การพัฒนาเครื่องเจาะแผ่นพลาสติกเพื่อทำแบบพรม

A DEVELOPMENT OF PLASTIC SHEET PUNCHING MACHINE FOR CARPET PATTERN

นายวรพงษ์ วงษ์สมศรี
นายวุฒิชัย ฉาหมัน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

การพัฒนาเครื่องเจาะแผ่นพลาสติกเพื่อทำแบบพรม

A DEVELOPMENT OF PLASTIC SHEET PUNCHING MACHINE FOR CARPET PATTERN

โดย

นายวรพงษ์ วงษ์สมศรี

นายวุฒิชัย ลาหมัน

อาจารย์ที่ปรึกษา

ดร. ณัฐวุฒิ เดไปวา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชา วิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเครื่องเจาะแผ่นพลาสติกเพื่อทำแบบพรม

A DEVELOPMENT OF PLASTIC SHEET PUNCHING MACHINE FOR CARPET PATTERN

ผู้จัดทำ

1. นายวรพงษ์ วงษ์สมศรี รหัสประจำตัว 47015371

2. นายวุฒิชัย ลาหมัน รหัสประจำตัว 47015409

ณัฐวุฒิ เดไปวา อาจารย์ที่ปรึกษา

(ดร. ณัฐวุฒิ เดไปวา)

การพัฒนาเครื่องเจาะแผ่นพลาสติกเพื่อทำแบบพรม

นายวรพงษ์ วงษ์สมศรี รหัสนักศึกษา 47015371
นายวุฒิชัย ลาหมัน รหัสนักศึกษา 47015409
ดร. ณัฐวุฒิ เดไปวา อาจารย์ที่ปรึกษา

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาการออกแบบและสร้างเครื่องเจาะรูแผ่นพลาสติกเพื่อนำไปใช้เป็นแบบในการทอพรมโดยอาศัยหลักการเคลื่อนที่ 2 แกนในเครื่องพริ้นเตอร์(Printer)มาเป็นพื้นฐานในการออกแบบส่วนประกอบสำคัญของเครื่องเจาะรูแผ่นพลาสติก 1.ระบบการเคลื่อนที่แบบ 2 แกน ที่ใช้สเต็ปมอเตอร์(Step Motor) 2 ตัวในการสร้างการเคลื่อนที่แบบ 2 แกน โดยตัวแรกใช้ในการควบคุมตำแหน่งของเข็มเจาะ ซึ่งใช้สายพานเป็นตัวส่งถ่ายกำลัง สเต็ปมอเตอร์ตัวที่สองใช้ในการขับลูกกลิ้งเพื่อดึงแผ่นพลาสติกเข้าไป ส่งกำลังโดยใช้ชุดเฟือง 2.ชุดไมโครคอนโทรลเลอร์ใช้ในการควบคุมชุดสเต็ปมอเตอร์ และตัวโซลีนอยด์(Solenoid) 3.โปรแกรมควบคุมการทำงานของเครื่อง จะรับสัญญาณพัลส์จากคอมพิวเตอร์มาใช้ในการสั่งให้สเต็ปมอเตอร์ทำงาน

A DEVELOPMENT OF PLASTIC SHEET PUNCHING MACHINE FOR CARPET PATTERN

Mr.Worapong Wongsomsri Student ID 47015371
Mr.Wuttichai Lamun Student ID 47015409
Dr.Nattawoot Depaiwa Advisor

ABSTRACT

This project is to develop the design and the construction of the plastic sheet used as a model in carpet weaving process punching machine base on the theory of 2 axis motion in printer. The main compositions of the plastic sheet punching machine are following, 1.Two step motors which used to create the 2-axis motion. One is used to control the position of the needles by using belt to transfer the power. The other is used to control the rollers which pull in the plastic sheet by using gears to transfer the power.2. Micro-controller which used to control the step motors and the solenoids.3.The program to control the receiver which receive from computer the pulse signal used to run the step motors.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จลุล่วงไปได้ด้วยดี ถ้าหากปราศจากคำแนะนำ และความอุปถัมภ์ทางด้านการเงินจากอาจารย์ ดร.ณัฐวุฒิ เดไปวา อาจารย์ที่ปรึกษาปริญญานิพนธ์

ขอขอบคุณ รุ่นพี่ และเพื่อนทุกคนที่ช่วยเหลือ และคอยแก้ไขปัญหาในเรื่องต่างๆทำให้งานสำเร็จลงได้

ท้ายที่สุด คณะผู้จัดทำขอกราบขอบพระคุณบิดา มารดา พี่น้อง รวมถึงครอบครัวของคณะผู้จัดทำที่ให้การสนับสนุนทางด้านการศึกษา กำลังใจ และแนะแนวทางในการดำเนินชีวิต ด้วยความรักและความปรารถนาดีตลอดมา

นายวรพงษ์ วงษ์สมศรี

นายวุฒิชัย ลาหมัน

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

หน้าที่

สารบัญตาราง

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในอุตสาหกรรมทอพรมประกอบด้วยกระบวนการผลิตหลายขั้นตอน เริ่มตั้งแต่การสร้างแบบเพื่อใช้ในการทอพรม โดยแบบที่สร้างขึ้นจะมีลักษณะเป็นแผ่นพลาสติกที่มีการเจาะรูเล็ก ๆ เดินเป็นรูปลายเส้น เมื่อได้แผ่นพลาสติกที่ถูกเจาะตามลายเส้นที่ต้องการจะทอลงบนพรมแล้ว ก็จะทำแผ่นพลาสติกเจาะรูที่ได้ไปทาบลงบนผ้าสำหรับทอพรม แล้วจึงทาสีบนแผ่นพลาสติกเพื่อให้เกิดลวดลายต่าง ๆ ตามลายเส้นที่ได้เจาะรู เมื่อนำแผ่นพลาสติกออก ก็จะได้รูแบบแนวเส้นที่จะใช้ในการทอพรมต่อไป ซึ่งขั้นตอนการผลิตที่สำคัญ และเป็นขั้นตอนที่เป็นเหตุให้เกิดการออกแบบและสร้างเครื่องเจาะแผ่นพลาสติกสำหรับทำแบบทอพรม (Plastic Sheet Punching Machine) ก็คือ การเจาะแผ่นพลาสติกเพื่อนำไปทาบลงบนผ้าสำหรับใช้ทอพรม ซึ่งในขั้นตอนนี้ โรงงานอุตสาหกรรมทั่วไป ยังไม่มีเครื่องมือเฉพาะทางที่ใช้ในการเจาะรูตามแบบลายเส้น โดยทั่วไปจะเป็นการใช้แรงงานคนในการตอกเพื่อเจาะรูตามลายเส้น เพื่อให้เกิดการพัฒนากระบวนการผลิตในโรงงานอุตสาหกรรมทอพรม ประกอบกับเพื่อให้ได้เครื่องจักรที่มีราคาถูก สามารถนำไปใช้งานได้จริง และยังนำไปปรับใช้กับอุตสาหกรรมชนิดอื่นได้ จึงได้มีการสร้างเครื่องเจาะแผ่นพลาสติกสำหรับทำแบบทอพรม โดยมีวัตถุประสงค์หลัก เพื่อต้องการลดขั้นตอนการผลิตให้ได้อัตราการผลิตที่มีประมาณมากขึ้น และยังสามารถจัดสรรทรัพยากรบุคคลไปทำงานในส่วนอื่นเพื่อเป็นการเพิ่มประสิทธิภาพในกระบวนการผลิตได้อีกด้วย

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

เครื่องเจาะแผ่นพลาสติกสำหรับสร้างแบบทอพรม เป็นการปฏิวัติทางด้านเทคโนโลยีในกระบวนการผลิตพรมจากแรงงานคนสู่เครื่องจักร จากขั้นตอนการผลิตหลาย ๆ ขั้นตอนก่อนทำการเจาะพลาสติก เหลือเพียงไม่กี่ขั้นตอนในการเจาะพลาสติก อย่างไรก็ตามโครงงานการออกแบบและสร้างเครื่องเจาะพลาสติกสำหรับทำแบบทอพรมก็ยังจะต้องมีการพัฒนาต่อไป ตามเทคโนโลยีใหม่ ๆ ที่เกิดขึ้นอย่างต่อเนื่องในโลกอุตสาหกรรม เพื่อให้เกิดพัฒนาในวงกว้างของระบบอุตสาหกรรมไทย

1.2 วัตถุประสงค์

1. พัฒนาระบบการทำงานของเครื่องเจาะให้มีความทนทานและใช้งานได้อย่างต่อเนื่อง
2. พัฒนาประสิทธิภาพการเจาะให้มีความผิดพลาดน้อยลง
3. แก้ไขและพัฒนาโปรแกรมให้สามารถใช้งานได้สะดวกตามความต้องการของผู้ใช้

1.3 ขอบเขตของโครงงาน

1. แก้ไขระบบการทำงานของเครื่องเจาะแผ่นพลาสติกให้มีประสิทธิภาพมากขึ้น

2.พัฒนาหน้าต่างโปรแกรมการทำงานของเครื่องเจาะแผ่นพลาสติก เพื่อให้มีความสะดวกในการใช้งานมากขึ้น

3. แก้ไขชุดวงจรควบคุมการทำงานของเครื่องเจาะแผ่นพลาสติก ให้มีเสถียรภาพมากขึ้น

1.4 ขั้นตอนการดำเนินโครงงาน

1. ศึกษาหลักการทำงานของระบบการเคลื่อนที่แบบ 2 แกน
2. ออกแบบและสร้างเครื่องเจาะพลาสติกในส่วนที่ต้องมีการพัฒนา
3. ศึกษาและออกแบบโปรแกรมวิชวลเบสิกที่ใช้ในการกำหนดขนาดภาพ
4. ติดตั้งและทำการทดลอง
5. สรุปผลการทดลองและข้อผิดพลาดที่เกิดจากการทดลอง
6. 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 ใช้สายพานแบบไทม์มิ่งเพื่อให้เกิดความไวในการตอบสนองต่อแรงขับเคลื่อนยที่ส่งมาจากสเต็ปเปอร์มอเตอร์ และป้องกันการลื่นไถล นอกจากนี้ การส่งกำลังโดยใช้สายพานยังมีข้อดี คือ

1. สามารถดูดซับการกระแทก และการสั่นสะเทือนได้ดี

2. ติดตั้งง่ายไม่ต้องการการหล่อลื่น

3. เสียงค่อนข้างเบา

สมการหาความยาวของสายพาน

$$\theta_d = \pi - 2\sin^{-1}\frac{D-d}{2c} \qquad (2.1)$$

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

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

โดยที่ θ_D คือ มุมของหน้าสัมผัสที่มูเลย์ตัวใหญ่

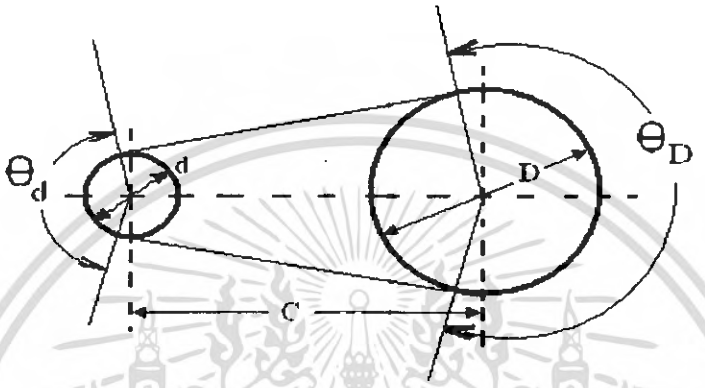
θ_d คือ มุมของหน้าสัมผัสที่มูเลย์ตัวเล็ก

c คือ ระยะห่างระหว่างมูเลย์ทั้งสอง

D คือ เส้นผ่านศูนย์กลางของมูเลย์ตัวใหญ่

d คือ เส้นผ่านศูนย์กลางของมูเลย์ตัวเล็ก

และ L คือ ความยาวของสายพาน



รูปที่ 2-2 แสดงถึงการติดตั้งสายพานแบบเปิดสำหรับขับเพลาที่ขนานกันให้หมุนไปในทิศเดียวกัน

2.2 เฟือง

การคำนวณหาความเร็วรอบของระบบเฟืองจะคำนวณได้จากการหาค่าการขับกันของเฟือง (Train Value)

$$e = \frac{product\ of\ driving\ tooth\ numbers}{product\ of\ driven\ tooth\ numbers} \quad (2.4)$$

$$n_L = en_F \quad (2.5)$$

$$P = \frac{N}{d} \quad (2.6)$$

$$m = \frac{d}{N} \quad (2.7)$$

$$p = \frac{\pi\ d}{N} = \pi\ m \quad (2.8)$$

$$pP = \pi \quad (2.9)$$

โดยที่ e คือ ค่าการขับกันของเฟือง (Train Value)

n_L คือ ความเร็วรอบของเฟืองตัวสุดท้ายในระบบการขับ , รอบต่อนาที

n_F คือ ความเร็วรอบของเฟืองตัวแรกในระบบการขับ , รอบต่อนาที

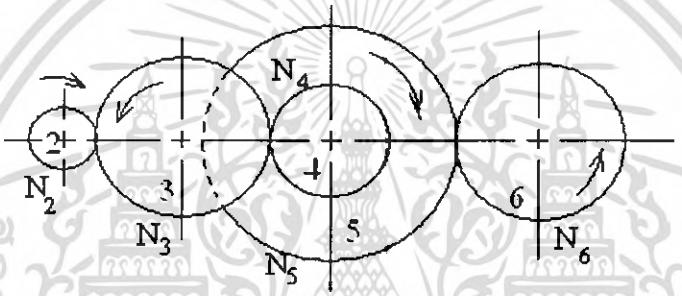
P คือ อัตราส่วนจำนวนฟันของเฟืองต่อระยะพิทช์ (Diametral pitch) , จำนวนฟันต่อนิ้ว

N คือ จำนวนฟัน

d คือ ระยะพิทช์ (Pitch diameter) , นิ้ว

m คือ โมดูลของเฟือง (Module) , มิลลิเมตร

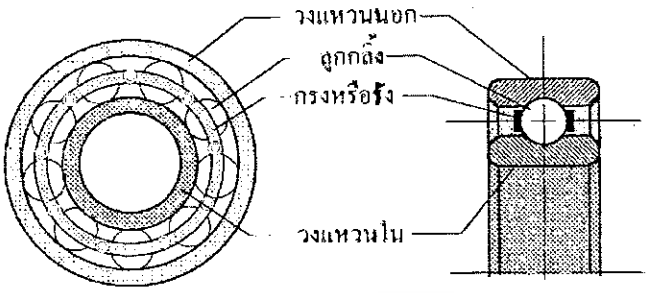
p คือ ระยะบนวงกลมพิทช์จากจุดบนฟันหนึ่งไปยังฟันเฟืองใกล้เคียง (Circular pitch) , มิลลิเมตร



รูปที่ 2-3 แสดงถึงการขับกันของระบบเฟือง

2.3 แบริ่ง

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



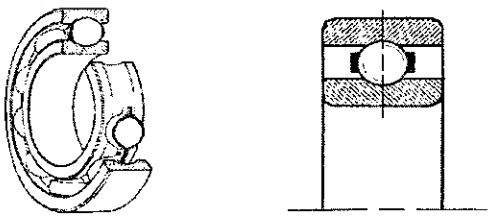
รูปที่ 2-4 แสดงถึงส่วนประกอบภายในของแบริ่งแบบกลิ้งหรือตลับลูกปืน

คุณสมบัติทั่ว ๆ ไปของแบริ่งแบบกลิ้ง ได้แก่

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

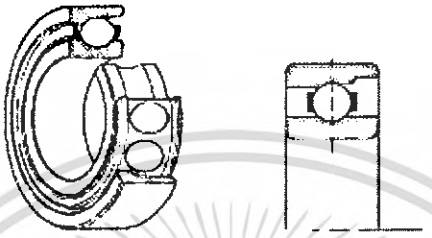
2.3.1 ประเภทแบริ่งแบบเม็ดกลม

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



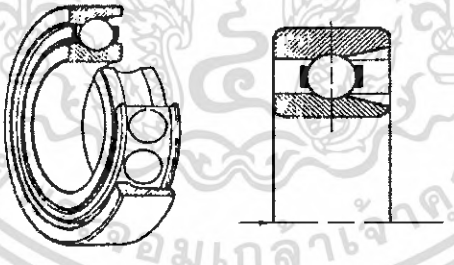
รูปที่ 2-5 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมร่องลึก

2. แบริ่งแบบเม็ดกลมร่องลึกมีรอยบากเติมเม็ดกลม (Filling Notch) จะมีรอยบากด้านหนึ่งของวงแหวนสำหรับเติมเม็ดกลม ซึ่งทำให้เพิ่มความสามารถในการรับภาระในแนวรัศมี แต่ความสามารถในการรับภาระในแนวแกนจะลดลง เนื่องจากเม็ดกลมชนรอยบาก



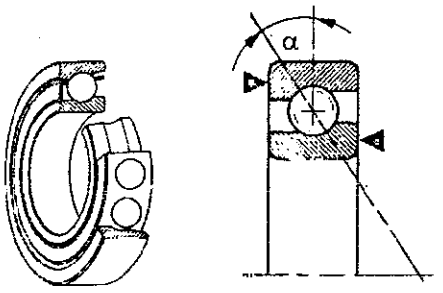
รูป 2-6 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมร่องลึกมีรอยบากเติมเม็ดกลม

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



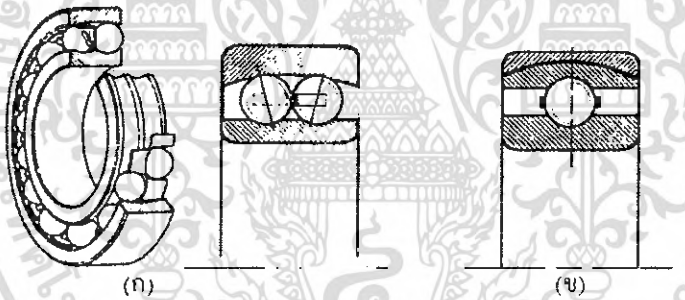
รูป 2-7 แสดงถึงลักษณะภายในของแม็กนีโตแบริ่ง

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



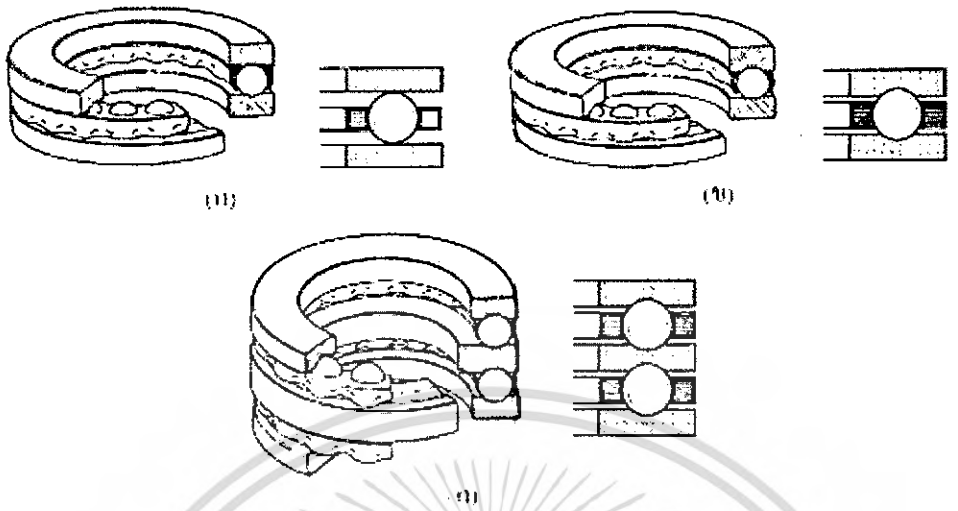
รูปที่ 2-8 แสดงถึงลักษณะภายในของแบริ่งแบบเม็ดกลมสัมผัสเชิงมุม

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



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

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



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

2.4 ลิเนียร์แบริ่ง (Linear Bearing)

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

รูปที่ 2-11 แสดงถึงลักษณะภายนอกของลิเนียร์แบริ่ง

2.5 มอเตอร์กระแสตรง

มอเตอร์กระแสตรงจะมีการทำงานเพียง 3 สเตจเท่านั้น คือ หมุนตามเข็มนาฬิกา หมุนทวนเข็มนาฬิกา และหยุดนิ่ง ซึ่งอัตราความเร็วในการหมุนและแรงบิดจะขึ้นอยู่กับกระแสและแรงดันที่จ่ายให้มัน ถ้าหากแรงบิดของมอเตอร์ไม่เพียงพอต่อการทำงานเราก็สามารถที่จะทำการทดรอบของการหมุนลงก็จะได้แรงบิดเพิ่มมากขึ้น

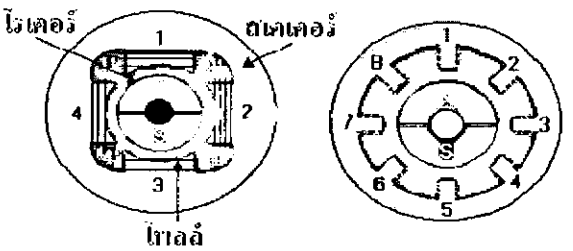
รูปที่ 2-12 แสดงถึงลักษณะภายนอกของมอเตอร์กระแสตรง

2.6 สเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์เป็นมอเตอร์ที่เมื่อป้อนกระแสไฟฟ้าให้กับมอเตอร์แล้ว จะทำให้มอเตอร์เกิดการหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งต่างจากมอเตอร์โดยทั่วไปที่จะหมุนทันทีและตลอดเวลาเมื่อป้อนแรงดันไฟฟ้า

รูปที่ 2-13 แสดงถึงโครงสร้างภายนอกของสเต็ปปิ้งมอเตอร์

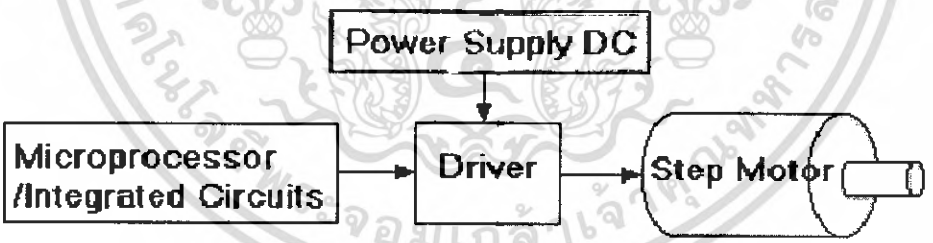
ข้อดีของสเต็ปปิ้งมอเตอร์ คือ สามารถกำหนดตำแหน่งของการหมุนด้วยตัวเลข , องศา หรือระยะทางได้อย่างละเอียดโดยใช้คอมพิวเตอร์ หรือไมโครคอนโทรลเลอร์เป็นเครื่องกำหนดและจัดเก็บตัวเลขโครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ทำจากแผ่นเหล็กวงแหวน ซึ่งแผ่นเหล็กแต่ละแผ่นจะมีซี่ยื่นออกมาประกบกันเป็นชั้น ๆ โดยที่แต่ละซี่นั้นจะมีขดลวดพันสวมอยู่ เมื่อมีการป้อนกระแสไฟฟ้าผ่านขดลวดจะทำให้เกิดสนามแม่เหล็กไฟฟ้า



รูปที่ 2-14 แสดงถึงโครงสร้างภายในของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์มักจะถูกนำไปใช้งานใช้งานลักษณะของระบบเปิด คือ สเต็ปปิ้งมอเตอร์สามารถทำงานได้โดยไม่ต้องมีการป้อนค่าพารามิเตอร์กลับมา แต่วิธีกำหนดตำแหน่งที่แน่นอนนั้นจะต้องมีการป้อนกลับไปยังระบบ และตัวบอกตำแหน่งว่าถูกต้องหรือผิดพลาดให้รับทราบ

ซึ่งสิ่งที่ใช้กับสเต็ปปิ้งมอเตอร์ คือ การนำลิมิตสวิตช์ติดตามตำแหน่งที่จะตรวจจับ เมื่อสเต็ปปิ้งมอเตอร์เริ่มทำงาน แล้วหมุนไปจนถึงตำแหน่งของสวิตช์ตรวจจับสัญญาณ ลิมิตสวิตช์ก็จะทำการป้อนสัญญาณกลับไปสู่ระบบ ดังนั้นระบบจะรู้ถึงตำแหน่งที่สเต็ปปิ้งมอเตอร์เคลื่อนที่ไปได้ตลอด ซึ่งตัวไมโครคอนโทรลเลอร์เองจะมีจุดอ้างอิงไว้ให้สำหรับการเริ่มต้นทำงาน และอ้างอิงตำแหน่งได้อย่างถูกต้อง



รูปที่ 2-15 แสดงถึงการควบคุมระบบสเต็ปปิ้งมอเตอร์

2.6.1 การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์

การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ จะทำการสั่งงานให้สเต็ปปิ้งมอเตอร์ทำงานไปทีละสเต็ป โดยการจ่ายกระแสไฟฟ้าเข้าไปยังขดลวดในแต่ละรอบบนของสเตเตอร์โดยการป้อนกระแสไฟฟ้าจะทำเป็นลำดับในรอบที่ถูกต้อง ซึ่งการป้อนกระแสไฟฟ้าเป็นลำดับจะแบ่งได้ 3 รูปแบบ ซึ่งทั้ง 3 แบบจะมีข้อดี-ข้อเสียต่างกันออกไป

2.6.1.1 การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบคลื่น

จะเป็นการกระตุ้นแบบที่ง่ายที่สุด ซึ่งจะทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่ง ๆ เรียงกันไป ตัวอย่างเช่น ขดที่ 1 , 2 , 3 , 4 , 1 , 2 , 3 , 4 เป็นลำดับอย่างนี้ หรือ ขด 1 , 4 , 3 , 2 , 1 , 4 , 3 , 2 เป็นลำดับกันไป ทั้งนี้ขึ้นอยู่กับทิศทางที่ต้องการให้มอเตอร์หมุน โดยวงจรที่นำมากระตุ้นนั้นจะมีราคาค่อนข้างจะถูกกว่า 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

2.6.1.2 การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบ 2 เฟส

การกระตุ้นขดลวดแบบนี้จะคล้ายกับการกระตุ้นในแบบคลื่น แต่จะแตกต่างกันตรงที่แบบ 2 phase จะกระตุ้นขดลวดทีละ 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และจะเรียงลำดับกันไปดังเช่นแบบเดียวกับแบบคลื่น ตัวอย่างเช่น ขดที่ 12 , 23 , 34 , 41 , 12, 23, 34 , 41 เรียงลำดับกันไปเรื่อย ๆ หรือจะเป็น 14 , 43 , 32 , 21 , 14 , 43 , 32 , 21 เรียงกันไปเรื่อย ๆ เช่นกัน

ข้อดีของการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบ 2 เฟส คือ จะให้แรงบิดได้มากกว่า แบบคลื่น เนื่องจากสเตเตอร์จะหมุนด้วยแรงดึงแบบเต็ม ๆ แรงจากทั้ง 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

2.6.1.3 การสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบครึ่งสเต็ป

เป็นรูปแบบผสมผสานของการกระตุ้นระหว่างแบบคลื่น กับ 2 เฟส เพื่อเพิ่มให้จำนวนรอบของสเต็ปมากขึ้นเป็น 2 เท่า ซึ่งในระบบนี้ จะทำการกระตุ้นขดลวดเรียงกันไปเรื่อย ๆ เป็นลำดับ ตัวอย่างเช่น ขดที่ 1 , 12 , 2 , 23 , 3 , 34 , 4 , 41 , 1 , 12 , 2 , 23 , 3 , 34 , 4 , 41 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

ข้อเสียของการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบนี้จะเป็นเช่นเดียวกับ แบบ 2 เฟส คือ ต้องจ่ายกำลังไฟเป็น 2 เท่าของแบบคลื่น 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			

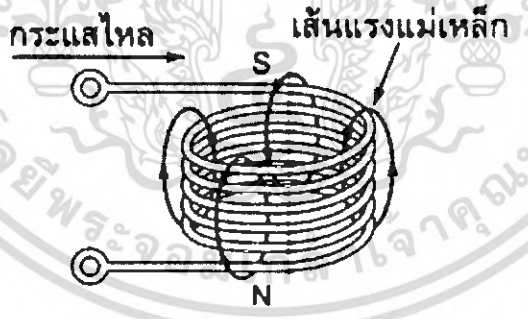
10	ON	ON		

ดารางที่ 2-3 แสดงถึงลำดับการสั่งงานควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบครึ่งเฟส

ซึ่งในการสร้างเครื่องเจาะแผ่นพลาสติกเพื่อสร้างแบบสำหรับทอพรม ใช้วิธีการกระตุ้นขดลวดแบบ 2 เฟส เพื่อให้ได้แรงบิดมากในช่วงสเต็ปสั้น ๆ เพื่อให้การเคลื่อนที่ในแต่ละรอบมีความละเอียดมากที่สุด

