ปีการศึกษา 2534

สแกนเนอร์

SCANNER

โดย

1.นายคมกริช ศิริแสงชัยกุล 31.1019

2.นายวิบูลย์ กอจรัญจิตต์ 31.1254

3.นายสุรเดช ตรีไตรลักษณะ 31.1365

อาจารย์ที่ปรึกษา

ผศ.พลผดุง ผดุงกุล

ปริญญานิพนธ์ปีการศึกษา 2534

ภาควิชาอิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง สแกนเนอร์

(Scanner)

ผู้จัดทำ

1.นายคมกริช ศิริแสงชัยกุล 31.1019

2.นายวิบูลย์ กอจรัญจิตต์ 31.1254

3.นายสุรเดช ตรีไตรลักษณ์ 31.1365

อาจารย์ที่ปรึกษา

(ผศ.พลผดุง ผดุงกุล)

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

สแกนเนอร์

SCANNER

คมกริช ศิริแสงชัยกุล รหัส 31.1019
วิบูลย์ กอจรัญจิตต์ รหัส 31.1254
สุรเดช ตรีไตรลักษณะ รหัส 31.1365

ผศ.พลผดุง ผดุงกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2534

บทคัดย่อ

เครื่องสแกนเนอร์ที่ได้ทำการพัฒนาขึ้นมานี้ มีวัตถุประสงค์เพื่อใช้เป็นอุปกรณ์รับข้อมูลภาพขาว ดำ โดยการควบคุมของไมโครโปรเซสเซอร์เบอร์ Z-80 ซึ่งใช้ควบคุมการอ่านข้อมูลทีละจุดจากหัวอ่านข้อมูลซึ่งดัดแปลงมาจากหัวอ่านรหัสแถบ ข้อมูลที่ได้จะถูกส่งเข้าไปเก็บไว้ในเครื่องไมโครคอมพิวเตอร์ ซึ่งได้เขียนโปรแกรมสนับสนุนไว้ โปรแกรมดังกล่าวสามารถนำข้อมูลที่อ่านเข้ามา มาแสดงผล ย่อภาพ ขยายภาพ และหมุนภาพได้ ผู้ใช้สามารถทำการแก้ไขและตกแต่งรูปใหม่และพิมพ์รูปภาพออกทางเครื่องพิมพ์ นอกจากนี้โปรแกรมสนับสนุน ยังสามารถแปลงข้อมูลภาพที่ได้ให้เป็นแบบมาตรฐานที่นิยมใช้กันทั่วไป คือ มาตรฐานไฟล์แบบพีซีเอ็กซ์(.PCX) ซึ่งจะสามารถนำไปใช้กับโปรแกรมสำเร็จรูปทางกราฟิกที่มีใช้งานกันอยู่แพร่หลายได้หลายโปรแกรม เครื่องสแกนเนอร์นี้จะสามารถสแกนภาพได้ด้วยความละเอียดสูงถึง 150 จุดต่อนิ้ว โดยที่ภาพนั้นจะต้องมีขนาดไม่เกิน 8 นิ้ว x 11 นิ้ว สำหรับการควบคุมตำแหน่งหัวอ่านข้อมูลนั้นใช้สเต็ปมอเตอร์ 2 ตัว ซึ่งถูกควบคุมโดยไมโครโปรเซสเซอร์

SCANNER

Comgrit Sirisangchaikul No. 31.1019

Viboon Korjaranjit No. 31.1254

Suradej Tritrailucksana No. 31.1365

Asst.Prof.Polphadung Phadungkul, Advisor

Academic year 1991

Abstract

The purpose of this project, SCANNER, is to create an input machine. This machine scans pictures or texts and then converts them to electronic signals. Controlled by Z-80 microprocessor, it reads data dot by dot from a scanner head, an input device, and sends them to a personal computer. Not only the microprocessor controls the reader, but also two stepping motors to get reader's position precisely. An application program was prepared for display the data on the personal computer. It has some special display functions. These are the functions of rotation, zoom in, zoom out, and edit. Moreover, data can be printed out and saved in the PCX standard. So, general graphic applications can use this advantage to edit them. This SCANNER can scan at the resolution of 150 dots per inch within the area of 8 inches by 11 inches.

บทที่ 1
บทนำ

ในระบบคอมพิวเตอร์ทั่วไป จะมีอุปกรณ์อินพุทและเอาท์พุทอยู่หลายชนิดซึ่งแต่ละชนิดก็แตกต่างกันออกไปทั้งโครงสร้างและหลักการทำงาน ทั้งนี้ย่อมขึ้นอยู่กับชนิดของข้อมูลที่อุปกรณ์อินพุทเอาท์พุทเหล่านั้นเกี่ยวข้องด้วย

สแกนเนอร์(Scanner) เป็นอุปกรณ์อินพุทอีกประเภทหนึ่ง ซึ่งมีการใช้งานโดยกว้างขวางมีหน้าที่รับข้อมูลที่เป็นภาพ แล้วทำการแปลงให้เป็นข้อมูลทางดิจิตอล ส่งให้กับคอมพิวเตอร์เพื่อประมวลผลต่อไป โดยสามารถแบ่งเป็นชนิดต่างๆ ตามลักษณะของข้อมูลที่รับได้เป็น

1.ชนิดที่รับข้อมูลเป็นจุด

2.ชนิดที่รับข้อมูลเป็นแถว

ในปัจจุบันเครื่องสแกนเนอร์ที่ใช้งานอยู่โดยทั่วไป จะรับข้อมูลเป็นแถวโดยใช้ซีซีดี (CCD: Charge Couple Device) เป็นตัวอ่านข้อมูล เนื่องจากมีประสิทธิภาพสูงสามารถอ่านข้อมูลได้ครั้งละหลายๆจุด ซึ่งข้อดีนี้ทำให้ซีซีดีถูกนำไปใช้อย่างกว้างขวางในงานต่างๆที่เกี่ยวกับภาพ เช่น กล้องวีดีโอ กล้องถ่ายรูป เครื่องถ่ายเอกสาร สำหรับสแกนเนอร์แบบอ่านข้อมูลเป็นจุดมีบทบาทน้อยมากเนื่องจากอ่านข้อมูลได้ช้ามาก แต่มีข้อดีคือสามารถรู้ตำแหน่งโคออร์ดิเนตต่างๆได้ทันทีขณะทำการสแกนข้อมูล

สำหรับสแกนเนอร์ที่ได้ทำการพัฒนาขึ้นมาในโครงงานนี้ ใช้วิธีการอ่านข้อมูลเป็นจุด เนื่องจากต้องการโครงสร้างทางกลไก ที่จะทำให้โครงงานนี้สามารถนำไปพัฒนาต่อเป็นเครื่องเจาะแผ่นวงจรพิมพ์อัตโนมัติได้ ซึ่งเป็นเป้าหมายในการพัฒนาโครงงานสำหรับรุ่นต่อไป

สแกนเนอร์ที่ทำการพัฒนาขึ้นมานี้มีลักษณะดังนี้

-ใช้ในการสแกนข้อมูลภาพขาวดำที่มีขนาดไม่เกิน 8 นิ้ว x 11 นิ้ว

-ความละเอียดในการสแกนภาพ 150 จุดต่อนิ้ว

-ใช้หัวอ่านรหัสแถบเป็นตัวรับข้อมูล

-โครงสร้างของแท่นสแกนดัดแปลงมาจากโครงสร้างของพล็อตเตอร์

-ใช้ไมโครโปรเซสเซอร์ Z-80 เป็นตัวควบคุมการทำงาน

-สามารถทำงานได้เองโดยไม่ต้องอาศัยการควบคุมจากไมโครคอมพิวเตอร์

ในปริญญานิพนธ์ฉบับนี้จะกล่าวถึงทฤษฎีต่างๆ ที่เป็นพื้นฐานในการพัฒนาโครงงานในบทที่ 2 เช่น โครงสร้างของสเต็ปมอเตอร์ รวมทั้งวิธีการขับสเต็ปมอเตอร์ ปัญหา และวิธีแก้ไขปัญหาที่อาจเกิดขึ้นในการขับสเต็ปมอเตอร์ นอกจากนี้ยังจะได้กล่าวถึงทฤษฎีการจัดการข้อมูลภาพซึ่งจะอธิบายถึงวิธีที่ใช้ในการเก็บข้อมูลภาพให้ประหยัดเนื้อที่หน่วยความจำ และสุดท้ายกล่าวถึงทฤษฎีในการติดต่อสื่อสารข้อมูลคอมพิวเตอร์

บทที่ 3 จะกล่าวถึงโครงสร้าง หลักการ และการทำงานของส่วนต่างๆในสแกนเนอร์ ซึ่งจะเริ่มจากหลักการในการรับภาพในส่วนของหัวอ่านข้อมูล โดยจะอธิบายตั้งแต่เรื่องการสะท้อนของแสงและอุปกรณ์หลักในการรับแสง รวมทั้งสัญญาณไฟฟ้าที่ได้ และวิธีแปลความหมายสัญญาณที่ได้ว่าภาพที่ได้มีสีอย่างไร และจะอธิบายถึงวงจรที่ใช้ ส่วนประกอบที่สองที่กล่าวถึงคือสเต็ปมอเตอร์และวงจรขับ ซึ่งอธิบายถึงวงจรและการทำงานแต่เพียงสั้นๆ เพราะเป็นการประยุกต์ใช้งานจากทฤษฎีที่ได้กล่าวไว้แล้วในบทที่ 2 เช่นเดียวกันกับส่วนของบอร์ดควบคุม ส่วนประกอบต่อไป คือส่วนที่ใช้ติดต่อข้อมูลซึ่งจะอธิบายถึงการนำความรู้พื้นฐานจากทฤษฎีการติดต่อข้อมูลในบทที่ 2 มาสร้างเป็นระบบและวิธีการติดต่อข้อมูล (Protocol) ที่เหมาะสมกับการใช้งานกับโครงงานนี้ และสุดท้ายจะอธิบายถึงโครงสร้างทางกลไกที่ใช้ในการจับหัวอ่านข้อมูลให้เลื่อนไปยังตำแหน่งต่างๆ เพื่อทำการอ่านข้อมูล

ในบทต่อมาอีกสองบทจะกล่าวถึงการพัฒนาโปรแกรมมอนิเตอร์ที่ใช้เป็นซอฟแวร์(Software) ของไมโครโปรเซสเซอร์ Z-80 ซึ่งใช้ควบคุมการทำงานของเครื่องสแกนเนอร์ และการพัฒนาโปรแกรมสนับสนุนที่ใช้บนไมโครคอมพิวเตอร์ตั้งแต่ในส่วนของการแสดงผล การตกแต่งแก้ไขรูปภาพ การพิมพ์ภาพออกทางเครื่องพิมพ์ และการแปลงไฟล์ข้อมูลเป็นแบบพีซีเอ็กซ์(.PCX) นอกจากนี้ในทั้งสองบทนี้จะกล่าวถึงการพัฒนาโปรแกรมในส่วนของการติดต่อสื่อสารข้อมูลระหว่างกันอีกด้วย

บทที่ 6 และ 7 เป็นการนำผลจากการทดลองโครงงานมานำเสนอเป็นรายงาน เพื่อแสดงประสิทธิภาพของสแกนเนอร์ สรุปผล สรุปปัญหาและการแก้ไข และกล่าวถึงแนวความคิดในการพัฒนาโครงงานต่อไป

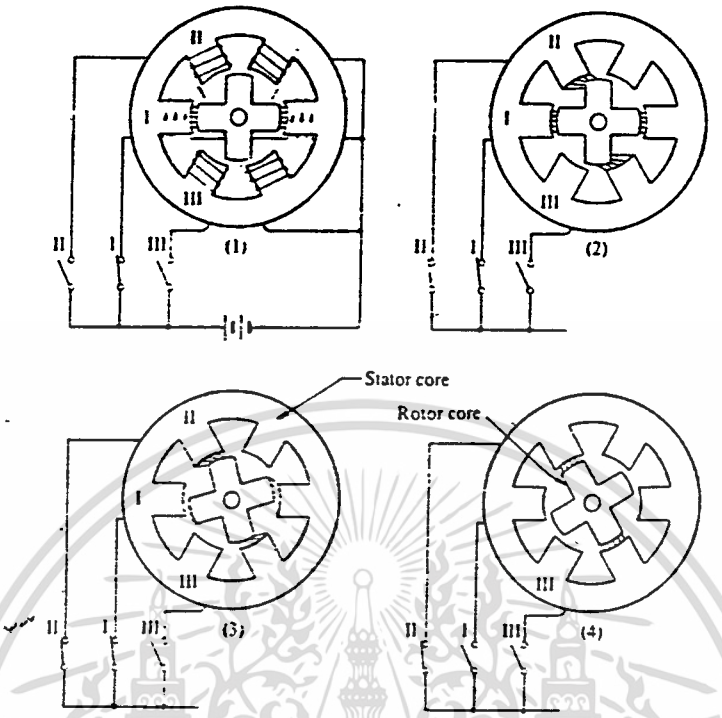
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 สเต็ปมอเตอร์และวงจรขับ

ในการควบคุมตำแหน่งของหัวอ่านข้อมูลให้เลื่อนไปยังตำแหน่งต่างๆ เพื่อทำการอ่านข้อมูลที่มีความละเอียดสูงถึง 150 จุดต่อนิ้วให้ได้ความถูกต้องแม่นยำนั้น สามารถทำได้โดยการควบคุมตำแหน่งของมอเตอร์ซึ่งได้แก่ ดีซีมอเตอร์ และสเต็ปมอเตอร์ เป็นต้น แต่เนื่องจากการควบคุมตำแหน่งของดีซีมอเตอร์นั้น แม้ว่าจะสามารถทำให้มีความละเอียดสูงได้มากกว่าสเต็ปมอเตอร์มาก แต่วงจรที่ใช้ในการควบคุมมีความยุ่งยากซับซ้อนมาก ดังนั้นในโครงงานนี้จึงใช้สเต็ปมอเตอร์แทน เนื่องจากสามารถควบคุมตำแหน่งได้สะดวกกว่าดีซีมอเตอร์ ขณะเดียวกันก็มีความแม่นยำสูงด้วย สำหรับในเรื่องของความละเอียดในการสแกนภาพสามารถแก้ปัญหาได้โดยการเปลี่ยนขนาดแกนโรเตอร์ของสเต็ปมอเตอร์ หรือการทดรอบ ซึ่งจะได้กล่าวถึงต่อไป รายละเอียดต่างๆที่ควรทราบเกี่ยวกับสเต็ปมอเตอร์มีดังนี้

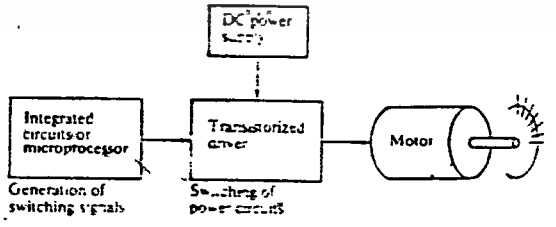
2.1.1 ลักษณะทั่วไปของสเต็ปมอเตอร์

จากรูปที่ 2-1 แสดงโครงสร้างภาคตัดขวางของสเต็ปมอเตอร์ชนิดหนึ่ง ที่มีใช้กันทั่วไปในปัจจุบัน เรียกว่า มอเตอร์สแตกเดียวแบบค่ารีลักแตนซ์เปลี่ยนแปลงได้ (Single-stack variable-reluctance motor) พิจารณารูปที่ 2-1 กำหนดให้แกนสเตเตอร์(Stator) มีขั้วแม่เหล็กถาวร (Salient pole) จำนวน 6 ขั้ว(Pole) ส่วนโรเตอร์(Rotor)มี 4 ขั้ว ทั้งสเตเตอร์และโรเตอร์ทำจากเหล็กอ่อน และมีการพันขดลวด(Coil) อยู่ 3 ชุด แต่ละชุดมีขดลวด 2 เส้นต่อกันอยู่อย่างอนุกรม ขดลวดแต่ละชุดที่พันอยู่เรียกว่า เฟส(Phase) ดังนั้นสเต็ปมอเตอร์แบบนี้จะมีสามเฟส เพราะมีขดลวด 3 ชุด กระแสไฟฟ้าจะถูกจ่ายจากแหล่งจ่ายไฟกระแสตรง(DC power supply) ไปยังขดลวดผ่านสวิทช์ ชุดที่ I,II และ III ตามทิศทางของลูกศร ในสถานะแรกขดลวดในเฟส I จะมีกระแสไหลผ่านสวิทช์ I นั่นคือเฟสที่ I ถูก กระตุ้น(Excited) ทำให้เกิดฟลักซ์แม่เหล็กจะเกิดขึ้นในช่องอากาศ (Air-gap)เนื่องจากการกระตุ้น ซึ่งในขณะนั้นขั้วแม่เหล็กถาวรของสเตเตอร์ทั้งสองขั้วของเฟสที่ I จะอยู่ในแนวเส้นตรงเดียวกันกับขั้ว ของโรเตอร์ ซึ่งเป็นสภาวะสมดุลย์ไดนามิค เมื่อสวิทช์ II ปิดวงจรเพื่อกระตุ้นเฟสที่ II แทนเฟสที่ I ฟลักซ์แม่เหล็กจะเกิดขึ้นที่ขั้วสเตเตอร์ของเฟส II ดังแสดงในรูปที่ 2-1 (2) และเกิดแรงบิด(Torque) ในทิศทางทวนเข็มนาฬิกาตามแนวการเกิดฟลักซ์แม่เหล็ก ดังนั้นโรเตอร์จะเข้าไปอยู่ในสถานะใหม่ดังรูปที่ 2-1 (3)



รูปที่ 2-1: โครงสร้างของสเต็ปมอเตอร์

ดังนั้นโรเตอร์จะหมุนไปด้วยมุมคงที่ค่าหนึ่ง (ในที่นี้มีค่า 15 องศา) ซึ่งเรียกว่า มุมสเต็ป (Step angle) ถ้าสวิทช์ I เปิดวงจร โรเตอร์จะหมุนไปอีก 15 องศาและเข้าสู่สถานะใหม่ดังรูปที่ 2-1 (4) จะเห็นว่าตำแหน่งเชิงมุมต่างๆ ของโรเตอร์สามารถควบคุมได้ด้วยสวิทช์ให้เลื่อนไปทีละสเต็ป ถ้าเราเปิดปิดสวิทช์ติดต่อกันเป็นลำดับที่ถูกต้อง โรเตอร์จะหมุนไปทีละสเต็ปติดต่อกัน ความเร็วเฉลี่ยก็จะสามารถควบคุมได้ด้วยความเร็วในการเปิดและปิดสวิทช์



รูปที่ 2-2: บล็อกไดอะแกรมของการขับสเต็ปมอเตอร์

ในปัจจุบันเราใช้ทรานซิสเตอร์เป็นสวิทช์อิเล็กทรอนิคส์ สำหรับใช้ในการขับสเต็ปมอเตอร์ โดยการป้อนสัญญาณสวิทชิ่ง(Switching signals)จากไอซี หรือไมโครโปรเซสเซอร์ไปควบคุมการทำงานของทรานซิสเตอร์ ดังแสดงในรูปที่ 2-2

จากที่ได้กล่าวมาแล้วทั้งหมดข้างต้นจะเห็นว่า **สเต็ปมอเตอร์ คือ มอเตอร์ไฟฟ้าชนิดหนึ่งซึ่งแปลงสัญญาณอินพุทที่อยู่ในรูปของสัญญาณดิจิตอลให้เป็นการเคลื่อนที่ทางกล** เมื่อเปรียบเทียบกับอุปกรณ์ชนิดอื่นๆ ซึ่งสามารถแสดงหน้าที่ได้ในลักษณะเดียวกันแล้ว ระบบที่ใช้สเต็ปมอเตอร์จะมีข้อดีกว่าระบบที่ใช้มอเตอร์ชนิดอื่น หลายประการดังนี้

1. ไม่จำเป็นต้องมีการควบคุมแบบป้อนกลับ(Feedback Control) ไม่ว่าจะเป็นการควบคุมทางตำแหน่งหรือเป็นการควบคุมความเร็วของสเต็ปมอเตอร์

2. สามารถควบคุมตำแหน่งได้อย่างแม่นยำ

3. สเต็ปมอเตอร์สามารถนำไปประยุกต์ใช้กับอุปกรณ์ประเภทดิจิตอลได้อย่างสะดวก

และด้วยเหตุผลเหล่านี้เองสเต็ปมอเตอร์จึงได้ถูกนำไปใช้อย่างกว้างขวางในเครื่องมือและระบบอุตสาหกรรมต่างๆ ซึ่งรวมทั้งในโครงงานนี้ด้วย

2.1.2 ระบบควบคุมสเต็ปมอเตอร์แบบเปิดลูป(Open loop)

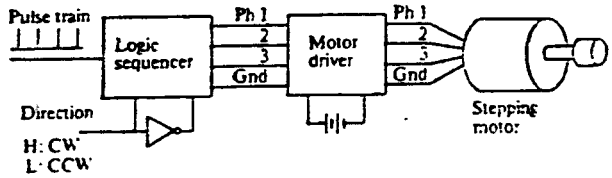
สเต็ปมอเตอร์เป็นมอเตอร์ที่มีลักษณะเฉพาะตัวแตกต่างกับมอเตอร์ไฟฟ้าโดยทั่วไป กล่าวคือสามารถขับให้หมุนได้โดยตรงจากแหล่งจ่ายพลังงาน(Power Supply) ยิ่งไปกว่านั้นยังสามารถควบคุมได้โดยไม่ต้องอาศัยการป้อนกลับที่มีราคาแพง วิธีการควบคุมแบบนี้เรียกว่าเป็นการควบคุมแบบเปิดลูป

แม้ว่าการควบคุมแบบเปิดลูปจะเป็นวิธีที่ประหยัด แต่ก็มีข้อจำกัดบางประการ ตัวอย่างเช่น การหมุนของโรเตอร์อาจจะเกิดการออสซิลเลต(Oscillation) หรือเกิดความผิดพลาดทางตำแหน่งเมื่อหมุนด้วยความเร็วบางค่า และด้วยเหตุนี้เองความเร็วและความเร่งของสเต็ปมอเตอร์ที่ควบคุมแบบเปิดลูปจึงไม่สามารถทำได้เร็วเท่ากับดีซีมอเตอร์ อย่างไรก็ดีปัญหานี้ก็สามารถแก้ไขให้ดีขึ้นได้ ซึ่งจะได้กล่าวถึงต่อไปในภายหลัง

หลักการพื้นฐานที่สำคัญของการควบคุมแบบเปิดลูปของสเต็ปมอเตอร์ มีดังนี้

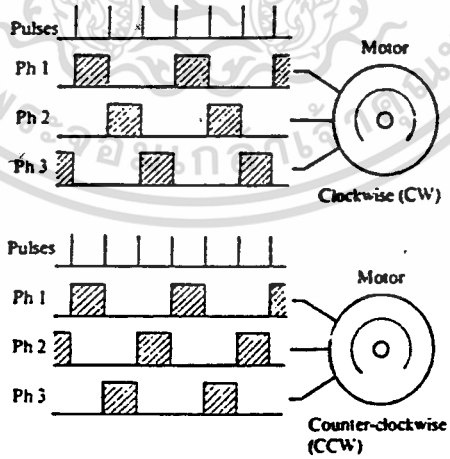
2.1.2.1 โครงสร้างของระบบควบคุม

ลักษณะที่สำคัญของสเต็ปมอเตอร์คือ มอเตอร์จะหมุนไปด้วยมุมคงที่ค่าหนึ่งเมื่อมีพัลส์(Pulse) หนึ่งลูกป้อนเข้าสู่วงจรเรียงลำดับลอจิก(Logic sequencer) แสดงดังรูปที่ 2-3



รูปที่ 2-3: ระบบขับสเต็ปมอเตอร์

เมื่อวงจรเรียงลำดับลอจิกได้รับพัลส์มาลูกหนึ่ง ก็จะพิจารณาหาเฟสของขดลวดที่จะทำการกระตุ้นแล้วส่งสัญญาณไปยังวงจรขับมอเตอร์(Motor driver) ซึ่งจะทำหน้าที่จ่ายกระแสให้แก่มอเตอร์ โดยทั่วไปวงจรเรียงลำดับลอจิกมักจะสร้างขึ้นจากวงจรรวมชนิดซีมอส(CMOS) หรือทีทีแอล(TTL) และยังสามารถควบคุมทิศทางการหมุนของมอเตอร์ได้ ดังรูปที่ 2-3 เมื่อจะให้มอเตอร์หมุนในทิศทางตามเข็มนาฬิกาจะต้องป้อนลอจิกค่าสูง(High level logic) และต้องป้อนลอจิกค่าต่ำ(Low level logic) เพื่อให้หมุนในทิศทางทวนเข็มนาฬิกา ตามรูปที่ 2-4 เมื่อมอเตอร์หมุนในทิศทางตามเข็มนาฬิกา ลำดับในการกระตุ้นจะเป็น 1-2-3-1 ... และเมื่อหมุนในทิศทวนเข็มนาฬิกาจะเป็น 1-3-2-1 ... สำหรับลักษณะการกระตุ้นในรูปที่ 2-4 นี้เป็นการกระตุ้นแบบเฟสเดียว(Single phase excitation)



รูปที่ 2-4: อินพุทพัลส์และลำดับในการกระตุ้น

2.1.2.2 คุณสมบัติที่สำคัญของสเต็ปมอเตอร์

1.มุมสเต็ป สเต็ปมอเตอร์จะหมุนไปด้วยมุมคงที่เมื่อได้รับพัลส์ ดังที่ได้อธิบายไว้แล้ว ค่ามุมนี้เรียกว่า "มุมสเต็ป" ซึ่งมีหน่วยเป็นองศา มุมสเต็ปที่มีค่าต่ำจะทำให้สามารถควบคุมตำแหน่งได้ละเอียดกว่ามุมสเต็ปที่มีค่าสูง อย่างไรก็ตามความสามารถในการควบคุมตำแหน่ง ก็ยังขึ้นอยู่กับปัจจัยอื่นด้วย เช่น ขนาดของแกนโรเตอร์ อัตราส่วนในการทดรอบ เป็นต้น ความสัมพันธ์ระหว่างมุมสเต็ป(θ_s) และจำนวนสเต็ปต่อรอบการหมุน(S:Steps per revolution) จะมีค่า

$$S = 360/\theta_s$$

สเต็ปมอเตอร์แบบสี่เฟสโดยทั่วไปมักจะมีจำนวนสเต็ป 200 สเต็ปต่อรอบ สเต็ปมอเตอร์บางตัวที่ออกแบบมาใช้กับงานที่ต้องการความละเอียดสูงอาจมีจำนวนสเต็ปสูงถึง 500 จนถึง 1000 สเต็ปต่อรอบ แต่ก็มีบางชนิดที่มีมุมสเต็ปมากถึง 90 องศา 45 องศา หรือ 15 องศาทำให้มีจำนวนสเต็ปต่อรอบต่ำ

2.ความแม่นยำทางตำแหน่ง(Positioning accuracy) ความแม่นยำทางตำแหน่งเป็นคุณสมบัติสำคัญอย่างหนึ่งที่ใช้พิจารณาคุณภาพของสเต็ปมอเตอร์ สเต็ปมอเตอร์ถูกออกแบบขึ้นเพื่อให้หมุนไปด้วยค่ามุมที่ต้องการ โดยการป้อนสัญญาณพัลส์ และสามารถหยุดยังตำแหน่งที่ต้องการได้อย่างแม่นยำ สำหรับค่าความแม่นยำทางตำแหน่งของมอเตอร์แต่ละตัวขึ้นอยู่กับคุณสมบัติทางกลของมอเตอร์และชนิดของวงจรขับ

3.อัตราส่วนระหว่างแรงบิดต่อแรงเฉื่อย(Torque-to-inertia ratio) ในการใช้งานสเต็ปมอเตอร์ มักจะมีความต้องการให้มอเตอร์หมุนได้เร็วที่สุดเท่าที่จะเป็นได้ และสามารถตอบสนองต่ออินพุทพัลส์ได้ รวมทั้งการเริ่มหมุนและหยุดหมุนที่รวดเร็วด้วย ถ้าขบวนพัลส์(Pulse train) ถูกหยุดอย่างกะทันหันขณะที่มอเตอร์กำลังหมุนอยู่ที่ความเร็วค่าหนึ่ง มอเตอร์ก็ควรที่จะสามารถหยุดตรงตำแหน่งที่ได้รับพัลส์สุดท้ายด้วย นั่นแสดงว่าอัตราส่วนระหว่างแรงบิด(Torque) ต่อแรงเฉื่อย(Inertia)ของโรเตอร์จะต้องมีค่ามาก

4.อัตราการสเต็ปและความถี่พัลส์(Stepping rate and pulse frequency) ความเร็วในการหมุนของสเต็ปมอเตอร์จะมีหน่วยเป็น จำนวนสเต็ปต่อวินาที(Steps per second) อัตราการสเต็ปเป็นตัวกำหนดความเร็วในการหมุนของมอเตอร์ เพราะในสเต็ปมอเตอร์ส่วนใหญ่ จำนวนพัลส์ที่ป้อนให้แก่วงจรเรียงลำดับลอจิกจะมีค่าเท่ากับจำนวนสเต็ปที่ต้องการให้มอเตอร์หมุน ความเร็วนี้อาจจะแสดงในรูปของความถี่ก็ได้

2.1.3 การกระตุ้นสเต็ปมอเตอร์

2.1.3.1 การกระตุ้นแบบเฟสเดียว(Single phase excitation)

ตารางที่ 2-1 แสดงลำดับในการกระตุ้นแบบเฟสเดียวสำหรับสเต็ปมอเตอร์แบบสามเฟสและสี่เฟส ส่วนที่แรเงาในตารางแสดงถึงสถานะที่มีการกระตุ้นและส่วนที่ว่างแสดงว่าเฟสนั้นไม่มีกระแสจ่ายให้แก่ขดลวด นั่นคือไม่ได้รับการกระตุ้นนั่นเอง เมื่อต้องการให้สเต็ปมอเตอร์หมุนในทิศทางตามเข็มนาฬิกา ลำดับการกระตุ้นจะเป็น เฟส1 เฟส2 เฟส3 ... และเมื่อต้องการให้หมุนทวนเข็มนาฬิกา ลำดับการกระตุ้นจะเป็น เฟส3 เฟส2 เฟส1 ...

(1) Three-phase motor

	R	1	2	3	4	5	6	7	8
Phase 1	▨			▨			▨		
Phase 2		▨			▨			▨	
Phase 3			▨			▨			▨

(2) Four-phase motor

	R	1	2	3	4	5	6	7	8
Phase 1	▨				▨				▨
Phase 2		▨				▨			
Phase 3			▨				▨		
Phase 4				▨				▨	
Pulses									

Note: Symbol R indicates 'reset'.

ตารางที่ 2-1: ลำดับการกระตุ้นแบบเฟสเดียว

2.1.3.2 การกระตุ้นแบบสองเฟส(Two phase excitation)

ลำดับการกระตุ้นแบบสองเฟสสำหรับสเต็ปมอเตอร์แบบสามเฟสและสี่เฟส แสดงดังตารางที่ 2-2 จากตารางจะเห็นได้ว่าเมื่อกระแสที่ใช้กระตุ้นขดลวดซึ่งจ่ายให้ขดลวดทั้งสองเฟสถูกเปลี่ยนจากสถานะหนึ่งไปเป็นอีกสถานะหนึ่ง จะยังคงมีเฟสหนึ่งถูกกระตุ้นอยู่ ดังแสดงในตารางที่ 2-2(1) ซึ่งเฟสที่ 2 ถูกเปิดวงจรให้กระแสหยุดผ่านในขณะที่เฟสที่ 1 ปิดวงจรให้กระแสไหลผ่าน ในเวลาเดียวกันนั้นเฟสที่ 3 ก็ยังคงถูกกระตุ้นอยู่

(1) Three-phase motor

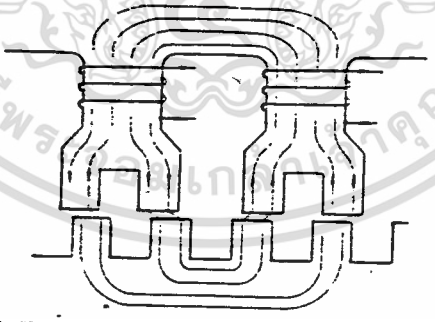
Clock state	R	1	2	3	4	5	6	7	8
Phase 1			→						
Phase 2			→						
Phase 3									

(2) Four-phase motor

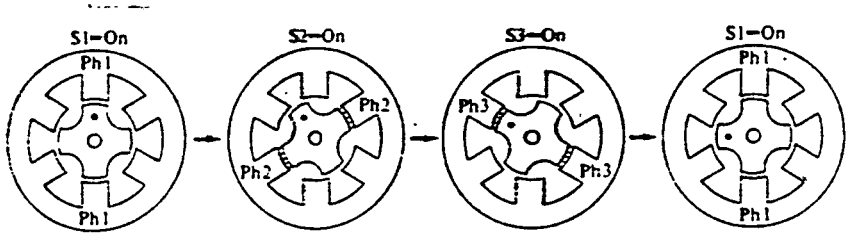
Clock state	R	1	2	3	4	5	6	7	8
Phase 1									
Phase 2									
Phase 3									
Phase 4									

ตารางที่ 2-2: ลำดับการกระตุ้นแบบสองเฟส

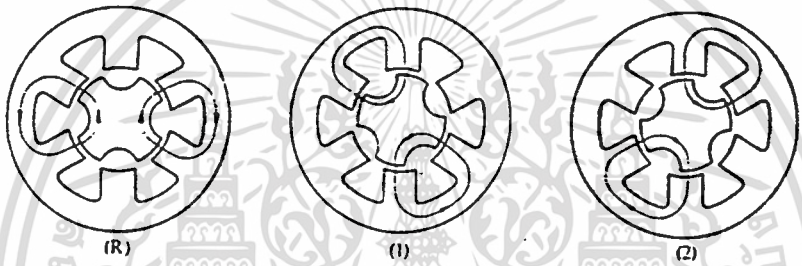
ความสัมพันธ์ของตำแหน่งระหว่างขั้วโรเตอร์และสเตเตอร์ ได้แสดงไว้ดังรูปที่ 2-5 ขั้วของสเตเตอร์และโรเตอร์จะไม่อยู่ในแนวเส้นตรงเดียวกันอย่างในกรณีของการกระตุ้นแบบเฟสเดียว การกระจายของสนามแม่เหล็กในการกระตุ้นแบบสองเฟสในสเต็ปมอเตอร์แบบสามเฟส แสดงได้ดังรูป 2-6



รูปที่ 2-5(1): ความสัมพันธ์ของตำแหน่งโรเตอร์และสเตเตอร์ในการกระตุ้นแบบสองเฟส

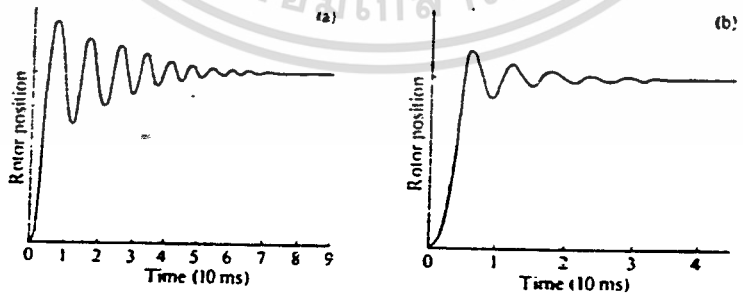


รูปที่ 2-5(2): แสดงความสัมพันธ์ของตำแหน่งโรเตอร์และสเตเตอร์ในการกระตุ้นแบบเฟสเดียว



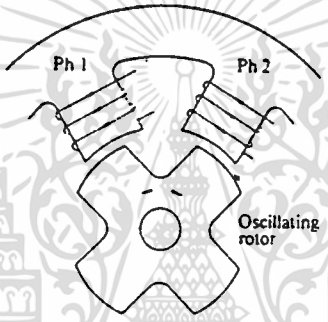
รูปที่ 2-6: การเปลี่ยนแปลงรูปแบบของสนามไฟฟ้าในการกระตุ้นแบบสองเฟส

สิ่งที่เปลี่ยนแปลงไปอย่างมากระหว่างการกระตุ้นแบบเฟสเดียวและสองเฟสนั้น ก็คือการตอบสนองทรานเซียนท์(Transient response) ซึ่งแสดงไว้ในรูปที่ 2-7

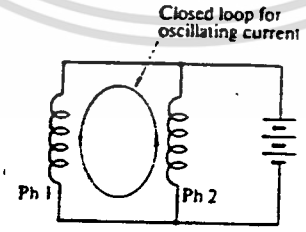


รูปที่ 2-7: ความแตกต่างของการตอบสนองแบบทรานเซียนท์

ในการกระตุ้นแบบสองเฟส การออสซิลเลตจะลดลงได้รวดเร็วกว่าในกรณีกระตุ้นแบบเฟสเดียว สามารถอธิบายให้เห็นได้ชัดเจนโดยใช้รูปที่ 2-8 และรูปที่ 2-9 ซึ่งจะเห็นได้ว่าวงจรในลักษณะการกระตุ้นแบบสองเฟสนั้น จะทำให้เกิดลูปของกระแส (Current Loop) ที่เกิดจากการเหนี่ยวนำทางแม่เหล็กไฟฟ้า อันเนื่องมาจากการออสซิลเลตหรือการสั่นของโรเตอร์ และเนื่องจากไม่มีเส้นทางให้กระแสไหลครบรอบในการกระตุ้นแบบเฟสเดียว ทำให้การออสซิลเลตค่อยๆลดลงได้ด้วยสาเหตุเพียงประการเดียว คือแรงเสียดทานทางกล



รูปที่ 2-8: การออสซิลเลตของโรเตอร์ในการกระตุ้นแบบสองเฟส



รูปที่ 2-9: ลูปกระแสที่เกิดขึ้นเมื่อเกิดการออสซิลเลตในการกระตุ้นแบบสองเฟส

2.1.3.3 การกระตุ้นแบบครึ่งสเต็ป(Half step mode)

การกระตุ้นแบบนี้เป็นการรวมการกระตุ้นแบบเฟสเดียวและแบบสองเฟสเข้าด้วยกัน ลำดับในการกระตุ้นของสเต็ปมอเตอร์แบบสามเฟสแสดงไว้ดังตารางที่ 2-3 ในที่นี้จำนวนของสัญญาณนาฬิกาจะถูกแบ่งออกเป็นสองลักษณะคือ แบบ A และแบบ B ในแบบ A การเข้าตำแหน่งจะกระทำในลักษณะของการกระตุ้นแบบเฟสเดียวเท่านั้น การกระตุ้นแบบสองเฟสจะเกิดขึ้นเมื่อมีการเลื่อนตำแหน่งจากจุดสมดุลย์จุดหนึ่งไปยังอีกจุดหนึ่งเท่านั้น การกระตุ้นแบบสองเฟสมีไว้เพื่อช่วยลดการออสซิลเลตที่เกิดขึ้น ส่วนในแบบ B จะมีการเข้าตำแหน่งทั้งในการกระตุ้นแบบเฟสเดียวและสองเฟส ดังนั้นสัญญาณนาฬิกาจะเป็นแบบ B ซึ่งจะมีผลช่วยลดมุมสเต็ปลงครึ่งหนึ่งด้วย

Clock state (A)	R	1		2		3		4		5	
Clock state (B)	R	1	2	3	4	5	6	7	8	9	
Phase 1	▨	▨				▨	▨	▨			
Phase 2		▨	▨	▨				▨	▨	▨	
Phase 3				▨	▨	▨				▨	▨

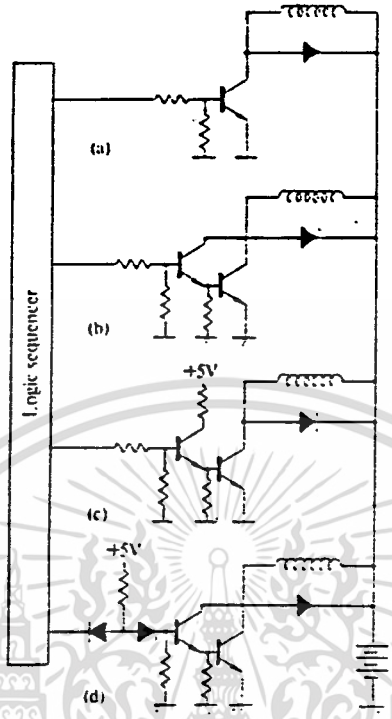
ตารางที่ 2-3: ลำดับการกระตุ้นแบบครึ่งสเต็ป

2.1.4 วงจรขับสำหรับสเต็ปมอเตอร์(Stepping motor drivers)

ปัญหาที่สำคัญที่สุดประการหนึ่งในการใช้งานสเต็ปมอเตอร์ก็คือวงจรขับสเต็ปมอเตอร์ ซึ่งแบ่งออกได้เป็นแบบลูปปิดและแบบลูปเปิดดังที่ได้กล่าวมาบ้างแล้ว ซึ่งต่อไปนี้จะได้กล่าวถึงวงจรขับมอเตอร์แบบลูปเปิด ระบบที่ใช้ในการควบคุมสเต็ปมอเตอร์นั้นนอกจากจะประกอบด้วยวงจรขับมอเตอร์แล้ว ยังประกอบด้วยวงจรเรียงลำดับลอจิกอีกด้วย ซึ่งได้แสดงไว้ในรูปที่ 2-3 แต่ในที่นี้เราจะไม่กล่าวถึงวงจรเรียงลำดับลอจิก เนื่องจากในโครงงานนี้จะใช้ไมโครโปรเซสเซอร์เป็นตัวเรียงลำดับลอจิกโดยการเขียนโปรแกรม

2.1.4.1 การเชื่อมต่อระหว่างวงจรเรียงลำดับลอจิกและวงจรขับ

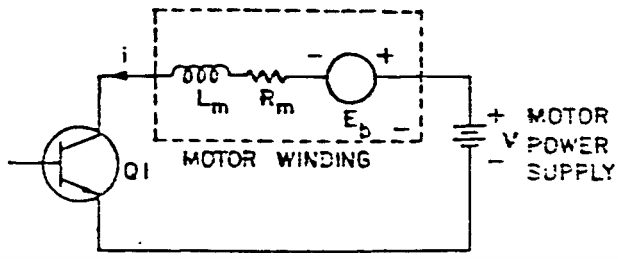
สัญญาณเอาท์พุทจากวงจรเรียงลำดับลอจิกจะถูกส่งเข้าไปยังอินพุทของวงจรขับมอเตอร์ วิธีการที่ง่ายที่สุดในการต่อวงจรทั้งสองเข้าด้วยกัน ก็คือการเชื่อมต่อกันโดยตรงดังแสดงในรูปที่ 2-10(a) และ (b) ถ้ากระแสเอาท์พุทของวงจรเรียงลำดับลอจิกมีค่าไม่พอที่จะจ่ายให้แก่อินพุทของวงจรขับ ก็จำเป็นที่จะต้องต่อบัฟเฟอร์(Buffer)คั่นระหว่างวงจรทั้งสอง ดังแสดงในรูปที่ 2-10 (c) และ (d)



รูปที่ 2-10: การเชื่อมต่อระหว่างวงจรเรียงลำดับลอจิกและวงจรขับ

2.1.4.2 สิ่งที่ควรพิจารณาในการออกแบบวงจรขับ

ในการออกแบบวงจรขับสเต็ปมอเตอร์ ผู้ออกแบบควรคำนึงถึงข้อจำกัดของสเต็ปมอเตอร์ ทรานซิสเตอร์กำลัง(Power transistor) และอุปกรณ์ทางอิเล็กทรอนิคส์ที่เกี่ยวข้อง คุณสมบัติของสเต็ปมอเตอร์แต่ละตัวสามารถเปลี่ยนแปลงไปได้ตามการผลิตและการใช้งาน ขดลวดของสเต็ปมอเตอร์นั้นเป็นโหลดประเภทตัวเหนี่ยวนำ(Inductive load) นอกจากนี้ขณะที่มอเตอร์หมุนก็จะสร้างแรงดันย้อนกลับ(Back emf,E_b)ซึ่งมีทิศตรงกันข้ามกับแหล่งจ่ายไฟของตัวสเต็ปมอเตอร์ เขียนเป็นวงจรสมมูลย์ได้ดังรูปที่ 2-11 และโดยทั่วไปแล้วค่าความต้านทานของขดลวด(R_m) จะมีค่าผิดพลาดไปได้ถึง 10 เปอร์เซ็นต์ เนื่องจากสเต็ปมอเตอร์ทั่วไปถูกออกแบบมาให้สามารถทำงานได้ในขณะที่อุณหภูมิตัวถังมีค่าสูง(ประมาณ 100 องศาเซลเซียส) และขณะที่มอเตอร์ร้อนมากขึ้น ค่าความต้านทานของขดลวดจะสูงขึ้น ซึ่งอาจเพิ่มขึ้นถึง 25 เปอร์เซ็นต์ในขณะที่มอเตอร์ทำงานที่อุณหภูมิปกติ นอกจากนี้สเต็ปมอเตอร์ส่วนใหญ่ถูกออกแบบมาให้ทำงานภายใต้ภาวะอิ่มตัว โดยเฉพาะอย่างยิ่งเมื่อความเร็วในการหมุนต่ำ ดังนั้นค่าความเหนี่ยวนำของขดลวดจะเปลี่ยนไปตามระดับกระแสดีซีในขดลวด นอกจากนี้ในสเต็ปมอเตอร์แบบค่ารีลักแตนซ์เปลี่ยนแปลงได้ ค่าความเหนี่ยวนำนี้ยังเปลี่ยนแปลงไปตามตำแหน่งของโรเตอร์อีกด้วย



รูปที่ 2-11: วงจรสมมูลย์ของขดลวดในสเต็ปมอเตอร์

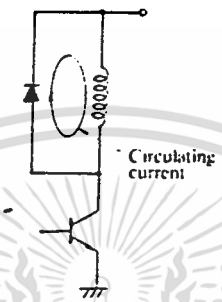
2.1.4.3 วงจรซัพเพรสเซอร์(Suppressor circuits)

ปัญหาสำคัญในการควบคุมสเต็ปมอเตอร์นั้น ได้แก่การเพิ่มขึ้นและการลดลงของกระแสในแต่ละเฟสเมื่อมีการเปิดและปิดสวิทช์ เมื่อความเร็วในการสวิทชิ่งเพิ่มขึ้นเช่นในการเร่งความเร็วของสเต็ปมอเตอร์ หรือในการหมุนด้วยความเร็วสูง มักจะพบว่ากระแสยังไม่ทันเพิ่มขึ้นจนถึงค่าที่ต้องจ่ายแก่ขดลวดของมอเตอร์ สวิทช์ก็ถูกปิดเสียก่อน ซึ่งเป็นผลมาจากค่าความเหนี่ยวนำในขดลวด และในตอนปิดสวิทช์กระแสก็ไม่ได้ลดลงในทันที ในขณะที่ขดลวดเฟสอื่นถูกกระตุ้นแล้ว กระแสในเฟสเดิมก็ยังลดไม่หมดทำให้เกิดปรากฏการณ์แดรกกิ้ง(Dragging effect)ในการหมุนมอเตอร์

พิจารณาจากรูปที่ 2-11 เมื่อขดลวดของมอเตอร์ได้รับการกระตุ้น กระแสในขดลวดจะมีค่า V/R_m เมื่อไม่คิดแรงดันคร่อมทรานซิสเตอร์ Q_1 เมื่อ Q_1 ไม่นำกระแส สนามแม่เหล็กตกค้างในขดลวดจะทำให้มีกระแสไฟฟ้าไหลไปในทิศทางเดิม แต่เนื่องจากอิมพีแดนซ์(Impedance)ของวงจรในเวลานี้เพิ่มขึ้นอย่างมากมายจนถือได้ว่าเป็นอนันต์ ดังนั้น V_{Q1} จะเพิ่มขึ้นเป็น $-L_m di/dt$ ทันที แรงดันนี้อาจมีผลให้เกิดเบรคดาวน์(Breakdown)ในทรานซิสเตอร์ได้ ในทางปฏิบัติแรงดันนี้อาจมีค่าสูงมากเป็น 50 ถึง 100 เท่าของแรงดันจากแหล่งจ่ายไฟของมอเตอร์ ดังนั้นจึงต้องใช้วงจรซัพเพรสเซอร์เข้าแก้ปัญหาซึ่งมีได้หลายวิธีดังนี้

