เครื่องเจาะแผ่นพลาสติกสำหรับทำแบบทอพรม

PLASTIC SHEET PUNCHING MACHINE

โดย

นายจาตุพล อนันตกูล

นายธงไชย ปรัชญบริรักษ์

นางสาวพิมพ์ศิริ ศิริข่าง

อาจารย์ที่ปรึกษา

ดร.ณัฐวุฒิ เดไปวา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องเจาะแผ่นพลาสติกสำหรับทำแบบทอพรม

PLASTIC SHEET PUNCHING MACHINE

ผู้จัดทำ

1. นายจาตุพล อนันตกุล รหัสประจำตัว 44010658
2. นายธงไชย ปรัชญบริรักษ์ รหัสประจำตัว 44010719
3. นางสาวพิมพ์ศิริ ศิริข่วง รหัสประจำตัว 44010780

อาจารย์ที่ปรึกษา

(ดร. ณัฐวุฒิ เดไปวา)

เครื่องเจาะแผ่นพลาสติกสำหรับสร้างแบบทอพรม

นายจาตุพล	อนันตกุล	รหัสนักศึกษา 44010600
นายธงไชย	ปรัชญบริรักษ์	รหัสนักศึกษา 44010719
นางสาวพิมศิริ	ศิริข่วง	รหัสนักศึกษา 44010780
ดร.ณัฐวุฒิ	เดไปวา	อาจารย์ที่ปรึกษา

บทคัดย่อ

โครงงานนี้เป็นการออกแบบและสร้างเครื่องเจาะแผ่นพลาสติก เพื่อนำไปใช้เป็นแบบในการทอพรม โดยอาศัยหลักการเคลื่อนที่แบบ 2 แกน ในเครื่องพริ้นเตอร์ มาเป็นพื้นฐานในการออกแบบ ส่วนประกอบที่สำคัญของเครื่องเจาะแผ่นพลาสติก ประกอบด้วย 3 ส่วน คือ 1. ระบบการเคลื่อนที่แบบ 2 แกน ที่ใช้สเต็ปปิ้งมอเตอร์ ในการขับเคลื่อน และใช้สายพานในการส่งกำลัง 2. ชุดไมโครคอลโทรลเลอร์ ที่ใช้ในการควบคุมการทำงานของสเต็ปปิ้ง มอเตอร์ และตัวเจาะแผ่นพลาสติก 3. โปรแกรมควบคุมการทำงานของเครื่อง โดยเครื่องเจาะแผ่นพลาสติกจะรับสัญญาณพัลส์จากคอมพิวเตอร์ เพื่อสั่งให้สเต็ปปิ้งมอเตอร์หมุน ซึ่งจะใช้ภาษาวิชวลเบสิคในการสร้างโปรแกรมควบคุม การทำงานของเครื่องจะเริ่มเมื่อ โปรแกรมที่สร้างจากภาษาวิชวลเบสิครับภาพที่เป็นลายเส้น จากนั้นคำนวณหาพิกัดที่ต้องการเจาะ แล้วจึงส่งสัญญาณให้ชุดไมโครคอนโทรลเลอร์สั่งงานเครื่องเจาะแผ่นพลาสติก ให้เจาะแผ่นพลาสติกตามพิกัดที่คำนวณได้จนครบทุกพิกัดตามรูปภาพ แผ่นพลาสติกที่เจาะเสร็จแล้วจะถูกนำไปใช้เป็นแม่แบบในการทอพรม โดยนำไปทาบลงบนผ้าที่จะทอเป็นพรม จากนั้นจึงทาสีลงไป เมื่อนำพลาสติกออกแล้วจะได้แนวเส้นในการทอพรม

Plastic Sheet Punching Machine

Mr. Jatupol	Anantakul	Student ID 44010658
Mr. Thongchai	Pratchayaborirak	Student ID 44010719
Ms. Pimsiri	Sirikhuang	Student ID 44010780
Dr. Nuttawut	Depaiwa	Advisor

ABSTRACT

This project is a designing and invention of a plastic sheet punching machine which using in carpet manufacturing. It refers to a method of 2-axis motion in printer machine to be a principle of designing. The major component of this machine consists of 3 parts, 1. 2-axis motion system which is driven by stepping motor and is transmitted power timing belt. 2.Microcontroller which is to control the operation of the stepping motor and the plastic punching. 3.Programme, is written in Visual Basic language, controls the operation of machine which will receive pulses signal from the computer and then this signal will command the rotation of stepper motor. The operation of the machine will start when the program which was written by Visual Basic 6 language, receives layout picture then calculates the coordinates that will be punched by punching machine. After that it will transfer the signal to microcontroller which will command the punching machine to punch the holes on the plastic sheet like the calculated coordinate until it punched all the coordinates of the layout picture. The punched plastic sheet will be taken as the pattern to weave the carpet. To weave the carpet, lay the plastic sheet over the cloth which will be weaven. Then paint color on the plastic sheet when taking the plastic sheet off the pattern line which will be used to weave the carpet further.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จลุล่วงไปได้ด้วยดี ถ้าหากปราศจากคำแนะนำและความช่วยเหลือทางด้านการเงินจากอาจารย์ ดร.ณัฐวุฒิ เดไปวา อาจารย์ที่ปรึกษาปริญญานิพนธ์

ขอขอบคุณ บริษัท สยามคาร์เพทแมนนูแฟคเจอร์ริ่ง จำกัด ที่อนุญาตให้เข้าเยี่ยมชมโรงงานทอพรมและช่วยอธิบายขั้นตอนการผลิตพรมอย่างละเอียด

ขอขอบคุณ นายสมเจตน์ สวนทอง นักศึกษาภาควิชาคอมพิวเตอร์ ที่ให้คำปรึกษาและช่วยสร้างวงจรไฟฟ้าที่ใช้ควบคุมการทำงานของเครื่องเจาะ

ขอขอบคุณพี่ออดที่ช่วยเป็นที่ปรึกษาและช่วยทำให้เครื่องเจาะแผ่นพลาสติกของเราทำงานได้

ท้ายที่สุด คณะผู้จัดทำขอกราบขอบพระคุณบิดา มารดา พี่น้อง รวมถึงครอบครัวของคณะผู้จัดทำที่ให้การสนับสนุนทางด้านการศึกษา กำลังใจ และแนะแนวทางในการดำเนินชีวิต ด้วยความรักและความปรารถนาดีตลอดมา

นายจาตุพล อนันตกุล
นายธงไชย ปรัชญบริรักษ์
นางสาวพิมพ์ศิริ ศิริข่วง

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูปภาพ

หน้าที่

สารบัญรูปภาพ (ต่อ)

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในอุตสาหกรรมทอพรมประกอบด้วยกระบวนการผลิตหลายขั้นตอน เริ่มตั้งแต่การสร้างแบบเพื่อใช้ในการทอพรม โดยแบบที่สร้างขึ้นจะมีลักษณะเป็นแผ่นพลาสติกที่มีการเจาะรูเล็กๆเดินเป็นรูปลายเส้น เมื่อได้แผ่นพลาสติกที่ถูกเจาะตามลายเส้นที่ต้องการจะทอลงบนพรมแล้ว ก็จะนำแผ่นพลาสติกเจาะรูที่ได้ไปทาบลงบนผ้าสำหรับทอพรม แล้วจึงทาสีบนแผ่นพลาสติกเพื่อให้เกิดลวดลายต่างๆตามลายเส้นที่ได้เจาะรู เมื่อนำแผ่นพลาสติกออก ก็จะได้รูแบบแนวเส้นที่จะใช้ในการทอพรมต่อไป ซึ่งขั้นตอนการผลิตที่สำคัญ และเป็นขั้นตอนที่เป็นเหตุให้เกิดการออกแบบและสร้างเครื่องเจาะแผ่นพลาสติกสำหรับทำแบบทอพรม (Plastic Sheet Punching Machine) ก็คือ การเจาะแผ่นพลาสติกเพื่อนำไปทาบลงบนผ้าสำหรับใช้ทอพรม ซึ่งในขั้นตอนนี้โรงงานอุตสาหกรรมทั่วไป ยังไม่มีเครื่องมือเฉพาะทางที่ใช้ในการเจาะรูตามแบบลายเส้น โดยทั่วไปจะเป็นการใช้แรงงานคนในการตอกเพื่อเจาะรูตามลายเส้น เพื่อให้เกิดการพัฒนากระบวนการผลิตในโรงงานอุตสาหกรรมทอพรม ประกอบกับ เพื่อให้ได้เครื่องจักรที่มีราคาถูก สามารถนำไปใช้งานได้จริง และยังนำไปปรับใช้กับอุตสาหกรรมชนิดอื่นได้ จึงได้มีการสร้างเครื่องเจาะแผ่นพลาสติกสำหรับทำแบบทอพรม โดยมีวัตถุประสงค์หลัก เพื่อต้องการลดขั้นตอนการผลิตให้ได้อัตราการผลิตที่มีปริมาณมากขึ้น และยังสามารถจัดสรรทรัพยากรบุคคลไปทำงานในส่วนอื่นเพื่อเป็นการเพิ่มประสิทธิภาพในกระบวนการผลิตได้อีกด้วย

การออกแบบและสร้างเครื่องเจาะแผ่นพลาสติกสำหรับสร้างแบบทอพรม เป็นการนำความรู้ในหลายด้านมาประกอบเข้าด้วยกัน ไม่ว่าจะเป็น ด้านเครื่องกล (Mechanics), ด้านวงจรไฟฟ้า (Electronics) หรือด้านโปรแกรมคอมพิวเตอร์ (Computer Program) ซึ่งหลักการการทำงานของเครื่องจะมีลักษณะคล้ายคลึงกับหลักการทำงานของเครื่องพิมพ์ (Printer) แต่ระบบกลไกต่างๆจะมีขนาดใหญ่กว่าเครื่องพิมพ์ เพื่อรองรับกับขนาดของแบบพลาสติกที่จะทำการเจาะ

เครื่องเจาะแผ่นพลาสติกสำหรับสร้างแบบทอพรม เป็นการปฏิวัติทางด้านเทคโนโลยีในกระบวนการผลิตพรมจากแรงงานคนสู่เครื่องจักร จากขั้นตอนการผลิตหลายๆขั้นตอนก่อนทำการเจาะพลาสติก เหลือเพียงไม่กี่ขั้นตอนในการเจาะพลาสติก อย่างไรก็ตามโครงงานการออกแบบและสร้างเครื่องเจาะพลาสติกสำหรับทำแบบทอพรม ก็ยังจะต้องมีการพัฒนาต่อไป ตามเทคโนโลยีใหม่ๆ ที่เกิดขึ้นอย่างต่อเนื่องในโลกอุตสาหกรรม เพื่อให้เกิดพัฒนาในวงกว้างของระบบอุตสาหกรรมไทย

1.2 วัตถุประสงค์

1. ออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อสร้างแบบสำหรับการทอพรม เพื่อนำไปใช้ในอุตสาหกรรมทอพรม

2. ออกแบบและสร้างโปรแกรมที่ใช้ในการรับข้อมูลรูปภาพ, คำนวณค่าพิกัดในการเจาะ และรับ-ส่งค่าพิกัดที่คำนวณได้กับวงจรควบคุม

3. ออกแบบและสร้างชุดวงจรควบคุม เพื่อที่จะรับ-ส่ง ข้อมูล กับโปรแกรมคอมพิวเตอร์ และควบคุมเครื่องเจาะให้ทำการเจาะตามค่าพิกัดที่คำนวณได้

4. เพื่อให้ได้เครื่องจักรที่สามารถนำไปใช้งานได้จริง และนำไปปรับใช้กับอุตสาหกรรมชนิดอื่นได้

1.3 ขอบเขตของโครงงาน

1. ออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อสร้างแบบสำหรับการทอพรม ให้สามารถนำไปใช้งานได้จริง

2. ออกแบบและสร้างโปรแกรมที่ใช้ในการกำหนดพิกัดของเข็มเจาะ

3. ออกแบบและสร้างไมโครคอนโทรลเลอร์ที่ใช้ควบคุมโปรแกรมที่จะการกำหนดพิกัดของเข็มเจาะและควบคุมการทำงานของเครื่องเจาะพลาสติก เพื่อให้เครื่องสร้างแบบสำหรับทอพรมสามารถเจาะแผ่นพลาสติกตามลายเส้นที่ต้องการได้

1.4 ขั้นตอนการดำเนินโครงงาน

1. ศึกษาหลักการทำงานของระบบการเคลื่อนที่แบบ 2 แกน
2. ออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อสร้างแบบสำหรับการทอพรม
3. ศึกษาโปรแกรมภาษาซีเพื่อสร้างโปรแกรมที่ใช้ในกำหนดพิกัดของเข็มเจาะ
4. ออกแบบและสร้างโปรแกรมที่ใช้ในกำหนดพิกัดของเข็มเจาะ
5. ศึกษาและสร้างไมโครคอนโทรลเลอร์ที่ใช้ควบคุมโปรแกรมที่ใช้ในการกำหนดพิกัดของเข็มเจาะและควบคุมการทำงานของเครื่องเจาะพลาสติก
6. ออกแบบและสร้างไมโครคอนโทรลเลอร์ที่ใช้ควบคุมโปรแกรมที่ใช้ในการกำหนดพิกัดของเข็มเจาะ และควบคุมการทำงานของเครื่องเจาะพลาสติก
7. ติดตั้งและทำการทดลอง
8. สรุปผลการทดลองและข้อผิดพลาดที่เกิดจากการ
9. นำเสนอผลงาน

บทที่ 2
อุปกรณ์และทฤษฎีที่ใช้ในเครื่องเจาะแผ่นพลาสติกสำหรับทอพรม

2.1 สายพาน

สายพานที่ใช้ในการส่งกำลังสามารถแบ่งได้เป็น 4 ประเภท คือ

1. สายพานแบน (Flat Belt) มีภาคตัดขวางเป็นรูปสี่เหลี่ยมผืนผ้า
2. สายพานวี (V Belt) มีภาคตัดขวางเป็นรูปสี่เหลี่ยมคางหมู
3. สายพานไทม์มิ่ง (Timing Belt) มีภาคตัดขวางเป็นรูปสี่เหลี่ยมผืนผ้าหลายรูปวางขนานกันและยึดปิดด้านบนร่วมกัน
4. สายพานกลม หรือสายพานเชือก (Round Belt) มีภาคตัดขวางเป็นรูปกลม

รูปที่ 2-1 แสดงถึงภาคตัดขวางของสายพาน

ซึ่งในการออกแบบและสร้างเครื่องเจาะแผ่นพลาสติกเพื่อสร้างแบบทอพรมนี้ ใช้สายพานแบบไทม์มิ่งเพื่อให้เกิดความไวในการตอบสนองต่อแรงขับเคลื่อยที่ส่งมาจากสเตปเปอร์มอเตอร์ และป้องกันการลื่นไถล นอกจากนี้ การส่งกำลังโดยใช้สายพานยังมีข้อดี คือ

1. สามารถดูดซับการกระแทก และการสั่นสะเทือนได้ดี
2. ติดตั้งง่ายไม่ต้องการการหล่อลื่น
3. เสียงค่อนข้างเบา

สมการหาความยาวของสายพาน

$$\theta d = \pi - 2\sin^{-1}\frac{D-d}{2c} \tag{2.1}$$

$$\theta_D = \pi + 2\sin^{-1}\frac{D-d}{2c} \tag{2.2}$$

$$L = \sqrt{4c^2 - (D-d)^2} + \frac{1}{2}(D\theta_D + d\theta_d) \tag{2.3}$$

โดยที่ θ_d คือ มุมของหน้าสัมผัสที่มูเลย์ตัวใหญ่

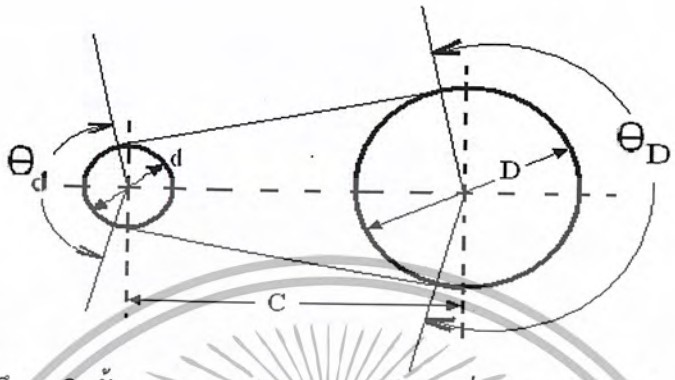
θ_D คือ มุมของหน้าสัมผัสที่มูเลย์ตัวเล็ก

C คือ ระยะห่างระหว่างมูเลย์ทั้งสอง

D คือ เส้นผ่านศูนย์กลางของมูเลย์ตัวใหญ่

d คือ เส้นผ่านศูนย์กลางของมูเลย์ตัวใหญ่

และ L คือ ความยาวของสายพาน



รูปที่ 2-2 แสดงถึงการติดตั้งสายพานแบบเปิดสำหรับขับเพลาที่ขนานกันให้หมุนไปในทิศเดียวกัน

2.2 เฟือง

การคำนวณหาความเร็วรอบของระบบเฟืองจะคำนวณได้จากการหาค่าการขับกันของเฟือง (Train Value)

$$e = \frac{product\ of\ driving\ tooth\ numbers}{product\ of\ driven\ tooth\ numbers} \tag{2.4}$$

$$n_L = en_F \tag{2.5}$$

$$P = \frac{N}{d} \tag{2.6}$$

$$m = \frac{d}{N} \tag{2.7}$$

$$p = \frac{\pi d}{N} = \pi m \tag{2.8}$$

$$pP = \pi \tag{2.9}$$

โดยที่ e คือ ค่าการขับกันของเฟือง (Train Value)

n_L คือ ความเร็วรอบของเฟืองตัวสุดท้ายในระบบการขับ, รอบต่อนาที

n_F คือ ความเร็วรอบของเฟืองตัวแรกในระบบการขับ, รอบต่อนาที

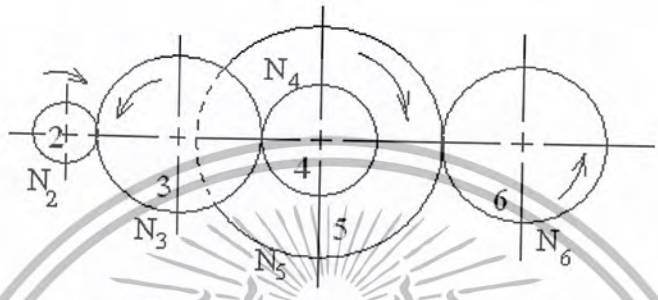
P คือ อัตราส่วนจำนวนฟันของเฟืองต่อระยะพิทช์ (Diametral pitch), จำนวนฟันต่อนิ้ว

N คือ จำนวนฟัน

d คือ ระยะพิทช์ (Pitch diameter), นิ้ว

m คือ โมดูลของเฟือง (Module), มิลลิเมตร

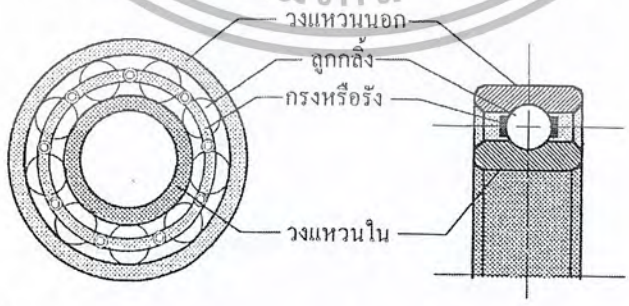
p คือ ระยะบนวงกลมพิทช์จากจุดบนฟันหนึ่งไปยังฟันเฟืองใกล้เคียง (Circular pitch), มิลลิเมตร



รูปที่ 2-3 แสดงถึงการขับกันของระบบเฟือง

2.3 แบริ่ง

แบริ่งแบบกลิ้งหรือตลับลูกปืน เป็นชิ้นส่วนที่ใช้รองรับเพลาและส่งถ่ายภาระจากเพลาผ่านลูกกลิ้ง (Rolling Element) ซึ่งอยู่ระหว่างวงแหวนในและวงแหวนนอก แบริ่งแบบกลิ้งประกอบด้วยวงแหวนในและวงแหวนนอก ซึ่งวงแหวนในใช้สวมเข้ากับเพลา และวงแหวนนอกยึดอยู่ในตัวเรือนของแบริ่ง ภายในตัวแบริ่งจะมีลูกกลิ้งแบบเม็ดกลม หรือเม็ดทรงกระบอกอยู่ระหว่างวงแหวนในและวงแหวนนอกโดยมีกรงหรือรัง (Cage) ยึดคั่นแยกลูกกลิ้งให้มีระยะห่างคงที่ และเมื่อวงแหวนใดวงแหวนหนึ่งหมุน ลูกกลิ้งจะกลิ้งอยู่ในรางของวงแหวนซึ่งทำให้ความเสียดทานระหว่างรางและลูกกลิ้งลดลงมาก แต่เนื่องจากมีพื้นที่ผิวสัมผัสระหว่างรางและลูกกลิ้งมีน้อย ประกอบกับภาวะต่อหนึ่งหน่วยพื้นที่มีค่าสูง ลูกกลิ้งและวงแหวนจึงต้องทำจากเหล็กกล้าที่มีความแข็งและความต้านแรงสูง



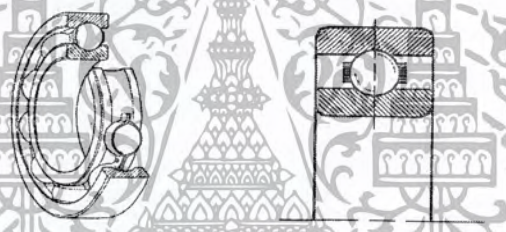
รูปที่ 2-4 แสดงถึงส่วนประกอบภายในของแบริ่งแบบกลิ้งหรือตลับลูกปืน

คุณสมบัติทั่วๆไปของแบริ่งแบบกลิ้ง ได้แก่

1. ความเสียดทานขณะเริ่มต้นหมุนและเมื่อหมุนแล้วเกือบเท่ากัน ยกเว้นที่ความเร็วรอบสูง
2. ต้องการการหล่อลื่นและบำรุงรักษาน้อย
3. ใช้เนื้อที่ตามแนวแกนน้อย
4. มีอายุการใช้งานจำกัดเนื่องจากรางหรือลูกกลิ้งมักจะเกิดการสึกหรอหรือความล้าที่ผิว
5. การถอดเปลี่ยนใหม่ทำได้ง่าย
6. แบริ่งแบบกลิ้งบางประเภทสามารถรับภาระได้ทั้งในแนวรัศมีและในแนวแกน

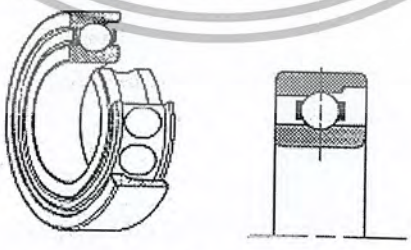
2.3.1 ประเภทแบริ่งแบบเม็ดกลม

1. แบริ่งแบบเม็ดกลมร่องลึก (Deep Groove Ball Bearing) เป็นแบริ่งแบบเม็ดกลมร่องลึกแถวเดียว มีรางเป็นร่องลึกสำหรับให้เม็ดกลมกลิ้ง โดยปกติใช้สำหรับรับภาระในแนวรัศมี แต่สามารถรับภาระในแนวแกนได้ถึงร้อยละ 70 ของภาระในแนวรัศมี เป็นแบริ่งที่ใช้งานอย่างกว้างขวางมากที่สุด



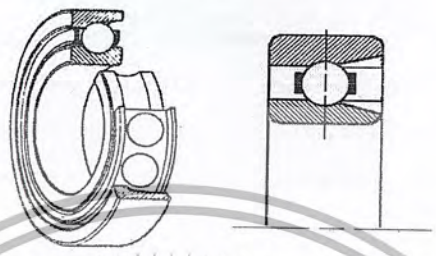
รูปที่ 2-5 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมร่องลึก

2. แบริ่งแบบเม็ดกลมร่องลึกมีรอยบากเติมเม็ดกลม (Filling Notch) จะมีรอยบากบนด้านหนึ่งของวงแหวนสำหรับเติมเม็ดกลม ซึ่งทำให้เพิ่มความสามารถในการรับภาระในแนวรัศมี แต่ความสามารถในการรับภาระในแนวแกนจะลดลง เนื่องจากเม็ดกลมชนรอยบาก



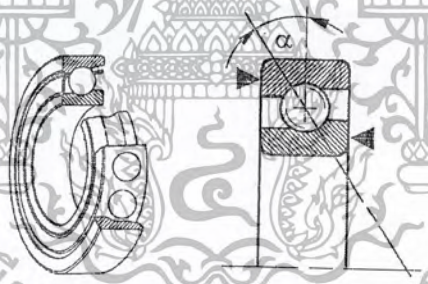
รูปที่ 2-6 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมร่องลึกมีรอยบากเติมเม็ดกลม

3. แมกนีโตแบริ่ง (Magneto Bearing) ร่องที่วงแหวนในของแบริ่งแบบนี้จะตื้นกว่าแบริ่งแบบเม็ดกลมร่องลึก ด้านหนึ่งของวงแหวนนอกจะมีบ่า ร่องอีกด้านไม่มีบ่า วงแหวนนอกสามารถแยกส่วนออกมาได้ ซึ่งนับเป็นข้อดีต่อการประกอบ แมกนีโตแบริ่งเป็นแบริ่งขนาดเล็กมีขนาดเส้นผ่านศูนย์กลางเพลาตั้งแต่ 4 ถึง 30 มิลลิเมตร



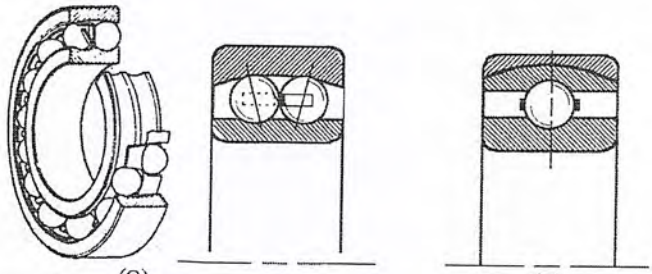
รูปที่ 2-7 แสดงถึงลักษณะภายในของแมกนีโตแบริ่ง

4. แบริ่งแบบเม็ดกลมสัมผัสเชิงมุม (Angular Contact Ball Bearing) แบริ่งแบบนี้แยกส่วนไม่ได้ แต่สามารถรับภาระในแนวแกนได้สูง โดยความสามารถในการรับภาระจะขึ้นอยู่กับมุมสัมผัส (Contact Angle) α โดยมุมสัมผัส α ที่โตกว่าจะสามารถรับภาระได้สูงกว่า ซึ่งในงานที่มีความเร็วรอบสูงมักจะใช้แบริ่งแบบเม็ดกลมที่มีมุมสัมผัสน้อยๆ



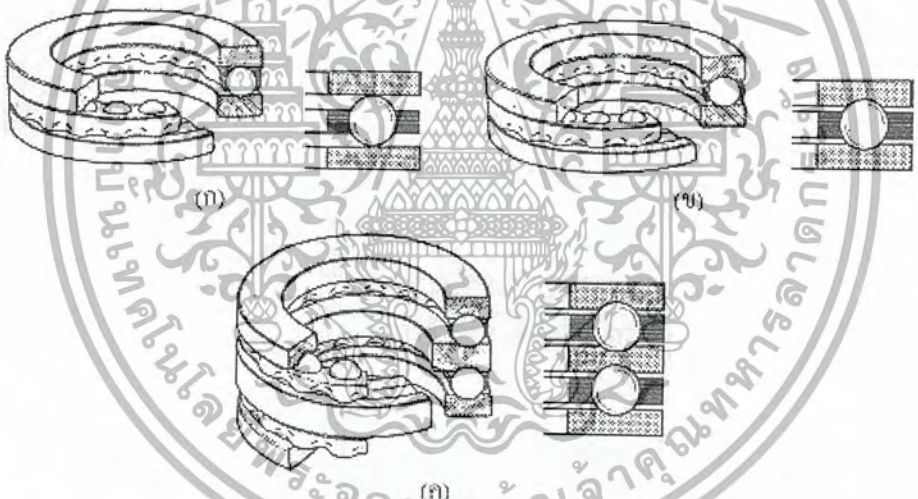
รูปที่ 2-8 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมสัมผัสเชิงมุม

5. แบริ่งแบบเม็ดกลมปรับแนวแกนได้เอง (Self-aligning Ball Bearing) เป็นแบริ่งแบบแยกส่วนไม่ได้ และมีความสามารถในการรับภาระต่ำกว่าแบบร่องลึก เนื่องจากรัศมีของวงแหวนโตขึ้นทำให้เกิดความเค้นสัมผัสสูง มีทั้งแบบปรับแนวแกนได้เองภายใน แบบปรับแนวแกนได้เองภายนอก



รูปที่ 2-9 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมปรับแนวแกนได้เองจากภายใน (ก) และแบบปรับแนวแกนได้เองภายนอก (ข)

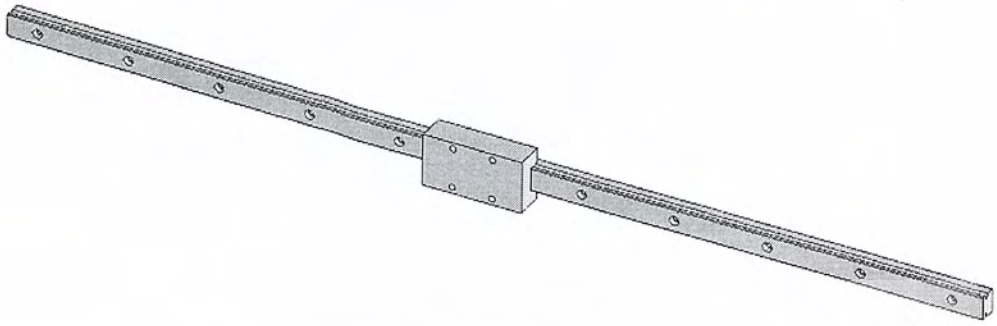
6. แบริ่งแบบเม็ดกลมกันรุน (Thrust Ball Bearing) มีอยู่ 3 ประเภท ได้แก่ แบบรางราบ, แบบรางร่องเดี่ยว และแบบรางร่องคู่ แบริ่งแบบนี้สามารถแยกส่วนได้หากต้องการได้แนวแกนที่เที่ยงตรงของเพลา และในการใช้งานนั้นแบริ่งชนิดนี้จะต้องการภาระต่ำสุดที่ความเร็วรอบสูง



รูปที่ 2-10 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมกันรุน แบบรางราบ(ก) แบบรางร่องเดี่ยว (ข) และแบบแบบรางร่องคู่ (ค)

2.4 ลิเนียร์แบริ่ง (Linear Bearing)

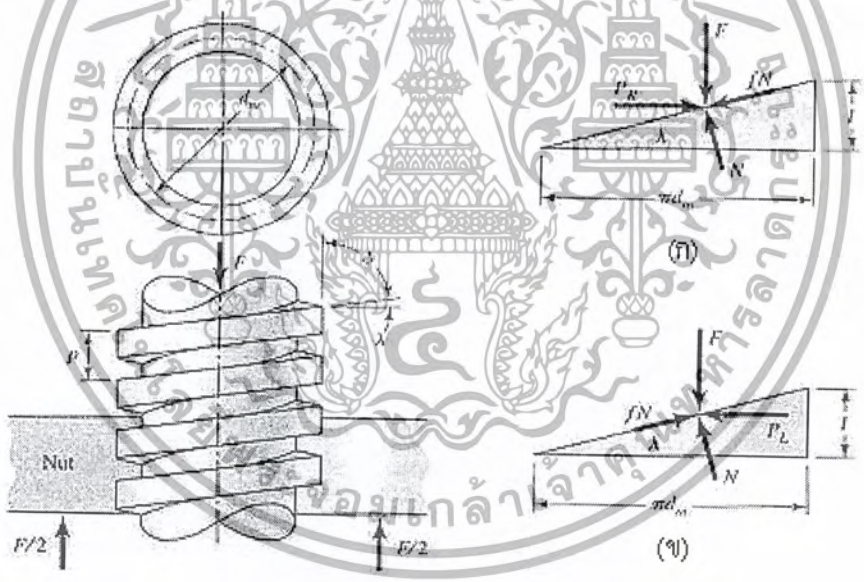
ลิเนียร์แบริ่ง เป็นแบริ่งชนิดหนึ่งที่คุณสมบัติทำให้เกิดความราบเรียบในการลื่นไถลตามแนวแกน มีความเสียดทานต่ำ และมีการเคลื่อนที่เป็นแนวเส้นตรง นอกจากนี้ลิเนียร์แบริ่งยังมีคุณสมบัติพิเศษที่เหมาะกับการทำงาน คือ ยิ่งลิเนียร์แบริ่งรับภาระในแนวตั้งกับแนวการเคลื่อนที่มากขึ้น ก็จะยิ่งส่งผลให้ลิเนียร์แบริ่งมีความราบเรียบในการลื่นไถลตามแนวการเคลื่อนที่มากขึ้น ลิเนียร์แบริ่งประกอบด้วย 2 ส่วน คือ ส่วนที่เป็นฐานที่อยู่กับที่ และ ส่วนข้างบนที่เลื่อนได้ โดยมีลูกปืนเป็นตัวกั้นระหว่างส่วนประกอบ 2 ส่วนนี้



รูปที่ 2-11 แสดงถึงลักษณะภายนอกของลิเนียร์แบริ่ง

2.5 สกรูส่งกำลัง

การยกน้ำหนักขึ้นหรือลงของสกรูส่งกำลังสามารถคำนวณได้โดย



รูปที่ 2-12 แสดงถึงทิศทางของแรงกระทำบนเกลียวของสกรูเมื่อยกภาระขึ้น (ก) และยกภาระลง (ข)

$$T_R = \frac{Fd_m}{2}\left(\frac{l + \pi f d_m}{\pi d_m - fl}\right) \tag{2.1}$$

$$T_L = \frac{Fd_m}{2}\left(\frac{\pi f d_m - l}{\pi d_m + fl}\right) \tag{2.2}$$

โดยที่ T_R คือ แรงที่จำเป็นต้องใช้ในการยกภาระขึ้น, นิวตัน-เมตร

T_L คือ แรงที่จำเป็นต้องใช้ในการยกภาระลง, นิวตัน-เมตร

F คือ ขนาดของภาระ, นิวตัน

D_m คือ เส้นผ่านศูนย์กลางเสมือน, มิลลิเมตร

f คือ สัมประสิทธิ์แรงเสียดทาน

2.6 มอเตอร์กระแสตรง

มอเตอร์กระแสตรงจะมีการทำงานเพียง 3 สเตจเท่านั้น คือ หมุนตามเข็มนาฬิกา หมุนทวนเข็มนาฬิกา และ หยุดนิ่ง ซึ่งอัตราความเร็วในการหมุนและแรงบิดจะขึ้นอยู่กับกระแสและแรงดันที่จ่ายให้มัน ถ้าหากแรงบิดของมอเตอร์ไม่เพียงพอต่อการทำงานเราก็สามารถที่จะทำการทดรอบของการหมุนลงก็จะได้แรงบิดเพิ่มมากขึ้น

รูปที่ 2-13 แสดงถึงลักษณะภายนอกของมอเตอร์กระแสตรง

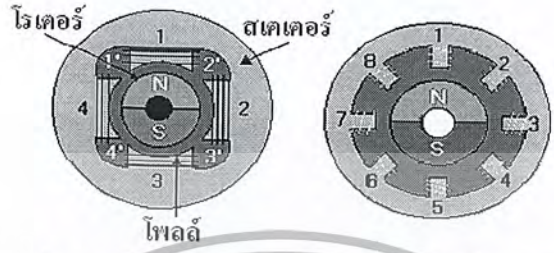
2.7 สเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์เป็นมอเตอร์ที่เมื่อป้อนกระแสไฟฟ้าให้กับมอเตอร์แล้ว จะทำให้มอเตอร์เกิดการหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งต่างจากมอเตอร์โดยทั่วไปที่จะหมุนทันทีและตลอดเวลาเมื่อป้อนแรงดันไฟฟ้า

รูปที่ 2-14 แสดงถึงโครงสร้างภายนอกของสเต็ปปิ้งมอเตอร์

ข้อดีของสเต็ปปิ้งมอเตอร์ คือ สามารถกำหนดตำแหน่งของการหมุน ด้วยตัวเลข, องศา หรือระยะทางได้อย่างละเอียดโดย ใช้คอมพิวเตอร์ หรือไมโครคอนโทรลเลอร์เป็นเครื่องกำหนดและจัดเก็บตัวเลข

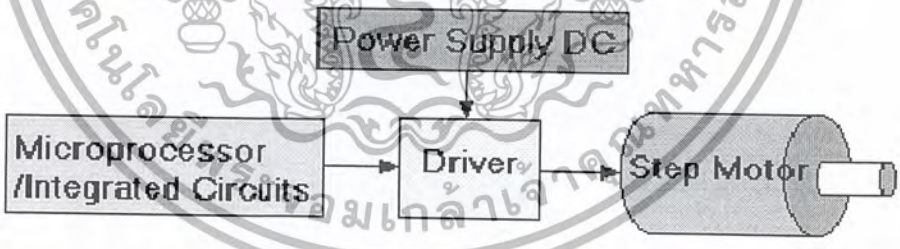
โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ทำจากแผ่นเหล็กวงแหวน ซึ่งแผ่นเหล็กแต่ละแผ่นจะมีซี่ยื่นออกมาประกบกันเป็นชั้นๆ โดยที่แต่ละซี่นั้นจะมีขดลวด พันสวมอยู่ เมื่อมีการป้อนกระแสไฟฟ้าผ่านขดลวดจะทำให้เกิดสนามแม่เหล็กไฟฟ้า



รูปที่ 2-15 แสดงถึงโครงสร้างภายในของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์มักจะถูกนำไปใช้งานใช้งานลักษณะของระบบเปิด คือ สเต็ปปิ้งมอเตอร์สามารถทำงานได้โดยไม่ต้องมีการป้อนค่าพารามิเตอร์กลับมา แต่วิธีกำหนดตำแหน่งที่แน่นอนนั้น จะต้องมีการป้อนกลับไปยังระบบ และตัวบอกตำแหน่งว่าถูกต้องหรือผิดพลาดให้รับทราบ

