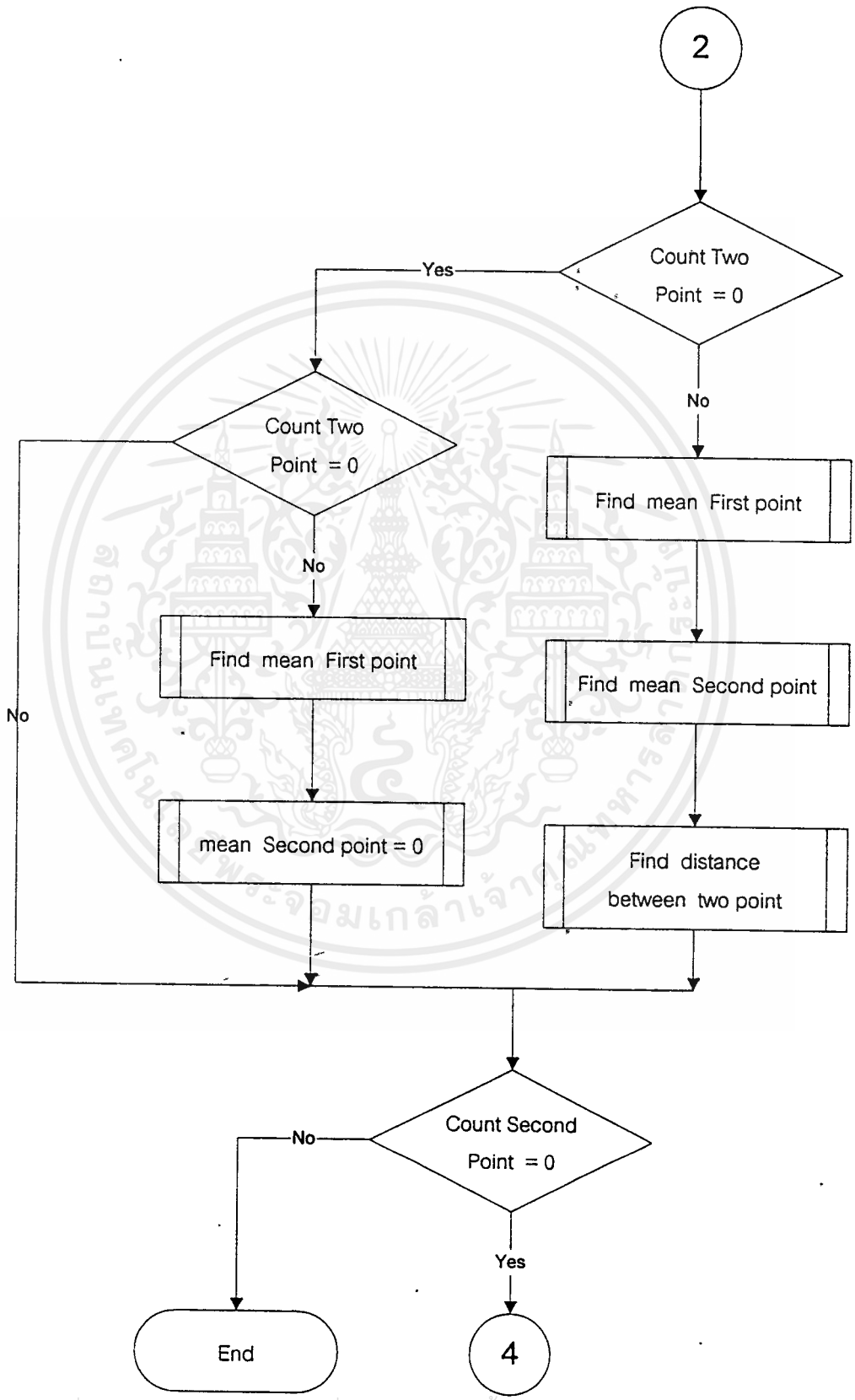


รูปที่ 5.8 แสดงโฟล์วชาร์ตของอัลกอลิทึมการหาจุดศูนย์กลางของเลน

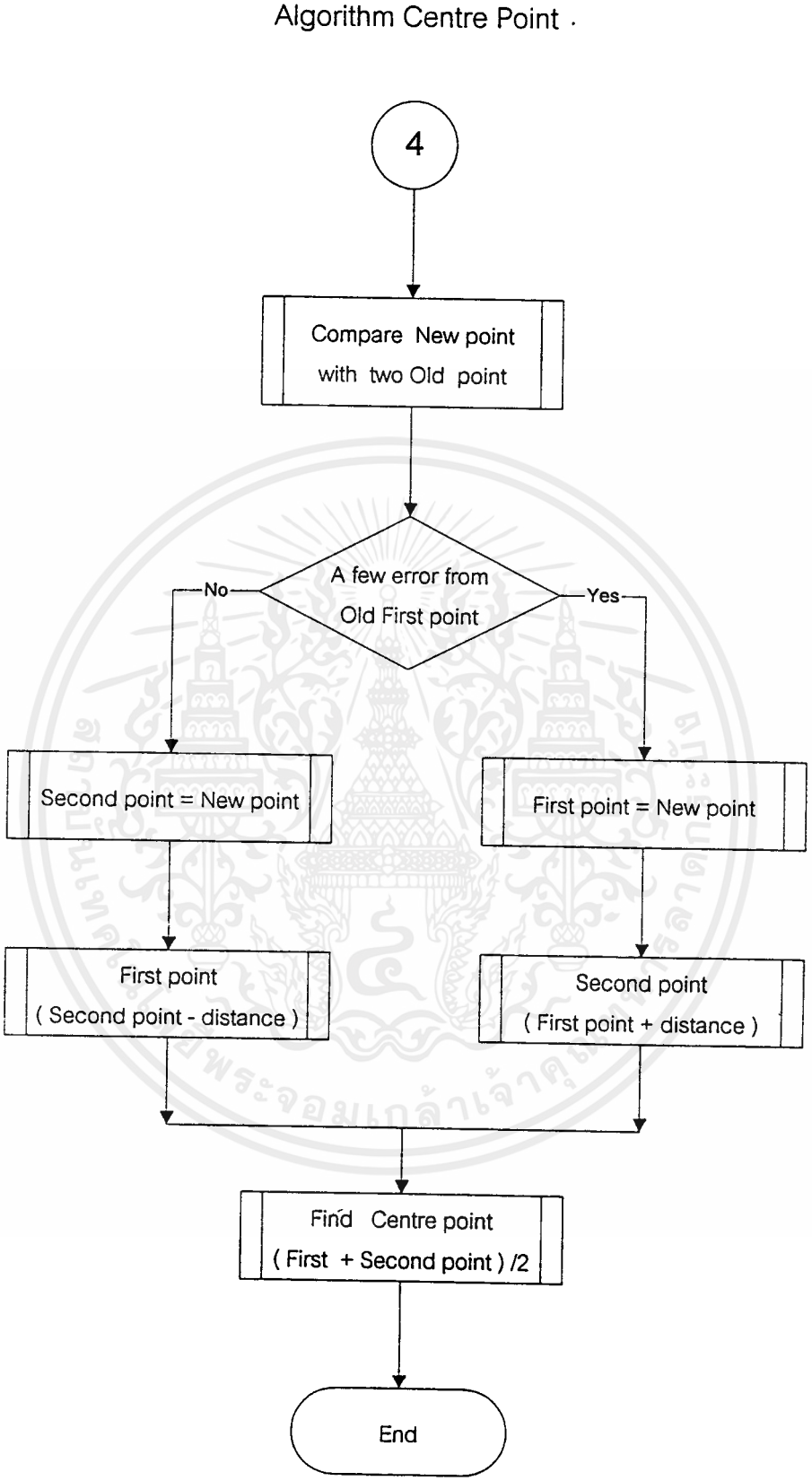


รูปที่ 5.9 แสดงโฟล์วชาร์ตของอัลกอลิทึมการหาจุดศูนย์กลางของเลน

Algorithm Centre Point



รูปที่ 5.10 แสดงโฟล์วชาร์ตของอัลกอลิทึมการหาจุดศูนย์กลางของเลน



รูปที่ 5.11 แสดงโฟล์วชาร์ตของอัลกอลิทึมการหาจุดศูนย์กลางของเลน

5.2.3 อัลกอลิทึมในการสร้างรหัสในการควบคุมทิศทาง

อัลกอลิทึมนี้จะสามารถรับรหัสการควบคุมจากที่ใดก็ได้ของโปรแกรมทั้งหมด ซึ่งแสดงว่าโปรแกรมสามารถสั่งการควบคุมทิศทางของรถขณะที่กำลังปฏิบัติงานอยู่ที่ส่วนใดของโปรแกรมก็ได้ โดยที่อัลกอลิทึมนี้จะรับค่าทิศทางและความเร็วในการเลี้ยวจากอัลกอลิทึมส่วนอื่น

โดยที่อัลกอลิทึมนี้จะถูกแบ่งประเภทของคำสั่งได้เป็น 6 แบบด้วยกันดังนี้

1. Start ซึ่งเป็นการสั่งให้ไมโครคอนโทลเลอร์อยู่ในสภาวะที่เตรียมพร้อมที่จะรับของมูลต่างๆ ที่โปรแกรมจะส่งให้ ซึ่งคำสั่ง Start นี้จะส่งค่าความเร็วที่ต่ำสุดของรถให้กับไมโครคอนโทลเลอร์ และไมโครคอนโทลเลอร์ก็จะสั่งการให้รถเคลื่อนที่ไปข้างหน้าด้วยความเร็วต่ำที่สุด

2. Stop เป็นการสั่งให้ไมโครคอนโทลเลอร์อยู่ในสภาพที่สั่งการให้รถหยุดการเคลื่อนที่ ซึ่งคำสั่ง Stop จะทำให้โปรแกรมส่งค่าความถี่ของการเลื่อนเฟสมากที่สุดไปให้ไมโครคอนโทลเลอร์ ซึ่งค่าความถี่นี้จะมีค่าสูงกว่าความถี่ที่สเต็ปมอเตอร์จะสามารถหมุนได้ ทำให้มอเตอร์อยู่ในสภาพที่หยุดนิ่ง

3. Speed เป็นการสั่งให้ไมโครคอนโทลเลอร์เพิ่มความเร็วในการหมุนสเต็ปมอเตอร์ ซึ่งมีผลทำให้สเตปปิ้งมอเตอร์ทั้งสองข้างมีความเร็วเพิ่มขึ้นและมีขนาดเท่ากัน และความเร็วนี้จะเป็นความเร็วของตัวรถ

