เครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์
Engraving Machine controlled by Microcontroller

โดย

นายเตชนิธิ จันตะเคียน

นางสาวปาณิศา กุลไทย

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

เครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์
Engraving Machine controlled by Microcontroller

โดย

นายเตชนิธิ จันตะเคียน 46010246

นางสาวปาณิศา กุลไทย 46010447

อาจารย์ที่ปรึกษา

รศ.ดร. สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาอิเล็คทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

เครื่องแกะสลักด้วยไมโครคอนโทรลเลอร์
Engraving Machine Controlled by Microcontroller

นายเตชนิธิ จันตะเคียน รหัส 46010246
นางสาวปาณิศา กุลไทย รหัส 46010447

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ.ดร. สุรพันธ์ เอื้อไพบูลย์)
อาจารย์ที่ปรึกษา

ปริญญานิพนธ์ ปีการศึกษา 2549

ภาควิชาอิเล็กทรอนิกส์

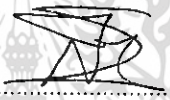
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องแกะสลักด้วยไมโครคอนโทรลเลอร์
Engraving Machine Controlled by Microcontroller

ผู้จัดทำ

1.นายเตชนิธิ จันตะเคียน 46010246

2.นางสาวปาณิศา กุลไทย 46010447



.....อาจารย์ที่ปรึกษา

(รศ.ดร. สุรพันธ์ เอื้อไพบูลย์)

เครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์

นายเตชนิธิ จันตะเคียน
นางสาวปาณิศา กุลไทย
รศ.ดร. สุรพันธ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2549

บทคัดย่อ

เครื่องกัดควบคุมด้วยไมโครคอนโทรลเลอร์ ที่ได้ประดิษฐ์ขึ้นนี้ มีระบบการทำงานโดยรับรับภาพต้นแบบจากกล้องเวบแคมผ่านคอมพิวเตอร์แล้วประมวลผลภาพส่งไปยังไมโครคอนโทรลเลอร์เพื่อสั่งให้เครื่องแกะสลักมีการเคลื่อนที่ในระนาบ 3 มิติ ด้วยสเตปปิ้งมอเตอร์

Engraving Machine controlled by Microcontroller

Mr. Techaniti Chantakian

Ms. Panisa Kulthai

Assist. Prof. Dr. Surapan Airphaiboon Advisor

Education Year 2006

Abstract

Microcontroller based engraving machine is operated by user , imported model picture , via computer serial interface. Then processes by microcontroller for using stepping motor to control 3-dimension. In addition, image processing is using for developing this machine.

กิตติกรรมประกาศ

รายงานและชิ้นงานเครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์ที่จัดทำขึ้นมานี้สำเร็จลุล่วงไปได้ด้วยดีเพราะได้รับการสนับสนุน ความช่วยเหลือและคำปรึกษาที่ดีจาก รศ.ดร.สุรพันธ์ เอื้อไพบูลย์ ที่คอมให้คำแนะนำรวมทั้งเอื้อเฟื้อสถานที่และอุปกรณ์ในการทำงานมาโดยตลอดปีการศึกษา รวมทั้งขอบคุณแรงใจจากคุณพ่อ คุณแม่ที่คอยให้กำลังใจในการทำงานตลอดมา

ผู้จัดทำ

สารบัญ

สารบัญรูป

หน้า

สารบัญตาราง

บทที่ 1 เครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์

ในปัจจุบันนี้มีการนำเทคนิคการประมวลผลภาพและเทคนิคการสั่งการควบคุมการเคลื่อนที่มาใช้ทั้งในภาคการวิจัยและการใช้งานทางภาคอุตสาหกรรม ทางผู้จัดทำโครงงานนี้จึงมีความคิดที่จะศึกษาและและผลิตชิ้นงานโดยใช้ความรู้ที่กล่าวมาข้างต้น จึงได้จัดทำโครงงานเครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์ ซึ่งเป็นโครงงานที่จัดทำขึ้นมาเพื่อนำความรู้ด้านอิเล็กทรอนิกส์และการประมวลผลภาพมาประยุกต์ให้เกิดการประดิษฐ์พัฒนาอุปกรณ์ขึ้นโดยสามารถนำมาใช้งานได้จริง รวมถึงเรียนรู้ถึงการแก้ปัญหาต่างๆที่เกิดขึ้นช่วงระหว่างทำโครงงาน

ลักษณะการทำงานโดยรวมของเครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์นั้น ในขั้นแรกจะรับข้อมูลจากผู้ใช้งานผ่านโปรแกรมคอมพิวเตอร์ซึ่งง่ายต่อการใช้งาน จากนั้นเครื่องคอมพิวเตอร์จะทำการประมวลผลภาพด้วยโปรแกรมvisual basic 6.0 แล้วทำการส่งข้อมูลหรือคำสั่งผ่านพอร์ต RS-232 ไปยังไมโครคอนโทรลเลอร์เพื่อประมวลผลการเคลื่อนที่อีกครั้งหนึ่ง หลังจากนั้นไมโครคอนโทรลเลอร์จะสั่งงานและควบคุมการทำงานเครื่องแกะสลักเคลื่อนที่ตามข้อมูลที่ได้รับจากผู้ใช้งานในขั้นแรกนั้นเอง ในการทำงานของเครื่องแกะสลักนี้จะใช้สเต็ปปิ้งมอเตอร์เข้ามาควบคุมการเคลื่อนที่ของหัวเจาะในระนาบ 3 มิติ ซึ่งจะเป็นการเคลื่อนที่ไปตามจุดบนระนาบ 2 มิติ แล้วเราสามรถกำหนดความลึกในการเจาะได้ตามต้องการ

รูปที่ 1.1 บล็อกไดอะแกรมของโครงงานชิ้นนี้

บทที่ 2 หลักการทฤษฎี

2.1ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์LPC2138 เป็นตัวประมวลผลแบบ 16/32 บิต ARM7TDMI-S™ ที่มีระบบจำลองการทำงานและติดตามผลการทำงานของคำสั่ง ไมโครคอนโทรลเลอร์LPC2138 ประกอบด้วยหน่วยความจำความเร็วสูงแบบแฟลชขนิด 512 กิโลบิต การติดต่อหน่วยความจำขนาด 128 บิต โมดูลเร่งความเร็วในการแปลคำสั่งขนาด 32 บิต บนความเร็วสัญญาณนาฬิกาสูงสุด พร้อมทั้งความสามารถลดขนาดคำสั่งลง 30% ด้วยการใช้คำสั่งแบบ 16 บิต

ด้วยขนาดที่เล็กและมีการกินกำลังไฟในการใช้งานน้อย ทำให้ไมโครคอนโทรลเลอร์ LPC2138 เหมาะสำหรับที่จะนำไปใช้งานที่ต้องการขนาดเล็ก เช่น การควบคุมแบบเข้าถึง (access control) เนื่องด้วยการติดต่อสื่อสารแบบอนุกรมขนาดใหญ่ และมีหน่วยความจำแรมแบบสเตติก (SRAM) ภายในชิพที่สามารถเลือกค่าได้ 8/16/32 กิโลบิต ทำให้เหมาะสมอย่างยิ่งสำหรับงานทางด้านสื่อสาร อีกทั้งใน LPC2138 มีบัฟเฟอร์ขนาดใหญ่และมีสมรรถนะการทำงานที่สูง ไทเมอร์ที่มีความหลายหลายถึง 32 บิต , วงจรแปลงจากอนาลอกเป็นดิจิตอล 8 ช่องทาง ขนาด 10 บิต ซึ่งสามารถเลือกเป็นแบบเดี่ยวหรือแบบคู่ , วงจรแปลงจากดิจิตอลเป็นอนาลอก , วงจรสร้างพัลส์วิดท์มอดดูเลชั่น (Pulse Width Modulation : PWM) และ 47 ขาสัญญาณอินพุท/เอาท์พุท (GPIO) ที่ขึ้นอยู่กับระดับความไวต่อขาสัญญาณอินเตอร์รัปจากภายนอก โดยทั้งหมดนี้ทำ LPC2138 เหมาะสำหรับงานด้านอุตสาหกรรม และทางด้านระบบการแพทย์

2.1.1ลักษณะทั่วไป

-ไมโครคอนโทรลเลอร์ขนาด 16/32 บิต แบบ ARM7TDMI-S™ ที่บรรจุอยู่ในตัวถังแบบ LQFP64

-หน่วยความจำแบบสเตติกแรม (Static RAM) ขนาด 32 กิโลไบท์ และหน่วยความจำโปรแกรมแบบแฟลช (Flash Program Memory) ขนาดมี 512 กิโลไบท์ การอินเตอร์เฟสหน่วยความจำแบบ 128 บิต

-โปรแกรมชิพได้ทันทีผ่าน In-System/In-Application Programming : ISP/IAP โดยใช้ซอฟต์แวร์ boot-loader ที่มีอยู่ภายในชิพ

- วงจรแปลงสัญญาณจากอนาลอกเป็นดิจิตอลขนาด 10 บิต จำนวน 8 ช่องสัญญาณ 2 ชุด โดยมีเวลาการแปลงต่ำถึง 2.44 ไมโครวินาทีต่อ 1 ช่องสัญญาณ

- วงจรแปลงสัญญาณดิจิตอลเป็นอนาลอกขนาด 10 บิต 1 ตัว

- ไทเมอร์/เคาท์เตอร์ ขนาด 32 บิต จำนวน 2 ชุด(มีช่องสัญญาณแคปเจอและคอมแพอย่างละ 4 ช่องสัญญาณ) , วงจรสร้างพัลวิดท์มอดดูเลชั่น (PWM unit) จำนวน 6 เอาท์พุต และมีวอทด์ดอกไทเมอร์ (watchdog timer)

-วงจรสัญญาณนาฬิกาตามเวลาจริง(Real-Time clock)ที่กินกำลังต่ำ และ สามารถใช้คริสตัลภายนอกขนาด 32 กิโลเฮิร์ต

-โมดูลควบคุมอินเตอร์รัปเวกเตอร์ด้วยรูปร่างลำดับก่อนหลังและที่อยู่เวกเตอร์

-มีขาสัญญาณอินพุทที่รองรับสัญญาณได้ถึง 5 โวทต์ 47 ขา

-สามารถขยายได้เป็น 9 ระดับต่อความไวของหน่วยอินเตอร์รัปภายนอก

-หน่วยประมวลผลกลางความถี่สูงสุด 60 เมกกะเฮิร์ต โดยการเขียนคำสั่งบนชิปที่ใช้วงจรเฟสลอกลูป (PLL) ภายในตัว

-มีวงจรออสซิลเลเตอร์ภายในชิปปฏิบัติการที่สามารถต่อคริสตอลภายนอกด้วยความถี่ 1 ถึง 30 เมกกะเฮิร์ต หรือ ต่อวงจรออสซิลเลเตอร์จากภายนอกด้วยความถี่ 1 ถึง 50 เมกกะเฮิร์ต

-โหมดประหยัดพลังงาน 2 โมดได้แก่ Idle และ Power-down

-สามารถกำหนดเอนเนเบิลหรือดิสเอเบิลในแต่ละฟังก์ชั่น

-สามารถปลุกการทำงานได้ผ่านทางการอินเตอร์รัปภายนอกหรือใช้นาฬิกาตามเวลาจริง (real-time clock)

-จ่ายกำลังไฟฟ้าด้วยโหมดพาวเวอร์ออนรีเซต (Power-On Reset : POR) และการป้องการไหม้ของวงจร (Brown-Out Detection)

-หน่วยประมวลผลกลางปฏิบัติการด้วยแรงดัน 3 ถึง 3.6 โวลต์ (3.3 V ± 10%) และระยะห่างระหว่างขาสัญญาณแบบแรงดันขนาด 5 โวลต์สำหรับขาสัญญาณอินพุท/เอาท์พุท

2.1.2 การประยุกต์ใช้งาน

-งานควบคุมอุตสาหกรรม (Idustrial Control)

-งานทางการแพทย์ (Medical Systems)

-งานควบคุมโดยตรง (Access control)

-งานด้านการสื่อสาร (Commynication gateway)

-โมเด็มแบบฝังตัวในระบบ

-งานด้านการประยุกต์ทั่วไป

2.1.3 สถาปัตยกรรม LPC2138

ไมโครคอนโทรลเลอร์ LPC2138 ประกอบด้วยหน่วยประมวลผลกลางและการสนับสนุนระบบต่าง ๆ แบบ ARM7TDMI-S ซึ่งระบบต่าง ๆ นั้นประกอบไปด้วยโลคอลบัสสำหรับอินเตอร์เฟสกับหน่วยความจำบนชิป , AMBA Advanced High-performance Bus : AHB สำหรับอินเตอร์เฟสกับอินเตอร์รัปของตัวควบคุม , VLSI Peripheral Bus : VPB ใช้สำหรับการเข้ากันได้ของ ARM's AMBA Advanced Peripheral Bus สำหรับการติดต่อกับฟังก์ชั่นภายนอก

อุปกรณ์พ่วงต่อที่ใช้ AHB ในการสื่อสารถูกแบ่งให้อยู่ในพื้นที่แอดเดรส 2 เมกกะไบต์ที่อยู่ส่วนบนสุดของหน่วยความจำ 4 กิกกะไบต์ของ ARM ในแต่ละอุปกรณ์ที่เชื่อมต่อด้วย AHB จะอยู่

ในพื้นที่ของแอดเดรสขนาด 16 กิโลบิต ส่วนฟังก์ชั่นภายนอกของ LPC2138 จะถูกเชื่อมต่อด้วยบัสแบบ VPB Bus การสื่อสารระหว่างAHB กับ VPB จะผ่านตัวเชื่อมต่อที่เรียกว่า AHB-to-VPB Bridge อุปกรณ์ที่เชื่อมต่อด้วย VPB จะถูกแบ่งหน่วยความจำขนาด 2 เมกกะไบต์เริ่มต้นที่หน่วยความจำที่ 3.5 กิกกะไบต์ อุปกรณ์แต่ละตัวที่ติดต่อด้วย VHB จะเก็บอยู่ในหน่วยความจำขนาด 16 กิโลไบต์

การกำหนดหน้าที่ของขาที่ทำการติดต่อกับอุปกรณ์ภายนอกจะกำหนดผ่านกล่องติดต่อขาสัญญาณ(Pin Connect Block) ที่กำหนดผ่านซอร์ฟแวร์เท่านั้น

2.1.4 ตัวประมวลผลของ ARM7TDMI-S

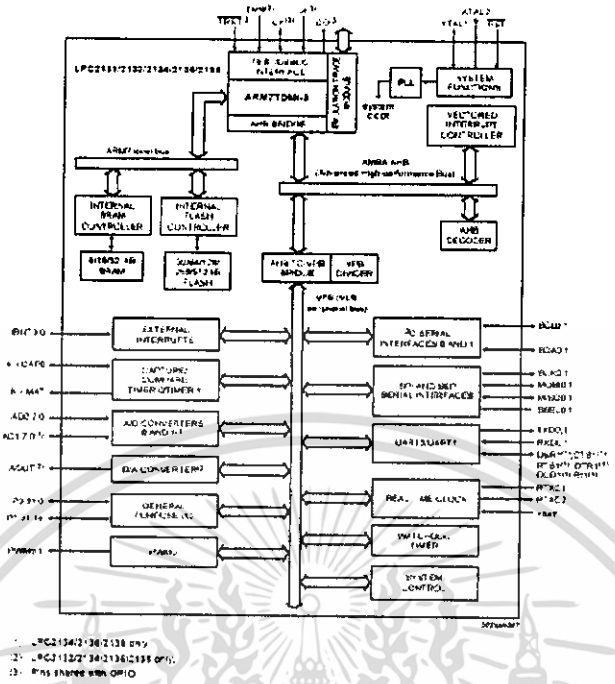
ARM7TDMI-S เป็นไมโครโพรเซสเซอร์ขนาด 32 บิต ที่สมรรถนะในการทำงานสูงและกินกำลังงานต่ำ สถาปัตยกรรมของ ARM อยู่บนพื้นฐานของชุดคำสั่งแบบลดรูป (RISC) ระบบชุดคำสั่งและตัวแปลคำสั่งมีความยุ่งยากน้อยกว่าชุดคำสั่งแบบComplex ผลของคำสั่งที่ง่ายขึ้นนี้ทำให้เกิดปริมาณงานที่มากขึ้นและผลตอบสนองของ real-time interrupt จากตัวประมวลผลมีผลที่น่าพึงพอใจมากขึ้น

เทคนิคการการทำงานแบบ Pipeline ถูกนำมาใช้งานทำให้ทุกส่วนของตัวประมวลผลและระบบหน่วยความจำสามารถทำงานได้อย่างต่อเนื่อง อย่างเช่น ขณะที่คำสั่งถูกเอกซีคิว (execute) ตัวประมวลผลจะทำการแปลคำสั่งและคำสั่งต่อไปจะถูกอ่านจากหน่วยความจำ

ตัวประมวลผล ARM7TDMI-S จะมีลักษณะการทำงานแปลกกว่าตัวประมวลผลอย่างหนึ่งที่เรียกว่า THUMB จะเป็นเพิ่มจำนวนคำสั่งอย่างมีประสิทธิภาพ หัวใจการทำงานของ THUMB คือการลดคำลดรูปของคำสั่งอย่างมาก โดยสรุป ARM7TDMI-S มีชุดคำสั่งอยู่ 2 ประเภทคือ

-ชุดคำสั่งแบบทั่วไป 32 บิต

-ชุดคำสั่งแบบ THUMB 16 บิต



รูปที่ 2.1 โครงสร้างของ LPC2138

2.2 สเต็ปปิ้งมอเตอร์ (Stepping Motor)

มอเตอร์เป็นเครื่องมือทางอิเล็กทรอนิกส์ที่เปลี่ยนพลังงานทางไฟฟ้ามาทำให้เกิดสนามแม่เหล็กขึ้นภายในมอเตอร์แล้วไปขับขดลวดตัวนำอาร์เมเจอร์หมุน ทำให้เพลาของมอเตอร์หมุน ซึ่งจะได้พลังงานกลออกไปใช้งาน

สเต็ปปิ้งมอเตอร์เป็นมอเตอร์นิยมใช้ในงานที่ต้องการความแม่นยำและทิศทางที่แน่นอน เนื่องจากสเต็ปปิ้งมอเตอร์จะเคลื่อนที่ทีละขั้น ขั้นละ 0.9 , 1.8 , 5 , 7.5 , 15 หรือ 50 องศา ซึ่งขึ้นอยู่กับคุณสมบัติแต่ละชนิดของสเต็ปปิ้งมอเตอร์ตัวนั้น ๆ ซึ่งมันจะแตกต่างจากมอเตอร์ไฟฟ้ากระแสตรงทั่วไป (DC Motor) ที่จะหมุนแบบต่อเนื่อง ไม่สามารถหมุนเป็นแบบสเต็ปๆ ได้ ดังนั้นในการนำไปกำหนดตำแหน่งจึงควบคุมได้ยากกว่า ส่วนใหญ่จะใช้สเต็ปปิ้งมอเตอร์มาทำการการควบคุมโดยใช้วิธีในระบบดิจิตอล เช่น ระบบขับเคลื่อนหัวแม่พิมพ์ในพรินเตอร์ (Printer) ระบบขับเคลื่อนตำแหน่งของปากกาในพล็อตเตอร์ (X-Y Plotter) ระบบขับเคลื่อนหัวอ่านในเครื่องอ่านบันทึกดิสก์ไดร์ฟ (Disk drive)

รูปที่ 2.2 รูปสเต็ปปิ้งมอเตอร์

ข้อดีของสเต็ปปิ้งมอเตอร์เมื่อเปรียบเทียบกับมอเตอร์กระแสตรง

1.การควบคุมไม่ต้องอาศัยตัวตรวจจับการหมุน

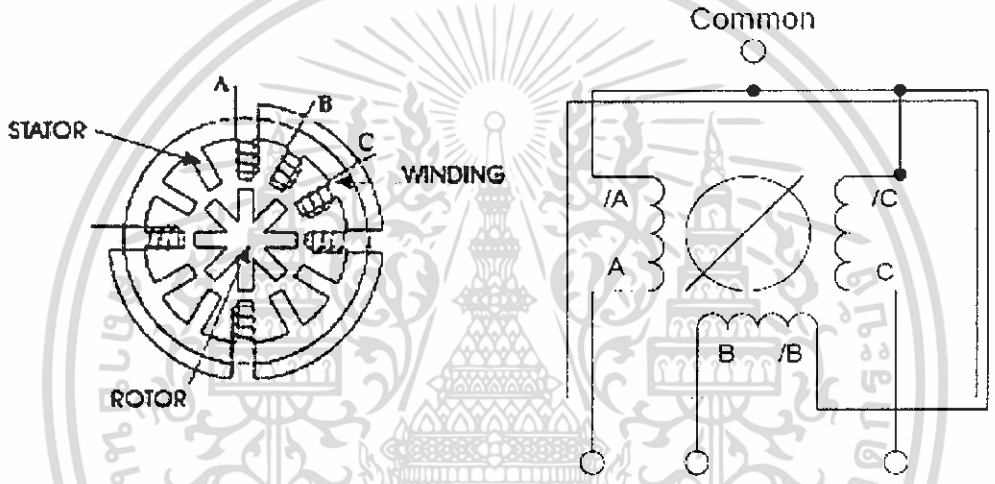
2.ไม่ต้องใช้แปรงถ่าน ดังนั้นจึงทำให้ไม่มีส่วนที่จะต้องสึกหรอ และปัญหาของการสปาร์คที่ทำให้เกิดสัญญาณรบกวน

3. การควบคุมโดยทางวงจรดิจิตอลหรือไมโครโพรเซสเซอร์ ทำได้ง่าย และสะดวก

2.2.1 ชนิดของสเต็ปมอเตอร์ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

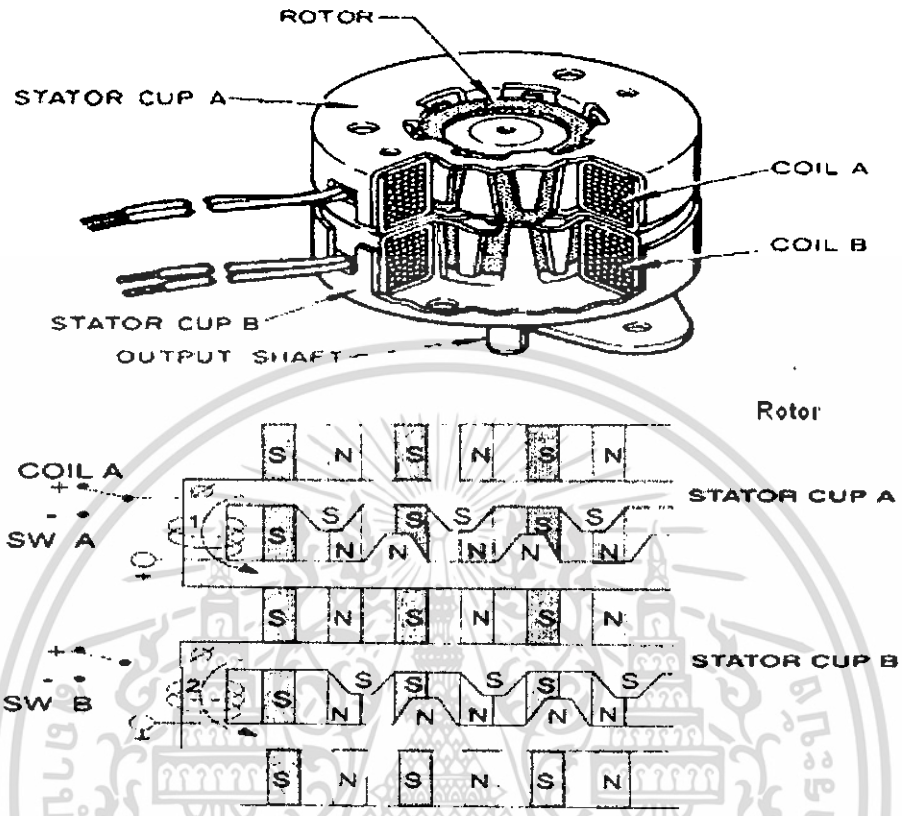
2.2.1.1 วาริเอเบิลรีลักแตนซ์ (Variable Reluctance : VR) โรเตอร์(Rotor) ทำด้วยเหล็กอ่อนรูปทรงกระบอกและทำเป็นลักษณะฟัน (teeth) สเตเตอร์(Stator)จะมีลวดพันและจะทำเป็นลักษณะของฟันเช่นกัน เมื่อจ่ายกระแส ไฟฟ้าให้กับขดลวดที่สเตเตอร์จะเกิดเป็นขั้วแม่เหล็กที่ฟันของสเตเตอร์และเหนี่ยวนำให้ฟันของโรเตอร์เกิด เป็นขั้วแม่เหล็กที่มีขั้วตรงกันข้ามกับสเตเตอร์

