ระบบหุ่นยนต์

ROBOTICS

โดย

นางสาวพันธิตรา ชัชวาลไกรวงศ์

นายวรเทพ กิตตินภากุล

นายวิศรุต ธรรมทัตโต

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบหุ่นยนต์

ROBOTICS

ผู้จัดทำ

น.ส.พันธิตรา	ชัชวาลไกรวงศ์	46010512
นายวรเทพ	กิตติดินภากุล	46010649
นายวิศรุต	ธรรมทัตโต	46010727

..... อาจารย์ที่ปรึกษา

(ผศ.สุมิตร พนาอุดมทรัพย์)

ระบบหุ่นยนต์

โดย

นางสาวพันธิตรา	ชัชวาลไกรวงศ์	46010512
นายวรเทพ	กิตตินภากุล	46010649
นายวิศรุต	ธรรมทัตโต	46010727

อาจารย์ที่ปรึกษา

ผศ.สุมิตร พนาอุดมทรัพย์

ปีการศึกษา 2549

บทคัดย่อ

ในปัจจุบันมีการใช้อุปกรณ์ที่สามารถทำงานแทนมนุษย์มากขึ้นเรื่อยๆ แขนกลเป็นอุปกรณ์ประเภทหนึ่งที่ได้รับความนิยม ปริญญานิพนธ์ฉบับนี้จึงเลือกแขนกลแบบอาร์ติคูเลตเป็นกรณีศึกษา นับตั้งแต่การเลือกแบบโครงสร้างและการวิเคราะห์ทฤษฎีการเคลื่อนที่ของหุ่นยนต์ จากนั้นนำมาเขียนเป็นโปรแกรม Visual Basic เพื่อไปควบคุมการเคลื่อนที่ของแขนกลให้เป็นไปตามที่ต้องการ เพื่อการประยุกต์ใช้ต่อไป

ROBOTICS

By

Miss Phanthitra Chatchawankraiwong 46010512

Mr.Worrathep Kittinapakul 46010649

Mr.Wisarut Thammathatto 46010727

Advisor

Asst.Prof. Sumit Panaudomsup

Academic Year 2006

ABSTRACT

Robot, the one of equipment is more efficiently than human, become popular at this time. Articulated Robot is chosen to be a case study. Begin we had chosen the model and analysis the kinematics of robot then made the control program by Visual Basic 6.0 to control robot for the advance application.

กิตติกรรมประกาศ

โครงงานและปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปด้วยดี ต้องขอขอบพระคุณคณาจารย์และท่านผู้มีพระคุณที่ให้คำแนะนำ และคำปรึกษาในการทำโครงงานนี้

ขอบคุณท่านอาจารย์ที่ปรึกษา คืออาจารย์สุมิตร พนาอุดมทรัพย์ ที่ให้ความกรุณาช่วยแนะนำแนวทางและตรวจสอบแก้ไขแนวความคิดที่ผิดพลาดต่างๆ จนโครงงานนี้สามารถบรรลุถึงวัตถุประสงค์ที่ตั้งไว้ได้

ขอบคุณเพื่อนๆ ที่ให้กำลังใจ สนับสนุนอุปกรณ์ที่ขาดเหลือและช่วยเหลือในการทำงานในครั้งนี้จนสำเร็จ

สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา และครอบครัว ที่คอยให้กำลังใจและให้การสนับสนุนในการทำโครงงานครั้งนี้ สุดที่ทำให้โครงงานนี้สำเร็จสมบูรณ์ลงได้

ผู้จัดทำ
น.ส.พันธิตรา ชัชวาลไกรวงศ์
นายวรเทพ กิตตินภากุล
นายวิศรุต ธรรมทัตโต

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงการ

ระบบการผลิตในอุตสาหกรรมประเภทต่างๆ ในปัจจุบันนี้ต้องการความรวดเร็ว แม่นยำในการผลิต ที่สำคัญก็คือ การลดต้นทุนในการผลิตลงเพื่อให้สามารถที่จะทำการแข่งขันกับคู่แข่งในท้องตลาดได้ จากเดิมจะใช้คนเป็นหลัก ต่อมาได้มีการนำเข้าเครื่องจักรเข้ามาช่วยในการผลิตรวมถึงงานที่มีอัตราการเสี่ยงต่อการเกิดอุบัติเหตุ ดังนั้นแขนกลจึงมีการนำเข้ามาใช้ในอุตสาหกรรมการผลิตมากขึ้นในปัจจุบัน จึงมีการพัฒนาคุณภาพของหุ่นยนต์ควบคู่ไปด้วย โดยที่หุ่นยนต์ที่ใช้ในปัจจุบันได้มีการนำเข้าจากต่างประเทศ ซึ่งมีราคาสูง ก่อให้เกิดปัญหาในการพัฒนาและการศึกษาวิจัยถึงประสิทธิภาพในการใช้งานของหุ่นยนต์อย่างยิ่ง

โครงงานระบบหุ่นยนต์ จัดทำขึ้นเพื่อศึกษาพื้นฐานหลักการทำงานของแขนกล โดยให้แขนกลสามารถวาดภาพตามที่ต้องการได้ โดยการทำงานเป็นไปอย่างอัตโนมัติ ซึ่งโครงงานแขนกลนี้อาจทำงานได้ไม่เป็นไปตามขอบเขตที่ตั้งไว้ ซึ่งความรู้ ความเข้าใจในหลักการของแขนกลในการวิจัยศึกษา และหวังว่าจะเป็นประโยชน์ต่อผู้สนใจระบบการทำงานของแขนกลต่อไป

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 ออกแบบและสร้างแขนกลวาดภาพ

1.3 ขอบเขตของโครงงาน

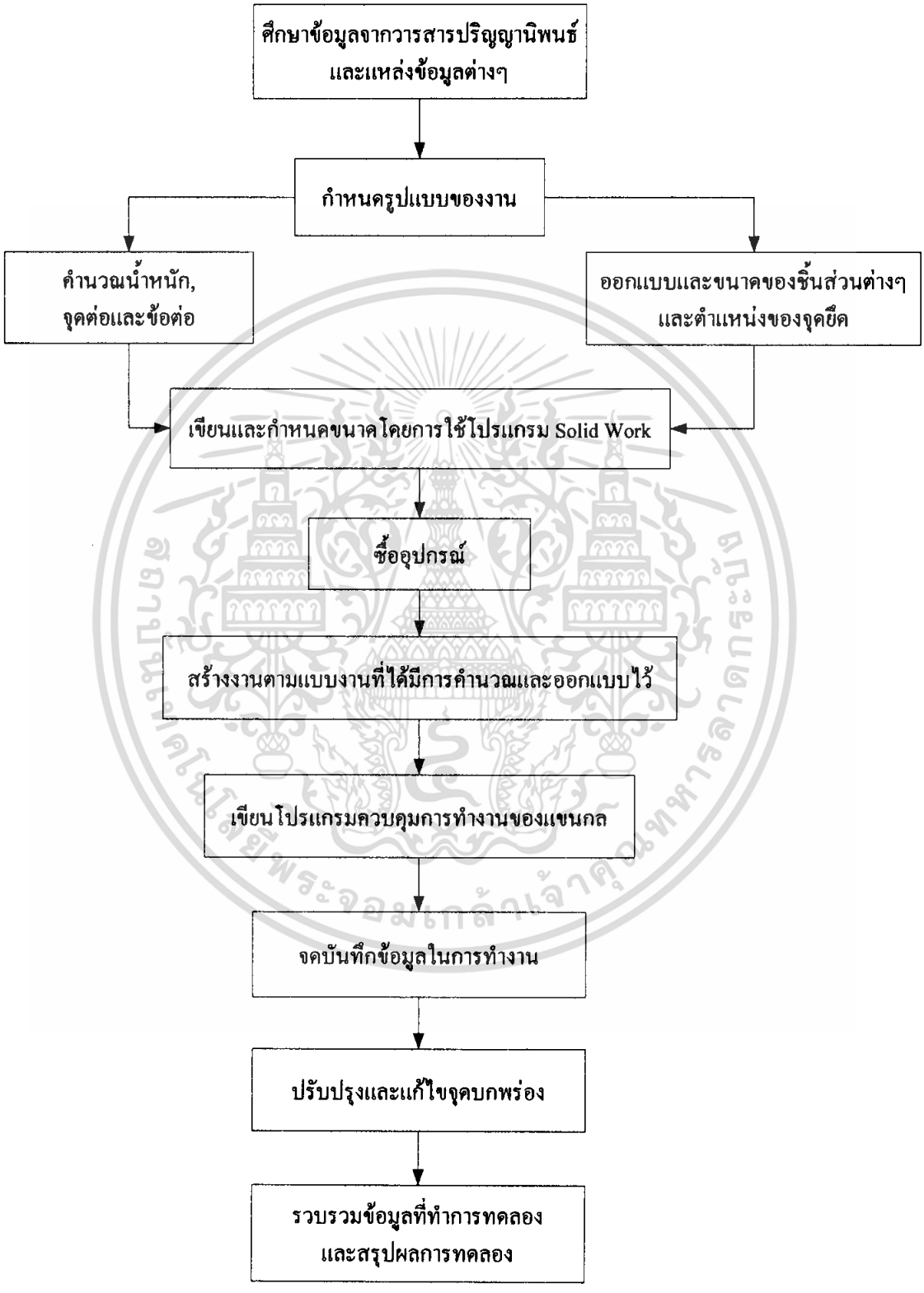
1.3.1 สร้างแขนกลแบบอาร์ติคูเลตแนวตั้ง และศึกษาการเคลื่อนที่ของแขนกล โดยสามารถวาดภาพในกระดาษขนาด A4 ซึ่งวางอยู่ในแนวราบ

1.3.2 ทำการศึกษาความสัมพันธ์ระหว่างสัญญาณพัลส์ที่ป้อนให้เซอร์โวมอเตอร์กับการหมุนของเซอร์โวมอเตอร์ เพื่อนำไปใช้ในการเคลื่อนที่ของแขนกล

1.3.3 สร้างหน้าต่างสำหรับวาดภาพบนหน้าจอคอมพิวเตอร์โดยใช้โปรแกรม Visual Basic

1.3.4 เขียนโปรแกรมควบคุมแขนกลให้สามารถวาดภาพในกระดาษ A4 ได้อย่างอัตโนมัติ ซึ่งการเชื่อมต่อระหว่างคอมพิวเตอร์กับแขนกล จะเชื่อมต่อผ่านทางพอร์ตอนุกรม

1.4 ขั้นตอนการศึกษาและการจัดทำโครงงาน



1.5 รายละเอียดปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวนำถึงวัตถุประสงค์ ขั้นตอนการศึกษา และการจัดทำโครงงาน พร้อมทั้งรายละเอียดของปริญญานิพนธ์ของแต่ละบท

บทที่ 2 ทฤษฎีและความรู้ที่เกี่ยวข้อง กล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องในการออกแบบแขนกล คุณสมบัติของ dsPIC การใช้งานพอร์ตอนุกรม และนำเอาความรู้ไปประยุกต์ใช้ในการจัดทำโครงงาน

บทที่ 3 หลักการออกแบบ นำเสนอโครงสร้างของแขนกล และหน้าต่างสำหรับวาดภาพบนหน้าจอคอมพิวเตอร์

บทที่ 4 การทดลองและผลการทดลอง เป็นการทดสอบองค์ประกอบต่างๆ ของแขนกล

บทที่ 5 บทวิจารณ์และสรุป จะสรุปผลการดำเนินงาน ปัญหาที่เกิดขึ้น และแนวทางการปรับ ปรุงพัฒนาโครงงานนี้ต่อไป

บทที่ 2
ทฤษฎีและความรู้ที่เกี่ยวข้อง

2.1 ความหมายของหุ่นยนต์

เป็นคำมาจากภาษา Czech (ภาษาของประเทศเชคโกสโลวาเกีย ซึ่งแปลว่างาน) และตามพจนานุกรมอังกฤษของ Webster แปลว่า an automatic device that perform functions ordinary described to human being ซึ่งมีความหมายกว้างมากเหมือนกันจึงได้มีการให้คำจำกัดความของ Robot หรือหุ่นยนต์อุตสาหกรรมเป็นภาษาอังกฤษว่า

A Robot is a reprogrammable multifunction al manipulator designed to move material part tools or specialized devices through variable programmed motions for the performance of a varity of tasked ซึ่งแปลว่า (เครื่องจักรที่สามารถตั้งโปรแกรมได้หลายๆ ครั้ง และสามารถปฏิบัติงานได้หลายๆ หน้าที่ โดยหุ่นยนต์ได้รับออกแบบเพื่อให้หยิบ จับหรือเคลื่อนย้ายวัตถุอุปกรณ์ เครื่องมือ หรือ เครื่องใช้พิเศษต่างๆ โดยอาศัยการควบคุมโปรแกรมในการเคลื่อนที่ของมันให้ทำงานหลายๆ อย่างตามต้องการ)

2.2 การแบ่งชนิดของหุ่นยนต์

การแบ่งชนิดของหุ่นยนต์ออกตามข้อต่อแบ่งได้ 5 ประเภทดังนี้

ประเภทที่ 1 คือ หุ่นยนต์แบบคาร์ทีเซียน (Cartesian)

ประเภทที่ 2 คือ หุ่นยนต์แบบทรงกระบอก (Cylindrical)

ประเภทที่ 3 คือ หุ่นยนต์แบบทรงกลม (Spherical)

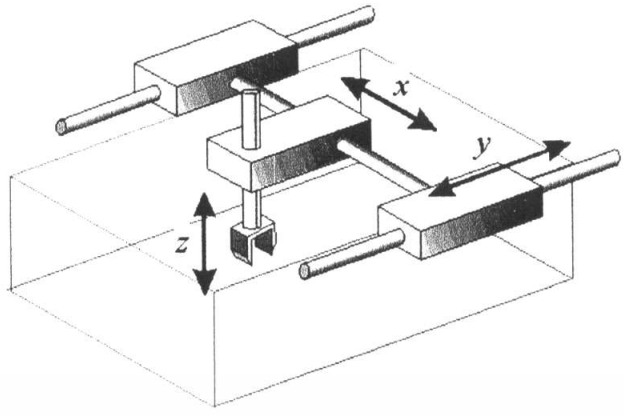
ประเภทที่ 4 คือ หุ่นยนต์แบบอาร์ติคูเลตแนวนอน (Horizontal articulated)

ประเภทที่ 5 คือ หุ่นยนต์แบบอาร์ติคูเลตแนวตั้ง (Vertical articulated)

2.2.1 หุ่นยนต์แบบคาร์ทีเซียน (Cartesian)

แขนกลของหุ่นยนต์ประเภทนี้มีข้อต่อเป็นแบบพริสเมติก (Prismatic) ทั้ง 3 ชิ้นส่วน โดยดู (จากรูป 2.1) และด้วยเหตุนี้จึงเรียกรหัสของหุ่นยนต์คาร์ทีเซียนว่า PPP

หุ่นยนต์ประเภทคาร์ทีเซียน (Cartesian Robot) ที่มีลักษณะขอบเขตการทำงานที่แคบ แต่มีระดับความตายตัวในทางกลไกนั้นสูงมาก และมีความสามารถในการวางตำแหน่งเอนเอฟเฟคเตอร์ (End Effector) ได้อย่างแม่นยำและเที่ยงตรงสูงมาก ดังนั้นหุ่นยนต์แบบคาร์ทีเซียนจึงเหมาะสมกับงานที่ต้องการความละเอียดในการวัดขนาดและงานผลิตเครื่องมือ

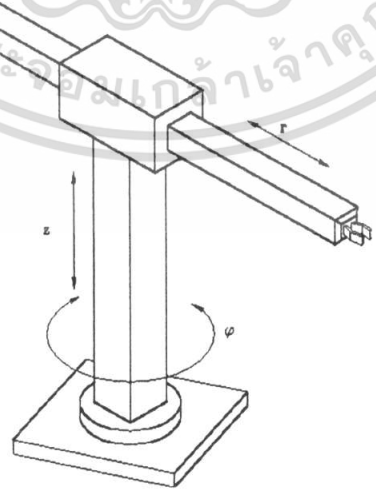


รูปที่ 2.1 ลักษณะท่อนแขน (Link) ของหุ่นยนต์แบบคาร์ทีเซียน

การควบคุมหุ่นยนต์แบบคาร์ทีเซียน (Cartesian Robot) นั้นทำได้ง่ายเนื่องจากการเคลื่อนที่ของท่อนแขนต่างๆ นั้นมีการเคลื่อนที่ในแนวเส้นตรงและมีชิ้นงานนั้นวางตามตำแหน่งไว้คงที่ ทำให้มีโมเมนต์ความเฉื่อย (Moments of inertia) นั้นคงที่ในตำแหน่งเดียวกันตลอดในระยะของการทำงาน

2.2.2 หุ่นยนต์แบบทรงกระบอก (Cylindrical Robots)

แขนกลของหุ่นยนต์ประเภทนี้ประกอบขึ้นจากข้อต่อแบบรีโวลูท (Revolute joint) จำนวนหนึ่งชิ้นและข้อต่อแบบพริสเมติก (Prismatic joint) อีกจำนวน 2 ชิ้น โดยรหัสของแขนกลประเภทนี้คือ RPP ดูตัวอย่างในรูปที่ 2.2

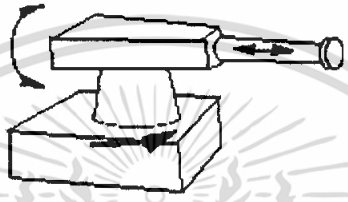


รูปที่ 2.2 ลักษณะของหุ่นยนต์แบบทรงกระบอก

2.2.3 หุ่นยนต์แบบทรงกลม (Spherical Robots)

แขนของหุ่นยนต์ประเภทนี้ประกอบขึ้นจากข้อต่อแบบรีโวลูบ (Revolute joint) จำนวน 2 ชิ้น และข้อต่อแบบพริสเมติก (Prismatic joint) อีก 1 ชิ้น ดังนั้นแขนกลประเภทนี้จึงมีรหัสเป็นแบบ RRP ดูตัวอย่างในรูปที่ 2.3

หุ่นยนต์ประเภทนี้มีขอบเขตของการทำงานที่กว้างขวางกว่าแบบทรงกระบอก (Cylindrical Robot) และระดับความตายตัวในทางกลไกนั้นน้อยกว่าแบบทรงกระบอก เพราะว่าข้อต่อ 2 ส่วนแรกนั้นเป็นส่วนที่ทำให้เกิดการเคลื่อนที่แบบหมุน



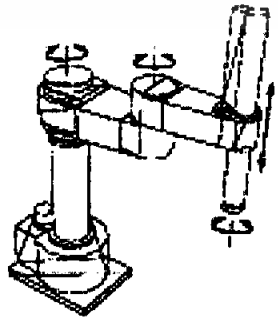
รูปที่ 2.3 ชนิดของหุ่นยนต์แบบทรงกลม

2.2.4 หุ่นยนต์แบบอาร์ติคูเลตแนวนอน (Horizontal Articulated Robot)

แขนของหุ่นยนต์ประเภทนี้ประกอบจากข้อต่อแบบรีโวลูท (Revolute joint) จำนวน 2 ชิ้น และมีข้อต่อแบบพริสเมติก (Prismatic joint) อยู่ 1 ชิ้น รหัสของแขนกลประเภทนี้คือ RRP ดูตัวอย่างในรูปที่ 2.4

ขอบเขตการทำงานของแขนหุ่นยนต์ประเภทนี้จะแคบกว่าขอบเขตการทำงานของหุ่นยนต์แบบทรงกลม (Spherical Robot) แต่จะมีขอบเขตการทำงานที่กว้างกว่าแบบคาร์ทีเซียน (Cartesian Robot) และแบบทรงกระบอก (Cylindrical Robot)

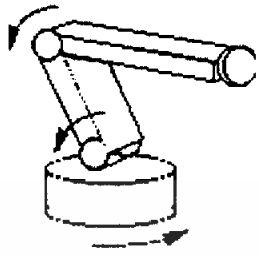
หุ่นยนต์ประเภทนี้มีส่วนประกอบของการทำงานที่พิเศษเนื่องจากมีการเคลื่อนที่เป็นเส้นตรงเป็นแนวตั้งในแกนทั้ง 3 แกน



รูปที่ 2.4 แสดงลักษณะหุ่นยนต์แบบอาติคูเลตแนวนอน

2.2.5 หุ่นยนต์แบบอาร์ติคูเลตแนวตั้ง (Vertical Articulated Robots)

แขนของหุ่นยนต์ประเภทนี้ประกอบด้วยข้อต่อแบบรีโวลูท (Revolute joint) ทั้ง 2 ชิ้น รหัสของหุ่นยนต์ประเภทนี้คือ RRR ดูตัวอย่างในรูปที่ 2.5



รูปที่ 2.5 ลักษณะของหุ่นยนต์แบบอาร์ติคูเลตแนวตั้ง

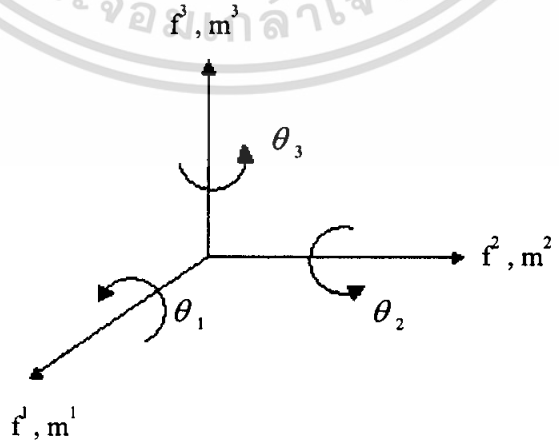
2.3 ทฤษฎีเกี่ยวกับแขนกล

2.3.1 การหมุน

ในการระบุตำแหน่ง และการหมุนของ End-effection ในเทอมของโครงพิกัดซึ่งติดอยู่กับฐานที่นิ่ง การถ่ายโอนจึงเกี่ยวข้องกับการหมุนและการเคลื่อนที่เปลี่ยนตำแหน่ง

2.3.1.1 การหมุนพื้นฐาน (Fundamental Rotation)

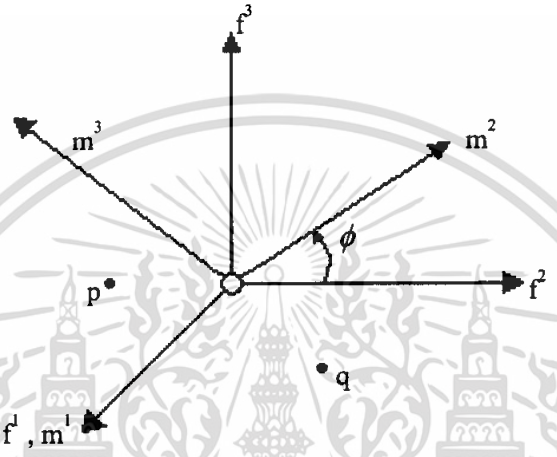
ถ้าโครงพิกัด M ได้จากการหมุน M รอบยูนิตเวคเตอร์อันหนึ่งของโครงพิกัดอยู่นิ่ง F ดังนั้นเมตริกซ์ถ่ายโอนพิกัดที่ได้เรียกว่า เมตริกซ์การหมุนพื้นฐานในปริภูมิ R^3 มีทางเป็นไปได้ 3 ทาง ดังแสดงในภาพที่ 2.6



รูปที่ 2.6 การหมุนพื้นฐานใน R^3

ถ้าหมุนโครงพิกัด M รอบแกน f^1 ของโครงพิกัดอยู่นิ่ง F ให้เป็นองศาการหมุนจะทำการวัดตามกฎมือขวา ดังภาพที่ 2.7 ให้ $R^3(\phi)$ เป็นเมตริกซ์ถ่ายโอนพิกัดช่วง M ไป F และจะได้

$$R^1(\phi) = \begin{bmatrix} f^1m^1 & f^1m^2 & f^1m^3 \\ f^2m^1 & f^2m^2 & f^2m^3 \\ f^3m^1 & f^3m^2 & f^3m^3 \end{bmatrix} \tag{2.1}$$



รูปที่ 2.7 การหมุนโครงพิกัด M รอบแกน f^1

เนื่องจากเป็นการหมุนรอบแกน f^1 ดังนั้นจะได้ว่า $f^1 = m^1$ และจากทฤษฎีบทของการหมุนที่ว่า เมื่อให้ x และ y เป็นเวคเตอร์ใน R และ ϕ เป็นหมุนจาก x ไป y ดังนั้น

$$x \cdot y = x \cdot y \cos\phi \tag{2.2}$$

ดังนั้นจะได้

$$R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.3}$$

ในทำนองเดียวกัน ถ้า $R^2(\phi)$ และ $R^3(\phi)$

$$R^2(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{2.4}$$

$$R^3(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

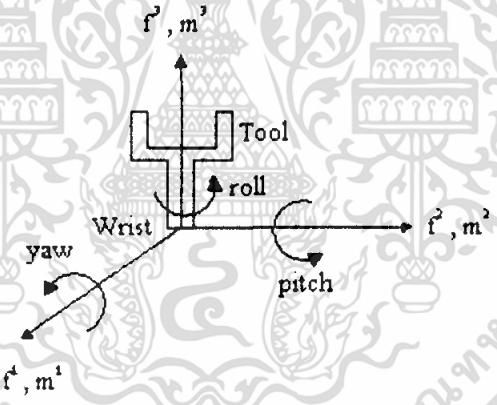
2.3.1.2 การหมุนผสม (Composite Rotations)

เมื่อทำการคูณเมตริกซ์การหมุนพื้นฐานเข้าด้วยกัน เมตริกซ์ลัพธ์ที่ได้นั้นจะแสดงถึงลำดับของการหมุนรอบยูนิตเวคเตอร์ เรียกเมตริกซ์ที่คูณกันนี้ว่าการหมุนผสม เมื่อพิจารณาภาพเครื่องมือ (End-effects) แสดงอยู่ในรูปที่ 2.8 โครงพิกัดเคลื่อนที่ $M = \{m^1 \quad m^2 \quad m^3\}$ จะหมุนไปกับเครื่องมือ ในขณะที่โครงพิกัดอยู่นิ่ง $F = \{f^1 \quad f^2 \quad f^3\}$ ติดอยู่ที่โคนของเครื่องมือ ดังนั้นจะได้การหมุนพื้นฐาน 3 ลักษณะคือ

yaw เป็นการหมุนรอบแกน f^1

pitch เป็นการหมุนรอบแกน f^2

roll เป็นการหมุนรอบแกน f^3



รูปที่ 2.8 yaw pitch และ roll ของเครื่องมือ

การหมุนพื้นฐานแต่ละอันแทนด้วยเมตริกซ์ และเมตริกซ์ไม่มีคุณสมบัติการสลับที่การคูณ ดังนั้น ในการหมุนพื้นฐานจึงมีผลต่อการหมุนผสมที่ได้ เพื่อเป็นการป้องกันความสับสนในการหาเมตริกซ์การหมุนผสมจึงควรใช้อัลกอริธึมต่อไปนี้

1. ให้เมตริกซ์การหมุน $R = I$ ได้จากเมื่อเริ่มต้นโครงพิกัดฉาก F และ M ทับกันสนิท

2. ถ้าโครงพิกัดเคลื่อนที่ M ถูกหมุนไป ϕ รอบยูนิตเวคเตอร์ที่ k^{th} ของโครงพิกัดอยู่นิ่ง ดังนั้นคูณเข้าทางซ้ายของ R ด้วย $R_k(\phi)$

3. ถ้าโครงพิกัดเคลื่อนที่ M ถูกหมุนไป ϕ รอบยูนิตเวคเตอร์ที่ k^{th} ของมันเอง ดังนั้นคูณเข้าทางขวาของ R ด้วย $R_k(\phi)$

4. ถ้ามีการหมุนพื้นฐานหลายครั้งให้กลับไปที่ขั้นตอน 2 มิฉะนั้นก็หยุดผลลัพธ์ที่ได้เป็นเมตริกซ์การหมุนผสม R ที่ใช้ถ่ายโอนโครงพิกัด M ไปยัง F

5. เมื่อมีการทำแบบ yaw, pitch และ roll สามารถหาเมตริกซ์ถ่ายโอนผสม yaw , pitch และ row ได้ โดยให้มุม yaw, pitch และ roll ใน R^3 ดังต่อไปนี้

ตามทฤษฎีบทการถ่ายโอน (yaw pitch roll Transformation) กำหนดให้ $YPR(\theta)$ เป็นเมตริกซ์หมุนผสม ที่ได้จากการหมุนโครงพิกัดเคลื่อนที่ $M = \{m^1 \quad m^2 \quad m^3\}$ ครั้งแรกโดยรอบแกน f^1 ด้วยมุม yaw θ_1 ต่อมาหมุนรอบแกน f^2 ด้วยมุม pitch θ_2 และสุดท้ายหมุนโดยรอบแกน f^3 ด้วยมุม roll θ_3 เมตริกซ์ผสม yaw pitch row $YPR(\theta)$ ซึ่งถ่ายพิกัด M ไปที่ F หาได้จาก

$$YPR(\theta) = \begin{bmatrix} c_2c_3 & s_1s_2c_3 - c_1c_3 & c_1s_2c_3 + c_1c_3 \\ c_2c_3 & s_1s_2s_3 + c_1c_3 & s_1s_2s_3 - s_1c_3 \\ -s_2 & s_1c_2 & c_1c_2 \end{bmatrix} \tag{2.6}$$

โดย $S_k = \sin\theta_k$ และ $C_k = \theta_k$

2.3.2 พิกัดโฮโมจีเนียส (Homogeneous Coordinates)

โดยปกติ เมตริกซ์ถ่ายโอนโฮโมจีเนียส (T) ขนาด 4 × 4 นี้ แบ่งได้เป็นเมตริกซ์ย่อย 4 ส่วน ดังนี้

$$T = \left[\begin{array}{c|c} \begin{matrix} rotation \\ R \end{matrix} & \begin{matrix} translation \\ P \end{matrix} \\ \hline \begin{matrix} h^T \\ perspective \end{matrix} & \begin{matrix} S \\ scale \end{matrix} \end{array}\right]$$

เมตริกซ์ย่อย 3×3 : R ในมุมซ้ายบนของ T เป็นเมตริกซ์การหมุน ซึ่งแสดงตำแหน่งการหมุนของโครงพิกัดเคลื่อนที่เทียบกับโครงพิกัดอยู่นิ่ง

คอลัมน์เวคเตอร์ 3×1 : P ในมุมขวาของ T เป็นเวคเตอร์การเลื่อน ซึ่งแสดงตำแหน่งของจุดกำเนิดของโครงพิกัดเคลื่อนที่เทียบกับโครงพิกัดอยู่นิ่ง

ค่า Scale : s ในมุมล่างขวาของ T เป็นสเกลแฟคเตอร์ที่ไม่เป็นศูนย์ ปกติให้เท่ากับ 1

เวคเตอร์แถว 1×3 : h^T ในมุมล่างซ้ายของ T เป็นเวคเตอร์สายตาปกติให้เท่ากับ 0

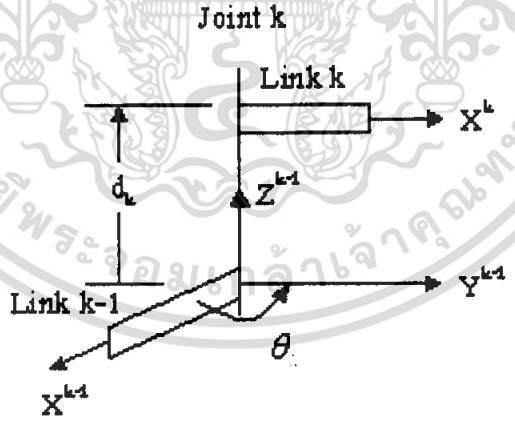
การถ่ายโอนโฮโมจีเนียสผสม (Composite Homogeneous Transformation) โดยปกติเมตริกซ์ถ่ายโอนผสมจะแสดงทั้งกาหมุน และการเลื่อนตำแหน่งของโครงเคลื่อนที่เทียบกับโครงพิกัดอยู่นิ่ง ลำดับของการหมุนและการเลื่อยในแต่ละครั้ง แสดงได้โดยผลคูณของเมตริกซ์ถ่ายโอนโฮโมจีเนียสพื้นฐาน เนื่องจากเมตริกซ์ไม่มีคุณสมบัติการสลับที่การคูณ ลำดับในการหมุนและการเลื่อนตำแหน่งจึงเป็นสิ่งสำคัญ ดังนั้นในการสร้างเมตริกซ์ จึงควรใช้ขั้นตอนของอัลกอริธึมการหมุนผสม

