ปีการศึกษา 2532

การใช้เครื่องพิมพ์สแกนภาพ

(USEING PRINTER AS IMAGE-SCANNER)

โดย

นาย จำลอง พิมพ์สวัสดิ์ 30.3402

นาย อดุลย์ ทุมานุสรณ์ 30.3433

อาจารย์ที่ปรึกษา

ดร. ไพศาล นาคนิยม

ปริญญานิพนธ์ ปีการศึกษา 2532

ภาควิชา เทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง การใช้เครื่องพิมพ์สแกนภาพ (USING PRINTER AS IMAGE-SCANNER)

ผู้จัดทำ

นาย จำลอง พิมพ์สวัสดิ์ 30.3402

นาย อดุลย์ ทุมานุสรณ์ 30.3433

.....อาจารย์ที่ปรึกษา

(.....)

.....อาจารย์ที่ปรึกษา

(.....)

.....อาจารย์ที่ปรึกษา

(.....)

.....อาจารย์ที่ปรึกษา

(.....)

การใช้เครื่องพิมพ์สแกนภาพ

จำลอง พิมพ์สวัสดิ์
อดุลย์ ทุมานุสรณ์

ดร. ไพศาล นาคนิพัฒน์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2532

บทคัดย่อ

ในปัจจุบันการพัฒนาอุปกรณ์ภายนอกที่สามารถนำมาใช้งานร่วมกับเครื่องไมโครคอมพิวเตอร์ได้ก้าวหน้าไปอย่างรวดเร็วมาก แต่ทว่าอุปกรณ์เหล่านี้มักมีราคาสูงจึงไม่เหมาะสำหรับงานที่มีงบประมาณจำกัด ดังนั้นการทดลองวิจัยชิ้นงานนี้จึงมุ่งเน้นในด้านราคาและขณะเดียวกันก็สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพด้วย

การพัฒนาและวิจัยชิ้นงานนี้ได้กระทำอย่างต่อเนื่องและสามารถพัฒนาส่วนของหัวอ่านที่ใช้กับระบบสแกนภาพนี้ได้ถึง 2 รุ่น โดยการพัฒนาหัวอ่านในแต่ละรุ่นจะมุ่งเน้นประสิทธิภาพที่สูงขึ้นและการนำไปประยุกต์ใช้งานได้สะดวกขึ้น ลักษณะโดยทั่วไปของเครื่องสแกนภาพโดยใช้เครื่องพิมพ์จะประกอบด้วยส่วนอิเล็กทรอนิคส์ที่ติดกับหัวอ่านของเครื่องพิมพ์ ซึ่งใช้สำหรับอ่านข้อมูลแล้วจึงนำข้อมูลผ่านเครื่องไมโครคอมพิวเตอร์และนำไปแสดงผลบนจอมอนิเตอร์ ภาพที่ได้จะมีความละเอียด 40x40 จุดต่อตารางนิ้ว โดยภาพที่ได้จากหัวอ่านรุ่นแรกจะมีระดับความเข้ม 3 ระดับ ส่วนภาพจากการใช้หัวอ่านรุ่นที่สองจะมีระดับความเข้มเพิ่มเป็น 8 ระดับ นอกจากนี้ยังสามารถเก็บข้อมูลภาพลงในแผ่นดิสค์เก็ตเพื่อนำไปประยุกต์ใช้งานอื่นต่อไป

USING PRINTER AS IMAGE-SCANNER

Chumlong Pimsawad

Adul Tumanusorn

Dr. Paisal Nakpipap Advisor

1989

ABSTRACT

At present the development of computer s peripheral devices is in progress. However the expense for these devices is so costly such that ordinary computer users can not afford it. As the result after facing this problem this project aims at developing a low-cost and efficient image scanner.

The continuous development and research has brought about two versions of seanner head. The increased number of version implies a more efficient and more applicable new scanner. The main part of the image seanner is its scanner head, which is attached to the printer's head. After reading in data, the computer will conseguently process and present the image off the moniter, with 40x40 dot-per-inch resolution. The scannned image will consist of three-lleveled color in the first version, and is incerased to eight levels of gray scale in the improved version. Moreover, the implemented program is able to save the image into the diskette for further application.

สารบัญ

บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎีและหลักการทำงาน	3
บทที่ 3 หลักการทำงานและโปรแกรมควบคุมของหัวอ่านรุ่นที่ 1	11
บทที่ 4 หลักการทำงานและโปรแกรมควบคุมของหัวอ่านรุ่นที่ 2	27
บทที่ 5 การทดลองและผลการทดลอง	39
บทที่ 6 สรุปผลการทดลองและข้อเสนอแนะ	46
ภาคผนวก ก.	48
ภาคผนวก ข.	54
ภาคผนวก ค.	69
ภาคผนวก ง.	75
หนังสืออ้างอิง	78

บทที่ 1 บทนำ

ระบบคอมพิวเตอร์ในปัจจุบันพัฒนาก้าวไปอย่างรวดเร็วจากเดิม ซึ่งมีการประมวลผลด้วยข้อความ (TEXT PROCESSING) ได้พัฒนามาสู่การประมวลผลด้วยรูปภาพ (IMAGE PROCESSING) เนื่องจากการประมวลผลด้วยรูปภาพนั้น ง่ายต่อการทำความเข้าใจ เช่น ระบบค้นหาประวัติพนักงาน ถ้ามีรูปภาพประกอบก็จะทำให้สามารถระบุบุคคลได้ทันทีที่เห็นจากรูปภาพ ดังมีคำกล่าวไว้ว่า " ภาพ 1 ภาพมีความหมายเท่ากับคำ 1000 คำ " ดังนั้น นับวันภาพจึงทวีความสำคัญมากกับระบบคอมพิวเตอร์ ภาพซึ่งนำมาใช้ได้กับระบบคอมพิวเตอร์นั้น ในปัจจุบันมีได้หลายวิธี เช่น ใช้กล้องถ่ายวีดีโอและแปลงจากสัญญาณอนาล็อกเป็นดิจิตอลเข้าสู่คอมพิวเตอร์ และอีกวิธีหนึ่งซึ่งในปัจจุบันนิยมใช้กันมากคือ การสแกนรูปโดยใช้เครื่องสแกนภาพ ซึ่งได้มีการพัฒนาไปมาก ทั้งด้านความละเอียด, ความเร็ว และระดับความเข้มที่ได้มากขึ้น เพราะได้มีการนำเทคนิค CCD (CHARGE-COUPLED DEVICES) มาใช้อย่างแพร่หลายและราคาก็ถูกลงมาก แต่ถึงกระนั้นก็ตาม ราคาก็ยังแพงอยู่มากคือประมาณเกือบครึ่งหนึ่งของราคาคอมพิวเตอร์ ดังนั้นจึงมีผู้ใช้น้อยมาก นอกจากงานที่ภาพมีความสำคัญมากๆในการประมวลผล เช่น ด้านการพิมพ์ (DESKTOP PUBLISHING) ระบบข้อมูลภาพ (PICTUREBASE) เป็นผลให้เครื่องสแกนภาพจำกัดอยู่ในวงแคบ โดยเฉพาะในการวิจัยค้นคว้าเกี่ยวกับการประมวลผลด้วยรูปภาพ เช่น การหาเส้นขอบรูป, PATTERNRECOGNITION แล้ว เครื่องสแกนภาพนับเป็นสิ่งสำคัญมาก แต่เนื่องจากเครื่องมีราคาแพงจึงมีใช้น้อยแม้ในมหาวิทยาลัยเองก็ยังไม่มีให้นักศึกษาใช้

ดังนั้นทางกลุ่มผู้จัดทำจึงมีความคิดที่จะทำเครื่องสแกนภาพในราคาถูกมีคุณภาพพอใช้ได้ ซึ่งแน่นอนเทียบกับเครื่องสแกนภาพในท้องตลาดไม่ได้ แต่ราคาถูกกว่ามาก ซึ่งมีวัตถุประสงค์เพียงเพื่อช่วยในการศึกษาเท่านั้นยังไม่อาจนำมาใช้ในงานจริงได้อย่างเต็มที่ กลุ่มผู้จัดทำมีความเห็นว่าจะสามารถปรับปรุงให้ดีที่สุด จนสามารถอ่านอักษรพิมพ์ขนาดกลางได้ เราอาจดัดแปลงเครื่องสแกนนี้ในงานอื่น ๆ ก็ได้ที่ไม่ต้องการความละเอียดของภาพมาก เช่น การให้จดจำลายเส้นทางเดินของเครื่องตัดโลหะ เพื่อตัดโลหะให้เป็นรูปต่างๆ (รายละเอียดดูใน รายงานการประชุมวิศวกรรมไฟฟ้า ครั้งที่ 10) หรือทำงานด้านอื่นซึ่งไม่เกี่ยวกับการสแกน เช่น นำหัวสแกนมาทำเป็นเครื่องอ่านบัตรเจาะรู

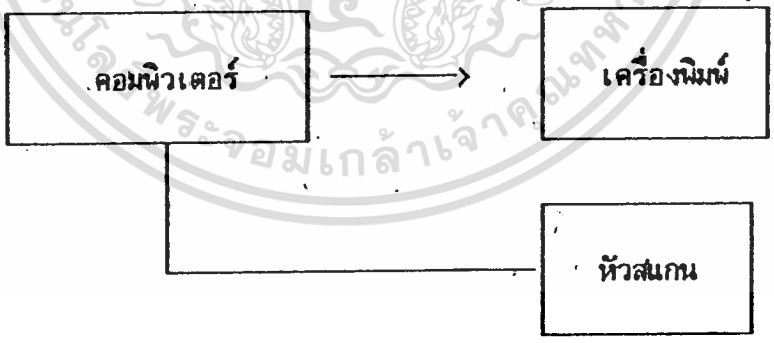
ภาพนั้นจะประกอบด้วยจุดต่าง ๆ มากมาย ดังเช่น จอมอนิเตอร์คอมพิวเตอร์ ถ้านำแว่นขยายส่องดูจะพบว่าประกอบขึ้นจากจุดเล็ก ๆ มากมาย ดังนั้นการสแกนภาพจะเป็นการกระทำในลักษณะตรงกันข้ามคือ แบ่งแยกภาพเป็นจุดเล็ก ๆ จุดเล็ก ๆ ที่ได้ คือ ข้อมูลของแต่

ละภาพ ถ้านำจุดเล็ก ๆ เหล่านั้นมาต่อกันก็จะได้ภาพกลับคืนมา ส่วนความละเอียดนั้นจะมากหรือน้อย ขึ้นกับว่าเราแบ่งแยกภาพเป็นจุดมากน้อยเพียงไร ยิ่งแบ่งจุดมากความละเอียดก็ยิ่งมาก หน่วยที่วัดความละเอียดในเครื่องสแกนทั่วไปคือ จุดต่อนิ้ว (DOT PER INCH, DPI) หมายถึงจำนวนจุดที่เครื่องสามารถแยกได้ใน 1 นิ้ว ซึ่งการแบ่งจุดตามแนวตั้งและแนวขวางนั้นมักจะเท่ากัน มิฉะนั้นแล้วภาพจะมีลักษณะผิดไปจากความจริง เช่น วงกลมเป็นวงรี หลักการของเครื่องสแกนที่ใช้ในปัจจุบันนั้น ส่วนใหญ่จะใช้ระบบ CCD โดยอาศัยแสงจากหลอดฟลูโอเรสเซนท์ส่องกระทบกับกระดาษที่ต้องการสแกน แสงที่กระทบกับกระดาษจะสะท้อนออกมาไม่เท่ากัน ตามความเข้มของกระดาษ (อันเนื่องมาจากสี และโทนของภาพ) แสงที่สะท้อนกลับมานี้ (เป็นแถบ) จะถูกเลนส์รวมแสงบีบให้เล็กลงจนเท่ากับหน้าต่างรับแสงของตัวไอซีจำพวก OPTICAL MEMORY ซึ่ง OPTICAL MEMORY นั้น คือหน่วยความจำที่ค่าของข้อมูลในหน่วยความจำนั้นจะขึ้นกับแสงที่ตกกระทบพื้นที่รับแสงของตัว IC โดยตัว IC จะมีกระจกสี่เหลี่ยม (คล้าย EPROM) เมื่อรับแสงช่องนี้จะทำหน้าที่เปลี่ยนจากแสงเป็นข้อมูลดิจิตอลโดยตรง และข้อมูลก็สามารถดึงมาใช้ได้ทันที (ซึ่งก็คือข้อมูลภาพนั้นเอง) จะเห็นว่าวิธีนี้รวดเร็วเนื่องจาก สแกนทีละบรรทัด เมื่อหมด 1 บรรทัด ก็จะให้สเตปปิ้งมอเตอร์เลื่อนขึ้น 1 สเต็ปเพื่อสแกนบรรทัดใหม่ ดังนั้น วิธีนี้จะเร็วมากกว่าวิธีที่จะนำมาใช้ในโครงงานนี้ รวมทั้งความละเอียดก็มีมากด้วย แต่การนำไอซีประเภท OPTICAL MEMORY มาใช้นั้นมีราคาแพง และต้องสั่งซื้อจากต่างประเทศโดยตรง รวมทั้งการทำเลนส์เอง การควบคุมสเต็ปปิ้งมอเตอร์การสร้างเครื่องกลไก และการเชื่อมต่อกับคอมพิวเตอร์ทำให้ค่าใช้จ่ายสูงมาก และคุณภาพย่อมสู้สินค้าในตลาดไม่ได้ เนื่องจากโรงงานผู้ผลิตมีทุกอย่างพร้อม ทั้งเทคโนโลยีและเงินทุน

2.1 แนวความคิดพื้นฐานและหลักการทำงาน

จากหลักที่ว่าภาพประกอบด้วยจุดเป็นจำนวนมาก ดังนั้นเราจึงต้องพยายามแยกภาพให้เป็นจุดเล็ก ๆ จำนวนมาก (DIGITIZE) พร้อมกับทั้งวัดระดับความเข้มของจุดได้ด้วย เพื่อที่สามารถให้เกิดการแยกแยะรูปภาพได้ โดยเฉพาะประเภทภาพถ่าย เราจึงต้องสร้างอุปกรณ์ชิ้นหนึ่งขึ้นมาเพื่อทำหน้าที่แยกจุดเหล่านี้ โดยการตรวจสอบระดับความเข้มของสัญญาณที่เข้ามาทีละจุด และเรียกว่าอุปกรณ์นี้ว่าหัวสแกน นอกจากนี้ยังต้องมีกลไกเพื่อควบคุมการเคลื่อนที่ของหัวสแกนและควบคุมภาพที่ต้องการสแกนให้เลื่อนไปพร้อม ๆ กันด้วย เพื่อให้หัวสแกนสามารถสแกนภาพได้อย่างถูกต้อง ดังนั้น เพื่อลดความยุ่งยากในการสร้างอุปกรณ์ควบคุมการสแกนนี้ เราจึงใช้เครื่องพิมพ์ชนิดดอทเมตริกซ์ (DOT MATRIX PRINTER) เป็นส่วนกลไกแทนเนื่องจากเครื่องพิมพ์นั้นมีกลไกในการเลื่อนกระดาษอยู่ในตัว พร้อมทั้งการเลื่อนหัวพิมพ์ได้ละเอียดมากพอสมควร ทั้งในแนวตั้งและแนวนอน คือ แนวตั้งสูงสุด 216 จุด/นิ้ว แนวนอน 240 จุด/นิ้ว (ขึ้นกับชนิดของ PRINTER) ถ้าสามารถติดหัวสแกนกับหัวเครื่องพิมพ์ได้ ก็สามารถทำการสแกนภาพได้เช่นกัน เพียงแต่ว่าจะช้ากว่าเครื่องสแกนระบบอื่นแต่ราคาจะถูกมาก เนื่องจากเสียค่าใช้จ่ายเฉพาะหัวสแกนเท่านั้น

หลักการเบื้องต้นของเครื่องสแกนที่ออกแบบไว้แสดงได้ดังรูป 2.1



รูป 2.1

เริ่มให้คอมพิวเตอร์จะส่งสัญญาณเพื่อให้หัวเครื่องพิมพ์เลื่อนไป พร้อมกับรับข้อมูลจากหัวสแกนมาด้วย ในตอนเริ่มแรกนั้นได้ทดลองให้หัวเครื่องพิมพ์เลื่อนไป 1 จุด พร้อมกับรับข้อมูลจากหัวสแกนมา 1 จุด หลังจากนั้น จึงสั่งให้หัวเครื่องพิมพ์เลื่อนไปอีก 1 จุด และรับข้อมูลมา

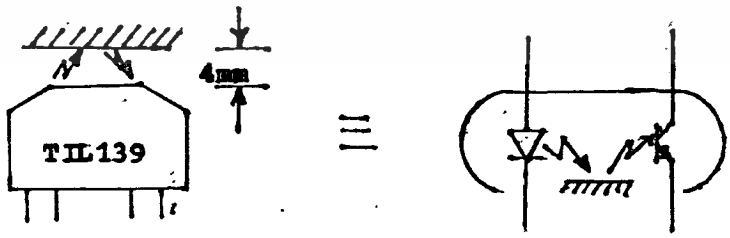
อีก เช่นนี้เรื่อยไป จากผลการทดลองพบว่า การสแกนจะช้ามาก เนื่องจากหัวเครื่องพิมพ์จะช้ากว่าการสั่งให้พิมพ์ที่ละบรรทัด จึงเปลี่ยนวิธีใหม่จากการสั่งให้หัวเครื่องพิมพ์เคลื่อนทีละจุดเป็นเคลื่อนทีละบรรทัด และในขณะที่หัวเครื่องพิมพ์เลื่อนนั้น ก็ให้รับข้อมูลจากหัวสแกนเข้ามาเลย โดยคำนวณเวลาที่หัวเครื่องพิมพ์เลื่อน 1 บรรทัดหารด้วย เวลาที่ใช้ในการรับข้อมูล 1 จุด (หาได้จากการคำนวณเวลาทำงานทางด้านซอฟท์แวร์ ซึ่งน้อยมาก) ก็จะได้จำนวนจุดที่สามารถรับเข้ามาได้ ซึ่งพบว่าจำนวนจุดมากกว่าความละเอียดของหัวสแกนเสียอีก จึงต้องใส่เวลาหน่วงในการรับข้อมูลเพื่อที่ว่า จะได้จำนวนจุดพอเหมาะกับความละเอียดของหัวสแกน ในการรับข้อมูลเข้าสู่คอมพิวเตอร์นั้น ทำได้ 2 วิธี คือ

1) การสร้างการ์ดอินเตอร์เฟสโดยตรง

2) การส่งข้อมูลผ่านพอร์ทที่มากับเครื่อง เช่น พอร์ทขนาน, อนุกรม

ในโครงงานนี้ได้ใช้วิธีที่ 2 โดยส่งข้อมูลผ่านพอร์ทเกม เหตุที่ใช้พอร์ทเกมเนื่องจากว่า ง่าย สะดวกและประหยัด ถ้าใช้พอร์ทอนุกรมจะต้องทำการเปลี่ยนข้อมูลจากขนานมาเป็นอนุกรมอีกที ถ้าใช้พอร์ทขนานจะทำให้สิ้นเปลือง เพราะใช้ข้อมูลเพียง 2 บิทเท่านั้น การใช้พอร์ทเกมนั้นจะประหยัดและมีการใช้พอร์ทน้อย SOCKET เสียบก็ถูกกว่าและทำให้สามารถใช้พอร์ทอื่นสำหรับงานที่มีความจำเป็นจริง ๆ โดยไม่ต้องซื้อการ์ดใหม่ เช่น ในเครื่องที่มีจอสีนั้น การ์ดแสดงผลจะไม่มีพอร์ทแบบขนานดังนั้น ถ้าต้องการใช้หัวสแกนนี้ต่อเข้าพอร์ทขนาน จำเป็นต้องซื้อพอร์ทเพิ่มอีกพอร์ทหนึ่งต่างหากทำให้สิ้นเปลือง (อีกพอร์ทหนึ่งใช้ต่อกับเครื่องพิมพ์อยู่แล้ว) รายละเอียดของพอร์ทเกมจะกล่าวในภายหลัง

สำหรับหัวสแกนได้ใช้ออปโต้คัปเปลอร์ (OPTOCOUPLER) แบบอินฟาเรด ซึ่งใช้ในงานตรวจจับวัตถุ แต่นำมาดัดแปลงใช้เป็นหัวสแกน มีลักษณะดังรูป 2.2



รูป 2.2

ภายในออพโต้คัปเปลอร์ประกอบด้วยหัวส่งอินฟาเรด และตัวรับโฟโต้ทรานซิสเตอร์อยู่ในตัวเดียวกัน ตัวส่งอินฟาเรดจะส่งแสงมากระทบกับวัตถุและจะสะท้อนกลับไปยังตัวรับ จำนวนแสงที่สะท้อนจะมากหรือน้อยจะขึ้นกับปัจจัยต่าง ๆ ดังนี้

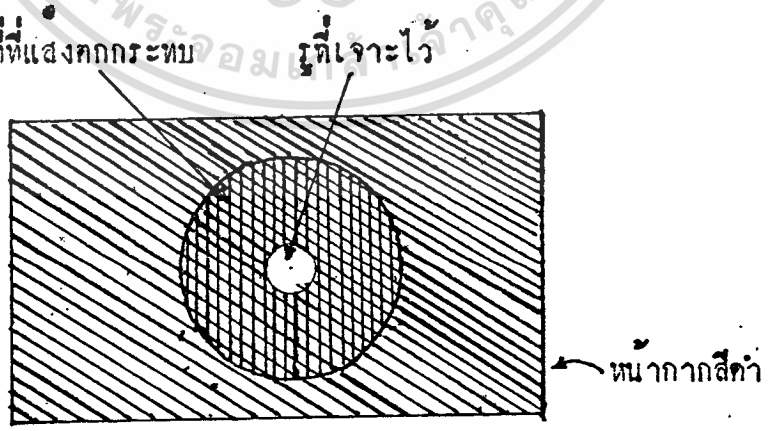
1. ลักษณะพื้นผิวของวัตถุ ถ้าวัตถุมีผิวขรุขระจะมีการสะท้อนน้อย ถ้ามีผิวเรียบหรือมันจะสะท้อนมาก

2. ระยะห่างระหว่างผิวสะท้อนกับตัวออพโต้คัปเปลอร์ ออพโต้คัปเปลอร์จะรับแสงสะท้อนได้มากที่สุดเมื่อระยะห่างของผิวสะท้อนอยู่ห่างเท่ากับระยะโฟกัส ดังในรูป 2.1 เราเลือกใช้ออพโต้คัปเปลอร์เบอร์ TIL 139 ซึ่งมีระยะโฟกัสเท่ากับ 3.8 มิลลิเมตร และเพื่อให้การตรวจวัดแสงสะท้อนได้ดีที่สุด เราจึงยึดตัวออพโต้คัปเปลอร์ให้ห่างจากภาพสแกน 3.8 มิลลิเมตรพอดี

3. สีของวัตถุ พื้นผิวสีต่าง ๆ จะสะท้อนแสงอินฟราเรดได้ไม่เท่ากัน เช่น สีฟ้าและขาวสะท้อนมาก สีดำสะท้อนน้อย เป็นต้น

อาศัยหลักการดังกล่าว เราจึงสามารถตรวจสอบโทนสีต่าง ๆ ของภาพที่จะนำมาสแกนได้โดยตรวจวัดความเข้มของแสงที่สะท้อนกลับมา ในโครงงานนี้จะแบ่งระดับความเข้มช่องแสงเป็น 4 ระดับ ดังนั้น ภาพ 1 จุด จะแทนด้วยข้อมูล 2 บิท

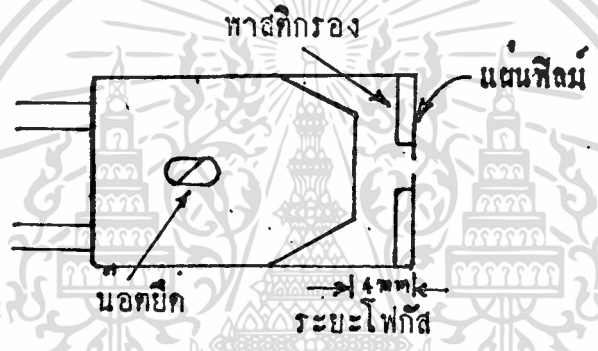
ปัญหาที่เกิดขึ้นคือ พื้นที่การสะท้อนแสงจากระยะห่างกับระยะโฟกัสยังใหญ่เกินไป ดังนั้นจึงต้องทำหน้ากากขึ้นมาบังส่วนหน้าของออพโต้คัปเปลอร์ไว้ ดังในรูป 2.3



รูป 2.3

หน้ากากนี้ทำจากแผ่นฟิล์มบางฟ่นสีดำด้าน และเจาะรูขนาดเส้นผ่าศูนย์กลาง 0.5 มิลลิเมตร การติดแผ่นฟิล์มนี้จะต้องอยู่ห่างจากหัวออพโต้คัปเปลอร์เท่าระยะโฟกัสพอดี และต้องติดอยู่กึ่งกลางระหว่างหัวรับและหัวส่งด้วย

ผลจากการติดหน้ากากนี้ จะทำให้การวัดความเข้มของแสงที่สะท้อนขึ้นอยู่กับสีที่อยู่บริเวณพื้นที่ของรูที่เจาะไว้เท่านั้น วิธีนี้มีผลให้ สามารถแยกแยะความแตกต่างดีขึ้น โดยเฉพาะเส้นตรงที่อยู่ใกล้ชิดกัน แต่ถ้าเจาะรูเล็กเกินไปจะทำให้สัญญาณที่วัดได้มีความแต่ต่างกันน้อยจนไม่สามารถวัดได้ แผ่นฟิลม์นี้จะติดอยู่ที่จุดโฟกัสพอดี ดังรูป 2.4



รูป 2.4

หัวสแกนนี้ต้องปรับให้ชิดติดกับกระดาษพอดีเพราะถ้าห่างจากกระดาษ จะทำให้รูที่เจาะเป็นเสมือนจุดกำเนิดแสง แสงกระจายออกตามหลักของทฤษฎีแสง ทำให้สัญญาณสะท้อนมีกำลังอ่อนลง

จากสัญญาณที่ได้จะนำไปเข้าวงจรขยายสัญญาณและผ่านวงจรคอมพาราเตอร์ เพื่อดูว่าความเข้มอยู่ระดับไหน วงจรสุดท้ายจะเป็นวงจรเข้ารหัส (ENCODER) เพื่อเปลี่ยนระดับสัญญาณเป็นรหัสข้อมูลของความเข้มในแต่ละจุด ดังไดอะแกรมข้างล่างนี้

ข้อมูลที่ได้จะนำเข้าคอมพิวเตอร์ผ่านเกมพอร์ท และนำไปแสดงที่วีดีโอแรมต่อไป จะเห็นว่าการสแกนลักษณะนี้จะช้ามากเมื่อเทียบกับแบบอื่นเพราะต้องเลื่อนหัวเครื่องพิมพ์ทีละบรรทัด เนื่องจากหัวสแกน สแกนได้ทีละจุด การสแกน 1 หน้ากระดาษจึงใช้เวลามาก ดังนั้นจึงได้มีการปรับปรุงให้สแกนได้เร็วขึ้น โดยให้สแกนเฉพาะพื้นที่ส่วนมีรูป (ตั้งระยะสแกนได้) เพราะว่าโดยส่วนใหญ่แล้วรูปจะไม่เต็มหน้ากระดาษ การตั้งระยะสแกนจะทำให้ลดเวลาที่ใช้ลงได้

2.2 <u>การทำงานของเครื่องพิมพ์</u>

เครื่องพิมพ์ที่ใช้ในโครงงานนี้ คือ เครื่องพิมพ์ยี่ห้อ EPSON รุ่น LX-800 โดยการเอาหัวปริ้นเตอร์ออก และ สามารถติดตั้งหัวสแกนแทนหัวปริ้นเตอร์ได้เลย ถ้าใช้รุ่นอื่น ยี่ห้ออื่นก็จะต้องดัดแปลง ตามแต่ชนิดของเครื่องพิมพ์ ซึ่งถ้าเข้าใจหลักการทำงานของสแกนเนอร์แล้วก็จะง่ายต่อการดัดแปลง เนื่องจากคำสั่งที่ใช้ล้วนเป็นคำสั่งมาตรฐาน แต่เครื่องพิมพ์นั้นควรจะเป็นแบบที่พิมพ์กราฟฟิคได้ เพราะจะมีความละเอียดดีกว่า ดังนั้นต่อไปนี้จะอ้างอิงถึงระบบของ LX-800 เท่านั้น

ในเครื่องพิมพ์ LX-800 นั้นมีโหมดการทำงาน 2 โหมดใหญ่ ๆ คือ โหมดกราฟฟิค และโหมดตัวอักษร ในโหมดกราฟฟิคยังแบ่งเป็น 7 โหมด ซึ่งขึ้นกับความละเอียดของกราฟฟิคที่ต้องการ ในโครงงานนี้เราใช้โหมดการทำงานแบบโหมดกราฟฟิค เพราะให้ความละเอียดและควบคุมหัวเครื่องพิมพ์ได้ละเอียดกว่าแบบโหมดตัวอักษร จุดประสงค์ของการใช้เครื่องพิมพ์นั้น เพื่อใช้หัวเครื่องพิมพ์นำหัวสแกนเคลื่อนผ่านกระดาษที่ต้องการอ่าน ดังนั้น เราจะพูดกันเฉพาะส่วนในการควบคุมหัวเครื่องพิมพ์ให้ได้ตามที่ต้องการเท่านั้น โดยจะแบ่งเป็นการควบคุมหัวตามแนวขวางและการให้คำสั่ง LINE FEED (LF) แก่กระดาษเมื่อสแกนเสร็จ 1 บรรทัด

