เครื่องกัดแนวดิ่งซีเอ็นซีชนิดพื้นฐานไมโครคอมพิวเตอร์

CNC VERTICAL MILLING MACHINE BASED ON MICROCOMPUTER

โดย

น.ส.สิริวรินทร์ เพชรรัตน์ 35104472

นาย สุลมพ์ฤทธิ์ เลาจเวชกุล 35104508

นาย อำนาจ คณะรัฐ 35104557

อาจารย์ที่ปรึกษา

อ.กวิน สนธิเพิ่มพูน

อ.จำลอง ปราบแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมเครื่องกล

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปีการศึกษา 2538

เครื่องกัดแนวดิ่งซีเอ็นซีบนพื้นฐานไมโครคอมพิวเตอร์

โดย

น.ส.สิริวรินทร์ เพชรรัตน์ 35104472

นาย สุลมพ์ฤทธิ์ เลาจเวชกุล 35104508

นาย อำนาจ คณะรัฐ 35104557

อาจารย์ที่ปรึกษา

(อ.กวิน สนธิเพิ่มพูน)

(อ.จำลอง ปราบแก้ว)

เครื่องกัดแนวดิ่งซีเอ็นซีบนพื้นฐานไมโครคอมพิวเตอร์

(CNC VERTICAL MILLING MACHINE BASED ON MICROCOMPUTER)

นักศึกษา สิริวรินทร์ เพชรรัตน์ อาจารย์ที่ปรึกษา อาจารย์ กวิน์ สนธิเพิ่มพูน
สุลมพ์ฤทธิ์ เลาจเวชกุล อาจารย์ จำลอง ปราบแก้ว
อำนาจ คณะรัฐ

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้ศึกษาถึงการควบคุมตำแหน่งและความเร็วของเอซีเซอร์โวมอเตอร์ (A.C. servo motor) โดยอาศัยการควบคุมผ่านทางการ์ดควบคุมสำเร็จรูป (PC SERVO-MOTOR CONTROLLER MODEL 5650) ซึ่งสามารถทำงานและสั่งงานบนเครื่องไมโครคอมพิวเตอร์โดยตรงได้ เพื่อนำไปประยุกต์ใช้ในการสร้างโปรแกรมสำหรับควบคุมเครื่องกัดแนวดิ่งซีเอ็นซีบนพื้นฐานไมโครคอมพิวเตอร์ (CNC VERTICAL MILLING MACHINE BASED ON MICROCOMPUTER) เนื่องจากการ์ดควบคุมสำเร็จรูปที่นำมาใช้นั้น ยังไม่สามารถรับคำสั่งมาตราฐาน G-Code และ M-Code ได้ จึงจำเป็นที่จะต้องเขียนซอฟท์แวร์ขึ้นมาเพื่อควบคุมการ์ดให้สามารถรับและทำตามคำสั่งที่เข้ามาในรูปคำสั่งมาตราฐานได้ รวมทั้งออกแบบให้มีฟังก์ชันพิเศษที่ช่วยอำนวยความสะดวกในการทำงาน และในส่วนของการพัฒนาโปรแกรมก็ยังสามารถทำได้โดยง่ายนอกจากนั้นเครื่องกัดแนวดิ่งที่จะนำมาใช้ก็ต้องอยู่ในสภาพพร้อมที่จะติดตั้งระบบควบคุมแบบคอมพิวเตอร์ได้ จากผลงานที่ได้ทำขึ้นจะได้ต้นแบบของเครื่องกัดแนวดิ่งซีเอ็นซี ที่มีราคาไม่แพง ใช้ต้นทุนในการผลิตต่ำ แต่สามารถใช้งานได้อย่างมีประสิทธิภาพทัดเทียมกับเครื่องกัดที่สั่งเข้ามาจากต่างประเทศ และสามารถพัฒนาโปรแกรมใช้งานให้อยู่ในรูปแบบที่เหมาะสมกับงานเฉพาะทางได้ อีกทั้งยังนำไปประยุกต์ใช้งานในเชิงอุตสาหกรรมได้อย่างกว้างขวางโดยเฉพาะอย่างยิ่งในอุตสาหกรรมการผลิตชิ้นส่วนรถยนต์ และแม่พิมพ์ เป็นต้น

ABSTRACT

This thesis is studying the AC servo motor velocity and position control, by using the Servo-motor controlling card model 5650 which can function and be operated directly via microcomputer. It is applied for the designing of CNC vertical milling machine based on microcomputer. Since the controlling card cannot be input by the standard G-code and M-code, it is essential to program the specific software for making the card adaptable and operatable by those standard instructions, and also containing particular functions for ease of operation, and contributing to further program development, as well. Moreover, the CNC vertical milling machine should also be ready for any computer control system installation, indeed. From the result of the tedious work, we acquired the prototype of the CNC vertical milling machine, which is inexpensive, low operating cost, and perform machining job efficiently as well as other import CNC vertical milling machines from abroad. We also have developed the compact multi-purpose program for all highly-specific application. It can be used in wide range industry fields ,for example: car assembly manufacturing and die casting and so on.

สารบัญ

บทที่ 1
บทนำ

ในสภาวะที่เศรษฐกิจเจริญเติบโตขึ้นมาเรื่อยๆ และจำนวนประชากรที่เพิ่มมากขึ้น ความต้องการทางด้านปัจจัย 4 ก็มีเพิ่มมากขึ้นตามลำดับ การแข่งขันทางการค้าก็ยิ่งทวีสูงขึ้นเรื่อยๆ เหตุต่างๆ เหล่านี้ ทำให้มนุษย์มีความจำเป็นที่จะต้องคิดค้นและพัฒนาการผลิตให้รวดเร็วและประหยัด เพื่อตอบสนองต่อความต้องการที่เพิ่มมากขึ้น เครื่องจักรกลอัตโนมัติได้ถูกออกแบบและพัฒนาสร้างขึ้นมาให้สามารถทำงานซ้ำๆ กันได้ทุกเวลาที่ต้องการ ซึ่งระบบการทำงานอัตโนมัติเป็นที่รู้จักกันอย่างแพร่หลาย เช่น เครื่องเล่นเปียโนอัตโนมัติซึ่งทำงานโดยระบบแมคคานิคควบคุม เครื่องกลึงอัตโนมัติที่ควบคุมการทำงานด้วยลูกเบี้ยว แต่เครื่องจักรเหล่านี้มีข้อเสียตรงที่ การเปลี่ยนผลิตภัณฑ์หรือชิ้นงานใหม่ต้องใช้เวลามาก และการเปลี่ยนลักษณะงานมีขีดจำกัด

ในปี ค.ศ. 1948 นักวิทยาศาสตร์ในสถาบัน MIT (Massachusetts Institute of Technology) ได้ริเริ่มทำโครงการพัฒนาเครื่องจักรกลที่ควบคุมด้วยระบบคอมพิวเตอร์ขึ้น โดยได้รับการสนับสนุนโครงการจากกองทัพอากาศของสหรัฐอเมริกา (U.S. Air Force)

เครื่องจักรระบบเอ็นซีเครื่องแรก คือ CINCINNATIC HYDROTEL VERTICAL-SPINDLE MACHINE และนำออกใช้งานในปี ค.ศ. 1957

สำหรับในประเทศไทยนั้น เริ่มต้นโดยการนำเข้าเครื่องเก่าจากต่างประเทศ ซึ่งมีราคาค่อนข้างต่ำเพื่อการเริ่มต้นศึกษาเรียนรู้ถึงความสามารถ เครื่องส่วนใหญ่ที่นำเข้าจะเป็นเครื่องที่ใช้ NC ในการควบคุม จึงมีขนาดใหญ่ในส่วนของคอมพิวเตอร์ในการควบคุม และในเวลาต่อมาก็เริ่มนำเข้าเครื่องจักรที่ควบคุมด้วย CNC จนถึงปัจจุบัน ในเวลาปัจจุบันเครื่องที่นำเข้าสมัยก่อนที่ควบคุมด้วย NC ก็เริ่มประสบปัญหาคือ ใช้งานไม่ได้ในส่วนของคอมพิวเตอร์หรือส่วนของอิเล็กทรอนิคส์ในการควบคุมเครื่องจักร ดังนั้น จึงมีอาจารย์และนักวิจัยหลายกลุ่มหลายสถาบันเริ่มให้ความสนใจในการวิจัยและพัฒนาส่วนควบคุมเครื่องจักรดังกล่าว[1]

บทที่ 2
ทฤษฎีเบื้องต้นเกี่ยวกับเครื่องจักรกลซีเอ็นซี

2.1 พัฒนาการของเครื่องจักรกลเอ็นซีและซีเอ็นซี

2.1.1 ความหมายของเอ็นซีและซีเอ็นซี

เอ็นซี (NC) ย่อมาจากคำว่า Numerical Control หมายถึง การควบคุมเครื่องจักรกลด้วยระบบตัวเลขและตัวอักษร ซึ่งคำจำกัดความนี้ได้จากประเทศสหรัฐอเมริกา กล่าวคือ การเคลื่อนที่ต่างๆ ตลอดจนการทำงานอื่นๆ ของเครื่องจักรกล จะถูกควบคุมโดยรหัสคำสั่งที่ประกอบด้วยตัวเลข ตัวอักษร และสัญลักษณ์อื่นๆ ซึ่งจะถูกแปลงเป็นคลื่นสัญญาณ (pulse)ของกระแสไฟฟ้าหรือสัญญาณออกอื่นๆ ที่จะไปกระตุ้นมอเตอร์หรืออุปกรณ์อื่นๆ เพื่อทำให้เครื่องจักรกลทำงานตามขั้นตอนที่ต้องการ

ซีเอ็นซี (CNC) ย่อมาจากคำว่า Computerized Numerical Control ระบบควบคุมเอ็นซีแบบนี้จะมีคอมพิวเตอร์ที่มีความสามารถสูงเพิ่มเข้าไปภายในระบบ ทำให้สามารถจัดการกับข้อมูลที่ป้อนเข้าไปในระบบเอ็นซี และประมวลผลข้อมูลเพื่อนำผลลัพธ์ที่ได้ไปควบคุมการทำงานของเครื่องจักรกล

2.1.2 ความแตกต่างระหว่างเครื่องจักรกลเอ็นซีกับเครื่องจักรกลทั่วไป

ความแตกต่างในการใช้เครื่องจักรกลเอ็นซี เมื่อเปรียบเทียบกับเครื่องจักรกลที่ใช้ทั่วไปก็คือ การตัดสินใจในการกำหนดขั้นตอนการทำงานต่างๆ จะกระทำเพียงครั้งเดียว กล่าวคือ จะกระทำในขั้นตอนการวางแผนและสร้างโปรแกรมสำหรับควบคุมเครื่องจักรกลเท่านั้น ต่อจากนั้น โปรแกรมก็จะถูกนำไปใช้ในการควบคุมการทำงานของเครื่องจักรกล สำหรับการผลิตชิ้นงานที่ต้องการ โดยสามารถทำการผลิตซ้ำๆกันกี่ครั้งก็ได้ตามต้องการ

			เครื่องจักรกลทั่วไป	เครื่องจักรกลเอ็นซี
1	การป้อนโปรแกรม		ไม่มี	มี
2	การจับยึดชิ้นงาน	ขั้น	มือ	มือ
3	การจับยึดเครื่องมือตัด	เ ต รี ย ม	มือ	มือ หรือชุดควบคุม
4	การตั้งจุดอ้างอิง	งาน	มือ	มือ
5	การตั้งความเร็วรอบ		มือ	ระบบควบคุม

6	การเลื่อนแท่นเลื่อน	ขั้น	มือหมุน	ระบบควบคุม
7	การเปรียบเทียบระยะ	ตัด	สายตา	ระบบควบคุม
8	การตรวจสอบขนาด	เฉือน	เครื่องมือวัด	ใช้เวลาน้อยกว่า

ตาราง 2.1 ตารางเปรียบเทียบการทำงานระหว่างเครื่องจักรกลทั่วไปกับเครื่องจักรกลเอ็นซี

2.1.3 ความแตกต่างระหว่างระบบเอ็นซีกับระบบซีเอ็นซี

ระบบซีเอ็นซีเป็นระบบที่พัฒนาต่อเนื่องมาจากระบบเอ็นซี ดังนั้น ความแตกต่างระหว่างระบบเอ็นซีกับระบบซีเอ็นซี ก็จะอยู่ที่ความสามารถของระบบควบคุม นั่นคือคอมพิวเตอร์ เมื่อนำระบบซีเอ็นซีไปควบคุมเครื่องจักรกล ความสามารถในการทำงานต่างๆ จะเพิ่มมากขึ้นเมื่อเปรียบเทียบกับเครื่องจักรกลเอ็นซีดังนี้

1. การแสดงภาพจำลอง (Simulation) การทำงานตามโปรแกรมที่ป้อนเข้าในระบบทางจอภาพ
2. ความจุของหน่วยความจำเพิ่มมากขึ้น สามารถเก็บข้อมูลโปรแกรมได้มาก
3. การแก้ไขและลบโปรแกรมสามารถกระทำได้ที่เครื่องจักรโดยตรง
4. สามารถส่งข้อมูลไปเก็บไว้ในหน่วยความจำภายนอกได้
5. ระบบความปลอดภัยเพิ่มมากขึ้น
6. มีการชดเชยความผิดพลาดที่เกิดจากการวัดและการส่งกำลัง
7. มีโปรแกรมสำเร็จสำหรับการคำนวณค่าต่างๆ เช่น ความเร็วรอบ อัตราป้อน เป็นต้น

2.1.4 ข้อดีและข้อเสียของเครื่องจักรกลเอ็นซีและซีเอ็นซี

เครื่องจักรกลเอ็นซีและซีเอ็นซี เป็นเครื่องจักรกลสมัยใหม่ที่มีประสิทธิภาพการทำงานสูง แต่ในขณะเดียวกันราคาก็สูงตามด้วย ดังนั้น ก่อนที่จะพิจารณาจัดซื้อเครื่องจักรกลประเภทนี้มาใช้ในกระบวนการผลิต จำเป็นที่จะต้องศึกษารายละเอียดต่างๆ เกี่ยวกับขีดความสามารถของเครื่อง ตลอดจนข้อดีและข้อเสียของเครื่องจักรกลประเภทนี้ก่อน

ข้อดีของเครื่องจักรกลเอ็นซีและซีเอ็นซี เมื่อเปรียบเทียบกับเครื่องจักรกลอัตโนมัติประเภทอื่นๆ พอจะสรุปได้ดังนี้

1. มีความยืดหยุ่นในการทำงานสูง การเปลี่ยนงานใหม่จะแก้ไขหรือเปลี่ยนแปลงเฉพาะโปรแกรมเท่านั้น
2. ความเที่ยงตรง (Accuracy) จะอยู่ระดับเดียวกันตลอดช่วงความเร็วรอบและอัตราป้อนที่ใช้ทำการผลิต
3. ใช้เวลาในการผลิต (Production Time) สั้นกว่า

4. สามารถใช้ผลิตชิ้นงานที่มีรูปทรงซับซ้อนได้ง่าย
5. การปรับตั้งเครื่องจักรกระทำได้ง่าย ใช้เวลาน้อยกว่าการผลิตด้วยวิธีอื่น ๆ
6. หลีกเลี่ยงความจำเป็นที่ต้องใช้ช่างควบคุมที่มีทักษะและประสบการณ์สูง
7. ช่างควบคุมเครื่องมีเวลาว่างจากการควบคุมเครื่อง สามารถที่จะจัดเตรียมงานอื่นๆ ไว้ล่วงหน้าได้
8. การตรวจสอบคุณภาพไม่จำเป็นต้องกระทำทุกขั้นตอนและทุกชิ้น

ส่วนข้อเสียของเครื่องจักรกลเอ็นซีและซีเอ็นซีมีดังนี้

1. ราคาของเครื่องจักรค่อนข้างสูง
2. การบำรุงรักษามีความซับซ้อนมาก
3. จำเป็นต้องใช้ช่างเขียนโปรแกรม (Part Programmer) ที่มีทักษะสูงและฝึกอบรมมาโดยเฉพาะ
4. ชิ้นส่วนหรืออะไหล่ที่ใช้ในการซ่อมบำรุง ไม่สามารถผลิตได้ในประเทศ จำเป็นต้องสั่งซื้อหรือนำเข้าจากต่างประเทศ
5. การซ่อมบำรุงจะต้องใช้ช่างที่มีประสบการณ์สูงและผ่านการฝึกอบรมมาโดยเฉพาะ
6. ราคาของเครื่องมือต่างๆ ที่ใช้ในกระบวนการตัดเฉือน เช่น แกนเพลายึดมีดกัด มีดกลึงแบบใช้อินเสิร์ท (Insert) เป็นต้น มีราคาสูง
7. พื้นที่ติดตั้งเครื่องจักร จะต้องควบคุมระดับอุณหภูมิ ความชื้น และฝุ่นละออง

ข้อมูลเหล่านี้ จำเป็นจะต้องพิจารณาอย่างรอบคอบก่อนที่จะพิจารณาจัดซื้อ ซึ่งสามารถสอบถามได้จากบริษัทผู้ผลิตหรือตัวแทนจำหน่ายได้โดยตรง

2.2 เครื่องจักรกลเอ็นซีและซีเอ็นซี

2.2.1 การทำงานของเครื่องจักรกลเอ็นซี

หลักการทำงานของเครื่องจักรกลเอ็นซีหรือซีเอ็นซี จะคล้ายคลึงกับเครื่องจักรกลทั่วไป กล่าวคือโดยพื้นฐานเบื้องต้นแล้วเครื่องจักรกลเอ็นซีก็จะทำการผลิตชิ้นงานเหมือนกับเครื่องจักรกลทั่วไป เช่น เครื่องกัดเอ็นซี ก็จะทำงานเหมือนกับเครื่องกัดทั่วไป เพียงแต่ว่าระบบควบคุมเอ็นซีของเครื่องจะทำงานในขั้นตอนต่างๆ แทนช่างควบคุมเครื่อง อย่างไรก็ตาม ก่อนที่เครื่องจักรกลเอ็นซีจะสามารถทำงานได้นั้น ระบบควบคุมของเครื่องจะต้องได้รับการบอกกล่าวเสียก่อนว่าจะให้ทำอะไร และจะต้องบอกกล่าวเป็นภาษาที่ระบบควบคุมสามารถเข้าใจได้ นั่นคือ จะต้องป้อนโปรแกรมเข้าไปในระบบควบคุมของเครื่องผ่านแป้นพิมพ์(keyboard)หรือเทปแม่เหล็ก(magnetic tape) ก็ได้

เมื่อระบบควบคุมอ่านโปรแกรมที่ป้อนเข้าไปแล้ว ก็จะนำไปควบคุมให้เครื่องจักรกลทำงาน แต่เนื่องจากเครื่องจักรกลเอ็นซีไม่มีมือสำหรับหมุนมือหมุนให้แท่นเลื่อนเคลื่อนที่ได้ ดังนั้น แท่นเลื่อนต่างๆ จะต้องมีมอเตอร์ป้อน (feed motor) ประกอบอยู่ เช่น เครื่องกัดซีเอ็นซีจะมีการเคลื่อนที่ 3 แนวแกน ก็จะมีมอเตอร์ป้อน 3 ตัว

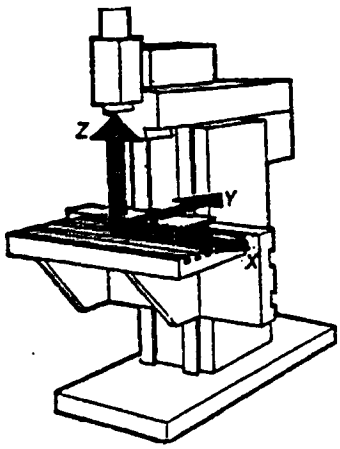
เมื่อระบบควบคุมอ่านโปรแกรมแล้ว ก็จะเปลี่ยนรหัสโปรแกรมนั้นให้เป็นสัญญาณทางไฟฟ้า เพื่อไปควบคุมให้มอเตอร์ทำงาน แต่เนื่องจากสัญญาณที่ออกจากระบบควบคุมนี้มีกำลังน้อย ไม่สามารถไปหมุนขับให้มอเตอร์ทำงานได้ ดังนั้น จึงต้องส่งสัญญาณนี้เข้าไปในภาคขยายสัญญาณของระบบขับ (drive amplified) และส่งต่อไปยังมอเตอร์ป้อนของแนวแกนที่ต้องการเคลื่อนที่

ความเร็วและระยะทางการเคลื่อนที่ของแท่นเลื่อน จะต้องกำหนดให้ระบบควบคุมรู้ ช่างควบคุมเครื่องอาศัยสายตามองดูตำแหน่งของคมตัดกับชิ้นงาน ก็จะรู้ว่าจะต้องเลื่อนแท่นเลื่อนไปอีกเป็นระยะทางเท่าใด แต่ระบบควบคุมเอ็นซีมองไม่ได้ ดังนั้น จึงต้องออกแบบอุปกรณ์หรือเครื่องมือที่สามารถจะบอกตำแหน่งของแท่นเลื่อนให้ระบบควบคุมรู้ได้ อุปกรณ์ชุดนี้เรียกว่า ระบบวัดขนาด (Measuring System) ซึ่งประกอบด้วยสเกลแนวตรง (Linear Scale) มีจำนวนเท่ากับจำนวนแนวแกนในการเคลื่อนที่ของเครื่องจักรกล ทำหน้าที่ส่งสัญญาณไฟฟ้าที่สัมพันธ์กับระยะทางที่แท่นเลื่อนเคลื่อนที่กลับไปยังระบบควบคุม ทำให้ระบบควบคุมรู้ว่าแท่นเลื่อนเคลื่อนที่ไปเป็นระยะทางเท่าใดแล้ว

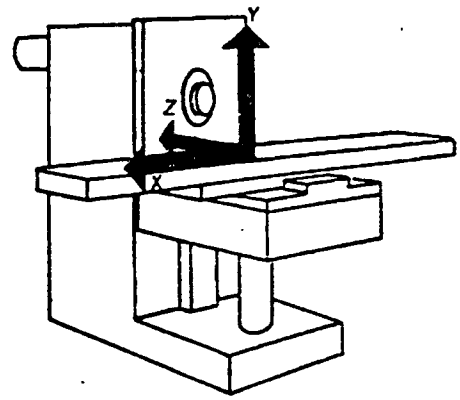
จากหลักการควบคุมการทำงานดังกล่าว ทำให้เครื่องจักรกลซีเอ็นซีสามารถผลิตชิ้นงานให้มีรูปทรงและขนาดที่ต้องการได้ จากลักษณะสร้างและการทำงานที่เหนือกว่าเครื่องจักรกลทั่วไป ทำให้เครื่องจักรกลเอ็นซีและซีเอ็นซี เป็นปัจจัยหนึ่งที่มีความสำคัญมากในอุตสาหกรรมอัตโนมัติ และมีปริมาณความต้องการใช้เพิ่มมากขึ้นเรื่อยๆ

2.2.2 เครื่องกัดเอ็นซี (NC Milling Machines)

เครื่องกัดเอ็นซีเป็นเครื่องจักรกลประเภทหนึ่งที่มีขอบข่ายการทำงานค่อนข้างกว้าง กล่าวคือ นอกจากจะสามารถทำงานกัดเช่นเดียวกับเครื่องกัดทั่วไปแล้ว ยังสามารถทำงานอื่นๆ เช่น เจาะรู ทำเกลียว คว้านรู ได้อีกด้วย โดยทั่วไปเครื่องกัดเอ็นซีจะแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ เครื่องกัดเอ็นซีเพลาตั้ง กับเครื่องกัดเอ็นซีเพลานอน ซึ่งขึ้นอยู่กับการวางตำแหน่งของเพลาหัวเครื่อง เครื่องกัดเอ็นซีจะมีแนวแกนการควบคุมตั้งแต่ 3 แกน 4 แกน 5 แกน และมากกว่า ดังแสดงในรูป 2.1 และ 2.2

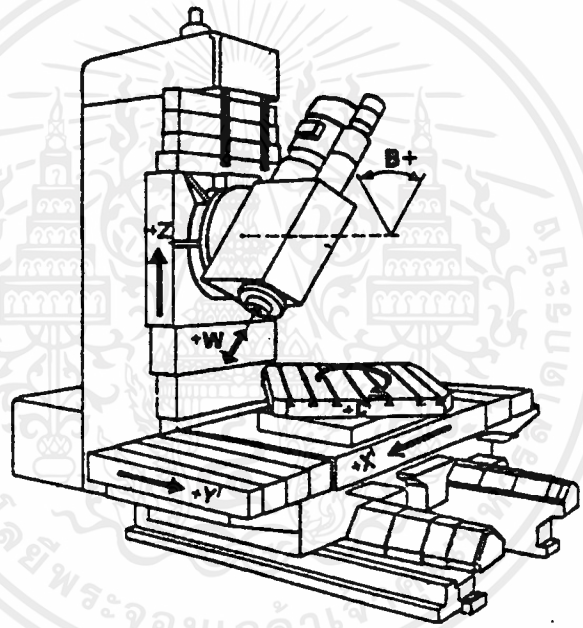


ก) เพลาตั้ง



ข) เพลานอน

รูปที่ 2.1 เครื่องกัด CNC แบบ 3 แกน

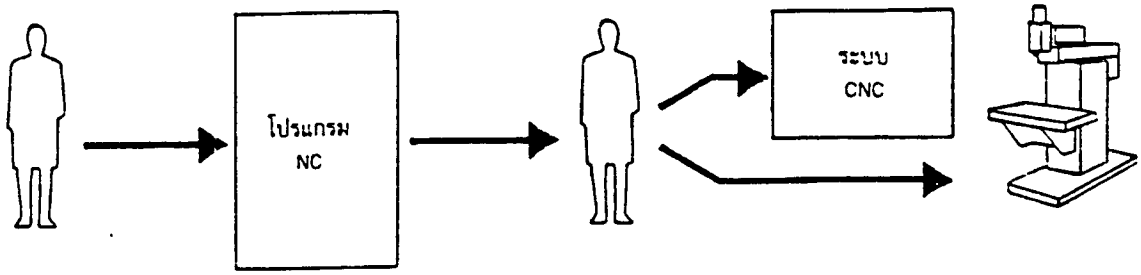


รูปที่ 2.2 เครื่องกัด CNC แบบ 5 แกนที่เอียงเพลามีดได้และมีโต๊ะงานหมุน

2.3 ระบบควบคุมเครื่องจักรกลด้วยตัวเลข

เครื่องจักรกลซีเอ็นซี จะประกอบด้วยองค์ประกอบใหญ่ๆ อยู่ 2 ส่วน คือ

1. เครื่องจักรกล เป็นส่วนที่ทำหน้าที่ตัดเฉือนชิ้นงานตามขั้นตอนการทำงานที่กำหนดไว้
2. ระบบซีเอ็นซี เป็นส่วนที่ทำหน้าที่ควบคุมขั้นตอนการตัดเฉือนทั้งหมด

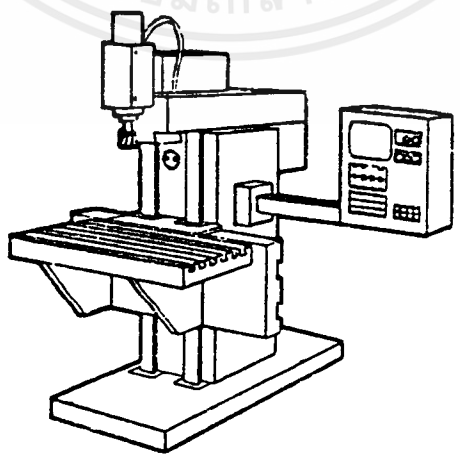


รูปที่ 2.3 องค์ประกอบของเครื่องจักรกล CNC

ข้อมูลเดิมที่อธิบายรายละเอียดของขั้นตอนที่ใช้ในการตัดเฉือนชิ้นงาน จะถูกป้อนเข้าไปในระบบควบคุมของเครื่องจักรกลก่อน ในรูปแบบของโปรแกรมเอ็นซี ซึ่งถูกจัดเตรียมโดยช่างเขียนโปรแกรม ช่างควบคุมเครื่องจะเป็นผู้ป้อนโปรแกรมเข้าไปในระบบควบคุม ซึ่งอาจป้อนด้วยมือผ่านแป้นพิมพ์โดยตรง หรือใช้แถบกระดาษเจาะรู (punched tape) ก็ได้ หลังจากนั้น ก็จะเดินเครื่องทดลองโปรแกรม และสังเกตสภาวะการตัดเฉือนชิ้นงานในแต่ละขั้นตอน บ่อยครั้งที่ช่างควบคุมเครื่องจะต้องจัดเตรียมโปรแกรม หรือเขียนโปรแกรมด้วยตนเอง หรือแก้ไขปรับปรุงโปรแกรมให้มีประสิทธิภาพในการตัดเฉือนสูงสุด ดังนั้น จึงเป็นสิ่งจำเป็นที่ช่างควบคุมเครื่องจะต้องมีความรู้ทั้งระบบควบคุมของเครื่องจักรกลและการเขียนโปรแกรมเอ็นซีด้วย

2.3.1 องค์ประกอบของเครื่องจักรกลที่ควบคุมได้

องค์ประกอบหรือชิ้นส่วนของเครื่องจักรกล ที่ทำหน้าที่เคลื่อนที่เข้าตัดเฉือนชิ้นงาน และองค์ประกอบอื่นๆ ที่ช่วยเสริมการทำงานตัดเฉือนให้สมบูรณ์ยิ่งขึ้น จะถูกควบคุมโดยโปรแกรมเอ็นซี ด้วยวิธีการควบคุมแบบต่าง ๆ กัน



รูปที่ 2.4 เครื่องกัด CNC

ช่างชำนาญงานที่ทำหน้าที่ควบคุมการทำงานของเครื่องจักรกลเอ็นซีหรือซีเอ็นซี จะต้องมีความคุ้นเคยกับหน้าที่การทำงานและขีดจำกัดในการทำงานของเครื่องจักรกลซีเอ็นซีนั้นเป็นอย่างดี ช่างจะใช้วิธีการทำงานแบบง่ายๆ โดยการจับยึดชิ้นงานเข้ากับโต๊ะงานและคาดว่าจะได้วิธีการตัดเฉือนที่ดีที่สุดไม่ได้ ในทางตรงข้ามช่างจะต้องจัดวางแผนขั้นตอนการทำงานไว้ล่วงหน้าเพื่อให้ได้ผลผลิตที่ดี ดังนั้นจึงเป็นสิ่งจำเป็นที่ช่างจะต้องรู้ว่าองค์ประกอบส่วนใดของเครื่องจักรกลซีเอ็นซีที่สามารถควบคุมได้และมีวิธีการควบคุมอย่างไร องค์ประกอบของเครื่องจักรกลเอ็นซีและซีเอ็นซีที่สามารถควบคุมได้ และจะกล่าวถึงในที่นี้ได้แก่

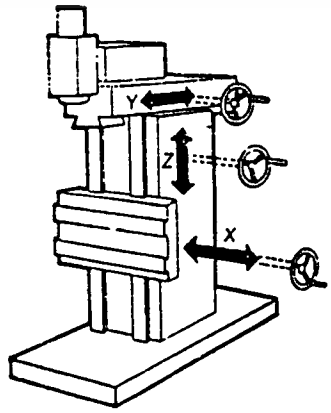
- แนวแกนป้อน (Feed axes)
- การขับป้อน (Feed drives)
- อุปกรณ์วัดขนาด (Measuring devices)
- เพลางาน (Work Spindle)
- อุปกรณ์จับยึดชิ้นงาน (Workpiece holding devices)

1) แนวแกนป้อน (Feed axes)

ในการกล่าวถึงเครื่องจักรกลซีเอ็นซีบ่อยครั้งที่เราจะได้ยินคำว่า แนวแกน (axes) ซึ่งหมายถึง แนวการเคลื่อนที่ขององค์ประกอบของเครื่องจักรกล เช่น โต๊ะงาน เพลาหัวเครื่อง อุปกรณ์ลำเลียงเครื่องมือ (Tool carriers) เป็นต้น

สำหรับเครื่องจักรกลทั่วไป การเคลื่อนที่ในแนวแกนต่างๆ จะเกิดจากการหมุนมือหมุนหรือโยกคันโยกป้อนอัตโนมัติ (Feed levers)

เครื่องจักรกลซีเอ็นซีมีแนวแกนป้อนรวมกันอยู่หลายแนวแกนทำให้ สามารถตัดเฉือนชิ้นงานให้เป็นรูปทรงต่างๆ ที่ต้องการได้ การกำหนดแนวแกนต่างๆของเครื่องจักรกลซีเอ็นซีจะกำหนดตามมาตรฐานสากลภายใต้หัวเรื่อง Coordinate axes and direction of movement for numerically controlled machinery ซึ่งจะกำหนดแนวแกนเหล่านี้โดยใช้ตัวอักษร x,y และ z ดังแสดงในรูป 2.5



รูปที่ 2.5 แท่นเลื่อนแบบ 3 แนวแกน

แนวแกนทั้ง 3 แนวแกนที่แสดงในรูป จะทำให้เกิดการเคลื่อนที่ต่างๆ ดังนี้

แนวแกน x : โต๊ะงานเคลื่อนที่ไปทางซ้ายและขวา

แนวแกน y : เพลาหัวเครื่องเคลื่อนที่เข้าและออก

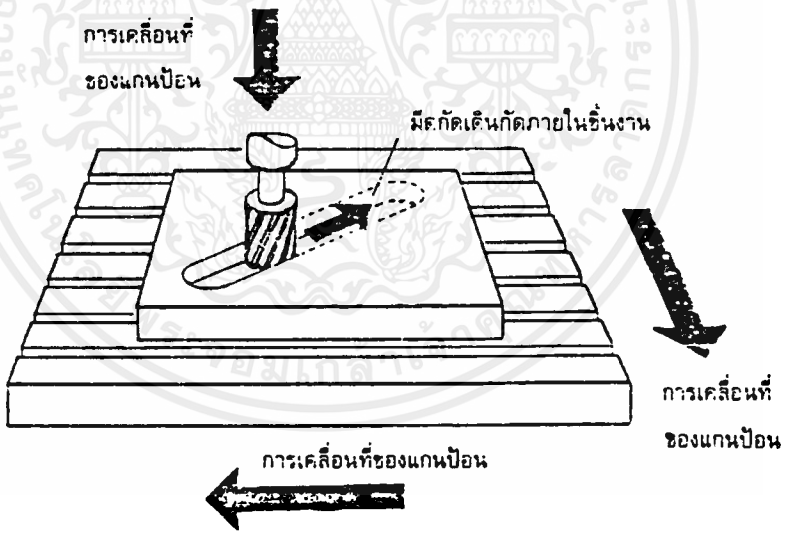
แนวแกน z : โต๊ะงานเคลื่อนที่ขึ้นและลง

เครื่องกัดจะมีแนวแกนป้อนอยู่ 3 แนวแกนด้วยกัน คือ แกน x , y และ z โดยทั่วไปจะมี 2 แกน สำหรับการเคลื่อนที่ของโต๊ะงาน ส่วนแกนที่ 3 จะเป็นการเคลื่อนที่ของเพลาหัวเครื่อง (เพลางาน) ถ้าเครื่องกัดนั้นเป็นแบบโต๊ะงานอยู่กับที่เพลาหัวเครื่องจะเคลื่อนที่ทั้ง 3 แนวแกน

สำหรับเครื่องจักรกลซีเอ็นซี ที่ใช้ผลิตชิ้นงานที่มีรูปทรงซับซ้อนมาก จะมีจำนวนแนวแกนป้อนเพิ่มมากขึ้น

2) การขับป้อน (Feed drives)

การเคลื่อนที่เรียงลำดับกันหรือพร้อม ๆ กันอย่างต่อเนื่องของแนวแกนป้อน จะทำให้เกิดการตัดเฉือนของเครื่องมือในชิ้นงาน ดังรูป 2.6

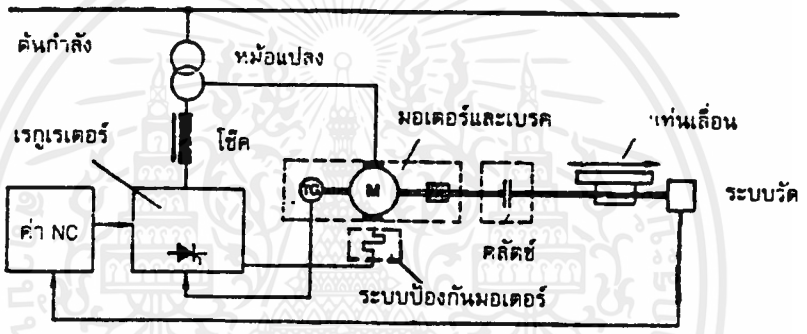


รูปที่ 2.6 การเคลื่อนที่ตัดเฉือนของเครื่องมือตัด

การขับป้อนจะทำให้เกิดการเคลื่อนที่ของแท่นเลื่อน ในขณะตัดเฉือน แท่นเลื่อนอาจพาให้ชิ้นงานเคลื่อนที่หรือคมตัดเคลื่อนที่ก็ได้

ระบบขับป้อนโดยทั่วไปจะใช้มอเตอร์กระแสตรงในการหมุนขับและควบคุมการทำงาน ด้วยวงจรอิเล็กทรอนิกส์จากภายนอก มอเตอร์ชนิดนี้จะสามารถหมุนและเบรคให้หยุดได้ทั้งสองทิศทางขณะตัดเฉือนชิ้นงาน การเคลื่อนที่ป้อนจะต้องเป็นไปอย่าง

สม่ำเสมอและสามารถต้านแรงกระทำจากภายนอกได้ เช่น แรงตัดเฉือน เป็นต้น ด้วยเหตุนี้ ระบบขับป้อนจึงต้องได้รับการออกแบบให้มีความแข็งแกร่งสูง มีการเคลื่อนที่คงที่และสม่ำเสมอ สามารถตอบสนองต่อการเปลี่ยนอัตราป้อนได้อย่างรวดเร็ว นอกจากนี้ในขณะทำงานคมตัดอาจทื่อ หรือการเคลื่อนที่ของแท่นเลื่อนถูกกีดขวาง หรือการเร่งอัตราป้อนให้เคลื่อนที่เร็วและหยุดโดยทันทีทันใด สาเหตุเหล่านี้จะทำให้มอเตอร์รับภาระมากเกินไป (over loading) ซึ่งอาจทำให้มอเตอร์เสียหายได้ ดังนั้น จึงต้องมีการป้องกันอุบัติเหตุเหล่านี้ โดยทั่วไปแล้วจะใช้คลัตช์แบบลูกกลิ้ง (Over running clutch) ร่วมกับวงจรอิเล็กทรอนิกส์ ปัจจัยหนึ่งที่จะทำให้ระบบขับป้อนสามารถทำงานได้อย่างมีประสิทธิภาพก็คือ การเลือกใช้อุปกรณ์ในระบบขับป้อนให้เหมาะสมกับการทำงานของเครื่องจักรและการออกแบบวงจรควบคุมการทำงานที่มีประสิทธิภาพดังแสดงในรูป2.7

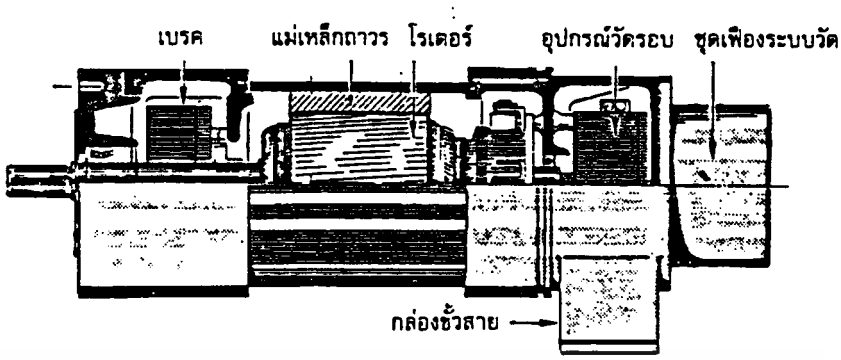


รูปที่ 2.7 Diagram ระบบขับป้อน

2.1) มอเตอร์

เครื่องจักรกลเอ็นซีสมัยใหม่จะออกแบบใช้ระบบขับป้อนแบบเซอโว (servo drives) ทำให้สามารถปรับอัตราป้อนและความเร็วรอบได้โดยไม่มีขีดจำกัดของขั้นความเร็วและอัตราป้อน มอเตอร์ที่ใช้ในระบบขับป้อนโดยทั่วไปจะมีอยู่ 3 ชนิดด้วยกันคือ

ก) <u>มอเตอร์กระแสตรง (DC motors)</u> ลักษณะสร้างของมอเตอร์กระแสตรงจะใช้เป็นแม่เหล็กถาวรทีมี 4 , 6 หรือ 8 ขั้ว ประกอบด้วยระบบเบรค (brake) แกนมอเตอร์ (Rotor) อุปกรณ์วัดรอบ (Tachogenerator) และอุปกรณ์วัด (Measuring box) ดังแสดงในรูป 2.8



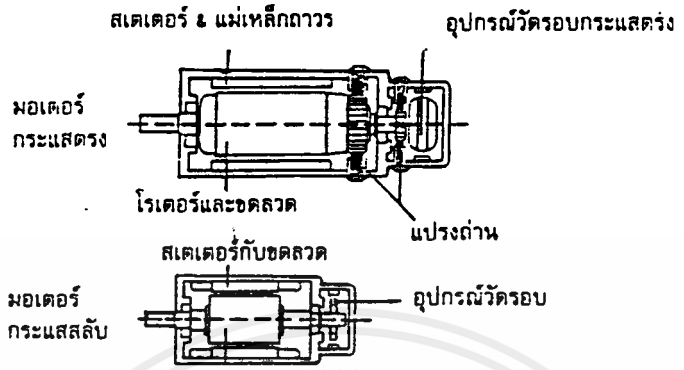
รูปที่ 2.8 ส่วนประกอบของมอเตอร์กระแสตรง

การใช้มอเตอร์กระแสตรง ทำให้สามารถปรับอัตราป้อนได้ละเอียดและมีวงจร ควบคุมที่ไม่ซับซ้อน แต่ก็มีข้อเสียตรงที่มอเตอร์ชนิดนี้ต้องใช้แปรงถ่าน ซึ่งจะต้องคอย ทำความสะอาดและเปลี่ยนเมื่อแปรงถ่านหมด นอกจากนี้ แปรงถ่านยังทำให้แกนมอเตอร์สึกหรออันเป็นผลทำให้กำลังมอเตอร์ลดลง ข้อเสียอีกประการหนึ่งก็คือ หากต้องการกำลังขับสูง มอเตอร์ก็จะมีขนาดใหญ่ด้วย และเมื่อใช้ความเร็วรอบสูงๆ จะทำให้แรงบิดลดลง ดังนั้น จึงมักใช้กับเครื่องจักรกลเอ็นซีขนาดเล็กและขนาดกลาง

ข) มอเตอร์แบบเป็นขั้น (Stepping motors) เป็นมอเตอร์ที่ทำงานแบบต่อเนื่อง โดยการแปลงคลื่นสัญญาณที่ป้อนเข้าไปในระบบให้เป็นการเคลื่อนที่เชิงมุม การหมุนในแต่ละมุมหรือขั้นที่เปลี่ยนไป 1 ขั้นจะเท่ากับ 1 คลื่นสัญญาณ ดังนั้น ตำแหน่งของเพลาจะถูกกำหนดโดยจำนวนคลื่นสัญญาณที่ป้อนเข้าไปในระบบ และความเร็วในการหมุนของเพลาจะวัดเป็นจำนวนขั้นต่อวินาที (Steps per second) ซึ่งจะเท่ากับความถี่ของคลื่นสัญญาณที่ป้อนเข้าในระบบที่วัดเป็นจำนวนคลื่นสัญญาณต่อวินาที (pulses per second) ความเที่ยงตรงของระบบจะขึ้นอยู่กับความสามารถของมอเตอร์ในการแบ่งขั้นการหมุนตามจำนวนคลื่นสัญญาณที่ป้อนเข้าในระบบ แรงบิดของมอเตอร์ชนิดนี้จะลดลงเมื่อความเร็วในการหมุนแบ่งเพิ่มขึ้น ดังนั้น จึงเหมาะสำหรับเครื่องจักรกลเล็กๆ ที่ไม่ต้องใช้กำลังขับมาก เช่น เครื่องพลอตเตอร์ (Plotter machine) เป็นต้น

ค) มอเตอร์กระแสสลับ (Alternate-current motor) ส่วนมากจะเป็นมอเตอร์แบบซิงโครนัส (Synchronous motor) ข้อดีของมอเตอร์ชนิดนี้คือ ไม่ต้องใช้แปรงถ่าน ทำให้สามารถลดงานบำรุงรักษาได้มาก และมอเตอร์ขนาดเดียวกันเมื่อเปรียบเทียบกับมอเตอร์กระแสตรง จะสามารถให้แรงบิดได้ดีกว่า และมีขนาดเล็กกว่าด้วย ดังแสดงในรูป 2.9

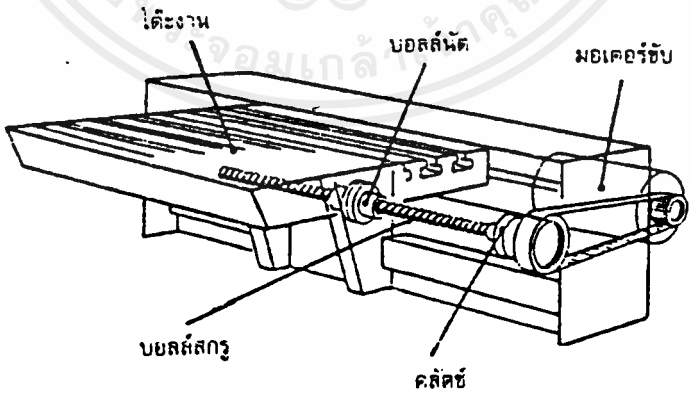
ส่วนข้อเสียของมอเตอร์แบบนี้คือ วงจรควบคุมจะมีความซับซ้อนมากกว่าวงจรควบคุมมอเตอร์กระแสตรง



รูปที่ 2.9 การเปรียบเทียบลักษณะสร้างและขนาดของมอเตอร์กระแสตรงกับมอเตอร์กระแสสลับแบบ 3 เฟส

2.2) บอลล์สกรู (Ball screws)

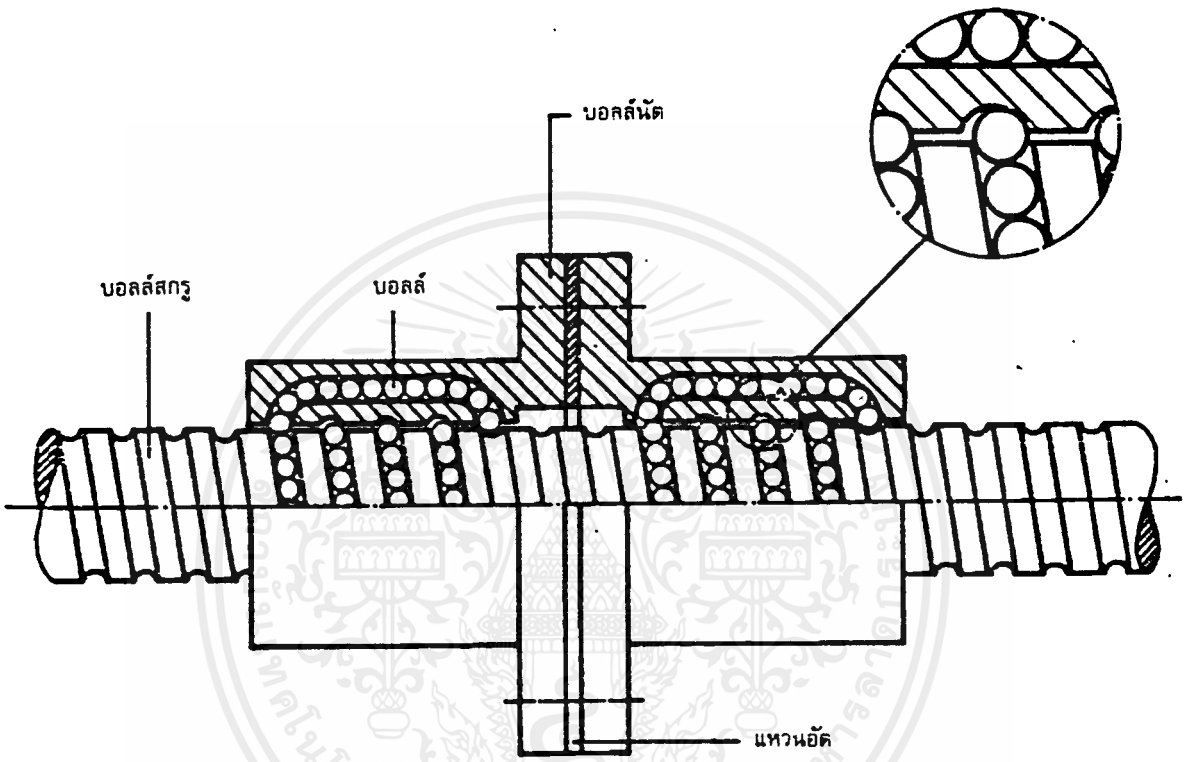
หัวใจของระบบขับป้อนของเครื่องจักรกลซีเอ็นซี ก็คือ การส่งกำลังขับด้วย บอลล์สกรู ซึ่งจะมีลูกบอลล์ไหลหมุนเวียนอยู่ตลอดเวลา บอลล์สกรูจะประกอบด้วยสกรูกับนัตที่มีลักษณะเป็นเกลียวกลม ร่องเกลียวกลมบนสกรูและในนัตจะชุบแข็งและเจียระไนผิวเรียบมันเพื่อลดความฝืดและเพิ่มความเที่ยงในการเคลื่อนที่ ดังแสดงในรูป 2.10



รูปที่ 2.10 การขับป้อนของโต๊ะงาน

เมื่อมอเตอร์หมุนขับสกรู นัตก็จะเคลื่อนที่ไปตลอดความยาวของสกรู พาให้แท่นเลื่อนและโต๊ะงานเคลื่อนที่ไปตามรางเลื่อน

ภายในของตัวนัตจะประกอบไปด้วยชุดของลูกบอลล์จำนวนมาก ดังแสดงในรูป2.11 ทำให้มั่นใจได้ว่าความเสียดทานในการส่งกำลังขับจากสกรูไปยังแท่นเลื่อนจะมีน้อยมาก นัตจะถูกแบ่งออกเป็นสองซีก และ ขันประกอบยึดเข้าด้วยกัน โดยมีการเตรียมอัดแรงไว้ก่อน (preloaded) ทำให้สามารถลดระยะคลอน (backlash) ให้เหลือน้อยที่สุดจนแทบจะไม่มีเลยได้ทำให้การเคลื่อนที่ของแท่นเลื่อนมีความเที่ยงตรงสูงสามารถทำงานซ้ำๆกัน ได้



รูปที่ 2.11 ลักษณะสร้างภายในของชุดบอลล์สกรู

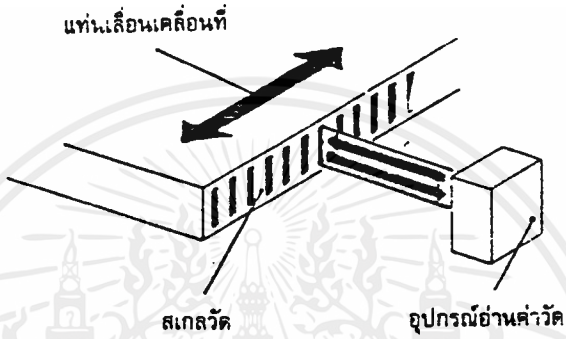
การต่อกำลังระหว่างมอเตอร์กับบอลล์สกรู จะมีชุดคลัตช์ความฝืดเป็นตัวเชื่อม ซึ่งนอกจากจะมีหน้าที่ต่อกำลังขับแล้ว ยังมีหน้าที่ป้องกันอุบัติเหตุที่เกิดจากแท่นเลื่อนหรือโต๊ะงานชนหรือกระแทกกับสิ่งกีดขวางไม่ให้เครื่องจักรกลซีเอ็นซีเกิดความเสียหายมากเกินไป กล่าวคือ เมื่อมีการชน หรือกระแทกกันขึ้นจนแรงมากถึงค่าหนึ่ง ชุดคลัตช์ก็จะตัดระบบการส่งกำลังขับระหว่างมอเตอร์กับตัวบอลล์สกรูทันที

3) ระบบวัดขนาด (Measuring System)

การเคลื่อนที่ไปที่ตำแหน่งต่างๆ ในแต่ละแนวแกนของแท่นเลื่อน จะถูกส่งไปยังระบบควบคุมโดยระบบวัดขนาด การวัดตำแหน่งของแท่นเลื่อนสามารถที่จะวัดได้ทั้งโดยตรง (Direct Measurement) และโดยทางอ้อม (Indirect Measurment)

3.1) <u>การวัดตำแหน่งโดยตรง</u>

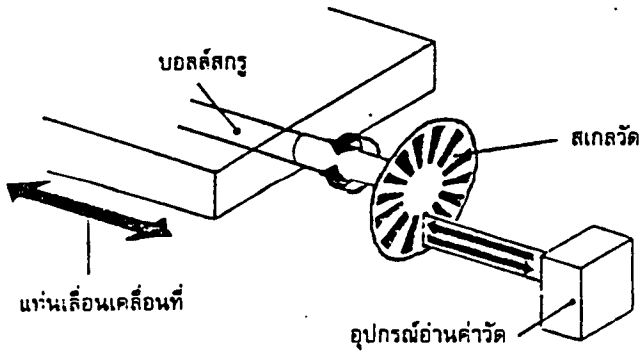
วิธีนี้จะใช้สเกลวัด (measuring scale) ยึดติดกับแท่นเลื่อนหรือโต๊ะงานโดยตรง ดังแสดงในรูป 2.12 ข้อดีของวิธีวัดแบบนี้ก็คือ ความไม่เที่ยงตรงของสกรูนำเลื่อน (leadscrew) ระบบขับจะไม่มีผลกระทบต่อค่าที่อ่านได้ อุปกรณ์อ่านค่าวัด(Measuring valve resolver) จะอ่านข้อมูลในการวัดจากขีดสเกลวัด (Measuring scale grid) และแปลงข้อมูลนี้เป็นสัญญาณไฟฟ้าและส่งกลับไปยังระบบควบคุม



