การควบคุมเครื่องเจาะอัตโนมัติโดยไมโครคอนโทรลเลอร์

Controlling of drilling machine by microcontroller

โดย

นายนพดล นภาอารักษ์

นายนราชัย ถนัดค้า

นายนรินทร แก้วศรีงาม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

การควบคุมเครื่องเจาะอัตโนมัติโดยไมโครคอนโทรลเลอร์

Controlling of drilling machine by microcontroller

โดย

นายนพดล นภาอารักษ์ 50010735

นายนราชัย ถนัดค้า 50010760

นายนรินทร แก้วศรีงาม 50010765

อาจารย์ที่ปรึกษา

รศ.ดร. สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวะกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

ปริญญานิพนธ์ ปีการศึกษา 2553

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมเครื่องเจาะอัตโนมัติโดยไมโครคอนโทรลเลอร์

(Controlling of drilling machine by microcontroller)

ผู้จัดทำ

1. นายนพดล นภาอารักษ์ 50010735
2. นายนราชัย ถนัดค้า 50010760
3. นายนรินทร แก้วศรีงาม 50010765

.....

(รศ.ดร. สุรพันธ์ เอื้อไพบูลย์)

อาจารย์ที่ปรึกษา

กิตติกรรมประกาศ

รายงานและชิ้นงานเครื่องแกะสลักควบคุมด้วยไมโครคอนโทรลเลอร์ที่จัดทำขึ้นมานี้สำเร็จลุล่วงไปได้ด้วยดีเพราะได้รับการสนับสนุน ความช่วยเหลือและคำปรึกษาที่ดีจาก รศ.ดร.สุรพันธ์ เอื้อไพบูลย์ ที่ให้คำแนะนำรวมทั้งเอื้อเฟื้อสถานที่และอุปกรณ์ในการทำงานมาโดยตลอดปีการศึกษา รวมทั้งขอบคุณแรงใจจากคุณพ่อ คุณแม่ที่คอยให้กำลังใจในการทำงานตลอดมา

การควบคุมเครื่องเจาะอัตโนมัติโดยไมโครคอนโทรลเลอร์

นายนพดล นภาอารักษ์ 50010735
นายนราชัย ถนัดค้า 50010760
นายนรินทร แก้วศรีงาม 50010765
รศ.ดร. สุรพันธ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2553

บทคัดย่อ

เครื่องเจาะควบคุมด้วยไมโครคอนโทรลเลอร์ ที่ได้ประดิษฐ์ขึ้นนี้ มีระบบการทำงานโดยการจำค่าของพิกัดต่างๆที่เราต้องการจะเจาะแล้วให้ไมโครคอนโทรเลอร์สั่งเจาะอัตโนมัติโดยใช้ไมโครคอนโทรเลอร์ในการป้อนสัญญาณพัลส์ให้กับตัวควบคุมสเตปปิ้งมอเตอร์และสั่งให้เครื่องเจาะอัตโนมัติที่มีการเคลื่อนที่ในระนาบ 3 มิติ

Controlling of drilling machine by microcontroller

Mr. Noppadon Napa-arak ID. 50010735
Mr.Narachai Tanadka ID. 50010760
Mr.Narintorn Kawsringam ID. 50010765
Assist. Prof. Dr. Surapan Airphaiboon Advisor
Education Year 2010

Abstract

Drilling Machine controlled by Microcontroller which invented ,There is the system to work by remembering the coordinates that we want to drill, then Microcontroller order to drill automation using Microcontroller for sent the pulse signals to control stepping motor and ordered the Drilling Machine controlled by Microcontroller move in three dimensions.

สารบัญ

สารบัญรูป

หน้า

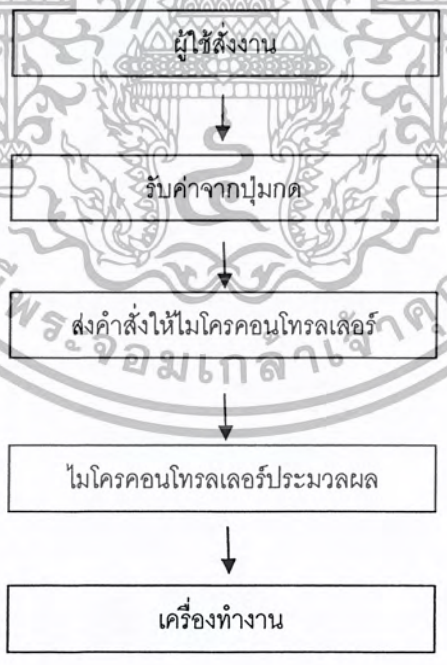
สารบัญตาราง

หน้า

บทที่ 1
การควบคุมเครื่องเจาะอัตโนมัติโดยไมโครคอนโทรลเลอร์
Controlling of drilling machine by microcontroller

ในปัจจุบันนี้มีการนำเทคนิคการเจาะและเทคนิคการสั่งการควบคุมการเคลื่อนที่มาใช้ทั้งในภาคการวิจัยและการใช้งานทางภาคอุตสาหกรรม ทางผู้จัดทำโครงงานนี้จึงมีความคิดที่จะศึกษาและและผลิตชิ้นงานโดยใช้ความรู้ที่กล่าวมาข้างต้น จึงได้จัดทำโครงงานเครื่องเจาะอัตโนมัติควบคุมด้วยไมโครคอนโทรลเลอร์ซึ่งเป็นโครงงานที่จัดทำขึ้นมาเพื่อนำความรู้ด้านอิเล็กทรอนิกส์มาประยุกต์ให้เกิดการประดิษฐ์พัฒนาอุปกรณ์ขึ้นโดยสามารถนำมาใช้งานได้จริง รวมถึงเรียนรู้ถึงการแก้ปัญหาต่างๆที่เกิดขึ้นช่วงระหว่างทำโครงงาน

ลักษณะการทำงานโดยรวมของเครื่องเจาะอัตโนมัติควบคุมด้วยไมโครคอนโทรลเลอร์นั้น ในขั้นแรกจะรับข้อมูลจากผู้ใช้งานผ่านปุ่มกดซึ่งง่ายต่อการใช้งานไปยังไมโครคอนโทรลเลอร์เพื่อประมวลผลการเคลื่อนที่อีกครั้งหนึ่ง หลังจากนั้นไมโครคอนโทรลเลอร์จะสั่งงานและควบคุมการทำงานเครื่องเจาะอัตโนมัติเคลื่อนที่ตามข้อมูลที่ได้รับจากผู้ใช้งานในขั้นแรกนั้นเอง ในการทำงานของเครื่องแกะสลักนี้จะใช้สเต็ปปิ้งมอเตอร์เข้ามาควบคุมการเคลื่อนที่ของหัวเจาะในระนาบ 3 มิติ ซึ่งจะเป็นการเคลื่อนที่ไปตามจุดบนระนาบ 2 มิติ



รูปที่ 1.1 บล็อกไดอะแกรมของโครงงานชิ้นนี้

CNC คืออะไร

CNC เป็นคำย่อของ Computer Numerical Control แปลว่าการควบคุมเชิงตัวเลขด้วย คอมพิวเตอร์ เป็นการใช้คอมพิวเตอร์ควบคุมการทำงานของเครื่องจักรกลต่าง ๆ เช่น เครื่องกัดซีเอ็นซี เครื่องกลึงซีเอ็นซี เครื่องเจียระไน เครื่อง EDM ฯลฯ ซึ่งสามารถทำให้ผลิตชิ้นส่วนได้รวดเร็วถูกต้อง และ เที่ยงตรง

เครื่องจักรซีเอ็นซีแต่ละแบบแต่ละรุ่นจะมีลักษณะเฉพาะ และการประยุกต์ใช้งานที่ต่างกันออกไปแต่เครื่องจักรกลซีเอ็นซีทั้งหมดมีข้อดีเหมือน ๆ กันคือ

1.เครื่องจักรกลซีเอ็นซีทุกเครื่องได้รับการปรับปรุงให้มีการทำงานอัตโนมัติทำให้ลดความวุ่นวายของผู้ควบคุมเครื่องจักรในการผลิตชิ้นงาน เครื่องจักรซีเอ็นซีหลายเครื่องสามารถทำงานโดยที่ผู้ควบคุมไม่ต้องคอยนั่งเฝ้าในระหว่างวัฏจักรการทำงานของเครื่อง (Machining cycle) และผู้ควบคุมสามารถไปทำงานอย่างอื่นได้ สิ่งนี้ทำให้ผู้ใช้เครื่องจักรซีเอ็นซีได้ประโยชน์หลายอย่างรวมทั้งลดความเหนื่อยล้าของผู้ปฏิบัติงาน ความผิดพลาดที่เกิดขึ้นจากคนมีน้อยมากมีความคงเส้นคงวาในการผลิตและสามารถทำนายเวลาในการผลิตแต่ละชิ้นได้

2.เทคโนโลยีซีเอ็นซีคือความคงเส้นคงวาและความถูกต้องแม่นยำของชิ้นงาน ซึ่งหมายความว่าเมื่อโปรแกรมที่เขียนทำงานอย่างถูกต้องแล้ว การผลิตชิ้นส่วน 2 ชิ้น 10 ชิ้น หรือ 1000 ชิ้นให้เหมือนกันทุกประการสามารถทำได้อย่างง่ายดายด้วยความสม่ำเสมอ

3.ความยืดหยุ่นในการทำงานเนื่องจากเครื่องจักรกลเหล่านี้ทำงานตามโปรแกรมการทำงานที่ต่างกันก็ง่ายเหมือนกับการโหลดโปรแกรมที่ต่างกัน เมื่อโปรแกรมประมวลผลและทำการผลิตชิ้นงานแล้ว เราสามารถเรียกโปรแกรมนั้นกลับมาใช้ใหม่ในครั้งต่อไปเมื่อต้องทำงานชิ้นนั้นอีก

บทที่ 2

หลักการและทฤษฎี

2.1ไมโครคอนโทรลเลอร์

Propeller มัลติคอร์ไมโครคอนโทรลเลอร์นวัตกรรมใหม่ของไมโครคอนโทรลเลอร์ 32 บิต

Propeller หรือ โพรเพลเลอร์ คือชื่อของไมโครคอนโทรลเลอร์ 32 บิตอนุกรมใหม่จาก **Parallax** (www.parallax.com) ที่มีสถาปัตยกรรมที่พิเศษคือ **มีซีพียูภายใน 8 ตัวหรือ 8 ค็อก (cog)** ที่สามารถทำงานแยกจากกันอย่างอิสระหรือร่วมกันทำงานก็ได้ นับเป็นแนวคิดใหม่ที่ร่วมสมัย และนับเป็นการปฏิวัติวงการไมโครคอนโทรลเลอร์ 32 บิตครั้งสำคัญ

การพัฒนาโปรแกรมสำหรับโพรเพลเลอร์ทำได้ด้วยโปรแกรมภาษาใหม่ที่เรียกว่า สปิน(spin) และภาษาแอสเซมบลี โดยภาษาสปินนั้นเป็นภาษาสูงที่มีการทำงานแบบออบเจ็กต์(high-level object-based language)

2.1.1 คุณสมบัติเด่นของโพรเพลเลอร์

- ประกอบด้วย 8ซีพียู หรือเรียกว่า 8 ค็อก ที่สามารถทำงานได้พร้อมกันอย่างเป็นอิสระ โดยมีการควบคุมการใช้ทรัพยากรร่วมกันผ่านทางตัวเชื่อมโยงกลางหรือ central hub
- มีความเร็วในการทำงานสูงและด้วยการทำงานที่เป็นอิสระของแต่ละค็อกทำให้สามารถรองรับการตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นในระบบได้อย่างเร็วเพียงพอ จึงไม่ต้องใช้กระบวนการอินเตอร์รัปต์ช่วยทำให้การเขียนโปรแกรมเพื่อรองรับการทำงานในแต่ละเหตุการณ์ลดความซับซ้อนลงได้อย่างมาก
- มีการใช้สัญญาณนาฬิกาของระบบร่วมกัน ทำให้สามารถอ้างอิงค่าเวลาหลักเดียวกันได้ทำให้การทำงานของแต่ละค็อกมีจังหวะที่สอดคล้องกัน
- ภาษาสปินซึ่งมีลักษณะเป็นโปรแกรมภาษาสูงแบบออบเจ็กต์ได้รับการออกแบบให้ง่ายต่อการเรียนรู้ และสามารถรองรับการทำงานของโพรเพลเลอร์ได้อย่างมีประสิทธิภาพสูงสุด
- ภาษาแอสเซมบลีของโพรเพลเลอร์ได้จัดเตรียมคำสั่งที่ใช้ในการตรวจสอบเงื่อนไข และมีตัวแปรที่ใช้ในการตรวจสอบการทำงานอย่างสมบูรณ์ ทั้งยังสามารถรองรับการทำงานในลักษณะที่ต้องมีการตัดสินใจพร้อมๆ กันหลาย

เงื่อนไขด้วย พร้อมกันนั้นยังมีการคำนึงถึงการลดสัญญาณรบกวนและความเพี้ยนของสัญญาณที่เกิดขึ้นจากการประมวลผลคำสั่งและตัวคำสั่งเองมีรูปแบบการทำงานที่ตรงไปตรงมา ชัดเจน ส่งผลให้ผู้พัฒนาโปรแกรมสามารถลดเวลาในการเขียนโปรแกรมลงได้อย่างมาก

- แต่ละค็อกจะประกอบด้วยตัวประมวลผลหรือโปรเซสเซอร์ที่มีการทำงานอย่างเป็นอิสระมีแรม 2 กิโลไบต์ ที่เมื่อกำหนดให้ทำงานเป็นรีจิสเตอร์ 32 บิต จะได้ทั้งสิ้นถึง 512 ตัว มีโมดูลตัวนับความสามารถสูงที่ทำงานร่วมกับเฟสล็อกลูป ทำให้แต่ละค็อกทำงานได้เร็วถึง 80 MHz มีวงจรกำเนิดสัญญาณและส่วนควบคุมพอร์ตอินพุตเอาต์พุตที่เป็นอิสระ
- สัญญาณนาฬิกาของระบบมาได้จาก 3 แหล่งคือ วงจรออสซิลเลเตอร์ RC ภายในเลือกได้ระหว่าง 12 หรือ 20 MHz จากแหล่งกำเนิดสัญญาณนาฬิกาภายนอก และจากการทวีคูณความถี่ของคริสตอลด้วยวงจรเฟสล็อกลูป โดยปกติแล้วจะเลือกใช้คริสตอล 5 MHz แล้วเลือก PLL×16 ทำให้ได้สัญญาณนาฬิกาของระบบที่มีความถี่ 80 MHz ในขณะที่ส่วนเชื่อมโยงกลางจะทำงานด้วยความถี่ที่ลดลงครึ่งหนึ่งของสัญญาณนาฬิกาหลัก
- มีขาพอร์ตอินพุตเอาต์รวม 32 ขา โดยกำหนดให้ใช้ 2 ขาสำหรับต่อหน่วยความจำอีอีพรอมสำหรับเก็บโปรแกรมของผู้ใช้งาน และอีก 2 ขาสำหรับการดาวน์โหลดโปรแกรม สามารถขับกระแสซิงค์และซอร์สสูงสุด 40mA ต่อขา
- โพรเพลเลอร์ใช้หน่วยความจำอีอีพรอมภายนอกในการเก็บโปรแกรมของผู้ใช้งาน ทำให้อายุการใช้งานของตัวชิปจึงไม่ขึ้นกับจำนวนรอบของการลบและโปรแกรมใหม่ของหน่วยความจำโปรแกรม
- การดาวน์โหลดโปรแกรมทำได้ง่ายมากเพียงต่อเข้ากับวงจรเชื่อมต่อพอร์ตอนุกรม อาทิวงจรของไอซี MAX3232 หรือต่อผ่านชิปแปลงสัญญาณพอร์ต USB เป็นพอร์ตอนุกรมอย่าง FT232RL ไม่ต้องการเครื่องโปรแกรมใดๆ เพิ่มเติม
- ด้วยความเร็วในการทำงานที่สูง และมีส่วนกำเนิดสัญญาณภาพที่มากถึง 8 ชุด ทำให้เหมาะสมอย่างมากในการนำโพรเพลเลอร์ไปใช้ในการกำเนิดสัญญาณภาพไม่ว่าจะแสดงผลด้วยจอโทรทัศน์ด้วยสัญญาณวิดีโอ หรือแสดงผลด้วยจอ VGA ด้วยสัญญาณแม่สีแสง นั่นคือพื้นฐานหลักในการสร้างเครื่องเล่นวิดีโอเกม และการสร้างระบบนำเสนอ (presentation) ภาพกราฟิกด้วยไมโครคอนโทรลเลอร์เพียงตัวเดียว
- สามารถเชื่อมต่อกับคีย์บอร์ดและเมาส์ได้ และเมื่อรวมกับความสามารถการสร้างสัญญาณภาพได้ จึงสามารถนำโพรเพลเลอร์ไปสร้างเครื่องคอมพิวเตอร์ขนาดเล็กแบบใช้จอโทรทัศน์เป็นตัวแสดงผลได้
- ใช้ไฟเลี้ยงในย่าน 2.7 ถึง 3.6 V กระแสไฟฟ้าสูงสุดเมื่อขับโหลดเต็มที่คือ 300 mA
- มีตัวถังให้เลือกใช้ 3 แบบคือ DIP 40 ขา, LQFP 44 ขา และ QFN 44 ขา

- มีซอฟต์แวร์ Propeller IDE ที่มีประสิทธิภาพสูง สามารถสร้างระบบไฟล์ที่มีรูปการต่อวงจรได้ ทำให้สามารถนำโค้ดไปถ่ายทอดต่อได้สะดวก และช่วยให้การเรียนรู้ทำได้อย่างสมบูรณ์ครบวงจรและที่สำคัญซอฟต์แวร์แจกฟรี สามารถดาวโหลดเวอร์ชั่นล่าสุดได้ที่ www.parallax.com

2.1.2 คุณสมบัติทางเทคนิคของโพรเพลเลอร์

- เป็นไมโครคอนโทรลเลอร์ที่ภายในประกอบไปด้วยโปรเซสเซอร์ขนาด 32 บิตถึง 8 ชุด
- ทำงานที่แรงดัน 3.3 โวลต์ (2.7V ถึง 3.6V)
- ขาพอร์ตสามารถจ่ายกระแสซิงก์และซอร์สได้ 40 mA ต่อขา และ 100mA ต่อพอรีต (8 ขา)
- มีวงจรกำเนิดสัญญาณนาฬิกาภายใน 12 MHz หรือ 20 kHz เลือกกำหนดค่าได้
- ทำงานด้วยสัญญาณนาฬิกาจากภายนอกความถี่ตั้งแต่ดีซีถึง 80 MHz
- สามารถใช้คริสตอล 4 MHz ถึง 8 MHz ร่วมกับตัวคูณสัญญาณนาฬิกาความถี่สูงสุด 80 MHz
- สามารถเชื่อมต่อคริสตอลภายนอกได้ โดยไม่จำเป็นต้องเชื่อมต่อตัวเก็บประจุ
- หน่วยความจำของทั้งระบบแบ่งเป็นหน่วยความจำอีอีพรอม 32 กิโลไบต์ (KB) และหน่วยความจำ แรม 32 KB
- ในแต่ละโปรเซสเซอร์มีหน่วยจำแรม ตัวละ 2 KB
- การจัดการหน่วยความจำเป็นแบบ 32 บิต
- จำนวนพอร์ตอินพุตเอาต์ 32 ขา

2.1.3 หลักการทำงานของโพรเพลเลอร์

รูปที่ 1-2 แสดงให้เห็นบล็อกไดอะแกรมการทำงานภายในของโพรเพลเลอร์ ซึ่งประกอบด้วยโปรเซสเซอร์ที่ทำงานแยกกันอิสระถึง 8 ชุด โดยจะเรียกโปรเซสเซอร์เหล่านี้ว่า Cog หมายเลข 0 ถึง 7

2.1.3.1 ค็อก (Cogs)

ในแต่ละค็อกจะประกอบไปด้วยหน่วยความจำแรม 2 KB โดยกำหนดเป็นหน่วยความจำแบบ 32 บิต จำนวน 512 ตัว นอกจากนี้ภายในโปรเซสเซอร์แต่ละตัวยังมีโมดูลเคาน์เตอร์แบบพิเศษพร้อมเฟสล็อกลูป 2 ตัว โมดูลสร้างสัญญาณวิดีโอ รีจิสเตอร์พอร์ตเอาต์พุต รีจิสเตอร์กำหนดทิศทางของพอร์ตอินพุต และรีจิสเตอร์ตัวอื่นๆ ซึ่งไม่ได้แสดงให้เห็นในบล็อกไดอะแกรม

ค็อกทั้ง 8 ตัวทำงานด้วยวงจรกำเนิดสัญญาณนาฬิกาหลัก ซึ่งค็อกแต่ละตัวจะอ้างอิงการทำงานกันได้ด้วยสัญญาณนาฬิกา และจะเริ่มต้นทำงานพร้อมกันและใช้ทรัพยากรด้วยกัน ค็อกแต่ละตัวสามารถสั่งให้ทำงานหรือหยุดทำงาน ได้ในขั้นตอนการทำงานของโปรแกรม และสามารถควบคุมให้แต่ละค็อกทำงานไปพร้อมๆ กันได้ โดยจะทำงานเป็นอิสระหรือ เชื่อมโยงถึงกันได้ผ่านหน่วยความจำแรมหลัก(Main RAM) ซึ่งแยกไปต่างหาก