การควบคุมหัวเครื่องพิมพ์ตามแนวขวางนั้น เราสามารถกำหนดระยะของการเลื่อนได้โดยการสั่งให้พิมพ์จุดมากน้อยตามที่ต้องการ เช่นในโหมดกราฟฟิคแบบ SINGLE DENSITY (ความหนาแน่น 60 จุด/นิ้ว) ถ้าต้องการให้หัวเลื่อนไป 5 นิ้ว ก็สั่งให้พิมพ์ = 60 x 5 =300 จุด แต่จุดที่จะให้พิมพ์นี้ เป็น BLANK ดังนั้น หัวเครื่องพิมพ์ก็จะเคลื่อนที่ไปเป็นระยะ 5 นิ้วตามต้องการ (รายละเอียดของคำสั่งที่ใช้ในโหมดกราฟิคนั้น หาอ่านได้จากคู่มือเครื่องพิมพ์) ส่วนการที่จะบังคับให้หัวเครื่องพิมพ์เริ่มเคลื่อนที่จากตำแหน่งใดนั้น ก็อาจทำได้โดยใช้คำสั่งตั้ง LEFT MARGIN ความจริงแล้วคำสั่งนี้เป็นคำสั่งในโหมดตัวอักษร แต่สามารถนำมาประยุกต์ใช้กับโหมดกราฟิคได้ เพื่อช่วยให้การควบคุมหัวยืดหยุ่นยิ่งขึ้น การตั้ง LEFT MARGIN นั้น เราตั้ง

เป็นจำนวนตัวอักษร โดยปรกติเครื่องพิมพ์ในโหมดตัวอักษรนั้นจะพิมพ์ได้ 80 ตัวอักษร 8 ตัว/นิ้ว หรือ 10 ตัว/นิ้ว ดังนั้นถ้าต้องการตั้งให้หัวเครื่องพิมพ์เริ่มเลื่อน ณ จุดที่ห่างจากจุดเริ่มต้น 2 นิ้วก็ตั้ง TAB = 20

การควบคุมทางแนวตั้งทำได้ง่ายมาก โดยการตั้ง LINE FEED ตามต้องการ เมื่อเสร็จการสแกน 1 บรรทัด ก็ส่งสัญญาณ LF ให้เครื่องพิมพ์เลื่อนกระดาษขึ้น 1 บรรทัดเพื่อทำการสแกนบรรทัดใหม่ต่อไป

สิ่งที่สำคัญอีกประการหนึ่งในการควบคุมหัวเครื่องพิมพ์คือ การทราบถึงตำแหน่งของหัวเครื่องพิมพ์ เพื่อว่าจะได้สามารถซิงค์กับสัญญาณจากหัวสแกนที่เข้าคอมพิวเตอร์ได้ และจะได้สามารถหยุดการรับข้อมูลได้ ในขณะที่หัวเครื่องพิมพ์เลื่อนกลับ เราเลือกการสแกนในทิศทางเดียวเนื่องจากว่าจะทำให้ภาพมีความเที่ยงตรงแม่นยำกว่าการสแกน 2 ทิศทาง ทั้งนี้เพราะเวลาหัวเครื่องพิมพ์เคลื่อนไปข้างหน้าและกลับไม่เท่ากัน จริง ๆ แล้วการรู้ตำแหน่งที่แน่นอนของหัวเครื่องพิมพ์ อาจทำได้โดยสั่งให้พิมพ์ทีละจุดแต่วิธีนี้ จะช้ามาก (สแกนเพียง 1 เส้นก็หมดไปหลายนาที) ดังนั้นจึงต้องใช้วิธีอ้อม ก็คือ หาเวลาเฉลี่ยในการเคลื่อนหัวพิมพ์โดยการจับเวลาเพื่อหาว่าหัวพิมพ์เลื่อนถึงตำแหน่งสุดท้ายหรือยัง และอีกตำแหน่งที่ต้องการทราบคือ ตำแหน่งเริ่มต้นการสแกน ตำแหน่งนี้จะเป็นการบอกให้คอมพิวเตอร์ทราบว่าได้ขึ้นรอบการสแกนใหม่แล้ว

การหาเวลาในการเลื่อนหัวเครื่องพิมพ์เป็นสิ่งจำเป็นมาก เพื่อนำไปคำนวณหาเวลาที่ใช้ในการหน่วงของการสแกนในแต่ละจุด เนื่องจากเวลาการอ่านข้อมูลเข้าคอมพิวเตอร์นั้นเร็วกว่าหัวอ่านมาก จึงต้องมีการหน่วงเวลาเพื่อให้ ข้อมูล กับหัวอ่านสัมพันธ์กัน การคำนวณเวลาหน่วงจึงจำเป็นต้องทราบถึง เวลาที่หัวอ่านเคลื่อนที่ และเวลาในการรับข้อมูล 1 จุด ซึ่งหาได้จากตาราง TIME CYCLE ของแต่ละคำสั่ง ถ้าหน่วงมากหรือน้อยไป จะทำให้มีผลต่อรูปที่ได้ อาจจะยาวหรือสั้นกว่าความจริง ควรให้พอดีกับขนาดความละเอียดของหัวอ่าน จะได้ผลที่ดีที่สุด เวลาหน่วงนั้นเรายังต้องพิจารณารุ่นและความถี่ที่ใช้อีกด้วย ซึ่งในทางปฏิบัติได้ทดลองเปลี่ยนค่าเวลาหน่วงเพื่อให้ได้เวลาที่เหมาะสม

การทำงานของทั้งระบบ เริ่มต้นที่คอมพิวเตอร์จะส่งสัญญาณให้หัวเครื่องพิมพ์เลื่อน ขณะที่หัวเครื่องพิมพ์เคลื่อนนั้น คอมพิวเตอร์จะรับข้อมูลจากหัวสแกนทันที เมื่อรับข้อมูลจนครบจำนวนที่ตั้งไว้แล้ว คอมพิวเตอร์จะส่งสัญญาณ LF ให้แก่เครื่องพิมพ์เพื่อขึ้นบรรทัดใหม่ พร้อมทั้งส่งสัญญาณควบคุมหัวเครื่องพิมพ์อีกชุดหนึ่ง เมื่อหัวเครื่องพิมพ์เลื่อนมาถึงจุดเริ่มต้น (เช็คได้จากสัญญาณ BUSY) ก็จะเริ่มการสแกนในรอบใหม่ จนครบจำนวนเส้น (บรรทัด) ที่ตั้งไว้ หลังจาก

นั้นก็จะกลับสู่โปรแกรมหลักเพื่อทำการ SAVE หรือ LOAD ข้อมูลต่อไป

2.3 การ์ดแสดงผล

ในปัจจุบันมีการ์ดแสดงผลหลายแบบที่ใช้กับเครื่อง IBM เช่น EGA,CGA, HERCULES และอื่น ๆ ในโครงงานนี้เราได้พัฒนาโปรแกรมสำหรับจอ CGA เท่านั้น เหตุที่เลือกจอ CGA เพราะว่า

1) หัวสแกนนั้นยังละเอียดไม่พอ การใช้จอ EGA

2) การเขียนโปรแกรมใช้กับจอ CGA นั้น สามารถนำไปใช้กับจออื่นๆได้ เช่น EGA และHERCULES (ผ่าน PROGRAM BOOTGAMES)

จอ CGA นั้นเราแบ่งได้เป็น 2 SCREEN คือ SCREEN 1 นั้นมี 4 สี ความละเอียด 320 x200 ในโหมดนี้จะได้การทำงานดีที่สุด เนื่องจากแสดงได้ 4 สีทำให้ภาพที่ได้จะเห็นความแตกต่างชัดเจน ส่วน SCREEN 2 นั้น จะมี 2 สีเท่านั้นคือ ขาวและดำ ดังนั้น การแสดงความแตกต่างสองแต่ละความเข้มจะทำไม่ได้ นอกจากใช้ความหนาแน่นของจุดในการแสดงความเข้ม ซึ่งสำหรับหัวอ่านรุ่นแรกนี้ เพื่อความง่ายเราได้กระจายจาก 1 จุดเป็น 2 จุด นั่นคือมีความละเอียด 320x200 เหมือน SCREEN 1 แต่ละจุดประกอบด้วย 2 จุด คือความเข้มขาวเป็น 11 เทา 1 เป็น 10 เทา 2 เป็น 01 และดำเป็น 00 จะเห็นว่าจะออกจอภาพเพียง 3 ระดับเท่านั้น (10 และ 01 ให้ความเข้มเหมือนกัน) ทำให้ภาพนั้นขาดความชัดเจนกว่า SCREEN 1 ทั้งทำให้ภาพกว้างกว่าเป็นจริง (แต่จริงๆแล้วกว้างพอ ๆ กับภาพใน SCREEN 1) ส่วนหลักการที่ใช้สำหรับหัวอ่านรุ่นที่สองจะต่างออกไป โดยเราจะสร้างแพทเทอร์นสำหรับระดับความเข้มต่างๆไว้ 8 โทน ซึ่งแต่ละโทนจะประกอบขึ้นจากการเรียงจุดขาวและดำรวม 8 จุดที่มีรูปแบบการเรียงต่างกัน เมื่อสามารถวัดระดับความเข้มของจุดใด ๆ บนภาพแล้ว ก็จะเลื่อนบิทตามแพทเทอร์นของระดับความเข้มสำหรับจุดจุดนั้นออกไปยังวีดีโอแรม เพื่อแสดงภาพบนจอมอนิเตอร์ ทำเช่นนี้ไปเรื่อย ๆ จนกระทั่งจุดใหม่มีระดับความเข้มที่ต่างออกไป จึงเปลี่ยนแพทเทอร์นตามความเข้มของจุดใหม่นั้น

การทำงานของการ์ดแสดงผลนี้ วงจรควบคุมจะนำค่าซึ่งเป็นข้อมูลของจุดและสีที่เก็บไว้ในหน่วยความจำที่เรียกว่าวีดีโอแรมออกจอ ถ้าเราทำการเปลี่ยนแปลงวีดีโอแรมจะมีผลให้ภาพบนจอเปลี่ยนแปลงไปด้วย ดังนั้นถ้าข้อมูลจากหัวสแกนถูกถ่ายลงวีดีโอแรมทันที ภาพบนจอจะเปลี่ยนแปลงพร้อม ๆ กับการเลื่อนของหัวสแกน หน่วยความจำวีดีโอนี้จะมีขนาดเท่ากับ 16 กิโลไบท์ หน่วยความจำส่วนนี้เราจะใช้ในการ SAVE และ LOAD ข้อมูลลงแผ่นดิสค์

มีข้อพึงระวังอยู่อย่างหนึ่งคือ หน่วยความจำของเส้นคู่และเส้นคี่จะอยู่ที่ตำแหน่งต่างกัน ภายในเซ็กเมนต์ (SEGMENT) เดียวกันคือ วีดีโอเซ็กเมนต์อยู่ที่ B800H ออฟเซตของเส้นคู่อยู่ที่ 0000H ออฟเซตของเส้นคี่อยู่ที่ 2000H ดังนั้นการถ่ายข้อมูลลงวีดีโอแรมจึงต้องมีการสลับกันระหว่างตำแหน่งเส้นคี่และเส้นคู่

2.4 พอร์ทเกม

หน้าที่หลักของการ์ด game control, คืออุปกรณ์ที่ใช้ในการต่อจอยสติ๊ก (joystick) เข้ากับเครื่อง IBM PC แต่ก็สามารถใช้ในการอินเทอร์เฟสอุปกรณ์อื่น ๆ เข้ากับเครื่อง PC ได้ ลักษณะสำคัญข้อหนึ่งของการ์ดนี้คือ สามารถรับข้อมูลได้อย่างเดียว โดยมีอินพุทที่รับสัญญาณดิจิตอลอยู่ 4 อินพุทและอินพุท 4 อินพุทที่รับสัญญาณผ่านทางความต้านทานเพื่อใช้เป็นค่าหน่วงเวลาให้กับวงจรโมโนสเตเบิ้ลซึ่งจะอธิบายต่อไป อินพุททั้งหมดนี้ใช้ในการตรวจจับสัญญาณจากปุ่มกดและจากตำแหน่งของความต้านทานปรับค่าได้ (potentiometer) ในจอยสติ๊ก แอดเดรสของพอร์ทนี้มีค่าเท่ากับ 0201 (ฐานสิบหก) การใช้คำสั่ง OUT เพื่อ OUT ข้อมูลใด ๆ มาที่พอร์ทนี้จะเป็นการสั่งให้วงจรโมโนสเตเบิ้ลทั้ง 4 ทำงาน เอาท์พุทที่ได้จากวงจรโมโนสเตเบิ้ลทั้ง 4 สามารถอ่านกลับไปใช้งานได้โดยใช้คำสั่ง IN กับแอดเดรส 0201 (ฐานสิบหก) สำหรับช่วงเวลาของเอาท์พุทที่ได้จากวงจรโมโนสเตเบิ้ลจะถูกกำหนดโดยความต้านทานและตัวเก็บประจุ ซึ่งถูกต่อเข้ากับวงจรโมโนสเตเบิ้ลแต่ละชุด สำหรับค่าความต้านทานที่ใช้เป็นค่า time-constant นี้ได้จากอุปกรณ์ภายนอกที่เรานำมาอินเทอร์เฟส เช่น มาจาก potentiometer ของจอยสติ๊ก ดังนั้นเมื่อเราวัดช่วงเวลาของสัญญาณเอาท์พุทที่ได้จากวงจรโมโนสเตเบิ้ล และทราบค่าตัวเก็บประจุที่ใช้ในวงจร เราจะสามารถทราบค่าของความต้านทานที่ใช้ได้ ค่าความต้านทานตัวนี้ต้องถูกต่อขึ้นไปบวก 5 โวลท์ เพื่อให้วงจรทำงานอย่างถูกต้อง จากหลักการที่กล่าวมานี้ ถ้าเราสร้างเงื่อนไขใดเงื่อนไขหนึ่งขึ้นมาซึ่งสามารถแทนตัวความต้านทานได้ เราจะสามารถตรวจจับและรับข้อมูลผ่านทางวงจรได้

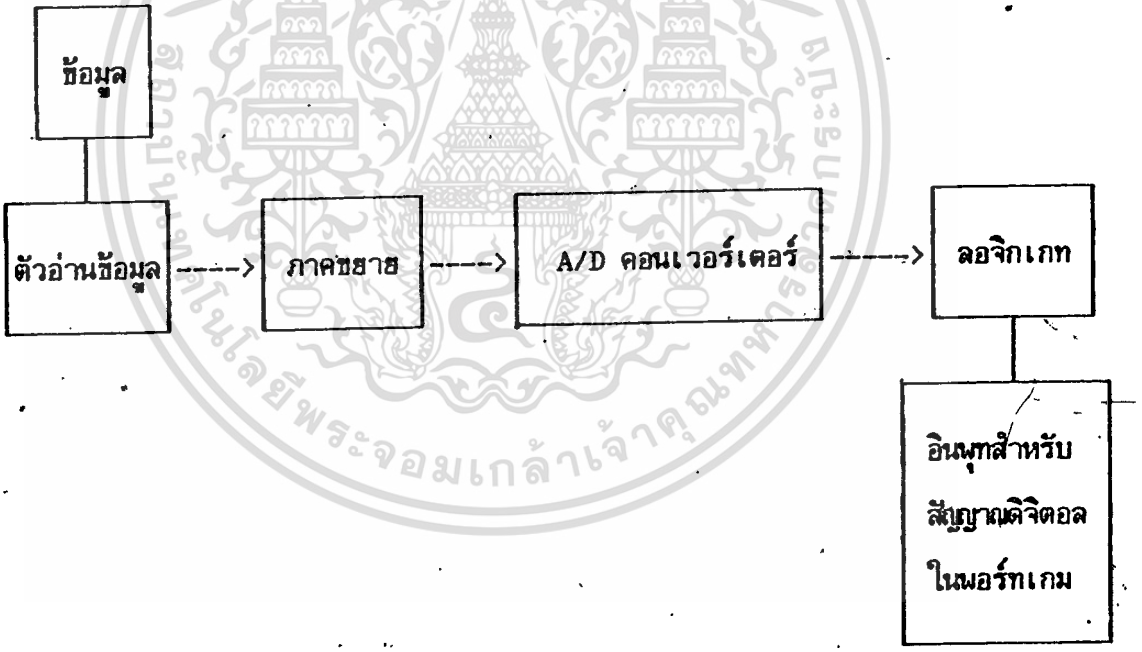
ส่วนของพอร์ทเกมที่ใช้ในวงจรนี้ จะเกี่ยวข้องกับการรับข้อมูลทางดิจิตอล โดยจะต่อกับไอซีเบอร์ 74LS244 ซึ่งเป็น TRISTATE BUFFER/LINE DRIVER/LINE RECEIVERS ทำหน้าที่รับข้อมูลจากภายนอกที่เป็นระดับดิจิตอล เข้าสู่คอมพิวเตอร์ ข้อมูลจากหัวสแกนจะถูกนำเข้าสู่คอมพิวเตอร์โดยผ่านส่วนนี้ โดยเราใช้เพียง 2 บิทเท่านั้น

3.1 วงจรส่วนอิเล็คทรอนิคส์

เนื่องจากสิ่งประดิษฐ์ชิ้นนี้ได้นำเครื่องพิมพ์มาประยุกต์ใช้งาน จึงจำเป็นต้องสร้างชิ้นส่วนทางด้านฮาร์ดแวร์ (HARDWARE) เพิ่มเติมเพื่อนำไปใช้ร่วมกับเครื่องพิมพ์ ฮาร์ดแวร์สำหรับชิ้นงานรุ่นแรกนี้ประกอบด้วย ส่วนหัวอ่าน (SCANNER HEAD) ส่วนของวงจรขยาย (AMPLIFIER CIRCUIT) วงจร A/D คอนเวอร์เตอร์ (A/D CONVERTER) และลอจิกเกท (LOGIC GATE) สำหรับส่งข้อมูลให้กับคอมพิวเตอร์

3.1.1 บล็อคไดอะแกรมของวงจร (CIRCUIT'S BLOCK DIAGRAM)

บล็อคไดอะแกรมในรูป 3.1 แสดงลักษณะการทำงานโดยทั่วไปของวงจรอิเล็คทรอนิคส์ของชิ้นงานนี้ส่วนรายละเอียดของอุปกรณ์แต่ละชิ้นจะกล่าวถึงในภายหลัง ซึ่งสามารถแบ่งออกได้เป็น 4 ขั้นตอนดังนี้ะ



รูป 3.1

ก.ตัวอ่านข้อมูล

ตัวอ่านข้อมูลในส่วนนี้ประกอบด้วยหัวอ่านที่ใช้การเชื่อมโยงทางแสง (OPTOCOUPLING) เป็นการส่งผ่านสัญญาณ ตัวอ่านนี้จะสามารถเปล่งแสงออกไปกระทบกับภาพที่เราต้องการ จากนั้นตัวรับสัญญาณแสงก็จะเก็บสัญญาณไว้ ซึ่งสัญญาณที่รับได้เหล่านี้จะมีค่าต่างกันขึ้นอยู่กับความเข้ม

ของแสงสะท้อนโดยที่จุดสีขาวจะสะท้อนแสงได้ดีกว่าจุดที่มีสีเข้มกว่า จากนั้นตัวรับแสงก็จะเปลี่ยนสัญญาณแสงที่รับได้ให้กลายเป็นสัญญาณไฟฟ้า ซึ่งขนาดของกระแสไฟฟ้านั้นจะแปรตามความเข้มของแสงที่ตัวรับตรวจจับได้

ข.ภาคขยายสัญญาณ

ภาคขยายสัญญาณนี้จะรับสัญญาณไฟฟ้าจากภาคอ่านข้อมูลแล้วจึงนำมาขยาย เหตุผลที่ต้องมีการขยายสัญญาณก็เนื่องจากว่าสัญญาณที่ได้จากภาคแรกนั้นจะมีค่าต่ำมาก ซึ่งจะทำให้ยากต่อการเปรียบเทียบความแตกต่างระหว่างสัญญาณในแต่ละระดับของความเข้ม อันจะเป็นผลทำให้ภาคต่อไปยุ่งยากซับซ้อนยิ่งขึ้น ดังนั้น จึงจำเป็นจะต้องขยายให้สัญญาณที่รับได้มีขนาดสูงพอที่จะตรวจวัดความแตกต่างได้โดยไม่ยากนัก

ส่วนของภาคขยายสัญญาณนี้ประกอบด้วยออปแอมป์เพียงหนึ่งตัวเท่านั้น โดยต่อในลักษณะของวงจรขยายไม่กลับเฟส (NONINVERTING AMPLIFIER) และจัดให้อัตราขยายแรงดันมีค่าประมาณ 21 เท่า แต่อัตราขยายนี้จะต้องเปลี่ยนแปลงไปในกรณีที่กระดาษที่ใช้มีความมันเงาต่างจากที่ได้ทดลอง

ค.ภาคเปลี่ยนสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (A/D CONVERTER)

หลังจากที่สัญญาณไฟฟ้าถูกขยายแล้วก็จะถูกนำมาตรวจระดับของแรงดัน เนื่องจากสัญญาณจากภาคที่สองนั้นจะอยู่ในรูปของสัญญาณอนาลอก (ANALOG) ดังนั้น จึงจำเป็นต้องเปลี่ยนให้เป็นสัญญาณแบบดิจิตอล (DIGITAL) เพื่อให้สัญญาณที่ออกสามารถป้อนเข้าสู่คอมพิวเตอร์ได้

อุปกรณ์ที่ในการแปลงสัญญาณจากอนาลอกเป็นดิจิตอลนั้น เราใช้วงจรเปรียบเทียบสัญญาณแบบธรรมดาโดยได้ตั้งช่วงของความเข้มของแสงที่รับได้ไว้ 4 ระดับด้วยกัน สัญญาณที่ออกจากภาคนี้จะอยู่ในรูปของสัญญาณดิจิตอล 3 บิทด้วยกัน โดยมีค่า 000 001 011 111 แทนสี ดำ เทาเข้ม เทาอ่อน และขาว ตามลำดับ

ในการประดิษฐ์โครงงานนี้วงจรคอมพาราเตอร์ที่ใช้ประกอบด้วยออปแอมป์ 3 ตัวเท่านั้น ซึ่งการใช้งานออปแอมป์ในลักษณะนี้จะทำให้สามารถเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอลได้รวดเร็วมาก ส่วนเหตุผลที่ไม่ใช้ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล (A/D CONVERTER) โดยตรงเนื่องจากอุปกรณ์ดังกล่าวมีราคาค่อนข้างสูง และมีความยุ่งยากมากกว่าเมื่อเทียบกับออปแอมป์แต่จะให้ผลลัพธ์ใกล้เคียงกัน

ง. ลอจิกเกท

สัญญาณที่ได้จากภาคเปรียบเทียบแรงดันสัญญาณนั้นจะอยู่ในรูปของสัญญาณดิจิตอลถึง 3 ตัว ซึ่งสัญญาณนี้จะใช้แทนความเข้มของสีเพียง 4 ระดับเท่านั้น ดังนั้นเพื่อลดความสิ้นเปลืองในการเก็บข้อมูลลงในหน่วยความจำของคอมพิวเตอร์ จึงควรประดิษฐ์ภาคของลอจิกเกทเพื่อลดจำนวนของตัวแสดงผลทางเอาท์พุทให้เหลือเพียง 2 ตัวเท่านั้น หลังจากผ่านวงจรของลอจิกเกทแล้วสัญญาณใหม่ที่ได้นี้จะอยู่ในลักษณะดังนี้คือ 00 แทนสีดำ 10 แทนเทาแก่ 01 แทนสีเทาอ่อน 11 แทนสีขาว ซึ่งจะเห็นว่าระดับความเข้มถูกลดลงจาก 4 เหลือ 3 (เนื่องจากระดับเทาแก่และเทาอ่อนไม่ต่างกันเมื่อแสดงผลออกจอภาพ) จึงทำให้ภาพที่ได้มีความคมชัดน้อยลง แต่ในขณะเดียวกันก็จะทำให้วงจรทำงานได้อย่างมีเสถียรภาพดีขึ้น หลังจากนั้นสัญญาณจะถูกนำออกจากเอาท์พุทของเกทไปป้อนให้กับพอร์ทเกมของเครื่องไมโครคอมพิวเตอร์เพื่อนำไปแสดงผลบนจอมอนิเตอร์

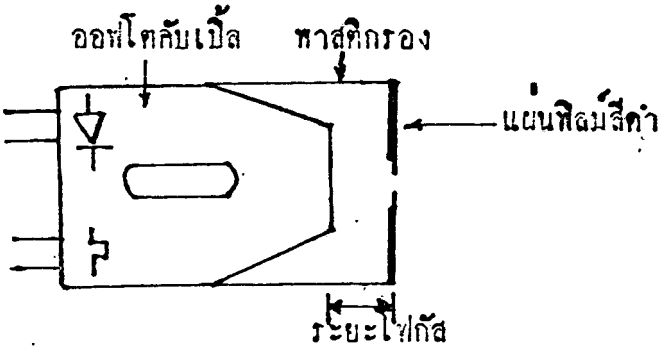
3.1.2 <u>รายละเอียดของวงจรในแต่ละภาค</u>

ในหัวข้อนี้จะแสดงส่วนของวงจรที่ประกอบขึ้นเป็นภาคต่าง ๆ ดังที่ได้กล่าวมาแล้ว โดยจะแสดงอุปกรณ์ที่ใช้ คุณสมบัติของอุปกรณ์เหล่านั้น และแสดงการคำนวณค่าต่าง ๆ ที่ได้จากการทดลอง

ก. วงจรของตัวอ่านข้อมูล

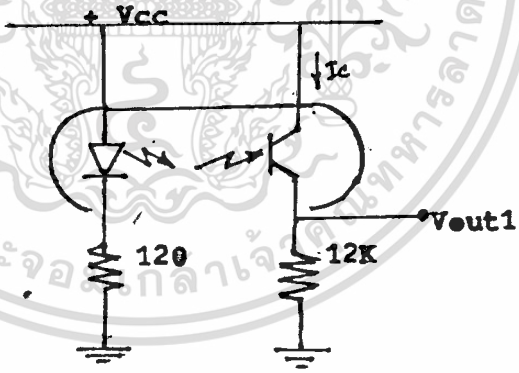
อุปกรณ์หลักที่ใช้ในการอ่านข้อมูลสำหรับชิ้นงานนี้เป็นออฟโตคับเปิ้ล (OPTOCOUPLER) ที่มีการบรรจุไว้ในภาชนะเรียบร้อยแล้ว และมีคุณสมบัติทางไฟฟ้าดังแสดงในภาคผนวก ง.