ทำ าให้ดึงดูดกันเกิดการหมุนของโรเตอร์ขึ้น มอเตอร์ชนิดนี้โดย ปกติจะมีขนาด 3 เฟส ในบางครั้งอาจพบถึง 4 เฟส มอเตอร์ชนิดนี้ถ้าไม่จ่ายกระแสไฟฟ้าให้กับขดลวดบนสเตเตอร์ ตัวโรเตอร์จะไม่เกิดแรงดึงดูดกับสเตเตอร์ มอเตอร์ชนิดนี้ไม่นิยมนำไปใช้ในงานอุตสาหกรรมแต่ จะถูกนำ าไปใช้กับงานที่มีขนาดเล็ก เช่น Micro-positioning table เป็นต้น เพราะ ไม่มีส่วนที่เป็นแม่เหล็กถาวร ดังนั้นในขณะไม่จ่ายกระแสไฟฟ้าให้กับขดลวดที่สเตเตอร์จึงไม่เกิดแรงดึงดูด วิธีการขับ(Driving)หรือการกระตุ้นเฟส(Phase Excitation) จะทำดังนี้คือ ต่อปลายด้านcommon เข้ากับแหล่งจ่ายไฟขั้วบวก (+) แล้วทำ การสวิทช์ให้ปลายด้าน A , B ,C ต่อลงกราวด์(Ground)ตามลำดับ ทีละปลายแล้วทำเช่นนี้เรื่อยไป แต่ถ้าต้องการให้หมุนกลับก็สวิทช์ย้อนกลับ และในการอธิบายต่อจากนี้ไปจะไม่ขอกล่าวถึงมอเตอร์ชนิดนี้อีก



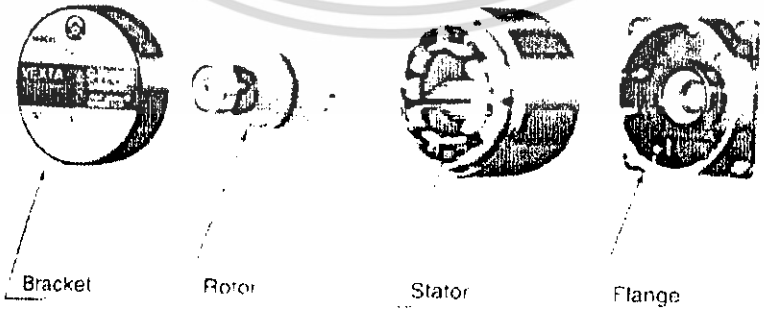
รูปที่ 2.3 ภาพแสดงโครงสร้างของสเต็ปมอเตอร์แบบวาริเวเบิลรีลักแตนซ์

2.2.1.2 แบบแม่เหล็กถาวร (Permanent Magnet : PM)โรเตอร์(Rotor) ทำด้วยแม่เหล็กถาวรรูปทรงกระบอกเรียบ สเตเตอร์(Stator)จะมีขดลวดพัน และก็จะทำเป็นฟัน เมื่อจ่ายกระแสไฟฟ้าให้กับขดลวดที่สเตเตอร์จะเกิดเป็นขั้วแม่เหล็กที่ฟันของสเตเตอร์และจะดึงดูดกับขั้วของแม่เหลกถาวรที่โรเตอร์ทำให้เกิดการหมุนของโรเตอร์ขึ้น มอเตอร์ชนิดนี้โดยจะมีตั้งแต่ขนาด 2 เฟสขึ้นไปมอเตอร์ชนิดนี้ไม่นิยมนำไปใช้ในงานอุตสาหกรรมแต่จะถูก นำไปใช้กับอุปกรณ์คอมพิวเตอร์เช่นตัวขับวงล้อที่ใช้หมุนเพื่อเลื่อนกระดาษของเครื่องพิมพ์ เป็นต้นเพราะความเร็วต่ำแรงบิดต่ำ และนอกจากนี้ด้วยโครงสร้างของมอเตอร์ชนิดนี้ทำให้มุมที่หมุนไปแต่ละสเต็ป(Step Angle)ไม่ละเอียด เช่น สเต็ปละ 3.6 ,7.5 , 15 ,18 องศาเป็นต้น มอเตอร์ชนิดนี้ถึงไม่จ่ายกระแสไฟฟ้าให้กับขดลวดบนสเตเตอร์ (Stator) ตัวโรเตอร์จะเกิดแรงดึงดูดกับสเตเตอร์ซึ่งเกิดจากอำนาจของแม่เหล็กถาวรที่โรเตอร์ทำ าให้หมุนได้ยาก จำนวนขั้วแม่เหล็กที่โรเตอร์สามารถ นับได้จากจำนวนขั้วแม่เหล็กที่จะเกิดขึ้นจากกระแสไฟฟ้าไหลผ่านขดลวดที่สเตเตอร์ชุดใดชุดหนึ่ง



รูปที่ 2.4 ภาพแสดงโครงสร้างของสเต็ปมอเตอร์แบบแม่เหล็กถาวร

2.2.1.3 แบบผสม (Hybrid : HB) ใช้หลักการทำงานของทั้งสองแบบมาออกแบบโดยที่สเตเตอร์จะคล้ายกับแบบ VR ส่วนโรเตอร์จะคล้ายแบบPM แต่จะทำเป็นฟัน มอเตอร์แบบนี้นิยมใช้ในงานอุตสาห-กรรมเพราะแรงบิดสูง ความละเอียดของสเต็ปในการหมุนสูง , ความเร็วสูงกว่าสองแบบที่กล่าวมาแล้ว มอเตอร์ชนิดนี้โดยปกติจะมีขนาด 2 เฟส ถึง 5 เฟส และมอเตอร์ชนิดนี้ได้มีการพัฒนาให้มีประสิทธิภาพ เหนือกว่าเดิมไปอีกโดยให้ชื่อว่า "Enhanced Hybrid" ซึ่งจะไม่อธิบาย

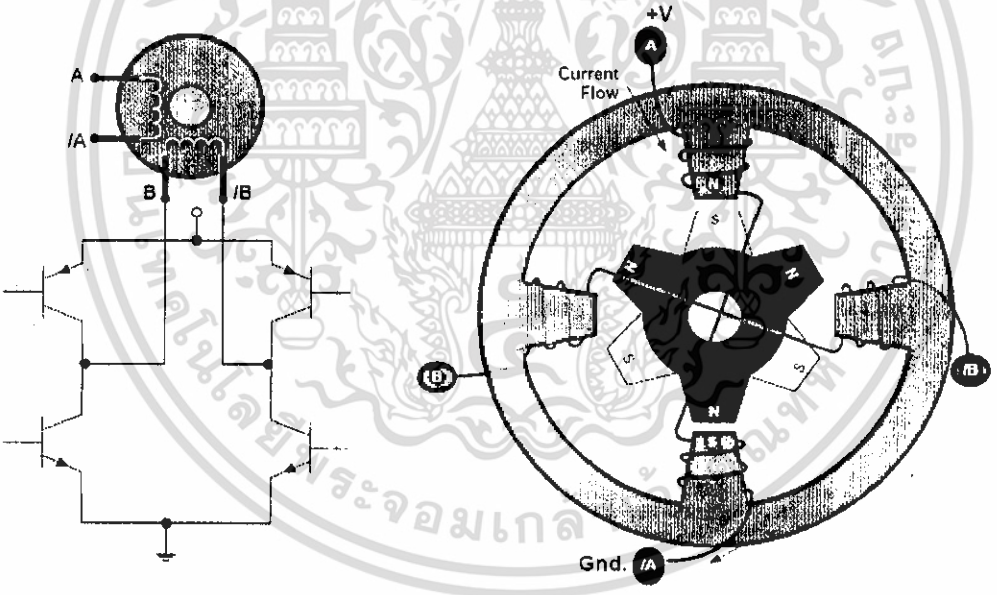


รูปที่2.5 สเต็ปเปอร์มอเตอร์ชนิดไฮบริดขนาด 5 เฟส

เฟสของสเต็ปเปอร์มอเตอร์ (**Stepper Motor Phase**) หมายถึง จำนวนขดลวดที่พันอยู่บนสเตเตอร์ซึ่งแยกออกจากกันอย่างอิสระ รูปที่2.3แสดงตัวอย่างมอเตอร์ขนาด 3 เฟส ในกรณีของมอเตอร์แบบยูนิโพลา 2 เฟสนั้นมักถูกจะเรียกเป็นมอเตอร์ขนาด 4 เฟสก็เพราะขดลวดที่พันอยู่บนสเตเตอรแต่ละชุดจะมี 2 ขด จึงเข้าใจว่ามี 4 ขดลวด แต่ถ้าพิจารณากันจริงจะพบว่าขดลวดทั้งสองนั้นเป็นขดลวดขดเดียวแต่มีจุดต่อตรงกลางขดเท่านั้น

2.2.2 ชนิดของสเต็ปเปอร์มอเตอร์แบ่งตามลักษณะสายที่ใช้ต่อกับวงจรขับ แบ่งออกได้ 2 แบบคือ

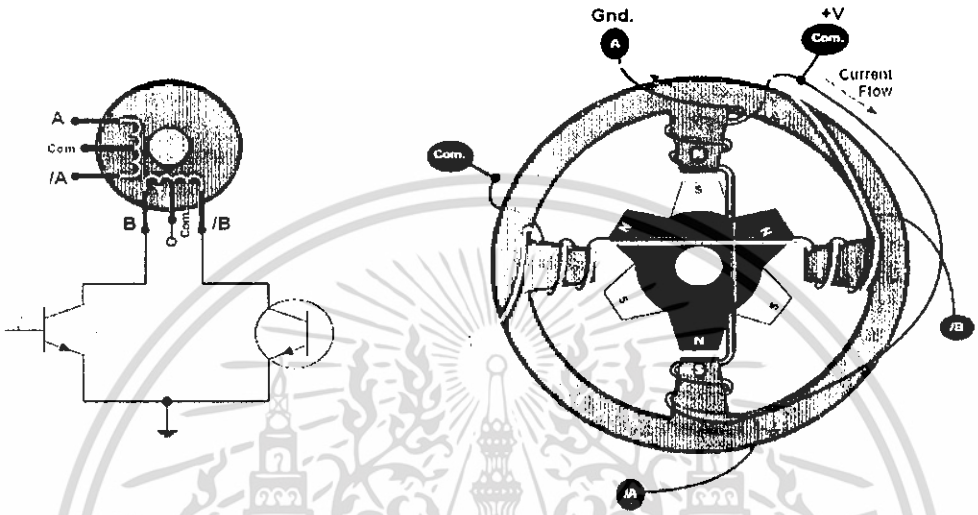
2.2.2.1 แบบไบโพลา (Bipolar) ขดลวดที่สเตเตอร์แต่ละชุดจะไม่มีจุดรวม การต่อเข้ากับวงจรขับจะใช้ปลายทั้งสอง ด้านของขดลวดแต่ละชุด การทำให้เกิดขั้วแม่เหล็กที่สเตเตอร์ทำได้โดยการจ่ายกระแสไฟจากปลายด้านหนึ่งไปยังปลายอีกด้านหนึ่งของขดลวดและการเปลี่ยนขั้วแม่เหล็กที่สเตเตอร์ชุดเดียวกันนี้ก็ทำได้โดยสลับทิศทางการไหลของกระแสไฟฟ้านั่นเอง ดังนั้นวงจรขับที่ใช้จึงจำเป็นต้องสามารถกลับทิศทางการไหลของกระแสได้ กรณีเป็นมอเตอร์ 2 เฟสจะมีสายที่ใช้ต่อขับกับวงจร 4 สาย



รูปที่ 2.6 สัญลักษณ์,โครงสร้างและวงจรขับที่ใช้กับมอเตอร์แบบไบ โพลา 2 เฟส

2.2.2.2 แบบยูนิโพลา (Unipolar) ขดลวดที่สเตเตอร์แต่ละชุดจะมีจุดรวม การพันขดลวดจะพันในแบบ Bifilar การต่อเข้ากับวงจรขับจะใช้ปลายของขดลวดแต่ละด้านต่อเข้ากับวงจรขับและใช้จุดรวมต่อเข้ากับขั้วบวกของแหล่งจ่ายไฟ การทำ าให้เกิดขั้วแม่เหล็กที่สเตเตอร์ทำได้โดยการจ่ายกระแสไฟให้ไหลจากจุดรวมลงกราวด์มาครบวงจรที่ปลายด้านหนึ่งของ

ขดลวด การเปลี่ยนขั้วแม่เหล็กที่สเตเตอร์ชุดเดียวกันนี้ก็ทำได้โดยเปลี่ยนการจ่ายกระแส ไฟฟ้าจากขดหนึ่งไปยังอีกขดหนึ่งของขดลวดที่พันอยู่บนสเตเตอร์ชุดเดียวกัน ดังนั้นวงจรขับจึงเป็นวงจรสวิทช์ เพื่อทำให้จ่ายกระแสไฟที่ผ่านขดลวดครบวงจรเท่านั้น กรณีเป็นมอเตอร์ 2 เฟสจะมีสายที่ใช้ต่อขับกับวงจร 5 หรือ 6 สาย

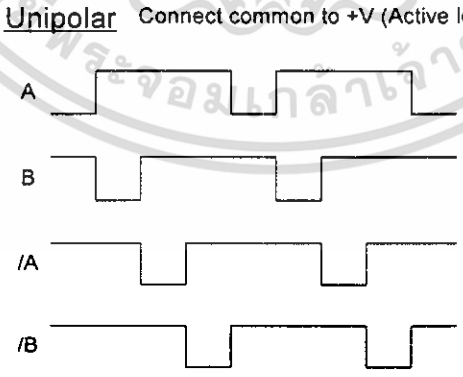


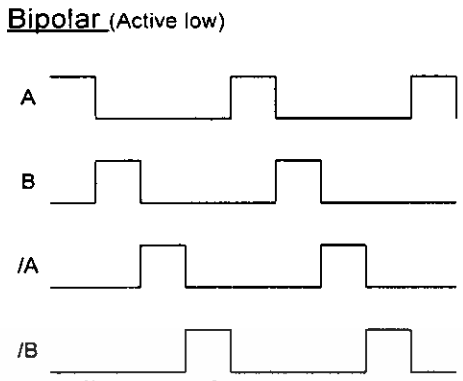
รูปที่ 2.7 สัญลักษณ์,โครงสร้างและวงจรขับที่ใช้กับมอเตอร์แบบยูนิโพลา 2 เฟส

2.2.3 วิธีการขับ(Driving)หรือวิธีการกระตุ้นเฟส (Phase Excitation)

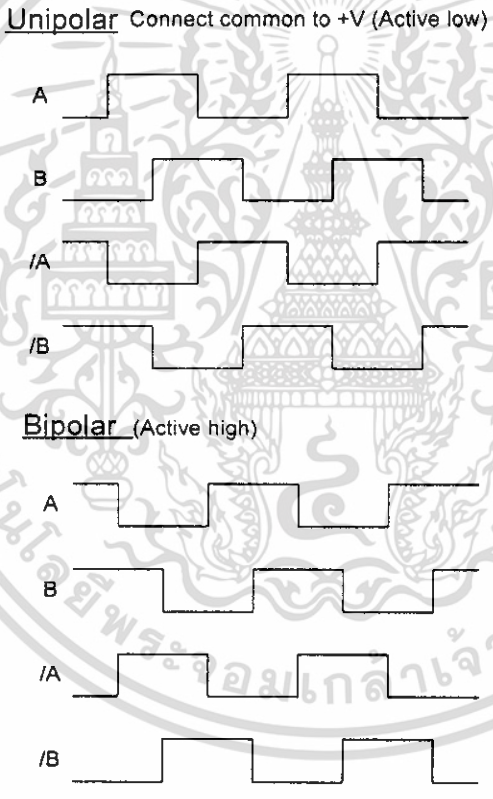
การกระตุ้นเฟสของสเต็ปเปอร์มอเตอร์คือ การจ่ายกระแสไฟฟ้าไปยังขดลวดที่สเตเตอร์ของแต่ละเฟสเพื่อทำให้มอเตอร์หมุนนั่นเอง แบ่งออกเป็น 3 วิธีคือ

2.2.3.1. การกระตุ้นเฟสแบบเฟสเดียว(One phase excitation) หรือ Half Drive การกระตุ้นเฟสแบบนี้ทำได้โดยการจ่ายกระแสไฟฟ้าไปยัง ขดลวดบนสเตเตอร์ครั้งละหนึ่งขด เรียงลำดับกันไปดังนี้

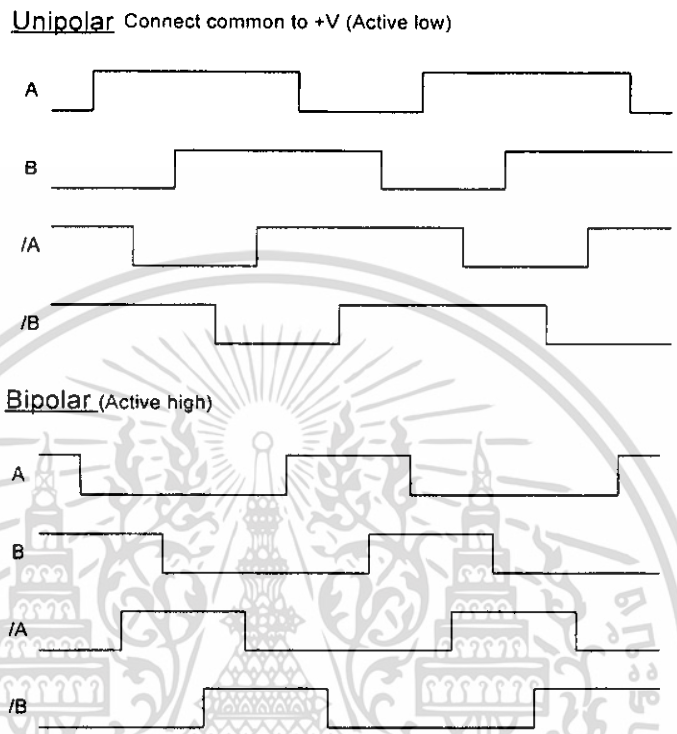




2.2.3.2. การกระตุ้นเฟสแบบคู่ Two phase excitation หรือ Full Stepการกระตุ้นเฟสแบบนี้ทำได้โดยการจ่ายกระแสไฟฟ้าไปยังขดลวดครั้งละสองขดที่อยู่ใกล้กันบนสเตเตอร์ ดังนี้

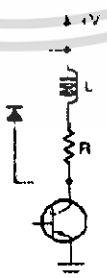


2.2.3.3. การกระตุ้นเฟสแบบคู่-เดี่ยว One - Two phase excitation หรือ Half Step การกระตุ้นเฟสแบบนี้ทำได้โดยการจ่ายกระแสไฟฟ้าไปยังขดลวดครั้งละสองขดที่อยู่ใกล้กัน บนสเตเตอร์สลับกับการจ่ายกระแสไฟฟ้าครั้งละหนึ่งขดบนสเตเตอร์ ดังนี้



2.2.4 วิธีกำจัดแรงดันที่เกิดจากสนามแม่เหล็กยุบตัว

2.2.4.1. ใช้ไดโอดมากำจัด(Diode suppression) คือการนำไดโอดมาต่อขนานกับขดลวดนั้น ในขณะที่ทรานซิสเตอร์ on จะไม่มีกระแสไหลผ่านไดโอดเพราะเป็นการให้ไบอัสย้อนกลับแก่ไดโอด เมื่อทรานซิสเตอร์ off Back emf voltage ที่มีค่ามากกว่าแหล่งจ่ายจะถูกทำให้ลัดวงจรโดยกระแสจะไหลผ่านไดโอดและค่าความต้านทานของขดลวด เราเรียกไดโอดที่ทำหน้าที่นี้ว่า Flyback Diode หรือ Free wheeling diode



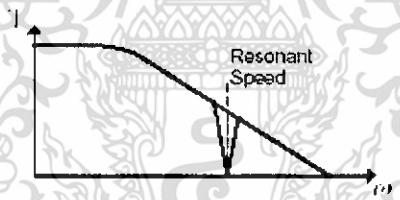
รูปที่ 2.8วงจรที่ใช้ไดโอดมากำจัด

2.2.4.2. ใช้ความต้านทานร่วมกับไดโอด(Diode + Resistance suppression) คือ วิธีการนี้จะทำให้เวลาที่ใช้ในการกำจัด Back emf voltage เร็วกว่าการการใช้ไดโอดเพียงอย่างเดียว การหาค่าความต้านทานหาได้ดังนี้ $Rs_{(max)}=[R*Vce_{(max)}/V]-1$

รูปที่ 2.9 วงจรที่ใช้ความต้านทานร่วมกับไดโอด

2.2.5 คุณสมบัติความเร็วของสเต็ปปิ้งมอเตอร์

คุณสมบัติความเร็วของสเต็ปปิ้งมอเตอร์คือ เมื่อความเร็วต่ำกว่าทอร์คจะมีค่าสูง แต่ที่ความเร็วสูง ๆ และที่ความเร็วรีโซแนนซ์ ทอร์คจะมีค่าต่ำ ความเร็วรีโซแนนซ์จะขึ้นกับรูปแบบการขับมอเตอร์และโหลดของมอเตอร์

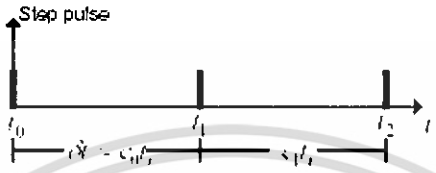


รูปที่ 2.10 Torque Vs. Speed

2.2.5.1 สมการพื้นฐานของสเต็ปปิ้งมอเตอร์

ความสำคัญของการขับสเต็ปปิ้งมอเตอร์ คือ การเปลี่ยนการไหลของกระแสผ่านขดลวดแต่ละขดของสเต็ปปิ้งมอเตอร์ ซึ่งเราได้จากตัววงจรขับสเต็ปปิ้งมอเตอร์ที่จะลำดับการจ่ายพัลส์ให้แก่ขดลวดในทิศทางที่ถูกต้อง

ในการควบคุมความเร็วคงที่ พัลส์ที่ส่งไปยังตัวขับมอเตอร์จะอยู่ในสถานะคงที่



รูปที่ 2.11 รูปลักษณะการป้องพัลส์

ตัวเคาท์เตอร์ทำงานที่ความถี่ f_i(Hz) คาบเวลา

$$t_i = \frac{1}{f_i} \quad \text{(s)} \tag{2.1}$$

ช่วงเวลา δt ควบคุมด้วยค่าเคาท์เตอร์ C

$$\delta t = C_0 t = \frac{C}{f_i} \tag{2.2}$$

มุมแต่ละสเต็ป (α)

$$\alpha = \frac{2\pi}{spr} \quad \text{(rad)} \tag{2.3}$$

เมื่อ spr = จำนวนสเต็ปต่อ 1รอบ

มุมการเคลื่อนที่ (θ)

$$\theta = n\alpha \quad \text{(rad)} \tag{2.4}$$

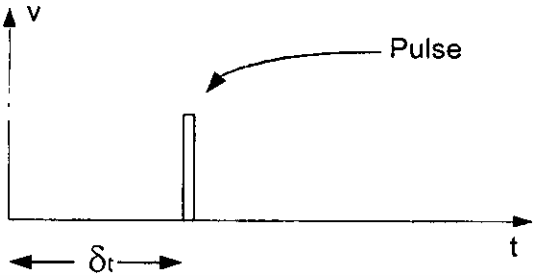
เมื่อ n = จำนวนสเต็ปต่อ 1 รอบ

ความเร็วเชิงมุมเฉลี่ยในการเคลื่อนที่ของแต่ละสเต็ป

$$\omega = \frac{\alpha}{\delta_i} \quad \text{(rad/sec)} \tag{2.5}$$

$$\omega = \frac{\alpha f}{C} \tag{2.6}$$

2.2.5.2 ความสัมพันธ์ระหว่างความเร็วเชิงเส้นและความเร็วเชิงมุม



รูปที่ 2.12 ภาพแสดงการกำเนิด Pulse

โดย Δx = ระยะที่เดินทางในแต่ละสเต็ป

t = เวลาที่ใช้ในการปล่อยพัลส์

$$\frac{\Delta x}{\delta t} = V \longrightarrow \delta t = \frac{\Delta x}{V} \tag{2.7}$$

