เครื่องตัดตราสัญลักษณ์

LABEL CUTTER

โดย

นายชยพล เจริญกิจชัยชนะ

นายฐิติรัช นกบิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เครื่องตัดตราสัญลักษณ์
LABEL CUTTER

โดย

นายชยพล เจริญกิจชัยชนะ รหัส 42010073
นายฐิติรัช นกบิน รหัส 42010097

อาจารย์ที่ปรึกษา
รศ.ดร. สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

เรื่อง เครื่องตัดตราสัญลักษณ์

ผู้จัดทำ

1. นาย ชยพล เจริญกิจชัยชนะ	รหัส 42010073	4C
2. นาย ฐิติรัช นกบิน	รหัส 42010097	4C

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

ลงชื่อ.....(อาจารย์ที่ปรึกษา)

(รศ.ดร. สุรพันธ์ เอื้อไพบูลย์)

ปริญญานิพนธ์ ปีการศึกษา 2545

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องตัดตราสัญลักษณ์

LABEL CUTTER

ผู้จัดทำ

1. นาย ชยพล เจริญกิจชัยชนะ รหัส 42010073 4C
2. นาย ฐิติรัช นกบิน รหัส 42010097 4C

ลงชื่อ.....(อาจารย์ที่ปรึกษา)

(รศ.ดร. สุรพันธ์ เอื้อไพบูลย์)

กิตติกรรมประกาศ

ผู้จัดทำขอขอบคุณ บิดา มารดา เป็นอย่างยิ่งที่เลี้ยงดูและส่งเสียให้สามารถเรียนมาจนถึงบัดนี้ได้ ขอขอบคุณอาจารย์ทุกท่านที่ได้อบรมและถ่ายทอดความรู้ต่างๆ เพื่อนำมาใช้ประโยชน์ทั้งในปัจจุบันและอนาคต

ขอขอบคุณรองศาสตราจารย์ ดร. สุรพันธ์ เอื้อไพบูลย์ ที่คอยเอาใจใส่ คอยให้คำแนะนำ ชี้แจงข้อบกพร่องต่าง ๆ รวมทั้งจัดหาอุปกรณ์ต่าง ๆ ที่ใช้ในโครงงาน และเป็นที่ปรึกษาการทำโครงงานนี้จนประสบความสำเร็จ

ขอขอบคุณพี่ ๆ เพื่อน ๆ และน้อง ๆ ที่คอยเป็นกำลังใจตลอดมา และช่วยเหลือในเรื่องต่าง ๆ แม้แต่การเคลื่อนย้ายอุปกรณ์ จนทำให้โครงงานนี้สามารถสำเร็จลุล่วงได้ด้วยดี

ชยพล เจริญกิจชัยชนะ
.....
นายชยพล เจริญกิจชัยชนะ

ฐิติรัช นกบิน
.....
นายฐิติรัช นกบิน

เครื่องตัดตราสัญลักษณ์

นายชยพล เจริญกิจชัยชนะ รหัส 42010073
นายฐิติรัช นกบิน รหัส 42010097
รศ.ดร. สุรพันธ์ เอื้อไพบูลย์ (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2545

บทคัดย่อ

เครื่องตัดตราสัญลักษณ์ที่นำเสนอในโครงงานนี้ ใช้การเขียนโปรแกรมด้วย Borland C++ Builder ร่วมกับการประมวลผลทางภาพ เช่น การค้นหาตำแหน่งของตราสัญลักษณ์ เพื่อนำไปควบคุมเครื่องพล็อตเตอร์ที่ทำหน้าที่ตัดตราสัญลักษณ์ในทางอุตสาหกรรม ทำให้เครื่องมีความสามารถในการตัดตราสัญลักษณ์ได้ถูกต้องแม้ว่าการป้อนตราสัญลักษณ์จะคลาดเคลื่อนไปก็ตาม

เครื่องพล็อตเตอร์ที่ใช้ในโครงงานนี้แบ่งเป็น เครื่องพล็อตเตอร์สำเร็จรูป และเครื่องพล็อตเตอร์แบบสร้างเองโดยใช้ไมโครคอนโทรลเลอร์ เอ็มซีเอส-51 ควบคุม และทำการจำลองการตัดตราสัญลักษณ์โดยการพล็อต ซึ่งการนำไปใช้งานจริงนั้นจะใช้การตัดตราสัญลักษณ์ด้วยหัวตัดเลเซอร์ อย่างไรก็ตาม หลักการต่าง ๆ ในโครงงานนี้สามารถนำไปไปประยุกต์ใช้งานได้

LABEL CUTTING

Mr. Chayapol Charoenkitchaichana (ID. 42010073)
Mr. Thitirach Nokbin (ID. 42010097)
Assoc. Prof. Dr. Surapan Airphaiboon (Advisor)
Academic Year 2002

ABSTRACT

This project presents Label Cutter which is controlled by Borland C++ Builder programming and Image Processing such as marks' position finding for controlling plotter in industrial. This plotter can cut correctly despite missing labels feeding.

Two types of plotter are used in this project. One is Intelligence Plotter and the other is a Plotter which is created by using Microcontroller MCS-51. Both of them simulate label cutting by pen plotting instead laser cutting that is used in the industrial. However, many processes can be applied for the suitable job.

สารบัญ

สารบัญภาพ

บทที่ 1
บทนำ (Introduction)

รายงานฉบับนี้ได้อธิบายถึงหลักการทำงานของเครื่องตัดตราสัญลักษณ์ การเขียนโปรแกรมคอมพิวเตอร์บอร์แลนด์ ซีพลัสพลัส บิลเดอร์ (Borland C++ Builder) ซึ่งเนื้อหาในรายงานฉบับนี้ได้แบ่งออกเป็นบทต่าง ๆ โดยในแต่ละบทจะมีเนื้อหาที่มีความเกี่ยวข้องสัมพันธ์กัน เริ่มตั้งแต่ทฤษฎีที่ใช้ , การออกแบบและแผนผังลำดับงาน (Flow Chart) , การทดลองและทดสอบประสิทธิภาพความแม่นยำของเครื่องตัดตราสัญลักษณ์ รวมไปถึงการสรุปและวิจารณ์โครงงานนี้

1.1 รายละเอียดโครงงานโดยย่อ

เครื่องตัดตราสัญลักษณ์ที่ใช้ในงานอุตสาหกรรมนั้น จะมีโครงสร้างและการทำงานเหมือนกับเครื่องพล็อตเตอร์ (Plotter) โดยผืนผ้าที่จะถูกนำมาตัดตราสัญลักษณ์นั้นจะมีตราสัญลักษณ์อยู่มากมาย ซึ่งแต่ละตราสัญลักษณ์จะมีเครื่องหมาย (Mark) อยู่ 3 ตำแหน่งในทุก ๆ ตราสัญลักษณ์ ซึ่งตำแหน่ง 3 ตำแหน่งนี้เองที่จะถูกใช้ในการอ้างตำแหน่งที่ถูกต้องในการตัดตราสัญลักษณ์

สำหรับโครงงานเครื่องตัดตราสัญลักษณ์นี้จะประกอบด้วย ส่วนของการเขียนโปรแกรมคอมพิวเตอร์บอร์แลนด์ ซีพลัสพลัส บิลเดอร์ (Borland C++ Builder) ซึ่งจะทำหน้าที่รับสัญญาณภาพจากกล้องกลับเข้าไปประมวลผลทางภาพ ซึ่งมีกระบวนการหลายกระบวนการที่ช่วยให้การตัดตราสัญลักษณ์นี้สมบูรณ์ขึ้น อันได้แก่ การหาค่าเทรชโฮล (Threshold) , การลดสัญญาณรบกวน (Noise Reduction) และการคำนวณหาจุดกึ่งกลางของเครื่องหมาย(Mark) จากนั้นจะนำค่าที่ได้ไปประมวลผลเพื่อปรับตำแหน่งของการตัดให้เหมาะสมกับตำแหน่งของตราสัญลักษณ์ที่จะถูกตัด และโปรแกรมจะส่งค่าที่ประมวลผลแล้วไปควบคุมเครื่องพล็อตเตอร์ที่ใช้ในการตัดตราสัญลักษณ์ผ่านทางพอร์ตของเครื่องคอมพิวเตอร์ เพื่อให้สามารถตัดตราสัญลักษณ์ได้ถูกต้อง โดยเครื่องพล็อตเตอร์ที่ใช้ในโครงงานนี้แบ่งเป็น 2 ชนิด คือ เครื่องพล็อตเตอร์สำเร็จรูปซึ่งใช้ในโรงงานอุตสาหกรรม (MUTOH Intelligence Plotter XF-302R) และเครื่องพล็อตเตอร์ที่สร้างขึ้นเองโดยใช้ไมโครคอนโทรลเลอร์ เอ็มซีเอส-51 (Microcontroller MCS-51) ในการควบคุมการทำงานร่วมกับเครื่องขับสเต็ปปิ้งมอเตอร์ (Stepping Motor Driver)

1.2 วัตถุประสงค์โครงงาน

สามารถเขียนโปรแกรมคอมพิวเตอร์บอร์แลนด์ ซีพลัสพลัส บิลเดอร์ (Borland C++ Builder) และไมโครคอนโทรลเลอร์ เอ็มซีเอส-51 (Microcontroller MCS-51) ร่วมกับการประมวลผลทางภาพเพื่อควบคุมเครื่องพล็อตเตอร์ที่ใช้ในการตัดตราสัญลักษณ์ได้อย่างถูกต้อง แม้ว่าการวางตราสัญลักษณ์ที่ต้องการตัดจะมีความผิดพลาดก็ตาม และสามารถนำโครงงานนี้ไปประยุกต์ใช้ในงานอุตสาหกรรมได้จริง

1.3 ขอบเขตของโครงงาน

โครงงานเครื่องตัดตราสัญลักษณ์นี้ จะสามารถจำลองการตัดตราสัญลักษณ์ได้อย่างถูกต้อง แม้ว่าตราสัญลักษณ์ที่ป้อนเข้ามาจะมีความคลาดเคลื่อนไป ซึ่งถ้าหากต้องการนำเครื่องตัดตราสัญลักษณ์นี้ไปใช้งานจริงในงานอุตสาหกรรม ก็สามารถใช้หลักการในโครงงานนี้เป็นแนวทางในการปฏิบัติงานได้

1.4 ประโยชน์หรือว่าผลที่คาดว่าจะได้รับ

โครงงานเครื่องตัดตราสัญลักษณ์นี้อาศัยการทำงานร่วมกันระหว่างโปรแกรมคอมพิวเตอร์บอร์แลนด์ ซีพลัสพลัส บิลเดอร์ (Borland C++ Builder) , ไมโครคอนโทรลเลอร์ เอ็มซีเอส-51 (Microcontroller MCS-51) , เครื่องพล็อตเตอร์ , กล้องวีดีโอ โดยการติดต่อผ่านทางพอร์ทของเครื่องคอมพิวเตอร์ ซึ่งมีหลักการและรายละเอียดต่าง ๆ มากมาย ทำให้ทราบข้อมูลด้านซอฟท์แวร์ (Software) และด้านฮาร์ดแวร์ (Hardware) เพิ่มขึ้น และสามารถนำความรู้ต่างๆที่ได้นี้ไปประยุกต์ใช้งานให้เกิดประโยชน์ต่อไปได้

บทที่ 2
ทฤษฎี

2.1 เครื่องตัดตราสัญลักษณ์และภาษา HPGL (Hewlett-Packard Graphic Language)

เครื่องตัดตราสัญลักษณ์ (Label Cutter) ที่ใช้ปกติในโรงงานอุตสาหกรรม จะมีลักษณะเหมือนกับเครื่องพล็อตเตอร์ (Plotter) ทั่วไป ๆ คือ เครื่องจะถูกควบคุมโดยคำสั่งภาษา HPGL ซึ่งลักษณะของการป้อนตำแหน่งจะเป็นแบบพิกัดเชิงตั้งฉาก (Rectangular Form) ในรูปแบบของคู่ลำดับ (X,Y) โดยมีคำสั่งที่ใช้งานหลัก ๆ ดังนี้

VS velocity,pen number; หมายถึง คำสั่งตั้งค่าความเร็วของการเคลื่อนที่ (cm/s)

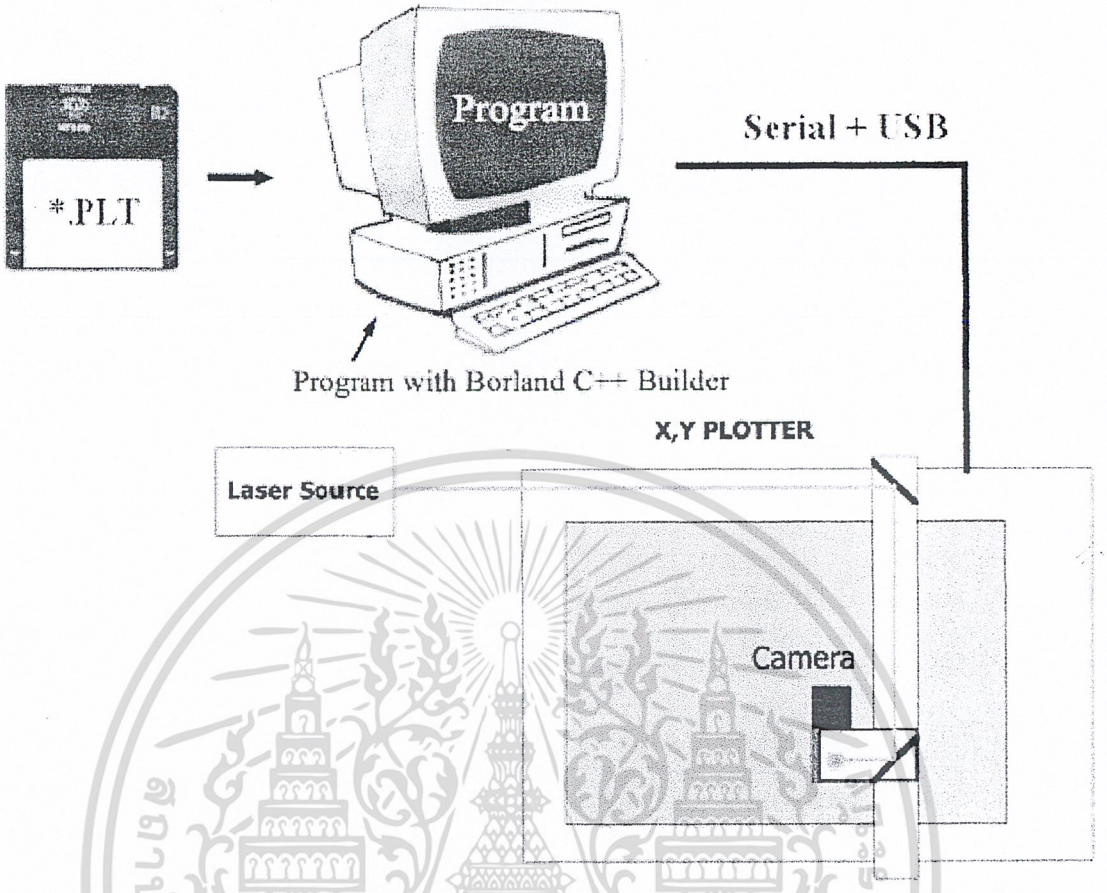
SP pen number; หมายถึง คำสั่งในการเลือกปากกาที่ต้องการ

PU X,Y; หมายถึง ให้เครื่องเคลื่อนที่แบบยกปากกาขึ้น (Pen Up) และไปที่ตำแหน่ง X, Y

PD X,Y; หมายถึง ให้เครื่องเคลื่อนที่แบบวางปากกาลง (Pen Down) และไปที่ตำแหน่ง X, Y

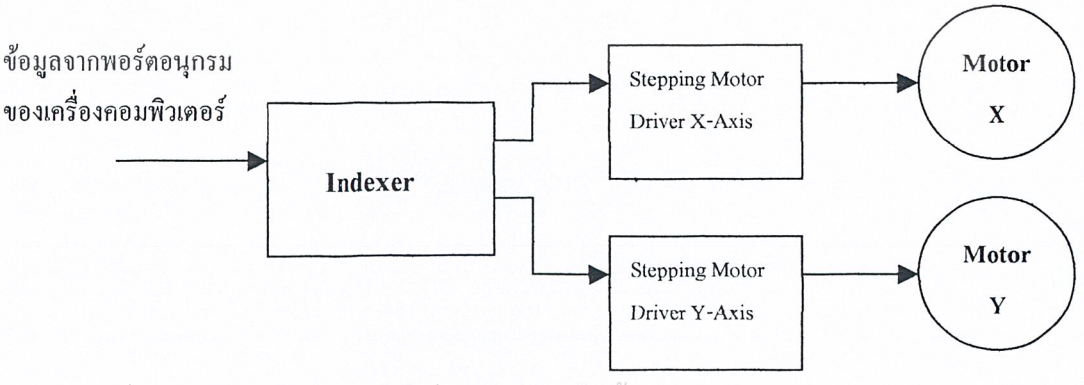
ดังนั้นไฟล์ที่นำมาใช้งาน หรือไฟล์ข้อมูลของตราสัญลักษณ์ (Label) ที่ต้องการตัดจะต้องอยู่ในรูปของไฟล์ HPGL หรือ *.plt ซึ่งจะประกอบไปด้วยพิกัดต่าง ๆ ในรูปของคู่ลำดับ X,Y รวมกันเป็นรูปของตราสัญลักษณ์ วิธีการที่สะดวกในการสร้างไฟล์ HPGL คือ การใช้โปรแกรม Corel Draw ในการสร้างไฟล์แล้วให้การเก็บไฟล์เป็น (Save as) ไฟล์ *.plt โปรแกรมจะทำการเก็บไฟล์ในรูปแบบของไฟล์ HPGL

สำหรับเครื่องพล็อตเตอร์ที่ใช้ในโครงงานนี้ ประกอบด้วยเครื่อง INTELLIGENT PLOTTER XF-300 Series ยี่ห้อ MUTOH ซึ่งจะใช้การควบคุมเครื่องด้วยภาษา MHGL ซึ่งมีลักษณะคล้าย ๆ กับภาษา HPGL มีเพียงบางคำสั่งเท่านั้นที่ไม่เหมือนกัน ให้ขั้นตอนของการเปิดไฟล์ HPGL ขึ้นมา จึงจำเป็นต้องตัดส่วนของคำสั่งที่ไม่มีในภาษา MHGL ออกไป เพื่อไม่ให้เกิดการทำงานที่ผิดพลาด



รูปที่ 2.1 แสดงลักษณะของเครื่องตัดตราสัญลักษณ์

ส่วนเครื่องพล็อตเตอร์อีกเครื่องจะมีโครงสร้างดังในรูปที่ 2.2 โดยในส่วนของ Indexer ซึ่งเป็นส่วนที่ทำหน้าที่ในการควบคุมทิศทาง, ความเร็วและลักษณะการพล็อต จะควบคุมด้วยไมโครคอนโทรลเลอร์เบอร์ในตระกูล MCS-51 ร่วมกับการทำงานของเครื่องขับสเต็ปปิงมอเตอร์ (Stepping Motor Driver)



รูปที่ 2.2 แสดงโครงสร้างของเครื่องพล็อตเตอร์

2.2 การเทรชโฮลดิ้ง (Thresholding)

การเทรชโฮลดิ้ง (Thresholding) เป็นการทำ segmentation ที่ง่ายที่สุด

$$g(i,j) = \begin{cases} 1 & for \quad f(i,j) \geq D \\ 0 & for \quad f(i,j) < D \end{cases}$$

สมมุติว่าวัตถุมีค่าของพิกเซลที่ต่างจากพื้นหลัง ภาพเอาท์พุตจะประกอบด้วยพื้นหลังที่เป็นค่า 0 และวัตถุที่มีค่าเป็น 1

ในบางกรณี ถ้ามีวัตถุหลาย ๆ ชิ้น และเราต้องการทำการเทรชโฮลค่าหลาย ๆ ค่า เราสามารถที่จะแบ่งระดับของการเทรชโฮลออกเป็นหลาย ๆ ระดับได้โดยการใช้ D_1, D_2, D_3 ดังนี้

$$g(i,j) = \begin{cases} 1 & \text{for} \quad f(i,j) \in D_1 \\ 2 & \text{for} \quad f(i,j) \in D_2 \\ 3 & \text{for} \quad f(i,j) \in D_3 \\ 4 & \text{for} \quad f(i,j) \in D_4 \\ \vdots & \\ n & \text{for} \quad f(i,j) \in D_n \\ 0 & \text{otherwise} \end{cases}$$

2.2.1 การเลือกค่าเทรชโฮล

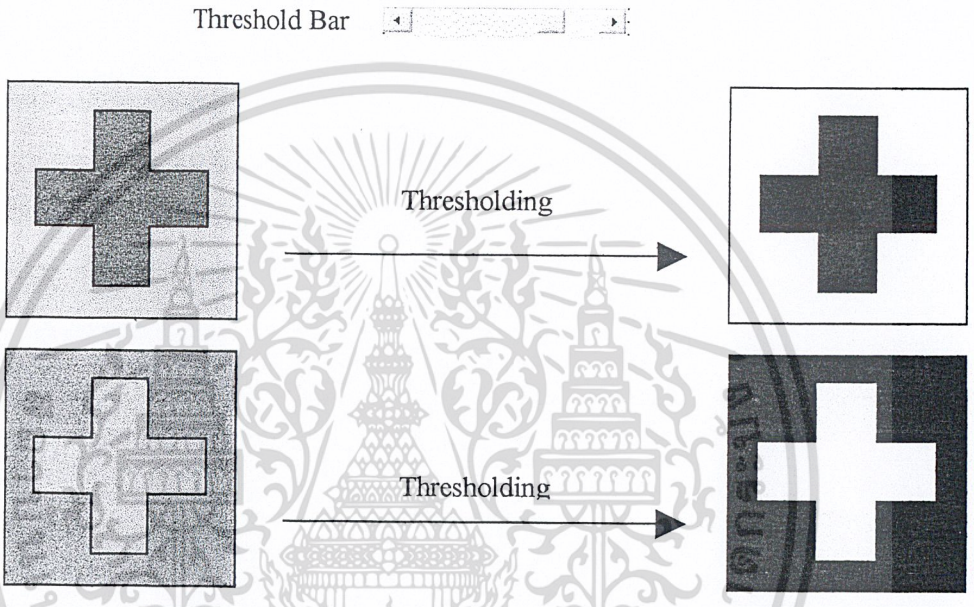
การเลือกค่าเทรชโฮลให้ได้เอาท์พุตที่ถูกต้อง แบ่งได้ 2 วิธี

1. ผู้ใช้เป็นผู้กำหนดค่าเอง (Manual)
2. การวิเคราะห์ภาพโดยอัลกอลิทึม (Automatic)

ในโครงงานนี้ ใช้การเลือกค่าเทรชโฮลแบบแรก คือ แบบผู้ใช้เป็นผู้กำหนดค่าเอง ซึ่งให้ค่าของเทรชโฮลเปลี่ยนแปลงไปตามสกอบาร์ (Scrollbar)

การเทรชโฮลโดยค่าเพียงค่าเดียวคือ D (ใช้กับทุกพิเซลในภาพ) จะเรียกว่า โกลบอลเทรชโฮล (global threshold) ซึ่งจะใช้ในกรณีที่รู้ฮิสโทแกรม (Histogram) ของทั้งภาพ ภาพที่ผ่านการเทรชโฮลจะเหลือสีเพียง 2 สีเท่านั้น วิธีนี้นิยมใช้เพราะว่าง่ายแต่ในทางปฏิบัติ การทำเทรชโฮลแบบนี้ ภาพจะต้องเป็นภาพที่มีคุณภาพดีมาก คือมีการควบคุมสภาพแวดล้อมที่จะแคปเจอร์ (Capture) ภาพเป็นอย่างดี ซึ่งจะได้ภาพที่มีคุณภาพสูง มีฮิสโทแกรมที่แยกกันอย่างชัดเจน แต่ในความเป็นจริง การแคปเจอร์ภาพจะไม่สามารถควบคุมได้ ทำให้ฮิสโทแกรมซับซ้อน ระดับความเข้ม

สีของวัตถุกับพื้นหลังจะเหลื่อมล้ำกันมาก เพราะฉะนั้นจึงต้องใช้เทรชโฮลที่มีค่าแตกต่างกันไปตามตำแหน่งของภาพ ดังนั้นจึงมีการกระทำการเทรชโฮลแบบตัวแปร (variable threshold) ซึ่งจะทำการแบ่งภาพออกเป็นภาพย่อย ๆ และมีการกำหนดเทรชโฮลในแต่ละภาพย่อย และทำการพิจารณาภาพย่อยต่าง ๆ นั้นแยกออกจากกัน



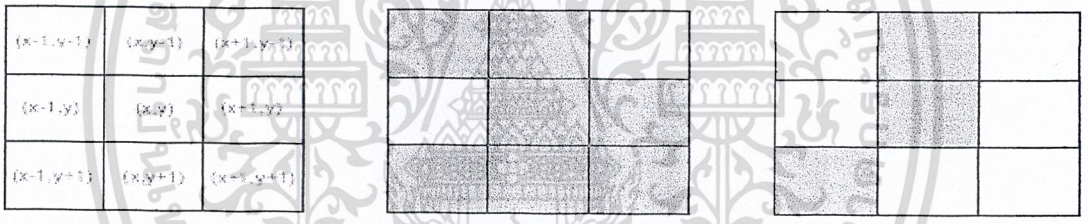
รูปที่ 2.3 แสดงรูปของการเทรชโฮล (Thresholding)

2.3 การลดสิ่งรบกวน (Noise Reduction)

การลดสิ่งรบกวน (Noise Reduction) จะเป็นการลดสิ่งที่เราไม่ต้องการออกจากภาพ ซึ่งในโปรเจคนี้ เป็นขั้นตอนที่ทำต่อจากการเทรชโฮลดิ้ง (Thresholding) ภาพแล้ว การลดสิ่งรบกวนเป็นสิ่งจำเป็นที่ต้องทำ เนื่องจากเรามีขั้นตอนที่ต้องการหาตำแหน่งจุดศูนย์กลางของเครื่องหมายบ่งบอกตำแหน่ง (Mark) ซึ่งตำแหน่งจุดศูนย์กลางของเครื่องหมาย (Mark) นี้จะผิดเพี้ยนไป ถ้าในภาพที่ทำการเทรชโฮลมาแล้วมีสิ่งรบกวนมาก ดังนั้น เราจึงจำเป็นต้องทำกระบวนการลดสิ่งรบกวนนี้

ในโปรเจคนี้ ภาพที่ได้จากการเทรชโฮล จะถูกกำหนดให้เหลือเพียง 2 สี คือสีขาว และสีดำ ซึ่งสีขาวและสีดำนี้จะถูกเก็บไว้ในแต่ละพิกเซลของรูปภาพที่แตกต่างกัน ในการลดสิ่งรบกวนของรูปภาพในโปรเจคนี้ จำเป็นต้องอาศัยการเข้าถึงพิกเซลของคำสั่งในโปรแกรม หลักการในการลดสิ่งรบกวนนี้ เริ่มต้นโดยการเลื่อนพิกเซลตามแกนนอนและแกนตั้งแต่ละพิกเซลไปเรื่อย ๆ จนครบทั้งภาพ เช่น ถ้าภาพมีขนาด 60x60 เราจะต้องทำการเลื่อนพิกเซลไปเรื่อย ๆ ทั้งหมด 3600 พิก

เซล ซึ่งแต่ละพิกเซลที่ทำการตรวจสอบนั้นจะทำการเปรียบเทียบสีกับพิกเซล 8 ตำแหน่งรอบตัวของมันเอง ถ้าพิกเซลที่ทำการตรวจสอบมีสีเหมือนกับพิกเซล 8 พิกเซลรอบตัวมันตั้งแต่ 4 พิกเซลขึ้นไป แสดงว่าพิกเซลที่ทำการตรวจสอบนั้นไม่ใช่สิ่งรบกวน แต่ถ้าพิกเซลที่ทำการตรวจสอบนั้นมีสีเหมือนกับพิกเซล 8 พิกเซลรอบตัวมันน้อยกว่า 4 พิกเซล แสดงว่า พิกเซลที่ทำการตรวจสอบนั้นเป็นสิ่งรบกวนซึ่งเราไม่ต้องการ เราจะทำการลดสิ่งรบกวนนี้โดยการเปลี่ยนสีให้กลายเป็นอีกสีหนึ่ง (ดังที่กล่าวข้างต้นว่า ภาพที่ผ่านการเทรชโฮลมาแล้วมาเหลือสีเพียง 2 สี เราจะทำการเปลี่ยนสีที่เป็นสิ่งรบกวน ให้กลายเป็นสีอีกสีหนึ่งแทน) ตัวอย่างเช่น ถ้าทราบว่าพิกเซลที่ตรวจสอบเป็นสิ่งรบกวนและมีสีดำ เราจะทำการเปลี่ยนให้พิกเซลนั้น กลายเป็นพิกเซลที่มีสีขาวแทน ตรวจสอบพิกเซลเช่นนี้ทั้งหมด 3600 พิกเซล เราก็จะได้ภาพที่มีสิ่งรบกวนน้อยลงหรือไม่มีสิ่งรบกวนเหลืออยู่เลย ซึ่งกระบวนการลดสิ่งรบกวน สามารถแสดงได้ดังรูป

