การวางแผนกระบวนการผลิตโดยใช้คอมพิวเตอร์ช่วยสำหรับเครื่องจักรเจาะรูแบบซีเอ็นซีเทอร์เรตพันช์

นาย นราธิป วีระกิจพานิช

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2543

ISBN 974-346-236-8

COMPUTER-AIDED PROCESS PLANNING FOR THE CNC TURRET PUNCH PRESS MACHINE

MR. NARATHIP WEERAKITPANICH

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Industrial Engineering

Department of Industrial Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2000
ISBN 974-346-236-8

หัวข้อวิทยานิพนธ์	การวางแผนกระบวนการผลิตโดยใช้คอมพิวเตอร์ช่วยสำหรับเครื่องจักรเจาะรูแบบซีเอ็นซีเทอร์เรตพันช์
โดย	นาย นราธิป วีระกิจพานิช
ภาควิชา	วิศวกรรมอุตสาหการ
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ปารเมศ ชุติมา
อาจารย์ที่ปรึกษาร่วม (ถ้ามี)	-

คณะวิศกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ จรูญ มหิทธาฟองกุล)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.ปารเมศ ชุติมา)

..... อาจารย์ที่ปรึกษาร่วม (ถ้ามี)
( )

..... กรรมการ
(รองศาสตราจารย์ สมชาย พวงเพิกศึก)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ จิรพัฒน์ เงาประเสริฐวงศ์)

บทคัดย่อวิทยานิพนธ์

นราธิป วีระกิจพานิช : การวางแผนกระบวนการผลิตโดยใช้คอมพิวเตอร์ช่วยสำหรับเครื่องจักรเจาะรูแบบซีเอ็นซีเทอร์เรตพันช์ (COMPUTER-AIDED PROCESS PLANNING FOR THE CNC TURRET PUNCH PRESS MACHINE) อ. ที่ปรึกษา : ผศ. ดร.ปารเมศ ชุติมา, 170 หน้า. ISBN 974-346-236-8.

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาซอฟต์แวร์คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตสำหรับกระบวนการเจาะรูของเครื่องจักร CNC Turret Punch Press ที่ใช้ในงานเจาะรูแผ่นโลหะในการผลิตกล่องโลหะ ซึ่งทำหน้าที่ในการสร้างรหัสโปรแกรม G Code เพื่อป้อนเข้าเครื่องจักร CNC โดยการใช้ซอฟต์แวร์ร่วมกับ Drawing รูปแบบ 2 มิติในโปรแกรม AutoCAD R14 ที่ถูกสร้างขึ้นให้เหมาะสมกับซอฟต์แวร์ หลักการในการทำงานเป็นการสร้างรหัส G Code สำหรับรูปของรูต่าง ๆ ที่ผู้วางแผนได้เลือกทีละรูป การสร้างรหัส G Code จะสร้างขึ้นทีละบรรทัดโดยอาศัยข้อมูลของรูปรูที่ได้เลือกอันได้แก่ ลักษณะ ตำแหน่งและขนาดรู และ ข้อมูลภายในของคำสั่ง G Code ในการเจาะที่ผู้วางแผนต้องการใช้เจาะรูลักษณะต่าง ๆ เช่น คำสั่งที่เป็นรูปแบบในการเจาะ ตำแหน่งอ้างอิงของคำสั่ง หัวเจาะที่ใช้เจาะและขนาดของหัวเจาะ เป็นต้น เพื่อที่จะสร้าง G Code สำหรับการเจาะภายในและการเจาะภายนอกชิ้นงาน แล้วนำรหัสดังกล่าวมาจัดเรียงและรวมกันเป็นรหัสโปรแกรม G Code ที่พร้อมสำหรับป้อนเข้าเครื่องจักร CNC เพื่อทำการเจาะชิ้นงาน โดยที่เมื่อเปรียบเทียบกับการวางแผนกระบวนการผลิตแบบเดิมนั้น การนำซอฟต์แวร์เข้ามาช่วยสามารถช่วยลดเวลาในการสร้างรหัสโปรแกรม G Code ลดความผิดพลาดของรหัสโปรแกรม G Code และลดภาระงานที่เกิดขึ้นกับผู้วางแผนได้อย่างน่าพอใจ

ภาควิชา วิศวกรรมอุตสาหการ
สาขาวิชา วิศวกรรมอุตสาหการ
ปีการศึกษา 2543

ลายมือชื่อนิสิต .....
ลายมือชื่ออาจารย์ที่ปรึกษา .....
ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

AN ABSTRACT

4170359021 : MAJOR INDUSTRIAL ENGINEERING

KEY WORD: COMPUTER-AIDED PROCESS PLANNING / COMPUTER NUMERICAL CONTROL

NARATHIP WEERAKITPANICH : COMPUTER-AIDED PROCESS PLANNING FOR THE CNC TURRET PUNCH PRESS MACHINE. THESIS ADVISOR : ASST. PROF. PARAMES CHUTIMA, Ph.D., THESIS COADVISOR : -, 170 pp. ISBN 974-346-236-8.

This thesis describes the development of computer-aided process planning for the CNC turret punch press machine which is used for punching metal sheets. The objective of the software is to help a process planner generating G code, one of the input-data formats used in CNC manufacturing, by using the software with a two-dimension drawing, which is created in a software-required format, in AutoCAD R14. In generating each line of G code, the software processes the information of each object in such drawing selected by a user and the user-input information for a specific punching format. The information of the selected object tells the software about the shape, position, and size of such object, and the user-input information tells it about punching format and required datas used for each punching format. From these information, the software continuously generates G code for every objects and collects every lines of G code in a text file which can be used for inside and outside punching for CNC metal sheet punching.

The result shows that when compared to the old process planning planning without CAPP software, the new one with this software can reduce process planning time in generating G code, reduce errors in G code, and process planner's work load is reduced.

ภาควิชา วิศวกรรมอุตสาหการ
สาขาวิชา วิศวกรรมอุตสาหการ
ปีการศึกษา 2543

ลายมือชื่อนิสิต .....
ลายมือชื่ออาจารย์ที่ปรึกษา .....
ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของ ผศ. ดร. ปารเมศ ชุติมา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้ให้คำแนะนำและข้อคิดเห็นต่าง ๆ ในงานวิจัยมาด้วยดีตลอด

ขอขอบพระคุณ บริษัท เอ เอ็ม พี เมทัลเวอคส์ จำกัด ที่สนับสนุนในด้านของเครื่องจักรและข้อมูลต่าง ๆ รวมถึงให้ยืมอุปกรณ์คอมพิวเตอร์ที่ใช้ในการดำเนินงาน

ขอขอบพระคุณ คุณมนูศักดิ์ จันทน์ทอง คุณณรงค์ อมรพิทักษ์พันธ์ คุณกิตติพงศ์ อมรพิทักษ์พันธ์ คุณศราวุธ บุญวัฒน์ และเพื่อน ๆ ทุกท่านที่ได้ให้ความช่วยเหลือและกำลังใจในการทำงานวิจัยนี้ด้วยดีเสมอมา และเนื่องจากทุนการวิจัยครั้งนี้บางส่วนได้รับมาจากทุนอุดหนุนวิจัยของบัณฑิตวิทยาลัย จึงขอขอบคุณบัณฑิตวิทยาลัยมา ณ ที่นี้ด้วย

ท้ายนี้ ผู้วิจัยใคร่ขอกราบขอบพระคุณ บิดา มารดา ซึ่งสนับสนุนในด้านการเงินและให้ความห่วงใยและกำลังใจเสมอมาจนสำเร็จการศึกษา

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1

บทนำ

1.1 บทนำ

การวางแผนกระบวนการผลิต เป็นกิจกรรมหนึ่งซึ่งมีความสำคัญในการออกแบบและผลิตผลิตภัณฑ์ ซึ่งมีความหมายที่เกี่ยวข้องกับหน้าที่ในการวางแผนและเตรียมรายละเอียดต่าง ๆ ของกระบวนการการทำงานในการเปลี่ยนงานออกแบบทางวิศวกรรมให้เป็นผลิตภัณฑ์ที่สมบูรณ์ ส่วนประกอบในรายละเอียดของการวางแผนกระบวนการผลิต มีตัวอย่างเช่น การเลือกชนิดของกระบวนการผลิต เครื่องมือในการเจาะ ตัด กลึง ไส เป็นต้น

สำหรับในปัจจุบันและในหลายปีที่ผ่านมา สภาพของตลาดจะมีความเปลี่ยนแปลงอย่างรวดเร็ว และมีการแข่งขันเพื่อที่จะทำให้คุณภาพของผลิตภัณฑ์สูงขึ้นและมีความหลากหลาย อันส่งผลให้สินค้าที่ผลิตออกมาจำเป็นจะต้องมีการเปลี่ยนแปลงรูปแบบบ่อยครั้งกว่า และจำนวนผลิตภัณฑ์ที่ผลิตต่อชุดน้อยลงกว่าในอดีต ด้วยสภาพดังกล่าวทำให้การวางแผนกระบวนการผลิตจะต้องเป็นกิจกรรมที่เกิดขึ้นบ่อยครั้ง ผู้วางแผนจำเป็นจะต้องทำกิจกรรมดังกล่าวซ้ำซากและจำเป็นที่จะทำอย่างรวดเร็ว ดังนั้นการนำระบบคอมพิวเตอร์เข้ามาช่วยวางแผนกระบวนการผลิต (Computer-Aided Process Planning) จึงเป็นเครื่องมือที่ได้รับความสนใจเพิ่มมากขึ้นเรื่อย ๆ ซึ่งจะทำให้เวลาที่ใช้ในการวางแผนลดลง ส่งผลให้ต้นทุนลดลง และยังทำให้เกิดความแน่นอนในการทำงานเพิ่มขึ้นอีกด้วย

ระบบคอมพิวเตอร์ที่นำมาช่วยจะมีรูปแบบหลากหลาย ซึ่งก็ขึ้นอยู่กับสภาพของงานแต่ละแบบที่ต้องการวางแผนกระบวนการผลิต ระบบดังกล่าวจึงมีอยู่หลากหลายและสามารถสร้างขึ้นโดยการประยุกต์วิธีการต่าง ๆของเทคโนโลยีด้านคอมพิวเตอร์ที่เกี่ยวข้องกับวิศวกรรม อาทิ เทคนิคเช่น ระบบผู้เชี่ยวชาญ (Expert System) ระบบปัญญาประดิษฐ์ (Artificial Intelligent) เป็นต้น

สำหรับงานทางด้านการออกแบบผลิตภัณฑ์ทางด้านวิศวกรรม ระบบคอมพิวเตอร์ที่มีบทบาทสำคัญที่เกี่ยวข้องได้แก่ ระบบคอมพิวเตอร์ช่วยออกแบบ (CAD) ซึ่งมีความสามารถในการช่วยในการเขียนรูปแบบในการออกแบบ รวมถึงการเป็นฐานข้อมูลที่สำคัญที่จะใช้ในการผลิตสำหรับเครื่องจักร CNC การพัฒนาโปรแกรมคอมพิวเตอร์เพื่อช่วยให้เกิดการใช้ข้อมูลจาก

ระบบคอมพิวเตอร์ช่วยออกแบบไปยังการผลิตที่เครื่องจักร CNC จึงถือเป็นส่วนหนึ่งในการนำคอมพิวเตอร์มาใช้ช่วยในการวางแผนกระบวนการผลิต ซึ่งจะอยู่ในรูปแบบของการใช้เทคนิคระบบผู้เชี่ยวชาญรูปแบบหนึ่ง

1.2 ความสำคัญของปัญหา

โรงงานที่ได้เข้ามาทำการศึกษาการวางแผนกระบวนการผลิตเป็นโรงงานผลิตแผ่นเหล็กที่ใช้ในการประกอบกล่องโลหะ โดยกล่องโลหะที่เป็นผลิตภัณฑ์จะเป็นกล่องที่สำหรับครอบอุปกรณ์ไฟฟ้า และคอมพิวเตอร์ เป็นต้น เครื่องจักร CNC ในการศึกษานี้เป็นเครื่อง CNC Turret Punch Press รุ่น CENTRUM 2500 โดย Murata Wiedemann เครื่องจักร CNC นี้มีหน้าที่โดยตรงในการเจาะรูของแผ่นโลหะที่จะนำมาประกอบเป็นกล่อง โดยเครื่องจักรจะมีหัวเจาะซึ่งวางอยู่บนตัวยึดซึ่งมีลักษณะเป็นแท่นหมุน การเจาะจะอาศัยการเคลื่อนที่ของแผ่นโลหะที่ถูกจับยึดไว้บนแท่นด้านล่าง การทำงานของเครื่องจักรนี้จะอาศัยข้อมูลที่อยู่ในรูปแบบตัวอักษรที่ทางผู้วางแผนกระบวนการผลิตได้ป้อนเข้าไป

กระบวนการการวางแผนกระบวนการผลิตของทางบริษัทจะเริ่มจากที่ได้รับรูปแบบมาจากลูกค้า ซึ่งอยู่ในรูปแบบของ Drawing ในกระดาษและในรูปแบบไฟล์ของ AutoCAD ซึ่งในปัจจุบันมีการสั่งจากลูกค้าในลักษณะแบบหลังเพิ่มมากขึ้น ผู้ทำการวางแผนกระบวนการผลิตจำเป็นจะต้องแปลงและคลี่แบบที่ได้รับให้อยู่ในรูปแบบของแผ่นเหล็กที่มีรูอยู่ตามที่ต้องการ การป้อนข้อมูลเพื่อให้เครื่องจักรทำงานจะทำโดยการอ่านแบบที่ได้จากการแปลงและคลี่แล้วนำมาเขียนเป็นรูปแบบอักษรตามลักษณะที่ทางผู้ผลิตเครื่องจักรได้กำหนดไว้ ข้อมูลที่ป้อนจะมีรายละเอียดของปัจจัย อันได้แก่ ตำแหน่งของแผ่นโลหะในรูปแบบของพิกัด ชนิดของหัวเจาะ ความเร็วของหัวเจาะ และลักษณะการเคลื่อนที่ของแท่นจับ ปัญหาที่พบบ่อยครั้งสำหรับผู้วางแผนกระบวนการผลิต ได้แก่ ความผิดพลาดในการป้อนข้อมูลด้านปัจจัยต่าง ๆ รวมถึงความเมื่อยล้าที่เกิดขึ้นจากการทำงานที่ซ้ำซากและมีรายละเอียดค่อนข้างสูง

การพัฒนาโปรแกรมคอมพิวเตอร์เพื่อช่วยในการทำงานนี้จะมีประโยชน์ในการลดปัญหาเหล่านี้ที่จะเกิดขึ้น โปรแกรมคอมพิวเตอร์จะทำการนำข้อมูลที่อยู่ในรูปแบบของ CAD แปลงให้อยู่ในรูปแบบของตัวอักษรที่มีรูปแบบตามที่กำหนดไว้ ซึ่งพร้อมที่จะคัดลอกไปยังเครื่อง CNC

1.3 วัตถุประสงค์ของงานวิจัย

เพื่อพัฒนาซอฟต์แวร์คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตสำหรับกระบวนการเจาะรูของเครื่องจักร CNC Turret Punch Press ที่ใช้ในงานเจาะรูแผ่นเหล็กที่ใช้ในการผลิตกล่องโลหะ

1.4 ขอบเขตของงานวิจัย

- พัฒนาซอฟต์แวร์ที่ใช้ช่วยในการวางแผนกระบวนการผลิตสำหรับเครื่องจักร CNC Turret Punch Press เท่านั้น
- ซอฟต์แวร์จะครอบคลุมถึงหน้าที่ในการป้อนคำสั่งสำหรับปัจจัยทั้งหมดที่จำเป็นในการเจาะรูของเครื่องจักรอันได้แก่ ตำแหน่งของแผ่นโลหะในรูปแบบของพิกัด ชนิดของหัวเจาะ ความเร็วของหัวเจาะ และลักษณะการเคลื่อนที่ของแท่นจับ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1. แสดงการประยุกต์โปรแกรม AutoCAD เพื่อใช้ช่วยในการวางแผนกระบวนการผลิตในกระบวนการเจาะรู

1.5.2. ช่วยลดเวลาสำหรับกระบวนการวางแผนกระบวนการผลิต และลดความผิดพลาดและของเสียในการผลิต

1.5.3. ช่วยลดความล้าและภาระที่เกิดกับผู้วางแผนกระบวนการผลิต

1.6 ขั้นตอนการศึกษาและวิจัย

ขั้นตอนในการพัฒนาซอฟต์แวร์คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตมีทั้งหมด 7 ขั้นตอน ได้แก่

1.6.1. ศึกษาหลักการ ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

1.6.2. เก็บและจัดเตรียมข้อมูลของเครื่องจักรและลักษณะรูปแบบที่ใช้สั่งงานเครื่องจักร

1.6.3. กำหนดโครงสร้างและองค์ประกอบของซอฟต์แวร์

1.6.4. สร้างและพัฒนาโปรแกรม

1.6.5. ทดสอบความถูกต้องของซอฟต์แวร์คอมพิวเตอร์

1.6.6. สรุปและวิเคราะห์

1.6.7. จัดทำรูปเล่มวิทยานิพนธ์

1.7 สรุปเนื้อหา

บทที่ 2 งานวิจัยที่เกี่ยวข้อง กล่าวถึง งานวิจัยต่างที่เกี่ยวข้องกับการนำระบบคอมพิวเตอร์เข้ามาช่วยในการวางแผนกระบวนการผลิต

บทที่ 3 คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิต กล่าวถึง ทฤษฎีของการวางแผนกระบวนการผลิต การนำคอมพิวเตอร์เข้ามาช่วยในการด้านออกแบบ การผลิตและการวางแผนกระบวนการผลิต รวมทั้งโปรแกรม AutoCAD และการโปรแกรมมิ่งใน AutoCAD

บทที่ 4 การวางแผนกระบวนการผลิตสำหรับการเจาะรูสำหรับเครื่องจักร CNC Turret Punch Press กล่าวถึง ข้อมูลเกี่ยวกับเครื่องจักร CNC Turret Punch Press ทางด้านส่วนประกอบและรูปแบบของการสั่งการโดยใช้รหัสโปรแกรม G Code ลักษณะต่าง ๆ ปัญหาที่เกิดจากการวางแผนกระบวนการผลิตแบบเดิมและจุดที่ได้นำซอฟต์แวร์เข้ามาช่วยในการสร้างรหัสโปรแกรม G Code

บทที่ 5 โปรแกรมคอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตสำหรับการเจาะรูสำหรับ CNC Turret Punch Press กล่าวถึง ลักษณะทั่วไป โครงสร้างการทำงาน และรายละเอียดของส่วนประกอบของกลไกหลักในการทำงานของซอฟต์แวร์ ZAmpPunch.arx

บทที่ 6 การทดสอบความถูกต้องของซอฟต์แวร์ กล่าวถึง การทดสอบความถูกต้องของซอฟต์แวร์โดยนำชิ้นงานทดสอบมาทำการเจาะโดยใช้ซอฟต์แวร์ช่วย โดยอธิบายถึงขั้นตอนในการใช้งานในการเจาะสำหรับรูปแบบคำสั่งต่าง ๆ และแสดงผลของการเจาะแผ่นโลหะจริงสำหรับทดสอบ

บทที่ 7 การเปรียบเทียบการวางแผนกระบวนการผลิตแบบเดิมและการนำซอฟต์แวร์เข้ามาช่วย กล่าวถึง การเปรียบเทียบเวลาที่ผู้วางแผนกระบวนการผลิตใช้ในการสร้างรหัสโปรแกรม G Code เปรียบเทียบกับการที่ใช้ซอฟต์แวร์ช่วย และแสดงความพึงพอใจของผู้วางแผนที่มีต่อการใช้ซอฟต์แวร์

บทที่ 8 บทสรุปและข้อเสนอแนะ กล่าวถึง การสรุปผลงานวิจัยและข้อเสนอแนะที่มีต่องานวิจัยทั้งหมด

บทที่ 2

งานวิจัยที่เกี่ยวข้อง

เนื้อหาในบทนี้กล่าวถึงงานวิจัยที่เกี่ยวข้องกับการพัฒนาระบบคอมพิวเตอร์สำหรับช่วยในการวางแผนกระบวนการผลิต ซึ่งเกี่ยวข้องกับการนำคอมพิวเตอร์ช่วยการออกแบบมาประยุกต์เป็นระบบที่ใช้ช่วยในกระบวนการวางแผนกระบวนการผลิต

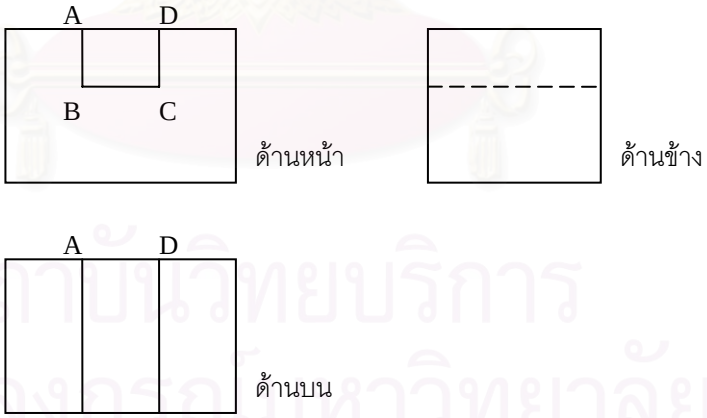
2.1. งานวิจัยที่เกี่ยวข้องกับระบบคอมพิวเตอร์ช่วยวางแผนกระบวนการผลิต

- **กสานต์ ปิ่นเวหาส์ (1998)** เสนอการพัฒนาซอฟต์แวร์สำหรับช่วยวางแผนกระบวนการผลิตสำหรับงานแมชชีนนิ่งชิ้นส่วนเครื่องยนต์ ซึ่งมีรูปทรงเป็นรูปหลายเหลี่ยมและประกอบไปด้วยรูปทรงทางการผลิตชนิดต่าง ๆ เช่น รูป ผิวปาดหน้า และผิวการปรับปรุงรูชนิดต่าง ๆ ซึ่งโปรแกรมที่พัฒนาขึ้นสามารถวางแผนกระบวนการผลิตได้อย่างอัตโนมัติ โดยอาศัยข้อมูลการออกแบบที่ได้มาจากการปรับปรุงบางส่วนของ AutoCAD ขั้นตอนการวางแผนเริ่มจากผู้วางแผนทำการกำหนดลักษณะรูปร่างการผลิต ขนาด และข้อมูลทางด้านเทคนิคให้กับพื้นผิวต่าง ๆ ที่ต้องการทำการแมชชีนนิ่ง จากนั้นพื้นผิวที่ทำการออกแบบจะถูกจัดกลุ่มตามทิศทางของการแมชชีนนิ่ง และทำการเลือกชนิดกระบวนการผลิต การเลือกเครื่องมือตัด การกำหนดค่าสภาวะเฉือน และการคำนวณเวลาที่ใช้ในการแมชชีนนิ่ง การพิจารณาแผนกระบวนการผลิตที่เหมาะสมที่สุดสำหรับแต่ละพื้นผิวจะพิจารณาจากจำนวนขั้นตอนการทำงานที่น้อยที่สุดและอัตราการผลิตสูงสุด

- **Rumana (1995)** เสนอเกี่ยวกับระบบการวางแผนกระบวนการผลิตแบบอัตโนมัติสำหรับการแมชชีนนิ่งส่วนประกอบในลักษณะหมุน โปรแกรมคอมพิวเตอร์ส่วนของ interface จะถูกพัฒนาขึ้นสำหรับข้อมูลที่เกี่ยวข้องกับ CAD และ CAPP เอาไว้ผู้ออกแบบกระบวนการผลิตสามารถป้อนลักษณะในการผลิตต่าง ๆ รวมถึงข้อมูลด้าน geometric โปรแกรมถูกพัฒนาให้อยู่ในรูปแบบของ Object-Orientation Environment ในรูปของ module หน้าที่ของโปรแกรมช่วยวางแผนกระบวนการผลิตที่พัฒนาขึ้นประกอบไปด้วย การประมวลผล file DXF ใน AutoCAD หลังจากที่ได้สร้างขึ้น การวางแผนในการ setup เครื่องจักร การเลือกเครื่องมือในการแมชชีนนิ่ง การเลือกตัวจับยึดเครื่องมือ และการคำนวณเวลาในการแมชชีนนิ่ง

- **Legoff และ Hascoet (1998)** พัฒนาโปรแกรมสำหรับการวางแผนกระบวนการเชื่อมที่เชื่อมโดยเครื่องจักรหุ่นยนต์ โดยอาศัยข้อมูลจากฐานข้อมูลรูปทรงชิ้นงานจากระบบคอมพิวเตอร์ช่วยออกแบบ โปรแกรมจะกำหนดเส้นทางการเชื่อมชิ้นงานให้โดยอัตโนมัติ และอาศัยเทคนิคโครงข่ายประสาทเทียม จากงานวิจัยแสดงให้เห็นว่าข้อมูลการออกแบบจาก CAD สามารถถูกนำมาใช้ในการวางแผนกระบวนการเชื่อมและคำนวณค่าพารามิเตอร์ของการเชื่อมได้อย่างอัตโนมัติ

- **Linardakis และ Mileham (1992)** ออกแบบตัวแปลข้อมูลจากระบบคอมพิวเตอร์ช่วยในการออกแบบซึ่งอาศัยข้อมูลจากแฟ้มข้อมูลแบบชิ้นงานที่มีรูปแบบเป็นมาตรฐานสำหรับอุตสาหกรรม (DXF File) เพื่อทำการถอดข้อความ และจำแนกข้อมูลสำหรับการวางแผนกระบวนการผลิตจากแบบทางวิศวกรรมซึ่งวาดโดยระบบคอมพิวเตอร์ช่วยในการออกแบบ (CAD) ชิ้นงานทรงเหลี่ยมจะถูกวาดอยู่ในรูปของภาพฉายทั้ง 3 ด้านของชิ้นงาน จากนั้นจะได้อาศัยอัลกอริทึมในการถอดข้อความและจำแนกรูปทรงย่อยต่างๆ (เช่น เส้นตรง วงกลม และส่วนโค้ง เป็นต้น) ที่ถูกวาดในภาพฉายทั้ง 3 ด้านให้เป็นรูปทรงสำหรับการผลิตชนิดต่างๆ เช่น จากรูปที่ 2.1 รูปทรงการกัดร่อง (Slot) หากพบการไม่ต่อเนื่องของเส้นขอบบนของภาพฉายแสดงด้านหน้า (A) และพบเส้นต่อเนื่องที่ตั้งฉากกับจุดที่ไม่ต่อเนื่องนั้นอีก 3 เส้น (AB, BC, และ CD) จากนั้นทำการตรวจสอบภาพฉายแสดงด้านบนหากพบเส้นสองเส้นที่เริ่มจากจุด A และ D โดยไปสิ้นสุดที่เส้นขอบล่างสุด แสดงว่ามีรูปร่างการกัดร่อง



รูปที่ 2.1 *ตัวอย่างรูปทรงเซาะร่อง (Slot)*

ในชิ้นงาน เป็นต้น จะเห็นได้ว่าระบบนี้สามารถแปลข้อมูลกับลักษณะรูปทรงการผลิตที่อยู่ภายนอกเท่านั้น

- **Wang และ Wysk (1987)** ได้พัฒนาระบบ Turbo-CAPP ซึ่งเป็นผลงานที่เกี่ยวกับการสร้างแผนกระบวนการผลิตใหม่แบบอัตโนมัติ (Generative Approach) ที่นำสู่ทิศทางการเชื่อมต่อกันของระบบ CAD และ CAM ระบบนี้ช่วยในการสร้างรหัสโปรแกรมของระบบ NC สำหรับส่วนประกอบที่มีลักษณะหมุนโดยใช้ข้อมูลนำเข้าของ CAD ระบบนี้ใช้การตีความจากข้อมูลที่เป็นลักษณะ DXF ของ AutoCAD ซึ่งมีกลไกในการตีความที่ค่อนข้างซับซ้อน Turbo-CAPP มีความสามารถดังนี้ (1) แปลความหมายของข้อมูลด้านรูปร่างของ Drawing ในระนาบสองมิติครึ่งได้ (2) ตรวจสอบความสม่ำเสมอของการให้ขนาดและพิกัดเผื่อ (3) จัดการฐานความรู้ด้านการผลิตและสามารถเชื่อมต่อกับผู้วางแผนกระบวนการผลิตที่มีประสบการณ์ และ (4) ใช้กระบวนการให้เหตุผลซึ่งขึ้นอยู่กับลักษณะพื้นผิวชิ้นงานและความสัมพันธ์ที่เกี่ยวกับการแต่งผิว การให้ขนาดและพิกัดเผื่อ ค่าคอนฟิกกูเรชั่นต่าง ๆ ของเครื่องจักร รวมทั้งเครื่องมือกลที่มีอยู่ เพื่อที่จะสร้างแผนกระบวนการผลิตที่แตกต่างกันออกไป

- **Davies, Joseph และ Kalta (1990 และ 1991)** พัฒนาระบบ EXCAP ซึ่งเป็นระบบฐานความรู้ในการสร้างแผนกระบวนการผลิตใหม่แบบอัตโนมัติ (Generative Approach) สำหรับส่วนประกอบที่มีลักษณะหมุนโดยใช้ระบบง่าย ๆ ที่เขียนขึ้นโดยภาษา BASIC ข้อมูลเข้าของระบบคือแบบจำลองทาง AutoCAD ของส่วนประกอบที่ต้องการและผลที่ได้จากระบบได้แก่รหัสโปรแกรมของระบบ NC ระบบมีจุดมุ่งหมายที่จะผลิตและตรวจสอบส่วนประกอบอย่างอัตโนมัติจากแผนกระบวนการผลิตที่ดีที่สุดและรหัสโปรแกรม NC จะถูกสร้างขึ้นจากการทำงานร่วมกันของระบบ EXCAP และระบบ TECHTURN ซึ่งเป็นระบบที่ใช้ช่วยในการเลือกเครื่องมือและสภาวะการตัดเฉือนที่เหมาะสม โดยระบบ TECHTURN ได้ถูกพัฒนาขึ้นโดย HINDUJA (1986)ซึ่งการทำงานจะใช้การวิเคราะห์ชิ้นงานที่ถูกนำเสนอแบบ IGESของซอฟต์แวร์ด้านCAD ซึ่งซอฟต์แวร์ที่ใช้นั้นไม่ได้มีการระบุไว้

- **Schulte, Padmanabhan และ Devgum (1992)** เสนอวิธีการที่จะนำระบบ CAD และ CAM มาผนวกรวมกันในใช้สำหรับแบบจำลองแบบ Wireframe ของชิ้นส่วนที่เป็นรูปเหลี่ยมผ่านโปรแกรม AutoCAD ซึ่งสามารถถูกใช้ในกระบวนการออกแบบผลิตภัณฑ์และกระบวนการผลิตได้ในเวลาเดียวกัน งานวิจัยเสนอวิธีการออกแบบที่มีพื้นฐานอยู่บนกระบวนการผลิตโดยใช้ข้อมูลด้านรูปร่าง ชนิดของวัสดุ ข้อกำหนดด้านเครื่องมือและเครื่องจักร และอาศัยข้อมูลที่คำนวณได้รวมถึงความรู้ที่เป็นแบบ Heuristic วิธีการนี้เหมาะสมกับงานทางด้านวางแผนกระบวนการผลิตในส่วนล่าง ๆ เช่น การสร้างรหัสโปรแกรมของระบบ NC เป็นต้น