4. Slow การสั่งงานจะตรงข้ามกันคำสั่ง Speed ซึ่งเป็นการสั่งให้ไมโครคอนโทลเลอร์ลดความเร็วในการขับสเตปปิ้งมอเตอร์ ทำให้ความเร็วของสเตปปิ้งมอเตอร์ทั้งสองข้างลดลง และมีความเร็วที่เท่ากัน และความเร็วนี้จะเป็นความเร็วของตัวรถ

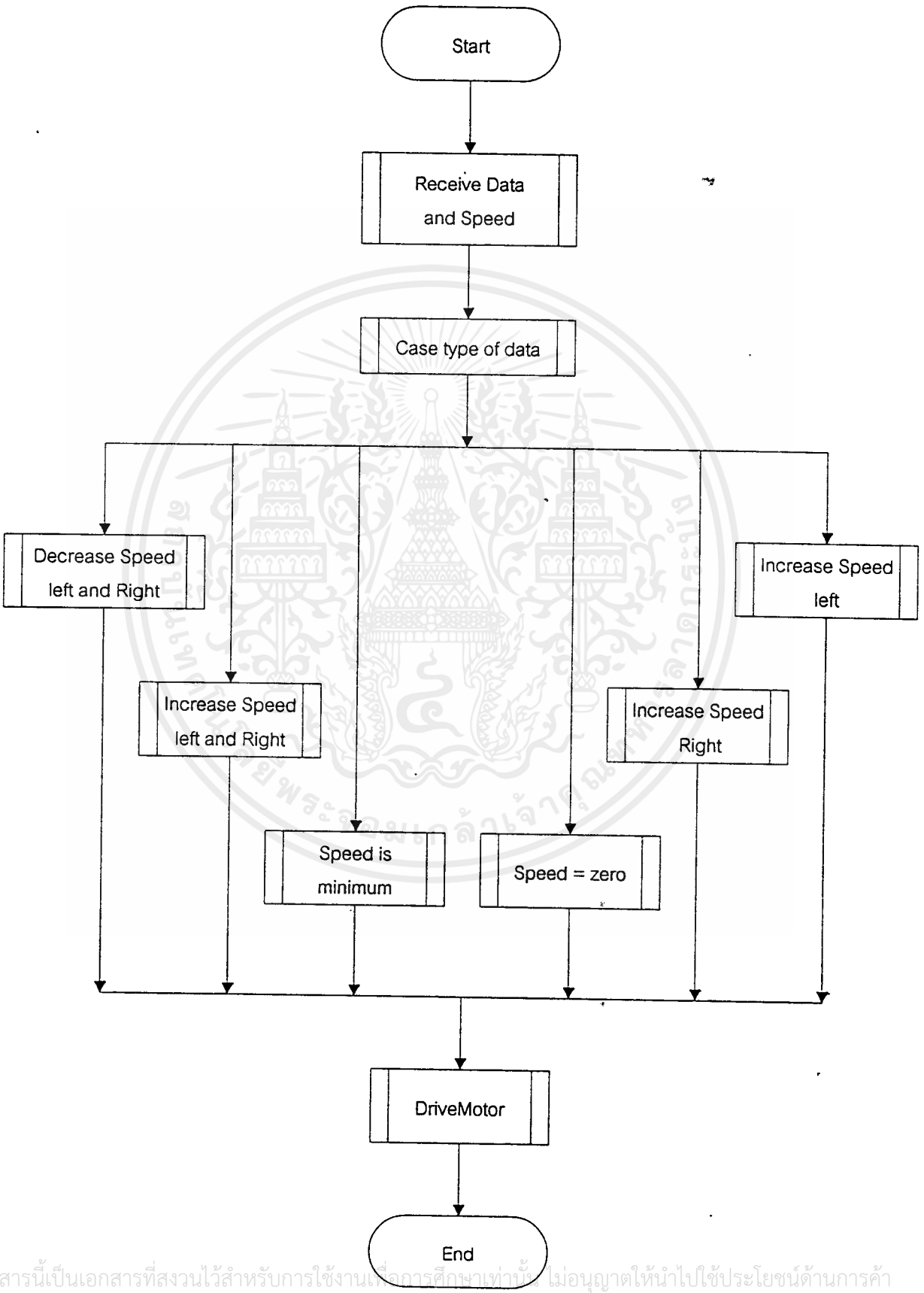
5. Turn Left เป็นการสั่งงานให้รถทำการเลี้ยวซ้าย โดยที่โปรแกรมจะสั่งให้ไมโครคอนโทลเลอร์ป้อนความเร็วให้กับสเตปปิ้งมอเตอร์แต่ละตัวด้วยค่าที่ไม่เท่ากัน โดยที่สเตปปิ้งมอเตอร์ด้านขวาของรถจะมีความเร็วเพิ่มขึ้น แต่ความเร็วของสเตปปิ้งมอเตอร์ทางด้านซ้ายจะมีความเร็วเท่ากับความเร็วของรถ ซึ่งจะมีค่าน้อยกว่าทางด้านซ้าย

6. Turn Right การสั่งงานจะตรงข้ามกับคำสั่ง Turn Left โดยความเร็วของสเตปปิ้งมอเตอร์ทางด้านซ้ายจะมีความเร็วเพิ่มขึ้น แต่สเตปปิ้งมอเตอร์ทางด้านขวาจะมีความเร็ว เท่ากับความเร็วของรถ

หลังจากโปรแกรมสามารถแปลงค่าคำสั่งเสร็จแล้ว โปรแกรมก็จะส่งข้อมูลที่ได้จากอัลกอลิทึมนี้ไปยังอัลกอลิทึมส่งข้อมูลให้ไมโครคอนโทลเลอร์ เพื่อที่จะแปลงข้อมูลต่างๆให้ไมโครคอนโทลเลอร์สามารถเข้าใจได้ แล้วจะส่งต่อให้กับไมโครคอนโทลเลอร์เพื่อที่จะใช้ในการควบคุมรถต่อไป

โฟล์วชาร์ตของอัลกอลิทึมในการสร้างรหัสในการควบคุมทิศทางแสดงดังรูปที่ 5.12

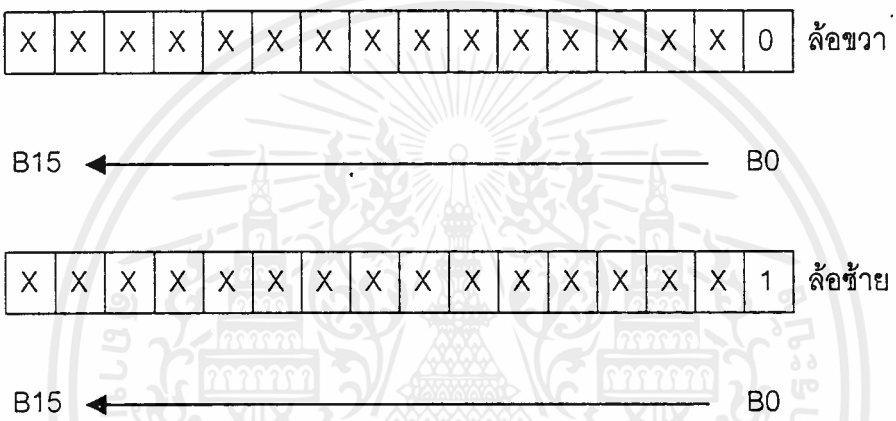
Algorithm Direction



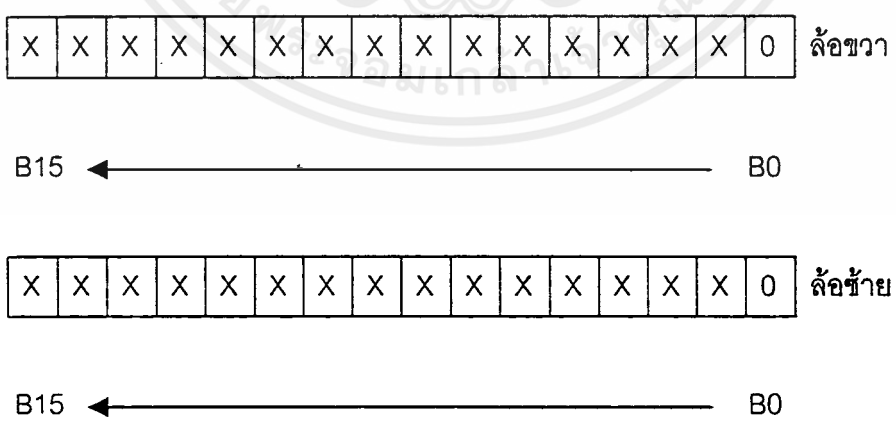
รูปที่ 5.12 แสดงโฟล์วชาร์ตของอัลกอลิทึมการสร้างรหัสในการควบคุมทิศทาง

5.2.4 อัลกอลิทึมการส่งข้อมูลให้ไมโครคอนโทลเลอร์

อัลกอลิทึมนี้จะเริ่มจากการรับข้อมูลจาก อัลกอลิทึมสร้างรหัสในการควบคุมทิศทางของรถ โดยจะรับของมูลจำนวน 4 ไบต์ แบ่งเป็นสเตปปิ้งมอเตอร์ซ้ายและขวาตัวละ 2 ไบต์ ดังแสดงในรูปที่ 5.13 ซึ่งบิต 0 จะบอกว่าเป็นข้อมูลของมอเตอร์ตัวใด ส่วนบิตที่ 1 ถึงบิตที่ 15 เป็นข้อมูลความเร็วในการหมุนของมอเตอร์ตัวนั้น