ในขณะเดียวกัน

$$\frac{\alpha}{\delta t} = \omega \longrightarrow \delta t = \frac{\alpha}{\omega} \tag{2.8}$$

$$\therefore \quad \frac{\alpha}{\omega} = \frac{\Delta x}{V} \tag{2.9}$$

2.2.5.3 การสร้างพัลส์ของความเร็วเชิงเส้นคงที่

จาก

$$\frac{\alpha}{\omega} = \frac{\Delta x}{V}$$

$$\omega = \frac{\alpha \cdot V}{\Delta x} \tag{2.10}$$

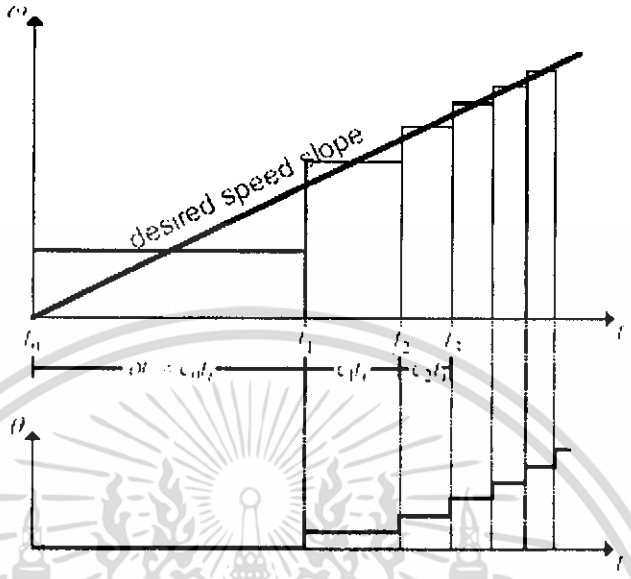
และ

$$\omega = \frac{\alpha f}{C}$$

$$\frac{\alpha f}{C} = \frac{\alpha \cdot V}{\Delta x} \tag{2.11}$$

$$\therefore \quad C = \frac{f \cdot \Delta x}{V} \tag{2.12}$$

2.2.5.4 การเพิ่มความเร็วแบบเชิงเส้น (ความเร่งคงที่)



รูปที่ 2.13 Speed VS. Pulse/sec

การเพิ่มความเร็วแบบเชิงเส้นนั้นเกิดจากความเร่งคงที่ โดย $\omega(t) = \omega'(t)$ อินทิเกรตความเร็วจะได้มุมของการเคลื่อนที่

$$\theta(t) = \int_0^t \omega(\tau)d\tau = \frac{\omega' t^2}{2} = n\alpha \qquad (2.13)$$

โดยที่ $n \geq 0$ เมื่อ n = จำนวนสเต็ปของการหมุน

$$t_n = \sqrt{\frac{2n\alpha}{\omega}} \qquad (2.14)$$

ช่วงเวลาระหว่างสเต็ปที่ n และ n+1 ใช้สมการ (2.2)

$$C_n t_1 = t_{n+1} - t_n \qquad (2.15)$$

$$C_n = f_1(t_{n+1} - t_n) \qquad (2.16)$$

$$C_0 = f\sqrt{\frac{2\alpha}{\omega'}} \qquad (2.17)$$

$$C_n = C_0(\sqrt{n+1} - \sqrt{n}) \qquad (2.18)$$

จะเห็นว่า C_0 เป็นตัวกำหนดความเร่ง ω'

ในสมการที่ (2.18) นั้นต้องมีการคำนวณรากที่สอง จะทำให้เสียเวลาในการคำนวณ

$$\frac{C_n}{C_{n-1}} = \frac{C_0(\sqrt{n+1} - \sqrt{n})}{C_0(\sqrt{n} - \sqrt{n-1})} = \frac{\sqrt{1 + 1/n} - 1}{1 - \sqrt{1 - 1/n}} \qquad (2.19)$$

ใช้อนุกรมเทเลอร์

$$\sqrt{1 \pm \frac{1}{n}} = 1 \pm \frac{1}{2n} - \frac{1}{8n^2} + 0 \cdot (\frac{1}{n^3}) \qquad (2.20)$$

จะได้

$$\frac{C_n}{C_{n-1}} = \frac{4n-1}{4n+1} \qquad (2.21)$$

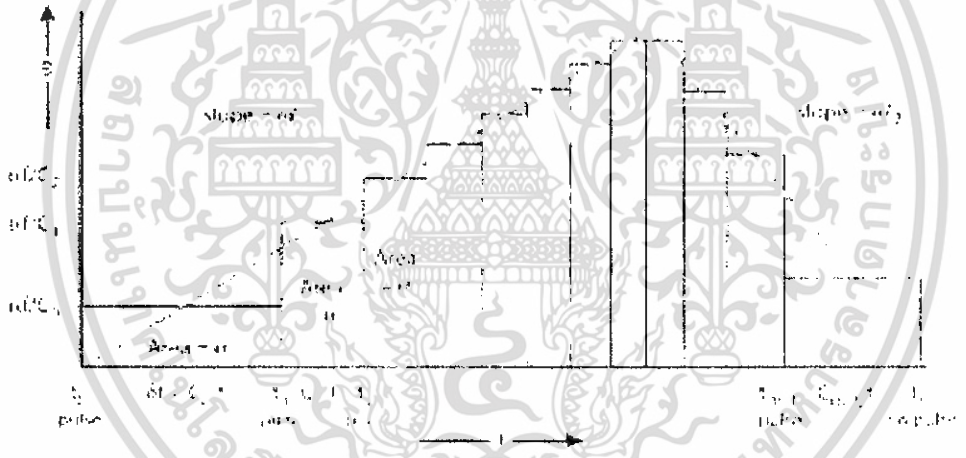
$$C_n = C_{n-1} - \frac{2C_{n-1}}{4n+1} \qquad (2.22)$$

เราสามารถแยกจำนวนสเต็ปที่มอเตอร์หมุน (i) จากจำนวนสเต็ปที่นับจากจุดเริ่มต้นที่ความเร็วเท่ากับศูนย์ (n) ได้ ในการเพิ่มความเร็วจากศูนย์ n_i=i

$$C_i = C_{i-1} - \frac{2C_i - 1}{4n_i + 1} \qquad (2.23)$$

ในการลดความเร็วของสเต็ปปิ้งมอเตอร์ คือการทำให้ n_i มีค่าเป็นลบ ถ้าให้ m เป็นจำนวนสเต็ปที่ใช้ตั้งแต่เพิ่มความเร็วตั้งแต่ศูนย์ถึงลดความเร็วเหลือศูนย์

$$C_i = C_{i-1} - \frac{2C_{i-1}}{4(i-m)+1} \quad \text{เมื่อ } i < m \qquad (2.24)$$



รูปที่ 2.14 การควบคุมโดยการเปลี่ยนแปลงความเร่ง

ความคลาดเคลื่อนจากการใช้อนุกรมเทเลอร์ แทนในสมการที่ (2.19) สามารถลดลงได้โดยใช้

$$C_0 = 0.676 f \sqrt{\frac{2\alpha}{\omega'}} \qquad (2.25)$$

2.2.5.5 การเปลี่ยนความเร่ง

จากความสัมพันธ์ $t_n = \frac{\omega_n}{\omega'}$ และ $n = \frac{\omega' t_n^{\,2}}{2\alpha}$ จะได้

$$n = \frac{\omega^2}{2\alpha\omega'} \tag{2.26}$$

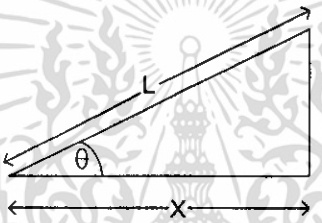
เมื่อ n = จำนวนสเต็ปที่จะต้องใช้ในการเพิ่มความเร็วเพื่อให้ได้ความเร็วที่ต้องการ

เมื่อเปลี่ยนความเร็ว เราจะใช้ความสัมพันธ์

$$n_1\omega_1' = n_2\omega_2' \tag{2.27}$$

2.2.6 การอินเตอร์โพเลชั่นแบบเชิงเส้นด้วยความเร็วคงที่ (Linear Interpolation at constant V)

ในการวาดเส้นตรงบนระนาบ x-y จะเกิดขึ้นได้ก็ต่อเมื่ออัตราส่วนความเร็วของแกน x-y จะต้องมีค่าคงที่ พิจารณารูป

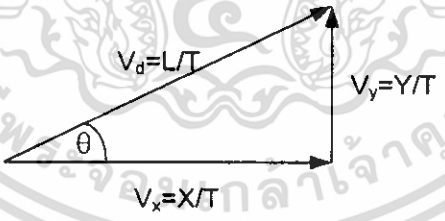


รูปที่ 2.15 ภาพแสดงความยาวของการเคลื่อนที่แต่ละแกน

$$\sin\theta = \frac{Y}{L} \tag{2.28}$$

$$\cos\theta = \frac{X}{L} \tag{2.29}$$

พิจารณาเมื่อเป็นความเร็ว โดยให้ T = เวลาที่ใช้ในการเดินทาง



รูปที่ 2.16 ภาพแสดงความเร็วของแต่ละแกน

$$\sin\theta = \frac{V_y}{V_d} \tag{2.30}$$

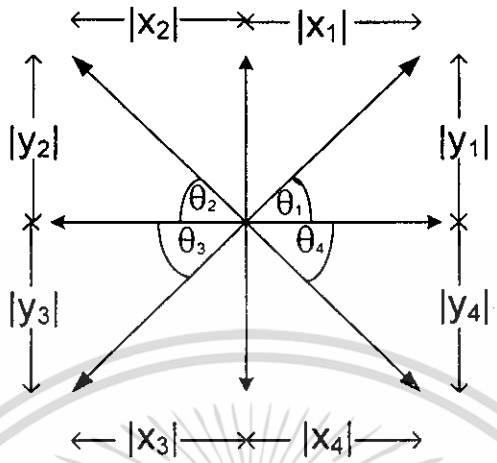
$$\cos\theta = \frac{V_x}{V_y} \tag{2.31}$$

ดังนั้นจะได้ค่า

$$V_x = V_d\cos\theta = V_d \cdot \frac{X}{L} \tag{2.32}$$

$$V_y = V_d\sin\theta = V_d \cdot \frac{Y}{L} \tag{2.33}$$

ภาพของเส้นตรงที่เกิดขึ้นได้



รูปที่ 2.17 ภาพแสดงความเร็วทั้งหมดที่สามารถเกิดขึ้นได้

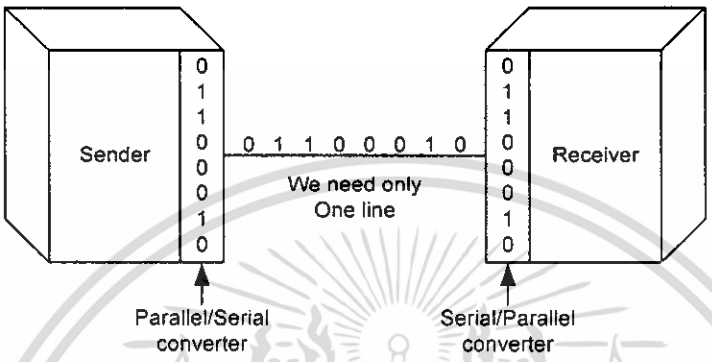
จากรูปจะได้

$$|V_{xi}| = |V_{di}| \bullet \frac{|X_i|}{|L_i|} \tag{2.34}$$

$$|V_{yi}| = |V_{di}| \bullet \frac{|Y_i|}{|L_i|} \tag{2.35}$$

2.3 การสื่อสารข้อมูลแบบอนุกรม(Serial Transmission)

วิธีการสื่อสารข้อมูลแบบอนุกรมนั้น สัญญาณจะทยอยส่งไปตามสายสื่อสารเพียงเส้นเดียว ซึ่งแสดงได้ดังรูปที่ 2.18 จะเห็นได้ว่าบิตจะทยอยส่งออกมาทีละบิตจากต้นทางไปยังปลายทาง โดยปลายทางจะทำการรวบรวมสัญญาณหรือบิตที่ทยอยส่งมาจนครบ 8 บิต หรือ 1 ไบท์ เพื่อนำไปใช้งานต่อไป



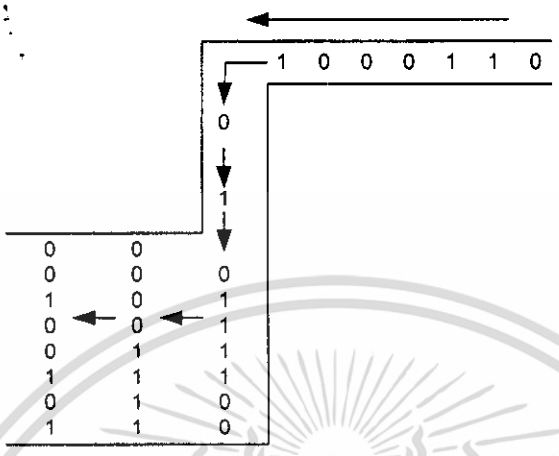
รูปที่ 2.18 การส่งข้อมูลแบบอนุกรม

2.3.1 การแปลงสัญญาณข้อมูลระหว่างแบบอนุกรมและแบบขนาน

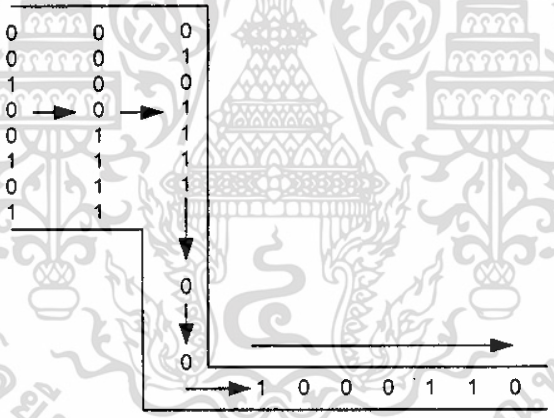
ในการแปลงรูปแบบข้อมูลระหว่างอนุกรมและขนาน จะอาศัยรีจิสเตอร์เพื่อเป็นบัฟเฟอร์ในการเก็บข้อมูลชั่วคราว เช่น ข้อมูลที่ส่งเข้ามาเป็นรูปแบบอนุกรมซึ่งจะส่งเรียงเข้ามาทีละบิต และเมื่อเข้ามาถึงปลายทาง บิตแต่ละบิตจะถูกนำมาจัดเก็บเรียงลำดับกันอยู่ในบัฟเฟอร์ จนกระทั่งครบตามจำนวนบิตที่ต้องการ เช่น เรียงกันจนครบ 8 บิต จากนั้นรีจิสเตอร์ก็จะส่งข้อมูลทั้งชุดนี้ออกไปด้วยการส่งสัญญาณให้หน่วยประมวลผลรับทราบ เพื่อให้โปรแกรมนำไบท์เหล่านั้นไปประมวลผลต่อไป ในขณะที่หากต้องการแปลงข้อมูลในรูปแบบขนานกลับไปเป็นแบบอนุกรม ก็สามารถกระทำได้ด้วยกระบวนการดังกล่าวในทิศทางตรงข้ามโดยพิจารณาจากรูปที่ 2.19 และรูปที่ 2.20 ที่แสดงขั้นตอนการแปลงข้อมูลระหว่างแบบอนุกรมและแบบขนาน ซึ่งกระบวนการแปลงสัญญาณข้อมูลเหล่านี้จะมีวงจรพิเศษที่เรียกว่า **UART(Universal Asynchronous Receiver Transmitter)** ที่ทำการแปลงข้อมูลแบบขนานเป็นแบบอนุกรมหรือแปลงกลับจากแบบอนุกรมเป็นแบบขนาน นอกจากนี้ยังมีวงจรที่เรียกว่า **USART(Universal Synchronous/Asynchronous Receiver Transmitter)** ซึ่งมีคุณสมบัติเช่นเดียวกับ UART แต่จะมีส่วนของการทำงานกับการซิงโครนัสข้อมูลด้วย

ปัญหาของการส่งข้อมูลแบบอนุกรม คือเรื่องแบ่งตัวอักขระแต่ละตัวว่าจะแบ่ง ณ ตำแหน่งบิตใดซึ่งทั้งฝ่ายต้นทางและปลายทางจะต้องมีข้อตกลงร่วมกัน กล่าวคือ ทั้งฝ่ายต้นทางและปลายทางจะต้องรับรู้ร่วมกันว่าจะต้องแบ่งแต่ละตัวอักขระ ณ ตำแหน่งบิตใด เนื่องจากบิตแต่ละบิต

จะทยอยส่งมาเป็นลำดับเรื่อยๆ ดังนั้น การส่งข้อมูลแบบอนุกรมจึงมีวิธีการอยู่ 2 วิธีด้วยกัน คือ "การส่งข้อมูลแบบอะซิงโครนัส" และ "การส่งข้อมูลแบบซิโครนัส"



รูปที่ 2.19 การแปลงข้อมูลจากแบบอนุกรมไปเป็นแบบขนาน(Serial-to-Parallel)



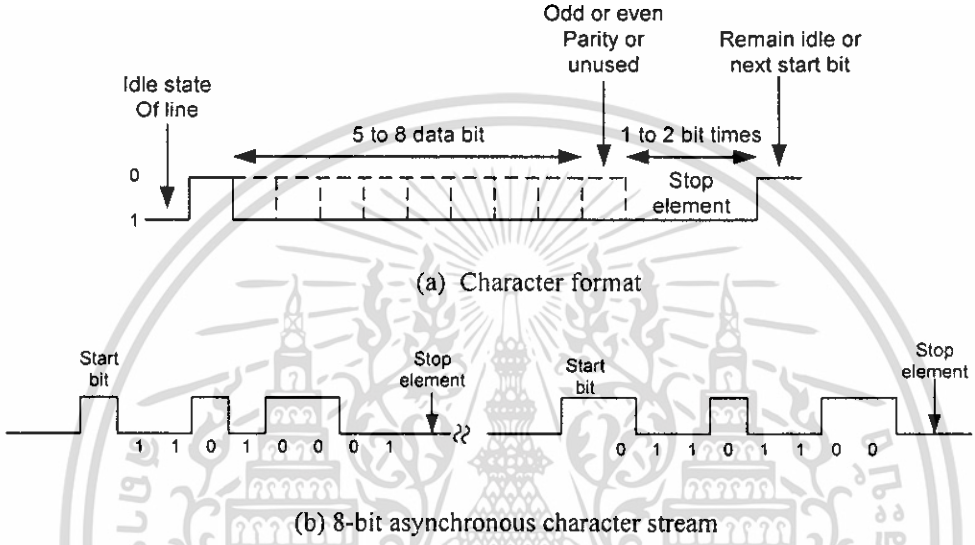
รูปที่ 2.20 การแปลงข้อมูลจากแบบขนานไปเป็นแบบอนุกรม(Parallel-to-Serial)

2.3.2 การส่งข้อมูลแบบอะซิงโครนัส(Asynchronous Transmission)

การส่งข้อมูลแบบอะซิงโครนัสเป็นการสื่อสารด้วยวิธีการส่งอักขระ(Character) แต่ละตัว ณ เวลาใดก็ได้โดยฝ่ายส่งข้อมูลและฝ่ายรับข้อมูลต่างก็มีสัญญาณนาฬิกาควบคุมจังหวะการทำงานด้วยตัวเอง จึงทำให้การทำงานของทั้งสองฝ่ายไม่สอดคล้องกันตามจังหวะนาฬิกา กล่าวคือจะเป็นอิสระต่อกัน แต่อย่างไรก็ตามสัญญาณนาฬิกาทั้งสองฝ่ายนั้นจะต้องมีความถี่เท่ากัน

ในสภาวะนิ่งเฉย(Idle State)ที่ไม่มีการส่งข้อมูลใดๆ(บางครั้งเรียกสภาวะ Marking) จะถูกกำหนดให้สัญญาณมีค่าเป็น "1" แต่เมื่อมีการส่งข้อมูลระดับสัญญาณจะถูกกำหนดให้เป็น "0" อยู่

ช่วงเวลาหนึ่งทำให้เกิดเป็นบิตขึ้นมาหนึ่งบิตที่เรียกว่า **บิตเริ่มต้น(Start Bit)** เพื่อบ่งบอกว่านับจากนี้ไปจะมีข้อมูลส่งมา และเมื่อฝ่ายส่งได้ส่งบิตข้อมูลจนครบตามจำนวนบิตที่ต้องการแล้ว จากนั้นก็จะส่งข้อมูลอีกบิตหนึ่ง ซึ่งระดับสัญญาณจะถูกกำหนดให้เป็น "1" เป็นตัวปิดท้ายที่เรียกว่า **บิตสิ้นสุด(Stop Bit)** เพื่อบ่งบอกให้รู้ว่าได้ส่งข้อมูลครบแล้ว โดยสัญญาณ "1" ที่เป็นบิตสิ้นสุดนี้จะส่งมานานช่วงระยะเวลาหนึ่ง ทำให้ฝ่ายรับได้รับรู้ทันทีว่ามีการส่งบิตสิ้นสุดมาแล้ว



(a) Character format

(b) 8-bit asynchronous character stream

รูปที่ 2.21 แสดงรายละเอียดของรูปแบบการส่งข้อมูลแบบอะซิงโครนัส

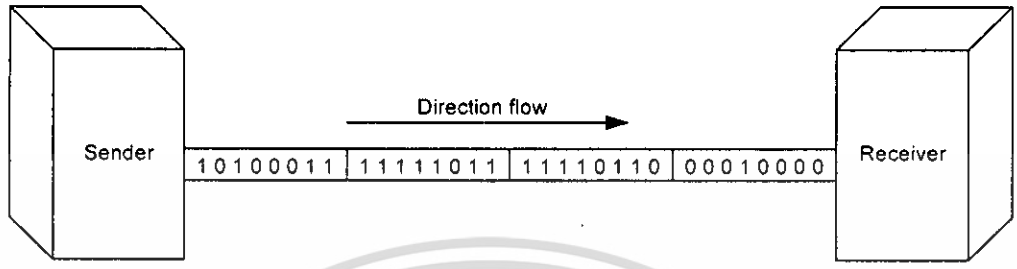
หลังจากที่ได้รับข้อมูลครบตามจำนวนบิต ตัวอักขระตัวที่สองที่จะส่งเป็นลำดับถัดไป ก็ไม่ต้องมีเวลาเท่ากันเสมอ ซึ่งหากยังไม่มีข้อมูล สัญญาณก็จะอยู่ในสถานะนิ่งเฉยเพื่อรอการส่งข้อมูลลำดับถัดไป

2.3.3 การส่งข้อมูลแบบซิงโครนัส(Synchronous Transmission)

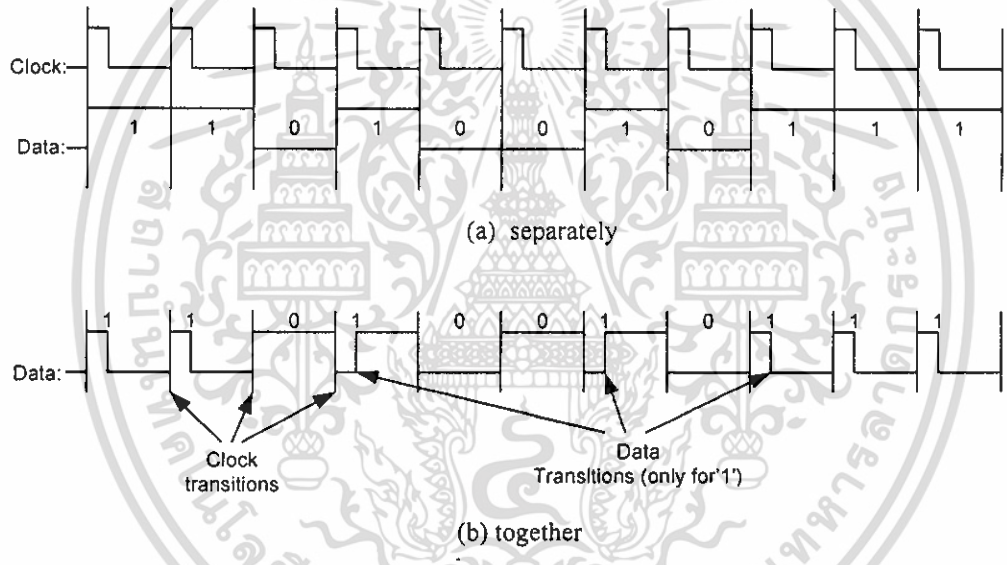
การส่งข้อมูลแบบซิงโครนัสเป็นการส่งข้อมูลแบบกลุ่ม โดยบิตที่ทยอยส่งเข้ามาจะมีการรวมกลุ่มกันให้มีขนาดใหญ่ขึ้นที่เรียกว่าเฟรมหรือบล็อกข้อมูล ซึ่งอาจจะประกอบด้วยหลายๆ อักขระด้วยกัน โดยระหว่างการส่งจะปราศจากช่องว่าง รวมถึงบิตเริ่มต้นและบิตสิ้นสุด ซึ่งทำให้ไม่มีอะไรมาคั่นระหว่างข้อมูลแต่ละตัว จึงทำให้การกะจังหวะ(Timing) กลายเป็นสิ่งสำคัญมากสำหรับการส่งข้อมูลแบบซิงโครนัส กล่าวคือ ทั้งฝ่ายส่งข้อมูลและฝ่ายรับข้อมูลจะต้องทำงานสอดคล้องกันตามจังหวะของสัญญาณนาฬิกาโดยฝ่ายรับข้อมูลจะได้รับสัญญาณนาฬิกามาจากฝ่ายส่งข้อมูล

