เฮกซะพอด

นายชิดชัย จรัลรัตนกุล
นางสาวพีรพร จงจิระศิริ
นางสาวมาเรียม ศิริกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2458

HEXAPOD

MR. CHITCHAI JARUNRATANAKUL

MISS PEERAPORN CHONGCHIRASIRI

MISS MARIAM SIRIKUL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INDUSTRIAL ENGINEERING

FACULTY OF ENGINEERING

KING MONKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2005

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	เฮกซะพอด		
	HEXAPOD		
นักศึกษา	นายชิดชัย	จรัลรัตนกุล	รหัสประจำตัว 45010185
	นางสาวพีรพร	จงจิระศิริ	รหัสประจำตัว 45010555
	นางสาวมาเรียม	ศิริกุล	รหัสประจำตัว 45010614
หลักสูตร	วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอุตสาหการ		

อาจารย์ผู้ควบคุมปริญญานิพนธ์

(อาจารย์พจชัย โชติปรายนกุล)

หัวข้อปริญญานิพนธ์	เฮกซะพอด
นักศึกษา	นายชิตชัย จรัสรัตนกุล
	นางสาวพีรพร จงจิระศิริ
	นางสาวมาเรียม ศิริกุล
หลักสูตร	วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอุตสาหการ
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา	2548
อาจารย์ผู้ควบคุมปริญญานิพนธ์	อาจารย์พลชัย โชติปรายนกุล

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้มีวัตถุประสงค์เพื่อศึกษา ออกแบบ สร้างโปรแกรมเพื่อจำลองและควบคุมการเคลื่อนที่ พร้อมทั้งทำแบบจำลองของเฮกซะพอด (Hexapod) ซึ่งเป็นหุ่นยนต์แบบขนานประเภทหนึ่งที่สามารถนำไปประยุกต์ใช้ได้หลากหลาย เช่น การประยุกต์ใช้ในกระบวนการผลิตแบบอัตโนมัติ (Automatic production system) ในรูปแบบของเครื่องจักรควบคุมด้วยคอมพิวเตอร์ (Computer Numerical Control : CNC) ใช้ในการเชื่อม การกัด ผู้วิจัยได้ทำการออกแบบและทำแบบจำลองเฮกซะพอด (Hexapod) เบื้องต้น โดยมีลักษณะโครงสร้างเป็นรูปทรงพีระมิดขอดสามเหลี่ยม และทำการเขียนโปรแกรมส่วนแสดงผลเพื่อจำลองการเคลื่อนที่ ซึ่งตัวโปรแกรมแบ่งเป็น 2 ส่วนคือ ส่วนควบคุม และส่วนแสดงผล ซึ่งเราสามารถดูการเคลื่อนที่ก่อนที่จะสั่งให้แบบจำลองเฮกซะพอดเคลื่อนที่

Thesis Title	Hexapod
Student	Mr. Chitchai Jarunratanakul
	Miss Peeraporn Chongchirasiri
	Miss Mariam Sirikul
Degree	Bachelor of Engineering in Industrial Engineering
	King Mongkut's Insitute of Technology Ladkrabang
Academic Year	2005
Thesis Advisor	Mr. Pholchai Chotiprayanakul

ABSTRACT

The objectives of this thesis are to study, design and create the program for modeling and controlling the movement of the Hexapod model, which is a kind of parallel robots and can be applied for various usages, for instance, jointing and milling metals in Automatic Production System in the form of Computer Numerical Control ; CNC. The researchers designed and devised the primary Hexapod model in the triangular pyramid shape, and also composed the program for simulating the Hexapod's movement. The program is divided into two parts which are controlling section and simulating section. The movement of Hexapod model is displayed on a monitor before operating the model.

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่อง เฮกซะพอด สามารถสำเร็จลุล่วงไปได้ด้วยดี กลุ่มผู้วิจัยขอกราบขอบพระคุณบุคคลที่มีส่วนเกี่ยวข้องส่งผลให้ปริญญานิพนธ์ฉบับนี้เสร็จสมบูรณ์

อาจารย์พลชัย โชติปราชนกุล อาจารย์ที่ปรึกษาปริญญานิพนธ์ กลุ่มผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงสำหรับการให้โอกาสในการศึกษาปริญญานิพนธ์ฉบับนี้ รวมทั้งความรู้ คำแนะนำ ความช่วยเหลือและความเอาใจใส่ในทุกๆด้านตลอดเวลาที่ผ่านมา

รศ.พรศักดิ์ อรรถวานิช หัวหน้าภาควิชาวิศวกรรมอุตสาหการ กลุ่มผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงสำหรับคำแนะนำ กำลังใจในการทำงาน ความเอาใจใส่และทุกสิ่งทุกอย่างตลอดระยะเวลาของการศึกษาในระดับปริญญาตรี หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอุตสาหการ

ผศ.ดร.สรรพสิทธิ์ ลิ่มนรรัตน์ กลุ่มผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงสำหรับคำแนะนำ กำลังใจในการทำงาน ความเอาใจใส่ ความช่วยเหลือในทุกๆด้านและทุกสิ่งทุกอย่างตลอดระยะเวลาของการศึกษาในระดับปริญญาตรี หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอุตสาหการ

ดร.สิทธิพร พิมพ์สกุล กลุ่มผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงสำหรับความรู้ คำแนะนำ และความช่วยเหลือทุกๆด้านในการจัดทำปริญญานิพนธ์ฉบับนี้

อาจารย์เชาวลิต หามนตรี กลุ่มผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงสำหรับความรู้ คำแนะนำ กำลังใจในการทำงานและความช่วยเหลือทุกๆด้าน

ขอบคุณเพื่อนทุกคนสำหรับความช่วยเหลือ และกำลังใจ จนทำให้ปริญญานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี

นายชิตชัย จรัลรัตนกุล
นางสาวพีรพร จงจิระศิริ
นางสาวมาเรียม ศิริกุล

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการออกแบบโครงสร้างของเครื่องจักรใหม่ คือแบบโครงสร้างเฮกซะพอด (Hexapod) เพื่อสนองตอบความต้องการของผู้ใช้ที่ต้องการเครื่องจักรที่แข็งแรงมากขึ้น และมีความคล่องตัวสูง โดยมีลักษณะคล้ายหุ่นยนต์แบบขนานที่มีโครงสร้างทางการเคลื่อนที่ที่สามารถบังคับเครื่องมือให้เคลื่อนที่ได้ถึง 5 แนว แกน มีความเที่ยงตรงสูง มีค่าอัตราส่วนระหว่างความแข็งแรงต่อน้ำหนักสูงกว่าเครื่องจักรในรูปแบบเดิม ที่ต้องมีโครงฐานที่มีน้ำหนักมากเพื่อลดการสั่นสะเทือน และเพิ่มความแข็งแรง และใช้รองรับการกัดรอบสูง (High Speed Machining) และสามารถนำไปประยุกต์ได้หลากหลาย เช่น เครื่องกลึง เครื่องกัด เครื่องเชื่อม หรือแม้แต่ในวงการแพทย์ ก็มีการประยุกต์ใช้โครงสร้างประเภทนี้ในการผ่าตัดสมองด้วยเลเซอร์ เป็นต้น

1.2 วัตถุประสงค์

1. เพื่อศึกษาการเคลื่อนที่ของเฮกซะพอด
2. เพื่อศึกษาการสร้างแบบจำลองบนคอมพิวเตอร์ของเฮกซะพอด
3. เพื่อศึกษาและจัดสร้างโครงสร้างจำลองของเครื่องเฮกซะพอด

1.3 ขอบเขตของปริญญานิพนธ์

1. สร้างโปรแกรมจำลองการเคลื่อนที่ของเฮกซะพอดโดยใช้โปรแกรม Visual Basic 6.0
2. สร้างโปรแกรมควบคุมการเคลื่อนที่ของเฮกซะพอดโดยใช้โปรแกรม Visual Basic 6.0
3. จัดทำโครงสร้างแบบจำลองเครื่องเฮกซะพอด

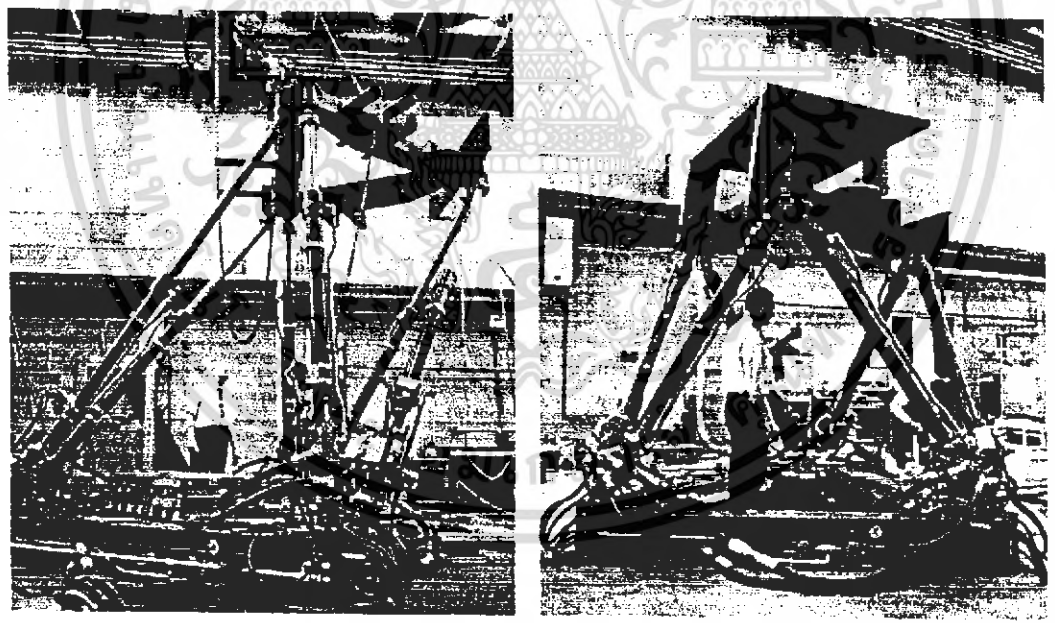
1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. โปรแกรมเพื่อใช้ในการควบคุมการเคลื่อนที่ของเครื่องจักร
2. แบบจำลองเฮกซะพอด (Hexapod) ที่สามารถนำมาประยุกต์ใช้จริงได้
3. เป็นแนวทางในการศึกษาและออกแบบทางด้านฮาร์ทแวร์และซอร์ฟแวร์ของระบบอัตโนมัติทางอุตสาหกรรม

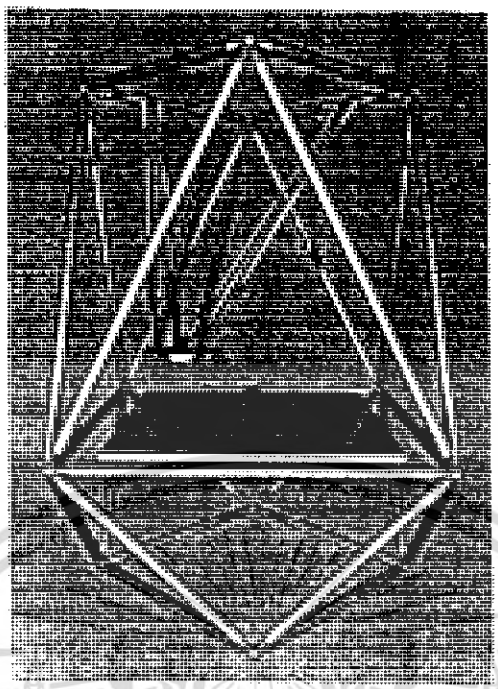
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีของเฮกซะพอด (Hexapod)

เฮกซะพอด (Hexapod) เป็น โครงสร้างรูปแบบขนาน (Parallel Architectures) ที่มีโครงสร้างรูปทรงพีระมิด มี 6 องศาอิสระ (6 degree of freedom) ที่สามารถบังคับเครื่องมือให้เคลื่อนที่ได้ถึง 5 แนวแกน โดยเริ่มแรกโครงสร้างถูกพัฒนาขึ้นมาเพื่อใช้เป็นเครื่องทดสอบยางรถยนต์ (Tire-testing Machine) และเครื่องจำลองการบิน (Flight Simulator) ต่อมาโครงสร้างแบบเดียวกันนี้แต่กลับคว่ำลงมาเป็นรูปร่างของเฮกซะพอดได้ถูกพัฒนาขึ้นมาเพื่อใช้ในงานต่างๆที่ต้องการโครงสร้างที่สามารถรับแรงได้มากและมีความคล่องตัวสูง โดยเฉพาะในการประยุกต์ใช้งานในกระบวนการผลิต เนื่องจากโครงสร้างนี้มีข้อดีเมื่อเปรียบเทียบกับโครงสร้างเครื่องจักรตามแนวแกนและหุ่นยนต์แบบอนุกรม (Serial Robot) ที่ใช้กันอยู่ คือ ค่าอัตราส่วนระหว่างความแข็งแรงต่อน้ำหนักสูง มีประสิทธิภาพในการทำงานที่ดีขึ้น และยังรองรับการกัดรอบสูง (High Speed Machining) อีกด้วย



รูปที่ 2.1 แสดงโครงสร้างของเครื่องจำลองการบิน (Flight Simulator)



รูปที่ 2.2 แสดงโครงสร้างของเฮกซะพอดที่คว่ำลงมา

2.1.1 หุ่นยนต์แบบขนาน (Parallel robot) กับหุ่นยนต์แบบอนุกรม (Serial robot)

เมื่อทำการเปรียบเทียบความแตกต่างกันในแของประสิทธิภาพของหุ่นยนต์แบบขนาน (Parallel robot) กับหุ่นยนต์แบบอนุกรม (Serial robot) มีข้อแตกต่างกันหลักๆ 4 ประการคือ พื้นที่ในการทำงาน (Workspace) ค่าอัตราส่วนระหว่างความแข็งแรงต่อน้ำหนัก (Payload-robot mass ratio) ความแม่นยำ (Accuracy) และลักษณะการเคลื่อนที่ของหุ่นยนต์ (Dynamic behavior)

1. พื้นที่ในการทำงาน (Workspace) พื้นที่ในการทำงานของหุ่นยนต์แบบขนานนั้นมีข้อเสียคือ มีพื้นที่ในการทำงานจำกัด ซึ่งพื้นที่ในการทำงานของหุ่นยนต์ชนิดนี้ถูกจำกัดด้วยตัวโครงสร้างที่สร้างขึ้นมา

2. อัตราส่วนระหว่างความแข็งแรงต่อน้ำหนัก (Payload-robot mass ratio) ในโครงสร้างแบบอนุกรม (Serial architecture) การที่จะควบคุมจุดปลาย (End-effecter) ของหุ่นยนต์เพื่อให้ทำงานใดๆ ได้นั้น จำเป็นต้องใช้กำลัง (Power) ในการขับเคลื่อนสูง เนื่องจากต้องใช้กำลังในการทำงานแล้ว ยังต้องใช้กำลังในการขับเคลื่อนตัวโครงสร้างทั้งหมดทั้งตัวแขน (Link) และข้อต่อ (Joint) ด้วย ซึ่งทำให้หุ่นยนต์แบบอนุกรมมีอัตราส่วนระหว่างความแข็งแรงต่อน้ำหนักที่ต่ำเมื่อเปรียบเทียบกับโครงสร้างแบบขนานแล้ว แรง (Load) ที่ให้กับโครงสร้างชนิดนี้จะใช้ในการทำงานโดยตรง ซึ่งเราสามารถทำให้แขน (Link) มีขนาดเล็กกว่าและมีอัตราส่วนระหว่างความแข็งแรงต่อน้ำหนักที่สูง

3. ความถูกต้อง (Accuracy) และการทำซ้ำ (Repeatability) ในหุ่นยนต์แบบอนุกรม (Serial robot) จะมีความผิดพลาดในการเคลื่อนที่สะสมกันมาในทุกๆแขน (Link) และข้อต่อ (Joint) ขึ้น ทำให้ความถูกต้อง (Accuracy) และการทำซ้ำ (Repeatability) น้อย แต่หุ่นยนต์แบบขนาน (Parallel robot) จะไม่ข้อเสียเช่นนี้ขึ้น

4. ลักษณะการเคลื่อนที่ของหุ่นยนต์ (Dynamic behavior) เนื่องจากโครงสร้างแบบขนานนี้มีอัตราส่วนระหว่างความแข็งแรงต่อน้ำหนักที่สูง และมีผลกระทบจากตัวข้อต่อ (Joint) น้อย จึงทำให้หุ่นยนต์แบบขนานมีรูปแบบการเคลื่อนที่ที่ดีกว่า

2.1.2 การประยุกต์ใช้งานเฮกซะพอด

โครงสร้างของเฮกซะพอดนี้ เนื่องจากเป็นโครงสร้างที่สามารถเคลื่อนที่ได้สูงถึง 5 แนวแกน และมีข้อดีหลากหลาย เราจึงสามารถนำโครงสร้างนี้ไปประยุกต์ใช้งานได้ทั้งในแบบคว่ำลง (downwards) และหงายขึ้น upwards) ได้แก่

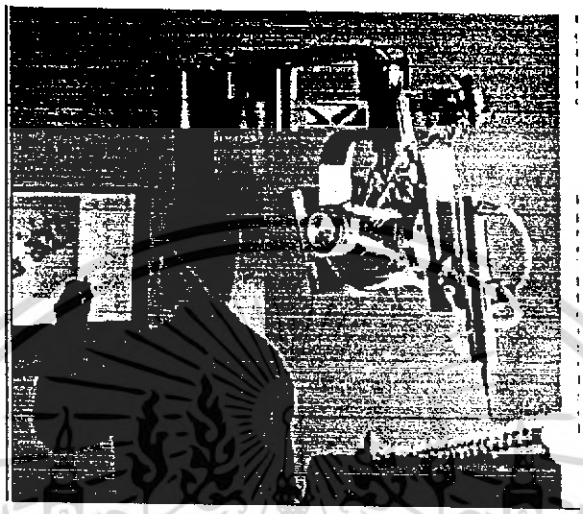
1. การประยุกต์โครงสร้างนี้เพื่องานแมชชินนิ่ง (Machining)

เนื่องจากตัวโครงสร้างเฮกซะพอดนี้จะมีจุดปลาย (End-effecter) ที่สามารถปรับเปลี่ยนให้ใช้ในงานแมชีนนิ่ง (Machining) ต่างๆได้ โดยการปรับเปลี่ยนหัวทูล (Tool) และปรับเปลี่ยนส่วนของโปรแกรมเพื่อให้เหมาะสมกับการใช้งานนั้นๆ ซึ่งจะได้เครื่องจักรที่รองรับการกัดรอบสูง (High Speed Machining) สามารถบังคับเครื่องมือให้เคลื่อนที่ได้ถึง 5 แกน เช่น เครื่องกัด เครื่องเจาะ เครื่องกลึง และเครื่องเชื่อม เป็นต้น

รูปที่ 2.3 แสดงการประยุกต์ใช้งานโครงสร้างเฮกซะพอดในงานกัด

2. การประยุกต์ใช้โครงสร้างนี้ในทางการแพทย์

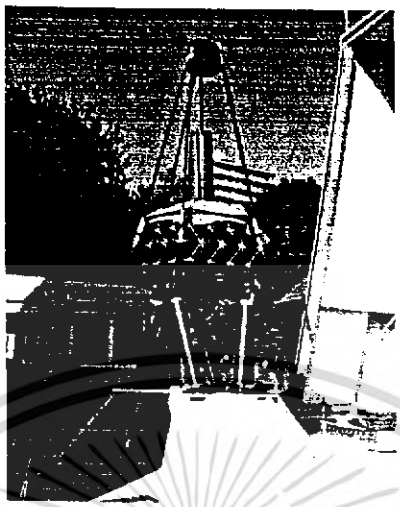
เฮกซะพอดถูกนำมาประยุกต์ใช้ในวงการแพทย์ ในการผ่าตัด ที่ต้องการความละเอียดและแม่นยำในการผ่าตัดสูง เช่น การผ่าตัดสมอง ผ่าตัดกระดูกไขสันหลัง และการรักษาทางจักษุแพทย์ โดยปรับการใช้โครงสร้างให้หงายขึ้น (upwards) และคิดตั้งอุปกรณ์เครื่องมือทางการแพทย์ที่ใช้ในการผ่าตัดลงไป



รูปที่ 2.4 แสดงการประยุกต์ใช้โครงสร้างเฮกซะพอดในการผ่าตัดกระดูกสันหลัง

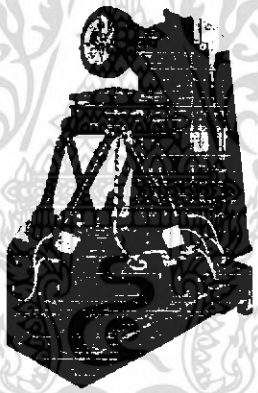
รูปที่ 2.5 แสดงการประยุกต์ใช้โครงสร้างเฮกซะพอดในการผ่าตัดสมอง

3. การประยุกต์ใช้งานในกล้องโทรทรรศน์



รูปที่ 2.6 แสดงการประยุกต์ใช้โครงสร้างเฮกซะพอดในกล้องโทรทรรศน์

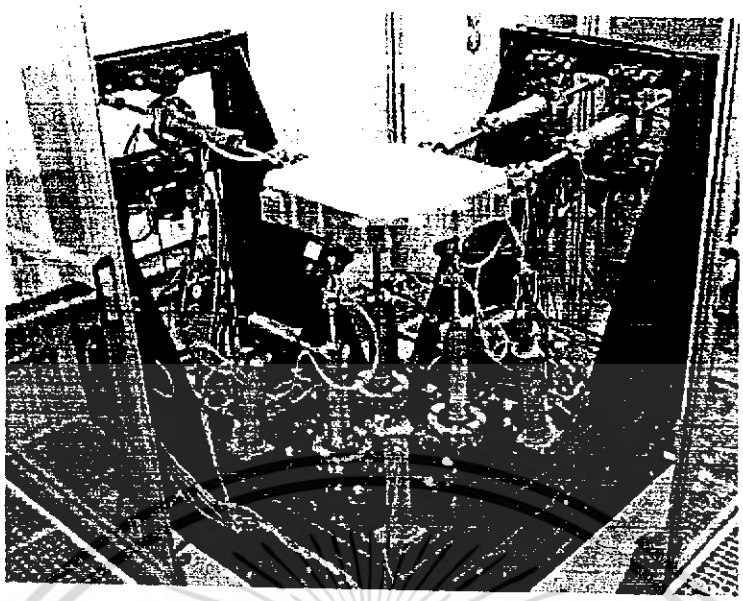
4. การประยุกต์ใช้งานในวงการรถยนต์



รูปที่ 2.7 แสดงเครื่องทดสอบยางรถยนต์

5. การประยุกต์ใช้งานในรูปแบบของเครื่องจำลองต่างๆ (Simulator)

เนื่องจากโครงสร้างเฮกซะพอดนี้มีข้อดีที่สามารถควบคุมการเคลื่อนที่ได้ถึง 5 แนวแกน จึงได้มีการนำโครงสร้างนี้มาประยุกต์ใช้ในการทำเครื่องทดสอบ เช่น เครื่องจำลองการบิน (Flight Simulator) เป็นต้น



รูปที่ 2.8 แสดงเครื่องการนำโครงสร้างเฮกซะพอดไปใช้ในการจำลองต่างๆ

2.2 การวิเคราะห์ตำแหน่งของการเคลื่อนที่ของหุ่นยนต์

ในการควบคุมการเคลื่อนที่ของหุ่นยนต์นั้นไม่ว่าจะเป็นหุ่นยนต์แบบขนาน (Parallel robot) หรือหุ่นยนต์แบบอนุกรม (Serial robot) เราสามารถควบคุมได้ 2 รูปแบบคือ การควบคุมการเคลื่อนที่แบบไปข้างหน้า (Forward Kinematics) และการควบคุมการเคลื่อนที่แบบย้อนกลับ (Inverse Kinematics)

การควบคุมการเคลื่อนที่แบบไปข้างหน้า (Forward Kinematics) คือ การควบคุมการเคลื่อนที่ของหุ่นยนต์ไปยังจุดปลายใดๆ โดยที่เรารู้ค่าตัวแปรต่างๆที่มีผลกับการเคลื่อนที่นั้นๆ

การควบคุมการเคลื่อนที่แบบย้อนกลับ (Inverse Kinematics) คือ การควบคุมการเคลื่อนที่ของหุ่นยนต์โดยเราทราบตำแหน่งจุดปลายนั้นแล้ว แต่เราต้องทำการคำนวณหาค่าตัวแปรต่างๆที่มีผลกับการเคลื่อนที่นั้นๆออกมา

โดยการวิเคราะห์การเคลื่อนที่ของหุ่นยนต์นั้นจะใช้เมตริก (Matrix) ในการคิดคำนวณ

2.2.1 ทรานสเลชั่นเมตริก (Translation Matrix)

คือเมตริกเอกลักษณ์ (Identity Matrix) ที่ได้มาจากเวกเตอร์ 1 หน่วย เป็นการแสดงการย้ายตำแหน่งของจุดปลาย (End-effecter) จากตำแหน่งอ้างอิงโดยที่ทิศทางไม่เปลี่ยนแปลง โดยเขียนทรานสเลชั่นเมตริก (Translation Matrix) ในรูปทั่วไปได้ดังนี้

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad (2.1)$$

โดยที่ $t_x\ t_y\ t_z$ คือ ระยะทางที่ถูกย้ายไปจากจุดอ้างอิง

x y z คือ ตำแหน่งจุดปลายแท่งก่อนที่จะถูกย้าย

x' y 'z' คือ ตำแหน่งจุดปลายใหม่ที่ถูกย้ายไป

2.2.2 โรเตชั่นเมตริก (Rotation Matrix)

คือเมตริกขนาด 4x4 ที่แสดงถึงทิศทางการหมุนรอบแกน x y z ใดๆ ไปตามองศาที่กำหนดไว้ ซึ่งโรเตชั่นเมตริก (Rotation Matrix) รวบแกนต่างๆเขียนในรูปทั่วไปได้ดังนี้

โรเตชั่นเมตริกรอบแกน x (Rotation Matrix around x axis)

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta & -\sin\Theta & 0 \\ 0 & \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad (2.2)$$

โดยที่ θ คือ มุมที่จุดปลาย (End-effecter) หมุนรอบแกน x

x y z คือ ตำแหน่งจุดปลายแท่งก่อนที่จะทำการหมุนรอบแกน x

x' y 'z' คือ ตำแหน่งจุดปลายใหม่ที่ถูกหมุนรอบแกน x ไปเป็นมุมθ

โรเตชั่นเมตริกรอบแกน y (Rotation Matrix around y axis)

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad (2.3)$$

โดยที่ θ คือ มุมที่จุดปลาย (End-effecter) หมุนรอบแกน y

x y z คือ ตำแหน่งจุดปลายแท่งก่อนที่จะทำการหมุนรอบแกน y

x' y 'z' คือ ตำแหน่งจุดปลายใหม่ที่ถูกหมุนรอบแกน y ไปเป็นมุมθ

โรเตชั่นเมตริกรอบแกน z (Rotation Matrix around z axis)

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad (2.4)$$

โดยที่ θ คือ มุมที่จุดปลาย (End-effecter) หมุนรอบแกน z

x y z คือ ตำแหน่งจุดปลายเก่าก่อนที่จะทำการหมุนรอบแกน z

x' y 'z' คือ ตำแหน่งจุดปลายใหม่ที่ถูกหมุนรอบแกน z ไปเป็นมุมθ

2.2.3 สเกลเมตริก (Scale Matrix)

สเกลเมตริก (Scale Matrix) คือเมตริกที่แสดงถึงการย่อหรือขยายขนาดขภาพรวมของการเคลื่อนที่ของหุ่นยนต์เมื่อแสดงผลออกมาบังหน้าจอคอมพิวเตอร์ ว่าต้องการขนาดใหญ่หรือเล็กเท่าไร แสดงในรูปทั่วไปได้ดังนี้

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad (2.5)$$

โดยที่ $s_x s_y s_z$ คือ ขนาด (Scale) ที่ต้องการย่อหรือขยายภาพที่แสดง

x y z คือ ตำแหน่งจุดปลายเก่าก่อนที่จะถูกย่อหรือขยาย

x' y 'z' คือ ตำแหน่งจุดปลายใหม่หลังจากที่ทำการย่อหรือขยายภาพเรียบร้อยแล้ว

2.3 สเต็ปมอเตอร์

สเต็ปมอเตอร์เป็นอุปกรณ์เชิงกลทางไฟฟ้าที่มีอินพุทเป็นกลุ่มของการหมุนเลขฐานสอง การเคลื่อนที่เป็นลักษณะของการเคลื่อนที่เชิงมุมหรือหมุนไปตามสเต็ป (แต่ละสเต็ปจะอยู่ในช่วง 0.1 – 90 องศา ขึ้นอยู่กับโครงสร้างของสเต็ปมอเตอร์) ตามสัญญาณพัลส์ที่ป้อนให้กับขั้วสเตเตอร์จึงเกิดแรงผลักต่อโรเตอร์ ซึ่งจะส่งผลให้โรเตอร์หมุนไปจากตำแหน่งเดิม ลักษณะของสเต็ปมอเตอร์จะมีขั้วของสเตเตอร์อยู่หลายขดซึ่งเรียกว่า "เฟส" ดังนั้นเมื่อป้อนสัญญาณพัลส์ในลักษณะของลำดับของเลขฐานสอง โดยผ่านวงจรขับ จะทำให้โรเตอร์หมุนได้ อย่างต่อเนื่อง

2.3.1 คุณประโยชน์ของสเต็ปมอเตอร์

เมื่อนำสเต็ปมอเตอร์ไปเปรียบเทียบกับมอเตอร์กระแสตรงแล้วนั้น จะเห็นได้ว่าการใช้สเต็ปมอเตอร์ในระบบการควบคุมตำแหน่งมีข้อได้เปรียบอยู่หลายประการ ดังนี้

1. เป็นลักษณะของการควบคุมแบบไม่ต้องการการป้อนกลับ ไม่ว่าจะเป็นตำแหน่งหรือความเร็ว นอกเสียจากในบางกรณีที่ความสัมพันธ์ของการเคลื่อนที่ระหว่างแกนหมุนของมอเตอร์กับตำแหน่งของโหลดไม่ตรงกัน อันเกิดจากความผิดพลาดของเฟืองเกียร์ หรือในกรณีที่โหลดบางอย่างทำให้เกิดความผิดพลาดของสเต็ปขึ้น ก็จำเป็นที่จะต้องมีการใช้สัญญาณป้อนกลับเพื่อควบคุมตำแหน่งและความเร็วให้ถูกต้อง

2. ความผิดพลาดเกี่ยวกับตำแหน่งแทบจะไม่มีเลย เนื่องจากการเคลื่อนที่ของสเต็ปมอเตอร์นั้นจะเคลื่อนที่เป็นสเต็ปด้วยจำนวนองศาที่แน่นอน

3. สเต็ปมอเตอร์จะถูกนำมาใช้กับเครื่องมือที่ต้องการความละเอียดแม่นยำ และใช้อยู่ในเครื่องมือประเภทดิจิตอล เช่น เครื่องวาดรูป

4. สเต็ปมอเตอร์ไม่จำเป็นต้องใช้วงจรแปลงสัญญาณเพื่อติดต่อกับคอมพิวเตอร์

5. สเต็ปมอเตอร์เป็นอุปกรณ์ที่เปลี่ยนสัญญาณดิจิตอลไปเป็นการเคลื่อนที่ทางกล ดังนั้นการติดต่อกับอุปกรณ์ดิจิตอลก็จะเป็นไปโดยง่าย และวงจรขยายกำลังจากสัญญาณดิจิตอลที่ใช้ ก็มีราคาถูกกว่าวงจรขยายกำลังเชิงเส้น