หน่วยความจำภายในค็อกแต่ละตัว เรียกว่า ค็อกแรม (Cog RAM) โดยค็อกแรมจะแบ่งหน่วยความจำเป็นรีจิสเตอร์ขนาด 32 บิตจำนวน 512 ตัวสามารถใช้งานได้อย่างอิสระ ยกเว้น รีจิสเตอร์ 16 ตำแหน่งสุดท้ายซึ่งสงวนไว้สำหรับรีจิสเตอร์ฟังก์ชั่น พิเศษ เช่น รีจิสเตอร์เคาน์เตอร์ , รีจิสเตอร์พอร์ตอินพุตเอาต์พุต เป็นต้น

2.1.3.2 ฮับ(Hub) : ส่วนเชื่อมโยงหลัก

ฮับทำหน้าที่จัดระเบียบการทำงานของระบบทั้งหมด โดยจะยอมให้ค็อกทีละตัวเท่านั้นที่จะติดต่อกับทรัพยากรหลักของระบบ โดยจะหมุนเวียนติดต่อกับค็อกตั้งแต่หมาย 0 ถึง 7 แล้วกลับไปที่หมายเลข 0 ใหม่เป็นลักษณะวนรอบ ส่วนของฮับและระบบบัสของมันทำงานด้วยความเร็วครึ่งหนึ่งของสัญญาณนาฬิกาของทั้งระบบ ทำให้ค็อก 1 ตัวจะติดต่อทุกๆ 16 ไซเกิลของสัญญาณนาฬิกา และใช้เวลา 7 ไซเกิลเพื่อเอ็กซิคิวซ์คำสั่ง ดังนั้น ฮับจะติดต่อกับค็อกตัวใดตัวหนึ่งได้ อาจใช้คาบเวลาเพียง 7 ไซเกิล หรือนานถึง 22 ไซเกิล เนื่องจากจะต้องรอให้ฮับวนจนครบรอบ

VX-Propeller

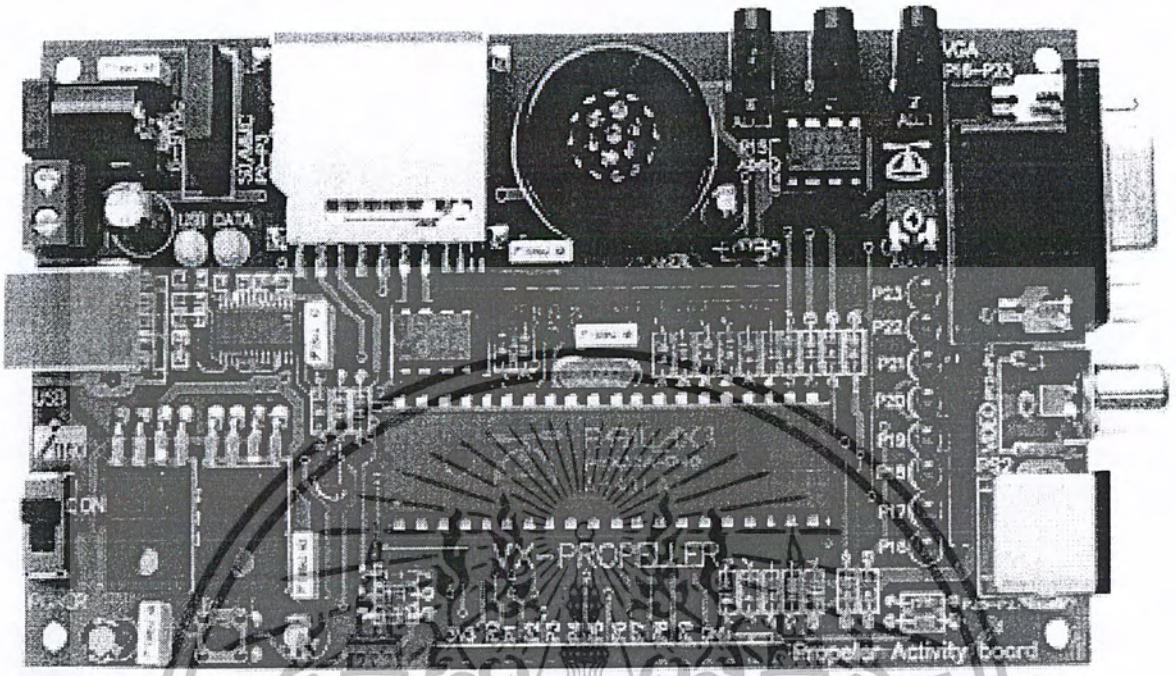
Propeller microcontroller Education Board

บอร์ดเรียนรู้และใช้งานไมโครคอนโทรลเลอร์ Propeller

VX-Propeller บนบอร์ดได้เลือกใช้ชิปโพรเพลเลอร์เบอร์ P8X32A-D40 ต่อร่วมกับชิป FT232RL เพื่อรองรับการดาวน์โหลดโปรแกรมผ่านพอร์ต USB มีการต่อชิปโพรเพลเลอร์เข้ากับอุปกรณ์อินพุตเอาต์พุตตั้งแต่ระดับพื้นฐานอย่าง LED และลำโพง จนถึงขั้นก้าวหน้าอย่าง SD การ์ด, จอภาพ VGA, จอโทรทัศน์ผ่านช่อง AV, คีย์บอร์ดหรือเมาส์แบบ PS2 และวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลแบบอนุกรม นอกจากนี้ยังมีการจัดสรรขาพอร์ตอิสระมากถึง 14 ขา ให้สามารถนำไปใช้เชื่อมต่อกับอุปกรณ์ภายนอกได้อย่างมากเพียงพอ

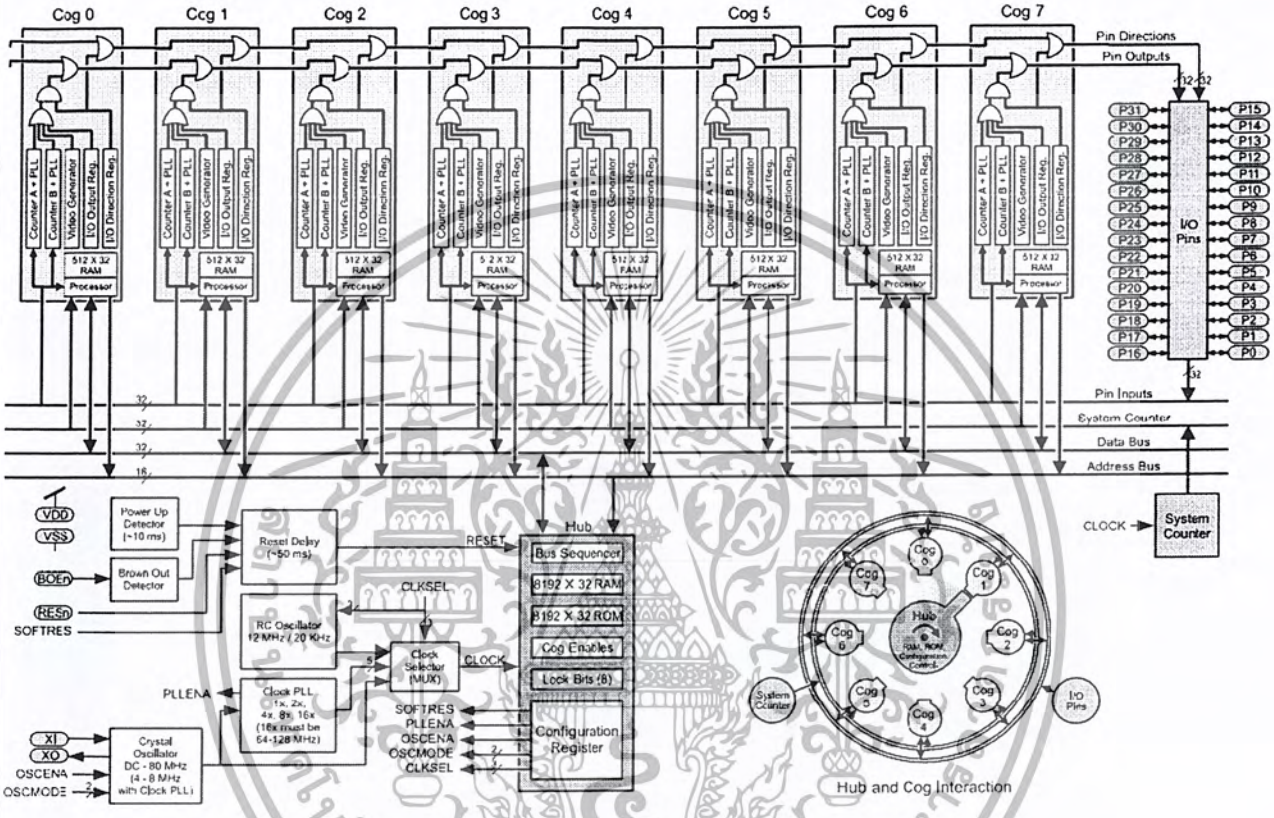
2.1.4 คุณสมบัติ VX-Propeller

- ใช้ชิปโพรเพลเลอร์เบอร์ P8X32-D40 พร้อมคริสตอล 5MHz สามารถรันด้วยความถี่สัญญาณนาฬิกาสูงสุด 80MHz ด้วยวงจรเฟสล็อกลูป (PLLx16) ภายในตัวชิป มีความเร็วในการประมวลผล 160MIPS จาก 8 ค็อก แต่ละค็อกมีความเร็ว 20MIPS
- หน่วยความจำอีอีพรอม 32 กิโลไบต์จากไอซีอีอีพรอมภายนอกเบอร์ 24LC256
- หน่วยความจำแรมภายใน 32 กิโลไบต์
- มีจุดต่อพอร์ตสำหรับติดต่ออุปกรณ์ภายนอก 14 ขา
- ดาวน์โหลดข้อมูลและโปรแกรมผ่านทางพอร์ต USB
- ต้องการแรงดันไฟตรงในย่าน +6 ถึง +12V บนบอร์ดมีวงจรควบคุมไฟเลี้ยงคงที่ที่ +5V และ +3.3V
- LED 8 ช่อง ทำงานด้วยลอจิก "1" (ต่อร่วมกับวงจรเชื่อมต่อจอภาพ VGA)
- มีวงจรเชื่อมต่อจอภาพ VGA (ต่อร่วมกับ LED)
- มีวงจรเชื่อมต่อจอภาพด้วยสัญญาณวิดีโอ รองรับทั้งระบบ PAL และ NTSC
- มีลำโพง 8W 100mW สำหรับขับเสียง
- มีวงจรเชื่อมต่อผ่านพอร์ต PS2 สามารถเชื่อมต่อกับคีย์บอร์ดและเมาส์ได้
- มีวงจรเชื่อมต่อซ็อกเก็ตของ SD การ์ด
- เชื่อมต่อไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอล 10 บิต 4 ช่อง แบ่งเป็นจุดต่ออินพุตสัญญาณอนาลอก 3 ช่อง และมีตัวต้านทานปรับค่าได้บนบอร์ดเพื่อทดสอบสัญญาณอนาลอกอีก 1 ช่อง

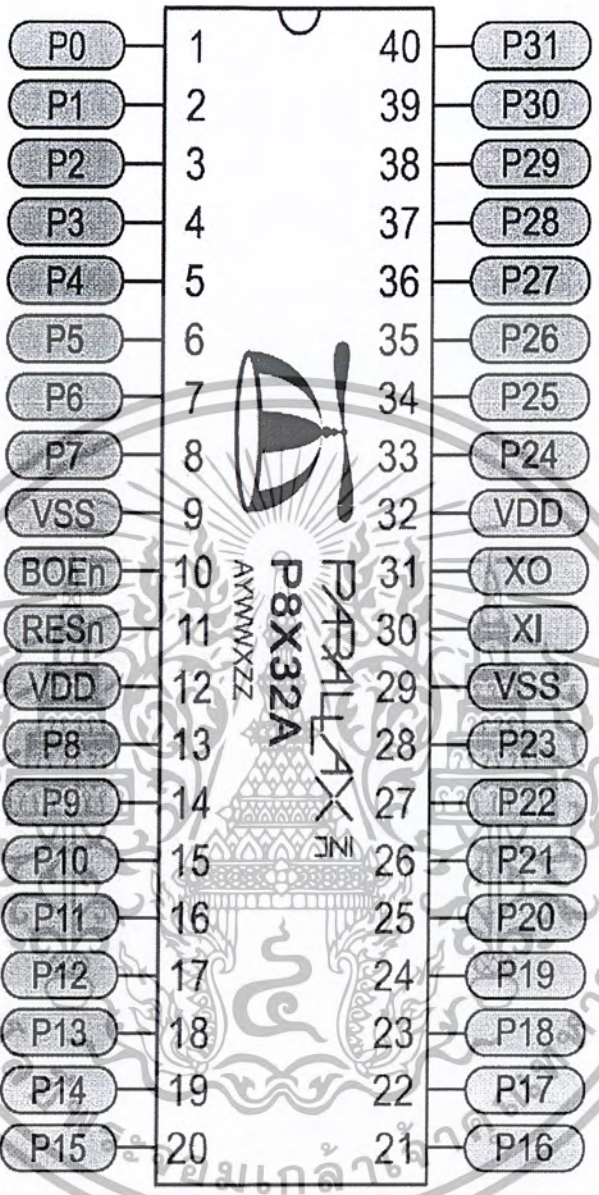


รูปที่ 2.1 บอร์ด VX-Propeller

2.1.5 โครงร้างและบล็อกไดอะแกรมของ MCU P8X32A



รูปที่ 2.2 แสดงบล็อกไดอะแกรมของ Chip P8X32A



รูปที่ 2.3 แสดงโครงสร้างของ Chip P8X32A(DIP 40 PIN)

ตารางที่ 2.1 รายละเอียด PIN

PIN Name	Direction	Description
P0-P31	I/O	เป็น Port I/O โดยมีระดับ Logic อยู่ที่ประมาณครึ่งหนึ่งของ VDD หรือ 1.6 VDC ที่แรงดัน 3.3 VDC ใน 32 Pin นี้จะมีอยู่ด้วยกัน 4 Pin ซึ่งจะถูกกำหนดให้ทำงานในหน้าที่พิเศษหลังจาก Power-up หรือ Reset คือ P28 – I2C SCL ซึ่งจะใช้ต่อไปยัง EEPROM ภายนอก P29 – I2C SDA ซึ่งจะใช้ต่อไปยัง EEPROM ภายนอก P30 – Serial Tx ใช้ต่อสำหรับ Download Code และ ส่งข้อมูล ทางทาง RS232 P31 – Serial Rx ใช้ต่อสำหรับ Download Code และ รับข้อมูล ทางทาง RS232
VDD	---	3.3 V Power (2.7-3.6 VDC)
VSS	---	Ground
BOEn	I	Brown Out Enable(Active Low) จะใช้ต่อไปยัง VDD หรือ VSS ถ้าขานี้เป็น Low ขา RESn จะ กลายเป็น Output แต่จะยังสามารถ Drive Low เพื่อ Reset Chip ได้ ถ้าเป็น high ขา RESn จะ ทำหน้าที่เป็น Input
RESn	I/O	Reset(Active Low) เมื่อเป็น Low ทั้ง Chip และ Cog ทั้งหมด จะถูก Disable PIN I/O จะถูกปล่อยลอย และ Chip จะ Restart ภายในเวลา 50ms หลังจาก Logic ที่ RESn เปลี่ยนจาก Low เป็น High
XI	I	Crystal Input
XO	O	Crystal Output

2.1.6 การทำงานของวงจร

บอร์ด VX-Propeller เป็นบอร์ดที่ดึงเอาความสามารถของไมโครคอนโทรลเลอร์ Protelller มาใช้อย่างเต็มที่ โดยแสดงรายละเอียดการเชื่อมต่อวงจรทั้งหมดรูปที่ 2.2 สามารถอธิบายการทำงานในส่วนต่างๆ ได้ดังนี้

ภาคจ่ายไฟจะรับแรงดันอินพุตจากแหล่งจ่ายไฟตรงสองแหล่งด้วยกันคือ จากจุดต่อแจ็คอะแดปเตอร์และจุดต่อเทอมินอลบล็อก สำหรับจุดต่อแจ็กอะแดปเตอร์จะมีไดโอดบริดจ์ต่อไว้เพื่อป้องกันการกลับขั่วของวงจร ก่อนส่งให้กับไอซี 78R05 เพื่อเรกูเลตแรงดันให้เหลือ 5 โวทต์ ใช้ไฟเลี้ยงวงจรในส่วนแรงดัน 5 โวทต์เช่นจุดต่อ VGA และจุดต่อคีบอร์ด จากนั้นส่งเข้าไปยังเรกูเลเตอร์เบอร์ 78R33 เพื่อเรกูเรตให้เหลือ 3.3 โวทต์ เป็นไฟเลี้ยงให้กับไมโครคอนโทรลเลอร์และอุปกรณ์ต่อพ่วงอื่นๆ

ภาคสื่อสารข้อมูลกับคอมพิวเตอร์และดาวน์โหลดโปรแกรม เชื่อมต่อคอมพิวเตอร์ผ่านพอร์ต USB โดยใช้ไอซี FT23RL ทำหน้าที่แปลงรูปแบบการสื่อสารข้อมูลจาก USB ให้เป็นการสื่อสารแบบอนุกรม ซึ่ง Propeller ต้องการ

สายสัญญาณในการเชื่อมต่อ 3 สาย ประกอบด้วย Tx, Rx และ DTR โดยขาสัญญาณ DTR จะต่อเข้ากับขา Reset ของ Propeller เพื่อรีเซตตัวมันและเข้าสู่โหมดโปรแกรม แต่เนื่องจากจะต้องป้อนสัญญาณรีเซตด้วยลอจิก "0" จึงจำเป็นต้องต่อทรานซิสเตอร์ Q1 ไว้เพื่อกลับสถานะลอจิกและจะรีเซตเมื่อขา DTR มีลอจิก "1" ไอซี FT23RL เมื่อติดต่อกับคอมพิวเตอร์เป็นที่เรียบร้อยจะแสดงสถานะที่ LED 3 โดย LED 3 (สีน้ำเงิน) จะติดสว่าง และเมื่อมีการส่งข้อมูล LED 4 (สีเหลือง) ก็จะติดสว่าง

ส่วนหัวใจหลักของวงจรของวงจรเป็นชิปโพรเพลเลอร์ทำงานด้วยคริสตอลภายนอก 5 MHz ซึ่งสามารถใช้วงจรคูณสัญญาณภายในให้ได้ความถี่สูงถึง 80 MHz ตัวมันจะมีขาอินพุตเอาท์พุตทั้งหมดทั้งหมด 32 ขา ถูกใช้งานเพื่อสื่อสารข้อมูลกับคอมพิวเตอร์ คือขา Rx และ Tx ใช้เชื่อมต่อกับหน่วยความจำอีอีพรอมเพื่อเก็บข้อมูลโปรแกรมอีก 2 ขา นอกนั้นใช้งานได้เอนกประสงค์

บอร์ด VX-Propeller ได้จัดการเชื่อมต่อขาพอร์ตกับขาอุปกรณ์ภายนอกไว้ดังนี้

P0 ถึง P3 ต่อกับซ็อกเก็ต SD การ์ด

P10 ต่อกับ ไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอลเบอร์ QP410

P11 ต่อกับลำโพง

P12 ถึง P15 ต่อกับวงจรจัดสัญญาณวีดีโอ

P16 ถึง P23 ต่อกับวงจรจัดสัญญาณเพื่อเชื่อมต่อกับมอนิเตอร์ VGA และต่อพ่วงเข้ากับ LED จำนวน 8 ดวง โดยมีสวิตซ์จั๊มเปอร์เลือกการทำงาน

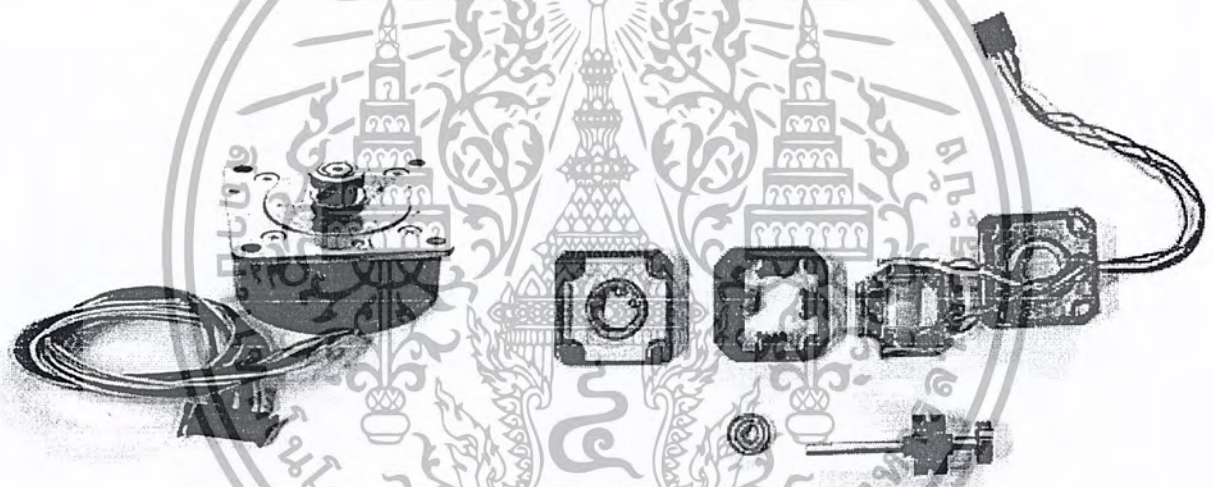
P26 ถึง P27 ต่อกับจุดต่อ PS2 สำหรับต่อคีบอร์ดภายนอก

