ระบบควบคุมการติดตามวัตถุอัตโนมัติโดยใช้ข้อมูลภาพ

Automatic control system of object following by using image data

โดย

นายธีรวัฒน์ มณีวงศ์

นายวิเชียร เชิดชู

นายสุทธิพงศ์ ศรีปทุมานุรักษ์

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมการวัดคุมทางอุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ ปีการศึกษา 1/2538

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขาวิชา วิศวกรรมการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง

ระบบควบคุมการติดตามวัตถุอัตโนมัติโดยใช้ข้อมูลภาพ

Automatic control system of object following by using image data

ผู้จัดทำ

1. นายธีรวัฒน์ มณีวงศ์ 36012102
2. นายวิเชียร เชิดชู 36012116
3. นายสุทธิพงศ์ ศรีปทุมานุรักษ์ 36012125

..... อาจารย์ที่ปรึกษา

(อาจารย์ ทรงชัย วีระทวีมาศ)

ระบบควบคุมการติดตามวัตถุอัตโนมัติโดยใช้ข้อมูลภาพ

Automatic control system of object following by using image data

โดย	นายธีรวัฒน์ มณีวงศ์	รหัส 36012102
	นายวิเชียร เชิดชู	รหัส 36012116
	นายสุทธิพงศ์ ศรีปทุมานุรักษ์	รหัส 36012125

อาจารย์ที่ปรึกษา อาจารย์ทรงชัย วีระทวีมาศ

บทคัดย่อ

การนำเอาระบบคอมพิวเตอร์มาประยุกต์ใช้งานในปัจจุบันนี้มีมากมายหลายด้าน การนำเอาคอมพิวเตอร์มาใช้ในระบบควบคุมก็เป็นอีกรูปแบบหนึ่งของการประยุกต์ใช้งาน "ระบบควบคุมการติดตามวัตถุอัตโนมัติโดยใช้ข้อมูลภาพ" ก็เป็นอีกวิธีหนึ่งในการนำเอาคอมพิวเตอร์มาใช้ในการควบคุม ซึ่งเป็นการนำข้อมูลภาพมาแปลงเป็นข้อมูลดิจิตอล จากนั้นก็ส่งข้อมูลภาพนี้ไปประมวลผลที่คอมพิวเตอร์ และนำสัญญาณที่ประมวลผลเสร็จแล้ว กลับมาควบคุมสเตปปิ้งมอเตอร์ที่ติดอยู่กับกล้องวิดีโอให้หมุนเคลื่อนที่ตามวัตถุนั้น โดยกล้องวิดีโอจะเป็นตัวจับภาพและส่งสัญญาณภาพไปเข้าวงจรแปลงข้อมูลให้เป็นดิจิตอล แล้วส่งข้อมูลไปให้คอมพิวเตอร์ประมวลผล

Abstract

Computer application system has a many system which use computer to control system and applied to the other elses. "**automatic control system of object following by using image data**" is the one of many application which also used computer to control the system by used image data invert to digital for processing by cpu of PC. The data which already processed will send to reference and control motor of vedio camera for movement. The vedio camera will be image tracker and send image signal to ADC (Analog to digital convertor) for the next processing

กิตติกรรมประกาศ

ขอขอบคุณอาจารย์ทรงชัย วีระทวีมาศ ที่ให้โอกาสได้มาศึกษาและทำโครงงานชิ้นนี้ ซึ่งในตอนแรกที่จะมาทำโครงงานนี้ ผู้ทำไม่เคยรู้เรื่องนี้มาก่อนเลย แต่อาจารย์ทรงชัย วีระทวีมาศ ก็ยังให้โอกาสทำและให้คำปรึกษาทุกอย่างเป็นอย่างดี จนงานนนี้สำเร็จเป็นที่น่าพอใจตรงตามจุดประสงค์ และสุดท้ายก็ขอขอบพระคุณ บิดา มารดา พี่ ที่ทำให้ผู้ทำโครงงานชิ้นนี้ได้เข้าศึกษาในสถาบันแห่งนี้ จนได้ความรู้และสำเร็จศึกษา

สารบัญ

หน้า

หน้า

บทที่ 1
บทนำ

ปริญญานิพนธ์นี้ มีชื่อว่า "**ระบบควบคุมการติดตามวัตถุอัตโนมัติโดยใช้ข้อมูลภาพ**" ซึ่งจะเหมือนกับโทรทัศน์วงจรปิด ที่มีไว้ตรวจจับการเปลี่ยนแปลงหรือการเคลื่อนที่ของวัตถุในบริเวณที่กำหนด ส่วนประกอบที่สำคัญก็คือกล้องวิดีโอหรือกล้องโทรทัศน์ จะนำมาต่อกับเครื่องโทรทัศน์ เพื่อจะเป็นตัวแสดงผลดูการเปลี่ยนแปลงที่เกิดขึ้น แต่สำหรับปริญญานิพนธ์นี้จะมีความแตกต่างจากที่กล่าวมา คือ ตัวแสดงผลจะแสดงบนจอคอมพิวเตอร์และตัวกล้องสามารถหมุนตามการเคลื่อนที่ของวัตถุได้ คอมพิวเตอร์ไม่เพียงแต่เป็นตัวแสดงผลยังเป็นตัวควบคุมหลักในการหมุนของกล้อง

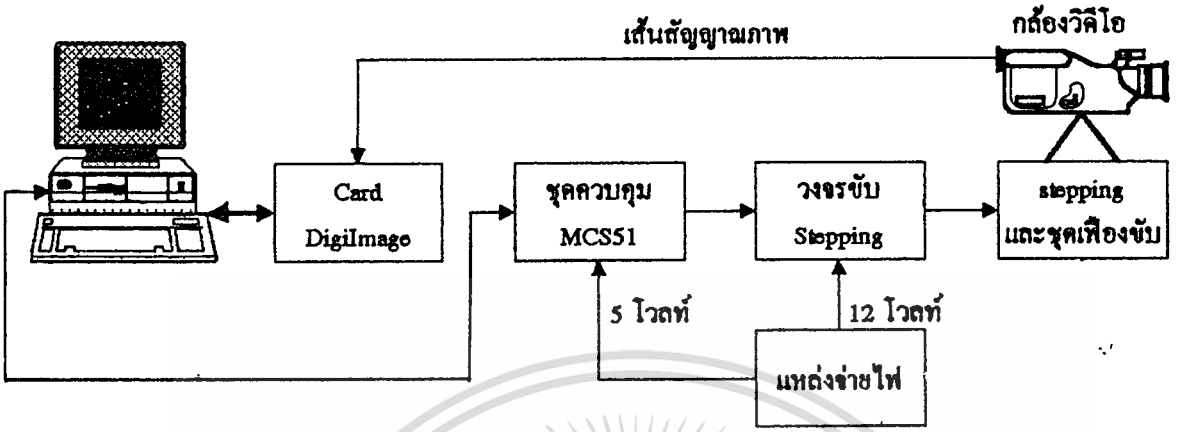
ปริญญานิพนธ์นี้ได้ปรับปรุงและพัฒนามาจากปริญญานิพนธ์เก่าที่ได้ทำเรื่องนี้ไว้ก่อนแล้ว ซึ่งของเก่าสามารถทำงานควบคุมให้กล้องหมุนตามได้แค่ภาพที่เป็นจุดดำบนฉากที่มีสีพื้นเป็นสีขาว แต่ที่ได้พัฒนาและแก้ไขขึ้นมาใหม่ในปริญญานิพนธ์ฉบับนี้ กล้องสามารถหมุนตามวัตถุในสภาพแวดล้อมใดก็ได้ ในบทนี้จะกล่าวถึงการทำงานของเครื่องที่พัฒนาขึ้นอย่างคร่าว ๆ พอสังเขป

1.1 ขอบเขตโครงงาน

1. กล้องสามารถถ่ายภาพมาแสดงผลบนจอคอมพิวเตอร์เป็น REAL TIME ได้
2. กล้องสามารถหมุนตามภาพวัตถุที่มีฉากและวัตถุรูปร่างแบบใดก็ได้
3. บริเวณที่ติดตั้งกล้องต้องมีการตรวจจับวัตถุเพียงสิ่งเดียว
4. แสงในบริเวณที่กำหนดหรือแสงที่ตกกระทบวัตถุต้องไม่มีการเปลี่ยนแปลงหรือเปลี่ยนแปลงน้อยมาก
5. วัตถุที่ต้องการตรวจจับต้องไม่เคลื่อนที่เร็วเกินไป

1.2 ขั้นตอนการทำงานของเครื่องที่ทำขึ้น

รูปที่ 1.1 เป็นบล็อกการต่อใช้งานและขั้นตอนการทำงานของเครื่องที่พัฒนาขึ้น ส่วนควบคุมหลัก ๆ มีอยู่สองส่วน คือ คอมพิวเตอร์กับชุดควบคุมไมโครคอนโทรลเลอร์ ในตระกูล MCS-51 เบอร์ 8031 โดยทั้งส่วนจะต้องทำงานสัมพันธ์กัน ในบทนี้จะกล่าวพอเป็นสังเขป การทำงานอย่างละเอียดจะกล่าวในบทอื่นต่อไป



รูปที่ 1.1 การต่อใช้งานของเครื่องที่พัฒนาขึ้น

ชุดควบคุมคอมพิวเตอร์ การทำงานในบล็อกนี้เป็นส่วนที่ใช้ในการประมวลผลการเปรียบเทียบภาพที่เปลี่ยนแปลงตามการเคลื่อนที่ของวัตถุ คอมพิวเตอร์จะรับข้อมูลจาก Card DigiIMAGE เป็นข้อมูลภาพที่ได้จากการถ่ายภาพของกล้องโทรทัศน์ Card DigiIMAGE ที่ว่านี้ ทำหน้าที่ควบคุมการถ่ายภาพของกล้องโทรทัศน์และแปลงข้อมูลจากอะนาล็อกเป็นดิจิตอลให้กับคอมพิวเตอร์ ข้อมูลที่อ่านจาก Card DigiIMAGE จะอ่านมาสองครั้ง ครั้งแรกเป็นข้อมูลของภาพต้นแบบ(reference) ข้อมูลครั้งที่สองเป็นข้อมูลที่กล้องถ่ายต่อหลังภาพแรก จากนั้นนำข้อมูลภาพทั้งสองมาเปรียบเทียบกันได้เป็นข้อมูลภาพใหม่ขึ้น ถ้าข้อมูลภาพใหม่ที่ได้มีจุดดำปรากฏอยู่ แสดงว่าข้อมูลภาพทั้งสองแตกต่างกัน แต่ถ้าภาพใหม่เป็นสีขาวหมด แสดงว่าภาพทั้งสองเหมือนกัน

เมื่อเป็นเช่นนี้คอมพิวเตอร์จะสั่งให้กล้องโทรทัศน์ถ่ายภาพข้อมูลใหม่มา นำข้อมูลภาพที่อ่านเข้ามาใหม่ไปเปรียบเทียบกับข้อมูลภาพต้นแบบ(reference)เหมือนกับตอนแรก ถ้าภาพที่ถ่ายมาใหม่ยังไม่เหมือนกับภาพต้นแบบอีก คอมพิวเตอร์จะสั่งให้ถ่ายภาพเข้ามาเปรียบเทียบอีก ทำเช่นนี้ไปเรื่อย ๆ จนกว่าภาพจะไม่เหมือนกัน จากนั้นก็จะส่งสัญญาณไปให้ไมโครคอนโทรลเลอร์เพื่อควบคุมให้สเตปปิ้งมอเตอร์หมุนตามวัตถุนั้น การส่งสัญญาณจะส่งไปทาง RS-232

Card DigiIMAGE เป็น Card ที่ใช้เสียบลง Expansion Slot ของคอมพิวเตอร์ ทำหน้าที่ควบคุมการถ่ายภาพ แปลงสัญญาณภาพให้เป็นข้อมูลดิจิตอลและเก็บข้อมูลนั้นไว้ รอการอ่านไปใช้งานจากคอมพิวเตอร์ โดย Card นี้จะมีวงจรการถ่ายภาพ วงจรแปลงสัญญาณอะนาล็อกเป็นดิจิตอลและมีหน่วยความจำสำหรับเก็บข้อมูลที่ได้จากการแปลงสัญญาณภาพ Card นี้สามารถควบคุมการถ่ายภาพได้ทั้งภาพสีและภาพขาวดำ ขึ้นอยู่กับกล้องโทรทัศน์และการสั่งงานจากคอมพิวเตอร์ แต่ที่ใช้กับปริญญานิพนธ์นี้เป็นแบบขาวดำ

ชุดควบคุม MCS-51 มีหน้าที่รับสัญญาณควบคุมจากคอมพิวเตอร์ โดยผ่านทาง RS-232 ข้อมูลที่รับมาจะมีสองค่า คือ ทิศทางการหมุนและจำนวนพัลซ์ที่จะต้องป้อนให้กับสเตปปิ้งมอเตอร์ จากนั้นชุดควบคุม MCS-51 จะกำเนิดพัลซ์ที่มีทิศทางตามข้อมูลที่คอมพิวเตอร์บอก ป้อนให้กับชุดขับสเตปปิ้งมอเตอร์ เพื่อให้มอเตอร์หมุนตามต้องการ

ชุดขับสเตปปิ้งมอเตอร์ เป็นวงจรอิเล็กทรอนิกส์ ทำหน้าที่ขยายสัญญาณและขับกระแสไฟฟ้าให้มีค่าสูงขึ้นเพียงพอกับความต้องการของสเตปปิ้งมอเตอร์ ชุดขับนี้จะใช้ทรานซิสเตอร์สองตัวต่อคาริงตันกัน มีสี่ชุดสำหรับขับสเตปปิ้งมอเตอร์สี่เฟส รายละเอียดของวงจรจะกล่าวในบทต่อไป

กล้องโทรทัศน์ กล้องที่ใช้เป็นกล้องขนาดเล็ก เพื่อความสะดวกในการควบคุม เป็นกล้องชนิดถ่ายภาพขาวดำ เหตุผลที่เลือกเป็นกล้องขาวดำเพราะจะทำให้คอมพิวเตอร์ประมวลผลได้เร็วขึ้น

1.3 เนื้อหาของแต่ละบท

บทที่ 1 เป็นการกล่าวถึงขอบเขตโครงงานวัตถุประสงค์และขั้นตอนการทำงานของเครื่องที่พัฒนาขึ้นพอสังเขป

บทที่ 2 เป็นทฤษฎีการประมวลผลภาพ เช่น การแทนข้อมูลภาพด้วยเมตริกซ์ ความเข้มของแสงที่ตกกระทบลงบนวัตถุแทนด้วยความเข้มระดับภาพเทาและการแปลงภาพจากหลายระดับให้เป็นภาพสองระดับ

บทที่ 3 คุณสมบัติของ Card DigiIMAGE การนำไปใช้งานและตัวอย่างการเขียนโปรแกรมควบคุม

บทที่ 4 กล่าวถึงทฤษฎีและวิธีการควบคุมสเตปปิ้งมอเตอร์แบบต่าง ๆ

บทที่ 5 การรับส่งข้อมูลทางพอร์ทอนุกรม RS-232 ของคอมพิวเตอร์และของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 และตัวอย่างการเขียนโปรแกรมโดยใช้ภาษา C

บทที่ 6 ขั้นตอนการทำงานของเครื่องที่พัฒนาขึ้นโดยละเอียด การทดลองและสรุปผล

ภาคผนวก ก. รายละเอียดของโปรแกรมหลัก เขียนโดยภาษา C

ภาคผนวก ข. รายละเอียดของโปรแกรมย่อยที่นำมาใช้ร่วมกับโปรแกรมหลัก เขียนโดยภาษาแอสแซมบลี้

ภาคผนวก ค. รายละเอียดของโปรแกรมภาษาแอสแซมบลี้ที่ใช้กับ MCS-51

หนังสืออ้างอิง

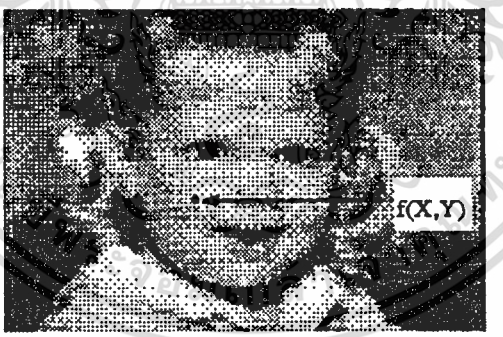
บทที่ 2

การประมวลผลภาพเบื้องต้น

รูปภาพที่แสดงตามจอคอมพิวเตอร์ที่เห็นเป็นภาพเหมือนของจริง ไม่ว่าจะเป็นรูปคน สัตว์ ต้นไม้หรือรูปภาพอื่นๆเกิดจากการนำภาพจริงมาแปลงให้อยู่ในรูปของสัญญาณดิจิตอลแล้วนำไปแสดงบนจอคอมพิวเตอร์ ข้อมูลภาพนี้อาจจะได้มาจากกล้อง วีดิโอ เครื่องสแกนภาพหรืออุปกรณ์อื่นๆ ที่ทำได้จากนั้นนำมาผ่านวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล (A/D Converter) ข้อมูลที่ได้จากการแปลงนี้จะนำไปประมวลผลต่อไป

2.1 การแทนภาพในระบบดิจิตอล

ภาพที่ได้จากการแปลงของวงจร A/D Converter จะนำไปแสดงผลเป็นลักษณะภาพ 2 มิติ ซึ่งสามารถอธิบายได้โดยการแทนด้วยระนาบแกน 2 แกน อักษรตัว X จะแทนด้วยแกนนอนและแทน Y ด้วยแกนในแนวตั้ง ณ จุดต่างๆในระนาบจะแทนด้วยจุดภาพ มีค่าความเข้มของภาพที่เรียกว่า ระดับเทา (Gray Level) หรืออาจจะแทนด้วยจุดสีต่างๆที่เรามองเห็น หากกล่าวในเชิงคณิตศาสตร์ก็คือ ที่จุดพิกัด (X,Y) ใด ๆ จะมีค่าความเข้มที่เขียนเป็นฟังก์ชันได้ว่า f(X,Y) ดังรูปที่ 2.1



รูปที่ 2.1 แสดงภาพเป็นฟังก์ชั่นคณิตศาสตร์

ถ้าจะมองในแง่คณิตศาสตร์อีกแบบจะมองในลักษณะของเมตริกซ์ดังรูปที่ 2.2 ขนาดของเมตริกซ์จะแทนได้ด้วยขนาดของภาพ ตำแหน่งสมาชิกต่างๆ ของเมตริกซ์จะแทนจุดภาพต่างๆ ซึ่งจะเรียกจุดนี้ว่า Picture element หรือ Pixel

$$\begin{bmatrix} f(0,0) & f(0,1) & .... & f(0,N-1) \\ f(1,0) & f(1,1) & .... & f(1,N-1) \\ : & : & : & : \\ f(N-1,0) & f(N-1,1) & .... & f(N-1,N-1) \end{bmatrix}$$

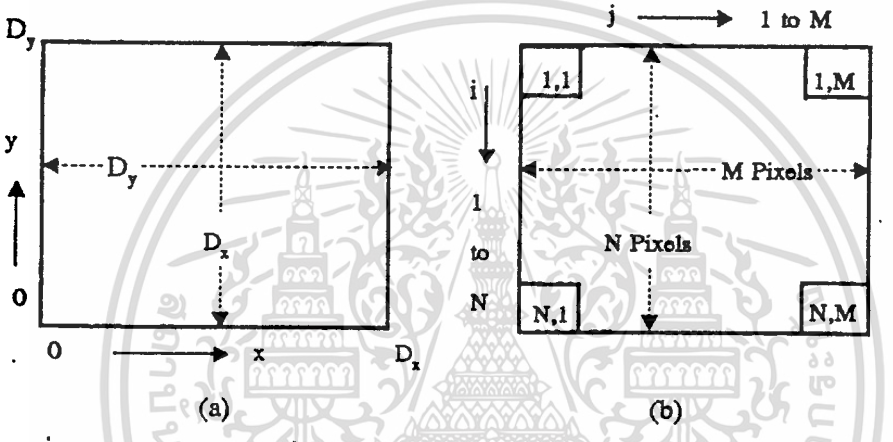
รูปที่ 2.2 แสดงข้อมูลภาพในลักษณะของเมตริกซ์

2.1.1 Pixel

ภาพสามารถอธิบายได้โดยค่าของ Pixel ที่อยู่ในรูปของเมตริกซ์ขนาด M×N แต่ละตำแหน่งของ Pixel จะแทนด้วย P_{ij} ซึ่งจะแสดงถึงความเข้มของแสงบนรูปภาพที่ตำแหน่ง (X,Y) ดังแสดงในรูปที่ 2.3 ความสัมพันธ์ระหว่าง Picture element ของภาพกับ Pixel ของเมตริกซ์ ของจุดเริ่มต้นจะอยู่คนละตำแหน่งกัน ภาพจะมีจุดเริ่มต้นที่มุมล่างด้านซ้าย ส่วนเมตริกซ์มี Pixel เริ่มต้นอยู่มุมบนด้านซ้าย

$$i = X \quad \text{เมื่อ} \quad 1 \le i \le N$$

$$j = (M-Y) \quad \text{เมื่อ} \quad 1 \le j \le M$$

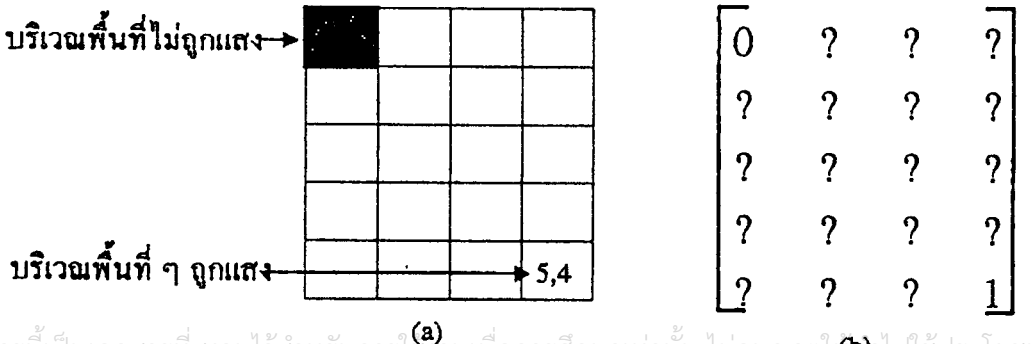


รูปที่ 2.3 แสดงความสัมพันธ์ของ Picture element (a) และการกระจาย Pixel ในเมตริกซ์ (b)

ค่าจำนวน (numerical Value) หรือขนาด (magnitude) ของ Pixel แสดงด้วยค่าเฉลี่ยความเข้มแสงบนพื้นที่ Pixel element ซึ่งแทนด้วย P_{ij} มีช่วง 0 ถึง 1

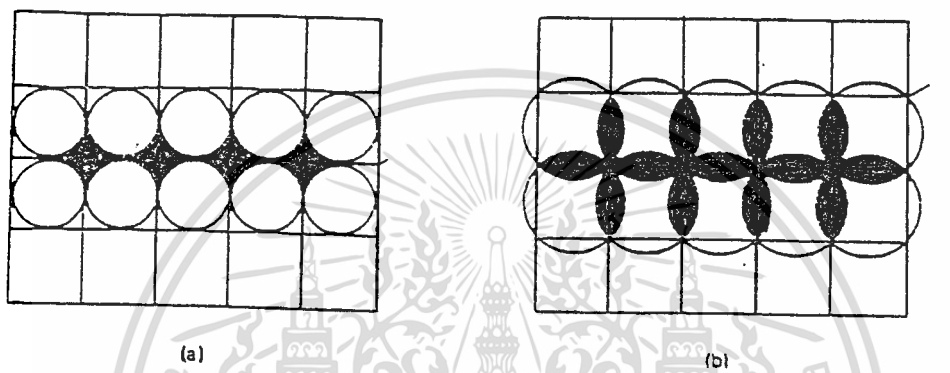
2.1.2 Pixel location

คุณสมบัติของ Pixel พื้นฐานจะเหมือนกันหมด โดยคู่ลำดับ (cordinate) ของ Pixel จะอยู่ในอะเรย์ขนาด N×M แทนเป็นรูปภาพ Pixel ที่ตำแหน่ง (n,m) จะมีค่าอยู่ค่าหนึ่ง ซึ่งถูกแทนด้วยค่าเฉลี่ยของแสงที่ตกกระทบลงบนพื้นที่ของภาพ รูปที่ 2.4 แสดงถึงจุดภาพบริเวณที่มีแสงตกกระทบใน



รูปที่ 2.4 แสดงถึงบริเวณพื้นที่ถูกแสงของภาพ (a) และค่าของ Pixel ของพื้นที่นั้น ๆ (b)

รูปของเมตริกซ์ บริเวณที่มีแสงสว่างจะแทนด้วย "1" บริเวณที่ไม่มีแสงสว่างจะแทนด้วย "0" Pixel ที่แทนจุดภาพมีอยู่ 2 แบบคือ วงกลมกับสี่เหลี่ยม จุดภาพจะเป็นแบบใดนั้นขึ้นอยู่กับตัวเซ็นเซอร์ที่จับภาพมา ในกรณีเป็นกล้องที่ใช้หลอด (tube camera) จะเป็นแบบวงกลม ซึ่งอาจจะเป็นจุดที่เหลื่อมกัน (overlap) หรือไม่เหลื่อมกัน ดังรูปที่ 2.5



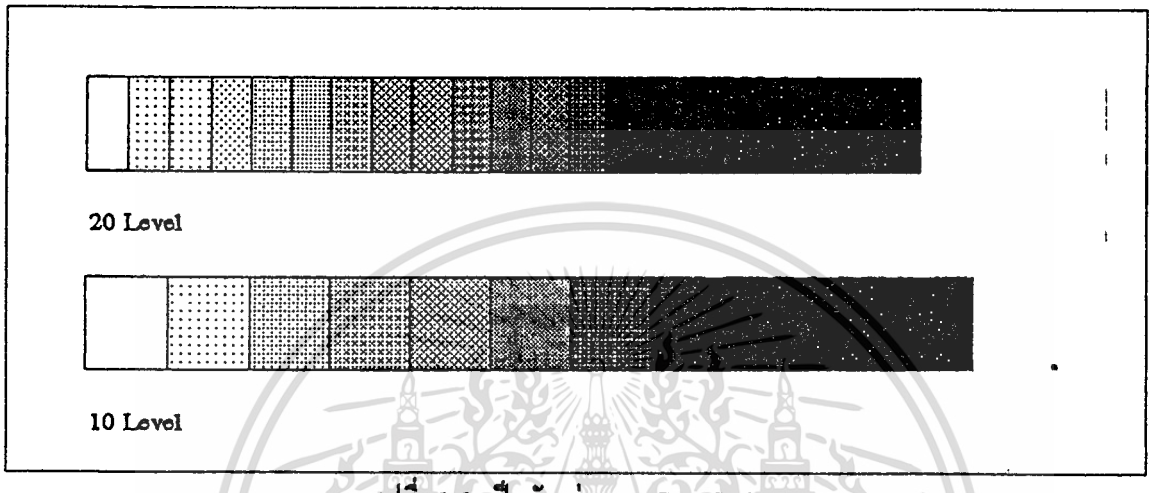
รูปที่ 2.5 แสดงจุด pixel แบบวงกลม

2.1.3 Gray Scale

ค่าความเข้มแสงของ Pixel ที่กล่าวมาข้างต้นนั้นเป็นการแทนด้วยข้อมูลเพียงบิตเดียว ภาพที่ได้ก็จะเป็นเพียงภาพ 2 ระดับขาวกับดำ แต่ถ้าต้องการให้ภาพนั้นแสดงเป็นความเข้มหลายระดับต้องแทนค่าในหนึ่ง Pixel ด้วยข้อมูลหลายบิต ภาพที่ได้ก็จะเรียกว่า ภาพระดับแสงสีเทาหรือ Gray Scale การจะทำให้ภาพเป็นกี่ระดับเทานั้นสามารถทำได้โดย เช่น ถ้าแทน 1 Pixel ด้วยข้อมูลจำนวน 2 บิตจะได้ระดับเทาเป็น 4 ระดับ ซึ่งหาได้จาก 2^n ตัวอักษร n คือจำนวนบิตที่แทนในหนึ่ง Pixel จะได้ค่าระดับความเข้มของแสงสีเทาเป็นเลขจำนวนเต็มจาก 0 ขึ้นไป เช่น

ข้อมูล	0000	ระดับแสงสีเทาเท่ากับ 0
	0001	ระดับแสงสีเทาเท่ากับ 1
	0010	ระดับแสงสีเทาเท่ากับ 2
	0011	ระดับแสงสีเทาเท่ากับ 3
	1111	ระดับแสงสีเทาเท่ากับ 15

โดยปกติแล้วการนำภาพมาประมวลผลจะใช้ 64 ระดับเทา เพราะความละเอียดขนาดนี้สายตาของคนก็ไม่สามารถแยกแยะได้ ดังนั้นถือว่าเป็นการเพียงพอแล้วที่จะนำไปใช้งาน อีกอย่างยังสามารถประหยัดหน่วยความจำที่ใช้เก็บข้อมูลภาพและลดเวลาในการประมวลผล ตัวอย่างในรูปที่ 2.6 แสดงถึงระดับสีเทา (Gray Scale)