- **สาโรช พรวิจิตรจินดา (1988)** สร้างการประยุกต์ใช้ไมโครคอมพิวเตอร์กราฟฟิกส์กับการควบคุมการทำงานในงานตัดแบบระนาบสองมิติของเครื่อง CNC โดยช่วยการวางแผนกระบวนการผลิตในส่วนการสร้างรหัสสั่งงานเครื่อง CNC ซึ่งจะถูกสั่งงานโดยใช้ภาษา G Code ซึ่งจัดเป็นภาษาระดับต่ำ โปรแกรมที่พัฒนาขึ้นจะทำการกำหนดสภาพการทำงานของเครื่องจากผู้ใช้โดยตรงส่วนหนึ่งและข้อมูลที่ใช้ในการกำหนดรูปร่างของชิ้นงานจากโปรแกรม AutoCAD อีกส่วนหนึ่ง โปรแกรมดังกล่าว จะทำการวิเคราะห์หาตำแหน่งของคัทเตอร์ที่เหมาะสมสำหรับขนาดของคัทเตอร์ที่ผู้ใช้กำหนด และทำการสร้างรหัสสั่งงานให้กับเครื่อง CNC

2.2. สรุปงานวิจัยและผลงานที่เกี่ยวข้อง

งานวิจัยและผลงานที่เกี่ยวข้องมีเนื้อหาเกี่ยวกับการนำเอาคอมพิวเตอร์ช่วยการออกแบบเข้ามาประยุกต์ใช้ในกิจกรรมการวางแผนกระบวนการผลิตต่าง ๆ เพื่อที่จะเชื่อมต่อกับระบบคอมพิวเตอร์ช่วยการผลิตเพื่อให้การผลิตมีประสิทธิภาพมากขึ้น ผลงานวิจัยส่วนใหญ่จะนำข้อมูลที่อยู่ในรูปแบบของ AutoCAD ที่เป็นฐานข้อมูลมาเป็นพื้นฐานเพื่อประยุกต์กับเทคนิคต่าง ๆ ของระบบคอมพิวเตอร์ อาทิ ระบบผู้เชี่ยวชาญและการโปรแกรมมิ่งแบบออปเจกต์ เป็นต้น เพื่อช่วยในส่วนการคำนวณและการตัดสินใจในการกำหนดวิธีการและสภาพแวดล้อมในกระบวนการผลิตทั้งแบบการผลิตธรรมดาและการผลิตที่นำคอมพิวเตอร์เข้ามาช่วย เช่น การผลิตโดยใช้เครื่องจักร NC และ CNC เป็นต้น

บทที่ 3

คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิต

ในบทนี้จะกล่าวถึงลักษณะของการวางแผนกระบวนการผลิตแบบเดิมและการนำคอมพิวเตอร์เข้ามาช่วยวางแผนการผลิต ซึ่งมีความเกี่ยวข้องกับระบบคอมพิวเตอร์ช่วยในการออกแบบและระบบคอมพิวเตอร์ช่วยในการผลิต รวมถึงเนื้อหาที่เกี่ยวข้องกับ AutoCAD และการโปรแกรมมิ่งใน AutoCAD ซึ่งจะเป็นเครื่องมือที่สำคัญในการสร้างซอฟต์แวร์ช่วยในการวางแผนกระบวนการผลิตสำหรับงานวิจัยนี้

3.1. การวางแผนกระบวนการผลิต

การวางแผนกระบวนการผลิตเป็นกระบวนการที่จะทำให้รู้ได้ว่าผลิตภัณฑ์จะถูกผลิตขึ้นมาได้อย่างไรและมันเป็นส่วนประกอบที่สำคัญอย่างยิ่งในกระบวนการผลิต การวางแผนการผลิตนั้นมีส่วนที่จะเป็นตัวกำหนดสิ่งต่างๆ ในการผลิตอาทิ ต้นทุนของวัตถุดิบที่ใช้ผลิต กิจกรรมในการผลิต ความสามารถในการแข่งขันของบริษัท การวางแผนการผลิต ประสิทธิภาพการผลิต รวมทั้งคุณภาพของผลิตภัณฑ์ การวางแผนการผลิตจึงเป็นส่วนที่สำคัญของการเชื่อมต่อกันระหว่างกระบวนการออกแบบและกระบวนการผลิต

การวางแผนกระบวนการผลิตถูกให้คำจำกัดความไปในหลาย ๆ ทิศทางเนื่องจากในแต่ละงานมีจุดประสงค์ที่แตกต่างกันออกไป ยิ่งไปกว่านั้นคำจำกัดความยังถูกเปลี่ยนแปลงไปเนื่องจากมีการคิดค้นเทคโนโลยีใหม่ขึ้นมาเรื่อย ๆ อย่างไรก็ตามคำจัดความของการวางแผนกระบวนการผลิตที่เป็นที่ยอมรับกันทั่วไป ถูกให้ไว้โดย Chang และ Wysk ได้แก่ “การเตรียมรายละเอียดของการทำงานในการเปลี่ยนงานออกบบทางวิศวกรรมให้เป็นชิ้นงานสำเร็จ”

กิจกรรมของการวางแผนการผลิตมีอยู่หลายระดับด้วยกัน ในระดับที่เกี่ยวข้องกับการออกแบบ การวางแผนกระบวนการผลิตรับผิดชอบในส่วนการกำหนดวิธีการทั่วไปในการผลิตซึ่งจะมีผลต่อข้อจำกัดในการออกแบบผลิตภัณฑ์ ซึ่งผู้ออกแบบจะต้องวางแผนเพื่อพิจารณาให้ง่ายต่อการผลิตเพื่อลดต้นทุนที่ไม่จำเป็น ข้อมูลในการออกแบบจะถูกส่งไปยังส่วนการผลิตและผู้วางแผนกระบวนการผลิตจะกำหนดรายละเอียดงานในการผลิตแต่ละส่วนประกอบของผลิตภัณฑ์ ในบางกรณีกิจกรรมเหล่านี้จะเกี่ยวข้องกับ

- การพิจารณาข้อกำหนดต่าง ๆ ของแบบชิ้นงาน

- การเลือกชนิดกระบวนการผลิต
- การกำหนดลำดับการผลิต
- การเลือกเครื่องมือ เครื่องมือ อุปกรณ์จับยึดและอุปกรณ์วัด
- การออกแบบอุปกรณ์จับยึด
- การกำหนดค่าพารามิเตอร์ที่เกี่ยวข้อง
- การเขียนโปรแกรมควบคุมเครื่อง CNC
- การตรวจสอบโปรแกรมควบคุมเครื่อง CNC

จะเห็นได้ว่างานในส่วนการวางแผนกระบวนการผลิตมีส่วนประกอบต่าง ๆ มากมาย และมีความสำคัญอย่างยิ่งในกระบวนการผลิต ถึงแม้ว่าการวางแผนกระบวนการผลิตจะมีความสำคัญมากก็ตาม แต่ก็ยังไม่มีวิธีการที่ถูกกำหนดแน่นอนในการวางแผนที่จะสามารถนำไปใช้และนำไปฝึกให้กับบุคคลที่ต้องทำงานนี้ กิจกรรมต่างๆ ที่เกี่ยวข้องจะค่อนข้างขึ้นอยู่กับประสบการณ์และความชำนาญของแรงงานที่ทำงานเป็นส่วนใหญ่ ด้วยเหตุนี้ทำให้มันเป็นข้อจำกัดในการวิเคราะห์และการหาวิธีการที่ดีที่สุดในการวางแผนกระบวนการผลิตซึ่งส่งผลถึงความเสียหายในด้านต้นทุน ความล่าช้า ความผิดพลาดและความไม่มีมาตรฐานในการผลิต

การตัดสินใจในการวางแผนกระบวนการผลิตแบบเดิม ๆ ผู้วางแผนต้องอาศัยข้อมูลในการผลิตในหลายด้าน เช่น ใช้ข้อมูลจากหนังสือกำหนดมาตรฐานด้านพิกัดเผื่อ ด้านวัสดุ ด้านความเรียบผิว ด้านเครื่องมือ ฯลฯ ซึ่งแหล่งข้อมูลมีลักษณะกว้างขวางและกระจัดกระจาย ซึ่งเป็นงานที่ใช้เวลามากในการเตรียมพร้อมก่อนที่จะส่งการตัดสินใจขั้นสุดท้ายให้ฝ่ายผลิต ข้อเสียของการวางแผนกระบวนการผลิตแบบเดิมสรุปได้ดังนี้

- การตัดสินใจขึ้นกับความชำนาญและจิตใจของผู้วางแผนแต่ละคน ทำให้ไม่มีมาตรฐาน
- เป็นงานที่ซับซ้อนและยุ่งยาก อาจทำให้เกิดการตัดสินใจที่เกิดจากการเดาของผู้วางแผน
- ผลที่ได้มักจะไม่สมบูรณ์หรือไม่มีความคงที่
- ข้อมูลในการผลิตมักไม่มีความทันสมัย

3.2. คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิต

ตามที่ได้กล่าวมาแล้ว ในปัจจุบันการผลิตจะเป็นในลักษณะที่ต้องการตอบสนองความต้องการของตลาดที่มีการเปลี่ยนแปลงอย่างรวดเร็ว สินค้าที่ผลิตจำเป็นต้องเปลี่ยนรูปแบบให้หลากหลายและจำนวนผลิตภัณฑ์ที่ผลิตต่อชุดน้อยลง กิจกรรมในการผลิตจึงมีรายละเอียดที่มากขึ้น ด้วยเทคโนโลยีต่าง ๆ ที่มนุษย์ได้คิดค้นขึ้น การนำคอมพิวเตอร์เข้ามาช่วยในกิจกรรมการ

ผลิต (Computer Integrated Manufacturing - CIM) จึงเป็นส่วนสำคัญอย่างมากในการผลิตในปัจจุบัน CIM จะเป็นระบบที่รวมกิจกรรมทุกชนิดในการผลิตไว้ด้วยกันซึ่งประกอบไปด้วยระบบต่าง ๆ เช่น Computer-Aided Design (CAD) Computer-Aided Manufacturing (CAM) Flexible Manufacturing Systems (FMS) Computer-Aided Process Planning (CAPP) Manufacturing Resources Planning (MRP) และอื่น ๆ อีกมากมาย

เนื่องจากการผลิตมีการเปลี่ยนแปลงรูปแบบไป การนำคอมพิวเตอร์เข้ามาช่วยในการวางแผนกระบวนการผลิต (CAPP) จึงเกิดขึ้นเพื่อทำให้การวางแผนกระบวนการผลิตมีความซับซ้อนน้อยลงและลดผลเสียต่าง ๆ ที่เกิดขึ้นจากการวางแผนกระบวนการผลิตแบบเดิม

สำหรับ CAPP แล้วจะมีหน้าที่หลักสำหรับระบบ CIM ก็คือเป็นตัวเชื่อมที่สำคัญในการรวมกันของระบบ CAD และ CAM เข้าด้วยกัน CAPP จะเป็นเหมือนสะพานเชื่อมระหว่างการออกแบบและการผลิต มันเป็นสิ่งที่จำเป็นสำหรับการทำให้ระบบการผลิตเป็นไปอย่างมีประสิทธิภาพมากที่สุดด้วยการใช้ข้อมูลที่มีอยู่ในส่วนการออกแบบได้อย่างเต็มที่ ทำให้ผลิตภัณฑ์ที่ได้มีคุณภาพตรงกับการออกแบบภายในระยะเวลาที่สั้นลง

ในส่วนของคอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตที่จะกล่าวถึงต่อไปจะเกี่ยวข้องกับ CAD CAM และ CAPP รวมไปถึงส่วนที่มีความเกี่ยวข้องกับงานวิจัยนี้ อันได้แก่ AutoCAD และการโปรแกรมมิ่งใน AutoCAD

3.2.1. CAD CAM และ CAPP

3.2.1.1. Computer-Aided Design (CAD)

คอมพิวเตอร์ช่วยในการออกแบบหรือ CAD เป็นการสร้างและการจัดการรูปภาพในการออกแบบโดยใช้คอมพิวเตอร์เพื่อช่วยในการออกแบบของผู้ใช้ CAD ถูกพัฒนามาตั้งแต่ช่วงกลางของศตวรรษที่ 20 เพื่อที่จะตอบสนองและช่วยเหลืองานออกแบบต่าง ๆ ที่เกี่ยวข้องกับเทคโนโลยี งานออกแบบ และอุตสาหกรรมต่าง ๆ CAD เป็นระบบที่เกิดจากการผสมผสานกันระหว่างความสามารถของมนุษย์และเครื่องจักรในการทำให้เกิดการออกแบบที่มีประสิทธิภาพมากที่สุด

ซอฟต์แวร์ CAD ที่ถูกพัฒนาขึ้นมีอยู่มากมาย ทั้งหมดมีลักษณะการทำงานโดยใช้ระบบพิกัดแบบคาร์ทีเซียนในการจัดการข้อมูล พิกัดด้าน x y และ z จะเป็นตัวอธิบายตำแหน่งของส่วนต่าง ๆ ในการออกแบบ ฟังก์ชันและความสามารถทั่วไปที่ซอฟต์แวร์ CAD จำเป็นต้องมี

เช่น การสร้างและแก้ไขภาพในแบบ 2 มิติและ 3 มิติ การย่อและลดขนาด การหมุนมุมมองของภาพ การเปลี่ยนแปลงตำแหน่งของภาพ การระบุขนาด การคำนวณค่าต่าง ๆ (อาทิ พื้นที่ ปริมาตร ฯลฯ) และอื่น ๆ อีกมากมาย และที่สำคัญสำหรับการใช้งานของผู้ใช้ที่ง่ายที่สุด ซอฟต์แวร์จะต้องมีความง่ายในการติดต่อสื่อสารกับผู้ใช้ ซึ่งเกณฑ์สำหรับระบบลักษณะนี้ที่จะต้องมีได้แก่

- ซอฟต์แวร์จะต้องถูกออกแบบให้ผู้ใช้ทำงานเกี่ยวกับด้านรูปภาพมากกว่าด้านคำสั่งที่ต้องป้อนให้ซอฟต์แวร์
- จำนวนขั้นตอนในการสั่งคำสั่งแต่ละคำสั่งควรจะมีให้น้อยที่สุด
- คำสั่งที่มีผลต่อฐานข้อมูลภายในจะต้องมีการป้องกันต่ออุบัติเหตุที่อาจเกิดขึ้นจากการลบหรือแก้ไขรูปภาพ
- เมนูสำหรับผู้ใช้ควรจัดเรียงเพื่อให้ง่ายในการสื่อสารกับผู้ใช้
- ลักษณะรูปภาพที่ปรากฏในซอฟต์แวร์ควรมีความง่ายในการอ่านมากที่สุด

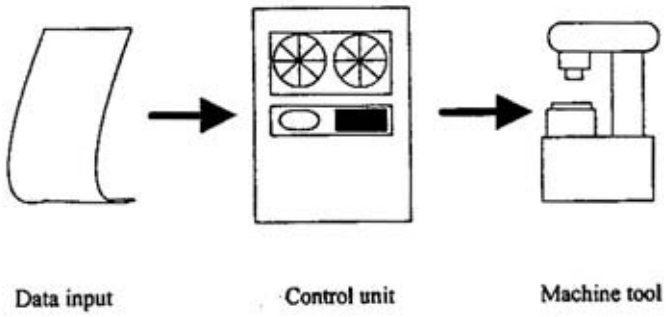
สำหรับงานวิจัยนี้ได้ใช้ CAD มาช่วยในการออกแบบโดยใช้ซอฟต์แวร์ AutoCAD (จะได้อธิบายต่อไปในหัวข้อ 3.2.2.) ซึ่งมีลักษณะงานในแบบ 2 มิติเท่านั้น

3.2.1.2. Computer-Aided Manufacturing (CAM)

ในอดีตอุตสาหกรรมการผลิตได้พัฒนาและปรับปรุงกระบวนการผลิตขึ้นมาอย่างต่อเนื่อง แต่สำหรับการเปลี่ยนแปลงครั้งสำคัญซึ่งถือได้ว่าเป็นการปฏิวัติซึ่งเริ่มเมื่อหลังสงครามโลกครั้งที่สอง ในขณะนั้น Massachusetts Institute of Technology (MIT) ได้คิดค้นเครื่องจักรที่สามารถใช้การสั่งการโดยวิธีการทางตัวเลข (Numerical Control) ได้ หลักการทำงานพื้นฐานของมันอยู่ที่การใช้รหัสตัวเลขที่เครื่องจักรสามารถตีความได้เพื่อสั่งการทำงานด้วยความได้เปรียบนี้ทำให้มันสามารถที่จะสร้างการทำงานที่มีลำดับงานที่ซับซ้อนและมีความแม่นยำสูง

เครื่องจักรที่มีระบบควบคุมแบบ NC ประกอบไปด้วย 3 ส่วนหลัก ๆ (รูปที่ 3.1) ได้แก่

1. **ข้อมูลเข้า (Data Input)** เป็นส่วนข้อมูลรหัสสั่งเครื่องจักรที่ถูกเก็บและบันทึกไว้ในสื่อต่าง ๆ เช่น เทปแม่เหล็ก และ Floppy Disk เป็นต้น
2. **หน่วยควบคุม (Control Unit)** เป็นส่วนที่มีหน้าที่ในการแปลงรหัสที่อยู่ในสื่อไปสู่รูปแบบคำสั่งที่เครื่องจักรสามารถเข้าใจและสั่งการได้
3. **เครื่องมือกล (Machine Tool)** เป็นส่วนประกอบที่รับการสั่งการจากหน่วยควบคุมเพื่อทำการผลิต เช่น เครื่องเจาะและเครื่องกัด เป็นต้น



รูปที่ 3.1 *ส่วนประกอบของระบบควบคุมแบบ NC*

อย่างไรก็ตามการเปลี่ยนแปลงการทำงานของเครื่องจักร NC เพื่อให้เหมาะกับการทำงานที่มีการเปลี่ยนแปลงมากนั้น จะทำได้ยาก เนื่องจากผู้ควบคุมเครื่องอาจจะต้องทำการเปลี่ยนแปลงอุปกรณ์หรือวงจรไฟฟ้าตามไป ภายหลังที่คอมพิวเตอร์ได้เป็นที่รู้จัก จึงได้มีการนำคอมพิวเตอร์เข้ามาใช้ร่วมกับระบบ NC เครื่องจักร CNC (Computer Numerical Control) จึงได้เกิดขึ้น

ระบบ CNC จะนำคอมพิวเตอร์มาช่วยในส่วนของความสามารถในการโปรแกรมมิ่งและการมีหน่วยความจำ คอมพิวเตอร์ขนาดเล็กก็สามารถนำมาประยุกต์ใช้กับระบบนี้ได้เช่นกัน ลักษณะและข้อได้เปรียบของ CNC นั้นมีดังนี้

- การที่มีการแสดงผลผ่านทางจอภาพ รวมถึงการจำลองกระบวนการผลิตที่เกิดจากรหัสคำสั่ง เช่น แสดงเส้นทางการเจาะ ทำให้สามารถแก้ไขรหัสคำสั่งและตรวจสอบกระบวนการผลิตได้เมื่อต้องการ
- รหัสที่ใช้กับระบบ NC จะถูกอ่านเพียงครั้งเดียวโดยไม่มีการเก็บบันทึกไว้ แต่ระบบ CNC สามารถเรียกกลับมาใช้สำหรับการผลิตครั้งต่อ ๆ ไปได้โดยไม่ต้องมีการอ่านผ่านสื่อทุกครั้งที่ต้องการใช้
- ด้วยการทำงานของคอมพิวเตอร์ที่มีอยู่ในระบบ CNC ทำให้สามารถติดต่อกันได้ระหว่างหน่วยการผลิตอื่น ๆ ได้เช่น ระบบหุ่นยนต์ รวมไปถึงติดต่อกับซอฟต์แวร์อื่น ๆ ทำให้มีหน้าที่การทำงานได้ยืดหยุ่นขึ้น
- มีความละเอียดแม่นยำสูงมากกว่าระบบ NC

โดยทั่วไป ในการผลิตชิ้นงานด้วยเครื่อง CNC จะต้องมีการเตรียมข้อมูลและจัดข้อมูลดังกล่าวให้อยู่ในรูปแบบที่คอมพิวเตอร์หรือหน่วยประมวลผลจะเข้าใจได้ ข้อมูลที่เป็นปัจจัยสำคัญสำหรับการผลิตชิ้นงานแบ่งออกได้เป็น 4 ส่วนได้แก่

1. **ข้อมูลที่ได้โดยตรงจากแบบ** ได้แก่ ขนาดของชิ้นงาน เช่น ความสูง และความกว้าง เป็นต้น รูปร่างของชิ้นงาน เช่น เส้นตรง ส่วนโค้ง เป็นต้น

2. **ข้อมูลที่ใช้กำหนดพารามิเตอร์ของเครื่องมือกล** การกำหนดพารามิเตอร์ของเครื่องจะขึ้นอยู่กับชนิดของผิววัสดุ ค่าความคลาดเคลื่อน ชนิดของชิ้นงาน อัตราการป้อนชิ้นงาน ความเร็วของเครื่องมือกลและอุปกรณ์เสริมต่าง ๆ เช่น การเปิดและปิดของสารระบายความร้อน
3. **ข้อมูลที่ได้จากผู้ป้อน** ผู้ป้อนจะต้องคุ้นเคยกับกระบวนการผลิตและมีความรู้เกี่ยวกับคุณสมบัติเฉพาะของเครื่องจักรนั้น ๆ เช่น ทิศทางการเคลื่อนที่ของเครื่องมือกล และการเปลี่ยนหัวเครื่องมือกล เป็นต้น
4. **ข้อมูลที่เกี่ยวข้องกับระบบควบคุม** ข้อมูลนี้จะใช้ในการสั่งงานกับระบบควบคุมโดยตรง เช่น ความเร่งและความหน่วง เป็นต้น

วิธีการเตรียมข้อมูลให้อยู่ในรูปแบบมาตรฐานที่เครื่อง CNC จะสามารถอ่านได้ สามารถแบ่งออกเป็น 2 วิธีได้แก่

1. **การเตรียมข้อมูลด้วยมือ (Manual Preparation)** ผู้ทำการป้อนข้อมูลจะเป็นผู้จัดเตรียมข้อมูลทั้งหมด โดยเริ่มจากการกำหนดค่าพารามิเตอร์ การหาขั้นตอนการดำเนินงานของเครื่องมือกลที่เหมาะสม และการกรอกข้อมูลลงในใบกรอกรายการ ในแต่ละบรรทัดของใบกรอกรายการจะหมายถึง 1 ชุดคำสั่ง ซึ่งประกอบด้วยการเคลื่อนที่ของเครื่องมือกลจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่ง โดยรวมถึงคำสั่งที่ใช้สั่งงานโดยตรงกับระบบควบคุม รหัสมาตรฐานที่นิยมใช้สั่งการกับเครื่อง CNC ได้แก่ จีโคด (Given Control Function: G code) รูปแบบทั่วไปการจัดเรียงข้อมูลทั่วไปสำหรับเครื่อง CNC ในแต่ละบรรทัดมีดังนี้

N G XYZAB F S T M EOB

ส่วนประกอบต่าง ๆ อธิบายได้ดังนี้

1. **รหัสสำหรับตัวเลขแสดงลำดับการทำงาน (Sequence Number)** ใช้รหัส N ในการแสดงลำดับการทำงานก่อนหลังสำหรับเครื่อง CNC ตัวเลขที่ตามหลังรหัสมีได้ไม่เกิน 4 หลัก (N0 – N9999) ในแต่ละบล็อกจะแสดงรหัสลำดับการทำงานได้เพียงตัวเดียว
2. **รหัสสำหรับกำหนดวิธีการเคลื่อนที่ (Preparatory Function)** ใช้รหัส G ตัวเลขที่ตามหลังรหัสจะบอกถึงวิธีการเคลื่อนที่ของเครื่องมือกล เช่น G01 เป็นรหัสกำหนดการเคลื่อนที่ในแนวเส้นตรง เป็นต้น ตัวเลขที่ตามหลังมีไม่เกิน 2 หลัก
3. **รหัสสำหรับกำหนดอุปกรณ์เสริมอื่น ๆ (Miscellaneous Function)** ใช้รหัส M ซึ่งนำมาใช้เพื่อร่วมในการผลิตให้มีประสิทธิภาพมากขึ้น

4. **รหัสสำหรับกำหนดเครื่องมือ (Tool Function)** รหัส T ใช้ในการกำหนดค่าพารามิเตอร์ของเครื่องมือที่ต้องใช้ในการผลิตชิ้นงาน
5. **รหัสสำหรับอัตราการป้อนชิ้นงาน (Feed rate)** รหัส F ใช้ในการกำหนดอัตราความเร็วในการกัดหรือเจาะชิ้นงาน
6. **รหัสสำหรับกำหนดความเร็วของเพลา (Speed)** รหัส S ในการกำหนดความเร็วของเพลา ตัวเลขที่ตามหลังไม่เกิน 4 หลัก
7. **รหัสสำหรับกำหนดตำแหน่งการเคลื่อนที่** ผู้ใช้สามารถกำหนดตำแหน่งการเคลื่อนที่ได้ 2 แบบ คือ กำหนดตำแหน่งทุกตำแหน่งเทียบกับจุดอ้างอิงจุดหนึ่ง และ การกำหนดตำแหน่งเทียบกับตำแหน่งที่ผ่านมา

2. **การเตรียมข้อมูลด้วยคอมพิวเตอร์ (Computer-Assisted Preparation)** ในการผลิตชิ้นงานที่มีความซับซ้อน มันเป็นการยากที่จะสามารถคำนวณค่าต่าง ๆ ที่เกี่ยวข้องในการผลิตได้ ดังนั้นจึงมีการนำคอมพิวเตอร์มาช่วยในการเตรียมข้อมูลที่ส่งเข้าไปยังเครื่อง CNC เพื่อให้เกิดความถูกต้องและลดเวลาในการเตรียมข้อมูล ภาษาที่นิยมใช้ในการเตรียมข้อมูลลักษณะนี้ที่เป็นที่นิยมมากคือ ภาษา APT (Automatically Programmed Tool) คิดค้นโดยสถาบัน MIT ภาษานี้เป็นภาษาระดับสูง ซึ่งมันจะทำหน้าที่คำนวณค่าต่าง ๆ ที่เหมาะสมกับชิ้นงานที่ต้องการผลิตโดยอัตโนมัติ นอกจากภาษา APT แล้วยังมีภาษาอื่น ๆ ที่ใช้กับเครื่อง CNC ได้ เช่น COMPACT ADAPT EXAPT และ SPLIT เป็นต้น

3.2.1.3. Computer-Aided Process Planning (CAPP)

ตามที่ได้กล่าวไปแล้วว่า คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิต (Computer-Aided Process Planning - CAPP) เป็นตัวเชื่อมที่สำคัญระหว่าง CAD และ CAM เพื่อให้เกิดเป็นระบบการผลิตที่มีประสิทธิภาพ

หน้าที่หลักของระบบคอมพิวเตอร์ช่วยวางแผนกระบวนการผลิต คือ การนำกระบวนการอื่นมาใช้แทนกิจกรรมที่วางแผนโดยมนุษย์ ระดับของการนำคอมพิวเตอร์เข้ามาช่วยในการวางแผนกระบวนการผลิตอาจจะแตกต่างกันไป ตัวอย่างเช่น บริษัทที่มีความมั่นคงและมีตลาดที่แน่นอนอาจมีความจำเป็นในการเปลี่ยนแปลงรูปแบบสินค้าเพียงแค่เล็กน้อย (Minor Modification) ในลักษณะเช่นนี้ระบบคอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตจะมุ่งเน้นที่การจัดเก็บและปรับปรุงแผนเดิมที่มีอยู่ ตัวอย่างอีกกรณีหนึ่ง คือ ในกรณีโรงงานอุตสาหกรรมขนาดกลาง อาจต้องพบกับผลิตภัณฑ์หรือชิ้นส่วนใหม่ทุกสัปดาห์หรือแม้กระทั่งทุก

วัน ทำให้มีความต้องการระบบคอมพิวเตอร์ช่วยวางแผนกระบวนการผลิตที่สามารถวางแผนกระบวนการผลิตได้อย่างอัตโนมัติ

จากที่กล่าวมาและโดยทั่วไปแล้ว การนำระบบคอมพิวเตอร์เข้ามาช่วยวางแผนกระบวนการผลิตถูกจำแนกออกเป็น 2 วิธี คือ ช่วยในการค้นหาและแก้ไขแผนเดิม (Variant Approach) และช่วยสร้างแผนกระบวนการผลิตใหม่แบบอัตโนมัติ (Generative Approach)

การนำระบบคอมพิวเตอร์เข้ามาช่วยในการค้นหาและแก้ไขแผนเดิม เป็นการนำเอาระบบคอมพิวเตอร์เข้ามาช่วยวิธีการทำงานด้วยมือ โดยอาศัยข้อได้เปรียบของระบบคอมพิวเตอร์ในแง่ของประสิทธิภาพในการจัดการข้อมูล การค้นหาข้อมูล การแก้ไขข้อมูล และการพิมพ์รายงานแผนกระบวนการผลิต การพัฒนาระบบคอมพิวเตอร์ช่วยวางแผนกระบวนการผลิตด้วยวิธีการปรับปรุงแผนเดิมนี้ เริ่มจากการจัดกลุ่มชิ้นงานที่มีลักษณะคล้ายๆกัน กำหนดแผนกระบวนการผลิตมาตรฐานของแต่ละกลุ่ม และจัดเก็บไว้ในระบบคอมพิวเตอร์ แผนกระบวนการผลิตมาตรฐานจะประกอบไปด้วยขั้นตอนของการผลิต และรายละเอียดการทำงานในแต่ละขั้นตอน โดยพิจารณาจากความสามารถของกระบวนการผลิตชนิดต่างๆและเครื่องมือที่มีอยู่ แผนกระบวนการผลิตมาตรฐานจะถูกจัดหมวดหมู่ และให้รหัสตามกลุ่มของชิ้นงาน วิธีการใช้งาน ระบบจะทำการวิเคราะห์งานออกแบบให้อยู่ในรูปของรหัสกลุ่มชิ้นงาน จากนั้นทำการเรียกแผนกระบวนการผลิตมาตรฐานของกลุ่มชิ้นงานนั้นๆ ผู้วางแผนกระบวนการผลิตจะทำการปรับปรุงแผนกระบวนการผลิตให้สอดคล้องกับชิ้นงานปัจจุบัน ข้อได้เปรียบของวิธีการนี้ต่อวิธี Manual คือ ช่วยลดเวลา และลดความน่าเบื่อหน่ายในการจัดการกับเอกสารจำนวนมาก จากการแบ่งชิ้นงานออกเป็นกลุ่มต่างๆ ทำให้เชื่อได้ว่าแผนกระบวนการผลิตที่ได้น่าจะมีมาตรฐานและมีความสม่ำเสมอมากขึ้น ข้อเสียเปรียบที่สำคัญของวิธีการนี้ คือ ความรู้และประสบการณ์ของผู้วางแผนยังคงเป็นปัจจัยสำคัญในการกำหนดคุณภาพของแผนกระบวนการผลิตที่ได้รับ