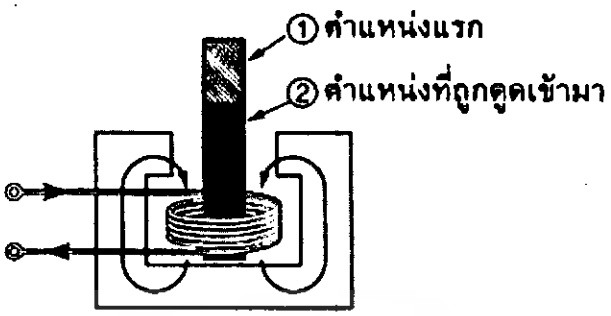
2.7 โซลินอยด์ไฟฟ้า

โซลินอยด์ไฟฟ้ามีหลักการทำงานดังนี้ คือ เมื่อมีกระแสไฟฟ้าไหลในขดลวดตัวนำใด ๆ ก็ตาม จะก่อให้เกิดสนามแม่เหล็กขึ้นรอบ ๆ ตัวนำนั้น และหากนำเส้นลวดมาขดเป็นวง ๆ หลาย ๆ วง ก็จะเกิดลักษณะของขดลวดขึ้น โดยสนามแม่เหล็กที่เกิดจากขดลวดแต่ละขดจะอยู่ในทิศทางเสริมกัน และก่อกำเนิดเป็นเส้นแรงของสนามแม่เหล็กถาวรแท่งหนึ่ง ซึ่งพร้อมที่จะดูดสารแม่เหล็กทันที แต่เนื่องจากสภาพรอบ ๆ ขดลวดอาจเป็นอากาศ เส้นแรงแม่เหล็กจึงไม่เข้มข้นมากนัก



รูป 2-16 แสดงถึงทิศทางของสนามแม่เหล็กที่เกิดขึ้นในขดลวดที่มีกระแสไหล

เพื่อไม่ให้สนามแม่เหล็กที่เกิดขึ้นกระจัดกระจาย จึงใส่แกนเหล็กอ่อนรูปตัวซีล้อมรอบ ๆ ขดลวดไว้เพื่อไม่ให้สนามแม่เหล็กที่เกิดขึ้นกระจัดกระจายออกไป และเพื่อให้สนามแม่เหล็กมีความเข้มมากขึ้น จากจุดนี้หากนำเอาแกนกระทุ้ง (Plunger) มาใส่เข้าไปตรงกลางวงของขดลวดในตำแหน่งที่ 1 แกนกระทุ้งจะถูกดูดให้ลึกเข้ามาจนสนิทในตำแหน่งที่ 2 ความสัมพันธ์ของแรงกับระยะช่วงชักของโซลินอยด์ ในช่วงชักไกล ๆ จะมีแรงน้อยมาก แต่ในทางตรงกันข้ามช่วงชักที่มีระยะใกล้ ๆ ก็จะมีแรงมากขึ้นเป็นทวีคูณ



รูปที่ 2 – 17 แสดงถึงการเพิ่มเหล็กอ่อนเข้ามาเพื่อเพิ่มความเข้มของสนามแม่เหล็ก

2.7.1 ประเภทของโซลินอยด์ไฟฟ้า

โซลินอยด์ไฟฟ้าสามารถแบ่งได้เป็น 2 ประเภท คือ โซลินอยด์ที่ใช้ไฟฟ้ากระแสตรง และโซลินอยด์ที่ใช้ไฟฟ้ากระแสสลับ ข้อแตกต่างระหว่างโซลินอยด์ใช้ไฟฟ้ากระแสตรง และโซลินอยด์ที่ใช้ไฟฟ้ากระแสสลับ คือ โซลินอยด์ที่ใช้ไฟฟ้ากระแสตรงจะก่อให้เกิดกระแสที่ไหลในขดลวดค่อนข้างคงที่ไม่เปลี่ยนแปลง ไม่ว่าแกนกระทุ้งจะอยู่ในตำแหน่งใดก็ตาม แต่โซลินอยด์ที่ใช้ไฟกระแสสลับจะมีกระแสในขณะที่แกนกระทุ้งอยู่นอกขดลวดค่อนข้างสูง และเมื่อแกนกระทุ้งถูกดูดเข้ามาจนสุดขดลวด กระแสจะลดต่ำลง ด้วยเหตุนี้เองที่ทำให้ต้องระวังอย่าให้เกิดการกระทุ้งในโซลินอยด์ไฟสลับ เพราะจะทำให้เกิดกระแสมาก ๆ ไหลค้างอยู่ ทำให้ขดลวดร้อนขึ้นและอาจจะไหม้เสียหายได้

ในการออกแบบและสร้างเครื่องเจาะแผ่นพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้จะใช้โซลินอยด์ที่ใช้ไฟกระแสตรงเพื่อเกิดกระแสในขดลวดที่คงที่ และมีอายุการใช้งานที่ยาวนาน

2.7.2 การเลือกใช้โซลินอยด์ไฟฟ้า

การเลือกใช้โซลินอยด์ไฟฟ้าควรพิจารณาถึงองค์ประกอบต่าง ๆ ดังนี้ คือ

1. แรงดันใช้งานซึ่งไม่ว่าจะเป็นไฟฟ้ากระแสตรง หรือไฟฟ้ากระแสสลับก็ตาม จำเป็นจะต้องดูความถี่ใช้งานให้ตรงตามความต้องการด้วย ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้โซลินอยด์ที่มีความถี่ 55 เฮิร์ซ

2. ช่วงชักในการใช้งาน (Operating Stroke) ของโซลินอยด์จะต้องเคลื่อนที่เป็นระยะทางเท่าใด มักจะกำหนดเป็นมิลลิเมตร ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้โซลินอยด์ที่มีช่วงชักในการใช้งานยาว 20 มิลลิเมตร

3. ขนาดของภาระ ว่าต้องใช้แรงขนาดเท่าใดมักจะกำหนดเป็นกรัม ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้โซลินอยด์รับภาระขนาดประมาณ 500 กรัม

4. เป็นการใช้งานอย่างต่อเนื่องหรือไม่ ซึ่งการใช้งานอย่างต่อเนื่องในที่นี้ หมายถึง การใส่แรงดันไฟเข้าขดลวดค้างไว้โดยขดลวดไม่ไหม้ หรือเป็นการใส่แรงดันแบบเป็นจังหวะ ๆ ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้การใส่แรงดันแบบเป็นจังหวะ ๆ เพื่อให้เกิดการกระทุ้งเพื่อทำการเจาะแผ่นพลาสติก

2.8 ไมโครคอนโทรลเลอร์ MCS-51

2.8.1 ความหมายของไมโครคอนโทรลเลอร์

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

2.8.2 โครงสร้างของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

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

1. หน่วยความจำโปรแกรมภายในตัวไมโครคอนโทรลเลอร์เป็นแบบแฟลช ทำให้สามารถลบและเขียนใหม่ได้นับพันครั้ง จึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดี่ยวได้โดยไม่ต้องใช้หน่วยความจำภายนอก ส่งผลให้สามารถใช้งานพอร์ตอินพุต-เอาท์พุตของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

2. การใช้ต้นทุนและเวลาในการพัฒนาระบบ ไมโครคอนโทรลเลอร์ลดลงอย่างมาก 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

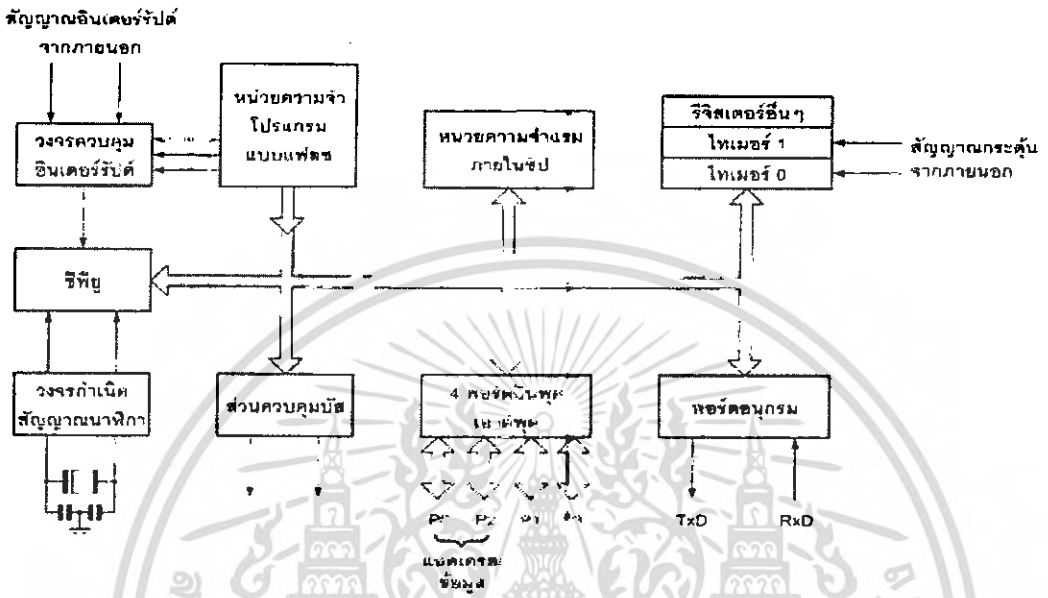
2.8.3 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx

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

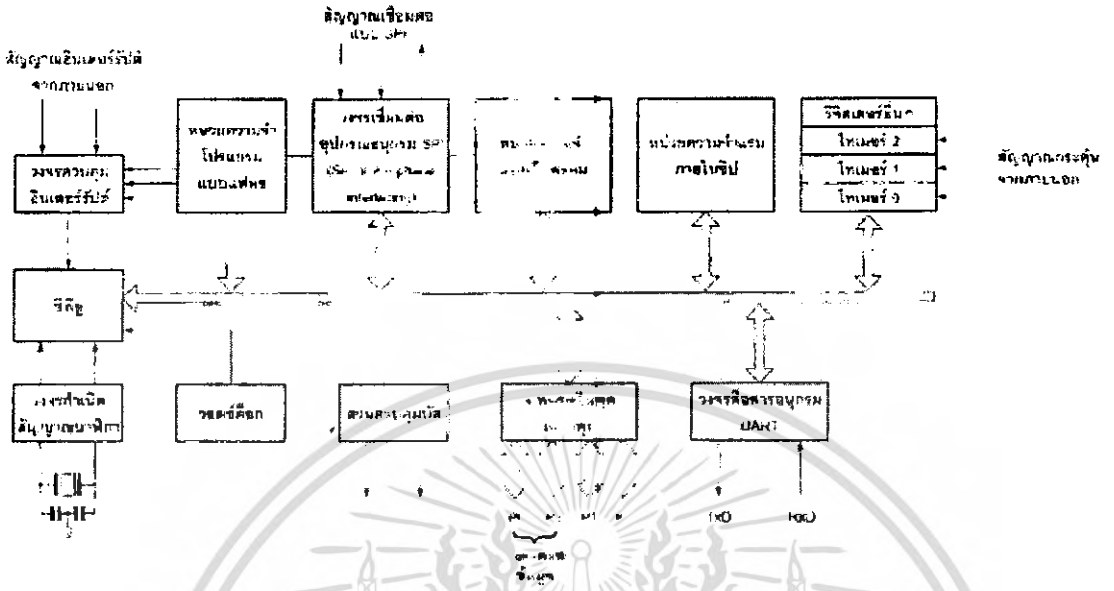
ด้านโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx พบว่าโครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแต่แตกต่างกันเฉพาะหน่วยความจำแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ใน

อนุกรม 87xx หน่วยความจำภายในจะเป็นแบบอีพรอม และบางเบอร์สามารถโปรแกรมได้เพียงอย่างเดียว



รูปที่ 2-18 แสดงถึงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx

ในโครงสร้างพื้นฐานของอนุกรม AT89xx พบว่า มีส่วนประกอบที่เพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน เช่น วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์อนุกรมนี้ใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบ หรือเรียกว่า การโปรแกรมบนวงจร และยังมีไทม์เมอร์/เคาน์เตอร์ขนาด 16 บิต ที่เพิ่มเติมเข้ามาอีก 1 ตัว เป็นไทม์เมอร์ และยังเพิ่มวงจรวอตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียูอีกด้วย



รูปที่ 2-19 แสดงถึงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Sxx

2.8.4 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังนี้

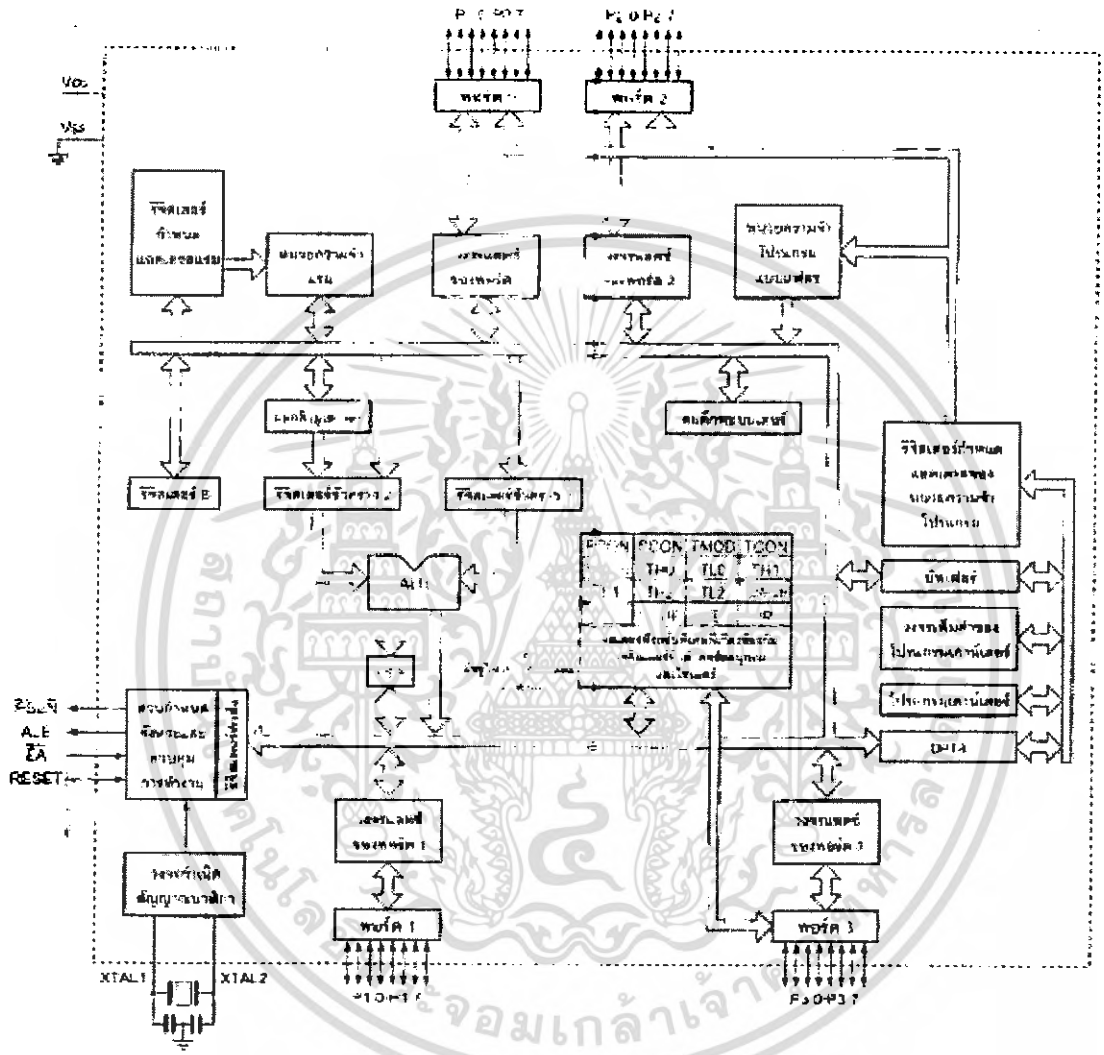
1. ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5v

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 นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบด์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

4. ขาพอร์ต 1 (P1.0-P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุต และเอาท์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทม์เมอร์ 2 และ P1.1 เป็นขาอินพุตริก

เกอร์ของไทม์เมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ



รูปที่ 2-20 แสดงถึงรายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของ Atmal

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

6. ขาพอร์ต 3 (P3.0-P3.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาท์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการตีดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (Float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดข้างต้นต่อไปนี้

- P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD
- P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD
- P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัป จากภายนอกช่อง 0 หรือขา *INT*0
- P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัป จากภายนอกช่อง 1 หรือขา *INT*1
- P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทม์เมอร์จากภายนอกช่อง 0 หรือขา T0
- P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเทอร์รัปจากภายนอกช่อง 1 หรือขา T1
- P3.6 ใช้เป็นขาสัญญาณ *WR* ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก
- P3.7 ใช้เป็นขาสัญญาณ *RD* ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายใน

รูปที่ 2-21 แสดงถึงการจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x

□ ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีนไซเคิล โดยที่วงจรดำเนินสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

□ ขา ALE/PROG (Address Latch Enable/Program Pulse Input) เป็นขาที่ใช้ในการควบคุมการแลตซ์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขาสำหรับรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีพรอม

□ ขา PSEN (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง ในแต่ละแมชชีนไซเคิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใด ๆ ออกมา

□ ขา EA/Vpp (External Access Enable/Programming Voltage Input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายนอก แต่ถ้าหากขานี้เป็น"1"เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูง สำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-5I แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12v

□ ขา XTAL1 และขา XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

2.8.5 โครงสร้างและการทำงานของพอร์ตในไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพรอ์ตให้ใช้งานทั้งสิ้น 4 พอร์ต คือ พอร์ต 0 จนถึง พอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้า และเอาท์พุตสำรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงแลตซ์ และวงจรขับตลอดจนบัฟเฟอร์อินพุต ดังแสดงในรูปที่ 2-20

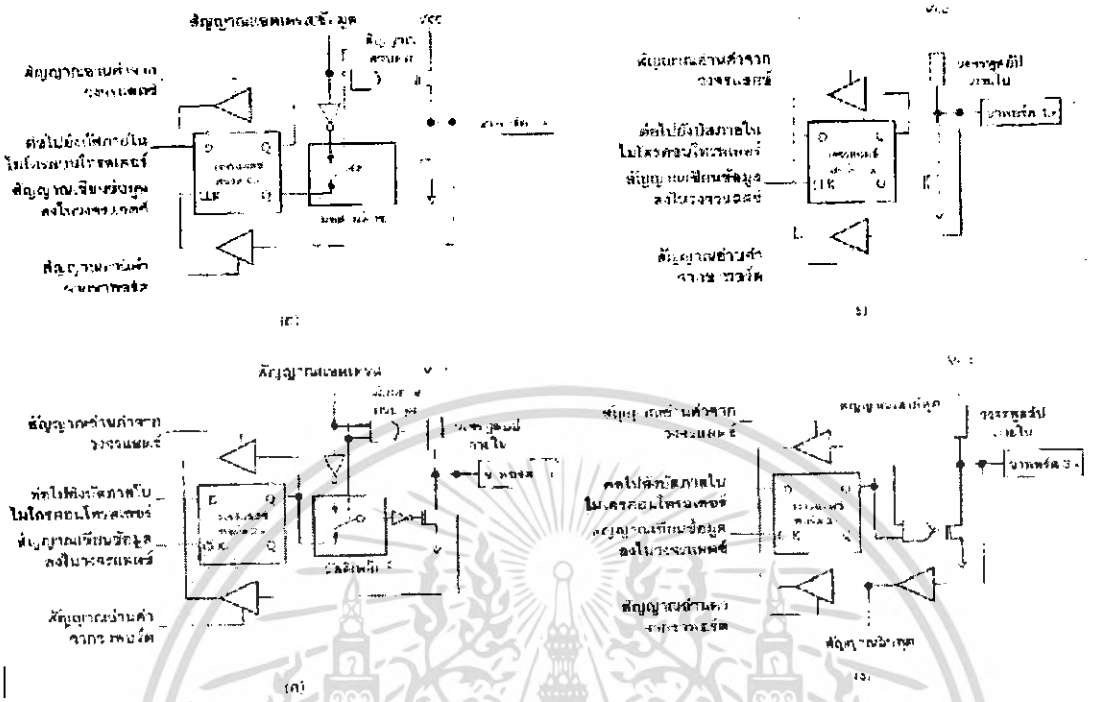
ที่พอร์ต 0 และพอร์ต 2 จะใช้งานเป็นพอร์ตอินพุต และเอาท์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ต และพอร์ต I บางขา นอกจาก

จะใช้เป็นขาพอร์ตอินพุต-เอาท์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด

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

พอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน

วงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงบจรพูลอัปเพิ่มเติมเข้ามา ส่วนวงจรภายในของพอร์ต 3 พบว่ามีลักษณะคล้ายกับพอร์ต 1 แต่มีการเพิ่มเติมวงจรบัฟเฟอร์ และวงจรอินพุต-เอาท์พุต เพื่อทำงานในฟังก์ชันพิเศษเข้า เนื่องจากพอร์ต 3 สามารถนำไปใช้งานหน้าที่พิเศษได้ทุกขา



รูปที่ 2-22 แสดงถึงวงจรภายในของพอร์ตทุกพอร์ต 0 (ก) , พอร์ต 1 (ข) พอร์ต 2 (ค) และพอร์ต 3 (ง) ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

2.8.6 การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตและเอาท์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุตต้องเริ่มต้นด้วยการเขียนข้อมูล “1” มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานอินพุต เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้น ๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น “1” สามารถรับสัญญาณลอจิก “0” จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนี้ อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ควรกำหนดให้ทำงานในสภาวะลอจิก “0” จะดีและสะดวกที่สุด ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก “0” แล้ว

2.8.7 การใช้งานเป็นพอร์ตเอาท์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาท์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล "0" ออกไปทางเอาท์พุตก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปยังเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก "0" ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล "1" ออกไปก็ให้เขียนข้อมูล "1" ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาท์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่งอย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาท์พุต

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 ในกรณีที่ใช้งานทุกพอร์ตเอาท์พุตจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาท์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาท์พุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง

บทที่ 3
ความรู้เบื้องต้นเกี่ยวกับพอร์ตขนาน

3.1 ทำไมถึงเลือกใช้งานพอร์ตขนาน

เมื่อเทียบกับการใช้งานการ์ดอินพุตเอาต์พุตที่ต้องติดตั้งอยู่ภายในเครื่องคอมพิวเตอร์แล้ว พอร์ตขนานมีข้อได้เปรียบอยู่หลายประการดังนี้

ในด้านความปลอดภัย การที่ต้องถอดฝาเครื่องคอมพิวเตอร์ออกมาเพื่อเสียบการ์ดเชื่อมต่อลงในสล็อตของคอมพิวเตอร์ อาจจะทำให้เกิดความเสียหายกับส่วนอื่น ๆ ของคอมพิวเตอร์ได้ ถ้าผู้ที่ใช้งานไม่มีความชำนาญหรือเกิดการต่อวงจรที่ผิดพลาด

ในด้านการเข้ากันได้กับคอมพิวเตอร์ส่วนใหญ่ การเชื่อมต่อโดยใช้การ์ดที่เสียบลงในสล็อตไม่สามารถใช้กับคอมพิวเตอร์ในปัจจุบันได้ทุกรุ่น ยกตัวอย่าง คอมพิวเตอร์โน้ตบุ๊ก จะไม่มีสล็อตเสียบแต่จะมีที่เสียบการ์ด PCMCIA แทน ในขณะที่พอร์ตขนานจะมีติดตั้งอยู่ในคอมพิวเตอร์ทุกเครื่อง ทั้งนี้เพื่อใช้ในการติดต่อกับเครื่องพิมพ์

ข้อจำกัดด้านพื้นที่ คอมพิวเตอร์บางเครื่องมีการเสียบการ์ดเชื่อมต่อตัวอื่น ๆ อยู่แล้ว อาทิ การ์ดเสียง การ์ดโมเด็ม เป็นต้น จนไม่มีสล็อตเหลือพอสำหรับการเสียบการ์ดเชื่อต่อเพิ่มเติม

ความสะดวกในการใช้งาน การเชื่อมต่อทางพอร์ตขนานสามารถทำได้ง่าย ๆ เพียงต่อสายสำหรับเชื่อมต่อเข้ากับคอนเน็กเตอร์ DB-25 ของพอร์ตขนาน

จำนวนช่องสัญญาณอินพุตเอาต์พุต พอร์ตขนานมีจำนวนพอร์ตอินพุตเอาต์พุตมากเพียงพอที่จะนำไปใช้งานต่าง ๆ และยังสามารถขยายให้มีจำนวนพอร์ตเพิ่มขึ้นได้ โดยพอร์ตขนานปกติมีจำนวนขาเอาต์พุต 12 ขา และขาอินพุต 5 ขา

ความเร็วในการสื่อสารข้อมูลกับพอร์ตขนาน มีความเร็วเท่ากับการติดต่อกับระบบบัสโดยตรง และมีความเร็วมากกว่าการติดต่อผ่านทางพอร์ตอนุกรม

อะไหล่และชิ้นส่วนประกอบ คอนเน็กเตอร์และสายเชื่อมต่อต่าง ๆ ของการเชื่อมต่อผ่านทางพอร์ตขนาน หาได้ง่ายและราคาไม่แพง หรือจะสร้างขึ้นเองก็สามารถทำได้อย่างง่ายดาย

จากคุณสมบัติดังที่ได้กล่าวมาแล้วนั้นทำให้พอร์ตขนานเหมาะอย่างยิ่งที่จะนำมาใช้ในการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกเพื่อควบคุมหรือรับสัญญาณข้อมูล นากจากนั้นหากนำคุณสมบัติของการเขียนโปรแกรมง่าย ๆ ผ่านระบบปฏิบัติการวินโดวส์ด้วยโปรแกรม Visual BASIC ก็จะสามารถสร้างระบบการเชื่อมต่อที่สมบูรณ์และใช้งานง่ายได้ไม่ยาก

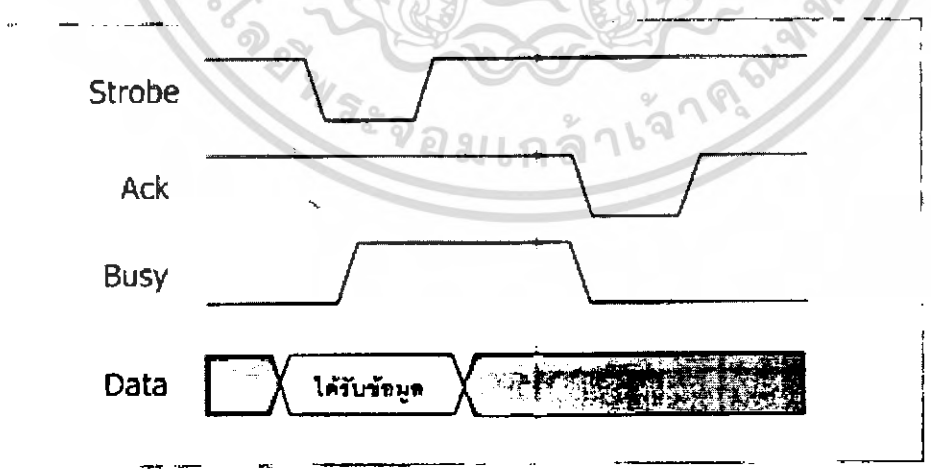
3.2 ความรู้เบื้องต้นของพอร์ตขนาน

พอร์ตขนาน (Parallel port) สาเหตุที่มีชื่อนี้ เนื่องจากการถ่ายทอดข้อมูลของพอร์ตนี้เป็นแบบขนาน สำหรับชื่อเรียกอีกชื่อของพอร์ตขนานคือ พอร์ตเครื่องพิมพ์ (Printer port) เนื่องจากพอร์ตนี้ใช้สำหรับต่อเครื่องพิมพ์นั่นเอง

ด้วยการถ่ายทอดข้อมูลแบบขนานนี้เอง ทำให้พอร์ตขนานมีอัตราการถ่ายทอดข้อมูลสูงกว่าการถ่ายทอดข้อมูลแบบอนุกรมประมาณ 8 - 10 เท่า และการประมวลผลข้อมูลส่วนใหญ่จะมีขนาด 8 บิต ดังนั้นพอร์ตขนานจึงสามารถรองรับการถ่ายทอดข้อมูล 8 บิตได้โดยไม่ต้องต่อส่วนเพิ่มเติมใด ๆ

3.3 ลักษณะทางกายภาพของพอร์ตขนาน

เพื่อให้เข้าใจถึงการนำเอาพอร์ตขนานไปใช้งาน ก่อนอื่นต้องมาทำความเข้าใจก่อนว่าปกตินั้นการสั่งพิมพ์งานจากคอมพิวเตอร์ไปยังพอร์ตขนานนั้นมีรูปแบบการทำงานภายในอย่างไร ในรูปที่ 3 -1 แสดงไดอะแกรมเวลาของติดต่อระหว่างพอร์ตขนานกับเครื่องพิมพ์ ซึ่งจะเห็นได้ว่ามีสัญญาณที่ใช้งานจริง ๆ มีไม่มาก เริ่มจากสัญญาณพอร์ต Data ถูกส่งออกไปยังเครื่องพิมพ์ พร้อมทั้งส่งสัญญาณ Strobe ออกไปด้วย เพื่อให้เครื่องพิมพ์รับรู้ว่ามีการส่งข้อมูลใหม่มาที่ขา Data แล้ว จากนั้น คอมพิวเตอร์จะต้องรอการตอบกลับจากเครื่องพิมพ์ นั่นคือเครื่องพิมพ์จะสร้างสัญญาณ Busy หรือพื่อบอกว่าเครื่องพิมพ์ยังไม่พร้อมที่จะรับข้อมูลใหม่ จนกระทั่งเมื่อเครื่องพิมพ์พร้อม เครื่องพิมพ์จะสร้างสัญญาน ACK ส่งไปยังคอมพิวเตอร์เพื่อแจ้งว่า พร้อมที่จะรับข้อมูลใหม่แล้ว