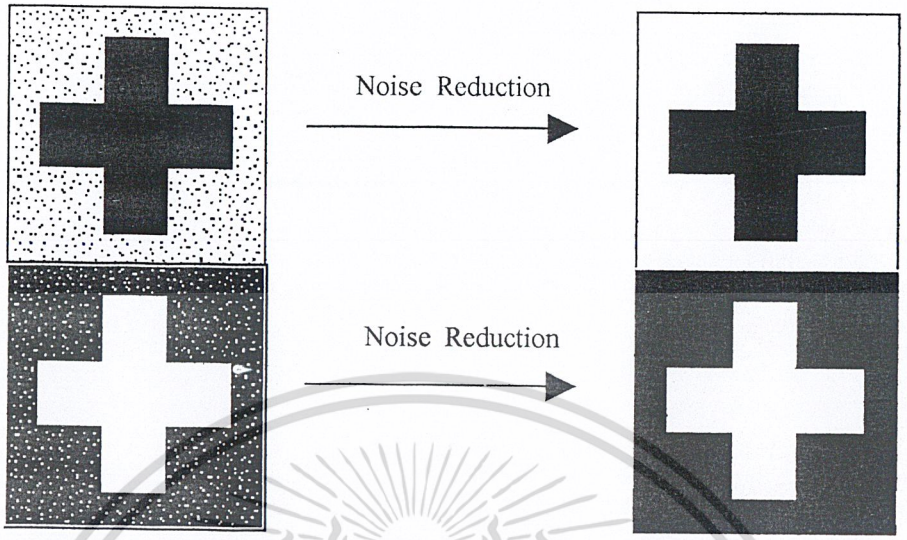


2.4ก **2.4ข** **2.4ค**

รูปที่ 2.4ก แสดงพิกัด 8 พิกัดรอบข้างของพิกเซล (x,y) ซึ่งเป็นพิกเซลที่ต้องการทำการ

รูปที่ 2.4ข แสดงให้เห็นว่าพิกัด (x,y) ซึ่งมีตำแหน่งอยู่ที่พิกเซลตรงกลางของรูป มีสีเหมือนรอบข้างมันถึง 6 พิกเซล ดังนั้น พิกัด (x,y) ที่ทำการตรวจสอบนี้ไม่ใช่สิ่งรบกวน

รูปที่ 2.4ค แสดงให้เห็นว่าพิกัด (x,y) ซึ่งมีตำแหน่งอยู่ที่พิกเซลตรงกลางของรูป มีสีเหมือนรอบข้างมันเพียง 2 พิกเซล ดังนั้น พิกัด (x,y) ที่ทำการตรวจสอบนี้เป็นสิ่งรบกวน



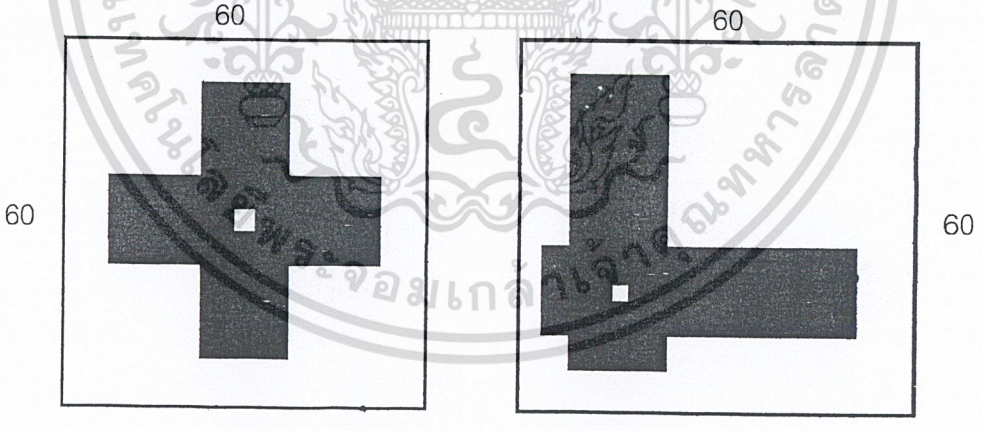
รูปที่ 2.5 แสดงรูปของการลดสิ่งรบกวน (Noise Reduction)

2.4 การหาตำแหน่งจุดศูนย์กลางเครื่องหมาย (Mark's center point finding)

การหาตำแหน่งจุดศูนย์กลางเครื่องหมาย (Mark's center point finding) เป็นขั้นตอนที่ทำหลังจากการเทรชโฮลดิ้ง (Thresholding) และการลดสิ่งรบกวน (Noise Reduction) แล้ว การหาตำแน่งจุดศูนย์กลางของเครื่องหมาย (Mark) เป็นสิ่งจำเป็นในโปรเจคนี้ เนื่องจาก เราจะใช้ตำแหน่งจุดศูนย์กลางเครื่องหมายที่หามาได้นี้เป็นตำแหน่งอ้างอิง ซึ่งเราจะใช้ประโยชน์จากตำแหน่งอ้างอิงที่หามาได้นี้หลายประการด้วยกัน เช่น การหาระยะห่างระหว่างรูปภาพรูปที่หนึ่งกับรูปภาพรูปต่อไป, การบ่งบอกว่า ภาพที่วางอยู่นี้ วางทำมุมผิดเพี้ยนไปจากมุมอ้างอิง (0 องศา) เป็นมุมเท่าไร เมื่อเราทราบค่าต่าง ๆ เหล่านี้แล้ว เราสามารถที่จะสั่งให้เครื่องพล็อตเตอร์ (Plotter) ทำการพล็อตภาพให้ได้เสมือนกับภาพวางตรงได้ และมีความแม่นยำมากขึ้น ซึ่งรายละเอียดของขั้นตอนเหล่านี้ จะกล่าวถึงในหัวข้อต่อ ๆ ไป

การหาตำแหน่งจุดศูนย์กลางของเครื่องหมายในการทำโปรเจคนี้ สามารถทำได้โดยอาศัยหลักการที่เรียกว่า โปรเจคชัน โปรไฟล์(Projection Profile) ซึ่งมีหลักการ คือ เป็นการเก็บข้อมูลของแกนสองแกน คือแกนในแนวดิ่ง และแกนในแนวราบ ซึ่งเราจะทำการเลื่อนพิกเซลและเก็บข้อมูลไปเรื่อย ๆ ในโปรเจคนี้ ภาพที่ใช้มีขนาด 60x60 และเป็นภาพเครื่องหมายกากบาท เราจะต้องทำการเลื่อนพิกเซลไปทีละแนว คือเลื่อนพิกเซลตามแนวราบให้ครบทั้ง 60 พิกเซลก่อน แล้วเก็บค่าข้อมูลที่ได้ไว้ในอาร์เร่ย์ตัวหนึ่ง (ในที่นี้ ข้อมูลคือจำนวนพิกเซลที่มีสีตามต้องการ) แล้วจึงเลื่อนพิกเซลตามแนวดิ่งลงมาทีละพิกเซลทุกการเลื่อนพิกเซลตามแนวราบครบ 60 พิกเซล ทำการเลื่อนพิกเซลไปเช่นนี้จนพิกเซลในแนวดิ่งถูกเลื่อนไป 60 พิกเซล ซึ่งเมื่อสิ้นสุดการทำงานนี้จะมีการเลื่อนพิกเซล

ทั้งหมด 3600 ตำแหน่ง ค่าที่ถูกเก็บไว้ในอาร์เร่ย์ทั้งหมดจะนำมาหาค่าความถี่สูงสุด จากนั้น เราจะทำการเลื่อนพิกเซล 2 ทิศทางคือ เลื่อนพิกเซลไปข้างหน้าเพื่อหาตำแหน่งที่ข้อมูลมีความถี่เท่ากับความถี่สูงสุด และเลื่อนพิกเซลไปข้างหลังเพื่อหาตำแหน่งที่ข้อมูลมีความถี่เท่ากับความถี่สูงสุดเช่นเดียวกัน การเลื่อนพิกเซล 2 ทิศทางนี้จะทำให้ได้ตำแหน่ง 2 ตำแหน่งที่แตกต่างกันที่มีความถี่สูงสุดเท่ากัน เมื่อดำเนินการมาจนถึงขั้นตอนนี้ เราสามารถหาตำแหน่งจุดศูนย์กลางในแนวดิ่งได้โดยการหาค่าเฉลี่ยตำแหน่ง 2 ตำแหน่งที่ได้มาจากการเลื่อนพิกเซล 2 ทิศทางนั้น สิ่งที่เราต้องการต่อไปคือตำแหน่งจุดศูนย์กลางในแนวราบ ซึ่งสามารถหาได้โดยอาศัยหลักการและวิธีการเช่นเดียวกับการหาตำแหน่งจุดศูนย์กลางในแนวดิ่งซึ่งได้หามาแล้ว ต่างกันตรงการเลื่อนพิกเซลเริ่มแรก ในการหาตำแหน่งจุดศูนย์กลางในแนวราบนี้ เราจะต้องการทำเลื่อนพิกเซลตามแนวดิ่งให้ครบทั้ง 60 พิกเซลก่อน แล้วเก็บค่าข้อมูลที่ได้ไว้ในอาร์เร่ย์ตัวหนึ่ง แล้วจึงเลื่อนพิกเซลตามแนวราบไปทีละพิกเซลทุกการเลื่อนพิกเซลตามแนวดิ่งครบ 60 พิกเซล ทำการเลื่อนพิกเซลไปเช่นนี้จนพิกเซลในแนวราบถูกเลื่อนไป 60 พิกเซล ซึ่งเมื่อสิ้นสุดการทำงานนี้จะมีการเลื่อนพิกเซลทั้งหมด 3600 ตำแหน่งเช่นเดียวกัน จากนั้น ขั้นตอนวิธีการหาตำแหน่งจุดศูนย์กลางจะเหมือนเดิมทุกประการ เมื่อการดำเนินการสิ้นสุดลง เราจะได้ตำแหน่งจุดศูนย์กลางในแนวราบ ซึ่งเราสามารถนำตำแหน่งจุดศูนย์กลางทั้ง 2 ตำแหน่งนี้ไปใช้ประโยชน์ต่อไปได้



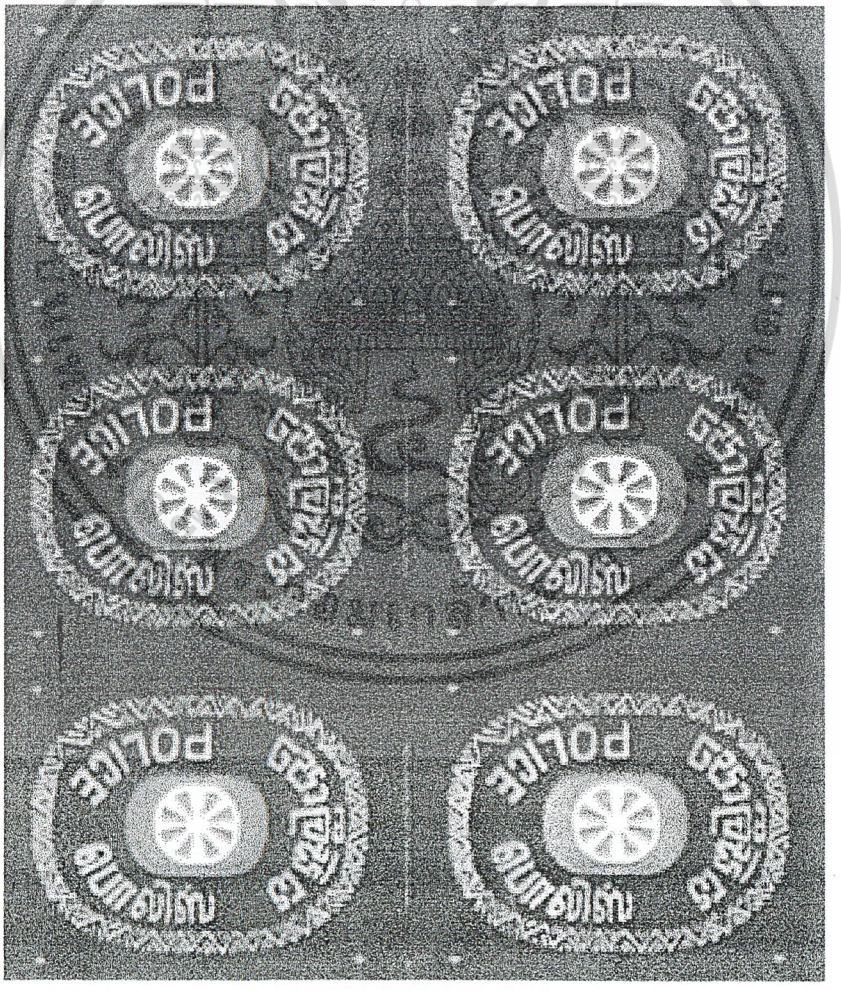
2.6ก **2.6ข**

รูปที่ 2.6ก แสดงการหาจุดศูนย์กลางของ Mark ที่ตำแหน่งกึ่งกลางปกติ (29,29)

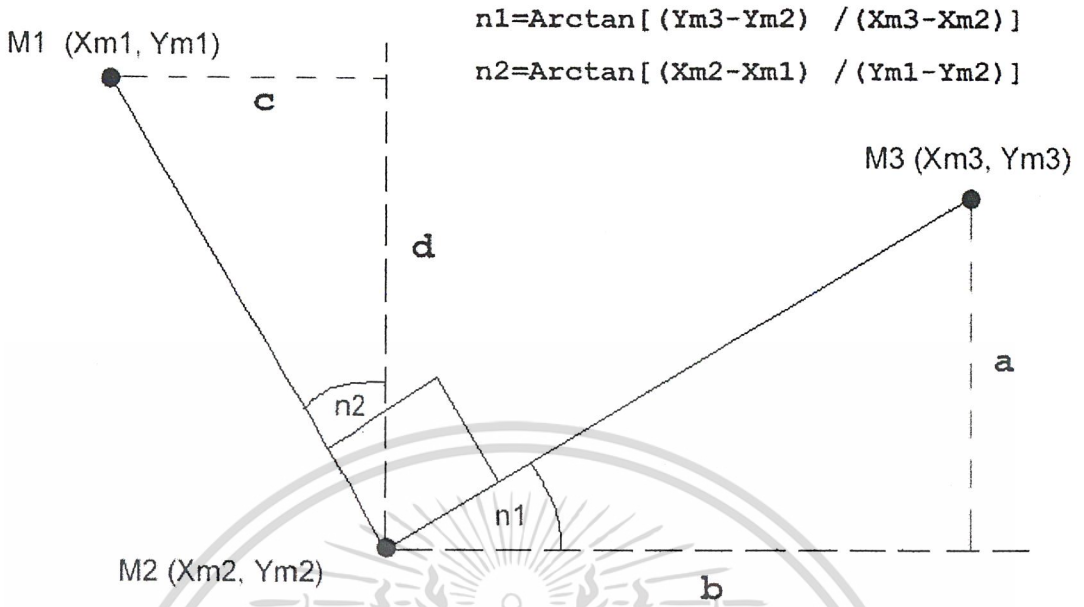
รูปที่ 2.6ข แสดงการหาจุดศูนย์กลางของ Mark ที่ตำแหน่งอื่น (13,39) ซึ่งเราสามารถนำค่าตำแหน่งนี้มาประมวลผล เพื่อเลื่อนเครื่องตัดไปยังตำแหน่งกึ่งกลางปกติ (29,29) ได้

2.5 การหาตำแหน่ง, มุมเอียง และการปรับพิกัดของการตัดตราสัญลักษณ์ (Label)

เนื่องจากการใช้งานจริงของเครื่องตัดตราสัญลักษณ์ (Label) ในระบบโรงงานอุตสาหกรรมนั้น จะใช้วิธีการตัดแยกแต่ละตราสัญลักษณ์ออกจากผ้าผืนใหญ่ที่มีตราสัญลักษณ์วางเรียงกันอยู่หลายตราสัญลักษณ์ ดังแสดงในรูปที่ 2.7 ด้วยหัวตัดแบบเลเซอร์ (Laser) ซึ่งมีความเป็นไปได้สูงว่าการตัดแยกตราสัญลักษณ์ออกมานั้นจะเกิดความผิดพลาดได้ ทั้งจากระบบการป้อนผ้า หรือการย่นของผืนผ้า จึงมีความจำเป็นที่จะต้องมีเครื่องหมาย (Mark) กำกับอยู่ในทุก ๆ ตราสัญลักษณ์ โดยแต่ละตราสัญลักษณ์จะมีเครื่องหมาย (Mark) อยู่ 3 ตำแหน่งด้วยกัน ซึ่งเครื่องหมาย (Mark) เหล่านี้เองที่จะถูกนำมาใช้ในการตรวจสอบตำแหน่งของตราสัญลักษณ์ที่จะถูกตัด แล้วนำค่าที่ได้มาเปลี่ยนแปลงพิกัด (Co-ordinate) เพื่อให้สามารถตัดตราสัญลักษณ์ได้ถูกต้องตามเดิม



รูปที่ 2.7 แสดงลักษณะของผืนผ้าที่ที่ใช้ในการตัด Label ในระบบอุตสาหกรรม



รูปที่ 2.8 แสดงการตรวจสอบตำแหน่งและมุมเอียงของตราสัญลักษณ์

ในขั้นตอนแรกของการตัดแต่ละตราสัญลักษณ์จะเริ่มด้วยการหาตำแหน่งอ้างอิงของตราสัญลักษณ์ โดยการหาตำแหน่งของเครื่องหมาย (Mark) ทั้ง 3 ตำแหน่ง จากนั้นโปรแกรมจะนำตำแหน่งที่ได้นี้มาคำนวณเพื่อตรวจสอบหาตำแหน่งและมุมเอียงของภาพได้ดังรูปที่ 2.8 ซึ่งจะเห็นว่ามุม n1 และ n2 จะมีค่าเท่ากันในกรณีปกติ (เช่น กรณีที่เครื่องหมายทั้ง 3 ตั้งฉากกันและตัวผ้า (Label) ไม่มีการย่น) จึงกำหนดให้มุมเอียง (n) ของตราสัญลักษณ์มีค่าเท่ากับค่าเฉลี่ยระหว่าง n1 และ n2 เขียนได้ดังสมการที่ (2.1)-(2.3)

$$n1 = \tan^{-1}[(Ym3-Ym2)/(Xm3-Xm2)] \quad (2.1)$$

$$n2 = \tan^{-1}[(Xm2-Xm1)/(Ym1-Ym2)] \quad (2.2)$$

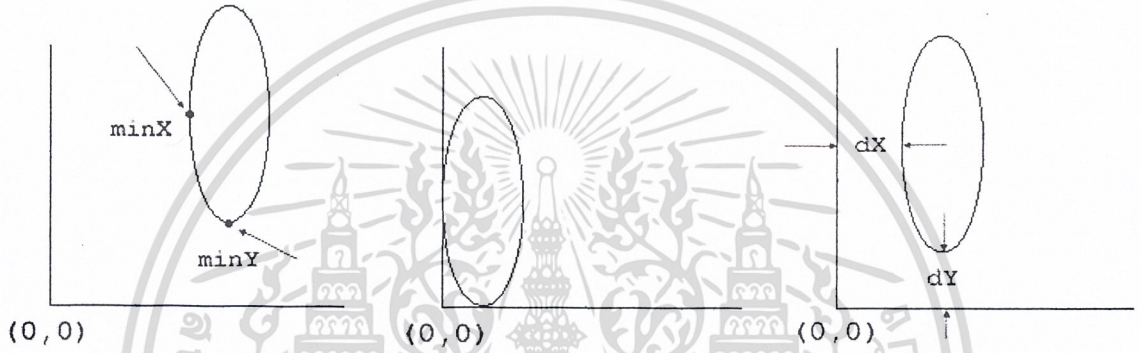
$$n = (n1 + n2) / 2 \quad (2.3)$$

มุมเอียง (n) ที่ได้นี้ จะถูกนำไปใช้ในการปรับหมุน (Rotation) พิกัดเพื่อทำให้การตัดตราสัญลักษณ์มีความถูกต้อง แต่ก่อนที่จะนำค่ามุม n ไปใช้นี้ จะต้องทำการปรับพิกัดของรูปที่เราจะตัดเสียก่อน ซึ่งการทำงานของเครื่องตัดตราสัญลักษณ์นั้นจะใช้การควบคุมด้วยไฟล์ HPGL ในขั้นแรกนั้นต้องย้ายพิกัดของรูป (Label) มาไว้ที่ตำแหน่งจุดตั้งต้น (Origin 0,0) ดังรูปที่ 2.9ก โดยการหาค่าพิกัด X และ Y ที่มีค่าน้อยที่สุด (minX, minY) มาลบออกจากทุกพิกัด จะได้ผลลัพธ์ดังรูปที่ 2.9ข จากนั้นจึงนำค่าพิกัดที่ได้มาบวกด้วยค่า dX, dY ดังรูปที่ 2.9ค ซึ่งค่า dX, dY ก็คือระยะที่น้อยที่สุด

ระยะ ของเครื่องหมายในแนวแกน X และแกน Y ดึงพิกัดตัด ตามลำดับ เป็นค่าที่ใช้อ้างอิงตำแหน่งของตราสัญลักษณ์ที่ต้องการตัดกับเครื่องหมาย Mark ทั้ง 3 ซึ่งสามารถเขียนเป็นสมการได้ดังสมการที่ (2.4) และสมการที่ (2.5)

$$X[a] = X[a] - \min X + dX \quad (2.4)$$

$$Y[a] = Y[a] - \min Y + dY \quad (2.5)$$



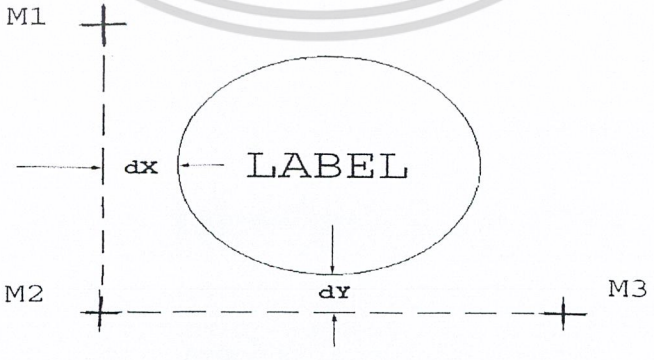
2.9ก **2.9ข** **2.9ค**

เมื่อ X[a], Y[a] เป็นพิกัดใด ๆ ในไฟล์ HPGL

รูปที่ 2.9ก แสดงค่าพิกัดเริ่มต้นของตราสัญลักษณ์ที่จะตัด

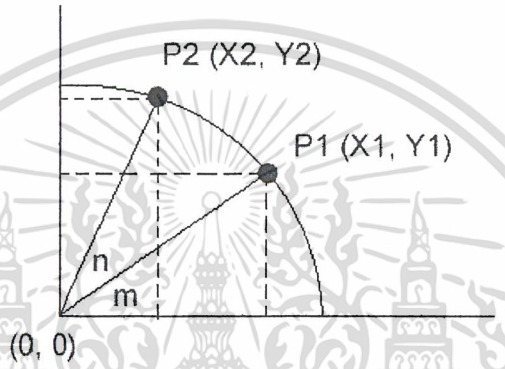
รูปที่ 2.9ข แสดงค่าพิกัดเมื่อถูกย้ายเข้ามาที่จุด Origin อ้างอิง (0,0)

รูปที่ 2.9ค แสดงค่าพิกัดเมื่อถูกปรับแต่งตามสมการที่ 2.4-2.5



รูปที่ 2.10 แสดงระยะของ dX, dY ซึ่งเป็นค่าที่ใช้อ้างอิงตำแหน่งของตราสัญลักษณ์ที่ต้องการตัดกับเครื่องหมาย Mark ทั้ง 3

ต่อไปจะเป็นขั้นตอนของการปรับหมุน (Rotation) พิกัด ซึ่งในขั้นตอนนี้จะต้องแปลงพิกัดต่าง ๆ ที่เดิมเป็นพิกัดระบบเชิงตั้งฉาก (Rectangular Form) ให้เปลี่ยนไปอยู่ในรูปพิกัดเชิงขั้ว (Polar Form) ดังในรูปที่ 2.11 ทำให้ได้ขนาดของพิกัด P ที่มุม m จากนั้นเราจะนำมุมเอียง n ที่ได้ในขั้นต้นมาบวกกับมุม m เกิดเป็นมุมเท่ากับ m+n ที่ขนาด P เท่าเดิม แสดงให้เห็นว่าเราสามารถหมุนภาพไปได้ n องศาโดยที่ภาพไม่มีการผิดรูป หรือแสดงได้ดังสมการที่ (2.6)-(2.9)



รูปที่ 2.11 แสดงการปรับหมุนพิกัด

พิกัดเดิม	P1	=	$\sqrt{(X_1^2 + Y_1^2)}$			(2.6)
	มุม	=	tan-1(Y1 / X1)	=	m	(2.7)
พิกัดใหม่	P1	=	$\sqrt{(X_1^2 + Y_1^2)}$	=	$\sqrt{(X_2^2 + Y_2^2)}$ = P2	(2.8)
	มุม	=	m + n			(2.9)

จากนั้นจึงแปลงพิกัดใหม่ที่ได้กลับมาเป็นระบบพิกัดเชิงตั้งฉากอีกครั้ง แล้วบวกด้วยพิกัดที่จุดกำเนิด (Origin 0,0) ซึ่งในที่นี้ก็คือจุด Mark2 แล้วจะได้ดังสมการที่ (2.10)-(2.11)

X2	=	P1*cos(m+n) + Xm2	(2.10)
Y2	=	P1*sin(m+n) + Ym2	(2.11)

ข้อควรระวังของการเขียนโปรแกรมคือ กรณีที่ตัวส่วนมีค่าเท่ากับ 0 ซึ่งจะทำให้โปรแกรมทำงานผิดพลาดได้ จะต้องแก้ปัญหาด้วยการแยกกรณีที่ส่วนเป็น 0 ออกมา

2.6 การตัดตราสัญลักษณ์ (Label) หลาย ๆ รูป

เนื่องจากในผืนผ้า 1 ผืน จะมีตราสัญลักษณ์ที่จะต้องถูกตัดแยกอยู่หลาย ๆ ตรา จึงมีความจำเป็นที่จะต้องวัดระยะห่างระหว่างแต่ละตราสัญลักษณ์ คือค่า mX และ mY ดังรูปที่ 2.12 เพื่อที่จะสามารถเลื่อนเครื่องตัดตราสัญลักษณ์ให้หาตำแหน่งของเครื่องหมาย (Mark) ทั้ง 3 จุดของตราสัญลักษณ์ตัวใด ๆ ในผืนผ้าด้วยความรวดเร็ว ด้วยการคำนวณด้วยสมการที่ 2.12-2.13

$$Xm[n] = Xm[0] + nx * mx \quad (2.12)$$

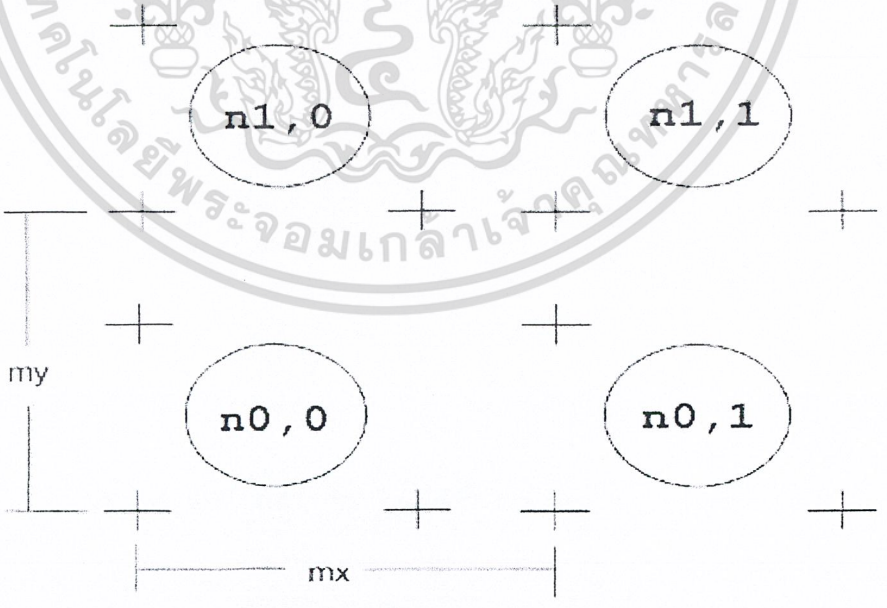
$$Ym[n] = Ym[0] + ny * my \quad (2.13)$$

เมื่อ nx = ตำแหน่ง label ทางแกน x = 0,1,2,...

ny = ตำแหน่ง label ทางแกน y = 0,1,2,...