2.3.3 พิกัด Link (Link Coordinates)

ดังได้กล่าวแล้วว่า แขนกลประกอบด้วย Link หลายๆ อันมาเชื่อมต่อกันด้วยข้อต่อ โดยจุดประสงค์ในหัวข้อนี้ คือ การกำหนดโครงพิกัดลงบนแต่ละ Link จากนั้นจึงทำการหาสมการแขน (arm equation) ที่แสดงการเคลื่อนที่ของ Link ของแขนกล โดยขั้นแรกจะศึกษาพารามิเตอร์ที่ใช้ในการออกแบบตัวแขนกลก่อน

2.3.3.1 Kinematics Parameters

ตำแหน่งและการหมุนสัมพันธ์ของ Link ที่อยู่ติดกัน สามารถแสดงด้วย 2 joint parameters ดังแสดงในภาพที่ 2.9



รูปที่ 2.9 มุมข้อต่อ θ และระยะข้อต่อ d

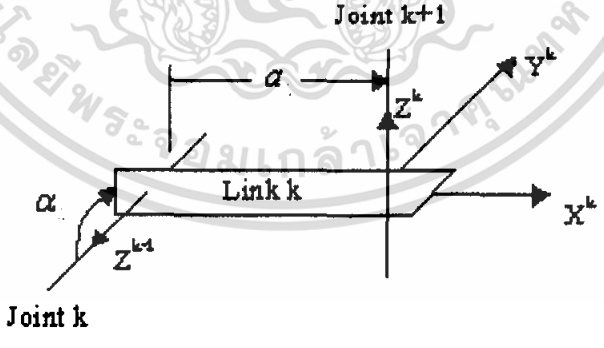
จากรูป ข้อต่อ k เชื่อม Link k-1 กับ Link k พารามิเตอร์ที่เกี่ยวกับข้อต่อ k ถูกนิยามเทียบกับแกน Z^{k-1} ซึ่งอยู่ในแนวเดียวกับแกนของ k joint parameter ตัวแรกเรียกว่า มุมข้อต่อ (joint angle) ; θ_k ซึ่งเป็นการหมุนรอบแกน Z^{k-1} เพื่อให้แกน X^{k-1} ขนานกับแกน X^k joint parameters ตัวที่สองเรียกว่า ระยะข้อต่อ (joint distance) ; d_k ซึ่งเป็นการเลื่อนตามแนว Z^{k-1} เพื่อให้แกน X^{k-1} ตัดกับ

แกน X^k ดังนั้น θ_k คือการหมุนรอบแกนข้อต่อ k และ d_k คือการเลื่อนไปตามแกนของข้อต่อ k โดยตามปกติ แต่ละข้อจะมี joint parameters ตัวหนึ่งมีค่าคงที่และมีอีกค่าเปลี่ยนแปลงได้ ซึ่งขึ้นอยู่กับชนิดของข้อต่อดังแสดงในตาราง 2.1

ตารางที่ 2.1 *Kinematic Parameters*

Arm parameter	Symbol	Rovolute Joint (R)	Prismatic Joint (P)
Joint angle	θ	Variable	Fixed
Joint distance	d	Fixed	Fixed
Link Length	a	Fixed	Variable
Link twist angle	α	Fixed	Fixed

ข้อต่อจะอยู่ระหว่าง Link 2 อันที่อยู่ติดกัน ในทางกลับกันจึงมี Link หนึ่งอันที่อยู่ระหว่างข้อต่อสองอัน ตำแหน่งและการหมุนสัมพันธ์ของแกนของข้อต่อทั้งสองอันแสดงได้ด้วย Link parameters จากภาพที่ 2.10 Link parameter ตัวแรกเรียกว่า ความยาว Link (Link length) ; a_k ซึ่งเป็นการเลื่อนตำแหน่งไปตามแนว X^k เพื่อให้แกน Z^{k-1} ตัดแกน Z^k และ Link parameter ตัวที่สองเรียกว่า มุมบิด Link (Link twist angle) α_k ซึ่งเป็นการหมุนรอบ X^k เพื่อให้แกน Z^{k-1} ขนานกับแกน Z^k

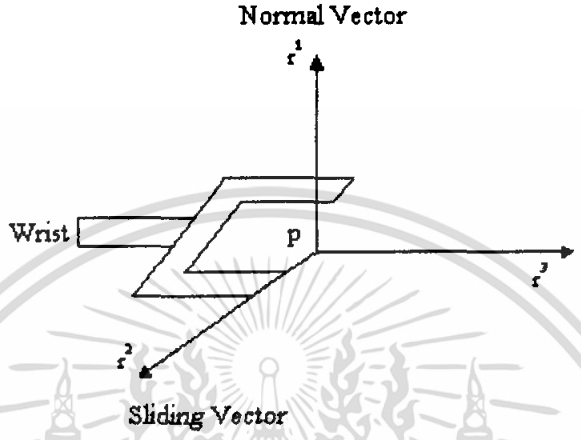


รูปที่ 2.10 ความยาวและมุมบิดของ Link

2.3.3.2 Normal, Sliding and Approach Vectors

ในการกำหนดโครงสร้างพิกัดลงบน Link ของแขนกล n แกน จะต้องให้ความสนใจเป็นพิเศษกับ Link สุดท้ายซึ่งเป็นเครื่องมือ หรือ End-effectors การหมุนของเครื่องมือสามารถแสดงในพิกัดฉากได้ด้วยเมตริกซ์การหมุน $R = \{r^1 \quad r^2 \quad r^3\}$ โดยที่ 3 คอลัมน์ของ R สอดคล้องกับ

Normal, Sliding and Approach Vectors ดังแสดงในรูปที่ 2.11 Approach Vectors; r^3 อยู่ในแนวเดียวกันกับแกนหมุนเครื่องมือและชี้ออกจากมือ Sliding Vectors; r^2 ตั้งฉากกับ Approach Vectors และอยู่ในแนวเดียวกับแกนเปิด-ปิดของเครื่องมือ Normal Vectors; r^1 ตั้งฉากกับระนาบของ Approach Vectors และ Sliding Vectors ที่ทำให้เกิดโครงพิกัดแกนตามกฎมือขวา



รูปที่ 2.11 Normal, Sliding and Approach Vectors ของ End-effectors

2.3.3.3 The Denavit-Hartenberg (D-H) Representation

เป็นวิธีกำหนดโครงพิกัดฉากตามกฎมือขวาลงบนแต่ละ Link ของแขนกล ซึ่งใช้สำหรับการถ่ายโอนพิกัดที่อยู่ติดกันด้วยเมตริกซ์ถ่ายโอนพิกัดโฮโมจีเนียส 4×4 โดยให้ L_k เป็นโครงพิกัดอยู่ที่ปลาย Link k เมื่อ

$$L_k = \{ X^k, Y^k, Z^k \} \; ; \; 0 \leq k \leq n \qquad (2.7)$$

อัลกอริธึม (D-H) Representation แสดงได้ดังนี้

0. กำหนดหมายเลขของข้อต่อจาก 1 ถึง n เริ่มจากฐาน ไปสิ้นสุดที่เครื่องมือตามการหมุน yaw pitch และ roll

1. กำหนดโครงสร้างพิกัดฉากตามกฎมือขวา L_k ลงบนฐานแขนกล โดยให้แกน Z_0 อยู่แนวเดียวกับแกนข้อต่อ 1 ให้ k = 1

2. วางแกน Z^k ให้อยู่แนวเดียวกับแกนข้อต่อ k+1

3. กำหนดจุดกำเนิดของ L_k ที่จุดตัดกันของแกน Z^k และ Z^{k-1} ถ้าไม่ตัดกันให้เซ็จุดตัดระหว่าง Z^k กับ Common normal ของ Z^k กับ Z^{k-1}

4. เลือกแกน X^k ให้ตั้งฉากกับแกน Z^k และ Z^{k-1} ถ้า Z^k และ Z^{k-1} ขนานกัน ให้แกน X^k ชี้ออกจาก Z^{k-1}

5. เลือก Y^k ให้ถูกตามกฎมือขวาของโครงพิกัดฉาก L_k

6. ให้ $k = k+1$ ถ้า $k < n$ กลับไปทำขั้นตอนที่ 2 มิฉะนั้นให้ทำตามขั้นตอนต่อไป

7. กำหนดจุดกำเนิดของให้อยู่ที่ปลายเครื่องมือ

โดยให้ Z^n อยู่แนวเดียวกับ Approach Vectors

Y^n อยู่แนวเดียวกับ Sliding Vectors

X^n อยู่แนวเดียวกับ Normal Vectors

8. กำหนดจุด b^k ที่จุดตัดกันของแกน X^k และ Z^{k-1} ถ้าไม่ตัดกันให้ใช้จุดตัดระหว่าง X^k กับ Common normal ของ X^k กับ Z^{k-1}

9. คำนวณ θ_k ซึ่งเป็นมุมจากการหมุนแกน X^{k-1} ไปยังแกน X^k โดยวัดรอบแกน Z^{k-1}

10. คำนวณ d_k ซึ่งเป็นระยะจากจุดกำเนิดของโครง L_{k-1} ไปยังจุด b^k วัดตามแนว Z^{k-1}

11. คำนวณ a_k ซึ่งเป็นระยะจากจุด b^k ไปยังจุดกำเนิดของโครง L_k วัดตามแนวแกน X^k

12. คำนวณ α_k ซึ่งเป็นมุมจากการหมุนแกน Z^{k-1} ไปยังแกน Z^k โดยวัดรอบแกน X^k

13. ให้ $k = k+1$ ถ้า $k \leq n$ กลับไปทำขั้นตอนที่ 8 มิฉะนั้นให้หยุด

2.3.4 สมการแขน (Arm Equation)

เมื่อกำหนดโครงพิกัด Link ได้แล้ว ก็ทำการย้ายโครงพิกัด k ไปยังโครงพิกัด k_{k-1} โดยใช้เมตริกซ์การถ่ายโอนพิกัดโฮโมจิเนียส เมื่อนำเมตริกซ์ที่ได้มาคูณกันทั้งหมด ก็จะได้เมตริกซ์ที่แสดงการย้ายพิกัดของเครื่องมือเทียบกับพิกัดฐาน โดยเมตริกซ์ที่ได้นี้เรียกว่า เมตริกซ์แขน (Arm Matrix)

2.3.4.1 การถ่ายโอนโครงพิกัด (Link-Coordinate Transformation)

การสร้างเมตริกซ์ถ่ายโอนโฮโมจิเนียสจากโครงพิกัด k มา k-1 แต่ละขั้นตอนจะเกี่ยวข้องกับ Kinematic Parameter หนึ่ง ในการพิจารณาจะทำการหมุนและเลื่อนตำแหน่ง โครงพิกัด k-1 จนทับกันสนิทกับโครงพิกัด k

ใช้ทฤษฎีบทการถ่ายโอนโครงพิกัด (Link-Coordinate Transformation) โดยกำหนดให้ $\{L_0, L_2, \ldots\ldots, L_n\}$ เป็นเซตของโครงสร้างที่ได้จากอัลกอลิธึม (D-H) Representation $[q]^k$ และ $[q]^{k-1}$ โดยจะได้ผลเป็นพิกัดโฮโมจิเนียสของจุด q เทียบกับโครง L_k และ L_{k-1} สำหรับ $1 \leq k \leq n$ จะได้ $[q]^{k-1} = T_{k-1}^{k}[q]^k$ โดย

$$T_{k-1}^{k} = \begin{bmatrix} c\theta_k & -c\alpha_k s\theta_k & s\alpha_k s\theta_k & a_k c\theta_k \\ s\theta_k & c\alpha_k c\theta_k & -s\alpha_k s\theta_k & a_k s\theta_k \\ 0 & s\alpha_k & c\alpha_k & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.8)$$

T_{k-1}^{k} คือการถ่ายโอนจากโครงพิกัด k ไปยังโครงพิกัด k-1

ในการแก้ปัญหา Direct Kinematics ต้องพิจารณาตำแหน่งและการหมุนของเครื่องมือเทียบกับโครงพิกัดที่ติดอยู่กับฐาน การถ่ายโอนจากพิกัดเครื่องมือไปยังพิกัดฐาน จะต้องเริ่มจากปลายเครื่องมือถ่ายโอนย้อนกลับไปทีละโครงจนถึงฐาน ถ้า T_{base}^{tool} แสดงการถ่ายโอนจากพิกัดปลายเครื่องมือ (Link n) ไปยังพิกัดฐาน (Link 0) ดังนั้น

$$T_{base}^{tool} = T_0^1(q_1)T_1^2(q_2).....T_{n-1}^n(q_n) \quad (2.9)$$

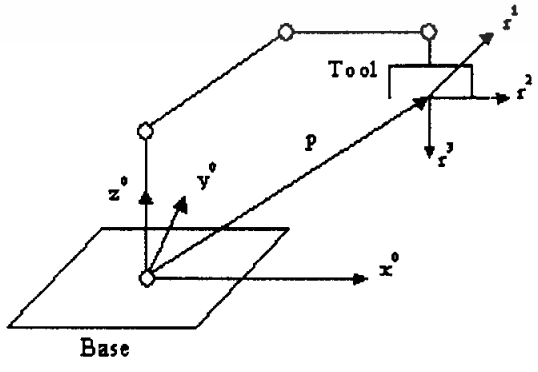
2.3.4.2 สมการแขน (Arm Equation)

จากสมการ (2.9) ได้เมตริกซ์ $T_{base}^{tool}(q)$ เมื่อแทนค่าลงไปจะได้สมการแขน (Arm Equation) ดังนี้

$$T_{base}^{tool} = \left[\begin{array}{c|c} R & P \\ \hline 000 & 1 \end{array}\right]$$

โดย R(q) เป็นเมตริกซ์ย่อย 3×3 แสดงการหมุนของเครื่อง คอลัมน์ทั้งสามของ R แสดงทิศทางของยูนิตเวคเตอร์ $\{r^1 \quad r^2 \quad r^3\}$ ของโครงเครื่องมือเทียบกับโครงฐาน

P(q) เป็นเมตริกซ์ย่อย 3×1 แสดงตำแหน่งของปลายเครื่องมือเป็นพิกัดของปลายเครื่องมือเมื่อเทียบกับโครงฐาน ดังแสดงในรูปที่ 2.12



รูปที่ 2.12 ตำแหน่งการหมุนของเครื่องมือในพิกัดฐาน

2.3.5 INVERSE KINEMATICS

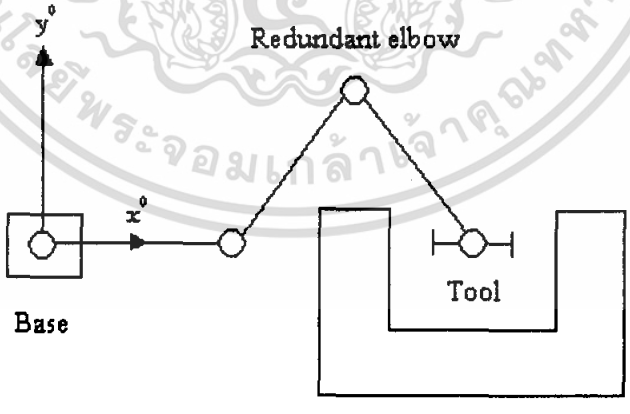
ในหัวข้อที่แล้วได้ทำการหาตำแหน่งและการหมุนของเครื่องมือแขนกล จากตัวแปรข้อต่อที่กำหนดให้ แต่ในหัวข้อนั้นจะพิจารณาย้อนกลับ คือการหาตัวแปรข้อต่อจากตำแหน่ง และการหมุนของเครื่องมือที่กำหนดให้ ปัญหา Inverse Kinematics เป็นปัญหาที่มีความสำคัญ และนำไปใช้งานในการแก้สมการแขน (Solve The Arm Equation) และยากกว่าปัญหา Direct Kinematics เนื่องจากไม่มีขั้นตอนที่แน่นอนและคำตอบที่ได้มักมีหลายคำตอบ

2.3.5.1 คุณสมบัติทั่วไปของคำตอบ (General Properties Of Solution)

ดังได้แสดงแล้วว่า การแก้ปัญหา Inverse Kinematics ไม่มีขั้นตอนที่แน่นอน ขึ้นอยู่กับชนิดและลักษณะของแขนกล แต่คำตอบที่ได้ก็ยังมีลักษณะบางประการที่เหมือนกัน

การมีอยู่ของคำตอบ (Existence Of Solution) พิจารณาในสภาวะที่คำตอบของ Inverse Kinematics มีอยู่จริง คือ ตำแหน่งปลายเครื่องมือ P อยู่ภายในของเขตการทำงาน และการหมุนเครื่องมือไม่เกินลิมิตของตัวแปรจากสมการแขน

ความเป็นหนึ่งของคำตอบ (Uniqueness Of Solution) เมื่อหาคำตอบของ Inverse Kinematics ได้แล้ว คำตอบที่ได้มักมีหลายคำตอบ เช่น แขนกล n แกนเมื่อ $n > 6$ คำตอบที่ได้มักมีมากมาย โดยเรียกหุ่นยนต์ที่มีมากกว่า 6 แกนว่า Kinematically Redundant Robots เพราะมี Degree of Freedom ที่เกินมานั้นช่วยเพิ่มความยืดหยุ่นให้กับแขนกล เช่น ใช้หลบหลีกสิ่งกีดขวาง ดังรูปที่ 2.13



รูปที่ 2.13 Redundant Robot หลบหลีกสิ่งกีดขวาง

2.3.5.2 Tool Configuration

ในการแก้ปัญหา Inverse Kinematics ต้องกำหนดสถานภาพที่ต้องการ โดยกำหนดให้สถานภาพถูกแทนโดย {p, R} ซึ่ง p แทนตำแหน่งของเครื่องมือโดยเทียบกับฐานและ R แทนทิศทางของเครื่องมือโดยเทียบกับฐาน

จากทฤษฎีบทเวคเตอร์การกำหนดสถานภาพ กำหนดให้ p และ R แทนตำแหน่งและทิศทางของโครงพิกัดเครื่องมือเทียบกับโครงพิกัดฐาน โดยที่ q_n แทนมุม roll ของเครื่องมือ ดังนั้นเวคเตอร์การกำหนดสถานภาพ : w ถูกกำหนดเป็น

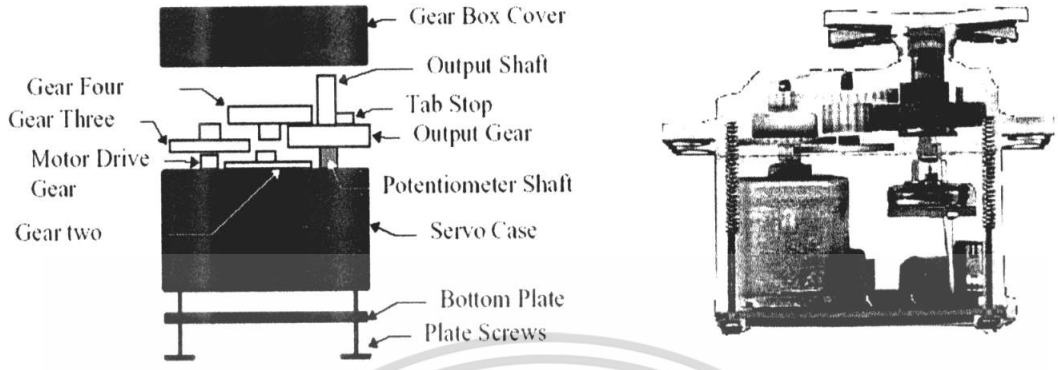
$$W = \begin{bmatrix} w^1 \\ w^2 \end{bmatrix} = \begin{bmatrix} p \\ [\exp(q_n / \pi)]r^3 \end{bmatrix} \tag{2.10}$$

จากทฤษฎีบทสามารถหามุม roll ได้จาก $q_n = p\ln(w_4^2 + w_5^2 + w_6^2)^{1/2}$

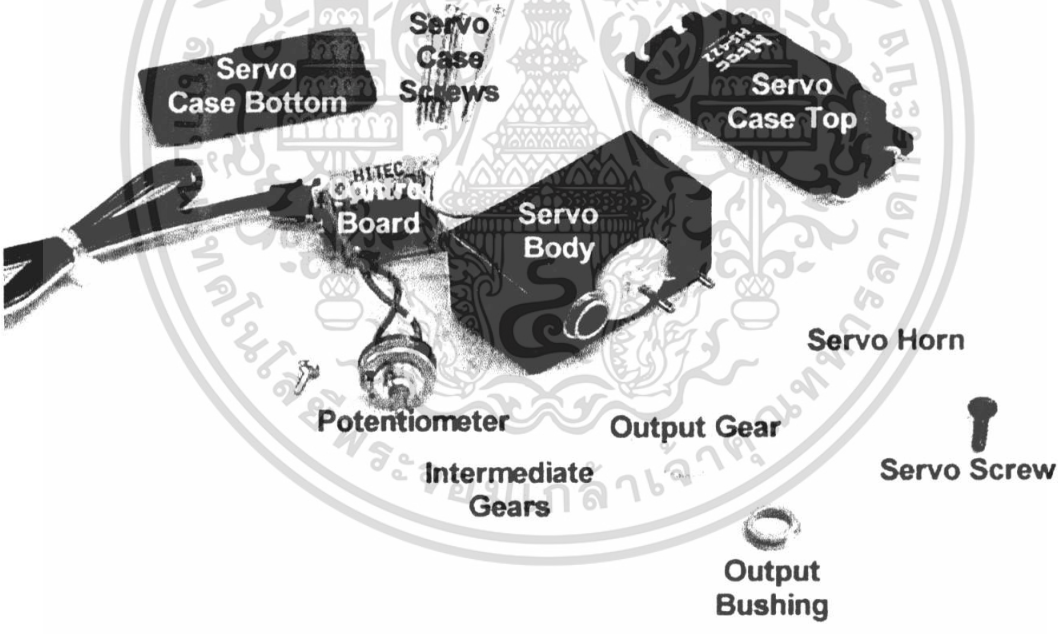
2.4 เซอร์โวมอเตอร์

Servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับชุดเกียร์และส่วนควบคุมต่างๆ ไว้ในโมดูลเดียวกันหรือภายในกล่องพาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานเพียง 3 เส้นเท่านั้น คือ VCC,GND และสายสัญญาณควบคุม (Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้ายหรือขวาได้จากสายสัญญาณเพียงเส้นเดียว โดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณพัลส์วิดมอด (PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลท์ ขึ้นอยู่กับคุณสมบัติของมอเตอร์แต่ละตัว ข้อดีของมอเตอร์ชนิดนี้ก็คือ จะมีขนาดเล็กน้ำหนักเบา, ให้แรงบิตสูง, กินพลังงานน้อยและสามารถควบคุมด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ (Driver) อื่นๆ เพราะมอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่งหรือทิศทางองศาที่ต้องการได้โดยอาศัยสัญญาณความกว้างพัลส์ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงในช่วงประมาณ 180° หรือครึ่งรอบเท่านั้น หรือบางรุ่นอาจหมุนได้ถึง 210° แต่จะไม่สามารถหมุนเป็นวงรอบได้ เนื่องจากโครงสร้างภายในจะประกอบด้วยตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้นเซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแค่ประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น เพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360°) นั้นก็สามารถทำได้ โดยจะต้องทำการปรับแต่ง (Modify) ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์

2.4.1 ส่วนประกอบต่างๆของเซอร์โวมอเตอร์



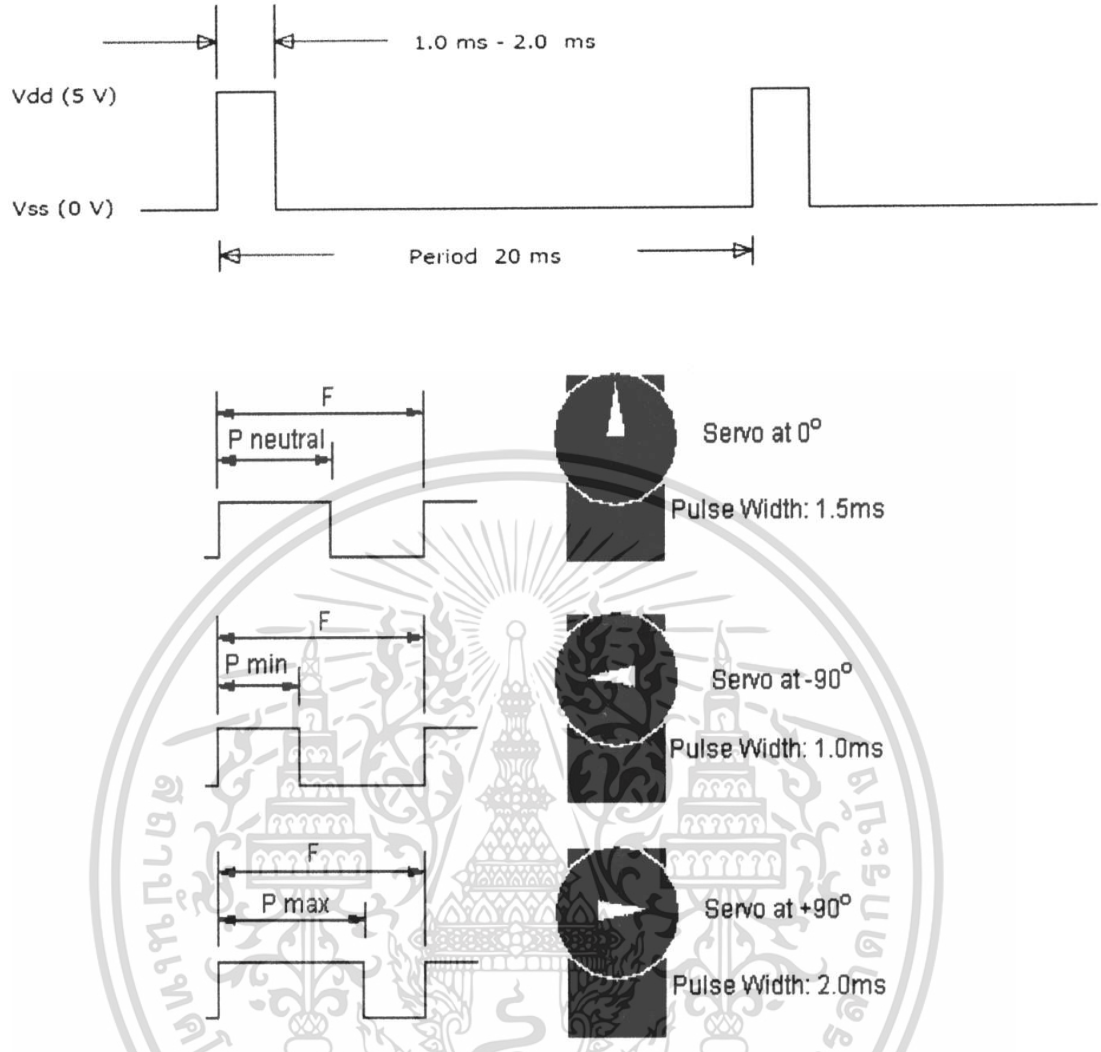
รูปที่ 2.14 ส่วนประกอบภายในของเซอร์โวมอเตอร์



รูปที่ 2.15 ส่วนประกอบภายนอกของเซอร์โวมอเตอร์

2.4.2 หลักการทำงานของเซอร์โวมอเตอร์

การควบคุมการทำงานของเซอร์โวมอเตอร์ทำได้โดยการป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้นๆ โดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังรูป คือ



รูปที่ 2.16 หลักการทำงานของเซอร์โวมอเตอร์

- สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือจุดกึ่งกลางของมอเตอร์
- สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม - 90 องศา หรือในทิศทางทวนเข็มนาฬิกา
- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม + 90 องศา หรือในทิศทางตามเข็มนาฬิกา

ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆ นั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆ โดยอ้างอิงจากจุด ทั้ง 3 จุดที่กล่าวมานี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม -45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 msเป็นต้น และสัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms(Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์

ไว้ โดยหลักการก็คือจะอาศัยกาเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR) เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุม สัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั้งคุาเวลาความกว้างพัลส์ของ วงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

2.5 dsPIC30F4011

dsPIC คือชื่อของไมโครคอนโทรลเลอร์ 16 บิตจาก Microchip Technology Inc. ผู้ผลิตไมโครคอนโทรลเลอร์ PIC ซึ่งรู้จักกันเป็นอย่างดีในแวดวงนักพัฒนาระบบไมโครคอนโทรลเลอร์ โดย Microchip Technology ได้กำหนดชื่ออย่างเป็นทางการสำหรับไมโครคอนโทรลเลอร์อนุกรมใหม่นี้ว่า Digital Signal Controller หรือ DSC นั่นหมายความว่า dsPIC เป็นไมโครคอนโทรลเลอร์ที่ได้รับการออกแบบมาเป็นพิเศษเพื่องานประมวลผลสัญญาณดิจิตอลสำหรับสร้างระบบควบคุมอัตโนมัติที่มีความสามารถสูง

2.5.1 คุณสมบัติเด่นโดยรวมของ dsPIC30F4011

2.5.1.1 คุณสมบัติของซีพียู

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูแบบ RISC
- ความเร็วในการทำงานสูงถึง 30 ล้านคำสั่งต่อวินาที
- มี 84 คำสั่งภาษาแอสเซมบลีมาตรฐาน รองรับรูปแบบการอ้างแอดเดรสได้อย่างอิสระ
- ชุดคำสั่งมีขนาด 24 บิต สามารถประมวลผลข้อมูลได้ 16 บิต
- มีหน่วยความจำโปรแกรมเป็นแบบแฟลช สามารถลบและเขียนใหม่ได้ไม่น้อยกว่า 100,000 ครั้ง สามารถป้องกันการอ่านได้ และสามารถโปรแกรมตัวเอง โดยใช้กระบวนการทางซอฟต์แวร์
- มีหน่วยความจำข้อมูลอีอีพรอมที่สามารถลบและเขียนใหม่ได้ไม่น้อยกว่า 1,000,000 ครั้ง
- มีอินเตอร์รัปต์เวกเตอร์จำนวนมาก จึงรองรับการตอบสนองสัญญาณอินเตอร์รัปต์ได้ดี
- มีวงจรตรวจจับแรงดันไฟเลี้ยงต่ำกว่ากำหนดแบบโปรแกรมได้
- มีเพาเวอร์-ออนรีเซต, เพาเวอร์-อัปไทเมอร์ และออสซิลเลเตอร์สตาร์ท-อัปไทเมอร์

- มีวอตช์ด็อกไทเมอร์แบบโปรแกรมได้
- มีวงจรตรวจสอบการทำงานของวงจรกำเนิดสัญญาณนาฬิกา
- รองรับการโปรแกรมในวงจรแบบอนุกรม (ICSP : In-Circuit Serial Programming)
- สามารถเลือกโหมดการใช้พลังงานได้

2.5.1.2 คุณสมบัติด้านการประมวลสัญญาณดิจิตอล

- มีแอกคิวมูเลเตอร์ขนาด 40 บิต 2 ตัว รองรับการประมวลผลทางคณิตศาสตร์ได้เป็นอย่างดี
- มีหน่วยประมวลผลด้านการคูณและหารเลข 17 บิตในรูปของฮาร์ดแวร์ จึงทำให้สามารถคูณและหารเลขได้อย่างรวดเร็ว
- ทำการคูณเลข 16 บิตได้ภายในสัญญาณนาฬิกาเพียง 1 ไซเกิล
- มีตัวเลื่อนข้อมูลบาร์เรล 40 สเตจ ช่วยให้การประมวลผลข้อมูลที่จำนวนบิตมากๆ สามารถทำได้อย่างรวดเร็ว
- มีวงจรเฟตช์ข้อมูลคู่ จึงทำให้สามารถประมวลผลข้อมูลได้อย่างรวดเร็ว