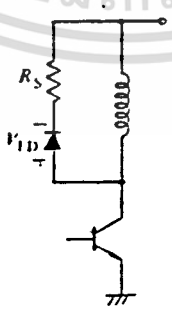
1.แบบใช้ไดโอด(Diode suppressor) จะมีไดโอดต่อขนานกันกับขดลวดของมอเตอร์ดังรูปที่ 2-12 เมื่อทรานซิสเตอร์ไม่นำกระแส จะเกิดกระแสหมุนวน(Circulating current)ขึ้น ดังรูปและจะค่อยๆลดลง ด้วยค่าคงที่เวลา(Time constant) L_m/R_m (สมมุติว่าไม่คิดค่าความต้านทานของไดโอดเมื่อถูกไบอัสตรง(Forward bias)ซึ่งมีค่าน้อยมาก) ดังนั้นในการใช้งานที่ความเร็วต่ำ การลดลงของกระแสอย่างช้าๆนี้ ยังคงอยู่ในช่วงที่พอยอมรับได้

ในทางปฏิบัติการลดลงของกระแสนั้นในขณะที่ไม่นำกระแส จะทำให้เกิดการสั่น (Damping effect)ของแกนโรเตอร์ได้ ทั้งนี้เนื่องจากกระแสหมุนวนทำให้เกิดแรงบิดตกค้างในขดลวดและจะเห็นได้ว่าแรงดันที่ขาคอลเล็กเตอร์(Collector)ของทรานซิสเตอร์ขณะไม่นำกระแสจะมีค่าเท่ากับ แรงดันของแหล่งจ่ายไฟรวมกับแรงดันคร่อมไดโอดขณะถูกไบอัสตรง



รูปที่ 2-12: วงจรไดโอดซัพเพรสเซอร์

2.<u>แบบใช้ไดโอดและความต้านทาน</u>(Diode-resistor suppressor) ในกรณีที่ต้องการให้สเต็ปมอเตอร์หมุนเร็วขึ้น พลังงานที่สะสมในขดลวดของมอเตอร์ตอนไม่นำกระแสควรจะต้องลดลงให้เร็วที่สุดเท่าที่จะทำได้ ซึ่งสามารถทำได้โดยการเพิ่ม R_s อนุกรมเข้ากับไดโอดเพื่อลดค่าคงที่เวลาลงให้มีค่า $L_m/(R_m+R_s)$ ดังรูปที่ 2-13 ค่า R_s ที่มีค่ามากจะทำให้กระแสหมุนวนลดลงได้รวดเร็วขึ้น แต่ก็จะทำให้แรงดันที่ขาคอลเล็กเตอร์ของทรานซิสเตอร์เพิ่มสูงขึ้นด้วย นั่นคือจะต้องใช้ทรานซิสเตอร์ที่ทนแรงดันได้สูงขึ้น



รูปที่ 2-13: วงจรซัพเพรสเซอร์แบบใช้ไดโอดและความต้านทาน

แรงดันที่ขาคอลเล็กเตอร์ของทรานซิสเตอร์(V_{CE})ขณะหยุดนำกระแส สามารถหาได้จาก

$$V_{CE} = E + IR_s + V_{DF}$$

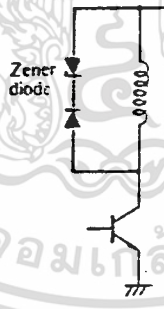
โดยที่ E คือ แรงดันที่จ่ายให้แก่สเต็ปมอเตอร์

I คือ กระแสที่ไหลเมื่อทรานซิสเตอร์หยุดนำกระแส

R_s คือ ความต้านทานชัพเพรสเซอร์

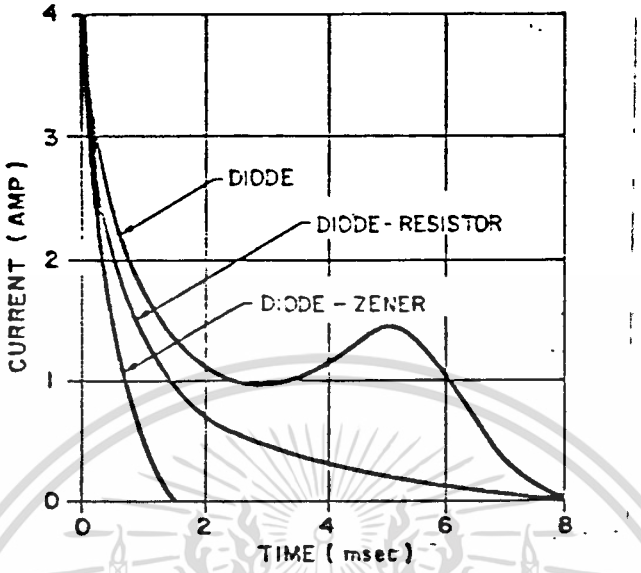
V_{DF} คือ แรงดันคร่อมไดโอดขณะไบอัสตรง

3.แบบใช้ซีเนอร์ไดโอด(Zener diode suppressor) หากว่าต้องการให้กระแสลดลงได้เร็วขึ้นอีก สามารถทำได้โดยการต่อซีเนอร์ไดโอดดังแสดงในรูปที่ 2-14 แรงดันขณะหยุดนำกระแสจะเพิ่มขึ้นจนกระทั่งถึงจุดเบรคดาวน์ของซีเนอร์ไดโอด ทำให้ซีเนอร์ไดโอดนำกระแส วิธีนี้จะทำให้แรงดันที่ขาคอลเล็กเตอร์ของทรานซิสเตอร์มีค่าคงที่ไม่เปลี่ยนแปลงตามกระแส (มีค่าเท่ากับแรงดันแหล่งจ่ายไฟของสเต็ปมอเตอร์รวมกับแรงดันซีเนอร์) ซึ่งจะช่วยให้พิจารณาหาทรานซิสเตอร์ที่เหมาะสมได้ง่ายกว่า



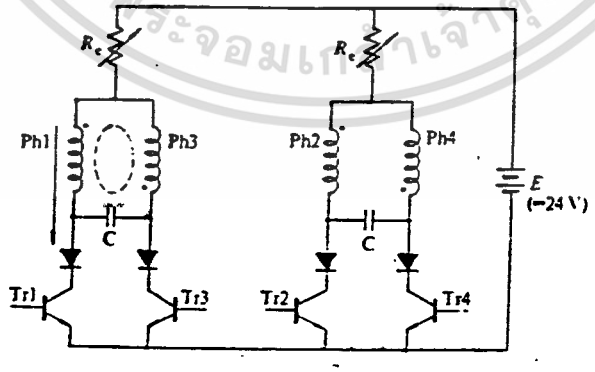
รูปที่ 2-14: วงจรซีเนอร์ไดโอดชัพเพรสเซอร์

การลดลงของกระแสขณะที่ทรานซิสเตอร์หยุดนำกระแส ของวงจรชัพเพรสเซอร์ทั้ง 3 แบบสามารถเปรียบเทียบกันได้ ดังแสดงในรูปที่ 2-15



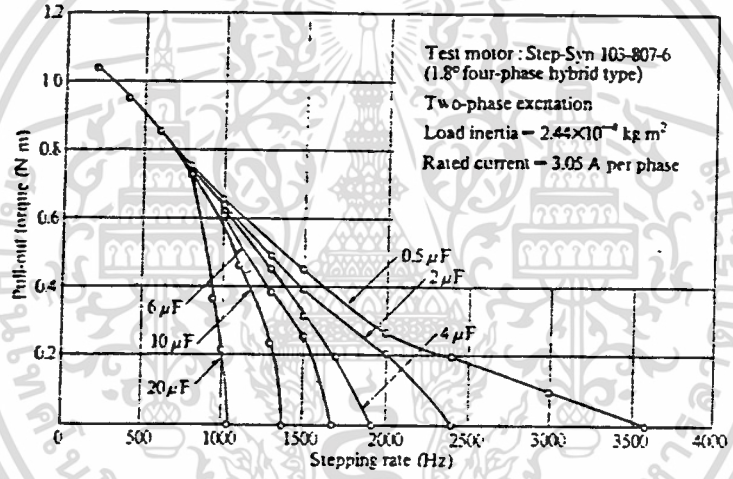
รูปที่ 2-15: เปรียบเทียบกระแสดอนหยุดนำกระแสของการใช้ซัพเพรสเซอร์แบบต่างๆ

4. แบบใช้ตัวเก็บประจุ(Condenser suppressor) วงจรดังรูปที่ 2-16 จะใช้กับสเต็ปมอเตอร์ที่พันลวดแบบไบไฟล่าร์[1](Bifilar wound)และเป็นสเต็ปมอเตอร์แบบสี่เฟส ตัวเก็บประจุจะต่อระหว่างเฟสที่ 1 กับเฟสที่ 3 และระหว่างเฟสที่ 2 กับเฟสที่ 4 ตัวเก็บประจุที่ใช้ มีหน้าที่สองประการ คือ



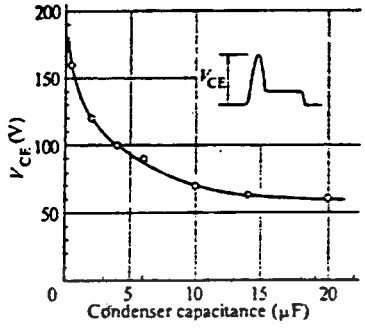
รูปที่ 2-16: วงจรขับที่ใช้ตัวเก็บประจุสำหรับมอเตอร์สี่เฟสแบบไบไฟล่าร์
ตัวต้านทาน R_c จะใช้ปรับค่ากระแสให้ได้ค่ากระแสที่ต้องการ

ประการแรก เมื่อทรานซิสเตอร์หยุดนำกระแสตัวเก็บประจุที่ต่ออยู่จะเป็นทางผ่านของกระแสที่ลดลงของขดลวดเพื่อป้องกันทรานซิสเตอร์เสียหาย พิจารณาในกรณีที่ Tr1 หยุดนำกระแสเมื่อมีการกระตุ้นแบบเฟสเดียว จะต้องมี Tr2 หรือ Tr4 ตัวใดตัวหนึ่งที่นำกระแสอยู่ แต่ Tr3 จะหยุดนำกระแส เนื่องจากเฟสที่ 1 และเฟสที่ 3 ถูกพันลวดแบบไบไฟล่าร์ จะมีกระแสทรานเซียนท์หมุนวนดังแสดงด้วยเส้นประในรูปที่ 2-16 ถ้า Tr3 นำกระแสตอนที่กระแสทรานเซียนท์ดังกล่าวเป็นศูนย์และประจุที่ถูกเก็บไว้ในตัวเก็บประจุมีค่าสูงสุด กระแสที่เป็นบวกก็สามารถที่จะไหลผ่านขดลวดเฟสที่ 1 ไปได้ วิธีการนี้สามารถใช้ได้กับการกระตุ้นแบบสองเฟสด้วย อย่างไรก็ดีวิธีนี้เหมาะสมกับการขับสเต็ปมอเตอร์ที่มีความเร็วในการหมุนเปลี่ยนแปลงอยู่ในช่วงแคบๆเท่านั้น



รูปที่ 2-17: ความสัมพันธ์ระหว่างแรงบิดตอนเริ่มหมุนกับความถี่เมื่อใช้ตัวเก็บประจุค่าต่างๆ

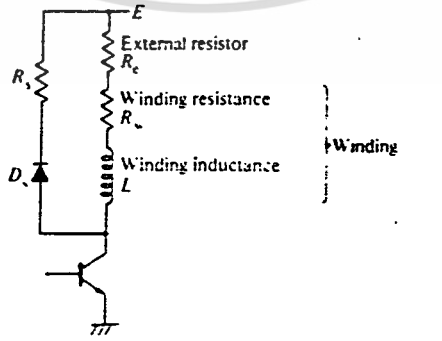
หน้าที่อีกประการหนึ่งของการใช้ตัวเก็บประจุก็คือ จะช่วยให้แรงบิดตอนเริ่มหมุนมีค่าสูงขึ้น ดังรูปที่ 2-17 ค่าความจุที่ต่ำจะทำให้แรงบิดนี้มีค่าสูงขึ้นที่ความเร็วสูง ซึ่งเป็นผลมากจากการลดลงของกระแสที่รวดเร็วหลังจากทรานซิสเตอร์หยุดนำกระแส แต่แรงดันที่ขาคอลเล็กเตอร์ของทรานซิสเตอร์จะมีค่าสูงขึ้นด้วย ดังรูปที่ 2-18



รูปที่ 2-18:ความสัมพันธ์ระหว่างแรงดันคอลเล็กเตอร์กับค่าตัวเก็บประจุที่แรงบิดเริ่มต้นสูงสุด

2.1.4.4 การขับมอเตอร์แบบโอเวอร์ไดร์ฟ(Overdriving methods for fast current buildup)

เมื่อทรานซิสเตอร์นำกระแสเพื่อกระตุ้นเฟสใดเฟสหนึ่ง แรงดันจากแหล่งจ่ายไฟจะต้องเอาชนะผลของค่าความเหนี่ยวนำของขดลวดก่อนที่จะสามารถจ่ายกระแสให้ถึงค่าที่ต้องการได้ ทั้งนี้เพราะความเหนี่ยวนำจะต่อต้านกับการเพิ่มขึ้นของกระแส ในขณะที่ความถี่ในการสวิทช์มีค่าสูงขึ้นเวลาที่จะใช้เพิ่มกระแสจะมีค่าไม่พอ และทำให้แรงบิดลดลง มีผลให้ความเร็วของมอเตอร์ลดต่ำลง การแก้ไขสามารถทำได้หลายวิธี แต่จะกล่าวเพียงวิธีเดียวคือการเพิ่มตัวต้านทานอนุกรม(Series resistance) ส่วนวิธีอื่นๆ มีความยุ่งยากในการสร้างและอธิบายจึงไม่กล่าวในที่นี้ (หาอ่านเพิ่มได้ใน [1] และ [2]) เช่น การขับมอเตอร์โดยใช้แรงดันสองค่า(Dual voltage control) แบบใช้พัลส์วิทธ์มอดดูเลชั่น (Pulse Width Modulation,Chopped-voltage control)

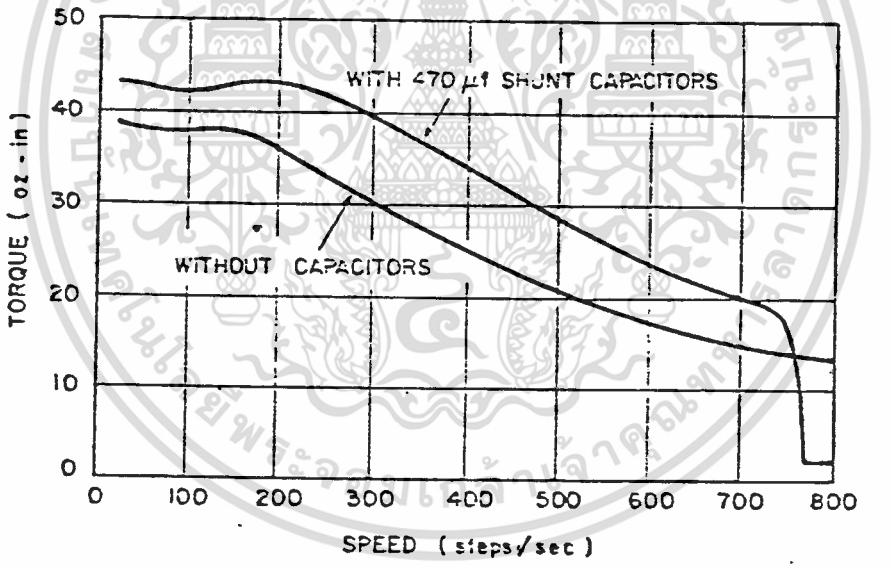


รูปที่ 2-19: แสดงการเพิ่ม R_e อนุกรมกับขดลวดของสเต็ปมอเตอร์

วิธีการนี้เป็นวิธีที่สะดวกและประหยัดที่สุด โดยการต่อตัวต้านทานอนุกรมเข้าไปกับขดลวดดังแสดงในรูปที่ 2-19 การคำนวณค่าแรงดันของแหล่งจ่ายไฟ(E) จะต้องพิจารณาค่ากระแสที่ต้องจ่ายผ่านขดลวดภายใต้สภาวะคงที่(Steady state condition) ค่าคงที่เวลาจะมีค่าลดลงจากเดิม L/R_w เป็น $L/(R_s + R_w)$

ข้อเสียของวิธีนี้คือจะเกิดการสูญเสียพลังงานมากในตัวต้านทาน R_s ถ้าความต้านทานของขดลวดแต่ละขดของสเต็ปมอเตอร์ชนิด 4 เฟสมีค่า 1.5 โอห์ม กระแสที่ต้องจ่ายแก่ขดลวดแต่ละขดมีค่า 4 แอมแปร์ และต้องการแหล่งจ่ายไฟขนาด 24 โวลท์ ดังนั้นค่าความต้านทานที่ต้องต่ออนุกรมเข้าไปจะมีค่า 4.5 โอห์ม ในแต่ละเฟส ดังนั้นพลังงานที่สูญเสียไปมีค่า 4.5×4^2 ซึ่งเท่ากับ 72 วัตต์ เมื่อกระตุ้นแบบเฟสเดียว และจะเพิ่มเป็น 2 เท่าเมื่อกระตุ้นแบบสองเฟส

นอกจากนี้ถ้าเราเอาตัวเก็บประจุมาต่อขนานกับตัวต้านทาน R_s จะทำให้อิมพีแดนซ์ลดลงในช่วงการดอบสนองแบบทรานเซียนท์ ซึ่งจะมีผลให้แรงบิดตอนเริ่มต้นดีขึ้น ดังแสดงในรูปที่ 2-20



รูปที่ 2-20: เปรียบเทียบแรงบิดตอนเริ่มต้นเมื่อเพิ่มตัวเก็บประจุขนานกับ R_s

2.2 การจัดการข้อมูลภาพ

การจัดการข้อมูลที่เกี่ยวข้องกับโครงงานนี้ได้แก่ การจัดการข้อมูลภาพ ซึ่งใช้หลักการพื้นฐานในเรื่องเกี่ยวกับคอมพิวเตอร์กราฟิก(Computer Graphics) จึงได้นำหลักการพื้นฐานของคอมพิวเตอร์กราฟิกมากล่าวไว้โดยสังเขป ณ ที่นี้ด้วย

2.2.1 พื้นฐานคอมพิวเตอร์กราฟิก

ในเรื่องของคอมพิวเตอร์กราฟิก คำว่า "ภาพ" หรือ "อิมเมจ(Image)" จะประกอบด้วย

-องค์ประกอบพื้นฐานของภาพ(Image Primitive) ได้แก่ เวคเตอร์(Vectors) และจุดภาพ(Pixels)

-คุณสมบัติของแต่ละองค์ประกอบ(Attribute) เช่น คุณสมบัติของเวคเตอร์ ได้แก่ ความกว้างของเส้น สี ความเข้ม ส่วนคุณสมบัติของจุดภาพ ได้แก่ สี ความเข้ม เป็นต้น

เซ็ตขององค์ประกอบพื้นฐานจะประกอบกันเป็นภาพ โดยที่แต่ละองค์ประกอบจะมีคุณสมบัติเฉพาะตัว ดังนั้น ภาพที่สร้างขึ้นโดยคอมพิวเตอร์จะสามารถแบ่งออกได้ตามชนิดขององค์ประกอบพื้นฐานของภาพนั้น ภาพที่สร้างขึ้นจากเซ็ตของเวคเตอร์มักจะถูกนำไปใช้ในงานคอมพิวเตอร์กราฟิก 3 มิติ ซึ่งจำเป็นต้องใช้เครื่องคอมพิวเตอร์ที่มีความเร็ว และมีความสามารถในทางคณิตศาสตร์สูงกว่าภาพที่สร้างขึ้นจากเซ็ตของจุดภาพมาก ในที่นี้จะกล่าวถึงภาพที่สร้างขึ้นจากเซ็ตของจุดภาพเท่านั้น

ภาพที่สร้างขึ้นโดยอาศัยเซ็ตของจุดภาพ หมายถึง ฟังก์ชันของความเข้มแสง f(x,y) โดยที่ x และ y เป็นพิกัดย่อยของแกนตั้งและแกนนอน ค่าของฟังก์ชันที่ตำแหน่ง x และ y ใดๆจะแสดงถึงคุณสมบัติของจุดภาพที่แทนตำแหน่งนั้นๆ ซึ่งในที่นี้จะหมายถึงสีเท่านั้น

ภาพที่คอมพิวเตอร์จะสามารถเข้าใจได้นั้น จะต้องอยู่ในรูปของข้อมูลดิจิตอลเท่านั้น กล่าวคือ ค่า x ค่า y และค่า f(x,y) ต่างก็จะต้องมีค่าเป็นเลขจำนวนเต็ม เพื่อให้คอมพิวเตอร์สามารถเข้าใจได้โดยสะดวก โดยภาพที่มีลักษณะนี้เรียกว่า ดิจิตอลอิมเมจ(Digital Image) โดยที่สามารถพิจารณาได้ว่า ภาพแต่ละภาพคือเมตริกซ์(Matrix) ซึ่งมีตัวแปร x แทนคอลัมน์ และตัวแปร y แทนแถว เพื่อใช้ในการระบุตำแหน่งของจุดภาพ โดยที่สมาชิกแต่ละตัวของเมตริกซ์ก็คือจุดภาพแต่ละจุด ค่าของสมาชิกแต่ละตัวในเมตริกซ์จะเป็นค่าที่แสดงคุณสมบัติ(Attribute) ของจุดภาพนั้นๆ

ในกรณีของภาพสีเดียว(Monochrome image) ค่าของฟังก์ชันที่ตำแหน่ง x และ y (สมาชิกของเมตริกซ์ที่คอลัมน์ x และแถว y) จะแทนค่าความสว่างของจุดภาพ ซึ่งค่าความสว่างนี้เรียกว่า ระดับความเข้ม(Gray scale)

ในภาพขาวดำความสว่างของแต่ละจุดภาพจะแปรเปลี่ยนไปเป็นขั้น ๆ ตั้งแต่ดำสนิทไปจนถึงขาว ค่าระดับความเข้มนี้ถ้ามีความแปรผันไปมาก ก็จะทำให้ภาพที่ได้จากคอมพิวเตอร์มีคุณภาพลดขึ้นตามไปด้วย เช่น ภาพที่มีระดับความเข้ม 4 ระดับ จะมีคุณภาพดีกว่าภาพที่มีระดับความเข้มเพียง 2 ระดับ

นอกจากระดับความเข้มแล้ว ปัจจัยอีกประการหนึ่งที่เป็นตัวกำหนดคุณภาพของภาพ ก็คือ ความละเอียดภาพ(Resolution) ซึ่งหมายถึง จำนวนจุดภาพต่อหนึ่งหน่วยความยาว ภาพที่มีความละเอียดสูงจะมีความคมชัดมากกว่าภาพที่มีความละเอียดต่ำ

2.2.2 <u>การเข้ารหัสข้อมูลภาพ</u>

จากหัวข้อ 2.2.1 จะเห็นได้ว่า คุณภาพของภาพสีเดียวขึ้นอยู่กับปัจจัย 2 ประการคือ

-ความละเอียดภาพ หรือจำนวนจุดภาพต่อหนึ่งหน่วยความยาว

-ค่าระดับความเข้มสีของจุดภาพ

อย่างไรก็ตาม คุณภาพของภาพจะถูกจำกัดด้วยขนาดของหน่วยความจำในเครื่องคอมพิวเตอร์ เนื่องจากภาพที่มีคุณภาพสูง (มีรายละเอียดมากและมีความคมชัดสูง) จะต้องใช้หน่วยความจำต่อหนึ่งหน่วยพื้นที่เป็นจำนวนมาก

พิจารณาความสัมพันธ์ของหน่วยความจำกับค่าความละเอียดภาพและระดับความเข้ม ค่าความละเอียดภาพ จะกำหนดจำนวนจุดภาพต่อหนึ่งหน่วยพื้นที่ เช่น ภาพขนาด 8 นิ้ว x 11 นิ้ว มีความละเอียด 150 จุดต่อนิ้ว จะประกอบด้วยจุดภาพจำนวน

$$8 \times 11 \times 150 \times 150 = 1,980,000 \text{ จุดภาพ}$$

นั่นคือจะต้องใช้เมตริกซ์ที่มีสมาชิก 1,980,000 ตัว เพื่อแทนภาพนี้ และใช้เนื้อที่หน่วยความจำเพื่อเก็บตัวเลขเป็นจำนวน 1,980,000 จำนวนด้วย ตัวเลขแต่ละจำนวนแสดงคุณสมบัติประจำจุดภาพแต่ละจุด ขนาดของตัวเลขที่ใช้แสดงคุณสมบัตินี้ขึ้นอยู่กับ ระดับความเข้มของจุดภาพ(ในกรณีของภาพสี ขนาดของตัวเลขขึ้นกับระดับความเข้มของจุดภาพ และจำนวนสีที่แสดงได้)

ถ้าจำนวนบิทที่ต้องใช้ในการเก็บตัวเลข 1 จำนวน มีค่าเท่ากับ n บิทแล้ว ค่าตัวเลขนั้นจะมีค่าที่แตกต่างกันได้ทั้งหมด 2^n ค่า ดังนั้น ภาพที่มีระดับความเข้มหลายระดับก็จะใช้เนื้อที่สำหรับแต่ละจุดภาพมากกว่าภาพที่มีระดับความเข้มน้อยกว่า เช่น ภาพที่มีระดับความเข้ม 256 ระดับ จะใช้เนื้อที่ 8 บิท หรือ 1 ไบท์ ต่อการเก็บจุดภาพ 1 จุด แต่ภาพที่มีระดับความเข้ม 4 ระดับจะใช้เพียง 2 บิท ต่อ 1 จุดภาพเท่านั้น

พิจารณาภาพขนาด 8 นิ้ว x 11 นิ้ว ที่มีความละเอียด 150 จุดต่อนิ้ว และมีระดับความเข้ม 2 ระดับ คือ ขาวกับดำ จะใช้หน่วยความจำ 1 บิทต่อ 1 จุดภาพ ดังนั้นหน่วยความจำทั้งหมดที่ต้องใช้ในการเก็บภาพจะมีขนาด 1,980,000 บิท หรือคิดเป็น 247,000 ไบท์(ประมาณ 241 กิโลไบท์) สำหรับเครื่องคอมพิวเตอร์ส่วนบุคคล(Personal computer)ที่ใช้งานกันทั่วไป หน่วยความจำขนาดนี้ถือว่ามีขนาดใหญ่มาก ดังนั้นจึงได้มีการคิดค้นวิธีการเข้ารหัสข้อมูลภาพ(Encoding)ขึ้น เพื่อให้สามารถลดขนาดข้อมูลภาพลงได้โดยยังคงคุณภาพของภาพไว้

วิธีการเข้ารหัสข้อมูลภาพมีหลายวิธี ซึ่งแต่ละวิธีต่างก็มีรูปแบบ(Format) การจัดเก็บข้อมูลภาพที่แตกต่างกัน ทำให้ไฟล์(File)ที่ใช้ในการเก็บข้อมูลภาพมีรูปแบบแตกต่างกันไปด้วย

ก่อนที่จะกล่าวถึงรูปแบบและวิธีการในการเข้ารหัสข้อมูลภาพที่ใช้ในโครงงานนี้ จะกล่าวถึงรูปแบบการเข้ารหัสที่เป็นมาตรฐานบางรูปแบบก่อน เนื่องจากมีหลักการที่คล้ายคลึงกันกับรูปแบบที่ใช้ในโครงงานนี้ ซึ่งก็คือ หลักการในการเข้ารหัสที่เรียกว่า RLE(Run Lenght Encoding) การเข้ารหัสแบบนี้จะช่วยให้ใช้หน่วยความจำน้อยลง ในการอธิบายหลักการทำงานของ RLE จะกำหนดให้ภาพที่จะเก็บเป็นภาพขาวดำที่มีระดับความเข้ม 2 ระดับ ดังนั้น ข้อมูลแต่ละบิทจะแทนจุดภาพแต่ละจุด ข้อมูล 1 ไบท์จึงแทนจุดภาพ 8 จุด การทำงานของ RLE เริ่มจากการพิจารณาจุดภาพที่ตำแหน่งบนซ้ายของภาพ ไล่มาตามแนวนอนจนถึงมุมบนขวาของภาพ แล้วกลับไปเริ่มที่ด้านซ้ายใหม่ในแถวต่อไป โดยที่หากจุดภาพใดเป็นสีขาว บิทที่แทนจุดภาพนั้นก็จะมีค่าเป็น 1 ถ้าจุดภาพเป็นสีดำ บิทที่แทนจุดภาพนั้นจะมีค่าเป็น 0 จะทำให้ข้อมูลแต่ละไบท์แทนจุดภาพ 8 จุดที่เรียงต่อกัน เมื่อพิจารณา 8 จุดภาพหรือไบท์ต่อไปจะมีการค่านำมาเปรียบเทียบกับไบท์ที่ผ่านมาด้วย ซึ่งมีรายละเอียดดังนี้

หากค่าในไบท์ที่ n ใดๆ มีค่าไม่เหมือนกับไบท์ที่ n-1 และถ้าค่า 2 บิทนัยสำคัญสูงสุดไม่เท่ากับ 11_2 ข้อมูลไบท์นั้นจะถูกเก็บลงไฟล์ทันที แต่หากมีค่าเหมือนกันจะทำการบวกค่าตัวนับ(Counter)ด้วย 1 ค่าในตัวนับจะบอกว่า ค่าไบท์หลังสุดมีค่าเหมือนกันกี่ไบท์แล้ว แต่ค่าในตัวนับนี้จะมีค่าได้ไม่เกิน 63 ถ้ามีจำนวนไบท์ที่เหมือนกันเกิน 63 ไบท์ ให้เก็บค่า FF_{16} และตามด้วยค่าไบท์นั้นลงไฟล์ จากนั้นจึงเริ่มนับ 1 ใหม่ ถ้าในกรณีของไบท์เดียวที่มีค่าสองบิทบนเป็น 11_2 ให้นำค่าตัวนับ คือ 1 ไปรวมกับค่า $C0_{16}$ เป็น $C1_{16}$ แล้วเขียนลงไฟล์ ตามด้วยค่าของไบท์นั้น จากวิธีการดังกล่าวจะเห็นว่ามีการเขียนข้อมูลลงไฟล์ก็ต่อเมื่อมีข้อมูลที่ไม่เหมือนกันหรือค่าในตัวนับมีค่าเกิน 63 เท่านั้น จึงสามารถลดจำนวนหน่วยความจำที่ต้องใช้ลงได้

พิจารณาการเข้ารหัสข้อมูลภาพที่ใช้ในโครงงานนี้ เป็นวิธีที่ดัดแปลงมาจากเทคนิค RLE เพื่อให้เหมาะสมกับโครงงานนี้ โดยที่ทางคณะผู้จัดทำโครงงานเป็นผู้กำหนดขึ้นเอง และเรียกการเข้ารหัสข้อมูลภาพแบบนี้ว่า EDE(Edge Detection Encoding) มีส่วนที่แตกต่างจาก RLE คือ EDE จะพิจารณาเปรียบเทียบข้อมูลทีละ 1 จุดภาพ ในขณะที่ RLE พิจารณาเปรียบเทียบข้อมูลทีละ 8 จุดภาพ อย่างไรก็ตามแม้ว่า EDE จะเปรียบเทียบข้อมูลทีละ 1 จุดภาพ แต่เวลาเก็บข้อมูลลงไฟล์ก็จะต้องเก็บเป็นไบท์เช่นกัน ข้อมูลแต่ละไบท์ของ EDE จะมีความหมายในตัวเองอย่างชัดเจนซึ่งต่างจาก RLE ที่ต้องนำข้อมูลหลายไบท์มาประกอบกัน

ข้อมูลแต่ละไบท์ของ EDE จะแบ่งออกเป็น 2 ส่วนคือ ส่วนที่เป็นบิทบอกสถานะ ได้แก่บิทที่ 7 (บิทนัยสำคัญสูงสุด) ส่วนบิทที่เหลือตั้งแต่บิทที่ 0 ถึงบิทที่ 6 จะทำหน้าที่เป็นตัวนับ บิทบอกสถานะจะเป็นตัวบอกสีของกลุ่มจุดภาพที่อยู่ติดกัน เป็นจำนวนเท่ากับค่าของตัวนับ ถ้าบิทนี้เป็น 1 แสดงว่าเป็นจุดภาพสีขาว และถ้าเป็น 0 จะแสดงว่าเป็นจุดภาพสีดำ บิทที่เหลือที่ประกอบขึ้นมาเป็นตัวนับจะเป็นตัวบอกว่ามีจุดภาพที่สีเหมือนกันอยู่เรียงติดกันทั้งหมดกี่จุดภาพ เช่น

1 0 0 1 0 1 1 0 แสดงว่ามีจุดภาพสีขาวติดกัน 38 จุดภาพ

0 1 1 1 1 1 1 1 แสดงว่ามีจุดภาพสีดำติดกัน 127 จุดภาพ

จากการจัดโครงสร้างลักษณะนี้ จะเห็นว่า หากมีจุดภาพที่มีสีเหมือนกันอยู่ติดกันไม่เกิน 127 จุดภาพ จุดภาพทั้งหมดจะถูกแทนด้วยข้อมูลเพียง ไบท์เดียว แต่ในกรณีที่มีจุดภาพสีเดียวกันติดต่อกันมากกว่า 127 จุดภาพ ก็จะต้องใช้ข้อมูลมากกว่า 1 ไบท์

รูปแบบของไฟล์ข้อมูลที่ใช้เทคนิคการเข้ารหัสแบบ RLE และ EDE จะกล่าวถึงรายละเอียดอีกครั้งในบทที่เกี่ยวกับโปรแกรมใช้งาน

2.3 การติดต่อข้อมูลระหว่างคอมพิวเตอร์

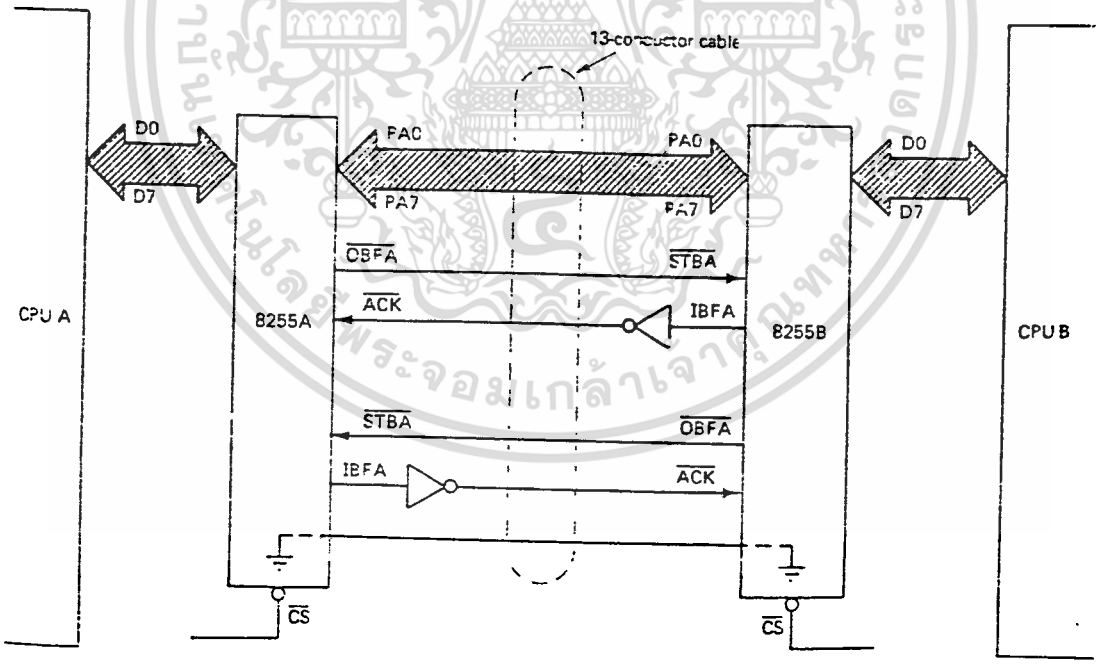
การติดต่อข้อมูลเป็นการโอนย้ายข้อมูลระหว่างอุปกรณ์ต่างๆของซีพียู(CPU) ซึ่งจะต้องทำงานให้สอดคล้องกันกับอุปกรณ์อินพุท และเอาท์พุทต่างๆ เพื่อเพิ่มประสิทธิภาพในการทำงานให้สมบูรณ์ยิ่งขึ้น

ข้อมูลที่จะโอนย้ายจะต้องมีแหล่งส่งข้อมูล และแหล่งรับข้อมูลเสมอ ซึ่งในขบวนการต่างๆของการทำงานร่วมกันของแหล่งรับส่งข้อมูลกับซีพียู จะต้องพิจารณาว่าข้อมูลนั้นเป็นแอดเดรส(Address) หรือดาต้า(Data) จะส่งไปยังจุดไหน และจะส่งเมื่อไร ขบวนการเหล่านี้จะต้องมีสัญญาณในการตรวจสอบความพร้อมของอุปกรณ์เสมอ เพื่อให้การรับส่งข้อมูลเป็นไปอย่างถูกต้อง

โดยทั่วไปหลักของการส่งข้อมูลในคอมพิวเตอร์หรือระหว่างคอมพิวเตอร์ จะมีการส่งข้อมูล 2 แบบ คือ การส่งแบบขนาน และการส่งแบบอนุกรม ในที่นี้จะได้กล่าวถึงเฉพาะการส่งข้อมูลแบบขนานระหว่างคอมพิวเตอร์ 2 ตัวโดยใช้พอร์ท 8255A ซึ่งใช้งานในโครงงานนี้ด้วย

ในที่นี้จะไม่กล่าวถึงรายละเอียดการใช้งาน 8255A ผู้ที่สนใจสามารถศึกษาเพิ่มเติมได้จากคู่มือการใช้งาน 8255A ทั่วไป

รูปที่ 2-21 แสดงตัวอย่างวิธีการแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง โดยการใช้ 8255A ที่ถูกโปรแกรมให้ทำงานในโหมด 2 เมื่อ CPU-A ส่งข้อมูลออกไปยัง 8255A จะมีสัญญาณ OBF ไปสโตรบ 8255A อีกตัวหนึ่งที่ต่อกับ CPU-B(ในที่นี้จะเรียกว่าเป็น 8255B) ทำให้ 8255B ส่งสัญญาณ IBF ซึ่งจะถูกกลับสถานะไปเป็นสัญญาณ ACK ในขณะเดียวกันก็จะอ่านข้อมูลเข้าไปยังบัสสองทิศทางของ CPU-B เมื่อ CPU-B อ่านข้อมูลเข้าไปแล้ว 8255B ก็จะหยุดส่ง IBF ทำให้ ACK มีค่าเป็น 1 ในขณะนั้นสัญญาณ OBF ก็จะเป็น 1 และการรับส่งข้อมูลก็จะสิ้นสุดลง การทำงานจะสลับกันเมื่อ CPU-B ต้องการส่งข้อมูล 1 ไบท์ไปยัง CPU-A



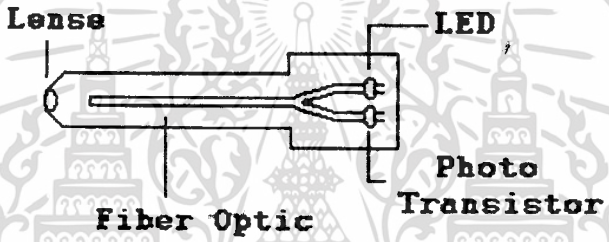
รูปที่ 2-21 แสดงการใช้ 8255A ในการเชื่อมต่อคอมพิวเตอร์สองตัวเพื่อสื่อสารข้อมูล

วิธีการรับส่งข้อมูลที่ใช้ในโครงงานนี้ จะมีลักษณะคล้ายคลึงกับวิธีข้างต้น ซึ่งจะได้กล่าวต่อไป

บทที่ 3
โครงสร้าง หลักการ และการทำงาน

3.1 หัวอ่านข้อมูล

ในโครงงานนี้ใช้หัวอ่านรหัสแถบเป็นอุปกรณ์ในการรับข้อมูล มีโครงสร้างดังรูปที่ 3-1 โดยมีโครงสร้างภายในประกอบไปด้วย แหล่งกำเนิดแสงและอุปกรณ์รับแสง แหล่งกำเนิดแสงเป็นไดโอดเปล่งแสง(Light Emitting Diode:LED) ส่วนอุปกรณ์รับแสงได้แก่ โฟโต้ทรานซิสเตอร์(Photo Transistor) โดยใช้เส้นใยนำแสง(Optic Fiber)เป็นอุปกรณ์นำแสง



รูปที่ 3-1: โครงสร้างของหัวอ่านรหัสแถบ

การทำงานของหัวอ่านรหัสแถบนี้จะใช้หลักการดังนี้ คือ สีของภาพที่แตกต่างกัน จะมีปริมาณการสะท้อนแสงไม่เท่ากัน โดยที่ไดโอดเปล่งแสงจะติดสว่างส่งแสงมาตามเส้นใยนำแสง ไปยังเลนส์เพื่อบังคับลำแสงให้มีมุมแคบลง เมื่อแสงไปตกกระทบภาพจะมีการสะท้อนแสงกลับมาผ่านเลนส์ เข้ามาตามเส้นใยนำแสงซึ่งต่ออยู่กับโฟโต้ทรานซิสเตอร์

ปริมาณของแสงที่สะท้อนกลับมายังโฟโต้ทรานซิสเตอร์ จะขึ้นอยู่กับปัจจัยต่างๆดังนี้

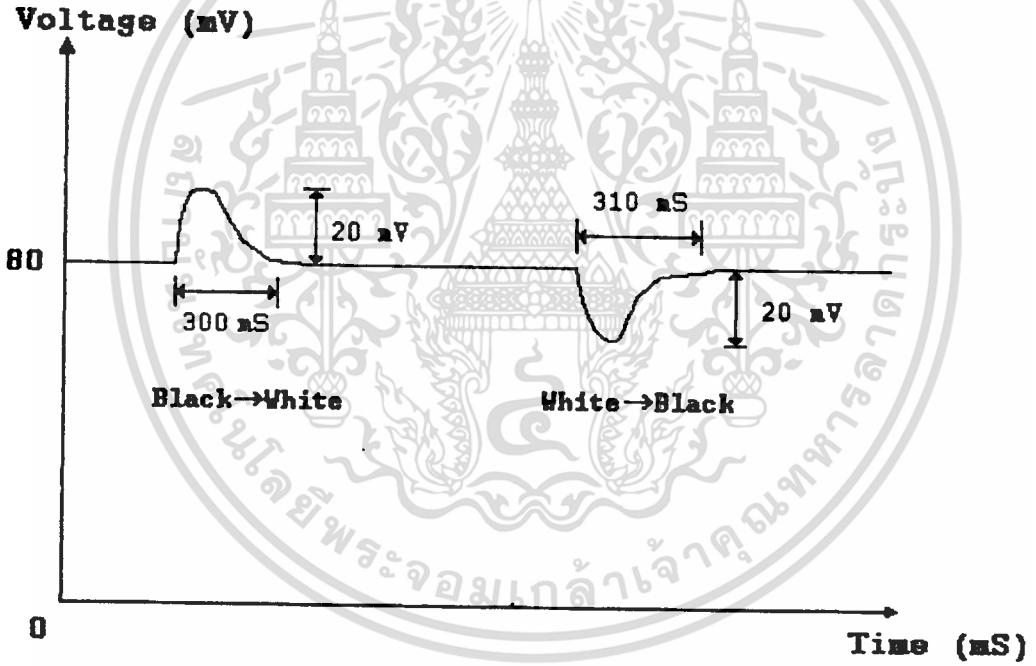
1.ความสว่างของแสงที่เกิดจากไดโอดเปล่งแสง

2.ระยะห่างระหว่างเลนส์กับภาพ

3.คุณสมบัติในการสะท้อนแสงของภาพที่ตำแหน่งของหัวอ่าน

ดังนั้น หากเราควบคุมความสว่างของแสงที่ได้จากไดโอดเปล่งแสงให้คงที่ และควบคุมระยะห่างระหว่างเลนส์กับภาพให้คงที่แล้ว ปริมาณของแสงที่สะท้อนกลับมายังโฟโต้ทรานซิสเตอร์จะขึ้นอยู่กับปัจจัยเพียงประการเดียว คือ คุณสมบัติในการสะท้อนแสงของภาพ ซึ่งก็คือสีของภาพ ภาพที่มีสีดำจะสะท้อนแสงกลับมาได้น้อยกว่าภาพที่มีสีขาว

โฟโต้ทรานซิสเตอร์จะทำหน้าที่ตรวจจับการเปลี่ยนแปลงของปริมาณแสงสะท้อน สัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์จะมีลักษณะดังรูปที่ 3-2 ในขณะที่ไม่มีการเปลี่ยนแปลงปริมาณของแสงสะท้อน สัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์จะมีลักษณะเป็นแรงดันไฟตรง มีขนาดประมาณ 80 มิลลิโวลท์ ซึ่งค่านี้จะขึ้นอยู่กับการไบอัส(Bias)ไดโอดเปล่งแสงและโฟโต้ทรานซิสเตอร์ เมื่อปริมาณแสงสะท้อนเปลี่ยนจากมากไปน้อย(หัวอ่านมีการเคลื่อนที่ผ่านภาพจากสีขาวไปเป็นสีดำ) สัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์จะลดลงประมาณ 20 มิลลิโวลท์เป็นเวลาประมาณ 300 มิลลิวินาที จึงกลับเข้าสู่ค่าเดิม เมื่อปริมาณแสงสะท้อนเปลี่ยนจากน้อยไปมาก(หัวอ่านมีการเคลื่อนที่ผ่านภาพจากสีดำไปเป็นสีขาว) สัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์จะเพิ่มขึ้นประมาณ 20 มิลลิโวลท์ เป็นเวลาประมาณ 300 มิลลิวินาที แล้วจึงกลับสู่ค่าเดิมเช่นกัน



รูปที่ 3-2: ลักษณะของสัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์

เพื่อความสะดวกในการเชื่อมต่อกับส่วนอื่นๆของระบบ สัญญาณอินพุทของระบบซึ่งก็คือสัญญาณที่จะได้จากส่วนหัวอ่านข้อมูลนี้ ควรจะมีลักษณะดังต่อไปนี้

1. เป็นสัญญาณดิจิตอลขนาด 0 โวลท์ หรือ 5 โวลท์

2. ระดับของสัญญาณที่ได้ จะต้องขึ้นอยู่กับสีของจุดภาพที่หัวอ่านข้อมูลกำลังสแกนอยู่เท่านั้น ไม่ขึ้นกับเวลา โดยกำหนดให้

-จุดภาพสีขาว สัญญาณอินพุทมีขนาด 5 โวลท์

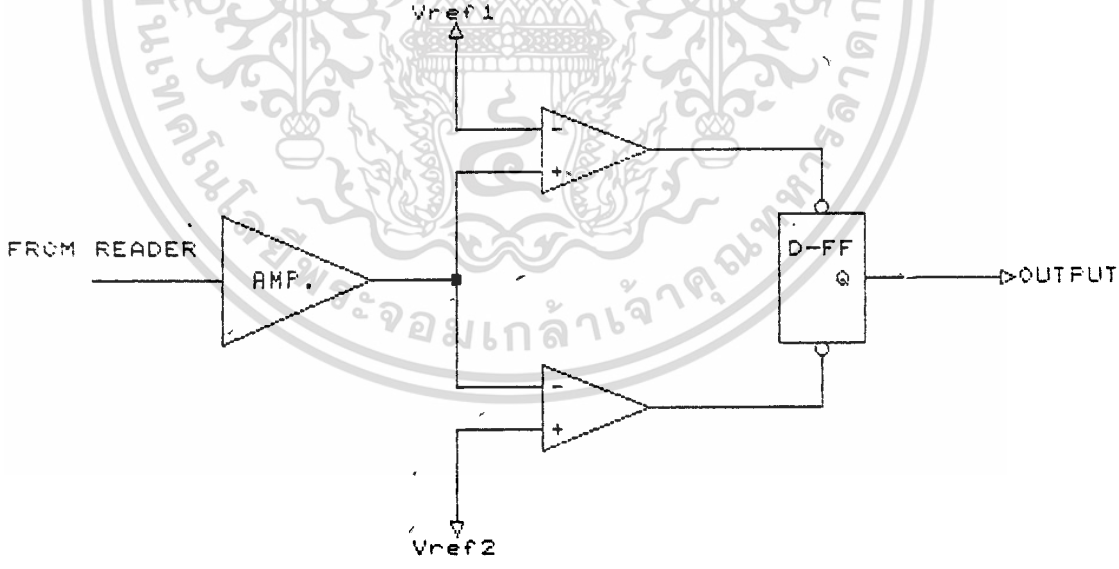
-จุดภาพสีดำ สัญญาณอินพุทมีขนาด 0 โวลท์

จากลักษณะของสัญญาณอินพุทที่กำหนดขึ้นทั้ง 2 ประการ จะเห็นได้ว่า สัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์ยังไม่เหมาะสมที่จะนำมาใช้เป็นสัญญาณอินพุทของระบบ จึงจำเป็นที่จะต้องมีการแปลงสัญญาณให้เหมาะสมเสียก่อนโดยใช้วงจรแปลงสัญญาณ