รูปที่ 3-1 แสดงไดอะแกรมเวลาของการส่งข้อมูลไปยังเครื่องพิมพ์

สัญญาณข้อมูลขนาด 8 บิต , สัญญาณ Strobe และสัญญาณ ACK (acknowledge) เป็นสัญญาณที่สำคัญในการส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ นอกจากสัญญาณทั้งสามแล้ว ส่วนใหญ่การติดต่อกับเครื่องพิมพ์ยังต้องมีสัญญาณอื่น ๆ ร่วมด้วย เนื่องจากเครื่องพิมพ์ต้องทำหน้าที่ถึง 3 อย่างด้วยกันคือ รับข้อมูลจากคอมพิวเตอร์ , พิมพ์ข้อมูลที่รับเข้ามา และตอบสนองต่อการใช้งานของผู้ใช้ เช่น การเปลี่ยนฟอนต์ เป็นต้น บางครั้งอาจเกิดเหตุการณ์ที่ไม่ปกติ เช่น บัฟเฟอร์สำหรับรับข้อมูลเต็ม (เนื่องจากเครื่องพิมพ์เป็นอุปกรณ์ที่ทำงานทางกลย่อมทำงานได้ช้ากว่าการส่งข้อมูลของคอมพิวเตอร์) เครื่องพิมพ์จะต้องแจ้งไปยังคอมพิวเตอร์ว่าให้หยุดส่งข้อมูลชั่วคราว เนื่องจากไม่สามารถรับข้อมูลมากกว่านี้ได้แล้ว สัญญาณที่ส่งจากเครื่องพิมพ์ไปยังคอมพิวเตอร์คือสัญญาณ Busy และเมื่อเครื่องพิมพ์เกิดข้อผิดพลาด เช่น กระดาษดิด เครื่องพิมพ์จะต้องแจ้งไปยังคอมพิวเตอร์เช่นกัน โดยสัญญาณที่แจ้งไปยังคอมพิวเตอร์เรียกว่าสัญญาณ Error นอกจากนี้เมื่อคอมพิวเตอร์ต้องการรีเซตเครื่องพิมพ์ คอมพิวเตอร์จะต้องส่งสัญญาณ Reset ไปยังเครื่องพิมพ์เพื่อรีเซตเครื่องพิมพ์ด้วย สามารถสรุปขาสัญญาณที่จำเป็นสำหรับการติดต่อดังในตารางที่ 3-1

จากตางรางที่ 3-1 จะเห็นได้ว่าพอร์ตขนานของคอมพิวเตอร์ยังแยกย่อยออกเป็นอีก 3 พอร์ต ได้แก่ พอร์ตเอาต์พุตที่ทำหน้าที่ส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ พอร์ตเอาต์พุตสำหรับสัญญาณ Strobe และ Reset พอร์ตอินพุตสำหรับการอ่านค่าสัญญาณ Acknowledge , Busy และสัญญาณ Error จากเครื่องพิมพ์

สัญญาณ	หน้าที่การทำงาน	ทิศทาง
ข้อมูล 8 บิต	ข้อมูลที่ส่งจากคอมพิวเตอร์ไปยังเครื่องพิมพ์	คอมพิวเตอร์
Strobe	แจ้งเครื่องพิมพ์ถึงข้อมูลที่ส่งมาใหม่	คอมพิวเตอร์
Acknowledge	เครื่องพิมพ์แจ้งมายังคอมพิวเตอร์ว่าได้รับข้อมูลแล้ว	เครื่องพิมพ์
Busy	แจ้งสถานะว่าเครื่องพิมพ์ไม่ว่าที่จะรับข้อมูลใหม่	เครื่องพิมพ์
Error	แจ้งสถานะว่าเครื่องพิมพ์เกิดข้อผิดพลาด	เครื่องพิมพ์
Reset	รีเซตเครื่องพิมพ์	คอมพิวเตอร์

ตารางที่ 3-1 สัญญาณสำคัญ ๆ ของพอร์ตขนานที่ใช้ติดต่อกับเครื่องพิมพ์

โดยปกติพอร์ตขนานออกแบบมาให้มีสายสัญญาณอยู่ทั้งหมด 17 เส้น สายสัญญาณเหล่านั้นจะมีรีจิสเตอร์ 3 ตัวควบคุมการทำงานดังนี้

1. พอร์ตเอาต์พุตสำหรับสัญญาณข้อมูล 8 เส้น มีรีจิสเตอร์ Data ควบคุม

2. พอร์ตอินพุตสำหรับการอ่านค่าสถานะต่าง ๆ จากภายนอกมีอยู่ด้วยกัน 5 เส้น ใช้รีจิสเตอร์ Status ในการควบคุม

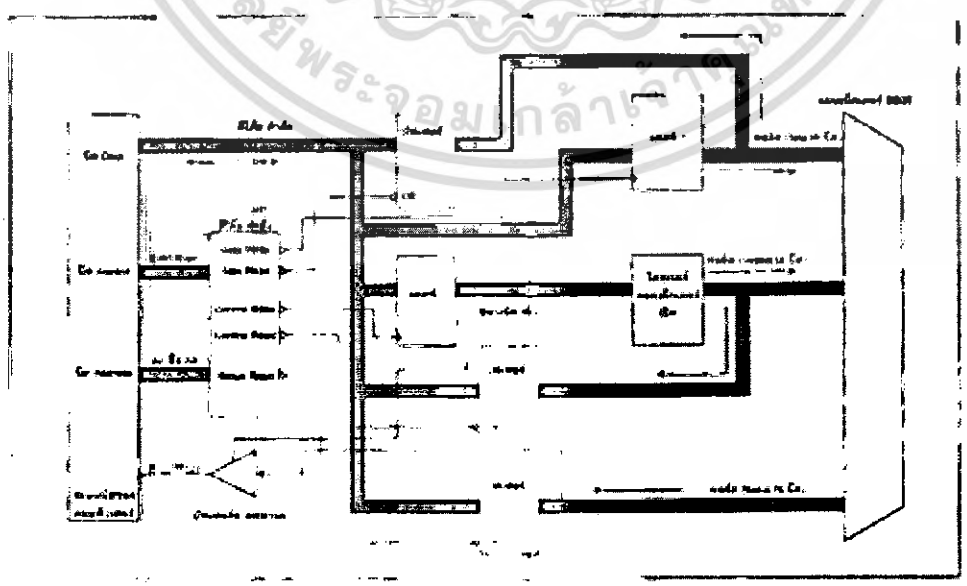
3. พอร์ตเอาต์พุตสำหรับส่งสัญญาณควบคุมไปยังอุปกรณ์ภายนอก มีอยู่ด้วยกัน 4 เส้น ใช้รีจิสเตอร์ Control ในการควบคุม

บล็อกไดอะแกรมในรูปที่ 3-2 แสดงระบบบัสของคอมพิวเตอร์สำหรับการติดต่อกับพอร์ตขนาน สัญญาณเอาต์พุตจากพอร์ตขนานจะถูกส่งไปยังคอนเน็กเตอร์แบบ DB – 25 สำหรับคอมพิวเตอร์ส่วนใหญ่ในปัจจุบันพอร์ตขนานจะมีมาพร้อมกับเมนบอร์ด ไม่จำเป็นรต้องใช้การ์ดเสียบเพิ่มเติมเหมือนในอดีต พร้อมทั้งมีฟังก์ชันการทำงานที่ซับซ้อนขึ้น แต่ยังคงสนับสนุนการทำงานของพอร์ตขนานในรูปแบบมาตรฐาน (SPP) อยู่

เมื่อดูจากรูปที่ 3–1 เทียบการทำงานโดยทั่วไปกับการเชื่อมต่อผ่านการ์ดที่เสียบลงในสล็อตของคอมพิวเตอร์แล้ว พอร์ตขนานจะมีลักษณะใกล้เคียงกัน โดยการติดต่อกับพอร์ตขนานจะต้องมีการอ้างแอดเดรส ตำแหน่งแอดเดรสที่ใช้อ้างถึงจะเป็นตำแหน่ง A0 – A9 และใช้ขา IOR และ IOW สำหรับเป็นตัวเลือกว่าต้องการอ่านหรือเขียนรีจิสเตอร์ตัวใด จากการดีโค้ดแอดเดรส A0 – A9 นี้เองทำให้ได้สัญญาณออกมาเพื่อไปควบคุมหรือเอ็นเอเบิลวงจรบัฟเฟอร์ต่าง ๆ ดังนี้

Data Write สัญญาณเอ็นเอเบิลสำหรับนำข้อมูลที่อยู่ในบัส Data ไปออกที่ขา Data ของพอร์ตขนาน

Data Read สัญญาณเอ็นเอเบิลสำหรับอ่านข้อมูลจากขา Data ของพอร์ตขนานมาเก็บไว้ในบัส Data



รูปที่ 3-2 แแสดงระบบบัสภายในของพอร์ตขนาน

ControlWrite สัญญาณเอ็นเอเบิลสำหรับนำข้อมูลที่อยู่ในบัส Data ไปออกที่ขา Control ของพอร์ตขนาน สำหรับพอร์ตนี้นอกจากจะส่งข้อมูลออกไปยังพอร์ตขนานแล้ว ยังทำหน้าที่อีนาเบิลการอินเตอร์รัปต์ของการเปลี่ยนแปลงสัญญาณที่พอร์ต Status อีกด้วย

ControlRead สัญญาณเอ็นเอเบิลสำหรับอ่านค่าข้อมูลจากขา Control มาเก็บไว้ไบัส Data

StatusRead สัญญาณเอ็นเอเบิลสำหรับอ่านค่าข้อมูลจากขาพอร์ต Status มาเก็บไว้ในบัส Data

ตารางที่ 3–2 แสดงชื่อและหน้าที่การทำงานของตำแหน่งขาต่าง ๆ บนพอร์ตขนาน ส่วนใน ตารางที่ 3–3 แสดงแอดเดรสของพอร์ตขนาน ซึ่งกำหนดไว้ 3 ตำแหน่ง คือ LPT1 , LPT2 และ LPT3

3.4 พอร์ตคาต้า (Data Port)

จากรูปที่ 3–3 แสดงให้เห็นว่าพอร์ต Data ประกอบไปด้วยบัฟเฟอร์ 1 ตัวและไอซีแลตช์อีก 1 ตัว เมื่อคอมพิวเตอร์ต้องการส่งข้อมูลไปยังเครื่องพิมพ์ คอมพิวเตอร์จะเขียนข้อมูลไปยังไอซีแลตช์ 1 ทั้ง 8 บิต เอาต์พุตของไอซีแลตช์ 1 คือ D0 – D7 ซึ่งเอาต์พุตนี้จะไปปรากฏอยู่ที่พอร์ตขนานในตำแหน่งขา 2 ถึงขา 9 และที่ขาเอาต์พุตนี้สัญญาณ Data จะส่งกลับไปเป็นอินพุตของบัฟเฟอร์ 1 ด้วย ทำให้คอมพิวเตอร์สามารถอ่านค่าสถานะปัจจุบันที่เกิดขึ้นกับพอร์ต Data ได้

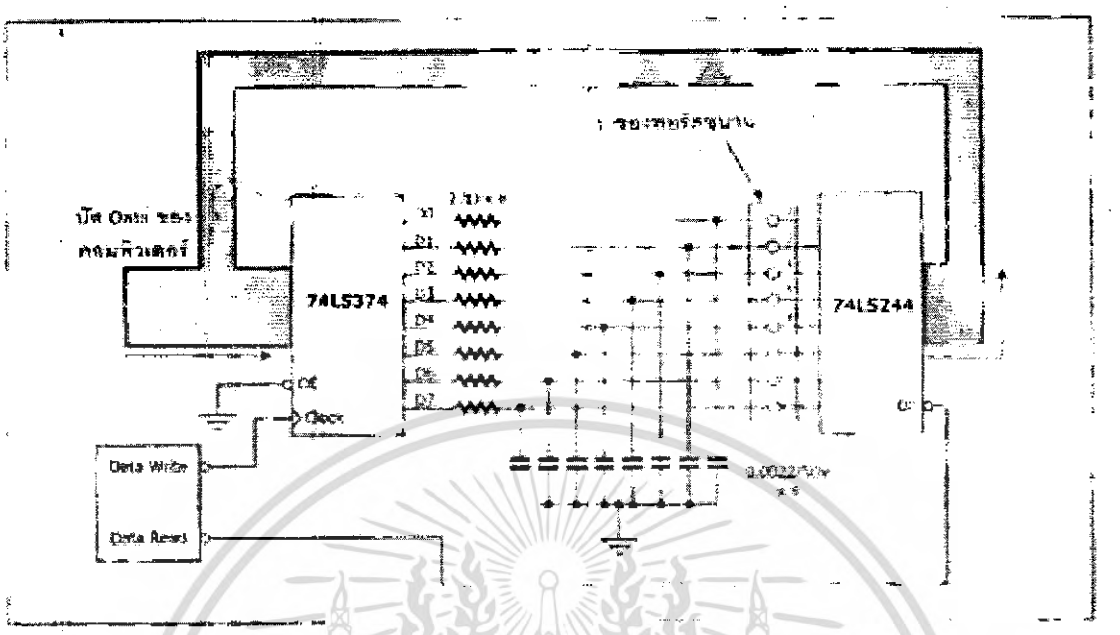
เมื่อคอมพิวเตอร์ส่งข้อมูล ข้อมูลจะถูกส่งมาจากบัสข้อมูลของคอมพิวเตอร์ผ่านไปให้กับไอซี 74LS374 ซึ่งเป็นไอซีแลตช์ข้อมูล และเมื่อต้องการให้ข้อมูลปรากฏที่เอาต์พุต คอมพิวเตอร์จะส่งสัญญาณ Data Write ออกไปที่ขา CLK ของ 74LS374 เอาต์พุตจาก 74LS374 จะถูกกรองด้วยวงจร RC ซึ่งประกอบด้วยตัวต้านทานค่า 27Ω และตัวเก็บประจุ 0.0022μF เพื่อให้ช่วงเวลาที่เปลี่ยนจากลอจิก “0” เป็นลอจิก “1” หรือจากลอจิก “1” เป็นลอจิก “0” เป็นไปอย่างช้า ๆ เนื่องจากการเปลี่ยนแรงดันที่รวดเร็วทำให้เกิดสัญญาณรบกวนเหนี่ยวนำข้ามไปยังข้อมูลบิตอื่น ๆ ได้ ทำให้ข้อมูลที่ส่งออกไปมีข้อผิดพลาด จากค่าตัวต้านทานและตัวเก็บประจุในวงจรทำให้เกิดการหน่วงเวลาไปประมาณ 60 นาโนวินาที จากวงจรในรูปที่ 3–3 ทำให้เอาต์พุตของพอร์ต Data มีคุณสมบัติดังนี้

- กระแสซิงก์สูงสุด 24 mA
- กระแสซอร์สสูงสุด 2.6 mA
- ระดับแรงดันของลอจิก “1” ต่ำสุดเท่ากับ 2.4V
- ระดับแรงดันสูงสุดสำหรับลอจิก “0” เท่ากับ 0.5V

สำหรับบัฟเฟอร์สำหรับการอ่านข้อมูลกลับได้แก่เบอร์ 74LS244 ซึ่งเมื่อต้องการอ่านค่าคอมพิวเตอร์จะส่งสัญญาณ DataRead ออกมาเพื่อเอ็นเอเบิลไอซี 74LS244 สำหรับพอร์ตขนานแบบมาตรฐาน (Standard Parallel Port : SPP) พอร์ต Data จะต้องใช้เพื่อการส่งค่าออกเอาต์พุตเท่านั้น แต่สำหรับพอร์ตขนานที่มีการสื่อสารสองทิศทาง (Bidirectional Parallel Port) สามารถอ่านค่าจากพอร์ต Data ได้ด้วย แต่ก่อนที่จะอ่านค่าต้องจำไว้เสมอว่าจะต้องป้อนค่าเอาต์พุตให้มีค่าลอจิก "1" ทั้งหมดก่อน

ขาของพอร์ตขนาน	รีจิสเตอร์	ทิศทาง	ตำแหน่งบิต	ชื่อสัญญาณ	หน้าที่การทำงาน
1	Control	Out	C0	STROBE	ลอจิค "0" ส่งออกไปเพื่อบอกว่า
2-9	Data	Out	D0-D7	DATA0-DATA7	สำหรับพอร์ตขนานมาตรฐานเส้นขานี้
10	Status	In	S6	nACK	เป็นพัลส์ลอจิก "0" ที่ส่งมาจากเครื่องพิมพ์
11	Status	In	S7	BUSY	เป็นสัญญาณแจ้งมาจากเครื่องพิมพ์ว่า
12	Status	In	S5	PE	แจ้งการขาดแคลน
13	Status	In	S4	SELECT	แจ้งว่าเครื่องพิมพ์ต่ออยู่
14	Control	Out	C1	AUTO FEED	สั่งเครื่องพิมพ์ให้ขึ้นบรรทัดใหม่
15	Status	In	S3	ERROR	สัญญาณจากเครื่องพิมพ์มาสั่งคอมพิวเตอร์
16	Control	Out	C2	INIT	รีเซตเครื่องพิมพ์โดยให้ลอจิก "0"
17	Control	Out	C3	SELECT-IN	ส่งสัญญาณไปสั่งเครื่องพิมพ์เพื่อแจ้งว่า
18-25				GND	กราวด์

ตารางที่ 3-2 แสดงสัญญาณทั้งหมดที่อยู่บนพอร์ตขนาน



รูปที่ 3-3 วงจรภายในของพอร์ต Data

3.5 พอร์ตคอนโทรล (Control Port)

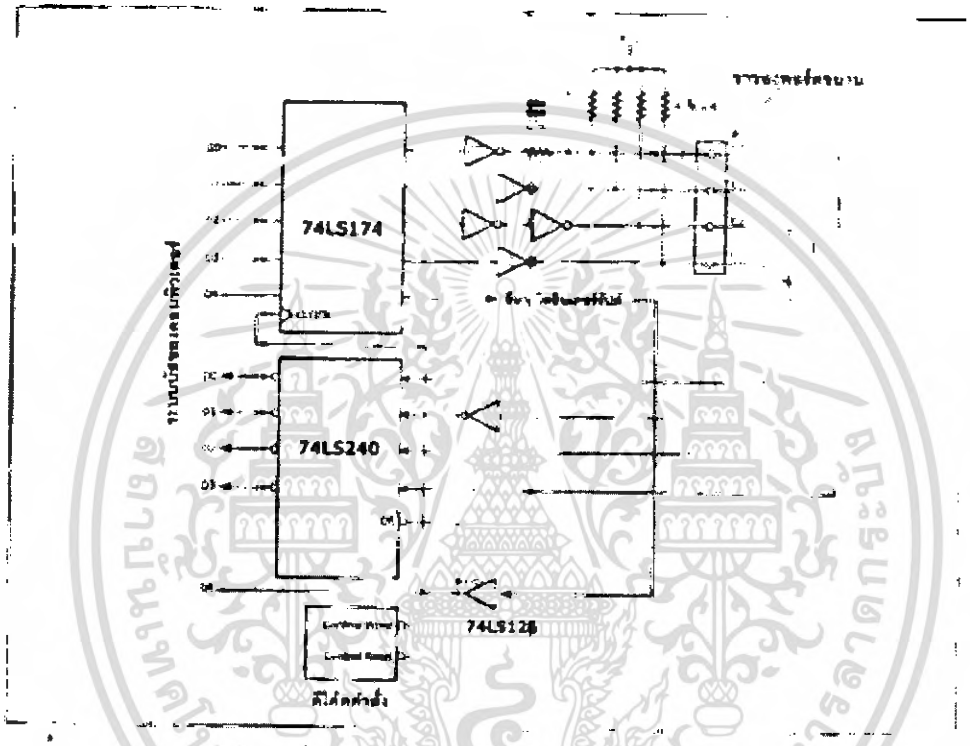
พอร์ต Control ใช้สำหรับคอมพิวเตอร์ควบคุมเครื่องพิมพ์ จากตารางที่ 3–2 จะเห็นว่าพอร์ต Control ประกอบไปด้วยบิตเอาต์พุต 4 บิตที่ต่อออกไปยังเครื่องพิมพ์ ส่วนบิตเอ็นเอเบิลอินเตอร์รัปต์ไม่ได้ถูกต่อออกไป รูปที่ 3–4 แสดงบล็อกไดอะแกรมของพอร์ต Control จะเห็นว่าเอาต์พุตของพอร์ต Control มีอินเวอร์เตอร์แบบคอลเล็กเตอร์เปิดต่อร่วมอยู่ โดยเอาต์พุตเหล่านี้จะถูกพูลอัปไว้ด้วยตัวต้านทานค่า 4.7kΩ สำหรับบิต C2 จะผ่านอินเวอร์เตอร์ถึงสองตัวทำให้ที่เอาต์พุตของบิต C2 ไม่มีการกลับสถานะลอจิก

สถานะของพอร์ต Control สามารถอ่านกลับได้และด้วยการใช้บัฟเฟอร์เบอร์ 74LS240 ซึ่งที่เอาต์พุตมีอินเวอร์เตอร์ต่ออยู่ภายใน ทกให้ค่าที่อ่านได้ตรงกับค่าที่ส่งออกไป การควบคุมการอ่านและเขียนข้อมูลกับพอร์ต Control คอมพิวเตอร์จะส่งข้อมูลมาที่ขา Control Write และ Control Read

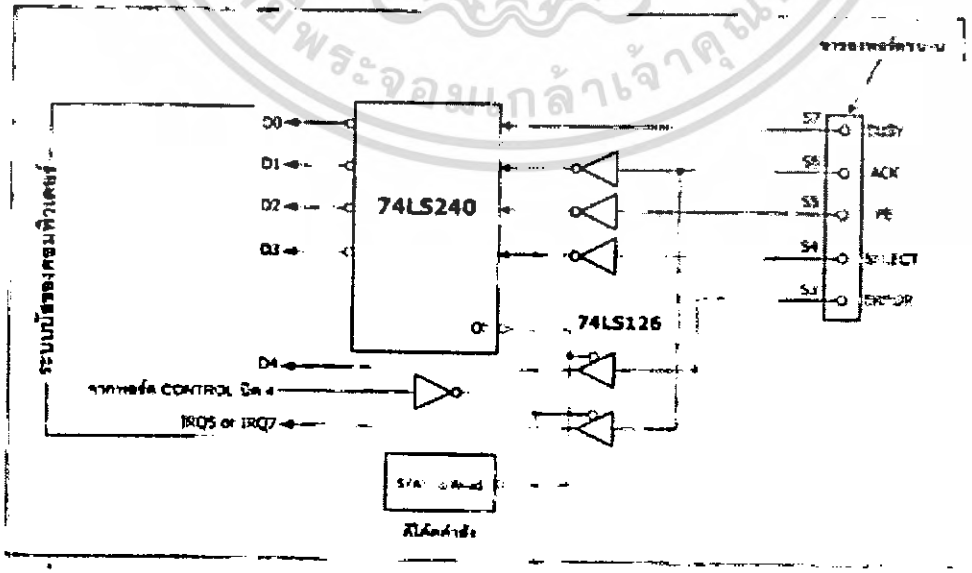
เนื่องจากเอาต์พุตของพอร์ต Control เป็นแบบคอลเล็กเตอร์เปิด ดังนั้นผู้ใช้งานสามารถใช้พอร์ตนี้ในการอ่านค่าสัญญาณอินพุตจากภายนอกได้ โดยก่อนที่จะอ่านค่าจะต้องทำให้ขาพอร์ตที่ต้องการอ่านค่ามีลอจิก “1” เสียก่อน

3.6 พอร์ตแสดงสถานะหรือพอร์ต Status

พอร์ต Status เป็นพอร์ตที่คอมพิวเตอร์ใช้สำหรับการอ่านค่าสถานะจากเครื่องพิมพ์ รูปที่ 3–5 แสดงรายละเอียดภายในของพอร์ต Status จะสังเกตได้ว่ามีขาสัญญาณอยู่ทั้งหมด 5 สัญญาณด้วยกันและจะเรียกชื่อเป็น S3 , S4 , S5 , S6 และ S7 ซึ่งตัวเลขนั้นหมายถึงตำแหน่งบิตของขาเหล่านี้



รูปที่ 3-4 วงจรภายในของพอร์ต Control



รูปที่ 3-5 แสดงวงจรภายในของพอร์ตแสดงสถานะ

ภายในรีจิสเตอร์ Status นั่นเอง สำหรับบิต S7 จะมีข้อแตกต่างจากบิตอื่น ๆ ที่เมื่อสัญญาณจากภายนอกส่งเข้ามาแล้วจะไม่ผ่านอินเวอร์เตอร์ ในขณะที่ขาอื่น ๆ ผ่านอินเวอร์เตอร์ทั้งหมด ดังนั้นเมื่อข้อมูลผ่านจากขาอินพุตไปยัง 74LS240 ซึ่งเอาต์พุตมีการกลับสถานะทำให้บิต S7 เป็นบิตเดียวที่มีการกลับสถานะ นอกจากนี้ในการใช้งานถ้าต้องการให้มีการสร้างอินเตอร์รัปต์จากขอบขาขึ้นของขา S6 สามารถกำหนดค่าได้จากพอร์ต Control บิต 4

3.7 การนำพอร์ตขนานไปใช้งาน

สำหรับพอร์ตขนานแบบมาตรฐาน ผู้ใช้งานสามารถนำพอร์ตอินพุต 5 บิต (พอร์ต Status) พอร์ตเอาต์พุต 4 บิต (พอร์ต Control) และพอร์ตเอาต์พุตอีก 8 บิต (พอร์ต Data) ไปใช้งานได้โดยตรง โดยที่ 4 บิตของพอร์ตเอาต์พุตหรือพอร์ต Control นั้นสามารถดัดแปลงให้ใช้งานเป็นพอร์ตอินพุตขนาด 4 บิตได้ด้วยดังนั้นผู้ใช้งานจึงสามารถนำสัญญาณจากพอร์ตขนานที่มีมากถึง 17 เส้นไปใช้งานในการควบคุมโดยใช้ระดับสัญญาณ TTL

3.8 การเขียนโปรแกรมติดต่อกับพอร์ตขนาน

พอร์ตขนานของคอมพิวเตอร์จะมีลักษณะเช่นเดียวกับอุปกรณ์อินพุตเอาต์พุตตัวอื่น ๆ คือเมื่อต้องการติดต่อจะต้องกำหนดแอดเดรสที่ต้องการติดต่อด้วย ตารางที่ 2 – 1 แสดงแอดเดรสของพอร์ตขนาน โดยแบ่งออกเป็น 3 ตำแหน่งคือ แอดเดรสของรีจิสเตอร์ Data , รีจิสเตอร์ Status และรีจิสเตอร์ Control โดยแอดเดรสนี้จะมีอยู่ทั้งหมด 3 ชุด สำหรับพอร์ตขนาน 3 ชุดคือ LPT1 , LPT2 และ LPT3

เมื่อต้องการติดต่อกับพอร์ตขนานในตำแหน่งใด ก็ให้ส่งค่าข้อมูลออกไปที่พอร์ตขนานในตำแหน่งนั้น ๆ ยกตัวอย่างการเขียนโปรแกรมด้วย QBASIC เพื่อส่งค่าลอจิก “1” ออกไปทุกบิตของพอร์ต Data ของ LPT1 จะต้องเขียนโปรแกรมดังนี้

```
OUT &H378 , &HFF
```

โดยที่ เครื่องหมาย &H ที่แสดงนั้นหมายถึงตัวเลขฐานสิบหก

คำสั่ง OUT เป็นการส่งค่าข้อมูลออกเอาต์พุตของพอร์ตอินพุตเอาต์พุต

ค่า 378 เป็นแอดเดรสของรจิสเตอร์ Data สำหรับ LPT1

ค่าข้อมูล FF เป็นข้อมูลเลขฐานสิบหก ซึ่งหมายถึงการให้บิตทุกบิตของรีจิสเตอร์ Data มีลอจิกเป็น “1” นั่นเอง

ส่วนการอ่านค่าของพอร์ตขนานมายังคอมพิวเตอร์ผ่านทางพอร์ต Status ของ LPT1 สามารถเขียนโปรแกรมด้วย QBASIC ได้ดังนี้

ชื่อพอร์ต	LPT1		LPT2		LPT3	
	ฐานสิบ	ฐานสิบหก	ฐานสิบ	ฐานสิบหก	ฐานสิบ	ฐานสิบหก
DATA	888	378H	956	3BCH	632	278H
STATUS	889	379H	957	3BDH	633	279H
CONTROL	890	37AH	958	3BEH	634	27AH

ตารางที่ 3-3 แสดงแอดเดรสของพอร์ตขนาน

เทอร์โบซี

การส่งค่าข้อมูลออกไปยังพอร์ตขนาน

Outportb (0x378 , 0x££)