2.5.1.3 คุณสมบัติของโมดูลฟังก์ชันพิเศษ

- สามารถจ่ายกระแสออกทางขาพอร์ตได้ 25 mA ทั้งแบบกระแสซิงก์และซอร์ส
- ไทเมอร์/เคาน์เตอร์มีขนาด 16 บิต 5 ตัว ต่อใช้งานร่วมกันเป็นไทเมอร์ 32 บิตได้
- มีโมดูลตรวจจับสัญญาณดิจิตอลขนาด 16 บิต 4 ชุด
- มีโมดูลเปรียบเทียบข้อมูลและกำเนิดสัญญาณ PWM ความละเอียด 16 บิต 4 ชุด
    - ในการเปรียบเทียบข้อมูลสามารถเลือกการทำงานได้ 2 โหมด
- มีส่วนเชื่อมต่ออุปกรณ์อนุกรมแบบ SPI
- มีส่วนเชื่อมต่ออุปกรณ์ผ่านระบบบัส I^2C ทั้งแบบ 7 และ 10 บิต กำหนดเป็นมาสเตอร์หรือสเลฟได้
- มีโมดูลสื่อสารข้อมูลอนุกรม UART พร้อมบัฟเฟอร์แบบ FIFO
- มีโมดูลสร้างสัญญาณ PWM สำหรับควบคุมมอเตอร์ 6 ช่อง
    - เลือกรูปแบบเอาท์พุตได้ทั้งแบบคอมพลีเมนต์และแบบอิสระ
    - มีโหมดปรับตำแหน่งการหมุนทั้งแบบปรับขอบสัญญาณและแบบกึ่งกลาง
    - มีส่วนกำเนิดดิวตี้ไซเกิล 4 ชุด
    - กำหนดฐานเวลาได้ 4 โหมด

- สามารถเลือกขั้วของสัญญาณทางเอาท์พุตได้
- มีสัญญาณกระตุ้นเพื่อให้ทำงานสัมพันธ์กับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลภายในไมโครคอนโทรลเลอร์
- สามารถควบคุมสัญญาณเอาท์พุตได้

- มีโมดูลเชื่อมต่อตัวเข้ารหัสแบบควอดราเจอร์
  - มีอินพุต Phase A, Phase B และรับสัญญาณพัลส์เพื่อกำหนดตำแหน่ง
  - มีตัวนับตำแหน่งขนาด 16 บิต นับได้ทั้งขึ้นและลง
  - แสดงสถานะของทิศทางการนับได้
  - กำหนดโหมดของการวัดตำแหน่งได้ 2 โหมดคือ x2 และ x4
  - มีวงจรกรองสัญญาณรบกวนแบบดิจิตอลจากอินพุตแบบโปรแกรมได้
  - สำหรับกำหนดให้ทำงานเป็นไทเมอร์/เคาน์เตอร์ขนาด 16 บิตได้
  - กำเนิดสัญญาณอินเตอร์รัปต์จากตำแหน่งที่นับเกิน (rollover) หรือนับขาด (underflow)
- มีวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ความละเอียด 10 บิต 9 ช่อง
  - อัตราการสุ่มและแปลงสัญญาณ 500 กิโลแซมเปิลต่อวินาที
  - สามารถแปลงสัญญาณเมื่อไมโครคอนโทรลเลอร์ทำงานในโหมดสลีปและไอเดิลได้

2.5.2 สถาปัตยกรรมโดยสรุปของ dsPIC30F4011

2.5.2.1 หน่วยประมวลผลกลาง

หน่วยประมวลผลของ dsPIC30F4011 ใช้คำสั่งที่มีความยาว 1 เวิร์ด ขนาด 24 บิต โดยมีโปรแกรมเคาน์เตอร์ขนาด 23 บิต (จริงๆ แล้วโดยโครงสร้างมี 24 บิต แต่ไม่สนใจบิต MSB ซึ่งก็คือบิต 23 และบิต LSB หรือบิต 0 กำหนดเป็น "0" จึงทำให้สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 4 เมกะเวิร์ด) เพื่อแจ้งแอดเดรสของหน่วยความจำโปรแกรมที่เข้าไปประมวลผล dsPIC30F4011 มีความจุของหน่วยความจำโปรแกรม 48 กิโลไบต์ เมื่อคำสั่งมีความยาว 24 บิต จึงบรรจุคำสั่งได้จริง 16 กิโลเวิร์ด

รีจิสเตอร์หลักที่ใช้ในการทำงานคือ รีจิสเตอร์ W (Working register) สำหรับใน dsPIC จะแตกต่างจากไมโครคอนโทรลเลอร์ PIC อย่างมาก โดยรีจิสเตอร์ W ได้รับการจัดโครงสร้างเป็นอะเรย์ขนาด 16 บิต จึงทำให้สามารถรองรับทั้งข้อมูล, ค่าแอดเดรส หรือค่าของรีจิสเตอร์ใดๆ ที่ต้องนำมาประมวลผล โดยใน dsPIC มีรีจิสเตอร์ W ให้ใช้งานถึง 16 ตัว ส่วนใหญ่ใช้ในการประมวลผล

หลัก ส่วนอีกตัวหนึ่งคือรีจิสเตอร์ W15 จะใช้ทำงานร่วมกับตัวชี้สแต็กในการทำงานของโปรแกรมย่อยและบริการอินเตอร์รัปต์

ด้านการตอบสนองอินเตอร์รัปต์นั้น dsPIC30F4011 มีการจัดสรรพื้นที่เก็บค่าอินเตอร์รัปต์เวกเตอร์ไว้มาถึง 54 ตำแหน่ง และยังสามารถกำหนดระดับความสำคัญได้อีก 8 ระดับด้วย

2.5.2.2 หน่วยความจำ

dsPIC30F4011 มีหน่วยความจำโปรแกรม 16 กิโลเวิร์ด แอดเดรสอยู่ในช่วง 0x000100 ถึง 0x007FFE สามารถโปรแกรมหรือเขียนข้อมูลลงไปได้ 2 วิธีคือ

1. โดยใช้การโปรแกรมในวงจรแบบอนุกรมหรือ ICSP ผ่านทางขา PGD และ PGC (ขาที่ 17 และ 18) แล้วป้อนสัญญาณพัลส์แรงดันสูงสำหรับโปรแกรมผ่านเข้ามาทางขา MCLR

2. โปรแกรมตัวเองในขณะทำงานหรือ RTSP (Run Time Self-Programming)

ส่วนหน่วยความจำข้อมูลแรมนั้น dsPIC30F4011 ได้จัดสรรเป็น 2 ส่วนคือ หน่วยความจำข้อมูลแรม X และ Y แต่ละส่วนมีขนาด 16 บิต ความจุ 256 ไบต์ รวมเป็น 512 ไบต์ โดยในแต่ละส่วนจะมีตัวกำหนดแอดเดรสแยกออกจากกัน เรียกว่า AGU (Address Generation Unit)

ในขณะที่หน่วยความจำข้อมูลอีอีพรอม dsPIC30F4011 จัดสรรไว้ที่แอดเดรส 0x7FFC00 ถึง 0x7FFFFE มีความจุ 1 กิโลไบต์

2.5.2.3 ส่วนประมวลผลสัญญาณดิจิตอล (DSP Engine)

นับเป็นส่วนประกอบที่สำคัญอย่างยิ่งของ dsPIC เนื่องจาก dsPIC ได้รับการออกแบบมาให้ทำงานในด้านการประมวลผลสัญญาณดิจิตอลเป็นหลัก ดังนั้นจึงต้องมีการเพิ่มความสามารถในหน่วยคำนวณทางคณิตศาสตร์และลอจิกอย่างมาก โดยในส่วนประมวลผลสัญญาณดิจิตอลมีหน่วยจัดการคูณเลขขนาด 17 x 17 บิตความเร็วสูง, หน่วยประมวลผลทางคณิตศาสตร์และลอจิกหรือ ALU ขนาด 40 บิต, แอกคิวมูเลเตอร์ขนาด 40 บิต อีก 2 ตัว และตัวเลื่อนข้อมูล 2 ทิศทางแบบบาเรล (barrel shifter) ขนาด 40 บิต จึงทำให้สามารถจัดการข้อมูลขนาด 16 บิตได้เสร็จสิ้นภายในสัญญาณนาฬิกาเพียงไซเกิลเดียว

2.5.2.4 โมดูลฟังก์ชันพิเศษ

dsPIC30F4011 ได้รวมเอาโมดูลสำหรับทำงานเฉพาะทางไว้อย่างมากมาย ไม่ว่าจะเป็นโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล ความละเอียด 10 บิต, โมดูลเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI, โมดูลสื่อสารข้อมูลระบบบัส I^2C, โมดูลสื่อสารข้อมูลผ่านพอร์ตอนุกรมหรือ UART, ไทเมอร์ขนาด 16 บิตถึง 5 ตัว และที่เป็นพิเศษอีก 2 โมดูลคือ โมดูลสร้างสัญญาณ PWM เพื่อการ

ควบคุมมอเตอร์และโมดูลเข้ารหัสแบบควอดราเจอร์ โดยสามารถใช้งานร่วมกันเพื่อสร้างระบบควบคุมมอเตอร์แบบปิดประสิทธิภาพสูง

2.5.2.5 พอร์ตอินพุตเอาต์พุต

dsPIC30F4011 มีพอร์ตให้ใช้งานมากถึง 5 พอร์ต รวม 30 ขา ดังนี้

พอร์ต B	มี 9 ขาคือ RB0-RB8 โดยทุกขาสามารถกำหนดให้เป็นพอร์ตอินพุตหรือเอาต์พุตได้ และยังสามารถขับกระแสทั้งแบบซิงก์และซอร์สได้สูงถึง 25 mA
พอร์ต C	มี 3 ขาคือ RC13-RC15
พอร์ต D	มี 4 ขาคือ RD0-RD3
พอร์ต E	มี 7 ขาคือ RE0-RE5 และ RE8
พอร์ต F	มี 7 ขาคือ RF0 - RF6

2.5.3 คุณสมบัติของไทเมอร์

ใน dsPIC30F4011 มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิต ให้ใช้งานรวม 5 ตัว คือ ไทเมอร์1 (T1), ไทเมอร์ 2 (T2), ไทเมอร์ 3 (T3), ไทเมอร์4 (T4) และ ไทเมอร์ 5 (T5)

- คุณสมบัติของไทเมอร์ 1
  - รีจิสเตอร์ตัวนับความละเอียด 16 บิต
  - ทำงานได้ทั้งแบบซิงโครนัสและอะซิงโครนัสเคาน์เตอร์
  - ทำงานร่วมกับขาอินพุตประจำตัวของไทเมอร์ได้
  - มีปรีสเกลเลอร์สำหรับหารความถี่การนับ
  - สามารถกำหนดอินเตอร์รัปต์จากการนับหรือจากการตรวจพบสัญญาณขอบขาลงที่ขาอินพุตของไทเมอร์

- คุณสมบัติของไทเมอร์ 2, 3, 4 และ 5
  - ไทเมอร์ 2 , 3, 4 และ 5 เมื่อทำงานแยกอิสระต่อกัน มีคุณสมบัติคล้ายกับไทเมอร์ 1
  - เมื่อนำไทเมอร์ 2 และ 3 มาทำงานร่วมกัน รีจิสเตอร์ตัวนับมีความละเอียดเพิ่มขึ้น 32 บิต
  - เมื่อนำไทเมอร์ 4 และ 5 มาทำงานร่วมกัน รีจิสเตอร์ตัวนับมีความละเอียดเพิ่มขึ้น 32 บิต
  - ทำงานร่วมกับขาอินพุตประจำตัวไทเมอร์ได้ (ขา TxCKI)
  - มีปรีสเกลเลอร์สำหรับหารความถี่การนับ
  - สามารถกำหนดอินเตอร์รัปจากการนับหรือจากการตรวจพบสัญญาณขอบขาลงที่ขา

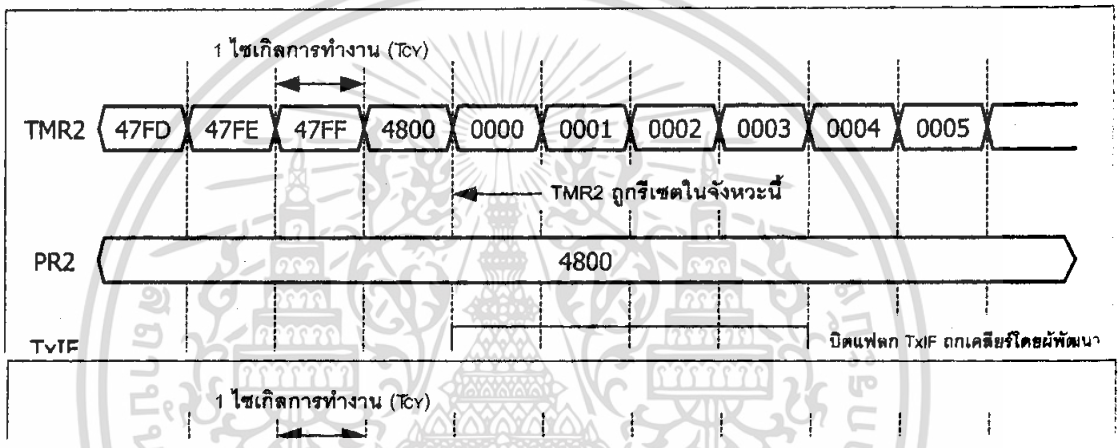
อินพุตของไทเมอร์

- สามารถกำเนิดสัญญาณกระตุ้นการทำงานไปยังโมดูล ADC ได้

2.5.4 การอินเตอร์รัปต์ในไทเมอร์

การอินเตอร์รัปเนื่องจากการทำงานของไทเมอร์มีบิตที่เกี่ยวข้องโดยตรง 5 บิตคือ

1. TxIE บิตเอ็นเอเบิลการอินเตอร์รัปต์เนื่องจากการทำงานของไทเมอร์
2. TxIP2 ถึง TxIP0 บิตเลือกระดับความสำคัญของการอินเตอร์รัปต์
3. TxIF บิตแฟลกแสดงสถานการณ์เกิดอินเตอร์รัปต์



รูปที่ 2.17 ไดอะแกรมเวลาแสดงการเกิดอินเตอร์รัปต์เมื่อค่าของรีจิสเตอร์ TMRx เท่ากับรีจิสเตอร์คาบเวลา PRx โดยสังเกตได้จากการเซตบิตแฟลก TxIF

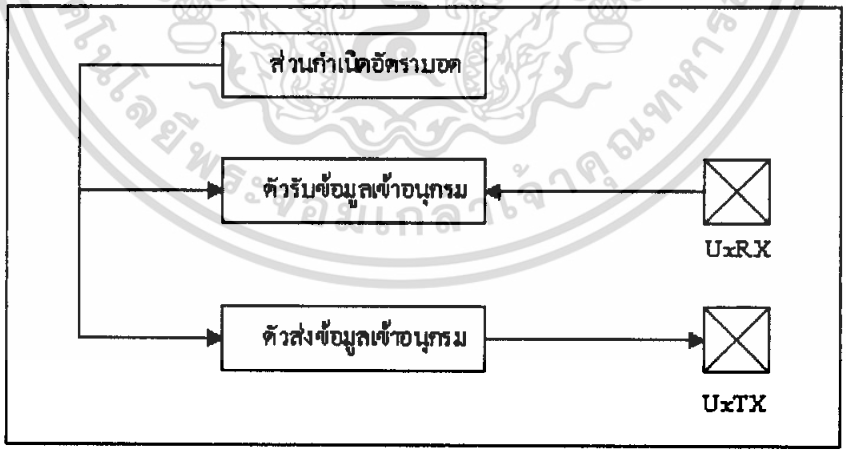
ในไทเมอร์ 16 บิตสามารถกำเนิดสัญญาณอินเตอร์รัปต์เมื่อ

1. ค่าของรีจิสเตอร์ TMRx เท่ากับรีจิสเตอร์คาบเวลา PRx
2. มีการเปลี่ยนจากลอจิกสูงไปต่ำหรือเกิดขอบขาลงของสัญญาณเกตในโหมดเกตไทเมอร์บิต TxIF จะถูกเซตเมื่อเงื่อนไขดังต่อไปนี้เป็นจริง
3. ค่าของรีจิสเตอร์ TMRx เท่ากับรีจิสเตอร์คาบเวลา PRx และไทเมอร์ต้องไม่ทำงานในโหมดเกตไทเมอร์
4. มีการเปลี่ยนลอจิกจากสูงไปต่ำหรือเกิดขอบขาลงของสัญญาณเกตในโหมดเกตไทเมอร์ การเคลียร์บิต TxIF ต้องกระทำด้วยกระบวนการทางซอฟแวร์เท่านั้น

2.5.5 การสื่อสารข้อมูลอนุกรมโดยใช้โมดูล UART

2.5.5.1 คุณสมบัติโดยสรุปของโมดูล UART ใน dsPIC30F4011

- สื่อสารข้อมูลแบบสองทิศทาง (ฟูลดูเพล็กซ์ : full-duplex) ในแบบ 8 และ 9 บิต
- เลือกการสื่อสารข้อมูลแบบตรวจสอบบิตพาริตี้คู่ (Even) หรือคี่ (Odd) และไม่ตรวจสอบบิตพาริตี้ (None) สำหรับรูปแบบสื่อสารข้อมูลในแบบ 8 บิต มี บิตหยุด (Stop bit) 1 หรือ 2 บิต
- มีส่วนกำเนิดอัตราบอด (Baud Rate Gemerator) ขนาด 16 บิต สำหรับกำหนดจังหวะและอัตราเร็วในการสื่อสารข้อมูลอนุกรมแยกอิสระเพื่อลดภาระการทำงานของโมดูลไทเมอร์
- กำเนิดอัตราบอดได้ตั้งแต่ 38 บิตต่อวินาที (bps) ถึง 1.875 เมกะบิตต่อวินาที (Mbps)
- บัฟเฟอร์ข้อมูลขาส่ง (TX) และขารับ (RX) ขนาด 4 เวิร์ด แยกส่วนกัน
- มีบิตแฟลกแจ้งข้อผิดพลาดในกรณีต่างๆ ของการสื่อสาร สามารถตรวจจับความผิดพลาดในการสื่อสารข้อมูลอนุกรม ได้แก่
  - ความผิดพลาดทางพาริตี้ (Parity Error : PE)
  - รับข้อมูลไม่ทัน (Buffer Overrun Error : OE)
  - เฟรมข้อมูลผิดพลาด (Framing Error : FE)
- สนับสนุนความสามารถในการอินเตอร์รัปต์แอสเดรส (ข้อมูลบิต 9 เป็น "1")
- อินเตอร์รัปต์เวกเตอร์แยกตำแหน่งกันระหว่างการส่งและรับข้อมูล (RX)
- สามารถทำงานในโหมด Loopback



รูปที่ 2.18 ไดอะแกรมส่วนประกอบหลักของโมดูล UART ในไมโครคอนโทรเลอร์ dsPIC

2.5.5.2 รีจิสเตอร์ที่ใช้ในโมดูล UART

มีทั้งสิ้น 5 ตัวคือ

1. UxMODE ใช้กำหนดโหมดการทำงานของโมดูล UART โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด

2. UxSTA ใช้แสดงสถานะการทำงานของโมดูล UART โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด

3. UxRXREG ใช้เก็บข้อมูลที่รับเข้ามาทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด

4. UxTXREG ใช้เก็บข้อมูลสำหรับส่งออกทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด

5. UxBRG ใช้เก็บค่าสำหรับกำหนดอัตราบอด โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด

2.5.5.3 การกำหนดให้โมดูล UART ทำงาน

โมดูล UART ในไมโครคอนโทรลเลอร์ dsPIC ใช้รูปแบบข้อมูลในมาตรฐาน NRZ นั่นคือ มีบิตเริ่มต้น 1 บิต, บิตข้อมูล 8 บิต หรือ 9 บิต และบิตปิดท้าย 1 หรือ 2 บิต ส่วนบิตพาริตี้สามารถเลือกได้แบบคู่ (even), คี่ (odd) หรือไม่มี (none) โดยปกติแล้วจะเลือกใช้รูปแบบ 8N1 คือ มีบิตเริ่มต้น 1บิต, บิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สำหรับ dsPIC30F4011 สามารถกำหนดได้ที่บิต PDSEL1, PDSEL0 และ STSEL ซึ่งเป็นบิต 2, 1 และ 0 ตามลำดับในรีจิสเตอร์ U1MODE ส่วนการกำหนดอัตราบอดนั้นกระทำผ่านรีจิสเตอร์ UXBRG ขนาด 16 บิต

การรับส่งข้อมูลในโมดูล UART นั้นจะรับและส่งข้อมูลบิต LSB หรือบิตนัยสำคัญต่ำสุดก่อน โดยตัวรับและส่งข้อมูลในโมดูล UART ของ dsPIC จะทำงานเป็นอิสระแยกจากกัน โดยใช้อัตราบอดและรูปแบบข้อมูลที่เหมือนกัน จึงสามารถรับส่งข้อมูลได้ 2 ทิศทางพร้อมกันตลอดเวลา

การเอ็นเอเบิลโมดูล UART ใน dsPIC30F4011 ให้ทำงานทำได้โดยการเซตบิต UAERTEN ซึ่งเป็นบิต 15 ในรีจิสเตอร์ U1MODE และเซตบิต UTXEN ซึ่งเป็นบิต 10 ในรีจิสเตอร์ U .STA ทันทีที่เอ็นเอเบิล ขา U1TX และ U1RX จะถูกกำหนดให้ทำงานเป็นขาพอร์ตเอาต์พุตและอินพุตตามลำดับ โดยไม่สนใจการกำหนดทิศทางที่เกิดขึ้นก่อนหน้านี้

การดิสเอเบิลโมดูล UART ใน dsPIC30F4011 ทำได้โดยการเคลียร์บิต UARTEN ซึ่งเป็นบิต 15 ในรีจิสเตอร์ U1MODE โดยปกติโมดูล UART จะถูกดิสเอเบิลหลังจากที่เกิดการรีเซต ซึ่งเป็นการกำหนดสถานะในเบื้องต้น จากนั้นหากต้องการให้ทำงานจะต้องมีการเอ็นเอเบิลภายหลัง เมื่อโมดูล UART ถูกดิสเอเบิล ขา U1TX และ U1RX จะถูกปลดออกจากโมดูล UART สามารถนำไปใช้งานเป็นพอร์ตอินพุตเอาต์พุตทั่วไปได้ทันที

นอกจากนั้นในภาวะที่โมดูล UART ถูกดิสเอเบิล ค่าในบัฟเฟอร์ทั้งหมดจะถูกเคลียร์ เช่นเดียวกับบิตแฟลกแจ้งสถานะความผิดพลาดทั้งหมดจะถูกเคลียร์ด้วย ส่วนตัวนับอัตราบอดจะอยู่ในภาวะรีเซต หลังจากที่ดิสเอเบิลโมดูล UART แล้วกลับมาเอ็นเอเบิลใหม่ การทำงานทั้งหมดจะเริ่มต้นใหม่ทั้งหมด

2.5.5.4 ความสัมพันธ์ระหว่างสัญญาณนาฬิการะบบกับอัตราบอด

ความสัมพันธ์ระหว่างความถี่สัญญาณนาฬิกาของระบบกับอัตราบอดในการสื่อสารข้อมูลอนุกรมนั้นสามารถแสดงในรูปของสมการดังนี้

$$\text{อัตราบอด} = \frac{F_{CY}}{16 \times (BRG + 1)} \tag{2.11}$$

และ

$$F_{CY} = \frac{F_{OSC}}{4} \tag{2.12}$$

โดยที่ BRG คือค่าข้อมูลของจิสเตอร์ UxBRG (x เท่ากับ 1 หรือ 2 ขึ้นอยู่กับใช้โมดูล UART ชุดใด)

F_{OSC} คือความถี่สัญญาณนาฬิกาหลัก

2.5.5.5 ขาพอร์ตเสริมของโมดูล UART (Alternate UART I/O Pins)

ในไมโครคอนโทรลเลอร์ dsPIC บางเบอร์ได้บรรจุขาพอร์ตสำหรับรับและส่งข้อมูลกับโมดูล UART เสริมเพิ่มเติมจากที่มีอยู่เดิมเป็นขา UxATX และ UxARX ทั้งนี้เพื่ออำนวยความสะดวกและเพิ่มช่องทางในการใช้งานขาพอร์ตของ dsPIC ที่อาจพบข้อจำกัดด้านการออกแบบแผ่นวงจรพิมพ์หรือด้านการเขียนโปรแกรม

ใน dsPIC30F4011 ได้บรรจุความสามารถนี้ไว้ด้วย นั่นคือขาพอร์ต U1ATX และ U1ARX โดยการเลือกใช้งานต้องทำการเซตบิต ALTIO ซึ่งเป็นบิต 10 ในรีจิสเตอร์ U1MODE เมื่อเลือกใช้ขาพอร์ตเสริมนี้แล้ว ขาพอร์ตหลักคือ U1TX และ U1RX จะกลายเป็นขาพอร์ตอินพุตเอาต์พุตสำหรับใช้ในงานอื่นแทน

2.6 บอร์ด JX-dsPIC40

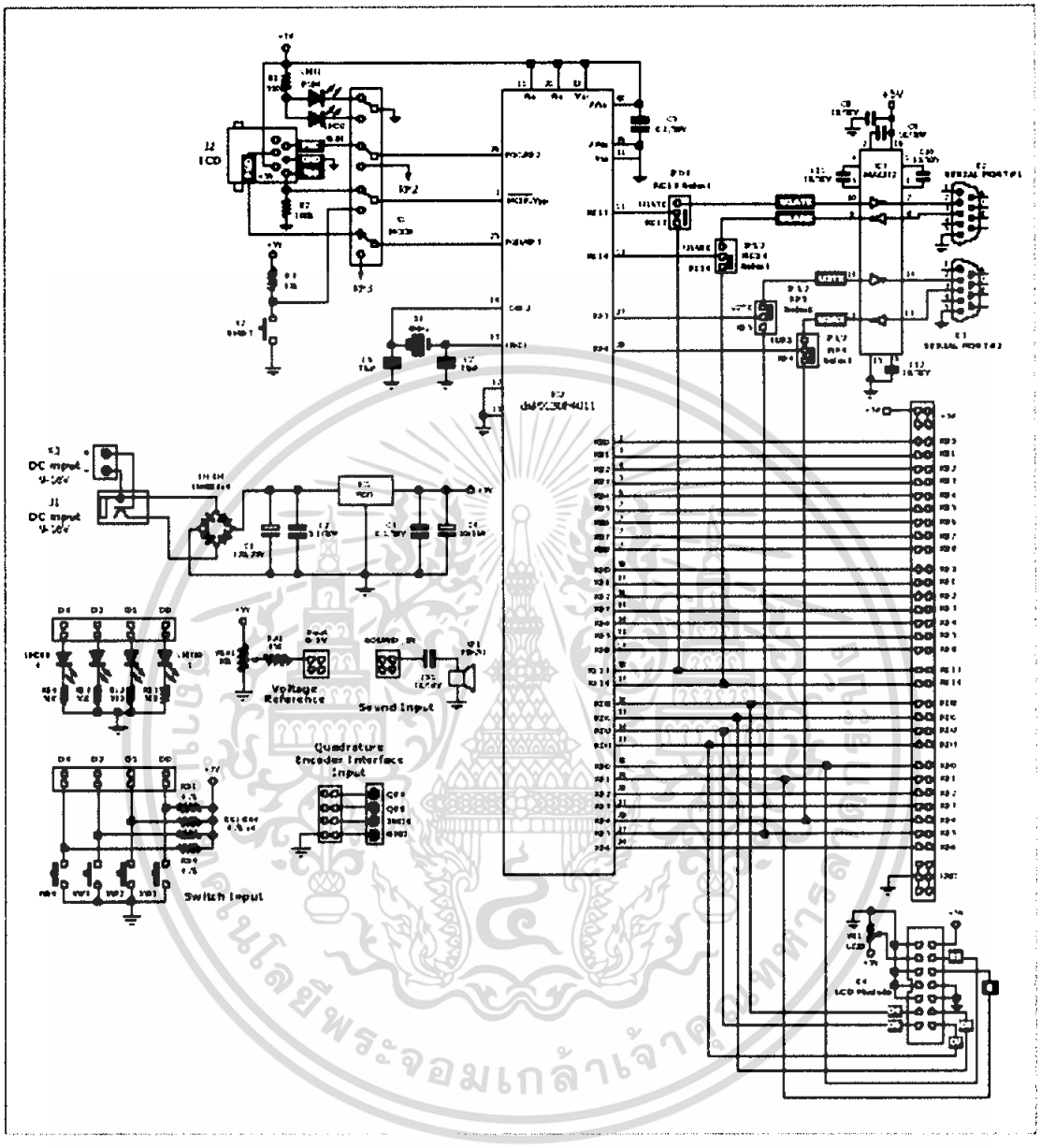
2.6.1 คุณสมบัติทางเทคนิค

- ใช้ได้กับไมโครคอนโทรลเลอร์ dsPIC ตัวถัง DIP 40 ขา บนบอร์ดใช้เบอร์ dsPIC30F4011สัญญาณนาฬิกา
- ทำการโปรแกรมและดีบักได้ด้วย ICDX-30 หรือ ICD2 ของ Microchip
- เลือกโหมดการโปรแกรมและรันด้วยสวิตช์กดเพียงตัวเดียว พร้อมไฟแสดงสถานะ
- มีจุดต่อขอพอร์ตสำหรับทดลองและใช้งาน 5 พอร์ตคือ

RB0-RB8	9 ขา	RC13-RC14	2 ขา	RD0-RD3	4 ขา
RE0-RE5,RE8	7 ขา	RF0-RF6	7 ขา		

- มีภาคจ่ายไฟ +5V 800 mA บนบอร์ด
- มีสวิตช์ RESET สำหรับรีเซตการทำงานของไมโครคอนโทรลเลอร์
- จุดต่อโมดูล LCD โดยต่อขา D4-D7 กับพอร์ต RD0-RD3, ขา RS ต่อกับพอร์ต RF0 และขา E ต่อกับพอร์ต RF1
- วงจรเชื่อมต่อคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS-232 จำนวน 2 ชุด
  UART1 ต่อกับพอร์ต RC13 และ RC14 เลือกด้วยจั๊มเปอร์
  UART2 ต่อกับพอร์ต RF4 และ RF5 เลือกด้วยจั๊มเปอร์
- LED แสดงผลพร้อมตัวต้านทานจำกัดกระแส ทำงานที่ลอจิก “1” 4 ช่อง พร้อมตัวต้านทางพูลอัป
- เทอร์มินอลบล็อก 4 ช่องสำหรับต่อกับโมดูล QEI (Quadrature Encoder Input)
- วงจรจ่ายแรงดัน 0-5V สำหรับทดลอง A/D ผ่านตัวต้านทานปรับค่าได้
- พื้นที่สำหรับสร้างวงจรหรือ Proto area 3x2.5 นิ้ว จุดบัดกรี 170 จุด สามารถติดตั้งแผงต่อวงจรขนาดกลางได้
- ใช้ไฟเลี้ยงจากภายนอกผ่านแจ็กอะแดปเตอร์หรือเทอร์มินอลบล็อก +9V ถึง +16V

2.6.2 วงจรบอร์ด JX-dsPIC40



รูปที่ 2.19 วงจรของบอร์ด JX-dsPIC40

2.7 การสื่อสารแบบอนุกรม

ถึงแม้ว่าการสื่อสารแบบอนุกรมในเครื่องคอมพิวเตอร์นั้นจะมีความเร็วในการสื่อสารช้ากว่าแบบขนาน ทั้งนี้เพราะการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลครั้งละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลได้ครั้งละหลายๆบิตพร้อมกัน ส่งผลให้การสื่อสารข้อมูลแบบอนุกรมนั้นมีความเร็วต่ำกว่าแบบขนาน