ซึ่งวิธีที่ใช้กับสเต็ปปิ้งมอเตอร์ คือ การนำลิมิตสวิตช์ติดตามตำแหน่งที่จะตรวจจับ เมื่อสเต็ปปิ้งมอเตอร์เริ่มทำงาน แล้วหมุนไปจนถึงตำแหน่งของสวิตช์ตรวจจับสัญญาณ ลิมิตสวิตช์ก็จะทำการป้อนสัญญาณกลับไปสู่ระบบ ดังนั้นระบบจะรู้ถึงตำแหน่งที่สเต็ปปิ้งมอเตอร์เคลื่อนที่ไปได้ตลอด ซึ่งตัวไมโครคอนโทรลเลอร์เองจะมีจุดอ้างอิงไว้ให้สำหรับการเริ่มต้นทำงาน และอ้างอิงตำแหน่งได้อย่างถูกต้อง



รูปที่ 2-16 แสดงถึงการควบคุมระบบสเต็ปปิ้งมอเตอร์

2.7.1การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์

การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ จะทำการสั่งงานให้สเต็ปปิ้งมอเตอร์ทำงานไปทีละสเต็ป โดยการจ่ายกระแสไฟฟ้าเข้าไปยังขดลวดในแต่ละขอบบนของสเตเตอร์โดยการป้อนกระแสไฟฟ้าจะทำเป็นลำดับในรอบที่ถูกต้อง ซึ่งการป้อนกระแสไฟฟ้าเป็นลำดับจะแบ่งได้เป็น 3 รูปแบบ ซึ่งทั้ง 3 แบบจะมีข้อดี-ข้อเสียต่างกันออกไป

2.7.1.1 การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบคลื่น

จะเป็นการกระตุ้นแบบที่ง่ายที่สุด ซึ่งจะทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งๆเรียงกันไป ตัวอย่างเช่น ขดที่ 1, 2, 3, 4, 1, 2, 3, 4 เป็นลำดับอย่างนี้ หรือ ขด 1, 4, 3, 2, 1, 4, 3, 2 เป็นลำดับกันไป ทั้งนี้ขึ้นอยู่กับทิศทางที่ต้องการให้มอเตอร์หมุน โดยวงจรที่นำมากระตุ้นนั้น จะมีราคาค่อนข้างจะถูกกว่า และง่ายกว่า

ลำดับการกระตุ้นของขดลวดแบบคลื่นสามารถแสดงได้ดังตารางต่อไปนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2		ON		
3			ON	
4				ON
5	ON			

ตารางที่ 2-1 แสดงถึงลำดับการสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบคลื่น

2.7.1.2 การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบ 2 เฟส

การกระตุ้นขดลวดแบบนี้จะคล้ายกับการกระตุ้นในแบบคลื่น แต่จะแตกต่างกันตรงที่ แบบ 2 phase จะกระตุ้นขอลวดทีละ 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และจะเรียงลำดับกันไป ดังเช่นแบบเดียวกับแบบคลื่น ตัวอย่างเช่น ขดที่ 12, 23, 34, 41, 12, 23, 34, 41 เรียงลำดับกันไปเรื่อยๆ หรือจะเป็น 14, 43, 32, 21, 14, 43, 32, 21 เรียงกันไปเรื่อยๆเช่นกัน

ข้อดีของการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบ 2 เฟส คือ จะให้แรงบิดได้มากกว่า แบบคลื่น เนื่องจากสเตเตอร์จะหมุนด้วยแรงดึงแบบเต็มๆแรงจากทั้ง 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน

ส่วนข้อเสียของการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบ 2 เฟส คือ การใช้กำลังไฟในการกระตุ้นขดลวดนั้นต้องใช้กำลังไฟมากขึ้นเป็น 2 เท่าของแบบคลื่น

ลำดับการกระตุ้นของขดลวดแบบ 2 เฟส สามารถแสดงได้ดังตารางต่อไปนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON	ON		
2		ON	ON	

3			ON	ON
4	ON			ON
5	ON	ON		
6		ON	ON	

ตารางที่ 2-2 แสดงถึงลำดับการสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบ 2 เฟส

2.7.1.3 การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบครึ่งสเต็ป

เป็นรูปแบบผสมผสานของการกระตุ้นระหว่างแบบคลื่น กับ 2 เฟส เพื่อเพิ่มให้จำนวนรอบของสเต็ปมากขึ้นเป็น 2 เท่า ซึ่งในระบบนี้ จะทำการกระตุ้นขดลวดเรียงกันไปเรื่อยๆ เป็นลำดับ ตัวอย่างเช่น ขดที่ 1, 12, 2, 23, 3, 34, 4, 41, 1, 12, 2, 23, 3, 34, 4, 41, 1 เป็นลำดับอยู่อย่างนี้เรื่อยไป ถ้าจะกลับทิศทางการหมุนก็จะได้เป็น 1, 41, 4, 43, 3, 32, 2, 21, 1, 41, 4, 43, 3, 32, 2, 21, 1 เป็นลำดับ

ข้อดีของการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบครึ่งสเต็ป คือ จะให้แรงบิดที่เพิ่มมากขึ้นในช่วงสเต็ปที่มีระยะสั้นลง นอกจากนี้ในแต่ละที่เกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกันก็จะส่งผลให้ค่าตำแหน่งความถูกต้องมากขึ้นอีกด้วย

ข้อเสียของการควบคุมการหมุนของสเต็ปเปอร์มอเตอร์แบบนี้จะเป็นเช่นเดียวกับ แบบ 2 เฟส คือ ต้องจ่ายกำลังไฟเป็น 2 เท่า ของแบบคลื่น หรือใช้กำลังไฟเท่ากับแบบ 2 เฟส นั่นเอง

ลำดับการกระตุ้นของขดลวดแบบครึ่งสเต็ป สามารถแสดงได้ดังตารางต่อไปนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2	ON	ON		
3		ON		
4		ON	ON	
5			ON	
6			ON	ON
7				ON
8	ON			ON
9	ON			

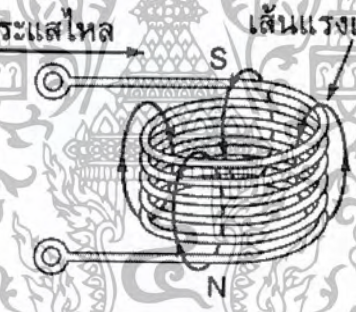
10	ON	ON		

ตารางที่ 2-3 แสดงถึงลำดับการสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบครึ่งเฟส

ซึ่งในการสร้างเครื่องเจาะแผ่นพลาสติกเพื่อสร้างแบบสำหรับทอพรม ใช้วิธีการกระตุ้นขดลวดแบบ 2 เฟส เพื่อให้ได้แรงบิดมาก ในช่วงสเต็ปสั้นๆ เพื่อให้การเคลื่อนที่ในแต่ละรอบมีความละเอียดมากที่สุด

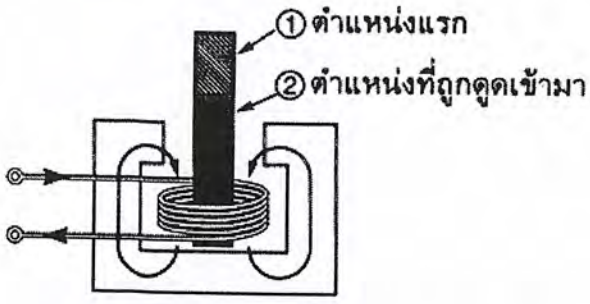
2.8 โซลินอยด์ไฟฟ้า

โซลินอยด์ไฟฟ้ามีหลักการทำงานดังนี้ คือ เมื่อมีกระแสไฟฟ้าไหลในขดลวดตัวนำใดๆก็ตาม จะก่อให้เกิดสนามแม่เหล็กขึ้นรอบๆตัวนำนั้น และหากนำเส้นลวดมาขดเป็นวงๆ หลายๆวง ก็จะเกิดลักษณะของขดลวดขึ้น โดยสนามแม่เหล็กที่เกิดจากขดลวดแต่ละขดจะอยู่ในทิศทางเสริมกัน และก่อกำเนิดเป็นเส้นแรงของสนามแม่เหล็กถาวรแท่งหนึ่ง ซึ่งพร้อมที่จะดูดสารแม่เหล็กทันที แต่เนื่องจากสภาพรอบๆ ขดลวดอาจเป็นอากาศ เส้นแรงแม่เหล็กจึงไม่เข้มข้นมากนัก



รูปที่ 2-17 แสดงถึงทิศทางของสนามแม่เหล็กที่เกิดขึ้นในขดลวดที่มีกระแสไหล

เพื่อไม่ให้สนามแม่เหล็กที่เกิดขึ้นกระจัดกระจาย จึงใส่แกนเหล็กอ่อนรูปตัวซีล้อมรอบๆขดลวดไว้ เพื่อให้สนามแม่เหล็กที่เกิดขึ้นกระจัดกระจายออกไป และเพื่อให้สนามแม่เหล็กมีความเข้มมากขึ้น จากจุดนี้หากนำเอาแกนกระทุ้ง (Plunger) มาใส่เข้าไปตรงกลางวงของขดลวดในตำแหน่งที่ 1 แกนกระทุ้งจะถูกดูด ให้ลึกเข้ามาจนสนิทในตำแหน่งที่ 2 ความสัมพันธ์ของแรงกับระยะช่วงชักของโซลินอยด์ ในช่วงชักไกลๆจะมีแรงน้อยมาก แต่ในทางตรงกันข้ามช่วงชักที่มีระยะใกล้ๆก็จะมีแรงมากขึ้นเป็นทวีคูณ



รูปที่ 2-18 แสดงถึงการเพิ่มเหล็กอ่อนเข้ามาเพื่อเพิ่มความเข้มของสนามแม่เหล็ก

2.8.1 ประเภทของโซลินอยด์ไฟฟ้า

โซลินอยด์ไฟฟ้าสามารถแบ่งได้เป็น 2 ประเภท คือ โซลินอยด์ที่ใช้ไฟฟ้ากระแสตรง และโซลินอยด์ที่ใช้ไฟฟ้ากระแสสลับ ข้อแตกต่างระหว่างโซลินอยด์ที่ใช้ไฟฟ้ากระแสตรง และโซลินอยด์ที่ใช้ไฟฟ้ากระแสสลับ คือ โซลินอยด์ที่ใช้ไฟกระแสตรงจะก่อให้เกิดกระแสที่ไหลในขดลวดค่อนข้างคงที่ไม่เปลี่ยนแปลง ไม่ว่าแกนกระทุ้งจะอยู่ในตำแหน่งใดก็ตาม แต่โซลินอยด์ที่ใช้ไฟกระแสสลับจะมีกระแสในขณะที่แกนกระทุ้งอยู่นอกขดลวดค่อนข้างสูง และเมื่อแกนกระทุ้งถูกดูดเข้ามาจนสุดขดลวด กระแสจะลดต่ำลง ด้วยเหตุนี้เองที่ทำให้ต้องระวังอย่าให้เกิดการกระทุ้งในโซลินอยด์ไฟสลับ เพราะจะทำให้เกิดกระแสมากๆไหลค้างอยู่ ทำให้ขดลวดร้อนขึ้น และ อาจจะไหม้เสียหายได้

ในการออกแบบและสร้างเครื่องเจาะแผ่นพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้จะใช้โซลินอยด์ที่ใช้ไฟกระแสตรงเพื่อเกิดกระแสในขดลวดที่คงที่ และมีอายุการใช้งานที่ยาวนาน

2.8.2 การเลือกใช้โซลินอยด์ไฟฟ้า

การเลือกใช้โซลินอยด์ไฟฟ้าควรพิจารณาถึงองค์ประกอบต่างๆดังนี้ คือ

1. แรงดันใช้งานซึ่งไม่ว่าจะเป็นไฟฟ้ากระแสตรง หรือไฟฟ้ากระแสสลับก็ตาม จำเป็นจะต้องดูความถี่ใช้งานให้ตรงตามความต้องต้องการด้วย ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้โซลินอยด์ที่มีความถี่ 55 เฮิร์ซ

2. ช่วงชักในการใช้งาน (Operating Stroke) ของโซลินอยด์จะต้องเคลื่อนที่เป็นระยะทางเท่าใดมักจะกำหนดเป็นมิลลิเมตร ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้โซลินอยด์ที่มีช่วงชักในการใช้งานยาว 20 มิลลิเมตร

3. ขนาดของภาระ ว่าต้องใช้แรงขนาดเท่าใดมักจะกำหนดเป็นกรัม ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้โซลินอยด์รับภาระขนาดประมาณ 500 กรัม

4. เป็นการใช้งานอย่างต่อเนื่องหรือไม่ ซึ่งการใช้งานอย่างต่อเนื่องในที่นี้ หมายถึง การใส่แรงดันไฟเข้าขดลวดค้างไว้โดยขดลวดไม่ไหม้ หรือเป็นการใส่แรงดันแบบเป็นจังหวะๆ ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้การใส่แรงดันแบบเป็นจังหวะๆ เพื่อให้เกิดการกระทุ้งเพื่อทำการเจาะแผ่นพลาสติก

2.9 ไมโครคอนโทรลเลอร์ MCS-51

2.9.1 ความหมายของ ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณเอาท์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิคส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบ ในขณะที่มีขีดความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม ดังนั้น ไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ

2.9.2 โครงสร้างของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในโครงงานนี้จะอ้างอิงถึงไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลช (Flash Memory) ของ Atmel Corporation มีเบอร์ขึ้นต้นด้วย AT89 เหตุผลที่ใช้ไมโครคอนโทรลเลอร์แบบนี้ มีด้วยกันหลายประการ คือ

1. หน่วยความจำโปรแกรมภายในตัวไมโครคอนโทรลเลอร์เป็นแบบแฟลช ทำให้สามารถลบและเขียนใหม่ได้นับพันครั้ง จึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดี่ยวได้โดยไม่ต้องใช้หน่วย ความจำภายนอก ส่งผลให้สามารถใช้งานพอร์ตอินพุต-เอาท์พุตของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

2. การใช้ต้นทุนและเวลาในการพัฒนาระบบไมโครคอนโทรลเลอร์ลดลงอย่างมาก เนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูเลเตอร์ และเครื่องโปรแกรมอีพรอม

3. บริษัทผู้ผลิตได้ทำการผลิตไมโครคอนโทรลเลอร์ตระกูลนี้ออกมาหลายเบอร์ และมีความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง

4. ด้วยการใช้หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ ทำให้สามารถป้องกันการคัดลอกข้อมูลของหน่วยความจำโปรแกรมได้เป็นอย่างดี

5. ในบางเบอร์ของไมโครคอนโทรลเลอร์ที่ผลิตโดย Atmel Corporation สามารถทำการโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยที่ไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาทำการโปรแกรมใหม่ หรือเรียกว่า "การโปรแกรมในวงจร" หรือ "ในระบบ (In-system Programming)" ทำให้การพัฒนาหรือการซ่อมบำรุง ตลอดจนการปรับปรุงหรืออัปเกรดข้อมูลในหน่วยความจำโปรแกรมทำได้อย่างสะดวก ภายใต้งบประมาณที่ไม่สูงมากนัก

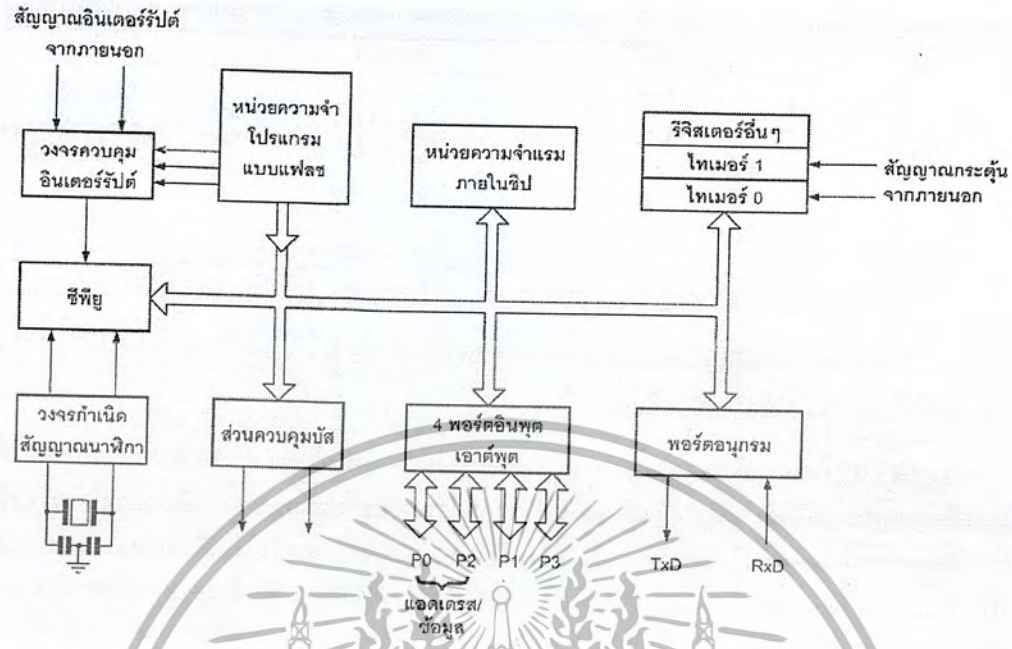
6. ชุดคำสั่งและสถาปัตยกรรมพื้นฐานเหมือนกับไมโครคอนโทรลเลอร์ MCS-51 ของผู้ผลิตอื่น

2.9.3 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx มีคุณสมบัติดังนี้คือ

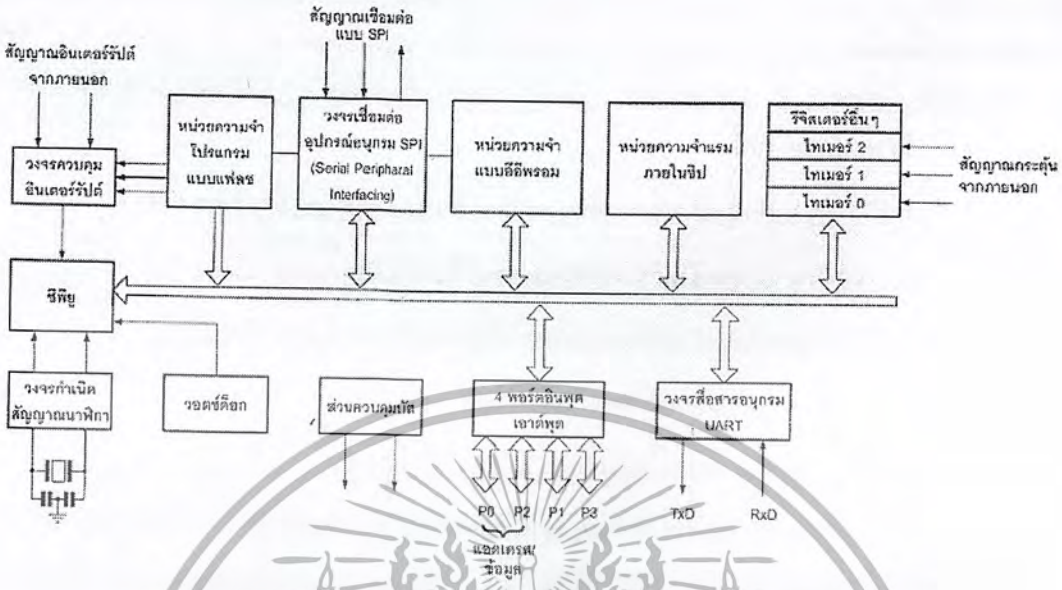
- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้เป็นพันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม และในบางเบอร์จะมีหน่วยความจำแบบอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุต-เอาท์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเทอร์รัปได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

ด้านโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx พบว่าโครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแต่แตกต่างกันเฉพาะหน่วยความจำแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ในอนุกรม 87xx หน่วยความจำภายในจะเป็นแบบอีพรอม และบางเบอร์สามารถโปรแกรมได้เพียงครั้งเดียว



รูปที่ 2-19 แสดงถึงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx

ในโครงสร้างพื้นฐานของอนุกรม AT89Sxx พบว่า มีส่วนประกอบที่เพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน เช่น วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์อนุกรมนี้ใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบ หรือเรียกว่า การโปรแกรมวงจร และยังมีไทม์เมอร์/เคาน์เตอร์ขนาด 16 บิต ที่เพิ่มเติมเข้ามาอีก 1 ตัว เป็นไทม์เมอร์ และยังเพิ่มวงจรว็อตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียูอีกด้วย



รูปที่ 2-20 แสดงถึงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Sxx

2.9.4 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

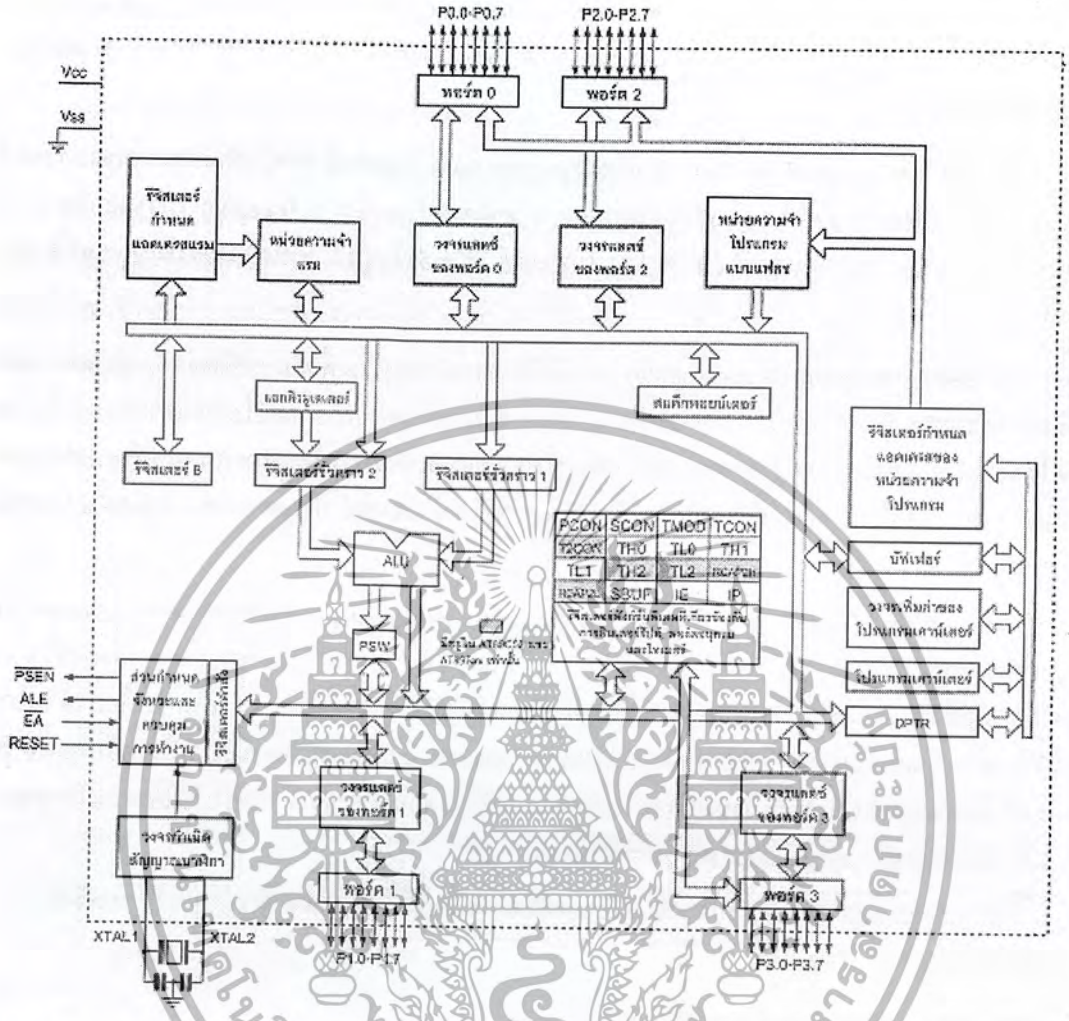
ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังนี้

1. ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5v

2. ขา GND เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

3. ขาพอร์ต 0 (P0.0-P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุต และเอาท์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (Float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงาน เป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

4. ขาพอร์ต 1 (P1.0-P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุต และเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทม์เมอร์ 2 และ P1.1 เป็นขาอินพุตริกเกอร์ของไทม์เมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ



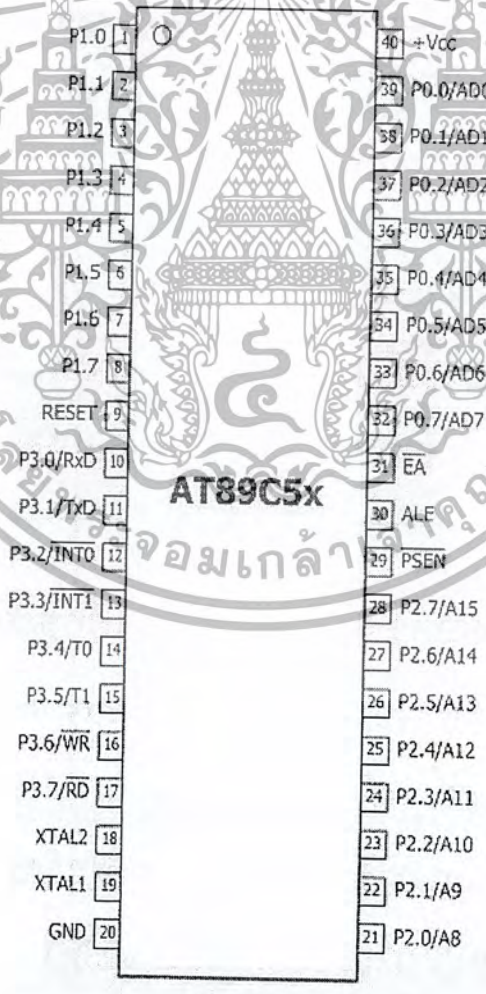
รูปที่ 2-21 แสดงถึงรายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของ Atml

5. ขาพอร์ต 2 (P2.0-P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (Float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

6. ขาพอร์ต 3 (P3.0-P3.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (Float) จึงมีอินพุตอิมพีแดนซ์สูง

สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดข้างต้นต่อไปนี้

- P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD
- P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD
- P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัป จากภายนอกช่อง 0 หรือขา $\overline{INT0}$
- P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัป จากภายนอกช่อง 1 หรือขา $\overline{INT1}$
- P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทม์เมอร์จากภายนอกช่อง 0 หรือขา T0
- P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเทอร์รัปจากภายนอกช่อง 1 หรือขา T1
- P3.6 ใช้เป็นขาสัญญาณ $\overline{WR}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก
- P3.7 ใช้เป็นขาสัญญาณ $\overline{RD}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก



รูปที่ 2-22 แสดงถึงการจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x

- ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีนไซเคิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ
- ขา $\overline{ALE}$ /PROG (Address Latch Enable/Program Pulse Input) เป็นขาที่ใช้ในการควบคุมการแลตซ์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขาสำหรับรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีพรอม
- ขา $\overline{PSEN}$ (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง ในแต่ละแมชชีนไซเคิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆ ออกมา
- ขา $\overline{EA}/V_{pp}$ (External Access Enable/Programming Voltage Input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอติดต่อกับหน่วยความจำภายนอก แต่ถ้าหากขานี้เป็น "1" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูง สำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12V
- ขา XTAL1 และขา XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

2.9.5 โครงสร้างและการทำงานของพอร์ตในไมโครคอนโทรลเลอร์

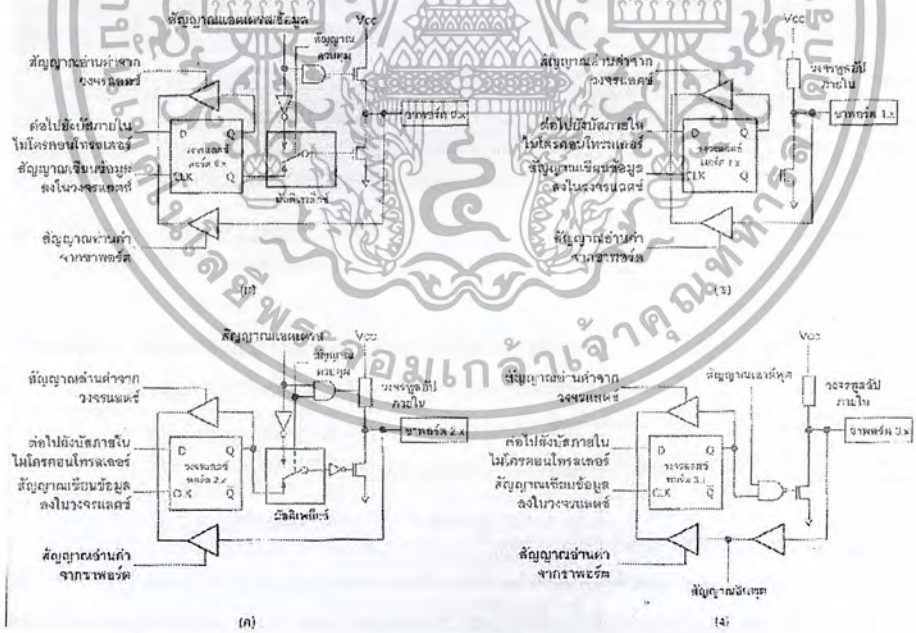
ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ต คือ พอร์ต 0 จนถึงพอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้า และเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงแลตซ์ และวงจรขับตลอดจนบัฟเฟอร์อินพุต ดังแสดงในรูปที่ 2-20

ที่พอร์ต 0 และ พอร์ต 2 จะใช้งานเป็นพอร์ตอินพุต และเอาต์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ต และพอร์ต 1 บางขา นอกจากจะใช้เป็นขาพอร์ตอินพุต-เอาต์พุต ตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด

วงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช เป็นวงจรของพอร์ต 0 และวงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกจากกัน นั่นคือสัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิปฟลอปในขณะที่ข้อมูลจะผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิปฟลอป ที่พอร์ตนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ต ว่าต้องการใช้งานเป็นขาพอร์ตอินพุต-เอาท์พุตปกติหรือใช้ในการติดต่อกับหน่วยความจำภายนอกไมโครคอนโทรลเลอร์ เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัปภายใน หากมีการนำพอร์ต 0ไปใช้งานเป็นพอร์ตอินพุตจะต้องต่อตัวต้านทานพูลอัปภายนอกเข้าที่ขาพอร์ต 0 ทุกขาด้วย

พอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน

วงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัปเพิ่มเติมเข้ามา ส่วนวงจรภายในของพอร์ต 3 พบว่ามีลักษณะคล้ายกับพอร์ต 1 แต่มีการเพิ่มเติมวงจรบัฟเฟอร์ และวงจรอินพุต-เอาต์พุตเพื่อทำงานในฟังก์ชันพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานหน้าที่พิเศษได้ทุกขา



รูปที่ 2-23 แสดงถึงวงจรภายในของพอร์ตทุกพอร์ต 0 (ก), พอร์ต 1 (ข) พอร์ต 2 (ค) และพอร์ต 3 (ง) ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

2.9.6 การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตและเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุตต้องเริ่มต้นด้วยการเขียนข้อมูล "1" มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้นๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น "1" สามารถรับสัญญาณลอจิก "0" จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนี้ อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ควรกำหนดให้ทำงานในสภาวะลอจิก "0" จะดีและสะดวกที่สุด ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก "0" แล้ว

2.9.7 การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล "0" ออกไปทางเอาต์พุตก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปยังเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก "0" ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล "1" ออกไป ก็ให้เขียนข้อมูล "1" ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานเป็นพอร์ตเอาต์พุตแต่ละขา หรือแต่ละบิต ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแส หรือที่เรียกว่า กระแสซอร์ส (Source Current) ได้สูงสุด 10 mA และทุกขารวมกันในแต่ละพอร์ต (ทั้ง 8 บิต) สูงสุด 26 mA สำหรับพอร์ต 0 และ 15 mA สำหรับพอร์ต 1-3 ในกรณีที่ใช้งานทุกพอร์ตเอาต์พุตจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาต์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถ ในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาต์พุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง

บทที่ 3
การประยุกต์ใช้ไมโครคอนโทรลเลอร์ MCS-51

3.1พอร์ตอนุกรม

การส่งข้อมูลของพอร์ตอนุกรมจะส่งข้อมูลครั้งละ 1 บิต ทำให้มีความเร็วในการส่งข้อมูลช้ากว่าของพอร์ตขนานซึ่งสามารถส่งข้อมูลได้ถึง 8 บิตในเวลาเดียวกัน แต่ข้อดีของพอร์ตอนุกรมที่เหนือกว่าพอร์ตขนานคือความสามารถในการส่งข้อมูลได้ในระยะไกลกว่า และยังใช้สายสัญญาณน้อยกว่าอีกด้วย

3.1.1มาตรฐาน RS-232C

มาตรฐานนี้ได้รับความนิยมอย่างสูงมาตั้งแต่อดีตถึงปัจจุบัน ถูกออกแบบมาเพื่อให้อุปกรณ์จากผู้ผลิตต่างกันสามารถทำงานร่วมกันได้ โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์(Electronic Industries Association) เป็นผู้ประกาศใช้

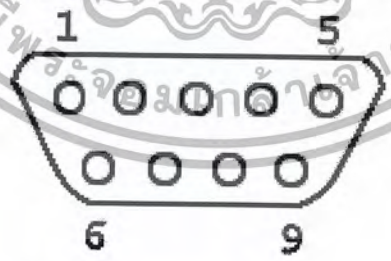
มาตรฐาน RS-232C ได้แบ่งอุปกรณ์ออกเป็น 2 ประเภท คือ

1.DTE (Data Terminal Equipment) เป็นอุปกรณ์สำหรับส่งข้อมูล (เอาต์พุต)

2. DCE (Data Communication Equipment) เป็นอุปกรณ์สำหรับรับข้อมูล (อินพุต)

ตามมาตรฐาน RS-232C แล้วคอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งคอนเน็กเตอรที่นิยมใช้อยู่ในปัจจุบันจะเป็นชนิด D-Type แบบ 9 ขา โดยจะติดอยู่หลังเครื่องคอมพิวเตอร์

3.1.2 ลักษณะของคอนเน็กเตอร์แบบ D-Type



รูปที่ 3-1 แสดงขาของพอร์ตอนุกรม

D-Type 9 Pin	สัญลักษณ์	ชื่อสัญญาณ	รายละเอียด
Pin 3	TD	Transmit Data	ใช้ส่งข้อมูลออกคอมพิวเตอร์
Pin 2	RD	Receive	ใช้รับข้อมูลเข้าคอมพิวเตอร์

Pin 7	RTS	Request To Send	ส่งคำขอให้อุปกรณ์ปลายทางส่งค่ากลับมา
Pin 8	CTS	Clear To Send	ตรวจสอบว่าอุปกรณ์พร้อมที่จะรับข้อมูลหรือไม่
Pin 6	DSR	Data Set Ready	ตรวจสอบการเชื่อมต่อระหว่างคอมฯและอุปกรณ์
Pin 5	SG	Signal Ground	กราวน์ของระบบ
Pin 1	CD	Carrier Detect	ขานี้จะActiveเมื่อมีการส่งสัญญาณCarrierจากโมเด็ม
Pin 4	DTR	Data Terminal Ready	บอกอุปกรณ์ปลายทางว่าต้องการติดต่อด้วย
Pin 9	RI	Ring Indicator	ขานี้จะActiveเมื่อโมเด็มได้รับสัญญาณเรียกเข้าจากสายโทรศัพท์

ตารางที่ 3-1 แสดงการใช้งานของขา Serial Port

3.1.3 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม

เนื่องจากการส่งสัญญาณและการรับสัญญาณจะใช้สายสัญญาณเพียงเส้นเดียว ดังนั้นข้อมูลที่อ่านได้ในแต่ละบิต จึงต้องถูกแยกแยะว่าใช้สำหรับวัตถุประสงค์ใด

1. Start Bit หรือบิตเริ่มต้น มีขนาด 1 บิต จะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝั่งรับว่าข้อมูลกำลังจะมาถึง
2. Data Character หรือบิตข้อมูล มีขนาด 7 หรือ 8 บิต ใช้สำหรับส่งข้อมูลแบบ ASCII Word
3. Parity Bit หรือบิตพาริตี้ ขนาด 1 บิต ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง ทั้งตัวรับและตัวส่งจะต้องรู้ว่าใช้พาริตี้แบบไหน
   - พาริตี้คู่ (Even Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุกๆบิตของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นจำนวนคู่ เช่น 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 0
   - พาริตี้คี่ (Odd Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุกๆบิตของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นจำนวนคี่ เช่น 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 1
   - ไม่มีพาริตี้(None) ถ้าตั้งบิตพาริตี้เป็น None ทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบบิตพาริตี้
4. Stop Bit หรือบิตจบ ใช้ปิดท้ายข้อมูล

3.1.4 อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม

การที่อุปกรณ์ 2 อย่างจะติดต่อกันได้นั้น จะต้องทำงานด้วยอัตราเร็วเท่ากัน ซึ่งอัตราเร็วในการสื่อสารคือค่าบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที สำหรับมาตรฐานแบบ RS-232C มีใช้ดังนี้ 110, 150, 300, 600, 1200,2400,4800,9600 และ 19200 บิตต่อวินาที

3.2 Microcontroller เขียนโปรแกรมควบคุม 8051 ผ่าน Serial Port

3.2.1 จัดการทำงาน Timer

TCON(Time Control Register) และ TMOD(Time Mode Register) จะนำเอาทั้งสอง มาทำหน้าที่สำหรับควบคุมการทำงานของ Timer เพื่อจะนำมาใช้เป็นตัววนลูปจับเวลาของ อัตราการรับส่งข้อมูล(Baud Rate) คือจะกำหนดค่าใน TMOD เช่น TMOD=00100000B ในที่นี้เราใช้รับส่งข้อมูล 11 bit คือจะมี 8 data bit , Start bit, Stop bit, Custom data bit ทั้งนี้เราจะกำหนดไม่ให้ Timer เกิด Interrupt โดยเราจะไปกำหนดที่ IE(Interrupt Enable Register)ตั้งค่าให้เป็นศูนย์ทุก bit ส่วน Baud rate ของการรับส่งข้อมูลนั้นจะอาศัยอัตราการเกิด Timer Overflow ร่วมกับค่าของ SMOD(Serial Mode Register) สำหรับค่าที่จะกำหนดให้ Timer เพื่อสร้างอัตรารับส่งของ Serial Port ในไมโครคอนโทรลเลอร์ สามารถใช้จากตารางด้านล่างนี้

ความถี่(MHz)	อัตรารับส่ง Serial Port(bit/sec)	ค่าของ Timer ที่จะกำหนด
11.059	1200	E8H
11.059	2400	F4H
11.059	4800	FAH
11.059	9600	FDH