6. การออกแบบวงจรควบคุมสเต็ปมอเตอร์สามารถทำได้ง่ายกว่าวงจรควบคุมแบบเซอร์โวมอเตอร์ และยังสามารถออกแบบวงจรในสเต็ปมอเตอร์ให้สามารถทำงานหรือหยุดได้แบบทันทีทันใดได้

2.3.2 คุณลักษณะของสเต็ปมอเตอร์

ความถูกต้องเที่ยงตรงของมุมสเต็ปมอเตอร์ขณะไม่มีโหลดจะถูกระบุไว้สำหรับมอเตอร์ทุกชนิด เช่น มอเตอร์ที่มี 7.5 องศา ขณะที่เคลื่อนที่ไปหนึ่งสเต็ป เป็นต้น

มอเตอร์ที่มีจำนวนสเต็ปต่อรอบเท่ากับ 4 จะมีความผิดพลาดเท่ากับศูนย์เมื่อหมุนครบ 1 รอบ เพราะขณะที่หมุนมา ณ ตำแหน่งเดิมขณะเริ่มต้น ขั้วแม่เหล็กและทิศทางของสนามแม่เหล็ก (Flux) วงเดิม ด้วยเหตุนี้การเปลี่ยนตำแหน่งของสเต็ปมอเตอร์ที่ต้องการความถูกต้องสูงๆ จะต้องแบ่งจำนวนสเต็ปต่อรอบเป็นจำนวนเท่าของ 4 สเต็ป เพื่อลดการสะสมของค่าผิดพลาด (Step Angle Error) ซึ่งเป็นรูปแบบการทำงาน 4 สเต็ป

ตารางที่ 2.1 แสดงตัวอย่างของมุมสเต็ป

มุมสเต็ป (องศา)	จำนวนสเต็ปต่อรอบ
0.9	400
1.8	200
3.6	100
3.75	93
7.5	48
15	24

2.3.3 แรงบิด (Torque)

การทำงานของสเต็ปมอเตอร์มีแรงบิดเกี่ยวข้องอยู่ 3 ชนิดคือ

1. โฮลดิ้งทอร์ก (Holding Torque) คือ แรงบิดที่ทำให้สเต็ปมอเตอร์หมุนไป 2 สเต็ปขณะหยุดนิ่ง ถ้าแรงบิดที่ให้สเต็ปมอเตอร์มีมากกว่าโฮลดิ้งทอร์ก จะทำให้สเต็ปมอเตอร์สูญเสียการหมุนแบบสเต็ป กลายเป็นการหมุนแบบต่อเนื่อง โดยปกติขณะการทำงานของมอเตอร์ Torque จะน้อยกว่าระดับ Holding

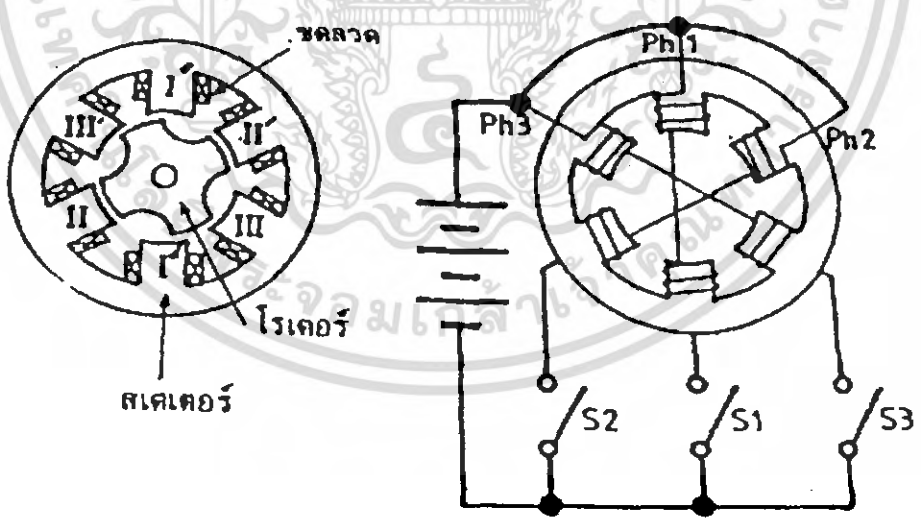
2. คีเทนท์ทอร์ก (Detent Torque) เป็นสเต็ปมอเตอร์แบบ Hybrid และแบบแม่เหล็กถาวร จะมีส่วนประกอบของโรเตอร์เป็นแม่เหล็ก ซึ่งจะสร้างแรงบิดมาเบรกการหมุนของมอเตอร์อย่างสม่ำเสมอในขณะที่ไม่มีการป้อนกระแสเข้าขดสเตเตอร์ แรงบิดดังกล่าวนี้เรียกว่า Detent Torque

3. ไดนามิคทอร์ก (Dynamic or Working Torque) คือ แรงบิดขณะทำงาน ซึ่งอาจเกิดการเปลี่ยนแปลงได้เนื่องมาจากการปรับเปลี่ยนอัตราเร็วของมอเตอร์ โดยปกติการเปลี่ยนอัตราเร็วของมอเตอร์จะอยู่ในย่านระหว่างเส้นโค้งพูลอิน (Pull-in Curve) และเส้นโค้งพูลเอาท์ (Pull-out Curve) เพราะถ้าปรับอัตราเร็ว ณ จุดนอกโค้งพูลเอาท์ มอเตอร์จะสูญเสียการหมุนแบบสเต็ปได้ เกิดเป็นการหมุนแบบต่อเนื่องนั่นเอง

2.3.4 ชนิดของสเต็ปมอเตอร์และการทำงาน

สเต็ปมอเตอร์แบ่งออกได้หลายชนิดด้วยกัน เช่น แบบแวริเอเบิลรีลักแทนซ์ (Variable Reluctance), แบบไฮบริด (Hybrid), แบบแม่เหล็กถาวร (Permanent Magnet) เป็นต้น แต่ที่ใช้งานกันบ่อยๆ คือแบบแวริเอเบิลรีลักแทนซ์และแบบแม่เหล็กถาวร ดังนั้นจะอธิบายหลักการทำงานของมอเดอร์ 2 ชนิดนี้เท่านั้น

แบบแวริเอเบิลรีลักแทนซ์ (Variable Reluctance) หรือเรียกสั้นๆ ว่า VR มอเตอร์ จะเป็นพื้นฐานที่สำคัญในการทำงานของสเต็ปมอเตอร์ ซึ่งจะช่วยให้เข้าใจการทำงานของสเต็ปมอเตอร์ชนิดอื่นๆได้ง่ายยิ่งขึ้น โดยสามารถพิจารณาส่วนประกอบของมอเตอร์ชนิดนี้ได้ดังรูปที่ 2.9 ซึ่งเป็นภาพหน้าตัดของมอเตอร์



รูปที่ 2.9 แสดงภาพหน้าตัดและการพันขดลวดของสเต็ปมอเตอร์ 3 เฟส

จากรูปแสดงถึงส่วนประกอบต่างๆของมอเตอร์ซึ่งเป็นมอเตอร์แบบ 3 เฟส และนอกจากนี้ยังแสดงถึงการพันขดลวดของมอเตอร์ด้วย โดนมอเตอร์นี้จะมีขั้วเหนือและขั้วได้อยู่ตรงข้ามกัน 3 คู่ โดยจะพันขดลวดแบบอนุกรมกันในแต่ละชุด

การทำงานจะเริ่มจากการกระตุ้นเฟส 1 ก่อน (S1 "ON") ซึ่งจะทำให้เส้นแรงแม่เหล็กเกิดขึ้น ตัวโรเตอร์จะพยายามวางตำแหน่งของตัวเองให้อยู่ในทิศทางที่ทำให้เกิดค่าความต้านทานแม่เหล็กน้อยที่สุด ดังรูปที่ 2.2

ในขณะเริ่มกระตุ้นเฟสที่ 2 (S1 "OFF", S2 "ON") ดังรูปที่ 2.3 เส้นแรงแม่เหล็กจะไม่อยู่ในแนวทางที่เดินสะดวก จึงทำให้ค่าความต้านทานแม่เหล็กมีค่าสูง ตัวโรเตอร์จะพยายามปรับตัวเองให้ค่าความต้านทานแม่เหล็กมีค่าน้อยที่สุด ด้วยการหมุนทิศทางทวนเข็มนาฬิกา ซึ่งแรงบิดที่ใช้ในการหมุนเกิดจากแรงของเส้นแรงแม่เหล็ก แล้วจะไปหยุดที่ตำแหน่งมีค่าความต้านทานแม่เหล็กน้อยที่สุด นั่นคือจะหมุนไป 1 สเต็ป หรือ 30 องศานั่นเอง ความสัมพันธ์ระหว่างจำนวนสเต็ปของการหมุนโรเตอร์ไป 1 รอบ แสดงได้ดังสมการนี้

$$S = \frac{360}{\theta_s} \qquad (2.6)$$

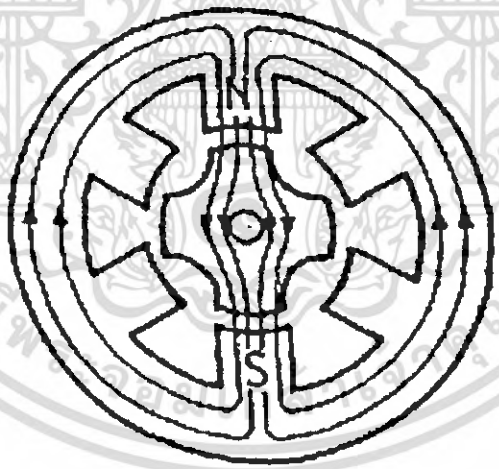
$$= mN_r$$

โดย S : จำนวนสเต็ปของการหมุนโรเตอร์ 1 รอบ

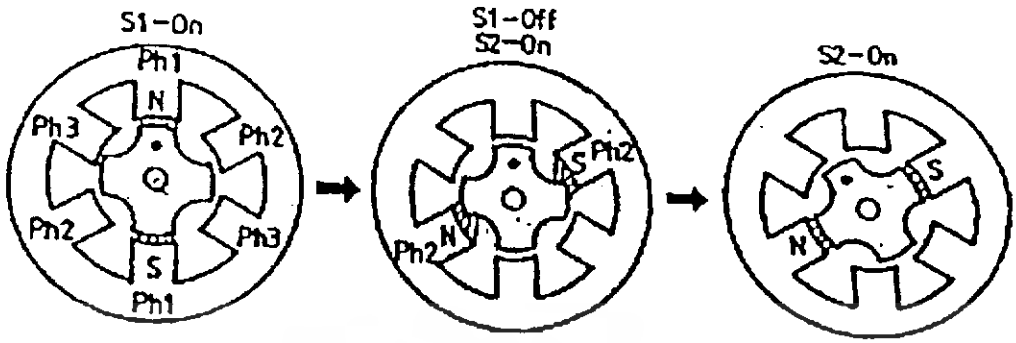
m : จำนวนของสเตเตอร์

θ_s : มุมที่เปลี่ยนไป 1 สเต็ป

N_r : จำนวนฟันของโรเตอร์

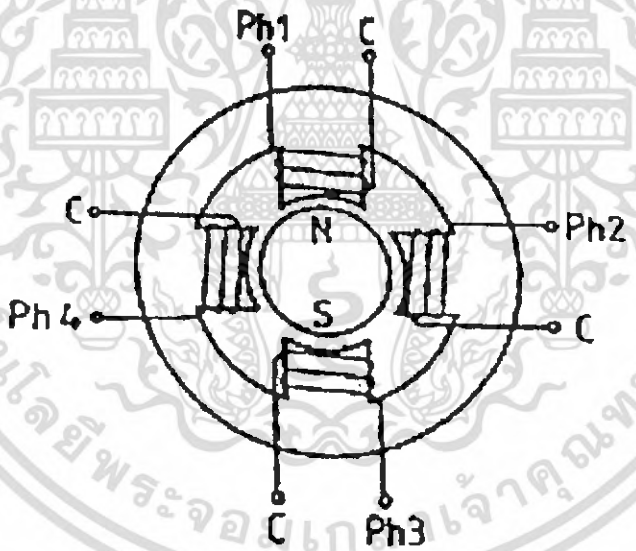


รูปที่ 2.10 แสดงเส้นแรงแม่เหล็กขณะกระตุ้นเฟสที่ 1



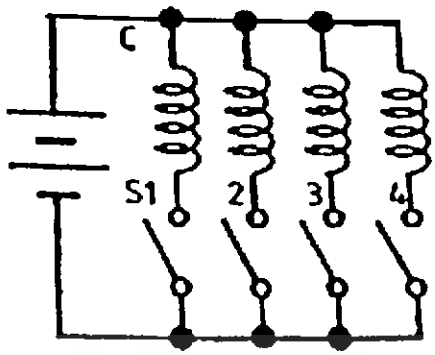
รูปที่ 2.11 แสดงขั้นตอนการหมุนเมื่อมีการกระตุ้นเฟสจากเฟส 1 ไปยังเฟส 2

สำหรับสเต็ปมอเตอร์ชนิดแม่เหล็กถาวร (Permanent Magnet) หรือเรียกสั้นๆว่า PM มอเตอร์ จะมีข้อแตกต่างที่สำคัญจาก VR มอเตอร์ คือ โรเตอร์จะเป็นแม่เหล็กถาวร จึงทำให้การพันขดลวดที่สเตเตอร์ต้องแตกต่างไปด้วย ดังแสดงในรูปที่ 2.12



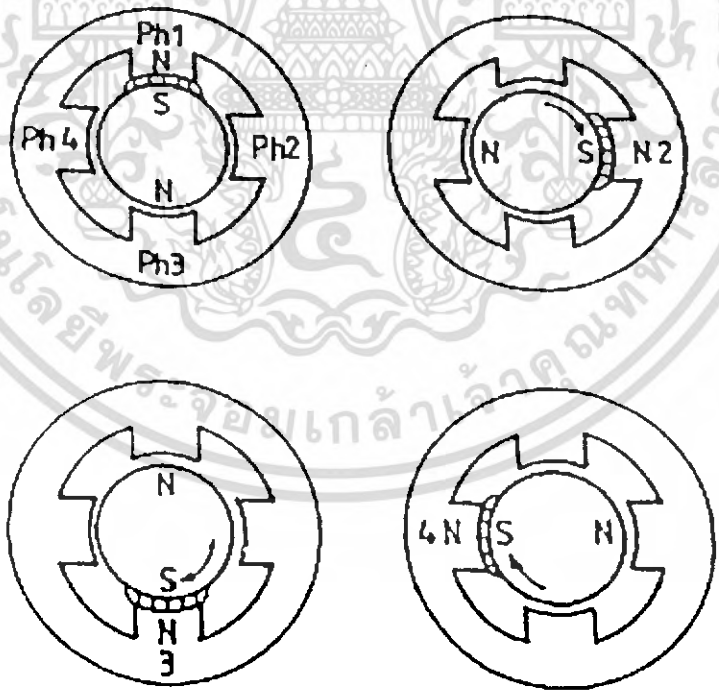
รูปที่ 2.12 แสดงภาพหน้าตัดของ PM มอเตอร์ แบบ 4 เฟส

จะเห็นได้ว่าสเตเตอร์ในแต่ละขั้วคือ 1 เฟส ดังนั้นจากรูปจึงมีทั้งหมด 4 เฟสด้วยกัน สำหรับการต่อวงจรกระตุ้นเฟสมอเตอร์อย่างง่ายแสดงได้ดังรูปที่ 2.13



รูปที่ 2.13 แสดงวงจรกระตุ้นเฟสขั้นพื้นฐาน สำหรับ PM มอเตอร์ 4 เฟส

จะเห็นได้ว่าปลายขดลวด (C) ของทุกเฟสจะต้องต่อร่วมกันของขั้วบวกของแหล่งจ่ายไฟ ดังนั้นเมื่อเกิดการกระตุ้นที่เฟสใดแล้ว ขั้วสเตเตอร์ที่เฟสนั้นก็จะกลายเป็นขั้วเหนือ ดูได้จากรูปที่ 2.7 จะเป็นการแสดงตำแหน่งของโรเตอร์ในแต่ละสเต็ป หลังจากถูกกระตุ้นที่เฟส 1-2-3-4 ตามลำดับ และจะหมุนไปในทิศทางตามเข็มนาฬิกา ทุก 90 องศาต่อสเต็ป ถ้าต้องการจะให้มุมองศาต่อสเต็ปมีค่าลดลงหรือมีความละเอียดในตำแหน่งมากขึ้น ก็จะต้องเพิ่มจำนวนเฟสของสเตเตอร์และจำนวนขั้วเหล็กของโรเตอร์ให้มากขึ้น ข้อเสียของแม่เหล็กแบบถาวร คือ มีราคาแพงมาก และความหนาแน่นของเส้นแรงแม่เหล็กจะถูกจำกัดโดยเส้นแรงแม่เหล็กภายในของแม่เหล็กถาวร ทำให้ไม่สามารถผลิตแรงบิดได้มาก

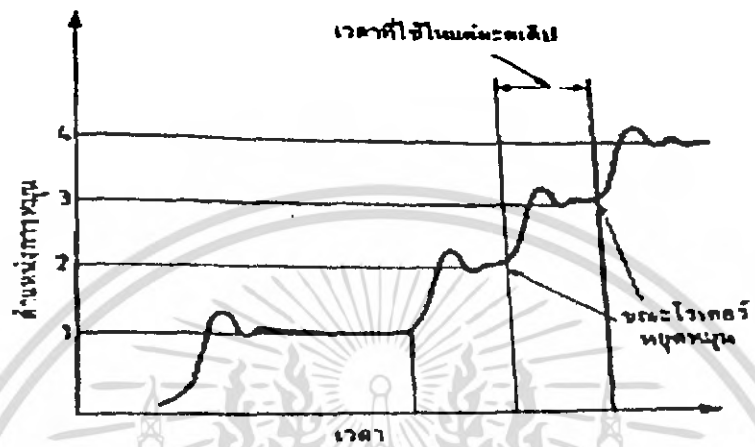


รูปที่ 2.14 แสดงลำดับขั้นการหมุนในมอเตอร์ 4 เฟส

2.3.5 โหมดการทำงานของสเต็ปมอเตอร์

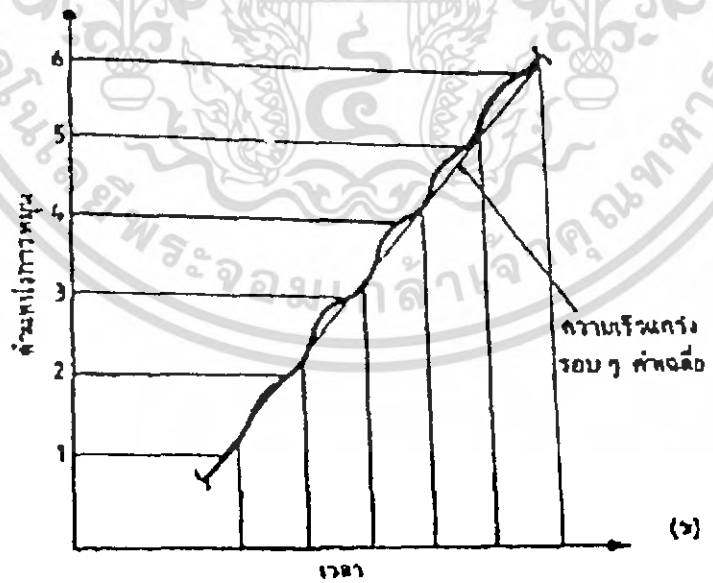
ถ้าจะแบ่งการทำงานของสเต็ปมอเตอร์ตามอัตราเร็วของมอเตอร์ของแต่ละสเต็ปจะแบ่งออกเป็น 2 โหมด คือ หมุนเป็นสเต็ป และหมุนอย่างต่อเนื่อง

โดยถ้าการหมุนเป็นแบบสเต็ปและมีเวลาหยุดนิ่งก่อนที่จะเปลี่ยนสเต็ปถัดไป จะเรียกการทำงานในโหมดนี้ว่า การหมุนเป็นสเต็ป ดังแสดงในรูปที่ 2.15



รูปที่ 2.15 กราฟแสดงการหมุนในโหมดการทำงานแบบหมุนเป็นสเต็ป

ถ้าเพิ่มอัตราเร็วในแต่ละสเต็ปให้เร็วขึ้น และเป็นไปอย่างต่อเนื่องไม่มีการหยุดนิ่ง จะเรียกการหมุนนี้ว่าการหมุนอย่างต่อเนื่อง ดังแสดงในรูปที่ 2.16



รูปที่ 2.16 กราฟแสดงสเต็ปการหมุนในโหมดการทำงานแบบหมุนต่อเนื่อง

2.3.6 วิธีการกระตุ้นเฟส

การที่จะทำให้สเต็ปมอเตอร์หมุนได้อย่างต่อเนื่องเหมือนกับการหมุนของมอเตอร์ไฟฟ้ากระแสตรงนั้น ต้องมีการจ่ายพัลส์เป็นลำดับอย่างต่อเนื่อง วิธีการที่จะกระตุ้นเฟสมีด้วยกันหลายวิธี แต่ที่นิยมใช้กันมีดังนี้

1. การกระตุ้นแบบเฟสเดี่ยว (Single-Phase Excitation)

วิธีนี้เป็นการกระตุ้นเฟสเพียงเฟสเดียวเท่านั้นที่จังหวะสัญญาณนาฬิกาหนึ่งๆ

จังหวะสัญญาณนาฬิกา	R	1	2	3	4	5	6	7	8	9	10
เฟส 1	■			■			■			■	
เฟส 2		■			■			■			■
เฟส 3			■			■			■		

รูปที่ 2.17 แสดงการกระตุ้นแบบเฟสเดี่ยว

2. การกระตุ้นแบบเฟสคู่ (Two-Phase Excitation)

วิธีนี้เป็นการกระตุ้นเฟสสองเฟสพร้อมกันในจังหวะสัญญาณนาฬิกาหนึ่งๆ

จังหวะสัญญาณนาฬิกา	R	1	2	3	4	5	6	7	8	9	10
เฟส 1	■	■			■	■			■	■	
เฟส 2		■	■			■	■			■	■
เฟส 3			■	■			■	■			■

รูปที่ 2.18 แสดงการกระตุ้นแบบเฟสคู่

3. การกระตุ้นแบบครึ่งสเต็ป (Half-Step Excitation)

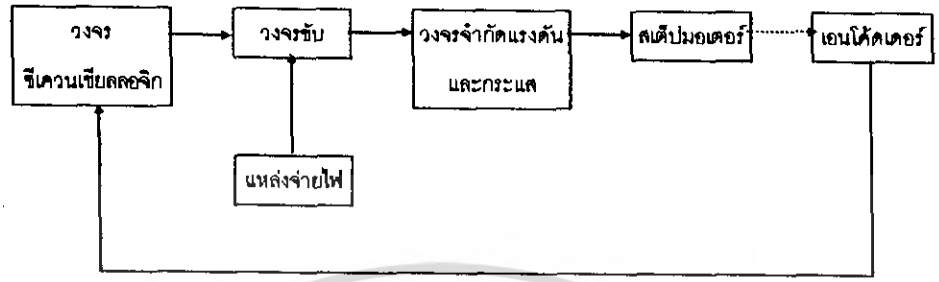
การกระตุ้นแบบนี้เป็นการรวมเอาการกระตุ้นทั้งสองแบบแรกเข้าด้วยกัน โดยจะกระตุ้นแบบเฟสเดี่ยวและจะกระตุ้นแบบเฟสคู่สลับกันไปอย่างต่อเนื่อง

จังหวะสัญญาณนาฬิกา	R	1	2	3	4	5	6	7	8	9	10
เฟส 1	■	■				■	■	■			
เฟส 2		■	■	■				■	■	■	
เฟส 3				■	■	■				■	■

รูปที่ 2.19 แสดงการกระตุ้นแบบครึ่งสเต็ป

2.3.7 การควบคุมสเต็ปมอเตอร์

บล็อกไดอะแกรมสำหรับการควบคุมสเต็ปมอเตอร์ แสดงได้ดังรูปที่ 2.20



รูปที่ 2.20 แสดงบล็อกไดอะแกรมของระบบควบคุมสเต็ปมอเตอร์

จากบล็อกไดอะแกรม วงจรซีเควนเชียลลอจิกจะรับพัลส์อินพุทและคำสั่งควบคุมทิศทาง แล้วจึงจ่ายพัลส์ที่ใช้กระตุ้นเฟสของสเต็ปมอเตอร์ออกไป แต่จะมีระดับของสัญญาณต่ำจึงต้องนำสัญญาณนี้ไปผ่านวงจรขับ เพื่อให้ระดับของสัญญาณสูงขึ้น และจะมีวงจรจำกัดแรงดันและกระแสซึ่งทำหน้าที่เป็นบัฟเฟอร์ให้กับวงจรขับและมอเตอร์ จากบล็อกไดอะแกรมจะเห็นได้ว่าไม่จำเป็นต้องต่อวงจรเอ็นโค้ดเดอร์ป้อนสัญญาณกลับมาเพื่อควบคุมตำแหน่งและความเร็ว แต่ในบางครั้งเช่นในกรณีที่ความสัมพันธ์ของการเคลื่อนที่ระหว่างแกนหมุนของมอเตอร์กับตำแหน่งของโหลดไม่ตรงกัน อันเกิดจากความผิดพลาดของเฟืองเกียร์ หรือในกรณีที่โหลดบางอย่างทำให้เกิดความผิดพลาดของสเต็ปขึ้น ก็จำเป็นที่จะต้องมีการใช้สัญญาณป้อนกลับเพื่อควบคุมตำแหน่งและความเร็วให้ถูกต้อง

2.3.8 วงจรขับ (Drive Circuit)

สัญญาณควบคุมที่ใช้สำหรับควบคุมการทำงานของสเต็ปมอเตอร์มักจะเป็นสัญญาณที่สร้างจากวงจรดิจิตอล เช่น จากไมโครคอนโทรลเลอร์ ซึ่งเป็นอุปกรณ์จำพวก TTL แรงดันที่ใช้มีค่าเท่ากับ 5 โวลต์ และสามารถจ่ายกระแสได้ไม่มากนัก แต่เนื่องจากการทำงานของสเต็ปมอเตอร์ต้องการแรงดันกระแสที่สูงกว่านั้น จึงจำเป็นต้องมีวงจรขับเพื่อทำหน้าที่จ่ายแรงดันกระแสที่เพียงพอให้กับตัวมอเตอร์ โดยทั่วๆไปวงจรขับมักสร้างจากไบโพลาร์ทรานซิสเตอร์ที่นำมาต่อใช้งานเป็นสวิตช์เปิด–ปิด ให้กระแสไหลผ่านไปยังขดลวดในทางเดียว เราเรียกวงจรขับแบบนี้ว่า ยูนิโพลาร์ ซึ่งมีการจ่ายกระแสไรทิศทางเดียว แต่ถ้าใช้สเต็ปมอเตอร์แบบไฮบริดจ์หรือแบบแม่เหล็กถาวร ซึ่งมักจะมี 2 เฟส จะต้องใช้วงจรขับที่สามารถจ่ายกระแสตรงได้ 2 ทิศทาง เราเรียกวงจรขับแบบนี้ว่า ไบโพลาร์ ซึ่งประกอบด้วยทรานซิสเตอร์หลายตัวต่อเป็นวงจรแบบบริดจ์

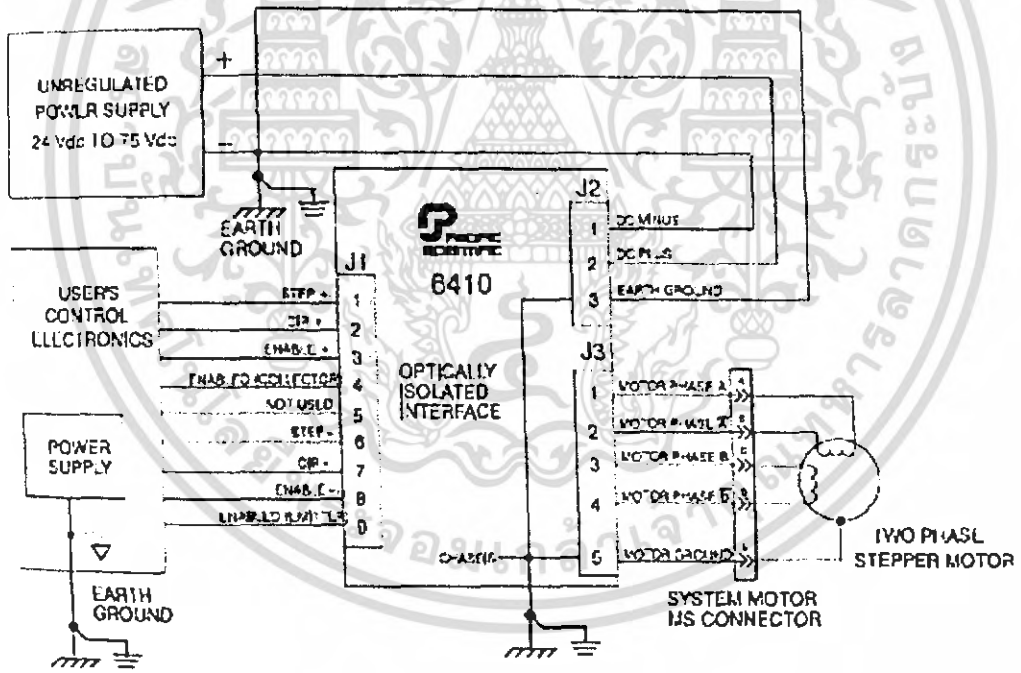
2.3.9 วงจรควบคุมมอเตอร์ (Motor Control Circuit)

ในการใช้งานมอเตอร์จำเป็นต้องมีการควบคุมข้อจำกัดให้เหมาะสมกับลักษณะงาน โดยเฉพาะอย่างยิ่งในงานที่เกี่ยวข้องกับการควบคุมการเคลื่อนที่ของหุ่นยนต์ซึ่งต้องการความละเอียดสูงแล้วนั้น วงจรที่จะเลือกมาใช้ควรที่จะสามารถเพิ่มความละเอียดของมอเตอร์เพื่อการควบคุมที่สมบูรณ์ยิ่งขึ้น

Pacific Scientific 6410 เป็นวงจรที่จะช่วยเพิ่มขีดความสามารถของสเต็ปมอเตอร์ชนิด 2 เฟส โดยอาศัยการควบคุมกระแสที่จ่ายให้แก่สเต็ปมอเตอร์ ค่าความละเอียดสูงสุดที่จะทำได้จะอยู่ที่ 50,000 สเต็ปต่อรอบ กระแสที่ออกจาก MA6410 สามารถปรับค่าได้โดย Dip Switch มีค่าอยู่ระหว่าง 5A rms (7.1A peak in microstep mode) จนถึง 0.625A rms (0.88A peak in microstep mode) แหล่งจ่ายไฟที่ใช้สำหรับวงจรนี้จะถูกออกแบบมาโดยเฉพาะ มีค่าความต่างศักย์อยู่ระหว่าง 24 Vdc จนถึง 75 Vdc

ภายในวงจร MA6410 ประกอบไปด้วยวงจรย่อยๆ ซึ่งจะทำงานพร้อมๆกันเพื่อควบคุมมอเตอร์ วงจรเหล่านั้นประกอบไปด้วย

1. Bipolar chopper drive
2. Microstepping
3. Digital Electronic Damping
4. Short circuit protection circuitry
5. MOSFET power devices
6. Optically isolated signal interface connection
7. UL Recognized-508C (Type R)



รูปที่ 2.21 แสดงการเชื่อมต่อวงจร MA 6410 กับอุปกรณ์ภายนอกต่างๆ

ตารางที่ 2.2 แสดง Step Size ที่วงจร MA 6410 สามารถควบคุมได้

Decimal		Binary	
Full	200	Half	400
Half	400	1/4	800
1/5	1,000	1/8	1,600
1/10	2,000	1/16	3,200
1.25	5,000	1/32	6,400
1/50	10,000	1/64	12,800
1/125	25,000	1/128	25,600
1/250	50,000	1/256	51,200