แต่ว่าการส่งข้อมูลแบบอนุกรมนั้นมีข้อดีที่เหนือกว่าการส่งข้อมูลแบบขนาน คือ การส่งข้อมูลได้ในระยะทางที่ไกลกว่าแบบขนาน อีกทั้งสายสัญญาณที่ใช้ยังมีน้อยกว่าการส่งข้อมูลแบบขนานอีกด้วยการสื่อสารแบบอนุกรมสามารถแบ่งออกเป็น 3 รูปแบบดังนี้

Simplex สามารถส่งข้อมูลได้อย่างเดียว เป็นการสื่อสารแบบทางเดียว

Half-Duplex สามารถส่งข้อมูลไปยังปลายทางและสามารถรับข้อมูลจากปลายทางได้ แต่ไม่สามารถทำการส่งและรับข้อมูลได้ในเวลาเดียวกัน

Full-Duplex สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน

นอกจากนี้ยังสามารถแบ่งประเภทของการสื่อสารแบบอนุกรมตามลักษณะสัญญาณในการส่งได้อีก 2 แบบคือ

การสื่อสารแบบซิงโครนัส (Synchronous) สำหรับการสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการรับการส่งสัญญาณ เช่น สายคีย์บอร์ด โดยจะมีสายสัญญาณหนึ่งเป็นสัญญาณนาฬิกา ส่วนอีกสายหนึ่งเป็นสายข้อมูล(และมักจะมีสายกราวน์ด้วย)

การสื่อสารแบบนี้เหมาะสำหรับการสื่อสารการทำงานระยะทางใกล้ ข้อมูลที่ส่งมีไม่มากนักเพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายสัญญาณหลายเส้นทำให้สิ้นเปลืองมาก

การสื่อสารแบบอะซิงโครนัส(Asynchronous)สำหรับการสื่อสารแบบอะซิงโครนัสนั้นจะใช้สายข้อมูลเพียงสายเดียว แต่รูปแบบที่ใช้จะใช้รูปแบบการส่งข้อมูล (Bit Pattern) เป็นตัวกำหนดว่าส่วนไหนเป็นตัวส่วนเริ่มต้นข้อมูล ส่วนไหนเป็นตัวส่วนข้อมูล ส่วนไหนเป็นส่วนตรวจสอบข้อมูล โดยต้องกำหนดสัญญาณนาฬิกาให้เท่าๆกันทั้งภาคส่งและภาครับ ซึ่งจะมีอุปกรณ์พิเศษที่ชื่อว่า UART หรือ Universal Asynchronous Receiver/Transmitter คอยควบคุมการรับและส่งข้อมูล

แต่สำหรับมาตรฐานของการรับส่งข้อมูลในปัจจุบันที่ได้รับความนิยมอย่างมากทั้งในอดีตและปัจจุบัน นั้นคือ มาตรฐาน RS-232

2.7.1 องค์ประกอบการรับส่งข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรมที่นิยมใช้กับคอมพิวเตอร์เป็นการสื่อสารข้อมูลแบบอะซิงโครนัส คือต้องใช้สายสัญญาณเส้นเดียวในการทำหน้าที่ทั้งส่งส่วนที่เป็นข้อมูลและส่วนที่ใช้ควบคุมการส่งข้อมูล ดังนั้นข้อมูลที่เราอ่านได้แต่ละบิตจากการส่งแบบอนุกรม จึงต้องถูกแยกแยะว่าใช้สำหรับวัตถุประสงค์ใด โดยเราแบ่งแยกออกเป็นได้ 4 ส่วนคือ

1. Start bit ขนาด 1 บิต
2. Data bit ขนาด 7-8 บิต
3. Parity bit ขนาด 1 บิต
4. Stop bit ขนาด 1-2 บิต

แต่ละตัวอักษรที่ถูกส่งออกไปเป็นกลุ่มจะประกอบไปด้วยบิตเริ่มต้น บิตข้อมูล บิตพาริตี้ (จะมีหรือไม่มีก็ได้) และบิตจบ โดยเราพอจะสรุปหน้าที่ของแต่ละส่วนได้ดังนี้

- Start Bit หรือบิตเริ่มต้น จะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง
- Data Character หรือบิตข้อมูล การส่งบิตข้อมูลจะส่งเป็นกลุ่มๆ โดยทั่วไปจะส่งเป็น 7 หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง Ascii Word
- Parity Bit หรือบิตพาริตี้ ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง เราจะใส่บิตพาริตี้เข้าไป แต่ทั้งตัวรับและตัวส่งจะต้องรู้กันว่าใช้พาริตี้แบบไหนในการส่งข้อมูล ซึ่งหลักการในการกำหนดบิตพาริตี้มีหลายแบบดังนี้
    - พาริตี้คู่ (Even Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุกๆ บิตของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคู่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 0
    - พาริตี้คี่ (Odd Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุกๆ บิตของข้อมูลแล้วจะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคี่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 1
    - ไม่มีพาริตี้ (None) ถ้าทั้งบิตพาริตี้เป็น None ทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบบิตพาริตี้
- Stop Bit หรือบิตจบ เป็นบิตที่มาปิดท้ายข้อมูล

2.7.2 อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม

การที่อุปกรณ์ 2 อย่างจะติดต่อสื่อสารกันได้นั้น จะต้องทำงานด้วยอัตราเร็วเท่ากัน ซึ่งอัตราเร็วในการสื่อสารแบบอะซิงโครนัสคือ ค่าบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที ซึ่งค่าอัตราเร็วในการสื่อสารแบบอนุกรมสำหรับมาตรฐาน RS-232 ได้แก่ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มอาจจะสามารถกำหนดค่าบอดเรตได้สูงถึง 115200 บิตต่อวินาที เนื่องจากบอดเรตคือจำนวนบิตของข้อมูลที่สามารถถ่ายทอดได้ภายใน 1 วินาที ยกตัวอย่างข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิตและบิตปิดท้าย 1 บิต ความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาริตี้ความเร็วในการรับส่งข้อมูลจะเหลือเป็น 872 ไบต์ต่อวินาที

2.7.3 มาตรฐานพอร์ตอนุกรมแบบ RS-232

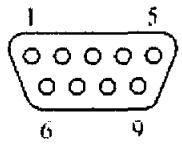
มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดหนึ่งซึ่งอยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association: EIA) ได้วางมาตรฐานที่มีชื่อเรียกว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์แบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3V ถึง -12V แสดงว่ามีข้อมูล (Mark) และ +3V ถึง +12V แสดงว่าเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์ที่เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) ไว้ว่า อุปกรณ์ CTE จะต้องเป็นอุปกรณ์ที่การประมวลผลในตัว เช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

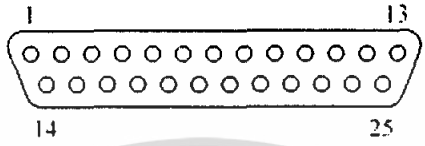
ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

2.7.4 คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 2.20



(ก) คอนเน็กเตอร์อนุกรม 9 ขาหรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)

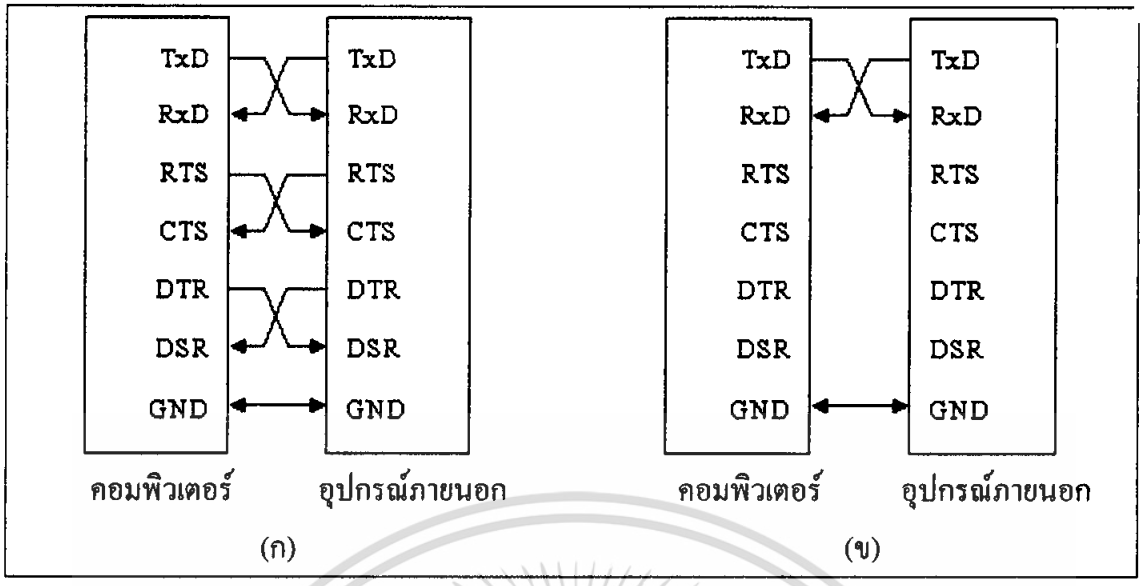


(ข) คอนเน็กเตอร์อนุกรม 25 ขาหรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์)

คอนเน็กเตอร์	คอนเน็กเตอร์	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Detect: DCD	อินพุต
2	3	Received Data: RxD	อินพุต
3	2	Transmitted Data: TxD	เอาต์พุต
4	20	Data Terminal Ready: DTR	เอาต์พุต
5	7	Signal Ground: GND	-
6	6	Data Set Ready: DSR	อินพุต
7	4	Request To Send: RTS	เอาต์พุต
8	5	Clear To Send: CTS	อินพุต
9	22	Ring Indicator: RI	อินพุต

รูปที่ 2.20 การจัดขาของคอนเน็กเตอร์พอร์ตอนุกรมตามมาตรฐาน RS-232 ทั้งแบบ DB-9 และ DB-25

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก แสดงดังในภาพที่ 2.21 ถูกศรในรูปแสดงถึงทิศทางของข้อมูล ในรูปที่ 2.21 (ก) เป็นการเชื่อมต่อแบบ Null Modem หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็กเต็มรูปแบบ ส่วนในรูปที่ 2.21 (ข) เป็นการเชื่อมต่อแบบ Null Modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูลและเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่การทำงานภายในและขาพอร์ตอนุกรม RS-232 มีดังนี้



รูปที่ 2.21 การต่ออุปกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่างๆ

(ก) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ Null modem

(ข) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ RS-232 โดยใช้สายสัญญาณเพียง 3 เส้น

- Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกทีฟเมื่อมีการส่งสัญญาณพาหะจากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกตินี้จะไม่ได้ถูกใช้งานมากนัก
- Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้ เก็บไว้ในรีจิสเตอร์บัฟเฟอร์
- Transmitted Data : TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลออกไป
- Data Treminal Ready : DTR เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์ เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบ Null Modem ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกัน และต้องต่อกับขา DCD ด้วย ในกรณีโปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาหะ
- Signal Ground : GND ขากราวด์ของระบบ
- Data Set Ready :DSR ขานั้นจะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR จะเป็นขาสำหรับรับข้อมูลจากภายนอก ซึ่งถูกส่งมาจากขา DTR

- Request To Send : RTS เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อ Null Modem 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา
- Clear To Send : CTS ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่
- Ring Indicator :RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในกาสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

2.7.5 UART

UART ย่อจากคำว่า Universal Asynchronous Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

หน้าที่หลักของ UART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรม แบบอะซิงโครนัส แล้วส่งออกไปและทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล (ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น

ภายใน UART มีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ (Programmable Buadrate Generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้มีขนาด 16 บิต ดังนั้นจึงสามารถกำหนดตัวหารอยู่ในช่วง 1 - 56,535 UART ซึ่งสามารถรับส่งข้อมูลได้ทั้งแบบ Half – Duplex และแบบ Full – Duplex

ในเครื่องคอมพิวเตอร์ทั่วไปมี UART ที่ใช้งานกันอยู่ 2 เบอร์คือ 8250 ซึ่งเป็น UART มาตรฐานที่มีใช้กันมายาวนาน UART เบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตำแหน่งเดียวกัน ทำให้การรับและส่งข้อมูลถูกจำกัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่ UART เบอร์นี้ถือว่าเป็นต้นแบบของ UART ที่ใช้ในคอมพิวเตอร์ โดยคอมพิวเตอร์ทุกๆ รุ่น จะต้องสนับสนุนการทำงานตามรูปแบบของ UART เบอร์นี้

UART อีกเบอร์หนึ่งคือ 16450 มีความสามารถรับส่งข้อมูลได้ที่ความเร็ว 115,200 บิตต่อวินาที และเพิ่มรีจิสเตอร์สำหรับพักข้อมูลสำหรับ UART นอกจากนั้นยังเพิ่มส่วนของชิฟรีจิสเตอร์แบบ FIFO (First In First Out) ขนาด 16 ไบต์เข้าไปทำให้สามารถสนับสนุนความเร็วในการรับส่ง

ข้อมูลที่ 26 กิโลบิตต่อวินาทีได้ โดยคอมพิวเตอร์ในปัจจุบันใช้ UART เบอร์นี้ หรือใหม่กว่า เช่น เบอร์ TL16C750 ซึ่งมีรีจิสเตอร์แบบ FIFO ขนาด 64 ไบต์ ทำงานได้ที่ระดับแรงดัน +5V และ +3V มีโหมดประหยัดพลังงาน สามารถรับส่งข้อมูลได้ที่ความเร็ว 1 เมกะบิตต่อวินาที เมื่อใช้สัญญาณนาฬิกา 16 MHz.

อย่างไรก็ตาม ความเร็วในการส่งข้อมูลที่มากมายของ UART เบอร์ใหม่ๆ ก็ไม่ได้ช่วยให้การรับส่งข้อมูลของคอมพิวเตอร์เร็วขึ้น เนื่องจากว่าคอมพิวเตอร์ยังใช้ความถี่ของสัญญาณนาฬิกาในการแปลงข้อมูลเพียง 1.8432 MHz. เท่านั้น

2.7.6 รีจิสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์โดยทั่วไปสามารถต่อพอร์ตอนุกรม RS-232 สูงสุดได้ 4 พอร์ต ซึ่งจะมีชื่อเรียกเป็น COM1, COM2, COM3 และ COM4 ซึ่งพอร์ตอนุกรมแต่ละตัวต่างก็ใช้งาน UART ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอกเช่นเดียวกัน

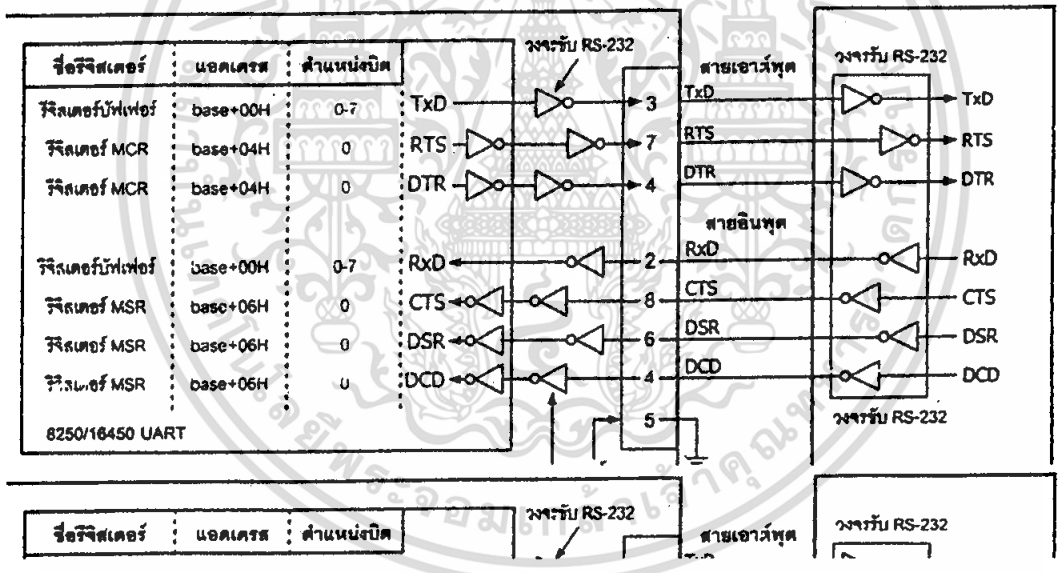
การทำงานภายในของพอร์ตอนุกรม ประกอบไปด้วยรีจิสเตอร์ขนาด 8 บิต 8 ตัวที่ใช้งานร่วมกับ UART แอดเดรสของรีจิสเตอร์ภายในพอร์ตอนุกรมสามารถคำนวณได้จากค่ารีจิสเตอร์พื้นฐานของพอร์ตอนุกรม ยกตัวอย่าง พอร์ตอนุกรม COM1 มีแอดเดรสอยู่ที่ 3F8H ตำแหน่งของรีจิสเตอร์ต่างๆ จะเป็นตำแหน่งที่บวกเข้าไปกับค่า 3F8H โดยรีจิสเตอร์ที่ใช้งานกับพอร์ตอนุกรมมีดังนี้

- 00H รีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลรับเข้ามาหรือเตรียมข้อมูลก่อนที่จะส่งออกไป
- 01H รีจิสเตอร์อีนาเบิลการอินเตอร์รัปต์ที่ใช้ในการเซตโหมดการอินเตอร์รัปต์ของพอร์ตอนุกรม
- 02H รีจิสเตอร์แสดงโหมดการอินเตอร์รัปต์ ใช้เพื่อตรวจสอบโหมดการอินเตอร์รัปต์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น
- 03H รีจิสเตอร์กำหนดรูปแบบของข้อมูล
- 04H รีจิสเตอร์ควบคุมโมเด็ม ใช้ตรวจสอบบิตสำหรับติดต่อกับโมเด็ม เช่น RTS หรือ DTR
- 05H รีจิสเตอร์แสดงสถานะการรับและส่งข้อมูลแบบอนุกรม
- 06H รีจิสเตอร์แสดงสถานะโมเด็ม ซึ่งจะแสดงสถานะของขา DCD, RI, DSR, และ CTS
- 07H รีจิสเตอร์สำหรับการเก็บข้อมูลชั่วคราว

ลักษณะสัญญาณอินพุตและเอาต์พุตของพอร์ต RS-232

สัญญาณเอาต์พุตที่ใช้ควบคุม (RTS และ DTR) และสัญญาณแสดงสถานอินพุต (CTS, DSR และ DCD) ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและรับจะไม่ถูกกลับสถานะ UART จะให้ระดับสัญญาณเอาต์พุตออกมาเป็นแบบทีทีแอลเท่านั้น ดังนั้นเมื่อสัญญาณถูกส่งออกมาจาก UART จึงต้องส่งเข้าสู่วงจรขับเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์ สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์ สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรที่ใช้ทั้งภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้น จะถูกกลับสถานะ

2.7.7 แอดเดรสของพอร์ตอนุกรม



รูปที่ 2.22 ไดอะแกรมแสดงโครงสร้างทางฮาร์ดแวร์ของพอร์ตอนุกรม

แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแหน่ง ดังนี้คือ

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไปออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหมด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรม ไบออสจะนำ

แอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับพอร์ตอนุกรม COM1 จะเก็บไว้ที่แอดเดรส 0000:0100H และ 0000:0401H ส่วนตำแหน่งอื่นๆ มีรายละเอียดดังนี้

COM2 = 0000:0402H - 0000:0403H

COM3 = 0000:0404H - 0000:0405H

COM4 = 0000:0406H - 0000:0407H

นอกจากนี้ ที่หน่วยความจำแอดเดรส 0000:0411H ยังใช้สำหรับแสดงจำนวนพอร์ตอนุกรมที่มีใช้อยู่ในคอมพิวเตอร์อีกด้วย โดยมีรายละเอียดดังแสดงในตารางที่ 2.3

ตารางที่ 2.3 ข้อมูลในแอดเดรส 0000:0411H ที่ใช้แจ้งจำนวนพอร์ตอนุกรม

บิต 3	บิต 2	บิต 1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์ตอนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 2 พอร์ต
0	1	1	มีพอร์ตอนุกรม 3 พอร์ต
1	0	0	มีพอร์ตอนุกรม 4 พอร์ต

2.7.8 ระดับแรงดันที่ใช้งานสำหรับพอร์ตอนุกรม

มาตรฐานการสื่อสารข้อมูลของพอร์ตอนุกรม ได้ระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่า ที่ลอจิก "0" จะมีระดับสัญญาณ +3V ถึง +15V ส่วนลอจิก "1" จะมีระดับสัญญาณ -3V ถึง -15V ระดับสัญญาณนี้ทำให้ไม่สามารถที่จะนำขาเอาต์พุตใดๆ ต่อเข้ากับลอจิกเกตเพื่อใช้งานได้โดยตรง จะต้องผ่านวงจรเพื่อเปลี่ยนระดับแรงดันเสียก่อน โดยปกติจะใช้ไอซีจำพวก RS-232 transceiver ที่นิยมมากคือ MAX232 หรือ ICL232 ไปใช้ในกลุ่มนี้ จะทำหน้าที่แปลงระดับแรงดันของ RS-232 ให้อยู่ในระดับ ทีทีแอล โดยลอจิก "0" ซึ่งเดิมมีระดับสัญญาณ +3V ถึง +15V จะถูกแปลงเป็น 0V ส่วนลอจิก "1" ซึ่งมีระดับสัญญาณ -3V ถึง -5V จะแปลงเป็น +5V ทั้งนี้เพื่อให้สามารถเชื่อมต่ออุปกรณ์ดิจิตอลอื่นที่ใช้ระดับแรงดัน ทีทีแอล ได้

2.8 การเขียนโปรแกรมเพื่อใช้งานพอร์ตอนุกรม

2.8.1 การกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรม

ก่อนการใช้งานพอร์ตอนุกรมนั้นจะต้องมีการกำหนดค่าเริ่มต้นให้กับตัวมันก่อน ซึ่งก็คือการกำหนดจำนวนบิตข้อมูลที่ต้องการจะส่ง, จำนวนปิดท้าย, ชนิดของพาริตี้ที่ใช้และบอดเรต

การกำหนดสามารถทำได้หลายวิธี วิธีแรกเป็นการกำหนดจากดอสพร้อมพ์ โดยใช้คำสั่ง MODE ซึ่งมีวิธีในการใช้งานดังนี้

Mode Comm: baud=b, parity=p, data=d, stop=s, retry=r

หรือ Mode Comm: b, p, d, s, r

ตัวอย่าง *MODE COM1 : 9600, n, 8, 1* จะเป็นการกำหนดให้พอร์ตอนุกรม COM1 มีบอดเรตเท่ากับ 9600 บิตต่อวินาที ไม่มีการตรวจสอบพาริตี้รับส่งข้อมูลแบบ 8 บิตและมีบิตปิดท้าย 1 บิต

วิธีที่ 2 เป็นการกำหนดโดยใช้อินเตอร์รัปต์ของดอส ตำแหน่งที่ 14H ซึ่งการใช้งานจะต้องกำหนดค่าต่างๆ ลงในรีจิสเตอร์ด้วย โดยจะต้องกำหนดให้รีจิสเตอร์ AH มีค่าเท่ากับ 0 รีจิสเตอร์ DX เก็บค่าของพอร์ตอนุกรมที่ต้องการกำหนดค่าเริ่มต้น โดย

DX=0 จะกำหนดให้พอร์ตอนุกรม COM1

DX=1 จะกำหนดให้พอร์ตอนุกรม COM2

DX=2 จะกำหนดให้พอร์ตอนุกรม COM3

DX=3 จะกำหนดให้พอร์ตอนุกรม COM4

รีจิสเตอร์ AL ซึ่งมีขนาด 8 บิตใช้เก็บค่าเริ่มต้นต่างๆ มีรายละเอียดดังนี้

BD2	BD1	BD0	PAR1	PAR0	STOP	DA1	DA0

BD2, BD1, BD0 ใช้สำหรับกำหนดค่าบอดเรต

"111" บอดเรตเท่ากับ 9,600 บิตต่อวินาที

"110" บอดเรตเท่ากับ 4,800 บิตต่อวินาที

"101" บอดเรตเท่ากับ 2,400 บิตต่อวินาที

"100" บอดเรตเท่ากับ 1,200 บิตต่อวินาที

"010" บอดเรตเท่ากับ 300 บิตต่อวินาที

"001" บอดเรตเท่ากับ 150 บิตต่อวินาที

"000" บอดเรตเท่ากับ 110 บิตต่อวินาที

PAR1, PAR0 ใช้กำหนดค่าพาริตี้โดย

"00" หรือ "10" ไม่มีการตรวจสอบพาริตี้

"01" พาริตี้คี่

"11" พาริตี้คู่

STOP ใช้กำหนดจำนวนของบิตปิดท้าย

"1" มีบิตปิดท้ายเท่ากับ 2 บิต

"0" มีบิตปิดท้ายเท่ากับ 1 บิต

DA1, DA0 ใช้กำหนดความยาวของข้อมูลโดย

"10" ความยาวข้อมูลเท่ากับ 7 บิต

"11" ความยาวข้อมูลเท่ากับ 8 บิต

2.8.2 การรับส่งข้อมูลแบบอนุกรม

มีหลากหลายวิธีในการรับและส่งข้อมูลแบบอนุกรมผ่านพอร์ต RS-232 เช่น ใช้คำสั่งพิมพ์ออกทางเครื่องพิมพ์ เรียกอินเทอร์รัปต์ของไบออสหรือของดอส การเขียนหรืออ่านไปยังแอดเดรสของพอร์ตโดยตรง วิธีสุดท้ายเป็นวิธีที่มีความยืดหยุ่นในการใช้งานที่สุด ยกตัวอย่าง ถ้าต้องการส่งข้อมูลไปยังพอร์ตอนุกรม COM1 สามารถเขียนข้อมูลโดยตรงไปที่รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (แอดเดรส 3F8H) โดยใช้คำสั่งภาษา QBASIC ง่ายๆ ดังนี้

OUT&H3F8, X

ค่า X ในที่นี้หมายถึงข้อมูลที่ต้องการส่งมีขนาด 8 บิต

สำหรับการอ่านค่าข้อมูลจากพอร์ตอนุกรม จะเป็นการอ่านข้อมูลมาจากรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (แอดเดรส 3F8H เช่นเดียวกัน) ซึ่งสามารถเขียนโปรแกรมง่ายๆ ได้ดังนี้

Y=INP[&3F8]=X

แต่เมื่อมีการใช้คำสั่งนี้ในขณะที่โปรแกรมทำงานผ่านระบบปฏิบัติการวินโดวส์ จะไม่สามารถใช้งานได้ เนื่องจากระบบปฏิบัติการวินโดวส์นั้น ได้เข้าฝังตัวพอร์ตอนุกรมเข้าเป็นส่วนหนึ่งของระบบปฏิบัติการ ดังนั้นการใช้งานต้องเรียกผ่านเครื่องมือที่ติดต่อผ่านระบบปฏิบัติการ เช่น MSCOMM32.OCX ของโปรแกรม Visual Basic

2.8.3 คอนโทรล MSCOMM

สำหรับการใช้งาน Visual Basic ตั้งแต่เวอร์ชัน 2 เป็นต้นมาใน Visual Basic จะมี Custom Control สำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์มาให้ โดยใน Visual Basic เวอร์ชัน 3 จะชื่อว่า MSCOMM.VBX ส่วนเวอร์ชัน 4 ใช้ชื่อว่า MSCOMM16.OCX สำหรับการทำงานกับระบบปฏิบัติการ 16 บิต และ MSCOMM32.OCX สำหรับระบบปฏิบัติการ 32 บิต สำหรับ Visual Basic เวอร์ชันต่อๆ มาจะมีเพียง MSCOMM32.OCX เท่านั้น

MSCOMM จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือการสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (event-driven communication) เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมาก สำหรับการตอบสนองแบบทันทีทันใด ยกตัวอย่างเช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรม หรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier Detect (DCD) หรือขา Request To Send (RTS) เหตุการณ์ Oncomm ของ MSCOMM จะสามารถตรวจจับได้ทันที ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้น ด้วยการดูค่าที่เปลี่ยนแปลงภายใน

คุณสมบัติ Commevent หลังจากให้โปรแกรมทำงานในฟังก์ชันต่างๆ ไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSCOMM 1 ตัว สามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งาน ต้องการติดต่อกับพอร์ตอนุกรมในแต่ละพอร์ตแอดเดรสในพอร์ตอนุกรมและแอดเดรสของการอินเทอร์รัปต์ สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel ถึงแม้ว่าคอนโทรล MSCOMM จะมีคุณสมบัติมากมายแต่จะสามารถทำความเข้าใจได้ไม่ยากดังนี้

2.8.3.1 CommPort

ใช้ในการกำหนดอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่ (COM1, COM2, COM3, COM4)

รูปแบบการใช้งาน

Object.Setting[=Value]

ค่า Value เป็นค่าของพอร์ตอนุกรมชนิดของข้อมูลเป็น Integer ค่า Value สามารถกำหนดได้ในช่วง 1-16 (ค่าเริ่มต้นกำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้ว ทำการเปิดพอร์ตอนุกรมโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSCOMM จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมา ซึ่งหมายถึงอุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คำสั่ง OpenPort

2.8.3.2 Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอดเรต, พาริตี้, จำนวนบิตข้อมูล, จำนวนของบิตปิดท้าย

รูปแบบการใช้งาน

Object.setting[=value]

ค่า value นี้ มีชนิดข้อมูลแบบ string มีรูปแบบเป็น “BBBB, P, D, S” โดย BBBB เป็นค่าอัตราบอดเรต, P เป็นค่าพาริตี้, D เป็นจำนวนของข้อมูลบิตข้อมูลและ S เป็นจำนวนของบิตปิดท้าย ปกติแล้วค่านี้ถูกกำหนดไว้เป็น “9600, N, 8, 1”

ค่าบอดเรตมาตรฐานที่ใช้กับ MSCOMM มีดังนี้

110 บิตต่อวินาที

300 บิตต่อวินาที

600 บิตต่อวินาที

1,200 บิตต่อวินาที

2,400 บิตต่อวินาที

9,600 บิตต่อวินาที

14,400 บิตต่อวินาที

19,200 บิตต่อวินาที

28,800 บิตต่อวินาที

38,400 บิตต่อวินาที

56,000 บิตต่อวินาที

128,000 บิตต่อวินาที

256,000 บิตต่อวินาที

สำหรับค่ามาตรฐานในการกำหนดค่าพาริตี้มีดังนี้

สัญลักษณ์	รายละเอียด
E	พาริตี้คู่
M	ลอจิก “1”
N	ไม่ใช้
O	พาริตี้คี่
S	ลอจิก “0”

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 ค่า คือ 4, 5, 6, 7 และ 8 (เป็นค่าปกติ)

ค่าที่ระบุจำนวนบิตปิดท้ายมี 3 ค่า คือ 1 (เป็นค่าปกติ), 1.5 และ 2

ตัวอย่างการใช้งานคำสั่ง Settings โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนข้อมูล 8 บิตและบิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

MSCOMM1.setting=”9600,N,8,1”

สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายในเครื่องหมายคำพูด ก็เนื่องจากค่าที่กำหนด อยู่ในรูปตัวแปร String

2.8.3.3 PortOpen

เพื่อใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อทำการเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

Object.portopen[=value]

ค่า value นี้ มีชนิดของข้อมูลเป็นแบบบูลีน คือมี True กับ False โดย True จะหมายถึงพอร์ตที่ได้ถูกเปิดออกแล้วและ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ Commport นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ได้ถูกต้องหรือไม่

มิฉะนั้นแล้ว MSCOMM จะแสดงข้อผิดพลาด error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ถ้าคุณสมบัติ DTR Enable หรือ RTS Enable ถูกกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบัติของ DTR Enable หรือ RTS Enable จะถูกเซตเป็น False หลังจากปิดพอร์ต แต่ถ้าเซตเป็น False หลังจากปิดโปรแกรมแล้ว ค่าที่กำหนดไว้จะเป็นค่าเดิม