ดังนั้นวิธีการส่งข้อมูลชนิดนี้สัญญาณนาฬิกาทั้งฝ่ายรับและฝ่ายส่งจะต้องซิงโครไนซ์กัน ฝ่ายส่งอาจส่งสัญญาณนาฬิกาแยกออกมาพร้อมกับข้อมูลดังรูปที่2.23(a) แต่วิธีการนี้จะใช้ได้ดีกับ

ระยะทางใกล้ๆ เพราะจะเกิดปัญหาเรื่องของสัญญาณที่อ่อนตัวลง แต่ก็มีอีกวิธีหนึ่งดังรูปที่ 2.23(b) ซึ่งจะเป็นการนำเอาสัญญาณนาฬิการวมเข้ากับสัญญาณข้อมูล เช่น การเข้ารหัสแบบแมนเชสเตอร์ ในระบบแลน เป็นต้น



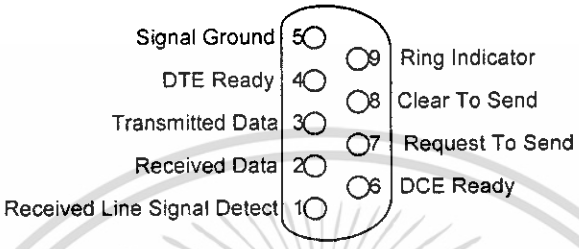
รูปที่ 2.22 ภาพแสดงการส่งข้อมูลแบบซิงโครนัสที่นำเสนอในรูปแบบอย่างง่าย



รูปที่2.23 ภาพแสดงการเข้าจังหวะระหว่างสัญญาณพาฬิกาและข้อมูลในการส่งข้อมูล

2.3.4 มาตรฐาน RS232-C

มาตรฐานRS232-C เป็นมาตรฐานในการรับและส่งข้อมูลระหว่าง DTE (Data Terminal Equipment)กับ DCE (Data Communication Equipment) โดยใช้เทคนิคการสื่อสารไบนารีแบบอนุกรม โดยการสื่อสารแบบอนุกรมที่นิยมใช้กับเครื่องคอมพิวเตอร์เป็นการสื่อสารแบบอะซิงโครนัส รูปที่ 2.24 แสดงลักษณะโครงสร้างของคอนเนคเตอร์ DB-9 ที่ใช้ในการติดต่อสื่อสาร



รูปที่ 2.24 แสดงโครงสร้างคอนเนคเตอร์ DB-9

Received Line Signal Detect : ขานี้จะ active เมื่อมีการส่งสัญญาณ Carrier จากโมเด็ม

Received Data : ขานี้สำหรับรับข้อมูลอนุกรมจากคอมพิวเตอร์

Transmitted Data : ขานี้สำหรับส่งข้อมูลอนุกรมจากคอมพิวเตอร์

DTE Ready : ใช้บอกอุปกรณ์ปลายทางว่าต้องการติดต่อ

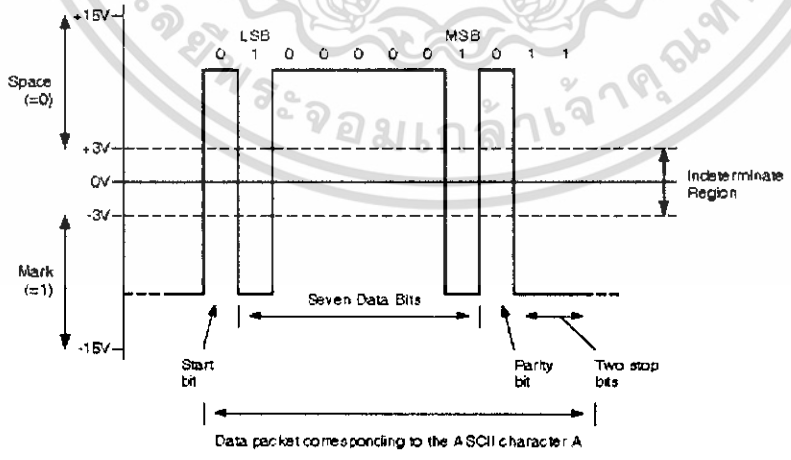
Signal Ground : กราวด์ของวงจร

DCE Ready : ใช้คู่กับขา DTE Ready ในการตรวจสอบการติดต่อ

Request To Send : ใช้บอกอุปกรณ์ปลายทางให้ส่งข้อมูลกลับมาให้

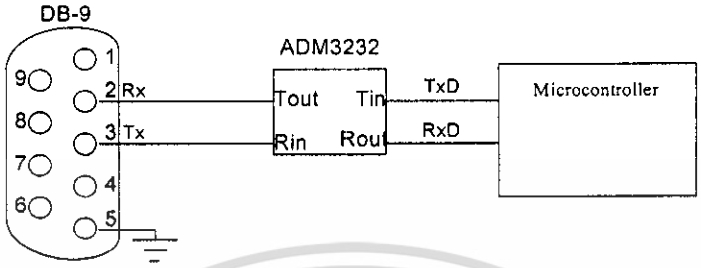
Clear To Send : ใช้ตรวจสอบว่าอุปกรณ์ปลายทางพร้อมส่งข้อมูลหรือไม่

Ring Indicator : ขานี้จะ active เมื่อโมเด็มได้รับสัญญาณจากโทรศัพท์



รูปที่ 2.25 แสดงระดับแรงดันที่ใช้ในการสื่อสารข้อมูลของ RS232-C

ในการเชื่อมต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์นั้นแต่ละส่วนมีมาตรฐานต่างกันจึงต้องทำการแปลงแรงดันให้อยู่ในมาตรฐานการใช้งานของไมโครคอนโทรลเลอร์โดยใช้วงจรรวมเบอร์ ADM3232



รูปที่ 2.26 แสดงการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับ
คอมพิวเตอร์ผ่าน RS232-C

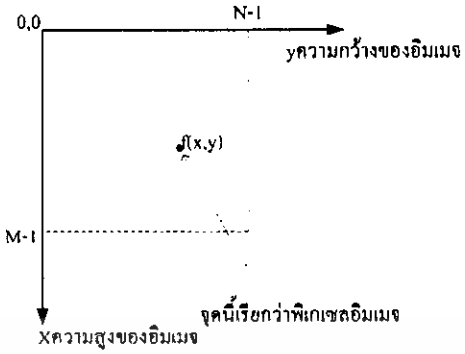
2.4 การทำกระบวนการอิมเมจดิจิตอล (Digital Image Processing)

2.4.1. คำนิยามและความหมายของการทำกระบวนการอิมเมจดิจิตอล

ก่อนที่จะเข้าสู่กรทำกระบวนอิมเมจดิจิตอล จะเริ่มต้นที่อิมเมจก่อน โดยปกติแล้วสายตาของบุคคลทั่วไปจะมองเห็นภาพทิวทัศน์วิวต่าง ๆ เป็นลักษณะแบบอนาลอก ซึ่งสามารถอธิบายได้ด้วยคณิตศาสตร์ที่มีตัวแปรแบบนับได้อย่างต่อเนื่อง แต่เครื่องคอมพิวเตอร์จะใช้เลขฐานสองเป็นหลักในการคำนวณ เมื่อนำภาพอิมเมจมาแปลงเข้าสู่เครื่องคอมพิวเตอร์

อิมเมจดิจิตอล

อิมเมจดิจิตอลเป็นผลมาจากการสุ่มค่าในระบบพิกัด Space หรือ Spatial Coordinate ดังรูปที่2.27 และการทำ Quantization ของค่าระดับความสว่าง (Brightness Value) หรือความเข้ม (Intensity) ระบบพิกัด Space จะใช้กับการแสดงอิมเมจดิจิตอล ซึ่งจะมีขนาดความกว้างและความสูงของอิมเมจแสดงในแกน Y และ X ตามลำดับ ส่วนจุดใด ๆ ที่วางบนระนาบ XY จะเป็นฟังก์ชัน f(x,y) และเรียกว่า พิกเซล (Pixel) ที่แสดงถึงค่าระดับความเข้ม ซึ่งจะเป็นจำนวนที่นับได้จำกัด (Finite Number) แบบไม่ต่อเนื่อง หรือเรียกว่า *Discreate Quantity* ค่า Discreate Quantity เป็นผลมาจากการทำ Quantization โดยจะใช้การแปลงจากอนาล็อก (Analog) เป็นดิจิตอล (Digital)

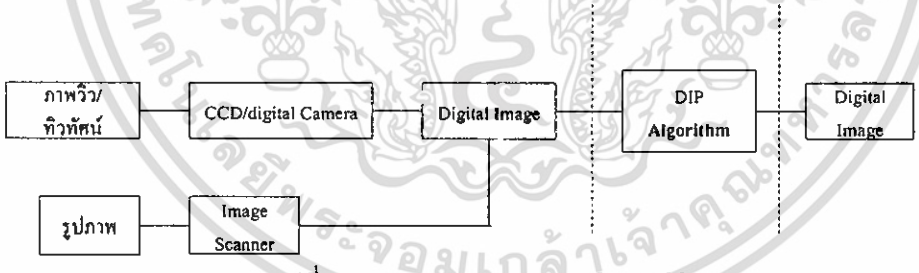


รูปที่2.27 ระบบพิกัด Space

2.4.2 การทำกระบวนการอิมเมจดิจิตอล

มีสองสาเหตุใหญ่ ๆที่ต้องการทำกระบวนการอิมเมจดิจิตอล คือเพื่อปรับปรุงอิมเมจดิจิตอลให้มองเห็นได้ง่ายขึ้น และเพื่อปรับอิมเมจให้หุ่นยนต์ตีความหมาย หรือเข้าใจจดจำรูปร่างลักษณะได้อย่างแม่นยำ ตัวอย่างเช่น การจำจำตัวอักษร หรือ Optical Character Recognition (OCR) ที่สามารถจดจำตัวอักขระได้ถึง 99.9% การปรับปรุงอิมเมจให้ใช้พื้นที่เก็บน้อยลง การตรวจสอบลายพิมพ์มือของแต่ละบุคคล เป็นต้น

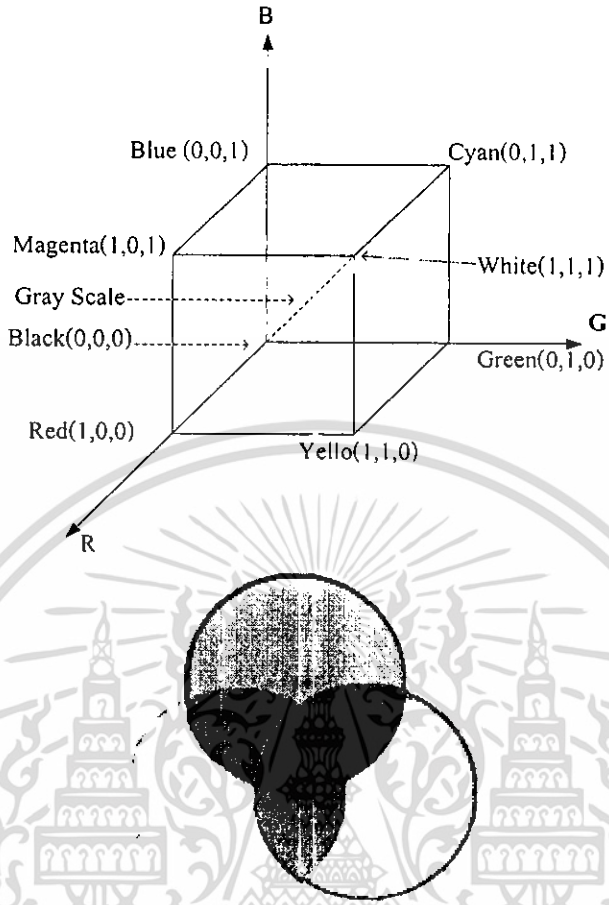
การทำกระบวนการอิมเมจดิจิตอล คือ การนำอิมเมจดิจิตอลเข้ามาทำกระบวนการโดยการใส่ฟังก์ชันอัลกอริทึมต่าง ๆ เข้าไป ก็จะได้เอาท์พุตเป็นอิมเมจดิจิตอล ที่ตรงตามแนวความคิดของการทำกระบวนการอิมเมจดิจิตอล เราสามารถแสดงภาพตั้งแต่การนำภาพวิว ทิวทัศน์ จนถึงเอาท์พุตดังรูปที่ 2.28



รูปที่ 2.28 การทำกระบวนการอิมเมจดิจิตอล

2.4.3 โมเดลสี (Color Model)

โมเดลสี หรือ color space ประกอบด้วย 3 แม่สีหลัก ได้แก่ สีแดง เขียว และน้ำเงิน ถ้านำแต่ละแม่สีมาพล็อตกราฟในระบบพิกัด color space โดยแต่ละสีมีค่า 0 ถึง 1 (0 แสดงถึงความมืด และ 1 แสดงถึงความสว่าง) จะได้ภาพการผสมสีทางแสดงหรือการบวกแม่สีเข้าด้วยกัน (Additive Primary Color) ดังรูปที่ 2.29



รูปที่ 2.29 โมเดลสีและการผสมสีทางแสง

ในบางครั้งถ้าต้องการแปลงโมเดลสีให้เป็นขาวดำ ซึ่งก็คือ Gray Scale จะใช้สมการที่ 2.36

$$\text{Gray Scale} = 0.299*R + 0.587*G + 0.114*B \quad (2.36)$$

แต่เราสามารถใช้อีกสมการ โดยการหาค่าเฉลี่ยทั้ง 3 สี ดังนี้

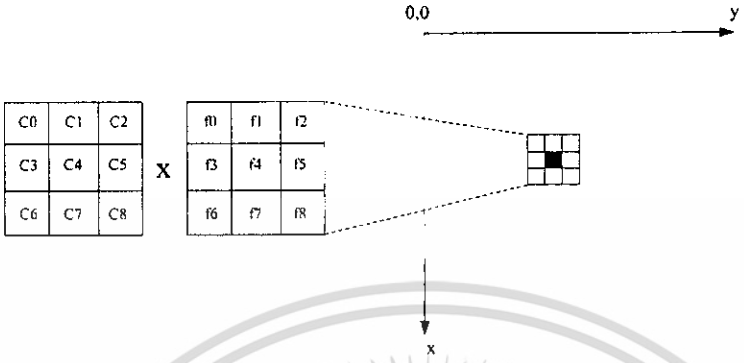
$$\text{Gray Scale} = \frac{R+G+B}{3} \quad (2.37)$$

2.4.4 การทำกระบวนการรอบ ๆ พิกเซล (Area Processing)

การทำกระบวนการนี้จะเอาค่าบริวณรอบ ๆ พิกเซลมาคำนวณ แล้วนำมาแทนค่าพิกเซลเดิม บริเวณรอบ ๆ พิกเซลนี้เรียกได้หลายๆ แบบ ตัวอย่างเช่น Filter Window , Kernel หรือ Mask ส่วนใหญ่มักจะนิยมเรียก Kernel หรือ Mask การคำนวณบริเวณรอบ ๆ พิกเซล อาจเรียกว่า การทำ Convolution

การทำConvolution คือ การหาผลรวมที่ได้ถ่วงน้ำหนักแล้วบริเวณรอบ ๆ จุดพิกเซล ค่าถ่วงน้ำหนักนี้เรียกว่า Mask Coefficient หรือ Kernel Coefficient โดยปกติจะกำหนดค่าถ่วงน้ำหนักเป็นเมตริกขนาด nxn โดย n เป็นเลขคี่ เช่น 3,5,7,... เป็นต้น เพราะว่าสามารถหาค่าจุด

พิกเซลตรงกลางได้ ในที่นี้จะใช้เมตริกขนาด 3x3 เป็นหลัก เราสามารถแสดงวิธีการคำนวณและสมการดังรูปที่ 2.30



รูปที่ 2.30 วิธีการคำนวณ Convolution

ค่าพิกเซลใหม่ f'(4) จะเท่ากับ $(c_0f_0+c_1f_1+c_2f_2+c_3f_3+c_4f_4+c_5f_5+c_6f_6+c_7f_7+c_8f_8)$

หรือ
$$f'(4)=\sum_{i=0}^{8} c_i f_i \qquad (2.38)$$

สมการที่ (2.38) เรียกว่า Linear filter ของอิมเมจ

ผลรวมค่าถ่วงน้ำหนัก (Mask Coefficient) แสดงดังสมการ

$$\sum_{i=0}^{8} c_i \qquad (2.39)$$

จากสมการที่ (2.39) ค่าผลรวมถ่วงน้ำหนักจะมีผลกระทบต่อระดับความสว่างตลอดทั้งอิมเมจ โดยทั่วไปแล้วผลรวมจะเป็นค่า 1 แต่ถ้าผลรวมเท่ากับหรือน้อยกว่า 0 อิมเมจจะมีความมืดมากขึ้น ซึ่งจะพบได้ในการตรวจหาขอบอิมเมจหรือ Edge Detection

จากสมการที่ (2.38) ถ้านำมาหารด้วยสมการที่ (2.39) เราจะได้สมการฟิลเตอร์ (Filter) ในรูปแบบทั่วๆ ไป ดังนี้

$$f'(4) = \frac{\sum_{i=0}^{8} c_i f_i}{\sum_{i=0}^{8} c_i} \qquad (2.40)$$

โดยที่สมการที่ (2.40) ต้องไม่เท่ากับ 0 หรือ ถ้าเท่ากับ 0 จะเซตเป็น 1

<u>หมายเหตุ</u> ถ้าค่าที่คำนวณได้มีค่ามากกว่าระดับความเข้มพิกเซล จะกำหนดให้เท่ากับค่าสูงสุด และถ้าค่าที่คำนวณได้น้อยกว่าระดับความเข้มพิกเซล จะกำหนดให้เท่ากับค่า 0

ค่า Mask Coefficient สามารถกำหนดได้หลายรูปแบบ ในแต่ละรูปแบบก็จะมีผลต่อพิกเซลตลอดทั้งอิมเมจทำให้ค่า Mask Coefficient มีชื่อเรียกแตกต่างกันไป ได้แก่ Filter Low Pass, Filter High Pass , Edge Detector ซึ่งจะกล่าวเฉพาะ Edge Detector

2.4.5 การตรวจหาขอบอิมเมจ (Edge Detector)

ขอบอิมเมจ ก็คือชุดของพิกเซลต่อ ๆ กันที่วางอยู่บนขอบระหว่างพื้นที่สองส่วนของอิมเมจ ขอบอิมเมจจะช่วยอธิบายถึงรูปร่าง ลักษณะ ขนาด และอื่น ๆ ของอิมเมจ การตรวหาขอบอิมเมจเป็นก้าวแรกในการแยกอิมเมจเป็นส่วน (Image Segmentation) แล้วนำมาประกอบรวมกันเป็นออบเจ็กต์หรือพื้นที่ใหม่ (Region) ในที่นี้เราจะพิจารณาการตรวจหาขอบอิมเมจด้วยอนุพันธ์อันดับหนึ่งและสอง

อนุพันธ์อันดับหนึ่ง (First Order Derivative) ได้แก่ โอเปอเรเตอร์ Sobel , Prewitt และ Frei-Chen ส่วนอนุพันธ์อันดับสอง (First Order Derivative) ได้แก่ โอเปอเรเตอร์ Laplacian

2.4.5.1 อนุพันธ์อันดับหนึ่ง ถ้านำมาหาขอบอิมเมจในทิศทางแนวนอน และแนวตั้ง ก็จะหาอนุพันธ์สมการที่ 1 เราเรียกว่าการหา Gradient ซึ่งเป็นเวคเตอร์เมตริก แสดงดังนี้

$$\nabla f = \begin{bmatrix} H_r(x,y) \\ H_c(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix} \tag{2.41}$$

โดยที่ $\frac{\partial}{\partial x} f(x,y)$ จะเป็นการหาขอบอิมเมจในทิศทางแนวนอน (x) และ $\frac{\partial}{\partial y} f(x,y)$ เป็นการหาขอบอิมเมจในทิศทางแนวตั้ง (y) ถ้าหาทั้งสองทิศทาง จะเป็นขนาดของเวคเตอร์(Magnitude Vector) ของสมการ(2.41) ซึ่งจะเขียนใหม่ได้เป็น

$$\nabla f = \left[H_r^2(x,y) + H_c^2(x,y) \right]^{1/2} \cong |H_r(x,y)| + |H_c(x,y)| \tag{2.42}$$

ส่วนทิศทางการตรวจหาขอบอิมเมจได้จากสมการที่ 7

$$\theta = \tan^{-1}\left[\frac{H_c(x,y)}{H_r(x,y)} \right] \tag{2.43}$$

2.4.5.1.1 โอเปอร์เรเตอร์ Sobel จะใช้ค่า Mask Coefficient ดังนี้

ในทิศทางแนวนอน$H_r(x,y)$

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{หรือ} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

ในทิศทางแนวตั้ง $H_c(x,y)$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{หรือ} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

2.4.5.1.2 โอเปอร์เรเตอร์ Prewitt จะใช้ค่า Mask Coefficient ดังนี้

ในทิศทางแนวนอน$H_r(x,y)$

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{หรือ} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

ในทิศทางแนวตั้ง $H_c(x,y)$

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{หรือ} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

2.4.5.1.3 โอเปอร์เรเตอร์ Fri-Chen จะใช้ค่า Mask Coefficient ดังนี้

ในทิศทางแนวนอน$H_r(x,y)$

$$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \text{หรือ} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}$$

ในทิศทางแนวตั้ง $H_c(x,y)$

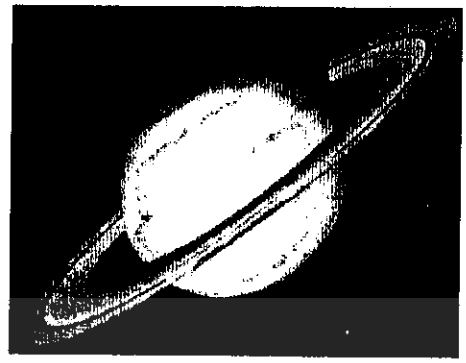
$$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \text{หรือ} \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$$

2.4.5.2 อนุพันธ์อันดับสอง โดยการนำสมการที่ (2.44) มาหาค่าอนุพันธ์อีกครั้ง จะได้สมการใหม่ดังนี้

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2}{\partial x} f(x,y) \\ \frac{\partial^2}{\partial y} f(x,y) \end{bmatrix} \tag{2.44}$$

อนุพันธ์อันดับสองจะมีโอเปอร์เรเตอร์ Laplacian ซึ่งมีค่า Mask Coefficient ดังนี้

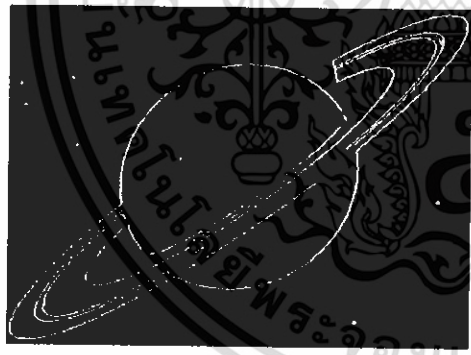
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \text{หรือ} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