การอ่านค่าข้อมูลจากพอร์ตขนาน

Temp = inportb (0x379)

อย่างไรก็ตามโปรแกรมทุกตัวต่างก็ใช้วิธีการเดียวกันคือ กำหนดแอดเดรสที่จะทำการติดต่อ จากนั้นจึงติดต่อกับแอดเดรสเหล่านั้นด้วยคำสั่งสำหรับการอ่านหรือเขียน

3.9 การเขียนโปรแกรมติดต่อกับพอร์ตขนานด้วย Visual BASIC

การเขียนโปรแกรมด้วย Visual BASIC ชุดคำสั่งส่วนใหญ่จะมีรูปแบบใกล้เคียงกับ QBASIC แต่ Visual BASIC จะไม่มีคำสั่งสำหรับการติดต่อกับพอร์ตโดยตรงคือ คำสั่ง Inp () และคำสั่ง OUT เหมือนกับ QBASIC ดังนั้นเพื่อให้สามารถติดต่อกับพอร์ตขนานได้จึงจำเป็นต้องเพิ่มโปรแกรมบางตัวเข้าไป โดยโปรแกรมที่เพิ่มเข้าไปนี้จะอยู่ในรูปของ DLL (Dynamic Linked Library)

ไฟล์ DLL นี้ที่นำมาแนะนำในหนังสือเล่มนี้คือ io.dll โดยสามารถใช้ได้กับระบบปฏิบัติการที่เป็น 32 บิต ซึ่งก็คือวินโดวส์ 95 ขึ้นไป ซึ่งก็คือวินโดวส์ 95 / 98 / ME / 2000 /NT / XP

สำหรับตำแหน่งที่ใช้เก็บไฟล์ io.dll นั้นจะต้องเก็บไว้ที่ไดเรกตอรี่ SYSTEM ซึ่งอยู่ภายในไดเรกตอรี่ที่เก็บโปรแกรมวินโดวส์ โดยส่วนใหญ่จะมีชื่อเป็น Windows

การกำหนดค่าในโปรแกรมเพื่อเรียกใช้งานไฟล์ DLL มีรูปแบบการกำหนดค่าดังนี้

```
Public Declare Function Inp Lib "io.dll" _
Alias "PortIn" (ByVal PortAddress As Integer) As Byte
Public Declare Sub Out Lib "io.dll" _
```

Alias "PortOut" (ByBal PortAddress As Integer , ByVal Value As Byte)

เมื่อมาถึงตรงนี้ อาจเกิดคำถามขึ้นว่า ทำไมไมโครซอฟต์ผู้พัฒนาซอฟต์แวร์ Visual BASIC จึงไม่รวมคำสั่ง Inp และคำสั่ง Out ไว้ในโปรแกรม Visual BASIC เนื่องจากว่าการเขียนและอ่านข้อมูลไปยังพอร์ตหรือหน่วยความจำโดยตรงนั้นอาจทำให้เกิดมีปัญหาแฮงก์หรือทำงานผิดพลาดได้ และ Visual BASIC เป็นระบบปฏิบัติการที่ทำงานบนวินโดวส์ ซึ่งมีการทำงานแบบมัลติทาร์สกิ้ง (multitasking) มีโปรแกรมหลาย ๆ ตัวทำงานอยู่พร้อมกัน ดังนั้นเมื่อเกิดความเสียหายกับโปรแกรมตัวหนึ่งก็อาจจะส่งผลให้โปรแกรมที่ทำงานอยู่ทั้งหมดเสียหายได้ นอกจากนี้การเขียนข้อมูลไปยังพอร์ตเช่นเดียวกัน ส่งผลให้โปรแกรมทำงานผิดพลาด

สำหรับระบบปฏิบัติการวินโดวส์ 95 ขึ้นไป นอกจากจะสามารถใช้งาน DLL ในการติดต่อกับพอร์ตโดยตรงแล้ว ยังสามารถใช้งานโปรแกรมประเภท Visual Device Driver (Vxd) ในการติดต่อกับอุปกรณ์อินพุตเอาต์พุต โดย Vxd จะดัดปัญหาเรื่องการเข้าถึงพอร์ตพร้อมกันของโปรแกรมหลาย ๆ ตัวได้ แต่สำหรับโปรแกรมสั้น ๆ เช่น โปรแกรมอ่านค่าอุณหภูมิ โปรแกรมควบคุมอุปกรณ์อินพุตเอาต์พุตปกติซึ่งไม่มีการติดต่อกับพอร์ตอยู่ตลอดเวลา คำสั่ง Inp และ Out ใน DLL ก็ยังทำงานได้ดีและมีรูปแบบการใช้งานที่ง่ายกว่า

รายละเอียดเพิ่มเติมเกี่ยวกับ io.dll

io.dll พัฒนาขึ้นโดยโปรแกรมเมอร์นิรนามที่ใช้ชื่อว่า Fred ซึ่งมีความเชี่ยวชาญด้านการพัฒนาระบบการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก โดยไฟล์ io.dll นี้ผู้เขียนโปรแกรมเผยแพร่โดยไม่คิดมูลค่าซึ่งผู้สนใจสามารถดาวน์โหลดเวอร์ชั่นใหม่ที่อาจมีได้ที่

http://www.geekhideout.com

ไฟล์ dll ตัวนี้มีประโยชน์อย่างมากสำหรับนักพัฒนาที่ต้องการติดต่อกับพอร์ตอินพุตเอาต์พุตของคอมพิวเตอร์ โดยเฉพาะอย่างยิ่งกับพอร์ตขนานในระบบปฏิบัติการวินโดวส์ ซึ่งในปัจจุบันการติดต่อพอร์ตกระทำได้ยากขึ้น แต่ด้วยการใช้ไฟล์ dll ตัวนี้จะช่วยให้สามารถเข้าถึงและควบคุมได้ง่ายและทรงประสิทธิภาพ

Io.dll สามารถใช้งานได้กับระบบปฏิบัติการวินโดวส์ 95 / 98 / 2000 / NT หรือกระทั่ง XP จึงช่วยลดเวลาและขั้นตอนในการพัฒนาโปรแกรมติดต่อพอร์ตที่รันบนวินโดวส์ได้อย่างมาก

ฟังก์ชั่นที่มีอยู่ใน io.dll

PortOut	ใช้ส่งข้อมูลขนาด 1 ไบต์ไปยังพอร์ตที่กำหนด
PortWordOut	ใช้ส่งข้อมูล 1 เวิร์ด (16 บิต) ไปยังพอร์ตที่กำหนด

PortDWordOut ใช้ส่งข้อมูลดับเบิลเวิร์ด (32 บิต) ไปยังพอร์ตที่กำหนด

PortIn ใช้อ่านข้อมูลขนาด 1 ไบต์จากพอร์ตที่กำหนด

PortWordIn ใช้อ่านข้อมูล 1 เวิร์ด (16 บิต) จากพอร์ตที่กำหนด

PortDWordIn ใช้อ่านข้อมูลดับเบิลเวิร์ด (32 บิต) จากพอร์ตที่กำหนด

SetPortBit ใช้เซตข้อมูลในระดับบิตของพอร์ตที่กำหนด

ClrPortBit ใช้เคลียร์ข้อมูลในระดับบิตของพอร์ตที่กำหนด

NotPortBit ใช้กลับข้อมูลในระดับบิตของพอร์ตที่กำหนด

GetPortBit ใช้อ่านสถานะของบิตที่กำหนด

RightPortShift ใช้เลื่อนข้อมูลของพอร์ตไปทางขวา จะได้ค่าของบิต LSB กลับมา

LeftPortShift ใช้เลื่อนข้อมูลของพอร์ตไปทางซ้าย จะได้ค่าของบิต MSB กลับมา

ISDriverInstalled ใช้ตรวจสอบการติดตั้งไฟล์ il.dll โดยถ้าหากมีการติดตั้งไฟล์จะได้ค่า "1" หรือ "non-zero" กลับมา

ในบทต่อไปจะเริ่มต้นการใช้งานคอมพิวเตอร์ในการเชื่อมต่อกับอุปกรณ์ภายนอกผ่านทางพอร์ตขนาน โดยใช้ชุดทดลอง NX-2000 ซึ่งประกอบด้วยบอร์ดทดลองย่อยมากมาย โดยจะทยอยกล่าวถึงเมื่อมีการเรียนรู้และทดลอง ส่วนโปรแกรมที่ใช้อ้างอิงในการทดลองนั้น ผู้เรียนต้องเขียนขึ้นเอง โดยใช้โปรแกรม Visual BASIC โดยในแต่ละการทดลองได้เตรียมโปรแกรมตัวอย่างไว้ให้เพื่อให้ผู้เรียนได้ศึกษาและทำการทดลองในขั้นต้นก่อน จากนั้นจึงเริ่มเขียนโปรแกรมด้วยตนเองจากง่ายไปจนถึงโปรแกรมที่มีความซับซ้อนเพิ่มมากขึ้น นอกจากนั้น ผู้เรียนยังได้รับข้อมูลความรู้เกี่ยวกับอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในแต่ละการทดลองเพิ่มเติมในด้วยในคราวเดียวกัน

บทที่ 4

โปรแกรมวิชวลเบสิก 6

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

4.1 ขั้นตอนการสร้างโปรแกรมประยุกต์

การสร้างโปรแกรมประยุกต์วิชวลเบสิก ประกอบด้วยขั้นตอนหลัก 3 ขั้นตอน คือ

1. การสร้างอินเตอร์เฟส (Interface) โดยมีฟอร์ม (Form) เป็นอ๊อบเจค (Object) พื้นฐานและเป็นที่วางตัวคอนโทรล (Control) สำหรับการติดต่อกับผู้ใช้
2. ตั้งค่าคุณสมบัติ เป็นการกำหนดพฤติกรรมและการทำงานให้กับอ๊อบเจคต่าง ๆ
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	Check Box	ใช้กับการเลือกแบบ ถูก/ผิด(True/False, Yes/No)
	Combo box	Combo Box	เป็นตัว Control เป็นการผสมระหว่าง Text box กับ List box ซึ่งจะปรากฏรายการเมื่อมีการคลิกลูกศรและ Combo box ไม่สนับสนุนการเลือกแบบหลายค่า
	Command button	Command Button	ปุ่มคำสั่งเป็นตัว Control ที่ใช้ในทุกฟอร์ม ตามปกติจะเขียนคำส่งใน Click Event Procedure ของตัว Control นี้

	Data	Data	เป็นตัว Control ที่สามารถรวมข้อมูลกับฐานข้อมูลได้และเป็นส่วนที่ Visual Basic ให้ผู้ใช้สามารถติดต่อระหว่างตัว Control บนฟอร์มกับฟิลด์ใน Table ของฐานข้อมูล โดย Data จะทำงานกับ Database Jet ของฐานข้อมูล แต่ไม่สามารถทำงานกับ ActiveX Data Object (ADO) ได้
	Directory List box	Dir List Box	เป็น List box แบบหนึ่ง ที่แสดงไดเรคทอรีและพาร์ทที่เลือก
	Drive List box	Drive List Box	คล้ายกับ Combo box ที่ใช้เลือกชื่อของไดร์ในระบบ
	File list box	File List Box	เป็น List box ชนิดพิเศษที่ใช้แสดงชื่อไฟล์ในไดเรคทอรี
	Frame	Frame	สามารถใช้เป็น Container สำหรับตัว Control อื่น
	Horizontal and Vertical Scroll Bar	HScrollBar and VScrollBar	ใช้เป็นแถบเลื่อนแบบ Stand-alone แต่มักจะไม่ค่อยมีการใช้ เพราะตัว Control อื่น ๆ ส่วนใหญ่จะมีแถบเลื่อนของตัวเอง แถบเลื่อนแบบ Stand-alone อาจจะใช้ในลักษณะ Slider ได้
	Image	Image	เป็นตัว Control ใช้เก็บภาพคล้ายกับ Picture Box แต่ไม่สามารถทำงานแบบ Container ได้ Image มีข้อดีที่ใช้ทรัพยากรของระบบน้อยกว่า Picture Box
	Label	Label	เป็นตัว Control ที่ใช้แสดงข้อความหรือป้ายชื่อ
	Line	Line	เป็นตัว Control ใช้สำหรับการตกแต่งด้านกราฟฟิก
	List box	List Box	เป็นตัว Control ที่เก็บรายการของค่าและให้ผู้ใช้เลือก ซึ่งสามารถเป็นการเลือกค่าเดียวหรือหลายค่าขึ้นกับการกำหนดคุณสมบัติ Multi Select
	OLE container	OLE	เป็นตัว Control ที่สามารถเป็น Host Window

			ให้กับโปรแกรมภายนอก เช่น Microsoft Excel หรืออาจจะกล่าวว่าเป็นการสร้าง Window ให้กับโปรแกรมอื่นบนโปรแกรมประยุกต์ Visual Basic
	Option button	Option Button	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	Picture Box	ใช้แสดงภาพในฟอร์แมต BMP, DIB (Bitmap), ไอคอน (Icon), WMF (Metafile), GIF และ JPEG เป็นต้น และสามารถใช้เป็น Container สำหรับตัว Control อื่น
	Shape	Shape	เป็นตัว Control ใช้สำหรับการตกแต่งด้านกราฟฟิก
	Text box	Text Box	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 Tdwips

```
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 , เลขฐานสิบหก 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 เป็นข้อความที่ปรากฏภายในตัวคอนโทรลและไม่สามารถแก้ไขได้โดยผู้ใช้เมื่อเรียกใช้โปรแกรม เช่น

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

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

6. MousePointer และ MouseIcon ใช้กำหนดไอคอนของเมาส์เมื่อเคลื่อนที่ผ่านตัวคอนโทรล เช่น

```
Text1.MousePointer = vbCrosshair
```

7. คุณสมบัติอื่น ๆ ได้แก่

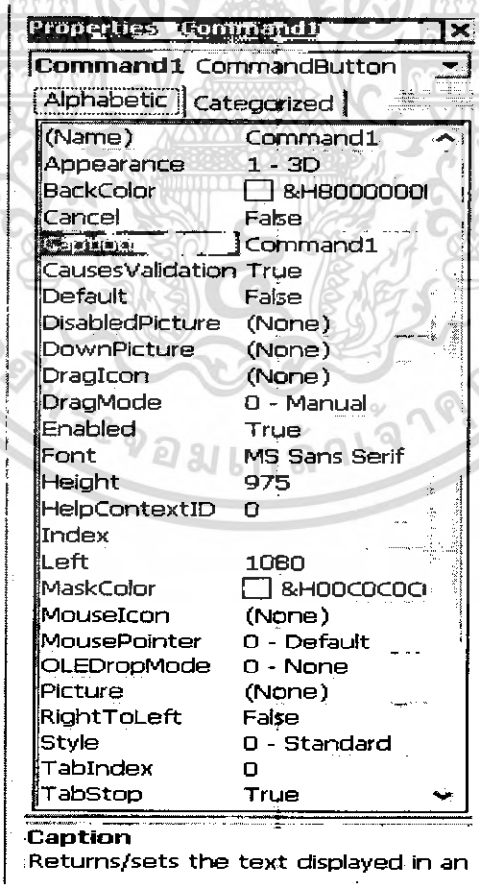
- คุณสมบัติ Value มีค่าตามชนิดของตัวคอนโทรล

- 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 เป็นหน้าต่างสีเหลืองขนาดเล็กแสดงคำอธิบายสั้น ๆ ซึ่งปรากฏขึ้นเมื่อเมาส์เคลื่อนที่ไปอยู่เหนือตัว control หรือไอคอน



ตารางที่ 4-2 แสดงถึงการกำหนดคุณสมบัติของคอนโทรลที่ใช้ในการสร้างโปรแกรม

4.4 เมธอดร่วม

1. Move ใช้ในการย้ายอ๊อบเจค หรือตัวคอนโทรล พร้อมทั้งปรับขนาดในเวลาเดียวกัน มีไวยากรณ์คือ

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

2. Refresh ใช้ในการวาดตัวคอนโทรลใหม่ ตามปกติไม่ใช้เนื่องจาก Visual Basic มีการปรับค่าโดยอัตโนมัติแต่สามารถใช้ได้ในกรณีที่ต้องการให้มีการปรับคุณสมบัติทันที 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 เป็นอีเวนต์พื้นฐานที่เกิดเมื่อข้อมูลของตัวคอนโทรลมีการเปลี่ยนแปลง

3. GotFocus และ LostFocus Event ซึ่ง GotFocus Event เกิดขึ้นเมื่อตัวคอนโทรลได้รับการโฟกัส 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 เป็นอีเวนต์ที่เกิดขึ้นเมื่อมีการคลิก, 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 ใช้ในการกำหนดค่าเริ่มต้นให้กับฟอร์มและอ๊อบเจค

2. Resize Event เกิดขึ้นก่อนที่จะเห็นฟอร์ม โดยอีเวนต์สามารถใช้ในการปรับตัวคอนโทรลให้พอดีกับฟอร์ม 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 จะทำงานเมื่อเกิดเหตุการณ์ต่าง ขึ้นกับอ๊อบเจค เช่น เมื่อผู้ใช้คลิกปุ่ม "Start" ถ้าหากภายในปุ่มมีโพรซีเจอร์อยู่ ก็จะเกิดการทำงานตามคำสั่งในโพรซีเจอร์นั้นทันที

2. โพรซีเจอร์ที่กำหนดเอง เป็นโพรซีเจอร์แบบทั่วไป หลังจากสร้างขึ้นแล้วเราสามารถเรียกใช้ได้ทันที เพียงแค่อ้างชื่อของโพรซีเจอร์นั้นเท่านั้น

โพรซีเจอร์และฟังก์ชั่น

ลักษณะของฟังก์ชั่นจะคล้ายกับโพรซีเจอร์ เพียงแต่ฟังก์ชั่นจะมีการส่งค่ากลับไปยังโปรแกรมที่เรียกฟังก์ชั่นนั้นมาใช้

4.8 การประกาศตัวแปร

การประกาศตัวแปรมีคำสั่งแตกต่างกันไปตามชนิดของการใช้งานดังนี้

- Static: ตัวแปรนี้จะใช้ได้เฉพาะภายในโพรซีเจอร์เท่านั้นและค่าในตัวแปรจะถูกเก็บไว้ตลอดเวลา แม้ว่าจะออกโพรซีเจอร์ไปแล้วก็ตาม

- Dim : มีลักษณะเป็น Static แต่สามารถประกาศตัวแปรได้ 2 ที่ คือ ในโพซีเจอร์ และ General Declarations (ส่วนแรกของโปรแกรมใช้ในการประกาศตัวแปร) หากประกาศในโพรซีเจอร์แล้ว เมื่อออกจากโพรซีเจอร์แล้ว ค่าในตัวแปรจะถูกรีเซ็ทใหม่อัตโนมัติ ดังนั้นในแต่ละโพรซี

เจอร์อาจมีตัวแปรชื่อซ้ำกันได้ โดยไม่มีผลต่อกัน แต่การประกาศตัวแปรบน General Declarations จะทำให้โพรซีเจอร์ทุกตัวบนฟอร์มนั้นสามารถเรียกใช้ได้ และหากมีการเปลี่ยนแปลงค่าในโพรซีเจอร์ใด ๆ แล้ว ก็จะมีผลไปกับโพรซีเจอร์อื่น ๆ ด้วย

- Private: มีรูปแบบเช่นเดียวกันกับ Dim แต่ประกาศได้ที่ General Declarations เท่านั้น และจะใช้ได้เฉพาะบนฟอร์มนั้นเท่านั้น

- Public: มีรูปแบบเช่นเดียวกันกับ Private แต่สามารถใช้ได้ในทุก ๆ ฟอร์มบนโปรแกรมนั้น

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 และเวลาใด ๆ Date ใช้ 8 ไบต์ เหมือนกับ Double แต่โครงสร้างมีความแตกต่างกัน โดยส่วนจำนวนเต็มเป็นสารสนเทศของวันและ

			ทศนิยมเป็นส่วนของเวลา

ตารางที่ 6-3 แสดงถึงประเภทของข้อมูลพื้นฐานที่ใช้ในการสร้างตัวแปร

4.9 คำสั่งพื้นฐาน

คำสั่งพื้นฐานที่ใช้ในการสร้างโปรแกรม ได้แก่

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