ตัวอย่างการใช้คำสั่งเปิดพอร์ตเพื่อติดต่อสื่อสารกับพอร์ตอนุกรม COM1 และมีบอดเรต 9,600 บิตต่อวินาที ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิตและบิตปิดท้าย 1 บิตมีดังนี้

```
Mscomm.setting = "9600,n,8,1"
Mscomm1.comport = 1
Mscomm1.portopen = true
```

2.8.3.4 Input

อ่านค่าและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

```
Object.input
```

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดให้คุณสมบัติ Input ทำการอ่านค่าข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้า InputLen ถูกกำหนดเป็น comInputModeText คุณสมบัติ Input จะส่งค่าข้อมูลกลับมาในรูปแบบของข้อความ ชนิดข้อมูลเป็นแบบ Variant ถ้า comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

ตัวอย่างโปรแกรมแสดงให้เห็นถึงวิธีการรับส่งข้อมูลจากบัฟเฟอร์ทั้งหมด

```
Private Sub Command_click()
Dim Instring as String
        Mscomm1.inputlen=0
        If Mscomm1.inbuffercount then
        Instring=Mscomm1.input
        End if
End sub
```

2.8.3.5 InBufferCount

ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.inbuffercount[=value]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ของภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไปสำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับ ทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

2.8.3.6 InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งานคำสั่ง

Object.inbuffersize[=value]

คำสั่ง Inbuffersize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ โดยค่าเริ่มต้นถูกกำหนดไว้ที่ 1,024 ไบต์

การกำหนดค่าบัฟเฟอร์ภาครับขนาดใหญ่ จะทำให้หน่วยความจำที่เหลือสำหรับการใช้งานในส่วนอื่นๆ จะเหลือน้อยลง อย่างไรก็ตาม การกำหนดค่าบัฟเฟอร์ภาครับที่น้อยเกินไป จะทำให้เกิดการโอเวอร์โฟลว์หรือข้อมูลล้นบัฟเฟอร์ เว้นแต่จะมีการใช้แฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมก็คือค่า 1,024 ซึ่งเป็นค่าเริ่มต้นนั่นเอง แต่ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลว์แล้วจึงค่อยปรับเพิ่มค่าขนาดของบัฟเฟอร์ให้มีค่ามากขึ้น

2.8.3.7 InputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.InputLen[=value]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ “0” การกำหนดค่าเท่ากับ “0” จะทำให้คำสั่ง Input ของ MSCOMM อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาครับมากเท่ากับจำนวน InputLen คำสั่ง Input จะส่งค่าว่างกลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลที่อยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้ มักจะใช้กับการอ่านค่าข้อมูลจากเครื่องมือหรือเครื่องจักรที่กำหนดค่าขนาดความยาวของข้อมูลเอาไว้แล้ว

ตัวอย่างโปรแกรมการอ่านค่าตัวอักษรออกมา 10 ตัวอักษร

```
Private Command_click()
Dim Commdata as string
Mscomm1.inputlen=10
Commdata = mscomm1.input
End sub
```

2.8.3.8 InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งานคำสั่ง

Object.inputmode[=value]

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภท คือ

ComInputModeText สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น "0" และค่าเริ่มต้นของการรับค่าข้อมูลก็จะเป็นค่านี้

ComInputModeBinary สำหรับข้อมูลอื่นๆ ซึ่งจะเก็บอยู่ในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูล

ตัวอย่างการใช้งาน InputMode ต่อไปนี้จะทำการอ่านค่าข้อมูล 10 ไบต์จากพอร์ตอนุกรมและเก็บข้อมูลไว้ในตัวแปรแบบอาร์เรย์ ชนิดข้อมูลเป็นแบบไบต์

```
Private sub
Dim Buffer as Variant
Dim Arr() as Byte
MSComm1.CommPort = 1
MSComm1.PortOpen = True
MSComm1.InputMode = comInputModeBinary
Do Until MSComm1.InBufferCount < 10
Input buffer
        Doevent
Loop
Buffer = MSComm1.Input
Arr = Buffer
End Sub
```

2.8.3.9 Output

ใช้ในการส่งขบวนข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

Object.output[=value]

ค่า value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารีก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลตัวอักษรเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่ จะต้องกำหนดชนิดของข้อมูลเป็นแบบ Variant และข้อมูลภายในแบบ Byte

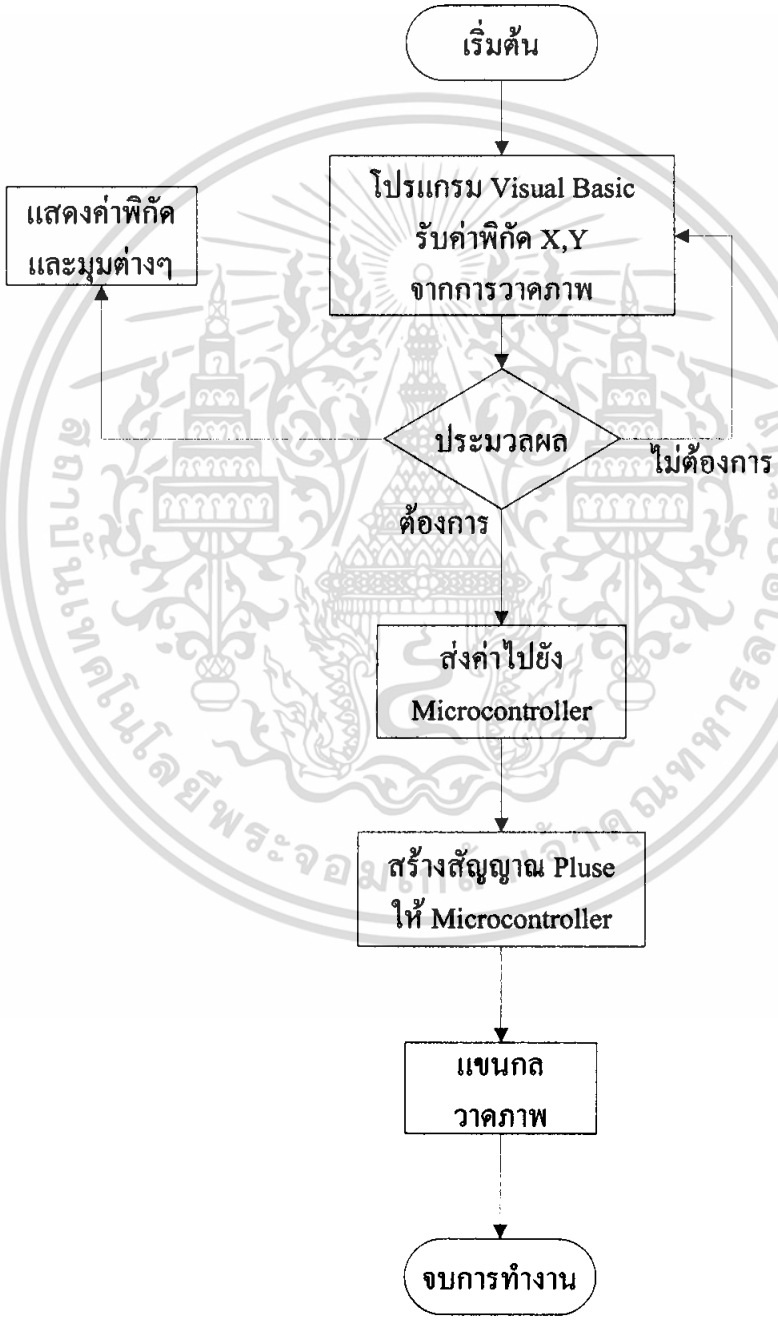
ตัวอย่างของโปรแกรมการส่งค่าที่ป้อนจากคีย์บอร์ดไปยังพอร์ตอนุกรม โดยการใช้คุณสมบัติ

```
Dim Buffer as Variant
MSComm1.Commport = 1
MSComm1.PortOpen = True
Buffer = chr$(KeyAsCII)
MSComm1.Output = Buffer
End Sub
```

บทที่ 3
หลักการออกแบบ

3.1 หลักการทำงานของแขนกล

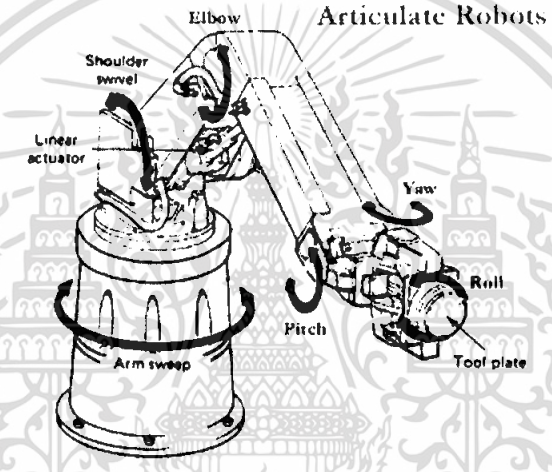
หลักการทำงานของแขนกล แสดงดัง Flow Chart ต่อไปนี้



3.2 โครงสร้างของแขนกล

รูปแบบของแขนกลมีหลายรูปแบบ หลายลักษณะการทำงาน ทำให้การที่จะออกแบบจึงแตกต่างด้วยรูปร่าง ลักษณะและมีเงื่อนไขในการทำงานมากมาย ดังนั้นการออกแบบโครงสร้างของแขนกลวาดภาพ ในเบื้องต้นต้องเข้าใจหลักการทำงานของแขนกลก่อน ว่ามีหลักการทำงานอย่างไรบ้าง และควรเลือกรูปแบบที่ง่าย เหมาะสมกับการใช้งาน ซึ่งมีขั้นตอนการออกแบบ ดังนี้

1. พิจารณาเงื่อนไขสิ่งที่ต้องการ จำนวนแกน รูปแบบ ซึ่งแขนกล 4 แกน ควรมีรูปแบบเป็นแบบอาร์ติคูเลตแนวตั้ง ดังรูปที่3.1 จะทำให้การทำงานได้พื้นที่การทำงานที่กว้างและมีความยืดหยุ่นสูงในการเข้าไปยังจุดต่าง

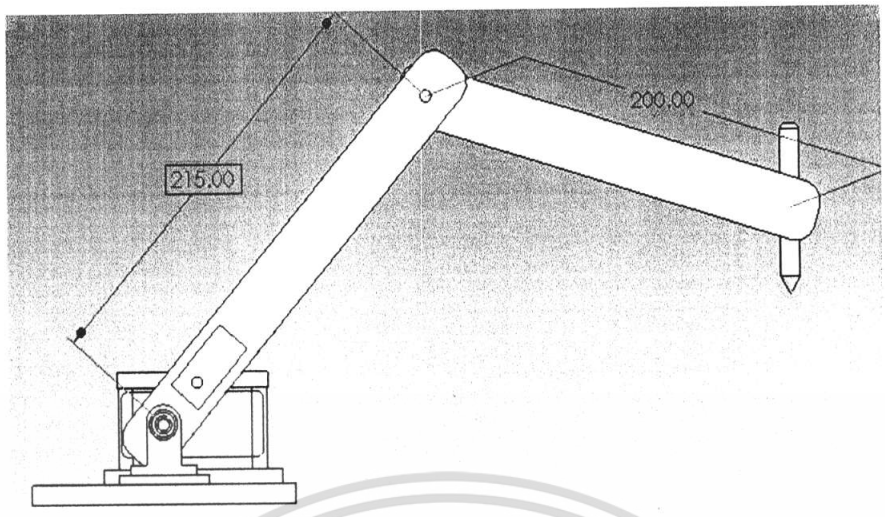


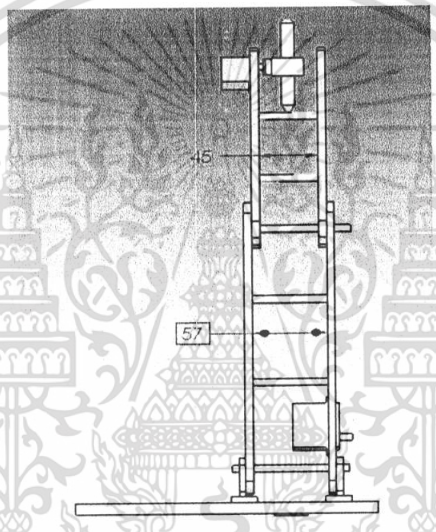
รูปที่ 3.1 แขนกลแบบอาร์ติคูเลตแนวตั้ง

2. เขียนแบบของแขนกล โดยได้รูปแบบดังรูปที่ 3.2 มีส่วนประกอบดังนี้

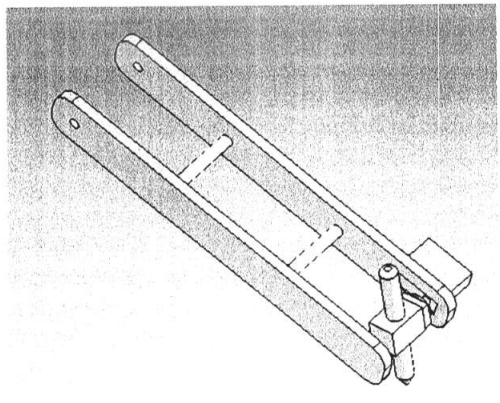
แขนท่อนบน ขนาดกว้าง 45 มม. ยาว 200 มม. ส่วนปลายของแขนยึดปากกาสำหรับวาดภาพและมอเตอร์ควบคุมจุดหมุนของปากกา

แขนท่อนล่าง ขนาดกว้าง 57 มม. ยาว 215 มม. ตัวแขนด้านล่าง เจาะรูยึดมอเตอร์สำหรับขับแขนท่อนบน ส่วนมอเตอร์ขับแขนท่อนล่างนั้น วางข้างๆ แขนท่อนล่าง

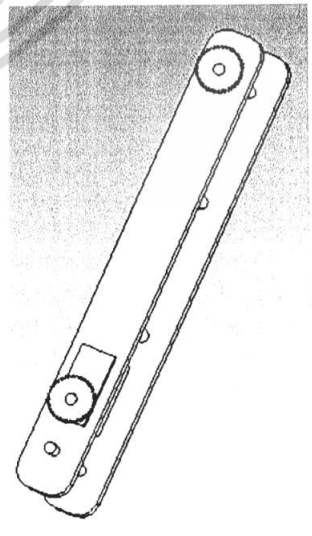




รูปที่ 3.2 แบบโครงสร้างของแขนกล

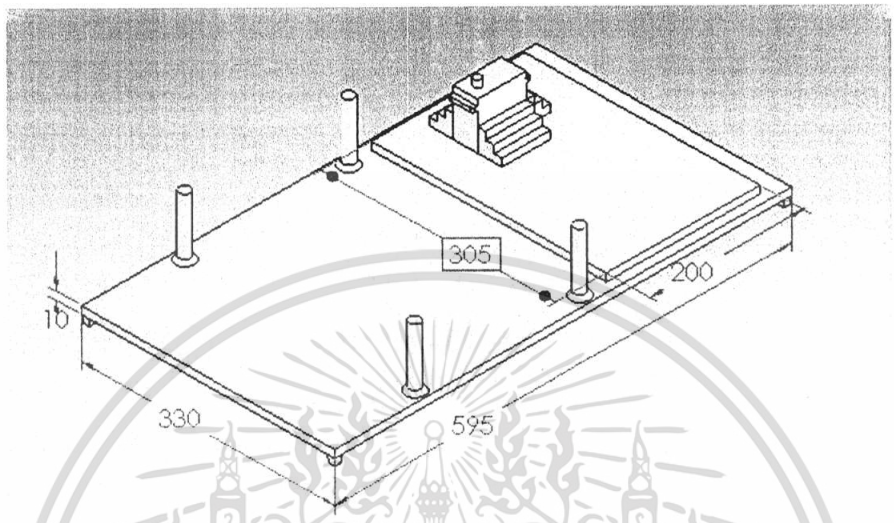


รูปที่ 3.3 แขนท่อนบน



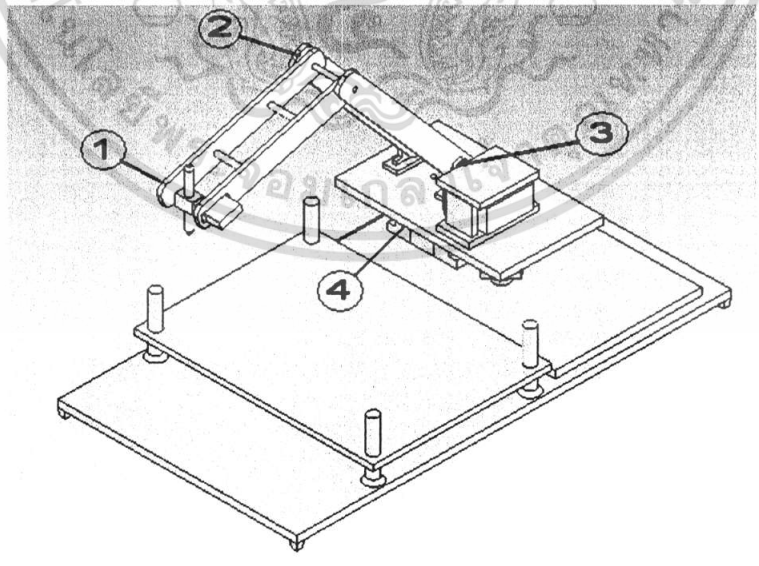
รูปที่ 3.4 แขนท่อนล่าง

3. ฐาน มีขนาดกว้าง 330 มม. ยาว 595 มม.และสูง 10 มม. ในส่วนของฐานประกอบด้วยพื้นที่สำหรับวางกระดาษ A4 โดยมีเสาตั้งขึ้นมาเพื่อวางแผ่นรองกระดาษ A4 ที่ยึดมอเตอร์สำหรับขับแขนกล และพื้นที่สำหรับวางบอร์ดควบคุมการทำงานแขนกล โดยบอร์ดนี้จะวางอยู่ข้างมอเตอร์

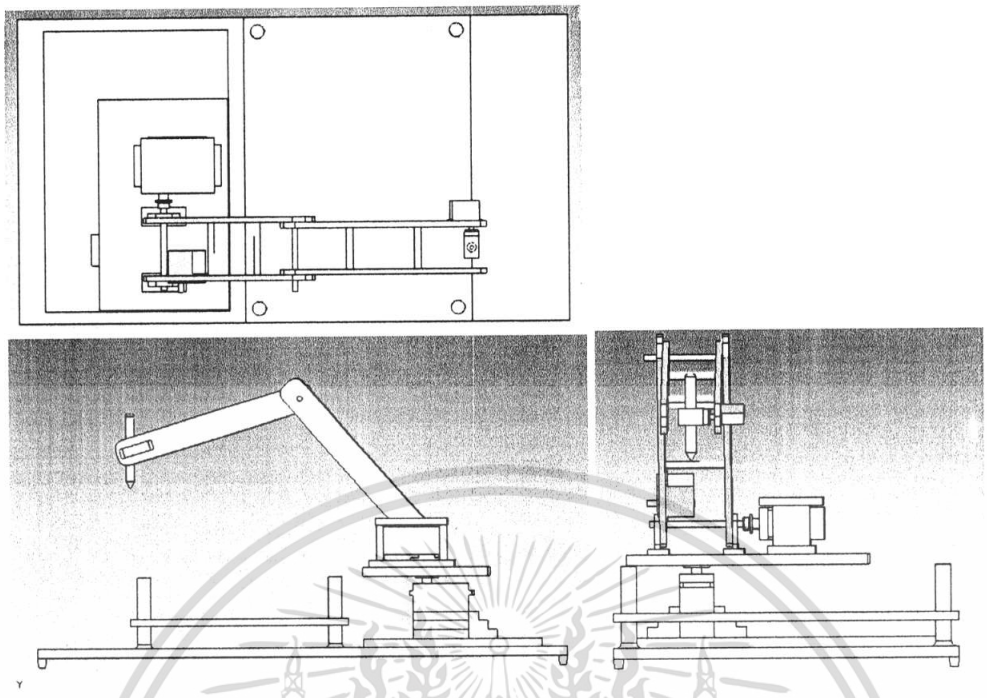


รูปที่ 3.5 ฐาน

4. เมื่อประกอบส่วนของแขนกลเข้ากับฐาน จะได้โครงสร้างของแขนกลและตำแหน่งของจุดหมุน ดังรูปที่ 3.6

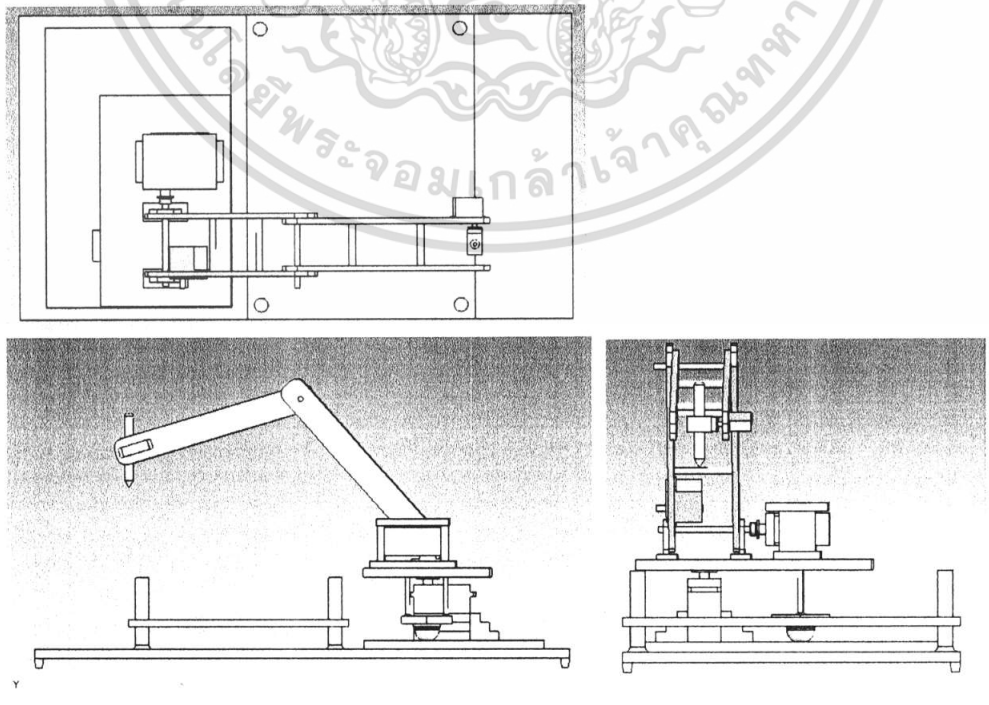


รูปที่ 3.6 โครงสร้างของแขนกลและตำแหน่งของจุดหมุน



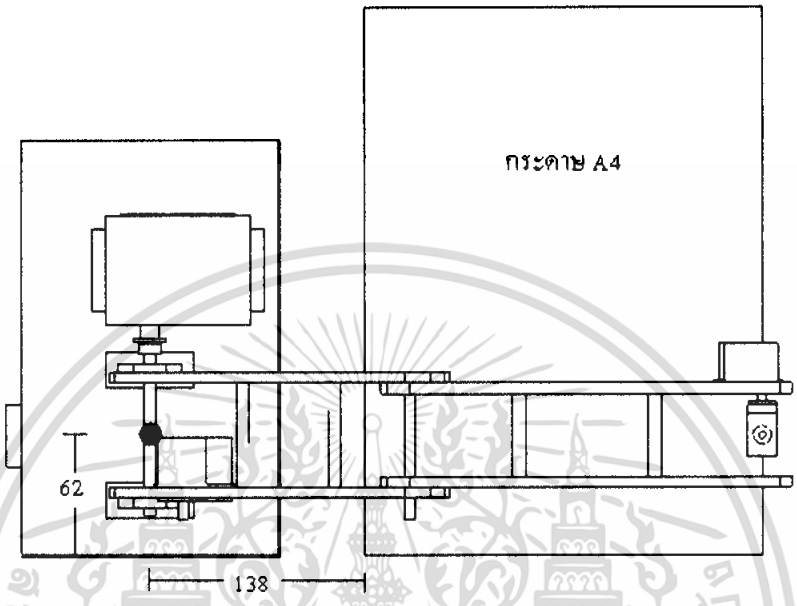
รูปที่ 3.7 Multiview โครงสร้างของแขนกล

5. จากรูปที่ 3.7 Multiview โครงสร้างของแขนกล สังเกตเห็นว่าจุดหมุนด้านล่างของแขนกลนั้นไม่สมดุล จึงทำการแก้ไขโดยติดตั้งล้อเพิ่มเข้าไปเพื่อรักษาความสมดุล ดังนั้นจะได้โครงสร้างของแขนกลใหม่ดังรูปที่ 3.8



รูปที่ 3.8 Multiview โครงสร้างของแขนกลเมื่อสมดุล

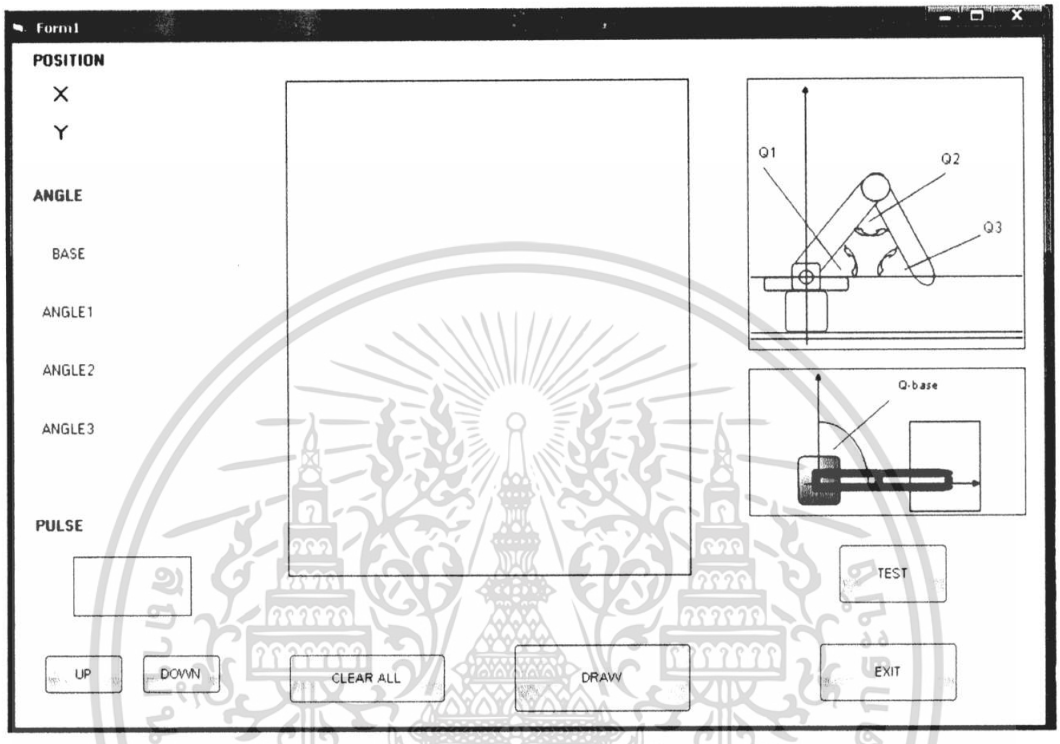
6. หาตำแหน่งระยะห่างระหว่างแขนกลกับกระดาษ A4 ในเบื้องต้นหาขอบเขตที่แขนกลสามารถวาดภาพได้ โดยใช้โปรแกรม Matlab ช่วยในการคำนวณ จะได้ขอบเขตที่แขนกลสามารถวาดภาพได้ และสามารถระบุตำแหน่งระยะห่างระหว่างแขนกลกับกระดาษ A4 ได้ดังรูปที่ 3.9



รูปที่ 3.9 ตำแหน่งระหว่างแขนกลกับกระดาษ A4

3.3 หน้าต่างสำหรับวาดภาพ

หน้าต่างสำหรับวาดภาพบนหน้าจอคอมพิวเตอร์ ออกแบบโดยใช้โปรแกรมVisual Basic 6 ซึ่งหน้าต่างที่ได้ออกแบบมีลักษณะดังรูปที่ 3.10

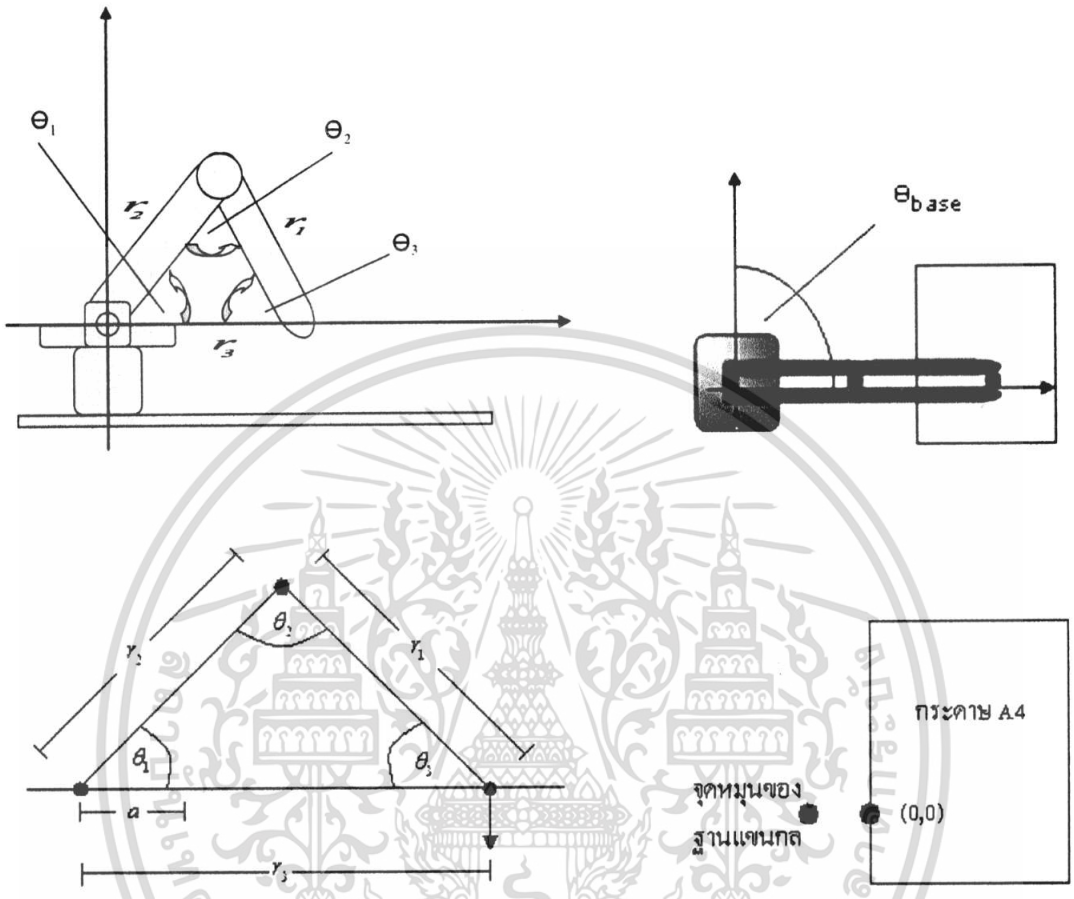


รูปที่ 3.10 หน้าต่างสำหรับวาดภาพบนหน้าจอคอมพิวเตอร์

โดยมีรายละเอียดดังนี้

POSITION	แสดงตำแหน่งพิกัด x และ y ในการเขียน
ANGLE	
- BASE	แสดงค่ามุมมอเตอร์ที่ใช้ขับทั้งแขน
- ANGLE 1	แสดงค่ามุมมอเตอร์ที่ใช้ขับแขนท่อนล่าง
- ANGLE 2	แสดงค่ามุมมอเตอร์ที่ใช้ขับแขนท่อนบน
- ANGLE 3	แสดงค่ามุมมอเตอร์ที่ใช้ขับปากกา
PULSE	แสดงค่าสัญญาณพัลส์ที่ส่งให้กับมอเตอร์
CLEAR ALL	ลบภาพทั้งหมดที่วาด
DRAW	วาดภาพ
EXIT	ออกจากโปรแกรม
TEST	แสดงการทดสอบว่ารับค่าสัญญาณพัลส์หรือไม่
รูปจำลองแขนกล	แสดงมุมองศาของแขน