ตารางที่ 3-2 แสดงการกำหนดค่าให้ Timer เพื่อใช้เป็น Baud Rate

3.2.2 การเขียนโปรแกรมตั้งค่าทำงานของ Timer

อาจสรุปการเขียนโปรแกรมตั้งค่าของ Timer เช่น Baud Rate ของ Serial Port = 9600 Bit/Sec

```
MOV TMOD,#00100000B ; ตั้งค่าใน TMOD Register ให้เริ่มค่าใหม่อัตโนมัติ
MOV IE,#00000000B   ; ตั้งค่าไม่ให้ Timer Interrupt
MOV TL1,#FDH        ; ตั้งค่าอัตรารับส่งข้อมูล 9600 bit/sec
MOV TH1,#FDH
SETB TR1            ; สั่ง Timer 1 ทำงาน
```

3.2.3 เขียนโปรแกรมติดต่อกับ Serial Port

Serial Port Control Register ชื่อว่า SCON ทำหน้าที่เป็นตัวควบคุมกระบวนการรับส่งข้อมูลของ Serial Port ในไมโครคอนโทรลเลอร์ว่าจะให้เป็น โหมดใดการกำหนดให้รับข้อมูล และรับส่งข้อมูล SCON นี้ จะคอยจัดการให้ แต่เราต้องกำหนดหน้าที่ ให้เจ้า SCON ก่อน ดังรายละเอียดในตารางต่อไปนี้

ตำแหน่ง Bit	ชื่อ	หน้าที่ทำงาน
0	RI	Interrupt ในการส่งข้อมูล
1	TI	Interrupt ในการรับข้อมูล
2	TR8	รับข้อมูลหากใช้ Mode 2,3 จะเป็น Bit No. 9
3	TB8	ส่งข้อมูลหากใช้ Mode 2,3 จะเป็น Bit No. 9
4	REN	กำหนดให้รับข้อมูลหรือไม่
5	SM2	ถ้าใช้ต่อไมโครคอนโทรลเลอร์ 2 ตัวขึ้นไป
6	SM1	เลือก Mode ของพอร์ต อนุกรม
7	SM0	เลือก Mode ของพอร์ตอนุกรม

ตารางที่ 3.3 แสดงส่วนประกอบของ SCON

Serial Port Buffer Register ชื่อว่า SBUF ซึ่งเป็นตัวเก็บข้อมูลสำหรับรับหรือส่งข้อมูลผ่าน Serial Port ในตัวไมโครคอนโทรลเลอร์ หากเราจะเขียน โปรแกรมรับข้อมูลก็ต้องไปเคลื่อนย้ายข้อมูลใน Buffer มาใช้ เช่นใช้คำสั่ง MOV A,SBUF ถ้าหากเราจะส่งข้อมูลออกไปที่ Serial Port ในไมโครคอนโทรลเลอร์ ก็จะต้องเคลื่อนย้ายข้อมูลไปไว้ที่ Buffer ตัวนี้เหมือนกัน เช่น MOV SBUF,A หรือ MOV SBUF,#31H

3.2.4 การเขียนโปรแกรมรับข้อมูล

```
JNB RI,$      ; เปรียบเทียบเงื่อนไขจนกว่าเป็นจริงเมื่อ Buffer Receive เรียบร้อยแล้ว
CLR RI        ; เคลียร์ค่าของ RI(Receive Interrupt) bit ใน SCON Register
MOV A,SBUF ; เก็บค่าจาก Buffer ไปเก็บไว้ใน Register A
```

3.2.5 การเขียนโปรแกรมส่งข้อมูล

```
MOV SBUF,#43H ; เอาค่าแอสกี้=43H ไปเก็บไว้ที่ Buffer ในการส่งข้อมูลผ่าน Serial Port
JNB TI,$      ; เปรียบเทียบเงื่อนไขเป็นจริงเมื่อ Buffer Send เรียบร้อยแล้ว
CLR TI        ; เคลียร์ค่าของ TI (Transmit Interrupt) bit ใน SCON Register
```

3.2.6 เขียนโปรแกรมควบคุม Microcontroller 8051 ผ่าน Serial Port

จะนำเอาหลักการที่เสนอไปมาประยุกต์เพื่อติดต่อแบบ RS-232 กับ PC ผ่านทาง Serial Port โดยจะมีส่วนของโปรแกรมที่ตัวไมโครคอนโทรลเลอร์และโปรแกรมบน PC ซึ่งใช้ Visual Basic สำหรับอุปกรณ์ด้าน Hardware นั้น ก็เป็นวงจรการต่อไมโครคอนโทรลเลอร์แบบพื้นฐาน มีเพิ่มในส่วนของ ชุดติดต่อ RS-232 ซึ่งจะใช้ ICL232 และชุดแสดง LED แสดงผล คอนเนคเตอร์ติดต่อ Serial Port กับ PC เกี่ยวกับตัวไมโครคอนโทรลเลอร์ตัวนี้ จะมีหน่วยความจำภายในตัวคอนโทรลเลอร์เอง ซึ่งสามารถใช้เครื่อง write โปรแกรมๆที่คอมไพล์แล้วได้เลย

3.2.6.1 โปรแกรมควบคุมไมโครคอนโทรลเลอร์ 8051

```
ORG 0000H ; กำหนดตั้งค่าให้กับ Timer1 เพื่อใช้สร้าง Baud Rate ในการรับส่งข้อมูล
MOV IE,#00000000B
MOV TMOD,#00100000
MOV TL1#,0FDH
MOV TH1,#0FDH ; กำหนดค่าให้กับ SCON Register โดยเลือกโหมด 1 และให้รับข้อมูล REN=1
MOV SCON,#01010000B
SETB TR1
MOV P0,#00000000B ; เซทค่าให้พอร์ตขนาน P0 เป็น 0 ทุกบิต
                  ; เข้าโปรแกรมหลัก
INDEX:  ACALL SUB_RXD ; ไปรับข้อมูลจาก Serial Port ที่ส่งมาจาก PC
        ACALL LEDOFF  ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าที่ออกพอร์ตดับหมด
        ACALL LED1    ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าออกพอร์ตP0.0
        ACALL LED2    ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าออกพอร์ตP0.1
        ACALL LED3    ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าออกพอร์ตP0.2
        ACALL LED4    ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าออกพอร์ตP0.3
        ACALL LED5    ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าออกพอร์ตP0.4
```

```
        ACALL LED6      ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าออกพอร์ตP0.5
        ACALL LED7      ; รูทีนกำหนดให้ไมโครคอนโทรลเลอร์ส่งค่าออกพอร์ตP0.6
        SJMP INDEX      ; กลับไปทำโปรแกรมหลัก
                        ; รูทีนรับข้อมูลทาง Serial Port
RXD : JNB RI,$
CLR RI
MOV A,SBUF
RET
LEDOFF : CJNE A,#30H,NEXT ; เปรียบเทียบเงื่อนไขถ้าเป็นจริงให้ทำขั้นตอนต่อไป
MOV P0,#00000000B       ; จะส่งค่าออกพอร์ตขนาน P0 ให้ LED ดับหมด
ACALL SUB_TXD           ; ไปทำขั้นตอนส่งข้อมูลให้ PC รู้ว่าคำสั่งทำงานเรียบร้อยแล้ว
SJMP INDEX              ; กลับสู่โปรแกรมหลัก
                        ; รูทีน LED2-LED7 จะทำงานหลักการเดียวกับ LED1 แต่จะส่งค่า
ออกพอร์ตขนานเรียงกันไป
LED1:CJNE A,#31H,NEXT   ; เปรียบเทียบเงื่อนไขถ้าเป็นจริงให้ทำขั้นตอนต่อไป
                        ; จะส่งค่าออกพอร์ตขนาน P0.0 โดยจะกลับค่า 0 เป็น 1 คือ
LED1 ติด และกลับ 1 เป็น 0 คือ LED1 ดับ
CPL P0.0                ; คำสั่ง Boolean = Not ใช้ควบคุมสถานะบิตของพอร์ต
ACALL SUB_TXD ; ไปทำขั้นตอนส่งข้อมูลให้ PC รู้ว่าคำสั่งทำงานเรียบร้อยแล้ว
SJMP INDEX      ; กลับสู่โปรแกรมหลัก
                ; รูทีน LED2-LED7 จะทำงานหลักการเดียวกับ LED1 แต่จะส่งค่าออกพอร์ต
ขนานเรียงกันไป
LED2:
        CJNE A,#32H,NEXT
        CPL P0.1
        ACALL SUB_TXD
        SJMP INDEX
LED3:
        CJNE A,#33H,NEXT
        CPL P0.2
```

```
        ACALL SUB_TXD
        SJMP INDEX
LED4:
        CJNE A,#34H,NEXT
        CPL P0.3
        ACALL SUB_TXD
        SJMP INDEX
LED5:
        CJNE A,#35H,NEXT
        CPL P0.4
        ACALL SUB_TXD
        SJMP INDEX
LED6:
        CJNE A,#36H,NEXT
        CPL P0.5
        ACALL SUB_TXD
        SJMP INDEX
LED7:
        CJNE A,#37H,NEXT
        CPL P0.6
        ACALL SUB_TXD
        SJMP INDEX
NEXT: RET ; รูทีนส่งข้อมูลทาง Serial Port เพื่อบอกให้โปรแกรมที่ PC รู้ว่าส่งคำสั่งทำงานเรียบร้อยแล้ว
TXD :   MOV SBUF,#43
        JNB TI,$
        CLR TI
        RET
        END
```

3.3 การรับข้อความจาก Serial port เก็บลงหน่วยความจำภายใน

1. ตั้งค่าตำแหน่ง Address ที่เริ่มใช้ของหน่วยความจำภายใน อาจจะกำหนดตอนเริ่มโปรแกรม

```
MOV R1,#20H
```

2. ในส่วนของการรับข้อมูลจาก Serial port โดยเราจะสร้างโปรแกรมย่อยชื่อ GetData

```
GetData : JNB  RI,$  ;  เปรียบเทียบเงื่อนไขจนกว่าเป็นจริงเมื่อ Buffer Receive data เรียบร้อยแล้ว
        CLR R ;  เคลียร์ค่าใน Register RI ในการที่จะไปรับข้อมูลใหม่
        MOV A,SBUF ; เคลื่อนย้ายค่าใน Buffer SBUF register ไปเก็บไว้ที่ register A
```

3. นำค่าที่รับจาก register A มาเก็บไว้ในหน่วยความจำภายใน

```
MOV @R1,A ; เคลื่อนย้ายข้อมูลไปเก็บไว้ในหน่วยความจำ ณ ที่ตำแหน่งที่ชี้ด้วยค่าใน R1(#20H)
INC R1 ; เพิ่มค่าใน Register R1
CJNE A,ODH,GetData ;เปรียบเทียบเงื่อนไขถ้าหากใน Register A  ว่ามีค่าแอสกี้ = ODH ซึ่งก็คือการ Enter สำหรับข้อความใน 1 บรรทัดที่ส่งมาจากโปรแกรมของคอมพิวเตอร์ หากไม่ใช่ก็กลับไป รับข้อมูลมาเพิ่มใหม่
MOV R1,#25H ; ตั้งค่าเริ่มต้นใน R1 ใหม่สำหรับข้อมูลที่จะนำมาเก็บชุดต่อไป
```

3.4 การส่งข้อความจากหน่วยความจำผ่าน Serial port

1. ตั้งค่าตัวแปรสำหรับกำหนดเป็นรหัส Ascii เพื่อเริ่มต้นตำแหน่งเคอเซอร์ซ้ายมือในโปรแกรมบนคอมพิวเตอร์ ในที่นี้สมมุติชื่อ ST และตัวแปร NL สำหรับบอกให้ขึ้นบรรทัดใหม่ในโปรแกรมบน PC เช่นกัน ซึ่งสามารถกำหนดดังต่อไปนี้

```
ST  EQU  ODH
NL  EQU  OAH
```

2.ตั้งค่าสำหรับกำหนดพื้นที่ในหน่วยความจำ

```
DATA : DS 20H ; กำหนดใช้พื้นที่ในหน่วยความจำไว้ที่ 32  byte
```

3. ตั้งค่าใน Register DPTR ในการที่จะเริ่มตำแหน่ง Address ในหน่วยความจำ สำหรับจะเอาไว้ เก็บข้อมูลหรือข้อความที่ส่งไปยังคอมพิวเตอร์

```
MOV  DPTR,#GiveWord
```

4. เคลียร์ค่าใน Register A โดยกำหนดค่า #00H และเคลื่อนย้ายข้อมูลที่เก็บในหน่วยความจำไปที่ Register A เพื่อจะในไปส่งข้อมูลผ่าน Serial port ให้กับคอมพิวเตอร์ต่อไป แนะนำว่าควรกำหนดเป็นโปรแกรมย่อยขึ้นใหม่ โดยอาจใช้ชื่อ Send_Data

Send_Data : MOV A,#00

MOV A,@A,+DPTR ; ในส่วนนี้จะเป็นการส่งข้อมูลผ่านออก serial port

MOV SBUF,A ; เอาค่าใน Register A ไปเก็บไว้ที่ Buffer เพื่อส่งข้อมูลผ่าน Serial port

JNB TI,$; เปรียบเทียบเงื่อนไขเป็นจริงเมื่อ Buffer Send Data เรียบร้อยแล้ว

CLR TI ; เคลียร์ค่าของ TI(Transmit Interrupt) bit ใน SCON Register เพื่อคอยส่งข้อมูลชุดใหม่ออกไป

JZ HOME ; ตรวจสอบข้อมูลหากพบว่าเป็น 00H แสดงว่าข้อมูลหรือข้อความซึ่งถูกเก็บในหน่วยความจำที่เขียนไว้ถูกส่งไปหมดแล้ว

INC DPTR ; หากยังไม่หมดข้อความก็ให้ส่งข้อมูลที่ตำแหน่ง Address ในลำดับต่อไป

SJMP Send_Data ; กลับไปทำขั้นตอนส่งข้อมูล

5.กำหนดโปรแกรมย่อยสำหรับการเก็บข้อมูลในหน่วยความจำที่จะส่งไปยังคอมพิวเตอร์ โดยอาจสมมุติชื่อ GiveWord ซึ่งจะมีข้อมูลที่ส่งเริ่มต้นจะบอกด้วยค่าแอสกี้ ก็คือ ST กับ NL ที่กำหนดไว้ตอนบนของ โปรแกรมนี้แล้วนั้น ก็ เพื่อจะให้โปรแกรมบนคอมพิวเตอร์รับรู้แล้วขึ้นบรรทัดใหม่ ต่อจากนั้นก็จะเป็น ข้อความ ปิดท้ายด้วยค่าแอสกี้ ST กับ NL และค่าเพื่อใช้ในการตรวจสอบว่าส่งข้อมูลในหน่วย ความจำ ไปหมดแล้วในที่นี้คือ 00H

```
GiveWord : DB  ST,NL,"MICROCONTROLLER INTERFACE SERAIL PORT"
           DB  " BY MECHANICAL DEPARTMENT",ST,NL
           DB  " SEND DATA TO COMPUTER COMPLETED! " ,ST,NL
           DB  00H
```

บทที่ 4
โปรแกรมวิชวลเบสิค 6

วิชวลเบสิก (Visual Basic) เป็นโปรแกรมที่ใช้สร้างโปรแกรมประยุกต์ สำหรับระบบปฏิบัติการวินโดว์ (Windows) เนื่องจากความง่ายของภาษาที่ใช้ในการเขียนและวิชวลเบสิกเป็นโปรแกรมที่เน้นในการสร้างแอพพลิเคชั่นเพื่อติดต่อกับผู้ใช้งานเป็นหลักทำให้ได้ผลงานออกมารวดเร็วสวยงาม และนอกจากนี้วิชวลเบสิกยังมีเครื่องมือที่ใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมซึ่งนำไปใช้ในการควบคุมเครื่องอีกด้วย

4.1 ขั้นตอนการสร้างโปรแกรมประยุกต์

การสร้างโปรแกรมประยุกต์วิชวลเบสิก ประกอบด้วยขั้นตอนหลัก 3 ขั้นตอน คือ

1. การสร้างอินเตอร์เฟส (Interface) โดยมีฟอร์ม (Form) เป็นอ๊อบเจค (Object) พื้นฐานและเป็นที่วางตัวคอนโทรล (Control) สำหรับการติดต่อกับผู้ใช้
2. ตั้งค่าคุณสมบัติ เป็นการกำหนดพฤติกรรมและการทำงานให้กับอ๊อบเจคต่างๆ
3. การเขียนคำสั่ง เป็นการควบคุมการประมวลผลผ่านโพรซีเจอร์ (Procedure) ที่กำหนด

4.2 คอนโทรลพื้นฐาน

กลุ่มคอนโทรลพื้นฐานของวิชวลเบสิกประกอบด้วย Text Box, Label, Command Button, Picture Box, Image, Check Box, Frame เป็นต้น

ไอคอน	ชื่อตัว Control	ชื่อ Class	คำอธิบาย
	Check box	CheckBox	ใช้กับการเลือกแบบ ถูก/ผิด (True/False, Yes/No)
	Combo box	ComboBox	เป็นตัว Control เป็นการผสมระหว่าง Text box กับ List box ซึ่งจะปรากฏรายการ เมื่อมีการคลิกลูกศร และ Combo box ไม่สนับสนุนการเลือกแบบหลายค่า
	Command button	CommandButton	ปุ่มคำสั่งเป็นตัว Control ที่ใช้ในทุกฟอร์ม ตามปกติจะเขียนคำสั่งใน Click Event Procedure ของตัว Control นี้
	Data	Data	เป็นตัว Control ที่สามารถรวมข้อมูลกับฐานข้อมูลได้ และเป็นส่วนที่ Visual Basic ให้ผู้ใช้สามารถติดต่อระหว่างตัว Control บนฟอร์มกับฟิลด์ใน Table ของฐานข้อมูล โดย Data จะทำงานกับ Database Jet

			ของฐานข้อมูล แต่ไม่สามารถทำงานกับ ActiveX Data Object (ADO) ได้
	Directory List box	DirListBox	เป็น List box แบบหนึ่ง ที่แสดงไดเรคทอรีและพาร์ทที่เลือก
	Drive List box	DriveListBox	คล้ายกับ Combo box ที่ใช้เลือกชื่อของไดร์ฟในระบบ
	File list box	FileListBox	เป็น List box ชนิดพิเศษที่ใช้แสดงชื่อไฟล์ในไดเรคทอรี
	Frame	Frame	สามารถใช้เป็น Container สำหรับตัว control อื่น
 	Horizontal และ Vertical Scroll Bar	HScrollBar และ VScrollBar	ใช้เป็นแถบเลื่อนแบบ Stand-alone แต่มักจะไม่ค่อยมีการใช้ เพราะตัว Control อื่น ๆ ส่วนใหญ่ จะมีแถบเลื่อนของตัวเอง แถบเลื่อนแบบ Stand-alone อาจจะใช้ในลักษณะ Slider ได้
	Image	Image	เป็นตัว Control ใช้เก็บภาพคล้ายกับ Picture Box แต่ไม่สามารถทำงานแบบ Container ได้ Image มีข้อดีที่ใช้ทรัพยากรของระบบน้อยกว่า Picture Box
	Label	Label	เป็นตัว Control ที่ใช้แสดงข้อความ หรือป้ายชื่อ
	Line	Line	เป็นตัว Control ใช้สำหรับการตกแต่งด้านกราฟฟิก
	List box	ListBox	เป็นตัว Control ที่เก็บรายการของค่า และให้ผู้ใช้เลือก ซึ่งสามารถเป็นการเลือกค่าเดียวหรือหลายค่า ขึ้นกับการกำหนดคุณสมบัติ MultiSelect
	OLE container	OLE	เป็นตัว Control ที่สามารถเป็น Host Window ให้กับโปรแกรมภายนอก เช่น Microsoft Excel หรืออาจจะกล่าวว่าเป็นการสร้าง Window ให้กับโปรแกรมอื่นบนโปรแกรมประยุกต์ Visual Basic
	Option button	OptionButton	เป็นตัว Control ใช้กับกลุ่มตัว Control โดยให้เลือกได้เพียงตัว Control เดียวต่อครั้งหนึ่ง เมื่อมีการเลือกตัว Control ในกลุ่มแล้ว ตัว Control อื่นในกลุ่มจะเปลี่ยนจากการเลือกโดยอัตโนมัติ ถ้ามีการใช้ Option Button

			มากกว่า 2 กลุ่ม ต้องวางแต่ละกลุ่มใน Container เช่น Frame
	Picture box	PictureBox	ใช้แสดงภาพในฟอร์แมต BMP, DIB (Bitmap), ไอคอน (Icon), WMF (Metafile), GIF และ JPEG เป็นต้น และสามารถใช้เป็น container สำหรับตัว Control อื่น
	Shape	Shape	เป็นตัว Control ใช้สำหรับการตกแต่งด้านกราฟฟิก
	Text box	TextBox	เป็นตัว Control ที่เป็นฟิลด์ ใช้เก็บตัวอักษรที่สามารถแก้ไขโดยผู้ใช้ได้ และได้รับการใช้งานมาก
	Timer	Timer	เป็นตัว Control พิเศษที่ไม่เห็นเมื่อเวลาเรียกใช้ วัตถุประสงค์การใช้คือการสร้าง Event ในฟอร์มแม่ โดยการเขียนคำสั่งใน Procedure ที่เจาะจงสำหรับการทำงานเบื้องหลัง เช่น การตรวจสถานะของอุปกรณ์ต่อพ่วง

ตารางที่ 4-1 แสดงถึงคอนโทรลพื้นฐานของวิชวลเบสิกที่ใช้ในการสร้างโปรแกรม

4.3 คุณสมบัติร่วม

1. Left, Top, Width, Height ใช้จัดตำแหน่งของอ๊อบเจค เช่น การวางฟอร์มที่มุมซ้ายบน มีความกว้าง 5000 Twips และสูง 3000 Twips

```
Form1.Left = 0
Form1.Top = 0
Form1.Width = 5000
Form1.Height = 3000
```

2. ForeColor และ BackColor ใช้กำหนดสีของข้อความและสีพื้นหลังของอ๊อบเจค ซึ่งสามารถกำหนดสีเป็นแบบ Palette และ system เช่น

```
กำหนดสีแบบ Palette     Label1.ForeColor = vbHighlightText
กำหนดสีแบบ System      Label1.BackColor = &H800000D
```

รวมถึงการกำหนดด้วยค่าคงที่, เลขฐานสิบ และเลขฐานสิบหก เช่นตัวอย่างเป็นสีเดียวกัน

ค่าคงที่	Text1.BackColor = vbCyan
เลขฐานสิบ	Text1.BackColor = 16776960
เลขฐานสิบหก	Text1.BackColor = &HFFF00

3. Font ใช้กำหนดลักษณะการแสดงตัวอักษร การตั้งค่าคุณสมบัติ Size, Bold, Italic, Underline และ Strikethrough เช่น

```
Text1.Font.Name = "Arial"
Text1.Font.Size = 12
Text1.Font.Bold = True
```

4. Caption, Text ใช้ในการแสดงข้อความ โดยคุณสมบัติ Caption เป็นข้อความที่ปรากฏภายในตัวคอนโทรล และไม่สามารถแก้ไขได้โดยผู้ใช้เมื่อเรียกใช้โปรแกรม เช่น

```
Label1.Caption = "Title"
Text1.Text = "Hello Word"
```

5. Enabled, Locked และ Visible โดยคุณสมบัติ Visible ใช้กำหนดการมองเห็น คุณสมบัติ Enabled และ Locked กำหนดการเข้าถึงตัว Control แต่ต่างกันที่ คุณสมบัติ Enabled ถ้ากำหนดไม่ให้เข้าถึง (Enabled = False) แล้ว ตัว Control ไม่สามารถรับโฟกัสได้ ส่วนคุณสมบัติ Locked ถ้ากำหนดไม่ให้เข้าถึง (Locked = True) แล้ว ตัว Control ยังสามารถรับโฟกัสได้

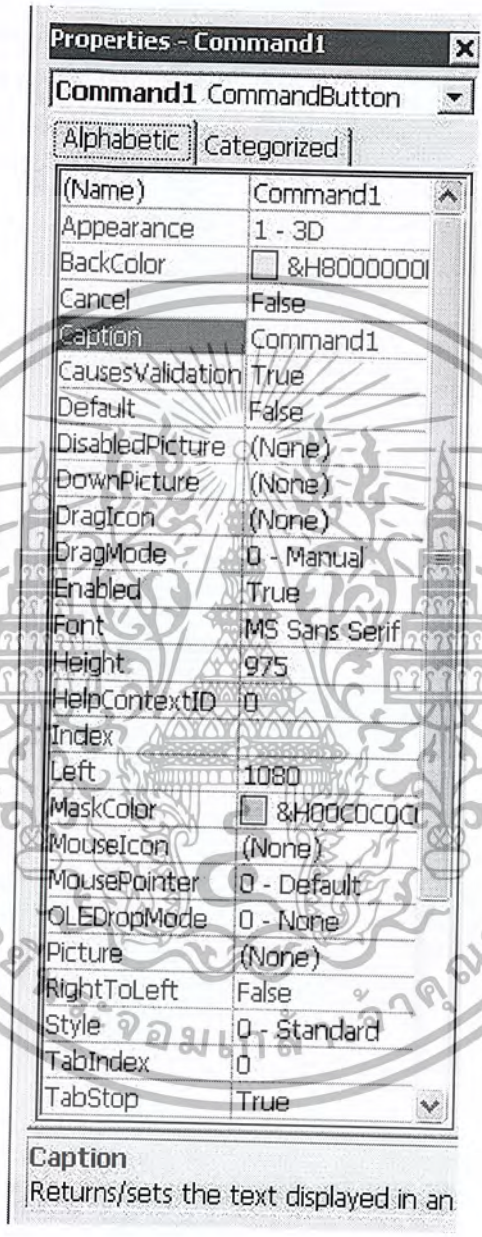
6. MousePointer และ MouseIcon ใช้กำหนดไอคอนของเมาส์เมื่อเคลื่อนที่ผ่านตัวคอนโทรล เช่น

```
Text1.MousePointer = vbCrosshair
```

7. คุณสมบัติอื่นๆ ได้แก่

- คุณสมบัติ Value มีค่าตามชนิดของตัวคอนโทรล
- คุณสมบัติ Index ใช้ในการสร้าง array ของตัวคอนโทรลและมีคุณสมบัติแบบอ่านอย่างเดียว
- คุณสมบัติ Appearance เป็นลักษณะหน้าตาของคอนโทรล ตามปกติจะกำหนดค่าคุณสมบัติเริ่มต้นเป็น 3 มิติ การกำหนดค่าทำได้ในเวลาออกแบบ และเป็นแบบอ่านอย่างเดียว เมื่อมีการเรียกใช้
- คุณสมบัติ Border Style คือคุณสมบัติเส้นขอบ มีคอนโทรลสนับสนุน ได้แก่ Text box, Label, Frame, Picture box, Image และ OLE โดยถ้าตั้งค่าเป็น 0-none จะไม่มีเส้นขอบ ถ้าตั้งค่าเป็น 1-Fixed จะมีเส้นขอบ

- คุณสมบัติ ToolTip เป็นหน้าต่างสีเหลืองขนาดเล็กแสดงคำอธิบายสั้นๆ ซึ่งปรากฏขึ้นเมื่อเมาส์เคลื่อนที่ไปอยู่เหนือตัว control หรือไอคอน



ตารางที่ 4-2 แสดงถึงการกำหนดคุณสมของคอนโทรลที่ใช้ในการสร้างโปรแกรม

4.4 เมธอดร่วม

1. Move ใช้ในการย้ายอ๊อบเจค หรือตัวคอนโทรล พร้อมทั้งปรับขนาดในเวลาเดียวกัน มีไวยากรณ์ คือ

   [object.]Move Left [, Top, Width, Height]

2. Refresh ใช้ในการวาดตัวคอนโทรลใหม่ ตามปกติไม่ใช้เนื่องจาก Visual Basic มีการปรับค่าโดยอัตโนมัติ แต่สามารถใช้ได้ในกรณีที่ต้องการให้มีการปรับคุณสมบัติทันที เช่น

```
Dim i As Integer
For i = 1000 to 1 Step - 1
Label1.Caption = (str(i))
Label1.Refresh
Next
```

3. SetFocus ใช้ย้ายโฟกัสไปที่คอนโทรลตัวที่ระบุ เช่น

```
Text1.SetFocus
```

4.5 อีเวนต์ร่วม

1. Click และ DblClick Event โดย Click Event จะเกิดเมื่อผู้ใช้กดปุ่มเมาส์ด้านซ้ายบนตัวคอนโทรล และ DblClick Event เกิดเมื่อเป็นการปุ่มเมาส์ด้านซ้ายแบบดับเบิลคลิก

2. Change Event เป็นอีเวนต์พื้นฐานที่เกิดเมื่อข้อมูลของตัวคอนโทรล มีการเปลี่ยนแปลง

3. GotFocus และ LostFocus Event ซึ่ง GotFocus Event เกิดขึ้นเมื่อตัวคอนโทรล ได้รับการโฟกัส เช่นเมื่อเมาส์เลื่อนมาถึง และ LostFocus Event จะเกิดเมื่อการโฟกัสออกไปจากตัวคอนโทรล

4. KeyPress, KeyDown และ KeyUp Event เป็นอีเวนต์ ที่เกิดเพื่อกดปุ่มแป้นพิมพ์ ขณะที่ตัวคอนโทรลได้รับการโฟกัส โดยมีลำดับดังนี้ KeyDown (เมื่อผู้ใช้กดปุ่มแป้นพิมพ์) KeyPress (Visual Basic แปลปุ่มนั้นเป็นรหัส ANSI) และ KeyUp (เมื่อปล่อยปุ่มแป้นจากพิมพ์)

5. MouseDown, MouseUp และ MouseMove Event เป็นอีเวนต์ ที่เกิดขึ้นเมื่อมีการคลิก, ปล่อย หรือย้ายของเมาส์บนตัว Control

โดยลำดับการเกิด Click Event มีลำดับ คือ MouseDown -> MouseUp -> Click -> MouseMove ลำดับการเกิด DblClick Event มีลำดับ คือ MouseDown -> MouseUp -> Click -> MouseMove -> DblClick -> MouseUp -> MouseUp

4.6 อีเวนต์ของฟอร์ม

อีเวนต์ของฟอร์มเป็นอ๊อบเจคพื้นที่ใช้ในการติดต่อกับผู้ใช้งาน ดังนี้

1. Load Event เป็นอีเวนต์ที่เกิดขึ้นต่อจาก Initialize Event ใช้ในการกำหนดค่าเริ่มต้นให้กับฟอร์มและอ๊อปเจคค์

2. Resize Event เกิดขึ้นก่อนที่จะเห็นฟอร์ม โดยอีเวนต์สามารถใช้ในการปรับตัว คอนโทรลให้พอดีกับฟอร์ม นอกจาก Resize event เกิดขึ้นผู้ใช้ปรับขนาดฟอร์มเอง

3. Unload Event เป็นช่วงสุดท้ายก่อนที่โปรแกรมจะถูกปิด อีเวนต์นี้เกิดเมื่อผู้ใช้ปิดโปรแกรมก่อนที่โปรแกรมจะถูกปิด เราสามารถแจ้งข่าวสารให้ผู้ใช้ได้ เช่น ถามถึงการบันทึกข้อมูลก่อนออกจากโปรแกรม ถ้าไม่มีการยกเลิกการ Unload ขั้นต่อไป วิชวลเบสิกจะทำการยกเลิกตัว คอนโทรล ปิดฟอร์ม และยกเลิกทรัพยากรต่าง ๆ ของวินโดว์

4.7 โพรซีเจอร์และฟังก์ชั่น

โพรซีเจอร์หรือฟังก์ชั่นเป็นส่วนที่ใช้เขียนโปรแกรมลงไป ซึ่งสามารถช่วยลดความซ้ำซ้อนของโปรแกรมแต่ละส่วน สำหรับโพรซีเจอร์สามารถแบ่งได้เป็น 2 ประเภท คือ

1. โพรซีเจอร์เหตุการณ์ จะทำงานเมื่อเกิดเหตุการณ์ต่างๆขึ้นกับอ็อปเจ็ค เช่น เมื่อผู้ใช้คลิกปุ่ม "Start" ถ้าหากภายในปุ่มมีโพรซีเจอร์อยู่ ก็จะเกิดการทำงานตามคำสั่งในโพรซีเจอร์นั้นทันที

2. โพรซีเจอร์ที่กำหนดเอง เป็นโพรซีเจอร์แบบทั่วไป หลังจากสร้างขึ้นแล้วเราสามารถเรียกใช้ได้ทันทีเพียงแค่อ้างชื่อของโพรซีเจอร์นั้นเท่านั้น

4.8 ฟังก์ชั่น

ลักษณะของฟังก์ชั่นจะคล้ายกับโพรซีเจอร์ เพียงแต่ฟังก์ชั่นจะมีการส่งค่ากลับไปยังโปรแกรมที่เรียกฟังก์ชั่นนั้นมาใช้

4.9 การประกาศตัวแปร

การประกาศตัวแปรมีคำสั่งแตกต่างกันไปตามชนิดของการใช้งานดังนี้

- Static : ตัวแปรนี้จะใช้ได้เฉพาะภายในโพรซีเจอร์เท่านั้นและค่าในตัวแปรจะถูกเก็บไว้ตลอดเวลา แม้ว่าจะออกโพรซีเจอร์ไปแล้วก็ตาม
- Dim : มีลักษณะเป็น Static แต่สามารถประกาศตัวแปรได้ 2 ที่ คือ ในโพรซีเจอร์ และ General Declarations (ส่วนแรกของโปรแกรม ใช้ในการประกาศตัวแปร) หากประกาศในโพรซีเจอร์แล้ว เมื่อออกจากโพรซีเจอร์แล้ว ค่าในตัวแปรจะถูกรีเซ็ทใหม่อัตโนมัติ ดังนั้นในแต่ละโพรซีเจอร์อาจมีตัวแปรชื่อซ้ำกันได้ โดยไม่มีผลต่อกัน แต่การประกาศตัวแปรบน General Declarations จะทำให้โพรซีเจอร์ทุกตัวบนฟอร์มนั้นสามารถเรียกใช้ได้ และหากมีการเปลี่ยนแปลงค่าในโพรซีเจอร์ใดๆแล้ว ก็จะมีผลไปกับโพรซีเจอร์อื่นๆด้วย
- Private : มีรูปแบบเช่นเดียวกันกับ Dim แต่ประกาศได้ที่ General Declarations เท่านั้น และจะใช้ได้เฉพาะบนฟอร์มนั้นเท่านั้น

- Public : มีรูปแบบเช่นเดียวกันกับ Private แต่สามารถใช้ได้ในทุกๆฟอร์มบนโปรแกรมนั้น

ประเภทข้อมูล	ประเภท	ขนาด	การเก็บข้อมูล หรือ ช่วงข้อมูล
Integer	จำนวนเต็ม	2 ไบต์	–32,768 ถึง 32,767
Long	จำนวนเต็ม	4 ไบต์	–2,147,483,648 ถึง 2,147,483,647
Boolean	จำนวนเต็ม	2 ไบต์	เก็บค่า 0 และ –1 ซึ่งแทน False หรือ True
Byte	จำนวนเต็ม	1 ไบต์	เก็บค่าในช่วง 0 ถึง 255
Single	จำนวนทศนิยม	4 ไบต์	ค่าลบ –3.402823E38 ถึง –1.401298E-45 ค่าบวก 1.401298E-45 ถึง 3.402823E38
Double	จำนวนทศนิยม	8 ไบต์	ค่าลบ –1.79769313486232E308 ถึง -4.94065645841247E-324 ค่าบวก 4.94065645841247E-324 ถึง 1.79769313486232E308
Currency	จำนวนทศนิยม (4 ตำแหน่ง)	8 ไบต์	–922,337,203,477.5808 ถึง –922,337,203,477.5807
Decimal	จำนวนทศนิยม	8 ไบต์	ค่าที่ไม่มีทศนิยม +/-79,228,162,514,264,337,593,543,950,335 ค่าที่มีทศนิยม +/- 7.92281625142643 37593543950335 และมีทศนิยม 28 ตำแหน่ง
String	ข้อความ	-	-
Date	วันที่/เวลา	8 ไบต์	เก็บค่าระหว่าง 1 มกราคม ค.ศ. 100 ถึง 31 ธันวาคม ค.ศ. 9999 และเวลาใดๆ Date ใช้ 8 ไบต์เหมือนกับ Double แต่โครงสร้างมีความแตกต่างกัน โดยส่วนจำนวนเต็มเป็นสารสนเทศของวัน และทศนิยมเป็นส่วนของเวลา

ตารางที่ 4-3 แสดงถึงประเภทของข้อมูลพื้นฐานที่ใช้ในการสร้างตัวแปร

4.10 คำสั่งพื้นฐาน

คำสั่งพื้นฐานที่ใช้ในการสร้างโปรแกรม ได้แก่

- คำสั่งเงื่อนไข If... Then... Else... End If มีรูปแบบดังนี้

```
IF
  < เงื่อนไขที่ทำการเปรียบเทียบ >
Then
  < ถ้าเงื่อนไขเป็นจริง เขียนโค้ดที่นี่ >
Else
  < ถ้าเงื่อนไขไม่เป็นจริง เขียนโค้ดที่นี่ >
End If
```

- คำสั่งเงื่อนไข Select Case มีรูปแบบดังนี้

```
Select Case
  < ชื่อตัวแปรที่ต้องหานำไปเปรียบเทียบ >
Case
  < ค่าที่กำหนดสำหรับเปรียบเทียบ >
  < กรณีค่าตัวแปรเปรียบเทียบตรงกับค่าที่กำหนดไว้ จะทำคำสั่งในส่วนนี้ >
Case
  < ค่าอื่นที่ต้องการเปรียบเทียบ >
  < กรณีค่าตัวแปรเปรียบเทียบตรงกับค่าที่กำหนดไว้ จะทำคำสั่งในส่วนนี้ >
Case Else
  < กรณีนอกเหนือจากข้างต้น จะทำคำสั่งในส่วนนี้ >
End Select
```

- คำสั่งวนรอบแบบ For...Next มีรูปแบบดังนี้ คือ

```
For
  < ชื่อตัวแปรประเภทตัวเลข > = < ค่าเริ่มต้น > To < ค่าสิ้นสุด >
  < โปรแกรมที่ต้องการให้ทำงานวนรอบ >
Next
  < ชื่อตัวแปร >
```