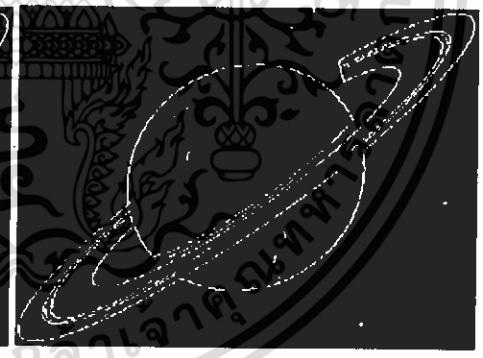
ก)อิมเมจต้นฉบับ

ข) ในทิศทาง H_c

ค) ในทิศทาง H_r



ง) การหาขอบจากโอเปอเรเตอร์ของPrewitt

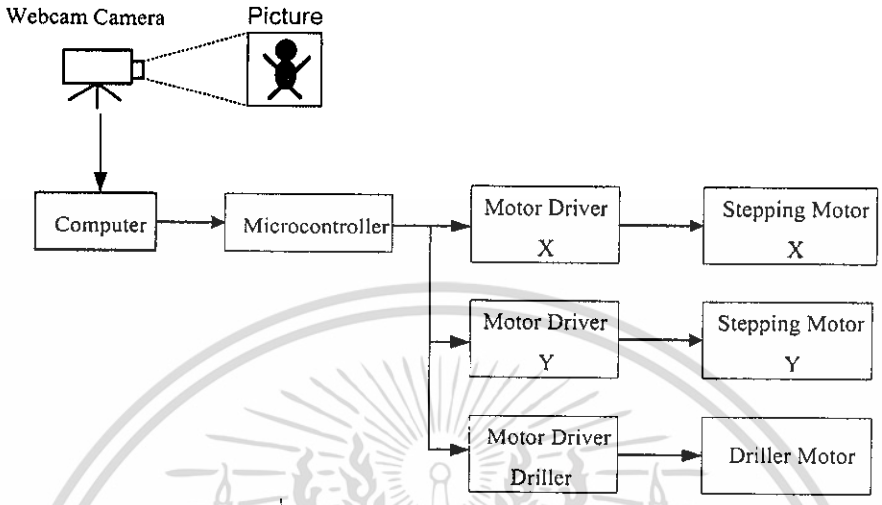


จ) การหาขอบจากโอเปอเรเตอร์ของSobel

รูปที่ 2.31 แสดงตัวอย่างการหาขอบภาพ

บทที่ 3 การออกแบบ

3.1 การทำงานของระบบ



รูปที่ 3.1 การทำงานโดยรวมของระบบ

ผู้ใช้งานจะกำหนดข้อมูลและคำสั่งต่างผ่านโปรแกรมคอมพิวเตอร์ที่เขียนขึ้นมาด้วย visual basic 6.0 โดยมีกล้องเวบแคมช่วยในการรับข้อมูลภาพจากผู้ใช้งานหลังจากนั้นคอมพิวเตอร์จะทำการประมวลผลภาพที่ได้รับจากผู้ใช้งาน เพื่อส่งข้อมูลหรือคำสั่งไปยังคอนโทรลเลอร์ ผ่านพอร์ท RS-232

ไมโครคอนโทรลเลอร์ที่ใช้ในโครงงานชิ้นนี้คือ ARM7 เบอร์ LCP2138 โดยเมื่อไมโครคอนโทรลเลอร์ได้รับคำสั่งจากคอมพิวเตอร์แล้วนำมาประมวลเพื่อส่งพัลส์และทิศทางไปยังตัวขับสเต็ปปิ้งมอเตอร์แกน X และแกน Y และสั่งเคลื่อนที่สว่านตัวเจาะ การทำงานในแต่ละส่วนมีดังนี้

Motor Driver ของแกน X เป็นตัวรับสัญญาณพัลส์และทิศทางจากไมโครคอนโทรลเลอร์เพื่อทำการส่งสัญญาณไฟฟ้าไปยังเฟสต่าง ๆ ของสเต็ปปิ้งมอเตอร์ที่ควบคุมแกน X

Motor Driver ของแกน Y เป็นตัวรับสัญญาณพัลส์และทิศทางจากไมโครคอนโทรลเลอร์เพื่อทำการส่งสัญญาณไฟฟ้าไปยังเฟสต่าง ๆ ของสเต็ปปิ้งมอเตอร์ที่ควบคุมแกน Y

Motor Driver of Driller เป็นตัวรับสัญญาณควบคุมจากไมโครคอนโทรลเลอร์เพื่อส่งพัลส์และทิศทางไปยังมอเตอร์ตัวเจาะ

Stepping Motor ของแกน X เป็นสเต็ปปิ้งมอเตอร์ควบคุมการเคลื่อนที่ทางแกน X

Stepping Motor ของแกน Y เป็นสเต็ปปิ้งมอเตอร์ควบคุมการเคลื่อนที่ทางแกน Y

Driller Motor เป็นมอเตอร์ควบคุมความสูงของหัวเจาะว่าต้องการความลึกขนาดไหน

Webcam Camera เป็นตัวจับภาพจากชิ้นงานเพื่อส่งสัญญาณภาพไปยังคอมพิวเตอร์เพื่อประมวลผลภาพต่อไป

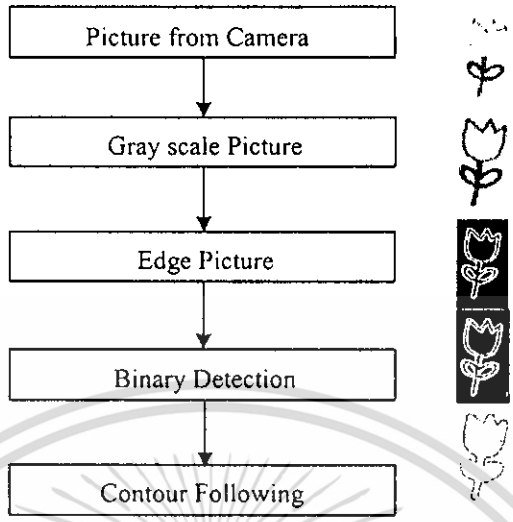
3.2 ขั้นตอนการออกแบบ

3.2.1 การออกแบบการประมวลผลภาพ

สำหรับการประมวลผลภาพ จะนำภาพดิจิตอลที่ได้จากกล้องเวปแคมซึ่งอยู่ในระบบพิกัด Space ซึ่งมีความกว้างและความสูงของภาพอยู่ในลักษณะ 2 มิติ และที่แต่ละจุดบนระนาบ xy หรือที่เรียกว่า พิกเซลจะมีค่าความสว่างหรือค่าสีที่แตกต่างกันไปในแต่ละพิกเซล ค่าความสว่างหรือค่าสีที่ได้นี้จะอยู่ในรูปของตัวเลขทางคณิตศาสตร์จึงนำค่าต่าง ๆ เหล่านี้มาผ่านการคำนวณและอัลกอริทึม ก็จะได้ดิจิตอลอิมเมจ จากนั้นนำอิมเมจดิจิตอลไปผ่านการปรับปรุงภาพเพื่อให้ลักษณะตรงตามที่เราต้องนำก่อนนำไปใช้งาน

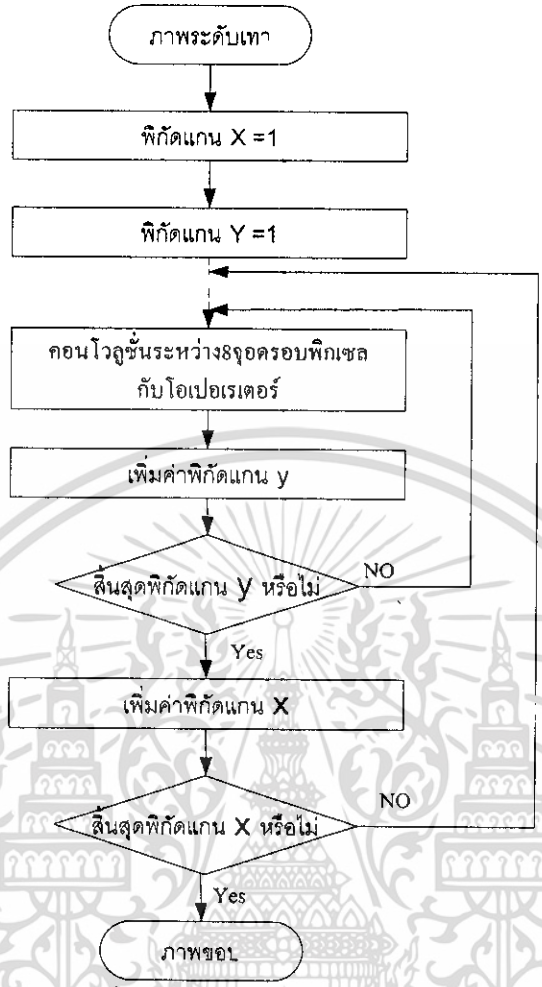
ในการออกแบบครั้งนี้จะนำภาพที่ได้จากกล้องเวปแคมมาทำให้อยู่ในรูปของภาพความเข้มระดับเทา(Gray Scale Picture) โดยการนำค่าสีที่อยู่ในรูป RGB ที่มีอยู่ความเข้มสี Red นำมา1 ค่า Green นำมา 1 ค่า และ Blue นำมา 1ค่า มาแปลงให้อยู่ในค่าความเข้มระดับเทาโดยจะเหลือค่าสีเพียง 1 ค่า คือ Gray ที่มีค่าความสว่างสูงสุดหรือสีขาวเท่ากับ 255 และค่าความสว่างต่ำสุดหรือสีดำเท่ากับ 0 จากนั้นหาขอบภาพโดยนำค่าในภาพระดับเทาแต่ละพิเซล ไปผ่านกระบวนการคำนวณด้วยวิธีการคอนโวลูชั่นของจุดรอบๆ พิกเซลกับโอเปอร์เรเตอร์ที่ใช้สำหรับการหาขอบภาพ ในครั้งนี้ใช้โอเปอร์เรเตอร์ของ Sobel ซึ่งเป็นการหาอนุพันธ์อันดับหนึ่ง จากนั้นจะนำภาพความเข้มระดับเท่าที่ผ่านกระบวนการคำนวณหาขอบภาพไปแบ่งเทรสโฮล (Threshold) เพื่อทำให้เป็นภาพไบนารี โดยใช้หลักที่ว่า หากค่าระดับเท่ามากกว่าค่าเทรสโฮลจะให้จุดนั้นมีค่าเป็น 255 แต่ถ้าน้อยกว่าค่าเทรสโฮลจะมีค่าเป็น0 ดังนั้นจึงได้ภาพไบนารี ภาพที่ได้นี้มีค่าเพียงสองในแต่ละพิกเซลเพียง 2 ค่าเท่านั้น จากนั้นนำภาพไบนารี่นี้ไปทำกระบวนการ Contour Following เพื่อหาจุดล้อมรอบขอบภาพที่เราหาได้ก่อนหน้านี้ โดยในการหาคอนทัวร์ของภาพจะใช้การเปลี่ยนค่าสีในภาพไบนารี่จากค่า 0 เป็น 255 เพื่อหาขอบรอบนอกของภาพไบนารี่ ก่อนที่จะสั่งจุดคอนทัวร์ของภาพไปให้ไมโครคอนโทรลเลอร์สั่งให้เครื่องเครื่องทำงานตามจุดต่าง ๆ ที่ผ่านกระบวนการประมวลผลภาพต่อไป

ขั้นตอนการทำงานในส่วนของการประมวลผลภาพมีดังนี้

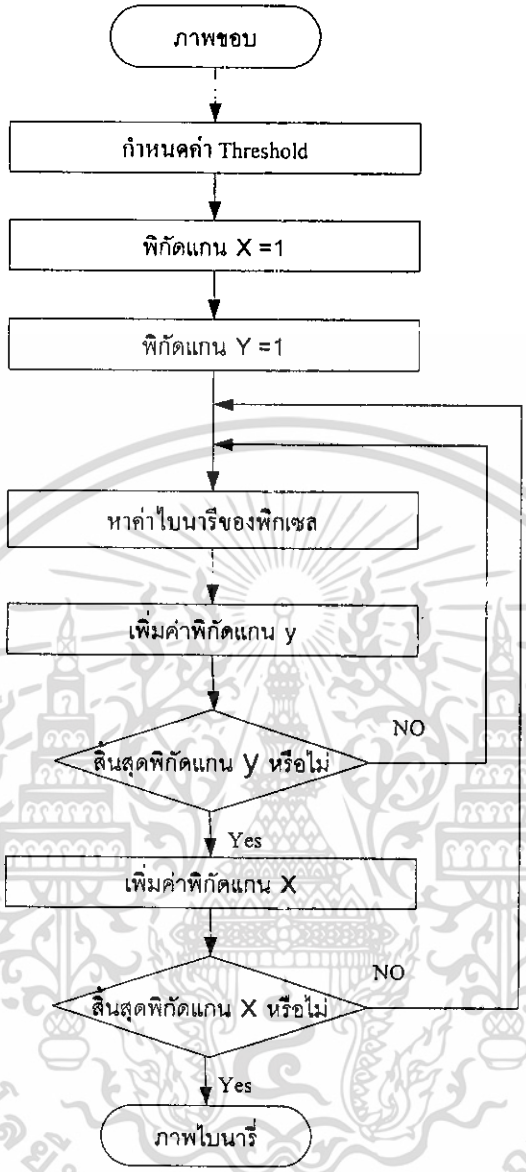


รูปที่ 3.2 ขั้นตอนการประมวลผลภาพ

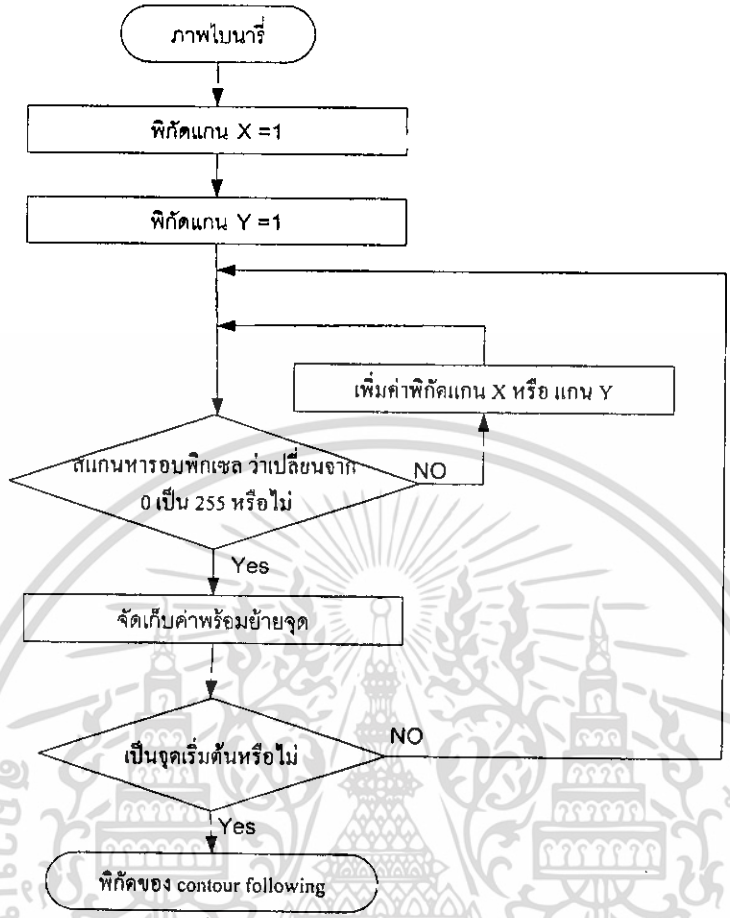
รูปที่ 3.3 โฟลวชาร์ตของการหาภาพระดับเทา



รูปที่ 3.4 โฟลวชาร์ตของการหาภาพขอบ



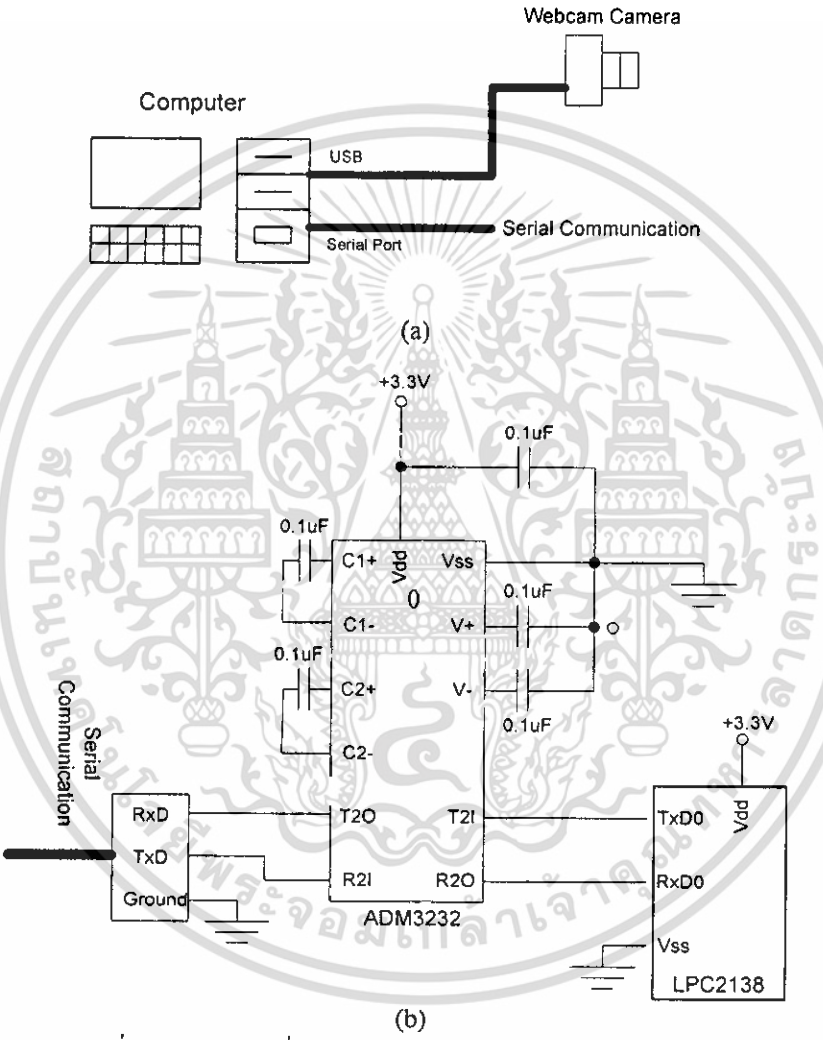
รูปที่ 3.4 โฟลวชาร์ตของการหาภาพไบนารี่



รูปที่ 3.6 โฟลวชาร์ตของการหาคอนทัวร์

3.2.2 ส่วนการออกแบบการติดต่อสื่อสาร

การสื่อสารระหว่างกล้องเวบแคมกับคอมพิวเตอร์จะติดต่อผ่านทางพอร์ตUSB ส่วนการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับไมโครคอนโทลเลอร์จะติดต่อผ่านทางพอร์ตสื่อสารอนุกรม (serial port) แสดงให้เห็นดังรูปที่ 3.7(a) ในการสื่อสารระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์นั้นมีมาตรฐานของระดับแรงดันที่ใช้ในการติดต่อสื่อสารต่างกันจึงจำเป็นที่ต้องมีวงจรรวมที่ทำหน้าที่แปลงระดับแรงดันให้สามารถติดต่อกันได้โดยใช้เบอร์ADM3232โดยแสดงดังรูปที่ 3.7 (b)



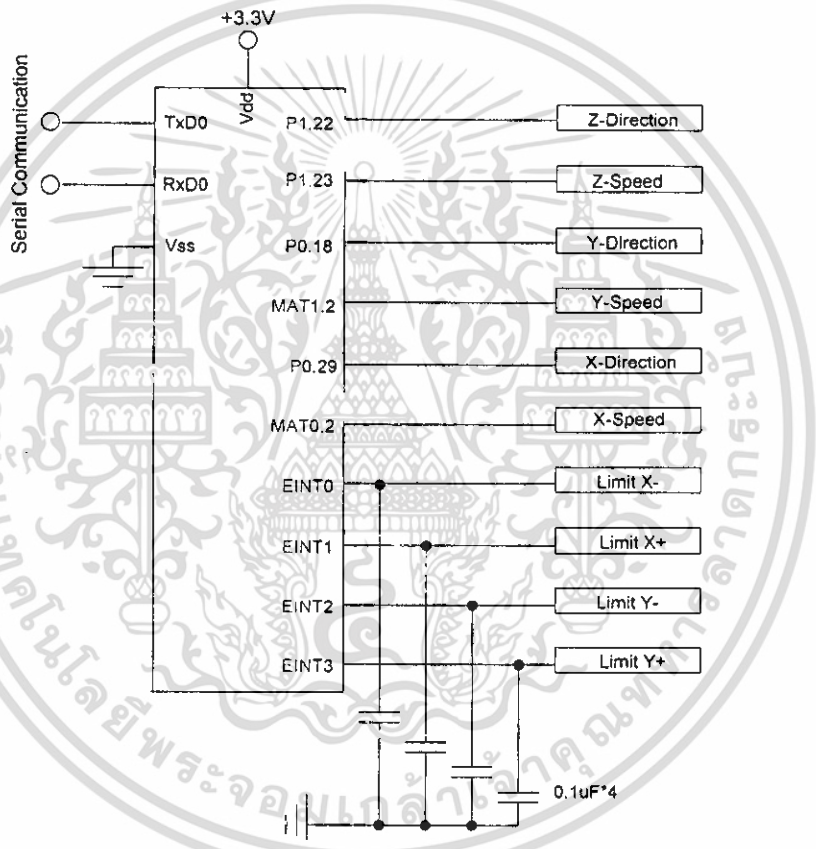
รูปที่ 3.7 แสดงการเชื่อมต่อระหว่างคอมพิวเตอร์กับกล้องเวบแคม และไมโครคอนโทรลเลอร์

ในการสื่อสารระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์เป็นการสื่อสารด้วยRS-232 ผ่านโมดูล UART ภายในไมโครคอนโทรลเลอร์ด้วยข้อมูลแบบ 8 บิต โดยการสื่อสารคอมพิวเตอร์จะส่งคำสั่งที่ต้องการมายังไมโครคอนโทรลเลอร์หากคอนโทรลเลอร์ได้รับคำสั่งแล้วจะนำไป

ดำเนินการต่อไป หากเป็นการส่งตำแหน่งคอมพิวเตอร์จะส่ง8 บิต มา 4 ครั้ง เนื่องจากไมโคร-คอนโทรลเลอร์เป็นแบบ 32 บิต

3.2.3 การออกแบบวงจรส่วนของไมโครคอนโทรลเลอร์

วงจรในส่วนของไมโครคอนโทรลเลอร์จะเป็นการรับข้อมูลจากการสื่อสารอนุกรมมาทำการประมวลผลและสั่งคำสั่งควบคุมการเคลื่อนที่ของสเต็ปมอเตอร์ทั้งในแกนX, แกนY และในแกนZ โดยเราจะป้อนพัลส์กำหนดความเร็วในการเคลื่อนที่และเลือกทิศทางการเคลื่อนที่ให้กับวงจรขับสเต็ปมอเตอร์ และยังมีการรับอินเตอร์รัปอินพุทเพื่อป้องกันการเคลื่อนที่เกินขอบเขตที่กำหนดไว้ด้วย