2.4 การติดต่อกับคอมพิวเตอร์

การอินเตอร์เฟสกับคอมพิวเตอร์ คือการทำงานติดต่อร่วมกันระหว่างซีพียูกับอุปกรณ์ภายนอกอื่นๆและการถ่ายข้อมูลระหว่างอุปกรณ์ต่างๆ นอกเหนือจากจะต้องทำงานติดต่อกับหน่วยความจำ (RAM, ROM) แล้วยังต้องมีการติดต่อกับอุปกรณ์ภายนอกที่มีการส่งข้อมูลอินพุท, เอาท์พุทอีกทางหนึ่ง ซึ่งเป็นการเพิ่มประสิทธิภาพให้กับระบบสมบูรณ์ในระบบต่างๆของอุปกรณ์อิเลคทรอนิกส์จะทำงานต่อเนื่องเป็นลูกโซ่ ดังเช่น การรับส่งข้อมูลจากซีพียูไปยังส่วนอื่นๆ เป็นต้น

การที่จะโอนย้ายข้อมูลทุกตัวนั้นจะต้องมีแหล่งที่ส่งข้อมูล และแหล่งที่รับข้อมูล สำหรับขบวนการเหล่านั้นจะมีส่วนสำคัญที่ว่า ข้อมูลนั้นเป็นแอดเดรสหรือว่าเป็นดาต้าจะส่งไปยังจุดไหน ด้วยอย่างเช่น ส่งไปยังหน่วยความจำ อุปกรณ์อินพุท/เอาท์พุท และจะส่งเมื่อไหร่ การทำงานเหล่านี้โดยทั่วไปจะต้องมีสัญญาณในการตรวจสอบความพร้อมเสมอ เพื่อที่จะให้ข้อมูลที่เราใช้งานนั้นๆเป็นระเบียบ ตัวอย่างเช่น ส่งข้อมูลจากซีพียูไปที่อุปกรณ์รอบข้าง เป็นต้น ซึ่งจุดรับส่งข้อมูลคู่หนึ่งๆอาจจะเป็นระหว่างซีพียูด้วยกัน หรือซีพียูกับหน่วยความจำก็ได้ สำหรับข้อมูลที่โอนย้ายไปมานั้นจะอยู่ในลักษณะของเลขฐานสอง ตัวอย่างเช่น 01101100_2 ซึ่งแต่ละตัวจะแทนด้วย 8 Bit หรือ 16 Bit ก็ขึ้นอยู่กับการใช้งานของระบบนั้นๆ ถ้าหากเป็นการต่อจากพอร์ตพีซี ไม่ว่าจะเป็น Parallel หรือ Serial ในสัญญาณที่ส่งมาจะมีระบบแรงดันไฟฟ้า สามารถอธิบายได้ดังนี้

1. พอร์ตขนาน (Printer Port) ใช้แรงดันไฟฟ้าประมาณ 0 ถึง +5 Vdc
2. พอร์ตอนุกรม (RS-232) ใช้แรงดันไฟฟ้าประมาณ +3 ถึง +25 Vdc

2.4.1 การติดต่อกับพอร์ตขนาน

ในการประยุกต์ใช้งานอุปกรณ์เชื่อมต่อ (Interface) นั้นเรามีการใช้งานพอร์ตขนาน หรือ Parallel Port อย่างแพร่หลาย โดยจุดเริ่มต้นคือการนำมาใช้งานกับพริ้นเตอร์ จนทำให้บางคนเรียกว่า Printer Port ก็มี ซึ่งพอร์ตขนานนั้นมีความสามารถและน่าสนใจอยู่หลายประการ จึงน่าจะทำความเข้าในการทำงานพื้นฐานเพื่อให้การสร้างแอพพลิเคชันภายหลังทำได้อย่างเข้าใจและรวดเร็ว

เราสามารถใช้พอร์ตขนานเป็นตัวรับสัญญาณอินพุทและเอาท์พุทแบบดิจิตอล เพื่อที่จะใช้ในการอินเตอร์เฟสกับอุปกรณ์อื่นๆ พอร์ตนี้ประกอบไปด้วยรีจิสเตอร์อยู่หลายแบบ ได้แก่

1. รีจิสเตอร์เอาท์พุทขนาด 8 บิท ซึ่งเราสามารถอ่านข้อมูลที่ส่งออกไปกลับมาตรวจสอบได้
2. รีจิสเตอร์ขนาด 4 บิท ซึ่งสามารถอ่านข้อมูลกลับมาตรวจสอบและใช้เป็นรีจิสเตอร์อินพุทได้
3. รีจิสเตอร์อินพุทขนาด 5 บิท
4. รีจิสเตอร์ขนาด 1 บิท ซึ่งสามารถใช้ส่งสัญญาณในระดับที่ 7 ได้

นอกจากนี้ค่าแอดเดรสของรีจิสเตอร์แต่ละตัวในการ์ด มีค่าแอดเดรสกำกับอยู่ 2 ค่า ดังนั้นสมมติว่าถ้าในระบบของเรามีการ์ดอยู่ 2 แผ่น เราสามารถกำหนดแอดเดรสที่ต่างกันให้แก่รีจิสเตอร์แต่ละตัวในการ์ดแต่ละแผ่น เพื่อนกันไม่ให้รีจิสเตอร์เดียวกันในการ์ดทั้ง 2 แผ่นทำงานขึ้นพร้อมกัน ซึ่งจะก่อให้เกิดจุดขัดแย้งขึ้น ซึ่งอาจทำให้ข้อมูลที่ส่งผิดพลาดไป อินพุทและเอาท์พุทของรีจิสเตอร์ที่กล่าวมาข้างต้น จะต่อเข้ากับคอนเนคเตอร์ 25 ขา แบบ D ที่อยู่ด้านหลังการ์ดทำให้การใช้งานทำได้สะดวก

ตารางที่ 2.3 แสดงลักษณะสัญญาณของพอร์ตขนานทั้งหมด

Pin No. (D-Type 25)	Signal Name	Bit	Direction (In/Out)
1	nStrobe	-C0	Output
2	Data 0 (Bit 0)	D0	Output
3	Data 1 (Bit 1)	D1	Output
4	Data 2 (Bit 2)	D2	Output
5	Data 3 (Bit 3)	D3	Output
6	Data 4 (Bit 4)	D4	Output
7	Data 5 (Bit 5)	D5	Output
8	Data 6 (Bit 6)	D6	Output
9	Data 7 (Bit 7)	D7	Output
10	nAck	S6	Input
11	Busy	-S7	Input
12	PaperEnd	S5	Input
13	Select	S4	Input
14	nAutoFeed	-C1	Output
15	nError	S3	Input
16	nInitialize	C2	Output
17	nSelectPrinter	-C3	Output
18 - 25	Ground		-

อย่างไรก็ตามพอร์ตขนานก็มีข้อเสียที่ไม่สามารถทำงานได้ในระยะทางไกลๆ เพราะจะเกิดความผิดพลาดของข้อมูลได้ง่ายเนื่องจากแรงดันไม่สม่ำเสมอ และสิ้นเปลืองค่าใช้จ่ายในเรื่องของสายสัญญาณที่ต้องใช้สายสัญญาณจำนวนมาก

2.4.2 วิธีการแก้ไขค่าแอดเดรสของการ์ด

ค่าแอดเดรสของรีจิสเตอร์ต่างๆในการ์ด Parallel Port มีค่าเท่ากับ 0378_{16}, 0379_{16} และ $037A_{16}$ เราสามารถแก้ไขค่าแอดเดรสนี้ให้มีค่าเป็น 0278_{16}, 0279_{16} และ $027A_{16}$ สำหรับรายละเอียดของการ์ดและวงจร สามารถดูได้จากคู่มือของการ์ด การแก้ไขค่าแอดเดรสให้เป็นค่าใหม่ทำได้โดยตัดทางเดินสัญญาณออกหนึ่งเส้น ตำแหน่งของทางเดินสัญญาณนี้อยู่ระหว่างขาทั้งสองขอ จั๊มเปอร์ (Jumper) สองขา ซึ่งบนการ์ดระบุไว้ด้วยอักษร J1 เมื่อเราตัดทางเดินระหว่างขาทั้งสองนี้ ในการถอดรหัสการเลือกใช้งานรีจิสเตอร์ต่างๆต้องใช้ค่าแอดเดรสใหม่ทั้งหมด

บทที่ 3
การออกแบบและวิธีการดำเนินงาน

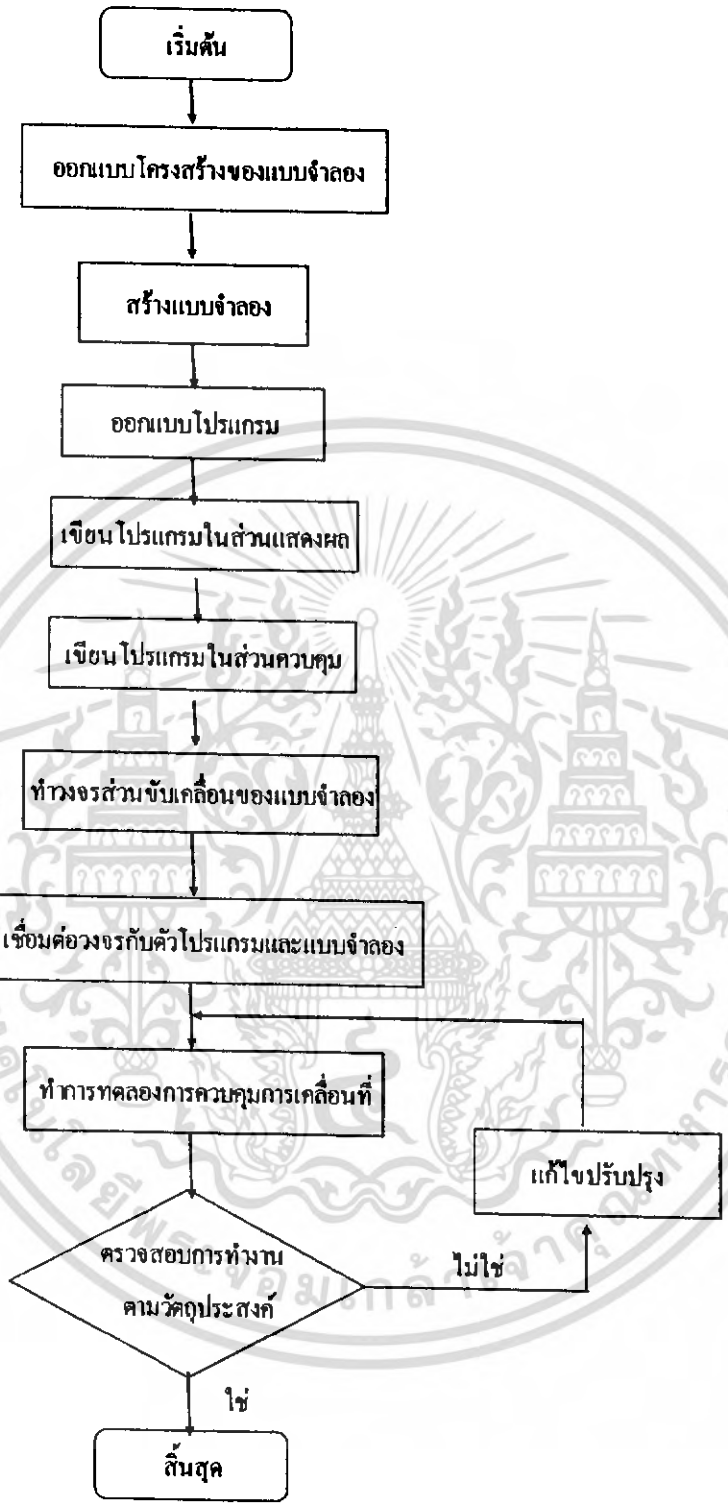
3.1 การวางแผนการดำเนินงาน

แผนการดำเนินงาน ประกอบด้วย

1. วางแผนการดำเนินงาน
2. ศึกษาวิธีการทำงาน และทฤษฎีที่เกี่ยวข้อง
3. ทำการออกแบบโครงสร้างของแบบจำลองเฮกซะพอด
4. ทำการไขปรับปรุงในส่วนที่บกพร่อง
5. สร้างแบบจำลอง
6. ศึกษาข้อมูลเกี่ยวกับวงจรควบคุม
7. ศึกษาการเขียนโปรแกรมด้วย Visual Basic 6.0
8. ออกแบบโปรแกรมและเขียนโปรแกรมในส่วนแสดงผลและในส่วนควบคุม
9. ทำการออกแบบและสร้างวงจรควบคุมในส่วนขับเคลื่อน
10. เชื่อมต่อกับตัวโปรแกรมเข้ากับแบบจำลอง
11. ทำการทดลองเพื่อศึกษาการเคลื่อนที่
12. สรุปผลและจัดทำปริญญานิพนธ์

3.2 แผนการทำงานและสร้างแบบจำลองเฮกซะพอดพร้อมโปรแกรมควบคุมการทำงานและจำลองการเคลื่อนที่

การวางแผนในการสร้างแบบจำลองของเฮกซะพอดและโปรแกรมควบคุมการทำงานพร้อมจำลองการเคลื่อนที่ของเฮกซะพอด สามารถแบ่งการทำงานออกเป็น 3 ส่วน คือ ส่วนของฮาร์ดแวร์ ส่วนของวงจรควบคุม และส่วนของโปรแกรมที่ใช้จำลองการเคลื่อนที่และควบคุมการทำงานของเฮกซะพอด ซึ่งในการทำงานจะเน้นไปในส่วนของการเขียนโปรแกรมที่ใช้จำลองการเคลื่อนที่และควบคุมการทำงาน ซึ่งจะเป็นขั้นตอนที่ใช้เวลานานที่สุด เนื่องจากต้องมีการออกแบบและคำนวณหาสมการเพื่อใช้เป็นตัวแทนในการเคลื่อนที่ของเฮกซะพอด ทดสอบการทำงานของโปรแกรม และทำการแก้ไขจนกระทั่งสามารถควบคุมการทำงานได้ตามวัตถุประสงค์

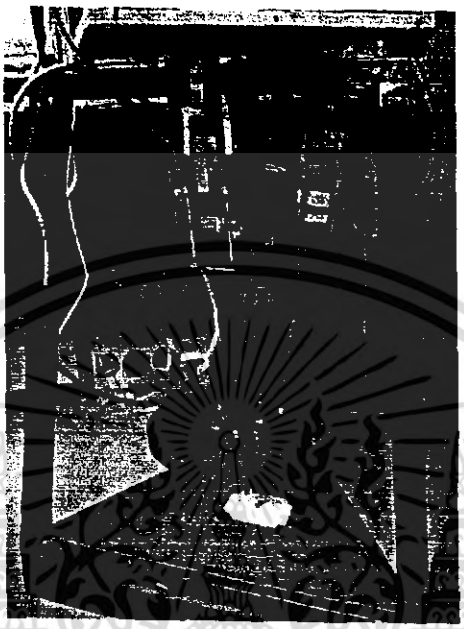


รูปที่ 3.1 แสดงขั้นตอนการดำเนินงาน

3.2.1 การดำเนินงานในส่วนฮาร์ดแวร์

ส่วนประกอบที่สำคัญในการสร้างแบบจำลองเฮกซะพอดมีดังนี้

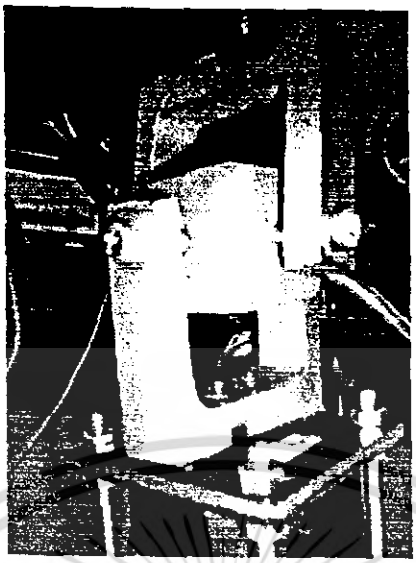
1. **โครงสร้างของเฮกซะพอด**



รูปที่ 3.2 แสดงโครงสร้างของเฮกซะพอด

2. **ข้อต่อ (Joint)**

เพื่อให้จุดปลายของเฮกซะพอดเคลื่อนที่ไปได้ ซึ่งข้อต่อนี้มีอยู่ 2 ส่วนด้วยกันคือ ข้อต่อที่ติดกับตัวโครงสร้าง และข้อต่อที่ติดกับจุดปลาย



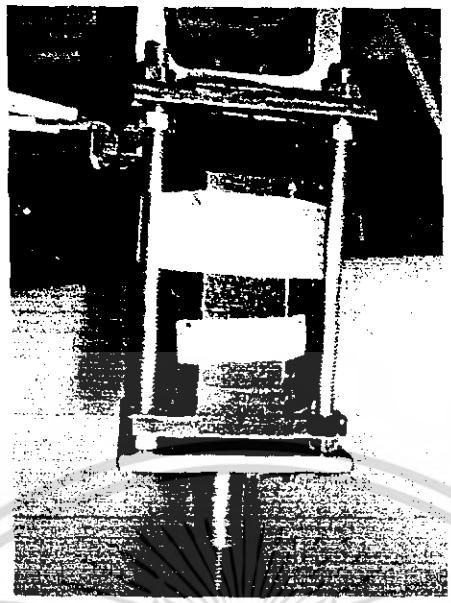
รูปที่ 3.3 แสดงข้อต่อที่ติดกับตัวโครงสร้าง

รูปที่ 3.4 แสดงข้อต่อที่ติดกับจุดปลาย

3. สเต็ปมอเตอร์

ที่มีความละเอียด 200 สเต็ปต่อรอบ

4. ที่ยึดสเต็ปมอเตอร์เข้ากับโครงสร้าง



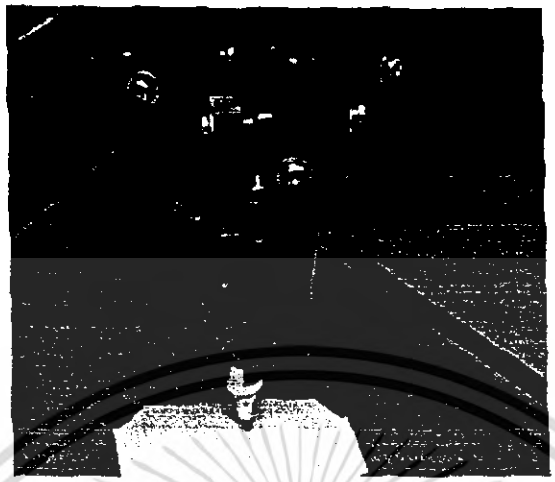
รูปที่ 3.5 แสดงสเต็ปมอเตอร์และที่ยึดสเต็ปมอเตอร์เข้ากับโครงสร้าง

5. สตัด (Stud)

ประกอบด้วยเกลียวและปลอก เพื่อให้สามารถยึดหดได้ตามการเคลื่อนที่

รูปที่ 3.6 แสดงสตัด (Stud)

6. จุดปลายของเฮกซะพอด ประกอบด้วยแผ่นเพลต (Plate) ที่ใช้ยึดจุดปลายเพื่อใช้ศึกษาการเคลื่อนที่

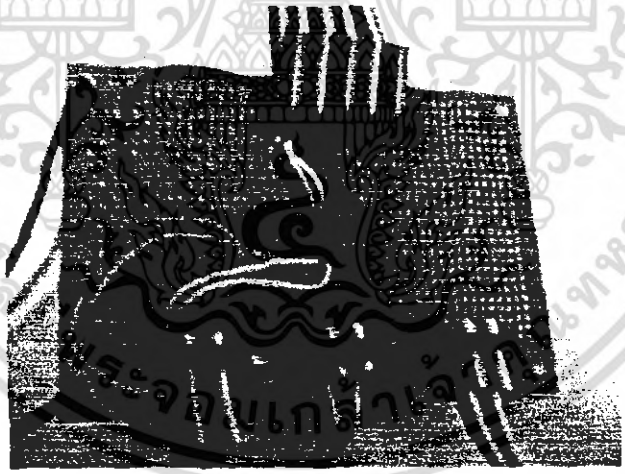


รูปที่ 3.7 แสดงแผ่นเพลตและจุดปลายของเฮกซะพอด

3.2.2 การดำเนินงานในส่วนวงจรควบคุม

อุปกรณ์ที่ใช้ในการควบคุมการทำงานประกอบด้วย

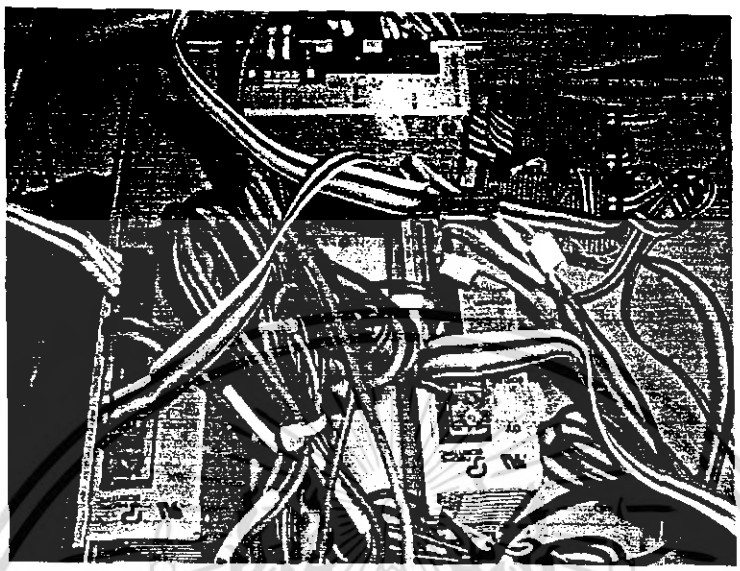
1. IC ULN2803



รูปที่ 3.8 แสดง IC ULN2803

2. ชุดควบคุมสเต็ปมอเตอร์แบบไมโครสเต็ปของแปซิฟิค (Pacific Microstep Controller)

ความละเอียดการควบคุม 50,000 สเต็ปต่อรอบ

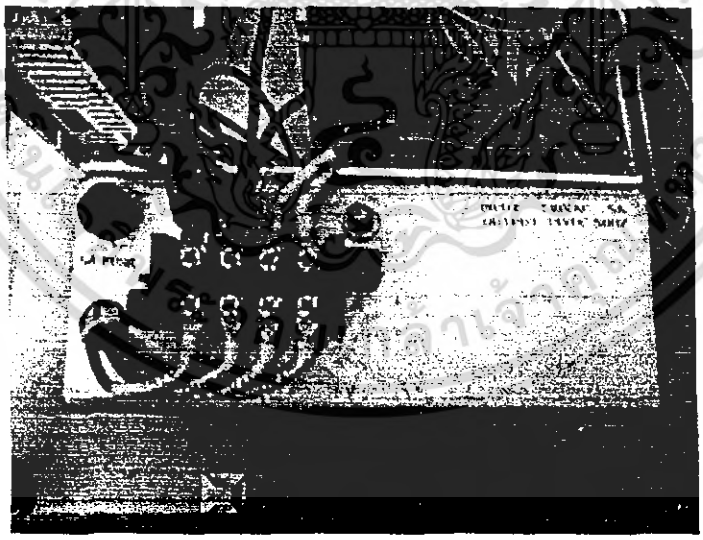


รูปที่ 3.9 แสดงชุดควบคุมสเต็ปมอเตอร์แบบไมโครสเต็ปของแปซิฟิค

3. แหล่งจ่ายไฟสำหรับมอเตอร์

I/P 230 Vdc

O/P 34 Vdc

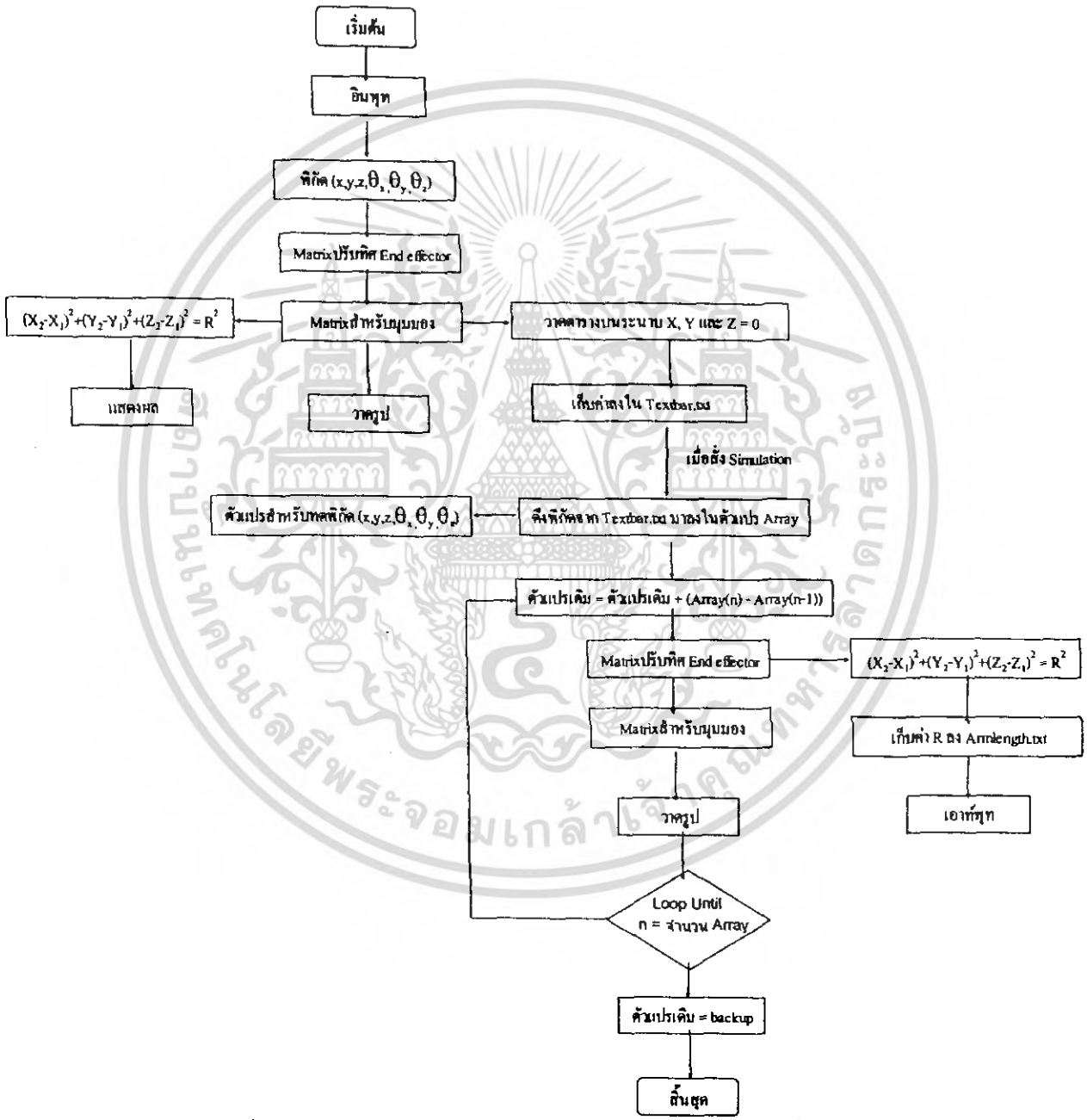


รูปที่ 3.10 แสดงแหล่งจ่ายไฟสำหรับมอเตอร์

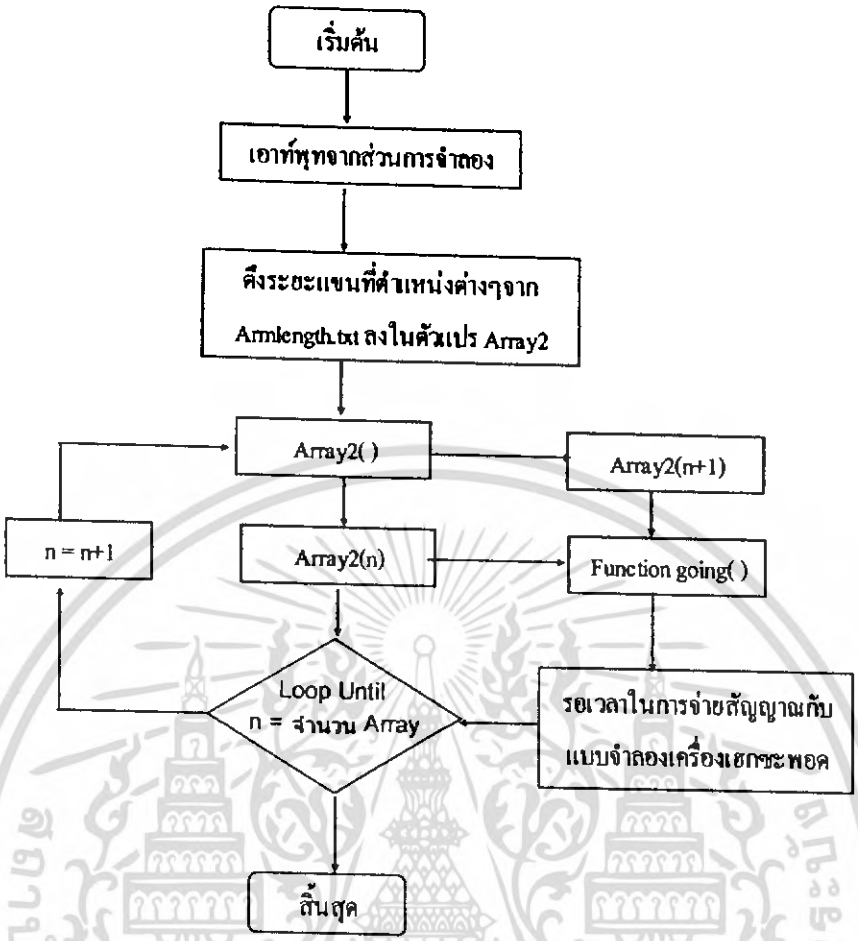
3.2.3 การดำเนินงานในส่วนของโปรแกรมที่ใช้จำลองการเคลื่อนที่และควบคุมการทำงาน

ขั้นตอนการทำงานคือ

1. ศึกษาการเขียนโปรแกรมด้วยโปรแกรม Visual Basic 6.0 เพื่อเขียนโปรแกรมให้สามารถรับคำสั่งและจำลองการเคลื่อนที่บนคอมพิวเตอร์ พร้อมทั้งส่งข้อมูลคำสั่งให้ควบคุมแบบจำลองที่สร้างขึ้น
2. ศึกษาการเคลื่อนที่ของหุ่นยนต์
3. ออกแบบตัวโปรแกรม และหาสมการที่ใช้ในการเคลื่อนที่ของเฮกซะพอด ซึ่งมีแนวทางในการเขียนโปรแกรมดังแสดงในแผนการดำเนินงานดังนี้