การนำระบบคอมพิวเตอร์เข้ามาช่วยสร้างแผนกระบวนการผลิตแบบอัตโนมัติ โครงสร้างของระบบคอมพิวเตอร์ช่วยวางแผนกระบวนการผลิตด้วยวิธีนี้ประกอบไปด้วย ส่วนการวิเคราะห์รูปร่างของชิ้นงาน ส่วนกลไกในการตัดสินใจ และสูตรคำนวนต่างๆ วิธีการนี้ไม่เหมือนกับวิธีการปรับปรุงแผนเดิม เนื่องจากไม่มีการกำหนดแผนกระบวนการผลิตมาตรฐานไว้ล่วงหน้า หรือกล่าวอีกอย่างหนึ่งก็คือไม่มีการเก็บแผนกระบวนการผลิตเดิมไว้ โดยคอมพิวเตอร์จะสร้างแผนการปฏิบัติงานสำหรับชิ้นส่วนใหม่ๆขึ้นเองอย่างอัตโนมัติทุกครั้งที่มีการสั่งชิ้นส่วนและปล่อยเข้าสู่ระบบการผลิต

3.2.2. AutoCAD และการโปรแกรมมิ่งใน AutoCAD

3.2.2.1. AutoCAD

AutoCAD เป็นซอฟต์แวร์ช่วยในการออกแบบ ซึ่งเป็นผลิตภัณฑ์จากบริษัท Autodesk จำกัด AutoCAD ถือได้ว่าเป็นซอฟต์แวร์ช่วยในการออกแบบแรก ๆ ที่ถูกพัฒนาขึ้นเพื่อให้เหมาะสมกับการใช้งานกับคอมพิวเตอร์ส่วนบุคคล ซึ่งในสมัยก่อนซอฟต์แวร์ช่วยในการออกแบบถูกพัฒนาให้ทำงานบนคอมพิวเตอร์เมนเฟรมหรือมินิคอมพิวเตอร์ จึงทำให้ AutoCAD ได้ถูกพัฒนาขึ้นมานาน จึงทำให้การออกแบบโดยใช้คอมพิวเตอร์ส่วนบุคคลมีความสามารถสูงและเป็นที่นิยมใช้กันในปัจจุบัน

AutoCAD มีความสามารถในด้านการช่วยออกแบบอยู่อย่างครบถ้วน ไม่ว่าจะเป็นการออกแบบสำหรับงานออกแบบ 2 มิติหรือ 3 มิติ และหน้าที่การใช้งานที่มากมายเพื่อช่วยในการออกแบบ ทั้งยังมีความคล่องตัวในการใช้งานสูง เช่น มีความสามารถในการที่จะให้ผู้ใช้สามารถปรับแต่งสภาพแวดล้อมการทำงานตามต้องการ มีฐานข้อมูลเพื่อติดต่อกับผู้ใช้ที่ต้องการเข้าไปใช้ และที่สำคัญคือผู้ใช้สามารถที่จะเพิ่มความสามารถในการทำงานด้วยคุณสมบัติการโปรแกรมมิ่งผ่าน AutoCAD ได้ จึงทำให้มันมีความยืดหยุ่นในการช่วยการออกแบบหรือแม้แต่การประยุกต์เข้ากับงานที่เกี่ยวกับการผลิตก็ได้เช่นเดียวกัน

ในประเทศไทย ปัจจุบัน AutoCAD ถือได้ว่าเป็นซอฟต์แวร์ช่วยในการออกแบบที่เป็นที่นิยมมากที่สุด ทำให้มีคนที่ใช้งานเป็นมืออยู่มาก เนื่องด้วยเหตุผลที่ได้กล่าวมาและทางโรงงานที่เข้าไปศึกษาก็เลือกที่จะใช้ซอฟต์แวร์นี้ในส่วนการช่วยออกแบบเช่นเดียวกัน ดังนั้นในงานวิจัยนี้จึงเลือก AutoCAD มาเป็นเครื่องมือในการช่วยการวางแผนกระบวนการผลิตด้วยเพื่อเพิ่มความสามารถของกระบวนการผลิตผ่านการโปรแกรมมิ่งของ AutoCAD

3.2.2.2. การโปรแกรมมิ่งใน AutoCAD

สำหรับ AutoCAD แล้ว ในปัจจุบันการโปรแกรมมิ่งของ AutoCAD ทำได้โดยใช้ภาษาคอมพิวเตอร์ 3 ภาษาด้วยกัน ได้แก่ AutoLISP ภาษา C และภาษา C++ ทั้งสามภาษามีลักษณะต่าง ๆ ที่แตกต่างกันดังอธิบายต่อไปนี้

AutoLISP

AutoLISP เป็นภาษาชั้นสูงเพื่อสร้างแมคโครของ AutoCAD ซึ่งถูกพัฒนาขึ้นมาจากพื้นฐานของภาษา LISP ในช่วงกลางศตวรรษที่ 20 ในสมัยนั้น LISP ถูกออกแบบมาให้ใช้ในการสร้างระบบคอมพิวเตอร์ปัญญาประดิษฐ์ (Artificial Intelligent) และมันยังคงเป็นพื้นฐานของระบบปัญญาประดิษฐ์บางระบบอยู่ในปัจจุบัน คำว่า LISP นั้นย่อมาจากคำว่า **LIS**t **P**rocessing ซึ่งแสดงถึงว่าเป็นภาษาที่เกี่ยวข้องและเหมาะสมกับข้อมูลประเภท List ซึ่งเป็นลักษณะที่เป็นตัวแทนและจัดการกับข้อมูลที่เป็นแบบกระบวนการและข้อมูลที่เป็นแบบโครงสร้าง ทางด้าน AutoCAD (จากบริษัท Autodesk จำกัด) ได้ดัดแปลงภาษา LISP เพื่อนำมาใช้เป็นภาษาในการโปรแกรมมิ่ง (Application Programming Interface – API) ของ AutoCAD ในกลางทศวรรษที่ 80 เนื่องจากธรรมชาติของภาษา LISP มีความเหมาะสมที่จะนำมาจัดการกับกระบวนการทาง CAD ที่เป็นกระบวนการออกแบบที่มีลักษณะข้อมูลที่ไม่เป็นแบบโครงสร้าง และเพื่อเพิ่มความสามารถในการโปรแกรมมิ่งให้กับ AutoCAD ด้วย

AutoLISP ทำให้ผู้ใช้ AutoCAD เพิ่มความสามารถที่จะปรับแต่งสภาพการใช้งานของ AutoCAD ให้เป็นไปตามความต้องการได้อย่างสะดวก ในการใช้งาน AutoLISP จะมีหลายลักษณะขึ้นอยู่กับความต้องการของผู้ใช้ ตั้งแต่งานง่าย ๆ เช่น การเป็นผู้ช่วยในการพิมพ์ตัวอักษร หรือการเป็นคำนวณค่าต่าง ๆ จนกระทั่งงานที่ซับซ้อนเช่น การสร้าง Drawing ทั้งหมดโดยการใช้ข้อมูลที่ถูกป้อนจากผู้ใช้หรือจากฐานข้อมูลก็ตาม

AutoLISP เป็นภาษาที่ถูกรวมไว้ในการทำงานของ AutoCAD อยู่แล้ว ดังนั้นมันจะทำงานอยู่ข้างหลัง (Transparent) ส่วน editor ของ AutoCAD ร่วมไปด้วยกัน AutoCAD ใช้ภาษา AutoLISP ในการปฏิบัติการต่าง ๆ ที่เกี่ยวข้องกับผู้ใช้ทั้งหมด อาทิเช่น ส่วนการสร้างเมนู ส่วนการสร้าง Toolbar และส่วนของคำสั่งที่ป้อนเข้าที่ AutoCAD Command Prompt ที่ใช้สั่งการ เป็นต้น AutoLISP สามารถที่จะสร้างคำสั่งต่าง ๆ เพื่อเพิ่มความสามารถตามต้องการได้เหมือนกับคำสั่งที่อยู่บนเมนูหรือ Toolbar AutoLISP ยังมีความสามารถอื่น ๆ มากมายที่จะช่วยทำให้การออกแบบง่ายขึ้นสำหรับผู้ใช้ที่ต้องการจะปรับแต่ง AutoCAD เช่นการสร้าง Dialog Box เพื่อรับข้อมูลและแสดงข้อมูลให้กับผู้ใช้ และการเพิ่มคำสั่งที่ผู้ใช้เป็นผู้กำหนดเองเข้าไปในหน่วยความจำเพื่อเป็นคำสั่งใหม่ที่ใช้ใน AutoCAD เป็นต้น

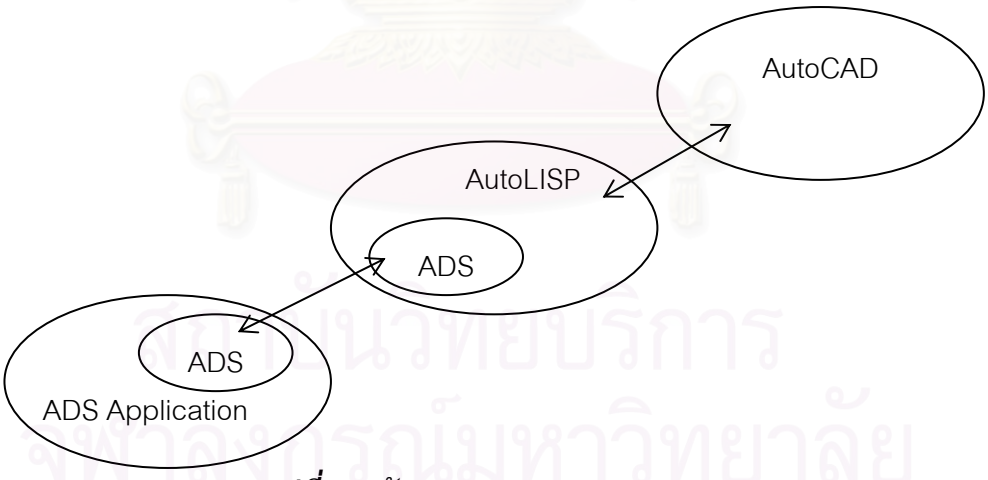
ภาษา C

ตามที่รู้กันมาว่าภาษา C เป็นภาษาที่ถูกพัฒนาขึ้นมาด้วยพื้นฐานของการโปรแกรมมิ่งโดยแท้จริง และมีความยืดหยุ่นในการใช้งานในด้านต่าง ๆ โดยเฉพาะด้านวิศวกรรมมากที่สุด

ภาษาหนึ่ง ภาษา C จึงถูกใช้ในการประยุกต์เข้ากับการโปรแกรมมิ่งสำหรับโปรแกรมด้านวิศวกรรมมากมาย ซึ่ง AutoCAD ก็เป็นหนึ่งในนั้นด้วย

ภาษา C เป็นภาษาที่ใช้โปรแกรมมิ่งสำหรับ AutoCAD โดยเริ่มนำมาใช้กับ AutoCAD Release 11 เป็นครั้งแรก การนำภาษา C มาใช้ในการโปรแกรมมิ่งสำหรับ AutoCAD นี้จะถูกเรียกว่า AutoCAD Development System (ADS) สำหรับการโปรแกรมมิ่งของ ADS จะเป็นการนำกลุ่มของ Included files (.H) และ Libraries ที่ทางบริษัท Autodesk จำกัด สร้างขึ้น ซึ่งประกอบไปด้วยฟังก์ชั่นที่จำเป็นทั้งหมดในการเข้าถึงและการจัดการกับฐานข้อมูลของ Drawing เพื่อให้นักโปรแกรมมิ่งนำไปใช้งานโดยผ่านกระบวนการโปรแกรมมิ่งแบบภาษา C

การทำงานของ ADS นั้น โปรแกรมที่สร้างขึ้นโดย ADS จะรันอยู่ภายนอกของ AutoCAD โปรแกรมจะไม่ได้เป็นส่วนหนึ่งของหน่วยความจำของ AutoCAD เลย นี่หมายความว่า การติดต่อสื่อสารระหว่างโปรแกรม ADS และ AutoCAD จะมีความเร็วที่ช้าลงแต่โปรแกรมก็จะทำให้ AutoCAD หยุดการทำงานโดยไม่ตอบสนองต่อข้อมูลป้อนเข้าได้ยากขึ้นซึ่งเป็นผลดีของมันการติดต่อสื่อสารระหว่างโปรแกรม ADS และ AutoCAD จะผ่านภาษา AutoLISP ซึ่งเป็นภาษาที่ AutoCAD ใช้อยู่ในการทำงานของมันดังรูปที่ 3.2 ซึ่งเป็นลักษณะการทำงานคล้ายการโปรแกรมมิ่งด้วย AutoLISP แต่จะสามารถโปรแกรมมิ่งได้กว้างขวางและทำงานได้ซับซ้อนกว่าโดยเฉพาะการจัดการภายในฐานข้อมูลของ Drawing



***รูปที่ 3.2** ลักษณะการทำงานของระบบ ADS*

ภาษา C++

ในหลายปีที่ผ่านมา จุดประสงค์ใหญ่ ๆ สำหรับการโปรแกรมมิ่งนั้นก็คือการที่จะเขียนรหัสคำสั่งให้น้อยที่สุดและสามารถทำงานได้เร็วที่สุด โปรแกรมที่เขียนขึ้นจำเป็นจะต้องมีขนาดเล็กและทำงานได้เร็วเนื่องจากหน่วยความจำและหน่วยประมวลผลของคอมพิวเตอร์มีราคาแพง

ซึ่งทำให้ต้นทุนสูงขึ้น แต่ในปัจจุบันความสำคัญในเรื่องนี้ได้ถูกเปลี่ยนแปลงเนื่องมาจากคอมพิวเตอร์สามารถหาได้ง่ายขึ้น ราคาส่วนประกอบต่าง ๆ ก็ลดลง ความเปลี่ยนแปลงที่เกิดขึ้นรวมถึงการแข่งขันทางด้านธุรกิจที่มากขึ้นทำให้ไม่จำเป็นที่จะต้องสร้างโปรแกรมที่มีขนาดเล็กมาก แต่ในทางตรงข้าม โปรแกรมควรมีลักษณะการเขียนที่มีระเบียบ มีแบบแผน มีความง่ายในการดูแลรักษาที่ดีที่สุด ซึ่งสุดท้ายจะส่งผลให้เกิดการพัฒนาด้านโปรแกรมได้อย่างกว้างขวางและพัฒนาได้ง่ายในขณะที่ใช้ค่าใช้จ่ายที่ไม่มากจนเกินไป

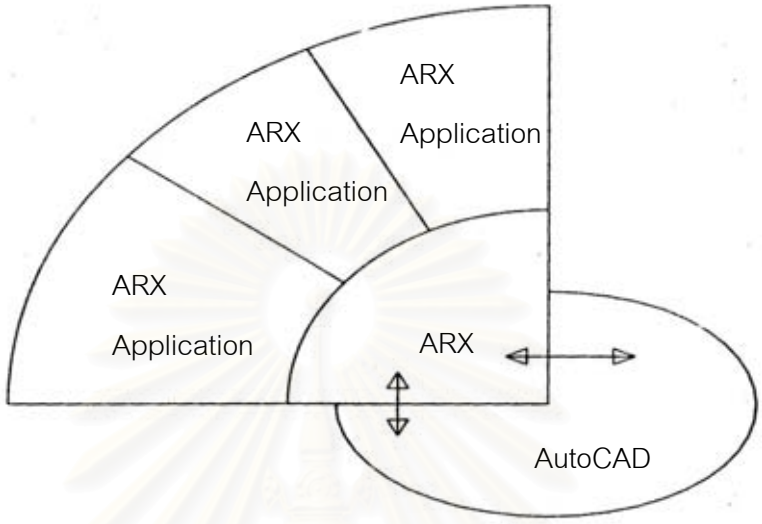
ด้วยเหตุผลต่าง ๆ ภาษา C++ ถูกพัฒนาขึ้นมาจากภาษา C อีกทีหนึ่งเพื่อที่จะทำให้มันมีความสามารถในการเพิ่มความสามารถของโปรแกรมที่เขียนขึ้นได้อย่างมีศักยภาพมากที่สุด ด้วยการใช้คุณลักษณะการโปรแกรมมิ่งที่เรียกว่า การโปรแกรมมิ่งแบบออปเจกต์ (Object-Oriented Programming – OOP) ซึ่งเป็นลักษณะของการโปรแกรมที่เป็นที่ยอมรับกันว่ามีความสามารถที่จะปรับเปลี่ยนไปตามโปรแกรมที่มีรูปแบบแตกต่างกันได้อย่างหลากหลาย และมีความเหมาะสมกับโปรแกรมที่ซับซ้อนซึ่งเป็นลักษณะของโปรแกรมในเชิงพาณิชย์ในปัจจุบัน

ทางด้าน AutoCAD ได้นำภาษา C++ มาเป็นภาษาในการโปรแกรมมิ่งครั้งแรกเมื่อนำมาใช้กับ AutoCAD Release 13 การใช้ภาษา C++ เพื่อโปรแกรมมิ่งจะถูกเรียกว่า AutoCAD Runtime eXtension (ARX) การโปรแกรมมิ่งจะมีลักษณะที่นำ Included Files และ Libraries ต่าง ๆ มาใช้เช่นเดียวกับการโปรแกรมมิ่งแบบ ADS แต่จะถูกเขียนโดยภาษา C++ ซึ่งมีข้อแตกต่างกันเรื่องการโปรแกรมมิ่งแบบออปเจกต์ ซึ่งทางด้าน ADS ไม่มีความสามารถด้านนี้ ARX จึงทำให้นักโปรแกรมมิ่งสามารถที่จะสร้างออปเจกต์เพิ่มขึ้นมาเองได้ ทำให้เกิดประโยชน์และความหลากหลายมากขึ้นในการใช้งาน

ฟังก์ชันของ ARX ที่สร้างขึ้นจะถูกสร้างให้อยู่ในรูปแบบของ Dynamic Link Libraries (DLLs) รูปแบบโปรแกรมแบบ DLLs จะเป็นเสมือนผู้ให้บริการทางด้านฟังก์ชัน ตัว DLLs เองไม่สามารถจะทำงานในตัวของมันเองแต่จะจัดหาฟังก์ชันมาให้โปรแกรม (ในที่นี้คือ AutoCAD) ใช้บริการ ดังนั้นฟังก์ชั่น DLLs จะเป็นส่วนที่ AutoCAD โหลดเข้าไปทุกครั้งเมื่อต้องการที่จะเข้าไปถึงฟังก์ชันต่าง ๆ ที่นักโปรแกรมมิ่งได้เขียนขึ้นด้วยภาษา C++ การทำงานของ ARX จึงต่างกับ ADS ตรงที่ ARX จะใช้หน่วยความจำร่วมไปด้วยกันกับหน่วยความจำของ AutoCAD ดังรูปที่ 3.3 ซึ่งส่งผลให้การทำงานของโปรแกรมที่เขียนมีความเร็วในการทำงานแต่จะไม่ค่อยปลอดภัย เพราะว่าหากโปรแกรมที่เขียนเกิดหยุดการทำงานโดยไม่ตอบสนองต่อข้อมูลป้อนเข้าจะทำให้ AutoCAD ซึ่งใช้หน่วยความจำร่วมกันเกิดการหยุดการทำงานไปด้วย

ความสามารถของ ARX จะมีมากกว่าการโปรแกรมมิ่งด้วย ADS และ AutoLISP อยู่มาก เนื่องจาก การโปรแกรมด้วย ARX จะสามารถเข้าถึงการโปรแกรมผ่าน ADS และ

AutoLISP ได้ทั้งหมดจึงทำให้ความสามารถต่าง ๆ ที่มีอยู่ใน ADS และ AutoLISP ได้รวมอยู่ใน ARX ทั้งหมดนั่นเอง ยิ่งไปกว่านั้น ARX ยังสามารถสร้างฟังก์ชันที่เป็นไปไม่ได้ใน ADS และ AutoLISP (เนื่องจากข้อจำกัดในแต่ละภาษา) ได้ด้วยลักษณะการโปรแกรมมิ่งแบบออปเจกต์นั่นเอง



รูปที่ 3.3 ลักษณะการทำงานของ ARX

ข้อดีและข้อเสียของการโปรแกรมมิ่งใน AutoCAD

ข้อดีและข้อเสียของลักษณะในการโปรแกรมมิ่งใน AutoCAD ทั้งสามแบบนั้นพอสรุปได้ดังต่อไปนี้

AutoLISP

ข้อดี

- AutoLISP ได้มาพร้อมกับ AutoCAD ดังนั้นไม่จำเป็นต้องเสียค่าใช้จ่ายในการซื้อ
- AutoLISP มีลักษณะที่โต้ตอบ (Interactive) กับผู้ใช้ ดังนั้นรอบในการผลิตโปรแกรม (Production Cycle) จึงสามารถทำให้สั้นลงได้มาก ซึ่งหมายถึงเมื่อเปลี่ยนรหัสคำสั่งโปรแกรมแล้วไม่จำเป็นต้องคอมไพล์ใหม่ให้เสียเวลา
- มีแหล่งความรู้ที่เกี่ยวข้องกับ AutoLISP ให้ศึกษามากมายเช่น หนังสือ และนิตยสารต่าง ๆ เป็นต้น
- มีความเหมาะสมที่สุดกับการใช้งานที่เล็ก ๆ ที่มีรหัสคำสั่งไม่กี่บรรทัด
- มีความปลอดภัยมากในการใช้งาน เนื่องจากไม่ได้ใช้หน่วยความจำร่วมกับ AutoCAD และเป็นการยากที่จะเกิดขึ้นด้วย

ข้อเสีย

- เนื่องจาก AutoLISP เป็นสับเซ็ตของภาษา LISP ซึ่งเป็นภาษาทางระบบปัญญาประดิษฐ์ที่เกิดขึ้นมานาน ทำให้ไม่ค่อยเป็นที่นิยมในการใช้สำหรับด้านเกี่ยวกับระบบปัญญาประดิษฐ์และด้านที่เกี่ยวกับ AutoCAD มากนัก เพราะได้มีเครื่องมือใหม่ ๆ ที่มีศักยภาพมากกว่าออกมาให้ผู้ใช้ได้เลือก
- AutoLISP มีความเร็วที่ต่ำในขณะที่รัน โดยเฉพาะโปรแกรมที่มีความซับซ้อนและเป็นงานที่ทำซ้ำ ๆ งานเหล่านี้อาจใช้เวลาเป็นชั่วโมงหรือเป็นวันในการที่จะรันให้สำเร็จ
- ทางบริษัท Autodesk จำกัด เองไม่สนับสนุนให้ใช้ AutoLISP สำหรับงานใหม่ ๆ ที่มีขนาดค่อนข้างใหญ่

ADS

ข้อดี

- ภาษา C เป็นภาษาที่เป็นที่รู้จักและยอมรับกันทั่วไปว่ามีศักยภาพที่สูงโดยเฉพาะในงานวิศกรรม
- ใช้เวลาในการรันน้อยกว่า เมื่อเทียบกับ AutoLISP
- สำหรับ AutoCAD ภาษา C เป็นภาษาโปรแกรมมิ่งที่ทางบริษัท Autodesk จำกัดให้ความสำคัญเรื่อยมา เนื่องจากมี Libraries ที่ทางบริษัทผลิตออกมาให้นักโปรแกรมมิ่งได้ใช้อยู่เรื่อย ๆ
- โปรแกรมที่เขียนด้วยภาษา C ไว้อยู่แล้วสามารถนำมาปรับใช้เข้ากับสภาพการทำงานแบบ ADS ได้
- มีความปลอดภัยสูงเมื่อเทียบกับ ARX เนื่องจากไม่ได้ใช้หน่วยความจำร่วมกับ AutoCAD

ข้อเสีย

- ภาษา C เป็นเครื่องมือในการโปรแกรมมิ่งที่ต้องซื้อ
- ภาษา C มีความซับซ้อนในการโปรแกรมมิ่ง
- ภาษา C ไม่มีลักษณะที่โต้ตอบ การจะผลิตและปรับแต่งโปรแกรมขึ้นจะต้องผ่านการคอมไพล์และการถูกโหลดจาก AutoCAD ทุกครั้ง ซึ่งเมื่อเทียบกับ AutoLISP แล้วรอบในการผลิตโปรแกรมจะใช้เวลามากกว่า
- ภาษา C ไม่สนับสนุนการโปรแกรมมิ่งแบบออปเจกต์
- การเข้าถึงฐานข้อมูลของ Drawing ของ AutoCAD โดย ADS ทำได้ช้ากว่าแบบ ARX เนื่องจากฐานข้อมูลของ AutoCAD ในปัจจุบันได้สนับสนุนการโปรแกรมมิ่ง

แบบออปเจกต์ ทำให้รูปแบบภายในฐานข้อมูลมีลักษณะสนับสนุนภาษา C++ มากกว่า

ARX

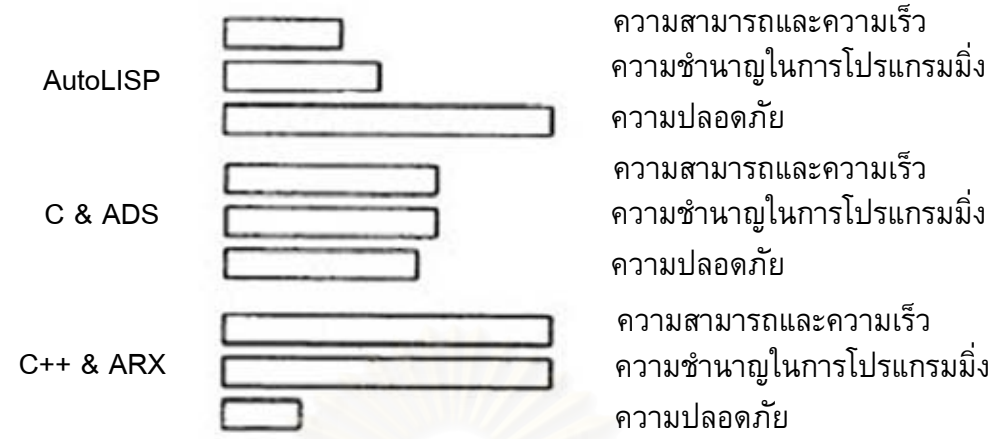
ข้อดี

- สนับสนุนการโปรแกรมมิ่งแบบออปเจกต์ ทำให้นักโปรแกรมมิ่งสามารถสร้างโปรแกรมได้กว้างขวางกว่า เช่น การสร้างออปเจกต์ขึ้นมาใช้เองภายในฐานข้อมูลของ AutoCAD
- ภาษา C++ เริ่มกลายเป็นภาษาโปรแกรมมิ่งมาตรฐานและมีความนิยมอย่างมากในการสร้างโปรแกรมและงานที่มีขนาดใหญ่
- ภาษา C++ ทำงานได้เร็วกว่าภาษา C และเร็วกว่ามากเมื่อเทียบกับ AutoLISP
- เช่นเดียวกันกับภาษา C ทางบริษัท Autodesk จำกัดให้การสนับสนุนในการใช้งาน ARX ในด้านการสร้าง Libraries ออกมาเรื่อย ๆ
- ภาษา C++ สามารถเข้าถึงฐานข้อมูลของ AutoCAD ได้ดีกว่าเนื่องจากรูปแบบภายในฐานข้อมูลของ AutoCAD มีลักษณะเป็นออปเจกต์
- เพิ่มความสามารถในการโปรแกรมมิ่งด้วย MFC (Microsoft Foundation Class) ซึ่งเป็นฐานข้อมูลของภาษา C++ ที่ทางบริษัท Microsoft จำกัด สร้างขึ้นเพื่อการสร้างโปรแกรมทางด้านภาษา C++

ข้อเสีย

- ภาษา C++ เป็นเครื่องมือในการโปรแกรมมิ่งที่ต้องซื้อ
- ภาษา C++ มีความซับซ้อนในการโปรแกรมมิ่งมากกว่าภาษา C
- ภาษา C++ ไม่มีลักษณะในการโต้ตอบ การจะผลิตและปรับแต่งโปรแกรมขึ้นมีความซับซ้อน ยุ่งยากมากกว่าเมื่อเทียบกับการใช้แบบ ADS
- ภาษา C++ ใช้หน่วยความจำมากกว่าภาษา C
- มีความปลอดภัยที่ต่ำเพราะต้องใช้หน่วยความจำร่วมกันกับ AutoCAD ถ้าหากโปรแกรมที่สร้างขึ้นโดย ARX เกิดหยุดการทำงานนั่นหมายถึง AutoCAD จะหยุดการทำงานไปด้วย ซึ่งอาจทำให้เกิดความเสียหายกับข้อมูลที่ยังไม่ได้บันทึกได้

ภาพโดยรวมในการเปรียบเทียบการโปรแกรมมิ่งใน AutoCAD ทั้งสามแบบนั้นอาจแสดงได้ในแผนภาพดังรูปที่ 3.4



รูปที่ 3.4 *แสดงการเปรียบเทียบการโปรแกรมมิ่งใน AutoCAD*

3.3. สรุป

การนำคอมพิวเตอร์เข้ามาช่วยการวางแผนกระบวนการผลิตเป็นสิ่งจำเป็นอย่างยิ่งในการที่จะนำมาทดแทนการวางแผนกระบวนการผลิตแบบเดิมสำหรับการผลิตในปัจจุบัน มันเป็นตัวเชื่อมระหว่างการออกแบบและการผลิตเพื่อช่วยให้เกิดการผลิตที่มีประสิทธิภาพ

ในงานวิจัยนี้จะเป็นการนำคอมพิวเตอร์เข้าไปช่วยในการวางแผนกระบวนการผลิตในส่วนกิจกรรมการเขียนโปรแกรมควบคุมเครื่อง CNC สำหรับกระบวนการเจาะแผ่นเหล็กของเครื่องจักร CNC Turret Punch Press ในกระบวนการผลิตจำเป็นต้องเตรียมข้อมูลเพื่อป้อนเข้าเครื่องจักรโดยมีรูปแบบในลักษณะเป็น G code โดยที่รูปแบบของมันถูกกำหนดโดยบริษัทผู้ผลิตเครื่องจักรเพื่อให้เหมาะสมกับกระบวนการเจาะโดยเฉพาะ ผลิตภัณฑ์ที่ได้จากกระบวนการนี้เป็นแผ่นเหล็กที่เป็นรูปแบบ 2 มิติซึ่งได้ถูกออกแบบโดยซอฟต์แวร์ AutoCAD โดยผู้วางแผนกระบวนการผลิต ซอฟต์แวร์ที่สร้างขึ้นเพื่อเข้าไปช่วยในกิจกรรมการเขียนโปรแกรมควบคุมเครื่อง CNC ถูกเขียนขึ้นโดยใช้ความสามารถในการโปรแกรมมิ่งของ AutoCAD โดยเลือกใช้ระบบ ARX ซึ่งใช้ภาษา C++ เนื่องจากซอฟต์แวร์ที่สร้างขึ้นจะต้องเข้าไปเกี่ยวข้องกับฐานข้อมูลของ AutoCAD เป็นส่วนใหญ่ (ภาษา AutoLISP ไม่มีความสามารถด้านนี้) และยังเป็นการโปรแกรมมิ่งแบบออปเจกต์ที่มีความสามารถและความยืดหยุ่นสูง รองรับความสามารถในการพัฒนาซอฟต์แวร์นี้เพื่อใช้งานที่ซับซ้อนขึ้นในอนาคต

บทที่ 4

การวางแผนกระบวนการผลิตสำหรับการเจาะรูสำหรับเครื่องจักร CNC Turret Punch Press

เนื้อหาในบทนี้กล่าวถึงข้อมูลเกี่ยวกับเครื่องจักรที่เข้าไปสร้างซอฟต์แวร์ช่วยในการสร้างรหัสโปรแกรม G Code ซึ่งเป็นข้อมูลที่ใช้สั่งการเครื่องจักรที่มีรูปแบบที่ทางบริษัทผู้ผลิตกำหนดขึ้น บทนี้ยังกล่าวถึงรูปแบบ G Code ถูกกำหนดไว้ให้มีหน้าที่สำหรับงานเจาะแผ่นโลหะที่มีรูเป็นลักษณะต่าง ๆ รวมถึงขั้นตอนและปัญหาที่เกิดขึ้นของการวางแผนกระบวนการผลิตแบบเดิมและจุดที่นำซอฟต์แวร์เข้าไปช่วยแก้ปัญหา

4.1. เครื่องจักร CNC Turret Punch Press MURATA C2500

เครื่องจักรที่เข้ามาทำการศึกษาได้แก่ เครื่องจักร CNC Turret Punch Press MURATA C2500 ซึ่งผลิตโดยบริษัท Murata Wiedemann จำกัด เป็นเครื่อง CNC ที่ใช้ในงานเจาะแผ่นโลหะโดยมีการสั่งการด้วยระบบ CNC ซึ่งทำการผลิตโดยการป้อนข้อมูลที่ทางผู้วางแผนกระบวนการผลิตป้อนเข้าไปในรูปแบบของ G Code ในหัวข้อนี้จะกล่าวถึงรายละเอียดเฉพาะที่เกี่ยวกับเครื่องจักรนี้ในส่วนที่สำคัญที่เกี่ยวข้องกับกิจกรรมการวางแผนกระบวนการผลิตที่เข้าไปศึกษา ซึ่งประกอบด้วยข้อมูลทั่วไปและส่วนประกอบที่สำคัญของเครื่องจักร สำหรับข้อมูลที่เกี่ยวกับข้อกำหนด (Specification) ของเครื่องจักรถูกรวบรวมอยู่ภาคผนวก ก.