เนื่องจากออฟโตคับเปิ้ลเบอร์นี้เหมาะสำหรับการตรวจหาตำแหน่งของวัตถุชิ้นใหญ่ ๆ เท่านั้น ดังนั้น ในการนำมาใช้กับชิ้นงานนี้จึงจำเป็นจะต้องเพิ่มชิ้นส่วนบริเวณหัวสแกนเพื่อให้ได้สัญญาณที่ละเอียดยิ่งขึ้น จากการทดลองใช้ออฟโตคับเปิ้ลเมื่อไม่มีอุปกรณ์เสริม ความละเอียดที่วัดได้จะมีค่าประมาณ 2.5 ตารางมิลลิเมตรต่อพิกเซล (PIXEL) ซึ่งจะให้ภาพที่ค่อนข้างหยาบ แต่เมื่อเพิ่มแผ่นฟิล์มเพื่อลดความกว้างของจุดโฟกัสดังรูป 3.2 ความละเอียดจะมีค่าประมาณ 0.8 ตารางมิลลิเมตรต่อพิกเซล จึงให้ภาพที่คมชัดขึ้น



รูป 3.2

ส่วนวงจรในภาคนี้จะแสดงในรูป 3.3 แรงดันจากไฟเลี้ยงที่ออกจากพอร์ทจอยสติ๊กมีค่าประมาณ 4.5 โวลท์ ตัวต้านทานที่ต่ออนุกรมกับไดโอดและโฟโต้ทรานซิสเตอร์มีค่าเท่ากับ 120 โอห์ม และ 12 กิโลโอห์มตามลำดับ จากการทดลองกระแสที่ไหลผ่านขาอิมิตเตอร์ของโฟโต้ทรานซิสเตอร์ ในขณะที่มีแสงสะท้อนต่ำสุด (สีดำ) และสูงสุด(สีขาว) จะมีค่าประมาณ 9.36 และ 13 ไมโครแอมป์ ตามลำดับ ทำให้แรงดัน V_{out} มีค่าอยู่ในช่วง 0.11 ถึง 0.15 โวลท์

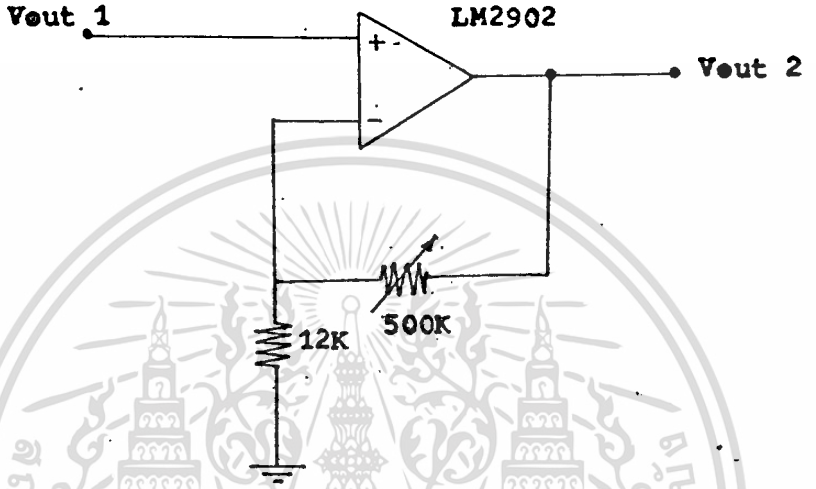


TIL139

รูป 3.3

ข. วงจรขยายสัญญาณ

หลังจากที่ได้สัญญาณออกจากโฟโต้ทรานซิสเตอร์แล้วก็จะนำแรงดันนี้มาขยาย โดยใช้ไอซีออปแอมป์เบอร์ LM2902 โดยต่อในลักษณะของวงจรขยายไม่กลับเฟสดังแสดงในรูป 3.4

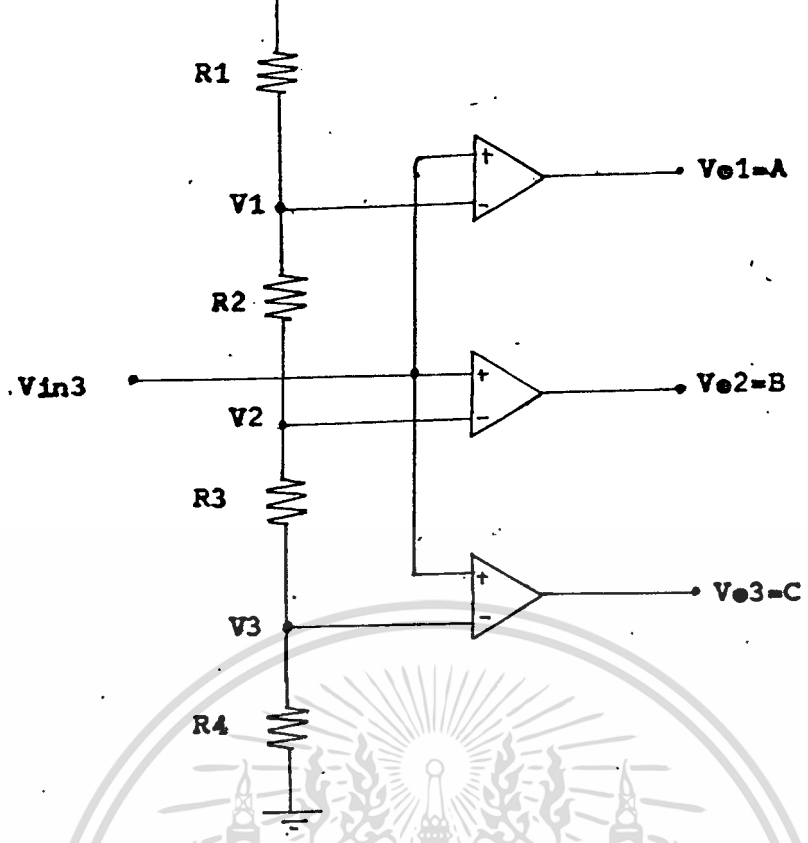


รูป 3.4

ตัวต้านทาน R_1 ที่ใช้มีขนาด 12 กิโลโอห์มเพื่อให้เกิดอิมพีแดนซ์แมทชิ่ง (IMPEDANCE MATCHING) กับภาคก่อนหน้านี้ และจัดให้ R_2 มีค่า 240 กิโลโอห์มเพื่อให้อัตราขยายมีขนาด 21 เท่า ดังนั้นแรงดันที่จุด V_{out2} จะมีค่าอยู่ในช่วง 2.36 ถึง 3.25 โวลท์ ซึ่งเป็นช่วงที่เหมาะสำหรับนำไปเปรียบเทียบแรงดันในภาคต่อไป

ค. วงจรเปรียบเทียบแรงดัน

หลังจากที่ผ่านวงจรขยายสัญญาณมาแล้ว แรงดัน V_{out2} จะถูกนำมาเปรียบเทียบกับแรงดันอ้างอิงที่ตั้งไว้เพื่อแบ่งช่วงของแรงดันออกเป็น 4 ระดับ โดยวงจรที่ใช้มีลักษณะดังรูป 3.5



รูป 3.5

การตั้งค่าของตัวต้านทาน R_1 ถึง R_4 จะได้จากการทดลองวัดระดับแรงดันของ V_{in3} ณ สีต่าง ๆ กันซึ่งผลปรากฏว่าแรงดันอ้างอิงที่จุด V_1 V_2 และ V_3 ควรมีค่าประมาณ 3.2, 2.8 และ 2.38 โวลท์ ตามลำดับ ซึ่งจะให้แรงดันที่เอาท์พุทของวงจรเปรียบเทียบสัญญาณเป็นไปตามตาราง 3.1

V_{in3}	Vo1	Vo2	Vo3	สี
น้อยกว่า 2.38 โวลท์	0	0	0	ดำ
2.38 < V_{in3} <2.8	0	0	1	เทาเข้ม
2.8 < V_{in3} < 3.2	0	1	1	เทาอ่อน
มากกว่า 3.2 โวลท์	1	1	1	ขาว

ตาราง 3.1

หมายเหตุ กระดาษที่ใช้เป็นกระดาษปอนด์สีขาว และใช้อัตราขยายแรงดันเท่ากับ 21

ง. วงจรลอจิกเกท

เนื่องจากระดับสัญญาณที่ได้จากส่วนเปรียบเทียบแรงดันประกอบด้วยตัวเลข 3 บิท ซึ่งในที่นี้จะใช้แทนความเข้มของสีเพียง 4 ระดับเท่านั้น ดังนั้นลดจำนวนบิทลงเพื่อประหยัดเนื้อที่ในหน่วยความจำในขณะบรรทึกข้อมูล โดยใช้วิธีลดข้อมูลซึ่งคำนวณได้จากคณิตศาสตร์ของบูลีน (Boolean equation) ดังรูป 3.6 ส่วนผลที่ได้จากการคำนวณในรูปของข้อมูล 2 บิทจะถูกป้อนเข้าสู่เครื่องคอมพิวเตอร์ และมีค่าดังแสดงในตาราง 3.2 ในหน้าถัดไป

AB C	00	01	11	10
0	0	d	d	d
1	1	0	1	d

$Y = A + BC = A(BC)$

หมายเหตุ A B C แทน V_{o1} V_{o2} V_{o3} ตามลำดับ และ d แทน don't care

รูป 3.6

ส่วนบิท X นั้นสามารถนำออกจากเอาท์พุท V_{o2} แล้วผ่านเข้าบัฟเฟอร์ (BUFFER) ได้เลย จากนั้นจะใช้ซอฟท์แวร์ (SOFTWARE) เพื่อลดระดับสัญญาณลงจาก 4 เป็น 3 เพื่อเพิ่มเสถียรภาพให้แก่ระบบ

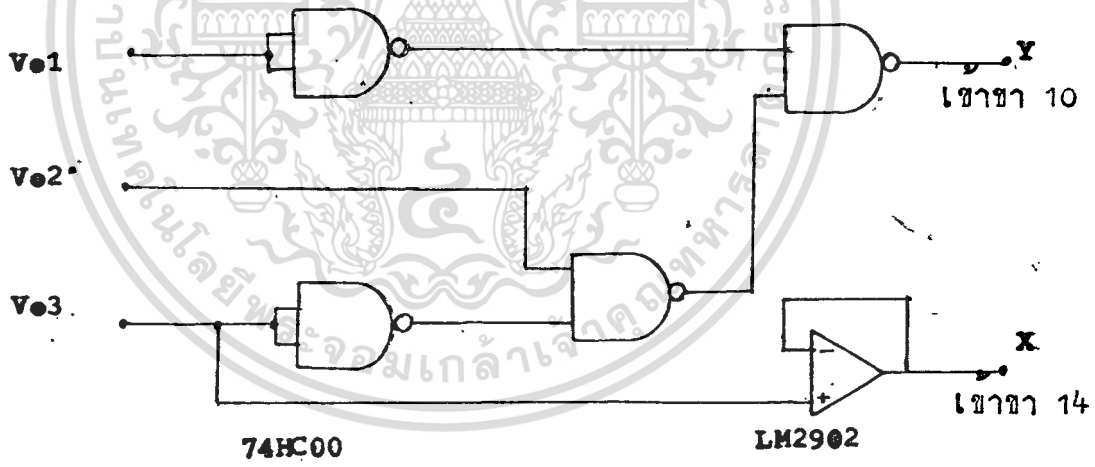
ABC	BIT X	BIT Y	สี
000	0	0	ดำ
001	0	1	เทาเข้ม
011	1	0	เทาอ่อน
111	1	1	ขาว

$$Y = \overline{A}(\overline{\overline{B}C})$$

$$X = B$$

ตาราง 3.2

อุปกรณ์ที่ใช้ทำหน้าที่เป็นวงจรลอจิกในที่นี้คือไอซีเบอร์ 74HC00 ที่ประกอบด้วยแนนด์เกท (NAND GATE) 4 ตัว ซึ่งเหมาะกับวงจรที่ได้จากสมการบูลีนพอดี โดยมีวงจรดังรูป 3.7

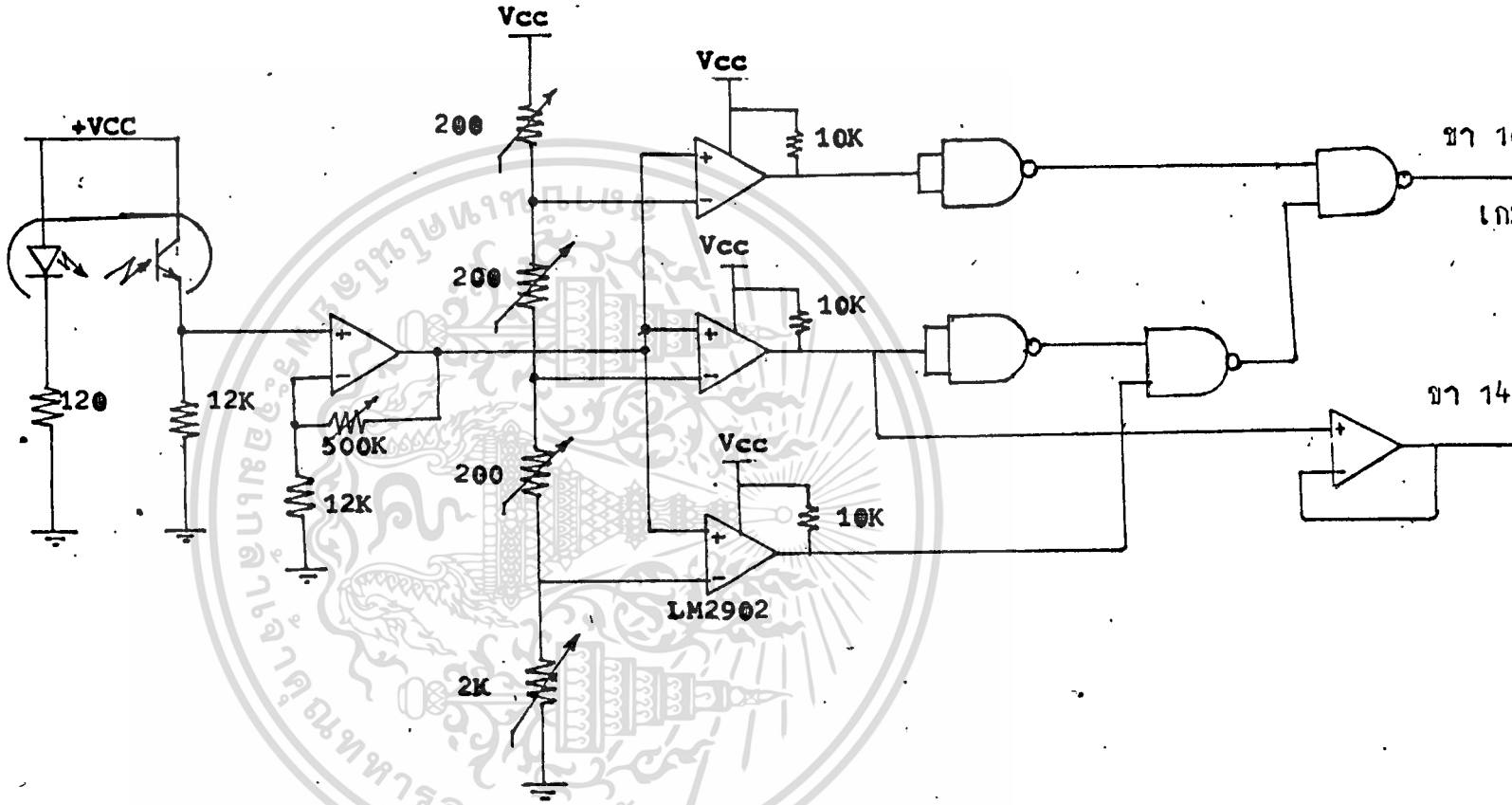


รูป 3.7

ส่วนรูป 3.8 ในหน้าถัดไปแสดงวงจรอิเล็คทรอนิคส์ทั้งหมด

3.2 โปรแกรมควบคุมเครื่องสแกน

โปรแกรมสำหรับควบคุมเครื่องสแกนเนอร์นี้ เลือกใช้ 2 ภาษาคือ ภาษาแอสเซมบลี้ (ASSEMBLY) และ QUICK BASIC V.3 เหตุที่เลือกใช้ 2 ภาษา นี้เพราะภาษาแอสเซมบลี้นั้นเหมาะกับการเชื่อมต่อกับฮาร์ดแวร์และความเร็วสูงกว่า ส่วนภาษาสูงนั้นเหมาะกับงานประเภท


Vcc
Vcc
+VCC
200
10K
ขา 1
Vcc
200
10K
120
12K
500K
12K
ขา 14
Vcc
200
10K
LM2902
2K

รูป 3.8

อินเทอร์เฟสกับผู้ใช้

สำหรับภาษาเบสิกนั้นเลือกใช้ QUICK BASIC เพราะเป็น BASIC แบบคอมไพเลอร์ ทำให้โปรแกรมมีความเร็ว และเชื่อมต่อกับภาษาแอสเซมบลี้ได้ง่าย ส่วนภาษาแอสเซมบลี้นั้นใช้ตัวแปลของ MICROSOFT จะขอแบ่งโปรแกรมออกเป็น 2 ส่วน คือส่วนภาษาเบสิกซึ่งเป็นส่วนโปรแกรมหลัก และส่วนภาษาแอสเซมบลี้

3.2.1 ส่วนโปรแกรมภาษาเบสิก

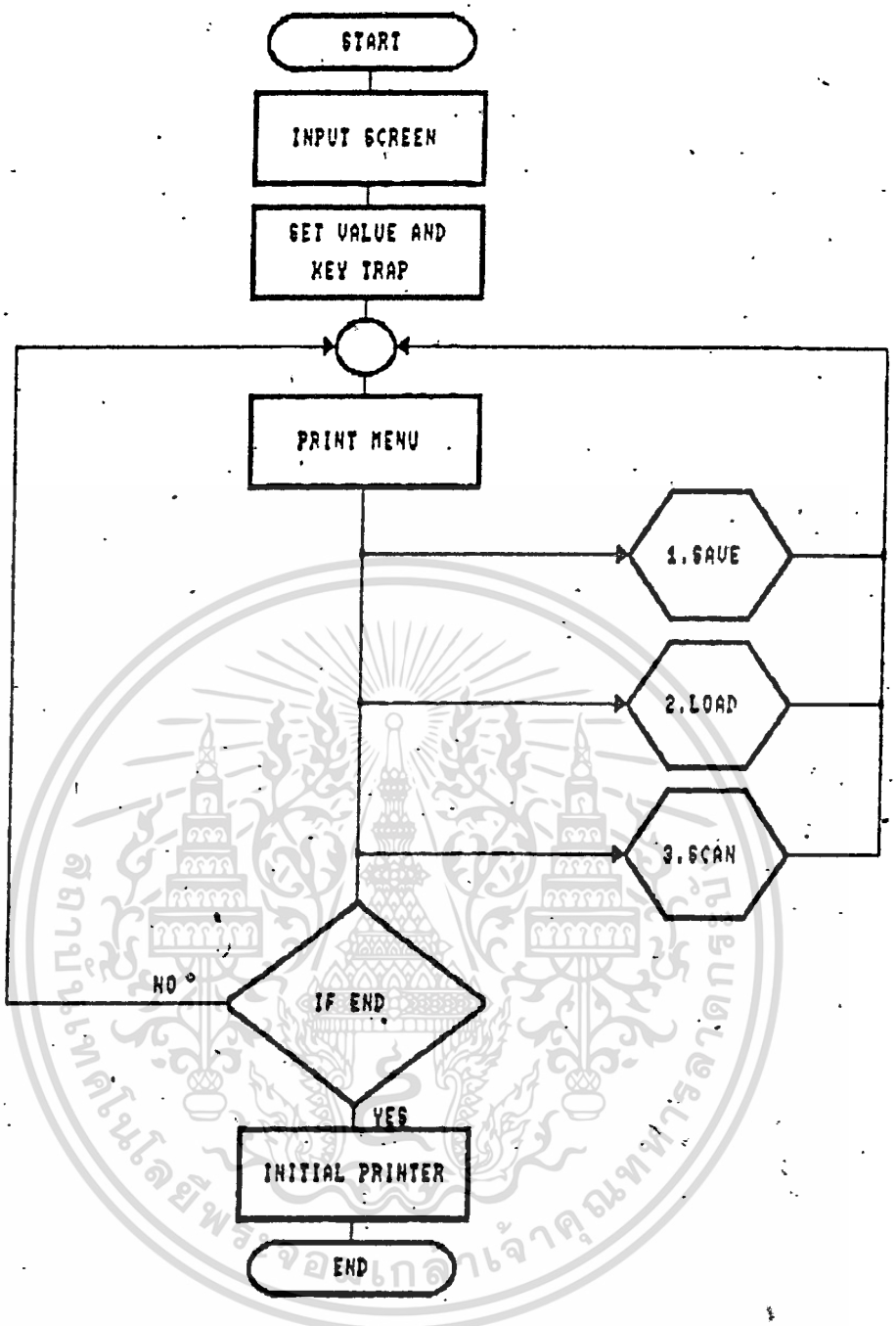
โปรแกรมที่เขียนนี้จะเป็นการทำเกี่ยวกับการติดต่อกับผู้ใช้ การตั้งค่าต่าง ๆ สำหรับโปรแกรมภาษาแอสเซมบลี้ และการจัดการเกี่ยวกับการ SAVE และ LOAD ภาพลงแผ่นดิสค์ เราอาจเขียน FLOW CHART ได้ดังรูป 3.9

จากรูป เริ่มต้นนั้นโปรแกรมจะทำการจองที่สำหรับเก็บภาพบริเวณ WINDOW MENU เพื่อจะสามารถนำภาพกลับคืนมาได้เมื่อลบ MENU ออกไป ถัดมาโปรแกรมจะถามถึงโหมดของสกรีนว่าโหมด 1 หรือ 2 ค่านี้จะนำไปใช้สำหรับตั้งค่าต่าง ๆ เช่น ตำแหน่งของ MENU และตำแหน่งที่เริ่มต้นเก็บภาพ คำสั่ง KEY ON และ ON KEY มีไว้สำหรับตั้งค่า KEY TRAP คือเมื่อมีการกดปุ่มที่ตั้งไว้โปรแกรมจะกระโดดไปทำโปรแกรมที่ตั้งไว้ทันที ในที่นี้เราเซ็ท F1 สำหรับการลบ MENU ออกไป เพื่อให้ภาพเต็มจอ ถัดมาจะพิมพ์ MENU พร้อมทั้งรอรับคีย์คำสั่งต่อไป คือ 1 SAVE, 2 LOAD, 3 SCAN, 4 EXIT จะขออธิบายการทำงานแยกเป็นส่วน ๆ ดังนี้

1 การ SAVE การ SAVE นี้จะกำหนดให้ผู้ใช้ระบุชื่อโดยจะเติมส่วนขยายเป็น .PIC ให้โดยอัตโนมัติ หลังจากนั้นจะทำการลบ MENU ออกไปและ RESTORE ภาพเดิมกลับมาแล้วทำการ SAVE การ SAVE ที่ใช้เป็นการ SAVE แบบไบนารีไฟล์ และเป็นการ SAVE ทั้งจอภาพซึ่งจะใช้หน่วยความจำทั้งสิ้น 16 K ไม่ว่าภาพเล็กหรือใหญ่ ซึ่งจะได้ทำการปรับปรุงให้เก็บเฉพาะภาพในส่วนที่ต้องการต่อไป

2 การ LOAD จะมีลักษณะเช่นเดียวกับการ SAVE ต่างกันเพียงแต่ว่าเมื่อ LOAD ภาพเสร็จแล้ว จะทำการ SAVE ภาพบริเวณ MENU ด้วยเพื่อให้สามารถเรียกคืนกลับมาได้เมื่อลบ MENU ออกไป

3 การ SCAN โปรแกรมในส่วนนี้จะทำหน้าที่ SET ค่าต่าง ๆ เพื่อส่งไปให้โปรแกรมภาษาแอสเซมบลี้อีกที ค่าที่จะให้ภาษาแอสเซมบลี้นั้นมี 5 ค่าด้วยกันคือ DELAY, MARGIN, LOOPSCAN, NOLINE, NOCOLUMN โดยจะอธิบายความหมายของค่าแต่ละค่าดังต่อไปนี้



รูป 3.9

DELAY หมายถึงเวลาหน่วงที่ใช้เพื่อทำให้ ข้อมูลที่อ่านได้กับหัวสแกนสัมพันธ์กัน ค่านี้ขึ้นอยู่กับความถี่และชนิดของเครื่องว่ารุ่นใด เช่น XT หรือ AT

MARGIN คือค่า TAB ของหัวสแกน การป้อนจะป้อนเป็นนิ้ว และจะนำมาคูณ 10 เพื่อแปลงเป็นจำนวนตัว ส่งให้แก่โปรแกรมย่อยถ้าให้ MARGIN เป็น 0 จะเซ็ทให้ MARGIN เท่ากับ 0.1 หรือ 1 ตัว

LOOPSCAN คือจำนวน LOOP ที่จะรับข้อมูลใน 1 LINE โดย 1 LOOP นั้นจะรับข้อมูลได้ 4 จุด (1 จุดแทนด้วยข้อมูล 2 บิท) LOOPSCAN จะหาได้โดย

(การคูณระยะที่ตั้งสแกนทางแนวนอน x ความละเอียดทางแนวนอน) / จำนวนจุดต่อ 1 LOOP

ซึ่งในที่นี้จะเท่ากับ

LOOPSCAN = (ระยะทางแนวนอน x 40) / 4

= ระยะทางแนวนอน x 10

NOLINE คือจำนวนเส้นที่จะให้เครื่องสแกนมีค่าเท่ากับระยะสแกนแนวตั้ง x 40

NOCOLUMN คือ จำนวนจุดที่จะให้เครื่องพิมพ์ พิมพ์เพื่อให้หัวเครื่องพิมพ์ พิมพ์ตามระยะที่ต้องการ จะมีค่าเท่ากับ

NOCOLUMN = ระยะสแกนตามแนวนอน x 60

60 คือ ค่าความหนาแน่นจุดของโหมดกราฟฟิคที่ใช้ ในการหาค่า NOCOLUMN นั้นจะต้องระวังค่าที่ได้มีค่าเกิน 7 เนื่องจากจะทำให้หัวสแกนชนกับขอบของเครื่องพิมพ์ เพราะว่าหัวสแกนยื่นมาทางข้างมากกว่าหัวเครื่องพิมพ์ การสั่งพิมพ์ยาวไปจะทำให้หัวสแกนชนขอบได้

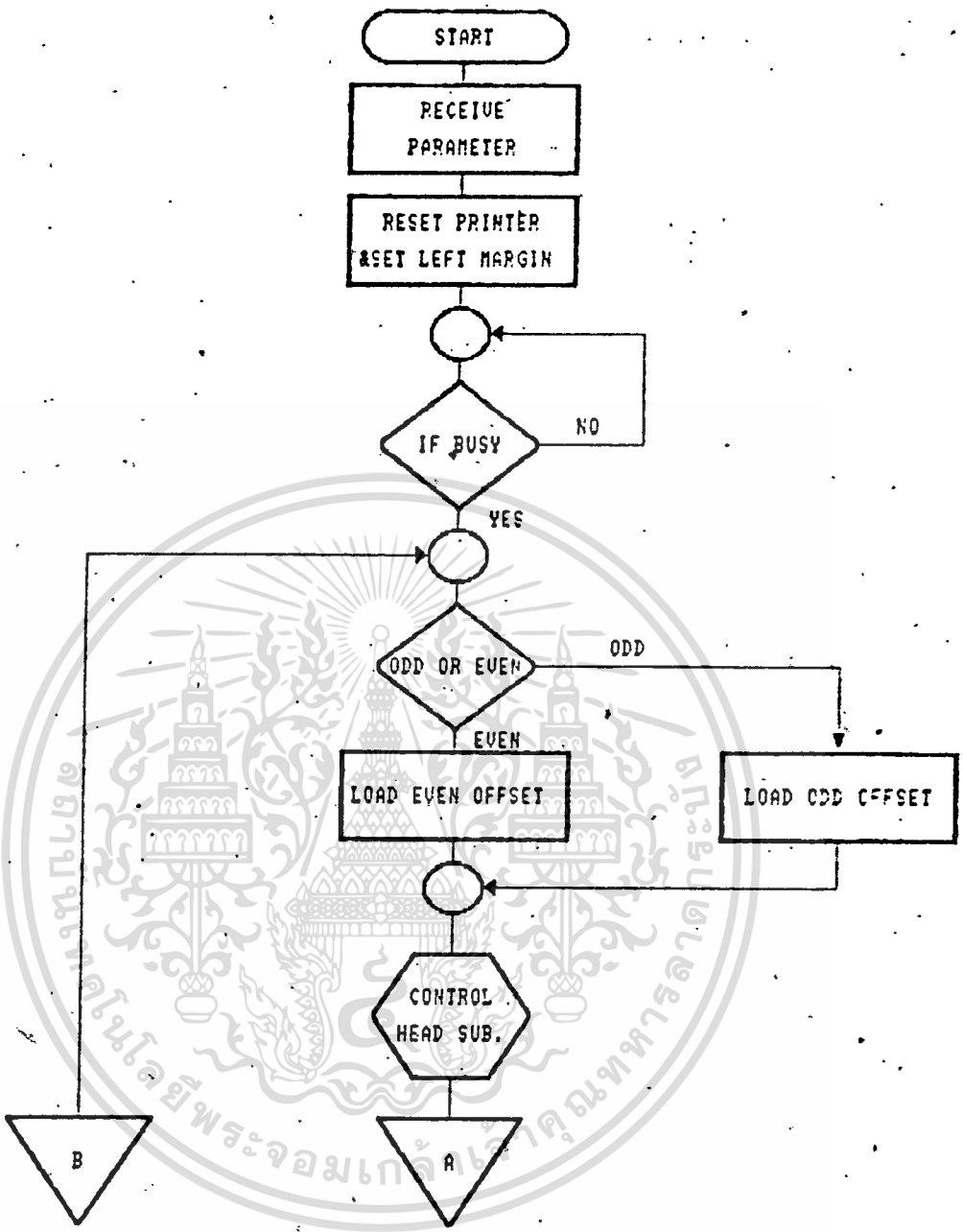
หลังจากได้ตั้งค่าต่าง ๆ แล้ว จะทำการส่งค่าไปให้โปรแกรมภาษาแอสเซมบลี้ต่อไป เมื่อสแกนเสร็จแล้ว โปรแกรมจะทำการ SAVE ภาพปริเวณ MENU เพื่อการเรียกกลับหลังจากลบ MENU ไปแล้ว

4 END โปรแกรมจะรีเซ็ทเครื่องพิมพ์ และจบการทำงาน

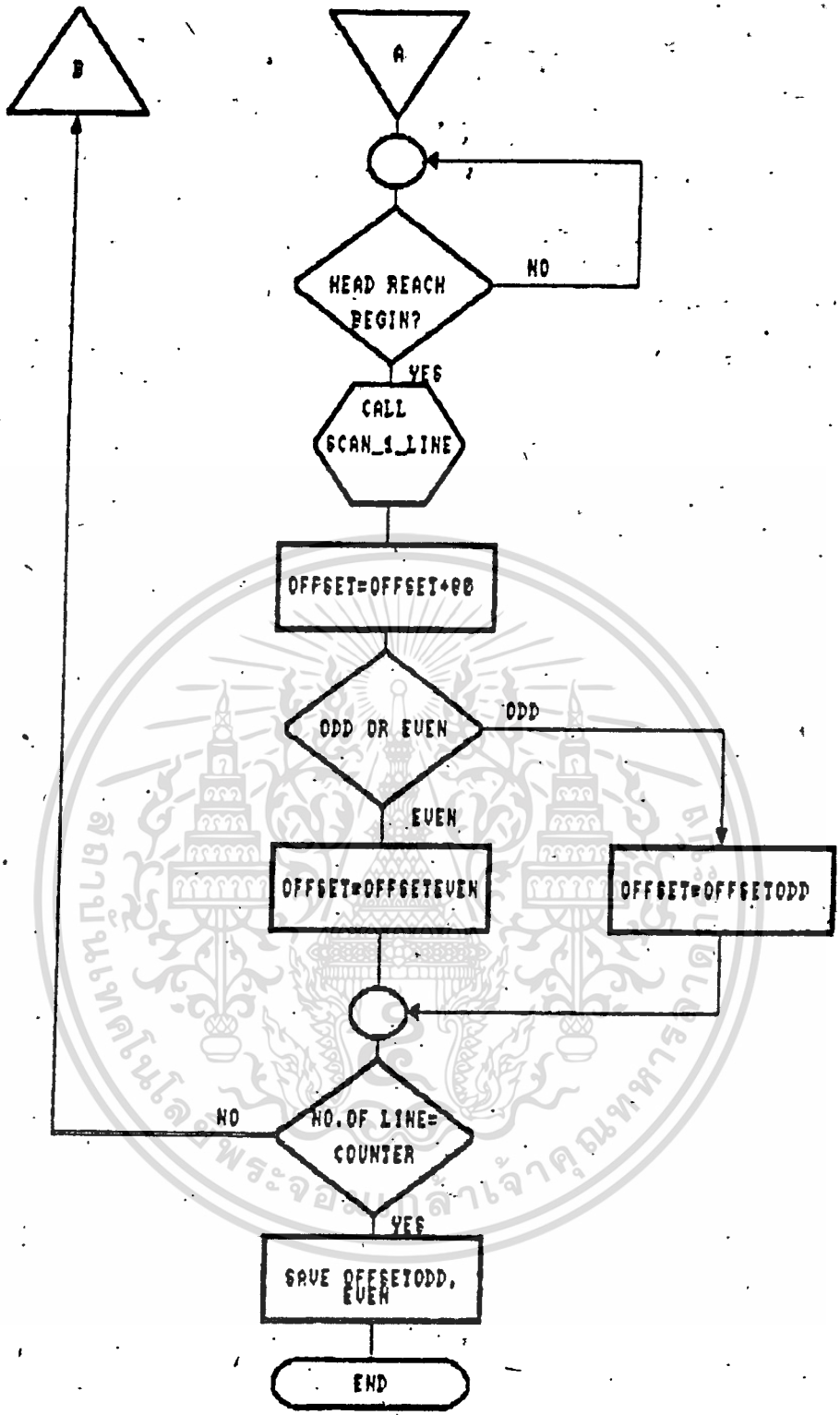
ดังนั้นจะเห็นว่าโปรแกรมทำงานพื้นฐานง่าย ๆ เท่านั้น ไม่เกี่ยวกับการควบคุมการรับส่งข้อมูล สั่งเครื่องพิมพ์แต่อย่างใด