- คำสั่งวนรอบแบบ Do Until หรือ While...Loop มีรูปแบบดังนี้ คือ

```
Do While/Until
  < เงื่อนไขการเปรียบเทียบ >
  < คำสั่งที่ต้องการให้ทำงานวนรอบ >
Loop
```

- คำสั่งวนรอบแบบ Do...Until หรือ While...Loop มีรูปแบบดังนี้

```
Do
  < คำสั่งที่ต้องการให้ทำงานวนรอบ >
Loop While/Until
  < เงื่อนไขการเปรียบเทียบ >
```

4.11 คำสั่งเกี่ยวกับการจัดการรูปภาพ

คำสั่งเกี่ยวกับการจัดการรูปภาพ ได้แก่

- LoadPicture : เพื่อทำการโหลดรูปภาพเข้ามาใส่ใน Picture Box หรือ Image มีรูปแบบ คือ
  [ชื่อ Picture Box].Picture = LoadPicture(ที่อยู่ของรูปภาพ)
- Point : ใช้อ่านค่าสีจากจุดที่กำหนดบนรูปภาพ ซึ่งจะให้ค่าสีในระบบ Long Color มีรูปแบบ คือ
  [ชื่อ Picture Box].Point (พิกัด X, พิกัด Y)
- PSet : ใช้เพื่อวาดจุดลงไปบนรูปภาพตามจุดที่กำหนด รูปแบบ คือ
  [ชื่อ Picture Box Name].PSet (พิกัด X, พิกัด Y), รหัสสีที่ต้องการ
- PaintPicture : ใช้เพื่อคัดลอกรูปภาพจากที่หนึ่งไปอีกที่ โดยสามารถปรับขนาดได้

ชื่อกล่องรูปภาพปลายทาง.PaintPicture(ชื่อกล่องรูปภาพต้นทาง, [ตำแหน่งเริ่มต้นของรูปต้นทางแกนX], [แกนY], [ขนาดความกว้างรูป], [ความสูงรูป], [ตำแหน่งเริ่มต้นของรูปต้นทางแกนX], [แกนY], [ขนาดความกว้างรูป], [ความสูงรูป], [รูปแบบการคัดลอกภาพ])

4.12 คำสั่งเกี่ยวกับการติดต่อพอร์ตอนุกรม

- CommPort : ใช้กำหนดหมายเลขของพอร์ตอนุกรมที่เราต้องการติดต่อด้วย รูปแบบ คือ

Object.CommPort = Value

เช่น

MSComm1.CommPort = 1

- Settings : ใช้ในการกำหนดอัตราบอด (Baud Rate) หรือความเร็วในการส่งข้อมูล มีหน่วยเป็นบิตต่อวินาที, พาริตี้, จำนวนของบิตข้อมูลมจำนวนของบิตปิดท้าย รูปแบบ คือ

Object. Settings = Value

เช่น

MSComm1.Settings = "9600, N, 8, 1"

- PortOpen : ใช้สำหรับเปิดและปิดการใช้งานพอร์ตอนุกรม รูปแบบ คือ

กรณีเปิดใช้งานพอร์ต MSComm1.PortOpen = True

กรณีปิดใช้งานพอร์ต MSComm1.PortOpen = False

- Input : ใช้สำหรับอ่านค่าข้อมูลจากพอร์ตอนุกรม รูปแบบ คือ

Object.Input

เช่น

Data = MSComm1.Input <เป็นการอ่านค่ามาเก็บในตัวแปรชื่อ Data >

- Output : ใช้สำหรับส่งข้อมูลออกจากพอร์ตอนุกรม รูปแบบ คือ

Object.Output

บทที่ 5
การออกแบบเครื่องเจาะแผ่นพลาสติก

5.1 ส่วนประกอบของเครื่องสร้างแบบสำหรับทอพรม

ส่วนประกอบของเครื่องสร้างแบบสำหรับทอพรมถูกแบ่งออกเป็น 3 ส่วน คือ

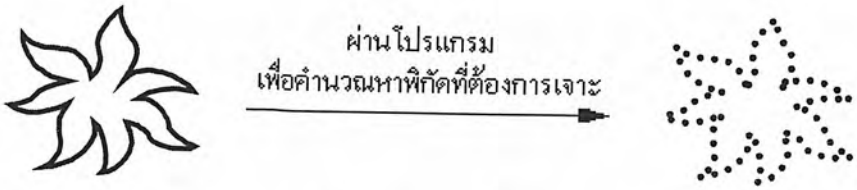
5.1.1 ระบบการเคลื่อนที่แบบ 2 แกน

ระบบการเคลื่อนที่แบบ 2 แกน ใช้สเต็ปปิ้งมอเตอร์ 2 ตัวในการสร้างการเคลื่อนที่แบบ 2 แกนโดยตัวแรกใช้ควบคุมตำแหน่งของเข็มเจาะ ซึ่งใช้การขับเคลื่อนของสายพานในการส่งถ่ายกำลัง และอีกตัวหนึ่งใช้ในการป้อนแผ่นพลาสติกเพื่อทำการเจาะ

รูปที่ 5-1 แสดงถึงระบบการเคลื่อนที่แบบ 2 แกน ซึ่งใช้สเต็ปปิ้งในการทำงาน

5.1.2 ส่วนโปรแกรมควบคุมการเจาะของหัวเจาะ

โปรแกรมที่ใช้ในการสร้างโปรแกรมสำหรับกำหนดพิกัดของเข็มเจาะ เป็นโปรแกรมที่ใช้ในการประมวลผลเพื่อหาตำแหน่งที่ต้องการเจาะ ซึ่งใช้ภาษาวิชชวลเบสิกในการเขียนโปรแกรมเพื่อคำนวณหาพิกัดที่ต้องการเจาะ แล้วจึงสั่งให้เข็มเจาะเคลื่อนที่ไปยังพิกัดที่คำนวณได้โดยผ่านชุดไมโครคอนโทรลเลอร์



รูปที่ 5-2 แสดงถึงรูปลายเส้นที่ผ่านโปรแกรมเพื่อคำนวณหาพิกัดที่ต้องการเจาะ

5.1.3 ส่วนของแผงควบคุม

ส่วนของแผงควบคุม จะเป็นชุดไมโครคอนโทรลเลอร์ที่ใช้ในการเชื่อมโยงข้อมูลระหว่างโปรแกรมที่สร้างขึ้น กับตัวเครื่องเจาะพลาสติก เพื่อคำนวณ และกำหนดพิกัดของการเจาะ รวมถึงควบคุมการทำงานของเครื่องเจาะพลาสติก ใช้ชิป MCS-51 ที่เขียนโดยภาษาแอสเซมบลี

รูปที่ 5-3 แสดงถึงการเชื่อมโยงข้อมูลจากโปรแกรมที่สร้างขึ้นไปยังตัวเครื่องเจาะพลาสติก

5.2 การป้อนพลาสติก

ในการดึงแผ่นพลาสติกเข้าไปในตัวเครื่องเพื่อให้เครื่องสามารถเจาะได้ถูกตำแหน่งนั้น ใช้ลูกกลิ้ง 2 ลูกใช้ในการหนีบแผ่นพลาสติก และใช้สเต็ปปิ้งมอเตอร์ในการหมุนลูกกลิ้งเพื่อดึงให้แผ่นพลาสติกเลื่อนเข้าไป และที่ปลายทั้ง 2 ด้านของลูกกลิ้งทั้งสอง เราได้ใช้แบริ่งแบบรับแรงตามแนวรัศมีมาช่วยในการลดแรงเสียดทาน

5.2.1 ระยะเคลื่อนที่ของลูกกลิ้งตัวล่าง

เนื่องจากใช้สเต็ปปิ้งมอเตอร์ซึ่งมีความละเอียดขนาด 1.8 องศาต่อสเต็ป และเลือกใช้เฟืองซึ่งมีโมดูลเท่ากับ 1 เส้นผ่านศูนย์กลางขนาด 14 มิลลิเมตร ดังนั้นเราจะหาระยะเคลื่อนที่เชิงเส้นได้จากผลคูณระหว่างองศาการหมุนกับรัศมีของเฟือง ดังนี้

$$S = R\theta \tag{5.1}$$

โดยที่ S คือ ระยะเคลื่อนที่เชิงเส้น

θ คือ องศาการหมุน

R คือ รัศมีของเฟือง, มิลลิเมตร

ดังนั้น

$$\text{ระยะเคลื่อนที่เชิงเส้น} = 7 \times 1.8 \times \frac{\pi}{180}$$

$$= 0.22 \text{ มิลลิเมตร}$$

เนื่องการข้อจำกัดในการเขียนโปรแกรมลงบนไมโครคอนโทลเลอร์ ซึ่งไม่สามารถจดจำได้ว่ากำลังอยู่ที่ขดลวดขั้วใด ดังนั้นการสั่งงานแต่ละครั้งทำให้ต้องเกิดการกระตุ้นขดลวดทุกขั้ว ทำให้ใน1รอบการสั่งงาน สเต็ปปิ้งมอเตอร์จะหมุนไป 4 สเต็ป หรือการเคลื่อนที่เท่ากับ 0.88 มิลลิเมตรต่อรอบนั่นเอง

จากการเลือกใช้เฟืองสำหรับลูกกลิ้ง ซึ่งมีขนาดเท่ากับเส้นผ่านศูนย์กลางของลูกกลิ้งพอดี ดังนั้นเมื่อสเต็ปปิ้งมอเตอร์หมุนไป 1 รอบ ก็จะทำให้ได้ระยะการเคลื่อนที่เชิงเส้นเท่ากับระยะการเคลื่อนที่ของเฟืองตัวเล็กด้วย ซึ่งจะเท่ากับ 0.88 มิลลิเมตร

รูปที่ 5-4 แสดงถึงระบบการขับเฟืองที่ใช้ในลูกกลิ้งตัวล่าง

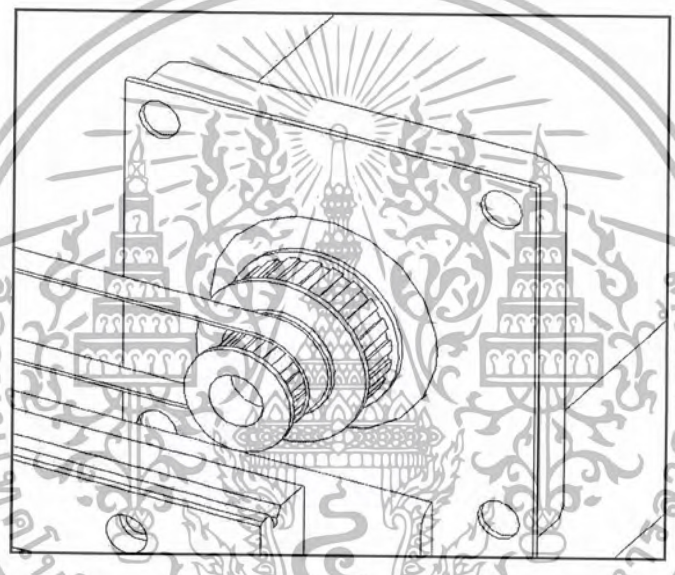
5.3 การเคลื่อนที่ของหัวเจาะ

ในการเจาะแผ่นพลาสติก เราใช้เข็มเจาะซึ่งทำจากท่อสแตนเลสสตีลลับคมที่ปลาย กระแทกลงไปบนแผ่นพลาสติก เพื่อตัดแผ่นพลาสติกให้ขาดเป็นรู โดยการเคลื่อนที่ขึ้นลงของเข็มเจาะนี้ใช้โซลินอยด์แม่เหล็กในการควบคุม และเราได้เลือกใช้โซลินอยด์ถึง 4 ตัวในการเจาะเพื่อเพิ่มความเร็วในการทำงาน

การเคลื่อนที่ของโซลินอยด์แม่เหล็กเหล่านี้ ตัวโซลินอยด์จะถูกยึดติดกับแผ่นเหล็กซึ่งติดอยู่กับลิเนียร์แบริ่ง ซึ่งจะทำให้เกิดการเคลื่อนที่ในแนวเส้นตรงและช่วยลดแรงเสียดทานในการเคลื่อนที่ โดยแผ่นเหล็กนี้จะถูกยึดติดกับสายพาน และคล้องผ่านสเต็ปมอเตอร์อีกตัว เพื่อให้สเต็ปมอเตอร์เป็นตัวควบคุมการเคลื่อนที่ของหัวเจาะ

5.3.1 ระยะเคลื่อนที่ของมูเล่สายพาน

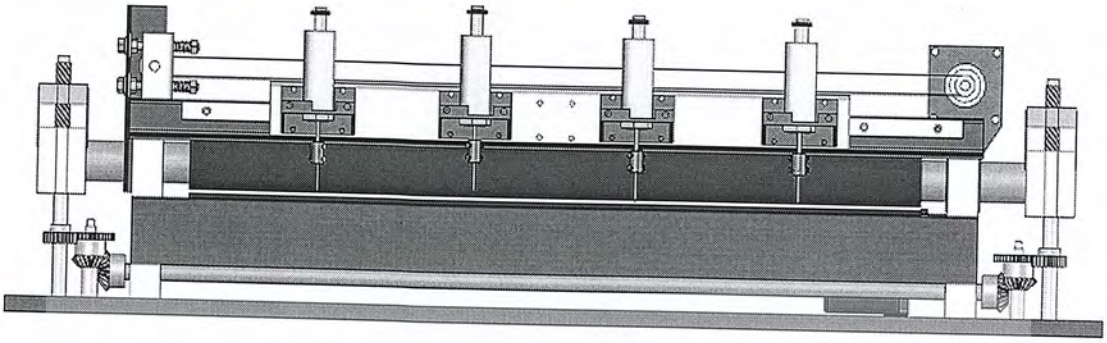
ระยะเคลื่อนที่ของมูเล่สายพานใช้หลักการเช่นเดียวกับการขับเคลื่อนลูกกลิ้ง โดยเลือกใช้มูเล่เส้นผ่านศูนย์กลาง 14 มิลลิเมตรเช่นกัน ดังนั้นเราจึงได้ระยะการเคลื่อนที่ของสายพาน 0.88 มิลลิเมตรรอบเช่นเดียวกัน



รูปที่ 5-5 แสดงถึงระบบการส่งกำลังโดยใช้สายพานเพื่อกำหนดตำแหน่งของเข็มเจาะ

5.3.2 การออกแบบระยะห่างของหัวเจาะแต่ละหัว

เนื่องจากเราเลือกใช้โซลินอยด์แม่เหล็ก 4 ตัวในการเจาะ เพื่อลดระยะการเคลื่อนที่ซึ่งจะทำให้เครื่องทำงานได้เร็วขึ้นเนื่องจากไม่ต้องวิ่งไปยังตำแหน่งต่างๆเป็นระยะทางมากนัก และนอกจากนี้ยังช่วยยืดอายุการทำงานของโซลินอยด์เนื่องจากการทำงานหนักได้อีกด้วย ดังนั้นการใช้โซลินอยด์ถึง 4 ตัวในการทำงานจะต้องมีการแบ่งขอบเขตของแต่ละตัวอย่างแน่นอน ซึ่งแต่ละตัวจะไม่ทำงานซ้ำซ้อนกันและยังเป็นตำแหน่งที่หัวเจาะสามารถเคลื่อนไปได้อีกด้วย



รูปที่ 5-6แสดงถึงการใช้โซลินอยด์แม่เหล็ก 4 ตัวในการเจาะ

จากระยะกว้างสุดที่หัวเจาะสามารถเลื่อนไปได้คือ ประมาณ 540 มิลลิเมตร และจำนวนหัวเจาะ 4 หัว เราจะได้สมการระยะเจาะดังนี้

$$0.88*S*4 < 540 \quad (5.2)$$

โดยที่ S คือ ระยะเคลื่อนที่เชิงเส้น

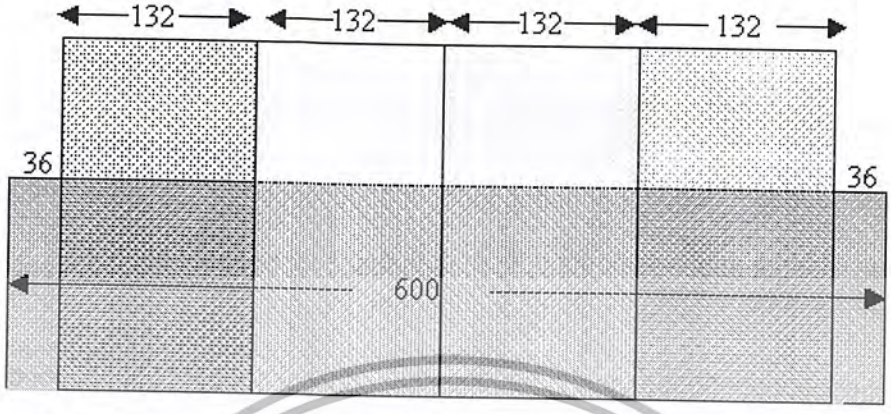
และเนื่องจากค่า S คือจำนวนสเต็ปซึ่งต้องมีค่าเป็นจำนวนเต็มเท่านั้น ดังนั้นค่า S มากที่สุดที่จะเป็นไปได้ คือ 125 สเต็ป ระยะเจาะสูงสุดจึงเท่ากับ 528 มิลลิเมตร และระยะห่างหัวเจาะแต่ละหัวจึงเท่ากับ 132 มิลลิเมตร

จากที่กล่าวมาทำให้เราสามารถสร้างสมการตำแหน่งของหัวเจาะแต่ละหัวได้ดังนี้

$$\text{ตำแหน่งหัวเจาะตัวที่ } k = i + (k - 1) *150*0.88 \quad (5.3)$$

โดยที่ i คือ ตำแหน่งหัวเจาะตัวที่ 1

k คือ หมายเลขหัวเจาะ

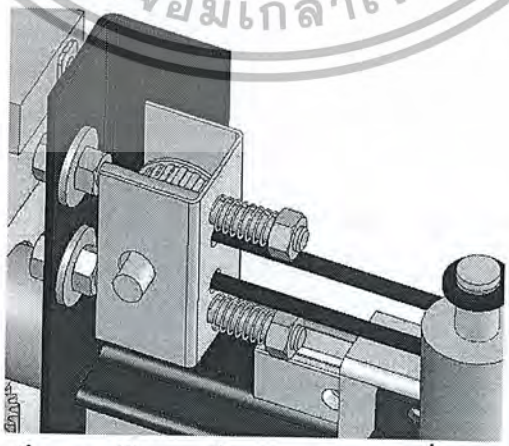


รูปที่ 5-7 แสดงถึงความกว้างของพื้นที่การทำงานของโซลินอยด์แต่ละตัวในหน่วยมิลลิเมตร

5.4 ระบบปรับความตึงของสายพาน

เนื่องจากการใช้สายพานในการขับเคลื่อนนั้น จำเป็นที่จะต้องได้ความตึงที่พอเหมาะพอดี หากตึงเกินไปก็จะทำให้เคลื่อนที่ยาก แต่ถ้าหากหย่อนเกินไปความสามารถในการส่งกำลังก็อาจจะลดลงหรือระยะทางในการเคลื่อนที่อาจถลาดเคลื่อนไป และเนื่องจากสายพานทำจากยาง ดังนั้นเมื่อใช้ไปเป็นเวลานานก็อาจเกิดการหย่อนขึ้นได้ ดังนั้นในเครื่องเจาะแผ่นพลาสติกนี้จึงได้เพิ่มเติมในส่วนของชุดปรับตึงสายพานขึ้นมาด้วย เพื่อทำให้ความตึงของสายพานมีค่าคงที่เสมอไม่ว่าจะใช้งานไปนานเท่าใด โดยระบบปรับตึงสายพานนี้จะใช้สปริงในการดันมูเล่ย์ให้ดึงสายพานให้ตึงตลอดเวลา และเรายังสามารถไขน็อตเข้าหรือออกเพื่อปรับระดับความตึงให้ได้ค่าพอเหมาะตามต้องการได้อีกด้วย

นอกจากนี้ระบบปรับตึงสายพานยังช่วยลดแรงกระชากที่เกิดจากการเคลื่อนที่ของสเต็ปมอเตอร์อย่างทันทีทันใดได้อีกด้วย



รูปที่ 5-8 แสดงถึงระบบปรับความตึงของสายพานในเครื่องเจาะแผ่นพลาสติก

5.5 ระบบยกลูกกลิ้ง

เนื่องจากแผ่นพลาสติกมีลักษณะบางและอ่อน ดังนั้นการใส่แผ่นพลาสติกเข้าไปในเครื่องเจาะจึงต้องมีการยกลูกกลิ้งตัวบนขึ้น เพื่อให้สามารถใส่แผ่นพลาสติกและจัดตำแหน่งได้อย่างง่ายดาย โดยการยกลูกกลิ้งขึ้นทำได้โดยใช้มอเตอร์ทดรอบซึ่งให้แรงบิดสูง บิดเพลาเพื่อหมุนชุดเฟืองดอกจอกซ้ายและขวาและไปขับระบบชุดเฟืองเพื่อนำไปหมุนสกรู เพื่อนำไปยกชิ้นเหล็กให้เลื่อนขึ้นหรือเลื่อนลงได้ โดยชิ้นเหล็กนี้จะถูกห้อยด้วยก้อนเหล็กซึ่งใช้เป็นฐานของลูกกลิ้งตัวบนนั่นเอง ซึ่งจากการใช้เพลาร่วมนั้นทำให้ชุดเฟืองทั้งสองด้านเกิดการเคลื่อนที่ที่เท่ากัน ลูกกลิ้งจึงเลื่อนขึ้นหรือลงในแนวระดับตลอดเวลา สาเหตุที่เราไม่ใช้สกรูไปยกก้อนเหล็กโดยตรงเนื่องจากระยะยกลงของลูกกลิ้งจะถูกกำหนดด้วยลิมิตสวิทช์ ดังนั้นถ้าหากลูกกลิ้งเลื่อนลงไปชนลิมิตสวิทช์แล้วลูกกลิ้งจะหยุดเคลื่อนที่ลง แต่ระยะที่ได้นั้นลูกกลิ้งอาจจะแค่สัมผัสกับลูกกลิ้งตัวล่างเท่านั้นแต่ไม่ได้กดแผ่นพลาสติกเลย ดังนั้นเราจึงใช้สกรูยกชิ้นเหล็กแทน เพื่อที่ว่าเราจะสามารถกำหนดให้ลูกกลิ้งที่เคลื่อนลงมานั้น ลงมาวางทับแผ่นพลาสติกและใช้น้ำหนักของตัวลูกกลิ้งเองในการกดแผ่นพลาสติกให้ติดอยู่กับลูกกลิ้งและทำให้ลูกกลิ้งสามารถดึงแผ่นพลาสติกให้เคลื่อนที่ไปได้

รูปที่ 5-9 แสดงถึงระบบยกลูกกลิ้งในเครื่องเจาะแผ่นพลาสติก

5.6 ระบบตรวจสอบสถานะการทำงาน

ระบบตรวจสอบสถานะการทำงาน แบ่งออกเป็น

5.6.1 ตรวจสอบตำแหน่งลูกกลิ้ง

จากที่กล่าวไปแล้วในเรื่องระบบยกลูกกลิ้งว่า ลูกกลิ้งบนจำเป็นต้องเคลื่อนที่ขึ้นลงได้เพื่อที่จะสามารถใส่แผ่นพลาสติกได้อย่างสะดวก ดังนั้นในส่วนนี้เราจึงมีลิมิตสวิทช์ 2 ตัวเพื่อตรวจสอบตำแหน่งของลูกกลิ้งว่ากำลังยกอยู่หรือกำลังกดแผ่นพลาสติก หากหลังจากใส่แผ่นพลาสติกแล้วแผงควบคุมตรวจสอบพบว่า

ลูกกลิ้งยังไม่ได้ถูกกดลงมาบนแผ่นพลาสติก เครื่องจะไม่ยอมเริ่มต้นทำงานจนกว่าผู้ใช้จะกดปุ่มเลื่อนลูกกลิ้งลงแล้วเพื่อที่จะได้แน่ใจว่าลูกกลิ้งจะสามารถดึงแผ่นพลาสติกได้

5.6.2 ตรวจสอบการมีอยู่ของแผ่นพลาสติก

เพื่อป้องกันการทำงานของเครื่องเจาะในขณะที่ไม่มีแผ่นพลาสติก เครื่องเจาะจึงมีลิมิตสวิทช์อีก1ตัว เพื่อทำการตรวจสอบว่ามีแผ่นพลาสติกอยู่ในเครื่องหรือไม่ โดยจะอยู่บริเวณหน้าลูกกลิ้งชิดขอบซ้าย โดยนอกจากจะตรวจสอบการมีอยู่ของแผ่นพลาสติกแล้ว ยังสามารถตรวจสอบการวางตำแหน่งของแผ่นพลาสติกว่าถูกต้องหรือไม่ได้อีกด้วย โดยการตรวจสอบนี้จะเริ่มตั้งแต่ช่วงการใส่แผ่นพลาสติกในตอนแรก คือถ้าหากแผงควบคุมตรวจสอบไม่พบแผ่นพลาสติกแล้ว แผงควบคุมจะไม่ยอมให้มีการยกลูกกลิ้งลง รวมถึงในขณะทำการเจาะแผ่นพลาสติกด้วยคือถ้าหากแผงควบคุมตรวจสอบพบว่าแผ่นพลาสติกหายไปหรือลิมิตสวิทช์ไม่ได้ถูกกด ซึ่งอาจเกิดจากแผ่นพลาสติกสั้นกว่าขนาดที่ต้องการ ก็จะสั่งให้เครื่องเจาะหยุดทำงาน

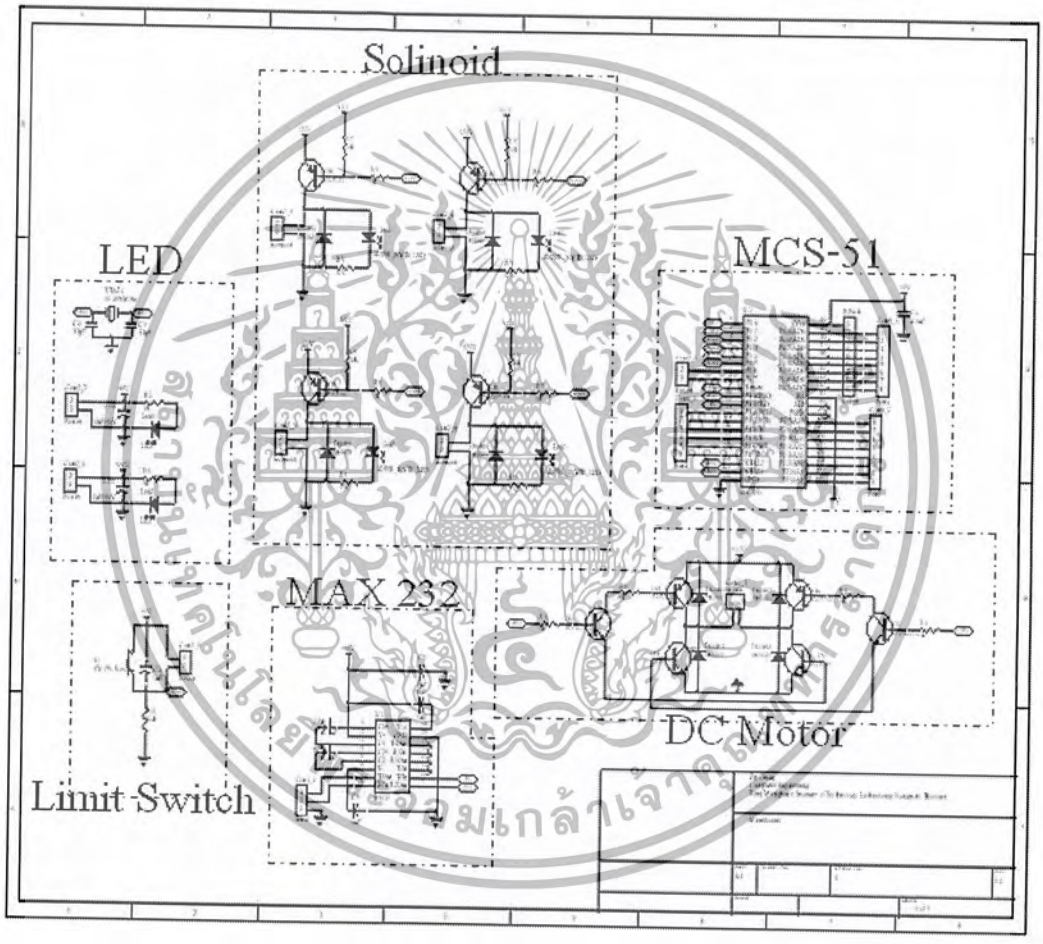
5.6.3 ตรวจสอบตำแหน่งของหัวเจาะ

ในส่วนนี้ก็จะมีลิมิตสวิทช์อีก 2 ตัว วางอยู่ทางด้านซ้ายสุดและขวาสุดของแนววิ่งหัวเจาะ โดยหน้าที่หลักของลิมิตสวิทช์ตัวซ้ายจะใช้เป็นจุดอ้างอิงตำแหน่งของหัวเจาะ โดยเมื่อเริ่มต้นเปิดเครื่องส่วนควบคุมจะสั่งให้หัวเจาะเคลื่อนที่ไปชนลิมิตสวิทช์ตัวซ้ายเพื่อกำหนดตำแหน่งเริ่มต้นก่อนการทำงาน ทำให้แผ่นพลาสติกที่จะถูกเจาะออกมา ได้รูปตรงตามตำแหน่งที่ต้องการ นอกจากนี้แล้วลิมิตสวิทช์ทั้ง2ตัวนี้ยังทำหน้าที่เป็นตัวป้องกันการผิดพลาดของการทำงานหรือป้องกันการเคลื่อนที่เลยออกนอกขอบเขตอีกด้วย เนื่องจากตำแหน่งที่จะเกิดการเจาะจะอยู่ในบริเวณระหว่างลิมิตสวิทช์ทั้งสองตัวนี้ ดังนั้นถ้าหากเกิดเหตุการณ์ที่หัวเจาะเคลื่อนที่ไปชนลิมิตสวิทช์ตัวใดตัวหนึ่งแล้ว จึงอาจเป็นไปได้ว่าได้เกิดการผิดพลาดเกี่ยวกับตำแหน่งเกิดขึ้นแล้ว ดังนั้นส่วนควบคุมจะสั่งให้หัวเจาะกลับไปเริ่มต้นกำหนดตำแหน่งตัวเองใหม่ และส่วนควบคุมจะส่งสัญญาณไปบอกโปรแกรมคอมพิวเตอร์ให้ทราบว่าได้เกิดข้อผิดพลาดขึ้นแล้วและให้โปรแกรมทำการกำหนดตำแหน่งปัจจุบันใหม่เพื่อให้ตำแหน่งหัวเจาะจริงและตำแหน่งหัวเจาะในคอมพิวเตอร์มีค่าตรงกัน

บทที่ 6
ระบบควบคุมของเครื่องสร้างแบบสำหรับทอพรม

6.1 การออกแบบวงจรไฟฟ้า

ในการออกแบบวงจรไฟฟ้าจะใช้ Protel 99 SE ในการออกแบบวงจร และการออกแบบ แผ่นปริ้นท์ โดยจะได้รูปแบบของวงจรในแต่ละส่วน ดังนี้



รูปที่ 6-1 วงจรไมโครคอนโทรลเลอร์ MCS-51 ที่ออกแบบโดย Protel 99 SE

ในส่วนของวงจรในการขับสเต็ปปิ้งมอเตอร์ ได้ใช้บอร์ด ET-SMCC V2.0 (R2) เป็นบอร์ด Driver Stepping Motor ซึ่งพัฒนาขึ้นมาจาก บอร์ด ET-SMCC V2.0 โดยทำการเพิ่มเติมในส่วนของ Connector 10 Pin ที่เรียกว่า 10 PIN ET

บอร์ด ET-SMCC V2.0 (R2) สามารถต่อใช้งานร่วมกับบอร์ดไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์ต่างๆได้อย่างง่ายดาย โดยที่ตัวบอร์ดเองต้องการสัญญาณในการควบคุมการทำงานทั้งหมด 6 เส้นสัญญาณ โดยสามารถรับสัญญาณลอจิกแบบ TTL มาตรฐานได้โดยตรง ซึ่งบอร์ดสามารถขับกระแสให้กับสเต็ปปิ้งมอเตอร์แบบ 2 เฟส หรือ Bipolar Stepping Motor ได้ทันทีโดยไม่ต้องดัดแปลงใดๆ

โดยในส่วนของวงจรขับกระแสให้กับมอเตอร์บนบอร์ด ET-SMCC V2.0 (R2) นั้น จะใช้ IC เบอร์ L298N ของ SGS-THOMSON และบอร์ด ET-SMCC V2.0 (R2) นี้จะใช้ได้กับสเต็ปปิ้งมอเตอร์แบบ 2 ขั้ว หรือมอเตอร์ที่มีสาย 4 เส้น คือ Bipolar Stepping Motor เท่านั้น

คุณสมบัติของบอร์ด ET-SMCC V2.0 (R2)

Channel Control : 2 Channel (1 Bipolar Stepper Motor หรือ 2 DC Motor)

Step Frequency : สูงสุด 40 KHz

Motor Control Type : Bipolar Stepper Motor หรือ DC Motor

Output Driver Current : 4 A/Phase

Input Logic Control : Standard TTL Logic Level

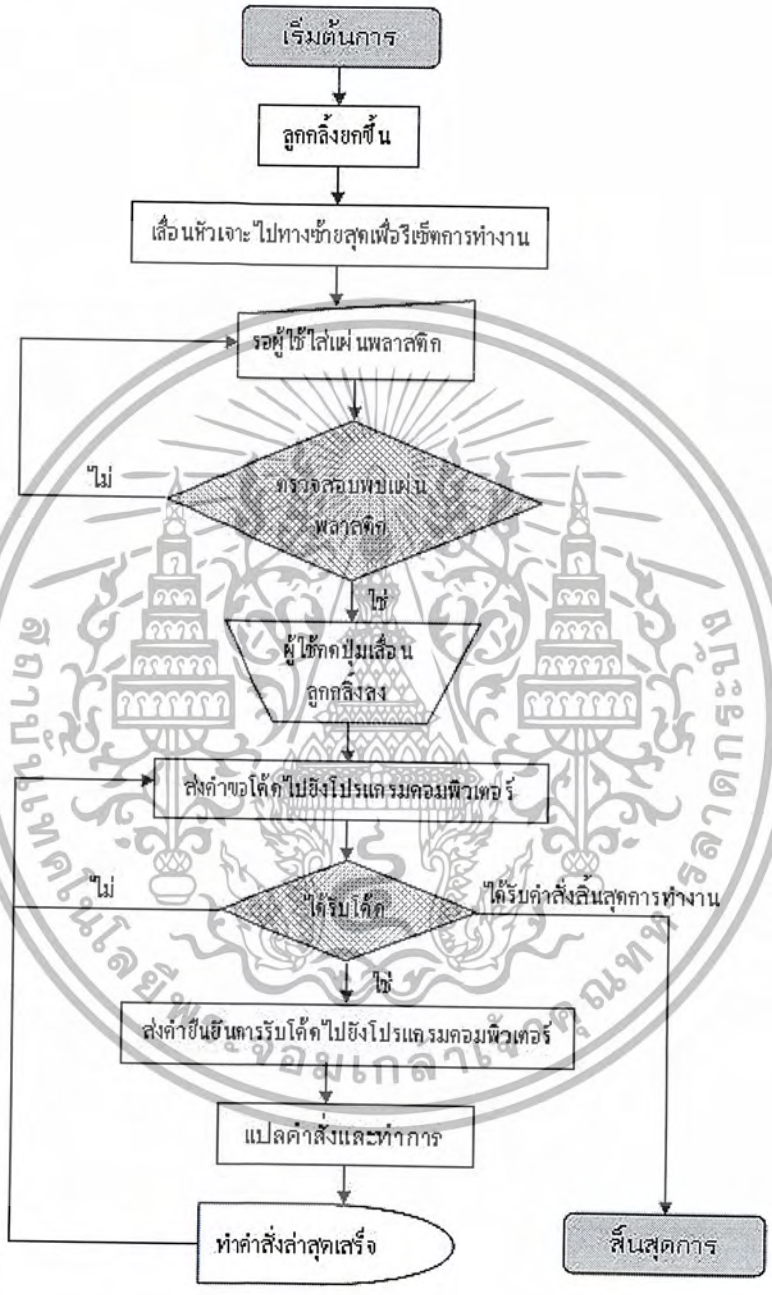
Power Supply

Logic Supply : +5 Vdc / 20 mA

Motor Supply : Motor Supply (Maximum 50 Vdc / 4A)

PCB size : 7.6 cm. X 8.4 cm.