3.4 การคำนวณค่ามุมองศาต่างๆ เพื่อส่งค่าไปยังแขนกล



เมื่อ $r_1 = 200$ mm., $r_2 = 215$ mm. และ $a = 125$ mm.

$$r_3 = \sqrt{(x+a)^2 + y^2}$$

$$r_1^2 = r_2^2 + r_3^2 - 2r_2 r_3 \cos\theta_1$$

$$\theta_1 = \cos^{-1}\left(\frac{r_2^2 + r_3^2 - r_1^2}{2r_2 r_3}\right)$$

$$r_3^2 = r_2^2 + r_1^2 - 2r_1 r_2 \cos\theta_2$$

$$\theta_2 = \cos^{-1}\left(\frac{r_2^2 + r_1^2 - r_3^2}{2r_1 r_2}\right)$$

$$r_2^2 = r_1^2 + r_3^2 - 2r_1 r_3 \cos\theta_3$$

$$\theta_3 = \cos^{-1}\left(\frac{r_1^2 + r_3^2 - r_2^2}{2r_1 r_2}\right)$$

$$\theta_{base} = \tan^{-1}\frac{y}{x+a}$$

ดังนั้นค่ามุมที่ส่งให้มอเตอร์คำนวณได้ดังนี้

มุมของมอเตอร์ฐาน $= \tan^{-1}\frac{y}{x+a}$

มุมของมอเตอร์ขับแขนท่อนล่าง $= \theta_1$

มุมของมอเตอร์ขับแขนท่อนบน $= \theta_2 - (180 - 90 - \theta_1)$

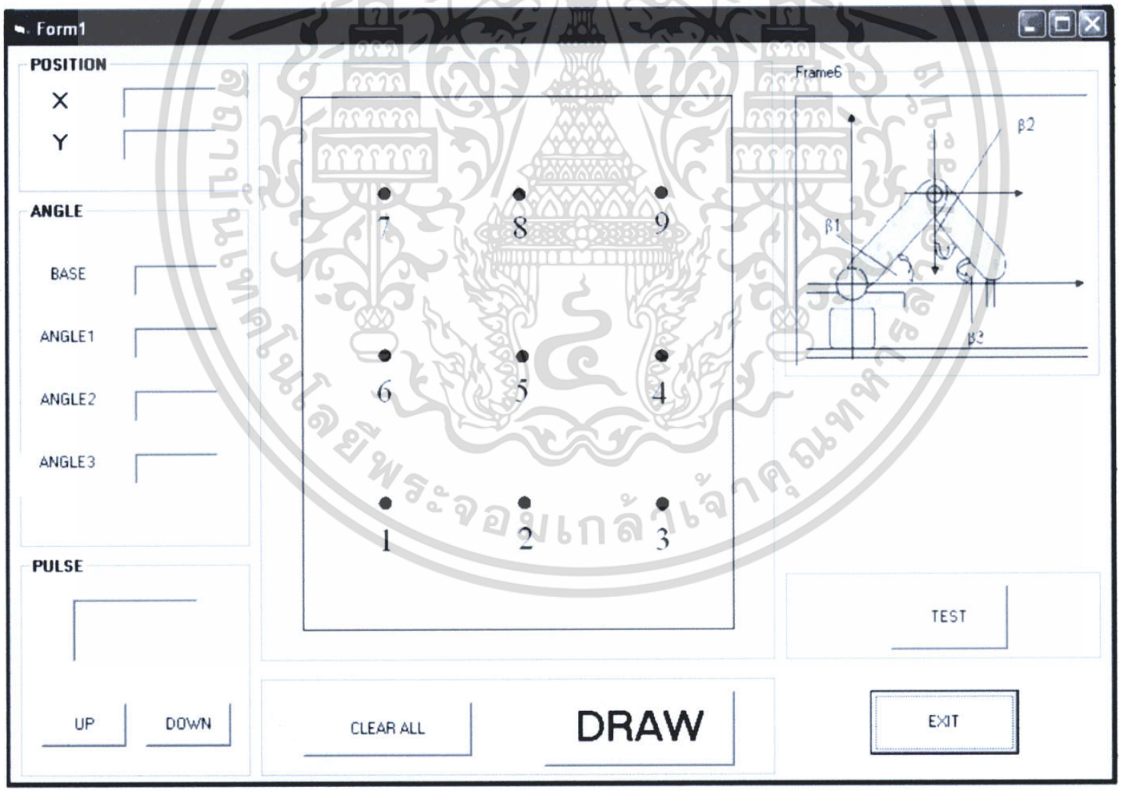
มุมของมอเตอร์ขับปากกา $= 180 - (\theta_1 + 90)$

บทที่ 4
การทดลองและผลการทดลอง

ในบทนี้กล่าวถึงการทดลองและผลการทดลองการทำงานของแขนกล เมื่อสั่งให้แขนกลเคลื่อนที่ไปยังตำแหน่งต่างๆ บนกระดาษ A4

4.1 การทดลองการทำงานของแขนกล

การทดลองนี้เป็นการทดลองหาค่าตำแหน่งมุมองศาของแขนกลเมื่อเคลื่อนที่ไปยังตำแหน่งต่างๆ ตามการวาดภาพในโปรแกรม Visual Basic เพื่อตรวจสอบความแม่นยำของแขนกลว่าเป็นไปตามที่กำหนดหรือไม่ โดยทำการทดลองให้แขนกลเคลื่อนที่ไปทีละจุด ดังรูปที่ 4.1 ซึ่งได้ผลการทดลองตามตารางที่ 4.1



รูปที่ 4.1 ตำแหน่งการเคลื่อนที่ของแขนกล

ตาราง 4.1 ผลการทดลองการเคลื่อนที่ของแขนกลไปยังตำแหน่งต่างๆ

ตำแหน่งที่	ตำแหน่งพิกัด		มุมในโปรแกรม VB (องศา)				มุมของแขนกล (องศา)			
	x	y	Base	θ_1	θ_2	θ_3	Base	θ_1	θ_2	θ_3
1	2	-4	-15	64	16	74	-8	45	40	85
2	10	-4	-10	54	30	60	-7	40	15	78
3	17	-4	-8	42	44	46	-8	30	17	75
4	17	7	13	41	45	45	15	30	15	75
5	10	8	19	52	32	58	20	40	18	75
6	2	8	26	62	19	71	25	50	18	83
7	2	19	38	52	32	58	36	45	20	76
8	10	19	38	52	32	58	36	45	20	76
9	18	18	27	30	57	33	30	22	50	70

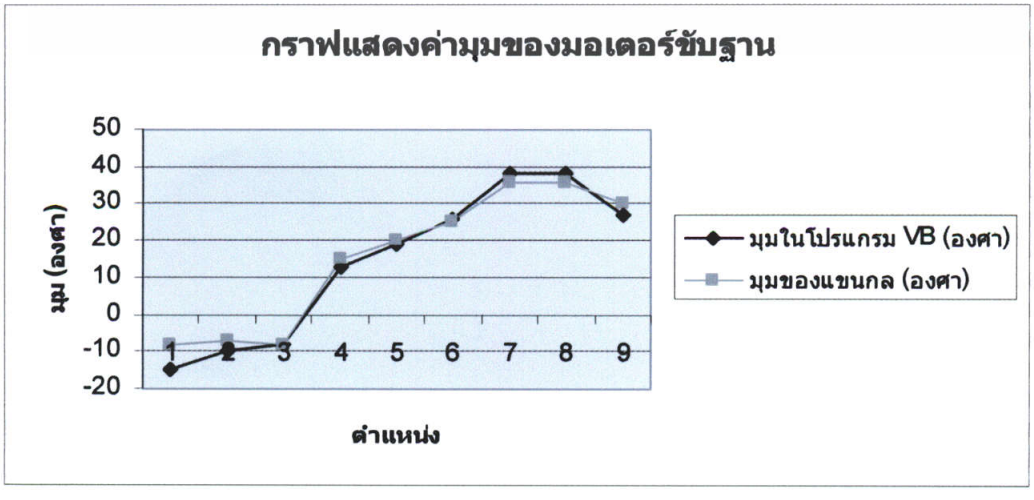
เมื่อ Base คือมุมมอเตอร์ที่ใช้ขับทั้งแขน

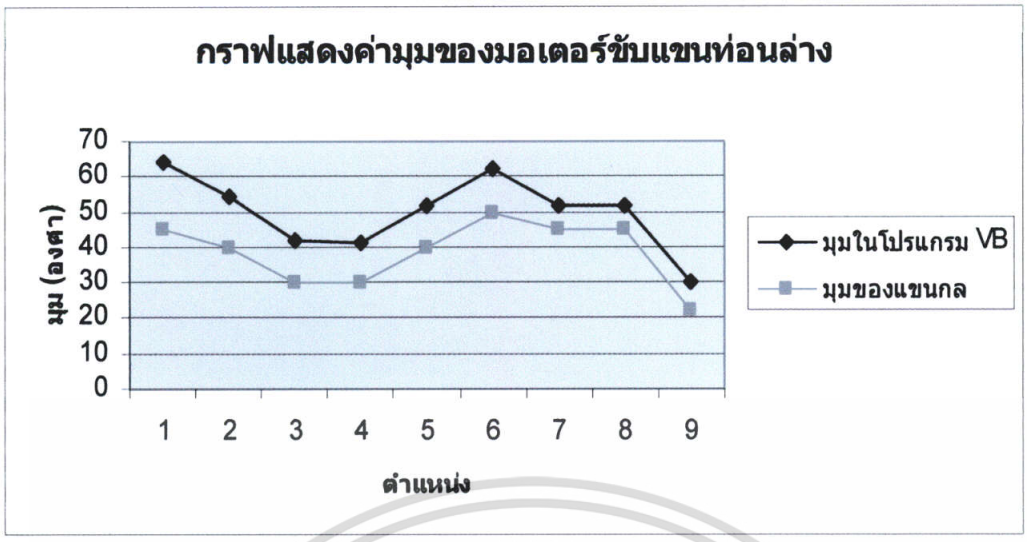
θ_1 มุมมอเตอร์ที่ใช้ขับแขนท่อนล่าง

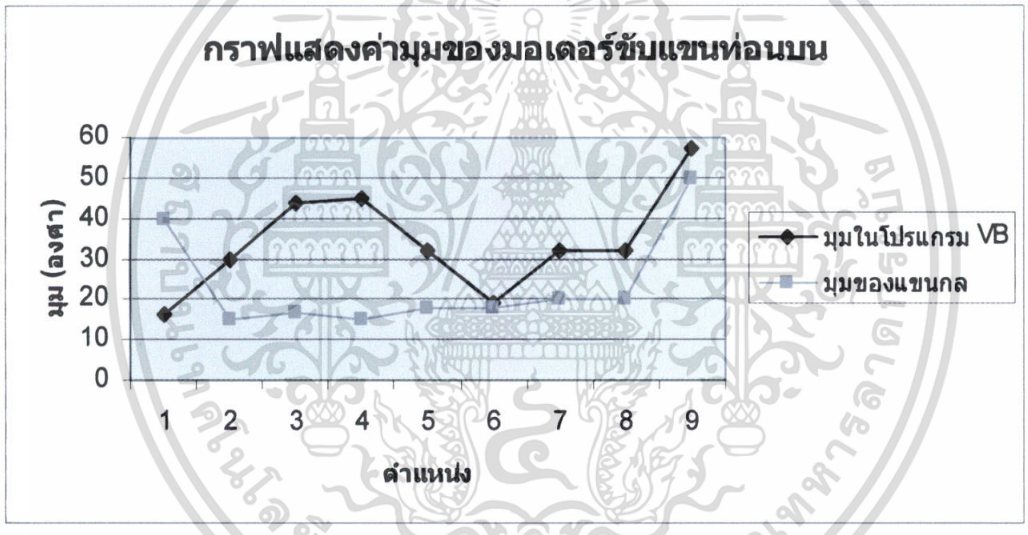
θ_2 มุมมอเตอร์ที่ใช้ขับแขนท่อนบน

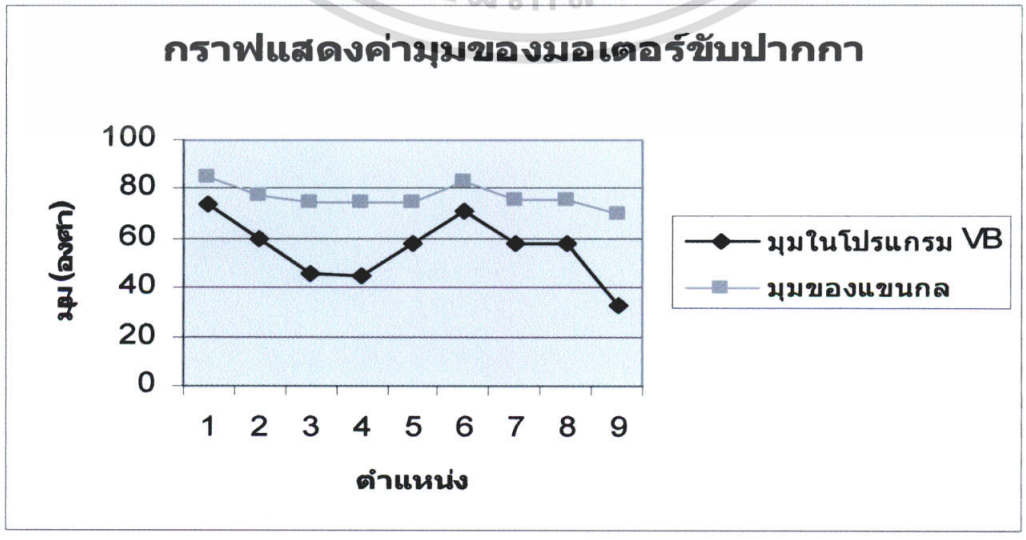
θ_3 มุมมอเตอร์ที่ใช้ขับปากกา

จากตารางสามารถนำมาวาดกราฟเปรียบเทียบค่ามุมองศาระหว่างค่ามุมองศาที่โปรแกรมคำนวณได้กับค่ามุมองศาที่วัดได้จริงของมอเตอร์แต่ละตัวได้ดังนี้

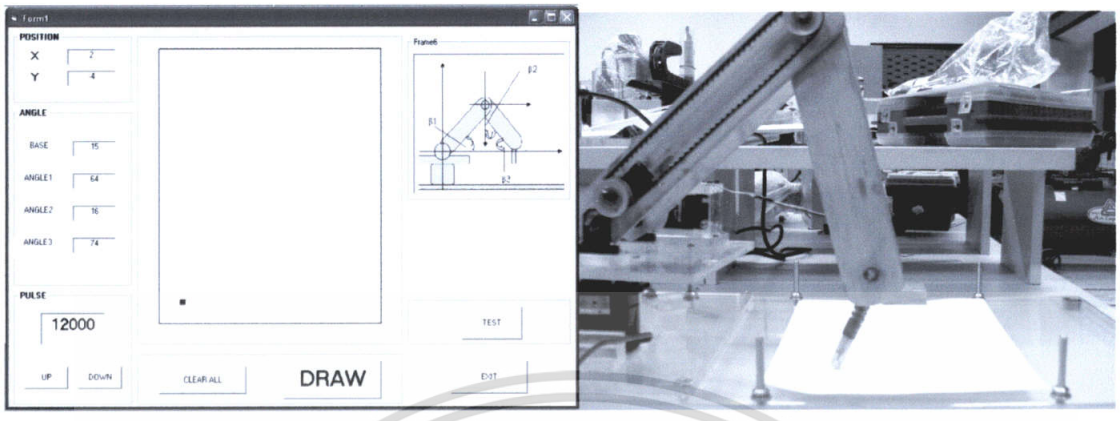



กราฟแสดงค่ามุมของมอเตอร์ขับแขนท่อนล่าง
มุม (องศา)
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9
ตำแหน่ง
มุมในโปรแกรม VB
มุมของแขนกล

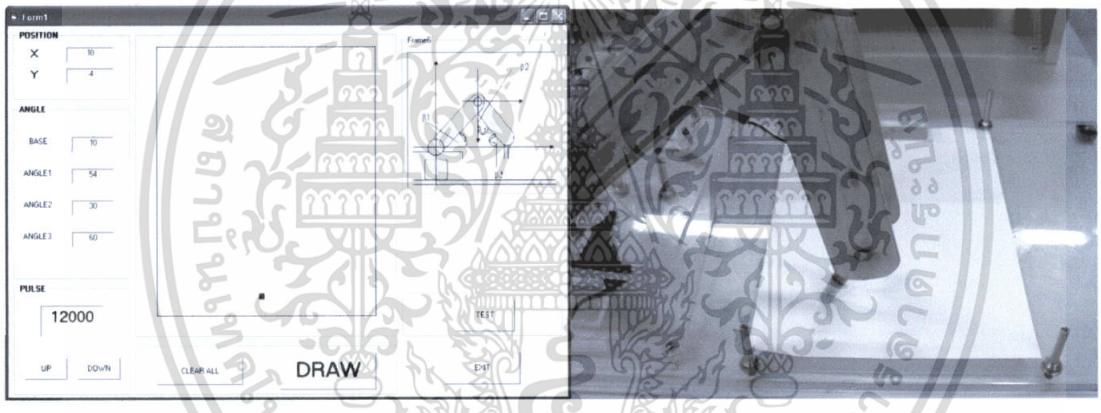

กราฟแสดงค่ามุมของมอเตอร์ขับแขนท่อนบน
มุม (องศา)
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9
ตำแหน่ง
มุมในโปรแกรม VB
มุมของแขนกล


กราฟแสดงค่ามุมของมอเตอร์ขับปากกา
มุม(องศา)
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9
ตำแหน่ง
มุมในโปรแกรม VB
มุมของแขนกล

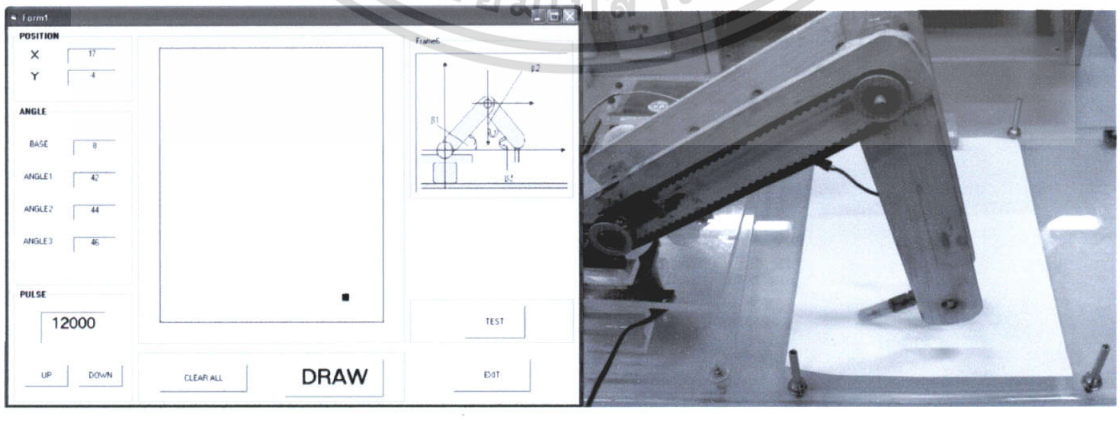
การทดลองการเคลื่อนที่ของแขนกลไปยังตำแหน่งต่างๆ สามารถดูจากรูปดังนี้



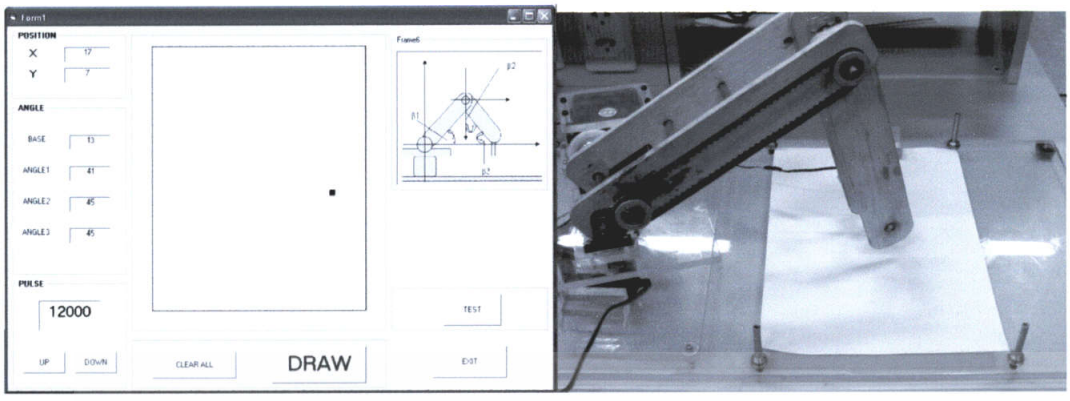
รูปที่ 4.2 เคลื่อนที่ไปยังตำแหน่งที่ 1



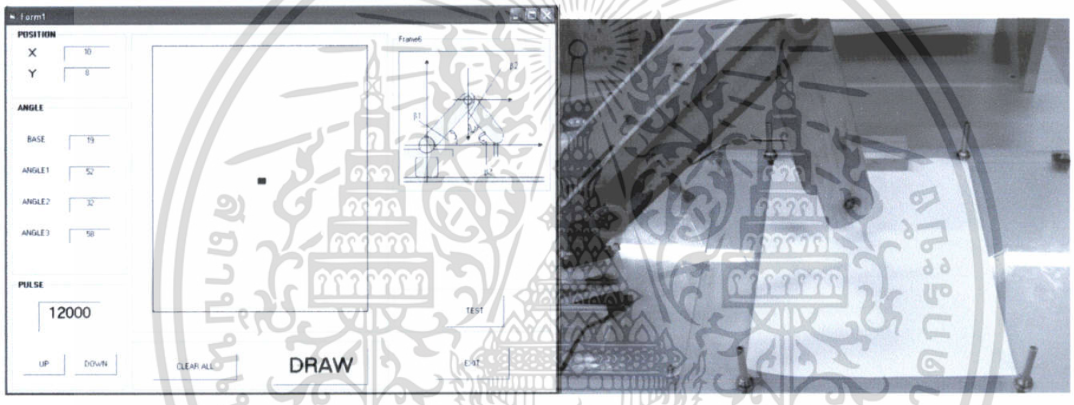
รูปที่ 4.3 เคลื่อนที่ไปยังตำแหน่งที่ 2



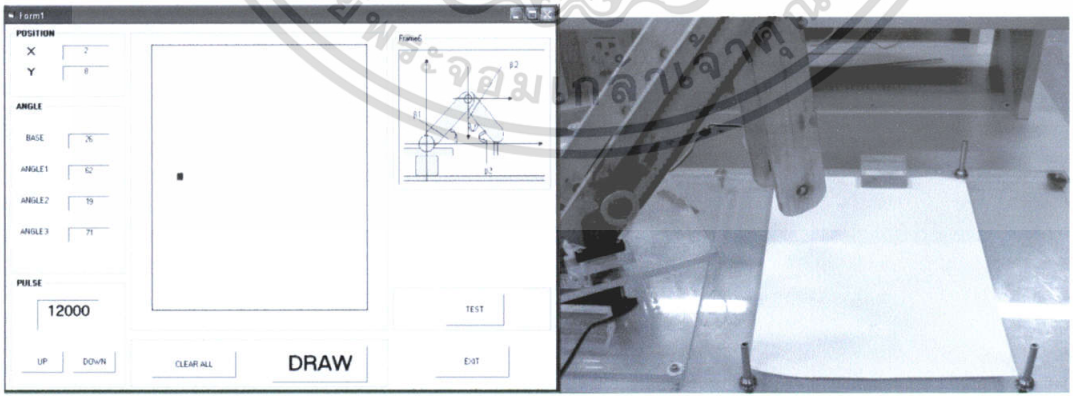
รูปที่ 4.4 เคลื่อนที่ไปยังตำแหน่งที่ 3



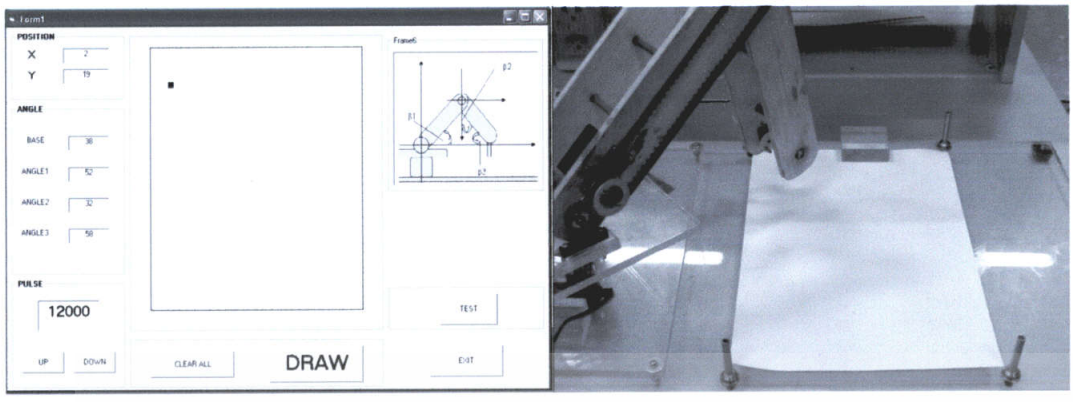
รูปที่ 4.5 เคลื่อนที่ไปยังตำแหน่งที่ 4



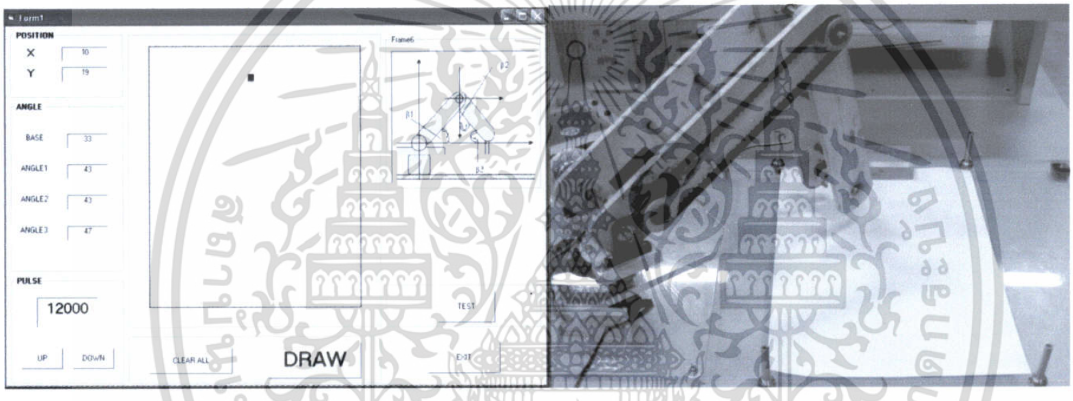
รูปที่ 4.6 เคลื่อนที่ไปยังตำแหน่งที่ 5



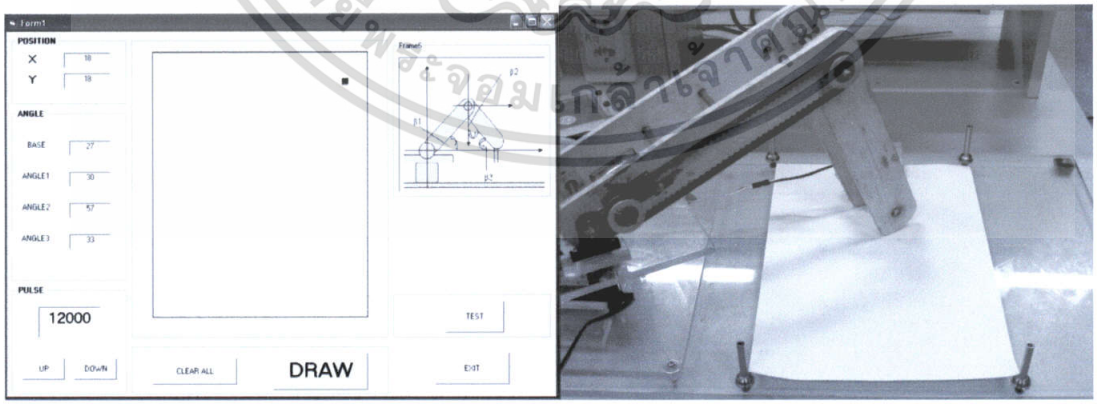
รูปที่ 4.7 เคลื่อนที่ไปยังตำแหน่งที่ 6



รูปที่ 4.8 เคลื่อนที่ไปยังตำแหน่งที่ 7



รูปที่ 4.9 เคลื่อนที่ไปยังตำแหน่งที่ 8



รูปที่ 4.10 เคลื่อนที่ไปยังตำแหน่งที่ 9

จากผลการทดลองสังเกตได้ว่าค่ามุมองศาจริงมีความคลาดเคลื่อนจากค่ามุมองศาที่ส่งมาจากโปรแกรม Visual Basic เป็นอย่างมาก ทั้งนี้น่าจะเป็นผลมาจากการติดตั้งแขนกลไม่สัมพันธ์กับโปรแกรม และเมื่อแขนกลเคลื่อนที่ไปยังตำแหน่งที่ไกลขึ้น แขนกลก็ยิ่งกดต่ำลง ซึ่งน่าจะมีสาเหตุมาจากค่าโมเมนต์ จึงได้ปรับแต่งแขนกลและทดลองใหม่อีกครั้ง

4.2 การทดลองการทำงานของแขนกลเมื่อปรับแต่งแขนกลแล้ว

การทดลองนี้เป็นการทดลองหาค่าตำแหน่งมุมองศาของแขนกลเมื่อเคลื่อนที่ไปยังตำแหน่งต่างๆ ตามการวาดภาพในโปรแกรม Visual Basic เมื่อมีการปรับแต่งแขนกลแล้ว ซึ่งได้ผลการทดลองตามตารางที่ 4.2

ตาราง 4.2 ผลการทดลองการเคลื่อนที่ของแขนกลที่ปรับแต่งแล้วไปยังตำแหน่งต่างๆ

ตำแหน่งที่	ตำแหน่งพิกัด		มุมในโปรแกรม VB (องศา)				มุมของแขนกล (องศา)			
	x	y	Base	θ_1	θ_2	θ_3	Base	θ_1	θ_2	θ_3
1	2	-4	-15	78	18	74	-14	70	22	85
2	10	-4	-10	65	39	60	-8	60	33	63
3	17	-4	-8	50	60	46	-5	55	50	45
4	17	7	13	49	61	45	15	36	44	42
5	10	8	19	63	42	58	23	50	40	53
6	2	8	26	76	22	71	30	68	30	65
7	2	19	38	63	42	58	40	53	39	55
8	10	19	33	51	59	47	35	40	50	45
9	18	18	27	34	80	33	30	30	65	35

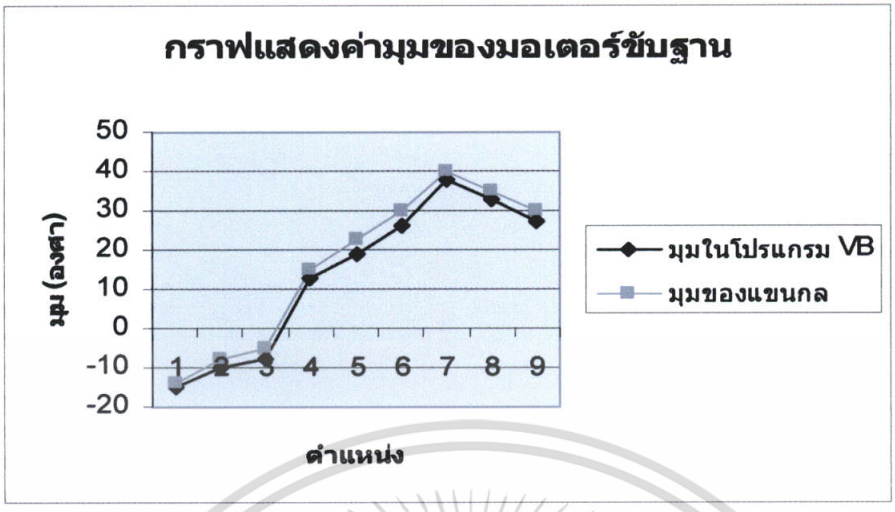
เมื่อ Base คือมุมมอเตอร์ที่ใช้ขับทั้งแขน