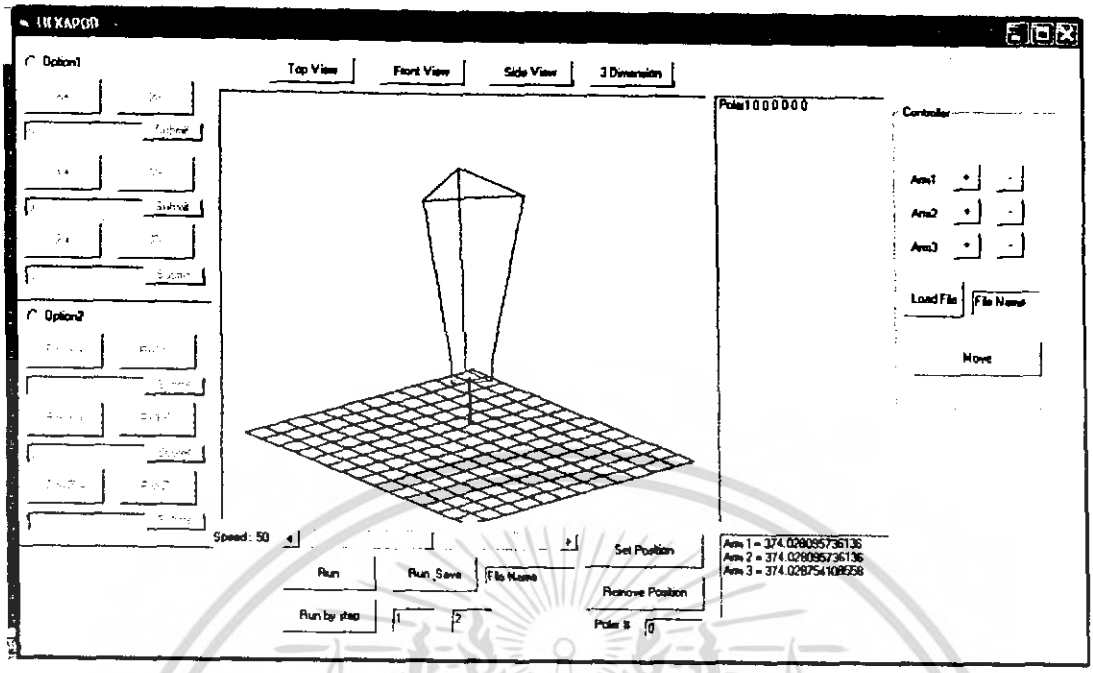


รูปที่ 3.11 แสดงแผนการการเขียนโปรแกรมในส่วนของการจำลองการเคลื่อนที่



รูปที่ 3.12 แสดงแผนการการเขียนโปรแกรมในส่วนของการควบคุมแบบจำลอง

4. เขียนโปรแกรมทั้งในส่วนที่ใช้จำลองการเคลื่อนที่ และส่วนที่ใช้เชื่อมต่อกับเฮกซะพอด
5. ตรวจสอบการทำงานของโปรแกรมทั้งการจำลองการเคลื่อนที่ และการควบคุมการทำงานของเฮกซะพอด



รูปที่ 3.13 แสดงหน้าจอโปรแกรม

บทที่ 4
ผลการดำเนินงาน

ในการจัดทำเฮกซะพอดนี้ ได้แบ่งการดำเนินงานออกเป็น 3 ส่วนคือ ส่วนของฮาร์ดแวร์ ซึ่งเกี่ยวข้องกับการศึกษาและออกแบบโครงสร้างของเฮกซะพอด ส่วนที่สองคือ ส่วนของวงจรควบคุมการทำงานของแบบจำลอง ได้มีการนำเอาชุดควบคุมสเต็ปมอเตอร์สำเร็จรูปมาปรับใช้สำหรับควบคุมการทำงานของแบบจำลอง ซึ่งวงจรนี้นำมาใช้นี้มีความสามารถในการปรับค่าความละเอียดของสเต็ปมอเตอร์จาก 200 สเต็ปต่อรอบ ให้มีค่าถึง 50,000 สเต็ปต่อรอบ เพื่อให้มีความละเอียดในการเคลื่อนที่มากขึ้น ส่วนสุดท้ายคือส่วนของโปรแกรมที่ใช้จำลองการเคลื่อนที่และควบคุมแบบจำลอง ได้ทำการศึกษาการเขียนโปรแกรมด้วย Visual Basic 6.0 และลักษณะการเคลื่อนที่ของเฮกซะพอด เพื่อให้โปรแกรมนี้สามารถรับข้อมูลการเคลื่อนที่และจำลองการเคลื่อนที่บนคอมพิวเตอร์ พร้อมทั้งส่งข้อมูลนั้นไปควบคุมการทำงานของแบบจำลอง

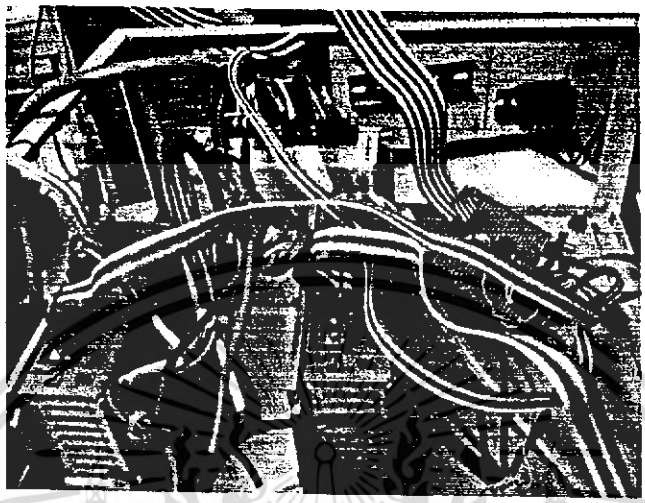
4.1 ผลการดำเนินงานด้านฮาร์ดแวร์

ผลการดำเนินงานทางด้านฮาร์ดแวร์ คือได้แบบจำลองเฮกซะพอดซึ่งสามารถแสดงรายละเอียดดังนี้

รูปที่ 4.1 แสดงแบบจำลองเฮกซะพอด

4.2 ผลการดำเนินงานด้านส่วนควบคุมแบบจำลอง

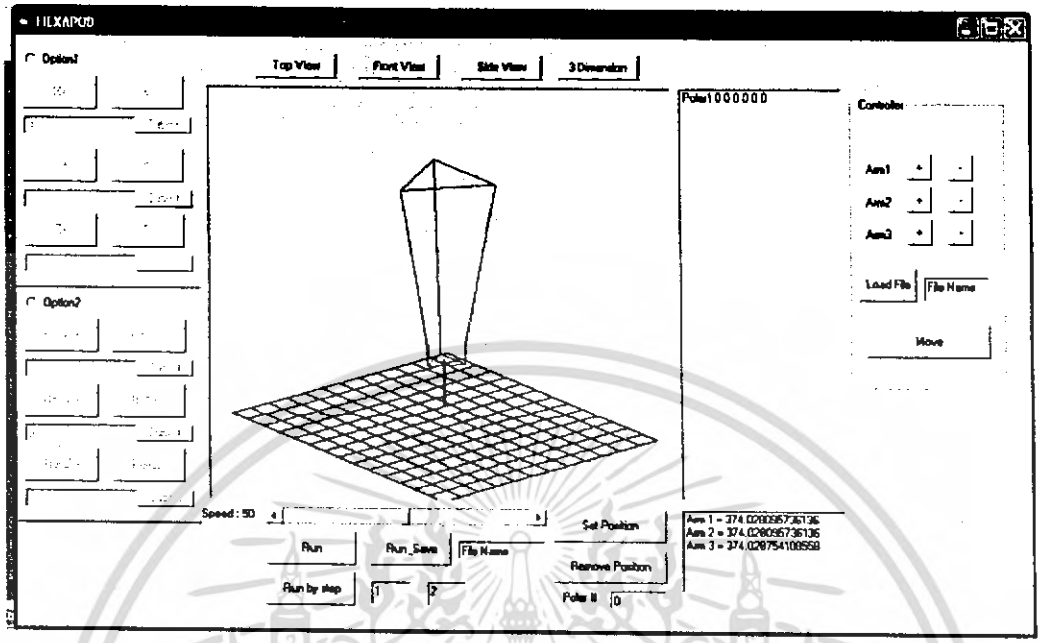
ผลของการดำเนินงานด้านวงจรควบคุมการทำงาน คือชุดควบคุมการทำงานของแบบจำลอง ซึ่งจะรับส่งข้อมูลระหว่างแบบจำลองกับโปรแกรมควบคุมผ่านทางพอร์ตขนาน



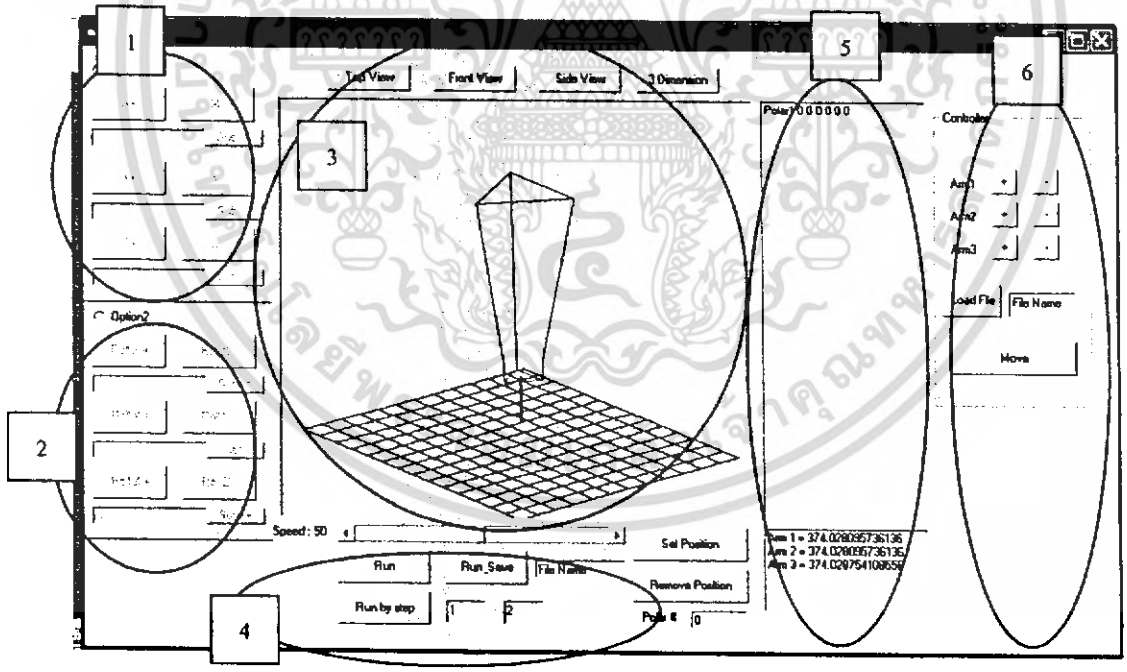
รูปที่ 4.2 แสดงชุดควบคุมการทำงาน

รูปที่ 4.3 แสดงแผงวงจรที่ใช้สำหรับรับ-ส่งข้อมูลเพื่อควบคุมการทำงาน

4.3 ผลการดำเนินงานด้านโปรแกรมที่ใช้จำลองการเคลื่อนที่และควบคุมแบบจำลอง



รูปที่ 4.4 แสดงหน้าจอโปรแกรม



รูปที่ 4.5 แสดงส่วนประกอบต่างๆของโปรแกรม

รายละเอียดส่วนประกอบต่างๆของโปรแกรม

ส่วนที่ 1 คือ ส่วนที่ผู้ใช้ใส่ค่าตำแหน่งของจุดปลาย ที่ต้องการให้เคลื่อนไป

ส่วนที่ 2 คือ ส่วนที่ผู้ใช้ใส่ค่ามุมที่ต้องการให้จุดปลายหมุนไปตามองศาที่กำหนด

ส่วนที่ 3 คือ ส่วนแสดงผลซึ่งผู้ใช้สามารถเลือกมุมมองได้

ส่วนที่ 4 คือ ส่วนที่ใช้สั่งให้โปรแกรมทำการจำลองการเคลื่อนที่

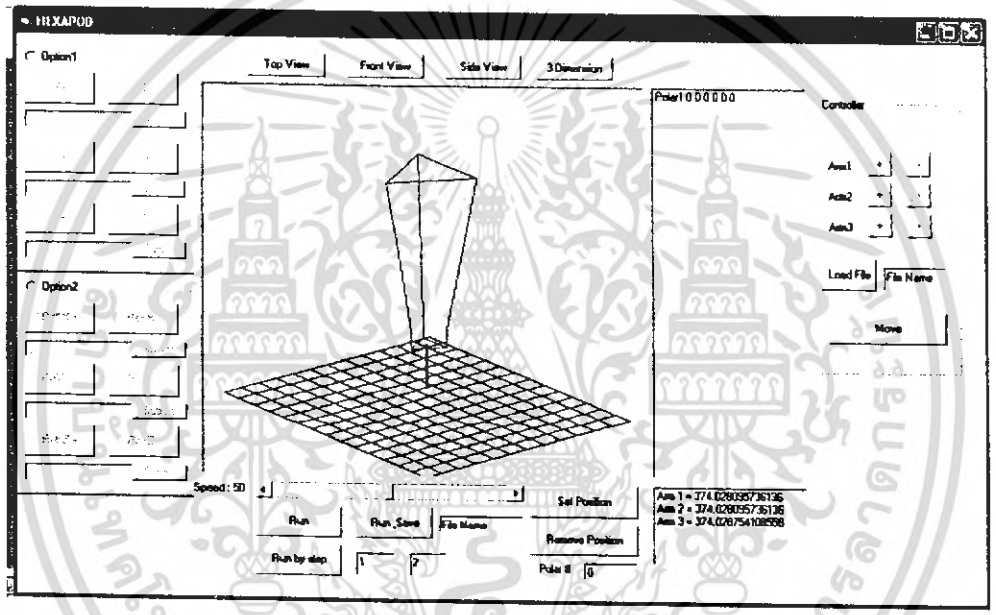
ส่วนที่ 5 คือ ส่วนแสดงตำแหน่งต่างๆที่ผู้ใช้บันทึกไว้ และขนาดของแขนในตำแหน่งสุดท้าย

ส่วนที่ 6 คือ ส่วนควบคุมการทำงานของแบบจำลองเฮกซะพอด

4.3.1 ผลการดำเนินงานด้านโปรแกรมที่ใช้จำลองการเคลื่อนที่

โปรแกรมที่ใช้จำลองการเคลื่อนที่มีขั้นตอนการทำงานของโปรแกรมดังนี้

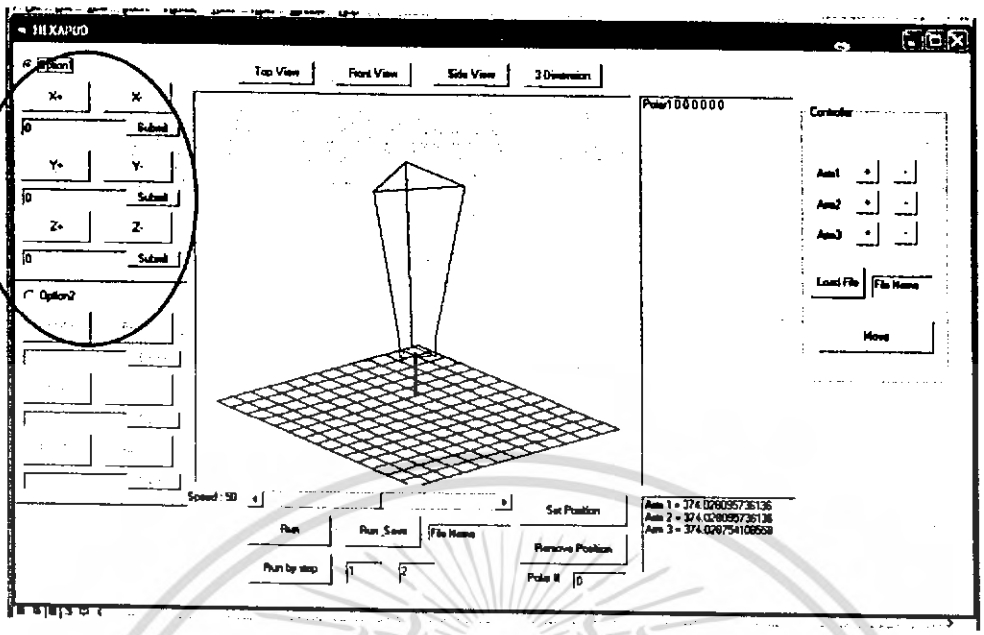
1. เริ่มต้นโปรแกรม



รูปที่ 4.6 แสดงหน้าจอตอนเริ่มต้นโปรแกรม

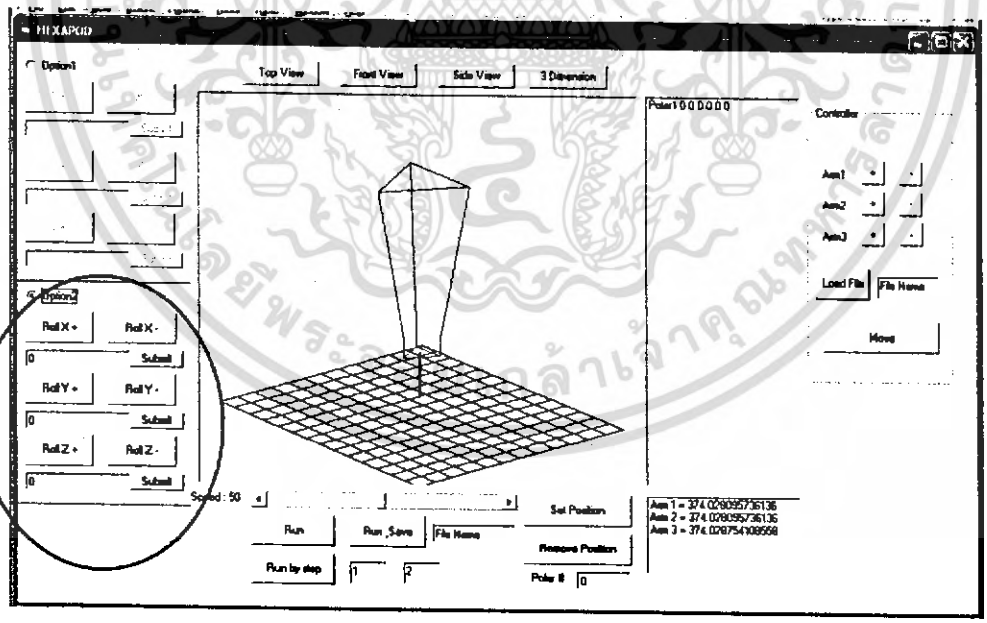
2. เลือกการควบคุมว่าต้องการควบคุมการเคลื่อนที่หรือการหมุนจุดปลายที่ Option 1 และ Option 2

- Option 1 จะเป็นการควบคุมการเคลื่อนที่ตามแกน x y z ซึ่งเป็นการควบคุมการเคลื่อนที่แบบย้อนกลับ (Inverse Kinematics) จะมี 2 วิธีในการควบคุมคือ การกดเพิ่มค่าตำแหน่งทีละ 1 หน่วย และการใส่ระยะทางที่ต้องการลงไป



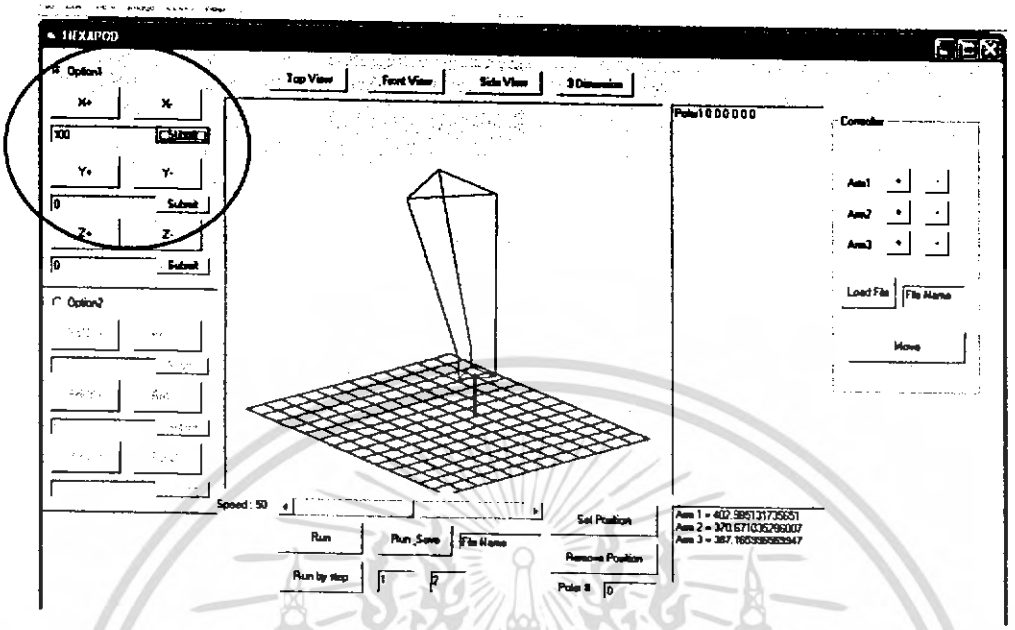
รูปที่ 4.7 แสดงการเลือก Option 1

- Option 2 จะเป็นการควบคุมการหมุนของจุดปลายของเฮกซะพอดครอบแกนต่างๆให้ได้องศาตามที่ต้องการ จะมี 2 วิธีในการควบคุมคือ การกดเพิ่มค่าองศาทีละ 1 องศา และการใส่มุมลงไปเลย ซึ่งในส่วน option 2 นี้จะไม่สามารถไปควบคุมตัวแบบจำลองจริงได้ จะทำได้เพียงจำลองบนคอมพิวเตอร์เท่านั้น

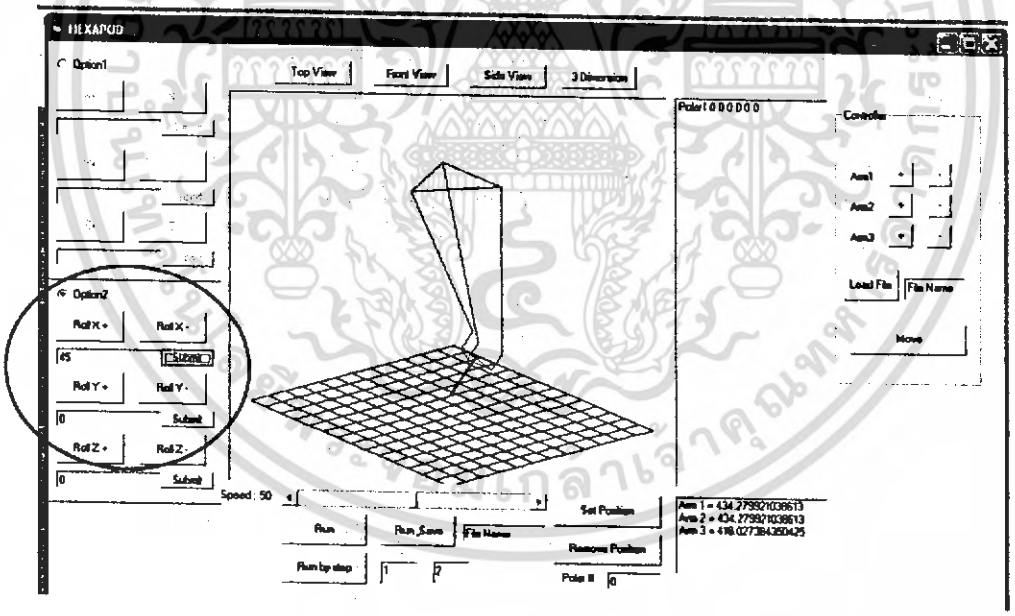


รูปที่ 4.8 แสดงการเลือก Option 2

3. ใส่ค่าตำแหน่งหรือองศาลงไป และกด Submit เพื่อให้โปรแกรมทำการแสดงตำแหน่งที่ป้อนลงไป

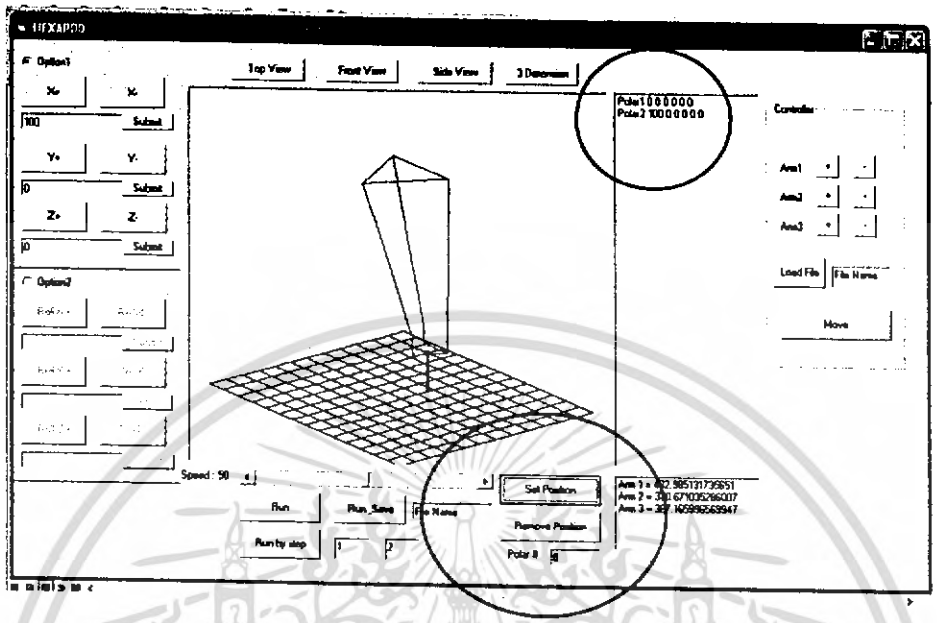


รูปที่ 4.9 แสดงการป้อนค่าตำแหน่งที่ต้องการ



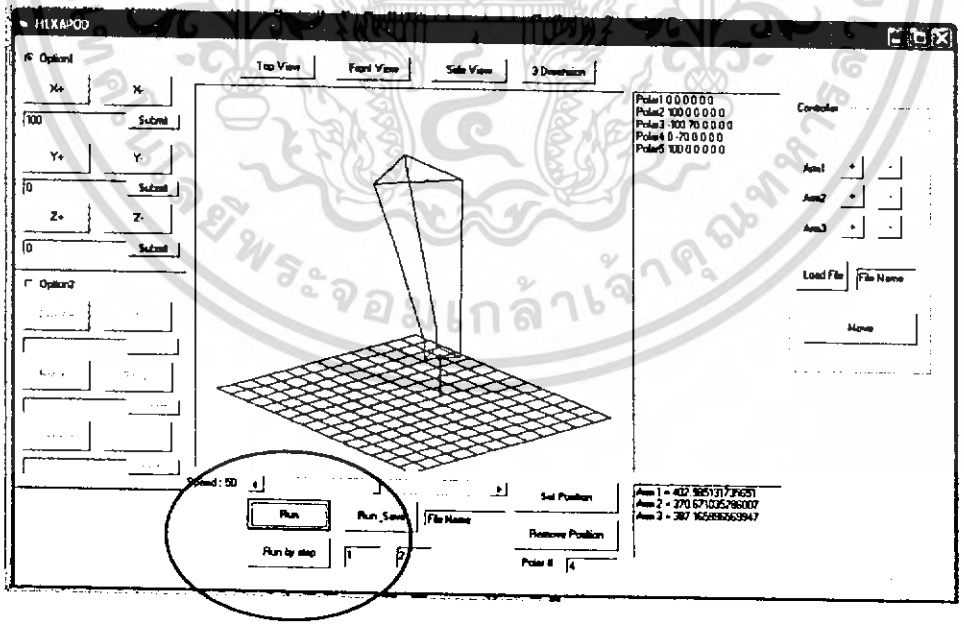
รูปที่ 4.10 แสดงการป้อนองศา

4. กด Set Position เมื่อต้องการให้โปรแกรมบันทึกตำแหน่งที่ได้ตั้งไว้ ซึ่งตำแหน่งแรกของโปรแกรมจะถูกตั้งไว้ที่ตำแหน่งเริ่มต้นให้เป็นตำแหน่ง Polar 1

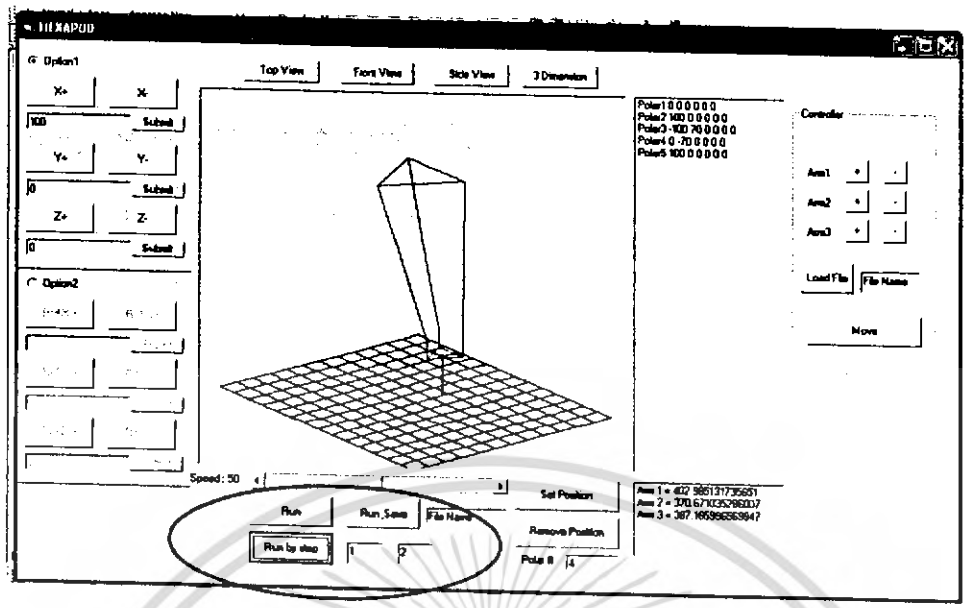


รูปที่ 4.11 แสดงการบันทึกตำแหน่ง

5. กด Run ถ้าต้องการให้โปรแกรมจำลองการเคลื่อนที่ที่ได้ป้อนลงไปทั้งหมด
กด Run by Step เพื่อจำลองการเคลื่อนที่ทีละตำแหน่ง
ซึ่งสามารถปรับความเร็วในการจำลองการเคลื่อนที่ได้

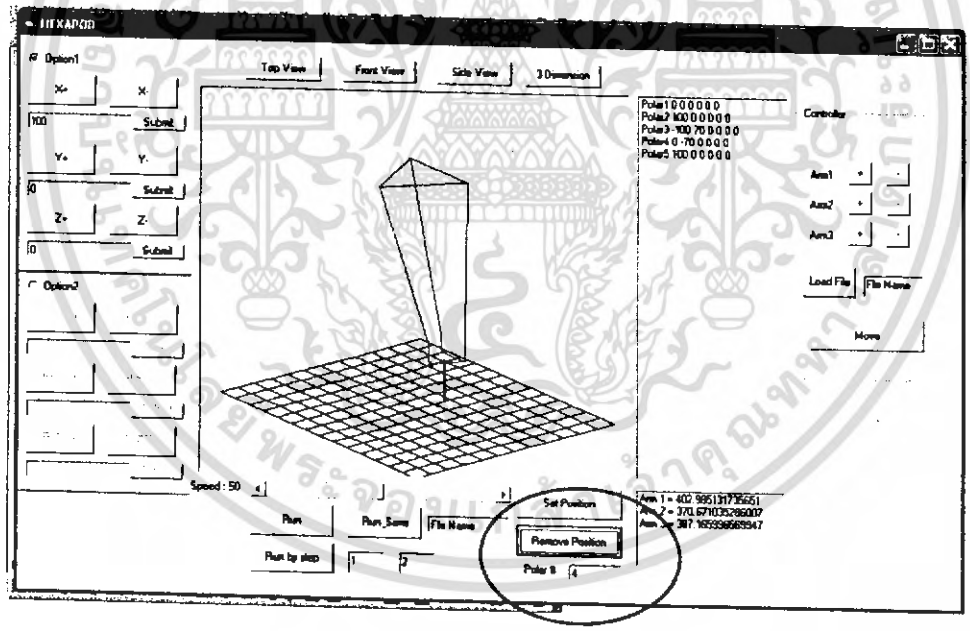


รูปที่ 4.12 แสดงการจำลองการเคลื่อนที่บนคอมพิวเตอร์



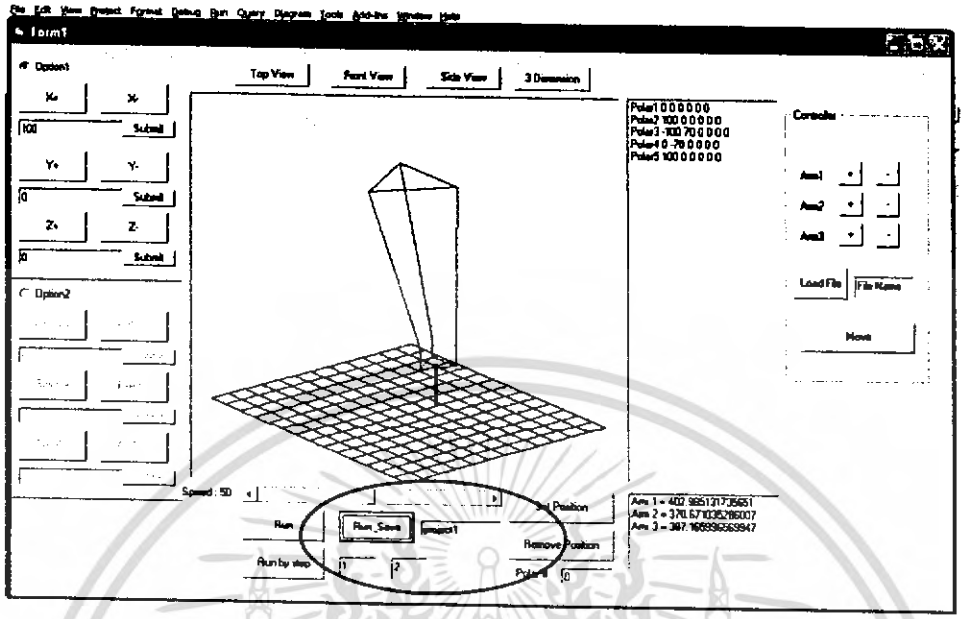
รูปที่ 4.13 แสดงการจำลองการเคลื่อนที่จากตำแหน่ง 1 ไปยังตำแหน่ง 2