6.2 การออกแบบระบบควบคุมการสั่งงาน



รูปที่ 6-2 แสดงถึงลำดับการทำงานของระบบควบคุมการสั่งงาน

บทที่ 7
การออกแบบโปรแกรมคำนวณพิกัดการเจาะ

7.1 สาเหตุที่เลือกใช้โปรแกรมวิชวลเบสิกในการเขียนโปรแกรม

1. เป็นภาษาระดับกลาง มีลักษณะคล้ายภาษาอังกฤษ จึงทำให้ง่ายต่อการทำความเข้าใจและพัฒนา
2. มีคำสั่งที่ใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมซึ่งสามารถนำไปควบคุมเครื่องได้
3. สามารถใช้งานกราฟฟิคได้อยู่ในระดับที่น่าพอใจ ทำให้สามารถพัฒนาโปรแกรมได้อย่างสะดวก
4. วิชวลเบสิกใช้พัฒนาโปรแกรมที่ใช้สื่อสารกับผู้ใช้เป็นหลัก ดังนั้นจึงสามารถพัฒนาหน้าตา (Interface) ของโปรแกรมออกมาได้สวยงามน่าใช้

7.2 โปรแกรมคำนวณพิกัดในการเจาะ

เนื่องจากเครื่องเจาะแผ่นพลาสติกนี้จะต้องติดต่อกับผู้ใช้งาน ดังนั้นโปรแกรมจึงถูกพัฒนาออกให้ให้ผู้ใจสามารถใช้งานได้อย่างง่ายดายและสามารถเข้าใจวิธีใช้งานได้ในเวลาอันรวดเร็ว โดยโปรแกรมจะแบ่งขั้นตอนต่างๆออกเป็น 3 ส่วน คือส่วนการปรับขนาดพลาสติก ,การจัดการรูปภาพและดำเนินการเจาะโดยหลังจากผู้ใช้งานนำรูปภาพเข้ามาในโปรแกรมแล้ว ผู้ใช้สามารถปรับขนาดรูปภาพ,หมุนภาพหรือเลื่อนตำแหน่งรูปภาพไปไว้ที่ใดก็ได้บนแผ่นพลาสติก หลังจากนั้นก็สามารถปรับขนาดพลาสติกให้พอดีกับรูปภาพทั้งยังสามารถกำหนดระยะขอบได้อีกด้วย ทำให้รู้ขนาดแผ่นพลาสติกที่จะต้องจัดเตรียมได้ทันที

หลังจากจัดการในส่วนของรูปภาพและแผ่นพลาสติกเรียบร้อยแล้วก็เข้ามาถึงส่วนของการเจาะ ก่อนทำการเจาะ ผู้ใช้สามารถสามารถเลือกได้ว่าต้องการดูภาพตัวอย่างของแผ่นพลาสติกหรือไม่ หากต้องการโปรแกรมก็จะทำการคำนวณหาพิกัดจุดที่จะถูกเจาะทั้งหมดบนภาพและแสดงขึ้นมาให้ผู้ใช้ได้ดู และถ้าหากพอใจผลงานก็สามารถสั่งให้เครื่องเจาะเริ่มทำงาน แต่ถ้าหากไม่พอใจผู้ใช้ก็สามารถย้อนกับไปจัดการกับรูปภาพได้ใหม่ตามต้องการ หรือถ้าหากผู้ใช้ไม่ต้องการรอดูตัวอย่างก็สามารถสั่งให้เครื่องทำงานได้ทันที จากนั้นก็เพียงปล่อยให้เครื่องทำงานโดยอัตโนมัติและผู้ใช้ก็สามารถมีเวลาว่างเพื่อไปทำงานอื่นๆได้

7.3 หลักการทำงานของโปรแกรม

หลักจากผู้ใช้กดปุ่มเริ่มแล้ว โปรแกรมก็จะทำการตรวจสอบว่าผู้ใช้ได้นำรูปลายเส้นเข้ามาในโปรแกรมแล้วหรือไม่รวมถึงรูปภาพนั้นถูกกำหนดความละเอียดแล้วหรือยังอีกด้วย หากผู้ใช้ยังไม่ได้นำเข้ารูปภาพหรือลืมกำหนดความละเอียด ตัวโปรแกรมจะแจ้งให้ผู้ใช้ได้ทราบบนจอ เพื่อให้ผู้ใช้ได้กลับไปแก้ไข หากทุกอย่างเรียบร้อยแล้ว โปรแกรมจะทำการปรับขนาดรูปภาพและแผ่นพลาสติกให้มีอัตราส่วน 100 เปอร์เซ็นต์ หรือทำให้ 1 พิกเซลมีค่าเท่ากับระยะการเคลื่อนที่ของสเต็ปปิ้งมอเตอร์นั่นเอง การทำเช่นนี้จะทำให้หัวเจาะสามารถเคลื่อนไปยังตำแหน่งที่ต้องการเจาะได้พอดี หลังจากปรับขนาดแล้วโปรแกรมจะทำการ

กำหนดค่าเริ่มต้นและคุณสมบัติต่างๆที่ต้องใช้ในโปรแกรม รวมถึงตำแหน่งเริ่มต้นของหัวเจาะด้วย จากนั้นโปรแกรมจะตรวจสอบว่าผู้ใช้ต้องการดูตัวอย่างภาพหรือไม่ หากต้องการโปรแกรมจะทำการคำนวณพิกัดต่างๆและแสดงตัวอย่างขึ้นมา หากผู้ใช้พอใจก็จะทำงานต่อ โดยจะเริ่มส่งโค้ดคำสั่งให้เครื่องเจาะโดยนำค่าพิกัดซึ่งได้คำนวณออกมาแล้วส่งออกไป แต่ในกรณีที่ผู้ใช้ไม่ต้องการดูตัวอย่างโปรแกรมจะเริ่มทำการคำนวณพิกัดที่ละจุดแล้วจึงทยอยส่งออกไป ทั้ง 2 กรณีนี้การส่งโค้ดคำสั่งออกไปจะสัมพันธ์กับความต้องการของเครื่องเจาะ โดยหลังจากส่งคำสั่งแรกออกไปแล้วโปรแกรมจะรอรับสัญญาณจากชุมควบคุมเครื่องเจาะ หากชุดควบคุมได้รับโค้ดแล้วชุดควบคุมจะส่งสัญญาณกลับมาให้โปรแกรมได้รับทราบว่าโค้ดที่ถูกส่งออกไปได้รับแล้วเรียบร้อย โปรแกรมจึงจะเริ่มทำการหากพิกัดต่อไปหรือในกรณีที่ได้คำนวณไว้แล้ว(เพื่อแสดงตัวอย่าง) โปรแกรมก็จะไปอ่านโค้ดของตำแหน่งถัดไป จากนั้นโปรแกรมจะรอให้เครื่องเจาะทำงานตามโค้ดที่ถูกส่งออกไปก่อนหน้านี้แล้วจนเสร็จ เมื่อเสร็จแล้วชุดควบคุมก็จะส่งคำสั่งร้องขอโค้ดใหม่เข้ามา เมื่อโปรแกรมได้รับคำร้องขอแล้วก็จะทำการส่งโค้ดที่เตรียมไว้ออกไปและจะรอคำยืนยันการรับเช่นเดิม หากได้รับคำยืนยันการรับแล้วโปรแกรมก็จะเริ่มทำการคำนวณหาค่าพิกัดต่อไป และจะทำงานวนเช่นนี้ไปเรื่อยจนเสร็จ

7.4 การตรวจสอบสถานะของเครื่องเจาะ

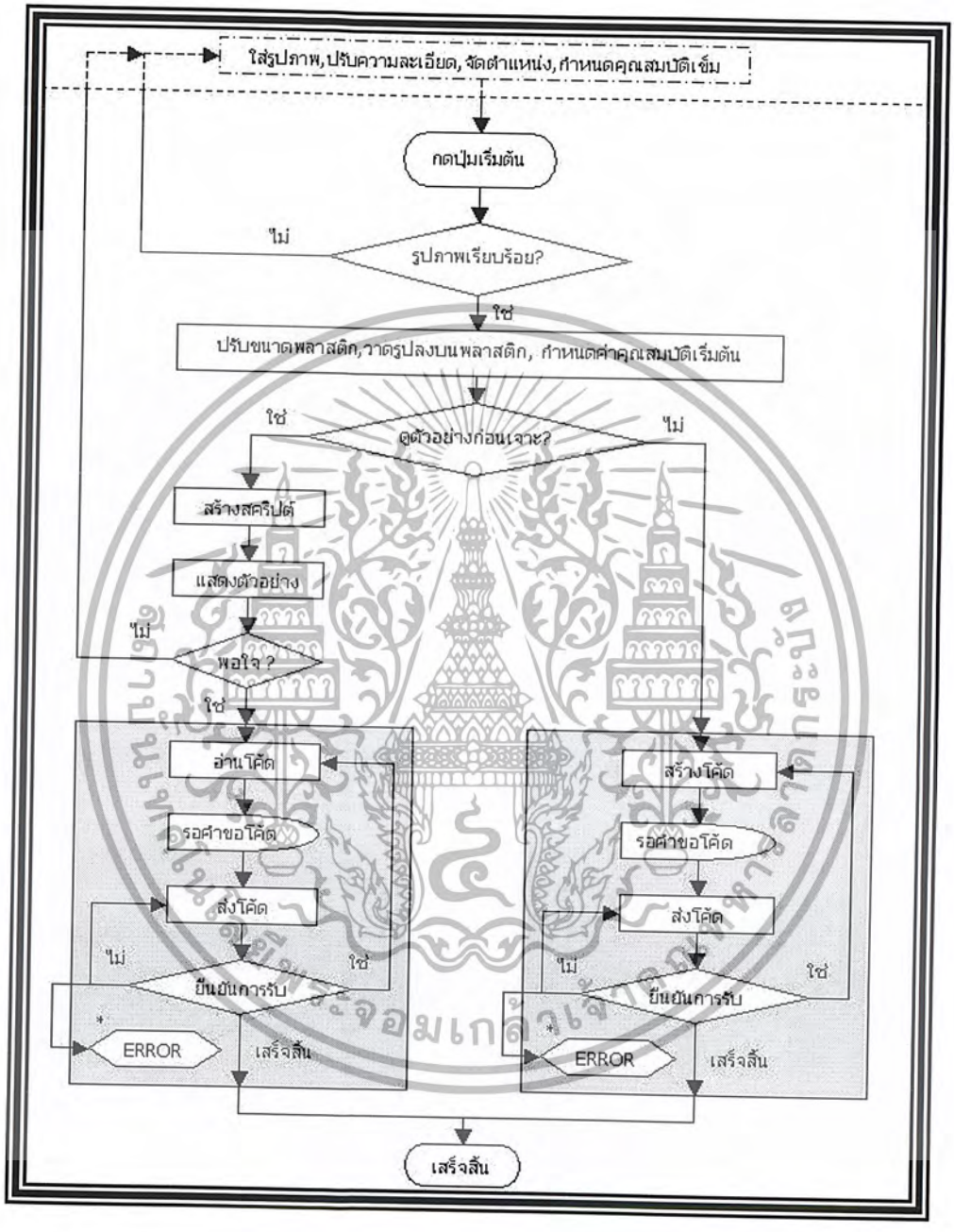
แม้ว่าพอร์ตอนุกรมจะมีเส้นสัญญาณจำนวนหลายเส้น แต่เนื่องจากความซับซ้อนของการเขียนโปรแกรมหากต้องการใช้เส้นสัญญาณหลายเส้น ดังนั้นเราจึงเลือกใช้เส้นสัญญาณเพียง 3 เส้นเท่านั้น นั่นคือ เส้นกราวนด์ของระบบ,เส้นรับข้อมูลและเส้นส่งข้อมูล โดยการตรวจสอบสถานะของเครื่องจะทำโดยการติดต่อสื่อสารกันเป็นหลัก หากไม่มีการสื่อสารกันเกิดขึ้นเป็นเวลานานก็อาจสันนิฐานได้ว่าอาจเกิดข้อผิดพลาดเกิดขึ้น

ในการตรวจสอบ จะทำในช่วงเวลาการรับส่งข้อมูล โดยหลังจากโปรแกรมส่งโค้ดคำสั่งออกไปแล้ว โปรแกรมจะทำการรอการตอบกลับการยืนยันการรับข้อมูล หากไม่มีการตอบกลับ โปรแกรมจะส่งโค้ดเดิมไปอีกครั้ง หากยังไม่มีการตอบกลับอีกโปรแกรมก็จะส่งโค้ดเดิมนั้นซ้ำไปเรื่อยๆจนกระทั่งเกินเวลาที่กำหนด โปรแกรมก็จะทำการแจ้งความผิดพลาดขึ้นมาบนจอคอมพิวเตอร์ เพื่อให้ผู้ใช้ได้ลองตรวจสอบความเรียบร้อยของเครื่อง

นอกจากนี้หลังจากการส่งโค้ดและได้รับคำยืนยันเรียบร้อยแล้ว โปรแกรมก็ยังจะจับเวลาการทำงานของเครื่องด้วย หากโปรแกรมรอคำขอโค้ดนานเกินไป เกินกว่าเวลานานที่สุดที่เครื่องควรทำคำสั่งเสร็จสิ้น โปรแกรมก็จะทำการแจ้งความผิดพลาดให้ผู้ใช้ได้รับรู้เช่นกัน

ด้วยวิธีการตรวจสอบรูปแบบนี้ ทำให้เราสามารถตรวจสอบความผิดพลาดที่เกิดขึ้นได้ แม้ว่าจะไม่สามารถระบุสาเหตุของความผิดพลาดให้รับรู้ได้ แต่อย่างน้อยก็ยังช่วยบอกผู้ใช้ให้รับรู้ถึงสิ่งผิดปรกติที่เกิดขึ้นกับเครื่องได้ ซึ่งสาเหตุเหล่านี้อาจเกิดจากการลืมเสียบปลั๊ก, ปลั๊กไม่แน่น, ไฟฟ้าดับ, สายสัญญาณไม่ได้เสียบหรือแม้กระทั่งการใส่แผ่นพลาสติกลงในเครื่องยังไม่เรียบร้อยทำให้เครื่องไม่ทำงาน

7.5 แผนผังลำดับการทำงาน



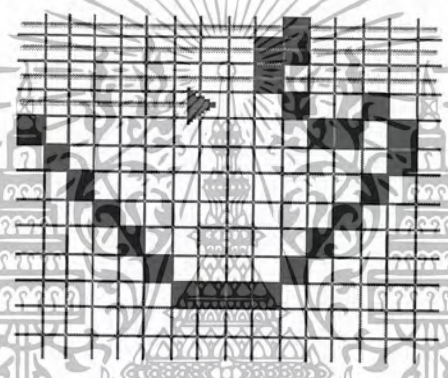
รูปที่ 7-1 แสดงแผนผังลำดับการทำงานของโปรแกรมในการคำนวณหาพิกัดที่ต้องการเจาะ

* เมื่อไม่มีการยืนยันการรับโค้ด โปรแกรมจะส่งโค้ดเดิมซ้ำไปเรื่อยๆทุกๆ 1 วินาที หากจำนวนครั้งที่ส่งข้อมูลเกินค่าที่กำหนด โปรแกรมจะแจ้งคำเตือนว่าเครื่องเจาะมีปัญหา

7.6 โปรแกรมย่อยการสร้างโค้ด (Sub Program)

จากการที่รูปภาพในคอมพิวเตอร์เกิดจากการนำจุดสีขนาดเล็ก(หรือที่เรียกกันว่า Pixels) หลายๆจุดมาเรียงต่อกัน ประกอบกันเป็นภาพ ซึ่งค่าสีในแต่ละจุดจะถูกจัดเก็บในระบบตัวเลข (Digital) โดยสีแต่ละสีนั้นจะมีค่าเฉพาะเป็นของตนเอง ดังนั้นเราจึงได้นำเอาหลักการในส่วนนี้มาใช้ในการเขียนโปรแกรมคำนวณพิกัดในการเจาะ

โปรแกรมคำนวณตำแหน่งจุดที่จะเจาะนี้ใช้หลักการการเปรียบเทียบค่าสีแต่ละจุดบนรูปภาพ โดยโปรแกรมจะทำการไล่อ่านค่าสีแต่จุดจากซ้ายไปขวาและจากบนลงล่าง เนื่องจากรูปภาพที่นำมาใช้จะเป็นรูปลายเส้น ซึ่งประกอบด้วยสีเพียง2 สี คือ เส้นสีดำและพื้นสีขาว ดังนั้นถ้าพบว่าจุดใดมีสีดำแล้ว จุดนั้นก็เป็นตำแหน่งที่ต้องทำการเจาะ



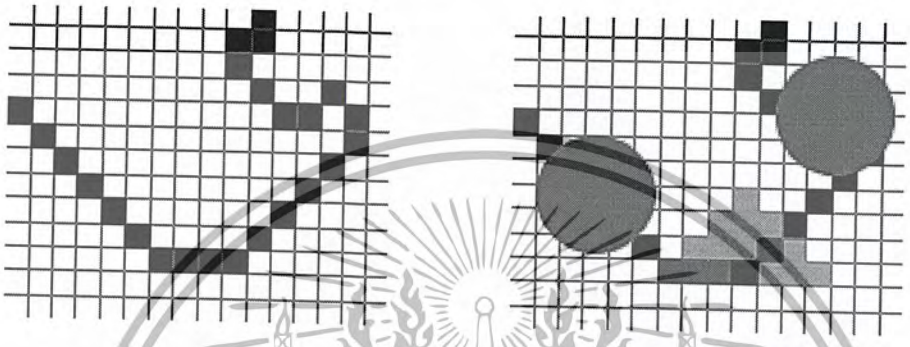
รูปที่ 7-2 แสดงวิธีการอ่านค่าสีแต่ละพิกเซล

เนื่องจากข้อจำกัดการเคลื่อนที่ของหัวเจาะ ดังนั้นก่อนที่เราจะเริ่มต้นเปรียบเทียบสี โปรแกรมจะทำการปรับอัตราส่วนของรูปภาพก่อน เพื่อทำให้แต่ละพิกเซลมีความระยะห่างเทียบเท่ากับระยะจริง 0.88 มิลลิเมตรหรือเท่ากับการหมุนสเต็ปปิ้งมอเตอร์ไป 1 รอบนั่นเอง เหตุที่ทำเช่นนี้ก็เพื่อทำให้หัวเจาะสามารถเคลื่อนที่ไปถึงจุดที่ต้องการได้พอดี

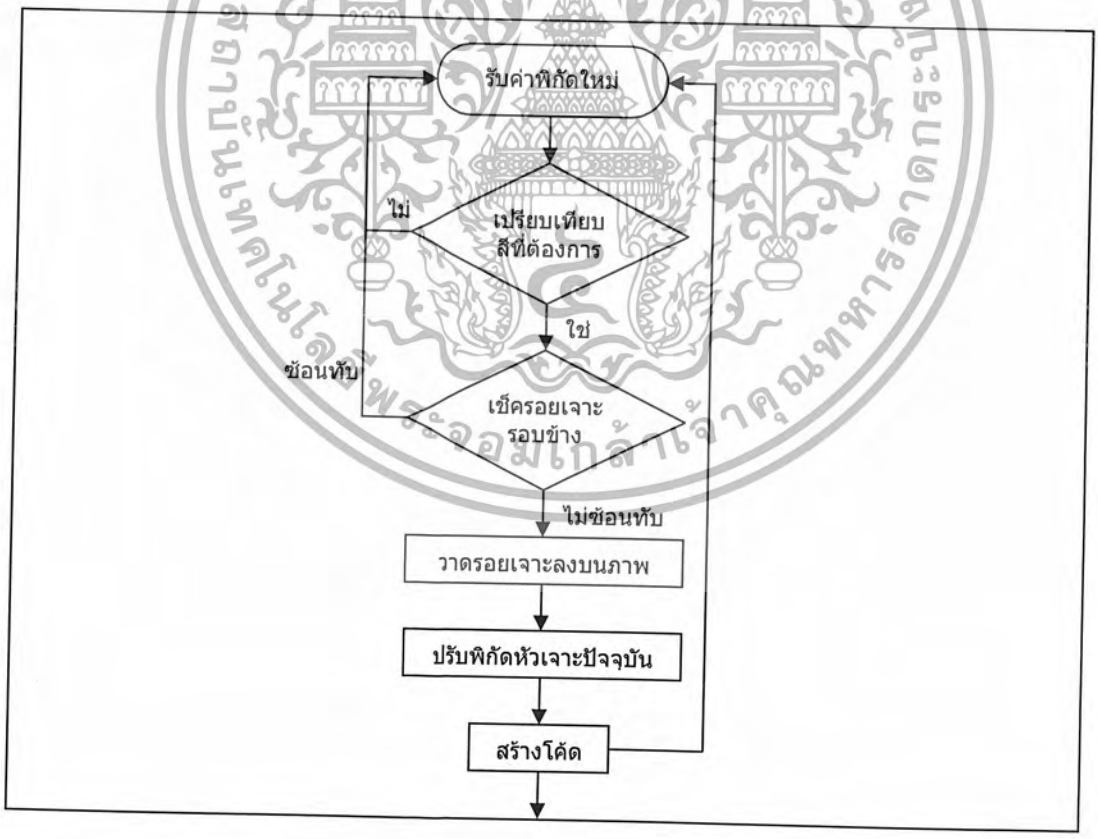
และเนื่องด้วยการปรับขนาดรูปภาพนี้เอง ทำให้รายละเอียดของรูปภาพบางส่วนอาจเสียไป เราจึงใช้ฟังก์ชั่นจัดการรูปภาพที่สามารถประมาณภาพจากพื้นที่รอบข้างได้ เพื่อเติมเต็มส่วนของรูปภาพที่ขาดหายไป และด้วยการประมาณค่าสีนี้ จะทำให้ภาพบางส่วนเกิดสีเทาขึ้นมา ดังนั้นการเปรียบเทียบสีจึงจำเป็นที่จะต้องใช้ช่วงสีที่อยู่ระหว่างสีเทาไปจนถึงสีดำสนิท

นอกจากนี้จะสังเกตได้ว่าขนาดของเข็มเจาะจะมีขนาดใหญ่กว่าขนาดของพิกเซล ดังนั้นถ้าหากเราเจาะทุกพิกเซลที่พบสีดำ จะทำให้รอยเจาะติดกันมากเกินไป เกิดรอยขาดเป็นแนวยาว พลาสติกอาจขาดออกจากกันไม่สามารถนำไปใช้ได้ ดังนั้นหลังจากที่เราเจอพิกเซลที่เป็นสีดำแล้ว เราจึงไม่สามารถนำไปเจาะได้ทันที แต่ต้องนำจุดนั้นไปตรวจสอบก่อนว่าหากเจาะที่จุดนั้นแล้ว รอยขาดจะไปทับกับรูอื่นหรือไม่ ถ้าหากทับ

จะยกเลิกการเจาะที่จุดนั้นและเริ่มต้นทำการเปรียบเทียบสีที่จุดถัดไป หลักการที่ใช้ในการตรวจสอบคือ จุดที่ถูกเจาะถูกวาดจุดสีแดงลงไปบนแบบ และจุดถัดไปที่ถูกตรวจสอบจะทำการตรวจสอบจุดสีแดงรอบตัวเองเป็นวงกลมรัศมีเท่ากับขนาดเข็มบวกกับระยะห่าง (Gap) ที่ต้องการ แต่จากการที่โปรแกรมไล่เช็คสีเรียงจากบนลงล่าง ดังนั้นด้านล่างของจุดที่ถูกตรวจสอบจะไม่พบจุดสีแดงแน่นอน เราจึงสามารถลดขนาดการตรวจสอบจากวงกลมเหลือแค่ครึ่งวงกลมได้ ทำให้โปรแกรมทำงานได้เร็วขึ้น



รูปที่ 7-3 รูปแสดงตัวอย่างการตรวจสอบจุดที่จะเจาะ



รูปที่ 7-4 โฟว์ลชาร์ทโปรแกรมย่อยการหาพิกัดเจาะและสร้างโค้ด

ตัวอย่างโค้ด SL007R012BE

ความหมาย

S : Start

L : สเต็ปปิ้งมอเตอร์สำหรับลูกกลิ้งหมุนซ้าย (R : หมุนขวา, N : ไม่หมุน)

007 : จำนวนสเต็ปสำหรับสเต็ปปิ้งมอเตอร์ลูกกลิ้ง 7 สเต็ป

R : สเต็ปปิ้งมอเตอร์สำหรับมูเล่ย์หมุนขวา (L : หมุนซ้าย, N : ไม่หมุน)

012 : จำนวนสเต็ปสำหรับสเต็ปปิ้งมอเตอร์มูเล่ 12 สเต็ป

B : โซลินอยด์แม่เหล็กตัวที่ B ทำการเจาะ (A,B,C,D)

E : End

บทที่ 8
รูปแบบและผลการทดลอง

8.1 รูปแบบการทดลอง

รูปแบบการทดลองเพื่อหาประสิทธิภาพรวมของเครื่องสร้างแบบสำหรับทอพรมนี้ จะแบ่งการทดลองออกเป็น 3 ส่วนด้วยกัน คือ

1. ส่วนโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะที่จะทำการเจาะแผ่นพลาสติก
2. ส่วนของหัวเจาะที่ใช้ในการเจาะแผ่นพลาสติก
3. ส่วนของระบบโต๊ะงานเคลื่อนที่แบบ 2 แกน

เพื่อทดสอบว่าเครื่องสร้างแบบสำหรับทอพรมยังมีข้อบกพร่องส่วนใดบ้างที่ต้องได้รับการแก้ไข ทั้งนี้เพื่อให้เครื่องสร้างแบบสำหรับทอพรมสามารถใช้งานได้จริงอย่างมีประสิทธิภาพมากที่สุด

8.2 วิธีการทดลอง

1. ส่วนโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะที่จะทำการเจาะแผ่นพลาสติก มีขั้นตอนดังนี้คือ

นำรูปลายเส้นเข้าโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะจำนวน 10 รูป ซึ่งรูปลายเส้นแต่ละรูปจะมีรูปแบบแนวเส้นแตกต่างกันไป จากนั้นจึงสั่งให้โปรแกรมเริ่มต้นทำงานเดินตามแนวเส้นแล้วแสดงพิกัดการทำงานออกมาบนรูปที่นำเข้าไปโดยแสดงเป็นจุด แล้วจึงนับจำนวนพิกัดการทำงานที่แสดงพิกัดไม่ตรงตามรูปลายเส้นที่นำเข้าไป เพื่อคำนวณหาประสิทธิภาพของตัวโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะโดยคำนวณจากสูตร

$$\text{ประสิทธิภาพของโปรแกรม} = \frac{\text{จำนวนรูเจาะที่เจาะตรงตามแนวเส้น}}{\text{จำนวนรูเจาะทุกรู}}$$

รูปที่ 8-1 แสดงถึงพิกัดการทำงานที่เดินตรง และไม่ตรงตามแนวเส้น

2. ส่วนของหัวเจาะที่ใช้ในการเจาะแผ่นพลาสติก มีขั้นตอนการทดลองดังนี้คือ

ทดสอบเจาะชิ้นงานจริงโดยใช้เครื่องสร้างแบบสำหรับทอพรมจำนวน 10 ชิ้น ซึ่งชิ้นงานแต่ละชิ้นจะมีรูปแบบแนวเส้นแตกต่างกันไป จากนั้นจึงนำชิ้นงานที่ได้มานับจำนวนรูที่เจาะไม่เข้า หรือรูที่เจาะเสีย เพื่อคำนวณหาประสิทธิภาพของหัวเจาะที่ใช้ในการเจาะแผ่นพลาสติก โดยคำนวณจากสูตร

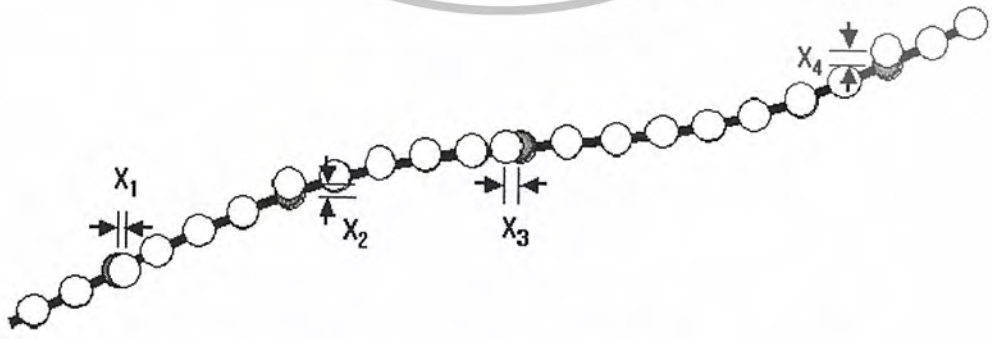
$$\text{ประสิทธิภาพของหัวเจาะ} = \frac{\text{จำนวนรูเจาะไม่เสียทุกรู}}{\text{จำนวนรูเจาะทุกรู}}$$

รูปที่ 8-2 แสดงถึงตำแหน่งของรูเจาะที่เจาะตรง และไม่ตรงตามแนวเส้น รวมถึงรูเจาะที่เจาะเสีย

3. ส่วนของระบบการเคลื่อนที่แบบ 2 แกน มีขั้นตอนการทดลองดังนี้คือ

ทำการกำหนดพิกัด เอ็กซ์-วาย เข้าเครื่องสร้างแบบสำหรับทอพรมจำนวน 100 พิกัด เพื่อให้เครื่องทำการเจาะตามพิกัดตามพิกัดที่ได้กำหนดไว้ จากนั้นจึงนำชิ้นงานที่ได้มาเปรียบเทียบพิกัด เพื่อวัดระยะคลาดเคลื่อนตามแนวแกนเอ็กซ์ และแกนวาย เพื่อคำนวณหาประสิทธิภาพของระบบโต๊ะงานเคลื่อนที่แบบ 2 แกน จากสูตร

$$\text{ระยะคลาดเคลื่อนต่อรู} = \frac{\sum x}{\text{จำนวนรูเจาะทุกรู}}$$



รูปที่ 8-3 แสดงถึงพิกัดของรูเจาะที่เจาะตรง และไม่ตรงตามแนวเส้น

8.3 ผลการทดลอง

จากการทดลองใช้เครื่องเจาะแผ่นพลาสติกพบว่า

1. การเคลื่อนที่ของเครื่องเจาะจะเสียเวลาค่อนข้างมากเนื่องจากเป็นแบบไล่อ่านสีทีละพิกเซล และจะเจาะเมื่อเจอจุดที่ต้องการเท่านั้น

2. ตำแหน่งของรูที่ต้องการเจาะมีระยะคลาดเคลื่อนประมาณ 4.37 มิลลิเมตรต่อรู

3. ประสิทธิภาพของตัวโปรแกรมคิดเป็น 87.88 %

4. ประสิทธิภาพของหัวเจาะคิดเป็น 89.72 %

5. ประสิทธิภาพของระบบคิดเป็น 78.84 % โดยมีระยะคลาดเคลื่อนต่อรูเป็น 4.37 มิลลิเมตรต่อรู

บทที่ 9
วิเคราะห์และสรุปผลการทดลอง

9.1 การวิเคราะห์ผลการทดลองและสรุปผลการทดลอง

การวิเคราะห์และสรุปผลการทดลองแบ่งออกเป็น 3 ส่วนคือ

9.1.1 ส่วนโปรแกรมควบคุมการเจาะ

- เนื่องจากการทำงานเป็นแบบไล่อ่านสีทีละพิกเซลและจะเจาะเมื่อเจอจุดที่ต้องการเท่านั้น ดังนั้นการเคลื่อนที่ของเครื่องเจาะจะมีช่วงการวิ่งที่สูญเปล่าค่อนข้างมาก
- คำสั่งที่ใช้ในการเขียนโปรแกรมบางส่วนมีประสิทธิภาพไม่สูงนัก เนื่องด้วยเวลาที่จำกัดผู้จัดเลือกจึงที่จะใช้คำสั่งพื้นฐานเป็นส่วนใหญ่ แม้ว่าจะให้ผลออกมาใกล้เคียงกันแต่คำสั่งที่เขียนขึ้นนั้นมีสภาพการทำงานที่ไม่เร็วนักหรือมีปัญหาได้ในบางกรณี
- จากโปรแกรมคำนวณตำแหน่งเจาะจะมีฟังก์ชั่นที่ตรวจสอบป้องกันการซ้อนทับกันของรูซึ่งใช้ได้ดีในกรณีที่เส้นค่อนข้างห่างกัน แต่โปรแกรมยังไม่สามารถวิเคราะห์รูปได้ หากรูปภาพนั้นมีความซับซ้อนและเส้นใกล้กันมากเกินไป อาจทำให้รูต่างๆนั้นอาจจะหลบกันในรูปแบบที่ไม่เหมาะสม

9.1.2 ส่วนของแผงควบคุม

- เนื่องจากวงจรควบคุมต้องควบคุมอุปกรณ์จำนวนมากในเครื่อง จึงมีแผงวงจรหลายแผงและมีสายไฟเป็นจำนวนโยงมากจากส่วนต่างๆของเครื่องเข้าสู่แผงควบคุม นอกจากนี้ยังมีการเชื่อมโยงสายสัญญาณต่างๆระหว่างแผงอีกด้วย ทำให้เกิดการสับสนวุ่นวายเป็นอย่างมาก
- แผงวงจรบางส่วนถูกออกแบบตามทฤษฎีโดยที่ไม่ได้ทดลองก่อน ทำให้วงจรบางส่วนไม่สามารถทำงานได้ตามที่ออกแบบไว้ เนื่องจากปัจจัยอื่นๆที่ไม่ได้คาดคิดไว้ก่อน

9.1.3 เครื่องเจาะแผ่นพลาสติก

- เนื่องจากงบประมาณและเวลาที่มีจำกัด ทำให้ชิ้นส่วนแต่ละชิ้นถูกออกแบบมาให้สามารถสร้างได้โดยง่าย วัสดุหาได้ทั่วไปตามท้องตลาด และอุปกรณ์บางตัวเป็นชิ้นส่วนที่หาได้รอบตัว
- แม้ว่าชิ้นส่วนแต่ละชิ้นจะถูกออกแบบมาให้ง่ายต่อการผลิต แต่ในความเป็นจริงแล้วการผลิตเครื่องเจาะแผ่นพลาสติกด้วยมือจะทำให้เกิดความคลาดเคลื่อนขึ้นทำให้ประสิทธิภาพบางส่วนลดลง

9.2 แนวทางในการแก้ไข

แนวทางในการแก้ไขจะแบ่งการวิเคราะห์ออกเป็น 3 ส่วนคือ

9.1.1 ส่วนโปรแกรมควบคุมการเจาะ

- เนื่องจากการทำงานเป็นแบบไล่อ่านสีทีละพิกเซล ดังนั้นควรมีปรับปรุงให้การเคลื่อนที่เป็นการวิ่งตามเส้นแทน เนื่องจากจะทำให้เกิดการเจาะตลอดเวลา ซึ่งจะช่วยลดเวลาทำงานได้
- คำสั่งที่ใช้ในการเขียนโปรแกรมบางส่วนอาจมีประสิทธิภาพไม่สูงนัก เนื่องด้วยเวลาที่จำกัดผู้จัดทำจึงเลือกที่จะใช้คำสั่งพื้นฐานเป็นส่วนใหญ่ แม้ว่าจะให้ผลออกมาใกล้เคียงกันแต่คำสั่งที่เขียนขึ้นนั้นอาจทำงานช้าหรือมีปัญหาได้ในบางกรณี ดังนั้นในการพัฒนาเครื่อง
- จากโปรแกรมคำนวณตำแหน่งเจาะซึ่งจะมีฟังก์ชั่นที่ตรวจสอบป้องกันการซ้อนทับกันของรูนั้น จะใช้ได้ดีในกรณีที่เส้นค่อนข้างห่างกัน แต่โปรแกรมยังไม่สามารถวิเคราะห์รูปได้ หากรูปภาพนั้นมีความซับซ้อนและเส้นใกล้กันมากเกินไป อาจทำให้รูต่างๆนั้นอาจจะหลบกันในรูปแบบที่ไม่เหมาะสม ดังนั้นจึงควรปรับปรุงโปรแกรมโดยการเพิ่มเงื่อนไขรูปแบบต่างๆเข้าไป

9.1.2 ส่วนของแผงควบคุม

- เนื่องจากวงจรควบคุมต้องควบคุมอุปกรณ์จำนวนมากในเครื่อง ดังนั้นจึงมีแผงวงจรหลายแผงและมีสายไฟเป็นจำนวนโยงมากจากส่วนต่างๆของเครื่องเข้าสู่แผงควบคุม และนอกจากนี้ยังมีการเชื่อมโยงสายสัญญาณต่างๆระหว่างแผงอีกด้วย ทำให้เกิดการสับสนวุ่นวายเป็นอย่างมาก ดังนั้นการออกแบบแผงวงจรจึงควรรวมแผงวงจรต่างๆเข้ามาไว้ในแผงเดียวกัน และจัดให้มีจุดรวมสัญญาณและจุดส่งสัญญาณต่างๆเป็นกลุ่มๆอย่างเป็นระเบียบ
- แผงวงจรบางส่วนถูกออกแบบตามทฤษฎีโดยที่ไม่ได้ทดลองก่อน ทำให้วงจรบางส่วนไม่สามารถทำงานได้ตามที่ออกแบบไว้ เนื่องจากปัจจัยอื่นๆที่ไม่ได้คาดคิดไว้ก่อน ดังนั้นแผงวงจรจึงควรได้รับการออกแบบหรือคำแนะนำจากผู้ที่มีประสบการณ์

9.1.3 เครื่องเจาะแผ่นพลาสติก

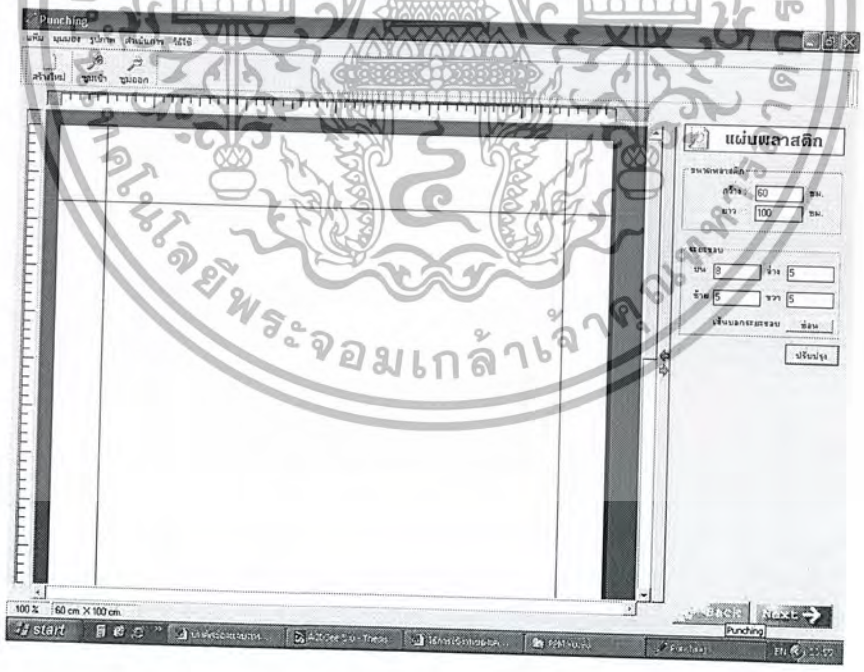
- เนื่องจากงบประมาณและเวลาที่มีจำกัด ทำให้ชิ้นส่วนแต่ละชิ้นถูกออกแบบมาให้สามารถสร้างได้โดยง่าย วัสดุหาได้ทั่วไปตามท้องตลาด และอุปกรณ์บางตัวเป็นชิ้นส่วนที่หาได้รอบตัว ดังนั้นเครื่องอาจดูไม่สวยงามมากนัก และนอกจากนี้ตัวเครื่องยังมีน้ำหนักค่อนข้างมาก ไม่สะดวกต่อการเคลื่อนย้าย จึงควรเลือกวัสดุที่มีน้ำหนักเบา
- แม้ว่าชิ้นส่วนแต่ละชิ้นจะถูกออกแบบมาให้ง่ายต่อการผลิต แต่ในความเป็นจริงแล้วการผลิตด้วยมือจะทำให้เกิดความคลาดเคลื่อนขึ้นทำให้ประสิทธิภาพบางส่วนลดลง ดังนั้นหากต้องการสร้างเพื่อการจำหน่ายแล้วอาจจะต้องออกแบบบางส่วนใหม่ให้สามารถทำงานได้ดีขึ้น และชิ้นส่วนควรสร้างด้วยเครื่องจักรอัตโนมัติเพื่อลดความคลาดเคลื่อนที่เกิดขึ้น