4.1.1. ข้อมูลทั่วไปของเครื่องจักร CNC Turret Punch Press MURATA Centrum 2500

เครื่องจักร CNC Turret Punch Press MURATA Centrum 2500 (รูปที่ 4.1) เป็นเครื่องจักรที่มีระบบควบคุมแบบ CNC ซึ่งถูกออกแบบให้ทำงานเจาะแผ่นโลหะด้วยความเร็วสูงและมีความแม่นยำสูง การเคลื่อนที่ไปยังตำแหน่งต่าง ๆ ของโต๊ะและแท่นหมุนจะทำโดยการเคลื่อนที่ในสามแกนที่เป็นอิสระกันโดยใช้ AC Servo Motor เป็นตัวขับ การเจาะในแต่ละครั้งจะถูกสั่งโดยรหัสคำสั่ง G Code ที่ใช้รหัสควบคุมหน้าที่การทำงานที่มีอยู่หลากหลายสำหรับชิ้นงานที่ต้องการผลิต ซึ่งรหัสคำสั่งดังกล่าวจะถูกเก็บไว้ในสื่อที่เป็นเทปหรืออยู่ในหน่วยความจำของเครื่องจักร การสั่งการด้วยมือในการเจาะ การเคลื่อนที่ การเปลี่ยนหัวเจาะและการตรวจสอบรหัสคำสั่ง ก็สามารถทำได้โดยตรง

รูปที่ 4.1 *เครื่องจักร CNC Turret Punch Press MURATA Centrum 2500*

เครื่องจักรนี้ประกอบไปด้วยส่วนที่เป็นโต๊ะ ส่วนควบคุม และส่วนการเจาะ ที่มีลักษณะรวมกันเป็นหน่วยผลิตหน่วยเดียวซึ่งทำให้ติดตั้งง่าย ไม่ซับซ้อน และขนาดของเครื่องไม่เทอะทะ ทำให้ใช้พื้นที่ไม่มากในการติดตั้งและใช้งาน

ลักษณะและความสามารถทั่วไปของเครื่องจักร CNC Turret Punch Press MURATA C2500 สรุปได้ดังนี้

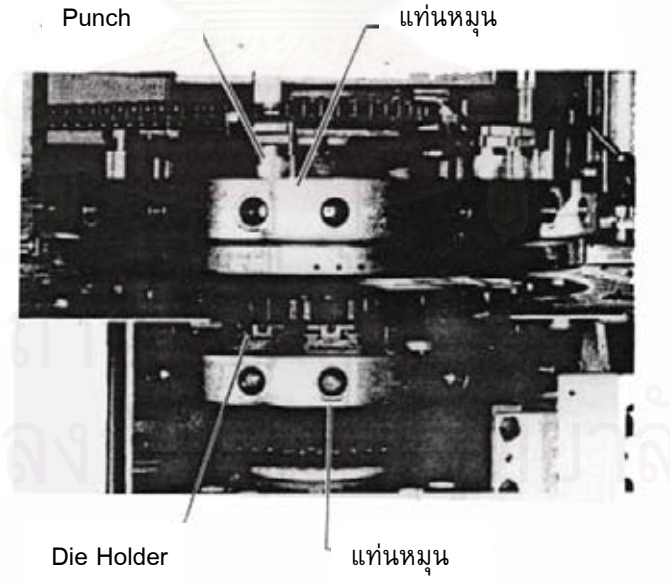
- ระบบการเคลื่อนที่แบบไฮโดรลิกที่มีเสียงที่เกิดขึ้นน้อย
- มีอัตราการผลิตสูง คือ มีการเจาะที่ความเร็วสูงสุด 200 ครั้งต่อนาที
- ความสามารถในการเจาะสูง สามารถรองรับหัวเจาะที่มีขนาดใหญ่ที่สุดเส้นผ่านศูนย์กลาง 76 มม.
- เจาะแผ่นโลหะได้หนาที่สุดที่ 6.35 มม.

- ใช้หัวเจาะที่มีต้นทุนต่ำ
- มีระบบหล่อลื่นแบบอัตโนมัติ
- มีระบบป้องกันมือจับชิ้นงานแบบอัตโนมัติ
- ระบบควบคุมแบบ CNC
- ใช้ AC Servo Motor ควบคุมการเคลื่อนที่
- มีระบบเตือนความผิดพลาดหากเกิดความผิดพลาดในการผลิต

4.1.2. ส่วนประกอบ

เครื่องจักร CNC Turret Punch Press MURATA C2500 มีส่วนประกอบสำคัญสามารถจำแนกได้ 4 ส่วนได้แก่

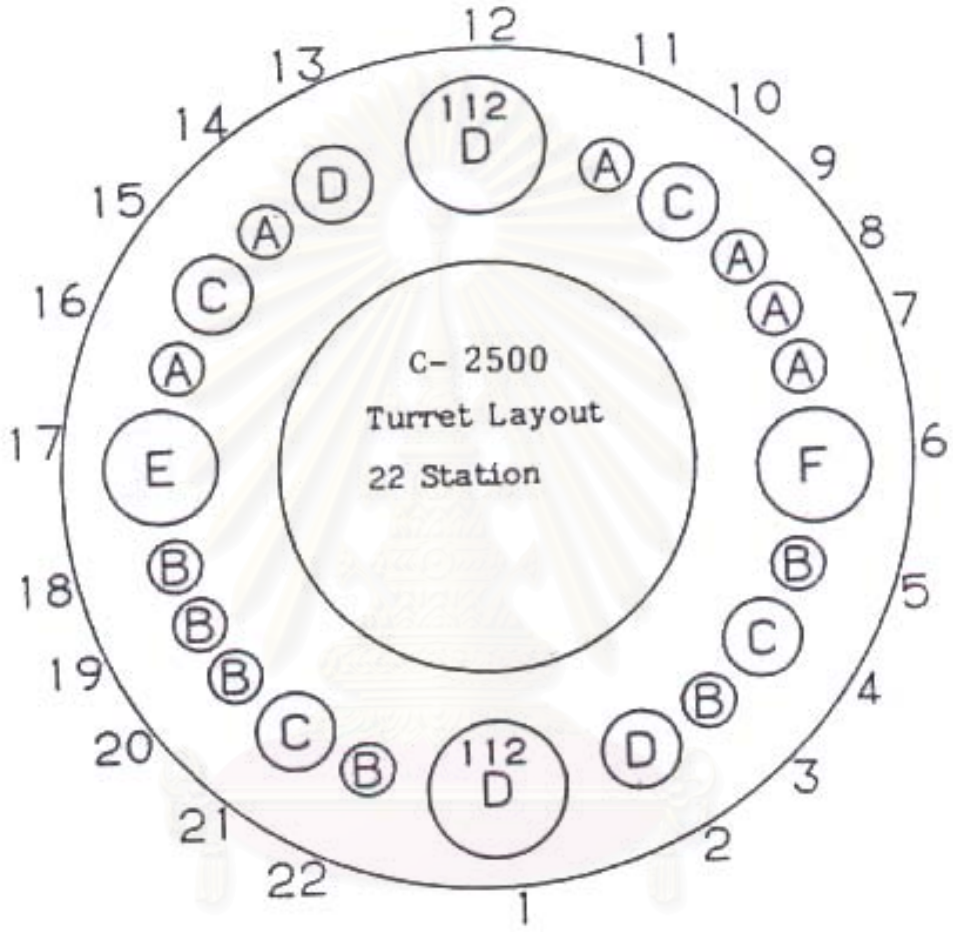
1. **Press Frame** คือส่วนที่เป็นโครงสร้างและส่วนกลไกที่ใช้ในการเคลื่อนที่ของหัวเจาะ ส่วนที่เป็นโครงสร้างทำมาจากแผ่นเหล็กกล้าแล้วนำมาเชื่อมเข้าด้วยกัน มันถูกออกแบบให้มีลักษณะเป็นรูปตัว C เพื่อกระจายการรับน้ำหนักและทำให้ลดความสั่นสะเทือนต่าง ๆ ที่เกิดขึ้นจากการเคลื่อนที่ของโต๊ะและแท่นหมุน ส่วนที่เป็นกลไก เป็นส่วนที่ขับเคลื่อนหัวเจาะซึ่งขับโดยมอเตอร์และมีอุปกรณ์ทางกลอยู่ภายใน
2. **แท่นหมุน (Turret)** ลักษณะทั่วไปแสดงดังรูปที่ 4.2 ซึ่งประกอบด้วยส่วนประกอบย่อย ๆ ได้แก่ Punch Holder แท่นหมุนบน แท่นหมุนล่าง และ Die Holder



***รูปที่ 4.2** ลักษณะของแท่นหมุน*

แท่นหมุนบนและแท่นหมุนล่างถูกติดตั้งอยู่ภายใน Press Frame และยึดไว้โดยโรลเลอร์แบริ่ง (Roller Bearing) 2 ตัว ซึ่งทั้งสองเคลื่อนที่โดยใช้โซ่ ระบบขับเคลื่อนของแท่นหมุนขับโดย AC Servo Motor และมีส่วนประกอบทางกลย่อยอยู่ภายใน

แท่นหมุนด้านบนเป็นส่วนที่ติดตั้งหัวเจาะ ซึ่งเครื่องจักรรุ่นนี้มีตำแหน่งสำหรับติดตั้งหัวเจาะอยู่ทั้งหมด 22 ตำแหน่งเรียงกันอยู่ตามเส้นรอบวงของแท่นหมุนดังรูปที่ 4.3 สำหรับเครื่องจักรรุ่นนี้มีหัวเจาะที่สามารถหมุนได้ (Index Tool) อยู่สองตำแหน่งได้แก่ ตำแหน่ง 1 และ 12 เครื่องหมายของหัวเจาะถูกจำแนกไว้เป็น 6 แบบ (A B C D E และ F) ตามขนาดของช่องใส่หัวเจาะ ซึ่งข้อกำหนดของขนาดของแต่ละช่องใส่หัวเจาะรวบรวมไว้ในตารางที่ 4.1

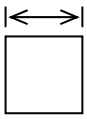
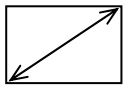
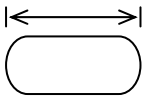


รูปที่ 4.3 *ลักษณะและตำแหน่งบนแท่นหมุน*

ตารางที่ 4.1 *ข้อกำหนดของขนาดของช่องใส่หัวเจาะบนแท่นหมุน (หน่วย: มม.)*

เครื่องหมายหัวเจาะ	วงกลม	สี่เหลี่ยมจัตุรัส	สี่เหลี่ยมผืนผ้า	Oblong
A	~ 12.7	-	-	-
B	~ 25	~ 16	~ 22	~ 25
C	~ 38	~ 22	~ 31	~ 38
D	~ 51	~ 32	~ 44	~ 51

ตารางที่ 4.1 (ต่อ) *ข้อกำหนดของขนาดของช่องใส่หัวเจาะบนแท่นหมุน (หน่วย: มม.)*

เครื่องหมายหัวเจาะ	วงกลม 	สี่เหลี่ยมจัตุรัส 	สี่เหลี่ยมผืนผ้า 	Oblong 
E	~ 64	~ 41	~ 57	~ 64
F	~ 76	~ 48	~ 67	~ 76

3. **หัวเจาะ (Tool)** หัวเจาะสำหรับเครื่องจักรมีรูปร่างต่าง ๆ กันขึ้นกับลักษณะชิ้นงานที่ต้องการเจาะ ซึ่งโดยทั่วไปมีหัวเจาะที่มีรูปแบบได้แก่ วงกลม สี่เหลี่ยมจัตุรัส สี่เหลี่ยมผืนผ้า แคปซูล (Oblong) สามเหลี่ยม และหัวเจาะวงกลมสำหรับคว้านเกลียว ซึ่งวิธีในการติดตั้งหัวเจาะเข้ากับแท่นหมุนจะไม่กล่าวถึงในที่นี้ แต่จะกล่าวถึงเฉพาะหัวเจาะทั้งหมดที่ทางโรงงานที่เข้าไปศึกษามีใช้อยู่ ซึ่งเป็นสิ่งจำเป็นที่ผู้วางแผนกระบวนการผลิตจะต้องเลือกเพื่อใช้เจาะรูที่มีลักษณะแตกต่างกัน หัวเจาะทั้งหมดแสดงในภาคผนวก ข.
4. **โต๊ะ (Table)** โต๊ะประกอบด้วยส่วนประกอบย่อย 3 ส่วนได้แก่ ฐานโต๊ะ Carriage ซึ่งเคลื่อนที่ในทิศทางซ้ายและขวา (แกน Y) และ Cross Slide ซึ่งเคลื่อนที่ในทิศทางไปข้างหน้าและหลัง (แกน X) ในส่วนของ Cross Slide จะมีชุดจับยึดแผ่นโลหะซึ่งควบคุมโดยผู้คุมเครื่องจักรและมีตัวดูดซับแรงกระแทก (Shock Absorber) ซึ่งมีหน้าที่ด้านความปลอดภัยและลดแรงกระแทกที่เกิดจากการเจาะ
5. **อุปกรณ์อัดอากาศ (Air Equipment)** มีหน้าที่โดยตรงในการขับดันแรงสำหรับการเคลื่อนที่ของหัวเจาะในแนวการเจาะในลักษณะการอัดความดันอากาศ

4.2. G Code ของเครื่องจักร CNC Turret Punch Press MURATA

4.2.1. คำสั่ง G Code ทั้งหมด

ลักษณะ G Code ของเครื่องจักรนี้ถูกกำหนดรูปแบบโดยบริษัท Murata Wiedemann จำกัด เพื่อให้เหมาะกับงานเจาะแผ่นโลหะ ซึ่งรูปแบบที่เป็นคำสั่งทั้งหมดถูกรวบรวมไว้ในตารางที่ 4.2

ตารางที่ 4.2 *รูปแบบคำสั่ง G Code ของเครื่องจักร CNC Turret Punch Press MURATA*

คำสั่ง	หน้าที่การใช้งาน
	คำสั่งหลักในการเจาะ
O	ลงทะเบียนหมายเลขโปรแกรม
N	ลงทะเบียนหมายเลขลำดับของบล็อก
X	ค่าพิกัดแนวแกน X (แบบอ้างอิงจากจุดเริ่มต้น)
Y	ค่าพิกัดแนวแกน Y (แบบอ้างอิงจากจุดที่ผ่านมา)
DX	ค่าพิกัดแนวแกน X (แบบอ้างอิงจากจุดเริ่มต้น)
DY	ค่าพิกัดแนวแกน Y (แบบอ้างอิงจากจุดที่ผ่านมา)
T	เปลี่ยนหัวเจาะ
C	หมุนหัวเจาะ
M	อื่น ๆ (ตารางที่ 4.3)
	คำสั่งเคลื่อนที่
OFS/	ออฟเซต (offset) ตำแหน่ง
MOV/	เคลื่อนที่ไปยังตำแหน่งอ้างอิงของคำสั่งที่เป็นรูปแบบใช้เจาะ
REP/	เคลื่อนที่ไปยังตำแหน่งแบบอัตโนมัติ
	คำสั่งที่เป็นรูปแบบในการเจาะรูชนิดต่าง ๆ
INC/	เจาะรูในแนวเส้นตรงแบบเพิ่มระยะจากจุดอ้างอิง
GRD/	เจาะรูแบบกริด (Grid)
LAA/	เจาะรูตามแนวมุมเอียง
ARC/	เจาะรูตามแนวส่วนโค้ง
RAD/	เจาะรูแบบเร็วและถี่ (Nibbling) ตามแนวส่วนโค้ง
HOL/	เจาะรูแบบเร็วและถี่เพื่อเปิดรูใหญ่ตามแนวเส้นรอบวง
OPN/	เจาะรูแบบเร็วและถี่เพื่อเปิดรูใหญ่โดยเริ่มจากจุดศูนย์กลาง
REC/	เจาะรูที่เป็นสี่เหลี่ยมทั้งแบบแกนเดียวและสองแกน
OBL/	เจาะรูที่เป็นสี่เหลี่ยมทั้งแบบแกนเดียวและสองแกน และมีการเจาะเศษโลหะด้านในออก
RRC/	เจาะรูสี่เหลี่ยมที่มีมุมทั้งสี่เป็นส่วนโค้งของวงกลม
CAA/	เจาะรูแบบเร็วและถี่ในแนวมุมเอียง
	คำสั่งควบคุมความเร็วของหัวเจาะ
NBL/	เจาะแบบต่อเนื่องด้วยความเร็วและถี่ (ต้องใช้คู่กับคำสั่ง RAD/, HOL/, OPN/, CAA/ เสมอ)
	คำสั่งเกี่ยวกับแมคโครฟังก์ชั่น
PAT/	ลงทะเบียนแมคโครฟังก์ชั่น (Macro Function)
END	จบแมคโครฟังก์ชั่น

ตารางที่ 4.2 (ต่อ) *รูปแบบคำสั่ง G Code ของเครื่องจักร CNC Turret Punch Press MURATA*

คำสั่ง	หน้าที่การใช้งาน
PAT	เรียกใช้แมคโครฟังก์ชั่น
MGR/	เรียกใช้แมคโครฟังก์ชั่นสำหรับการเจาะหลายชิ้นงาน
	คำสั่งพิเศษอื่น ๆ
SYM/	เจาะลักษณะรูที่สมมาตรในแบบเหมือนการส่องกระจก
SYC/	ยกเลิกการเจาะแบบสมมาตร

4.2.2. คำสั่ง G Code ที่โรงงานใช้ในการผลิต

สำหรับโรงงานที่เข้าไปศึกษา ผลิตภัณฑ์ที่ผลิตขึ้นส่วนใหญ่มีความซับซ้อนไม่มาก การใช้คำสั่ง G Code ของทางผู้วางแผนกระบวนการผลิตจึงเป็นการเลือกใช้คำสั่งที่จำเป็นในการเจาะจริง ๆ ทั้งนี้เพื่อให้ง่ายต่อความเข้าใจของผู้ที่ไม่มีประสบการณ์ในการวางแผนที่จะเข้ามารับงานส่วนนี้ ในส่วนนี้จึงอธิบายเฉพาะคำสั่งที่ใช้กันบ่อยในการวางแผนการเจาะชิ้นงาน

4.2.2.1. คำสั่งหลักในการเจาะ

(1). หมายเลขโปรแกรม (Program Number – “O”)

ใช้ตัวอักษรโอในการบอกหมายเลขโปรแกรม ตัวอักษรนี้จะถูกวางไว้ในตำแหน่งเริ่มต้นของโปรแกรม G Code รูปแบบการบอกหมายเลขโปรแกรมทำโดยวางตัวโอแล้วตามด้วยตัวเลขสี่หลักหรือน้อยกว่า ตัวเลขเหล่านี้มีไว้เพื่อแยกแยะให้โปรแกรม G Code ที่สร้างขึ้นถูกเก็บในส่วนความจำของส่วนควบคุมเครื่องจักรแยกจากโปรแกรม G Code อื่น ๆ รูปแบบคือ

O [ ] [ ] [ ] [ ]

หมายเลขที่ใช้ได้คือ 1 ถึง 9999 เมื่อเริ่มโปรแกรมด้วยรูปแบบนี้จะต้องจบโปรแกรมด้วยการใช้รหัสคำสั่ง M30 หรือ M99 (อธิบายต่อไปในส่วนของ M function)

(2). หมายเลขลำดับของบล็อก (Sequence Number - “N”)

ใช้ตัวอักษรเอ็น ในการบอกลำดับของบล็อกรหัสคำสั่ง ตัวอักษรนี้จะถูกวางไว้ในตำแหน่งเริ่มต้นของบล็อกรหัสคำสั่งที่สร้างขึ้น มีไว้เพื่อบอกว่าบล็อกรหัสคำสั่งใดจะถูกนำมาใช้ก่อน รูปแบบคือ

N [ ] [ ] [ ] [ ]

หมายเลขที่ใช้ได้คือ 1 ถึง 9999

(3). ตัวอักษรระบุค่าตัวเลข (Numerical Words)

ใช้ตัวอักษรเอ็กซ์ (“X”) และวาย (“Y”) ในการระบุพิกัดทางด้านแนวนอน (แกน X) และแนวตั้ง (แกน Y) การระบุตำแหน่งทำได้โดยการวางตัวอักษรแล้วตามด้วยตัวเลขในช่วง 0.01 ถึง 999999.99 มิลลิเมตร

ใช้ตัวอักษรซี (“C”) ในการระบุองศาที่จะให้หัวเจาะหมุนในแนวระนาบ การระบุองศาทำได้โดยการวางตัวอักษรแล้วตามด้วยตัวเลขในช่วง –359.99 ถึง 359.99 องศา

(3). ตัวอักษรในการเปลี่ยนหัวเจาะ (Tool Change Function - “T”)

ใช้ตัวอักษรที เพื่อใช้ในการเจาะชิ้นงานที่มีการเปลี่ยนหัวเจาะ การระบุหัวเจาะทำโดยวางตัวอักษรแล้วตามด้วยเลขสองหลัก เลขสองหลักนี้แสดงถึงหมายเลขของตำแหน่งในแท่นหัวเจาะ สำหรับตำแหน่งที่ของหัวเจาะที่เป็นเลขหลักเดียว ให้ใส่เลขศูนย์นำหน้าเช่น T01 เป็นต้น เครื่องจักรรุ่นนี้มีจำนวนตำแหน่งหัวเจาะอยู่ทั้งหมด 22 ตำแหน่ง

(4). ตัวอักษรแสดงคำสั่งอื่น ๆ (Miscellaneous - “M”)

ใช้ตัวอักษรเอ็ม (“M”) เพื่อใช้ในการสั่งการเครื่องจักรในรูปแบบอื่น ๆ ทำโดยการวางตัวอักษรแล้วตามด้วยเลขสองหลัก รูปแบบคำสั่ง M แสดงได้ในตารางที่ 4.3

ตารางที่ 4.3 *รูปแบบคำสั่ง M*

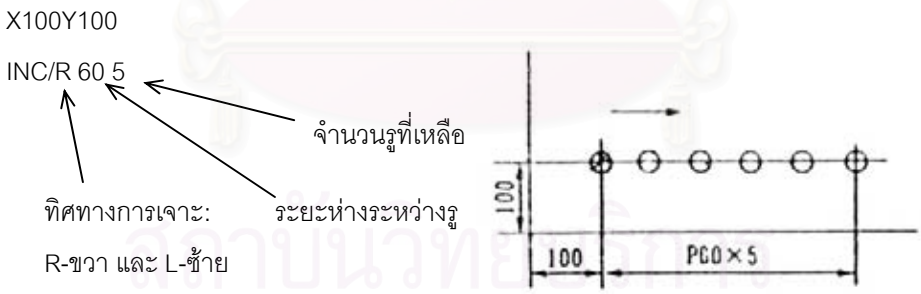
รหัส M	**หน้าที่และการใช้งาน**
M00	หยุดการเจาะของเครื่องจักรหลังจากอ่านรหัสเสร็จแล้ว
M02	หยุดการเจาะของเครื่องจักรหลังจากอ่านรหัสเสร็จแล้ว และเป็นการ reset เครื่องจักรให้อยู่ในตำแหน่งเริ่มต้น
M06	หยุดการเจาะในตำแหน่งนั้นเพื่อนำเศษโลหะออก
M12	เริ่มการเจาะแบบเร็วและถี่
M13	ยกเลิกการเจาะแบบเร็วและถี่

ตารางที่ 4.3 (ต่อ) *รูปแบบคำสั่ง M*

รหัส M	หน้าที่และการใช้งาน
M30	หยุดการเจาะของเครื่องจักรหลังจากอ่านรหัสเสร็จแล้ว และเป็นการ reset เครื่องจักรให้อยู่ในตำแหน่งเริ่มต้น สำหรับการป้อนข้อมูลแบบใช้เทป
M50	เจาะที่ความเร็วสูง
M51	เจาะที่ความเร็วต่ำ
M55	เริ่มการหน่วงจากภายนอก
M56	ยกเลิกการหน่วงจากภายนอก
M80	ให้โต๊ะเคลื่อนที่ที่ความเร็ว 100%
M81	ให้โต๊ะเคลื่อนที่ที่ความเร็ว 75%
M82	ให้โต๊ะเคลื่อนที่ที่ความเร็ว 50%
M83	ให้โต๊ะเคลื่อนที่ที่ความเร็ว 25%
M98	เรียกโปรแกรมรหัสคำสั่งย่อย
M99	ยกเลิกการเรียกโปรแกรมรหัสคำสั่งย่อย

4.2.2.2. คำสั่งที่เป็นรูปแบบในการเจาะรูชนิดต่าง ๆ

(1). "**INC/**" (Incremental) เป็นการเจาะรูในแนวเส้นตรงแบบเพิ่มระยะจากจุดอ้างอิง แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.4



รูปที่ 4.4 *รูปแบบคำสั่ง "INC/"*

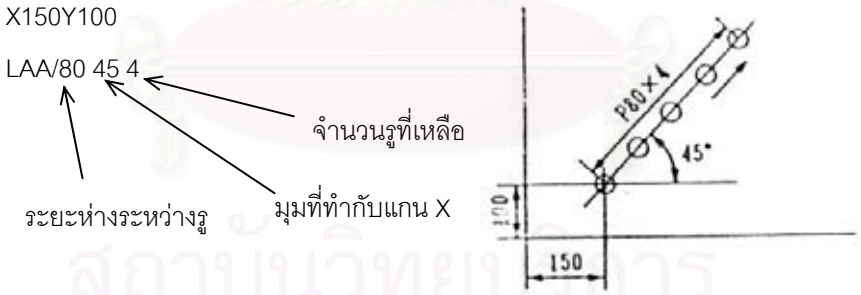
ตามตัวอย่าง เริ่มจากเจาะรูที่พิกัด (100,100) ในคำสั่งบรรทัดแรกแล้วเจาะเพิ่มในทิศทางด้านขวาในระยะห่าง 60 และเจาะอีก 5 รู

(2). “**GRD/**” (Grid) เป็นการเจาะรูแบบกริด (Grid) แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.5

รูปที่ 4.5 *รูปแบบคำสั่ง “GRD/”*

ตัวอย่างจะเจาะที่พิกัด (250,170) ก่อนแล้วเจาะตามทิศทางในรูปที่ 4.5 ซึ่งมีการกำหนดระยะห่างด้านแนวตั้งและแนวนอนและการกำหนดจำนวนรูที่เหลือในแต่ละด้าน

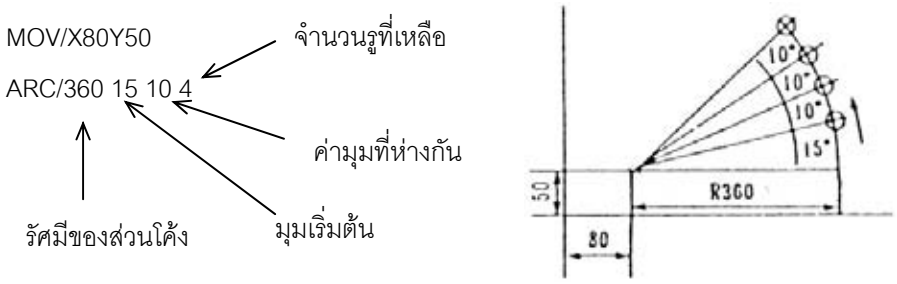
(3). “**LAA/**” (Line At Angle) เป็นการเจาะรูตามแนวมุมเอียง แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.6



รูปที่ 4.6 *รูปแบบคำสั่ง “LAA/”*

ตัวอย่างเริ่มเจาะรูที่พิกัด (150,100) แล้วเจาะรูตามแนวเอียงที่ทำกับแกน X 45 องศา และเจาะอีก 4 รู

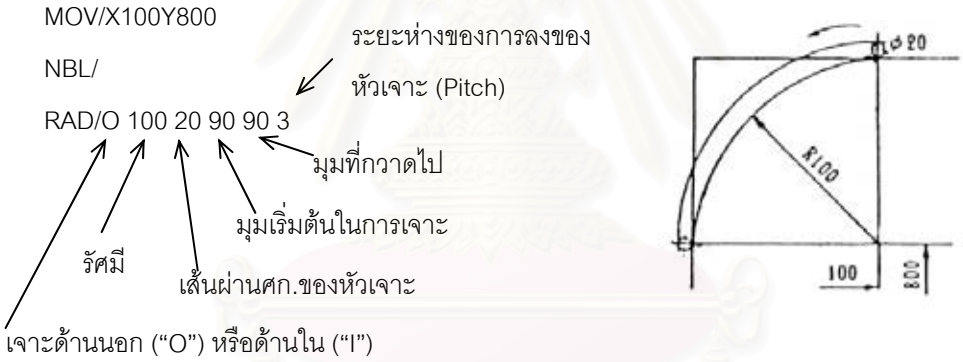
(4). “**ARC/**” (Arc) เป็นการเจาะรูตามแนวส่วนโค้ง แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.7



รูปที่ 4.7 *รูปแบบคำสั่ง "ARC/"*

ตัวอย่างเริ่มจากการเคลื่อนที่ไปยังพิกัด (80,50) ด้วยคำสั่ง MOV/ ซึ่งเป็นจุดอ้างอิงในการเจาะแต่ไม่ได้เจาะในจุดนี้ จากนั้นจะเริ่มเจาะที่รัศมียาว 360 และเริ่มด้วยมุม 15 องศา แล้วเจาะต่อไปอีก 4 รูโดยห่างกัน 10 องศา