รูปที่ 2.6 เป็นตัวอย่างของ Gray Scales

2.2 การแปลงภาพเป็น 2 ระดับ

ดังที่กล่าวมาแล้วว่าในจุด pixel จะมีค่าระดับของแสงสีเทาของมันอยู่ แต่การประมวลผลภาพจะเร็วขึ้น ถ้าภาพนั้นเป็นภาพ 2 ระดับ ดังนั้นจึงมีการแปลงภาพจากระดับเทาให้เป็นภาพ 2 ระดับ โดยใช้หลักการเปรียบเทียบค่าของ pixel แต่ละจุดกับค่ากลางของภาพ ถ้าค่าของ pixel ณ จุดใดๆ มีค่าน้อยกว่าหรือเท่ากับค่ากลางของภาพ จะให้ pixel จุดนั้นเป็นค่า 0 และในทางตรงกันข้าม ซึ่งมีวิธีการดังนี้คือ จะตรวจสอบว่าจุด pixel ที่แสดงบนจอภาพมีค่าสูงสุดและต่ำสุดเป็นเท่าไร เมื่อได้ค่านี้มาแล้วจะนำมาหาค่าเฉลี่ย (Average) หรือค่ากลางนั่นเอง จากนั้นก็นำค่านี้ไปเปรียบเทียบกับจุด pixel ทุกจุด ถ้าเปรียบเทียบแล้วปรากฏว่าค่าจุด pixel นั้นมีค่าน้อยกว่าค่ากลางจะเปลี่ยนค่า pixel นั้นให้เป็นศูนย์ แต่ถ้าเปรียบเทียบแล้วมีค่ามากกว่าค่ากลางจะปรับค่า pixel นั้นให้เป็นค่าสูงสุด ดังสมการข้างล่าง

$$P_{ij} = \begin{cases} 1 & \text{if } Y_{ij} > \text{Average} \\ 0 & \text{if } Y_{ij} < \text{Average} \end{cases}$$

P_{ij} คือ ค่าของจุดภาพที่มีแสงสีเทา 2 ระดับ

Y_{ij} คือ ค่าของจุดภาพระดับเทา

บทที่ 3

คุณสมบัติของแผงวงจร DigiIMAGE รุ่น DZ-II

3.1 การถ่ายภาพ

1. บนแผงวงจรมีหน่วยความจำสำหรับภาพขนาด 192K ไบต์ ซึ่งไมโครคอมพิวเตอร์สามารถติดต่อกับหน่วยความจำนี้ได้โดยตรง

2. วงจรถ่ายภาพเป็นแบบ REAL TIME สำหรับภาพขาวดำและเกือบ REAL TIME สำหรับภาพสี

3. สามารถดูภาพ LIFE TIME จากจอ VGA ได้ขณะถ่ายภาพ (DIGITIZE)

4. การปรับแต่งภาพสามารถควบคุมแสงสว่าง (BRIGHTNESS) ความชัดเจน (CONTRAST)และความเข้มของสี (SATURATION) ได้จากซอฟท์แวร์

5. รับสัญญาณอินพุทแบบ COMPOSITE ขาวดำหรือสี (ระบบ พาว(PAL)) ก็ได้ เช่น สัญญาณจากกล้องถ่ายวีดิโอ

ความละเอียดของภาพ (ที่ได้จากการถ่ายภาพด้วยโปรแกรม DZ2DI.COM)

1. ความละเอียดของภาพสีได้ถึง 320x200 จุด 256 สี (สีจริง)
2. ความละเอียดของภาพขาวดำได้ถึง 320x200 จุด 64 ระดับเทา
3. ภาพที่ถ่ายได้สามารถเก็บในรูปแบบ PIC ของ PC STORY BOARD PLUS หรือ TIFF

3.2 หลักการทำงานของแผงวงจร DigiIMAGE

แผงวงจร DigiIMAGE ประกอบด้วยส่วนต่างๆที่สำคัญ ดังนี้คือ

3.1 หน่วยความจำสำหรับเก็บภาพอิมเมจ (RAM) 3 BANK BANK ละ 64K ไบต์

3.2 ส่วนแปลงสัญญาณวีดีโอจากอะนาล็อกให้เป็นข้อมูลดิจิตอล (A/D)

3.3 ส่วนแยกสัญญาณสี

3.4 ส่วนปรับ CONTRAST, BRIGHTNESS, SATURATION

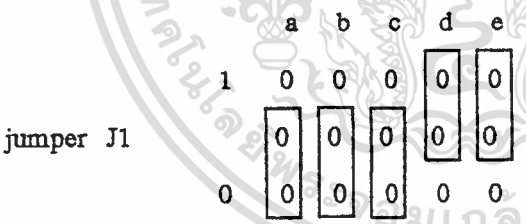
3.5 ส่วนควบคุมการติดต่อระหว่างส่วนต่างๆจากคอมพิวเตอร์

เมื่อถ่ายภาพสีสัญญาณสีจากกล้องวีดีโอจะเข้ามาทางอินพุทของวงจรไปเข้าส่วนแยกสัญญาณสีออกมาเป็นสี 3 สี คือ สีแดง(R) สีเขียว(G) สีน้ำเงิน(B) แล้วจึงส่งไปส่วนแปลงสัญญาณอะนาล็อกเป็นดิจิตอล เพื่อนำสัญญานภาพไปเก็บในหน่วยความจำ(RAM) ในส่วนของวงจรแยกสีนี้ เราสามารถควบคุมระดับความเข้มสีจากไม่มีสี ไปจนถึงมีสีเข้มมากได้ เรียกว่า การปรับ SATURATION ซึ่งควบคุมได้ด้วยซอฟแวร์ส่วนในวงจร A/D เราสามารถควบคุมความชัดเจนของ

ภาพได้ด้วยการปรับ Reference+ (Ref+) และ Reference- (Ref-) ซึ่งโดยทั่วไป Ref+ จะอยู่สูงกว่าสัญญาณภาพ และ Ref- จะอยู่ต่ำกว่าสัญญาณภาพ เราควบคุมได้ด้วยซอฟแวร์ ลักษณะการนำสัญญาณภาพที่ถูกแปลงเป็นดิจิตอลแล้วไปเก็บในหน่วยความจำนั้น จะทำการเก็บทีละสีลงในหน่วยความจำสีละ BANK จนครบ 3 BANK ตามลำดับ เมื่อถ่ายภาพขาวดำ(สั่งจากซอฟแวร์) สัญญาณจะนำไปเข้าส่วนวงจรแปลงอะนาล็อก เป็นดิจิตอลเลย ดังนั้นการเก็บข้อมูลภาพจึงอาจเก็บไว้ในหน่วยความจำเพียง BANK เดียวเท่านั้น ไมโครคอมพิวเตอร์สามารถติดต่อกับหน่วยความจำ(RAM)บนการ์ดได้ โดยการสั่งผ่านพอร์ทควบคุม ปกติหน่วยความจำภาพนี้จะแยกออกจากระบบไมโครคอมพิวคอมพิวเตอร์ ในการถ่ายภาพ ผู้ใช้สามารถเลือกติดต่อกับหน่วยความจำ BANK .ใดก็ได้ เพื่อทำการเก็บข้อมูลภาพที่ถ่ายนั้น ในทำนองเดียวกัน ผู้ใช้ก็สามารถเลือกติดต่อระหว่างหน่วยความจำ BANK ใดก็ได้เพื่อทำการอ่านข้อมูลที่เก็บไว้แล้วมาใช้ แต่จะต้องสั่งให้วงจรนั้นหยุดถ่ายภาพเสียก่อน (FREEZE) จึงทำการอ่านข้อมูลได้ และเมื่ออ่านข้อมูลจากหน่วยความจำเรียบร้อย จึงสามารถสั่งให้หน่วยความจำนั้นกลับไปอยู่ในโหมดการถ่ายภาพนั้น เพื่อถ่ายภาพใหม่ได้อีก

3.3 การติดตั้งระบบฮาร์ดแวร์

แผงวงจร DigiIMAGE รุ่น DZ-II สามากำหนดตำแหน่งของพอร์ทและหน่วยความจำได้โดยผู้ใช้ ด้วยการติดตั้งตำแหน่งของ jumper J1 ตำแหน่งของแอดเดรสที่เลือกได้จะมีดังนี้

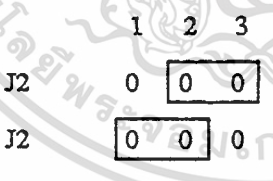


แอดเดรสของพอร์ทเลือกที่ Jumper a,b			แอดเดรสของหน่วยความจำเลือกที่ Jumper c,d,e			
I/O Address	a	b	Memory Address	c	d	e
210-21Fh	0	0	C400 - C3FFH	0	0	0
2A0-2AFh	0	1	C800 - CBFFH	0	0	1
310-31Fh	1	0	CC00 - CFFFH	0	1	0
3A0-3AFh	1	1	D000 - D3FFH	0	1	1
			D800 - DBFFH	1	0	0
			DC00 - DFFFH	1	0	1
			E000 - E3FFH	1	1	0
			Not Select	1	1	1

โดยปกติแผงวงจรรุ่น DZ-II จะถูกติดตั้งแอดเดรสของพอร์ทไว้ที่ 210H และแอดเดรสของหน่วยความจำไว้ที่ D000H หากต้องการเลือกแอดเดรสของพอร์ทไว้ที่ตำแหน่งอื่น ให้เลือกตำแหน่งของ jumper J1 ตรง a,b หากต้องการเลือกแอดเดรสของหน่วยความจำใหม่ ก็ให้เลือกตำแหน่งของ jumper J1 c,d,e ผู้ใช้สามารถทำการติดตั้งระบบฮาร์ดแวร์ด้วยตัวเองได้โดยการเสียบแผงวงจร DigiIMAGE ลงในช่องขยาย (Expansion Slot) ช่องใดช่องหนึ่งของเครื่องคอมพิวเตอร์ สำหรับเครื่องคอมพิวเตอร์ทั่วไปไม่ว่าจะเป็นระบบ PC XT,AT หรือ AT386 ที่ประกอบด้วยแผงวงจรจอภาพ (ภาษาไทย 25 บรรทัด,Mono,HGC,CGA,EGA,VGA) แผงควบคุมดิสค์ไดรฟ์, ฮาร์ดิสค์และแผงวงจร Multi I/O (หรือ Multi Function) จะสามารถติดตั้งแผงวงจร DigiIMAGE ลงไปได้ทันที

หากเครื่องคอมพิวเตอร์มีแผงวงจรอื่นที่กล่าวมาแล้วติดตั้งอยู่ เช่น แผงวงจร LAN, แผงวงจรสแกนเนอร์, แผงวงจร EMS และแผงวงจรพิเศษอื่น ๆ รวมทั้งวงจรพิเศษที่มีการติดตั้งอยู่บนเมนบอร์ด เช่น วงจร EMS (สำหรับเครื่อง AT บางรุ่น) หากแอดเดรสของพอร์ทหรือหน่วยความจำของวงจรพิเศษเหล่านี้ ถูกเลือกไว้ตรงกับแผงวงจร DigiIMAGE ก็จะต้องมีการย้ายตำแหน่งแอดเดรสของแผงวงจรอันใดอันหนึ่งใหม่ เพื่อไม่ให้ตำแหน่ง Address ของแผงวงจร DigiIMAGE ไปชนกับแอดเดรสของแผงวงจรอื่น

ส่วนตำแหน่ง jumper ที่ J2 และ J3 ใช้สำหรับการเลือกสัญญาณจาก input เพื่อให้เหมาะสมกับแผงวงจร DigiIMAGE ความหมายและตำแหน่งมีให้เลือกดังนี้



J2 เชื่อมระหว่าง

1,2 = เลือกใช้กับกล้องที่ให้สัญญาณออกมาเป็น ขาว/ดำ

2,3 = เลือกใช้กับกล้องที่ให้สัญญาณออกมาเป็น สี

J3 เชื่อมระหว่าง

1,2 = สำหรับสัญญาณวิดีโอที่ต้องการ INPUT IMPEDANCE สูง

2,3 = สำหรับสัญญาณวิดีโอที่ต้องการ INPUT IMPEDANCE 75 โอห์ม

โดยปกติตำแหน่งของ jumper J2 จะติดตั้งไว้ที่ 2,3 และ J3 จะตั้งไว้ที่ 1,2

เมื่อทำการติดตั้งแผงวงจร DigiIMAGE ลงไปในเครื่องเรียบร้อยแล้ว นำสัญญาณวิดีโอป้อนเข้าที่แจ็ค VDO-IN จากนั้นให้เรียกโปรแกรม DZ2DI.COM เลือก menu ไปที่ Digitize now เพื่อนำสัญญาณวิดีโอไปแสดงผลบนจอภาพตามขั้นตอนต่อไป

3.4 การติดต่อกับ Card DZ-II

บนแผงวงจร DigiIMAGE DZ-II มีพอร์ท Output Port อยู่ 4 Port , Input Port อยู่ 2 Port ซึ่งมีหน้าที่ต่าง ๆ ดังนี้

3.4.1 Output Port 4 Port มีดังนี้

CtrlPort : เป็นพอร์ทคำสั่งในการถ่ายภาพ รวมทั้งการเลือก BANK , PAGE ของหน่วยความจำและสีที่ต้องการถ่าย ตำแหน่งของพอร์ทนี้อยู่ที่ 210H, 2A0H, 310H, 3A0H ตามการตั้งตำแหน่งของ jumper J1 ความหมายของพอร์ทนี้มีดังนี้

d7 เป็นคำสั่งในการถ่ายหรือหยุดภาพ

0 = สั่งให้หยุดถ่ายภาพ (digitize)

1 = สั่งให้ถ่ายภาพ (digitize)

d6 สัญญาณนี้ต่อออกไปที่ช่อง J-OUT สามารถนำไปควบคุมอะไรก็ได้ระดับของสัญญาณจะเหมือนสัญญาณที่ออกมาจาก IC TTL

d5,d4 เลือกให้สัญญาณที่ต้องการถ่ายไปเข้าวงจร A/D

00 = เลือกถ่ายสัญญาณภาพขาวดำ

01 = เลือกถ่ายสัญญาณภาพสีแดง

10 = เลือกถ่ายสัญญาณภาพสีเขียว

11 = เลือกถ่ายสัญญาณภาพสีน้ำเงิน

d3,d2 เลือกหน่วยความจำ BANK ที่ต้องการเก็บภาพที่ถูกแปลงเป็นดิจิตอลหรือเลือกหน่วยความจำ BANK ที่ต้องอ่านหรือเขียนข้อมูล ซึ่งมีอยู่ 3 BANK

00 = เลือก RAM BANK 0

01 = เลือก RAM BANK 1

10 = เลือก RAM BANK 2

d1,d0 เลือก PAGE ของหน่วยความจำที่ต้องการอ่านและเขียน โดยในหน่วยความจำแต่ละ BANK จะติดต่อกับ CPU เป็น PAGE ซึ่งมี BANK ละ 4 PAGE แต่ละ PAGE มีหน่วยความจำอยู่ 16K Byte

00 = เลือก RAM PAGE 0

01 = เลือก RAM PAGE 1

10 = เลือก RAM PAGE 2

11 = เลือก RAM PAGE 3

RefhPort : มีตำแหน่งอยู่ที่ CtrlPort+1 เป็นค่าของ reference+ สำหรับควบคุมการทำงานของวงจรแปลงอะนาล็อกเป็นดิจิตอล

d5 - d0 = ค่าตั้งแต่ 0 - 63

ReflPort : มีตำแหน่งอยู่ที่ CtrlPort+2 เป็นค่าของ reference- สำหรับควบคุมการทำงานของวงจรแปลงอะนาล็อกเป็นดิจิตอล

d5 - d0 = ค่าตั้งแต่ 0 - 63

SetPort : มีตำแหน่งอยู่ที่ CtrlPort+3 เป็นค่าของ Seturation สำหรับควบคุมการทำงานของวงจรแยกสี

d5 - d0 = ค่าตั้งแต่ 0 - 63

3.4.2 Input Port 2 Port ดังนี้

StatPort : มีตำแหน่งอยู่ที่ CtrlPort เป็นพอร์ทอ่านสัญญาณที่ใช้ในการตรวจสอบเมื่อจะถ่ายภาพ รวมทั้งอ่านข้อมูลของ CtrlPort บางอย่างกลับมาด้วย

d7 อ่านมาจาก d7 ของ CtrlPort

d6 ตรวจสอบสัญญาณจากช่อง J-IN เพื่อนำไปใช้ในโปรแกรมต่าง ๆ ผู้ใช้สามารถตรวจสอบไปใช้งานได้ สัญญาณที่ต้องการจะต้องมีระดับแรงดันเท่ากับที่อินพุทของ IC TTL ต้องการ ปกติจะถูกทำให้เป็นไฟบวก 5 Volt ด้วยการต่อค่าความต้านทาน 4.7 kohm เข้ากับไฟ 5 Volt

d5 ตรวจสอบสัญญาณการถ่ายภาพ digi_stat (เป็น PULSE)

d4 ตรวจสอบสัญญาณ Hor , Sync (เป็น PULSE)

d3 - d0 อ่านมาจาก d3 - d0 ของ CtrlPort

SWport : มีตำแหน่งของพอร์ทอยู่ที่ CtrlPort+1 เป็นพอร์ทที่ใช้อ่านตำแหน่งของ jumper J1 ตัว c,d,eที่ได้ตั้งไว้พอร์ทนี้มีคุณสมบัติพิเศษคือจะต้องอ่านอย่างน้อย 2 ครั้ง โดยครั้งแรกอาจจะได้ข้อมูลที่ผิด

d3,d2,d1 อ่านจากตำแหน่งของ jumper J1 ตรง c,d,e ตามลำดับ

d0 มีค่าเป็น 0 (สำหรับตรวจสอบว่าเป็นแผงวงจรสี)

3.4.3 ตัวอย่างการติดต่อกับ Card DZ-II

การสั่งให้ DZ-II ถ่ายภาพ (digitize)

```
out CtrlPort+2,(63 - brightness)      ; Ref-
out CtrlPort+1,(reference-) + ((63 - reference-) * (63 - contrast))/63      ; Ref+
out CtrlPort+3,Saturation
out CtrlPort,80h   ; ถ่ายสี Y ลง RAM BANK 0
out CtrlPort,90h   ; ถ่ายสี R ลง RAM BANK 0
out CtrlPort,0a4h  ; ถ่ายสี G ลง RAM BANK 1
out CtrlPort,0b8h  ; ถ่ายสี B ลง RAM BANK 2
out CtrlPort,0     ; หยุดการถ่าย (freeze)
```

การสั่งให้ติดต่อกับหน่วยความจำ BANK 0 ของ DZ-II

```
out CtrlPort,00    ; PAGE 0
out CtrlPort,01    ; PAGE 1
out CtrlPort,02    ; PAGE 2
out CtrlPort,03    ; PAGE 3
```

การสั่งให้ติดต่อกับหน่วยความจำ BANK 1 ของ DZ-II

```
out CtrlPort,04    ; PAGE 4
out CtrlPort,05    ; PAGE 5
out CtrlPort,06    ; PAGE 6
out CtrlPort,07    ; PAGE 7
```

การสั่งให้ติดต่อกับหน่วยความจำ BANK 2 ของ DZ-II

```
out CtrlPort,08    ; PAGE 8
out CtrlPort,09    ; PAGE 9
out CtrlPort,0ah   ; PAGE 10
out CtrlPort,0bh   ; PAGE 11
```

3.5 ขั้นตอนการควบคุม DZ-II

1. ก่อนที่เราจะสั่งถ่ายภาพ เราจะต้องสั่งคำสั่งออกไปทาง RefhPort, ReflPort และ SatPort ก่อนเพื่อกำหนดสภาวะของการทำงานของวงจรถ่ายภาพโดย RefhPort และ ReflPort คือ

คำนวณมาจาก Contrast และ Brightness ที่กำหนดไว้ก่อน ส่วน SatPort คือค่า Saturation ของภาพ ปกติควรให้ค่าของ Contrast = 32 , Britghness = 50 และ Saturation = 32

2. ตรวจสัญญาณ Hor-Sync (ที่บิต d4 ของ StatPort) รอจนกว่าจะเป็นขอบขาขึ้น (จาก 0 ไป 1) สั่งถ่ายภาพ(digitize) โดยการส่งคำสั่งออก CtrlPort ซึ่งประกอบไปด้วยคำสั่งในการถ่าย, เลือกสัญญาณที่ต้องการถ่ายไปเข้าวงจรอะนาล็อกเป็นดิจิตอลและเลือกหน่วยความจำ BANK ที่ต้องการ

3. ตรวจสัญญาณ digi_stat (ที่บิต d5 ของ SatPort) จะต้องมี PLUSE (1) ภายในไม่เกิน 20 ms แล้วรอสัญญาณนี้จนตกเป็น 0 หมายถึงถ่ายภาพครบ 1 frame

4. หากต้องการสั่งให้หยุดถ่ายภาพ (digitize freeze) ให้สั่งคำสั่งหยุดถ่ายภาพออกมาที่ CtrlPort (บิต d7 เป็น 0)

5. สำหรับการถ่ายภาพในโหมด color RGB จะต้องสั่งถ่ายภาพสี R,G,B ลงหน่วยความจำ BANK 0,1,2 ตามลำดับ ดังนั้นจึงต้องสั่งถ่ายภาพทั้งหมด 3 ครั้ง และในแต่ละครั้งจะต้องรอสัญญาณ digi_stat ตกจาก PLUSE 1 เป็น 0 เสมอ

6. เมื่อต้องการอ่านข้อมูลที่อยู่บนหน่วยความจำของ Card DZ-II มายังหน่วยความจำของคอมพิวเตอร์ เราสามารถอ่านได้ครั้งละ 16kbyte หรือเท่ากับ 1 PAGE ฉะนั้นการอ่านภาพสี 64 kbyte 3 BANK จะต้องอ่านถึง 12 ครั้ง ซึ่งในการอ่านแต่ละครั้งจะต้องส่งคำสั่งออก CtrlPort เพื่อทำการเลือก BANK และ PAGE ที่ต้องการอ่าน

3.5.1 ขั้นตอนต่าง ๆ สามารถเขียนเป็นกระบวนความได้ดังนี้

1. การตรวจสอบของ Card และสแกนหา Address ของ CtrlPort

เป็นการตรวจหาพอร์ททั้ง 4 ตำแหน่งคือ 210h,2a0h,310h และ 3a0h โดยการตรวจดูสัญญาณที่บ่งบอกว่าเป็น Crad DZ-II สัญญาณที่ตรวจจะอยู่ที่พอร์ทห่างออกไป 8001h เช่น 210h จะตรวจที่ 8211h เป็นต้น

```
PROCEDURE Chk_ID_Card
    PortBase [ ARRAY 1 TO 4 ] = 210h,2a0h,310h,3a0h
    Index = 1
    FOR Count = 1 TO 4 DO
        CtrlPort = PortBase[Index]
        SWPort = CtrlPort + 1
        ChkPort = CtrlPort OR 8001h
        IN PORT  SWPort TO Chk1        (Skip first byte)
```

```
        IN PORT  SWPort TO Chk1
        Chk1 = ((Chk1 XOR 0fh) AND ofh)
        Chk2 = Chk1
        FOR Time = 0 TO 200 DO
            IN PORT  ChkPort TO Chk1
            Chk1 = ((Chk XOR 0fh) AND 0fh)
            IF  (Chk1 <> Chk2)
                GOTO NextBase
            END IF
            Chk2 = ((Chk2 XOR 0fh) AND 0fh)
        END FOR
        EXIT (with CtrlPort)
        NextBase:
        Index = Index+1
    END FOR
PRINT "Card DZ-II Not Found  !!"
END PORCEDURE
```

2. การตรวจสัญญาณ Digi_Stat (ที่บิต d5 ของ StatPort)

จะต้องตรวจหลังจากคำสั่งถ่ายภาพออกไปแล้ว ซึ่งเมื่อออกจาก PROCEDURE นี้แล้วจะหมายถึงถ่ายภาพได้ครบ 1 เฟรม แต่ถ้าสัญญาณ Digi_Stat ไม่เคยเป็น 1 เลยภายในช่วงเวลาที่กำหนดแสดงว่า ไม่สัญญาณ Video ส่งมาที่การ์ด

```
PROCEDURE Chk_Digi_Stat
    FOR Pulse = 1 DOWD TO 0
        Timer = 0
        Timer_Out = 4000
        WHILE digi_stat <> Pulse
            Timer = Timer+1
            IF (Timer > Timer_Out)
                PRINT "VIDEO Signal Not Found."
                EXIT
```

```
            END IF
        END WHILE
    END FOR
END PROCEDURE
```

3. การสั่งถ่ายภาพ

เราสามารถเลือกถ่ายได้ทั้งโหมดสีและขาวดำ สำหรับ PROCEDURE Digitize_RGB จะเป็นการสั่งถ่ายภาพสี R,G,B ขนาด 256x256 จุด ลงหน่วยความจำ BANK 0,1,2 ตามลำดับและ PROCEDURE Digitize_B&W เป็นการสั่งถ่ายภาพขาวดำขนาด 256x256 จุด ลงหน่วยความจำ BANK 0 เพียง BANK เดียว จะเห็นว่าขั้นตอนการสั่งถ่ายภาพแต่ละครั้งจะต้องทำการตรวจสัญญาณ Digi_Stat เสมอ เพื่อเป็นการบอกให้รู้ว่าภาพที่กำลังถ่ายอยู่นั้นถ่ายครบเฟรมหรือยัง เมื่อกระบวนการถ่ายเสร็จสิ้นไปแล้วเราจะสั่งให้หยุดถ่ายเป็นการถ่ายครบ 1 ภาพและเมื่อต้องการถ่ายภาพต่อไปก็ให้ทำตามขั้นตอนของกระบวนการนี้

การสั่งถ่ายภาพสี

```
PROCEDURE Digitize_RGB
    DigiCom_R = 90h (10010000b)
    DigiCom_G = A4h (10100100b)
    DigiCom_B = B8h (10111000b)
    DigiComStop = 0 (0000000b)
        OUT CtrlPort,DigiCom_R
            CALL Chk_Digi_Stat
        OUT CtrlPort,DigiCom_G
            CALL Chk_Digi_Stat
        OUT CtrlPort,DigiCom_B
            CALL Chk_Digi_Stat
        OUT CtrlPort,DigiComStop
END PROCEDURE
```

การสั่งถ่ายภาพขาวดำ


```
PROCEDURE Digitize_B&W
    DigiCom_Y = 80h (10000000b)
```

```
    DigiComStop = 0 (00000000b)
        OUT CtrlPort,DigiCom_Y
            CALL Chk_Digi_Stat
        OUT CtrlPort,DigiComStop
END PROCEDURE
```

4. การปรับค่าของ Contarst,Brightness และ Saturation

สามารถเปลี่ยนค่านี้ได้ตั้งแต่ 0-63 ระดับ และค่าที่ได้จากการเปลี่ยนแปลงเราจะนำมาโปรแกรมตามขั้นตอนของกระบวนการนี้

```
PROCEDURE Con_Bri_Sat_Level
    Contarst = 32 (Vary from 0 to 63)
    Brightness = 50 (Vary from 0 to 63)
    Saturation = 32 (Vary from 0 to 63)
    Reference_ = 63-Brightness
    RefhPort = CtrlPort+1
    ReflPort = CtrlPort+2
    SatPort = CtrlPort+3
        OUT ReflPort,Reference_
        OUT RefhPort,Reference_+[[63-Reference-]x[63-Contrast]]/63
END PROCEDURE
```

5. การอ่านตำแหน่ง Address ของหน่วยความจำเป็น Card DZ-II

ทำได้โดยการอ่านค่าของ J1 จาก SWPort แล้วจึงเปิดตารางหาตำแหน่งของ Address

```
PROCEDURE Read_Address
    SWPort = CtrlPort+1
    IN SWPort TO ChkSW (LATCH DATA)
    IN SWPort TO ChkSW (GET DATA)
    ChkSW = ShkSW AND 0eh (00001110b)
        CASE ChkSW DO
            0 : CardSeg = C400h
```

```
        2 :  CardSeg = C800h
        4 :  CardSeg = CC00h
        6 :  CardSeg = D000h
        8 :  CardSeg = D800h
        10 : CardSeg = DC00h
        12 : CardSeg = E000h
        14 : CardSeg = NULL
    END CASE
END PROCEDURE
```

6. การติดต่อกับหน่วยความจำของ DZ-II

เราจะแยกเป็น PROCEDURE Read_RGB สำหรับอ่านข้อมูลจากหน่วยความจำ BANK 0,1,2 ของ Card DZ-II ซึ่งเป็นหน่วยความจำที่เก็บข้อมูลของภาพสี R,G,B ตามลำดับลงหน่วยความจำของระบบที่จองไว้ 64Kbyte และ PROCEDURE Read_Y จะเป็นการอ่านข้อมูลจากหน่วยความจำ BANK 0 ที่เก็บข้อมูลของภาพขาวดำเพียง BANK เดียว เพื่อนำลงหน่วยความจำของระบบที่จองไว้ 64Kbyte ส่วน PROCEDURE Read_RAM_Bank จะเป็นโปรแกรมย่อยเพื่อให้เรียกใช้โดยจะส่งค่าพารามิเตอร์ RamBank และ BufSeg ไปให้