บทที่ 10
วิธีการใช้งานของเครื่องเจาะแผ่นพลาสติก

เครื่องเจาะแผ่นพลาสติกนี้ ถูกออกแบบมาเพื่อช่วยลดขั้นตอนที่ยุ่งยากซับซ้อนในการทำพรมออกมาแต่ละผืน ซึ่งเครื่องเจาะแผ่นพลาสติกเครื่องนี้จะช่วยลดขั้นตอนในการพิมพ์ภาพลายเส้นของลวดลายพรม,การนำลายเส้นที่ได้นั้นไปให้ช่างฝีมือ เขียนภาพลงบนแผ่นพลาสติกและที่สำคัญที่สุดคือช่วยลดขั้นตอนในการใช้คนเจาะรูตามลายเส้นนั้นออกมา เพียงแค่ผู้ใช้ทำการอิมพอร์ตภาพลายเส้นที่ต้องการเจาะเข้าไปในโปรแกรมและกดปุ่มสตาร์ทเท่านั้น เครื่องก็จะทำการเจาะแผ่นพลาสติกออกมาได้ตามต้องการ

10.1 การใช้โปรแกรมคำนวณพิกัดการเจาะ

โปรแกรมนี้จะช่วยให้ผู้ใช้ติดต่อกับเครื่องเจาะได้อย่างสะดวก ด้วยความต้องการให้ผู้ใช้สามารถทำงานได้อย่าง่ายดาย โปรแกรมจึงถูกออกแบบมาให้มีคำสั่งต่างๆถูกแบ่งออกเป็นหมวดหมู่ โดยโปรแกรมจะแบ่งคำสั่งออกเป็น 3 ส่วน คือ การจัดการพลาสติก,การจัดการรูปภาพและการดำเนินงาน เมื่อผู้ใช้เปิดโปรแกรมขึ้นมา จะพบรูปพลาสติกวางเปล่าวางอยู่ และที่แถบด้านขวาจะเป็นที่ที่ให้ผู้ใช้ได้กำหนดคุณลักษณะต่างๆของแผ่นพลาสติก ทั้งขนาดและระยะขอบ แต่ถ้าหากผู้ใช้ยังไม่ทราบขนาดพลาสติกที่แน่นอนก็อาจจะข้ามขั้นตอนนี้ไปก่อนก็ได้ โดยที่แถบขวาด้านล่างจะมีปุ่ม "ถัดไป" เพื่อเข้าไปในส่วนของการจัดการรูปภาพ



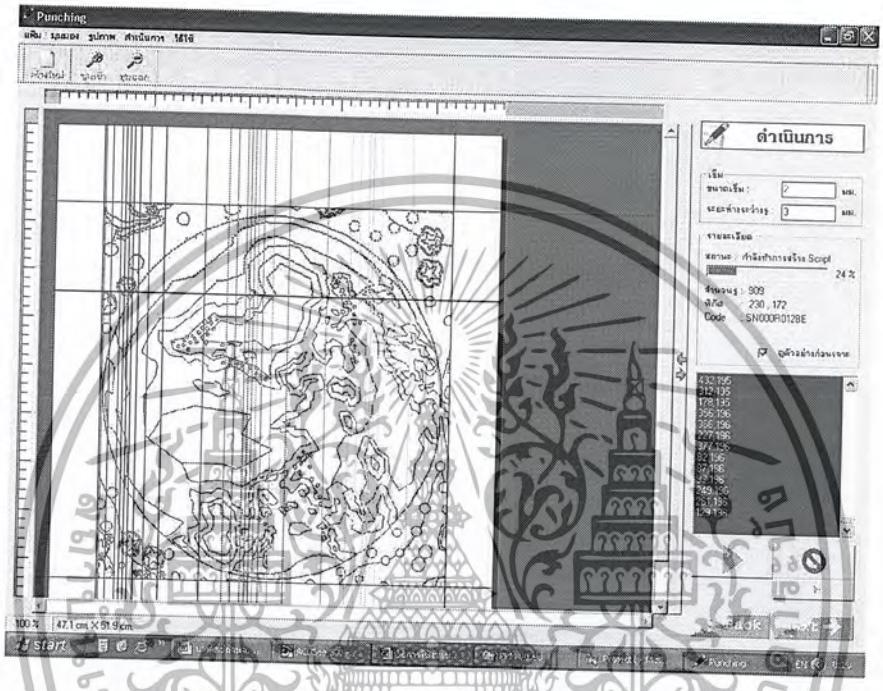
รูปที่ 10-1 แสดงการใช้โปรแกรมในส่วนของการจัดขนาดพลาสติก

ในส่วนของการจัดการรูปภาพ ที่ด้านบนจะมีปุ่มเพื่อให้ผู้ใช้ทำการอิมพอร์ตรูปภาพ โดยผู้ใช้สามารถนำรูปเข้ามาได้สูงสุด 5 รูป หลังจาได้นำรูปเข้ามาแล้ว ผู้ใช้จำเป็นที่จะต้องกำหนดความละเอียดของรูปภาพในหน่วย "จุดต่อนิ้ว (dot per inch,dpi)" เพื่อให้โปรแกรมสามารถปรับขนาดรูปได้ถูกต้องตามขนาดจริง ในหมวดจัดการรูปภาพนี้ผู้ใช้สามารถทำการหมุนรูปภาพหรือลบรูปภาพที่ไม่ต้องการได้ โดยคลิกที่ปุ่ม หมุนซ้าย, หมุนขวาและลบตามลำดับ นอกจากนี้ผู้ใช้สามารถจัดตำแหน่งของรูปภาพได้โดยการกดเมาส์ปุ่มซ้ายค้างไว้และเลื่อนรูปภาพไปวางตามตำแหน่งที่ต้องการ จากนั้นเมื่อจัดตำแหน่งได้เป็นที่พอใจแล้ว หากผู้ใช้ยังไม่ได้กำหนดขนาดแผ่นพลาสติก ผู้ใช้สามารถปรับขนาดได้โดยเลื่อนเมาส์ไปวางที่ขอบของพลาสติกและคลิกเมาส์ค้างและเลื่อนปรับขนาดได้ตามต้องการ ซึ่งสามารถดูขนาดปัจจุบันได้ที่มุมซ้ายล่างของโปรแกรม หรือถ้าหากผู้ใช้ไม่ต้องการจัดขนาดเอง ก็สามารถให้โปรแกรมจัดขนาดได้โดยอัตโนมัติ โดยดับเบิลคลิกส่วนที่ว่างเปล่าของแผ่นพลาสติก โปรแกรมก็จะจัดขนาดให้พอเหมาะกับรูปภาพโดยอัตโนมัติ และผู้ใช้ก็สามารถนำขนาดที่แสดงในโปรแกรมนี้ ไปใช้เพื่อจัดเตรียมแผ่นพลาสติกได้

รูปที่ 10-2 แสดงการใช้โปรแกรมในส่วนของการจัดการรูปภาพ

เมื่อผู้ใช้ได้จัดการกับรูปภาพและพลาสติกจนพอใจแล้ว ก็จะเข้าสู่ส่วนของการดำเนินการ ในส่วนนี้ผู้ใช้สามารถกำหนดขนาดและระยะห่างของรูเข็มได้ แต่เนื่องจากค่าที่เป็นค่าเริ่มต้นที่อยู่ในโปรแกรม ได้ถูกทดสอบมาแล้วว่าเป็นค่าที่เหมาะสมที่สุด ดังนั้นหากไม่มีความจำเป็นใดๆ ผู้ใช้ก็ไม่ควรเปลี่ยนแปลงค่าใดๆ

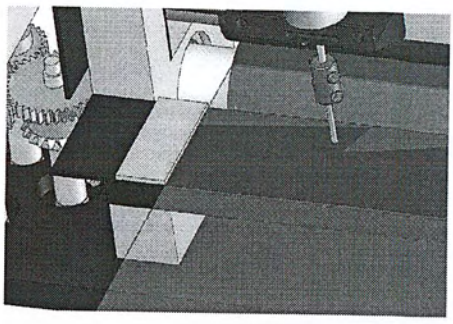
จากนั้นก่อนจะเริ่มทำการเจาะ ผู้ใช้สามารถเลือกได้ว่าต้องการดูตัวอย่างก่อนเจาะหรือไม่ หากต้องการดู โปรแกรมจะทำการคำนวณหาพิกัดต่างๆออกมาก่อน แล้วจะแสดงตัวอย่างให้ผู้ใช้ได้ชมบนหน้าจอ คอมพิวเตอร์ จากนั้นก่อนที่จะเริ่มดำเนินการต่อหรือเริ่มเจาะ ผู้ใช้ต้องนำแผ่นพลาสติกที่จัดเตรียมไว้ไปใส่ใน เครื่องเจาะเสียก่อน

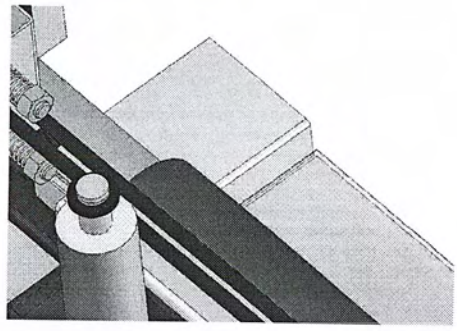


รูปที่ 10-3 แสดงการใช้โปรแกรมในส่วนของการดำเนินการ

10.2 การใช้เครื่องเจาะแผ่นพลาสติก

เมื่อผู้ใช้เปิดสวิทช์แล้ว เครื่องจะทำการยกลูกกลิ้งขึ้นให้โดยอัตโนมัติ จากนั้นให้ผู้ใช้นำแผ่นพลาสติก สอดเข้าไปในเครื่อง โดยให้ขอบริมซ้ายของแผ่นพลาสติกไปชนกับแนวที่ได้กำหนดไว้ให้ และดึงพลาสติกเข้า ไปให้ปลายบนของแผ่นพลาสติกขนานกับแนวของแผ่นเหล็กที่เตรียมไว้ เมื่อจัดตำแหน่งพร้อมแล้ว ให้ผู้ใช้กด ปุ่มเลื่อนลูกกลิ้งลง ตัวเครื่องจะยกลูกกลิ้งลงและหนีบแผ่นพลาสติกไว้กับเครื่อง จากนั้นผู้ใช้ก็สามารถกดปุ่ม เริ่มต้นในโปรแกรมบนคอมพิวเตอร์เพื่อให้เครื่องเจาะเริ่มทำงาน





รูปที่ 10-4 แสดงการจัดระยะพลาสติกก่อนเริ่มทำการเจาะ

เมื่อเครื่องเจาะทำงานเสร็จสิ้นแล้ว ผู้ใช้ก็สามารถกดปุ่มยกลูกกลิ้งขึ้น และนำพลาสติกชิ้นที่สำเร็จไปใช้งานได้ทันที

บรรณานุกรม

[1] อภิชาติ ภู่พลับ, "เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual basic 6", Infopress Develop Book, 2546

[2] ธนพล ฉันจรัสวิชัย, "ปฏิบัติการ Visual Basic สำหรับ Common Windows", ซีเอ็ดยูเคชั่น, 2546

[3] โชติพันธุ์ หล่อเลิศสุนทรและฐิตะพันธุ์ หล่อเลิศสุนทร, "สอนเขียน Visual Basic 6.0 ให้เป็น Project", Soft Express&Publishing, 2543

[4] สมศักดิ์ ศรีขจรเกียรติ, "Visual Basic 6. Teach Yourself", บิบลิโอไฟล์, 2542

[5] "Visual Basic Graphic Programming" John Wiley & Sons, Inc, 1997

[6] A.F. Bakker, "Design of a high speed low friction XY-table", Philips Centre For industrial Technology (CFT)

[7] Hussain Z. Tameem, "Design and development of x-y positioning table using stepper motor and belt drive", Department of Mechanical and Production Engineering Yeshwantrao Chavan college of engineering Wanadongri, Nagpur

[8] วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล, "เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช" ฉบับ AT89C5x ของ Atmel", Innovative Experiment Co.,Ltd.

[9] Joseph E. Shigley, Charles R. Mischke, Richard G. Budynas, "Mechanical Engineering Design", Seventh Edition, McGrawHill.

[10] www.thaiio.com

[11] www.pantip.com

[12] www.howstuffwork.com

[13] www.vbcode.com

ภาคผนวก

ภาคผนวก ก

วิชวลเบสิกซอร์สโค้ด (Visual Basic Source Code)

```
'*************************************************************************************************************
' Control PART
'*************************************************************************************************************
Private Sub btnStart_Click()
    Dim p As Integer
    Dim X, Y
    Dim Rim

    '-----------------
    ' จัดการรูปภาพ
    '-----------------
    If NumPic = 0 Then
        MsgBox "คุณยังไม่ได้ใส่รูปภาพ", vbInformation, "คำเตือน"
        Exit Sub
    End If
    For p = 1 To 5
        If PicProp(p, 3) <> 0 Then ' เช็คว่ามีตัวตนอยู่ไหม
            If Shape1(p).Visible = True Then
                MsgBox "มีภาพที่ยังไม่ได้กำหนดความละเอียด", vbInformation, "คำเตือน"
                Exit Sub
            End If
        End If
    Next

    MousePointer = vbHourglass
    btnStart.Enabled = False
```

```
    mnuStart.Enabled = False
    btnStop.Enabled = True
    mnuStop.Enabled = True
    btnBack.Enabled = False
    btnExit.Enabled = False
    btnRedo.Visible = False
    mnuRedo.Visible = False
    For p = 1 To Toolbar1.Buttons.Count
       Toolbar1.Buttons(p).Enabled = False
    Next

    ' ปิดเมนู
    mnuFile.Enabled = False
    mnuView.Enabled = False
    mnuImage.Enabled = False

    ' ปรับเป็น 100%
    If Factor <> 1 Then
    lblStatus.Caption = "สถานะ : กำลังปรับอัตราส่วนเป็น100% "
    DoEvents
       Factor = 1
       PlasticSize
       ArrangeScroll
       For p = 1 To picLayout.UBound
          ArrangePicture p
          'DoEvents
       Next
       StatusBar.Panels.Item(1).Text = Factor * 100 & " % "
    End If
```

```
PicPlasticTemp.Cls
picPlastic.Picture = PicPlasticTemp.Picture
' Copy รูปลงบน picPlastic

    lblStatus.Caption = "สถานะ : กำลังวาดภาพลงบนพลาสติก"
    DoEvents
    For p = 1 To 5
      If PicProp(p, 0) <> 0 Then ' เช็คว่ามีตัวตนอยู่ไหม
        If PicShowListbtn(p).Appearance = vb3D Then
            picPlastic.PaintPicture picLayout(p), picLayout(p).Left, picLayout(p).Top,
picLayout(p).Width, picLayout(p).Height, _
                    0, 0, picLayout(p).Width, picLayout(p).Height, vbSrcAnd

        End If
        picLayout(p).Visible = False
      End If
    Next

    ' ตั้งค่าต่างๆ
    lblStatus.Caption = "สถานะ : กำลังทำการสร้าง Script "
    NeedleSize = Round(txtNeedleSize.Text / 0.88) ' Diameter 1 pixel = 0.88 mm.
    picPlastic.DrawWidth = NeedleSize
    PicPlasticTemp.DrawWidth = NeedleSize
    Gap = Round(txtGapNeedle.Text / 0.88) - NeedleSize
    txtScript.Text = ""
    StopPunching = False
    Quatity = 0
    Qty = 0
```

```
' ใช้หัวเจาะ 4 หัว แต่ละหัวห่างกัน 150 Pixels = 132 mm.
' |---3.6--|----------------------------------52.8----------------------------------|---3.6---|        cm.
'           0---------------132---------------264-------------396------------528                    mm.
'          40---------------190---------------340-------------490------------640                   pixel

' เส้นแสดงตำแหน่งโซลินอยด์
Solinoid1.Visible = True
Solinoid2.Visible = True
Solinoid3.Visible = True
Solinoid4.Visible = True
SolinoidLine.Visible = True
Solinoid1.Move 40, 0, 1, picPlastic.Height
Solinoid2.Move 190, 0, 1, picPlastic.Height
Solinoid3.Move 340, 0, 1, picPlastic.Height
Solinoid4.Move 490, 0, 1, picPlastic.Height
SolinoidLine.Move 0, picPlastic.Top, picPlastic.Width, 1
'ยกเลิกการปรับขนาดพลาสติก
PlasRimRight.Visible = False
PlasRimBottom.Visible = False
PlasRimBoth.Visible = False

'หาจำนวนหัวเจาะสูงสุดที่ใช้

    If picPlastic.Width - txtVRim(2).Text * 10 * 1 / 0.88 > Round(txtVRim(1) * 10 * 1 / 0.88) And
picPlastic.Width - txtVRim(2).Text * 10 * 1 / 0.88 <= 190 Then
      Max_k = 1
    End If
    If picPlastic.Width - txtVRim(2).Text * 10 * 1 / 0.88 > 190 And picPlastic.Width - txtVRim(2).Text
* 10 * 1 / 0.88 <= 340 Then
      Max_k = 2
```

```
        End If
        If picPlastic.Width - txtVRim(2).Text * 10 * 1 / 0.88 > 340 And picPlastic.Width - txtVRim(2).Text
* 10 * 1 / 0.88 <= 490 Then
            Max_k = 3
        End If
        If picPlastic.Width - txtVRim(2).Text * 10 * 1 / 0.88 > 490 And picPlastic.Width - txtVRim(2).Text
* 10 * 1 / 0.88 <= 640 Then
            Max_k = 4
        End If

        CurrentPosX = 3.6 * 10 / 0.88
        CurrentPosY = 14 * 10 / 0.88
        MousePointer = vbDefault

        'TimeTest.Enabled = True
        'MSComm.PortOpen = True
        If Check1.value = Checked Then 'ถ้าดูตัวอย่างก่อนเจาะก็ต้องสร้าง Script ก่อน
            CrateScript
        Else
        If Check1.value = Unchecked Then
            PunchNow
        End If
        End If
End Sub

' กรณีที่ต้องการสร้างสคริปท์
Private Sub CrateScript()

    ' ทำตัวอย่าง
    PicPlasticTemp.Width = picPlastic.Width
```

```
PicPlasticTemp.Height = picPlastic.Height

For j = Round(txtHRim(1) * 10 * 1 / 0.88) To Round((txtT1P_hgt - txtHRim(2)) * 10 * 1 / 0.88)
DoEvents
    For i = 40 To 189
        For k = 1 To Max_k
            If StopPunching = True Then Exit Sub
            FindPoint
        Next k
    Next i
Next j

Finished = True
txtScript.Text = txtScript.Text + "Finish"
txtScript.SelStart = Len(txtScript.Text)

lblStatus.Caption = "สถานะ : รอคอย"
lblPercent.Caption = "100%"
ProgressBar1.value = 100

If Check1.value = Checked Then      'เช็คอีกครั้ง เผื่อเปลี่ยนใจ
' ก็อปรูปบน picplasticTemp > PicpIstic
    MousePointer = vbHourglass
    HRim(1).Visible = False
    HRim(2).Visible = False
    VRim(1).Visible = False
    VRim(2).Visible = False
    Solinoid1.Visible = False
    Solinoid2.Visible = False
    Solinoid3.Visible = False
```

```
        Solinoid4.Visible = False
        SolinoidLine.Visible = False
        ResizePicture PicPlasticTemp, picPlastic, _
        0, 0, _
        PicPlasticTemp.Width, PicPlasticTemp.Height, _
        0, 0, _
        picPlastic.Width, picPlastic.Height
        MousePointer = vbDefault

    Toolbar1.Buttons(3).Enabled = True
    Toolbar1.Buttons(4).Enabled = True
    btnContinue.Visible = True
    mnuContinue.Visible = True
    Else

    End If

End Sub
Private Sub btnContinue_Click()

    btnContinue.Visible = False
    mnuContinue.Visible = False
    Toolbar1.Buttons(3).Enabled = False
    Toolbar1.Buttons(4).Enabled = False
    Check1.value = vbUnchecked
    Check1.Enabled = False
    btnStart_Click
End Sub

'*************************************************************************************************************
```

```
Private Sub PunchNow()

    ' เปิด port
    OpenPort

    ' คำสั่ง Reset > ใช้วิธีทำให้เข็มวิ่งไปชน Limit ขวาแล้วมันจะกลับมารีเซ็ทตัวเอง
    MSComm.Output = "SN000R200NE" ' ใส่ให้วิ่งขวาเกินไปเลย
    ' แต่เนื่องจากเราต้องรอมันรีเซ็ทเสร็จ แล้วค่อยส่งข้อมูล ดังนั้นเราจะนับเวลาถอยหลังสักพัก ก่อนจะเปิด
นาฬิกาทำงาน
    TimerReset.Enabled = True

      GetW = True
      GetY = True
      j = Round(txtHRim(1) * 10 * 1 / 0.88) - 1   ' ที่ต้อง -1 เนื่องจากใน GetW ด้านล่างจะทำการเพิ่มค่า
ก่อน แล้วจึงหาพิกัด
      i = 40 - 1
      k = 1 - 1
      Finished = False
      Found = False
      PastTime = 0
      WTime = Val(txtWTime.Text)
      txtWTime.Visible = False
      Check1.Enabled = False
      Label5.Enabled = False

End Sub

Private Sub TimerReset_Timer()
Static CT ' Count
```

```
        CT = CT + TimerReset.Interval
        If CT = 8000 Then ' จับเวลารีเซ็ทตัวเอง 8 วินาที
            ' เปิดนาฬิกาอ่านข้อมูล
            TimerReceive.Enabled = True
            'เปิดนาฬิกาทำงาน2
            Timer2.Enabled = True
            'เปิดนาฬิกานับเวลา
            Hr = 0
            Min = 0
            Sec = 0
            TimerWorkTime = True

          TimerReset.Enabled = False
          CT = 0
        End If

End Sub

Private Sub Timer2_Timer()
    ' ถ้าได้ W คือ MCS ได้รับโค้ดเรียบร้อยแล้ว ดังนั้นเตรียมโค้ดใหม่ได้เลย
  If GetW = True Then
     ' Timer2.Enabled = False
      'Label21.Caption = "T2-->Close"
      GetW = False
      Received = True

      If GetH = False Then
      'เตรียมพิกัดใหม่
      Do Until Found
      'เพิ่มค่า i,j,k
```

```
            k = k + 1
            If k > Max_k Then
               k = 1
               i = i + 1
               If i >= 190 Then
                  i = 40
                  j = j + 1
                  DoEvents
                  If j > Round((txtT1P_hgt - txtHRim(2)) * 10 * 1 / 0.88) Then
                        Finished = True
                        Exit Do
                  End If
               End If
            End If
            '------------
         FindPoint
         Loop
         Else
            GetH = False
         End If

            'Timer2.Enabled = True
            'Label21.Caption = "T2-->Open"
            PastTime = 0

      End If
```

' ถ้าได้ H คือค่าล่าสุดที่ส่งไปมันเสีย ต้องรีเซ็ทตำแหน่งใหม่ให้ตรงกัน และใช้พิกัดเก่าคำนวณโค้ดใหม่แล้วส่งไปเจาะอีกที

```
   If GetH = True Then
```

```
        CurrentPosX = 3.6 * 10 / 0.88
        CurrentPosY = Lastj ' เนื่องจาก CurrentPosY จะถูกกำหนดใน CreateCode และตอนนี้มันได้ค่าใหม่ไปแล้วทั้งๆที่หัวเจาะยังอยู่ที่เก่า เราจึงต้องบอกค่าที่แท้จริงให้มัน
        CreateCode Lasti, Lastj, Lastk
        Label31.Caption = Lasti
        Label32.Caption = Lastj
        Label33.Caption = Lastk
        MSComm.Output = PositionCode ' ส่งเสร็จแล้วจะได้ค่า W กลับมาให้หาโค้ดใหม่ แต่ก่อนหน้านี้หาไปแล้ว จึงต้องกำหนดให้ GetW ไม่ต้องทำงานถ้ายัง GetH อยู่
    End If

    'ถ้าได้ Y คือ MCS เรียกขอโค้ดใหม่ ก็ส่งโค้ดที่หาได้ไป
    If GetY = True Then
     '  Timer2.Enabled = False
      'Label21.Caption = "T2-->Close"
      GetY = False
         Lasti = i
         Lastj = j
         Lastk = k
      CreateCode i, j, k
      If Finished = True Then
           PositionCode = "O" 'OK
      End If
      On Error Resume Next
      MSComm.Output = PositionCode
      Label24.Caption = PositionCode

      Found = False ' เตรียมหาค่าใหม่
      If Finished = False Then
'        Timer2.Enabled = True
```

```
        Label21.Caption = "T2-->Open"
        PastTime = 0
        Received = False
      Else
        Complete
      End If
    End If

  CountTimeError
End Sub

'#################################################################################
'############################ SHARE FUNCTIONS ###################################
'#################################################################################
Private Sub FindPoint()
    Dim longcolor, hexcolor, dy, dx, rangeX
    Dim SideDrill As Boolean

        SolinoidLine.Top = j
        On Error Resume Next
        ProgressBar1.value = (SolinoidLine.Top - Round(txtHRim(1) * 10 * 1 / 0.88)) /
(Round((txtT1P_hgt - (txtHRim(2)) - txtHRim(1)) * 10 * 1 / 0.88)) * 100
        lblPercent.Caption = Round(ProgressBar1.value, 0) & " %"

        ' เช็คว่าอยู่ในขอบเขตหรือไม่
            If i + (k - 1) * 150 < Round(txtVRim(1) * 10 * 1 / 0.88) Then
              k = k + 1
            End If
            If i + (k - 1) * 150 > Round((txtT1P_wid - txtVRim(2)) * 10 * 1 / 0.88) Then
              Exit Sub
```

```
        End If

    'เช็คสี
    longcolor = picPlastic.Point(i + (k - 1) * 150, j)

    'ถ้าเจอเส้นดำแล้ว
    If longcolor < 13158600 Then '&HC8C8C8
        '-----------------------------------------------------------------------------------------
        'เช็ครอบข้าง
        SideDrill = False ' กำหนดค่าเริ่มต้นว่า รอบข้างยังไม่ได้ถูกเจาะ
        For dy = 0 To NeedleSize + Gap / 2
            rangeX = Round(Sqr(Abs((NeedleSize / 2) ^ 2 - (dy / 2) ^ 2))) + Gap
            For dx = -rangeX To rangeX
                'ถ้าเจอจุดที่ถูกเจาะแล้ว ให้บอกว่าเจาะแล้ว
                If picPlastic.Point(i + (k - 1) * 150 + dx, j - dy) = &HFF& Then
                SideDrill = True
                End If
                'ถ้าถูกเจาะแล้วให้ออกไป
                If SideDrill = True Then Exit For
            Next
            If SideDrill = True Then Exit For
        Next

        '-----------------------------------------------------------------------------------------
    'ถ้ารอบข้างยังไม่ถูกเจาะ ก็เจาะรูนี้ซะ
        If SideDrill = False Then
            'เส้น
            Solinoid1.Left = i
            Solinoid2.Left = i + 150
            Solinoid3.Left = i + 300
```

```
                Solinoid4.Left = i + 450
                Quatity = Quatity + 1
                lblQuatity.Caption = "จำนวนรู : " & Str(Quatity)
                lblOrdinate.Caption = "พิกัด      : " & Round((i + (k - 1) * 150) * 0.88) & " , " &
Round(j * 0.88)
                picPlastic.PSet (i + (k - 1) * 150, j), &HFF&
                PicPlasticTemp.PSet (i + (k - 1) * 150 - 1, j - 1), PicPlasticShadow.BackColor
                PicPlasticTemp.PSet (i + (k - 1) * 150, j), picBackground.BackColor
                'CreateCode i, j, k
                Found = True
                lblcode.Caption = "Code      : " & PositionCode
                txtScript.Text = txtScript.Text & i
                txtScript.Text = txtScript.Text & "   " & i + (k - 1) * 150 & "," & j
                txtScript.Text = txtScript.Text & "   " & PositionCode & vbCrLf
                txtScript.SelStart = Len(txtScript.Text)
            End If
        End If 'เส้นคำ
End Sub

Private Sub CreateCode(ByVal u As Integer, ByVal v As Integer, ByVal s As Integer)

        PositionCode = ""
            ' สร้าง Code การสั่งงาน
            ' หาระยะการเคลื่อนที่
            DiffX = u - CurrentPosX
            DiffY = v - CurrentPosY

            'เริ่มต้น
            PositionCode = "S" 'Start
```

```
'หาทิศการเคลื่อนที่ของมอเตอร์ตัวที่1 (ดูว่าค่าเป็น + หรือ - )
If DiffY = 0 Then
   PositionCode = PositionCode + "N"
Else
If DiffY > 0 Then
   PositionCode = PositionCode + "L"
Else
If DiffY < 0 Then
   PositionCode = PositionCode + "R"
End If
End If
End If

'ใส่โค้ดจำนวนพัลส์
   DiffY = Abs(DiffY)
   Select Case Len(DiffY)
   Case 1
      PositionCode = PositionCode + "00" + Mid(Str(DiffY), 2, 1)
   Case 2
      PositionCode = PositionCode + "0" + Mid(Str(DiffY), 2, 2)
   Case 3
      PositionCode = PositionCode + Mid(Str(DiffY), 2, 3)
   End Select

'หาทิศการเคลื่อนที่ของมอเตอร์ตัวที่2 (ดูว่าค่าเป็น + หรือ - )
If DiffX = 0 Then
   PositionCode = PositionCode + "N"
Else
If DiffX > 0 Then
```

```
        PositionCode = PositionCode + "R"
    Else
    If DiffX < 0 Then
        PositionCode = PositionCode + "L"
    End If
    End If
    End If
    ' ใส่โค้ดจำนวนพัลส์
    DiffX = Abs(DiffX)
    Select Case Len(DiffX)
    Case 1
        PositionCode = PositionCode + "00" + Mid(Str(DiffX), 2, 1)
    Case 2
        PositionCode = PositionCode + "0" + Mid(Str(DiffX), 2, 2)
    Case 3
        PositionCode = PositionCode + Mid(Str(DiffX), 2, 3)
    End Select

'ใส่เบอร์โซลินอยด์ + End
Select Case s
Case 1
    PositionCode = PositionCode + "A"
Case 2
    PositionCode = PositionCode + "B"
Case 3
    PositionCode = PositionCode + "C"
Case 4
    PositionCode = PositionCode + "D"
End Select
```

```
            ' จบ
             PositionCode = PositionCode + "E" 'End

            'กำหนดตำแหน่งปัจจุบัน
            CurrentPosX = u
            CurrentPosY = v

End Sub

Private Sub OpenPort()
   MSComm.CommPort = 1
   MSComm.Settings = "9600,n,8,1"
   MSComm.PortOpen = True
End Sub
Private Sub TimerReceive_Timer()
    MSComm.DTREnable = False
    MSComm.DTREnable = True
    MSComm.InputLen = 0
    On Error Resume Next
    DATA = MSComm.Input
    If DATA = "W" Then
       Label24.Caption = DATA
       GetW = True
    Else
    If DATA = "Y" Then
       Label24.Caption = DATA
       GetY = True
    Else
    If DATA = "H" Then
       Label30.Caption = DATA
```

```
            GetH = True
        End If
        End If
        End If
End Sub

Private Sub TimerWorkTime_Timer()
Dim tsec As String
Dim tmin As String
Dim thr As String

  Sec = Sec + TimerWorkTime.Interval / 1000

  If Sec = 60 Then
     Min = Min + 1
     Sec = 0
  End If

  If Min = 60 Then
     Hr = Hr + 1
     Min = 0
  End If

  ' ทำเป็น 00
  If Sec < 10 Then
     tsec = "0" + Mid(Str(Sec), 2, 1)
  Else
     tsec = Str(Sec)
  End If
```

```
    If Min < 10 Then
       tmin = "0" + Mid(Str(Min), 2, 1)
    Else
       tmin = Str(Min)
    End If

    If Hr < 10 Then
       thr = "0" + Mid(Str(Hr), 2, 1)
    Else
       thr = Str(Hr)
    End If

    lblWorkTime.Caption = "เวลางาน : " & thr & " : " & tmin & " : " & tsec

End Sub

Private Sub CountTimeError()
  ' ไม่ส่งคำขอ > จับเวลา ถ้ารอนานเกินไปให้แจ้งผู้ใช้(นับช่วงเวลาตั้งแต่ ส่งโค้ดจนได้รับคำขอโค้ดอีกครั้ง)
If MSComm.PortOpen = True Then
      PastTime = PastTime + Timer1.Interval / 1000 ' pasttime > sec
      lblWTime.Caption = " Waitting Time : " & Round(PastTime, 1) & " Sec"

      If Received = False Then
        If Round(PastTime, 1) - Round(PastTime, 0) = 0 Then 'ดูเศษ ถ้าเท่ากับ 0 ให้ส่งค่าใหม่ทุกๆ1
วินาที
           MSComm.Output = PositionCode
           Label24.Caption = Round(PastTime, 0)
        End If
        If PastTime >= 5 Then
           ShowError
```

```
        End If
    End If

        If PastTime > WTime Then '  60 sec = 1 min
            ShowError
        End If
End If
End Sub

Private Sub ShowError()
Dim MsgPress
  MsgPress = MsgBox("เครื่องเจาะเกิดข้อผิดพลาด กรุณาตรวจสอบความเรียบร้อยก่อนดำเนินการต่อหรือ
ยกเลิกการดำเนินการ", vbRetryCancel + vbCritical, "คำเตือน")
            If MsgPress = 4 Then ' ไม่ยกเลิก ทำงานต่อ
              PastTime = 0
              Label24.Caption = ""
            Else
              MsgPress = MsgBox("คุณแน่ใจว่าต้องการหยุดการทำงาน?", vbYesNo +
vbQuestion, "Stop!")
                If MsgPress = vbYes Then ' แน่ใจว่าจะยกเลิก
                  MSComm.Output = "O"
                  TimerReceive.Enabled = False
                  Timer1.Enabled = False
                  Timer2.Enabled = False
                  TimerWorkTime.Enabled = False
                  Complete
                  lblStatus.Caption = "สถานะ : ถูกยกเลิก"
                End If
            End If
End Sub
```

```
Private Sub Complete()
    ProgressBar1.value = 100
    lblPercent.Caption = "100 %"
    TimerReceive.Enabled = False
    btnStart.Enabled = True
    mnuStart.Enabled = True
    btnStop.Enabled = False
    mnuStop.Enabled = False
    btnRedo.Visible = True
    mnuRedo.Visible = True
    btnRedo.Enabled = True
    mnuRedo.Enabled = True
    btnExit.Enabled = True
    mnuExit.Enabled = True
    lblStatus.Caption = "สถานะ : เสร็จสิ้น."
    Check1.Enabled = True
    Label5.Enabled = True
    Toolbar1.Buttons(1).Enabled = True
    Toolbar1.Buttons(3).Enabled = False
    Toolbar1.Buttons(4).Enabled = False
    Toolbar1.Buttons(6).Enabled = True
    MSComm.PortOpen = False
End Sub

'##########################################################################
'##########################################################################
'##########################################################################

Private Sub btnStop_Click()
```

```
Dim intKeyPress
intKeyPress = MsgBox("คุณแน่ใจว่าต้องการหยุดการทำงาน?", vbYesNo + vbQuestion, "Stop!")
If intKeyPress = vbNo Then Exit Sub

' ปิดนาฬิกาทั้งหมด
TimerReceive.Enabled = False
Timer1.Enabled = False
Timer2.Enabled = False
TimerReset = False
TimerWorkTime = False

StopPunching = True
btnStart.Enabled = True
mnuStart.Enabled = True
btnStop.Enabled = False
mnuStop.Enabled = False
lblStatus.Caption = "สถานะ : ถูกยกเลิก"
btnRedo.Visible = True
mnuRedo.Visible = True
btnContinue.Visible = False
btnContinue.Visible = False
Toolbar1.Buttons(1).Enabled = True
btnExit.Enabled = True
'txtWTime.Visible = True
Check1.Enabled = True
Label5.Enabled = True
   Solinoid1.Visible = False
   Solinoid2.Visible = False
   Solinoid3.Visible = False
   Solinoid4.Visible = False
```

```
    SolinoidLine.Visible = False

    ' ปิดพอร์ต
    On Error Resume Next
    MSComm.PortOpen = False
End Sub

Private Sub btnRedo_Click()
    Dim p As Integer

    MousePointer = vbHourglass
    btnBack.Enabled = True
    btnContinue.Visible = False
    mnuContinue.Visible = False
    Finished = False
    PicPlasticTemp.Cls
    picPlastic.Picture = PicPlasticTemp.Picture

      HRim(1).Visible = True
      HRim(2).Visible = True
      VRim(1).Visible = True
      VRim(2).Visible = True

  For p = 1 To picLayout.UBound
     If PicShowListbtn(p).Appearance = vb3D Then
        On Error Resume Next
        ArrangePicture p
        picLayout(p).Visible = True
     End If
  Next
```

```
For p = 1 To Toolbar1.Buttons.Count
   Toolbar1.Buttons(p).Enabled = True
Next
lblStatus = "สถานะ : -"
lblOrdinate = "พิกัด      : -"
lblQuatity = "จำนวนรู : 0"
lblcode = "Code      :"
lblPercent = "0%"
ProgressBar1.value = 0
txtScript.Text = ""
btnRedo.Visible = False
mnuRedo.Visible = False

  ' เปิดเมนู
mnuFile.Enabled = True
mnuView.Enabled = True
mnuImage.Enabled = True

    'การปรับขนาดพลาสติก
   PlasRimRight.Visible = True
   PlasRimBottom.Visible = True
   PlasRimBoth.Visible = True

MousePointer = vbDefault

End Sub
```