6. ถ้าต้องการลบตำแหน่งที่บันทึกไว้ ให้ใส่ตำแหน่งที่ต้องการลบแล้วกด Remove Position



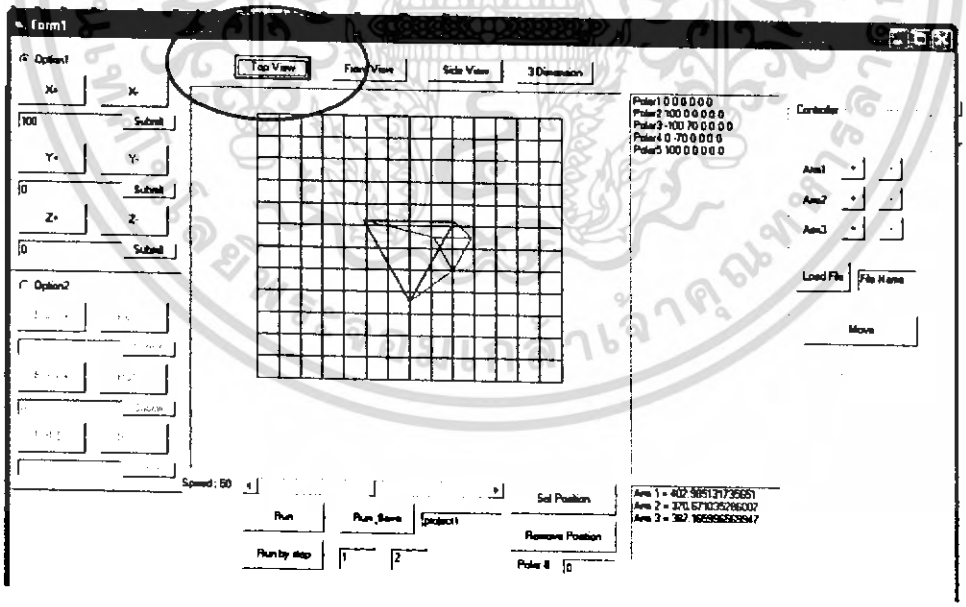
รูปที่ 4.14 แสดงต้องการลบตำแหน่งที่ 4

7. เมื่อต้องการบันทึกการเคลื่อนที่ทั้งหมด พิมพ์ชื่อไฟล์ที่ต้องการ และใช้คำสั่ง Run_Save เพื่อทำการบันทึก

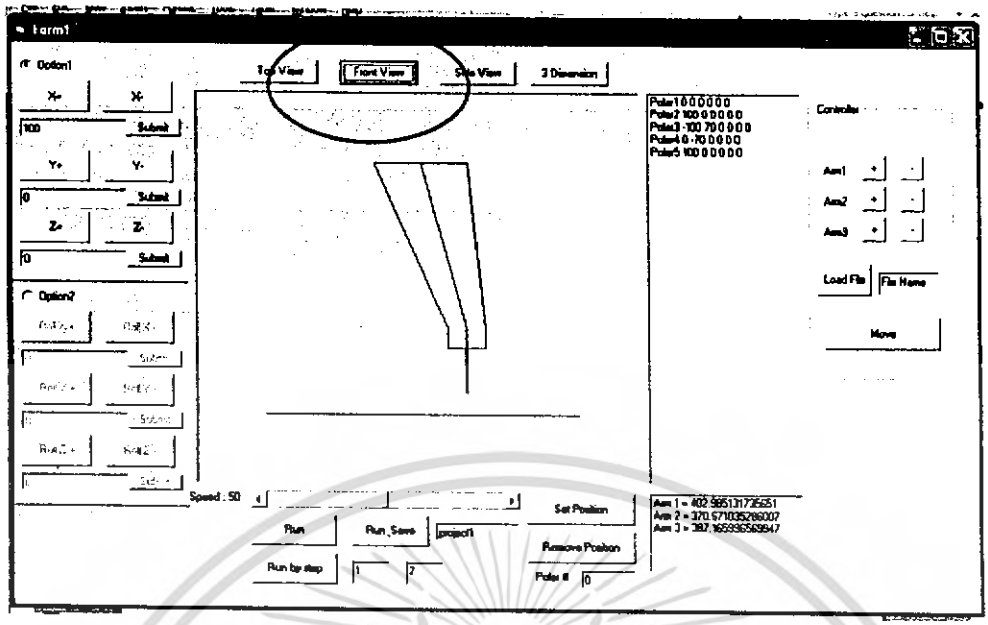


รูปที่ 4.15 แสดงการบันทึกไฟล์ชื่อ project1

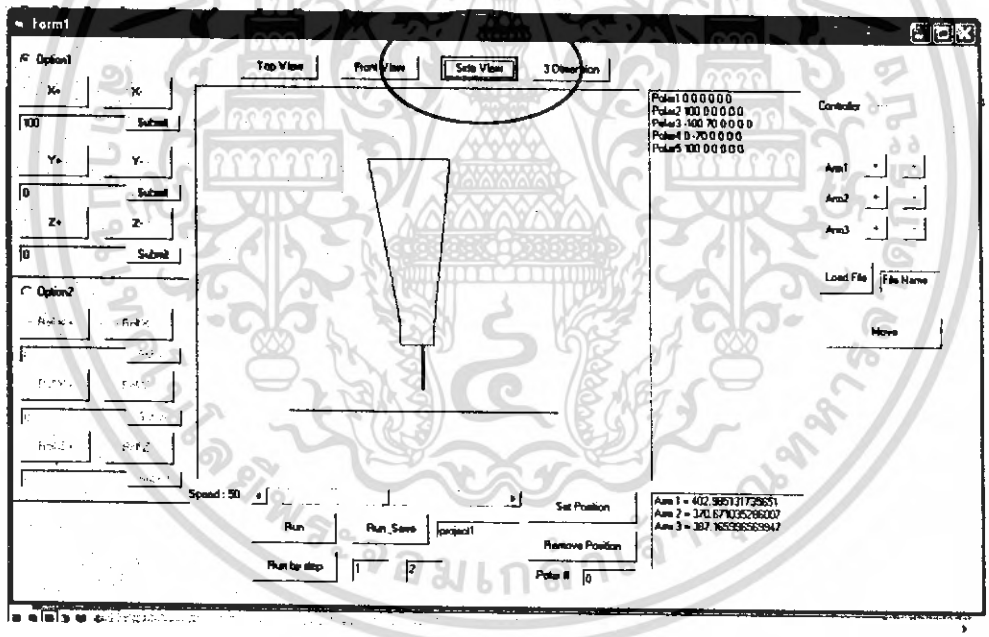
8. ผู้ใช้ยังสามารถเลือกมุมมองในการจำลองการเคลื่อนที่ได้ทั้ง 3 มุมมองคือ top front side และมุมมองในลักษณะที่เป็น 3 มิติ



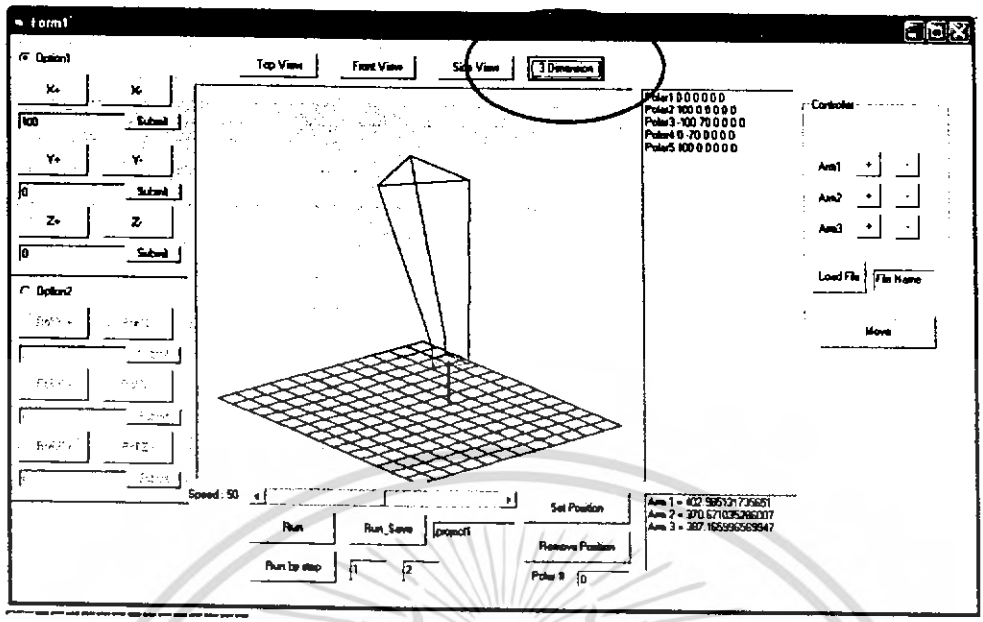
รูปที่ 4.16 แสดงมุมมอง top view



รูปที่ 4.17 แสดงมุมมอง front view



รูปที่ 4.18 แสดงมุมมอง side view

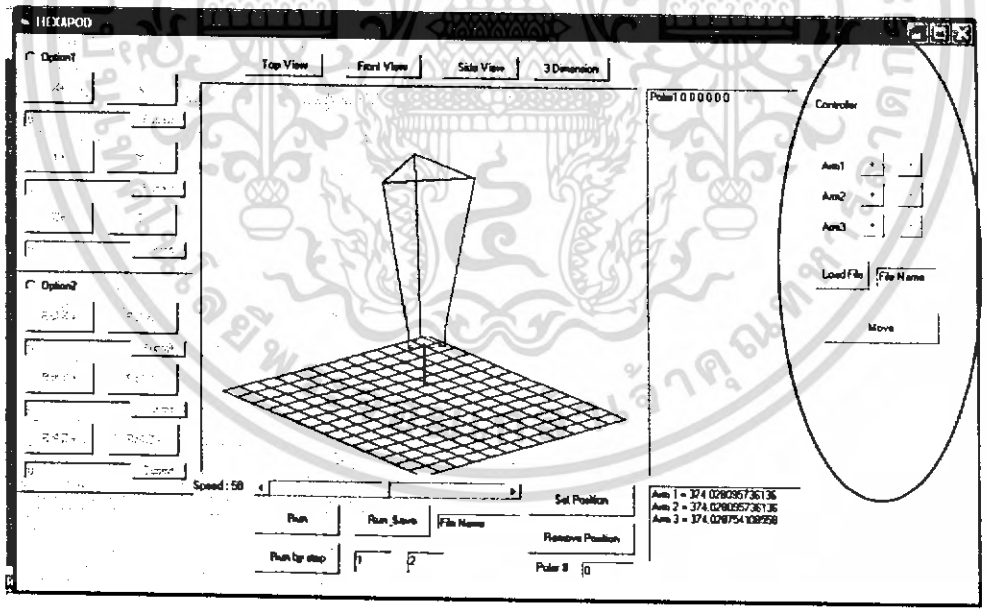


รูปที่ 4.19 แสดงมุมมองในลักษณะที่เป็น 3 มิติ

4.3.2 ผลการดำเนินงานด้านโปรแกรมที่ใช้ควบคุมการทำงานของแบบจำลอง

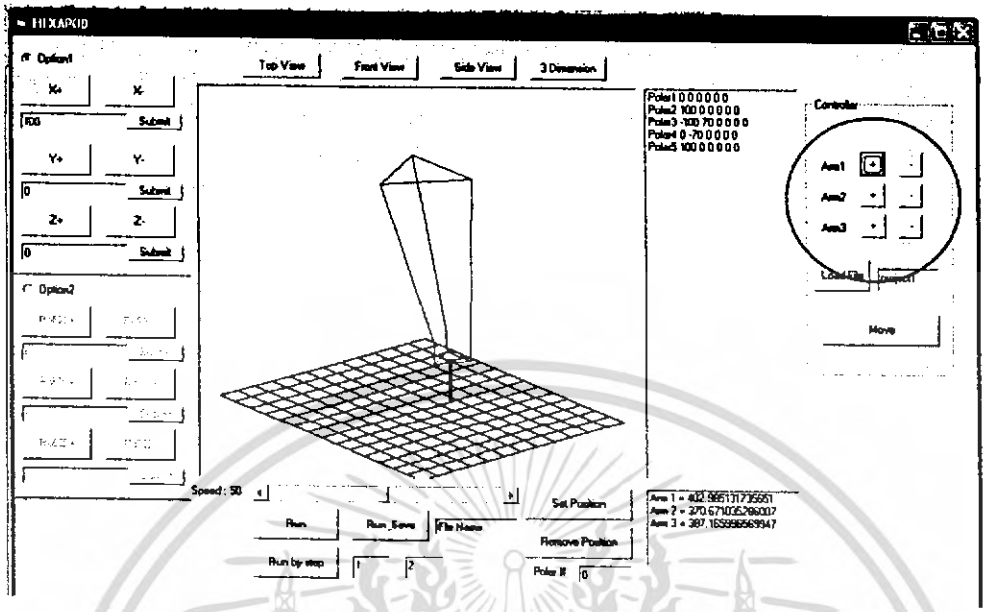
รายละเอียดโปรแกรมที่ใช้ควบคุมแบบจำลองเสกระพอดมีดังนี้

1. เริ่มต้นโปรแกรม



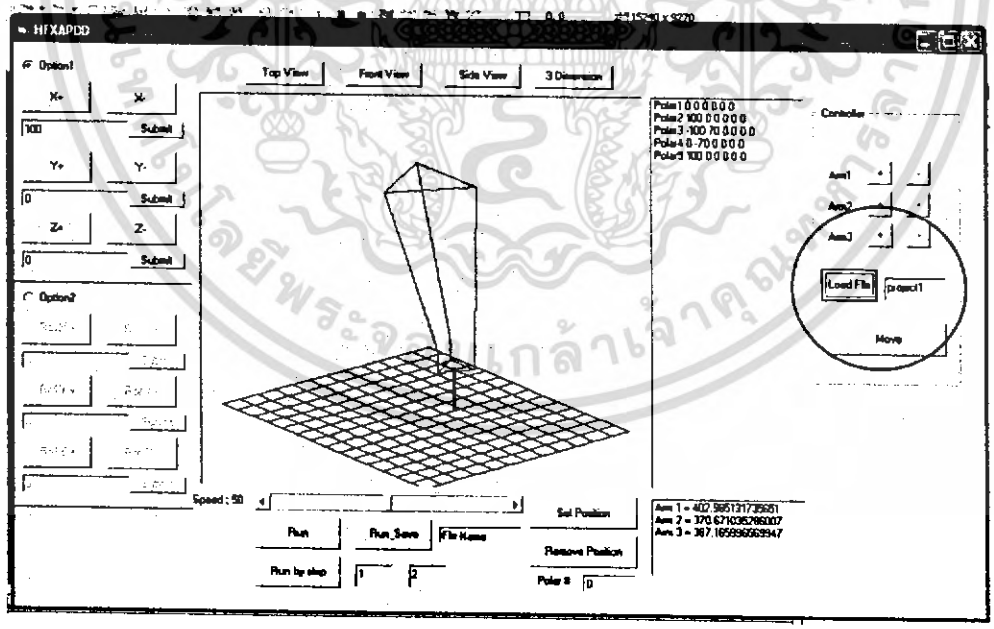
รูปที่ 4.20 แสดงหน้าจอโปรแกรมในส่วนควบคุม

2. ผู้ใช้สามารถควบคุมระยะยืด-หด ของแขนทั้ง 3 แขนได้ โดยใช้คำสั่ง Arm1 Arm2 และ Arm3



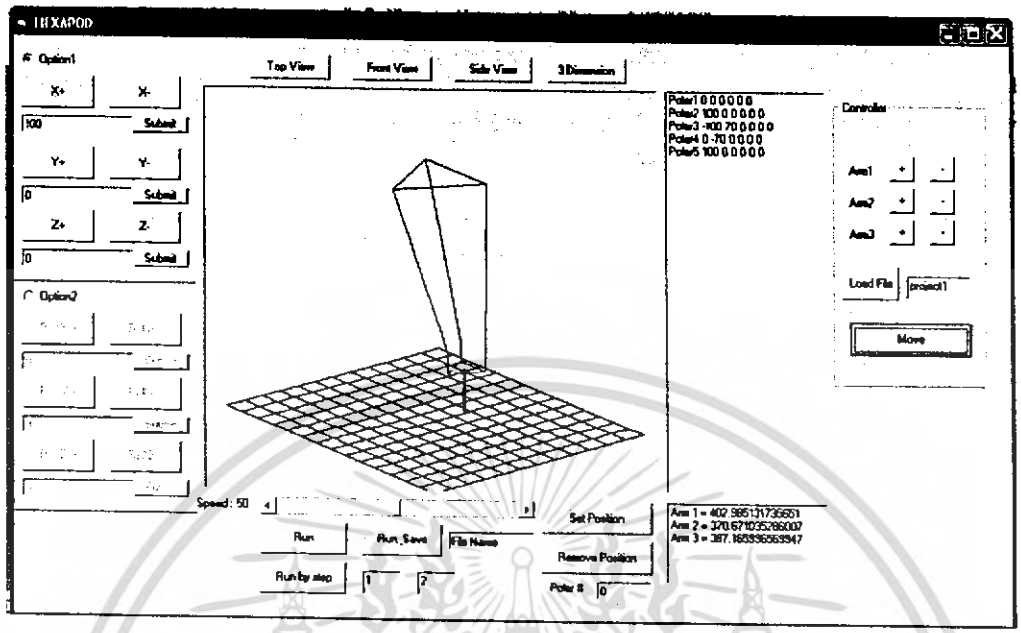
รูปที่ 4.21 แสดงการใช้คำสั่ง Arm1

3. ผู้ใช้สามารถโหลดไฟล์ที่ได้ทำการจำลองการเคลื่อนที่เพื่อนำมาใช้ควบคุมแบบจำลอง โดยพิมพ์ชื่อไฟล์ แล้วกด Load File

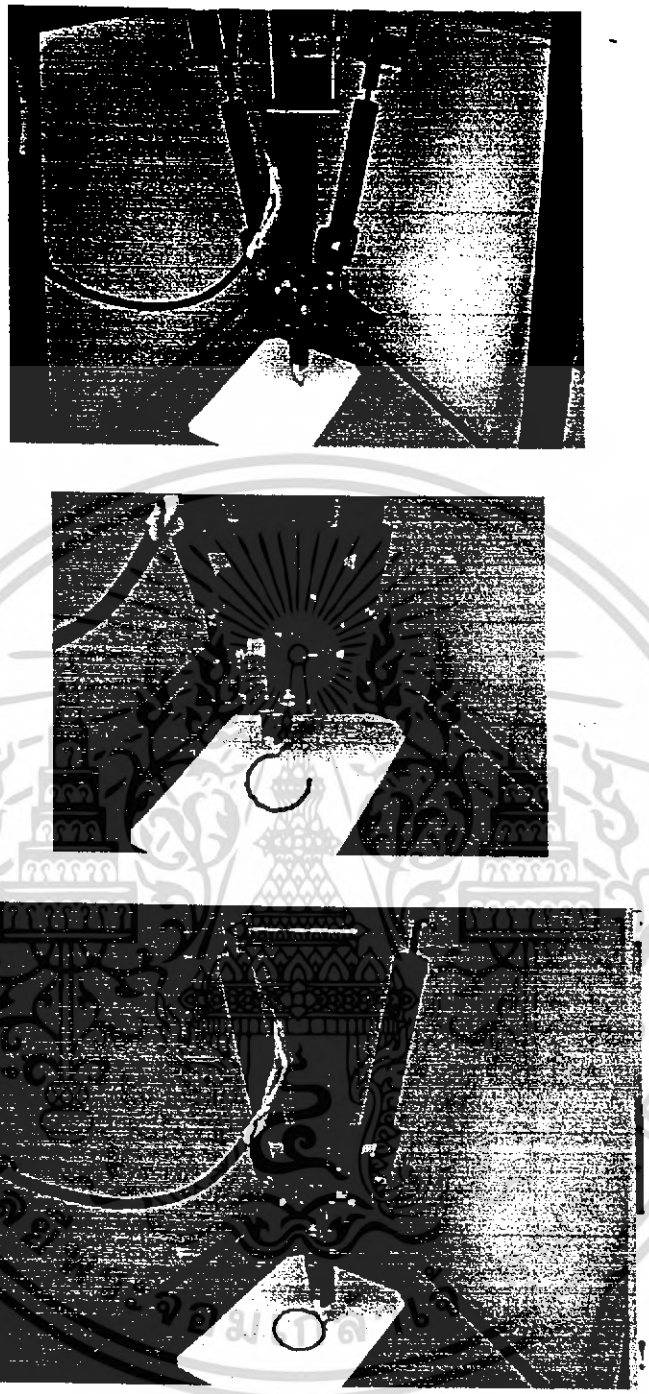


รูปที่ 4.22 แสดงการโหลดไฟล์ project1

4. ทำการสั่งให้แบบจำลองเคลื่อนที่ตามที่ต้องการ เลือกคำสั่ง Move



รูปที่ 4.23 แสดงการจำลองการเคลื่อนที่



รูปที่ 4.24 แสดงการสั่งงานให้จุดปลายของแบบจำลองเคลื่อนที่เป็นรูปวงกลม

บทที่ 5
สรุปผลและวิเคราะห์การดำเนินงาน

5.1 สรุปผลและวิเคราะห์การดำเนินงาน

โครงงานนี้เป็นการเขียนโปรแกรมเพื่อควบคุมและจำลองการเคลื่อนที่ของจุดปลายของเฮกซะพอด ซึ่งผู้ใช้ต้องทำการป้อนค่าตำแหน่งต่างๆที่ต้องการเข้ามา แล้วตัวโปรแกรมจะแสดงภาพจำลองการเคลื่อนที่ ทำให้เราทราบตำแหน่งของจุดปลายสุดของเฮกซะพอด แล้วจึงทำการสั่งงานควบคุมให้แบบจำลองเฮกซะพอดเคลื่อนที่ไปตามตำแหน่งที่ได้กำหนดไว้ ซึ่งสิ่งที่ผู้วิจัยสนใจมากที่สุดคือ ลักษณะการเคลื่อนที่จากตัวโปรแกรมควบคุมสั่งงานให้มอเตอร์ทำงานอย่างสัมพันธ์กันทั้ง 3 ตัวและทำให้เกิดการเคลื่อนที่ของจุดปลายเป็นรูปร่างต่างๆได้หรือไม่ จากการทดลองพบว่าลักษณะการเคลื่อนที่ของจุดปลายของแบบจำลองเฮกซะพอด สามารถเคลื่อนที่ได้เป็นรูปร่างตามที่ป้อนค่าเข้ามาได้ แต่จะพบว่ามีความผิดพลาดของเส้นทางที่เคลื่อนที่เล็กน้อย ถึงอย่างไรผลการทดลองที่ได้ถือเป็นที่น่าพอใจ หากแต่ต้องทำการปรับปรุงเมื่อต้องการให้มีการเคลื่อนที่ที่มีความแม่นยำกว่านี้ ซึ่งผู้ใดสนใจก็สามารถนำปริญญานิพนธ์ฉบับนี้ไปใช้ประกอบการศึกษา ออกแบบ และพัฒนาต่อไปได้

5.2 ข้อดีของแบบจำลองเฮกซะพอดและโปรแกรมจำลองการเคลื่อนที่พร้อมควบคุมการทำงาน

จากการศึกษาและดำเนินงานพบว่าข้อดีของแบบจำลองเฮกซะพอดและโปรแกรมจำลองการเคลื่อนที่พร้อมควบคุมการทำงาน มีดังนี้

1. สามารถใช้แบบจำลองนี้ในการศึกษาลักษณะการเคลื่อนที่ของโครงสร้างแบบเฮกซะพอดได้
2. การควบคุมการเคลื่อนที่ของแบบจำลองโครงสร้างเฮกซะพอดที่สร้างขึ้นนี้ มีความละเอียดของการเคลื่อนที่สูง
3. ตัวโปรแกรมและแบบจำลองมีความสามารถในการทำซ้ำ (repeatability) ของตำแหน่งที่เคลื่อนที่สูง
4. ตัวโปรแกรมที่ใช้จำลองการเคลื่อนที่และควบคุมแบบจำลองสามารถทำความเข้าใจการใช้งานง่าย
5. การจำลองการเคลื่อนที่ในโปรแกรมนั้น สามารถให้มุมมองในการจำลองที่หลากหลาย ไม่ว่าจะเป็นมุมมอง top view, front view, side view และมุมมอง 3 มิติ

5.3 ข้อเสียของแบบจำลองเฮกซะพอดและโปรแกรมจำลองการเคลื่อนที่พร้อมควบคุมการทำงาน

อย่างไรก็ตาม จากการศึกษาและดำเนินงานพบว่าแบบจำลองเฮกซะพอดและโปรแกรมจำลองการเคลื่อนที่พร้อมควบคุมการทำงานยังมีข้อจำกัดในการใช้งาน คือ

1. ต้องทำการออกแบบและทำโครงสร้างและส่วนประกอบต่างๆของเฮกซะพอดเอง เนื่องจากไม่มีชิ้นส่วนสำเร็จรูป และโครงสร้างเฮกซะพอดนี้ยังไม่เป็นที่แพร่หลายมากนัก

2. มีข้อจำกัดในการใช้งานของโปรแกรมโปรแกรมในส่วนของการควบคุมการเคลื่อนที่ ไม่สามารถปฏิบัติงานบนระบบ Windows XP ได้ เนื่องจากความไม่สอดคล้องกันของระบบส่งผ่านข้อมูล จึงจำกัดการใช้งานได้เพียง Windows 98 และ Windows ME เท่านั้น แต่โปรแกรมในส่วนของการจำลองการเคลื่อนที่นั้น ไม่มีข้อจำกัดในการใช้งาน
3. ความแม่นยำในการเคลื่อนที่

5.4 ข้อเสนอแนะและแนวทางการพัฒนา

จากการจัดทำปริญญานิพนธ์นี้สามารถสรุปข้อเสนอแนะ แนวทางการพัฒนาและปรับปรุงโครงงานทั้งในส่วนของโครงสร้างแบบจำลองและโปรแกรม

5.4.1 ข้อเสนอแนะและแนวทางการพัฒนาในส่วนของโครงสร้างแบบจำลอง

1. ในการทำแบบจำลองโครงสร้างเฮกซะพอดนั้นทางผู้วิจัยได้จำลองให้มี 3 แขน เพื่อศึกษาลักษณะการเคลื่อนที่เนื่องจากการที่องศาอิสระ (Degree of Freedom) ของแบบจำลองเหลือ จึงส่งผลให้เกิดผลกระทบจากการเปลี่ยนตำแหน่งของจุดศูนย์ถ่วงของจุดปลายขึ้น จึงควรออกแบบโครงสร้างให้มี 6 แขน เพื่อป้องกันปัญหาที่อาจเกิดขึ้น
2. พัฒนาให้สามารถทำการควบคุมการทำงานของข้อต่อให้ครบทุกข้อต่อ เพื่อที่จะสามารถศึกษาการเคลื่อนที่และการหมุนของจุดปลายของเฮกซะพอดได้
3. จัดทำตัววัดบอกขนาดและระยะทางตามแนวแกนต่างๆไว้บนตัวโครงสร้าง เพื่อบอกระยะทางและรูปร่างที่จุดปลายเคลื่อนที่ไปยังตำแหน่งต่างๆ

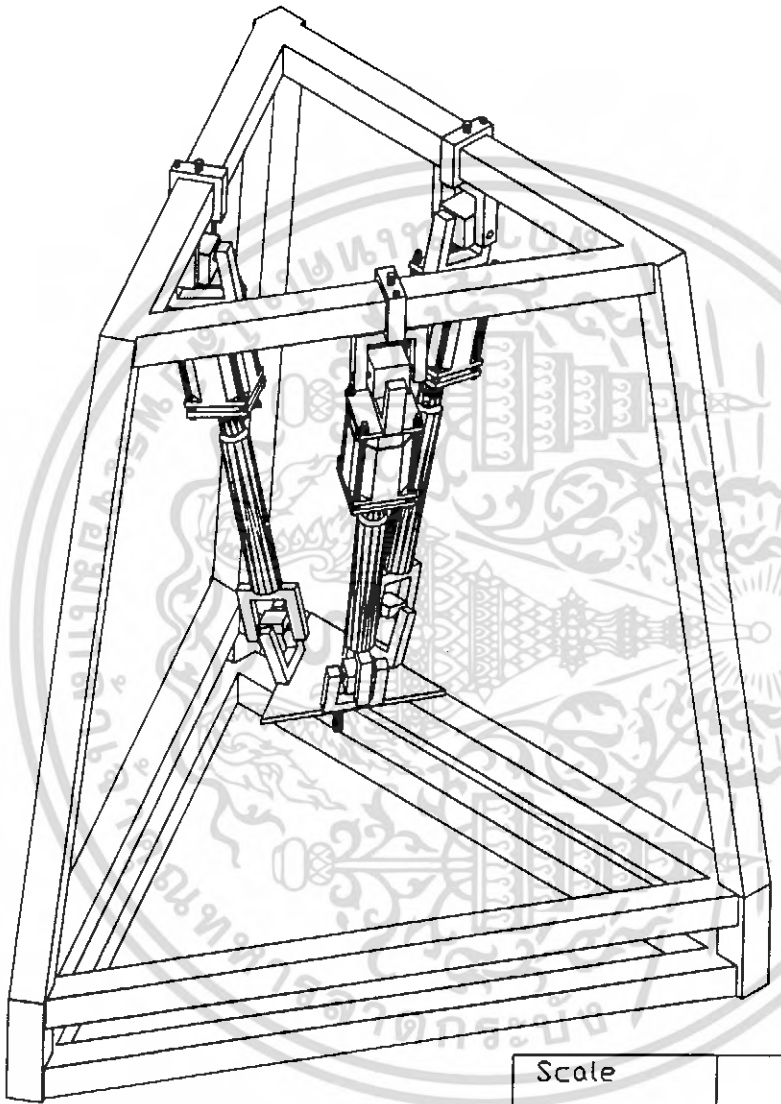
5.4.2 ข้อเสนอแนะและแนวทางการพัฒนาในส่วนของโปรแกรม