การติดต่อกับหน่วยความจำภาพสี

```
PROCEDURE Read_RGB
    Bank_R = 10h (00010000b)
    Bank_G = 24h (00100100b)
    Bank_B = 38h (00111000b)
    BufSeg_R = 64Kbyte ALLOCATE MEMORY
    BufSeg_G = 64Kbyte ALLOCATE MEMORY
    BufSeg_B = 64Kbyte ALLOCATE MEMORY
        CALL Read_RAM_Bank WITH Bank0_R,BufSeg_R
        CALL Read_RAM_Bank WITH Bank1_R,BufSeg_G
        CALL Read_RAM_Bank WITH Bank2_R,BufSeg_B
END PROCEDURE
```

การติดต่อกับหน่วยความจำสำหรับภาพขาวดำ

```
PROCEDURE Read_Y
    Bank_Y = 0 (00000000b)
    BufSeg_Y = 64kbyte ALLOCATE MEMORY
        CALL Read_RAM_Bank WITH Bank0_Y,BufSeg
END PROCEDURE
```

โปรแกรมย่อยสำหรับอ่านหน่วยความจำจากคาร์ดลงหน่วยความจำของระบบ

```
PROCEDURE Read_RAM_Bank (RamBank,BufSeg)
(CardSeg from PROCEDURE Read_Address)
CardOffs = 0
BufOffs = 0
    FOR Count = 1 TO 4
        OUT CtrlPort,RanBank
        MOVE DATA FROM CardSeg:CardOffs TO BufSeg:BufOffs LENGTH 16kbyte
        RamBank = RamBank+1 (to next page)
        BufOffs = BufOffs+16kbyte
    END FOR
END PROCEDURE
```

3.6 ลักษณะของข้อมูลภาพ

ภาพขาวดำที่เก็บอยู่บนหน่วยความจำนั้นจะมีขนาด 256x256 จุด โดยใช้จำนวนบิตทั้งหมด 6 บิตบน คือตั้งแต่บิต 2 ถึงบิต 7 แบ่งระดับเทาเป็น 0-63 ระดับ และแต่ละจุดจะต้องใช้เนื้อที่ขนาด 1 ไบต์เรียงกันไปในหน่วยความจำขนาด 64Kbyet ของ BANK 0 โดยข้อมูลภาพจุดแรกจะอยู่ที่หน่วยความจำแอดเดรสแรกคือ 0 ข้อมูลภาพจุดถัดมา(ตามแนวนอน) ก็จะอยู่ที่แอดเดรส 1 เรียงกันไปจนจบข้อมูลภาพ จะมีแอดเดรสเป็น 65535 หรือ FFFFH

สำหรับภาพสีจะใช้หน่วยความจำ 3 BANK แบ่งเป็นสี R,G,B สีละ BANK แต่ละ BANK สามารถจุได้ทั้งหมด 256x256 จุด แต่ละจุดจะมีข้อมูลภาพอยู่ 6 บิตบน คือตั้งแต่บิต 2 ถึงบิต 7 ดังนั้นบิต 0 กับบิต 1 จะเป็นข้อมูลที่ไม่ได้ใช้ และการเก็บข้อมูลแต่ละ BANK จะเหมือนกับเก็บข้อมูลของภาพขาวดำ โดยปกติข้อมูลบิต 0 และ บิต 1 ของทุก ๆ ไบต์ที่เก็บอยู่ในหน่วยความจำจะเป็นตัวบอกว่าข้อมูลนี้เป็นสีอะไรคือ

D1 D0

0 0 = ข้อมูลจากสัญญาณ Y

0 1 = ข้อมูลจากสัญญาณ R

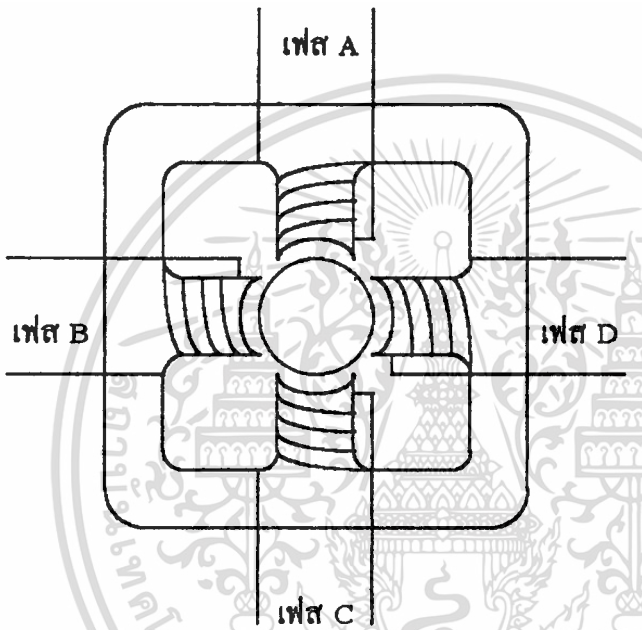
1 0 = ข้อมูลจากสัญญาณ G

1 1 = ข้อมูลจากสัญญาณ B

บทที่ 4

สเตปปิ้งมอเตอร์ (Stepping motor)

สเตปปิ้งมอเตอร์ เป็นมอเตอร์อีกชนิดหนึ่งแต่มีการหมุนเป็นแบบสเตป ในแต่ละสเตปจะบอกเป็นองศาที่หมุนไป และควบคุมการหมุนได้โดยใช้สัญญาณพัลซ์ป้อนให้กับขดลวดสเตเตอร์ ซึ่งจะมีอยู่หลายขดแต่ละขดเรียกว่า "เฟส" ดังรูป



รูปที่ 4.1 แสดงโครงสร้างภายในของสเตปปิ้งมอเตอร์

4.1 ชนิดของสเตปปิ้งมอเตอร์

1. ชนิดแม่เหล็กถาวร (Permanent Magnet)
2. ชนิดแปรค่ารีลักแตนซ์ (Vriable Reluctance)
3. ชนิดผสม (Hybrid)

สเตปปิ้งมอเตอร์ชนิดแม่เหล็กถาวร นั้นจะมีสเตเตอร์ที่พันด้วยขดลวดไว้หลาย ๆ โพล มีโรเตอร์เป็นรูปทรงกระบอกเป็นฟันเหมือนเฟืองทำด้วยแม่เหล็กถาวร เมื่อป้อนสัญญาณพัลซ์ให้กับขดลวดสเตเตอร์ จะทำให้เกิดแรงผลักต่อตัวโรเตอร์ ด้วยอำนาจแม่เหล็กไฟฟ้าทำมอเตอร์หมุนไปได้ ในขณะที่ไม่ได้ป้อนไฟให้มอเตอร์โรเตอร์จะมีแรงยึดที่เกิดจากแม่เหล็กถาวรจึงเหมือนกับว่ามีความฝืด

ชนิดแปรค่ารีลักแตนซ์ การหมุนของโรเตอร์จะหมุนได้อย่างอิสระแม้ว่าจะไม่ได้จ่ายไฟให้ก็ตาม โรเตอร์จะทำมาสารแม่เหล็กกำลังอ่อนมีรูปร่างเป็นทรงกระบอกมีฟันเหมือนเฟือง โดย

จำนวนฟันเฟืองจะมีความพันธ์กับจำนวนโพลในสเตเตอร์ ทำหน้าที่เป็นตัวกำหนดมุมที่หมุนไปแต่ละสเตป ความเฉื่อยของโรเตอร์น้อย จึงมีความเร็วรอบสูงกว่าชนิดแม่เหล็กถาวร

ชนิดผสม เป็นการผสมของทั้งสองชนิดที่กล่าวมา มีสเตเตอร์คล้ายกับชนิดแปรค่ารีลัคแตนซ์ โรเตอร์ทำจากสารแม่เหล็กที่มีกำลังสูง มุมที่เปลี่ยนไปของการหมุนมีความแม่นยำสูง ให้แรงบิดสูง มีขนาดเล็ก ขณะไม่จ่ายไฟมีแรงยึดโรเตอร์ให้นิ่งอยู่กับที่

จะเห็นว่าการหมุนของสเตปปิ้งมอเตอร์นั้น จะหมุนไปเป็นสเตป แต่ละสเตปที่หมุนไปจะบอกเป็นมุมองศา ดังนั้นการควบคุมการหมุนแบบการควบคุมตำแหน่งโดยใช้สเตปปิ้งมอเตอร์นั้นจึงเป็นเรื่องง่ายและสะดวกมาก มุมที่เปลี่ยนไปหาได้จากสูตร

มุมที่เปลี่ยนไป = ค่าของมุมแต่ละสเตปของมอเตอร์แต่ละตัว X จำนวนพัลซ์ที่ป้อนให้

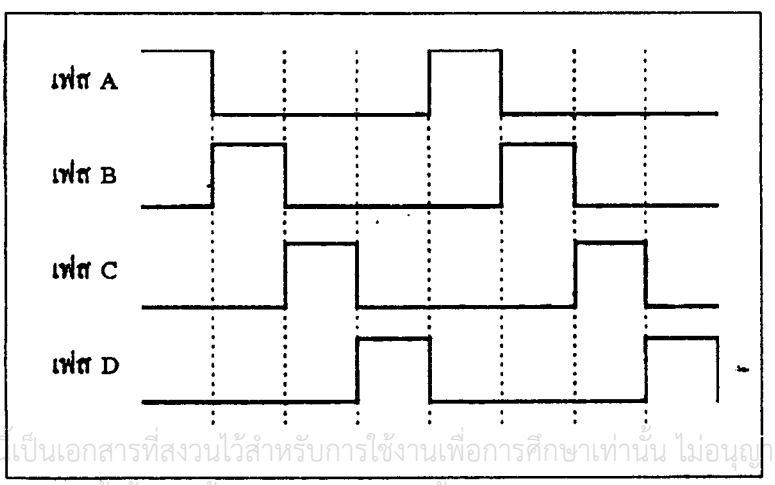
4.2 การกระตุ้นเฟสของขดลวด สเตปปิ้งมอเตอร์

การกระตุ้นเฟสคือ การป้อนสัญญาณพัลซ์ให้กับขดลวดสเตเตอร์ เพื่อจะให้มอเตอร์หมุนตามที่ต้องการ ไม่ว่าจะเป็นการหมุนไปทางซ้ายหรือทางขวาก็สามารถทำได้ โดยการเลื่อนเฟสของสัญญาณ
พัลซ์ไปทางซ้ายหรือทางขวา

การกระตุ้นเฟสมีอยู่ 3 แบบคือ

1. การกระตุ้นแบบเฟสเดียว (single phase excitation)

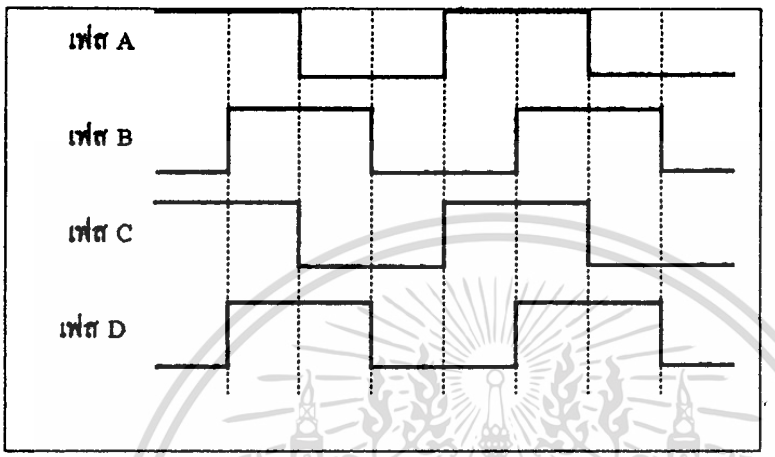
รูปที่ 3.2 แสดงถึงการกระตุ้นแบบเฟสเดียว ซึ่งเป็นการป้อนสัญญาณพัลซ์ให้กับขดลวดสเตเตอร์ทีละขดของสเตปปิ้งมอเตอร์ 4 เฟส เมื่อต้องการให้หมุนตามเข็มนาฬิกา จะป้อนพัลซ์เรียงเฟสเป็น 1→2→3→4 ถ้าต้องการให้หมุนทวนเข็มป้อน 4→3→2→1 แบบนี้แรงบิดน้อย



รูปที่ 4.2 แสดงการกระตุ้นแบบ 1 เฟส

2. การกระตุ้นแบบ 2 เฟส (two phase excitation)

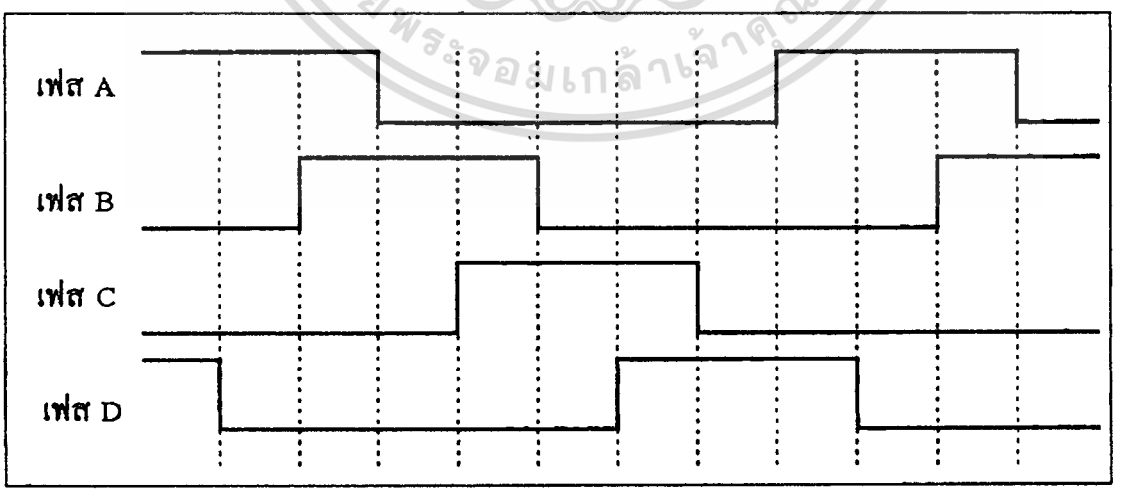
รูปที่ 3.3 แสดงการกระตุ้นแบบ 2 เฟส โดยจะป้อนสัญญาณให้กับขดลวดสเตเตอร์ 2 ขด พร้อมกัน แบบนี้แรงบิดมาก



รูปที่ 4.3 แสดงการกระตุ้นแบบ 2 เฟส

3. การกระตุ้นแบบครึ่งสเตป (half step operation)

รูปที่ 3.4 การกระตุ้นแบบนี้ จะเรียกได้อีกอย่างคือ one-two phase excitation ซึ่งเป็นการกระตุ้นแบบ 1 เฟสและ 2 เฟส สลับกัน ดังรูปที่ 3.4 การกระตุ้นแบบนี้ จำนวนสเตปจะเพิ่มเป็น 2 เท่าของแบบที่ 1 กับแบบที่ 2 แรงบิดเฉลี่ยน้อย



รูปที่ 4.4 แสดงการกระตุ้นแบบครึ่งสเตป

4.3 การนำสเตปปิ้งมอเตอร์มาใช้กับโครงการนี้

จุดประสงค์ของการนำสเตปปิ้งมาใช้ร่วมกับโครงงานนี้ เพื่อนำมาขับกล้องวิดีโอ ให้เคลื่อนที่ตามวัสดุที่ผ่านเข้ามา ซึ่งจะอธิบายได้ว่า เมื่อวัตถุเคลื่อนที่เปลี่ยนตำแหน่งไปจากตำแหน่งเดิม กล้องวิดีโอจะเคลื่อนที่ตาม โดยควบคุมจากสเตปปิ้งมอเตอร์ ส่วนสเตปปิ้งมอเตอร์จะถูกควบคุมการหมุนจาก ตัวควบคุม ซึ่งใช้ไมโครคอนโทรเลอร์ MCS51 อีกทีหนึ่ง รายละเอียดของวงจรดูได้จากบทที่ 6

คุณลักษณะของสเตปปิ้งมอเตอร์ที่ใช้ เป็นแบบขดลวด 4 เฟส มีขนาดเล็ก ใช้ไฟ 12 โวลท์ กระแส 0.42 แอมแปร์ หมุน 1 สเตปต่อ 1 องศา

บทที่ 5
การรับส่งข้อมูลด้วยพอร์ทอนุกรม

5.1 การรับส่งข้อมูลอนุกรมซิงโครนัส

ก่อนที่จะได้เรียนรู้ถึงการทำงานของพอร์ทแบบอนุกรมนั้น ควรจะทำความเข้าใจกับการสื่อสารแบบอะซิงโครนัส (asynchronous) เสียก่อน ในการสื่อสารแบบอะซิงโครนัสนั้นข้อมูลจะถูกส่งผ่านพอร์ทแบบอนุกรมครั้งละ 1 บิต ซึ่งแตกต่างจากการส่งแบบขนานที่จะส่งครั้งละ 1 ไบต์ และระยะเวลาที่ใช้ในการส่งข้อมูลแบบอะซิงโครนัสแต่ละไบต์นั้น ไม่จำเป็นต้องเท่ากันจึงได้ชื่อว่า การส่งข้อมูลแบบอะซิงโครนัส

ในการส่งข้อมูลผ่านพอร์ทแบบอนุกรมข้อมูลแต่ละไบต์จะประกอบด้วย

1. บิตเริ่มต้น(start bit) 1 บิต
2. บิตข้อมูล(data bit) 7 หรือ 8 บิต
3. พาริตี้บิต(parity bit) จะมีหรือไม่มีก็ได้
4. บิตสิ้นสุด(stop bit) 1 หรือ 2 บิต

สถานะของสายส่งในขณะที่ไม่มีข้อมูลจะมีสถานะเป็นลอจิก 1 ข้อมูลใดที่มีค่าเป็นลอจิก 0 จะทำให้สายส่งมีสถานะลอจิกเป็น 0 ตาม ถ้าข้อมูลบิตใดมีลอจิกเป็น 1 ก็จะทำให้สายส่งคงอยู่ในสภาพปกติคือลอจิกเป็น 1 บิตเริ่มต้นใช้สำหรับบอกจุดเริ่มต้นของไบต์ข้อมูล โดยการทำให้สถานะของสายส่งมีลอจิกเป็น 0 เป็นเวลา 1 รอบ(cycle) จากนั้นจะเป็นบิตของข้อมูลตามด้วยพาริตี้บิต ซึ่งจะมีหรือไม่มีก็ได้ สุดท้ายคือบิตสิ้นสุด ซึ่งจะมี 1 บิตหรือ 2 บิตก็ได้

พาริตี้บิตถ้าหากมีในไบต์ข้อมูลจะทำหน้าที่ตรวจเช็คความผิดพลาดของข้อมูล พาริตี้มี 2 อย่างคือ เป็น คู่ หรือ คี่ (even or odd) ถ้าเป็นคู่ หมายความว่า เมื่อรวมพาริตี้บิตแล้วจำนวนของบิตข้อมูลที่มีลอจิกเป็น 1 จะเป็นจำนวนคู่ และถ้าพาริตี้บิตเป็นคี่ เมื่อรวมแล้วจะเป็นจำนวนคี่

อัตราการส่งข้อมูลมีหน่วยเป็น baud (bit per second) ค่า baud rate ที่ต่ำที่สุดที่ใช้กันคือ 300 baud ซึ่งจะใช้กับโมเด็มรุ่นเก่า ส่วนเครื่องคอมพิวเตอร์ระดับ IBM PC สามารถใช้ค่า baud rate ได้สูงถึง 9600 baud

5.2 มาตราฐาน RS-232

การที่จะเข้าใจว่าปัญหามากมายที่เกิดกับพอร์ทอนุกรมนั้นเกิดขึ้นได้อย่างไร และทำไมจึงเกิดขึ้นได้ จะต้องเข้าใจมาตราฐานของการสื่อสารอนุกรมอะซิงโครนัสของ RS-232 มากพอ

สมควร พอร์ทอนุกรมส่วนใหญ่จะมีรูปร่างขึ้นอยู่กับมาตราฐานของ RS-232 คือ มี 25 ขาและ 9 ขา ถึงแม้จำนวนขาจะเป็นมาตราฐานเดียวกันหมด แต่จะมีขาสัญญาณไม่เหมือนกันทั้งหมด เพราะว่าบางสัญญาณไม่จำเป็นต้องใช้ บริษัทผู้ผลิตบางบริษัทจึงตัดออกเพื่อลดค่าใช้จ่าย สัญญาณพื้นฐานของ RS-232 ดังแสดงในตารางที่ 5.1

สัญญาณ	ชื่อย่อ	หมายเลขขา
Request to send	RTS	4
Clear to send	CTS	5
Data set ready	DSR	6
Data terminal ready	DTR	20
Transmit data	TxD	2
Receive data	RxD	3
Ground	GND	7

ตารางที่ 5.1

สัญญาณทั้งหมดมีมากกว่านี้ เพราะว่าแรกเริ่มนั้นพอร์ทอนุกรมถูกออกแบบมาเพื่อใช้ร่วมกับโมเด็ม ดังนั้นเมื่อนำไปใช้ร่วมกับอุปกรณ์อื่นบางสัญญาณจึงไม่จำเป็น เพราะสัญญาณเหล่านี้มีเพื่อใช้เป็นข้อตกลงระหว่างโมเด็มกับคอมพิวเตอร์ว่า

1. คอมพิวเตอร์จะไม่ส่งข้อมูลให้แก่โมเด็ม ก่อนที่โมเด็มจะพร้อมส่งข้อมูล
2. คอมพิวเตอร์จะไม่อ่านข้อมูลจากโมเด็ม ก่อนที่โมเด็มจะพร้อม

5.2.1 ความผิดพลาดของกรอบข้อมูล

ความผิดพลาดของกรอบข้อมูล(framing error) คือ ความผิดพลาดของการส่งข้อมูลที่เกิดจากสัญญาณนาฬิกา(clock) ที่ควบคุมการทำงานของอุปกรณ์ทั้ง 2 ด้านไม่เท่ากัน เพราะว่าจากการทำงานของพอร์ทอนุกรม เมื่อพอร์ทได้รับบิตเริ่มต้นก็จะสุ่มอ่านค่าจากส่วนรับข้อมูล 1 ครั้งต่อ 1 รอบ เพื่ออ่านบิตต่อไป ซึ่งระยะเวลาในการสุ่มอ่านแต่ละรอบกำหนดได้จาก baud rate ถ้าหากว่าคอมพิวเตอร์ทั้งสองเครื่องมีสัญญาณนาฬิกาไม่ตรงกัน คอมพิวเตอร์ด้านรับก็จะอ่านข้อมูลจากส่วนรับข้อมูลของตนช้าเกินไปหรือเร็วเกินไป ก่อนที่ข้อมูลจะถูกส่งมาจากคอมพิวเตอร์ด้านส่งทำให้เกิดการผิดพลาดขึ้น

5.2.2 ฮาร์ดแวร์แฮนด์เชคกิ้ง

อาร์ดแวร์แฮนด์เชคกิ้ง(hardware handshaking) คือ วิธีที่ใช้ในการรับส่งข้อมูลผ่านพอร์ทอนุกรม โดยจะต้องตรวจดูสถานะของคอมพิวเตอร์ด้านการรับข้อมูลว่าพร้อมจะรับข้อมูลหรือไม่ คอมพิวเตอร์ด้านส่งจะไม่ส่งข้อมูลออกไปจนกว่าสัญญาณพร้อมรับข้อมูลจะส่งกลับมาจากคอมพิวเตอร์ด้านรับ สัญญาณพร้อมรับข้อมูลของคอมพิวเตอร์ด้านรับคือ clear to send (CTS)

หมายความว่า จะมีการตรวจสถานะ CTS ตลอดเวลาว่าพร้อมจะรับข้อมูลหรือไม่ ถ้าไม่พร้อมก็จะรอไปเรื่อยๆ แต่ถ้าพร้อมก็จะส่งข้อมูลไปให้ จะทำเช่นนี้ไปเรื่อยๆตลอดที่ยังมีข้อมูลอยู่

5.3 ปัญหาของการสื่อสาร

เพื่อที่จะให้การสื่อสารเป็นไปอย่างถูกต้อง สัญญาณหลายๆ สัญญาณถูกใช้เพื่อตรวจสอบว่า ข้อมูลจะพร้อมเมื่อใดหรือข้อมูลไบต์ต่อไปจะส่งมาเมื่อใด แต่ต่อมาเมื่อมีการสื่อสารผ่านคอมพิวเตอร์ สัญญาณบางสัญญาณได้ถูกตัดทิ้งไปเพื่อจะได้ลดน้อยลงและลดค่าใช้จ่ายเกี่ยวกับสายส่ง สัญญาณจึงเหลือเพียง GRD,TxD และ RxD ซึ่งในทางทฤษฎีแล้วการรับส่งข้อมูลแบบอนุกรมจะต้องมีการตรวจเช็คความพร้อมเพื่อความถูกต้องของข้อมูล ดังนั้นจึงเกิดปัญหาขึ้น ซึ่งเป็นปัญหาของข้อมูลถูกเขียนทับ(overrun error)

5.3.1 ปัญหาข้อมูลถูกเขียนทับ

เมื่อการติดต่อผ่านพอร์ทอนุกรมใช้สายส่งเพียง 2 เส้นนั้น จะต้องใช้วิธีที่พิเศษเล็กน้อยเพื่อให้พอร์ทตัวส่งเข้าใจว่าพอร์ทด้านรับพร้อมที่จะรับข้อมูลเสมอ โดยการต่อขา 6, 8 และ 20 ของคอนเน็คเตอร์เข้าด้วยกัน วิธีการนี้เป็นการตัดฮาร์ดแวร์แฮนด์เชคกิ้งออกนั้นเอง แต่การทำเช่นนี้จะทำให้เกิดปัญหาข้อมูลถูกเขียนทับได้ง่าย ซึ่งก็คือความผิดพลาดในการรับส่งข้อมูลอย่างหนึ่งที่เกิดจากการที่คอมพิวเตอร์เครื่องส่งๆข้อมูลใหม่มาให้คอมพิวเตอร์ด้านรับ ในขณะที่คอมพิวเตอร์ด้านรับไม่พร้อมจะรับข้อมูล เนื่องจากในขณะนั้นข้อมูลเดิมยังไม่ได้ถูกอ่านเข้าไปเก็บ แต่ข้อมูลใหม่ก็ถูกส่งมาแล้วข้อมูลเดิมจึงถูกเขียนทับไป จึงทำข้อมูลเกิดความผิดพลาด

5.4 การใช้งานพอร์ทอนุกรมผ่าน BIOS

การใช้งานพอร์ทอนุกรมสามารถทำได้ 3 วิธีคือ ผ่านทาง BIOS ผ่านทางดอสและการเขียนโปรแกรมควบคุมโดยตรง

การเรียกใช้พอร์ทอนุกรมผ่านทางดอสนั้น ไม่เหมาะสมเท่าใดนักเพราะว่าดอสไม่มีวิธีที่จะตรวจดูสถานะของการรับส่งว่าการรับส่งถูกต้องหรือไม่เพียงใด

การเขียนโปรแกรมควบคุมโดยตรงนั้นคงไม่จำเป็น เพราะมีวิธีที่ดีกว่าคือ การเรียกใช้ผ่าน BIOS อินเตอร์รัพท์หมายเลข 14

5.4.1 การเตรียมสถานะเริ่มต้นของพอร์ท

ในการเตรียมสถานะของพอร์ทอนุกรมเราสามารถทำได้ โดยผ่านอินเตอร์รัพท์หมายเลข 14 ฟังก์ชั่นหมายเลข 0 โดยมีรีจิสเตอร์ AH เป็นตัวผ่านค่าหมายเลขของฟังก์ชั่น รีจิสเตอร์ AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ทอนุกรม โดยความหมายของแต่ละบิตดังแสดงในตารางที่ 5.2

หมายเลขประจำบิต	ความหมาย
7, 6, 5	อัตรารับส่งข้อมูล
	000 = 110 baud
	001 = 150 baud
	010 = 300 baud
	011 = 600 baud
	100 = 1200 baud
	101 = 2400 baud
	110 = 4800 baud
	111 = 9600 baud
4, 3	พาริตี้
	00 หรือ 10 = ไม่พาริตี้
	01 = พาริตี้คี่
	11 = พาริตี้คู่
2	จำนวนของบิตสิ้นสุด
	0 = 1 บิตสิ้นสุด
	1 = 2 บิตสิ้นสุด
	จำนวนบิตในข้อมูล 1 ไบต์
	10 = 7 บิต
1, 0	11 = 8 บิต

ตารางที่ 5.2

ตัวอย่างการใช้รหัสเพื่อเตรียมสถานะเช่น ถ้าต้องการตั้งให้พอร์ทมีอัตรา 9600 baud มีพาริตี้คู่ มี 1 บิตสิ้นสุดและใช้ 8 บิตต่อ 1 ไบต์ จะได้รูปแบบของรหัสดังตารางที่ 5.3