สำหรับขาที่เหลือต่อไปยังคอนเน็กเตอร์ PORT-1 และ PORT-2 ซึ่งออกแบบตำแหน่งให้สามารถสร้างบอร์ดเสริมในภายหลังได้โดยง่ายไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอล 4 ช่องจะสื่อสารกับโพรเพลเลอร์ในแบบอนุกรมด้วยสายสัญญาณเพียงเส้นเดียว มีการต่อขาอินพุตอนาลอกเข้ากับจุกต่อแบบแจ็คสเตอริโอ 3 ช่อง ดังนั้นหากมีความต้องการการต่อใช้งานจะต้องทำสายสัญญาณที่มีหัวปลั๊กเป็นแบบสเตอริโอ

2.2 สเต็ปปิ้งมอเตอร์ (Stepping Motor)

มอเตอร์เป็นเครื่องมือทางอิเล็กทรอนิกส์ที่เปลี่ยนพลังงานทางไฟฟ้ามาทำให้เกิดสนามแม่เหล็กขึ้นภายในมอเตอร์แล้วไปขับขดลวดตัวนำอาร์เมเจอร์หมุน ทำให้เพลาของมอเตอร์หมุน ซึ่งจะได้พลังงานกลออกไปใช้งาน

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



รูปที่ 2.4 สเต็ปปิ้งมอเตอร์

ข้อดีของสเต็ปปิ้งมอเตอร์เมื่อเปรียบเทียบกับมอเตอร์กระแสตรง

1.การควบคุมไม่ต้องอาศัยตัวตรวจจับการหมุน

2.ไม่ต้องใช้แปรงถ่าน ดังนั้นจึงทำให้ไม่มีส่วนที่จะต้องสึกหรอ และปัญหาของการสปาร์คที่ทำให้เกิดสัญญาณรบกวน

3. การควบคุมโดยทางวงจรดิจิตอลหรือไมโครโพรเซสเซอร์ ทำได้ง่าย และสะดวก

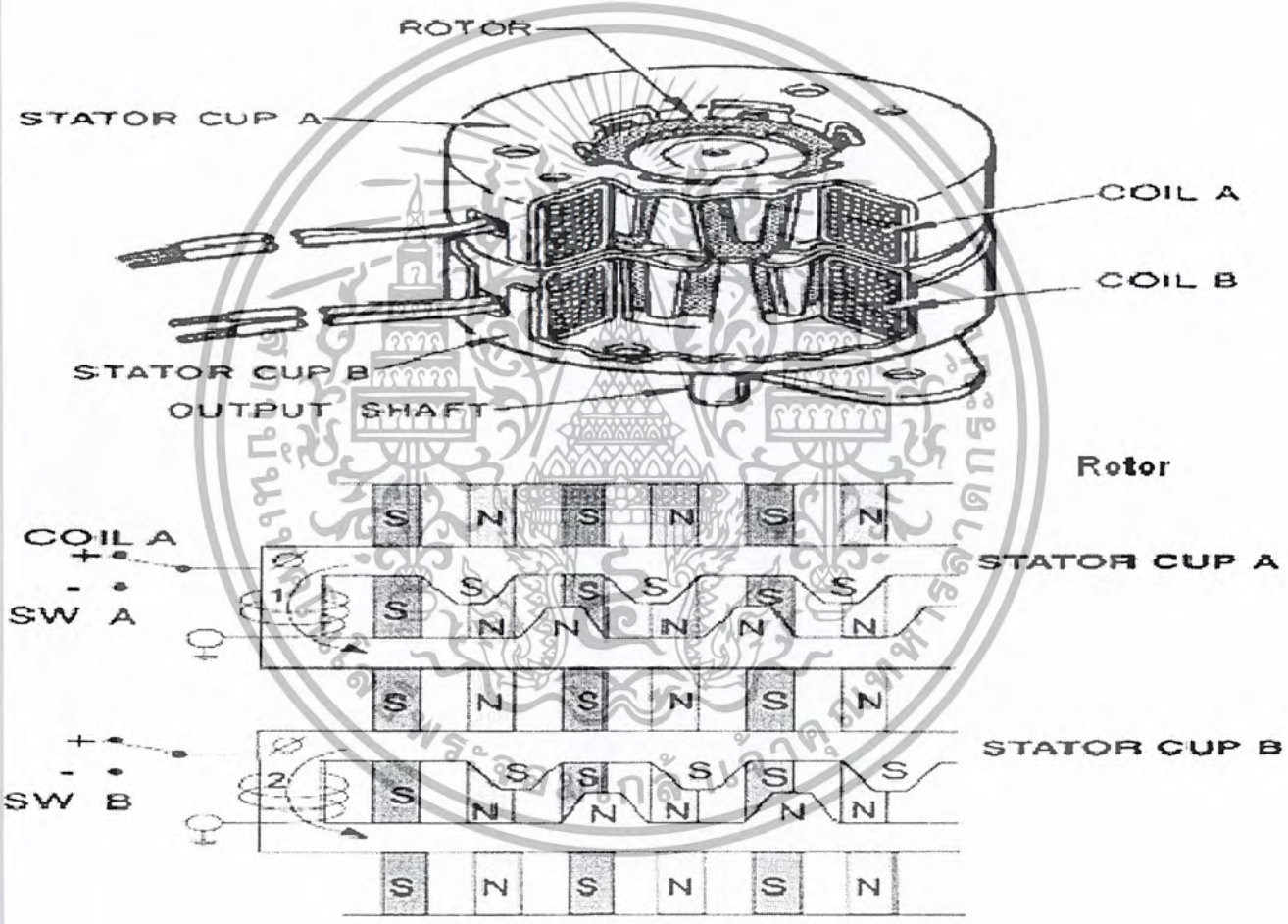
2.2.1 ชนิดของสเต็ปมอเตอร์ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

2.2.1.1 วาริเอเบิลรีลักแตนซ์ (Variable Reluctance : VR) โรเตอร์(Rotor) ทำด้วยเหล็กอ่อนรูปทรงกระบอก และทำเป็นลักษณะฟัน (teeth) สเตเตอร์(Stator)จะมีลวดพันและจะทำเป็นลักษณะของฟันเช่นกันเมื่อจ่ายกระแสไฟฟ้าให้กับขดลวดที่สเตเตอร์จะเกิดเป็นขั้วแม่เหล็กที่ฟันของสเตเตอร์และเหนี่ยวนำให้ฟันของโรเตอร์เกิดเป็นขั้วแม่เหล็กที่มีขั้วตรงกันข้ามกับสเตเตอร์ทำให้ดึงดูดกันเกิดการหมุนของโรเตอร์ขึ้น มอเตอร์ชนิดนี้โดยปกติจะมีขนาด 3 เฟส ในบางครั้งอาจพบถึง 4 เฟส มอเตอร์ชนิดนี้ถ้าไม่จ่ายกระแสไฟฟ้าให้กับขดลวดบนสเตเตอร์ตัวโรเตอร์จะไม่เกิดแรงดึงดูดกับสเตเตอร์ มอเตอร์ชนิดนี้ไม่นิยมนำไปใช้ในงานอุตสาหกรรมแต่จะถูกนำไปใช้กับงานที่มีขนาดเล็ก เช่น Micro-positioning table เป็นต้น เพราะ ไม่มีส่วนที่เป็นแม่เหล็กถาวร ดังนั้นในขณะไม่จ่ายกระแสไฟฟ้าให้กับขดลวดที่สเตเตอร์จึงไม่เกิดแรงดึงดูด วิธีการขับ(Driving)หรือการกระตุ้นเฟส(Phase Excitation) จะทำดังนี้คือ ต่อปลายด้านcommon เข้ากับแหล่งจ่ายไฟขั้วบวก(+) แล้วทำ การสวิทช์ให้ปลายด้าน A , B ,C ต่อลงกราวด์(Ground)ตามลำดับ ทีละปลายแล้วทำเช่นนี้เรื่อยไป แต่ถ้าต้องการให้หมุนกลับก็สวิทช์ย้อนกลับ และในการอธิบายต่อจากนี้ไปจะไม่ขอกล่าวถึงมอเตอร์ชนิดนี้อีก

รูปที่ 2.5 ภาพแสดงโครงสร้างของสเต็ปมอเตอร์แบบวาริเวเบิลรีลักแตนซ์

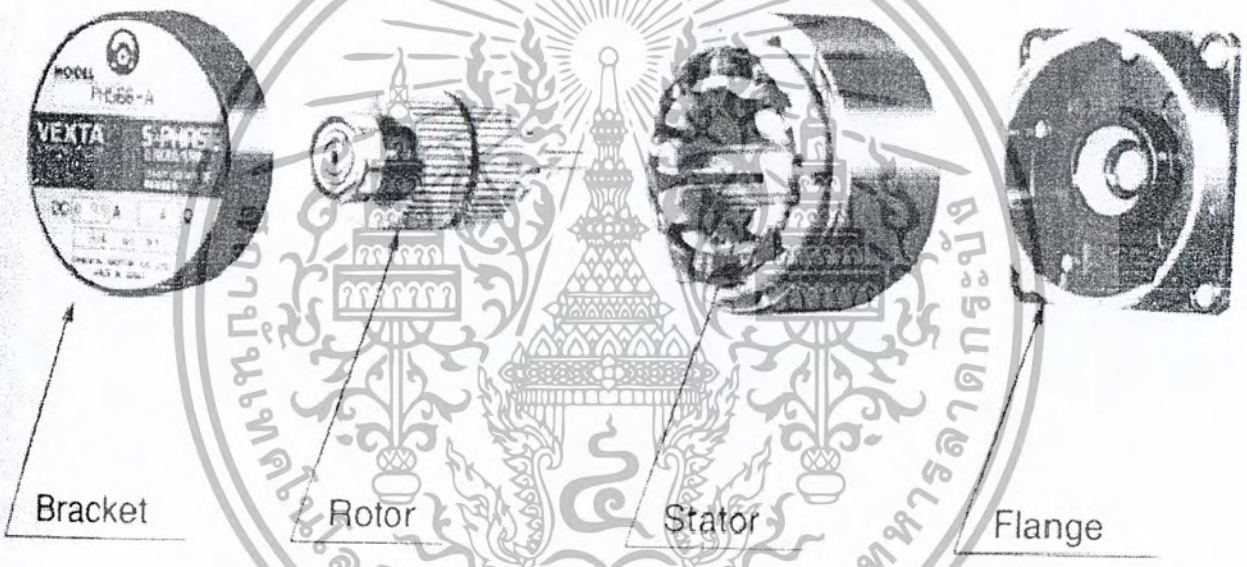
2.2.1.2 แบบแม่เหล็กถาวร (Permanent Magnet : PM)โรเตอร์(Rotor) ทำด้วยแม่เหล็กถาวรรูปทรงกระบอกเรียบ สเตเตอร์(Stator)จะมีขดลวดพัน และก็จะทำเป็นฟัน เมื่อจ่ายกระแสไฟฟ้าให้กับขดลวดที่สเตเตอร์จะเกิดเป็นขั้วแม่เหล็กที่ฟันของสเตเตอร์และจะดึงดูดกับขั้วของแม่เหลกถาวรที่โรเตอร์ทำให้เกิดการหมุนของโรเตอร์ขึ้น มอเตอร์ชนิดนี้โดยจะมีตั้งแต่ขนาด 2 เฟสขึ้นไปมอเตอร์ชนิดนี้ไม่นิยมนำไปใช้ในงานอุตสาหกรรมแต่จะถูก นำไปใช้กับอุปกรณ์

คอมพิวเตอร์เช่นตัวขับวงล้อที่ใช้หมุนเพื่อเลื่อนกระดาษของเครื่องพิมพ์ เป็นต้นเพราะความเร็วต่ำแรงบิดต่ำ และนอกจากนี้ด้วยโครงสร้างของมอเตอร์ชนิดนี้ทำให้มุมที่หมุนไปแต่ละสเต็ป(Step Angle)ไม่ละเอียดเช่น สเต็ปละ 3.6 ,7.5 , 15 ,18 องศาเป็นต้น มอเตอร์ชนิดนี้ถึงไม่จ่ายกระแสไฟฟ้าให้กับขดลวดบนสเตเตอร์ (Stator) ตัวโรเตอร์จะเกิดแรงดึงดูดกับสเตเตอร์ซึ่งเกิดจากอำนาจของแม่เหล็กถาวรที่โรเตอร์ทำให้หมุนได้ยาก จำนวนขั้วแม่เหล็กที่โรเตอร์สามารถนับได้จากจำนวนขั้วแม่เหล็กที่จะเกิดขึ้นจากกระแสไฟฟ้าไหลผ่านขดลวดที่สเตเตอร์ชุดใดชุดหนึ่ง



รูปที่ 2.6 ภาพแสดงโครงสร้างของสเต็ปมอเตอร์แบบแม่เหล็กถาวร

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

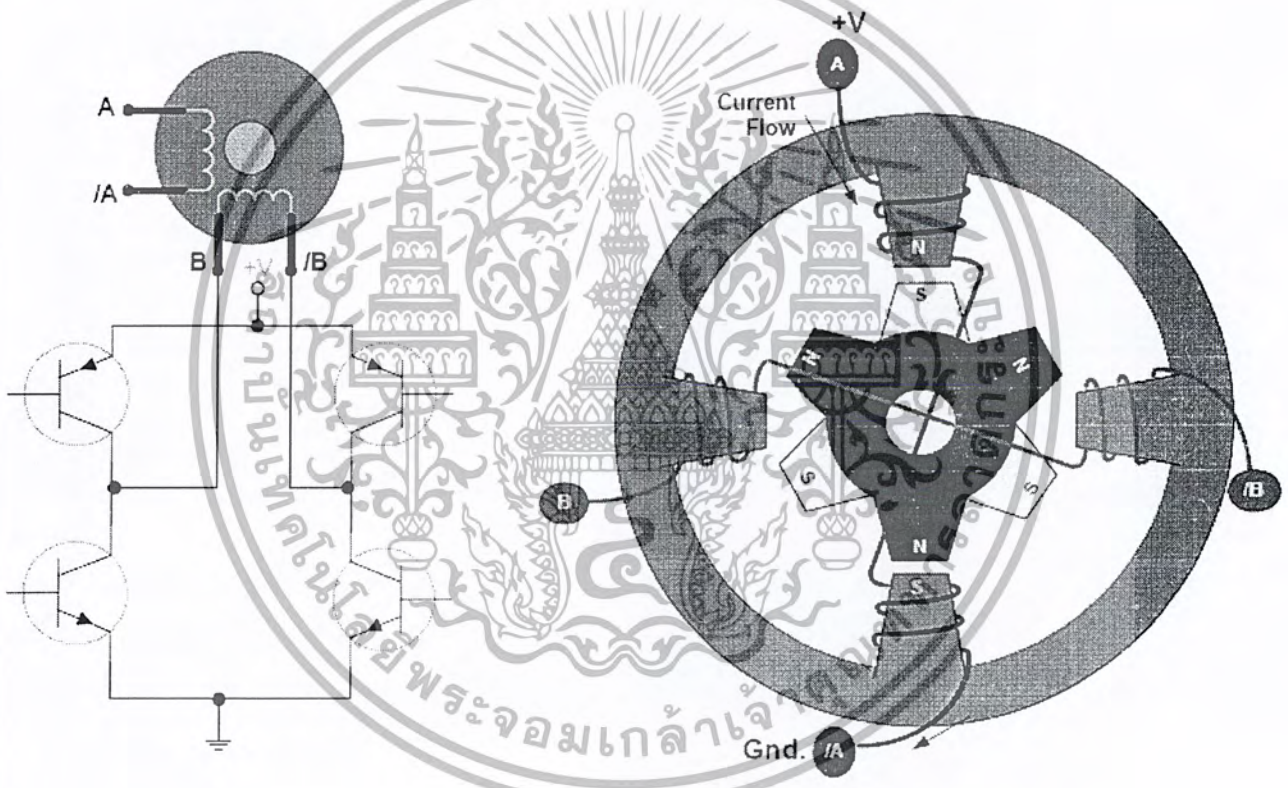


รูปที่ 2.7 สเต็ปเปอร์มอเตอร์ชนิดไฮบริดขนาด 5 เฟส

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

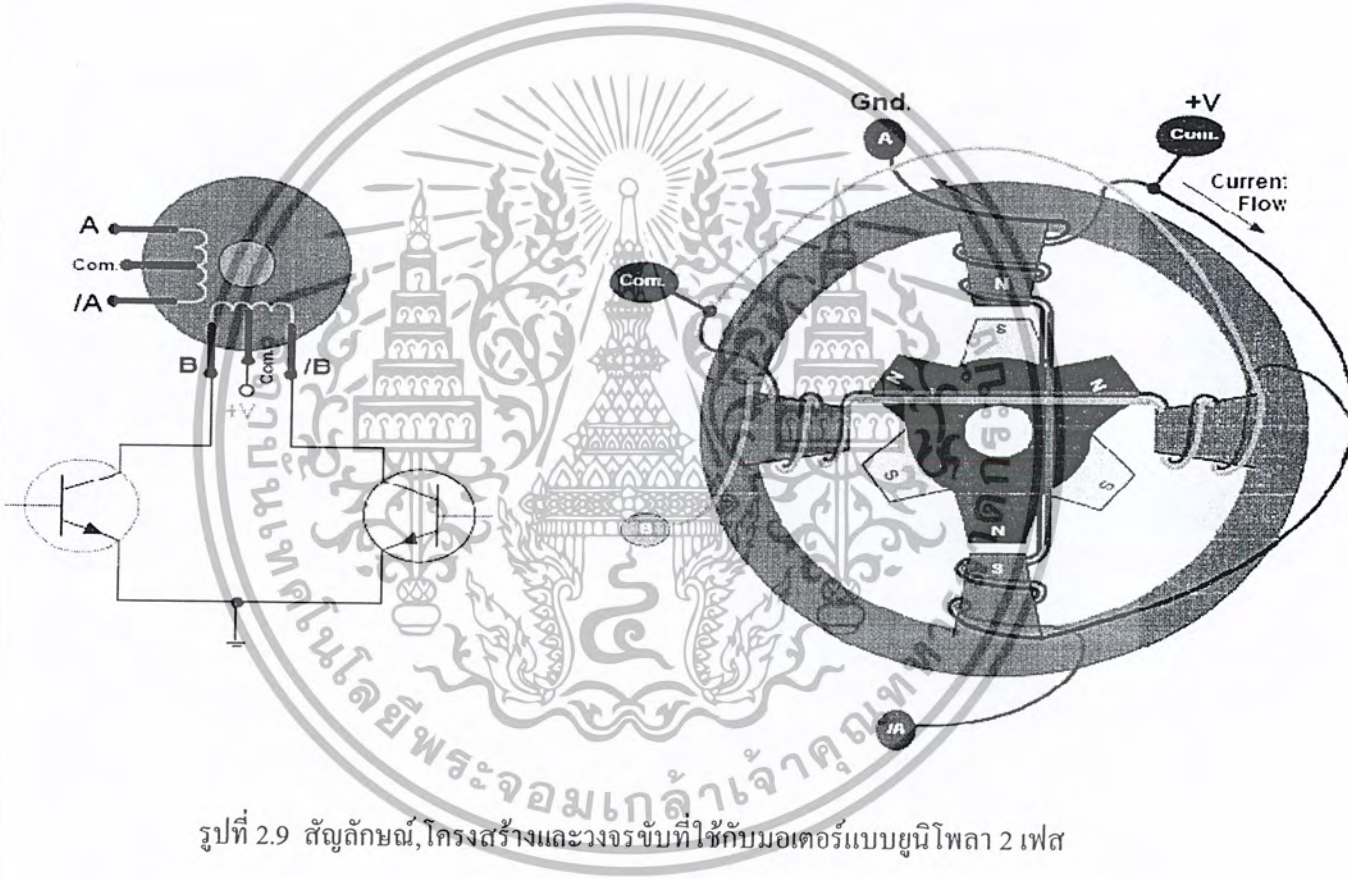
2.2.2 ชนิดของสเต็ปเปอร์มอเตอร์แบ่งตามลักษณะสายที่ใช้ต่อกับวงจรขับ แบ่งออกได้ 2 แบบคือ

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



รูปที่ 2.8 สัญลักษณ์,โครงสร้างและวงจรขับที่ใช้กับมอเตอร์แบบไบโพลาร์ 2 เฟส

2.2.2.2 แบบยูนิโพลา (Unipolar) ขดลวดที่สเตเตอร์แต่ละชุดจะมีจุดรวม การพันขดลวดจะพันในแบบ Bifilar การต่อเข้ากับวงจรขับจะใช้ปลายของขดลวดแต่ละด้านต่อเข้ากับวงจรขับและใช้จุดรวมต่อเข้ากับขั้วบวกของแหล่งจ่ายไฟ การทำ ให้เกิดขั้วแม่เหล็กที่สเตเตอร์ทำได้โดยการจ่ายกระแสไฟให้ไหลจากจุดรวมลงกราวด์มาครบวงจรที่ปลายด้านหนึ่งของขดลวด การเปลี่ยนขั้วแม่เหล็กที่สเตเตอร์ชุดเดียวกันนี้ก็ทำได้โดยเปลี่ยนการจ่ายกระแส ไฟฟ้าจากขดหนึ่งไปยังอีกขดหนึ่งของขดลวดที่พันอยู่บนสเตเตอร์ชุดเดียวกัน ดังนั้นวงจรขับจึงเป็นวงจรสวิทช์ เพื่อทำให้จ่ายกระแสไฟที่ผ่านขดลวดครบวงจรเท่านั้น กรณีเป็นมอเตอร์ 2 เฟสจะมีสายที่ใช้ต่อขับกับวงจร 5 หรือ 6 สาย