รูปที่ 2.12 การวัดตำแหน่งโดยตรง

3.2) <u>การวัดตำแหน่งทางอ้อม</u>

การเคลื่อนที่ของแท่นเลื่อนจะได้รับกำลังขับมาจากการหมุนของบอลล์สกรูอุปกรณ์เปลี่ยนค่าวัด (Resolver) จะบันทึกการเคลื่อนที่หมุนของแผ่นจานสัญญาณ (pulse disc) ที่ต่อติดอยู่กับบอลล์สกรู และส่งต่อไปยังระบบควบคุมของเครื่อง ระบบควบคุมก็จะใช้สัญญาณ ได้รับนี้ไปคำนวณหาระยะทางการเคลื่อนที่ของแท่นเลื่อนจากสัญญาณการหมุน (rotation pulses) ของแผ่นจานสัญญาณ ดังแสดงในรูป 2.13



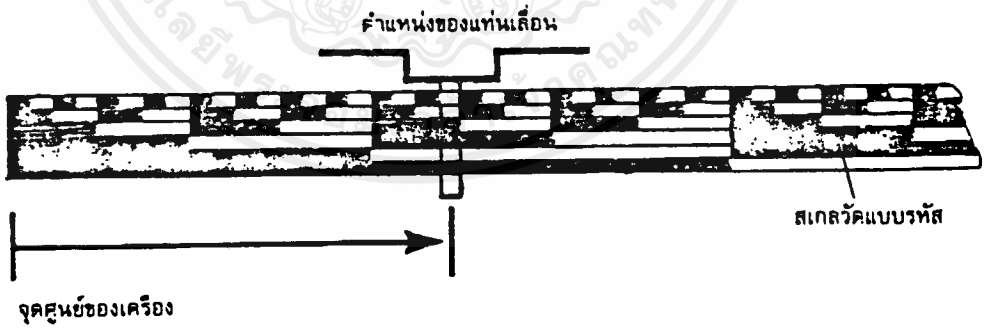
รูปที่ 2.13 การวัดตำแหน่งทางอ้อม

นอกจากการวัดตำแหน่งทางตรงและทางอ้อมแล้ว ในระบบการวัดขนาดของเครื่องจักรกลเอ็นซีที่ต้องการให้การวัดตำแหน่งมีความเที่ยงตรงตลอดแนวแกนป้อน จะต้องต่อระบบขับป้อนเข้ากับอุปกรณ์วัดที่เหมาะสม อุปกรณ์วัดโดยทั่วไปจะประกอบด้วยสเกลกับอุปกรณ์อ่านค่าวัดที่สามารถอ่านสเกลได้ สเกลที่ใช้ในอุปกรณ์วัดมีอยู่ด้วยกัน 2 ชนิด คือ สเกลวัดแบบรหัส (coded measuring scale) กับสเกลวัดแบบช่อง (division grid) การใช้สเกลวัดทั้ง 2 ชนิดนี้จะขึ้นอยู่กับวิธีการวัดตำแหน่ง (position measurement) วิธีการวัดตำแหน่งที่นิยมใช้กันทั่วไปมีอยู่ 2 วิธีคือ การวัดตำแหน่งแบบสัมบูรณ์ absolute position measurement) กับการวัดตำแหน่งแบบต่อเนื่อง หรือแบบลูกโซ่ (incremental or chain position measurement) ซึ่งมีความแตกต่างกันดังรายละเอียดต่อไปนี้

3.3) <u>การวัดตำแหน่งแบบสัมบูรณ์</u>

คำว่า **สัมบูรณ์** (absolute) ที่ใช้ร่วมกับการวัดตำแหน่งนี้จะหมายความว่า ค่าตำแหน่งต่างๆ สามารถวัดได้ตลอดเวลาและเป็นอิสระจากสถานะของเครื่องและระบบควบคุม ทั้งนี้เพราะค่าต่างๆเหล่านี้จะวัดอ้างอิงจากจุดศูนย์อ้างอิง (fixed zero datum) เสมอ

การวัดตำแหน่งแบบสัมบูรณ์ ดังแสดงในรูป 2.14 จะใช้สเกลวัดแบบรหัส (coded measuring scale) ซึ่งจะชี้ตำแหน่งของแท่นเลื่อนที่ถูกต้องตลอดเวลา โดยอ้างอิงจากตำแหน่งจุดศูนย์ของเครื่อง (Machine zero point) ซึ่งเป็นตำแหน่งศูนย์ที่มีจุดอ้างอิงที่แน่นอนและถาวรของเครื่องจักรกลเอ็นซี



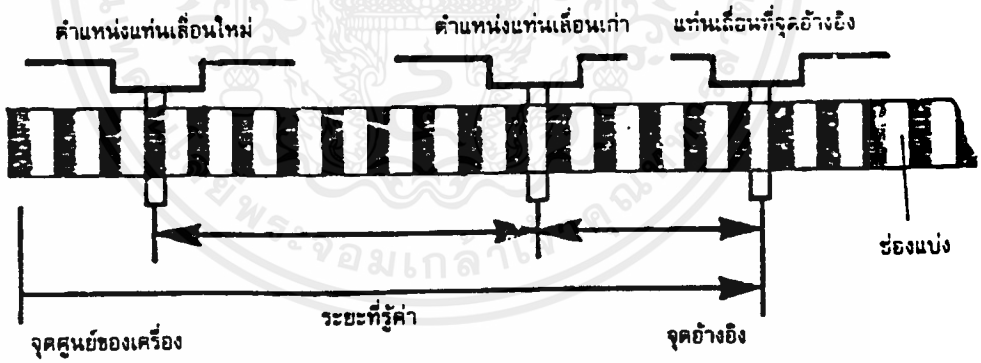
รูปที่ 2.14 การวัดตำแหน่งแบบสัมบูรณ์

ข้อสำคัญของการใช้วิธีการวัดตำแหน่งแบบนี้ก็คือ ความยาวของช่วงอ่านค่าวัดของสเกลจะต้องยาวกว่าระยะเลื่อนทำงานของแท่นเลื่อนเพื่อให้ระบบควบคุมของเครื่องสามารถอ่านค่าวัดได้ทุกตำแหน่งสเกลนี้จะใช้รหัสเป็นระบบตัวเลขฐานสอง (Binary system)

3.4) <u>การวัดตำแหน่งแบบต่อเนื่อง</u>

คำว่า ต่อเนื่อง (incremental) แปลว่า ระยะเลื่อนสั้นๆ ตามความยาวที่กำหนดดังนั้นในการวัดตำแหน่งอาจจะเป็นการเพิ่มหรือลดขนาดความยาวในการเคลื่อนที่ที่วัดอยู่ก็ได้ ในระหว่างการเคลื่อนที่ของแท่นเลื่อนระบบควบคุมจะทำการนับจำนวนส่วนแบ่ง (divisions)ที่ตำแหน่งใหม่แตกต่างจากตำแหน่งก่อนหน้านี้เสมอ

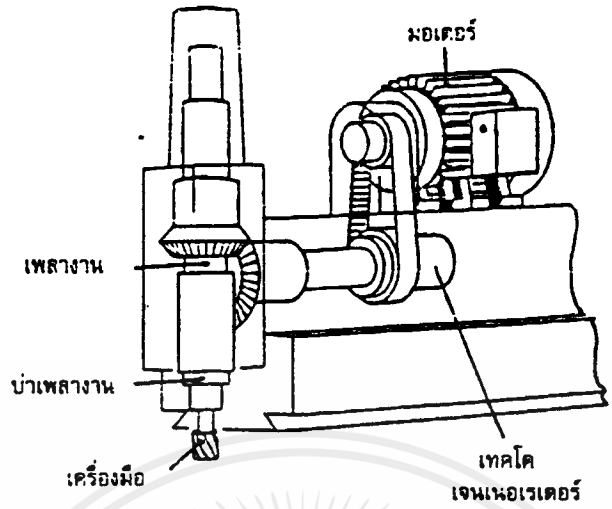
วิธีการวัดตำแหน่งแบบต่อเนื่องที่แสดงในรูป 2.15 สเกลวัดจะแบ่งเป็นช่อง (grid) แบบง่ายๆโดยที่แต่ละช่องจะมีพื้นที่สว่างกับมืดสลับกันไป เมื่อแท่นเลื่อนเคลื่อนที่ช่องนี้ก็จะวิ่งผ่านอุปกรณ์เปลี่ยนค่าวัด (resolver) ซึ่งจะทำหน้าที่นับ จำนวนช่องพื้นที่สว่างและมืด จากนั้นก็จะส่งเป็นสัญญาณไฟฟ้าไปยังระบบควบคุมของเครื่อง ระบบควบคุมก็จะนำสัญญาณนี้มาคำนวณหาตำแหน่งสุดท้ายของแท่นเลื่อนที่แตกต่างจากตำแหน่งก่อนหน้านี้ ในทางปฏิบัติที่ต้องการให้วิธีการวัดแบบนี้ทำงานได้อย่างถูกต้อง เมื่อเริ่มเปิดสวิตช์ระบบควบคุมของเครื่องควรจะเลื่อนไปยังจุดที่ทราบค่าระยะห่างจากจุดศูนย์ของเครื่องจุดนี้จะเรียกว่า "จุดอ้างอิง" (Reference Point) หลังจากที่แท่นเลื่อนในแนวแกนต่างๆ เลื่อนไปยังจุดอ้างอิงแล้ว อุปกรณ์อ่านค่าวัดก็จะสามารถทำหน้าที่วัดตำแหน่งด้วยช่องสเกลได้



รูปที่ 2.15 การวัดตำแหน่งแบบต่อเนื่อง

4) เพลางาน (Work Spindle)

เพลางานเป็นชิ้นส่วนหรือองค์ประกอบของเครื่องจักรกลที่มีความสำคัญมาก มีหน้าที่หลักในการทำงาน คือ จะทำหน้าที่ขับพาให้เครื่องมือ เช่น มีดกัด ดอกสว่าน เป็นต้น หมุนตัดเฉือนชิ้นงาน

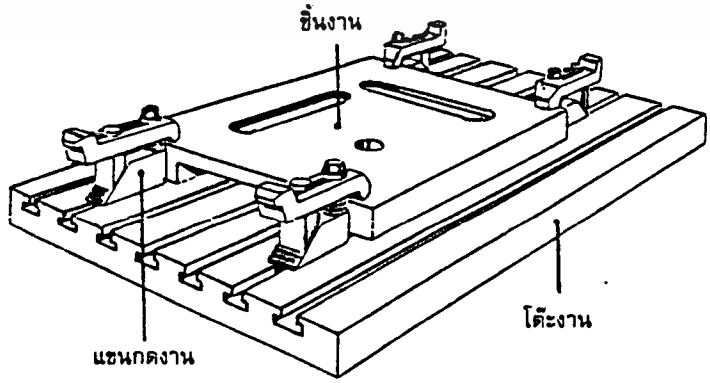


รูปที่ 2.16 เพลางานของเครื่องกัด

5) อุปกรณ์จับยึดชิ้นงาน (Workpiece holding devices)

อุปกรณ์จับยึดชิ้นงานจะจัดเตรียมไว้สำหรับยึดชิ้นงานเข้ากับโต๊ะงานในงานกัด (Milling) สามารถเลือกใช้อุปกรณ์จับยึดชิ้นงานแบบต่างๆ กันได้ดังนี้

- แขนกดชิ้นงาน
- แองเกิล เพลท (angle plate)
- ปากกาจับชิ้นงาน
- แท่นแม่เหล็ก
- อุปกรณ์จับชิ้นงานที่ออกแบบเฉพาะงาน



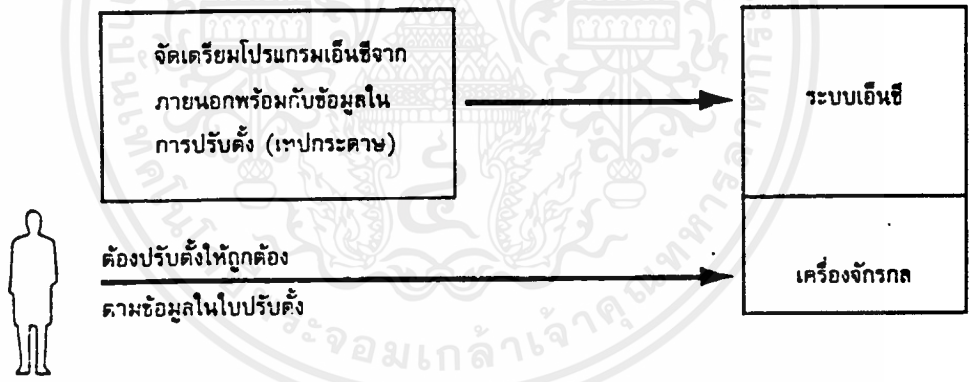
รูปที่ 2.17 แขนกดจับชิ้นงานกัด

2.4 ระบบควบคุมซีเอ็นซี (CNC Control System)

2.4.1 หน้าที่การทำงานที่โปรแกรมได้ (Programmable Functions)

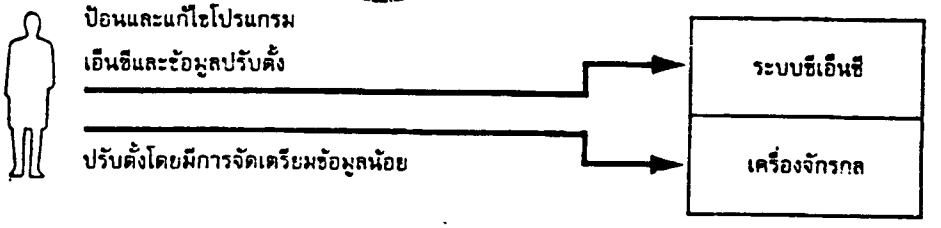
ในปัจจุบัน ระบบควบคุมการทำงานของเครื่องจักรกลสมัยใหม่เกือบทั้งหมด จะควบคุมด้วยระบบซีเอ็นซี แต่เนื่องจากยังคงอ้างอิงถึงโปรแกรมเอ็นซี (NC Program) และเทคโนโลยีเอ็นซี (NC Technology) อยู่ ดังนั้น จึงเป็นเรื่องสำคัญที่จะต้องรู้ถึงความแตกต่างในการทำงานระหว่างระบบเอ็นซีกับระบบซีเอ็นซี

ระบบเอ็นซี (NC System) ดังแสดงในรูป 2.18 จะมีระบบควบคุมประกอบอยู่กับเครื่องจักรกลซึ่งจะต้องจัดเตรียมโปรแกรมเอ็นซีจากภายนอกก่อน แล้วจึงป้อนเข้าไปในระบบควบคุม โดยอาศัยสื่อข้อมูล (data carries) เช่น เทปกระดาษ (Punched tape) เป็นต้น โปรแกรมเอ็นซีที่ป้อนเข้าไปในระบบควบคุมของเครื่องจะถูกนำไปใช้เพื่อสั่งให้เครื่องเริ่มทำงานและหยุดชั่วคราวได้ แต่จะไม่สามารถแก้ไขโปรแกรมโดยช่างควบคุมเครื่องได้ ขนาดของเครื่องมือและอุปกรณ์จับยึดชิ้นงานจะถูกเลือกใช้ในขณะเขียนโปรแกรมไว้ก่อน และกำหนดไว้ในใบปรับตั้ง (set-up sheet) ซึ่งช่างควบคุมเครื่องจะต้องจัดเตรียมและประกอบยึดเครื่องมือ ตลอดจนอุปกรณ์จับยึดชิ้นงานให้ถูกต้องตามข้อมูลที่กำหนดไว้ในใบปรับตั้ง



รูปที่ 2.18 ระบบ NC

ระบบซีเอ็นซี (CNC system) จะมีคอมพิวเตอร์ประกอบอยู่ด้วย ดังนั้น ช่างควบคุมเครื่องไม่เพียงแต่จะสามารถใช้โปรแกรมเอ็นซีสั่งให้เครื่องจักรทำงานได้เท่านั้น แต่จะยังสามารถเขียนและป้อนโปรแกรมด้วยตนเอง ตลอดจนการแก้ไขโปรแกรมได้หลังจากป้อนเข้าไปในระบบควบคุมของเครื่องแล้ว ดังแสดงในรูป 2.19



รูปที่ 2.19 ระบบ CNC

ขนาดต่างๆของเครื่องมือตัดและอุปกรณ์จับยึดชิ้นงาน สามารถที่จะเลือกใช้และป้อนเข้าไปในระบบควบคุมซีเอ็นซี ขณะทำการปรับตั้ง (setting-up) และเป็นอิสระจากตัวโปรแกรมเอ็นซี ขนาดต่างๆ ของเครื่องมือจะถูกนำไปใช้โดยอัตโนมัติในขณะทำการตัดเฉือน ด้วยเหตุนี้ช่างควบคุมเครื่องจึงไม่จำเป็นต้องมีข้อมูลในการปรับตั้งมากและสามารถที่จะเลือกใช้เครื่องมือและอุปกรณ์จับยึดชิ้นงานได้ด้วยตนเอง

หากพิจารณาถึงภาษาโปรแกรม (Programming language) และเทคโนโลยีทางด้านการตัดเฉือนของเครื่องจักรกลที่ใช้ในระบบเอ็นซีกับซีเอ็นซีแล้วจะไม่แตกต่างกัน

2.4.2 ชนิดของการควบคุม (Control modes)

ลักษณะการควบคุมการเคลื่อนที่ทำงานของแท่นเลื่อนต่าง ๆ ในเครื่องจักรกลเอ็นซีและซีเอ็นซี จะมีการเคลื่อนที่อยู่ 2 ลักษณะ คือ

การเคลื่อนที่ในแนวเส้นตรง(Linear Interpolation หรือ straight lineInterpolation) การเคลื่อนที่ลักษณะนี้ ระบบซีเอ็นซีจะคำนวณหาตำแหน่งของจุดต่างๆที่ต่อกันเป็นลูกโซ่ในแนวเส้นตรงระหว่างตำแหน่งของเครื่องมือ 2 ตำแหน่งในขณะที่เครื่องมือเคลื่อนที่จากจุดหนึ่งไปยังอีกจุดหนึ่งนั้น ระบบควบคุมซีเอ็นซีจะตรวจสอบและแก้ไขแนวแกนในการเคลื่อนที่ให้ถูกต้องอยู่ตลอดเวลาทำให้การเคลื่อนที่ของเครื่องมือไม่ผิดพลาดหรือคลาดเคลื่อนออกจากจุดต่อของเส้นตรงมากกว่าค่าพิกัดความเผื่อของเครื่องที่กำหนดไว้

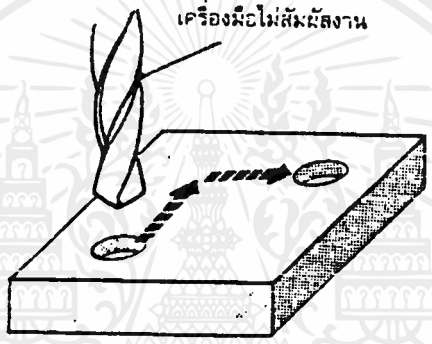
การเคลื่อนที่ในแนวเส้นโค้ง (Circular Interpolation) ระบบควบคุมซีเอ็นซีจะคำนวณหาตำแหน่งของจุดต่างๆ ที่ต่อกันเป็นเส้นโค้งตามขนาดรัศมีที่กำหนดระหว่างตำแหน่งของเครื่องมือที่กำหนดไว้ 2 ตำแหน่ง ระบบควบคุมจะอาศัยจุดเหล่านี้ในการตรวจสอบและแก้ไขแนวการเคลื่อนที่ของเครื่องมือให้ถูกต้องและอยู่ภายในพิกัดความเผื่อของเครื่องจักรกลที่กำหนด

ในระบบควบคุมซีเอ็นซีจะแบ่งการควบคุมการเคลื่อนที่ทั้งสองลักษณะตามลักษณะการเคลื่อนที่ป้อนออกเป็น 3 ชนิด คือ

1) การควบคุมจุดต่อจุด (Point to point control)

การควบคุมแบบนี้จะควบคุมการเคลื่อนที่ของเครื่องมือระหว่างจุดสองจุดที่โปรแกรมไว้ในลักษณะการเคลื่อนที่เร็ว (Rapid traverse) โดยที่เครื่องมือจะต้องไม่สัมผัสชิ้นงาน ดังแสดงในรูป 2.20

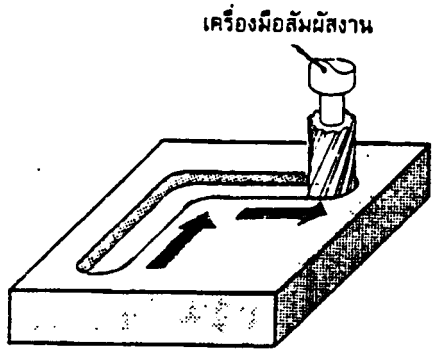
แนวแกนในการเคลื่อนที่ขึ้นอยู่กับชนิดของระบบควบคุม กล่าวคือมอเตอร์ขับของระบบป้อนอาจจะเริ่มทำงานหลายๆ แนวแกนพร้อมกัน หรือทำงานทีละแนวแกนจนกว่าจะเคลื่อนที่ถึงตำแหน่งของเครื่องมือที่โปรแกรมไว้ ทำให้ไม่สามารถควบคุมทางเดินของเครื่องมือ (Tool path) ได้ การควบคุมแบบจุดต่อจุดมักจะใช้กับเครื่องเจาะ (drilling machine) เครื่องเชื่อมจุด(spot drilling) เป็นต้น



รูปที่ 2.20 การควบคุมจุดต่อจุด

2. การควบคุมการตัดเฉือนแนวเส้นตรง (Straight-cut control)

การควบคุมชนิดนี้ นอกจากจะสามารถควบคุมการเคลื่อนที่ของเครื่องมือแบบเคลื่อนที่เร็วได้แล้ว ยังสามารถควบคุมการเคลื่อนที่ของเครื่องมือในแนวขนานกับแนวแกนของเครื่องจักรกล ตามค่าอัตราป้อนที่ต้องการได้อีกด้วย แต่จะสามารถควบคุมการเคลื่อนที่ได้ครั้งละ 1 แนวแกนเท่านั้น การเคลื่อนที่ของเครื่องมือจะถูกควบคุมด้วยอัตราป้อนและความยาวในการเคลื่อนที่ ดังแสดงในรูป 2.21 ระบบการควบคุมการตัดเฉือนแนวเส้นตรงชนิดนี้จะใช้กับเครื่องกัดและเครื่องกลึงแบบง่ายๆ

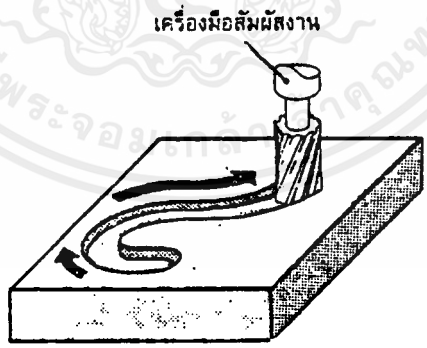


รูปที่ 2.21 การควบคุมแบบเส้นตรง

3) การควบคุมตามเส้นขอบรูป (Contouring control)

การควบคุมแบบนี้จะสามารถควบคุมการเคลื่อนที่ทำงานได้ดังนี้

- ควบคุมเครื่องมือให้เคลื่อนที่ไปยังตำแหน่งที่ต้องการแบบเคลื่อนที่เร็วได้
- ควบคุมเครื่องมือให้เคลื่อนที่ขนานกับแนวแกนไปยังตำแหน่งที่ต้องการตามค่าอัตราป้อนได้
- ควบคุมเครื่องมือให้เคลื่อนที่ไปยังตำแหน่งใดๆ บนชิ้นงานที่กำหนดในแนวเส้นตรงและเส้นโค้งตามค่าอัตราป้อนได้



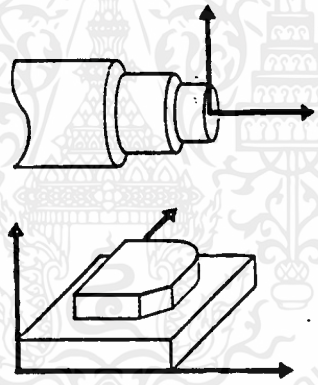
รูปที่ 2.22 การควบคุมตามเส้นขอบรูป

การควบคุมตามเส้นขอบรูปนี้ ยังสามารถแยกย่อยได้อีกเป็น 3 ระดับ ซึ่งขึ้นอยู่กับความสามารถของระบบควบคุมคือ ความสามารถในการควบคุมการเคลื่อนที่ของเครื่องมือได้ 2 หรือ 3 แกน พร้อมๆ กัน โดยไม่คำนึงถึงว่าเครื่องจักรกลซีเอ็นซีนั้นๆ จะมีกี่แนว

แกน จุดสำคัญอยู่ที่ว่าจะสามารถควบคุมแนวแกนป้อนได้พร้อมๆกันกี่แนวแกน ระดับความสามารถทั้งสามระดับของระบบควบคุมมีรายละเอียดดังนี้

ก) <u>การควบคุมตามเส้นขอบรูปแบบ 2 แกน (2D Contouring control)</u> ระบบควบคุมจะสามารถควบคุมเครื่องมือให้เคลื่อนที่ในระนาบ (plane) ที่กำหนดเฉพาะได้ 2 แนวแกนพร้อมๆ กัน ทำให้สามารถเคลื่อนที่ได้ทั้งในแนวเส้นตรงและเส้นโค้ง แต่จะไม่สามารถเปลี่ยนระนาบในการทำงานได้ นั่นหมายความว่า 2 แนวแกนที่เคลื่อนที่พร้อมกันได้นั้น จะถูกกำหนดตายตัวจากบริษัทผู้ผลิต ไม่สามารถเปลี่ยนแนวแกนได้

ถ้าเครื่องจักรกลซีเอ็นซีนั้นมี 3 แนวแกน และระบบควบคุมเป็นแบบการควบคุมตามเส้นขอบรูป 2 แกนแล้ว แนวแกนที่ 3 จะถูกควบคุมเป็นอิสระจาก 2 แนวแกนข้างต้น เช่น เครื่องกัด (Milling Machine) จะใช้แนวแกนหนึ่งสำหรับการเคลื่อนที่ป้อนกินลึก ส่วนอีก 2 แนวแกนจะใช้สำหรับการเดินกัดตามเส้นขอบรูป เป็นต้น ดังแสดงในรูป 2.23

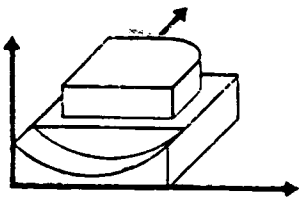


รูป 2.23 การควบคุมตามเส้นขอบรูปแบบ 2 แกน

ข) <u>การควบคุมตามเส้นขอบรูปแบบ 2 แกนครึ่ง (2 ½ -D Contouring control)</u>

ระบบควบคุมแบบนี้จะควบคุมเครื่องมือให้เคลื่อนที่ในแนวเส้นตรงและเส้นโค้งบนระนาบใดๆ ที่ต้องการก็ได้ แต่จะสามารถควบคุมการเคลื่อนที่ได้พียง 2 แนวแกนพร้อมกันเท่านั้น

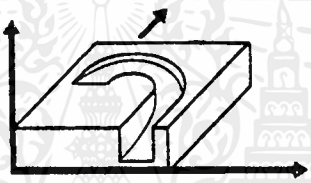
สำหรับเครื่องจักรกลซีเอ็นซีที่มี 3 แนวแกนคือ X,Y และ Z แนวแกนคู่ใดคู่หนึ่ง คือ X/Y หรือ Y/Z หรือ X/Z จะสามารถควบคุมให้เคลื่อนที่พร้อมกันได้นั่นหมายความว่า สำหรับเครื่องกัด การเคลื่อนที่ป้อนกินลึก (In-feed)สามารถทำในแนวแกนใดๆ ได้ทั้ง 3 แนวแกน ส่วนอีก 2 แนวแกนที่เหลือจะใช้สำหรับเดินกัดตามเส้นขอบรูป ดังแสดงในรูป 2.24



รูปที่ 2.24 การควบคุมเส้นขอบรูปแบบ 2 แกนครึ่ง

ค) <u>การควบคุมตามเส้นขอบรูปแบบ 3 แกน (3-D Contouring control)</u>

ระบบควบคุมจะสามารถควบคุมเครื่องมือให้เคลื่อนที่ในแนวเส้นตรงและเส้นโค้งได้พร้อมกันทั้ง 3 แนวแกน เป็นลักษณะ 3 มิติได้ ดังแสดงในรูปที่ 2.25



รูปที่ 2.25 การควบคุมเส้นขอบรูปแบบ 3 แกน

การควบคุมตามเส้นขอบรูปจะสามารถใช้เป็นการควบคุมการตัดเฉือนแนวเส้นตรงได้และการควบคุมการตัดเฉือนแนวเส้นตรงจะใช้เป็นการควบคุมแบบจุดต่อจุดได้ แต่ในทางกลับกันจะกระทำไม่ได้

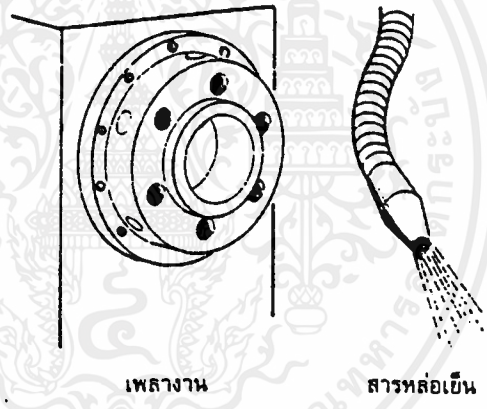
2.4.3 <u>การควบคุมหน้าที่การทำงานของเครื่องจักรกล(Control of machine function)</u>

กระบบควบคุมซีเอ็นซีนอกจากจะสามารถควบคุมการเคลื่อนที่ของเครื่องมือ ตามรูปทรงเรขาคณิตของชิ้นงานแล้ว ยังสามารถควบคุมหน้าที่การทำงานอื่นๆที่ช่วยเสริมการทำงานตัดเฉือนของเครื่องจักรกลให้เหมาะสมกับสภาวะการทำงานในขณะนั้นได้อีกด้วย จำนวนหน้าที่การทำงานและวิธีการควบคุมจะไม่ขึ้นอยู่กับตัวเครื่องจักรกลเพียงอย่างเดียว แต่ยังขึ้นอยู่กับชนิดระบบควบคุมอีกด้วย

ตัวอย่างหน้าที่การทำงานต่างๆ ที่จำเป็นจะต้องโปรแกรมเพื่อช่วยในการทำงาน ดังแสดงในรูป 2.26 มีดังนี้

- การเริ่มหมุนของเพลางาน ทิศทางการหมุนและการเปลี่ยนความเร็วรอบ
- การกำหนดตำแหน่งของเพลางาน

- การเปิดสารหล่อเย็น และความดันของสารหล่อเย็น
- การรักษาอัตราป้อนให้คงที่
- การเปลี่ยนตำแหน่งของเครื่องมือ
- การรักษาความเร็วตัดให้คงที่
- การเริ่มทำงานหรือควบคุมการทำงานของอุปกรณ์ช่วยงานอื่นๆ เช่น อุปกรณ์เปลี่ยนชิ้นงาน ได้แก่ โต๊ะเปลี่ยนงาน (pallet shuttle) เป็นต้น
- ชุดยันศูนย์ท้ายแท่น (Tail-stock)
- อุปกรณ์ใส่และถอดชิ้นงาน (loader and unloader)
- แท่นประคองศูนย์ (Steady rest)
- อุปกรณ์ลำเลียงเศษ (chip conveyor)
- Sorter

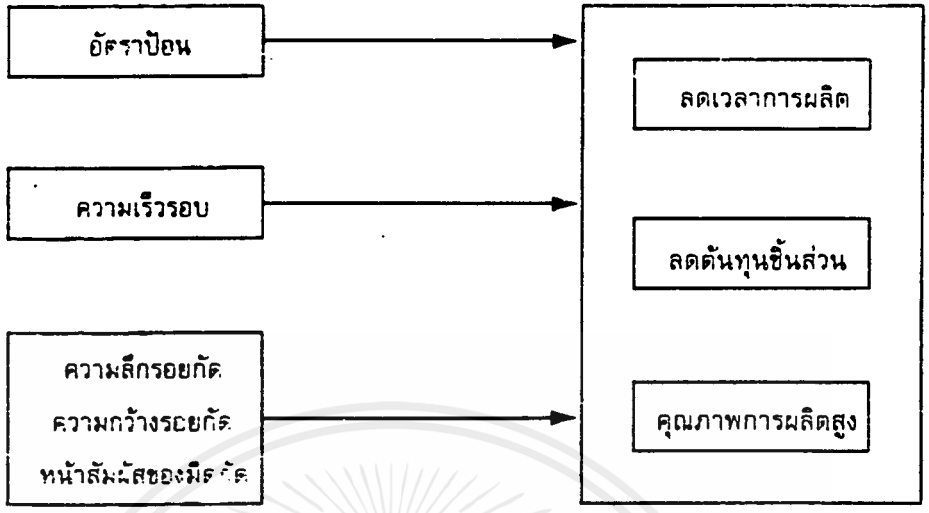


รูปที่ 2.26 หน้าที่การทำงานของเครื่องจักรกล

เครื่องจักรกลซีเอ็นซีที่สามารถใช้ระบบควบคุมสั่งการทำงานในหน้าที่ต่างๆ ได้ยิ่งมากเท่าใดก็จะเป็นระบบที่มีความเหมาะสมสำหรับการควบคุมแบบอัตโนมัติมากยิ่งขึ้น

2.5 การตัดเฉือนโลหะด้วยเครื่องจักรกลซีเอ็นซี

2.5.1 ข้อมูลการตัดเฉือนโลหะสำหรับงานกัด



รูปที่ 2.27 ความสัมพันธ์ของข้อมูลการตัดเฉือนโลหะสำหรับงานกัด

ข้อมูลที่ช่างเขียนโปรแกรมจะต้องจัดเตรียมสำหรับการทำงานกัดได้แก่ ความเร็วรอบของเพลามีดกัด อัตราป้อน ความลึกหรือความกว้างรอยกัดและหน้าสัมผัสของมีดกัด แฟคเตอร์เหล่านี้จะต้องนำมาพิจารณาร่วมกันเพื่อให้บรรลุเป้าหมาย 3 ประการ คือ

เป้าหมายที่ 1 : ลดเวลาการผลิต (Short cycle time)

แฟคเตอร์สำคัญเกี่ยวกับเวลาการผลิต (Cycle time) ที่สามารถควบคุมได้โดยช่างเขียนโปรแกรมเอ็นซี ได้แก่ ปริมาตรการตัดเฉือนเนื้อโลหะออกต่อนาที (Stock removal rate per minute) ซึ่งเป็นผลคูณของค่าอัตราป้อนกับหน้าสัมผัสของมีดกัด (Cut engagement) กับความลึกหรือความกว้างรอยกัด ค่าแฟคเตอร์ค่าหนึ่งค่าใดใน 3 ค่านี้ หากมีค่าใดสูงขึ้นก็จะเป็นผลให้ปริมาตรการตัดเฉือนเนื้อโลหะออกมีอัตราที่สูงขึ้นก็จะเป็นผลให้เครื่องมือตัดสึกหรอเร็วขึ้นด้วย ทำให้ค่าเฉลี่ยของเวลาการผลิตสูงขึ้นด้วย เนื่องจากการเปลี่ยนเครื่องมือตัด หรือเปลี่ยนคมตัด

เป้าหมายที่ 2 : ลดต้นทุนชิ้นส่วน (Lower part costs)

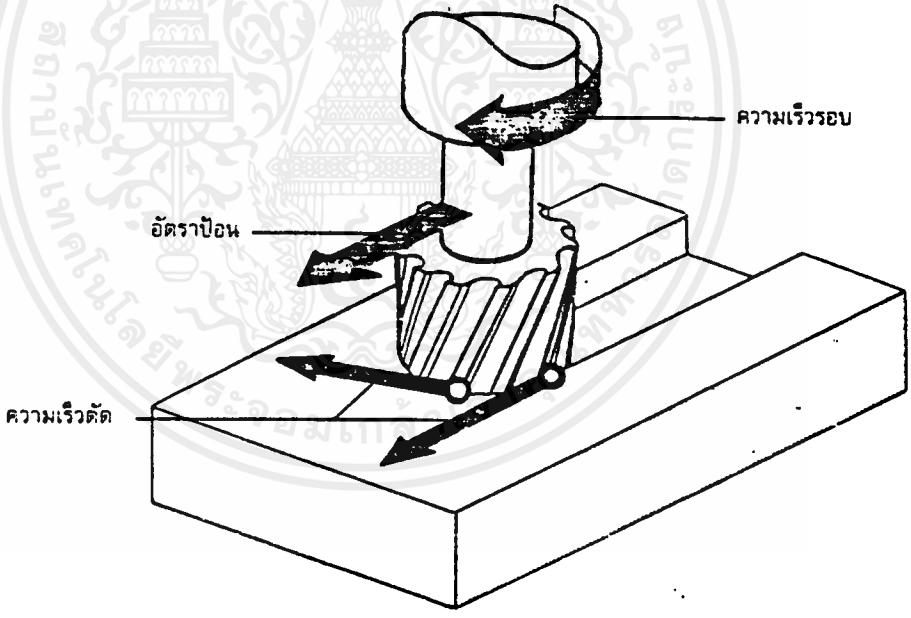
การเพิ่มขึ้นของค่าข้อมูลการตัดเฉือนใดๆ ก็ตามจะมีผลทำให้เวลาการผลิตต่อชิ้นลดลง ซึ่งจะเป็นการช่วยลดค่าแรงงานและค่าเครื่องจักรลงด้วยแต่ค่าเครื่องมือตัดจะสูงขึ้นเนื่องจากมีการสึกหรอสูง ดังนั้นจึงไม่ควรเลือกใช้ค่าข้อมูลการตัดเฉือนที่จะมีผลทำให้ค่าเครื่องมือตัดที่เกิดจากการสึกหรอสูงเกินระดับหนึ่ง ซึ่งอาจพิจารณาเลือกใช้สารหล่อเย็นเพื่อเพิ่มอายุคมมีดให้ยาวขึ้นด้วยก็ได้

เป้าหมายที่ 3 : คุณภาพการผลิตสูง (High production quality)

การเลือกใช้ข้อมูลการตัดเฉือนจะถูกจำกัดด้วยผลิตภัณฑ์ที่ต้องการคุณภาพสูง ซึ่งจะเกี่ยวข้องกับผิวสำเร็จและพิกัดความเผื่อของขนาดชิ้นงานสำเร็จ การเลือกใช้ข้อมูลการตัดเฉือนจะต้องพิจารณาให้สัมพันธ์กันกับข้อมูลอื่นๆ ได้แก่

- หมวดงานกัด (Milling mode) เช่น งานกัดตาม งานกัดทวน งานกัดปาดหน้า เป็นต้น
- รูปทรงของมีดกัด
- ชนิดของขอบคมตัดที่ใช้ เช่น รูปทรงของขอบคมตัด วัสดุมีดกัด เป็นต้น
- ภาระงานของเครื่องจักร เช่น ความสามารถในการรับความเค้น (Stressability)
- คุณสมบัติการสั่นสะเทือนของเครื่องจักร เครื่องมือตัด และวัสดุงาน

1) ความเร็วรอบและอัตราป้อน



รูปที่ 2.28 ความเร็วรอบ ความเร็วตัด และอัตราป้อน

1.1) <u>ความเร็วรอบของมีดกัด</u>

สามารถป้อนค่าในโปรแกรมเอ็นซีเป็นค่าความเร็วรอบโดยตรง หน่วยเป็น จำนวนรอบต่อนาที (r.p.m.)

ตัวอย่าง : S = 630 r.p.m. หมายถึง ค่าความเร็วรอบ 630 รอบต่อนาที

ค่าความเร็วรอบที่เลือกใช้จะเป็นตัวกำหนดค่าความเร็วตัดในงานกัดด้วย (ดังรูปที่2.28) ความเร็วตัดจะมีค่าเท่ากับความเร็วขอบของมีดกัด การเปลี่ยนแปลงของค่าความเร็วตัดนี้จะไม่ขึ้นอยู่กับค่าความเร็วรอบเพียงอย่างเดียว แต่ยังขึ้นอยู่กับขนาดเส้นผ่านศูนย์กลางของมีดกัดด้วย กล่าวคือ ถ้าความเร็วรอบมีค่าสูงและมีดกัดมีขนาดเส้นผ่านศูนย์กลางโตจะเป็นผลให้ความเร็วตัดมีค่าสูงด้วย

ในการป้อนค่าความเร็วรอบ จะต้องมั่นใจว่าได้กำหนดทิศทางการหมุนของเพลามีดกัดไว้ถูกต้องด้วย

1.2) <u>อัตราป้อน</u>

อัตราป้อนเป็นการเคลื่อนที่ของมีดกัดในทิศทางทำงานตัดเฉือน (ดูรูปที่2.29) โดยทั่วไปอัตราป้อนจะกำหนดเป็นระยะทางการเคลื่อนที่ต่อนาที นอกจากนี้อาจกำหนดอัตราป้อนเป็นระยะทางการเคลื่อนที่ต่อรอบการหมุนของมีดกัด หรือต่อฟันมีดก็ได้

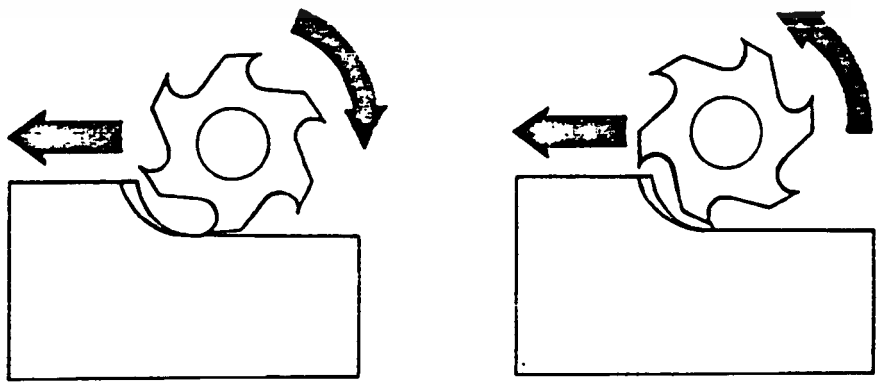
ตัวอย่าง : F = 100 mm/min หมายถึง ระยะทางการเคลื่อนที่ของมีดกัด 100 มม.ในเวลา 1 นาที

F = 0.1 mm/rev หมายถึง ระยะทางการเคลื่อนที่ของมีดกัด 0.1 มม.เมื่อมีดกัดหมุนครบ 1 รอบ

F = 0.02 mm/T หมายถึง ระยะทางการเคลื่อนที่ของมีดกัด 0.02 มม.ต่อ 1 ฟันมีดกัด

โดยทั่วไป การเคลื่อนที่ป้อนในงานกัด จะเกิดจากการเคลื่อนที่ของโต๊ะงานอย่างต่อเนื่องสม่ำเสมอ และการเคลื่อนที่หมุนของมีดกัด แต่ในการเขียนโปรแกรมเอ็นซีสำหรับงานกัดจะโปรแกรมในลักษณะที่สมมุติให้โต๊ะงานอยู่กับที่และมีดกัดเคลื่อนที่ (ตามหลักความสัมพันธ์ในการเคลื่อนที่)

เมื่อกำหนดให้ความเร็วรอบของมีดกัดมีค่าคงที่ การเลือกใช้อัตราป้อนจะมีผลต่อความหนาของเศษและผิวสำเร็จของชิ้นงาน



ก) งานกัดตาม ข) งานกัดทวน

รูปที่ 2.29 ลักษณะการเคลื่อนที่กัด

การเลือกใช้ลักษณะงานกัดระหว่างงานกัดตามกับงานกัดทวนจะมีผลกระทบต่อการเปลี่ยนรูปของเศษ(chip formation)และความดันตัดเฉือน (cutting pressure)

ก) งานกัดตาม (Conventional milling) ในงานกัดตาม ความหนาของเศษและความดันในการตัดเฉือนจะเพิ่มขึ้นเรื่อยๆ ที่ฟันมีดกัด และจะมีค่าสูงสุดก่อนที่ฟันมีดกัดจะเลื่อนพ้นวัสดุงานเล็กน้อย เมื่อฟันมีดกัดเลื่อนพ้นวัสดุงานแล้ว จะเกิดสภาวะที่ติดตามมาคือ ความดันตัดเฉือนจะหมดไปทันที ทำให้มีดกัดเคลื่อนที่ไปข้างหน้าโดยเร็ว และฟันมีดกัดถัดไปจะเลื่อนเข้ากัดวัสดุงานในลักษณะการกระตุก (jerking) เป็นผลให้เกิดเป็นรอยสั่น (chatter marks) ขึ้นที่ผิวงาน

ข) งานกัดทวน (Climb milling) ในงานกัดทวน ลักษณะการเกิดเศษจะกลับกันกับงานกัดตาม กล่าวคือ เมื่อฟันมีดกัดเริ่มเข้าตัดเฉือนชิ้นงาน ความหนาของเศษและความดันตัดเฉือนจะมีค่าสูงสุด และเมื่อฟันมีดกัดเลื่อนออกจากวัสดุงาน เศษจะมีขนาดบางที่สุดและความดันตัดเฉือนมีค่าน้อยสุด ดังนั้น จึงทำให้เกิดรอยสั่นสะเทือนน้อยและชิ้นงานมีผิวสำเร็จที่ดีกว่าเมื่อเปรียบเทียบกับงานกัดตาม งานกัดทวนจะใช้เครื่องกัดที่มีกำลังน้อยกว่าได้ แต่ต้องการความแข็งแรงของเครื่องกัดมากกว่า และมีโต๊ะงานที่ปราศจากระยะคลอน (backlash)

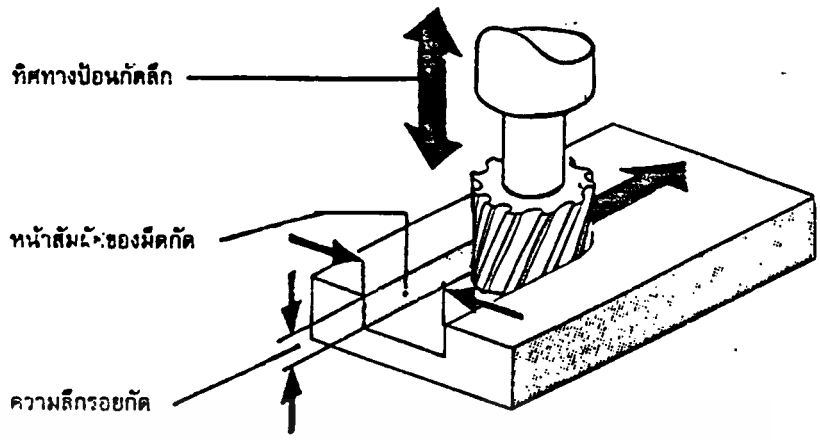
เนื่องจากเครื่องจักรกลซีเอ็นซีจะมีคุณสมบัติเหล่านี้อยู่แล้ว ดังนั้น จึงมักนิยมเลือกใช้งานกัดทวนมากกว่า

2) ความลึกหรือความกว้างรอยกัดและหน้าสัมผัสของมีดกัด

2.1) <u>ความลึกหรือความกว้างรอยกัด (Depth or width of cut)</u>

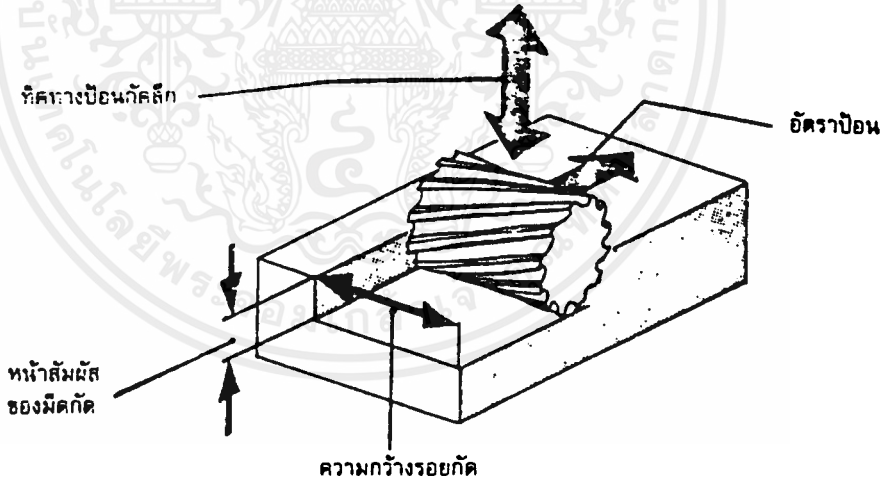
ความลึกหรือความกว้างรอยกัด หมายถึง ระยะทางที่มีดกัดจมลึกเข้าไปในผิวงานในทิศทางป้อนกัด (Infeed direction)

ก) ความลึกรอยกัด (Depth of cut) จะเรียกใช้สำหรับงานกัดด้วยเครื่องกัดเพลาตั้ง เช่น งานกัดปาดหน้า (face milling) เป็นต้น ดังรูปที่ 2.30



รูปที่ 2.30 ความลึกของรอยกัดและหน้าสัมผัสของมีดกัดในงานกัดปาดหน้า

ข) ความกว้างรอยกัด (Width of cut) จะเรียกใช้สำหรับงานกัดด้วย เครื่องกัดเพลานอน เช่น งานกัดราบ (peripheral or plain or slab milling) เป็นต้น ดังรูปที่ 2.31



รูปที่ 2.31 ความกว้างของรอยกัดและหน้าสัมผัสของมีดกัดในงานกัดราบ

2.2) <u>หน้าสัมผัสของมีดกัด (Cutter engagement)</u>

หน้าสัมผัสของมีดกัด คือ ความกว้างของมีดกัดที่สัมผัสอยู่กับชิ้นงาน โดยวัดในระนาบการทำงาน (working plane) ในทิศทางที่ตั้งฉากกับอัตราป้อน (ดูรูปที่ 2.30 และ 2.31)

ความลึกหรือความกว้างของรอยกัด และหน้าสัมผัสของมีดกัด จะเป็นผลมาจาก

ก) การโปรแกรมการเคลื่อนที่ของมีดกัด และ

ข) ขนาดและรูปทรงของมีดกัด

มีดกัดที่เลือกใช้จะต้องมีขนาดไม่ยาวเกินความจำเป็นในการทำงานกัดนั้นๆ มีดกัดยิ่งมีขนาดยาว จะทำให้เกิดการเปลี่ยนแปลงของขนาดชิ้นงานมากขึ้น ทั้งนี้เนื่องจากการโก่งงอของด้ามมีดกัด

ในการเขียนโปรแกรมเส้นทางเดินของมีดกัด (cutter path) ในชิ้นงานจะต้องพิจารณาเลือกใช้ความลึกหรือความกว้างรอยกัดและหน้าสัมผัสของมีดกัดที่สัมพันธ์กับข้อมูลอื่นๆ ด้วย ได้แก่

- ความเร็วในการตัดเฉือนของมีดกัดที่ใช้ ที่เป็นไปได้ กับวัสดุชิ้นงานที่กัด
- ขนาดผิวสำเร็จที่ต้องการ

2.6 เรขาคณิตเบื้องต้นสำหรับการทำโปรแกรมเอ็นซี

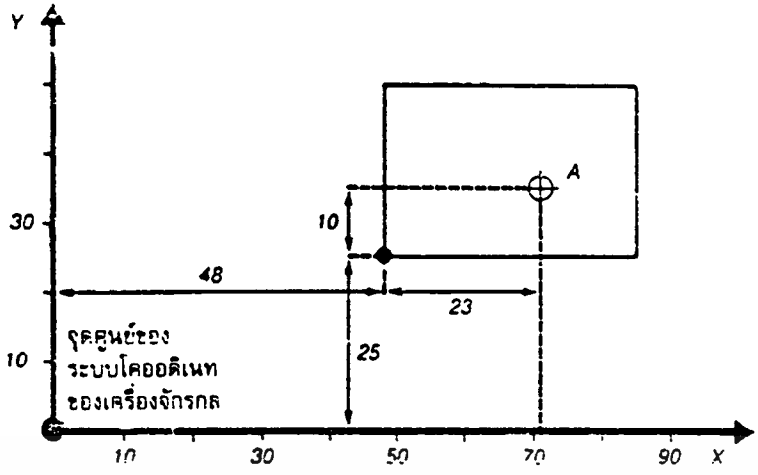
2.6.1 จุดศูนย์และจุดอ้างอิง (Zero points and Reference points)

การเคลื่อนที่ต่างๆ ของเครื่องจักรกลซีเอ็นซี จะถูกควบคุมด้วยระบบโคออดิเนท ตำแหน่งต่างๆที่ถูกต้องภายในพื้นที่ทำงานของเครื่องจักรกล จะวัดระยะมาจากจุดศูนย์ (Zero points) นอกจากจุดศูนย์แล้ว เครื่องจักรกลซีเอ็นซียังมีจุดอ้างอิง (Reference points)อื่นๆอีก เพื่อช่วยเสริมการทำงานและการทำโปรแกรม

1) จุดศูนย์ของเครื่อง (Machine zero point , M)

การจับยึดชิ้นงานเข้ากับเครื่องจักรกลซีเอ็นซีจะต้องให้สัมพันธ์กันระหว่างขนาดที่กำหนดในแบบงานกับระบบโคออดิเนทของเครื่องเพื่อให้สามารถเปรียบเทียบขนาดซึ่งกันและกันได้ เครื่องจักรกลซีเอ็นซีทุกเครื่องจะมีระบบโคออดิเนทประกอบอยู่ ระบบนี้จะกำหนดจากการเคลื่อนที่กับระบบวัดระยะการเคลื่อนที่ของเครื่องจักรกลที่มีอยู่

รูป 2.32 แสดงให้เห็นแบบชิ้นงานที่วางอยู่บนระบบโคออดิเนทของเครื่องจักรกล รู A ที่มีขนาดระยะกำหนดในแบบงาน คือ 23 และ 10 มม. จะมีค่าโคออดิเนทที่สัมพันธ์กับระบบโคออดิเนทของเครื่องจักรกล คือ โคออดิเนท X = 71 และ Y = 35 มม.



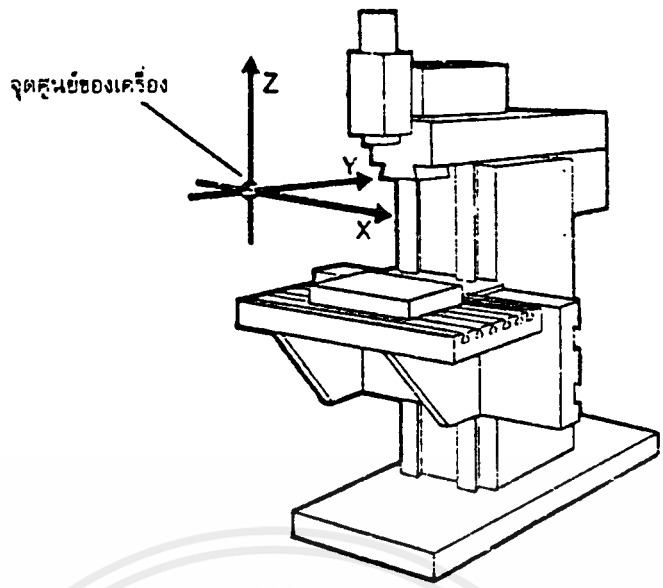
รูปที่ 2.32 แบบชิ้นงานในระบบโคออดิเนทของเครื่องจักรกล CNC

จุดเริ่มต้นของการกำหนดขนาดในแบบงาน (มุมซ้ายมือด้านล่าง) จะมีระยะเยื้องศูนย์จากระบบโคออดิเนทของเครื่องจักรกลซีเอ็นซี คือ X = 48 และ Y = 25 มม.