รูปที่ 5.13 แสดงข้อมูลที่ใช้ควบคุมมอเตอร์ของล้อซ้ายและขวา



รูปที่ 5.14 แสดงข้อมูลจริงที่ใช้ควบคุมมอเตอร์ของล้อซ้ายและขวา

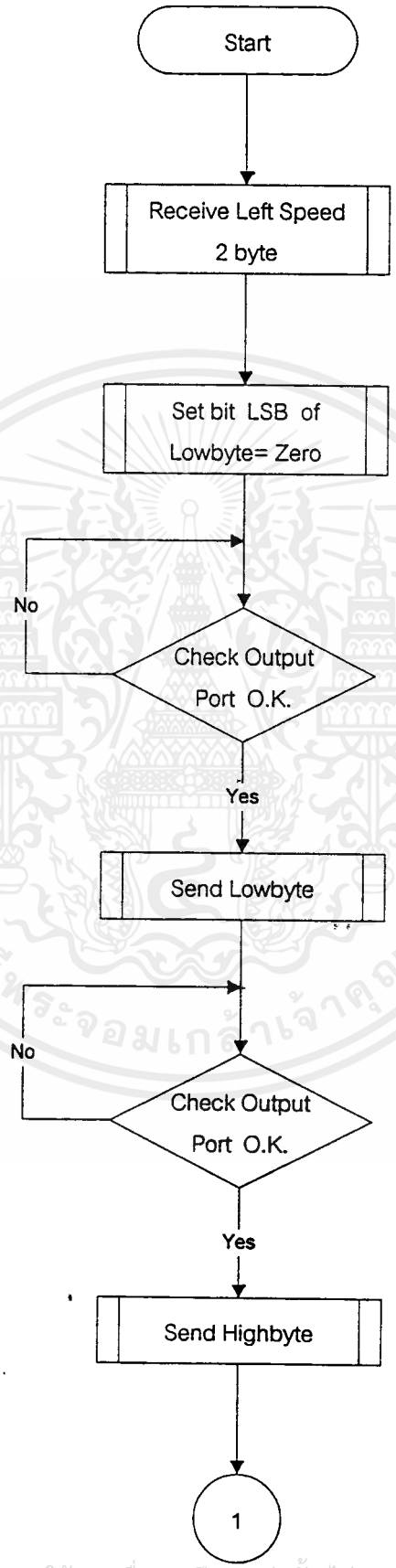
เมื่อได้รับข้อมูลเข้ามาแล้วจะทำการแปลงข้อมูลเพื่อให้ไมโครคอนโทลเลอร์สามารถเข้าใจได้ โดยจะแบ่งข้อมูล 4 ไบท์ออกเป็นการส่งครั้งละ 2 ไบท์

ก่อนที่จะมีการแปลงข้อมูล โปรแกรมจะทำการตรวจสอบค่าของข้อมูลที่จะทำการแปลงว่าเป็นค่าเดียวกับข้อมูลเดิมหรือไม่ ถ้าหากเป็นข้อมูลเดิมแล้ว โปรแกรมจะทำการข้ามขั้นตอนในการส่งค่าข้อมูลนี้ออกไป แล้วไปตรวจข้อมูลที่เหลือว่าเป็นค่าเดิมที่พึ่งส่งออกไปหรือไม่ ถ้าเป็นค่าเดิมก็จะไม่ทำการแปลงค่าและไม่ส่งข้อมูลออกไป แต่ถ้าหากว่าข้อมูลที่กำลังจะส่งออกไปเป็นข้อมูลที่ไม่ซ้ำกับข้อมูลเดิม โปรแกรมก็จะทำการแปลงค่าข้อมูลแล้วทำการส่งออกไป

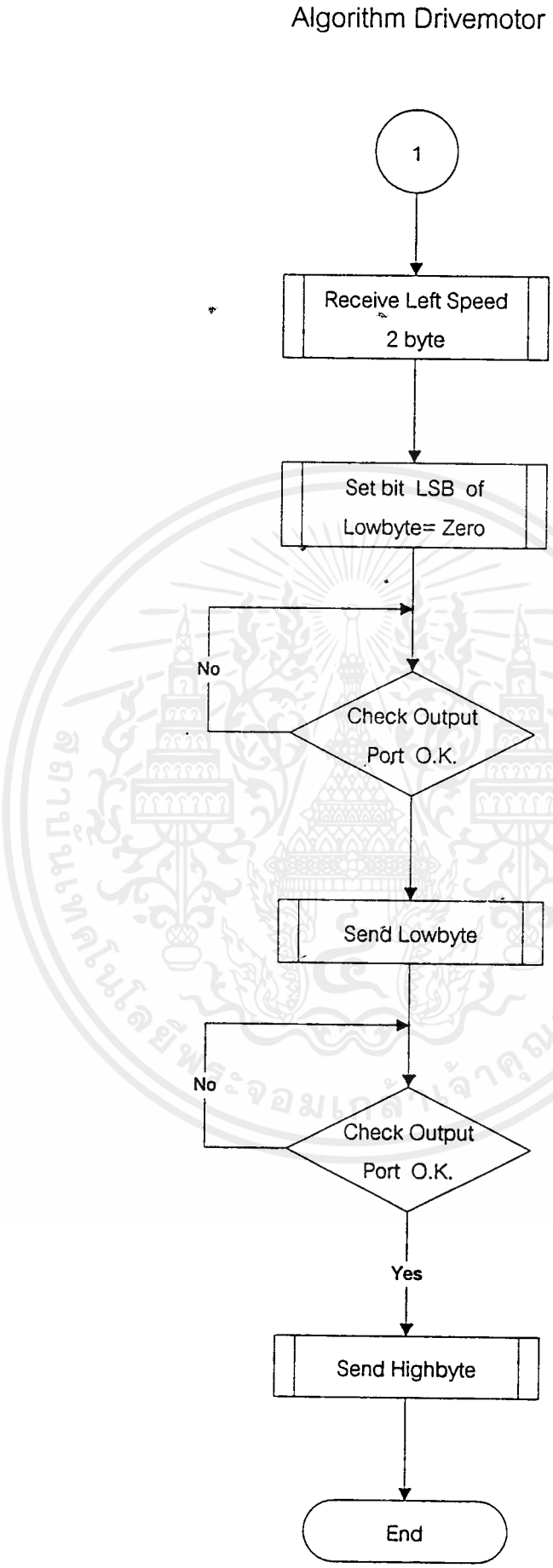
ก่อนที่จะส่งข้อมูลออกไปแต่ละค่านั้น โปรแกรมจะทำการตรวจสอบพอร์ทก่อนว่า ขณะนี้พอร์ทพร้อมที่จะส่งข้อมูลใหม่ได้หรือยัง ถ้ายังไม่พร้อมโปรแกรมก็จะทำการตรวจสอบพอร์ทใหม่จนกว่าพอร์ทจะว่าง จึงจะสามารถส่งข้อมูลออกไปได้

โฟล์วชาร์ตของอัลกอลิทึมการส่งข้อมูลให้ไมโครคอนโทลเลอร์รูปที่ 5.15 และ 5.16

Algorithm Drivemotor



รูปที่ 5.15 แสดงโฟล์วชาร์ตของอัลกอลิทึมการส่งข้อมูลให้ไมโครคอนโทลเลอร์

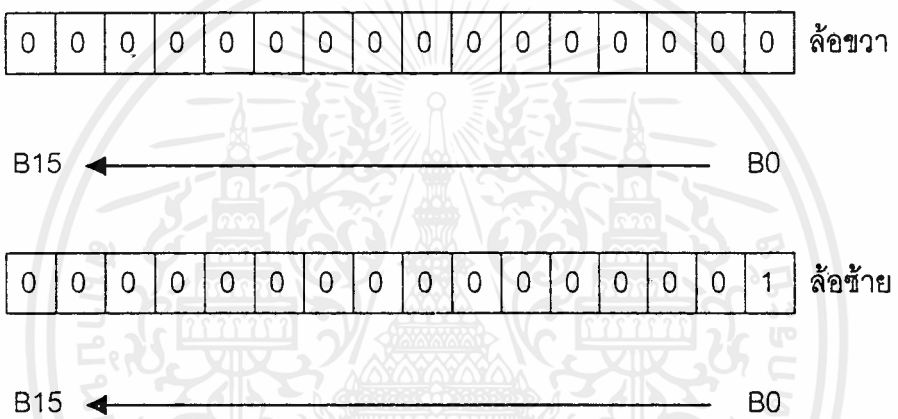


รูปที่ 5.16 แสดงโฟล์วชาร์ตของอัลกอลิทึมการส่งข้อมูลให้ไมโครคอนโทลเลอร์

5.2.5 อัลกอลิทึมการทำงานของไมโครคอนโทลเลอร์

การทำงานของไมโครคอนโทลเลอร์ เริ่มจากการเลือกโหมดการอินเตอร์รัป ในโครงงานนี้จะใช้อินเตอร์รัปทั้งหมด 3 ตัวด้วยกัน คือ อินเตอร์รัปภายนอก(External Interrupt)และ ไทม์เมอร์เคาท์เตอร์ทั้งสองตัว (Timer counter) โดยที่อินเตอร์รัปภายนอกจะใช้ในการรับของมูลที่ส่งมาจากคอมพิวเตอร์ และไทม์เมอร์เคาท์เตอร์ทั้งสองตัวจะใช้ในการ Shift phase ของสเต็ปมอเตอร์

หลังจากนั้นจะทำการเซ็ตค่าเริ่มต้นให้กับรีจิสเตอร์ต่างๆ ที่ใช้ในโปรแกรม โดยเฉพาะรีจิสเตอร์ที่ใช้ในการควบคุมสเต็ปมอเตอร์ โปรแกรมจะเซ็ตค่า 0000 เป็นค่าเริ่มต้นให้ ซึ่งทำให้สเต็ปมอเตอร์หมุนด้วยความเร็วต่ำสุด ความถี่ 62 เฮิร์ท