```
If
  < เงื่อนไขที่ทำการเปรียบเทียบ >
Then
  < ถ้าเงื่อนไขเป็นจริงเขียนโค้ดที่นี่ >
Else
  < ถ้าเงื่อนไขไม่เป็นจริง เขียนโค้ดที่นี่ >
End If
```

- คำสั่งเงื่อนไข Select Case มีรูปแบบดังนี้

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

- คำสั่งวนรอบแบบ for...Next มีรูปแบบดังนี้ คือ

For

< ชื่อตัวแปรประเภทตัวเลข > = < ค่าเริ่มต้น > to < ค่าสิ้นสุด >

<โปรแกรมที่ต้องการให้คำงานวนรอบ >

Next < ชื่อตัวแปร >

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

Do While/Until

< เงื่อนไขการเปรียบเทียบ >

< คำสั่งที่ต้องการให้ทำงานวนรอบ >

Loop

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

Do

< คำสั่งที่ต้องการให้ทำงานวนรอบ >

Loop While/Until

< เงื่อนไขการเปรียบเทียบ >

บทที่ 5
ความรู้เบื้องต้นของพอร์ตขนาน

- รู้จักกับพอร์ตขนาน (Parallel Port/printer Port)
- ลักษณะสัญญาณของพอร์ตขนาน
- รูปแบบการตัดต่อข้อมูลผ่านพอร์ตขนาน
- เหตุผลในการเลือกใช้งานพอร์ตขนาน

ความรู้เบื้องต้นของพอร์ตขนาน

ในการประยุกต์ใช้งานอุปกรณ์เชื่อมต่อ (Interface) นั้นเรามีการใช้งานพอร์ตขนาน หรือ Parallel Port อย่างแพร่หลาย โดยจุดเริ่มต้นคือการนำมาใช้งานกับพริ้นเตอร์จนทำให้บางคนเรียกว่า Printer Port ก็มี ซึ่งพอร์ตขนานนั้นมีความสามารถ และความน่าสนใจอยู่หลายอย่างจึงจำจะเข้าใจพื้นฐานกันก่อนเพื่อให้การสร้างแอพพลิเคชันในภายหลังทำได้อย่างเข้าใจ และรวดเร็ว

5.1 รู้จักกับพอร์ตขนาน

พอร์ตขนาน หรือ parallel Port นั้นเดิมเรียกว่า Printer Port เพราะการใช้งานส่วนใหญ่กับพอร์ตขนานเป็นการใช้งานโดยต่อกับพริ้นเตอร์เป็นหลัก โดยที่พอร์ตขนานนั้นสามารถให้ความเร็วในการส่งผ่านข้อมูลได้รวดเร็วกว่าพอร์ตอนุกรมราว 8 – 10 เท่า ซึ่งสามารถส่งข้อมูลขนาน 8 บิตได้เลย

ลักษณะหัวต่อของพอร์ตขนานจะเป็นแบบ D-type 25 pin ตัวเมียอยู่หลังเครื่องคอมพิวเตอร์ทั่วไป ปกติแล้วจะใช้ในการติดต่อกับเครื่องพริ้นเตอร์

อย่างไรก็ตามนอกจากพอร์ตขนานจะใช้ติดต่อกับเครื่องพริ้นเตอร์แล้ว ยังสามารถใช้ในงานเชื่อมต่อกับอุปกรณ์ชนิดอื่น ๆ ได้อีก ซึ่งการใช้พอร์ตขนานในงานในการเชื่อมต่อนั้นถูกใช้งานกันอย่างแพร่หลายมาก ทั้งนี้เพราะสามารถรับและส่งข้อมูลในลักษณะขนานได้ ทำให้นำไปใช้ในการควบคุมอุปกรณ์ภายนอกได้เป็นอย่างดี อีกทั้งลักษณะแรงดันที่จ่ายออกมาก็เป็น TTL โดยสัญญาณลอจิก “1” จะเท่ากับ 5 โวลต์และลอจิก “0” จะเท่ากับศูนย์โวลต์ทำให้ง่ายในการออกแบบวงจรและการประยุกต์ใช้งาน

ข้อเสียก็คือไม่สามารถทำงานในระยะทางไกล ๆ ได้เพราะจะเกิดความผิดพลาดของข้อมูลได้ง่าย เนื่องจากแรงดันไม่สม่ำเสมอ และสิ้นเปลืองค่าใช้จ่ายในเรื่องของสายเพราะต้องใช้สายสัญญาณหลายเส้น

พอร์ตขนานของเครื่องคอมพิวเตอร์ประกอบด้วยสัญญาณทั้งหมด 25 เส้นสัญญาณ แต่ใช้งานกันจริง ๆ 17 เส้นสัญญาณ โดยสัญญาณจะแบ่งออกได้เป็น 3 กลุ่มใหญ่ ๆ ตามลักษณะหน้าที่ของสัญญาณ ประกอบด้วย

- Data Port จำนวน 8 เส้นสัญญาณ
- Status Port จำนวน 5 เส้นสัญญาณ
- Control Port จำนวน 4 เส้นสัญญาณ

5.2 ลักษณะของสัญญาณ

ขาสัญญาณของพอร์ตขนานแบ่งได้เป็น 3 กลุ่มดังนี้

DATA PORT

Data Port จะมีอยู่ 8 ขา หรือ 8 pin (ตั้งแต่ขาที่ 2 ถึงขาที่ 9) บางทีมักจะถูกเรียกว่า DATA REGISTER ซึ่ง Register ตัวนี้จะส่งค่าได้อย่างเดียวไม่สามารถรับค่าได้

Name	Read / Write	Bit No.	Signal Name
Data Port	Write	Bit 7	Data 7 (pin 9)
		Bit 6	Data 6 (pin 8)
		Bit 5	Data 5 (pin 7)
		Bit 4	Data 4 (pin 6)
		Bit 3	Data 3 (pin 5)
		Bit 2	Data 2 (pin 4)
		Bit 1	Data 1 (pin 3)
		Bit 0	Data 0 (pin 2)

STATUS PORT

Status Port เป็นพอร์ตที่อ่านได้อย่างเดียวไม่สามารถเขียนข้อมูลได้ พอร์ตนี้จะมีสัญญาณเข้าอยู่ 5 สัญญาณ และสัญญาณ IRQ กับสัญญาณสงวนไว้อีกสองบิต โดยสัญญาณ Busy จะ Active Low

Name	Read / Write	Bit No.	Signal Name
Status Port	Read	Bit 7	Busy
		Bit 6	nAck
		Bit 5	PaperEnd
		Bit 4	Select
		Bit 3	nError
		Bit 2	IRQ (Not)
		Bit 1	Reserved
		Bit 0	Reserved

สำหรับลักษณะการทำงานของแต่ละบิตใน Status Port

- Bit 7 Busy เมื่อ Active หมายถึงพริ้นเตอร์จะไม่รับข้อมูล
- Bit 6 nAck เมื่อ Active หมายถึงพริ้นเตอร์พร้อมที่จะทำงาน (Active Low)
- Bit 5 Paper End เมื่อ Active หมายถึงพริ้นเตอร์ไม่มีกระดาษ
- Bit 4 Select เมื่อ Active หมายถึงเลือกพริ้นเตอร์
- Bit 3 nError เมื่อ Active หมายถึงพริ้นเตอร์เกิดข้อผิดพลาด (Active Low)
- Bit 2 , Bit 1 , Bit 0 ไม่ใช้

5.3 CONTROL PORT

Control Port เป็นพอร์ตที่ใช้ในการควบคุมพรินเตอร์ สัญญาณในกลุ่มนี้จะ Active Low ยกเว้น สัญญาณ Intialize เท่านั้นที่ไม่ถูก Invert

Name	Read / Write	Bit No.	Signal Name
Control Port	Read / Write	Bit 3	nESelect (pin 17)
		Bit 2	nInitialize (pin 16)
		Bit 1	nAutoFeed (pin 14)
		Bit 0	nStrobe (pin 1)

ลักษณะการทำงานของแต่ละบิตใน Control Port

- Bit 3 nSelect Printer เมื่อ Active หมายถึงเลือกพริ้นเตอร์
- Bit 2 nInitialize เมื่อ Active หมายถึงรีเซตพริ้นเตอร์
- Bit 1 nAuto Feed เมื่อ Active หมายถึงพริ้นเตอร์กระทำ Line Feed
- Bit 0 nStrobe เมื่อ Activeหมายถึงการบอกให้พริ้นเตอร์ทราบว่ามีข้อมูลเข้ามาแล้ว

สรุปลักษณะสัญญาณของพอร์ตขนานทั้งหมด

สำหรับลักษณะของสัญญาณของพอร์ตขนานทั้งหมดมีรายละเอียดดังตาราง

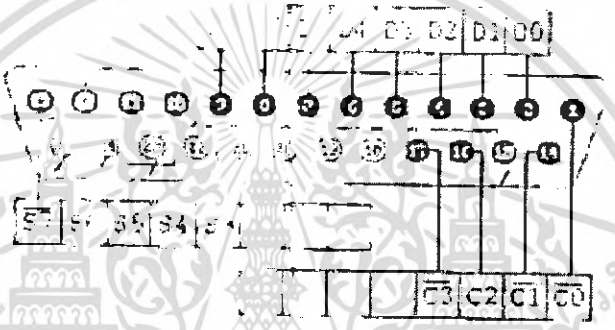
Pin No. (D-Type 25)	Signal Name	Bit	Direction (In / Out)
1	nStrobe	-C0	Output
2	Data 0 (Bit 0)	D0	Output
3	Data 0 (Bit 0)	D1	Output
4	Data 0 (Bit 0)	D2	Output
5	Data 0 (Bit 0)	D3	Output
6	Data 0 (Bit 0)	D4	Output
7	Data 0 (Bit 0)	D5	Output
8	Data 6 (Bit 6)	D6	Output
9	Data 7 (Bit 7)	D7	Output
10	nAcK	S6	Input
11	Busy	-S7	Input
12	PaperEnd	S5	Input
13	Select	S4	Input
14	nAutoFeed	-C1	Output
15	nError	S3	Input
16	nInitialize	C2	Output

17	nSelectPrinter	-C3	Output
18 - 25	Ground		

Pin Outs

สำหรับการต่ออุปกรณ์ภายนอกเข้ากับพอร์ตขนานที่ขาต่าง ๆ จะมีทิศทางของการรับส่งข้อมูล หรือ Pin Out ของพอร์ตขนานดังนี้

ส่วนหน้าตาของพอร์ตจริง ๆ มีรายละเอียดดังนี้



รูปที่ 5- 1 แผนผังของสายสัญญาณในพอร์ตขนาน

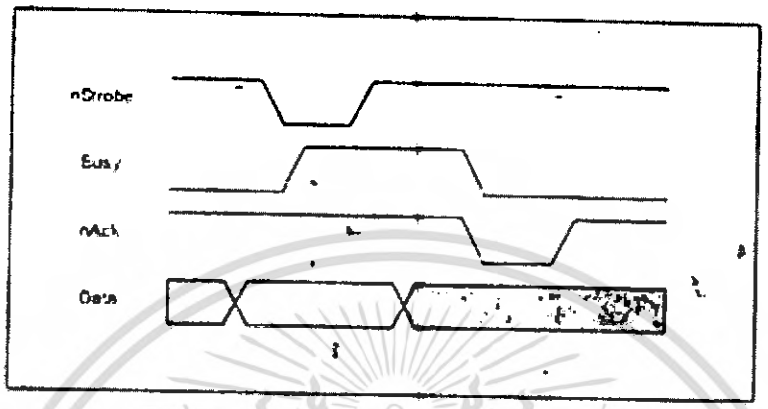
- 8 Output pins DATA PORT
- 5 Input pins (1 invert)
- 4 Output pins (3 invert) CONTROL PORT
- 8 pins Ground

5.4 รูปแบบการติดต่อผ่านทางพอร์ตขนาน

เมื่อมีการติดต่อกับอุปกรณ์ หรือพริ้นเตอร์ใด ๆ ผ่านทางพอร์ตขนานนั้นการทำงานจะเริ่มจากคอมพิวเตอร์จะส่งสัญญาณข้อมูลขนาด 8 บิต ออกทาง Data Port แล้วสร้างสัญญาณ Strobe ให้เป็น Low ส่งไปยังอุปกรณ์ หรือพรินเตอร์เพื่อบอกให้ทราบว่ามีข้อมูลพร้อมเตรียมจะส่งให้แล้ว

จากนั้นคอมพิวเตอร์จะรอรับการตอบกลับจากอุปกรณ์ หรือพริ้นเตอร์ที่ต่อกับพอร์ตขนานนั้น โดยสิ่งที่ตอบกลับมาจะมีอยู่ 2 ลักษณะคือ

- สัญญาณ nAck เพื่อเป็นการแสดงว่าอุปกรณ์ หรือพริ้นเตอร์พร้อมที่จะรับสัญญาณข้อมูล โดยจะสร้างสัญญาณ Acknowledge เป็น Low
- สัญญาณ Busy เพื่อเป็นการแสดงว่าอุปกรณ์ หรือพริ้นเตอร์ไม่ว่าง ไม่พร้อมรับข้อมูล



รูปที่ 5–2 ลักษณะสัญญาณ

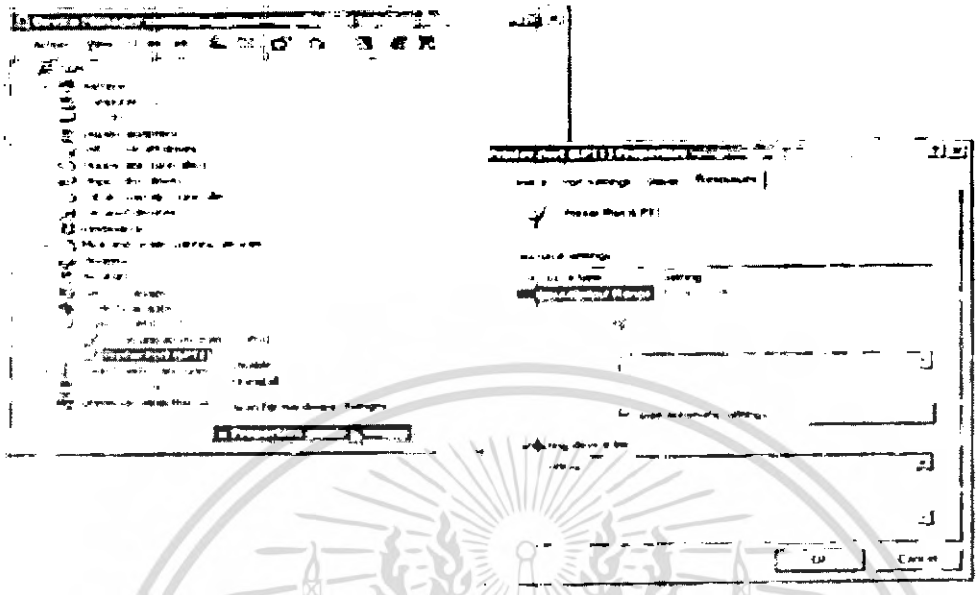
จากรูปข้างต้นเราสามารถสรุปรายละเอียดเกี่ยวกับสัญญาณที่ใช้ในพอร์ตขนานได้ดังตาราง

สัญญาณ	ผู้ส่ง	สิ่งที่ส่ง	ผู้รับ
ข้อมูล 8 บิต	คอมพิวเตอร์	ข้อมูลที่คอมพิวเตอร์ส่งไปให้ที่พอร์ต	พอร์ตขนาน
Strobe	คอมพิวเตอร์	แจ้งให้พอร์ตทราบว่ามีข้อมูลชุดใหม่ส่งมา	พอร์ตขนาน
Acknowlege	พอร์ตขนาน	ตอบกลับมาว่าพร้อมรับข้อมูลแล้ว	คอมพิวเตอร์
Busy	พอร์ตขนาน	ตอบกลับมาว่าไม่พร้อม	คอมพิวเตอร์
Error	พอร์ตขนาน	แจ้งข้อผิดพลาดกลับมาให้คอมพิวเตอร์	คอมพิวเตอร์
Reset	คอมพิวเตอร์	ให้รีเซตข้อมูล หรือรีเซตพรินเตอร์	พอร์ตขนาน

Port Address

แอดเดรสของพอร์ตขนานนั้นมักจะอยู่ที่ตำแหน่ง 378H สำหรับ LPT1 และ 278H สำหรับ LPT2 ซึ่งการหาหมายเลขพอร์ตของพริ้นเตอร์นั้นเราสามารถกระทำได้ง่าย ๆ โดยการคลิกที่ Start > Settings > Control Panel

จากนั้นให้เลือกที่ System > Device Manager > Ports(COM & LPT) จากนั้นให้เลือก Printer Port แล้วคลิกที่ Properties แล้วเลือก Resources เราก็จะทราบ Address ของหมายเลขพอร์ตตามที่เราต้องการ



รูปที่ 5-3 วิธีตรวจสอบ Address ของพอร์ตขนาน

5.5 ทำไมจึงเลือกใช้งานพอร์ตขนาน

หลายคนอาจสงสัยว่าพอร์ตขนานนั้นน่าจะใช้งานเฉพาะกับพริ้นเตอร์ ทำไมจึงนำมาแนะนำ หรือมีการประยุกต์ใช้งานกันหลากหลาย ซึ่งก็มีเหตุผลหลาย ๆ อย่างที่ทำให้มีการเลือกใช้งานพอร์ตขนาน

1. ความเข้ากันได้กับคอมพิวเตอร์ เพราะพอร์ตขนานนั้นใช้งานได้กับพีซีแทบทุกรุ่น ซึ่งพอร์ตขนานนั้นถือเป็นพอร์ตมาตรฐาน ใครจะนำมาประยุกต์ใช้ในงานอื่น ๆ ก็ได้ ไม่จำเป็นต้องเป็นพรินเตอร์อย่างเดียว ทำให้เราไม่ต้องกังวลว่าสร้างวงจรแล้วจะไม่สามารถใช้งานร่วมกันได้

2. ความประหยัด การที่ใช้งานพอร์ตขนานจะช่วยลดความจำเป็นในการที่ต้องการการ์ดเข้าไปเสียบในเมนบอร์ดของพีซี ซึ่งถือว่ามีจำนวนช่องเสียบ หรือสล็อตที่ค่อนข้างจำกัด ทำให้การนำไปใช้งานประหยัดทั้งเนื้อที่ และยืดหยุ่นมากเมื่อต้องใช้กับเครื่องอื่น ๆ อีก ทั้งปัจจุบันอุปกรณ์ที่สนับสนุนชิ้นส่วนอะไหล่เกี่ยวกับพอร์ตขนานก็มีมากทำให้หาซื้อได้ง่ายมาก

3. ความปลอดภัย การที่สร้างอุปกรณ์แล้วเชื่อมต่อด้วยพอร์ตขนานนั้นช่วยให้เราไม่ต้องแกะเคสเพื่อเสียบการ์ด ซึ่งจะช่วยลดความเสียหายระว่างการทำงาน การสึกหรอของช่องเสียบเนื่องจากต้องเสียบเข้าถอดออกการ์ดบ่อย ๆ

4. งานที่ต้องการช่องสัญญาณกว้าง และเร็ว การใช้งานพอร์ตขนานทำให้เราส่งข้อมูลแบบขนานซึ่งทำให้ส่งข้อมูลได้พร้อม ๆ กัน การทำงานจึงรวดเร็วกว่าพอร์ตอนุกรม (แต่อาจให้ระยะทำการที่สั้นกว่าพอร์ตอนุกรม)

การเขียนโปรแกรมเพื่อส่งข้อมูลออกทางพอร์ตขนาน

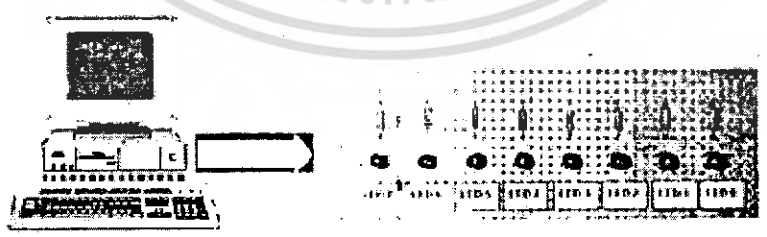
- รูปแบบการเชื่อมต่อบอร์ดทดลองกับพีซี
- เขียนโปรแกรมควบคุมหลอด LED
- สร้างโปรแกรมไฟวิ่งด้วย Visual Basic
- สร้างไฟวิ่งแบบ Advance

5.6 การเขียนโปรแกรมเพื่อส่งข้อมูลออกทางพอร์ตขนาน

ในบทนี้จะเริ่มเข้าสู่การเขียนโปรแกรมเพื่อติดต่อกับบอร์ดทดลองโดยการส่งข้อมูลออกไปผ่านพอร์ตขนาน ซึ่งในบทนี้เราจะเขียนโปรแกรมติดต่อพอร์ตขนานของเครื่องคอมพิวเตอร์ เพื่อใช้ในการควบคุมการทำงานของหลอดแสดงผล (LED) ให้แสดงผลในลักษณะติด หรือดับตามรูปแบบที่เราต้องการ ซึ่งบทนี้จะเป็นพื้นฐานสำคัญมากสำหรับการควบคุมฮาร์ดแวร์ชนิดอื่น ๆ ผ่านพอร์ตขนาน

5.7 รูปแบบการเชื่อมต่อบอร์ดกับพีซี

ในการนำบอร์ดทดลองที่ได้มาทดสอบการทำงานร่วมกับพีซีของเรานั้นมีรูปแบบการเชื่อมต่อ (Interface) ดังนี้



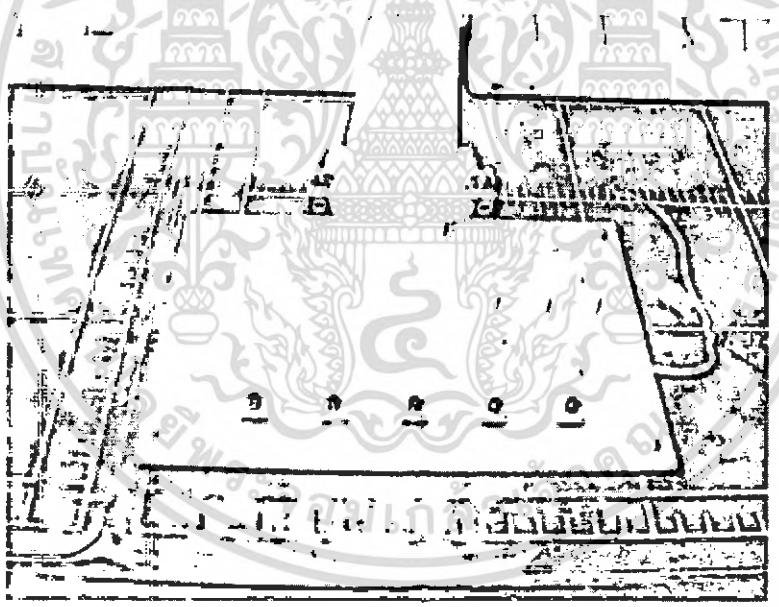
รูปที่ 5-4 รูปแบบของการเชื่อมต่อบอร์ดทดลองในการส่งข้อมูล

จากรูปเราจะนำบอร์ดทดลองที่ได้จากการสร้างตามวิธีการในบทที่แล้วมาเชื่อมต่อกับพีซี ซึ่งจะต้องมีอุปกรณ์สำคัญก็คือ สายเชื่อมพอร์ตขนาน จากนั้นเราจะทดลองเขียนโปรแกรมด้วย Visual Basic แล้วสั่งงานการแสดงผล LED ที่บอร์ดทดลอง

ในการเขียนโปรแกรมจะใช้โพรซีเจอร์ Out จากไฟล์ inpout32.dll โดยมีรูปแบบการเรียกใช้งานดังนี้

```
Out port number , data
```

โพรซีเจอร์ Out จะใช้สำหรับส่งข้อมูลออกทางพอร์ตขนาน โดยที่ค่าของ portnumber จะเท่ากับ H378 สำหรับ LPT1 และ H278 สำหรับ LPT2 ส่วน data นั้นสามารถกำหนดค่าให้มีค่าอยู่ระหว่าง 0 ถึง 255



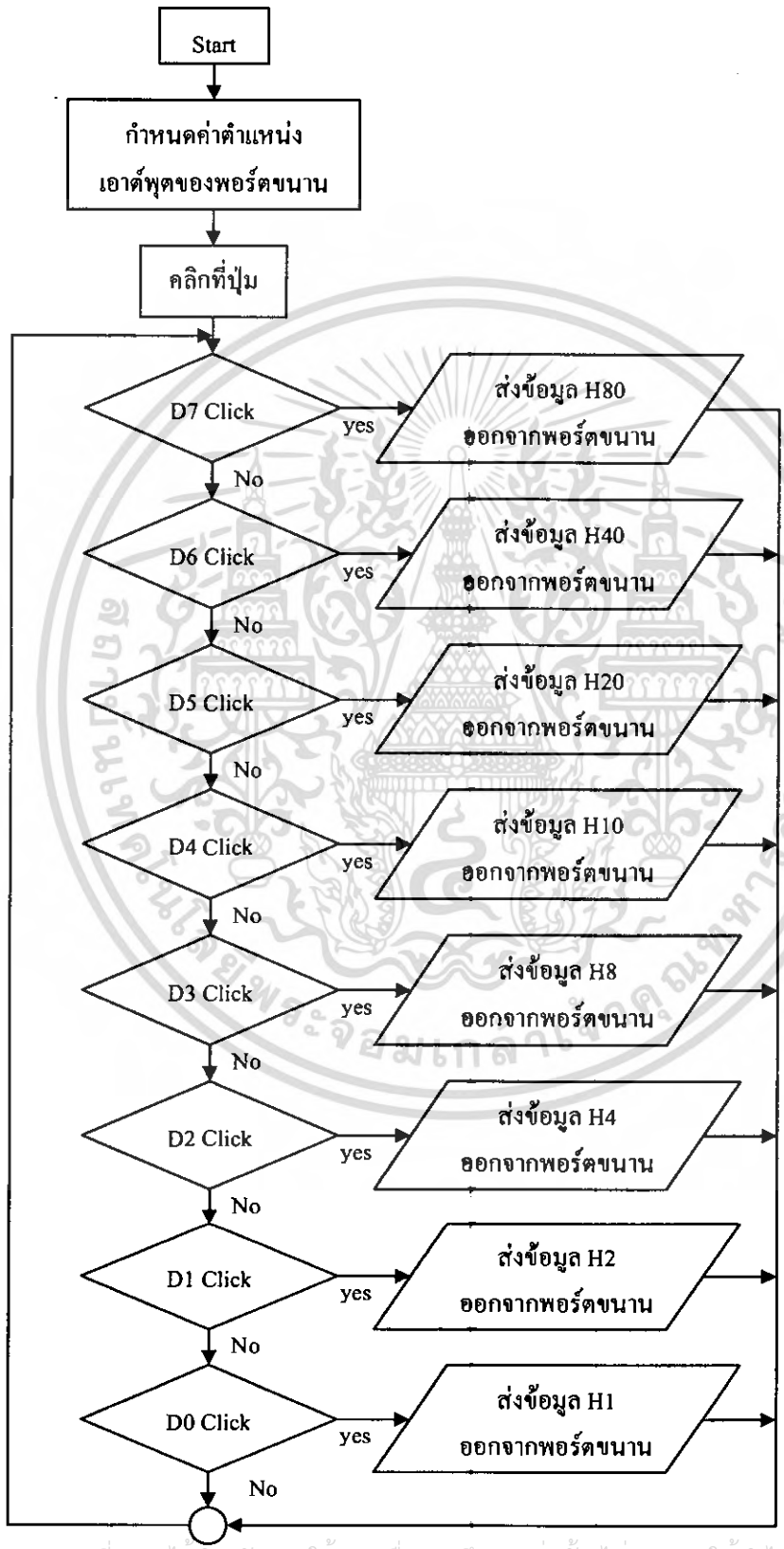
รูปที่ 5-5 การเชื่อมต่อบอร์ดทดลองกับ PC ผ่านสายเชื่อมต่อ

5.8 เริ่มเขียนโปรแกรมควบคุมหลอดแสดงผล

ก่อนที่เราจะเริ่มเขียนโปรแกรม เราควรเขียนระบบงานด้วยโฟลว์ชาร์ตกันก่อน ทั้งนี้ก็เพื่อทำความเข้าใจกับระบบงานของโรปกรมที่สร้างขึ้น และการทำงานของบอร์ดทดลองที่สัมพันธ์กัน

สำหรับโปรแกรมที่เราจะเขียนในตัวอย่างแรกนี้ จะเป็นการส่งข้อมูลออกจากพอร์ตขนานเพื่อนำไปควบคุมหลอดแสดงผล โดยข้อมูลในแต่ละข้อมูลจะถูกส่งออกไปเมื่อมีการคลิกที่ปุ่ม (ที่

สร้างด้วยคอนโทรล CommandButton) ซึ่งแต่ละปุ่มจะมีค่าประจำตำแหน่งเพื่อไปขับหลอด LED ในแต่ละตำแหน่ง



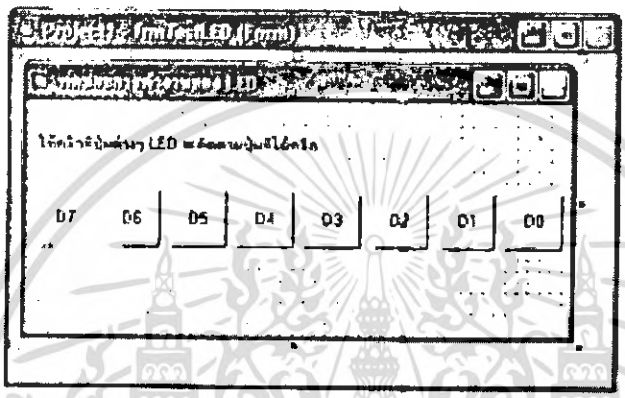
รูปที่ 5-6 โฟลว์ชาร์ตการทำงานของโปรแกรมควบคุมหลอดแสดงผล

ขั้นตอนการเขียนโปรแกรม

สำหรับตัวอย่างแรกเราจะเริ่มการทำงานแบบง่าย ๆ ดังนี้

1. ให้ทำการเปิด Project ใหม่โดยคลิกที่ File > New Project ให้เลือก Standard EXE เพื่อเปิดฟอร์มขึ้นมา

2. เมื่อมีฟอร์มขึ้นมาแล้วให้นำเอาคอนโทรล CommandButton จำนวน 8 ตัวมาวางดังรูป



รูปที่ 5 – 7 หน้าตาของโปรแกรม

3. กำหนดค่าพร็อพเพอร์ตี้ของคอนโทรลต่าง ๆ ดังตารา

คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
Form	Name	frmTestLED
	Caption	ทดสอบการทำงานของ LED
Label	Name	lblInfo
	Caption	ให้คลิกที่ปุ่มต่าง ๆ LED จะดิดตามปุ่มที่ได้คลิก
CommandButton	Name	BtnD7
	Caption	D7
CommandButton	Name	BtnD6
	Caption	D6
CommandButton	Name	BtnD5
	Caption	D5
CommandButton	Name	BtnD4
	Caption	D4

CommandButton	Name	BtnD3
	Caption	D3
CommandButton	Name	BtnD2
	Caption	D2
CommandButton	Name	BtnD1
	Caption	D1
CommandButton	Name	BtnD0
	Caption	D0

4. เขียนโค้ดคำสั่งเพื่อควบคุมการทำงานของโปรแกรม โดยเริ่มจากการประกาศโพรซีเจอร์ที่จำเป็นต้องใช้ โดยเป็นโพรซีเจอร์ที่เก็บไนไฟล์ inpout32.dll ดังนี้

```
' ประกาศโพรซีเจอร์โดยเรียกใช้จากไฟล์ inpout32.dll
Private Declare Sub Out Lib "inpout32.dll" Alias "Out32" (ByVal PortAddress
As Integer,_ ByVal Value As Integer)
' ตัวแปรแบบ Global สำหรับเก็บหมายเลขพอร์ดขนาน
Public pwrite As Integer
```

5. สำหรับการทำงานเมื่อเริ่มต้น เราจะใช้ตัวแปรแบบ Global ที่ชื่อ pwrite เก็บค่าไว้

```
Private Sub Form_Load()
    Pwrite = &H378
End Sub
```

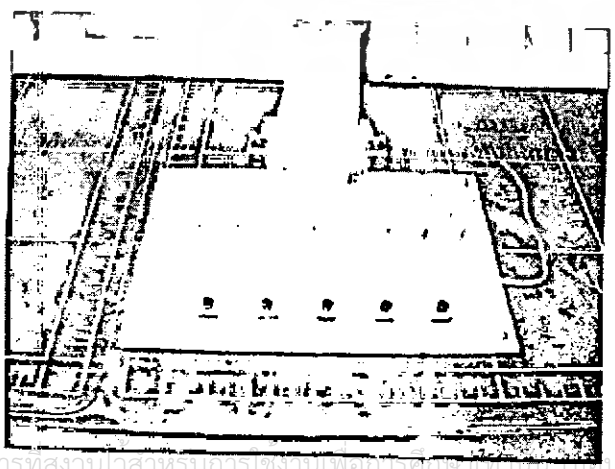
6. สำหรับโค้ดในการทำงานเมื่อคลิกปุ่มต่าง ๆ มีรายละเอียดดังนี้

```
Private Sub btnD7_Click()
    Out pwrite , &H80
End Sub
```

```
Private Sub btnD6_Click()
    Out pwrite , &H40
```

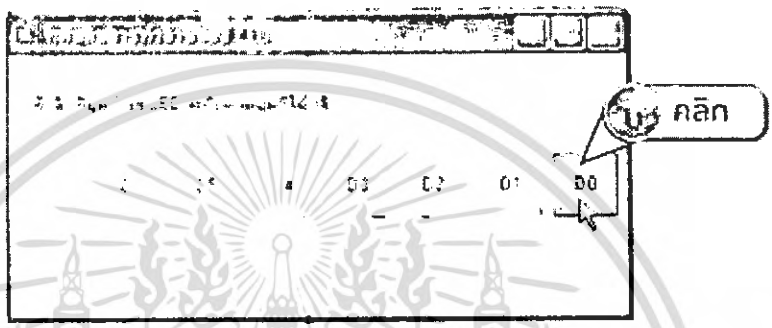
```
End Sub

Private Sub btnD5_Click()
    Out pwrite , &H20
End Sub

Private Sub btnD4_Click()
    Out pwrite , &H10
End Sub

Private Sub btnD3_Click()
    Out pwrite , &H8
End Sub

Private Sub btnD2_Click()
    Out pwrite , &H4
End Sub

Private Sub btnD1_Click()
    Out pwrite , &H2
End Sub

Private Sub btnD0_Click()
    Out pwrite , &H1
End Sub
```

7. ก่อนจะรันโปรแกรมให้ต่อบอร์ดที่เราประกอบไว้แล้วกับพอร์ตขาน LPT1 ของคอมพิวเตอร์ผ่านสายเชื่อมพอร์ตขนาน จะเห็นว่าหลอดไฟทุกหลอดของ LED จะติดหมด



รูปที่ 5 – 8 LED ทุกหลอดติดหมดเมื่อเริ่มทำงาน

8. เมื่อเรียบร้อยแล้วจึงกดปุ่ม <F5> เพื่อรันโปรแกรม ให้ทดลองคลิกที่ปุ่มต่าง ๆ ตั้งแต่ D0 ถึง D7 ลองสังเกตที่หลอด LED จะดิดสว่างตามจังหวะการคลิกที่ปุ่มต่าง ๆ เช่น ถ้าคลิกที่ D7 หลอด LED ดวงที่ 7 ก็จะดิดสว่าง ส่วนหลอด LED ดวงอื่น ๆ ก็จะดับ และถ้าเราคลิกที่ D5 หลอด LED ดวงที่ 5 ก็จะดิดสว่างเช่นกัน



รูปที่ 5 – 9 ผลการทำงานของแอพพลิเคชัน

คำอธิบาย

จากที่กล่าวไปแล้วว่า Visual Basic ไม่สามารถเข้าถึงระบบฮาร์ดแวร์ได้โดยตรง แต่ต้องใช้การสั่งงานจากโพรซีเจอร์ Out ที่เก็บในไฟล์ inpout32.dll เข้ามาช่วย ซึ่งเราจะเรียกใช้โพรซีเจอร์ Out ได้เช่นเดียวกับวิธีการเรียกใช้ฟังก์ชัน Windows API นั่นคือ จะต้องประกาศชื่อโพรซีเจอร์ที่เก็บในไฟล์ดังกล่าวเสียก่อน

ต่อมาเราจะเขียนโค้ดควบคุมการทำงาน โดยเรากำหนดให้มีการส่งข้อมูลออกทางพอร์ดขนาน โดยกำหนดแอดเดรสของพอร์ดขนานผ่านตัวแปร Pwrite = &H378 และทำการเขียนคำสั่งให้โปรแกรมส่งข้อมูลออกไปจากพอร์ดขนาน ซึ่งเมื่อมีการคลิกที่ปุ่มใดในหน้าจอโปรแกรมก็จะทำให้หลอด LED ติดสว่างตรงกับที่ได้คลิกผ่านหน้าจอ โดยหลอด LED แต่ละหลอดก็เปรียบเหมือนแต่ละบิตของข้อมูลที่ส่งไปยังพอร์ดขนาน

- ปุ่ม D0 แทนด้วย &H1 ซึ่งเท่ากับ 1 หรือ 20
- ปุ่ม DI แทนด้วย &H2 ซึ่งเท่ากับ 2 หรือ 21
- ปุ่ม D2 แทนด้วย &H4 ซึ่งเท่ากับ 4 หรือ 22
- ปุ่ม D3 แทนด้วย &H8 ซึ่งเท่ากับ 8 หรือ 23
- ปุ่ม D4 แทนด้วย &H10 ซึ่งเท่ากับ 16 หรือ 24

- ปุ่ม D5 แทนด้วย &H20 ซึ่งเท่ากับ 32 หรือ 25
- ปุ่ม D6 แทนด้วย &H40 ซึ่งเท่ากับ 64 หรือ 26
- ปุ่ม D7 แทนด้วย &H80 ซึ่งเท่ากับ 128 หรือ 27

การเขียนโปรแกรมรับข้อมูลจากสวิตช์ผ่านพอร์ตขนาน

- รูปแบบการเชื่อมต่อเพื่อการรับข้อมูลจากพอร์ตขนาน
- เขียนโปรแกรมเพื่อรับข้อมูลจากการกดสวิตช์
- สร้างโปรแกรมควบคุมกระบอกสูบจำลองจากการกดสวิตช์

5.9 การเขียนโปรแกรมรับข้อมูลจากสวิตช์ผ่านพอร์ตขนาน

จากบทที่ผ่านมาเราได้ทำการเขียนโปรแกรมเพื่อส่งข้อมูลออกจากพอร์ตขนาน เพื่อควบคุมอุปกรณ์ภายนอก ในรูปของการแสดงผล LED ส่วนบทนี้เราจะเรียนรู้ในทิศทางตรงกันข้าม นั่นคือ การเขียนโปรแกรมเพื่อรับข้อมูลจากพอร์ตขนานบ้าง ซึ่งจะใช้งานผ่านฟังก์ชันในไฟล์ inpout32.dll เหมือนเดิม

5.10 รูปแบบการเชื่อมต่อคอมพิวเตอร์ในการรับข้อมูล

สำหรับการนำเอาบอร์ดมาเชื่อมต่อกับคอมพิวเตอร์ในบทนี้มีรูปแบบดังนี้

รูปที่ 5-10 รูปแบบของการเชื่อมต่อบอร์ดทดลองในการอ่านข้อมูล

ในที่นี้เราจะสร้างโปรแกรมง่าย ๆ ในการอ่านข้อมูลจากพอร์ตขนาน โดยอ่านข้อมูลว่าเราได้กดสวิตช์อะไร ในที่นี้จะใช้ฟังก์ชัน Inp ซึ่งเป็นฟังก์ชันที่อยู่ในไฟล์ inpout32.dll

สำหรับฟังก์ชัน Inp มีรูปแบบการเรียกใช้งานดังนี้