รูปที่ 2.9 สัญลักษณ์,โครงสร้างและวงจรขับที่ใช้กับมอเตอร์แบบยูนิโพลา 2 เฟส

2.2.3 วิธีการขับ(Driving)หรือวิธีการกระตุ้นเฟส (Phase Excitation)

การกระตุ้นเฟสของสเต็ปเปอร์มอเตอร์คือ การจ่ายกระแสไฟฟ้าไปยังขดลวดที่สเตเตอร์ของแต่ละเฟสเพื่อทำให้มอเตอร์หมุนนั่นเอง แบ่งออกเป็น 3 วิธีคือ

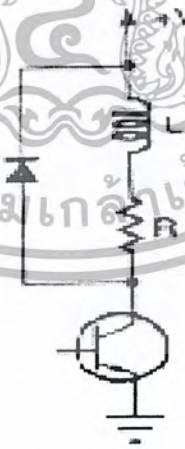
2.2.3.1. การกระตุ้นเฟสแบบเฟสเดียว(One phase excitation) หรือ Half Drive
การกระตุ้นเฟสแบบนี้ทำได้โดยการจ่ายกระแสไฟฟ้าไปยัง ขดลวดบนสเตเตอร์ครั้งละหนึ่งขดเรียงลำดับกันไปดังนี้

2.2.3.2. การกระตุ้นเฟสแบบคู่ Two phase excitation หรือ Full Step
การกระตุ้นเฟสแบบนี้ทำ ได้โดยการจ่ายกระแสไฟฟ้าไปยังขดลวดครั้งละสองขดที่อยู่ใกล้กันบนสเตเตอร์ดังนี้

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

2.2.4 วิธีกำจัดแรงดันที่เกิดจากสนามแม่เหล็กยุบตัว

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

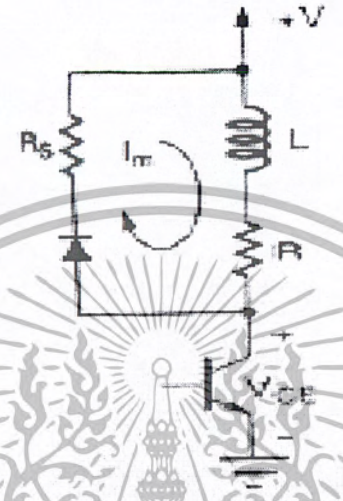


รูปที่ 2.10 วงจรที่ใช้ไดโอดมากำจัด

2.2.4.2. ใช้ความต้านทานร่วมกับไดโอด(Diode + Resistance suppression) คือ

วิธีการนี้จะทำให้เวลาที่ใช้ในการกำจัด Back emf voltage เร็วกว่าการการใช้ไดโอดเพียงอย่างเดียว

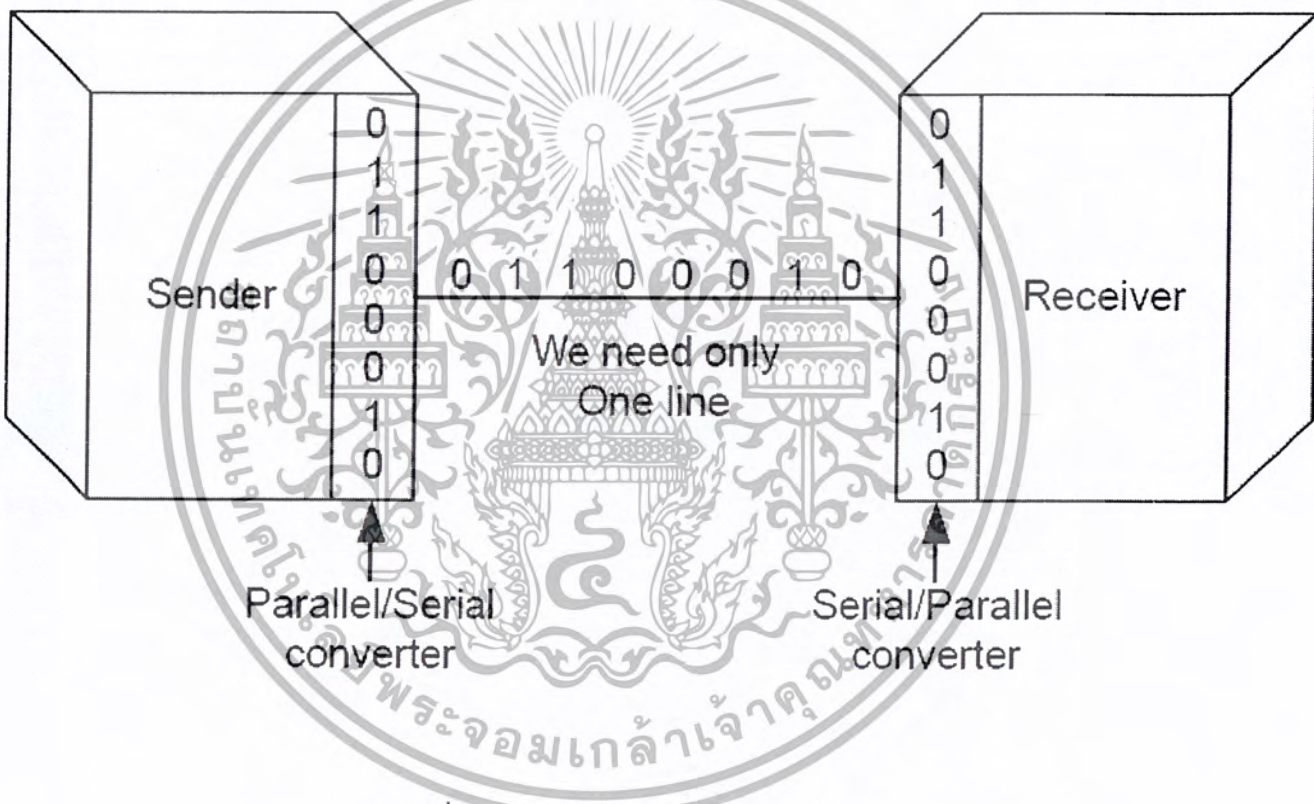
การหาค่าความต้านทานหาได้ดังนี้ Rs(max)=[R*Vce(max)/V]-1



รูปที่ 2.11 วงจรที่ใช้ความต้านทานร่วมกับไดโอด

2.3 การสื่อสารข้อมูลแบบอนุกรม(Serial Transmission)

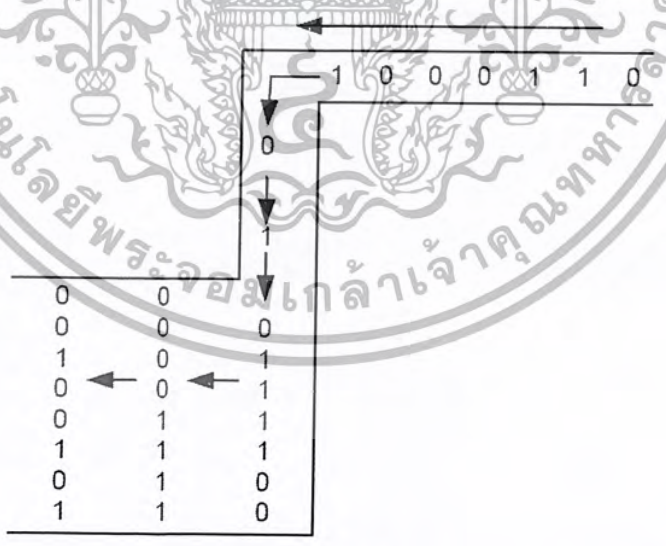
วิธีการสื่อสารข้อมูลแบบอนุกรมนั้น สัญญาณจะทยอยส่งไปตามสายสื่อสารเพียงเส้นเดียวซึ่งแสดงได้ดังรูปที่ 2.18 จะเห็นได้ว่าบิตจะทยอยส่งออกมาทีละบิตจากต้นทางไปยังปลายทาง โดยปลายทางจะทำการรวบรวมสัญญาณหรือบิตที่ทยอยส่งมาจนครบ 8 บิต หรือ 1 ไบท์ เพื่อนำไปใช้งานต่อไป



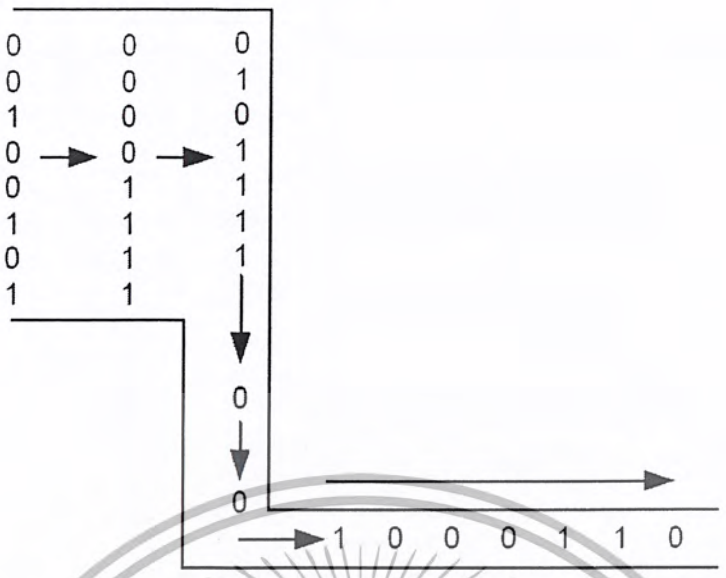
รูปที่ 2.12 การส่งข้อมูลแบบอนุกรม

2.3.1 การแปลงสัญญาณข้อมูลระหว่างแบบอนุกรมและแบบขนาน

ในการแปลงรูปแบบข้อมูลระหว่างอนุกรมและขนาน จะอาศัยรีจิสเตอร์เพื่อเป็นบัฟเฟอร์ในการเก็บข้อมูลชั่วคราว เช่น ข้อมูลที่ส่งเข้ามาเป็นรูปแบบอนุกรมซึ่งจะส่งเรียงเข้ามาทีละบิต และเมื่อเข้ามาถึงปลายทาง บิตแต่ละบิตจะถูกนำมาจัดเก็บเรียงลำดับกันอยู่ในบัฟเฟอร์ จนกระทั่งครบตามจำนวนบิตที่ต้องการ เช่น เรียงกันจนครบ 8 บิต จากนั้นรีจิสเตอร์ก็จะส่งข้อมูลทั้งชุดนี้ออกไปด้วยการส่งสัญญาณให้หน่วยประมวลผลรับทราบ เพื่อให้โปรแกรมนำไบท์เหล่านั้นไปประมวลผลต่อไป ในขณะที่หากต้องการแปลงข้อมูลในรูปแบบขนานกลับไปเป็นแบบอนุกรม ก็สามารถกระทำได้ด้วยกระบวนการดังกล่าวในทิศทางตรงข้ามโดยพิจารณาจากรูปที่ 2.19 และรูปที่ 2.20 ที่แสดงขั้นตอนการแปลงข้อมูลระหว่างแบบอนุกรมและแบบขนาน ซึ่งกระบวนการแปลงสัญญาณข้อมูลเหล่านี้จะมีวงจรพิเศษที่เรียกว่า **UART(Universal Asynchronous Receiver Transmitter)**ที่ทำการแปลงข้อมูลแบบขนานเป็นแบบอนุกรมหรือแปลงกลับจากแบบอนุกรมเป็นแบบขนานนอกจากนี้ยังมีวงจรที่เรียกว่า **USART(Universal Synchronous/Asynchronous Receiver Transmitter)** ซึ่งมีคุณสมบัติเช่นเดียวกับ UART แต่จะมีส่วนของการทำงานกับการซิงโครนัสข้อมูลด้วย ปัญหาของการส่งข้อมูลแบบอนุกรม คือเรื่องแบ่งตัวอักขระแต่ละตัวว่าจะแบ่ง ณ ตำแหน่งบิตใดซึ่งทั้งฝ่ายต้นทางและปลายทางจะต้องมีข้อตกลงร่วมกัน กล่าวคือ ทั้งฝ่ายต้นทางและปลายทางจะต้องรับรู้ร่วมกันว่าจะต้องแบ่งแต่ละตัวอักขระ ณ ตำแหน่งบิตใด เนื่องจากบิตแต่ละบิตจะทยอยส่งมาเป็นลำดับเรื่อยๆ ดังนั้น การส่งข้อมูลแบบอนุกรมจึงมีวิธีการอยู่ 2 วิธีด้วยกัน คือ การส่งข้อมูลแบบอะซิงโครนัส และ การส่งข้อมูลแบบซิโครนัส



รูปที่ 2.13 การแปลงข้อมูลจากแบบอนุกรมไปเป็นแบบขนาน(Serial-to-Parallel)

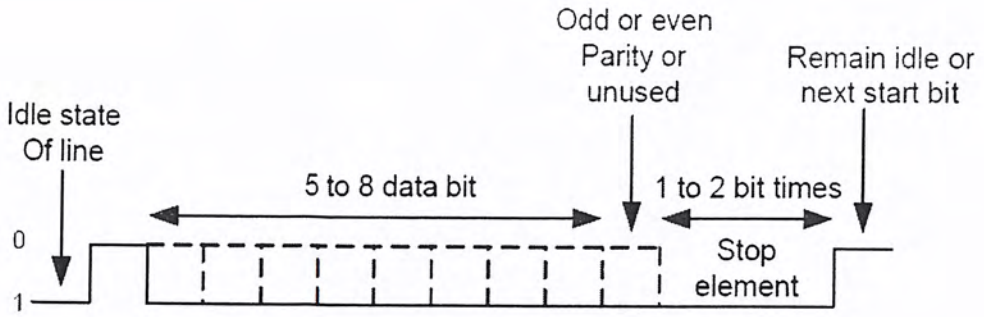


รูปที่ 2.14 การแปลงข้อมูลจากแบบขนานไปเป็นแบบอนุกรม(Parallel-to-Serial)

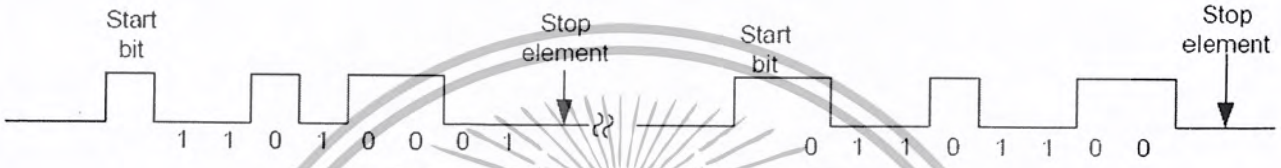
2.3.2 การส่งข้อมูลแบบอะซิงโครนัส(Asynchronous Transmission)

การส่งข้อมูลแบบอะซิงโครนัสเป็นการสื่อสารด้วยวิธีการส่งอักขระ(Character) แต่ละตัวณ เวลาใดก็ได้โดยฝ่ายส่งข้อมูลและฝ่ายรับข้อมูลต่างก็มีสัญญาณนาฬิกาควบคุมจังหวะการทำงานด้วยตัวเอง จึงทำให้การทำงานของทั้งสองฝ่ายไม่สอดคล้องกันตามจังหวะนาฬิกา กล่าวคือจะเป็นอิสระต่อกัน แต่อย่างไรก็ตามสัญญาณนาฬิกาทั้งสองฝ่ายนั้นจะต้องมีความถี่เท่ากัน

ในสภาวะนิ่งเฉย(Idle State)ที่ไม่มีการส่งข้อมูลใดๆ(บางครั้งเรียกสภาวะ Marking) จะถูกกำหนดให้สัญญาณมีค่าเป็น "1" แต่เมื่อมีการส่งข้อมูลระดับสัญญาณจะถูกกำหนดให้เป็น "0" อยู่ช่วงเวลาหนึ่งทำให้เกิดเป็นบิตขึ้นมาหนึ่งบิตที่เรียกว่า **บิตเริ่มต้น(Start Bit)** เพื่อบ่งบอกว่านับจากนี้ไปจะมีข้อมูลส่งมา และเมื่อฝ่ายส่งได้ส่งบิตข้อมูลจนครบตามจำนวนบิตที่ต้องการแล้ว จากนั้นก็จะส่งข้อมูลอีกบิตหนึ่ง ซึ่งระดับสัญญาณจะถูกกำหนดให้เป็น "1" เป็นตัวปิดท้ายที่เรียกว่า **บิตสิ้นสุด(Stop Bit)** เพื่อบ่งบอกให้รู้ว่าได้ส่งข้อมูลครบแล้ว โดยสัญญาณ "1" ที่เป็นบิตสิ้นสุดนี้จะส่งมานานช่วงระยะเวลาหนึ่ง ทำให้ฝ่ายรับได้รับรู้ทันทีว่ามีการส่งบิตสิ้นสุดมาแล้ว



(a) Character format



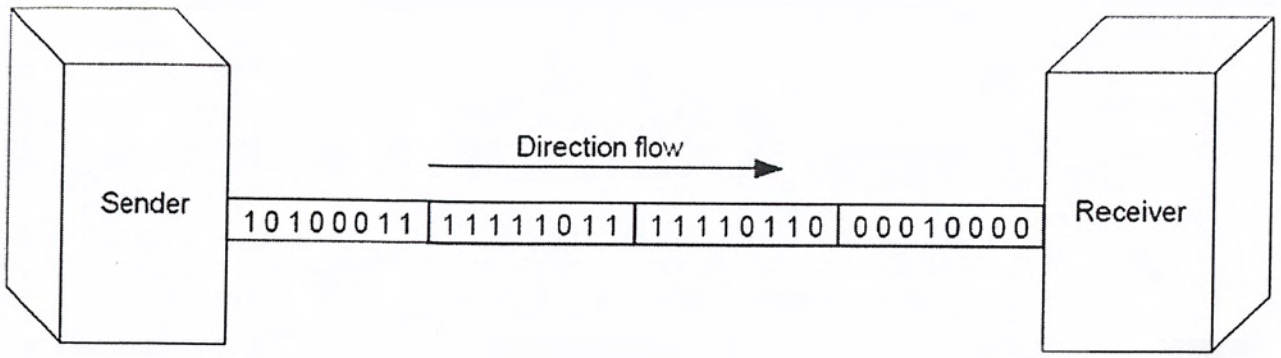
รูปที่ 2.15 แสดงรายละเอียดของรูปแบบการส่งข้อมูลแบบอะซิงโครนัส

หลังจากที่ได้รับข้อมูลครบตามจำนวนบิต ตัวอักขระตัวที่สองที่จะส่งเป็นลำดับถัดไป ก็ไม่ต้องมีเวลาเท่ากันเสมอ ซึ่งหากยังไม่มีข้อมูล สัญญาณก็จะอยู่ในสถานะนิ่งเฉยเพื่อรอการส่งข้อมูลลำดับถัดไป

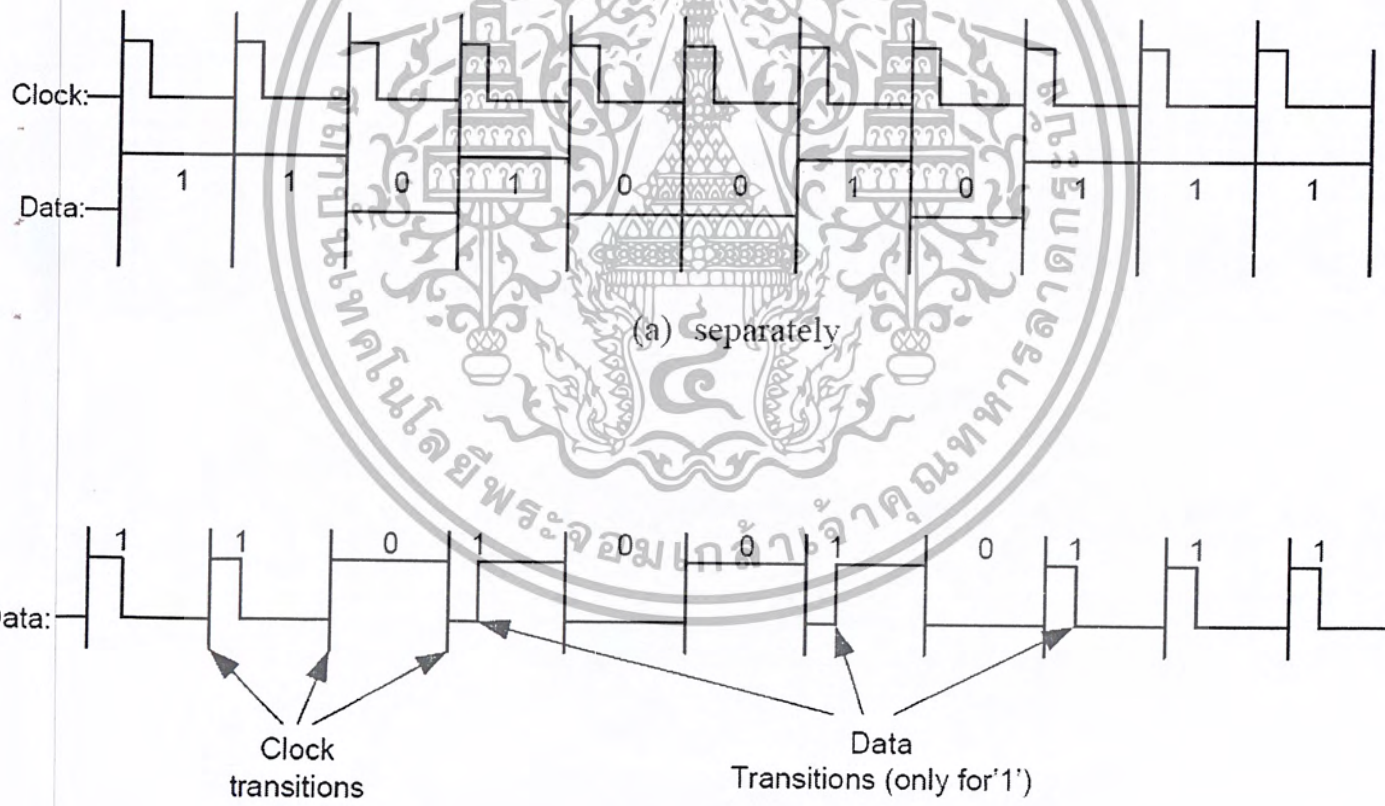
2.3.3 การส่งข้อมูลแบบซิงโครนัส(Synchronous Transmission)