Xm[n], Ym[n] = เป็นคู่ลำดับของ Mark ต่างๆ

Xm[0], Ym[0] = เป็นคู่ลำดับของ Mark ของตราสัญลักษณ์ตัวแรก (n0,0)



รูปที่ 2.12 แสดงระยะห่างระหว่าง Label

โดยปกติแล้ว ตราสัญลักษณ์จะถูกเรียงอยู่ในผืนผ้าอย่างมีระเบียบ จะมีจำนวนของตราสัญลักษณ์ทั้งหมดเท่ากับ จำนวนของตราสัญลักษณ์ในแกน X คูณกับจำนวนของตราสัญลักษณ์ในแกน Y ดังนั้นในขั้นตอนของการเขียนโปรแกรมจึงต้องมีส่วนของการรับค่าจำนวนของตราสัญลักษณ์ ในแกน X (nX) และจำนวนของตราสัญลักษณ์ในแกน Y (nY)

การตัดตราสัญลักษณ์แบบหลายๆ รูปจึงจำเป็นต้องเริ่มต้นด้วยการสอน (Teach) ให้โปรแกรมรู้จักตำแหน่งของเครื่องหมาย (Mark) ของตราสัญลักษณ์ตัวแรก (n0, 0) 3 ตำแหน่ง, ระยะห่างของตราสัญลักษณ์ในแกน X และ Y (mX, mY ตามลำดับ) และระยะห่างที่น้อยที่สุดจากตราสัญลักษณ์ถึงแนวแกน Y และ X ตามลำดับ รวมเป็นทั้งหมด 7 ค่าที่จะต้องทำการสอนโปรแกรมในการตัดตราสัญลักษณ์ตัวแรก

จากนั้น พล็อตเตอร์จะทำการตัดตราสัญลักษณ์ตัวแรกจนเสร็จเรียบร้อย เครื่องก็จะเลื่อนไปหาตำแหน่งของเครื่องหมาย (Mark) ทั้ง 3 ตำแหน่งของตราสัญลักษณ์ตัวถัดไปอย่างอัตโนมัติ ซึ่งในขั้นตอนนี้เอง จะใช้การประมวลผลทางภาพที่ได้กล่าวไว้ในหัวข้อที่ 2.2-2.4 มาช่วยทำการหาตำแหน่งของเครื่องหมาย (Mark) ที่ถูกต้อง แล้วจึงนำตำแหน่งที่ได้มาทำการปรับหมุนภาพให้ถูกต้อง ดังสมการที่ (2.6)-(2.9) จากนั้นจึงทำการแปลงค่ากลับสู่ระบบพิกัดเชิงตั้งฉาก ดังสมการที่ (2.10)-(2.11) เพียงเปลี่ยนค่า Xm2 , Ym2 เป็นพิกัดของ Mark2 ของตราสัญลักษณ์แต่ละตัว ก็จะสามารถตัดตราสัญลักษณ์ได้อย่างถูกต้อง การทำงานของเครื่องจะวนทำงานเช่นนี้ไปเรื่อยๆ จนกว่าการตัดตราสัญลักษณ์จะครบทุกตัว

บทที่ 3

การออกแบบและ Flow Chart

หลังจากเราทราบถึงหลักการทำงานและโปรแกรมส่วนต่างๆ ของเครื่องตัดตราสัญลักษณ์แล้ว ในบทนี้เราจะกล่าวถึงหลักการออกแบบการเขียนโปรแกรมและ Flow Chart โดยเริ่มพิจารณาดังนี้

3.1 การควบคุมพล็อตเตอร์ผ่านทางพอร์ตอนุกรม (Serial Port)

เนื่องจากเครื่องพล็อตเตอร์ที่ใช้ในโครงงานนี้สื่อสารกับเครื่องคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม (Serial Port) และในส่วนของโปรแกรม Borland C++ Builder เองนั้นไม่ได้มีคอมโพเนนท์ (Component) สำหรับการสื่อสารผ่านพอร์ตอนุกรมโดยตรง ดังนั้นจึงมีความจำเป็นที่จะต้องเปิดพอร์ตเอง โดยมีหลักการที่สำคัญดังนี้

- ชื่อของพอร์ตต้องตรงกับพอร์ตที่ใช้งาน เช่น COM1, COM2
- BAUD RATE, PARITY, WORD SIZE และ STOP BITS จะต้องตั้งให้มีค่าตรงกับอุปกรณ์ที่ต้องการทำการติดต่อสื่อสาร

โดยในโครงงานนี้ได้ทำการติดต่อเครื่องพล็อตเตอร์ผ่านทางพอร์ต COM1 และมีค่าBAUD RATE = 9600, PARITY = N, WORD SIZE = 8 bits และ STOP BIT = 1 สำหรับรายละเอียดในส่วนนี้สามารถจะดูได้ใน Win32 Developer's References หรือจากไฟล์ WIN32SDK.HLP

3.2 การรับภาพจากกล้องวีดีโอ

เครื่องตัดตราสัญลักษณ์นี้จะต้องมีการนำภาพจากกล้องวีดีโอเข้ามาในโปรแกรม เพื่อทำการประมวลผลและปรับค่าพิกัดเพื่อที่จะสามารถทำการตัดตราสัญลักษณ์ที่ถูกต้อง โดยกล้องที่นำมาใช้ในโครงงานนี้เป็นกล้องวีดีโอชนิดส่งผ่านข้อมูลผ่านพอร์ต USB(Universal Serial Bus) ซึ่งในส่วนของโปรแกรม Borland C++ Builder ก็ไม่ได้มี Component ที่สนับสนุนการรับภาพจากกล้องวีดีโอ เราจึงต้องหา Component ที่สนับสนุนการรับภาพจากกล้องวีดีโอ ซึ่งโครงงานนี้ได้เลือกใช้ Component ที่มีชื่อว่า ImageEn โดยสามารถดาวน์โหลดได้ที่ http://www.hicomponents.com

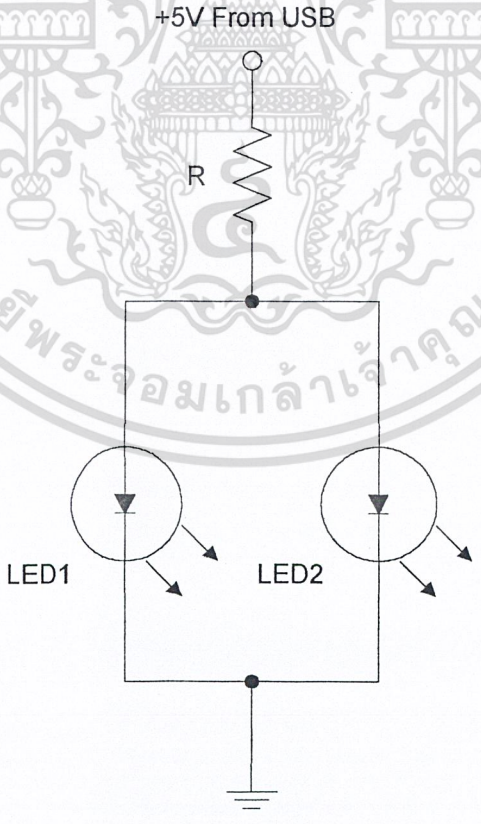
3.3 การจัดแสงสว่างให้กับกล้องวีดีโอ

ในการติดตั้งกล้องวีดีโอเข้ากับตัวพล็อตเตอร์นั้น เราจะติดในระดับที่ต่ำพอสมควร เพื่อที่จะให้ได้ภาพที่มีขนาดใหญ่ พอที่จะนำไปทำการประมวลผล จึงจำเป็นที่จะต้องมีแหล่งกำเนิดแสงให้กับกล้องวีดีโอด้วย โดยในโครงงานนี้ได้เลือกใช้หลอดแอลอีดี(LED) ที่ให้แสงสีขาวจำนวน 2 ดวง เพื่อทำให้เกิดแสงสว่างที่เพียงพอ ซึ่ง LED ที่ใช้นี้จะทำงานที่แรงดันประมาณ 3.6 โวลต์และกินกระแสประมาณ 30 มิลลิแอมป์/ดวง เมื่อเรานำ LED มาต่อกับกล้องวีดีโอซึ่งได้รับไฟเลี้ยงจาก USB เท่ากับ 5 โวลต์ แล้วเราจะสามารถต่อวงจรได้ดังรูปที่ 3.1 และคำนวณหาค่าตัวต้านทาน R ได้จากสมการที่ 3.1

$$R = V/I = (5 - 3.6)/(30m * 2) \quad (3.1)$$

$$= 23.33 \text{ โอห์ม}$$

ดังนั้นเราเลือกให้ตัวต้านทาน R = 22 โอห์ม



รูปที่ 3.1 แสดงวงจร LED ที่ให้แสงสว่างแก่กล้องวีดีโอ

3.4 การสร้างพล็อตเตอร์

เนื่องจากในโครงงานนี้ได้ใช้พล็อตเตอร์สองแบบด้วยกัน คือ พล็อตเตอร์สำเร็จรูปยี่ห้อ MUTOH และพล็อตเตอร์ที่สร้างขึ้นมาเอง ซึ่งในหัวข้อนี้จะกล่าวถึงเพียงพล็อตเตอร์ที่สร้างขึ้นมาเองเท่านั้น โดยส่วนประกอบของพล็อตเตอร์นั้นได้ถูกแสดงไว้แล้วในรูปที่ 2.2 สามารถแบ่งออกเป็นส่วนประกอบต่าง ๆ ได้ดังนี้

3.4.1 ส่วนควบคุมการพล็อต (Indexer)

สำหรับส่วนควบคุมนี้แสดงได้ดังรูปที่ 3.2 โดยใช้ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 เบอร์ 89C51 ของบริษัท Atmel เนื่องจากมีความสามารถในการรับส่งข้อมูลผ่านทางพอร์ตนุกรม (Serial Port) เพื่อติดต่อกับโปรแกรมที่เครื่องคอมพิวเตอร์ได้ และมีความไวในการทำงานที่เพียงพอ โดยจะต่อร่วมกับหน่วยความจำข้อมูล (RAM) ขนาด 8 กิโลไบต์ เบอร์ 6264 ทำหน้าที่ในการเก็บข้อมูลที่ใช้ในการพล็อต

ในการเชื่อมต่อข้อมูลแบบอนุกรมระหว่างเครื่องคอมพิวเตอร์และไมโครคอนโทรลเลอร์นั้น จะเป็นที่จะต้องใช้ไอซีเบอร์ MAX232 เพื่อทำหน้าที่ในการแปลงแรงดันให้เหมาะสม และต้องใช้ไอซีแลทช์ (Latch) เบอร์ 74LS373 เพื่อทำหน้าที่ในการแลทช์ค่าแอดเดรส (Address) เมื่อเชื่อมต่อกับหน่วยความจำข้อมูลภายนอก (External RAM)

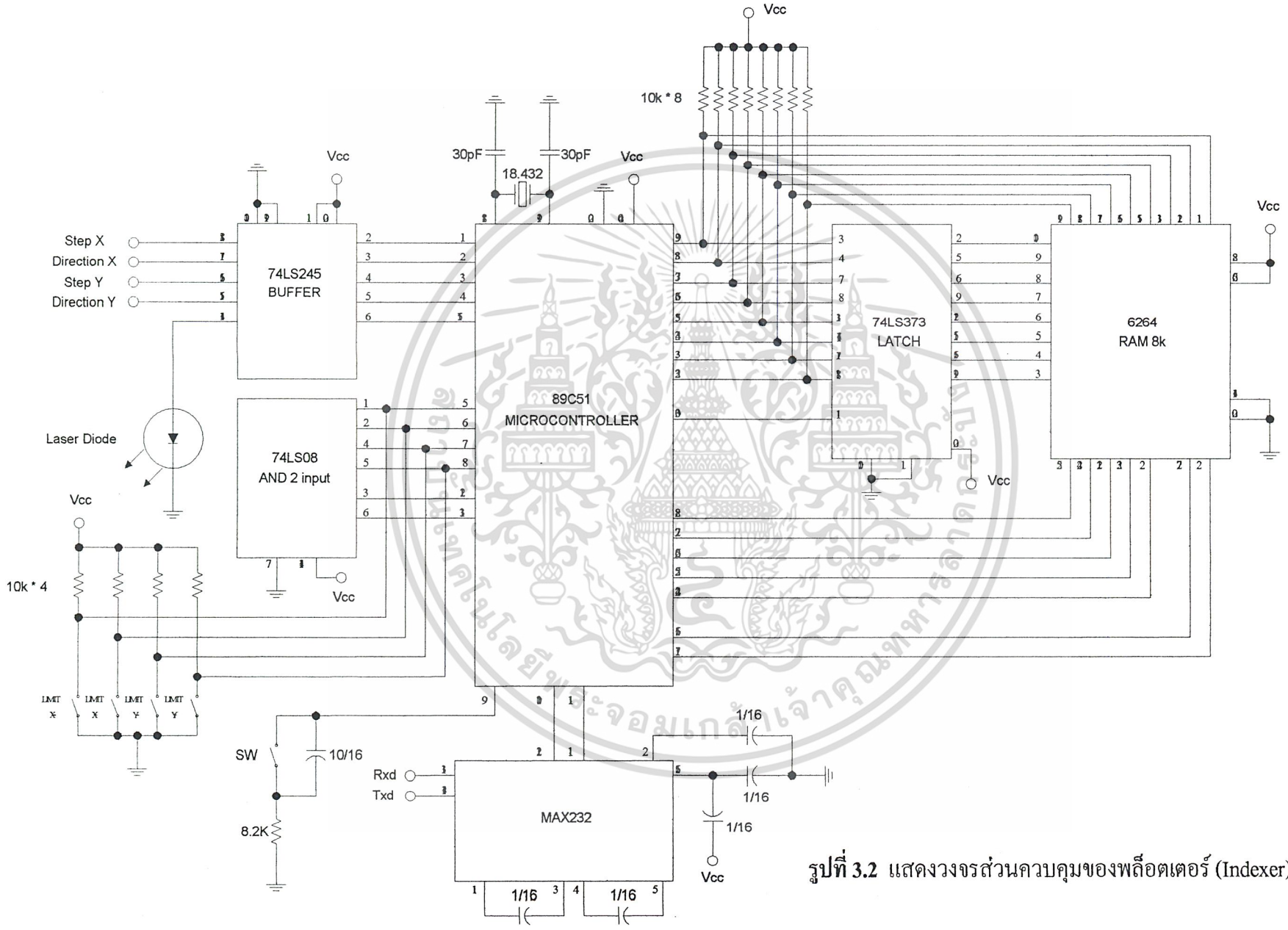
3.4.2 ส่วนขับสเต็ปปิ้งมอเตอร์ (Stepping Motor Driver)

ส่วนนี้ทำหน้าที่ในการขับสเต็ปปิ้งมอเตอร์ โดยใช้ Compumotor รุ่น OEM650 ของบริษัท Parker ซึ่งสามารถกำหนดความละเอียดของการขับได้เป็นจำนวนตำแหน่งต่อการหมุน 1 รอบ ซึ่งในโครงงานนี้จะกำหนดความละเอียดไว้ที่ 1,000 ตำแหน่ง/รอบ โดยจะถูกแบ่งออกเป็น 2 ชุด คือ ชุดของแกน X และชุดของแกน Y

ในการทำงานนั้น Compumotor ต้องการอินพุต 2 ค่า คือ อินพุตที่กำหนดทิศทาง (Direction) การเคลื่อนที่ของสเต็ปปิ้งมอเตอร์ และอินพุตที่กำหนดสเต็ปการเคลื่อนที่ กำหนดโดยการป้อนพัลส์ 1 พัลส์ ต่อการเคลื่อนที่ไป 1 สเต็ป

3.4.3 โครงของพล็อตเตอร์และสเต็ปปิ้งมอเตอร์

โครงสร้างประกอบด้วยส่วนที่สามารถเลื่อนได้ในแนวแกน X และแกน Y และมีสเต็ปปิ้งมอเตอร์ (Stepping Motor) จำนวน 2 ตัว ที่ทำหน้าในการเลื่อนส่วนที่เคลื่อนที่ในแนวแกน X และแกน Y รวมไปถึงลิมิตสวิตช์ (Limit Switch) จำนวน 4 ตัว ที่ทำหน้าที่ในการกำหนดขอบเขตในการพล็อต



รูปที่ 3.2 แสดงวงจรส่วนควบคุมของพล็อตเตอร์ (Indexer)

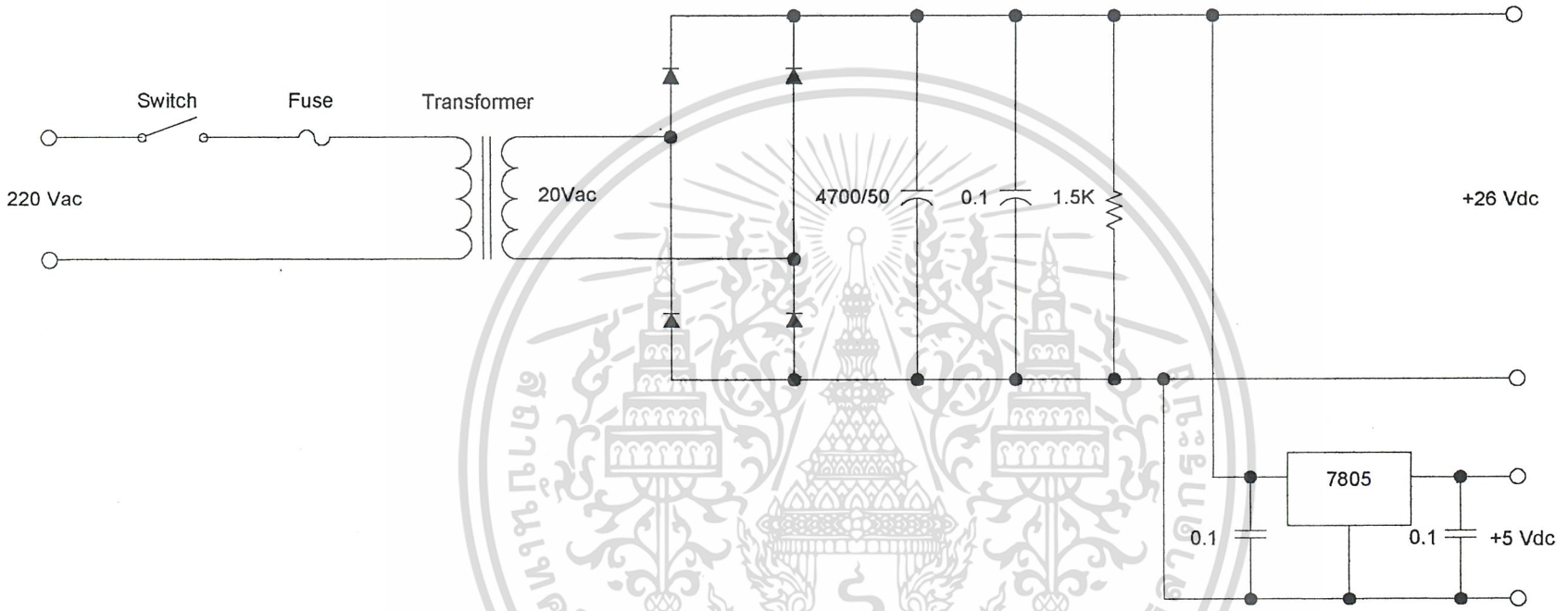
3.4.4 ภาคจ่ายไฟ

แรงดันที่ใช้ในพล็อตเตอร์จะแบ่งออกเป็น 2 ชุด โดยชุดแรกจะเป็นแรงดันที่ป้อนให้กับ Compumotor ซึ่งสามารถใช้แรงดันได้ในช่วง 24-75 โวลท์ ส่วนอีกชุดจะเป็นแรงดันที่ป้อนให้กับไอซีในส่วนควบคุม (Indexer) ซึ่งใช้แรงดัน 5 โวลท์ เราจึงเลือกใช้หม้อแปลงที่ให้แรงดัน 20 Vac เราสามารถคำนวนได้ว่า

$$Vdc \approx (Vac * 1.414) - 2Vdiode \qquad (3.2)$$

$$\approx (20 * 1.414) - 1.4 = 26.8 \text{ โวลท์}$$

เลือกใช้ตัวเก็บประจุขนาด 4700 ไมโครฟารัด ทนแรงดันได้ 50 โวลท์ ทำหน้าที่ในการกรองกระแส และใช้ไอซีเบอร์ 7805 ในการสร้างแรงดัน 5 โวลท์ ภาคจ่ายไฟนี้สามารถแสดงได้ดังในรูปที่ 3.3



รูปที่ 3.3 แสดงวงจรภาคจ่ายไฟ

3.5 การเขียนโปรแกรม

ในการเขียนโปรแกรมสำหรับเครื่องตัดตราสัญลักษณ์นั้น จะถูกแบ่งออกเป็น 2 ส่วน คือ

- โปรแกรมในคอมพิวเตอร์ด้วย Borland C++ Builder
- โปรแกรมในส่วนควบคุม (Indexer) ด้วย Assembly

ในส่วนของโปรแกรมในคอมพิวเตอร์ด้วย Borland C++ Builder นั้นจะทำหน้าที่ในการรับข้อมูลจากไฟล์ HPGL (*.plt) , รับค่าพารามิเตอร์ต่างๆ ที่ใช้ในการตัด เช่น จำนวนของตราสัญลักษณ์ , รับค่าตำแหน่งของพล็อตเตอร์ และการประมวลผลทางภาพ แล้วนำค่าที่ได้เหล่านี้มาคำนวณเพื่อหาพิกัดในการกัดที่ถูกต้อง แล้วจึงส่งค่าพิกัดต่างๆ ที่ได้ในรูปแบบของไฟล์ HPGL ไปยังเครื่องพล็อตเตอร์เพื่อทำการตัด

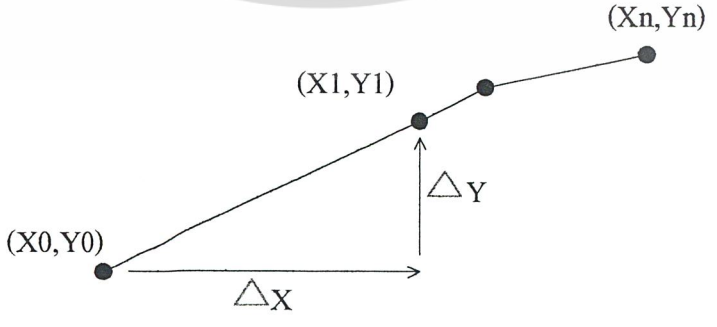
สำหรับเครื่องพล็อตเตอร์สำเร็จรูปนั้นสามารถที่จะรับไฟล์ HPGL เพื่อนำไปใช้งานได้ทันที แต่สำหรับเครื่องพล็อตเตอร์ที่สร้างขึ้นเองนั้น จะต้องมีส่วนของโปรแกรมที่เพิ่มเติมขึ้นมาด้วย เพื่อทำหน้าที่ในการแปลงไฟล์ HPGL ให้เป็น

- จำนวนพัลส์ (Pulse) หรือ สเต็ป (Step) เพื่อกำหนดระยะทางที่พล็อตเตอร์เคลื่อนที่ทั้งในแนวแกน X และ Y
- ความถี่ของสเต็ป (Step) เพื่อกำหนดความเร็วของการเคลื่อนที่ทั้งในแนวแกน X และ Y

ซึ่งจากไฟล์ HPGL ที่มีลักษณะเป็นคู่ลำดับ X, Y หลายๆ คู่เรียงกันอยู่ เราจึงสามารถหาความแตกต่างของระยะทางในแนวแกน X (ΔX) และแกน Y (ΔY) ได้ดังแสดงสมการที่ 3.3-3.4 และในรูปที่ 3.4

$$\text{ระยะทาง X} = \Delta X = X_{n+1} - X_n \quad ; n=0,1,2,... \qquad (3.3)$$

$$\text{ระยะทาง Y} = \Delta Y = Y_{n+1} - Y_n \quad ; n=0,1,2,... \qquad (3.4)$$



รูปที่ 3.4 แสดงลักษณะของคู่ลำดับในไฟล์ HPGL

ค่าที่คำนวณได้จะมีทั้งค่าบวก (+) และค่าลบ (-) ซึ่งเครื่องหมายนี้จะแทนทิศทางของการเคลื่อนที่นั่นเอง จากนั้นนำค่า ΔX, ΔY ที่ได้ซึ่งมีหน่วยเป็น Plot Unit (ค่าในไฟล์ HPGL จะมีหน่วยเป็น Plot Unit) มาคำนวณเป็นจำนวน Step ตามสมการที่ 3.5-3.6

$$\text{Step X} = \text{Scale} * \Delta X \qquad (3.5)$$

$$\text{Step Y} = \text{Scale} * \Delta Y \qquad (3.6)$$

โดยค่า Scale คือ อัตราส่วนของระยะทางที่ Plot ได้จริงใน 1 Step ต่อ 1 Plot Unit ซึ่งค่า Scale นี้จะขึ้นกับโครงสร้างพล็อตเตอร์และความละเอียด (Resolution) ของตัวขับสเต็ปปิงมอเตอร์ จากการทดลอง เราได้ค่า Scale เท่ากับ 1.25

สำหรับการคำนวณความถี่ของ Step นั้น สามารถคำนวณได้ตามสมการที่ 3.7-3.8 โดย

ถ้า $\Delta X >= \Delta Y$ จะได้ว่า

$$\text{ความถี่ Step X} = \text{Ref. Frequency} \qquad (3.7)$$

$$\text{ความถี่ Step Y} = \text{Ref. Frequency} * (\Delta Y / \Delta X)$$

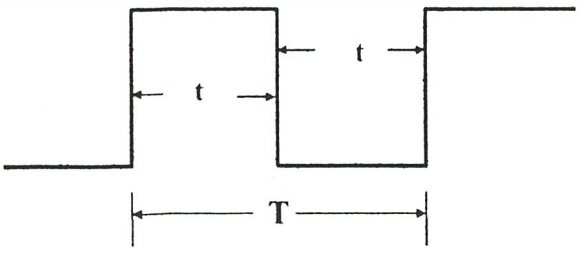
ถ้า $\Delta Y >= \Delta X$ จะได้ว่า

$$\text{ความถี่ Step Y} = \text{Ref. Frequency} \qquad (3.8)$$

$$\text{ความถี่ Step X} = \text{Ref. Frequency} * (\Delta X / \Delta Y)$$

โดย Ref. Frequency คือ ความถี่สูงสุดที่กำหนดความเร็วสูงสุดของการเคลื่อนที่ของพล็อตเตอร์ในขณะใดขณะหนึ่ง แต่สำหรับโครงงานนี้แล้ว เรากำหนดให้ Ref. Frequency เท่ากันตลอดทุกช่วงของเวลา

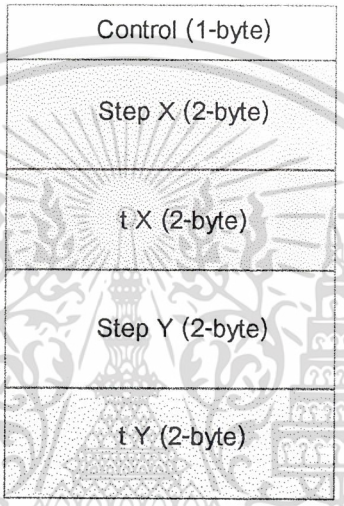
จากลักษณะของ Step ที่เป็นแบบ Square Wave แสดงดังในรูปที่ 3.5 เราสามารถกำหนดความถี่ของ Step ได้จากการกำหนดค่า T หรือค่า t (จากรูป t = T/2) โดยค่า T และ t สามารถคำนวณได้จากความถี่ Step ในสมการที่ 3.7-3.8 แล้วค่า t นี้จะถูกส่งให้กับส่วนควบคุม (Indexer) พร้อมกับจำนวน Step เพื่อให้ส่วนควบคุมสร้าง Step ที่มีจำนวนและความถี่ที่ต้องการ



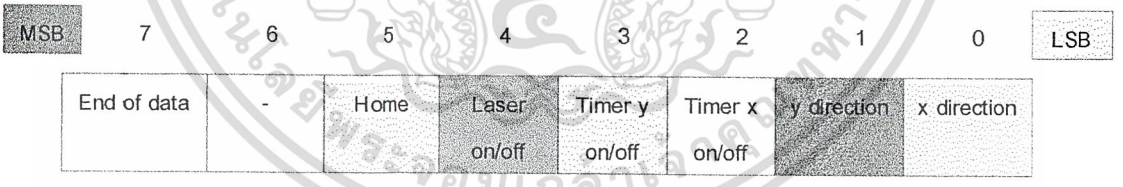
รูปที่ 3.5 แสดงลักษณะของ Step

ในการควบคุมการเปิด/ปิด ของเลเซอร์ที่ใช้ในการตัดตราสัญลักษณ์นั้น โดยปกติแล้วเลเซอร์จะถูกปิดไว้ เลเซอร์จะเปิดก็ต่อเมื่อคำสั่งในไฟล์ HPGL เป็น PD (Pen down) และเลเซอร์จะกลับมาปิดอีกครั้งเมื่อมีคำสั่ง PU (Pen up) เข้ามา

สำหรับค่าจากโปรแกรมในคอมพิวเตอร์ด้วย Borland C++ Builder ที่ต้องส่งให้กับส่วนควบคุม (Indexer) จะมีทั้งหมด 9 Byte ดังแสดงในรูปที่ 3.6 ซึ่งรายละเอียดของ Control Byte สามารถแสดงได้ดังรูปที่ 3.7 โดยค่าต่างๆ เหล่านี้จะถูกเก็บไว้ใน RAM ของส่วนควบคุม (Indexer)



รูปที่ 3.6 แสดงข้อมูลที่ถูกส่งให้กับส่วนควบคุม (Indexer)

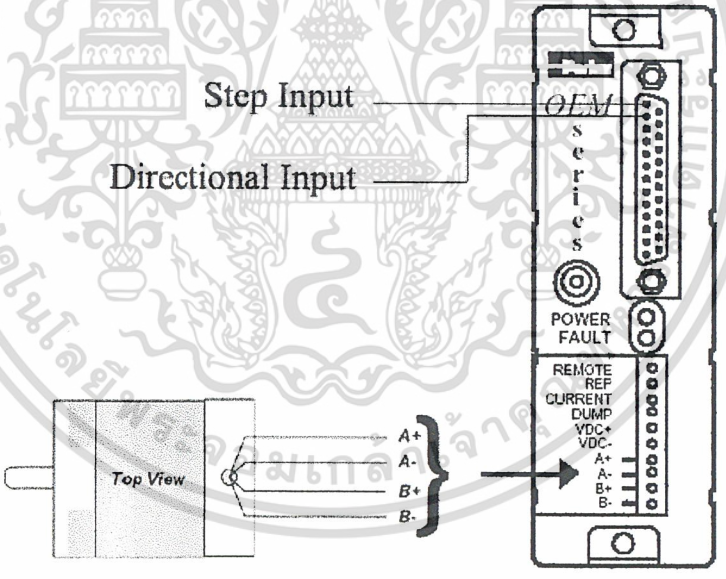


รูปที่ 3.7 แสดงรายละเอียดของ Control Byte

- X direction, Y direction : 0 = เคลื่อนที่ไปข้างหน้า
  1 = เคลื่อนที่ไปข้างหลัง
- Timer X, Timer Y : 0 = เปิด Timer ที่ใช้สร้าง Step
  1 = ปิด Timer ที่ใช้สร้าง Step
- Laser : 0 = ปิด Laser
  1 = เปิด Laser