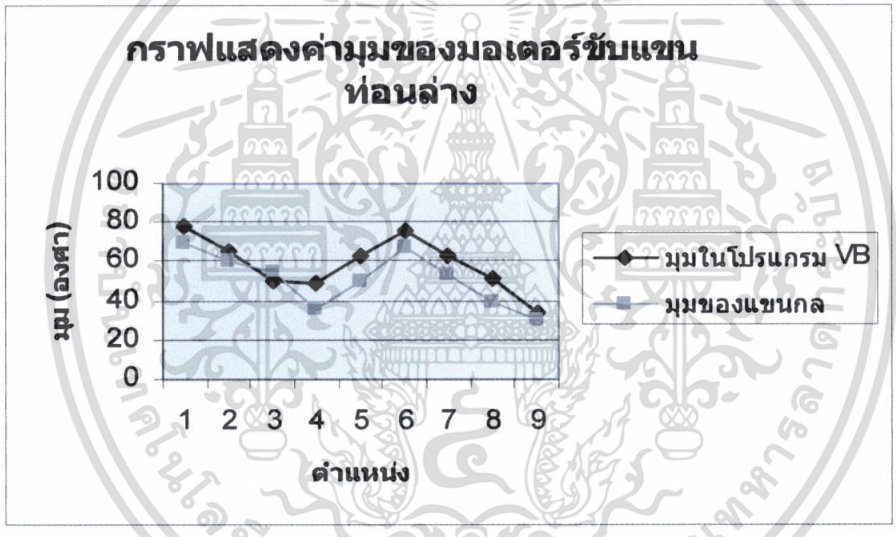
θ_1 มุมมอเตอร์ที่ใช้ขับแขนท่อนล่าง

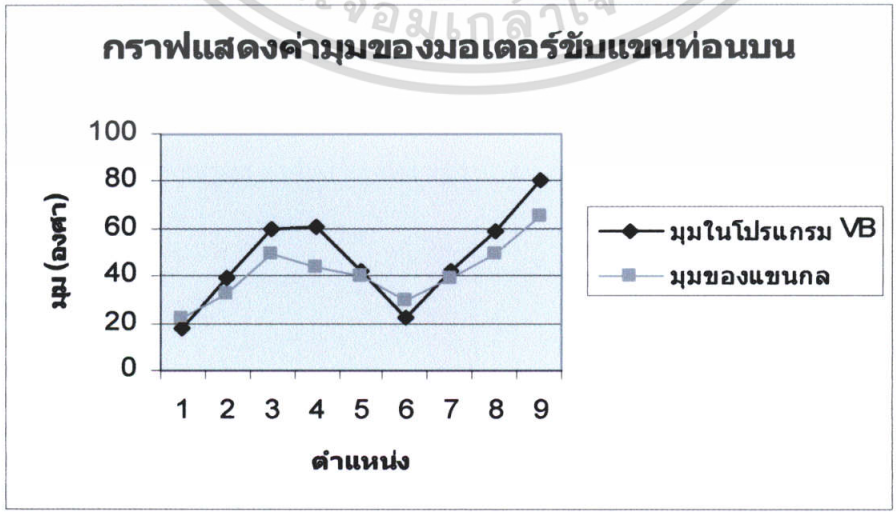
θ_2 มุมมอเตอร์ที่ใช้ขับแขนท่อนบน

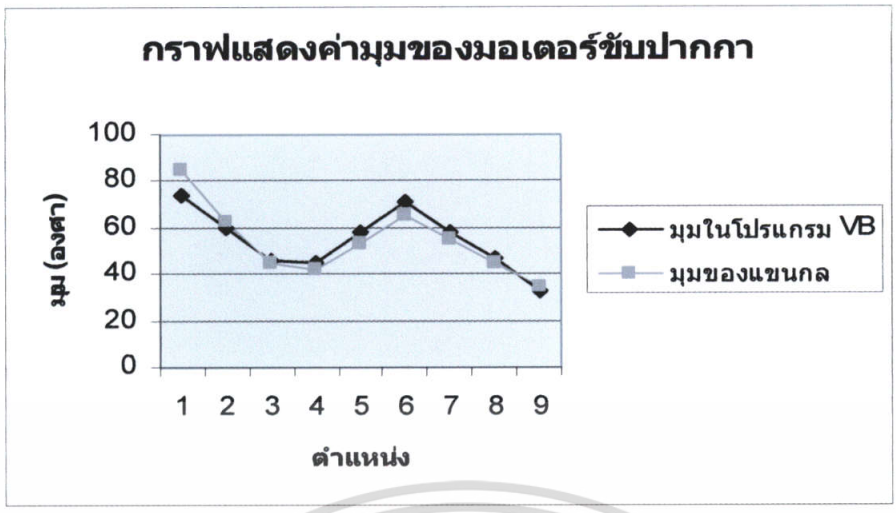
θ_3 มุมมอเตอร์ที่ใช้ขับปากกา

จากตารางสามารถนำมาวาดกราฟเปรียบเทียบค่ามุมองศาระหว่างค่ามุมองศาที่โปรแกรมคำนวณได้กับค่ามุมองศาที่วัดได้จริงของมอเตอร์แต่ละตัวได้ดังนี้


กราฟแสดงค่ามุมของมอเตอร์ขับฐาน
มุม (องศา)
50
40
30
20
10
0
-10
-20
1 2 3 4 5 6 7 8 9
ตำแหน่ง
มุมในโปรแกรม VB
มุมของแขนกล

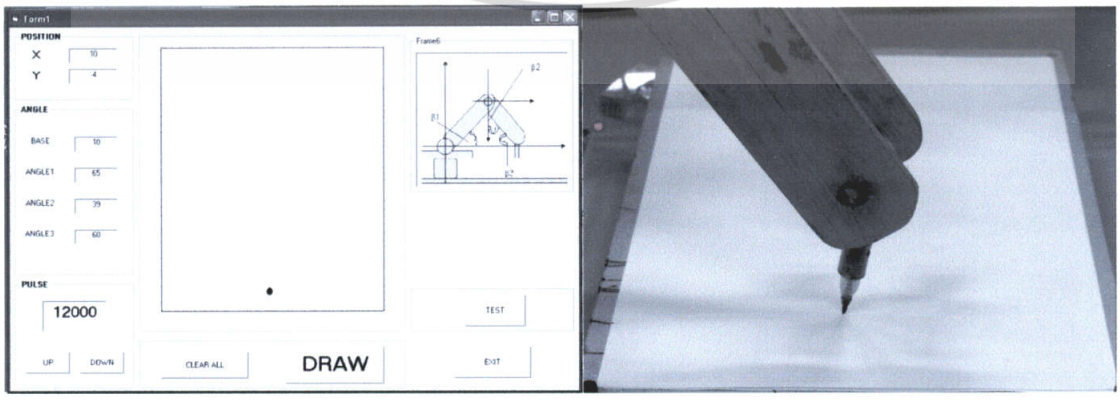

กราฟแสดงค่ามุมของมอเตอร์ขับแขนท่อนล่าง
มุม (องศา)
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9
ตำแหน่ง
มุมในโปรแกรม VB
มุมของแขนกล


กราฟแสดงค่ามุมของมอเตอร์ขับแขนท่อนบน
มุม (องศา)
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9
ตำแหน่ง
มุมในโปรแกรม VB
มุมของแขนกล

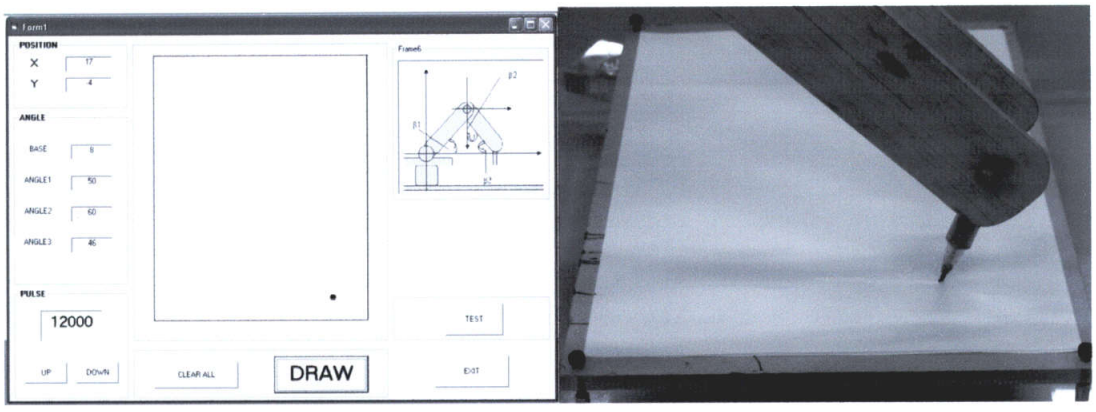


การทดลองการเคลื่อนที่ของแขนกลไปยังตำแหน่งต่างๆ เมื่อปรับแก้แขนกลแล้ว สามารถดูจากรูปดังนี้

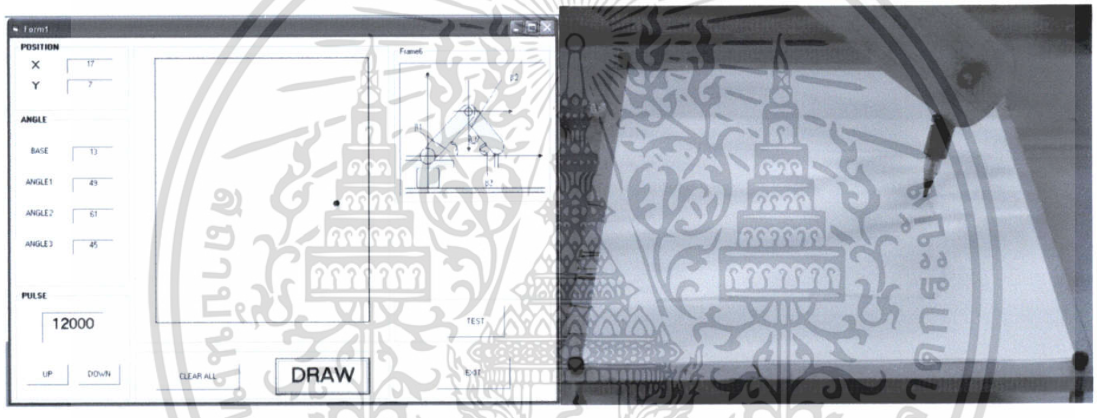
รูปที่ 4.11 เคลื่อนที่ไปยังตำแหน่งที่ 1



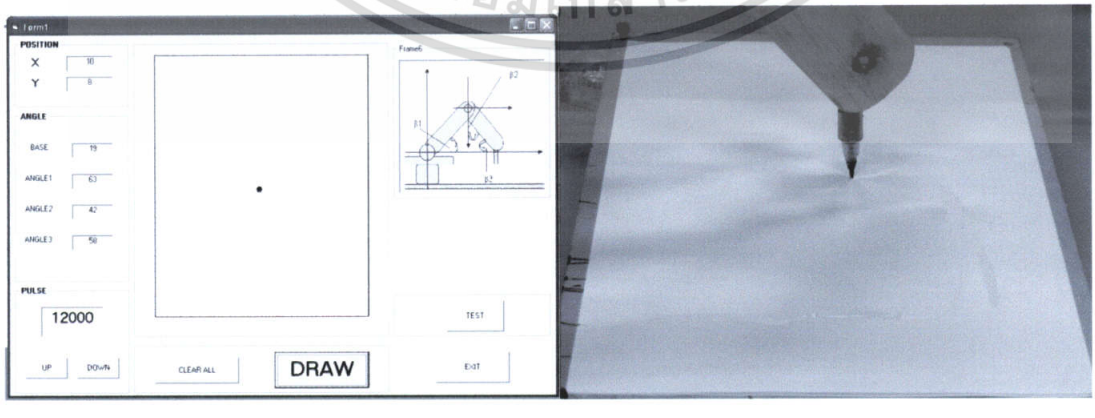
รูปที่ 4.12 เคลื่อนที่ไปยังตำแหน่งที่ 2



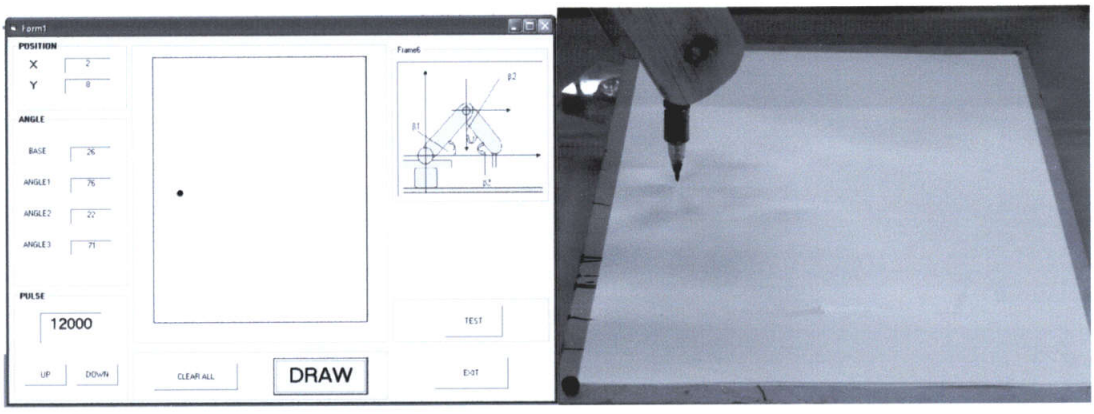
รูปที่ 4.13 เคลื่อนที่ไปยังตำแหน่งที่ 3



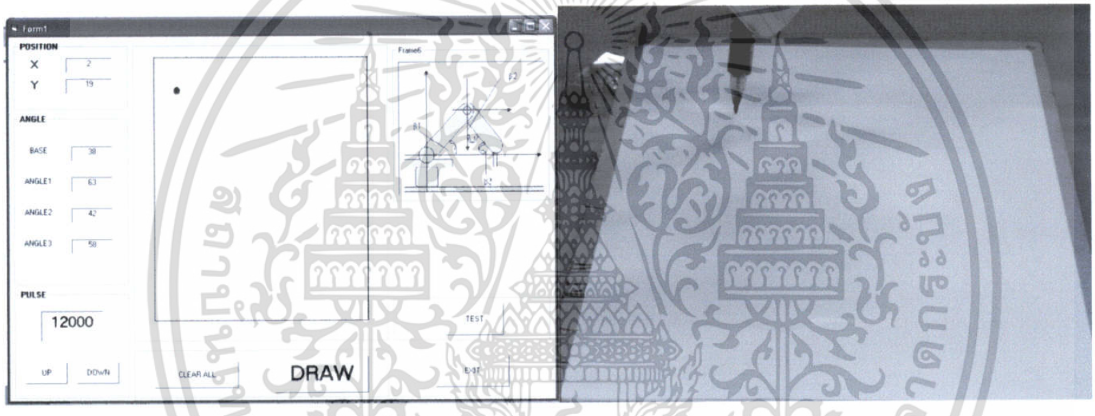
รูปที่ 4.14 เคลื่อนที่ไปยังตำแหน่งที่ 4



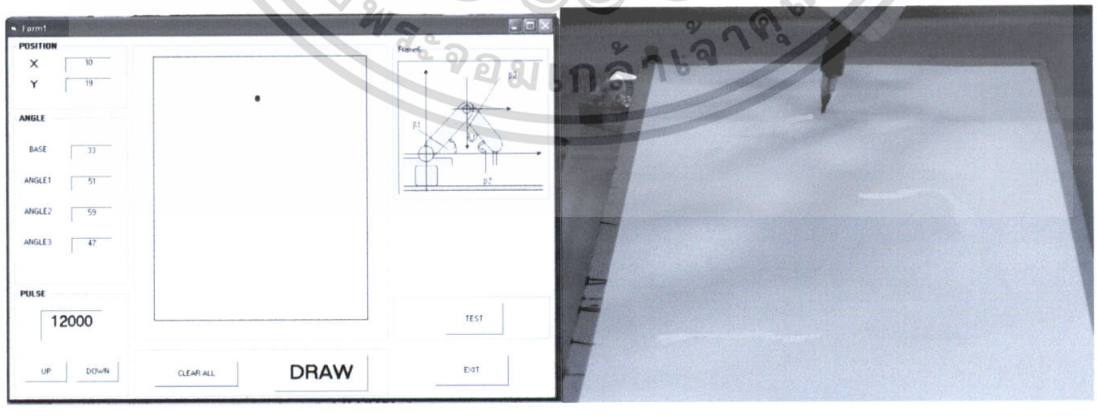
รูปที่ 4.15 เคลื่อนที่ไปยังตำแหน่งที่ 5



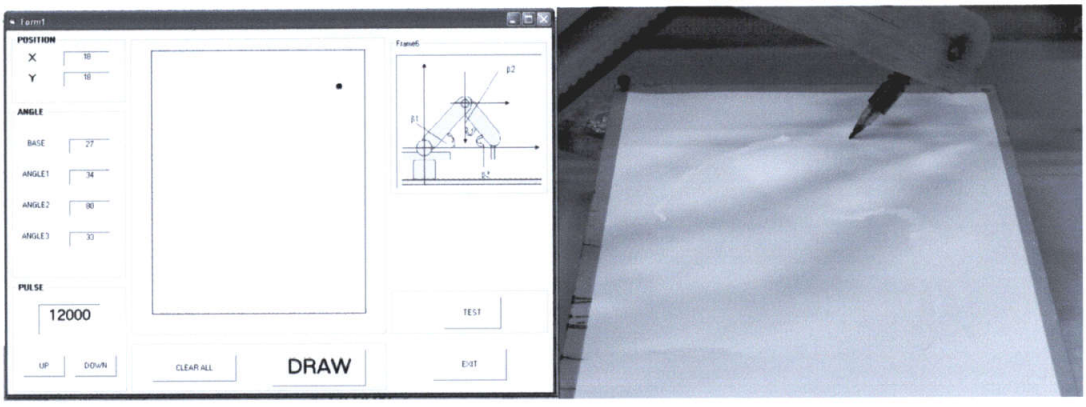
รูปที่ 4.16 เคลื่อนที่ไปยังตำแหน่งที่ 6



รูปที่ 4.17 เคลื่อนที่ไปยังตำแหน่งที่ 7



รูปที่ 4.18 เคลื่อนที่ไปยังตำแหน่งที่ 8



รูปที่ 4.19 เคลื่อนที่ไปยังตำแหน่งที่ 9

ในการทดลองครั้งที่ 2 แขนกลสามารถเคลื่อนที่ไปยังตำแหน่งต่างๆ ได้ดีกว่าการทดลองครั้งแรก แต่ยังมีความผิดพลาดอยู่ จึงได้ทำการตรวจสอบในส่วนของโปรแกรม ซึ่งพบว่าโปรแกรมมีข้อบกพร่อง จึงปรับแต่งแก้ไขโปรแกรม แล้วทำการทดลองซ้ำอีกครั้ง

4.3 การทดลองการทำงานของแขนกลเมื่อปรับแต่งโปรแกรมแล้ว

การทดลองนี้เป็นการทดลองหาค่าตำแหน่งมุมองศาของแขนกลเมื่อเคลื่อนที่ไปยังตำแหน่งต่างๆ ตามการวาดภาพในโปรแกรม Visual Basic เมื่อทำการปรับแต่งโปรแกรมแล้ว ซึ่งได้ผลการทดลองตามตารางที่ 4.3

ตาราง 4.3 ผลการทดลองการเคลื่อนที่ของแขนกลที่ปรับแต่งโปรแกรมแล้วไปยังตำแหน่งต่างๆ

ตำแหน่งที่	ตำแหน่งพิกัด		มุมในโปรแกรม VB (องศา)				มุมของแขนกล (องศา)			
	x	y	Base	θ_1	θ_2	θ_3	Base	θ_1	θ_2	θ_3
1	2	-4	-15	64	42	74	-12	68	43	69
2	10	-4	-10	54	67	59	-8	60	70	50
3	17	-4	-8	42	92	46	-6	45	90	45
4	17	7	13	41	94	45	15	45	95	40
5	10	8	19	52	70	58	22	55	75	50
6	2	8	26	62	47	71	30	75	42	68
7	2	19	38	52	70	58	41	60	70	50
8	10	19	33	43	90	47	35	47	90	43
9	18	18	27	30	117	33	30	33	120	27

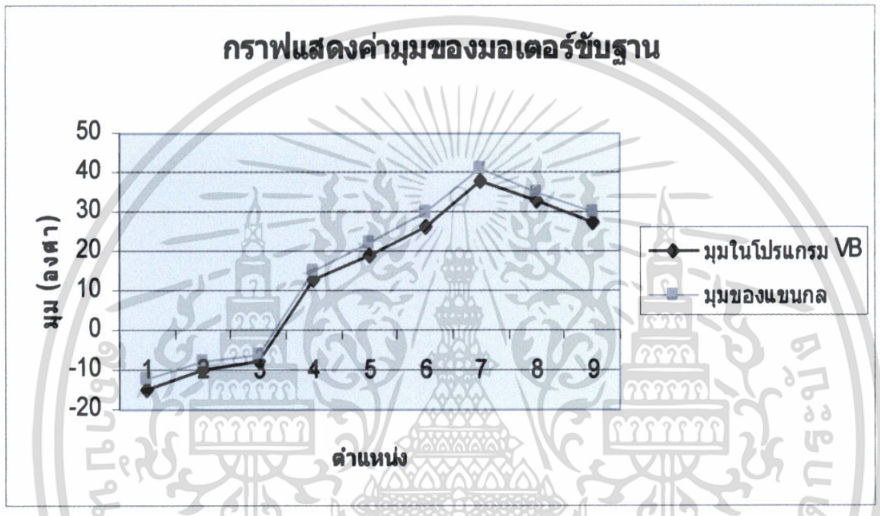
เมื่อ Base คือมุมมอเตอร์ที่ใช้ขับทั้งแขน

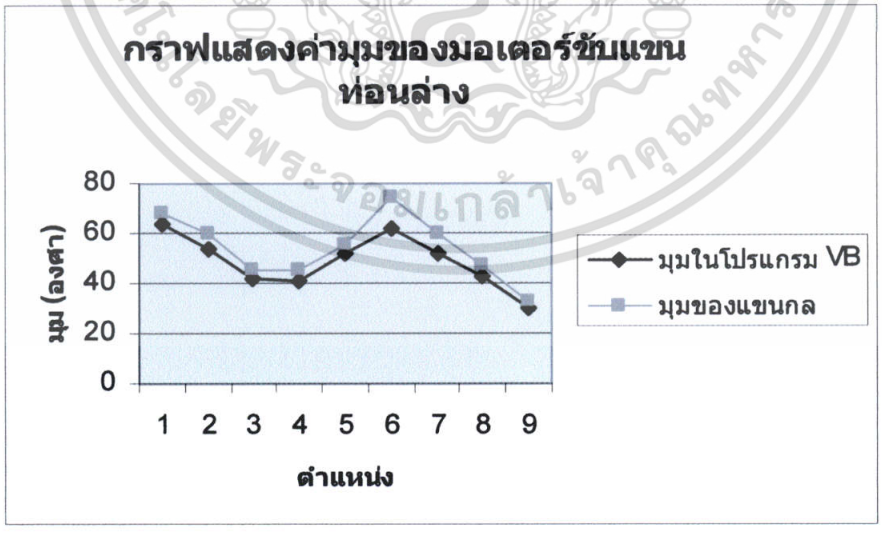
θ_1 มุมมอเตอร์ที่ใช้ขับแขนท่อนล่าง

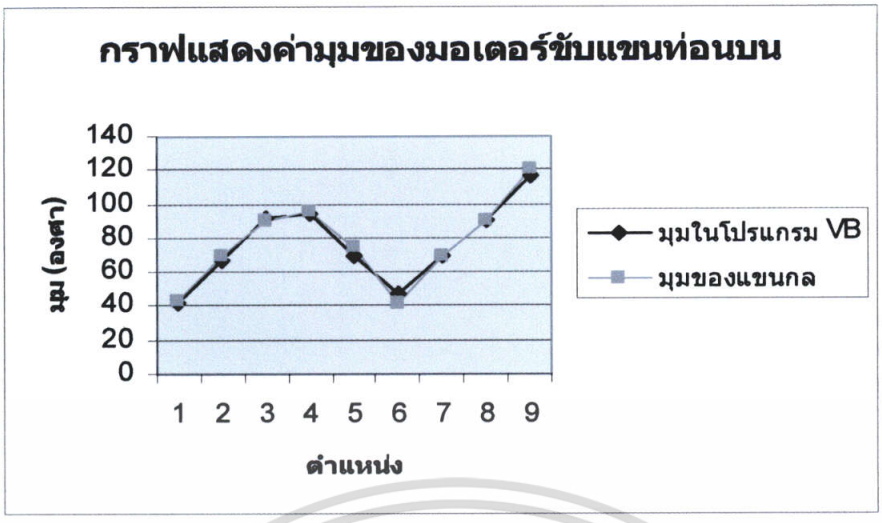
θ_2 มุมมอเตอร์ที่ใช้ขับแขนท่อนบน

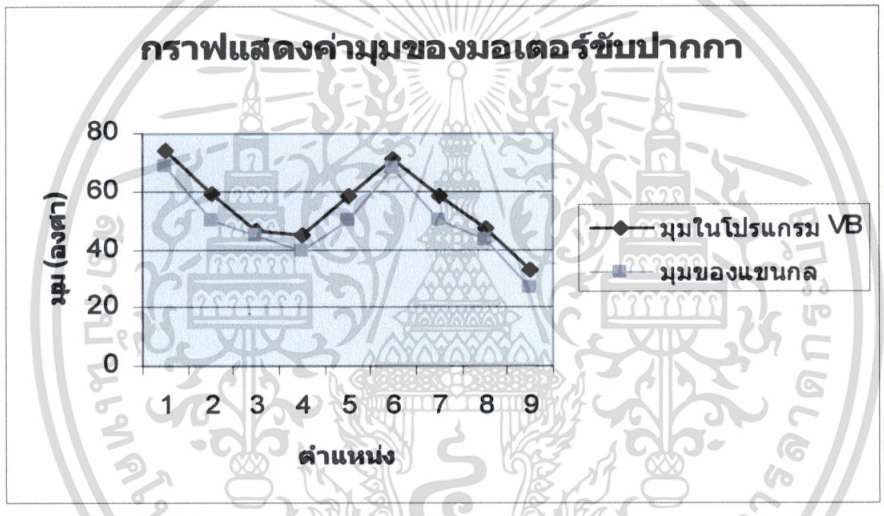
θ_3 มุมมอเตอร์ที่ใช้ขับปากกา

จากตารางสามารถนำมาวาดกราฟเปรียบเทียบค่ามุมองศาระหว่างค่ามุมองศาที่โปรแกรมคำนวณได้กับค่ามุมองศาที่วัดได้จริงของมอเตอร์แต่ละตัวได้ดังนี้









เมื่อมีการปรับแก้โปรแกรมและแขนกลแล้ว การเคลื่อนที่ของแขนกลมีความแม่นยำมากขึ้น สามารถเคลื่อนที่ไปยังตำแหน่งต่างๆ ได้ดีขึ้น

บทที่ 5
บทวิจารณ์และสรุป

5.1 สรุปผลการทดลอง

จากการทดลองการวาดภาพของแขนกลไปยังจุดต่างๆ ที่กำหนดขึ้นบนหน้าต่างวาดภาพ แขนกลสามารถเคลื่อนที่ไปยังบริเวณใกล้ๆ ตำแหน่งนั้นได้ แต่ยังมีความผิดพลาดอยู่มาก โดยสังเกตได้จากการทดลองว่า ค่ามุมองศาของแขนกลมีความคลาดเคลื่อนจากค่ามุมองศาที่ส่งมาจากโปรแกรม Visual Basic เป็นอย่างมาก ทั้งนี้น่าจะเป็นผลมาจากการติดตั้งแขนกลไม่สัมพันธ์กับโปรแกรม แต่หลังจากปรับแต่งแขนกลให้สัมพันธ์กับโปรแกรมแล้ว แขนกลสามารถไปยังตำแหน่งต่างๆ ได้ดีกว่าเดิม ค่าความคลาดเคลื่อนของมุมแขนกลลดน้อยลง แต่ก็ยังมีค่ามุมบางจุดที่ยังคลาดเคลื่อนอยู่ จึงไม่สามารถทำให้วาดรูปได้ครบตามที่ต้องการ คือบางช่วงของภาพแขนจะเคลื่อนที่สูงจากระนาบวาดภาพ และบางช่วงแขนเคลื่อนที่ต่ำกว่าระนาบการวาดภาพ สังเกตได้จากค่าพิกัดที่ไกลจากตำแหน่งแขนกล ยิ่งไกลจากตำแหน่งแขนมากเท่าไร แขนก็จะกดลงมาขึ้น ซึ่งความผิดพลาดนี้น่าจะเกิดจากน้ำหนักของแขนที่ทำให้ค่าโมเมนต์มากขึ้น จึงทำให้แขนถูกกดลงที่ระยะพิกัดไกลๆ

5.2 ปัญหาที่พบและแนวทางแก้ไข

- จากการทดลองพบว่า แขนไม่เคลื่อนที่ไปยังตำแหน่งที่ถูกต้อง เนื่องจากส่วนของแขนกลไม่สัมพันธ์กับโปรแกรม ทำการแก้ไขโดยการปรับแต่งแขนกลใหม่ ให้สัมพันธ์กับโปรแกรม
- เมื่อแขนเคลื่อนที่ไประยะพิกัดที่ไกลๆ ปรากฏว่าแขนกดต่ำลง เนื่องจากน้ำหนักของแขนที่ทำให้เกิดค่าโมเมนต์ สามารถแก้ไขได้โดยการเปลี่ยนมอเตอร์ให้มีค่าทอร์กสูงๆเพื่อเพียงพอต่อแรงกดจากน้ำหนักของแขนที่ระยะไกล และปรับแต่งโปรแกรมเพื่อชดเชยค่าโมเมนต์ที่เกิดขึ้น
- หัวปากกาไม่สามารถตอบสนองได้ดีเท่าที่ควรเนื่องจากถูกกดด้วยน้ำหนักของแขน สามารถแก้ไขได้โดยทำให้หัวปากกามีความยืดหยุ่นพอสมควรเพื่อชดเชยแรงเสียดทานที่มีในหัวปากกา

5.3 ข้อเสนอแนะ

- การทดลองยังขาดการควบคุมที่ดีเนื่องจากเซอร์โวมอเตอร์เป็นการควบคุมแบบพีคอลโทรล ซึ่งเป็นระบบที่ยากต่อการควบคุมให้เสถียรดังนั้นควรที่จะสร้างระบบที่เป็นการป้อนกลับขึ้นมาเพื่อเป็นตัวช่วยให้ระบบทำงานได้อย่างมีประสิทธิภาพ

- การต่อและประกอบโครงสร้างของแขนกลต้องคำนึงถึงเรื่องน้ำหนักของแขนกลและค่าโมเมนต์ที่เกิดขึ้นให้ละเอียดถี่ถ้วนเพื่อเป็นการช่วยลดการชดเชยที่จะเกิดขึ้น

- การออกแบบหัวปากกาควรควรคำนึงถึงปลายปากกาไม่ให้เกิดแรงเสียดทานกับพื้นเพื่อความยืดหยุ่นในการวาดภาพ

เอกสารอ้างอิง

[1] บุญฤทธิ์ วรรณทรัพย์, ธีระชัย ไชยยศ. **"หุ่นยนต์จับวางชิ้นงาน."** ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาเครื่องกล คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ. 2543