การออกแบบวงจรแปลงสัญญาณ ต้องพิจารณาลักษณะสัญญาณอินพุท เปรียบเทียบกับลักษณะสัญญาณเอาท์พุทที่ต้องการก่อน แล้วจึงกำหนดคุณสมบัติของวงจร ในกรณีของวงจรแปลงสัญญาณหัวอ่านข้อมูลที่ใช้ในโครงงานนี้ จะพบปัญหา 2 ประการคือ

1.ทิศทางการเปลี่ยนแปลงของสัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์ จะมีการเปลี่ยนแปลงไปในลักษณะตรงกันข้ามกับที่ต้องการ

2.ขนาดของสัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์เปลี่ยนแปลงไปตามเวลา โดยจะมีการเปลี่ยนแปลงก็ต่อเมื่อจุดภาพมีการเปลี่ยนสีเท่านั้น



รูปที่ 3-3: บล็อกไดอะแกรมของวงจรแปลงสัญญาณหัวอ่าน

จากปัญหาทั้ง 2 ประการนี้ จะกำหนดลักษณะของวงจรแปลงสัญญาณหัวอ่านข้อมูลได้ดังนี้

-ต้องมีวงจรขยายแบบกลับเฟส(Inverting Amplifier) เพื่อขยายและกลับเฟสของสัญญาณ

-ต้องมีส่วนวงจรเปรียบเทียบ(Comparator) เพื่อตรวจจับการเปลี่ยนแปลงของสัญญาณ

-ต้องมีวงจรแลทช์(Latch) เพื่อรักษาสถานะของสัญญาณไว้ไม่ให้เปลี่ยนแปลงตามเวลา

บล็อกไดอะแกรมของวงจรแปลงสัญญาณเป็นดังรูปที่ 3-3 ในหน้า 28

ในส่วนของวงจรเปรียบเทียบสัญญาณ จะประกอบไปด้วยออปแอมป์(OpAmp) 2 ตัวทำหน้าที่เป็นวงจรเปรียบเทียบสัญญาณ 2 วงจร คอยตรวจสอบระดับสัญญาณอินพุทว่า มีการเปลี่ยนแปลงในลักษณะที่เพิ่มขึ้นหรือลดลง ดังนั้นวงจรเปรียบเทียบทั้งสองวงจรนี้จะทำงานสลับกัน โดยวงจรเปรียบเทียบวงจรแรก จะคอยตรวจว่าหัวอ่านสัญญาณมีการเคลื่อนที่ผ่านจุดภาพสีดำไปเป็นจุดภาพสีขาวหรือไม่ โดยการพิจารณาสัญญาณว่าเพิ่มขึ้นจาก 80 มิลลิโวลท์หรือไม่ ส่วนวงจรเปรียบเทียบวงจรที่ 2 จะตรวจสอบการเคลื่อนที่จากจุดภาพสีขาวไปเป็นจุดภาพสีดำ โดยพิจารณาว่าสัญญาณลดลงจาก 80 มิลลิโวลท์หรือไม่ สัญญาณที่ออกจากวงจรเปรียบเทียบทั้งสองวงจรนี้ จะนำไปป้อนเข้ากับวงจรฟลิปฟลอป(Flip flop) เพื่อให้คงค่าสัญญาณเอาท์พุทไว้ จนกว่าจะมีการเปลี่ยนแปลงของสัญญาณครั้งต่อไป โดยจะนำสัญญาณทั้ง 2 นี้ป้อนให้กับขาพรีเซท(Preset) และขาเคลียร์(Clear)ของฟลิปฟลอปตามลำดับ วงจรสมบูรณ์ของวงจรแปลงสัญญาณหัวอ่านข้อมูลจะเป็นดังรูปที่ 3-4

3.1.1 การทำงานของวงจร

จากรูปที่ 3-4 U_{16} เป็นวงจรรักษาระดับแรงดัน(Regulator) ขนาด 2 โวลท์ ทำหน้าที่จ่ายแรงดันให้แก่ไดโอดเปล่งแสงและโฟโต้ทรานซิสเตอร์ โดยมี R_{18} และ R_{19} เป็นตัวจำกัดกระแส สัญญาณที่ได้จากโฟโต้ทรานซิสเตอร์จะถูกป้อนให้กับวงจรขยายสัญญาณ ซึ่งประกอบด้วย $U_{17A,B,C}$ ซึ่งเป็นออปแอมป์ 4 ตัวในตัวถังเดียวกัน(Quad OP AMP)เบอร์ LM324

U_{17C} ต่อเป็นวงจรบัฟเฟอร์(Buffer) โดยขาอินพุทไม่กลับเฟส(Non inverting input) หรือขาอินพุท(+) ของ U_{17C} จะต่อกับ U_{17B} ซึ่งทำหน้าที่ป้องกันการลดทอนของสัญญาณที่มาจากหัวอ่าน เนื่องจากวงจรนี้ใช้ไฟเลี้ยงเพียงชุดเดียว คือ ไฟบวกและกราวด์(Single Supply) ทำให้ขาอินพุท(+) ของ U_{17B} ต้องมีแรงดันค่าหนึ่งไบอัสไว้ประมาณ 0.95 โวลท์ และถ้าเราต้องการให้สัญญาณที่ขาอินพุท(+) ของวงจรมีการเปลี่ยนแปลงตามสัญญาณจากหัวอ่าน ก็จะต้องต่อขาอินพุท(+)ของ U_{17C} ไปยังขาอินพุท(+) ของ U_{17A} แต่ที่ขานี้จะมีการตั้งค่าแรงดันไว้แล้ว จึงทำให้สัญญาณที่จะเข้ามาแล้วผ่านออกไปจาก U_{17C} จะต้องมีแรงดันสูงกว่าค่าที่ตั้งไว้ ดังนั้นจึงได้ต่อวงจรของ U_{17B} เข้ามา

รูปที่ 3-4: วงจรสแกนเนอร์อ่านบาร์โค้ด

เพื่อทำให้ผลดังกล่าวหายไป แต่สัญญาณก็จะยังเปลี่ยนแปลงตามสัญญาณที่มาจากหัวอ่าน โดยการต่อคร่อมไว้ด้วย R_{21} แล้วต่อจากขาอินพุทของหัวอ่านไปยังขาอินพุทกลับเฟส(Inverting Input) หรือขาอินพุท(-) ของ U_{17B} ซึ่งต่อกับ C_3 ซึ่งมีค่า 0.068 uF(ค่านี้จะน้อยกว่านี้ได้แต่ต้องไม่ต่ำเกินไป เพราะทำให้รูปสัญญาณมีความชันมากเกินไป และค่านี้จะต้องไม่มากเกินกว่าค่านี้ เพราะจะทำให้เอาท์พุทเปลี่ยนแปลงตามสัญญาณอินพุทไม่ทัน) C_3 จะทำหน้าที่เก็บประจุหรือคายประจุสัญญาณที่ได้จากหัวอ่าน การเก็บประจุจะผ่านทางไดโอด D_{10} ส่วนการคายประจุจะผ่านทาง C_4 และ R_{20} เพราะที่จุดนี้จะมีแรงดันคร่อมต่ำกว่าเสมอ เมื่อเทียบกับการคายประจุผ่านทาง R_{21} ไปยังขาอินพุท(+)ของ U_{17B} ซึ่งที่จุดทั้งสองนี้จะมีแรงดันตกคร่อมใกล้เคียงกัน จะทำให้ขาอินพุท(+) ของ U_{17C} เสมือนถูกยกระดับแรงดัน ด้วยค่าแรงดันเท่ากับแรงดันที่ขาอินพุท(+)ของ U_{17A} จึงทำให้เอาท์พุทจาก U_{17A} เป็นสัญญาณที่เกิดจากการเปลี่ยนแปลงของสัญญาณของหัวอ่านที่เคลื่อนที่ผ่านจุดภาพ (ซึ่งถ้าเป็นวงจรแบบที่ไม่มี U_{17B} แล้วสัญญาณเอาท์พุทที่ออกมา จะเป็นสัญญาณที่มีแรงดันสูงกว่า 0.95 โวลท์เท่านั้น) โดยมี R_{26} เป็นตัวควบคุมอัตราการขยายสัญญาณ สัญญาณที่ออกจาก U_{17A} จะส่งต่อไปยังของวงจรเปรียบเทียบต่อไป

วงจรส่วนเปรียบเทียบสัญญาณประกอบด้วยออปแอมป์ 2 ตัว คือ U_{17D} และ U_{18A} ทำหน้าที่เป็นวงจรเปรียบเทียบ ซึ่งจะถูกกำหนดแรงดันอ้างอิงไว้ต่างกัน ไดโอด D_{11} และD_{12} ทำหน้าที่ป้องกันไม่ให้สัญญาณที่เข้ามายังขาอินพุทของออปแอมป์ทั้ง 2 มีค่าแตกต่างกันมากกว่า 0.6 โวลท์ พิจารณาในส่วนของ U_{17D} จะเห็นว่า มีการต่อวงจรให้เอาท์พุทมีค่าเป็น 0 โวลท์ เมื่อสัญญาณที่มาจากส่วนขยายเข้าที่ขาอินพุท(-) มีค่าสูงกว่าค่าแรงดันอ้างอิงที่กำหนดไว้ ซึ่งจะเกิดกรณีนี้ได้ก็ต่อเมื่อ หัวอ่านมีการเคลื่อนที่ผ่านจุดภาพสีขาวแล้วเปลี่ยนเป็นจุดภาพสีดำ สัญญาณเอาท์พุทที่ได้จะมีลักษณะเป็นพัลส์ซึ่งควบคุมความกว้างโดย C_5 0.068 uF และถูกส่งไปยังวงจรส่วนแลทช์ต่อไป

ส่วน U_{18A} จะเป็นการต่อแบบวงจรเปรียบเทียบพื้นฐาน โดยมี R_{27} และR_{28} เป็นตัวกำหนดแรงดันอ้างอิง เอาท์พุทของ U_{18A} จะเป็น 0 โวลท์ เมื่อสัญญาณจากวงจรขยายมีค่าต่ำกว่าแรงดันอ้างอิง ซึ่งจะเกิดขึ้นก็ต่อเมื่อหัวอ่านมีการเคลื่อนที่ผ่านจุดภาพสีดำ แล้วเปลี่ยนเป็นจุดภาพสีขาว สัญญาณเอาท์พุทจะถูกส่งไปยังวงจรแลทช์ต่อไป

วงจรแลทช์ในที่นี้ใช้ U_{19A} ซึ่งเป็นฟลิปฟลอปแบบ D โดยนำสัญญาณจาก U_{17D} มาป้อนเข้าที่ขาเคลียร์ซึ่งทำงานเมื่อได้รับลอจิกต่ำ เมื่อจุดภาพที่สแกนเปลี่ยนจากขาวเป็นดำ จะทำให้ฟลิปฟลอปถูกเคลียร์ และจะคงค่าต่อไปเรื่อยๆจนกว่าจะมีสัญญาณจาก U_{18A} มาป้อนเข้าที่ขาพรีเซท เนื่องจากจุดภาพที่สแกนเปลี่ยนจากดำเป็นขาว ทำให้เอาท์พุทของฟลิปฟลอปเป็นลอจิกสูง จนกว่าจะมีสัญญาณเคลียร์เข้ามาอีก ดังนั้น ลักษณะของสัญญาณที่ออกมาจากเอาท์พุทของฟลิปฟลอปจึงตรงกับลักษณะสัญญาณที่ต้องการ

3.2 มอเตอร์และวงจรขับ

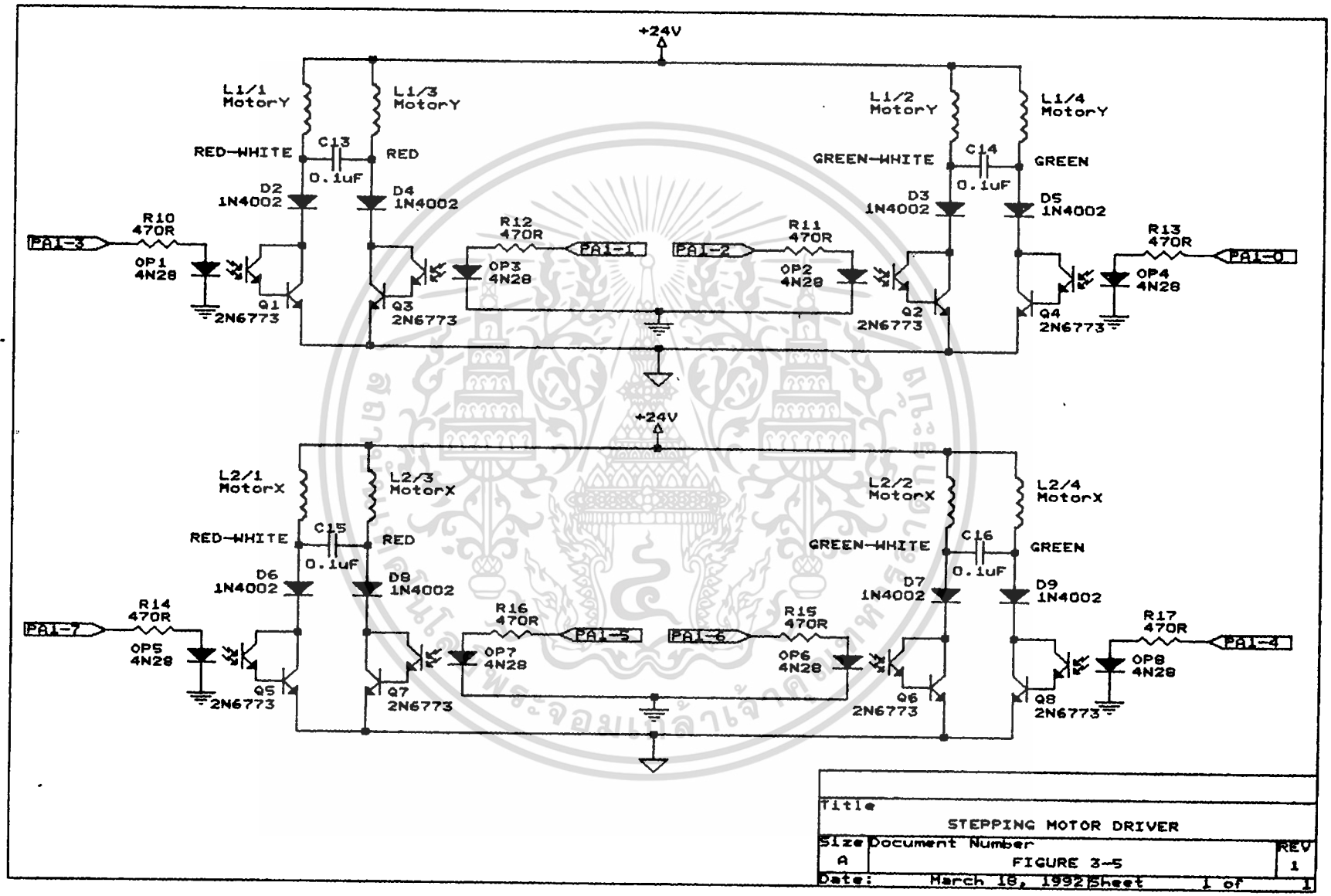
สเต็ปมอเตอร์เป็นส่วนประกอบสำคัญส่วนหนึ่งในโครงงานนี้ เนื่องจากมีหน้าที่เคลื่อนหัวอ่านข้อมูลไปยังตำแหน่งต่างๆ ของภาพที่ต้องการสแกน ซึ่งหลักการในการสแกนนั้นจะเลื่อนหัวอ่านไปตามแกน X และแกน Y คล้ายกับการทำงานของพล็อตเตอร์ ดังนั้นเราจึงต้องใช้สเต็ปมอเตอร์สองตัวเพื่อควบคุมการเลื่อนหัวอ่านทั้งในแกน X และแกน Y สำหรับสเต็ปมอเตอร์ที่ใช้ในโครงงานนี้เป็นแบบสี่เฟส ใช้แรงดัน 24 โวลท์ ต้องการกระแสในการขับมอเตอร์ 0.3 แอมแปร์ต่อเฟส และมีมุมสเต็ป 1.8 องศา

จากทฤษฎีของสเต็ปมอเตอร์และวงจรขับที่ได้กล่าวไว้ในบทที่ 2 นั้น ทำให้ทราบว่าการขับมอเตอร์ด้วยวิธีการกระตุ้นแบบสองเฟส มีข้อดีในเรื่องการเข้าสู่ตำแหน่งในสภาวะคงที่(Steady-State) จะเกิดขึ้นเร็วกว่าการกระตุ้นแบบเฟสเดียว ดังนั้นในโครงงานนี้จึงเลือกใช้การกระตุ้นแบบสองเฟสเพื่อให้ภาพที่สแกนได้มีความถูกต้องและชัดเจนมากยิ่งขึ้น

วงจรขับสเต็ปมอเตอร์ในโครงงานนี้ แสดงดังรูปที่ 3-5

3.2.1 การทำงานของวงจร

จากรูปที่ 3-5 ประกอบด้วยวงจรสองส่วนที่คล้ายคลึงกัน รูปบนเป็นวงจรขับสเต็ปมอเตอร์ในแกน Y และรูปล่างเป็นวงจรขับสเต็ปมอเตอร์แกน X ทางด้านอินพุทของวงจรขับมอเตอร์แกน Y จะต่อกับเอาท์พุทพอร์ท A ของ 8255A(U_4) ใน 4 บิทล่าง คือ PA_{1-3}-PA_{1-0} ซึ่งจะใช้ขับมอเตอร์เฟสที่ 1-4($L_{1/1}$-$L_{1/4}$) ตามลำดับ และ 4 บิทบน คือ PA_{1-7}-PA_{1-4} สำหรับมอเตอร์แกน X ในเฟสที่ 1-4($L_{2/1}$-$L_{2/4}$) ตามลำดับ สัญญาณเอาท์พุทที่ได้จากพอร์ท A ของ U_4 จะนำไปป้อนเข้าอินพุทของออปโต้คัปเปอร์(Opto-coupler) OP_1-OP_4 สำหรับมอเตอร์แกน Y และ OP_5-OP_8 สำหรับมอเตอร์แกน X เพื่อทำหน้าที่แยกกราวด์ของวงจรควบคุมและวงจรขับมอเตอร์ออกจากกัน ซึ่งจะช่วยป้องกันสัญญาณรบกวนและความเสียหายที่เกิดจากวงจรขับมอเตอร์ ซึ่งมีแรงดันและกระแสสูงกว่าทางด้านอินพุทที่ต่ออยู่กับไมโครโปรเซสเซอร์ ไปมีผลให้เกิดสัญญาณรบกวนหรือทำให้วงจรของบอร์ดควบคุมเสียหายไปด้วย ทางด้านเอาท์พุทของออปโต้คัปเปอร์จะต่อแบบดาร์ลิงตันกับทรานซิสเตอร์กำลังขนาด 400 โวลท์ 40 วัตต์ 1 แอมแปร์ เพื่อช่วยให้จ่ายกระแสให้มากขึ้น สำหรับรายละเอียดต่างๆเกี่ยวกับการทำงานของวงจรในรูปที่ 3-5 จะเหมือนกับที่ได้กล่าวไว้แล้วในบทที่ 2 รูปที่ 2-16 จึงไม่นำมากล่าวไว้ในที่นี้อีก



รูปที่ 3-5: วงจรขับมอเตอร์สเต็ป

3.3 บอร์ดควบคุม

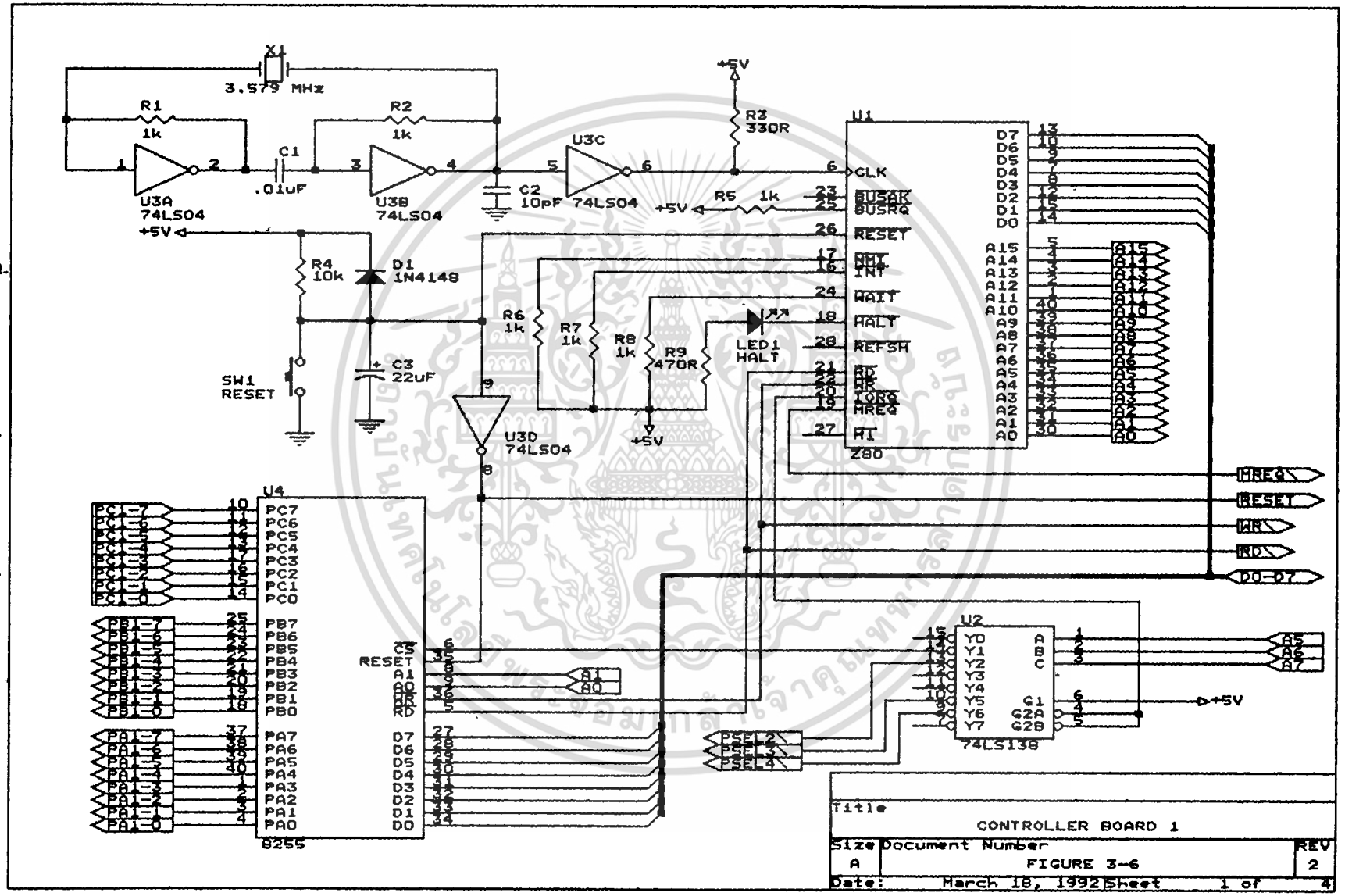
วงจรในส่วนของบอร์ดควบคุมประกอบด้วย ไมโครโปรเซสเซอร์เบอร์ Z-80(U_1) ซึ่งมีหน้าที่ดังต่อไปนี้

1. ควบคุมการอ่านข้อมูล และเข้ารหัสข้อมูล
2. เก็บข้อมูลลงในหน่วยความจำของบอร์ดควบคุม
3. ควบคุมการเลื่อนตำแหน่งของหัวอ่านข้อมูลโดยการขับสเต็ปมอเตอร์
4. ควบคุมการส่งข้อมูลไปยังไมโครคอมพิวเตอร์

นอกจากนี้ยังประกอบด้วยพอร์ทแบบโปรแกรมได้เบอร์ 8255A อีกสองตัว ตัวแรก(U_4)เป็นเอาท์พุทสำหรับส่งข้อมูลไปกระตุ้นสเต็ปมอเตอร์ ส่งเสียงเตือน และสัญญาณเตือนการทำงานต่างๆด้วยไดโอดเปล่งแสงซึ่งได้แก่ ไดโอดที่บอกว่าหน่วยความจำเต็ม(Memory Full)จะต้องส่งข้อมูลไปเครื่องไมโครคอมพิวเตอร์ก่อนที่จะทำการสแกนต่อไป และไดโอดที่ใช้บอกว่าข้อมูลพร้อมที่จะส่งแล้ว(Ready to send) นอกจากนี้ยังใช้เป็นอินพุทรับข้อมูลจากหัวอ่านข้อมูล สวิทช์ควบคุมต่างๆ ซึ่งประกอบด้วย SW_3 ใช้หาขอบเขตของภาพที่ต้องการสแกน SW_4 สั่งให้เริ่มทำการสแกนภาพ และ SW_5 สั่งให้เริ่มทำการส่งข้อมูลไปให้ไมโครคอมพิวเตอร์ นอกจากนี้ยังเป็นอินพุทสำหรับตรวจสอบตำแหน่งเริ่มต้นของหัวอ่านข้อมูลโดยการใช้ตัวตรวจจับซึ่งทำงานด้วยแสง(Optical detector) เพื่อกำหนดตำแหน่งเริ่มต้นการสแกนของหัวอ่านทุกครั้งก่อนที่จะเริ่มต้นการอ่านข้อมูล สำหรับพอร์ท 8255A อีกตัวหนึ่ง(U_{15}) ใช้สำหรับเป็นเอาท์พุทพอร์ทเพื่อส่งข้อมูลไปยังไมโครคอมพิวเตอร์

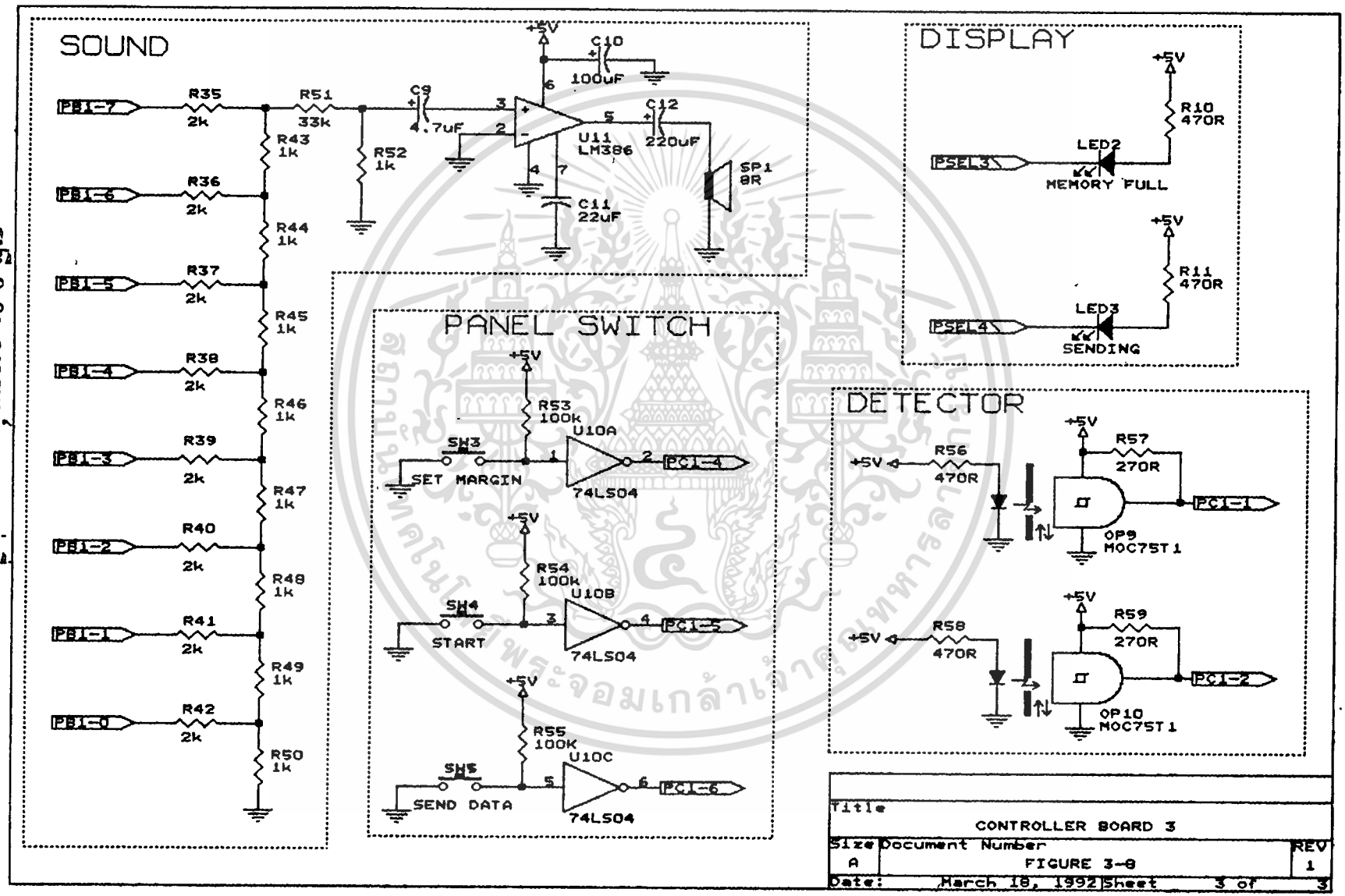
ส่วนประกอบต่อไปของบอร์ดควบคุม ได้แก่หน่วยความจำชั่วคราว(RAM)ขนาด 64 กิโลไบท์ คือ U_8 และ U_9 ซึ่งใช้เก็บข้อมูลภาพไว้ชั่วคราวก่อนที่จะทำการส่งข้อมูลไปยังไมโครคอมพิวเตอร์ต่อไป นอกจากนี้ยังมีหน่วยความจำถาวร(ROM)ขนาด 2 กิโลไบท์(U_7) ที่ใช้เก็บโปรแกรมมอนิเตอร์(Monitor program)สำหรับควบคุมการทำงานของบอร์ดควบคุม วงจรต่างๆทั้งหมดแสดงไว้ดังรูปที่ 3-6 จนถึงรูปที่ 3-8

สำหรับรูปของพอร์ท 8255A ตัวที่ 2 (U_{15}) นำไปแสดงรวมไว้กับวงจรส่วนติดต่อข้อมูล ในรูปที่ 3-9



รูปที่ 3-6: วงจรบอร์ดหลัก แผ่นที่ 1

รูปที่ 3-7: วงจรควบคุม แผ่นที่ 2



รูปที่ 3-8: แสดงวงจรของ Controller Board 3

3.4 ส่วนติดต่อข้อมูล

การติดต่อข้อมูลระหว่างเครื่องสแกนเนอร์กับคอมพิวเตอร์ จะกระทำผ่านอินเทอร์เฟสการ์ด (Interface Card)ซึ่งเสียบอยู่บนสล็อตของเครื่องคอมพิวเตอร์โดยมีวงจรดังรูปที่ 3-9 ประกอบไปด้วย 8255A(U_{14}) ซึ่งทำหน้าที่เป็นพอร์ทรับข้อมูลแบบขนาน และถูกโปรแกรมให้ทำงานในโหมด 0 โดยมี 74138(U_{13}) และ 74133(U_{12})ประกอบกันเป็นวงจรถอดรหัสแอดเดรสเพื่อสร้างสัญญาณเลือกพอร์ท สัญญาณที่ออกมาจาก U_{13} จะผ่านดิพสวิทช์(SW_6)เพื่อเลือกแอดเดรสที่จะนำมาเป็นพอร์ทอีกครั้งหนึ่ง ซึ่งทำให้หมายเลขพอร์ทที่จะใช้สามารถเปลี่ยนแปลงได้ตามการเลือกดิพสวิทช์ สำหรับโครงงานนี้กำหนดหมายเลขพอร์ทไว้ดังนี้

พอร์ท A หมายเลข $03EC_{16}$

พอร์ท B หมายเลข $03ED_{16}$

พอร์ท C หมายเลข $03EE_{16}$

พอร์ทควบคุม หมายเลข $03EF_{16}$

ให้การทำงานของ 8255A ทั้ง 2 ตัว บนบอร์ดควบคุม(U_{15}) และบนอินเทอร์เฟสการ์ด (U_{14}) ทำงานในโหมด 0 ซึ่งเป็นโหมดพื้นฐาน แต่เพื่อให้การรับส่งข้อมูลมีความถูกต้อง จึงได้เพิ่มสัญญาณแฮนด์เชค(Handshaking signals)ขึ้น เพื่อช่วยป้องกันปัญหาที่อาจเกิดขึ้น เนื่องจากความเร็วในการทำงานที่แตกต่างกันระหว่างไมโครคอมพิวเตอร์กับสแกนเนอร์ นอกจากนั้นยังได้สร้างข้อมูลซึ่งบอกสถานะของข้อมูลหลัก เพื่อให้คอมพิวเตอร์สามารถทราบได้ว่า ข้อมูลที่ได้รับมาเป็นข้อมูลส่วนใดของภาพ

ข้อมูลหลัก คือ ข้อมูลภาพที่ถูกเข้ารหัสแบบ EDE แล้ว จะถูกส่งออกมาทางพอร์ท A ของ U_{15} เข้ามายังพอร์ท A ของ U_{14} ข้อมูลบอกสถานะมี 3 บิท ถูกส่งมาทางพอร์ท C บน(PC_4-PC_7) ของ U_{15} เข้ามายังพอร์ท C บน(PC_4-PC_7)ของ U_{14} สัญญาณแฮนด์เชค จะถูกส่งมาทางบิท PC_3 ของ U_{14} เข้ามายังบิท PC_3 ของ U_{15} และจากบิท PC_7 ของ U_{15} เข้ามายังบิท PC_7 ของ U_{14}

จากที่กล่าวมา สามารถสรุปการกำหนดหน้าที่ของพอร์ทต่างๆ ได้ดังนี้

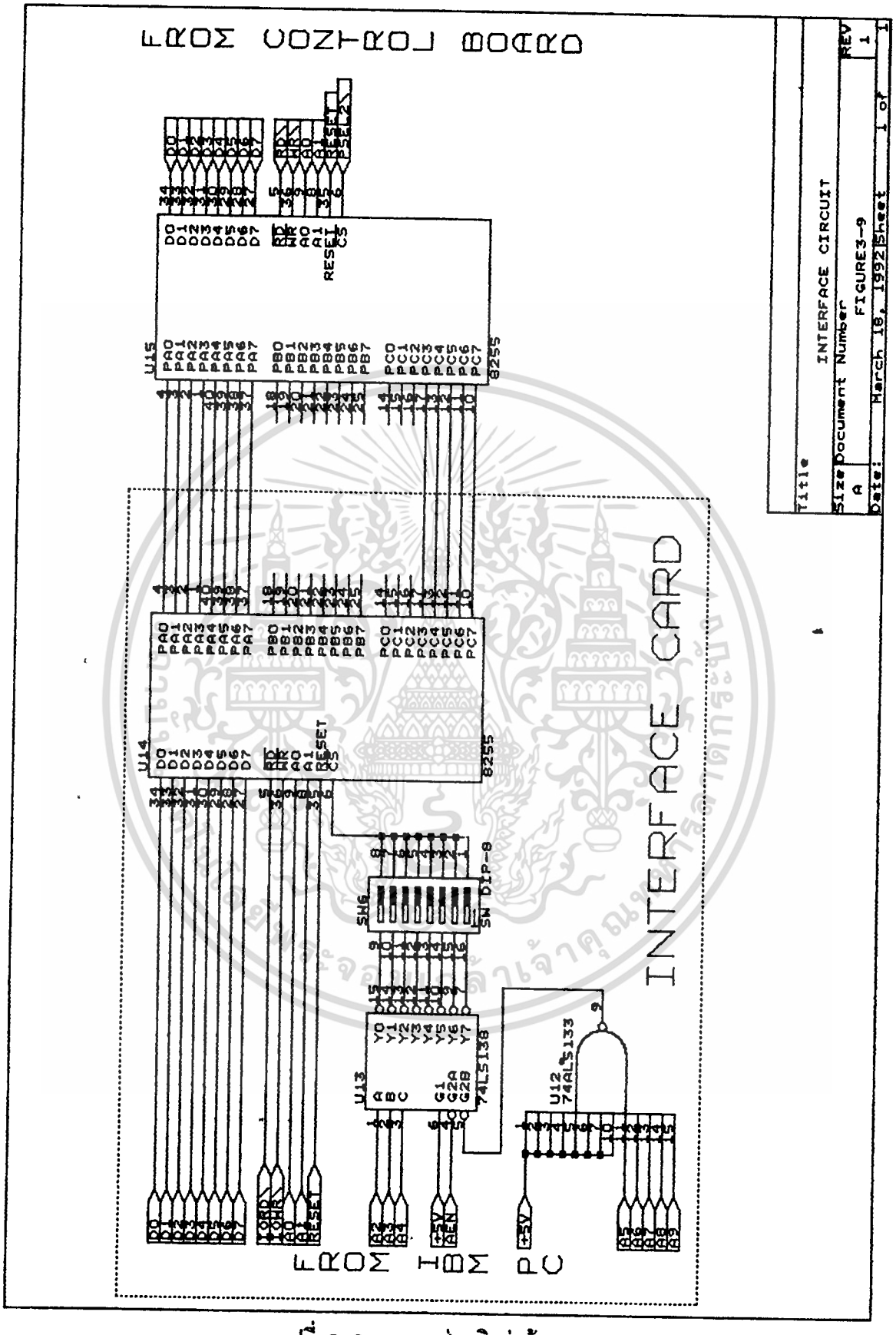
1.ด้านส่งข้อมูลหรือเครื่องสแกนเนอร์

-พอร์ท A ทำหน้าที่เป็นเอาท์พุท

-พอร์ท C บน ทำหน้าที่เป็นเอาท์พุท

-พอร์ท C ล่าง ทำหน้าที่เป็นอินพุท

ดังนั้น รหัสควบคุมการทำงานในโหมด 0 ของ U_{15} จะมีค่า 83_{16}



รูปที่ 3-9: วงจรส่วนติดต่อข้อมูล

2.ด้านรับข้อมูล หรืออินเทอร์เฟสการ์ด

-พอร์ท A ทำหน้าที่เป็นอินพุท

-พอร์ท C บน ทำหน้าที่เป็นอินพุท

-พอร์ท C ล่าง ทำหน้าที่เป็นเอาท์พุท

ดังนั้นจะได้รหัสควบคุมการทำงานเป็น 98_{16}

สัญญาณแฮนด์เชค จะประกอบไปด้วยสัญญาณต่างๆ ดังนี้

1.สัญญาณ READY เป็นสัญญาณซึ่งมีลอจิกต่ำ ที่ไมโครคอมพิวเตอร์ส่งมาเพื่อบอกสแกนเนอร์ว่าพร้อมที่จะรับข้อมูลแล้ว

2.สัญญาณ BUSY เป็นสัญญาณที่มีลอจิกสูง ที่ไมโครคอมพิวเตอร์ส่งให้แก่สแกนเนอร์ เพื่อบอกว่ายังไม่พร้อมที่จะรับข้อมูล ให้หยุดรอก่อน

3.สัญญาณ STROBE เป็นสัญญาณที่มีลอจิกต่ำ ที่สแกนเนอร์ส่งให้กับไมโครคอมพิวเตอร์เพื่อบอกให้ไมโครคอมพิวเตอร์รับข้อมูลได้

สัญญาณ READY และ สัญญาณ BUSY จะมีลอจิกตรงข้ามกันเสมอ จึงกำหนดให้อยู่บนบิทเดียวกันของสัญญาณแฮนด์เชคได้

กำหนดให้พอร์ท C แต่ละบิทมีความหมายดังนี้

PC0-PC2 : ไม่ใช้

PC3 : สัญญาณ READY และสัญญาณ BUSY

PC4-PC6 : ข้อมูลบอกสถานะ

PC7 : สัญญาณ STROBE

ความหมายของข้อมูลบอกสถานะเป็นดังตาราง 3-1

PC6	PC5	PC4	ความหมาย
1	0	0	เป็นข้อมูลไบท์แรกของภาพ
0	1	0	เป็นข้อมูลส่วนกลางของภาพ
0	0	1	เป็นข้อมูลไบท์สุดท้ายของภาพ
1	1	1	เป็นข้อมูลไบท์สุดท้ายของการส่งข้อมูล

ตารางที่ 3-1: ความหมายของข้อมูลบอกสถานะ

การรับส่งข้อมูลจะเกิดขึ้นเมื่อสแกนเนอร์อ่านข้อมูลจนเต็มหน่วยความจำ หรือจนกระทั่งข้อมูลได้ถูกอ่านหมดแล้ว โปรแกรมมอนิเตอร์จะสั่งให้ไดโอดเปล่งแสง "Ready to send" สว่าง เพื่อบอกให้ผู้ใช้ทราบว่าต้องกดสวิทช์ "SEND" เพื่อส่งข้อมูล ในขณะที่สแกนเนอร์รอการกดสวิทช์ "SEND" อยู่ ผู้ใช้จะต้องเตรียมโปรแกรมใช้งานบนไมโครคอมพิวเตอร์ เมื่อเตรียมโปรแกรมเรียบร้อยแล้วจึงจะสามารถกดสวิทช์ "SEND" ได้

เมื่อผู้ใช้กดสวิทช์ "SEND" แล้ว ขั้นตอนการรับส่งข้อมูลจะเป็นดังนี้

-โปรแกรมมอนิเตอร์จะตรวจสอบสัญญาณ READY จากไมโครคอมพิวเตอร์ แล้วจึงส่งข้อมูลภาพออกทางพอร์ท A

-หลังจากส่งข้อมูลภาพออกไปแล้ว โปรแกรมควบคุมจะรีเซท PC_7 ให้มีค่าเป็น 0 ซึ่งก็คือการส่งสัญญาณ STROBE ไปให้ไมโครคอมพิวเตอร์เพื่อบอกให้รับข้อมูลไปได้

-ไมโครคอมพิวเตอร์จะอ่านค่าสถานะของข้อมูลจากพอร์ท C เข้ามาเก็บไว้ แล้วตรวจสอบบิท PC_7 ว่ามีค่าเป็น 0 หรือไม่ หากมีค่าเป็น 0 แสดงว่ามีการส่งสัญญาณ STROBE มา ดังนั้นไมโครคอมพิวเตอร์จะอ่านข้อมูลจากพอร์ท A เข้ามา แล้วจึงเซทบิท PC_3 ให้เป็น 1 แล้วส่งออกไปทางพอร์ท C เพื่อเป็นสัญญาณ BUSY แล้วจึงนำข้อมูลที่อ่านได้ไปประมวลผลต่อไปตามแต่สถานะของข้อมูลนั้นๆ

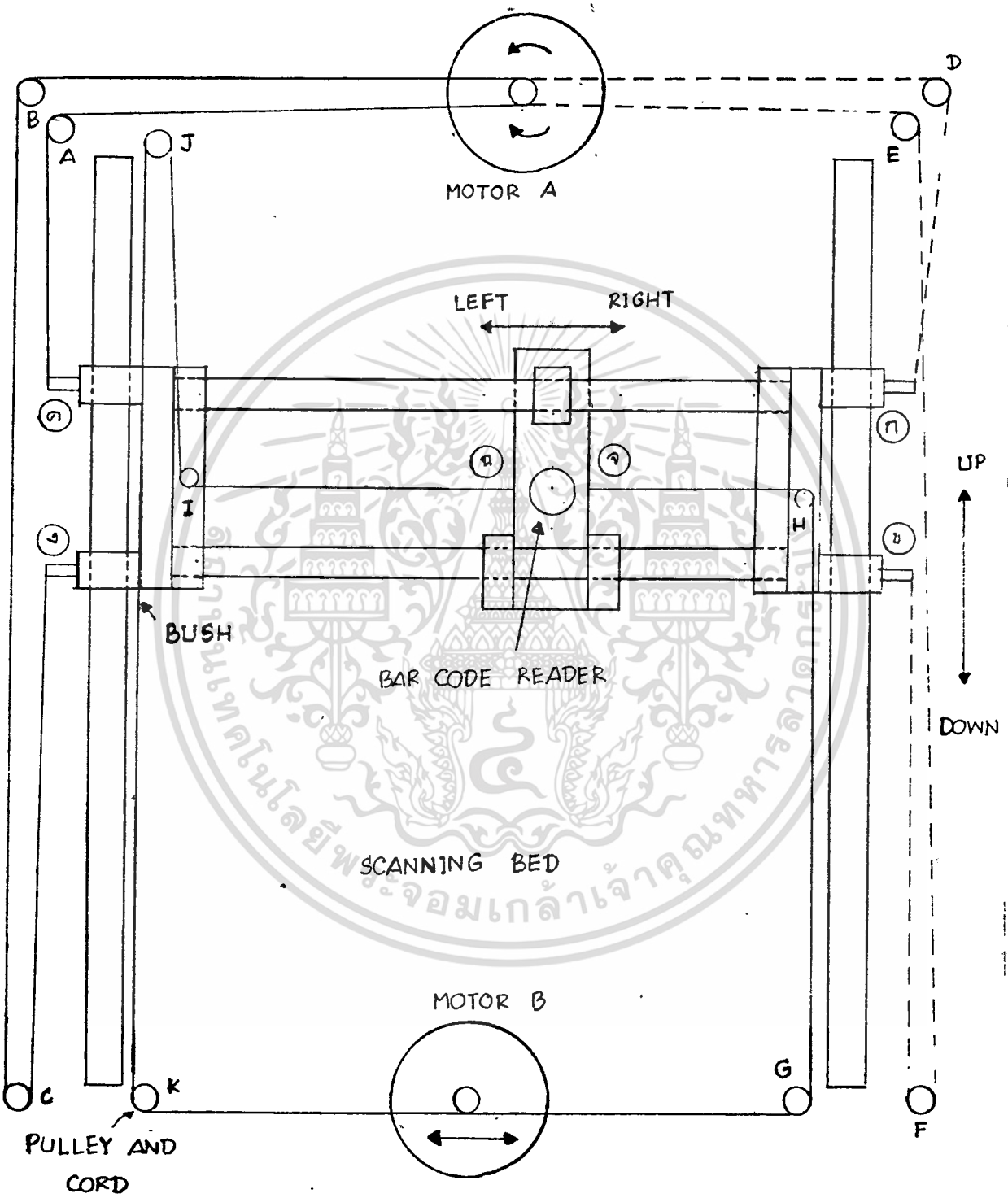
-เมื่อสแกนเนอร์ส่งสัญญาณ STROBE ออกไปแล้ว จะอ่านค่าพอร์ท C เพื่อตรวจสอบบิท PC_3 ว่าเป็น BUSY หรือ READY หากมีค่าเป็น BUSY แสดงว่าไมโครคอมพิวเตอร์ยังไม่พร้อมที่จะรับข้อมูลไบท์ต่อไป ก็จะต้องรอจนกว่าสัญญาณที่ได้จะเป็น READY จึงจะทำการส่งข้อมูลไบท์ต่อไปได้

3.5 ส่วนกลไกและแท่นเครื่อง

องค์ประกอบที่สำคัญประการหนึ่งของโครงงานนี้ได้แก่ ส่วนกลไก ซึ่งจะใช้เป็นส่วนที่ใช้เลื่อนหัวอ่านข้อมูลไปยังตำแหน่งต่างๆที่ต้องการเพื่อสแกนเอาข้อมูลไปเก็บไว้ หลังจากที่ได้ทำการศึกษาค้นคว้าและทดลอง จะได้โครงสร้างโดยรวมของส่วนกลไกดังรูปที่ 3-10

3.5.1 อุปกรณ์และส่วนประกอบ

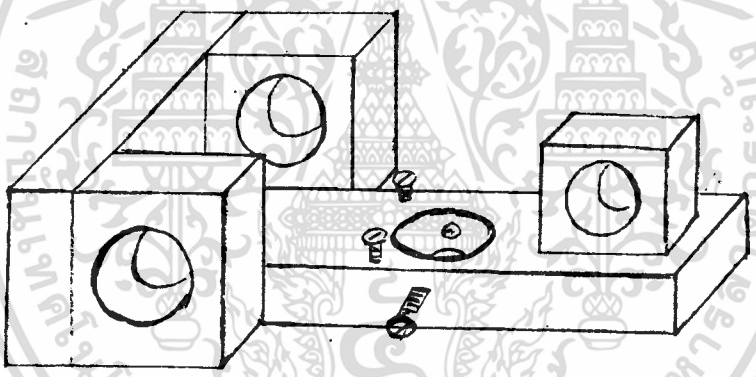
โครงสร้างของส่วนกลไกประกอบไปด้วยส่วนต่างๆดังนี้