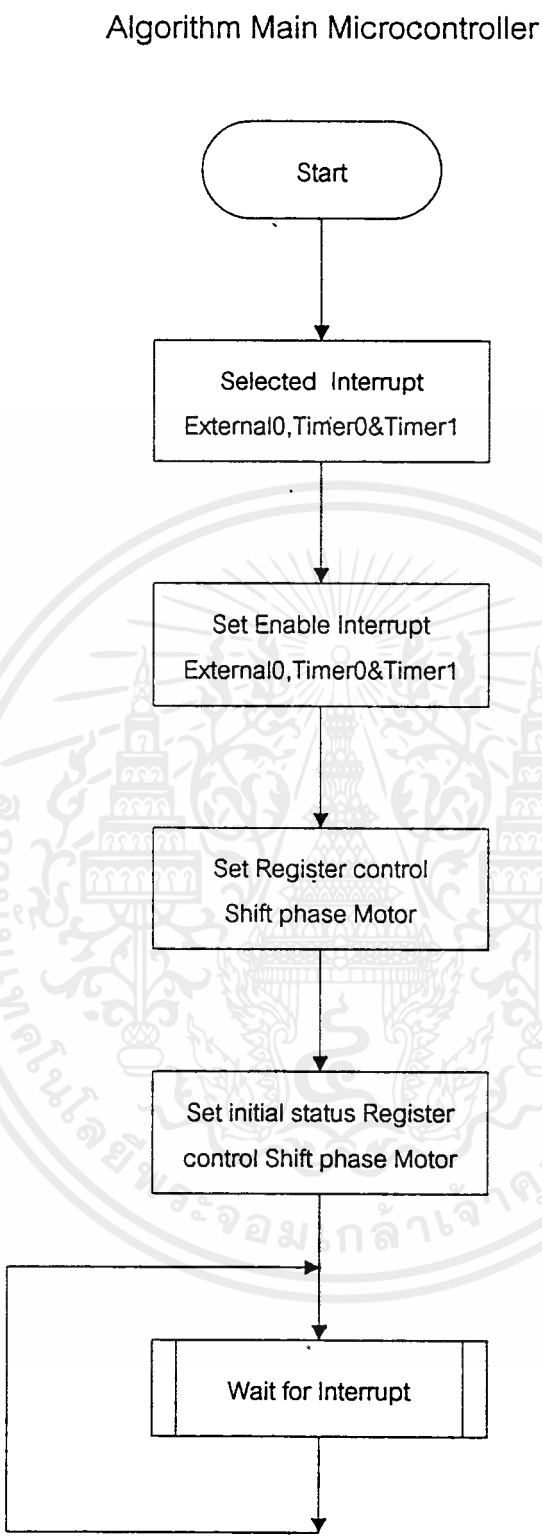
รูปที่ 5.17 ข้อมูลที่ใช้ควบคุมมอเตอร์ของล้อซ้ายและขวา เมื่อให้ค่าความเร็วเริ่มต้น 0000

โฟล์วชาร์ตของอัลกอริทึมการทำงานของไมโครคอนโทรลเลอร์แสดงดังรูปที่ 5.18

5.2.6 อัลกอริทึมการอินเตอร์รัปภายนอก

การส่งข้อมูลระหว่างคอมพิวเตอร์และไมโครคอนโทลเลอร์ จะเป็นที่จะต้องมีอัลกอลิทึมในการควบคุมการส่งข้อมูล เพื่อที่จะให้การส่งข้อมูลเป็นไปได้โดยถูกต้อง ในโครงงานนี้เราใช้โหมดการส่งข้อมูลเป็นแบบขนานจากคอมพิวเตอร์ไปยังไมโครคอนโทลเลอร์ และมีการตอบกลับจากไมโครคอนโทลเลอร์สู่คอมพิวเตอร์อีกทางหนึ่งด้วย เหตุผลที่ต้องใช้การส่งข้อมูลแบบขนาน ก็เนื่องจากว่าในการควบคุมการขับเคลื่อนของรถจำลองเราสามารถควบคุมโดยใช้คอมพิวเตอร์โดยตรง ผ่านสายแพร์จำนวน 8 เส้นได้ เพราะสเต็ปมอเตอร์แต่ละตัวต้องใช้สายควบคุมจำนวน 4 เส้น เนื่องจากในโครงงานนี้ใช้การขับสเต็ปมอเตอร์แบบ 4 เฟส หรือจะใช้คอมพิวเตอร์สั่งการผ่านไมโครคอนโทลเลอร์อีกทีหนึ่ง

สำหรับการควบคุมรถโดยผ่านไมโครคอนโทลเลอร์ที่ใช้ในโครงงานนี้จะมีอัลกอลิทึมดังต่อไปนี้



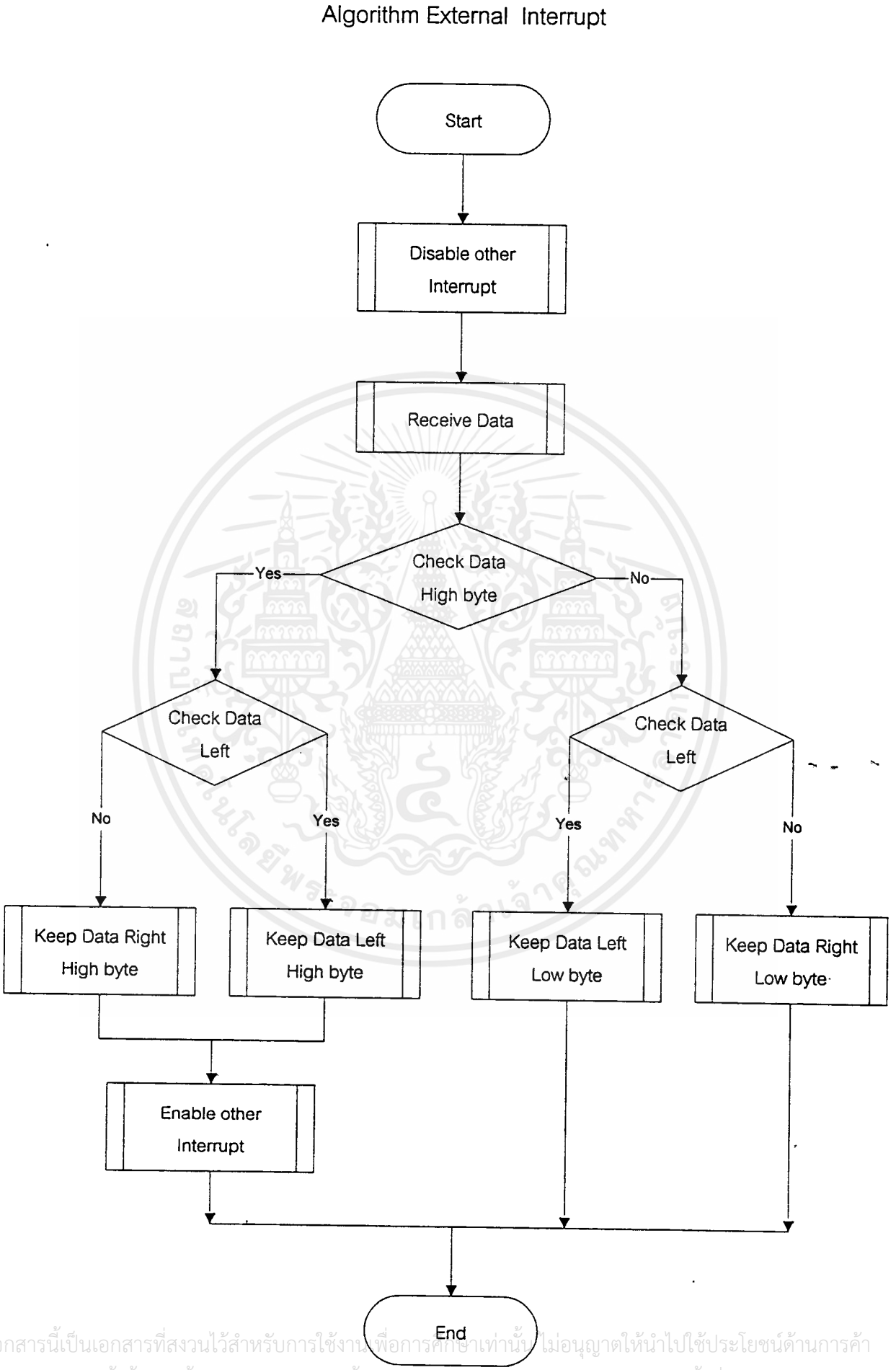
รูปที่ 5.18 แสดงโฟล์วชาร์ตของอัลกอลิทึมการทำงานของไมโครคอนโทลเลอร์

เริ่มจากคอมพิวเตอร์จะส่งข้อมูลให้ไมโครคอนโทลเลอร์ครั้งละ 8 บิต แต่ว่าข้อมูลทั้งหมดที่ส่งมาจะมีจำนวน 2 ไบท์หรือ 16 บิต ฉะนั้นเมื่อคอมพิวเตอร์ส่งขั้อมูลมาให้กับไมโครคอนโทลเลอร์ ไมโครคอนโทลเลอร์จำเป็นที่จะต้องรับข้อมูลให้ครบจำนวน 2 ไบท์เสียก่อน ถึงจะสามารถไปปฏิบัติงานอย่างอื่นได้ โดยที่คอมพิวเตอร์จะส่งไบท์ล่างของข้อมูลให้กับไมโครคอนโทลเลอร์เสียก่อน เสร็จแล้วจึงค่อยส่งไบท์บนของข้อมูลตามมาทีหลัง

เมื่อคอมพิวเตอร์ส่งข้อมูลมาครั้งแรก ไมโครคอนโทลเลอร์จะถูกทำให้เกิดอินเตอร์รัปจากภายนอกขึ้น ไมโครคอนโทลเลอร์จะ Disable อินเตอร์รัปทั้งหมด เพื่อจะไม่ให้เกิดการอินเตอร์รัปซ้อนกันได้ หลังจากที่ Disable อินเตอร์รัปแล้ว ไมโครคอนโทลเลอร์จะทำการตรวจสอบว่าข้อมูลที่เข้ามาเป็นไบท์บนหรือไบท์ล่างของข้อมูล

การตรวจสอบว่าข้อมูลไบท์ล่างที่รับเข้ามานั้นเป็นข้อมูลของมอเตอร์ตัวใด จะสามารถดูได้จากบิตท้ายสุดของข้อมูลว่าเป็นค่าอะไร และอีก 15 บิตที่เหลือ จะเป็นค่าที่บอกความเร็วของมอเตอร์จากอัลกอลิทึมนี้กำหนดให้บิตสุดท้ายถ้าเป็น 1 จะเป็นข้อมูลของมอเตอร์ขับล้อซ้าย ถ้าบิตเป็น 0 จะเป็นข้อมูลของล้อขวา แล้วข้อมูลไบท์สุดท้ายนี้จะถูกตัดทิ้งไปไม่นำมาคำนวณรวมกับค่าความเร็ว