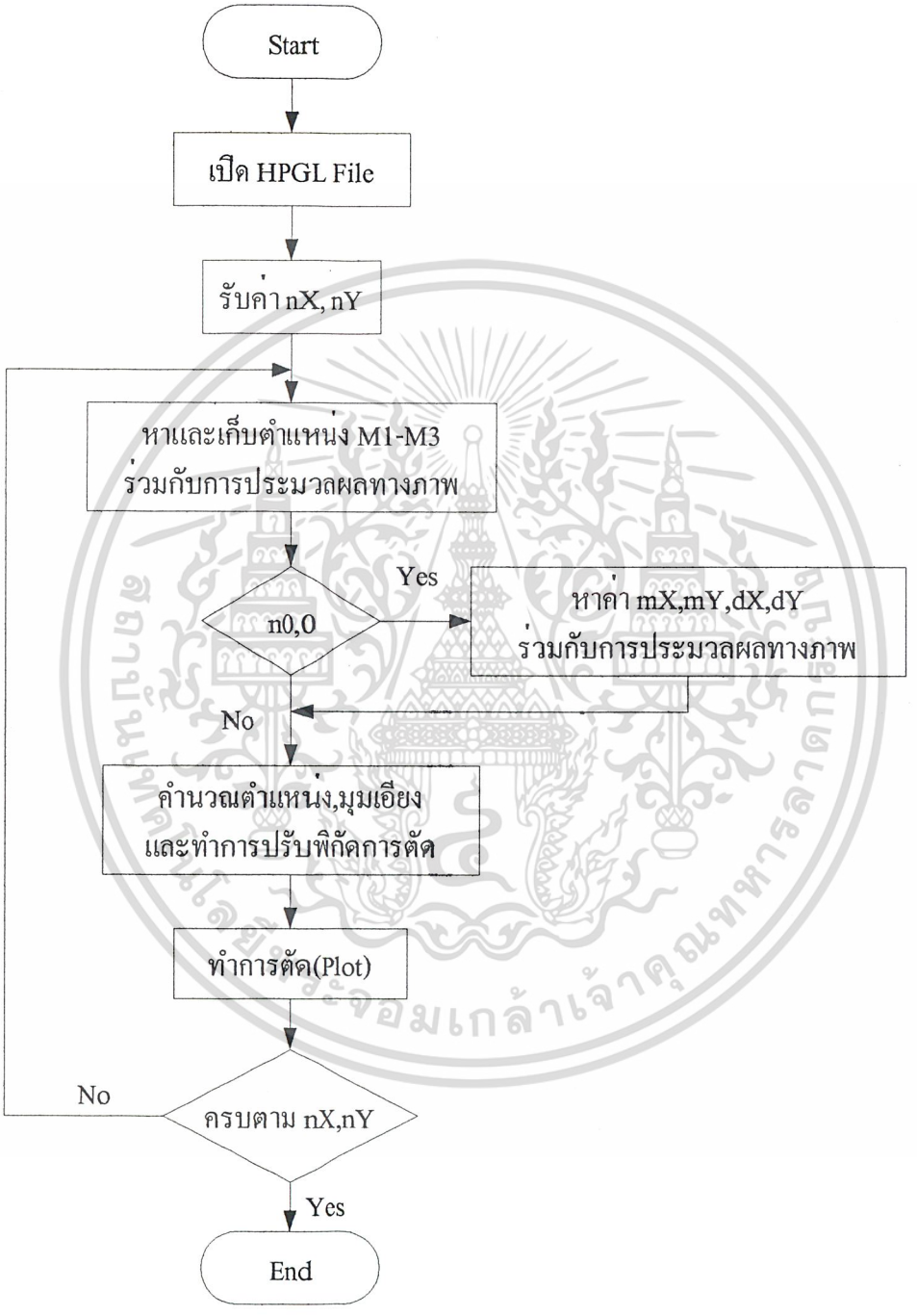
- Home : 0 = -
  1 = กลับไปยังตำแหน่ง Home (0,0)
- End of data : 0 = การส่งข้อมูลยังไม่สิ้นสุด
  1 = การส่งข้อมูลสิ้นสุดแล้ว

สำหรับในส่วนของโปรแกรมในส่วนควบคุม (Indexer) ด้วย Assembly นั้นจะถูกเขียนลงบนไมโครคอนโทรลเลอร์ MCS-51 โดยส่วนควบคุมนี้จะทำหน้าที่ให้การรับค่าจากคอมพิวเตอร์มาเก็บไว้ใน RAM แล้วดึงค่าออกมาใช้ในการควบคุมพล็อตเตอร์ เช่น ทำการสร้าง Step ที่มีจำนวน Step และความถี่ตามค่าที่ได้คำนวณไว้แล้วด้วยโปรแกรมในคอมพิวเตอร์ โดยอาศัยหลักการทำงานของ Timer0 และ Timer1 เข้ามาช่วย ซึ่งค่า Step และ Direction ของทั้งแกน X และแกน Y นี้เอง จะถูกส่งให้กับตัวขับสเต็ปปิงมอเตอร์ เพื่อควบคุมให้พล็อตเตอร์เคลื่อนที่ไปในทิศทาง, ระยะทาง และความเร็วที่ต้องการดังแสดงในรูปที่ 3.8

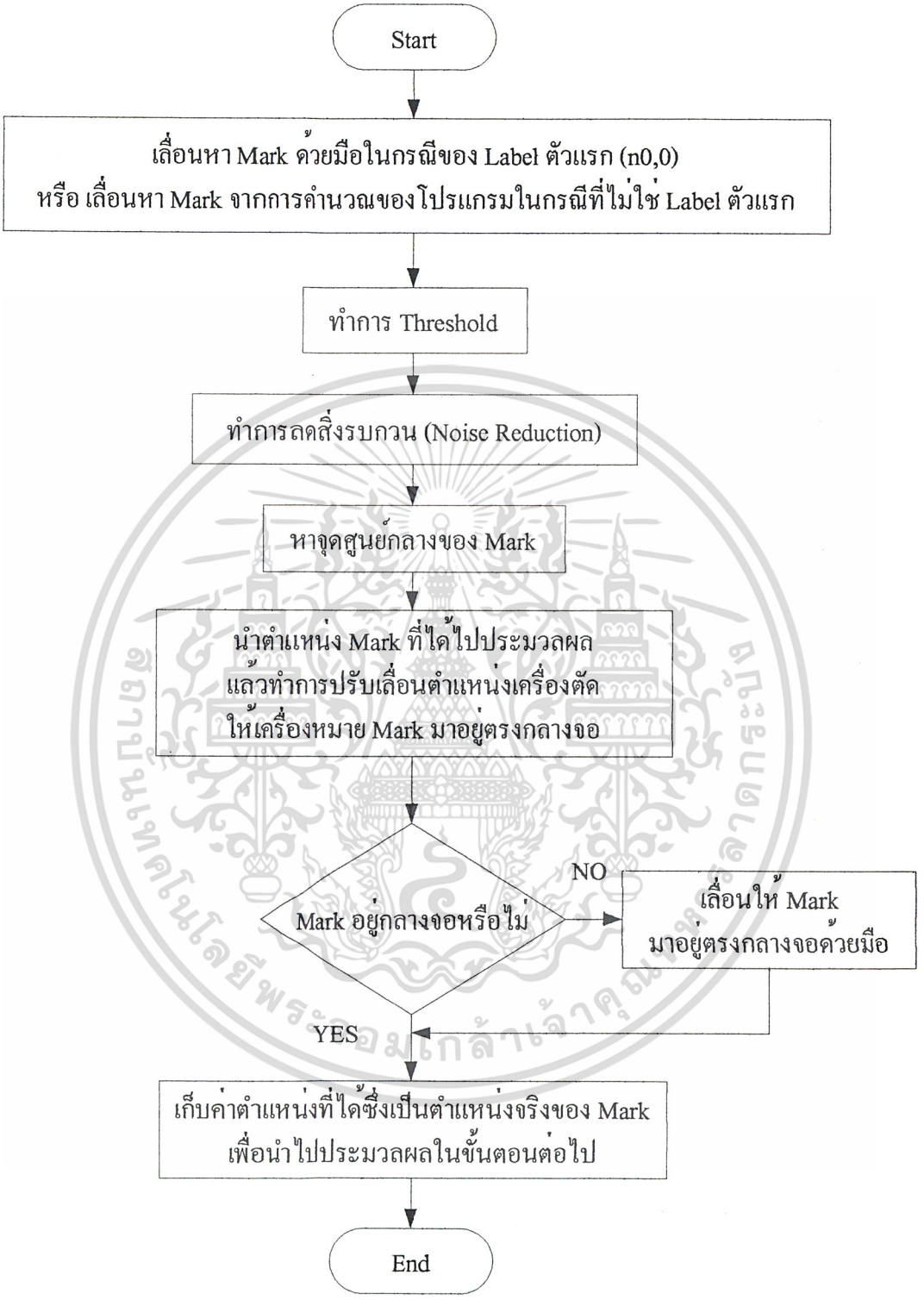


รูปที่ 3.8 แสดงลักษณะการต่อของตัวขับสเต็ปปิงมอเตอร์

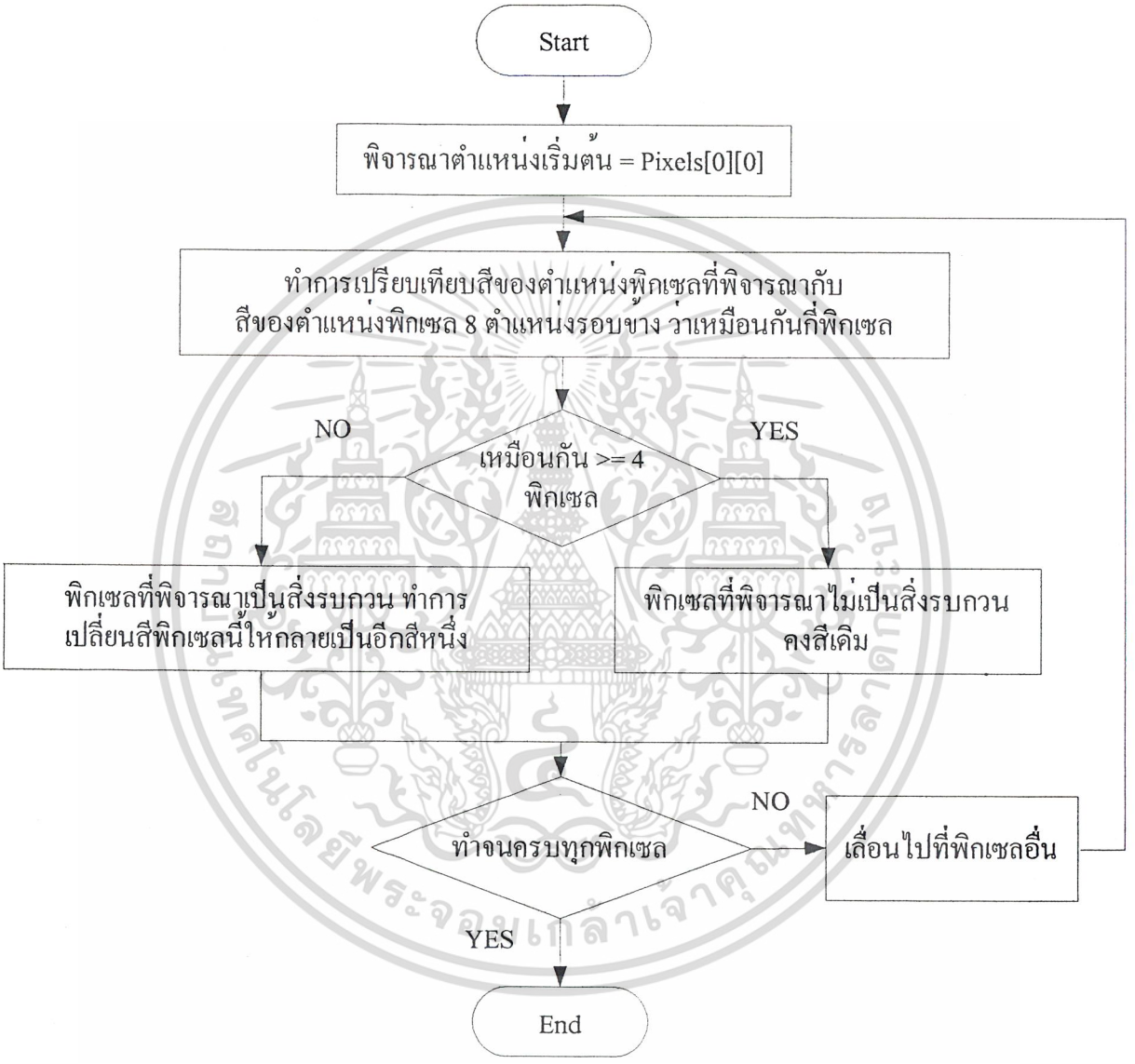
3.5.1 Flow Chart ของโปรแกรมในคอมพิวเตอร์ด้วย Borland C++ Builder



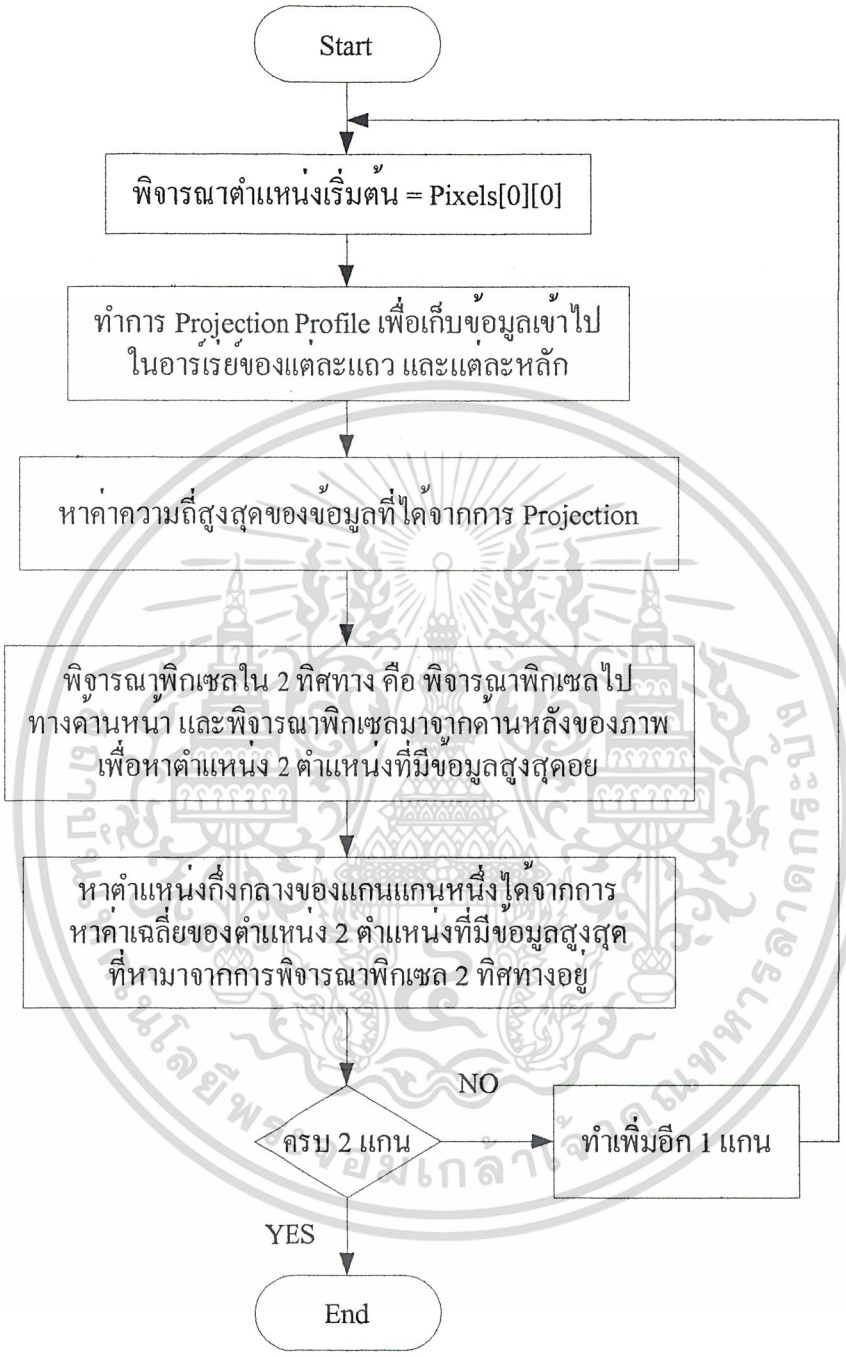
รูปที่ 3.9 แสดง Flow Chart การทำงานของเครื่องตัดตราสัญลักษณ์



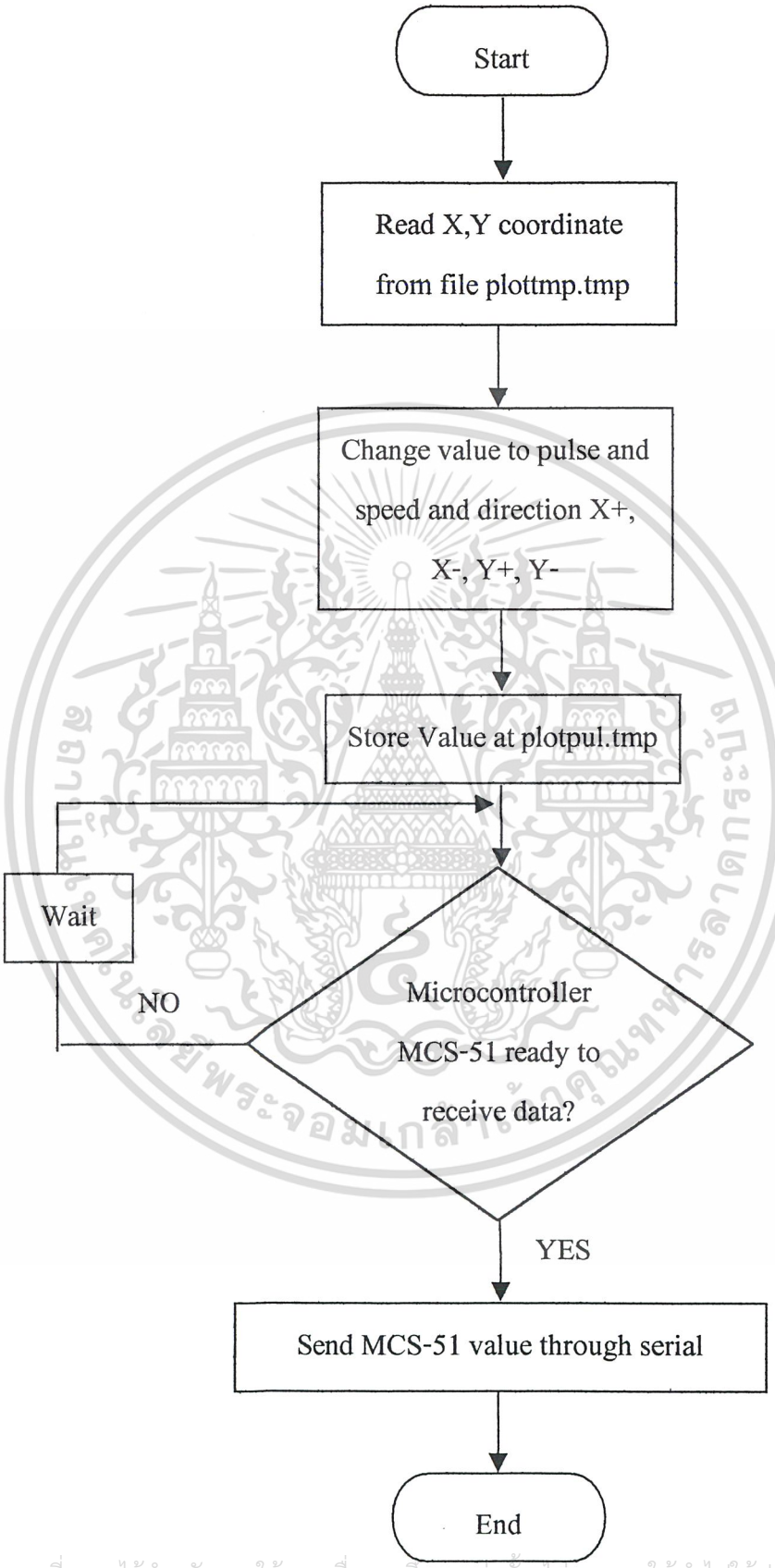
รูปที่ 3.10 แสดง Flow Chart การทำงานในส่วนประมวลผลทางภาพที่ใช้ในเครื่องตัดตราสัญลักษณ์



รูปที่ 3.11 แสดง Flow Chart การลดสิ่งรบกวน (Noise Reduction) ในการประมวลผลทางภาพ

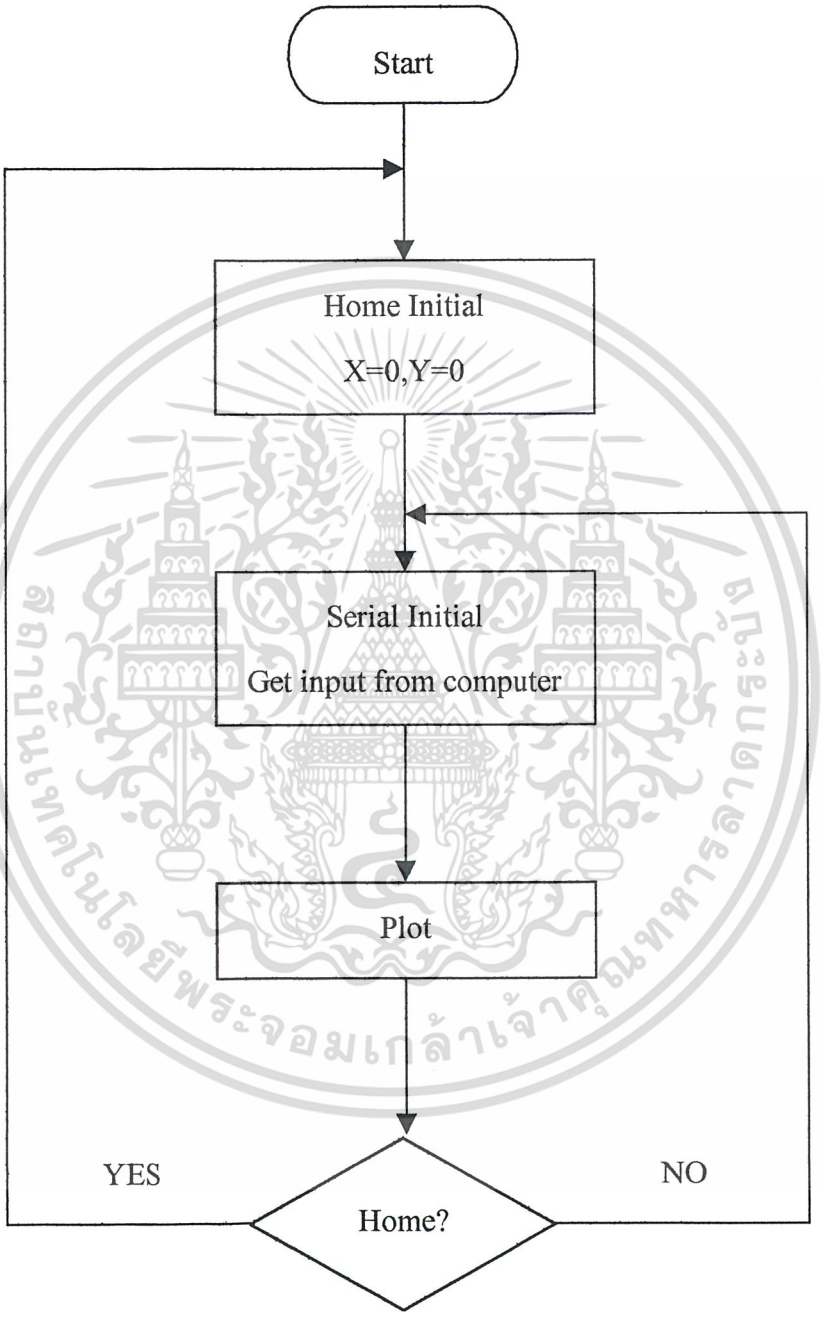


รูปที่ 3.12 แสดง Flow Chart การหาตำแหน่งจุดศูนย์กลางของ Mark ในการประมวลผลทางภาพ

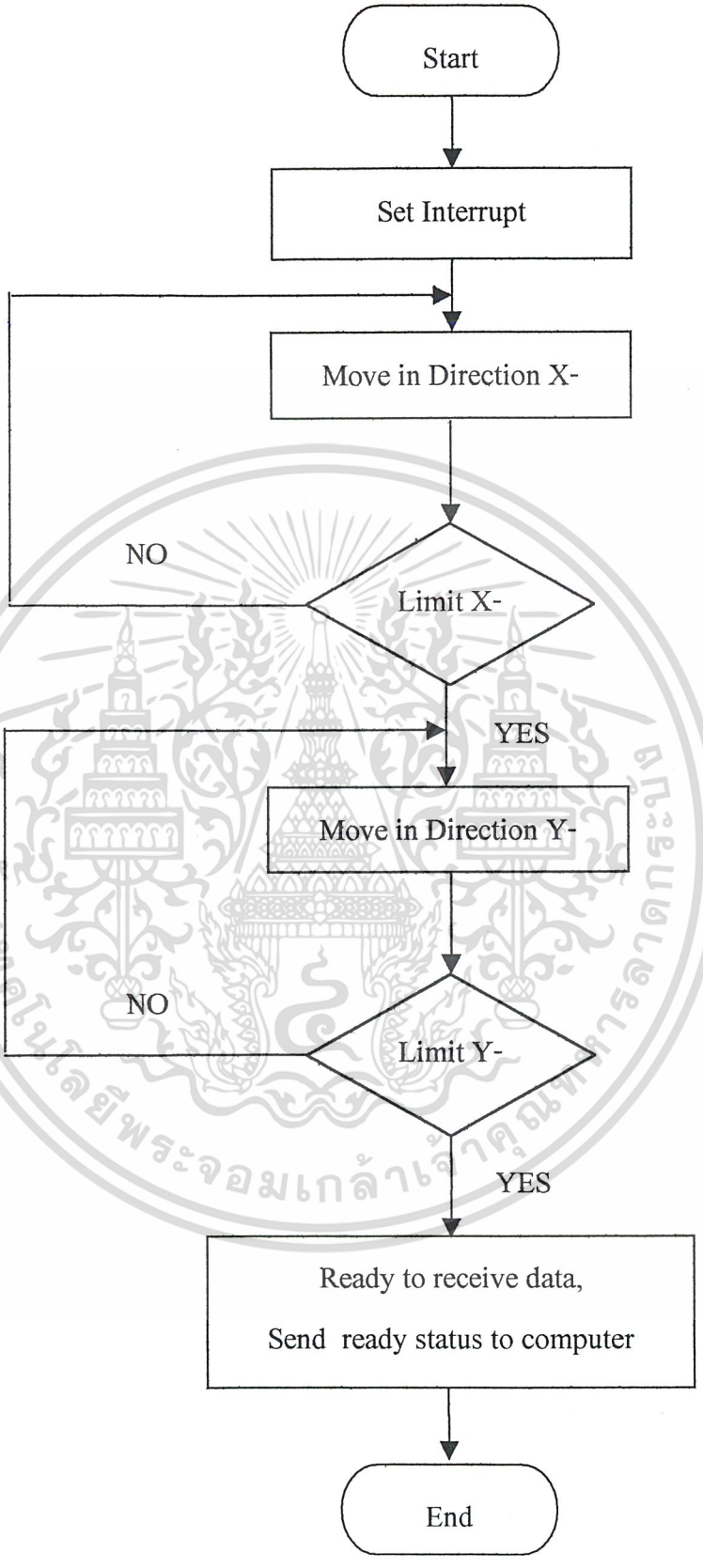


รูปที่ 3.13 แสดง Flow Chart การแปลงข้อมูลและส่งข้อมูลในการพล็อต

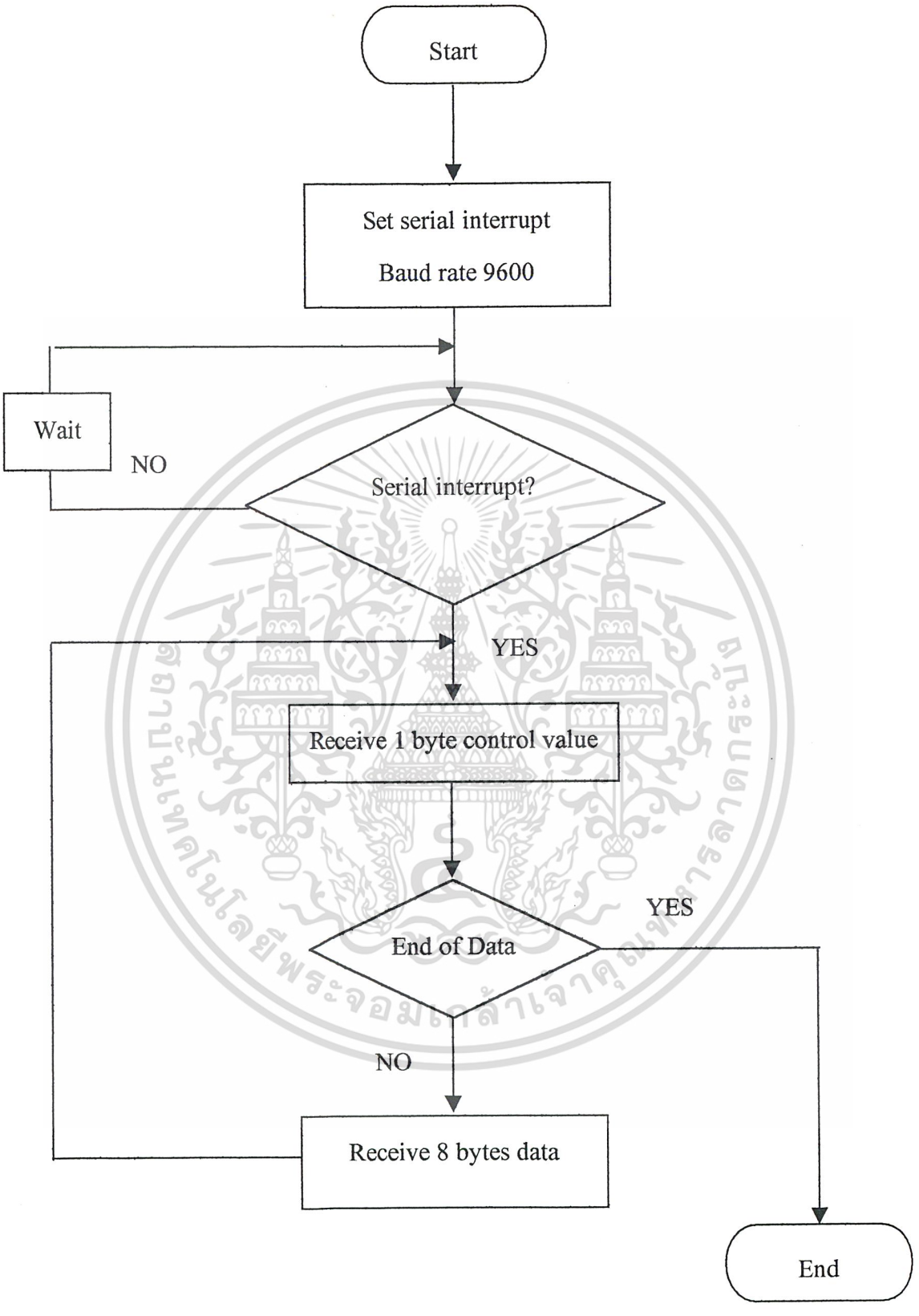
3.5.2 Flow Chart ของโปรแกรมในส่วนควบคุม (Indexer)