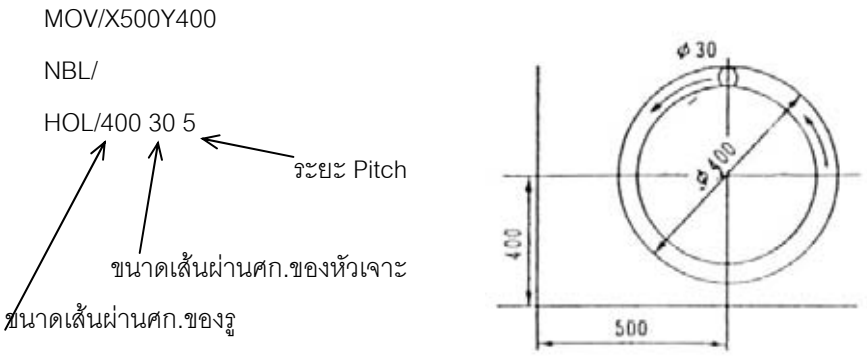
(5). "**RAD/**" (Radius) เป็นการเจาะรูแบบเร็วและถี่ (Nibbling) ตามแนวส่วนโค้ง แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.8



รูปที่ 4.8 *รูปแบบคำสั่ง "RAD/"*

ตัวอย่างเริ่มจากการเคลื่อนที่มาจุดอ้างอิงพิกัด (100,800) แล้วระบุการเจาะแบบเร็วและถี่ จากนั้นจะเจาะส่วนโค้งแบบภายนอก (ภายนอกรัศมีที่ป้อนเข้า) โดยใช้รัศมี 100 จากนั้นระบุเส้นผ่านศูนย์กลางของหัวเจาะด้วย มุมเริ่มต้นถูกกำหนดไว้ 90 และให้กวาดไป 90 องศา โดยให้ลงหัวเจาะที่ระยะห่างกัน 3

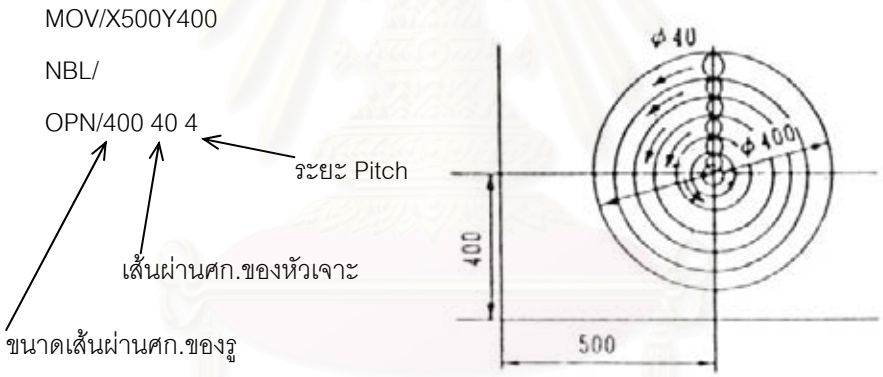
(6). "**HOL/**" (Hole) เป็นการเจาะรูแบบเร็วและถี่เพื่อเปิดรูใหญ่ตามแนวเส้นรอบวง แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.9



รูปที่ 4.9 *รูปแบบคำสั่ง "HOL/"*

ตัวอย่างเริ่มจากการเคลื่อนที่มาจุดอ้างอิงพิกัด (500,400) แล้วระบุการเจาะแบบเร็วและถี่ จากนั้นจะเจาะแบบเปิดรูโดยการเจาะรอบเส้นรอบวง โดยเจาะรูขนาดเส้นผ่านศูนย์กลาง 400 และใช้หัวเจาะขนาด 30 โดยระยะ Pitch อยู่ที่ 5

(7). "**OPN/**" (Open) เป็นการเจาะรูแบบเร็วและถี่เพื่อเปิดรูใหญ่โดยเริ่มจากจุด

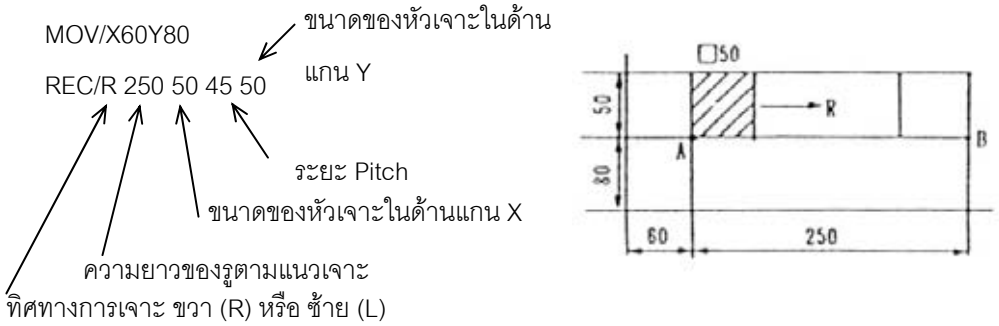


ศูนย์กลาง แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.10

รูปที่ 4.10 *รูปแบบคำสั่ง "OPN/"*

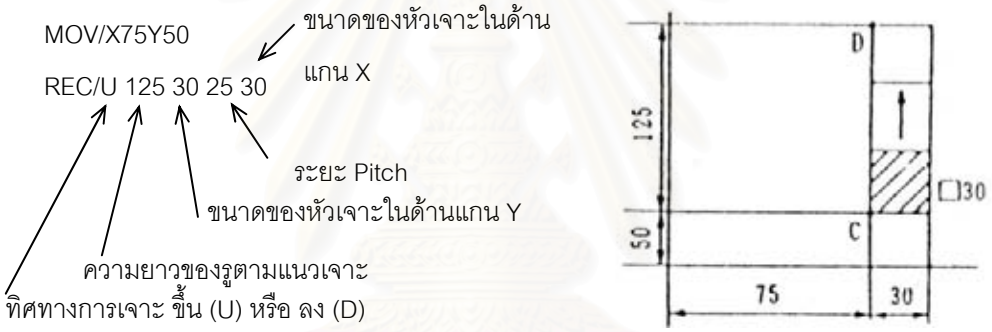
ตัวอย่างเริ่มจากการเคลื่อนที่มาจุดอ้างอิงพิกัด (500,400) แล้วระบุการเจาะแบบเร็วและถี่ จากนั้นจะเจาะแบบเปิดรูโดยการเจาะจะเริ่มจากภายในไล่ไปยังเส้นรอบวง โดยเจาะรูขนาดเส้นผ่านศูนย์กลาง 400 และใช้หัวเจาะขนาด 40 โดยระยะ Pitch อยู่ที่ 4

(8). "**REC/**" (Rectangular) เป็นการเจาะรูที่เป็นสี่เหลี่ยมทั้งแบบแกนเดียวและสองแกน แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.11 4.12 และ 4.13



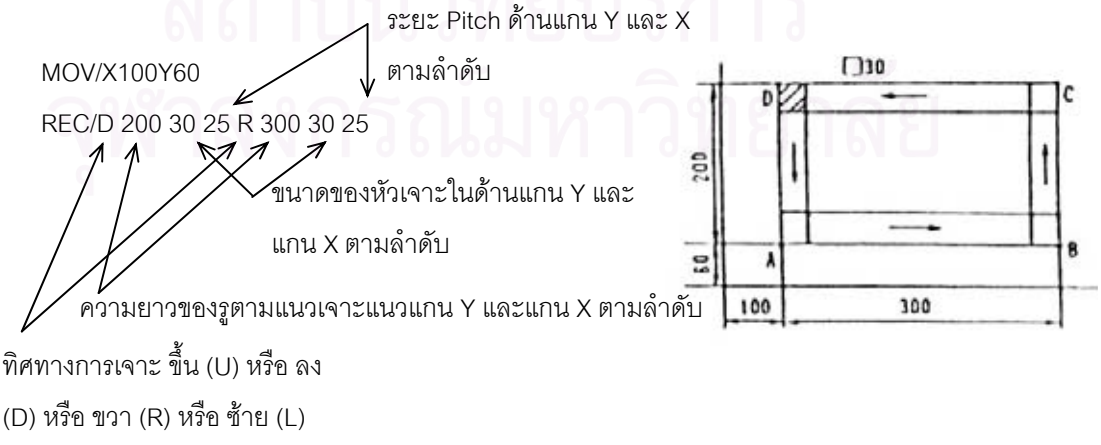
รูปที่ 4.11 *รูปแบบคำสั่ง "REC/" แบบแนวนอน*

ตัวอย่างเริ่มจากการเคลื่อนที่มาจุดอ้างอิง (60,80) (กรณีนี้สามารถใช้จุดอ้างอิงได้ทั้งจุด A หรือ B) จากนั้นเจาะไปด้านขวาด้วยระยะ 250 โดยต้องระบุขนาดหัวเจาะด้านแกน X ระยะ Pitch รวมถึงขนาดหัวเจาะด้านแกน Y ตามลำดับ



รูปที่ 4.12 *รูปแบบคำสั่ง "REC/" แบบแนวตั้ง*

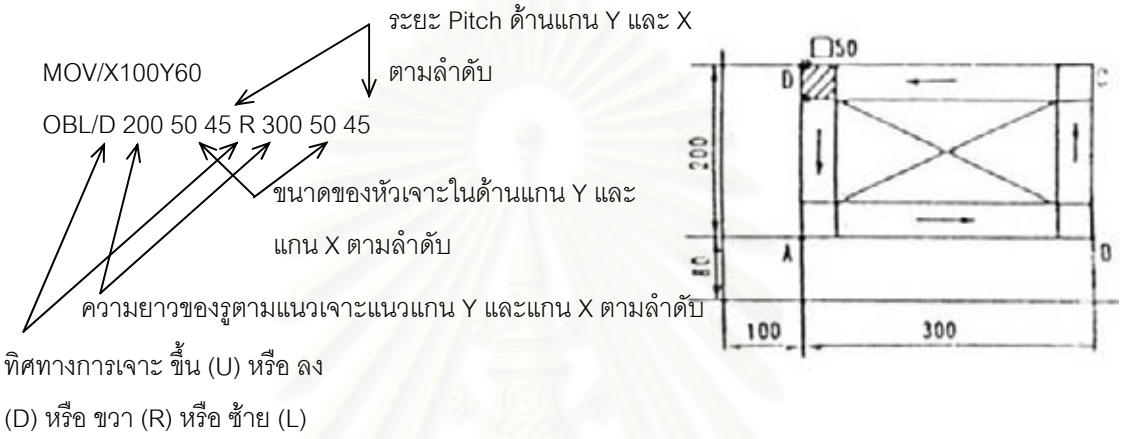
ตัวอย่างเริ่มจากการเคลื่อนที่มาจุดอ้างอิง (75,50) (กรณีนี้สามารถใช้จุดอ้างอิงได้ทั้งจุด C หรือ D) จากนั้นเจาะไปด้านบนด้วยระยะ 125 โดยต้องระบุขนาดหัวเจาะด้านแกน Y ระยะ Pitch รวมถึงขนาดหัวเจาะด้านแกน X ตามลำดับ



รูปที่ 4.13 *รูปแบบคำสั่ง "REC/" แบบแนวนอนและแนวตั้ง*

ตัวอย่างเริ่มจากการเคลื่อนที่มาจุดอ้างอิง (100,60) (กรณีนี้สามารถใช้จุดอ้างอิงได้ทั้งจุด A B C หรือ D) จากนั้นเจาะไปทิศทางลงด้วยระยะ 200 แล้วระบุขนาดหัวเจาะด้านแกน Y และระยะ Pitch ด้านแกน Y จากนั้นระบุการเจาะไปในทิศทางด้านขวา ด้วยระยะ 300 แล้วระบุขนาดหัวเจาะด้านแกน X และระยะ Pitch ด้านแกน X ตามลำดับ

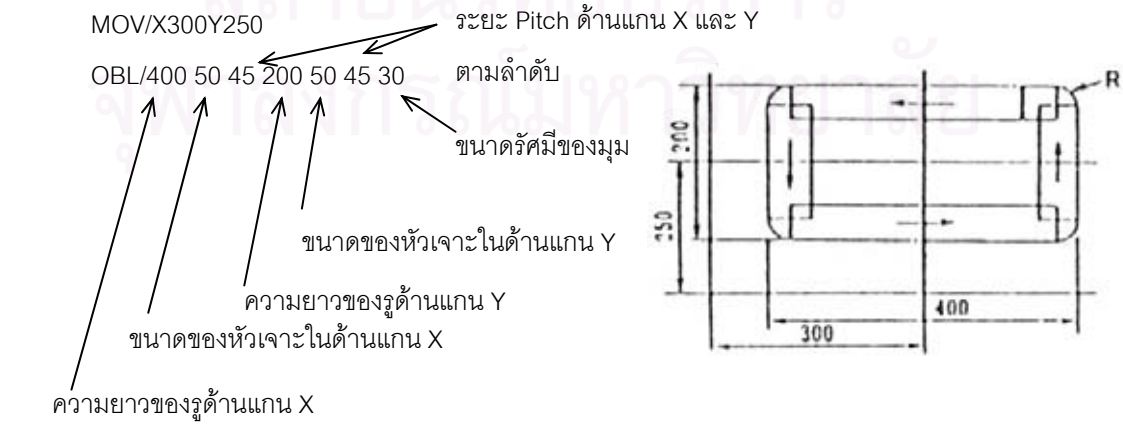
(9). "**OBL/**" (Oblong) เป็นการเจาะรูที่เป็นสี่เหลี่ยมทั้งแบบแกนเดียวและสองแกน และมีการเจาะเศษโลหะด้านในออก แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.14



รูปที่ 4.14 *รูปแบบคำสั่ง "OBL/"*

ตัวอย่างการเจาะจะมีรูปแบบเหมือนคำสั่ง "REC/" แบบแนวนอนและแนวตั้งในข้อ (8). แต่ต่างกันตรงที่มันจะเจาะภายในให้ก่อนทำให้ไม่มีเศษโลหะตกอยู่บนโต๊ะซึ่งจะเป็นอุปสรรคในการเจาะของเครื่องจักร

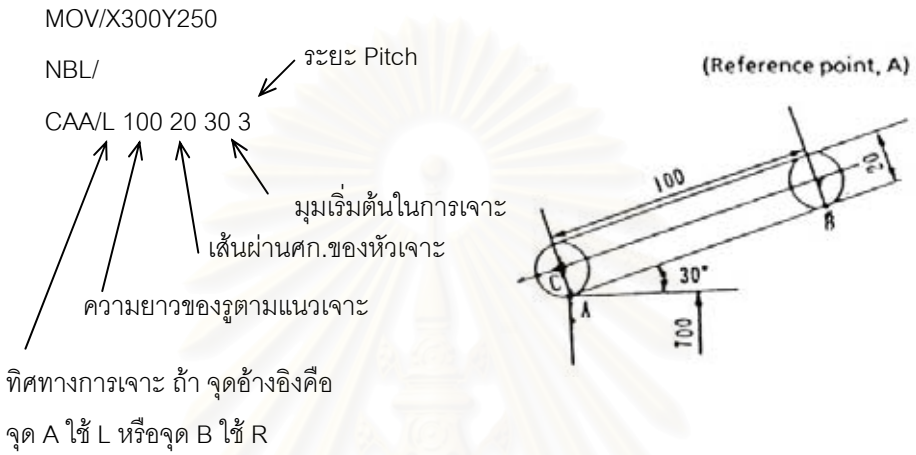
(10). "**RRC/**" (Roundable Rectangular) เป็นการเจาะรูสี่เหลี่ยมที่มีมุมทั้งสี่เป็นส่วนโค้งของวงกลม แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.15



รูปที่ 4.15 *รูปแบบคำสั่ง "RRC/"*

ตัวอย่างเริ่มจากการเคลื่อนที่มาจุดอ้างอิง (300,250) ซึ่งเป็นจุดศูนย์กลางของรู จากนั้นเจาะโดยเจาะในแนวแกน X ยาว 400 โดยขนาดหัวเจาะแกน X คือ 50 จากนั้นจะเจาะในแนวแกน Y ในระยะ 200 เช่นเดียวกัน ขนาดรัศมี 30 ต้องระบุไว้ในส่วนสุดท้ายด้วย

(11). "**CAA/**" (Cut At Angle) เป็นการเจาะรูแบบเร็วและถี่ในแนวมุมเอียง แสดงตัวอย่างการใช้และส่วนประกอบของคำสั่งในรูปที่ 4.16



รูปที่ 4.16 *รูปแบบคำสั่ง "CAA/"*

ตัวอย่างเริ่มจากการเคลื่อนที่ไปยังจุดอ้างอิงที่จุด A (อาจใช้จุด B ก็ได้) ซึ่งทิศทางสำหรับจุด A ให้ใช้ L ต่อจากนั้นใส่ระยะที่ต้องการเจาะคือ 100 ตามด้วยเส้นผ่านศูนย์กลางของหัวเจาะ มุมเริ่มต้นในการเจาะและระยะ Pitch

4.3. การวางแผนกระบวนการผลิตแบบเดิม

ผลิตภัณฑ์ที่ทางโรงงานผลิตจะเป็นการผลิตแบบตามลูกค้าสั่ง อันได้แก่ กล่องโลหะ ซึ่งใช้เป็นกล่องสำหรับครอบอุปกรณ์ต่าง ๆ เช่น แผงวงจรไฟฟ้า อุปกรณ์ดับเพลิงและอุปกรณ์โทรคมนาคม เป็นต้น ซึ่งวัตถุดิบที่ใช้คือ แผ่นโลหะชนิดต่าง ๆ เช่น อลูมิเนียมแผ่นและเหล็กแผ่น เป็นต้น

กระบวนการผลิตที่ใช้ผลิตกล่องทั่วไปเริ่มจากการนำแบบที่ได้จากลูกค้า มาสร้างเป็น Drawing ใน AutoCAD แล้วทำการวางแผนการเจาะโดยใช้เครื่องเจาะโดยการป้อนรหัสโปรแกรม G Code เมื่อได้แผ่นโลหะที่มีรูตามที่ได้วางแผนแล้ว จะนำไปเข้ากระบวนการพับและประกอบแผ่นเหล็กเข้าด้วยกันโดยการยึดด้วยวิธีการต่าง ๆ จากนั้นกล่องที่ได้จะถูกนำไปพ่นสีและตกแต่งขั้นสุดท้าย

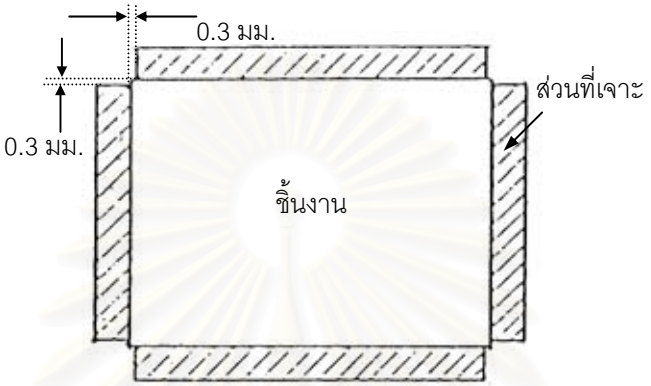
ในส่วนการผลิตที่เข้าไปศึกษา จะเข้าไปในส่วนของการวางแผนกระบวนการผลิตในกระบวนการเจาะของเครื่องจักร CNC Turret Punch Press MURATA C2500 ในกระบวนการผลิตส่วนนี้มีวัตถุดิบที่ป้อนเข้าได้แก่ แผ่นโลหะมีขนาดเหมาะสมกับเครื่องจักร และผลิตภัณฑ์ที่ได้จากส่วนนี้คือ แผ่นโลหะที่มีรูตามที่กำหนดไว้

4.3.1. ขั้นตอนในการวางแผนกระบวนการผลิตแบบเดิม

กระบวนการวางแผนการผลิตสำหรับเครื่องจักรนี้มีขั้นตอนซึ่งมีรายละเอียดดังต่อไปนี้

1. รับแบบจากลูกค้า รูปแบบของแบบที่ได้มาจากลูกค้าสามารถจำแนกได้ 3 ประเภท คือ
   - กล่องโลหะจริง ซึ่งเป็นผลิตภัณฑ์ที่ทางลูกค้าต้องการให้ผลิตจริง
   - แบบพิมพ์เขียว โดยกล่องจะมีรูปแบบอยู่ในกระดาษซึ่งมีทั้งแบบเป็นพิมพ์เขียวของกล่อง 3 มิติและแผ่นโลหะที่อยู่ในรูปแบบ 2 มิติ
   - Drawing ของ AutoCAD ซึ่งมีทั้งรูปแบบของกล่อง 3 มิติและแผ่นโลหะ 2 มิติเช่นเดียวกัน
2. พิจารณาความสามารถในการผลิต เช่น พิจารณาขนาดของกล่องและความหนาของแผ่นโลหะที่จะเจาะได้
3. คลี่แบบและสร้างแบบให้อยู่ในรูปแบบแผ่นโลหะ 2 มิติ โดยทำการสร้างไว้ใน AutoCAD เก็บไว้เพื่อเป็นฐานข้อมูลของผลิตภัณฑ์ โดยอาศัยข้อมูลทางด้านข้อกำหนดต่าง ๆ เช่น ข้อกำหนดของการพับโลหะ เป็นต้น
4. พิมพ์แบบที่สมบูรณ์ออกมาในกระดาษเพื่อนำออกมาวางแผนการเจาะ
5. พิจารณาลักษณะของรูทั้งหมดเพื่อเลือกหัวเจาะที่จะนำมาติดตั้งลงบนแท่นหมุนของเครื่องจักร
6. กำหนดตำแหน่งหัวเจาะให้กับหัวเจาะที่ได้เลือกมาใช้ เช่น การเลือกหัวเจาะวงกลมขนาดเส้นผ่านศูนย์กลาง 20 มม. ให้อยู่ในตำแหน่งที่ 2 และการเลือกหัวเจาะที่หมุนได้ (Index Tool) ให้อยู่ในตำแหน่งที่ 1 และ 12 เป็นต้น
7. สร้างรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบ เพื่อทดลองการเจาะก่อนที่จะผลิตจริง ในส่วนนี้มีสิ่งที่ต้องพิจารณาอยู่มากมายในการสร้าง อันได้แก่
   - การวัดระยะและกำหนดเป็นพิกัดด้วยเครื่องมือวัด
   - การเจาะภายนอกหรือการตัดขอบโดยใช้หัวเจาะสี่เหลี่ยมผืนผ้าที่ต้องมีการหมุนหัวเจาะ

- การเว้นระยะขอบบาง ๆ เพื่อไม่ให้ชิ้นงานที่เจาะเสร็จแล้วตกลงไปบนโต๊ะ ซึ่งจะทำให้เป็นปัญหาในการทำงานของเครื่องจักร การเว้นระยะส่วนนี้จะถูกเรียกว่า Microjoint ซึ่งมีลักษณะดังรูปที่ 4.17 โดยความกว้างของ Microjoint ในแต่ละด้านถูกกำหนดไว้ที่ 0.3 มม. โดยหลังจากที่การเจาะเสร็จแล้วนำชิ้นงานออกแล้ว ชิ้นงานจะถูกเคาะให้หลุดจากแผ่นโลหะใหญ่ในตอนหลังสุด



รูปที่ 4.17 *ลักษณะการเว้นระยะเพื่อไม่ให้ชิ้นงานตกบนโต๊ะ*

- การเจาะภายในโดยใช้คำสั่งที่เป็นรูปแบบในการเจาะรูชนิดต่าง ๆ (คำสั่งในหัวข้อ 4.2.2.2.)
- เศษโลหะที่อาจตกหล่นที่เกิดจากการเจาะภายใน ซึ่งมีวิธีการเอาออกได้ 2 วิธีได้แก่ การให้เครื่องจักรหยุดเจาะชั่วคราวโดยป้อนคำสั่ง M00 หรือ M06 เพื่อให้ผู้คุมเครื่องนำเศษโลหะออก และอีกวิธีคือการใช้คำสั่งที่เป็นรูปแบบในการเจาะที่มีความสามารถในการเจาะเศษโลหะออก
- การกำหนดเส้นทางการเคลื่อนที่ของโต๊ะเพื่อให้ใช้เวลาน้อยที่สุดในการเจาะชิ้นงาน สำหรับเกณฑ์ที่ทางโรงงานกำหนดขึ้นโดยประสบการณ์ของการใช้งานเครื่องจักรเพื่อใช้เวลาน้อยที่สุด ก็คือ จะใช้หัวเจาะตัวหนึ่งให้เจาะเสร็จสำหรับทุกรูที่กำหนดแล้วค่อยเปลี่ยนหัวเจาะ สำหรับเส้นทางในการเคลื่อนที่จะออกแบบให้โต๊ะเคลื่อนที่น้อยที่สุด
- การกำหนดความเร็วในการเจาะและความเร็วของโต๊ะ ในส่วนนี้ทางโรงงานจะไม่มีการปรับเปลี่ยน ซึ่งจะใช้ค่าเดียวกันสำหรับทุก ๆ การผลิต

8. ตรวจสอบเพื่อความแน่นอนและจัดเก็บรหัสโปรแกรม G Code เป็นโปรแกรมย่อยไว้ในแผ่น Floppy Disk เพื่อสามารถเคลื่อนย้ายข้อมูลไปยังส่วนควบคุมของเครื่องจักร CNC
9. ติดตั้งหัวเจาะทั้งหมดลงไปในแท่นเจาะของเครื่องจักร
10. ติดตั้งแผ่นโลหะและตั้งค่าเครื่องจักร

11. ทดลองเจาะโดยป้อนโปรแกรมที่จัดเก็บไว้ใน Floppy Disk ลงไปยังส่วนควบคุมของเครื่องจักร CNC หากชิ้นงานที่ได้ไม่ได้ตามที่ต้องการก็กลับไปตรวจสอบรหัสโปรแกรม G Code
12. หากการทดลองเจาะได้ชิ้นงานตามที่ต้องการ ก็เริ่มผลิตจริงตามปริมาณที่ลูกค้าสั่ง โดยนำรหัสโปรแกรมต้นแบบมาเพิ่มเติมโดยใช้การโปรแกรมแบบแมคโครง่าย ๆ เข้ามาสร้างรหัสโปรแกรมสำหรับผลิตจริง

4.3.2. ปัญหาที่พบ

ส่วนที่เข้าไปศึกษา จะทำการเขียนซอฟต์แวร์เพื่อช่วยเฉพาะในส่วนขั้นตอนการสร้างรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบ ซึ่งปัญหาที่พบในส่วนนี้ที่ทำให้การทำงานผิดพลาดและล่าช้าได้แก่ ปัญหาที่เกี่ยวกับการป้อนรหัสด้านพิกัดและระยะ เนื่องจากรหัสส่วนใหญ่จะประกอบอยู่ในรหัสโปรแกรม การสร้างรหัสโดยการใช้มือวัดและการคำนวณจะเป็นสิ่งที่เกิดความผิดพลาดได้ง่าย และสร้างความเมื่อยล้าอย่างมากในการทำงานโดยเฉพาะส่วนที่ต้องการความละเอียดสูง เช่น การเว้นระยะไว้สำหรับ Microjoint ซึ่งต้องทำงานในความละเอียดถึง 0.01 มม. เป็นต้น ความผิดพลาดที่เกิดขึ้นยังทำให้สิ้นเปลืองแผ่นโลหะที่ใช้ทำผลิตภัณฑ์ต้นแบบอีกด้วย

4.4. การวางแผนกระบวนการผลิตโดยใช้คอมพิวเตอร์ช่วย

การนำซอฟต์แวร์เข้ามาช่วยในส่วนขั้นตอนการสร้างรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบ จะทำให้ปัญหาที่เกิดขึ้นกับการวางแผนกระบวนการผลิตเดิมลดลง โดยเฉพาะในด้านความถูกต้องของการป้อนข้อมูลที่เกี่ยวกับพิกัดและระยะ เนื่องจากรูปที่สร้างขึ้นใน AutoCAD นั้นมีฐานข้อมูลที่เกี่ยวกับพิกัดและระยะอยู่แล้ว การนำข้อมูลในฐานข้อมูลมาใช้เลยจึงทำให้รหัสโปรแกรมมีความเที่ยงตรงกับรูปที่สร้างขึ้น

อย่างไรก็ตามการนำซอฟต์แวร์เข้ามาช่วยจะทำให้ขั้นตอนการวางแผนกระบวนการผลิตเดิมเปลี่ยนแปลงไปเล็กน้อย ได้แก่ ขั้นตอนการพิมพ์แบบในกระดาษเพื่อนำมาวางแผนการเจาะจะไม่จำเป็นอีกต่อไป แต่จะกลายเป็นขั้นตอนในการสร้างแบบที่มีความเหมาะสมกับการใช้งานกับซอฟต์แวร์ที่สร้างขึ้นมา (จะกล่าวถึงในบทที่ 5) ซึ่งเป็นขั้นตอนที่จำเป็นอย่างมากเพื่อให้รหัสคำสั่งที่ได้มีความถูกต้องกับชิ้นงานมากที่สุด ในขั้นตอนนี้จะไม่ซับซ้อนและใช้เวลาไม่มาก เนื่องจากสามารถนำแบบที่โรงงานสร้างไว้แล้วใน AutoCAD มาปรับเปลี่ยนเพียงเล็กน้อยเพื่อให้เหมาะสมกับซอฟต์แวร์มากที่สุด

4.5. สรุป

การผลิตชิ้นงานแผ่นโลหะของเครื่องจักร CNC Turret Punch Press นั้นจะต้องป้อนข้อมูลที่เป็นรหัสโปรแกรม G Code โดยการตัดสินใจของผู้วางแผน ซึ่งต้องใช้ความเข้าใจในรูปแบบคำสั่งต่าง ๆ เพื่อให้สร้างรหัสโปรแกรมที่ไม่ผิดพลาดและใช้เวลาน้อยในการวางแผน ซึ่งการวางแผนในลักษณะเดิมเกิดความผิดพลาดได้ง่ายเนื่องจากส่วนใหญ่ของรหัส เป็นการป้อนข้อมูลด้านพิกัดและระยะที่ได้จากการวัดโดยใช้มือ ส่งผลให้เกิดความสูญเสียด้านวัตถุดิบและความเมื่อยล้าของผู้วางแผน การนำคอมพิวเตอร์เข้ามาช่วยวางแผนกระบวนการผลิตสำหรับขั้นตอนการสร้างรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบจึงเป็นสิ่งที่จำเป็นที่จะช่วยลดปัญหาที่เกิดขึ้นได้

บทที่ 5

โปรแกรมคอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตสำหรับการเจาะรูสำหรับเครื่องจักร CNC Turret Punch Press

เนื้อหาในบทนี้กล่าวเกี่ยวกับลักษณะโดยรวมทั่ว ๆ ไปเกี่ยวกับซอฟต์แวร์ ZAmpPunch.arx ว่ามีหน้าที่ ขอบเขตและลักษณะการใช้งานรวม ๆ เป็นอย่างไร โดยการใช้ซอฟต์แวร์นี้จะต้องใช้ร่วมกับ Drawing ใน AutoCAD R14 ที่เป็นไปตามข้อกำหนดเพื่อให้เหมาะสมกับการใช้ซอฟต์แวร์เพื่อให้ได้ G Code ที่มีความถูกต้องแม่นยำมากที่สุด ในบทนี้ยังกล่าวถึงโครงสร้างการทำงานภายในซอฟต์แวร์รวมไปถึงฟังก์ชั่นภายในที่สร้างขึ้นว่ามีหน้าที่การทำงานอย่างไร

5.1. ลักษณะทั่วไปของซอฟต์แวร์

ลักษณะทั่วไปของซอฟต์แวร์ รวมถึงหน้าที่ในการสร้างรหัสโปรแกรม G Code และขอบเขตการทำงานของซอฟต์แวร์ สามารถสรุปได้ดังนี้