เมื่อตรวจสอบว่าเป็นไบท์ล่างถูกต้องแล้ว ไมโครคอนโทลเลอร์ก็จะตรวจสอบอีกว่าเป็นข้อมูลของมอเตอร์ที่ใช้ขับล้อซ้ายหรือล้อขวา เมื่อตรวจสอบเสร็จแล้วข้อมูลต่างๆเสร็จแล้ว ไมโครคอนโทลเลอร์ก็จะทำการเก็บข้อมูลไว้ในรีจิสเตอร์เพื่อที่จะส่งค่าเหล่านั้นให้กับ Timer count ต่อไป หลังจากที่ไมโครคอนโทลเลอร์รับข้อมูลไบท์บน และทำการจัดเก็บข้อมูลเสร็จแล้ว ไมโครคอนโทลเลอร์จะทำการ Enable อินเตอร์รัปทั้งหมดที่ถูก Disable ไว้เมื่อตอนที่รับข้อมูลไบท์ล่างเข้ามาหลังจากนั้น ไมโครคอนโทลเลอร์ก็จะนำข้อมูลที่ได้ไปควบคุมสเต็ปมอเตอร์อีกที โฟล์วชาร์ตของอัลกอริทึมการอินเตอรัปภายนอกแสดงดังรูปที่ 5.19



รูปที่ 5.19 **แสดงโฟล์วชาร์ตของอัลกอลิทึมการอินเตอร์รัปภายนอกของไมโครคอนโทลเลอร์**

5.2.7 อัลกอลิทึมไทม์เมอร์อินเตอร์รัปเพื่อใช้ควบคุมความเร็วของสเต็ปมอเตอร์

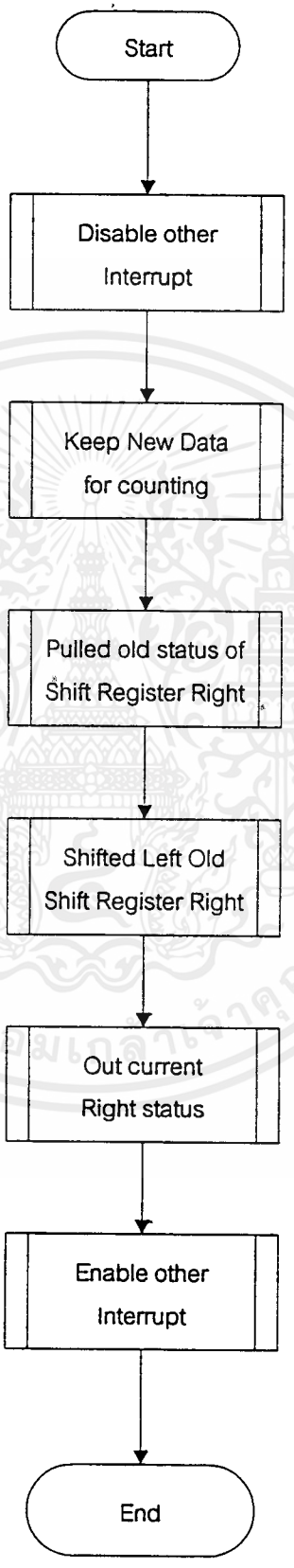
การทำงานของอัลกอลิทึมนี้เริ่มจาก เมื่อค่ารีจิสเตอร์ที่ใช้ในการนับของไทม์เมอร์อินเตอร์รัป ขนาด 16 บิต เพิ่มค่าไปเรื่อยๆจนถึงค่าสูงสุดคือ FFFF จะทำให้เกิดไทม์เมอร์อินเตอร์รัปขึ้น

เมื่อเกิดไทม์เมอร์อินเตอร์รัปแล้ว ไมโครคอนโทลเลอร์จะทำการ Disable อินเตอร์รัปทุกชนิด และทำการหยุดเพิ่มค่ารีจิสเตอร์ของไทม์เมอร์อินเตอร์รัปด้วย เพื่อที่จะไม่ให้มีการเกิดอินเตอร์รัปซ้อนกันขึ้น ซึ่งจะเป็นผลทำให้เกิดความผิดพลาดขึ้นในตัวไมโครคอนโทลเลอร์ แล้วหลังจากนั้นไมโครคอนโทลเลอร์ก็จะทำการเก็บค่าความเร็วที่ได้จากอินเตอร์รัปภายนอกมาเก็บไว้ใช้รีจิสเตอร์ของไทม์เมอร์ เพื่อที่จะใช้เป็นค่าใหม่ในการเพิ่มค่าของรีจิสเตอร์ไทม์เมอร์ เพื่อใช้ในการเปลี่ยนแปลงความเร็วของสเต็ปมอเตอร์

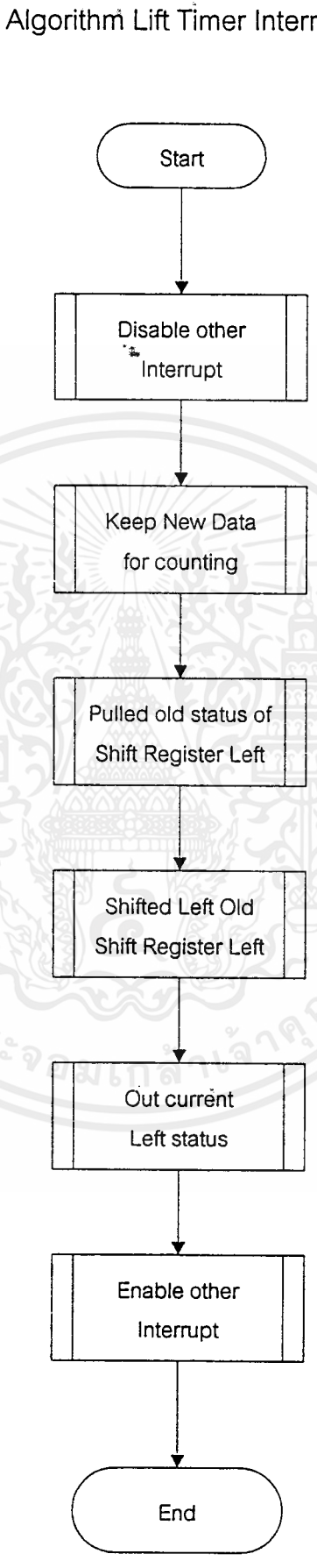
หลังจากที่นำค่าความเร็วของสเต็ปมอเตอร์เก็บไว้ในรีจิสเตอร์แล้ว ไมโครคอนโทลเลอร์ก็จะทำการ Enable อินเตอร์รัปทุกตัวที่ถูก Disable ไว้เมื่อตอนต้นของไทม์เมอร์อินเตอร์รัป และทำการเพิ่มค่ารีจิสเตอร์ของไทม์เมอร์ พร้อมทั้งรอการอินเตอร์รัปอีกครั้งเมื่อไทม์เมอร์อินเตอร์รัปเพิ่มค่าจนถึงค่าสูงสุดคือ FFFF

อัลกอลิทึมของไทม์เมอร์อินเตอร์รัปที่ใช้ควบคุมสเต็ปมอเตอร์ล้อซ้ายและขวา ของรถจะมีอัลกอลิทึมที่เหมือนกัน โดยจะแสดงได้ดังโฟล์วชาร์ตรูปที่ 5.20 และ 5.21

Algorithm Right Timer Interrupt



รูปที่ 5.20 แสดงโฟล์วชาร์ตของอัลกอลิทึมของไทม์เมอร์อินเตอร์รัปด้านขวาของไมโครคอนโทลเลอร์



รูปที่ 5.21 แสดงโฟล์วชาร์ตของอัลกอลิทึมของไทม์เมอร์อินเตอร์รัปด้านซ้ายของไมโครคอนโทลเลอร์

5.2.8 อัลกอลิทึมที่ใช้ในการควบคุมรถ

โดยสรุปอัลกอลิทึมที่ใช้ในการควบคุมรถนี้ เป็นการลอกเลียนวิธีควบคุมการขับเคลื่อนรถยนต์ของมนุษย์ โดยที่จำเป็นจะต้องทำให้โปรแกรมรู้ว่าขณะที่โปรแกรมกำลังทำงานอยู่นั้น รถที่ควบคุมกำลังอยู่ที่ตำแหน่งใดของเลนส์ ดังนั้นจึงจำเป็นที่จะต้องทำให้โปรแกรมสามารถรักษาตำแหน่งของจุดกึ่งกลางของเลนส์ให้คงที่ หรือมีความคลาดเคลื่อนน้อยที่สุดเท่าที่จะทำได้ โดยที่ในความเป็นจริง เราสามารถหาวิธีในการควบคุมรถได้หลายวิธี แต่ให้โครงงานนี้จะกล่าวถึงหลักการง่ายในการควบคุมรถ เนื่องจากสาเหตุที่ความสามารถทางด้าน Hardware ที่ยังใช้คอมพิวเตอร์ความเร็วต่ำในการประมวลผล โดยจะได้อธิบายวิธีการที่ใช้ดังต่อไปนี้

เริ่มจากการนำจุดกึ่งกลางของเลนส์ ที่สามารถคำนวณได้จากอัลกอลิทึมที่ใช้หาจุดกึ่งกลางของเลนส์ มาทำการเปรียบเทียบกับจุดที่คงที่จุดใดจุดหนึ่ง ซึ่งเรากำหนดให้เป็นจุดที่ใช้ในการอ้างอิงตำแหน่งของรถ โดยที่จุดอ้างอิงดังกล่าวนั้นจะสามารถถูกกำหนดให้เป็นจุดใดก็ได้แล้วแต่ความเหมาะสม

เมื่อทำการเปรียบเทียบระหว่างจุดอ้างอิงกับจุดกึ่งกลางของเลนส์ แล้วโปรแกรมจะทำการตรวจสอบดูว่า ขณะที่กำลังทำการตรวจสอบอยู่นี้ ตำแหน่งของกล้องกำลังจับภาพที่บริเวณใดของเลนส์ ซึ่งจะมีความสัมพันธ์ทิศทางของรถที่กำลังเคลื่อนที่

หากการเปรียบเทียบปรากฏว่าเกิดความผิดพลาดระหว่างสองจุดนั้น โปรแกรมจะทำการตรวจสอบว่าเป็นการผิดพลาดประเภทใด โดยสามารถแบ่งได้ดังนี้

- ตำแหน่งของจุดอ้างอิงมีค่ามากกว่าตำแหน่งของจุดกึ่งกลางของเลนส์ แสดงว่าตอนนี้รถกำลังเคลื่อนที่อยู่บนฝั่งขวาของเลนส์