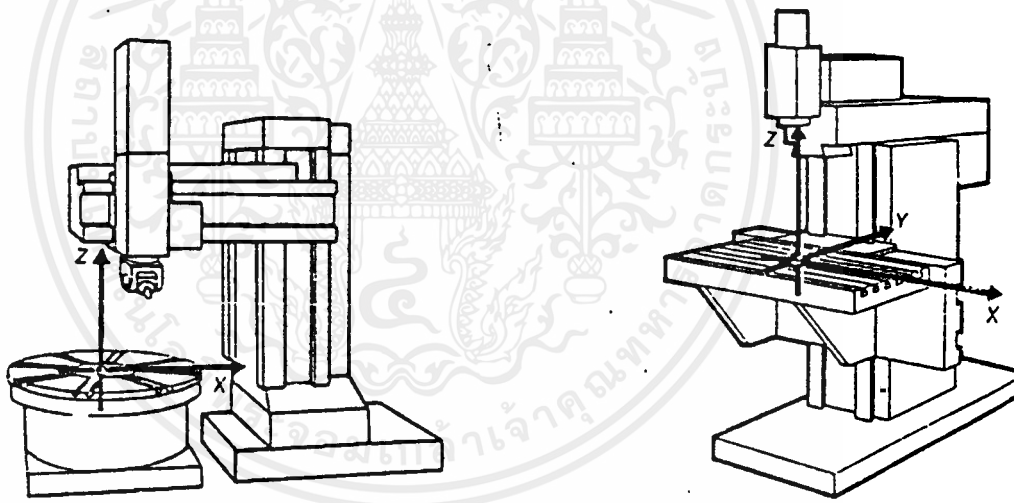
จุดศูนย์ของเครื่อง (M) โดยทั่วไปจะใช้แทนด้วยสัญลักษณ์ ตำแหน่งจุดศูนย์ของเครื่องจะถูกกำหนดโดยบริษัทผู้ผลิตเครื่องจักรกลซีเอ็นซี จุดศูนย์ของเครื่องจะใช้เป็นจุดศูนย์ของระบบโคออดิเนทของเครื่องจักรกล และใช้เป็นจุดเริ่มต้นสำหรับระบบโคออดิเนทอื่นๆ และยังใช้เป็นจุดอ้างอิงในเครื่องจักรกลด้วย

สำหรับเครื่องกัดจะมีตำแหน่งจุดศูนย์ของเครื่องที่แตกต่างกัน ซึ่งขึ้นอยู่กับบริษัทผู้ผลิตเครื่องกัด ดังนั้น ตำแหน่งจุดศูนย์ของเครื่องและทิศทางของแนวแกนที่ถูกต้อง จึงต้องศึกษาจากคู่มือการปฏิบัติงานที่จัดเตรียมโดยบริษัทผู้ผลิตแต่ละบริษัท

ตัวอย่างตำแหน่งจุดศูนย์ของเครื่องของเครื่องจักรกลซีเอ็นซีอื่นๆ ที่เป็นไปได้พร้อมด้วยระบบโคออดิเนทของเครื่อง ได้แสดงไว้ในรูปที่ 2.34 สำหรับเครื่องกัดตำแหน่งจุดศูนย์ของเครื่องอาจอยู่ที่จุดศูนย์กลางของโต๊ะงาน หรือที่จุดตรงขอบของช่วงที่มีการเคลื่อนที่ หรือแท่นเลื่อนก็ได้



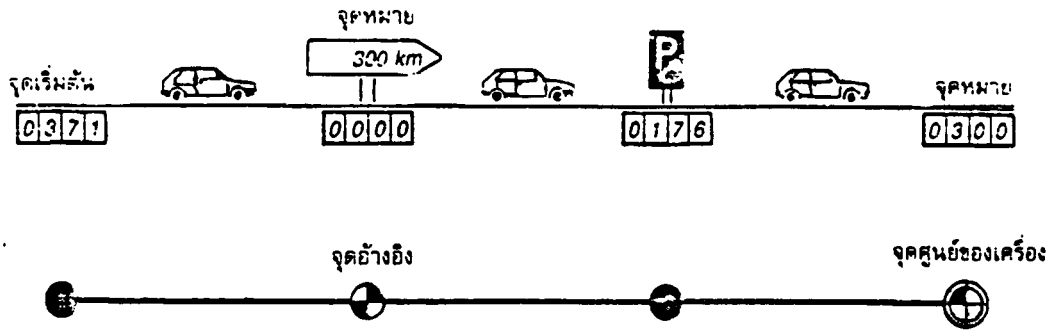
รูปที่ 2.33 ตำแหน่งจุดศูนย์ของเครื่องกัด



รูปที่ 2.34 ตัวอย่างตำแหน่งจุดศูนย์ของเครื่องจักรกล CNC อื่นๆ

2) จุดอ้างอิง (Reference point , R)

วัตถุประสงค์เบื้องต้นของการใช้จุดอ้างอิง สามารถอธิบายให้เข้าใจได้ง่ายโดยเปรียบเทียบกับการใช้หลักกิโลเมตรตามเส้นทางหลวงต่างๆ ดังแสดงในรูป 2.35



รูปที่ 2.35 วัตถุประสงค์ของการใช้จุดอ้างอิง

จากรูปที่ 2.35 สมมุติว่าขณะที่ท่านขับรถยนต์อยู่ มีเหตุการณ์ต่างๆ เกิดขึ้นตามลำดับขั้นตอนดังต่อไปนี้

2.1) ณ จุดเริ่มต้นการเดินทาง ท่านจะยังไม่ทราบว่า จุดหมายที่จะไปนั้นอยู่ห่างไกลออกไปเป็นระยะทางเท่าใด

2.2) ตลอดเส้นทาง ท่านจะคอยสังเกตดูหลักกิโลเมตร (จุดอ้างอิง) ซึ่งจะบอกว่าจุดหมายที่จะไปนั้นอยู่ห่างออกไปเท่าไร ท่านก็จะปรับตั้งมิเตอร์วัดระยะทางให้ตรงขีดศูนย์

2.3) จากนี้ไปท่านก็จะสามารถบอกได้ว่า ณ จุดต่างๆ ที่ท่านไปถึงนั้นอยู่ห่างจากจุดหมายของท่านเท่าไร หรืออยู่ห่างจากจุดอ้างอิงเท่าไร (หลักกิโลเมตร)

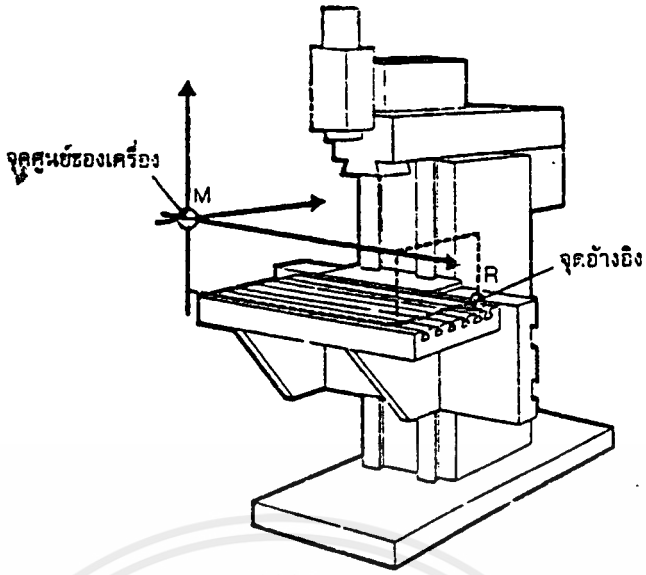
ต่างๆ ดังนี้จากรูปที่ 2.35 จะมีสถานีอยู่ 3 แห่งด้วยกัน ซึ่งเมื่อเปรียบเทียบกับเครื่องจักรกลซีเอ็นซีแล้ว จะมีความหมาย

สถานีที่ 1 หมายถึง การเปิดสวิตช์เครื่อง

สถานีที่ 2 หมายถึง การเลื่อนไปยังจุดอ้างอิง และปรับตั้งระบบวัดระยะเลื่อนให้เริ่มต้นที่ศูนย์

สถานีที่ 3 หมายถึง ตำแหน่งของเครื่องมือที่เคลื่อนที่ไปอย่างต่อเนื่อง

จุดอ้างอิง (R) มักจะใช้แทนด้วยสัญลักษณ์  เป็นจุดที่ใช้ช่วยในการปรับค่าและควบคุมระบบวัดขนาดระยะการเคลื่อนที่ของแท่นเลื่อนและเครื่องมือ ตำแหน่งของจุดอ้างอิงจะถูกกำหนดไว้ก่อนล่วงหน้าอย่างเที่ยงตรงในทุกแนวแกนของการเคลื่อนที่ด้วยสวิตช์จำกัดระยะ หรือสวิตช์ขาเตะ (Limit switches หรือ Trip dogs) ดังนั้น ค่าโคออดิเนทของจุดอ้างอิงจะมีค่าเท่าเดิมเสมอ และรู้ค่าตัวเลขที่แน่นอนที่สัมพันธ์กับจุดศูนย์ของเครื่อง

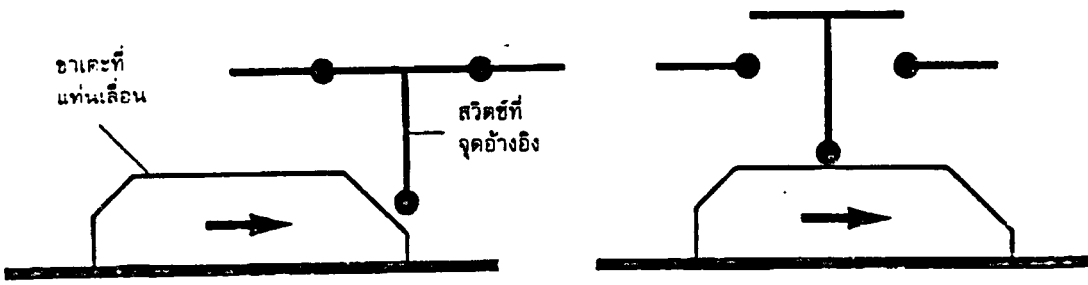


รูปที่ 2.36 ตำแหน่งจุดอ้างอิงของเครื่องกัด

ข้อสำคัญ : หลังจากเปิดสวิตช์ระบบควบคุมแล้ว แนวแกนทั้งหมดจะต้องเลื่อนไปยังจุดอ้างอิง ก่อนเสมอ เพื่อปรับค่าระบบวัดระยะการเคลื่อนที่

ถ้าเกิดเหตุขัดข้องขึ้นจนทำให้ข้อมูลของตำแหน่งแท่นเลื่อนและเครื่องมือในปัจจุบัน สูญหายไปจากระบบควบคุม ซึ่งอาจมีสาเหตุมาจากไฟฟ้าดับ เป็นต้น จะต้องเลื่อนแท่นเลื่อนต่างๆ กลับไปหาจุดอ้างอิงก่อนเริ่มทำงานใหม่เสมอ เพื่อปรับค่าของระบบวัดระยะการเคลื่อนที่ให้ถูกต้อง

เครื่องจักรกลซีเอ็นซีที่มีระบบวัดระยะการเคลื่อนที่แบบสัมบูรณ์(Absolute traverse measurement) อาจจะไม่ต้องใช้จุดอ้างอิง ทั้งนี้เพราะว่าสามารถอ่านค่าโคออดิเนทการเคลื่อนที่ของแนวแกนต่างๆ ได้โดยตรงและทุกเวลาที่ต้องการ อย่างไรก็ตาม สำหรับเครื่องจักรกลซีเอ็นซีที่ใช้ระบบวัดระยะการเคลื่อนที่แบบต่อเนื่อง (Incremental traverse measuring system) ยังต้องการใช้จุดอ้างอิงสำหรับการปรับค่าระบบวัดระยะการเคลื่อนที่

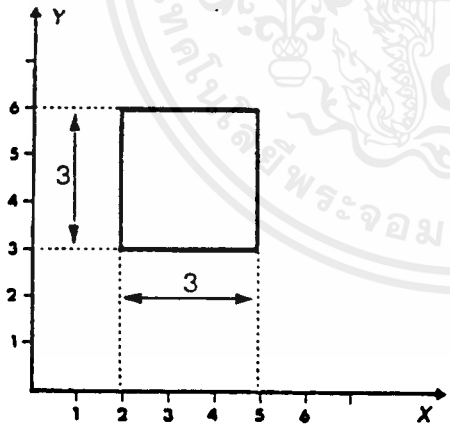


รูปที่ 2.37 การเคลื่อนที่ไปยังตำแหน่งของจุดอ้างอิง

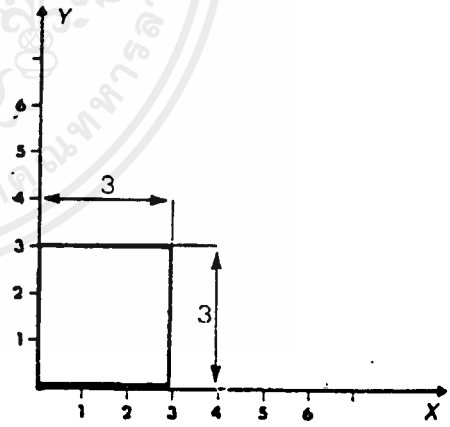
เมื่อขาเตะ (trip dog) เลื่อนสวิตช์ที่จุดอ้างอิงให้เปิดออกแล้ว ระบบวัดระยะการเคลื่อนที่ก็จะปรับค่าไปที่ศูนย์หรือค่าที่กำหนดไว้ล่วงหน้า เพื่อให้ได้ระดับความถูกต้องที่ต้องการ ในขณะที่ทำการปรับค่าของระบบวัดขนาดการเคลื่อนที่ของแท่นเลื่อนไปยังจุดอ้างอิงควรจะลดความเร็วลง และเลื่อนไปในทิศทางเดียวกันตลอด สำหรับเครื่องจักรบางแบบ เช่น เครื่องกัด ระบบวัดขนาดสามารถที่จะปรับค่าโดยการเลื่อนแท่นเลื่อนไปที่จุดศูนย์ของเครื่องก็ได้ อย่างไรก็ตาม ในกรณีทั่วไป การเคลื่อนที่ไปยังจุดศูนย์ของเครื่องจะไม่สามารถทำได้ เมื่อเครื่องมือและชิ้นงานอยู่ในตำแหน่งทำงานแล้ว ดังนั้น จึงต้องเพิ่มจุดอ้างอิงเข้าไปด้วย

3) จุดศูนย์ของชิ้นงาน (Workpiece zero point)

ในระบบโคออดิเนทสามารถที่จะเลือกข้อมูลโคออดิเนทได้ด้วยวิธีง่ายๆ โดยการกำหนดตำแหน่งที่จะวางชิ้นงานลงในระบบโคออดิเนทที่สะดวกต่อการอ่านค่าจุดโคออดิเนท รูปที่ 2.38 แสดงให้เห็นการวางแบบชิ้นงานสี่เหลี่ยมตรงจุดใดๆ ในระบบโคออดิเนท ส่วนรูปที่2.39 เป็นแบบชิ้นงานเดิมที่วางให้ขอบงานสองด้านซ้อนทับแกน X และแกน Y ซึ่งวิธีหลังนี้จะสามารถใช้ค่าขนาดที่กำหนดในแบบชิ้นงานเป็นค่าโคออดิเนทได้เลย และสามารถตรวจสอบค่าได้ง่ายกว่า และยังช่วยหลีกเลี่ยงการคำนวณหาค่าโคออดิเนทเพิ่มเติมได้



รูปที่ 2.38 การวางแบบชิ้นงานที่จุดใดๆ ในระบบโคออดิเนท

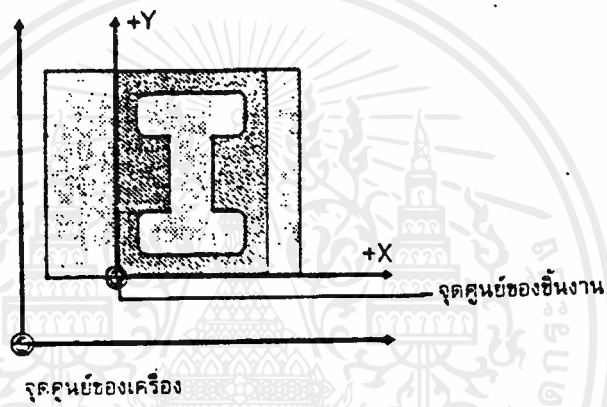


รูปที่ 2.39 การวางแบบชิ้นงานโดยใช้แนวแกนของระบบโคออดิเนทเป็นหลัก

จุดศูนย์ของชิ้นงาน (W) มักจะแสดงด้วยสัญลักษณ์  จะเป็นจุดที่ช่วยในการกำหนดระบบโคออดิเนทของชิ้นงานที่สัมพันธ์กับจุดศูนย์ของเครื่อง จุดศูนย์ของชิ้นงานจะถูกเลือกใช้โดยผู้เขียนโปรแกรม และป้อนเข้าไปในระบบซีเอ็นซีในขั้นตอน

ของการปรับตั้งตำแหน่งของจุดศูนย์ของชิ้นงานสามารถที่จะกำหนดเลือกใช้ได้อย่างอิสระ โดยผู้เขียนโปรแกรม แต่ต้องอยู่ภายในขอบเขตการทำงานของเครื่องจักรกล โดยมีหลักเกณฑ์ง่ายๆ คือ การกำหนดตำแหน่งจุดศูนย์ของชิ้นงานควรจะกำหนดไว้ในตำแหน่งที่เป็นจุดอ้างอิงต่างๆ ที่กำหนดไว้ในแบบชิ้นงานอยู่แล้ว กล่าวคือ เมื่อกำหนดตำแหน่งจุดศูนย์ของชิ้นงานแล้ว สามารถที่จะเปลี่ยนขนาดที่กำหนดในแบบชิ้นงานให้เป็นค่าโคออดิเนทได้โดยสะดวก และหลีกเลี่ยงการคำนวณค่าโคออดิเนทเพิ่มเติมได้

สำหรับชิ้นงานกัดที่มีรูปทรงของชิ้นงานไม่สมมาตร มักนิยมใช้ขอบมุมของชิ้นงานด้านใดด้านหนึ่งเป็นจุดศูนย์ของชิ้นงาน ส่วนชิ้นงานกัดที่สมมาตร มักจะใช้จุดศูนย์กลางของชิ้นงานเป็นตำแหน่งเป็นตำแหน่งจุดศูนย์ของชิ้นงาน



รูปที่ 2.40 ตัวอย่างการกำหนดจุดศูนย์ของชิ้นงานกัด

โดยทั่วไปแล้ว ตำแหน่งจุดศูนย์ของชิ้นงานมักนิยมใช้เป็นตำแหน่งเดียวกันกับจุดศูนย์ของโปรแกรม (Program zero point)

สรุป : ตำแหน่งจุดศูนย์ของชิ้นงาน มีข้อพิจารณาในการเลือกใช้ดังนี้

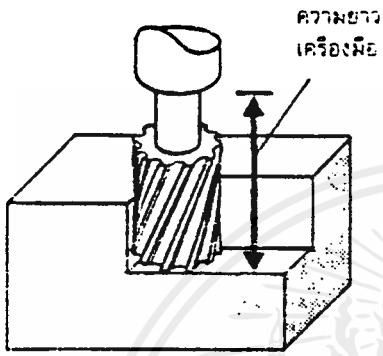
3.1) สามารถกำหนดค่าโคออดิเนทจากขนาดกำหนดในแบบงานได้มากที่สุด

3.2) สามารถจับยึดชิ้นงาน ปรับตั้ง และตรวจสอบ ตลอดจนการควบคุมด้วยระบบวัดระยะการเคลื่อนที่ได้สะดวก

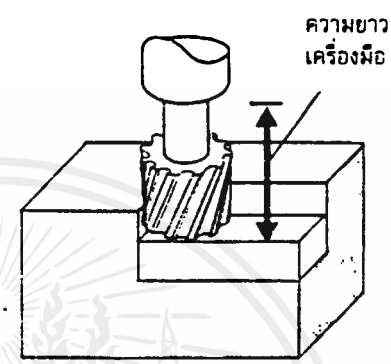
4) จุดอ้างอิงของเครื่องมือ (Tool reference points)

ในการเขียนโปรแกรมสำหรับการตัดเฉือนชิ้นงานตามเส้นขอบรูป สิ่งสำคัญที่จะต้องคำนึงถึง คือ การเคลื่อนที่ของขอบคมตัดของเครื่องมือที่ยึดอยู่กับชุดพาเครื่องมือ (Tool carrier) เช่น เพลางานของเครื่องกัด เป็นต้น จะต้องเคลื่อนที่ในลักษณะที่ทำให้ขอบคมตัดของมีดกัดเคลื่อนที่ตามเส้นขอบรูปของชิ้นงานอย่างถูกต้อง

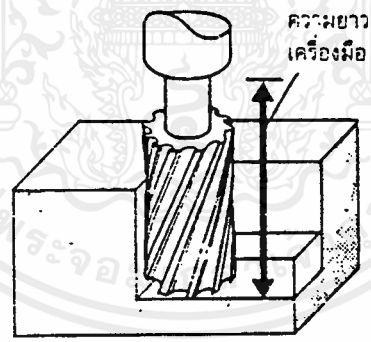
การเขียนโปรแกรมขนาดความยาวของเครื่องมือ จะต้องตรงกับขนาดความยาวของเครื่องมือที่เป็นจริง (รูปที่ 2.41) ถ้าระบบควบคุมเริ่มทำงานด้วยขนาดความยาวของเครื่องมือที่ไม่ถูกต้องก็จะทำให้ไม่ได้ขนาดของเส้นขอบรูปที่ต้องการ ถ้าขนาดของเครื่องมือสั้นเกินไป (รูปที่ 2.42) ก็จะตัดเฉือนเนื้อวัสดุออกไม่หมด และถ้าเครื่องมือยาวเกินไป (รูปที่ 2.43) ก็จะทำให้เกิดการตัดเฉือนเนื้อวัสดุชิ้นงานออกมากเกินไป



รูปที่ 2.41 ความยาวของเครื่องมือที่ถูกต้อง



รูปที่ 2.42 เครื่องมือสั้นเกินไป

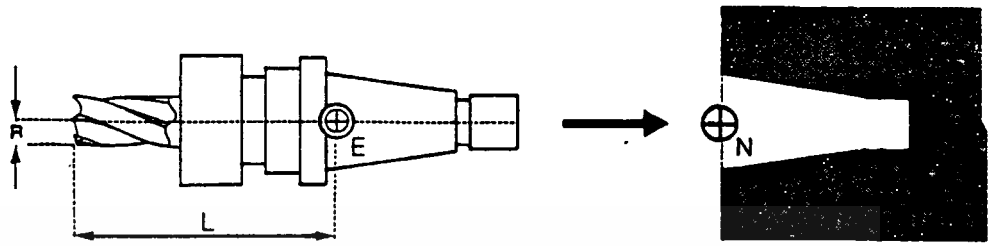


รูปที่ 2.43 เครื่องมือยาวเกินไป

ดังนั้น ขนาดความยาวของเครื่องมือ จึงต้องทำการวัดขนาดก่อนที่เครื่องจักรกลจะเริ่มทำงานตามโปรแกรม และข้อมูลที่วัดได้จะถูกป้อนเข้าไปไว้ในระบบความจำส่วนที่ทำหน้าที่เก็บข้อมูลของเครื่องมือ (Tool data storage) ของระบบควบคุมซีเอ็นซี

ในการตัดเฉือนชิ้นงาน จุดสำคัญ คือ จะต้องสามารถควบคุมจุดปลายเครื่องมือหรือขอบคมตัดที่สัมพันธ์กับขนาดของเครื่องมือตลอดเส้นทางการเคลื่อนที่ตัดเฉือน (Machining path) เนื่องจากเครื่องมือมีรูปทรงและขนาดที่แตกต่างกัน ดังนั้น ขนาดของ

เครื่องมือที่ถูกต้องจะต้องหาให้ได้ก่อนและป้อนเข้าไปในระบบควบคุม ซึ่งขนาดของเครื่องมือจะต้องปรับตั้งให้สัมพันธ์กับจุดปรับตั้งเครื่องมือ (Tool setting point) ที่อยู่คงที่



รูปที่ 2.44 จุดปรับแต่งเครื่องมือของเพลามิดกัด

4.1) จุดปรับตั้งเครื่องมือ(Tool setting point , E)จะแสดงด้วยสัญลักษณ์ ⊕ ซึ่งจะเป็นจุดที่มีตำแหน่งที่แน่นอนบนด้ามจับยึดเครื่องมือ ดังแสดงในรูป 2.44 จุดปรับตั้งเครื่องมือนี้จะอยู่ในตำแหน่งที่ทำให้สามารถวัดขนาดความยาวของเครื่องมือว่าห่างจากปลายเพลายึดเครื่องมือนั้นเท่าไร โดยทั่วไปค่าขนาดของเครื่องมือที่จะต้องป้อนเข้าไปในระบบควบคุมจะประกอบด้วย

- ความยาวของเครื่องมือ จะใช้ค่าโคออดิเนท Z หรือ L
- ระยะเยื้องศูนย์ของจุดปลายเครื่องมือหรือรัศมีของเครื่องมือ ซึ่งจะใช้ค่าโคออดิเนท X,R หรือ Q

ตำแหน่งที่จุดปรับตั้งเครื่องมือสัมผัสกับเพลายึดเครื่องมือ คือ จุดปลายรูสวมยึดเครื่องมือ (Tool socket point , N) ซึ่งมักจะแสดงด้วยสัญลักษณ์ ⊕ เมื่อสวมเครื่องมือหรือด้ามยึดเครื่องมือเข้าไปในรูของเพลายึดเครื่องมือของชุดพาเครื่องมือ เช่น ชุดเทอเรท เป็นต้น จุดปรับตั้งเครื่องมือจะซ้อนทับกับจุดปลายรูสวมยึดเครื่องมือพอดี

สรุป : จุดอ้างอิงของเครื่องมือมีความสำคัญสำหรับการปรับตั้งเครื่องมือมาก ข้อมูลเกี่ยวกับเครื่องมือจะต้องป้อนเข้าไปและเก็บไว้ในหน่วยความจำที่ทำหน้าที่เก็บข้อมูลของเครื่องมือก่อนที่จะเริ่มทำงานตัดเฉือนตามโปรแกรมที่เตรียมไว้

ระบบควบคุมจะใช้จุดปลายรูสวมยึดเครื่องมือร่วมกับขนาดของเครื่องมือที่ป้อนข้อมูลไว้ ทำให้สามารถกำหนดตำแหน่งของจุดปลายเครื่องมือที่สัมพันธ์กับจุดศูนย์ของเครื่องได้อย่างถูกต้อง สำหรับเครื่องจักรกลที่มีชุดพาเครื่องมือที่ซับซ้อน เช่น ชุดป้อมมีดเทอเรทของเครื่องกลึง แม็กกาซีนเครื่องมือของแมชชีนนิ่ง เซนเตอร์ เป็นต้น นอกจากจะต้องคำนวณจุดอ้างอิงของเครื่องมือที่สัมพันธ์กับจุดปลายรูสวมยึดเครื่องมือแล้ว ยังต้อง

พิจารณาให้สัมพันธ์กับจุดอ้างอิงการเคลื่อนที่ของชิ้นส่วนต่างๆ ของเครื่องจักรกลด้วย เช่น จุดอ้างอิงของชุดพาเครื่องมือ จุดอ้างอิงของแท่นเลื่อนต่างๆ เป็นต้น

4.2) จุดอ้างอิงของชุดพาเครื่องมือ (Tool carrier reference point , T)

มักจะใช้แทนด้วยสัญลักษณ์ ⊕ ซึ่งเป็นจุดที่อยู่คงที่บนชุดพาเครื่องมือ เช่น ชุดเทอเรท ชุดแม็กกาซีน เป็นต้น และโดยทั่วไปตำแหน่งของจุดอ้างอิงชุดพาเครื่องมือจะอยู่ที่ตำแหน่งจุดศูนย์กลางของการหมุน หรือจุดศูนย์กลางของการเคลื่อนที่ของชุดพาเครื่องมือ

4.3) จุดอ้างอิงของแท่นเลื่อน (Slide reference point , F)

จะใช้แทนด้วยสัญลักษณ์ ⊕ เป็นจุดที่อยู่ในพื้นที่ทำงานของเครื่องจักรกลที่ถูกกำหนดตำแหน่งไว้อย่างถูกต้อง

2.7 ทฤษฎีทางคณิตศาสตร์ สำหรับ CNC (Mathematic theory for CNC)

2.7.1 ทฤษฎีทางการคำนวณทิศทางการเคลื่อนที่ (Feed rate speed)[2]

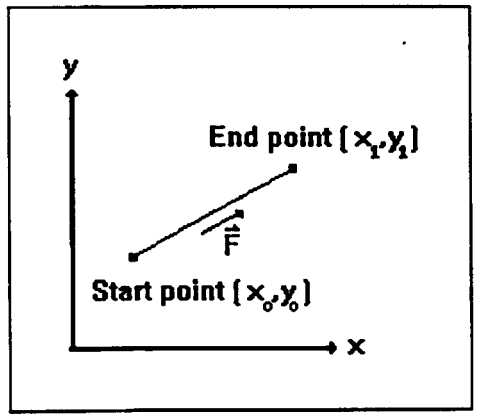
อัตราการป้อนของคำสั่ง G01,G02,G03 จะถูกกำหนดด้วยตัวเลขหลังคำสั่ง F____ และจะคงค่าเดิมตลอดจนกว่าจะมีการเปลี่ยนแปลงค่าใหม่ค่าอัตราการป้อนนี้เรียกว่า Tangential speed constant

1) Linear Interpolation

เมื่อมี Straight cut linear interpolation (G01) ส่วนควบคุมจะควบคุมแกนของเครื่องจักร 2 แกนหรือมากกว่านั้นพร้อมๆกัน ในส่วนการตัดเชิงมุมส่วนควบคุมจะใช้ข้อมูลที่ได้รับการโปรแกรมเข้ามาเก็บไว้แล้วเพื่อคำนวณ องศาหรือความชันของการตัดชิ้นส่วนของเส้นตรง ความยาวที่เปลี่ยนแปลงจากจุดเริ่มต้นจนถึงจุดสุดท้ายจะเป็นตัวกำหนด การแบ่งเส้นและความชันของแต่ละแกน เพื่อทำการควบคุมการเคลื่อนที่ของ Cutter ให้เคลื่อนที่เสมือนกับ การเคลื่อนที่ไปในทิศทางเดียว ในโครงงานนี้กำหนดให้ Linear interpolation เคลื่อนที่ในระนาบเดียวเท่านั้น

การใช้งานอย่างอื่นของ Linear interpolation คือใช้ในการประมาณเส้นโค้ง หรือวงกลม (Circular interpolation)ซึ่งจะกล่าวถึงวิธีการประมาณเส้นโค้งในช่วงต่อไป

การคำนวณใน Linear interpolation



รูป 2.45 การเคลื่อนที่แบบ Linear Interpolation

F - อัตราป้อนในแนวการเคลื่อนที่

F_x - อัตราป้อนในแนวแกน X

F_y - อัตราป้อนในแนวแกน Y

$$F_x = \frac{F(\text{ระยะทางที่เคลื่อนที่ในแนวแกนx})}{(\text{ระยะทางที่เคลื่อนที่ทั้งหมด})} = \frac{F*(x_1-x_0)}{\sqrt{(x_1-x_0)^2+(y_1-y_0)^2}}$$

$$F_y = \frac{F(\text{ระยะทางที่เคลื่อนที่ในแนวแกนy})}{(\text{ระยะทางที่เคลื่อนที่ทั้งหมด})} = \frac{F*(y_1-y_0)}{\sqrt{(x_1-x_0)^2+(y_1-y_0)^2}}$$

2) Circular Interpolation

Circular interpolation เป็นความสามารถในการตัดส่วนโค้งของวงกลม,จำนวนที่ตัดแปรผันตามขนาดของส่วนโค้งซึ่งขึ้นอยู่กับ รัศมีของส่วนโค้ง ส่วนควบคุมจะคำนวณทิศทางของ cutter จากข้อมูลที่โปรแกรมมา

ข้อมูลที่ต้องการได้แก่

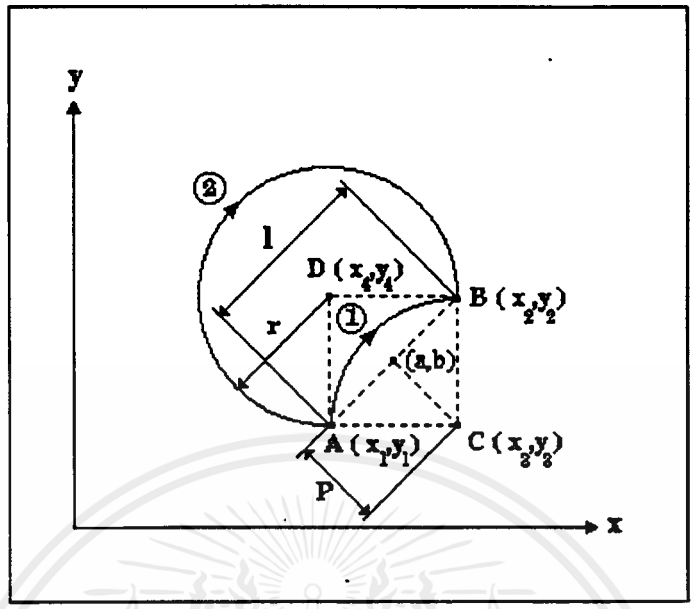
1. ทิศทางของการเคลื่อนที่ (CW หรือ CCW)
2. จุดเริ่มต้นของส่วนโค้ง (arc)
3. จุดสิ้นสุดของส่วนโค้ง (arc)
4. จุดศูนย์กลางของส่วนโค้ง (arc)

ข้อมูลที่ทราบจากการโปรแกรมแบบ Radius method (เป็นการโปรแกรมแบบที่ใช้ในโครงงานนี้)ได้แก่

1. ทิศทาง
2. จุดเริ่มต้น
3. จุดสิ้นสุด
4. รัศมี

ต้องนำมาคำนวณหา จุดศูนย์กลางของส่วนโค้ง

วิธีการคำนวณเพื่อหาจุดศูนย์กลาง



รูป 2.46 การเคลื่อนที่แบบ Circular Interpolation

$$a = (x1+x2)/2$$

$$b = (y1+y2)/2$$

$$l = \sqrt{(x2-x1)2+(y2-y1)2}$$

$$n = -(x2 - x1)/(y2 - y1)$$

$$p = \sqrt{R2 - (l/2)2}$$

จะได้

$$x3 = a + \sqrt{p2/(1+n2)}$$

$$y3 = b + (n*\sqrt{p2/(1+n2)}$$

และ

$$x4 = a - \sqrt{p2/(1+n2)}$$

$$y4 = b - (n*\sqrt{p2/(1+n2)}$$

ในการเลือกว่าจะใช้จุด (x3,y3) หรือ (x4,y4) เป็นจุดศูนย์กลางของวงกลมนั้นจะต้องพิจารณาค่ารัศมีของวงกลมว่าเป็น + หรือ -

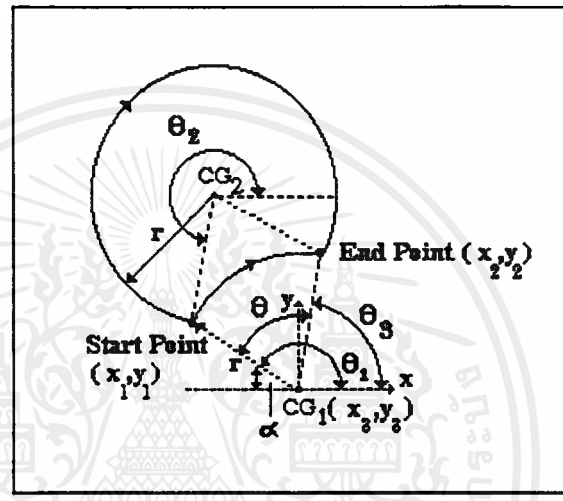
- ถ้าเป็น + จะใช้จุด (x3,y3) เป็นจุดศูนย์กลาง
- ถ้าเป็น - จะใช้จุด (x4,y4) เป็นจุดศูนย์กลาง

โดยจะใช้ผลของการ cross vector ระหว่าง vector 2 คู่ คือ AB*AC กับ AB*AD เป็นตัวชี้ คือ

- ถ้ารัศมีของวงกลมเป็น + ให้ใช้จุดศูนย์กลางที่ให้ผลของการ cross vector มีค่าสัมประสิทธิ์ของ k เป็น -
- ถ้ารัศมีของวงกลมเป็น - ให้ใช้จุดศูนย์กลางที่ให้ผลของการ cross vector มีค่าสัมประสิทธิ์ของ k เป็น +

ก็จะได้จุดศูนย์กลางของวงกลมตามที่เราต้องการ หลังจากที่เราได้ข้อมูลที่ต้องการนำมาทำการประมาณจุดบนโค้งด้วยการ ใช้ Linear interpolation

การหามุมเริ่ม



รูปที่ 2.47 การหามุมเริ่ม มุมสุดท้าย และมุมเคลื่อนที่

จากรูป 2.47 มุมเริ่ม คือมุม θ_1 และ θ_2 ของจุดศูนย์กลาง CG_1 และ CG_2 ตามลำดับ โดยการหามุมเริ่มนี้จะมีวิธีการหาเหมือนกัน ในที่นี้จะยกตัวอย่างการหามุมเริ่ม θ_1 ของจุดศูนย์กลาง CG_1 ดังมีขั้นตอนดังต่อไปนี้

1. ตั้งแกน X , Y ที่จุด CG_1 โดยให้จุด Origin อยู่ที่จุด CG_1
2. มุมเริ่มจะวัดจากแกน X ในทิศทวนเข็มนาฬิกา
3. หาค่ามุม α จาก $\alpha = \tan^{-1} (y1 - y3)/(x1 - x3)$
4. ตรวจสอบว่าจุด Start point อยู่ในควอดแดรนท์ใดของแกน X,Y ที่สร้างขึ้น โดยการดูเครื่องหมายของ (x1 - x3) และ (y1 - y3)
5. ถ้าจุด start point อยู่ควอดแดรนท์ที่ 1 ; $\theta_1 = \alpha$

   ถ้าจุด start point อยู่ควอดแดรนท์ที่ 2 ; $\theta_1 = \pi - \alpha$

   ถ้าจุด start point อยู่ควอดแดรนท์ที่ 3 ; $\theta_1 = \pi + \alpha$

   ถ้าจุด start point อยู่ควอดแดรนท์ที่ 4 ; $\theta_1 = 2\pi - \alpha$

   ก็จะได้ค่าของมุมเริ่มตามต้องการ

การหามุมสุดท้าย

จากรูปที่ 2.47 มุมสุดท้าย คือมุม θ_3 ซึ่งวิธีการหาก็จะคล้ายกับการหามุม θ_1 เพียงแต่เปลี่ยนค่ามุม α เป็น $\alpha = \tan^{-1} (y2 - y3)/(x2 - x3)$ และใช้จุด end point แทน

การหามุมเคลื่อนที่

จากรูปที่ 2.47 มุมเคลื่อนที่คือมุม θ ซึ่งมีค่าเท่ากับ

$$\theta = \theta_1 - \theta_3$$

Linear approximations of A Circle

รูปที่ 2.48 Linear Approximation of A Circle

จำนวนช่องที่แบ่ง จะใช้ความยาวของส่วนโค้งและ cord เป็นตัวกำหนดโดยที่ chord จะให้ยาว = 0.01 mm

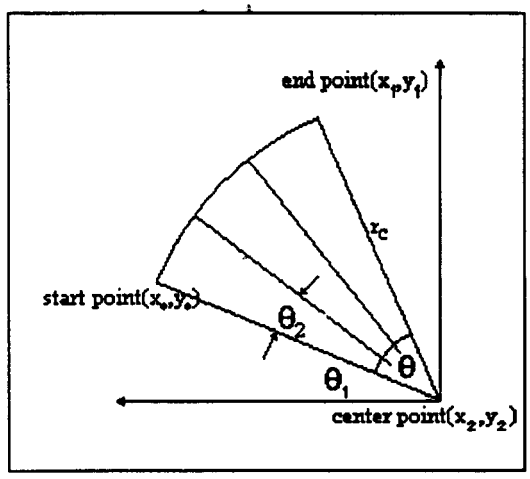
จำนวนช่องที่แบ่ง(n) = ความยาวของส่วนโค้ง / ความยาวของchord.

$$n = \theta R/0.01$$

note n จะต้องเป็นจำนวนเต็มที่ปัดขึ้นเสมอ

∴ องศาที่เปลี่ยนไปในการเคลื่อนที่แต่ละchord

$$\theta_2 = \theta/n$$



รูปที่ 2.49 Use of chord segments to approximate arc

และ $x_{1,0} = x_0$ $y_{1,0} = y_0$

n = 1 $x_{1,1} = x_2 + \cos(1\theta_2+\theta_1)$ $y_{1,1} = y_2 + \sin(\theta_2+\theta_1)$

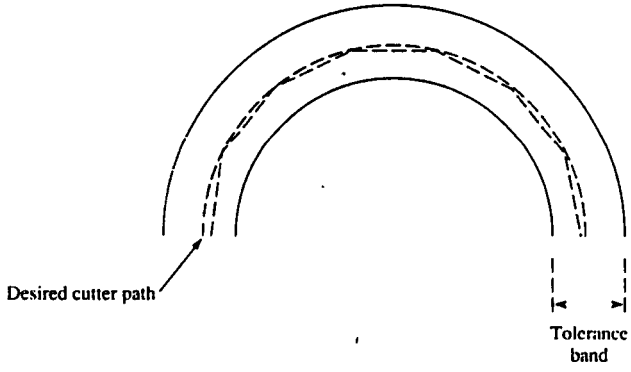
n = 2 $x_{1,2} = x_2 + \cos(2\theta_2+\theta_1)$ $y_{1,2} = y_2 + \sin(2\theta_2+\theta_1)$

n = 3 $x_{1,3} = x_2 + \cos(3\theta_2+\theta_1)$ $y_{1,3} = y_2 + \sin(3\theta_2+\theta_1)$

$x_{1,n} = x_1$ $y_{1,n} = y_1$

$$x_{1,n} = x_3 + \cos(n\theta_2+\theta_1) \qquad n = 1.....n$$

$$y_{1,n} = y_3 + \sin(n\theta_2+\theta_1) \qquad n = 1.....n$$



รูปที่ 2.50 Circular Interpolation Using Chord Segments Method

วิธีนี้เรียกว่า **Chord Segments Method**

2.7.2 การคำนวณโปรแกรมที่มีคำสั่งชดเชยขนาดของ Cutter ตามเส้นรอบรูป(Offset Calculation)

การเขียนโปรแกรมงานวัดตามเส้นรอบรูปในแนวเส้นตรงและโค้งจะมีการทำงานพร้อมกัน 2 แนวแกน ดังนั้นในการกำหนดจุดโคออดิเนทต่างๆจะต้องคำนวณเผื่อขนาดดอกกัดด้วยทำให้โปรแกรมมีความยุ่งยาก

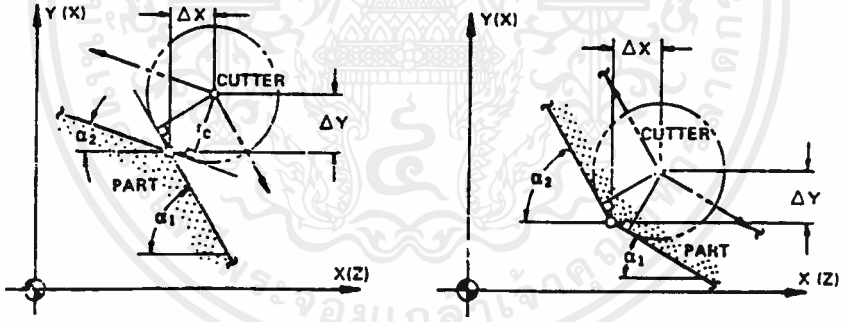
ระบบควบคุมในการทำงานของเครื่องจักร CNC จะสามารถคำนวณเผื่อขนาดของดอกกัดได้โดยอัตโนมัติ ซึ่งเรียกว่า การชดเชยขนาดดอกกัดตามเส้นรอบรูปในการโปรแกรมงานกัด จะใช้คำสั่ง

"G40" = การยกเลิกการชดเชยรัศมี

"G41" = การชดเชยรัศมีทางด้านซ้าย

"G42" = การชดเชยรัศมีทางด้านขวา

และสามารถเขียนโปรแกรมจากขนาดกำหนดในแบบงานได้โดยตรงและเปลี่ยนขนาดของดอกกัดแทนทำให้สามารถใช้ดอกกัดได้หลายขนาดโดยใช้โปรแกรมเดิม



รูปที่ 2.51 Cutter Centerline Intersection Point of Two Lines

$$\Delta x = r_c * \frac{\sin\frac{(\alpha_2+\alpha_1)}{2}}{\cos\frac{(\alpha 2-\alpha 1)}{2}} \qquad \Delta y = r_c * \frac{\cos\frac{(\alpha_2+\alpha_1)}{2}}{\cos\frac{(\alpha 2-\alpha 1)}{2}}$$

2.8 กลุ่มคำสั่งและความหมายของชุดคำสั่ง[3]

2.8.1.ชุดคำสั่ง G (G Function หรือ Preparatory Function)

ตัวเลขที่ตามชุดคำสั่ง G จะเป็นตัวกำหนดความหมายและการทำงานของเครื่อง CNC ซึ่งแบ่งเป็น 2 ประเภท ได้แก่

ก) One-shot G code G code ชนิดนี้จะมีผลให้ CNC ทำงานเฉพาะบรรทัดที่กำหนด เท่านั้น ได้แก่ G code ในกลุ่ม 00

ข) Modal G code G code ชนิดนี้จะมีผลต่อเนื่องไปยังบรรทัดต่อๆ ไป จนกว่าจะมี G code ในกลุ่มเดียวกันมากำหนด G code ใหม่

ชุดคำสั่ง G ที่ได้พัฒนาขึ้นมาสำหรับโครงงานนี้ (รูปแบบมาตรฐานจากเครื่อง Fanuc) ได้แก่

G code	กลุ่ม	ความหมาย
G00	01	เข้าสู่ตำแหน่งที่กำหนดอย่างรวดเร็ว
G01		ตัดตามระยะทางที่เป็นเส้นตรง
G02		ตัดตามแนวเส้นโค้งตามเข็มนาฬิกา CW
G03		ตัดตามแนวโค้งทวนเข็มนาฬิกา CCW
G04	00	หยุดชั่วขณะหนึ่งแล้วถอยกลับ
G20*	06	ป้อนหน่วยเป็นนิ้ว
G21		ป้อนหน่วยเป็นมิลลิเมตร
G40*	07	ยกเลิกการชดเชยรัศมีดอกกัด
G41		ชดเชยรัศมีดอกกัดจากซ้ายไปขวา
G42		ชดเชยรัศมีดอกกัดจากขวาไปซ้าย
G80*	09	ยกเลิกจังหวะที่กำหนดตายตัว
G81		เจาะเป็นจังหวะ , คว้านรูเป็นจุดๆ
G82		เจาะเป็นจังหวะ , การคว้ายผายปากรู
G83		เจาะกดเป็นจังหวะ
G85		คว้านรูเป็นจังหวะ
G86		คว้านรูเป็นจังหวะ
G89		คว้านรูเป็นจังหวะ (เจาะคายเศษ)
G90*	03	โปรแกรมคิดระยะจุดแรกไปถึงจุดสุดท้าย
G91		โปรแกรมคิดระยะจุดหนึ่งไปถึงอีกจุดหนึ่ง

G98*		ถอยกลับจังหวะสูงสุด
G99	04	กลับถึงจุดอ้างอิง (จุด R)

หมายเหตุ *1. ที่มี '*' จะเป็น G code ที่กำหนดหลังจากเปิดเครื่อง สามารถใช้ได้ทันทีโดยไม่ต้องอ้างถึง G code ก่อน*

2. G00 แม้จะเป็นคำสั่งในกลุ่ม 01 แต่ก็เป็น One-shot G code

รายละเอียดเกี่ยวกับ G code

G cede สามารถแบ่งเป็นกลุ่มต่างๆ ซึ่งแต่ละกลุ่มมีความหมายต่างๆ กันดังนี้

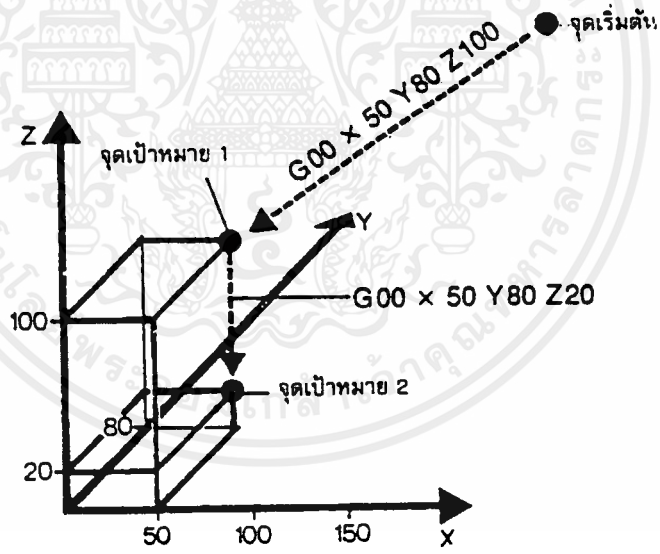
1) กลุ่ม 01 (Interpolation Functions)

แบ่งเป็น

1.1) เข้าสู่ตำแหน่งที่กำหนดอย่างรวดเร็ว (Positioning , G00) การเคลื่อนที่ไปจุดที่กำหนดด้วยความเร็วสูงสุด (2.5 m/min)

รูปแบบ G00 X__ Y__ *

เช่น G00 X10.0 Y20.0 *

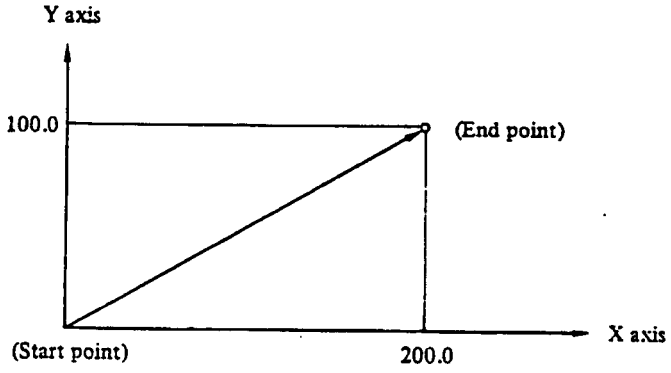


รูปที่ 2.52 การเคลื่อนที่ด้วยคำสั่ง G00

1.2) ตัดตามระยะทางที่เป็นเส้นตรง (Linear-Interpolation , G01) การเคลื่อนที่เป็นเส้นตรงไปยังจุดที่กำหนดด้วยความเร็วตาม Feed Rate (Feed Function) ที่กำหนด ซึ่งตำแหน่งที่กำหนดขึ้นอยู่กับการกำหนดว่าเป็น G91 หรือ G90 (Incremental หรือ Absolute)

รูปแบบ G01 X__ Y__ * หรือ G01 X__ Z__ *

เช่น (G91) G01 X200.0 Y100.0 F200.0 *

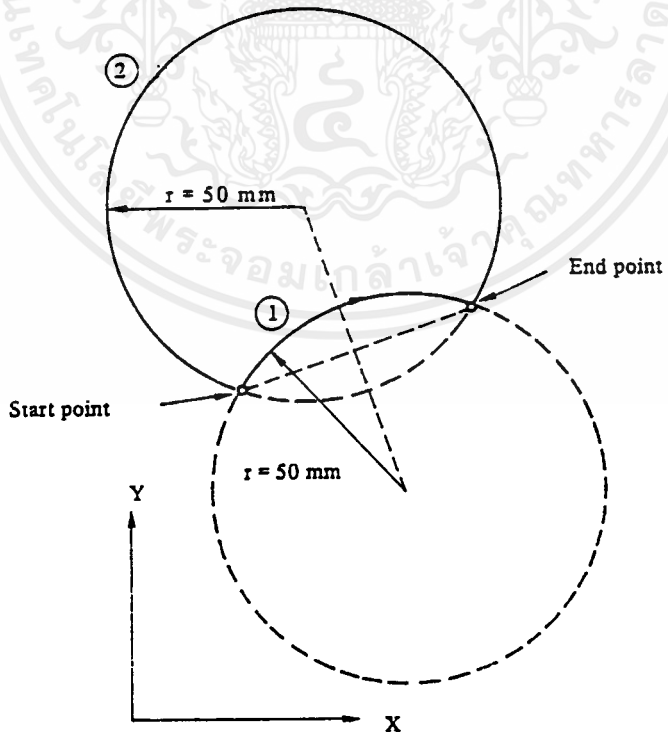


รูปที่ 2.53 การเคลื่อนที่ด้วยคำสั่ง G01

1.3) <u>ตัดตามแนวโค้งตามเข็มนาฬิกา และ ทวนเข็มนาฬิกา (Circular Interpolation CW and CCW , G02 & G03)</u> การเคลื่อนที่เป็นเส้นโค้งตามเข็มนาฬิกา และทวนเข็มนาฬิกา ไปตามรัศมี R (ตามรัศมีที่กำหนด) ไปยังจุดที่กำหนด

รูปแบบ G02 X__ Y__ R__ F__ *

เช่น (G91) G02 X60.0 Y20.0 R50.0 F300.0 *



รูปที่ 2.54 การเคลื่อนที่ด้วยคำสั่ง G02

สำหรับ arc (1) (น้อยกว่า 180°)

G91 G02 X60.0 Y20.0 R50.0 F300.0 *

สำหรับ arc (2) (มากกว่า 180°)

G91 G02 X60.0 Y20.0 R50.0 F300.0 *

G02 จะเป็นการเคลื่อนที่ตามเข็มนาฬิกา (CW)

ส่วน G03 จะเป็นการเคลื่อนที่ทวนเข็มนาฬิกา (CCW)