- ซอฟต์แวร์ ZAmpPunch.arx สร้างขึ้นโดยใช้ระบบการโปรแกรมมิ่งแบบ ARX ซึ่งใช้ภาษา C++ ในการพัฒนา ZAmpPunch.arx เป็นโปรแกรมแบบ DLLs ซึ่งเป็นเสมือนผู้ให้บริการทางฟังก์ชั่นเพื่อให้ AutoCAD R14 นำฟังก์ชั่นและคำสั่งต่าง ๆ ที่สร้างขึ้นไปใช้งานในสภาพแวดล้อมของ AutoCAD R14 เอง
- การสั่งการให้ ZAmpPunch.arx ทำงานจะสั่งผ่านโปรแกรม AutoCAD โดยทำการป้อนคำสั่งที่กำหนดไว้ลงใน AutoCAD Command Prompt โดยซอฟต์แวร์จะทำงานร่วมกับ Drawing ที่ถูกสร้างขึ้นตามข้อกำหนดที่จะนำมาใช้กับซอฟต์แวร์ ทั้งนี้เพื่อที่จะทำให้ได้รหัสคำสั่ง G Code ที่มีความถูกต้องและแม่นยำ และเพื่อไม่ให้เกิดความผิดพลาดอันเนื่องมาจากการที่ผู้ใช้สร้างรูปแบบของ Drawing ที่ไม่ได้กำหนดไว้ ข้อกำหนดต่าง ๆ ได้อธิบายในหัวข้อ 5.2.
- คำสั่งที่สร้างขึ้นใน ZAmpPunch.arx ที่เพิ่มเข้าไปประกอบด้วย 2 คำสั่งได้แก่
    1. **คำสั่งสำหรับเจาะภายใน** ผู้ใช้พิมพ์คำสั่ง “inpunch” เพื่อทำการสร้างรหัสโปรแกรม G Code สำหรับเจาะรูที่อยู่ภายในชิ้นงาน

2. **คำสั่งสำหรับเจาะภายนอก** ผู้ใช้พิมพ์คำสั่ง “outpunch” เพื่อทำการสร้างรหัสโปรแกรม G Code สำหรับเจาะรูที่อยู่ภายนอกชิ้นงาน เช่น การเจาะขอบชิ้นงานให้ได้ตามขนาด

- ซอฟต์แวร์ ZAmpPunch.arx จะสร้างรหัส G Code ที่ให้เครื่องจักร CNC สามารถอ่านและตีความได้ ซึ่งข้อมูลต่าง ๆ ภายในที่เป็นรูปแบบของ G Code ได้แสดงไว้ในบทที่ 4 โดยครอบคลุมการช่วยสร้างปัจจัยซึ่งเป็นส่วนประกอบภายในของ G Code ทั้งหมดที่จำเป็นกับการเจาะชิ้นงาน ซึ่งมีดังต่อไปนี้

1. **ตำแหน่งในรูปแบบพิกัด** ปัจจัยด้านนี้เกี่ยวข้องกับ G Code ในส่วนพิกัดซึ่งแสดงได้โดยใช้พิกัดทางด้านแกน X และแกน Y ตัวอย่างของปัจจัยนี้ได้แก่ X50Y100 หมายถึงให้โต๊ะเคลื่อนที่ไปยังตำแหน่ง (50,100) หรือ MOV/X100Y200 หมายถึงให้โต๊ะเคลื่อนที่ไปยังจุดอ้างอิงที่ (100,200) สำหรับคำสั่งที่ต้องใช้จุดอ้างอิงในการเจาะ เป็นต้น
2. **หัวเจาะที่ใช้เจาะ** ปัจจัยด้านนี้เกี่ยวข้องกับการระบุหมายเลขของตำแหน่งหัวเจาะที่ได้ติดตั้งลงไปที่ต้องการใช้งาน ตัวอย่างของปัจจัยนี้ได้แก่ X50Y100T02 หมายถึงให้เจาะที่ตำแหน่ง (50,100) โดยใช้หัวเจาะในตำแหน่งที่ 02 เป็นต้น
3. **ความเร็วในการเจาะ** ปัจจัยนี้เกี่ยวกับการสั่งให้หัวเจาะนั้นเคลื่อนที่เจาะด้วยความเร็วอย่างไร ซึ่งความเร็วที่เจาะจะแบ่งได้เป็น 2 แบบหลัก คือ เจาะด้วยความเร็วปกติ ซึ่งเป็นการเจาะโดยไม่ต้องระบุข้อมูลใด ๆ และการเจาะแบบต่อเนื่องด้วยความเร็วและถี่ ซึ่งต้องระบุคำสั่ง “NBL/” ลงไป
4. **ลักษณะและลำดับของการเคลื่อนที่ของโต๊ะ** ปัจจัยนี้ได้แก่ ลำดับของ G Code ในแต่ละบรรทัด ซึ่งการตีความของเครื่องจักรจะตีความเรียงไปตามบรรทัด บรรทัดใดถูกตีความก่อนก็จะได้รับการเจาะในส่วนนั้นก่อน
5. **ลักษณะรูปแบบของการเจาะ** ได้แก่ คำสั่งที่เป็นรูปแบบต่าง ๆ (หัวข้อ 4.2.2.2.) ที่จำเป็นในการเจาะรูปร่างของรูที่อยู่ในขอบเขตของซอฟต์แวร์ คำสั่งที่ซอฟต์แวร์ครอบคลุม ได้แก่ คำสั่งสำหรับเจาะครั้งเดียวโดยระบุพิกัดศก.ของหัวเจาะ เช่น X50Y100T01 คำสั่ง “HOL/” คำสั่ง “OPN/” คำสั่ง “RAD/’ คำสั่ง “REC/” คำสั่ง “OBL/” และคำสั่ง “RRC”

- ซอฟต์แวร์ ZAmpPunch.arx จะใช้ได้กับลักษณะที่ต้องการเจาะรูที่มีรูปร่างทั่ว ๆ ไปได้แก่ รูปร่างรูที่เป็นเรขาคณิตง่าย ๆ ได้แก่ วงกลม สี่เหลี่ยม (เฉพาะที่ด้านขนานกับแกน X และแกน Y เท่านั้น) และส่วนโค้ง ซึ่งรูปร่างดังกล่าวสามารถนำมาประยุกต์เพื่อสร้างรูปร่างรูที่ซับซ้อนขึ้นได้โดยนำส่วนต่าง ๆ มารวมกัน

5.2. Drawing ที่เหมาะสมกับการใช้ซอฟต์แวร์

การใช้งานซอฟต์แวร์ ZAmpPunch.arx นั้นจะต้องใช้ร่วมกับ Drawing เฉพาะในรูปแบบ 2 มิติของ AutoCAD R14 ที่มีความเหมาะสมซึ่งเป็นไปตามข้อกำหนดในการสร้างรูปต่าง ๆ ทั้งนี้เพื่อให้ได้รหัสโปรแกรม G Code ที่ถูกต้องกับ Object ที่ต้องการเจาะอย่างแท้จริง การสร้างรูปต่าง ๆ ใน Drawing จึงต้องมีการกำหนดรูปแบบให้เป็นในลักษณะเดียวกัน การสร้างรูปภายใน Drawing ที่ไม่ตรงกับข้อกำหนด อาจทำให้ G Code ที่ได้มีความผิดพลาดหรืออาจทำให้ซอฟต์แวร์หยุดการทำงานได้ ตัวอย่างเช่น ต้องการเจาะรูที่มีรูปร่างเป็นสี่เหลี่ยม การสร้างรูปนั้นจะต้องใช้คำสั่งสร้างสี่เหลี่ยมใน AutoCAD จริง ๆ เพื่อทำให้ซอฟต์แวร์รู้ว่ารูปนั้นเป็น Object สี่เหลี่ยมจริง ๆ การสร้างสี่เหลี่ยมในลักษณะที่ไม่ตรงกับข้อกำหนด เช่น การนำเส้น 4 เส้นมาต่อกันนั้น ซอฟต์แวร์ไม่สามารถตัดสินใจในการสร้างรหัสโปรแกรม G Code สำหรับสี่เหลี่ยมได้ เนื่องจาก Object ที่สร้างขึ้นเป็นเพียงเส้นเท่านั้น

สำหรับข้อกำหนดของ Drawing ที่เหมาะสมกับการใช้ซอฟต์แวร์ถูกแบ่งออกเป็น 3 ส่วน ได้แก่ ข้อกำหนดทั่วไปสำหรับการสร้าง Drawing โดยรวม ข้อกำหนดสำหรับรูภายใน และข้อกำหนดสำหรับรูภายนอก

ข้อกำหนดทั่วไปสำหรับการสร้าง Drawing โดยรวม

1. Drawing ที่สร้างขึ้นหรือถูกคัดลอกมา จะต้องมีตำแหน่ง (Coordinate) ด้านแกน X และแกน Y ของส่วนต่าง ๆ ที่แน่นอนตามที่ลูกค้าต้องการและที่ต้องการให้เครื่องจักร CNC เจาะจริง
2. หาก Drawing ที่นำมาใช้กับโปรแกรมได้จากการคัดลอกมา ผู้ใช้ต้องใช้คำสั่ง “explode” ภายใน AutoCAD เพื่อให้ Drawing ที่คัดลอกมาซึ่งรวมกันอยู่เป็น Object รวมกระจายออกเป็น Object ย่อย
3. หากต้องการใส่ข้อมูลทางด้านขนาด (Dimension) ควรจะสร้าง Layer ขึ้นมาเพื่อเก็บข้อมูลทางขนาดโดยเฉพาะ เพื่อที่จะแยกส่วนของ Object ที่จะใช้กับซอฟต์แวร์ ZAmpPunch.arx และส่วนของขนาด และเมื่อต้องการใช้งานควรจะปิด Layer ของขนาดดังกล่าว เพื่อให้เห็นเพียงแต่ส่วนของ Drawing ที่ต้องนำไปใช้งานจริง
4. เมื่อสร้าง Drawing ที่เป็นไปตามข้อกำหนดของโปรแกรมเรียบร้อยแล้ว ผู้ใช้ควรจะบันทึก (Save) Drawing ไว้ให้เป็นมาตรฐานเอาไว้สำหรับชิ้นงานนั้น ๆ สำหรับใช้ซอฟต์แวร์สร้างรหัสโปรแกรมการเจาะในแบบต่าง ๆ ที่ผู้ใช้ต้องการ นั่นคือเมื่อจะนำมาใช้ก็เปิด Drawing มาใช้กับซอฟต์แวร์ และเมื่อใช้ซอฟต์แวร์กับ Drawing ดังกล่าวเสร็จแล้วก็ไม่ต้องมีการบันทึก Drawing ซ้ำเพื่อที่จะสามารถนำ Drawing เดิมมาใช้ใหม่ได้ถ้าต้องการ

5. ห้ามใช้คำสั่งหมุน (“rotate”) กับ Object ที่จะนำมาใช้กับซอฟต์แวร์

ข้อกำหนดสำหรับรูภายใน

1. Object สำหรับการเจาะภายในจะเป็นได้เฉพาะ วงกลม สี่เหลี่ยม ส่วนโค้ง และ Oblong เท่านั้น หากต้องการสร้างรูปร่างที่ซับซ้อนขึ้นก็สามารถนำ Object เหล่านี้มาประกอบเข้าด้วยกันให้เป็นรูปร่างที่ต้องการ สำหรับ Object ที่ระบุไว้มีรายละเอียดในตารางที่ 5.1
2. Object ที่เป็นรูป Oblong ใช้คำสั่ง “pline” สร้างขึ้นโดยสร้างเส้นตรงสลับกับส่วนโค้ง (ซึ่งกวาดมุม 180 องศา) และการป้อนค่าครั้งสุดท้ายให้ใช้คำสั่ง “close” เพื่อทำให้เป็น Object ที่ปิด

ข้อกำหนดสำหรับรูภายนอก

1. Object สำหรับการเจาะภายนอกจะเป็นได้เฉพาะเส้นเท่านั้น รายละเอียดถูกระบุไว้ในตารางที่ 5.2
2. Object ที่เป็นมุมจะต้องเกิดจากคำสั่ง Polyline (“pline”) ที่สร้างเส้นตรงสองเส้นที่ตั้งฉากกันเท่านั้น ห้ามเกิดจากการที่สร้างเส้นตรงมากกว่าหรือน้อยกว่าสองเส้น (จากคำสั่ง “pline”)
3. Object ที่เป็นเส้นขอบสามเหลี่ยมที่ด้านข้างสำหรับพับจะต้องเกิดจากคำสั่ง “pline” ที่สร้างเส้นตรงสองเส้นที่ตั้งฉากกันเท่านั้น และแต่ละเส้นจะต้องเอียงในลักษณะ 45 องศา

ตารางที่ 5.1 *ข้อกำหนดสำหรับ Object ใน Drawing สำหรับการเจาะภายใน*

การเจาะภายใน		
Object	**คำสั่งใน AutoCAD ที่ใช้สร้าง Object**	**ค่าที่ต้องป้อนเข้า**
1. วงกลม	“circle”	จุดศก. และ รัศมี (หรือเส้นผ่านศก.)
2. สี่เหลี่ยม	“rectangle”	จุดมุมเริ่ม และ จุดมุมตรงข้าม
3. ส่วนโค้ง	“arc”	จุดศก. จุดเริ่ม และ จุดสุดท้าย
4. Oblong	“pline”	จุดเริ่ม จุดต่อไป และจุดสิ้นสุด

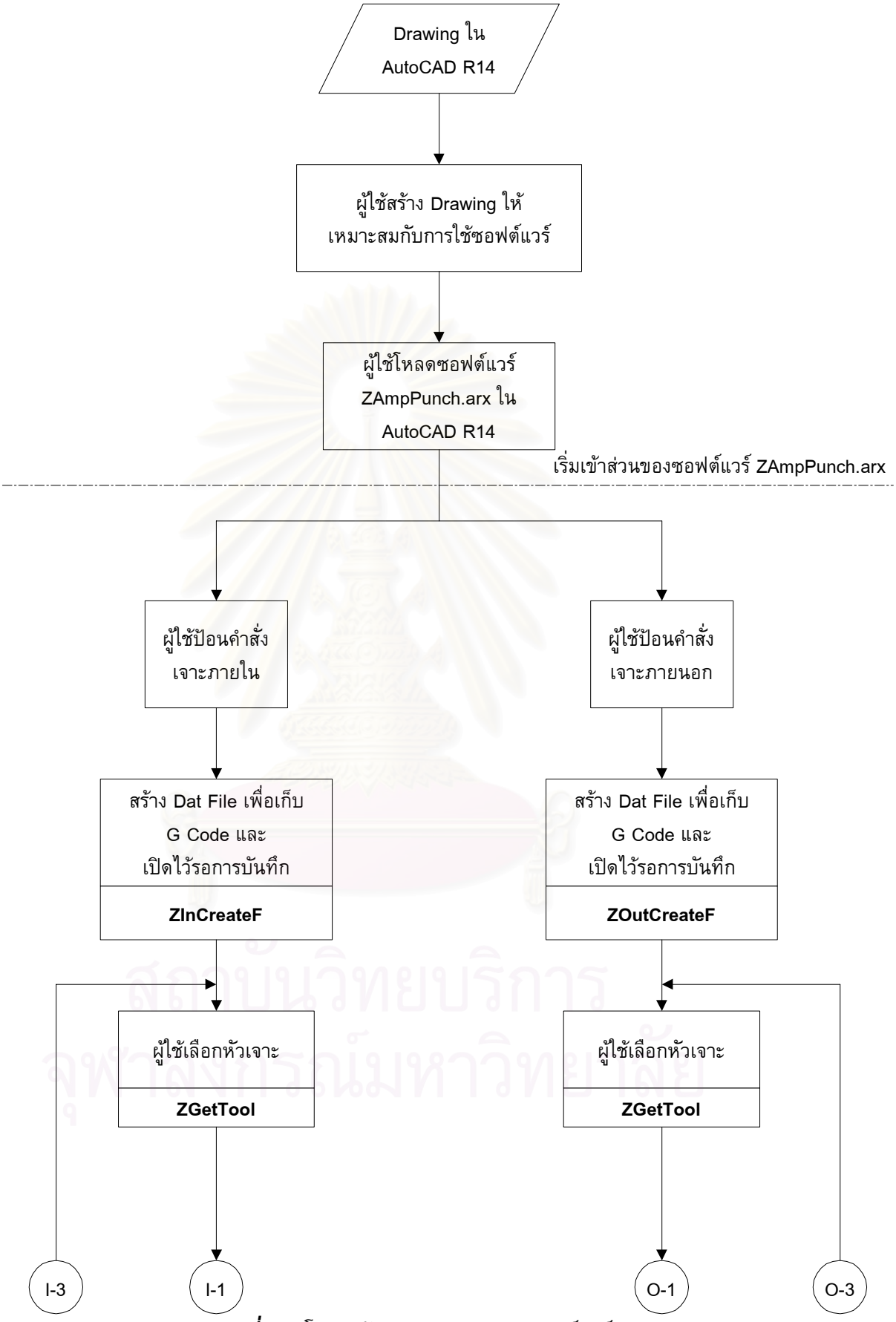
ตารางที่ 5.2 *ข้อกำหนดสำหรับ Object ใน Drawing สำหรับการเจาะภายนอก*

การเจาะภายนอก		
Object	**คำสั่งใน AutoCAD ที่ใช้สร้าง Object**	**ค่าที่ต้องป้อนเข้า**
1. เส้นขอบ	“line”	จุดเริ่ม และจุดสิ้นสุด
2. เส้นมุม	“pline”	จุดเริ่ม จุดต่อไป และจุดสิ้นสุด
3. เส้นขอบสามเหลี่ยมด้านข้าง	“pline”	จุดเริ่ม จุดต่อไป และจุดสิ้นสุด

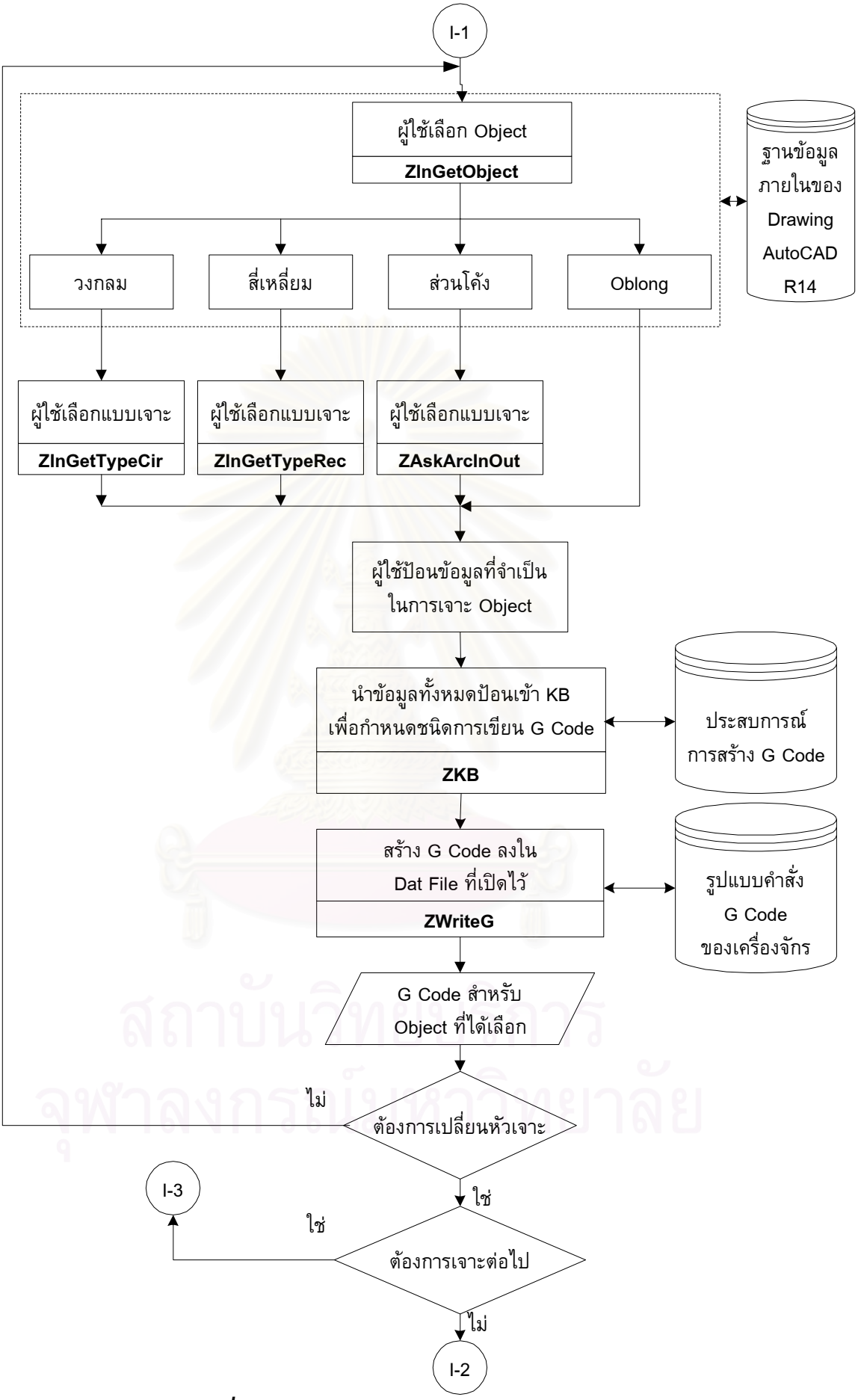
5.3. โครงสร้างการทำงาน

ในการใช้นำคอมพิวเตอร์เข้ามาช่วยกระบวนการสร้างรหัสโปรแกรม G Code จะเป็นการใช้ซอฟต์แวร์ ZAmpPunch.arx ร่วมกับ Drawing ของ AutoCAD ที่เหมาะสมกับการใช้ซอฟต์แวร์ในสภาพแวดล้อมการทำงานของ AutoCAD โดยใช้คำสั่งที่สร้างขึ้นเพื่อการเจาะภายในและการเจาะภายนอก

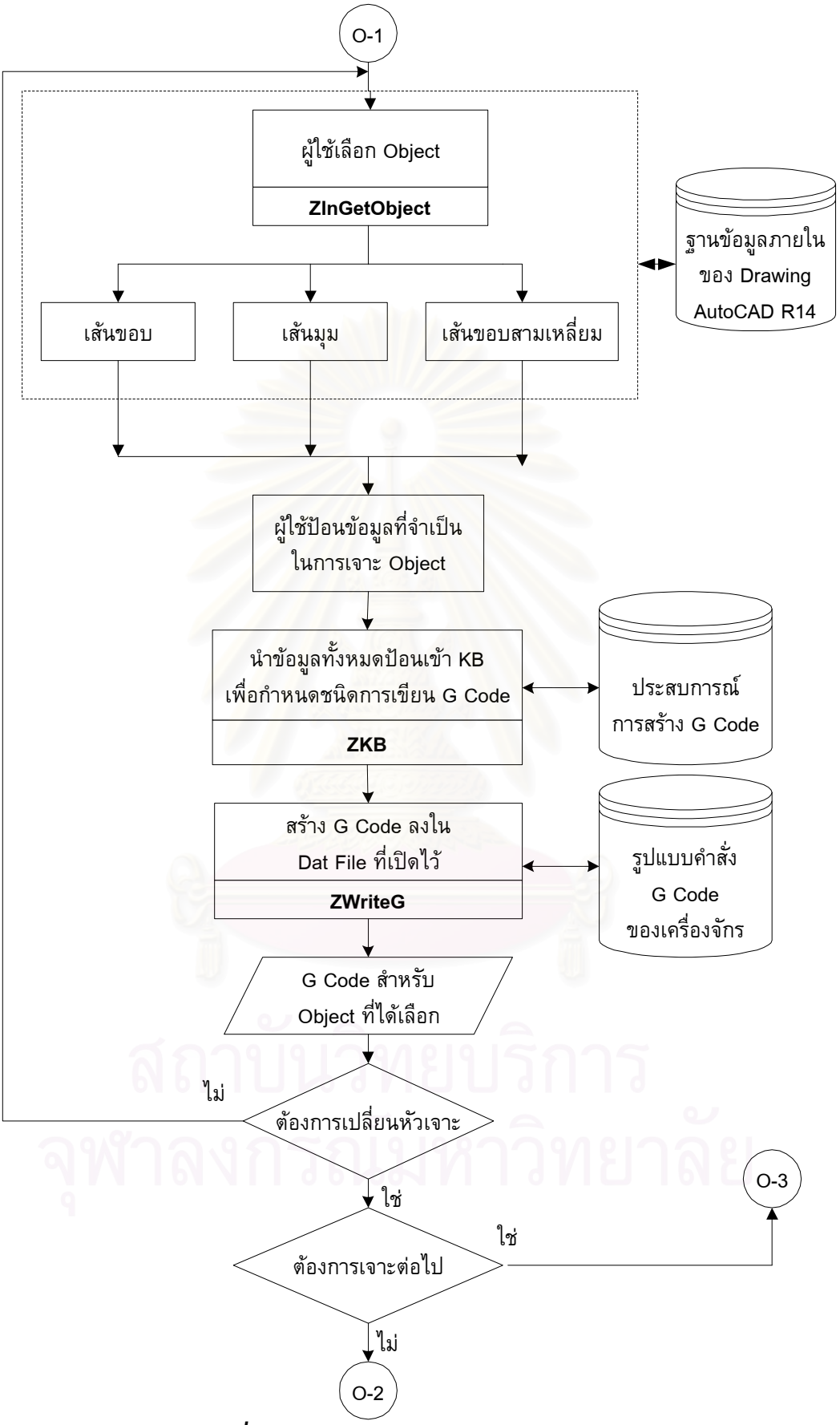
หลักการของการทำงานของ ZAmpPunch.arx คือ ซอฟต์แวร์จะทำการเขียนรหัสคำสั่ง G Code ทีละบรรทัด ต่อ ๆ กันไปจนได้รหัสคำสั่งโปรแกรม G Code สำหรับการเจาะภายในและการเจาะภายนอก โดยที่รหัสคำสั่ง G Code ที่เกิดขึ้นในแต่ละบรรทัดเกิดจากการที่ผู้วางแผนได้เลือก Object ที่อยู่ใน Drawing ที่เหมาะสมกับซอฟต์แวร์ทีละ Object และการป้อนข้อมูลที่จำเป็นต่อความต้องการในการเจาะของผู้วางแผนในแต่ละคำสั่ง โครงสร้างการทำงานภายใน ZAmpPunch.arx แสดงได้ใน Flow Chart ในรูปที่ 5.1



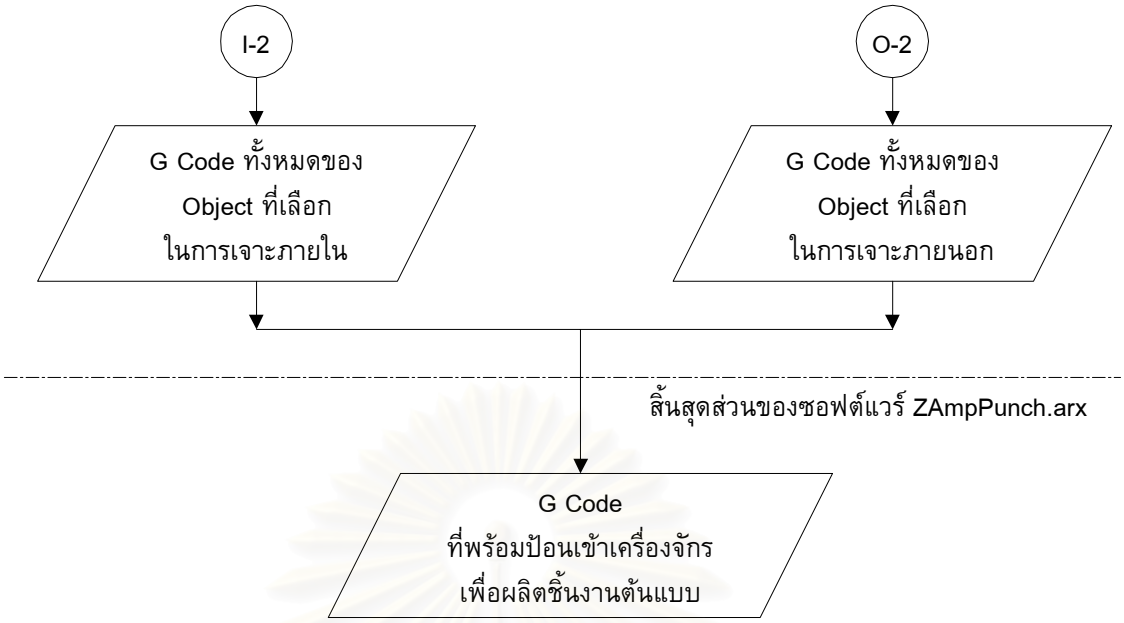
รูปที่ 5.1 *โครงสร้างการทำงานของซอฟต์แวร์*



รูปที่ 5.1(ต่อ) *โครงสร้างการทำงานของซอฟต์แวร์*



รูปที่ 5.1 (ต่อ) *โครงสร้างการทำงานของซอฟต์แวร์*



รูปที่ 5.1 (ต่อ) *โครงสร้างการทำงานของซอฟต์แวร์*

สำหรับซอฟต์แวร์ ZAmpPunch.arx สร้างขึ้นจากการโปรแกรมมิ่งโดยภาษา C++ โดยใช้เครื่องมือในการพัฒนาคือ Visual C++ 6.0 ซึ่งเป็นซอฟต์แวร์แบบ DLLs ซึ่งถูกสร้างขึ้นโดยกระบวนการของภาษา C++ ซึ่งต้องอาศัยส่วนประกอบสำคัญประกอบกันขึ้นมา ส่วนประกอบดังกล่าวและรหัสคำสั่งของส่วนประกอบเหล่านั้นรวบรวมอยู่ในภาคผนวก ค. โดยที่ส่วนประกอบที่เป็นกลไกหลักในการทำงานคือ ZAmpPunch.cpp ซึ่งประกอบด้วยฟังก์ชั่นในการทำงานต่าง ๆ ซึ่งแบ่งออกได้เป็น 3 ส่วนได้แก่ ฟังก์ชั่นสำหรับการเจาะภายใน ฟังก์ชั่นสำหรับเจาะภายนอก และฟังก์ชั่นที่ใช้ร่วมกันระหว่างการเจาะภายในและภายนอก ซึ่งมีรายละเอียดดังต่อไปนี้

ฟังก์ชั่นสำหรับการเจาะภายใน

1. InPunch() มีหน้าที่เริ่มต้นในการสร้าง (Initialize) Object ของ C++ ของการเจาะภายในที่เก็บตัวแปรภายในการใช้งานทั้งหมด
2. ZInMain(ZPunch &zObj) มีหน้าที่ควบคุมกลไกหลักในการเจาะภายในภายในซอฟต์แวร์ทั้งหมด
3. ZInCreateF(ZPunch &zObj) มีหน้าที่สร้าง Dat File ตอนเริ่มต้นเพื่อเก็บ G Code ของแต่ละ Object ของการเจาะภายใน
4. ZInGetObject(ZPunch &zObj) มีหน้าที่รับ Object ใน AutoCAD ของการเจาะภายในที่ผู้ใช้เลือกและทำการแปลงค่าพิกัดของ Object ดังกล่าวให้อยู่ในรูปแบบมาตรฐานเพื่อส่งต่อไปยังฟังก์ชั่น ZKB(ZPunch &zObj) เพื่อคำนวณให้ถูกต้อง
5. ZInGetTypeCir(ZPunch &zObj) มีหน้าที่รับค่าของแบบการเจาะวงกลม

6. ZInAskToolDia(ZPunch &zObj) มีหน้าที่รับค่าของขนาดเส้นผ่านศก.ของหัวเจาะสำหรับคำสั่งที่ต้องใช้ค่านี้ในการคำนวณ เช่น “HOL/” เป็นต้น
7. ZInGetTypeRec(ZPunch &zObj) มีหน้าที่รับค่าของแบบการเจาะสี่เหลี่ยม
8. ZInGetRecRef(ZPunch &zObj) มีหน้าที่รับค่าของจุดอ้างอิงในการเจาะสี่เหลี่ยมโดยให้ผู้วางแผนเลือกใน Drawing ของ AutoCAD
9. ZInAskCornRad(ZPunch &zObj) มีหน้าที่รับค่าของขนาดรัศมีของมุมสี่เหลี่ยมในแบบการเจาะโดยคำสั่ง “RRC/”
10. ZAskArcInOut(ZPunch &zObj) มีหน้าที่รับค่าของด้านที่จะเจาะส่วนโค้งของคำสั่ง “RAD/”

ฟังก์ชั่นสำหรับเจาะภายนอก

1. OutPunch() มีหน้าที่มีหน้าที่เริ่มต้นในการสร้าง (Initialize) Object ของ C++ ของการเจาะภายนอกที่เก็บตัวแปรภายในการใช้งานทั้งหมด
2. ZOutMain(ZPunch &zObj) มีหน้าที่มีหน้าที่ควบคุมกลไกหลักในการเจาะภายนอกภายในซอฟต์แวร์ทั้งหมด
3. ZOutCreateF(ZPunch &zObj) มีหน้าที่มีหน้าที่สร้าง Dat File ตอนเริ่มต้นเพื่อเก็บ G Code ของแต่ละ Object ของการเจาะภายนอก
4. ZOutGetObject(ZPunch &zObj) มีหน้าที่มีหน้าที่รับ Object ใน AutoCAD ของการเจาะภายนอกที่ผู้ใช้เลือกและทำการแปลงค่าพิกัดของ Object ดังกล่าวให้อยู่ในรูปแบบมาตรฐานเพื่อส่งต่อไปยังฟังก์ชั่น ZKB(ZPunch &zObj) เพื่อคำนวณให้ถูกต้อง
5. ZOutGetLineRef(ZPunch &zObj) มีหน้าที่รับค่าของจุดอ้างอิงในการเจาะเส้นขอบโดยให้ผู้วางแผนเลือกใน Drawing ของ AutoCAD
6. ZOutGetLineDirec(ZPunch &zObj) มีหน้าที่รับค่าของตำแหน่งที่ต้องการเจาะขอบ
7. ZOutAskForJoint(ZPunch &zObj) มีหน้าที่รับค่าที่ต้องการให้มี Microjoint บนเส้นขอบหรือไม่
8. ZCreateJoint(ZPunch &zObj) มีหน้าที่สร้างรูปของ Microjoint ให้อยู่ใน Drawing ของ AutoCAD

ฟังก์ชั่นที่ใช้ร่วมกันระหว่างการเจาะภายในและภายนอก

1. ZGetTool(ZPunch &zObj) มีหน้าที่รับค่าตำแหน่งของหัวเจาะที่จะใช้เจาะ Object ที่จะเลือก
2. ZKB(ZPunch &zObj) มีหน้าที่คำนวณค่าพิกัดของ Object เพื่อส่งข้อมูลที่จะต้องเขียนไปยังฟังก์ชั่น ZWriteG(ZPunch &zObj)

3. ZWriteG(ZPunch &zObj) มีหน้าที่กำหนดรูปแบบรหัส G Code จากข้อมูลที่ส่งมาจาก ฟังก์ชั่น ZKB(ZPunch &zObj) เพื่อเขียนรหัส G Code ลงใน Dat File ที่สร้างไว้
4. ZLoopAsk(ZPunch &zObj) มีหน้าที่รับค่าจากผู้วางแผนเมื่อสิ้นสุดการสร้างรหัส G Code ในแต่ละ Object
5. ZAskToolDim(ZPunch &zObj) มีหน้าที่รับค่าขนาดของหัวเจาะในด้านแกน X และแกน Y

สำหรับการติดตั้งและการใช้งานซอฟต์แวร์ ZAmpPunch.arx นั้นถูกรวบรวมอยู่ในภาคผนวก ง.