1. ออกแบบโปรแกรมให้มีการสื่อสารกับผู้ใช้ คือจัดทำ แถบคำสั่ง (Manu Bar) เพื่อความสะดวกในใช้งานและการศึกษาด้วยตนเอง
2. พัฒนาให้โปรแกรมสามารถจำลองการเคลื่อนที่จากไฟล์ AutoCAD หรือจากสมการเส้นทางการเคลื่อนที่

หนังสืออ้างอิง

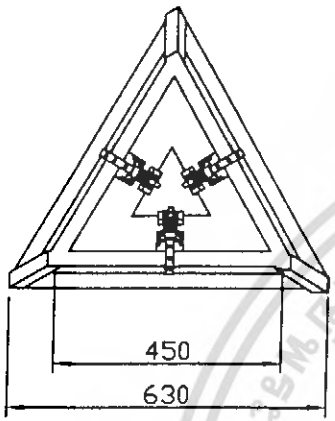
- บริษัทซัคเซส มิเดีย , 2547. Microsoft Visual Basic 6.0 กรุงเทพฯ : ซัคเซส มิเดีย
- W Khalil and E Dombre, 2004. Modeling Identification and Control of Robots. : Kogan Page Science
- Thomas R. Kurfess, 2004. Robotics and Automation Handbook. : CIC press
- Yoshihito Nakamura, 1992. Advanced Robotics Redundancy and Optimization. University of California Santa Barbara : Addison-Wesley Publishing Company
- RK Mital and IJ Nagrath, 2004. Robotics and Control. : Mcgraw-Hill Publishing Company Limited
- http://www.hexapods.net/
- http://www.i-way.co.uk/~storrs/lme/LMEHexapodMachine.html
- http://mfgshop.sandia.gov/1400_ext_Hexapod.htm
- http://www.datasheetcatalog.com

ภาคผนวก ก
แบบโครงสร้างและส่วนประกอบของ
แบบจำลองโครงสร้างเฮกซะพอด

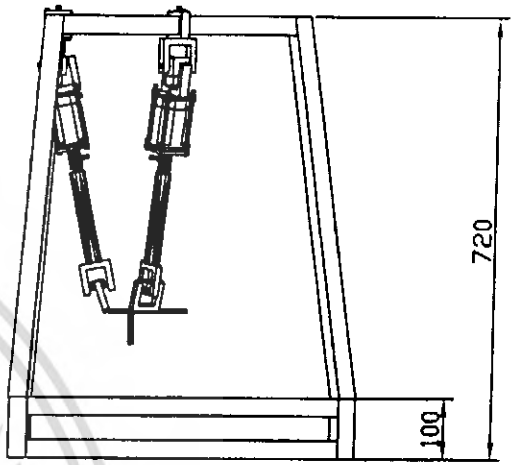


Scale 1 : 6 (mm.)	HEXAPOD
	Isometric : Hexapod Structure
1 / 6	BY Mr. Chitchai Jarunratanakul Miss Peeraporn Chongchirasiri Miss Mariam Sirikul
King Mongkut's Institute of Technology Ladkrabang	

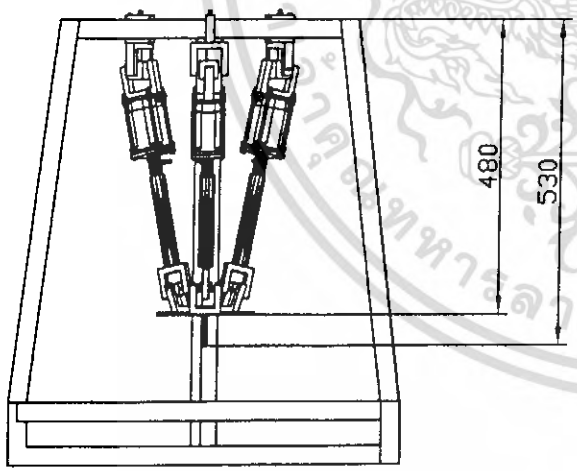
A4 2



Top View

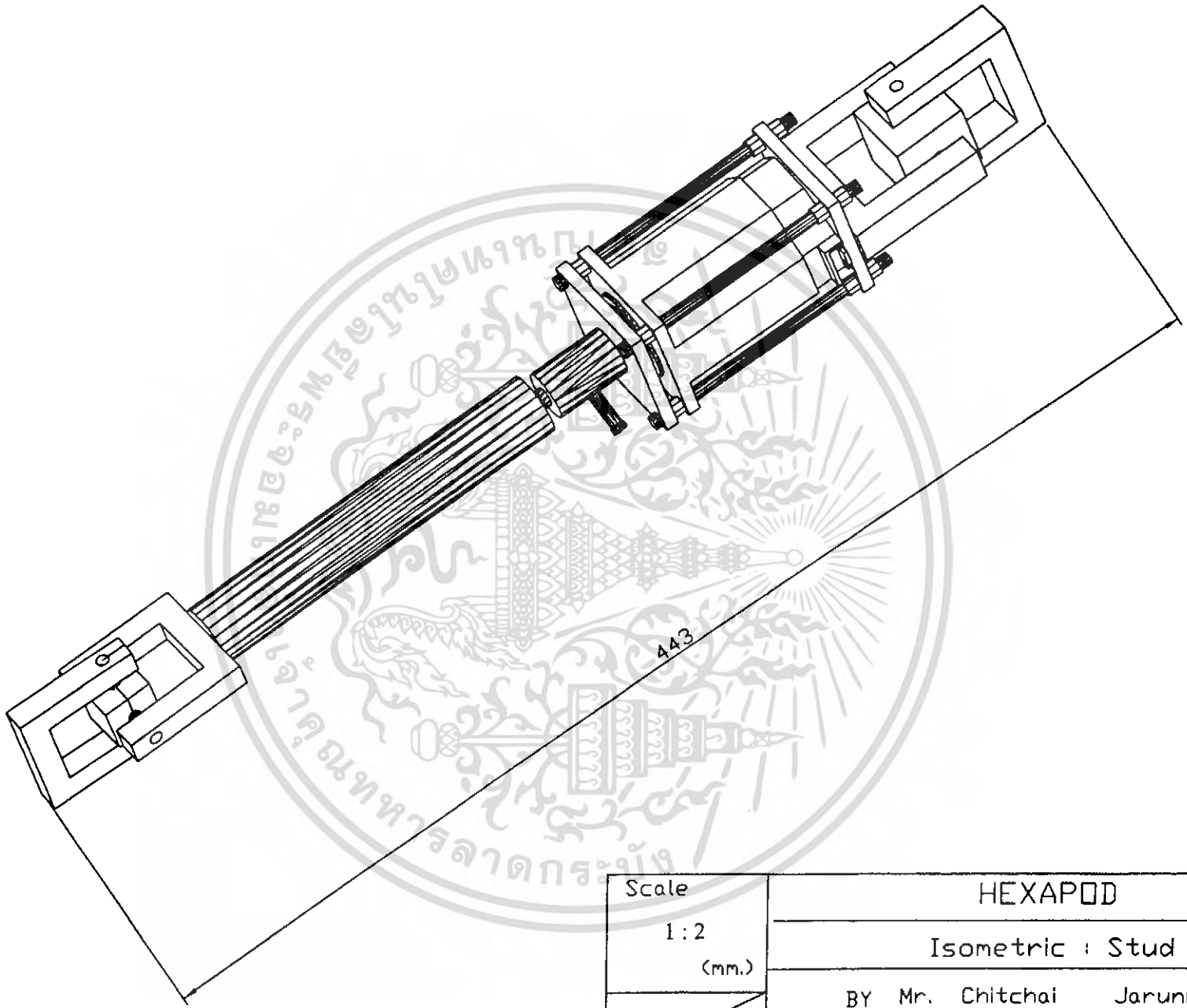


Side View



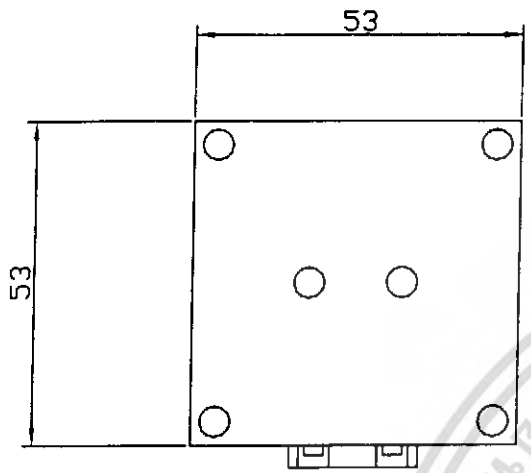
Front View

Scale 1 : 12 (mm.)	HEXAPOD
2 / 6	3 View : Hexapod Structure
	BY Mr. Chitchai Jarunratanakul Miss Peeraporn Chongchirasiri Miss Mariam Sirikul
King Mongkut's Institute of Technology Ladkrabang	

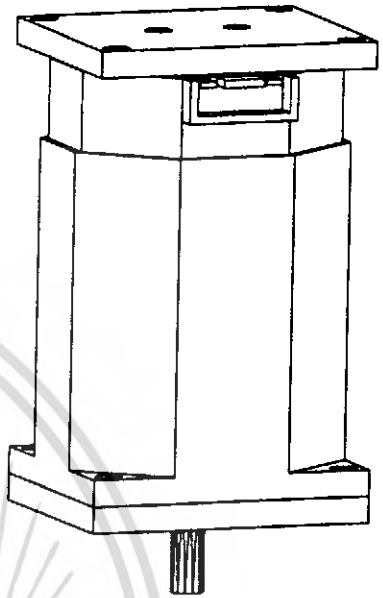


Scale 1 : 2 (mm.)	HEXAPOD
	Isometric : Stud
3 / 6	BY Mr. Chitchai Jarunratanakul Miss Peeraporn Chongchirasiri Miss Mariam Sirikul
King Mongkut's Institute of Technology Ladkrabang	

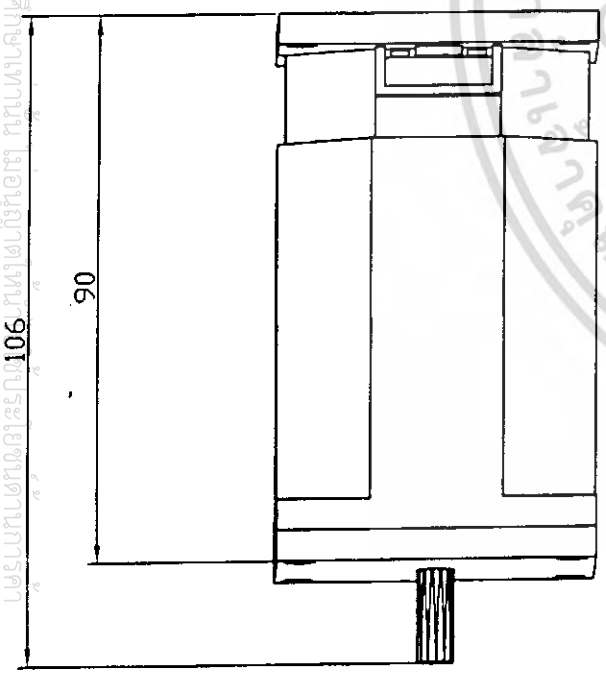
A4



Top View

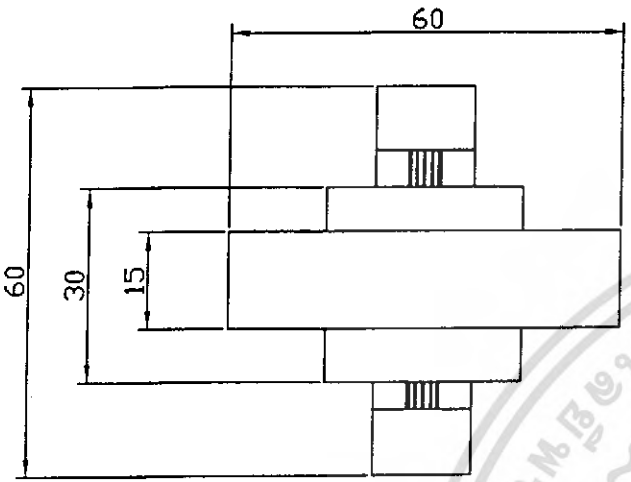


Isometric

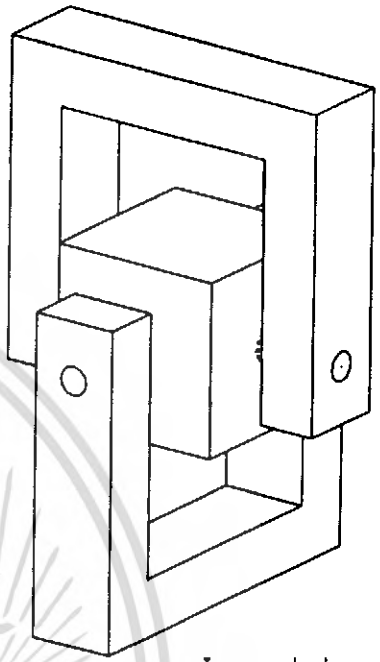


Front View

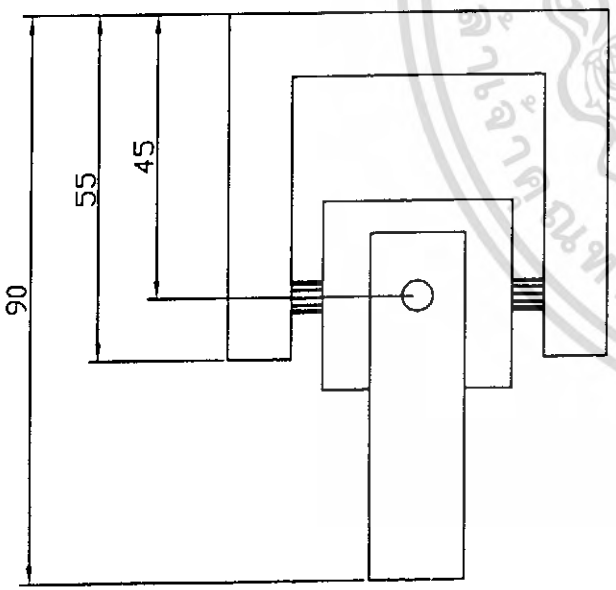
Scale 1 : 1.5 (mm.)	HEXAPOD	
	Detail : Step Motor	
4 / 6	BY Mr. Chitchai Jarunratanakul Miss Peeraporn Chongchirasiri Miss Mariam Sirikul	
King Mongkut's Institute of Technology Ladkrabang		



Top View

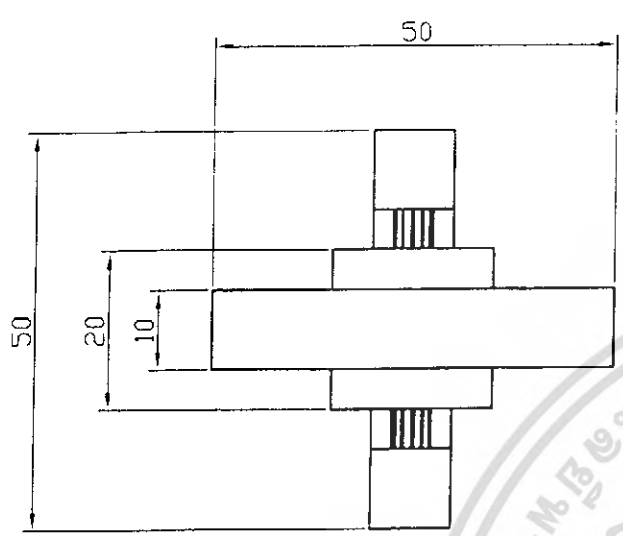


Isometric

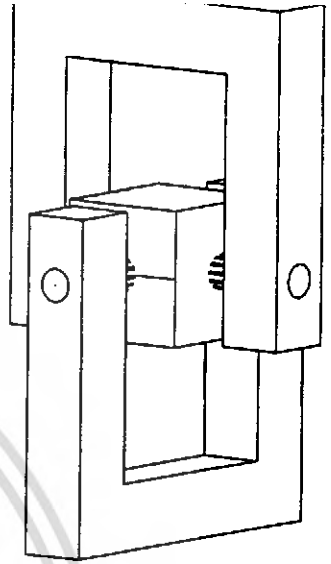


Front View

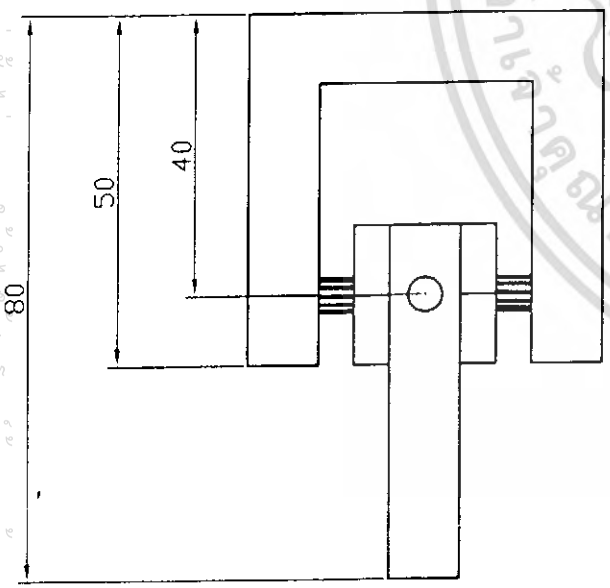
Scale 1 : 1.2 (mm.)	HEXAPOD	
	Detail : Joint 1	
5 / 6	BY Mr. Chitchai Jarunratanakul Miss Peeraporn Chongchirasiri Miss Mariam Sirikul	
King Mongkut's Institute of Technology Ladkrabang		



Top View



Isometric



Front View

Scale 1 : 1 (mm.)	HEXAPOD
	Detail : Joint 2
6 / 6	BY Mr. Chitchai Jarunratanakul Miss Peeraporn Chongchirasiri Miss Mariam Sirikul
King Mongkut's Institute of Technology Ladkrabang	

ภาคผนวก ข

เอกสารประกอบไอซี

(Datasheet)

Order this document by ULN2803/D

MOTOROLA

Octal High Voltage, High Current Darlington Transistor Arrays

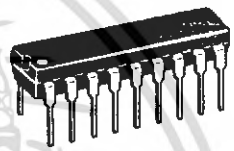
ULN2803
ULN2804

OCTAL PERIPHERAL DRIVER ARRAYS

SEMICONDUCTOR TECHNICAL DATA

The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open–collector outputs and free wheeling clamp diodes for transient suppression.

The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 6 to 15 volt high level CMOS or PMOS.



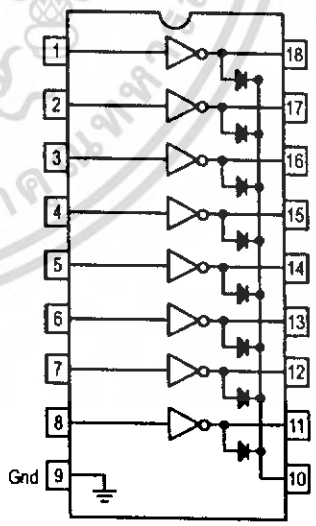
A SUFFIX
PLASTIC PACKAGE
CASE 707

MAXIMUM RATINGS (T_A = 25°C and rating apply to any one device in the package, unless otherwise noted.)

Rating	Symbol	Value	Unit
Output Voltage	V_O	50	V
Input Voltage (Except ULN2801)	V_I	30	V
Collector Current – Continuous	I_C	500	mA
Base Current – Continuous	I_B	25	mA
Operating Ambient Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C
Junction Temperature	T_J	125	°C

$R_{\theta JA}$ = 55°C/W
Do not exceed maximum current limit per driver.

PIN CONNECTIONS



ORDERING INFORMATION

Device	Characteristics		
	Input Compatibility	$V_{CE(Max)}/I_{C(Max)}$	Operating Temperature Range
ULN2803A ULN2804A	TTL, 5.0 V CMOS 6 to 15 V CMOS, PMOS	50 V/500 mA	T_A = 0 to + 70°C

ELECTRICAL CHARACTERISTICS (T_A = 25°C, unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Output Leakage Current (Figure 1)		I_{CEX}				µA
(V_O = 50 V, T_A = +70°C)	All Types		–	–	100	
(V_O = 50 V, T_A = +25°C)	All Types		–	–	50	
(V_O = 50 V, T_A = +70°C, V_I = 6.0 V)	ULN2802		–	–	500	
(V_O = 50 V, T_A = +70°C, V_I = 1.0 V)	ULN2804		–	–	500	
Collector–Emitter Saturation Voltage (Figure 2)		$V_{CE(sat)}$				V
(I_C = 350 mA, I_B = 500 µA)	All Types		–	1.1	1.6	
(I_C = 200 mA, I_B = 350 µA)	All Types		–	0.95	1.3	
(I_C = 100 mA, I_B = 250 µA)	All Types		–	0.85	1.1	
Input Current – On Condition (Figure 4)		$I_{I(on)}$				mA
(V_I = 17 V)	ULN2802		–	0.82	1.25	
(V_I = 3.85 V)	ULN2803		–	0.93	1.35	
(V_I = 5.0 V)	ULN2804		–	0.35	0.5	
(V_I = 12 V)	ULN2804		–	1.0	1.45	
Input Voltage – On Condition (Figure 5)		$V_{I(on)}$				V
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2802		–	–	13	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2803		–	–	2.4	
(V_{CE} = 2.0 V, I_C = 250 mA)	ULN2803		–	–	2.7	
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2803		–	–	3.0	
(V_{CE} = 2.0 V, I_C = 125 mA)	ULN2804		–	–	5.0	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2804		–	–	6.0	
(V_{CE} = 2.0 V, I_C = 275 mA)	ULN2804		–	–	7.0	
(V_{CE} = 2.0 V, I_C = 350 mA)	ULN2804		–	–	8.0	
Input Current – Off Condition (Figure 3) (I_C = 500 µA, T_A = +70°C)	All Types	$I_{I(off)}$	50	100	–	µA
DC Current Gain (Figure 2) (V_{CE} = 2.0 V, I_C = 350 mA)	ULN2801	h_{FE}	1000	–	–	–
Input Capacitance		C_I	–	15	25	pF
Turn–On Delay Time (50% E_I to 50% E_O)		t_{on}	–	0.25	1.0	µs
Turn–Off Delay Time (50% E_I to 50% E_O)		t_{off}	–	0.25	1.0	µs
Clamp Diode Leakage Current (Figure 6) (V_R = 50 V)	T_A = +25°C T_A = +70°C	I_R	–	–	50 100	µA
Clamp Diode Forward Voltage (Figure 7) (I_F = 350 mA)		V_F	–	1.5	2.0	V

TEST FIGURES

(See Figure Numbers in Electrical Characteristics Table)

Figure 1.

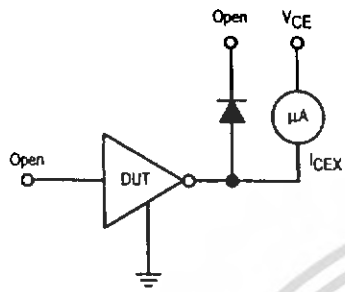


Figure 2.

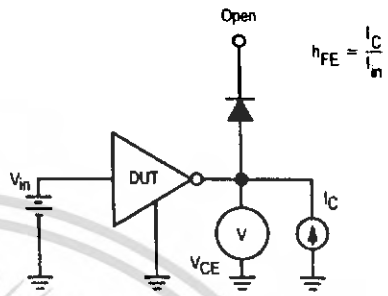


Figure 3.

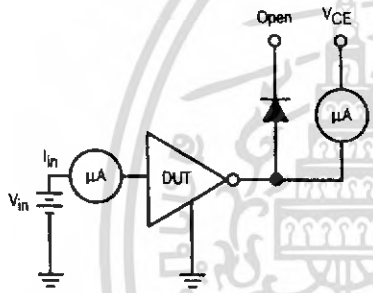


Figure 4.

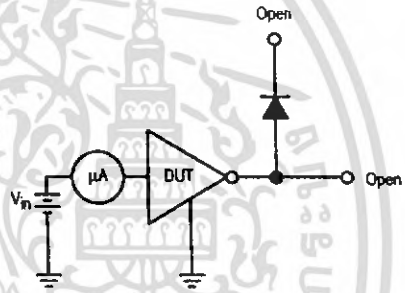


Figure 5.

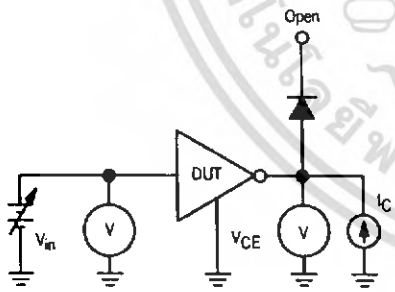


Figure 6.

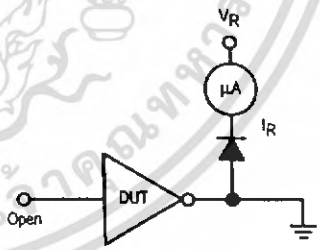
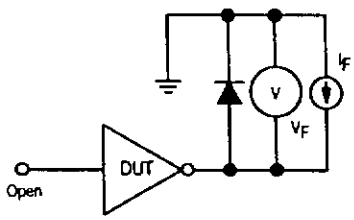


Figure 7.



TYPICAL CHARACTERISTIC CURVES – T_A = 25°C, unless otherwise noted

Output Characteristics

Figure 8. Output Current versus Saturation Voltage

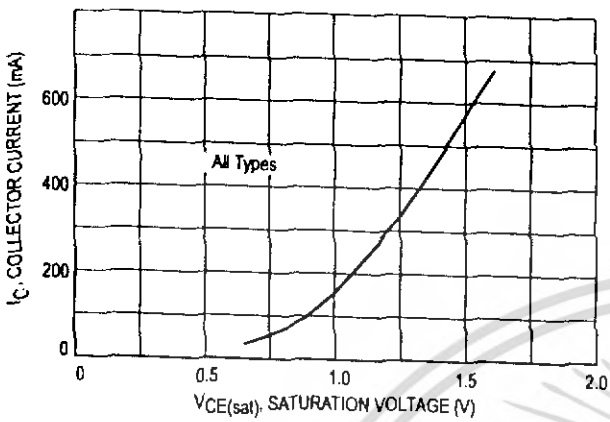
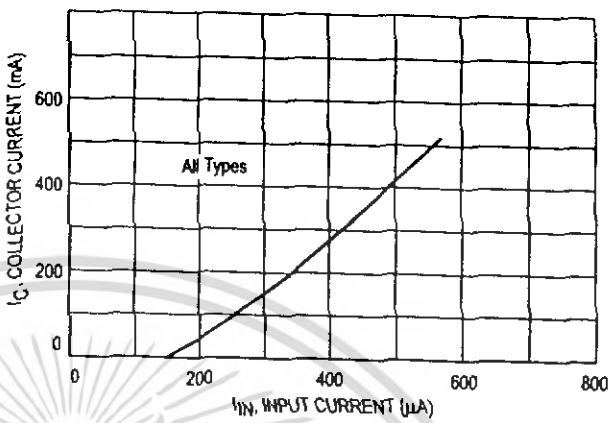


Figure 9. Output Current versus Input Current



Input Characteristics

Figure 10. ULN2803 Input Current versus Input Voltage

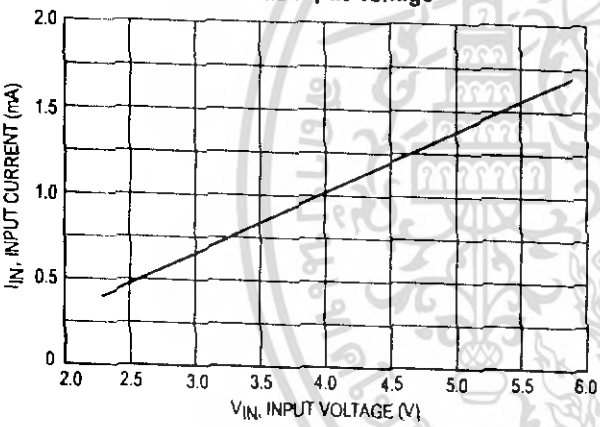


Figure 11. ULN2804 Input Current versus Input Voltage

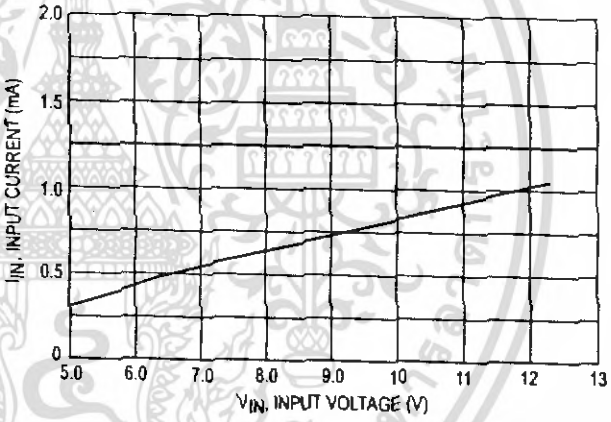
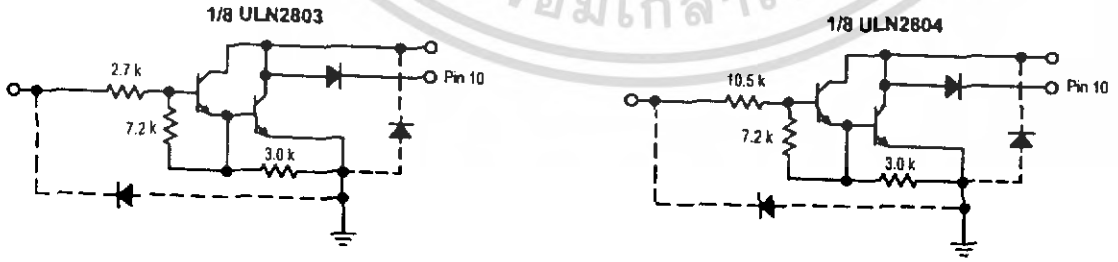
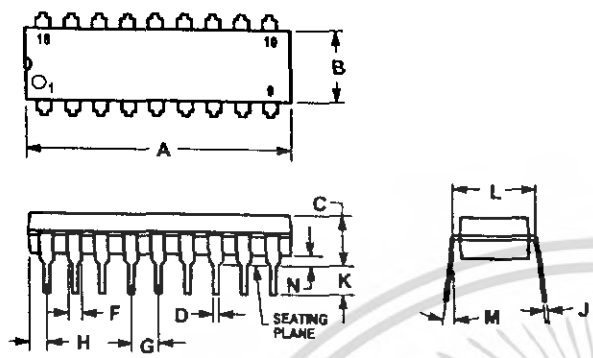


Figure 12. Representative Schematic Diagrams



OUTLINE DIMENSIONS

A SUFFIX
PLASTIC PACKAGE
CASE 707–02
ISSUE C



NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	22.22	23.24	0.875	0.915
B	6.10	6.60	0.240	0.260
C	3.56	4.57	0.140	0.180
D	0.36	0.56	0.014	0.022
F	1.27	1.78	0.050	0.070
G	2.54 BSC		0.100 BSC	
H	1.02	1.52	0.040	0.060
J	0.20	0.30	0.008	0.012
K	2.92	3.43	0.115	0.135
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447 or 602–303–5454

MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE 602–244–6609
INTERNET: http://Design–NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi–SPD–JLDC, 6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–81–3521–8315

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298

PACIFIC SCIENTIFIC
AUTOMATION TECHNOLOGY GROUP

MOTION TECHNOLOGY DIVISION
110 Fordham Road
Wilmington, MA 01887
(978) 988-9800
Fax (978) 988-9940

Part# MA6410
List Price $25 U.S.
October, 1998
Rev E

MA6410

6410 Drive

Installation & Hardware Reference Manual

Pacific Scientific Model 6410
24-75 Volt, 5A rms Microstepping Stepper Drive

FEATURES

- **Single power supply**
- **Patented 4-phase Bipolar Chopper Drive for superior current regulation and low ripple current**
- **Output current adjustable from 0.625 A to 5 A rms with 3 position dipswitch**
- **Microstepping for smooth operation and increased resolution.**
- **Patented digital electronic damping reduces instability at speeds in middle of operating range.**
- **Idle current reduction to reduce motor heating in many applications**
- **Fault protection:**
  - Line-to-line and line-to-neutral shorts
  - Internal power supply under-voltage
  - Bus overvoltage
- **Optically isolated command interface:**
  - Step
  - Direction
  - Enable
  - Enabled output
- **Selectable step filter**
  - Rejection of electrical noise on step input
- **Small size** - Only 7.5 in^2 of panel space (5" x 1.5" x 4.3")
- **UL recognized**

APPLICATIONS

- **X-Y tables and slides**
- **Packaging machinery**
- **Robotics**
- **Specialty machinery**
- **Index feed of materials**
- **Labeling machines**

PRODUCT DESCRIPTION

The Pacific Scientific 6410 is a low cost, compact stepper drive converting step and direction inputs into winding currents for two-phase stepper motors.