3.2.2 โปรแกรมภาษาแอสเซมบลี้

โปรแกรมภาษาแอสเซมบลี้นี้จะใช้ในการควบคุมทางฮาร์ดแวร์ต่าง ๆ เช่น เครื่องพิมพ์, การ์ดแสดงผล และพอร์ทเกม หลักการทำงานของโปรแกรมจะเห็นได้จากโฟลว์ชาร์ตดังแสดงในรูป 3.10



รูป 3.10



รูป 3.10 (ต่อ)

โปรแกรมจะแบ่งเป็น 3 ส่วนใหญ่ ๆ คือ

1) โปรแกรมหลัก ทำหน้าที่ควบคุมการทำงานทั้งหมด

2) โปรแกรมย่อย ชื่อ CONTROL-HEAD ทำหน้าที่ควบคุมหัวเครื่องพิมพ์

3) โปรแกรมย่อย ชื่อ SCAN-1-LINE ทำหน้าที่ในการรับข้อมูลจากพอร์ทเกม และนำออกแสดงที่วีดีโอแรม

1) <u>โปรแกรมหลัก</u> จะมีหลักการทำงานดังต่อไปนี้ เริ่มแรกโปรแกรมรับค่าตัวแปรต่าง ๆ ที่ส่งมาจากส่วนภาษาเบสิกเก็บไว้ใน DATASEGMENT จากนั้นจึงทำการรีเซ็ตเครื่องพิมพ์ และตั้งระยะ MARGIN และ LINE FEED ต่อมาโปรแกรมจะเช็คว่าเครื่องพิมพ์พร้อมจะสแกนหรือยัง คืออยู่ ณ ตำแหน่งเริ่มต้นหรือยัง ถ้าพร้อมแล้วก็จะเริ่มเข้าสู่การสแกน ก่อนการสแกนจะทำการตรวจสอบว่าเป็นเส้นคู่หรือเส้นคี่ เพื่อตั้ง POINTER ในการเก็บข้อมูลลงวีดีโอแรมได้ถูกต้อง ตั้งเสร็จแล้วจึงจะทำการเรียกโปรแกรมย่อย CONTROL-HEAD เพื่อควบคุมให้หัวเครื่องพิมพ์เริ่มเคลื่อนที่ ในขณะหัวเครื่องพิมพ์เคลื่อนที่นั้น ก็จะทำตรวจสอบว่าหัวเคลื่อนกลับมายังจุดเริ่มต้นเพื่อเริ่มต้นสแกนหรือยัง ถ้าถึงจุดเริ่มต้นแล้ว ก็จะเรียกโปรแกรมย่อย SCAN-1-LINE เมื่อสแกนครบ 1 บรรทัดจะทำการบวกค่า OFFSET ด้วย 80 และเก็บสู่ตัวแปร OFFSET ODD หรือ EVEN ตามแต่ค่าเริ่มต้นว่าใช้ OFESET ODD หรือ EVEN การบวกด้วย 80 เพื่อว่าในการสแกนบรรทัดใหม่บนจอจะได้ขึ้นบรรทัดใหม่ด้วยเสมอ เมื่อจบการสแกน 1 บรรทัดแล้วจะตรวจสอบว่าครบจำนวนที่ตั้งไว้หรือไม่ ถ้าครบแล้วก็จะ RESTORE ค่า OFESET ODD และ EVEN และกลับสู่ส่วนโปรแกรมภาษาเบสิกถ้าไม่ครบก็จะไปทำงานสแกนเส้นใหม่ต่อไป

2) <u>โปรแกรมย่อย CONTROL-HEAD</u> โปรแกรมนี้ทำหน้าที่ในการควบคุมหัวเครื่องพิมพ์ โดยจะทำการเซ็ตโหมดกราฟฟิค และตั้งจำนวนจุดเพื่อให้หัวเลื่อนไปตามระยะที่ต้องการ และทำการส่ง LF 1 ครั้ง ดังนั้นเครื่องพิมพ์จะเลื่อนขึ้น 1 บรรทัด เมื่อพิมพ์เสร็จแล้ว

3) <u>โปรแกรมย่อย SCAN-1-LINE</u> โปรแกรมนี้จะทำการรับข้อมูลจากพอร์ทเกมส่งไปยังวีดีโอแรม เป็นจำนวน 1 เส้น ซึ่ง 1 เส้นจะสแกนได้กี่จุดนั้นจะขึ้นอยู่กับ ค่า LOOP SCAN ที่ส่งมาจากส่วนภาษาเบสิก โปรแกรมในส่วนนี้จะทำการรับข้อมูล 2 บิท จากพอร์ทเกมนำมาต่อกันเป็นข้อมูล 1 ไบท์ (4จุด) จึงจะนำลงไปเก็บในวีดีโอแรม 1 ที รวมทั้งมีการหน่วงเวลาไว้ด้วย เนื่องจากหัวเครื่องพิมพ์เลื่อนช้ากว่าการทำงานของโปรแกรม ถ้าไม่มีการหน่วงเวลาจะทำให้ได้จุดเดียวเป็นจำนวนหลายค่า ซึ่งผิดกับความจริง การหน่วงเวลานี้จะไม่เท่ากัน เนื่องจากแต่ละส่วนในการต่อข้อมูลเป็น 1 ไบท์ นั้นใช้คล็อกไซเคิ้ลไม่เท่ากัน จึงจำเป็นต้อง

หน่วงเวลาเพื่อให้แต่ละจุดที่ได้มีระยะห่างเท่า ๆ กัน ข้อควรระวัง คือ หลังจากออกจาก โปรแกรมย่อยนี้ จะต้องทำการ LOAD DATASEGMENT เดิม กลับมาด้วย มิฉะนั้นจะทำให้การทำงานของโปรแกรมผิดพลาด การพิมพ์ภาพจากจอสู่เครื่องพิมพ์ทำได้โดยใช้โปรแกรม GRAPHICS ที่มีใน DOS 3.2 (หาอ่านได้จากคู่มือ DOS) การพิมพ์นั้นเราสามารถตั้งว่าจะ REVERSE ภาพหรือไม่ก็ได้ ข้อเสียของการใช้โปรแกรมนี้ก็คือ ไม่สามารถกำหนดจุดที่ต้องการพิมพ์ภาพได้ จึงต้องพิมพ์ทั้งจอเสมอ ซึ่งจะเสียเวลามากในกรณีที่ต้องการรูปเพียงนิดเดียว ภาพที่ได้จะมี 4 ระดับเช่นเดียวกับการสแกน

4.1 วงจรอิเล็คทรอนิคส์

จากการศึกษาและวิจัยระบบสแกนโดยใช้หัวอ่านชนิดที่กล่าวมาในบทก่อนหน้านี้ พบว่าเราสามารถปรับปรุงและพัฒนาให้ระบบนี้มีขนาดเล็กลง และยังสามารถเพิ่มประสิทธิภาพของการแสดงผลบนจอมอนิเตอร์ได้อีกด้วย

ข้อแตกต่างประการสำคัญระหว่างหัวอ่านรุ่นแรกและรุ่นที่พัฒนาขึ้นใหม่คือหัวอ่านทั้งสองนี้จะส่งสัญญาณไปยังอินพุทของพอร์ทเกมที่ต่างกัน ดังที่ได้กล่าวในบทที่แล้วว่า หัวอ่านชนิดแรกจะประกอบด้วยฮาร์ดแวร์ที่เปลี่ยนสัญญาณภาพให้เป็นสัญญาณดิจิตอล แล้วจึงป้อนสัญญาณนี้เข้าไปยังอินพุทสำหรับสัญญาณดิจิตอลของพอร์ทเกม แต่หัวอ่านรุ่นที่สอง สัญญาณอนาล็อกที่ได้จะไม่ผ่านขั้นตอนประมวลผลใด ๆ หากแต่จะถูกป้อนเข้าสู่อินพุทสำหรับสัญญาณอนาล็อกของพอร์ทเกมโดยตรง การแปลงสัญญาณและการประมวลผลต่าง ๆ จะกระทำโดยใช้วงจรต่าง ๆ ที่มีอยู่ในเครื่องไมโครคอมพิวเตอร์แล้ว ซึ่งเป็นผลให้วงจรอิเล็คทรอนิคส์สำหรับหัวอ่านชนิดนี้มีขนาดเล็กลงมาก และราคาของชิ้นงานจริงถูกลงด้วย

นอกจากการเปลี่ยนแปลงในส่วนของวงจรอิเล็คทรอนิคส์แล้ว เครื่องสแกนที่ใช้หัวอ่านรุ่นที่สองนี้ยังได้ถูกพัฒนาให้สามารถแบ่งระดับความเข้มของสัญญาณออกเป็น 8 ระดับ การแสดงข้อมูลภาพบนจอมอนิเตอร์จึงสามารถแสดงภาพที่มีระดับสีถึง 8 โทน ทำให้การแยกแยะสีเห็นได้ชัดเจนยิ่งขึ้น

การลดอุปกรณ์ด้านฮาร์ดแวร์ จะเป็นผลให้ซอฟท์แวร์สำหรับใช้ร่วมกับหัวอ่านนี้ยุ่งยากมากขึ้นตามลำดับ นอกจากจะต้องควบคุมการแปลงสัญญาณอนาล็อกเป็นดิจิตอลแล้ว โปรแกรมยังจะต้องแบ่งระดับสัญญาณออกเป็น 8 ระดับด้วย ซึ่งรายละเอียดของโปรแกรมจะนำมาอธิบายในหัวข้อต่อไป

4.1.1 อินพุทสำหรับสัญญาณอนาล็อกในพอร์ทเกม

การทำงานของหัวอ่านชนิดที่สองจะต่างกับการทำงานของหัวอ่านชนิดแรกโดยสิ้นเชิง ซึ่งก่อนที่จะศึกษาการทำงานโดยใช้หัวอ่านชนิดที่สองนี้ ควรทำความเข้าใจหลักการทำงานและรายละเอียดของอินพุทสำหรับสัญญาณอนาล็อกของพอร์ทเกมเสียก่อน

จากรูป 4.1 จะเห็นว่าอินพุทของพอร์ทเกมจะมีอยู่ 8 อินพุทด้วยกัน แบ่งออกเป็นอินพุทสำหรับสัญญาณดิจิตอล 4 อินพุท และสำหรับสัญญาณอนาล็อกอีก 4 อินพุท โดยหัวอ่านชนิดแรกใช้

อินพุทสำหรับสัญญาณดิจิตอล 2 บิท ส่วนหัวอ่านชนิดที่สองจะใช้อินพุทสำหรับสัญญาณอนาล็อกเพียงบิทเดียว

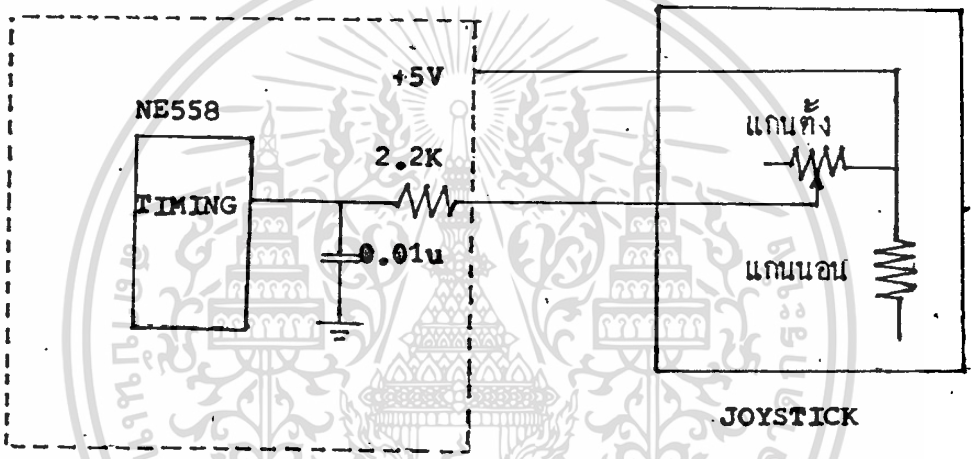
การใช้งานอินพุทสำหรับสัญญาณอนาล็อกมีหลักการดังนี้คือ เมื่อโปรแกรมส่งสัญญาณ OUT

รูป 4.1

มายังแอดเดรสของพอร์ทเกม ไอซีเบอร์ 558 ที่ต่ออยู่เป็นวงจรโมโนสเตเบิ้ล (monostable circuit) ภายในพอร์ทเกมจะเริ่มทำงาน พัลส์ที่เอาท์พุทของวงจรโมโนสเตเบิ้ลจะมีความกว้างที่ขึ้นอยู่กับความต้านทานของโพเทนชิโอมิเตอร์ที่ต่อระหว่างไฟบวก 5 โวลท์กับอินพุท ตามรูป 4.2 แล้วจึงใช้คำสั่ง IN เพื่อทดสอบสถานะของพัลส์ในขณะนั้น (ความต้านทานของ potentiometer จะมีผลต่อเวลาในการชาร์จตัวเก็บประจุตามค่าคงตัวของเวลา RC)

ซึ่งสามารถคำนวณเวลาหน่วงสำหรับเอาท์พุทจากวงจรโมโนสเตเบิ้ล ที่ใช้แรงดันคงที่ค่าหนึ่งในการชาร์จประจุ ดังสมการนี้

$$T = 24.2 + (0.011 \times R) \quad \text{ไมโครวินาที}$$



รูป 4.2

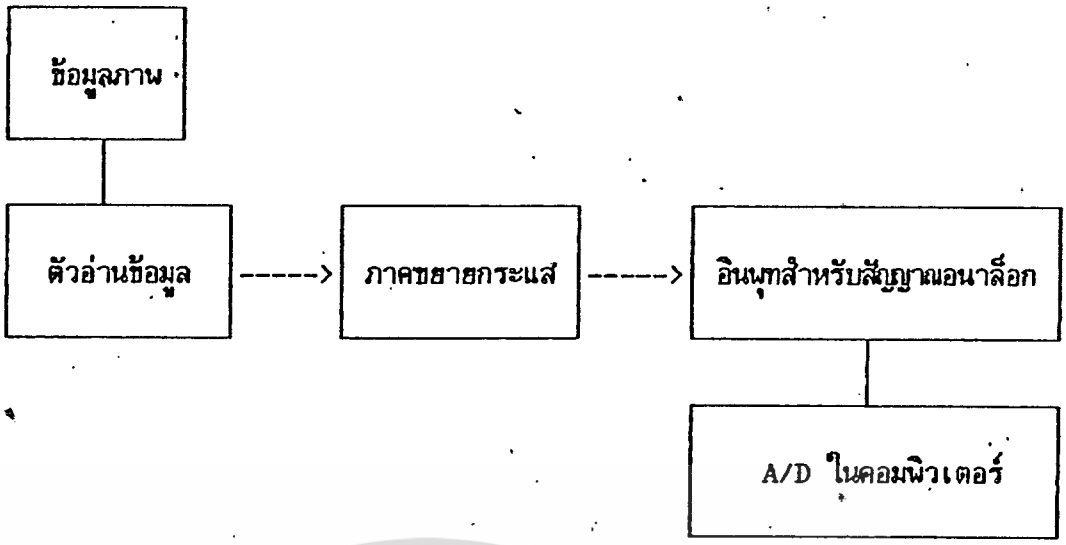
R : ค่าความต้านทาน

T : ความกว้างของสัญญาณเอาท์พุท

สมการนี้จะใช้ได้เฉพาะกรณีที่เราใช้แรงดันคงที่ (VOLTAGE SOURCE) ในการชาร์จตัวเก็บประจุเท่านั้น เช่นการทำงานเมื่อใช้จอยสติ๊ก เป็นต้น แต่ในโครงงานนี้ เราจะใช้การชาร์จที่แตกต่างออกไป คือจะใช้การชาร์จด้วยแหล่งจ่ายกระแสคงที่ (CURRENT SOURCE) ซึ่งต้องใช้การคำนวณที่ต่างออกไป ดังจะได้อธิบายในหัวข้อต่อไป

4.1.2 <u>โครงสร้างของวงจรอิเล็คทรอนิคส์ (รุ่นที่ 2)</u>

บล็อคไดอะแกรมในรูป 4.3 แสดงลักษณะการทำงานของเครื่องสแกนที่ใช้หัวอ่านรุ่นที่สอง



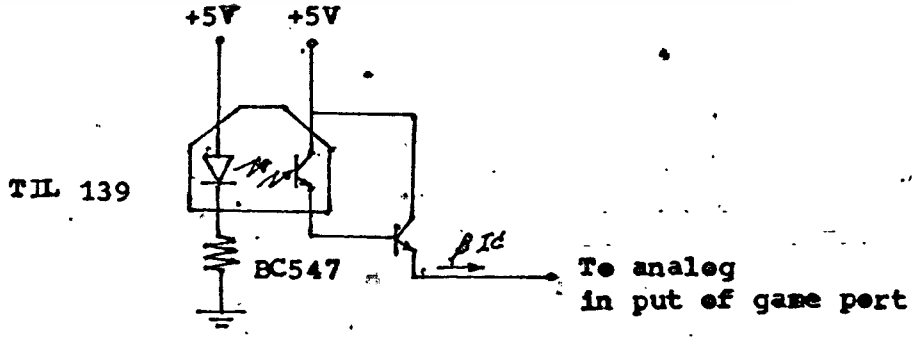
รูป 4.3

วงจรส่วนอิเล็คทรอนิคส์นี้ประกอบด้วย

1. ภาคเปลี่ยนสัญญาณแสงเป็นสัญญาณไฟฟ้า ภาคนี้ประกอบด้วยออพโต้คัปเปลอร์ที่ใช้นี้ยังคงเป็นเบอร์ TIL 139 เช่นเดียวกับที่ใช้ในหัวอ่านชนิดแรก โดยที่ไดโอดทางด้านส่งจะต่อกับตัวต้านทานค่า 120 โอห์ม ส่วนตัวโฟโต้ทรานซิสเตอร์ที่ด้านรับจะถูกต่อเข้ากับภาคขยายกระแส และมีหน้ากากบังไว้เช่นเดิม สัญญาณจากภาคนี้จะถูกส่งไปยังภาคขยายกระแส

2. ภาคขยายกระแส เนื่องจากสัญญาณที่ใช้ในหัวอ่านชนิดที่สองนี้คือกระแส ดังนั้นจึงสามารถนำสัญญาณที่ได้ไปใช้ได้เลย แต่อย่างไรก็ตาม กระแส I_c ที่ได้จากโฟโต้ทรานซิสเตอร์มีค่าต่ำมาก จึงจำเป็นต้องนำเข้าภาคขยายกระแสนี้ ซึ่งประกอบขึ้นจากการต่อทรานซิสเตอร์เบอร์ BC 547 แบบดาร์ลิงตันกับโฟโต้ทรานซิสเตอร์ แล้วจึงป้อนสัญญาณกระแสเข้าสู่พอร์ทเกมต่อไป

รูป 4.4 แสดงส่วนของวงจรอิเล็คทรอนิคส์สำหรับหัวอ่านรุ่นที่ 2

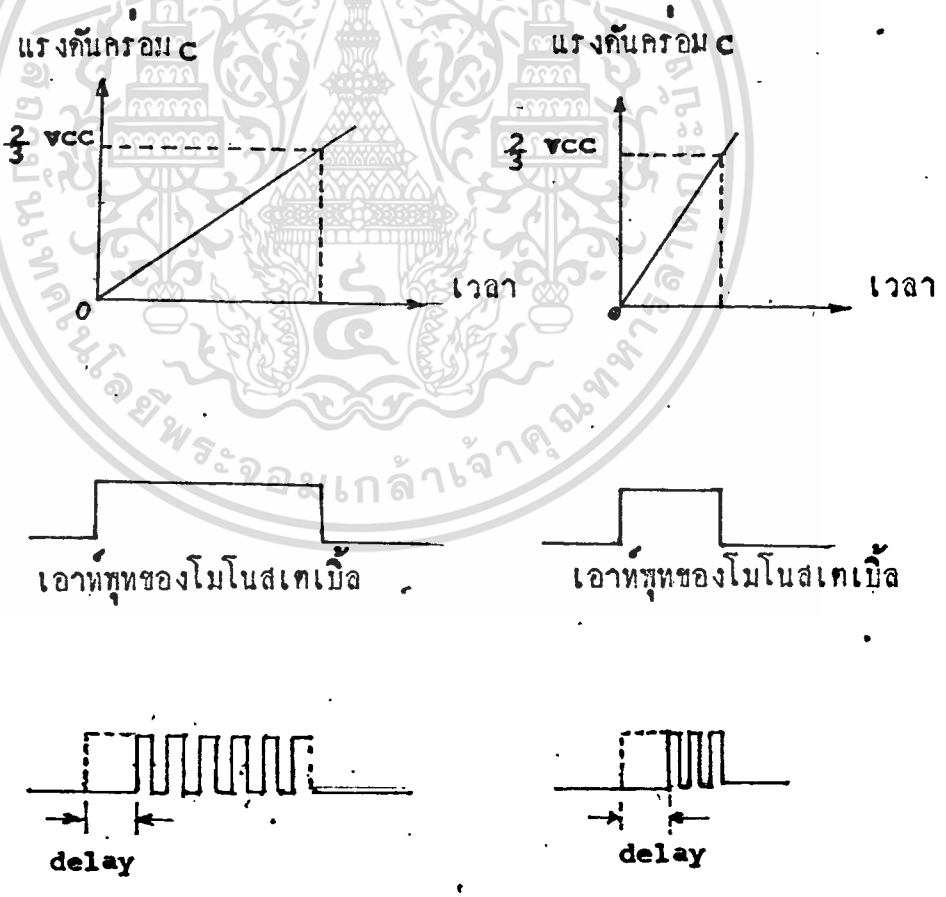


รูป 4.4

4.1.3 หลักการทำงาน

จากการศึกษาหลักการทำงานภายในพอร์ทเกม เมื่อใช้อินพุทสำหรับสัญญาณอนาล็อก พบว่าหากเราสามารถใช้วิธีอื่นในการชาร์จตัวเก็บประจุที่ต่อกับไอซีเบอร์ 558 (นอกเหนือจากชาร์จประจุโดยต่อความต้านทานปรับค่าได้ดังที่กล่าวมาแล้ว) ก็จะสามารถตรวจวัดและรับข้อมูลผ่านอนาล็อกบิทของพอร์ทเกมได้เช่นกัน โดยยังจะต้องรักษาเงื่อนไขเดิมที่ว่าอัตราการชาร์จตัวเก็บประจุนี้ต้องแปรตามความเข้มแสงที่ได้จากจุดของข้อมูลภาพ

การนำออฟโตคับเปิ้ลมาใช้กับอินพุทสำหรับสัญญาณอนาล็อกนี้จึงมีความเหมาะสมมาก เนื่องจากอัตราการชาร์จตัวเก็บประจุจะแปรตามกระแส I_c ตามสมการ $i = c(dv/dt)$ และกระแส I_c นี้จะแปรตามความเข้มของแสงสะท้อนตามจุดต่าง ๆ บนภาพ นั่นคือเมื่อจุดนั้นมีสีดำ แสงจะถูกดูดกลืน ทำให้ I_c มีค่าต่ำ และการชาร์จตัวเก็บประจุจนกระทั่งระดับแรงดันมีค่าสูงกว่า $(2V_{cc}/3)$ จะกินเวลานานกว่าจุดที่มีสีขาว ทำให้เวลาหน่วงของพัลส์(ที่ได้จากเอาท์พุทของวงจรโมโนสเตเบิ้ล)สำหรับจุดสีดำมีค่ามากกว่า ดังแสดงในรูป 4.5



รูป 4.5

หลังการส่งสัญญาณ OUT และวงจรโมโนสเตเบิ้ลสร้างพัลส์ออกมาแล้ว จึงส่งคำสั่ง IN ไปยังแอดเดรสของพอร์ทเกม เพื่อตรวจสถานะของพัลส์ และใช้ซอฟท์แวร์นับเวลาหน่วงของเอาท์พุทนั้น ดังรูป 4.5 b ก็จะได้จำนวนคลื่อกสำหรับความเข้มของจุดนั้น ๆ และเมื่อใช้ซอฟท์แวร์แบ่งระดับจำนวนคลื่อกที่ได้ ก็จะเกิดโทนสีต่าง ๆ ตามต้องการ

จากการทดลองพบว่าถ้าป้อนกระแส I_c เข้าสู่อินพุทสำหรับสัญญาณอนาล็อกโดยตรง เอาท์พุทของวงจรโมโนสเตเบิ้ลจะมีความกว้างมากเกินไป เนื่องจากกระแส I_c ที่ชาร์จตัวเก็บประจุจะชาร์จด้วยอัตราที่ต่ำมาก ซึ่งอาจทำให้เวลาที่หน่วงเอาท์พุทนี้ นานเกินกว่าเวลาที่หัวของเครื่องพิมพ์ใช้ในการเคลื่อนจากจุดหนึ่งไปยังจุดถัดไป เป็นผลให้จำนวนคลื่อกที่วัดได้ผิดไปด้วย ดังนั้นเพื่อความแน่นอน จึงจำเป็นต้องเพิ่มภาคขยายกระแสเพื่อลดเวลาหน่วงดังกล่าวให้แคบลง นอกจากนี้ เวลาส่วนดีเลย์ที่ใช้ก่อนเริ่มต้นนับจะช่วยลดจำนวนพัลส์ที่นับได้ให้จำกัดอยู่เพียง 256 ลูกเท่านั้นเพื่อลดจำนวนบิทที่ใช้นับให้เหลือ 8 บิท และการเพิ่มเวลาหน่วงนี้ยังช่วยให้สามารถใช้โปรแกรมนี้กับเครื่องคอมพิวเตอร์ที่ใช้ซีพียูความเร็วสูงหรือเครื่องเทอร์โบได้ มิเช่นนั้นแล้ว จำนวนคลื่อกที่นับได้จะเกิน 256 ลูกไปมาก และทุกครั้งที่เกินมันก็จะเริ่มต้นนับเป็นศูนย์ใหม่ ทกให้ภาพที่ได้ผิดเพี้ยนไป

4.2 <u>โปรแกรมควบคุมเครื่องสแกน (รุ่นที่ 2)</u>

โปรแกรมส่วนควบคุมยังคงแบ่งเป็น 2 ส่วนเช่นเดิม โดยในส่วนโปรแกรมภาษาเบสิกไม่ได้เปลี่ยนแปลงอะไร ยกเว้นค่าคงที่ของการหน่วงเวลา (DLY) เปลี่ยนจาก 648 เป็น 10288 และ 1000 เป็น 22588 ในส่วนของภาษาแอสเซมบลี้นั้นได้เปลี่ยนใหม่หมด โดยมีโฟลว์ชาร์ตดังรูป 4.6 และแสดงไว้ในภาคผนวก ข. โปรแกรมภาษาแอสเซมบลี้ทำหน้าที่ในการสแกนภาพเพื่อนำออกแสดงทางวีดีโอแรม นั้นคือ บังคับการเคลื่อนที่ของหัวเครื่องพิมพ์ รับข้อมูลมาทางพอร์ทจอยสติ๊ก ซึ่งอาศัยพอร์ทนี้แปลงสัญญาณภาพจากอนาลอตเป็นค่าตัวเลขทางดิจิตอล และทำการประมวลผลตัวเลขเหล่านี้แปลงเป็นรูปภาพ

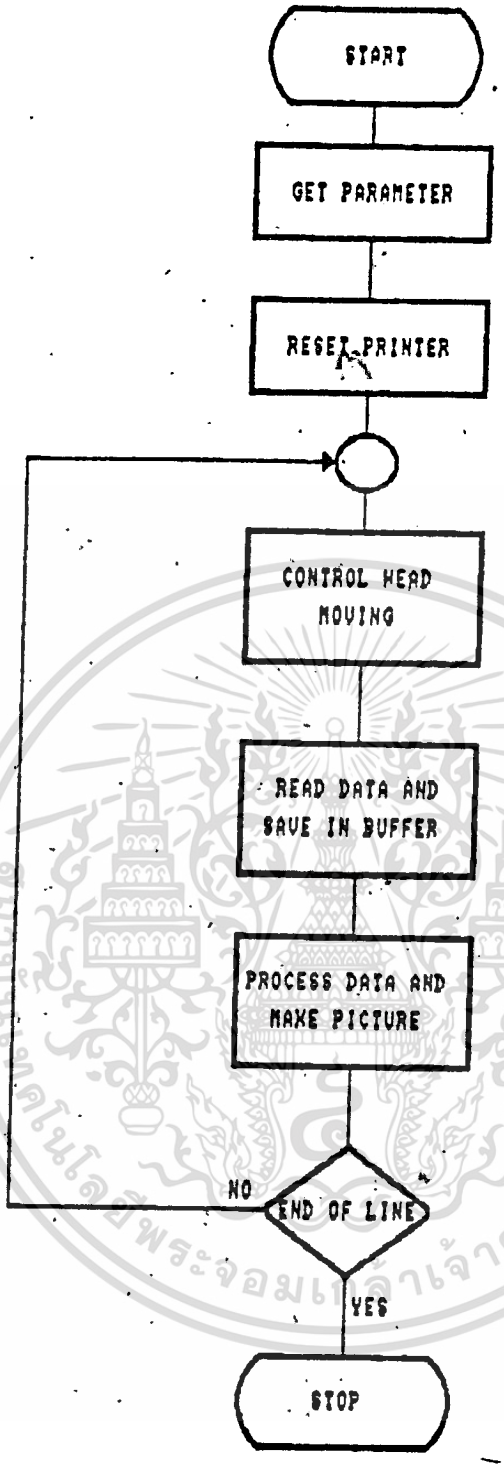
โปรแกรมนี้มีโพรซิเดอร์หลักชื่อ BEGIN โดยเริ่มแรกรับพารามิเตอร์ต่าง ๆ จากส่วนโปรแกรมภาษาเบสิกซึ่งได้แก่