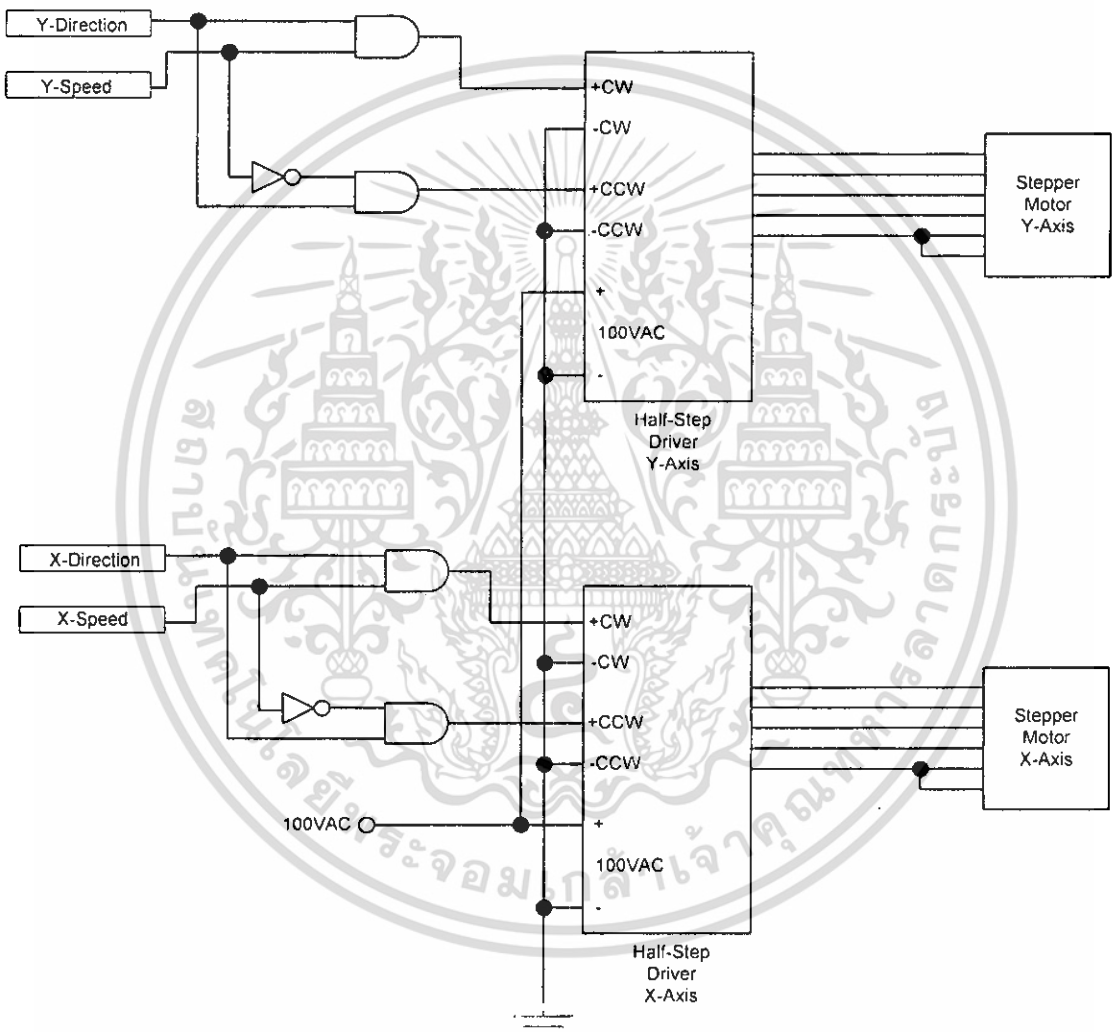
รูปที่ 3.8 แสดงวงจรเชื่อมต่อของไมโครคอนโทรลเลอร์

จากรูปที่ 3.8 ขาP1.22 ควบคุมทิศทางการเคลื่อนที่ในแกนZ ขาP1.23 ป้อนพัลส์ให้วงจรขับสเต็ปมอเตอร์ควบคุมความเร็วการเคลื่อนที่ในแกนZ ขาP0.18 ควบคุมทิศทางการเคลื่อนที่ในแกน Y ขา MAT1.2 ป้อนพัลส์ให้วงจรขับสเต็ปมอเตอร์ควบคุมความเร็วการเคลื่อนที่ในแกนY ขา P0.29 ควบคุมทิศทางการเคลื่อนที่ในแกนX ขา MAT0.2ป้อนพัลส์ให้วงจรขับสเต็ปมอเตอร์ควบคุมความเร็วการเคลื่อนที่ในแกน X ขาEINT0,EINT1,EINT2,EINT3 ทำหน้าที่รับอินเตอร์รัป

อินพุทจากลิมิตสวิตซ์เพื่อป้องกันการเคลื่อนที่ออกนอกขอบเขตที่ต้องการ ตัวเก็บประจุทำหน้าที่บายพาสต์สัญญาณความถี่สูงลงกราวด์เพื่อป้องกันการอิเตอร์รัปซ้ำซ้อน

3.2.4 การออกแบบวงจรขับสเต็ปมอเตอร์

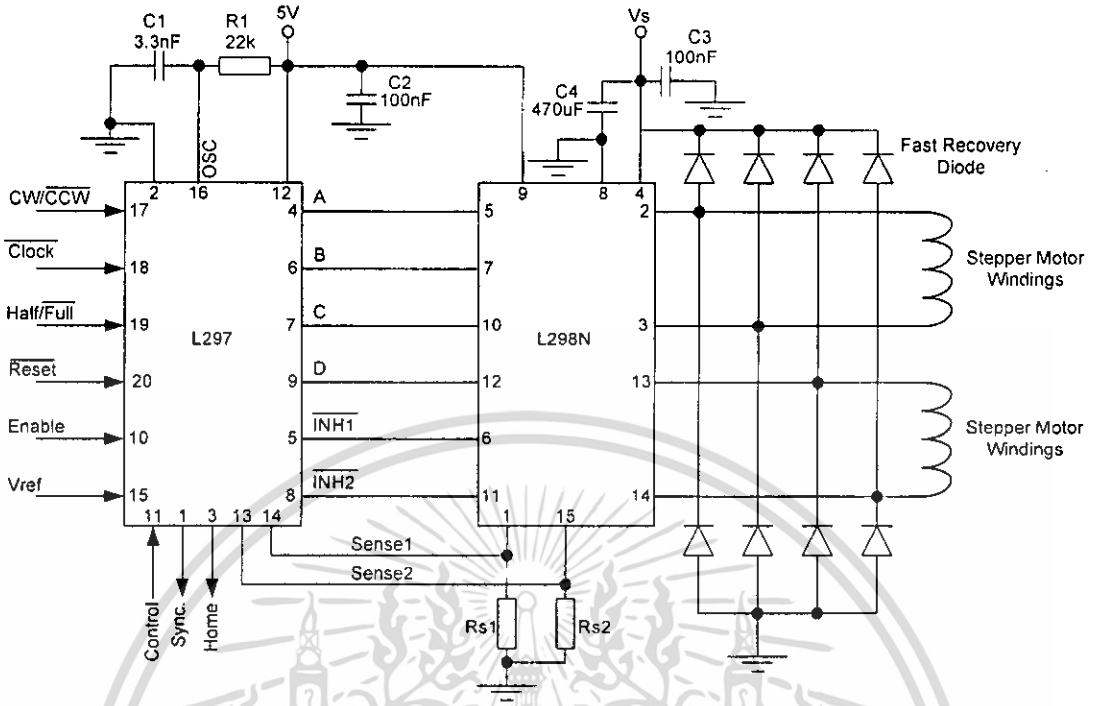
ในการทำโครงงานครั้งนี้ใช้ตัวขับสเต็ปมอเตอร์แบบยูนิโพลาแบบสำเร็จรูปที่มีอยู่เดิมแล้วมาใช้ วงจรขับสเต็ปนี้เราจะป้อนทิศทางและพัลส์ความถี่เพื่อควบคุมความเร็ว แล้วตัวขับสเต็ปมอเตอร์จะทำการป้อนแรงดันเรียงเฟสของสเต็ปมอเตอร์โดยอัตโนมัติ



รูปที่ 3.9 แสดงวงจรขับสเต็ปมอเตอร์แบบยูนิโพลาร์

จากรูปที่ 3.9 จะเห็นได้ว่าเราจะป้อนสัญญาณลอจิก '0' หรือ '1' หลังจากนั้นสัญญาณผ่านวงจรดิจิตอลเพื่อกำหนดทิศทางและป้อนพัลส์เพื่อกำหนดความเร็ว โมดูลขับสเต็ปมอเตอร์แบบยูนิโพลาร์จะทำการจ่ายกระแสตามลำดับเฟสของสเต็ปมอเตอร์ต่อไป

การเคลื่อนที่ในแกนZเราจะใช้สเต็ปมอเตอร์แบบไบโพลาร์ ดังนั้นวงจรขับสเต็ปมอเตอร์จะแตกต่างจากแกนXและแกนY โดยวงจรจะเป็นในลักษณะตามรูปที่ 3.10



รูปที่ 3.10 วงจรขับสเต็ปมอเตอร์แบบไบโพลาร์

จากวงจรในรูปข้างต้นป้อนสัญญาณอินพุทคงที่ขาสัญญาณลักษณะการขับเป็นแบบครึ่งสเต็ป ป้อนลอจิก'1'ที่ขาEnable V_{ref} สามารถปรับค่าได้โดยใช้โพเทนชิโอมิเตอร์ ป้อนลอจิก'1'ที่ขา control เพื่อเลือกลักษณะการchop ขาสัญญาณที่เหลือจะมีเลือกลักษณะการหมุนของมอเตอร์ และขาสัญญาณป้อนพัลส์เพื่อควบคุมความเร็วของมอเตอร์ ซึ่งสัญญาณจะมาจากไมโครคอนโทรลเลอร์

3.2.5 การออกแบบโปรแกรมของไมโครคอนโทรลเลอร์

เมื่อส่งพัลส์ไปยังตัวขับสเต็ปปิ้งมอเตอร์แกน X และแกน Y เพื่อให้วิ่งตามความเร็วที่ต้องการ โดยใช้ความสัมพันธ์ตามสมการที่ (3.1) พร้อมทั้งมีการควบคุมจำนวนพัลส์ที่ส่งออกไป

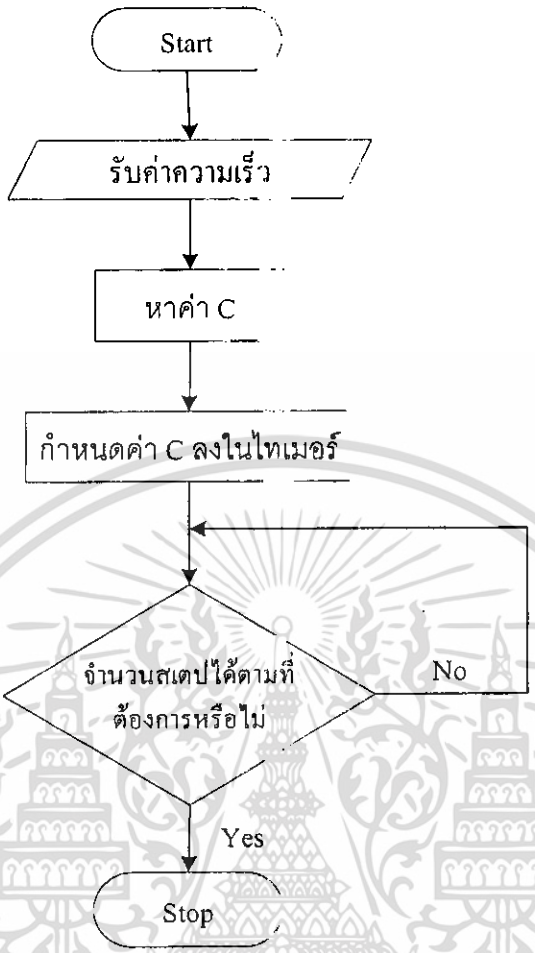
$$C = \frac{f \cdot \Delta x}{V} \qquad (3.1)$$

โดย V = ความเร็ว(mm./sec)

f = ความถี่ของการนับในไทเมอร์ภายในไมโครคอนโทรลเลอร์

Δx = ระยะทางการเคลื่อนที่ 1 สเตป(mm.)

C = จำนวนการนับของไทเมอร์ เพื่อให้มีการส่งพัลซ์ไปยังตัวขับสเตปปิ้งมอเตอร์



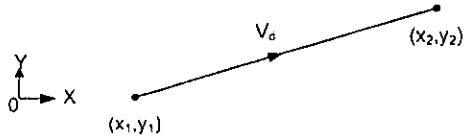
รูปที่ 3.11 โฟลวชาร์ตของโปรแกรม

3.2.6 การออกแบบการวาดเส้นตรงด้วย ไมโครคอนโทรลเลอร์ด้วยวิธี Point-to-Point

การลากเส้นแบบจุดต่อจุด (Point-to-Point) มีการวาดอยู่ 8 แบบ ด้วยกัน ดังนี้

รูปที่ 3.12 การลากเส้นแบบจุดต่อจุดแบบที่ 1

$\Delta x = x_2-x_1$
$\Delta y = 0$
$V_x = V_d$
$V_y = 0$



รูปที่ 3.13 การลากเส้นแบบจุดต่อจุดแบบที่ 2

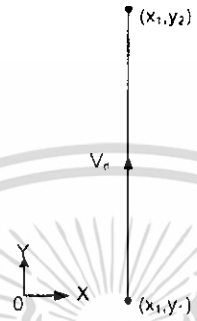
$\Delta x = x_2-x_1$

$$\Delta y = y_2 - y_1$$

$$L = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2}$$

$$V_x = V_d \bullet \left(\frac{x_2 - x_1}{L} \right)$$

$$V_y = V_d \bullet \left(\frac{y_2 - y_1}{L} \right)$$



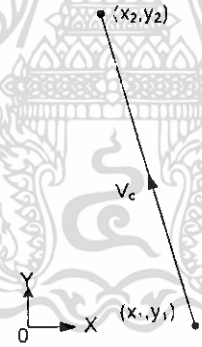
รูปที่ 3.14 การลากเส้นแบบจุดต่อจุดแบบที่ 3

$$\Delta x = 0$$

$$\Delta y = y_2 - y_1$$

$$V_x = 0$$

$$V_y = V_d$$



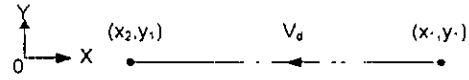
รูปที่ 3.15 การลากเส้นแบบจุดต่อจุดแบบที่ 4

$$\Delta x = x_2 - x_1 \qquad \Delta x \in R^-$$

$$\Delta y = y_2 - y_1$$

$$V_x = -V_d \bullet \left(\frac{x_2 - x_1}{L} \right)$$

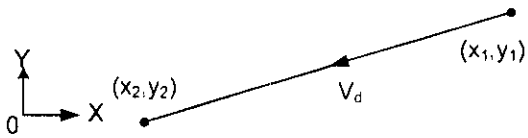
$$V_y = V_d \bullet \left(\frac{y_2 - y_1}{L} \right)$$



รูปที่ 3.16 การลากเส้นแบบจุดต่อจุดแบบที่ 5

$$\Delta x = x_2 - x_1 \qquad \Delta x \in R^-$$

$\Delta y = 0$
$V_x = -V_d$
$V_y = 0$



รูปที่ 3.17 การลากเส้นแบบจุดต่อจุดแบบที่ 6

$\Delta x = x_2 - x_1$
$\Delta y = y_2 - y_1$

$$V_x = -V_d \bullet \left(\frac{x_2 - x_1}{L} \right)$$

$$V_y = -V_d \bullet \left(\frac{y_2 - y_1}{L} \right)$$

รูปที่ 3.18 การลากเส้นแบบจุดต่อจุดแบบที่ 7

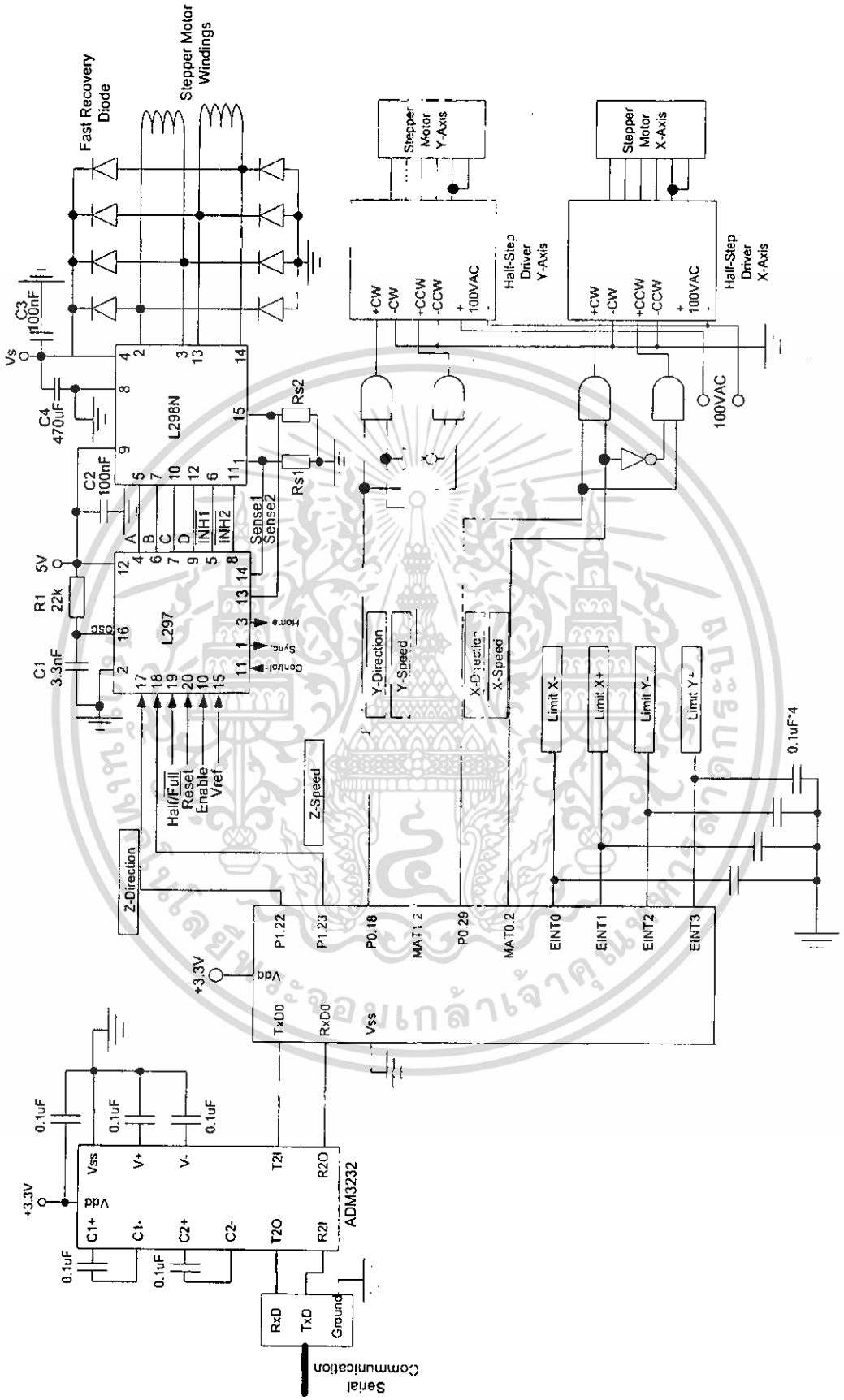
$\Delta x = 0$
$\Delta y = y_2 - y_1$ $\quad \Delta y \in R^-$
$V_x = 0$
$V_y = -V_d$

รูปที่ 3.19 การลากเส้นแบบจุดต่อจุดแบบที่ 8

$\Delta x = x_2 - x_1$
$\Delta y = y_2 - y_1$ $\quad \Delta y \in R^-$

$$V_x = V_d \bullet \left(\frac{x_2 - x_1}{L} \right)$$

$$V_y = -V_d \bullet \left(\frac{y_2 - y_1}{L} \right)$$



รูปที่ 3.20 วงจรการเชื่อมต่อทั้งหมด

บทที่ 4 การทดสอบและผลการทดสอบประสิทธิภาพ

4.1 การทดสอบผลการประมวลผลภาพ

การทดสอบผลการประมวลผลภาพ ทดสอบโดยให้ตัวโปรแกรม Visial Basic6.0 ที่ทำขึ้นมาในโครงงานครั้งนี้ทำงาน แล้วทำการจับภาพของกล้องเวปแคม แล้วนำมาประมวลผลภาพ โดยจะเปรียบเทียบกับภาพดิจิตอลที่มีไฟล์รูปภาพ โดยจะทำการทดสอบการประมวลภาพของการหาภาพขอบ และการหา contour ของภาพ โดยมีผลการทดสอบดังนี้

ผลการทดสอบการหาขอบภาพพบว่า ภาพที่จับได้จากกล้องเวปแคม เมื่อนำไปหาขอบแล้วสามารถหาขอบภาพได้ชัดเจน ซึ่งขึ้นอยู่กับค่าระดับเทรสโฮลที่เรากำหนดขึ้นจากหน้าโปรแกรมที่กำลังทำงานอยู่ ดังรูปที่ 4.3 สำหรับการหาขอบภาพจากภาพดิจิตอลที่มีไฟล์รูปภาพอยู่แล้ว ดังรูปที่ 4.3 จะสามารถหาขอบภาพได้ชัดเจนกว่าภาพจากกล้องเวปแคม

ผลการทดสอบการหา contour ของภาพพบว่า ภาพที่ได้จากกล้องเวปแคม จะมีเส้น contour ต่างจากภาพต้นแบบไป ส่วนในภาพดิจิตอลนั้น contour ของภาพจะเป็นไปตามขอบนอกของภาพต้นแบบ

ก.)ภาพต้นแบบ

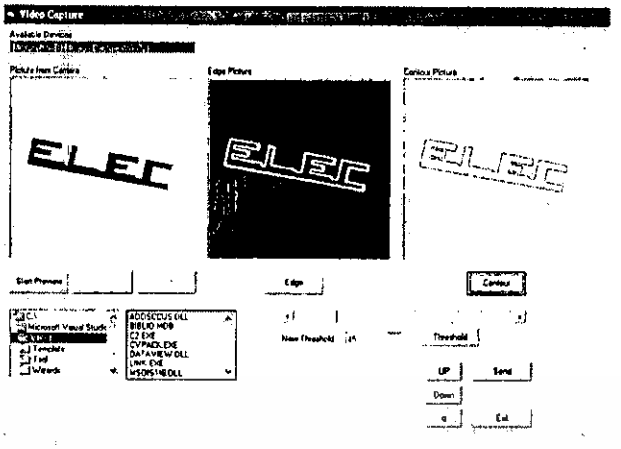
ข.)ภาพระดับเทา

ค.)ภาพขอบ

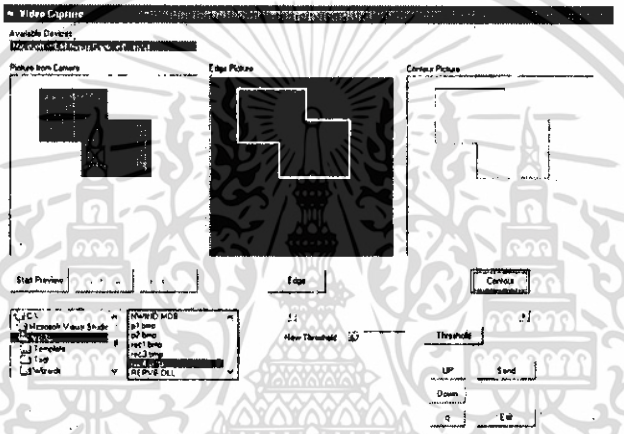
ง.)ภาพไบนารี่

จ.)คอนทัวของภาพ

รูปที่4.1 ภาพที่ได้จากขั้นตอนการประมวลผลภาพ



รูปที่ 4.2 การทดสอบการประมวลผลภาพจากภาพที่ได้จากกล้องเวปแคม



รูปที่ 4.3 การหาขอบภาพจากภาพที่มีไฟล์รูปภาพดิจิตอล

4.2 ทดสอบสเต็ป-ระยะทาง

ทำการทดสอบโดยนับพัลส์ที่ส่งออกไปเพื่อให้มอเตอร์หมุนเคลื่อนที่ไปทางแกน X เป็นระยะ 5 cm. และแกน Y เป็นระยะ 5 cm.

ตารางที่ 4.1 ผลการทดสอบสเต็ป-ระยะทางในแกน X

ครั้งที่	1	2	เฉลี่ย
พัลส์ (ลูก)	504	506	505

ดังนั้น ป้อนพัลส์ 1 ลูก เคลื่อนที่ในแกน X เป็นระยะ 99 μm. หรือประมาณ 0.1 mm.

ตารางที่ 4.2 ผลการทดสอบสเต็ป-ระยะทางในแกน Y

ครั้งที่	1	2	เฉลี่ย
พัลส์ (ลูก)	502	506	504

ดังนั้น ป้อนพัลส์ 1 ลูก เคลื่อนที่ในแกน Y เป็นระยะ 99.206 μm. หรือประมาณ 0.1 mm.

4.3 การทดสอบความเร็วและทิศทาง

กำหนดให้ C_x = จำนวนการนับไทเมอร์เพื่อให้มีส่งพัลส์ไปยังสเต็ปปิ้งมอเตอร์แกน X

C_y = จำนวนการนับไทเมอร์เพื่อให้มีส่งพัลส์ไปยังสเต็ปปิ้งมอเตอร์แกน Y

L = ระยะเส้นทางทั้งหมดที่เคลื่อนที่

$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$V_x = V_{max} \times \frac{dx}{L}$$

$$V_Y = V_{max} \times \frac{dy}{L}$$

จากความสัมพันธ์ $$C = \frac{f \cdot \Delta x}{V}$$

กำหนดให้ f = 100k Hz

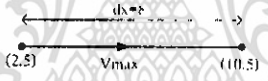
Δx = 99.206 μm

$V_{max} = 49.063 \ mm/s$ ที่ C_x, C_Y = 200 ครั้ง

$C_x = 9920.6 / V_x$

$C_Y = 9920.6 / V_Y$

ทดสอบทิศทางการทำงานโดยการสั่งให้เคลื่อนที่ไปในทิศทางต่าง ๆ ตามรูป



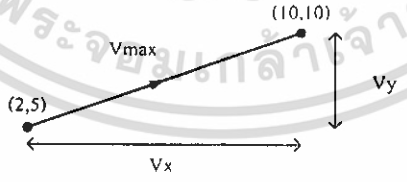
รูปที่ 4.5 การทดสอบในทิศทางที่ 1

<u>คำนวน</u> $L = \sqrt{(10-2)^2 + (0-0)^2} = 8$ mm.