การส่งข้อมูลแบบซิงโครนัสเป็นการส่งข้อมูลแบบกลุ่ม โดยบิตที่ทยอยส่งเข้ามาจะมีการรวมกลุ่มกันให้มีขนาดใหญ่ขึ้นที่เรียกว่าเฟรมหรือบล็อกข้อมูล ซึ่งอาจจะประกอบด้วยหลายๆอักขระด้วยกัน โดยระหว่างการส่งจะปราศจากช่องว่าง รวมถึงบิตเริ่มต้นและบิตสิ้นสุด ซึ่งทำให้ไม่มีอะไรมาคั่นระหว่างข้อมูลแต่ละตัว จึงทำให้การกะจังหวะ(Timing) กลายเป็นสิ่งสำคัญมากสำหรับการส่งข้อมูลแบบซิงโครนัส กล่าวคือ ทั้งฝ่ายส่งข้อมูลและฝ่ายรับข้อมูลจะต้องทำงานสอดคล้องกันตามจังหวะของสัญญาณนาฬิกา โดยฝ่ายรับข้อมูลจะได้รับสัญญาณนาฬิกามาจากฝ่ายส่งข้อมูล

ดังนั้นวิธีการส่งข้อมูลชนิดนี้สัญญาณนาฬิกาทั้งฝ่ายรับละฝ่ายส่งจะต้องซิงโครไนซ์กันฝ่ายส่งอาจส่งสัญญาณนาฬิกาแยกออกมาพร้อมกับข้อมูลดังรูปที่2.23(a) แต่วิธีการนี้จะใช้ได้ดีกับระยะทางใกล้ๆ เพราะจะเกิดปัญหาเรื่องของสัญญาณที่อ่อนตัวลง แต่ก็มีอีกวิธีหนึ่งดังรูปที่ 2.23(b)ซึ่งจะเป็นการนำเอาสัญญาณนาฬิการวมเข้ากับสัญญาณข้อมูล เช่น การเข้ารหัสแบบแมนเชสเตอร์ในระบบแลน เป็นต้น

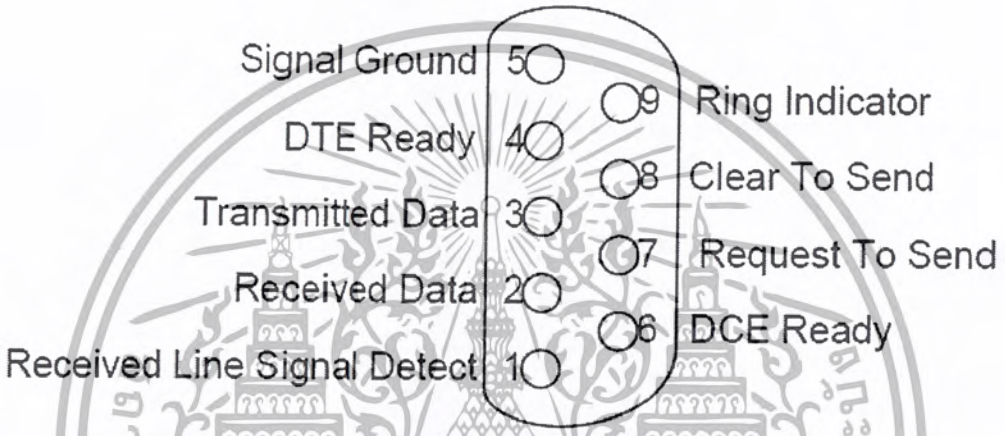


รูปที่ 2.16 ภาพแสดงการส่งข้อมูลแบบซิงโครนัสที่นำเสนอในรูปแบบง่าย



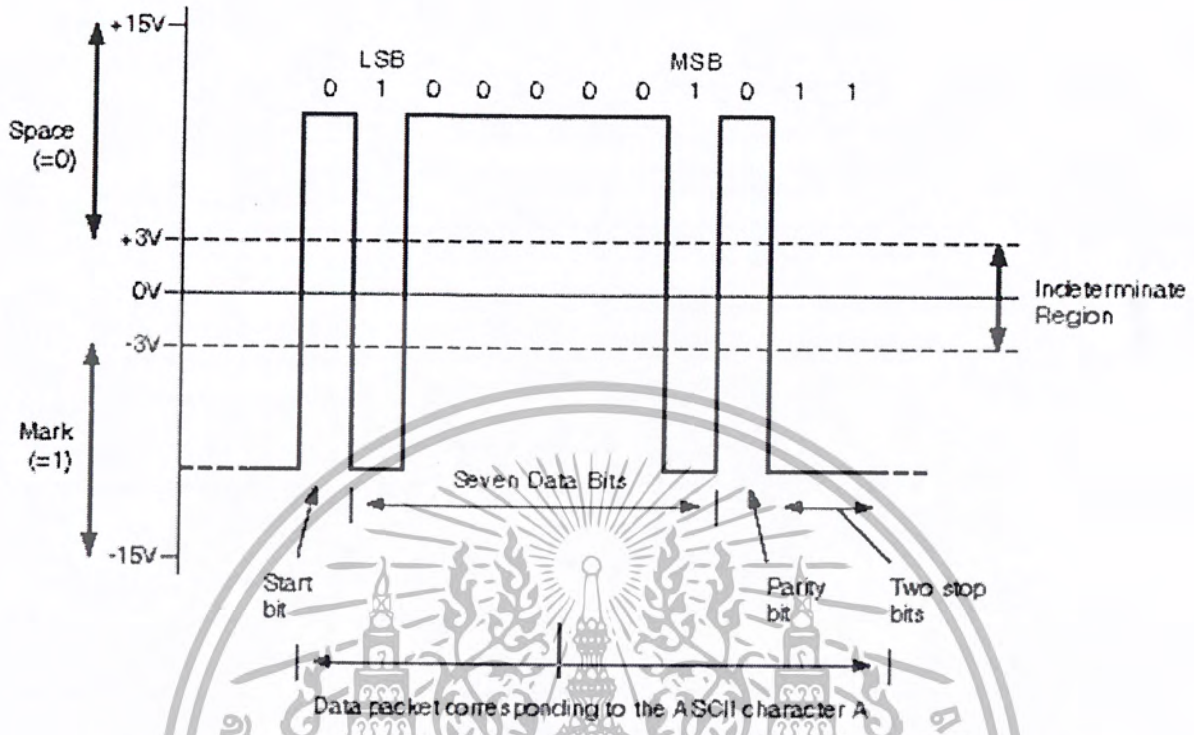
รูปที่ 2.17 ภาพแสดงการเข้าจังหวะระหว่างสัญญาณนาฬิกาและข้อมูลในการส่งข้อมูล

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

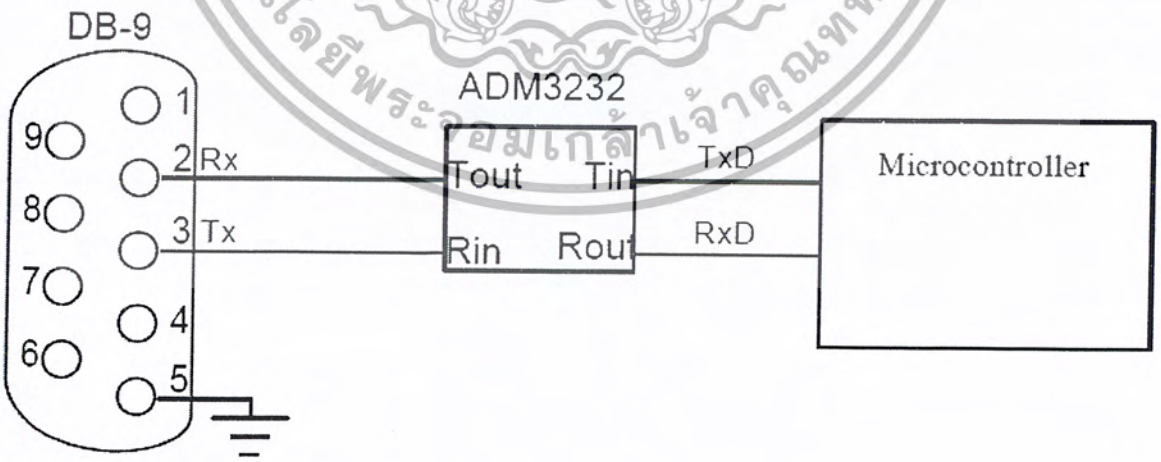


รูปที่ 2.18 แสดงโครงสร้างคอนเนคเตอร์ DB-9

Received Line Signal Detect : ขานี้จะ active เมื่อมีการส่งสัญญาณ Carrier จากโมเด็ม
Received Data : ขานี้สำหรับรับข้อมูลอนุกรมจากคอมพิวเตอร์
Transmitted Data : ขานี้สำหรับส่งข้อมูลอนุกรมจากคอมพิวเตอร์
DTE Ready : ใช้บอกอุปกรณ์ปลายทางว่าต้องการติดต่อ
Signal Ground : กราวด์ของวงจร
DCE Ready : ใช้คู่กับขา DTE Ready ในการตรวจสอบการติดต่อ
Request To Send : ใช้บอกอุปกรณ์ปลายทางให้ส่งข้อมูลกลับมาให้
Clear To Send : ใช้ตรวจสอบว่าอุปกรณ์ปลายทางพร้อมส่งข้อมูลหรือไม่
Ring Indicator : ขานี้จะ active เมื่อ โมเด็มได้รับสัญญาณจากโทรศัพท์



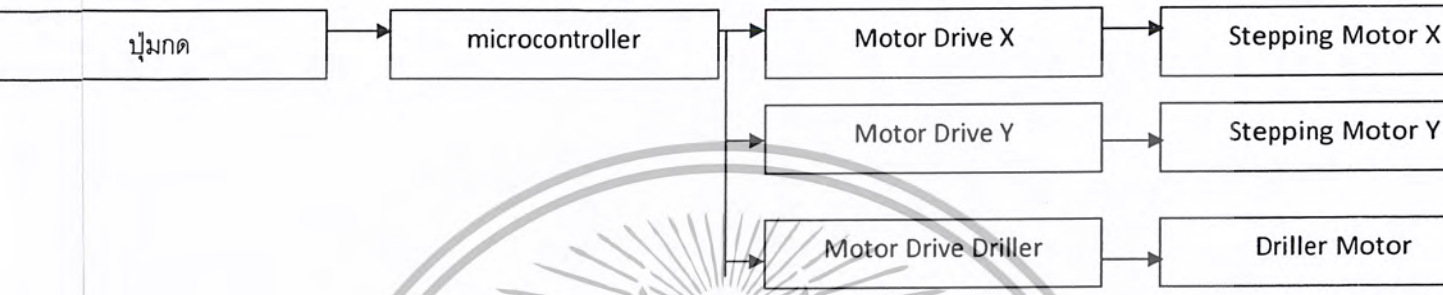
ในการเชื่อมต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์นั้นแต่ละส่วนมีมาตรฐานต่างกันจึงต้องทำการแปลงแรงดันให้อยู่ในมาตรฐานการใช้งานของไมโครคอนโทรลเลอร์โดยใช้วงจรรวมเบอร์ ADM3232



รูปที่ 2.20 แสดงการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ผ่าน RS232-C

บทที่ 3
การออกแบบ

3.1 การทำงานของระบบ



รูปที่ 3.1 การทำงานโดยรวมของระบบ

ผู้ใช้งานจะกำหนดข้อมูลและคำสั่งต่างๆผ่านปุ่มกดในการรับข้อมูลจากผู้ใช้งานหลังจากนั้นจะส่งข้อมูลหรือคำสั่งไปยังคอนโทรลเลอร์ โดยไมโครคอนโทรลเลอร์ที่ใช้ในโครงงานชิ้นนี้คือ VX-Propeller โดยใช้ชิปโพรเพลเลอร์เบอร์ P8X32-D40 LCP2138 โดยเมื่อไมโครคอนโทรลเลอร์ได้รับคำสั่งจากปุ่มกดแล้วนำมาประมวลเพื่อส่งพัลส์และทิศทางไปยังตัวขับสเต็ปปิ้งมอเตอร์แกน X และแกน Y และสั่งเคลื่อนที่สว่านตัวเจาะ การทำงานในแต่ละส่วนมีดังนี้

Motor Driver ของแกน X เป็นตัวรับสัญญาณพัลส์และทิศทางจากไมโครคอนโทรลเลอร์เพื่อทำการส่งสัญญาณไฟฟ้าไปยังเฟสต่าง ๆ ของสเต็ปปิ้งมอเตอร์ที่ควบคุมแกน X

Motor Driver ของแกน Y เป็นตัวรับสัญญาณพัลส์และทิศทางจากไมโครคอนโทรลเลอร์เพื่อทำการส่งสัญญาณไฟฟ้าไปยังเฟสต่าง ๆ ของสเต็ปปิ้งมอเตอร์ที่ควบคุมแกน Y

Motor Driver of Driller เป็นตัวรับสัญญาณควบคุมจากไมโครคอนโทรลเลอร์เพื่อส่งพัลส์และทิศทางไปยังมอเตอร์ตัวเจาะ

Stepping Motor ของแกน X เป็นสเต็ปปิ้งมอเตอร์ควบคุมการเคลื่อนที่ทางแกน X

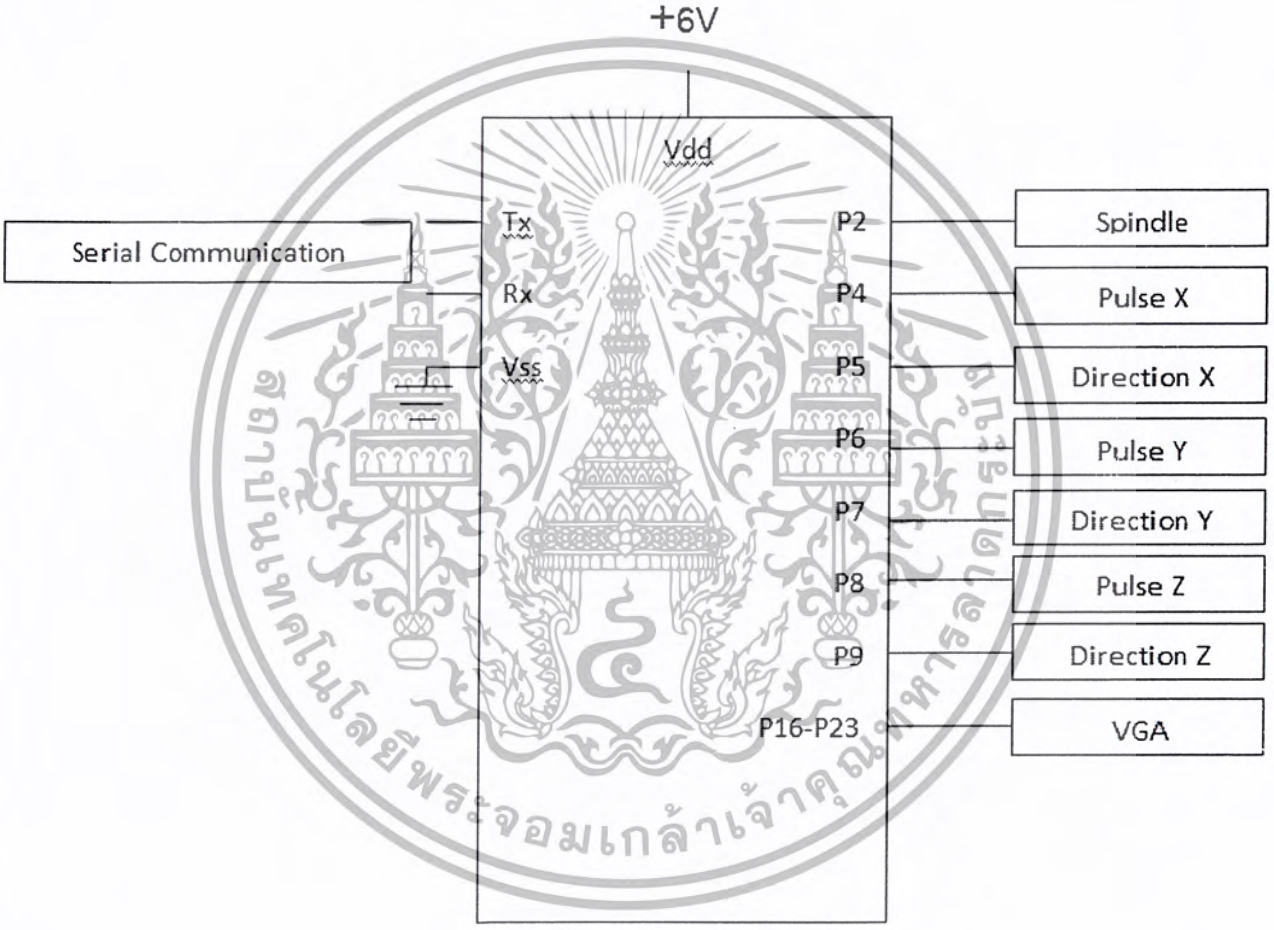
Stepping Motor ของแกน Y เป็นสเต็ปปิ้งมอเตอร์ควบคุมการเคลื่อนที่ทางแกน Y

Driller Motor เป็นมอเตอร์ควบคุมความสูงของหัวเจาะว่าต้องการความลึกขนาดไหน

3.2 ขั้นตอนการออกแบบ

3.2.1 การออกแบบวงจรส่วนของไมโครคอนโทรลเลอร์

วงจรในส่วนของไมโครคอนโทรลเลอร์จะเป็นการรับข้อมูลจากการกดปุ่มทิศทางมาทำการประมวลผลและสั่งคำสั่งควบคุมการเคลื่อนที่ของสเต็ปมอเตอร์ทั้งในแกน$_X$, แกน$_Y$ และในแกน$_Z$ โดยเราจะป้อนพัลส์กำหนดความเร็วในการเคลื่อนที่และเลือกทิศทางการเคลื่อนที่ให้กับวงจรขับสเต็ปมอเตอร์



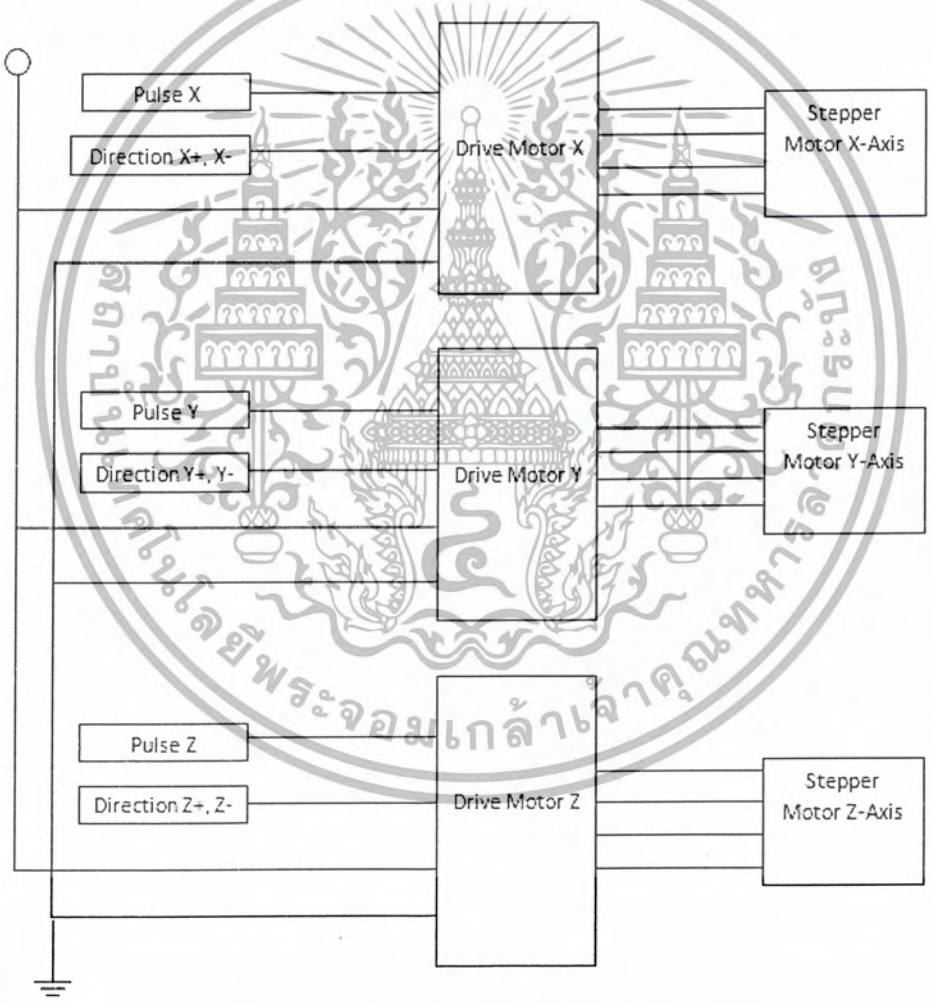
รูปที่ 3.2 แสดงวงจรเชื่อมต่อของไมโครคอนโทรลเลอร์