5.4. สรุป

ซอฟต์แวร์ ZAmpPunch.arx มีหน้าที่สร้างรหัสคำสั่ง G Code สำหรับการเจาะภายในและการเจาะภายนอก โดยต้องทำงานร่วมกับ Drawing ที่ถูกสร้างขึ้นให้มีความเหมาะสมในการใช้ซอฟต์แวร์ในสภาพการทำงานบนโปรแกรม AutoCAD เพื่อให้รหัสโปรแกรมดังกล่าวมีความถูกต้องมากที่สุด การทำงานของซอฟต์แวร์มีหลักการในการทำงาน คือ ให้ผู้ใช้เลือก Object ทีละ Object ที่ต้องการ แล้วป้อนค่าที่จำเป็นของคำสั่ง G Code ที่ต้องการ จากนั้นซอฟต์แวร์จะคำนวณและสร้างเป็นรหัสคำสั่ง G Code เขียนลงใน Dat File สำหรับการเจาะภายในและการเจาะภายนอกแยกจากกัน เมื่อนำทั้งสองส่วนมารวมกันก็จะได้เป็นรหัสโปรแกรม G Code ที่พร้อมจะนำเข้าเครื่องจักร CNC เพื่อเจาะชิ้นงานต้นแบบที่สมบูรณ์

บทที่ 6

การทดสอบความถูกต้องของซอฟต์แวร์

เนื้อหาในบทนี้เกี่ยวกับการทดสอบความถูกต้องของซอฟต์แวร์ช่วยวางแผนกระบวนการผลิตสำหรับการเจาะของเครื่องจักร CNC Turret Punch Press MURATA C2500 โดยจะทดสอบกับ Drawing ของชิ้นงานทดสอบที่มีลักษณะ 3 แบบ ที่ได้ถูกสร้างขึ้นเพื่อให้เหมาะสมกับการใช้งานซอฟต์แวร์ ซึ่งใน Drawing ที่นำมาทดสอบจะมีรูปของชิ้นงานซึ่งประกอบด้วยรูลักษณะต่าง ๆ ที่อยู่ในขอบเขตของซอฟต์แวร์ที่จะสร้าง G Code ได้ การตรวจสอบจะทำโดยการแสดงรหัสโปรแกรม G Code ให้เห็นและการนำรหัสโปรแกรมดังกล่าวป้อนเข้าไปยังเครื่องจักร CNC เพื่อทดลองผลิต เพื่อให้เห็นว่า G Code ที่สร้างขึ้นด้วยการใช้ซอฟต์แวร์ช่วยทำให้ได้ชิ้นงานเหมือนกับใน Drawing ของ AutoCAD ที่ได้ออกแบบไว้

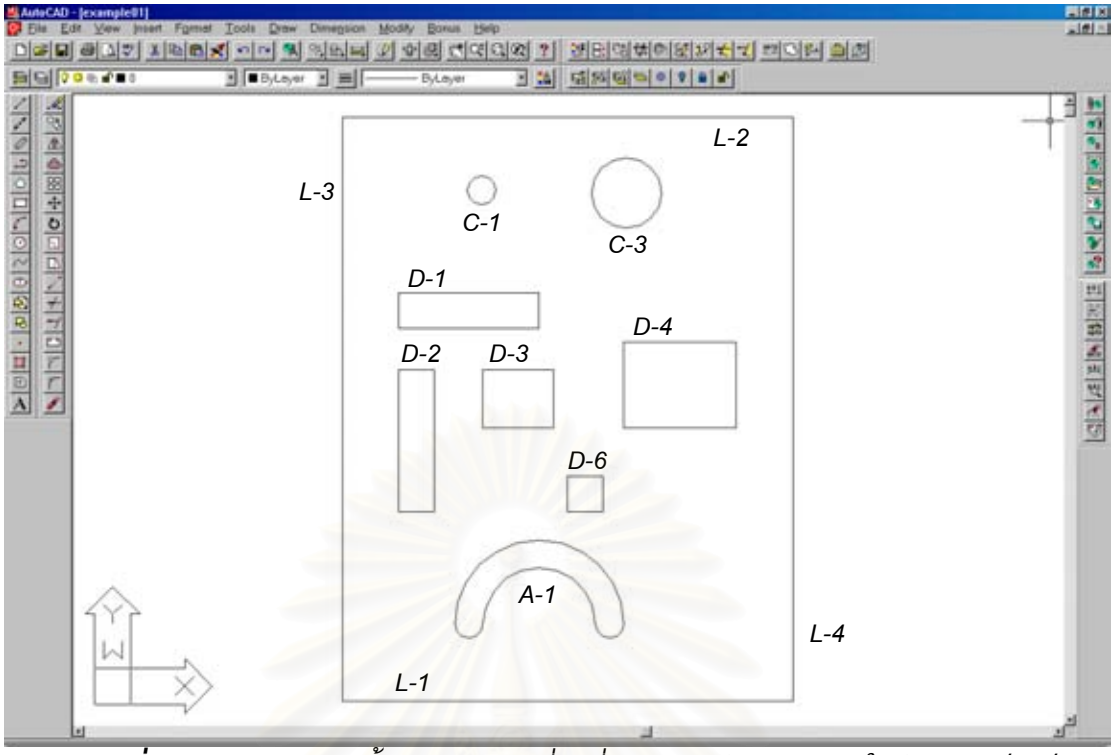
6.1. ลักษณะของชิ้นงานทดสอบ

ชิ้นงานที่นำมาทำการทดสอบเป็น Drawing ของแผ่นโลหะซึ่งถูกสร้างขึ้นภายใน AutoCAD ตามข้อกำหนดทั้งหมดที่เหมาะสมกับการนำมาใช้กับซอฟต์แวร์ ชิ้นงานที่นำมาทดสอบมีทั้งหมด 4 ชิ้นงานซึ่งมีลักษณะแตกต่างกันไป ในบทนี้จะนำเสนอสำหรับ 3 ชิ้นงานทดสอบซึ่งมีการเจาะที่ไม่ซับซ้อนมาก สำหรับอีกหนึ่งตัวอย่างจะมีความซับซ้อนอยู่พอสมควร ซึ่งได้รวบรวมไว้ในภาคผนวก จ.

ในการสร้าง Drawing ทั้งหมดจะใช้หน่วยวัดเป็นมิลลิเมตรเพื่อให้ตรงกับหน่วยที่ใช้กับเครื่องจักร CNC

6.1.1. ชิ้นงานทดสอบที่ 1

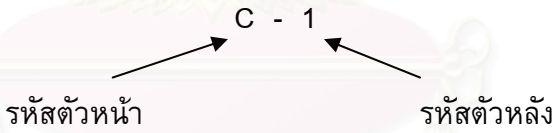
การเจาะชิ้นงานทดสอบที่ 1 เป็นการแสดงการเจาะรูที่มีรูปร่างเรขาคณิตง่าย ๆ โดยลักษณะการเจาะรูจะเป็นการใช้คำสั่งที่เป็นรูปแบบในการเจาะแตกต่างกันไปตามความเหมาะสมของลักษณะรูที่จะเจาะ ลักษณะของชิ้นงานทดสอบที่ 1 ที่สร้างไว้ใน AutoCAD เพื่อใช้กับซอฟต์แวร์แสดงในรูปที่ 6.1 โดยมีรหัสของแต่ละ Object เพื่อใช้ในการอธิบายในภายหลัง



รูปที่ 6.1 Drawing ของชิ้นงานทดสอบที่ 1 ที่เหมาะสมกับการนำมาใช้กับซอฟต์แวร์

รายละเอียดของชิ้นงานทดสอบที่ 1 สรุปได้ดังนี้

- รหัสที่ใช้มีรายละเอียดแสดงในรูปที่ 6.2 และตารางที่ 6.1



รูปที่ 6.2 แสดงรหัสของ Object ในชิ้นงานทดสอบ

ตารางที่ 6.1 รายละเอียดของรหัสของ Object ในชิ้นงานทดสอบที่ 1

รหัสตัวหน้า	รหัสตัวหลัง	ความหมาย	หมายเหตุ
A	1	ส่วนโค้ง / เจาะภายใน	คำสั่ง “RAD/”
C	1	วงกลม / เจาะแบบที่ 1	เจาะครั้งเดียวตำแหน่งศก.หัวเจาะ
	3	วงกลม / เจาะแบบที่ 3	คำสั่ง “OPN/”
D	1	สี่เหลี่ยม / เจาะแบบที่ 1	คำสั่ง “REC/” แนวนอน
	2	สี่เหลี่ยม / เจาะแบบที่ 2	คำสั่ง “REC/” แนวตั้ง
	3	สี่เหลี่ยม / เจาะแบบที่ 3	คำสั่ง “REC/” แนวนอนและตั้ง
	4	สี่เหลี่ยม / เจาะแบบที่ 4	คำสั่ง “OBL/”

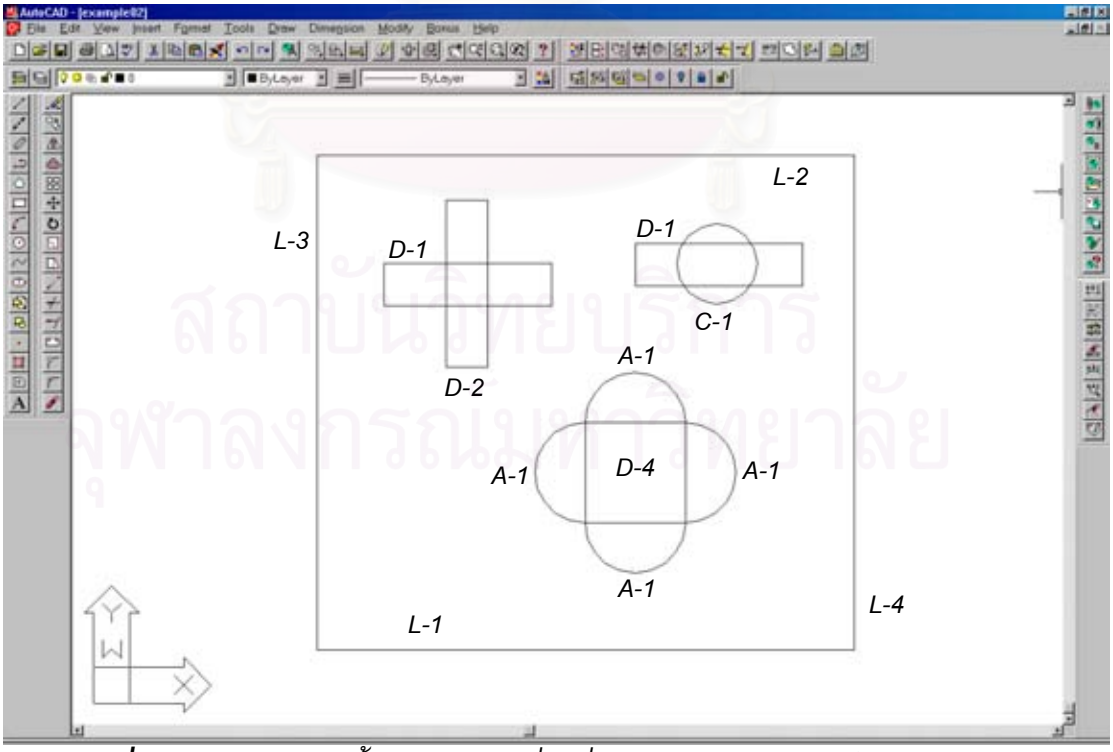
ตารางที่ 6.1 (ต่อ) *รายละเอียดของรหัสของ Object ในชิ้นงานทดสอบที่ 1*

รหัสตัวหน้า	รหัสตัวหลัง	ความหมาย	หมายเหตุ
D	6	สี่เหลี่ยม / เจาะแบบที่ 6	เจาะครั้งเดียวตำแหน่งศก.หัวเจาะ
L	1	เส้นตรง / เจาะส่วนแรก	คำสั่ง “REC/” แนวนอน
	2	เส้นตรง / เจาะส่วนที่สอง	คำสั่ง “REC/” แนวนอน
	3	เส้นตรง / เจาะส่วนที่สาม	คำสั่ง “REC/” แนวตั้ง
	4	เส้นตรง / เจาะส่วนที่สี่	คำสั่ง “REC/” แนวตั้ง

- มีส่วนของ Microjoint ที่ถูกกำหนดขึ้นเพื่อทำให้ชิ้นงานไม่ตกหล่นลงบนโต๊ะซึ่งกำหนดเอาไว้ที่มุมทั้ง 4 ของชิ้นงาน

6.1.2. ชิ้นงานทดสอบที่ 2

การเจาะชิ้นงานทดสอบที่ 2 เป็นการแสดงการเจาะรูที่มีรูปร่างซับซ้อนขึ้นซึ่งเกิดจากการนำเอารูปร่างเรขาคณิตมาประกอบกัน ซึ่งจะใช้คำสั่งที่เป็นรูปแบบตามความเหมาะสมของรูปร่างของรูเช่นเดียวกัน ลักษณะของชิ้นงานทดสอบที่ 2 ที่สร้างไว้ใน AutoCAD เพื่อใช้กับซอฟต์แวร์แสดงในรูปที่ 6.3 โดยมีรหัสของแต่ละ Object เพื่อใช้ในการอธิบาย



รูปที่ 6.3 *Drawing ของชิ้นงานทดสอบที่ 2 ที่เหมาะสมกับการนำมาใช้กับซอฟต์แวร์*

รายละเอียดของชิ้นงานทดสอบที่ 2 สรุปได้ดังนี้

- รหัสที่ใช้มีรายละเอียดแสดงในรูปที่ 6.2 และตารางที่ 6.2

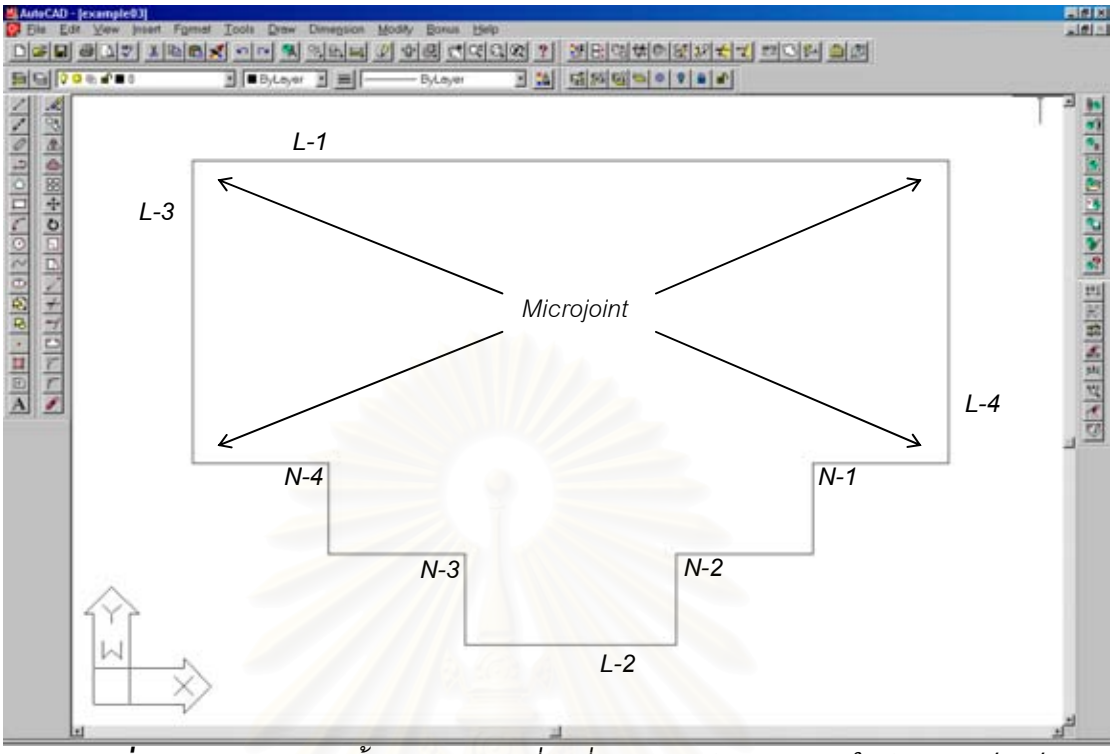
ตารางที่ 6.2 *รายละเอียดของรหัสของ Object ในชิ้นงานทดสอบที่ 2*

รหัสตัวหน้า	รหัสตัวหลัง	ความหมาย	หมายเหตุ
A	1	ส่วนโค้ง / เจาะภายใน	คำสั่ง “RAD/”
C	1	วงกลม / เจาะแบบที่ 1	เจาะครั้งเดียวตำแหน่งศก.หัวเจาะ
D	4	สี่เหลี่ยม / เจาะแบบที่ 4	คำสั่ง “OBL/”
L	1	เส้นตรง / เจาะส่วนแรก	คำสั่ง “REC/” แนวนอน
	2	เส้นตรง / เจาะส่วนที่สอง	คำสั่ง “REC/” แนวนอน
	3	เส้นตรง / เจาะส่วนที่สาม	คำสั่ง “REC/” แนวตั้ง
	4	เส้นตรง / เจาะส่วนที่สี่	คำสั่ง “REC/” แนวตั้ง

- มีส่วนของ Microjoint ที่ถูกกำหนดขึ้นเพื่อทำให้ชิ้นงานไม่ตกหล่นลงบนโต๊ะซึ่งกำหนดเอาไว้ที่มุมทั้ง 4 ของชิ้นงาน

6.1.3. ชิ้นงานทดสอบที่ 3

การเจาะชิ้นงานทดสอบที่ 3 เป็นการแสดงการเจาะแผ่นโลหะในลักษณะการเจาะภายนอกเพื่อให้ได้รูปร่างตามต้องการ ลักษณะของชิ้นงานทดสอบที่ 3 ที่สร้างไว้ใน AutoCAD เพื่อใช้กับซอฟต์แวร์แสดงในรูปที่ 6.4 โดยมีรหัสของแต่ละ Object เพื่อใช้ในการอธิบาย



รูปที่ 6.4 *Drawing ของชิ้นงานทดสอบที่ 3 ที่เหมาะสมกับการนำมาใช้กับซอฟต์แวร์*

รายละเอียดของชิ้นงานทดสอบที่ 3 สรุปได้ดังนี้

- รหัสที่ใช้มีรายละเอียดแสดงในรูปที่ 6.2 และตารางที่ 6.3

ตารางที่ 6.3 *รายละเอียดของรหัสของ Object ในชิ้นงานทดสอบที่ 3*

รหัสตัวหน้า	รหัสตัวหลัง	ความหมาย	หมายเหตุ
L	1	เส้นตรง / เจาะส่วนแรก	คำสั่ง “REC/” แนวนอน
	2	เส้นตรง / เจาะส่วนที่สอง	คำสั่ง “REC/” แนวนอน
	3	เส้นตรง / เจาะส่วนที่สาม	คำสั่ง “REC/” แนวตั้ง
	4	เส้นตรง / เจาะส่วนที่สี่	คำสั่ง “REC/” แนวตั้ง
N	1	ขอบมุม / เจาะส่วนแรก	คำสั่ง “REC/” แนวนอนและตั้ง
	2	ขอบมุม / เจาะส่วนที่สอง	คำสั่ง “REC/” แนวนอนและตั้ง
	3	ขอบมุม / เจาะส่วนที่สาม	คำสั่ง “REC/” แนวนอนและตั้ง
	4	ขอบมุม / เจาะส่วนสี่	คำสั่ง “REC/” แนวนอนและตั้ง

- มีส่วนของ Microjoint ที่ถูกกำหนดขึ้นเพื่อทำให้ชิ้นงานไม่ตกหล่นลงบนโต๊ะซึ่งกำหนดเอาไว้ที่มุมดังรูปที่ 6.4

6.2. การสร้างรหัสโปรแกรม G Code โดยซอฟต์แวร์

เมื่อได้ Drawing จากการสร้างชิ้นงานทดสอบที่เหมาะสมกับการใช้ซอฟต์แวร์แล้ว ขั้นตอนต่อไปผู้วางแผนจะต้องพิจารณาว่ารูที่มีอยู่ทั้งหมดจะต้องใช้หัวเจาะอะไรบ้าง แล้ววางแผนการติดตั้งหัวเจาะเหล่านั้นลงบนตำแหน่งหัวเจาะบนแท่นหมุน สำหรับชิ้นงานทดสอบทั้ง 3 ชิ้นงาน การวางแผนด้านการติดตั้งหัวเจาะแสดงในตารางที่ 6.4

ตารางที่ 6.4 *การวางแผนการเจาะด้านติดตั้งหัวเจาะลงบนแท่นหมุนสำหรับชิ้นงานทดสอบ*

ตำแหน่งบนแท่นหมุน	รูปร่างของ หัวเจาะ	ขนาด
01 (หมุนได้)	สี่เหลี่ยมผืนผ้า	40.00 x 5.00 มม. (X x Y)
08	วงกลม	∅ 10.00 มม.
11	สี่เหลี่ยมจัตุรัส	12.50 x 12.50 มม.
15	วงกลม	∅ 24.00 มม.

ลักษณะการเจาะที่ซอฟต์แวร์ได้สร้างรหัสโปรแกรม G Code ขึ้นนั้นจะกระทำโดยใช้หัวเจาะตัวหนึ่งสำหรับ Object ต่าง ๆ ให้เสร็จก่อนแล้วจึงเปลี่ยนหัวเจาะ ซึ่งในแต่ละชิ้นงานทดสอบจะมีการใช้หัวเจาะที่ได้ติดตั้งแตกต่างกันออกไป

6.2.1. ชิ้นงานทดสอบที่ 1

การใช้หัวเจาะที่ติดตั้งเรียบร้อยแล้วสำหรับ Object ต่าง ๆ ภายในชิ้นงานทดสอบที่ 1 ได้ถูกวางแผนการใช้หัวเจาะไว้ในตารางที่ 6.5

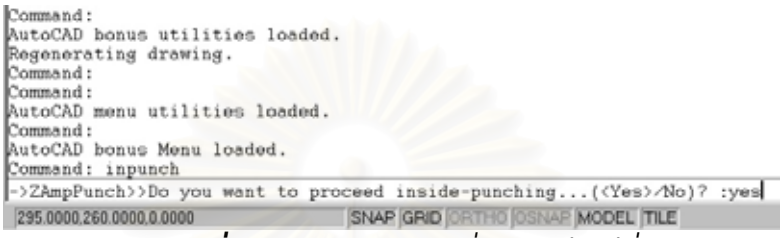
ตารางที่ 6.5 *การวางแผนด้านการใช้หัวเจาะสำหรับชิ้นงานทดสอบที่ 1*

ลำดับการใช้หัวเจาะ	ตำแหน่งหัวเจาะบนแท่นหมุน	Object ที่จะเจาะ (ตามลำดับ)
1	08	C-1, C-3 และ A-1
2	11	D-1, D-2, D-3, D-4 และ D-6
3	01 (หมุนได้)	L-1, L-2, L-3 และ L-4

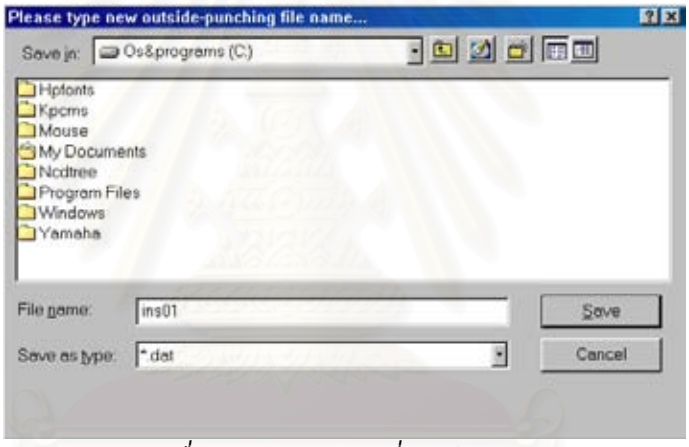
จากนั้นเริ่มใช้ซอฟต์แวร์เพื่อช่วยในการสร้างรหัสโปรแกรม G Code ซึ่งการเจาะชิ้นงานทดสอบจะเริ่มจากการเจาะภายใน แล้วตามด้วยการเจาะภายนอก ขั้นตอนมีดังนี้

การเจาะภายใน

- ป้อนคำสั่ง “inpunch” เพื่อเจาะ Object ภายใน ที่ AutoCAD Command Prompt
- ซอฟต์แวร์จะถามว่าต้องการจะทำต่อหรือไม่ ดังรูปที่ 6.5 ถ้าทำต่อ มันก็จะเข้าไปสร้างแฟ้มข้อมูลที่เป็นตัวอักษรขึ้นเพื่อพร้อมที่จะเขียน G Code ลงไป โดยที่ผู้ใช้ตั้งชื่อแฟ้มที่ต้องการจัดเก็บ ลงใน Dialog Box ซึ่งแฟ้มข้อมูลที่สร้างขึ้นมีนามสกุลเป็น “dat” ดังรูปที่ 6.6 และจากนั้นกด Save

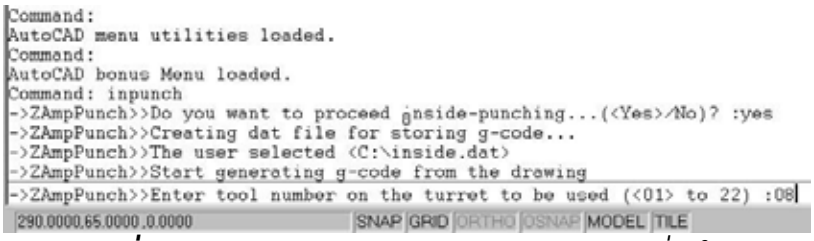


รูปที่ 6.5 *แสดงข้อความเมื่อซอฟต์แวร์เริ่ม*



รูปที่ 6.6 *แสดงหน้าจอรับชื่อของแฟ้มข้อมูลที่จะเก็บ G Code ของการเจาะภายใน*

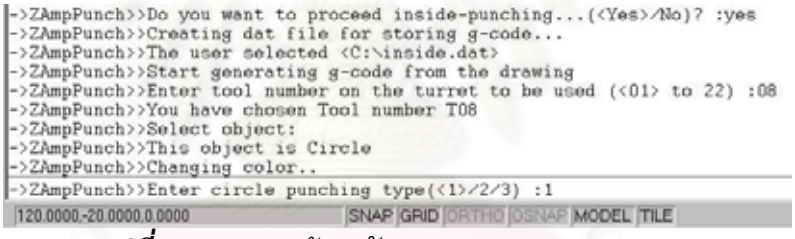
- ลักษณะการเจาะนั้นจะกระทำโดยใช้หัวเจาะตัวหนึ่งสำหรับ Object ต่าง ๆ ให้เสร็จก่อนแล้วจึงเปลี่ยนหัวเจาะ ดังนั้นต่อมาซอฟต์แวร์จะถามว่าต้องการจะใช้หัวเจาะหมายเลขใดในการเจาะก่อนที่จะเลือก Object ที่จะเจาะ ซึ่งเมื่อใช้หัวเจาะตัวนั้นเสร็จแล้วก็จะมีการเปลี่ยนหัวเจาะ หมายเลขที่ป้อนจะเป็น 01 ถึง 22 ส่วนต่อไปจะอธิบายย่อยสำหรับหัวเจาะแต่ละหมายเลขในการเจาะ Object ของชิ้นงานทดสอบที่ 1 ซึ่งเริ่มจากการใช้หัวเจาะหมายเลข 08 ดังรูปที่ 6.7