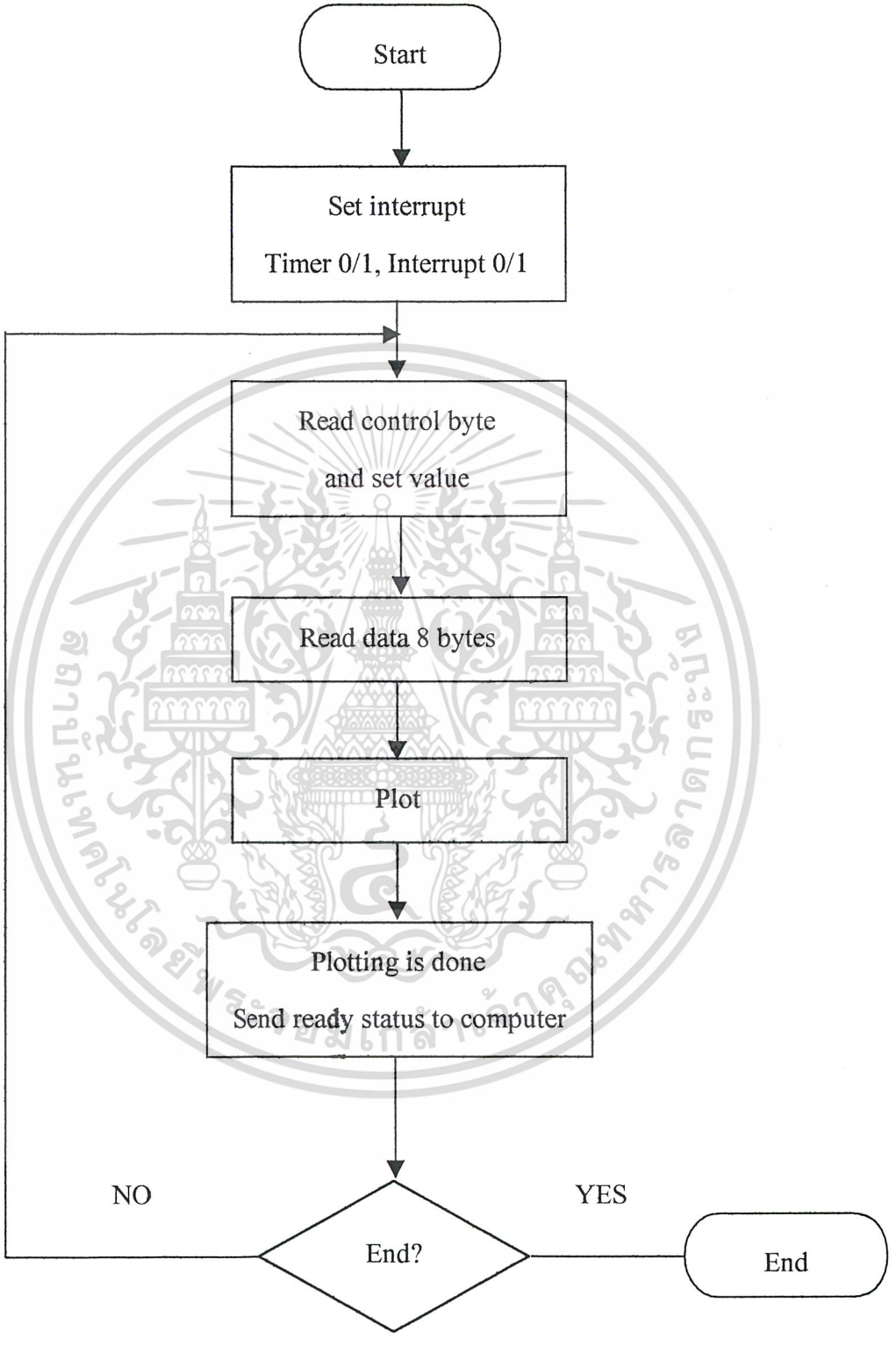
รูปที่ 3.14 แสดง Flow Chart การทำงานของเครื่องตัดตราสัญลักษณ์ในส่วนควบคุม (Indexer)



รูปที่ 3.15 แสดง Flow Chart การเลื่อนพล็อตเตอร์มาที่ตำแหน่ง HOME



รูปที่ 3.16 แสดง Flow Chart การรับข้อมูลจากพอร์ตอนุกรม



รูปที่ 3.17 แสดง Flow Chart การพล็อตของเครื่องพล็อตเตอร์

บทที่ 4
การทดลอง สรุป และวิจารณ์ผลการทดลอง

ในบทนี้จะเป็นการทดลองลองเพื่อทดสอบว่าเครื่องตัดตราสัญลักษณ์ สามารถทำงานได้อย่างถูกต้องหรือไม่ มีความแม่นยำมากน้อยเพียงไร

4.1 การทดลอง

ในการทดลองนี้ จะทำการสร้างรูปแบบของสัญลักษณ์ขึ้นมาหลายๆ แบบด้วยโปรแกรม Corel Draw เพื่อทดสอบว่าเครื่องตัดตราสัญลักษณ์สามารถทำงานได้ถูกต้องจริงหรือไม่ ซึ่งจะจำลองการตัดตราสัญลักษณ์ด้วยการพล็อตแทน ด้วยสร้างไฟล์ที่มีตราสัญลักษณ์เป็นตัวอักษรคำว่า "TEST" พร้อมเครื่องหมาย (Mark) ทั้ง 3 จุด โดยทำการหมุนภาพไปในมุมต่างๆ และเพิ่มจำนวนของตราสัญลักษณ์ แล้วนำไฟล์ที่ได้นี้มาพล็อตลงบนกระดาษ จะได้รูปตราสัญลักษณ์ "TEST" พร้อมกับ Mark ทั้ง 3 จุด

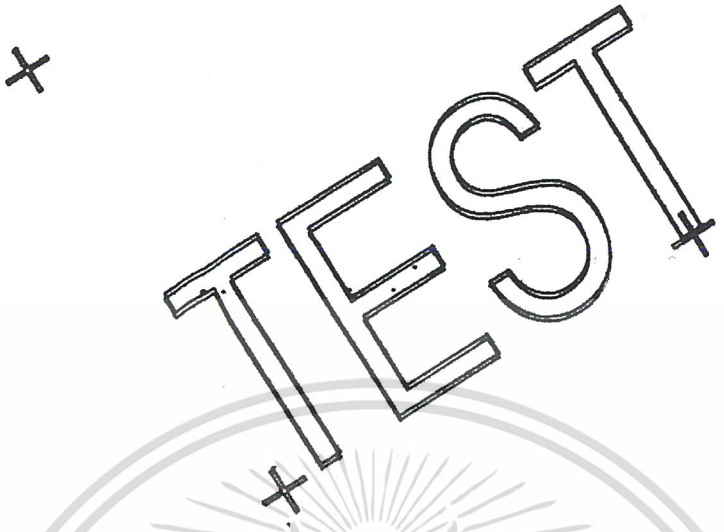
จากนั้นทำการลบ Mark ทั้ง 3 จุดออกจากไฟล์ โดยให้เหลือแค่คำว่า "TEST" อยู่ ซึ่งเราจะทำการพล็อตคำว่า "TEST" เพียงอย่างเดียว ด้วยการอ้างตำแหน่งจาก Mark ทั้ง 3 ที่ได้ทำไว้แล้ว ถ้าเครื่องตัดสัญลักษณ์ทำงานได้ถูกต้อง คำว่า "TEST" ที่พล็อตลงไปจะต้องไปทับกับคำว่า "TEST" ที่ถูกพล็อตไว้ก่อนหน้านี้แล้วพอดี โดยเครื่องพล็อตเตอร์สำเร็จรูป (MUTOH Intelligent Plotter) นั้นจะจำลองการตัดโดยการพล็อตด้วยปากกา จึงสามารถที่จะนำรูปผลการทดลองมาแสดงได้อย่างชัดเจน แต่สำหรับเครื่องพล็อตเตอร์ที่สร้างขึ้นนั้น จะจำลองการตัดด้วยการใช้ลำแสงจากเลเซอร์พอยน์เตอร์ (Laser Pointer) การแสดงผลการทดลองจึงทำได้เพียงการแสดงภาพเพียงบางส่วนของการทดลองเท่านั้น

4.2 ผลการทดลอง

จากการทดลอง ได้ผลการทดลองดังต่อไปนี้

- **ผลการทดลองของการจำลองการตัดด้วยการพล็อตจากเครื่องพล็อตเตอร์สำเร็จรูป (MUTOH Intelligent Plotter)**

รูปที่ 4.1 แสดงผลของการพล็อตเมื่อตราสัญลักษณ์อยู่ในระดับตรง

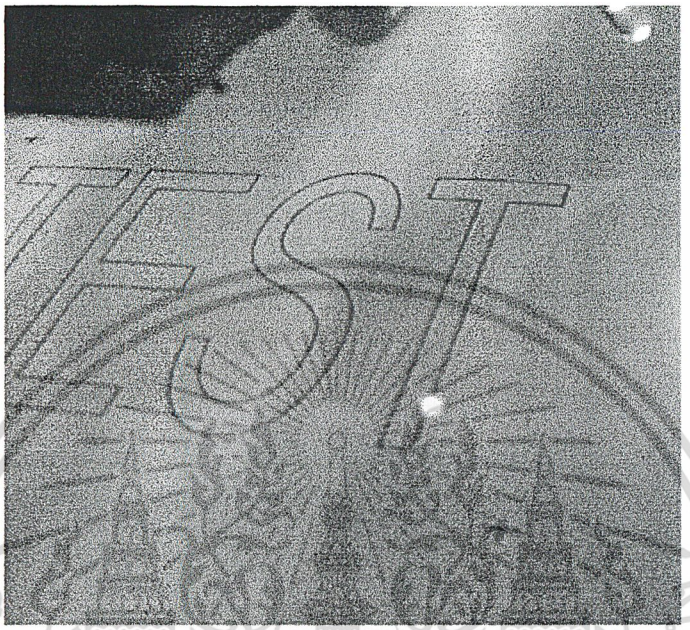


รูปที่ 4.2 แสดงผลของการพล็อตเมื่อตราสัญลักษณ์อยู่ในระดับเอียงซ้าย

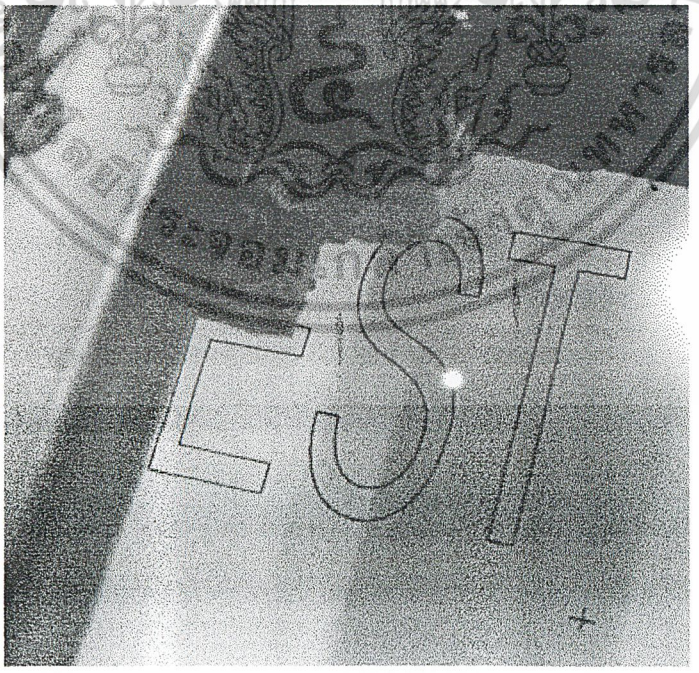
รูปที่ 4.3 แสดงผลของการพล็อตเมื่อตราสัญลักษณ์อยู่ในระดับเอียงขวา

รูปที่ 4.4 แสดงผลของการพล็อตเมื่อตราสัญลักษณ์อยู่ในระดับตรงและมีจำนวน 4 ตราสัญลักษณ์

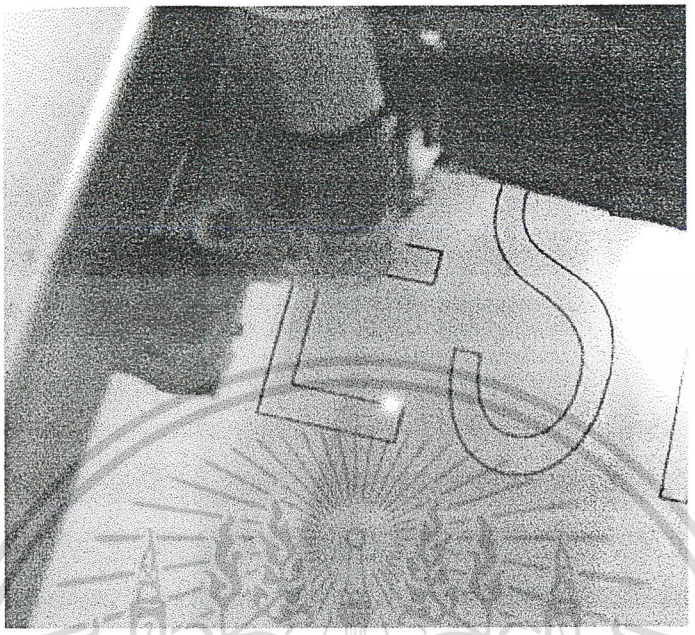
- **ผลการทดลองของการจำลองการตัดด้วยการพล็อตจากเครื่องพล็อตเตอร์ที่สร้างขึ้นเอง**



รูปที่ 4.5 ก



รูปที่ 4.5 ข



รูปที่ 4.5 ค

รูปที่ 4.5ก, 4.5ข, 4.5ค แสดงผลของการจำลองการตัดด้วยการพล็อตจากเครื่องพล็อตเตอร์ที่สร้างขึ้นมาเอง

จากผลการทดลองเห็นว่า คำว่า "TEST" ที่พล็อตด้วยเครื่องพล็อตเตอร์ทั้ง 2 แบบนั้น จะสามารถพล็อตไปทับกับคำว่า "TEST" ที่มีอยู่เดิมเกือบพอดี แสดงให้เห็นว่าเครื่องสามารถตัดตราสัญลักษณ์ได้อย่างถูกต้อง แม้ว่าตราสัญลักษณ์จะวางไม่ตรงตำแหน่งเดิม แต่เครื่องก็สามารถแก้ไขปรับพิกัด ให้สามารถพล็อตหรือตัดตราสัญลักษณ์ได้อย่างถูกต้อง

ความคลาดเคลื่อนที่เกิดขึ้นนั้นมีค่าค่อนข้างน้อย โดยคลาดเคลื่อนในช่วงไม่เกิน 1 มิลลิเมตร ซึ่งสาเหตุของความคลาดเคลื่อนนี้น่าจะเป็นผลมาจากความไม่มั่นคงของกล้อง

4.3 สรุปและวิจารณ์ผลการทดลอง

- เครื่องสามารถจำลองการตัดตราสัญลักษณ์หรือพล็อตค่าได้ถูกต้อง แม้ว่าตราสัญลักษณ์จะมีตำแหน่งที่ไม่ตรงก็ตาม
- ความไม่มั่นคงของกล้องจะมีผลทำให้การจำลองการตัดหรือการพล็อตเกิดการคลาดเคลื่อนได้ จึงควรที่จะปรับปรุงให้มีความมั่นคงแข็งแรงมากขึ้น

- การประมวลผลของโปรแกรมในบางตอนใช้เวลามากเกินไป หรือบางครั้งให้ผลไม่ถูกต้องเท่าที่ควร
- ตัวแปรที่ใช้ในการเขียนโปรแกรมมีจำนวนมากและไม่มีการตั้งชื่อหรือจัดระเบียบ จะทำให้เกิดความสับสนเมื่อกลับมาแก้ไขโปรแกรมอีกครั้ง
- ชนิดของวัสดุที่นำมาทำตราสัญลักษณ์ที่มีชนิดแตกต่างกัน จะทำให้ผลการสะท้อนของแสงต่างกัน บางครั้งจึงมีความจำเป็นที่ต้องทำการปรับแสงสว่างเพื่อให้เหมาะสมกับวัสดุแต่ละชนิด

4.4 ปัญหาและอุปสรรคในโครงงาน

- เนื่องจากเครื่องพล็อตเตอร์ที่ใช้ในโครงงานนี้เป็นแบบ Intelligent Plotter จึงเกิดอุปสรรคในการทำงานได้ เช่น ต้องมีการลงปากกา(PEN DOWN, PD) ทุกครั้ง เครื่องจึงจะสามารถเคลื่อนไปในตำแหน่งที่ต้องการได้ ทั้งที่โดยปกติแล้วสามารถใช้คำสั่ง PEN UP, PU ได้ทันที หรือว่าขณะที่ยังไม่มีคำสั่งใดๆ เข้ามา เครื่องจะเก็บปากกาให้โดยอัตโนมัติ ซึ่งเป็นผลให้ทำการ Mark ค่าได้ค่อนข้างลำบาก
- กล้องวีดีโอที่ใช้มีขนาดค่อนข้างใหญ่ ทำให้ยึดกับตัว Plotter ได้ลำบาก

บรรณานุกรม

นุกูล กระจาย , การเขียนโปรแกรมระบบวิชวลด้วย C++Builder 5 ISBN : 974-298-069-1, จัดพิมพ์โดย สุวีริยาสาสน์ , พิมพ์ครั้งที่ 1 , 2544

วรรณวิภา ติตถะสิริ , การเขียนโปรแกรม C ด้วยตนเอง ISBN : 974-512-164-9 , จัดพิมพ์โดย บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน) , 2538

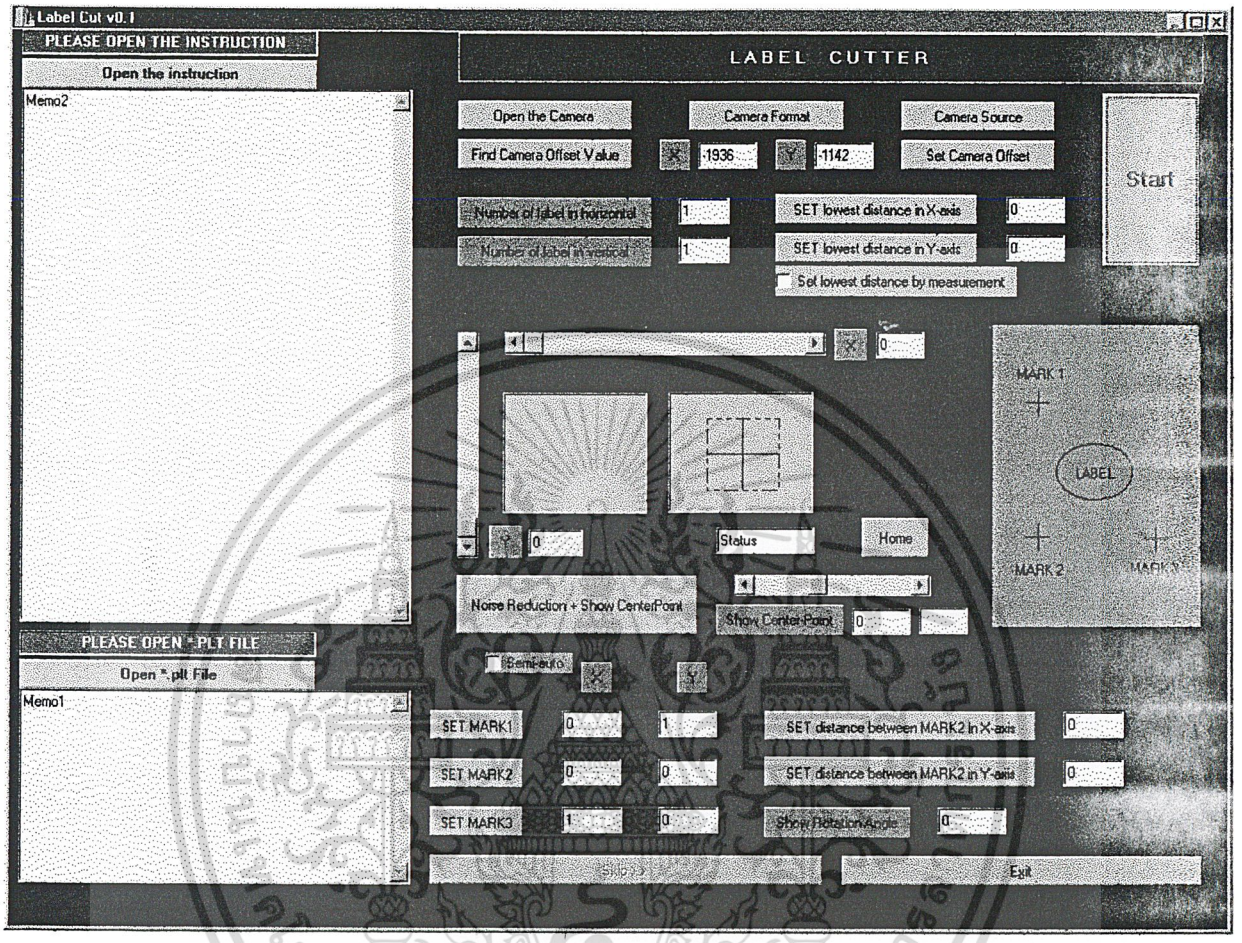
วิทยา เรืองพรวิสุทธิ์ , คู่มือโปรแกรมภาษา C สำหรับผู้เริ่มต้น ISBN : 974-509-542-7 , จัดพิมพ์โดย บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน) , 2544

วรพจน์ กรแก้ววัฒนกุล , ชัยวัฒน์ ลิ้มพรจิตรวิไล , เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ฉบับ AT89C5X ของ Atmel ISBN : 974-87935-5-9 , จัดพิมพ์โดย บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

Wayne Niblack , An Introduction to DIGITAL IMAGE PROCESSING ISBN : 0-13-480674-3 , Prentice-Hall International (UK) Ltd , 1986

Milan Sonka , Image Processing , Analysis , and Machine Vision , An International Thomson Publishing Company , Second Edition

ภาคผนวก



แสดงหน้าต่างของโปรแกรมในเครื่องคอมพิวเตอร์ด้วย Borland C++ Builder 5

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include "math.h"
#include "labelMO.h"
#include "stdio.h"
#include "stdlib.h"
#include "io.h"
#include "GifLZW.hpp"
#include "TIFLZW.hpp"
#include "ImageEnIO.hpp"
#include "Unit2.h"

// GLOBAL VARIABLES
HANDLE hComm = NULL;
TRead *ReadThread;
COMMTIMEOUTS ctmoNew = {0}, ctmoOld;
int l,change=0;
int plotx,ploty,nxx,nyy,minx,miny,nx,ny;
int
dx,dy,mmx,mmy,xm1,x1m,ym1,y1m,xm2,x2m,ym2,y2m,xm3,x3m,ym
3,y3m;
int xoffset,yoffset,resulty,resultx,absmmx,absmmy;
double angle,angle3,marktempx,marktempy,absmm,yyy,xxx;
//add
int
nowx=0,nowy=0,config,status=0,mvplot=0,busy=0,offset=0,limit=0;
int tchange=0,cauto=0,autoc=0,timer1done=0;

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "IEOpenSaveDlg"
#pragma link "IEVect"
#pragma link "ieview"
#pragma link "ImageEnIO"
#pragma link "ImageEnProc"
#pragma link "ImageEnView"
#pragma link "VideoCap"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void TForm1::DisplayVideoSize()
{
        TRect r=ImageEnVideoView1->GetVideoSize();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction
&Action)
{
 // TERMINATE THE THREAD.
 ReadThread->Terminate(); // TERMINATE THE THREAD

 // IF A PORT WAS OPENED, CLOSE IT
 if(hComm)
 {
  Sleep(250); // WAIT
  SetCommTimeouts(hComm, &ctmoOld);
        CloseHandle(hComm);
 }
 ImageEnVideoView1->ShowVideo=FALSE;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
 DCB dcbCommPort;
 // OPEN THE COMM PORT.
 hComm = CreateFile("COM1",
        GENERIC_READ | GENERIC_WRITE,
        0,
        0,
        OPEN_EXISTING,
        0,
        0);

 // IF THE PORT CANNOT BE OPENED, BAIL OUT.

 if(hComm == INVALID_HANDLE_VALUE) Application->
Terminate();

 // SET THE COMM TIMEOUTS.

 GetCommTimeouts(hComm,&ctmoOld);
        ctmoNew.ReadTotalTimeoutConstant = 100;
 ctmoNew.ReadTotalTimeoutMultiplier = 0;
 ctmoNew.WriteTotalTimeoutMultiplier = 0;
 ctmoNew.WriteTotalTimeoutConstant = 0;
 SetCommTimeouts(hComm, &ctmoNew);

 // SET BAUD RATE, PARITY, WORD SIZE, AND STOP BITS.
 // THERE ARE OTHER WAYS OF DOING SETTING THESE BUT
THIS IS THE EASIEST.
 // IF YOU WANT TO LATER ADD CODE FOR OTHER BAUD
RATES, REMEMBER
 // THAT THE ARGUMENT FOR BuildCommDCB MUST BE A
POINTER TO A STRING.
 // ALSO NOTE THAT BuildCommDCB() DEFAULTS TO NO
HANDSHAKING.

 dcbCommPort.DCBlength = sizeof(DCB);
 GetCommState(hComm, &dcbCommPort);
 BuildCommDCB("9600,N,8,1", &dcbCommPort);
 SetCommState(hComm, &dcbCommPort);

 // ACTIVATE THE THREAD. THE FALSE ARGUMENT SIMPLY
MEANS IT HITS THE
 // GROUND RUNNING RATHER THAN SUSPENDED.