ภาคผนวก ข

แอสเซมบลีซอร์สโค้ด (Assembly Source Code)

```
LMf             EQU   P0.0
LMb             EQU   P0.1
LMup            EQU   P0.2
LMdown          EQU   P0.3
LMleft          EQU   P0.4
LMright         EQU   P0.5
SWdown          EQU   P0.7

DCMotorA        EQU   P1.0
DCMotorB        EQU   P1.1

LEDWait         EQU   P1.6
LEDReady        EQU   P1.7

SWup            EQU   P3.2
;SWdown         EQU   P3.3

SoA             EQU   P1.2
SoB             EQU   P1.3
SoC             EQU   P1.4
SoD             EQU   P1.5

CR              EQU   0Dh
LF              EQU   0Ah

;################ Internal Memory ######################
Mvalue          EQU   51h
```

```
Mrange      EQU   50h

Mend        EQU   4Ah
Msoli       EQU   49h

Mux0        EQU   48h
Mux1        EQU   47h
Mux2        EQU   46h
Munlr       EQU   45h

Mdx0        EQU   44h
Mdx1        EQU   43h
Mdx2        EQU   42h
Mdnlr       EQU   41h

Mstart      EQU   40h

;-------------------------------------------------------------
            ORG   0000H
            LJMP  Main
;#############################################################
;#########    Interrupt Vector          ####
;#############################################################
;INT0 (P3.2)
            ORG   0003h
            RETI
;TF0 (Timer0)
            ORG   000Bh
            RETI
;INT1 (P3.3)
            ORG   0013h
```

```
                RETI
;TF1 (Timer1)
                ORG     001Bh
                RETI

;RI and TI (Serial)
                ORG     0023h
                RETI

;#########################################################
;#############       Main        ###############
;#########################################################
                ORG     0030h

Main:           mov     P0,#0FFh
                mov     P1,#0FFh
                mov     P2,#0FFh
                mov     P3,#0FFh
                mov     IE, #00h            ;Disable all interrupt
                clr     A

                lcall   Delay_1s
                lcall   Delay_1s

                lcall   Reset1
                lcall   InitSerial

WaitLMfb:       jnb     LMf, WaitLMfb_1     ;if LMf pressed jump
                lcall   SetLEDWait
                jb      LMf, WaitLMfb   ;--> Wait plastic front
```

```
WaitLMfb_1:   jnb     LMb, WaitSWDown      ;LMb pressed
              lcall   SetLEDWait
              jb      LMb, WaitLMfb  ;--> Wait plastic back

;################ Place plastic #################
WaitSWDown:   jnb     LMdown, Get
              lcall   SetLEDReady
              jb      SWdown, WaitSwDown
              lcall   DCMotorDown
              jnb     LMdown, Get
              ljmp    WaitSwDown_1

WaitSwDown_1: jnb     LMup, Get
              jb      SWup, WaitLMfb
              lcall   DCMotorUp
              jnb     LMup, Get
              ljmp    WaitLMfb

Get:
              lcall   SetLEDReady
              lcall   GetData
              mov     a, Mend
              CJNE    A, #'E', Main_1  ;รับข้อมูลยังไม่ครบ, Error jump

;--------- Run JOB -----------

              lcall   RunJob
              nop
;
Main_1:
              lcall   ClrMen
```

```
                jb      LMdown, WaitLMfb      ;Jump if not LMdown


                jb      LMf, WaitLMfb  ;พลาสติกหลุด
                jb      LMb, WaitLMfb
                ljmp    WaitLMfb
;End WaitLMfb
End Main

;#################### Run Job ###############
RunJob:
;Mend        EQU   4Ah
;Msoli       EQU   49h

;Mux0        EQU   48h
;Mux1        EQU   47h
;Mux2        EQU   46h
;Munlr       EQU   45h

;Mdx0        EQU   44h
;Mdx1        EQU   43h
;Mdx2        EQU   42h
;Mdnlr       EQU   41h
;Mstart      EQU   40h

            nop
ChMd_N:         MOV   A, Mdnlr
            CJNE  A, #'N', ChMd_L
            ljmp  ChMu_N
ChMd_L:         CJNE  A, #'L', ChMd_R
            lcall MdLRunSetup
```

```
                ljmp    ChMu_N
ChMd_R:         CJNE    A, #'R', ChError
                lcall   MdRRunSetup
                ljmp    ChMu_N

ChMu_N:         MOV     A, MunIr
                CJNE    A, #'N', ChMu_L
                ljmp    ChSoli_N
ChMu_L:         CJNE    A, #'L', ChMu_R
                lcall   MuLRunSetup
                ljmp    ChSoli_N
ChMu_R:         CJNE    A, #'R', ChError
                lcall   MuRRunSetup
                ljmp    ChSoli_N
ChSoli_N:       MOV     A, MsoIi
                CJNE    A, #'N' ,ChSoli_A
                ljmp    EndJob
ChSoli_A:       CJNE    A, #'A' ,ChSoli_B
                lcall   PunchSoliA
                ljmp    EndJob
ChSoli_B:       CJNE    A, #'B' ,ChSoli_C
                lcall   PunchSoliB
                ljmp    EndJob
ChSoli_C:       CJNE    A, #'C' ,ChSoli_D
                lcall   PunchSoliC
                ljmp    EndJob
ChSoli_D:       CJNE    A, #'D' ,ChError
                lcall   PunchSoliD
                ljmp    EndJob

ChError:
```

```
                lcall   Delay_02s
                mov     A, #'E'
                lcall   Send
                ljmp    EndJob2
EndJob:
                lcall   Delay_02s
                mov     A, #'Y'
                lcall   Send

EndJob2:
                ret
;End RunJob

;############ Motor Down ####################
;#---------------- MdLSetup ----------------#
MdLRunSetup:
                mov     Mrange, #100d
                mov     Mvalue, Mdx2
                lcall   MdLRun

                mov     Mrange, #10d
                mov     Mvalue, Mdx1
                lcall   MdLRun

                mov     Mrange, #01d
                mov     Mvalue, Mdx0
                lcall   MdLRun
                ret
;EndMdLRunSetup

;---------------------- Run Pluse ----------------------
```

```
;Input   :       Mvalue = ค่า
;        :       Mrange = หลัก (100, 10, 1)
;-------------------------------------------------------------

MdLRun:
                MOV     R5, Mvalue
                CJNE    R5, #00h, MdLRun_1
                LJMP    EndMdLRun
MdLRun_1:       MOV     R4, Mrange
MdLRun_2:       LCALL   LowMotorLeft
                DJNZ    R4, MdLRun_2
                DJNZ    R5, MdLRun_1
EndMdLRun:
                RET
;End MdLRun


;#---------------- MdRSetup ----------------#
MdRRunSetup:
                mov     Mrange, #100d
                mov     Mvalue, Mdx2
                lcall   MdRRun

                mov     Mrange, #10d
                mov     Mvalue, Mdx1
                lcall   MdRRun

                mov     Mrange, #01d
                mov     Mvalue, Mdx0
                lcall   MdRRun
                ret
```

```
;EndMdRRunSetup
;---------------------- Run Pluse ----------------------
;Input   :       Mvalue = ค่า
;       :       Mrange = หลัก (100, 10, 1)
;-------------------------------------------------------------
MdRRun:
                MOV     R5, Mvalue
                CJNE    R5, #00h, MdRRun_1
                LJMP    EndMdRRun
MdRRun_1:       MOV     R4, Mrange
MdRRun_2:       LCALL   LowMotorRight
                DJNZ    R4, MdRRun_2
                DJNZ    R5, MdRRun_1
EndMdRRun:
                RET
;End MdRRun

;############ End Motor Down ####################

;############ Motor Up  ####################
;#---------------------- MuLSetup ---------------------- #
MuLRunSetup:
                mov     Mrange, #100d
                mov     Mvalue, Mux2
                lcall   MuLRun

                mov     Mrange, #10d
                mov     Mvalue, Mux1
                lcall   MuLRun

                mov     Mrange, #01d
```

```
                mov     Mvalue, Mux0
                lcall   MuLRun
                ret
;EndMuLRunSetup

;---------------------- Run Pluse ----------------------
;Input   :      Mvalue = ค่า
;        :      Mrange = หลัก (100, 10, 1)
;-------------------------------------------------------

MuLRun:
                MOV     R5, Mvalue
                CJNE    R5, #00h, MuLRun_1
                LJMP    EndMuLRun
MuLRun_1:       MOV     R4, Mrange
MuLRun_2:       LCALL   HighMotorLeft
                DJNZ    R4, MuLRun_2
                DJNZ    R5, MuLRun_1
EndMuLRun:
                RET
;End MuLRun

;#---------------------- MuRSetup ---------------------- #
MuRRunSetup:
                mov     Mrange, #100d
                mov     Mvalue, Mux2
                lcall   MuRRun

                mov     Mrange, #10d
                mov     Mvalue, Mux1
                lcall   MuRRun
```

```
                mov     Mrange, #01d
                mov     Mvalue, Mux0
                lcall   MuRRun
                ret
;EndMuRRunSetup

;----------------------- Run Pluse -----------------------
;Input   :       Mvalue = ค่า
;        :       Mrange = หลัก (100, 10, 1)
;-------------------------------------------------------------

MuRRun:
                MOV     R5, Mvalue
                CJNE    R5, #00h, MuRRun_1
                LJMP    EndMuRRun
MuRRun_1:       MOV     R4, Mrange
MuRRun_2:       LCALL   HighMotorRight
                DJNZ    R4, MuRRun_2
                DJNZ    R5, MuRRun_1
EndMuRRun:
                RET
;End MuRRun

;############ End Motor Up ###################

;################## Punch ###################
PunchSoliA:     clr     SoA
                lcall   Delay_02s
                setb    SoA
                ret
```

```
;End PunchSoliA

PunchSoliB:     clr     SoB
                lcall   Delay_02s
                setb    SoB
                ret
;End PunchSoliB

PunchSoliC:     clr     SoC
                lcall   Delay_02s
                setb    SoC
                ret
;End PunchSoliC

PunchSoliD:     clr     SoD
                lcall   Delay_02s
                setb    SoD
                ret
;End PunchSoliD

;############### End Punch ####################


;################ Get Data (LMdown pressed) ##################
;
;-------------------------------------------------------------------------------
GetData:
                lcall   Recv
                CJNE    A, #'S', EndGetData_1
                mov     Mstart, A
                ljmp    GetData_dNLR
```

```
;********* Start Get Data **********
GetData_dNLR:
                lcall   Recv

                CJNE    A, #'N', GetData_dL
                mov     MdnIr, A
                ljmp    GetData_dX
GetData_dL:
                CJNE    A, #'L', GetData_dR
                mov     MdnIr, A
                ljmp    GetData_dX
GetData_dR:
                CJNE    A, #'R', EndGetData_1
                mov     MdnIr, A
                ljmp    GetData_dX
GetData_dX:
                lcall   Recv
                lcall   Char2Num
                mov     Mdx2, A

                lcall   Recv
                lcall   Char2Num
                mov     Mdx1, A

                lcall   Recv
                lcall   Char2Num
                mov     Mdx0, A
                ljmp    GetData_uNLR
```

```
EndGetdata_1:  ljmp    EndGetdata

GetData_uNLR:
               lcall   Recv

               CJNE    A, #'N', GetData_uL
               mov     MunIr, A
               ljmp    GetData_uX
GetData_uL:
               CJNE    A, #'L', GetData_uR
               mov     MunIr, A
               ljmp    GetData_uX
GetData_uR:
               CJNE    A, #'R', EndGetData
               mov     MunIr, A
               ljmp    GetData_uX
GetData_uX:
               lcall   Recv
               lcall   Char2Num
               mov     Mux2, A

               lcall   Recv
               lcall   Char2Num
               mov     Mux1, A

               lcall   Recv
               lcall   Char2Num
               mov     Mux0, A
               ljmp    GetData_so

GetData_so:
```

```
                lcall   Recv

                CJNE    A, #'N', GetData_soA
                mov     Msoli, A
                ljmp    GetDataOk
GetData_soA:
                CJNE    A, #'A', GetData_soB
                mov     Msoli, A
                ljmp    GetDataOk
GetData_soB:
                CJNE    A, #'B', GetData_soC
                mov     Msoli, A
                ljmp    GetDataOk
GetData_soC:
                CJNE    A, #'C', GetData_soD
                mov     Msoli, A
                ljmp    GetDataOk
GetData_soD:
                CJNE    A, #'D', EndGetData
                mov     Msoli, A
                ljmp    GetDataOk

;********* End Get Data **********
GetDataOk:
                lcall   Recv
                CJNE    A, #'E', EndGetData
                mov     Mend, A

                mov     A, #'W'
                lcall   Send
```

```
EndGetData:
                ret
;End GetData

;############## Convert Char to Number #############
;Input   :       A
;Output :       A
;################################################
Char2Num:
                CLR     C
                SUBB    A, #30h
                RET
;End Char2Num

;############### Clear Menory ###############
ClrMen:
                mov     Mvalue, #00h
                mov     Mrange, #00h

                mov     Mend, #00h
                mov     Msoli, #00h

                mov     Mux0, #00h
                mov     Mux1, #00h
                mov     Mux2, #00h
                mov     MunIr, #00h

                mov     Mdx0, #00h
                mov     Mdx1, #00h
                mov     Mdx2, #00h
```

```
                mov     Mdnlr, #00h

                mov     Mstart, #00h
                ret
;End ClrMen

;###############################################
;##############          Set LED#####################
;###############################################
SetLEDReady:
                setb    LEDWait                 ;turn off LEDWait
                nop
                clr     LEDReady                ;turn on LEDReady
                ret
;End SetLEDReady

SetLEDWait:
                setb    LEDReady                ;turn off LEDReady
                nop
                clr     LEDWait                 ;turn on LEDWait
                ret
;End SetLEDWait

SetLEDReadyBrink:
                clr     LEDReady
                lcall   Delay_02s
                setb    LEDReady
                lcall   Delay_02s
                ret
;End SetLEDReadyBrink
```

```
SetLEDWaitBrink:
        clr     LEDWait
        lcall   Delay_02s
        setb    LEDWait
        lcall   Delay_02s
EndSetLEDWaitBrink:
        setb    LEDWait
        ret
;EndSetLEDWaitBrink

SetLED2Brink:
        clr     LEDWait
        nop
        clr     LEDReady
        lcall   Delay_05s
        setb    LEDWait
        nop
        setb    LEDReady
        lcall   Delay_05s
        ret
;EndSetLED2Brink

;#########################################################
;#########     Reset          ####################
;#########################################################
Reset1:
        lcall   DCMotorStop
        lcall   DCMotorUp           ;Wait for DC Motor Up

Reset1_1:       jnb     LMleft, EndReset1
        lcall   SetLEDWait
```

```
                lcall   HighMotorLeft
                jb      LMup, Reset1
                jb      LMleft, Reset1_1
EndReset1:
                ret
;End Reset1


;#########################################################
;##########    RS-232          ####################
;#########################################################
;############ @ 11.059Mhz ###############
InitSerial:
                mov     PCON, #00000000b
                mov     SCON, #01010000b    ; Send mode 2 , Resive Enable
                mov     TMOD, #00100001b    ;Timer1 mode2, Timer0 mode1
                mov     TH1, #0FDh
                mov     TL1, #0FDh
                setb    TR1
                ret
;End InitSerial

;########## Send Data ###################
; Input  :      A               #
;---------------------------------------------------------
Send:
                clr     TI
                mov     SBUF, A
                jnb     TI, $
                clr     TI
                ret
;End Send
```

```
;########## Resive Data #################
; Output:       A               #
;-------------------------------------------------------------
Recv:
              clr     RI
              jnb     RI, $
              mov     A, SBUF
              clr     RI
              ret
;End Recv

;######## Send String ####################
; Input  :     mov     DPTR, #label data (Data stop with #00h)
;-------------------------------------------------------------
SendString:
              mov     A, #00h
              movc    A, @A+DPTR
              jz      EndSendString
              lcall   Send
              inc     DPTR
              ljmp    SendString
EndSendString:
              ret
;End SendString

;########################################
;########              Motor   ###############
;########################################

;############ Motor High #################
```

```
HighMotorRight:jnb      LMright, EndHighMotorRight
               mov      p2, #00110000b
               lcall    DelayHightMortor
               mov      p2, #01100000b
               lcall    DelayHightMortor
               mov      p2, #11000000b
               lcall    DelayHightMortor
               mov      p2, #10010000b
               lcall    DelayHightMortor
               mov      p2, #00000000b
EndHighMotorRight:
               mov      p2, #00000000b
               ret
;End HighMotorRight

HighMotorLeft: jnb      LMleft, EndHighMotorLeft
               mov      p2, #11000000b
               lcall    DelayHightMortor
               mov      p2, #01100000b
               lcall    DelayHightMortor
               mov      p2, #00110000b
               lcall    DelayHightMortor
               mov      p2, #10010000b
               lcall    DelayHightMortor
               mov      p2, #00000000b
EndHighMotorLeft:
               mov      p2, #00000000b
               ret
;End HighMotorLeft
;-------------------------------------------------------------------
DelayHightMortor:
```

```
                LCALL  Delay_0002s
                RET
;EndDelayHightMortor
;-------------------------------------------------------------------

;############## Motor Low ###############
LowMotorRight: mov     p2, #00000011b
               lcall   Delay_0005s
               mov     p2, #00000110b
               lcall   Delay_0005s
               mov     p2, #00001100b
               lcall   Delay_0005s
               mov     p2, #00001001b
               lcall   Delay_0005s
               mov     p2, #00000000b
               ret
;End LowMotorRight

LowMotorLeft:  mov     p2, #00001100b
               lcall   Delay_0005s
               mov     p2, #00000110b
               lcall   Delay_0005s
               mov     p2, #00000011b
               lcall   Delay_0005s
               mov     p2, #00001001b
               lcall   Delay_0005s
               mov     p2, #00000000b
               ret
;End LowMotorLeft

;############## DC Motor ###############
```

```
DCMotorUp:      jnb     LMup, EndDCMotorUp  ;Jump if LMup pressed
                setb    DCMotorA
                nop
                clr     DCMotorB
                lcall   SetLEDWaitBrink
                jnb     SWdown, EndDCMotorUp        ; มีการกดหมุนลงขณะที่หมุนขึ้น
                nop
                jb      LMup, DCMotorUp             ;Jump if LMup not pressed (Loop)
EndDCMotorUp:
                lcall   DCMotorStop
                ret
;End DCMotorUp

DCMotorDown:
WaitLMfb1:      jnb     LMf, WaitLMfb1_1            ;Jump if LMf pressed
                lcall   SetLEDWait
                lcall   DCMotorStop
                jb      LMf, WaitLMfb1              ;--> Wait plastic front

WaitLMfb1_1:    jnb     LMb, WaitSWDown1            ;Jump if LMb pressed
                lcall   SetLEDWait
                lcall   DCMotorStop
                jb      LMb, WaitLMfb1              ;--> Wait plastic back

;------ plastic Placed ------
WaitSWDown1:    nop
                jnb     SWup, EndDCMotorDown        ; มีการกดหมุนขึ้นขณะที่หมุนลง
                lcall   SetLEDReady
                clr     DCMotorA
                nop
                setb    DCMotorB
```

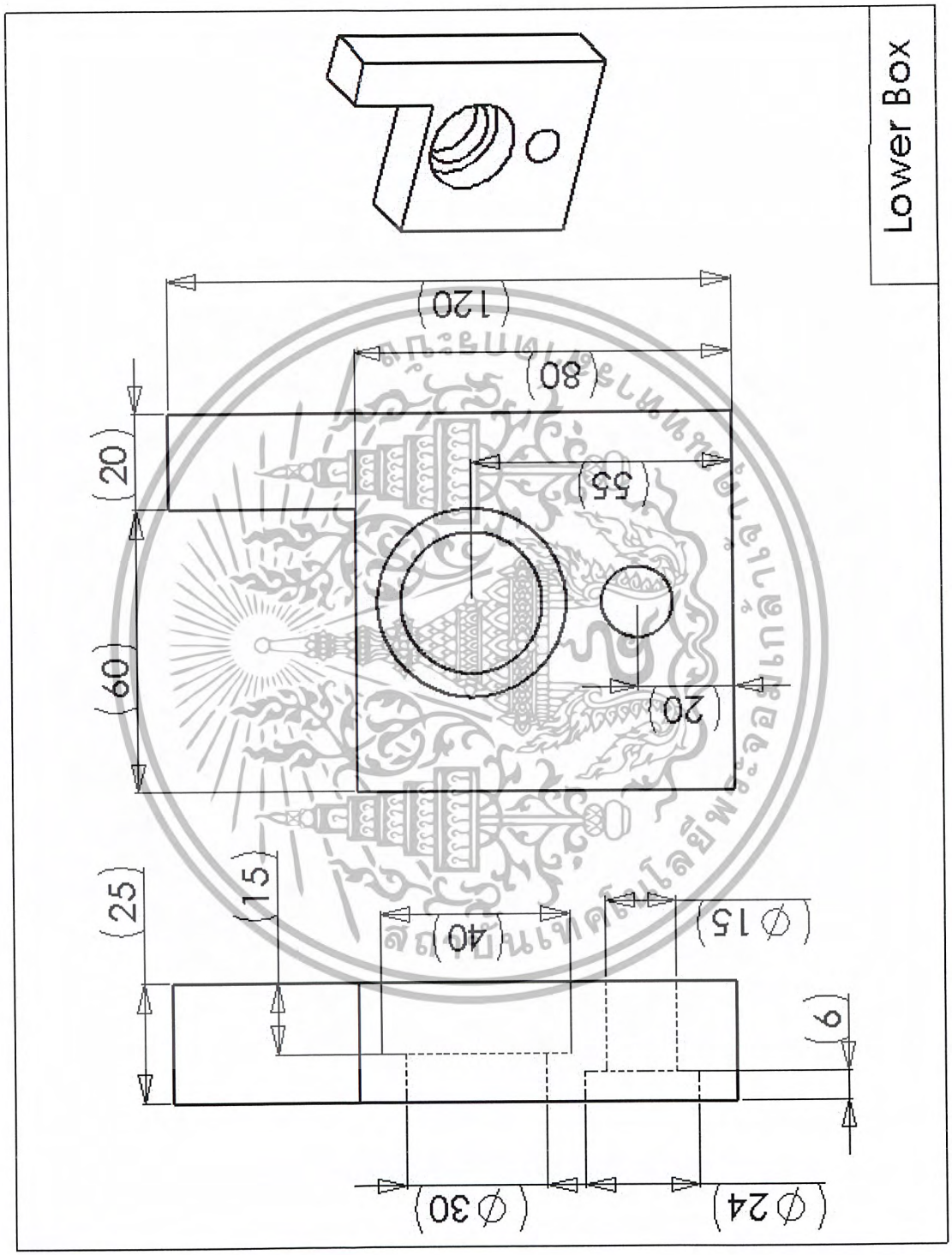
```
                jb      LMdown, DCMotorDown;Jump if LMdown not pressed (Loop)
EndDCMotorDown:
                lcall   DCMotorStop
                ret
;End DCMotorDown1

DCMotorStop:    setb    DCMotorA
                nop
                setb    DCMotorB
                ret
;End DCMotorStop

;#########################################
;###############     Delay     ###################
;#########################################

;######## Delay 1 mSec  @ 11.059Mhz ##########
Delay_1ms:      MOV     TMOD, #00100001b    ;Timer1 mode2, Timer0 mode1
                MOV     TH0, #0FCh
                MOV     TL0, #66h
                SETB    TR0                 ;Start Timer0
                JNB     TF0, $              ;Wait for over flow TF0 is in Timer0
                CLR     TR0                 ;Stop Timer0
                CLR     TF0                 ;Clear Flag Over flow in TCON Timer0
                RET
;End Delay_1ms

;############### Delay 1 sec ##############
Delay_1s:
                MOV     R7, #10d
Delay_1s_1:     MOV     R6, #100d
```

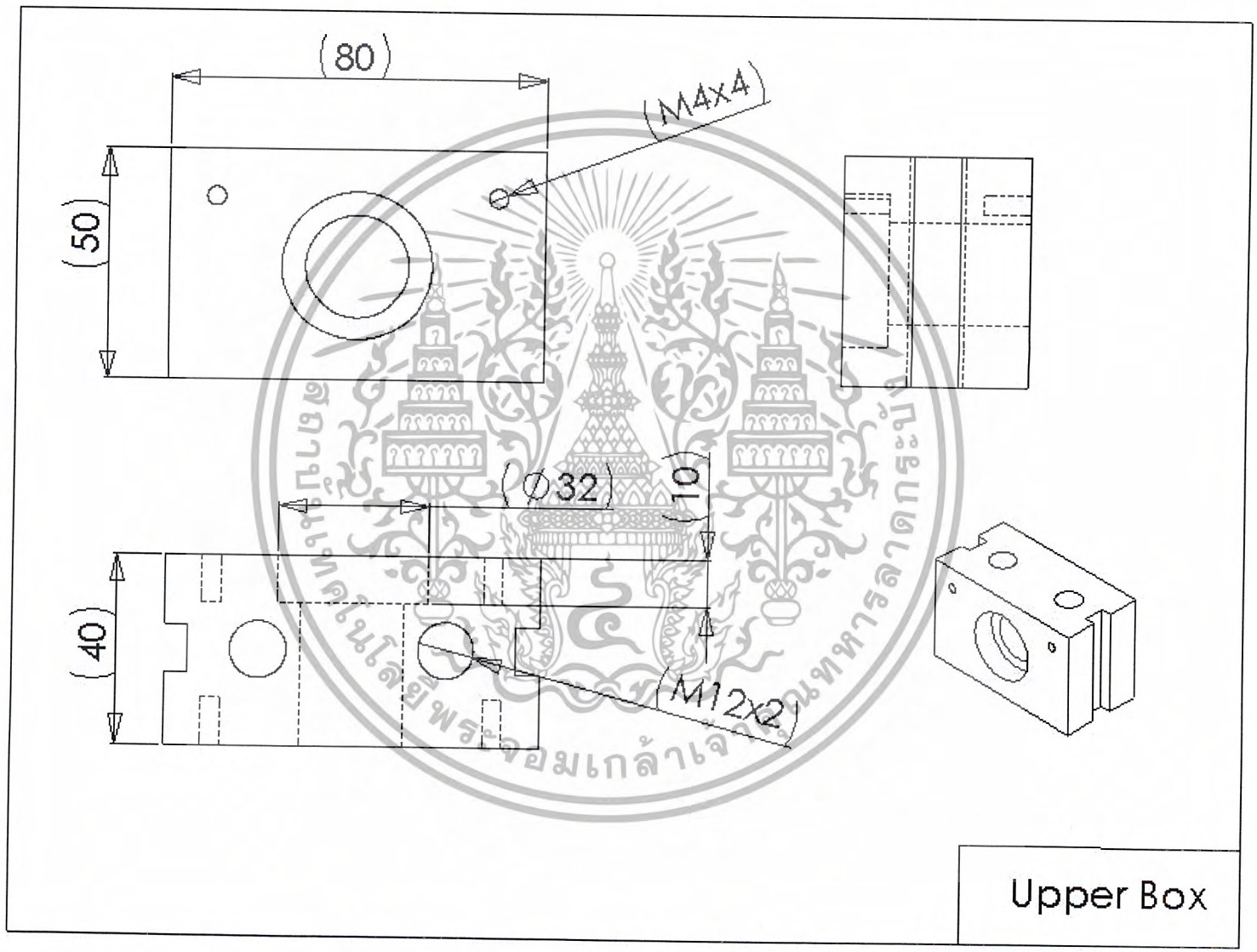
```
Delay_1s_2:     LCALL  Delay_1ms
                DJNZ   R6, Delay_1s_2
                DJNZ   R7, Delay_1s_1
                RET
;End Delay_1s

;############### Delay 0.5 sec ###############
Delay_05s:
                MOV    R7, #05d
Delay_05s_1:    MOV    R6, #100d
Delay_05s_2:    LCALL  Delay_1ms
                DJNZ   R6, Delay_05s_2
                DJNZ   R7, Delay_05s_1
                RET
;End Delay_05s

;############### Delay 0.2 sec ###############
Delay_02s:
                MOV    R7, #02d
Delay_02s_1:    MOV    R6, #100d
Delay_02s_2:    LCALL  Delay_1ms
                DJNZ   R6, Delay_02s_2
                DJNZ   R7, Delay_02s_1
                RET
;End Delay_02s

;############### Delay 0.1 sec ###############
Delay_01s:
                MOV    R7, #01d
Delay_01s_1:    MOV    R6, #100d
Delay_01s_2:    LCALL  Delay_1ms
```

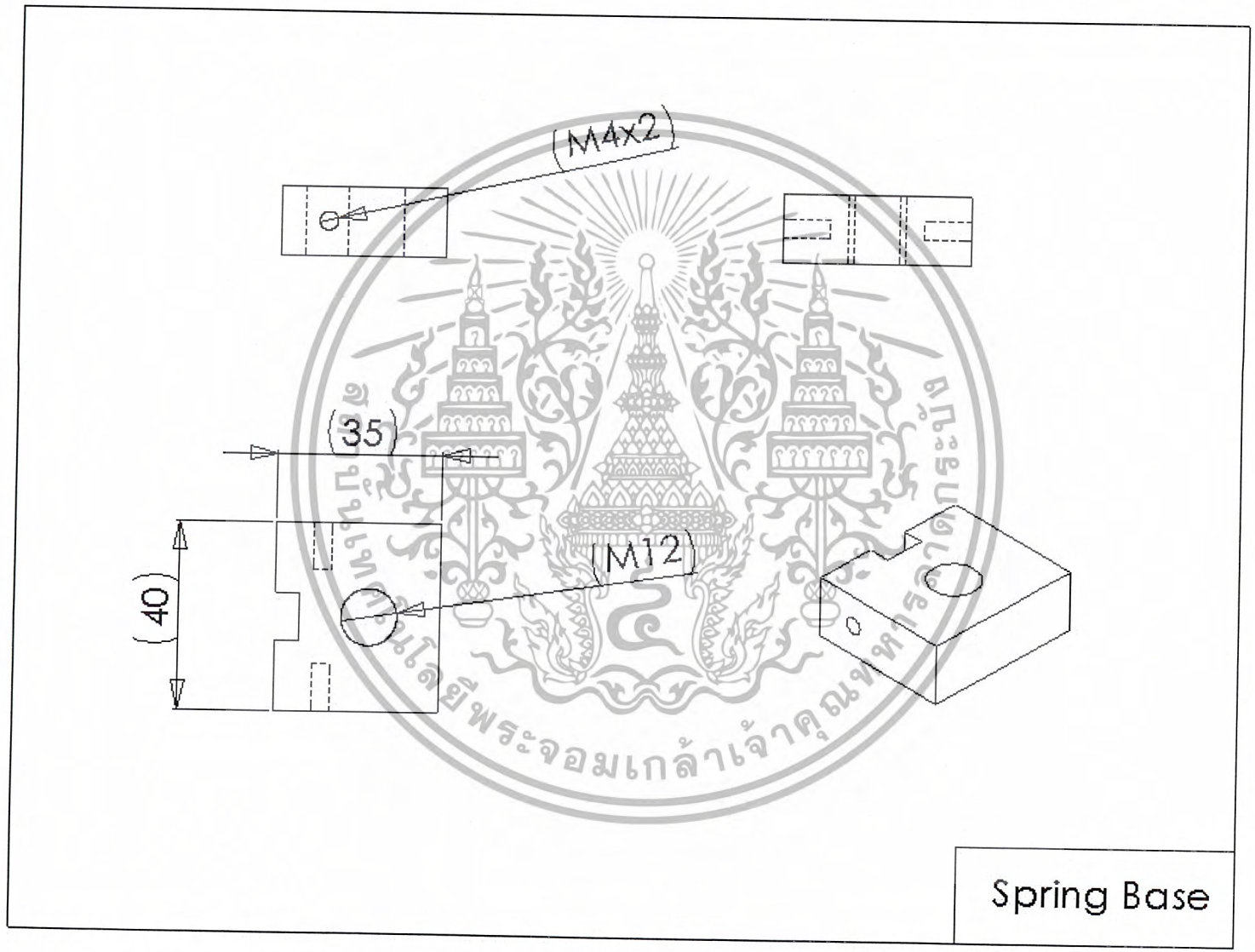
```
                DJNZ    R6, Delay_01s_2
                DJNZ    R7, Delay_01s_1
                RET
;End Delay_01s

;############### Delay 0.02 sec 50Hz###############
Delay_002s:
                MOV     R7, #02d
Delay_002s_1:   MOV     R6, #10d
Delay_002s_2:   LCALL   Delay_1ms
                DJNZ    R6, Delay_002s_2
                DJNZ    R7, Delay_002s_1
                RET
;End Delay_002s

;############### Delay 0.005 sec 200 Hz ###############
Delay_0005s:
                MOV     R7, #05d
Delay_0005s_1:  MOV     R6, #01d
Delay_0005s_2:  LCALL   Delay_1ms
                DJNZ    R6, Delay_0005s_2
                DJNZ    R7, Delay_0005s_1
                RET
;End Delay_0005s

;############### Delay 0.002 sec 500 Hz###############
Delay_0002s:
                MOV     R7, #05d
Delay_0002s_1:  MOV     R6, #01d
Delay_0002s_2:  LCALL   Delay_1ms
                DJNZ    R6, Delay_0002s_2
```

```
                DJNZ    R7, Delay_0002s_1
                RET
;End Delay_0002s
;################################################
;#############          Define Data      ###########
;################################################
Data_1:         DB      "a", 00h
Data_2:         DB      "bb", 00h
                end
```

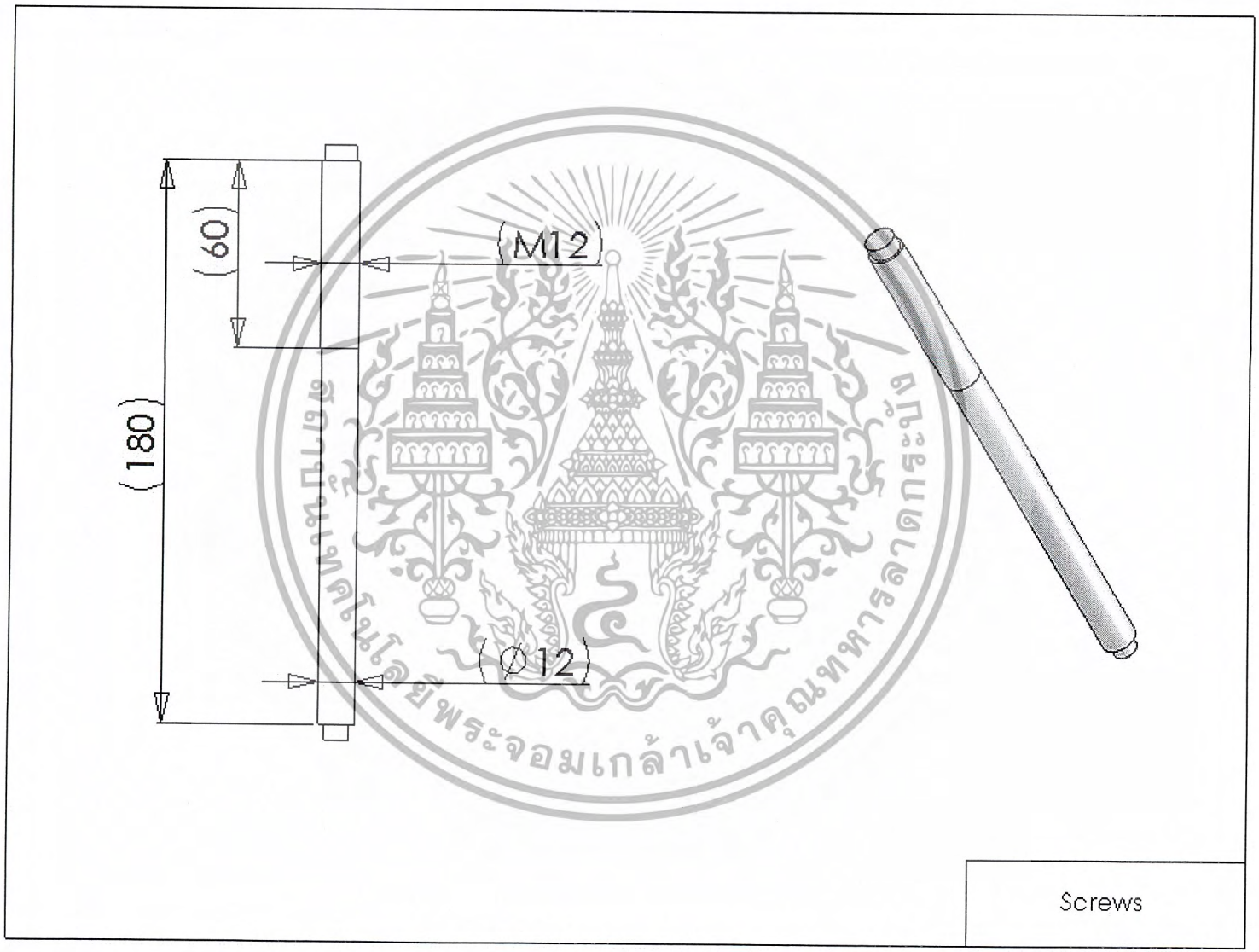
ภาคผนวก ค
แบบชิ้นส่วนของเครื่องเจาะแผ่นพลาสติก