หมายเหตุ ในคำสั่ง G02 และ G03 จะมีคำสั่ง R เป็นตัวกำหนดขนาดของรัศมีด้วย

2) กลุ่ม 00 ได้แก่

2.1) หยุดชั่วขณะหนึ่งแล้วถอยกลับ (Dwell or Exact Stop , G04)

ทำให้การเคลื่อนที่ของ Cutter หยุดชั่วขณะตามเวลา P (หน่วยเป็น sec) ก่อนที่จะทำในบรรทัดถัดไป

รูปแบบ G04 P__ * (0.001 sec - 999.999 sec)

เช่น G04 P20.0 *

3) กลุ่ม 06 ได้แก่

3.1) ป้อนหน่วยเป็นนิ้ว (Input in inch , G20)

3.2) ป้อนหน่วยเป็นมิลลิเมตร (Input in mm. , G21)

Unit systems	G code	Least input icrement
Inch	G20	0.0001 inch
Millimeter	G21	0.001 mm

หมายเหตุ โดยธรรมดาถ้าไม่ใส่ G20 / G21 จะถือว่าเป็น G21 ก่อน

4) กลุ่ม 07 ได้แก่

4.1) ยกเลิกการชดเชยรัศมีดอกกัด (Cutter Compensation cancel , G40)

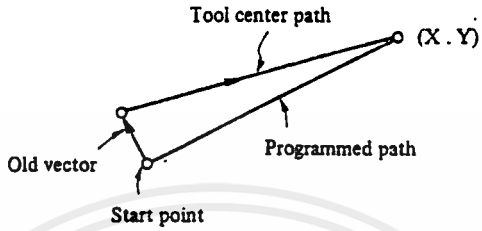
4.2) ชดเชยรัศมีดอกกัดด้านซ้าย (Cutter Compensation left , G41)

4.3) ชดเชยรัศมีดอกกัดด้านขวา (Cutter Compensation right , G42)

หมายเหตุ เมื่อเปิดเครื่องใหม่ จะถือว่าไม่มีการชดเชยรัศมี (G40) อยู่แล้ว โดยไม่ต้องกำหนด G40

4.1) <u>ยกเลิกการชดเชยรัศมีดอกกัด (Cutter Compensation , G40)</u> ใช้ร่วมกับ G00 หรือ G01 จะกำหนดในแต่ละแกน เคลื่อนที่ตรงจากจุดของ old vector บน start point ไปยัง end point โดยตรง ตามที่กำหนด (ดูรูปประกอบ)

รูปแบบ G40 X__ Y__ *



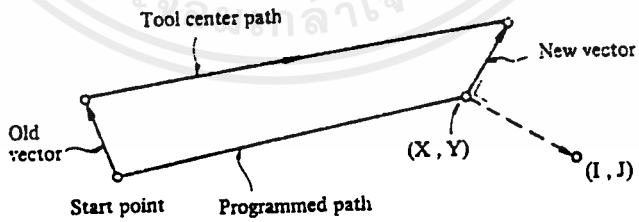
รูปที่ 2.55 Cutter compnsation cancel

4.2) <u>การชดเชยรัศมีดอกกัดด้านซ้าย (Cutter Compensation left , G41)</u> สามารถแบ่งรูปแบบการใช้เป็น 2 กรณี ได้ดังนี้

1) ใช้ร่วมกับ G00 , G01 การเคลื่อนที่จะเคลื่อนที่ขนานกับแนวการเคลื่อนที่ของ Program (Programmed path) ไปยังจุดบน new vector ซึ่ง new vector จะตั้งฉากกับทิศทางที่จะเคลื่อนที่ในคำสั่งต่อไป

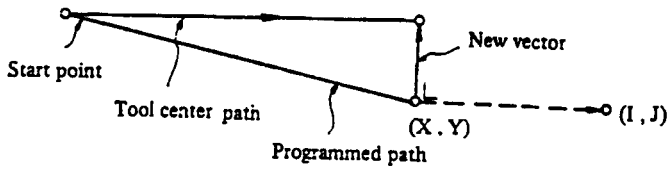
รูปแบบ G41 X__ Y__ F__ *

เช่น G41 X50.0 Y50.0 F150.0 *



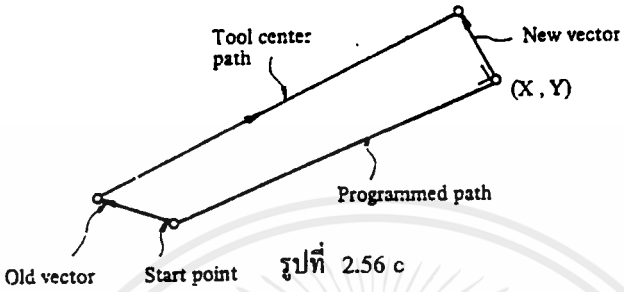
รูปที่ 2.56 a Cutter compensate left

ถ้า old vector = 0 การเคลื่อนที่จะเริ่มจาก start point ไปยังจุดบน new vector ทันที (ดูรูปที่ 2.56 b)



รูปที่ 2.56 b

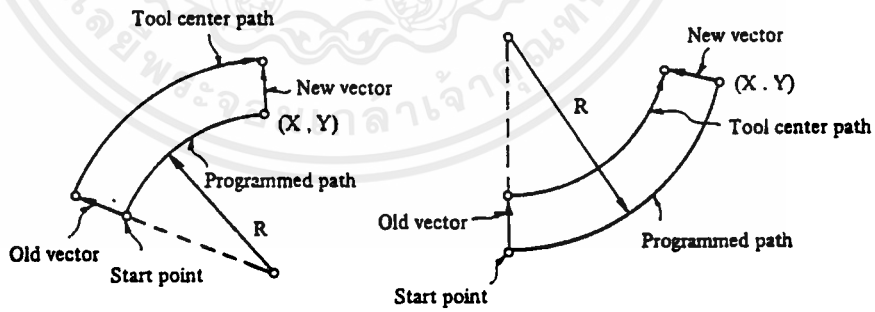
ถ้าไม่มีคำสั่งต่อไป new vector จะตั้งฉากกับ Programmed path (ดูรูป 2.56 c)



รูปที่ 2.56 c

<u>หมายเหตุ</u> ขนาดของ new vector จะกำหนด Tool center path ซึ่งการคำนวณได้กล่าวถึงในส่วนทฤษฎีทางคณิตศาสตร์สำหรับ CNC

2) ใช้ร่วมกับ G02 , G03 การพิจารณา new vector จะเหมือนกับกรณีแรกคือพิจารณาทิศทางการเคลื่อนที่เป็นหลัก ซึ่งการเคลื่อนที่จะขนานไปตาม Programmed path แต่มีรัศมีการเคลื่อนที่เพิ่มขึ้นตามรัศมี Cutter และ new vector จะมีทิศทางขนานกับทิศทางจาก arc center ไปยัง end point (ดูรูป 2.56 d)

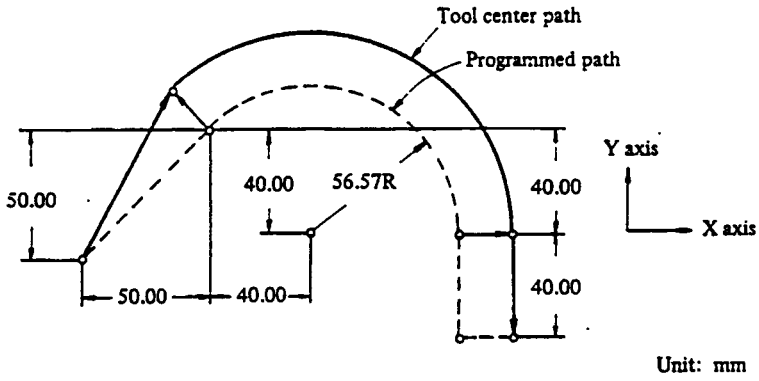


รูปที่ 2.56 d

4.3) <u>การชดเชยรัศมีดอกกัดด้านขวา (Cutter Compensation right , G42)</u> การชดเชยทางด้านขวา รูปแบบจะเหมือนกับการชดเชยทางด้านซ้าย แต่ทิศทางของ old vector และ new vector จะเป็นในทิศทางตรงกันข้าม จึงไม่ขอกล่าวรายละเอียด

<u>หมายเหตุ</u> การชดเชยดอกกัดทั้งซ้ายและขวาในโครงงานนี้จัดทำเฉพาะในแกน X-Y เท่านั้น

Example of Program of cutter compensation



รูปที่ 2.57 ตัวอย่างการใช้โปรแกรม Compensate

N1 G91 G41 G01 X50.0 F150.0 D20.0 *

N2 G02 X96.57 Y40.0 R56.57 *

N3 G01 Y40.0 * (Incremental programming)

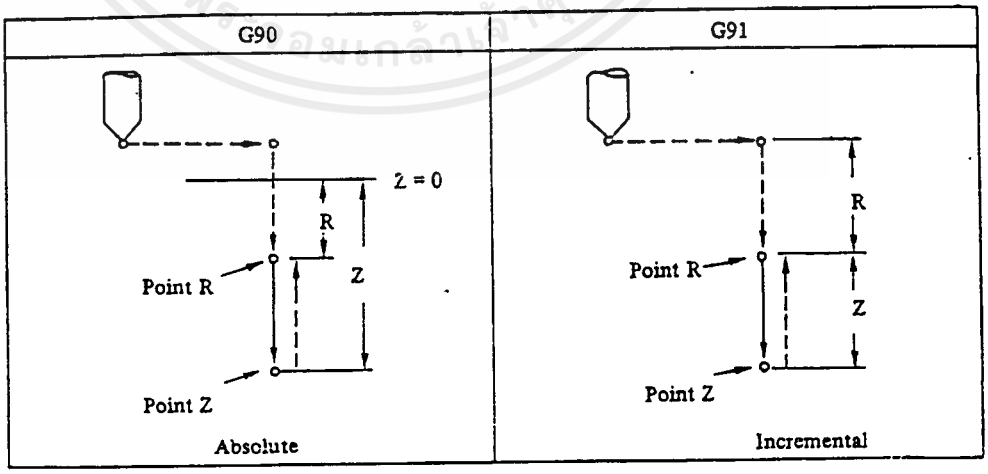
* *หมายเหตุ D หมายถึง เส้นผ่านศูนย์กลางดอกกัด (Cutter)*

5) **กลุ่ม 03**

เป็นการกำหนดวิธีการให้ขนาดของแบบในชิ้นงาน แบ่งเป็น

5.1) การให้ขนาดแบบสัมบูรณ์ (Absolute Dimensions , G90) จะใช้การอ้างอิงจากจุดคงที่จุดหนึ่งในแบบของชิ้นงาน

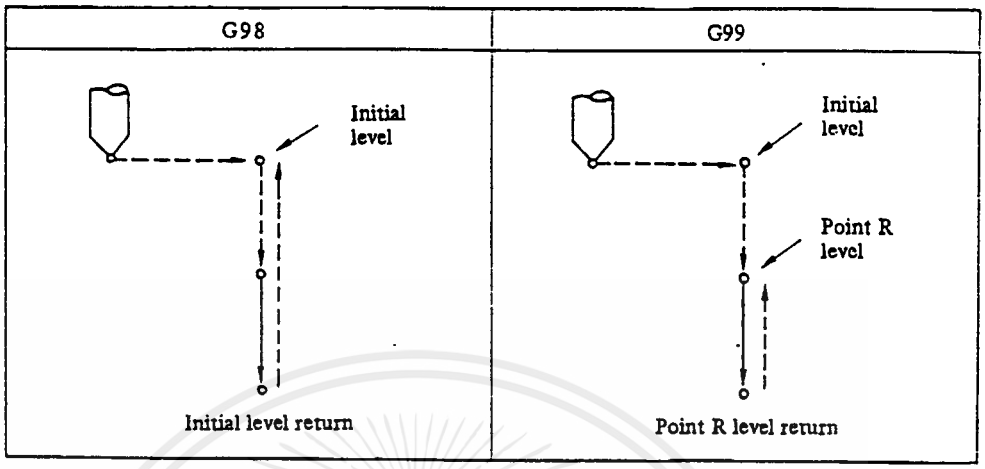
5.2) การให้ขนาดแบบต่อเนื่อง (Incremental Dimensions , G91) การวัดขนาดต่างๆ จะอ้างอิงจากตำแหน่งการให้ขนาดครั้งสุดท้าย



รูปที่ 2.58 Absolute and Incremental Programming

6) กลุ่ม 04

เป็นการกำหนดรูปแบบของการถอยกลับของ Cutter หลังจากทำการขุดเจาะเสร็จ (รูปแบบของการขุดเจาะจะอยู่ในกลุ่มคำสั่ง 09 ซึ่งจะกล่าวถึงต่อไป)



รูปที่ 2.59 Initial level and Point R level

แบ่งเป็น

6.1) ถอยกลับสู่ความสูงเริ่มต้น (Initial Level , G98)

6.2) ถอยกลับสู่ความสูงอ้างอิง (Reference Level , G99)

7) กลุ่ม 09

เป็นกลุ่มคำสั่งกำหนดรูปแบบการขุดเจาะลงบนชิ้นงาน ในลักษณะต่างๆ กัน ซึ่งสำหรับโครงงานนี้จะมีรูปแบบของการขุดเจาะ ดังนี้

การทำการขุดเจาะจะแบ่งรายละเอียดปลีกย่อยของคำสั่งที่ใช้เป็น

1) รูปแบบข้อมูล (Data format) ได้แก่ G90 หรือ G91 (Absolute หรือ Incremental)

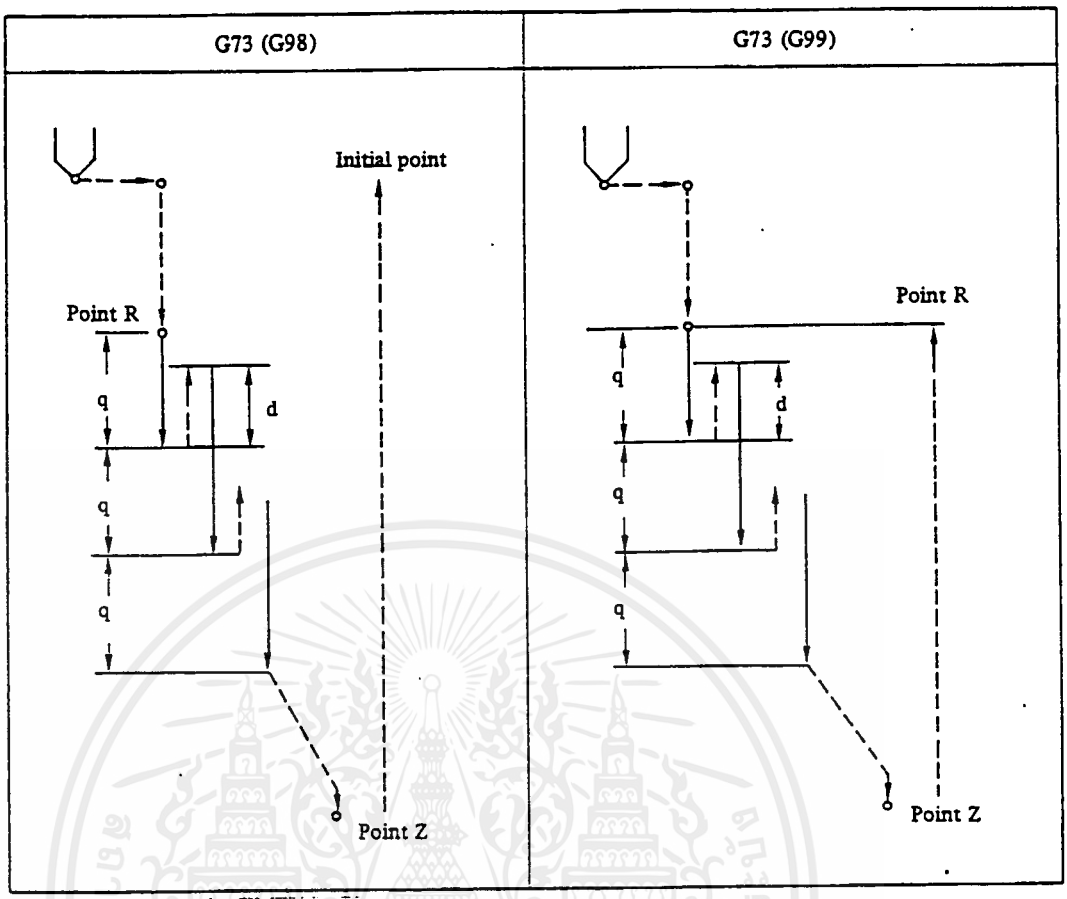
2) ตำแหน่งถอยกลับหลังจากขุดเจาะเสร็จ (Return point level) ได้แก่ G98 หรือ G99

3) รูปแบบการขุดเจาะ (Drilling Mode)

7.1) การขุดเจาะกระแทกเป็นจังหวะ (High-Speed Peck-Drilling Canned Cycle , G73) ขั้นตอนการทำงานของ G73 จะเป็นดังรูป และเมื่อเจาะเสร็จ จะถอยกลับไป R level หรือ Initial level ขึ้นอยู่กับว่าจะใช้ร่วม G98 หรือ G99

รูปแบบ G73 X__ Y__ Z__ Q__ R__ F__ *

* โดยที่ Q หมายถึง ระยะ Cut-in



รูปที่ 2.60

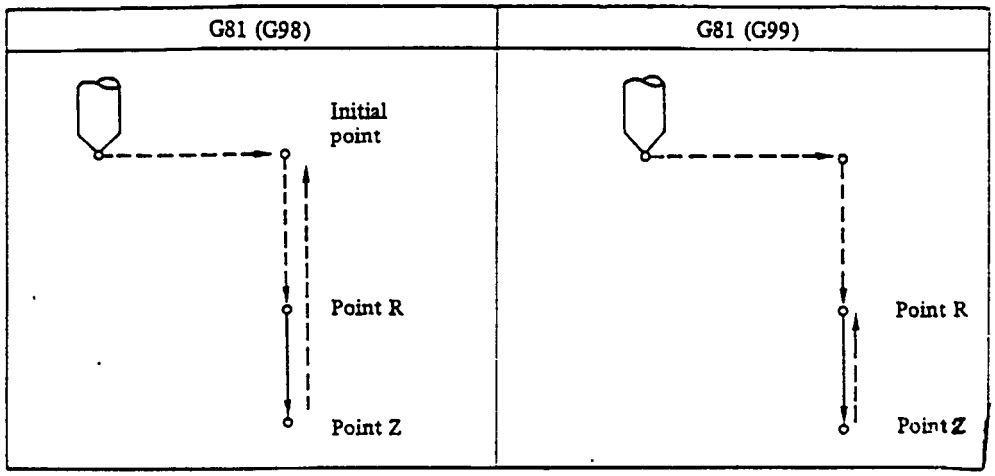
เช่น G91 G98 G73 X10.0 Y15.0 Z-1.72 Q0.61 R-6.3 F30 *

Q = ระยะ cut-in

F = อัตราเร็วการขุด

7.2) <u>การขุดเจาะเป็นจังหวะ ถ้วานรูเป็นจุดๆ (Drilling Canned Cycle , SpoyBoring Cycle , G81)</u> ขั้นตอนการทำงาน จะเคลื่อนที่ด้วยความเร็วสูงสุดไปที่ R point และเจาะรูด้วย Feed rate และเมื่อเจาะเสร็จ จะถอยกลับไปที่ R level หรือ Initial level ขึ้นอยู่กับว่าจะใช้ร่วมกับ G99 หรือ G98 ด้วยความเร็วสูงสุด (Rapid traverse)

รูปแบบ G81 X__ Y__ Z__ R__ F__ *



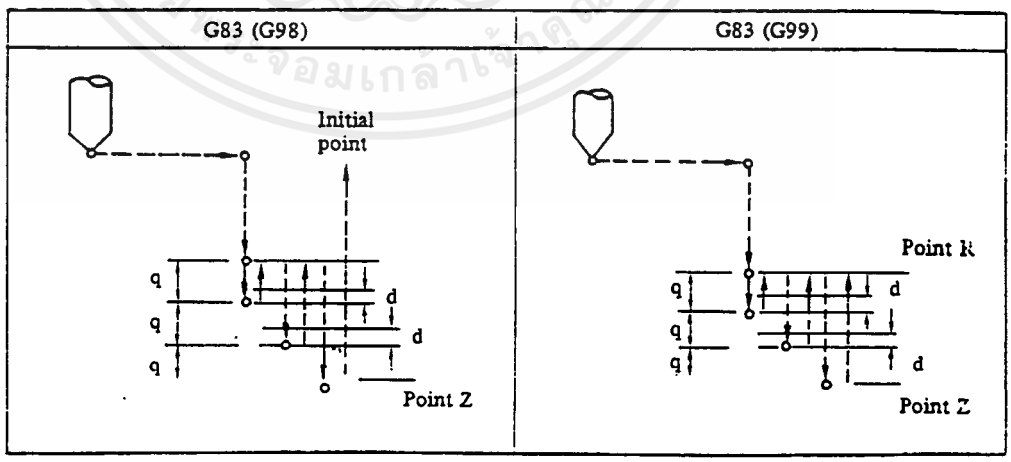
รูปที่ 2.61

7.3) <u>การขุดเจาะเป็นจังหวะ การคว้านผายปากรู (Drilling Cycle , Counter Boring Cycle , G82)</u> การทำงานจะเหมือนกับ G81 แต่จะหยุด (เป็นเวลา P sec) เมื่อถึงจุดล่างสุดของรู และถอยกลับด้วยความเร็วสูงสุด

รูปแบบ G82 X__ Y__ Z__ R__ P__ F__ *

7.4) <u>การขุดเจาะกดเป็นจังหวะ (Peck drilling Canned Cycle , G83)</u> ขั้นตอนการขุดเจาะจะคล้ายกับ G73 แต่จะแตกต่างตรงที่ หลังจาก Cut-in ระยะ Q จะถอยกลับมาที่ R level ทุกครั้ง ซึ่งต่างกับ G73 ซึ่งจะถอยกลับมาเป็นระยะ D เท่านั้น

รูปแบบ G83 X__ Y__ Z__ R__ Q__ F__ *



รูปที่ 2.62

7.5) <u>การคว้านรูเป็นจังหวะ (Boring Cycle , G85)</u> การทำงานจะคล้ายกับ G81 แต่จะถอยกลับด้วยความเร็ว Feed rate

7.6) การคว้านรูเป็นจังหวะ (Boring Cycle , G86) การทำงานจะคล้าย G81 แต่ดอกัดจะหยุดเมื่อถึงจุดล่างสุดของรู และจะถอยกลับด้วยความเร็วสูงสุด

7.7) การคว้านรูเป็นจังหวะ , เจาะกายเศษ (Boring Cycle , G89) การทำงาน จะคล้าย G85 แต่จะหยุด(เป็นเวลา P sec) เมื่อถึงจุดล่างสุดของรูและถอยกลับด้วยความเร็ว Feed rate

2.8.2. ชุดคำสั่ง M (M Function หรือ Miscellaneous Function)

ชุดคำสั่ง M จะใช้ควบคุม Cutter หรือ Spindle ชุดคำสั่งในโครงงานนี้ที่ได้จัดทำ (มาตรฐานจาก Fanuc)

คำสั่งรหัส	ความหมาย
M02	จบโปรแกรม
M03	ให้ Cutter หมุนตามเข็มนาฬิกา
M04	ให้ Cutter หมุนทวนเข็มนาฬิกา
M05	ให้ Cutter หยุดหมุน

นอกจากชุดคำสั่ง G และ M แล้ว ยังมีคำสั่งย่อย ซึ่งใช้ร่วมกับชุดคำสั่ง G และ M ได้แก่

2.8.3 คำสั่ง F (Feed Function)

จะเป็นตัวกำหนดความเร็วของการ Cut-in (Cutting Feed Rate) ซึ่งค่าป้อนเข้าไปจะเป็น inch / minute หรือ mm. / minute

2.8.4 คำสั่ง S (Spinle Speed Function)

เป็นคำสั่งกำหนดความเร็วของ Spinle มีหน่วยเป็น rpm.

บทที่3
เครื่องมือและอุปกรณ์

ในระบบของเครื่องกัดแนวดิ่งซีเอ็นซีควบคุมโดยไมโครคอมพิวเตอร์นั้นจะมีส่วนประกอบหลักๆ อยู่ 2 ส่วนดังนี้

3.1. ส่วนของเครื่องกัดแนวดิ่งซีเอ็นซี

3.2. ส่วนของระบบควบคุมเซอร์โวมอเตอร์

โดยในแต่ละส่วน ประกอบไปด้วยอุปกรณ์และเครื่องมือต่างๆ ต่อไปนี้

3.1 ส่วนของเครื่องกัดแนวดิ่งซีเอ็นซี

เครื่องกัดแนวดิ่งซีเอ็นซีที่ใช้ในการทดลอง มีรายละเอียดทางด้านเทคนิคดังนี้[1]

3.1.1. โต๊ะงานทำด้วยเหล็กหล่อ ขนาดกว้าง 430 ยาว 600 หนา 90 มม. มีร่องตัวทีขนาด 10 มม. ตามแนวยาวของโต๊ะงานจำนวน 3 ร่อง

3.1.2. องค์ประกอบของโต๊ะงานทำด้วยเหล็กหล่อ ขนาดกว้าง 350 ยาว 1150 สูง 180 มม.

3.1.3. ฐานเครื่องทำด้วยเหล็กหล่อขนาดกว้าง 650 ยาว 1260 สูง 315 มม.

3.1.4. Column ทำด้วยเหล็กหล่อขนาดกว้าง 380 ยาว 400 สูง 1200 มม.

3.1.5. Ball screw ขนาด 25 มม. ยาว 650 มม. ระยะ pitch 5 มม. ระยะ lead 5 มม. ในแนวแกน X,Y

3.1.6. Ball screw ขนาด 25 มม. ยาว 875 มม. ระยะ pitch 5 มม. ระยะ lead 5 มม. ในแนวแกน Z

3.1.7. Linear Motion ในแนวแกน X,Y และ Z

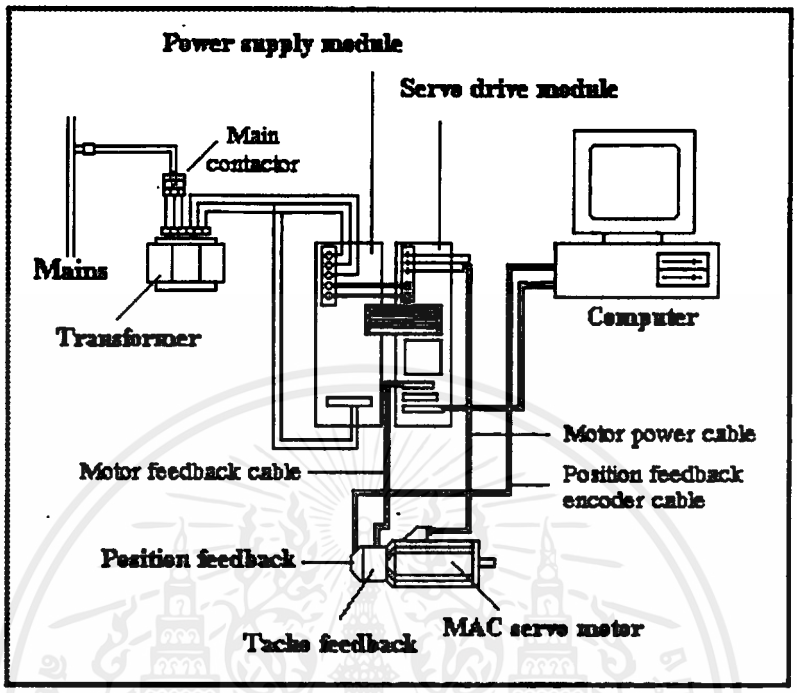
3.1.8. ชุดเพลาขับหัวเครื่อง มีรายละเอียดดังนี้

- มีขนาดของรูเพลาที่ใช้กับด้ามมีด ขนาดมาตรฐานไม่เล็กกว่า BT30
- ทนความเร็วรอบสูงสุดไม่ต่ำกว่า 3500 รอบ / นาที
- ขับเคลื่อนหัว Spindle ด้วยสายพาน
- ลูกปืนที่ใช้เป็นแบบ high precision bearing

รูปที่ 3.1 แสดงโครงสร้างของเครื่องกัดแนวดิ่งซีเอ็นซี

3.2 ส่วนของระบบควบคุมเซอร์โวมอเตอร์

ประกอบด้วยอุปกรณ์ต่างๆ ดังแสดงในรูปที่ 3.2



รูปที่ 3.2 แสดงระบบควบคุมเซอร์โวมอเตอร์

จากภาพจะพบว่าในระบบควบคุมมีอุปกรณ์ที่สำคัญๆ ดังนี้

3.2.1. การ์ดควบคุมสำเร็จรูป Model 5650[4]

3.2.1.1. ลักษณะและความสามารถ

1) ใช้ชุดชิพของ PMD รุ่น MC1401
2) รีจิสเตอร์เก็บค่าตำแหน่ง,ความเร็ว,ความเร่ง และเจิร์ค ขนาด 32 บิต
3) สามารถรับค่าจากลิมิตสวิตช์ ป้องกันการเคลื่อนที่เกินขอบเขตได้ 2 ตัว ต่อ 1 แกน
4) มีรูปแบบการเคลื่อนที่ 3 แบบ คือ S-curve,Trapezoidal และ Velocity contouring
5) ใช้เวลา 100 µS ต่อการ update ข้อมูลหนึ่งครั้ง
6) มี Electronic Gearing
7) มีรูปแบบของ Digital Filter 2 แบบ คือ PID และ PIVFF
8) สามารถรับสัญญาณพัลซ์ที่มาจากเอนโคดเดอร์ได้สูงถึง 1 ล้านพัลซ์ต่อวินาที

9) มีรีจิสเตอร์ที่ใช้เก็บค่าตำแหน่ง Home และ Index ความเร็วสูง

10) มีสัญญาณควบคุม 2 แบบให้เลือก คือ 16-bit ± 10V DAC และ 10-bit PWM

11) สามารถเปลี่ยนแปลงรูปแบบการเคลื่อนที่และพารามิเตอร์ได้ขณะที่มีการทำงานอยู่

12) มี counter แบบ 1/T สำหรับการเคลื่อนที่ที่ความเร็วต่ำๆ

13) มีการหยุดมอเตอร์โดยอัตโนมัติเมื่อมีข้อผิดพลาดเกิดขึ้น

14) ผู้ใช้สามารถกำหนดอินเตอร์รัพได้

15) ใช้พอร์ต I/O ขนาด 8 บิต

3.2.1.2. การใช้งานการ์ดควบคุมสำเร็จรูป Model 5650

1) <u>การติดตั้ง</u>

การ์ดควบคุม ต้องการใช้ไฟเลี้ยงขนาด +5 Vdc และ +12 Vdc จาก ISA busในเครื่องคอมพิวเตอร์ จึงต้องทำการเสียบการ์ดควบคุมลงในช่องเสียบของ ISA bus ที่มีอยู่บน Main Board ของคอมพิวเตอร์ แล้วทำการเซ็ต Address ของการ์ดควบคุมให้อยู่ในตำแหน่งที่ยังว่างอยู่ ในที่นี้ เซ็ตค่า Address เป็น 300H สำหรับการเซ็ตค่า Address ของการ์ดควบคุมนั้นสามารถทำได้โดย การเซ็ต Jumper W2 เสียใหม่(ดูรูป ก.1,ก.2 ในภาคผนวก ก.)

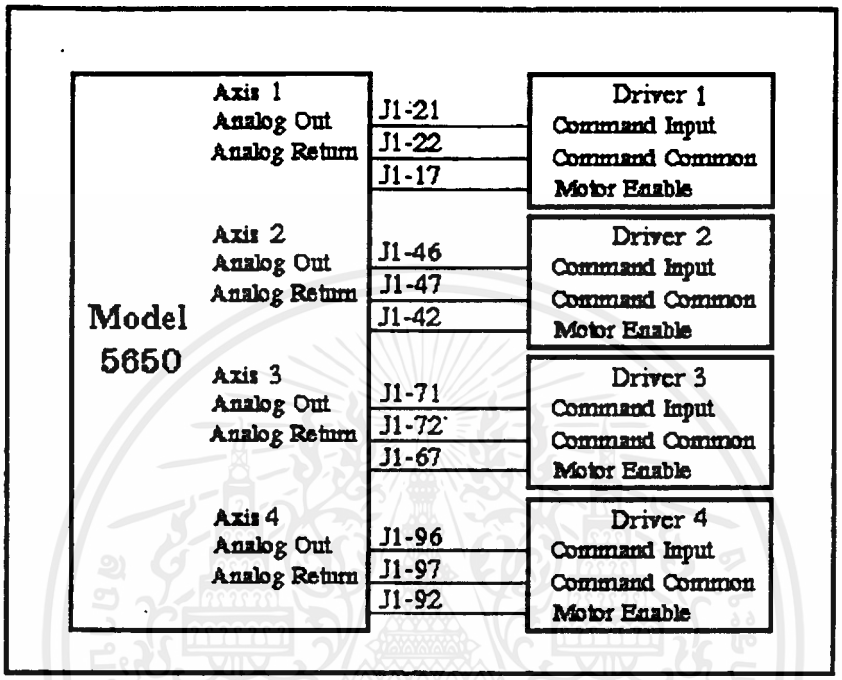
2) <u>การเชื่อมต่อ</u>

การเชื่อมต่อระหว่างการ์ดควบคุมกับอุปกรณ์ภายนอกต่างๆเช่น Driver,Encoder และ Limit Switchs สามารถทำได้โดย วิธีการต่อไปนี้

2.1) Driver - สัญญาณควบคุมที่ส่งจากการ์ดควบคุมไปยัง Driver นั้นเราจะใช้สัญญาณไฟฟ้าขนาด 0 ถึง ± 10V ในการควบคุมความเร็วและทิศทางการหมุนของ Servo Motor ดังนั้นสายสัญญาณที่เชื่อมระหว่างการ์ดควบคุม และ Driver จะใช้ 2 เส้นต่อการควบคุม Servo Motor 1แกน โดยเส้นหนึ่งเป็นสายของสัญญาณไฟฟ้าแบบAnalogส่วนอีกเส้นหนึ่งจะเป็นสายสัญญาณอ้างอิง(Analog Return)สำหรับสายสัญญาณทั้ง Analog Output และ Analog Return นั้นจะต่อออกมาจากConnector J1 (ดูรูป ก.3,ก.4 ในภาคผนวก ก.)

สัญญาณอีกตัวหนึ่งที่ส่งจากการ์ดควบคุมไปยัง Driver ก็คือ Motor Enable Signal ซึ่งทำหน้าที่ในการยกเลิกการใช้งาน (Disable) Driver ชั่งคราวในช่วงที่การ์ดควบคุมเริ่มทำงาน (Power-up) เพื่อป้องกันไม่ให้เกิดความผิดพลาดขึ้นในระหว่างเริ่มเปิดเครื่องหลังจากที่การ์ดควบคุมพร้อมที่จะทำงานแล้วจึงทำการเรียกใช้(Enable) Driver ตาม

ปกติ โดยในแต่ละแกนจะมีสัญญาณนี้ 1 เส้น โดยทำการต่อออกมา จาก Connector J1 (ดูรูป ก.3,ก.5 ในภาคผนวก ก.)ในกรณีที่ต้องการเปลี่ยนสัญญาณ Output ที่สายสัญญาณนี้จากเดิมในสถานะ Power-up (± 5 Vdc) และ สถานะพร้อมทำงาน (0 Vdc) ไปเป็นสถานะ Power-up(0 Vdc)และสถานะพร้อมทำงาน(± 5 Vdc) ก็สามารถทำได้โดยการเซ็ต Jumper W3 ใหม่(ดูรูป ก.6 ในภาคผนวก ก.)



รูปที่ 3.3 แสดงการเชื่อมต่อระหว่าง Model 5650 กับ Driver

2.2) Encoder - สำหรับ Encoder ที่ใช้นั้นจะเป็นแบบ Differential Encoder ซึ่งสัญญาณจะมีทั้งหมด 3-เฟสคือ เฟสA,เฟสB,และ Index แต่ละเฟสก็จะมีสายสัญญาณ 2 เส้นคือสายสัญญาณจริงหนึ่งเส้นและสัญญาณ Completement อีกหนึ่งเส้นรวม 6 เส้นอีก 2 เส้นจะเป็นสายไฟเลี้ยงที่จ่ายให้กับ Encoder รวมทั้งหมดจะมีสายสัญญาณอยู่ 8 เส้นต่อ Encoder 1 ตัว ในส่วนของการต่อเข้ากับการ์ดควบคุมนั้นสามารถทำได้โดยต่อสายเข้ากับ Connector J1(ดูรูป ก.3,ก.7ในภาคผนวก ก.)

2.3) Limit Switchs - ในแต่ละแกนของ Servo Motor จะต้องใช้ Limit Switch จำนวน 3 ตัว โดยแบ่งหน้าที่เป็น

- Over-travel Limit Input จำนวน 2 ตัว ทำหน้าที่ในการส่งสัญญาณไปยังการ์ดควบคุมเพื่อบอกให้ทราบว่าขณะนั้นโต๊ะงานได้เคลื่อนที่ไปจนสุดขอบเขตของการเคลื่อนที่แล้ว

- Home Inpuut จำนวน 1 ตัวทำหน้าที่ในการส่งสัญญาณไปยังการ์ดควบคุมเพื่อเพิ่มหรือลดค่าใน Position Capture Register ซึ่งสามารถนำผลจากการเปลี่ยนแปลงนี้ไปประยุกต์ได้

สำหรับสัญญาณจาก Limit Switch ที่จะต่อเข้าการ์ดควบคุมนั้น จะต้องต่อเข้ากับ Connector J1(ดูรูป ก.3,ก.8 ในภาคผนวก ก.)

3) การควบคุม

การควบคุมการทำงานของการ์ดควบคุมให้สามารถทำงานได้ตามที่เราต้องการนั้น สามารถทำได้โดยการเขียนโปรแกรมสั่งงานการ์ดผ่านทางคอมพิวเตอร์ โดยภาษาที่จะใช้การเขียนโปรแกรมนั้น ในที่นี้จะใช้ภาษา C เนื่องจาก เราจำเป็นจะต้องใช้ชุดคำสั่งที่ให้มาพร้อมกับการ์ดควบคุม ชุดคำสั่งนี้ปรากฏอยู่ในโปรแกรม host_io.c ซึ่งเขียนขึ้นด้วยภาษา C จึงเป็นการง่ายที่จะใช้ภาษา C ในการเขียนโปรแกรมควบคุม

ในส่วนรายละเอียดของโปรแกรมควบคุมที่จะเขียนนั้น ก็จะเป็นคำสั่งที่เรียกใช้งานชุดคำสั่งในโปรแกรม host_io.c เสียเป็นส่วนใหญ่ ดังนั้น หัวข้อต่อไปที่จะกล่าวถึงจึงเป็นเนื้อหาเกี่ยวกับความหมายของคำสั่งแต่ละตัวในโปรแกรม host_io.c

4) ชุดคำสั่ง

แบ่งตามลักษณะการใช้งานได้ 8 กลุ่ม ดังนี้

4.1) ควบคุมแกน (Axis Control)

SET_1 เปลี่ยนแกนปัจจุบันไปยังแกน 1

คำอธิบาย SET_1 จะเปลี่ยนแกนปัจจุบันไปยังแกน 1 คำสั่งทั้งหมดที่ตามมาภายหลังจะมีผลต่อแกน 1 ทั้งหมด ดูคำสั่ง GET_STATUS ประกอบ

SET_2 เปลี่ยนแกนปัจจุบันไปยังแกน 2

คำอธิบาย SET_2 จะเปลี่ยนแกนปัจจุบันไปยังแกน 2 คำสั่งทั้งหมดที่ตามมาภายหลังจะมีผลต่อแกน 2 ทั้งหมด ดูคำสั่ง GET_STATUS ประกอบ

SET_3 เปลี่ยนแกนปัจจุบันไปยังแกน 3

คำอธิบาย SET_3 จะเปลี่ยนแกนปัจจุบันไปยังแกน 3 คำสั่งทั้งหมดที่ตามมาภายหลังจะมีผลต่อแกน 3 ทั้งหมด ดูคำสั่ง GET_STATUS ประกอบ

SET_4 เปลี่ยนแกนปัจจุบันไปยังแกน 4

คำอธิบาย SET_4 จะเปลี่ยนแกนปัจจุบันไปยังแกน 4 คำสั่งทั้งหมดที่ตามมาภายหลังจะมีผลต่อแกน 4 ทั้งหมด ดูคำสั่ง GET_STATUS ประกอบ

4.2) **ควบคุมรูปแบบการเคลื่อนที่**

SET_PRFL_TRAP เปลี่ยนรูปแบบการเคลื่อนที่เป็นแบบ trapezoidal point-to-point

คำอธิบาย SET_PRFL_TRAP จะเปลี่ยนรูปแบบการเคลื่อนที่เป็นแบบ trapezoidal point-to-point ในโหมดนี้จะต้องมีการระบุตำแหน่งที่จะเคลื่อนที่ไป (คำสั่ง SET_POS) , ความเร็วสูงสุด(คำสั่ง SET_VEL) และความเร่ง (คำสั่ง SET_ACC) ที่ใช้ โดยสามารถจะทำการเปลี่ยนค่าตำแหน่งและความเร็วได้ตลอดเวลาแม้ในขณะที่มีการเคลื่อนที่ แต่ความเร่งจะเปลี่ยนไม่ได้แกนใดที่ถูกกำหนดให้มีโหมดการเคลื่อนที่เช่นนี้ ก็จะอยู่ในโหมดนี้ตลอดไปจนกว่าจะมีการกำหนดโหมดการเคลื่อนที่ใหม่ให้ สำหรับในการเปลี่ยนโหมดการเคลื่อนที่จากโหมดอื่นมาเป็นโหมดนี้นั้น แกนที่ควบคุมอยู่จะต้องหยุดนิ่งเสียก่อน

SET_PRFL_VEL เปลี่ยนรูปแบบการเคลื่อนที่เป็นแบบ velocity contouring

คำอธิบาย SET_PRFL_VEL จะเปลี่ยนรูปแบบการเคลื่อนที่เป็นแบบ velocity contouring ในโหมดนี้จะต้องมีการกำหนดความเร่ง (คำสั่ง SET_ACC) และความเร็วสูงสุด (คำสั่งSET_VEL) ที่ใช้ โดยสามารถทำการเปลี่ยนค่าความเร่งและความเร็วสูงสุดได้ตลอดเวลา แม้ในขณะกำลังเคลื่อนที่ แกนใดที่ถูกกำหนดให้มีโหมดการเคลื่อนที่เช่นนี้ก็จะอยู่ในโหมดนี้ตลอดไป จนกว่าจะมีการกำหนดโหมดการเคลื่อนที่ใหม่ให้ นอกเหนือจากนั้น ยังสามารถที่จะเปลี่ยนโหมดการเคลื่อนที่จากโหมดอื่นมาเป็นโหมดนี้ได้ทุกขณะ

SET_POS กำหนดค่าตำแหน่ง

คำอธิบาย SET_POS จะกำหนดตำแหน่งปลายทางที่จะเคลื่อนที่ไป ซึ่งจะอยู่ในรูปจำนวนของ counts ซึ่งมีค่าในช่วง -1,073,741,824 ถึง +1,073,741,823 ค่าตำแหน่งที่กำหนดให้จะยังไม่ถูกนำไปใช้จนกว่าจะมีการใช้คำสั่ง UPDATE

SET_VEL กำหนดความเร็ว

คำอธิบาย SET_VEL จะกำหนดขนาดความเร็วสูงสุดที่จะมีได้ ซึ่งจะอยู่ในรูปจำนวน counts/sample มีค่าอยู่ระหว่าง 0 ถึง +16,383 ค่าความเร็วที่กำหนดให้จะยังไม่ถูกนำไปใช้จนกว่าจะมีการสั่ง UPDATE

SET_ACC กำหนดความเร่ง

คำอธิบาย SET_ACC จะกำหนดขนาดความเร่งที่จะใช้ ซึ่งจะอยู่ในรูปจำนวน counts/sample/sample โดยถ้าอยู่ในรูปแบบการเคลื่อนที่แบบ trapezoidal point-to-point จะมีค่าอยู่ระหว่าง 0 ถึง +8,191 แต่ถ้าอยู่ในรูปแบบการเคลื่อนที่แบบ velocity contouring นั้นจะมีค่าอยู่ระหว่าง -8,191 ถึง +8,191 ค่าความเร่งที่กำหนดให้จะยังไม่ถูกนำไปใช้จนกว่าจะมีการสั่ง UPDATE คำสั่งนี้จะใช้ได้เมื่ออยู่ในโหมด trapezoidal point-to-point หรือ velocity contouring เท่านั้น

CLR_PRFL กำหนดค่าความเร็ว ความเร่ง และตำแหน่งให้เป็น 0

คำอธิบาย CLR_PRFL กำหนดค่าความเร็ว ความเร่ง และตำแหน่งให้เป็นศูนย์ ในแต่ละรูปแบบของการเคลื่อนที่ ฟังก์ชันนี้จะยังไม่ทำงานจนกระทั่งมีการสั่ง UPDATE คำสั่งนี้มีประโยชน์ในการหยุดมอเตอร์อย่างกระทันหัน

SYNCH_PRFL กำหนด target position ให้มีค่าเท่ากับ actual position

คำอธิบาย SYNCH_PRFL จะทำการกำหนด target position ให้มีค่าเท่ากับ actual position ขณะนั้น คำสั่งนี้ใช้ได้กับรูปแบบการเคลื่อนที่ทุกชนิด มักใช้บ่อยๆ หลังจากเกิดความผิดพลาดในการเคลื่อนที่ เพื่อที่จะทำให้ระยะเป้าหมายที่ตั้งไว้กับระยะที่เคลื่อนที่ไปจริงมีค่าเท่ากันอีกครั้ง ฟังก์ชันนี้จะไม่ทำงานจนกระทั่งมีคำสั่ง UPDATE

ZERO_POS กำหนดค่า actual position และ target position ให้เป็นศูนย์

คำอธิบาย ZERO_POS จะทำการกำหนดค่า actual position และ target position ในแกนนั้นๆ ให้เป็นศูนย์ คำสั่งนี้จะไม่ทำงานจนกระทั่งมีคำสั่ง UPDATE และสามารถใช้ร่วมกับคำสั่ง SYNCH_PRFL ได้

GET_POS อ่านค่าตำแหน่งจากการ์ดควบคุม

คำอธิบาย GET_POS จะทำการอ่านค่าตำแหน่งปลายทางที่ถูกกำหนดโดยคำสั่ง SET_POS ค่าที่ส่งกลับมาจะอยู่ในหน่วยของจำนวน counts ซึ่งจะเป็นเลขจำนวนเต็มไม่มีเครื่องหมายขนาด 32 bit

GET_VEL อ่านค่าความเร็วจากการ์ดควบคุม

คำอธิบาย GET_VEL จะทำการอ่านค่าความเร็วที่ถูกกำหนดโดยคำสั่ง SET_VEL ค่าที่ส่งกลับมาจะอยู่ในหน่วยของ counts/sample ซึ่งจะเป็นเลขจำนวนเต็มไม่มีเครื่องหมาย ขนาด 32 bit

GET_ACC อ่านค่าความเร่งจากการ์ดควบคุม

คำอธิบาย GET_ACC จะทำการอ่านค่าความเร่งที่ถูกกำหนดโดยคำสั่ง SET_ACC ค่าที่ส่งกลับมาจะอยู่ในหน่วยของ counts/sample/sample ซึ่งจะเป็นเลขจำนวนเต็ม อาจจะมีเครื่องหมายหรือไม่มีเครื่องหมาย ขึ้นอยู่กับรูปแบบของการเคลื่อนที่ที่ใช้

GET_TRGT_POS อ่านค่า target position

คำอธิบาย GET_TRGT_POS จะทำการอ่านค่าตำแหน่งที่จะเคลื่อนที่ไปซึ่งคำนวณโดยการ์ดควบคุม ค่านี้แสดงถึง target position สำหรับแกนนั้นๆที่จะเคลื่อนที่ไปในช่วงเวลาหนึ่งๆ คำสั่งนี้สามารถทำงานในทุกๆ รูปแบบของการเคลื่อนที่ โดยจะทำการส่งค่ากลับมาอยู่ในหน่วยของ counts ซึ่งเป็นเลขจำนวนเต็มมีค่าอยู่ระหว่าง -1,073,741,824 ถึง +1,073,741,823 คำสั่งนี้มีประโยชน์ในการดูค่าตำแหน่งที่ได้จากการคำนวณของการ์ดควบคุม และประสิทธิภาพของระบบเซอร์โว

GET_ACTL_POS อ่านค่า actual position

คำอธิบาย GET_ACTL_POS จะทำการอ่านค่าตำแหน่งที่เคลื่อนที่ไปได้จริงของมอเตอร์ ค่าที่ส่งกลับมาจะอยู่ในหน่วยของ counts ซึ่งจะเป็นเลขจำนวนเต็มมีเครื่องหมายขนาด 32 bit

GET_TRGT_VEL อ่านค่า target velocity

คำอธิบาย GET_TRGT_VEL จะทำการอ่านค่าความเร็วที่คำนวณโดยการ์ดควบคุม ค่านี้จะแสดงถึง target velocity สำหรับแกนนั้นๆ ที่จะเคลื่อนที่ไปในช่วงเวลาหนึ่งๆ ค่าที่ส่งกลับมาจะอยู่ในหน่วยของ counts/sample ซึ่งเป็นเลขจำนวนเต็มมีเครื่องหมายขนาด 32 bit ในช่วง-16,384 ถึง +16,383 คำสั่งนี้มีประโยชน์ในการดูค่าความเร็วที่คำนวณโดยการ์ดควบคุม และประสิทธิภาพของระบบเซอร์โว

4.3) **ควบคุม Digital Filter**

SET_FLTR_PID เปลี่ยนโหมดของ digital filter เป็นแบบ PID

คำอธิบาย SET_FLTR_PID จะทำการเปลี่ยนโหมดของ digital filter เป็นแบบ PID ในโหมดนี้ ผู้ใช้จะต้องกำหนดค่า Propotional gain (คำสั่ง SET_KP) , Derivative gain (คำสั่งSET_KD) , Integral gain (คำสั่ง SET_KI) และ Integration limit (คำสั่ง SET_I_LM) พารามิเตอร์ทั้งหมดเหล่านี้สามารถจะเปลี่ยนในขณะที่มอเตอร์ทำงานอยู่ได้

SET_KP กำหนดค่า Propotional gain

คำอธิบาย SET_KP ใช้กำหนดค่า Propotional gain สำหรับ Digital Filter โดยจะสามารถมีค่าได้ตั้งแต่ 0 ถึง 32767 ค่าที่กำหนดจะยังไม่ถูกนำไปใช้งานจนกว่าจะสั่ง UPDATE

SET_KD กำหนดค่า Derivative gain

คำอธิบาย SET_KD ใช้กำหนดค่า Derivative gain สำหรับ Digital Filter โดยจะสามารถมีค่าได้ตั้งแต่ 0 ถึง 32767 ค่าที่กำหนดจะยังไม่ถูกนำไปใช้งานจนกว่าจะสั่ง UPDATE

SET_KI กำหนดค่า Integral gain

คำอธิบาย SET_KI ใช้กำหนดค่า Integral gain สำหรับ Digital Filter โดยจะสามารถมีค่าได้ตั้งแต่ 0 ถึง 32767 ค่าที่กำหนดจะยังไม่ถูกนำไปใช้งานจนกว่าจะสั่ง UPDATE

SET_I_LM กำหนดค่า Integration limit

คำอธิบาย SET_I_LM ใช้กำหนดค่า Integration limit สำหรับ Digital Filter โดยจะสามารถมีค่าได้ตั้งแต่ 0 ถึง 32767 ค่าที่กำหนดจะยังไม่ถูกนำไปใช้งานจนกว่าจะสั่ง UPDATE

SET_POS_ERR กำหนดค่า position error สูงสุด

คำอธิบาย SET_POS_ERR ใช้กำหนดค่า position error สูงสุดสำหรับ Digital Filter โดยมีค่าอยู่ระหว่าง 0 ถึง 32767 ในแต่ละรอบของ servo loop ขนาดของ position error ที่คำนวณโดย Digital Filter จะถูกนำมาเปรียบเทียบกับค่า position error สูงสุดที่กำหนดไว้ ถ้าค่า position error มีค่าเกินกว่าที่กำหนดไว้ มอเตอร์จะหยุดหมุน ค่า position error ที่ถูกกำหนดโดยคำสั่งนี้สามารถนำไปใช้ได้เลยโดยไม่ต้องสั่ง UPDATE

GET_KP อ่านค่า Propotional gain

คำอธิบาย GET_KP จะทำการอ่านค่า Propotional gain ที่ถูกกำหนดโดยคำสั่ง SET_KP ค่าที่ส่งกลับมาจะอยู่ในรูปเลขจำนวนเต็มไม่มีเครื่องหมาย ขนาด 16 bit

GET_KD อ่านค่า Derivative gain

คำอธิบาย GET_KD จะทำการอ่านค่า Derivative gain ที่ถูกกำหนดโดยคำสั่ง SET_KD ค่าที่ส่งกลับมาจะอยู่ในรูปเลขจำนวนเต็มไม่มีเครื่องหมาย ขนาด 16 bit

GET_KI อ่านค่า Integral gain

คำอธิบาย GET_KI จะทำการอ่านค่า Integral gain ที่ถูกกำหนดโดยคำสั่ง SET_KI ค่าที่ส่งกลับมาจะอยู่ในรูปเลขจำนวนเต็มไม่มีเครื่องหมาย ขนาด 16 bit

GET_I_LM อ่านค่า Integration limit

คำอธิบาย GET_I_LM จะทำการอ่านค่า Integration limit ที่ถูกกำหนดโดยคำสั่ง SET_I_LM ค่าที่ส่งกลับมาจะอยู่ในรูปเลขจำนวนเต็มไม่มีเครื่องหมาย ขนาด 16 bit

GET_POS_ERR อ่านค่า Position error สูงสุด

คำอธิบาย GET_POS_ERR จะทำการอ่านค่า position error สูงสุด ที่ถูกกำหนดโดยคำสั่ง SET_POS_ERR ค่าที่ส่งกลับมาจะอยู่ในรูปเลขจำนวนเต็มไม่มีเครื่องหมาย ขนาด 16 bit

GET_ACTL_POS_ERR อ่านค่า Position error ที่เกิดขึ้น

คำอธิบาย GET_ACTL_POS_ERR จะทำการอ่านค่า position error ที่เกิดขึ้นในขณะนั้น โดยค่าที่ส่งกลับมาจะเป็นค่าผลต่างระหว่าง target position และ actual position (actual position ลบด้วย target position) ซึ่งเป็นตัวเลขมีเครื่องหมาย ขนาด 16 bit ในช่วงระหว่าง -32768 ถึง +32767 คำสั่งนี้ใช้ประโยชน์ในการดูและวิเคราะห์การติดตามค่าความผิดพลาดของระบบ