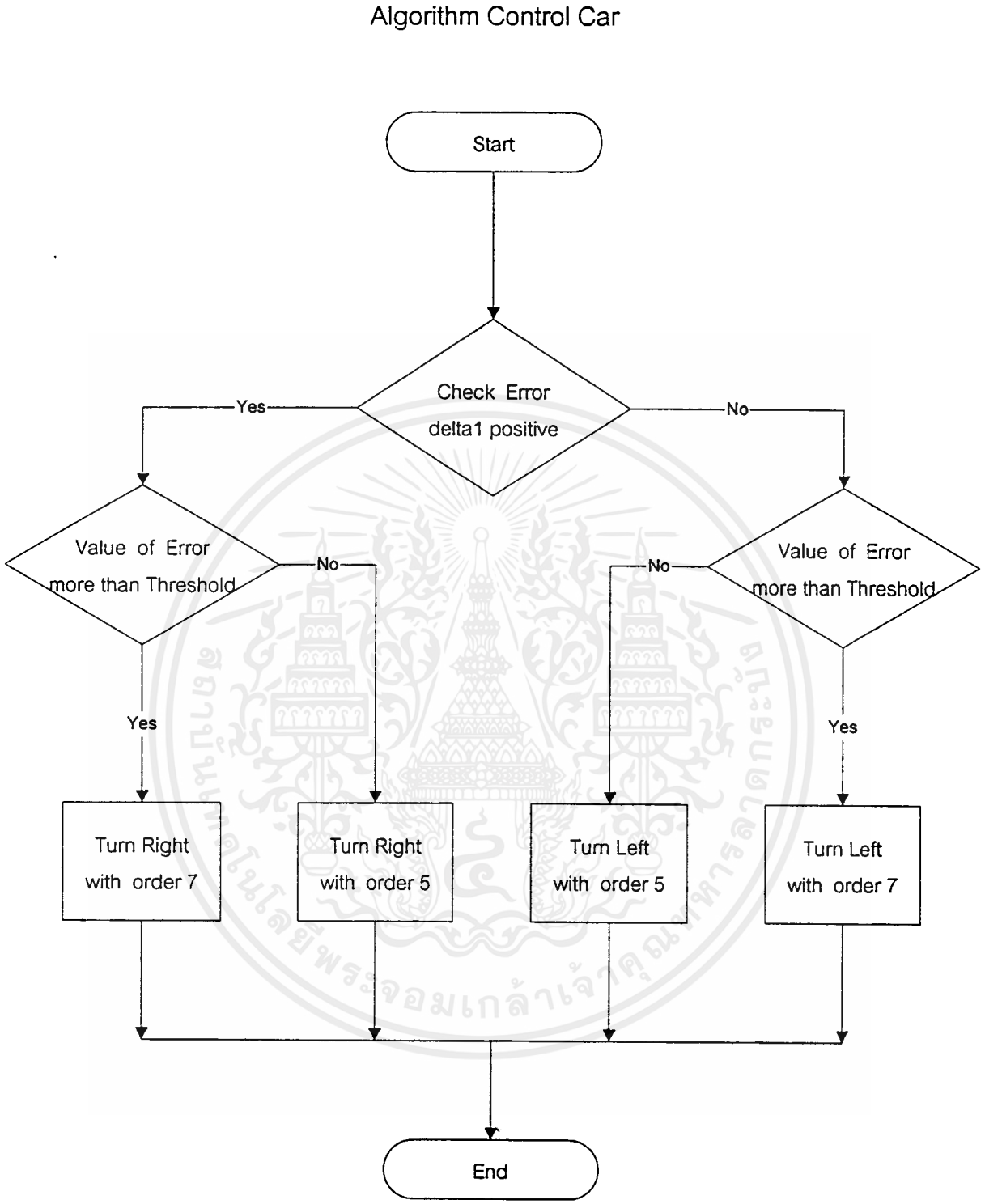
- ตำแหน่งของจุดอ้างอิงมีค่าน้อยกว่าตำแหน่งของจุดกึ่งกลางของเลนส์ แสดงว่าตอนนี้รถกำลังเคลื่อนที่อยู่บนฝั่งซ้ายของเลนส์

เมื่อโปรแกรมตรวจสอบพบว่ามีความคลาดเคลื่อนเกิดขึ้น โปรแกรมจะทำการคำนวณระยะที่คลาดเคลื่อนจากจุดกึ่งกลางของเลนส์ โดยที่เมื่อรถอยู่ทางซ้ายของเลนส์โปรแกรมจะพยายามปรับทิศทางรถให้เคลื่อนที่ไปทางขาว และในทางกลับกัน เมื่อรถอยู่ทางขาวโปรแกรมก็จะพยายามปรับให้รถเคลื่อนไปทางซ้าย

โดยค่าที่ใช้ในการปรับจะมีความสัมพันธ์กับค่าความคลาดเคลื่อนที่คำนวณได้ โปรแกรมจะแบ่งวิธีการในการควบคุมออกเป็นสองช่วง โดยจะใช้ค่า Thedhold เป็นตัวแบ่งขนาดในการควบคุม

ในโปรแกรมนี้จะใช้ค่า Thedhold เท่ากับ 8 คือผลต่างระหว่างจุดกึ่งกลางของเลนส์และจุดอ้างอิง ถ้าค่าผลต่างมีค่าน้อยกว่า 8 แสดงว่า รถอยู่ห่างจากจุดกึ่งกลางของเลนส์มีขนาดไม่มาก โปรแกรมจะใช้ตัวคูณเท่ากับ 5 ในการเปลี่ยนค่าความเร็วของล้อ เพื่อที่จะทำให้รถสามารถเลี้ยวได้

โดยที่จะทำให้การขับเคลื่อนเป็นไปอย่างต่อเนื่อง แต่ถ้าค่าผลต่างมีค่ามากกว่า 8 แสดงว่ารถอยู่ห่างจากจุดกึ่งกลางของเลนส์ อาจจะเกิดจากการเหลื่อมล้ำไปทางเลนส์ด้านขวา หรืออาจจะตกถนนไปทางด้านซ้าย โปรแกรมจะปรับทิศทางในการเคลื่อนที่ให้เลี้ยวขวาหรือซ้าย โดยที่จะใช้ตัวคูณเท่ากับ 7 เพื่อที่จะทำให้สามารถปรับความคลาดเคลื่อนให้มีขนาดลดลงได้โดยเร็ว โฟล์วชาร์ตของอัลกอริทึมการควบคุมรถแสดงดังรูปที่ 5.22

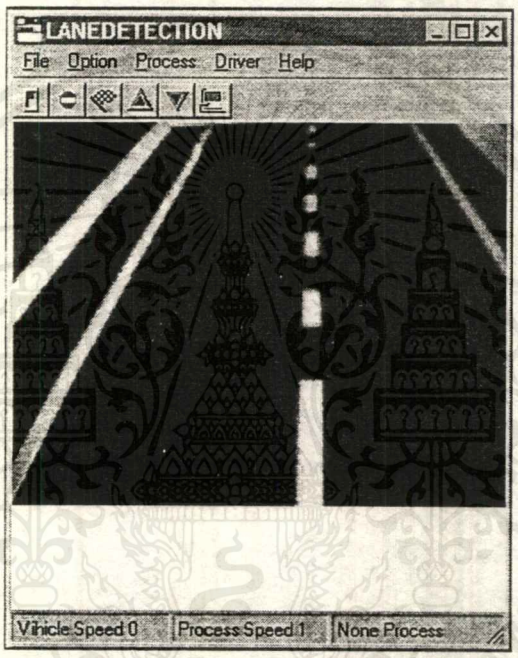


รูปที่ 5.22 แสดงโฟล์วชาร์ตของอัลกอลิทึมของการควบคุมทิศทางของรถ.

บทที่ 6
การทดลองและผลการทดลอง

6.1 รายละเอียดการทำงานของโปรแกรม Lane Detection

ในส่วนของโปรแกรม Lane Detection เมื่อเราเข้าสู่การทำงานของโปรแกรม จะปรากฏส่วนของเมนูการติดต่อระหว่างโปรแกรมกับผู้ใช้ดังรูปที่ 6.1



รูปที่ 6.1 แสดงส่วนเมนูและส่วนแสดงผลของโปรแกรม Lane Detection

โดยรายละเอียดต่าง ๆ ของเมนูการทำงานมีดังนี้

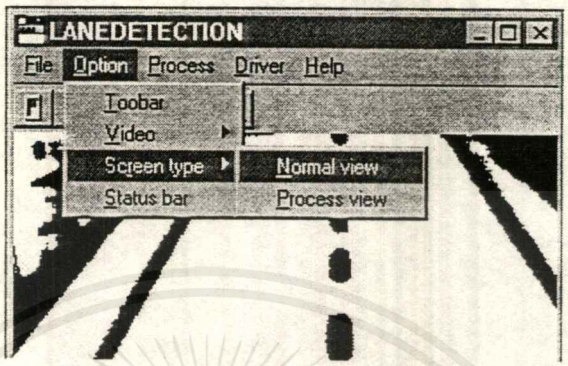
6.1.1 Option Menu จะแบ่งเป็น

1. Tool bar จะเป็นส่วนในการกำหนดการแสดงทูลบาร์ (tool bar) ว่าจะให้แสดงทูลบาร์หรือไม่แสดงทูลบาร์

2. Video จะใช้เป็นส่วนในการกำหนดค่าพารามิเตอร์ต่าง ๆ ของภาพที่ได้จากกล้องโดยจะแบ่งย่อยออกเป็น

- compression ใช้เลือกวิธีการบีบอัดข้อมูลที่ส่งมาจากกล้องเพื่อใช้แสดงผล
- source ใช้ในการปรับค่าพารามิเตอร์ของภาพ เช่น ความสว่าง , saturation, black or white level เป็นต้น

- format ใช้ในการกำหนดขนาดของภาพที่แสดง ความทึบของสีที่แสดง และความคมชัดของภาพ



รูปที่ 6.2 แสดง Option Menu

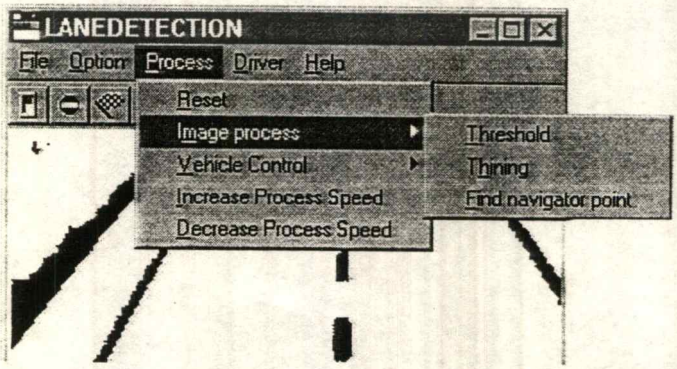
3. Screen Type ใช้กำหนดรูปแบบการแสดงผลของโปรแกรม แบ่งเป็น