จากรูปที่ 3.2 ขา P4 ป้อนพัลส์ให้วงจรขับสเต็ปมอเตอร์ควบคุมความเร็วการเคลื่อนที่ในแนวแกน X ขาP5 ควบคุมทิศทางการเคลื่อนที่ในแกน X+ และควบคุมทิศทางการเคลื่อนที่ในแกน X- ขา P6 ป้อนพัลส์ให้วงจรขับสเต็ปมอเตอร์ควบคุมความเร็วการเคลื่อนที่ในแนวแกน Y ขาP7 ควบคุมทิศทางการเคลื่อนที่ในแกน Y+ และควบคุมทิศ

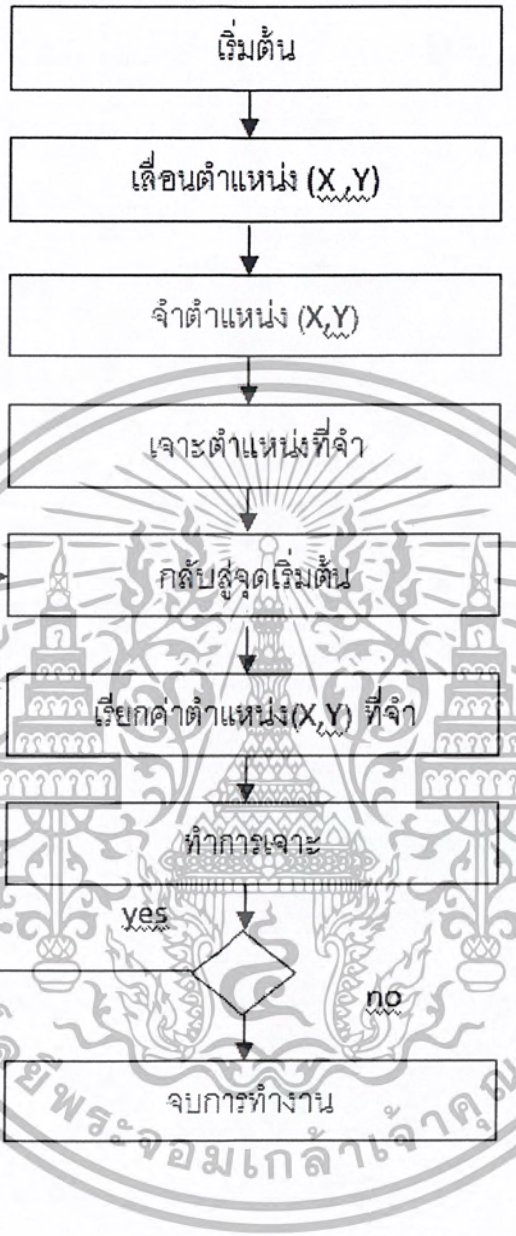
ทางการเคลื่อนที่ในแกน Y- ขา P8 ป้อนพัลส์ให้วงจรขับสเต็ปมอเตอร์ควบคุมความเร็วการเคลื่อนที่ในแนวแกน Z ขาP9 ควบคุมทิศทางการเคลื่อนที่ในแกน X+ และควบคุมทิศทางการเคลื่อนที่ในแกน X- และขา P2 ควบคุมการหมุนของหัวเจาะ

3.2.2 การออกแบบวงจรขับสเต็ปมอเตอร์

ในการทำโครงงานครั้งนี้ใช้ตัวขับสเต็ปมอเตอร์แบบไบโพล่าร์แบบสำเร็จรูปที่มีอยู่เดิมแล้วมาใช้ วงจรขับสเต็ปนี้เราจะป้อนทิศทางและพัลส์ความถี่เพื่อควบคุมความเร็ว แล้วตัวขับสเต็ปมอเตอร์จะทำการป้อนแรงดันเรียงเฟสของสเต็ปมอเตอร์โดยอัตโนมัติ



รูปที่ 3.3 แสดงวงจรขับสเต็ปมอเตอร์แบบยูนิโพลาร์



รูปที่ 3.4 แสดงขั้นตอนการทำงานของเครื่องเจาะcnc

บทที่ 4

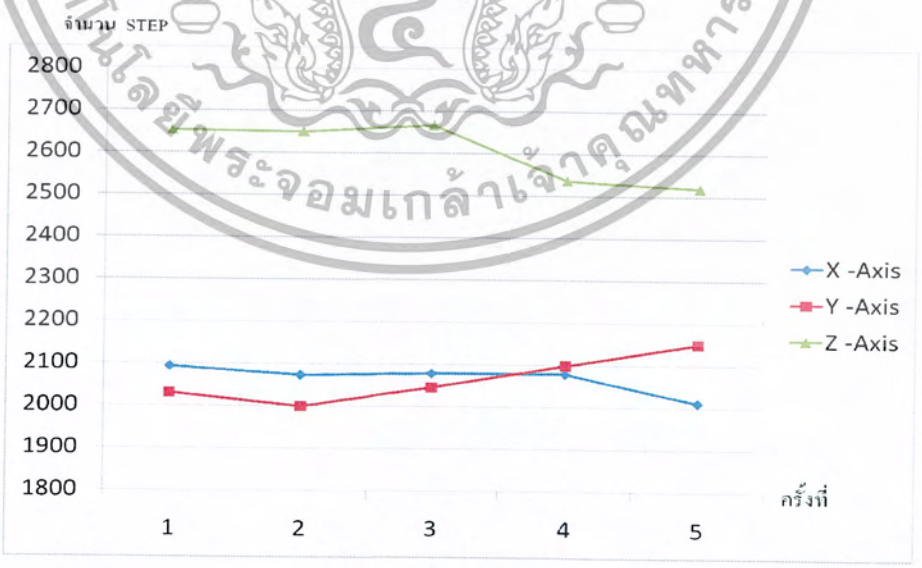
การทดลอง

4.1. การทดสอบหา step ต่อระยะทาง 1 นิ้ว

ตารางที่ 4.1 ตารางแสดง Step ต่อระยะทาง 1นิ้ว

แกน / ครั้งที่	X	Y	Z
1	2093	2031	2653
2	2073	1999	2650
3	2078	2045	2666
4	2078	2098	2536
5	2011	2151	2519
เฉลี่ย	2066.6	2064.8	2604.8

กราฟแสดงการหา step ต่อระยะทาง 1 นิ้ว



รูปที่ 4.1 กราฟแสดงการหา step ต่อระยะทาง 1 นิ้ว

จากตารางแสดงผลการหา Step ต่อระยะทาง 1นิ้ว เพื่อที่จะหาความละเอียดของ1Step ซึ่งจากการคำนวณ

Steping Motor X-AXIS 1Step = 12.3 um

Steping Motor Y-AXIS 1Step = 12.3 um

Steping Motor Z-AXIS 1Step = 9.75 um

การหาความถี่ของ Pluse ที่ออกจาก Microcontroller

หาจาก การเทียบจากclock ของตัวMicrocontroller

12_000_000 ได้ 1 วินาที

10_000 ได้ 1/1200 วินาที

จาก f = 1/T T = 1/1200 วินาที

ดังนั้น **F= 1200** Hz

การหาระยะทางของ 1 Step ของ Steping Motor X-AXIS

หาจากเทียบระยะทาง

1 นิ้ว ได้ 2066.6 Step , 1 นิ้ว = 2.54 cm

หาระยะทาง 1 Step

2.54/2066.6 = 0.00123 cm or 12.3 um

ดังนั้น **1 Step = 12.3 um**

การหาระยะทางของ 1 Step ของ Steping Motor Y-AXIS

หาจากเทียบระยะทาง

1 นิ้ว ได้ 2064.8 Step , 1 นิ้ว = 2.54 cm

หาระยะทาง 1 Step

2.54/2064.8 = 0.00123 cm or 12.3 um

ดังนั้น **1 Step = 12.3 um**

การหาระยะทางของ 1 Step ของ Steping Motor Z-AXIS

หาจากเทียบระยะทาง

1 นิ้ว ได้ 2604.8 Step , 1 นิ้ว = 2.54 cm

หาระยะทาง 1 Step

2.54/2604.8 = 0.000975 cm or 9.75 um

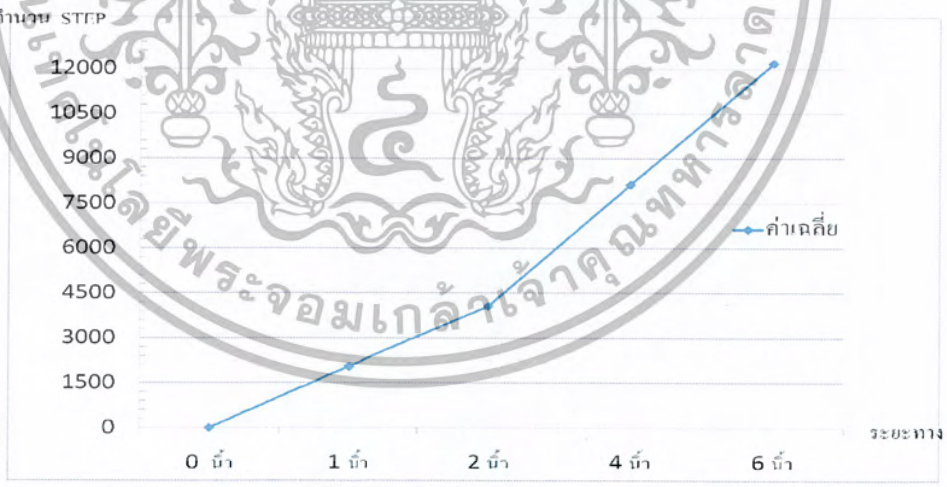
ดังนั้น **1 Step = 9.75 um**

4.2. การทดสอบหาความเป็นเชิงเส้น

ตารางที่ 4.2 ตารางแสดงความเป็น Linear ของแกน X

ระยะทาง(นิ้ว) / ครั้งที่	1	2	4	6
1	2111	4090	8111	12190
2	2013	4091	8107	12191
3	2036	4084	8058	12183
4	2026	4009	8156	12196
5	2012	3995	8194	12151
เฉลี่ย	2039.6	4053.8	8125.2	12182.2

กราฟแสดงความเป็น เชิงเส้น ของแกน X



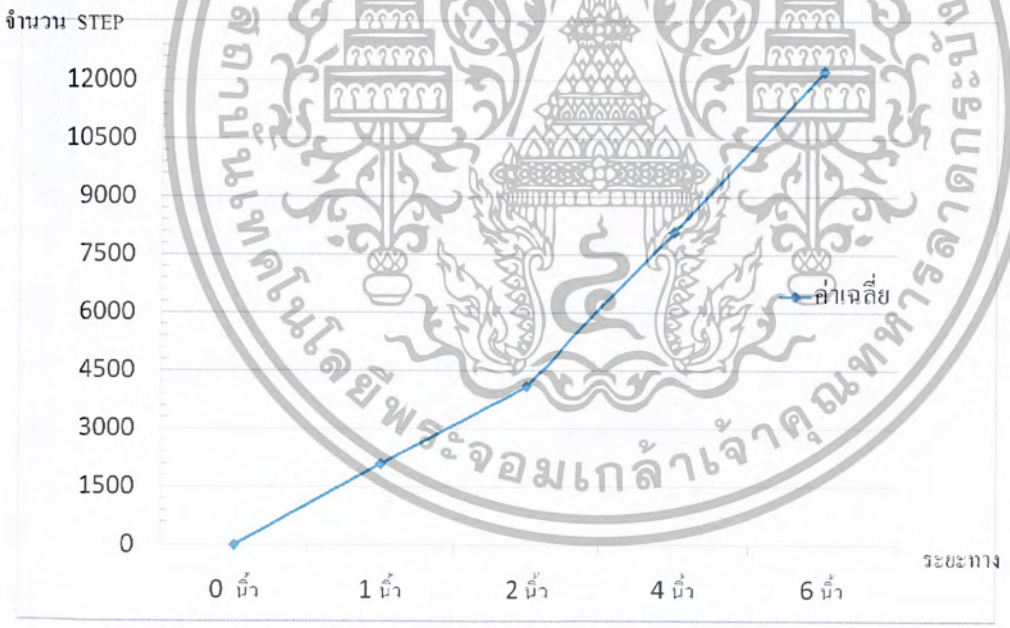
รูปที่ 4.2 กราฟแสดงความเป็น เชิงเส้น ของแกน X

จากตารางแสดงผลการหาความเป็น Linear ของแกน X ซึ่งคิดจากการวัดระยะทางที่แตกต่างกัน 1,2,4,6 นิ้ว เพื่อดูการเคลื่อนที่ของมอเตอร์

ตารางที่ 4.3 ตารางแสดงความเป็น Linear ของแกน Y

ระยะทาง(นิ้ว) / ครั้งที่	1	2	4	6
1	2100	4100	8050	12200
2	2075	4045	8113	12273
3	2096	4119	8083	12157
4	2111	4147	8116	12301
5	2074	4093	8036	12215
เฉลี่ย	2091.2	4100.8	8079.6	12229.2

กราฟแสดงความเป็น เชิงเส้น ของแกน Y



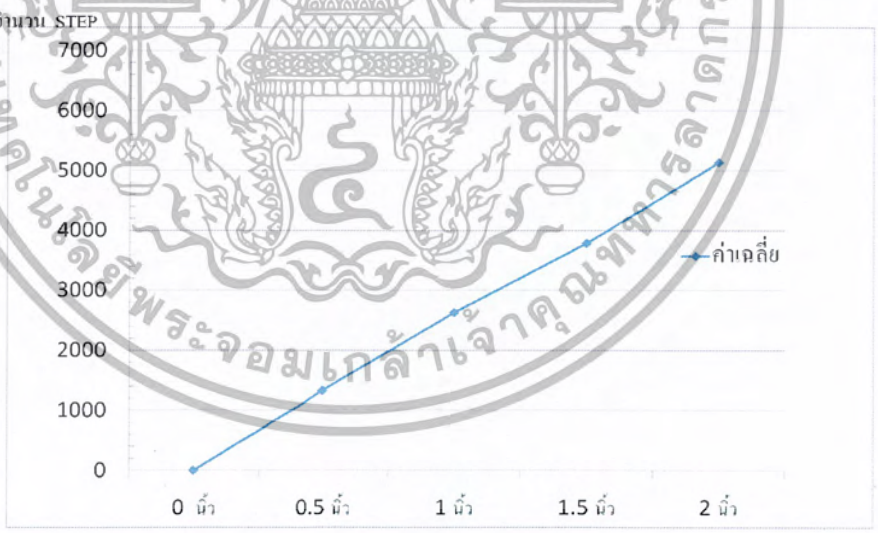
รูปที่ 4.3 กราฟแสดงความเป็น เชิงเส้น ของแกน Y

จากตารางแสดงผลการหาความเป็น Linear ของแกน Y ซึ่งคิดจากการวัดระยะทางที่แตกต่างกัน 1,2,4,6 นิ้ว เพื่อดูการเคลื่อนที่ของมอเตอร์

ตารางที่ 4.3 ตารางแสดงความเป็น Linear ของแกน Z

ระยะทาง(นิ้ว) / ครั้งที่	0.5	1	1.5	2
1	1320	2600	3799	5228
2	1333	2622	3750	5177
3	1300	2589	3772	5100
4	1260	2676	3766	5085
5	1416	2642	3820	5063
เฉลี่ย	1325.8	2625.8	3781.4	5130.6

กราฟแสดงความเป็น เชิงเส้น ของแกน Z



รูปที่ 4.4 กราฟแสดงความเป็น เชิงเส้น ของแกน Z

จากตารางแสดงผลการหาความเป็น Linear ของแกน Z ซึ่งคิดจากการวัดระยะทางที่แตกต่างกัน

0.5 , 1 , 1.5 , 2 นิ้ว เพื่อดูการเคลื่อนที่ของมอเตอร์

4.3. การทดสอบหาความผิดพลาดของเครื่องเจาะ

ตารางที่ 4.4 ตารางแสดงความผิดพลาดของเครื่องเจาะ โดยการใช้ 2 **Operator**

Operator 1

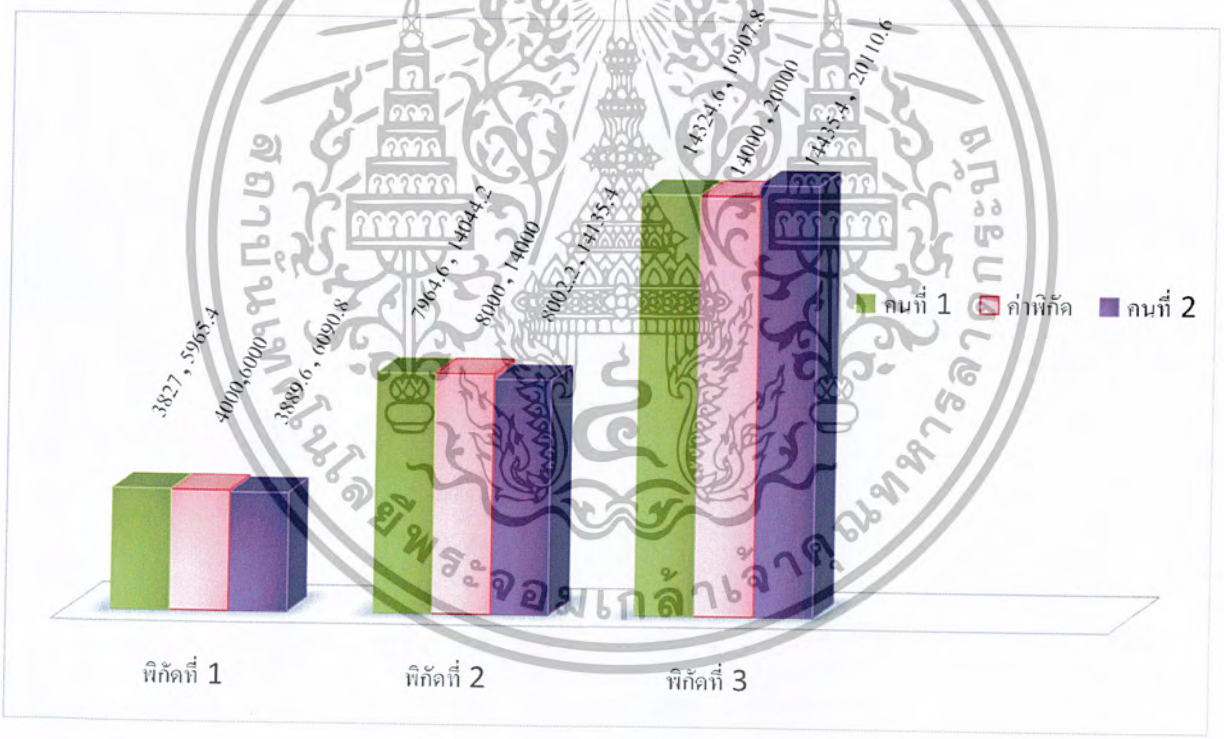
พิกัด / ครั้งที่	พิกัดที่ 1 4000 , 6000	พิกัดที่ 2 8000 , 14000	พิกัดที่ 3 14000 , 20000
1	4018 , 5917	7982 , 14171	14021 , 19939
2	3703 , 5989	7998 , 14087	14196 , 19878
3	3809 , 5960	7984 , 13894	14744 , 19858
4	3722 , 5969	7928 , 14071	14177 , 19963
5	3883 , 5992	7931 , 13998	14485 , 19901
ค่าเฉลี่ย	3827 , 5965.4	7964.6 , 14044.2	14324.6 , 19907.8
ความผิดพลาด	0.04325 , 0.00577	0.00443 , 0.00316	0.02318 , 0.00461

Operator 2

พิกัด / ครั้งที่	พิกัดที่ 1 4000 , 6000	พิกัดที่ 2 8000 , 14000	พิกัดที่ 3 14000 , 20000
1	3940 , 6262	7815 , 14312	14365 , 20200
2	3953 , 6121	7883 , 14249	14311 , 20079
3	3766 , 5974	7984 , 14052	14246 , 20045

4	3869 , 5976	7890 , 13966	14143 , 20109
5	3920 , 6121	8539 , 14098	15112 , 20120
ค่าเฉลี่ย	3889.6 , 6090.8	8002.2 , 14135.4	14435.4 , 20110.6
ความผิดพลาด	0.0276 , 0.01513	0.00027 , 0.00967	0.03110 , 0.00553