SET_AUTO_STOP_ON อนุญาตให้มอเตอร์หยุดโดยอัตโนมัติ

คำอธิบาย SET_AUTO_STOP_ON จะอนุญาตให้มอเตอร์หยุดหมุนในทันทีที่มีการเคลื่อนที่ผิดพลาดขึ้น ในโหมดนี้ มอเตอร์จะหยุดหมุน (เหมือนในคำสั่ง MTR_OFF) เมื่อเกิดการเคลื่อนที่ผิดลาดขึ้น (ดูคำสั่ง SET_POS_ERR) สามารถทำให้กลับเป็นปกติ (มอเตอร์ทำงานได้ตามปกติ) ได้โดยใช้คำสั่ง MYR_ON

SET_AUTO_STOP_OFF ไม่อนุญาตให้มอเตอร์หยุดโดยอัตโนมัติ

คำอธิบาย SET_AUTO_STOP_OFF จะไม่อนุญาตให้มอเตอร์หยุดหมุนซึ่งมีสาเหตุมาจากการเคลื่อนที่ที่ผิดพลาด ในโหมดนี้ มอเตอร์จะไม่หยุดหมุนเมื่อเกิดความผิดพลาดขึ้น

4.4) ควบคุมการเปลี่ยนค่าของพารามิเตอร์ต่างๆ

UPDATE ทำการเปลี่ยนค่าพารามิเตอร์ทันที

คำอธิบาย UPDATE จะทำการเปลี่ยนค่าพารามิเตอร์ใหม่ โดยใช้ค่าที่กำหนดให้ในคำสั่งที่มาก่อนหน้าคำสั่งนี้

MULTI_UPDATE ทำการเปลี่ยนค่าพารามิเตอร์ของแกนต่างๆ ที่กำหนดพร้อมกันทันที

คำอธิบาย MULTI_UPDATE จะทำการเปลี่ยนค่าพารามิเตอร์ต่างๆ ที่ได้กำหนดให้กับแต่ละแกนพร้อมกัน

4.5) **ควบคุมสถานะ / โหมด**

CLR_STATUS เคลียร์บิตแสดงสถานะทุกตัวเฉพาะแกนที่กำหนด

คำอธิบาย CLR_STATUS จะทำการรีเซ็ตบิตแสดงสถานะของแกนที่กำหนดให้ (bit - 6 ของ status word) คำสั่งนี้มีประโยชน์สำหรับการเคลียร์บิตสถานะทุกตัวระหว่างที่เริ่มเปิดเครื่องและระหว่างที่ใช้งานอยู่ สำหรับรายละเอียดเกี่ยวกับ status word นั้น ดูได้จากคำสั่ง GET_STATUS

GET_STATUS อ่านค่าของ Status word ของแกนที่กำหนด

คำอธิบาย GET_STATUS จะส่งค่าสถานะของแกนที่ต้องการทราบกลับมาโดยความหมายของแต่ละ bit ใน status word ปรากฏอยู่ในตาราง 3.1

Bit	Event
15-14	Unused (set to 0)
13-12	Current axis Bit 13 Bit 12 Axis 0 0 1 0 1 2 1 0 3 1 1 4
11-10	Reserved (may be 0 or 1)
9	Axis on/off status (1 indicates on)
8	Motor on/off status (1 indicates on)
7	Unusud (set to 0)
6	Negative limit switch (1 indicates negative limit triped)
5	Positive limit switch (1 indicates positive limit tripped)
4	Motion error (1 indicates motion error)
3	Position capture received (1 indicates posiion capture has occured)

2	Update breakpoint reached (1 indicates breakpoint reached)
1	Position wrap-around (1 indicates wrap)
0	Motion complete (1 indicates complete)

ตาราง 3.1 แสดงความหมายของแต่ละบิตในAxis Status

4.6) **ควบคุม Encoder**

SET_CAPT_INDEX กำหนดให้ค่าในรีจิสเตอร์เก็บค่าตำแหน่งความเร็วสูงเปลี่ยนแปลงเมื่อได้รับสัญญาณ index

คำอธิบาย SET_CAPT_INDEX จะกำหนดให้ค่าในรีจิสเตอร์เก็บค่าตำแหน่งความเร็วสูงเปลี่ยนแปลงไป เมื่อสัญญาณ index เข้ามา คำสั่งนี้จะใช้เมื่อโหมดของการรับค่าตำแหน่งเป็นแบบ incremental เท่านั้น

SET_CAPT_HOME กำหนดให้ค่าในรีจิสเตอร์เก็บค่าตำแหน่งความเร็วสูงเปลี่ยนแปลงเมื่อได้รับสัญญาณ home

คำอธิบาย SET_CAPT_HOME จะกำหนดให้ค่าในรีจิสเตอร์เก็บค่าตำแหน่งความเร็วสูงเปลี่ยนแปลงไป เมื่อสัญญาณ home เข้ามา คำสั่งนี้จะใช้เมื่อโหมดของการรับค่าตำแหน่งเป็นแบบ incremental เท่านั้น

GET_CAPT อ่านค่าที่เก็บไว้ในรีจิสเตอร์เก็บค่าตำแหน่งความเร็วสูง

คำอธิบาย GET_CAPT จะส่งกลับค่าที่เก็บอยู่ในรีจิสเตอร์เก็บค่าตำแหน่งความเร็วสูง โดยจะอยู่ในรูปเลขจำนวนเต็มมีเครื่องหมายขนาด 32 bit ในหน่วยของ counts

4.7) **ควบคุมมอเตอร์**

SET-OUTPUT-DAC16 กำหนดให้สัญญาณที่จะส่งไปควบคุมมอเตอร์อยู่ในรูปAnalog

คำอธิบาย SET_OUTPUT_DAC16 จะกำหนดโหมดสัญญาณที่จะส่งไปควบคุมมอเตอร์! อยู่ในโหมด 16-bit-DAC คำสั่งนี้จะมีผลต่อทุกๆแกน

MTR-ON ยอมให้มีสัญญาณออกไปควบคุมมอเตอร์ได้

คำอธิบาย MTR_ON จะอณุญาตให้ สัญญาณมอเตอร์สามารถออกไปได้เฉพาะ แกนที่กำหนดเท่านั้น

MTR-OFF ไม่ยอมให้มีสัญญาณออกไปควบคุมมอเตอร์ได้

คำอธิบาย MTR_OFF จะห้ามไม่ให้สัญญาณควบคุมมอเตอร์สามารถออกไปได้เฉพาะแกนที่กำหนด แต่จะส่งสัญญาณซึ่งมีค่า 0 ออกไปแทนคำสั่งนี้มีประโยชน์ในการยกเลิกการทำงานอย่างกะทันหัน

SET-MTR-CMD กำหนดค่าสัญญาณควบคุม ไปยังมอเตอร์โดยตรง

คำอธิบาย SET_MTR_CMD ใช้กำหนดค่าของสัญญาณควบคุมที่จะส่งไปมอเตอร์ โดยสามารถกำหนดได้โดยตรง ซึ่งมีค่าอยู่ระหว่าง-32767 ถึง +32676 แสดงถึงค่าสูงสุดของความเร็วที่หมุนตามเข็มนาฬิกาของมอเตอร์ ค่า 0 แสดงถึงมอเตอร์หยุด นิ่งค่า 32767 แสดงถึงค่าสูงสุดของความเร็วที่หมุนทวนเข็มนาฬิกาของมอเตอร์ ก่อนที่จะใช้คำสั่งนี้ต้องใช้คำสั่ง MTR_OFF นำก่อน

4.8) ควบคุมส่วนปลีกย่อย

AXIS-ON ยอมให้มีการส่งสัญญาณควบคุมไปยังแกนที่กำหนด(เปิดมอเตอร์)

คำอธิบาย AXIS_ONจะยอมให้มีการส่งสัญญาณควบคุมไปยังแกนที่กำหนดซึ่งสามารถที่จะสั่งให้ทำการเปิดมอเตอร์ได้ทุกขณะ แต่ไม่ควรทำเพราะอาจทำให้การเคลื่อนที่เกิดการผิดพลาดขึ้น

AXIS-OFF ไม่ยอมให้มีการส่สัญญาณควบคุมไปยังแกนที่กำหนด

คำอธิบาย AXIS_OFFจะไม่ยอมให้มีการส่งสัญญาณควบคุมไปยังแกนที่กำหนด สัญญาณที่ส่งออกจะมีค่าเป็น 0 เราสามารถที่จะสั่งให้ทำการเปิดหรือปิดมอเตอร์ได้ทุกขณะแต่ไม่ควรทำเพราะอาจทำให้การเคลื่อนที่เกิดการผิดพลาดขึ้น

SET-LMT-SENSE กำหนดรูปแบบการรับรู้สัญญาณจาก Limit Switch ของการ์ดควบคุม

คำอธิบาย SET_LMT_SENSE เป็นคำสั่งที่ใช้กำหนดรูปแบบการรับรู้สัญญาณของ

การ์ดควบคุมที่ส่งมาจาก Limit Switch โดยรูปแบบการรับรู้สัญญาณจะมี 2 แบบคือ จะทำงานที่ขาขึ้นของสัญญาณ(Active high) หรือ จะให้ทำงานที่ขาลงของสัญญาณ(Active low)

Bit	Description
15-8	Not used (set to 0)
7	Axis 4 negative limit switch (0 = active high)
6	Axis 4 positive limit switch (0 = active high)
5	Axis 3 negative limit switch (0 = active high)
4	Axis 3 positive limit switch (0 = active high)
3	Axis 2 negative limit switch (0 = active high)
2	Axis 2 positive limit switch (0 = active high)
1	Axis 1 negative limit switch (0 = active high)
0	Axis 1 positive limit switch (0 = active high)

ตาราง 3.2 แสดงค่าและความหมายในแต่ละบิตของ Limit Switch Register

จาก ตารางจะเห็นได้ว่าถ้าเราต้องการเซ็ตค่าของ Limit Switch ทั้งทางด้านบวกและด้านลบของแกนใดๆให้ทำงาน(Trig) ที่ขาขึ้นของสัญญาณก็ทำได้โดยกำหนดให้บิตนั้นมีค่าเป็น 0 หรือในทางตรงกันข้ามก็กำหนดให้เป็น 1แล้วแต่เราจะต้องการขึ้นอยู่กับระบบของเรา

GET-LMT-SWITCH อ่านค่าสถานะของ Register แสดงสถานะของ Limit Swtch

คำอธิบาย GET_LMT_SWITCH จะส่งค่าสถานะของ Register แสดงสภาวะของ Limit Switch แต่ละตัวในระบบ ซึ่งค่าที่ส่งกลับมาจะเป็นเลขจำนวนเต็มที่มีรูปแบบดังนี้

Bit	Description
15-8	Not used (set to 0)
7	Axis 4 negative limit switch (1 = tripped)
6	Axis 4 positive limit switch (1 = tripped)
5	Axis 3 negative limit switch (1 = tripped)
4	Axis 3 positive limit switch (1 = tripped)
3	Axis 2 negative limit switch (1 = tripped)

2	Axis 2 positive limit switch (1 = tripped)
1	Axis 1 negative limit switch (1 = tripped)
0	Axis 1 positive limit switch (1 = tripped)

ตาราง 3.3 แสดงค่าและความหมายใน Limit Switch Triggered Register

LMTS-ON กำหนดให้มีการรับสัญญาณจาก Limit Switch

LMTS-OFF กำหนดให้ไม่มีการสัญญาณจาก Limit Switch

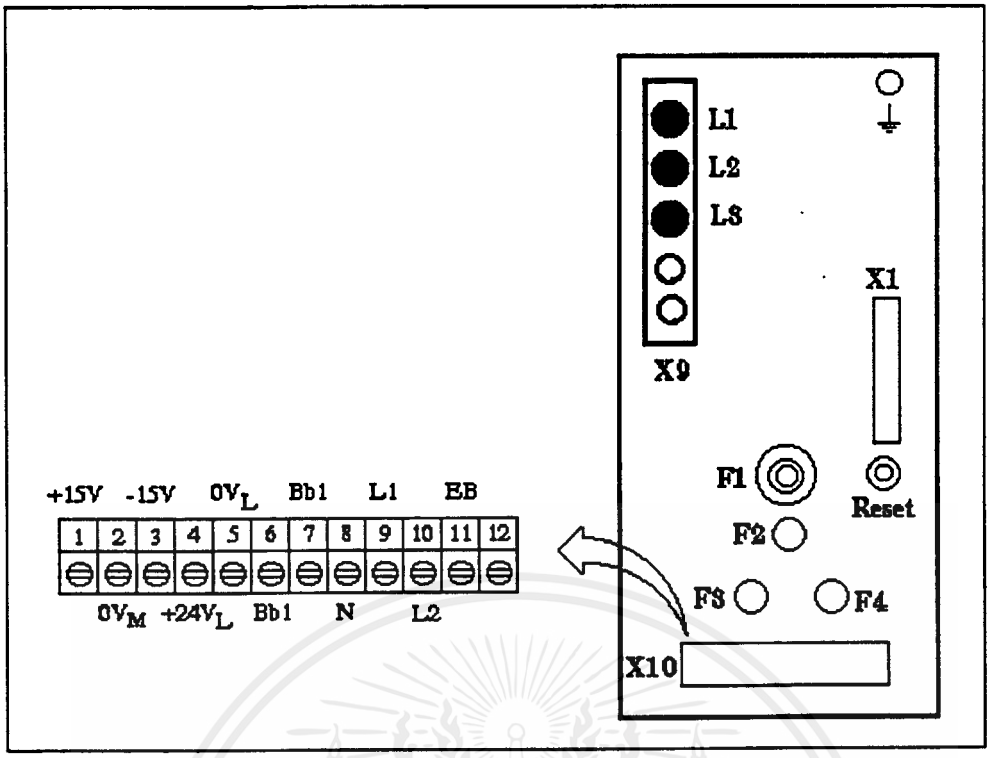
คำอธิบาย LMTS_OFF จะทำให้ค่าในRegister แสดงสภาวะของ Limit Switch ไม่มีการเปลี่ยนแปลงจึงส่งผลต่อคำสั่ง GET_LMT_SWITCH ด้วย

RESET ทำการ Reset ชิพของการ์ดควบคุม

คำอธิบาย Reset จะทำการReset ชิพของการ์ดควบคุม หลังจากที่มีการ Reset แล้งค่าต่างๆภายในตัวการ์ดควบคุมจะเป็นค่าดั้งเดิม (Default)

3.2.2. Power Supply Module[5]

ใช้ในการสร้างกระแส D.C. จากกระแส A.C. ที่ได้รับ เพื่อป้อนให้กับ Servo Drive Module สำหรับ Power Supply Module ที่ใช้ในการทดลองเป็นของ INDRAMAT รุ่น TVM 1.2 เป็น Power Supply ซึ่งสามารถจ่ายไฟให้กับ Servo Drive Module ได้ถึง 4 ตัว มีกำลังขับสูงสุด 4.1 KW มีลักษณะภายนอกดังรูป 3.3



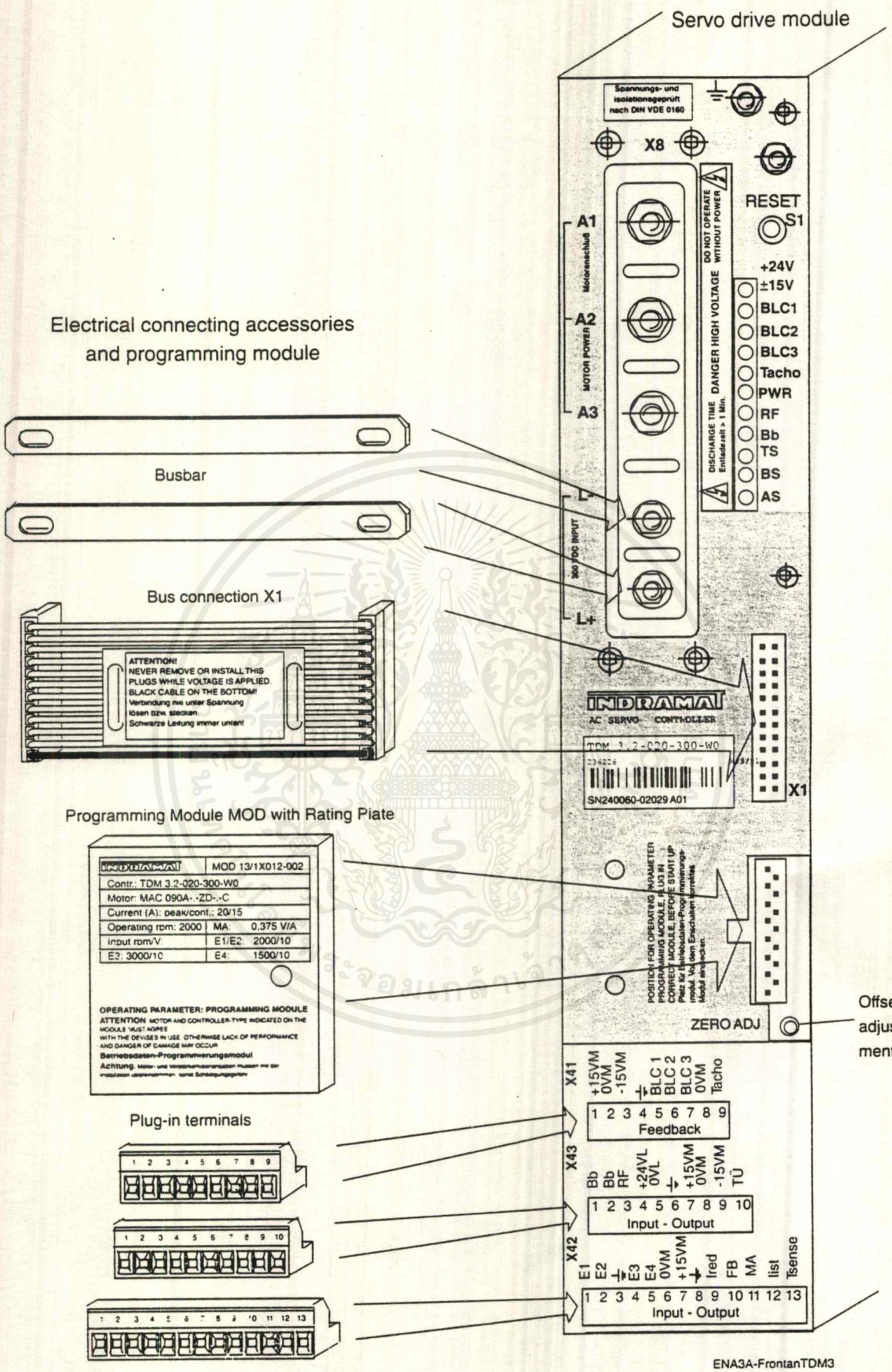
รูปที่ 3.3 แสดงลักษณะของ Power Supply Module

3.2.2.1. การใช้งาน

Power Supply Module (PSM) จะต้องเชื่อมต่อกับอุปกรณ์ 2 ตัว คือ Transformer และ Servo Drive Module (SM) Transformer จะต่อกับ PSM ผ่านสาย cable 3 เฟส โดยแต่ละเฟสจาก Transformer จะต่อเข้ากับ PSM ที่ Connector X9 ที่ตำแหน่ง L1 , L2 และ L3 ตามลำดับ SDM จะต่อกับ PSM ผ่าน Busbar และ Bus Connection X1 โดย Busbar จะต่อระหว่าง Connector X9 ของ PSM ที่ตำแหน่ง L- , L+ กับ Connector X8 ที่ตำแหน่ง L- , L+ ตามลำดับ ส่วน Bus Connection X1 จะต่อระหว่าง Connector X1 ของ PSM กับ Connector X1 ของ SDM และสำหรับในส่วนของ PSM เอง จะต้องมีการต่อไฟเลี้ยงขนาด 220 VAC จาก Connector L1 และ L2 ลงมายัง Connector X10 ของ PSM ที่ตำแหน่ง 8 และ 9 ตามลำดับด้วย

3.2.3.3. <u>Servo Drive Module</u>

ใช้ในการขับเซอร์โวมอเตอร์โดยจะรับไฟเลี้ยงจาก Power Supply Module และรับคำสั่งจากการ์ดควบคุม สำหรับ Servo Drive Module ที่ใช้ในการทดลองเป็นของ INDRAMAT รุ่น TDM 3.2 ซึ่งใช้สำหรับขับเซอร์โวมอเตอร์ตั้งแต่รุ่น MAC 63 ถึง MAC 112B มีลักษณะภายนอกดังรูป 3.4



รูปที่ 3.4 แสดงลักษณะของ Servo Drive Module

3.2.3.1 การใช้งาน

Servo Drive Module (SDM) จะต้องต่อเชื่อมกับอุปกรณ์อีก 2 ตัว คือ การ์ดควบคุมสำเร็จรูป Model 5650 และเซอร์โวมอเตอร์ ในส่วนของการต่อเชื่อมกับการ์ดควบคุมนั้นได้กล่าวไปแล้วในตอนต้น ดังนั้น ในที่นี้จะกล่าวถึงเฉพาะการเชื่อมต่อกับ เซอร์โวมอเตอร์เท่านั้น

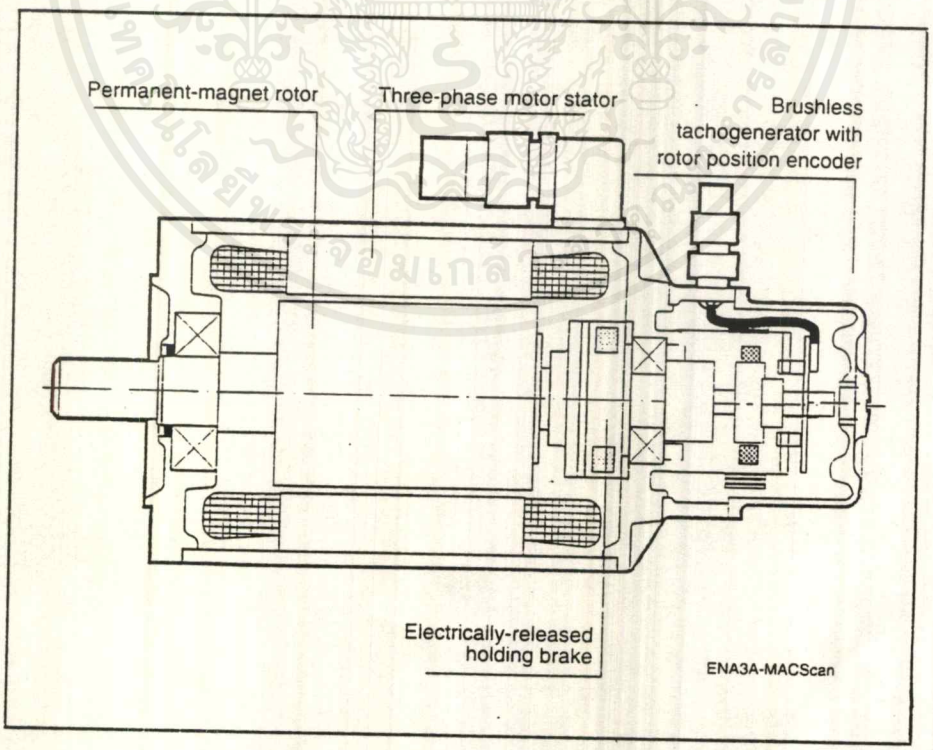
เซอร์โวมอเตอร์จะต่อกับ SDM ผ่านทางสาย Cable 2 เส้น โดยเส้นแรกเป็น Motor Power Cable (Indramat IN250) ต่อระหว่างเซอร์โวมอเตอร์กับ Connector X8 ของ SDM ที่ตำแหน่ง A1,A2 และ A3 ตามลำดับ (ต้องต่อให้ตรงกับอักษรที่เขียนไว้บนสายไฟ) ส่วนอีกเส้นหนึ่งเป็น Motor Feedback Cable (Indramat IN208) ต่อระหว่าง Tacho Feedback กับ Connector X41 ที่ตำแหน่ง 1 ถึง 8

3.2.4. MAC Servo Motor

เป็นเอซีเซอร์โวมอเตอร์แบบ 3 เฟส ที่มีลักษณะดังนี้

- สเตเตอร์แบบ 3 เฟส
- โรเตอร์แบบแม่เหล็กถาวร
- มี Electrical-release Brake

โดยมีโครงสร้างและส่วนประกอบต่างๆ ดังรูป 3.5



รูปที่ 3.5 แสดงลักษณะโครงสร้างของ MAC Servo Motor

สำหรับเซอร์โวมอเตอร์ที่ใช้ในการทดลองเป็นเซอร์โวมอเตอร์ของ Indramat รุ่น MAC 63

3.2.4.1 การใช้งาน

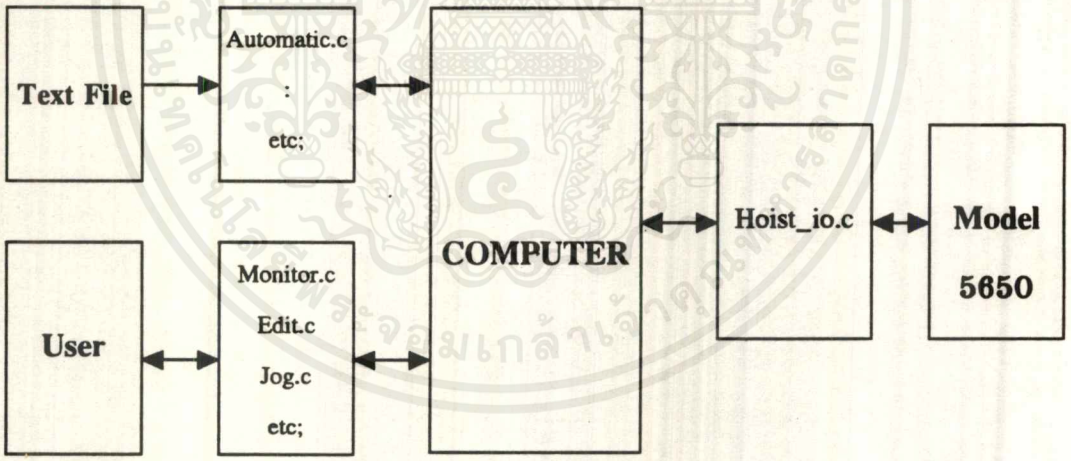
MAC Servo Motor จะต้องเชื่อมต่อกับอุปกรณ์ 2 ตัว คือ SDM และการ์ดควบคุม Model 5650 ซึ่งการเชื่อมต่อกับอุปกรณ์แต่ละตัวนั้นได้กล่าวมาแล้วข้างต้น

บทที่ 4
การออกแบบและวิธีการใช้โปรแกรม

ในบทนี้จะกล่าวถึงการออกแบบ วิธีการใช้โปรแกรม และแนวทางการพัฒนาโปรแกรมที่เขียนขึ้นมา เพื่อใช้สำหรับควบบคุมการทำงานของต้นแบบเครื่องกัดแนวดิ่งซีเอ็นซีที่มีอยู่ โดยใช้ภาษา C ในการเขียน และคอมไพล์บน TURBO C Version 2.0 ซึ่งตัวโปรแกรมจะแบ่งเป็นส่วนสำคัญๆ 3 ส่วนดังต่อไปนี้

1. ส่วนที่ติดต่อระหว่างผู้ใช้กับโปรแกรม (User Interface) ได้แก่ โมดูล Monitor.C โมดูล Edit.C โมดูล Jog.C
2. ส่วนที่เชื่อมต่อระหว่างคอมพิวเตอร์กับการ์ดควบคุม ได้แก่ โมดูล Host_io.C
3. ส่วนการติดต่อสื่อสารกับข้อมูลภายนอกที่อยู่ในรูปของ Text File ได้แก่ โมดูล Automatic.C

นอกจากนี้ก็เป็นส่วนประกอบปลีกย่อยซึ่งช่วยในการทำงานของโปรแกรมโดยทั้งหมดทุกส่วนที่กล่าวมาสามารถแสดงดังในรูปที่ 4.1



รูปที่ 4.1 แสดงส่วนการทำงานของโปรแกรม

4.1 การออกแบบ

ได้ทำการออกแบบโปรแกรมในแต่ละส่วนดังที่ได้กล่าวมาแล้ว โดยตั้งอยู่บนข้อกำหนดดังต่อไปนี้

- ส่วนที่ติดต่อระหว่างผู้ใช้กับโปรแกรม

ประกอบด้วย โหมดการทำงานต่างๆ ดังนี้

4.1.1. โหมดการเขียนและแก้ไขโปรแกรม (Edit mode)

4.1.2. โหมดอัตโนมัติ (Automatic mode)

4.1.3. โหมดความจำ (Memory mode)

4.1.4. โหมดแบบช่วง (Jog mode)

โดยแต่ละโหมดดังกล่าวข้างต้น มีรายละเอียดของฟังก์ชันต่างๆ ดังนี้

4.1.1. <u>โหมดการเขียนและแก้ไขโปรแกรม (Edit mode)</u>

ในโหมดดังกล่าวเป็นส่วนของการเขียนโปรแกรมเพื่อควบคุมเครื่องกัดแนวดิ่งซีเอ็นซีให้ทำงานตามที่ต้องการ โดยการเขียนโปรแกรม ต้องอ้างอิงกับ G-code และ M-code มาตรฐาน ซึ่งมีความสามารถในการทำงานดังนี้

- สามารถเขียนข้อมูล (Edit) โดยในระหว่างการเขียนข้อมูลจะมีการเขียนเลขบรรทัด และรูปแบบคำสั่งของ G-code หรือ M-code ที่เลือกใช้ให้ เพื่อสะดวกต่อการเขียนข้อมูล
- สามารถลบข้อมูล (Delete) ในตำแหน่งบรรทัดที่ต้องการได้ โดยการเลื่อนตัวชี้ขึ้นหรือลงเพื่อเลือกบรรทัดที่ต้องการ
- สามารถแทรกข้อมูล (Insert) ในตำแหน่งบรรทัดที่ต้องการเพิ่มข้อมูลเข้าไปได้
- สามารถเลื่อนตัวชี้ขึ้นไปหนึ่งบรรทัด (Arrow up) เพื่อใช้ในการแก้ไขหรือตรวจสอบโปรแกรม
- สามารถเลื่อนตัวชี้ลงไปหนึ่งบรรทัด (Arrow down) เพื่อใช้ในการแก้ไขหรือตรวจสอบโปรแกรม
- สามารถเก็บข้อมูลที่เขียน ลงบนแผ่นดิสค์เกตต์ได้

โปรแกรมสำหรับโหมดนี้จะอยู่ในโมดูล Edit.C ทั้งหมด ซึ่งมีผังการทำงาน (Flow Chart) ดังรูปที่ 4.2

4.1.2. <u>โหมดอัตโนมัติ (Automatic mode)</u>

ในโหมดดังกล่าวเป็นส่วนของการอ่านข้อมูล การแปลข้อมูล และการทำงานตามข้อมูลที่ได้รับ โดยโปรแกรมจะทำการอ่านข้อมูลที่มาในรูปเทกซ์ไฟล์ทีละบรรทัด แปลความหมายและปฏิบัติตามคำสั่งทีละบรรทัดจนกว่าจะทำเสร็จ และจะวนรอบการทำงานเช่นนี้ไปเรื่อยๆ จนจบไฟล์ข้อมูล ซึ่งในโหมดนี้จะมีความสามารถในการทำงานดังนี้

- สั่งให้โปรแกรมทำงาน (Cycle start) โปรแกรมจะทำงานทั้งหมดและติดต่อกับฮาร์ดแวร์เพื่อควบคุมการทำงานให้สอดคล้องกับโปรแกรม

- สั่งให้โปรแกรมหยุดทำงาน (Cycle stop) เพื่อการหยุดโปรแกรมและส่งสัญญาณไปบอกระบบฮาร์ดแวร์ในการควบคุมมอเตอร์หยุดการทำงาน ถ้าจะเริ่มการทำงานใหม่ก็จะต้องโหลดโปรแกรมเข้าไปใหม่
- สั่งให้โปรแกรมทำงานทีละ Block โปรแกรม(Singleblock) ในโหมดดังกล่าวโปรแกรมจะกำหนดให้ฮาร์ดแวร์มีการสั่งการทำงานทีละบรรทัดใน Block ที่กำหนดไว้ตั้งแต่ต้นจนจบ

โปรแกรมสำหรับโหมดนี้ได้เขียนขึ้นบางส่วนเฉพาะในส่วนของ Cycle start ซึ่งอยู่ในโมดูล Automatic.C ทั้งหมด ซึ่งมีผังการทำงาน (Flow Chart) ดังรูปที่ 4.3

4.1.3. <u>โหมดความจำ (Memory mode)</u>

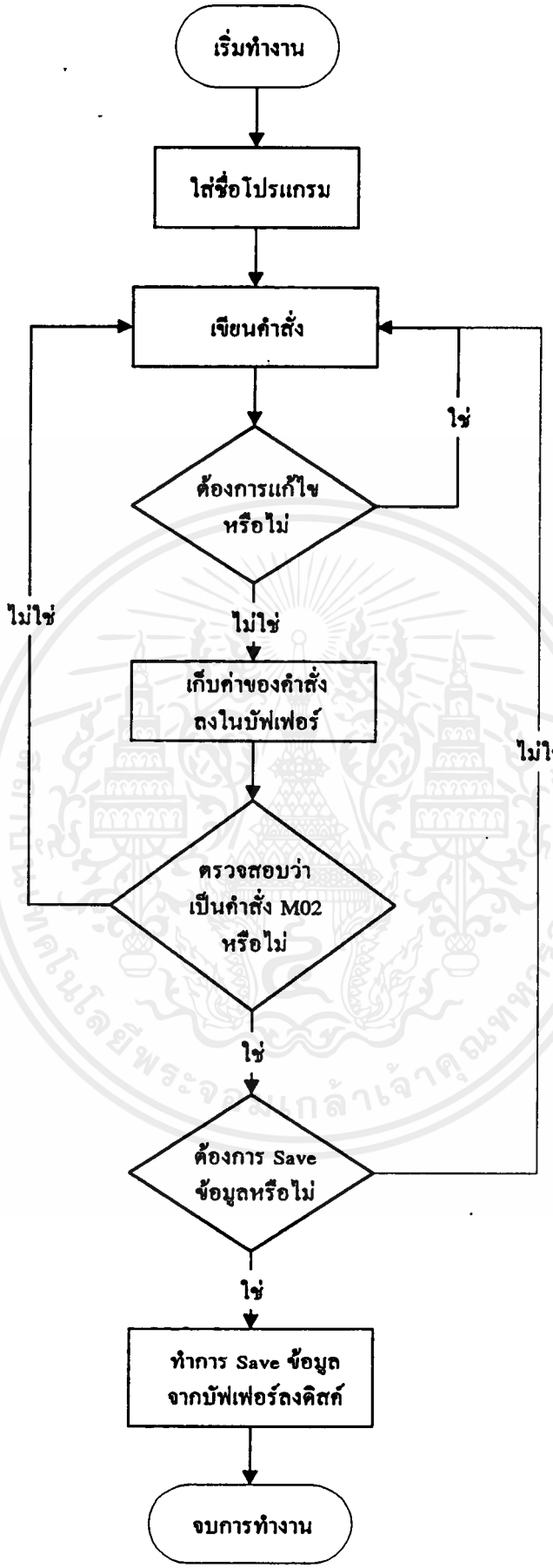
ในโหมดดังกล่าวเป็นส่วนของการปรับเปลี่ยนค่าของ Feed rate , Spindle rate และ Jog speed ในช่วงระหว่างที่โปรแกรมยังทำงานอยู่โดยผู้ใช้เป็นผู้กำหนดเอง โปรแกรมสำหรับโหมดนี้ยังไม่ได้เขียนขึ้น แต่สามารถเขียนเพิ่มเติมได้โดยการเขียนโมดูล Memory.C ขึ้นมา แล้วใช้ฟังก์ชันในโมดูล Monitor.C เป็นตัวเรียกใช้งาน ซึ่งผังการทำงาน (Flow Chart) ของโหมดนี้มีลักษณะดังรูปที่ 4.4

4.1.4. <u>โหมดแบบช่วง (Jog mode)</u>

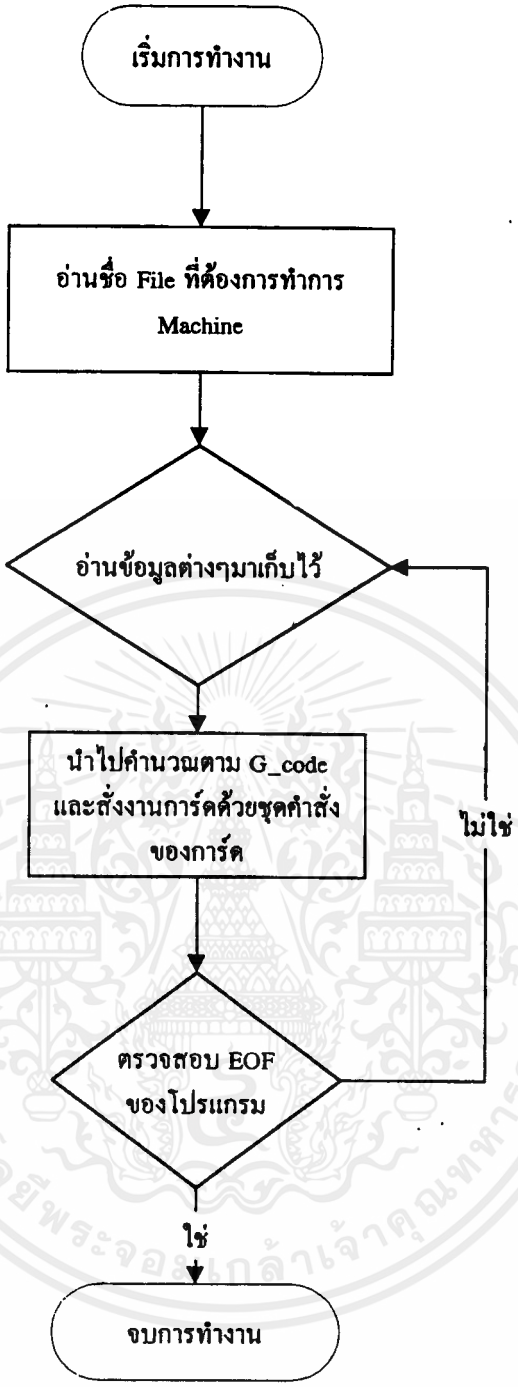
ในโหมดดังกล่าวจะเป็นการเคลื่อนที่ของเครื่องจักรในแบบธรรมดา โดยอาศัยการกดปุ่ม (Keyboard) และสามารถสั่งการเคลื่อนที่ได้ทีละแกน ในโหมดนี้จะมีความสามารถต่างๆ ดังนี้

- การเคลื่อนที่ทีละแกน (Jog) จะสามารถควบคุมมอเตอร์ได้ทีละ 1 แกน ด้วยความเร็วที่ได้กำหนดไว้ในโหมดความจำ (Memory mode)
- กลับสู่ตำแหน่งเริ่มต้น (Home) สั่งให้มอเตอร์ทั้ง 3 แกนเคลื่อนที่กลับไปที่ตำแหน่ง Home
- เซ็ตศูนย์โปรแกรม (Set Zero) ทำการเซ็ตค่าศูนย์โปรแกรม
- หยุดฉุกเฉิน (Emergency Stop) เป็นการหยุดการทำงานของเครื่องในกรณีฉุกเฉิน

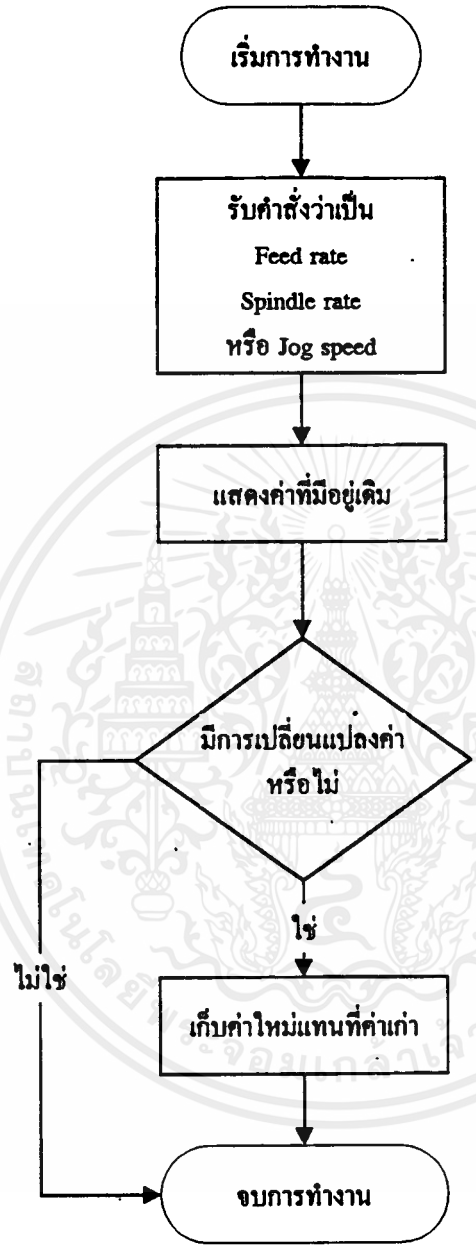
โปรแกรมสำหรับโหมดนี้ได้เขียนขึ้นบางส่วนเฉพาะในส่วนของการเคลื่อนที่ทีละแกน (Jog) ซึ่งเขียนไว้ในโมดูล Jog.C สามารถควบคุมได้เฉพาะแกน X และ Y เท่านั้น ยังขาดแกน Z ซึ่งจะต้องมีการเขียนเพิ่มเติมลงไป สำหรับโมดูล Jog.C นั้น มีผังการทำงาน (Flow Chart) ดังรูปที่ 4.5



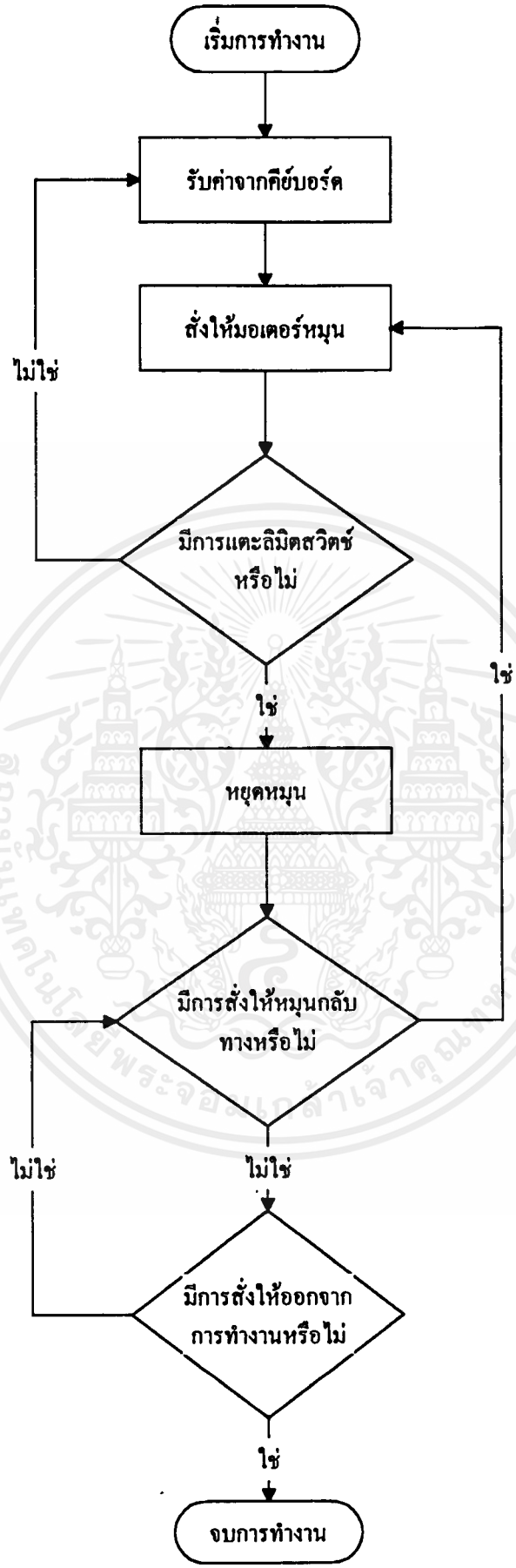
รูปที่ 4.2 ผังการทำงานของ Edit.C



รูปที่ 4.3 แสดงผังการทำงานของ Automatic.C



รูปที่ 4.4 แสดงผังการทำงานของ Memory.C

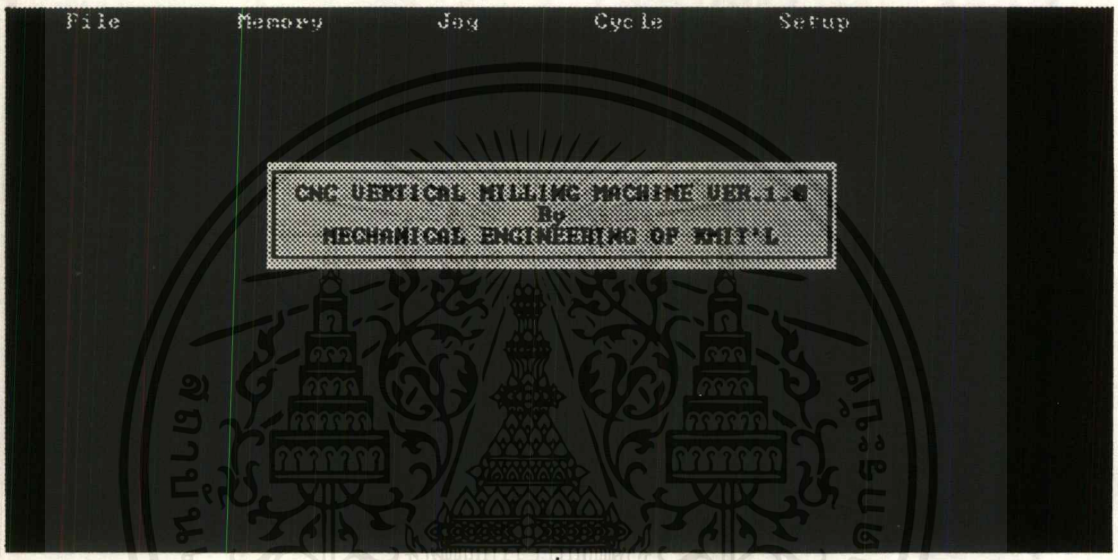


รูปที่ 4.5 แสดงผังการทำงานของ Jog.C

4.2 วิธีการใช้โปรแกรม

ต่อไปนี้จะกล่าวถึงการใช้โปรแกรม CNC VERTICAL MILLING MACHINE VER.1.0 ซึ่งเป็นโปรแกรมที่ใช้ในการควบคุมเครื่องกัดแนวดิ่งซีเอ็นซี ดังมีขั้นตอนดังต่อไปนี้

1. นำแผ่น source code ที่แนบมากับปริญญานิพนธ์เล่มนี้ใส่ใน Drive A หรือ B
2. พิมพ์ชื่อโปรแกรม CNC แล้วกด Enter
3. เมื่อเครื่องโหลดโปรแกรม CNC ขึ้นแล้ว จะมีหน้าต่างแสดงชื่อโปรแกรมปรากฏที่กลางจอภาพ ดังรูปที่ 4.6



รูปที่ 4.6

4. กด Enter เพื่อเข้าสู่ Menu
5. เลือก Menu หลัก โดยใช้คีย์ลูกศร เลื่อนแถบสว่างไปยัง Menu หลักที่ต้องการ

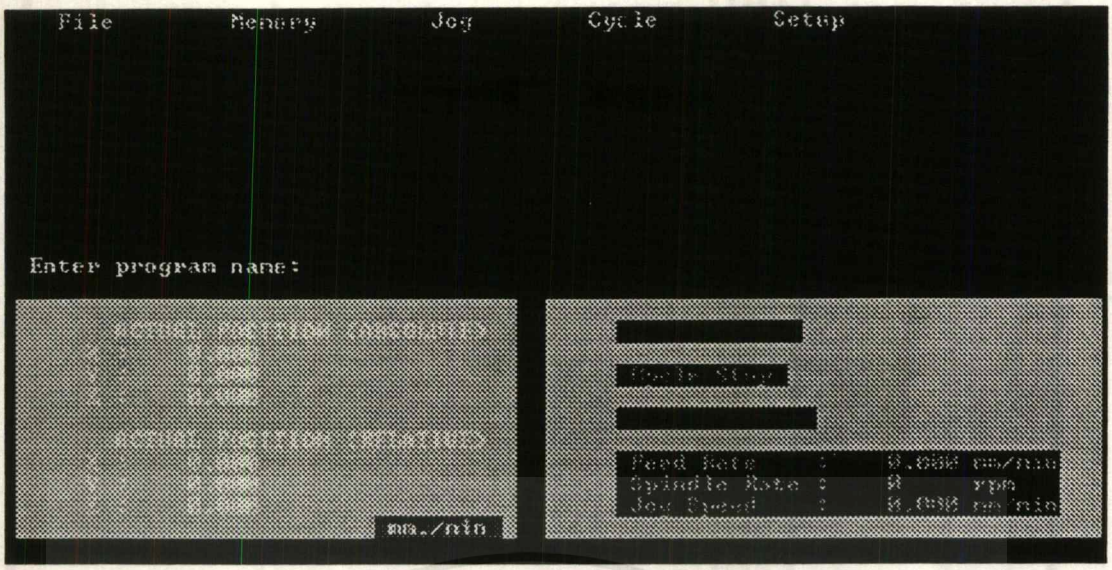
จะเห็นได้ว่า Menu หลัก จะประกอบด้วย 5 ส่วน คือ File , Memory , Jog , Cycle และ Set up โดยแต่ละ Menu หลัก ประกอบด้วย Menu ย่อยๆ ซึ่งมีวิธีการใช้งานดังนี้

4.2.1. File

ประกอบด้วย Menu ย่อย 3 ส่วน คือ

1) Load Program

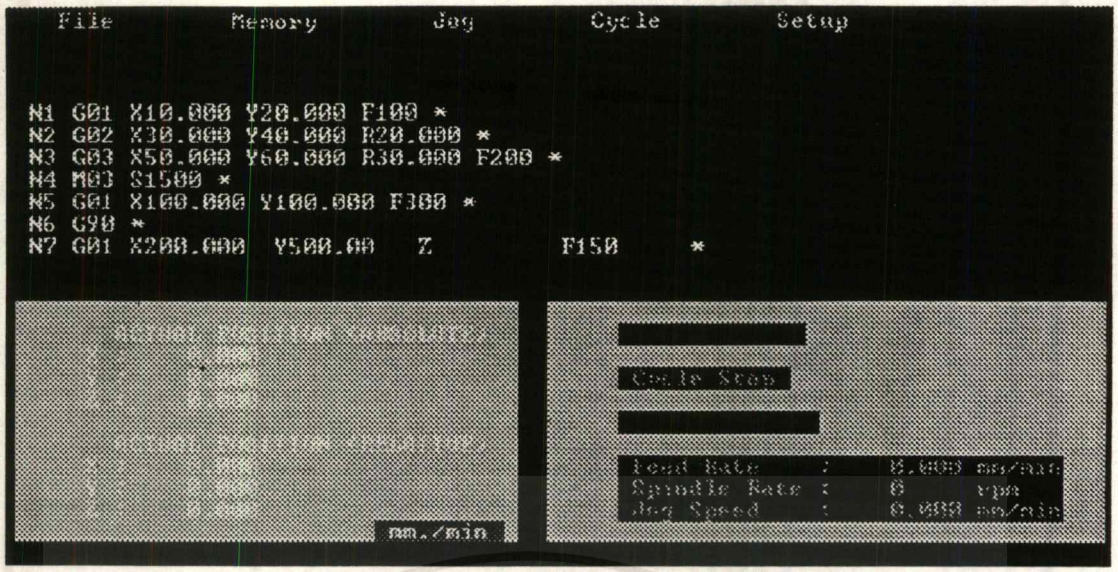
เลื่อนแถบสว่างไปยัง Load Program แล้วกด Enter จะปรากฏข้อความให้ใส่ชื่อโปรแกรมที่ต้องการโหลด ดังรูปที่ 4.7 ซึ่งจะต้องเป็นโปรแกรมที่อยู่ในรูปเท็กซ์ไฟล์เท่านั้น และต้องมีชื่ออยู่ในไดเรกเตอรีเดียวกันกับโปรแกรม CNC.EXE ด้วย แล้วกด Enter จอภาพจะแสดงชื่อโปรแกรมที่โหลดขึ้นมาแต่จะไม่มีรายละเอียดของข้อมูลในโปรแกรมให้เห็น