$V_x = V_{max} \rightarrow C_x = 200$

$V_Y = 0 \rightarrow C_Y = 0$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 8 mm



รูปที่ 4.6 การทดสอบในทิศทางที่ 2

<u>คำนวน</u> $L = \sqrt{(10-2)^2 + (10-5)^2} = 9.43$ mm.

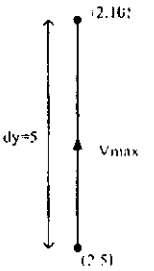
$$V_x = V\,max \cdot \frac{8}{9.43} \rightarrow C_x = \frac{9920.6}{42.063}$$

$$= 42.063 \qquad = 235.85$$

$$V_Y = V\,max \cdot \frac{5}{9.43} \rightarrow C_Y = \frac{9920.6}{26.289}$$

$$= 26.289 \qquad = 377.36$$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 9.5 mm



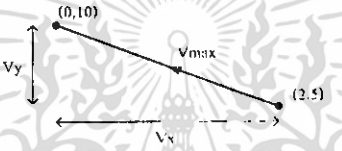
รูปที่ 4.7 การทดสอบในทิศทางที่ 3

<u>คำนวน</u> $L = \sqrt{(0-0)^2 + (10-5)^2} = 5$ mm.

$V_x = 0 \rightarrow C_x = 0$

$V_Y = V_{max} \rightarrow C_Y = 200$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 5 mm



รูปที่ 4.8 การทดสอบในทิศทางที่ 4

<u>คำนวน</u> $L = \sqrt{(0-2)^2 + (10-5)^2} = 5.38$ mm.

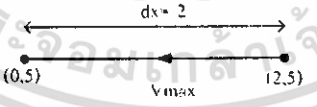
$V_x = -V\max \cdot \frac{2}{5.38} \rightarrow C_x = \frac{9920.6}{18.439}$

$= -18.439 \qquad = 538.02$

$V_Y = V\max \cdot \frac{5}{5.38} \rightarrow C_Y = \frac{9920.6}{46.099}$

$= 46.099 \qquad = 215.20$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 5.4 mm



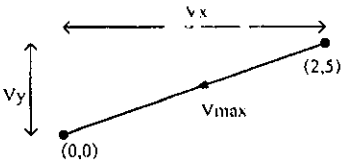
รูปที่ 4.9 การทดสอบในทิศทางที่ 5

<u>คำนวน</u> $L = \sqrt{(2-10)^2 + (0-0)^2} = 2$ mm.

$V_x = -V_{max} \rightarrow C_x = 200$

$V_Y = 0 \rightarrow C_Y = 0$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 2.1 mm



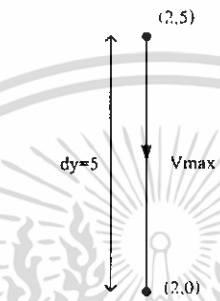
รูปที่ 4.10 การทดสอบในทิศทางที่ 6

<u>คำนวน</u> $L = \sqrt{(2-0)^2 + (5-0)^2} = 5.385$ mm.

$V_x = -V\,max \cdot \frac{2}{5.38} = -18.42$ → $C_x = \frac{9920.6}{18.42} = 538.577$

$V_Y = -V\,max \cdot \frac{5}{5.38} = -46.099$ → $C_Y = \frac{9920.6}{46.099} = 215.20$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 5.4 mm



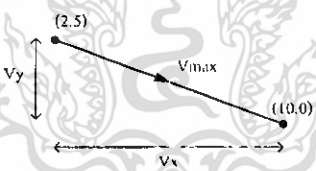
รูปที่ 4.11 การทดสอบในทิศทางที่ 7

<u>คำนวน</u> $L = \sqrt{(0-0)^2 + (5-0)^2} = 5$ mm.

$V_x = 0$ → $C_x = 0$

$V_Y = -V_{max}$ → $C_Y = 200$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 5.1 mm



รูปที่ 4.12 การทดสอบในทิศทางที่ 8

<u>คำนวน</u> $L = \sqrt{(10-2)^2 + (5-0)^2} = 9.43$ mm.

$V_x = V\,max \cdot \frac{8}{9.43} = 42.067$ → $C_x = \frac{9920.6}{42.067} = 538.577$

$V_Y = -V\,max \cdot \frac{5}{9.43} = -26.292$ → $C_Y = \frac{9920.6}{26.292} = 377.32$

<u>ผลการทดสอบ</u> ระยะที่สเต็ปปิ้งมอเตอร์เคลื่อนที่เท่ากับ 9.3 mm

บทที่ 5 สรุปและวิจารณ์ผล

5.1 สรุปผลการทดสอบ

การทดสอบการหาขอบภาพในการประมวลผลภาพ พบว่าภาพที่ได้จากกล้องเวปแคม สามารถหาขอบภาพได้ชัดเจนซึ่งก็ขึ้นอยู่กับค่าระดับเทรสโฮลที่เราเลือก ซึ่งส่งผลให้ contour ของภาพมีลักษณะเส้นที่คมชัดใกล้เคียงกับภาพต้นแบบมากขึ้น

การทดสอบการเคลื่อนที่ของชุดขับเคลื่อนพบว่า เมื่อทำการส่งพัลส์ 1 ลูก ไปยัง Motor Driver of X Axis หรือ Motor Driver of Y Axis จะได้ระยะทาง 99.206 μm หรือประมาณ 0.1 mm. และมุมที่มอเตอร์หมุนคือ $\frac{2\pi}{400}$ rad.

การทดสอบการเคลื่อนที่แบบ point-to-point ซึ่งเป็นการเขียนโปรแกรมคำนวณบนไมโครคอนโทรลเลอร์มีการใช้เทคนิค Linear Interpolation มาใช้ในการกำหนดทิศทางและขนาดของความเร็วที่ใช้ในแต่ละแกนของการเคลื่อนที่ แล้วนำมาคำนวณระยะเวลาในการส่งพัลซ์เพื่อกำหนดค่าลงในไทเมอร์โดยใช้สมการ

$$C = \frac{f \cdot \Delta x}{V}$$

โดย V = ความเร็ว(mm./sec)

f = ความถี่ของการนับในไทเมอร์ภายในไมโครคอนโทรลเลอร์

Δx = ระยะทางการเคลื่อนที่ 1 สเตป(mm.)

C = จำนวนการนับของไทเมอร์ เพื่อให้มีการส่งพัลซ์ไปยังตัวขับสเตปปิ้งมอเตอร์

เมื่อถึงเวลาที่กำหนดไมโครคอนโทรลเลอร์จะส่งพัลส์ไปยังชุดขับสเต็ปมอเตอร์ เพื่อให้มอเตอร์หมุนตามความเร็วที่ต้องการ ซึ่งผลการทดสอบที่ออกมานั้นพบความคลาดเคลื่อนน้อยมาก ซึ่งอุปกรณ์ที่ใช้ในการวัดไม่สามารถวัดได้

5.2 ปัญหาที่พบจากการทำโครงงาน

1. อุปกรณ์ที่ใช้ในการวัดไม่มีความละเอียดเพียงพอ
2. ความคลาดเคลื่อนของอุปกรณ์ที่ใช้งาน
3. ความคลาดเคลื่อนจากระบบทางกลของเครื่องมือที่ใช้ในการทำโครงงาน
4. การตรวจสอบความผิดพลาดในการสื่อสารข้อมูลแบบอนุกรมทำได้ยาก
5. ภาพที่ได้จากกล้องเวปมีค่าความแตกต่างของความเข้มสีไม่เพียงพอ ทำให้หาขอบภาพได้ยาก จึงต้องการเลือกค่าระดับเทรสโฮลของภาพเองเพื่อให้ภาพที่ได้ออกมามีความคมชัดมากขึ้น

หนังสืออ้างอิง

ไชยชาย หินเกิด , "เครื่องกลไฟฟ้ากระแสตรง", สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น) , พิมพ์ครั้งที่2 , 2544

ถาวร อมตกิตติ์ , "การส่งกำลังและการประหยัดพลังงานของมอเตอร์ไฟฟ้ากระแสสลับ" , พิมพ์ครั้งที่ 1 , บริษัท เอ็มแอนดี จำกัด , 2548

ยุทธนา ลีลาศวัฒนากุล *"คู่มือการเขียนโปรแกรมวินโดว์ขั้นสูงด้วย Visual C++"* ,หจก.ไทยเจริญการพิมพ์ , พิมพ์ครั้งที่ 1 , กรุงเทพมหานคร ,สิงหาคม , 2546

โอภาส เอี่ยมสิริวงศ์ , "เครือยข่ายคอมพิวเตอร์และการสื่อสาร", ซีเอ็ดยูเคชั่น , พิมพ์ครั้งที่1 , กรุงเทพมหานคร , 2548

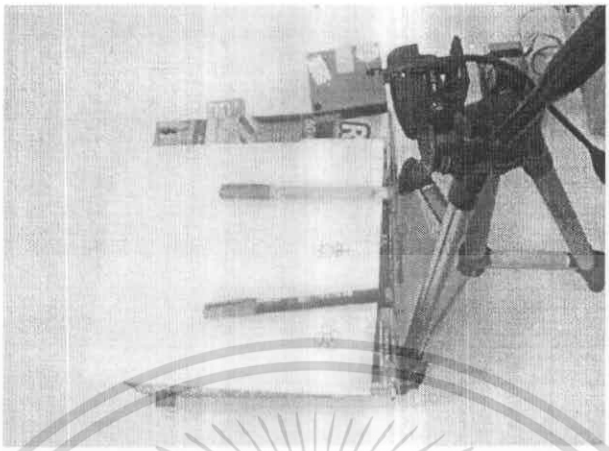
Alasdair McAnDrew, "Introduction to Digital Image Processing with MATLAB",THOMSON course technology,2004

Gyril G.Veinott, Joseph E.Martin: Fraction and subfractional electric motor., Mcgraw-Hill Book Company, 1987.

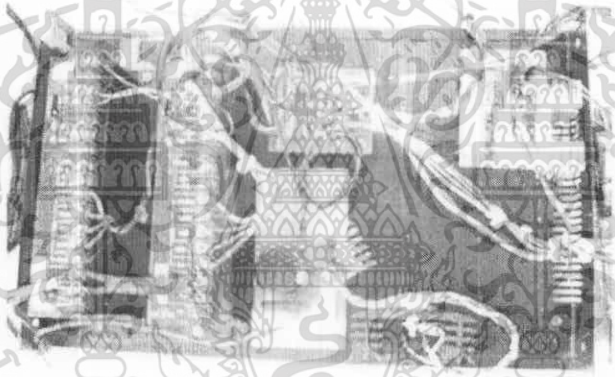
A.Kent Stiffler: Design with microprocessor for mechanical engineers., Mcgraw-Hill Book Company, 1992.

ภาคผนวก

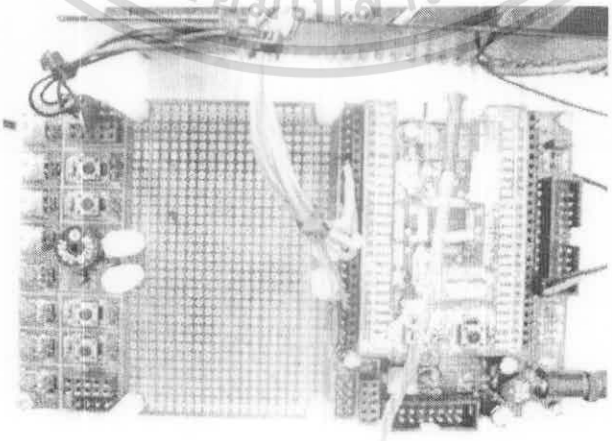
ภาคผนวก ก. รูปภาพส่วนประกอบของเครื่องแกะสลัก



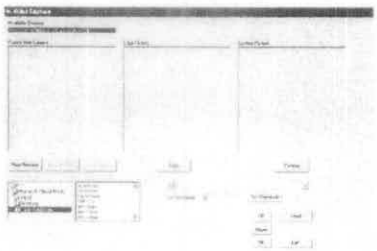
รูปที่ ก.1 ส่วนการจับภาพด้านบนของกล้องเวปแคม



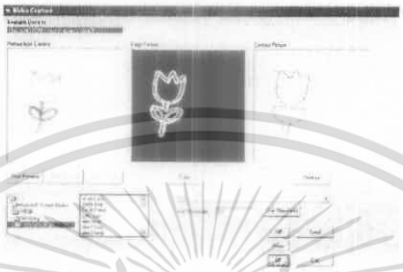
รูปที่ ก.2 ส่วนวงจรขับสเตปปิ้งมอเตอร์



รูปที่ ก.3 ส่วนไมโครคอนโทรลเลอร์

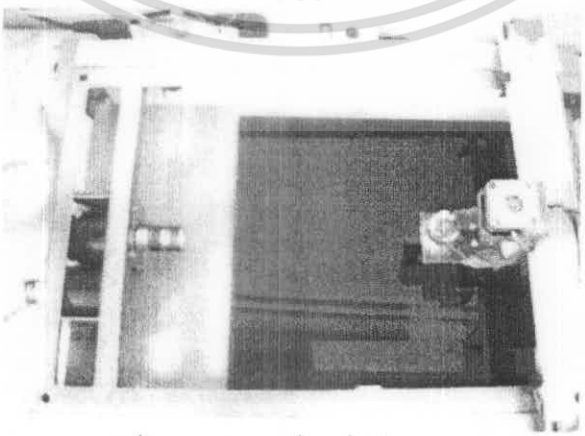


รูปที่ ก.4 ส่วนหน้าโปรแกรม Visual Basic ที่ให้ผู้ใช้สั่งงาน ขณะยังไม่รันโปรแกรม



รูปที่ ก.5 ส่วนหน้าโปรแกรม Visual Basic ที่ให้ผู้ใช้สั่งงาน ขณะรันโปรแกรมยังไม่รันโปรแกรม

รูปที่ ก.6 ส่วนหัวเจาะ



รูปที่ ก.76 ส่วนเครื่องที่ใช้แกะสลัก

ภาคผนวก ข. ARM7 Source code

```
#include <LPC213x.h>
#include <LCD.c>
#include <math.h>
#include <stdio.h>
#include<serial.c>
#define x_direction 0x20000000
#define y_direction 0x00040000

unsigned int
        counterX,limitX,counterY,limitY,MinusX=0.PositiveX=0,MinusY=0,PositiveY=0;
int imax;
char x[2000],y[2000],x_pre,y_pre;

void ExtIntInit         (void);
void OverMinusX                 (void)__irq;
void OverPositiveX      (void)__irq;
void OverMinusY                 (void)__irq;
void OverPositiveY      (void)__irq;
void x_speed_init(void);
void y_speed_init(void);
void x_speed(char dir,int Cx,int cycle);
void y_speed(char dir,int Cy,int cycle);
void PointToPoint(float x1,float y1,float x2,float y2,int Vmax);
void WriteArray(void);
void x_speed_interrupt(void)    __irq;
void y_speed_interrupt(void)    __irq;
void home(void);
void z_up(void);
void z_down(void);
void z_tune(void);
void SetOrigin(void);

void main(void){
char se_in=0;
int aa,j;
//Initial System
x_speed_init();
y_speed_init();
ExtIntInit();
uart0_init(4800);
while(1){
        SetOrigin();
        while(se_in!= 0x80){
                se_in = getchar();}
        imax=getchar();
        aa=getchar();
        imax=imax+(aa*256);
//save point
        for(j=1;j<=imax;j++){
                x[j]=getchar();
                y[j]=getchar();}
        WriteArray();
        z_tune();
```

```
        }
}
/***********************************ExtIntInit*******************************
*************/
void ExtIntInit (void){
PINSEL0              |= 0xA0000000;              //Set pin 0.14toEINT1 ,0.15toEINT2
PINSEL1              |= 0x00000301;              //Set pin 0.16toEINT0 ,0.20toEINT3
EXTMODE              |= 0x0000000F;              //Select to Edge Sensitive Interrupt
EXTPOLAR             |= 0x00000000;              //Select Falling Edge Sensitive
Intertupt
VICVectCntl0  = 0x0000002E;                      //Connect EINT0 to VICVectAddr0
VICVectAddr0 = (unsigned)OverMinusX;                    //Fill IRQ Address to
VICVectAddr0
VICVectCntl1  = 0x0000002F;                      //Connect EINT1 to VICVectAddr1
VICVectAddr1 = (unsigned)OverPositiveX;                 //Fill IRQ Address to
VICVectAddr1
VICVectCntl2  = 0x00000030;                      //Connect EINT1 to VICVectAddr2
VICVectAddr2 = (unsigned)OverMinusY;                    //Fill IRQ Address to
VICVectAddr2
VICVectCntl3  = 0x00000031;                      //Connect EINT1 to VICVectAddr3
VICVectAddr3 = (unsigned)OverPositiveY;                 //Fill IRQ Address to
VICVectAddr3
VICIntEnable  =0x3C000;}                         //Enable
EINT0,EINT1,EINT2,EINT3
void OverMinusX              (void)__irq{               //ExtINT0
T0TCR                        = 0x00000002;
MinusX                       = 1;
limitX                  = counterX;
lcd_clear();
lcd_string("OverMinusX");
EXTINT           |= 0x00000001;                          //Clear the peripheral
interrupt flag
VICVectAddr   = 0x00000000;}                             //Dummy write to signal end
of interrupt
void OverPositiveX     (void)__irq{                      //ExtINT1
T0TCR                        = 0x00000002;
PositiveX               = 1;
limitX                  = counterX;
lcd_clear();
lcd_string("OverPositiveX");
EXTINT           |= 0x00000002;                          //Clear the peripheral
interrupt flag
VICVectAddr   = 0x00000000;}                             //Dummy write to signal end
of interrupt
void OverMinusY              (void)__irq{               //ExtINT2
T1TCR                        = 0x00000002;
MinusY                       = 1;
limitY                  = counterY;
lcd_clear();
lcd_string("OverMinusY");
EXTINT           |= 0x00000004;                          //Clear the peripheral
interrupt flag
VICVectAddr   = 0x00000000;}                             //Dummy write to signal end
of interrupt
```

```
void OverPositiveY (void)__irq{                                    //ExtINT3

T1TCR                   = 0x00000002;
PositiveY               = 1;
limitY                  = counterY;
lcd_clear();
lcd_string("OverPositiveY");
EXTINT                  |= 0x00000008;          //Clear the peripheral interrupt flag
VICVectAddr             = 0x00000000;}          //Dummy write to signal end of interrupt


/*****************************X-Axis*****************************/
void x_speed_init(void){
IODIR0                  |= x_direction;                  //Output port0.29 as X-Direction
PINSEL1                 |= 0x03000000;                   //Match 0.2 as output
VPBDIV                   = 0x00000002;                   //Configure the  VPB divi
T0TCR                    = 0x00000002;                   //Reset counter and prescaler
T0PR                     = 0x12C;                        //Load prescaler to 100 kHz
T0MCR                    = 0x00000003;          //On match reset the counter and generate an
interrupt
T0EMR                    = 0x00000104;          //On match clear MAT2 (MAT2 will set
after use this instruction)
VICVectAddr5  = (unsigned)x_speed_interrupt;            //Set the timer ISR vector address
VICVectCntl5  = 0x00000024;                     //Set channel
VICIntEnable |= 0x00000010;}                    //Enable the interrupt
/*****************************X-Speed*****************************/
void x_speed(char dir,int Cx,int cycle){
counterX        = 0;
limitX          = cycle;
T0TCR           = 0x00000002;                   //Reset counter and prescaler
T0MR0           = Cx;                           //Set the X cycle time
T0MR2           = T0MR0-100;                    //Geterrate pulse
if((dir=='f')&&(Cx!=0)&&(cycle!=0)&&(PositiveX!=1)){
        MinusX                  = 0;
        IOSET0                  |= x_direction;
        T0EMR                   = 0x00000104;                   //Set MAT2 high for
begining of the cycle(MAT2 will set after use this instruction)
        T0TCR                   = 0x00000001;}                          //enable timer
else if((dir=='b')&&(Cx!=0)&&(cycle!=0)&&(MinusX!=1)){
        PositiveX       = 0;
        IOCLR0          |= x_direction;
        T0EMR                   = 0x00000104;                   //Set MAT2 high for
begining of the cycle(MAT2 will set after use this instruction)
        T0TCR                   = 0x00000001;}                          //enable timer
else{
        T0MR0           = 0;
        T0MR2           = 0;
        T0TCR                   = 0x00000002;}

/************************X-Interrupt*********************************/
void x_speed_interrupt(void)    __irq{
counterX++;
if(counterX<limitX){
        T0EMR                   = 0x00000104;                   //Set MAT2 high for
begining of the cycle(MAT2 will set after use this instruction)
```

```
        T0IR            |= 0x00000001;}                    //Clear match 0 interrupt
else {
        T0TCR                   = 0x00000002;             //Reset and Disable Timer0
        T0IR            |= 0x00000001;}                   //Clear match 0 interrupt
VICVectAddr     = 0x00000000;}                            //Dummy write to signal end
of interrupt

/***********************************Y-
Axis*****************************************/
void y_speed_init(void){
IODIR0                  |= y_direction;                   //Output port0.18 as Y-Direction
PINSEL1                 |= 0x0000000C;                    //Match 1.2 as output
VPBDIV                  = 0x00000002;                     //Configure the  VPB divi
T1TCR                   = 0x00000002;                     //Reset counter and prescaler
T1PR                    = 0x12C;                          //Load prescaler to 100 kHz
T1MCR                   = 0x00000003;                     //On match reset the counter and
generate an interrupt
T1EMR                   = 0x00000104;                     //On match clear MAT2 (MAT2 will
set after use this instruction)
VICVectAddr6  = (unsigned)y_speed_interrupt;              //Set the timer ISR vector address
VICVectCntl6  = 0x00000025;                               //Set channel
VICIntEnable |= 0x00000020;}                              //Enable the interrupt
/***********************Y-Speed*********************************/
void y_speed(char dir,int Cy,int cycle){
counterY        = 0;
limitY          = cycle;
T1TCR           = 0x00000002;                   //Reset counter and prescaler
T1MR0           = Cy;                           //Set the Y cycle time
T1MR2           = T1MR0-100;                    //Geterrate pulse
if((dir=='f')&&(Cy!=0)&&(cycle!=0)&&(PositiveY!=1)){
        MinusY          = 0;
        IOCLR0          |= y_direction;
        T1EMR           = 0x00000104;
        T1TCR           = 0x00000001;}                  //enable timer
else if((dir=='b')&&(Cy!=0)&&(cycle!=0)&&(MinusY!=1)){
        PositiveY       = 0;
        IOSET0          |= y_direction;
        T1EMR           = 0x00000104;
        T1TCR           = 0x00000001;}                  //enable timer
else{
        T1MR0           = 0;
        T1MR2           = 0;
        T1TCR           = 0x00000002;}
/***************************Y-Interrupt*********************************/
void y_speed_interrupt(void)    __irq{
counterY++;
if(counterY<limitY){
        T1EMR           = 0x00000104;           /*Set MAT2 high for begining of the
                                                cycle(MAT2 will set after use this
                                        instruction)*/
        T1IR            |= 0x00000001;}                 //Clear match 0 interrupt
else {
        T1TCR           = 0x00000002;                   //Reset and Disable Timer0
        T1IR            |= 0x00000001;}                 //Clear match 0 interrupt
```

```
VICVectAddr    = 0x00000000;}                    //Dummy write to signal end of
interrupt
/************************Point-To-Point Function************************/
void PointToPoint(float x1,float y1,float x2,float y2,int Vmax){
float Delta_x, Delta_y;
int Cx, Cy;
float Vx,Vy;
double path;
Delta_x = fabs(x2-x1);
Delta_y = fabs(y2-y1);
path   = sqrt((Delta_x*Delta_x)+(Delta_y*Delta_y));
Vx              = Vmax*(Delta_x/path);
Vy              = Vmax*(Delta_y/path);
Cx              =       100000*0.1/Vx;
Cy              =       100000*0.1/Vy;
if(x2>x1)
        x_speed('f',Cx,(Delta_x*5));

else if (x2<x1)
        x_speed('b',Cx,(Delta_x*5));
else;
if(y2>y1)
        y_speed('f',Cy,Delta_y*5);
else if (y2<y1)
        y_speed('b',Cy,Delta_y*5);
else;
while(counterX<limitX);
while(counterY<limitY);}
/************************\***App Func**********************************/
void home(void){
PINSEL0             &= 0x0FFFFFFF;          //Set pin 0.14toGPIO ,0.15toGPIO
PINSEL1             &= 0xFFFFFC0C;          //Set pin 0.16toGPIO ,0.20toGPIO
IODIR0              &= 0xFFEE3FFF;          //X-Axis
if((IOPIN0&0x00014000)==0x00004000){        //X-
        PINSEL0                 |= 0x20000000; //Set pin 0.14toEINT1(PositiveX)

        PINSEL1             |= 0x00000001;  //Set pin 0.16toEINT0(MinusX)

        MinusX                      =1;}
else{                                       //X+,X is Floating
        PINSEL0             |= 0x20000000;  //Set pin 0.14toEINT1(PositiveX)

        PINSEL1             |= 0x00000001;  //Set pin 0.16toEINT0(MinusX)
        x_speed('b',200,700000);}
//Y-Axis
if((IOPIN0&0x00108000)==0x00100000){        //Y-
        PINSEL0             |= 0x80000000;  //Set pin 0.15toEINT2(MinusY)
        PINSEL1             |= 0x00000300;  //Set pin 0.20toEINT3(PositiveY)

        MinusY                      =1;}
else{
        //Y+,Y is Floating
        PINSEL0             |= 0x80000000;  //Set pin 0.15toEINT2(MinusY)
        PINSEL1             |= 0x00000300; //Set pin 0.20toEINT3(PositiveY)
```

```
        y_speed('b',200,700000);}
        while(counterX<limitX);
        while(counterY<limitY);}

void z_down(void){
unsigned int i;
IODIR1          |=      0x00C00000;
IOSET1          |=      0x00400000;
IOCLR1          |=      0x00800000;
for(i=0;i<195;i++){
        IOSET1          |=      0x00800000;
        delay_us(5000);
        IOCLR1          |=      0x00800000;
        delay_us(5000);
        }
}
void z_up(void){
unsigned int i;
IODIR1          |=      0x00C00000;
IOCLR1          |=      0x00400000;
IOCLR1          |=      0x00800000;
for(i=0;i<195;i++){
        IOSET1          |=      0x00800000;
        delay_us(5000);
        IOCLR1          |=      0x00800000;
        delay_us(5000);
        }
}
void z_tune(void){
char z_com;
z_com=getchar();
while(z_com != 'q'){
        printf("%c\n",z_com);
        if(z_com =='u')         z_up();

        else if(z_com =='d')    z_down();
        z_com=getchar();
        }
        printf("%c\n",z_com); }

void SetOrigin(void){
home();
delay_us(50000);
PointToPoint(0,0, 0,50,50);
delay_us(50000);
PointToPoint(0,0, 70,0,50);
delay_us(50000);
x_pre = 0;
y_pre = 0;}

void WriteArray(){
int i;
for(i=1;i<=imax;i++){
        PointToPoint(x_pre,y_pre,x[i],y[i],4);
        if (i==1)
```

```
                z_tune();
        x_pre = x[i];
        y_pre = y[i];
        }
}

/*********************โปรแกรมส่วน Serial*****************************/
#include<LPC213x.h>
#include<stdio.h>
#define CR 0x0D

void uart0_init(unsigned int baudrate){
unsigned short u0dl;
u0dl            = 30000000/(16*baudrate);
PINSEL0           |= 0x00000005;
U0LCR           = 0x83;
U0DLL           = u0dl&0xFF;
U0DLM                   = (u0dl>>8);
U0LCR   &= 0x7F;}

int putchar(int ch){
if(ch=='\n'){
        while(!(U0LSR&0x20));
        U0THR = CR;}
while(!(U0LSR&0x20));
return(U0THR=ch);}

int getchar(void){
while(!(U0LSR&0x01));
return(U0RBR);}
/*************************โปรแกรมส่วน LCD*****************************/
#include<LPC213x.h>
#define  LCD_D4     0x00010000                  // P1.16
#define  LCD_D5     0x00020000                  // P1.17
#define  LCD_D6     0x00040000                  // P1.18
#define  LCD_D7     0x00080000                  // P1.19
#define  LCD_RS     0x00100000                  // P1.20
#define  LCD_EN     0x00200000                  // P1.21
#define  lcd_rs_set() IOSET1 |= LCD_RS          // RS = 1 (Select Instruction)
#define  lcd_rs_clr() IOCLR1 |= LCD_RS          // RS = 0 (Select Data)
#define  lcd_en_set() IOSET1 |= LCD_EN          // EN = 1 (Enable)
#define  lcd_en_clr() IOCLR1 |= LCD_EN          // EN = 0 (Disable)
#define  lcd_clear()        lcd_com(0x01)       // Clear Display
#define  lcd_cursor_home()   lcd_com(0x02)      // Set Cursor = 0
#define  lcd_display_on()    lcd_com(0x0E)      // LCD Display Enable
#define  lcd_display_off()   lcd_com(0x08)      // LCD Display Disable
#define  lcd_cursor_blink()  lcd_com(0x0F)      // Set Cursor = Blink
#define  lcd_cursor_on()     lcd_com(0x0E)      // Enable LCD Cursor
#define  lcd_cursor_off()    lcd_com(0x0C)      // Disable LCD Cursor
#define  lcd_cursor_left()   lcd_com(0x10)      // Shift Left Cursor
#define  lcd_cursor_right()  lcd_com(0x14)      // Shift Right Cursor
#define  lcd_display_sleft()  lcd_com(0x18)     // Shift Left Display
#define  lcd_display_sright() lcd_com(0x1C)     // Shift Right Display
//Function Prototype
```

```
void lcd_delay(void);
void delay_us(unsigned int time);
void nibble(void);
void lcd_com(unsigned int com);
void lcd_init(void);
void lcd_write(char c);
void lcd_string(char str[40]);
/**********************Delay************************************/
void delay_us(unsigned int time){
unsigned int count1;
for(count1=0;count1<=(time*5);count1++);}
/********************LCD_Delay**********************************/
void lcd_delay(void){
unsigned delay;
for(delay=0;delay<=4000;delay++);}                    //delay 5ms
/********************Nibble*************************************/
void nibble(void){
lcd_delay();                                          //Delay 5ms
lcd_en_set();                                         //Set EN bit
lcd_delay();                                          //Delay 5ms
lcd_en_clr();                                         //Clear EN bit
lcd_delay();}                                         //Delay 5ms
/******************LCD_Command**********************************/
void lcd_com(unsigned int com){
unsigned int out;
lcd_rs_clr();                                         //Clear RS bit
out=com;
IOSET1=(0xF0000&(out<<12));                           //send 4 high bits
out=com;
IOCLR1=(0xF0000&((~out)<<12));
nibble();
out=com;
IOSET1=(0xF0000&(out<<16));                           //send 4 low bits
out=com;
IOCLR1=(0xF0000&((~out)<<16));
nibble();}
/************************LCD_Write*****************************/
void lcd_write(char c){
unsigned int out;
lcd_rs_set();                                         //SET RS bit
out=c;
IOSET1=(0xF0000&(out<<12));                           //send 4 high bits
out=c;
IOCLR1=(0xF0000&((~out)<<12));
nibble();
out=c;
IOSET1=(0xF0000&(out<<16));                           //send 4 low bits
out=c;
IOCLR1=(0xF0000&((~out)<<16));
nibble();}
/**********************LCD_String*********************************/
void lcd_string(char str[40]){
int i;
i=0;
while(str[i]!=0){
```

```
        lcd_write(str[i]);
        i++;
        }
}
/*********************LCD_Init*****************************************/
void lcd_init(void){
IODIR1|=0x3F0000;
IOCLR1|=0x3F0000;
lcd_delay();                                    //Delay 5ms
lcd_delay();                                    //Delay 5ms
lcd_delay();                                    //Delay 5ms
//Sent Data 0x03 for excite 2 times
lcd_com(0x33);
//Sent Data 0x03 for excite 1 time and send 0x02 1 time
lcd_com(0x32);
lcd_com(0x28);                                          //Select display line and
front
lcd_com(0x08);                                  //Display off
lcd_com(0x01);                                  //Display clear
lcd_com(0x02);                                  //Entry mode
lcd_com(0x0C); }                                        //Open display,open
cursor,blink cursor
```