บิตที่	7	6	5	4	3	2	1	0
รหัส	1	1	1	1	1	0	1	1

ตารางที่ 5.3

เครื่องคอมพิวเตอร์โดยทั่วไปจะมีพอร์ทอนุกรมได้มากถึง 7 พอร์ท โดยหมายเลขพอร์ทที่จะใช้สามารถกำหนดผ่านรีจิสเตอร์ DX พอร์ทแรกเป็นหมายเลข 0 พอร์ทต่อไปหมายเลข 1 เป็นเช่นนี้ต่อไปเรื่อยๆ ฟังก์ชั่นที่แสดงต่อไปนี้ชื่อว่า init_port() มีหน้าที่เตรียมสถานะของพอร์ท

```
void init_port(port,code)
int port;
unsigned char code;
{
   union REGS r;
   r.x.dx = port;          /* หมายเลขพอร์ทอนุกรม */
   r.h.ah = 0;             /* เรียกฟังก์ชั่นหมายเลข 0 */
   r.h.al = code;          /* รหัสตั้งสถานะเริ่มต้น */
   int86(0x14,&r,&r);
}
```

5.4.2 การส่งข้อมูลผ่านพอร์ทอนุกรม 1 ไบต์

อินเตอร์รัพท์หมายเลข 14 ฟังก์ชั่นหมายเลข 1 ของ BIOS ทำหน้าที่ส่งข้อมูล 1 ไบต์ออกทางพอร์ทอนุกรม หมายเลขของพอร์ทอนุกรมที่จะทำการส่งข้อมูลสามารถกำหนดผ่านรีจิสเตอร์ DX และข้อมูลที่ต้องการส่งจะอยู่ในรีจิสเตอร์ AL เมื่อทำการส่งเรียบร้อยแล้ว สถานะของการส่งข้อมูลจะปรากฏในรีจิสเตอร์ AH เพื่อใช้สำหรับตรวจดูว่า การส่งข้อมูลถูกต้องหรือไม่ รายละเอียดของฟังก์ชั่น sport() มีดังนี้

```
void sport(port,c)
int port;
char c;
{
   union REGS r;
   r.x.dx = port;          /* กำหนดหมายเลขพอร์ท */
   r.h.al = c;             /* ตัวอักษรที่ต้องการส่ง */
   r.x.ah = 1;             /* ฟังก์ชั่นหมายเลข 1 ทำหน้าที่ส่งข้อมูล 1 ไบต์ */
   int86(ox14,&r,&r);
   if(r.h.ah & 128)        /* ตรวจดูบิตที่ 7 */
   {
      printf("send error deteced in serial port");
```

```
        exit(1);
    }
}
```

ถ้าบิต 7 ของรีจิสเตอร์ AH มีค่าเป็น 1 เมื่อส่งข้อมูลเสร็จสิ้นแสดงว่าเกิดความผิดพลาดในการส่งข้อมูลขึ้น ซึ่งเมื่อเป็นเช่นนี้จะต้องมีการตรวจดูสถานะของพอร์ทเพื่อดูว่าเกิดความผิดพลาดอะไร ดังจะกล่าวต่อไป

5.4.3 การตรวจสอบสถานะของพอร์ทอนุกรม

ฟังก์ชั่นหมายเลข 3 ของอินเตอร์รัพท์หมายเลข 14 ของ BIOSใช้ตรวจดูสถานะของพอร์ทอนุกรม รีจิสเตอร์ DX ใช้สำหรับหมายเลขพอร์ทที่จะตรวจสอบ สถานะของพอร์ทสามารถแปลความหมายได้จากรหัสที่ปรากฏในรีจิสเตอร์ AH และ AL ดังตารางที่ 5.4

สถานะของ AH

ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Data Ready	0
Overrun Error	1
Parity Error	2
Framing Error	3
Break-Detect Error	4
Transfer hold-register empty	5
Transfer shift-register empty	6
Time-out Error	7

สถานะของ AL

ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Change in clear-to-send	0
Change in data-set-ready	1
Trailing-edge ring dectect	2
Change in line signal	3
Clear-to-send	4
Data-set-ready	5
Ring indicator	6
Line signal detect	7

ตารางที่ 5.4

จะเห็นว่าสัญญาณสถานะต่างๆ ส่วนใหญ่จะใช้งานกับโมเด็ม ดั้งนั้นเมื่อนำมาใช้งานกับอุปกรณ์อื่นจึงลดความสำคัญลง แต่ก็ยังมีสัญญาณอีกสัญญาณหนึ่งที่มีความสำคัญมากก็คือ Data Ready ซึ่งจะเป็นตัวบอกว่า เมื่อไหร่ที่ข้อมูลถูกรับเข้ามาและพร้อมที่จะถูกอ่านเข้าไปเก็บ ฟังก์ชั่นในหัวข้อต่อไปนี้ชื่อว่า rport() มีหน้าที่อ่านข้อมูลจากพอร์ท โดยจะใช้สถานะ Data Ready นี้เป็นตัวบอกว่าข้อมูลพร้อมที่จะอ่านหรือยัง

5.4.4 การรับข้อมูลผ่านพอร์ทอนุ กรม 1 ไบต์

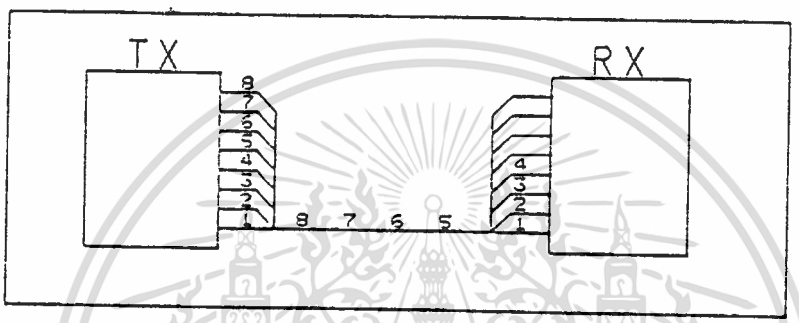
การรับข้อมูลจากพอร์ทอนุกรม สามารถทำได้โดยเรียกผ่าน BIOS อินเตอร์รัพท์หมายเลข 14 ฟังก์ชั่นหมายเลข 2 รีจิสเตอร์ DX ใช้สำหรับกำหนดหมายเลขพอร์ท ข้อมูลที่อ่านได้จะเก็บในรีจิสเตอร์ AL เมื่ออ่านข้อมูลเสร็จแล้ว สถานะของข้อมูลและพอร์ทจะสามารถดูได้ที่บิต 7 ของรีจิสเตอร์ AH ฟังก์ชั่น rport() มีดังต่อไปนี้

```
rport(port)
int port;
{
   union REGS r;
   while(!(check_stat(port)&256))   /* รอจนกว่าจะได้รับตัวอักษร */
     if(kbhit())                    /* ยกเลิกเมื่อมีการกดคีย์ */
     {
       getch();
       exit(1);
     }
   r.x.dx = port;                   /* กำหนดหมายเลขพอร์ท */
   r.h.ah = 2;                      /* ฟังก์ชั่นหมายเลข 2 ทำหน้าที่อ่านตัวอักษร */
   int86(0x14,&r,&r);
   if(r.h.ah & 128)
     printf("read error detected in serial port");
   return (r.h.al);
}
```

การทำงานของฟังก์ชั่นนี้คือจะรอจนกระทั่งข้อมูลถูกรับมาผ่านพอร์ทอนุกรมแล้วส่งค่าตัวอักษรกลับมา แต่ว่าการทำงานเช่นนี้อาจทำให้โปรแกรมไม่หลุดจากลูปการทำงาน เช่น เมื่อสายส่งสัญญาณของพอร์ทเกิดเสีย ดังนั้นจึงต้องมีการตรวจสอบสถานะของพอร์ทอนุกรมเสียก่อน ดังที่ได้กล่าวในหัวข้อที่ผ่านมา ฟังก์ชั่น kbhit() ใช้ตรวจสอบการกดคีย์ ถ้าหากไม่มีข้อมูลผู้ใช้ก็สามารถกดคีย์ใดๆ เพื่อออกจากลูปได้ แต่ถ้ามีข้อมูลรับเข้ามา ฟังก์ชั่นก็จะผ่านไปเรียกอินเตอร์รัพท์เพื่ออ่านข้อมูลเข้ามา และเช่นเดียวกับการส่งข้อมูล บิต 7 ของรีจิสเตอร์ AH ใช้บอกว่าการอ่านข้อมูลมีข้อผิดพลาดหรือไม่

5.5 การสื่อสารอนุกรม RS-232 ใน MCS-51

การสื่อสารข้อมูลแบบ อนุกรม เป็นการส่ง หรือรับข้อมูลครั้งละ 1 บิต การส่งข้อมูล หรือรับข้อมูลแบบอนุกรมจะทำโดยการนำข้อมูลที่เป็นแบบขนานมาทำการเลื่อน (Shift) ข้อมูลไปทางซ้าย หรือ ขวา รูปที่ 5.1 จะแสดงการรับส่งข้อมูลแบบอนุกรม



รูปที่ 5.1

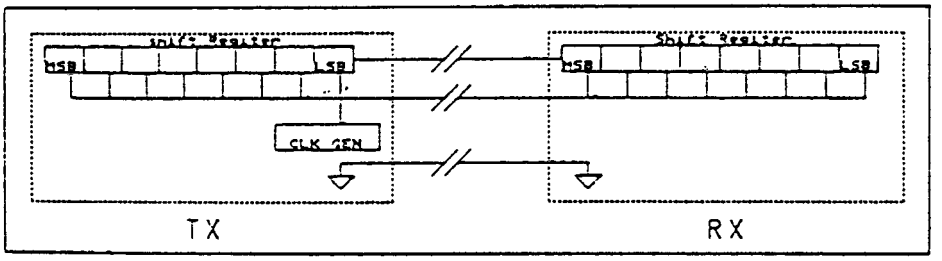
จากรูปที่ 5.1 แสดงให้เห็นการส่งข้อมูลแบบอนุกรม ข้อมูลจากจุดส่งจะถูกเปลี่ยนให้เป็นแบบ อนุกรมก่อนที่จะถูกส่งออกไป จะทยอยส่งออกทีละ 1 บิตจนครบทุกบิตที่จะทำการส่งถือว่าจบการส่งข้อมูลแบบอนุกรม ส่วนของการรับข้อมูลแบบอนุกรมนั้นจะมีกลไกในการรับข้อมูลแบบอนุกรม โดยการรับนั้นจะถูกเลื่อนข้อมูลเข้ามาทีละ 1 บิตจนครบทุกบิตที่ถูกส่งมาถือว่าจบการรับข้อมูลแบบอนุกรม

5.5.1 รูปแบบการรับส่งแบบอนุกรม

แบบ ซิงโครนัส (Synchronous)

เป็นการส่งข้อมูลแบบอนุกรม ซึ่งจะอาศัยสัญญาณความถี่ที่จุดส่งมาทำการเลื่อนข้อมูลที่ส่งออกมาดังจะแสดงดังรูปที่ 5.2

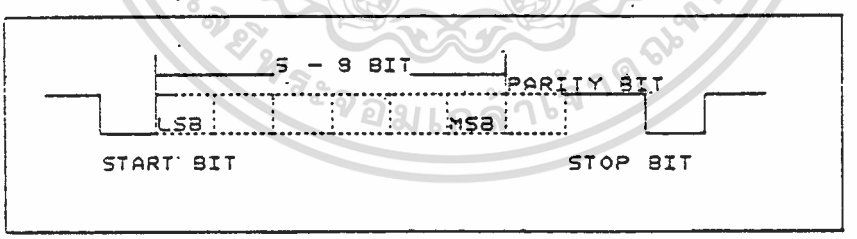
จากรูปที่ 5.2 จะเห็นการส่งข้อมูลแบบอนุกรมแบบซิงโครนัส (Synchronous) นั้นจะต้องใช้สายสัญญาณในการติดต่อ 3 สายด้วยกัน คือ สายของข้อมูล,สาย Clock และ สาย GND ซึ่งในการส่ง หรือ รับข้อมูลแบบนี้จะถูกควบคุมการส่ง และ การรับ โดยสัญญาณ Clock



รูปที่ 5.2 การส่งข้อมูลอนุกรมแบบ Synchronous

แบบ อะซิงโครนัส (ASynchronous)

เป็นการส่งข้อมูลที่ไม่ต้องใช้สัญญาณ Clock ในการควบคุมในการส่ง หรือ รับข้อมูลแบบ อนุกรมเพียงแต่ต้องเพิ่มข้อมูลเข้าไปอีก 2 - 3 บิต โดยที่บิตที่ถูกเพิ่มมานั้นจะเป็นสัญญาณการควบคุมการส่ง และรับข้อมูลแบบอนุกรม ซึ่งบิตที่เพิ่มเข้ามานั้นได้แก่ Start Bit,Stop Bit และ Parity Bit ความหมายของสัญญาณดังกล่าวนั้นจะเป็นตัวกำหนดจุดเริ่มต้นของข้อมูล (Start Bit) กำหนดจุดจบของข้อมูล (Stop Bit) และ ตรวจสอบข้อมูล (Parity Bit) จะแสดงในรูปที่ 5.3



รูปที่ 5.3 การส่ง และรับข้อมูลแบบ อนุกรม แบบ ASynChronous

จากรูปที่ 5.3 เป็นการส่งข้อมูลขนาด 8 บิต โดยจะมีบิตที่เพิ่มเติมเพื่อเป็นตัวในการสัมพันธ์กันระหว่าง ตัวส่ง และ ตัวรับ

BAUD RATE คือ อัตราของการส่งข้อมูลแบบอนุกรม เทียบกับเวลา 1 วินาที เช่น ส่งข้อมูลแบบอนุกรม BAUD RATE 9600 หมายถึงการส่งข้อมูลทั้งหมด 9600 Bit ต่อ 1 วินาที ซึ่งในการส่งข้อมูลแบบอนุกรมจะส่ง BAUD RATE ได้ 110,1200,2400,4800 และ 9600 เป็นต้น

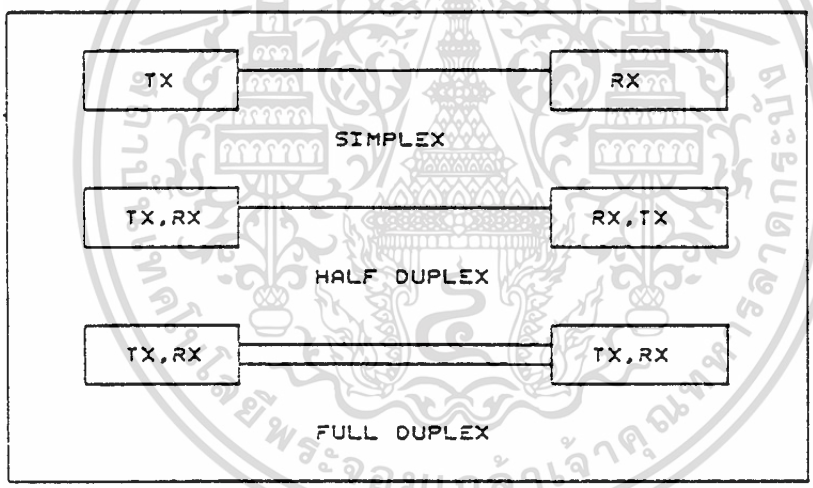
การรับ และส่งข้อมูลแบบอนุกรมแบบตามลักษณะการส่งและรับข้อมูลได้ 3 แบบด้วยกัน คือ

1. แบบซิมเพลกซ์ (Simplex) เป็นการรับส่งข้อมูลแบบทางเดียวเท่านั้นบางครั้งก็เรียกว่าการส่งแบบนี้ว่าการส่งทิศทางเดียว

2. แบบ ฮาล์ฟดูเพลกซ์ (Half Duplex) เป็นการส่งและรับข้อมูลได้ทั้ง 2 สถานี ในสายส่งและรับข้อมูลในเส้นทางเดียว ซึ่งการรับ และ ส่งของข้อมูลนั้นจะต้องสลับกัน คือ ถ้าด้านหนึ่งส่งอีกด้านหนึ่งต้องเป็นตัวรับ

3. แบบฟลูดูเพลกซ์ (Full Duplex) เป็นการส่ง และ รับข้อมูลในเวลาเดียวกัน ซึ่งจำเป็นในการใช้สายของข้อมูล 2 สายในการส่งและรับข้อมูลแบบนี้

ในการรับส่งข้อมูลแบบต่าง ๆ จะแสดงในรูปที่ 5.4



รูปที่ 4 การส่ง และ รับข้อมูลแบบต่าง ๆ

ในตัว Micro Controller MCS-51 มีพอร์ตอนุกรมให้ใช้ เป็นแบบ Full Duplex คือ สามารถส่งและรับข้อมูลแบบ อนุกรมในเวลาเดียวกัน ซึ่งในพอร์ทอนุกรมของ Micro Controller MCS-51 มีรีจิสเตอร์อยู่ด้วย 2 ตัวที่ใช้ในการติดต่อข้อมูลแบบอนุกรมคือ

SBUF เป็นรีจิสเตอร์ที่ใช้เป็นตัวเก็บข้อมูลที่จะทำการส่ง หรือ รับข้อมูลแบบอนุกรม ซึ่งภายในตัว MCS-51 นี้จะแยกส่วนข้อ SBUF เป็นส่วนของด้านรับและด้านส่ง ออกจากกัน ข้อควรระวังในการใช้การอ่านข้อมูลจาก SBUF นี้จะต้องอ่านก่อนที่ข้อมูลถัดไปจะถูกส่งเพราะถ้าไม่ทำการอ่านข้อมูลที่ เข้ามาก่อนนั้นจะทำให้ข้อมูลที่ส่งต่อมาจะทับข้อมูลตัวที่เข้ามาก่อนนั้นทำให้ข้อมูลนั้นสูญหายไป

SCON เป็นรีจีสเตอร์ที่ใช้ในการกำหนดควบคุม การทำงานของการส่งข้อมูลแบบ อนุกรม ซึ่งความหมายของแต่ละบิตใน SCON มีดังต่อไปนี้

SM0,SM1 เป็นบิตที่จะใช้ในการเลือกโหมดของการทำงานของ การรับและ ส่งข้อมูล แบบอนุกรม ดังนี้

SM0	SM1	MODE	ลักษณะการทำงาน	อัตราของ BAUD RATE
0	0	0	ชิงโครนัส	Fosc/12
0	1	1	8 บิต อะซิงโครนัส	เปลี่ยนไปตามตัวจับเวลา
1	0	2	9 บิต อะซิงโครนัส	Fosc/64 fosc/32
1	1	3	9 บิต อะซิงโครนัส	แปรผัน

REN เป็นตัวอีนาเบิ้ลการรับข้อมูลแบบอนุกรม

1 = อีนาเบิ้ลการรับข้อมูลแบบอนุกรม

0 = ดิสเอเบิ้ลการรับข้อมูลแบบอนุกรม

TR8 เป็นข้อมูลบิตที่ 9 ซึ่งจะถูกส่งในโหมด 2 และ 3 ซึ่งจะให้เป็น "1" หรือ "0" สามารถเปลี่ยนด้วยโปรแกรม

RB8 เป็นข้อมูลบิตที่ 9 ที่ถูกรับเข้ามา ในโหมด 2 และ 3

โหมด 0 จะไม่ใช้

โหมด 1 ถ้า SM2 = 0 จะกลายเป็น STOP BIT ที่ถูกรับไป

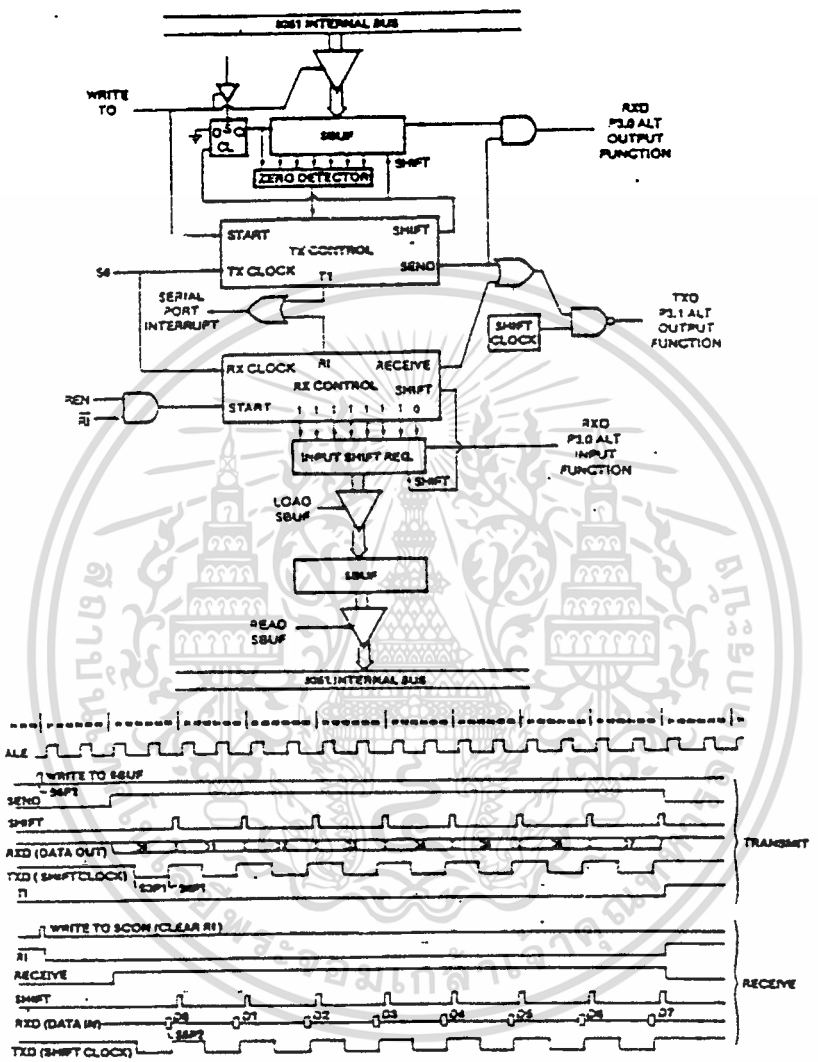
TI เป็นแฟลกอินเตอร์รัพต์ในการส่ง จะ SET ด้วยฮาร์ดแวร์ คือสัญญาณที่ปลายช่วงเวลาของบิตที่ 8 ในโหมด 0 หรือจุดเริ่มต้นของ STOP BIT ในโหมดอื่น ในการส่งข้อมูลแบบอนุกรมของทุกโหมดจะต้องทำการ เคลียร์ บิตนี้ด้วยโปรแกรมหลังจากการส่ง

RI เป็นแฟลกอินเตอร์รัพท์ในการรับ จะ SET ด้วยฮาร์ดแวร์ คือสัญญาณที่ปลายช่วงเวลาเวลาของบิตที่ 8 ในโหมด 0 หรือจุดครึ่งทางของ STOP BIT ในโหมดอื่น ในการรับข้อมูลแบบอนุกรมของทุกโหมดจะต้องทำการ เคลียร์ บิตนี้ด้วยโปรแกรมหลังจากการรับ

ในพอร์ตอนุกรมของ MSC-51 นั้นสามารถกำหนด การทำงาน ของการรับ และส่งข้อมูลแบบ อนุกรมได้ 4 Mode ด้วยกัน

MODE 0 เป็นการรับส่งข้อมูลแบบ ชิงโครนัส ซึ่งจะรับ และ ส่งข้อมูลขนาด 8 บิต โดยผ่านขา RXD และ ใช้สัญญาณ TXD เป็นสัญญาณในการเลื่อนข้อมูลในการรับ โดยจะทำการเลื่อน

บิต LSB (DO) เป็นตัวแรก อัตรา BUAD RATE คงที่ 1/12 ของความถี่ที่ป้อนให้กับตัว Micro Controller MCS-51 การทำงานของการส่งและรับข้อมูล ในโหมด 1 จะแสดงในรูปที่ 5.5

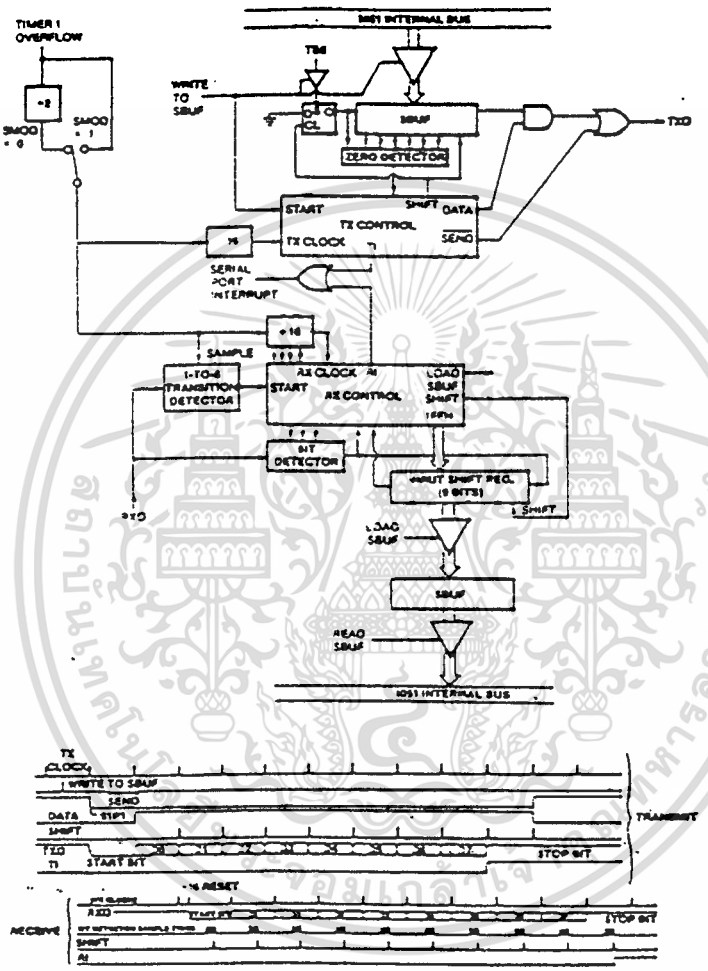


รูปที่ 5.5

จากรูปที่ 5.5 จะเห็นว่าการรับและการส่งข้อมูลแบบอนุกรมในโหมดนี้จะใช้ขา RXD เป็นตัวรับข้อมูล และส่งข้อมูล ส่วน TXD จะใช้เป็น Clock ในการควบคุมการรับส่ง

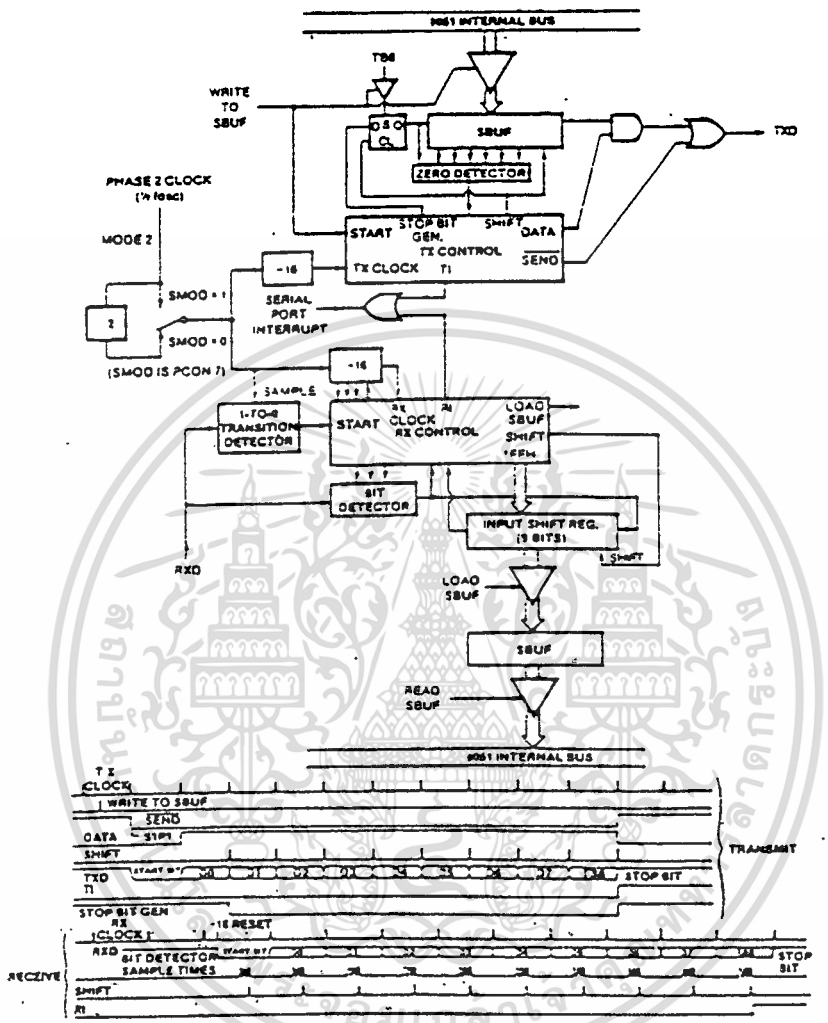
MODE 1 เป็นการรับ และ ส่งข้อมูลแบบ อะซิงโครนัส ซึ่งจะรับและส่งข้อมูลขนาด 8 บิต โดยจะมี 2 บิตเพิ่มเติมคือ Start Bit และ Stop Bit ซึ่งรวมกับบิตของข้อมูลก็ได้ทั้งหมด 10 บิต การรับ Stop Bit จะถูกส่งเข้าไปยัง RB2 ของ SCON การตั้ง BAUD RATE จะขึ้นอยู่กับการตั้งตัว

จับเวลาที่ 1 หรือ 2 อาจใช้สัญญาณ Clock การทำงานของการรับ และ ส่งข้อมูลแบบอนุกรมใน โหมด 1 จะแสดงในรูปที่ 5.6