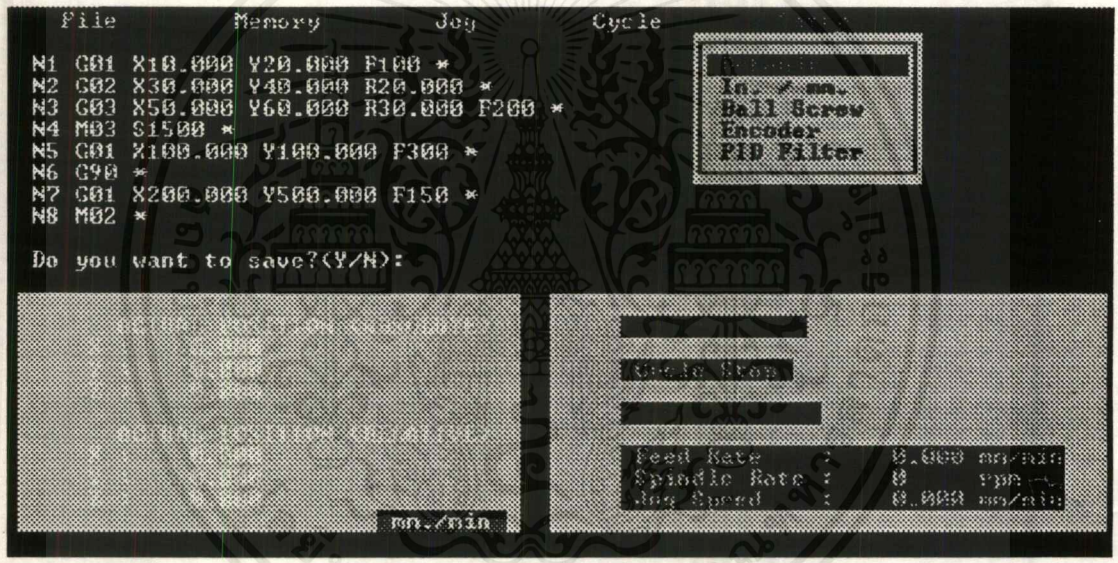
รูปที่ 4.7

2) Edit Program

เลื่อนแถบสว่างไปยัง Edit Program แล้วกด Enter จะปรากฏข้อความให้ใส่ชื่อโปรแกรมที่ต้องการจะเขียน ดังรูปที่ 4.7 โดยชื่อโปรแกรมที่จะใส่ต้องไม่ซ้ำกับชื่อโปรแกรมเดิมที่เคยมีอยู่ มิฉะนั้น โปรแกรมเดิมจะหายไป จากนั้นกด Enter หน้าจอจะเข้าสู่ Edit mode โดยการเขียนโปรแกรมแต่ละบรรทัด จะมีการขึ้นเลขบรรทัดให้โดยอัตโนมัติ ส่วน G-code และ M-code ที่จะเขียน จะมีรูปแบบของคำสั่งแต่ละตัวให้ เพื่อป้องกันการเขียนคำสั่งผิดรูปแบบมาตรฐาน การใส่ค่าตัวเลขจะต้องมีการกด Enter ตามทุกครั้ง ถ้าไม่ใส่ให้กด Enter ผ่าน เมื่อจบบรรทัดจะมีเครื่องหมาย * ปิดท้าย ดังรูปที่ 4.8 หลังจากนั้นถ้าต้องการยืนยันว่าบรรทัดที่เพิ่งเขียนมาถูกต้องให้กด Enter ถ้าต้องการแก้ไขใหม่ให้กด Del จากนี้จะสามารถใช้คีย์ลูกศรเลื่อนดูโปรแกรมในลักษณะขึ้นลงได้ หากต้องการแก้ไขบรรทัดใด หรือแทรกบรรทัดใดก็สามารถทำได้ด้วยการกด Del และ Insert ตามลำดับ และดำเนินวิธีการเขียนคำสั่งดังที่ได้กล่าวมาแล้วข้างต้น หากกด Enter ในขณะที่อยู่ในช่วงการเลื่อนโปรแกรมขึ้นลงไม่ว่าตัวชี้จะอยู่ที่บรรทัดใดก็ตาม จะมีการขึ้นบรรทัดใหม่เพื่อการเขียนคำสั่งทันที การออกจาก Edit mode จะต้องเขียนบรรทัดสุดท้ายด้วยคำสั่ง M02 แล้วจะปรากฏข้อความให้ทำการ Save ข้อมูล ดังรูปที่ 4.9



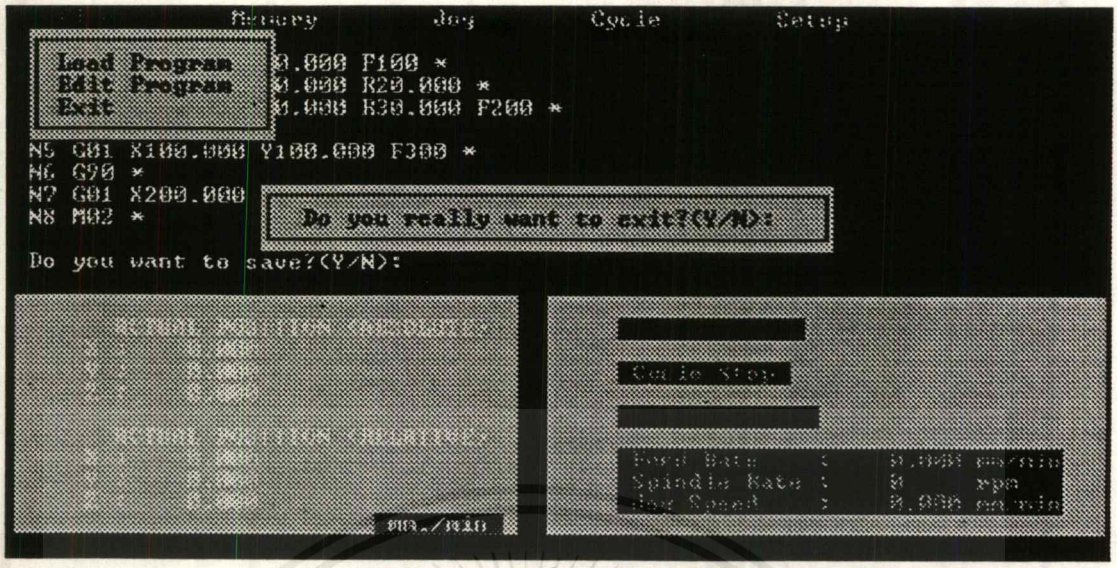
รูปที่4.8



รูปที่4.9

3) Exit

เลื่อนแถบสว่างไปยัง Exit แล้วกด Enter จะปรากฏข้อคามดังรูปที่ 4.10 ถ้ากด Y ก็จะออกจะออกจากโปรแกรม CNC และ กลับไปบนระบบปฏิบัติการอีกครั้ง ถ้ากด N ก็จะกลับเข้าสู่การใช้งานโปรแกรม CNC ตามเดิม



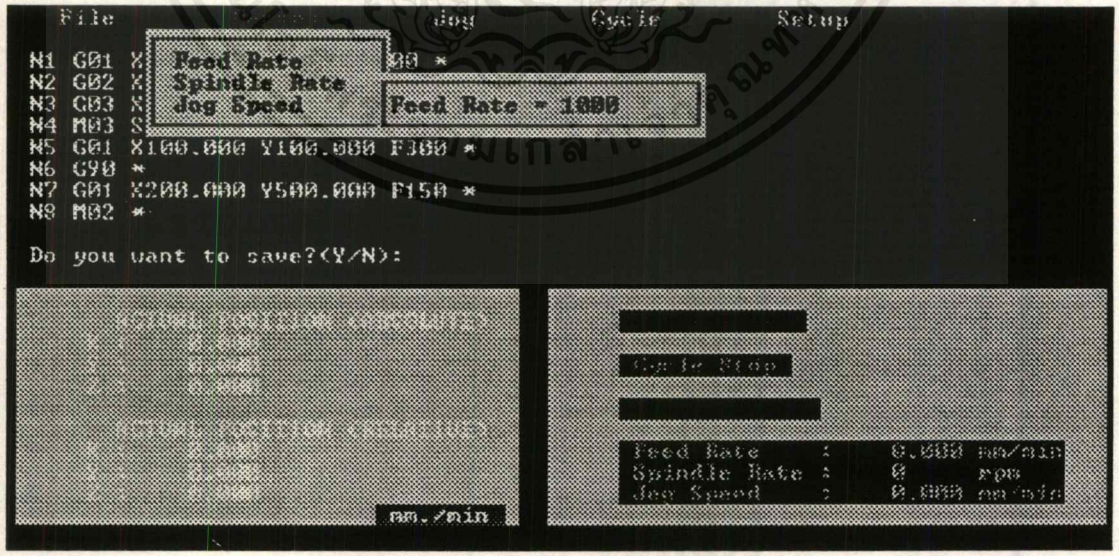
รูปที่ 4.10

4.2.2 Memory

ประกอบด้วย Menu ย่อยอีก 3 ส่วนคือ

1) Feed rate

เมื่อมีการเลือก Menu นี้จะปรากฏข้อความให้ใส่ค่าของ Feed rate ใหม่ ซึ่งจะมีค่า Feed rate เดิมแสดงให้หืนด้วย ถ้าไม่ต้องการเปลี่ยนให้กด Enter 2 ครั้ง ถ้าต้องการเปลี่ยนให้กด Enter แล้วพิมพ์ค่าที่ต้องการ ตามด้วยการกด Enter อีก 1 ครั้ง ดังรูปที่ 4.11 โดยค่าของ Feed rate ที่เปลี่ยนใหม่นี้จะมีผลต่อทุกโหมดในโปรแกรม CNC



รูปที่ 4.11

2) Spindle speed

การใช้งานและหน้าที่จะเหมือนกับการใช้งานของ Feed rate

3) Jog speed

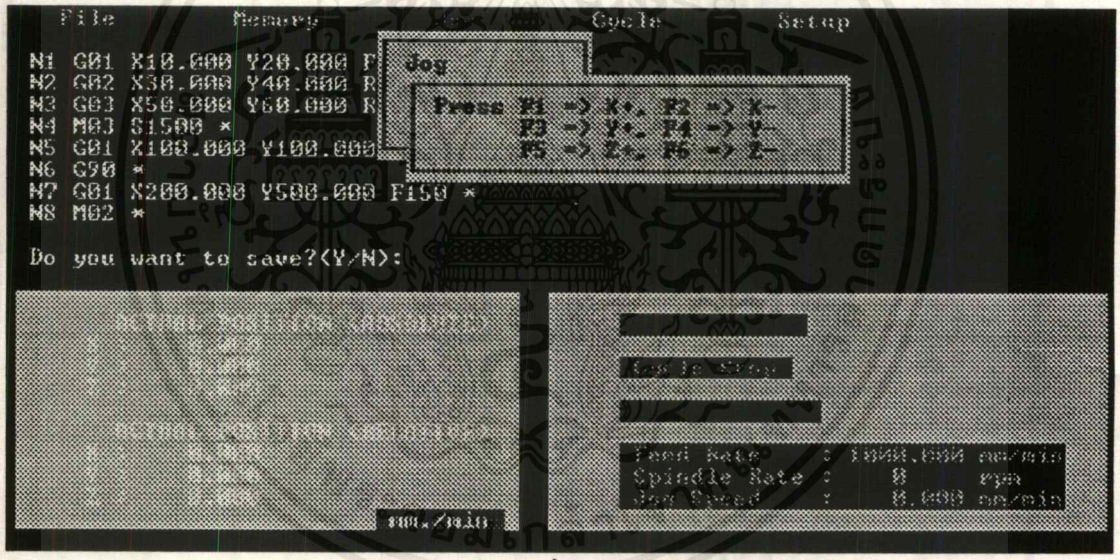
การใช้งานและหน้าที่จะเหมือนกับการใช้งานของ Feed rate

4.2.3 Jog

ประกอบด้วย Menu ย่อย 4 ส่วนคือ

1) Jog

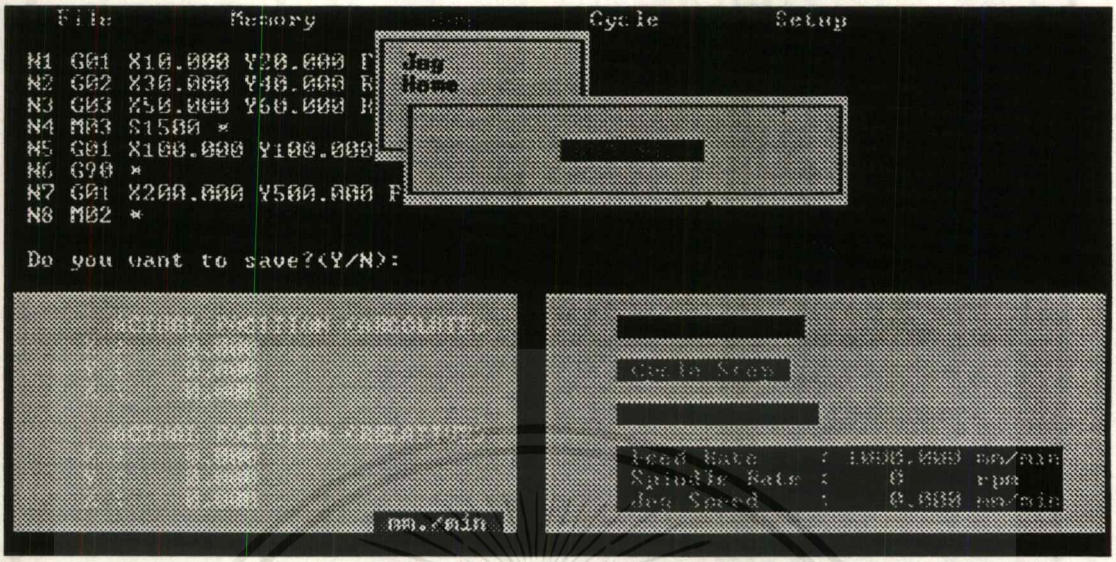
เมื่อมีการเลือก Menu นี้ จะปรากฏข้อความถึงปุ่ม (Keyboard) ที่จะใช้ควบคุม การเคลื่อนที่ของ มอเตอร์ให้ไปในทิศทางที่เราต้องการ ดังรูปที่ 4.12 การควบคุมสามารถกระทำได้โดยการกดปุ่มโดยตรง



รูปที่ 4.12

2) Home

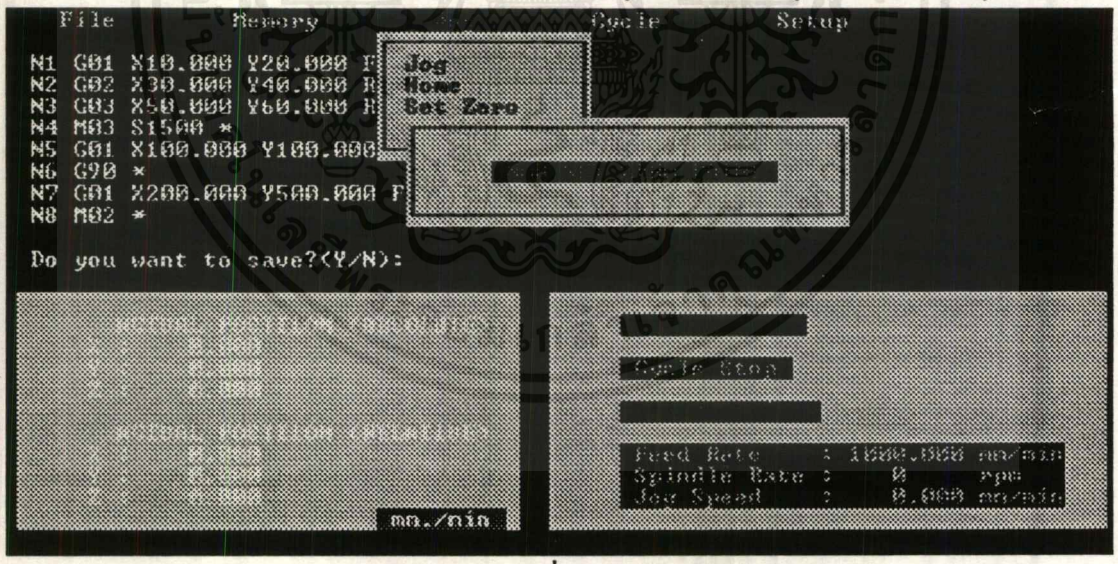
เมื่อมีการเลือก Menu นี้จะปรากฏข้อความบนจอภาพ ดังรูปที่ 4.13 และในแต่ละแกนจะมีการเคลื่อนที่กลับไปยังตำแหน่ง Home ถ้าต้องการหยุดให้เลือก Menu Emergincy Stop



รูปที่ 4.13

3) Set Zero

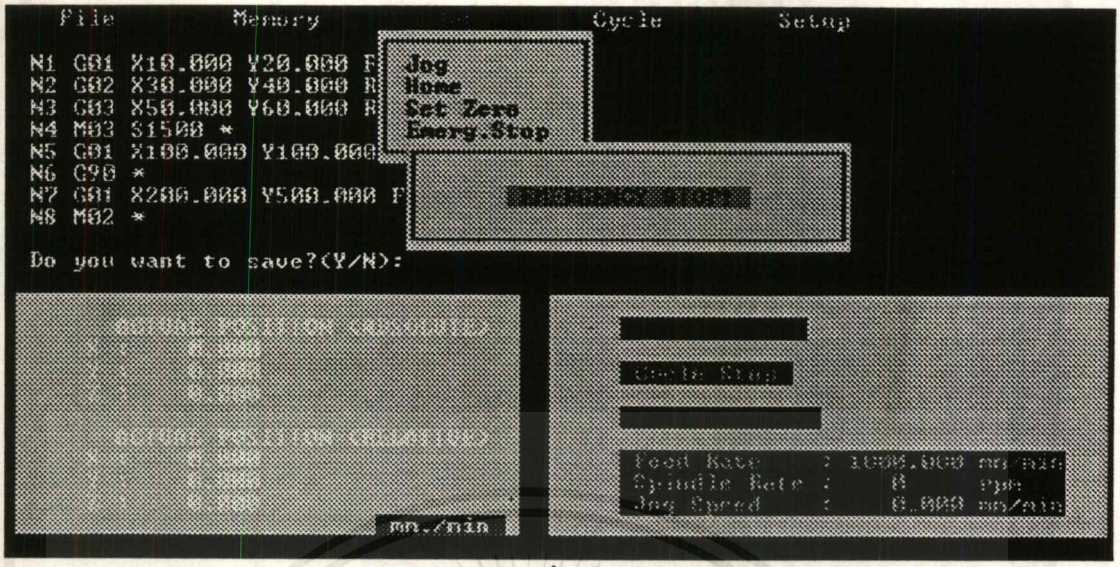
เมื่อมีการเลือก Menu นี้จะปรากฏข้อความบนจอภาพ และค่าตำแหน่งที่แสดงบนจอภาพ (Actual Position <Relative>) จะถูกรีเซ็ตให้เป็นศูนย์ทั้งหมด ดังรูปที่ 4.14



รูปที่ 4.14

4) Emergency Stop

เมื่อมีการเลือก Menu นี้ จะปรากฏข้อความบนจอภาพ ดังรูปที่ 4.15 และเครื่องจักรจะหยุดการทำงาน



รูปที่ 4.15

สำหรับทุกๆ Menu ย่อย ดังที่ได้กล่าวมาแล้วนี้ การที่จะยกเลิกการทำงานหรือออกจาก Menu ย่อยนั้นๆ สามารถทำได้โดยการกดปุ่ม ESC นอกจากจะได้กำหนดไว้เป็นอย่างอื่น

4.2.4. Cycle

ประกอบด้วย Menu ย่อย 3 ส่วน คือ

1) Cycle Start

เลื่อนแถบสว่างไปยัง Cycle Start แล้วกด Enter จะปรากฏข้อความกระพริบ "Cycle Start" ขึ้นบนหน้าจอ แล้วเครื่องจักรจะทำตามโปรแกรมที่ได้โหลดขึ้นมาโดยคำสั่ง LOAD Program ก่อนหน้านี้

2) Cycle Stop

เลื่อนแถบสว่างไปยัง Cycle Stop แล้วกด Enter จะปรากฏแถบสว่างบนข้อความ "Cycle Stop" แล้วเครื่องจักรจะหยุดทำงาน ถ้าต้องการเริ่มการทำงานใหม่ให้เลือก Menu Cycle Start

3) Single Block

เลื่อนแถบสว่างไปยัง Single Block แล้วกด Enter จะปรากฏข้อความกระพริบ "Single Block" บนหน้าจอ แล้วเครื่องจักรจะทำตาม Block ของโปรแกรมที่ได้กำหนดไว้

4.2.5. Setup

ประกอบด้วย Menu ย่อย 5 ส่วน คือ

1) Default

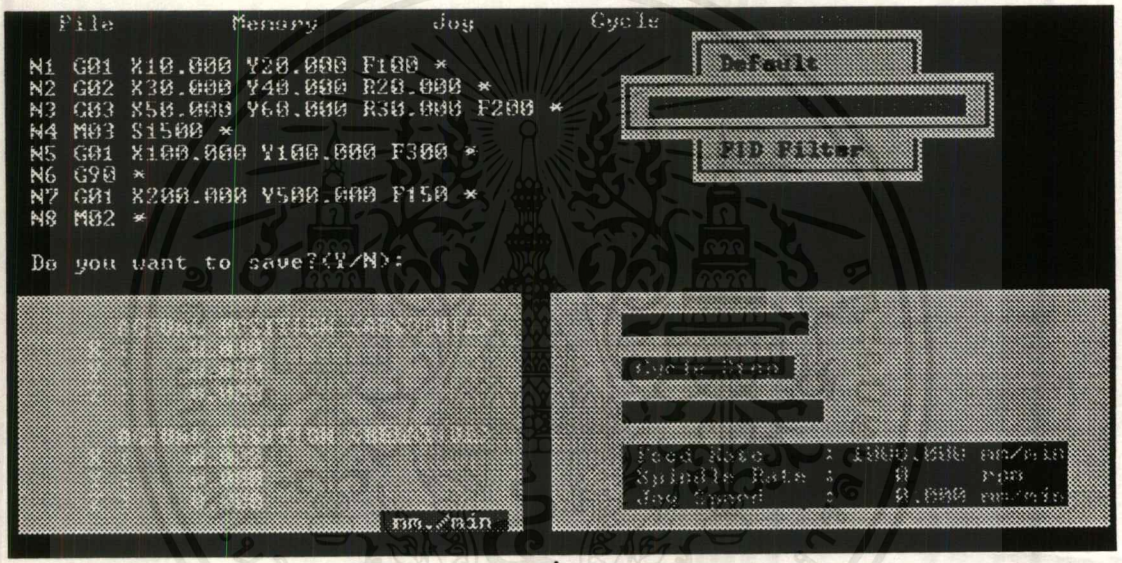
เมื่อ Menu นี้ถูกเลือก จะปรากฏข้อความดังรูปที่ 4.16 และจะมีการเซ็ตค่าพารามิเตอร์ต่างๆ ดังต่อไปนี้

1.1 ค่า PID Digital Filter

1.2 ค่า pulse / rev. ของ Encoder ที่ใช้

1.3 ค่า Pitch ของ Ball Screw ที่ใช้

ให้เป็นค่าดั้งเดิม ซึ่งเป็นค่าเฉพาะของเครื่องต้นแบบนี้เท่านั้น



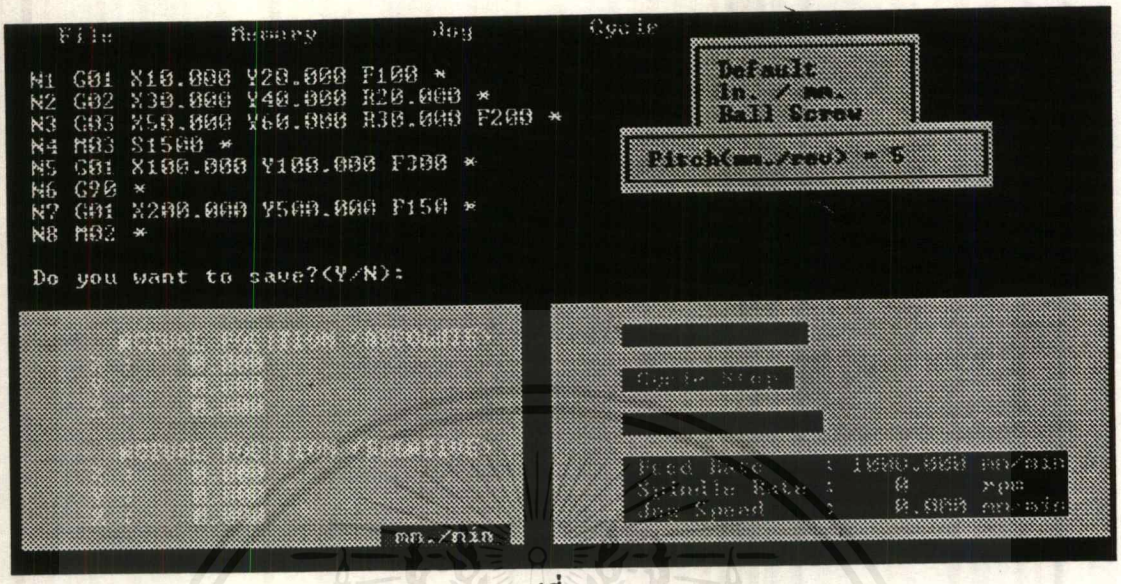
รูปที่ 4.16

2) In. / mm.

เมื่อ Menu นี้ถูกเลือก ค่าของตำแหน่ง (ทั้ง Abs. และ Rel.) , Feed Rate และ Jog Speed จะถูกเปลี่ยนให้อยู่ในหน่วยของ มิลลิเมตร หรือ นิ้ว ได้ตามต้องการ และจะมีผลต่อหน่วยของข้อมูลที่อยู่ในโปรแกรมที่ได้ทำการโหลดไว้ก่อนหน้านี้ด้วย

3) Ball Screw

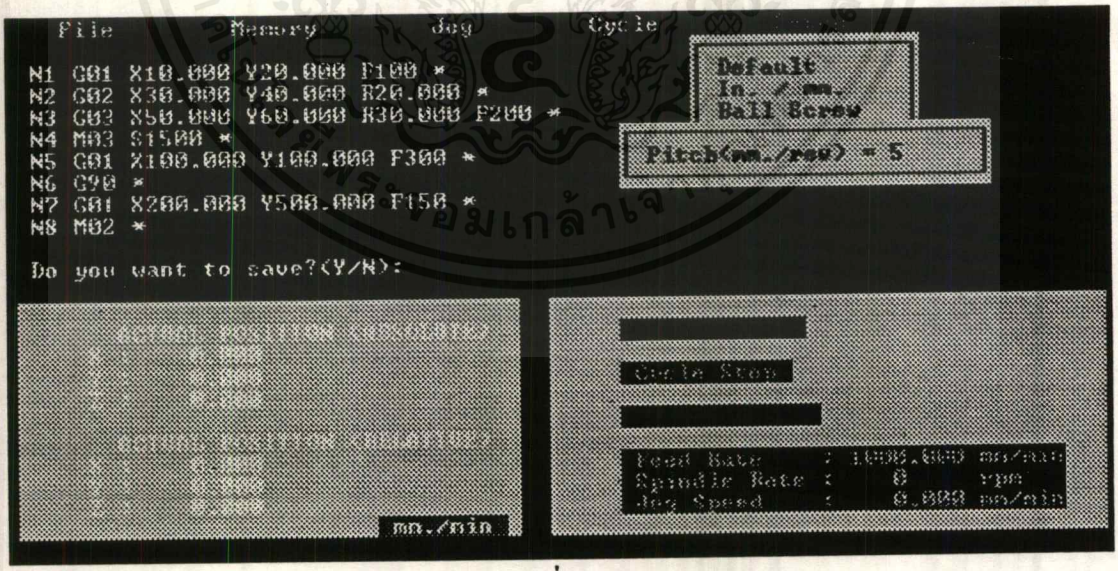
เมื่อ Menu นี้ถูกเลือก จะปรากฏข้อความให้ใส่ค่า Pitch ในหน่วย มิลลิเมตร ของ Ball Screw ลงไปโดยวิธีการเปลี่ยนค่าจะคล้ายกับFeedRateดังรูปที่4.17



รูปที่ 4.17

4) Encoder

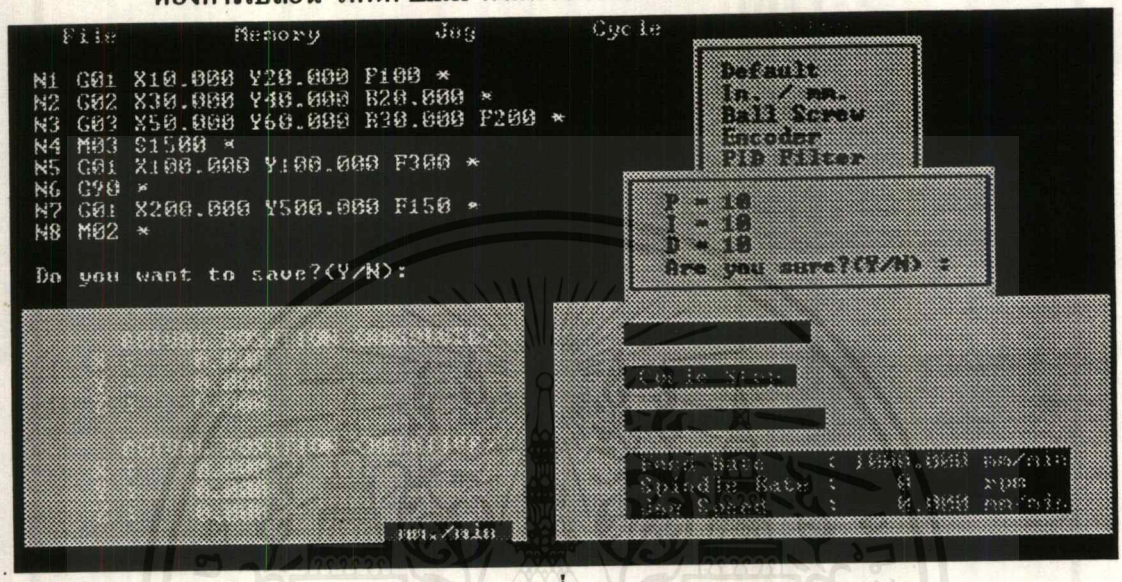
เมื่อ Menu นี้ถูกเลือก จะปรากฏข้อความให้ใส่ค่า pulse/rev. ของ Encoder ลงไป โดยวิธีการเปลี่ยนค่าจะคล้ายกับ Feed Rate ดังรูปที่ 4.18



รูปที่ 4.18

5) PID

เปลี่ยน แล้วกด Enter ทำเช่นนี้ไปเรื่อยๆ จนครบทั้ง 3 ค่า ก็จะปรากฏข้อความยืนยันการเปลี่ยนแปลง ถ้ากด Y หมายถึง ถูกต้อง ถ้ากด N หมายถึง ไม่ถูกต้อง ดังรูปที่ 4.19เมื่อ Menu นี้ถูกเลือก จะปรากฏข้อความให้ใส่ค่าพารามิเตอร์ P , I และ D โดยจะแสดงค่าเดิมที่ใช้อยู่ด้วย หากไม่ต้องการเปลี่ยนค่าให้กด Enter 2 ครั้ง หากต้องการเปลี่ยน ให้กด Enter ตามด้วยค่าที่ต้องการ



รูปที่ 4.19

บทที่ 5
ผลการทดลอง

จากผลการทดลองซึ่งได้ทำการทดลองให้ทำงานกับฟังก์ชัน G-Code 3 ฟังก์ชัน ได้แก่

1. G01
2. G02
3. G03

โดยใช้ค่าพารามีเตอร์ต่างๆในดิจิตอลฟิลเตอร์ดังนี้

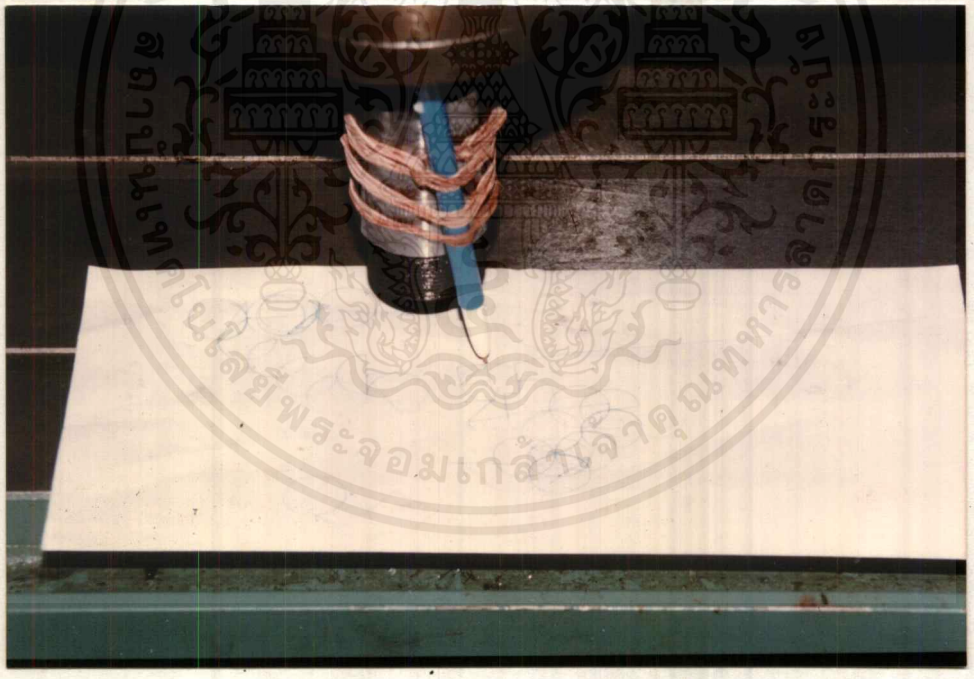
P = 10

I = 10

D = 10

ระยะ Chord = 0.1 mm.

สามารถทำงานได้ผลดังรูป 5.1



รูปที่ 5.1 ผลที่ได้จากการทดลองการทดลอง

บทที่ 6
สรุปและข้อเสนอแนะ

ปริญญานิพนธ์ฉบับนี้กล่าวถึง การสร้างโปรแกรมเพื่อใช้ในการควบคุมต้นแบบของเครื่องกัดแนวดิ่งซีเอ็นซี โดยได้มีการศึกษาการใช้งานในหลายๆ ส่วนดังนี้ประกอบกัน เช่น ส่วนการ์ดควบคุมสำเร็จรูป MODEL 5650 ส่วนฮาร์ดแวร์ ระบบควบคุมเซอร์โวมอเตอร์ และในส่วนภาษาที่จะใช้เขียนโปรแกรม รวมทั้งทฤษฎีทางซีเอ็นซีต่างๆ เพื่อเป็นพื้นฐานที่จะนำมาใช้ในการสร้างและพัฒนาโปรแกรมควบคุมดังกล่าว ซึ่งจากผลการทดลองที่ได้รับ ก็นับได้ว่าประสบความสำเร็จในขั้นหนึ่ง แต่ถึงกระนั้นก็ยังมีข้อบกพร่องและปัญหาที่จะต้องทำการแก้ไขและพัฒนาอีกต่อไปดังจะได้กล่าวในหัวข้อต่อไปนี้

6.1. ด้านส่วนควบคุม (การ์ดควบคุม MODEL 5650)

ในส่วนของการ์ดควบคุมที่ใช้ในการทดลองครั้งนี้นั้น จะประสบปัญหาเกี่ยวกับการควบคุมมอเตอร์ หลังจากที่โต๊ะงานเลื่อนไปชนกับลิมิตสวิตช์ด้านใดด้านหนึ่ง โดยตามปกติแล้วการ์ดควบคุมจะสั่งให้มอเตอร์หยุดหมุนในทิศทางนั้น และอนุญาตให้สามารถสั่งให้มอเตอร์หมุนไปในทิศทางตรงกันข้ามได้ แต่จากการทดลอง ในช่วงที่โต๊ะงานชนลิมิตสวิตช์ค้างอยู่นั้น กลับพบว่ามอเตอร์ไม่ได้หยุดนิ่งจริง ซึ่งเป็นผลให้ประสบความยุ่งยากอย่างมากในการควบคุม โดยเหตุการณ์อย่างที่พบนี้จะเกิดกับลิมิตสวิตช์ด้านใดด้านหนึ่งของแต่ละแกนควบคุมเท่านั้น

6.2. ด้านโปรแกรมควบคุม

ในส่วนของโปรแกรมควบคุมที่ได้สร้างขึ้นนั้นจะต้องทำการปรับปรุงในรายละเอียดด้านต่างๆ ดังนี้

6.2.1. ฟังก์ชัน Load Program ในเมนู File ควรปรับปรุงให้สามารถโหลดตัวโปรแกรมขึ้นมาบนหน้าจอได้ ทำการแก้ไขเพิ่มเติมได้ และในช่วงการทำงานของโปรแกรมควรจะมีแถบสว่างขึ้นตรงบรรทัดที่กำลังทำงานอยู่ด้วย ซึ่งสามารถเขียนเพิ่มเติมใหม่เป็น โมดูล Load.C แล้วเรียกใช้โดยโมดูล Monitor.C

6.2.2. Edit Program ในเมนู File ควรปรับปรุงให้มีความยืดหยุ่นในการเขียนแก้ไข ลบ แทรก ข้อมูลให้มากขึ้นโดยการเพิ่มเติมฟังก์ชันการทำงานลงในโมดูล Edit.C

6.2.3. ฟังก์ชัน Jog ในเมนู Jog ควรทำการเพิ่มเติมโปรแกรมให้สามารถควบคุมการเคลื่อนที่ของแกน Z ได้โดยแก้ไขในโมดูล Jog.C

6.2.4. เขียนฟังก์ชันต่อไปนี้เพิ่มเติม โดยเรียกใช้ผ่าน Monitor.C

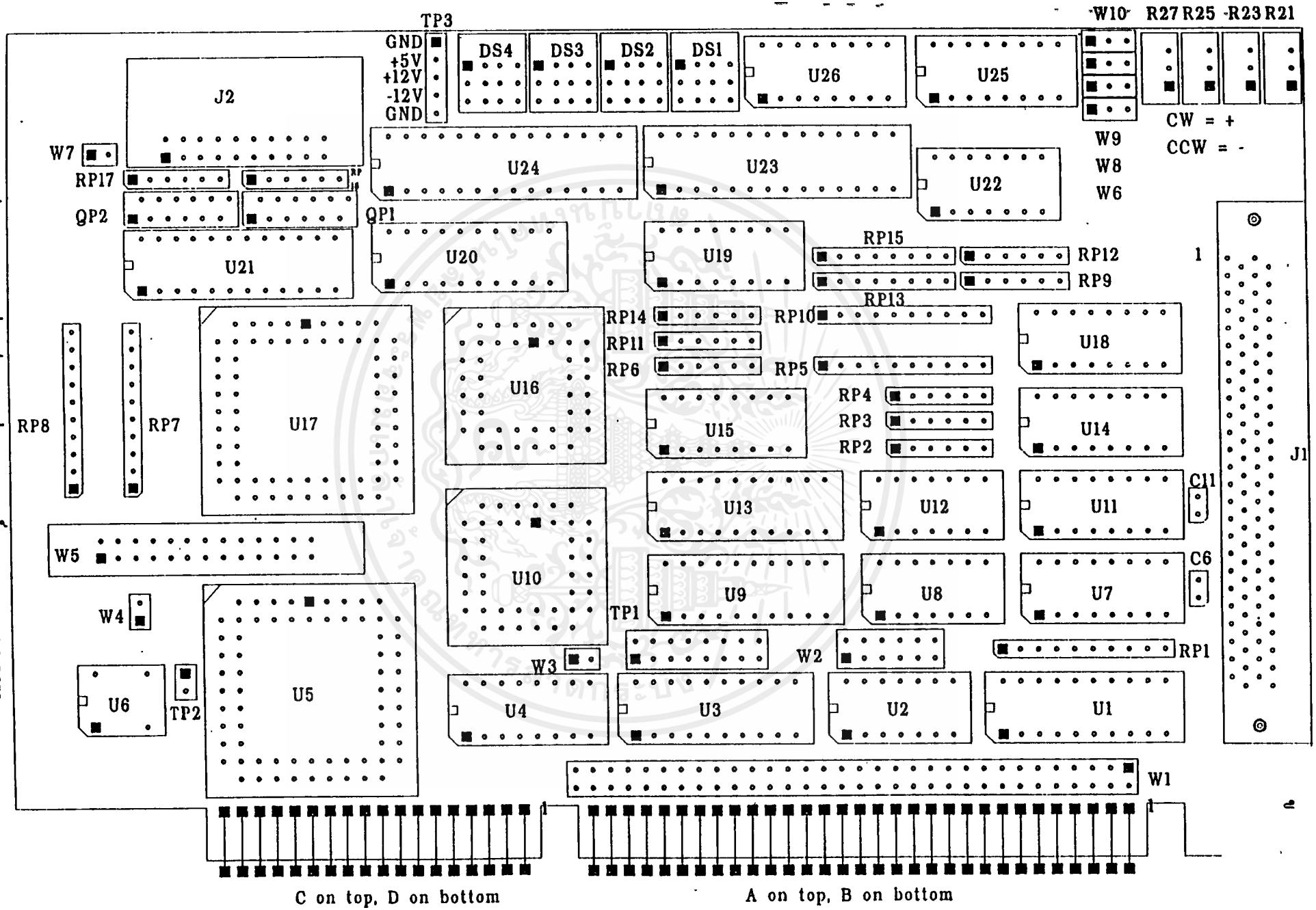
- HOME ในเมนู Jog
- SET ZERO ในเมนู Jog
- EMERG. STOP ในเมนู Jog
- SINGLE BLOCK ในเมนู Cycle
- In. / mm. ในเมนู Setup

6.2.5. ทำการทดลองฟังก์ชัน G-Code และ M-Code ทั้งหมดที่เขียนไว้ในโมดูล Automatic.C เพื่อดูการทำงานของฟังก์ชันว่ามีข้อผิดพลาดหรือไม่อย่างไร พร้อมทั้งทำการแก้ไข

เอกสารอ้างอิง

1. กวิน สนธิเพิ่มพูน , รายงานการวิจัยฉบับสมบูรณ์ เรื่อง การสร้างส่วนควบคุมเครื่องกัดแนวดิ่งซีเอ็นซี ระยะที่ 1 , ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ , 2534.
2. Larry Horath , Computer Numerical Control Programming of Machines , 1993.
3. FANUC LTD , FANUC OM-MODEL A operator's manual , FANUC LTD , 1985.
4. PMD , 5650 Board Technical Reference Version 0.2 , Technology 80 Inc. , 1994.
5. INDRAMAT , MAC Servo Drives with TDM and KDS servo drive modules , INDRAMAT , 1994.
6. ชาลี ตระการกูล , เทคโนโลยีซีเอ็นซี , สมาคมส่งเสริมเทคโนโลยี่ (ไทย-ญี่ปุ่น) , 2538.

ภาคผนวก ก.



รูป ก.1 แสดงตำแหน่งของอุปกรณ์ต่างๆ บนแผ่นวงจรพิมพ์ Model 5650

Comprehensive Addressing Jumpering Map (Table 5-7A & B)

Base Addr In Hex	Address Bit/ W2 Jumper Position					
	SA9 W2 1-2	SA8 W2 3-4	SA7 W2 5-6	SA6 W2 7-8	SA5 W2 9-10	SA4 W2 11-12
100	S	O	S	S	S	S
110	S	O	S	S	S	O
120	S	O	S	S	O	S
130	S	O	S	S	O	O
140	S	O	S	O	S	S
150	S	O	S	O	S	O
160	S	O	S	O	O	S
170	S	O	S	O	O	O
180	S	O	O	S	S	S
190	S	O	O	S	S	O
1A0	S	O	O	S	O	S
1B0	S	O	O	S	O	O
1C0	S	O	O	O	S	S
1D0	S	O	O	O	S	O
1E0	S	O	O	O	O	S
1F0	S	O	O	O	O	O

Table 5-7A. Complete Address Table with SA9 Low (W2 1-2 shorted)

Note:
Addresses below 100H are used by the motherboard and are therefore invalid.

S = Shorted
O = Open

รูป ก.2

Base Addr In Hex	Address Bit/ W2 Jumper Position					
	SA9 W2 1-2	SA8 W2 3-4	SA7 W2 5-6	SA6 W2 7-8	SA5 W2 9-10	SA4 W2 11-12
200	O	S	S	S	S	S
210	O	S	S	S	S	O
220	O	S	S	S	O	S
230	O	S	S	S	O	O
240	O	S	S	O	S	S
250	O	S	S	O	S	O
260	O	S	S	O	O	S
270	O	S	S	O	O	O
280	O	S	O	S	S	S
290	O	S	O	S	S	O
2A0	O	S	O	S	O	S
2B0	O	S	O	S	O	O
2C0	O	S	O	O	S	S
2D0	O	S	O	O	S	O
2E0	O	S	O	O	O	S
2F0	O	S	O	O	O	O
300	O	O	S	S	S	S
310	O	O	S	S	S	O
320	O	O	S	S	O	S
330	O	O	S	S	O	O
340	O	O	S	O	S	S
350	O	O	S	O	S	O
360	O	O	S	O	O	S
370	O	O	S	O	O	O
380	O	O	O	S	S	S
390	O	O	O	S	S	O
3A0	O	O	O	S	O	S
3B0	O	O	O	S	O	O
3C0	O	O	O	O	S	S
3D0	O	O	O	O	S	O
3E0	O	O	O	O	O	S
3F0	O	O	O	O	O	O

Table 5-7B. Complete Address Table with SA9 High (W2 1-2 open)

รูป ก.2 (ต่อ)

J1 Connector Pinout

Table 5-1 details the signals available at J2.

J2 Pin Numbers By Axis					
#1	#2	#3	#4	Name	Description
1	26	51	76	Encoder +5Vdc	Encoder Power Output
2	27	52	77	Encoder Gnd	
3	28	53	78	Phase A	Encoder Phase A and Compliment
4	29	54	79	Phase /A	
5	30	55	80	Phase B	Encoder Phase B and Compliment
6	31	56	81	Phase /B	
7	32	57	82	Phase I	Encoder Index and Compliment
8	33	58	83	Phase /I	
9	34	59	84	+5Vdc	Power Out
10	35	60	85	Gnd	
11	36	61	86	CW Limit	Directional Limit Inputs
12	37	62	87	CCW Limit	
13	38	63	88	Home	Home Position Capture Input
14	39	64	89	Open	Not Used - Floating
15	40	65	90	+5Vdc	Power Out
16	41	66	91	Gnd	
17	42	67	92	Motor Enable	Motor Enable Output (Open-Collector)
18	43	68	93	Open	Not Used - Floating
19	44	69	94	Reserved	Do Not Connect
20	45	70	95	PWM Mag	PWM Magnitude Motor Output
21	46	71	96	Analog Out / PWM Dir	Analog Motor Output or PWM Direction Output
22	47	72	97	Analog Ret	Analog Return
23	48	73	98	+12Vdc	Reference Voltages used for Analog Output
24	49	74	99	-12Vdc	
25	50	75	100	Gnd	

Note: If the Technology 80 100-conductor to two 50-conductor cable is used the Axis 3 and 4 pin numbers are duplicates of the Axis 1 and 2 pin numbers.

ıble 5-1. J1 Connector Pinout

รูป ก.3

Signal	Axis 1	Axis 2	Axis 3	Axis 4
Analog Out	J1-21	J1-46	J1-71	J1-96
Analog Ret	J1-22	J1-47	J1-72	J1-97

Table 3-1. J1 Analog Output Pins

รูป ก.4

Signal	Axis 1/2	Axis 3/4
Motor Enable 1/2	J1-17/J1-42	
Motor Enable 3/4		J1-67/J1-92

Table 3-6. J1 Motor Enable Pins

รูป ก.5

W3	Active State	Power-Up State
Open	High (+5Vdc)	Low (0Vdc)
Shorted	Low (0Vdc)	High (+5Vdc)

Table 2-4. Motor Enable Polarity (W3)
Note: Shaded sections are default.

รูป ก.6

Signal	Axis 1	Axis 2	Axis 3	Axis 4
Enc Pwr	J1-1	J1-26	J1-51	J1-76
Enc Gnd	J1-2	J1-27	J1-52	J1-77
Phase A	J1-3	J1-28	J1-53	J1-78
Phase /A	J1-4	J1-29	J1-54	J1-79
Phase B	J1-5	J1-30	J1-55	J1-80
Phase /B	J1-6	J1-31	J1-56	J1-81
Index	J1-7	J1-32	J1-57	J1-82
/Index	J1-8	J1-33	J1-58	J1-83

Table 3-4. J1 Differential Encoder Input Pins

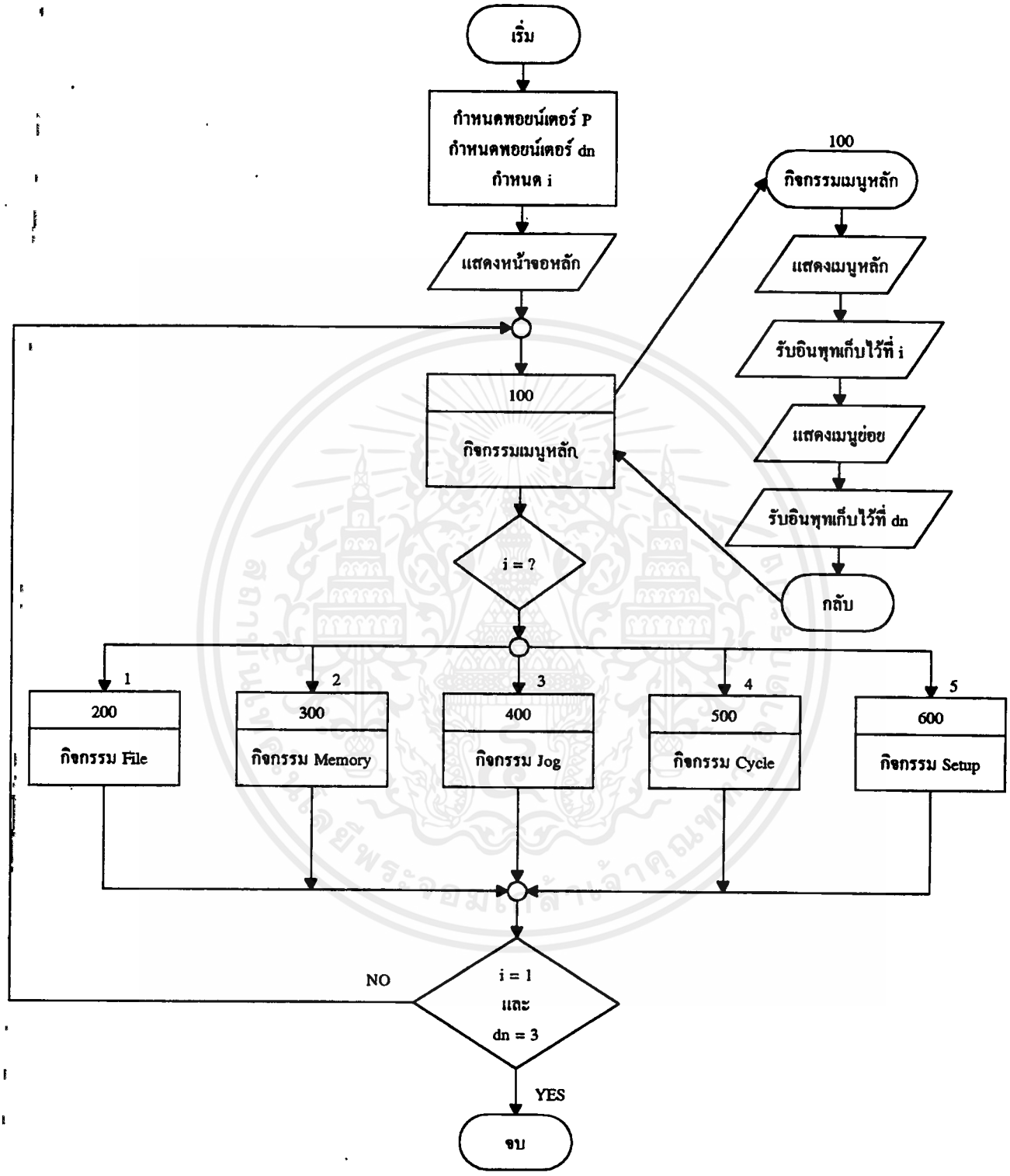
รูป ก.7

Signal	Axis 1	Axis 2	Axis 3	Axis 4
Negative Limit	J1-11	J1-36	J1-61	J1-86
Positive Limit	J1-12	J1-37	J1-62	J1-87
Home	J1-13	J1-38	J1-63	J1-88

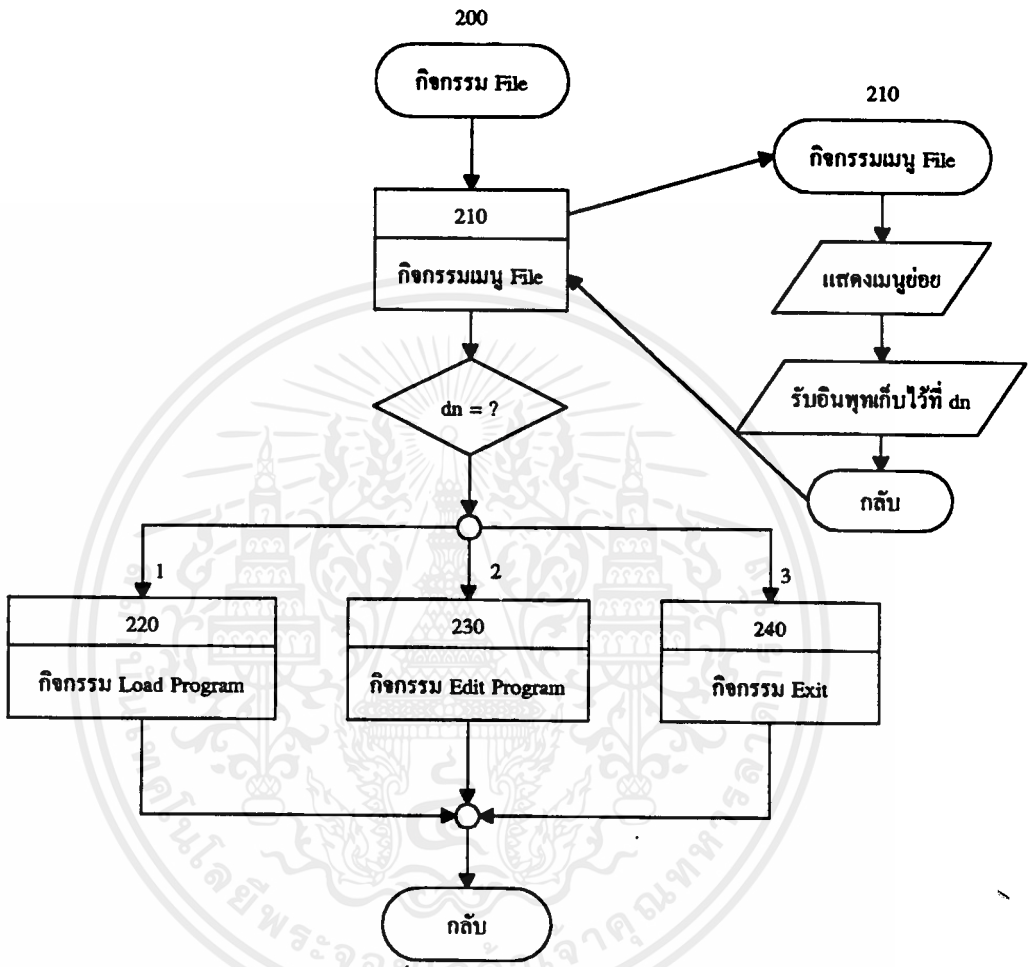
Table 3-5. J1 Limit/Home Input Pins

รูป ก.8

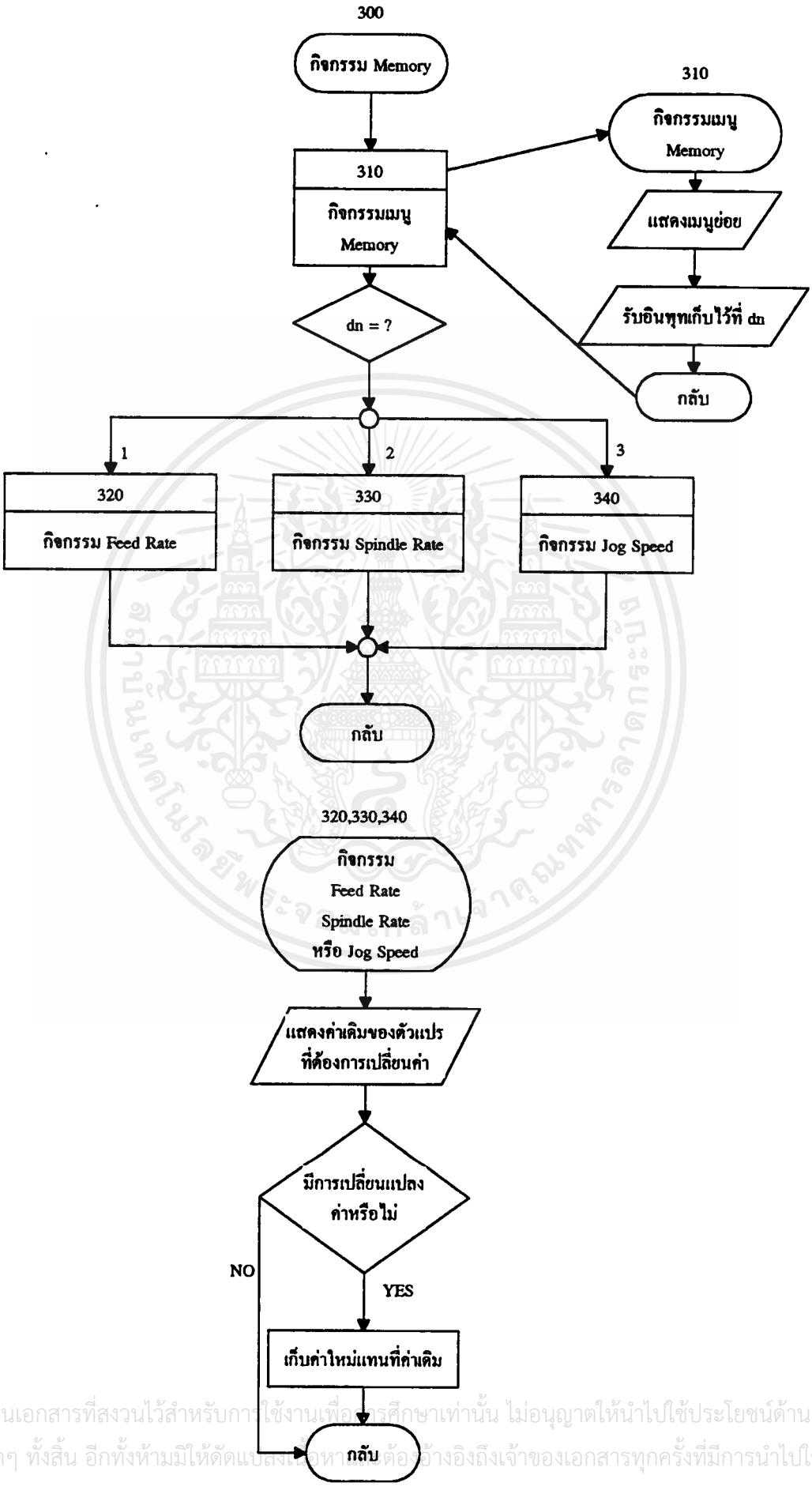
ภาคผนวก ข.