NOLINE : จำนวนบรรทัดที่ทำกับสแกน

TAB : กำหนดกั้นหน้าซ้าย (LEFT MARGIN)

CONST1 : ค่าคงที่ของการหน่วงเวลา (DLY)

NOCOLUM : จำนวนของคอลัมน์ที่ส่งให้เครื่องพิมพ์ในโหมดกราฟิค



รูป 4.6

LOOPSCAN : จำนวนครั้งของการสแกนใน 1 บรรทัด

หลังจากรับค่าพารามิเตอร์ต่าง ๆ แล้ว ก็จะรีเซ็ตเครื่องพิมพ์ เพื่อให้เครื่องพิมพ์อยู่ในสภาวะปกติจากนั้นก็จะกำหนดค่าการทำงานของเครื่องพิมพ์ให้มีระยะ LINE FEED = 5/216 นิ้ว ตั้งระยะกั้นหน้าซ้าย และรอให้หัวเครื่องพิมพ์เคลื่อนมาถึงขอบซ้ายสุด เพื่อพร้อมที่จะทำการสแกนต่อไป

เมื่อหัวสแกนมาถึงขอบซ้ายสุดในระยะที่กำหนดไว้แล้ว ก็จะกำหนดจำนวนบรรทัดที่ต้องการสแกนในรีจีสเตอร์ BX โดยในการสแกนแต่ละบรรทัดจะมีขั้นตอนดังนี้

1 เรียกโปรแกรมย่อย ชื่อ HEAD

2 รอจนหัวเครื่องพิมพ์เคลื่อนมาถึงขอบซ้ายสุด

3 เรียกโปรแกรมย่อยชื่อ ONE-LINE

4 เรียกโปรแกรมย่อยชื่อ PROCIMAGE

5 ตรวจสอบว่าเป็นบรรทัดสุดท้ายหรือไม่ ถ้าใช่ให้หยุด ถ้าไม่ใช่ให้เริ่มทำใหม่

4.2.1 <u>โปรแกรมย่อย HEAD</u>

ในส่วนนี้จะกำหนดให้เครื่องพิมพ์ทำการพิมพ์ในโหมดกราฟิค โดยเลือกใช้โหมด K หรือ 60 จุด/นิ้ว ทั้งนี้เพราะเวลาในการพิมพ์ของโหมดนี้สั้นสุด ซึ่งจะทำให้การสแกนทำได้เร็วขึ้น การสั่งเครื่องพิมพ์ให้อยู่ในโหมดกราฟิค K นี้ทำได้โดยส่งอักขระ "ESC", "K", N1, N2 ไปยังเครื่องพิมพ์ตามลำดับ โดยที่ค่า N1 และ N2 จะเป็นตัวบอกเครื่องพิมพ์ว่าให้พิมพ์ในโหมดกราฟิคกี่คอลัมน์ (จุด) ซึ่งได้จากค่า NOCOLUM โดย N1 และ N2 คือไบท์ล่างและไบท์บนของ NOCOLUM ตามลำดับ จำนวนคอลัมน์ทั้งหมดที่เครื่องพิมพ์จะตีความคือ N1+(N2x 256) เช่นถ้าต้องการให้พิมพ์ในโหมดกราฟิค 400 คอลัมน์จะต้องให้ N1 มีค่า 144 และ N2 มีค่า 1 จำนวนคอลัมน์ในการพิมพ์ของโหมดกราฟิคนี้หาได้จาก ความกว้างของภาพ (นิ้ว)x60

เมื่อเครื่องพิมพ์รับคำสั่งหรืออักขระทั้ง 4 ตัวนี้แล้ว มันจะรอรับข้อมูลเพื่อพิมพ์ในโหมดกราฟิคนี้ตามจำนวนของค่า N1 และ N2 เช่น กำหนด N1 = 200 และ N2 = 0 หมายความว่าข้อมูลที่จะส่งให้เครื่องพิมพ์อีก 200 ไบท์ต่อไป จะเป็นข้อมูลในการพิมพ์ของโหมดกราฟิค ซึ่งลักษณะการพิมพ์จะเป็นอย่างไรนั้นศึกษาได้จากคู่มือเครื่องพิมพ์ แต่ในที่นี้ของกล่าวแต่เพียงว่าถ้าเราส่งค่า 0 ให้กับเครื่องพิมพ์ เครื่องพิมพ์จะไม่พิมพ์อะไรออกมา เหมือนเป็นการพิมพ์ช่องว่าง (BLANK) เท่านั้น ดังนั้นเมื่อเรากำหนดค่า N1 และ N2 และส่งค่า 0 ให้เครื่องพิมพ์ตามจำนวนของ N1 และ N2 ก็จะทำให้หัวเครื่องพิมพ์เคลื่อนที่ไปโดยไม่ได้พิมพ์

อะไรออกมาเลย

หลังจากส่งคำสั่งในการพิมพ์โหมดกราฟฟิคและข้อมูลทั้งหมดแล้ว หัวเครื่องพิมพ์จะยังไม่เคลื่อนที่จนกว่าเราจะส่งคำสั่ง LF ดังนั้นเราจึงต้องส่งคำสั่ง LF ไปให้เครื่องพิมพ์ด้วย

4.2.2 โปรแกรมย่อย ONE LINE

ในส่วนนี้จะทำงานในขณะที่หัวเครื่องพิมพ์เริ่มเคลื่อนที่ออกจากตำแหน่งซ้ายสุด ที่กำหนดไว้จนกระทั่งหยุดการทำงานพร้อม ๆ กับเมื่อหัวเครื่องพิมพ์เคลื่อนไปได้ระยะทางที่กำหนดไว้ (ความกว้างของภาพ)

เริ่มแรกจะมีการหน่วงเวลาเอาไว้ค่าหนึ่งทั้งนี้เพราะ หัวเครื่องพิมพ์จะไม่เคลื่อนที่ออกไปทันทีหลังจากที่มันมาถึงตำแหน่งซ้ายสุด มันจะหยุดอยู่สักพักเราจึงต้องหน่วงเวลาการทำงานของเราเอาไว้ด้วย หลังจากนั้นจะกำหนดจำนวนครั้งของการสแกน (LOOPSCAN) เพื่อทำการสแกนและรับข้อมูลมาครั้งละ 1 จุด การสแกนครั้งละ 1 จุดนี้จะเรียกโปรแกรมย่อยชื่อ TRIGGER ซึ่งจะทำให้ได้ข้อมูลจากการสแกนเก็บไว้ในรีจีสเตอร์ BL ค่าที่ได้นี้จะเป็นตัวเลขค่าหนึ่ง โดยจะเป็นค่าเท่าใดนั้นขึ้นอยู่กับลักษณะของภาพในจุดนั้น ๆ จากนั้นจะนำค่าใน BL นี้มาเปรียบเทียบค่าโทนสีต่าง ๆ (TONE 1 – TONE 7) เพื่อพิจารณาว่าเป็นระดับความเข้มเท่าใด ในที่นี้กำหนดไว้ 8 ระดับ จาก 0 ถึง 7 เมื่อเปรียบเทียบเสร็จแล้วก็จะกำหนดระดับที่วัดได้ให้กับรีจีสเตอร์ AL ซึ่งค่าใน AL นี้จะมีค่าระหว่าง 0 ถึง 7 เท่านั้น จะเห็นว่าสามารถเก็บข้อมูลไว้โดยใช้หน่วยความจำ เพียงแค่ 3 บิต เท่านั้น ดังนั้นจึงเพิ่มขบวนการอีกนิดหน่อยเพื่อทำการเก็บข้อมูล 2 จุด ลงไว้ในไบต์เดียวกัน คือ จุดแรกไว้ที่นิบเบิ้ลบน (TYPE A) ส่วนจุดที่ 2 ที่นิบเบิ้ลล่าง (TYPE B) เมื่อสแกนครบ 2 จุดจะได้ข้อมูล 1 ไบต์ ซึ่งจะนำไปเก็บไว้ในส่วนของข้อมูลที่ชื่อว่า IMAGE ทำดังนี้เรื่อยไปจนกระทั่งครบ 1 บรรทัด

ก่อนจะกล่าวถึงการทำงานในส่วนโปรแกรมย่อย TRIGGER นั้น ขอกล่าวถึงในส่วนโมโนสเตเบิ้ลของพอร์ทจอยสติ๊กก่อน จากหัวสแกนที่วัดระดับสัญญาณของแสงอินฟาเรดนั้นจะแปลงเป็นกระแสไปชาร์จตัวเก็บประจุที่โมโนสเตเบิ้ล ทำให้ระดับแรงดันที่ตกคร่อมตัวเก็บประจุเพิ่มขึ้นเรื่อย ๆ ในขณะที่แรงดันเตรียมตัวเก็บประจุยังต่ำอยู่ เอาท์พุตของโมโนสเตเบิ้ลจะมีค่าเป็น "1" ซึ่งเราสามารถตรวจสอบได้จากการอ่านข้อมูลทางพอร์ทจอยสติ๊ก เมื่อแรงดันที่ตกคร่อมตัวเก็บประจุนั้นมีค่ามากกว่าค่า ๆ หนึ่ง จะทำให้เอาท์พุตของโมโนสเตเบิ้ลเปลี่ยนเป็น "0" ถ้ากระแสที่ไหลเข้าชาร์จตัวเก็บประจุมีค่ามากจะทำให้ แรงดันคร่อมตัวเก็บประจุสูงขึ้นอย่างรวดเร็ว นั่นหมายความว่าระยะเวลาที่เอาท์พุตของโมโนสเตเบิ้ลเป็น "1" จะน้อย จากที่กล่าวมานี้

ถ้า ตรวจสอบเวลาว่าเอาท์พุทของโมโนสเตเบิ้ลเป็น "1" นานมากน้อยแค่ไหน เราก็จะทราบระดับความเข้มของภาพได้ โปรแกรมย่อย TRIGGER ได้ทำงานในส่วนนี้ได้ โดยเริ่มแรกจะส่งคำสั่งไปที่พอร์ทจอยสติ๊กเพื่อทริกโมโนสเตเบิ้ลก่อน จากนั้น จะทำงานวนลูบเพื่อคอยตรวจสอบว่าเอาท์พุทของโมโนสเตเบิ้ลเป็น "0" หรือยัง โดยการตรวจสอบแต่ละครั้งจะเพิ่มค่ารีจีสเตอร์ BL ขึ้น 1 ทุกครั้ง เมื่อเอาท์พุทของโมโนสเตเบิ้ลเปลี่ยนเป็น "0" ก็จะออกจากลูปเป็นการจบการทำงานของโปรแกรมย่อยนี้ จะเห็นว่าถ้า BL มีค่ามากหมายความว่าเอาท์พุทของโมโนสเตเบิ้ลเป็น "1" นาน ดังนั้นค่าใน BL จะบ่งบอกถึงประมาณกระแสที่ทำการชาร์จตัวเก็บประจุ นั้นหมายถึงบ่งบอกถึงระดับความเข้มของภาพ ณ จุดนั้นได้ด้วย

ในส่วนของโปรแกรมย่อย ONE-LINE ยังมีโปรแกรมย่อยอีกโปรแกรมหนึ่งที่ถูกเรียกใช้ คือโปรแกรมย่อย DELAY โปรแกรมนี้จะทำหน้าที่ในการหน่วงเวลาการทำงานของโปรแกรม ทั้งนี้เพราะเวลาที่ใช้ในการรับสัญญาณภาพแล้วเปลี่ยนมาเป็นค่าโทนต่าง ๆ นั้นใช้เวลาน้อยกว่า เวลาเฉลี่ยในการทำงานแต่ละจุด เวลาเฉลี่ยในการทำงานแต่ละจุดหาได้จากเวลาในการเคลื่อนที่หารด้วยจำนวนจุด ซึ่งจากการทดลองจับเวลาในการเคลื่อนที่ของหัวเครื่องพิมพ์และคำนวณดูจะได้เวลาเฉลี่ยในการทำงานแต่ละจุดเป็น 2.75 ms ซึ่งมีค่ามากกว่าเวลาที่โปรแกรมย่อย TRIGGER ทำงานเสร็จร่วมทั้งการเปรียบเทียบโทนต่าง ๆ ด้วย จึงต้องมีโปรแกรมให้เครื่องรอให้ได้ครบเวลาเฉลี่ยนี้โดยการวนลูปที่ไม่มีการกระทำใด ๆ (NO OPERATION) แต่จากการทำงานของโปรแกรมย่อย TRIGGER จะเห็นว่าเวลาที่โปรแกรมย่อยนี้ใช้ไปจะไม่เท่ากันทุกครั้ง ขึ้นอยู่กับเวลาในการชาร์จตัวเก็บประจุว่าจะเร็วหรือช้าแค่ไหน ดังนั้น จึงต้องมีการคำนวณก่อน เพื่อหาจำนวนลูปในการหน่วงเวลา

ลักษณะการคำนวณการหาจำนวนลูปเพื่อหน่วงเวลาเป็นดังนี้ เริ่มแรกกำหนดค่าคงที่ขึ้นมาค่าหนึ่งก่อนโดยค่านี้หาได้จากเวลาเฉลี่ยที่ใช้ในการทำงานใน 1 จุด หารด้วย เวลา 1 คล็อกไซเคิ้ล ก็จะได้จำนวนคล็อกไซเคิ้ลในการทำงาน 1 จุด จากนั้นก็คำนวนว่าในการทำงานแต่ละจุดนั้นจากโปรแกรมจะต้องใช้คล็อกไซเคิลที่ลูก แล้วนำไปลบก็จะได้จำนวนคล็อกไซเคิ้ลที่เหลือ จากนั้น ก็นับจำนวนคล็อกที่ต้องใช้ในการวนลูป 1 ลูป แล้วนำไปหารคล็อกที่เหลือก็จะได้จำนวนลูป จากการนับคล็อกไซเคิ้ลของคำสั่งต่าง ๆ ทำให้ได้ผลลัพธ์ ดังนี้

$$\text{จำนวนลูปที่ต้องวน} = \frac{13117.5 - (329 + (AL \times 20) + (BL \times 42)}{}$$

โดยที่ 13117.5 คือจำนวนคล็อกไซเคิ้ลทั้งหมดที่ต้องทำในการทำงาน 1 จุด

([329+(ALx20)+(BLx42)) คือจำนวนคล็อกไซเคิ้ลทั้งหมดที่โปรแกรมใช้ไป

AL คือค่าในรีจีสเตอร์ AL ซึ่งเป็นค่าของโทนที่เปรียบเทียบแล้ว (1-7) ยิ่งระดับโทนมากขึ้นก็ต้องมีการเปรียบเทียบหลายครั้ง ทำให้ใช้เวลามากขึ้น โดยในการเปรียบเทียบนี้ใช้คล็อกไซเคิ้ลเท่ากับ (20xAL)+27

BL คือค่าในรีจีสเตอร์ BL ซึ่งเป็นค่าที่โปรแกรมย่อย TRIGGER นับได้ ยิ่ง BL มีค่ามากยิ่งใช้เวลามาก โดยการนับนี้ใช้คล็อกไซเคิ้ลเท่ากับ (42xBL)+28

17 คือจำนวนคล็อกไซเคิ้ลที่ใช้ในการวนลูปเพื่อหน่วงเวลา 1 รอบ

แต่เวลาที่ใช้ในการทำงานจริง ๆ มิได้เท่ากับจำนวนคล็อกไซเคิ้ล ดังนั้นจึงต้องทดลองเปลี่ยนค่าไปเพื่อให้ได้การหน่วงเวลาที่ถูกต้อง ซึ่งต้องเปลี่ยนจาก 12588-(ALx20)-(BLx42) เป็น 10288-(ALx22)-(BLx82) (และเป็น 22588-(ALx25)-(BLx170) สำหรับเครื่องเทอร์โบ)

4.2.3 <u>โปรแกรมย่อย PROCIMGAGE</u>

ในการสร้างภาพให้ได้โทนต่าง ๆ ถึง 8 โทนนั้น เราอาศัยหลักที่ว่า ความเข้มมากเกิดจากความหนาแน่นของจุดสีดำมีมาก ส่วนความหนาแน่นของจุดสีดำน้อยจะทำให้มีโทนสีอ่อน จากหลักการนี้เราจึงกำหนดรูปแบบ (PATTERN) ขึ้นมา 8 แบบ เพื่อแสดงโทนทั้ง 8 โดยอาศัยความหนาแน่นของจุด ดังนี้

PATT0 ---> 1111 1111

PATT1 ---> 0111 1111

PATT2 ---> 0111 0111

PATT3 ---> 0011 0011

PATT4 ---> 0101 0101

PATT5 ---> 0001 0001

PATT6 ---> 0000 0001

PATT7 ---> 0000 0000

เมื่อรูปแบบเหล่านี้ไปปรากฏบนจอภาพ PATT0 จะเป็นสีขาว และจะจางลงเรื่อย ๆ จนกระทั่งถึง PATT7 จะเป็นสีดำสนิท

หลังจากโปรแกรมย่อย ONE-LINE ทำงานเสร็จ จะมีข้อมูลของภาพใน 1 บรรทัด

เก็บไว้ ที่ส่วนข้อมูลชื่อ IMAGE โปรแกรมย่อย PROCIMAGE จะนำข้อมูลส่วนนี้มาทำการประมวลผล เพื่อส่งเป็นระดับโทนออกไปที่วีดีโอแรม โดยมีการทำงานดังนี้

ขั้นแรกลด LOOPSCAN ลงเหลือครึ่งหนึ่งเพื่อกำหนดจำนวนครั้งในการโหลดข้อมูลจาก IMAGE มาประมวลผล จากนั้นกำหนดแอดเดรสของข้อมูลที่ IMAGE ให้กับรีจีสเตอร์ SI ต่อจากนั้นจะต้องตรวจสอบว่า บรรทัดที่กำลังทำการประมวลผลอยู่นี้เป็นเส้นคู่หรือเส้นคี่ เพราะแอดเดรสของเส้นคู่และเส้นคี่ของวีดีโอแรมจะไม่เท่ากัน เมื่อตรวจสอบได้ก็จะโหลดค่าออฟเซ็ทแอดเดรสไว้ให้แก่รีจีสเตอร์ DI จากนั้นจะโหลดข้อมูลจาก IMAGE มา 1 ไบท์ซึ่งมีข้อมูล 2 จุดด้วยกัน จึงต้องทำการประมวล 2 ครั้ง การประมวลผลทำได้โดย นำข้อมูลนั้นไปเปรียบเทียบว่าควรจะเป็นความเข้มระดับไหน เพื่อที่จะเลือกรูปแบบต่าง ๆ และส่งไปให้วีดีโอแรมต่อไป แต่การสแกนใช้ความละเอียด 40 จุดต่อนิ้ว จะทำให้ได้ว่าข้อมูล 1 จุดจะปรากฏบนจอภาพ 2 จุด หรือเป็นข้อมูลในวีดีโอแรม 2 บิท เท่านั้น ดังนั้น เมื่อเปรียบเทียบข้อมูลรูปภาพเพื่อเลือกรูปแบบต่าง ๆ แล้ว 1 จุด โดยเก็บไว้ในรีจีสเตอร์ AH แล้ว จะต้องเลื่อนค่าในรีจีสเตอร์ AH เข้าไปเก็บไว้ในรีจีสเตอร์ BL 2 บิท เป็นข้อมูล 1 จุด จากนั้นก็ทำการหมุน (ROTAGE) รูปแบบทั้งแปด 2 บิท ด้วยเช่นกัน เพื่อให้การเลื่อนรูปแบบครั้งต่อไปไปไว้ในรีจีสเตอร์ BL จะได้เป็นรูปแบบที่ถูกต้อง เมื่อเลื่อนรูปแบบเข้าสู่รีจีสเตอร์ BL ครบ 8 บิท (4 ครั้ง) แล้ว ก็จะดึงค่าจากรีจีสเตอร์ BL ไปยังวีดีโอแรมทันทีและทำการประมวลผลข้อมูลรูปภาพอีกต่อไป ทำเช่นนี้จะครบข้อมูลใน 1 บรรทัด

เมื่อประมวลผลข้อมูลเสร็จ 1 บรรทัดแล้วจะต้องทำการหมุนรูปแบบเหล่านี้อีกครั้ง ซึ่งจำนวนครั้งที่หมุนก็แล้วแต่รูปแบบ ทั้งนี้เพื่อในการนำรูปแบบออกแสดงที่จอภาพในบรรทัดต่อไปนั้นบิทที่เป็น "0" และ "1" จะได้ไม่ตรงกัน ทำให้เกินเป็นลายสลับ ซึ่งจะทำให้เกินเป็นระดับโทนอ่อนแก่ได้ ถ้าไม่มีการหมุนรูปแบบแล้ว จะทำให้ส่วนที่เป็นบิท "0" และ "1" อยู่ตรงกันจะเกิดเป็นเส้นตรงขึ้น ซึ่งจะไม่เกิดเป็นโทนที่จอภาพ

การพัฒนาเครื่องสแกนภาพโดยใช้เครื่องพิมพ์โครงงานนี้ ได้กระทำขึ้นจนสามารถพัฒนาหัวสแกนได้ถึงสองรุ่นด้วยกัน ซึ่งแต่ละรุ่นก็มีทั้งข้อดีและข้อเสียแตกต่างออกไปดังที่ได้กล่าวมาบ้างแล้ว และผู้จัดทำได้ทดลองสแกนภาพโดยใช้หัวสแกนทั้งสองชนิด พบว่าภาพที่ได้จากการใช้หัวสแกนรุ่นแรกจะสามารถให้ความคมชัดได้ดีกว่า เนื่องจากมีระดับความเข้มน้อยกว่า ทำให้สามารถแบ่งแยกระดับความเข้มได้อย่างชัดเจน และจากทดลองใช้กระดาษหลายๆชนิด จะเห็นว่า ภาพที่ได้จากการใช้กระดาษมันจะให้ผลดีที่สุด เนื่องจากแสงอินฟราเรดที่ส่งออกมาจะสะท้อนได้มาก ทำให้แรงดันที่ได้จากสีขาวและสีดำมีความแตกต่างสูง ถ้าใช้กระดาษด้านจะพบว่าความแตกต่างของแรงดันระหว่างสีดำและสีขาวจะน้อยมากจนยากแก่การตรวจวัด ทำให้ภาพที่ได้จากกระดาษด้านไม่คมชัดเท่าที่ควร แต่อย่างไรก็ตาม เราสามารถปรับปรุงแก้ไขได้โดยการปรับระดับแรงดันอ้างอิงที่ภาคคอมพาราเตอร์ในส่วนฮาร์ดแวร์เสียใหม่ ซึ่งจะให้ภาพที่คมชัดขึ้น

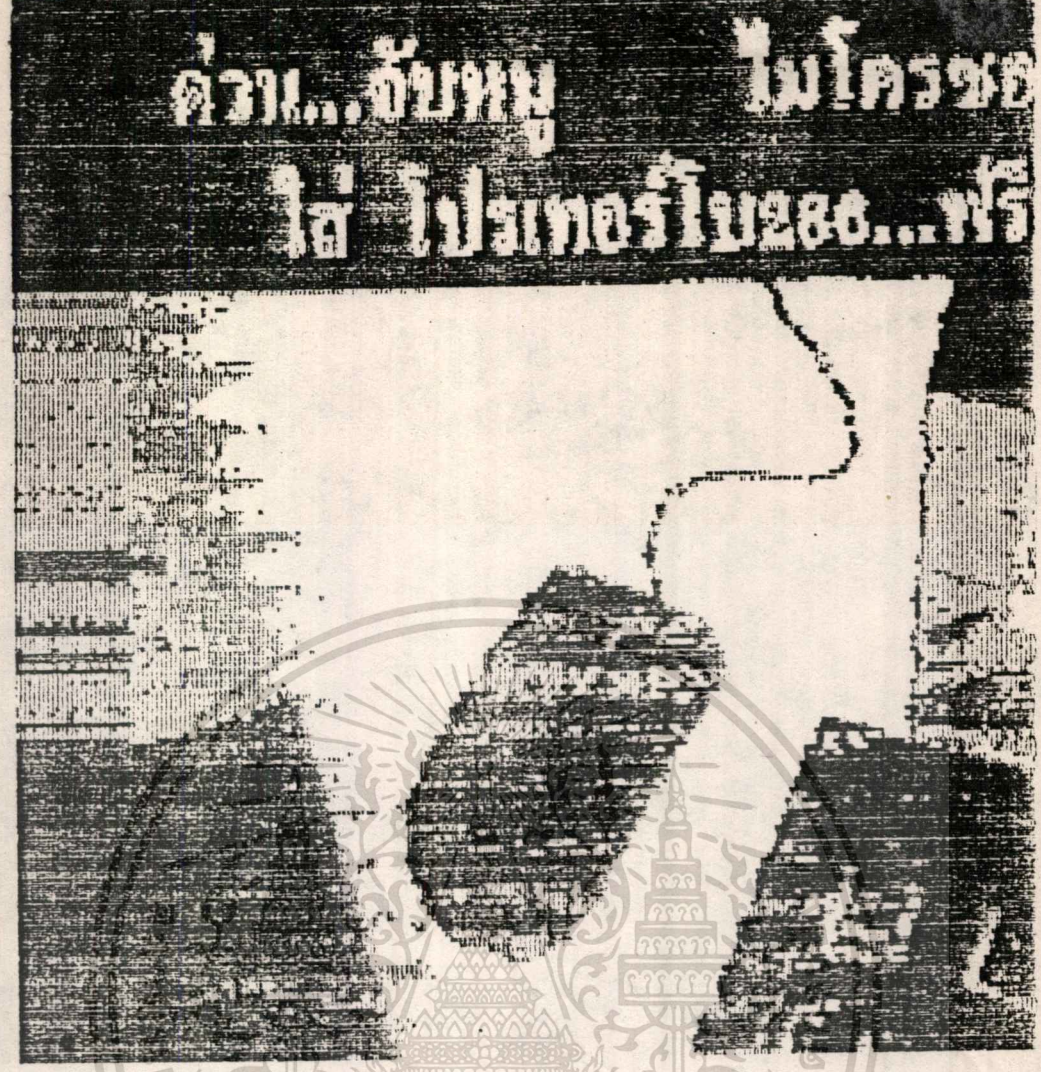
ส่วนการสแกนภาพโดยใช้หัวสแกนรุ่นที่สองนั้น มีข้อดีที่ผู้ใช้สามารถประหยัดค่าใช้จ่ายลงไปได้มาก โดยเน้นการใช้งานวงจรที่มีอยู่ภายในเครื่องไมโครคอมพิวเตอร์แทน ทำให้การควบคุมชิ้นงานทั้งระบบง่ายขึ้น และได้ปรับปรุงให้ภาพที่ได้มีระดับความเข้มถึง 8 ระดับ ทำให้สามารถแสดงรายละเอียดของสีต่าง ๆ ได้ดีกว่าภาพที่ใช้หัวสแกนรุ่นแรก ซึ่งมีระดับความเข้มเพียง 3 ระดับเท่านั้น แต่อย่างไรก็ตาม จากการทดลองพบว่ายังมีข้อที่ยังต้องปรับปรุงคือ การที่ระดับโทนสีมากขึ้นนั้นจะทำให้การสแกนตัวอักษรขนาดเล็ก ๆ ไม่ชัดเจนนัก เนื่องจากจำนวนบิทจากแพทเทอร์นสีที่ถูกเลื่อนออกไปแสดงผล (สำหรับโทนสีนั้น ๆ) อาจสั้นเกินไปจนไม่สามารถแสดงตัวอักษรแคบ ๆ ให้ชัดเจนได้ ตัวอย่างเช่น กรณีที่หัวสแกนเปลี่ยนจากการอ่านสีขาวเป็นสีเทาที่มีขนาดแคบ ๆ แล้วไปอ่านสีขาวอีก จะเห็นว่า ถ้าหากแพทเทอร์นของสีเทานั้นเป็น 00001111 บิทที่ถูกเลื่อนออกมาคือ 00 แล้วจะตามด้วย 00, 11 และ 11 ตามลำดับ ซึ่งหากส่วนของสีเทานั้นแคบมาก สีที่เราจะได้จึงมีเพียงสีขาวเท่านั้น ซึ่งทำให้ภาพหรือขอบบางส่วนเลือนหายไป นอกจากนี้ จากการทดลองพบว่า หากต้องการแสดงภาพให้เห็นโทนครบทั้ง 8 ระดับ กระดาษที่ใช้สำหรับเป็นภาพต้นแบบควรเป็นกระดาษปอนด์ด้าน ส่วนการสแกนภาพบนกระดาษมันนั้นจะให้ภาพที่ไม่สู้ดีนัก เนื่องจากระดับโทนที่ได้จริงๆนั้นจะน้อยกว่าที่กำหนดไว้ สาเหตุเกิดจากแสงที่สะท้อนจากกระดาษมันจะมากเกินไป ทำให้กระแสที่ได้มีค่าสูงมาก และเป็นผลให้การชาร์จตัวเก็บประจุที่ต่อกับวงจรโมโนสเตเบิ้ลใช้เวลาน้อยมาก นั่นคือ พัลส์ที่ได้จากเอาท์พุท

ของวงจรดังกล่าวจะสั้น ซึ่งเร็วเกินกว่าที่ซอฟท์แวร์จะสามารถแยกแยะความแตกต่างของช่วงเวลาหน่วงให้ละเอียดได้ เป็นผลให้ระดับความเข้มถูกลดลงโดยปริยาย