```
Inp(portnumber)
```

ฟังก์ชัน Inp จะใช้ในการอ่านข้อมูลจากพอร์ตขนาน โดยที่ค่าของ portmunber จะเท่ากับแอดเดรสของพอร์ดขนานปกติบวกอีกหนึ่ง เช่น ถ้าเราใช้ LPT1 ซึ่งมีแอดเดรสอยู่ที่ H378 ดังนั้นหมายเลขพอร์ตอินพุดจะเท่ากับ H379 เป็นต้น

รูปที่ 5-11 โฟลว์ชาร์ตการทำงานของการรับข้อมูลจากสวิตช์

สำหรับการสร้างโปรแกรมนี้มีขั้นตอนดังนี้

1. ให้ทำการเปิด Project ใหม่โดยคลิกที่ File > New Project ให้เลือก Standard EXE เพื่อเปิดฟอร์มขึ้นมา

2. เมื่อมีฟอร์มขึ้นมาแล้วให้นำเอาคอนโทรลต่าง ๆ มาจัดวางดังรูป

3. กำหนดค่าพร็อพเพอร์ตี้ให้กับคอนโทรลต่างๆ ดังตาราง

คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
Form	Name	frmInputParallelPort
	Caption	ทดสอบการอ่านข้อมูลจากพอร์ตขนาน
Label	Name	lblInfo

	Caption	ข้อมูลจาก Parallel Port:
Label	Name	lblDataIn
	Caption	Data
Timer	Name	Timer1
	Interval	1

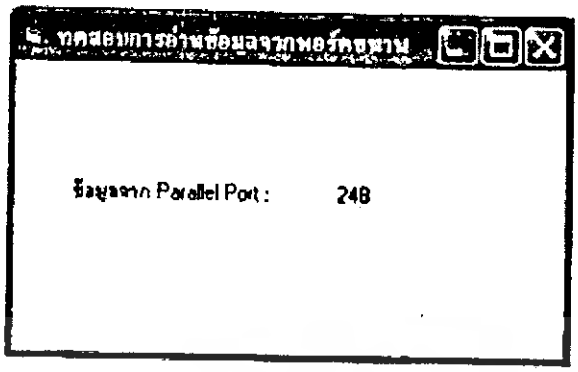
4. เขียนโค้ดคำสั่งเพื่อควบคุมการทำงานของโปรแกรมดังนี้

```
Private Declare Function Inp Lib "inpout32.dll" Alias "Inp32"  (ByVal PortAddress As_
Integer) As Integer

Public pread As Integer
Public N As Integer

Private Sub Form_Load()
    Pwrite = &H379          ' แอดเดรสของพอร์ตขนาน
End Sub
```

```
Private Sub Timer1_Timer()
    N = Inp(pread)
    lblDataIn.Caption = N
End Sub
```

5. ก่อนรันโปรแกรมให้ต่อบอร์ดที่เราประกอบไว้แล้วเข้ากับพอร์ตขนานให้เรียบร้อย

6. กดปุ่ม <F5> เพื่อรันโปรแกรม โดยโปรแกรมจะแสดงค่าที่อ่านได้จากพอร์ตขนานซึ่งจะเท่ากับ 127 ทำให้หน้าจอแสดงค่าที่อ่านได้เป็น 127 จากนั้นให้ทดลองกดสวิตช์ต่าง ๆ บนบอร์ดทดลอง ค่าที่คอมพิวเตอร์อ่านได้ก็จะเปลี่ยนแปลงไปด้วย



รูปที่ 5-12 ผลการทำงานอ่านข้อมูลจากพอร์ตขนานเมื่อกดสวิตช์ใด ๆ เลย

อธิบายการทำงาน

การรับข้อมูลจากบอร์ดทดลองนั้นสามารถสรุปได้ดังนี้

- เมื่อไม่มีการกดสวิตช์ใด ๆ ค่าที่อ่านได้จะเท่ากับ 120

Bit	Bit7	Bit6	Bit5	Bit4	Bit3
Data	128	64	32	16	8
Pin No.	11	10	12	13	15
Status	0	1	1	1	1

จากตารางเมื่อเรานำมาคำนวณการที่ไม่กดสวิตช์ตัวใด ๆ เลย

$$0 + 64 + 32 + 16 + 8 = 120$$

ในสภาวะปกติเมื่อไม่มีการกดสวิตช์ใด ๆ สถานะของบิตที่ 7 จะเป็นลอจิก 0 แต่ถ้ามีการกดสวิตช์ SW4 จะเป็นลอจิก 1

- เมื่อกด SW0 ค่าที่อ่านได้จะเท่ากับ 112

Bit	Bit7	Bit6	Bit5	Bit4	Bit3
Data	128	64	32	16	8
Pin No.	11	10	12	13	15
Status	0	1	1	1	0

จากตาราง เรานำมาคำนวณเมื่อกดสวิตช์ SW0

0 + 64 + 32 + 16 + 0 = 112

- เมื่อกด SW1 ค่าที่อ่านได้จะเท่ากับ 104

Bit	Bit7	Bit6	Bit5	Bit4	Bit3
Data	128	64	32	16	8
Pin No.	11	10	12	13	15
Status	0	1	1	0	1

จากตาราง เรานำมาคำนวณเมื่อกดสวิตช์ SW1

0 + 64 + 32 + 0 + 8 = 104

- เมื่อกด SW2 ค่าที่อ่านได้จะเท่ากับ 88

Bit	Bit7	Bit6	Bit5	Bit4	Bit3
Data	128	64	32	16	8
Pin No.	11	10	12	13	15
Status	0	1	0	1	1

จากตาราง เรานำมาคำนวณเมื่อกดสวิตช์ SW2

0 + 64 + 0 + 16 + 8 = 88

- เมื่อกด SW3 ค่าที่อ่านได้จะเท่ากับ 56

Bit	Bit7	Bit6	Bit5	Bit4	Bit3

Data	128	64	32	16	8
Pin No.	11	10	12	13	15
Status	0	0	1	1	1

จากตาราง เรานำมาคำนวณเมื่อกดสวิตช์ SW3

```
0 + 0 + 32 + 16 + 8 = 56
```

- เมื่อกด SW4 ค่าที่อ่านได้จะเท่ากับ 248

Bit	Bit7	Bit6	Bit5	Bit4	Bit3
Data	128	64	32	16	8
Pin No.	11	10	12	13	15
Status	1	1	1	1	1

จากตาราง เรานำมาคำนวณเมื่อกดสวิตช์ SW4