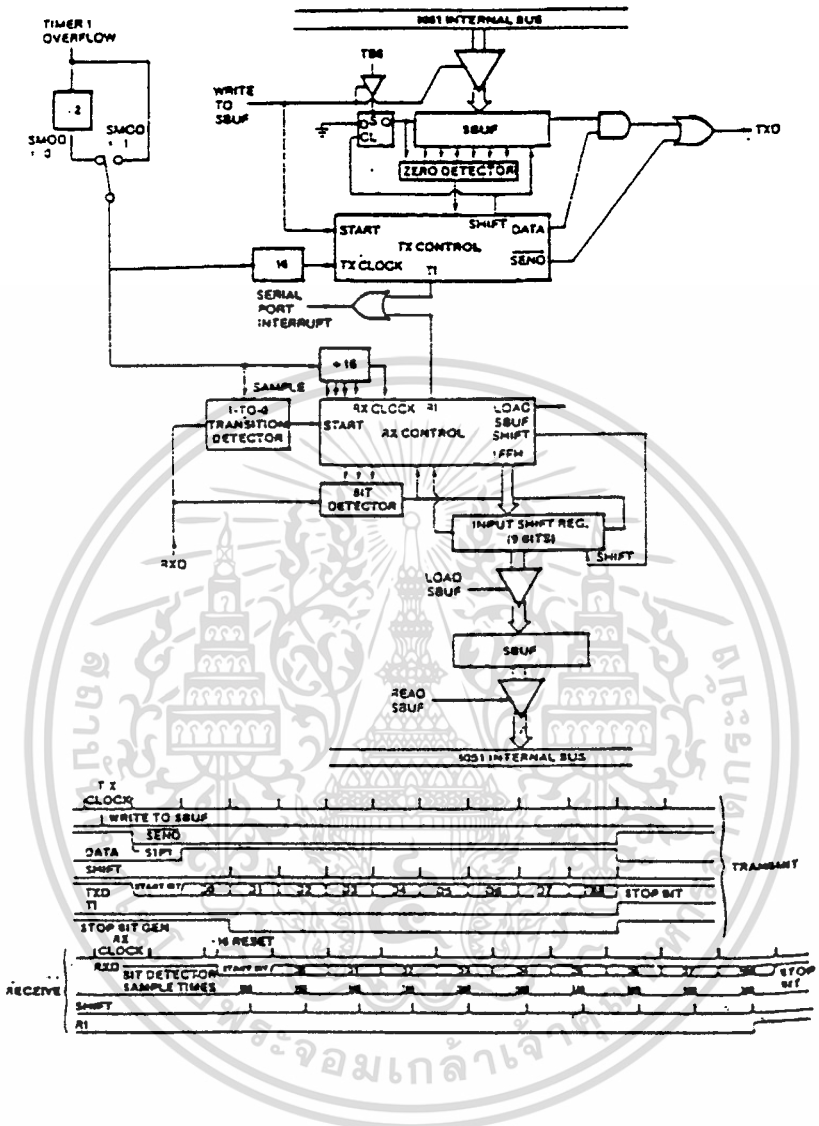
รูปที่ 5.6

MODE 2 เป็นการรับ และ ส่งข้อมูลแบบ อะซิงโครนัส ซึ่งจะรับและส่งข้อมูลขนาด 8 บิต โดย จะมี 3 บิตเพิ่มเติมคือ บิตที่ 9 (TB8 ในการส่งข้อมูล และ RB8 ในการรับข้อมูล) Start Bit และ Stop Bit รวมกับบิตของข้อมูลก็ได้ทั้งหมด 11 บิต การรับ Stop Bit การตั้ง BAUD RATE สามารถ โปรแกรมได้ทั้งแบบ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ในโหมด 2 การทำงาน ของการรับ และ ส่งข้อมูลแบบ อนุกรมจะแสดงในรูปที่ 5.7



รูปที่ 5.7

MODE 3 การทำงานจะคล้ายโหมด 2 แต่สามารถตั้ง BAUD RATE โดยจะใช้ตัวแปรหลายค่าของ BAUD RATE ให้ตัวจับเวลาที่ 1 หรือ 2 ขึ้นอยู่กับสถานะ TCLK และ RCLK การทำงานในโหมด 3 แสดงในรูปที่ 5.8



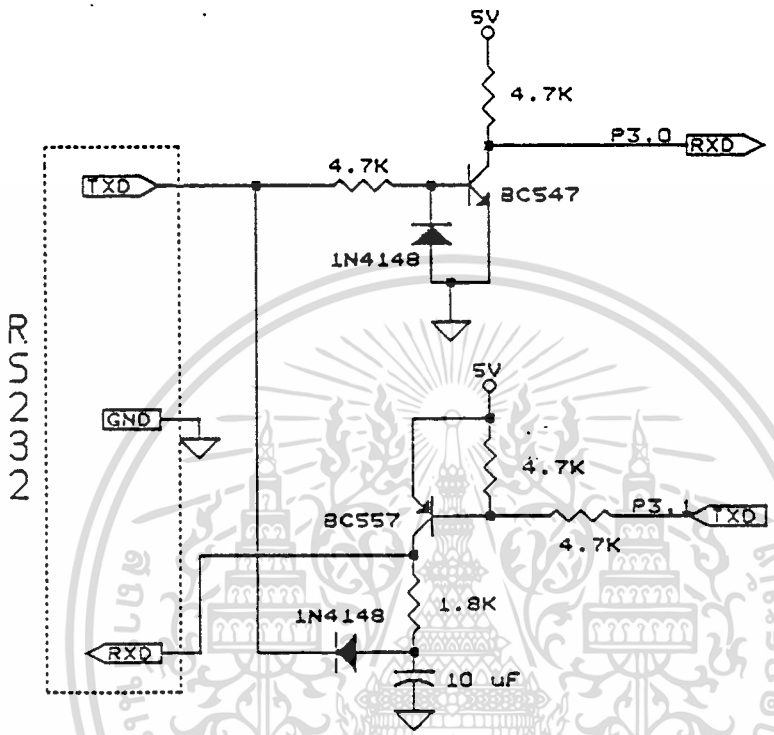
รูปที่ 5.8

การใช้ตัวจับเวลา (Timer/Counter) เป็นตัวสร้าง BAUD RATE สามารถใช้ตารางในการสร้าง BAUD RATE

5.6 การใช้วงจรในการ Drivers ในการติดต่อ RS-232

ในการต่อกันระหว่าง พอร์ต แบบอนุกรมของ MCS-51 จำเป็นต้องวงจรในการปรับระดับของแรงไฟมาตรฐานของการสื่อสาร คือ ± 12 ถึง ± 15 V มี Buffer ที่จะใช้นิยมในการต่อได้แก่ ไอซีเบอร์ MC1488 ในการส่งข้อมูลไปที่ RS-232 และ เบอร์ MC1489 ในการแปลงระดับสัญญาณจาก RS-232 ซึ่งการใช้ Buffer นี้จำเป็นต้องใช้แรงดันไฟในการขับ Buffer เบอร์นี้ดังนั้น เรา

สามารถต่อด้วยวงจรที่ใช้ Transistor ในการแปลงสัญญาณโดยเราจะอาศัยแรงไฟจาก RS-232 มาใช้งาน ซึ่งการวงจรตามรูปที่ 5.9



รูปที่ 5.9 วงจรในการแปลงสัญญาณจาก RS-232 โดยใช้ Transistor

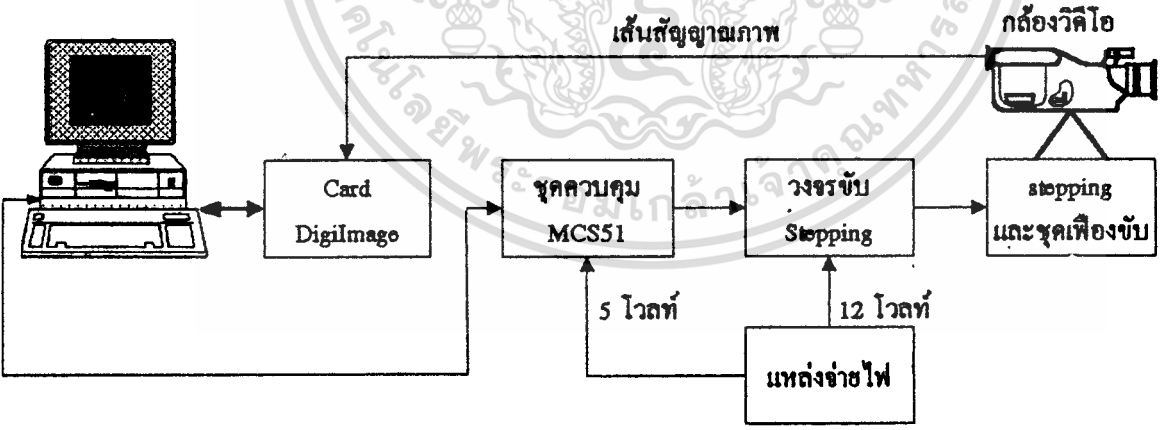
บทที่ 6

รายละเอียดการทำงานและการทดลอง

6.1 อุปกรณ์ที่ใช้ในการทดลอง

1. คอมพิวเตอร์ส่วนบุคคล รุ่น 486DX2-66 จอภาพ VGA สี
2. Card DigiIMAGE
3. ชุดควบคุม ไมโครคอนโทรลเลอร์ MCS-51
4. EPROM EMURATON
5. กล้องวิดีโอ ขนาดเล็ก ถ่ายภาพขาว-ดำ
6. ชุดจ่ายไฟให้สเตปปิ้งมอเตอร์
7. ชุดขับสเตปปิ้งมอเตอร์

6.2 การต่อใช้งานและการทำงานของเครื่อง



รูปที่ 6.1 แสดงการต่อใช้งาน

จากวงจรการต่อจะมีสายสัญญาณอยู่ 2 เส้น คือ สายสัญญาณข้อมูลภาพ ต่อจากกล้องมาเข้า Card DigiIMAGE ซึ่งเป็นสัญญาณอินพุทของคอมพิวเตอร์ และสายสัญญาณควบคุมสเตปปิ้งมอเตอร์ ซึ่งเป็นสัญญาณเอาท์พุทของคอมพิวเตอร์ ที่จะต้องต่อออกจากพอร์ท RS-232 ไปเข้าชุดควบคุม MCS-51 จากนั้นก็ต่อเข้าชุดอื่น ๆ ต่อไป ตามรูปที่ 6.1

การทำงานอย่างคร่าว ๆ ของเครื่องและการทำงานของโปรแกรมได้กล่าวถึงไปบ้างแล้วในบทที่ 1 ในบทนี้จะกล่าวถึงขั้นตอนโดยละเอียด โดยแบ่งออกเป็น 2 ส่วน คือ โปรแกรม (Software) และตัวเครื่อง (Hardware)

6.2.1 การทำงานของโปรแกรม

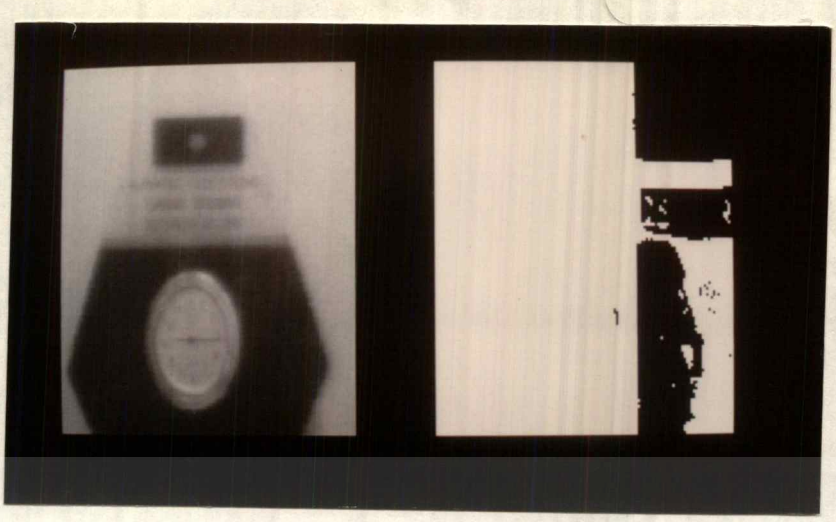
ส่วนของโปรแกรมเป็นการรวมคำสั่งต่าง ๆ ที่ใช้ในการควบคุมการทำงานของตัวเครื่องตามจุดประสงค์ที่ตั้งไว้ แบ่งออกเป็น 2 โปรแกรม

1. โปรแกรมบนคอมพิวเตอร์
2. โปรแกรมบนไมโครคอนโทรลเลอร์ MCS-51

โปรแกรมบนคอมพิวเตอร์

เป็นคำสั่งที่ใช้ในการประมวลผลการเปรียบเทียบภาพที่เปลี่ยนแปลงตามการเคลื่อนที่ของวัตถุ โดยจะสั่งงานให้คอมพิวเตอร์รับข้อมูลจาก Card DigiIMAGE มา 2 ครั้ง ข้อมูลครั้งแรกเป็นข้อมูลภาพต้นแบบ (reference) ข้อมูลครั้งที่สองเป็นภาพที่ถ่ายต่อจากภาพต้นแบบ นำข้อมูลภาพทั้งสองนั้นมาเปรียบเทียบกัน ซึ่งอาจจะเป็นภาพที่เหมือนกับภาพต้นแบบหรืออาจจะไม่เหมือนขึ้นอยู่กับว่าในการถ่ายภาพครั้งที่สองในขณะนั้นมีวัตถุอื่นเคลื่อนที่ผ่านเข้ามาหรือไม่ ข้อมูลใหม่ที่ได้จากการเปรียบเทียบจึงมี 2 ลักษณะคือ ถ้าภาพทั้งสองเหมือนกันจะได้ภาพใหม่เป็นสีขาวหมด แต่ถ้าภาพทั้งสองไม่เหมือนกัน ตรงบริเวณตำแหน่งภาพที่แตกต่างกันจะเป็นสีดำ บริเวณที่เหมือนกันจะเป็นสีขาว ดังแสดงในรูปที่ 6.2 และภาพที่เหมือนกันทั้งภาพดังรูปที่ 6.3

6.2 ก. ภาพต้นแบบ (reference)



6.2 ข. ด้านซ้ายเป็นภาพที่นำมาเปรียบเทียบ ด้านขวาเป็นภาพที่ได้จากการเปรียบเทียบ

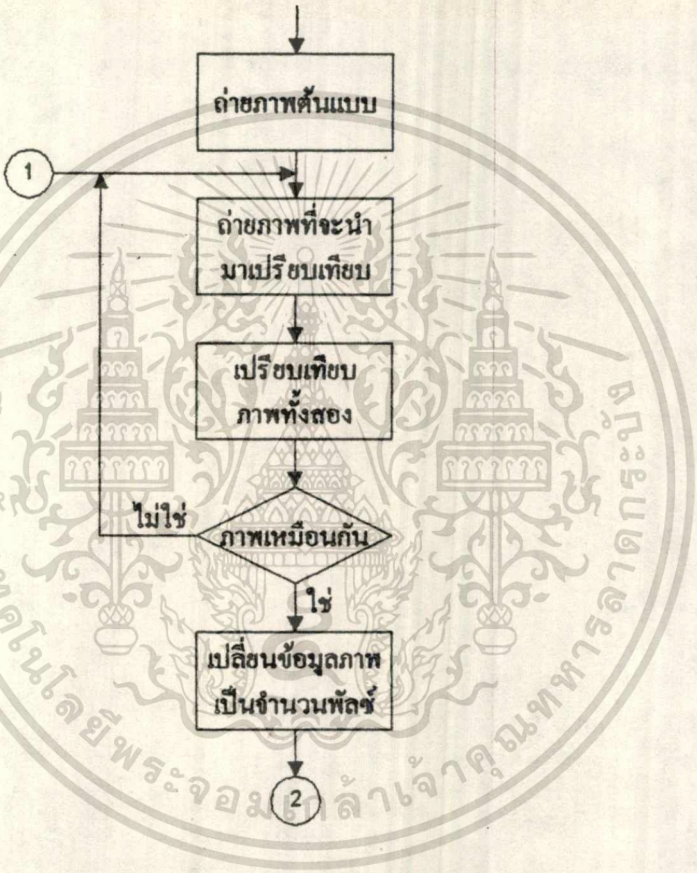
รูปที่ 6.2 แสดงถึงการเปรียบเทียบภาพ 2 ภาพที่แตกต่างกัน

6.3 ก ภาพต้นแบบ (reference)

6.3 ข. ด้านซ้ายเป็นภาพเปรียบเทียบ ด้านขวาภาพได้จากการเปรียบเทียบ

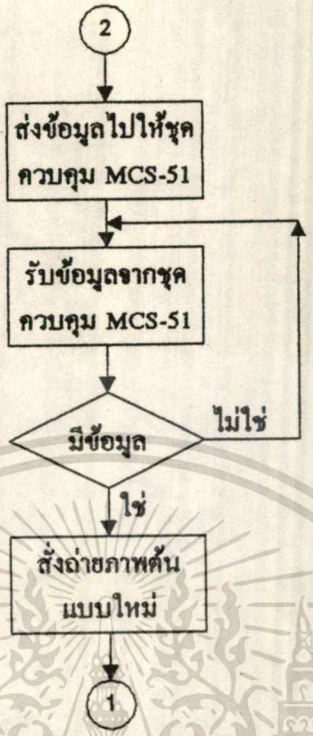
รูปที่ 6.3 แสดงถึงการเปรียบเทียบภาพ 2 ภาพที่เหมือนกัน

ในกรณีที่ทั้งสองมีความเหมือนกัน การเปรียบเทียบภาพจะเกิดขึ้นต่อไปเรื่อย ๆ โดยภาพจะถูกถ่ายต่อเนื่องเข้ามาเปรียบเทียบเรื่อย ๆ จนกว่าภาพใหม่ที่ถ่ายเข้ามาครั้งหลังสุดแตกต่างไปจากภาพต้นแบบ (reference) ความแตกต่างนี้จะถูกแปรเปลี่ยนไปเป็นข้อมูลจำนวนพัลซ์และทิศทางในการหมุนของสเตปปิ้งมอเตอร์ การเปรียบเทียบภาพแสดงในรูปโฟล์ชาร์ทที่ 6.4



รูปที่ 6.4 โฟล์ชาร์ทการเปรียบเทียบภาพ

จากนั้นส่งข้อมูลทั้งสองออกทางพอร์ท RS-232 ไปให้ชุดควบคุม MCS-51 ทำงานในขั้นตอนต่อไป หลังจากคอมพิวเตอร์ส่งข้อมูลออกไปแล้ว คอมพิวเตอร์จะรอรับข้อมูลที่จะส่งกลับมาจากชุดควบคุม MCS-51 ข้อมูลนี้จะเป็นการบอกให้คอมพิวเตอร์ทราบว่า การหมุนของกล้องได้หมุนไปยังตำแหน่งที่ต้องการเรียบร้อยแล้ว จากนั้นคอมพิวเตอร์จะสั่งให้ถ่ายภาพใหม่มาเป็นข้อมูลภาพต้นแบบ (reference) ใหม่ และถ่ายภาพต่อไปมาเปรียบเทียบกับภาพต้นแบบนี้เหมือนกับที่กล่าวมาในข้างต้น ทำเช่นนี้ต่อไปเรื่อย ๆ จนกว่าจะออกจากโปรแกรม การส่งและรับข้อมูลระหว่างคอมพิวเตอร์กับชุดควบคุม MCS-51 แสดงในรูปโฟล์ชาร์ทที่ 6.5



รูปที่ 6.5 โฟล์ชาร์ทการรับส่งข้อมูลระหว่างคอมพิวเตอร์กับชุดควบคุม MCS-51

การใช้งานโปรแกรม

การทำงานของโปรแกรมจะแบ่งออกเป็นเมนูให้เลือกการทำงาน มีทั้งหมด 6 ตัวเลือก ดังนี้

1. Digitize and Save to file
2. Display image from file
3. Threshold
4. Tutorial
5. Automatic Following
6. Exit

Press select chiose:

การใช้งานแต่ตัวเลือก

1. ตัวเลือกที่ 1 ใช้ในการถ่ายภาพมาแสดงที่หน้าจอคอมพิวเตอร์ สามารถปรับความสว่างของภาพได้ ถ้าเป็นการปรับ Contrast ให้กดแป้นคีย์ + และ - ถ้าเป็นการปรับ Brighness ให้กดแป้นคีย์ Q และ W ถ้าต้องการบันทึกภาพเก็บไว้ในไฟล์ ให้กดแป้นคีย์ Enter ภาพจะ

หยุดนิ่งเพื่อจะให้เราดูรายละเอียดของภาพ จากนั้นกดแป้นคีย์ใด ๆ เครื่องจะถามว่า ต้องการบันทึกภาพนี้หรือไม่ ถ้าต้องการให้กดคีย์ Y ถ้าไม่ต้องการให้กดคีย์ N นอกจากนี้ในตัวเลือกนี้ยังสามารถควบคุมให้กล้องหมุนได้โดยการกดแป้นคีย์ลูกศรซ้ายและขวา

2. เป็นการนำภาพที่เก็บไว้ในไฟล์มาแสดงที่หน้าจอคอมพิวเตอร์ โดยที่ส่วนขยายไฟล์เป็น .img ซึ่งเป็นเพียงการสมมุติขึ้นมา จะสามารถใส่ส่วนขยายเป็นอย่างไรก็ได้ แต่ตอนเรียกขึ้นมาแสดงภาพต้องเขียนให้เหมือนเดิม ถ้าไม่รู้หรือจำชื่อไฟล์ไม่ได้ ให้พิมพ์ *.* หรือ *.img เครื่องจะแสดงชื่อไฟล์ออกมา

3. เป็นการเทรสโฮล ทำให้ภาพแสดงออกหน้าจอเป็น 2 ระดับ โดยทำกับภาพที่แสดงอยู่ในขณะนั้น ดังแสดงในรูปที่ 6.6

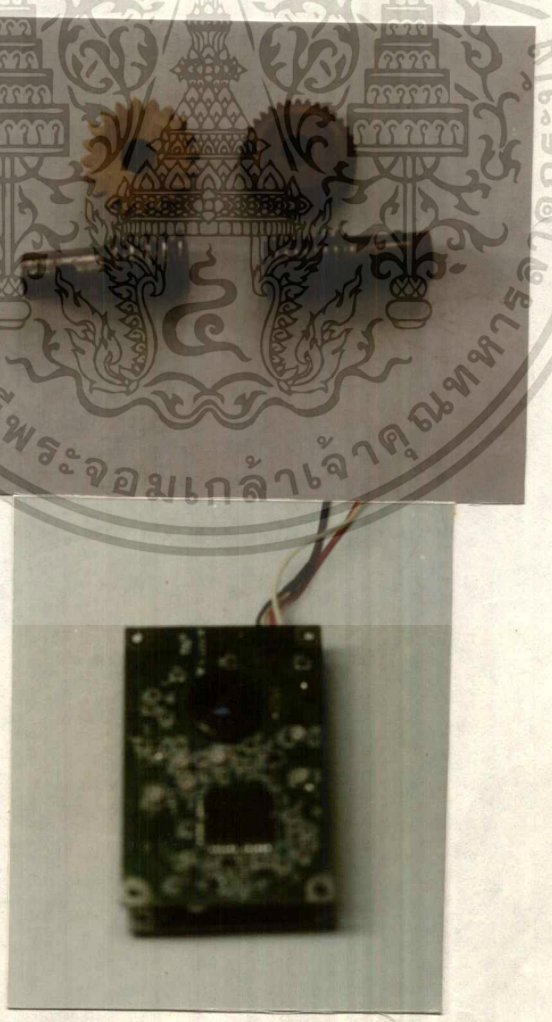
รูปที่ 6.6 ด้านบนเป็นภาพปกติ ด้านล่างเป็นภาพที่ได้จากการเทรสโฮล

4. เป็นการ Tutorial วิธีการทำงานของส่วนโปรแกรมการติดตามภาพอัตโนมัติ ในตอนแรกเป็นการปรับความสว่างเหมือนในข้อ 1 เมื่อปรับได้ตามต้องการแล้ว ให้กดคีย์ Esc โปรแกรมจะถ่ายภาพแรกเข้าไปเก็บบัฟเฟอร์ จากนั้นให้กดคีย์ Enter เพื่อเป็นการถ่ายภาพที่ 2 ต่อจากนั้นโปรแกรมก็จะทำการเปรียบเทียบภาพทั้งสอง ภาพใหม่ที่ได้จากการเปรียบเทียบจะถูกแสดงออกมาที่หน้าจอ เมื่อกดคีย์ใด ๆ อีกครั้ง กล้องจะหมุนไปที่ตำแหน่งของวัตถุนั้น

5. ให้กล้องเคลื่อนที่ตามภาพอัตโนมัติ

6. ออกจากโปรแกรม

จากตัวเลือกทั้งหมดมี 6 ตัวเลือก ตัวเลือกที่ 5 จะเป็นการเลือกให้กล้องหมุนตามวัตถุแบบอัตโนมัติ ซึ่งจะรวมเอาวิธีการทำงานและเทคนิคการเขียนโปรแกรมของตัวเลือกทุกหัวข้อมารวมไว้ในตัวเลือกนี้ ดังนั้นจึงขอกล่าวเฉพาะตัวเลือกที่ 5



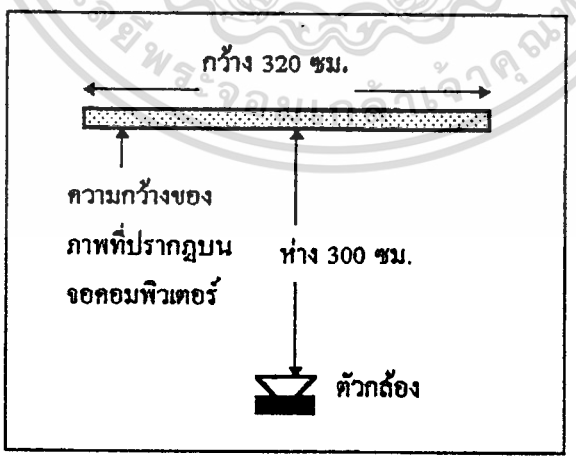
รูปที่ 6.7 เฟืองทดและกล้องโทรทัศน์

6.2.2 วิธีการและขั้นตอนการทำงานของโปรแกรมการติดตามวัตถุอัตโนมัติ

1. ในขั้นตอนแรก เครื่องจะให้เราป้อนระยะห่างระหว่างตัวกล้องกับวัตถุที่จะให้เคลื่อนที่ตาม โดยจะต้องป้อนเป็นหน่วยเซ็นติเมตร จุดประสงค์ของขั้นตอนนี้ก็เพื่อต้องการรู้จำนวนพัลซ์ที่จะป้อนให้กับสเตปปิ้งมอเตอร์ต่อระยะทางที่เปลี่ยนตำแหน่งไป และระยะทางต่อหนึ่งจุดพิกเซล คุณลักษณะของตัวสเตปปิ้งมอเตอร์และขนาดอัตราทดของเฟืองที่ใช้ขับกล้องจะเกี่ยวข้องอยู่ในขั้นตอนนี้ด้วย ซึ่งจะนำไปใช้คำนวณหาสิ่งที่ต้องการโดยโปรแกรม รูปของเฟืองและกล้องในรูป 6.7

อัตราทดเฟือง 20:1 รอบ
คุณลักษณะของสเตปปิ้ง 1 สเตป : 1.8 องศา
จำนวนพัลซ์ที่ป้อนต่อการหมุน 1 รอบ 360/1.8 = 200 พัลซ์
องศาของการหมุนสเตปปิ้งต่อการหมุนของกล้อง 20:1 องศา
จำนวนพัลซ์ที่ป้อนต่อการหมุนที่เปลี่ยนไป 1 องศาของกล้อง 20/1.8 = 11.11 พัลซ์

จากการทดลองวัดระยะทางที่ความห่างจากกล้องถึงวัตถุ 300 เซ็นติเมตร จะได้ความกว้างของภาพที่หน้าจอแสดงภาพ 320 เซ็นติเมตร เมื่อขนาดของพิกเซลที่ใช้ 256 พิกเซล ดังแสดงในรูปที่ 6.8 ส่วนในรูปที่ 6.9 เป็นการแสดงถึงจุดพิกเซลที่แทนด้วยภาพจริงที่มีความกว้าง 1.25 เซ็นติเมตร

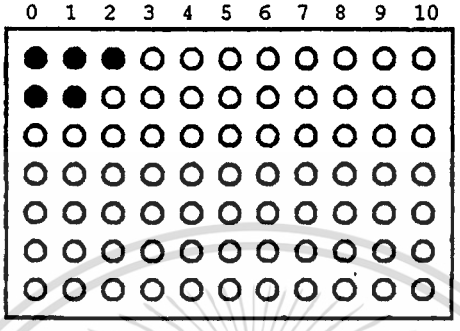


รูปที่ 6.8 มุมมองของกล้อง

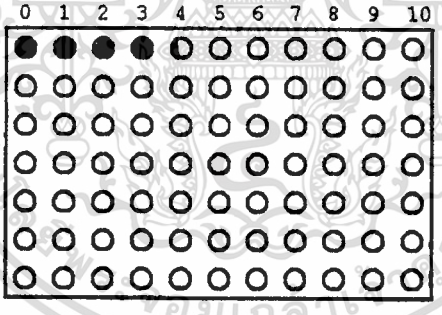
ระยะทางของภาพจริงต่อ 1 พิกเซล 320/256 = 1.25 เซ็นติเมตร