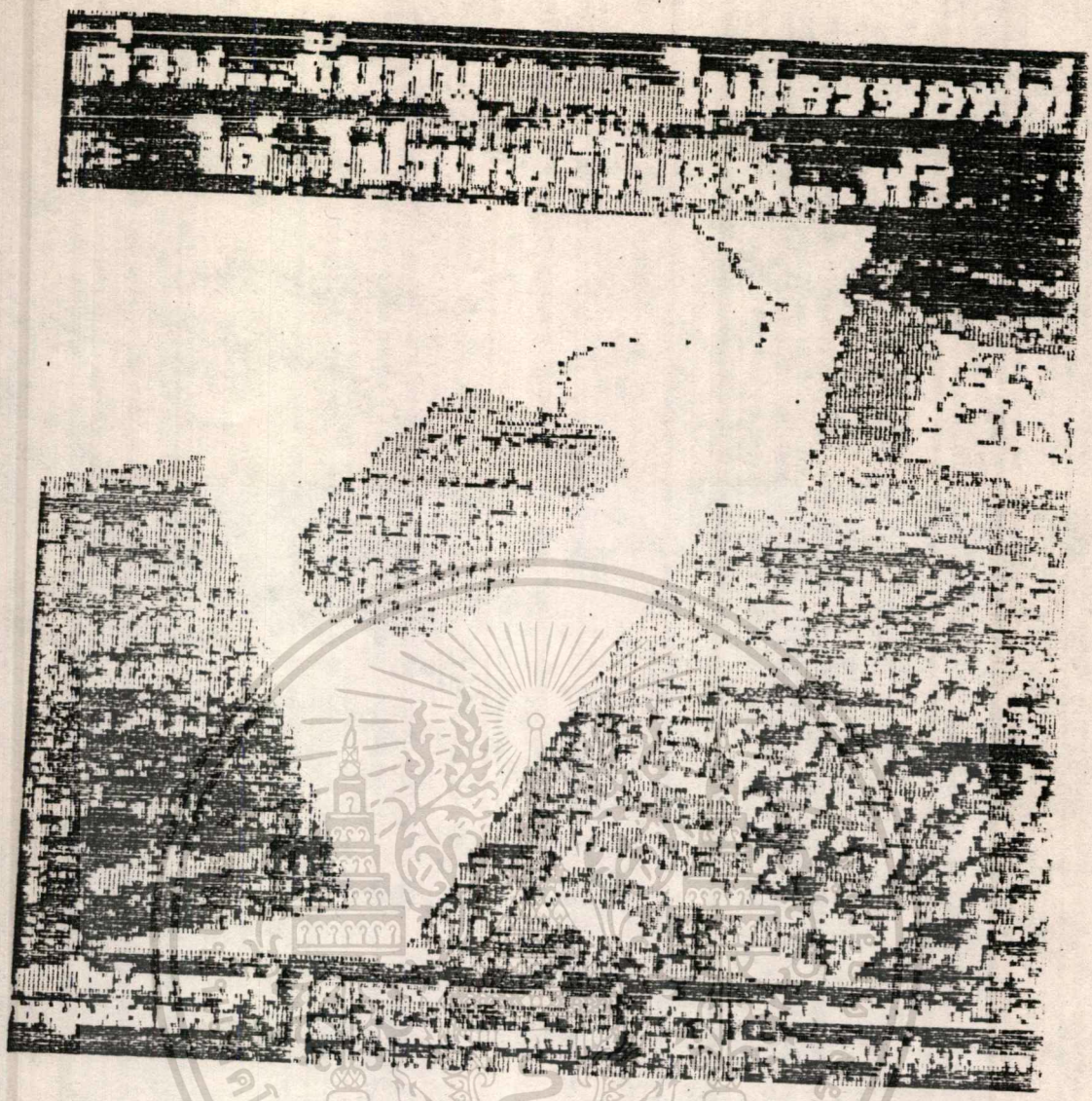
โดยสรุปแล้ว ภาพที่สแกนได้จากหัวสแกนทั้งสองรุ่นนั้นต่างก็ให้ผลเป็นที่น่าพอใจ คือสามารถแสดงโครงร่างของภาพได้ใกล้เคียงภาพจริงได้ดีพอสมควร และสามารถอ่านตัวหนังสือขนาดประมาณ 0.8 เซนติเมตรได้ แต่ภาพนั้นอาจผิดสัดส่วนไปบ้างเล็กน้อย ซึ่งสามารถแก้โดยการปรับจำนวนรอบสแกนที่เหมาะสมที่สุด เช่นถ้าปรับให้จำนวนรอบมากขึ้น ก็จะได้ภาพที่กว้างขึ้นและภาพจะแคบลงถ้าลดจำนวนรอบลง ส่วนการปรับระยะ LINEFEED จะมีผลต่อความยาวของภาพ ถ้าระยะ LINEFEED มากภาพก็จะเตี้ย และถ้าปรับให้มีค่าน้อยลง ภาพก็จะดูยาวขึ้น นอกจากนี้ ยังพบว่าบางส่วนของภาพก็มีสัญญาณรบกวนเข้ามาทั้งที่ไม่มีภาพอะไรเลย ทำให้ภาพเปื้อน แต่ในบางส่วนของภาพก็จะหายไปเลย ซึ่งสาเหตุเกิดจากสีบางสีไม่อาจตรวจสอบโดยใช้แสงอินฟราเรดได้ เช่น สีแดงและสีเขียว เป็นต้น ดังนั้น ภาพส่วนนั้นจึงไม่ปรากฏเลย ซึ่งเป็นข้อเสียที่ไม่สามารถแก้ไขได้นอกจากการเปลี่ยนชนิดของแสงที่ใช้ในการเชื่อมโยงสัญญาณ

ตัวอย่างการสแกนภาพจากรุ่นที่ 1 แสดงถึงผลจากการที่ไม่ได้หน่วงเวลาก่อนการสแกน ทำให้เกิดเป็นแถบขึ้น เปรียบเทียบกับภาพที่ได้หน่วงเวลาไว้แล้ว

ตัวอย่างการสแกนภาพจากรุ่นที่ 2 แสดงถึงผลจากการที่ตั้งค่าคงที่ของการหน่วงเวลาที่ไม่เหมาะสม ทำให้รูปที่ได้เบี้ยวทางด้านขวาของภาพ เนื่องจากว่าเวลาหน่วงของจุดสีดำและขาวไม่เท่ากัน จึงต้องเปลี่ยนให้ได้ค่าที่เหมาะสมโดยทดลองไปเรื่อย ๆ

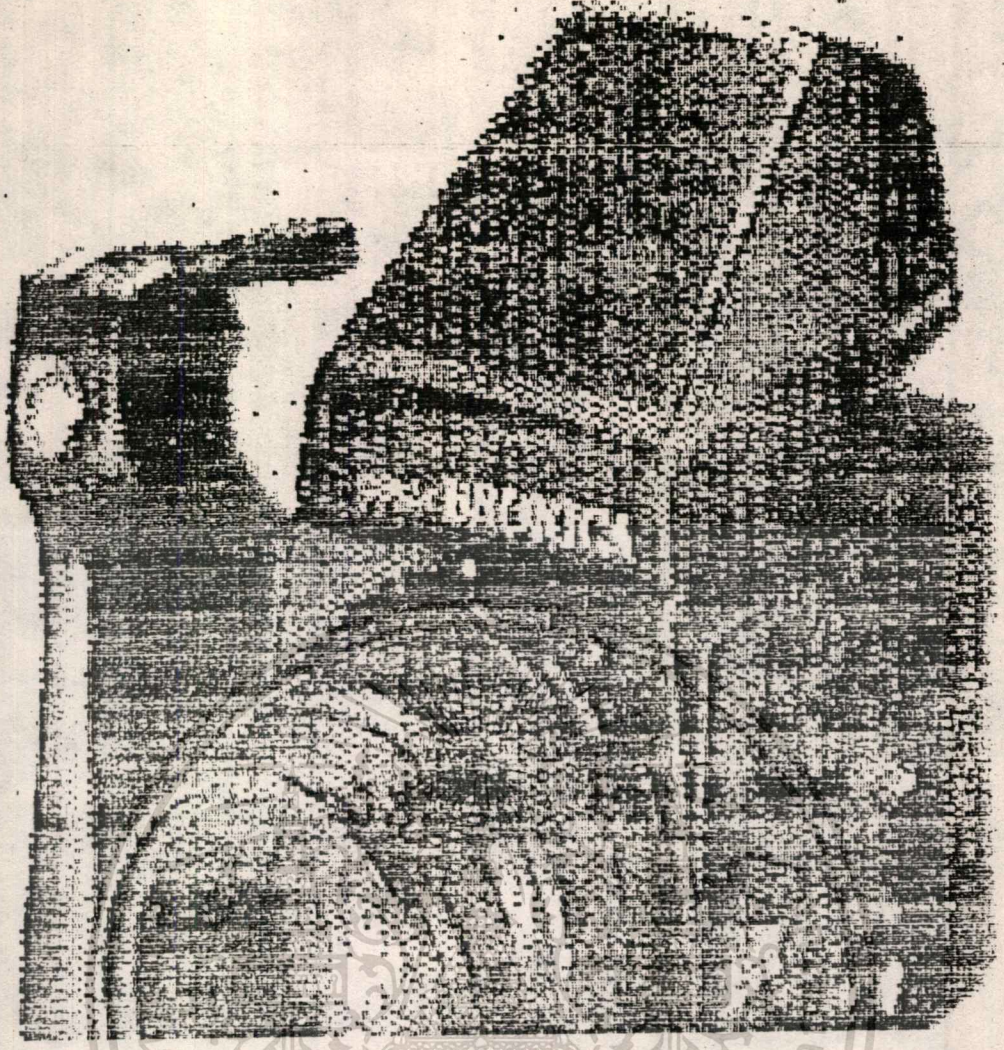


รูป 5.1 แสดงการสแกนโดยไม่มีการหน่วงเวลา

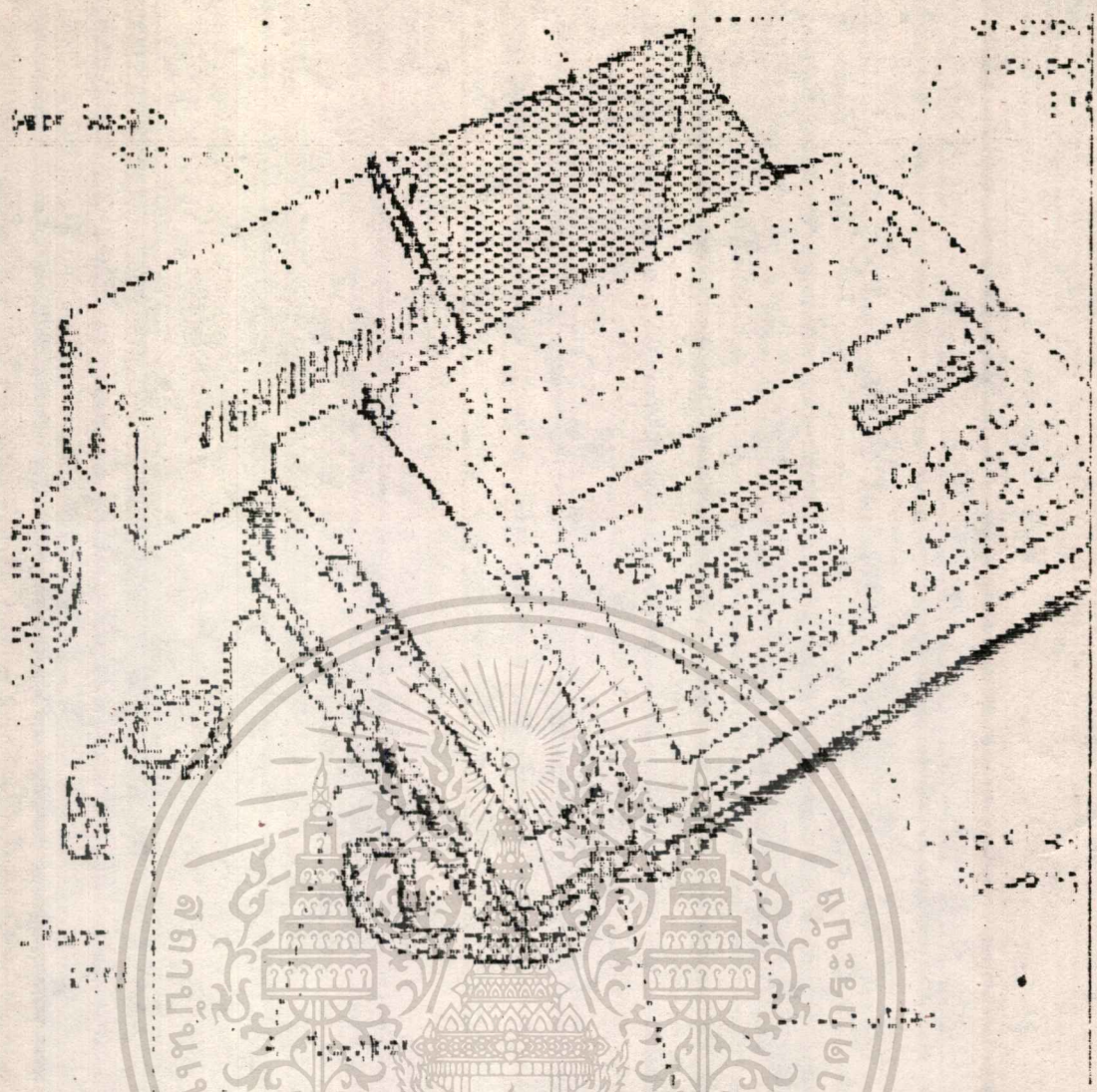


รูป 5.2 แสดงการสแกนโดยมีการหน่วงเวลาที่เหมาะสม ในหัวสแกนรุ่นที่ 1

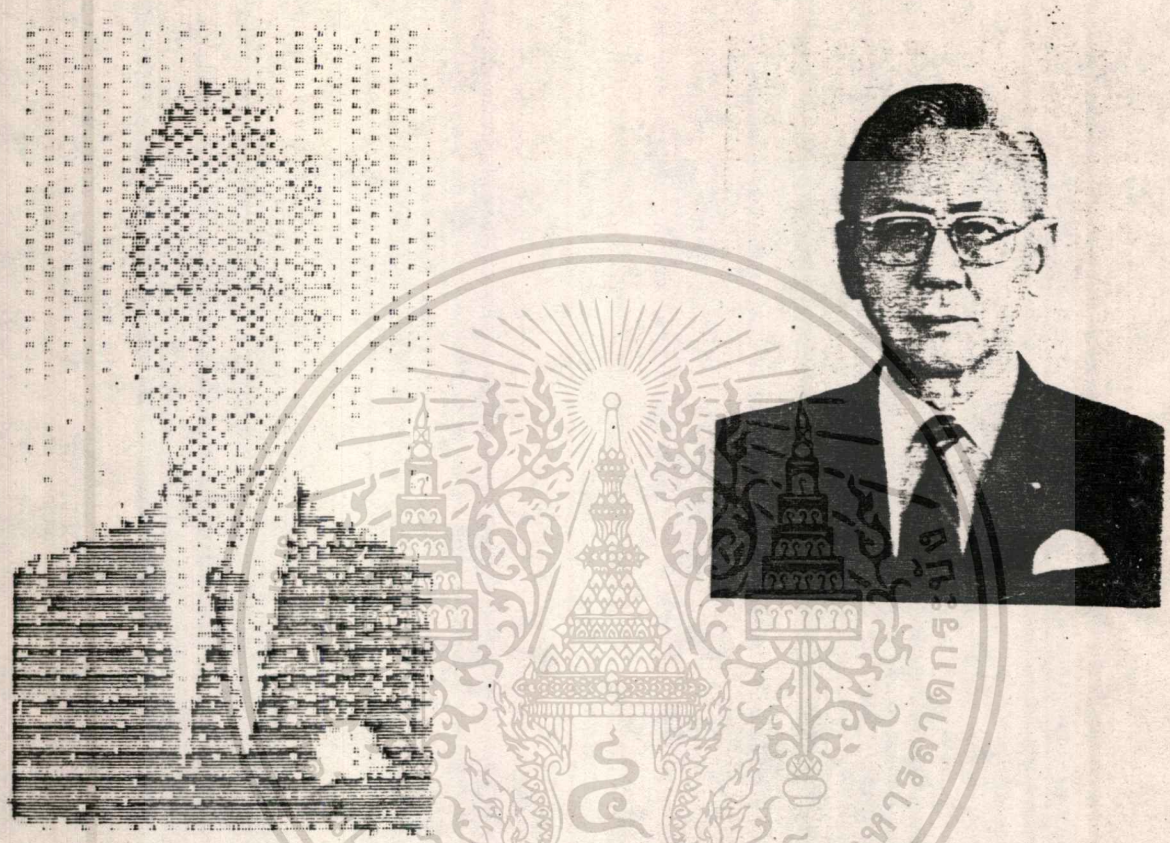
รูป 5.3 แสดงการตั้งค่าคงที่ตามการคำนวณ



รูป 5.4 แสดงการตั้งค่าคงที่ของการหน่วงเวลาที่เหมาะสม



รูป 5.5 แสดงการตั้งค่าคงที่ของการหน่วงเวลาที่เหมาะสม



รูป 5.6 เปรียบเทียบภาพจากการสแกนโดยหัวสแกนรุ่นที่ 2 กับภาพจริง

บทที่ 6 สรุปผลการทดลองและข้อเสนอแนะ

ผลงานที่ได้จากการพัฒนาเครื่องสแกนภาพโดยใช้เครื่องพิมพ์นี้เป็นที่น่าพอใจมาก เนื่องจากสามารถนำชิ้นงานมาใช้ได้อย่างมีประสิทธิภาพและใกล้เคียงกับเป้าหมายและคุณสมบัติที่ตั้งไว้ คือ ความละเอียดของภาพที่สูง 40 จุดต่อนิ้วทั้งในแนวตั้งและแนวนอน ระดับความเข้มสีที่ถูกพัฒนาจาก 3 ระดับเป็น 8 ระดับ ความสะดวกในการนำไปใช้งานที่ได้พัฒนาขึ้นเมื่อใช้หัวอ่านรุ่นที่สอง และที่สำคัญคือ ประสิทธิภาพของชิ้นงานมิได้ลดต่ำลงตามราคาของหัวสแกนรุ่นที่ได้พัฒนาขึ้นใหม่เลย

อย่างไรก็ตาม ชิ้นงานที่ถูกพัฒนาขึ้นนี้ก็มิได้สมบูรณ์ทุกประการ ซึ่งมีข้อที่สามารถปรับปรุงได้อีกบางประการ คือ

1. หัวสแกนรุ่นที่หนึ่งซึ่งจะให้ภาพที่มีระดับความเข้มเพียง 3 ระดับอาจปรับปรุงให้มีระดับสูงขึ้นได้โดยเพิ่มส่วนทางฮาร์ดแวร์ แต่หากใช้งานในลักษณะนั้นความคมชัดของตัวอักษรอาจลดลงดังที่เป็นปัญหาที่พบในหัวสแกนรุ่นที่สอง ฉะนั้นจึงเป็นส่วนของผู้ใช้เองที่จะต้องตัดสินใจเลือกระหว่างจำนวนโทนสีหรือความคมชัดของภาพ ถึงกระนั้นก็ดี หากมีการพัฒนาซอฟท์แวร์ให้ดีขึ้น ก็จะสามารถลดปัญหานี้ลงได้พอสมควร
2. สามารถปรับปรุงภาพให้ละเอียดกว่านี้ได้ ทั้งนี้เพราะในการใช้เครื่องพิมพ์ร่วมกับหัวอ่านในชิ้นงานนี้ ขีดความสามารถของเครื่องพิมพ์ยังไม่ถูกนำมาใช้อย่างสูงสุด ซึ่งถ้าสามารถปรับปรุงหัวสแกนให้ละเอียดขึ้น (อาจใช้หัวอ่านชนิด BAR CODE) ก็จะสามารถได้ภาพที่ละเอียดกว่าเดิม ส่วนในแนวตั้งสามารถตั้ง LINE FEED ให้ละเอียดถึง 1/216 นิ้ว(ซึ่งในโครงงานนี้ใช้เพียง 3/72 นิ้วเท่านั้น) และหากใช้งานในลักษณะที่กล่าวมานี้ ควรปรับปรุงโปรแกรมให้ใช้กับจอที่มีความละเอียดสูงเช่น EGA เป็นต้น
3. หัวสแกนที่ใช้ในโครงงานนี้ประกอบด้วยตัวออฟโตคับเปิ้ล ซึ่งมีคุณสมบัติเฉพาะตัว จากการทดลองพบว่า ออฟโตคับเปิ้ลนี้ไม่สามารถตรวจจับภาพที่มีสีแดง ส้ม หรือแม้กระทั่งสีอ่อน ๆ ได้ดีนัก อันเป็นปัญหาที่เกิดขึ้นเมื่อใช้ตัวส่งเป็นชนิดอินฟราเรด ซึ่งการแก้ไขสำหรับข้อบกพร่องนี้อาจทำได้โดยการใช้หัวอ่านชนิดอื่นที่มีคุณภาพสูงขึ้นเท่านั้น
4. การเก็บข้อมูลลงดิสค์ที่ใช้ในโครงงานนี้เป็นการนำข้อมูลทั้งหมดบนจอเก็บลงดิสค์ ซึ่งในบางครั้งอาจทำให้สิ้นเปลืองเนื้อที่มากเกินความจำเป็น ดังนั้น อาจใช้เทคนิคการเก็บข้อมูลชนิดอื่น เช่นการบีบข้อมูล เป็นต้น และควรใช้ฟอร์แมทรูปแบบการเก็บข้อมูล

ให้เป็นไฟล์มาตรฐานกราฟฟิค เช่น TIFF เพื่อที่จะทำให้ข้อมูลที่เก็บได้นั้นสามารถถูกนำไปใช้ร่วมกับโปรแกรมกราฟฟิคอื่น ๆ ได้

5. ซอฟท์แวร์สำหรับชิ้นงานในโครงงานนี้ถูกใช้ในการควบคุมการรับข้อมูล การแบ่งระดับความเข้มของจอภาพ และการแสดงผลบนจอมอนิเตอร์เท่านั้น ซึ่งเราอาจปรับปรุงซอฟท์แวร์เพื่อเพิ่มคุณภาพของภาพได้ โดยการใช้คณิตศาสตร์วิเคราะห์ทำให้ได้ภาพที่คมชัดยิ่งขึ้น หรืออาจปรับปรุงโดยใช้เทคนิค IMAGE PROCESSING เป็นต้น

6. ซอฟท์แวร์ที่ใช้ร่วมกับเครื่องสแกนที่หัวอ่านรุ่นที่สองนั้นยังไม่สามารถทำงานได้ดีมากนัก เช่น ขอบรูปหรือตัวอักษรอาจไม่ตรงและคมเท่าที่ควร ปัญหาดังกล่าวเกิดขึ้นจากการใช้คำสั่งบางตัวที่มีคล็อกไซเคิ้ลที่ไม่แน่นอน เช่น คำสั่ง MUL และ DIV เป็นต้น เป็นผลให้เวลาห่วงระหว่างข้อมูลภาพแต่ละจุดไม่คงที่ ซึ่งสามารถแก้ไขได้โดยการปรับปรุงซอฟท์แวร์โดยหลีกเลี่ยงการใช้คำสั่งดังกล่าว ซึ่งกลุ่มผู้จัดทำได้พยายามพัฒนาและนำไปใช้ร่วมกับหัวสแกนรุ่นที่ 3 แล้ว

7. เทคนิคการเลื่อนแพทเทอร์นเข้าไปเก็บในวีดีโอแรม สามารถรองรับหัวอ่านที่มีความละเอียดสูงได้โดยการปรับปรุงซอฟท์แวร์อีกเล็กน้อย แต่ถ้าต้องการเพิ่มความละเอียดในการอ่านเป็น 80 จุดต่อนิ้ว จะทำได้ง่ายมากคือลดค่าคงที่ของการหน่วงเวลาลงและทำการเลื่อนข้อมูลครั้งละ 1 บิทต่อจุดเท่านั้น