รูปที่ 3-10: โครงสร้างรวมของส่วนกลไก

1.แท่งสแตนเลสกลม 2 คู่ คู่แรกเป็นแกนของการเคลื่อนที่ในแนวแกน Y ถูกยึดขนานกันอยู่บนแผ่นพลาสติกพีวีซี ซึ่งทำหน้าที่เป็นฐานรองและเป็นส่วนที่ใช้วางภาพที่ต้องการสแกน คู่ที่สองจะติดตั้งอยู่บนตัวยึดระหว่างแกน X และ Y ดังรูปที่ 3-10 โดยที่แท่งสแตนเลสนี้จะเป็นแกนการเคลื่อนที่ในแนวแกน X

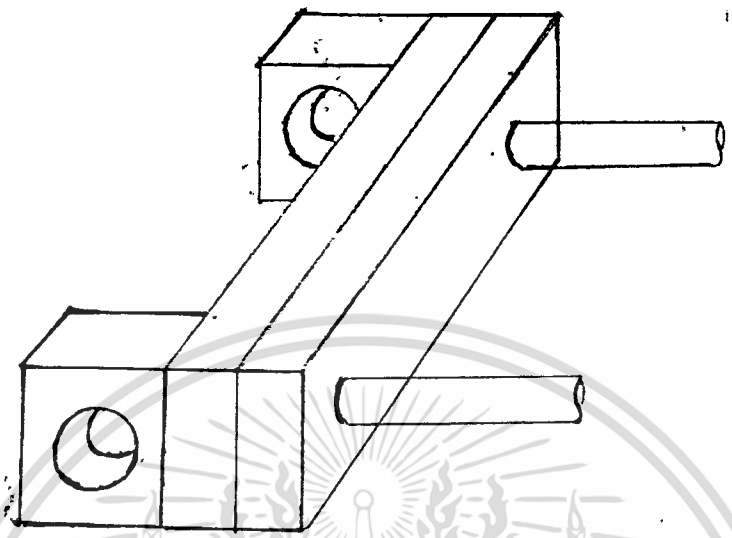
2.ตัวยึดหัวอ่าน สร้างจากชิ้นส่วนพลาสติกประกอบกัน มีลักษณะดังรูปที่ 3-11 ประกอบด้วยชิ้นพลาสติกที่สวมบูชทองเหลืองเพื่อใช้เคลื่อนที่ไปมาตามแนวแกน X ซึ่งจะถูกเชื่อมเข้าด้วยกันด้วยพลาสติกที่ใช้ยึดหัวอ่านข้อมูล การยึดหัวอ่านข้อมูลทำได้โดยการเจาะรูบนชิ้นพลาสติก แล้วนำหัวอ่านมาสวมลงในรูดังกล่าว จากนั้นขันสกรูเพื่อยึดหัวอ่านให้อยู่กับที่ ตัวยึดหัวอ่านนี้จะถูกดึงให้เคลื่อนที่ไปมาด้วยเชือกที่ผูกอยู่กับปลายสกรู การผูกเชือกและขันสกรูจะสัมพันธ์กันเพื่อให้สามารถปรับความตึงของเชือกได้ ชิ้นส่วนตัวยึดหัวอ่านนี้จะสวมลงบนแกนสแตนเลสแกน X



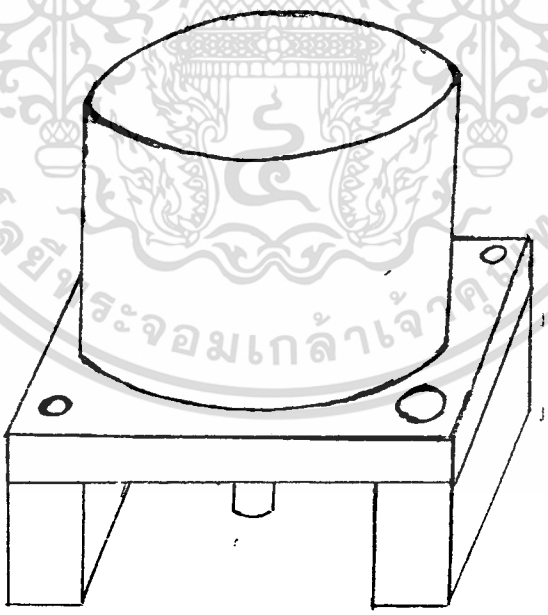
รูปที่ 3-11: ตัวยึดหัวอ่าน

3.ตัวยึดระหว่างแกน X และ Y ทั้งสองชิ้นนี้มีลักษณะเหมือนกัน ดังรูปที่ 3-12 ประกอบด้วยพลาสติกสองชิ้นขนาดเท่ากันรูปสี่เหลี่ยมผืนผ้าวางประกบกันอยู่ แผ่นแรกจะเจาะรูเพื่อสวมแกนสแตนเลสแกน X ดังรูป และมีพลาสติกรูปสี่เหลี่ยมจัตุรัสสองชิ้นที่เจาะรูตรงกลางและสวมบูชเพื่อนำไปสวมบนสแตนเลสแกน Y เพื่อให้เคลื่อนที่ไปมาได้สะดวกตามแนวแกน Y

4.แท่นยึดมอเตอร์ มีลักษณะดังรูปที่ 3-13 ใช้วางมอเตอร์ให้มีลักษณะคว่ำเอาแกนโรเตอร์ของมอเตอร์ลงให้ตั้งฉากกับพื้น และให้แกนโรเตอร์อยู่ในแนวระดับเดียวกันกับเชือกที่จะพันรอบแกนโรเตอร์



รูปที่ 3-12: ตัวยึดระหว่างแกน X และแกน Y



รูปที่ 3-13: แท่นยึดมอเตอร์

5. อุปกรณ์ส่งกำลัง ได้แก่เชือกและรอก เชือกที่ใช้เป็นเชือกที่ใช้ในวิทยุ (ใช้สำหรับยึดกับเข็มบอกสถานีวิทยุให้เข็มเคลื่อนที่ไปในตำแหน่งต่างๆเมื่อมีการจูนคลื่นสถานี) เหตุที่เลือกใช้เชือกประเภทนี้

เพราะมีคุณสมบัติเหนียว ทนแรงดึงได้สูง สัมประสิทธิ์การยืดตัวต่ำ และสามารถโค้งงอเป็นรูปต่างๆได้สะดวก สำหรับรอกใช้เปลี่ยนทิศทางของเชือก

3.5.2 หลักการทำงาน

การเคลื่อนที่ในแกน X หัวอ่านจะเคลื่อนที่ไปทางซ้ายและขวา โดยมีมอเตอร์ B เป็นตัวขับเคลื่อนและส่งกำลังไปตามเชือกที่คล้องผ่านรอก G,H,I,J และ K ปลายเชือกจะถูกยึดอยู่กับสกรู(จ) และ (ฉ) ซึ่งสามารถปรับความตึงเชือกได้ เชือกจะคล้องผ่านรอกต่างๆและพันรอบแกนโรเตอร์ของมอเตอร์ B จำนวน 4 รอบ เมื่อมอเตอร์หมุนตามเข็มนาฬิกาหัวอ่านข้อมูลจะถูกเชือกดึงที่สกรู(จ)ให้เลื่อนมาทางด้านขวา ส่วนเชือกที่สกรู(ฉ) ก็จะถูกผ่อนตามไปด้วย เมื่อมอเตอร์หมุนทวนเข็มนาฬิกาจะทำให้การทำงานกลับกันกับกรณีหมุนตามเข็มนาฬิกา เป็นผลให้หัวอ่านเลื่อนมาทางซ้าย

การเคลื่อนที่ในแกน Y หัวอ่านจะเคลื่อนที่ขึ้นลง โดยมีมอเตอร์ A เป็นตัวขับเคลื่อนและส่งกำลังไปตามเชือกสองเส้น เชือกแต่ละเส้นจะควบคุมการเคลื่อนที่ในแต่ละด้านของแกน ดังแสดงเป็นเส้นประและเส้นทึบในรูปที่ 3-10 พิจารณาเชือกเส้นทึบที่ผูกกับสกรู(ค) ผ่านรอก A มาพันแกนมอเตอร์ A แล้วย้อนกลับไปผ่านรอก B และ C มาผูกกับสกรู(ง) สำหรับสกรู(ค) และ (ง) ใช้ปรับความตึงของเชือก เชือกเส้นที่แสดงด้วยเส้นประ จะผูกกับสกรู(ก) ผ่านรอก D มาพันแกนของมอเตอร์ A แล้วย้อนกลับไปผ่านรอก E และ F มาผูกกับสกรู(ข) โดยที่สกรู(ก) และ (ข) สามารถปรับความตึงของเชือกได้ เมื่อมอเตอร์หมุนตามเข็มนาฬิกาจะทำให้เชือกทั้งสองเส้นมีแรงดึงส่งไปยังสกรู(ง) และ (ข) แล้วทำให้แกน X และหัวอ่านเลื่อนลงมาทางด้านล่าง ส่วนเชือกที่ผูกกับสกรู(ค) และ (ก) จะผ่อน ถ้ากำหนดให้แกนมอเตอร์ B อยู่กับที่ในขณะเดียวกับที่แกน X เลื่อนต่ำลง เชือกที่ควบคุมการเคลื่อนที่ของหัวอ่านในแนวแกน X จะมีการเลื่อนตำแหน่งไปด้วย โดยเชือกที่ผูกกับสกรู(ฉ) จะมีความตึงมากขึ้น จึงเกิดแรงดึงให้หัวอ่านเคลื่อนที่มาทางซ้ายด้วย ในทำนองเดียวกันถ้ามอเตอร์หมุนทวนเข็มนาฬิกา จะทำให้เชือกทั้งสองเส้นมีแรงดึงส่งไปยังสกรู(ก) และ (ค) ทำให้แกน X และหัวอ่านเลื่อนขึ้นไปด้านบน ส่วนเชือกที่ผูกกับสกรู(ข) และ (ง) จะผ่อน ดังนั้นถ้ากำหนดให้แกนมอเตอร์ A อยู่กับที่ในขณะเดียวกับที่แกน X เลื่อนขึ้น เชือกที่ควบคุมการเคลื่อนที่ของหัวอ่านในแนวแกน X จะมีการเลื่อนตำแหน่งไปด้วย โดยเชือกที่ผูกกับสกรู (จ) จะมีความตึงมากขึ้น จึงเกิดแรงดึงให้หัวอ่านเคลื่อนที่มาทางขวาด้วย จะเห็นได้ว่าเมื่อสั่งให้มอเตอร์ A หมุน นอกจากหัวอ่านจะเคลื่อนที่ในแนวขึ้นลงแล้วยังมีการเคลื่อนที่ในแนวซ้ายขวาอีกด้วย ลักษณะดังกล่าวนี้จะก่อให้เกิดปัญหาได้ในขณะที่ทำการสแกนภาพอยู่ การแก้ไขสามารถทำได้โดยการสั่งให้มอเตอร์ B หมุนในทิศทางที่สวนกับการเคลื่อนที่ของเชือกเพื่อชดเชยการเคลื่อนที่ของหัวอ่าน

3.5.3 การคำนวณขนาดของแกนโรเตอร์

ความละเอียดของการสแกนภาพ นอกจากจะขึ้นอยู่กับมุมสเต็ปของสเต็ปมอเตอร์แล้ว ยังขึ้นอยู่กับอัตราการทดรอบ ตลอดจนขนาดของแกนโรเตอร์อีกด้วย ในโครงงานนี้จะใช้วิธีปรับขนาดของแกนโรเตอร์ให้ได้ความละเอียดของการสแกนเป็น 150 จุดต่อนิ้ว โดยไม่ต้องมีการทดรอบ

พิจารณาความละเอียดภาพที่ต้องการ คือ 150 จุดต่อนิ้ว หมายความว่า หัวอ่านข้อมูลต้องเคลื่อนที่ไปได้ 150 สเต็ป ในระยะทาง 1 นิ้ว จากโครงสร้างทางกลไกซึ่งใช้เชือกเป็นอุปกรณ์ส่งกำลัง การควบคุมหัวอ่านทำได้โดยการดึงเชือกในทิศทางที่เหมาะสม ดังนั้น การดึงเชือกแต่ละครั้งจะต้องให้ได้ระยะเท่ากับ 1/150 นิ้ว แต่เนื่องจากมุมสเต็ปของสเต็ปมอเตอร์ที่ใช้มีค่าเท่ากับ 1.8 องศา หรือ 1/200 รอบ และกำหนดให้สเต็ปมอเตอร์เคลื่อนที่ 1 สเต็ปมีค่าเท่ากับ 1 จุดภาพ

ระยะทาง 1/200 รอบ มีความยาว 1/150 นิ้ว

ระยะทาง 1 รอบ มีความยาว 200/150 = 1.333 นิ้ว

ดังนั้นแกนโรเตอร์จะต้องมีเส้นรอบวงเท่ากับ 1.333 นิ้ว นั่นคือแกนโรเตอร์จะมีเส้นผ่านศูนย์กลางเท่ากับ 1.333/3.14159 = 0.4243 นิ้ว หรือมีรัศมีเท่ากับ 0.2122 นิ้ว

เนื่องจากสเต็ปมอเตอร์ที่ใช้ในโครงงานนี้ มีเส้นผ่านศูนย์กลางของแกนโรเตอร์เท่ากับ 0.25 นิ้ว ดังนั้นจึงต้องกลึงแกนเหล็กที่มีเส้นผ่านศูนย์กลาง 0.4243 นิ้ว มาสวมลงบนแกนโรเตอร์ เพื่อที่จะให้ความละเอียดในการสแกนมีค่าเท่ากับ 150 จุดต่อนิ้วพอดี

บทที่ 4
การพัฒนาโปรแกรมบนบอร์ดควบคุม

เนื่องจากสแกนเนอร์ที่พัฒนาขึ้นนี้ มีจุดมุ่งหมายให้สามารถทำงานได้เองโดยไม่ต้องอาศัยการควบคุมจากเครื่องไมโครคอมพิวเตอร์ เพราะต้องใช้เวลาในการสแกนภาพเป็นเวลานาน หากใช้ไมโครคอมพิวเตอร์เข้าควบคุม จะทำให้ไม่สามารถใช้ไมโครคอมพิวเตอร์ทำงานอื่นได้ ดังนั้นการทำงานของสแกนเนอร์นี้จะถูกควบคุมโดยไมโครโปรเซสเซอร์ Z-80 ให้อ่านข้อมูลมาเข้ารหัสและเก็บไว้ในหน่วยความจำบนบอร์ดควบคุม แล้วจึงส่งข้อมูลเข้าไมโครคอมพิวเตอร์ต่อไป การทำงานของ Z-80 จะถูกควบคุมโดยโปรแกรมมอนิเตอร์ซึ่งบันทึกอยู่ในหน่วยความจำถาวรบนบอร์ดควบคุม

4.1 หน้าที่ของโปรแกรมมอนิเตอร์

โปรแกรมมอนิเตอร์ที่พัฒนาขึ้น มีหน้าที่ดังนี้

1. ควบคุมการหมุนของสเต็ปมอเตอร์
2. ควบคุมการอ่านและเข้ารหัสข้อมูลภาพ
3. เก็บข้อมูลภาพที่เข้ารหัสแล้ว ลงในหน่วยความจำชั่วคราวบนบอร์ดควบคุม
4. ควบคุมการส่งข้อมูลแบบขนานระหว่างบอร์ดควบคุม กับเครื่องไมโครคอมพิวเตอร์

นอกจากหน้าที่ดังกล่าวนี้แล้ว ยังได้ออกแบบโปรแกรมมอนิเตอร์ให้สามารถติดต่อกับผู้ใช้ได้ นั่นคือการควบคุม การแสดงผลด้วยไดโอดเปล่งแสง (LED) การส่งเสียงเตือน และการรับคำสั่งผ่านทางสวิทช์กด จากผู้ใช้

4.2 ขั้นตอนการพัฒนาและการทำงานของโปรแกรมมอนิเตอร์

ก่อนที่จะพัฒนาโปรแกรมมอนิเตอร์ได้นั้น มีความจำเป็นจะต้องกำหนดขั้นตอนการทำงานของสแกนเนอร์เสียก่อน ซึ่งสรุปขั้นตอนการทำงานได้ดังนี้

1.หาตำแหน่งโคออร์ดิเนต X,Y = 0,0 เพื่อกำหนดตำแหน่งเริ่มต้นในการสแกน ขั้นตอนนี้จะเรียกสั้นๆต่อไปว่าการหาตำแหน่งเริ่มต้นการสแกน

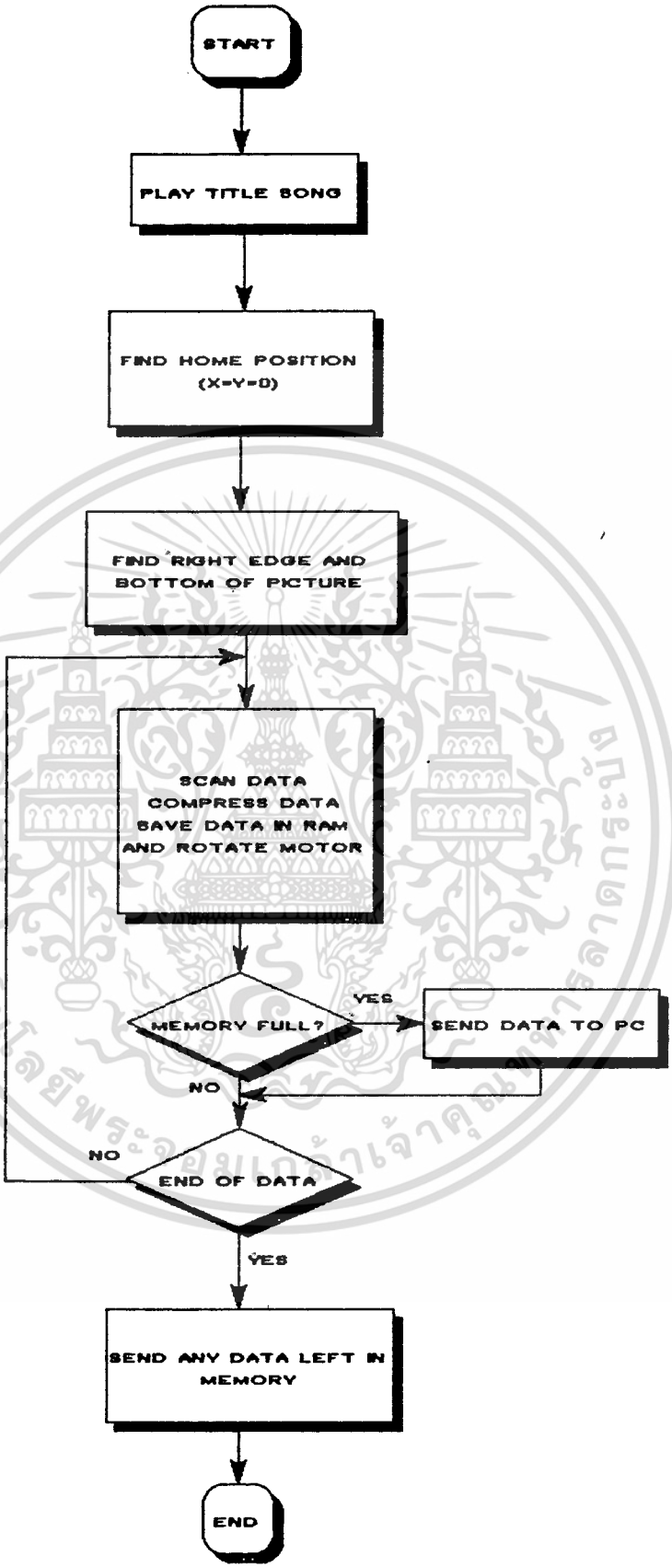
2.หาขอบเขตการสแกนภาพ เพื่อกำหนดขอบขวาของภาพ หรือจำนวนจุดการสแกนในแต่ละแถวสแกน(Scan line) และกำหนดขอบล่างสุดของภาพ หรือ จำนวนแถวที่จะสแกนนั่นเอง ขั้นต่อนนี้เรียกว่าการหาขอบเขต การทำงานในขั้นตอนนี้จะเริ่มเมื่อผู้ใช้กดสวิทช์ "MARGIN" หัวอ่านข้อมูลจะเลื่อนไปทางขวาตามแนวแกน X จนกว่าผู้ใช้จะกดสวิทช์ "MARGIN" อีกครั้งหนึ่ง หรือเลื่อนไปจนสุดขอบเขตการสแกนสูงสุด จึงจะเลื่อนกลับมาทางซ้ายจนถึงตำแหน่งเริ่มต้น เมื่อผู้ใช้กดสวิทช์ "MARGIN" อีกครั้งหัวอ่านจะเลื่อนลงไปตามแนวแกน Y จนกว่าจะสุดขอบเขตการสแกนสูงสุด หรือจนกว่าผู้ใช้จะกดสวิทช์ "MARGIN" อีกครั้ง หัวอ่านจึงจะเลื่อนกลับมายังตำแหน่งเริ่มต้น

3.เริ่มทำการอ่านข้อมูลภาพจากหัวอ่าน เมื่อผู้ใช้กดสวิทช์ "START" โดยเลื่อนหัวอ่านจากซ้ายไปขวา จนสุดขอบเขตที่กำหนดไว้ จึงเลื่อนลง 1 แถว แล้วเลื่อนหัวอ่านจากขวามาซ้าย ทำเช่นนี้สลับกันไปจนกระทั่งสุดขอบเขตของภาพที่จะสแกน ในระหว่างนั้น จะมีการอ่านข้อมูลภาพ นำมาเข้ารหัสแบบ EDE แล้วจึงจัดเก็บไว้ในหน่วยความจำชั่วคราว นอกจากนี้ยังจะต้องทำการตรวจสอบหน่วยความจำอยู่เสมอทุกครั้งที่เก็บข้อมูลลงไปในหน่วยความจำ ถ้าไม่มีที่ว่างที่จะเก็บข้อมูลได้ต่อไป จะต้องส่งข้อมูลไปให้เครื่องไมโครคอมพิวเตอร์ก่อน แล้วจึงสแกนภาพต่อไปจนหมดจึงส่งข้อมูลที่เหลือเข้าเครื่องไมโครคอมพิวเตอร์อีกครั้ง ซึ่งขั้นตอนนี้จะเป็นขั้นตอนที่มีอัลกอริทึม(Algorithm)ที่ยุ่งยากซับซ้อนและเกี่ยวข้องสัมพันธ์กันมากในการพัฒนาโปรแกรมมอนิเตอร์ จนไม่สามารถที่จะแยกแยะออกเป็นขั้นตอนย่อยๆได้โดยง่าย ดังนั้นในที่นี้จึงถือว่าการทำงานทั้งหมดในข้อนี้เป็นขั้นตอนเดียว

ดังนั้นจะสรุปขั้นตอนในการพัฒนาโปรแกรมมอนิเตอร์ได้ดังรูปที่ 4-1 ซึ่งจะช่วยให้เข้าใจการทำงานของสแกนเนอร์ และแนวทางในการพัฒนาโปรแกรมมอนิเตอร์ได้ดียิ่งขึ้น สำหรับรายละเอียดต่างๆในแต่ละขั้นตอนจะได้กล่าวถึงต่อไป

4.2.1 การทำงานของโปรแกรมหาตำแหน่งเริ่มต้นการสแกน

จากการทำงานของเครื่องสแกนเนอร์ที่กล่าวมาแล้ว เราได้ทราบแล้วว่า การหาตำแหน่งเริ่มต้นการสแกนของหัวอ่านข้อมูล จะทำได้โดยการใช้ตัวตรวจจับทางแสง ถ้าวัตถุทึบแสงมาบังลำแสงของตัวตรวจจับ จะทำให้สัญญาณที่ออกจากตัวตรวจจับเปลี่ยนแปลงไป ซึ่งทำให้ตรวจสอบได้ว่าหัวอ่านเคลื่อนที่เข้ามาถึงตำแหน่งเริ่มต้นแล้วหรือยัง จากจุดนี้เองจึงนำมาเป็นแนวทางในการพัฒนาโปรแกรมส่วนนี้ ดังรูปที่ 4-2 ซึ่งสามารถอธิบายได้ดังนี้

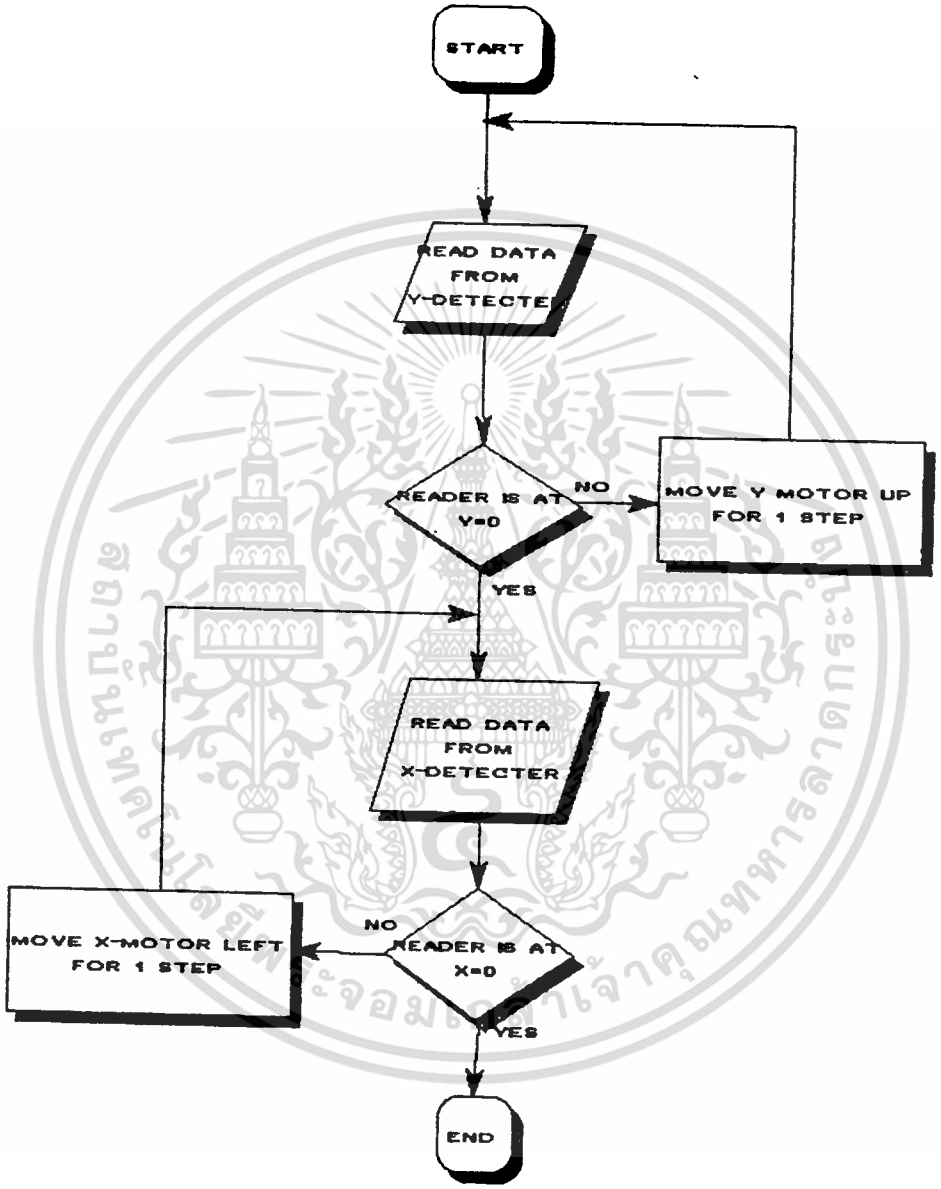


รูปที่ 4-1: การทำงานหลักของโปรแกรมมอนิเตอร์

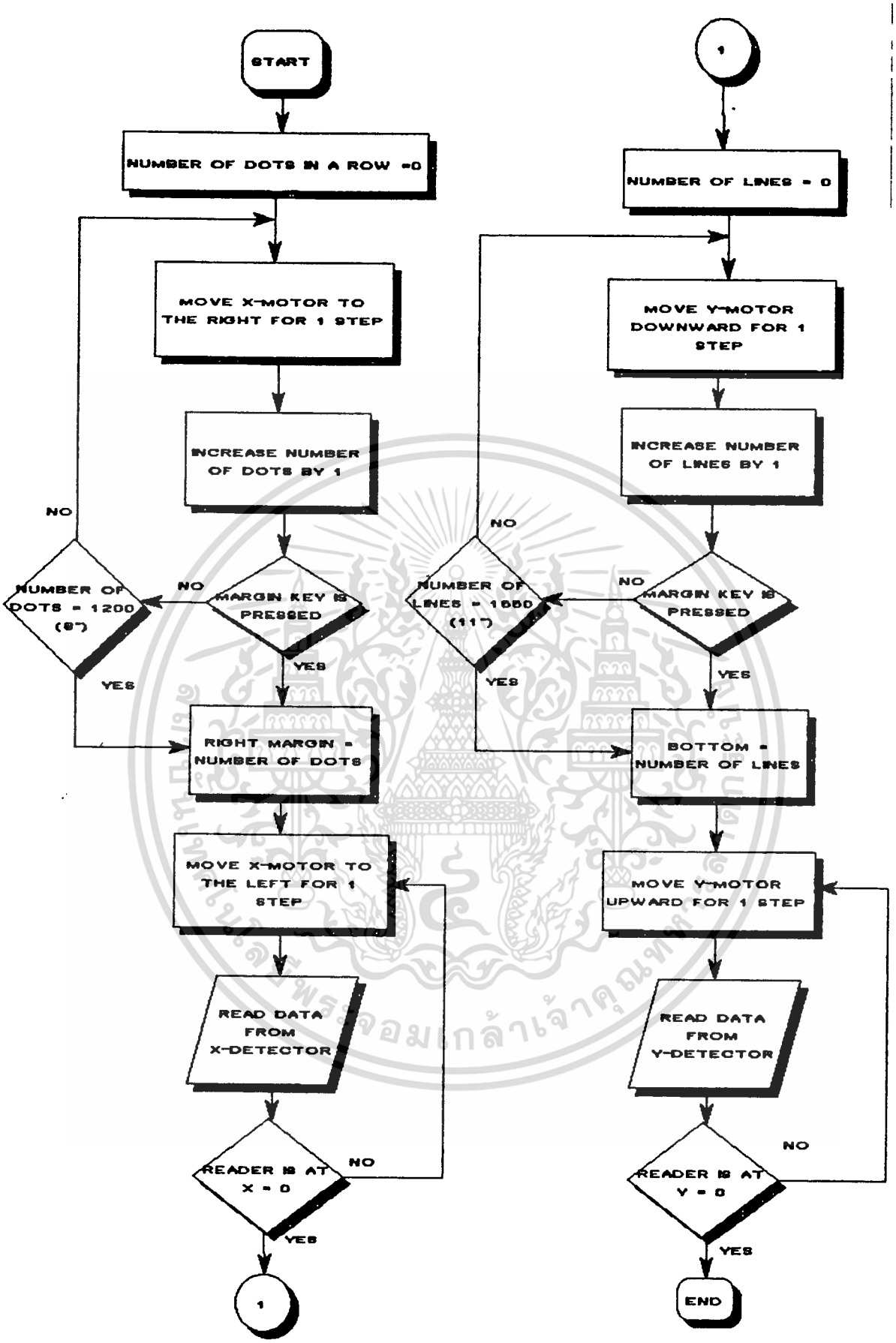
เมื่อจ่ายไฟให้แก่สแกนเนอร์ สแกนเนอร์จะหาตำแหน่งเริ่มต้นจากแกน Y ก่อน แล้วจึงหาจากแกน X ซึ่งในแต่ละแกนจะมีหลักการเหมือนกัน กล่าวคือ จะต้องทำการอ่านข้อมูลเข้ามาทางพอร์ท C ของ 8255A ที่ต่ออยู่กับตัวตรวจจับทางแสง แล้วพิจารณาสัญญาณที่ได้ ถ้าตรวจสอบได้ว่าเป็นตำแหน่งเริ่มต้นแล้วจะสิ้นสุดการทำงาน แต่ถ้าไม่ใช่ตำแหน่งเริ่มต้นจะต้องหมุนมอเตอร์เพื่อให้หัวอ่านเลื่อนขึ้นตามแนวแกน Y หรือเพื่อให้หัวอ่านเลื่อนมาทางซ้ายตามแนวแกน X แล้วจึงตรวจสัญญาณจากตัวตรวจจับทางแสงอีกครั้ง จนกว่าหัวอ่านจะมาอยู่ที่ตำแหน่งเริ่มต้นจึงจะหยุดการทำงานในส่วนนี้ ดังนั้นเราสามารถสรุปการทำงานในขั้นนี้ได้ดังรูปที่ 4-2 ในหน้าที่ 51

4.2.2 <u>การทำงานของโปรแกรมในส่วนการหาขอบเขตภาพ</u>

เริ่มต้นจะทำการหาขอบเขตของภาพในแนวแกน X ก่อน แล้วจึงหาขอบเขตในแนวแกน Y การทำงานจะเริ่มจากการหมุนมอเตอร์แกน X ให้หัวอ่านข้อมูลเลื่อนไปทางขวาแล้วนับจำนวนสเต็ปที่เลื่อนไปทีละจุด ในการเลื่อนหัวอ่านแต่ละครั้งจะมีการตรวจสอบว่ามีการกดสวิทช์ "MARGIN" หรือไม่ ถ้าผู้ใช้ไม่กดสวิทช์ ก็จะหมุนมอเตอร์ต่อไปแล้วนับจำนวนสเต็ปเพิ่มไปเรื่อยๆ จนกว่าจะสุดขอบขวาที่ระยะทาง 8 นิ้ว(คิดเป็น 1200 สเต็ป) หรือจนกว่าผู้ใช้กดสวิทช์ "MARGIN" จึงเก็บจำนวนสเต็ปไว้ในหน่วยความจำชั่วคราวของบอร์ดควบคุม จำนวนสเต็ปที่เก็บไว้นี้จะนำไปใช้กำหนดขอบเขตของภาพในเวลาสแกนภาพ แล้วจึงหมุนมอเตอร์แกน X ให้หัวอ่านเลื่อนกลับทางซ้ายมาทีละ 1 สเต็ป ทุกครั้งที่หัวอ่านเลื่อนกลับมาจะต้องตรวจสอบตำแหน่งเริ่มต้นเสมอ จนกว่าจะถึงตำแหน่งเริ่มต้นจึงหยุดหมุนมอเตอร์ จากนั้นจึงเป็นการหาขอบล่างโดยการหมุนมอเตอร์ให้หัวอ่านเลื่อนลงตามแนวแกน Y การทำงานก็คล้ายกับแกน X ต่างกันที่ระยะล่างสุดเป็น 11 นิ้ว ไม่ใช่ 8 นิ้ว ดังนั้นจึงสามารถสรุปการทำงานออกมาได้ดังรูปที่ 4-3 ในหน้าที่ 52



รูปที่ 4-2: ขั้นตอนการหาตำแหน่งเริ่มต้นการสแกน



รูปที่ 4-3: ขั้นตอนการหาขอบเขตในการสแกนภาพ

4.2.3 การทำงานในขั้นตอนสุดท้ายของโปรแกรม

ดังที่ได้กล่าวมาข้างต้นแล้วว่า การทำงานในขั้นตอนนี้มีความยุ่งยากซับซ้อนมาก เนื่องจากจะต้องทำงานสัมพันธ์กันจนแยกออกเป็นส่วนย่อยๆได้ยาก แต่ก็สามารถสรุปหน้าที่ในขั้นตอนนี้ได้ดังต่อไปนี้

1. รับข้อมูลเข้ามาทีละจุดภาพ

2. เข้ารหัสข้อมูลแบบ EDE(รายละเอียดอ่านได้ในบทที่ 2) แล้วจึงนำข้อมูลที่เข้ารหัสแล้วไปเก็บไว้ในหน่วยความจำชั่วคราวของบอร์ดควบคุม

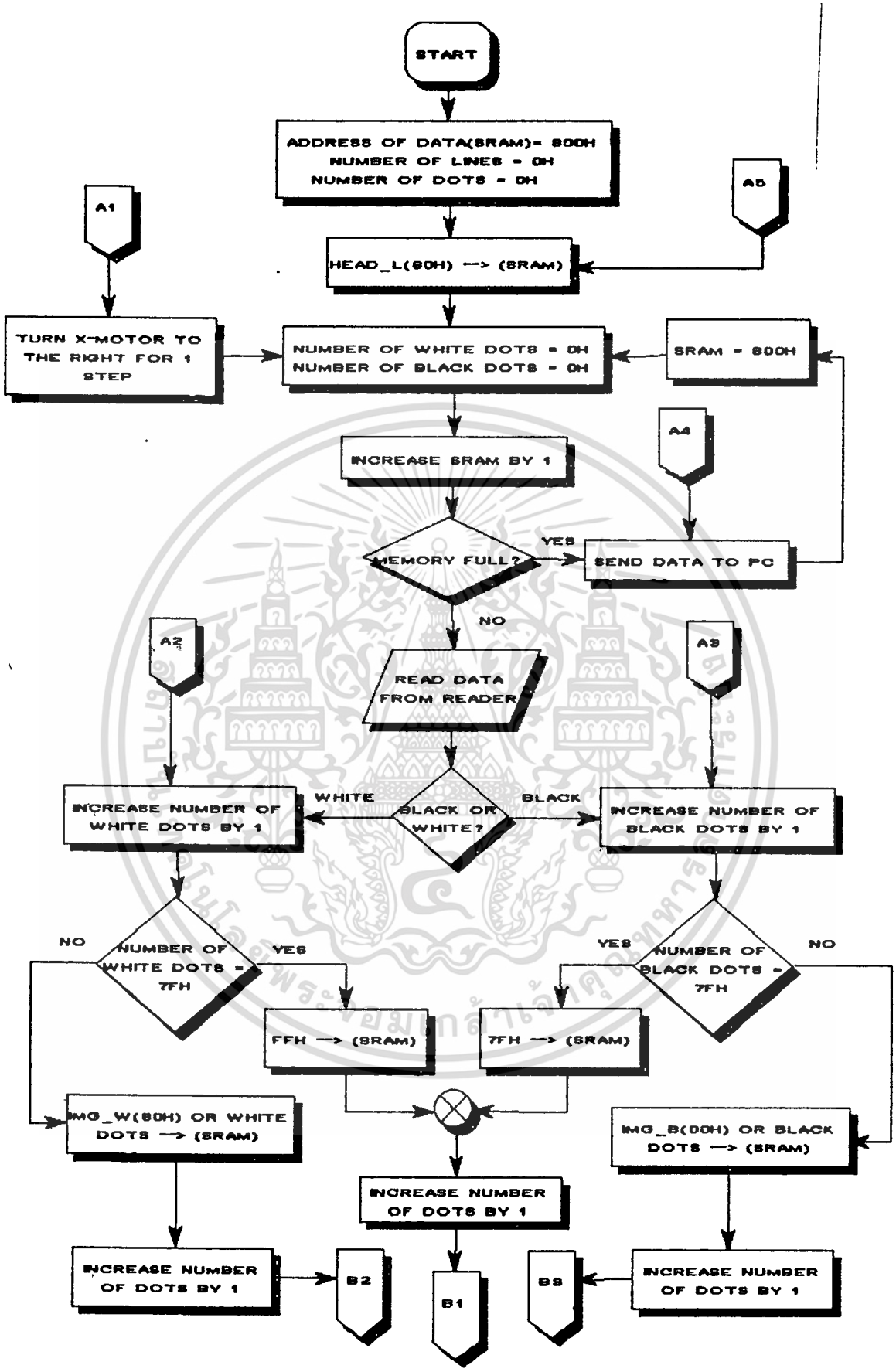
3. การแทรกรหัสบอกทิศทาง(รหัสจบแถวสแกน)ลงในหน่วยความจำชั่วคราวของบอร์ดควบคุม

4. ตรวจสอบว่าหน่วยความจำพร้อมที่จะเก็บข้อมูลตัวต่อไปหรือไม่

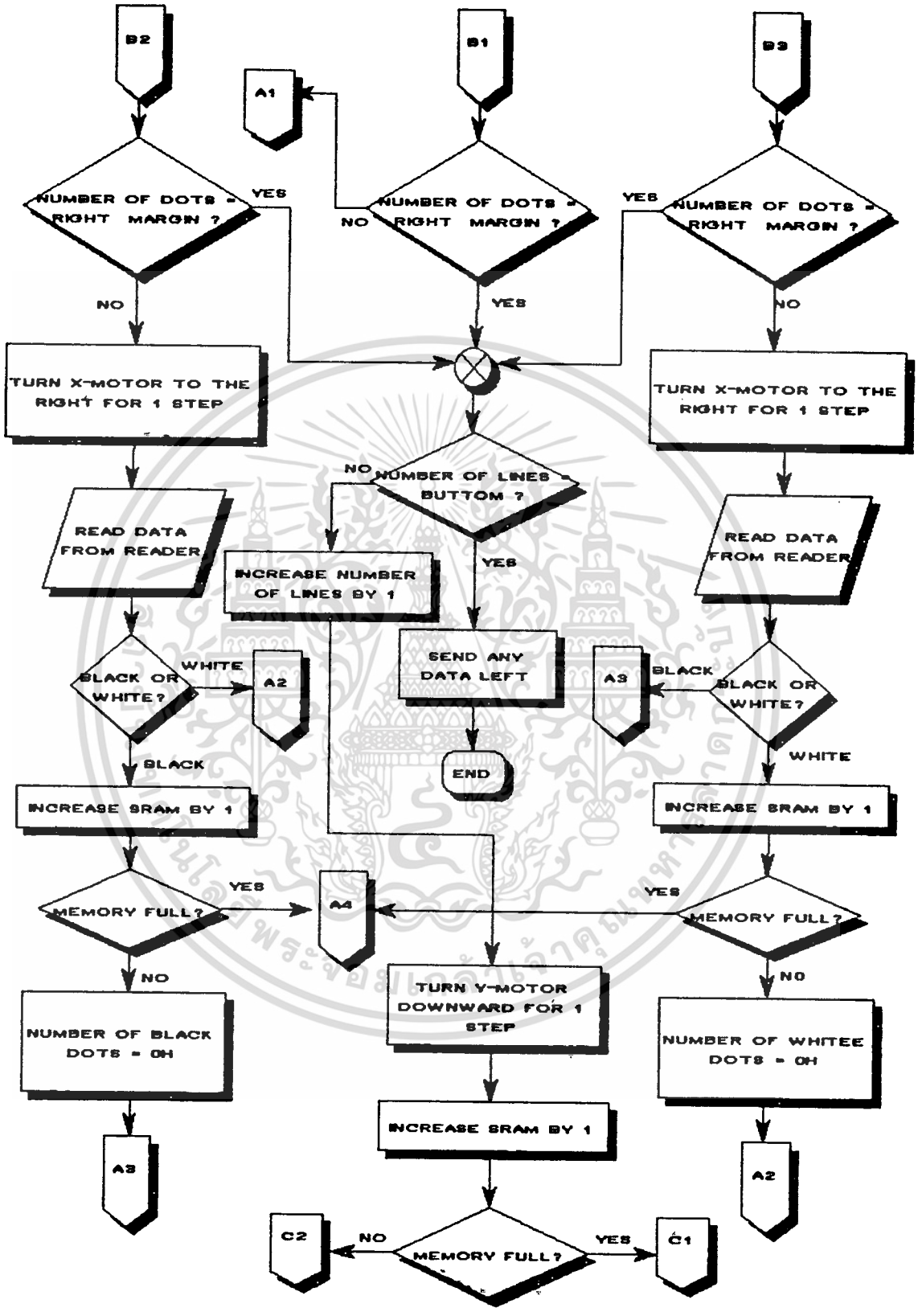
5. ควบคุมการหมุนของมอเตอร์ทั้งสองแกน

6. ส่งข้อมูลไปให้เครื่องไมโครคอมพิวเตอร์

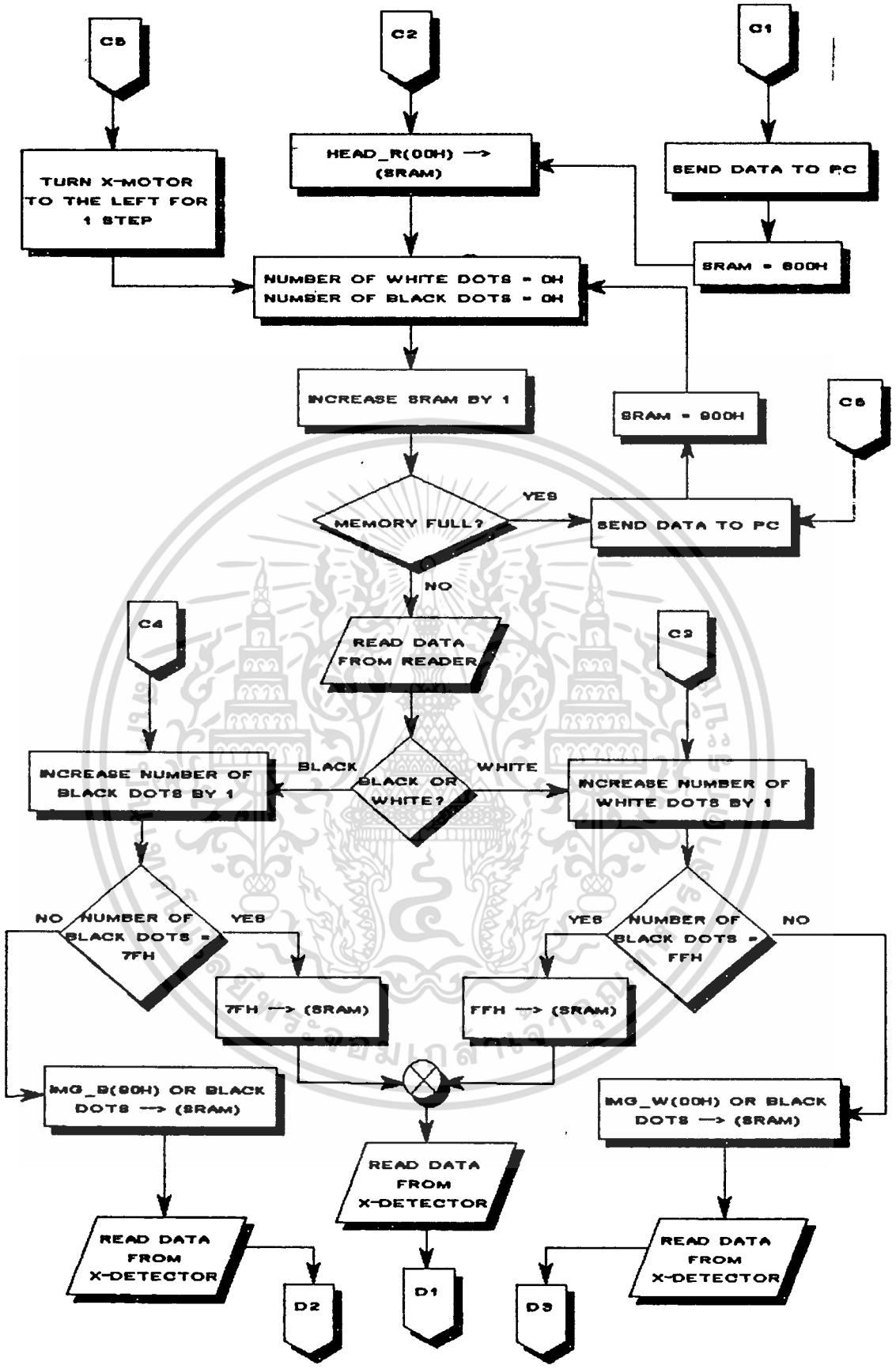
จะเห็นได้ว่าในขั้นนี้จะมีการทำงานอยู่มากมายหลายอย่าง ที่เกี่ยวข้องสัมพันธ์กันอยู่อย่างซับซ้อนจนยากที่จะแยกออกมากล่าวได้ทีละส่วน จึงต้องนำมาแสดงรวมกันไว้ดังรูปที่ 4-4 ในหน้า 54-57 ดังนั้นจะไม่อธิบายรายละเอียดเป็นตัวอักษร เนื่องจากการพิจารณาขั้นตอนต่างๆจากรูปภาพจะสะดวกกว่ามาก