ภาคผนวก ข. Visual Basic Source code

```
Const WM_CAP As Integer = &H400
Const WM_CAP_DRIVER_CONNECT As Long = WM_CAP + 10  'normal=10
Const WM_CAP_DRIVER_DISCONNECT As Long = WM_CAP + 11
Const WM_CAP_EDIT_COPY As Long = WM_CAP + 30   'normal set =30
Const WM_CAP_SET_PREVIEW As Long = WM_CAP + 50  'normal set=50
Const WM_CAP_SET_PREVIEWRATE As Long = WM_CAP + 52
Const WM_CAP_SET_SCALE As Long = WM_CAP + 51 'normal=53 to fit for window ,
normal set=51
Const WS_CHILD As Long = &H40000000
Const WS_VISIBLE As Long = &H10000000
Const SWP_NOMOVE As Long = &H2
Const SWP_NOSIZE As Integer = 1
Const SWP_NOZORDER As Integer = &H4
Const HWND_BOTTOM As Integer = 1
Dim iDevice As Long
Dim hHwnd As Long

Public H As Byte
Public W As Byte
Public pixel As Variant
Public Threshold As Byte
Public index_x As Integer
Public index_y As Integer
Private pos_array_x(5000) As Byte
Private pos_array_y(5000) As Byte
Public aa As String
Private bi(5000, 5000) As Byte
Public val

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
   (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, _
   lParam As Any) As Long
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, _
   ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, _
   ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
Private Declare Function DestroyWindow Lib "user32" (ByVal hndw As Long) As Boolean
Private Declare Function capCreateCaptureWindowA Lib "avicap32.dll" _
   (ByVal lpszWindowName As String, ByVal dwStyle As Long, _
   ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, _
   ByVal nHeight As Integer, ByVal hWndParent As Long, _
   ByVal nID As Long) As Long
Private Declare Function capGetDriverDescriptionA Lib "avicap32.dll" (ByVal wDriver As
Long, _
   ByVal lpszName As String, ByVal cbName As Long, ByVal lpszVer As String, _
   ByVal cbVer As Long) As Boolean
```

```
Sub delay()
Dim ioi, joj As Integer
        For ioi = 0 To 100
                For joj = 0 To 300
                Next
                Next
End Sub
```

```
Private Sub cmd_edge_Click()
   Dim R(1000, 1000) As Integer      'Array of R color
   Dim G(1000, 1000) As Integer      'Array of G color
   Dim b(1000, 1000) As Integer      'Array of B color
   Dim max As Integer
   Dim min As Integer
   Dim i, j, m, n, x, y As Integer
   'Dim Threshold As Integer
   Dim Gray(1000, 1000) As Integer
   Dim Px(1000, 1000) As Integer
   Dim Py(1000, 1000) As Integer
   Dim Pxy(1000, 1000) As Double
   Dim H, W As Integer

   H = Picture2.ScaleHeight
   W = Picture2.ScaleWidth
'********  Read & mem Gray value  **************************
   For i = 0 To W - 1
      For j = 0 To H - 1
         pixel = picCapture.Point(i, j)                    'read level colour form picture1
         R(i, j) = pixel Mod 256                           'define RED level
         G(i, j) = (pixel \ 256) Mod 256                   'define Green level
         b(i, j) = pixel \ 65536                      'define Blue level
         Gray(i, j) = (R(i, j) + G(i, j) + b(i, j)) \ 3  'define gray level
      Next
   Next
'**************  Find Pxy(1000,1000)  *******************************
   For i = 1 To W - 2
      For j = 1 To H - 2
         Px(i, j) = Abs((((Gray(i - 1, j - 1) * (-1)) + (Gray(i + 1, j - 1) * (1)) + _
                    (Gray(i - 1, j) * (-2)) + (Gray(i + 1, j) * (2)) + (Gray(i - 1, j + 1) * _
                    (-1)) + (Gray(i + 1, j + 1) * (1)))))
         Py(i, j) = Abs((((Gray(i - 1, j - 1) * (-1)) + (Gray(i, j - 1) * (-2)) +_
                      (Gray(i + 1, j - 1) * (-1)) +(Gray(i - 1, j + 1) * (1)) + _
                      (Gray(i, j + 1) * (2)) + (Gray(i + 1, j + 1) * (1)))))
         Pxy(i, j) = Px(i, j) + Py(i, j)
      Next
   Next
'***************find binary Picture*********************************
   For i = 1 To W - 2
     For j = 1 To H - 2
        If Pxy(i, j) >= Threshold Then
           bi(i, j) = 255
        ElseIf Pxy(i, j) < Threshold Then
           bi(i, j) = 0
        End If
        Picture2.PSet (i, j), RGB(bi(i, j), bi(i, j), bi(i, j))
```

```
    Next
  Next
End Sub
```

```
Private Sub cmdcon_Click()
  Dim i, j, m, n As Integer
  Dim x, y As Integer  ' for transfer to count index
  Dim scan As Byte
  Dim pos_yy As Byte
  Dim pos_xx As Byte
  Dim rr_i(9), rr_j(9) As Byte
  Dim state As Integer
  Dim pos_x As Byte
  Dim pos_y As Byte
  H = Picture2.ScaleHeight
  W = Picture2.ScaleWidth
'***********find contour following*****************************
  index_x = count_pos_x(x)
  index_y = count_pos_y(y)
For i = 3 To W - 4
  For j = 3 To H - 4
    Do While bi(i, j) = 255
    'for first position
    If index_x = 1 Then
      pos_array_x(1) = i
      pos_array_y(1) = j
    End If

    rr_i(1) = i - 1: rr_j(1) = j - 1
    rr_i(2) = i: rr_j(2) = j - 1
    rr_i(3) = i + 1: rr_j(3) = j - 1
    rr_i(4) = i + 1: rr_j(4) = j
    rr_i(5) = i + 1: rr_j(5) = j + 1
    rr_i(6) = i: rr_j(6) = j + 1
    rr_i(7) = i - 1: rr_j(7) = j + 1
    rr_i(8) = i - 1: rr_j(8) = j
    rr_i(9) = i - 1: rr_j(9) = j - 1   'same rr_i(1)and rr_j(1)

    If bi(rr_i(1), rr_j(1)) = 0 And bi(rr_i(2), rr_j(2))   255 Then
      scan = 1
      pos_x = i: pos_y = j - 1
    ElseIf bi(rr_i(2), rr_j(2)) = 0 And bi(rr_i(3), rr_j(3)) = 255 Then
      scan = 2
      pos_x = i + 1: pos_y = j - 1
    ElseIf bi(rr_i(3), rr_j(3)) = 0 And bi(rr_i(4), rr_j(4)) = 255 Then
      scan = 3
      pos_x = i + 1: pos_y = j
    ElseIf bi(rr_i(4), rr_j(4)) = 0 And bi(rr_i(5), rr_j(5)) = 255 Then
      scan = 4
      pos_x = i + 1: pos_y = j + 1
    ElseIf bi(rr_i(5), rr_j(5)) = 0 And bi(rr_i(6), rr_j(6)) = 255 Then
      scan = 5
     pos_x = i: pos_y = j + 1
    ElseIf bi(rr_i(6), rr_j(6)) = 0 And bi(rr_i(7), rr_j(7)) = 255 Then
      scan = 6
```

```
    pos_x = i - 1: pos_y = j + 1
  ElseIf bi(rr_i(7), rr_j(7)) = 0 And bi(rr_i(8), rr_j(8)) = 255 Then
    scan = 7
    pos_x = i - 1: pos_y = j
  ElseIf bi(rr_i(8), rr_j(8)) = 0 And bi(rr_i(1), rr_j(1)) = 255 Then
    scan = 8
    pos_x = i - 1: pos_y = j - 1
  End If

  state = 666

  If index_x >= 2 Then
  'check that is the first position
    If (pos_array_x(1) = pos_x) And (pos_array_y(1) = pos_y) Then
      state = 111
      MsgBox "Found First position", vbOKOnly
      Exit Do
    End If
  'check that is not have next position
    For m = 2 To index_x
      If pos_array_x(m) = pos_x And pos_array_y(m) = pos_y Then
        If scan = 8 Then
          scan = 0
        End If

        scan = scan + 1

        For n = scan To 8
          If (bi(rr_i(n), rr_j(n)) = 0) And (bi(rr_i(n + 1), rr_j(n + 1)) = 255) Then

            Select Case n
              Case 1
                pos_xx = i: pos_yy = j - 1
              Case 2
                pos_xx = i + 1: pos_yy = j - 1
              Case 3
                pos_xx = i + 1: pos_yy = j
              Case 4
                pos_xx = i + 1: pos_yy = j + 1
              Case 5
                pos_xx = i: pos_yy = j + 1
              Case 6
                pos_xx = i - 1: pos_yy = j + 1
              Case 7
                pos_xx = i - 1: pos_yy = j
              Case 8
                pos_xx = i - 1: pos_yy = j - 1
              End Select
            'check that not have next point
            If (pos_xx = emtry) And (pos_yy = emtry) Then
              state = 555
            End If

            'check again that not same in not same in array
            For mm = 2 To index_x
```

```
                    If pos_array_x(mm) = pos_xx And pos_array_y(mm) = _
                        pos_yy Then
                      'same position in array
                      'MsgBox "not found next pos", vbOKOnly
                      state = 999

                      'clear array
                      For nn = 1 To index_x
                        pos_array_x(nn) = 0
                        pos_array_y(nn) = 0
                      Next
                      index_x = 1
                      index_y = 1
                      Exit Do
                    Else
                      pos_x = pos_xx
                      pos_y = pos_yy
                    End If
                  Exit For
                  Next
                Exit For
                End If
              Next
            End If
          Next
        End If

        If state = 666 Then
        'call function for count index in pos_array_x and pos_array_y
        index_x = count_pos_x(index_x)
        index_y = count_pos_y(index_y)

        'put pos_x in pos_array_x  &  put pox_y in pos_array_y
        pos_array_x(index_x) = pos_x
        pos_array_y(index_y) = pos_y
        'change next new pixel
        i = pos_x
        j = pos_y
        End If
      Loop
      If state = 111 Then
        i = W
        j = H
      End If
      Next
    Next
      If state = 111 Then
        For c = 1 To index_x
          'Debug.Print pos_array_x(c)
          'MsgBox "plot", vbOKonly
          Picture3.PSet ((pos_array_x(c)), (pos_array_y(c))), RGB(255, 0, 0)
        Next
      ElseIf state = 999 Then
        MsgBox "not found Contour", vbOKOnly
      End If
```

```
End Sub
_____
Private Sub cmd_Send_Click()
Dim aaa As Integer
Dim bb As Byte
Dim rr_i(9), rr_j(9) As Byte
Dim iop, pos_x, pos_y As Byte
Dim i, j, m, n, x, y As Integer
'**************check************************
For dd = 1 To index_x
   Debug.Print dd
   Debug.Print pos_array_x(dd)
   Debug.Print pos_array_y(dd)
Next
'*********'**************send index_x*****************************
   MSCSe.Output = Chr(128)       ' send REQ=128=&H80
   Debug.Print val
   aaa = index_x And &HFF
   bb = aaa
     Debug.Print bb
   MSCSe.Output = Chr(bb)
   delay
   aaa = index_x And &HFF00
   aaa = index_x \ 256
   bb = aaa
     Debug.Print bb
   MSCSe.Output = Chr(bb)
   delay
'**********send position*****************************************
         For ii = 1 To index_x
            MSCSe.Output = Chr(pos_array_x(ii))
            delay
            'val = MSCSe.Input
            MSCSe.Output = Chr(pos_array_y(ii))
            delay
         Next
End Sub
Private Sub cmdd_Click()
   MSCSe.Output = "d"
End Sub
Private Sub cmdExit_Click()
   Unload Me
End Sub
Private Sub Cmdq_Click()
   MSCSe.Output = "q"
End Sub
_____
Private Sub cmdSave_Click()
   Dim bm As Image
   SendMessage hHwnd, WM_CAP_EDIT_COPY, 0, 0
   ClosePreviewWindow
   picCapture.Picture = Clipboard.GetData
   CommonDialog1.CancelError = True
   CommonDialog1.FileName = "Webcam1"
   CommonDialog1.Filter = "Bitmap |*.bmp"
```

```
    On Error GoTo NoSave
    CommonDialog1.ShowSave
    SavePicture picCapture.Image, CommonDialog1.FileName
NoSave:
    cmdStop.Enabled = False
    cmdSave.Enabled = False
    cmdStart.Enabled = True
End Sub

---

Private Sub cmdStart_Click()
    iDevice = lstDevices.ListIndex
    OpenPreviewWindow
End Sub

---

Private Sub cmdStop_Click()
    ClosePreviewWindow
    cmdStop.Enabled = False
    cmdSave.Enabled = False
    cmdStart.Enabled = True
End Sub

---

Private Sub cmdThr_Click()
    Threshold = txtThr.Text
End Sub

---

Private Sub cmdU_Click()
    MSCSe.Output = "u"
End Sub

---

Private Sub Command1_Click()
    Picture2.PSet (10, 20), RGB(255, 0, 0)
End Sub

---

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub

---

Private Sub File1_Click()
    picCapture.Picture.= LoadPicture(File1.Path & "\" & File1.FileName)
End Sub

---

Private Sub Form_Load()
    MSCSe.CommPort = 1
    MSCSe.Settings = "4800,n,8,1"
    MSCSe.PortOpen = True

    LoadDeviceList
    If lstDevices.ListCount > 0 Then
        lstDevices.Selected(0) = True
    Else
    cmdStart.Enabled = False
        lstDevices.AddItem ("ไม่พบกล้อง WebCam")
    End If
    cmdStop.Enabled = False
```

```
   cmdSave.Enabled = False
End Sub

---

Private Sub LoadDeviceList()
   Dim strName As String
   Dim strVer As String
   Dim iReturn As Boolean
   Dim x As Long
   x = 0
   strName = Space(100)
   strVer = Space(100)
   Do
      iReturn = capGetDriverDescriptionA(x, strName, 100, strVer, 100)
      If iReturn Then lstDevices.AddItem Trim$(strName)
      x = x + 1
   Loop Until iReturn = False
End Sub

---

Private Sub OpenPreviewWindow()
   ' Open Preview window in picturebox
   hHwnd = capCreateCaptureWindowA(iDevice, WS_VISIBLE Or WS_CHILD, 0, 0, 640 _
      480, picCapture.hwnd, 0)
   ' Connect to device
   If SendMessage(hHwnd, WM_CAP_DRIVER_CONNECT, iDevice, 0) Then
      'Set the preview scale
      SendMessage hHwnd, WM_CAP_SET_SCALE, True, 0
      'Set the preview rate in milliseconds
      SendMessage hHwnd, WM_CAP_SET_PREVIEWRATE, 66, 0
      'Start previewing the image from the camera
      SendMessage hHwnd, WM_CAP_SET_PREVIEW, True, 0
      ' Resize window to fit in picturebox
      SetWindowPos hHwnd, HWND_BOTTOM, 0, 0, picCapture.ScaleWidth,
picCapture.ScaleHeight, _
            SWP_NOMOVE Or SWP_NOZORDER
      cmdSave.Enabled = True
      cmdStop.Enabled = True
      cmdStart.Enabled = False
   Else
      ' Error connecting to device close window
      DestroyWindow hHwnd
      cmdSave.Enabled = False
   End If
 End Sub

---

Private Sub ClosePreviewWindow()
   ' Disconnect from device
   SendMessage hHwnd, WM_CAP_DRIVER_DISCONNECT, iDevice, 0
   ' close window
   DestroyWindow hHwnd
End Sub

---

Private Sub Form_Unload(Cancel As Integer)
   If cmdStop.Enabled Then
      ClosePreviewWindow
   End If
```

```
End Sub
```

```
Private Function count_pos_x(ByVal a As Integer) As Integer
   a = a + 1
   count_pos_x = a
End Function
```

```
Private Function count_pos_y(ByVal b As Integer) As Integer
   b = b + 1
   count_pos_y = b
End Function
```

```
Private Sub HScroll1_Change()
   txtThr.Text = HScroll1.Value
End Sub
```