ส่วนโปรแกรมภาษาเบสิกในเครื่องสแกนรุ่นที่ 1

```
'PROGRAM MAIN FOR CALL ASSEMBLY  PROGRAM
DEFINT  A-Z
    DIM IMAGE(1500)
    INPUT  "SCREEN 1 OR 2 : ",SRN
    SCREEN SRN
    X2 = (320*SRN)-1 : X1 = X2-88
    COL = 70
    IF SRN = 1 THEN COL =30
    KEY(1) ON
    ON KEY(1) GOSUB BACKIMAGE
    GET (X1,0)-(X2,100),IMAGE
MENU:
    LOCATE 1,COL:  PRINT "          "
    LOCATE 2,COL:  PRINT "[1] SAVE  "
    LOCATE 3,COL:  PRINT "          "
    LOCATE 4,COL:  PRINT "[2] LOAD  "
    LOCATE 5,COL:  PRINT "          "
    LOCATE 6,COL:  PRINT "[3] SCAN  "
    LOCATE 7,COL:  PRINT "          "
    LOCATE 8,COL:  PRINT "[4] EXIT  "
    LOCATE 9,COL:  PRINT "          "
    LOCATE 10,COL: PRINT "          "
    LOCATE 11,COL: PRINT "          "
    LOCATE 12,COL: PRINT "          "
CHK:   SEL$ = INKEY$
       IF SEL$ = "" THEN GOTO CHK
       IF SEL$ = "1" THEN GOTO SAVEPROG
       IF SEL$ = "2" THEN GOTO LOADPROG
       IF SEL$ = "3" THEN GOTO SCAN
       IF SEL$ = "4" THEN GOTO INIT
GOTO MENU
'-------------------------------------------------------------
SAVEPROG :
    LOCATE 10,COL+1
    PRINT "SAVE.."
    LOCATE 11,COL+1
    INPUT "", NAM$
    NAM$ = NAM$ + ".PIC"
    GOSUB BACKIMAGE
    DEF SEG = &HB800
    BSAVE NAM$,0,&H4000
    DEF SEG
GOTO MENU
'-------------------------------------------------------------
```

```
LOADPROG :
    LOCATE 10,COL+1
    PRINT "LOAD.."
    LOCATE 11,COL+1
    INPUT "",LNAM$
    LOCATE 10,COL+1
    PRINT "          "
    LOCATE 11,COL+1
    PRINT "          "
    LNAM$ = LNAM$+ ".PIC"
    DEF SEG =&HB800
    BLOAD LNAM$,0
    DEF SEG
    GET (X1,0)-(X2,100),IMAGE
GOTO MENU
'-------------------------------------------------------------
SCAN :
     cls 0
     INPUT "LENGTH FOR VERTICAL SCAN : ",VER!
     PRINT
     INPUT "LENGTH FOR HORIZONTAL SCAN : ",HOR!
     PRINT
     INPUT "LEFT MARGIN (IN INCHES) : ",INCH!
     PRINT
     INPUT "1. XT  ;  2.AT    " ;KIND
' CHECK FOR CORRECT CONDITION
     IF KIND = 1 THEN DLY = 648  ELSE  DLY = 1000
     IF INCH! = 0 THEN INCH! = 0.1
     IF VER! > 5 THEN VER! = 5
     LONG! = HOR! + INCH!
     IF LONG! > 6.5 THEN HOR! = 6.5  AND INCH! = 0.1
     NOL =  INT(VER!*40)
     NOC =  INT(60*HOR!)
     MARGIN = INT(INCH!*10)
     LOP =  INT(HOR!*10)
     CLS 0
     CALL BEGIN (MARGIN,DLY,NOC,NOL,LOP)
     GET (X1,0)-(X2,100),IMAGE
GOTO MENU
'-------------------------------------------------------------
BACKIMAGE :
     PUT (X1,0),IMAGE,PSET
RETURN
INIT :
     LPRINT CHR$(27)+"@"
END
```

```
                page    60,132
TITLE           ASSEMBLY PROGRAM FOR RECEIVE DATA FROM SCANNER
STACKSG         SEGMENT  PARA STACK 'STACK'
                DW       64 DUP(?)
STACKSG         ENDS
;-------------------------------------------------------------------
DATA            SEGMENT  PARA  PUBLIC 'DATA'
INIT            DB       27,'@'              ; INITIAL
                DB       27,'A',3            ; LINE FEED N/72
                DB       27,'l'              ; LEFT MARGIN
TAB             DW       ?
SETGRA          DB       27                  ; 'Esc'
                DB       75                  ; 'K'
NOCOLUM         DW       ?                   ; {(n2 x 256) + n1} = WIDTHx12
NOLINE          DW       ?                   ; NO. OF LINE SCANE (40xLONG-1
LOOPSCAN        DW       ?                   ; (WIDTH IN INCH)x10
DLY             DW       ?                   ; DELAY TIME
WSEGMENT        DW       0B800H              ; START VIDEO SEGMENT
OFFSETODD       DW       0000H               ; OFFSET ODD LINES
OFFSETEVEN      DW       2000H               ; OFFSET EVEN LINES
TEMP            DW       ?                   ; TEMPORARY VARIABLE
CHK             DB       0                   ; CHECK EVEN OR ODD
DATA            ENDS
DGROUP          GROUP    DATA
;-------------------------------------------------------------------
CODESG          SEGMENT  PARA PUBLIC 'CODE'
                PUBLIC   BEGIN
BEGIN     PROC   FAR
                ASSUME   CS:CODESG,DS:DGROUP,SS:STACKSG
; START RECEIVE VALUE FROM BASIC -------
                PUSH     BP
                MOV      BP,SP
                MOV      BX,[BP]+6          ;RECEIVE LOOPSCAN
                MOV      DX,[BX]
                MOV      LOOPSCAN,DX
                MOV      BX,[BP]+8          ;RECEIVE NOLINE
                MOV      DX,[BX]
                MOV      NOLINE,DX
                MOV      BX,[BP]+10         ;RECEIVE NOCOLUM
                MOV      DX,[BX]
                MOV      NOCOLUM,DX
                MOV      BX,[BP]+12         ;RECEIVE DELAY
                MOV      DX,[BX]
                MOV      DLY,DX
                MOV      BX,[BP]+14         ;RECEIVE TAB
                MOV      DX,[BX]
                MOV      TAB,DX
```

```
; FINISH RECEIVE VALUE ----------------------
;************************************************************
;RESET PRINTER -------------------------------
                    MOV     CX,8
                    LEA     SI,INIT
RESET:
                    MOV     AH,05           ;SET MODE OUT TO PRINTER
                    MOV     DL,[SI]         ;SELECT CHARATER
                    INT     21H             ;CALL DOS
                    INC     SI              ;NEXT CHARACTER
                    LOOP    RESET
;RESET COMPLETE ------------------------------
GETSTATUS1:         MOV     AH,02           ;READ STATUS OF PRINTER
                    MOV     DX,00
                    INT     17H
                    AND     AH,10000000B
                    JZ      GETSTATUS1
;************************************************************
; START SCAN AND RECEIVE DATA -----------------
                    MOV     BX,NOLINE       ;SET LOOP OF NO. OF LINE
LINELOOP:           MOV     DI,OFSETODD
                    XOR     CHK,01          ;CHECK FOR ODD OR EVEN
                    JNZ     START
                    MOV     DI,OFSETEVEN
      START:        MOV     TEMP,DI         ;SAVE DI TEMPORARY
                    CALL    CONTROL_HEAD
      GETSTATUS:    MOV     AH,02           ;READ STATUS OF PRINTER
                    MOV     DX,00
                    INT     17H             ;WHETHER HEAD REACH THE RIGHT
                    AND     AH,10000000B    ;MARGIN
                    JZ      GETSTATUS       ;NO:WAIT FOR IT
                    CALL    SCAN_1_LINE
                    MOV     DI,TEMP         ;RESTORE VALUE OF DI
                    ADD     DI,80           ;ADD FOR 1 LINE
                    MOV     AL,CHK          ;CHECK FOR ODD OR EVEN
                    AND     AL;01
                    JNZ     ODD
                    MOV     OFSETEVEN,DI
                    JMP     FINISH
      ODD:          MOV     OFSETODD,DI
      FINISH:       DEC     BX
                    JNZ     LINELOOP
;COMPLETE SCAN, RECIEVE DATA AND SAVE IN RAM
;************************************************************
                    MOV     OFSETODD,0000H  ;RESTORE OFFSETODD
                    MOV     OFSETEVEN,2000H ;RESTORE OFFSETEVEN
                    POP     BP
                    RET 2                   ;RETURN TO DOS
    BEGIN           ENDP
                    START SET HEAD PRINTER MOVING
                    ------------------------------
```

```
CONTROL_HEAD PROC NEAR
             MOV     CX,4
             LEA     SI,SETGRA
SCANLINE:
             MOV     AH,05          ;SET MODE OUT TO PRINTER
             MOV     DL,[SI]        ;SELECT CHARATER
             INT     21H            ;CALL DOS
             INC     SI             ;NEXT CHARACTER
             LOOP    SCANLINE
;SEND BLANK TO PRINTER
             MOV     CX,NOCOLUM
BLANK:       MOV     AH,05
             MOV     DL,00
             INT     21H
             LOOP    BLANK
;SEND "LINEFEED" CHANGE TO 1/40" LINEFEED
             MOV     AH,05          ;SET MODE OUT TO PRINTER
             MOV     DL,0AH         ;'LINEFEED'
             INT     21H            ;CALL DOS
             RET                    ;RETURN TO BEGIN
CONTROL_HEAD ENDP
;*************************************************************


;       STRAT SCAN TO GET INPUT AND SAVE IN RAM ONLY 1 LINE
;       ---------------------------------------------------
SCAN_1_LINE  PROC    NEAR
; DELAY FOR PRINTER HEAD ----------------
             MOV     CX,0F818H      ;
DLAY1:       LOOP    DLAY1          ;
; END DELAY LOOP ------------------------
             PUSH    BX
             MOV     BX,DLY
             MOV     CX,LOOPSCAN
             PUSH    DS             ; CHANGE DATA SEG.
             MOV     AX,VSEGMENT    ; TO VIDEO RAM
             MOV     DS,AX          ;
             MOV     DX,201H        ;SELECT JOYSTICK
GET1BITE:
             PUSH    CX
             SUB     AH,AH          ;CLEAR AH
             IN      AL,DX          ;INPUT PORT JOYSTICK
;DELAY
             MOV     CX,BX          ;set delaytime
             ADD     CX,5
delay1:      nop
             loop    delay1
             AND     AL,11000000B   ;SELECT TWO FIRST BIT
             OR      AH,AL          ;STORE IN AH
             IN      AL,DX          ;INPUT PORT JOYSTICK
;DELAY
```

```
                MOV     CX,BX           ;set delaytime
                ADD     CX,4
delay2:         nop
                loop    delay2
                AND     AL,11000000B    ;SELECT TWO FIRST BIT
                MOV     CL,02
                SHR     AL,CL           ;SET IN CORRECT BIT POSITION
                OR      AH,AL           ;SAVE IN AH
                IN      AL,DX           ;INPUT PORT JOYSTICK
;DELAY
                MOV     CX,BX           ;set delaytime
                ADD     CX,3
delay3:         nop
                loop    delay3
                AND     AL,11000000B    ;SELECT TWO FIRST BIT
                MOV     CL,04
                SHR     AL,CL           ;SET IN CORRECT BIT POSITION
                OR      AH,AL           ;SAVE IN AH
                IN      AL,DX           ;INPUT PORT JOYSTICK
;DELAY
                MOV     CX,BX           ;set delaytime
delay4:         nop
                loop    delay4
                AND     AL,11000000B    ;SELECT TWO FIRST BIT
                MOV     CL,06
                SHR     AL,CL           ;SET IN CORRECT BIT POSITION
                OR      AH,AL           ;SAVE IN AH
;COMPLET GETING DATA 1 BYTE
                MOV     [DI],AH         ;SAVE TO STACK
                INC     DI
                POP     CX
                LOOP    GET1BITE
                POP     DS
                POP     BX
                RET                     ;RETURN TO BEGIN
SCAN_1_LINE     ENDP
;*****************************************************************
CODESG          ENDS
                END     BEGIN
```

```
        PAGE 60,132
SCANNER-- RESET THE PRINTER'S HEAD THEN CONTROL IT TO MOVE.
          SEND TRIGGER, CHANGES TO NO. OF COUNT IN BL, COMPARE
          ASSIGN TO 1 OF THE 8 TONES.  PUT DATA IN AL (2 DOTS)

   JOY    EQU    0201H    ;GAME PORT ADDRESS
   TONE1  EQU    2BH      ;WHITE COLOR
   TONE2  EQU    2EH      ;       COLOR
   TONE3  EQU    32H      ;       COLOR
   TONE4  EQU    36H      ;       COLOR
   TONE5  EQU    39H      ;       COLOR
   TONE6  EQU    03CH     ;       COLOR
   TONE7  EQU    3FH      ; BLACK COLOR

*****************************************************************
 STACKSG SEGMENT   PARA STACK 'STACK'    ;DEFINE STACK SEGMENT
         DW        64 DUP (?)
 STACKSG     ENDS

*****************************************************************

 DATA    SEGMENT PARA PUBLIC 'DATA'  ;DEFINE DATA SEGMENT

 IMAGE   DB    640D DUP (?)          ;SPACE FOR IMAGE
 INIT    DB    27,'@'                ;INITIAL FOR PRINTER
         DB    27,'3',5              ;LINE FEED N/216
         DB    27,'1'                ;LEFT MARGIN
 TAB     DW    ?                     ;BEGINNING POSITION FOR SCANNING
 SETGRA  DB    27                    ;'ESC'
         DB    75                    ;'K'
 NOCOLUM DW    ?                     ;(WIDTH *60) PRINTER'S HORIZONTAL
 NOLINE  DW    ?                     ;NO. OF VERTICAL LINE SCAN (40*LONG
 LOOPSCAN DW   ?                     ;(WIDTH IN INCH)*20
 CONST1  DW    ?                     ;CONSTANT FOR DELAY
 PAPER   DW    ?
 PATT0   DB        11111111B
 PATT1   DB        01111111B
 PATT2   DB        01110111B
 PATT3   DB        00110011B
 PATT4   DB        01010101B
 PATT5   DB        00010001B
 PATT6   DB        00000001B
 PATT7   DB        00000000B
 TONE    DB        ?
 VIDEOSG DW        0B800H          ; START VEDIO SEGMENT
 OFSODD  DW        2000H           ; OFFSET ODD LINES
 OFSEVEN DW        0000H           ; OFFSET EVEN LINES
 LINE    DB        00H             ; CURRENT LINE NUMBER (ODD/EVEN)

 DATA    ENDS
 DGROUP  GROUP     DATA

*****************************************************************
```

```
CODESG    SEGMENT PARA PUBLIC 'CODE'        ;DEFINE CODE SEGMENT
          PUBLIC  BEGIN

;-------------------------------------------------------
BEGIN     PROC    FAR                       ;MAIN PART OF PROGRAM

          ASSUME  CS:CODESG, DS:DGROUP, SS:STACKSG, ES:DGROUP

;START RECEIVING VALUES FROM BASIC-----------------------------
          PUSH    BP
          MOV     BP,SP
          MOV     BX,[BP]+6               ;RECEIVE LOOPSCAN
          MOV     DX,[BX]
          MOV     LOOPSCAN,DX
          MOV     BX,[BP]+8               ;RECEIVE NOLINE
          MOV     DX,[BX]
          MOV     NOLINE,DX
          MOV     BX,[BP]+10              ;RECEIVE NOCOLUM
          MOV     DX,[BX]
          MOV     NOCOLUM,DX
          MOV     BX,[BP]+12              ;RECEIVE CONSTANT 1
          MOV     DX,[BX]
          MOV     CONST1,DX
          MOV     BX,[BP]+14              ;RECEIVE TAB
          MOV     DX,[BX]
          MOV     TAB,DX
          MOV     BX,[BP]+16
          MOV     DX,[BX]
          MOV     PAPER,DX                ;RECEIVE TYPE OF PAPER

;FINISH RECEIVING VALUES ------------------------
;*************************************************************
;RESET PRINTER-------------------------------------

          MOV     CX,8
          LEA     SI,INIT

RESET:    MOV     AH,5                    ;SET MODE OUT TO PRINTER
          MOV     DL,[SI]                 ;SELECT CHARACTER
          INT     21H                     ;CALL DOS
          INC     SI                      ;GET NEXT CHARACTER
          LOOP    RESET

;RESET PRINTER COMPLETED---------------------------------------
;
;GET STATUS FROM PRINTER
STATUS:   MOV     AH,2
          MOV     DX,0
          INT     17H                     ;READ STATUS OF PRINTER
          AND     AH,10000000B
          JZ      STATUS
;
;-------------------------------------------------------------
```

```
;START SCAN AND RECEIVE DATA
;SET INITIAL VALUES
            MOV     BX,NOLINE           ;BX HOLDS NO. OF LINES
            MOV     LINE,00
 LINELOOP:  CALL    HEAD                ;CALL 'HEAD' CONTROLLING SUBROUTI

;GET PRINTER'S STATUS AGAIN
  STATUS2:  MOV     AH,2
            MOV     DX,0
            INT     17H                 ;READ STATUS OF PRINTER
            AND     AH,10000000B
            JZ      STATUS2
            LEA     DI,IMAGE            ;DI HOLDS OFFSET ADDRESS OF IMAGE
            SUB     AH,AH               ;AH=0(AH IDENTIFIES TYPE OF DOT)
            CALL    ONE_LINE            ;CALL 'SCAN 1 LINE' SUBROUTINE
            PUSH    BX
            CALL    PROCIMAGE
            POP     BX
  FINISH:   DEC     BX                  ;FINISH SCANNING ?
            JNZ     LINELOOP            ;NO, SCAN THE NEXT LINE
            MOV     OFSEVEN,0000H
            MOV     OFSODD,2000H
            POP     BP
            RET     2
  BEGIN     ENDP
-------------------------------------------------------------------------
  HEAD      PROC    NEAR
            MOV     CX,4
            LEA     SI,SETGRA
  SCANLINE:
            MOV     AH,5                ;SET MODE OUT TO PRINTER
            MOV     DL,[SI]             ;SELECT CHARACTER
            INT     21H                 ;CALL DOS
            INC     SI                  ;NEXT CHARACTER
            LOOP    SCANLINE

SEND BLANK TO PRINTER
            MOV     CX,NOCOLUM
   BLANK:   MOV     AH,5
            MOV     DL,0                ;ASCII FOR BLANK
            INT     21H
            LOOP    BLANK

SEND LINEFEED TO PRINTER
            MOV     AH,5
            MOV     DL,0AH              ;ASCII FOR LINEFEED
            INT     21H
            RET

  HEAD      ENDP
```

```
;-------------------------------------------------
;ONE_LINE IS THE MAIN PROCEDURE TO SCAN ONE ENTIRE LINE
 ONE_LINE PROC    NEAR

;DELAY FOR PRINTER'S HEAD--------------------------------
          CMP     CONST1,10288D
          JE      XT
          MOV     CX,0FFFFH
 DLAY:    LOOP    DLAY
          MOV     CX,3000H
 DLAY1:   LOOP    DLAY1
          JMP     D1
 XT:      MOV     CX,9000H
 DLAY2:   LOOP    DLAY2
;END DELAY FOR PRINTER'S HEAD----------------------------

 D1:      PUSH    BX                ;SAVE 'NO. OF LINE' ON STACK
          MOV     CX,LOOPSCAN       ;MOVE HORIZONTAL SCAN IN CX


 TRIG2:   PUSH    CX                ;SAVE CX ON STACK
 TRIG:    CALL    TRIGGER           ;CALL 'TRIGGER' SUBROUTINE

;-------------------------------------------------------------
;THE NO. OF COUNT IS NOW IN BL
;COMPARE WITH REFERENCE 8 TONES
          CMP     PAPER,1
          JG      TURBO
          SHL     BL,1
 TURBO:   CMP     CONST1,10288D
          JG      TONES
          SHL     BL,1
 TONES:   CMP     BL,TONE1          ;COMPARE WITH TONE1
          JG      TEST2             ;IS IT > TONE1 ? YES, GOTO TEST
          MOV     AL,00H            ;IT'S TONE1, GOTO SHIFT
          JMP     SHIFT
 TEST2:   CMP     BL,TONE2          ;COMPARE WITH TONE2
          JG      TEST3             ;IS IT > TONE2 ? YES, GOTO TEST
          MOV     AL,01H            ;IT'S TONE2, GOTO SHIFT
          JMP     SHIFT
 TEST3:   CMP     BL,TONE3          ;COMPARE WITH TONE3
          JG      TEST4             ;IS IT > TONE3 ? YES, GOTO TEST
          MOV     AL,02H            ;IT'S TONE3, GOTO SHIFT
          JMP     SHIFT
 TEST4:   CMP     BL,TONE4
          JG      TEST5
          MOV     AL,03H
          JMP     SHIFT
 TEST5:   CMP     BL,TONE5
          JG      TEST6
          MOV     AL,04H
          JMP     SHIFT
 TEST6:   CMP     BL,TONE6
          JG      TEST7
          MOV     AL,05H
```

```
        JMP     SHIFT
ST7:    CMP     BL,TONE7
        JG      TEST8
        MOV     AL,06H
        JMP     SHIFT
ST8:    MOV     AL,07H

FT THE 4-BIT VALUE IN AL TO THE LEFT (IF IT'S TYPE A)
HIFT:   INC     AH                      ;INCREASE AH TO IDENTIFY TYPE OF DOT
        TEST    AH,02H                  ;IS IT TYPE A DOT ?
        JNZ     TYPE_B                  ;NO, DON'T SHIFT. GO TO TYPE_B
        CALL    DELAY                   ;GOTO SUBROUTINE DELAY
        MOV     CL,4                    ;PUT NO. OF BIT TO BE SHIFTED
        SHL     AL,CL                   ;SHIFT LEFT AL
        PUSH    AX                      ;PUSH AL IN STACK FOR LATER USE
        JMP     TRIG                    ;GO AND GET TYPE_B DOT

CESS FOR TYPE_B DOT
PE_B:   CALL    DELAY                   ;CALL SUBROUTINE DELAY
        MOV     BH,AL                   ;PUT TYPE_B DOT'S TONE IN BH
        POP     AX                      ;POP AX FROM STACK
        ADD     AL,BH                   ;ADD WITH TYPE_A DOT, PUT IN AL
        MOV     [DI],AL                 ;MOVE TO MEMORY
        INC     DI                      ;INCREASE MEMORY'S ADDRESS
        SUB     AH,AH                   ;RESET AH
        POP     CX                      ;GET VALUE OF LOOPSCAN FROM STACK
        LOOP    TRIG2                   ;DECREASE CX BY 1 ONE_LINE DONE? IF N
G2
        POP     BX                      ;GET NO. OF REMAINING LINES FROM STAC
        RET                             ;RETURN

_LINE ENDP                              ;END 'SCAN 1 LINE' PROCESS
-------------------------------------------------------------------
GGER    PROC    NEAR

        MOV     AL,0                    ;INITIALIZE AL
        MOV     BL,0                    ;INITIALIZE NO. OF COUNT
        MOV     DX,JOY                  ;MOVE GAME PORT ADDRESS

 TRIGGER TO GAME PORT
        OUT     DX,AL                   ;SEND TRIGGER TO GAME PORT
        NOP
        NOP
        NOP
        NOP
        NOP
P:      IN      AL,DX                   ;GET GAME PORT'S OUTPUT
        TEST    AL,01H                  ;TEST BIT 0 STATUS
        JZ      TEST1                   ;IS IT 0 ? YES, GO TO TEST1
        INC     BL                      ;NO  INCREASE BL BY 1
        TEST    BL,01111111B
        JZ      TEST1
        JMP     LOOP                    ;CHECK BIT 0 STATUS AGAIN
T1:     SHR     BL,1                    ;DIVIDE BL BY 2
        RET
```

```
RIGGER  ENDP

;-------------------------------------------------------------
DELAY   PROC    NEAR                    ;DELAY PROCEDURE

        CMP     AH,01H                  ;CHECK WHETHER IT'S TYPE A OR B
        JZ      DELAY2                  ;TYPE B--GO TO DELAY2
        MOV     CX,CONST1               ;TYPE A--PUT CONSTANT 1 IN CX
        JMP     DELAY1                  ;JUMP TO DELAY1
        SUB     CONST1,8D               ;CONST2 = CONST1 - 8
ELAY2:  MOV     CX,CONST1               ;PUT CONSTANT 2 IN CX
ELAY1:  PUSH    AX                      ;SAVE AX ON STACK
        MOV     AH,25D                  ;PUT AH=20D FOR DELAY DUE TO TONE-CO
                                        ;TIME SPENT = 27 + (TONE)*20
        MUL     AH                      ;MULTIPLY 20 AND [TONE]
        SUB     CX,AX                   ;SUBTRACT FROM TOTAL TIME
        MOV     AL,BL                   ;MOVE NO. OF COUNT TO AL
        MOV     AH,170                  ;PUT AH=33D FOR DELAY DUE TO COUNT
                                        ;TIME SPENT = 28 + (COUNT)*33
        MUL     AH                      ;MUTIPLY 33 AND [COUNT]
        SUB     CX,AX                   ;SUBTRACT FROM REMAINING TIME
        MOV     AX,CX                   ;PUT NO. OF DELAY CLOCK IN AX
        MOV     DX,00
        MOV     CX,20                   ;CL HOLDS NO. OF CLOCK PER 1 DELAY LO
        DIV     CX                      ;DIVISION TO GET NO. OF DELAY LOOP
        MOV     CX,AX
:       LOOP    T
        POP     AX                      ;GET THE OLD VALUE OF AH AND AL

        RET                             ;RETURN FROM DELAY SUBROUTINE

LAY     ENDP

;-------------------------------------------------------------

IMAGE   PROC  NEAR
              LEA       SI,IMAGE        ; SET ADDRESS OF DATA
              TEST      LINE,0001H      ; IS LINE ODD OR EVEN
              JZ        T10             ; IF EVEN GOTO T10
              MOV       DI,OFSODD       ; IF ODD
              ADD       OFSODD,80       ; LOAD ODD OFFSET OF VIDEO RAM
              JMP       T20
   T10:
              MOV       DI,OFSEVEN      ; IF EVEN
              ADD       OFSEVEN,80      ; LOAD EVEN OFFSET OF VIDEO RAM
   T20:
              INC       LINE            ; CHANGE TO NEXT CURRENT LINE
              MOV       DX,LOOPSCAN     ; SET NO.BYTE_IN_LINE
              SHR       DX,1
   B20:
   ;START PROCESS 1 BYTE
              MOV       CX,0002         ; PROCESS 4 POINT IN 1 BYTE
              MOV       AL,[SI]         ; LOAD DATA
              MOV       TONE,AL         ; STORE DATA TO TONE
```

```
 B10:
 ;PROCESS 1 POINT
            CALL     READTONE
            CALL     READPATT
            CALL     MAKEPATT
            CALL     SHIFPATT
;COMPLETE PROCESS 1 POINT
            LOOP     B10
            INC      SI
;COMPLETE PROCESS 2 POINT
            MOV      CX,0002          ; PROCESS 4 POINT IN 1 BYTE
            MOV      AL,[SI]          ; LOAD NEXT DATA
            MOV      TONE,AL          ; STORE DATA TO TONE
 B11:
 ;PROCESS 1 POINT
            CALL     READTONE
            CALL     READPATT
            CALL     MAKEPATT
            CALL     SHIFPATT
;COMPLETE PROCESS 1 POINT
            LOOP     B11
            INC      SI
;COMPLETE PROCESS 1 BYTE
            PUSH     DS               ; SAVE DATA SEGMENT
            MOV      AX,VIDEOSG
            MOV      DS,AX
            MOV      [DI],BL          ; STORE PATTERN TO VIDEO
            POP      DS               ; RESTORE DATA SEGMENT
            INC      DI
            DEC      DX
            JNZ      B20
;COMPLETE PROCESS 1 LINE
            CALL     SHF_PATT         ; CHANGE PATTERN FOR ANOTHER LI
            RET
PROCIMAGE   ENDP
;-------------------------------------------------
READTONE    PROC     NEAR
            SUB      AL,AL            ; CLEAR AL
            MOV      AH,TONE          ; SET DATA TO AH
            RCL      AH,1             ; SHIFT DATA TO AL
            RCL      AH,1             ; FOR
            RCL      AL,1             ; CHECK TONE
            RCL      AH,1
            RCL      AL,1
            RCL      AH,1
            RCL      AL,1
            MOV      TONE,AH
            RET
READTONE    ENDP
;-------------------------------------------------
READPATT    PROC     NEAR
            CMP      AL,00
            JE       READ0
            CMP      AL,01
```

```
                JE          READ1
                CMP         AL,02
                JE          READ2
                CMP         AL,03
                JE          READ3
                CMP         AL,04
                JE          READ4
                CMP         AL,05
                JE          READ5
                CMP         AL,06
                JE          READ6
                MOV         AH,PATT7
                JMP         STOPREAD
READ0:
                MOV         AH,PATT0
                JMP         STOPREAD
READ1:
                MOV         AH,PATT1
                JMP         STOPREAD
READ2:
                MOV         AH,PATT2
                JMP         STOPREAD
READ3:
                MOV         AH,PATT3
                JMP         STOPREAD
READ4:
                MOV         AH,PATT4
                JMP         STOPREAD
READ5:
                MOV         AH,PATT5
                JMP         STOPREAD
READ6:
                MOV         AH,PATT6
STOPREAD:
                RET
READPATT        ENDP
;-----------------------------------------
MAKEPATT        PROC        NEAR
                RCL         AH,1            ; SHIFT PATTERN
                RCL         BL,1            ; FROM AH
                RCL         AH,1            ; TO BL
                RCL         BL,1
                RET
MAKEPATT        ENDP
;-----------------------------------------
SHIFPATT        PROC        NEAR
                PUSH        SI
                PUSH        CX
                MOV         CX,06
                LEA         SI,PATT1
SHF10:
                MOV         AH,[SI]
                ROL         AH,1
                MOV         [SI],AH
```

```
                INC     SI
                LOOP    SHF10
                POP     CX
                POP     SI
                RET
SHIFPATT        ENDP
;-----------------------------------------------
SHF_PATT        PROC    NEAR
                MOV     CX,04
                ROL     PATT1,CL        ; 01111111B
                ROL     PATT6,CL        ; 00000001B

                ROL     PATT2,1         ; 01110111B
                ROL     PATT2,1

                ROL     PATT3,1         ; 00110011B
                ROL     PATT3,1

                ROL     PATT5,1         ; 00010001B
                ROL     PATT5,1

                RET
SHF_PATT        ENDP

CODESG      ENDS

;*****************************************************************
        END     BEGIN
```

```
'PROGRAM MAIN FOR CALL ASSEMBLY  PROGRAM
DEFINT  A-Z
     DIM IMAGE(1500)
     DIM POSX(500),POSY(500)
     DIM BUFX(640),BUFY(200)
     INPUT  "SCREEN 1 OR 2 : ",SRN
     SCREEN SRN
     X2 = (320*SRN)-1 : X1 = X2-88
     COL = 70
     IF SRN = 1 THEN COL =30
'     KEY(1) ON
'     ON KEY(1) GOSUB BACKIMAGE
     GET (X1,0)-(X2,100),IMAGE
MENU:
     LOCATE 1,COL:  PRINT "ฉฉฉฉฉฉฉฉฉฉฉฉฉฉฉฉ"
     LOCATE 2,COL:  PRINT "ค Save     ค"
     LOCATE 3,COL:  PRINT "ค Load     ค"
     LOCATE 4,COL:  PRINT "ค sCan     ค"
     LOCATE 5,COL:  PRINT "ค segMent  ค"
     LOCATE 6,COL:  PRINT "ค Block    ค"
     LOCATE 7,COL:  PRINT "ค Print    ค"
     LOCATE 8,COL:  PRINT "ค Exit     ค"
     LOCATE 9,COL:  PRINT "ค          ค"
     LOCATE 10,COL: PRINT "ค          ค"
     LOCATE 11,COL: PRINT "ค          ค"
     LOCATE 12,COL: PRINT "ชฉฉฉฉฉฉฉฉฉฉฉฉ"
CHK:    SEL$ = INKEY$
        IF SEL$ = "" THEN GOTO CHK
        IF SEL$ = "s" OR SEL$ = "S" THEN GOTO SAVEPROG
        IF SEL$ = "l" OR SEL$ = "L" THEN GOTO LOADPROG
        IF SEL$ = "c" OR SEL$ = "C" THEN GOTO SCAN
        IF SEL$ = "p" OR SEL$ = "P" THEN GOTO PRI
        IF SEL$ = "b" OR SEL$ = "B" THEN GOTO BLOCK
        IF SEL$ = "e" OR SEL$ = "E" THEN GOTO INIT
        IF SEL$ = "m" OR SEL$ = "M" THEN GOTO SEGM
GOTO MENU
'-------------------------------------------------------------------
SAVEPROG :
     LOCATE 10,COL+1
     PRINT "Save..."
     LOCATE 11,COL+1
     INPUT "", NAM$
     NAM$ = NAM$ + ".PIC"
     GOSUB BACKIMAGE
     DEF SEG = &HB800
     BSAVE NAM$,0,&H4000
     DEF SEG
GOTO MENU
'-------------------------------------------------------------------
LOADPROG :
   LOCATE 10,COL+1
   PRINT "Load..."
   LOCATE 11,COL+1
   INPUT "",LNAM$
```

```
    LOCATE 10,COL+1
    PRINT "          "
    LOCATE 11,COL+1
    PRINT "          "
    LNAM$ = LNAM$+ ".PIC"
    DEF SEG =&HB800
    BLOAD LNAM$,0
    DEF SEG
    GET (X1,0)-(X2,100),IMAGE
GOTO MENU
'-------------------------------------------------------------------
SCAN :
     cls 0
     INPUT "LENGTH FOR VERTICAL SCAN : ",VER!
     PRINT
     INPUT "LENGTH FOR HORIZONTAL SCAN : ",HOR!
     PRINT
     INPUT "LEFT MARGIN (IN INCHES) : ",INCH!
     PRINT
     INPUT "1. XT  ¦  2.TURBO   " ;KIND
     PRINT
     PRINT "TYPE OF PAPER "
     INPUT "1.SHINY   2.NORMAL ";PAPER
' CHECK FOR CORRECT CONDITION
     IF KIND = 1 THEN CONSTANT1 = 10288  ELSE  CONSTANT1 = 22588
     IF INCH! = 0 THEN INCH! = 0.1
     IF VER! > 5 THEN VER! = 5
     LONG! = HOR! + INCH!
     IF LONG! > 6.5 THEN HOR! = 6.5  AND INCH! = 0.1
     NOL =  INT(VER!*40)
     NOC =  INT(60*HOR!)
     MARGIN = INT(INCH!*10)
     LOP =  INT(HOR!*20)
     CLS 0
     CALL BEGIN (PAPER,MARGIN,CONSTANT1,NOC,NOL,LOP)
     GET (X1,0)-(X2,100),IMAGE
GOTO MENU
'-------------------------------------------------------------------
BACKIMAGE :
     PUT (X1,0),IMAGE,PSET
RETURN
'-------------------------------------------------------------------
SEGM :
     LOCATE 10,COL+1
     PRINT "END x,y.."
     LOCATE 11,COL+1
     INPUT "",CLUM,ROW
     LOCATE 10,COL+1
     GOSUB BACKIMAGE
     GOSUB  SEGMENT
     GET (X1,0)-(X2,100),IMAGE
GOTO MENU
'-------------------------------------------------------------------
```

```
SEGMENT;
     COUNT = 0
     FOR CCOLUMN = 1 TO CLUM
        FOR CROW = 1 TO  ROW
              CHK = POINT (CCOLUMN,CROW)
              IF CHK =0 THEN GOSUB APART
        NEXT CROW
     NEXT CCOLUMN
RETURN
'-------------------------------------------------------------------
APART :
         TRACK = 1
         GOSUB SETDISPLAY
         TCOLUMN = CCOLUMN : TROW = CROW
         DO WHILE  TRACK <> 0
         GOSUB FILL
         GOSUB CHECK
         GOSUB SHIFT
        LOOP
RETURN
'-------------------------------------------------------------------
SETDISPLAY:
        COUNT  = COUNT+1
        FACTOR =  INT(COUNT/15)
        DISPLAYX = (COUNT-(FACTOR*15))*20
        DISPLAYY = FACTOR*40+80
       LOCATE 22,1 : PRINT "COUNT : ";COUNT
RETURN
'-------------------------------------------------------------------
FILL:
       X = TCOLUMN  : Y = TROW    :YY = Y
       PSET (X,Y),2
       OFFX =  CCOLUMN-TCOLUMN
       OFFY1 =  CROW - TROW
       PSET (DISPLAYX-OFFX,DISPLAYY-OFFY1),3
'FILL ABOVE
        FLAG   =  0
1:      FLAG1 = POINT(X,Y-1)
        IF FLAG1 <> 0 THEN FLAG = 1
        IF FLAG = 1 THEN GOTO UBND
        Y = Y-1
        PSET (X,Y),2
        OFFY2 = CROW-Y
        PSET (DISPLAYX-OFFX,DISPLAYY-OFFY2),3
        GOTO 1
UBND:   YMIN = Y
'FILL BELOW
       FLAG2 = 0
2:      FLAG3 = POINT(X,YY+1)
        IF FLAG3 <> 0 THEN FLAG2 = 1
        IF FLAG2 = 1 THEN GOTO LBND
```

```
            YY= YY+1
            PSET (X,YY),2
            OFFY3 = CROW-YY
            PSET (DISPLAYX-OFFX,DISPLAYY-OFFY3),3
            GOTO 2
LBND:
          YMAX = YY
RETURN
'-------------------------------------------------------------------------
CHECK: XTEM = TCOLUMN
      MOREL  = 0
      MORER  = 0
      MOREAL = 0
      MOREAR = 0
      MOREBL = 0
      MOREBR = 0
       FOR C = YMIN TO YMAX
         FLAGR  = POINT (XTEM+1,C)
         FLAGL  = POINT (XTEM-1,C)
         FLAGAR = POINT (XTEM+1,C-1)
         FLAGBR = POINT (XTEM+1,C+1)
         FLAGAL = POINT (XTEM-1,C-1)
         FLAGBL = POINT (XTEM-1,C+1)
         IF FLAGAR = 0                          THEN GOSUB GPOSAR
         IF (FLAGR  = 0 AND FLAGAR <> 0) THEN GOSUB GPOSR
         IF (FLAGBR = 0 AND FLAGR  <> 0) THEN GOSUB GPOSBR
         IF FLAGAL = 0                          THEN GOSUB GPOSAL
         IF (FLAGL  = 0 AND FLAGAL <> 0) THEN GOSUB GPOSL
         IF (FLAGBL = 0 AND FLAGL  <> 0) THEN GOSUB GPOSBL
         IF FLAGR  <> 0 THEN MORER = 0
         IF FLAGL  <> 0 THEN MOREL = 0
         IF FLAGAR <> 0 THEN MOREAR = 0
         IF FLAGBR <> 0 THEN MOREBR = 0
         IF FLAGAL <> 0 THEN MOREAL = 0
         IF FLAGBL <> 0 THEN MOREBL = 0
       NEXT C
RETURN
'-------------------------------------------------------------------------
GPOSR :
        IF MORER = 1 THEN GOTO EGR
        POSX(TRACK) =  XTEM+1
        POSY(TRACK) =  C
        TRACK = TRACK +1
        MORER = 1
EGR:
RETURN
'-------------------------------------------------------------------------
GPOSL :
        IF MOREL = 1 THEN GOTO EGL
        POSX(TRACK) =  XTEM-1
        POSY(TRACK) =  C
        TRACK = TRACK + 1
        MOREL = 1
EGL:
RETURN
'-------------------------------------------------------------------------
```

```
GPOSAR :
        IF MOREAR = 1 THEN GOTO EGAR
        POSX(TRACK) =  XTEM+1
        POSY(TRACK) =  C-1
        TRACK = TRACK + 1
        MOREAR = 1
EGAR:
RETURN
'-------------------------------------------------------------------
GPOSBR :
        IF MOREBR = 1 THEN GOTO EGBR
        POSX(TRACK) =  XTEM+1
        POSY(TRACK) =  C+1
        TRACK = TRACK + 1
        MOREBR = 1
EGBR:
RETURN
'-------------------------------------------------------------------
GPOSAL :
        IF MOREAL = 1 THEN GOTO EGAL
        POSX(TRACK) =  XTEM-1
        POSY(TRACK) =  C-1
        TRACK = TRACK + 1
        MOREAL = 1
EGAL:
RETURN
'-------------------------------------------------------------------
GPOSBL :
        IF MOREBL = 1 THEN GOTO EGBL
        POSX(TRACK) =  XTEM-1
        POSY(TRACK) =  C+1
        TRACK = TRACK + 1
        MOREBL = 1
EGBL:
RETURN
'-------------------------------------------------------------------
SHIFT:
         TRACK = TRACK -1
         IF TRACK = 0 THEN GOTO ESFT
         TCOLUMN =  POSX(TRACK)
         TROW = POSY(TRACK)
ESFT:
RETURN
'-------------------------------------------------------------------
```

```
PRT :
'PROGRAM TO PRINT IMAGE
      XPRT1 = XBLOCK1   : YPRT1  = YBLOCK1
      XPRT2 = XBLOCK2   : YPRT2  = YBLOCK2
      LOCATE 1,COL:  PRINT "ฉวววววววววว"
      LOCATE 2,COL:  PRINT "ค Condense ค
      LOCATE 3,COL:  PRINT "ค Double   ค"
      LOCATE 4,COL:  PRINT "ค Crt      ค"
      LOCATE 5,COL:  PRINT "ค Quad     ค"
      LOCATE 6,COL:  PRINT "ค Color    ค"
      LOCATE 7,COL:  PRINT "ค Inverse  ค"
      LOCATE 8,COL:  PRINT "ค Dot-Dot  ค"
      LOCATE 9,COL:  PRINT "ค          ค"
      LOCATE 10,COL: PRINT "ค          ค"
      LOCATE 11,COL: PRINT "ค          ค"
      LOCATE 12,COL: PRINT "ยววววววววว"

'
'     FOR I = YPRT1   TO  YPRT2   STEP 8
'           FOR J = XPRT1 TO XPRT2
'                 FOR K = 1 TO 8
'
'
'
'                 NEXT K
'           NEXT J
'     NEXT I
GOTO MENU
'-------------------------------------------------------------------------------
BLOCK:
      XBLOCK1 = 1   : YBLOCK1 = 1
      XBLOCK2 = 320 : YBLOCK2 = 200
GOTO MENU
'-------------------------------------------------------------------------------
INIT :
    LPRINT CHR$(27)+"@"
END
```