 ReadThread = new TRead(false);
 int a=32;
 TransmitCommChar(hComm, a);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Run(TObject *Sender)
{
 //Set first plot position -> nxx, nyy.
 //and send to BeforePlot procedure.

 nxx=0;
 nyy=0;
 BeforePlot();


}
//---------------------------------------------------------------------------

void __fastcall TForm1::Terminate(TObject *Sender)
{
   Application->Terminate();          //Exit program
}
//---------------------------------------------------------------------------


void __fastcall TForm1::Open(TObject *Sender)
{
 //Open *.plt file

 OpenDialog1->Execute();
 Edit2->Text=OpenDialog1->FileName;
 ProcessPanel->Caption="Press start and follow the instructions.";

 //Open *.plt file to memo1, modify file for
 //compatible MH-GL command and save to temp file.
 FILE *fp;     //Declare Variables
 FILE *hp;
 FILE *bb;
 FILE *plot;
 int j=1,k=1;
```

```
int x[1023],y[1023];
char d[1000],c;

//Open *.plt file to memo1. and save to plottemp.tmp
{
  if(Memo1->Lines->Count>0)  Memo1->Clear();
  Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
  Memo1->Lines->SaveToFile("plottemp1.tmp");
}

//Open Plottemp.tmp, modify file and save to plottemp2.tmp

fp = fopen("Plottemp1.tmp", "r");       //Open files.
hp = fopen("plottemp2.tmp", "w");

//Choose the compatible MH-GL command line.

while (j<=12)
{
  fscanf(fp, "%c", &d[j]);
  fprintf(hp, "%c", d[j]);
  j=j++;
}
j=1;
while (j<=149)
{
  fscanf(fp, "%c", &d[j]);
  j=j++;
}
j=1;
while (j<=9)
{
  fscanf(fp, "%c", &d[j]);
  fprintf(hp, "%c", d[j]);
  j=j++;
}

//Read co-ordinate and find minx, miny
//which will be used to transpose co-ordinate.

minx=9000; miny=9000;   //Assume for finding minx, miny.
j=k=1;
do
{
fscanf(fp, "%c", &d[j]);

//Read co-ordinate from "PU" and "PD" command.

if (d[j]=='P')
  {
  fscanf(fp, "%c%d%c%d%c%c",&d[j+1],&x[k],&d[j+2],&y[k],&d
[j+3],&d[j+4]);
  fprintf(hp, "%c%c%d%c%d%c%c", d[j],d[j+1],x[k],d[j+2],y[k],d
[j+3],d[j+4]);

  //Find minx, miny.

  if (x[k] < minx)  minx = x[k];
  if (y[k] < miny)  miny = y[k];

  }
else
  {
  fprintf(hp, "%c", d[j]);
  do
    {
    fscanf(fp, "%c", &d[j]);
    fprintf(hp, "%c", d[j]);
    }
  while (d[j]!=';');
    {
    fscanf(fp, "%c", &d[j]);
    fprintf(hp, "%c", d[j]);
    }
  }
}
while (!feof(fp));
fclose(fp);          //close files.
fclose(hp);

  if(Memo1->Lines->Count>0)  Memo1->Clear();
  Memo1->Lines->LoadFromFile("plottemp2.tmp");  //Update memo1.

}
//---------------------------------------------------------------------------

void __fastcall TForm1::CamOffsetButtonClick(TObject *Sender)
{
  //This procedure is used for finding camera offset
  //by drawing mark at (3500,4000) and finding plotter position
  //which offset position is.

  FILE *fp;
  int a;

  offset=1;
  ProcessPanel->Caption="Please move mark to laser, then move until
mark is in center";
  plotx=3500;
  ploty=4000;
  MovePlot();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ScrollBarYChange(TObject *Sender)
{
  ScrollBarChange();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ScrollBarXChange(TObject *Sender)
{
  ScrollBarChange();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Source(TObject *Sender)
{
  // Camera Source
  if( !ImageEnVideoView1->DoConfigureSource() )
        MessageDlg("Configure Source dialog not
available",mtInformation,TMsgDlgButtons() << mbOK,0);
  else
  {
    DisplayVideoSize();
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Format(TObject *Sender)
{
  // Camera Format
  if( !ImageEnVideoView1->DoConfigureFormat() )
    MessageDlg("Configure Format dialog not
available",mtInformation, TMsgDlgButtons() << mbOK,0);
  else
        DisplayVideoSize();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ImageEnView1Job(TObject *Sender, TIEJob
job,
    int per)
{
  switch( job )
  {
    case iejNOTHING:
          break;
        case iejVIDEOCAP_CONNECTING:
          break;
  }
  Application->ProcessMessages();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ImageEnVideoView1VideoFrame(TObject
*Sender,
    Graphics::TBitmap *Bitmap)
```

```
{
    //Draw camera image to ImageEnView1.

    ImageEnView1->Assign(Bitmap);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::OpenCameraClick(TObject *Sender)
{
      ImageEnVideoView1->ShowVideo=true;
      DisplayVideoSize();
      ImageEnVideoView1-
>OnVideoFrame=ImageEnVideoView1VideoFrame;

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button10Click(TObject *Sender)
{
int y1[60];
int x,y,positionX,positionY;
int countblack,countwhite;
int count;

// Find that the mark is white or black.
// If the mark is white then background is black.
// If the mark is black then background is white.

ThresChange();

count=0;
countblack=0;
countwhite=0;

for(int y=0;y<60;y++)
{
            for (int x=0; x<60;x++)
            {
            if (Image1->Canvas->Pixels[x][y]==clBlack)
{countblack++;}
      else countwhite++;
            }
}
x=0;
y=0;

// In this case , the mark is black and the background is white.

if (countwhite > countblack)
{
x=0;
y=0;
count=0;

// Noise Reduction.

 for(int y=0;y<60;y++)
 {
  for (int x=0; x<60;x++)
     {
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y-1])  count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y])    count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y+1])  count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y-1])    count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y+1])    count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y-1])  count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y])    count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y+1])  count++;
     else count=count;
     if (count>=4) Image1->Canvas->Pixels[x][y] = Image1->Canvas-
>Pixels[x][y];
     else Image1->Canvas->Pixels[x][y] = clWhite;

     count=0;
     }
 }
 x=0;
 y=0;
 count=0;

// Store pixels which color is black.Find pixels in row.

 for(int y=0;y<60;y++)
 {
     for (int x=0; x<60;x++)
          {
          if (Image1->Canvas->Pixels[x][y]==clBlack) {count++;}
          }
     y1[y]=count;
     count=0;
 }

// Find the maximum value of black pixels.

 count=0;
 for(int y=0;y<60;y++)
 {
 if (y1[y]>count) {count=y1[y];}
 }

// Find row's position that has maximum black pixels.Find forward.

 int tmp1,tmp2;
 y=0;
 while (y1[y]!=count) y++;
 tmp1=y;

// Find row's position that has maximum black pixels.Find backward.

 y=60;
 while (y1[y]!=count) y--;
 tmp2=y;

// Find the middle position.

 if (tmp1!=tmp2) {resulty=(tmp1+tmp2)/2;}
 else resulty=tmp1;
 positionY=resulty;

// Store pixels which color is black.Find pixels in column.

 count=0;
 for(int x=0;x<60;x++)
 {
            for (int y=0; y<60;y++)
            {
            if (Image1->Canvas->Pixels[x][y]==clBlack) {count++;}
            }
 y1[x]=count;
 count=0;
 }

// Find the maximum value of black pixels.

 count=0;
 for(int x=0;x<60;x++)
 {
 if (y1[x]>count) {count=y1[x];}
 }

// Find column's position that has maximum black pixels.Find forward.
```

```
 x=0;
 while (y1[x]!=count) x++;
 tmp1=x;

// Find column's position that has maximum black pixels.Find
backward.

 x=60;
 while (y1[x]!=count) x--;
 tmp2=x;

// Find the middle position.

 if (tmp1!=tmp2) {resultx=(tmp1+tmp2)/2;}
 else resultx=tmp1;
 positionX=resultx;

// Show center value in column and row. ( column , row )

 Edit13->Text=(IntToStr(resultx)+","+IntToStr(resulty));

 x=0;
 y=0;

// Show center point to the picture.

 for(int y=positionY-2;y<positionY+3;y++)
 {
     for(int x=positionX-2;x<positionX+3;x++)
     {
     Image1->Canvas->Pixels[x][y]=clWhite;
     }
 }
}

else  Edit14->Text=("End");

// In this case , the mark is white and the background is black.

if (countwhite < countblack)
{
x=0;
y=0;
count=0;

// Noise Reduction.

 for(int y=0;y<60;y++)
 {
     for (int x=0;x<60;x++)
          {
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y-1])  count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y+1])  count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y-1])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y+1])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y-1])  count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y+1])  count++;
     else count=count;
     if (count>=4) Image1->Canvas->Pixels[x][y] = Image1->Canvas-
>Pixels[x][y];
     else Image1->Canvas->Pixels[x][y] = clBlack;
     count=0;
     }
 }
 x=0;
 y=0;
 count=0;

// Store pixels which color is white.Find pixels in row.

 for(int y=0;y<60;y++)
 {
     for (int x=0; x<60;x++)
          {
          if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
          }
 y1[y]=count;
 count=0;
 }

// Find the maximum value of white pixels.

 count=0;
 for(int y=0;y<60;y++)
 {
 if (y1[y]>count) {count=y1[y];}
 }

// Find row's position that has maximum white pixels.Find forward.

 int tmp1,tmp2;
 y=0;
 while (y1[y]!=count) y++;
 tmp1=y;

// Find row's position that has maximum white pixels.Find backward.

 y=60;
 while (y1[y]!=count) y--;
 tmp2=y;

 // Find the middle position.

 int resulty;
 if (tmp1!=tmp2) {resulty=(tmp1+tmp2)/2;}
 else resulty=tmp1;
 positionY=resulty;

// Store pixels which color is white.Find pixels in column.

 count=0;
 for(int x=0;x<60;x++)
 {
          for (int y=0; y<60;y++)
          {
          if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
          }
 y1[x]=count;
 count=0;
 }

// Find the maximum value of white pixels.

 count=0;
 for(int x=0;x<60;x++)
 {
 if (y1[x]>count) {count=y1[x];}
 }

// Find column's position that has maximum white pixels.Find forward.

 x=0;
 while (y1[x]!=count) x++;
tmp1=x;
// Find column's position that has maximum white pixels.Find
backward.

 x=60;
 while(y1[x]!=count) x--;
```

```
 tmp2=x;

// Find the middle position.

 int resultx;
 if (tmp1!=tmp2) {resultx=(tmp1+tmp2)/2;}
 else resultx=tmp1;
 positionX=resultx;

// Show center value in column and row. ( column , row )

 Edit13->Text=(IntToStr(resultx)+","+IntToStr(resulty));

 x=0;
 y=0;

// Show center point to the picture.

 for(int y=positionY-2;y<positionY+3;y++)
 {
     for(int x=positionX-2;x<positionX+3;x++)
     {
     Image1->Canvas->Pixels[x][y]=clBlack;
     }
 }
}
else  Edit14->Text=("End");

// In this case , the mark is white or black , the background is also.

if (countwhite == countblack)
{
x=0;
y=0;
count=0;

// Noise Reduction.

 for(int y=0;y<60;y++)
 {
     for (int x=0;x<60;x++)
          {
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y-1]) count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y+1]) count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y-1])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y+1])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y-1]) count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y+1]) count++;
     else count=count;
     if (count>=4) Image1->Canvas->Pixels[x][y] = Image1->Canvas-
>Pixels[x][y];
     else Image1->Canvas->Pixels[x][y] = clBlack;

     count=0;
     }
 }
 x=0;
 y=0;
 count=0;

// Store pixels which color is white.Find pixels in row.

 for(int y=0;y<60;y++)
 {
     for (int x=0; x<60;x++)
          {
          if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
          }
 y1[y]=count;
 count=0;
 }

// Find the maximum value of white pixels.

 count=0;
 for(int y=0;y<60;y++)
 {
 if (y1[y]>count) {count=y1[y];}
 }

// Find row's position that has maximum white pixels.Find forward.

 int tmp1,tmp2;
 y=0;
 while (y1[y]!=count) y++;
 tmp1=y;

// Find row's position that has maximum white pixels.Find backward.

 y=60;
 while (y1[y]!=count) y--;
 tmp2=y;

// Find the middle position.

 int resulty;
 if (tmp1!=tmp2) {resulty=(tmp1+tmp2)/2;}
 else resulty=tmp1;
 positionY=resulty;

// Store pixels which color is white.Find pixels in column.

 count=0;
 for(int x=0;x<60;x++)
 {
          for (int y=0; y<60;y++)
          {
          if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
          }
 y1[x]=count;
 count=0;
 }

// Find the maximum value of white pixels.

 count=0;
 for(int x=0;x<60;x++)
 {
 if (y1[x]>count) {count=y1[x];}
 }

// Find column's position that has maximum white pixels.Find forward.

 x=0;
 while (y1[x]!=count) x++;
tmp1=x;

// Find column's position that has maximum white pixels.Find
backward.

 x=60;
 while(y1[x]!=count) x--;
 tmp2=x;

// Find the middle position.
 int resultx;
 if (tmp1!=tmp2) {resultx=(tmp1+tmp2)/2;}
 else resultx=tmp1;
 positionX=resultx;
```

```
// Show center value in column and row. ( column , row )

 Edit13->Text=(IntToStr(resultx)+","+IntToStr(resulty));

 x=0;
 y=0;

// Show center point to the picture.

 for(int y=positionY-2;y<positionY+3;y++)
 {
     for(int x=positionX-2;x<positionX+3;x++)
     {
     Image1->Canvas->Pixels[x][y]=clBlack;
     }
 }
}
else {Edit14->Text=("End");}
     SemiAuto();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::ScrollBar3Change(TObject *Sender)
{

// Thresholding the picture which captured from camera.

int Threshold;
Threshold = ScrollBar3->Position;

int x,y,b;
 for (y=0;y<60;y++)
 {
     for (x=0;x<60;x++)
     {
     b=ImageEnView1->Bitmap->Canvas->Pixels[x+30][y+20];
     b=b&0x0000ff00;
     b=b>>8;
     b=256-b;

     if (b>=Threshold)
      {Image1->Canvas->Pixels[x][y]=clBlack;}
     else
      {Image1->Canvas->Pixels[x][y]=clWhite;}
     }
 }
}
//---------------------------------------------------------------------------

void TForm1::MovePlot()
{
  //Move Plotter procedure.

  FILE *fp;
  char c;
  change=1;
  ScrollBarX->Position=plotx;
  ScrollBarY->Position=-ploty;
  Edit23->Text=-ScrollBarY->Position;
  Edit26->Text=ScrollBarX->Position;
  change=0;

 //If Plotter is printing, it's busy.
 //Ploting is prohibit and set busy.

 if (status==0)
 {
 fp = fopen("marktmp.tmp", "w");
 fprintf(fp,"IN;\n");
 fprintf(fp,"VS32,1;\n");
 fprintf(fp,"SP1;\n");
 fprintf(fp,"LT;\n");
 fprintf(fp,"PU%d %d;",plotx,ploty);
 fclose(fp);
 mvplot=1;
 Plot1();
 Plot2();
 }
 else
  busy=1;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Mark1Click(TObject *Sender)
{
     setm1clk();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Mark2Click(TObject *Sender)
{
     setm2clk();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Mark3Click(TObject *Sender)
{
     setm3clk();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SetCamOffsetClick(TObject *Sender)
{
   //Set and show camera offset position.

   Edit16->Text=StrToInt(ScrollBarX->Position)-3500;
   Edit17->Text=-StrToInt(ScrollBarY->Position)-4000;
}
//---------------------------------------------------------------------------

void TForm1::Plot()
{
  //This procedure will get all mark positions,
  //calculate rotation and transpose co-ordinates
  //then plot them.

  char c,d[5];
  int x[1000],y[1000],j,k,i,l;
  int xplot[1000],yplot[1000];
  double angle2,xx,yy,xx1,yy1,xx2,yy2;
  FILE *hp,*fp;

  //Read dx, dy values.

  dx=StrToInt(Edit9->Text);
  dy=StrToInt(Edit10->Text);

  //Read positions from plottemp.tmp,
  //transpose co-ordinates and save to plot.tmp.

  j=1;
  fp = fopen("plottemp2.tmp", "r");
  hp = fopen("plot.tmp", "w");
  while (j<=12)
  {
   fscanf(fp, "%c", &d[j]);
   fprintf(hp, "%c", d[j]);
   j=j++;
  }
  j=1;
  while (j<=9)
  {
   fscanf(fp, "%c", &d[j]);
   fprintf(hp, "%c", d[j]);
   j=j++;
  }
  j=1;
  l=1;
  do
  {
  fscanf(fp, "%c", &d[j]);
  if (d[j]=='P')
   {
   fscanf(fp, "%c%d%c%d%c%c",&d[j+1],&x[l],&d[j+2],&y[l],&d
[j+3],&d[j+4]);

  //Modify to initial positions at bottom-left.
```

```
  x[l]=x[l]+dx-minx;
  y[l]=y[l]+dy-miny;
  xx=x[l];
  yy=y[l];

  //Change positions to polar form.
  //Add rotation angle and return values to
  //rectangular form.

  if (x[l]==0)
   angle2=(3.1416/2)+angle;
  else
   angle2=atan2(yy,xx)+angle;


   //Use Mark2(xm2, ym2) to be plot reference.

   xplot[l]=(sqrt(x[l]*x[l]+y[l]*y[l]))*cos(angle2)+xm2;
   yplot[l]=(sqrt(x[l]*x[l]+y[l]*y[l]))*sin(angle2)+ym2;

   //Save to plot.tmp.

   fprintf(hp, "%c%c%d%c%d%c%c", d[j],d[j+1],xplot[l],d[j+2],yplot
[l],d[j+3],d[j+4]);

   if ((xplot[l]>6950)||(yplot[l]>8550))
   {
     limit=1;
   }
   if ((xplot[l]<0)||(yplot[l]<0))
   {
     limit=1;
   }

   l=l++;
   }
  else
   {
   fprintf(hp, "%c", d[j]);
   do
     {
     fscanf(fp, "%c", &d[j]);
     fprintf(hp, "%c", d[j]);
     }
   while (d[j]!=';');
     {
     fscanf(fp, "%c", &d[j]);
     fprintf(hp, "%c", d[j]);
     }
   }
  }
  while (!feof(fp));
  fclose(fp);
  fclose(hp);

  //This is a plotting process.

  if (limit==0)
  {
  Plot1();
  Plot2();

  Mark1->Down=false;
  Mark2->Down=false;
  Mark3->Down=false;

   CheckPlot();
  }
  else
  {
  ProcessPanel->Caption="Plot area exceeded the limit.";
  if ((nxx==nx-1)&&(nyy==ny-1))
   {
   MessageBox(NULL, "Plot size exceed. Press OK to cancel
Plotting", "ERROR!", MB_OK | MB_ICONSTOP);
   limit=0;
   CheckPlot();
   }
  else
   {
   int Result = MessageBox(NULL, "Plot size exceed. Press OK to
skip curently plotting. Press CANCEL to cancel Plotting", "ERROR!",
MB_OKCANCEL | MB_ICONWARNING);
   if (Result==IDOK)
     CheckPlot();
   if (Result==IDCANCEL)
     {
     nxx=nx-1;
     nyy=ny-1;
     CheckPlot();
     }
   }
  Mark1->Down=false;
  Mark2->Down=false;
  Mark3->Down=false;
  limit=0;
  }
}
//---------------------------------------------------------------------------

void TForm1::BeforePlot()
{
  //Set Mark1.

  xoffset=StrToInt(Edit16->Text);
  yoffset=StrToInt(Edit17->Text);
  nx=StrToInt(Edit24->Text);
  ny=StrToInt(Edit25->Text);

  if ((nxx==0)&&(nyy==0))
   ProcessPanel->Caption="Please Enter nx, ny and Set Mark1";
  else
  {
   absmm=sqrt(nxx*nxx*mmx*mmx+nyy*nyy*mmy*mmy);

  if (nxx*mmx==0)
   angle3=(3.141/2)+angle;
  else
  {
   yyy=nyy*mmy;
   xxx=nxx*mmx;
   angle3=(atan2(yyy,xxx))+angle;
  }

  absmmx=absmm*cos(angle3);
  absmmy=absmm*sin(angle3);

   plotx=x1m+absmmx+xoffset;      //Move to next label's Mark1.
   ploty=y1m+absmmy+yoffset;
  if ((plotx>6950)||(ploty>8550))
  {
   CheckLimit();
  }
  else
  {
  if ((plotx<0)||(ploty<0))
  {
   CheckLimit();
  }
  else
  {
  MovePlot();
  ProcessPanel->Caption="Please Set Mark1";
   if (CheckBox2->Checked==true)
   {
    cauto=1;
   }
  }
  }
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SetMXClick(TObject *Sender)
{
  //Set X-distance between labels(mx).
```

```
  int xa,ya;
  xoffset=StrToInt(Edit16->Text);
  yoffset=StrToInt(Edit17->Text);
  xm2=StrToInt(Edit4->Text);
  ym2=StrToInt(Edit7->Text);
  xa=StrToInt(ScrollBarX->Position)-xoffset;
  ya=StrToInt(-ScrollBarY->Position)-yoffset;
  mmx=sqrt((xa-xm2)*(xa-xm2)+(ya-ym2)*(ya-ym2));
  Edit11->Text=IntToStr(mmx);

  if (ny>1)
  {
   plotx=xm1+xoffset;
   ploty=ym1+yoffset;
   MovePlot();
   ProcessPanel->Caption="Please Set my.";
  }
  else
  {
   if (CheckBox1->Checked)
     DxDy();
   else
     Plot();
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SetMYClick(TObject *Sender)
{
  //Set Y-distance between labels(my).

  int xa,ya;
  xoffset=StrToInt(Edit16->Text);
  yoffset=StrToInt(Edit17->Text);
  xm2=StrToInt(Edit4->Text);
  ym2=StrToInt(Edit7->Text);
  xa=StrToInt(ScrollBarX->Position)-xoffset;
  ya=StrToInt(-ScrollBarY->Position)-yoffset;
  mmy=sqrt((xa-xm2)*(xa-xm2)+(ya-ym2)*(ya-ym2));
  Edit12->Text=IntToStr(mmy);

  if (CheckBox1->Checked)
    DxDy();
  else
    Plot();
}
//---------------------------------------------------------------------------


void TForm1::ScrollBarChange()
{
  //Change plotter X,Y-position by using ScrollBarX,Y.

  if (change==0)
  {
   int x,y;
   FILE *fp,*hp;
   char c;
//   Sleep(100);
   Edit23->Text=-ScrollBarY->Position;
   Edit26->Text=ScrollBarX->Position;
   plotx=ScrollBarX->Position;
   ploty=-ScrollBarY->Position;
   MovePlot();
  }
}
//---------------------------------------------------------------------------

void TForm1::ThresChange()
{

// Thresholding the picture which captured from camera.

int Threshold;
Threshold = ScrollBar3->Position;
int x,y,b;
 for (y=0;y<60;y++)
 {
     for (x=0;x<60;x++)
     {
     b=ImageEnView1->Bitmap->Canvas->Pixels[x+30][y+20];
     b=b&0x0000ff00;
     b=b>>8;
     b=256-b;

     if (b>=Threshold)
      {Image1->Canvas->Pixels[x][y]=clBlack;}
     else
      {Image1->Canvas->Pixels[x][y]=clWhite;}
     }
 }
}
//---------------------------------------------------------------------------

void TForm1::Center()
{
int y1[60];
int x,y,positionX,positionY;
int countblack,countwhite;
int count;

// Find that the mark is white or black.
// If the mark is white then background is black.
// If the mark is black then background is white.

ThresChange();

count=0;
countblack=0;
countwhite=0;

for(int y=0;y<60;y++)
{
        for (int x=0; x<60;x++)
        {
        if (Image1->Canvas->Pixels[x][y]==clBlack)
{countblack++;}
    else countwhite++;
        }
}
x=0;
y=0;

// In this case , the mark is black and the background is white.

if (countwhite > countblack)
{
x=0;
y=0;
count=0;

// Noise Reduction.

 for(int y=0;y<60;y++)
 {
  for (int x=0; x<60;x++)
     {
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y-1]) count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y+1]) count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y-1])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y+1])   count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y-1]) count++;
     else count=count;
     if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y])   count++;
```

```
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y+1])  count++;
      else count=count;
      if (count>=4) Image1->Canvas->Pixels[x][y] = Image1->Canvas-
>Pixels[x][y];
      else Image1->Canvas->Pixels[x][y] = clWhite;

      count=0;
      }
  }
  x=0;
  y=0;
  count=0;

// Store pixels which color is black.Find pixels in row.

  for(int y=0;y<60;y++)
  {
      for (int x=0; x<60;x++)
            {
            if (Image1->Canvas->Pixels[x][y]==clBlack) {count++;}
            }
      y1[y]=count;
      count=0;
  }

// Find the maximum value of black pixels.

  count=0;
  for(int y=0;y<60;y++)
  {
  if (y1[y]>count) {count=y1[y];}
  }

// Find row's position that has maximum black pixels.Find forward.

  int tmp1,tmp2;
  y=0;
  while (y1[y]!=count) y++;
  tmp1=y;

// Find row's position that has maximum black pixels.Find backward.

  y=60;
  while (y1[y]!=count) y--;
  tmp2=y;

// Find the middle position.

  if (tmp1!=tmp2) {resulty=(tmp1+tmp2)/2;}
  else resulty=tmp1;
  positionY=resulty;

// Store pixels which color is black.Find pixels in column.

  count=0;
  for(int x=0;x<60;x++)
  {
            for (int y=0; y<60;y++)
            {
            if (Image1->Canvas->Pixels[x][y]==clBlack) {count++;}
            }
  y1[x]=count;
  count=0;
  }

// Find the maximum value of black pixels.

  count=0;
  for(int x=0;x<60;x++)
  {
  if (y1[x]>count) {count=y1[x];}
  }

// Find column's position that has maximum black pixels.Find forward.

  x=0;
  while (y1[x]!=count) x++;
  tmp1=x;

// Find column's position that has maximum black pixels.Find
backward.

  x=60;
  while (y1[x]!=count) x--;
  tmp2=x;

// Find the middle position.

  if (tmp1!=tmp2) {resultx=(tmp1+tmp2)/2;}
  else resultx=tmp1;
  positionX=resultx;

// Show center value in column and row. ( column , row )

  Edit13->Text=(IntToStr(resultx)+","+IntToStr(resulty));

  x=0;
  y=0;

// Show center point to the picture.

  for(int y=positionY-2;y<positionY+3;y++)
  {
      for(int x=positionX-2;x<positionX+3;x++)
      {
      Image1->Canvas->Pixels[x][y]=clWhite;
      }
  }
}

else  Edit14->Text=("End");

// In this case , the mark is white and the background is black.

if (countwhite < countblack)
{
x=0;
y=0;
count=0;

// Noise Reduction.

 for(int y=0;y<60;y++)
 {
      for (int x=0;x<60;x++)
            {
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y-1])  count++;
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y])   count++;
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y+1])  count++;
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y-1])   count++;
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y+1])   count++;
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y-1])  count++;
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y])   count++;
      else count=count;
      if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y+1])  count++;
      else count=count;
      if (count>=4) Image1->Canvas->Pixels[x][y] = Image1->Canvas-
>Pixels[x][y];
      else Image1->Canvas->Pixels[x][y] = clBlack;

      count=0;
      }
```

```
}
x=0;
y=0;
count=0;

// Store pixels which color is white.Find pixels in row.

for(int y=0;y<60;y++)
{
    for (int x=0; x<60;x++)
        {
        if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
        }
y1[y]=count;
count=0;
}

// Find the maximum value of white pixels.

count=0;
for(int y=0;y<60;y++)
{
if (y1[y]>count) {count=y1[y];}
}

// Find row's position that has maximum white pixels.Find forward.

int tmp1,tmp2;
y=0;
while (y1[y]!=count) y++;
tmp1=y;

// Find row's position that has maximum white pixels.Find backward.

y=60;
while (y1[y]!=count) y--;
tmp2=y;

// Find the middle position.

int resulty;
if (tmp1!=tmp2) {resulty=(tmp1+tmp2)/2;}
else resulty=tmp1;
positionY=resulty;

// Store pixels which color is white.Find pixels in column.

count=0;
for(int x=0;x<60;x++)
{
        for (int y=0; y<60;y++)
        {
        if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
        }
y1[x]=count;
count=0;
}

// Find the maximum value of white pixels.

count=0;
for(int x=0;x<60;x++)
{
if (y1[x]>count) {count=y1[x];}
}

// Find column's position that has maximum white pixels.Find forward.

x=0;
while (y1[x]!=count) x++;
tmp1=x;

// Find column's position that has maximum white pixels.Find
backward.

x=60;
while(y1[x]!=count) x--;
tmp2=x;

// Find the middle position.

int resultx;
if (tmp1!=tmp2) {resultx=(tmp1+tmp2)/2;}
else resultx=tmp1;
positionX=resultx;

// Show center value in column and row. ( column , row )

Edit13->Text=(IntToStr(resultx)+","+IntToStr(resulty));

x=0;
y=0;

// Show center point to the picture.

for(int y=positionY-2;y<positionY+3;y++)
{
    for(int x=positionX-2;x<positionX+3;x++)
    {
    Image1->Canvas->Pixels[x][y]=clBlack;
    }
}
}
else  Edit14->Text=("End");

// In this case , the mark is white or black , the background is also.

if (countwhite == countblack)
{
x=0;
y=0;
count=0;

// Noise Reduction.

for(int y=0;y<60;y++)
{
    for (int x=0;x<60;x++)
        {
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y-1]) count++;
    else count=count;
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y])  count++;
    else count=count;
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x-
1][y+1]) count++;
    else count=count;
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y-1])  count++;
    else count=count;
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels[x]
[y+1])  count++;
    else count=count;
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y-1]) count++;
    else count=count;
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y])  count++;
    else count=count;
    if (Image1->Canvas->Pixels[x][y] == Image1->Canvas->Pixels
[x+1][y+1]) count++;
    else count=count;
    if (count>=4) Image1->Canvas->Pixels[x][y] = Image1->Canvas-
>Pixels[x][y];
    else Image1->Canvas->Pixels[x][y] = clBlack;

    count=0;
    }
}
x=0;
y=0;
count=0;

// Store pixels which color is white.Find pixels in row.

for(int y=0;y<60;y++)
{
```

```
    for (int x=0; x<60;x++)
            {
            if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
            }
  y1[y]=count;
  count=0;
  }

 // Find the maximum value of white pixels.

  count=0;
  for(int y=0;y<60;y++)
  {
  if (y1[y]>count) {count=y1[y];}
  }

 // Find row's position that has maximum white pixels.Find forward.

  int tmp1,tmp2;
  y=0;
  while (y1[y]!=count) y++;
  tmp1=y;

 // Find row's position that has maximum white pixels.Find backward.

  y=60;
  while (y1[y]!=count) y--;
  tmp2=y;

 // Find the middle position.

  int resulty;
  if (tmp1!=tmp2) {resulty=(tmp1+tmp2)/2;}
  else resulty=tmp1;
  positionY=resulty;

 // Store pixels which color is white.Find pixels in column.

  count=0;
  for(int x=0;x<60;x++)
  {
            for (int y=0; y<60;y++)
            {
            if (Image1->Canvas->Pixels[x][y]==clWhite) {count++;}
            }
  y1[x]=count;
  count=0;
  }

 // Find the maximum value of white pixels.

  count=0;
  for(int x=0;x<60;x++)
  {
  if (y1[x]>count) {count=y1[x];}
  }

 // Find column's position that has maximum white pixels.Find forward.

  x=0;
  while (y1[x]!=count) x++;
 tmp1=x;

 // Find column's position that has maximum white pixels.Find
 backward.

  x=60;
  while(y1[x]!=count) x--;
  tmp2=x;

 // Find the middle position.

  int resultx;
  if (tmp1!=tmp2) {resultx=(tmp1+tmp2)/2;}
  else resultx=tmp1;
  positionX=resultx;

 // Show center value in column and row. ( column , row )

  Edit13->Text=(IntToStr(resultx)+","+IntToStr(resulty));

  x=0;
  y=0;

 // Show center point to the picture.

  for(int y=positionY-2;y<positionY+3;y++)
  {
      for(int x=positionX-2;x<positionX+3;x++)
      {
      Image1->Canvas->Pixels[x][y]=clBlack;
      }
  }
 }
 else {Edit14->Text=("End");}
 }
 //---------------------------------------------------------------------------

 void __fastcall TForm1::Button6Click(TObject *Sender)
 {
   //Find dx.
   int a;
   double tx,ty,ta;
   tx=StrToInt(ScrollBarX->Position)-xoffset-xm2;
   ty=StrToInt(-ScrollBarY->Position)-yoffset-ym2;
   if (tx==0)
     ta=3.141/2;
   else
     ta=atan2(ty,tx);

   a=sqrt(tx*tx+ty*ty)*cos(ta-angle);
   Edit9->Text=a;

   plotx=xm2+xoffset;
   ploty=ym2+yoffset;
   MovePlot();
   ProcessPanel->Caption= "Please find the lowest point.";
 }
 //---------------------------------------------------------------------------

 void __fastcall TForm1::Button9Click(TObject *Sender)
 {
   //Find dy.
   int b;
   double tx,ty,ta;
   tx=StrToInt(ScrollBarX->Position)-xoffset-xm2;
   ty=StrToInt(-ScrollBarY->Position)-yoffset-ym2;
   if (tx==0)
     ta=3.141/2;
   else
     ta=atan2(ty,tx);

   b=sqrt(tx*tx+ty*ty)*sin(ta-angle);
   Edit10->Text=b-10;

   Plot();
 }
 //---------------------------------------------------------------------------

 void TForm1::DxDy()
 {
   plotx=xm2+xoffset;
   ploty=ym2+yoffset;
   MovePlot();
   ProcessPanel->Caption="Please find the leftest point.";
 }
 //---------------------------------------------------------------------------

 void __fastcall TForm1::Button11Click(TObject *Sender)
 {
   CheckPlot();
 }
 //---------------------------------------------------------------------------
 void TForm1::CheckPlot()
 {
   //Check plot result.

   if ((nxx==nx-1)&&(nyy==ny-1))
```

```
    {
    nxx=0;
    nyy=0;
    ProcessPanel->Caption="Plotting is done.";    //for complete
 plotting.
    Button11->Enabled=false;
    }
   else      //for not complete plotting.
   {
    Button11->Enabled=true;
    if (nxx<nx-1)
     nxx=nxx++;
    else
    {
     nxx=0;
     nyy=nyy++;
    }
    BeforePlot();
   }
}
//---------------------------------------------------------------------------

void TForm1::SemiAuto()
{
    plotx=StrToInt(ScrollBarX->Position)+(29-resultx)*6;
    ploty=StrToInt(-ScrollBarY->Position)-(29-resulty)*6;
    if ((plotx>6950)||(ploty>8550))
    {
     CheckLimit();
    }
    else
    {
    MovePlot();
    tchange=1;
    }

}
//---------------------------------------------------------------------------

void TForm1::Plot1()
{
  //This is the translation steps changing data from HPGL file
  //to pulse, speed and direction format
  //and be saved in file "plotpul.txt.