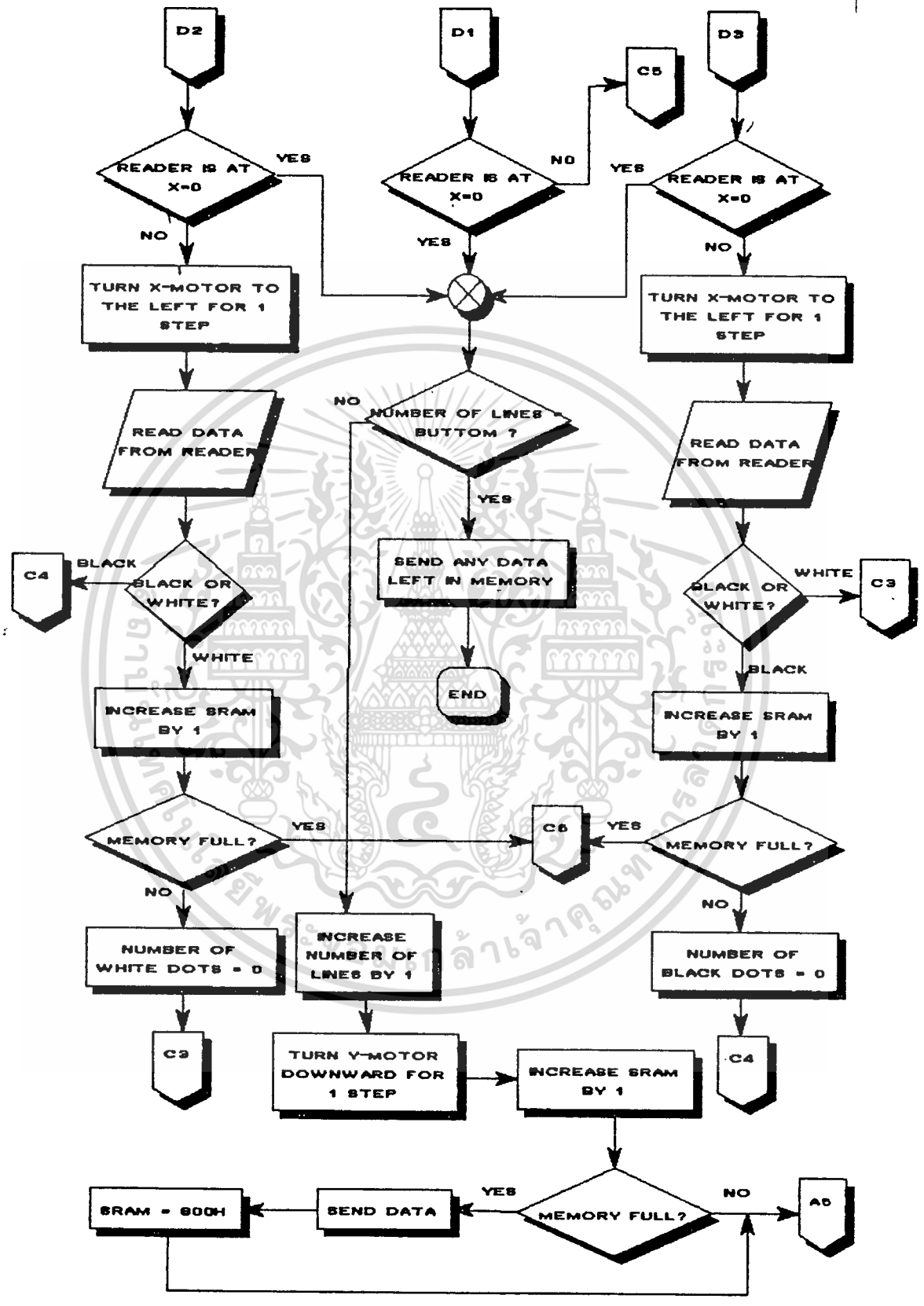
รูปที่ 4-4: ขั้นตอนการทำงานในส่วนสุดท้ายของโปรแกรม



รูปที่ 4-4: ขั้นตอนการทำงานในส่วนสุดท้ายของโปรแกรม(ต่อ1)



รูปที่ 4-4: ขั้นตอนการทำงานในส่วนสุดท้ายของโปรแกรม(ต่อ2)



รูปที่ 4-4: ขั้นตอนการทำงานในส่วนสุดท้ายของโปรแกรม(ต่อ3)

4.3 การแบ่งพื้นที่หน่วยความจำบนบอร์ดควบคุม

บนบอร์ดควบคุมการทำงานประกอบด้วยหน่วยความจำสองชนิด ดังนี้คือ

1.หน่วยความจำถาวร ใช้เก็บโปรแกรมมอนิเตอร์ที่ใช้ควบคุมการทำงานของบอร์ดควบคุม

2.หน่วยความจำชั่วคราว ใช้เก็บข้อมูลดังต่อไปนี้

-ข้อมูลภาพที่เข้ารหัสแล้ว และรหัสบอกทิศทางการสแกน

-ข้อมูลที่ถูกพุช(Push)ลงสแตก(Stack)เพื่อเก็บไว้ชั่วคราว ในระหว่างการทำงานของโปรแกรมมอนิเตอร์

-ค่าตัวแปรต่างๆ ได้แก่

-ตำแหน่งขอบเขตขวาสุด(RIGHTMAR)

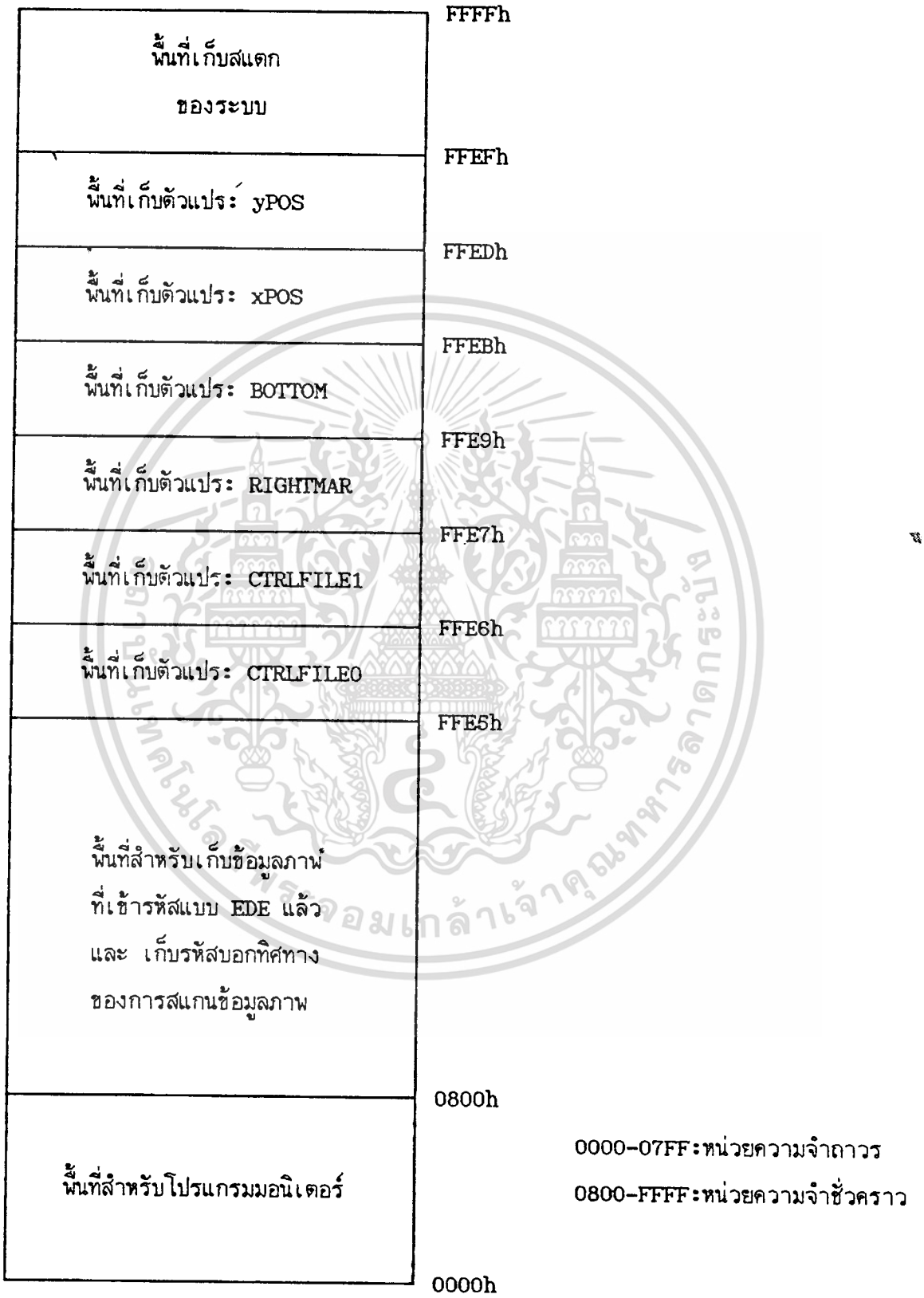
-ตำแหน่งล่างสุดในการสแกนภาพ(BOTTOM)

-แอดเดรสของตารางข้อมูลที่ใช้ขับสเต็ปมอเตอร์ครั้งสุดท้าย(xPOS,yPOS)

-ข้อมูลบอกสถานะ(CTRLFILE0,CTRLFILE1)

ในการพัฒนาโปรแกรมมอนิเตอร์ ผู้เขียนโปรแกรมมีความจำเป็นที่จะต้องแบ่งพื้นที่หน่วยความจำต่างๆเอง เพื่อความสะดวกในการอ้างถึง และป้องกันการซ้อนทับกันของข้อมูลต่างๆ หากไม่ได้มีการวางโครงสร้างการแบ่งพื้นที่หน่วยความจำขึ้นแล้ว อาจทำให้การทำงานของบอร์ดควบคุมเกิดความผิดพลาดได้ ดังนั้นจึงได้แบ่งพื้นที่หน่วยความจำออกเป็นส่วนต่างๆดังรูปที่ 4-5 ในหน้าที่ 59

จากที่ได้กล่าวมาแล้วทั้งหมดในบทนี้ เมื่อนำมาเป็นแนวทางในการพัฒนาโปรแกรมมอนิเตอร์จะได้โปรแกรมมอนิเตอร์ในภาคผนวก ก



รูปที่ 4-5: การแบ่งพื้นที่หน่วยความจำบนบอร์ดควบคุม

บทที่ 5
การพัฒนาโปรแกรมใช้งานบนเครื่องไมโครคอมพิวเตอร์ส่วนบุคคล

ในการพัฒนาโครงงานนี้ องค์ประกอบที่สำคัญอย่างหนึ่งได้แก่ โปรแกรมใช้งานบนเครื่องไมโครคอมพิวเตอร์ ซึ่งมีความสำคัญในการช่วยเพิ่มประสิทธิภาพและขอบเขตการใช้งานของสแกนเนอร์ให้กว้างขวางและสะดวกมากยิ่งขึ้น

ในการพัฒนาเครื่องสแกนเนอร์นี้ได้กำหนดไว้ว่า ให้ตัวเครื่องสามารถทำงานได้โดยไม่ต้องอาศัยการควบคุมจากเครื่องไมโครคอมพิวเตอร์ เพื่อให้สามารถทำงานได้อย่างสะดวกคล่องตัว และเป็นการใช้ทรัพยากรอย่างคุ้มค่าอีกด้วย ดังนั้นการนำเอาไมโครคอมพิวเตอร์มาใช้ร่วมกับสแกนเนอร์นี้จึงมิได้มีจุดมุ่งหมายเพื่อการควบคุมสแกนเนอร์ แต่มีจุดมุ่งหมายดังนี้

1. เก็บข้อมูลภาพที่สแกนเนอร์อ่านมาได้ ให้อยู่ในรูปแบบที่สามารถนำออกมาใช้ได้โดยสะดวกในภายหลัง

2. แสดงผลข้อมูลภาพที่อ่านมาได้ หรือที่เก็บเอาไว้แล้วออกทางจอภาพ หรือทางเครื่องพิมพ์

3. จัดการข้อมูลภาพที่อ่านมาได้ ให้สามารถนำไปใช้กับโปรแกรมสำเร็จรูปทางกราฟิกอื่นๆได้

ไมโครคอมพิวเตอร์ที่เลือกมาใช้งานร่วมกับสแกนเนอร์นั้น ได้แก่ เครื่องไอบีเอ็มพีซี เนื่องจากมีราคาไม่แพง และใช้กันแพร่หลาย สำหรับโปรแกรมใช้งานที่พัฒนาขึ้นมาบนไมโครคอมพิวเตอร์นี้ใช้ภาษาซีเป็นเครื่องมือในการพัฒนา โดยใช้โปรแกรมเทอร์โบซี(Turbo C 2.0) เป็นตัวแปลภาษา

5.1 รูปแบบของไฟล์ข้อมูล

ก่อนจะกล่าวถึงรายละเอียดเกี่ยวกับการพัฒนาโปรแกรม ควรทราบถึงรูปแบบของไฟล์ข้อมูลซึ่งจะเป็นตัวกำหนดขั้นตอนการทำงานของโปรแกรม รูปแบบของไฟล์ข้อมูลขึ้นอยู่กับปัจจัยหลายประการได้แก่

-วิธีการเข้ารหัสข้อมูลภาพ

-ลักษณะการเก็บข้อมูลของสแกนเนอร์

-ลักษณะการติดต่อข้อมูลระหว่างสแกนเนอร์กับไมโครคอมพิวเตอร์

-วิธีการที่ใช้ในการจัดการข้อมูล เช่น การจัดเก็บ หรือการนำข้อมูลออกแสดง

จากปัจจัยที่กำหนดรูปแบบของไฟล์ข้อมูล จะเห็นได้ว่า ในแต่ละขั้นตอนของการทำงาน จะมีรูปแบบข้อมูลที่เหมาะสมแตกต่างกันไป เพื่อให้สอดคล้องกับปัจจัยแต่ละประการ เช่น รูปแบบข้อมูลที่เหมาะสมสำหรับการจัดเก็บ อาจจะไม่เหมาะสมสำหรับการแสดงผล

จากรูปแบบข้อมูลแบบต่างๆ เมื่อนำมารวบรวมเป็นหมวดหมู่ จะสามารถกำหนดเป็นรูปแบบของไฟล์ได้ 2 รูปแบบคือ

1.รูปแบบของไฟล์ข้อมูลชั่วคราว(Temporary File)

2.รูปแบบของไฟล์ข้อมูลภาพ(Image File:IMG File)

5.1.1 รูปแบบของไฟล์ข้อมูลชั่วคราว(Temporary File)

เป็นไฟล์ที่สร้างขึ้นจากข้อมูลภาพที่ได้จากสแกนเนอร์ ซึ่งส่งผ่านมาทางอินเทอร์เฟสการ์ด โดยยังไม่ผ่านกระบวนการจัดการข้อมูลใดๆ มีลักษณะสำคัญดังนี้

1.ข้อมูลในไฟล์จะมีหน่วยย่อยที่สุดเป็นไบท์

2.ข้อมูลในไฟล์จะถูกเข้ารหัสแบบ EDE

3.มีรหัสจบบรรทัด(หรือรหัสบอกทิศทางการสแกน) ซึ่งแบ่งออกเป็น 2 แบบคือ EOLL (End Of Line Left) และ EOLR (End Of Line Right) เหตุที่ต้องมีรหัสจบบรรทัด ก็เนื่องจากวิธีในการเข้ารหัสข้อมูลแบบ EDE ซึ่งจะมีผลทำให้จำนวนข้อมูลที่ถูกเข้ารหัสแล้วในแต่ละแถว มีจำนวนข้อมูลไม่เท่ากัน แม้จะมีจำนวนจุดภาพทางแกน X เท่ากันก็ตาม จำนวนข้อมูลในแต่ละแถวสแกน จะขึ้นอยู่กับจำนวนครั้งของการเปลี่ยนแปลงสีของจุดภาพจากขาวไปดำ และจากดำไปขาว ซึ่งเป็นเหตุให้ไม่สามารถระบุจำนวนข้อมูลที่แน่นอนได้ในแต่ละแถวสแกน ส่วนการที่ต้องมีรหัสจบบรรทัด 2 แบบ คือ EOLL และ EOLR ก็เนื่องมาจากการที่กำหนดให้สแกนเนอร์อ่านข้อมูลได้ทั้งในทิศทางไปและกลับเพื่อประหยัดเวลา ดังนั้นข้อมูลที่ได้จึงมี 2 ลักษณะ คือ ข้อมูลที่อ่านจากซ้ายไปขวา และข้อมูลที่อ่านจากขวาไปซ้าย

4.จำนวนของ EOLL จะต่างจากจำนวนของ EOLR อยู่ไม่เกิน 1 ไบท์เสมอ เนื่องจากการสแกนทำในลักษณะที่สลับจากซ้ายไปขวา และจากขวาไปซ้ายสลับกันไปมาเสมอ

5.ไม่มีข้อมูลอื่น นอกจากข้อมูลภาพและรหัสจบบรรทัดอยู่ในไฟล์เลย

5.1.2 รูปแบบของไฟล์ข้อมูลภาพ(Image File : IMG File)

เป็นไฟล์ที่สร้างจากไฟล์ข้อมูลชั่วคราว เพื่อความสะดวกในการจัดเก็บและแสดงผล เหตุที่ต้องมีการสร้างรูปแบบของไฟล์ข้อมูลภาพขึ้น เนื่องจากรูปแบบของไฟล์ชั่วคราวมีข้อเสียหลายประการดังนี้

-ข้อมูลไม่ได้มีลักษณะเดียวกันตลอดทั้งไฟล์ แต่แบ่งเป็นสองลักษณะสลับกันทุกแถวสแกน โดยที่ข้อมูลลักษณะแรกมีทิศทางจากซ้ายไปขวา และลักษณะที่สองมีทิศทางจากขวาไปซ้าย ทำให้การพัฒนาโปรแกรมย่อยเพื่อการแสดงภาพและจัดเก็บภาพที่มีรูปแบบของไฟล์ชั่วคราว ทำได้อย่างยากลำบาก และมีประสิทธิภาพต่ำ

-การที่ไม่มีข้อมูลหรือรายละเอียดอื่นๆอยู่กับไฟล์เลย ทำให้เกิดความยุ่งยากในการตรวจสอบชนิดของไฟล์ และทำให้เพิ่มเติมความสามารถต่อไปได้ยาก

ไฟล์ข้อมูลภาพจะมีลักษณะสำคัญดังนี้

1.ข้อมูลในไฟล์จะมีหน่วยย่อยที่สุดเป็นไบท์

2.ข้อมูลในไฟล์จะถูกเข้ารหัสแบบ EDE

3.ลำดับของข้อมูลจะเรียงกันไปจากจุดภาพที่อยู่มุมบนซ้ายไปยังมุมบนขวา แล้วกลับมาเริ่มที่ด้านซ้ายอีกครั้ง ดังนั้นข้อมูลจึงมีลักษณะเดียวกันตลอดทั้งไฟล์

4.มีรหัสจบบรรทัดแบบเดียว คือ EOLR

5.นอกจากข้อมูลภาพและรหัสจบบรรทัดในไฟล์แล้ว ในตอนท้ายของไฟล์จะมีข้อมูลสั้นๆ ที่เรียกว่า ข้อมูลประจำไฟล์(TAG) เพื่อเป็นตัวบอกรายละเอียดของไฟล์นั้น ข้อมูลประจำไฟล์มีความยาว 6 ไบท์ โดยที่แต่ละไบท์จะมีความหมายดังนี้

ไบท์ที่ 1(Password) ใช้บอกว่าไฟล์นี้เป็นไฟล์ข้อมูลภาพแบบ IMG โดยที่ข้อมูลไบท์นี้จะมีค่าเป็น 11(OB_{16}) เมื่อไฟล์นี้เป็นไฟล์แบบ IMG เท่านั้น ซึ่งจะทำให้การตรวจสอบชนิดของไฟล์ข้อมูลภาพทำได้สะดวก เพียงแต่ตรวจข้อมูลไบท์ที่ 6 นับจากท้ายไฟล์ว่าเป็น OB_{16} หรือไม่เท่านั้น

ไบท์ที่ 2 และ 3(SizeZ) ใช้บอกขนาดของภาพในแนวแกน X โดยที่มีหน่วยเป็น จุดภาพ เหตุที่ต้องมี 2 ไบท์ก็เพื่อให้สามารถเก็บค่าได้ตั้งแต่ 0 ถึง 65535

ไบท์ที่ 4 และ 5(SizeY) ใช้บอกขนาดของภาพในแนวแกน Y โดยที่มีหน่วยเป็น จุดภาพ เหตุที่ต้องมี 2 ไบท์ก็เพื่อให้สามารถเก็บค่าได้ตั้งแต่ 0 ถึง 65535

ไบท์ที่ 6(Resolution) ใช้บอกความละเอียดภาพของไฟล์ข้อมูลภาพนั้น เนื่องจากความละเอียดภาพที่กำหนดไว้ 150 จุดต่อนิ้วนั้น มีค่าสูงเกินกว่าจะแสดงผลทางจอภาพและเครื่องพิมพ์รุ่นเก่าได้โดยสะดวก จึงได้มีการกำหนดค่าความละเอียดภาพไว้หลายค่า เพื่อให้สะดวกต่อการเลือกใช้ในโปรแกรมแสดงผล ความละเอียดภาพของข้อมูลที่เก็บไว้ในไฟล์จะหาได้จาก 150/Resolution ดังนี้

1.ถ้า Resolution มีค่าเป็น 1 ภาพที่เก็บไว้ในไฟล์จะมีความละเอียด 150 จุดต่อนิ้ว
2.ถ้า Resolution มีค่าเป็น 2 ภาพที่เก็บไว้ในไฟล์จะมีความละเอียด 75 จุดต่อนิ้ว
3.ถ้า Resolution มีค่าเป็น 3 ภาพที่เก็บไว้ในไฟล์จะมีความละเอียด 50 จุดต่อนิ้ว
4.ถ้า Resolution มีค่าเป็น 5 ภาพที่เก็บไว้ในไฟล์จะมีความละเอียด 30 จุดต่อนิ้ว

การกำหนดข้อมูลประจำไฟล์ลักษณะนี้ ทำให้สามารถคำนวณขนาดของภาพออกมาเป็นหน่วยความยาวในแต่ละแกนได้ โดยนำเอาความละเอียดภาพคูณด้วยจำนวนจุดภาพในแต่ละแกน จะได้ขนาดของภาพออกมามีหน่วยเป็นนิ้ว

5.2 หลักการทำงานของโปรแกรม

5.2.1 การรับข้อมูลจากสแกนเนอร์

สแกนเนอร์จะติดต่อกับไมโครคอมพิวเตอร์ผ่านทางอินเทอร์เฟสการ์ด ซึ่งเสียบอยู่ในสล็อตของเครื่องไมโครคอมพิวเตอร์ โดยได้กำหนดหมายเลขของพอร์ทไว้ดังนี้

-พอร์ท A หมายเลข $03EC_{16}$
-พอร์ท B หมายเลข $03ED_{16}$
-พอร์ท C หมายเลข $03EE_{16}$
-พอร์ทควบคุม หมายเลข $03EF_{16}$

ในการรับข้อมูลจากสแกนเนอร์ จะต้องมีการส่งรหัสควบคุมไปให้ 8255A ที่อยู่บนอินเทอร์เฟสการ์ด เพื่อให้ทำงานได้สอดคล้องกับ 8255A ที่อยู่บนสแกนเนอร์ รหัสควบคุมที่ต้องส่งไปนั้น จะเป็นกำหนดหน้าที่ให้กับพอร์ทต่างๆ และเป็นคำสั่งควบคุมการทำงานของ 8255A ให้ทำงานในโหมดที่ต้องการ(โหมด 0) ลักษณะการทำงานของ 8255A ในอินเทอร์เฟสการ์ด คือ

-ทำงานในโหมด 0
-พอร์ท A เป็นพอร์ทอินพุท
-พอร์ท C บนเป็นพอร์ทอินพุท
-พอร์ท C ล่างเป็นพอร์ทเอาท์พุท
-พอร์ท B ไม่ได้ใช้งาน สำรองไว้เพื่อการพัฒนาเพิ่มเติมในอนาคต

จากลักษณะการทำงานข้างต้น ทำให้สามารถกำหนดรหัสควบคุมที่จะต้องส่งให้กับ 8255A ได้เป็น 98_{16}

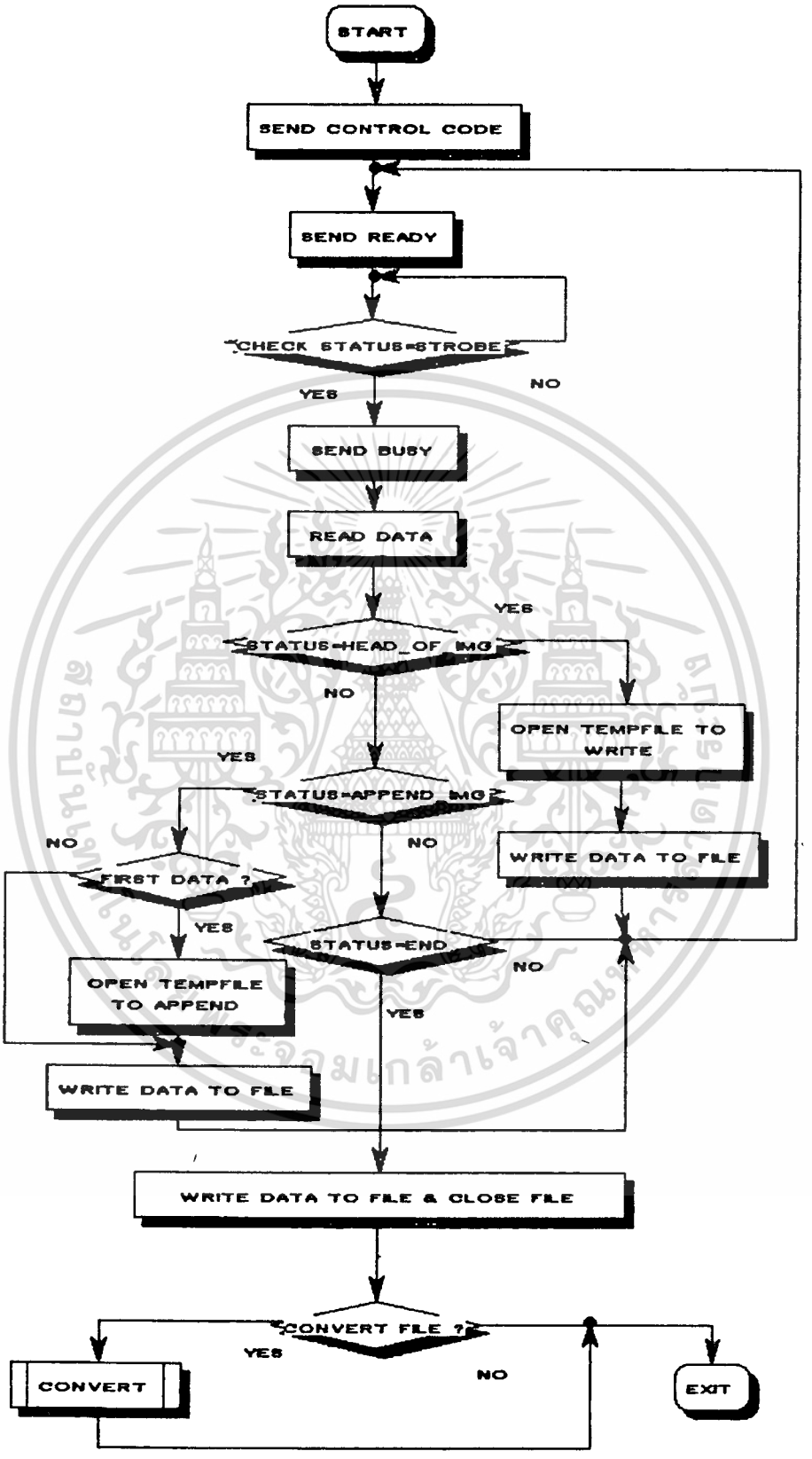
การติดต่อระหว่างเครื่องไมโครคอมพิวเตอร์กับสแกนเนอร์ จะเป็นไปตามรายละเอียดที่ได้กล่าวไว้แล้วในบทที่ 3 โดยที่เมื่อไมโครคอมพิวเตอร์รับข้อมูลมาแล้ว จะสร้างไฟล์ข้อมูลชั่วคราวขึ้นเพื่อเก็บข้อมูลที่ได้รับมา ชื่อของไฟล์ข้อมูลชั่วคราวนี้จะเก็บไว้ในตัวแปร TempFile ซึ่งโดยปกติจะใช้ชื่อ PICTURE.TMP แต่ผู้ใช้ก็สามารถเปลี่ยนแปลงชื่อไฟล์ข้อมูลชั่วคราวนี้ได้ตามต้องการด้วย

พิจารณาการทำงานของเครื่องไมโครคอมพิวเตอร์ จะเป็นไปตามขั้นตอนดังรูปที่ 5-1

5.2.2 การแปลงไฟล์ข้อมูลชั่วคราวเป็นไฟล์ข้อมูลภาพ

ไฟล์ข้อมูลชั่วคราวที่สร้างขึ้นในหัวข้อ 5.2.1 ไม่สามารถนำมาแสดงผลได้ ต้องนำมาแปลงรูปแบบและเพิ่มเติมข้อมูลประจำไฟล์ เพื่อให้กลายเป็นไฟล์ข้อมูลภาพเสียก่อน โดยหลักการสำคัญของการแปลงไฟล์ข้อมูลได้แก่ การจัดเรียงลำดับของข้อมูลภาพ และการสร้างข้อมูลประจำไฟล์ไว้ที่ส่วนท้ายของไฟล์

จากเดิมข้อมูลภาพในไฟล์ข้อมูลชั่วคราวจะมีสองลักษณะ คือ เรียงจากซ้ายไปขวา และจากขวาไปซ้ายสลับแถวกัน โปรแกรมในส่วนแปลงข้อมูลจะต้องทำการแปลงข้อมูลให้มีการเรียงจากซ้ายไปขวาเท่านั้น วิธีการแปลงข้อมูลทำได้โดยการพิจารณาแถวสแกนว่าเป็นแถวคู่หรือเป็นแถวคี่ หากเป็นแถวสแกนคู่(เริ่มนับแถวสแกนจากแถวที่ 0) ให้อ่านข้อมูลจากไฟล์ข้อมูลชั่วคราวมาเขียนลงไฟล์ข้อมูลภาพทีละไบท์ จนกระทั่งพบรหัส EOLR(มีค่าเป็น 0) ซึ่งแสดงว่าหมดแถวสแกนนั้นแล้ว และต้องเขียนรหัส EOLR ลงไปในไฟล์ข้อมูลภาพด้วย แล้วจึงเริ่มแปลงข้อมูลในแถวสแกนคี่ต่อไป ในแถวสแกนคี่จะต้องทำการสลับทิศทางของข้อมูล โดยการกำหนดตัวแปรอาเรย์ให้มีขนาดเท่ากับจำนวนจุดภาพสูงสุดในแต่ละแถวสแกน และให้ตัวชี้(Index)เริ่มต้นของอาเรย์มีค่าสูงสุด แล้วจึงอ่านข้อมูลจากไฟล์ข้อมูลชั่วคราวมาถอดรหัสทีละไบท์ ข้อมูลที่ถอดรหัสออกมาแล้วแต่ละตัวจะแทนจุดภาพแต่ละจุด แล้วนำข้อมูลจุดภาพนั้นเก็บลงในตัวแปรอาเรย์ทีละจุดภาพ โดยลดค่าตัวชี้จากสูงสุดลงมาเรื่อยๆ จนพบรหัส EOLL จึงหยุดอ่านข้อมูล หลังจากนั้นจะอ่านข้อมูลจากตัวแปรอาเรย์มาเข้ารหัสทีละจุดภาพ ด้วยวิธีเข้ารหัสแบบ EDE แล้วนำข้อมูลที่เข้ารหัสแล้วไปเขียนลงในไฟล์ข้อมูลภาพ การอ่านข้อมูลจะเริ่มอ่านข้อมูลจากอาเรย์ที่มีตัวชี้เป็น 0 แล้วจึงเพิ่มค่าตัวชี้ขึ้นเรื่อยๆ จนกระทั่งตัวชี้มีค่าสูงสุด แสดงว่าหมดแถวสแกนคี่แล้ว จึงเขียนรหัส EOLR ลงในไฟล์ข้อมูลภาพ แล้วจึงเริ่มอ่านข้อมูลจากไฟล์ข้อมูลชั่วคราวในแถวสแกนคู่ต่อไป ทำเช่นนี้สลับกันไปเรื่อยๆ จนกระทั่งหมดไฟล์ข้อมูลชั่วคราว ก็จะได้ไฟล์ข้อมูลภาพตามที่ต้องการ



รูปที่ 5-1: ผังการทำงานของฟังก์ชันรับข้อมูล

นอกจากการจัดเรียงลำดับข้อมูลภาพแล้ว หน้าที่สำคัญอีกประการหนึ่งในการแปลงไฟล์ข้อมูลชั่วคราวเป็นไฟล์ข้อมูลภาพ คือการสร้างข้อมูลประจำไฟล์ซึ่งจะเป็นข้อมูลที่บอกลักษณะของภาพ ดังรายละเอียดที่กล่าวไว้แล้วในหัวข้อ 5.1.2 เมื่อทำการเรียงลำดับข้อมูลจากไฟล์ข้อมูลชั่วคราวแล้ว โปรแกรมจะทำการสร้างข้อมูลประจำไฟล์โดยที่

-ข้อมูลที่เป็น Password จะมีค่าเป็น $0B_{16}$ เสมอ

-ข้อมูลที่เป็น Resolution จะมีค่าเป็น 1 ซึ่งหมายถึงความละเอียดภาพ 150 จุดต่อนิ้ว ทั้งนี้เนื่องจากในขั้นตอนการแปลงไฟล์นี้ ไฟล์ข้อมูลชั่วคราวซึ่งได้จากสแกนเนอร์จะมีความละเอียดภาพเป็น 150 จุดเสมอ ค่า Resolution ที่เป็นค่าอื่นนอกจาก 1 จะใช้ในกรณีที่มีการแก้ไขภาพแล้วเท่านั้น

-ข้อมูลที่เป็น SizeX จะได้จากการถอดรหัสข้อมูลในแถวสแกนแถวแรก แล้วนับจำนวนจุดภาพที่ได้ ซึ่งจะถือว่าภาพที่ได้จากสแกนเนอร์จะมีขนาดกว้างเท่ากันตลอดทั้งภาพ

-ข้อมูลที่เป็น SizeY จะได้จากการนับแถวสแกนทั้งแถวคู่และแถวคี่ โดยการนับจำนวน EOLL และ EOLR ในขณะที่ทำการเรียงลำดับข้อมูล

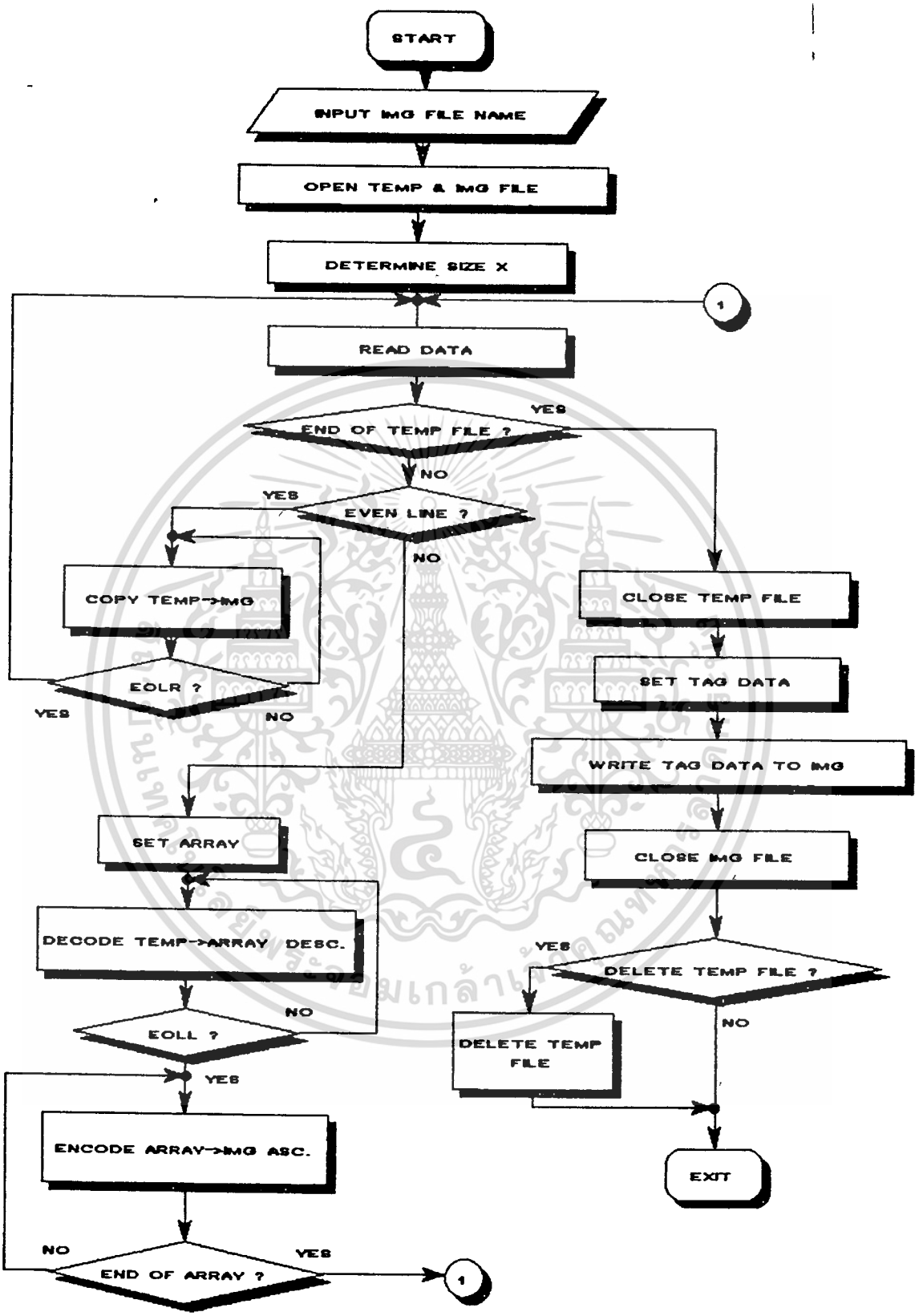
เมื่อสร้างข้อมูลประจำไฟล์ ทั้ง 6 ไบท์แล้ว จึงจะนำไปเขียนต่อท้ายข้อมูลภาพแล้วปิดไฟล์ ซึ่งจะทำให้ข้อมูลประจำไฟล์อยู่ที่ท้ายไฟล์ เพื่อความสะดวกในการตรวจสอบภายหลัง

การทำงานของโปรแกรมในส่วนที่ทำหน้าที่แปลงไฟล์ข้อมูลชั่วคราวเป็นไฟล์ข้อมูลภาพ เป็นไปตามรูปที่ 5-2

5.2.3 การนำภาพออกแสดงผลทางจอภาพ

การนำไฟล์ข้อมูลภาพออกแสดงผลทางหน้าจอ มีข้อควรพิจารณาอยู่หลายประการ คือ ประการแรก การแสดงผลออกทางจอภาพของเครื่องไมโครคอมพิวเตอร์ ซึ่งโดยทั่วไปจะมีการแสดงผลได้ 2 ลักษณะคือ โหมดตัวอักษร(Text Mode) และกราฟิกโหมด(Graphics Mode) ในโหมดตัวอักษรนั้นจะสามารถแสดงผลได้เฉพาะตัวอักษรหรือสัญญลักษณ์พิเศษที่มีขนาดคงที่เท่านั้น ไม่สามารถแสดงผลในลักษณะของรูปภาพได้ ซึ่งต่างจากการแสดงผลในกราฟิกโหมด

ในกราฟิกโหมดจะแบ่งจอภาพทั้งหมดออกเป็นจุดภาพย่อยๆ แต่ละจุดภาพสามารถควบคุมได้โดยอิสระไม่ขึ้นต่อกัน จำนวนของจุดภาพและความสามารถในการแสดงสีของจุดภาพ จะแตกต่างกันไปตามชนิดของจอภาพและการ์ดควบคุมการแสดงผล ตัวอย่างลักษณะและความสามารถในการแสดงผลของจอภาพและการ์ดควบคุมการแสดงผลที่นิยมใช้งานบางชนิด แสดงดังตารางที่ 5-1 ในหน้า 68



รูปที่ 5-2: ผังการทำงานของฟังก์ชันแปลงไฟล์ข้อมูลภาพ

จอภาพ	การ์ดควบคุม	โหมดตัวอักษร	กราฟิกโหมด	จำนวนสี
Monochrome	Hercules	80 x 25	720 x 350	2
Enhanced Color	EGA	40 x 25	640 x 200	16
			640 x 350	16
VGA Color	VGA	40 x 25	320 x 480	256
		80 x 25	640 x 480	256

ตารางที่ 5.1: ความสามารถในการแสดงผลของจอภาพและการ์ดควบคุมชนิดต่างๆ

ในจอภาพและการ์ดควบคุมบางชนิดจะสามารถปรับการทำงานได้ในหลายลักษณะ และยังอาจปรับให้มีความสามารถเหมือนกับจอภาพและการ์ดควบคุมชนิดอื่นได้อีกด้วย รายละเอียดเกี่ยวกับจอภาพและการ์ดควบคุมการแสดงผลสามารถหาเพิ่มเติมได้จาก [6]

ในการใช้เครื่องไมโครคอมพิวเตอร์เพื่อนำภาพจากไฟล์ข้อมูลภาพมาแสดงผล ต้องกำหนดให้จอภาพทำงานในกราฟิกโหมด ในโปรแกรมใช้งานนี้ได้กำหนดให้ใช้จอภาพและการ์ดควบคุมการแสดงผลเป็นชนิด EGA(Enhanced Graphics Adapter) ซึ่งมีความละเอียดของการแสดงผลในกราฟิกโหมดสูงสุดเป็น 640 จุดภาพในแนวนอน และ 350 จุดภาพในแนวตั้ง(เมื่อใช้กับจอภาพแบบ Enhanced Color Display) เหตุที่เลือกความละเอียดค่านี้ เพื่อให้สะดวกต่อการใช้งานกับไมโครคอมพิวเตอร์ได้กว้างขวางขึ้น เพราะจอภาพและการ์ดควบคุมการแสดงผลแบบ VGA ก็สามารถปรับให้ทำงานในลักษณะเช่นเดียวกับการ์ดควบคุมการแสดงผลแบบ EGA ได้เช่นกัน

พิจารณาความละเอียดภาพที่ได้จากการสแกน มีความละเอียดภาพ 150 จุดต่อนิ้ว ซึ่งหมายความว่า ภาพขนาด 1 ตารางนิ้ว จะมีจุดภาพ 150 x 150 จุดภาพ เมื่อนำมาแสดงผลในจอภาพโดยให้ 1 จุดภาพบนจอภาพแทน 1 จุดภาพสแกน จะทำให้จอภาพสามารถแสดงภาพได้สูงสุดใน 1 จอภาพได้เพียง 4.27 นิ้ว x 2.33 นิ้ว หากภาพที่นำมาสแกนมีขนาดใหญ่กว่านี้จะไม่สามารถแสดงผลได้ทั้งภาพในจอภาพเดียว วิธีแก้ปัญหาดังกล่าวทำได้โดยการปรับการแสดงผลให้ 1 จุดภาพบนจอภาพแทนจุดภาพมากกว่า 1 จุดภาพจากภาพที่สแกน

จากหลักการดังกล่าวทำให้ต้องกำหนดความละเอียดภาพเพิ่มเติมจาก 150 จุดภาพ โดยกำหนดให้ความละเอียดภาพที่จะนำมาแสดงผลมี 4 ค่าคือ 150,75,50 และ 30 จุดภาพ ดังนั้น ภาพที่ปรากฏบนจอภาพ 1 จุดภาพจะแทนภาพจริง 1,2,3 และ 5 จุดภาพตามลำดับ

ความสามารถในการแสดงภาพใน 1 จอภาพ จะเป็นดังนี้

-ที่ 150 จุดภาพต่อนิ้ว แสดงได้สูงสุด 4.27 นิ้ว x 2.33 นิ้ว
-ที่ 75 จุดภาพต่อนิ้ว แสดงได้สูงสุด 8.54 นิ้ว x 4.66 นิ้ว
-ที่ 30 จุดภาพต่อนิ้ว แสดงได้สูงสุด 12.80 นิ้ว x 7.00 นิ้ว
-ที่ 50 จุดภาพต่อนิ้ว แสดงได้สูงสุด 21.30 นิ้ว x 11.60 นิ้ว

ความละเอียดภาพ เป็นข้อมูลที่สำคัญตัวหนึ่งที่ถูกเก็บไว้ในส่วนข้อมูลประจำไฟล์ ของไฟล์ข้อมูลภาพ เพื่อบอกว่าภาพที่เก็บไว้นั้นมีความละเอียดภาพเป็นเท่าใด

ในการใช้งานจริง ภาพที่นำมาสแกนอาจจะมีขนาดแตกต่างกันไปตั้งแต่ขนาดเล็กไปจนถึงขนาดใหญ่ ดังนั้นในโปรแกรมใช้งานส่วนที่ทำหน้าที่เกี่ยวกับการแสดงผลทางจอภาพ จึงมีการกำหนดความละเอียดภาพที่จะแสดงผลทางจอภาพ โดยที่จะเก็บไว้ในตัวแปร CurRes เพื่อปรับให้จอภาพแสดงภาพให้เหมาะสมกับขนาดจริงของภาพ

หลักการทำงานของโปรแกรมในส่วนแสดงผลทางจอภาพ มีขั้นตอนดังนี้

1.อ่านไฟล์ที่ต้องการแสดงผล ตรวจสอบข้อมูลประจำไฟล์ ซึ่งเป็นข้อมูล 6 ไบท์สุดท้ายของไฟล์ เพื่อดูว่าไฟล์ดังกล่าวเป็นไฟล์ข้อมูลภาพแบบ IMG หรือไม่ และดูขนาดของภาพตลอดจนความละเอียดภาพ

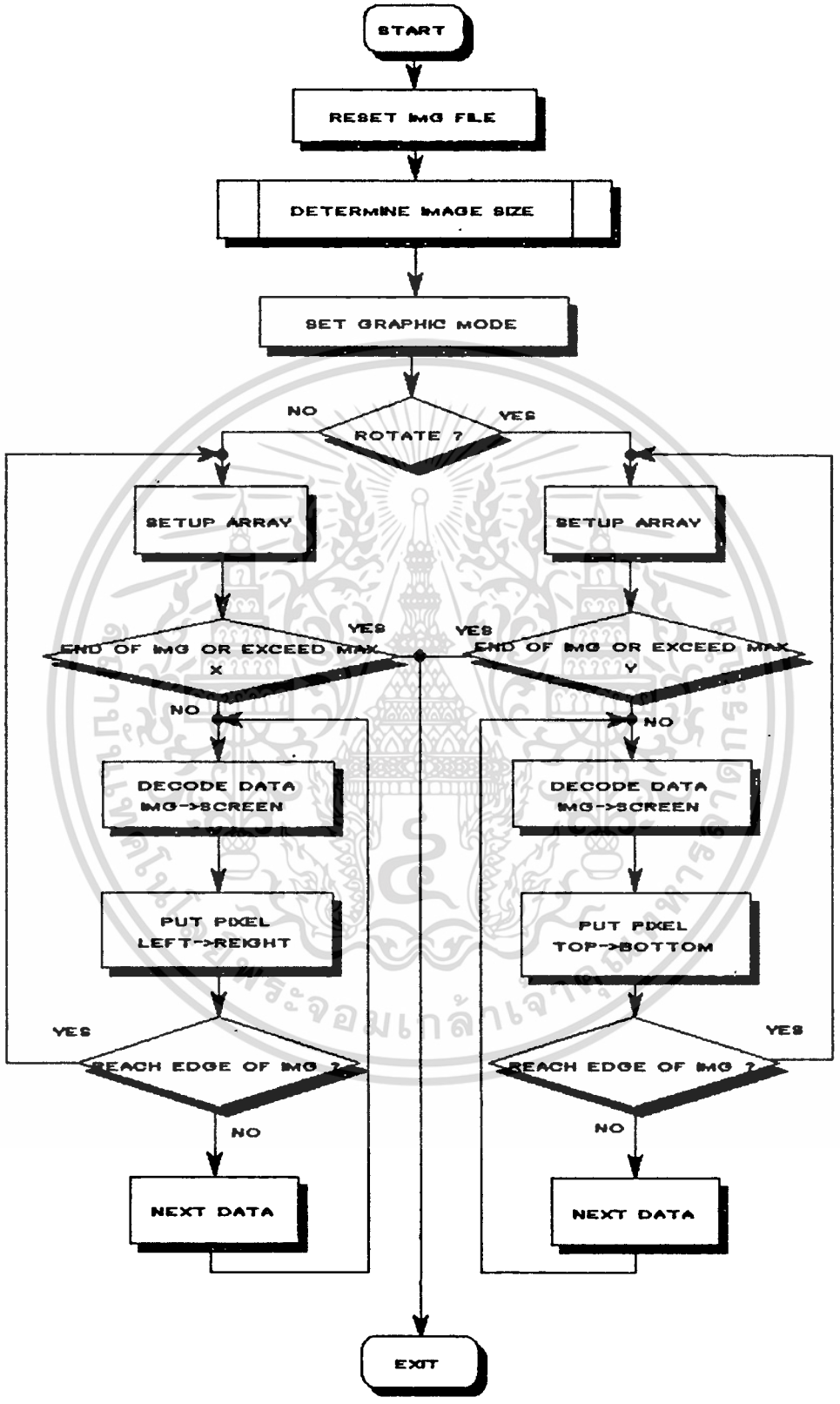
2.นำข้อมูลประจำไฟล์ มาคำนวณหาขนาดที่แท้จริงของภาพในหน่วยตารางนิ้ว ซึ่งขนาดของภาพหาได้จากความสัมพันธ์ดังนี้