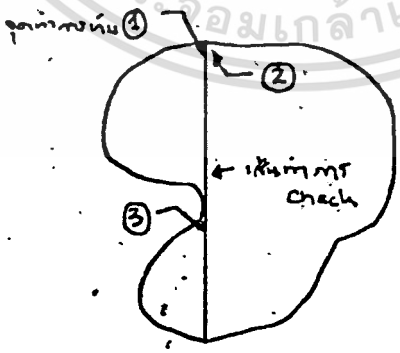
ส่วนนี้จะพูดถึงการนำเครื่องสแกนเนอร์ไปใช้งานจริง โดยยกตัวอย่างโปรแกรมที่เขียนขึ้นเพื่อแยกตัวอักษรออกจากประโยค เพื่อนำไปให้เครื่องคอมพิวเตอร์วิเคราะห์ว่า แต่ละตัวอักษรนั้นคือตัวอะไร ซึ่งการศึกษาในวิชา PATTERN RECOGNITION แต่เดิมนั้น ต้องอาศัยเครื่องสแกนที่ราคาแพงเพื่อป้อนข้อมูลภาพตัวอักษรให้แก่โปรแกรม หรือใช้วิธีป้อนข้อมูลซึ่งแบบแรกนั้นมีราคาแพง ส่วนแบบหลังก็กินเวลานานและยากต่อการกำหนดรูปแบบ (FORNT) ของตัวอักษรหลาย ๆ แบบ การใช้เครื่องสแกนที่ประดิษฐ์จะทำให้ลดความยุ่งยากและมีราคาถูกเหมาะแก่นักศึกษาใช้ในการทำวิจัยต่อไป

โปรแกรมแยกตัวอักษรจากประโยคนี้ เป็นโปรแกรมง่าย ๆ เขียนขึ้นเพื่อให้เห็นประโยชน์ของเครื่องสแกนเท่านั้น ดังนั้น การนำไปใช้งานจริงจะต้องปรับปรุงประสิทธิภาพเพิ่มขึ้น โดยเฉพาะเพื่อความเร็วอาจใช้ภาษาแอสเซมบลี้แทน ส่วนหลักการที่สำคัญของโปรแกรม SEGMENTATION อาจอธิบายได้ดังนี้

โปรแกรมจะเริ่มตรวจสอบตั้งแต่แถว (ROW) ที่ 1 จนถึงแถวที่ตั้งไว้ว่าพบตัวอักษรหรือไม่ (ตัวอักษรจะมีสีพื้นเป็นสีดำต่างกับพื้นของ BACKGROUND) ถ้าพบแล้วจะเริ่มทำการลอกรูปตัวอักษรนั้น ๆ ออกมา การลอกรูปออกมานั้น จะใช้หลักการเดียวกับการ PAINT คือ หาจุดต่อเนื่องที่เป็นสีเดียวกับพื้นตัวอักษรและลอกออกมาในบริเวณที่กำหนดไว้ ขณะที่ลอกนั้นก็จะทำการเปลี่ยนสีพื้นตัวอักษรเพื่อจะได้ไม่มีการตรวจเจอตัวอักษรเดิมอีก การหาจุดต่อเนื่องที่เป็นสีเดียวกับพื้นตัวอักษรอาจแบ่งได้เป็น 3 ส่วนใหญ่คือ

1) ส่วนที่หาจุดต่อเนื่องในแนวดิ่ง ส่วนนี้จะทำหน้าที่หาจุดต่อเนื่องในแนวดิ่งว่ามีจุดสิ้นสุดและเริ่มต้นที่ใดและจะทำการเปลี่ยนสีตัวอักษรไปด้วย

2) ส่วนทำการตรวจว่า จุดด้านข้างของเส้นตรงในข้อ 1 ทั้งซ้ายและขวา นั้นเป็นจุดต่อเนื่องหรือไม่ โดยจะตรวจสอบทั้งด้านซ้าย, ขวา, ซ้ายบน, ซ้ายล่าง, ขวาบน, ขวาล่าง, และจะทำการเก็บไว้ใน ARRAY การเก็บนี้จะเก็บเฉพาะตำแหน่งเริ่มต้นแต่ละแห่งของจุดต่อเนื่องเท่านั้น จะไม่เก็บทุกจุด ดังรูป



3) การเลื่อนเส้นตรงไปยังตำแหน่งใหม่ที่เก็บไว้ใน ARRAY ข้อ 2 จะทำการเลื่อนตำแหน่งไปยังจุดใหม่ที่เก็บไว้ และโปรแกรมจะเริ่มทำข้อ 1 ใหม่ เป็นอย่างนี้เรื่อยไปจนกระทั่งหมด ARRAY ที่เก็บไว้ก็จะได้ครบ 1 ตัวอักษร

โปรแกรมที่เขียนนี้ เขียนด้วย QUICK BASIC จึงอาจทำให้ความเร็วไม่ดีเท่าที่ควร รวมทั้งการเก็บตำแหน่งของจุดในข้อ 2 ต้องใช้หน่วยความจำมาก เนื่องจาก QUICK BASIC ใช้ 2 ไบท์ต่อ 1 ARRAY (1 BYTE ก็เพียงพอแล้วสำหรับเก็บตำแหน่ง COORDINATE)

นอกจากใช้แยกตัวอักษรแล้ว ยังอาจใช้แยกวัตถุใด ๆ ก็ได้ แต่มีข้อแม้ว่าวัตถุ 2 ชิ้น ต้องแยกออกจากกันเสียก่อน ไม่มีส่วนหนึ่งส่วนใดติดกันนั่นเอง

ต่อไปนี้คือโปรแกรมแยกตัวอักษร

```
SEGM :
      LOCATE 10,COL+1
      PRINT "END x,y.."
      LOCATE 11,COL+1
      INPUT "",CLUM,ROW
      LOCATE 10,COL+1
      GOSUB BACKIMAGE
      GOSUB  SEGMENT
      GET (X1,0)-(X2,100),IMAGE
GOTO MENU
'-------------------------------------------------------------
BACKIMAGE :
     PUT (X1,0),IMAGE,PSET
RETURN
'-------------------------------------------------------------
SEGMENT:
     COUNT = 0
     FOR CCOLUMN = 1 TO CLUM
        FOR CROW = 1 TO  ROW
              CHK = POINT (CCOLUMN,CROW)
              IF CHK =0 THEN GOSUB APART
        NEXT CROW
     NEXT CCOLUMN
RETURN
'-------------------------------------------------------------
APART :
         TRACK = 1
         GOSUB SETDISPLAY
         TCOLUMN = CCOLUMN : TROW = CROW
         DO WHILE  TRACK <> 0
         GOSUB FILL
         GOSUB CHECK
         GOSUB SHIFT
        LOOP
```

```
RETURN
'-------------------------------------------------------------------------
SETDISPLAY:
        COUNT  = COUNT+1
        FACTOR =  INT(COUNT/15)
        DISPLAYX = (COUNT-(FACTOR*15))*20
        DISPLAYY = FACTOR*40+80
       LOCATE 22,1 : PRINT "COUNT : ";COUNT
RETURN
'-------------------------------------------------------------------------
FILL:
       X = TCOLUMN  : Y = TROW    :YY = Y
       PSET (X,Y),2
       OFFX =  CCOLUMN-TCOLUMN
       OFFY1 =  CROW - TROW
       PSET (DISPLAYX-OFFX,DISPLAYY-OFFY1),3
'FILL ABOVE
         FLAG   =  0
1:       FLAG1 = POINT(X,Y-1)
         IF FLAG1 <> 0 THEN FLAG = 1
         IF FLAG = 1 THEN GOTO UBND
         Y = Y-1
         PSET (X,Y),2
         OFFY2 = CROW-Y
         PSET (DISPLAYX-OFFX,DISPLAYY-OFFY2),3
         GOTO 1
UBND:    YMIN = Y
'FILL BELOW
      FLAG2 = 0
2:       FLAG3 = POINT(X,YY+1)
         IF FLAG3 <> 0 THEN FLAG2 = 1
         IF FLAG2 = 1 THEN GOTO LBND
         YY= YY+1
         PSET (X,YY),2
         OFFY3 = CROW-YY
         PSET (DISPLAYX-OFFX,DISPLAYY-OFFY3),3
         GOTO 2
LBND:
       YMAX = YY
RETURN
'-------------------------------------------------------------------------
CHECK:  XTEM = TCOLUMN
    MOREL  = 0
    MORER  = 0
    MOREAL = 0
    MOREAR = 0
    MOREBL = 0
    MOREBR = 0
     FOR C = YMIN TO YMAX
       FLAGR  = POINT (XTEM+1,C)
```

```
        FLAGL  = POINT (XTEM-1,C)
        FLAGAR = POINT (XTEM+1,C-1)
        FLAGBR = POINT (XTEM+1,C+1)
        FLAGAL = POINT (XTEM-1,C-1)
        FLAGBL = POINT (XTEM-1,C+1)
        IF FLAGAR = 0                          THEN GOSUB GPOSAR
        IF (FLAGR  = 0 AND FLAGAR <> 0) THEN GOSUB GPOSR
        IF (FLAGBR = 0 AND FLAGR  <> 0) THEN GOSUB GPOSBR
        IF FLAGAL = 0                          THEN GOSUB GPOSAL
        IF (FLAGL  = 0 AND FLAGAL <> 0) THEN GOSUB GPOSL
        IF (ELAGBL = 0 AND FLAGL  <> 0) THEN GOSUB GPOSBL
        IF FLAGR  <> 0 THEN MORER = 0
        IF FLAGL  <> 0 THEN MOREL = 0
        IF FLAGAR <> 0 THEN MOREAR = 0
        IF FLAGBR <> 0 THEN MOREBR = 0
        IF FLAGAL <> 0 THEN MOREAL = 0
        IF FLAGBL <> 0 THEN MOREBL = 0
      NEXT C
RETURN
'-------------------------------------------------------------------------
GPOSR :
      IF MORER = 1 THEN GOTO EGR
      POSX(TRACK) =  XTEM+1
      POSY(TRACK) =  C
      TRACK = TRACK +1
      MORER = 1
EGR:
RETURN
'-------------------------------------------------------------------------
GPOSL :
      IF MOREL = 1 THEN GOTO EGL
      POSX(TRACK) =  XTEM-1
      POSY(TRACK) =  C
      TRACK = TRACK + 1
      MOREL = 1
EGL:
RETURN
'-------------------------------------------------------------------------
GPOSAR :
      IF MOREAR = 1 THEN GOTO EGAR
      POSX(TRACK) =  XTEM+1
      POSY(TRACK) =  C-1
      TRACK = TRACK + 1
      MOREAR = 1
EGAR:
RETURN
'-------------------------------------------------------------------------
GPOSBR :
```

```
        IF MOREBR = 1 THEN GOTO EGBR
        POSX(TRACK) =  XTEM+1
        POSY(TRACK) =  C+1
        TRACK = TRACK + 1
        MOREBR = 1
EGBR:
RETURN
'-------------------------------------------------------------------
GPOSAL :
        IF MOREAL = 1 THEN GOTO EGAL
        POSX(TRACK) =  XTEM-1
        POSY(TRACK) =  C-1
        TRACK = TRACK + 1
        MOREAL = 1
EGAL:
RETURN
'-------------------------------------------------------------------
GPOSBL :
        IF MOREBL = 1 THEN GOTO EGBL
        POSX(TRACK) =  XTEM-1
        POSY(TRACK) =  C+1
        TRACK = TRACK + 1
        MOREBL = 1
EGBL:
RETURN
'-------------------------------------------------------------------
SHIFT:
        TRACK = TRACK -1
        IF TRACK = 0 THEN GOTO ESFT
        TCOLUMN =  POSX(TRACK)
        TROW = POSY(TRACK)
ESFT:
RETURN
'-------------------------------------------------------------------
```

COUNT : 29

ตัวอย่างการทำงานของโปรแกรม SEGMENTATION กับเครื่องสแกน

TYPE TIL139
SOURCE AND DETECTOR ASSEMBLY

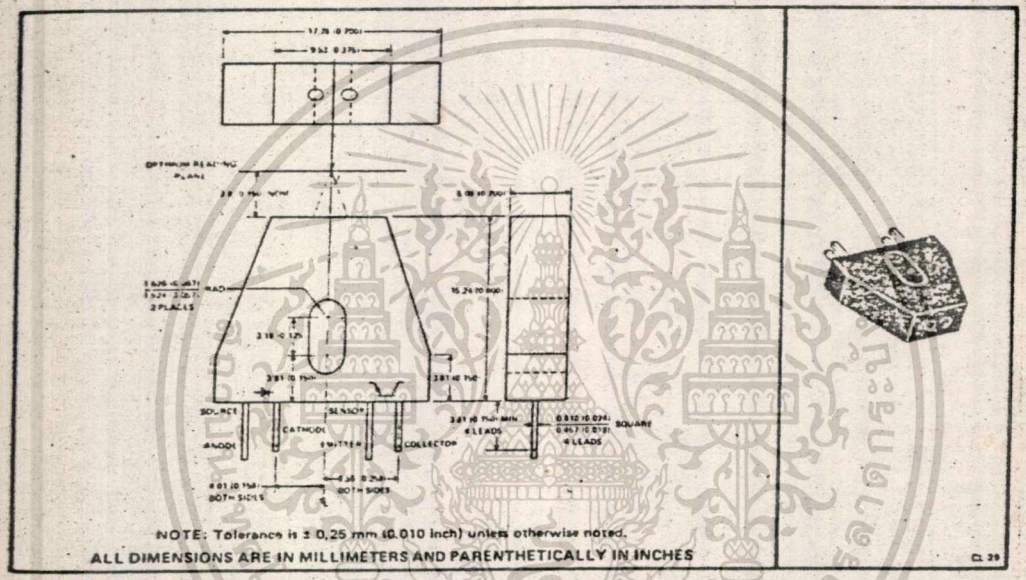
D1100, SEPTEMBER 1971–REVISED MARCH 1983

OPTOELECTRONIC MODULE FOR REFLECTIVE SENSING APPLICATIONS

- **Adaptable for Printed Circuit Board Mounting**
- **Designed for Sensing Applications such as Line Finders, Batch Counters, Level Indicators, and Beginning-of-Tape/End-of-Tape Indicators**

mechanical data

The assembly consists of an infrared emitting diode and an n-p-n silicon phototransistor mounted in a plastic housing. The assembly will withstand soldering temperature with no deformation and device performance characteristics remain stable when operated in high-humidity conditions. Total assembly weight is approximately 1.2 grams.



NOTE: Tolerance is ± 0,25 mm (0.010 inch) unless otherwise noted.

ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

CL 39

absolute maximum ratings at 25°C free-air temperature (unless otherwise noted)

Source Reverse Voltage	2 V
Source Continuous Forward Current (See Note 1)	40 mA
Sensor Collector-Emitter Voltage	50 V
Sensor Emitter-Collector Voltage	7 V
Sensor Continuous Dissipation at (or below) 25°C Free-Air Temperature (See Note 2)	50 mW
Operating Free-Air Temperature Range	−40°C to 80°C
Storage Temperature Range	−40°C to 85°C
Lead Temperature 1,6 mm (1/16 Inch) from Assembly for 5 Seconds	240°C

NOTES. 1. Derate linearly to 80°C free-air temperature at the rate of 0.73 mA/°C.
2. Derate linearly to 80°C free-air temperature at the rate of 0.91 mW/°C.

NE/SE/SA558/559-F,N

DESCRIPTION

The SA/SE/NE558 and 559 Quad Timers are monolithic timing devices which can be used to produce four entirely independent timing functions. The 558 output sinks current whereas the 559 sources current. These highly stable, general purpose controllers can be used in a monostable mode to produce accurate time delays, from microseconds to hours. In the time delay mode of operation, the time is precisely controlled by one external resistor and one capacitor. Astable operation can be achieved by using two of the four timer sections.

The four timing sections in the 558 and 559 are edge triggered; therefore, when connected in tandem for sequential timing applications, no coupling capacitors are required. Output current capability of 100mA is provided in both devices.

FEATURES

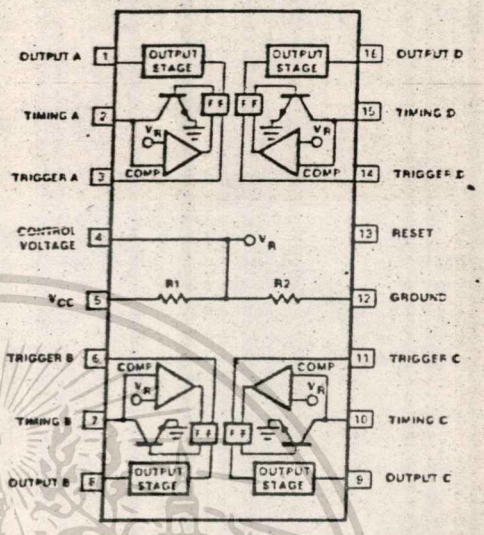
- 100mA output current per section
- Edge triggered (no coupling capacitor)
- Output independent of trigger conditions
- Wide supply voltage range 4.5V to 18V
- Timer intervals from microseconds to hours
- Time period equals RC
- Military qualifications pending

APPLICATIONS

- Sequential timing
- Time delay generation
- Precision timing
- Industrial controls
- Quad one-shot

PIN CONFIGURATION

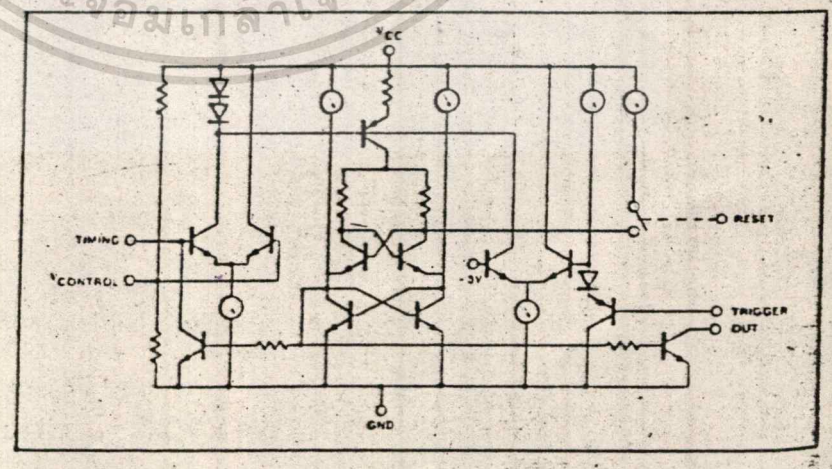
F,N PACKAGE



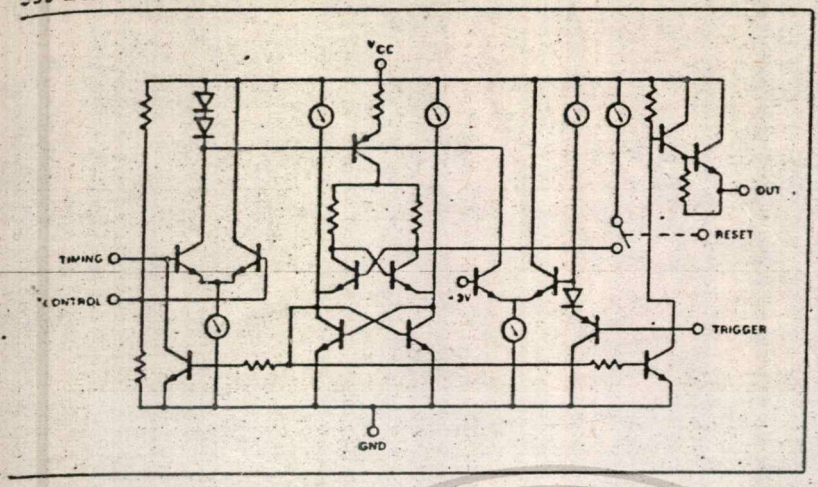
ABSOLUTE MAXIMUM RATINGS

PARAMETER	RATING	UNIT
Supply voltage		
SE558, SE559	+18	V
NE558, NE559	+16	V
SA558, SA559	+16	V
Power dissipation	1.25	W
Operating temperature range		
NE558, NE559	0 to +70	°C
SA558, SA559	-40 to +85	°C
SE558, SE559	-55 to +125	°C
Storage temperature range	-65 to +150	°C
Lead temperature (soldering, 60sec)	+300	°C

558 EQUIVALENT CIRCUIT



559 EQUIVALENT CIRCUIT



ELECTRICAL CHARACTERISTICS $T_A = 25°C$, $V_{CC} = +5V$ to $+15V$ unless otherwise specified.

PARAMETER	TEST CONDITIONS	SE558/SE559			NE558/NE559 SA558/SA559			UNIT
		Min	Typ	Max	Min	Typ	Max	
Supply voltage		4.5		18	4.5		16	V
Supply current (558)	V_{CC} = Reset = 15V		21	32		27	36	mA
(559)	V_{CC} = Reset = 15V		9	16		12	18	mA
Timing accuracy ($T = RC$)	R = 2kΩ to 100kΩ C = 1μF							
Initial accuracy			1.0	3		2		%
Drift with temperature			150			150		ppm/°C
Drift with supply voltage			0.1			0.1		%/V
Trigger voltage[1]	V_{CC} = 15V	0.8	1.5	2.4	0.8	1.5	2.4	V
Trigger current	Trigger = 0V		5	30		5	100	μA
Reset voltage[2]		0.8	1.5	2.4	0.8	1.5	2.4	V
Reset current	Reset		50	300		50		μA
Threshold voltage			0.63			0.63		xVCC
Threshold leakage			15			15		nA
Output voltage (558)[3]	I_L = 10mA		0.1	0.2		0.1	0.4	V
	I_L = 100mA		0.7	1.5		1.0	2.0	V
Output voltage (559)[4]	I_L = 10mA	13	13.6		12.5	13.3		V
	I_L = 100mA	12.5	13.3		12.0	13.0		V
Output leakage			10			10		nA
Propagation delay (558)			1.0			1.0		μs
(559)			0.4			0.4		μs
Risetime of output	I_L = 100mA		100			100		ns
Falltime of output	I_L = 100mA		100			100		ns

NOTES:

1. Trigger functions only on the falling edge of the trigger pulse only after previously the trigger is high. After reset the trigger must be brought high and then low to implement triggering.
2. For reset below 0.8 volts, outputs set low and trigger inhibited. For reset above 2.4 volts, trigger enabled.
3. The 558 output structure is open collector which requires a pull up resistor to V_{CC} to source current. The output is normally low sinking current.
4. The 559 output structure is a darlington emitter follower which requires a pull down resistor to ground to source current. The output is normally low and sources current when switched high.

หนังสืออ้างอิง

1. ยืน ภู่วรวรรณ, สุรศักดิ์ สงวนพงษ์ "โปรแกรมคอมพิวเตอร์ภาษาแอสเซมบลี้" สำนักพิมพ์ซีเอ็ดยูเคชั่น พ.ศ. 2529
2. ยืน ภู่วรวรรณ "ไมโครคอมพิวเตอร์ 16 บิต" สำนักพิมพ์ซีเอ็ดยูเคชั่น พ.ศ. 2530
3. Microsoft Inc., "Disk Operating System", 1983
4. Microsoft Inc., "Microsoft Quick Basic V3.0 User's Guide", 1985
5. Microsoft Inc., "Microsoft Quick Basic V3.0 Compiler", 1985
6. Robert Lafore, "Assembly Language Primer for IBM PC/XT", 1984
7. Texas Instrument Inc., "Optoelectronic Data Book", 1983-1984
8. National Semiconductor Corporation, "Logic Databook", 1981