- normal view จะแสดงผลการทำงานในรูปแบบของภาพที่ได้จะเป็นภาพจริง

- process view จะแสดงผลการทำงานในรูปแบบของภาพที่ได้จะเป็นภาพที่ผ่านการประมวลผล

4. Status bar ใช้ในการกำหนดว่าจะแสดงสเตตัสบาร์ (status bar) หรือไม่ โดยในส่วนของสเตตัสบาร์จะใช้บอกสถานะของโปรแกรม คือค่าความเร็วของรถ ความเร็วในการประมวลผลภาพ และสถานะการทำงานของโปรแกรม

6.1.2 Process Menu



รูปที่ 6.3 แสดง Process Menu

ชุดคำสั่งในโปรเซสเมนูมีดังนี้

1. Reset ใช้ในการรีเซ็ทโปรแกรม คือเริ่มต้นการทำงานของโปรแกรมใหม่ทั้งหมด

2. Image process เป็นส่วนที่ใช้ทำการประมวลผลภาพในรูปแบบต่าง ๆ ประกอบไปด้วยฟังก์ชันต่าง ๆ ดังนี้

- Threshold เมื่อใช้ฟังก์ชันนี้จะทำให้ภาพที่ได้เป็นภาพขาว – ดำ โดย pixel จะแสดงค่าสีขาวหรือสีดำ จะขึ้นอยู่กับค่า Threshold compare ที่เราได้เซ็ทไว้ในตัวโปรแกรม

- Thinning เมื่อใช้ฟังก์ชันนี้ประมวลผล ภาพที่ได้จะเป็นภาพของเส้นขอบของเลนถนน (โดยในโปรแกรมของเรานี้จะทำการทินนิ่งเพียงบางส่วนของภาพเท่านั้น)

- Find navigator point เมื่อใช้ฟังก์ชันนี้แล้วเราจะได้ภาพของเลนถนนที่ผ่านฟังก์ชันทินนิ่งและจะปรากฏจุดกึ่งกลางของเลนถนนขึ้น 3 จุดซึ่งเราจะใช้จุดทั้งสามจุดนี้ในการตรวจจับเลนถนน

3. Vehicle Control เป็นส่วนที่ใช้ในการควบคุมตัวรถ โดยแบ่งเป็น

- Start ใช้ในการทำให้รถเริ่มเคลื่อนที่เมื่อรถกรณีที่รถหยุดการเคลื่อนที่อยู่

- Stop ใช้ในการทำให้รถหยุดการเคลื่อนที่

- Speed down ใช้ในการลดความเร็วในการเคลื่อนที่ของรถ

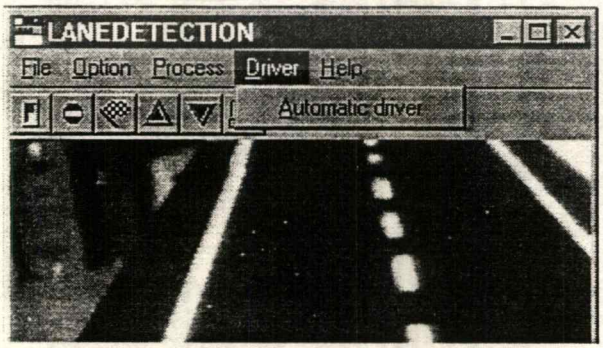
- Speed up ใช้ในการเพิ่มความเร็วในการเคลื่อนที่ของรถ

4. Increase Process Speed ใช้ในการเพิ่มความเร็วในการประมวลผลของภาพ

5. Decrease Process Speed ใช้ในการลดความเร็วในการประมวลผลของภาพ

6.1.3 Driver Menu ประกอบด้วย

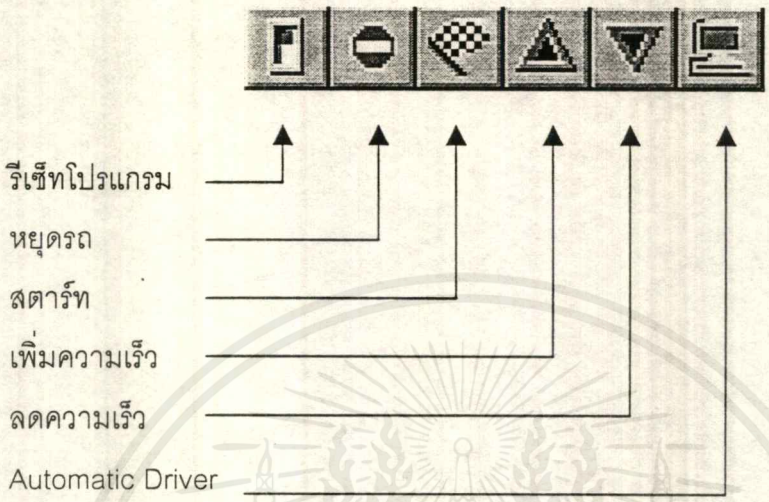
Automatic driver เป็นฟังก์ชันที่ใช้ในการตรวจจับเลนถนนของรถ และทำการควบคุมการเคลื่อนที่ของรถโดยคอมพิวเตอร์



รูปที่ 6.4 แสดง Driver Menu

6.1.4 ส่วนของ Tool bar

ส่วนต่าง ๆ ของ tool bar จะแสดงได้ดังรูปที่ 6.5



รูปที่ 6.5 แสดงส่วน Tool bar ของโปรแกรม

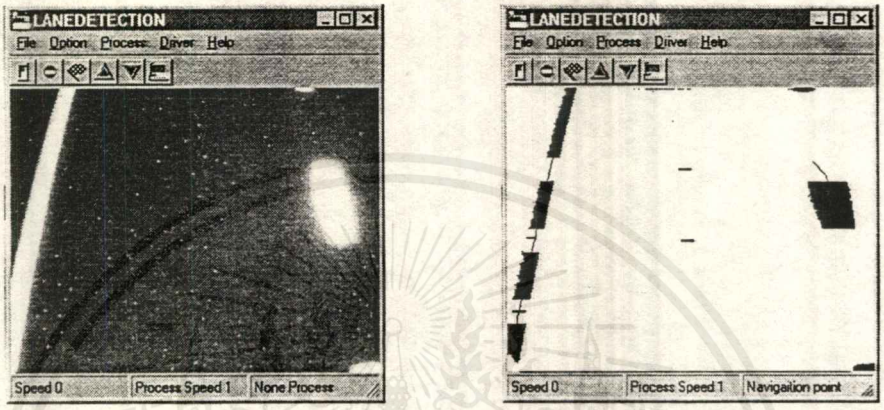
6.2 การใช้งานโปรแกรม Lane Detection

6.2.1 การตรวจจับเลนถนนโดยคอมพิวเตอร์

ในส่วนนี้จะแนะนำถึงการใช้งานโปรแกรม Lane Detection เพื่อทำการควบคุมการเคลื่อนที่ของรถ โดยเมื่อเราเริ่มต้นเรียกโปรแกรม Lane Detection หน้าจอจะปรากฏส่วนการแสดงผลของโปรแกรมดังรูปที่ 6.6

รูปที่ 6.6 แสดงผลเมื่อเริ่มโปรแกรม

ภาพที่กล้องรับเข้ามาแสดงผลจะเป็นภาพจริงของเลนถนน เราจะต้องทำการเซ็ทลักษณะของภาพที่จะให้แสดงผลขณะที่รถวิ่งโดยเลือกที่ Option Menu ในส่วนของ Screen type ซึ่งจะมีให้เลือก 2 ลักษณะคือ normal view และ process view จะได้ผลของการทำงานดังรูปที่ 6.7



(ก) (ข)

รูปที่ 6.7 แสดงผลการทำงานของคำสั่ง Screen type

(ก) การแสดงผลแบบ Normal view

(ข) การแสดงผลแบบ Process view

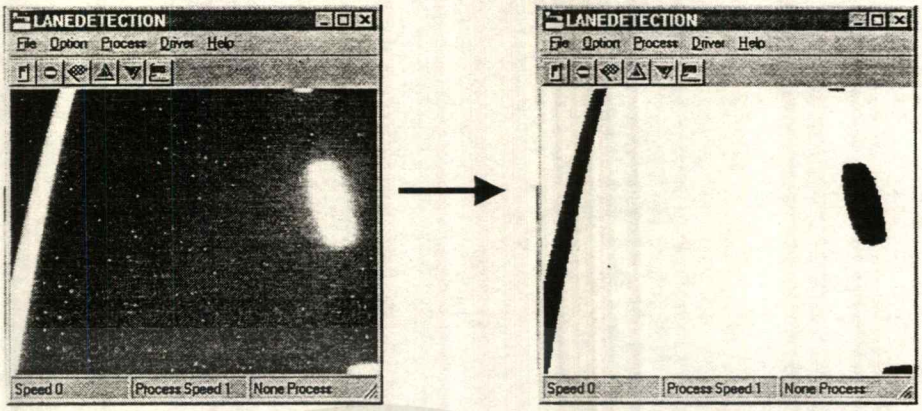
หลังจากนั้นให้ทำการเริ่มขับเคลื่อนรถโดยก่อนรถเคลื่อนที่ให้เราเลือก Automatic driver ที่ Driver Menu โปรแกรมจะทำการตรวจจับเลนถนน การขับเคลื่อนรถให้เลือกที่ Process Menu ในส่วนของ Vehicle control คำสั่ง Start หรือปุ่มสตาร์ทที่ส่วนทูลบาร์ รถจะเริ่มเคลื่อนที่ เมื่อรถเคลื่อนที่เราสามารถเพิ่มหรือลดความเร็วของรถได้โดยการเลือกที่ Vehicle control คำสั่ง speed up หรือ speed down หรืออาจใช้ปุ่มเพิ่มหรือลดความเร็วที่ทูลบาร์ก็ได้

เราจะสามารถเปลี่ยนภาพที่แสดงผลเป็นแบบภาพจริงหรือภาพที่ใช้ประมวลผลได้โดยเลือกที่ Option Menu ส่วนของ Screen type