-ขนาดทางแกน X เท่ากับ ข้อมูล SizeX คูณกับ ข้อมูลความละเอียดภาพ
-ขนาดทางแกน Y เท่ากับ ข้อมูล SizeY คูณกับ ข้อมูลความละเอียดภาพ

3.เมื่อได้ขนาดของภาพ จึงนำมาคำนวณเปรียบเทียบกับค่าความละเอียดภาพในการแสดงผล ซึ่งเก็บไว้ในตัวแปร CurRes เพื่อตรวจสอบว่า ภาพที่มีขนาดดังกล่าวสามารถแสดงได้ทั้งหมดภายในจอภาพเดียวหรือไม่ หากไม่สามารถแสดงได้ก็จะเตือนให้ผู้ใช้ทราบและเปิดโอกาสให้ผู้ใช้เลือกปรับความละเอียดในการแสดงผลได้ การตรวจสอบนี้จะทำทั้ง 2 แกนของภาพ และยังเปิดโอกาสให้ผู้ใช้เลือกว่าจะให้แสดงภาพแบบที่มีการหมุน 90 องศาหรือไม่อีกด้วย

4.แสดงภาพออกทางจอภาพ โดยการอ่านข้อมูลจากไฟล์ทีละไบท์ นำมาถอดรหัส ข้อมูลที่ถอดรหัสแล้วจะแทนจุดภาพแต่ละจุด จากนั้นจึงนำจุดภาพดังกล่าวไปแสดงออกทางจอภาพ ตามเงื่อนไขความละเอียดภาพและทิศทางการหมุนภาพที่กำหนดไว้แล้ว

การทำงานของโปรแกรมในส่วนแสดงภาพออกทางจอภาพ แสดงดังรูปที่ 5-3



รูปที่ 5-3: ผังการทำงานของฟังก์ชันแสดงภาพบนจอภาพ

5.2.4 การพิมพ์ภาพออกทางเครื่องพิมพ์

เครื่องพิมพ์ที่ใช้ในโครงงานนี้ เป็นเครื่องพิมพ์แบบด็อทเมตริกซ์(Dot matrix printer) ที่ใช้งานกับเครื่องไมโครคอมพิวเตอร์โดยทั่วไป เครื่องพิมพ์ที่ใช้ในปัจจุบันมีมากมายหลายยี่ห้อ และหลายรุ่น การเขียนโปรแกรมเพื่อควบคุมการทำงานของเครื่องพิมพ์จึงจำเป็นต้องอ้างอิงเครื่องพิมพ์รุ่นใดรุ่นหนึ่งไว้ เพื่อความสะดวกในการเขียนโปรแกรม ในโครงงานนี้เลือกใช้เครื่องพิมพ์ EPSON รุ่น LX-86 เป็นเครื่องพิมพ์อ้างอิง แต่ก็สามารถใช้กับเครื่องพิมพ์รุ่นที่ใกล้เคียงกันได้ด้วย

การทำงานของเครื่องพิมพ์ก็คล้ายคลึงกับจอภาพ คือ แบ่งการทำงานออกเป็น 2 โหมดคือ โหมดตัวอักษรและกราฟิกโหมด ในโหมดตัวอักษรนั้น ตัวอักษรที่จะพิมพ์ออกทางเครื่องพิมพ์ จะถูกบันทึกไว้ในหน่วยความจำถาวรภายในเครื่องพิมพ์โดยผู้ผลิตแล้ว ไม่สามารถเปลี่ยนแปลงได้ ทำให้ไม่สามารถพิมพ์ภาพออกมาในโหมดตัวอักษรได้ ต้องพิมพ์ในกราฟิกโหมดเท่านั้น

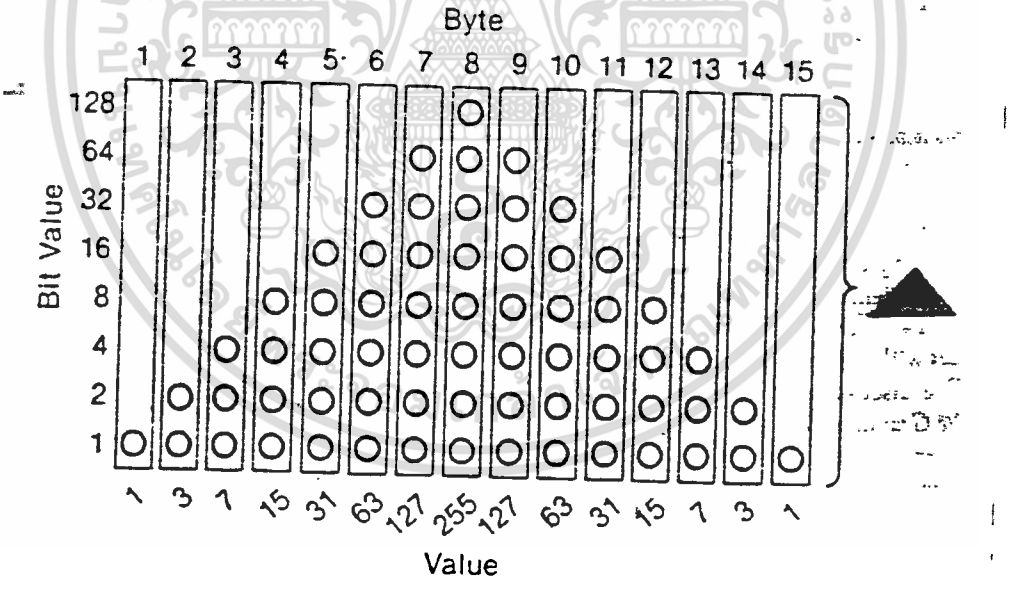
โดยปกติตัวเครื่องพิมพ์จะทำงานในโหมดตัวอักษรเท่านั้น นอกจากจะมีการส่งคำสั่งไปเปลี่ยนแปลงการทำงานของเครื่องพิมพ์ให้เป็นแบบกราฟิกโหมด และในการทำงานในโหมดกราฟิกของเครื่องพิมพ์ยังแบ่งออกได้อีกหลายโหมด โดยที่แต่ละโหมดจะมีความละเอียดในการพิมพ์และความเร็วในการพิมพ์ที่แตกต่างกัน นอกจากนี้เครื่องพิมพ์แต่ละรุ่นแต่ละยี่ห้อก็มีจำนวนโหมดไม่เท่ากันด้วย

พิจารณารูปแบบของการส่งคำสั่งให้แก่เครื่องพิมพ์ในกราฟิกโหมด จะเริ่มด้วยรหัสคำสั่งซึ่งกำหนดโหมดกราฟิกของเครื่องพิมพ์(รวมทั้งความละเอียดในการพิมพ์ด้วย) ตามด้วยจำนวนไบท์ของข้อมูลกราฟิก และตามด้วยข้อมูลกราฟิกเท่ากับจำนวนที่ระบุไว้ แต่ไม่เกิน 1 บรรทัด และเมื่อพิมพ์ข้อมูลกราฟิกไปเท่ากับจำนวนที่ระบุไว้แล้ว เครื่องพิมพ์จะกลับมาทำงานในโหมดตัวอักษรโดยอัตโนมัติโดยไม่จำเป็นต้องมีคำสั่งให้กลับไปทำงานในโหมดตัวอักษร ดังนั้นการส่งข้อมูลเพื่อพิมพ์ภาพกราฟิกจึงต้องระวังเรื่องจำนวนข้อมูลเป็นพิเศษ เนื่องจากหากส่งข้อมูลไปเกินจำนวนที่ระบุไว้ตอนแรก จะทำให้ข้อมูลที่เกินมาถูกพิมพ์ออกไปในโหมดตัวอักษรได้

พิจารณาคำสั่งควบคุมเครื่องพิมพ์ โดยปกติคำสั่งที่ใช้ควบคุมเครื่องพิมพ์จะมี 2-3 ไบท์ และไบท์แรกจะมีค่าเป็น 27 เสมอ ซึ่งตรงกับรหัส ESC ในรหัสแอสกี(ASCII) จึงเรียกคำสั่งควบคุมเครื่องพิมพ์ว่า Escape code ในกรณีที่สั่งให้เครื่องพิมพ์ทำงานในกราฟิกโหมด คำสั่งจะนำหน้าด้วยรหัส ESC ตามด้วยข้อมูล 1 ไบท์ ที่ใช้ระบุกราฟิกโหมดที่ต้องการ ข้อมูล 2 ไบท์ต่อมาจะบอกจำนวนข้อมูลกราฟิกที่ส่งตามมาเพื่อพิมพ์ออกมาในกราฟิกโหมด โดยที่ไบท์แรกจะต้องเป็นข้อมูลนัยสำคัญต่ำกว่าไบท์หลัง วิธีง่ายๆในการคำนวณหาตัวเลขทั้ง 2 ไบท์นี้ ทำได้ดังนี้

จำนวนข้อมูล(ฐาน 10)/256 ---> เมื่อปัดเศษผลหารแล้วจะได้ข้อมูลไบท์สูงซึ่งเป็นไบท์ที่ 2 ---> เศษที่ได้เป็นข้อมูลไบท์ต่ำ ส่งไปเป็นไบท์แรก

หลังจากส่งตัวเลขบอกจำนวนข้อมูลกราฟิกแล้ว จึงจะส่งข้อมูลกราฟิกออกมาพิมพ์ได้ ข้อมูลกราฟิกที่จะส่งออกมาพิมพ์นี้จะต้องอยู่ในรูปของบิทแพทเทิร์น(Bit pattern) ซึ่งหมายความว่า ข้อมูลแต่ละบิทในไบท์นั้น จะแทนจุดแต่ละจุดในแนวตั้ง ซึ่งจุดแต่ละจุดนี้ก็คือหัวพิมพ์แต่ละหัวนั่นเอง โดยที่จุดต่ำสุดจะมีค่าประจำจุดเป็น 1 และจุดสูงสุดจะมีค่าประจำจุดเป็น 128 เช่น รูปที่ 5-4 ซึ่งแสดงข้อมูลบิทแพทเทิร์นจำนวน 15 ไบท์ซึ่งประกอบกันเป็นรูปปิรามิด จะเห็นว่าหากต้องการให้จุดใดในจำนวน 8 จุดถูกพิมพ์ จะต้องนำ 1 ไปคูณกับค่าประจำบิทนั้น และหากไม่ต้องการพิมพ์บิทใด ก็นำ 0 ไปคูณกับค่าประจำบิทนั้น แล้วจึงนำผลคูณของค่าประจำบิททุกบิทมารวมกัน กลายเป็นข้อมูลกราฟิกซึ่งจะต้องส่งให้เครื่องพิมพ์ ด้วยวิธีนี้ทำให้สามารถควบคุมการปรากฏของจุดภาพใดๆได้ตามต้องการ



รูปที่ 5-4: ตัวอย่างข้อมูลแบบบิทแพทเทิร์นซึ่งพิมพ์ออกมาเป็นรูปปิรามิด

ในกราฟิกโหมด แบ่งเป็นโหมดต่างๆได้หลายโหมด ขึ้นอยู่กับชนิดของเครื่องพิมพ์ ในเครื่องพิมพ์ที่ใช้อ้างอิงมีโหมดการทำงานโหมดต่างๆ แสดงดังตารางที่ 5-2

รหัสควบคุม	ชื่อโหมดการทำงาน	จำนวนจุด(ไบท์)ต่อแถว	หมายเหตุ
27,75	Normal	480	
27,76	Double density	960	
27,89	High speed double density	960	ไม่สามารถพิมพ์จุดติดกันได้
27,90	Quaruple density	1920	พิมพ์ 1 จุดต้องเว้น 2 จุด

ตารางที่ 5-2: โหมดต่างๆในการพิมพ์แบบกราฟิกของเครื่องพิมพ์

ในการพัฒนาโปรแกรมใช้งานนี้ ใช้กราฟิกโหมดแบบ Double density ซึ่งสามารถพิมพ์ได้ 960 จุดต่อหนึ่งบรรทัด โดยสามารถพิมพ์จุดที่ติดกันได้ด้วย เนื่องจากเครื่องพิมพ์ในรุ่นนี้มีความกว้างของแคร่พิมพ์เป็น 8 นิ้ว ดังนั้นความละเอียดภาพสูงสุดในแนวนอน ที่สามารถพิมพ์ออกทางเครื่องพิมพ์ได้จึงเท่ากับ

960/8 = 120 จุดต่อนิ้ว

ส่วนความละเอียดของภาพในแนวตั้ง สามารถพิจารณาได้จากปัจจัย 2 ประการคือ ระยะห่างระหว่างหัวเข็มแต่ละหัว และความสามารถในการเลื่อนหน้ากระดาษ ซึ่งปัจจัยแรกเราไม่สามารถควบคุมได้ แต่สามารถควบคุมการเลื่อนกระดาษได้ โดยการกำหนดระยะห่างระหว่างบรรทัดให้เหมาะสม การสั่งให้เครื่องพิมพ์เลื่อนกระดาษทำได้โดยส่งคำสั่งขึ้นบรรทัดใหม่(Line feed) ซึ่งโดยปกติมักจะส่งตามหลังคำสั่งจบบรรทัด(Carriage return) คำสั่งจบบรรทัดจะทำให้หัวพิมพ์เลื่อนกลับไปที่ขอบซ้ายของกระดาษ ส่วนคำสั่งขึ้นบรรทัดใหม่จะเลื่อนกระดาษขึ้นเป็นระยะเท่ากับที่เราได้กำหนดไว้ก่อนแล้ว

เครื่องพิมพ์รุ่นนี้สามารถกำหนดระยะเลื่อนกระดาษได้ละเอียดที่สุดเพียง 1/72 นิ้ว ซึ่งก็หมายความว่าความละเอียดภาพสูงสุดในแนวตั้งที่สามารถพิมพ์ได้ คือ 72 จุดต่อนิ้ว และจากการทดลองพบว่าหากกำหนดให้ระยะเลื่อนกระดาษเป็น 8/72 นิ้ว และสั่งพิมพ์โดยให้หัวเข็มทุกหัวทำงานพร้อมกันหมด จะได้ภาพที่ต่อกันสนิทโดยไม่มีช่วงเว้นระหว่างบรรทัดเลย จึงสามารถสรุปได้ว่าหัวเข็มทั้ง 8 หัวนั้น แต่ละหัวอยู่ห่างกัน 1/72 นิ้ว

จะเห็นว่าความละเอียดภาพของเครื่องพิมพ์ในแนวตั้ง และในแนวนอนมีค่าไม่เท่ากัน และค่าความละเอียดภาพของเครื่องพิมพ์ ไม่สามารถจัดให้เป็นอัตราส่วนลงตัวกับความละเอียดภาพที่สแกนได้ พิจารณาอัตราส่วนระหว่างภาพจากเครื่องพิมพ์และภาพจริง

ในแกนนอน: 120/150 หรือ 4/5

ในแกนตั้ง : 72/150 หรือ 12/25

การที่อัตราส่วนของภาพไม่เท่ากัน มีผลให้ต้องมีการพิจารณาแปลงข้อมูลในแกนนอน ข้อมูลภาพในแกนนอน 5 จุดจะต้องถูกพิมพ์ออกเพียง 4 จุด และในแกนตั้งข้อมูลภาพ 25 จุด จะถูกพิมพ์ออกเพียง 12 จุดเท่านั้น วิธีในการแทนข้อมูลภาพจะมีผลต่อคุณภาพของภาพเป็นอย่างมาก วิธีการแทนข้อมูลที่ง่ายที่สุด คือการตัดข้อมูลจุดที่เกินมาทิ้งไป ดังนั้นในโปรแกรมพิมพ์ข้อมูลจึงต้องทำหน้าที่สำคัญสองประการ คือ การจัดการกับจำนวนจุดภาพที่จะพิมพ์ เพื่อให้ภาพที่พิมพ์ออกมามีขนาดและลักษณะเหมือนกับต้นฉบับ ประการที่สอง คือ การส่งข้อมูลออกไปยังเครื่องพิมพ์ จะต้องทำการจัดข้อมูลจำนวน 8 แถวสแกนให้อยู่ในรูปบิทแพทเทิร์น แล้วส่งออกไปทีละไบท์

ในการทำงานของโปรแกรม จำเป็นต้องกำหนดตัวแปรชั่วคราวขึ้นมาในรูปของอาเรย์สองมิติขนาด 960 ไบท์ x 8 ไบท์ เพื่อใช้แทนจุดภาพที่จะพิมพ์ออกทางเครื่องพิมพ์ การทำงานของโปรแกรมเริ่มจากการอ่านไฟล์ข้อมูลภาพ แล้วนำข้อมูลมาถอดรหัส จากนั้นต้องนำไปเก็บไว้ในตัวแปรอาเรย์ด้วยอัตราส่วนที่กำหนดไว้แล้ว คือ ในแนวนอน จุดภาพที่ถอดรหัสมา 5 จุดภาพ จะนำไปเก็บในอาเรย์เพียง 4 จุดภาพ ส่วนจุดภาพที่ 5 จะไม่นำมาพิจารณาเลย ส่วนในแนวตั้งจะนำแถวสแกนของจุดภาพมาบันทึกไว้ในอาเรย์เพียงแถวเว้นแถวเท่านั้น

เมื่อเก็บค่าจุดภาพลงในอาเรย์จนเต็มแล้ว จึงนำอาเรย์นั้นมาแปลงค่าให้เป็นข้อมูลแบบบิทแพทเทิร์นจำนวน 960 ไบท์ เพื่อส่งให้กับเครื่องพิมพ์ต่อไป เมื่อส่งข้อมูลไปครบทั้ง 960 ไบท์แล้วจึงจะส่งรหัสจบบรรทัดและรหัสขึ้นบรรทัดใหม่ตามไป เนื่องจากการทำงานในกราฟิกโหมดของเครื่องพิมพ์ถูกจำกัดด้วยจำนวนข้อมูลซึ่งระบุไว้ตอนต้นคำสั่ง (ในที่นี้จำนวนข้อมูลเป็น 960 ไบท์) เมื่อส่งข้อมูลไปในแบบบิทแพทเทิร์นครบ 960 ไบท์แล้ว เครื่องพิมพ์จะกลับมาทำงานในโหมดตัวอักษรโดยอัตโนมัติ ดังนั้นเมื่อจะเริ่มพิมพ์ภาพบรรทัดต่อไป จึงต้องมีการส่งคำสั่งกำหนดโหมดกราฟิกและจำนวนข้อมูลไปอีกเสมอจึงจะสามารถพิมพ์บรรทัดต่อไปได้

การติดต่อกับเครื่องพิมพ์ อาจเกิดเหตุผิดพลาดขึ้นได้ เช่น เครื่องพิมพ์ไม่พร้อม หรือกระดาษหมด ซึ่งเหตุผิดพลาดเหล่านี้อาจเกิดขึ้นระหว่างการพิมพ์ โปรแกรมในส่วนนี้จึงมีการจัดการเกี่ยวกับข้อผิดพลาดที่อาจเกิดขึ้น โดยตรวจสอบสถานะของเครื่องพิมพ์ทุกครั้งที่ส่งข้อมูลไปให้เครื่องพิมพ์ แล้วนำสถานะของเครื่องพิมพ์มาวิเคราะห์ว่าเกิดเหตุผิดพลาดหรือไม่ เพื่อจะได้แจ้งให้แก่ผู้ใช้ทราบต่อไป

5.2.5 การแก้ไขไฟล์ข้อมูลภาพและการแปลงไฟล์ข้อมูลภาพให้เป็นมาตรฐานพีซีเอ็กซ์

5.2.5.1 การแก้ไขไฟล์ข้อมูลภาพ

ทำได้โดยการนำไฟล์ข้อมูลภาพออกแสดงทางจอภาพ เพื่อให้ผู้ใช้แก้ไขภาพบนจอภาพ เมื่อผู้ใช้แก้ไขภาพเรียบร้อยแล้ว จึงอ่านข้อมูลจากหน่วยความจำสำหรับการแสดงผล(Display memory) มาเข้ารหัสและสร้างเป็นไฟล์ข้อมูลภาพใหม่ โดยที่หลักการในการอ่านข้อมูลและเข้ารหัสข้อมูลภาพจะมีลักษณะการทำงานในส่วนของการเข้ารหัสจะเหมือนกับโปรแกรมมอนิเตอร์ในบอร์ดควบคุม แต่ไฟล์ที่สร้างขึ้นจะเก็บข้อมูลในรูปแบบที่เป็นไฟล์ข้อมูลภาพแบบ IMG

ในส่วนของการแก้ไขภาพบนจอภาพ ทำได้โดยการสร้างตัวชี้ตำแหน่งขึ้นเพื่อเป็นเครื่องมือช่วยในการแก้ไขภาพ ผู้ใช้สามารถเลื่อนตัวชี้ตำแหน่งไปมายังจุดต่างๆ บนจอภาพได้ ตัวชี้ตำแหน่งจะทำหน้าที่ได้ 3 ลักษณะคือ

1. เป็นจุดบอกตำแหน่ง คือ ทำหน้าที่บอกตำแหน่งเพียงอย่างเดียว ซึ่งเมื่อมีการเคลื่อนที่ผ่านส่วนต่างๆ ของจอภาพจะไม่มีผลกระทบต่อภาพที่แสดงอยู่บนจอ

2. เป็นเครื่องมือวาดภาพ การทำงานหน้าที่นี้ ตัวชี้ตำแหน่งจะเปรียบเสมือนปลายปากกาที่สร้างจุดภาพขึ้นทุกตำแหน่งที่เคลื่อนผ่าน

3. เป็นเครื่องมือลบภาพ การทำงานหน้าที่นี้ ตัวชี้ตำแหน่งจะเปรียบเสมือนปลายยางลบ คือเมื่อเลื่อนตัวชี้ตำแหน่งไป จุดภาพที่อยู่ในตำแหน่งเดียวกับตัวชี้จะถูกลบออกจากจอภาพ

จากหน้าที่ทั้ง 3 ลักษณะนี้เอง ทำให้ผู้ใช้สามารถแก้ไขภาพได้

ขั้นตอนสุดท้ายของการแก้ไขไฟล์ข้อมูลภาพ คือ การแปลงภาพจากจอภาพที่แก้ไขแล้วให้เป็นไฟล์ข้อมูลภาพใหม่ วิธีที่สะดวกคือ พิจารณาว่าจอภาพทั้งจอเป็นภาพทั้งภาพ ดังนั้นในกรณีที่ภาพที่นำมาแก้ไขมีขนาดใหญ่เกินกว่าจะแสดงได้ทั้งภาพพร้อมกัน เมื่อถึงขั้นตอนนี้จะเกิดปัญหาขึ้นเนื่องจากส่วนของภาพที่เกินจอภาพไป จะไม่ถูกนำมาเก็บในไฟล์ข้อมูลใหม่ด้วย หรือในกรณีที่นำภาพมาแสดงผลด้วยความละเอียดภาพค่าอื่นที่ไม่ใช่ 150 จุดต่อนิ้ว เมื่อทำการแปลงไฟล์ข้อมูลจะทำให้ได้ไฟล์ข้อมูลภาพใหม่ที่มีขนาดผิดเพี้ยนไปจากขนาดจริง

5.2.5.2 การแปลงไฟล์ข้อมูลภาพเป็นมาตรฐานพีซีเอ็กซ์

รูปแบบไฟล์ข้อมูลภาพมาตรฐานพีซีเอ็กซ์ ใช้วิธีการเข้ารหัสข้อมูลภาพแบบ RLE ซึ่งได้กล่าวถึงรายละเอียดไปแล้วในบทที่ 2 โดยที่ไฟล์พีซีเอ็กซ์จะขึ้นต้นด้วยข้อมูลที่บอกลักษณะต่างๆ ของภาพเป็นจำนวน 128 ไบท์ ดังตารางที่ 5-3

รายละเอียดเกี่ยวกับไฟล์พีซีเอ็กซ์ ผู้สนใจสามารถหาเพิ่มเติมได้จาก [8]

ไบต์ที่	ขนาด (Byte)	ชื่อข้อมูล	รายละเอียด
0	1	Password	ค่า '0aH' = ไฟล์ .PCX
1	1	Version	เก็บค่า Version (รุ่น) ของ PC Paintbrush ดังนี้
			0=Version 2.5
			2=Version 2.8 with Palette
			3=Version 2.8 without Palette
			5=Version 3.0
2	1	Encoding	ค่า = 1
3	1	Bits per Pixel	จำนวนของบิตที่ใช้แสดง
			1 Pixel จาก 1 Plane (ระนาบสี)
			1 = EGA, VGA, or HERC
			4 = CGA
4	8	Windows-Dimemsions	ค่า integer 4 ค่า (ค่าละ 2 Byte) ให้ค่ามุมบนซ้ายและมุมล่างขวาของภาพ กำหนดให้รูปแบบ x1, y1, x2, y2
12	2	Horizontal-Resolution	ความละเอียดของการแสดงภาพในแนวนอน 640 = EGA, VGA
			320 = CGA
			720 = HERCULES
14	2	Vertical-Resolution	ความละเอียดของการแสดงภาพในแนวตั้ง 480 = VGA
			350 = EGA
			200 = CGA
			348 = HERC
16	48	Color Map	ข้อมูล Color Palette ดูรูป 2
64	1	Reserved	
65	1	No. of Plane	จำนวนของ Planes ที่ใช้แสดงภาพ
66	1	Byte per Line	จำนวนไบต์ต่อการสแกน 1 บรรทัด
68	2	Palette Info	How to interpret the palette
70	8	Maximum x value	Special for fractal files
78	8	Minimum x value	Special for fractal files
86	8	Maximum y value	Special for fractal files
94	8	Minimum y value	Special for fractal files
102	8	P Value	Special for fractal files
110	8	Q Value	Special for fractal files
118	10	Not used	

หมายเหตุ ข้อมูลไบต์ที่ 70-127 ใช้เฉพาะ Fractal Programming

ตารางที่ 5-3: ข้อมูลบอกลักษณะภาพของไฟล์พีซีเอ็กซ์

การทำงานในขั้นตอนการแปลงไฟล์ข้อมูลภาพเป็นไฟล์พีซีเอ็กซ์นั้น จะมีหลักการคล้ายคลึงกันกับขั้นตอนสุดท้ายของการแก้ไขไฟล์ข้อมูลภาพ แตกต่างกันเพียงวิธีการเข้ารหัสข้อมูลภาพและการสร้างข้อมูลบอกลักษณะภาพเท่านั้น ดังนั้นการพัฒนาโปรแกรมแปลงไฟล์ข้อมูลภาพเป็นไฟล์พีซีเอ็กซ์จึงพบปัญหาเช่นเดียวกับโปรแกรมแก้ไขไฟล์ข้อมูลภาพ

จากปัญหาที่เกิดขึ้นกับโปรแกรมทั้งสองส่วนนี้ ทำให้คณะผู้พัฒนาโครงงานตัดสินใจแก้ปัญหาโดยการแยกโปรแกรมทั้งสองส่วนนี้ออกจากโปรแกรมหลัก เพื่อให้ผู้สนใจสามารถศึกษาและพัฒนาต่อได้โดยสะดวก

5.3 รายละเอียดของโปรแกรม

โปรแกรมหลักที่พัฒนาขึ้นนี้แสดงไว้ในภาคผนวก ข โดยมีฟังก์ชันต่างๆ ที่ใช้ในโปรแกรมดังนี้

SetScreen - ทำหน้าที่จัดการลักษณะของจอภาพ

ShowDir - ทำหน้าที่แสดงรายละเอียดสถานะต่างๆของโปรแกรม ได้แก่ ไดเร็กทอรีที่ใช้งานอยู่ ไฟล์ข้อมูลที่เปิดอยู่ ขนาดของภาพ ความละเอียดของภาพ และความละเอียดในการแสดงผลปัจจุบัน

PullDown - ทำหน้าที่จัดการเกี่ยวกับระบบเมนู เช่น การเลือกเมนูหลักและเมนูย่อย

SetMenu - ทำหน้าที่กำหนดค่าคงที่ต่างๆให้กับตัวแปรเมนู เช่น ข้อความในเมนู พิกัดตำแหน่งของเมนู ซึ่งค่าในตัวแปรเมนูนี้จะถูกใช้ในการสร้างพูลดาวน์เมนู

GetCurDir - ทำหน้าที่อ่านค่าไดเร็กทอรีปัจจุบันที่กำลังใช้งานอยู่

ChangeDir - ทำหน้าที่เปลี่ยนไดเร็กทอรีปัจจุบันตามแต่ผู้ใช้จะต้องการ

OpenImage - ทำหน้าที่เปิดไฟล์ข้อมูลภาพแบบ IMG และอ่านข้อมูลประจำไฟล์ซึ่งเป็นตัวบอกรายละเอียดต่างๆ ของภาพ

ChangeMenu - ทำหน้าที่จัดการจอภาพในการเปลี่ยนหัวข้อเมนูหลัก

ChangeChoice - ทำหน้าที่จัดการจอภาพในการเปลี่ยนตัวเลือกย่อยในแต่ละเมนู

Box - ทำหน้าที่สร้างตารางสี่เหลี่ยมขึ้นตามพิกัดที่กำหนด

HotKey - ทำหน้าที่ตรวจสอบการกดคีย์ในระหว่างการเลือกเมนู ว่าเป็นคีย์ที่กำหนดไว้หรือไม่

DialogBox - ทำหน้าที่สร้างตารางสี่เหลี่ยมขึ้นล้อมรอบข้อความที่กำหนด

ErrorBox - ทำหน้าที่สร้างตารางสี่เหลี่ยมเพื่อบอกข้อผิดพลาดให้ผู้ใช้ทราบ

SetGraph - ทำหน้าที่กำหนดการทำงานของจอภาพให้อยู่ในกราฟิกโหมด

DisplayZoom - ทำหน้าที่นำภาพจากไฟล์ข้อมูลภาพที่เปิดเอาไว้แล้ว ออกมาแสดงผลตามความละเอียดที่กำหนดไว้ ทั้งในลักษณะเดิมของภาพ และในลักษณะที่มีการหมุนภาพแล้ว

Print - ทำหน้าที่นำภาพจากไฟล์ข้อมูลภาพที่เปิดเอาไว้แล้ว ไปพิมพ์ออกทางเครื่องพิมพ์

SetGraphScreen - ทำหน้าที่จัดการจอภาพในกราฟิกโหมด เพื่อแสดงรายละเอียดต่างๆ ของภาพ

Receive - ทำหน้าที่รับข้อมูลจากสแกนเนอร์ โดยผ่านทางอินเทอร์เฟสการ์ด แล้วนำมาสร้างเป็นไฟล์ข้อมูลชั่วคราว

ConvertToIMG - ทำหน้าที่แปลงไฟล์ข้อมูลชั่วคราว ให้กลายเป็นไฟล์ข้อมูลภาพ ตามรูปแบบที่กำหนด

ClearDialogBox - ทำหน้าที่ลบ DialogBox ที่สร้างเอาไว้

ChangeTempFile - ทำหน้าที่เปลี่ยนชื่อไฟล์ซึ่งจะนำมาเป็นไฟล์ข้อมูลชั่วคราว

ImageWarn - ทำหน้าที่ตรวจสอบความเป็นไปได้ในการแสดงภาพออกทางจอภาพและส่งข้อความเตือนให้ผู้ใช้ทราบ

PrintByte - ทำหน้าที่ส่งข้อมูลจำนวน 1 ไบท์ออกไปทางเครื่องพิมพ์เพื่อทำการพิมพ์ พร้อมกับทำการตรวจสอบสถานะของเครื่องพิมพ์ด้วย

DefinePrinterError - วิเคราะห์สถานะของเครื่องพิมพ์ เพื่อตรวจสอบข้อผิดพลาดที่อาจเกิดขึ้นได้ในระหว่างการพิมพ์

ตัวแปรที่สำคัญที่ใช้ในโปรแกรมได้แก่

Header เป็นตัวแปรแบบโครงสร้าง ทำหน้าที่เก็บข้อมูลประจำไฟล์ ของไฟล์ข้อมูลภาพ มีรายละเอียดดังนี้

-Password เป็นตัวบอกว่าไฟล์นี้เป็น Image File หรือไม่ ถ้ามีค่าเป็น OB_{16} ก็แสดงว่าไฟล์นี้เป็นไฟล์ข้อมูลภาพ

-Resolution เป็นตัวบอกว่าไฟล์นี้เป็นภาพที่มีความละเอียดเท่าใด โดยที่
ถ้า Resolution เป็น 1 หมายถึง 150 จุดต่อนิ้ว
เป็น 2 หมายถึง 75 จุดต่อนิ้ว
เป็น 3 หมายถึง 50 จุดต่อนิ้ว
เป็น 5 หมายถึง 30 จุดต่อนิ้ว

-SizeX เป็นตัวบอกว่าภาพนี้มีขนาดทางแกน X เป็นกี่จุดภาพ

-SizeY เป็นตัวบอกว่าภาพนี้มีขนาดทางแกน Y เป็นกี่จุดภาพ

Menu เป็นตัวแปรแบบอาเรย์ของโครงสร้าง ใช้บอกรายละเอียดของระบบเมนู มีฟิลด์ต่างๆ ดังนี้

-Name เป็นตัวแปรแบบตัวอักษรระบุชื่อของเมนูนั้น ๆ

-Win เก็บค่าพิกัดของหน้าต่างของเมนูนั้น ๆ

-Msg เป็นอาเรย์ของตัวอักษรใช้เก็บข้อความของตัวเลือกแต่ละตัวในแต่ละเมนู

-LastChoice เป็นตัวแปรสำหรับเก็บค่าจำนวนตัวเลือกของแต่ละเมนู

CurMenu ใช้เก็บหมายเลขของเมนูปัจจุบัน

CurChoice เป็นอาเรย์ใช้เก็บหมายเลขของตัวเลือกปัจจุบันของแต่ละเมนู

CurFileName เป็นตัวแปรแบบตัวอักษรใช้เก็บชื่อของไฟล์ข้อมูลภาพในปัจจุบัน

TempFile เป็นตัวแปรแบบตัวอักษรใช้เก็บชื่อของไฟล์ข้อมูลชั่วคราวในปัจจุบัน

CurFile เป็นตัวแปรพอยน์เตอร์ของไฟล์ จะชี้ไปที่ไฟล์ที่ติดต่อด้วย

CurRes เป็นตัวแปรแบบตัวเลขใช้เก็บค่าความละเอียดภาพที่จะแสดงผลในปัจจุบัน

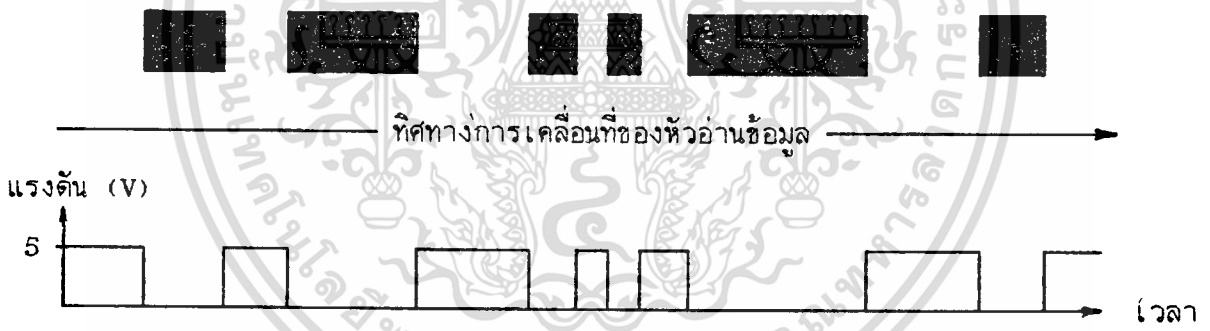
error_type เป็นตัวแปรอาเรย์ของตัวอักษรใช้เก็บข้อความที่จะแสดง เมื่อตรวจพบข้อผิดพลาดเกี่ยวกับเครื่องพิมพ์

บทที่ 6

ผลการทดลอง

6.1 ส่วนหัวอ่านข้อมูล

เมื่อนำหัวอ่านข้อมูลมาทดลองกับรูปภาพขาวดำและลายวงจรพิมพ์ โดยติดหัวอ่านไว้บนแท่นเครื่องแล้วให้มอเตอร์หมุนเพื่อเลื่อนหัวอ่านไปด้วยความเร็วต่างๆกัน ปรากฏว่า ลักษณะของสัญญาณที่ได้จากวงจร เป็นไปตามที่คาดไว้คือมีระดับ 0 โวลท์ หรือ 5 โวลท์ เปลี่ยนแปลงไปตามสีของจุดภาพที่หัวอ่านเคลื่อนที่ผ่าน อย่างไรก็ตามหัวอ่านข้อมูลนี้ก็ไม่สามารถแยกความแตกต่างระหว่างสีที่มีความใกล้เคียงกันได้ นอกจากนี้ระยะระหว่างหัวอ่านกับรูปภาพก็มีผลต่อการทำงานของวงจรด้วย รูปที่ 6-1 แสดงตัวอย่างข้อมูลที่ใช้ทดลองและสัญญาณที่ได้



รูปที่ 6-1: ตัวอย่างข้อมูลและสัญญาณที่ได้จากหัวอ่านข้อมูล

6.2 วงจรขับสเต็ปมอเตอร์

จากการทดลองปรับปรุงวงจรขับสเต็ปมอเตอร์ได้ผลว่า วงจรขับมอเตอร์ที่เลือกใช้ในโครงงานนี้สามารถควบคุมให้มอเตอร์หมุนได้ด้วยความเร็วสูงสุดประมาณ 800 สเต็ปต่อวินาที หรือ 240 รอบต่อนาที เมื่อนำมาประกอบกันเข้าในโครงสร้างของแท่นเครื่อง ปรากฏว่ามอเตอร์สามารถรับโหลดได้ดี นั่นคือสามารถดึงเชือกที่ต่อกับส่วนเคลื่อนไหวต่างๆให้เลื่อนไปมายังตำแหน่งที่ต้องการได้

6.3 บอร์ดควบคุมและโปรแกรมมอนิเตอร์

จากการทดลองบอร์ดควบคุมโดยการทดสอบหน่วยความจำชั่วคราว และพอร์ทต่างๆ พบว่าทุกส่วนทำงานได้ถูกต้อง และเมื่อนำโปรแกรมมอนิเตอร์มาควบคุมการทำงานตามขั้นตอนต่างๆดังที่ได้กล่าวไว้ในบทที่ 4 จะได้ผลการทดลองดังนี้

1. สามารถหาตำแหน่งเริ่มต้นการสแกนและขอบเขตการสแกนภาพได้

2. สามารถควบคุมการหมุนของสเต็ปมอเตอร์ได้

3. สามารถอ่านข้อมูล เข้ารหัสข้อมูล เก็บข้อมูลในหน่วยความจำ และส่งข้อมูลให้แก่ไมโครคอมพิวเตอร์ได้อย่างถูกต้อง

6.4 อินเทอร์เฟสการ์ดและโปรแกรมใช้งานบนคอมพิวเตอร์

จากการทดลองรับส่งข้อมูลผ่านทางอินเทอร์เฟสการ์ด พบว่าการรับส่งข้อมูลเป็นไปอย่างถูกต้อง และจากการทดลองใช้งานโปรแกรมหลักบนไมโครคอมพิวเตอร์ พบว่าโปรแกรมสามารถทำงานกับข้อมูลจำลองที่สร้างขึ้นได้อย่างถูกต้อง คือสามารถ

1. แปลงไฟล์ข้อมูลชั่วคราวให้เป็นไฟล์ข้อมูลภาพแบบ IMG ได้

2. นำไฟล์ข้อมูลภาพมาแสดงผลทางจอภาพได้ และสามารถหมุนภาพไป 90 องศาได้อีกด้วย

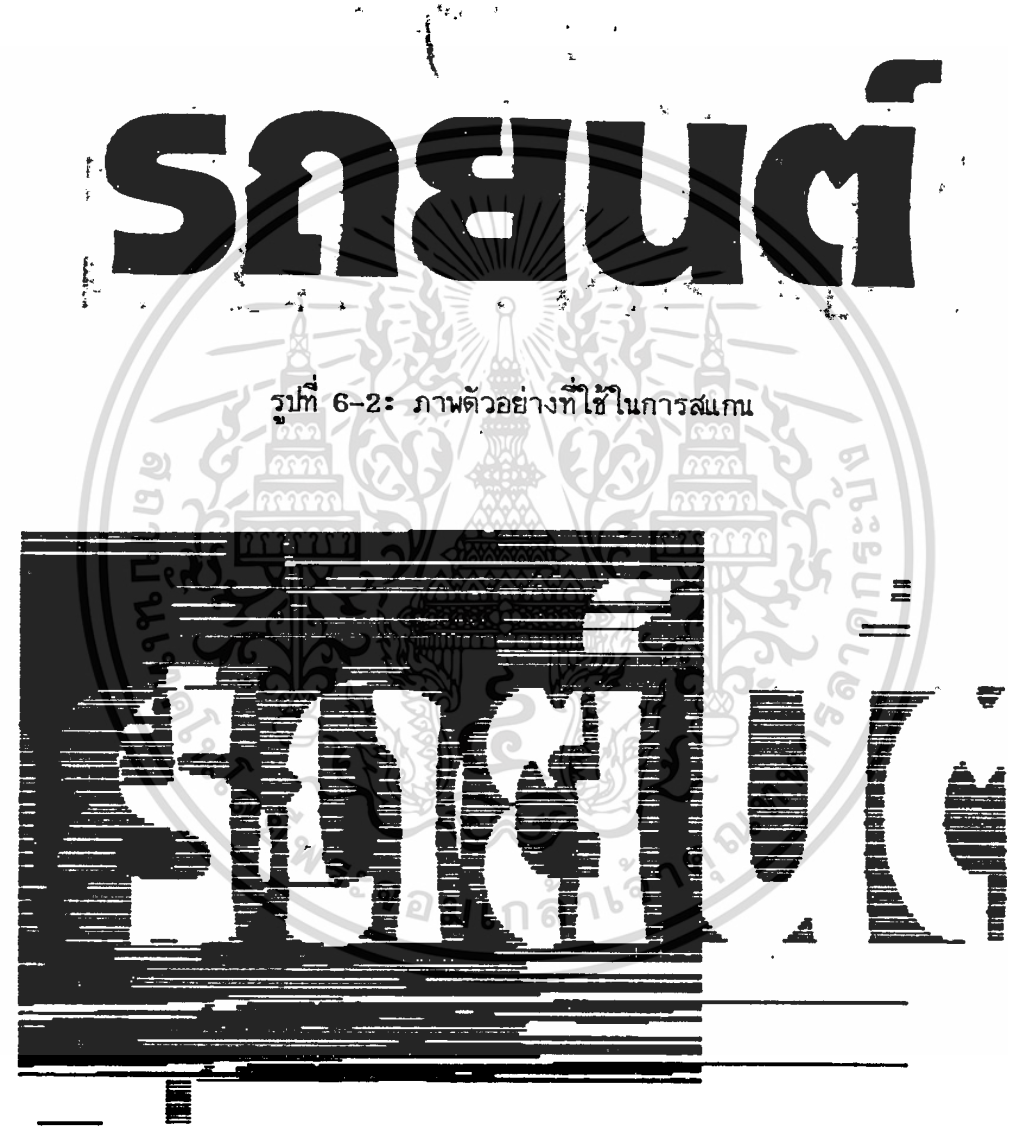
3. พิมพ์ภาพที่เก็บไว้ในไฟล์ข้อมูลภาพออกทางเครื่องพิมพ์ EPSON LX-86 และ RX-80 ได้

6.5 ส่วนกลไกแท่นเครื่อง

จากการทดลองพบว่าตามโครงสร้างที่กล่าวมาในบทที่ 3 ส่วนกลไกสามารถทำงานได้ดีพอสมควร กล่าวคือเมื่อยึดเชือกและมอเตอร์เข้ากับแท่นเครื่องแล้ว หัวอ่านสามารถเคลื่อนที่ได้โดยสะดวก แต่ยังมีปัญหาในเรื่องการชดเชยตำแหน่งหัวอ่าน เมื่อสั่งให้หัวอ่านเลื่อนขึ้นลงตามแนวแกน Y จะทำให้หัวอ่านเคลื่อนที่ในแนวแกน X ด้วย แม้ว่าจะได้สั่งให้มอเตอร์หมุนสวนทางเพื่อชดเชยตำแหน่งแล้วก็ตาม พบว่าบางครั้งสามารถทำให้หัวอ่านหยุดอยู่กับที่ในแนวแกน X ได้ แต่ก็มีบ้างบางครั้งที่หัวอ่านเคลื่อนที่ไปในทิศทางที่สั่งให้ชดเชย

6.6 ผลการทดลองของโครงงานโดยรวม

จากการทดลองสแกนภาพตัวอย่างดังรูปที่ 6-2 จะได้ตัวอย่างรูปที่แสดงบนจอภาพดังรูปที่ 6-3 และรูปที่พิมพ์ออกมาทางเครื่องพิมพ์ดังรูปที่ 6-4



รูปที่ 6-2: ภาพตัวอย่างที่ใช้ในการสแกน

รูปที่ 6-3: ภาพที่แสดงบนจอภาพเมื่อสแกนแล้ว

รูปที่ 6-4: แสดงภาพที่พิมพ์ออกทางเครื่องพิมพ์

บทที่ 7
สรุปผลการทดลอง

จากผลการทดสองที่ผ่านมาทั้งหมดสามารถสรุปได้ว่า ส่วนประกอบแต่ละส่วนในโครงงานนี้ทำงานได้ถูกต้อง ยกเว้นส่วนกลไกและแท่นเครื่องที่ไม่สามารถทำงานได้ตามที่ต้องการ ซึ่งเป็นผลมาจากการเลือกใช้โครงสร้างในลักษณะดังที่ได้กล่าวมาแล้ว ปัญหานี้สามารถแก้ไขได้โดยการใช้โครงสร้างทางกลไกในลักษณะอื่นแทน เช่น การนำสเต็ปมอเตอร์ซึ่งควบคุมการเลื่อนหัวอ่านในแนวแกน X ไปไว้บนแกน Y ซึ่งจะช่วยให้การเคลื่อนที่ของแกน Y ไม่มีผลต่อแกน X อย่างไรก็ตามลักษณะโครงสร้างแต่ละชนิดต่างก็มีข้อดีข้อเสียที่แตกต่างกันไป การเลือกใช้จึงควรศึกษาให้ละเอียดรอบคอบก่อน สำหรับการแก้ไขปัญหาที่เกิดขึ้นนี้ยังสามารถทำได้โดยทางซอฟท์แวร์(Software) เพื่อลดปัญหานี้ให้น้อยลง

จากปัญหาของโครงสร้างทางกลไกที่กล่าวมาข้างต้น ทำให้ผลการทดลองโดยรวมได้รูปภาพที่แสดงออกมาทั้งทางเครื่องพิมพ์และบนจอภาพผิดเพี้ยนไปจากภาพต้นฉบับที่นำมาสแกน

ส่วนความสามารถต่างๆ ของเครื่องสแกนเนอร์ที่พัฒนาขึ้น สามารถสรุปได้ดังนี้

1.ความเร็วในการทำงาน มีความเร็วในการสแกนภาพค่อนข้างช้า ซึ่งเป็นไปตามที่คาดไว้ล่วงหน้าแล้ว

2.ความละเอียดของภาพที่สแกนได้ ซึ่งเดิมกำหนดไว้เป็น 150 จุดต่อนิ้ว โครงงานที่พัฒนาขึ้นนี้ สามารถปรับแต่งให้ทำงานได้ที่ความละเอียดภาพตามที่กำหนด

3.คุณภาพของภาพที่แสดงผลทางจอภาพ มีคุณภาพพอใช้ เนื่องจากข้อจำกัดทางการแสดงผลของจอภาพคอมพิวเตอร์ เช่น จำนวนจุดภาพสูงสุด และ อัตราส่วนระหว่างภาพแกนนอนและแกนตั้ง

4.คุณภาพของภาพที่พิมพ์ออกมาจากเครื่องพิมพ์ มีคุณภาพค่อนข้างดี เพราะมีอัตราส่วนของภาพใกล้เคียงกับภาพต้นฉบับ อย่างไรก็ตาม ยังมีข้อจำกัดอยู่อีกหลายประการเกี่ยวกับเครื่องพิมพ์ เช่น ความละเอียดของเครื่องพิมพ์ต่างชนิดกันจะมีค่าไม่เท่ากัน ทำให้ไม่สามารถใช้งานกับเครื่องพิมพ์ได้ทุกรุ่น

แนวทางการพัฒนาโครงงานนี้ต่อไปในอนาคต มีดังนี้

1.พัฒนาหัวอ่านข้อมูล ให้สามารถแยกแยะความแตกต่างระหว่างสี 2 สีใดๆ นอกจากสีขาวและสีดำได้ โดยการปรับปรุงวงจรของหัวอ่านให้มีประสิทธิภาพสูงขึ้น

2.พัฒนาวงจรขับมอเตอร์ ให้ขับมอเตอร์ได้เร็วขึ้นและมีแรงบิดสูงขึ้น เช่นการใช้วงจรขับมอเตอร์แบบแรงดันคู่

3. พัฒนาบอร์ดควบคุมและโปรแกรมมอนิเตอร์ โดยเพิ่มวงจรเก็บรักษาข้อมูลเมื่อไฟดับระหว่างทำการสแกนภาพ เมื่อไฟฟ้าใช้การได้ตามปกติก็สามารถที่จะทำการสแกนภาพที่ค้างไว้ได้ โดยไม่ต้องเริ่มสแกนใหม่

4. พัฒนาโปรแกรมใช้งานเพิ่มเติมดังต่อไปนี้

-ทำให้สามารถใช้กับจอภาพได้หลายชนิด

-ทำให้สามารถใช้กับเครื่องพิมพ์ได้หลายยี่ห้อ และหลายรุ่น

-พัฒนาการแสดงผลทางจอภาพให้ดีขึ้น เช่น ในกรณีที่ไม่สามารถแสดงทั้งภาพได้ในจอภาพเดียว ก็ให้สามารถเลื่อนภาพ(Scroll)ได้

-พัฒนาโปรแกรมที่ทำหน้าที่แก้ไขไฟล์ข้อมูลภาพ และแปลงไฟล์ข้อมูลภาพเป็นมาตรฐานพีซีเอ็กซ์ ให้สามารถใช้งานร่วมกับโปรแกรมหลักได้

หนังสืออ้างอิง

[1] Takashi Kenjo, Stepping motors and their microprocessor controls, Oxford University Press, New York, 1986.