ระยะทางของภาพจริงต่อมุม 1 องศาที่เปลี่ยนไปของกล้อง $2*\pi*300/360 = 5.236$ ซ.ม

มุม 1 องศาที่เปลี่ยนไปของกล้องต่อจำนวนพิกเซล 5.236/1.25 = 4.19 พิกเซล ดังแสดงในรูปที่ 6.10



รูปที่ 6.9 พิกเซล 1 จุดแทนขนาดภาพจริง 1.25 เซ็นติเมตร



รูปที่ 6.10 พิกเซล 4.19 จุดแทนด้วยการหมุนของกล้องไป 1 องศา

จำนวนสเตปที่สเตปปิ้งหมุนต่อ 1 พิกเซล 11.11/4.19 = 2.65 สเตป

ข้อมูลที่จะเอาไปใช้งานในขั้นตอนต่อไปก็คือ 2.65 สเตป : 1 พิกเซล

2. เป็นการปรับความสว่างของภาพ เพื่อให้ได้ความคมชัดและความสว่างที่เหมาะสม
3. ถ่ายภาพมาเก็บไว้ใน imagetwo[x][y] เป็นภาพต้นแบบ (reference)
4. ถ่ายภาพมาเก็บไว้ใน imageone[x][y] เป็นภาพที่จะนำมาเปรียบเทียบ

5. ขั้นตอนนี้เป็นขั้นตอนที่สำคัญที่สุด ซึ่งเป็นการเปรียบเทียบภาพ เพื่อหาความแตกต่าง โดยใช้สมการ

newpic[x][y] = MAX - abs(imagetwo[x][y] - imageone[x][y])

MAX เป็นค่าระดับเทาสูงสุดของพิกเซลในภาพต้นแบบ

จากสมการจะเป็นการนำข้อมูลภาพ 2 ภาพแต่ละพิกเซลมาลบกัน ภายในเครื่องหมายสัมบูรณ์ ค่าที่ได้จึงออกมาเป็นค่าบวก แล้วนำค่าที่ได้มาลบออกจากค่าระดับเทาสูงสุดของพิกเซลในภาพต้นแบบ ดังตัวอย่างที่แสดงในรูปของเมตริกซ์

ตัวอย่างที่ 1

$$\begin{bmatrix} 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 \end{bmatrix} \qquad \begin{bmatrix} 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 30 & 64 & 64 \\ 64 & 20 & 15 & 20 & 64 \\ 64 & 15 & 20 & 15 & 64 \end{bmatrix}$$

ข้อมูลภาพต้นแบบ (ภาพที่ 1) ข้อมูลภาพวัตถุ (ภาพที่ 2)

เมื่อนำมาคำนวณตามสมการ ข้อมูลใหม่ที่ได้เมื่อค่าสูงสุดของภาพฉากคือ 64

$$\begin{bmatrix} 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 30 & 64 & 64 \\ 64 & 20 & 15 & 20 & 64 \\ 64 & 15 & 20 & 15 & 64 \end{bmatrix}$$

ข้อมูลของภาพใหม่

ตัวอย่างที่ 2

$$\begin{bmatrix} 50 & 45 & 55 & 45 & 50 \\ 20 & 40 & 64 & 64 & 50 \\ 30 & 60 & 20 & 35 & 60 \\ 10 & 50 & 25 & 40 & 60 \\ 64 & 50 & 30 & 50 & 60 \end{bmatrix}$$

ข้อมูลภาพต้นแบบ (ภาพที่ 1)

$$\begin{bmatrix} 50 & 45 & 10 & 15 & 10 \\ 20 & 40 & 25 & 15 & 20 \\ 30 & 60 & 20 & 30 & 30 \\ 10 & 50 & 25 & 40 & 60 \\ 64 & 50 & 30 & 50 & 60 \end{bmatrix}$$

ข้อมูลภาพวัตถุ (ภาพที่ 2)

$$\begin{bmatrix} 64 & 64 & 19 & 34 & 24 \\ 64 & 64 & 25 & 15 & 34 \\ 64 & 64 & 64 & 59 & 34 \\ 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 \end{bmatrix}$$

ข้อมูลของภาพใหม่

จะเห็นว่าตัวอย่างที่ 1 ข้อมูลของภาพใหม่ที่ได้จะเป็นข้อมูลเดียวกับข้อมูลภาพวัตถุ แสดงว่ามีวัตถุเข้ามาในบริเวณนั้น ตัวอย่างภาพจริงแสดงในรูปที่ 6.2

จากตัวอย่างที่ 2 เมื่อสังเกตจะเห็นว่า ถ้าข้อมูลภาพฉากกับข้อมูลภาพวัตถุมีค่าใกล้เคียงกันมาก เมื่อนำไปคำนวนตามสมการจะได้ข้อมูลที่มีค่าใกล้เคียงกับค่าสูงสุดของภาพต้นแบบ นั้นก็แสดงให้เห็นว่า ยิ่งระดับสีเทาของภาพวัตถุและภาพต้นแบบมีค่าใกล้เคียงกันมาก ยิ่งทำให้การตรวจจับวัตถุเกิดความผิดพลาดได้มาก ยิ่งมีระดับสีเทาที่เหมือนกันจะทำให้ตรวจจับวัตถุไม่ได้เลย ซึ่งเป็นข้อเสียของวิธีการเปรียบเทียบแบบนี้

ตัวอย่างที่ 3

$$\begin{bmatrix} 60 & 61 & 59 & 59 & 60 \\ 55 & 53 & 61 & 60 & 59 \\ 55 & 55 & 53 & 60 & 61 \\ 31 & 30 & 35 & 45 & 40 \\ 15 & 25 & 20 & 25 & 30 \end{bmatrix}$$

ข้อมูลภาพต้นแบบ (ภาพที่ 1)

$$\begin{bmatrix} 60 & 61 & 59 & 59 & 60 \\ 55 & 53 & 61 & 60 & 59 \\ 55 & 55 & 53 & 60 & 61 \\ 31 & 30 & 35 & 45 & 40 \\ 15 & 25 & 20 & 25 & 30 \end{bmatrix}$$

ข้อมูลภาพวัตถุ (ภาพที่ 2)

$$\begin{bmatrix} 61 & 61 & 61 & 61 & 61 \\ 61 & 61 & 61 & 61 & 61 \\ 61 & 61 & 61 & 61 & 61 \\ 61 & 61 & 61 & 61 & 61 \\ 61 & 61 & 61 & 61 & 61 \end{bmatrix}$$

ข้อมูลภาพใหม่

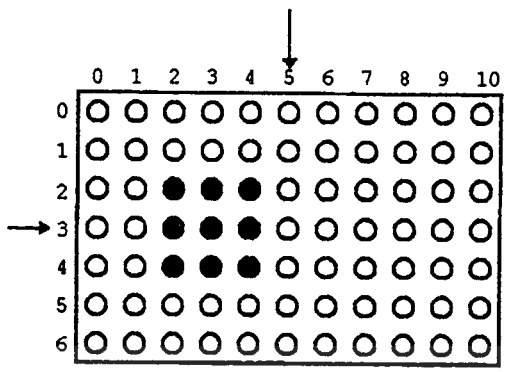
ตัวอย่างที่ 3 เป็นข้อมูลของภาพที่เหมือนกัน ดังนั้นข้อมูลของภาพใหม่ที่ได้แต่ละจุดพิกเซลจะเท่ากับค่าระดับเทาสูงสุดของพิกเซลในภาพต้นแบบ แสดงว่าไม่มีวัตถุผ่านเข้ามาในบริเวณนี้

6. นำภาพใหม่ที่ได้จากขั้นตอนที่ 5 มาแปลงเป็นภาพ 2 ระดับ โดยทำการเทรสโฮล

7. ในขั้นตอนนี้ก็เป็นอีกขั้นตอนหนึ่งที่มีความสำคัญ ซึ่งเป็นขั้นตอนที่บอกว่าภาพที่นำมาเปรียบเทียบกันนั้นแตกต่างกันหรือเหมือนกัน คือ เป็นการหาตำแหน่งจุดกึ่งกลางภาพวัตถุ สามารถอธิบายได้ตามโฟล์ชาร์ท สมมุติว่าเป็นการหาจุดกึ่งกลางของภาพในรูปที่ 6.11 จะได้จุดกึ่งกลางที่บอกเป็นคู่ลำดับ (cordinate) จะได้ Xc = 3 และ Yc = 3

ถ้า Xc และ Yc มีค่าเท่ากับ 64 แสดงว่าภาพทั้ง 2 ภาพที่นำมาเปรียบเทียบกันเหมือนกัน โปรแกรมจะวนลูปกลับไปเริ่มทำงาน ในขั้นตอนที่ 4 แต่ถ้า Xc หรือ Yc มีค่าไม่ใช่ 64 หรือทั้งสอง แสดงว่าภาพทั้ง 2 ที่นำมาเปรียบเทียบกันแตกต่างกัน โปรแกรมจะทำงานขั้นตอนต่อไป

ตัวอย่างการหาจุดกึ่งกลางของวัตถุและการหาความห่างระหว่างจุดกึ่งกลางของภาพกับวัตถุดังนี้ จากรูปสมมุติว่ากรอบสี่เหลี่ยมเป็นกรอบของรูปภาพ วงกลมเล็ก ๆ ข้างในแทนด้วยจุดพิกเซล จุดพิกเซลที่เป็นสีดำแทนด้วยภาพวัตถุที่ตรวจพบได้ด้วยสมการ



(x,y)

(2,2) (3,2) (4,2)

(2,3) (3,3) (4,3)

(2,4) (3.4) (4,4)

ตำแหน่งของวัตถุ (X,Y) ที่มีจุดพิกเซลเป็นสีดำ

ผลรวมของ X = (2x3) + (3x3) + (4x4) = 27

ผลรวมของ Y = (2x3) + (3x3) + (4x4) = 27

พื้นที่ area = ผลรวมของพิกเซล = 9

จะได้จุดกึ่งกลางของแกน X คือ Xc = 27/9 = 3

จะได้จุดกึ่งกลางของแกน Y คือ Yc = 27/9 = 3

เพราะฉะนั้นจะได้จุดกึ่งกลางของวัตถุ (X,Y) = (3,3)

หาความห่างระหว่างจุดกึ่งกลางภาพกับวัตถุ

จากรูปจุดกึ่งกลางภาพอยู่ที่ (X,Y) = (5,3)

ความห่างทางแกน Xc = 5 - 3 = 2

ความห่างทางแกน Yc = 3 - 3 = 0

8. จากขั้นตอนที่ 8 นี้เป็นต้นไป โปรแกรมจะทำงานก็ต่อเมื่อมีวัตถุเข้ามาในบริเวณที่กำหนดไว้ ขั้นตอนนี้เป็นการหาตำแหน่งของวัตถุว่าอยู่ทางด้านซ้ายหรือด้านขวาของกรอบภาพในจอแสดงผล โดยจะเปรียบเทียบ Xc กับค่า 64 ถ้า Xc มากกว่า 64 แสดงว่าวัตถุอยู่ทางด้านขวา ถ้า Xc น้อยกว่า 64 แสดงว่าอยู่ทางด้านซ้าย เมื่อทราบว่าวัตถุอยู่ด้านซ้ายหรือขวาแล้ว จะมาเช็คต่ออีกว่าอยู่ห่างจุดกึ่งกลางจอแสดงภาพกี่พิกเซล ดังตัวอย่างในข้อ 7

9. เมื่อทราบว่าห่างเป็นจำนวนกี่พิกเซลแล้ว ก็เอาจำนวนสเตปต่อพิกเซลที่คำนวณได้ในขั้นตอนแรกมาคำนวณหาจำนวนพัลซ์ที่ต้องป้อนให้กับสเตปปิ้งมอเตอร์ จากขั้นตอนแรกได้สเตปต่อพิกเซล 2.65 สเตป : 1 พิกเซล สมมุติว่าได้ระยะห่างจากขั้นตอนที่ 8 เป็น 50 พิกเซล สามารถคำนวณหาจำนวนพัลซ์ได้

2.65*50 = 132.5 หรือ 132 พัลซ์

จากนั้นก็ส่งจำนวนพัลซ์และทิศทางที่จะให้สเตปปิ้งหมุน ออกทางพอร์ท RS-232 ซึ่งพอร์ทนี้จะต่ออยู่กับชุดไมโครคอนโทรลเลอร์ MCS-51 โดยที่ชุดควบคุม MCS-51 จะทำหน้าที่เป็นตัวควบคุมการหมุนของสเตปปิ้ง ตามจำนวนพัลซ์และทิศทางที่ส่งมาจากกคอมพิวเตอร์

10. เมื่อสเตปปิ้งได้หมุนครบตามจำนวนพัลซ์แล้ว ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณกลับมาให้คอมพิวเตอร์ได้ทราบว่าได้หมุนไปยังตำแหน่งที่ต้องการแล้ว เมื่อคอมพิวเตอร์ได้รับสัญญาณ โปรแกรมก็จะสั่งให้ทำการถ่ายภาพใหม่เข้าเก็บใน imagetwo[x][y] เพื่อเป็นภาพต้นแบบ (reference) อันใหม่และวนลูปกลับไปเริ่มทำงานในขั้นตอนที่ 4 ทำเช่นนี้ไปเรื่อย ๆ จนกว่าจะออกจากโปรแกรม

ทั้ง 10 ข้อนี้เป็นขั้นตอนการทำงานของโปรแกรมการติดตามการเคลื่อนที่ของวัตถุโดยใช้ข้อมูลภาพ รายละเอียดรวม ๆ ได้กล่าวในหัวข้อของโปรแกรมไปแล้ว ซึ่งจะใช้อ่านประกอบกันและรายละเอียดขั้นตอนการทำงานยังสามารถดูได้จากโฟล์ชาร์ท

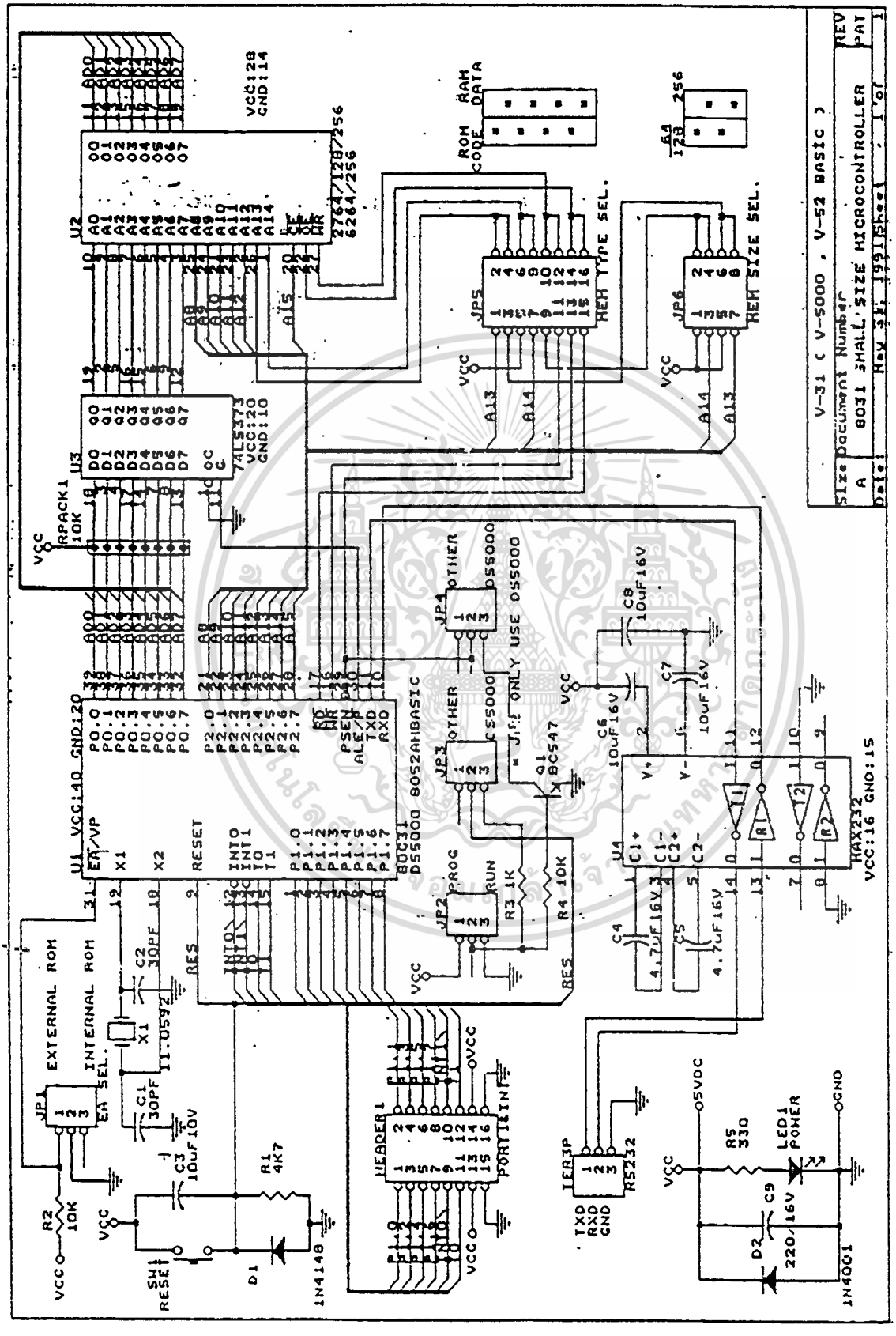
โปรแกรมบนไมโครคอนโทรลเลอร์ MCS-51

โปรแกรมในส่วนนี้ทำหน้าที่ควบคุมให้ชุดไมโครคอนโทรลเลอร์ MCS-51 ควบคุมการหมุนของสเตปปิ้งมอเตอร์ โดยชุด MCS-51 จะรับข้อมูลจากคอมพิวเตอร์ผ่านทางพอร์ทอนุกรม RS-232 ข้อมูลที่รับมานี้จะเป็นข้อมูลที่บอกว่า ให้สเตปปิ้งมอเตอร์หมุนไปทางด้านซ้ายหรือขวา และหมุนไปเป็นจำนวนสเตปเท่าไหร่ นำข้อมูลที่รับมาได้กำเนิดเป็นพัลซ์ในทิศทางตามที่กำหนดป้อนให้กับชุดขับสเตปปิ้งมอเตอร์ ให้หมุนไปยังตำแหน่งที่ต้องการ ดังแสดงในโฟล์ชาร์ท

6.2.3 ส่วนประกอบของตัวเครื่อง (Hardware)

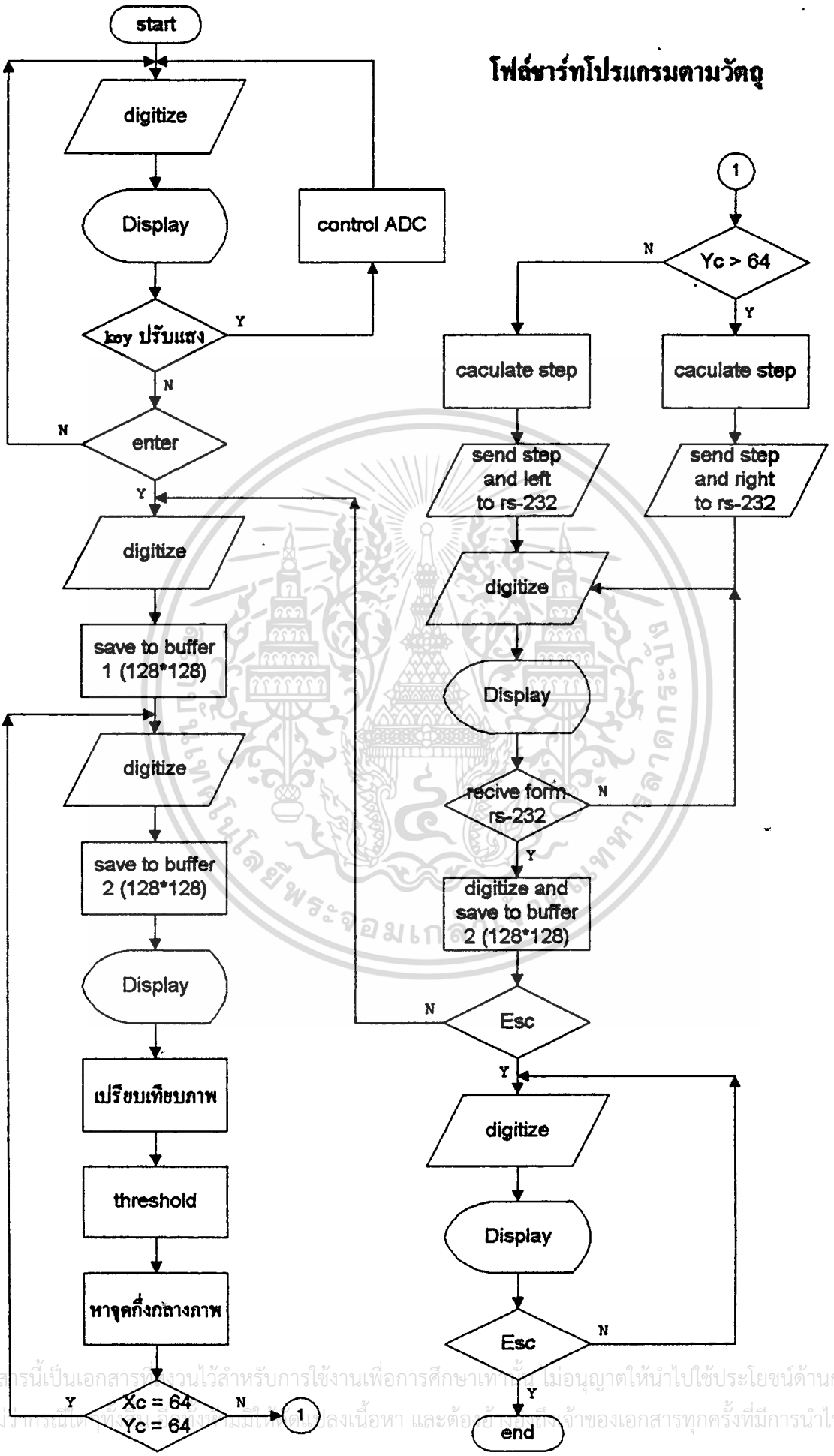
ชุดควบคุมไมโครคอนโทรลเลอร์ เป็นชุดควบคุมแบบสำเร็จรูปที่นำมาเขียนโปรแกรมประยุกต์ใช้งานได้ทันที อยู่ในตระกูล MCS-51 เบอร์ 8031 รุ่น V-31 ของบริษัทศิลารีเสิร์ช วงจรแสดงในรูป 6.11 เอาท์พุทที่ต่อเข้ากกับชุดขับสเตปปิ้งมอเตอร์ ต่อออกทางพอร์ท 1 (P1)

ชุดขับสเตปปิ้งมอเตอร์ ทำหน้าที่ขับกระแสให้สูงขึ้น เพื่อให้เพียงพอกับความต้องการของสเตปปิ้งมอเตอร์ โดยใช้ทรานซิสเตอร์ต่อแบบดาริงตันกัน 4 ชุด สำหรับขับสเตปปิ้ง 4 เฟส ดังแสดงในรูปวงจรที่ 6.12