```
128 + 64 + 32 + 16 + 8 = 248
```

จากโค้ดควบคุมโปรแกรม เรากำหนดให้มีการรับข้อมูลทาง LPT1 โดยการกำหนดให้ pread = &H379 ข้อมูลที่รับมานั้นจะถูกเก็บไว้ในตัวแปร N และแสดงค่าที่ Label2 ด้วยคำสั่ง Label2.Caption = N

จากโปรแกรมตัวอย่าง นอกจากเราจะกดสวิตช์ทีละตัวแล้ว เรายังสามารถกดสวิตช์ได้ครั้งละหลาย ๆ ตัว โดยที่ค่าที่อ่านได้จะไม่เท่ากันขึ้นอยู่กับค่าประจำตำแหน่งในแต่ละบิต

บทที่ 6

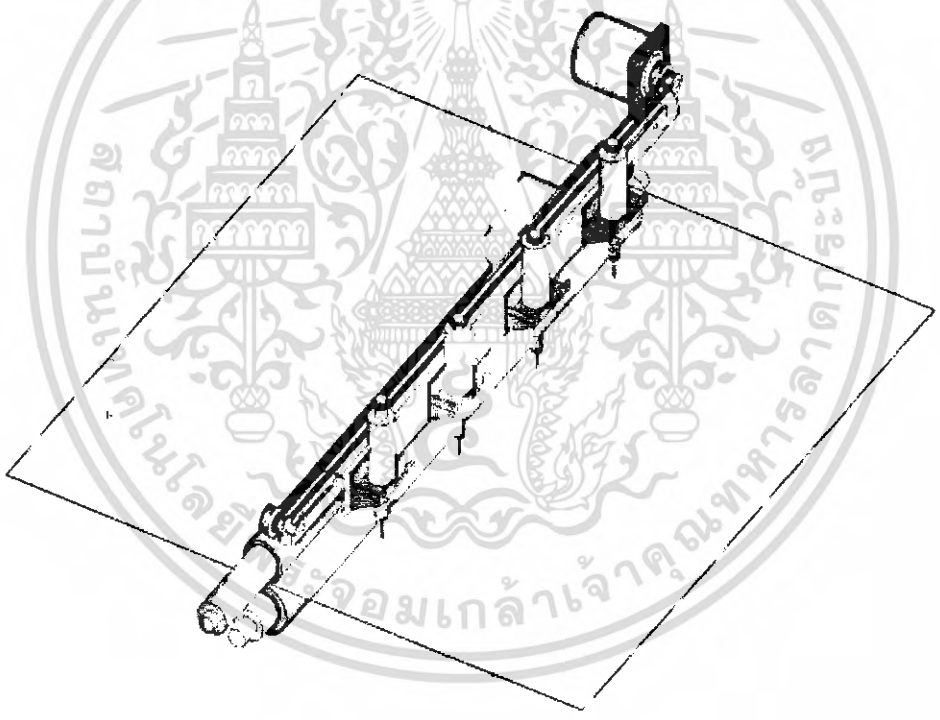
การออกแบบเครื่องเจาะแผ่นพลาสติก

6.1 ส่วนประกอบของเครื่องสร้างแบบสำหรับทอพรม

ส่วนประกอบของเครื่องสร้างแบบสำหรับทอพรมถูกแบ่งออกเป็น 3 ส่วน คือ

6.1.1 ระบบการเคลื่อนที่แบบ 2 แกน

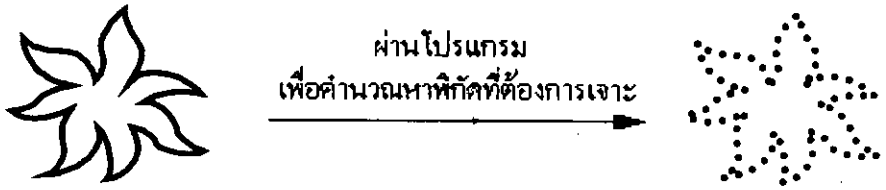
ระบบการเคลื่อนที่แบบ 2 แกน ใช้สเต็ปปิ้งมอเตอร์ 2 ตัวในการสร้างการเคลื่อนที่แบบ 2 แกน โดยตัวแรกใช้ควบคุมตำแหน่งของเข็มเจาะ ซึ่งใช้การขับเคลื่อนของสายพานในการส่งถ่ายกำลัง และอีกตัวหนึ่งใช้ในการป้อนแผ่นพลาสติกเพื่อทำการเจาะ



รูปที่ 6-1 แสดงถึงระบบการเคลื่อนที่แบบ 2 แกน ซึ่งใช้สเต็ปปิ้งในการทำงาน

6.1.2 ส่วนโปรแกรมควบคุมการเจาะของหัวเจาะ

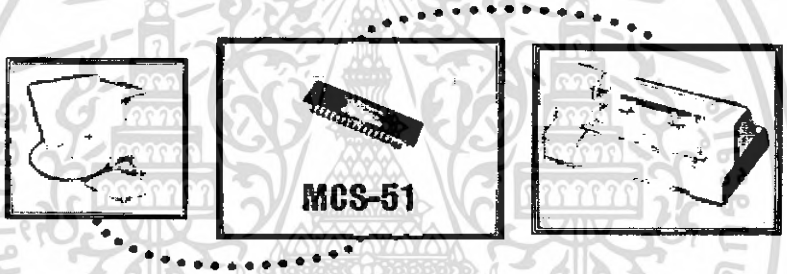
โปรแกรมที่ใช้ในการสร้างโปรแกรมสำหรับกำหนดพิกัดของเข็มเจาะ เป็นโปรแกรมที่ใช้ในการประมวลผลเพื่อหาตำแหน่งที่ต้องการเจาะ ซึ่งใช้ภาษาวิชวลเบสิกในการเขียนโปรแกรมเพื่อคำนวณหาพิกัดที่ต้องการเจาะ แล้วจึงสั่งให้เข็มเจาะเคลื่อนที่ไปยังพิกัดที่คำนวณได้โดยผ่านชุดไมโครคอนโทรลเลอร์



รูปที่ 6-2 แสดงถึงรูปลายเส้นที่ผ่านโปรแกรมเพื่อคำนวณหาพิกัดที่ต้องการเจาะ

6.1.3 ส่วนของแผงควบคุม

ส่วนของแผงควบคุม จะเป็นชุดไมโครคอนโทรลเลอร์ที่ใช้ในการเชื่อมโยงข้อมูลระหว่างโปรแกรมที่สร้างขึ้น กับตัวเครื่องเจาะพลาสติก เพื่อคำนวณ และกำหนดพิกัดของการเจาะ รวมถึงควบคุมการทำงานของเครื่องเจาะพลาสติก ใช้ชิป MCS-51 ที่เขียนโดยภาษาซี



รูปที่ 6-3 แสดงถึงการเชื่อมโยงข้อมูลจากโปรแกรมที่สร้างขึ้นไปยังตัวเครื่องเจาะพลาสติก

6.2 การป้อนพลาสติก

ในการดึงแผ่นพลาสติกเข้าไปในตัวเครื่องเพื่อให้สามารถเจาะได้ถูกตำแหน่งนั้น ใช้ลูกกลิ้ง 2 ลูกใช้ในการหนีบแผ่นพลาสติก และใช้สเต็ปปิ้งมอเตอร์ในการหมุนลูกกลิ้งเพื่อดึงให้แผ่นพลาสติกเลื่อนเข้าไป และที่ปลายทั้ง 2 ด้านของลูกกลิ้งทั้งสอง เราได้ใช้แบริ่งแบบรับแรงตามแนวรัศมีมาช่วยในการลดแรงเสียดทาน

6.2.1 ระยะเคลื่อนที่ของลูกกลิ้งตัวล่าง

เนื่องจากใช้สเต็ปปิ้งมอเตอร์ซึ่งมีความละเอียดขนาด 1.8 องศาต่อสเต็ป และเลือกใช้เฟืองซึ่งมีโมดูลเท่ากับ 1 เส้นผ่านศูนย์กลางขนาด 14 มิลลิเมตร ดังนั้นเราจะหาระยะเคลื่อนที่เชิงเส้นได้จากผลคูณระหว่างองศาการหมุนกับรัศมีของเฟือง ดังนี้

$$S = R\theta \qquad (7.1)$$

โดยที่ S คือ ระยะเคลื่อนที่เชิงเส้น

θ คือ องศาการหมุน

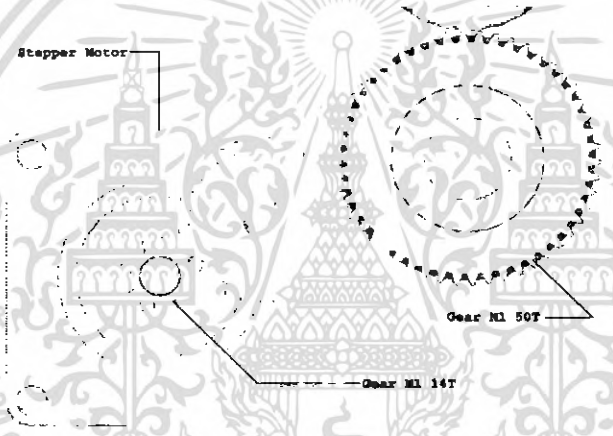
R คือ รัศมีของเฟือง, มิลิเมตร

ดังนั้น

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

$$= 0.22 \quad \text{มิลลิเมตร/ 1 สเต็ป}$$

จากการเลือกใช้เฟืองสำหรับลูกกลิ้ง ซึ่งมีขนาดเท่ากับเส้นผ่านศูนย์กลางของลูกกลิ้งพอดี ดังนั้นเมื่อสเต็ปปิ้งมอเตอร์หมุนไป 1 รอบ ก็จะทำให้ได้ระยะการเคลื่อนที่เชิงเส้นเท่ากับระยะการเคลื่อนที่ของเฟืองตัวเล็กด้วย ซึ่งจะเท่ากับ 0.88 มิลลิเมตร



รูปที่ 6-4 แสดงถึงระบบการขับเฟืองที่ใช้ในลูกกลิ้งตัวล่าง

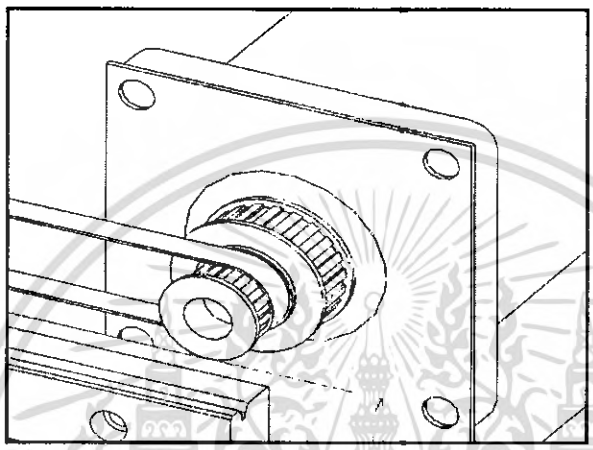
6.3 การเคลื่อนที่ของหัวเจาะ

ในการเจาะแผ่นพลาสติก เราใช้เข็มเจาะซึ่งทำจากท่อสแตนเลสสตีลลับคมที่ปลาย กระแทกลงไปบนแผ่นพลาสติก เพื่อตัดแผ่นพลาสติกให้ขาดเป็นรู โดยการเคลื่อนที่ขึ้นลงของเข็มเจาะนี้ใช้โซลินอยด์แม่เหล็กในการควบคุม และเราได้เลือกใช้โซลินอยด์ถึง 4 ตัวในการเจาะเพื่อเพิ่มความเร็วในการทำงาน

การเคลื่อนที่ของโซลินอยด์แม่เหล็กเหล่านี้ ตัวโซลินอยด์จะถูกยึดติดกับแผ่นเหล็กซึ่งติดอยู่กับลิเนียร์แบริ่ง ซึ่งจะทำให้เกิดการเคลื่อนที่ในแนวเส้นตรงและช่วยลดแรงเสียดทานในการเคลื่อนที่ โดยแผ่นเหล็กนี้จะถูกยึดติดกับสายพาน และคล้องผ่านสเต็ปมอเตอร์อีกตัว เพื่อให้สเต็ปมอเตอร์เป็นตัวควบคุมการเคลื่อนที่ของหัวเจาะ

6.3.1 ระยะเคลื่อนที่ของมูเล่สายพาน

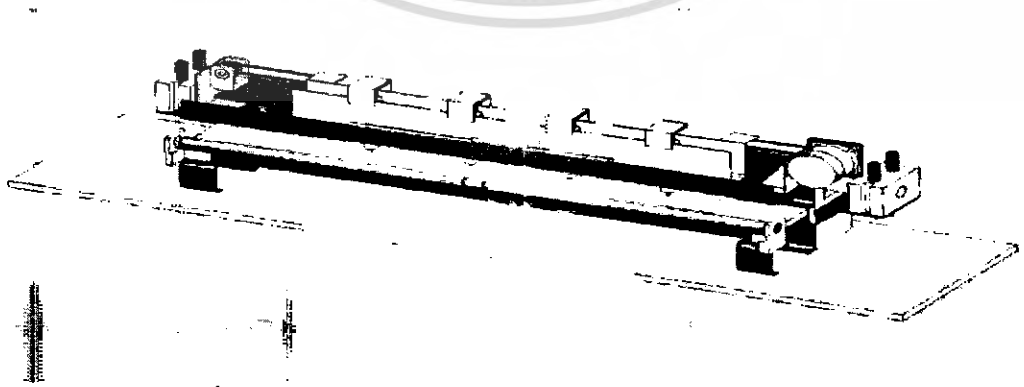
ระยะเคลื่อนที่ของมูเล่สายพานใช้หลักการเช่นเดียวกับการขับเคลื่อนลูกกลิ้ง โดยเลือกใช้มูเล่เส้นผ่านศูนย์กลาง 14 มิลลิเมตรเช่นกัน ดังนั้นเราจึงได้ระยะการเคลื่อนที่ของสายพาน 0.88 มิลลิเมตร/สเต็ป เช่นเดียวกัน



รูปที่ 6-5 แสดงถึงระบบการส่งกำลังโดยใช้สายพานเพื่อกำหนดตำแหน่งของเข็มเจาะ

6.3.2 การออกแบบระยะห่างของหัวเจาะแต่ละหัว

เนื่องจากเราเลือกใช้โซลินอยด์แม่เหล็ก 4 ตัวในการเจาะ เพื่อลดระยะการเคลื่อนที่ซึ่งจะทำให้เครื่องทำงานได้เร็วขึ้นเนื่องจากไม่ต้องวิ่งไปยังตำแหน่งต่าง ๆ เป็นระยะทางมากนัก และนอกจากนี้ยังช่วยยืดอายุการทำงานของโซลินอยด์เนื่องจากการทำงานหนักได้อีกด้วย ดังนั้นการใช้โซลินอยด์ถึง 4 ตัว ในการทำงานจะต้องมีการแบ่งขอบเขตของแต่ละตัวอย่างแน่นอน ซึ่งแต่ละตัวจะไม่ทำงานซ้ำซ้อนกันและยังเป็นตำแหน่งที่หัวเจาะสามารถเคลื่อนไปได้อีกด้วย



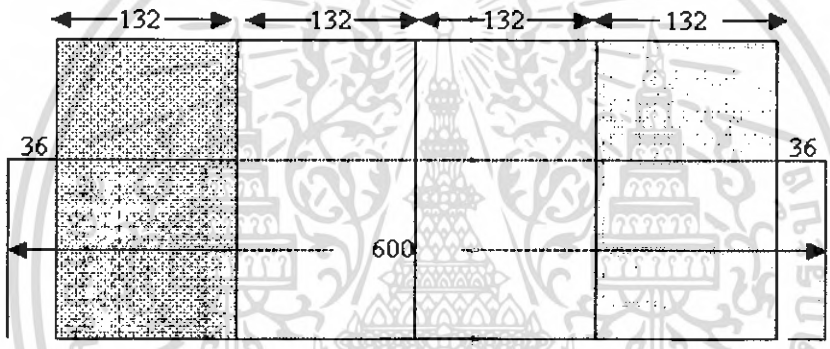
รูปที่ 6-6 แสดงถึงการใช้โซลินอยด์แม่เหล็ก 4 ตัวในการเจาะ

จากระยะกว้างสุดที่หัวเจาะสามารถเลื่อนไปได้คือ ประมาณ 540 มิลลิเมตร และจำนวนหัวเจาะ 4 หัว เราจะได้สมการระยะเจาะดังนี้

$$0.88 \times S \times 4 < 540 \qquad (7.2)$$

โดยที่ S คือ ระยะเคลื่อนที่เชิงเส้น

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



รูปที่ 6-7 แสดงถึงความกว้างของพื้นที่การทำงานของโซลินอยด์แต่ละตัวในหน่วยมิลลิเมตร

บทที่ 7

การออกแบบโปรแกรมคำนวณพิกัดการเจาะและวิธีการใช้งานของเครื่องเจาะแผ่นพลาสติก

7.1 สาเหตุที่เลือกใช้โปรแกรมวิชวลเบสิกในการเขียนโปรแกรม

1. เป็นภาษาระดับกลาง มีลักษณะคล้ายภาษาอังกฤษ จึงทำให้ง่ายต่อการทำความเข้าใจและพัฒนาโปรแกรม

2. มีคำสั่งที่ใช้ในการติดต่อสื่อสารกับพอร์ตขนานซึ่งสามารถนำไปควบคุมเครื่องได้

3. มีความสามารถในงานประเภทกราฟฟิคอยู่ในระดับที่น่าพอใจ ทำให้สามารถพัฒนาโปรแกรมได้อย่างสะดวก

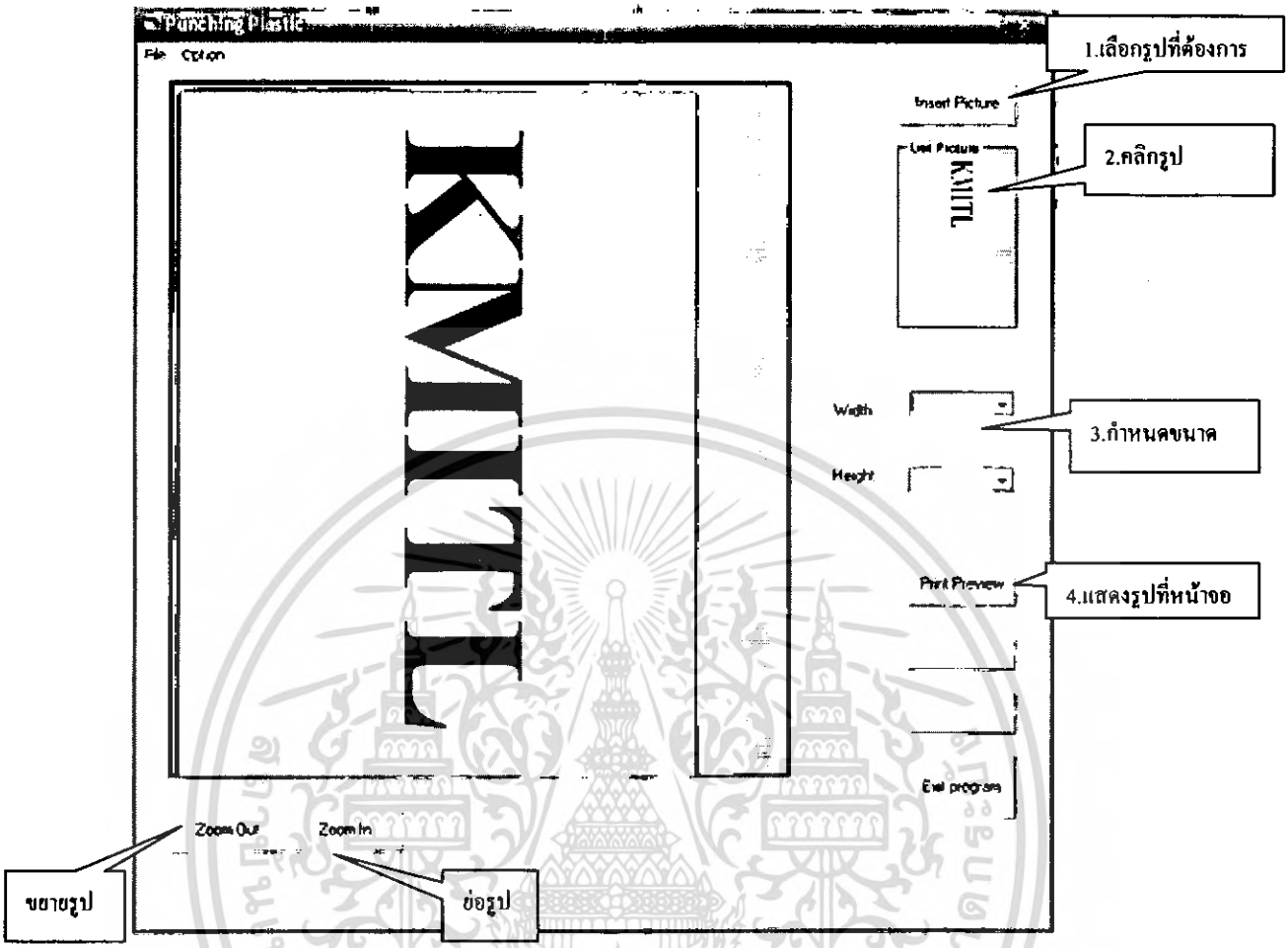
4. วิชวลเบสิกใช้พัฒนาโปรแกรมที่ใช้สื่อสารกับผู้ใช้เป็นหลัก ดังนั้นจึงสามารถพัฒนาหน้าตา(Interface) ของโปรแกรมออกมาได้สวยงามน่าใช้

7.2 หลักการคำนวณหาพิกัดของโปรแกรม

เมื่อมีการนำภาพเข้ามาในโปรแกรม โปรแกรมจะจัดภาพให้เป็นขนาดมาตรฐานเดียวคือไม่ว่ารูปที่เข้ามาจะเป็นรูปเล็กหรือใหญ่ โปรแกรมก็จะทำภาพให้เป็นขนาดมาตรฐานเท่ากับ 4900 * 7500 พิกเซล แต่เครื่องเจาะสามารถทำได้ 196 * 300 รู ดังนั้นจึงทำการแบ่งภาพออกเป็นส่วนให้ได้ขนาด 196 * 300 ส่วน โดยแต่ละส่วนจะมีขนาดเท่ากับ 25 * 25 พิกเซล ต่อจากนั้นก็จะทำการแบ่งภาพออกเป็น 4 ส่วน เพื่อให้ตรงกับหัวเจาะทั้ง 4 หัว จากนั้นก็ทำการสแกนในแต่ละส่วน(ขนาด 25 * 25 พิกเซล)ถ้าพบว่ามีพิกเซลใดพิกเซลหนึ่งเป็นสีดำก็จะทำการบันทึกค่าไว้ พอสแกนครบทุกส่วนแล้วก็จะนำค่าพิกัดเหล่านั้นมาสร้างเป็นโค๊ด หลังจากนั้นก็ทำการส่งโค๊ดไปยังเครื่องเจาะต่อไป

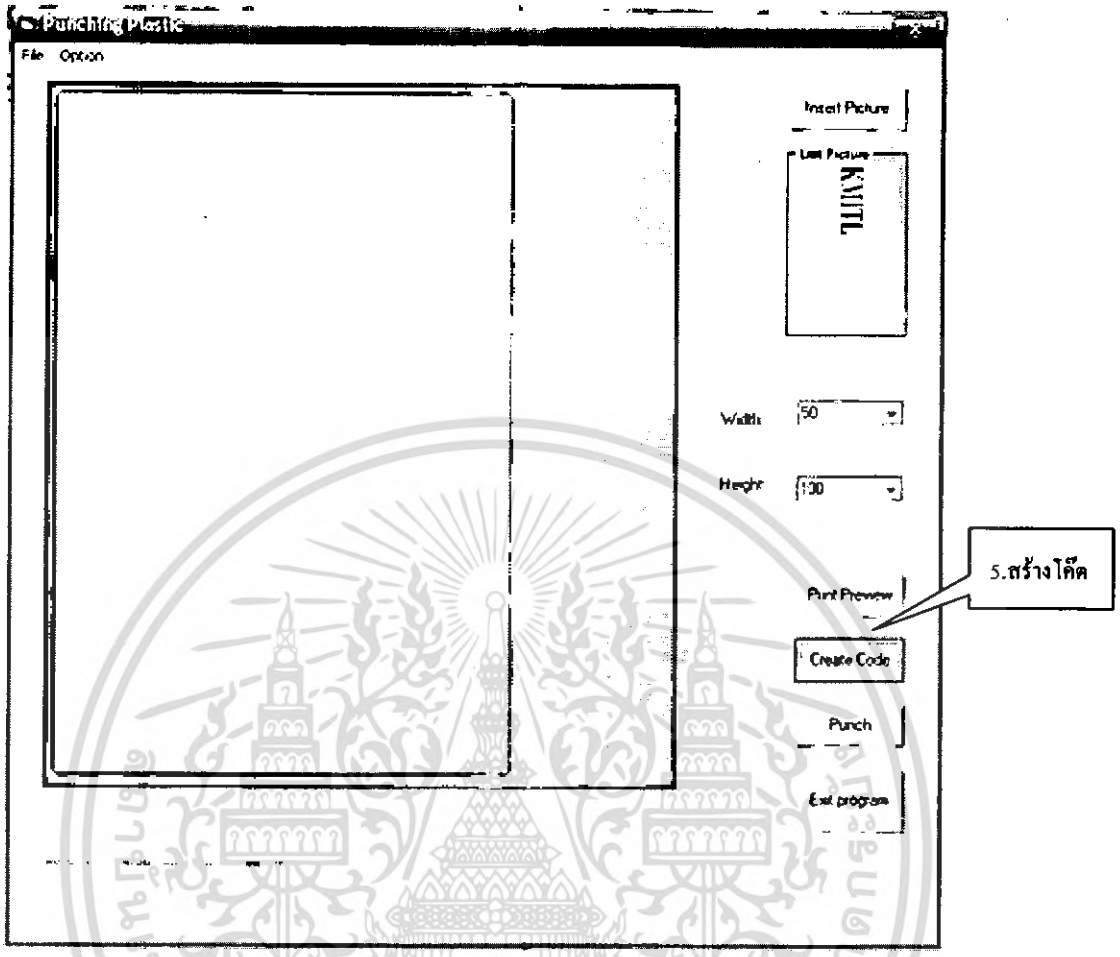
7.3 การใช้โปรแกรมคำนวณพิกัดการเจาะ

หลังจากผู้ใช้เปิดโปรแกรมขึ้นมาแล้ว กดปุ่ม Insert Picture บนโปรแกรม เป็นการเลือกภาพลายเส้นเข้ามาในโปรแกรม แล้วกดปุ่มตกลง ภาพลายเส้นก็จะเข้ามาโชว์ตรงช่องรูปภาพในโปรแกรม ซึ่งตัวโปรแกรมสามารถรับภาพเข้ามาได้ 2 ภาพ เมื่อผู้ใช้ต้องการจะเจาะรูปใดก็จะทำการคลิกที่รูปนั้น แล้วรูปที่คลิกก็จะโชว์ตรงช่องแสดงผลผู้ใช้สามารถคลิกตรงปุ่ม Zoom In หรือ Zoom Out เพื่อที่จะดูภาพในส่วนต่างๆ



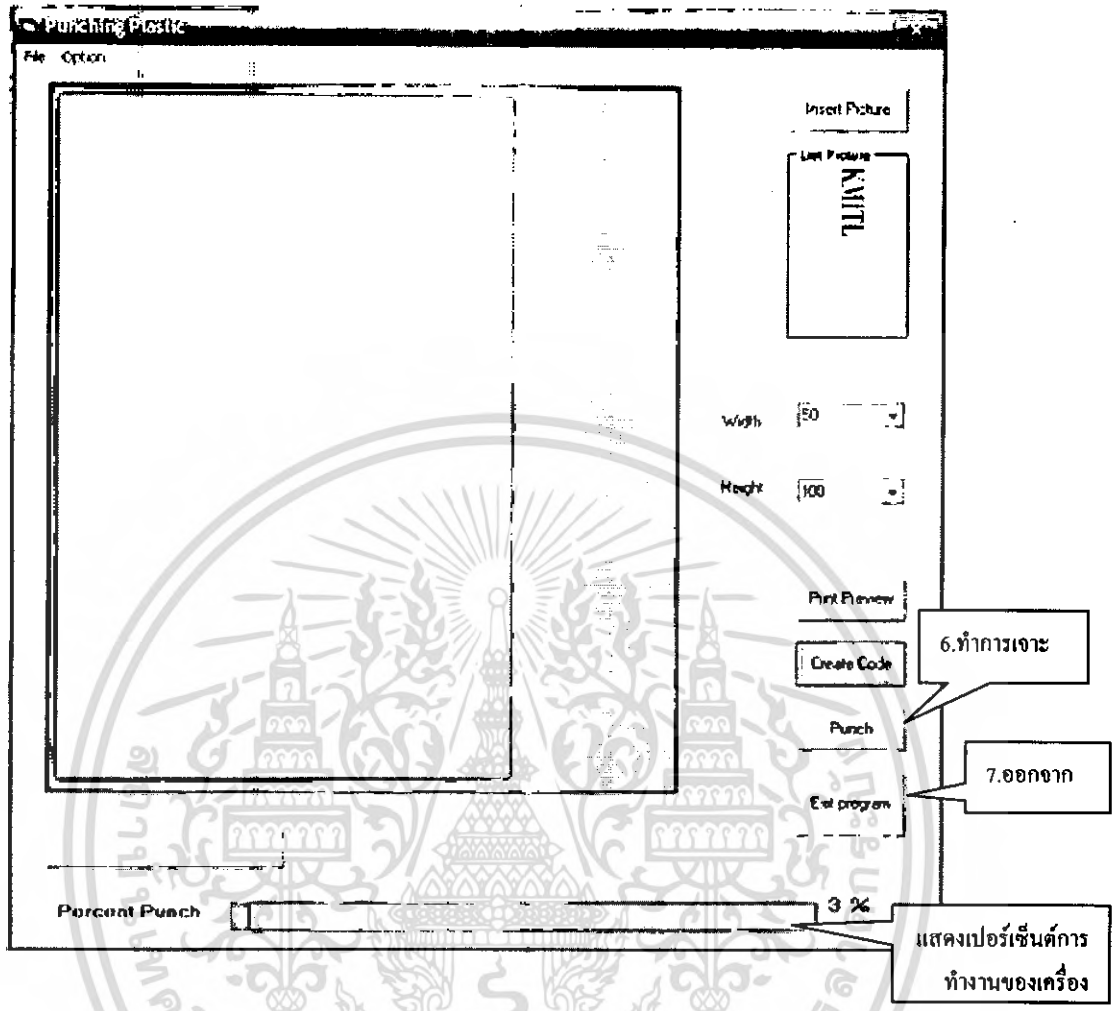
รูปที่ 7-1 แสดงการใช้โปรแกรมในส่วนของการนำรูปเข้าโปรแกรม

หากผู้ใช้ยังไม่ได้นำรูปภาพเข้า ตัวโปรแกรมจะแจ้งให้ผู้ใช้ได้ทราบบนจอ เพื่อให้ผู้ใช้ได้กลับไปแก้ไข หากทุกอย่างเรียบร้อยแล้วโปรแกรมจะทำการปรับขนาดรูปภาพ หลังจากปรับขนาดแล้วก็เข้าที่ช่องกำหนดขนาดเพื่อกำหนดขนาดของภาพที่ต้องการ จากนั้นถ้าผู้ใช้ต้องการดูตัวอย่างภาพก่อนเจาะ ก็ให้คลิกที่ปุ่ม Print Preview จากนั้นก็คลิกที่ปุ่ม Create Code โปรแกรมจะทำการคำนวณพิกัดต่างๆและแสดงตัวอย่างขึ้นมาเป็นจุดสีแดง



รูปที่ 7-2 แสดงการใช้โปรแกรมในส่วนของการสร้างโค๊ด

หากผู้ใช้ต้องการเจาะก็สามารถคลิกที่ปุ่ม Punch ได้เลย โดยโปรแกรมนำค่าพิกัดซึ่งได้คำนวณออกมาแล้วส่งออกไปเป็นโค้ดคำสั่งให้เครื่องเจาะ โดยหลังจากส่งคำสั่งแรกออกไปแล้ว โปรแกรมจะรอรับสัญญาณจากชุดควบคุมเครื่องเจาะ หากชุดควบคุมได้รับโค้ดแล้ว ชุดควบคุมจะส่งสัญญาณกลับมาให้โปรแกรมได้รับทราบว่าโค้ดที่ถูกส่งออกไปได้รับแล้วเรียบร้อย โปรแกรมจึงจะส่งพิกัดต่อไป และจะทำงานวนเช่นนี้ไปเรื่อยจนเสร็จการทำงานเมื่อเสร็จการทำงานแล้ว ผู้ใช้ก็สามารถคลิกตรงปุ่ม Exit เพื่อออกจากโปรแกรม

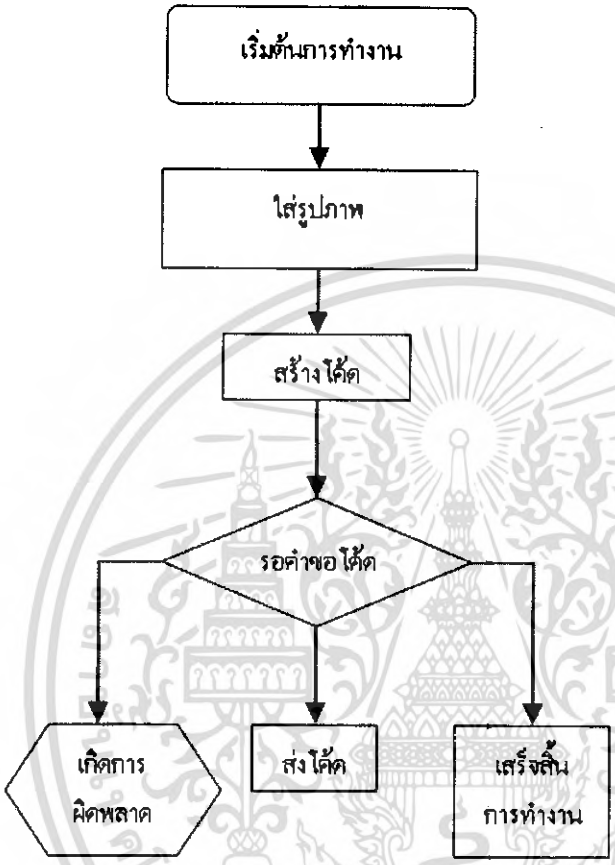


รูปที่ 7-3 แสดงการใช้โปรแกรมในส่วนของการดำเนินการเจาะ

7.4 การใช้เครื่องเจาะแผ่นพลาสติก

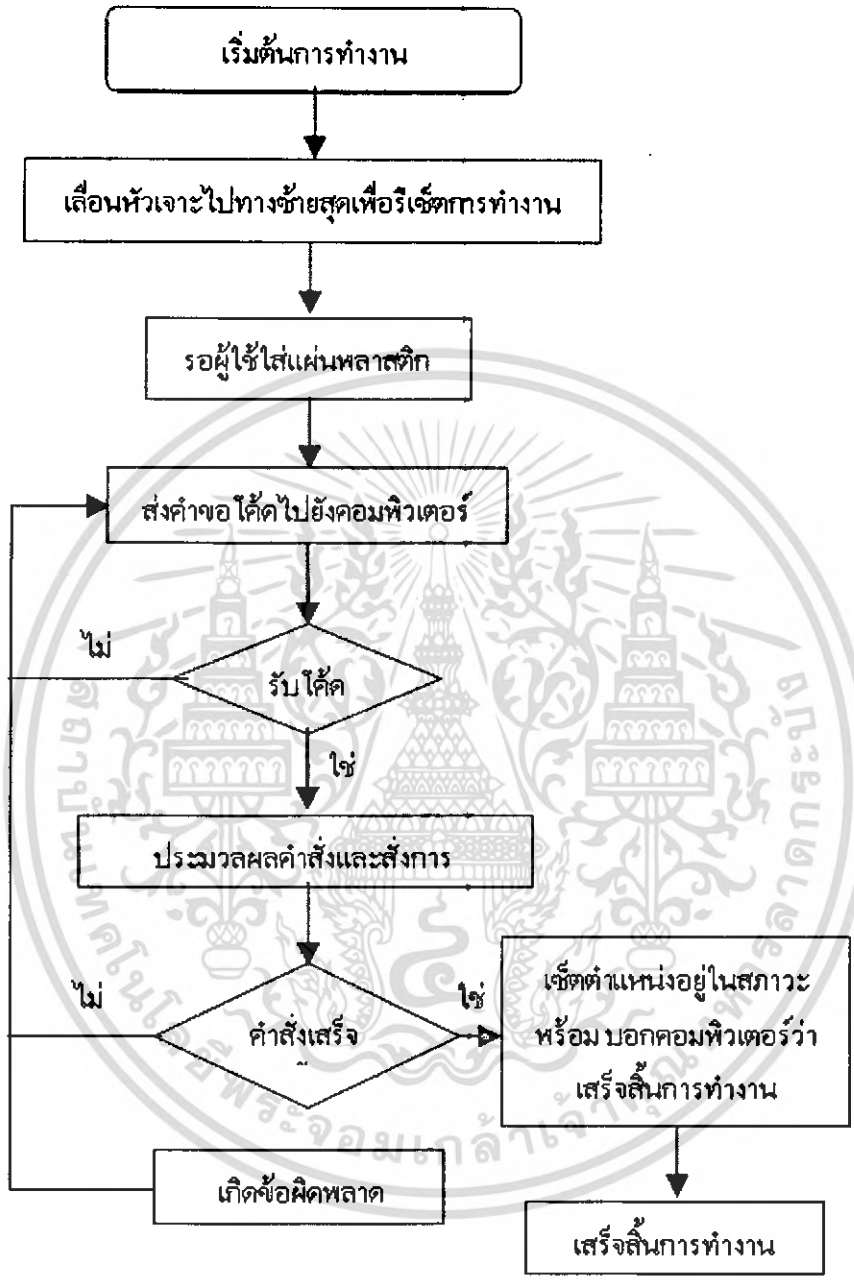
การทำงานของเครื่องเริ่มจากเครื่องจะทำการรีเซ็ตตัวเองมาอยู่ในตำแหน่งเริ่มต้นในตอนเปิดการทำงานของเครื่อง จากนั้นก็ให้ทำการใส่แผ่นพลาสติกที่เครื่องจะมีปุ่มกดเพื่อฟีดแผ่นพลาสติกเข้าไปแล้วเครื่องก็จะรอคำสั่งจากคอมพิวเตอร์เมื่อเครื่องได้รับโค๊ดจากคอมพิวเตอร์ก็จะทำการแปลงโค๊ดแล้วสั่งให้มอเตอร์กับโซลีนอยด์ให้ทำงานในตำแหน่งตามโค๊ดที่ได้รับ

การออกแบบโปรแกรมสั่งงาน



รูปที่ 7-4 แสดงถึงลำดับการทำงานของโปรแกรมสั่งงาน

การออกแบบระบบควบคุมการสั่งงานเครื่องเจาะ



รูปที่ 7-5 แสดงถึงลำดับการทำงานของระบบควบคุมการสั่งงาน

บทที่ 8

รูปแบบและผลการทดลอง

8.1 รูปแบบการทดลอง

รูปแบบการทดลองเพื่อหาประสิทธิภาพรวมของเครื่องสร้างแบบสำหรับทอพรมนี้ จะแบ่งการทดลองออกเป็น 4 ส่วนด้วยกัน คือ

1. ส่วนโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะที่จะทำการเจาะแผ่นพลาสติก
2. ส่วนของหัวเจาะที่ใช้ในการเจาะแผ่นพลาสติก
3. ส่วนของระบบโต๊ะงานเคลื่อนที่แบบ 2 แกน
4. ส่วนของความเร็วในการเจาะ

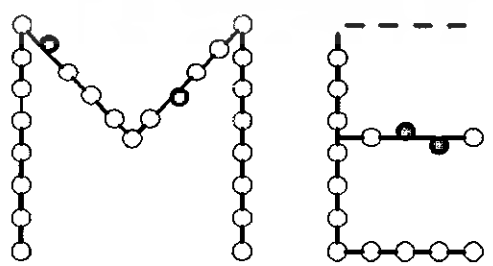
เพื่อทดสอบว่าเครื่องสร้างแบบสำหรับทอพรมยังมีข้อบกพร่องส่วนใดบ้างที่ต้องได้รับการแก้ไข ทั้งนี้เพื่อให้เครื่องสร้างแบบสำหรับทอพรมสามารถใช้งานได้จริงอย่างมีประสิทธิภาพมากที่สุด

8.2 วิธีการทดลอง

1. ส่วนโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะที่จะทำการเจาะแผ่นพลาสติก มีขั้นตอนดังนี้คือ

นำรูปลายเส้นเข้าโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะจำนวน 10 รูป ซึ่งรูปลายเส้นแต่ละรูปจะมีรูปแบบแนวเส้นที่เหมือนกัน จากนั้นจึงสั่งให้โปรแกรมเริ่มต้นทำงานเดินตามแนวเส้นแล้วแสดงพิกัดการทำงานออกมาบนรูปที่นำเข้าไปโดยแสดงเป็นจุด แล้วจึงนับจำนวนพิกัดการทำงานที่แสดงพิกัดไม่ตรงตามรูปลายเส้นที่นำเข้าไป เพื่อคำนวณหาประสิทธิภาพของตัวโปรแกรมที่ใช้ในการกำหนดตำแหน่งของเข็มเจาะโดยคำนวณจากสูตร

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



รูปที่ 8-1 แสดงถึงพิกัดการทำงานที่เดินตรง และไม่ตรงตามแนวเส้น

2. ส่วนของหัวเจาะที่ใช้ในการเจาะแผ่นพลาสติก มีขั้นตอนการทดลองดังนี้คือ

ทดสอบเจาะชิ้นงานจริงโดยใช้เครื่องแม่แบบสำหรับทอพรมจำนวน 10 ชุด ซึ่งชิ้นงานแต่ละชุดจะมีรูปแบบแนวเส้นไม่แตกต่างกัน จากนั้นจึงนำชิ้นงานที่ได้มานับจำนวนรูที่เจาะไม่เข้า หรือรูที่เจาะเสีย เพื่อคำนวณหาประสิทธิภาพของหัวเจาะที่ใช้ในการเจาะแผ่นพลาสติก โดยคำนวณจากสูตร

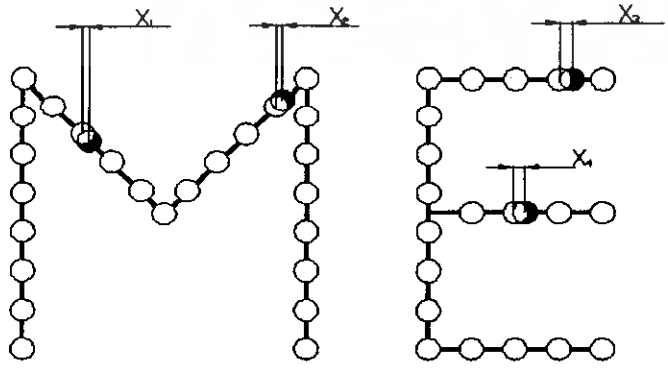
$$\text{ประสิทธิภาพของหัวเจาะ} = \frac{\text{จำนวนรูเจาะไม่เสียทุกรู}}{\text{จำนวนรูเจาะทุกรู}}$$

รูปที่ 8-2 แสดงถึงตำแหน่งของรูเจาะที่เจาะตรง และไม่ตรงตามแนวเส้น รวมถึงรูเจาะที่เจาะเสีย

3. ส่วนของระบบการเคลื่อนที่แบบ 2 แกน มีขั้นตอนการทดลองดังนี้คือ

ทำการกำหนดพิกัด X-Y เข้าเครื่องแม่แบบสำหรับทอพรมจำนวน 100 พิกัด เพื่อให้เครื่องทำการเจาะตามพิกัดตามพิกัดที่ได้กำหนดไว้ จากนั้นจึงนำชิ้นงานที่ได้มาเปรียบเทียบพิกัด เพื่อวัดระยะคลาดเคลื่อนตามแนวแกนเอ็กซ์ และ X แกน Y เพื่อคำนวณหาประสิทธิภาพของระบบโต๊ะงานเคลื่อนที่แบบ 2 แกน จากสูตร

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

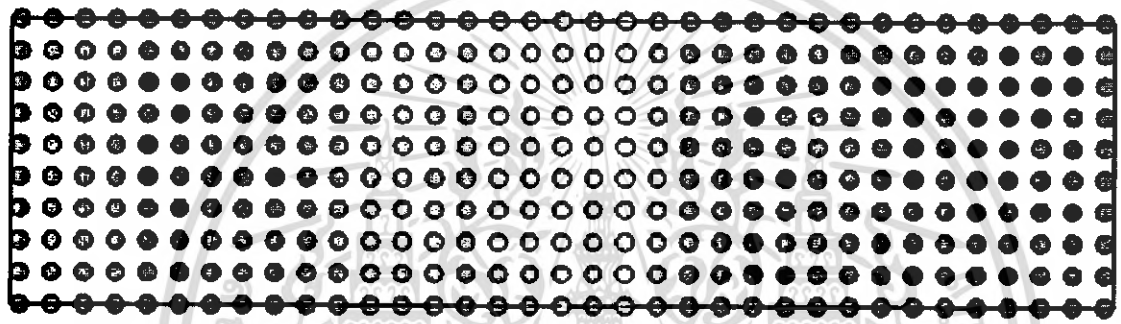


รูปที่ 8-3 แสดงถึงพิกัดของรูเจาะที่เจาะตรง และไม่ตรงตามแนวเส้น

4. ส่วนของระบบความเร็วของหัวเจาะมี ขั้นตอนการทดลองดังนี้คือ

ทำการเจาะแผ่นงานพลาสติกด้วยหัวเจาะ โดยให้ชิ้นงานที่มีแถวเป็นแนวเส้นตรงและมีระยะที่เท่ากันตลอด โดยแต่ละแผ่นจะมีขนาด 10 × 86 รู และทำการเจาะจำนวณ 10 ชุดด้วยกัน และมีการคำนวณจากสูตรหาความเร็วดังต่อไปนี้

$$\text{ความเร็วในการเจาะ} = \frac{\text{จำนวนรูทั้งหมด}}{\text{เวลา}}$$

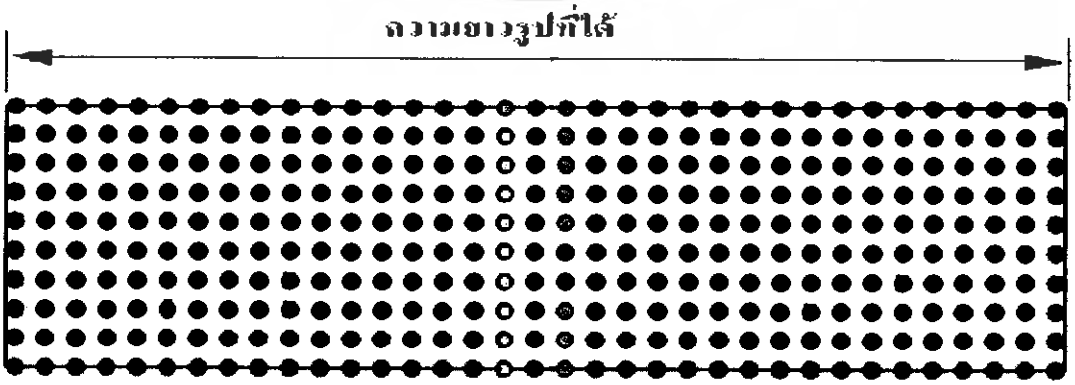


รูปที่8-4 แสดงถึงจำนวนดของรูเจาะที่เจาะตรง และวัดเทียบกับเวลา

5. ความผิดพลาดในการกำหนดขนาด

ทำการเจาะแผ่นงานพลาสติกด้วยหัวเจาะ โดยให้ชิ้นงานที่มีแถวเป็นแนวเส้นตรงและมีระยะที่เท่ากันตลอด โดยแต่ละแผ่นจะมีขนาด 10 × 86 รู และทำการเจาะจำนวณ 10 ชุดด้วยกัน และมีการคำนวณจากสูตรหาความผิดพลาดในการกำหนดขนาดดังต่อไปนี้

$$\text{ความผิดพลาดในการกำหนดขนาด} = \frac{\text{ความยาวที่ต้องการ} - \text{ความยาวรูปที่ได้}}{\text{ความยาวที่ต้องการ}}$$



รูปที่8-5 แสดงถึงขนาดของรูปที่เจาะ

8.3 ผลการทดลอง

จากการทดลองใช้เครื่องเจาะแผ่นพลาสติกพบว่า

1. ตำแหน่งของรูที่ต้องการเจาะมีระยะคลาดเคลื่อนประมาณ 1.23 มิลลิเมตรต่อรูเนื่องจากความคลาดเคลื่อนในการติดตั้งหัวเจาะ

2. ประสิทธิภาพของตัวโปรแกรมคิดเป็น 94.6 %

3. ประสิทธิภาพของหัวเจาะคิดเป็น 87.2 %

4. ความผิดพลาดในการกำหนดขนาดประมาณ 4.3 %

5. ความเร็วในการเจาะคิดเป็น 12 รูต่อวินาที

บทที่ 9

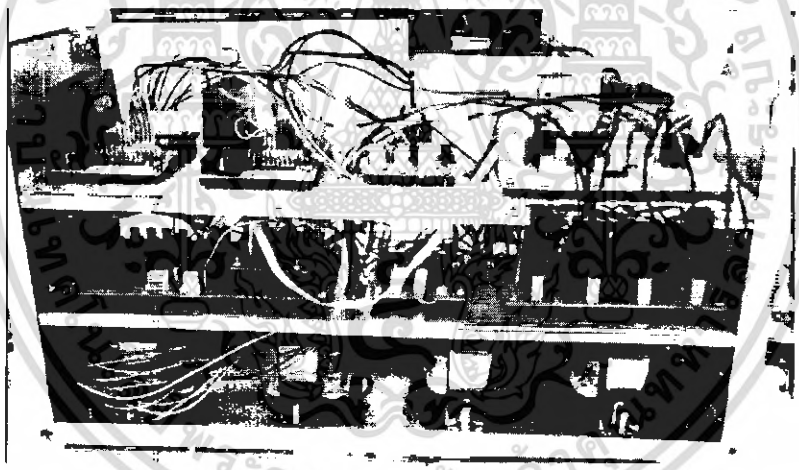
สรุปและวิเคราะห์ผลการทดลอง

9.1 ส่วนโปรแกรมควบคุมการเจาะ

โปรแกรมวิชวลแบสิกที่ได้พัฒนาและแก้ไขมีการใช้งานได้สะดวกมากขึ้น อีกทั้งยังมีปุ่มให้เลือกขนาดภาพที่ต้องการอีกได้ด้วย

9.2 ส่วนของแผงควบคุม

- จากการปรับปรุงและแก้ไขวงจรสามารถทำงานได้อย่างมีประสิทภาพ อีกทั้งวงจรที่ได้ยังมีความซับซ้อนน้อยลง



รูปที่ 9-1 แสดงส่วนของแผงควบคุม

9.3 เครื่องเจาะแผ่นพลาสติก

- เนื่องจากงบประมาณและเวลาที่มีจำกัด ทำให้ชิ้นส่วนแต่ละชิ้นถูกออกแบบมาให้สามารถสร้างได้โดยง่าย วัสดุหาได้ทั่วไปตามท้องตลาด และอุปกรณ์บางตัวเป็นชิ้นส่วนที่หาได้รอบตัว

- แม้ว่าชิ้นส่วนแต่ละชิ้นจะถูกออกแบบมาให้ง่ายต่อการผลิต แต่ในความเป็นจริงแล้วการผลิตเครื่องเจาะแผ่นพลาสติกด้วยมือจะทำให้เกิดความคลาดเคลื่อนขึ้นทำให้ประสิทธิภาพบางส่วนลดลง

9.4 แนวทางในการแก้ไข

แนวทางในการแก้ไขจะแบ่งการวิเคราะห์ออกเป็น 3 ส่วนคือ

9.4.1 ส่วนโปรแกรมควบคุมการเจาะ

- เนื่องจากการนำมาใช้งานยังไม่ค่อยดีนักจึงควรหลีกเลี่ยง การคอนโทลในงานที่ซับซ้อน

- เรื่องของความเร็วในการสแกนขึ้นอยู่กับปัจจัยหลายอย่าง เช่น คอมพิวเตอร์มีเวอร์ชันสูงก็จะทำงานได้เร็วขึ้น

9.4.2 ส่วนของแผงควบคุม

- เนื่องจากวงจรควบคุมได้ทำขึ้นมานี้เพื่อการทดลองและเพื่อให้ง่ายต่อการแก้ไขจึงยังมีขนาดในบางส่วนที่ใหญ่เกินไป ดังนั้นการออกแบบแผงวงจรจึงควรรวมแผงวงจรต่าง ๆ เข้ามาไว้ในแผง เดียวกัน และจัดให้มีจุดรวมสัญญาณและจุดส่งสัญญาณต่าง ๆ เป็นกลุ่ม ๆ เพื่อลดขนาดของวงจรลง

9.4.3 เครื่องเจาะแผ่นพลาสติก

- เนื่องจากงบประมาณและเวลาที่มีจำกัด ทำให้ชิ้นส่วนแต่ละชิ้นถูกออกแบบมาให้สามารถสร้างได้โดยง่าย วัสดุหาได้ทั่วไปตามท้องตลาด และอุปกรณ์บางตัวเป็นชิ้นส่วนที่หาได้รอบตัว ดังนั้นเครื่องอาจดูไม่สวยงามมากนัก และนอกจากนี้ตัวเครื่องยังมีน้ำหนักค่อนข้างมาก ไม่สะดวกต่อการเคลื่อนย้าย จึงควรเลือกวัสดุที่มีน้ำหนักเบา

แม้ว่าชิ้นส่วนแต่ละชิ้นจะถูกออกแบบมาให้ง่ายต่อการผลิต แต่ในความเป็นจริงแล้วการผลิตด้วยมือจะทำให้เกิดความคลาดเคลื่อนขึ้นทำให้ประสิทธิภาพบางส่วนลดลง ดังนั้นหากต้องการสร้างเพื่อการจำหน่ายแล้วอาจจะต้องออกแบบบางส่วนใหม่ให้สามารถทำงานได้ดีขึ้น และชิ้นส่วนควรสร้างด้วยเครื่องจักรอัตโนมัติ เพื่อลดความคลาดเคลื่อนที่เกิดขึ้น

บรรณานุกรม

[1] อภิชาติ ภู่พลับ, "เริ่มต้นเขียนโปรแกรมติดต่อกันและควบคุมฮาร์ดแวร์ด้วย Visual Basic 6", Infopress Develoop 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", Phillips CentreFor industrial Technology (CFT)

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

[8] วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิดรวิไล, "เรียรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชฉบับ 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