[2] Step Motors and Control Systems

[3] Rafael C Gonzalez and Paul Wintz, Digital Image Processing, Addison-Wesley, USA, 1987.

[4] Rod Salwon and Mel Slater, Computer Graphics Systems & Concepts, Addison-Wesley, UK, 1987.

[5] John Uffenbeck, Microcomputers and Microprocessors: The 8080, 8085, and Z-80 programming, interfacing, and troubleshooting. Prentice-Hall, New Jersey.

[6] George Sutty and Steve Blair, Programmer's Guide to the EGA/VGA, Brady, New York, 1986.

[7] Robert Jourdain, Programmer's Problem Solver for the IBM PC,XT & AT, Brady, New York, 1986.

[8] ชัยยุทธ ตันประทุมวงษ์. แกะรอย PCX ด้วยเทอร์โบซี. วารสารไมโครคอมพิวเตอร์. ฉบับที่ 78. หน้า 256-265. มกราคม 2535.

[9] ชูชัย ธนสารตั้งเจริญ และ ทินกร ดุ๊ก. การสื่อสารข้อมูล. กรุงเทพฯ. สำนักพิมพ์ฟิสิกส์เซ็นเตอร์.

[10] อินทามระ 6. PRINTER ERRORS & C LANGUAGE. นิตยสารคอมพิวเตอร์รีวิว. ฉบับที่ 89. หน้า 222-225. มกราคม 2535

กิตติกรรมประกาศ

คณะผู้จัดทำโครงงานขอกราบขอบพระคุณ บิดา มารดา ที่อนุญาตให้ไม่กลับบ้านบ่อยๆ ได้ และยังยอมเชื่อว่าไม่ได้ไปไถลวิ่งเล่นที่ไหน

ขอขอบพระคุณ ผศ.พลผดุง ผดุงกุล อาจารย์ที่ปรึกษาของพวกเรา ซึ่งคอยให้คำปรึกษา แนะนำวิธีแก้ปัญหา ตลอดจนคอยกระตุ้นให้ขยันทำโปรเจ็ค

ขอขอบคุณ รศ.ดร.โยธิน เปรมปราณีรัชต์ ที่ช่วยให้คำแนะนำเกี่ยวกับสเต็ปมอเตอร์

ขอขอบคุณอาจารย์โกศล ชวนขยัน ที่ให้คำแนะนำให้การปรับปรุงโครงสร้างทางกลไก

ขอบคุณพี่ๆ นักศึกษาปริญญาโททุกท่าน โดยเฉพาะอย่างยิ่ง พี่สมภพ พี่ไพบูลย์ และพี่หญิง ที่เอื้อเฟื้อให้คำแนะนำ ตลอดจนให้ยืมอุปกรณ์ต่างๆ โดยไม่ต้องคืน

ขอบคุณเพื่อนร่วมห้องโปรเจ็ค ที่ช่วยให้ความบันเทิงระหว่างทำโปรเจ็ค

ขอบคุณกำลังใจพิเศษจากสาวๆ ซึ่งก็เข้าใจดีว่าไม่มีเวลาให้ แต่ต้องทำเป็นงอนเอาฟอร์มไว้ก่อน,

ภาคผนวก ก
โปรแกรมมอนิเตอร์

```
;  ***********************************************
;  *                                             *
;  *       CONTROL-BOARD MONITOR PROGRAM.        *
;  *                                             *
;  *             PROJECT: SCANNER                *
;  *                                             *
;  *                   BY                        *
;  *                                             *
;  *   Mr.Comgrit Sirisangchaikul   4C  31.1019  *
;  *   Mr.Vibool  Korjaranjit       4C  31.1254  *
;  *   Mr.Suradej Tritrailucksana   4C  31.1365  *
;  *                                             *
;  *   ADVISOR: Asst.Prof.Polphadung Phadungkul  *
;  *                                             *
;  *          Department of Electronic           *
;  *           Faculty of Engineering            *
;  *                 KMIT'L                      *
;  *                                             *
;  *             Academic Year 1991              *
;  *                                             *
;  ***********************************************
;
          CPU "Z80.TBL"
          HOF "INT8"
          ORG  0000H
          ;
PortA1:   EQU  20H          ;PORT NUMBER OF A1(DRIVE STEPPING MOTOR)
PortB1:   EQU  21H          ;PORT NUMBER OF B1(SOUND OUTPUT)
PortC1:   EQU  22H          ;PORT NUMBER OF C1(SW,DETECTOR,BAR CODE READER)
CtrlPort1:EQU  23H          ;CONTROL NUMBER OF PORT A1,B1,C1
PortA2:   EQU  40H          ;PORT NUMBER OF A2(COMMUNICATION OUTPUT)
                            ;PORT B2 NOT USED
PortC2:   EQU  42H          ;PORT NUMBER OF C2(HANDSHAKING SIGNALS)
CtrlPort2:EQU  43H          ;CONTROL NUMBER OF PORT A2,B2,C2
YULAST:   EQU  06H          ;LAST VALUE FOR EXCITING Y-MOTOR UPWARD
YDLAST:   EQU  03H          ;LAST VALUE FOR EXCITING Y-MOTOR DOWNWARD
XLLAST:   EQU  60H          ;LAST VALUE FOR EXCITING X-MOTOR LEFT
XRLAST:   EQU  30H          ;LAST VALUE FOR EXCITING X-MOTOR RIGHT
HeadL:    EQU  80H          ;HEADER OF DATA SCANNED FROM LEFT TO RIGET
HeadR:    EQU  00H          ;HEADER OF DATA SCANNED FORM RIGHT TO LEFT
B128th:   EQU  7FH          ;DATA OF THE 128th BLACK DOT
W128th:   EQU  0FFH         ;DATA OF THE 128th WHITE DOT
DetectX:  EQU  00000010B    ;PC1
DetectY:  EQU  00000100B    ;PC2
IMG_W:    EQU  10000000B    ;BIT 7 OF IMG_W SHOWS THAT IT'S WHITE DOT
IMG_B:    EQU  00000000B    ;BIT 7 OF IMG_B SHOWS THAT IT'S BLACK DOT
          ; --- NOTE ---
LAL:      EQU  82H          ;NOTE A LOW
TEL:      EQU  71H          ;NOTE B LOW
DO:       EQU  6DH          ;NOTE C
RE:       EQU  61H          ;NOTE D
ME:       EQU  57H          ;NOTE E
FA:       EQU  51H          ;NOTE F
SOL:      EQU  49H          ;NOTE G
LA:       EQU  40H          ;NOTE A
TE:       EQU  39H          ;NOTE B
DOH:      EQU  36H          ;NOTE C HIGH
REH:      EQU  30H          ;NOTE D HIGH
```

```
MEH:        EQU  2BH            ;NOTE B HIGH
FAH:        EQU  28H            ;NOTE F HIGH
SOH:        EQU  24H            ;NOTE G HIGH
LAH:        EQU  20H            ;NOTE A HIGH
TEH:        EQU  1CH            ;NOTE B HIGH
DOHH:       EQU  18H            ;NOTE C ULTRAHIGH
            ; --- STOP NOTE ---
STP:        EQU  01H            ;NOTE STOP SOUND
            ; --- RHYTHM ---
R1_8:       EQU  04H            ;ONE-EIGHTH BEAT
R1_4:       EQU  15H            ;ONE-FOURTH BEAT
HALF:       EQU  2AH            ;HALF BEAT
ONE:        EQU  55H            ;1 BEAT
TWO:        EQU  0AAH           ;2 BEATS
TEE:        EQU  0FFH           ;3 BEATS
            ;
            ;   ***             ***
            ;   *** SYSTEM RESET ***
            ;   ***             ***
            ;
RESET:      LD   HL,4000H       ;DELAY
RESET0:     DEC  HL
            LD   A,H
            OR   L
            JR   NZ,RESET0
            LD   SP,SYSSTK      ;LOAD SYSTEM STACK
            LD   A,89H          ;LOAD CONTROL WORD 89h(10001001b)
                                ; D7=1:CONTROL WORD FORMAT
                                ; D6D5=00:MODE 0 SELECT
                                ; D4=0:PORT A OUTPUT
                                ; D3=1:PORT C(UPPER) INPUT
                                ; D2=0:MODE 0 SELECT
                                ; D1=0:PORT B OUTPUT
                                ; D0=1:PORT C(LOWER) INPUT
            OUT  (CtrlPort1),A  ;SEND CONTROL WORD TO 8255 #1
            LD   A,83H          ;LOAD CONTROL WORD 83h(10000011b)
                                ; D7=1:CONTROL WORD FORMAT
                                ; D6D5=00:MODE 0 SELECT
                                ; D4=0:PORT A OUTPUT
                                ; D3=0:PORT C(UPPER) OUTPUT
                                ; D2=0:MODE 0 SELECT
                                ; D1=1:PORT B INPUT
                                ; D0=1:PORT C(LOWER) INPUT
          OUT  (CtrlPort2),A   ;SEND CONTROL WORD TO 8255 #2
            LD   A,80H          ;SET STROBE\=1,CTRLFILE0=000
            OUT  (PortC2),A     ;SEND STROBE\ AND CTRLFILE0
            LD   HL,YDDRIVE     ;INITIAL EXCITING VALUE FOR Y-MOTOR
            LD   (yPOS),HL
            LD   HL,XDRIVE      ;INITIAL EXCITING VALUE FOR X-MOTOR
            LD   (xPOS),HL
            ;
            ;   --- PLAY TITLE SONG ---
            ;
            LD   D,0CH          ;LOAD NUMBER OF NOTES
            LD   IX,TABLE2      ;LOAD ADDRESS OF NOTE TABLE
            CALL SONG
            ;
            ;   --- DELAY FOR A WHILE ---
```

```
                ;
                LD   B,0
                CALL DELAY          ;DELAY FOR A WHILE
                ;
                ;  ***                      ***
                ;  *** TAKE HEADER TO HOME. ***
                ;  ***                      ***
                ;
                ;  --- MOVE Y-MOTOR TO POSITION Y=0 ---
                ;
Y_HOME:         IN   A,(PortC1)     ;READ DATA FROM PORT
                AND  DetectY        ;MASK BIT TO GET DATA FROM DETECTER
                CP   DetectY        ;CHECK IF READER IS AT POSITION=0
                JR   Z,Y_HOME1      ;POSITION IS AT 0, JUMP TO Y_HOME1
                LD   B,6H           ;LOAD DELAY TIME
                CALL YUP            ;MOVE Y-MOTOR UPWARD
                JR   Y_HOME         ;JUMP BACK TO CHECK HOME AGAIN
Y_HOME1:        LD   A,0H
                OUT  (PortA1),A     ;CLEAR MAGNITIC FLUX
                ;
                ;  --- SEND SOUND OUT ---
                ;
                LD   BC,0A036H      ; B = LOUDNESS   C = FREQUENCY OF SOUND
                LD   HL,8000H       ;LOAD SOUND LENGHT
                CALL SOUND
                ;
                ;  --- MOVE X-MOTOR TO POSITION X=0 ---
                ;
X_HOME:         IN   A,(PortC1)     ;READ DATA FROM PORT
                AND  DetectX        ;MASK BIT TO GET DATA FROM DETECTOR
                CP   DetectX        ;CHECK IF READER IS AT POSITION=0
                JR   Z,X_HOME1      ;POSITION IS AT 0, JUMP TO X_HOME1
                LD   B,6H           ;LOAD DELAY TIME
                CALL XLEFT          ;MOVE X-MOTOR TO THE LEFT
                JR   X_HOME         ;JUMP BACK TO CHECK HOME AGAIN
X_HOME1:        LD   A,0H
                OUT  (PortA1),A     ;CLEAR MAGNITIC FLUX
                ;
                ;  --- SEND SOUND OUT ---
                ;
                LD   BC,0A036H      ; B = LOUDNESS   C = FREQUENCY
                LD   HL,8000H       ;LOAD SOUND LENGHT
                CALL SOUND
                ;
                ;  ***                              ***
                ;  *** FIND RIGHT MARGIN FOR SCANNING ***
                ;  ***                              ***
                ;
                ;  ---WAIT TO START FINDING RIGHT MARGIN ---
                ;
MargW1:         IN   A,(PortC1)     ;GET DATA FROM PORT C1
                BIT  4,A            ;CHECK IF MARGIN SWITCH IS PRESSED
                JR   Z,MargW1       ;WAIT UNTIL MARGIN SWITCH IS PRESSED
                LD   B,0H           ;LOAD DELAY PARAMETER TO REDUCE DEBOUNCE AND
                CALL DELAY          ;   TOO LONG TIME PRESSING SWITCH PROBLEM.
                ;
                ;  --- NOW! START FINDING ---
                ;
```

```
        LD   HL,0H          ;SET NUMBER OF DOT FOR COUNTING
CNTx:   LD   B,6H           ;PASS PARAMETER:-DELAY TIME
        PUSH HL
        CALL XRIGHT         ;MOVE X-MOTOR TO THE RIGHT
        POP  HL
        INC  HL             ;COUNT DOT(S) FOR SCANNING IN X-AXIS
        IN   A,(PortC)      ;READ DATA FROM PORT C1
        BIT  4,A            ;CHECK IF MARGIN SWITCH IS PRESSED
        JR   NZ,MOVEBACK    ;MOVE HEADER BACK TO HOME
        ;
        ;  --- CHECK IF HEADER IS AT THE RIGHT EDGE. ---
        ;
        ;  The lenght for scaning in x-axis is 8 inches. In one inch,
        ;  there are 150 dots, so the right edge is at the 1200th dot.
        ;  (1200d=4B0h)
        ;
        LD   A,H
        CP   4H
        JR   NZ,CNTx        ;JUMP BACK TO FIND THE EDGE
        LD   A,L
        CP   0B0H
        JR   NZ,CNTx        ;JUMP BACK TO FIND THE EDGE
        ;
        ;  --- MOVE BACK ---
        ;
MOVEBACK: LD   (RightMar),HL  ;SAVE NUMBER OF DOT(S)
        LD   BC,0101H       ; B = LOUDNESS   C = FREQUENCY
        LD   HL,6000H       ;LOAD SOUND LENGHT
        CALL SOUND          ;SEND SOUND OUT
        LD   B,6H
        CALL DELAY          ;DELAY FOR A WHILE TO REDUCE INERT FORCE
GOBACK: LD   B,3H           ;PASS PARAMETER:-DELAY TIME
        CALL XLEFT          ;MOVE MOTOR TO THE LEFT
        IN   A,(PortC)      ;READ DATA FROM PORTC1
        AND  DetectX        ;MASK BIT TO GET DATA FROM DETECTER X
        CP   DetectX        ;CHECK IF READER IS AT X=0
        JR   NZ,GOBACK      ;IT'S NOT AT X=0, JUMP TO MOVE BACK AGAIN
        LD   A,0H
        OUT  (PortA1),A     ;OUTPORT EXCITING VALUE TO CLEAR MAGNETIC FLUX
        ;
        ;  ***                         ***
        ;  *** FIND BOTTOM FOR SCANNING ***
        ;  ***                         ***
        ;
        ;  --- WAIT TO START FINDING BOTTOM ---
        ;
MargW2: IN   A,(PortC)      ;GET DATA FROM PORT C1
        BIT  4,A            ;CHECK IF MARGIN SWITCH IS PRESSED
        JR   Z,MargW2       ;WAIT UNTIL MARGIN SWITCH IS PRESSED
        LD   B,0H           ;LOAD DELAY PARAMETER TO REDUCE DEBOUNCE AND
        CALL DELAY          ;   TOO LONG TIME PRESSING SWITCH PROBLEM.
        ;
        ;  --- NOW! START FINDING ---
        ;
        LD   HL,0H          ;SET NUMBER OF DOT FOR COUNTING
CNTy:   LD   B,6H           ;PASS PARAMETER:-DELAY TIME
        PUSH HL
        CALL YDOWN          ;MOVE Y-MOTOR DOWNWARD
```

```
            POP  HL
            INC  HL             ;COUNT DOT(S) FOR SCANNING IN Y-AXIS
            IN   A,(PortC1)     ;READ DATA FROM PORT C1
            BIT  4,A            ;CHECK IF MARGIN SWITCH IS PRESSED
            JR   NZ,MOVEBACK2   ;MOVE HEADER BACK TO HOME
            ;
            ;   --- CHECK IF READER IS AT THE BOTTOM. ---
            ;
            ;   The lenght for scaning in y-axis is 11 inches. In one inch,
            ;   there are 150 dots, so the right edge is at the 1650th dot.
            ;   (1650d=672h)
            ;
            LD   A,H
            CP   6H
            JR   NZ,CNTy        ;JUMP BACK TO FIND THE BOTTOM
            LD   A,L
            CP   72H
            JR   NZ,CNTy        ;JUMP BACK TO FIND THE BOTTOM
            ;
            ;   --- MOVE BACK ---
            ;
MOVEBACK2:  LD   (BOTTOM),HL    ;SAVE NUMBER OF DOT(S)
            LD   BC,0A016H      ; B = LOUDNESS   C = FREQUENCY
            LD   HL,6000H       ;LOAD SOUND LENGHT
            CALL SOUND          ;SEND SOUND OUT
            LD   B,6H
            CALL DELAY          ;DELAY FOR A WHILE
GOBACK2:    LD   B,6H           ;PASS PARAMETER:-DELAY TIME
            CALL YUP            ;MOVE Y-MOTOR UPWARD
            IN   A,(PortC1)     ;READ DATA FROM PORTC1
            AND  DetectY        ;MASK BIT TO GET DATA FROM DETECTER Y
            CP   DetectY        ;CHECK IF READER IS AT Y=0
            JR   NZ,GOBACK2     ;IF'S NOT AT Y=0, JUMP TO MOVE BACK AGAIN
            LD   A,0H
            OUT  (PortA1),A     ;OUTPORT EXCITING VALUE TO CLEAR MAGNETIC FLUX
            ;
            ;   ***         ***
            ;   *** SCANNING ***
            ;   ***         ***
            ;
            ;   --- WAIT TO START SCANNING ---
            ;
SCANNING:   IN   A,(PortC1)     ;READ DATA FROM PORT C1
            BIT  5,A            ;CHECK IF START SWITCH IS PRESSED
            JR   Z,SCANNING     ;WAIT UNTIL SWITCH IS PRESSED
            LD   B,0H           ;LOAD DELAY PARAMETER TO REDUCE DEBOUNCE AND
            CALL DELAY          ;   TOO LONG TIME PRESSING SWITCH PROBLEM.
            LD   A,01000000B    ;LOAD A WITH SOP(START OF FILE) CODE
            LD   (CTRLFILE0),A  ;MOVE INTO CTRLFILE0
            XOR  A              ;CLEAR REGISTER A
            LD   (CTRLFILE1),A  ;SAVE EOF=00000000b IN CTRLFILE1
            ;
            ;   --- NOW! START SCANNING ---
            ;
            ;   * Scan from left to right *
            ;
            LD   BC,1H          ;SET INITIAL NUMBER OF LINES
            EXX                 ;SAVE NUMBER OF LINES IN REGISTER BC'
```

```
        LD   HL,RAM         ;LOAD ADDRESS OF RAM INTO REGISTER PAIR HL
        LD   BC,0H          ;SET NUMBER OF DOT IN ONE LINE TO ZERO
SCAN:   LD   (HL),HeadL     ;SAVE LEFT HEADER
LOOP1:  LD   DE,0H          ;SET NUMBER OF BLACK AND WHITE DOT TO ZERO
                            ;    WHITE --> REGISTER D
                            ;    BLACK --> REGISTER E
        INC  HL             ;INCREASE ADDRESS OF DATA
        PUSH BC
        CALL MEMORY         ;CHECK IF MEMORY IS FULL
        POP  BC
        IN   A,(PortC1)     ;READ DATA FROM PORT C1
        BIT  0,A            ;CHECK IF IT'S WHITE DOT OR BLACK DOT
        JR   Z,BLACK1       ;JUMP IF IT'S BLACK
WHITE1: INC  D              ;COUNT NUMBER OF WHITE DOT
        LD   A,D
        CP   W128th         ;COMPARE IF IT'S THE 128th DOT
        JR   Z,FULL_W1      ;IT'S THE 128th DOT. JUMP TO FULL_W1
        OR   IMG_W          ;CREATE IMAGE DATA OF WHITE DOT
        LD   (HL),A         ;IMAGE DATA ---> (SRAM)
        INC  BC             ;INCREASE NUMBER OF DOTS BY 1
        PUSH HL             ;ADDRESS OF DATA --> STACK
        LD   HL,(RightMar)  ;RIGHT-MARGIN ---> REGISTER PAIR HL
        CALL CMP16          ;CHECK IF IT'S AT RIGHT MARGIN
        JR   Z,LAST1        ;IT'S AT RIGHT MARGIN, JUMP TO LAST1
        PUSH BC
        LD   B,2H           ;PASS PARAMETER:-DELAY TIME
        CALL XRIGHT         ;TURN X-MOTOR TO THE RIGHT FOR 1 STEP
        POP  BC
        POP  HL             ;GET ADDRESS OF DATA BACK FROM STACK
        IN   A,(PortC1)     ;READ DATA FROM PORT C1
        BIT  0,A            ;CHECK IF IT'S WHITE DOT OR BLACK DOT
        JR   NZ,WHITE1      ;IT'S WHITE DOT, THEN JUMP TO WHITE1
        INC  HL             ;INCREASE ADDRESS OF DATA
        PUSH BC
        CALL MEMORY         ;CHECK IF MEMORY IS FULL
        POP  BC
        LD   E,0H           ;SET NUMBER OF BLACK DOT TO ZERO
        JR   BLACK1         ;JUMP TO BLACK1
FULL_W1: LD  (HL),W128th    ;SAVE OFFH,AN IMAGE DATA,TO (SRAM)
        JR   FULL1
BLACK1: INC  E              ;COUNT NUMBER OF BLACK DOT
        LD   A,E
        CP   B128th         ;COMPARE IF IT'S THE 128th DOT
        JR   Z,FULL_B1      ;IT'S THE 128th DOT. JUMP TO FULL_B1
        OR   IMG_B          ;CREATE IMAGE DATA OF BLACK DOT
        LD   (HL),A         ;IMAGE DATA ---> (SRAM)
        INC  BC             ;INCREASE NUMBER OF DOTS BY 1
        PUSH HL             ;ADDRESS OF DATA --> STACK
        LD   HL,(RightMar)  ;RIGHT-MARGIN ---> REGISTER PAIR HL
        CALL CMP16          ;CHECK IF IT'S AT RIGHT MARGIN
        JR   Z,LAST1        ;IT'S AT RIGHT MARGIN, JUMP TO LAST1
        PUSH BC
        LD   B,2H
        CALL XRIGHT         ;TURN X-MOTOR TO THE RIGHT FOR 1 STEP
        POP  BC
        POP  HL             ;GET ADDRESS OF DATA BACK FROM STACK
        IN   A,(PortC1)     ;READ DATA FROM PORT C1
        BIT  0,A            ;CHECK IF IT'S WHITE DOT OR BLACK DOT
```

```
            JR   Z,BLACK1       ;IT'S BLACK DOT, THEN JUMP TO BLACK1
            INC  HL             ;INCREASE ADDRESS OF DATA
            PUSH BC
            CALL MEMORY         ;CHECK IF MEMORY IS FULL
            POP  BC
            LD   D,0E           ;SET NUMBER OF WHITE DOT TO ZERO
            JR   WHITE1         ;JUMP TO WHITE1
FULL_B1:    LD   (HL),B128th    ;SAVE TYPE,AN IMAGE DATA,TO (SRAM)
FULL1:      INC  BC             ;INCREASE NUMBER OF DOTS BY 1
            PUSH HL             ;ADDRESS OF DATA --> STACK
            LD   HL,(RightMar,  ;RIGHT-MARGIN ---> REGISTER PAIR HL
            CALL CMP16          ;CHECK IF IT'S AT RIGHT MARGIN
            JR   Z,LAST1        ;IT'S AT RIGHT MARGIN, JUMP TO LAST1
            PUSH BC
            LD   B,2H
            CALL XRIGHT         ;TURN X-MOTOR TO THE RIGHT FOR 1 STEP
            POP  BC
            POP  HL             ;GET ADDRESS OF DATA BACK FROM STACK
            JR   LOOP1
LAST1:      POP  HL             ;GET ADDRESS OF DATA BACK FROM STACK
            EXX                 ;GET NUMBER OF LINES FROM BC' TO BC
                                ; AND SAVE NUMBER OF DOTS FROM BC TO BC'
            LD   HL,(BOTTOM)    ;BOTTOM ---> REGISTER PAIR HL
            CALL CMP16          ;CHECK IF IT'S AT THE BOTTOM
            JP   Z,STOP         ;IT'S AT THE BOTTOM, JUMP TO STOP SCANNING
            ;
            ;   * Scan from right to left *
            ;
            PUSH BC
            LD   B,3E
            CALL DELAY          ;DELAY FOR A WHILE, TO REDUCE INERT FORCE
            POP  BC
            INC  BC             ;INCREASE THE NUMBER OF LINES BY 1
            PUSH BC
            LD   B,6H
            CALL YDOWN          ;TURN Y-MOTOR DOWN FOR 1 STEP
            POP  BC
            EXX                 ;GET NUMBER OF DOTS FROM BC' TO BC
                                ; AND SAVE NUMBER OF LINES FROM BC TO BC'
            INC  HL             ;INCREASE THE NUMBER OF ADDRESS
            PUSH BC
            CALL MEMORY         ;CHECK IF MEMORY IS FULL
            POP  BC
            LD   (HL),HeadR     ;RIGHT HEADER ---> (SRAM)
LOOP2:      LD   DE,0E          ;SET NUMBER OF BLACK AND WHITE DOT TO ZERO
                                ;    WHITE --> REGISTER D
                                ;    BLACK --> REGISTER E
            INC  HL             ;INCREASE THE NUMBER OF ADDRESS
            PUSH BC
            CALL MEMORY         ;CHECK IF MEMORY IS FULL
            POP  BC
            IN   A,(PortC1)     ;READ DATA FROM PORT C1
            BIT  0,A            ;CHECK IF IT'S WHITE DOT OR BLACK DOT
            JR   Z,BLACK2       ;JUMP IF IT'S BLACK
WHITE2:     INC  D              ;COUNT NUMBER OF WHITE DOT
            LD   A,D
            CP   W128th         ;COMPARE IF IT'S THE 128th DOT
            JR   Z,FULL_W2      ;IT'S THE 128th DOT. JUMP TO FULL_W1
```

```
        OR   IMG_W          ;CREATE IMAGE DATA OF WHITE DOT
        LD   (HL),A         ;IMAGE DATA ---> (SRAM)
        IN   A,(PortC1)     ;READ DATA FROM PORTC1
        AND  DetectX        ;MASK BIT TO GET DATA FROM DETECTER X
        CP   DetectX        ;CHECK IF READER IS AT X=0
        JR   Z,LAST2        ;IT'S AT X=0, JUMP TO LAST2
        PUSH HL             ;ADDRESS OF DATA ---> STACK
        PUSH BC
        LD   B,2H
        CALL XLEFT          ;TURN X-MOTOR TO THE LEFT FOR 1 STEP
        POP  BC
        POP  HL             ;GET ADDRESS OF DATA BACK
        IN   A,(PortC1)     ;READ DATA FROM PORT C1
        BIT  0,A            ;CHECK IF IT'S WHITE DOT OR BLACK DOT
        JR   NZ,WHITE2      ;IT'S WHITE DOT, THEN JUMP TO WHITE2
        INC  HL             ;INCREASE ADDRESS OF DATA
        PUSH BC
        CALL MEMORY         ;CHECK IF MEMORY IS FULL
        POP  BC
        LD   E,0H           ;SET NUMBER OF BLACK DOT TO ZERO
        JR   BLACK2         ;JUMP TO BLACK2
FULL_W2: LD  (HL),W128th    ;SAVE 0FFH,AN IMAGE DATA,TO (SRAM)
        JR   FULL2
BLACK2: INC  E              ;COUNT NUMBER OF BLACK DOT
        LD   A,E
        CP   B128th         ;COMPARE IF IT'S THE 128th DOT
        JR   Z,FULL_B2      ;IT'S THE 128th DOT. JUMP TO FULL_B1
        OR   IMG_B          ;CREATE IMAGE DATA OF BLACK DOT
        LD   (HL),A         ;IMAGE DATA ---> (SRAM)
        IN   A,(PortC1)     ;READ DATA FROM PORTC1
        AND  DetectX        ;MASK BIT TO GET DATA FROM DETECTER X
        CP   DetectX        ;CHECK IF READER IS AT X=0
        JR   Z,LAST2        ;IT'S AT X=0, JUMP TO LAST2
        PUSH HL             ;ADDRESS OF DATA ---> STACK
        PUSH BC
        LD   B,2H
        CALL XLEFT          ;TURN X-MOTOR TO THE LEFT FOR 1 STEP
        POP  BC
        POP  HL             ;GET ADDRESS OF DATA BACK
        IN   A,(PortC1)     ;READ DATA FROM PORT C1
        BIT  0,A            ;CHECK IF IT'S WHITE DOT OR BLACK DOT
        JR   Z,BLACK2       ;IT'S BLACK DOT, THEN JUMP TO BLACK2
        INC  HL             ;INCREASE ADDRESS OF DATA
        PUSH BC
        CALL MEMORY         ;CHECK IF MEMORY IS FULL
        POP  BC
        LD   D,0H           ;SET NUMBER OF WHITE DOT TO ZERO
        JR   WHITE2         ;JUMP TO WHITE2
FULL_B2: LD  (HL),B128th    ;SAVE 0FFH,AN IMAGE DATA,TO (SRAM)
FULL2:  IN   A,(PortC1)     ;READ DATA FROM PORTC1
        AND  DetectX        ;MASK BIT TO GET DATA FROM DETECTER X
        CP   DetectX        ;CHECK IF READER IS AT X=0
        JR   Z,LAST2        ;IT'S AT X=0, JUMP TO LAST2
        PUSH HL             ;ADDRESS OF DATA ---> STACK
        PUSH BC
        LD   B,2H
        CALL XLEFT          ;TURN X-MOTOR TO THE LEFT FOR 1 STEP
        POP  BC
```

```
          POP  HL              ;GET ADDRESS OF DATA BACK
          JR   LOOP2
LAST2:    EXX                  ;GET NUMBER OF LINES FROM BC' TO BC
                               ; AND SAVE NUMBER OF DOTS FROM BC TO BC'
          LD   HL,(BOTTOM)     ;BOTTOM ---> REGISTER PAIR HL
          CALL CMP16           ;CHECK IF IT'S AT THE BOTTOM
          JR   Z,STOP          ;IT'S AT THE BOTTOM, JUMP TO STOP
          INC  BC              ;INCREASE NUMBER OF LINES BY 1
          PUSH BC
          LD   B,3H
          CALL DELAY           ;DELAY FOR A WHILE, TO REDUCE INERT FORCE
          LD   B,6H
          CALL YDOWN           ;TURN Y-MOTOR DOWN FOR 1 STEP
          POP  BC
          EXX                  ;SAVE NUMBER OF DOTS FROM BC TO BC'
                               ; AND SAVE NUMBER OF LINES FROM BC' TO BC
          INC  HL              ;INCREASE ADDRESS OF DATA
          PUSH BC
          CALL MEMORY          ;CHECK IF MEMORY IS FULL
          POP  BC
          JP   SCAN            ;JUMP TO SCAN FROM LEFT TO RIGHT
STOP:     EXX                  ;GET NUMBER OF DOTS FROM BC' TO BC
                               ; AND SAVE NUMBER OF LINES FROM BC TO BC'
          LD   A,0H
          OUT  (PortA1),A      ;CLEAR MAGNETIC FLUX
          ;
          ;  --- WAIT FOR SENDING LAST GROUP OF DATA TO PC ---
          ;
SEND0:    IN   A,(PortC1)      ;READ DATA FROM PORT C1
          BIT  6,A             ;CHECK IF SEND DATA SWITCH IS PRESSED.
          JR   Z,SEND0         ;WAIT UNTIL SWITCH IS PRESSED.
          LD   A,00010000B     ;LOAD REGISTER A WITH EOF(END OF FILE) CODE
          LD   (CTRLFILE),A    ;SAVE IN CTRLFILE
          CALL SENDDATA        ;SEND DATA OUT TO IBM-PC
          HALT
          ;
          ; <************** SUBROUTINE AREA **************>
          ;
          ;  ====================================
          ;  TURN X-MOTOR TO THE RIGHT SUBROUTINE
          ;  ====================================
          ;  REG = ABEL
          ;  PASS PARAMETER = B
          ;
XRIGHT:   LD   HL,(xPOS)       ;GET LAST-ADDRESS OF DATA FOR EXCITING
          LD   A,(HL)          ;EXCITING DATA --> REGISTER A
          CP   XRLAST          ;CHECK IF IT'S THE LAST VALUE
          JR   NZ,NEXT_R       ;IT'S NOT THE LAST, THEN JUMP TO NEXT_R
          LD   HL,XRDRIVE      ;IT'S THE LAST, GET THE FIRST VALUE
          JR   DRIVE_R         ;JUMP TO EXCITE MOTOR
NEXT_R:   INC  HL              ;GET THE NEXT VALUE
DRIVE_R:  LD   A,(HL)          ;READ DATA FOR EXCITING MOTOR
          OUT  (PortA1),A      ;SEND DATA FOR EXCITING MOTOR
          LD   (xPOS),HL       ;SAVE POSITION OF X-AXIS IN MEMORY
          CALL DELAY
          RET
          ;
          ;  ====================================
```

```
            ;   TURN X-MOTOR TO THE LEFT SUBROUTINE
            ;   ===================================
            ;   REG = BHL
            ;   PASS PARAMETER = NONE
            ;
XLEFT:      LD  HL,(xPOS)       ;GET LAST ADDRESS OF DATA FOR EXCITING
            LD  A,(HL)          ;EXCITING DATA --> REGISTER A
            CP  XLLAST          ;CHECK IF IT'S THE LAST VALUE
            JR  NZ,NEXT_L       ;IT'S NOT THE LAST, THEN JUMP TO NEXT_L
            CALL XLADDR         ;IT'S THE LAST, GET THE FIRST VALUE
            JR  DRIVE_L         ;JUMP TO EXCITE MOTOR
NEXT_L:     DEC HL              ;GET THE NEXT VALUE
DRIVE_L:    LD  A,(HL)          ;READ DATA FOR EXCITING MOTOR
            OUT (PortA1),A      ;SEND DATA FOR EXCITING MOTOR
            LD  (xPOS),HL       ;SAVE POSITION OF X-AXIS IN MEMORY
            CALL DELAY
            RET
            ;
            ;   ==============================
            ;   TURN Y-MOTOR UPWARD SUBROUTINE
            ;   ==============================
            ;   REG = BHL
            ;   PASS PARAMETER = NONE
            ;
YUP:        LD  HL,(yPOS)       ;GET LAST ADDRESS OF DATA FOR EXCITING
            LD  A,(HL)          ;EXCITING DATA --> REGISTER A
            CP  YULAST          ;CHECK IF IT'S THE LAST VALUE
            JR  NZ,NEXT_U       ;IT'S NOT THE LAST, THEN JUMP TO NEXT_U
            CALL YUADDR         ;IT'S THE LAST, GET THE FIRST VALUE
            JR  ANTI_XR         ;JUMP TO GET EXCITING VALUE FOR X-MOTOR
                                ;   TO ANTI X-MOTOR SLIDE TO THE RIGHT
NEXT_U:     DEC HL              ;GET THE NEXT VALUE
ANTI_XR:    LD  (yPOS),HL       ;SAVE POSITION OF Y-AXIS IN MEMORY
            LD  HL,(xPOS)       ;GET LAST ADDRESS OF DATA FOR X-MOTOR
            LD  A,(HL)          ;EXCITING DATA --> REGISTER A
            CP  XLLAST          ;CHECK IF IT'S THE LAST VALUE
            JR  NZ,ANTI_NR      ;IT'S NOT THE LAST, THEN JUMP TO ANTI_NR
            CALL XLADDR         ;IT'S THE LAST, GET THE FIRST VALUE
            JR  DRIVE_YUXL      ;JUMP TO EXCITE X AND Y MOTOR SIMULTANEOUSLY
ANTI_NR:    DEC HL              ;GET THE NEXT VALUE
DRIVE_YUXL: LD  (xPOS),HL       ;SAVE POSITION OF X-AXIS IN MEMORY
            LD  A,(HL)          ;EXCITING DATA --> REGISTER A
            LD  HL,(yPOS)       ;GET LAST ADDRESS OF DATA FOR Y-MOTOR
            OR  (HL)            ;COMPINE X AND Y MOTOR EXCITING VALUE
            OUT (PortA1),A      ;SEND DATA FOR EXCITING MOTOR
            CALL DELAY
            RET
            ;
            ;   ================================
            ;   TURN Y-MOTOR DOWNWARD SUBROUTINE
            ;   ================================
            ;   REG = BHL
            ;   PASS PARAMETEL = NONE
            ;
YDOWN:      LD  HL,(yPOS)       ;GET LAST ADDRESS OF DATA FOR EXCITING
          LD  A,(HL)          ;EXCITING DATA --> REGISTER A
            CP  YDLAST          ;CHECK IF IT'S THE LAST VALUE
            JR  NZ,NEXT_D       ;IT'S NOT THE LAST, THEN JUMP TO NEXT_D
```

```
            LD   HL,YDDRIVE      ;IT'S THE LAST, GET THE FIRST VALUE
            JR   ANTI_XL         ;JUMP TO GET EXCITING VALUE FOR X-MOTOR
                                 ;   TO ANTI X-MOTOR SLIDE TO THE LEFT
NEXT_D:     INC  HL              ;GET THE NEXT VALUE
ANTI_XL:    LD   (yPOS),HL       ;SAVE POSITION OF Y-AXIS IN MEMORY
            LD   HL,(xPOS)       ;GET LAST ADDRESS OF DATA FOR X-MOTOR
            LD   A,(HL)          ;EXCITING DATA --> REGISTER A
            CP   XRLAST          ;CHECK IF IT'S THE LAST VALUE
            JR   NZ,ANTI_NL      ;IT'S NOT THE LAST, THEN JUMP TO ANTI_NL
            LD   HL,XRDRIVE      ;IT'S THE LAST, GET THE FIRST VALUE
            JR   DRIVE_YDXR      ;JUMP TO EXCITE X AND Y MOTOR SIMULTANEOUSLY
ANTI_NL:    INC  HL              ;GET THE NEXT VALUE
DRIVE_YDXR: LD   (xPOS),HL       ;SAVE POSITION OF X-AXIS IN MEMORY
            LD   A,(HL)          ;EXCITING DATA --> REGISTER A
            LD   HL,(yPOS)       ;GET LAST ADDRESS OF DATA FOR Y-MOTOR
            OR   (HL)            ;COMBINE X AND Y MOTOR EXCITING VALUE
            OUT  (Port81),A      ;SEND DATA FOR EXCITING MOTOR
            CALL DELAY
            RET
            ;
            ;   ================================================
            ;   GET ADDRESS FOR DRIVING Y-MOTOR UPWARD SUBROUTINE
            ;   ================================================
            ;   REG = HL
            ;   PASS PARAMETER = HL
            ;
YUADDR:     LD   HL,YDDRIVE
            INC  HL
            INC  HL
            INC  HL
            RET
            ;
            ;   ======================================================
            ;   GET ADDRESS FOR DRIVING X-MOTOR TO THE LEFT SUBROUTINE
            ;   ======================================================
            ;   REG = HL
            ;   PASS PARAMETER = HL
            ;
XLADDR:     LD   HL,XRDRIVE
            INC  HL
            INC  HL
            INC  HL
            RET
            ;
            ;   ============================
            ;   CHECK MEMORY-FULL SUBROUTINE
            ;   ============================
            ;   REG = ABCHL
            ;   PASS PARAMETER = HL
            ;
MEMORY:     LD   BC,EOD          ;PROHIBITED AREA ---> REGISTER BC
            PUSH HL
            CALL CMP16           ;CHECK IF SRAM IS AT THAT AREA
            POP  HL
            RET  NZ              ;IT ISN'T(MEMORY ISN'T FULL), CONTINUE SCANNING
LED:        OUT  (0A0H),A        ;YES, MEMORY IS FULL. THEN
                                 ;  --> DISPLAY "MEMORY FULL" LED
                                 ;  --> SEND SOUND OUT
```

```
                        ; --> WAIT TO SEND DATA OUT TO IBM-PC
                        ; --> DISPLAY READY TO SEND LED
        OUT  (0C0H),A       ;DISPLAY "READY TO SEND" LED
        PUSH HL             ;SAVE ADDRESS OF IMAGE DATA
        LD   BC,0E01AH      ; B = LOUDNESS   C = FREQUENCY
        LD   HL,5000H       ;LOAD SOUND LENGHT
        CALL SOUND          ;SEND SOUND OUT    .
        LD   B,0
        LD   HL,3000H
        CALL SOUND          ;SOUND OFF
        POP  HL             ;LOAD ADDRESS OF IMAGE DATA BACK FROM STACK
        IN   A,(PortC1)     ;READ DATA FROM PORT C1
        BIT  6,A            ;CHECK IF SEND DATA SWITCH IS PRESSED.
        JR   Z,LED          ;WAIT UNTIL SWITCH IS PRESSED.
        DEC  HL             ;GET LAST ADDRESS OF IMAGE DATA (HL=0F7E4h)
        CALL SENDDATA       ;SEND DATA OUT TO IBM-PC
        LD   HL,RAM         ;LOAD ADDRESS OF RAM INTO REGISTER PAIR HL
        RET
        ;
        ;   ========================
        ;   SEND DATA OUT SUBROUTINE
        ;   ========================
        ;   REG = ABCDEL
        ;   PASS PARAMETER = HL
        ;
SENDDATA: LD   BC,800H      ;LOAD ADDRESS OF TOP OF IMAGE DATA
SEND1:  IN   A,(PortC2)     ;READ DATA FROM PORT C2
        BIT  3,A            ;TEST PC3 PUSH OR READY\
        JR   NZ,SEND1       ;BUSY, WAIT UNTIL READY\
        XOR  A              ;CLEAR REGISTER A
        SET  7,A            ;SET STROBE\=1
        OUT  (PortC2),A     ;SEND STROBE\=1 SIGNAL
        LD   A,(BC)         ;LOAD IMAGE DATA TO REGISTER A
        OUT  (PortA2),A     ;SEND IMAGE DATA OUT
        PUSH HL             ;SAVE LAST ADDRESS OF IMAGE DATA IN STACK
        CALL CMP16          ;CHECK BOTTOM OF IMAGE DATA
        POP  HL             ;GET LAST ADDRESS OF IMAGE DATA BACK FROM STACK
        JR   Z,SENDEND0     ;JUMP TO SEND LAST DATA
        INC  BC             ;INCREASE ADDRESS OF IMAGE DATA BY 1
        LD   A,00100000B    ;LOAD A WITH APDAT(APPEND DATA) CODE
        LD   (CTRLFILE0),A  ;MOVE APDAT INTO CTRLFILE0
        OUT  (PortC2),A     ;SEND STROBE\=0,AND CTRLFILE0=SOF,APDAT
        PUSH BC
        LD   B,1H
        CALL DELAY          ;WAIT FOR BUSY SIGNAL,DELAY FOR A WHILE
        POP  BC
        JR   SEND1          ;JUMP BACK TO SEND DATA AGAIN
SENDEND0: LD   A,(CTRLFILE1)  ;LOAD CTRLFILE1 TO REGISTER A
        AND  00010000B      ;CHECK EOP(END OF FILE) CODE
        JR   NZ,SENDEND1    ;YES, EOP WAS LOADED,JUMP TO SENDEND1
        LD   A,01110000B    ;LOAD A WITH EOD(END OF DATA) CODE
SENDEND1: LD   (CTRLFILE0),A  ;MOVE EOD INTO CTRLFILE0
        OUT  (PortC2),A     ;SEND STROBE\=0,AND CTRLFILE0=EOD
        RET
        ;
        ;   ==============================
        ;   COMPARE 16 BIT DATA SUBROUTINE
        ;   ==============================
```

```
;    REG = ABCHL
;    PASS PARAMETER = BCHL (HL NOT PASS VALUE OUT)
;
CMP16:  OR   A              ;CLEAR CARRY
        SBC  HL,BC          ;SUBTRACT SUBTRAHEND FROM MINUEND
        RET  PO             ;RETURN IF NO OVERFLOW
        LD   A,H            ;OVERFLOW THEN INVERT SIGN FLAG
        RRA                 ;SAVE CARRY IN BIT 7
        XOR  01000000B      ;COMPLEMENT BIT 6 (SIGN BIT)
        SCF                 ;ENSURE A NON-ZERO RESULT
        ADC  A,A            ;RESTORE CARRY, COMPLEMENTED SIGN
                            ;ZERO FLAG =0 FOR SURE
        RET
;
;    ================
;    SONG SUBROUTINE
;    ================
;    REG = BCDEHL IX
;    PASS PARAMETER = D IX
;
;    B: LOUDNESS
;    C: FREQUENCY OF NOTE
;    D: NUMBER OF NOTE IN THE SONG
;    IX: ADDRESS OF NOTE AND RHYTHM TABLE
;    HL: RHYTHM OF NOTE
;
SONG:   LD   L,0            ;RESET LOW BYTE RHYTHM TO BE ZERO.
        LD   H,(IX+0)       ;LOAD RHYTHM FROM TABLE
        LD   B,0A0H         ;LOAD LOUDNESS
        INC  IX             ;INCREASE IX TO GET FREQUENCY
        LD   C,(IX+0)       ;LOAD FREQUENCY TO REGISTER C
        CALL SOUND          ;SEND SOUND OUT
        INC  IX             ;INCREASE IX TO GET NEW RHYTHM
        DEC  D              ;DECREASE NUMBER OF NOTES LEFT
        JR   NZ,SONG        ;JUMP BACK TO PLAY SONG TILL NO NOTE LEFT
        RET
;
;    ================
;    SOUND SUBROUTINE
;    ================
;    REG = ABCDEHL
;    PASS PARAMETER = DEHL
;
;    B: LOUDNESS
;    C: SOUND FREQUENCY
;    HL: SOUND LENGHT
;
SOUND:  PUSH HL             ;SAVE TIME LENGHT OF SOUND IN STACK
        PUSH DE             ;SAVE NUMBER OF NOTE IN STACK
SOUND1: LD   D,0            ;RESET REGISTER D
        LD   A,B            ;LOAD LOUDNESS TO REGISTER A
        CALL SOUND2         ;TURN SOUND ON
        XOR  A              ;CLEAR LOUDNESS TO ZERO
        CALL SOUND2         ;TURN SOUND OFF
        DEC  D
        JR   NZ,SOUND1
        POP  DE             ;LOAD NUMBER OF NOTE BACK FROM STACK
        POP  HL             ;LOAD TIME LENGHT OF SOUND BACK FROM STACK
```

```
            RET
SOUND2:     OUT  (PortB1),A
            LD   B,C
SOUND3:     DEC  HL
            LD   A,H
            OR   L
            JR   NZ,SOUND4
            INC  D
SOUND4:     DEC  E
            JR   NZ,SOUND3
            RET
            ;
            ;    =====================
            ;    DELAY TIME SUBROUTINE
            ;    =====================
            ;    REG = BC
            ;    PASS PARAMETER = B
            ;
DELAY:      LD   C,99H
DLY:        DEC  C
            JR   NZ,DLY
            DJNZ DELAY
            RET
            ;
            ; ######## STEP-MOTOR DRIVE VALUE ########
            ;
XPDRIVE:    DFB  60H,0C0H,90H,30H
YPDRIVE:    DFB  06H,0C0H,09H,03H
            ;
            ; ############ SONG TABLE ############
            ;
            ; --- MEMORY SONG ---
            ; 50 NOTES (32h)
            ;
TABLE1:     DFB  TWO,DOH,R1_8,STP,TWO,DOH
            DFB  ONE,TE,ONE,DOH,ONE,REH,ONE,DOH,ONE,LA
            DFB  TWO,DOH,TWO,DOH,R1_4,STP
            DFB  ONE,TE,ONE,DOH,ONE,REH
            DFB  ONE,DOH,ONE,SOL,TWO,LA,TWO,LA,R1_4,STP
            DFB  ONE,FA,ONE,SOL,ONE,LA,ONE,SOL,ONE,FA,TEE,ME,HALF,STP
            DFB  ONE,ME,ONE,SOL,R1_8,STP,ONE,SOL,TWO,RE
            DFB  ONE,ME,ONE,FA,ONE,SOL,ONE,LA,R1_8,STP
            DFB  ONE,TE,ONE,DOH,HALF,TE,HALF,LA,TWO,SOL,HALF,STP
            DFB  ONE,ME,ONE,DO,TWO,SOL,R1_8,STP,TWO,SOL
            DFB  ONE,LA,ONE,DOH,TWO,DOH
            ;
            ; --- WALLS ICE-CREAM ---
            ; 12 NOTES (0Ch)
            ;
TABLE2:     DFB  TWO,ME,ONE,SOL,ONE,DO,TWO,STP
            DFB  TWO,LA,ONE,DOH,ONE,FA,TWO,STP
            DFB  TWO,TE,ONE,DOH,ONE,REH,ONE,DOH
            ;
            ; --- CHARIOTS OF FIRE ---
            ; 55 NOTES (37h)
            ;
TABLE3:     DFB  ONE,DO,ONE,ME,ONE,SOL,ONE,LA,TEE,SOL,ONE,ME,HALF,STP
            DFB  ONE,DO,ONE,ME,ONE,SOL,ONE,LA,TEE,SOL,HALF,STP
```

```
            DFB  ONE,DO,ONE,ME,ONE,SOL,ONE,LA,TEE,SOL,ONE,ME,HALF,STP
            DFB  ONE,ME,ONE,FA,ONE,ME,ONE,DO,ONE,DO,HALF,STP
            DFB  ONE,DOH,ONE,TE,ONE,LA,ONE,SOL,TWO,TE
            DFB  HALF,SOL,TWO,LA,ONE,SOL,HALF,STP
            DFB  ONE,DOH,ONE,TE,ONE,LA,ONE,SOL,TEE,TE,HALF,STP
            DFB  ONE,DOH,ONE,TE,ONE,LA,ONE,SOL,TWO,TE
            DFB  HALF,SOL,TWO,LA,ONE,SOL,HALF,STP
            DFB  ONE,ME,ONE,FA,ONE,ME,ONE,DO,ONE,DO
            ;
            ; (********** RAM AREA **********)
            ;
            ORG  0800H
            ;
            ; --- STORE IMAGE DATA ---
            ;
RAM:        DFS  0F7E5H          ;FROM 800h TO F7E4h(STORE IMAGE DATA)
EOD:                             ;START PROHIBITED AREA FOR IMAGE DATA
            ;
            ; --- STORE VARIABLE IN MONITOR PROGRAM ---
            ;
CTRLFILE0:  DFS  1H              ;STORE CONTROL CODE OF DATA FILE ON IBM PC
CTRLFILE1:  DFS  1H              ;STORE CONTROL CODE OF DATA FILE ON IBM PC
RightMar:   DFS  2H              ;STORE RIGHT MARGIN DATA
BOTTOM:     DFS  2H              ;STORE BOTTOM LINE DATA
xPOS:       DFS  2H              ;STORE X POSITION
yPOS:       DFS  2H              ;STORE Y POSITION
            ;
            ; --- STORE STACK ---
            ;
STACK:      DFS  10H             ;SYSTEM STACK AREA FROM FFEFh TO FFFFh
SYSSTK:                          ;SYSTEM STACK BOTTOM
            ;
            END
```