[2] ประจิน พลังสันติกุล. **เรียนรู้และใช้งาน CCS C คอมไพล์เลอร์ เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์ PIC.** พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : อินโนเวตีฟ เอ็กเพอริเมนต์. 2547

[3] อภิชาต ภู่พลับ. **เขียนโปรแกรม Hardware interface ด้วย VB6.** พิมพ์ครั้งที่ 1. นนทบุรี : ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์. 2548

[4] มณฑล ใจกุศล และคณะ. **กลศาสตร์วิศวกรรม.** พิมพ์ครั้งที่ 1 : บริษัทวิทยพัฒน์ จำกัด. 2546

ภาคผนวก

ภาคผนวก ก
ค่ามุมองศาของมอเตอร์เมื่อป้อนสัญญาณพัลส์

ตารางที่ ก.1 ค่ามุมองศาของมอเตอร์ขับแขนกลส่วนฐาน

มุม (องศา)	สัญญาณพัลส์							
	การทดลองครั้งที่ 1		การทดลองครั้งที่ 2		การทดลองครั้งที่ 3		ค่าที่นำไปใช้	
	ไป	กลับ	ไป	กลับ	ไป	กลับ	ไป	กลับ
90	12000	12000	12000	12000	12000	12000	12000	12000
91	11940	11930	11940	11940	11940	11930	11940	11940
92	11880	11880	11890	11880	11880	11870	11880	11880
93	11800	11790	11790	11780	11800	11780	11800	11780
94	11700	11760	11690	11760	11700	11760	11700	11670
95	11620	11590	11620	11590	11610	11580	11620	11590
96	11550	11530	11550	11520	11560	11520	11550	11520
97	11450	11440	11460	11440	11460	11430	11460	11440
98	11370	11360	11360	11350	11370	11340	11370	11350
99	11260	11260	11260	11250	11270	11250	11260	11250
100	11180	11180	11180	11180	11180	11180	11180	11180
101	11120	11090	11110	11100	11120	11090	11120	11090
102	10910	11020	11910	11020	11900	11030	10910	11020
103	10970	10960	10970	10970	10960	10960	10970	10960
104	10890	10900	10900	10890	10890	10900	10890	10900
105	10840	10830	10850	10820	10840	10820	10840	10820
106	10770	10780	10780	10770	10770	10780	10770	10780
107	10660	10630	10670	10640	10660	10630	10660	10630
108	10590	10550	10580	10540	10590	10550	10590	10550
109	10490	10460	10480	10450	10490	10460	10490	10460

110	10400	10370	10400	10360	10400	10370	10400	10370
111	10330	10310	10330	10300	10320	10310	10330	10310
112	10270	10240	10270	10240	10260	10240	10270	10240
113	10200	10170	10190	10160	10190	10170	10190	10170
114	10110	10110	10110	10110	10100	10110	10110	10110
115	10030	10030	10020	10030	10030	10020	10030	10030
116	9950	9980	9930	9970	9920	9980	9930	9980
117	9840	9870	9830	9860	9830	9860	9830	9860
118	9760	9830	9750	9830	9760	9820	9760	9830
119	9700	9760	9700	9750	9700	9760	9700	9760
120	9650	9700	9650	9700	9650	9700	9650	9700
121	9560	9530	9550	9530	9560	9520	9560	9530
122	9480	9440	9480	9430	9480	9440	9480	9440
123	9390	9370	9390	9360	9380	9370	9390	9370
124	9320	9310	9320	9320	9310	9320	9320	9320
125	9120	9300	9110	9300	9130	9300	9120	9300
126	9060	9220	9050	9230	9060	9230	9060	9230
127	8990	9160	8980	9150	8980	9160	8980	9160
128	8890	9070	8890	9060	8900	9070	8890	9070
129	8830	9050	8830	9040	8830	9050	8830	9050
130	8760	9040	8760	9030	8650	9040	8760	9040
131	8690	8950	8690	8950	8680	8950	8690	8950
132	8640	8850	8630	8840	8650	8840	8640	8840
133	8620	8800	8620	8800	8630	8800	8620	8800
134	8600	8660	8590	8660	8600	8650	8600	8660
135	8550	8620	8550	8620	8560	8610	8550	8620
136	8470	8590	8470	8580	8470	8590	8470	8590
137	8420	8400	8410	8400	8420	8400	8420	8400
138	8340	8330	8320	8330	8340	8320	8340	8330

139	8130	8260	8120	8250	8140	8260	8130	8260
140	8020	8190	8020	8200	8020	8190	8020	8190
141	7950	8160	7930	8170	7940	8170	7940	8170
142	7880	8130	7880	8130	7880	8120	7880	8130
143	7840	8100	7830	8100	7840	8100	7840	8100
144	7770	8000	7770	8010	7770	8010	7770	8010
145	7750	7960	7750	7970	7750	7970	7750	7970
146	7730	7890	7730	7890	7730	7900	7730	7890
147	7710	7800	7710	7810	7700	7820	7710	7810
148	7630	7730	7620	7720	7630	7730	7630	7730
149	7550	7560	7560	7550	7550	7560	7550	7560
150	7500	7460	7500	7450	7500	7460	7500	7460
151	7420	7370	7410	7360	7420	7370	7420	7370
152	7160	7330	7150	7330	7160	7330	7160	7330
153	7100	7320	7090	7310	7080	7320	7090	7320
154	7030	7300	7030	7300	7030	7300	7030	7300
155	6970	7280	6960	7270	6970	7280	6970	7280
156	6900	7230	6910	7220	6920	7220	6910	7220
157	6800	7140	6810	7130	6810	7140	6810	7140
158	6790	7030	6800	7030	6800	7040	6800	7030
159	6780	6980	6790	6980	6790	6980	6790	6980
160	6770	6880	6780	6880	6780	6880	6780	6880
161	6640	6760	6660	6750	6650	6780	6640	6760
162	6400	6650	6420	6650	6400	6650	6400	6650
163	6320	6570	6340	6560	6300	6570	6320	6570

ตารางที่ ก.2 ค่ามุมองศาของมอเตอร์ขับแขนท่อนล่าง

มุม (องศา)	สัญญาณพัลส์							
	การทดลองครั้งที่ 1		การทดลองครั้งที่ 2		การทดลองครั้งที่ 3		ค่าที่นำไปใช้	
	ไป	กลับ	ไป	กลับ	ไป	กลับ	ไป	กลับ
90	12000	12000	12000	12000	12000	12000	12000	12000
89	12240	12080	12240	12060	12240	12080	12240	12080
88	12290	12130	12290	12130	12300	12120	12290	12130
87	12390	12190	12390	12180	12400	12190	12390	12190
86	12460	12270	12450	12260	12470	12270	12460	12270
85	12550	12360	12550	12350	12550	12360	12550	12360
84	12360	12470	12350	12480	12360	12470	12630	12470
83	12700	12530	12720	12530	12710	12520	12710	12530
82	12780	12620	12790	12610	12780	12610	12780	12610
81	12850	12680	12860	12680	12850	12670	12850	12680
80	12930	12770	12930	12760	12920	12770	12930	12770
79	13030	12850	13020	12860	13040	12860	13030	12860
78	13100	12930	13100	12950	13100	12940	13100	12940
77	13170	13000	13160	13020	13170	13010	13170	13010
76	13260	13100	13250	13100	13260	13100	13260	13100
75	13350	13150	13340	13160	13350	13140	13340	13150
74	13400	13190	13420	13200	13420	13190	13420	13190
73	13490	13240	13500	13250	13490	13240	13490	13240
72	13580	13380	13590	13390	13580	13380	13580	13380
71	13650	13450	13650	13470	13650	13480	13650	13470
70	13750	13530	13740	13530	13740	13550	13740	13530
69	13810	13640	13810	13650	13820	13650	13810	13650
68	13880	13720	13880	13730	13880	13730	13880	13730

67	13960	13830	13950	13820	13960	13830	13960	13830
66	14050	13900	14020	13900	14050	13900	14050	13900
65	14120	13970	14110	13960	14120	13970	14120	13970
64	14200	13990	14200	13990	14200	14000	14200	13990
63	14300	14060	14290	14050	14290	14060	14290	14060
62	14350	14150	14340	14150	14340	14160	14340	14150
61	14450	14220	14440	14220	14440	14230	14440	14220
60	14510	14360	14510	14350	14500	14360	14510	14360
59	14610	14410	14610	14400	14600	14410	14610	14410
58	14690	14530	14680	14520	14680	14530	14680	14530
57	14750	14580	14750	14580	14740	14580	14750	14580
56	14830	14680	14820	14680	14830	14680	14830	14680
55	14900	14720	14900	14710	14900	14730	14900	14720
54	14980	14800	14970	14800	14970	14800	14970	14800
53	15070	14830	15060	14820	15070	14830	15070	14830
52	15140	14890	15150	14900	15160	14890	15150	14890
51	15220	14950	15220	14960	15220	14960	15220	14960
50	15290	15020	15300	15020	15280	15030	15290	15020
49	15330	15090	15330	15100	15330	15090	15330	15090
48	15440	15170	15440	15180	15450	15170	15440	15170
47	15520	15250	15530	15250	15520	15260	15520	15250
46	15590	15330	15600	15320	15580	15330	15590	15330
45	15660	15430	15660	15440	15660	15430	15660	15430
44	15730	15500	15740	15510	15730	15510	15730	15510
43	15820	15570	15830	15580	15820	15570	15820	15570
42	15900	15640	15900	15650	15900	15650	15900	15650
41	15990	15720	16000	15720	15990	15730	15990	15720
40	16130	15790	16130	15790	16140	15800	16130	15790
39	16170	15810	16170	15810	16180	15820	16170	15810

38	16260	15900	16270	15900	16260	15900	16260	15900
37	16350	16000	16340	16000	16340	16000	16340	16000
36	16440	16100	16430	16090	16430	16090	16430	16090
35	16480	16160	16480	16150	16480	16180	16480	16160
34	16550	16250	16560	16250	16560	16270	16560	16250
33	16700	16330	16700	16330	16700	16340	16700	16330
32	16800	16400	16790	16400	16790	16400	16790	16400
31	16880	16500	16890	16490	16880	16490	16880	16490
30	16960	16550	16960	16540	16970	16540	16960	16540
29	17020	16620	17030	16620	17020	16630	17020	16620
28	17100	16660	17100	16670	17100	16670	17100	16670
27	17160	16650	17160	16650	17160	16650	17160	16650
26	17170	16750	17170	16760	17170	16760	17170	16760
25	17180	17190	17180	17180	17180	17180	17180	17180

ตารางที่ ก.3 ค่ามุมองศาของมอเตอร์ขับแขนท่อนบน

มุม (องศา)	สัญญาณพัลส์							
	การทดลองครั้งที่ 1		การทดลองครั้งที่ 2		การทดลองครั้งที่ 3		ค่าที่นำไปใช้	
	ไป	กลับ	ไป	กลับ	ไป	กลับ	ไป	กลับ
90	12000	12000	12000	12000	12000	12000	12000	12000
91	12080	12170	12080	12170	12090	12180	12080	12170
92	12210	12270	12200	12270	12210	12280	12210	12270
93	12330	12360	12330	12360	12330	12370	12330	12360
94	12460	12460	12470	12450	12470	12450	12470	12450
95	12600	12590	12580	12600	12610	12600	12600	12600
96	12850	12700	12860	12710	12860	12710	12860	12710
97	12940	12800	12940	12810	12950	12810	12940	12810
98	13030	12940	13030	12950	13030	12940	13030	12940
99	13100	13030	13090	13040	13090	13030	13090	13030
100	13190	13140	13180	13140	13180	13130	13180	13140
101	13280	13220	13270	13220	13270	13220	13270	13220
102	13380	13320	13370	13310	13370	13310	13370	13310
103	13440	13400	13440	13400	13450	13390	13440	13400
104	13540	13520	13560	13500	13570	13520	13560	13520
105	13600	13610	13610	13600	13610	13610	13610	13610
106	13690	13690	13690	13690	13700	13700	13690	13690
107	13770	13800	13770	13790	13790	13800	13770	13800
108	13850	13930	13840	13900	13840	13920	13840	13920
109	13920	14020	13910	14000	13910	14010	13910	14010
110	13990	14140	13980	14130	13980	14140	13980	14140
111	14050	14250	14050	14250	14040	14250	14050	14250
112	14150	14350	14170	14350	14180	15350	14170	14350

113	14250	14440	14260	14440	14260	14440	14260	14440
114	14350	14500	14360	14510	14360	14510	14360	14510
115	14440	14590	14440	14600	14440	14590	14440	14590
116	14510	14670	14510	14680	14500	14670	14510	14670
117	14590	14750	14590	14750	14590	14750	14590	14750
118	14680	14830	14680	14830	14680	14840	14680	14830
119	14770	14880	14760	14880	14760	14890	14760	14880
120	14830	15020	14820	15020	14820	15030	14830	15020
121	14890	15080	14880	15080	14880	15090	14880	15080
122	14960	15150	14960	15150	14950	15150	14960	15150
123	15060	15220	15060	15220	15050	15220	15060	15220
124	15130	15310	15130	15310	15120	15320	15130	15310
125	15190	15430	15190	15440	15180	15440	15190	15440
126	15280	15500	15290	15500	15280	15500	15280	15500
127	15350	15600	15350	15600	15350	15600	15350	15600
128	15430	15670	15430	15680	15430	15680	15430	15680
129	15530	15760	15520	15750	15520	15760	15520	15760
130	15580	15830	15580	15830	15570	15830	15580	15830
131	15670	15900	15680	15900	15670	15900	15670	15900
132	15750	15990	15760	15990	15750	15980	15750	15990
133	15820	16060	15810	16050	15820	16060	15820	16060
134	15900	16130	15900	16120	15900	16130	15900	16130
135	15970	16200	15970	16200	15960	16200	15970	16200
136	16050	16290	16050	16280	16050	16280	16050	16280
137	16150	16360	16140	16360	16140	16360	16140	16360
138	16200	16420	16200	16430	16200	16450	16200	16430
139	16270	16500	16280	16520	16260	16510	16270	16510
140	16340	16590	16350	16590	16340	16590	16340	16590

141	16420	16650	16430	16650	16420	16650	16420	16650
142	16500	16700	16490	16710	16480	16710	16490	16710
143	16570	16800	16560	16800	16560	16800	16560	16800
144	16660	16890	16660	16890	16660	16900	16660	16890
145	16730	16990	16710	16990	16720	17000	16720	16990
146	16790	17050	16780	17050	16790	17060	16790	17050
147	16870	17140	16870	17140	16870	17150	16870	17140
148	16950	17200	16950	17200	16960	17200	16950	17200
149	17020	17290	17020	17280	17030	17290	17020	17290
150	17090	17360	17090	17360	17100	17360	17090	17360
151	17170	17440	17180	17440	17170	17450	17170	17440
152	17250	17520	17250	17520	17250	17530	17250	17520
153	17320	17600	17320	17600	17330	17600	17320	17600
154	17390	17660	17390	17660	17400	17670	17390	17660
155	17470	17750	17470	17750	17480	17760	17470	17750
156	17550	17810	17550	17810	17550	17810	17550	17810
157	17640	17900	17640	17900	17650	17900	17640	17900
158	17720	17960	17700	17940	17730	17960	17720	17960
159	17790	18050	17780	18040	17790	18040	17790	18040
160	17880	18120	17870	18110	17870	18120	17870	18120
161	17960	18180	17950	18180	17950	18180	17950	18180
162	18020	18260	18000	18260	18010	18270	18010	18260
163	18100	18360	18090	18360	18090	18370	18090	18360
164	18170	18420	18160	18420	18160	18420	18160	18420
165	18240	18500	18240	18500	18250	18500	18240	18500
166	18300	18570	18310	18570	18310	18560	18310	18570
167	18440	18650	18450	18650	18440	18650	18440	18650
168	18550	18730	18550	18730	18550	18730	18550	18730
169	18630	18830	18630	18840	18640	18830	18630	18830

170	18650	18900	18650	18910	18650	18910	18650	18910
171	18720	18950	18720	18960	18720	18960	18720	18960
172	18800	19050	18810	19060	18810	19050	18810	19050
173	18890	19910	18900	19920	18890	19910	18890	19110
174	18960	19170	18970	19180	18960	19170	18960	19170
175	19030	19260	19030	19270	19030	19260	19030	19260
176	19170	19330	19170	19330	19170	19330	19170	19330
177	19200	19390	19210	19390	19210	19390	19210	19390
178	19250	19450	19250	19450	19250	19450	19250	19450
179	19320	19530	19330	19540	19320	19530	19320	19530
180	19400	19600	19390	19600	19390	19600	19390	19600

ตารางที่ ก.4 ค่ามุมองศาของมอเตอร์ขับปากกา

มุม (องศา)	สัญญาณพัลส์							
	การทดลองครั้งที่ 1		การทดลองครั้งที่ 2		การทดลองครั้งที่ 3		ค่าที่นำไปใช้	
	ไป	กลับ	ไป	กลับ	ไป	กลับ	ไป	กลับ
90	12000	12000	12000	12000	12000	12000	12000	12000
91	11890	11840	11890	11840	11900	11850	11890	11840
92	11770	11730	11770	11730	11780	11740	11770	11730
93	11680	11650	11680	11650	11690	11650	11680	11650
94	11580	11580	11590	11590	11590	11580	11590	11580
95	11520	11470	11510	11480	11520	11470	11520	11470
96	11450	11350	11450	11350	11450	11350	11450	11350
97	11330	11240	11340	11250	11340	11240	11340	11240
98	11220	11150	11220	11150	11220	11150	11220	11150
99	11150	11070	11150	11050	11150	11070	11150	11070
100	11020	10990	11010	10990	11020	10990	11020	10990
101	10960	10860	10950	10850	10960	18060	10960	10860
102	10840	10760	10840	10750	10850	10760	10840	10760
103	10730	10670	10730	10660	10740	10670	10730	10670
104	10660	10590	10660	10600	10670	10590	10660	10590
105	10580	10490	10570	10500	10590	10490	10580	10490
106	10470	10400	10470	10410	10480	10400	10470	10410
107	10380	10340	10390	10350	10370	13040	10380	10340
108	10300	10180	10300	10190	10300	10180	10300	10180
109	10190	10160	10200	10160	10190	10150	10190	10160
110	10090	10000	10100	10000	10090	10000	10090	10000
111	10010	9930	10010	9920	10010	9930	10010	9930
112	9880	9830	9880	9840	9880	9840	9880	9840

113	9780	9750	9780	9750	9770	9750	9780	9750
114	9720	9660	9720	9660	9710	9660	9720	9660
115	9630	9550	9620	9550	9630	9550	9630	9550
116	9540	9450	9540	9450	9530	9440	9540	9450
117	9430	9350	9440	9350	9430	9340	9430	9350
118	9350	9250	9340	9250	9350	9240	9350	9250
119	9280	9150	9270	9150	9280	9140	9280	9150
120	9200	9010	9200	9010	9200	9000	9200	9010
121	9080	8860	9070	8860	9080	8850	9080	8860
122	9000	8830	9000	8820	9000	8810	9000	8820
123	8890	8750	8900	8750	8890	8750	8890	8750
124	8780	8690	8790	8700	8780	8690	8780	8690
125	8670	8590	8690	8600	8680	8590	8680	8590
126	8620	8540	8630	8550	8620	8540	8620	8540
127	8540	8430	8540	8420	8520	8430	8540	8430
128	8450	8360	8430	8350	8420	8360	8430	8360
129	8270	8250	8250	8250	8270	8260	8270	8250
130	8180	8160	8170	8150	8180	8160	8180	8160
131	8110	8080	8110	8060	8110	8080	8110	8080
132	8050	8040	8050	8040	8050	8040	8050	8040
133	7970	7980	7980	7990	7980	7990	7970	7990
134	7910	7920	7900	7930	7920	7930	7910	7930
135	7810	7850	7800	7850	7820	7860	7810	7850
136	7730	7750	7720	7750	7730	7760	7730	7750
137	7660	7690	7660	7680	7660	7690	7660	7690
138	7580	7490	7580	7480	7570	7480	7580	7480
139	7500	7410	7500	7410	7500	7400	7500	7410
140	7370	7320	7370	7320	7360	7310	7370	7320
141	7260	7240	7270	7240	7260	7230	7260	7240

142	7200	7110	7200	7110	7190	7110	7200	7110
143	7110	7050	7120	7050	7110	7040	7110	7050
144	7030	6980	7030	6980	7020	6970	7030	6980
145	6940	6910	6940	6900	6930	6910	6940	6910
146	6830	6850	6850	6840	6850	6850	6850	6850
147	6770	6770	6780	6770	6780	6770	6780	6770
148	6700	6660	6710	6660	6710	6670	6710	6660
149	6640	6600	6640	6600	6640	6600	6640	6600
150	6540	6500	6540	6490	6550	6490	6540	6490
151	6450	6440	6440	6450	6450	6440	6450	6440
152	6360	6390	6350	6400	6360	6390	6360	6390
153	6280	6290	6270	6300	6280	6290	6280	6290
154	6190	6200	6190	6200	6180	6200	6190	6200
155	6120	6090	6120	6090	6110	6090	6120	6090
156	6030	6020	6030	6030	6020	6020	6030	6020
157	5940	5950	5950	5940	5940	5950	5940	5950
158	5850	5870	5860	5860	5850	5870	5850	5870
159	5780	5780	5780	5770	5770	5780	5780	5780
160	5700	5720	5690	5710	5690	5720	5690	5720
161	5630	5660	5620	5670	5630	5660	5630	5660
162	5550	5540	5530	5520	5520	5510	5550	5520
163	5460	5500	5450	5490	5450	5490	5450	5490
164	5390	5380	5370	5390	5390	5380	5390	5380
165	5280	5320	5270	5330	5280	5320	5280	5320
166	5220	5240	5210	5230	5220	5240	5220	5240
167	5120	5140	5120	5130	5110	5140	5120	5140
168	5050	5060	5050	5070	5050	5070	5050	5070
169	4970	4980	4980	4970	4970	4970	4970	4970
170	4880	4880	4880	4880	4880	4880	4880	4880

ภาคผนวก ข

โปรแกรมควบคุมการทำงานแขนกล

```
#include <p30f4011.h>
#include <uart.h>
#include <timer.h>
_FOSC(CSW_FSCM_ON & XT_PLL8);

_FWDT(WDT_OFF);                 // disable WDT
_FBORPOR(PBOR_OFF & MCLR_EN);   // disable PBOR, enable MCLR
_FGS(CODE_PROT_OFF);
#define L1          PORTDbits.RD2   //base
#define L2          PORTCbits.RC14 //big
#define L3          PORTEbits.RE0   //middle
#define L4          PORTBbits.RB1  //small
//#define led     PORTBbits.RB1
char aaa1=0;
char aaa2=0;
char aaa3=0;
char aaa4=0;
int ti=0;
int dat;
int deg=0;
long set1=160000;
long set2=160000;
long set3=160000;
long set4=160000;
int D1=12000;
int D2=12000;
int D3=12000;
int D4=12000;
```

```
int motorbase[106]={12000,11940,11880,11800,11700,11620,11550,11460,11370,11260,11180,
11090,11020,10960,10900,10820,10780,10630,10550,10460,10370,10310,10240,10170,10110,1
0030,9980,9860,9830,9760,9700,9560,9480,9390,9320,9120,9060,8980,8890,8830,8760,8690,86
60,8630,8600,8550,8470,8420,8340,8130,8020,7940,7880,7840,7770,7740,7710,7630,7550,7500
,7420,7160,7090,7030,6970,6910,6810,6800,6790,6780,6780,6780,6780,6780,6780,6780,6780,6
780,6780,6780,6780,6780,6780,6780,6780,6780,6780,6780,6780,6780,6780};

int motorbig[91]={16960,16960,16960,16960,16960,16960,16960,16960,16960,16960,16960,
16960,16960,16960,16960,16960,16960,16960,16960,16960,16960,16960,16960,16960,16960,1
6960,16960,16960,16960,16960,16880,16790,16700,16560,16480,16430,16340,16260,16170,16
130,15990,15900,15820,15730,15660,15590,15520,15440,15330,15290,15220,15150,15070,149
70,14900,14830,14750,14680,14610,14510,14440,14340,14290,14200,14120,14050,13960,1388
0,13810,13740,13650,13580,13490,13420,13340,13260,13170,13100,13030,12930,12850,12780,
12710,12630,12550,12460,12390,12290,12240,12000};

int motormid[111]={12000,12080,12210,12330,12470,12600,12860,12940,13030,13090,13180,
13270,13370,13440,13530,13610,13690,13770,13840,13910,13980,14050,14170,14260,14360,1
4440,14510,14590,14680,14760,14830,14880,14960,15060,15130,15190,15280,15350,15430,15
520,15580,15670,15750,15820,15900,15970,16050,16140,16200,16270,16340,16420,16490,165
60,16660,16720,16790,16870,16950,17020,17090,17170,17250,17320,17390,17470,17550,1764
0,17720,17790,17870,17950,18010,18090,18160,18240,18310,18440,18550,18630,18650,18720,
18810,18890,18960,19030,19170,19220,19250,19320,19390,19540,19550,19680,19760,19820,1
9970,20030,20120,20170,20260,20360,20450,20530,20600,20700,20770,20870,20940,21010,21
110};

int motorsmall[91]={12000,11890,11770,11680,11590,11520,11450,11340,11220,11150,11020,
10960,10840,10730,10660,10580,10470,10380,10300,10190,10090,10010,9880,9780,9720,9630,
9540,9430,9350,9280,9200,9080,9000,8890,8780,8680,8620,8540,8430,8270,8180,8110,8050,79
70,7910,7810,7730,7660,7580,7500,7370,7260,7200,7110,7030,6940,6850,6780,6710,6640,6540
,6450,6360,6280,6190,6120,6030,5940,5850,5780,5690,5630,5550,5450,5390,5280,5220,5120,5
050,4970,4880,4880,4880,4880,4880,4880,4880,4880,4880,4880,4880};
```

```
void _ISR _U2TXInterrupt(void)
{
  IFS1bits.U2TXIF=0;
}
void _ISR _U2RXInterrupt(void)
{
  IFS1bits.U2RXIF=0;
}
/*void _ISR _U1TXInterrupt(void)
{
  IFS0bits.U1TXIF=0;
}
void _ISR _U1RXInterrupt(void)
{
  IFS0bits.U1RXIF=0;// if(L5==0) {L5=1;} else {L5=0;}
  //L5=1;
}*/
void __attribute__((__interrupt__)) _T1Interrupt( void )
{  T1CONbits.TON = 0;
        IFS0bits.T1IF = 0;

   if(aaa1==0) {  L1=0;
            //PR1=D1;
             if(set1>D1) {PR1=D1; set1=set1-D1;} else {PR1=set1;aaa1=1;}

           T1CONbits.TON = 1;

           }
         else {L1=1;
            PR1=D1;
            aaa1=0;
            set1=160000-D1;
```

```
            T1CONbits.TON = 1;

            }

}
void __attribute__((__interrupt__)) _T2Interrupt( void )
{  T2CONbits.TON = 0;
        IFS0bits.T2IF = 0;

  if(aaa2==0) {  L2=0;
            //PR1=D1;
             if(set2>D2) {PR2=D2; set2=set2-D2;} else {PR2=set2;aaa2=1;}

            T2CONbits.TON = 1;

            }
          else {L2=1;
             PR2=D2;aaa2=0;
             set2=160000-D2;
             T2CONbits.TON = 1;

            }

}
void __attribute__((__interrupt__)) _T3Interrupt( void )
{
        T3CONbits.TON = 0;
        IFS0bits.T3IF = 0;

if(aaa3==0) {  L3=0;
            //PR1=D1;
```

```
            if(set3>D3) {PR3=D3; set3=set3-D3;} else {PR3=set3;aaa3=1;}

           T3CONbits.TON = 1;

           }
         else {L3=1;
            PR3=D3;aaa3=0;
            set3=160000-D3;
            T3CONbits.TON = 1;

            }

}
void __attribute__((__interrupt__)) _T4Interrupt( void )
{
        T4CONbits.TON = 0;
        IFS1bits.T4IF = 0;

  if(aaa4==0) {  L4=0;
            //PR1=D1;
            if(set4>D4) {PR4=D4; set4=set4-D4;} else {PR4=set4;aaa4=1;}

           T4CONbits.TON = 1;

           }
         else {L4=1;
            PR4=D4;aaa4=0;
            set4=160000-D4;
            T4CONbits.TON = 1;

            }
```

```
}
void Init_Timer1(void)
{  T1CON = 0;
        IFS0bits.T1IF = 0;
        IPC0bits.T1IP = 1;
        IEC0bits.T1IE = 1;

        PR1 = D1;
        T1CONbits.TCS = 0;

        T1CONbits.TON = 1;
}
void Init_Timer2( void )
{

   T2CON = 0;
        IFS0bits.T2IF = 0;
        IPC1bits.T2IP = 2;
        IEC0bits.T2IE = 1;

        PR2 = D2;
        T2CONbits.TCS = 0;

        T2CONbits.TON = 1;

}

void Init_Timer3(void)
{  T3CON = 0;
        IFS0bits.T3IF = 0;
```

```
        IPC1bits.T3IP = 3;
        IEC0bits.T3IE = 1;

        PR3 = D3;
        T3CONbits.TCS = 0;

        T3CONbits.TON = 1;
}
void Init_Timer4( void )
{

   T4CON = 0;
        IFS1bits.T4IF = 0;
        IPC5bits.T4IP = 4;
        IEC1bits.T4IE = 1;

        PR4 = D4;
        T4CONbits.TCS = 0;

        T4CONbits.TON = 1 ;

}
void delay(int t)
{
  int i,j;
   for(i=0;i<t;i++)
   for(j=0;j<250;j++);
}
```

```
void init()
{
 unsigned int baudvalue;
 unsigned int U2MODEvalue;
 unsigned int U2STAvalue;

 CloseUART2();
 ConfigIntUART2(UART_RX_INT_EN
&UART_RX_INT_PR6&UART_TX_INT_EN&UART_TX_INT_PR2 );
 baudvalue=51;//
  U2MODEvalue=UART_EN &
        UART_IDLE_CON &
        UART_RX_TX &
        UART_DIS_WAKE &
        UART_DIS_LOOPBACK &
        UART_DIS_ABAUD &
        UART_NO_PAR_8BIT &
        UART_1STOPBIT;
 U2STAvalue= UART_INT_TX_BUF_EMPTY &
        UART_TX_PIN_NORMAL &
        UART_TX_ENABLE &
        UART_INT_RX_3_4_FUL &
        UART_ADR_DETECT_DIS &
        UART_RX_OVERRUN_CLEAR ;
  OpenUART2(U2MODEvalue,U2STAvalue,baudvalue);
}

main()
{
 //char Txdata[]="\r\n TOM  \r\n";
 //int i,j;
```

```
//int way;
  TRISB=0;
  TRISC=0;
  TRISD=0;
  TRISE=0;
  //uart1_init();
  //uart2_init();
   init();
  Init_Timer1();
  Init_Timer2();
  Init_Timer3();
  Init_Timer4();

   //putsUART1((unsigned int *)Txdata);
  // while(BusyUART1());
    putsUART2((unsigned int *)"ready");
   while(BusyUART2());
 while(1)
{

  if(DataRdyUART2())
     {
     /* dat = ReadUART2();       //database
      if(deg==0)  {if(dat==90) {D1=D1+100;deg=1;} else {D1=D1-100;deg=1;}}
      else if(deg==1)   {if(dat==90) {D2=D2+100;deg=2;} else {D2=D2-100;deg=2;}}
      else if(deg==2){if(dat==90) {D3=D3+100;deg=3;} else {D3=D3-100;deg=3;}}
      else {if(dat==90) {D4=D4+100;deg=0;} else {D4=D4-100;deg=0;}}*/


      dat = ReadUART2();  //send
      if(deg==0)       {D1=motorbase[dat];deg=1;}
      else if(deg==1)  {D2=motorbig[dat];deg=2;}
```

```
    else if(deg==2)   {D3=motormid[dat];deg=3;}
    else              {D4=motorsmall[dat];deg=0;}


  }

}

}
```