Resolution with 1.8° motors is adjustable to 200, 400, 1000, 2000, 5000, 10,000, 25,000, or 50,000 steps per revolution with decimal step size selected, and 400, 800, 1600, 3200, 6400, 12800, 25600, or 51200 steps per revolution with binary step size selected. Higher resolution (microstepping) provides smoother operation through resonance regions as well as increased position resolution.

The 6410 operates from a single supply voltage ranging from 24 Vdc to 75 Vdc.

The default output current is 5A rms. The current can be reduced in increments of 0.625 A using a 3 position DIP switch.

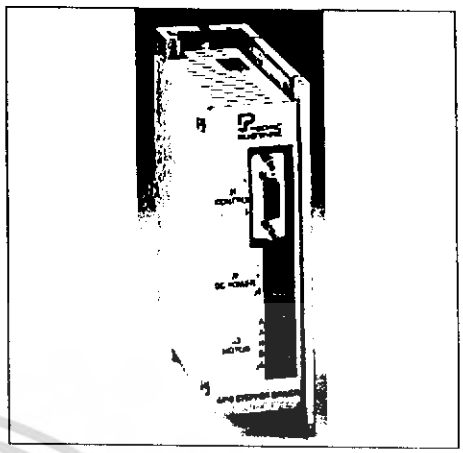
A patented 4-phase PWM (pulse width modulated) chopper electronically controls the motor winding currents at 20 KHz. This combines the best of recirculating and non-recirculating current regulation to provide high back EMF rejection with low ripple current. Benefits include reduced heat dissipation, low electrical noise and improved current control during dynamic braking.

The patented digital electronic damping eliminates motor oscillations common with stepper systems at speeds in the middle of the operating region.

This, along with 4-phase chopping gives significantly more motor output power than competitive systems.

Idle current reduction can be used to reduce motor current by 50% if no step commands have been received for 0.1 second (0.05 and 1.0 seconds can also be selected). Current is restored to full amplitude upon arrival of a step command.

The 6410 uses just 7.5 in^2 of panel space. The optional side mounted heat sink increases width by 1.0" to 2.5". Heat can be removed either from the rear or side of the driver (cold plate mounting) allowing maximum flexibility for system packaging.



Pacific Scientific 6410

SPECIFICATIONS

Input Power

Voltage	24 Vdc to 75 Vdc
Current	Motor and load dependent. Usually < motor phase current.

Output motor phase current	5 A rms max. 5 A peak full step 7.1 A peak microstepping Adjustable from 0.625 to 5 A rms in 0.625 amp increments (See Figure 3)

Inputs (See Figures 1 and 2)

STEP

Optically isolated TTL compatible
Minimum opto current (opto on): 5.5 ma
Maximum opto current (opto on): 10 ma
Minimum pulse width: 250 ns (**1 µs**)
Maximum frequency: 2 MHz (**500 KHz**)
Motion occurs on low-to-high transition of STEP- input
Note: BOLD values indicate step filter enabled

DIR

Optically isolated TTL compatible
For normal motor connections:
Current in opto (opto on): Rotation CCW looking at motor shaft
Minimum opto current (opto on): 3 ma
Maximum opto current (opto on): 4.5 ma
Minimum setup time: 50.0 µs
Minimum hold time: zero

ENABLE

Optically isolated TTL compatible
Sense of ENABLE input can be changed using ENBL_SENSE jumper:
Jumper In: Current in opto (opto on) enables driver
Jumper Out: Current in opto (opto on) disables driver
Minimum opto current (opto on): 3 ma
Maximum opto current (opto on): 4.5 ma

SPECIFICATIONS (continued)

ıtput (See Figures 1 and 2)

ENABLED Optically isolated open collector, open emitter
Drive Enabled: opto transistor on,
V_{sat} = 0.5 V max. @ 2.0 ma
Drive Disabled: opto transistor off,
V_{cc} max. = 35 V

:p Size Set using 3 positions of DIP switch and decimal jumper (See Figure 3).

Step Size	Steps per Revolution (1.8° motor)
Full (1/2)	200 (400)
1/2 (1/4)	400 (800)
1/5 (1/8)	1000 (1600)
1/10 (1/16)	2000 (3200)
1/25 (1/32)	5000 (6400)
1/50 (1/64)	10000 (12800)
1/125 (1/128)	25000 (25600)
1/250 (1/256)	50000 (51200)

Note: Binary values are in BOLD

rent uction Enabled or disabled with DIP switch
50% output current reduction after 0.1 second from last step command.
(0.05 and 1.0 second time-outs can also be selected using a plug-on jumper. Consult factory for other current reduction options)
See Figure 3.

-band ibility pensation Enabled or disabled with DIP switch (See Figure 3)
Max. delay from input step to change in motor excitation:
Step frequency < 500 full steps/sec: 500 µs
Step frequency > 500 full steps/sec: 270° of step period

ction (Any fault disables the drive and must be cleared by cycling input power)
Line to Line Short

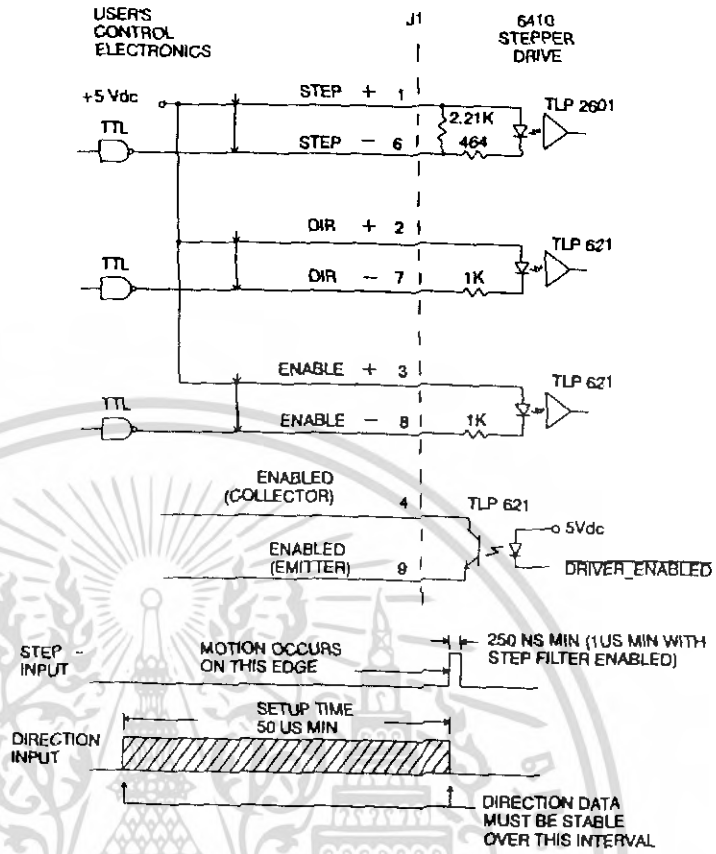


Figure 2 - Interface Circuits

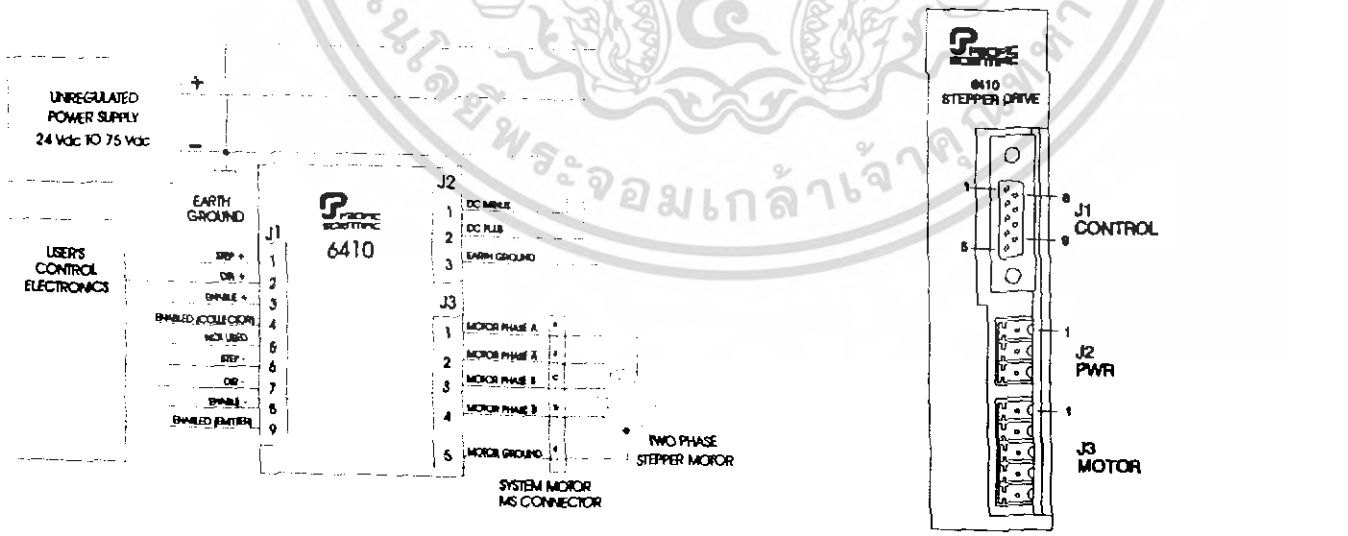


Figure 1 - 6410 Connection Diagram

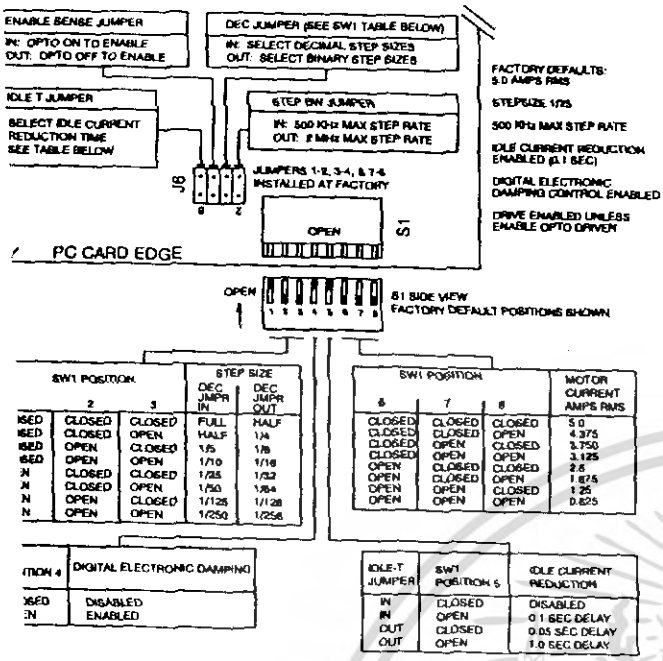


Figure 3 - Dip Switch Settings

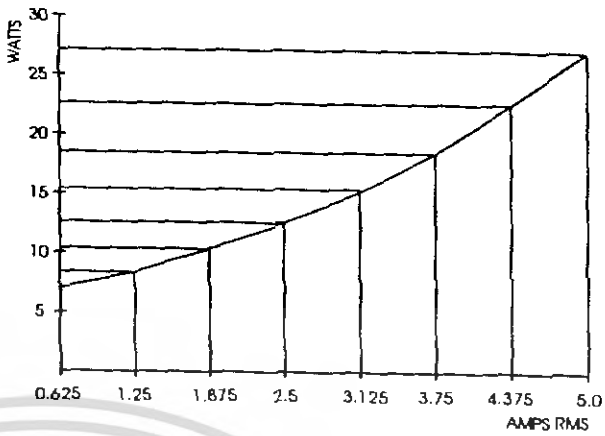


Figure 4 - Power Dissipation vs. Output Current

ronmental uirements	
rage nperature	-55°C to +70°C
erating nperature	0 to 50°C ambient air with or without cover
ximum asis nperature	60°C **Note:** *For optimal thermal performance, mount the 6410 chassis (back or side) to a cooling plate or heatsink. Use a themal pad or grease if surface is irregular. A fan or idle current reduction may be employed to keep chassis below 60°C.*
vection ling	(6410 not mounted on cooling plate)
/optional eat sink	Full rating (5A) at 25°C Ambient 2.5A max at 45°C Ambient
ithout eat sink	2.5A max at 25°C Ambient 1.25A max at 45°C Ambient
	See Figure 4 for plot of drive power dissipation vs. output current.
idity ge	10 to 90%, non-condensing
anical	
ensions	5" x 1.5" x 4.3" (without cover) 5" x 1.7" x 4.3" (with cover & screws)
ht	1 lb. nominal
ctors	(see Figure 1)
r Supply r	3 contact plug-in screw terminal 5 contact plug-in screw terminal
l	9 Socket D Sub miniature

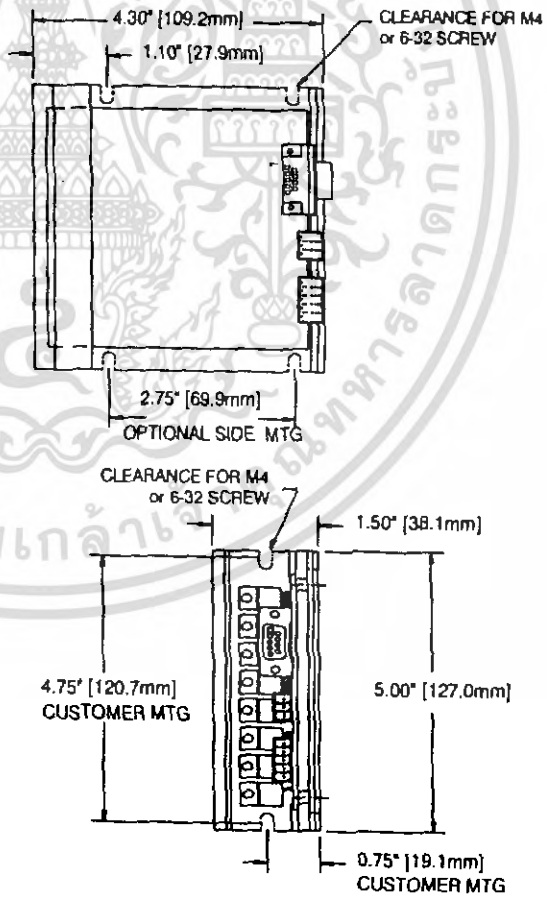


Figure 5 - Mounting Dimensions

ภาคผนวก ค
คำสั่งที่ใช้เขียนโปรแกรมที่ใช้จำลองการเคลื่อนที่และควบคุมการทำงาน

Hexapod (form1.frm)

```
Private Sub CmdFront_Click()
viewer = 2
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
Picture1.Visible = False
Picture2.Visible = False
Picture3.Visible = True
Picture4.Visible = False
End Sub
Private Sub Cmdside_Click()
viewer = 3
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
Picture1.Visible = False
Picture2.Visible = False
Picture3.Visible = False
Picture4.Visible = True
End Sub
Private Sub Cmdtop_Click()
viewer = 1
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
Picture1.Visible = False
Picture2.Visible = True
Picture3.Visible = False
Picture4.Visible = False
End Sub
```

```
Private Sub Command1_Click()
centere.x = centere.x + (BLU)
center.x = center.x + (BLU)
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + (BLU)
MovpointE(i).x = MovpointE(i).x + (BLU)
Next i
tx = centere.x
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub
Private Sub Command10_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
```

```
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dy = Dy - 1
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
thy = Dy
End Sub
Private Sub Command11_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
```

```
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dz = Dz + 1
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
```

```
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
thz = Dz
End Sub

Private Sub Command12_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
```

```
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dz = Dz - 1
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
thz = Dz
End Sub

Private Sub Command13_Click()
numPo = numPo + 1
co = co + 1
```

```
Polar(co).cex = centere.x
Polar(co).cey = centere.y
Polar(co).cez = centere.z - ZOS
Polar(co).cx = center.x
Polar(co).cy = center.y
Polar(co).cz = center.z - ZOS
Polar(co).rx = Dx
Polar(co).ry = Dy
Polar(co).rz = Dz
For i = 1 To 3
Polar(co).mov(i) = Movpoint(i)
Polar(co).move(i) = MovpointE(i)
Next i
save2 "Textbar"
For i = 1 To co
Print #1, "Polar" & i & "_____cex" & " "; Polar(i).cex
Print #1, "Polar" & i & "_____cey" & " "; Polar(i).cey
Print #1, "Polar" & i & "_____cez" & " "; Polar(i).cez
Print #1, "Polar" & i & "_____cx" & " "; Polar(i).cx
Print #1, "Polar" & i & "_____cy" & " "; Polar(i).cy
Print #1, "Polar" & i & "_____cz" & " "; Polar(i).cz
Print #1, "Polar" & i & "_____rx" & " "; Polar(i).rx
Print #1, "Polar" & i & "_____ry" & " "; Polar(i).ry
Print #1, "Polar" & i & "_____rz" & " "; Polar(i).rz
Print #1, "Polar" & i & "_mov(1)x" & " "; Polar(i).mov(1).x
Print #1, "Polar" & i & "_mov(1)y" & " "; Polar(i).mov(1).y
Print #1, "Polar" & i & "_mov(1)z" & " "; Polar(i).mov(1).z
Print #1, "Polar" & i & "_mov(2)x" & " "; Polar(i).mov(2).x
Print #1, "Polar" & i & "_mov(2)y" & " "; Polar(i).mov(2).y
Print #1, "Polar" & i & "_mov(2)z" & " "; Polar(i).mov(2).z
Print #1, "Polar" & i & "_mov(3)x" & " "; Polar(i).mov(3).x
Print #1, "Polar" & i & "_mov(3)y" & " "; Polar(i).mov(3).y
Print #1, "Polar" & i & "_mov(3)z" & " "; Polar(i).mov(3).z
Print #1, "Polar" & i & "move(1)x" & " "; Polar(i).move(1).x
Print #1, "Polar" & i & "move(1)y" & " "; Polar(i).move(1).y
Print #1, "Polar" & i & "move(1)z" & " "; Polar(i).move(1).z
```

```
Print #1, "Polar" & i & "move(2)x" & " "; Polar(i).move(2).x
Print #1, "Polar" & i & "move(2)y" & " "; Polar(i).move(2).y
Print #1, "Polar" & i & "move(2)z" & " "; Polar(i).move(2).z
Print #1, "Polar" & i & "move(3)x" & " "; Polar(i).move(3).x
Print #1, "Polar" & i & "move(3)y" & " "; Polar(i).move(3).y
Print #1, "Polar" & i & "move(3)z" & " "; Polar(i).move(3).z
Print #1,
Next i
Close
Hexapod.Lst1.Clear
For i = 1 To co
Hexapod.Lst1.AddItem "Polar" & i & " " & Polar(i).cex & " " & Polar(i).cey & " " & Polar(i).cez & " " & Polar(i).rx
& " " & Polar(i).ry & " " & Polar(i).rz
Next i
End Sub

Private Sub Command14_Click()
Dim diff As Double
diff = CDbl(tx) - centere.x
centere.x = centere.x + diff
center.x = center.x + diff
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + diff
MovpointE(i).x = MovpointE(i).x + diff
Next i
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command15_Click()
Dim diff As Double
diff = CDbl(tz) - centere.z + ZOS
centere.z = centere.z + diff
center.z = center.z + diff
```

```
For i = 1 To 3
Movpoint(i).z = Movpoint(i).z + diff
MovpointE(i).z = MovpointE(i).z + diff
Next i
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command16_Click()
viewer = 0
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
Picture1.Visible = True
Picture2.Visible = False
Picture3.Visible = False
Picture4.Visible = False
End Sub

Private Sub Command17_Click()
If (CInt(TRE) <= 0) Then
Else
Dim Cofake As Integer
co = co - 1
Cofake = TRE - 1
For i = 1 To co
Cofake = Cofake + 1
Polar(Cofake).cex = Polar(Cofake + 1).cex
Polar(Cofake).cey = Polar(Cofake + 1).cey
Polar(Cofake).cez = Polar(Cofake + 1).cez
Polar(Cofake).cx = Polar(Cofake + 1).cx
Polar(Cofake).cy = Polar(Cofake + 1).cy
Polar(Cofake).cz = Polar(Cofake + 1).cz
```

```
Polar(Cofake).rx = Polar(Cofake + 1).rx
Polar(Cofake).ry = Polar(Cofake + 1).ry
Polar(Cofake).rz = Polar(Cofake + 1).rz
Next i
For j = 1 To co
Cofake = Cofake + 1
For i = 1 To 3
Polar(Cofake).mov(i) = Polar(Cofake + 1).mov(i)
Polar(Cofake).move(i) = Polar(Cofake + 1).move(i)
Next i
Next j
save2 "Textbar"
For i = 1 To co
Print #1, "Polar" & i & "_____cex" & " "; Polar(i).cex
Print #1, "Polar" & i & "_____cey" & " "; Polar(i).cey
Print #1, "Polar" & i & "_____cez" & " "; Polar(i).cez
Print #1, "Polar" & i & "_____cx" & " "; Polar(i).cx
Print #1, "Polar" & i & "_____cy" & " "; Polar(i).cy
Print #1, "Polar" & i & "_____cz" & " "; Polar(i).cz
Print #1, "Polar" & i & "_____rx" & " "; Polar(i).rx
Print #1, "Polar" & i & "_____ry" & " "; Polar(i).ry
Print #1, "Polar" & i & "_____rz" & " "; Polar(i).rz
Print #1, "Polar" & i & "_mov(1)x" & " "; Polar(i).mov(1).x
Print #1, "Polar" & i & "_mov(1)y" & " "; Polar(i).mov(1).y
Print #1, "Polar" & i & "_mov(1)z" & " "; Polar(i).mov(1).z
Print #1, "Polar" & i & "move(1)x" & " "; Polar(i).move(1).x
Print #1, "Polar" & i & "move(1)y" & " "; Polar(i).move(1).y
Print #1, "Polar" & i & "move(1)z" & " "; Polar(i).move(1).z
Next i
Close
Hexapod.Lst1.Clear
For i = 1 To co
Hexapod.Lst1.AddItem "Polar" & i & " " & Polar(i).cex & " " & Polar(i).cey & " " & Polar(i).cez & " " & Polar(i).rx
& " " & Polar(i).ry & " " & Polar(i).rz
Next i
End If
```

```
End Sub

Private Sub Command18_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dx = thx
Dy = thy
Dz = thz
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
```

```
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command19_Click()
QuToRec = 0
Hexapod.Pic1.Cls
Hexapod.Pic1.AutoRedraw = False
Dim iii As Integer
For iii = 2 To numPo
   motion (iii - 1), iii
Next iii
Hexapod.Pic1.AutoRedraw = True
QuToRec = 0
Debug.Print numPo
End Sub

Private Sub Command2_Click()
```

```
centere.x = centere.x - (BLU)
center.x = center.x - (BLU)
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - (BLU)
MovpointE(i).x = MovpointE(i).x - (BLU)
Next i
tx = centere.x
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command20_Click()
Dim diff As Double
diff = CDbl(ty) - centere.y
centere.y = centere.y + diff
center.y = center.y + diff
For i = 1 To 3
Movpoint(i).y = Movpoint(i).y + diff
MovpointE(i).y = MovpointE(i).y + diff
Next i
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command21_Click(Index As Integer)
For ii = 1 To 10000
outport &H378, 1
For i = 1 To 1
Next i
outport &H378, 0
For i = 1 To 1
```

```
Next i
Next ii
End Sub

Private Sub Command212_Click(Index As Integer)
For ii = 1 To 10000
outport &H378, 1 + 2
For i = 1 To 1
Next i
outport &H378, 0
For i = 1 To 1
Next i
Next ii
End Sub

Private Sub Command213_Click(Index As Integer)
For ii = 1 To 10000
outport &H378, 4
For i = 1 To 1
Next i
outport &H378, 0
For i = 1 To 1
Next i
Next ii
End Sub

Private Sub Command214_Click(Index As Integer)
For ii = 1 To 10000
outport &H378, 4 + 8
For i = 1 To 1
Next i
outport &H378, 0
For i = 1 To 1
Next i
Next ii
End Sub
```

```
Private Sub Command215_Click(Index As Integer)
For ii = 1 To 10000
outport &H378, 16
For i = 1 To 1
Next i
outport &H378, 0
For i = 1 To 1
Next i
Next ii
End Sub

Private Sub Command216_Click(Index As Integer)
For ii = 1 To 10000
outport &H378, 16 + 32
For i = 1 To 1
Next i
outport &H378, 0
For i = 1 To 1
Next i
Next ii
End Sub

Private Sub Command22_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
```

```
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dx = thx
Dy = thy
Dz = thz
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
```

```
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command23_Click()
QuToRec = 1
stepper = 0
Hexapod.Pic1.Cls
Hexapod.Pic1.AutoRedraw = False
Dim iii As Integer
For iii = 2 To numPo
   motion (iii - 1), iii
Next iii
Hexapod.Pic1.AutoRedraw = True
QuToRec = 0
End Sub

Private Sub Command24_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
```

```
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dx = thx
Dy = thy
Dz = thz
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
```

```
drawarm
drawstru
End Sub

Private Sub Command25_Click()
QuToRec = 0
Hexapod.Pic1.Cls
Hexapod.Pic1.AutoRedraw = False
motion Hexapod.Text1, Hexapod.Text2
Hexapod.Pic1.AutoRedraw = True
center = Vba_cenTer
centere = Vba_cenTerE
For i = 1 To 3
Movpoint(i) = Vba_Movpoint(i)
MovpointE(i) = Vba_MovpointE(i)
Next i
Hexapod.Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command26_Click()
Dim n As Integer, m As Integer
n = 1
m = 10001
Open Text4 & ".txt" For Input As #2
Do
Input #2, rec0
If Left(rec0, 8) = "PointL1_" Then
If CInt(Mid(rec0, 9, 5)) = m Then
  MLength(n).L1 = CDbl(Trim(Mid(rec0, 14, 7)))
End If
End If
If Left(rec0, 8) = "PointL2_" Then
If CInt(Mid(rec0, 9, 5)) = m Then
```

```
  MLength(n).L2 = CDbl(Trim(Mid(rec0, 14, 7)))
End If
End If
If Left(rec0, 8) = "PointL3_" Then
If CInt(Mid(rec0, 9, 5)) = m Then
  MLength(n).L3 = CDbl(Trim(Mid(rec0, 14, 7)))
  n = n + 1
  m = m + 1
End If
End If
Loop Until EOF(2)
Close
End Sub

Private Sub Command27_Click()
Dim co(3) As Double
co(1) = MLength(1).L1
co(2) = MLength(1).L2
co(3) = MLength(1).L3
For i = 1 To n - 1
going (MLength(i).L1 - co(1)) * 500, (MLength(i).L2 - co(2)) * 500, (MLength(i).L3 - co(3)) * 500
Debug.Print (MLength(i).L1 - co(1)) * 500, (MLength(i).L2 - co(2)) * 1000, (MLength(i).L3 - co(3)) * 700
For j = 1 To 50000
Next j
Next i
End Sub

Private Sub Command3_Click()
centere.y = centere.y + (BLU)
center.y = center.y + (BLU)
For i = 1 To 3
Movpoint(i).y = Movpoint(i).y + (BLU)
MovpointE(i).y = MovpointE(i).y + (BLU)
Next i
ty = centere.y
Pic1.Cls
```

```
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command4_Click()
centere.y = centere.y - (BLU)
center.y = center.y - (BLU)
For i = 1 To 3
Movpoint(i).y = Movpoint(i).y - (BLU)
MovpointE(i).y = MovpointE(i).y - (BLU)
Next i
ty = centere.y
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command5_Click()
centere.z = centere.z + (BLU)
center.z = center.z + (BLU)
For i = 1 To 3
Movpoint(i).z = Movpoint(i).z + (BLU)
MovpointE(i).z = MovpointE(i).z + (BLU)
Next i
tz = centere.z - ZOS
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub

Private Sub Command6_Click()
centere.z = centere.z - (BLU)
center.z = center.z - (BLU)
```

```
For i = 1 To 3
Movpoint(i).z = Movpoint(i).z - (BLU)
MovpointE(i).z = MovpointE(i).z - (BLU)
Next i
tz = centere.z - ZOS
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
End Sub


Private Sub Command7_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
```

```
Next i
Dx = Dx + 1
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
thx = Dx
End Sub

Private Sub Command8_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
```

```
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dx = Dx - 1
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
```

```
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
thx = Dx
End Sub

Private Sub Command9_Click()
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - tx
Movpoint(i).y = Movpoint(i).y - ty
Movpoint(i).z = Movpoint(i).z - tz
MovpointE(i).x = MovpointE(i).x - tx
MovpointE(i).y = MovpointE(i).y - ty
MovpointE(i).z = MovpointE(i).z - tz
Next i
center.x = center.x - tx
center.y = center.y - ty
center.z = center.z - tz
centere.x = centere.x - tx
centere.y = centere.y - ty
centere.z = centere.z - tz
dtx = Dx: dty = Dy: dtz = Dz
center = rollB(center, -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
```

```
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = Dx: dty = Dy: dtz = Dz
Next i
Dy = Dy + 1
dtx = Dx: dty = Dy: dtz = Dz
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = Dx: dty = Dy: dtz = Dz
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = Dx: dty = Dy: dtz = Dz
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + tx
Movpoint(i).y = Movpoint(i).y + ty
Movpoint(i).z = Movpoint(i).z + tz
MovpointE(i).x = MovpointE(i).x + tx
MovpointE(i).y = MovpointE(i).y + ty
MovpointE(i).z = MovpointE(i).z + tz
Next i
center.x = center.x + tx
center.y = center.y + ty
center.z = center.z + tz
centere.x = centere.x + tx
centere.y = centere.y + ty
centere.z = centere.z + tz
Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
thy = Dy
End Sub

Private Sub Form_Load()
Command1.Enabled = False
Command2.Enabled = False
```

```
Command3.Enabled = False
Command4.Enabled = False
Command5.Enabled = False
Command6.Enabled = False
Command7.Enabled = False
Command8.Enabled = False
Command9.Enabled = False
Command10.Enabled = False
Command11.Enabled = False
Command12.Enabled = False
Command14.Enabled = False
Command15.Enabled = False
Command20.Enabled = False
Command18.Enabled = False
Command22.Enabled = False
Command24.Enabled = False
thx.Enabled = False
thy.Enabled = False
thz.Enabled = False
tx.Enabled = False
ty.Enabled = False
tz.Enabled = False
clearfile "Textbar"
open1 "Basiclength"
Do
Input #2, rec0
If Left(rec0, 4) = "Past" Then
  Fixpoint(1).x = CDbl(Trim(Mid(rec0, 6, 7)))
  Fixpoint(1).y = CDbl(Trim(Mid(rec0, 14, 7)))
  Fixpoint(1).z = CDbl(Trim(Mid(rec0, 21, 7)))
End If
If Left(rec0, 4) = "Pand" Then
  Fixpoint(2).x = CDbl(Trim(Mid(rec0, 6, 7)))
  Fixpoint(2).y = CDbl(Trim(Mid(rec0, 14, 7)))
  Fixpoint(2).z = CDbl(Trim(Mid(rec0, 21, 7)))
End If
```

```
If Left(rec0, 4) = "Pard" Then
  Fixpoint(3).x = CDbl(Trim(Mid(rec0, 6, 7)))
  Fixpoint(3).y = CDbl(Trim(Mid(rec0, 14, 7)))
  Fixpoint(3).z = CDbl(Trim(Mid(rec0, 21, 7)))
End If
If Left(rec0, 4) = "Pbst" Then
  Movpoint(1).x = CDbl(Trim(Mid(rec0, 6, 7)))
  Movpoint(1).y = CDbl(Trim(Mid(rec0, 14, 7)))
  Movpoint(1).z = CDbl(Trim(Mid(rec0, 21, 7)))
End If
If Left(rec0, 4) = "Pbnd" Then
  Movpoint(2).x = CDbl(Trim(Mid(rec0, 6, 7)))
  Movpoint(2).y = CDbl(Trim(Mid(rec0, 14, 7)))
  Movpoint(2).z = CDbl(Trim(Mid(rec0, 21, 7)))
End If
If Left(rec0, 4) = "Pbrd" Then
  Movpoint(3).x = CDbl(Trim(Mid(rec0, 6, 7)))
  Movpoint(3).y = CDbl(Trim(Mid(rec0, 14, 7)))
  Movpoint(3).z = CDbl(Trim(Mid(rec0, 21, 7)))
End If
If Left(rec0, 4) = "Cent" Then
  center.x = CDbl(Trim(Mid(rec0, 6, 7)))
  center.y = CDbl(Trim(Mid(rec0, 14, 7)))
  center.z = CDbl(Trim(Mid(rec0, 21, 7)))
End If
If Left(rec0, 4) = "Ende" Then
  endeffector = CDbl(Trim(Mid(rec0, 13, 9)))
End If
If Left(rec0, 4) = "Leng" Then
  lengtherror = CDbl(Trim(Mid(rec0, 13, 9)))
End If
If Left(rec0, 3) = "BLU" Then
  BLU = CDbl(Trim(Mid(rec0, 5, 9)))
End If
Loop Until EOF(2)
Close
```

```
For i = 1 To 3
MovpointE(i).x = Movpoint(i).x
MovpointE(i).y = Movpoint(i).y
MovpointE(i).z = Movpoint(i).z - lengtherror
Next i
center.z = center.z - lengtherror
centere.x = center.x
centere.y = center.y
centere.z = center.z - endeffector
ZOS = centere.z
tx = centere.x
ty = centere.y
tz = centere.z - ZOS
thx = Dx
thy = Dy
thz = Dz
drawarm
drawgrid 15, 13
drawstru
Picture1.Visible = True
Picture2.Visible = False
Picture3.Visible = False
Picture4.Visible = False
Call Command13_Click
tx = 100
Call Command14_Click
Call Command13_Click
ty = 100
Call Command20_Click
tx = 50
Call Command14_Click
Call Command13_Click
ty = 0
Call Command20_Click
tx = 0
Call Command14_Click
```

```
Call Command13_Click
tz = 50
Call Command15_Click
Call Command13_Click
thx = 30
Call Command18_Click
Call Command13_Click
End Sub

Private Sub HSC1_Change()
Hexapod.Lb3 = Hexapod.HSC1
End Sub
Private Sub Option1_Click()
Command1.Enabled = True
Command2.Enabled = True
Command3.Enabled = True
Command4.Enabled = True
Command5.Enabled = True
Command6.Enabled = True
Command14.Enabled = True
Command15.Enabled = True
Command20.Enabled = True
tx.Enabled = True
ty.Enabled = True
tz.Enabled = True
Command7.Enabled = False
Command8.Enabled = False
Command9.Enabled = False
Command10.Enabled = False
Command11.Enabled = False
Command12.Enabled = False
Command18.Enabled = False
Command22.Enabled = False
Command24.Enabled = False
thx.Enabled = False
thy.Enabled = False
```

```
thz.Enabled = False
End Sub

Private Sub Option2_Click()
Command7.Enabled = True
Command8.Enabled = True
Command9.Enabled = True
Command10.Enabled = True
Command11.Enabled = True
Command12.Enabled = True
Command18.Enabled = True
Command22.Enabled = True
Command24.Enabled = True
thx.Enabled = True
thy.Enabled = True
thz.Enabled = True
Command1.Enabled = False
Command2.Enabled = False
Command3.Enabled = False
Command4.Enabled = False
Command5.Enabled = False
Command6.Enabled = False
Command14.Enabled = False
Command15.Enabled = False
Command20.Enabled = False
tx.Enabled = False
ty.Enabled = False
tz.Enabled = False
End Sub
```