  FILE *fp;
  FILE *gp;
  FILE *hp;
  int j=1,l,minx,miny,difx,dify,laser=0;
  int pulsex,pulsey,speedx,speedy,phx,plx,shx,slx,phy,ply,shy,sly;
  double speedxd,speedyd;
  int x[1023],y[1023];
  char d[1000],c;

  j=1;
  if (mvplot==0)
    fp = fopen("plot.tmp", "r");
  else
    fp = fopen("marktmp.tmp", "r");
  gp = fopen("plotpul.tmp", "w");
  hp = fopen("plot1.tmp", "w");
  while (j<=21)
  {
    fscanf(fp, "%c", &d[j]);
//  fprintf(hp, "%c", d[j]);
    j=j++;
  }

  j=1;
  l=1;
  do
  {
  fscanf(fp, "%c", &d[j]);
  if (d[j]=='P')
    {
   fscanf(fp, "%c%d%c%d%c%d%c",&d[j+1],&x[l],&d[j+2],&y[l],&d
[j+3],&d[j+4]);

   //Calculate pulse and speed
  config=0;              //config is a control byte
  difx=(x[l]-nowx);
  dify=(y[l]-nowy);
  if (difx<0)
    config=config+1;
  if (dify<0)
    config=config+2;
  nowx=x[l];
  nowy=y[l];
  pulsex=((abs(difx)*5/4));   //adjust pulse to real distance
  pulsey=((abs(dify)*5/4));
  phx=pulsex/256;            //divide value to two bytes
  plx=pulsex%256;
  phy=pulsey/256;
  ply=pulsey%256;

  //This is a speed calculation step by using delay principle

    if ((pulsex)>=(pulsey))
    {
     if (pulsex!=0)
     {
     speedx=65536-850;
     if (dify!=0)
     {
      speedyd=(850*pulsex/pulsey);
     }
     else
     {
      speedyd=65536;
      config=config+8;
     }
     speedy=65536-abs(speedyd);
      if (speedy<0)
      {
       speedy=0;
       config=config+8;
      }
     }
     else
     {
     config=config+12;
     speedx=0;
     speedy=0;
     }
    }
    else
    {
    if (difx!=0)
    {
     speedxd=(850*pulsey/pulsex);
    }
    else
    {
     speedxd=65536;
     config=config+4;
    }
      speedx=65536-abs(speedxd);
      speedy=65536-850;
    if (speedx<0)
    {
      speedx=0;
      config=config+4;
    }
    }

  shx=speedx/256;
  slx=speedx%256;
  shy=speedy/256;
  sly=speedy%256;

  //Save to plot.tmp and plotpul.tmp.
  fprintf(hp, "%c%c%d%c%d%c%c", d[j],d[j+1],x[l],d[j+2],y[l],d
[j+3],d[j+4]);
  if (laser==0)
   {
   laser=1;
   }
```

```
    else
      {
      config=config+16;
      }

fprintf(gp,"%d,%d,%d,%d,%d,%d,%d,%d,%d\n",config,plx,phx,slx,sh
x,ply,phy,sly,shy);

    l=l++;
    }
   else
    {
    fprintf(hp, "%c", d[j]);
    do
      {
      fscanf(fp, "%c", &d[j]);
      fprintf(hp, "%c", d[j]);
      }
    while (d[j]!=';');
      {
      fscanf(fp, "%c", &d[j]);
      fprintf(hp, "%c", d[j]);
      laser=0;
      }
    }
   }
   while (!feof(fp));
      config=128;
      fprintf(gp, "%d", config);

   fclose(fp);
   fclose(hp);
   fclose(gp);

   mvplot=0;
   status=1;

   Edit20->Text="Transfering...";
}
//---------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
  OpenDialog2->Execute();
  Edit18->Text=OpenDialog2->FileName;

  if(Memo2->Lines->Count>0)
  Memo2->Clear();
  Memo2->Lines->LoadFromFile(OpenDialog2->FileName);
}
//---------------------------------------------------------------------

void TForm1::Plot2()
{
  //This is file transfering steps

  FILE *fp;     //Declare Variables
  FILE *hp;
  int a=0;
  char k;
  fp = fopen("plotpul.tmp", "r");       //Open files.
  hp = fopen("plotpul1.tmp", "w");

  do
  {
  fscanf(fp, "%d", &a);
  fscanf(fp, "%c", &k);
  fprintf(hp, "%d", a);
  TransmitCommChar(hComm, a);
  Sleep(1);
  }
  while (!feof(fp));
  fclose(fp);
  fclose(hp);

  if (mvplot==0)
  {
  change=1;
  ScrollBarX->Position=nowx;
  ScrollBarY->Position=-nowy;
  Edit23->Text=-ScrollBarY->Position;
  Edit26->Text=ScrollBarX->Position;
  change=0;
  }
}
//---------------------------------------------------------------------


void TForm1::Busy()
{
  //This part will be called when program try to send data
  //while plotter is busy.

  FILE *hp;
  char c;
 if (busy==1)
 {
  change=1;
  Edit23->Text=-ScrollBarY->Position;
  Edit26->Text=ScrollBarX->Position;
  plotx=StrToInt(ScrollBarX->Position);
  ploty=-StrToInt(ScrollBarY->Position);
  change=0;

  hp = fopen("marktmp.tmp", "w");
  fprintf(hp,"IN;\n");
  fprintf(hp,"VS32,1;\n");
  fprintf(hp,"SP1;\n");
  fprintf(hp,"LT;\n");
  fprintf(hp,"PU%d %d;",plotx,ploty);
  fclose(hp);
  mvplot=1;

  Plot1();
  Plot2();
 }
 mvplot=0;
 busy=0;
}
//---------------------------------------------------------------------

void __fastcall TForm1::Edit20Change(TObject *Sender)
{
  //This will check status whenever plotting is done.

  int a=144,b=32;
  if ((Edit20->Text)=="Ready")
  {
   if (busy==1)
   {
    Busy();
   }
   else
    status=0;
   if (offset==1)    //for camera offset setting
   {
    TransmitCommChar(hComm, a);
    offset=0;
   }
   if (tchange==1)
   {
    Timer1->Enabled=true;
    timer1done=1;
    tchange=0;
   }
   if (cauto==1)
   {
    Auto1();
   }
}
if ((Edit20->Text)=="Initializing..")
{
 change=1;
 ScrollBarY->Position=0;
 change=1;
 ScrollBarX->Position=0;
 Edit23->Text="0";
 Edit26->Text="0";
```

```
    nowx=0;
    nowy=0;
    change=0;
   }
   if ((Edit20->Text)=="Limit Exceed!!!")
   {
    busy=0;
    status=0;
    mvplot=0;
    //show messagebox
    MessageBox(NULL, "Program error. Press OK to Reset Program",
"ERROR!", MB_OK | MB_ICONWARNING);
    {
    change=1;
    ScrollBarY->Position=0;
    change=1;
    ScrollBarX->Position=0;
    Edit23->Text="0";
    Edit26->Text="0";
    nowx=0;
    nowy=0;
    change=0;
    TransmitCommChar(hComm, b);
    }
   }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button12Click(TObject *Sender)
{
 int a=32;
    change=1;
    ScrollBarY->Position=0;
    change=1;
    ScrollBarX->Position=0;
    Edit23->Text="0";
    Edit26->Text="0";
    nowx=0;
    nowy=0;
    change=0;
    TransmitCommChar(hComm, a);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
      Timer1->Enabled=false;
      ThresChange();
      timer1done=0;
}
//---------------------------------------------------------------------------

void TForm1::CheckLimit()
{
    ProcessPanel->Caption="Plot area exceeded the limit.";
    if ((nxx==nx-1)&&(nyy==ny-1))
    {
    MessageBox(NULL, "Plot size exceed. Press OK to cancel
Plotting", "ERROR!", MB_OK | MB_ICONSTOP);
    CheckPlot();
    }
    else
    {
    int Result = MessageBox(NULL, "Plot size exceed. Press OK to
skip curently plotting. Press CANCEL to cancel Plotting", "ERROR!",
MB_OKCANCEL | MB_ICONWARNING);
     if (Result==IDOK)
       CheckPlot();
     if (Result==IDCANCEL)
       {
       nxx=nx-1;
       nyy=ny-1;
       CheckPlot();
       }
    }
    Mark1->Down=false;
    Mark2->Down=false;
    Mark3->Down=false;
}
//---------------------------------------------------------------------------

void TForm1::Auto1()
{
 if (timer1done==1)
 {
  Timer2->Enabled=true;
 }
 else
 {
  Timer3->Enabled=true;
 }
}
//---------------------------------------------------------------------------

void TForm1::Auto2()
{
  autoc++;
  if (autoc==1)
  {
    Center();
    SemiAuto();
  }
  if (autoc==2)
  {
    autoc=0;
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
    Timer2->Enabled=false;
    Auto1();
}
//---------------------------------------------------------------------------

void TForm1::setm1clk()
{
  //Set and show Mark1 position.

  xoffset=StrToInt(Edit16->Text);
  yoffset=StrToInt(Edit17->Text);
  Edit3->Text=StrToInt(Edit26->Text)-xoffset;
  Edit6->Text=StrToInt(Edit23->Text)-yoffset;
  Mark1->Down=true;

  //Prepare to set Mark2.

  if ((nxx==0)&&(nyy==0))
  {
   ProcessPanel->Caption="Please Set mark2";
  }
  else
  {
   plotx=x2m+absmmx+xoffset;     //Move to next label's Mark1.
   ploty=y2m+absmmy+yoffset;

   if ((plotx>6950)||(ploty>8550))
   {
     CheckLimit();
   }
   else
   {
    if ((plotx<0)||(ploty<0))
     {
     CheckLimit();
     }
    else
     {
     MovePlot();
     ProcessPanel->Caption="Please Set Mark2";
     }
   }
  }
//---------------------------------------------------------------------------

void TForm1::setm2clk()
```

```
{
  //Set and show Mark2 position.

  Edit4->Text=StrToInt(Edit26->Text)-xoffset;
  Edit7->Text=StrToInt(Edit23->Text)-yoffset;
  Mark2->Down=true;

  //Prepare to set Mark3.

  if ((nxx==0)&&(nyy==0))
  {
   ProcessPanel->Caption="Please Set mark3";
  }
  else
  {
   plotx=x3m+absmmx+xoffset;     //Move to next label's Mark1.
   ploty=y3m+absmmy+yoffset;
  if ((plotx>6950)||(ploty>8550))
  {
    CheckLimit();
  }
  else
  {
   if ((plotx<0)||(ploty<0))
   {
    CheckLimit();
   }
   else
   {
    MovePlot();
    ProcessPanel->Caption="Please Set Mark3";
   }
  }
  }
}
//---------------------------------------------------------------------

void TForm1::setm3clk()
{
  double xx1,yy1,xx2,yy2;
   //Set and show Mark3 position.

  autoc++;

   Edit5->Text=StrToInt(Edit26->Text)-xoffset;
   Edit8->Text=StrToInt(Edit23->Text)-yoffset;
   Mark3->Down=true;

   //Calculate rotation angle.
   //Get values.

  dx=StrToInt(Edit9->Text);
  dy=StrToInt(Edit10->Text);
  xoffset=StrToInt(Edit16->Text);
  yoffset=StrToInt(Edit17->Text);
  xm1=StrToInt(Edit3->Text);
  ym1=StrToInt(Edit6->Text);
  xm2=StrToInt(Edit4->Text);
  ym2=StrToInt(Edit7->Text);
  xm3=StrToInt(Edit5->Text);
  ym3=StrToInt(Edit8->Text);
  if ((nxx==0)&&(nyy==0))
  {
   x1m=xm1;
   y1m=ym1;
   x2m=xm2;
   y2m=ym2;
   x3m=xm3;
   y3m=ym3;
  }
  yy1=(ym3-ym2);
  xx1=(xm3-xm2);
  xx2=(xm2-xm1);
  yy2=(ym1-ym2);

  //Calculate rotation.

  if ((xx1==0)||(yy2==0))
   angle=(3.141/2);
  else
   angle=(atan2(yy1,xx1)+atan2(xx2,yy2))/2;

  Edit15->Text=angle*180/3.141;

   //Check and prepare to set mx, my.

   nx=StrToInt(Edit24->Text);
   ny=StrToInt(Edit25->Text);
   if ((nxx==0)&&(nyy==0))
   {
    if (nx>1)
     ProcessPanel->Caption="Please Set mx.";
    else
    {
     if (ny>1)
      ProcessPanel->Caption="Please Set my.";
     else
     {
      if (CheckBox1->Checked)
        DxDy();
      else
        Plot();
     }
    }
   }
   else
    Plot();
}
//---------------------------------------------------------------------
void __fastcall TForm1::Timer3Timer(TObject *Sender)
{
  Timer3->Enabled=false;
  Auto2();
}
//---------------------------------------------------------------------
```

โปรแกรมของส่วนควบคุมพล็อตเตอร์(Indexer) ด้วย Assembly

```
tmp     equ     20h
countl  equ     21h
counth  equ     22h
ctx     equ     23h
cty     equ     24h
tlo0    equ     25h
thi0    equ     26h
tlo1    equ     27h
thi1    equ     28h
cnt8    equ     29h
tmp1    equ     2ah
tmp2    equ     2bh
tmp3    equ     2ch
init    equ     2dh
laser   equ     2eh

        org     0000h
        sjmp    start
        org     0003h
        ajmp    limitx
        org     000bh
        ajmp    timer0
        org     0013h
        ajmp    limity
        org     001bh
        ajmp    timer1
        org     0023h
        ajmp    serial

        org     0030h
;initial value and set home

start:  acall   delay
        mov     laser,#00h   ;initial laser checker
        setb    init.0
main:   acall   begin
        jnb     init.0,mloop1
        mov     sbuf,#'i'    ;send status i to com
        acall   ldelay
        acall   home
        acall   hmove
        mov     sbuf,#'a'    ;send status a to com
        acall   begin
        mov     init,#00h

mloop1: jnb     tmp.0,mloop1     ;wait for
finishing serial(use tmp.0 as serial checker)
        jnb     tmp.1,mloop2
        ajmp    bhome1           ;laser control
mloop2: jnb     tmp.2,mloop3
        acall   home
        acall   hmove
        ajmp    bhome1
mloop3:         ajmp        plot

;************************
;subroutine section
;************************

serial: clr     ri
        mov     a,sbuf      ;move data from serial
                            buffer
        jnb     ti,reci1
        clr     ti
        reti
reci1:  movx    @dptr,a     ;move data to ram
                            @DPTR
        mov     r0,a        ;use r0 as sbuf temp

;check the function from control byte

chk1:   cjne    r1,#09h,movdat
chkend: cjne    r0,#128d,ichklas ;check end data
          setb  tmp.0
          reti
ichklas: cjne   r0,#144d,chkhom     ;check laser
          setb  laser.0
          setb  tmp.0
          setb  tmp.1
          reti
chkhom: cjne    r0,#32d,movdat      ;check home
          setb  tmp.0
          setb  tmp.2
          reti

;move data to Ext.RAM
movdat: dec     r1
          inc   dptr
          cjne  r1,#00h,movdat1
          mov   r1,#09h
movdat1:  reti

;after finishing moving data,it's a good time to
plot.
plot:   clr     tr1
        clr     tf1
        setb    p3.2
        setb    p3.3
        clr     ie0
        clr     ie1
        mov     ie,#08fh    ;enable interrupt
                            timer0,1, ext0, ext1
        dec     dpl         ;decrease DPTR-1
        jnc     decd1
        dec     dph
        clr     c
decd1:  mov     countl,dpl
        mov     counth,dph
```

```
        mov   dptr,#0000h
plot1:  movx  a,@dptr
        mov   tmp,a
chklas: jnb   tmp.4,nolas   ;check laser
                            on(1)/off(0)
        setb  p3.5          ;laser on p3.5
        sjmp  chkdix
nolas:  clr   p3.5
chkdix: jnb   tmp.0,dix_p   ;check x direction
                            0(+)/1(-)
        setb  p1.1
        sjmp  chkdiy
dix_p:  clr   p1.1
chkdiy: jnb   tmp.1,diy_p   ;check y direction
                            0(+)/1(-)
        setb  p1.3
        sjmp  chkctx
diy_p:  clr   p1.3
chkctx: jnb   tmp.2,ctxon   ;check x counter
                            0(on)/1(off)
        mov   ctx,#01h
        sjmp  chkcty
ctxon:  mov   ctx,#00h
chkcty: jnb   tmp.3,ctyon   ;check y counter
                            0(on)/1(off)
        mov   cty,#01h
        sjmp  plot2
ctyon:  mov   cty,#00h

plot2:  inc   dptr          ;move x values
        movx  a,@dptr
        mov   r1,a
        inc   dptr
        movx  a,@dptr
        mov   r2,a
        inc   dptr
        movx  a,@dptr
        mov   tlo0,a
        inc   dptr
        movx  a,@dptr
        mov   thi0,a
        inc   dptr

        movx  a,@dptr       ;move y values
        mov   r3,a
        inc   dptr
        movx  a,@dptr
        mov   r4,a
        inc   dptr
        movx  a,@dptr
        mov   tlo1,a
        inc   dptr
        movx  a,@dptr
        mov   thi1,a

        setb  p1.0          ;set pulsex to 1
        setb  p1.2          ;set pulsey to 2
        mov   r7,#02h       ;check pulsex,y
                            complete
        mov   r5,#02h       ;check  rising
(1)/falling(0) period of pulsex
        mov   r6,#02h       ;check rising
(1)/falling(0) period of pulsey

plot3:  mov   tmod,#11h     ;set timer0,1
     to mode1(16-bit) for controlling x,y speed
        mov   tl0,tlo0
        mov   th0,thi0
        mov   tl1,tlo1
        mov   th1,thi1
        mov   r0,ctx
        cjne  r0,#01h,next1 ;check if
                            timer0(x) is off, decrease wait1
        dec   r7
        sjmp  next2
next1:  setb  tr0
next2:  mov   r0,cty
        cjne  r0,#01h,next3
        dec   r7            ;check if timer1(y) is
off,                        decrease wait1
        sjmp  loop1
next3:  setb  tr1
loop1:  cjne  r7,#00h,loop1 ;wait until
                            pulsex,y complete

chkct:  mov   a,dph         ;check last plot data
                            by comparing with counth,l
        cjne  a,counth,plot4
        mov   a,dpl
        cjne  a,countl,plot4
        clr   p3.5          ;turn off laser
        clr   tr0
        clr   tr1
        clr   tf0
        clr   tf1
bhome1: jb    tmp1.0,bhome2
        acall begin         ;not back to home
                            position
        mov   sbuf,#'a'
        ajmp  bhome3
bhome2: acall begin
        mov   sbuf,#'l'
        mov   tmp1,#00h
bhome3: ajmp  mloop1
```

```
plot4:  inc     dptr
        ajmp    plot1

timer0: cpl     p1.0        ;timer 0 for x speed
        clr     tr0
        dec     r5
        cjne    r5,#00h,nodecx
chkpux: dec     r1          ;check last pulsex
        cjne    r1,#00h,decx3
        cjne    r2,#00h,decx2
        dec     r7
        reti
decx3:  jnc     decx1
        clr     c
decx2:  dec     r2
decx1:  mov     r5,#02h
nodecx: mov     th0,thi0
        mov     tl0,tlo0
        setb    tr0
        reti

timer1: cpl     p1.2        ;timer 1 for y speed
        clr     tr1
        dec     r6
        cjne    r6,#00h,nodecy
chkpuy: dec     r3          ;check last pulsey
        cjne    r3,#00h,decy3
        cjne    r4,#00h,decy2
        dec     r7
        reti
decy3:  jnc     decy1
        clr     c
decy2:  dec     r4
decy1:  mov     r6,#02h
nodecy: mov     th1,thi1
        mov     tl1,tlo1
        setb    tr1
        reti

home:   setb    p1.1
        setb    p1.3
        cpl     p3.5
        setb    p3.2
        jb      p3.2,home1
        setb    p1.4
        jnb     p1.4,home2
home1:  acall   delay
        cpl     p1.0
        setb    p1.4
        jb      p1.4,home1
home2:  setb    p3.3
        jb      p3.3,home4
        setb    p1.6
        jnb     p1.6,home3
home4:  acall   delay
        cpl     p1.2
        setb    p1.6
        jb      2fh.6,home2
home3:  clr     p1.1
        clr     p1.3
        ret
home5:  ret

delay:  mov     r0,#00h
delay1: djnz    r0,delay1
        ret

limitx: mov     r1,#01h;set to end of loop
        mov     r2,#00h
        mov     r3,#01h
        mov     r4,#00h
        mov     tlo0,#24d       ;set speed
        mov     thi0,#252d
        mov     tlo1,#24d
        mov     thi1,#252d
        mov     dph,counth      ;set end of data
        mov     dpl,countl
        setb    tmp1.0
        reti
limity: sjmp    limitx

begin:  mov     r1,#09h
        clr     c               ;clear C flag
        mov     dptr,#00d       ;initial DPTR
        clr     p3.4            ;set status indicator
        jnb     laser.0,begin1
        cpl     p3.5            ;toggle laser
        sjmp    begin2
begin1: clr     p3.5            ;turn off laser
begin2: mov     laser,#00h
        mov     tmod,#20h
        mov     th1,#-05h
        mov     tl1,#-05h

        setb    tr1             ;start timer1
        mov     scon,#50h
        mov     ie,#90h
        mov     ip,#0ah
        setb    it0             ;edge trig
        setb    it1
        mov     tmp,#00h
        mov     tmp1,#00h
        ret
```