กราฟแสดงความผิดพลาดของเครื่องเจาะ



รูปที่ 4.5 กราฟแสดงความผิดพลาดของเครื่องเจาะ

จากตารางแสดงความผิดพลาดของเครื่องเจาะนี้โดยการใช้ 2 operator ในการเจาะพิกัดจุด 3 จุด ที่ตำแหน่ง (X,Y) คือ (4000,6000) , (8000,14000) , (14000,20000) โดยทดลองทั้งหมด 5 ครั้ง ที่ใช้ในการทำการทดลองเพื่อหาความผิดพลาดในการทดลอง

บทที่ 5

สรุปและวิจารณ์ผล

5.1 สรุปผลการทดลอง

จากผลการทดลองเราสามารถเขียนโปรแกรมเพื่อควบคุมเครื่องเจาะอัตโนมัติ ให้รับ input ได้ และสามารถแสดงจำนวน Step ออกทางหน้าจอ VGA ได้ และสามารถสั่งการทำงานของเครื่องเจาะให้ทำงานอัตโนมัติตามค่าที่ได้ทำการจดจำไว้ ซึ่งเมื่อทำการเจาะเสร็จก็สามารถเคลื่อนที่กลับสู่จุดเริ่มต้นได้

จากการทดลองหาจำนวน Step ต่อระยะทาง 1 นิ้ว ซึ่งเราจะได้ระยะแกน X-AXIS 1Step = 12.3 um , Y-AXIS 1Step = 12.3 um , Z-AXIS 1Step = 9.75 um

จากการทดลองหาความผิดพลาด จะเห็นว่ามีความผิดพลาดเกิดขึ้นในการทำการทดลอง ซึ่งอาจจะเกิดจากตัวอุปกรณ์หรือตัวบุคคล ตัวอุปกรณ์เช่น เลเซอร์ ซึ่งมีความละเอียดของลำแสงน้อยจึงทำให้การคาดคะเนอาจจะเกิดการคลาดเคลื่อนเกิดขึ้น วิธีแก้ไขอาจจะใช้ส่วนโปรแกรมอื่นๆ เสริมในการdetect find coordinate ที่แม่นยำยิ่งขึ้น

จากการทดลองตรวจสอบความเป็นเชิงเส้นสำหรับส่วนการทดลองหาความแป็น Linear จะเป็นการทดสอบการเคลื่อนที่ในระยะต่างๆ คือ 1,2,4,6 นิ้ว เพื่อดูความคงที่ของมอเตอร์(ความเป็นLinear)ซึ่งจากการทดลองจะเห็นว่าการเคลื่อนที่ของมอเตอร์มีความเป็นเชิงเส้น

5.2.ปัญหาที่พบจากการทำโครงงาน

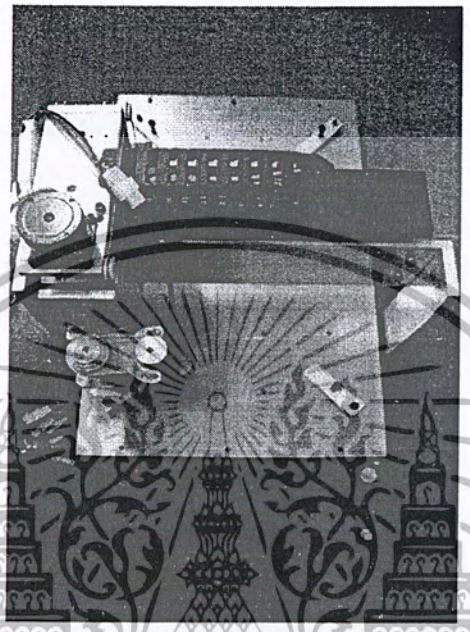
1. อุปกรณ์ที่ใช้ในการวัดไม่มีความละเอียดเพียงพอ
2. ความคลาดเคลื่อนที่เกิดขึ้นจากอุปกรณ์ที่ใช้งานและตัวบุคคลที่ทำการใช้งาน
3. ความคลาดเคลื่อนจากระบบทางกลของเครื่องมือที่ใช้ในการทำโครงงาน
4. ภาษาที่ใช้เขียนโปรแกรมเป็นภาษาใหม่จึงยากต่อการศึกษา
5. คู่มือที่ใช้ในการศึกษามีน้อย

5.3. ประโยชน์ของ เครื่องเจาะอัตโนมัติ

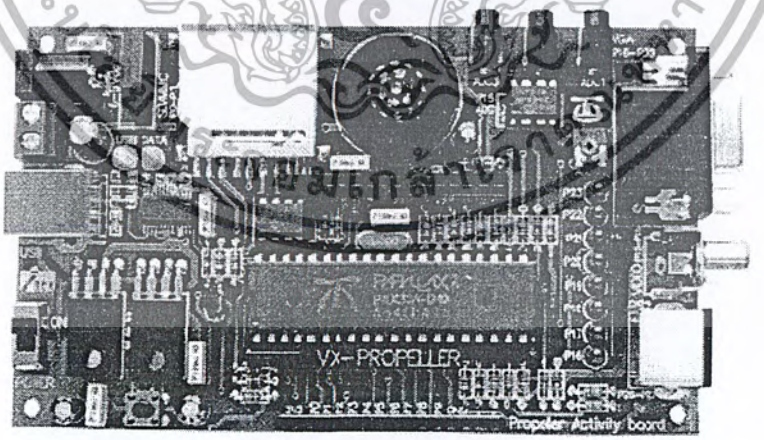
1. ลดความวุ่นวายของผู้ควบคุม
2. ลดความเหนื่อยล้าของผู้ปฏิบัติงาน
3. สามารถทำนายเวลาในการผลิตแต่ละชิ้นได้
4. ความคงเส้นคงวาในการผลิตและความถูกต้องแม่นยำของชิ้นงาน
5. เราสามารถเรียกโปรแกรมนั้นกลับมาใช้ใหม่ในครั้งต่อไปเมื่อต้องทำงานชิ้นนั้นอีก
6. สามารถเห็นพิกัดที่เจาะออกทางหน้าจอ VGA ได้

ภาคผนวก

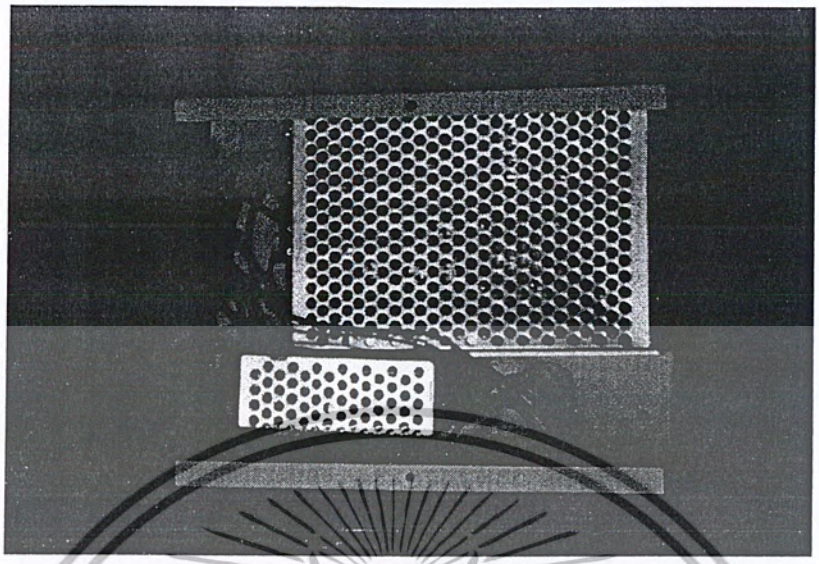
ภาคผนวก ก.
รูปภาพส่วนประกอบของเครื่องเจาะอัตโนมัติควบคุมด้วยไมโครคอนโทรลเลอร์



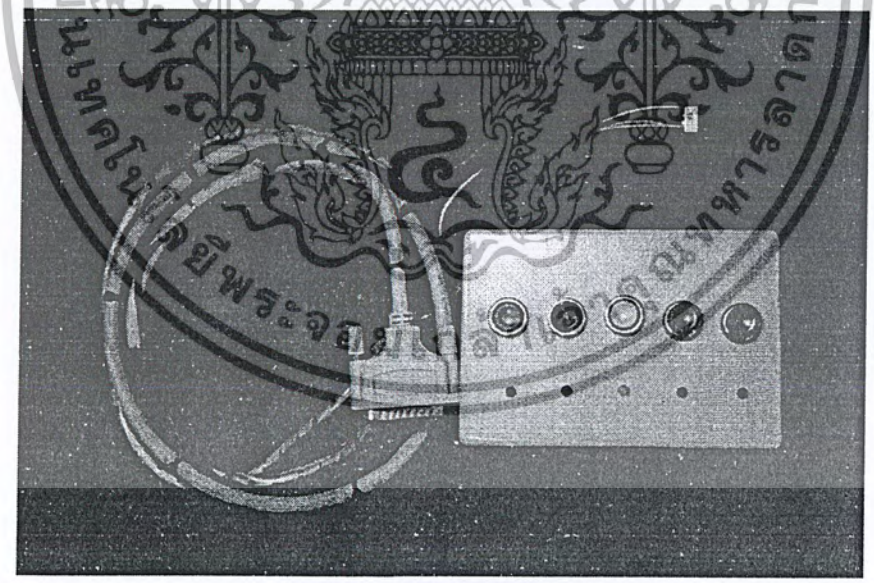
รูปที่ ก.1 เครื่องเจาะอัตโนมัติควบคุมด้วยไมโครคอนโทรลเลอร์



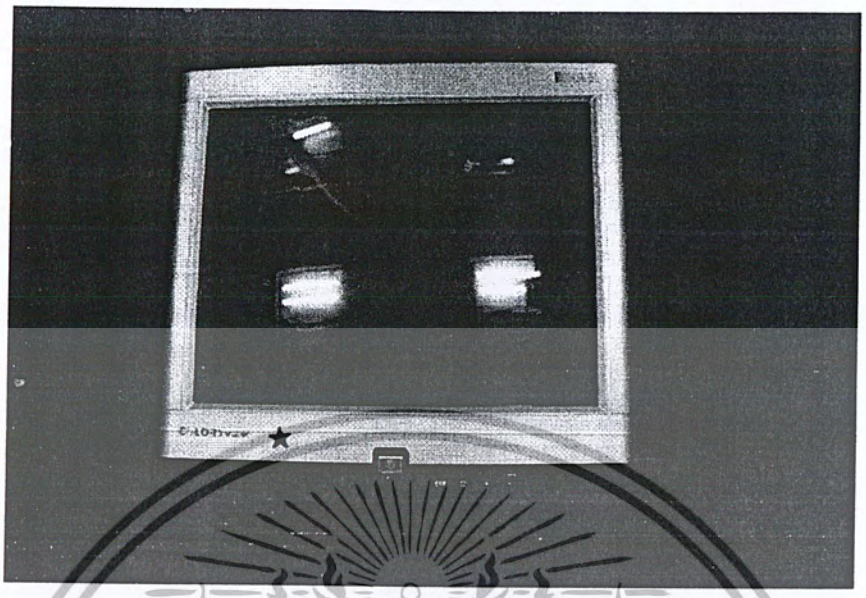
รูปที่ ก.2 ส่วนไมโครคอนโทรลเลอร์ (บอร์ด VX-Propeller)