Module1 (Module1.bas)

```
Dim grl(168) As TwoPD
Dim da(3) As TwoPD
Public Sub Plot(Rln As TwoPD, color As Integer)
  Dim thetax, thetaz As Double
  Dim zoom As Single
  zoom = 0.7
  If viewer = 0 Then
  thetax = (70 * pi) / 180
  thetaz = (50 * pi) / 180
  Else
  If viewer = 1 Then
   thetax = (0 * pi) / 180
  thetaz = (0 * pi) / 180
  Else
   If viewer = 2 Then
    thetax = (90 * pi) / 180
  thetaz = (0 * pi) / 180
     Else
     If viewer = 3 Then
     thetax = (90 * pi) / 180
  thetaz = (90 * pi) / 180
  End If
   End If
    End If
     End If
      If viewer = 0 Then
moview(1) = 3500
moview(2) = 5000
  Else
  If viewer = 1 Then
moview(1) = 3500
moview(2) = 2500
  Else
```

```
    If viewer = 2 Then
moview(1) = 3500
moview(2) = 5000
      Else
      If viewer = 3 Then
moview(1) = 3500
moview(2) = 5000
   End If
    End If
     End If
      End If
   xr1 = (((Rln.x1 * Cos(thetaz)) - (Rln.y1 * Sin(thetaz))) * zoom) + moview(1)
   yr1 = (((Rln.x1 * Cos(thetax) * Sin(thetaz)) + (Rln.y1 * Cos(thetax) * Cos(thetaz)) - (Rln.z1 * Sin(thetax))) *
zoom) + moview(2)
   xr2 = (((Rln.x2 * Cos(thetaz)) - ((Rln.y2) * Sin(thetaz))) * zoom) + moview(1)
   yr2 = (((Rln.x2 * Cos(thetax) * Sin(thetaz)) + (Rln.y2 * Cos(thetax) * Cos(thetaz)) - (Rln.z2 * Sin(thetax))) *
zoom) + moview(2)
Hexapod.Pic1.Line (xr1, yr1)-(xr2, yr2), color
End Sub
Sub drawgrid(x As Single, y As Single)
Dim xoff As Single, yoff As Single
xoff = ((x - 1) / 2) * 500
yoff = ((y - 1) / 2) * 500
ii = 0
For i = 1 To x
ii = ii + 1
grl(ii).x1 = ((i - 1) * 500) - xoff
grl(ii).y1 = 0 - yoff
grl(ii).x2 = ((i - 1) * 500) - xoff
grl(ii).y2 = ((y - 1) * 500) - yoff
Next i
For i = 1 To y
ii = ii + 1
grl(ii).x1 = 0 - xoff
grl(ii).y1 = ((i - 1) * 500) - yoff
grl(ii).x2 = ((x - 1) * 500) - xoff
```

```
grl(ii).y2 = ((i - 1) * 500) - yoff
Next i
For i = 1 To ii
Plot grl(i), 8
Next i
End Sub
Sub drawarm()
Hexapod.lst2.Clear
For i = 1 To 3
da(i).x1 = Fixpoint(i).x * sc
da(i).y1 = Fixpoint(i).y * sc
da(i).z1 = Fixpoint(i).z * sc
Next i
For i = 1 To 3
da(i).x2 = Movpoint(i).x * sc
da(i).y2 = Movpoint(i).y * sc
da(i).z2 = Movpoint(i).z * sc
Next i
For i = 1 To 3
Plot da(i), vbRed
Next i
For i = 1 To 3
da(i).x1 = Movpoint(i).x * sc
da(i).y1 = Movpoint(i).y * sc
da(i).z1 = Movpoint(i).z * sc
Next i
For i = 1 To 3
da(i).x2 = (MovpointE(i).x) * sc
da(i).y2 = (MovpointE(i).y) * sc
da(i).z2 = (MovpointE(i).z) * sc
Next i
For i = 1 To 3
Plot da(i), vbRed
Next i
da(1).x1 = center.x * sc
da(1).y1 = center.y * sc
```

```
da(1).z1 = center.z * sc
da(1).x2 = centere.x * sc
da(1).y2 = centere.y * sc
da(1).z2 = centere.z * sc
Hexapod.Pic1.DrawWidth = 3
Plot da(1), vbRed
Hexapod.Pic1.DrawWidth = 1
Hexapod.lst2.Clear
Hexapod.lst2.AddItem "Arm 1 = " & (Sqr((((Fixpoint(1).x) - (Movpoint(1).x)) * ((Fixpoint(1).x) - (Movpoint(1).x)))
+ (((Fixpoint(1).y) - (Movpoint(1).y)) * ((Fixpoint(1).y) - (Movpoint(1).y))) + (((Fixpoint(1).z) - (Movpoint(1).z)) *
((Fixpoint(1).z) - (Movpoint(1).z)))))
Hexapod.lst2.AddItem "Arm 2 = " & (Sqr((((Fixpoint(2).x) - (Movpoint(2).x)) * ((Fixpoint(2).x) - (Movpoint(2).x)))
+ (((Fixpoint(2).y) - (Movpoint(2).y)) * ((Fixpoint(2).y) - (Movpoint(2).y))) + (((Fixpoint(2).z) - (Movpoint(2).z)) *
((Fixpoint(2).z) - (Movpoint(2).z)))))
Hexapod.lst2.AddItem "Arm 3 = " & (Sqr((((Fixpoint(3).x) - (Movpoint(3).x)) * ((Fixpoint(3).x) - (Movpoint(3).x)))
+ (((Fixpoint(3).y) - (Movpoint(3).y)) * ((Fixpoint(3).y) - (Movpoint(3).y))) + (((Fixpoint(3).z) - (Movpoint(3).z)) *
((Fixpoint(3).z) - (Movpoint(3).z)))))
End Sub
Sub drawstru()
For i = 1 To 3
da(i).x1 = Fixpoint(i).x * sc
da(i).y1 = Fixpoint(i).y * sc
da(i).z1 = Fixpoint(i).z * sc
Next i
For i = 1 To 2
da(i).x2 = Fixpoint(i + 1).x * sc
da(i).y2 = Fixpoint(i + 1).y * sc
da(i).z2 = Fixpoint(i + 1).z * sc
Next i
da(3).x2 = Fixpoint(1).x * sc
da(3).y2 = Fixpoint(1).y * sc
da(3).z2 = Fixpoint(1).z * sc
For i = 1 To 3
Plot da(i), 5
Next i
For i = 1 To 3
```

```
da(i).x1 = MovpointE(i).x * sc
da(i).y1 = MovpointE(i).y * sc
da(i).z1 = MovpointE(i).z * sc
Next i
For i = 1 To 2
da(i).x2 = MovpointE(i + 1).x * sc
da(i).y2 = MovpointE(i + 1).y * sc
da(i).z2 = MovpointE(i + 1).z * sc
Next i
da(3).x2 = MovpointE(1).x * sc
da(3).y2 = MovpointE(1).y * sc
da(3).z2 = MovpointE(1).z * sc
For i = 1 To 3

Plot da(i), vbRed
Next i
End Sub
Function roll(Var As point, beta As Double, alpha As Double, theta As Double) As point
theta = (theta * pi) / 180
alpha = (alpha * pi) / 180
beta = (beta * pi) / 180
roll.x = ((Var.x + centere.x) * Cos(alpha) * Cos(theta)) - ((Var.y + centere.y) * Cos(alpha) * Sin(theta) * Cos(beta))
+ ((Var.y + centere.y) * Sin(alpha) * Sin(beta)) + ((Var.z - centere.z) * Cos(alpha) * Sin(theta) * Sin(beta)) + ((Var.z
- centere.z) * Sin(alpha) * Cos(beta)) - centere.x
roll.y = ((Var.x + centere.x) * Sin(theta)) + ((Var.y + centere.y) * Cos(theta) * Cos(beta)) - ((Var.z - centere.z) *
Cos(theta) * Sin(beta)) - centere.y
roll.z = (-(Var.x + centere.x) * Sin(alpha) * Cos(theta)) + ((Var.y + centere.y) * Sin(alpha) * Sin(theta) * Cos(beta))
+ ((Var.y + centere.y) * Cos(alpha) * Sin(beta)) - ((Var.z - centere.z) * Sin(alpha) * Sin(theta) * Sin(beta)) + ((Var.z
- centere.z) * Cos(alpha) * Cos(beta)) + centere.z
End Function
Function rollB(Var As point, beta As Double, alpha As Double, theta As Double) As point
theta = (theta * pi) / 180
alpha = (alpha * pi) / 180
beta = (beta * pi) / 180
rollB.x = ((Var.x + centere.x) * Cos(alpha) * Cos(theta)) - ((Var.y + centere.y) * Sin(theta)) + ((Var.z - centere.z) *
Cos(theta) * Sin(alpha)) - centere.x
```

```
rollB.y = ((Var.x + centere.x) * Sin(theta) * Cos(beta) * Cos(alpha)) + ((Var.x + centere.x) * Sin(beta) * Sin(alpha))  -
+ ((Var.y + centere.y) * Cos(beta) * Cos(theta)) + ((Var.z - centere.z) * Cos(beta) * Sin(theta) * Sin(alpha)) - ((Var.z
- centere.z) * Sin(beta) * Cos(alpha)) - centere.y
rollB.z = ((Var.x + centere.x) * Sin(beta) * Sin(theta) * Cos(alpha)) - ((Var.x + centere.x) * Cos(beta) * Sin(alpha)) +
((Var.y + centere.y) * Sin(beta) * Cos(theta)) + ((Var.z - centere.z) * Sin(alpha) * Sin(theta) * Sin(beta)) + ((Var.z -
centere.z) * Cos(alpha) * Cos(beta)) + centere.z
End Function
```

Module2 (Module2.bas)

```
Declare Function inport Lib "inpout32.dll" (ByVal x As Integer) As Integer
Declare Sub outport Lib "inpout32.dll" (ByVal x As Integer, ByVal y As Integer)
Public posx As Long
Public posy As Long
Public posz As Long
Sub drivem(k As Integer)
For ii = 1 To 50
outport &H378, k
For i% = 1 To 100
Next i%
outport &H378, 0
For i% = 1 To 100
Next i%
Next ii
End Sub
Sub spdrv(x As Integer, y As Integer, z As Integer)
If x = 1 Then sd% = sd% + 1: posx = posx + 1
If x = -1 Then sd% = sd% + 1 + 2: posx = posx - 1
If y = 1 Then sd% = sd% + 4: posy = posy + 1
If y = -1 Then sd% = sd% + 4 + 8: posy = posy - 1
If z = 1 Then sd% = sd% + 16: posz = posz + 1
If z = -1 Then sd% = sd% + 16 + 32: posz = posz - 1
drivem (sd%)
End Sub
Sub going(x As Double, y As Double, z As Double)
px! = x - posx
py! = y - posy
pz! = z - posz
qx! = 0
qy! = 0
qz! = 0
If Abs(px!) >= Abs(py!) Then Max! = Abs(px!) Else Max! = Abs(py!)
If Abs(pz!) >= Max! Then Max! = Abs(pz!)
```

```
k = Max!
For i = 0 To k - 1
qx! = qx! + Abs(px!)
If qx! >= k - 1 And px! > 0 Then qx! = qx! - k: dsx% = 1
If qx! >= k - 1 And px! < 0 Then qx! = qx! - k: dsx% = -1
qy! = qy! + Abs(py!)
If qy! >= k - 1 And py! > 0 Then qy! = qy! - k: dsy% = 1
If qy! >= k - 1 And py! < 0 Then qy! = qy! - k: dsy% = -1
qz! = qz! + Abs(pz!)
If qz! >= k - 1 And pz! > 0 Then qz! = qz! - k: dsz% = 1
If qz! >= k - 1 And pz! < 0 Then qz! = qz! - k: dsz% = -1
spdrv dsx%, dsy%, dsz%
dsx% = 0
dsy% = 0
dsz% = 0
Next i
End Sub
Public Sub save2(Name As String)
Open Name & ".txt" For Output As #1
End Sub
Public Sub open1(Name2 As String)
Open Name2 & ".txt" For Input As #2
End Sub
Public Sub clearfile(Name3 As String)
Open Name3 & ".txt" For Output As #3
Close #3
End Sub
Public Sub Delay()
Dim spd As Integer
spd = 100 - CInt(Hexapod.HSC1)
For i = 0 To spd
For j = 0 To 20000
Next j
Next i
End Sub
```

Module3 (Module3.bas)

```
Public Ve_cenTer As point, Ve_cenTerE As point, Ve_Movpoint(1 To 3) As point, Ve_MovpointE(1 To 3) As point,
Ve_rollx As Double, Ve_rolly As Double, Ve_rollz As Double
Public Vs_cenTer As point, Vs_cenTerE As point, Vs_Movpoint(1 To 3) As point, Vs_MovpointE(1 To 3) As point,
Vs_rollx As Double, Vs_rolly As Double, Vs_rollz As Double
Public Vba_cenTer As point, Vba_cenTerE As point, Vba_Movpoint(1 To 3) As point, Vba_MovpointE(1 To 3) As
point
Public rate(1 To 6) As Double, n As Integer, rateMax As Double, rollMax As Double, vareal(1 To 6) As Double
Public Sub motion(Am As Integer, Bm As Integer)
Vba_cenTer = center
Vba_cenTerE = centere
For i = 1 To 3
Vba_Movpoint(i) = Movpoint(i)
Vba_MovpointE(i) = MovpointE(i)
Next i
n = Am
open1 "Textbar"
Do
Input #2, rec0
If Left(rec0, 14) = "Polar" & n & "_____cex" Then
  Vs_cenTerE.x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____cey" Then
  Vs_cenTerE.y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____cez" Then
  Vs_cenTerE.z = CDbl(Trim(Mid(rec0, 15, 7))) + ZOS
End If
If Left(rec0, 14) = "Polar" & n & "______cx" Then
  Vs_cenTer.x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "______cy" Then
  Vs_cenTer.y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
```

```
If Left(rec0, 14) = "Polar" & n & "_____cz" Then
  Vs_cenTer.z = CDbl(Trim(Mid(rec0, 15, 7))) + ZOS
End If
If Left(rec0, 14) = "Polar" & n & "_____rx" Then
  Vs_rollx = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____ry" Then
  Vs_rolly = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____rz" Then
  Vs_rollz = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(1)x" Then
  Vs_Movpoint(1).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(1)y" Then
  Vs_Movpoint(1).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(1)z" Then
  Vs_Movpoint(1).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(2)x" Then
  Vs_Movpoint(2).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(2)y" Then
  Vs_Movpoint(2).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(2)z" Then
  Vs_Movpoint(2).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(3)x" Then
  Vs_Movpoint(3).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(3)y" Then
  Vs_Movpoint(3).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
```

```
If Left(rec0, 14) = "Polar" & n & "_mov(3)z" Then
  Vs_Movpoint(3).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If

If Left(rec0, 14) = "Polar" & n & "move(1)x" Then
  Vs_MovpointE(1).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(1)y" Then
  Vs_MovpointE(1).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(1)z" Then
  Vs_MovpointE(1).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(2)x" Then
  Vs_MovpointE(2).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(2)y" Then
  Vs_MovpointE(2).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(2)z" Then
  Vs_MovpointE(2).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(3)x" Then
  Vs_MovpointE(3).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(3)y" Then
  Vs_MovpointE(3).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(3)z" Then
  Vs_MovpointE(3).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
Loop Until EOF(2)
Close

n = Bm
open1 "Textbar"
```

```
Do
Input #2, rec0
If Left(rec0, 14) = "Polar" & n & "_____cex" Then
  Ve_cenTerE.x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____cey" Then
  Ve_cenTerE.y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____cez" Then
  Ve_cenTerE.z = CDbl(Trim(Mid(rec0, 15, 7))) + ZOS
End If
If Left(rec0, 14) = "Polar" & n & "_____cx" Then
  Ve_cenTer.x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____cy" Then
  Ve_cenTer.y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____cz" Then
  Ve_cenTer.z = CDbl(Trim(Mid(rec0, 15, 7))) + ZOS
End If
If Left(rec0, 14) = "Polar" & n & "_____rx" Then
  Ve_rollx = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____ry" Then
  Ve_rolly = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_____rz" Then
  Ve_rollz = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(1)x" Then
  Ve_Movpoint(1).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(1)y" Then
  Ve_Movpoint(1).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(1)z" Then
```

```
  Ve_Movpoint(1).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(2)x" Then
  Ve_Movpoint(2).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(2)y" Then
  Ve_Movpoint(2).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(2)z" Then
  Ve_Movpoint(2).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(3)x" Then
  Ve_Movpoint(3).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(3)y" Then
  Ve_Movpoint(3).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "_mov(3)z" Then
  Ve_Movpoint(3).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If

If Left(rec0, 14) = "Polar" & n & "move(1)x" Then
  Ve_MovpointE(1).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(1)y" Then
  Ve_MovpointE(1).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(1)z" Then
  Ve_MovpointE(1).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(2)x" Then
  Ve_MovpointE(2).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(2)y" Then
  Ve_MovpointE(2).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
```

```
If Left(rec0, 14) = "Polar" & n & "move(2)z" Then
  Ve_MovpointE(2).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(3)x" Then
  Ve_MovpointE(3).x = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(3)y" Then
  Ve_MovpointE(3).y = CDbl(Trim(Mid(rec0, 15, 7)))
End If
If Left(rec0, 14) = "Polar" & n & "move(3)z" Then
  Ve_MovpointE(3).z = CDbl(Trim(Mid(rec0, 15, 7)))
End If
Loop Until EOF(2)
Close
calculate
For i = 1 To 3
Movpoint(i) = Vs_Movpoint(i)
MovpointE(i) = Vs_MovpointE(i)
Next i
centere = Vs_cenTerE
center = Vs_cenTer
Hexapod.Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
vareal(1) = centere.x
vareal(2) = centere.y
vareal(3) = centere.z
vareal(4) = Vs_rollx
vareal(5) = Vs_rolly
vareal(6) = Vs_rollz
If rateMax <> 0 Then
For i = 1 To rateMax
centere.x = centere.x + rate(1)
center.x = center.x + rate(1)
centere.y = centere.y + rate(2)
```

```
center.y = center.y + rate(2)
centere.z = centere.z + rate(3)
center.z = center.z + rate(3)
For j = 1 To 3
Movpoint(j).x = Movpoint(j).x + rate(1)
MovpointE(j).x = MovpointE(j).x + rate(1)
Movpoint(j).z = Movpoint(j).z + rate(3)
MovpointE(j).z = MovpointE(j).z + rate(3)
Movpoint(j).y = Movpoint(j).y + rate(2)
MovpointE(j).y = MovpointE(j).y + rate(2)
Next j
vareal(1) = vareal(1) + rate(1)
vareal(2) = vareal(2) + rate(2)
vareal(3) = vareal(3) + rate(3)
Hexapod.Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
If QuToRec = 1 Then
stepper = stepper + 1
RecordArm(stepper).L1 = (Sqr(((Fixpoint(1).x) - (Movpoint(1).x)) * ((Fixpoint(1).x) - (Movpoint(1).x))) +
(((Fixpoint(1).y) - (Movpoint(1).y)) * ((Fixpoint(1).y) - (Movpoint(1).y))) + (((Fixpoint(1).z) - (Movpoint(1).z)) *
((Fixpoint(1).z) - (Movpoint(1).z)))))
RecordArm(stepper).L2 = (Sqr(((Fixpoint(2).x) - (Movpoint(2).x)) * ((Fixpoint(2).x) - (Movpoint(2).x))) +
(((Fixpoint(2).y) - (Movpoint(2).y)) * ((Fixpoint(2).y) - (Movpoint(2).y))) + (((Fixpoint(2).z) - (Movpoint(2).z)) *
((Fixpoint(2).z) - (Movpoint(2).z)))))
RecordArm(stepper).L3 = (Sqr(((Fixpoint(3).x) - (Movpoint(3).x)) * ((Fixpoint(3).x) - (Movpoint(3).x))) +
(((Fixpoint(3).y) - (Movpoint(3).y)) * ((Fixpoint(3).y) - (Movpoint(3).y))) + (((Fixpoint(3).z) - (Movpoint(3).z)) *
((Fixpoint(3).z) - (Movpoint(3).z)))))
save2 Hexapod.Text3
For ii = 1 To stepper
Print #1, "PointL1_" & 10000 + ii & RecordArm(ii).L1
Print #1, "PointL2_" & 10000 + ii & RecordArm(ii).L2
Print #1, "PointL3_" & 10000 + ii & RecordArm(ii).L3
Print #1, " "
Next ii
```

```
Close
Else
End If
Delay
Next i

Else

For j = 1 To rollMax
Dim dtx As Double, dty As Double, dtz As Double
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x - vareal(1)
Movpoint(i).y = Movpoint(i).y - vareal(2)
Movpoint(i).z = Movpoint(i).z - vareal(3)
MovpointE(i).x = MovpointE(i).x - vareal(1)
MovpointE(i).y = MovpointE(i).y - vareal(2)
MovpointE(i).z = MovpointE(i).z - vareal(3)
Next i
center.x = center.x - vareal(1)
center.y = center.y - vareal(2)
center.z = center.z - vareal(3)
centere.x = centere.x - vareal(1)
centere.y = centere.y - vareal(2)
centere.z = centere.z - vareal(3)
dtx = vareal(4): dty = vareal(5): dtz = vareal(6)
center = rollB(center, -dtx, -dty, -dtz)
dtx = vareal(4): dty = vareal(5): dtz = vareal(6)
For i = 1 To 3
Movpoint(i) = rollB(Movpoint(i), -dtx, -dty, -dtz)
dtx = vareal(4): dty = vareal(5): dtz = vareal(6)
MovpointE(i) = rollB(MovpointE(i), -dtx, -dty, -dtz)
dtx = vareal(4): dty = vareal(5): dtz = vareal(6)
Next i

vareal(4) = vareal(4) + rate(4)
vareal(5) = vareal(5) + rate(5)
```

```
vareal(6) = vareal(6) + rate(6)


dtx = vareal(4): dty = vareal(5): dtz = vareal(6)
center = roll(center, dtx, dty, dtz)
For i = 1 To 3
dtx = vareal(4): dty = vareal(5): dtz = vareal(6)
Movpoint(i) = roll(Movpoint(i), dtx, dty, dtz)
dtx = vareal(4): dty = vareal(5): dtz = vareal(6)
MovpointE(i) = roll(MovpointE(i), dtx, dty, dtz)
Next i
For i = 1 To 3
Movpoint(i).x = Movpoint(i).x + vareal(1)
Movpoint(i).y = Movpoint(i).y + vareal(2)
Movpoint(i).z = Movpoint(i).z + vareal(3)
MovpointE(i).x = MovpointE(i).x + vareal(1)
MovpointE(i).y = MovpointE(i).y + vareal(2)
MovpointE(i).z = MovpointE(i).z + vareal(3)
Next i
center.x = center.x + vareal(1)
center.y = center.y + vareal(2)
center.z = center.z + vareal(3)
centere.x = centere.x + vareal(1)
centere.y = centere.y + vareal(2)
centere.z = centere.z + vareal(3)
vareal(1) = centere.x
vareal(2) = centere.y
vareal(3) = centere.z
Hexapod.Pic1.Cls
drawgrid 15, 13
drawarm
drawstru
If QuToRec = 1 Then
stepper = stepper + 1
RecordArm(stepper).L1 = (Sqr((((Fixpoint(1).x) - (Movpoint(1).x)) * ((Fixpoint(1).x) - (Movpoint(1).x))) +
(((Fixpoint(1).y) - (Movpoint(1).y)) * ((Fixpoint(1).y) - (Movpoint(1).y))) + (((Fixpoint(1).z) - (Movpoint(1).z)) *
((Fixpoint(1).z) - (Movpoint(1).z)))))
```

```
RecordArm(stepper).L2 = (Sqr((((Fixpoint(2).x) - (Movpoint(2).x)) * ((Fixpoint(2).x) - (Movpoint(2).x))) +
(((Fixpoint(2).y) - (Movpoint(2).y)) * ((Fixpoint(2).y) - (Movpoint(2).y))) + (((Fixpoint(2).z) - (Movpoint(2).z)) *
((Fixpoint(2).z) - (Movpoint(2).z)))))
RecordArm(stepper).L3 = (Sqr((((Fixpoint(3).x) - (Movpoint(3).x)) * ((Fixpoint(3).x) - (Movpoint(3).x))) +
(((Fixpoint(3).y) - (Movpoint(3).y)) * ((Fixpoint(3).y) - (Movpoint(3).y))) + (((Fixpoint(3).z) - (Movpoint(3).z)) *
((Fixpoint(3).z) - (Movpoint(3).z)))))
save2 Hexapod.Text3
For ii = 1 To stepper
Print #1, "PointL1_" & RecordArm(ii).L1
Print #1, "PointL2_" & RecordArm(ii).L2
Print #1, "PointL3_" & RecordArm(ii).L3
Print #1, " "
Next ii
Close
Else
End If
Delay
Next j


End If
End Sub
Sub calculate()
rate(1) = Ve_cenTerE.x - Vs_cenTerE.x
rate(2) = Ve_cenTerE.y - Vs_cenTerE.y
rate(3) = Ve_cenTerE.z - Vs_cenTerE.z
rate(4) = Ve_rollx - Vs_rollx
rate(5) = Ve_rolly - Vs_rolly
rate(6) = Ve_rollz - Vs_rollz
rollMax = Abs(rate(4))
For i = 5 To 6
If Abs(rate(i)) > rollMax Then rollMax = Abs(rate(i))
Next i
If rollMax <> 0 Then
For i = 4 To 6
rate(i) = rate(i) / rollMax
Next i
```

```
End If
rateMax = rate(1)
For i = 2 To 3
If Abs(rate(i)) > rateMax Then rateMax = Abs(rate(i))
Next i
If rateMax <> 0 Then
For i = 1 To 3
rate(i) = rate(i) / rateMax
Next i
End If
End Sub
```

Variable (Variable.bas)

```
Type TwoPD
   x1 As Double
   x2 As Double
   y1 As Double
   y2 As Double
   z1 As Double
   z2 As Double
End Type
Type point
   x As Double
   y As Double
   z As Double
End Type
Type PointPolar
   cx As Double
   cy As Double
   cz As Double
   cex As Double
   cey As Double
   cez As Double
   mov(1 To 3) As point
   move(1 To 3) As point
   rx As Double
   ry As Double
   rz As Double
End Type
Type ContainArm
   L1 As Double
   L2 As Double
   L3 As Double
End Type
Public Fixpoint(1 To 3) As point
Dim rec0 As String
```

```
Public center As point, endeffector As Double, lengtherror As Double, centere As point, BLU As Double, ZOS As
Double
Public Movpoint(1 To 3) As point, MovpointE(1 To 3) As point, Dx As Double, Dy As Double, Dz As Double, co
As Long
Public Polar(1000) As PointPolar, viewer As Integer, moview(2) As Double, numPo As Long, stepper As Integer
Global Const sc = 10, pi = 3.14159265358979
Public RecordArm(10000) As ContainArm, QuToRec As Integer
Type Arm
  L1 As Double
  L2 As Double
  L3 As Double
End Type
Public MLength(10000) As Arm
```