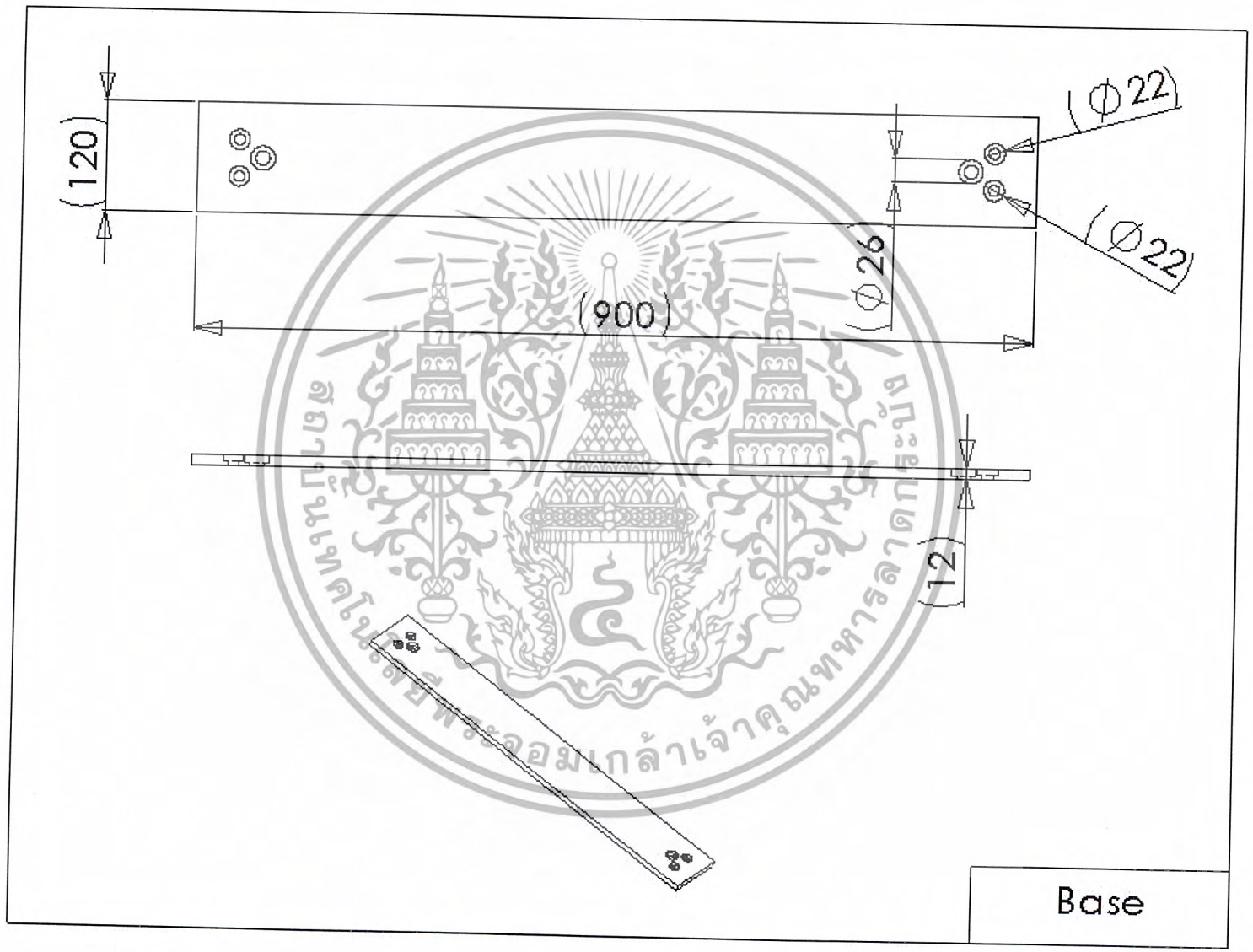

Lower Box
120
80
20
55
60
20
25
15
40
Ø15
9
Ø30
Ø24

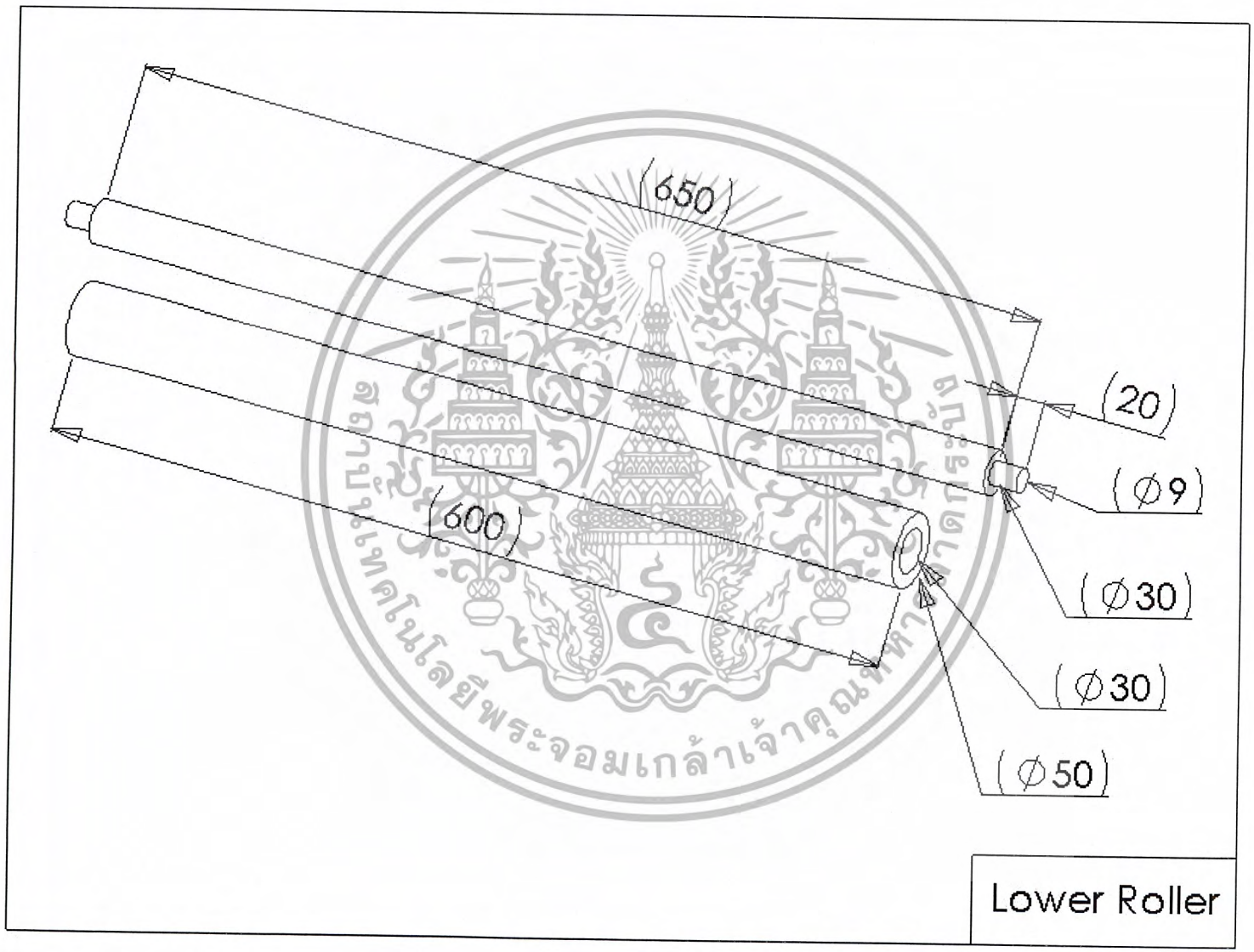

80
M4x4
50
Ø32
10
40
M12x2
Upper Box

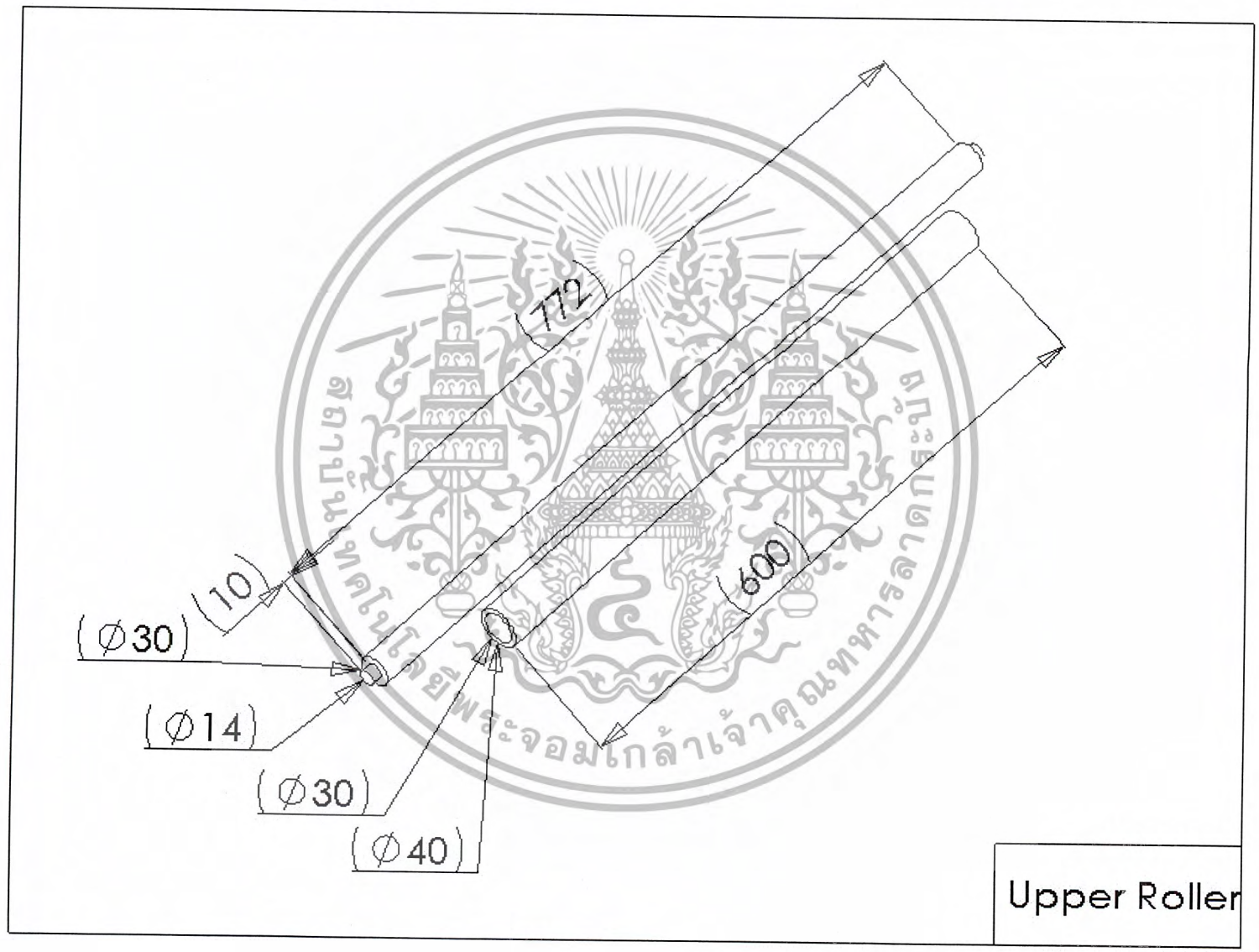


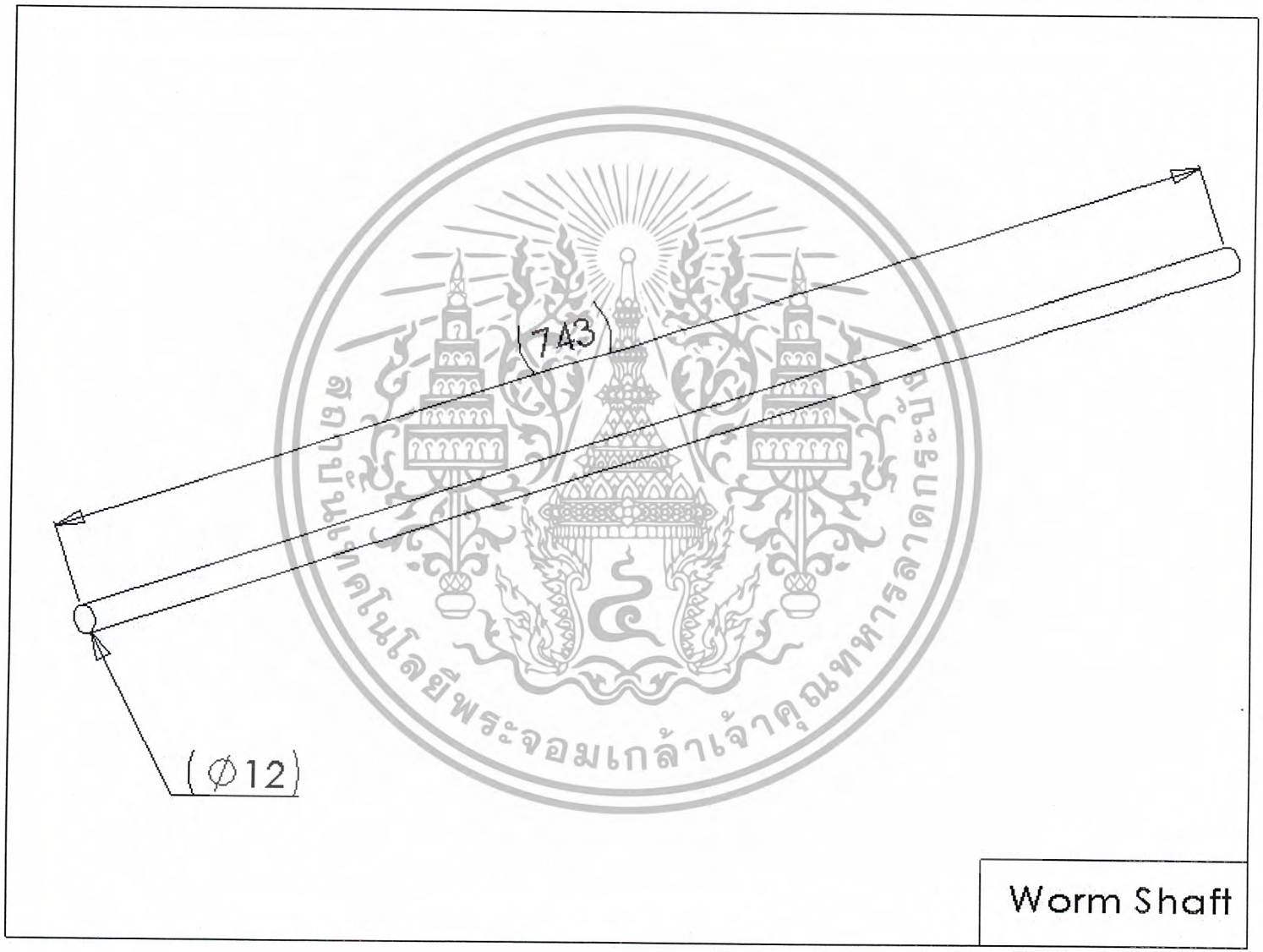
Spring Base

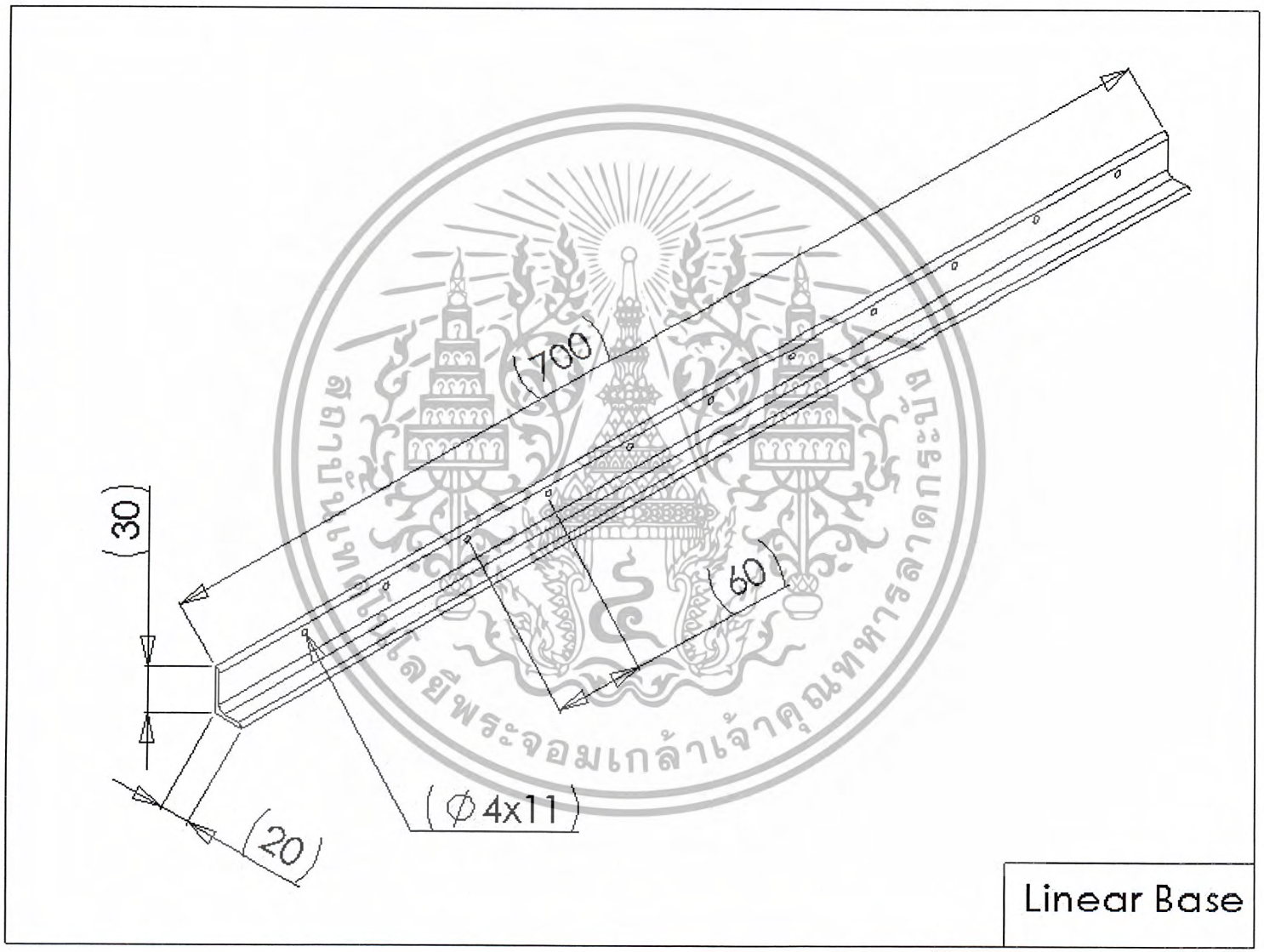

(M12)
(60)
(180)
(∅12)
Screws

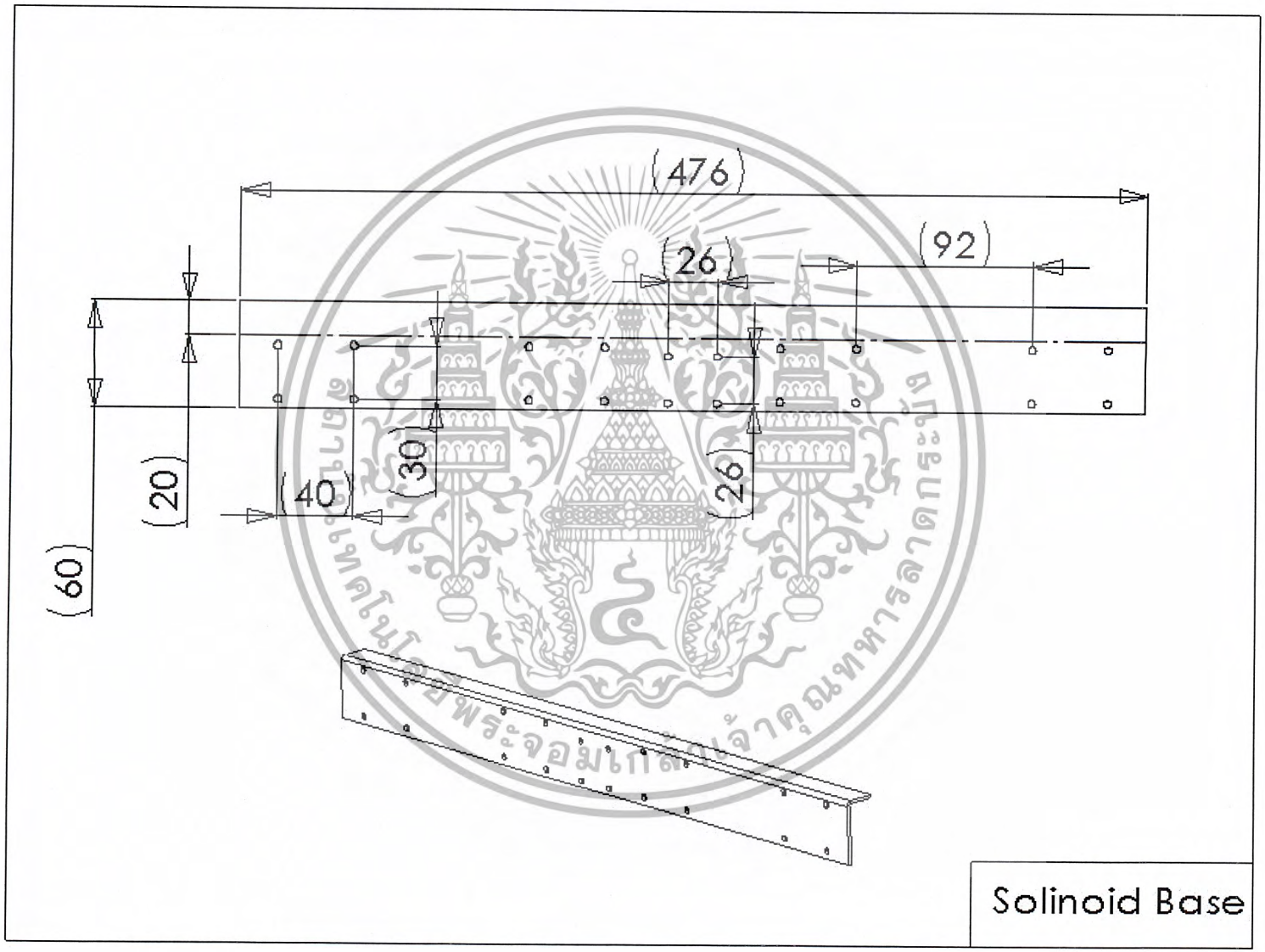

120
900
Ø22
Ø22
Ø26
12
Base

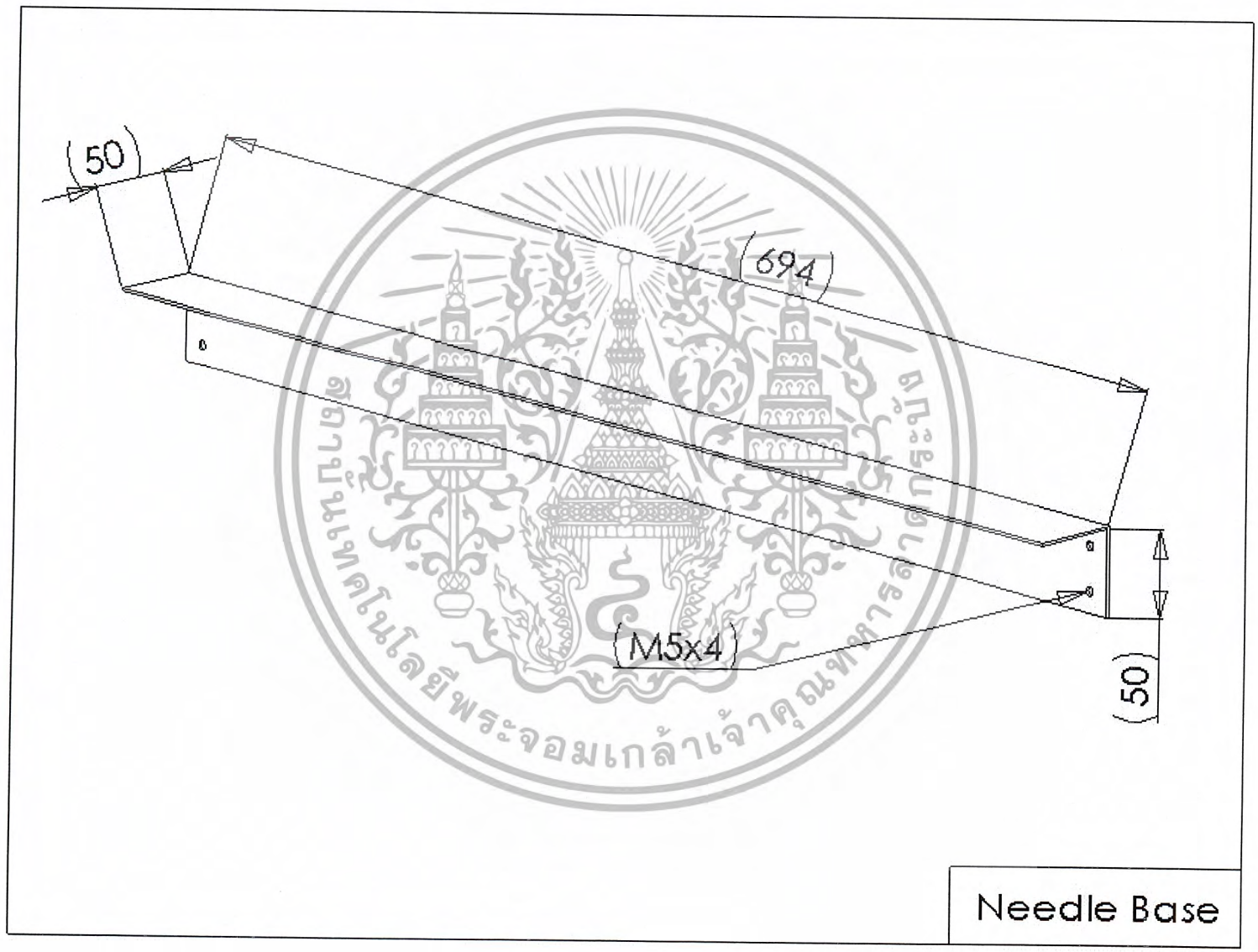

650
20
Ø9
600
Ø30
Ø30
Ø50
Lower Roller

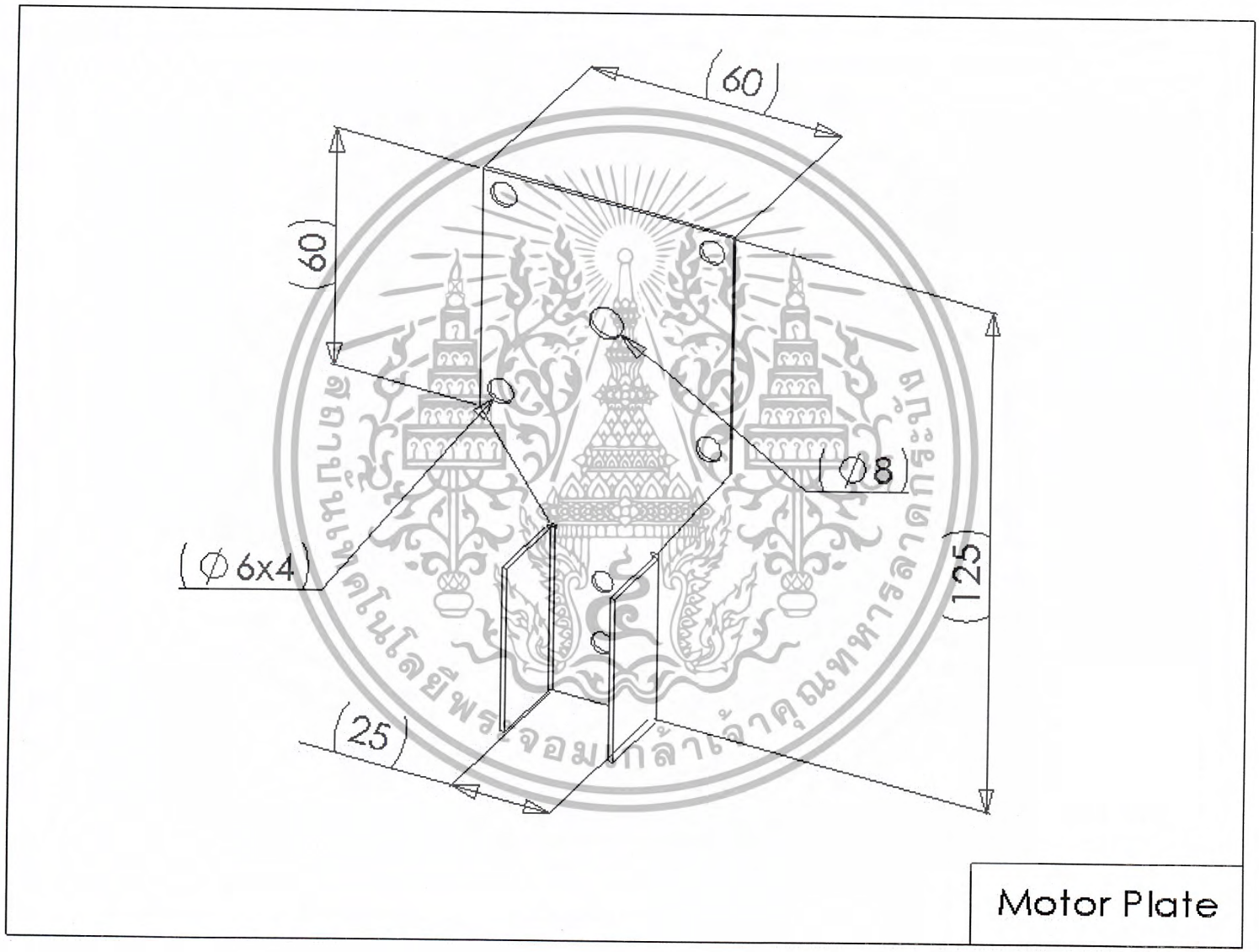

172
600
10
Ø30
Ø14
Ø30
Ø40
Upper Roller

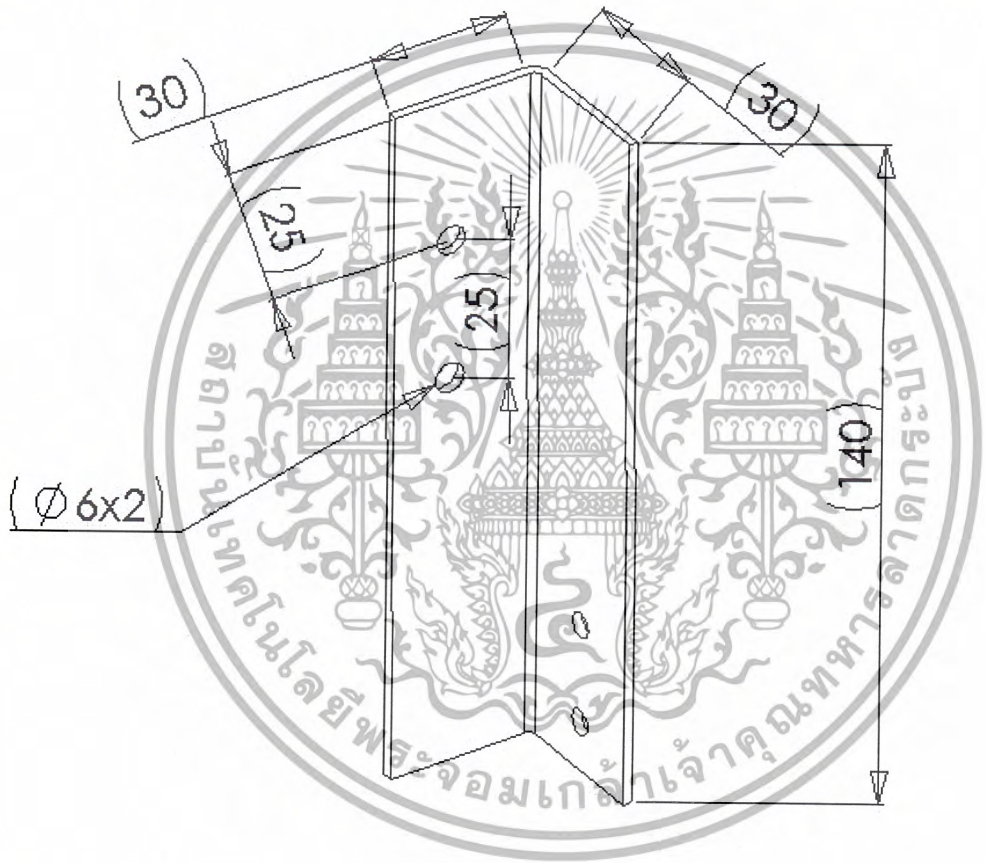

743
Ø12
Worm Shaft


700
30
60
Ø4x11
20
Linear Base

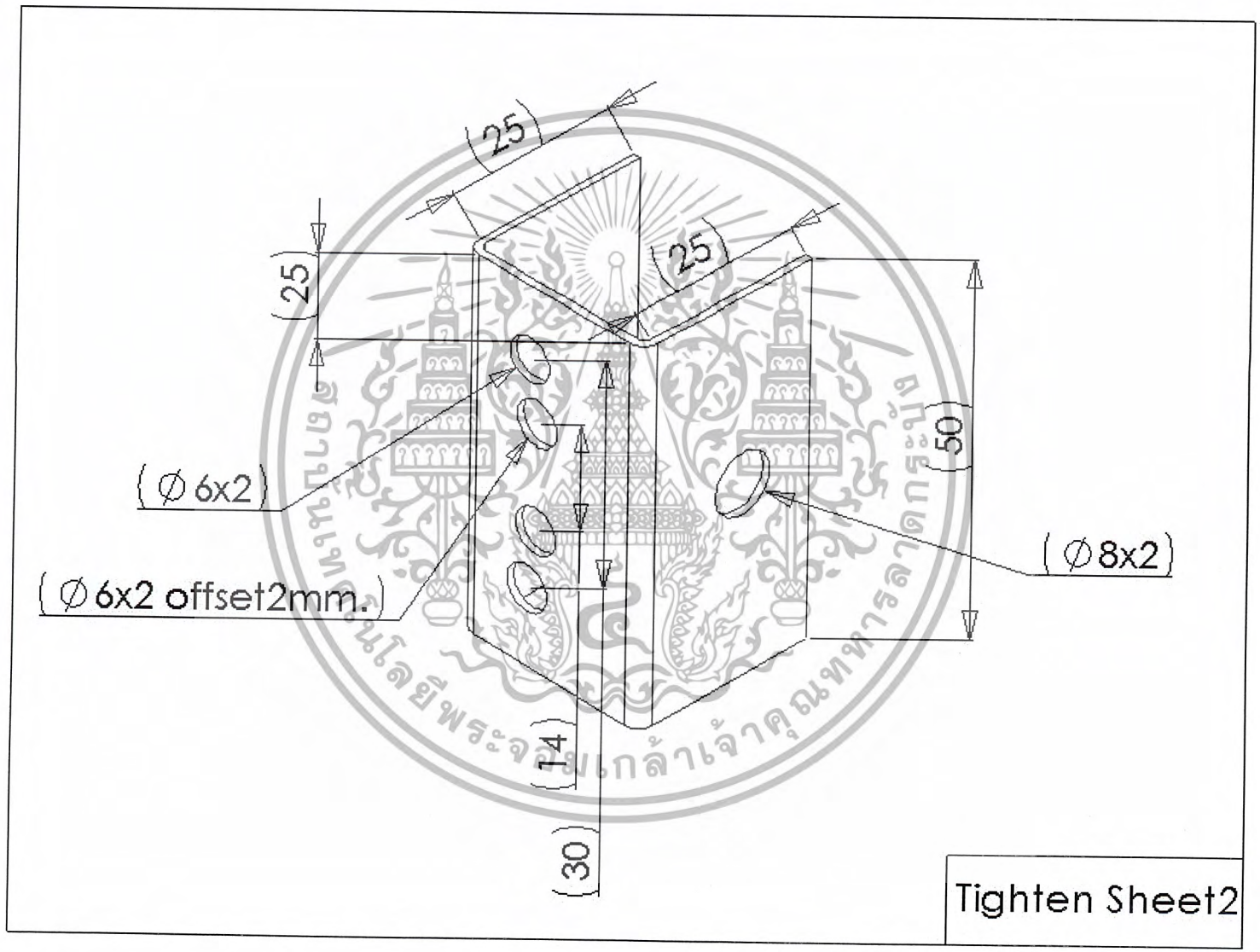

476
92
26
20
30
40
26
60
Solinoid Base

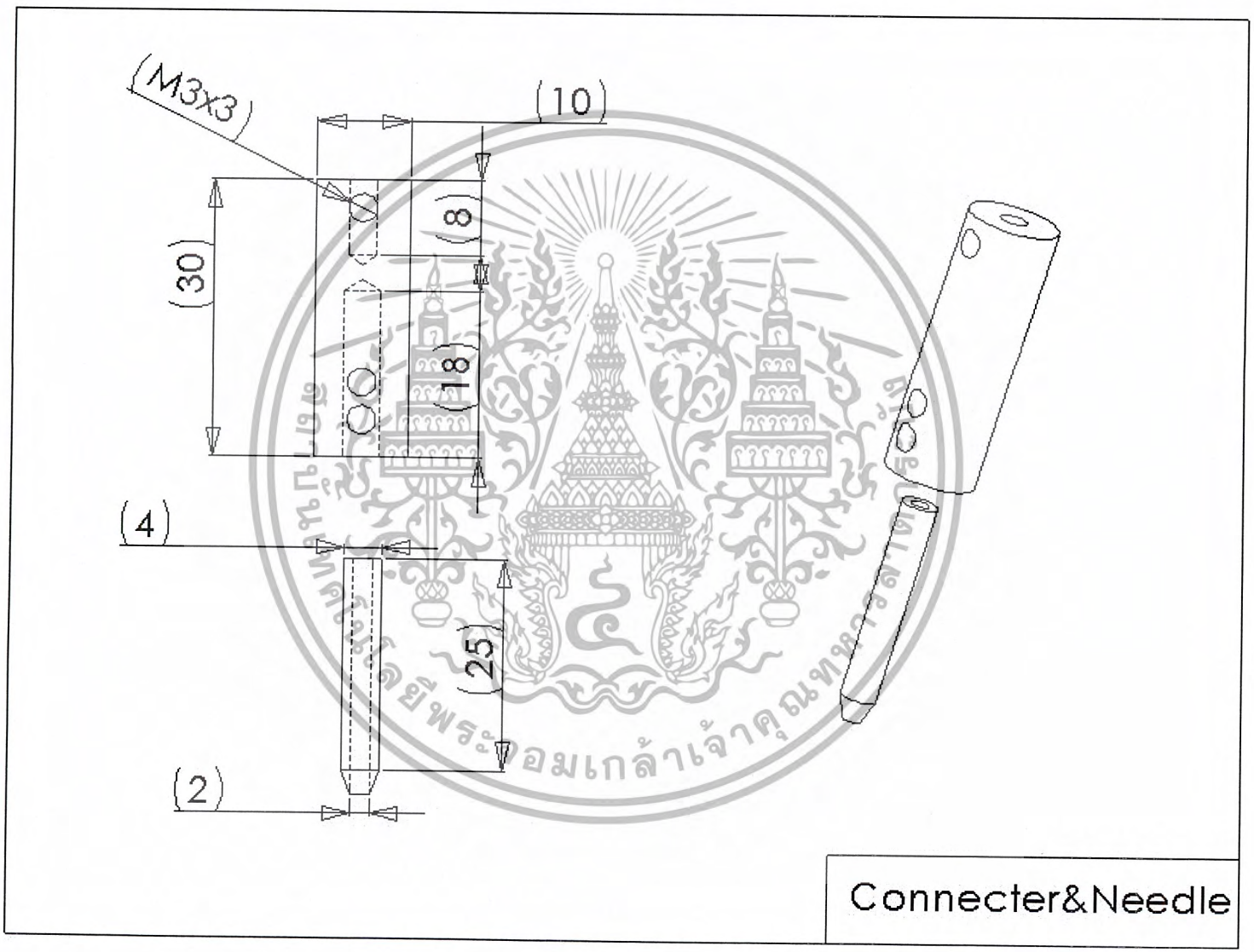

50
694
M5x4
50
Needle Base


60
60
∅8
∅6x4
125
25
Motor Plate



Tighten Sheet1


25
25
25
50
(∅6x2)
(∅6x2 offset2mm.)
(∅8x2)
14
(30)
Tighten Sheet2



Connecter&Needle