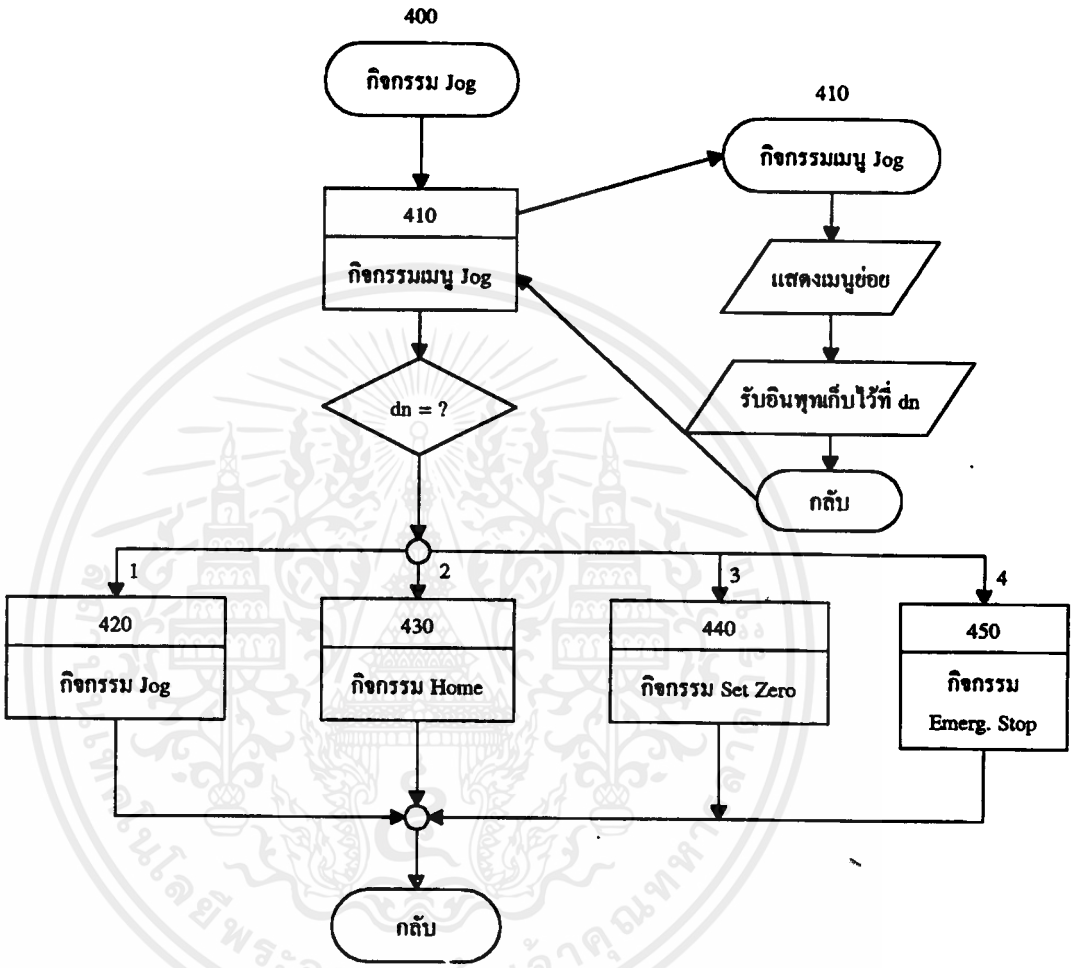


ผังโปรแกรมโครงสร้างของ Monitor.C

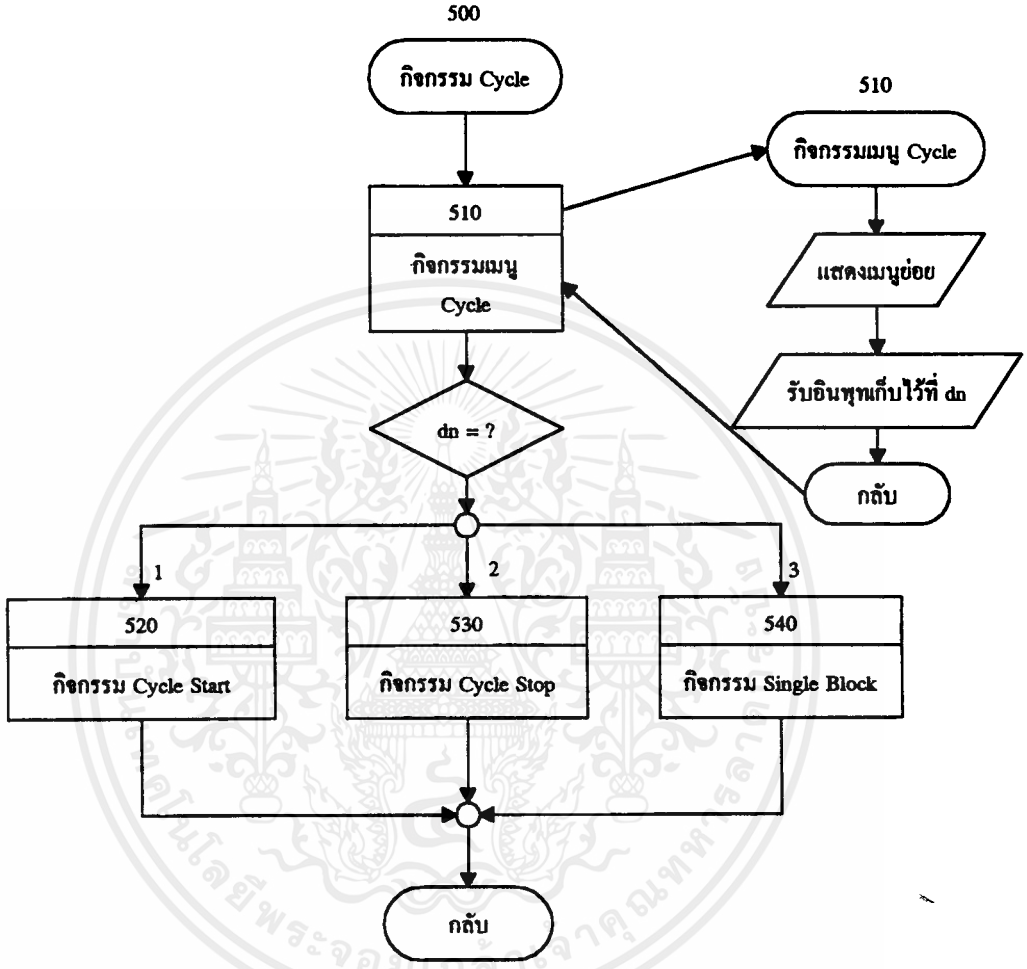


ผังโปรแกรมโครงสร้างของ Monitor.C (ต่อ)

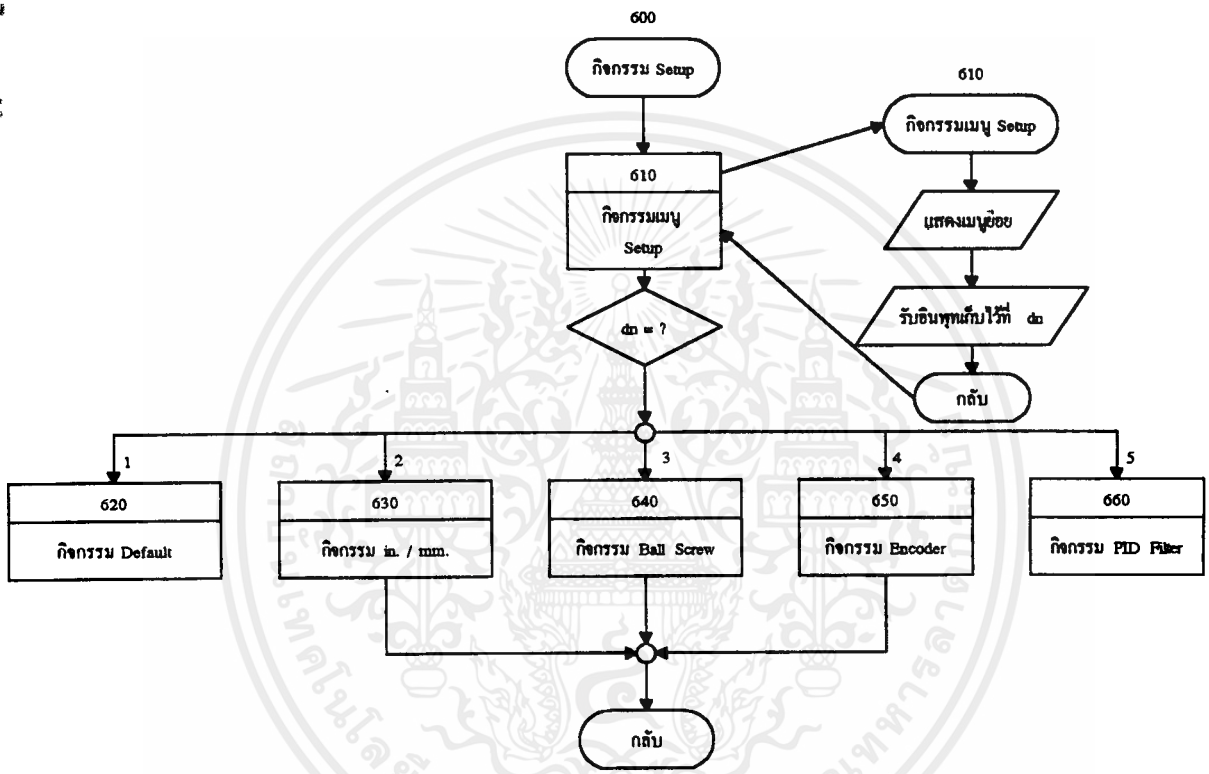




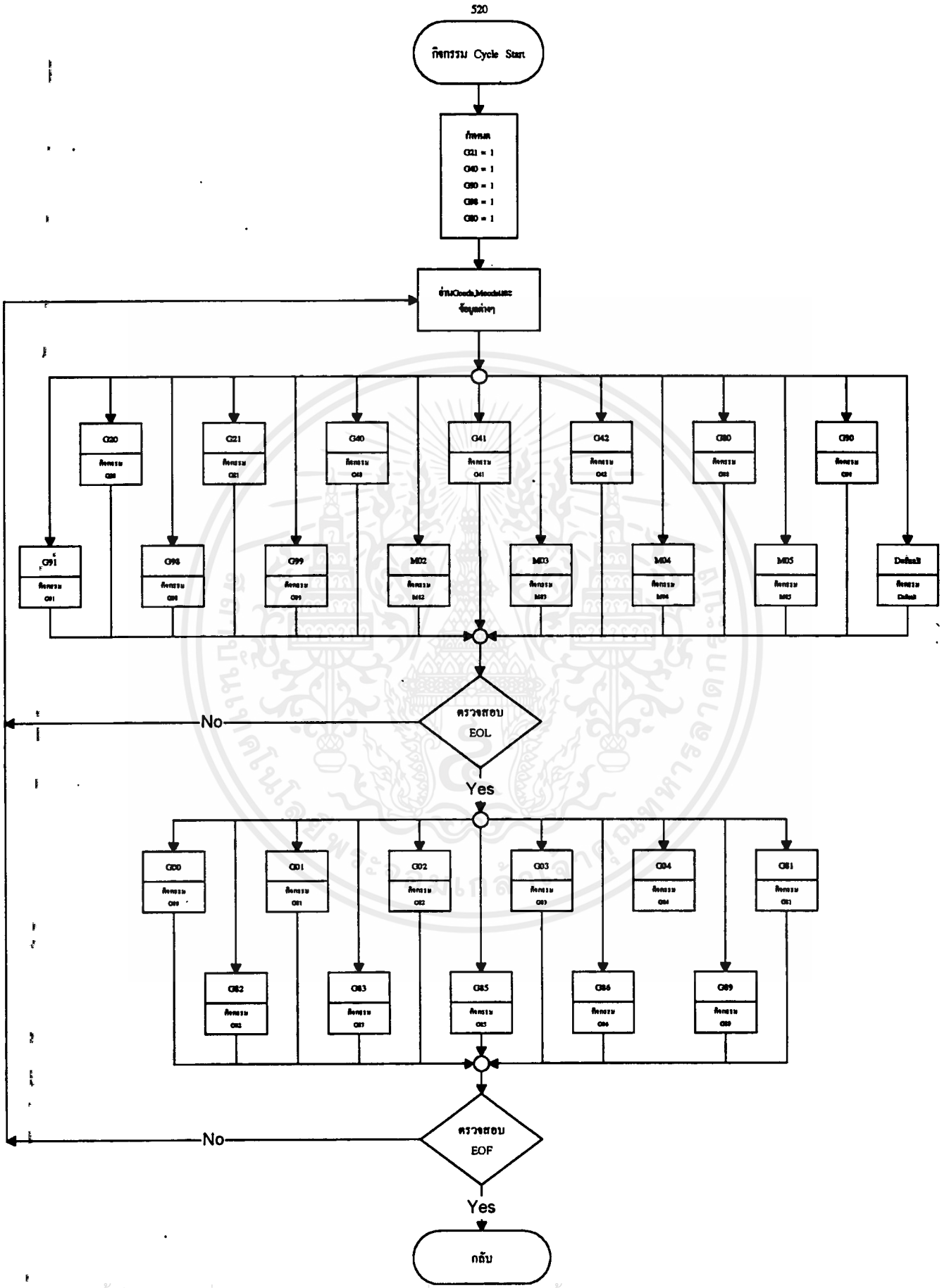
ผังโปรแกรมโครงสร้างของ Monitor.C (ต่อ)



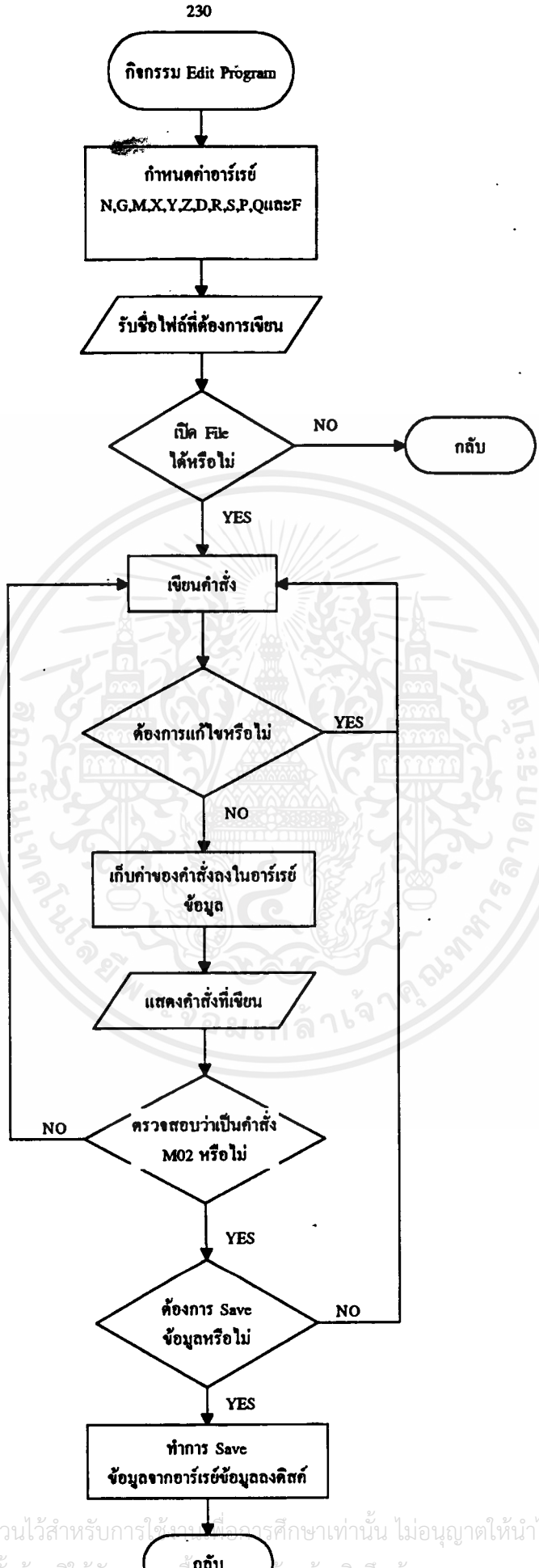
ผังโปรแกรมโครงสร้างของ Monitor.C (ต่อ)



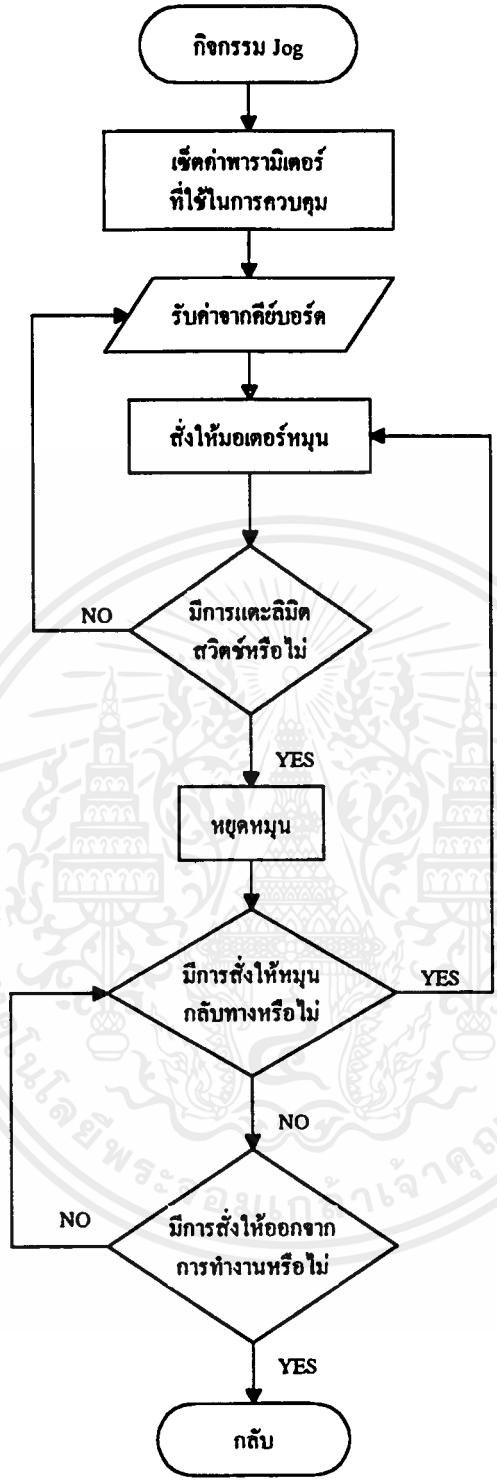
ผังโปรแกรมโครงสร้างของ Monitor.C (ต่อ)



ผังโปรแกรมโครงสร้างของ Automatic.C



ผังโปรแกรมโครงสร้างของ Edit.C



ผังโปรแกรมโครงสร้างของ Jog.C

Monitor . C

```
#include<conio.h>
#include<stdio.h>
#include<dos.h>
#include<math.h>
#include<ctype.h>
#include<graphics.h>

#define MAXMAINMENU  5
#define high         18  /*define color*/
#define norm         113
#define color1       07
#define color2       112
#define XMAX         319
#define YMAX         199
#define BLUE         1
#define GREEN        2
#define RED          4
#define INTENSE      8
#define BLUE_BACK    16
#define GREEN_BACK   32
#define RED_BACK     64
#define BLINK        128
#define KBD_PORT     0x60
#define UP     0
#define DOWN   1

char *menu[MAXMAINMENU]={" File  ",
                         " Memory  ",
                         " Jog  ",
                         " Cycle  ",
                         " Setup  "}; /*title menu bar*/

typedef struct{
                int l,t,r,b,max;
                char *text,*save;
        }PULLDOWN;  /*STRUCTURE OF PULLDOWN MENU*/
```

```
float  feed,jog;
float  ax,ay,az,rx,ry,rz;
int    prop,i,d;
int    startx[30],starty[30],endx[30],endy[30];
int    start=0,block=0;
int    con1=1,con2=1,con3=1;
int    tog=1,quit,ch;
int    ball,pulse,spindle;
int    file;
char   nme[14];
FILE   *fp;
cursoroff()
{
        union REGS reg;

        reg.h.ah=1;
        reg.x.cx=0x2000;
        int86(0x10,&reg,&reg);
}

cursoron()
{
        union REGS reg;

        reg.h.ah=1;
        reg.h.ch=6;
        reg.h.cl=7;
        int86(0x10,&reg,&reg);
}

/*-------------------------------------------------------------------
    FUNCTION  BORDER(*PULLDOWN)
     - create the box of menu that pass into function
        input : pointer of PULLDOWN
        output: draw box on screen
```

```
-------------------------------------------------------------------*/

void Border(PULLDOWN *p)
{
        register i;

        textattr(norm);
        for(i=p->l;i<p->r;i++)
        {
          gotoxy(i,p->t);
          cprintf("%c",196);
          gotoxy(i,p->b);
          cprintf("%c",196);
        }
        for(i=p->t;i<p->b;i++)
        {
          gotoxy(p->l,i);
          cprintf("%c",179);
          gotoxy(p->r,i);
          cprintf("%c",179);
        }
          gotoxy(p->l,p->t);cprintf("%c",218);
          gotoxy(p->r,p->t);cprintf("%c",191);
          gotoxy(p->l,p->b);cprintf("%c",192);
          gotoxy(p->r,p->b);cprintf("%c",217);
}

/*-------------------------------------------------------------------
      Function Pull(*PULLDOWN)
      - save old screen
      - write text into box
          input  : pointer of PULLDOWN
          output : box with text
-------------------------------------------------------------------*/

void Pull(PULLDOWN *p)
```

```
{
        int buffer;

        buffer=(p->r-p->l+4)*
              (p->b-p->t+4)*2;
        p->save=(char *)malloc(buffer);
        gettext(p->l,p->t,p->r+2,p->b+2,p->save);
        Border(p);
        window(p->l+1,p->t+1,p->r-1,p->b-1);
        textattr(norm);
        clrscr();
        if(p->text != NULL) cputs(p->text);
}

/*-------------------------------------------------------------------
        Function Restor(*PULLDOWN)
         - restore old screen from Function Pull
           input : pointer of PULLDOWN
           output: old screen
-------------------------------------------------------------------*/

void Restor(PULLDOWN *p)
{
        puttext(p->l,p->t,p->r+2,p->b+2,p->save);
        free(p->save);
        p->save=NULL;
}

/*-------------------------------------------------------------------
        Function Rewrite(*PULLDOWN,int,char)
         - write text with attribute
           input : 1.pointer of PULLDOWN
                   2.row
                   3.attribute (63=highlight,127=normal)
           output: text with attribute row
-------------------------------------------------------------------*/
```

```
void Rewrite(PULLDOWN *p,int row,char attri)
{
        int c,chars;
        union REGS reg;

        chars=p->r-p->l;
        for(c=1;c<=chars;c++)
        {
          gotoxy(c,row);
          reg.h.ah=8;
          reg.h.bh=0;
          int86(0x10,&reg,&reg);
          reg.h.ah=9;
          reg.h.bl=attri;
          reg.h.bh=0;
          reg.x.cx=1;
          int86(0x10,&reg,&reg);
         }
}

/*-----------------------------------------------------------------
          Function Menubar()
          - write main menu bar at the top of screen
            input : none
            output: main menu
-----------------------------------------------------------------*/

void Menubar()
{
        int i,s=4;

        textattr(color1);cprintf("%80c",32);
        for(i=0;i<MAXMAINMENU;i++)
        {
          gotoxy(s,1);
```

```
            s=s+strlen(menu[i])+4;
            cprintf(menu[i]);
        }
}

/*-------------------------------------------------------------
        Function Selmenubar(int)
        - write highlight menubar
        input : order of menubar
        output: highlight menubar
-------------------------------------------------------------*/

void Selmenubar(int s)
{
        int i,m=4;

        for(i=0;i!=s-1;i++)
        m=m+strlen(menu[i])+4;
        gotoxy(m,1);
        textattr(high);
        cprintf(menu[i]);
        textattr(norm);
}

/*-------------------------------------------------------------
        Function Normalmenubar(int)
        - write normal menubar
        input : order of menubar
        output: normal attribute of menubar
-------------------------------------------------------------*/

void Normalmenubar(int s)
{
        int i,m=4;

        for(i=0;i!=s-1;i++)
```

```
        m=m+strlen(menu[i])+4;
        gotoxy(m,1);
        textattr(color1);
        cprintf(menu[i]);
        textattr(norm);
}

/*-------------------------------------------------------------------
        Function (int)Selection(*PULLDOWN,int *)
        - draw pull down menu and wait for keyboard input
        - process about all pull down menu
        input : 1.pointer of PULLDOWN
                2.pointer of int
        output: 1.return choice of menu bar
                2.pass back by value of int*  that is choice of pull down menu
-------------------------------------------------------------------*/

int Selection(PULLDOWN *p,int *down,int *vtab)
{
        int se,mainmenu=*down,old=mainmenu,row,oldrow;

        row=oldrow=*vtab;
        window(1,1,80,25);
        Selmenubar(*down);
        Pull(p);
        do
        {
          Rewrite(p,row,high);
          se=getch();
          if(se==0)
          {
            se=getch();
            switch(se)
            { case 72 :  row--;
                         if(row<1) row=p->max;
                         break;
```

```
                case 80 :  row++;
                           if(row>p->max) row=1;
                           break;
                case 75 :  mainmenu--;
                           if(mainmenu<1) mainmenu=MAXMAINMENU;
                           break;
                case 77 :  mainmenu++;
                           if(mainmenu>MAXMAINMENU) mainmenu=1;
                           break;
                default :  break;
              }
              Rewrite(p,oldrow,norm);
            }
            oldrow=row;
          }while(se != 13 && old==mainmenu);
          window(1,1,80,25);
          Restor(p);
          *vtab=row;
          if (se==13)
          {
            *down=row;
            return(mainmenu);
          }
          else
          {
            *down=mainmenu;
            Normalmenubar(old);
            return(0);
          }
}

void About()
{
          PULLDOWN *p;

          p=(PULLDOWN *)malloc(sizeof(PULLDOWN));
```

```
        p->l=20;
        p->t=8;
        p->r=59;
        p->b=12;
        p->text=" CNC VERTICAL MILLING MACHINE VER.1.0\r\n                By\r\n
MECHANICAL ENGINEERING OF KMIT'L";
        p->save=NULL;
        Pull(p);
        getch();
        Restor(p);
}

float get_option() /*read characters and return value of characters*/
{
        char s[20],c;
        int i;

        for(i=0; (c=getchar()) != '\n';i=i+1) s[i] = c;
        s[i] = '\0';
        return(atof(s));
}

void cycle_start()
{
        if( start == 1)
        {
          window(45,15,57,16);
          textattr(norm);
          clrscr();
          textcolor(LIGHTGREEN|BLINK);
          textbackground(BLUE);
          gotoxy(1,1);cprintf(" Cycle Start ");
          window(45,17,57,18);
          textattr(norm);
          clrscr();
          textcolor(RED);
```

```
            textbackground(BLUE);
            gotoxy(1,1);cprintf(" Cycle Stop ");
            textattr(norm);
            start = 0;
            con1=1;
            con2=1;
            con3=0;
            Automatic();
        }
}

void cycle_stop()
{
        window(45,17,57,18);
        textattr(norm);
        clrscr();
        textcolor(LIGHTGREEN);
        textbackground(BLUE);
        gotoxy(1,1);cprintf(" Cycle Stop ");
        window(45,15,57,16);
        textattr(norm);
        clrscr();
        textcolor(RED);
        textbackground(BLUE);
        gotoxy(1,1);cprintf(" Cycle Start ");
        window(45,19,59,20);
        textattr(norm);
        clrscr();
        textcolor(RED);
        textbackground(BLUE);
        gotoxy(1,1);cprintf(" Single Block ");
        textattr(norm);
        start = 1;
        block = 1;
        con1=1;
        con2=1;
```

```
        con3=1;
}

void single_block()
{
        if( block == 1 )
        {
          window(45,19,59,20);
          textattr(norm);
          clrscr();
          textcolor(LIGHTGREEN|BLINK);
          textbackground(BLUE);
          gotoxy(1,1);cprintf(" Single Block ");
          window(45,17,57,18);
          textattr(norm);
          clrscr();
          textcolor(RED);
          textbackground(BLUE);
          gotoxy(1,1);cprintf(" Cycle Stop ");
          textattr(norm);
          block = 0;
          con1=0;
          con2=1;
          con3=1;
        }
}

void real()  /*display the value of position that return
                from controller card on the screen*/
{
        textcolor(YELLOW);
        textbackground(WHITE);
        gotoxy(10,3);cprintf("%8.3f",ax);
        gotoxy(10,4);cprintf("%8.3f",ay);
        gotoxy(10,5);cprintf("%8.3f",az);
        gotoxy(10,8);cprintf("%8.3f",rx);
```

```
        gotoxy(10,9);cprintf("%8.3f",ry);
        gotoxy(10,10);cprintf("%8.3f",rz);
        textattr(norm);
}

void show()  /*display the value of variables in memory menu
               in form of in. or mm. on the screen*/
{
        window(45,21,78,21);
        textattr(norm);
        clrscr();
        textcolor(LIGHTGREEN);
        textbackground(BLUE);
        gotoxy(1,1);
        if( tog == 1 ) cprintf(" Feed Rate    : %8.3f mm/min",feed);
        else cprintf(" Feed Rate    : %8.3f in/min",feed);
        textattr(norm);
        cursoroff();
        window(45,22,78,22);
        textattr(norm);
        clrscr();
        textcolor(LIGHTGREEN);
        textbackground(BLUE);
        gotoxy(1,1);cprintf(" Spindle Rate : %4d     rpm   ",spindle);
        textattr(norm);
        cursoroff();
        window(45,23,78,23);
        textattr(norm);
        clrscr();
        textcolor(LIGHTGREEN);
        textbackground(BLUE);
        gotoxy(1,1);
        if( tog == 1 ) cprintf(" Jog Speed    : %8.3f mm/min",jog);
        else cprintf(" Jog Speed    : %8.3f in/min",jog);
        textattr(norm);
```

```
        cursoroff();
}

void toggle()  /*toggle between in. and mm.*/
{
        window(28,24,37,24);
        textcolor(YELLOW);
        textbackground(GREEN);
        if( tog == 1 )
        {
          gotoxy(1,1);cprintf(" mm./min ");
          feed = feed*25.4;
          jog = jog*25.4;
          show();
          tog=0;
        }
        else
        {
          gotoxy(1,1);
          cprintf(" in./min ");
          feed = feed/25.4;
          jog = jog/25.4;
          show();
          tog=1;
        }
}

void display() /* print pattern of position displaying */
{
        window(3,14,37,24);
        textattr(norm);
        clrscr();
        textcolor(YELLOW);
        textbackground(WHITE);
        gotoxy(1,2);cprintf("       ACTUAL POSITION (ABSOLUTE) ");
        gotoxy(1,3);cprintf("     X : ");
```

```
        gotoxy(1,4);cprintf("     Y : ");
        gotoxy(1,5);cprintf("     Z : ");
        gotoxy(1,7);cprintf("       ACTUAL POSITION (RELATIVE) ");
        gotoxy(1,8);cprintf("     X : ");
        gotoxy(1,9);cprintf("     Y : ");
        gotoxy(1,10);cprintf("     Z : ");
        real();
        textattr(norm);

}

void initial() /* initializing the parameters of the program */
{
        prop = 10;
        i = 10;
        d = 10;
        ball = 5;
        pulse = 500;
        feed = 0;
        spindle = 0;
        jog = 0;
}

void Pull_for_file(PULLDOWN *p)
{
        int buffer,Num;
        float get_option(),test;

        buffer=(p->r-p->l+4)*
              (p->b-p->t+4)*2;
        p->save=(char *)malloc(buffer);
        gettext(p->l,p->t,p->r+2,p->b+2,p->save);
        Border(p);
        window(p->l+1,p->t+1,p->r-1,p->b-1);
        textattr(norm);
        clrscr();
```

```
        cursoron();
        gotoxy(1,1);cprintf("  Do you really want to exit?(Y/N): ");
        test = getch();
        if( (test==0x59)||(test==0x79) ) quit = 3;
        else quit = 4;
}

void Pull_for_memory(PULLDOWN  *p)
{
        int buffer,Num,M,K,i;
        float get_option(),test;

        buffer=(p->r-p->l+4)*
              (p->b-p->t+4)*2;
        p->save=(char *)malloc(buffer);
        gettext(p->l,p->t,p->r+2,p->b+2,p->save);
        Border(p);
        window(p->l+1,p->t+1,p->r-1,p->b-1);
        textattr(norm);
        clrscr();
        Num = p->t - 4;
        if(Num==0)
        {
          cursoroff();
          clrscr();gotoxy(1,1);cprintf("Feed Rate = %8.3f",feed);
          getch();
          cursoron();
          if( tog == 0 )
          {
            do
            {
                  clrscr();
                  gotoxy(1,1);cprintf("Feed Rate = ");
                  test=get_option();
                  if( ( test>=0)&&( test<=1200) ) feed=test;
                  else
```

```
            {
              clrscr();
              gotoxy(1,1);
              cprintf(" Value out of limit!");
              getch();
            }
      }while( !(( test>=0)&&( test<=1200)) );
   }
   else
   {
      do
      {
            clrscr();
            gotoxy(1,1);cprintf("Feed Rate = ");
            test=get_option();
            if( ( test>=0)&&( test<=47.244) ) feed=test;
            else
            {
               clrscr();
               gotoxy(1,1);
               cprintf(" Value out of limit!");
               getch();
            }
      }while( !(( test>=0)&&( test<=47.244)) );
   }
window(45,21,78,21);
textattr(norm);
clrscr();
textcolor(LIGHTGREEN);
textbackground(BLUE);
gotoxy(1,1);
if( tog== 0 )cprintf(" Feed Rate    : %8.3f mm/min",feed);
else cprintf(" Feed Rate    : %8.3f in/min",feed);
textattr(norm);
cursoroff();
}
```

```
if(Num==1)
{
   cursoroff();
   clrscr();
   gotoxy(1,1);cprintf("Spindle Rate = %4d",spindle);
   getch();
   cursoron();
   do
   {
      clrscr();
      gotoxy(1,1);cprintf("Spindle Rate = ");
      test=get_option();
      if( ( test>=0)&&( test<=1500) )spindle=test;
      else
      {
            clrscr();
            gotoxy(1,1);
            cprintf(" Value out of limit!");
            getch();
      }
   }while( !(( test>=0)&&( test<=1500)) );
   window(45,22,78,22);
   textattr(norm);
   clrscr();
   textcolor(LIGHTGREEN);
   textbackground(BLUE);
   gotoxy(1,1);cprintf(" Spindle Rate : %4d     rpm  ",spindle);
   textattr(norm);
   cursoroff();
}
if(Num==2)
{
   cursoroff();
   clrscr();
   gotoxy(1,1);cprintf("Jog Speed = %8.3f",jog);
   getch();
```

```
cursoron();
if( tog == 0)
{
  do
  {
        clrscr();
        gotoxy(1,1);cprintf("Jog Speed = ");
        test=get_option();
        if( ( test>=0)&&( test<=2500) )jog=test;
        else
        {
          clrscr();
          gotoxy(1,1);
          cprintf(" Value out of limit!");
          getch();
        }
  }while( !(( test>=0)&&( test<=2500)) );
}
else
{
  do
  {
        clrscr();
        gotoxy(1,1);cprintf("Jog Speed = ");
        test=get_option();
        if( ( test>=0)&&( test<=98.425) )jog=test;
        else
        {
          clrscr();
          gotoxy(1,1);
          cprintf(" Value out of limit!");
          getch();
        }
  }while( !(( test>=0)&&( test<=98.425)) );
}
window(45,23,78,23);
```

```
            textattr(norm);
            clrscr();
            textcolor(LIGHTGREEN);
            textbackground(BLUE);
            gotoxy(1,1);
            if ( tog == 0)cprintf(" Jog Speed    : %8.3f mm/min",jog);
            else cprintf(" Jog Speed    : %8.3f in/min",jog);
            textattr(norm);
            cursoroff();
          }
}

void Pull_for_jog(PULLDOWN *p)
{
          int buffer,Num;
          float get_option(),test;

          buffer=(p->r-p->l+4)*
                (p->b-p->t+4)*2;
          p->save=(char *)malloc(buffer);
          gettext(p->l,p->t,p->r+2,p->b+2,p->save);
          Border(p);
          window(p->l+1,p->t+1,p->r-1,p->b-1);
          textattr(norm);
          clrscr();
          Num = p->t - 4;
          if(Num==0 || Num==1 || Num==2 || Num==3 || Num==4)
          {
            if(Num==0)
            {
              cprintf(" Press F1 => X+, F2 => X-\r\n");
              cprintf("       F3 => Y+, F4 => Y-\r\n");
              cprintf("       F5 => Z+, F6 => Z- ");
              Jog();
              while( getch()!= 0x1b );
            }
```

```
        if(Num==1)
        {
          textcolor(GREEN|BLINK);
          textbackground(BLUE);
          gotoxy(11,2);cprintf(" HOMING...");
        textattr(norm);
          while( getch()!= 0x1b );
        }
        if(Num==2)
        {
          textcolor(GREEN|BLINK);
          textbackground(BLUE);
          gotoxy(6,2);cprintf(" NOW,SET ZERO POINT ");
          textattr(norm);
          while( getch()!= 0x1b );
        }
        if(Num==3)
        {
          textcolor(RED|BLINK);
          textbackground(GREEN);
          gotoxy(7,2);cprintf(" EMERGENCY STOP! ");
        textattr(norm);
          while( getch()!= 0x1b );
        }
      }
      cursoroff();
}

void Pull_for_setup(PULLDOWN *p)
{
      int buffer,Num;
      float get_option(),test;

      buffer=(p->r-p->l+4)*
            (p->b-p->t+4)*2;
      p->save=(char *)malloc(buffer);
```

```
gettext(p->l,p->t,p->r+2,p->b+2,p->save);
Border(p);
window(p->l+1,p->t+1,p->r-1,p->b-1);
textattr(norm);
clrscr();
Num = p->t - 4;
if(Num == 0)
{
   textcolor(GREEN|BLINK);
   textbackground(BLUE);
   gotoxy(2,1);
   cprintf(" Use default to setup ");
   initial();
   while( getch() != 0x1b );
   textattr(norm);
}
if(Num == 2)
{
   cursoroff();
clrscr();
   gotoxy(1,1);cprintf(" Pitch(mm./rev) = %d",ball);
   getch();
   cursoron();
   do
   {
      clrscr();
      gotoxy(1,1);cprintf(" Pitch(mm./rev) = ",ball);
      test = (int)get_option();
      if( test > 0) ball=test;
      else
      {
            gotoxy(1,1);
            if( test == 0 )break;
            cprintf("  Value out of limit!");
            getch();
      }
```

```
    }while(  test<=0 );
  }
  if(Num == 3)
  {
    cursoroff();
    clrscr();
    gotoxy(1,1);cprintf("  Pulse/rev = %d",pulse);
    getch();
    cursoron();
    do
    {
      clrscr();
      gotoxy(1,1);cprintf("  Pulse/rev = ");
      test = (int)get_option();
      if(  test > 0) pulse=test;
      else
      {
            clrscr();
            if( test == 0 )break;
            gotoxy(1,1);
            cprintf("  Value out of limit!");
            getch();
      }
    }while(  test<=0 );
  }
  if(Num == 4)
  {
    cursoroff();
    do
    {
      clrscr();
      gotoxy(1,1);clreol();
      cprintf("  P = %d",prop);
      getch();
      cursoron();
      do
```

```
{
      gotoxy(1,1);
      clreol();
      cprintf("  P = ");
      test = (int)get_option();
      if( (test > 0)&&(test < 20)) prop=test;
      else
      {
         gotoxy(1,1);
         if ( test == 0 )break;
         cprintf("  Value out of limit!");
         getch();
      }
}while( !((test > 0)&&(test<20)) );
gotoxy(1,1);clreol();
cprintf("  P = %d",prop);
cursoroff();
gotoxy(1,2);clreol();
cprintf("  I = %d",i);
getch();
cursoron();
do
{
      gotoxy(1,2);
      clreol();
      cprintf("  I = ");
      test = (int)get_option();
      if( (test > 0)&&(test<20)) i=test;
      else
      {
         gotoxy(1,2);
         if ( test == 0 )break;
         cprintf("  Value out of limit!");
         getch();
      }
 }while( !((test > 0)&&(test<20)) );
```

```
            gotoxy(1,2);clreol();
            cprintf("  I = %d",i);
            cursoroff();
            gotoxy(1,3);
            clreol();
            cprintf("  D = %d",d);
            getch();
            cursoron();
            do
            {
                gotoxy(1,3);
                clreol();
                cprintf("  D = ");
                test = (int)get_option();
                if( (test > 0)&&(test<20)) d=test;
                else
                {
                   gotoxy(1,3);
                   if( test == 0)break;
                   cprintf("  Value out of limit!");
                   getch();
                }
            }while( !((test > 0)&&(test<20)) );
            gotoxy(1,3);
            clreol();
            cprintf("  D = %d",d);
            gotoxy(1,4);cprintf("  Are you sure?(Y/N) : ");
            test = getch();
          }while( (test!=0x59)&&(test!=0x79) );
        }
}


void Select_file(Num)
int Num;
```

```
{
        int buffer;
        PULLDOWN *p;

        p=(PULLDOWN *)malloc(sizeof(PULLDOWN));
        p->l=20;
        p->t=9;
        p->r=59;
        p->b=11;
        p->save=NULL;
        Pull_for_file(p);
        cursoroff();
        Restor(p);
}

void Select_memory(Num)
int Num;
{
        int buffer;
        PULLDOWN *p;

        p=(PULLDOWN *)malloc(sizeof(PULLDOWN));
        p->l=28;
        p->t=4+(Num-1);
        p->r=50;
        p->b=6+(Num-1);
        p->save=NULL;
        Pull_for_memory(p);
        cursoroff();
        Restor(p);
}

void Select_jog(Num)
int Num;
{
        PULLDOWN *p;
```

```
        p=(PULLDOWN *)malloc(sizeof(PULLDOWN));
        p->l=30;
        p->t=4+(Num-1);
        p->r=60;
        p->b=8+(Num-1);
        p->save=NULL;
        Pull_for_jog(p);
        Restor(p);
}

void Select_setup(Num)
int Num;
{
        int buffer;
        PULLDOWN *p;

        p=(PULLDOWN *)malloc(sizeof(PULLDOWN));
        if(Num == 5)
        {
          p->l=45;
          p->t=4+(Num-1);
          p->r=70;
          p->b=9+(Num-1);
        }
        else
        {
          p->l=45;
          p->t=4+(Num-1);
          p->r=70;
          p->b=6+(Num-1);
        }
        p->save=NULL;
        Pull_for_setup(p);
        cursoroff();
        Restor(p);
```

```
}

/*************************************************************************/
main()
{
        PULLDOWN *p[MAXMAINMENU+1];
        int i,j,*dn,mainmenu=1,row[MAXMAINMENU+1];

        cursoroff();
        dn=(int *)malloc(sizeof(int));
        for(i=1;i<MAXMAINMENU+1;i++)
        p[i]=(PULLDOWN *)malloc(sizeof(PULLDOWN));
        for(i=1;i<MAXMAINMENU+1;i++) row[i]=1;

        p[1]->l=4;
        p[1]->t=2;
        p[1]->r=20;
        p[1]->max=3;
        p[1]->b=p[1]->t+p[1]->max+1;
        p[1]->text=" Load Program \r\n Edit Program \r\n Exit";
        p[1]->save=NULL;

        p[2]->l=12;
        p[2]->t=2;
        p[2]->r=28;
        p[2]->max=3;
        p[2]->b=p[2]->t+p[2]->max+1;
        p[2]->text=" Feed Rate \r\n Spindle Rate \r\n Jog Speed ";
        p[2]->save=NULL;

        p[3]->l=28;
        p[3]->t=2;
        p[3]->r=42;
        p[3]->max=4;
        p[3]->b=p[3]->t+p[3]->max+1;
        p[3]->text=" Jog \r\n Home\r\n Set Zero\r\n Emerg.Stop ";
```

```
p[3]->save=NULL;

p[4]->l=38;
p[4]->t=2;
p[4]->r=55;
p[4]->max=3;
p[4]->b=p[4]->t+p[4]->max+1;
p[4]->text=" Cycle Start\r\n Cycle Stop\r\n Single Block ";
p[4]->save=NULL;

p[5]->l=50;
p[5]->t=2;
p[5]->r=65;
p[5]->max=5;
p[5]->b=p[5]->t+p[5]->max+1;
p[5]->text=" Default\r\n In. / mm.\r\n Ball Screw\r\n Encoder\r\n PID Filter";
p[5]->save=NULL;

textattr(high);
clrscr();
Menubar();
About();
initial();
window(3,14,37,24);
textattr(norm);clrscr();
window(3,3,78,12);
textcolor(WHITE);
textbackground(BLACK);clrscr();
window(40,14,78,24);
textattr(norm);clrscr();
window(1,1,80,25);
textattr(norm);
cycle_stop();
display();
toggle();
do
```

```
{
window(3,3,78,12);
   textcolor(14);
   textbackground(0);
   clrscr();
   gotoxy(1,10);clreol();
   cprintf(" Program loaded: %ls",nme);
   file = 1;
   *dn=mainmenu;
   do
   {
      i=Selection(p[mainmenu],dn,&row[mainmenu]);
      if(i==0) mainmenu=*dn;
   }while(i==0);
   window(1,1,80,25);
   Pull(p[i]);
   window(1,1,80,25);
   switch(i)
   {
      case 1 : switch(*dn)
               {
                  case 1 : window(1,1,80,1);
                           clrscr();
                           Restor(p[i]);
                           Menubar();
                           window(3,3,78,12);
                           textcolor(14);
                           textbackground(0);
                           clrscr();
                           gotoxy(1,10);clreol();
                           cprintf(" Enter program name: ");
                           gets(nme);
                           if((fp=fopen(nme,"r")) == NULL)
                           {
                            clrscr();
                            gotoxy(1,10);clreol();
```

```
                    cprintf(" Error in loading program!");
                    for(i=1;i>=8;i++) nme[i] = 0 ;
                    getch();
                    gotoxy(1,10);clreol();
                   }
                   else
                   {
                    clrscr();
                    gotoxy(1,10);clreol();
                    cprintf(" Program loaded: %ls",nme);
                    fclose(fp);
                   }
                    file = 0;
                    break;
               case 2  : window(1,1,80,1);
                    clrscr();
                    Restor(p[i]);
                    Menubar();
                    Edit();
                    file = 0;
                    break;
               case 3  : Select_file(*dn);
                    *dn=quit;
                    break;
               default : break;
          }
          break;
case 2 : switch(*dn)
          {
               case  1 : Select_memory(*dn);
                         break;
               case  2 : Select_memory(*dn);
                         break;
               case  3 : Select_memory(*dn);
                         break;
               default : break;
```

```
            }
            break;
case 3 : switch(*dn)
            {
                case  1 : Select_jog(*dn);
                          break;
                case  2 : Select_jog(*dn);
                          break;
                case  3 : Select_jog(*dn);
                          break;
                case  4 : Select_jog(*dn);
                          break;
                default : break;
            }
            break;
case 4 : switch(*dn)
            {
                case  1 : if(con1)cycle_start();
                          break;
                case  2 : if(con2)cycle_stop();
                          break;
                case  3 : if(con3)single_block();
                          break;
                default : break;
            }
            break;
case 5 : switch(*dn)
            {
                case  1 : Select_setup(*dn);
                          break;
                case  2 : toggle();
                          break;
                case  3 : Select_setup(*dn);
                          break;
                case  4 : Select_setup(*dn);
                          break;
```

```
                        case  5 : Select_setup(*dn);
                                  break;
                        default : break;
                    }
        }
        if (file == 1)Restor(p[i]);
        mainmenu=i;
    }while(i!=1 || *dn!=3  );
    textattr(07);
    window(1,1,80,25);
    clrscr();
}
```