```
Option Explicit
Option Base 1
Private Declare Sub Out Lib "inpout32.dll" Alias "Out32" (ByVal portaddress As Integer, ByVal
value As Integer)
Public portoutput As Integer
Private Declare Function Inp Lib "inpout32.dll" Alias "Inp32" (ByVal portaddress As Integer) As
Integer
Public portinput As Integer
Dim ana As Integer, movex As Single, movey As Single
Private i As Integer, j As Integer, k As Integer, L As Integer
Private a(210, 340) As Integer, code(135 * 240) As String, con As Long, hexcode(135 * 240) As
String
Dim opsetW As Integer, opset As Integer, opsetH As Integer
Dim xais, yais As Integer
Dim mychoice() As String

Private Sub cmdClear_Click()
ClearAll
End Sub

Private Sub Combo1_Click()
Preview.Enabled = True
End Sub

Private Sub Combo2_Change()
Preview.Enabled = True
End Sub
```

```
Private Sub comload_Click()
Dim abc As String
Static i As Integer
If i <= 3 Then
   cmd.CancelError = True
   cmd.Filter = _
      "JPEGs (*.jpg)|*.jpg;*.jpeg|" & _
      "Bitmaps (*.bmp)|*.bmp|" & _
      "GIFs (*.gif)|*.gif|" & _
      "Graphic Files|*.bmp;*.gif;*.jpg;*.jpeg|" & _
      "All Files (*.*)|*.*"
   cmd.Flags = &H4
   On Error Resume Next
   cmd.ShowOpen
   If Err.Number = cdlCancel Then
      Screen.MousePointer = vbDefault
      Exit Sub
   End If
   abc = cmd.FileName
   On Error GoTo 0
ima2(i).Picture = LoadPicture(abc)
i = i + 1
Else
i = 0
End If
End Sub

Private Sub Command1_Click()
End Sub
```

```
Private Sub ext_Click()
Dim exitprogram As Variant
Out portoutput, &HFF
For i = 0 To 3
ima2(i).Enabled = False
Next i
exitprogram = MsgBox("You sure exit program", vbInformation + vbOKCancel +
vbDefaultButton2, "Punching Plastic 2006")
If exitprogram = vbOK Then
pic1.Cls
pic2.Cls
pic3.Cls
MsgBox "Good Bye Project", vbInformation, "Punching Plastic 2006"
End
Else
Call Form_Load
End If
End Sub

Private Sub Finnit_Click(Index As Integer)
Dim exitprogram As Variant
exitprogram = MsgBox("You sure exit program", vbInformation + vbOKCancel +
vbDefaultButton2, "Punching Plastic 2006")
If exitprogram = vbOK Then
pic1.Cls
pic2.Cls
pic3.Cls
Out portoutput, &HFF
For i = 0 To 3
ima2(i).Enabled = False
```

```
Next i
 MsgBox "Good Bye Project", vbInformation, "Punching Plastic 2006"
End
End If
End Sub

Private Sub Form_Load()
Form1.Show
Form1.Refresh
portoutput = &H378
portinput = &H379
Timer1.Enabled = False
Timer2.Enabled = False
punch.Enabled = False
pic2.DrawWidth = 1
pic2.Visible = False
pic3.Visible = False

Dim aaaa  As Integer
ReDim mychoice(52)
For aaaa = 13 To 52
 mychoice(aaaa) = "" & aaaa
 Next aaaa
 Combo1.Clear
 For aaaa = 13 To 52
 Combo1.AddItem mychoice(aaaa)
 Next aaaa

 Dim bbbb As Integer
ReDim mychoice(80)
```

```
For bbbb = 13 To 80
 mychoice(bbbb) = "" & bbbb
 Next bbbb
 Combo2.Clear
 For bbbb = 13 To 80
 Combo2.AddItem mychoice(bbbb)
 Next bbbb
End Sub

Private Sub Form_Unload(Cancel As Integer)
Dim exitprogram As Variant
Out portoutput, &HFF
For i = 0 To 3
ima2(i).Enabled = False
Next i
exitprogram = MsgBox("You sure exit program", vbInformation + vbOKCancel +
vbDefaultButton2, "Punching Plastic 2006")
If exitprogram = vbOK Then
pic1.Cls
pic2.Cls
pic3.Cls
MsgBox "Good Bye Project", vbInformation, "Punching Plastic 2006"
End
Else
Call Form_Load
End If
End Sub

Private Sub hsc_Change()
If pic1.Visible = True Then
```

```
pic1.Left = 120 - hsc.value
End If
If pic2.Visible = True Then
pic2.Left = 120 - hsc.value
End If
If pic3.Visible = True Then
pic3.Left = 120 - hsc.value
End If
End Sub

Private Sub Open_Click()
'Load picture
Dim abc As String
Static i As Integer
If i <= 3 Then
  cmd.CancelError = True
   cmd.Filter = _
      "JPEGs (*.jpg)|*.jpg;*.jpeg|" & _
      "Bitmaps (*.bmp)|*.bmp|" & _
      "GIFs (*.gif)|*.gif|" & _
      "Graphic Files|*.bmp;*.gif;*.jpg;*.jpeg|" & _
      "All Files (*.*)|*.*"
  cmd.Flags = &H4
   On Error Resume Next
   cmd.ShowOpen
   If Err.Number = cdlCancel Then
      Screen.MousePointer = vbDefault
      Exit Sub
   End If
   abe = cmd.FileName
```

```
    On Error GoTo 0
ima2(i).Picture = LoadPicture(abc)
i = i + 1
Else
i = 0
End If

End Sub

Private Sub pic1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
movex = X
movey = Y
End If
End Sub

Private Sub pic1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
pic1.Move X - movex + pic1.Left, Y - movey + pic1.Top
End If
End Sub

Private Sub pic2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
movex = X
movey = Y
End If
End Sub

Private Sub pic2_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
If Button = 1 Then
pic2.Move X - movex + pic2.Left, Y - movey + pic2.Top
End If
End Sub

Private Sub pic3_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
movex = X
movey = Y
End If
End Sub

Private Sub pic3_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
pic3.Move X - movex + pic3.Left, Y - movey + pic3.Top
End If
End Sub

Private Sub port_Click()
vsc.Visible = False
hsc.Visible = False
com1.Visible = False
If pic3.Visible = False Then
MsgBox "Please insert picture", vbInformation, "Warnning"
Exit Sub
End If
Screen.MousePointer = 11
'Call coppy
Dim scanwidth As Integer
Dim scanheight As Integer
```

```
Dim part As Integer
Dim ii As Integer, jj As Integer, iii As Integer, jjj As Integer, iiii As Integer, jjjj As Integer
Dim cord As Integer, adum1 As Integer, adum2 As Integer, adum3 As Integer, adum4 As Integer
Dim n As String, s As String, R As Integer, g As Integer, b As Integer
Dim savecord1 As Long, savecord2 As Long, savecord3 As Long, savecord4 As Long
Dim step As Integer
Dim Z As Integer, Y As Integer
Dim RR As Integer, LL As Integer, aa As Integer, bb As Integer
Dim XX As Integer, YY As String
Dim AAA As Integer
Dim hec1, hec2, hec3, hec4, hec5, hec6, hec7, hec8 As Integer
  con = 1
  RR = 0
  LL = 1
  aa = 0
  bb = 0
  opsetW = 196
  opsetH = 300
  opset = opsetW / 4
  pic2.ScaleHeight = opsetH
  pic2.ScaleWidth = opsetW
  scanwidth = Int(pic3.Width / opsetW)
  scanheight = Int(pic3.Height / opsetH)
  part = Int(pic3.Width / 4)
      step = 6
      Y = 0
  For L = 0 To pic3.Height - 100 Step scanheight
      Y = Y + 1
      Z = 0
    For k = 0 To pic3.Width / 4 Step scanwidth
```

```
        Z = Z + 1
        'clear varible
        a(Z, Y) = 0
        a(Z + opset, Y) = 0
        a(Z + 2 * opset, Y) = 0
        a(Z + 3 * opset, Y) = 0
      ' part 1
      For j = 1 To scanheight Step step
        For i = 1 To scanwidth Step step
          If a(Z - 1, Y) = 1 Or a(Z, Y - 1) = 1 Or a(Z - 1, Y - 1) = 1 Or a(Z + 1, Y - 1) = 1 Then
            Exit For
          End If
          s = pic3.Point(i + k, j + L)
          If s < 13158600 Then
            a(Z, Y) = 1
            adum1 = 1
          Exit For
          End If
        Next i
      Next j
      ' part 2
    For j = 1 To scanheight Step step
        For i = 1 To scanwidth Step step
          If a(Z + opset - 1, Y) = 1 Or a(Z + opset, Y - 1) = 1 Or a(Z + opset - 1, Y - 1) = 1 Or
a(Z + opset + 1, Y - 1) = 1 Then
            Exit For
          End If
          s = pic3.Point(i + part + k, j + L)
          If s < 13158600 Then
            a(Z + opset, Y) = 1
```

```
            adum2 = 1
          Exit For
          End If
        Next i
      Next j
      ' part 3
      For j = 1 To scanheight Step step
        For i = 1 To scanwidth Step step
          If a(Z + 2 * opset - 1, Y) = 1 Or a(Z + 2 * opset, Y - 1) = 1 Or a(Z + 2 * opset - 1, Y -
1) = 1 Or a(Z + 2 * opset + 1, Y - 1) = 1 Then
            Exit For
          End If
          s = pic3.Point(i + 2 * part + k, j + L)
          If s < 13158600 Then
            a(Z + 2 * opset, Y) = 1
            adum3 = 1
          Exit For
          End If
        Next i
      Next j
      ' part 4
      For j = 1 To scanheight Step step
        For i = 1 To scanwidth Step step
          If Z > 45 Then
          Exit For
          End If
          If a(Z + 3 * opset - 1, Y) = 1 Or a(Z + 3 * opset, Y - 1) = 1 Or a(Z + 3 * opset - 1, Y -
1) = 1 Or a(Z + 3 * opset + 1, Y - 1) = 1 Then
            Exit For
          End If
```

```
            s = pic3.Point(i + 3 * part + k, j + L)
            If s < 13158600 Then
               a(Z + 3 * opset, Y) = 1
               'adum4 = 1
            Exit For
            End If
         Next i
           If Z > 47 Then '''''''''''''''''''''''''''''''''''''
            Exit For
            End If
      Next j
   Next k
 Next L
 pic2.Cls
pic3.Visible = False
pic2.Visible = True
For j = 1 To opsetH Step 1
   For i = 1 To opsetW Step 1
    If a(i, j) = 1 Then
        pic2.PSet (i, j), RGB(255, 0, 0)
        pic2.Refresh
     End If
    Next i
Next j
  ' code---------------------------------------------code
code(con) = Str(1) + Str(1) + Str(1) + Str(1) + Str(0) + Str(0) + Str(0) + Str(0) 'start
 con = con + 1
For j = 1 To opsetH Step 1
     aa = 0
     For i = 1 To opset Step 1
```

```
XX = a(i, j) + a(i + opset, j) + a(i + 2 * opset, j) + a(i + 3 * opset, j)
  If XX = 0 Then
    aa = aa + 1
  Else
  Exit For
  End If
Next i

If aa = opset Then
  code(con) = Str(0) + Str(0) + Str(1) + Str(1) + Str(0) + Str(0) + Str(0) + Str(0)
  con = con + 1
   GoTo AAA
End If
If aa < opset Then
  For i = 1 To opset Step 1
    YY = Str(1 - a(i, j)) + Str(1 - a(i + opset, j)) + Str(1 - a(i + 2 * opset, j)) + Str(I - a(i +
3 * opset, j))
    bb = bb + 1
    If bb < opset Then
      code(con) = Str(0) + Str(0) + Str(RR) + Str(LL) + YY
      con = con + 1
    End If
    If bb = opset Then
      code(con) = Str(0) + Str(0) + Str(RR) + Str(LL) + YY
      con = con + 1
      code(con) = Str(0) + Str(0) + Str(1) + Str(1) + Str(0) + Str(0) + Str(0) + Str(0)
      con = con + 1
      code(con) = Str(0) + Str(0) + Str(LL) + Str(RR) + Str(1) + Str(1) + Str(1) + Str(1)
      con = con + 1
    End If
```

```
            Next i
            bb = 0
            LL = 0
            RR = 1
        End If
        '''''''''''''''''''''''''''''''''''''''''''''
BBB:
        j = j + 1
        aa = 0
        For i = opset To 1 Step -1
            XX = a(i, j) + a(i + opset, j) + a(i + 2 * opset, j) + a(i + 3 * opset, j)
                If XX = 0 Then
                    aa = aa + 1
                Else
                Exit For
                End If
        Next i
        If aa = opset Then
            code(con) = Str(0) + Str(0) + Str(1) + Str(1) + Str(0) + Str(0) + Str(0) + Str(0)
            con = con + 1
            If j > opsetH Then
            Exit For
            End If
            GoTo BBB
        End If
        If aa < opset Then
            For i = opset To 1 Step -1
                YY = Str(1 - a(i, j)) + Str(1 - a(i + opset, j)) + Str(1 - a(i + 2 * opset, j)) + Str(1 - a(i +
3 * opset, j))
                bb = bb + 1
```

```
            If bb < opset Then
               code(con) = Str(0) + Str(0) + Str(RR) + Str(LL) + YY
               con = con + 1
            End If
            If bb = opset Then
               code(con) = Str(0) + Str(0) + Str(RR) + Str(LL) + YY
               con = con + 1
               code(con) = Str(0) + Str(0) + Str(1) + Str(1) + Str(0) + Str(0) + Str(0) + Str(0)
               con = con + 1
               code(con) = Str(0) + Str(0) + Str(LL) + Str(RR) + Str(1) + Str(1) + Str(1) + Str(1)
               con = con + 1
            End If
         Next i
         bb = 0
         LL = 1
         RR = 0
      End If
AAA:
   Next j
   code(con) = Str(0) + Str(0) + Str(0) + Str(0) + Str(1) + Str(1) + Str(1) + Str(1) 'finit
   For i = 1 To con
      hec1 = Val("&H" & Left(code(i), 2)) * 2 ^ 7
      hec2 = Val("&H" & Mid(code(i), 3, 2)) * 2 ^ 6
      hec3 = Val("&H" & Mid(code(i), 5, 2)) * 2 ^ 5
      hec4 = Val("&H" & Mid(code(i), 7, 2)) * 2 ^ 4
      hec5 = Val("&H" & Mid(code(i), 9, 2)) * 2 ^ 3
      hec6 = Val("&H" & Mid(code(i), 11, 2)) * 2 ^ 2
      hec7 = Val("&H" & Mid(code(i), 13, 2)) * 2
      hec8 = Val("&H" & Mid(code(i), 15, 2))
      s = hec1 + hec2 + hec3 + hec4 + hec5 + hec6 + hec7 + hec8
```

```
        hexcode(i) = ("&H" & Hex(s))
      Next i
     ' Print hexcode(con)
    Screen.MousePointer = 1
punch.Enabled = True
Form1.Show
End Sub

Private Sub ima2_Click(Index As Integer)
pic2.Visible = False
If ima2(Index) = 0 Then
MsgBox "Please insert picture", vbInformation, "warnning"
Exit Sub
End If
pic1.Cls
pic1.Picture = ima2(Index)
pic1.Visible = True
If pic1.Height > Picture1.Height Then
vsc.Visible = True
com1.Visible = True
vsc.Max = pic1.Height - Picture1.Height + 200
End If
If pic1.Width > Picture1.Width Then
hsc.Visible = True
com1.Visible = True
hsc.Max = pic1.Width - Picture1.Width + 200
End If
ZN.Enabled = True
ZU.Enabled = True
Combo1.Enabled = True
```

```
Combo2.Enabled = True
Preview.Enabled = True
End Sub

Private Sub PPn_Click()
 If pic1.Visible = False Then
MsgBox "Please insert picture", vbInformation, "Warnning"
Exit Sub
End If
pic3.Picture = pic1.Picture
pic3.PaintPicture pic3, 0, 0, pic3.Width, pic3.Height
port.Enabled = True
pic1.Visible = False
vsc.Visible = False
hsc.Visible = False
com1.Visible = False
ZN.Enabled = False
ZU.Enabled = False
pic3.Visible = True
End Sub

Private Sub Preview_Click()
pic3.BackColor = vbWhite
 Dim resp As Boolean
resp = Checkinput(Combo1.Text, Combo2.Text)
If resp = False Then
    MsgBox "คุณกำหนดความก้างความยาวถูกต้อง", vbOKOnly + vbExclamation, "ผิดพลาด"
    ClearAll
    Exit Sub
End If
```

```
 xais = (5000 / 52) * Val(Combo1.Text)
 yais = (7885 / 80) * Val(Combo2.Text)
pic3.PaintPicture pic1, 80, 80, xais, yais
port.Enabled = True
pic1.Visible = False
vsc.Visible = False
hsc.Visible = False
com1.Visible = False
ZN.Enabled = False
ZU.Enabled = False
pic3.Visible = True
End Sub

Private Sub Pun_Click()
Timer1.Enabled = True
    Out &H37A, &H1
    mybar.Visible = True
End Sub

Private Sub Timer1_Timer()
ana = Inp(portinput)
If ana = 119 Then
   Timer2.Enabled = True
   End If
End Sub


Private Sub Timer2_Timer()
Static na As Integer
Out &H37A, &H80
```

```
Out portoutput, &H0
ana = Inp(portinput)
lab1.Visible = True
lab2.Caption = Round((na / con) * 100) & " %"
mybar.value = Round((na / con) * 100)
  If ana = 111 Then
     Out &H37A, &H1
     na = na + 1
     Out portoutput, hexcode(na)
     Timer1.Enabled = True
     Timer2.Enabled = False
  End If
  If ana = 255 Then
  Out portoutput, &HFF
  Out &H37A, &H1
  MsgBox "Mission Complete", vbInformation, "Punching Plastic 2006"
  na = 0
  Timer1.Enabled = False
  Timer2.Enabled = False
   mybar.Visible = False
   lab1.Visible = False
  End If
End Sub

Private Sub punch_Click()
  Timer1.Enabled = True
    Out &H37A, &H1
    mybar.Visible = True
End Sub
```

```
Private Sub vsc_Change()
  If pic1.Visible = True Then
  pic1.Top = 120 - vsc.value
 End If
  If pic2.Visible = True Then
  pic2.Top = 120 - vsc.value
 End If
  If pic3.Visible = True Then
  pic3.Top = 120 - vsc.value
 End If
End Sub

Private Sub ZN_Click()
 If pic1.Visible = True Then
 If pic1.Height < Picture1.Height / 4 Or pic1.Width < Picture1.Width / 4 Then
Exit Sub
End If
  pic1.Height = pic1.Height / 1.2
  pic1.Width = pic1.Width / 1.2
  pic1.PaintPicture pic1.Picture, 0, 0, pic1.ScaleWidth, pic1.ScaleHeight
If pic1.Height <= Picture1.Height Then
  vsc.Visible = False
  com1.Visible = False
  vsc.value = 0
  pic1.Top = 120
End If
If vsc.Visible = False And hsc.Visible = True Then
  com1.Visible = True
 End If
If vsc.Visible = True And hsc.Visible = True Then
```

```
    com1.Visible = True
  End If

 If pic1.Width <= Picture1.Width Then
    hsc.Visible = False
    comI.Visible = False
    hsc.value = 0
    pic1.Left = 120
  End 1f
  End 1f
 End Sub
 Private Sub ZU_Click()
 If pic1.Visible = True Then
 If pic1.Height > Picture1.Height * 4 Or pic1.Width > Picture1.Width * 4 Then
 Exit Sub
  End If
    pic1.Height = pic1.Height * 1.2
    pic1.Width = pic1.Width * 1.2
    pic1.PaintPicture pic1.Picture, 0, 0, pic1.ScaleWidth, pic1.ScaleHeight
If pic1.Height > Picture1.Height Then
  vsc.Visible = True
  com1.Visible = True
   vsc.Max = pic1.Height - Picture1.Height + 200
End If
  If pic1.Width > Picture1.Width Then
   hsc.Visible = True
   hsc.Max = pic1.Width - Picture1.Width + 200
End If
 If pic1.Height <= Picture1.Height Then
   vsc.Visible = False
```

```
End If
If vsc.Visible = False And hsc.Visible = True Then
  Com1.Visible = True
  com1.Visible = True
End If
If vsc.Visible = True And hsc.Visible = True Then
End If
If pic1.Width <= Picture1.Width Then
  hsc.Visible = False
  com1.Visible = False
End If
End If
End Sub
Sub ClearAll()
    Combo1.Text = ""
    Combo2.Text = ""
  MsgBox "ให้กำหนดขนาดใหม่", vbOKOnly + vbInformation, ""
End Sub

Function Checkinput(Width As String, Height As String) As Boolean
  If IsNumeric(Width) And IsNumeric(Height) Then
    If Width >= 13 And Height >= 13 Then
      If Width <= 52 And Height <= 100 Then
      Checkinput = True
      Exit Function
      End If
    End If
  End If
  Checkinput = False
End Function
```

ภาคผนวก ข
ภาษาซีซอร์สโค้ด

```
#include<reg51rx.h>
 sbit  x = P3^2;//limit switch
 sbit  y = P3^3;//chek plastic
 sbit  fin = P3^4;//finit
 sbit  w = P3^5;//conferm data
 sbit  z = P3^6;//give data
 sbit  zz = P3^7;// data form com
 code unsigned char step[]= {0x00,0x03,0x06,0x0c,0x09,0x30,0x60,0xc0,0x90};
 /*********************************************/
 void deley (int time)
        {
              do
              { time--;}
              while (time>0);
        }
        /*********************************************/
 void motor1_r (int k)
        {
              char i;
              int j;
              i=0;
              for(j=0;j<k;j++)
              {
                    for(i=1;i<5;i++)
                          {
                          //deley(0x500);
                          P2 = step[i];
```

```
                    deley(0x250);
                    }

            }

        }

/*************************************************/
void motor1_1 (int k)
        {
            char i;
            int j;
            i=4;
            for(j=0;j<k;j++)
            {
                for(i=4;i>0;i--)
                    {
                    //deley(0x500);
                    P2 = step[i];
                    deley(0x250);
                    }

            }
        }
/*************************************************/
void motor2 (int k)
        {
            char i;
            int j;
            i=5;
```

```
                for(j=0;j<k;j++)
                {
                        for(i=5;i<9;i++)
                                {
                                //deley(0x500);
                                P2 = step[i];
                                deley(0x250);
                                }
                }
        }
void reset ()
{
 x = 1;
 while(x==1)
                {
                        motor1_l(1);
                        if(x == 0)
                                { x = 0;}
                }
motor1_r(14);
P2 = 0;
 }
//*****************************************************//
//*****************************************************/
void punching (int A)
        {
                int star = 1;
                int B;
                int countreset;
        if (A == 0xcf)
```

```
        {
        P1 = 0;
        motor2(3);
        P0 = 0xff;
        countreset = 0;
        }
    if (A >= 0xe0 && A <= 0xef) //right
        {
        B = A - 0xe0;
        if(B!=0)
        {
        P1 = B;
        deley(3000);
        P1 = 0;
        deley(200);
        }
        motor1_r(3);
        P0 = 0xff;
        countreset = 0;
        }
if (A >= 0xd0 && A <= 0xdf)//left
        {
        B = A - 0xd0;
        if(B!=0)
        {
        P1 = B;
        deley(3000);
        P1 = 0;
        deley(200);
        }
```

```
            motor1_l(3);
            P0 = 0xff;
            countreset++;
            if (countreset == 50)
            {
            reset();
            countreset = 0;
            }
}
    if (A == 0xf0)//finit
            {
            P1 = 0;
            reset();
            while ( star==1)
            {
            z = 0;

                while(zz == 0)
                {
                        z = 1;
                        zz = 1;
                        fin = 0;
                        star = 0;
                        deley(50);

                }
            }
            fin = 1;
            deley(50);
            }
```

```
            if (A == 0x0f)//finit
            {
            reset();
            }
        else
        {
        P0 = 0xff;
        fin = 1;
        }

        }

/*****************************************************/
 void main ()
        {
                P0 = 0xff;
                P1 = 0x00;
                y = 1;
                x = 1;
                zz = 1;
                z = 1;
                w = 1;
                fin = 1;
                reset();

                while (1)
                {
                z = 0;
                P2 = 0;
```

```
deley(10);
        while (y == 0)
        {
        motor2(1);
        }
        while(zz == 0)
        {
                z = 1;
                zz = 1;
                w = 0;
                deley(100);
                w = 1;
                if(P0 != 0xff)
                {
                punching(P0);
                }
        }
}
}
```

ภาคผนวก ค
แบบชิ้นส่วนของเครื่องเจาะแผ่นพลาสติก