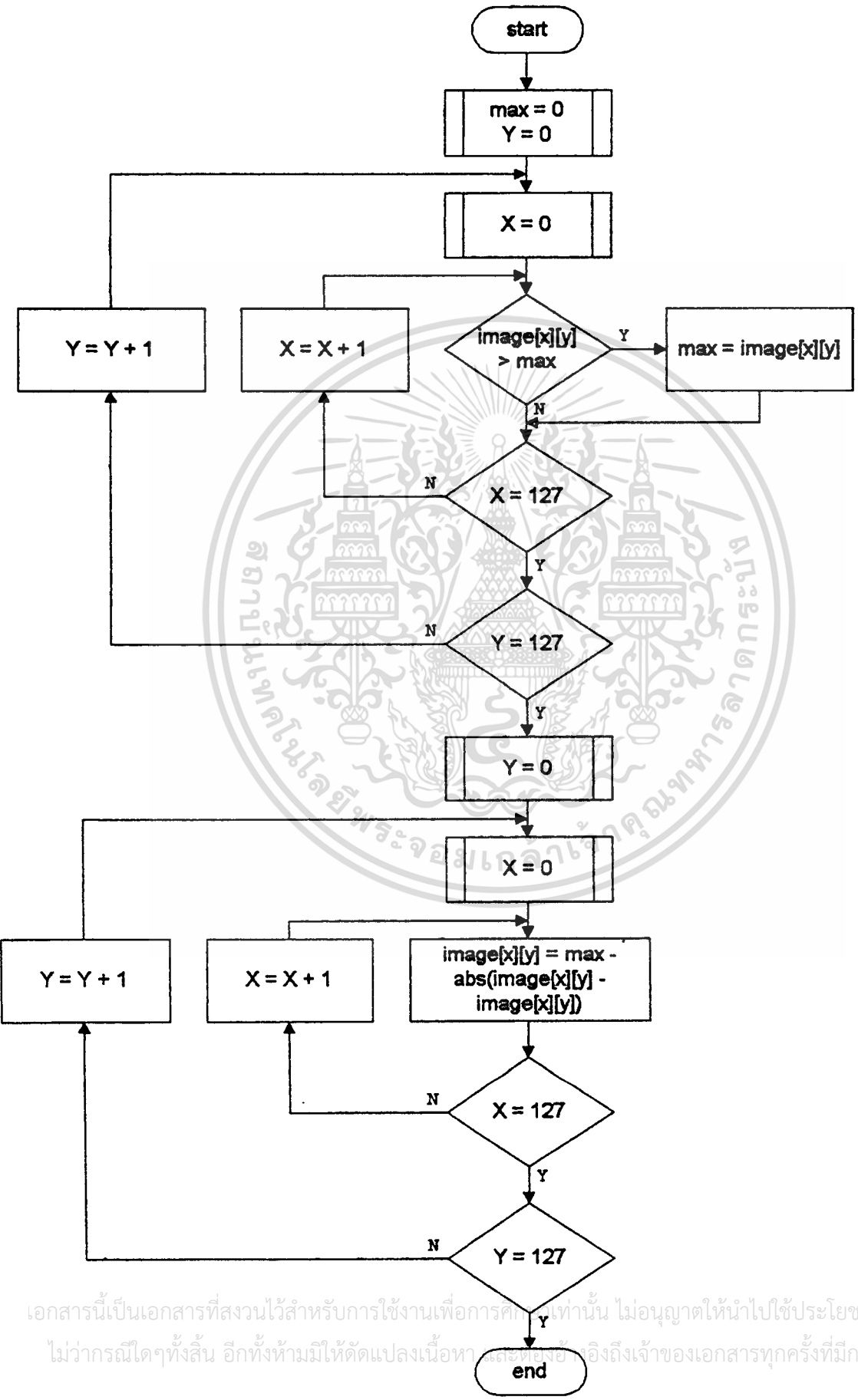


วงจรควบคุมใช้ในโทรศัพท์ MCS-51

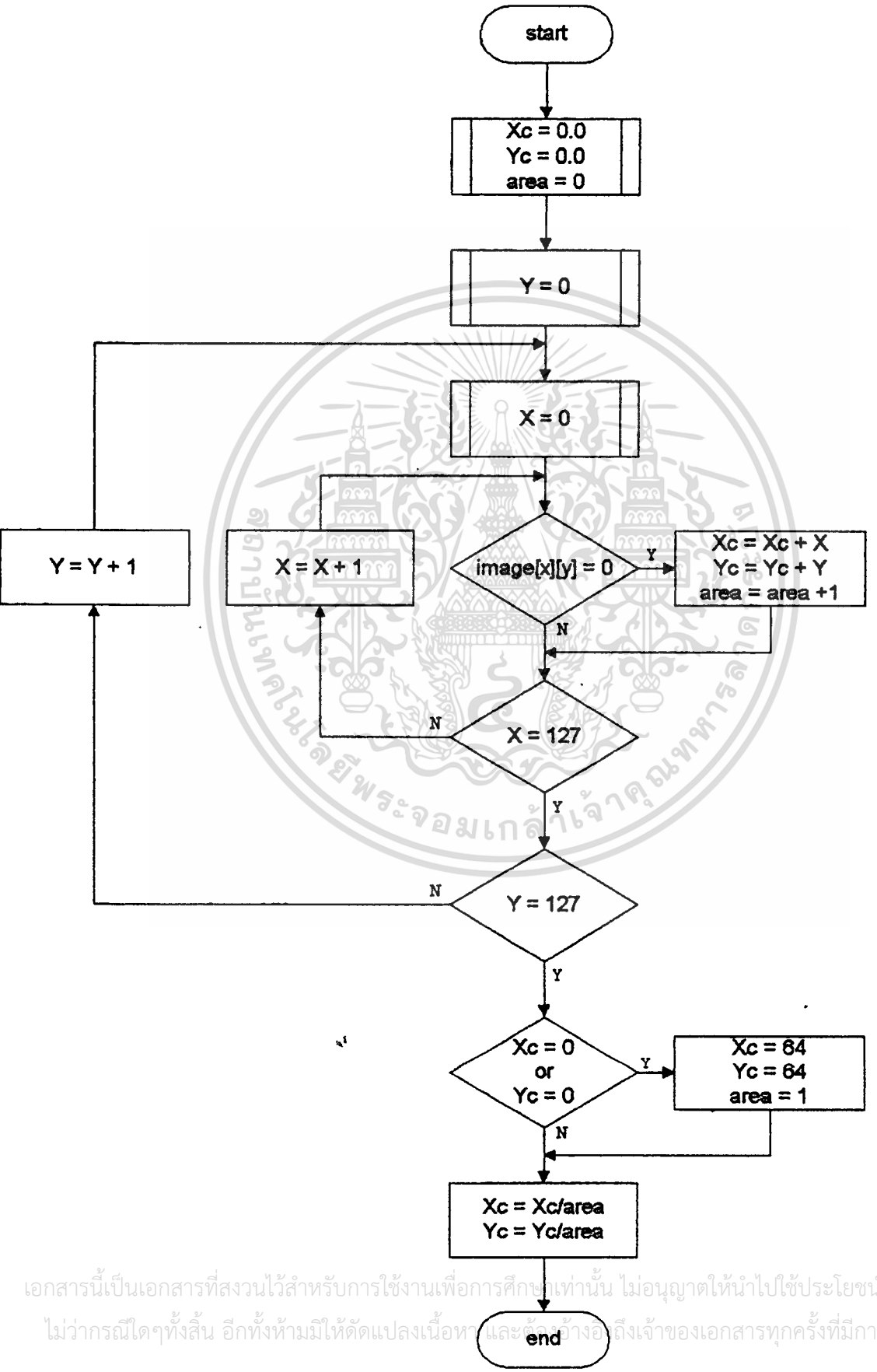
โฟล์ชาร์ทโปรแกรมตามวัตถุ



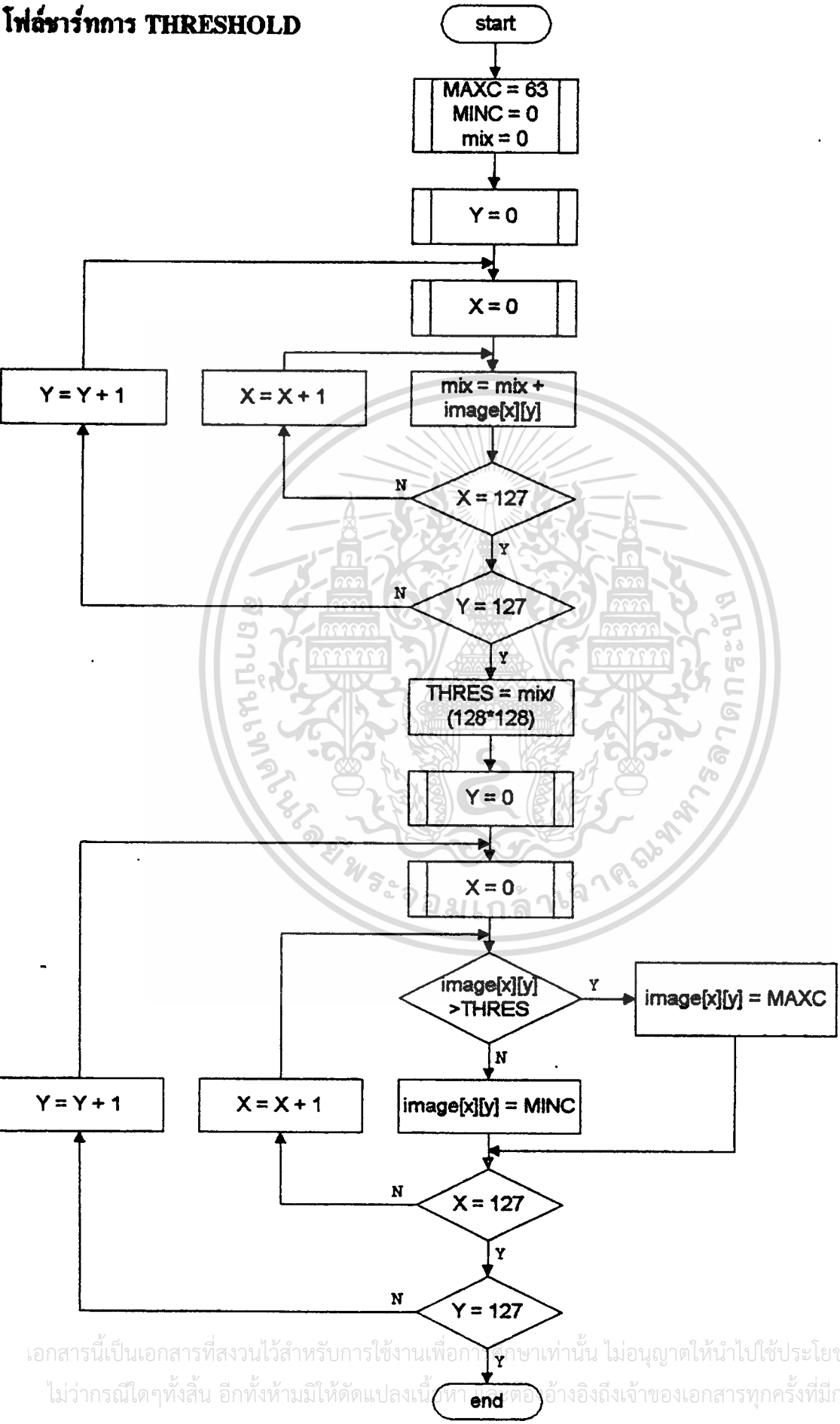
โฟล์ชาร์ทการเปรียบเทียบภาพ



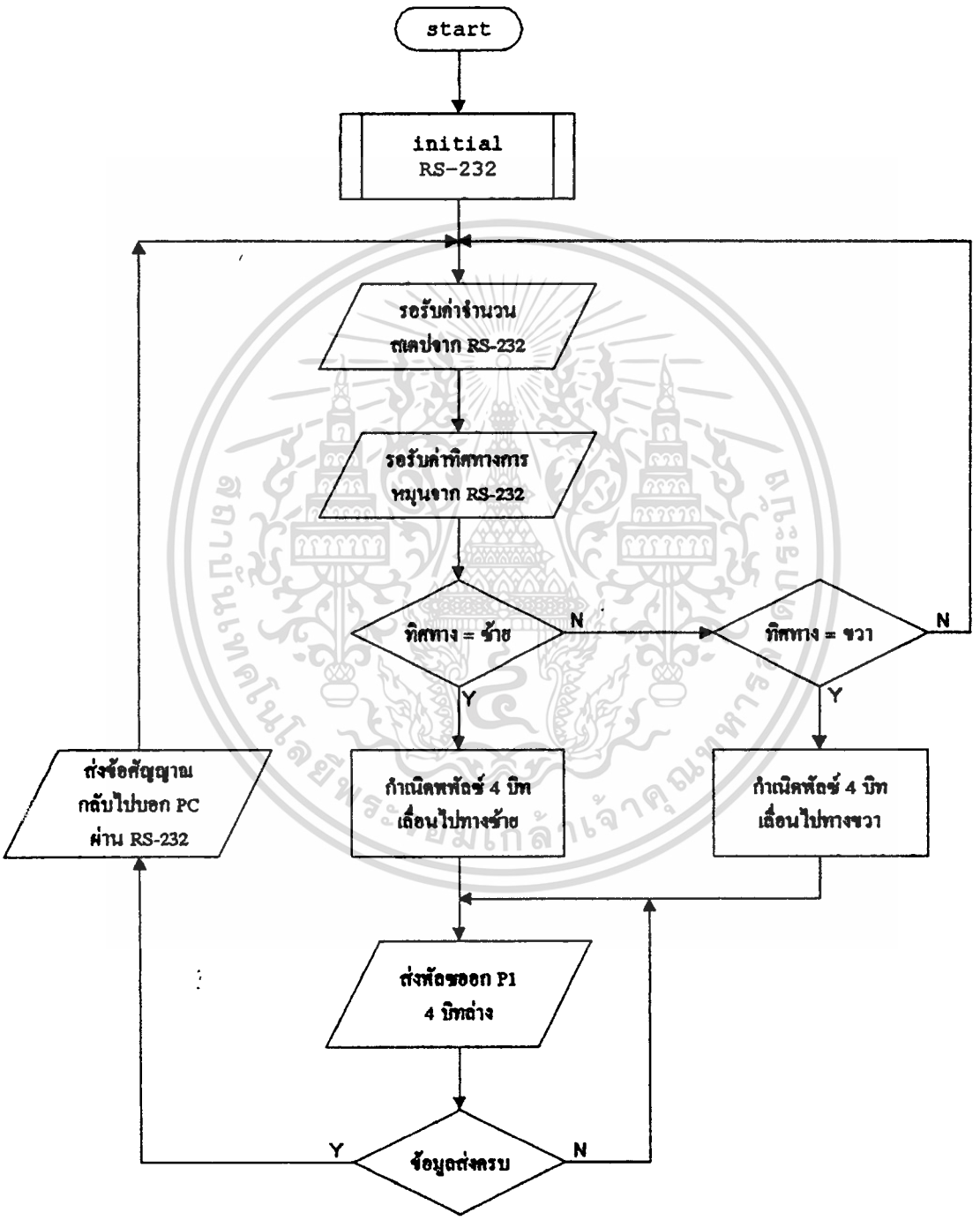
โฟล์ชาร์ทการหาจุดกึ่งกลางวัตถุ



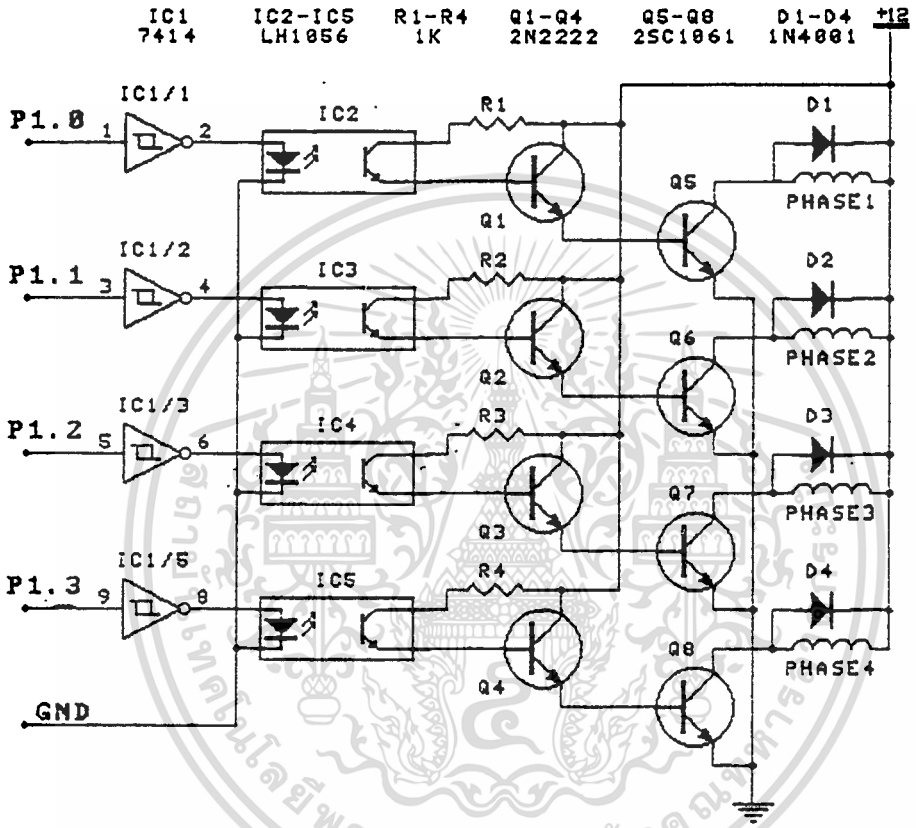
โฟล์ชาร์ทการ THRESHOLD



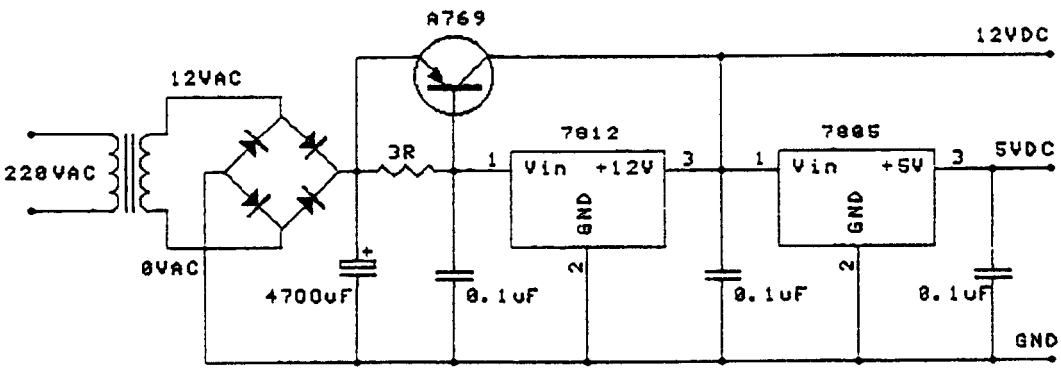
โฟล์ชาร์ทการทำงานของไมโครคอนโทรลเลอร์



ชุดแหล่งจ่ายไฟ แหล่งจ่ายไฟจะจ่ายออกมาสองค่า คือ 12 โวลท์สำหรับจ่ายให้กับชุดขับสเตปปิ้งมอเตอร์และสเตปปิ้งมอเตอร์ และ 5 โวลท์จ่ายให้กับชุดไมโครคอนโทรลเลอร์ MCS-51 วงจรแสดงในรูปที่ 6.13



รูปที่ 6.12 วงจรขับสเตปปิ้งมอเตอร์

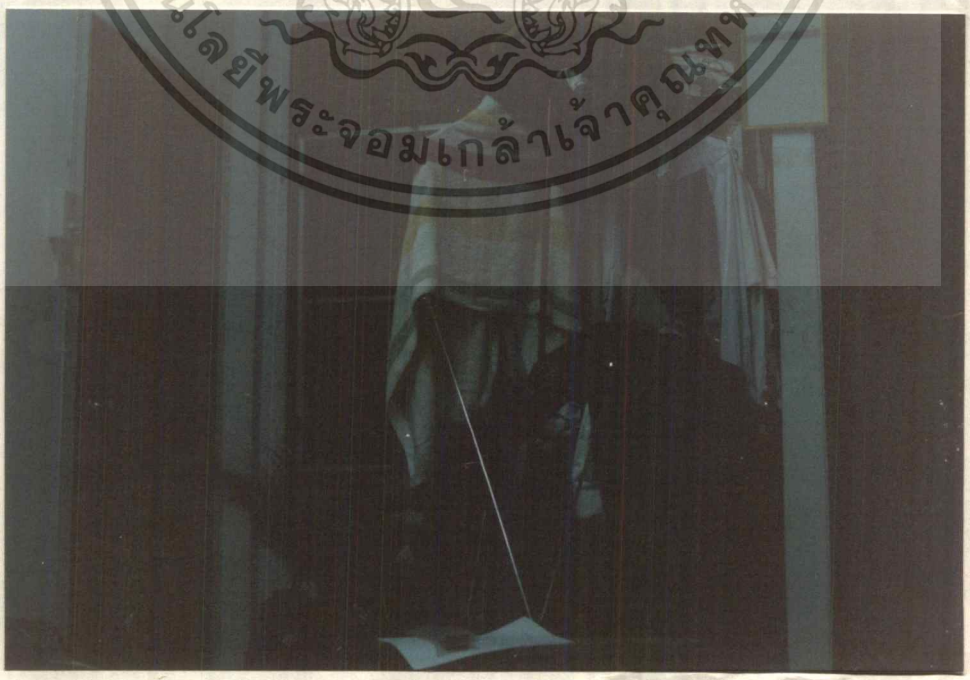


รูปที่ 6.13 วงจรแหล่งจ่ายไฟ

6.3 การทดลอง

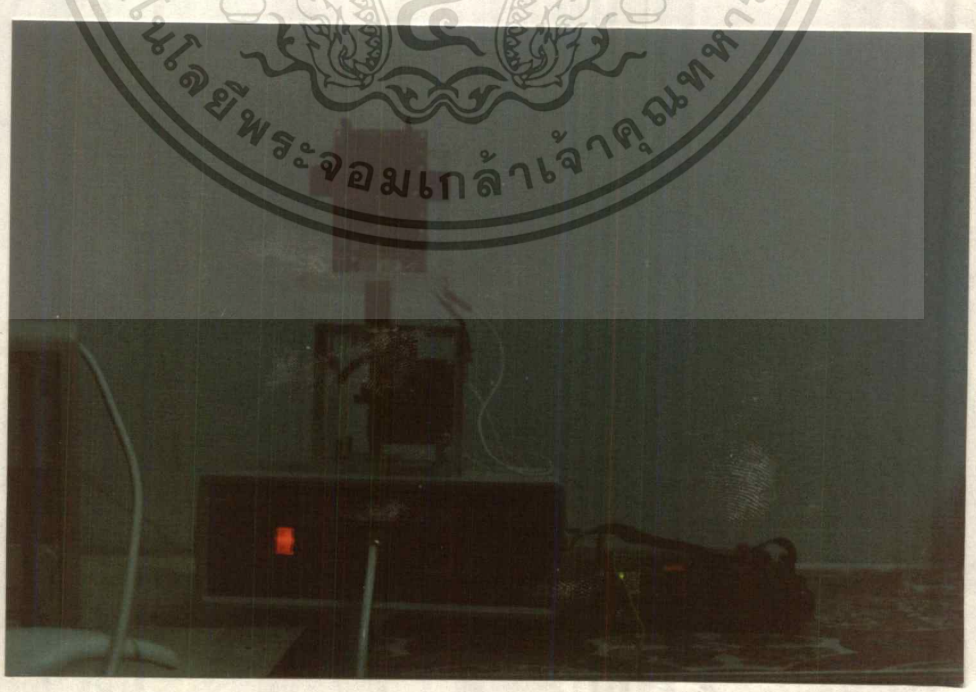
การทดลองการตามภาพวัตถุอัตโนมัติจะทำในห้องที่มีสภาพเต็มไปด้วยสิ่งของหลายสิ่งหลายอย่าง ดังแสดงในรูป 6.14 และรูปที่ 6.15 เป็นรูปของเครื่องต้นแบบที่ทำขึ้น ในตอนแรกการทดลองได้ทำตอนกลางคืน แสงที่ตกกระทบวัตถุมีเพียงแสงสว่างของหลอดฟูออเรสเซนต์ขนาด 40 วัตต์ ภายในห้องขนาด 3.5 x 5 เมตร ถ้าภายในห้องนั้นมีการเคลื่อนที่ของวัตถุเพียงสิ่งเดียวและเคลื่อนที่ไม่เร็วจนเกินไป ผลที่ได้ของการหมุนตามภาพวัตถุของกล้องโทรทัศน์ ได้ผลเกือบจะร้อยเปอร์เซ็นหรืออาจจะเต็มร้อยเปอร์เซ็นก็ว่าได้ แต่สีของวัตถุต้องเป็น 2 สีที่แตกต่างกันมาก เช่น วัตถุนั้นเป็นบุคคล ๆ นั้นต้องใส่เสื้อลายขวาดำ

ถ้าการเคลื่อนที่ของวัตถุมีมากว่าหนึ่งสิ่ง จะทำให้การหมุนตามภาพวัตถุของกล้องจะเกิดความสับสนหมุนไม่เป็นไปตามต้องการ และอีกอย่างคือความเข้มของสีภาพจะมีผลอย่างมากกับการหมุนตามภาพวัตถุของกล้อง หมายความว่า ถ้าวัตถุที่เคลื่อนที่มีสีใกล้เคียงกับภาพต้นแบบ (reference) มาก ๆ หรือเหมือนกับต้นแบบ (reference) เลย การเคลื่อนที่ตามภาพของกล้องจะได้ผลน้อยมากหรืออาจจะไม่ได้เลยในบางครั้ง ดังนั้นในขณะทำการทดลองเดินผ่านกล้องจะใส่เสื้อ 2 สีที่แตกต่างกันมาก ซึ่งจะแก้ปัญหาตรงจุดนี้ได้

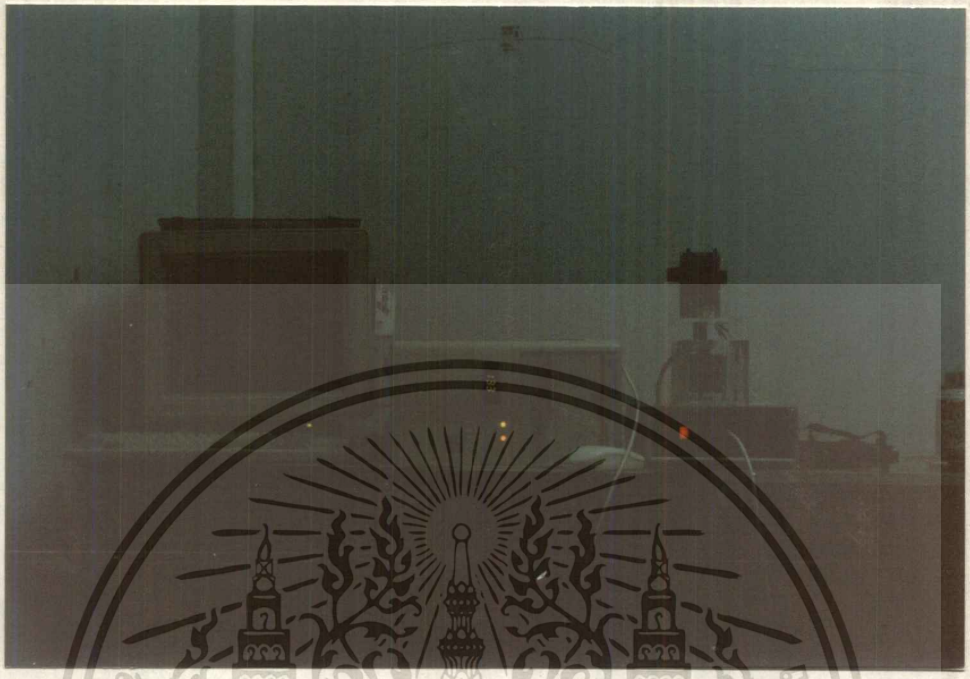


รูปที่ 6.14 สภาพของห้องที่ใช้ในการทดลอง

รูปที่ 6.14 (ต่อ) สภาพของห้องที่ใช้ในการทดลอง



รูปที่ 6.15 เครื่องต้นแบบที่ทำขึ้น



รูปที่ 6.15 (ต่อ) เครื่องต้นแบบที่ทำขึ้น

เปลี่ยนการทดลองมาเป็นตอนกลางวัน แสงที่ตกกระทบวัตถุจึงเป็นแสงที่เกิดจากธรรมชาติ คือ แสงจากดวงอาทิตย์ ดังนั้นการเปลี่ยนแปลงความสว่างจึงเกิดขึ้นอยู่ตลอดเวลา ความสว่างของภาพจึงไม่คงที่ตาม การหมุนของกล้องในการตามภาพวัตถุจึงมีปัญหา เช่น ภาพวัตถุไม่มีการเปลี่ยนแปลงตำแหน่งภาพแต่อย่างใดยังเหมือนเดิมทุกอย่าง กล้องยังหมุนเหมือนกับว่ามีวัตถุเคลื่อนที่ผ่าน เหตุเกิดเพราะว่าแสงที่ตกกระทบลงบนวัตถุทั้งสองเวลานั้นมีความสว่างไม่เท่ากัน จึงทำให้เหมือนกับว่าภาพมีความแตกต่างกัน ซึ่งปัญหานี้จะแก้ไขไม่ได้เลยนอกจากใช้แสงที่กำเนิดขึ้นเอง ความสว่างของแสงจึงจะมีค่าคงที

ดังนั้นจึงสรุปได้ว่าแสงมีผลกระทบอย่างมาก รองลงมาก็จะเป็นสีของภาพวัตถุกับสีของภาพฉาก ต้องมีความแตกต่างกัน การทำงานของเครื่องจึงจะได้ผลดี และการเคลื่อนที่ของวัตถุที่ต้องการตรวจจับต้องมีเพียงสิ่งเดียว

ภาคผนวก ก.