Automatic . C

```
#include<stdio.h>
#include<math.h>
#include<dos.h>
#include"mc1401.h"
#include"hrdwr.h"
#include"host_io.h"

#define pi 3.1415926
#define acc 100000

Getfi(char nme[41]);
void change_axis(int axis);
void findcnt2(int axis1,int axis2,double pos1,double pos2,double radius);
void findcnt3(int axis1,int axis2,double pos1,double pos2,double radius);

void G01(int axis1,int axis2,double pos1,double pos2,long feedrate);
void G02(int axis1,int axis2,double pos1,double pos2,double radius,long fedrate);
void G03(int axis1,int axis2,double pos1,double pos2,double radius,long fedrate);
void G04(float time);
void G73(double pos1,double pos2,double Z,double q,double R,long feedrate);
void G81(double pos1,double pos2,double Z,double R,long feedrate);
void G82(double pos1,double pos2,double Z,double R,long time,long feedrate);
void G83(double pos1,double pos2,double Z,double q,double R,long feedrate);
void G85(double pos1,double pos2,double Z,double R,long feedrate);
void G86(double pos1,double pos2,double Z,double R,long feedrate);
void G89(double pos1,double pos2,double Z,double R,long time,long feedrate);

void M03(long speed);
void M04(long speed);
void M05(void);

void Automatic(void)
{
        char    ch,nme[14];
        int  axis1,axis2,i,line,act_Gcode,Gcode,Mcode,time,k;
        int  G40,G41,G42,G90,G91,G98,G99,group09,group07;
```

```
float feedrate,speed,angle1,angle2,angle;
float multiplier,rc,dx,dy;
double    pos1,pos2,loc1,loc2,act_pos1,act_pos2,cntr1,cntr2,m,arc,radius;
double    deep,cut_in,retract,act_rad;


 group09 = 0;
 G40 = 1;
 G41 = 0;
 G42 = 0;
 G91 = 0;
 G90 = 1;
 G98 = 0;
 G99 = 0;
 multiplier = 400;
 Getfi(nme);
 if((fp = fopen(nme,"r"))==NULL)
 {
  puts("CAN'T OPEN FILE\n");
  exit(1);
 }

 while( fscanf(fp," %c",&ch) != EOF)
 {while(ch != '*')
 {
          if(ch == 'N')
                    fscanf(fp,"%5d",&line);
          if(ch == 'G')
           {fscanf(fp,"%2d",&Gcode);
            switch(Gcode)
             {
              `     /*  group 06  */
              case 20:
                    multiplier = 400*25.4;
              break;
              case 21:
                    multiplier = 400;
```

```
break;

        /*  group 07  */
case 40:
        G40 = 1;
        printf("Compensate off \n");
break;
case 41:
        group07 = 1;
        G41 = 1;
        G40 = 0;
        printf("Compensation Left on \n");
break;
case 42:
        group07 = 1;
        G42 = 1;
        G40 = 0;
        printf("Compensation Right on \n");
break;

        /*  group  09  */
case 80:
        group09 = 0;
break;
case 81:
        group09 = 1;
break;
case 82:
        group09 = 1;
break;
case 83:
        group09 = 1;
break;
case 85:
        group09 = 1;
break;
```

```
    case 86:
        group09 = 1;
    break;
    case 89:
        group09 = 1;
    break;

        /* group 03 */
    case 90:
        G90 = 1;
        G91 = 0;
        printf("Absolute Program\n");
    break;
    case 91:
        G90 = 0;
        G91 = 1;
        printf("Incremental Program\n");
    break;

        /* group 04 */
    case 98:
        G98 = 1;
        G99 = 0;
    break;
    case 99:
        G98 = 0;
        G99 = 1;
    break;
        }
 }
if(ch == 'X')
        { axis1 = 0;
            fscanf(fp,"%7f",&pos1);
        }
if(ch == 'Y')
        { axis2 = 1;
```

```
                fscanf(fp,"%7f",&pos2);
            }
    if(ch == 'Z')
            {
             if(group09)
                    fscanf(fp,"%7f",&deep)
             else
             if(axis2 == 1)
             { axis1 = 1;
                    pos1  = pos2;
             }
                    axis2 = 2;
                    fscanf(fp,"%7f",&pos2);
             }
    if(ch == 'R')
            if (group09)
             fscanf(fp,"%7f",&retract)
            else
            fscanf(fp,"%7f",&radius);
    if(ch == 'F')
            fscanf(fp,"%7f",&feedrate);
    if(ch == 'Q')
            fscanf(fp,"%7f",&cut_in);
    if(ch == 'K')
            fscanf(fp,"%5d",&k);
    if(ch == 'P')
            fscanf(fp,"%4d",&time);
    if(ch == 'S')
            fscanf(fp,"%7f",&speed);
    if(ch == 'D')
            fscanf(fp,"%3d",&rc);
    if(ch == 'M')
     {fscanf(fp,"%2d",&Mcode);
            switch(Mcode)
             {
             case 2:
```

```
                printf("M02 End of Program");
            break;
            case 3:
                printf("M03 Spindle CW On\n");
                M03(speed);
            break;
            case 4:
                printf("M04 Spindle CCW On\n");
                M04(speed);
            break;
            case 5:
                printf("M05 Spindle off\n");
                M05;
            break;
            }
        }
        fscanf(fp," %c",&ch);
}
        if (group07)
        {
          change(axis_1);
          loc1 = get_pos(1)/multiplier;
          change(axis_2);
          loc2 = get_pos(2)/multiplier;
    switch(act_Gcode)
        {
        case 0:
            angle1 = atan((act_pos2-loc2)/(act_pos1-loc1));
            if (((act_pos2-loc2)>0)&&((act_pos1-loc1)>0))
               angle1 = angle1;
            if (((act_pos2-loc2)>0)&&((act_pos1-loc1)<0))
               angle1 = pi - angle1;
            if (((act_pos2-loc2)<0)&&((act_pos1-loc1)>0))
               angle1 = pi + angle1;
            if (((act_pos2-loc2)<0)&&((act_pos1-loc1)<0))
               angle1 = 2*pi - angle1;
```

```
switch(Gcode)
{
 case 1:
     angle2 = atan((pos2-act_pos2)/(pos1-act_pos1));
     if (((pos2-act_pos2)>=0)&&((pos1-act_pos1)>=0))
          angle2 = angle2;
     if (((pos2-act_pos2)>0)&&((pos1-act_pos1)<0))
          angle2 = pi - angle2;
     if (((pos2-act_pos2)<=0)&&((pos1-act_pos1)<=0))
          angle2 = pi + angle2;
     if (((pos2-act_pos2)<0)&&((pos1-act_pos1)>0))
          angle2 = 2*pi - angle2;
 break;
 case 2:
     findcnt2(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
     if( r > 0)
      angle = asin(loc2-cntr2)/act_rad;
     else
      angle = asin(loc2-cntr2)/(-act_rad);
     if (angle<0)
          angle = - angle ;
     if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
          angle2 = angle + pi;
     if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
          angle2 = -(pi + angle);
     if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
          angle2 = pi - angle;
     if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
          angle2 = -(pi - angle);
 break;
 case 3:
     findcnt3(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
     if( r > 0)
      angle = asin(loc2-cntr2)/act_rad;
     else
      angle = asin(loc2-cntr2)/(-act_rad);
```

```
                if (angle<0)
                    angle = - angle ;
                if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
                    angle2 = angle + pi;
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                    angle2 = -(pi + angle);
                if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
                    angle2 = -(pi - angle);
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                    angle2 = pi - angle;
             break;
            }
            if (G41)
            {
             dx = -rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
             dy = rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
            }
            else
            {
             dx = rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
             dy = -rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
            }
            act_pos1 = act_pos1 + dx;
            act_pos2 = act_pos2 + dy;
            G01(axis1,axis2,act_pos1,act_pos2,60000);
act_Gcode = Gcode;
            act_pos1  = pos1;
            act_pos2  = pos2;
            act_radius= radius;
            if(G40)
            {
             group07 = 0;
             G41 = 0;
             G42 = 0;
            }
  break;
```

```
 case 1:
angle1 = atan((act_pos2-loc2)/(act_pos1-loc1));
        if (((act_pos2-loc2)>0)&&((act_pos1-loc1)>0))
           angle1 = angle1;
        if (((act_pos2-loc2)>0)&&((act_pos1-loc1)<0))
           angle1 = pi - angle1;
        if (((act_pos2-loc2)<0)&&((act_pos1-loc1)>0))
           angle1 = pi + angle1;
        if (((act_pos2-loc2)<0)&&((act_pos1-loc1)<0))
           angle1 = 2*pi - angle1;
        switch(Gcode)
        {
         case 1:
            angle2 = atan((pos2-act_pos2)/(pos1-act_pos1));
            if (((pos2-act_pos2)>=0)&&((pos1-act_pos1)>=0))
                angle2 = angle2;
            if (((pos2-act_pos2)>0)&&((pos1-act_pos1)<0))
                angle2 = pi - angle2;
            if (((pos2-act_pos2)<=0)&&((pos1-act_pos1)<=0))
                angle2 = pi + angle2;
            if (((pos2-act_pos2)<0)&&((pos1-act_pos1)>0))
                angle2 = 2*pi - angle2;
          break;
          case 2:
            findcnt2(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
            if( r > 0)
             angle = asin(loc2-cntr2)/act_rad;
            else
             angle = asin(loc2-cntr2)/(-act_rad);
            if (angle<0)
                angle = - angle ;
            if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
                angle2 = -(angle + pi);
            if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                angle2 = pi + angle;
            if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
```

```
                angle2 = pi - angle;
            if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                angle2 = -(pi - angle);
        break;
        case 3:
            findcnt3(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
            if( r > 0)
             angle = asin(loc2-cntr2)/act_rad;
            else
             angle = asin(loc2-cntr2)/(-act_rad);
            if (angle<0)
                angle = - angle ;
            if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
                angle2 = angle + pi;
            if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                angle2 = -(pi + angle);
            if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
                angle2 = -(pi - angle);
            if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                angle2 = pi - angle;
         break;
        }
if (G41)
        {
         dx = -rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
         dy = rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
        }
        else
        {
         dx = rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
         dy = -rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
        }
        act_pos1 = act_pos1 + dx;
        act_pos2 = act_pos2 + dy;
        G01(axis1,axis2,act_pos1,act_pos2,feedrate);
act_Gcode = Gcode;
```

```
        act_pos1  = pos1;
        act_pos2  = pos2;
        act_radius= radius;
        if(G40)
        {
         group07 = 0;
         G41 = 0;
         G42 = 0;
        }
 break;
 case 2:
        findcnt2(axis1,axis2,act_pos1,act_pos2,loc1,loc2);
        angle = asin(act_pos2-cntr2)/rc;
        if (angle<0)
           angle = -angle;
        if (((act_pos2-cntr2)>=0)&&((act_pos1-cntr1)>=0))
           angle1 = pi+angle;
        if (((act_pos2-cntr2)> 0)&&((act_pos1-cntr1)< 0))
           angle1 = -(pi+angle);
        if (((act_pos2-crnt2)<=0)&&((act_pos1-crnt1)<=0))
           angle1 = -(pi-angle);
        if (((act_pos2-crnt2)< 0)&&((act_pos1-crnt1)> 0))
           angle1 = pi-angle;
switch(Gcode)
        {
         case 0:
            angle2 = atan((pos2-act_pos2)/(pos1-act_pos1));
            if (((pos2-act_pos2)>=0)&&((pos1-act_pos1)>=0))
                angle2 = angle2;
            if (((pos2-act_pos2)>0)&&((pos1-act_pos1)<0))
                angle2 = pi - angle2;
            if (((pos2-act_pos2)<=0)&&((pos1-act_pos1)<=0))
                angle2 = pi + angle2;
            if (((pos2-act_pos2)<0)&&((pos1-act_pos1)>0))
                angle2 = 2*pi - angle2;
         break;
```

```
case 1:
    angle2 = atan((pos2-act_pos2)/(pos1-act_pos1));
    if (((pos2-act_pos2)>=0)&&((pos1-act_pos1)>=0))
        angle2 = angle2;
    if (((pos2-act_pos2)>0)&&((pos1-act_pos1)<0))
        angle2 = pi - angle2;
    if (((pos2-act_pos2)<=0)&&((pos1-act_pos1)<=0))
        angle2 = pi + angle2;
    if (((pos2-act_pos2)<0)&&((pos1-act_pos1)>0))
        angle2 = 2*pi - angle2;
break;
case 2:
    findcnt2(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
    if( r > 0)
     angle = asin(loc2-cntr2)/act_rad;
    else
     angle = asin(loc2-cntr2)/(-act_rad);
    if (angle<0)
        angle = - angle ;
    if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
        angle2 = -(angle + pi);
    if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
        angle2 = pi + angle;
    if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
        angle2 = pi - angle;
    if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
        angle2 = -(pi - angle);
break;
case 3:
    findcnt3(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
    if( r > 0)
     angle = asin(loc2-cntr2)/act_rad;
    else
     angle = asin(loc2-cntr2)/(-act_rad);
    if (angle<0)
        angle = - angle ;
```

```
                if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
                    angle2 = angle + pi;
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                    angle2 = -(pi + angle);
                if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
                    angle2 = -(pi - angle);
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                    angle2 = pi - angle;
            break;
            }
if (G41)
            {
             dx = -rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
             dy = rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
             act_radius = act_radius + rc;
            }
            else
            {
             dx = rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
             dy = -rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
             act_radius = act_radius - rc;
            }
            act_pos1 = act_pos1 + dx;
            act_pos2 = act_pos2 + dy;
            G02(axis1,axis2,act_pos1,act_pos2,act_radius,feedrate);
act_Gcode = Gcode;
            act_pos1  = pos1;
            act-pos2  = pos2;
            if(G40)
            {
             group07 = 0;
             G41 = 0;
             G42 = 0;
            }
  break;
  case 3:
```

```
            findcnt3(axis1,axis2,act_pos1,act_pos2,loc1,loc2);
            angle = asin(act_pos2-cntr2)/rc;
            if (angle<0)
               angle = -angle;
            if (((act_pos2-cntr2)>=0)&&((act_pos1-cntr1)>=0))
               angle1 = -(pi+angle);
            if (((act_pos2-cntr2)> 0)&&((act_pos1-cntr1)< 0))
               angle1 = pi+angle;
            if (((act_pos2-crnt2)<=0)&&((act_pos1-crnt1)<=0))
               angle1 = pi-angle;
            if (((act_pos2-crnt2)< 0)&&((act_pos1-crnt1)> 0))
               angle1 = -(pi-angle);
      switch(Gcode)
            {
             case 0:
                angle2 = atan((pos2-act_pos2)/(pos1-act_pos1));
                if (((pos2-act_pos2)>=0)&&((pos1-act_pos1)>=0))
                    angle2 = angle2;
                if (((pos2-act_pos2)>0)&&((pos1-act_pos1)<0))
                    angle2 = pi - angle2;
                if (((pos2-act_pos2)<=0)&&((pos1-act_pos1)<=0))
                    angle2 = pi + angle2;
                if (((pos2-act_pos2)<0)&&((pos1-act_pos1)>0))
                    angle2 = 2*pi - angle2;
              break;
              case 1:
                  angle2 = atan((pos2-act_pos2)/(pos1-act_pos1));
                  if (((pos2-act_pos2)>=0)&&((pos1-act_pos1)>=0))
                      angle2 = angle2;
                  if (((pos2-act_pos2)>0)&&((pos1-act_pos1)<0))
                      angle2 = pi - angle2;
                  if (((pos2-act_pos2)<=0)&&((pos1-act_pos1)<=0))
                      angle2 = pi + angle2;
                  if (((pos2-act_pos2)<0)&&((pos1-act_pos1)>0))
                      angle2 = 2*pi - angle2;
               break;
```

```
            case 2:
                findcnt2(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
                if( r > 0)
                 angle = asin(loc2-cntr2)/act_rad;
                else
                 angle = asin(loc2-cntr2)/(-act_rad);
                if (angle<0)
                      angle = - angle ;
                if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
                      angle2 = -(angle + pi);
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                      angle2 = pi + angle;
                if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
                      angle2 = pi - angle;
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                      angle2 = -(pi - angle);
             break;
            case 3:
                findcnt3(axis1,axis2,pos1,pos2,act_pos1,act_pos2,radius);
                if( r > 0)
                 angle = asin(loc2-cntr2)/act_rad;
                else
                 angle = asin(loc2-cntr2)/(-act_rad);
                if (angle<0)
                      angle = - angle ;
                if (((act_pos2-cntr2)>=0)&&(act_pos1-cntr1)>=0))
                      angle2 = angle + pi;
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                      angle2 = -(pi + angle);
                if (((act_pos2-cntr2)<=0)&&(act_pos1-cntr1)<=0))
                      angle2 = -(pi - angle);
                if (((act_pos2-cntr2)> 0)&&(ant_pos1-cntr1)< 0))
                      angle2 = pi - angle;
             break;
            }
if (G41)
```

```
                {
                 dx = -rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
                 dy = rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
                 act_radius = act_radius - rc;
                }
                else
                {
                 dx = rc * sin((angle2+angle1)/2) / cos((angle2-angle1)/2);
                 dy = -rc * cos((angle2+angle1)/2) / cos((angle2-angle1)/2);
                 act_radius = act_radius + rc;
                }
                act_pos1 = act_pos1 + dx;
                act_pos2 = act_pos2 + dy;
                G02(axis1,axis2,act_pos1,act_pos2,act_radius,feedrate);
        act_Gcode = Gcode;
                act_pos1  = pos1;
                act-pos2  = pos2;
                if(G40)
                {
                 group07 = 0;
                 G41 = 0;
                 G42 = 0;
                }
          break;
          } /*for G_code*/
        }/*for  act_Gcode*/
        if (G40)
        switch(Gcode) /* group for movement */
{
 case 0:
        printf("N %d G00 %d %f %d %f\n",line,axis1,pos1,axis2,pos2);
        /*G01(axis1,axis2,pos1,pos2,60000);*/
    if (G91)
        {
                 pos1 = 0;
                 pos2 = 0;
```

```
        }
        break;

case 1:
        printf("N %d G01 %d %f %d %f %f\n",line,axis1,pos1,axis2,pos2,feedrate);
        if (G91)
        {
                change(axis1);
                pos1 = pos1 + (get_pos(0)/multiplier);
                change(axis2);
                pos2 = pos2 + (get_pos(0)/multiplier);
        }
        G01(axis1,axis2,pos1,pos2,feedrate);
        if (G91)
        {
                pos1 = 0;
                pos2 = 0;
        }
        break;

case 2:
        printf("N %d G02 %d %f %d %f %f %f\n",line,axis1,pos1,axis2,pos2,radius,feedrate);
        if (G91)
        {
                change(axis1);
                pos1 = pos1 + (get_pos(0)/multiplier);
                change(axis2);
                pos2 = pos2 + (get_pos(0)/multiplier);
        }
        G02(axis1,axis2,pos1,pos2,radius,feedrate);
        if (G91)
        {
                pos1 = 0;
                pos2 = 0;
        }
        break;
```

```
case 3 :
        printf("N %d G03 %d %f %d %f %f %f\n",line,axis1,pos1,axis2,pos2,radius,feedrate);
        if (G91)
        {
                change(axis1);
                pos1 = pos1 + (get_pos(0)/multiplier);
                change(axis2);
                pos2 = pos2 + (get_pos(0)/multiplier);
        }
        G03(axis1,axis2,pos1,pos2,radius,feedrate);
        if (G91)
        {
                pos1 = 0;
                pos2 = 0;
        }
        break;

case 73:
        G73(pos1,pos2,deep,cut_in,retract,feedrate);
        if (G91)
        {
                pos1 = 0;
                pos2 = 0;
        }
        break;

case 81:
        G81(pos1,pos2,deep,retract,feedrate);
        if (G91)
        {
                pos1 = 0;
                pos2 = 0;
        }
        break;
```

```
                {
                        pos1 = 0;
                        pos2 = 0;
                }
                break;
            }
        }
            fclose(fp);
}


/*internal functions*/

Getfi(name)
char name[41];
{
 int coutn,flag;
 char ch;

 flag = 1;
 while (flag)
    {
        cputs("ENTER DNC FILE NAME: ");
        gets(name);
        flag = 0;
        if (strlen(name)>12)
        {
                puts("FILE NAME IS TOO LONG.\n");
                flag = 1 ;
        }
    }
 }


 void change_axis(int axis)
 {
```

```
        switch(axis)
        {
         case 0:
                set_1(0);
                break;
         case 1:
                set_2(0);
                break;
         case 2:
                set_3(0);
                break;
         case 3:
                set_4(0);
                break;
         default:
                break;
         }
}

void findcnt2(
int axis1,
int axis2,
double pos1,
double pos2,
double x1,
double y1,
double radius)
{  double x2,y2,x3,y3,x4,y4


         x2 = pos1;
         y2 = pos2;
         r  = radius;
         if( (y2-y1)==0 )
                { a = (x1+x2)/2;
                        b = (y1+y2)/2;
                        l = x2-x1;
```

```
                p = pow(r,2)-pow((l/2),2);
                x3= a;
                y3= b-sqrt(p);
                x4= a;
                y4= b+sqrt(p);}
else{n = -((x2-x1)/(y2-y1));
a = (x1+x2)/2;
b = (y1+y2)/2;
l = sqrt((pow((x2-x1),2)+pow((y2-y1),2)));
p = pow(r,2)-pow((l/2),2);
x3 = sqrt(p/(1+pow(n,2)));
y3 = n*x3;
x4 = -x3;
y4 = n*x4;
x3 = x3+a;
y3 = y3+b;
x4 = x4+a;
y4 = y4+b;}

if ( r > 0 )
{
  if ( (((x2-x1)*(y3-y1))-((y2-y1)*(x3-x1))) < 0 )
   {
        cntr1 = x3;
        cntr2 = y3;
   }
  else
   {
        cntr1 = x4;     /* find center finish*/
        cntr2 = y4;
   }
 }
 else
 {
  if ( (((x2-x1)*(y3-y1))-((y2-y1)*(x3-x1))) >= 0 )
   {
```

```
                cntr1 = x3;
                cntr2 = y3;
            }
            else
            {
                cntr1 = x4;     /* find center finish*/
                cntr2 = y4;
            }
        }
}

void findcnt3(
int axis1,
int axis2,
double pos1,
double pos2,
double x1,
double y1,
double radius)
{  double x2,y2,x3,y3,x4,y4

        x2 = pos1;
        y2 = pos2;
        r  = radius;
        if( (y2-y1)==0 )
                { a = (x1+x2)/2;
                        b = (y1+y2)/2;
                        l = x2-x1;
                        p = pow(r,2)-pow((l/2),2);
                        x3= a;
                        y3= b-sqrt(p);
                        x4= a;
                        y4= b+sqrt(p);}
        else{n = -((x2-x1)/(y2-y1));
        a = (x1+x2)/2;
        b = (y1+y2)/2;
```

```
l = sqrt((pow((x2-x1),2)+pow((y2-y1),2)));
p = pow(r,2)-pow((l/2),2);
x3 = sqrt(p/(1+pow(n,2)));
y3 = n*x3;
x4 = -x3;
y4 = n*x4;
x3 = x3+a;
y3 = y3+b;
x4 = x4+a;
y4 = y4+b;}

if ( r > 0 )
{
  if ( (((x2-x1)*(y3-y1))-((y2-y1)*(x3-x1))) >= 0 )
   {
        cntr1 = x3;
        cntr2 = y3;
   }
  else
   {
        cntr1 = x4;      /* find center finish*/
        cntr2 = y4;
   }
 }
 else
 {
  if ( (((x2-x1)*(y3-y1))-((y2-y1)*(x3-x1))) < 0 )
   {
        cntr1 = x3;
        cntr2 = y3;
   }
  else
   {
        cntr1 = x4;      /* find center finish*/
        cntr2 = y4;
   }
```

```
        }
}

void G01(
int axis_1,
int axis_2,
double X1,
double X2,
long feedrate)
{ long   length,loc1,loc2,v1,v2;
         long    s,t,pos1,pos2;
         change_axis(axis_1);
         loc1 = get_pos(0);
         clr_status(0);
         change_axis(axis_2);
         loc2 = -get_pos(0);
         clr_status(0);
         pos1 = (long)(multiplier*X1);
         pos2 = -(long)(multiplier*X2);
         if( ((pos1-loc1)!=0)||((pos2-loc2)!=0) )
         length = (long)(sqrt(pow((pos1-loc1),2)+pow((pos2-loc2),2)));
         else length = 1 ;
         v1 = (long)(abs(feedrate*(pos1-loc1)/length));
         v2 = (long)(abs(feedrate*(pos2-loc2)/length));

         change_axis(axis_1);
         set_pos(pos1);
         set_vel(v1);
         change_axis(axis_2);
         set_pos(pos2);
         set_vel(v2);
         multi_update(0x7);
         if( ((pos1-loc1)!=0)&&((pos2-loc2)!=0) ){
                  do{
                  change_axis(axis_1);
                  s=get_status(0);
```

```
                    change_axis(axis_2);
                    t=get_status(0);
                    }while( (((s!=0x301)&&(s!=0x309))||((t!=0x1301)&&(t!=0x1309)))&&(((s!=0
x301)&&(s!=0x309))||((t!=0x2301)&&(t!=0x2309)))&&(((s!=0x1301)&&(s!=0x1309))||((t!=0x2301)&&(t!=0
x2309))) );}
          else if((pos2-loc2)!=0){
          change_axis(axis_2);
          do{ s=get_status(0);
          }while( (s!=0x1301)&&(s!=0x1309)&&(s!=0x2301)&&(s!=0x2309) );}
          else if((pos1-loc1)!=0){
          change_axis(axis_1);
          do{ s=get_status(0);
          }while( (s!=0x301)&&(s!=0x309)&&(s!=0x1301)&&(s!=0x1309) );}
}

void G02(int axis_1,int axis_2,double pos1,double pos2,double radius,long feedrate)

{  double x1,y1,x2,y2,x3,y3,x4,y4,X,Y,tempX,tempY,a,b,r,n,l,p,feed;
          double start,step,arc,angle;
          long i,loop;

          findcnt2(axis1,axis2,pos1,pos2,radius);
          /* find start angle */
          change_axis(axis1);
          x1 = get_pos(1);
          clr_status(0);
          change_axis(axis2);
          y1 = -get_pos(2);
          clr_status(0);
          if ( r > 0 ) start = asin((y1-cntr2)/r);
          else start = asin((y1-cntr2)/r);
          if ( start < 0 )start=-start;
        if( ((y1-cntr2) >= 0)&&((x1-cntr1)>= 0) )
                              start = start;
          if( ((y1-cntr2) >  0)&&((x1-cnre1) < 0) )
                              start = pi-start;
```

```
        if( ((y1-cntr2) <= 0)&&((x1-cntr1)<= 0) )
                        start = pi+start;
        if( ((y1-cntr2) <  0)&&((x1-cntr1) > 0) )
                        start = (2*pi)-start;

        if ( r > 0 ) arc = 2*asin(l/2/r);
        else arc = 2*asin(-l/2/r);

        /*find step angle*/
        loop = (long)(abs(50*r*arc));
        if( loop > 0 )  step = arc/loop;
        else step = 0;
        for(i=1;i<=loop;i++)
        { angle = start-(i*step);
          if ( r > 0 )
          {
                tempX = cntr1 + (r*cos(angle));
                tempY = cntr2 + (r*sin(angle));
          }
          else
          {
                tempX = cntr1 - (r*cos(angle));
                tempY = cntr2 - (r*sin(angle));
          }
          X1=tempX ;
          X2=tempY ;
          G01(axis_1,axis_2,X1,X2,feed);
        }
        X1 = x2;
        X2 = y2;
        G01(axis_1,axis_2,X1,X2,feed);
}

void G03(int axis_1,int axis_2,double pos1,double pos2,double radius,long feedrate)
```

```
{  double x1,y1,x2,y2,x3,y3,x4,y4,X,Y,tempX,tempY;a,b,r,n,l,p,feed;
        double start,step,arc,angle;
        long i,loop;

        change_axis(axis_1);
        x1 = (get_pos(0)/multiplier);
        clr_status(0);
        change_axis(axis_2);
        y1 = -(get_pos(0)/multiplier);
        clr_status(0);
        /*find start angle*/
        if( r > 0 )  start = asin((y1-Y)/r);
        else start = asin((y1-Y)/(-r));
        if(start < 0)start=-start;
        if( ((y1-cntr2) >= 0)&&((x1-cntr1)>= 0) )
                        start = start;
        if( ((y1-cntr2) >  0)&&((x1-cnre1) < 0) )
                        start = pi-start;
        if( ((y1-cntr2) <= 0)&&((x1-cntr1)<= 0) )
                        start = pi+start;
        if( ((y1-cntr2) <  0)&&((x1-cntr1) > 0) )
                        start = (2*pi)-start;

        if ( r > 0 )  arc = 2*asin(l/2/r);
        else arc = 2*asin((-l)/2/r);

        /*find step angle*/
        loop = (long)(abs(50*r*arc));
        if( loop > 0 )  step = arc/loop;
        else step = 0;
        for(i=1;i<=loop;i++)
                        { angle = start+(i*step);
        if ( r > 0 ){
        tempX = cntr1 + (r*cos(angle));
        tempY = cntr2 + (r*sin(angle));}
        else{
```

```
        tempX = cntr1 - (r*cos(angle));
        tempY = cntr2 - (r*sin(angle));}
        X1=tempX ;
        X2=tempY ;
        G01(axis_1,axis_2,X1,X2,feed);
                        }
        X1 = x2;
        X2 = y2;
        G01(axis_1,axis_2,X1,X2,feed);
}


void G04(float time)
{
        delay(time);
}


void G73(
double pos1,
double pos2,
double Z,
double q,
double R,
long feedrate)
{
        double I_level;
        int i,d;
        d = 0.05;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
        {
```

```
    G01(0,2,pos1,R,60000);
        i=1
        while ( abs(Z-R) - ((i-1)*(q-d)) > q )
        {
       G01(0,2,pos1,R-(i*(q-d)+i*d),feedrate);
       G01(0,2,pos1,R-(i*(q-d)),60000);
          ++i;
        }
    G01(0,2,pos1,Z,feedrate);
      G01(0,2,pos1,R,60000);
        if (G98)
          G01(0,2,pos1,I_level,60000);
       }
      if (G91)
       {
    G01(0,2,0,R,60000);
        i=1;
        while( abs(Z) - ((i-1)*(q-d)) > q)
        {
              G01(0,2,0,-q,feedrate);
              G01(0,2,0,d,60000);
              ++i;
         }
        }
       G01(0,2,0,Z+((i-1)*(q-d)),feedrate);
     G01(0,2,0,-Z,60000);
        if (G98)
     G01(0,2,0,R,60000);
        }
}

void G83(
double pos1,
double pos2,
double Z,
double q,
```

```
double R,
long feedrate)
{
        double I_level;
        int i,d;
        d = 0.05;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
          {
      G01(0,2,pos1,R,60000);
          i=1
          while ( abs(Z-R) - ((i-1)*(q-d)) > q )
          {
        G01(0,2,pos1,R-(i*(q-d)+i*d),feedrate);
        G01(0,2,pos1,R,60000);
        G01(0,2,pos1,R-(i*(q-d)),60000);
            ++i;
          }
      G01(0,2,pos1,Z,feedrate);
       G01(0,2,pos1,R,60000);
          if (G98)
            G01(0,2,pos1,I_level,60000);
        }
       if (G91)
         {
       G01(0,2,0,R,60000);
          i=1;
          while( abs(Z) - ((i-1)*(q-d)) > q)
          {
                G01(0,2,0,-q,feedrate);
                G01(0,2,0,(i*(q-d)+d),60000);
                G01(0,2,0,(i*(q-d)),60000);
```

```
                ++i;
              }
            }
          G01(0,2,0,Z+((i-1)*(q-d)),feedrate);
        G01(0,2,0,-Z,60000);
          if (G98)
        G01(0,2,0,R,60000);
          }
}

void G81(
double pos1,
double pos2,
double Z,
double R,
long feedrate)
{
        double I_level;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
          {
            G01(0,2,pos1,R,60000);
            G01(0,2,pos1,Z,feedrate);
            G01(0,2,pos1,R,60000);
            if (G98)
              G01(0,2,pos1,I_level,60000);
          }
        if (G91)
          {
            G01(0,2,0,R,60000);
            G01(0,2,0,Z,feedrate);
            G01(0,2,0,-Z,600000);
```

```
            if (G98)
              G01(0,2,0,-R,60000);
          }
}

void G82(
double pos1,
double pos2,
double Z,
double R,
long time,
long feedrate)
{
        double I_level;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
          {
            G01(0,2,pos1,R,60000);
            G01(0,2,pos1,Z,feedrate);
            G04(time)
            G01(0,2,pos1,R,60000);
            if (G98)
              G01(0,2,pos1,I_level,60000);
          }
        if (G91)
          {
            G01(0,2,0,R,60000);
            G01(0,2,0,Z,feedrate);
            G04(time);
            G01(0,2,0,-Z,600000);
            if (G98)
              G01(0,2,0,-R,60000);
```

```
        }
}

void G85(
double pos1,
double pos2,
double Z,
double R,
long feedrate)
{
        double I_level;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
          {
            G01(0,2,pos1,R,60000);
            G01(0,2,pos1,Z,feedrate);
            G01(0,2,pos1,R,feedrate);
            if (G98)
              G01(0,2,pos1,I_level,60000);
          }
        if (G91)
          {
            G01(0,2,0,R,60000);
            G01(0,2,0,Z,feedrate);
            G01(0,2,0,-Z,feedrate);
            if (G98)
               G01(0,2,0,-R,60000);
          }
}

void G86(
double pos1,
```

```
double pos2,
double Z,
double R,
long feedrate)
{
        double I_level;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
          {
            G01(0,2,pos1,R,60000);
            G01(0,2,pos1,Z,feedrate);
            M05;
            G01(0,2,pos1,R,60000);
            if (G98)
              G01(0,2,pos1,I_level,60000);
          }
        if (G91)
          {
            G01(0,2,0,R,60000);
            G01(0,2,0,Z,feedrate);
            M05;
            G01(0,2,0,-Z,600000);
            if (G98)
               G01(0,2,0,-R,60000);
          }
}

void G89(
double pos1,
double pos2,
double Z,
double R,
```

```
long time,
long feedrate)
{
        double I_level;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
          {
            G01(0,2,pos1,R,60000);
            G01(0,2,pos1,Z,feedrate);
            G04(time);
            G01(0,2,pos1,R,feedrate);
            if (G98)
              G01(0,2,pos1,I_level,60000);
          }
        if (G91)
          {
            G01(0,2,0,R,60000);
            G01(0,2,0,Z,feedrate);
            G04(time);
            G01(0,2,0,-Z,feedrate);
            if (G98)
              G01(0,2,0,-R,60000);
          }
}

void G86(
double pos1,
double pos2,
double Z,
double R,
int k,
long feedrate)
```

```
{
        double Z1,I_level,R_point;
        int i;

        G01(0,1,pos1,pos2,60000);
        change_axis(2);
        I_level = get_pos(0)/multiplier;

        if (G90) /* for absolute */
          {
            G01(0,2,pos1,-R,60000);
            if (k==1)
              z1 = Z - R;
            else
            Z1 = (Z-R)/(k-1);
            for (i=1;i<k;i++)
            {
              G01(0,2,pos1,-R-(i*Z1),feedrate);
              M05;
              G01(0,2,pos1,-R,60000);
            }
            if (G98)
              G01(0,2,pos1,I_level,60000);
          }
        if (G91)
          {
            G01(0,2,0,-R,60000);
            if (k==1)
              z1 = Z;
            Z1 = Z/(k-1);
            for (i=1;i<k;i++)
              {
                    G01(0,2,0,-(i*Z1),feedrate);
                    M05;
                    G01(0,2,0,(i*Z1),60000);
              }
```

```
            if (G98)
                G01(0,2,0,R,60000);
            }
}

M03(
float speed)
{       int a;

        set_4(a);
        mtr_off(a);
        set_mtr_cmd(speed);
        synch_prfl(a);
        update(a);
}

M04(
float speed)
{       int a;

        set_4(a);
        mtr_off(a);
        set_mtr_cmd(-speed);
        synch_prfl(a);
        update(a);
}

M05(void);
{       int a;

        set_mtr_cmd(0L);
        update(a);
        mtr_on(a);
        synch_prfl(a);
        update(a);
}
```

Edit . C

```
#include <stdio.h>
#include <stdlib.h>

#define KBD_PORT 0x60

float  X[1000],Y[1000],Z[1000],D[1000],R[1000],P[1000],Q[1000];
int    N[1000],G[1000],M[1000],F[1000],S[1000];
char   x,y,z,n,g,m,r,p,q,k,f,s,dim,star;
int    j=8,line,col,enteri,enterf;
FILE   *fp;

void go() /* return value of position of cursor */
{
        col  = wherex();
        line = wherey();
}

int get_int() /* return integer value of characters which user enter */
{
        char s[10],c;
        int i;

        while(kbhit())getch();
        for(i=0; (c=getchar()) != '\n';i=i+1) s[i]=c;
        if( i < 1 )enteri=1;
        else enteri = 0;
        s[i]='\0';
        return(atoi(s));
}

float get_float() /* return floating point value */
{
        char st[10],c;
        int i;

        while(kbhit())getch();
```

```
        for(i=0; (c=getchar()) != '\n';i=i+1) st[i]=c;
        if( i < 1 )enterf=1;
        else enterf=0;
        st[i]='\0';
        return(atof(st));
}

void print_data(int k) /* print data which user enter on screen */
{
        if( (N[k] >= 1)&&(N[k] != 10000) ) cprintf(" N%d",N[k]);
        if( (G[k] < 10000)&&( N[k] >= 1) ) cprintf(" G%02d",G[k]);
        if( (M[k] < 10000)&&( N[k] >= 1) ) cprintf(" M%02d",M[k]);
        if( X[k] != 10000 ) cprintf(" X%.3f",X[k]);
        if( Y[k] != 10000 ) cprintf(" Y%.3f",Y[k]);
        if( Z[k] != 10000 ) cprintf(" Z%.3f",Z[k]);
        if( D[k] != 10000 ) cprintf(" D%.3f",D[k]);
        if( Q[k] != 10000 ) cprintf(" Q%.3f",Q[k]);
        if( R[k] != 10000 ) cprintf(" R%.3f",R[k]);
        if( S[k] != 10000 ) cprintf(" S%d",S[k]);
        if( P[k] != 10000 ) cprintf(" P%.3f",P[k]);
        if( F[k] != 10000 ) cprintf(" F%d",F[k]);
        if( (N[k] >= 1)&&(N[k]!=10000) ) cprintf(" *");
        clreol();
}

void shift_data(int k) /* shift data from one line to another */
{
        N[k+1] = N[k]+1;
        G[k+1] = G[k];
        M[k+1] = M[k];
        X[k+1] = X[k];
        Y[k+1] = Y[k];
        Z[k+1] = Z[k];
        D[k+1] = D[k];
        R[k+1] = R[k];
        S[k+1] = S[k];
```

```
        P[k+1] = P[k];
        Q[k+1] = Q[k];
        F[k+1] = F[k];
}

void save_data(int k) /* save data into file */
{
        if( (N[k] >= 1)&&(N[k] != 10000) ) fprintf(fp," %c%d",n,N[k]);
        if( (G[k] < 10000)&&( N[k] >= 1) ) fprintf(fp," %c%02d",g,G[k]);
        if( (M[k] < 10000)&&( N[k] >= 1) ) fprintf(fp," %c%02d",m,M[k]);
        if( X[k] != 10000 ) fprintf(fp," %c%.3f",x,X[k]);
        if( Y[k] != 10000 ) fprintf(fp," %c%.3f",y,Y[k]);
        if( Z[k] != 10000 ) fprintf(fp," %c%.3f",z,Z[k]);
        if( D[k] != 10000 ) fprintf(fp," %c%.3f",dim,D[k]);
        if( Q[k] != 10000 ) fprintf(fp," %c%.3f",q,Q[k]);
        if( R[k] != 10000 ) fprintf(fp," %c%.3f",r,R[k]);
        if( S[k] != 10000 ) fprintf(fp," %c%d",s,S[k]);
        if( P[k] != 10000 ) fprintf(fp," %c%.3f",p,P[k]);
        if( F[k] != 10000 ) fprintf(fp," %c%d",f,F[k]);
        if( (N[k] >= 1)&&(N[k]!=10000) ) fprintf(fp," %c\n",star);

}

void  refresh(int direct) /* display data on screen */
{
        int i;

        clrscr();
        for(i=1;i<=8;++i)
        {
          gotoxy(1,i);
          print_data(direct-8+i);
        }
}

void clear(int l) /* clear data in buffer array*/
```

```
{
        G[l] = 10000;
        M[l] = 10000;
        X[l] = 10000;
        Y[l] = 10000;
        Z[l] = 10000;
        D[l] = 10000;
        R[l] = 10000;
        S[l] = 10000;
        P[l] = 10000;
        Q[l] = 10000;
        F[l] = 10000;
}

/* PATTERN OF G_code FUNCTION */
void g00(void)
{
        float temp;

        cprintf(" X");
        go();
        temp=get_float();
        if( enterf!= 1 )X[j]=temp;
        gotoxy(col+8,line);
        cprintf(" Y");
        go();
        temp=get_float();
        if( enterf!=1)Y[j]=temp;
        gotoxy(col+8,line);
        cprintf(" Z");
        go();
        temp=get_float();
        if(enterf!=1)Z[j]=temp;
        gotoxy(col+8,line);
        cprintf("*");
}
```

```
        if( enterf!= 1 )X[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Y");
        go();
        tempf=get_float();
        if( enterf!=1)Y[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Z");
        go();
        tempf=get_float();
        if(enterf!=1)Z[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" R");
        go();
        tempf=get_float();
        if(enterf!=1)R[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" F");
        go();
        tempi=get_int();
        if(enteri!=1)F[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}

void g03()
{
        float tempf;
        int tempi;

        cprintf(" X");
        go();
        tempf=get_float();
        if( enterf!= 1 )X[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Y");
```

```
void g01()
{
        float tempf;
        int tempi;

        cprintf(" X");
        go();
        tempf=get_float();
        if( enterf!= 1 )X[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Y");
        go();
        tempf=get_float();
        if( enterf!=1)Y[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Z");
        go();
        tempf=get_float();
        if(enterf!=1)Z[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" F");
        go();
        tempi=get_int();
        if(enteri!=1)F[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}

void g02()
{
        float tempf;
        int tempi;
        cprintf(" X");
        go();
        tempf=get_float();
```

```
        go();
        tempf=get_float();
        if( enterf!=1)Y[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Z");
        go();
        tempf=get_float();
        if(enterf!=1)Z[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" R");
        go();
        tempf=get_float();
        if(enterf!=1)R[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" F");
        go();
        tempi=get_int();
        if(enteri!=1)F[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}

void g04()
{
        float tempf;

        cprintf(" P");
        go();
        tempf=get_float();
        if( enterf!= 1 )P[j]=abs(tempf);
        gotoxy(col+8,line);
        cprintf("*");
}

void g41()
{
```

```
        float tempf;

        cprintf(" D");
        go();
        tempf=get_float();
        if( enterf!= 1 )D[j]=abs(tempf);
        gotoxy(col+8,line);
        cprintf("*");
}

void g42()
{
        float tempf;

        cprintf(" D");
        go();
        tempf=get_float();
        if( enterf!= 1 )D[j]=abs(tempf);
        gotoxy(col+8,line);
        cprintf("*");
}
void g73()
{
        float tempf;
        int tempi;

        cprintf(" X");
        go();
        tempf=get_float();
        if( enterf!= 1 )X[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Y");
        go();
        tempf=get_float();
        if( enterf!=1)Y[j]=tempf;
        gotoxy(col+8,line);
```

```
        cprintf(" Z");
        go();
        tempf=get_float();
        if(enterf!=1)Z[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Q");
        go();
        tempf=get_float();
        if(enterf!=1)Q[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" R");
        go();
        tempf=get_float();
        if(enterf!=1)R[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" F");
        go();
        tempi=get_int();
        if(enteri!=1)F[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}

void g81()
{
        float tempf;
        int tempi;

        cprintf(" X");
        go();
        tempf=get_float();
        if( enterf!= 1 )X[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Y");
        go();
        tempf=get_float();
```

```
        if( enterf!=1)Y[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Z");
        go();
        tempf=get_float();
        if(enterf!=1)Z[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" R");
        go();
        tempf=get_float();
        if(enterf!=1)R[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" F");
        go();
        tempi=get_int();
        if(enteri!=1)F[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}

void g82()
{
        float tempf;
        int tempi;

        cprintf(" X");
        go();
        tempf=get_float();
        if( enterf!= 1 )X[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Y");
        go();
        tempf=get_float();
        if( enterf!=1)Y[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Z");
```

```
        go();
        tempf=get_float();
        if(enterf!=1)Z[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" R");
        go();
        tempf=get_float();
        if(enterf!=1)R[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" P");
        go();
        tempf=get_float();
        if(enterf!=1)P[j]=abs(tempf);
        gotoxy(col+8,line);
        cprintf(" F");
        go();
        tempi=get_int();
        if(enteri!=1)F[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}

void g83()
{
        float tempf;
        int tempi;

        cprintf(" X");
        go();
        tempf=get_float();
        if( enterf!= 1 )X[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Y");
        go();
        tempf=get_float();
        if( enterf!=1)Y[j]=tempf;
```

```
        gotoxy(col+8,line);
        cprintf(" Z");
        go();
        tempf=get_float();
        if(enterf!=1)Z[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" Q");
        go();
        tempf=get_float();
        if(enterf!=1)Q[j]=tempf;
        gotoxy(col+8,line);
        cprintf(" R");
        go();
        tempf=get_float();
        if(enterf!=1)R[j]=abs(tempf);
        gotoxy(col+8,line);
        cprintf(" F");
        go();
        tempi=get_int();
        if(enteri!=1)F[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}

/* PATTERN OF M_code FUNCTION */
void m03()
{
        int tempi;

        cprintf(" S");
        go();
        tempi=get_int();
        if(enteri!=1)S[j]=abs(tempi);
        gotoxy(col+8,line);
        cprintf("*");
}
```

```
void build(int tempj)
{
        int tempg,t;

        window(3,3,78,12);
        do
        {
          gotoxy(1,9);clreol();
          N[tempj]=tempj-7;
          cprintf(" N%d",N[tempj]);
          while(kbhit())getch();
          cprintf(" G");
          go();
          t=get_int();
          if((enteri != 1) &&((t==00)||(t==01)||(t==02)||(t==03)||
            (t==04)||(t==20)||(t==21)||(t==40)||(t==41)||(t==42)||
            (t==73)||(t==80)||(t==81)||(t==82)||(t==83)||(t==85)||
            (t==86)||(t==89)||(t==90)||(t==91)||(t==98)||(t==99)))
          {
            G[tempj]=t;
            gotoxy(col+2,line);
          }
          else
          {
            gotoxy(col-1,line);
            clreol();
            cprintf("M");
            go();
            t=get_int();
            if( (enteri != 1)&&((t==02)||(t==03)||(t==04)||(t==05)) )
            M[tempj]=t;
            gotoxy(col+2,line);
          }
        }while( (G[tempj]==10000)&&(M[tempj]==10000) );
        switch(G[tempj])
```

```
{
  case 00: g00();
           break;
  case 01: g01();
           break;
  case 02: g02();
           break;
  case 03: g03();
           break;
  case 04: g04();
           break;
  case 41: g41();
           break;
  case 42: g42();
           break;
  case 73: g73();
           break;
  case 81: g81();
           break;
  case 82: g82();
           break;
  case 83: g83();
           break;
  case 85: g81();
           break;
  case 86: g81();
           break;
  case 89: g82();
           break;
  default: break;
}
switch(M[tempj])
{
  case 03: m03();
           break;
  case 04: m03();
```

```
                    break;
            default: break;
        }
}

void Edit()
{
        char file_name[8];
        int  i,c,test,turn,move,tempi,tempj;

        for(i=0;i<=1000;i++)N[i]=10000;
        for(i=0;i<=1000;i++)G[i]=10000;
        for(i=0;i<=1000;i++)M[i]=10000;
        for(i=0;i<=1000;i++)X[i]=10000;
        for(i=0;i<=1000;i++)Y[i]=10000;
        for(i=0;i<=1000;i++)Z[i]=10000;
        for(i=0;i<=1000;i++)D[i]=10000;
        for(i=0;i<=1000;i++)R[i]=10000;
        for(i=0;i<=1000;i++)S[i]=10000;
        for(i=0;i<=1000;i++)P[i]=10000;
        for(i=0;i<=1000;i++)Q[i]=10000;
        for(i=0;i<=1000;i++)F[i]=10000;
        j = 8;
        n ='N';
        g ='G';
        m ='M';
        x ='X';
        y ='Y';
        z ='Z';
        dim ='D';
        r ='R';
        p ='P';
        q ='Q';
        s ='S';
        f ='F';
        star ='*';
```

```
        window(3,3,78,12);
        textcolor(14);
        textbackground(0);
        clrscr();
        gotoxy(1,10);
        cprintf(" Enter program name: ");
        gets(file_name);
        test = 0;
        if((fp=fopen(file_name,"w")) == NULL )
        {
           gotoxy(1,10);clreol();
           cprintf(" Error in creating program!");
           getch();
           gotoxy(1,10);clreol();
        }
        else test = 1;
if( test == 1 )
{
 gotoxy(1,10);clreol();
 cprintf(" Program : %s",file_name);
 do
 {
    i = 0x1c;
    do
    {
        if( i == 0x1c)
        {
          build(j);
          do
          {
            getch();
            i = inportb(KBD_PORT);
          }while( (i!= 0x1c)&&(i!= 0x53) );
        }
        do
        {
```

```
if( i == 0x53 )
{
  go();
  if( line != 9 )
  {
    gotoxy(1,8);
    clreol();
  }
  else
  {
    gotoxy(1,9);
    clreol();
  }
  clear(j);
  build(j);
  getch();
  while((inportb(KBD_PORT)!=0x1c)&&(inportb(KBD_PORT)!=0x53));
  i = inportb(KBD_PORT);
 }
}while( i == 0x53 );
refresh(j);
getch();
i = inportb(KBD_PORT);
if( i != 0x1c )
{
  tempj = j;
  move = j;
  getch();
  i = inportb(KBD_PORT);
  do
  {
    switch(i)
    {
      case 0x48: if(move > 8)refresh(--move);
                 delay(400);
                 break;
```

```
            case 0x50: if(move < j)refresh(++move);
                       delay(400);
                       break;
            case 0x53: do
                       {
                        tempj = j;
                        j = move;
                        go();
                        if( line != 9 )
                        {
                          gotoxy(1,8);
                          clreol();
                        }
                        else
                        {
                          gotoxy(1,9);
                          clreol();
                        }
                        clear(j);
                        build(j);
                  while((inportb(KBD_PORT)!=0x1c)&&(inportb(KBD_PORT)!=0x53))
                        getch();
                        i = inportb(KBD_PORT);
                       }while( i == 0x53 );
                       j = tempj;
                       break;
             case 0x52: refresh(move-1);
                       c = 8;
                       while( N[c] != 10000 ) c = c+1;
                       for(i=c-1;i>=move;i--) shift_data(i);
shift_data(i);
                       j = i+1;
                       do
                       {
                          clear(j);
                          build(j);
```

```
                    /*getch();*/
            while((inportb(KBD_PORT)!=0x1c)&&(inportb(KBD_PORT)!=0x53))
                    getch();
                    i = inportb(KBD_PORT);
                  }while( i == 0x53 );
                  j = tempj+1;
                  tempj = j;
                  break;
          default  : break;
         }
         refresh(move);
         getch();
         i = inportb(KBD_PORT);
        }while( i != 0x1c);
      }
      refresh(j);
      j=j+1;
  }while( (M[j-1] != 02)&&( M[move]!= 02) );
  gotoxy(1,10);clreol();
  cprintf(" Do you want to save?(Y/N): ");
  getch();
  i = inportb(KBD_PORT);
  if( i != 0x15 )
  {
      j = j-1;
      refresh(j-1);
      clear(j);
  }
}while( i != 0x15 );

/* SAVE DATA INTO FILE */
i = 8;
while( N[i] != 10000 )
{
  save_data(i);
  i = i+1;
```

```
    }
    fclose(fp);
   }
   window(3,3,78,12);
   textcolor(14);
   textbackground(0);
   clrscr();
}
```

Jog . C

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include "mc1401.h"
#include "hrdwr.h"
#include "host_io.h"

#define KBD_PORT 0x60

long count;
long keystroke,a,x,y,c,d,speed;

void Jog(void)
{
        long vel,per;
        clrscr();
        reset(a);
        update(a);

        /*AXIS#1*/
        set_1(a);
        axis_on(a);
        mtr_on(a);
        set_output_dac16(a);
        set_prfl_vel(a);
        set_acc(5L);
        set_fltr_pid(a);
        set_kp(10L);
        set_ki(10L);
        set_kd(10L);
        zero_pos(a);
        synch_prfl(a);
        set_lmt_sense(0L);
        lmts_on(a);
        set_capt_home(a);
```

```
update(a);

/*AXIS#2*/
set_2(a);
axis_on(a);
mtr_on(a);
set_output_dac16(a);
set_prfl_vel(a);
set_acc(5L);
set_fltr_pid(a);
set_kp(10L);
set_ki(10L);
set_kd(10L);
zero_pos(a);
synch_prfl(a);
set_capt_home(a);
update(a);

/*MANUAL CONTROL*/
clrscr();
printf("set max speed(NOT MORE THAN 10000) = ");scanf("%ld",&vel);
speed = vel;
gotoxy(1,3);printf("value=%x",get_lmt_swtch(a));
keystroke = inportb(KBD_PORT);
y = 1;
d = 1;
do
{
   keystroke = inportb(KBD_PORT);
   switch( keystroke )
   {
     case 0x3b:
      set_1(a);
      mtr_off(a);
      update(a);
      if ( get_lmt_swtch(a) != 2 )
```

```
{
      clr_status(a);
      update(a);
}
gotoxy(1,2);printf("AXIS#1  LEFT\n");
gotoxy(1,3);printf("value=%ld",get_lmt_swtch(a));
gotoxy(1,4);printf("value of int=%ld",get_status(a));
do
{
      keystroke = inportb(KBD_PORT);
      set_mtr_cmd(-speed);
      synch_prfl(a);
      update(a);
}while( keystroke != 0xbb );
if ( d != get_capt(a)) c = get_capt(a);
set_mtr_cmd(0L);
update(a);
mtr_on(a);
synch_prfl(a);
update(a);
break;

case 0x3c:
set_1(a);
mtr_off(a);
update(a);
if( get_lmt_swtch(a) != 1 )
{
      clr_status(a);
      update(a);
}
gotoxy(1,2);printf("AXIS#1 RIGHT\n");
gotoxy(1,3);printf("value=%ld",get_lmt_swtch(a));
gotoxy(1,4);printf("value of int=%ld",get_status(a));
if( c == get_capt(a))
{
```

```
        c = get_capt(a);
        do
        {
           keystroke = inportb(KBD_PORT);
           set_mtr_cmd(speed);
           synch_prfl(a);
           update(a);
        }while( (keystroke != 0xbc)&&( c == get_capt(a)));
        if( c != get_capt(a) )
        {
           set_mtr_cmd(0L);
           update(a);
           mtr_on(a);
           synch_prfl(a);
           update(a);
           do
           {
              keystroke = inportb(KBD_PORT);
           }while((keystroke != 0x3b)&&(keystroke !=0x3d)&&
                    (keystroke != 0x3e)&&(keystroke!=0x1b));
        }
        d = get_capt(a);
    }
    set_mtr_cmd(0L);
    update(a);
    mtr_on(a);
    synch_prfl(a);
    update(a);
    break;

   case 0x3d:
    set_2(a);
    mtr_off(a);
    update(a);
    gotoxy(1,2);printf("AXIS#2  LEFT\n");
    gotoxy(1,3);printf("value=%ld",get_lmt_swtch(a));
```

```
gotoxy(1,4);printf("value of int=%ld",get_status(a));
gotoxy(1,6);printf("HOME = %ld",x);
if(get_lmt_swtch(a) != 8 )
{
    clr_status(a);
    update(a);
}
if( x == get_capt(a) )
{
    x = get_capt(a);
    do
    {
      keystroke = inportb(KBD_PORT);
      set_mtr_cmd(-speed);
      synch_prfl(a);
      update(a);
    }while( (keystroke != 0xbd) && (x == get_capt(a)));
    if( x != get_capt(a) )
    {
      set_mtr_cmd(0L);
      update(a);
      mtr_on(a);
      synch_prfl(a);
      update(a);
      do
      {
        keystroke = inportb(KBD_PORT);
      }while((keystroke != 0x3e)&&(keystroke !=0x3c)&&
            (keystroke != 0x3b)&&(keystroke!=0x1b));}
      y = get_capt(a);
    }
    set_mtr_cmd(0L);
    update(a);
    mtr_on(a);
    synch_prfl(a);
    update(a);
```

```
            break;
            clr_prfl(a);
            update(a);
            synch_prfl(a);
            update(a);
            mtr_on(a);
            update(a);
            keystroke = inportb(KBD_PORT);
            count = getch();
    }while ( count != 0x1b );
    set_1(a);
    mtr_on(a);
    synch_prfl(a);
    update(a);
    set_2(a);
    mtr_on(a);
    synch_prfl(a);
    update(a);
}
```

กิตติกรรมประกาศ

ขอขอบคุณ อ. กวิน สนธิเพิ่มพูน และ อ. จำลอง ปราบแก้ว ที่ได้ให้คำแนะนำและช่วยเหลือในการทำ Project

ขอขอบคุณพี่ SPOT ที่ช่วยอำนวยความสะดวกในเรื่องอุปกรณ์และสถานที่ทำงาน

ขอขอบคุณท่านอาจารย์ทุกท่านที่ถ่ายทอดความรู้และประสบการณ์ให้กับพวกเรา

ขอขอบคุณพี่มณฑาสำหรับความสะดวกในการใช้ห้องทำ Project และการจัดหาอุปกรณ์

ขอขอบคุณเพื่อนๆทุกคนที่ให้กำลังใจซึ่งกันและกันอย่างไม่ขาดสาย

ขอขอบคุณคุณพ่อ คุณแม่ ที่คอยห่วงใยและให้กำลังใจเสมอมา

ขอขอบคุณพลังสติปัญญาของสมาชิกในกลุ่มทุกคนที่ช่วยกันผลักดันให้รายงานฉบับนี้เสร็จสมบูรณ์ออกมาได้