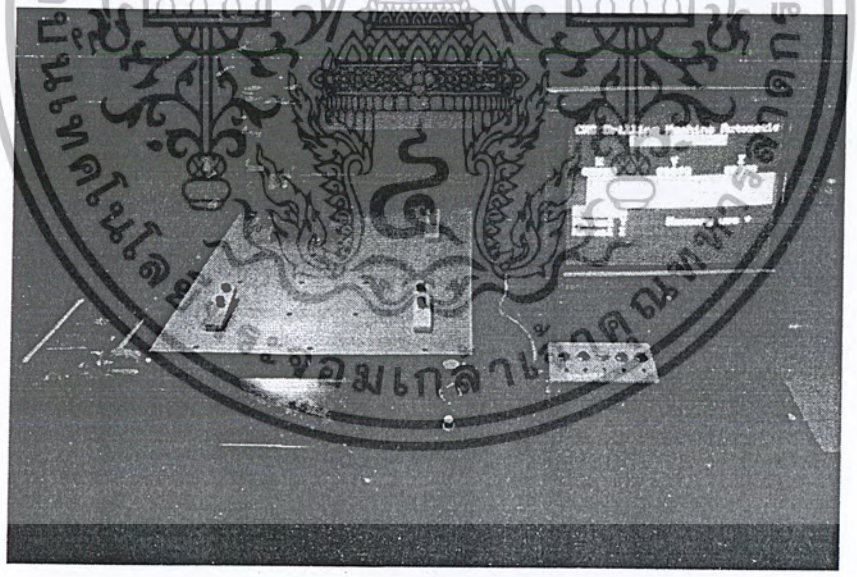
รูปที่ ก.3 Power Supply



รูปที่ ก.4 ปุ่มกดที่ใช้งาน



รูปที่ ก.5 หน้าจอที่ใช้แสดงผล



รูปที่ ก.6 เครื่องเจาะอัตโนมัติควบคุมด้วยไมโครคอนโทรลเลอร์

ภาคผนวก ข. Code

```
{
++++++++++++++++++++++++++++++++++++++++++++
 ***  DRIILLING MACHINE **********************
    15/03/2011
    // AE BIG PURE //

 ++++++++++++++++++++++++++++++++++++++++++++
*** OUTPUTS ****
Pin 0 WD LED..... watch dog blinks when processor is running
stepper outputs are active LOW
Pin 2    is OUT for Spindle
Pin 4    is OUT for X stepper motor
Pin 5    is OUT for X direction
Pin 6    is OUT for Y stepper motor
Pin 7    is OUT for Y direction
Pin 8    is OUT for Z stepper motor
Pin 9    is OUT for Z direction
Pin16-Pin23  are OUTs for VGA
Use main VGA screen to align PB's in order
     will be in order for other screens
Pin 24 PB w/ pull up    SELECT   [Far Right]
Pin 25 PB w/ pull up    DOWN
Pin 26 PB w/ pull up    UP
Pin 27 PB w/ pull up    RESET    [Far Left]
Pin 1 PB w/ pull up     MEMMORY

}
CON _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

VAR long  N1, Xstepper_stack[50] , Ystepper_stack[50] , Zstepper_stack[50]   ' COG Space
VAR long Item,Mam, Old_Item,Old_Mam,  base_color, sel_color, now_color  '  item selected on VGA
monitor
var LONG   WDvga, Z1' Watch dog flash to vga
VAR Long  Xstep , Ystep, Zstep   ' current motor postions
VAR long  Xtime, Ytime, Ztime     ' current motor speed
VAR Long  width1,width2,width3,Deep,Radius   ' Shared VAR Between shapes
Var long  Xcut, Ycut, Zcut         ' Target motor positions
VAR long  CX ,CY, CZ,A1,A2,A3,B1,B2,B3    'Moves X stepper motor   CY   'Moves Y stepper motor
CZ   'Moves Z stepper motor
VAR long  OKcut ' motors may be trned on in autiomatic
var long  Length1,Length2,Length3,spindle

OBJ
```

```
  NUM : "Simple_numbers"
  VGA : "vga_text"
  CIR : "ED_Trig"  'uses COG....  does math for circles

PUB Main ' inits Watch dog & cogs & VGA
 '++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ' set up Watch Dog  WD with just one counter ( takes NO processor time)
  Dira[0] := 1 ' WD output
  CtrB :=  %00010000000000000000000000000000 'MODE
  FrqB := 325    'WD flash Speed BIGGER is faster
 '++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

  cognew (Xstepper ,@Xstepper_stack) 'STARTS  COG for X stepper motor
  cognew (Ystepper ,@Ystepper_stack) 'STARTS  COG for Y stepper motor
  cognew (Zstepper ,@Zstepper_stack) 'STARTS  COG for Z stepper motor
  Cir.Start  'starts trig COG
  VGA.start(16)  ' SetUp VGA
  Start 'Run Mill Machine
 ' ++++++++++  END of MAIN  ++++++++++++++++++

PUB Start   'loop that keeps it all Happening
  'splash_SCREEN ' SHOW current version !!!!!!!!!!!!!!!!!!!!!!!!!!!!
  VGA.out($0C) 'ready to set color
  VGA.out(0)   ' Color
  VGA.str(string($0A,1,$0B,1,"CNC Drilling Machine Automatic",$0A,9,$0B,2,$C,7,"BY AE BIG
PURE"))
  VGA.str(string($0A,6,$0B,6,$C,6,"2000 steps = 1 Inch"))
   ' send PB discriptions  to VGA
  Item := $FF  ' UPDATE
  Sequencer      ' Select Item Helps Init main screen
  Item := 4
  Mam  := $FF

 Mam := 4
Repeat   ' MAIN Loop
     updateXYZ  'Display XYZ & spin DOG wheel  on VGA
     Sequencer
      ' Select Item updtae screen
     CheckPB    'read PB  (select)
     CheckMem
++++++++++++++++++++++++++++++++++++++++++++++++++++++

Pub Sequencer |  temp     ' Picks Item to HighLight
 base_color :=0
 sel_color := 1
```

```
    iF Old_Item == Item     ' No change
       Return

  temp := item   ' deselect old
  now_color := base_color
  Item:= Old_Item   ' deselect old
  Update_Item    ' DeSelect
  Item := Temp   ' Select New
  Now_color := sel_color
  Update_Item    ' Select New
  Old_Item := Item
RETURN
 '++++++++++++ END  SEQENCER ++++++++++++++++++++++++++++

PUB Update_Item

         IF Item== $FF
           Now_color := base_color
' Always update X Y Z
 UpdateXYZ   'vga Displ

 IF  Item  == 4    OR Item == $FF  ' $FF update ALL  for refresh
       VGA.out($0C) 'ready to set color
       VGA.out(now_color)   ' Color
       VGA.str(string($0A,1,$0B,9,"Manual X"))

 IF  Item  == 5    OR Item == $FF  ' $FF update ALL  for refresh
       VGA.out($0C) 'ready to set color
       VGA.out(now_color)   ' Color
       VGA.str(string($0A,1,$0B,10,"Manual Y"))

 IF  Item  == 6    OR Item == $FF  ' $FF update ALL  for refresh
       VGA.out($0C) 'ready to set color
       VGA.out(now_color)   ' Color
       VGA.str(string($0A,1,$0B,11,"Manual Z"))


 IF  Item  == 7    OR Item == $FF  ' $FF update ALL  for refresh
       VGA.out($0C) 'ready to set color
       VGA.out(now_color)   ' Color
       VGA.str(string($0A,15,$0B,10,"Memmory use ",$04 ))


RETURN  ' default back to calling Prgm
 'Item:= -200    ' set to rectangle Mode
 'Item:= -100    ' set to circle Mode
```

```
'++++++++++++++ END  update item ++++++++++++++++++++++++++++

PUB Xstepper   'Moves X stepper motor
 ' uses own COG
 ' 1100  0110  0011 1001 sequence
 CX :=0 ' starting DATA Location
 xtime :=20__000
  DirA[4] := %1 ' make outputs  TURN ON Motor sig
  Dira[5] := %1
 Repeat
   CX :=0
  Repeat

    If Item ==4         ' Must be in Manual X mode
      if INA[26] ==0   ' FORWARDS active LO  [5]
            Xfor    'Move motor by sequence of outpus ONE STEP per call
      if INA [25] ==0 ' BACKWARDS active LO [6]
            Xrev 'Move motor by sequence of outpus ONE STEP per call
      if ina [1] ==0
            Update_Mam
    if OKcut > 0
                Check_Pause  ' see if Pause // STOP // RESET  PB is pressed &  do it
      If OKcut == 1    ' CUT if 1 //  Pause if 3  //  STOP if 0
        if  Xstep < Xcut  ' FORWARDS active LO  [5]   OR AUTO Matic
              Xfor   'Move motor by sequence of outpus ONE STEP per call

        if  Xstep > Xcut ' BACKWARDS active LO [6]   OR AUTO Matic
              Xrev 'Move motor by sequence of outpus ONE STEP per call

   waitcnt (cnt + 1000) ' Motor settle TIME  ???? Necessary ?

   OutA[4] := %1111 '   TURN OFF motor sig  [Active LO steppers]

PUB Xfor   'Move motor by sequence of outpus ONE STEP per call
        outa[5] := 1
       CX := CX +1  'DATA INC
       Xstep++    'Display Position
      IF CX > 3
        CX:=0
      Outa [4]:=Step_Seq [CX]    ' MOVE Motor
      waitcnt ( Xtime + cnt) ' sets speed of stepper
'
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++
PUB Xrev     'Move motor by sequence of outpus ONE STEP per call
```

```
        outa[5] := 0
        CX := CX -1    'DATA  DEC
        Xstep--      'Display Position
        IF CX < 0
          CX:=3
       Outa [4]:=Step_Seq [CX]    ' MOVE Motor
       waitcnt ( Xtime + cnt) ' sets speed of stepper
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++

   return
PUB Ystepper 'CY   'Moves Y stepper motor
 ' uses own COG
 ' 1100  0110  0011 1001 sequence
     CY :=0 ' starting DATA Location
  xtime :=20__000
  DirA[6] := %1 ' make outputs  TURN ON Motor sig
  Dira[7] := %1
 Repeat
   CY :=0
  Repeat

      IF Item == 5  ' Must be in Manual Y mode
         if INA[26] ==0  ' FORWARDS active LO  [5]
           Yfor    'Move motor by sequence of outpus ONE STEP per call

         if INA [25] ==0  ' BACKWARDS active LO [6]
            Yrev     'Move motor by sequence of outpus ONE STEP per call
         if ina [1] ==0
            Update_Mam

     if OKcut > 0
                  Check_Pause  ' see if Pause // STOP // RESET  PB is pressed &  do it
      If OKcut ==1      ' CUT if 1 //  Pause if 3  //   STOP if 0
         if  Ystep < Ycut  ' FORWARDS active LO  [5]   OR AUTO Matic
                  Yfor    'Move motor by sequence of outpus ONE STEP per call

         if  Ystep > Ycut ' BACKWARDS active LO [6]   OR AUTO Matic
                  Yrev 'Move motor by sequence of outpus ONE STEP per call

    waitcnt (cnt + 1000) ' Motor settle TIME  ???? Necessary ?
     OutA[6] := %1111  '   TURN OFF motor sig  [Active LO steppers]
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++

PUB Yfor    'Move motor by sequence of outpus ONE STEP per call
```

```
        outa[7] := 1
        CY := CY +1  'DATA INC
        Ystep++    'Display Position
      IF CY > 3
        CY:=0
      Outa [6]:=Step_Seq [CY]   ' MOVE Motor
      waitcnt ( Xtime + cnt) ' sets speed of stepper

PUB Yrev     'Move motor by sequence of outpus ONE STEP per call
        outa[7] := 0
       CY := CY -1   'DATA  DEC
       Ystep--     'Display Position
       IF CY < 0
         CY:=3
      Outa [6]:=Step_Seq [CY]   ' MOVE Motor
      waitcnt ( Xtime + cnt) ' sets speed of stepper


PUB Update_Mam



  IF  Mam  == 4   OR Mam == $FF  ' $FF update ALL  for refresh
        A1 := Xstep
        B1 := Ystep
  IF  Mam  == 5   OR Mam == $FF  ' $FF update ALL  for refresh
        A2 := Xstep
        B2 := Ystep

  IF  Mam  == 6   OR Mam == $FF  ' $FF update ALL  for refresh
        A3 := Xstep
        B3 := Ystep


RETURN  ' default back to calling Prgm
 'Item:= -200    ' set to rectangle Mode
 'Item:= -100    ' set to circle Mode
'++++++++++++++ END  update item ++++++++++++++++++++++++++++
PUB Zstepper  ' CZ   'Moves Y stepper motor
' uses own COG
' 1100  0110  0011 1001 sequence
    CZ :=0 ' starting DATA Location
 xtime :=20__000
```

```
  DirA[8] := %1 ' make outputs  TURN ON Motor sig
  Dira[9] := %1
 Repeat
   CZ :=0
  Repeat

    if  Item == 6 ' Must be in Manual Z mode
         if INA[26] ==0   ' FORWARDS active LO  [5]
                    Zfor    'Move motor by sequence of outpus ONE STEP per call

        if INA [25] ==0  ' BACKWARDS active LO   [6]
                    Zrev     'Move motor by sequence of outpus ONE STEP per call
    if OKcut > 0
            Check_Pause  ' see if Pause // STOP // RESET  PB is pressed &  do it
      If OKcut ==1      ' CUT if 1 //  Pause if 3  //   STOP if 0
        if  Zstep < Zcut  ' FORWARDS active LO  [5]   OR AUTO Matic
                 Zfor    'Move motor by sequence of outpus ONE STEP per call

       if  Zstep > Zcut ' BACKWARDS active LO [6]   OR AUTO Matic
                Zrev 'Move motor by sequence of outpus ONE STEP per call
   waitcnt (cnt + 1000) ' Motor settle TIME  ???? Necessary ?
   OutA[8] := %1111  '   TURN OFF motor sig  [Active LO steppers]


PUB Zfor    'Move motor by sequence of outpus ONE STEP per call
        outa[9] := 1
       CZ := CZ +1  'DATA INC
       Zstep++     'Display Position
      IF CZ > 3
        CZ:=0
      Outa [8]:=Step_Seq [CZ]    ' MOVE Motor
      waitcnt ( Xtime + cnt) ' sets speed of stepper

PUB Zrev      'Move motor by sequence of outpus ONE STEP per call
        outa[9] := 0
        CZ := CZ -1    'DATA  DEC
        Zstep--      'Display Position
        IF CZ < 0
          CZ:=3
      Outa [8]:=Step_Seq [CZ]    ' MOVE Motor
      waitcnt ( Xtime + cnt) ' sets speed of stepper


PUB Pause_disp     ' Run / Pause  to VGA

      VGA.out($0C) 'ready to set color
```

```
      VGA.out(3)   ' Color

  RETURN
 '-------------------------------------------------------

 '++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
PUB UpdateXYZ   ' Refresh VGA Screen for XYZ only

 ' Always update X Y Z
         VGA.out($0C) 'ready to set color
         VGA.out(base_color)   ' Color
         VGA.str(string($0A,4,$0B,4,"X", $0A,2,$0B,5 ))
         VGA.str(num.decx (Xstep,5))
         VGA.str(string("   "))   ' Gost Hunter

         VGA.out($0C) 'ready to set color
         VGA.out(base_color)   ' Color
         VGA.str(string($0A,15,$0B,4,"Y",$0A,13,$0B,5))
         VGA.str(num.decx (Ystep,5))
         VGA.str(string("   "))   ' Gost Hunter

         VGA.out($0C) 'ready to set color
         VGA.out(base_color)   ' Color
         VGA.str(string($0A,25,$0B,4,"Z",$0A,23,$0B,5))
         VGA.str(num.decx (Zstep,5))
         VGA.str(string("   "))   ' Gost Hunter
       'spin wheel on VGA
    IF OKcut == 1    'display status
          VGA.str(string($0A,10,$0B,8,"**** Drilling ****"))

    IF OKcut == 3    'display status
          VGA.str(string($0A,10,$0B,8,"---- STOPPED ----"))
  RETURN  ' default back to calling Prgm
 '++++++++++++++ END  update XYZ  ++++++++++++++++++++++++++++

PUB CheckPB   | ttt
    Dira [27..24] := %0000  ' setup inputs for PBs

    If ina [27..24] == %1111  ' NO PB  Active Lo
       Return

    IF ina[24] == 0   ' Check for select PB  [D]
      waitcnt (1000 +cnt)
      repeat
        UpdateXYZ ' ??  keep XYZ Fresh
      until  ina [24] ==1      ' Debounce
```

```
    Item := Item +1  ' Next item
    if item >7
       item:=4
    RETURN
  IF ina[1] == 0   ' Check for select PB  [D]
    waitcnt (1000 +cnt)
    repeat
      Update_Mam ' ??  keep XYZ Fresh
    until  ina [1] ==1      ' Debounce
   Mam := Mam +1  ' Next item
    if Mam >6
       Mam:=4
    RETURN


 if item == 7
          Waitcnt (cnt + 50000)  ' De bounce ?
       repeat
             ' wait for key release
       Until InA[26] == 1
       Rectangle


  Return



 if ina[27] == 0  ' RESET PB   [4]
    Xstep :=0
    Ystep :=0
    Zstep :=0
  Return

RETURN  ' default back to calling Prgm
'-------------------------------------------------------------
PUB CheckMem   | ddd

  IF ina[1] == 0   ' Check for select PB  [D]
    waitcnt (1000 +cnt)
    repeat
      Update_Mam ' ??  keep XYZ Fresh
    until  ina [1] ==1      ' Debounce
   Mam := Mam +1  ' Next item
    if Mam >6
       Mam:=4
    RETURN
```

```
 RETURN  ' default back to calling Prgm

PUB Rectangle | i, ii ' Prep to auto rectagle

  Item:= -200     ' set to rectangle Mode
     VGA.out($0C) 'ready to set color
     VGA.out( 4)   ' Color center of screen
  repeat i from 7 to 13       ' clear center of screen
     VGA.out ($0A )  'X
     VGA.out (1 )
     VGA.out ($0B )   'Y
     VGA.out (i )
      repeat ii from 1  to  30   ' Do row
         VGA.out (" " )

  VGA.out($0C) 'ready to set color
  VGA.out(2)   ' Color
  VGA.str(string($0A,2,$0B,8, "   Auto Drilling mode   "))


     Xstep :=A3   'reset
     Ystep :=B3   'reset
     Zstep :=0    'reset
     Xcut  :=0    ' Reset
     Ycut  :=0    ' Reset
     Zcut  :=0    ' Reset

  Length1 := A1   ' INIT
  Length2 := A2   ' INIT
  Length3 := A3   ' INIT
  Width1  := B1   ' INIT
  Width2  := B2   ' INIT
  Width3  := B3   ' INIT
  Deep    :=-3500   ' INIT
  repeat until ina[24] == 0   ' Check for select PB  [D]
    VGA.str(string($0A,2,$0B,10,"L1 "))
    VGA.str(num.decx ( Length1, 5))

           'spin wheel on VGA  'slow down update rate
         waitcnt (cnt + 1_000_000)  'slow down update rate
           'spin wheel on VGA  'slow down update rate
       if ina [26] == 0   'UP PB  [5]
          length1 := length1 + 1
```

```
        if ina[25] == 0  ' Down PB   [6]
             if Length1 >0
               Length1 := Length1 -1
'.....................................................
     waitcnt (cnt + 1_000_000)  'slow down
     repeat

     until ina[24] == 1   ' Check for select PB  to be RELEASED
'.....................................................
     repeat until ina[24] == 0   ' Check for select PB  [D]
       VGA.str(string($0A,11,$0B,10,"L2 "))
       VGA.str(num.decx ( Length2,5))

                'spin wheel on VGA  'slow down update rate
              waitcnt (cnt + 1_000_000)  'slow down update rate
                'spin wheel on VGA  'slow down update rate
            if ina [26] == 0   'UP PB  [5]
              length2 := length2 + 1

        if ina[25] == 0  ' Down PB   [6]
             if Length2 >0
               Length2 := Length2 -1
'.....................................................
     waitcnt (cnt + 1_000_000)  'slow down
     repeat

     until ina[24] == 1   ' Check for select PB  to be RELEASED
.....................................................
     repeat until ina[24] == 0   ' Check for select PB  [D]
       VGA.str(string($0A,20,$0B,10,"L3 "))
       VGA.str(num.decx ( Length3,5))

                'spin wheel on VGA  'slow down update rate
              waitcnt (cnt + 1_000_000)  'slow down update rate
                'spin wheel on VGA  'slow down update rate
            if ina [26] == 0   'UP PB  [5]
              length3 := length3 + 1

        if ina[25] == 0  ' Down PB   [6]
             if Length3 >0
               Length3 := Length3 -1
.....................................................
     waitcnt (cnt + 1_000_000)  'slow down
     repeat
```

```
    until ina[24] == 1   ' Check for select PB  to be RELEASED
'.....................................................
    repeat until ina[24] == 0   ' Check for select PB  [D]
      VGA.str(string($0A,2,$0B,11,"W1 "))
      VGA.str(num.decx ( width1,5))

            'spin wheel on VGA  'slow down update rate
          waitcnt (cnt + 1_000_000)  'slow down update rate
            'spin wheel on VGA  'slow down update rate
       if ina [26] == 0   'UP PB  [5]
            Width1 := width1 + 1

       if ina[25] == 0  ' Down PB   [6]
          if Width1 >0
            Width1 := Width1 -1
'.....................................................
    waitcnt (cnt + 1_000_000)
    repeat
   ' Check for select PB  to be RELEASED
    until ina[24] == 1   ' Check for select PB  to be RELEASED
    repeat until ina[24] == 0   ' Check for select PB  [D]
      VGA.str(string($0A,11,$0B,11,"W2 "))
      VGA.str(num.decx ( width2,5))

            'spin wheel on VGA  'slow down update rate
          waitcnt (cnt + 1_000_000)  'slow down update rate
            'spin wheel on VGA  'slow down update rate
       if ina [26] == 0   'UP PB  [5]
            Width2 := width2 + 1

       if ina[25] == 0  ' Down PB   [6]
          if Width2 >0
            Width2 := Width2 -1
'.....................................................
    waitcnt (cnt + 1_000_000)
    repeat
   ' Check for select PB  to be RELEASED
    until ina[24] == 1   ' Check for select PB  to be RELEASED
    repeat until ina[24] == 0   ' Check for select PB  [D]
      VGA.str(string($0A,20,$0B,11,"W3 "))
      VGA.str(num.decx ( width3,5))

            'spin wheel on VGA  'slow down update rate
          waitcnt (cnt + 1_000_000)  'slow down update rate
            'spin wheel on VGA  'slow down update rate
       if ina [26] == 0   'UP PB  [5]
```

```
            Width3 := width3 + 1

      if ina[25] == 0  ' Down PB   [6]
           if Width3 >0
             Width3 := Width3 -1
'.....................................................
   waitcnt (cnt + 1_000_000)
   repeat
  ' Check for select PB  to be RELEASED
   until ina[24] == 1   ' Check for select PB  to be RELEASED
'.....................................................
   repeat until ina[24] == 0   ' Check for select PB  [D]
    VGA.str(string($0A,6,$0B,12," Enter  DEEP "))
    VGA.str(num.decx (Deep,5))
              'spin wheel on VGA  'slow down update rate
           waitcnt (cnt + 1_000_000)  'slow down update rate
              'spin wheel on VGA  'slow down update rate
      if ina [26] == 0   'UP PB  [5]
            Deep := deep + 1

      if ina[25] == 0  ' Down PB   [6]
           if deep >0
             Deep := Deep -1
 repeat
  ' Check for select PB  to be RELEASED
 until ina[24] == 1   ' Check for select PB  to be RELEASED
VGA.str(string($0A,2,$0B,13,"  press SELECT to Drilling"))
.....................................................
   repeat
       updateXYZ  'Display XYZ & spin DOG wheel  on VGA
   until ina[24] == 0   ' Check for select PB  to be PRESSED

   repeat

   until ina[24] == 1   ' Check for select PB  to be RELEASED
.....................................................
.....................................................
 VGA.str(string($0A,2,$0B,13,"                    ")) ' Blank Previos Message
 VGA.out($0C) 'ready to set color
 VGA.out ($03)
 OKcut :=3  'stop // ON permission for AUTO FEED
 Pause_disp     ' Header to VGA
.....................................................
  CUT_Rect  ' Moves cutter for rectangle
set up for multiple passes
Repeat
```

```
        if InA [24] == 0
            Pause_disp    ' Run / Pause  to VGA
            OKcut :=1  'ON permission for AUTO FEED
            CUT_Rect  ' Moves cutter for rectangle
            OKcut :=3  'stop // ON permission for AUTO FEED
                VGA.str(string($0A,3,$0B,8,"                    "))  ' Blank Line
                VGA.str(string($0A,2,$0B,9,$0C, 3,"     %% Antoher PASS %%   "))
                VGA.out($0C) 'ready to set color
                VGA.out(3)   ' Color

    UpdateXYZ '  keep XYZ & Dog Fresh
  until InA [27] == 0   OR OKcut == 0      ' in 27 is "done" PB
 '-----------------------------------------
  ' RAise cutter to clear workpiece
  OKcut :=0  'OFF  permission for AUTO FEED
'------------ DONE move cutter ---------------------------------
  VGA.str(string($0A,5,$0B,8,"%% DONE Drill %%"))


 repeat
          UpdateXYZ '  keep XYZ & Dog Fresh
   IF ina[27] == 0   ' Check for ---- PB  GET out of Rectangle  mode
       repeat
         UpdateXYZ ' ??  keep XYZ Fresh
           'spin wheel on VGA
       until  ina [24] ==1      ' Debounce

     VGA.out($0C) 'ready to set color
     VGA.out( base_color)   ' Color center of screen
    repeat i from 7 to 13      'NORMAL center of screen
      VGA.out ($0A )  'X
      VGA.out (1 )
      VGA.out ($0B )   'Y
      VGA.out (i )
      repeat ii from 1  to  30   ' Do row
          VGA.out (" " )
    Item:= $FF     ' Done with Rectangle  Mode  REFRESH main screen
          '  to VGA
    Return   ' Done with Rectangle
 ------------   END  ---------------------
PUB CUT_Rect   ' Moves cutter for rectangle
  dira[2] := %1
------------ START move cutter ----------------------------------
------------ START move cutter ----------------------------------
  outa[2] := 0
  Xcut := Length1 'turn on cutter
```

```
  Repeat
       ' Wait while other COG moves motor
        UpdateXYZ    ' Refresh VGA Screen for XYZ only
  Until Xcut == Xstep   OR OKcut ==0
 Ycut := Width1 'turn on cutter
  Repeat
       ' Wait while other COG moves motor
       UpdateXYZ    ' Refresh VGA Screen for XYZ only
  Until Ycut == Ystep  OR OKcut ==0
outa[2] := 1
waitcnt (cnt + 80_000_000)
Zcut := Deep
  Repeat
      UpdateXYZ
  Until Zcut == Zstep   OR OKcut ==0
Zcut := Zstep - Deep
  Repeat
       UpdateXYZ
  Until Zcut == Zstep   OR OKcut ==0
outa[2] := 0
Xcut :=  Length2 'turn on cutter MOVE BACKWARDS
  Repeat
       ' Wait while other COG moves motor
       UpdateXYZ    ' Refresh VGA Screen for XYZ only
  Until Xcut == Xstep  OR OKcut ==0

Ycut := Width2 'turn on cutter
  Repeat
       ' Wait while other COG moves motor
       UpdateXYZ    ' Refresh VGA Screen for XYZ only
  Until Ycut == Ystep   OR OKcut ==0
 outa[2] := 1
 waitcnt (cnt + 80_000_000)
Zcut := Deep
  Repeat

      UpdateXYZ
  Until Zcut == Zstep   OR OKcut ==0
Zcut := Zstep - Deep
  Repeat

       UpdateXYZ
  Until Zcut == Zstep   OR OKcut ==0
  outa[2] := 0
Xcut := length3 'turn on cutter MOVE BACKWARDS
  Repeat
```

```
        ' Wait while other COG moves motor
        UpdateXYZ    ' Refresh VGA Screen for XYZ only
    Until Xcut == Xstep  OR OKcut ==0

  Ycut := Width3 'turn on cutter
    Repeat
        ' Wait while other COG moves motor
        UpdateXYZ    ' Refresh VGA Screen for XYZ only
    Until Ycut == Ystep   OR OKcut ==0
    outa[2] := 1
   waitcnt (cnt + 80_000_000)
  Zcut := Deep
    Repeat

       UpdateXYZ
    Until Zcut == Zstep   OR OKcut ==0
  Zcut := Zstep - Deep
    Repeat

       UpdateXYZ
    Until Zcut == Zstep   OR OKcut ==0
    outa[2] := 0
  Xcut := Xstep - length3 'turn on cutter MOVE BACKWARDS
    Repeat
        ' Wait while other COG moves motor
        UpdateXYZ    ' Refresh VGA Screen for XYZ only
    Until Xcut == Xstep  OR OKcut ==0

  Ycut := Ystep - Width3 'turn on cutter
    Repeat
        ' Wait while other COG moves motor
        UpdateXYZ    ' Refresh VGA Screen for XYZ only
    Until Ycut == Ystep   OR OKcut ==0
' done elselware  OKcut :=0  'OFF  permission for AUTO FEED
  ---------- DONE ----------
PUB Check_Pause  ' see if Pause // STOP    PB is pressed & Pause if SEW

     If InA [24..27] ==%1111  ' NO keys pressed
          REturn

     if InA [24] == 0    ' PAUSE/ run
          Waitcnt (cnt + 50000)  ' De bounce
          If  OKcut == 3        ' GO from STOP  to RUN
               OKcut := 1       ' CUT if 1 //  Pause if 3  //   STOP if 0
          repeat
            ' wait for key release
```

```
        Until InA[24] == 1

    if InA [25] == 0   ' STOP ..... TOGGLE type PB
        OKcut := 3      ' CUT if 1 // Pause if 3 //  STOP if 0

    if InA [26] == 0   ' DONE stop ..... all 3 Axis
          OKcut := 3     ' CUT if 1 // Pause if 3 //  STOP if 0
          Xcut := Xstep  ' ALSO Stops cutting
          Ycut := Ystep
          Zcut := Zstep
      Waitcnt (cnt + 50000)  ' De bounce ?
        repeat
         ' wait for key release
        Until InA[26] == 1
       OKcut := 0  'clear out old stuff   ' CUT if 1 // Pause if 3 //  STOP if 0

    if InA [27] == 0   ' ABORT  END ..... all 3 Axis
          OKcut := 3     ' CUT if 1 // Pause if 3 //  STOP if 0
          Xcut := Xstep
          Ycut := Ystep  ' !!!!!  NO Debounce so that multi resets send it back to MAIN Screen
          Zcut := Zstep
          Waitcnt (cnt + 50000) ' xtra time to sort stuff out
          OKcut := 0     ' CUT if 1 // Pause if 3 //  STOP if 0
 ' !!!!!  NO Debounce so that multi resets send it back to MAIN Screen
 ' <<<<<<<<<<<<<<<  END Check Pause >>>>>>>>>>>>>>>>>>>>

DAT
    Step_Seq Byte   %1100,%0011,%1100,%0011  ' to motor

' HALF STEPS  @@@@@@@@@@@@@@@@  NOT used  just referance

  Half_Step Byte %1000, %1100, %0100, %0110, %0010, %0011, %0001, %1001     ' to motor
```

เอกสารอ้างอิง

1.คู่มือการใช้งานมัลติคอร์ไมโครคอนโทรลเลอร์

2 . www.parallax.com

3 . www.cncroom.com

4 . www.cncinformation.com