6.2.2 การประมวลผลภาพ

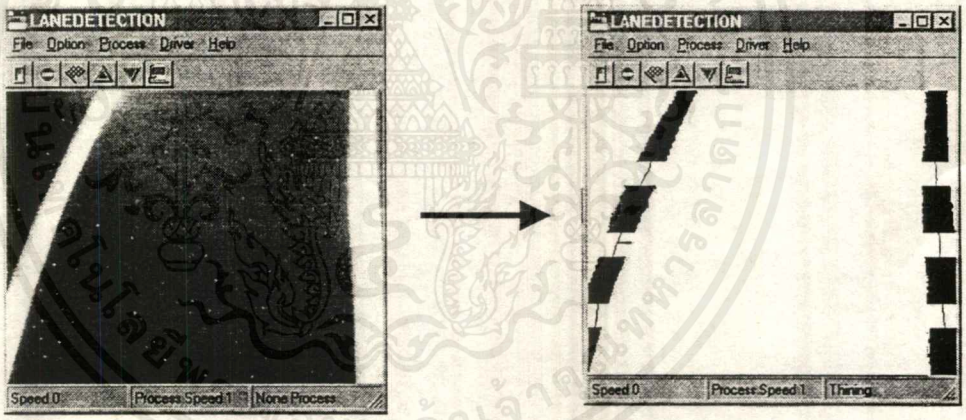
ในส่วนนี้จะกล่าวถึงการนำเอาภาพที่ได้จากกล้องมาทำการประมวลผลโดยวิธีการต่าง ๆ (การทำงานในส่วนนี้เราจะทำเมื่อรถอยู่ในสถานะที่ไม่เคลื่อนที่) ผลลัพธ์ที่ได้จะแสดงทางส่วนแสดงผลของโปรแกรมโดยจะแบ่งเป็น

- Threshold เมื่อใช้คำสั่งนี้จะทำให้ภาพที่ผ่านการประมวลผลจะเป็นภาพที่มีเพียง 2 ระดับ คือขาวกับดำ ดังรูปที่ 6.8



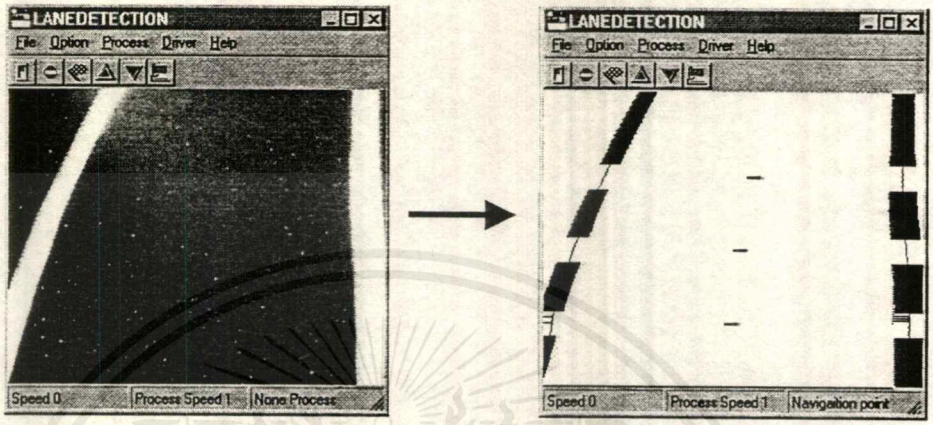
รูปที่ 6.8 แสดงการทำ Threshold

- Thinning เมื่อใช้คำสั่งนี้จะทำให้ภาพของเลนถนนที่รับจากกล้องจะถูกทำให้เหลือเพียงเส้นขอบของถนนเพียงอย่างเดียว ดังเช่นในรูปที่ 6.9



รูปที่ 6.9 แสดงผลของการ Thinning

- Find navigator point เมื่อใช้คำสั่งนี้เราจะได้จุดกึ่งกลางของเลนถนน 3 จุด ซึ่งในส่วนของการตรวจจับเลนถนนโดยคอมพิวเตอร์ คอมพิวเตอร์จะใช้จุดทั้งสามนี้ในการประมวลผล ผลของคำสั่งนี้แสดงได้ดังรูปที่ 6.10



รูปที่ 6.10 แสดงการใช้คำสั่ง Find navigator point

บทที่ 7
สรุปและวิจารณ์

การตรวจจับเลนถนนโดยคอมพิวเตอร์ในโครงงานนี้จะประกอบด้วย 2 ส่วนหลัก ๆ คือส่วนของวงจรควบคุมการเคลื่อนที่ของรถ และส่วนของโปรแกรมการตรวจจับเลนถนน (Lane Detection) โดยเราจะนำเอาทั้ง 2 ส่วนมาประยุกต์ใช้ร่วมกันเพื่อทำการขับเคลื่อนรถโดยอัตโนมัติ

ในส่วนของวงจรการควบคุมการเคลื่อนที่ของรถ เรายังสามารถแบ่งออกได้เป็น กล้องที่ใช้จับภาพเลนถนน ส่วนวงจรไดรฟ์สเต็ปเปอร์มอเตอร์ และส่วนของไมโครคอนโทรลเลอร์

ในส่วนของโปรแกรมการตรวจจับเลนถนนจะเป็นส่วนที่ผู้ใช้งานจะทำการติดต่อกับตัวรถโดยจะมีส่วนการแสดงผล คือภาพที่จับได้จากกล้องที่ติดที่ตัวรถ และส่วนของการควบคุมการเคลื่อนที่ของรถ

การควบคุมรถจะเริ่มที่กล้องจะจับภาพของเลนถนน และนำภาพที่ได้ไปผ่านการประมวลผล โดยการใช้ฟังก์ชัน Threshold ทำการแปลงภาพที่ได้เป็นภาพที่มีเพียงสองระดับคือ ขาวกับดำ หลังจากนั้นจะนำเอาภาพนี้ไปทำการ Thinning และจะได้ภาพของเส้นขอบของถนนเพื่อหาจุดกึ่งกลางของเลนถนน และนำจุดกึ่งกลางเหล่านี้ไปพิจารณาว่าภาพของเลนถนนที่ได้เป็นทางตรงหรือทางโค้ง เพื่อที่จะทำการควบคุมทิศทางการเคลื่อนที่ของรถได้

จากการวิจัยในโครงงานนี้เพื่อศึกษาถึงความเป็นไปได้ในการนำไปใช้งานจริงเราจะยังพบว่ายังมีข้อจำกัดอยู่อีกหลายประการที่จะเป็นอุปสรรคในการนำไปใช้งานจริง คือ ในเรื่องความไว และความละเอียดของกล้อง ซึ่งกล้องที่ใช้ในการวิจัยจะยังมีความไวที่ต่ำ รวมทั้งความไวในการประมวลผลของคอมพิวเตอร์ ทำให้ในงานวิจัยความเร็วของรถที่สามารถแล่นได้จะประมาณ 520 เมตรต่อชั่วโมง ทำให้การที่จะนำไปใช้งานจริงจะยังต้องปรับปรุงจุดด้อยในทั้งสองเรื่องที่กล่าวมา

แนวทางในการพัฒนางานวิจัย

ในงานวิจัยนี้ยังต้องมีการปรับปรุงและแก้ไขข้อจำกัดบางประการที่ยังมีอยู่ดังที่ได้กล่าวไว้ข้างต้นเพื่อที่จะเพิ่มประสิทธิภาพของการทำงานของงานวิจัยที่จะสามารถนำไปประยุกต์ใช้งานจริงในชีวิตประจำวัน เช่นในเรื่องของตัวกล้องที่ใช้รับภาพ อาจมีการปรับปรุงโดยการใช้กล้องที่มีประสิทธิภาพที่ดีขึ้น คือ มีความไวในการรับ - ส่งข้อมูลมากขึ้น มีความละเอียดของภาพที่จับได้มากขึ้น ในส่วนของการประมวลผลข้อมูลภาพที่รับได้อาจมีการปรับปรุงโดยการใช้ตัวประมวลผลที่มีประสิทธิภาพมากขึ้น หรืออาจปรับปรุงโดยการใช้ตัวประมวลผลที่สามารถติดตั้งได้บนตัวรถทั้งหมดเพื่อลดปัญหาเรื่องการเชื่อมต่อและเรื่องความยาวของสายส่งข้อมูลที่มีจำกัดได้

ในส่วนของโปรแกรมยังสามารถพัฒนาให้มีการตรวจจับเลนถนนในกรณีที่ภาพเลนถนนที่จับได้ไม่มีเส้นแบ่งเลนกลางถนนอาจใช้การปรับมุมกล้องให้กล้องจับภาพเลนถนนทั้งสองเลนแทนและหาขอบเขตของเลนถนนทั้งหมดแทนเลนเดียว หรือในกรณีที่เส้นแบ่งเลนหายไปเกินกว่าที่โปรแกรมจะยอมรับได้อาจให้มีการเตือนผู้ขับขี่และมีการปรับเปลี่ยนระบบบังคับเป็นแบบขับขี่ด้วยคนแทน และอาจจะพัฒนาให้มีการตรวจจับทางแยกของถนน หรือสัญญาณต่าง ๆ ที่ปรากฏบริเวณถนนเพื่อความสมบูรณ์ของการควบคุมการขับเคลื่อนรถโดยอัตโนมัติ

เอกสารอ้างอิง

1. Scott Stanfield with Palph Arveson, " VISUAL C++ HOW – TO " Waite Group Press ™
2. อนิรุต ลิ่วหาทอง , " การเขียนโปรแกรมบนวินโดวส์ด้วย Microsoft C++ ", ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2539
3. ธานินทร์ ถาวรศาสนวงศ์ , ทินกร ดุ๊ก , " การอินเทอร์เฟส IBM PC ", ฟิสิกส์เซ็นเตอร์ การพิมพ์
4. พงเทพ ชุติวิฑูร์ชัย , มงคล สุขาภิรมณ์ , วิชัย วัชรินทร์พร, " Stepping Motor Drive ", KMIT'L, 1991
5. สุเจตน์ จันทรังษ์, " ไมโครคอนโทรลเลอร์ซีพเดี่ยว 8051, มหาวิทยาลัยเทคโนโลยีมหานคร
6. R.C. Gonzalez and P.Wintz, " Digital Image Processing, Addison Wesley, 1995