```
ldelay:  mov     r2,#0ah
lloop3:  mov     r1,#00h
lloop2:  mov     r0,#00h
lloop1:  djnz    r0,lloop1
         djnz    r1,lloop2
         djnz    r2,lloop3
         ret

mdelay: mov     r1,#02h
mdelay1:        mov     r0,#00h
mdelay2:        djnz    r0,mdelay2
        djnz    r1,mdelay1
        ret

hmove: clr      p1.0
       clr      p1.1
       clr      p1.2
       clr      p1.3
       mov      r2,#200d
hloop: acall    mdelay
       cpl      p1.0
       cpl      p1.2
       djnz     r2,hloop
       ret

end
```

รายการอุปกรณ์ที่ใช้

IC

6264 - RAM 8 KByte	1	ตัว
74LS08 -AND Gate 2 input	1	ตัว
74LS245 – BUFFER	1	ตัว
74LS373 – LATCH	1	ตัว
7508 - 5V Regulator	1	ตัว
89C51 - Microcontroller	1	ตัว
MAX232	1	ตัว
Bridge Diode	1	คัว

Socket

14 ขา	1	ตัว
18 ขา	1	ตัว
20 ขา	2	ตัว
28 ขา	1	ตัว
40 ขา	1	ตัว

ตัวต้านทาน

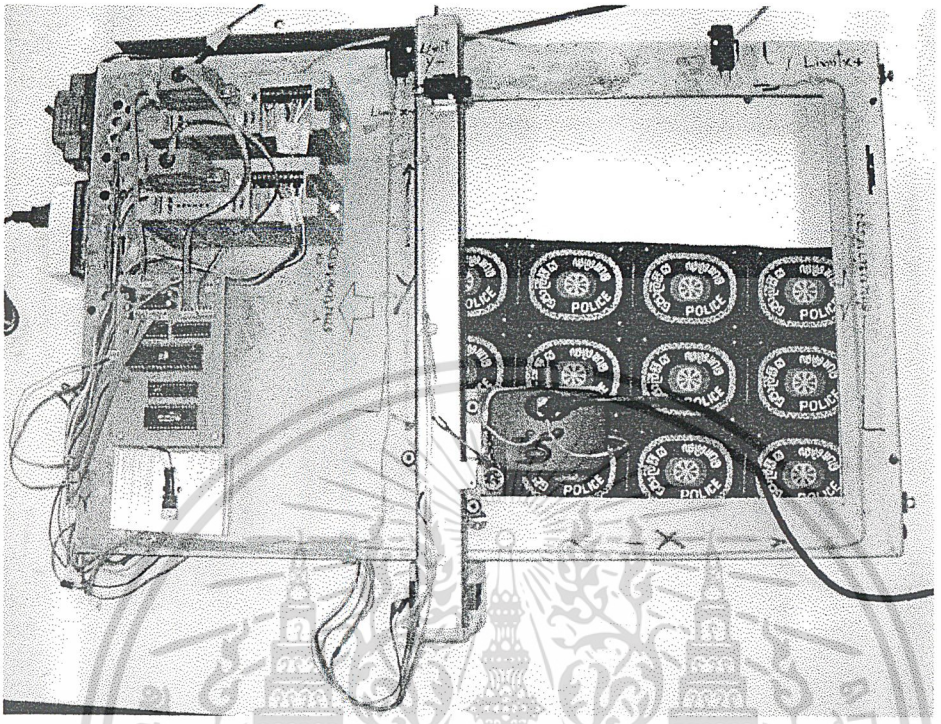
1.5 kΩ	1	ตัว
8.2 kΩ	1	ตัว
R Packed(10k * 8)	2	ตัว

ตัวเก็บประจุ

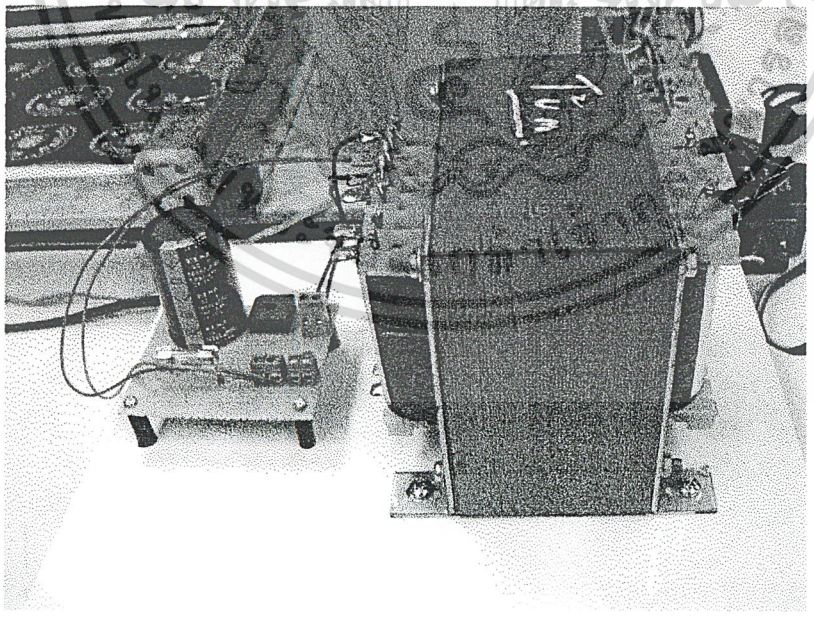
30 pF	2	ตัว
0.01 uF	3	ตัว
1 uF/16V	5	ตัว
10 uF/16V	1	ตัว
4700 uF/16V	1	ตัว

อื่นๆ

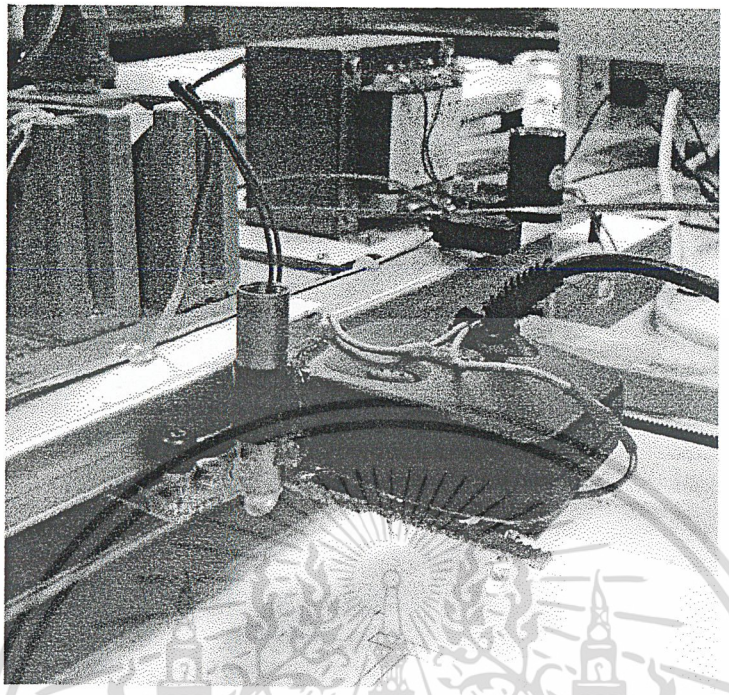
หม้อแปลง 220V/20V	1	ตัว
Crystal 18.432Mhz	1	ตัว
สวิตช์	1	ตัว
Laser Pointer	1	ตัว
Connector		
แผ่นปรินท์อเนกประสงค์		
สายไฟ		



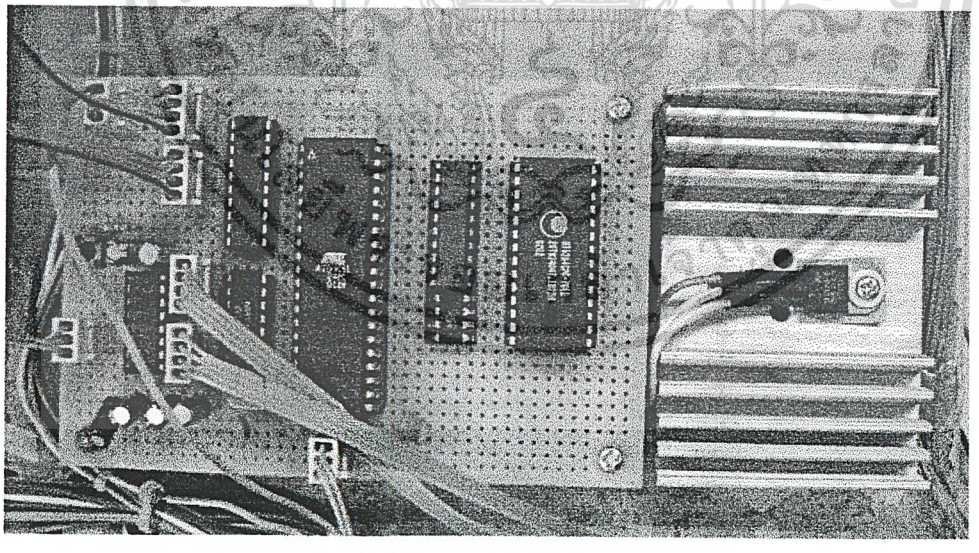
แสดงเครื่องพล็อตเตอร์ที่สร้างขึ้นเอง



แสดงภาคจ่ายไฟของพล็อตเตอร์



แสดงการติดตั้งกล้อง, Laser Pointer และ LED



แสดงส่วนควบคุม (Indexer) ของพล็อตเตอร์

1.3 MH−GL

コマンド			データフォーマット	コ　マ　ン　ド　機　能
直線移動コマンド	PLOT ABSOLUTE	PA	PA　X_1, Y_1 (, X_2, Y_2・・・・) ;	ヘッドを絶対座標系で移動します。
	PEN DOWN	PD	PD　X_1, Y_1 (, X_2, Y_2・・・・) ;	ヘッドのペンをダウンして移動します。
	PLOT RELATIVE	PR	PR　X_1, Y_1 (, X_2, Y_2・・・・) ;	ヘッドを相対座標系で移動します。
	PEN UP	PU	PU　X_1, Y_1 (, X_2, Y_2・・・・) ;	ヘッドのペンをアップさせて移動します。
	SELECT PEN	SP	SP　Pen number ;	ペンを選択します。
円・長方形・ポリゴンコマンド	ARC ABSOLUTE	AA	AA　x, y, angle (, chord tolerance) ;	絶対座標（X, Y）を中心座標として角度（θ_1）の円、円弧を（θ_2）の分割角度で作図します。
	ARC RELATIVE	AR	AR　x, y, angle (, chord tolerance) ;	相対座標（X, Y）を中心座標として角度（θ_1）の円、円弧を（θ_2）の分割角度で作図します。
	CIRCLE	CI	CI　radius (, chord tolerance) ;	全円を作図します。
	EDGE RECTANGLE ABSOLUTE	EA	EA　x, y ;	絶対座標で与えられた四角形の外周を作図します。
	EDGE POLYGON	EP	EP　;	ポリゴンバッファに登録されているポリゴンの外周を作図します。
	EDGE RECTANGLE RELATIVE	ER	ER　x, y ;	相対座標で与えられた四角形の外周を作図します。
	EDGE WEDGE	EW	EW　radius, start angle, sweep angle (, chord tolerance) ;	現在位置を中心とする扇形の外周を作図します
	FILL POLYGON	FP	FP　;	ポリゴンバッファに登録されているポリゴンの内部を塗りつぶします。
	POLYGON MODE	PM	PM　n ;	図形をポリゴンバッファに登録します。
	FILL RECTANGLE ABSOLUTE	RA	RA　x, y ;	絶対座標で与えられた四角形の塗りつぶしを作図します。
	FILL RECTANGLE RELATIVE	RR	RR　x, y ;	相対座標で与えられた四角形の塗りつぶしを作図します。
	FILL WEDGE	WG	WG　radius, start angle, sweep angle (, chord tolerance) ;	現在位置を中心とする扇形の塗りつぶしをします。
その他の図形作図コマンド	SYMBOL MODE	SM	SM　character (characeter) ;	シンボルモードの設定、解除を行います。
	X−TICK	XT	XT　;	X軸に垂直な直線を作図します。
	Y−TICK	YT	YT　;	Y軸に垂直な直線を作図します。

	コマンド		データフォーマット	コ マ ン ド 機 能
文字作図コマンド	BUFFERED LABEL STRING	BL	BL (CCC･･･C) term	指定された文字列を文字バッファに入れます。
	ALTERNATE CHARACTER SET	CA	CA set;	補助文字セットを設定します。
	CHARACTER CHORD ANGLE	CC	CC chord angle;	円弧文字の滑らかさを指定します。
	CHARACTER SELECTION MODE	CM	CM swith mode (, fallback mode) ;	文字セット選択モードを指定します。
	CHARACTER PLOT	CP	CP space, line;	指定された文字セル分ペンを移動します。
	STANDARD CHARACTER SET	CS	CS set;	標準文字セットを設定します。
	ABSOLUTE DIRECTION	DI	DI run, rise;	作図される文字列の傾きを説明します。
	DOWNLOADABLE CHARACTER	DL	DL character number (, pen control) , x-coordinate, y-coordinate (, ･･･,) (pen control) (, ･･･,) ;	ユーザ文字セット（文字セット－1）を定義します。
	RELATIVE DIRECTION	DR	DR run, rise;	作図する文字列の傾きを説明します。
	DESIGNATE CHARACTER SET INTO SLOT	DS	DS slot, set;	スロットに文字セットを設定します。
	DEFINE TERMINATOR	DT	DT term;	LB, BL, WD命令のターミネータを設定します。
	DIRECTION VERTICAL	DV	DV n ;	文字列を横書きで作図するか、縦書きで作図するか指定します。
	EXTRA SPACE	ES	ES space (, line) ;	文字間隔と行間隔を設定します。
	INVOKE CHARACTER SLOT	IV	IV slot (, left) ;	GLまたはGRにスロットを割り付けます。
	BUFFERED KANJI LABEL STRING	KB	KB JIS code (, ･･･) ;	漢字を文字バッファに登録します。
	KANJI LABEL	KL	KL JIS code (, ･･･) ;	漢字を作図します。
	LABEL	LB	LB (ccc･･･c) term	文字を作図します。
	LABEL ORIGIN	LO	LO position number;	文字列の作図開始点を指定します。
	OUTPUT LABEL LENGTH	OL	OL ;	文字バッファ内の文字列に関する情報を問い合わせます。
	PRINT BUFFERED LABEL	PB	PB ;	文字バッファに登録されている文字列を作図します。
	SELECT ALTERNATE SET	SA	SA ;	補助文字セットを選択します。
	ABSOLUTE CHARACTER SIZE	SI	SI width, height;	文字幅と文字高さを指定します。
	CHARACTER SLANT	SL	SL tangent;	文字に傾斜をつけます。
	RELATIVE CHARACTER SIZE	SR	SR width, height;	文字幅と文字高さを指定します。
	SELECT STANDARD SET	SS	SS ;	標準文字セットを選択します。
	USER DEFINED CHARACTER	UC	UC (pen control,) x-increment, y-increment (, ･･･) (, pen control) (, ･･･) ;	ユーザ文字を作図します。

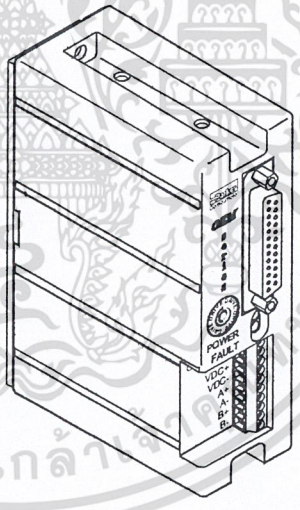
付録１　コマンド一覧表

コマンド			データフォーマット	コ　マ　ン　ド　機　能
作図指定コマンド	CHORD TOLERANCE	CT	CT　n ;	円弧と多角形近似方法を設定します。
	FILL TYPE	FT	FT　type (, spacing(, angle)) ;	塗りつぶしのタイプを指定します。
	LINE TYPE	LT	LT　n (, p) ; LT　－7, 1, dt1, ut ; LT　－7, 2, dt1, ut, dt2 ; LT　－7, 3, dt1, ut, dt2 ;	作図の線種を指定します。
	PEN THICKNESS	PT	PT　thickness ;	ペタ塗りの時の塗りつぶし線の間隔を指定します。
	TICK LENGTH	TL	TL　tp (, tn) ;	XT, YT命令で作図する直線の長さを指定します。
	USER DEFINED FILL TYPE	UF	UF　gap1 (, gap2･･･, gap20) ;	ユーザ定義の塗りつぶしのパターンを指定します。
プロッタ制御コマンド	AUTOMATIC PEN OPERATIONS	AP	AP　n ;	ペン制御の各機能を設定します。
	ACCELERATION SELECT	AS	AS　acceleration (, pen number) ;	作図時の加速度を指定します。
	CURVED LINE GENERATOR	CV	CV　n (, delay) ;	スムージング処理のオン・オフを指定します。
	DEFAULT	DF	DF　;	プロッタをデフォルト状態にします。
	FORCE SELECT	FS	FS　force (, pen number) ;	作図時の筆圧を指定します。
	GRAPHICS MEMORY	GM	GM　polygon buffer (, downloadable character buffer (, 0, dummy1 (, dummy2))) ;	入力バッファを除いた各種バッファのサイズを変更します。
	GROUP PEN	GP	GP　group number (, pen number (, number of pens (, length))) ;	ストッカ内の各ペンをグループに分類します。
	INITIALIZE	IN	IN　; IN　－1 ;	プロッタを初期化状態にします。
	NOT READY	NR	NR　;	プロッタをローカル状態にします。
	SELECT PEN GROUP	SG	SG　group number ;	GPコマンドで設定されたペングループを選択します。
	VELOCITY SELECT	VS	VS　velocity (, pen number) ;	作成時の移動速度を指定します。
座標系コマンド	INPUT P1 AND P2	IP	IP　P1x, P1y (, P2x, P2y) ;	スケーリングポイントP1, P2を設定します。
	INPUT WINDOW	IW	IW　X1, Y1, X2, Y2 ;	ウィンドウ領域を設定します。
	OUTPUT HARD-CLIP LIMITS	OH	OH　;	ハードクリップ領域の左下と右上の座標値を問い合わせます。
	OUTPUT P1 AND P2	OP	OP　;	スケーリングポイントP1, P2の座標値を問い合わせます。
	OUTPUT WINDOW	OW	OW　;	作図可能領域の左下および右上の座標値を問い合わせます。
	ROTATE COORDINATE SYSTEM	RO	RO　n ;	プロッタ座標系を90度回転させたり、回転を解除したりします。
	SCALE	SC	SC　Xmin, Xmax, Ymin, Ymax, (, type (, left, bottom)) ; SC　Xmin, Xfactor, Ymin, Yfactor, type ;	ユーザ座標系の設定、または解除をします。
デジタイズコマンド	DIGITIZE CLEAR	DC	DC　;	デジタイズモードを解除します。
	DIGITIZE POINT	DP	DP　;	プロッタをデジタイズモードにします。
	OUTPUT DIGITIZED POINT	OD	OD　;	デジタイズデータを問い合わせます。

コマンド			データフォーマット	コマンド機能
キーボードモードコマンド	GROUP COUNT	GC	GC　count;	グループ番号を指定します。
	DEFINE KEY	KY	KY　key (, function) ;	オペレーションパネルのF1からF4キーに機能を割り付けます。
	OUTPUT GROUP COUNT	OG	OG　;	グループカウントおよびエスケープステータスを問い合わせます。
	OUTPUT KEY	OK	OK　;	キーボードモードになっているパネルのキーの押下状態を問い合わせます。
	WRITE TO DISPLAY	WD	WD　(ccc…c) term	文字列を表示し、キーボードモードへ移行します。
プロッタの各種情報問い合わせコマンド	INPUT MASK	IM	IM　E-mask (, S-mask (, P-mask)) ;	エラーマスク、シリアルポールマスク、パラレルポールマスクを設定します。
	OUTPUT ACTUAL POSITION	OA	OA　;	現在のペン位置とペンのアップダウン状態を問い合わせます。
	OUTPUT COMMANDED POSITION	OC	OC　;	MH-GLのコマンドせ与えられる現在のペン位置とペンのアップダウン状態を問い合わせます。
	OUTPUT ERROR	OE	OE　;	MH-GL命令によるエラーの有無と種類を問い合わせます。
	OUTPUT FACTORS	OF	OF　;	1mmの長さをプロッタ単位で出力するよう問い合わせます。
	OUTPUT IDENTIFICATION	OI	OI　;	プロッタ型式名を問い合わせます。
	OUTPUT OPTIONS	OO	OO　;	プロッタのオプションの状態を問い合わせます。
	OUTPUT STATUS	OS	OS　;	プロッタステータスを問い合わせます。
	OUTPUT TYPE STOCKER	OT	OT　;	ストッカの種類とストッカ内のペン有無状態を問い合わせます。

Compumotor

OEM650/OEM650X
OEM350/OEM350X
Drive and Drive/Indexer
User Guide



Compumotor Division
Parker Hannifin Corporation
p/n 88-013157-02 A

Parker

Motor Size	Current	Resistor	Voltage
OEM57-40-MOS	2.65A	21.0 kΩ	48 - 75VDC
OEM57-40-MOP	5.3A	5.76 kΩ	24 - 48VDC
OEM57-51-MOS	3.3A	15.8 kΩ	48 - 75VDC
OEM57-51-MOP	6.6A	2.05 kΩ	24 - 48VDC
OEM57-83-MOS	3.8A	12.7 kΩ	48 - 75VDC
OEM57-83-MOP	7.5A	0.00 kΩ	24 - 48VDC
OEM83-62-MO*	4.4A	9.53 kΩ	24 - 75VDC
OEM83-93-MO*	5.6A	4.87 kΩ	24 - 75VDC
OEM83-135-MO*	6.9A	1.27 kΩ	24 - 75VDC

S: Series Configuration P: Parallel Configuration
**34 size motors are internally wired in parallel*

Table 2-7. OEM Drive Motor Current (Compumotor Motors)

If you use a *non-Compumotor motor*, carefully follow the motor manufacturer's instructions regarding motor wiring and the proper operating current. Compumotor recommends a motor inductance of 2 mH measured in series or parallel (0.5 mH - 10 mH is acceptable). Table 2-8 shows resistor values that you must use to properly set motor current when using the **OEM650/ OEM650X** with a non-Compumotor motor. When the ***motor current range jumper*** (jumper 1—see Figure 2-1) is installed, the drive can generate 2.5 to 7.5 amps. When jumper 1 is removed, the drive can generate 0.83 to 2.5 amps. If you use the **OEM350/ OEM350X**, use Table 2-9 for resistor and current values to use with high-inductance (10 mH to 80 mH), low current motors.

Jumper #1 Installed				Jumper #1 Removed	
Current (Amps)	*Resistance (Ohms)*	*Current (Amps)*	*Resistance (Ohms)*	*Current (Amps)*	*Resistance (Ohms)*
7.5	0 Ω	4.9	7.32 kΩ	2.5	0 Ω
7.4	205 Ω	4.8	7.68 kΩ	2.4	619 Ω
7.3	412 Ω	4.7	8.06 kΩ	2.3	1.27 kΩ
7.2	619 Ω	4.6	8.45 kΩ	2.2	2.05 kΩ
7.1	825 Ω	4.5	8.87 kΩ	2.1	2.80 kΩ
7.0	1.02 kΩ	4.4	9.53 kΩ	2.0	3.57 kΩ
6.9	1.27 kΩ	4.3	10.0 kΩ	1.9	4.53 kΩ
6.8	1.54 kΩ	4.2	10.5 kΩ	1.8	5.49 kΩ
6.7	1.78 kΩ	4.1	10.0 kΩ	1.7	6.49 kΩ
6.6	2.05 kΩ	4.0	11.5 kΩ	1.6	7.68 kΩ
6.5	2.26 kΩ	3.9	12.1 kΩ	1.5	8.87 kΩ
6.4	2.55 kΩ	3.8	12.7 kΩ	1.4	10.5 kΩ
6.3	2.80 kΩ	3.7	13.3 kΩ	1.3	12.1 kΩ
6.2	3.09 kΩ	3.6	13.7 kΩ	1.2	13.7 kΩ
6.1	3.32 kΩ	3.5	14.3 kΩ	1.1	15.8 kΩ
6.0	3.57 kΩ	3.4	15.0 kΩ	1.0	18.2 kΩ
5.9	3.92 kΩ	3.3	15.8 kΩ	0.9	20.5 kΩ
5.8	4.22 kΩ	3.2	16.5 kΩ	0.83	22.6 kΩ
5.7	4.53 kΩ	3.1	17.4 kΩ		
5.6	4.87 kΩ	3.0	18.2 kΩ		
5.5	5.11 kΩ	2.9	19.1 kΩ		
5.4	5.49 kΩ	2.8	20.0 kΩ		
5.3	5.76 kΩ	2.7	20.5 kΩ		
5.2	6.19 kΩ	2.6	21.5 kΩ		
5.1	6.49 kΩ	2.5	22.6 kΩ		
5.0	6.81 kΩ				

*Table 2-8. **OEM650/650X** Resistor Selection for Motor Current*

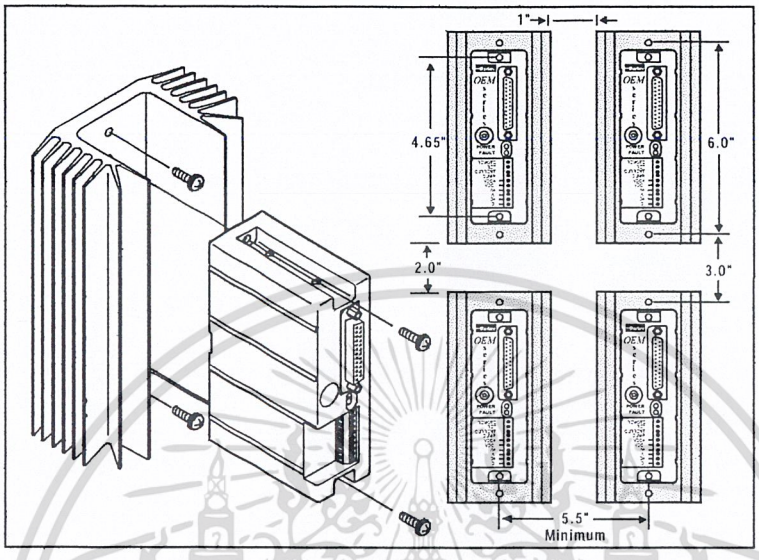


Figure 2-17. OEM650/OEM650X OEM-HS2 Minimum Area Panel Layout

Jumper Functions

Figure 2-1 shows the location and function of the 11 OEM650/OEM650X jumpers. When the unit is shipped to you, all 11 jumpers are installed. Each jumper's function is defined in this section.

Jumper #1: Motor Current Range

This jumper sets the range of user configurable motor current settings. *Refer to Tables 2-8 and 2-9 for motor current values with jumper 1 installed and removed.*

Jumpers #2 - #5: Motor Resolution

These jumpers control motor resolution (how many steps are in one revolution). *Although higher resolutions typically result in finer positioning and improved low-speed smoothness, it does not necessarily result in improved accuracy.*

Resolution	JU2	JU3	JU4	JU5
50,800 Steps/Rev	on	on	on	off
50,000 Steps/Rev	on	on	off	on
36,000 Steps/Rev	on	on	off	off
25,600 Steps/Rev	on	off	on	on
25,400 Steps/Rev	on	off	on	off
25,000 Steps/Rev *	on	on	on	on
21,600 Steps/Rev	on	off	off	on
20,000 Steps/Rev	on	off	off	off
18,000 Steps/Rev	off	on	on	on
12,800 Steps/Rev	off	on	on	off
10,000 Steps/Rev	off	on	off	on
5,000 Steps/Rev	off	on	off	off
2,000 Steps/Rev	off	off	on	on
1,000 Steps/Rev	off	off	on	off
400 Steps/Rev	off	off	off	on
200 Steps/Rev	off	off	off	off

* *Default Setting*

Table 2-10. Motor Resolution Jumper Settings

Your indexer (if you are using an OEM650) and drive should be set to the same resolution. If the drive and indexer's motor resolution settings do not match, commanded accelerations and velocities will not be performed accurately.

Jumpers #6 - #8: Motor Waveform Shape

These jumpers control the shape or waveform of the commanded motor current. Motor waveforms can reduce resonance problems and allow the motor to run smoothly. This function will not operate when the 200-step or 400-step motor resolutions are used.

Motor Waveform	JU6	JU7	JU8
Pure sine	off	off	on
-2% 3rd Harmonic	off	on	off
-4% 3rd Harmonic*	on	on	on
-4% 3rd Harmonic	off	off	off
-4% 3rd Harmonic	off	on	on
-6% 3rd Harmonic	on	off	off
-8% 3rd Harmonic	on	off	on
-10% 3rd Harmonic	on	on	off

* *Default Setting*

Table 2-11. Motor Waveform Jumper Settings

Jumpers #9 - #10: Auto Standby

The Automatic Standby function allows the motor to cool when it is not moving. This function reduces the current to the motor when the drive does not receive a step pulse for one second. Full current is restored upon the first step pulse that the drive receives. *Do not use this function in systems that use an indexer and an encoder for position maintenance. If used in this environment, the system will go in and out of the Auto Standby mode.*

Standby Current	*JU9*	*JU10*
Full Current*	on	on
75% Current	off	on
50% Current	on	off
25% Current	off	off

* *Default Setting*

Table 2-12. Auto Standby Jumper Settings

Jumper #11: Auto Test

The Automatic Test function turns the motor shaft slightly less than six revolutions in Alternating mode at 1 rps. The Automatic Standby function and motor resolution settings are disabled when you use the Automatic Test function.

Auto Test	*JU11*
Enabled	off
Disabled*	on

* *Default Setting*

Table 2-13. Auto Test Jumper Settings

OEM650 Inputs and Outputs

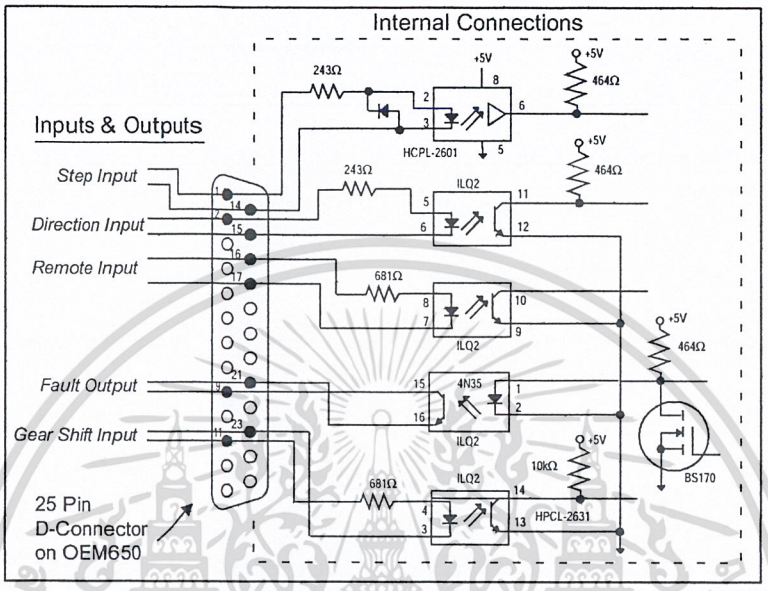


Figure 2-18. OEM650 Inputs & Output Schematic

Step Input Signal Specification

The OEM650's inputs are optically isolated and may be driven (activated) by providing a positive pulse to the plus input with respect to the minus input. This input may also be differentially driven. The input driver must provide a minimum of 6.5 mA—*approximately 3.5 VDC* (15 mA maximum).

Step Pulse Input

Operate the step pulse input within the following guidelines:

- ❑ 200 nanosecond-pulse minimum
- ❑ 40% - 60% duty cycle (2 MHz maximum pulse rate

Direction Input Signal Specification

The OEM650's inputs are optically isolated and may be driven (activated) by providing a positive pulse to the plus input with respect to the minus input. The input may also be differentially driven. The input driver must provide a minimum of 10 mA—*approximately 3.5 VDC*—to ensure adequate operation.

Direction Input

The direction may change polarity coincident with the last step pulse. The direction input must be stable for at least 120 µsec before the drive receives the first pulse.

Remote Input

The Remote input is an optically isolated input that uses an ILQ2 quad OPTO isolator. The REMOTE+ terminal is connected to the anode of the OPTO lead via a 681Ω current limiting resistor. The REMOTE- terminal is connected to the cathode of the OPTO lead. The OPTO requires a minimum of 3.5 mA (≈3.5VDC) to ensure proper system operation.

This input allows you to reduce current to a motor from a remote location. This is accomplished by changing the current select resistor via the remote input. When the remote input is enabled, the open collector transistor connected to the REMOTE screw terminal will conduct to ground. If the CURRENT and REMOTE terminals are shorted together (with a wire) motor current will be reduced to zero.

Motor current can also be reduced by a percentage if CURRENT and REMOTE are shorted with the appropriate resistor. A remote motor current value must be selected (see Table 2-8) to set the operating current. Identify the current resistor associated with the operating current you select. Use the resistor values to determine the remote resistor that must be installed between the CURRENT and REMOTE terminals. Use the following equation to detemine R_{REMOTE}.

$$R_{REMOTE} = -13{,}300\ (3650 + R_C)\ /\ (R_C - R_S)$$

R_C = Resistor associated with the operating current
R_S = Resistor associated with the desired standby current

Fault Output

This output is an open-collector, open emitter output from a ILQ2 OPTO isolator. The output transistor will conduct when the drive is functioning properly. The transistor will not conduct properly when any of the following conditions exist.

- ❑ No power is applied to the drive
- ❑ There is insufficient voltage (<24VDC)
- ❑ The driver detects a motor fault
- ❑ The remote input is enabled

This output has the following electrical characteristics:

- ❑ V_{CE} = 35VDC
- ❑ V_{CESAT} = 0.3VDC
- ❑ Collector Current = 10 mA maximum
- ❑ Dissipation = 100 mW maximum