โปรแกรมหลัก (Main program)

```
/*********************************************
                  digitz.c
*********************************************/

#include <dos.h>
#include <graphics.h>
#include <stdlib.h>
#include <bios.h>
#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <mem.h>
#include <dir.h>
#include <string.h>
#include <fcntl.h>

#define mem_vrm 0xa000
#define pi 3.14159
#define constant 1.87
#define right 0x4d
#define left 0x4b
#define up 0x48
#define down 0x50
#define stepts 200

void vdo2buff2(void);
void chk_ID_card(void);
void digitizeBW(unsigned,unsigned);
void read_addr(void);
void set_mode(int);
void chk_digi_state(void);
void colorpalet(void);
void save_file(void);
void read_mem(void);
void read_mem_disp(void);
void dis_play(void);
void menu(void);
void read_data(char name[]);
void read_data1(char name[]);
void binsub(void);
void dis_play1(void);
void swap(void);
void threshold(void);
char findmax(void);
void newpic(void);
void init_rs232(void);
void send_data(int);
int rport(void);
int  readkey(void);
void centroid(void);
void subtract(void);

extern digit();
extern display();
extern setpalat();
unsigned ctrlport,Cardseg;
unsigned char  *imdisp,*imone,*imtwo;
int sfile=0;
unsigned char max = 0;
float Yc,Xc;
```

```
/*****************************
     binary  subtractor
*****************************/
void binsub(void)
{
  unsigned long cnt;
  for(cnt=0;cnt<=(unsigned char)128*128-1;cnt++)

  {
    imone[cnt] = imone[cnt] - imtwo[cnt];
    abs(imone[cnt]);
  }
  return;
}

/*****************************
        new picture
*****************************/
void newpic(void)
{
  unsigned char MAX;
  unsigned long cnt;
  MAX = findmax();
  for(cnt=0;cnt<=(unsigned char)128*128-1;cnt++)
    imone[cnt] = MAX - abs(imtwo[cnt] - imone[cnt]);
}

/*****************************
          subtract
*****************************/
void subtract(void)
{
  unsigned long cnt;
  for(cnt=0;cnt<=(unsigned char)128*128-1;cnt++)
  {
    imone[cnt] = imtwo[cnt] - imone[cnt];
    if(imone[cnt] != 0)imone[cnt] = 255;
  }
}

/*****************************************/
/*         read data form file           */
/*****************************************/
void read_data1(char fname[])
{
  FILE *stream;
  int handle;
  unsigned long cnt;
  unsigned char buff,row,col;

  if((handle = open(fname,O_BINARY))!=-1)
  {
    stream = fdopen(handle,"r");
    for (cnt=0;cnt < (unsigned char)128*128-1;cnt++)
       imdisp[cnt] = fgetc(stream);
    fclose(stream);
  }
  else
  {
    printf("file can't open...");
    exit(1);
  }
  for(row=0;row<128;row++)
    for(col=0;col<128;col++)
    {
      buff = (peekb(FP_SEG(imdisp),FP_OFF(imdisp)+(row*128)+col)>>2)&0x3f;
      imone[row*128+col] = buff;
    }
```

```
}

/*****************************
        swap data
*****************************/
void swap(void)
{
  unsigned long cnt;
  unsigned char buff1,buff2;

  for(cnt=0;cnt<(unsigned char)128*128-1;cnt++)
  {
       buff1 = imone[cnt];
     imone[cnt] = imtwo[cnt];
       imtwo[cnt] = buff1;
  }
}

/*****************************/
/*       dir file            */
/*****************************/
void dirfile(void)
{
    char fname[15],fname1[15];
      struct ffblk blk;
      int done,col,line;
      clrscr();
    printf("input file name [file.img]: ");
      gets(fname);
     .clrscr();
    strcpy(fname1,fname);
    if(strcmp(fname1,"*.img") == 0)
      {
      done = findfirst(fname1,&blk,0);
      if(done == -1)
      {
        printf("\n    file not found\n");
        printf("\npress any key to main menu");
        getch();
        sfile = 1;
        return ;
      }
      printf("\n\n");
      line = 3;
      col = 0;
      do{
        if(col > 4)
        {
          printf("\n");
          line++;
          col = 0;
        }
        printf("%s",blk.ff_name);
        col++;
        gotoxy(col*15,line);
        done = findnext(&blk);
      }while(done == NULL);
    }
    else
    {
      if(strcmp(fname1,"*.*") == 0)
      {
        printf("\n    file not found");
        printf("\n\npress any key to main menu");
        getch();
        sfile = 1;
        return;
      }
      else
```

```
      {
        read_data1(fname);
        sfile = 0;
        return;
      }
    }
  gotoxy(1,1);
  printf("input file name [file.img]: ");
  gets(fname);
  read_data1(fname);
}

/*************************************
              check ID card
*************************************/
void chk_ID_card(void)
{
  int port[] = {0x210,0x2a0,0x310,0x3a0};
  int index = 0,cnt;
  unsigned swport,time,chk1,chk2,chkport;

  for(cnt=1;cnt!=4;cnt++)
  {
    ctrlport = port[index];
    swport = ctrlport+1;
    chkport = ctrlport | 0x8001;
    chk1 = inportb(swport);
    chk1 = inportb(swport);
    chk1 = (chk1 ^ 0xf) & 0xf;
    chk2 = chk1;

    for(time=0;time<200;time++)
    {
      chk1 = inportb(chkport);
      chk1 = (chk1 ^ 0xf) & 0xf;

      if(chk1!=chk2)
        goto nextbase;
      chk2 = (chk2 ^ 0xf) & 0xf;
    }
    return ;
    nextbase: index++;
  }
  set_mode(0x03);
  printf("Card DZ-II Not Found !!");
  exit(0);
}

/*****************************************
            check digit state
*****************************************/
void chk_digi_state(void)
{
  int pluse,time;

  for(pluse=1;pluse!=0;pluse--)
  {
    time = 0;
    while((inportb(ctrlport) >> 5) & 1 != pluse)
    {
      time++;
      if(time>4000)
      {
        set_mode(0x03);
        printf("video signal not found");
        exit(0);
      }
    }
  }
```

```
}

/*************************************
              Digitize B&W
*************************************/
void digitizeBW(unsigned contrast,unsigned brightness)
{
  unsigned digicom_y = 0x80;
  unsigned digicomstop = 0;
  unsigned ref_neg;

  ref_neg = 63 - brightness;
  outportb(ctrlport+2,ref_neg);
  outportb(ctrlport+1,ref_neg+((63 - ref_neg)*(63 - contrast))/63);
  outportb(ctrlport,digicom_y);
  delay(40);
  chk_digi_state();
  outportb(ctrlport,digicomstop);
}

/*************************************
              read address
*************************************/
void read_addr(void)
{
  unsigned chksw,swport;

  swport = ctrlport+1;
  chksw = inportb(swport);
  chksw = inportb(swport);
  chksw = chksw & 0xe;
  switch(chksw)
  {
    case 0: Cardseg = 0xc400;break;
    case 2: Cardseg = 0xc800;break;
    case 4: Cardseg = 0xcc00;break;
    case 6: Cardseg = 0xd000;break;
    case 8: Cardseg = 0xd800;break;
    case 10: Cardseg = 0xdc00;break;
    case 12: Cardseg = 0xe000;break;
    case 14: Cardseg = NULL;
  }
}
/*************************************/
/*  move data from video ram to obj     */
/*************************************/
void vdo2buff2(void)
{
  unsigned char cnt;

  outportb(ctrlport,1);
  for(cnt=0;cnt<=63;cnt++)
    memcpy(MK_FP(mem_vrm,(cnt+36)*320+10),MK_FP(Cardseg,cnt*256+63),127);
  outportb(ctrlport,2);
  for(cnt=0;cnt<=63;cnt++)
    memcpy(MK_FP(mem_vrm,(cnt+100)*320+10),MK_FP(Cardseg,cnt*256+63),127);

}

/*************************************
              read ram bank
*************************************/
void read_mem(void)
{
  unsigned char col,row;
  unsigned char buff;
  outportb(ctrlport,1);
  for(row=0;row<64;row++)
    for(col=0;col<128;col++)
```

```
    {
      buff = (peekb(Cardseg,(row*256)+col+63)>>2)&0x3f;
      imone[row*128+col] = buff;
    }
  outportb(ctrlport,2);
  for(row=0;row<64;row++)
    for(col=0;col<128;col++)
    {
      buff = (peekb(Cardseg,(row*256)+col+63)>>2)&0x3f;
      imone[row*128+col+128*64] = buff;
    }
}

void read_mem_two(void)
{
  unsigned char col,row;
  unsigned char buff;
  outportb(ctrlport,1);
  for(row=0;row<64;row++)
    for(col=0;col<128;col++)
    {
      buff = (peekb(Cardseg,(row*256)+col+63)>>2)&0x3f;
      imtwo[row*128+col] = buff;
    }
  outportb(ctrlport,2);
  for(row=0;row<64;row++)
    for(col=0;col<128;col++)
    {
      buff = (peekb(Cardseg,(row*256)+col+63)>>2)&0x3f;
      imtwo[row*128+col+128*64] = buff;
    }
}

void read_mem_disp(void)
{
  unsigned char col,row;
  unsigned char buff;
  outportb(ctrlport,1);
  for(row=0;row<64;row++)
    for(col=0;col<128;col++)
      imdisp[row*128+col] = peekb(Cardseg,(row*256)+col+63);

  outportb(ctrlport,2);
  for(row=0;row<64;row++)
       for(col=0;col<128;col++)
            imdisp[row*128+col+128*64] = peekb(Cardseg,(row*256)+col+63);

/*  for(row=0;row<128;row++)
       for(col=0;col<128;col++)
       {
       buff = (peekb(FP_SEG(imdisp),FP_OFF(imdisp)+(row*128)+col)>>2)&0
x3f;
       imone[row*128+col] = buff;
       }*/
}

/*****************************************
                  set_mode
*****************************************/
void set_mode(int mode)
{
  _AH = 0;
  _AL = mode;
  geninterrupt(0x10);
}
/*************************************
        fine maximum pixel
*************************************/
```

```
char findmax(void)
{
  unsigned char x,y;
  for(y=0;y<128;y++)
    for(x=0;x<128;x++)
    {
      if(peekb(FP_SEG(imone),FP_OFF(imone)+y*128+x) > max)
        max = peekb(FP_SEG(imone),FP_OFF(imone)+y*128+x);
    }
  return max;
}

/****************************************
              threshold
****************************************/
void threshold(void)
{
  unsigned char x,y,/*smin,smax,*/MAXC,MINC,THRES;
  /*smax = 0;
  smin = 255;*/
  MAXC = 255;
  MINC = 0;
  /*for(y=0;y<128;y++)
    for(x=0;x<128;x++)
    {
      if(peekb(FP_SEG(imone),FP_OFF(imone)+y*128+x) > smax)
        smax = peekb(FP_SEG(imone),FP_OFF(imone)+y*128+x);
      else
        smin = peekb(FP_SEG(imone),FP_OFF(imone)+y*128+x);
    } */
  THRES = /*(smin + smax)/2;*/50;
  for(y=0;y<128;y++)
      for(x=0;x<128;x++)
    {
      if(peekb(FP_SEG(imone),FP_OFF(imone)+y*128+x) > THRES)
        pokeb(FP_SEG(imone),FP_OFF(imone)+y*128+x,MAXC);
      else
        pokeb(FP_SEG(imone),FP_OFF(imone)+y*128+x,MINC);
        }
}

/****************************************
            set color palette
****************************************/
void colorpalet(void)
{
  int i;
  unsigned char VGAcolor[256][3];

  for(i=0;i<=255;i++)
  {
       VGAcolor[i][0] = i/4 & 0x3f;
       VGAcolor[i][1] = i/4 & 0x3f;
       VGAcolor[i][2] = i/4 & 0x3f;
  }
  _DX = FP_OFF(VGAcolor);
  _BX = 0;
  _CX = 256;
  _AX = 0x1012;
  geninterrupt(0x10);
}

/*************************************
            save file
*************************************/
void save_file(void)
{
  char ch,file_name[20];
  FILE *file;
```

```
  printf("do you want save to file Yes/NO");
  ch = getch();
  if(ch == 'y'|| ch == 'Y')
  {
    printf("\ninput file name;[ file.img]:");
    gets(file_name);
    if((file = fopen(file_name,"w")) == NULL)
    {
      printf("cannot open file \n");
      printf("press any key");
      getch();
      exit(1);
    }
       fwrite(imdisp,1,128*128-1,file);
       fclose(file);
       if(imdisp < 0)
     {
       printf("\nfile not save:");
       getch();
       return;
     }
    printf("file save complete press any key");
    getch();
  }
}

/*********************************
              display
*********************************/
void dis_play(void)
{
  unsigned char cnt;

  for(cnt=0;cnt<128;cnt++)
       movedata(FP_SEG(imdisp),FP_OFF(imdisp)+128*cnt,0xa000,(cnt+36)
*320+10,127);
  getch();
}

/*********************************
              display
*********************************/
void dis_play_sh(void)
{
  unsigned char cnt;

  for(cnt=0;cnt<128;cnt++)
       movedata(FP_SEG(imone),FP_OFF(imone)+128*cnt,0xa000,(cnt+36)
*320+170,127);
  getch();
}

/*************************
            menu
*************************/
void menu(void)
{
  gotoxy(10,2);
  printf("MAIN MENU");
  gotoxy(8,4);
  printf("1. Digitize and Save to file");
  gotoxy(8,5);
  printf("2. Display image from file");
  gotoxy(8,6);
  printf("3. Threshold");
  gotoxy(8,7);
  printf("4. Subtractor");
  gotoxy(8,8);
```

```
  printf("5. Automatic following");
  gotoxy(8,9);
  printf("6. Exit");
  gotoxy(10,11);
  printf("Plase choice select:");
}

/***********************************
            read_data
***********************************/
void read_data(char fname[])
{
  unsigned char row,col,buff;
  FILE *fload;

  fload = fopen(fname,"r");
  if (fload < 0)
  {
    printf("connot open file \n");
    set_mode(0x03);
    exit(1);
  }
  fread(imdisp,1,128*128-1,fload);
  fclose(fload);
    if(imdisp<0)
    {
      printf("error on reading file\n");
      return;
    }
  for(row=0;row<128;row++)
    for(col=0;col<128;col++)
    {
      buff = (peekb(FP_SEG(imdisp),FP_OFF(imdisp)+(row*128)+col)>>2)&0x3f;
      imone[row*128+col] = buff;
    }
}
/***************************************
             initial rs232
***************************************/
void init_rs232(void)
{
  union REGS regs;
  regs.x.dx = 1;
  regs.h.ah = 0;
  regs.h.al = 0xe3;  /*0xe3 = 9600 , 0xc3 = 4800 , 0xa3 = 2400*/
  int86(0x14,&regs,&regs);
}

/***************************************
           send data to rs232
***************************************/
void send_data(int key)
{
  union REGS regs;
  regs.x.dx = 1;
  regs.h.al = key;
  regs.h.ah = 1;
  int86(0x14,&regs,&regs);
}

/***********************************
                  reciver from rs232
***********************************/
int rport(void)
{
  union REGS regs;
  regs.x.dx = 1;
  regs.h.ah = 2;
  int86(0x14,&regs,&regs);
```

```
  return regs.h.al;
}

/*********************************
         read keyboard
*********************************/
int  readkey(void)
{
  int keycheck;
  if(kbhit()/*bioskey(1)*/)
  {
    keycheck = bioskey(0);
    return (keycheck&0xff);
  }
  return 0;
}

/****************************************
         comput center of picture
****************************************/
void centroid(void)
{
  int area =0,i,j;
  Xc = 0.0;
  Yc = 0.0;
  for(i=0;i<128;i++)
    for(j=0;j<128;j++)
      if(imone[i*128+j] == 0)
      {
        Xc = Xc+i;
        Yc = Yc+j;
        area++;
      }
  if(Xc==0||Yc==0)
  {
    Xc = 64;
    Yc = 64;
    area = 1;
  }
  Xc = Xc/area;
  Yc = Yc/area;
}

/*********************************
               MAIN
*********************************/
void main(void)
{
  unsigned char x,y;
  float d,r,ld,ls,lp,Xc1;
  int ch2,leng,step;
  char ch,ch1=0,check;
  unsigned contrast = 32,brightness = 50;

  imdisp = (unsigned char *)malloc(128*128-1);
  if(imdisp == NULL)
  {
    printf("allocate failed imdisp \n");
    exit(1);
  }
  imone = (unsigned char *)malloc(128*128-1);
  if(imone == NULL)
  {
    printf("allocate failed imone \n");
    exit(1);
  }
  imtwo = (unsigned char *)malloc(128*128-1);
  if(imtwo == NULL)
  {
```

```
  printf("allocate failed imtwo \n");
  exit(1);
}

chk_ID_card();
read_addr();
init_rs232();
while(ch1!=-1)
{
  set_mode(0x03);
  menu();
  ch = getch();
  set_mode(0x13);
  setpalat(); /*colorpalet();*/
  outportb(ctrlport+1,contrast);
  outportb(ctrlport+2,brightness);
  switch(ch)
  {
    case '1': do{
                digit();
                vdo2buff2();
                ch2 = 0;
                if(kbhit())
                {
                  ch2 = bioskey(0);
                  if((ch2&0xff) == 0x002d)
                  {
                    if(contrast < 63)
                      contrast++;
                    else
                      contrast = 63;
                  }
                  if((ch2&0xff) == 0x002b)
                  {
                    if(contrast > 0)
                      contrast--;
                    else
                      contrast = 0;
                  }
                  if((ch2&0xff) == 0x0071)
                  {
                    if(brightness < 63)
                      brightness++;
                    else
                      brightness = 63;
                  }
                  if((ch2&0xff) == 0x0077)
                  {
                    if(brightness > 0)
                      brightness--;
                    else
                      brightness = 0;
                  }
                  if((ch2&0xff00) == 0x4800)
                  {
                    send_data(stepts);
                    send_data(up);
                  }
                  if((ch2&0xff00) == 0x4b00)
                  {
                    send_data(stepts);
                    send_data(left);
                  }
                  if((ch2&0xff00) == 0x4d00)
                  {
                    send_data(stepts);
                    send_data(right);
                  }
                  if((ch2&0xff00) == 0x5000)
```

```
            {
              send_data(stepts);
              send_data(down);
            }
          }
          outportb(ctrlport+1,contrast);
          outportb(ctrlport+2,brightness);
        }while((ch2&0xff) != 0x001b);
        outportb(ctrlport+1,contrast);
        outportb(ctrlport+2,brightness);
        getch();
        read_mem();
        read_mem_disp();
        set_mode(0x03);
        save_file();
        break;

case '2': set_mode(0x03);
        for(y=0;y<128;y++)
          for(x=0;x<128;x++)
            pokeb(FP_SEG(imdisp),FP_OFF(imdisp)+128*y+x,0);
        dirfile();
        if(sfile == 1) break;
        set_mode(0x13);
        setpalat(); /*colorpalet();*/
        dis_play();
        break;

case '3': dis_play();
        threshold();
        colorpalet();
        dis_play_sh();
        set_mode(0x03);
        gotoxy(10,4);
        printf("value fo cotrast = %d",contrast);
        gotoxy(10,5);
        printf("value fo brightness = %d",brightness);
        getch();
        break;
case '4': setpalat();
        do{
          digit();
          vdo2buff2();
          ch2 = 0;
          if(kbhit())
          {
            ch2 = bioskey(0);
            if((ch2&0xff) == 0x002d)
            {
              if(contrast < 63)
                contrast++;
              else
                contrast = 63;
            }
            if((ch2&0xff) == 0x002b)
            {
              if(contrast > 0)
                contrast--;
              else
                contrast = 0;
            }
            if((ch2&0xff) == 0x0071)
            {
              if(brightness < 63)
                brightness++;
              else
                brightness = 63;
            }
            if((ch2&0xff) == 0x0077)
```

```
    {
      if(brightness > 0)
        brightness--;
      else
        brightness = 0;
    }
  }
  outportb(ctrlport+1,contrast);
  outportb(ctrlport+2,brightness);
}while((ch2&0xff) != 0x001b);
digit();
vdo2buff2();
getch();
read_mem_two();
ch2 = 0;
do{
   digit();
   vdo2buff2();
}while(!bioskey(1));
getch();
digit();
vdo2buff2();
read_mem();
newpic();
colorpalet();
dis_play_sh();
threshold();
dis_play_sh();
centroid();
set_mode(3);
gotoxy(10,10);
printf("center of Xc = %f\n",Xc);
gotoxy(10,12);
printf("center of Yc = %f\n",Yc);
getch();
set_mode(0x13);
setpalat();
if(Yc>64)
{
  Xc1 = Yc-64;
  (int)step = (1.25*Xc1)/0.30;
  /*gotoxy(10,14);
  printf("step of Yc > 64 = %d\n",step);
  getch();*/
  send_data(step);
  send_data(left);
}
else
{
  Xc1 = 64 - Yc;
  (int)step = (1.25*Xc1)/0.30;
  /*gotoxy(10,14);
  printf("step of Yc <= 64 = %d\n",step);
  getch();*/
  send_data(step);
  send_data(right);
}
do{
  digit();
  vdo2buff2();
}while(!kbhit()/*treadkey() != 0x1b*/);
/*set_mode(0x13);*/
/*swap();
threshold();
binsub();
colorpalet();
dis_play_sh();*/
break;
```

```
case '5': set_mode(3);
          gotoxy(10,3);
          printf("enter long is cm value: ");
          scanf("%d",&leng);
          d = (float)leng/constant;
          r = pi * (float)leng;
          ld = r/180;
          ls = ld/20;
          lp = d/128;
          printf("\nd = %f",d);
          printf("\nr = %f",r);
          printf("\nld = %f",ld);
          printf("\nls = %f",ls);
          printf("\nlp = %f",lp);
          getch();
          set_mode(0x13);
          setpalat();
          do{
            digit();
            vdo2buff2();
            ch2 = 0;
            if(kbhit())
            {
              ch2 = bioskey(0);
              if((ch2&0xff) == 0x002d)
              {
                if(contrast < 63)
                  contrast++;
                else
                  contrast = 63;
              }
              if((ch2&0xff) == 0x002b)
              {
                if(contrast > 0)
                  contrast--;
                else
                  contrast = 0;
              }
              if((ch2&0xff) == 0x0071)
              {
                if(brightness < 63)
                  brightness++;
                else
                  brightness = 63;
              }
              if((ch2&0xff) == 0x0077)
              {
                if(brightness > 0)
                  brightness--;
                else
                  brightness = 0;
              }
            }
            outportb(ctrlport+1,contrast);
            outportb(ctrlport+2,brightness);
          }while((ch2&0xff) != 0x001b);
          digit();
          read_mem_two();
          do{
            do{
                digit();
                vdo2buff2();
                read_mem();
                newpic();
                threshold();
                centroid();
                /*set_mode(3);
                gotoxy(10,10);
                printf("center of Xc = %f\n",Xc);
```

```
                    gotoxy(10,12);
                    printf("center of Yc = %f\n",Yc);
                    getch();
                    set_mode(0x13);
                    setpalat();*/
                  }while(Xc==64 || Yc==64);
                  /*set_mode(3);
                  gotoxy(10,10);
                  printf("center of Xc = %f\n",Xc);
                  gotoxy(10,12);
                  printf("center of Yc = %f\n",Yc);
                  getch();
                  set_mode(0x13);
                  setpalat();*/
                  if(Yc>64)
                  {
                    Xc1 = Yc-64;
                    step = lp*Xc1/ls;
                    send_data(step);
                    send_data(left);
                  }
                  if(Yc<=64)
                  {
                    Xc1 = 64 - Yc;
                    step = lp*Xc1/ls;
                    send_data(step);
                    send_data(right);
                  }
                  do{
                    digit();
                    vdo2buff2();
                    check = rport();
                  }while(check == 0x05);
                    read_mem_two();
                 }while(readkey() != 0x1b);
                 do{
                   digit();
                   vdo2buff2();
                 }while(readkey() != 0x1b);
                 break;

      case '6': set_mode(0x03);
                free(imdisp);
                free(imone);
                free(imtwo);
                exit(0);
    }
  }
}
```

ภาคผนวก ข.

โปรแกรมย่อยการเซต palat

```
_TEXT   SEGMENT  BYTE  PUBLIC  'CODE'
        PUBLIC _setpalat
        ASSUME  CS:_TEXT
        LutSym  db  256*3 dup(0)

_setpalat PROC  NEAR
        push  bp
        mov   bp,sp
        push  si
        push  di
        push  ds
        ;
        mov   ax,cs
        mov   es,ax
        mov   di,offset LutSym
        mov   al,0
        mov   cx,64
NextLev:
        stosb
        stosb   ;Y
        stosb

        stosb
        inc di   ;R
        inc di

        inc di
        stosb    ;G
        inc di

        inc di
        inc di    ;B
        stosb

        inc al
        loop NextLev
        mov   ax,cs
        mov   es,ax
        mov   ah,10h
        mov   al,12h
        mov   bx,0
        mov   cx,256
        mov   dx,offset LutSym
        int   10h
        ;
        pop   ds
        pop   di
        pop   si
        pop   bp
        ret

_setpalat ENDP
_TEXT   ends
        end
```

โปรแกรมย่อยการสั่งถ่ายภาพ

```
_TEXT   SEGMENT  BYTE  PUBLIC  'CODE'
        PUBLIC _digit
        ASSUME  CS:_TEXT

_digit  PROC  NEAR
        push  bp
        mov   bp,sp
        push  si
        push  di
        push  ds
        ;
        mov   ax,cs
        mov   ds,ax
        mov   dx,210h
        mov   al,80h
        out   dx,al
        mov   cx,3          ;chk 10 big loops
digi1:
        push  cx
        mov   cx,0          ;chk loop 65536 loops
digi1a:
        in    al,dx         ;wait for digiactv = 1
        and   al,20h
        jnz   digi1b        ;if digiactv=1 then jmp next
        loop  digi1a        ;if digiactv=0 then count loop
        pop   cx
        loop  digi1         ;count big loops
        stc                  ;if time out (camera not ready) then
        jmp   digi2c        ;   set carry flag,quit
digi1b:
        pop   cx            ;pop up old cx
        mov   cx,3          ;chk 10 big loops
digi2:
        push  cx
        mov   cx,0          ;chk loop 65536 loops
digi2a:
        in    al,dx         ;wait for digiactv = 0
        and   al,20h
        jz    digi2b        ;if digiactv=0 then jmp next
        loop  digi2a        ;if digiactv=1 then count loop
        pop   cx
        loop  digi2         ;count big loops
        stc                  ;if time out (camera not ready) then
        jmp   digi2c        ;   set carry flag,quit
digi2b:
        pop   cx            ;pop up old cx
        clc                  ;   clear carry flag if ok
digi2c:
        mov   al,0
        out   dx,al
        ;
        pop   ds
        pop   di
        pop   si
        pop   bp
        ret

_digit  ENDP
_TEXT   ends
        end
```

โปรแกรมย่อยการอ่านข้อมูลภาพมาแสดงผลหน้าจอ

```
_TEXT   SEGMENT  BYTE  PUBLIC  'CODE'
        PUBLIC _display
        ASSUME  CS:_TEXT

        Count   dw  1fh
        Step    db  153,1

_display PROC  NEAR
        push  bp
        mov   bp,sp
        push  si
        push  di
        push  ds
        ;
        mov   ax,0a000h     ;VDO Ram
        mov   es,ax
        mov   dx,210h
        mov   al,0
        out   dx,al
        mov   bp,0
        mov   ax,0cc00h
        mov   ds,ax
        mov   di,0          ;320*20+32
        mov   Count,0
        mov   ax,Count
        sub   al,al
        mov   si,ax
        mov   cx,160        ;160 Line to VGA
mos1:
        push  cx
        push  si
        push  di
        mov   cx,256
        rep   movsb
        pop   di
        pop   si
        pop   cx
        add   di,320
        mov   ax,word ptr cs:Step
        add   Count,ax
        mov   ax,Count
        sub   al,al
        mov   si,ax
        mov   al,0
cls2:
        cmp   si,4000h
        jb    cls3
        sub   si,4000h
        inc   al          ;Increment page
        jmp   cls2
cls3:
        out   dx,al
        loop  mos1
        ;
        pop   ds
        pop   di
        pop   si
        pop   bp
        ret

_display ENDP
_TEXT   ends
        end
```

ภาคผนวก ค.

โปรแกรมภาษาแอสเซมบลี้ของ MCS-51

```
                ; ****************************************
                ;              project.asm
                ; ****************************************

0000            cpu "8051.tbl"
0000            hof "int8"
0098 =          scon: equ 098h
0089 =          tmod: equ 089h
008D =          th1:  equ 08dh
008E =          tr1:  equ 08eh
00E0 =          a:    equ 0e0h
0098 =          ri:   equ 098h
0099 =          ti:   equ 099h
0083 =          dph:  equ 083h
0082 =          dpl:  equ 082h
0090 =          port1: equ 090h
0099 =          sbuf: equ 099h
00A8 =          ie:   equ 0a8h
0088 =          it0:  equ 088h
008A =          it1:  equ 08ah
00F0 =          b:    equ 0f0h


0000                    org  0000h

0000 7A99       start:      mov r2,#99h
0002 7B99                   mov r3,#99h
0004 759852                 mov scon,#52h
0007 758920                 mov tmod,#20h
000A 758DFD                 mov th1,#0fdh
000D D28E                   setb tr1
000F E599                   mov a,sbuf

0011 3098FD     rx:         jnb ri,rx
0014 C298                   clr ri
0016 E599                   mov a,sbuf
0018 FC                     mov r4,a
0019 3098FD     rx1:        jnb ri,rx1
001C C298                   clr ri
001E E599                   mov a,sbuf

                ;call sub porgram
0020 B44B08     case_0:     cjne a,#04bh,case_1
0023 12004C                 lcall left
0026 1200AD                 lcall send
0029 80E6                   sjmp rx

002B B44D08     case_1:     cjne a,#04dh,case_2
002E 120061                 lcall right
0031 1200AD                 lcall send
0034 80DB                   sjmp rx

0036 B44808     case_2:     cjne a,#048h,case_3
0039 120076                 lcall up
003C 1200AD                 lcall send
003F 80D0                   sjmp rx

0041 B450CD     case_3:     cjne a,#050h,rx
0044 12008B                 lcall down
0047 1200AD                 lcall send
004A 80C5                   sjmp rx
```

```
                ;stepping motor control
004C EC         left:       mov a,r4
004D 9401                   subb a,#1
004F 400F                   jc exit
0051 EA         left1:      mov a,r2
0052 23                     rl a
0053 FA                     mov r2,a
0054 54F0                   anl a,#0f0h
0056 FE                     mov r6,a
0057 EB                     mov a,r3
0058 540F                   anl a,#0fh
005A 4E                     orl a,r6
005B 1200A0                 lcall send_data
005E DCF1                   djnz r4,left1
0060 22         exit:       ret

0061 EC         right:      mov a,r4
0062 9401                   subb a,#1
0064 400F                   jc exit1
0066 EA         right1:     mov a,r2
0067 03                     rr a
0068 FA                     mov r2,a
0069 54F0                   anl a,#0f0h
006B FE                     mov r6,a
006C EB                     mov a,r3
006D 540F                   anl a,#0fh
006F 4E                     orl a,r6
0070 1200A0                 lcall send_data
0073 DCF1                   djnz r4,right1
0075 22         exit1:      ret

0076 EC         up:         mov a,r4
0077 9401                   subb a,#1
0079 400F                   jc exit2
007B EB         up1:        mov a,r3
007C 03                     rr a
007D FB                     mov r3,a
007E 540F                   anl a,#0fh
0080 FE                     mov r6,a
0081 EA                     mov a,r2
0082 54F0                   anl a,#0f0h
0084 4E                     orl a,r6
0085 1200A0                 lcall send_data
0088 DCF1                   djnz r4,up1
008A 22         exit2:      ret

008B EC         down:       mov a,r4
008C 9401                   subb a,#1
008E 400F                   jc exit3
0090 EB         down1:      mov a,r3
0091 23                     rl a
0092 FB                     mov r3,a
0093 540F                   anl a,#0fh
0095 FE                     mov r6,a
0096 EA                     mov a,r2
0097 54F0                   anl a,#0f0h
0099 4E                     orl a,r6
009A 1200A0                 lcall send_data
009D DCF1                   djnz r4,down1
009F 22         exit3:      ret

                ;send data to port1
00A0 F590       send_data:  mov port1,a
00A2 7811                   mov r0,#11h

00A4 7930       delay:      mov r1,#30h
00A6 00         del:        nop
00A7 00                     nop
```

```
00A8 D9FC               djnz r1,del
00AA D8F8               djnz r0,delay
00AC 22                 ret

                ;send data to rs232
00AD 7405       send:   mov a,#05h
00AF F599               mov sbuf,a
00B1 3099FD     sendl:  jnb ti,sendl
00B4 C299               clr ti
00B6 22                 ret
0000                    end
```

หนังสืออ้างอิง

1. วรวัฒน์ บัณฑวังกูร, สราวุฒิ เอ็งอุทัยวัฒน์ "การเขียนโปรแกรมใช้งาน EGA/VGA" จัดพิมพ์โดยบริษัทซีเอ็ดยูเคชั่น พ.ศ. 2536
2. พิพัฒน์ เลาหสงคราม "การทดลองไมโครคอนโทรลเลอร์" ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
3. ทีมงาน ETT "ET-01 MICRO TRAINING USER'S MANUAL" กุมภาพันธ์ 1990
4. ETT CO., LTD. "MCS-51 SINGLE BOARD CONTROLLER"
5. ศิวัฒน์ ศิวะบวร, พรชัย จักรธำรงค์, จิรศักดิ์ ชัยวิริยะกุล "การประยุกต์ใช้งานภาษาซี" จัดพิมพ์โดยบริษัท ซีเอ็ดยูเคชั่น จำกัด พ.ศ 2536
6. ธนา ศรีแสง, ประจักษ์ มณีมงคล, ปัญญา พงษ์วิรัตน์ "ระบบควบคุมกการติดตามวัตถุอัตโนมัติด้วยข้อมูลภาพ" ปริญญานิพนธ์ปริญญาบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2535
7. Harley R. Myler, Arthur R. Weeks "Computer Imaging Recipes in C" PTR Prentice Hell 1993