ภาคผนวก ข

โปรแกรมใช้งานบนเครื่องไมโครคอมพิวเตอร์

```
/*******************************************/
/*                                         */
/*        Main Program for SCANNER         */
/*                                         */
/*******************************************/

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

#define  FULL_GRAPH        0,0,639,349    /*     Full Screen       */
#define  DISPLAY_AREA      114,55,526,295 /*     Display Area      */
#define  ESC               0x1B           /*      Esc key          */
#define  ENTER             0x0D           /*     Enter key         */
#define  LINE_FEED         0x0A           /*   Line feed ascii     */
#define  MAXMENU           5              /*    Maximum menu       */
#define  MAXCHOICE         6              /*   Maximum choice      */
#define  MAX_MSG           80             /*   Maximum message     */
#define  TEXT_SCREEN       1,1,80,22      /*     Text screen       */
#define  TEXT_SCREEN2      1,2,80,22      /* (not include menu bar) */
#define  FULL_SCREEN       1,1,80,25      /*   Full text screen    */
#define  STATUS_WIN        3,23,78,25     /*      Status box       */
#define  PATH_POS          5,24           /*     Path position     */
#define  FILE_POS          40,24          /*     File position     */
#define  LPT1              0              /*     Printer port      */
#define  FUNCTION_00       0              /*  Function 0 of print  */
#define  FUNCTION_01       1              /*  Function 1 of print  */
#define  G_STATUS_WIN      0,330,639,349  /* Graphic status window */
#define  G_STATUS_POS      10,338         /* Status position in gr */
#define  X_POS             450,338        /* Xpos position in graph */
#define  Y_POS             500,338        /* Ypos position in graph */
#define  MODE_POS          550,338        /* Mode position in graph */
#define  PORTA             0x03EC         /*        Port A         */
#define  PORTB             0x03ED         /*        Port B         */
#define  PORTC             0x03EE         /*        Port C         */
#define  CONTROL_PORT      0x03EF         /*     Control port      */
#define  INITIAL_SIZE      0x8000         /* Initail size of image */
#define  STROBE            0x00           /*   Strobe active low   */
#define  HEAD_OF_IMG       0x40           /* Head of image status  */
#define  END_OF_IMG        0x10           /*  End of image status  */
#define  APPEND_IMG        0x20           /* Append of image status */
#define  END_OF_TX         0x70           /* End of sending status */
#define  EOLR              0x00           /*   End Of Line Right.  */
#define  EOLL              0x80           /*   End Of Line Left    */
#define  MAXX              639            /*   Maximum X in EGA    */
#define  MAXY              329            /*   Maximum Y in EGA    */
#define  TIME_OUT_BIT      0x0001         /*-----------------------*/
#define  IO_ERROR_BIT      0x0008         /*                       */
#define  SELECTED_BIT      0x0010         /*        Define         */
#define  PAPER_OUT_BIT     0x0020         /*        printer        */
#define  ACKNOWLEDGE_BIT   0x0040         /*        status         */
```

```
#define  NOT_BUSY_BIT      0x0080        /*       and errors        */
                                         /*-------------------------*/
#define  ERROR_BITS        (TIME_OUT_BIT|IO_ERROR_BIT|PAPER_OUT_BIT)
#define  PRINTER_OFF       (SELECTED_BIT|ACKNOWLEDGE_BIT)
#define  NOT_ON_LINE       (SELECTED_BIT|IO_ERROR_BIT)
#define  OUT_OF_PAPER      (SELECTED_BIT|PAPER_OUT_BIT)
#define  PRINTER_BUSY      (NOT_BUSY_BIT|PAPER_OUT_BIT)


void  SetScreen();                       /*      Setup screen       */
void  ShowDir();                         /*   Display current dir   */
void  PullDown();                        /*       Select menu       */
void  SetMenu();                         /*   Define menu varible   */
char  *GetCurDir();                      /*  Get Current Directory  */
void  ChangeDir();                       /*    Change Directory     */
void  OpenImage();                       /*     Open image file     */
void  SaveImage();                       /*     Save image file     */

void  ChangeMenu(char CurMenu,
      char Choice);                      /*     Change menu         */

void  ChangeChoice(char CurMenu,
      char OldChoice,char NewChoice);    /*    Change selection     */

void  Box(int FirstX,int FirstY,
      int SecondX,int SecondY);          /*       Draw box          */

char  HotKey(char CurMenu,char key);     /*   Check for hot key     */
void  DialogBox(char *Message);          /*     Draw Dialog Box     */
void  ErrorBox(char *Message);           /*     Draw Error Box      */
char  SetGraph();                        /*   Init graphics mode    */
void  Display(char Mode);                /*     Display image       */
void  Print();                           /*      Print Image        */
void  Edit();                            /*      Edit image         */
void  SetGraphScreen();                  /*  Setup graphic screen   */

void  UpdatePos(int Xpos,int Ypos,
      int Mode);                         /*  Update graphic screen  */

void  DisplayZoom(char Mode,
      char ZoomValue);                   /*     Display image       */

void  ConvertImage();                    /*  Convert img to *Image  */
void  Receive(char Is_Convert);          /*      Receive data       */
void  ConvertToIMG();                    /* Convert TempFile to IMG */
void  ClearDialogBox();                  /*    Clear Dialog box     */
void  ChangeTempFile();                  /*    Change Temp File     */
char  ImageWarn(char *warning);          /*   Image warning part    */
int   PrintByte(char ch,char function);  /* print a byte with check */
int   DefinePrintError(int status);      /* Define Error when print */


/*-------------------------- Global Variables ---------------------*/

struct                                   /*    Header IMG variable  */
```

```
        unsigned char  Password;
        char           Resolution;
        int            SizeX;
        int            SizeY;
      }Header;

struct   StructureMenu                  /* Store content of menu */
     {
       char     Name[MAX_MSG];
       char     Win[4];
       char     Msg[MAXCHOICE][MAX_MSG];
       char     LastChoice;
     };

struct   StructureMenu  Menu[MAXMENU];  /*     Menu variable     */

char          CurMenu;                  /* Current menu status   */

char          CurChoice[MAXMENU]
              ={0,0,0,0,0};

unsigned char *Image;                   /*     Image pointer     */
int           ImageSize=0x800;          /* Size of image in byte */
char          CurFileName
              [MAXFILE+MAXEXT]="NONE";
                                        /*   Current file name   */

char          TempFile[MAXFILE+MAXEXT]
              ="PICTURE.TMP";           /*    Temporary file     */

FILE          *CurFile;                 /*  Current File Pointer */
char          CurRes=1;                 /*   Current Resolution  */

char          *error_type[5]=
               {"Printer swich is off.",
                "Printer is out of paper.",
                "Printer is off line and cannot receive data.",
                "Printer is bust or printer queue is not empty.",
                "Printer is not ready."};
                                        /* Printer Error Message */


/*-------------------- Begining of main program --------------------*/
main()
{
  SetScreen();                          /*    Set up screen      */
  do{                                   /*   Loop until 'QUIT'   */
  PullDown();                           /*     Select menu       */
  switch(CurMenu)
  {
    case 0                              /*      File menu        */
    :switch(CurChoice[CurMenu])
    {
      case 0                            /*      Open File        */
      : window(FULL_SCREEN);
        OpenImage();
        ShowDir();
        break;
```

```
    case 1                          /*    Change Temp File    */
    : window(FULL_SCREEN);
      ChangeTempFile();
      break;
    case 2                          /*       Change Dir       */
    : window(FULL_SCREEN);
      textattr(YELLOW+(BLUE<<4));
      ChangeDir();
      ShowDir();
      break;
    case 3                          /*          Quit          */
    : textattr(LIGHTGRAY+(BLACK<<4));
      textmode(C80);
      clrscr();
      fcloseall();
      exit(1);
      break;
 }
 break;

 case 1                             /*      Display Menu      */
 :switch(CurChoice[CurMenu])
 {
   case 0                           /*     Normal Display     */
   : DisplayZoom(0,CurRes);         /*      (Mode = 1)        */
     restorecrtmode();
     SetScreen();
     break;
   case 1                           /*     Rotate Display     */
   : DisplayZoom(1,CurRes);         /*      (Mode = 0)        */
     restorecrtmode();
     SetScreen();
     break;
 }
 break;

 case 2                             /*      Print Menu        */
 :switch(CurChoice[CurMenu])
 {
   case 0                           /*     Normal Print       */
   :   window(FULL_SCREEN);
     Print();
     break;
 }
 break;

case 3                              /*      Set Res Menu      */
:switch(CurChoice[CurMenu])
{
   case 0                           /*    Set to file's Res   */
   :   CurRes=Header.Resolution;    =
     ShowDir();
     break;
   case 1                           /*    Set Res = 150 dpi   */
   : CurRes=1;
     ShowDir();
     break;
```

```
            case 2                          /*     Set Res = 75 dpi    */
            : CurRes=2;
              ShowDir();
              break;
            case 3                          /*     Set Res = 50 dpi    */
            : CurRes=3;
              ShowDir();
              break;
            case 4                          /*     Set Res = 30 dpi    */
            : CurRes=5;
              ShowDir();
              break;
         }
         break;

         case 4                             /*       Receive Menu      */
         :switch(CurChoice[CurMenu])
         {
           case 0                           /* Only Receive no convert */
           : Receive(0);
             break;
           case 1                           /*  Receive and convert    */
           : Receive(1);
             break;
         }
         break;

      }
      }while((CurMenu)||(CurChoice[CurMenu]
        !=(Menu[0].LastChoice-1)));         /*          QUIT          */
}
/*-------------------- End of main program ----------------------*/


void  Receive(char Is_Convert)              /* Is_Convert=1 : convert  */
                                            /*      =0 : not convert   */
{
  unsigned char     data,status;
  char              ch;
  struct            ffblk ffblk;
  FILE              *File;
  long    int       number=1;             /*   Number of data        */
  int               i;


  window(FULL_SCREEN);
  outportb(CONTROL_PORT,0x98);              /*   Send Control word     */
  DialogBox("  Press SEND switch (on SCANNER)  ");
  while(1)                                  /*     Inifinit loop       */
  {
    outportb(PORTC,0x00);                   /*      Send READY         */
    do                                      /*   Loop until STROBE/    */
      status=inportb(PORTC);                /*      Read status        */
    while((status&0x80)!=STROBE);
    outportb(PORTC,0x08);                   /*      Send BUSY          */
    data=inportb(PORTA);                    /* Read data from SCANNER */
    switch(status)                          /*      Check status       */
```

```
        {
          case HEAD_OF_IMG :              /*       New image       */
                                          /*   Open new temp file  */
            if((File=fopen(TempFile,"wb"))==NULL)
            {
              ErrorBox("Error open file!");
              return;
            }
            fwrite(&data,sizeof(unsigned char),1,File);
            for(i=0;i<=100;i++);
            number++;
            break;

          case APPEND_IMG :               /*   Continued image     */
                                          /*   Append temp file    */
            if(number==1)
              if((File=fopen(TempFile,"wb"))==NULL)
              {
                ErrorBox("Error open file!");
                return;
              }
            fwrite(&data,sizeof(unsigned char),1,File);
            for(i=0;i<=100;i++);
            number++;
            break;

          case END_OF_IMG :               /*     End  of Image     */
            fwrite(&data,sizeof(unsigned char),1,File);
            for(i=0;i<=100;i++);
            number++;
            fclose(File);
            DialogBox(" End of Image ");
            if(Is_Convert)                /*   Check for Convert   */
              ConvertToIMG();
            return;

          case END_OF_TX :                /*   End of Transmission */
            fwrite(&data,sizeof(unsigned char),1,File);
            for(i=0;i<=100;i++);
            number++;
            fclose(File);
            DialogBox(" End of transmission ");
            if(Is_Convert)                /*   Check for Convert   */
              ConvertToIMG();
            return;
        }
    }
}


void  ConvertToIMG()                      /* Convert TempFile to IMG */
{
  unsigned char     *data1,data2,count;
  char              ch,flag=0;
  struct            ffblk ffblk;
  FILE              *File1,*File2;
  int               index;
  char              FileName[MAXFILE+MAXEXT];
```

```
char            prompt_overwrite[MAX_MSG];
char            prompt_delete[MAX_MSG];
unsigned char   temp[1200];
long    int     num;
int             shift=0;


textattr(YELLOW+(BLUE<<4));
window(FULL_SCREEN);
DialogBox("Enter Image File (.IMG) :              ");
gotoxy(wherex()-15,wherey());
gets(FileName);                     /*   Enter IMG Filename   */
ClearDialogBox();
strupr(FileName);
if(!findfirst(FileName,&ffblk,0))   /*      if file exist     */
{
   strcpy(prompt_overwrite,"Overwrite ");
   strcat(prompt_overwrite,FileName);
   strcat(prompt_overwrite,"? (Y/N) ");
   DialogBox(prompt_overwrite);
   while((ch=toupper(getch()))!='Y')
      if(ch=='N')
        return;
}
findfirst(TempFile,&ffblk,0);       /*     Allocate memory    */
num=ffblk.ff_fsize;
data1=calloc(num,sizeof(unsigned char));
if(!data1)
{
   ErrorBox("Allocation Error!");
   return;
}
else
{                                   /*     Open temp  file    */
   if((File1=fopen(TempFile,"rb"))==NULL)
   {
      ErrorBox("Error open file!");
      return;
   }
   else
   {
      if(fread(data1,num,1,File1)!=1)  /*   Read data from file  */
      {
        ErrorBox("Error reading file !");
        return;
      }
   }
}

if((File2=fopen(FileName,"wb"))==NULL) /*      Open IMG file     */
{
   ErrorBox("Error open file!");
   return;
}
textattr(YELLOW+(BLUE<<4)+BLINK);
DialogBox("Convert image");
Header.SizeX=0;
Header.SizeY=0;
```

```
while(*data1!=EOLR)                    /*      Check for EOLR     */
{
   count=*data1&(0x7F);
   Header.SizeX=Header.SizeX+count;
   data1++;
   shift++;
}
index=Header.SizeX;
data1=data1-shift;                     /* Reset to begin of file  */
while(num>0)                           /* Check for same filesize */
{
  if(!flag)                            /*   Flag = 0  even line   */
  {
     while(*data1!=EOLR)
     {
        if(*data1!=EOLL)
        {
           data2=*data1;
                                       /*    Copy data temp->IMG  */
           fwrite(&data2,sizeof(unsigned char),1,File2);
           num--;
        }
        data1++;
     }
     flag=1;
  }
  else                                 /*   Flag = 1  odd line    */
  {
     index=Header.SizeX;
     while(*data1!=EOLL)
     {
        count=(*data1)&(0x7F);
        for(;count>0;count--)
        {
           index--;
                                       /* Copy data temp->matrix  */
           temp[index]=(*data1&0x80);
        }
        data1++;
     }
                                       /* Copy data matrix->IMG   */
                                       /*    in reverse order     */
     count=1;
     for(index=1;index<=Header.SizeX;index++)
     {
        if(temp[index]==temp[index-1])
        {
           if(count>=0x7F)
           {
              data2=temp[index]+count;
              fwrite(&data2,sizeof(unsigned char),1,File2);
              num--;
              count=0;
           }
           count++;
        }
        else
```

```
            {
               data2=temp[index-1]+count;
               fwrite(&data2,sizeof(unsigned char),1,File2);
               num--;
               count=1;
            }
         }
         flag=0;
      }
                                    /*  Add EOLR to IMG file  */
      HeaderL.SizeY++;
      data2=EOLR;
      fwrite(&data2,sizeof(unsigned char),1,File2);
      num--;
   }
   fclose(File1);
   Header.Password=0xB;
   Header.Resolution=1;
                                   /* Resolution  1 : 150 dpi
                                                  2 : 75  dpi
                                                  3 : 50  dpi
                                                  5 : 30  dpi */

   fwrite(&Header,6,1,File2);          /*   Write header to IMG  */
   fclose(File2);
   free(data1);
   textattr(YELLOW+(BLUE<<4));
   ClearDialogBox();
   strcpy(prompt_delete,"Do you want to delete ");
   strcat(prompt_delete,TempFile);
   strcat(prompt_delete,"? (Y/N) ");
   DialogBox(prompt_delete);
   while((ch=toupper(getch()))!='Y')
      if(ch=='N')
        return;
   unlink(TempFile);                   /*   Delete temp file     */
}


void  ClearDialogBox()                 /*   Clear Dialog box     */
{
   window(FULL_SCREEN);
   window(1,12,80,14);
   clrscr();
   window(FULL_SCREEN);
}


void  Box(int FirstX,int FirstY,int SecondX,int SecondY)
                                       /*        Draw box        */
{
   int      Row,Column;

                                       /*   Draw vertical line   */
   for(Row=FirstY+1;Row<=SecondY-1;Row++)
   {
      gotoxy(FirstX,Row);
      putch(179);
```

```
      gotoxy(SecondX,Row);
      putch(179);
   }
                                        /*  Draw horizontal line  */
   for(Column=FirstX+1;Column<=SecondX-1;Column++)
   {
      gotoxy(Column,FirstY);
      putch(196);
      gotoxy(Column,SecondY);
      putch(196);
   }
                                       /*       Draw angle       */
   gotoxy(FirstX,FirstY);
   putch(218);
   gotoxy(FirstX,SecondY);
   putch(192);
   gotoxy(SecondX,FirstY);
   putch(191);
   gotoxy(SecondX,SecondY);
   putch(217);
}


void  ErrorBox(char *Message)          /*     Draw Error Box     */
{
   DialogBox(Message);
   printf("\a");
   do {while(getch()!=ESC);            /*      Wait for ESC      */
}


void  DialogBox(char *Message)         /*     Draw Dialog Box    */
{
   int   MessageLenght;

   gotoxy(1,12);                       /*  Goto center of screen  */
   clreol();
   gotoxy(1,13);
   clreol();
   gotoxy(1,14);
   clreol();
   MessageLenght=strlen(Message);
   Box(40-MessageLenght/2-2,12,40+MessageLenght/2+2,14);
   gotoxy(40-MessageLenght/2,13);
   printf("%s",Message);
}


void  PullDown()                       /*       Select menu      */
{
   char     OldChoice;
   int      ch,i;

   SetMenu();                          /*   Define menu varible  */
   window(TEXT_SCREEN);
   gotoxy(1,1);
   printf("      File          Display         ");
   printf("Print         Resolution        Receive");
                                       /*    Draw first menu     */
```

```
    ChangeMenu(CurMenu,CurChoice[CurMenu]);
    do
    {
      if(!(ch=getch())) ch=getch();       /*   Check for arrow key   */
      switch(ch)
      {
        case 75 : CurMenu--;              /*        Left arrow       */
             if(CurMenu<0) CurMenu=MAXMENU-1;
             ChangeMenu(CurMenu,CurChoice[CurMenu]);
             break;

        case 77 : CurMenu++;              /*       Right arrow       */
             if(CurMenu>MAXMENU-1)  CurMenu=0;
             ChangeMenu(CurMenu,CurChoice[CurMenu]);
             break;

                                          /*        Up arrow         */
        case 72 : OldChoice=CurChoice[CurMenu];
             CurChoice[CurMenu]--;
             if(CurChoice[CurMenu]<0)
             CurChoice[CurMenu]=(Menu[CurMenu].LastChoice)-1;
             ChangeChoice(CurMenu,OldChoice,CurChoice[CurMenu]);
             break;

                                          /*       Down arrow        */
        case 80 : OldChoice=CurChoice[CurMenu];
             CurChoice[CurMenu]++;
             if(CurChoice[CurMenu]>((Menu[CurMenu].LastChoice)-1))
             CurChoice[CurMenu]=0;
             ChangeChoice(CurMenu,OldChoice,CurChoice[CurMenu]);
             break;

                                          /*       Others key        */
        default : OldChoice=CurChoice[CurMenu];
                                          /*    Check for Hot key    */
             if(HotKey(CurMenu,ch)!=-1)
             {
               CurChoice[CurMenu]=HotKey(CurMenu,ch);
               ChangeChoice(CurMenu,OldChoice,
                      CurChoice[CurMenu]);
               ch=ENTER;
             }
             break;
      }
    }while(ch!=ENTER);                    /*   Until press ENTER    */
}


char  HotKey(char CurMenu,char key)       /*   Check for Hot key    */
                                          /*  Return to CurChoice   */
{
  char    i;

  for(i=0;i<=Menu[CurMenu].LastChoice;i++)
  {
    if(toupper(key)==Menu[CurMenu].Msg[i][1])
       return   i;                        /*  Yes! it's a Hot key   */
  }
  return   -1;                            /*    It is'n Hot Key     */
}
```

```
void  ChangeChoice(char CurMenu,char OldChoice,char NewChoice)
                                        /*   Change menu's choice  */
{
   textattr(BLACK+(LIGHTGRAY<<4));
   gotoxy(2,OldChoice+2);
   cputs(Menu[CurMenu].Msg[OldChoice]);
   textcolor(RED);
   gotoxy(3,OldChoice+2);
   putch(Menu[CurMenu].Msg[OldChoice][1]);
   textattr(YELLOW+(BLACK<<4));
   gotoxy(2,NewChoice+2);
   cputs(Menu[CurMenu].Msg[NewChoice]);
   textattr(BLACK+(LIGHTGRAY<<4));
}

void  ChangeMenu(char CurMenu,char Choice)
                                       /*   Change menu heading   */
{
   char     i;

   textattr(YELLOW+(BLUE<<4));
   window(TEXT_SCREEN2);
   clrscr();
   textattr(BLACK+(LIGHTGRAY<<4));
   window(TEXT_SCREEN);
   Box(Menu[CurMenu].Win[0],Menu[CurMenu].Win[1],
       Menu[CurMenu].Win[2],Menu[CurMenu].Win[3]);
   window(Menu[CurMenu].Win[0],Menu[CurMenu].Win[1],
        Menu[CurMenu].Win[2],Menu[CurMenu].Win[3]);
   for(i=0;i<Menu[CurMenu].LastChoice;i++)
   {
     if(i==Choice)                     /*  Hight light on choice  */
     {
        textattr(YELLOW+(BLACK<<4));
             gotoxy(2,i+2);
        cputs(Menu[CurMenu].Msg[i]);
        textattr(BLACK+(LIGHTGRAY<<4));
     }
     else
     {
        gotoxy(2,i+2);
        cputs(Menu[CurMenu].Msg[i]);
        textcolor(RED);
        gotoxy(3,i+2);
        putch(Menu[CurMenu].Msg[i][1]);
        textattr(BLACK+(LIGHTGRAY<<4));
     }
   }
}


void  SetMenu()                        /*   Define menu varible   */
{

/*---------------------- Menu  Pile -------------------------*/

        strcpy(Menu[0].Name," Pile ");
        Menu[0].LastChoice=4;
```

```
        strcpy(Menu[0].Msg[0]," Open File  ");
        strcpy(Menu[0].Msg[1]," Temp File  ");
        strcpy(Menu[0].Msg[2]," Change Dir ");
        strcpy(Menu[0].Msg[3]," Quit       ");

        Menu[0].Win[0]=3;
        Menu[0].Win[1]=2;
        Menu[0].Win[2]=Menu[0].Win[0]+strlen(Menu[0].Msg[0])+1;
        Menu[0].Win[3]=Menu[0].Win[1]+Menu[0].LastChoice+1;



/*---------------------- Menu  Display ----------------------*/

        strcpy(Menu[1].Name," Display ");
        Menu[1].LastChoice=2;

        strcpy(Menu[1].Msg[0]," Normal ");
        strcpy(Menu[1].Msg[1]," Rotate ");

        Menu[1].Win[0]=21;
        Menu[1].Win[1]=2;
        Menu[1].Win[2]=Menu[1].Win[0]+strlen(Menu[1].Msg[0])+1;
        Menu[1].Win[3]=Menu[1].Win[1]+Menu[1].LastChoice+1;



/*---------------------- Menu  Print ----------------------*/
        strcpy(Menu[2].Name," Print ");


        strcpy(Menu[2].Msg[0]," Print IMG  ");

        Menu[2].LastChoice=1;

        Menu[2].Win[0]=36;
        Menu[2].Win[1]=2;
        Menu[2].Win[2]=Menu[2].Win[0]+strlen(Menu[2].Msg[0])+1;
        Menu[2].Win[3]=Menu[2].Win[1]+Menu[2].LastChoice+1;



/*---------------------- Menu  Set Res ----------------------*/

        strcpy(Menu[3].Name," Set Res ");

        strcpy(Menu[3].Msg[0]," Res of File ");
        strcpy(Menu[3].Msg[1]," 150 dpi     ");
        strcpy(Menu[3].Msg[2]," 75  dpi     ");
        strcpy(Menu[3].Msg[3]," 50  dpi     ");
        strcpy(Menu[3].Msg[4]," 30  dpi     ");

        Menu[3].LastChoice=5;
```

```
        Menu[3].Win[0]=50;
        Menu[3].Win[1]=2;
        Menu[3].Win[2]=Menu[3].Win[0]+strlen(Menu[3].Msg[0])+1;
        Menu[3].Win[3]=Menu[3].Win[1]+Menu[3].LastChoice+1;



/*---------------------- Menu Recieve ----------------------*/

        strcpy(Menu[4].Name," Recieve ");

        strcpy(Menu[4].Msg[0]," Only Receive      ");
        strcpy(Menu[4].Msg[1]," Receive & Convert ");
        strcpy(Menu[4].Msg[2]," Convert File      ");

        Menu[4].LastChoice=3;

        Menu[4].Win[0]=58;
        Menu[4].Win[1]=2;
        Menu[4].Win[2]=Menu[4].Win[0]+strlen(Menu[4].Msg[0])+1;
        Menu[4].Win[3]=Menu[4].Win[1]+Menu[4].LastChoice+1;



/*---------------------- End of SetMenu ----------------------*/
}


void  SetScreen()                       /*     Setup screen      */
{
  textmode(C80);
  textattr(YELLOW+(BLUE<<4));
  clrscr();
  window(FULL_SCREEN);
  ShowDir();
}


void  ShowDir()                         /*  Display current dir  */
                                        /*     and image status  */
{
  char     Path[MAX_MSG];
  char     buf[MAX_MSG];

  textattr(YELLOW+(BLUE<<4));
  window(STATUS_WIN);
  clrscr();
  window(FULL_SCREEN);
  Box(STATUS_WIN);
  textattr(YELLOW+(BLUE<<4));
  gotoxy(PATH_POS);
  strcat(getcwd(Path,MAX_MSG),"    ");
  strcat(Path,CurFileName);
  if(strcmp(CurFileName,"NONE"))
  {
    strcat(Path,"   RES:");
    strcat(Path,itoa((150/Header.Resolution),buf,10));
```

```
        strcat(Path,"  SIZE: ");
        strcat(Path,gcvt((Header.SizeX/150.0*Header.Resolution),4,buf));
        strcat(Path,"x");
        strcat(Path,gcvt((Header.SizeY/150.0*Header.Resolution),4,buf));
        strcat(Path,"   CUR RES:");
        strcat(Path,itoa((150/CurRes),buf,10));
      }
      cputs(Path);
}


void ChangeTempFile()                   /*    Change Temp file     */
{
   char     prompt_change[MAX_MSG];
   char     FileName[MAXFILE+MAXEXT];

   window(FULL_SCREEN);
   textattr(YELLOW+(BLUE<<4));
   strcpy(prompt_change,"Change TempFile from ");
   strcat(prompt_change,TempFile);
   strcat(prompt_change," to :                ");
   DialogBox(prompt_change);
   gotoxy(wherex()-15,wherey());
   gets(FileName);
   strupr(FileName);
   if(strcmp(FileName,""))
     strcpy(TempFile,FileName);
   ClearDialogBox();
}

void ChangeDir()                        /*    Change Directory     */
{
   char  i,CurDrive,Path[MAX_MSG],UpPath[MAX_MSG];

   DialogBox("New Directory :                                  ");
   gotoxy(wherex()-30,wherey());
   gets(Path);                          /*       Read new Path     */
   strupr(Path);                        /* Transform to Uppercase  */
   if(Path[1]==':')                     /*  Check for new drive    */
     setdisk(Path[0]-65);
   if(chdir(Path)==-1)                  /*     Change Directory    */
     ErrorBox("   Path not found.   Press ESC.  ");
}


void OpenImage()                        /*    Open Image file      */
{
   char     FileName[MAXFILE+MAXEXT];
   struct   ffblk ffblk;                /*   Information Of File   */

   textattr(YELLOW+(BLUE<<4));
   DialogBox(" Open Image File :              ");
   gotoxy(wherex()-15,wherey());
   gets(FileName);                      /*     Input file name     */
   strupr(FileName);
   if(*FileName!=0x00)
   {
```

```
    if(!findfirst(FileName,&ffblk,0))
    {
      fclose(CurFile);
      if((CurFile=fopen(FileName,"rb"))==NULL)
      {
        ErrorBox("Error open file!");
        return;
      }
      else
      {
        fseek(CurFile,-6,SEEK_END);
        if(fread(&Header,6,1,CurFile)!=1)
        {
          ErrorBox("Error reading file !");
          fclose(CurFile);
          return;
        }
        if(Header.Password!=0xB)      /*   Check for IMG file   */
        {
          ErrorBox("This is not IMG file !");
          fclose(CurFile);
          return;
        }
        strcpy(CurFileName,FileName);
      }
    }
    else
      ErrorBox(" File not found!  Press Esc. ");
  }
}


char SetGraph()                       /* Initialize Graphic Mode */
{
  int   GraphDriver,GraphMode,GraphError;
  char ErrorMsg[MAX_MSG];

  detectgraph(&GraphDriver,&GraphMode); /* Detect Graphic Adapter  */
  if(GraphDriver<0)
  {
    ErrorBox("No graphic hardware detected !\n");
    return(-1);
  }
  if ((GraphDriver!=EGA)&&(GraphDriver!=VGA))
  {
    ErrorBox("This Program require EGA or VGA card !\n");
    return(-1);
  }
  initgraph(&GraphDriver,&GraphMode,""); /*   Set Graphic Mode     */
  GraphError=graphresult();             /*   Get Error Massege    */
  if(GraphError<0)
  {
    strcpy(ErrorMsg,"initgraph error: ");
    strcat(ErrorMsg,grapherrormsg(GraphError));
    return(-1);
  }
  return(1);
}
```

```
void  SetGraphScreen()                  /*  Setup graphic screen   */
{
   char     Status[MAX_MSG];
   char     buf[MAX_MSG];

   rectangle(G_STATUS_WIN);
   getcwd(Status,MAXPATH);
   strcat(Status," ");
   strcat(Status,CurFileName);
   strcat(Status," ");
   strcat(Status,Menu[CurMenu].Name);
   strcat(Status,"   RES:");
   strcat(Status,itoa((150/Header.Resolution),buf,10));
   strcat(Status,"  SIZE: ");
   strcat(Status,gcvt((Header.SizeX/150.0*Header.Resolution),4,buf));
   strcat(Status,"x");
   strcat(Status,gcvt((Header.SizeY/150.0*Header.Resolution),4,buf));
   strcat(Status,"   CUR RES:");
   strcat(Status,itoa((150/CurRes),buf,10));

   outtextxy(G_STATUS_POS,Status);
}


char  ImageWarn(char *warning)          /*  Image warning part    */
{
   char        prompt[MAX_MSG]=" Display anyway (Y/N) ";
   char        ch;

   ErrorBox(warning);
   DialogBox(prompt);
   while((ch=toupper(getch()))!='Y')
      if(ch=='N')
         return -1;                     /*      Not display       */
   return 1;                            /*    Display anyway      */

}


void DisplayZoom(char Mode,char ZoomValue)/*  Display image and Zoom */
{
   int         Xpos,Ypos;
   char        temp[MAXX];
   int         index;
   unsigned char  data,count;
   char        warning[MAX_MSG];

   window(FULL_SCREEN);
   textattr(YELLOW+(BLUE<<4));
   fseek(CurFile,0,SEEK_SET);
   strcpy(warning,"Can't Display ");
   strcat(warning,CurFileName);

   if(!((Header.SizeX<MAXX*ZoomValue)     /* Check if image too big  */
      &&(Header.SizeY<MAXY*ZoomValue)))
   {
      if(!((Header.SizeX<MAXY*ZoomValue)
         &&(Header.SizeY<MAXX*ZoomValue)))
```

```
    {
      strcat(warning,
      " with both mode in one screen !  Try another res.");
      if(ImageWarn(warning)==-1)
         return;
    }
    else
    {
      strcat(warning,
      " with Normal mode in one screen !  Try Rotate mode.");
      if(ImageWarn(warning)==-1)
         return;
    }
  }
  else
  {
    if(!((Header.SizeX<MAXX*ZoomValue)
      &&(Header.SizeY<MAXX*ZoomValue)))
    {
      strcat(warning,
      " with Rotate mode in one screen !  Try Normal mode.");
      if(ImageWarn(warning)==-1)
         return;

    }
  }

  SetGraph();
  SetGraphScreen();
  if(!Mode)                          /* if mode==0 Normal Display */
  {
    for(index=0;index<=MAXX;index++)
      temp[index]=0;
    Ypos=0;
    while((Ypos<=(MAXY*ZoomValue))&&(!feof(CurFile)))
    {
      for(Xpos=0;Xpos<=(MAXX*ZoomValue);)
      {
        fread(&data,sizeof(unsigned char),1,CurFile);
        count=data&0x7F;
        if(data==EOLR)
           break;
        for(;count>0;count--)
        {
          if((data&0x80)==0x80)
             temp[Xpos/ZoomValue]++;
          Xpos++;
        }
        if(Xpos>(MAXX*ZoomValue))
        {
          while((data!=EOLR)&&(!feof(CurFile)))
             fread(&data,sizeof(unsigned char),1,CurFile);
          break;
        }
      }
      Ypos++;
      if(!(Ypos%ZoomValue))
```

```
        {
          for(index=0;index<=MAXX;index++)
            if(temp[index]>(ZoomValue*2/3))
              putpixel(index,Ypos/ZoomValue,WHITE);
          for(index=0;index<=MAXX;index++)
            temp[index]=0;
        }
      }
    }
    else                            /* if mode!=0 Rotate Display */
    {
      for(index=0;index<=MAXY;index++)
        temp[index]=0;
      Xpos=0;
      while((Xpos<=(MAXX*ZoomValue))&&(!feof(CurFile)))
      {
        for(Ypos=(MAXY*ZoomValue);Ypos>=0;
        {
          fread(&data,sizeof(unsigned char),1,CurFile);
          count=data&0x7F;
          if(data==EOLR)
            break;
          for(;count>0;count--)
          {
            if((data&0x80)==0x80)
              temp[Ypos/ZoomValue]++;
            Ypos--;
          }
          if((Ypos<=0)&&(data!=EOLR))
          {
            while((data!=EOLR)&&(!feof(CurFile)))
              fread(&data,sizeof(unsigned char),1,CurFile);
            break;
          }
        }
        Xpos++;
        if(!(Xpos%ZoomValue))
        {
          for(index=0;index<=MAXY;index++)
            if(temp[index]>(ZoomValue*2/3))
              putpixel(Xpos/ZoomValue,index,WHITE);
          for(index=0;index<=MAXY;index++)
            temp[index]=0;
        }
      }
    }

    printf("\a");
    getch();
}


int   DefinePrintError(int status)        /* Define Error when print */
{
    int   tem=status;

    if((tem &= PRINTER_OFF)==PRINTER_OFF)
      return 0;
```

```
      else
      {
        tem = status;
        if((tem &= OUT_OF_PAPER)==OUT_OF_PAPER)
          return 1;
        else
        {
          tem = status;
          if((tem &= NOT_ON_LINE)==NOT_ON_LINE)
            return 2;
          else
          {
            tem = status;
            if((tem &= PRINTER_BUSY)==PRINTER_BUSY)
              return 3;
            else
              return 4;
          }
        }
      }
    }


int   PrintByte(char ch,char function)    /* Print a byte with check */
{
   unsigned    status;
   char     ans;
   char     PrinterErrorMsg[MAX_MSG]=" Error: ";

   status=biosprint(function,ch,LPT1);
   if(status&ERROR_BITS)
   {
     textattr(YELLOW+(BLUE<<4));
     strcat(PrinterErrorMsg,error_type[DefinePrintError(status)]);
     ErrorBox(PrinterErrorMsg);
     return 1;
   }
   return 0;
}


void Print()                          /*      Print image      */
{
   unsigned char  Data;
   unsigned char  Matrix[1201][17];
   int            i,j,count;
   unsigned char  Value;


   PrintByte(FUNCTION_01,0);              /*  Initialize printer    */

   PrintByte(FUNCTION_00,ESC);   /* Set Line Spacing to 8/72 inches  */
   PrintByte(FUNCTION_00,65);
   PrintByte(FUNCTION_00,8);

   PrintByte(FUNCTION_00,ENTER);          /*      New line          */
   PrintByte(FUNCTION_00,LINE_FEED);
```

```
PrintByte(FUNCTION_00,ESC);   /* Set graphic print in 960 byte   */
PrintByte(FUNCTION_00,76);
PrintByte(FUNCTION_00,192);
PrintByte(FUNCTION_00,3);


for(i=1;i<=1200;i++)
   for(j=1;j<=16;j++)
      Matrix[i][j]=0;                 /*       Clear Matrix      */

fseek(CurFile,0,SEEK_SET);
fread(&Data,1,1,CurFile);

while(!feof(CurFile))                 /* Loop until end of file */
{
   i=j=1;
   while(j<=16)
   {
      if(Data==0x00)
      {
         j++;
         i=1;
      }
      count=Data&0x7F;
      for(;count>0;count--)
      {
         Matrix[i][j]=Data&0x80;
         i++;
      }
      fread(&Data,1,1,CurFile);
      continue;
   }
   for(i=1;i<=1200;i++)
   {
      if(i%5)
      {
         Value=Matrix[i][1]+          /* Convert data to print  */
            (Matrix[i][3]>>1)+
            (Matrix[i][5]>>2)+
            (Matrix[i][7]>>3)+
            (Matrix[i][9]>>4)+
            (Matrix[i][11]>>5)+
            (Matrix[i][13]>>6)+
            (Matrix[i][15]>>7);

         PrintByte(FUNCTION_00,Value);
      }
   }

   PrintByte(FUNCTION_00,ENTER);      /*       New line         */
   PrintByte(FUNCTION_00,LINE_FEED);

   PrintByte(FUNCTION_00,ESC);   /* Set graphic print in 960 byte */
   PrintByte(FUNCTION_00,76);
   PrintByte(FUNCTION_00,192);
   PrintByte(FUNCTION_00,3);

}
```

ภาคผนวก ค

ฟังก์ชันแก้ไขและแปลงไฟล์ข้อมูลภาพเป็นมาตรฐานพีซีเอ็กซ์

```
/*************************************************************/
/*                                                           */
/*    ROUTINE FOR EDIT & CONVERT IMAGE FILE TO PCX FORMAT    */
/*                                                           */
/*************************************************************/

#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>

#define  FULL_GRAPH     0,0,639,349          /*     Full Screen       */
#define  ESC            0x1B
#define  MAX_MSG        80

char  SetGraph();                            /*   Init graphics mode  */
void  ConvertImage2();

main()
{
   struct      viewporttype     viewinfo;
   int         view_width,view_hight;
   int         Xpos=1,Ypos=1,Mode=WHITE,oldcolor;
   char        ch;

   SetGraph();
   setviewport(FULL_GRAPH,1);
   getviewsettings(&viewinfo);
   view_width=viewinfo.right-viewinfo.left;
   view_hight=viewinfo.bottom-viewinfo.top;
   do
   {
     if(!(ch=getch())) ch=getch();          /* Check for arrow key */
     if(Mode==WHITE)   putpixel(Xpos,Ypos,oldcolor);
     switch(ch)
     {
/* Left */  case 75 : Xpos--;
               if(Xpos<0) Xpos=0;
               break;
/* Right */ case 77 : Xpos++;
               if(Xpos>view_width) Xpos=view_width;
               break;
/* Up */    case 72 : Ypos--;
               if(Ypos<0) Ypos=0;
               break;
/* Down */  case 80 : Ypos++;
               if(Ypos>view_hight) Ypos=view_hight;
               break;
        default :
               if(toupper(ch)=='A')          /* Draw */
                  Mode=YELLOW;
               else
               if(toupper(ch)=='Z')          /* Erase */
                  Mode=BLACK;
               else
                  Mode=WHITE;
               break;
     }
```

```
        if(Mode==WHITE)
        {
           oldcolor=getpixel(Xpos,Ypos);
           putpixel(Xpos,Ypos,Mode);
        }
        else
           putpixel(Xpos,Ypos,Mode);
     }while(ch!=ESC);
     ConvertImage2();
     restorecrtmode();
}


void  ConvertImage2()
{
    /* define struct header file */
   struct   {
        unsigned char  Password;
        unsigned char  Version;
        unsigned char  Encoding;
        unsigned char  BPP;
        int         WinDemX1;
        int         WinDemY1;
        int         WinDemX2;
        int         WinDemY2;
        int         HoriResolut;
        int         VertResolut;
        unsigned char  ColorMap[16][3];
        unsigned char  reserve;
        unsigned char  NumofPlane;
        int         BytePLine;
        int         PaletteInfo;
        }  Header;

   FILE        *FilePCX;

   struct      viewporttype     viewinfo;
   int         view_width,view_hight;
   int         Xpos,Ypos;
   unsigned  char Full=0xFF,None=0xC1,first=1;
   unsigned char  value=0,older=0,byte_count=1,oldest=0;
   int         count=0;
   unsigned    offset=0;


   Header.Password=0xA;
   Header.Version=3;
   Header.Encoding=1;
   Header.BPP=1;
   Header.WinDemX1=0;
   Header.WinDemY1=0;
   Header.WinDemX2=639;
   Header.WinDemY2=479;
   Header.HoriResolut=640;
   Header.VertResolut=479;
   Header.reserve=0;
   Header.NumofPlane=1;
   Header.BytePLine=80;
```

```
FilePCX=fopen("test.pcx","wb");
if(FilePCX==NULL)
{
   printf("\a");
   exit(0);
}
fwrite(&Header,70,1,FilePCX);
fseek(FilePCX,128,SEEK_SET);

getviewsettings(&viewinfo);
view_width=viewinfo.right-viewinfo.left;
view_hight=viewinfo.bottom-viewinfo.top;

for(Ypos=0;Ypos<=view_hight;Ypos++)
{
   for(Xpos=0;Xpos<=79;Xpos++)
   {
        value=peekb(0xA000,offset++);
        if(first)
        {
           oldest=older;
           older=value;
           first=0;
           continue;
        }
        if(older==value)
        {
           byte_count++;
           if(byte_count>0x3F)
           {
              fwrite(&Full,1,1,FilePCX);
              fwrite(&older,1,1,FilePCX);
              byte_count=1;
           }
        }
        else
           if((oldest==older)&&(older!=value))
           {
              byte_count=byte_count+0xC0;
              fwrite(&byte_count,1,1,FilePCX);
              fwrite(&older,1,1,FilePCX);
              byte_count=1;
           }
           else
              if((oldest!=older)&&(older!=value))
              {
                 if(older>0xC0)
                 {
                    fwrite(&None,1,1,FilePCX);
                    fwrite(&older,1,1,FilePCX);
                 }
                 else
                    fwrite(&older,1,1,FilePCX);
              }
        oldest=older;
        older=value;
        value=0;
   }
}
```

```
    if(byte_count>1)
    {
       byte_count=byte_count+0xC0;
       fwrite(&byte_count,1,1,FilePCX);
       fwrite(&older,1,1,FilePCX);
    }
    else
       fwrite(&older,1,1,FilePCX);
    fclose(FilePCX);
    printf("\a");
}

char SetGraph()                         /* Initialize Graphic Mode */
{
   int   GraphDriver,GraphMode,GraphError;
   char ErrorMsg[MAX_MSG];

   detectgraph(&GraphDriver,&GraphMode);   /* Detect Graphic Adapter */
   if(GraphDriver<0)
   {
      return(-1);
   }
   if ((GraphDriver!=EGA)&&(GraphDriver!=VGA))
   {
      return(-1);
   }
   initgraph(&GraphDriver,&GraphMode,"");  /* Set Graphic Mode   */
   GraphError=graphresult();               /* Get Error Massege  */
   if(GraphError<0)
   {
      strcpy(ErrorMsg,"initgraph error: ");
      strcat(ErrorMsg,grapherrormsg(GraphError));
      return(-1);
   }
  return(1);
}
```