รูปที่ 6.7 *แสดงหน้าจอรับค่าของตำแหน่งหัวเจาะที่จะใช้*

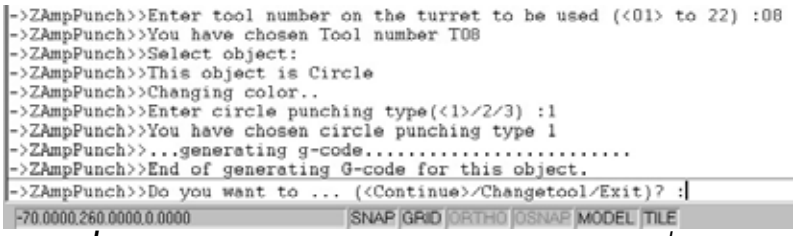
การเจาะภายในด้วยหัวเจาะหมายเลข 08

- ซอฟต์แวร์จะให้ผู้ใช้เลือก Object ที่ต้องการใช้หัวเจาะหมายเลข 08 ในการเจาะ ซอฟต์แวร์จะให้ผู้ใช้เป็นผู้กำหนดเส้นทางในการเจาะว่าต้องการจะเจาะ Object ใดก่อนหลัง ลำดับจะขึ้นอยู่กับการวางแผนในตารางที่ 6.5 ซึ่งจะเริ่มจากการเจาะวงกลม C-1 ก่อน
- เมื่อผู้ใช้เลือกวงกลม C-1 แล้ว ซอฟต์แวร์จะบอกว่า Object ที่เลือกเป็นวงกลม สีของ Object ที่เลือกจะเปลี่ยนเป็นสีฟ้าเช่นเดียวกับ Object ต่าง ๆ ที่จะถูกเลือก ต่อมามันจะถามว่าต้องการเจาะวงกลมในลักษณะใด สำหรับวงกลม C-1 จะเจาะแบบที่ 1 ซึ่งก็ป้อน 1 เข้าไป ดังรูปที่ 6.8



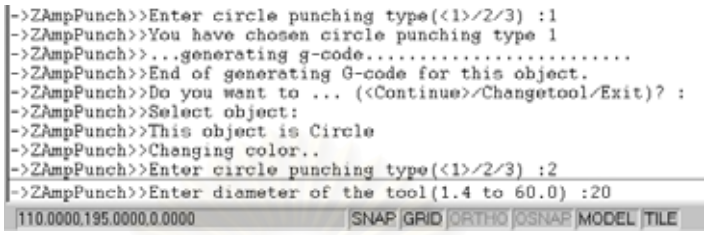
รูปที่ 6.8 *แสดงหน้าจอรับค่าแบบของการเจาะวงกลม*

- สำหรับ C-1 เป็นการเจาะครั้งเดียวในตำแหน่งจุดศก.ของหัวเจาะ ดังนั้นไม่ต้องป้อนค่าอะไรเพิ่มเติม ซอฟต์แวร์จะสร้าง G Code สำหรับ C-1 ไว้ในแฟ้มที่สร้างไว้เรียบร้อยแล้ว ซึ่งผลจากการสร้าง G Code สำหรับแต่ละ Object เสร็จทุกครั้งจะมีรูปแบบดังรูปที่ 6.9 ซึ่งมันจะถามต่อไปว่าต้องการจะใช้หัวเจาะนี้เจาะต่อ (ป้อน “C”) หรือต้องการจะเปลี่ยนหัวเจาะ (ป้อน “Ch”) หรือจะสิ้นสุดการใช้ซอฟต์แวร์ (ป้อน “E”)



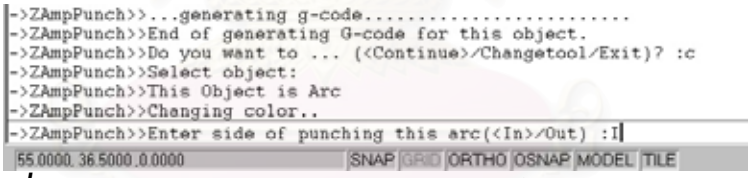
รูปที่ 6.9 *แสดงหน้าจอหลังจากจบการเจาะ Object หนึ่ง ๆ แล้ว*

- ป้อน “C” เนื่องจากต้องใช้หัวเจาะตัวนี้เจาะต่อ สำหรับ Object ต่อไปที่จะต้องเลือกคือวงกลม C-3 ซึ่งจะเจาะแบบที่ 3 (คำสั่ง “OPN/”) ซึ่งก็ป้อนค่า “3” เข้าไป ในการเจาะวงกลมแบบที่ 3 จะต้องมีการป้อนค่าของขนาดเส้นผ่านศก.ของหัวเจาะด้วย ดังรูปที่ 6.10 ซึ่งเมื่อป้อนแล้วก็จะเสร็จสำหรับการสร้าง G Code ของ Object C-3



รูปที่ 6.10 *แสดงหน้าจอรับค่าของขนาดเส้นผ่านศก.หัวเจาะในการเจาะวงกลม*

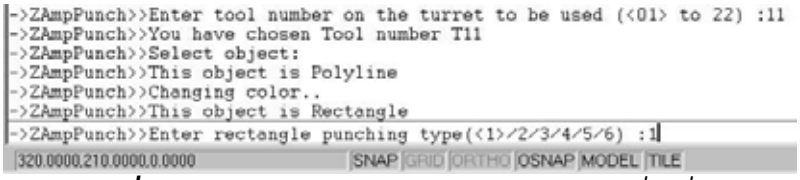
- Object ต่อไปซึ่งเป็น Object สุดท้ายสำหรับการเจาะด้วยหัวเจาะหมายเลข 08 คือ A-1 ซึ่งเป็นรูที่มีลักษณะโค้งซึ่งมีความกว้าง 10 มม. เมื่อเลือกส่วนโค้ง A-1 แล้ว (จะเลือกโค้งด้านในหรือโค้งด้านนอกก็ได้) ซอฟต์แวร์จะให้ป้อนว่าต้องการเจาะด้านในหรือด้านนอกของส่วนโค้งที่ได้เลือก ดังแสดงในรูปที่ 6.11 ซึ่งก็ขึ้นอยู่กับผู้ใช้ว่าได้เลือกส่วนโค้งด้านในหรือนอก จากนั้นต้องป้อนค่าของขนาดเส้นผ่านศก.ของหัวเจาะด้วย จากนั้นถือว่าเสร็จในส่วนการสร้าง G Code สำหรับ Object A-1 และสิ้นสุดการใช้หัวเจาะหมายเลข 08



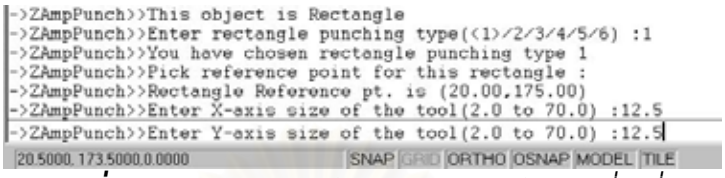
รูปที่ 6.11 *แสดงหน้าจอรับค่าของการเจาะด้านในหรือด้านนอกส่วนโค้ง*

การเจาะภายในด้วยหัวเจาะหมายเลข 11

- เมื่อเจาะ Object ที่ผ่านมาเสร็จแล้ว ผู้ใช้ป้อน “Ch” เพื่อเปลี่ยนหัวเจาะในการเจาะ Object ต่อไป หัวเจาะที่ใช้ต่อมาคือหมายเลข 11 ซึ่งจะเริ่มเจาะจากสี่เหลี่ยม D-1 ซึ่งเป็นการเจาะแบบที่ 1 (ใช้คำสั่ง “REC/” แนวนอน) เมื่อเริ่มเลือกสี่เหลี่ยม D-1 ซอฟต์แวร์จะถามว่าต้องการเจาะแบบใด ผู้ใช้ป้อน “1” ดังรูปที่ 6.12 โดยที่การเจาะแบบที่ 1 จะให้ผู้ใช้เลือกจุดอ้างอิงในการเริ่มเจาะ ซึ่งจุดที่เป็นจุดอ้างอิงจะต้องเป็นจุดมุมล่างด้านซ้ายหรือด้านขวาเท่านั้น จากนั้นต้องป้อนขนาดหัวเจาะในด้านแกน X และแกน Y ด้วยดังรูปที่ 6.13 จากนั้นถือว่าเสร็จสิ้นการเจาะสี่เหลี่ยม D-1



รูปที่ 6.12 *แสดงหน้าจอรับค่าแบบของการเจาะสี่เหลี่ยม*



รูปที่ 6.13 *แสดงหน้าจอรับค่าขนาดของหัวเจาะสี่เหลี่ยม*

- Object ต่อไปคือ D-2 ซึ่งเป็นการเจาะแบบที่ 2 (ใช้คำสั่ง “REC/” แนวตั้ง) ซึ่งเริ่มจากการเลือกสี่เหลี่ยมเช่นกันแล้วเลือกจุดอ้างอิง สำหรับแบบที่ 2 จุดอ้างอิงจะเป็นได้เฉพาะจุดด้านซ้ายบนหรือล่างของสี่เหลี่ยมเท่านั้น ซึ่งเมื่อเลือกเสร็จ การป้อนค่าอื่น ๆ จะเหมือนกับ Object D-1
- สำหรับ D-3 เป็นการเจาะแบบที่ 3 (ใช้คำสั่ง “REC/” ในแนวนอนและตั้ง) ซึ่งเป็นการเปิดรูสี่เหลี่ยมโดยที่ไม่มีการเอาเศษเหล็ก (ถ้ามี) ตรงกลางออก ซึ่งค่าที่ป้อนคือ 3 เมื่อซอฟต์แวร์ถามว่าจะให้เจาะแบบใด แล้วทำการเลือกจุดเริ่ม โดยที่จุดเริ่มจะต้องเป็นจุดมุมใดมุมหนึ่งของสี่เหลี่ยมเท่านั้น และต้องป้อนค่าขนาดของหัวเจาะเช่นเดียวกัน
- จากนั้นเจาะ D-4 ซึ่งเป็นการเจาะแบบที่ 4 (ใช้คำสั่ง “REC/” ในแนวนอนและตั้ง) ซึ่งเป็นการเปิดรูสี่เหลี่ยมโดยที่มีการเอาเศษเหล็ก (ถ้ามี) ตรงกลางออก ค่าที่ป้อนคือ 4 ลักษณะการป้อนค่าอื่น ๆ จะมีลักษณะเหมือนกับ D-3
- สุดท้ายสำหรับหัวเจาะหมายเลข 11 ได้แก่การเจาะ D-6 ซึ่งเป็นการเจาะรูสี่เหลี่ยมที่มีขนาดเดียวกับหัวเจาะ การเจาะก็จะเกิดขึ้นเพียงครั้งเดียวที่จุดศก.ของหัวเจาะ การป้อนคำสั่งเพียงป้อน 6 เมื่อถามว่าจะเจาะแบบใดเท่านั้น ตอนนี้ถือว่าสิ้นสุดการเจาะด้วยหัวเจาะหมายเลข 11 และสิ้นสุดการสร้างรหัส G Code สำหรับการเจาะภายในเรียบร้อยแล้ว ดังนั้นจึงป้อน “E” เมื่อซอฟต์แวร์จบการสร้าง G Code สำหรับ D-6

การเจาะภายนอก

- ป้อนคำสั่ง “outpunch” เพื่อเจาะ Object ภายนอก ที่ AutoCAD Command Prompt

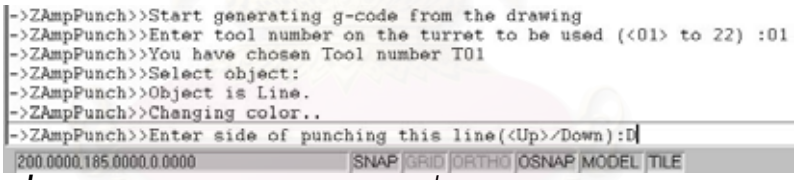
- คำสั่งนี้จะมีขั้นตอนเหมือนกันกับการเจาะภายในซึ่งจะให้ผู้ใช้สร้างแฟ้มข้อมูลที่เป็นตัวอักษรเก็บไว้ซึ่งจะแยกกับแฟ้มสำหรับเก็บรหัส G Code ของการเจาะภายใน โดยลักษณะจะเหมือนกันกับรูปที่ 6.5 และรูปที่ 6.6
- การเจาะภายนอกจะใช้เพียงหัวเจาะหมายเลข 01 ในการเจาะขอบเท่านั้น

การเจาะภายนอกด้วยหัวเจาะหมายเลข 01

หัวเจาะหมายเลข 01 เป็นหัวเจาะสี่เหลี่ยมที่มีขนาดด้านแกน X คือ 40 มม.และแกน Y คือ 5 มม. หัวเจาะหมายเลขนี้หมุนได้ ดังนั้นการติดตั้งหัวเจาะเริ่มต้นเป็นสิ่งสำคัญที่ต้องคำนึงถึง สำหรับชิ้นงานนี้หัวเจาะหมายเลข 01 ติดตั้งในลักษณะขนาดแนวแกน X เป็น 40 มม. และแกน Y คือ 5 มม. (ซึ่งเมื่อหัวเจาะหมุนไป 90 องศาจะทำให้ขนาดด้านแกน X เป็น 5 มม. และด้านแกน Y เป็น 40 มม. แทน)

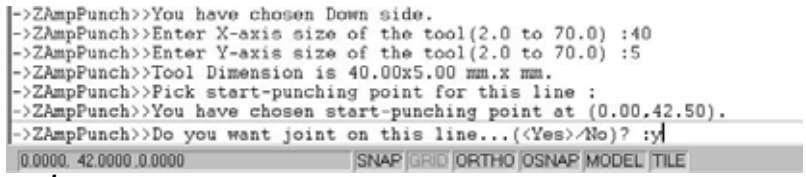
อย่างไรก็ตามการเจาะแบบใช้หัวเจาะหมุนได้ จะต้องเพิ่ม G Code ส่วนของ “C” โดยเพิ่มไว้หลัง Code “T” เช่น MOV/X10Y120T01C00 เป็นต้น โดยที่ตัวเลขตามหลัง “C” คือองศาที่จะให้หมุนไป อาทิหากเป็น C90 แสดงว่าให้หมุนหัวเจาะไป 90 องศา G Code ในส่วนของ “C” เป็นส่วนที่ผู้วางแผนต้องเขียนเพิ่มลงไปเองในแฟ้มข้อมูลที่ซอฟต์แวร์สร้างขึ้น

- เริ่มด้วยการเจาะ Object ที่เป็นเส้น L-1 ซึ่งจะเจาะโดยใช้คำสั่ง REC (เหมือนการเจาะสี่เหลี่ยมในแนวนอน) การเจาะเริ่มจากเลือกเส้น L-1 แล้วซอฟต์แวร์จะถามว่าจะเจาะเหนือเส้น (ป้อน “Up”) หรือใต้เส้น (ป้อน “Down”) ในที่นี้ป้อน “Down” ดังรูปที่ 6.14



รูปที่ 6.14 *แสดงหน้าจอรับค่าของด้านที่ต้องการเจาะของเส้นขอบแนวนอน*

ป้อน “Down” ลงไป แล้วซอฟต์แวร์จะถามขนาดของหัวเจาะด้านแกน X และแกน Y ซึ่งก็ป้อน 40 และ 5 ลงไปตามลำดับจากนั้นให้เลือกจุดเริ่มในการเจาะขอบ L-1 ซึ่งเมื่อเลือกแล้ว ซอฟต์แวร์ถามว่าจะให้มี Microjoint ที่ปลายทั้งสองของขอบหรือไม่ ดังรูปที่ 6.15 เมื่อตอบต้องการซอฟต์แวร์จะแสดงตำแหน่ง Microjoint ที่ต้องการให้เห็นในรูป Drawing ซึ่งถือว่าเป็นการเสร็จสิ้นสำหรับเจาะ L-1

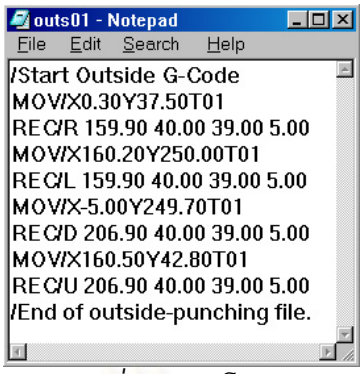


รูปที่ 6.15 *แสดงหน้าจอรับค่าของการเลือกให้ขอบมี Microjoint หรือไม่*

- ต่อไปจะเจาะ L-2 ซึ่งเป็นเส้นแนวนอนเหมือนกับ L-1 การเจาะจึงทำในลักษณะเดียวกันแต่ต่างกันตรงที่เป็นการเจาะเหนือเส้นที่เลือก
- การเจาะ L-3 จะเจาะโดยใช้การหมุนหัวเจาะไป 90 องศา ซึ่งเมื่อเลือกเส้น L-3 แล้วซอฟต์แวร์จะถามว่าจะให้เจาะด้านซ้าย (ป้อน “Left”) หรือด้านขวา (ป้อน “Right”) สำหรับ L-3 แล้วเป็นการเจาะด้านซ้ายของขอบ เมื่อป้อนด้านแล้วซอฟต์แวร์จะให้ป้อนค่าซึ่งมีขั้นตอนเหมือน L-1 โดยที่ค่าของขนาดหัวเจาะที่ต้องระบุจะเปลี่ยนไป เนื่องจากได้มีการหมุนหัวเจาะแล้ว ขนาดหัวเจาะที่ป้อนในแนวแกน X จะกลายเป็น 5 และแกน Y เป็น 40 สำหรับเส้นแนวตั้ง L-3 และ L-4 จะต้องมี Microjoint อยู่เช่นเดียวกัน
- การเจาะ L-4 จะทำในลักษณะเดียวกับ L-3 แต่แตกต่างกันที่ด้านจะเป็นการเจาะในด้านขวาของขอบ จากนั้นถือว่าใช้หัวเจาะหมายเลข 01 เสร็จแล้ว

หลังจากที่ได้รหัสโปรแกรม G Code สำหรับการเจาะภายในและภายนอกสำหรับชิ้นงานทดสอบที่ 1 แล้ว (แสดงได้ในรูปที่ 6.16 และ รูปที่ 6.17) ผู้วางแผนจะต้องนำเอารหัสโปรแกรมทั้งสองส่วนมารวมเข้าด้วยกันเนื่องจากมันถูกเก็บอยู่คนละแฟ้มข้อมูล แล้วบันทึกเก็บเอาไว้เป็นรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบ ซึ่งพร้อมที่จะนำไปทดสอบกับเครื่องจักร CNC

รูปที่ 6.16 *G Code การเจาะภายในที่เปิดด้วยโปรแกรม Notepad ของชิ้นงานทดสอบที่ 1*



รูปที่ 6.17 *G Code การเจาะภายนอกที่เปิดด้วยโปรแกรม Notepad ของชิ้นงานทดสอบที่ 1*

6.2.2. ชิ้นงานทดสอบที่ 2

สำหรับชิ้นงานทดสอบที่ 2 มีการวางแผนด้านการใช้หัวเจาะแสดงไว้ในตารางที่ 6.6

ตารางที่ 6.6 *การวางแผนด้านการใช้หัวเจาะสำหรับชิ้นงานทดสอบที่ 2*

ลำดับการใช้หัวเจาะ	ตำแหน่งหัวเจาะบนแท่นหมุน	Object ที่จะเจาะ (ตามลำดับ)
1	11	D-1, D-2 และ D-4
2	15	C-1 และ A-1
3	01 (หมุนได้)	L-1, L-2, L-3 และ L-4

จากนั้นเริ่มใช้ซอฟต์แวร์เพื่อช่วยในการสร้างรหัสโปรแกรม G Code ซึ่งการเจาะชิ้นงานทดสอบจะเริ่มจากการเจาะภายใน แล้วตามด้วยการเจาะภายนอก ขั้นตอนมีดังนี้

การเจาะภายใน

- ป้อนคำสั่ง “inpunch” เพื่อเจาะ Object ภายใน ที่ AutoCAD Command Prompt เพื่อเริ่มต้นเจาะด้านในเหมือนกับการเจาะชิ้นงานทดสอบที่ 1
- ตั้งชื่อแฟ้มที่ต้องการจะเก็บรหัสโปรแกรม G Code สำหรับการเจาะภายในของชิ้นงานทดสอบที่ 2
- เช่นเดียวกันกับการเจาะชิ้นงานทดสอบที่ 1 และการเจาะแผ่นโลหะทั่วไป ลักษณะการเจาะจะเป็นการเจาะโดยใช้หัวเจาะตัวหนึ่งสำหรับ Object ต่าง ๆ ให้เสร็จก่อนแล้วจึงเปลี่ยนหัวเจาะ ชิ้นงานทดสอบนี้จะเริ่มจากการใช้หัวเจาะหมายเลข 11 และ 15 เพื่อเจาะภายใน

การเจาะภายในด้วยหัวเจาะหมายเลข 11

- การเจาะด้วยหัวเจาะสี่เหลี่ยมจัตุรัสขนาด 12.5 x 12.5 มม. จะใช้เจาะ Object ที่เป็นสี่เหลี่ยมทั้งหมดโดยเริ่มจาก Object D-1 ทีละรูป ซึ่งเป็นการเจาะสี่เหลี่ยมใน

แบบที่ 1 โดยใช้คำสั่ง “REC/” เจาะไปตามแนวนอน ซึ่งเริ่มจากการเริ่มป้อนหมายเลขของหัวเจาะที่ต้องการใช้ จากนั้นเลือกไปที่ Object D-1 ซึ่งจะเห็นว่าสีของมันจะเปลี่ยนไปเป็นสีฟ้า จากนั้นซอฟต์แวร์จะให้ป้อนแบบในการเจาะสี่เหลี่ยม D-1 ซึ่งก็ป้อนค่า “1” ลงไป ทั้งนี้ซอฟต์แวร์จะถามจุดอ้างอิง และขนาดของหัวเจาะ เช่นเดียวกันกับการเจาะ Object D-1 สำหรับชิ้นงานทดสอบที่ 1

- Object ต่อไปที่จะเจาะคือ D-2 ซึ่งมีขั้นตอนในการป้อนค่าเช่นเดียวกันกับ Object D-2 สำหรับชิ้นงานทดสอบที่ 1
- Object สุดท้ายที่จะใช้หัวเจาะหมายเลข 11 เจาะคือ D-4 ซึ่งการป้อนค่าต่าง ๆ มีลักษณะเหมือนกันกับการเจาะ Object D-4 สำหรับชิ้นงานทดสอบที่ 1 เช่นเดียวกัน

การเจาะภายในด้วยหัวเจาะหมายเลข 15

- จะเริ่มจากการเจาะ Object วงกลม C-1 ซึ่งเป็นการเจาะแบบครั้งเดียวในตำแหน่ง ศก.ของหัวเจาะ ซึ่งหลังจากเปลี่ยนหัวเจาะที่จะใช้โดยการป้อนหมายเลข 15 เมื่อเปลี่ยนหัวเจาะแล้ว ขั้นตอนในการเจาะ C-1 ก็มีขั้นตอนเดียวกันกับการเจาะ Object C-1 สำหรับชิ้นงานทดสอบที่ 1
- Object ที่ต้องการเจาะต่อไปคือส่วนโค้ง A-1 ซึ่งมีด้วยกัน 4 รูป เมื่อได้เลือกส่วนโค้งดังกล่าวแล้ว ซอฟต์แวร์จะถามว่าต้องการเจาะด้านในหรือด้านนอกส่วนโค้ง ในที่นี้ต้องการเจาะด้านใน จึงป้อน “In” และซอฟต์แวร์จะให้ผู้ใช้ใส่ค่าขนาดของเส้นผ่านศก.ของหัวเจาะที่ใช้ด้วย การเจาะส่วนโค้ง A-1 จะมีขั้นตอนเหมือนกันกับส่วนโค้ง A-1 ในชิ้นงานทดสอบที่ 1 เช่นเดียวกัน

การเจาะภายนอก

- เนื่องจากลักษณะของชิ้นงานทดสอบที่ 2 สำหรับการเจาะด้านนอกเหมือนกันกับชิ้นงานทดสอบที่ 1 ดังนั้นขั้นตอนในการใช้ซอฟต์แวร์จะเหมือนกันกับชิ้นงานทดสอบที่ 1 ซึ่งจะไม่กล่าวซ้ำในส่วนนี้

หลังจากที่ได้รหัสโปรแกรม G Code สำหรับการเจาะภายในและภายนอกสำหรับชิ้นงานทดสอบที่ 2 แล้ว (แสดงได้ในรูปที่ 6.18 และ รูปที่ 6.19) ผู้วางแผนจะต้องนำเอารหัสโปรแกรมทั้งสองส่วนมารวมเข้าด้วยกัน แล้วบันทึกเก็บเอาไว้เป็นรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบสำหรับชิ้นงานทดสอบที่ 2 ซึ่งพร้อมที่จะนำไปทดสอบกับเครื่องจักร CNC

รูปที่ 6.18 *G Code การเจาะภายในที่เปิดด้วยโปรแกรม Notepad ของชิ้นงานทดสอบที่ 2*

รูปที่ 6.19 *G Code การเจาะภายนอกที่เปิดด้วยโปรแกรม Notepad ของชิ้นงานทดสอบที่ 2*

6.2.3. ชิ้นงานทดสอบที่ 3

สำหรับชิ้นงานทดสอบที่ 3 มีการวางแผนด้านการใช้หัวเจาะแสดงไว้ในตารางที่ 6.7

ตารางที่ 6.7 *การวางแผนด้านการใช้หัวเจาะสำหรับชิ้นงานทดสอบที่ 3*

ลำดับการใช้หัวเจาะ	ตำแหน่งหัวเจาะบนแท่นหมุน	Object ที่จะเจาะ (ตามลำดับ)
1	01 (หมุนได้)	L-1, L-2, L-3 และ L-4
2	11	N-1, N-2, N-3 และ N-4

การเจาะชิ้นงานทดสอบที่ 3 เป็นการเจาะเฉพาะภายนอกเพื่อให้ได้แผ่นโลหะตามรูปร่างที่ต้องการ ซึ่งการเจาะจะใช้หัวเจาะหมายเลข 01 ในการเจาะเส้นขอบ และหมายเลข 11 สำหรับการเจาะลักษณะเส้นที่เป็นมุม ตามลำดับ

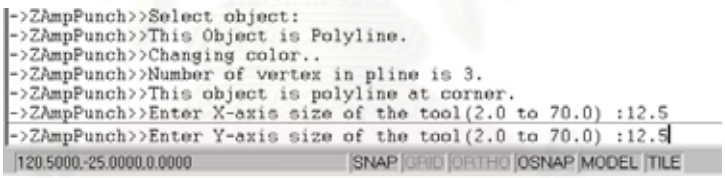
- ป้อนคำสั่ง “outpunch” เพื่อการเจาะภายนอก ที่ AutoCAD Command Prompt เพื่อเริ่มต้นเจาะด้านนอกเหมือนกับการเจาะชิ้นงานทดสอบที่ 1
- ตั้งชื่อแฟ้มที่ต้องการจะเก็บรหัสโปรแกรม G Code สำหรับการเจาะภายในของชิ้นงานทดสอบที่ 3

การเจาะภายนอกด้วยหัวเจาะหมายเลข 01

- หัวเจาะหมายเลข 01 จะใช้เจาะขอบทั้ง 4 ด้านที่เป็นเส้นอันได้แก่ L-1 L-2 L-3 และ L-4 ตามลำดับ ซึ่งเป็นเส้นในแนวนอนและแนวตั้ง โดยที่ขั้นตอนในการใช้ซอฟต์แวร์ในส่วนนี้จะเหมือนกันกับการเจาะภายนอกของชิ้นงานทดสอบที่ 1 ยกเว้นเพียงสำหรับเส้น L-1 นั้นไม่ต้องการให้มี Microjoint อยู่ที่ปลายทั้งสองของเส้น ดังนั้นจึงไม่กล่าวซ้ำในที่นี้

การเจาะภายนอกด้วยหัวเจาะหมายเลข 11

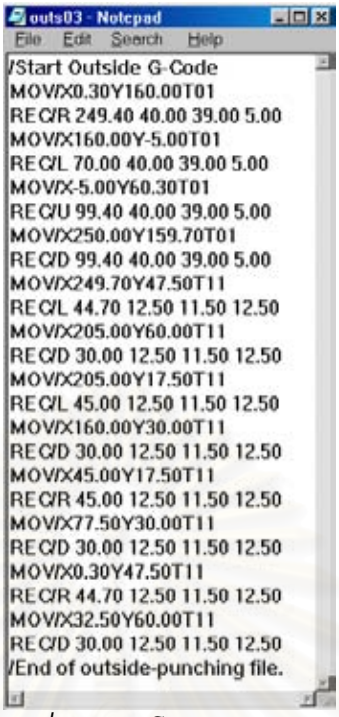
- หัวเจาะหมายเลข 11 จะเริ่มจากการเจาะมุมรหัส N-1 ก่อน โดยที่เมื่อเลือก Object นี้แล้ว ซอฟต์แวร์จะบอกว่าได้เลือก Object ที่เป็นเส้นสำหรับมุม ซึ่งต่อมาจะให้ผู้ใช้ป้อนค่าของขนาดของหัวเจาะที่ใช้เจาะ ดังรูปที่ 6.20 จากนั้นถือว่าเสร็จสิ้นการเจาะมุม N-1



รูปที่ 6.20 *แสดงหน้าจอรับค่าขนาดของหัวเจาะสำหรับการเจาะ Object ที่เป็นเส้นสำหรับมุม*

- สำหรับมุม N-2 N-3 และ N-4 มีขั้นตอนในการเจาะเช่นเดียวกันกับมุม N-1 ซึ่งเมื่อเจาะเสร็จหมดแล้วก็ถือว่าสิ้นสุดการเจาะชิ้นงานทดสอบที่ 3

เมื่อได้รหัสโปรแกรม G Code ของการเจาะชิ้นงานทดสอบที่ 3 เรียบร้อยแล้ว (ดังรูปที่ 6.21) ผู้วางแผนบันทึกเก็บเอาไว้เป็นรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบสำหรับชิ้นงานทดสอบที่ 3 ซึ่งพร้อมที่จะนำไปทดสอบกับเครื่องจักร CNC



รูปที่ 6.21 *G Code การเจาะที่เปิดด้วยโปรแกรม Notepad ของชิ้นงานทดสอบที่ 3*

6.3. การนำรหัสโปรแกรม G Code ป้อนเข้าเครื่องจักร CNC

หลังจากที่ได้รหัสคำสั่งโปรแกรม G Code สมบูรณ์พร้อมจะนำไปทดสอบสำหรับชิ้นงานทดสอบทั้งหมดแล้ว ผู้วางแผนกระบวนการผลิตทำการติดตั้งหัวเจาะที่ได้วางแผนไว้ ดังรูปที่ 6.22 แล้วนำแผ่นโลหะแผ่นใหญ่ที่ต้องการเจาะมาติดตั้งที่โต๊ะ ดังรูปที่ 6.23

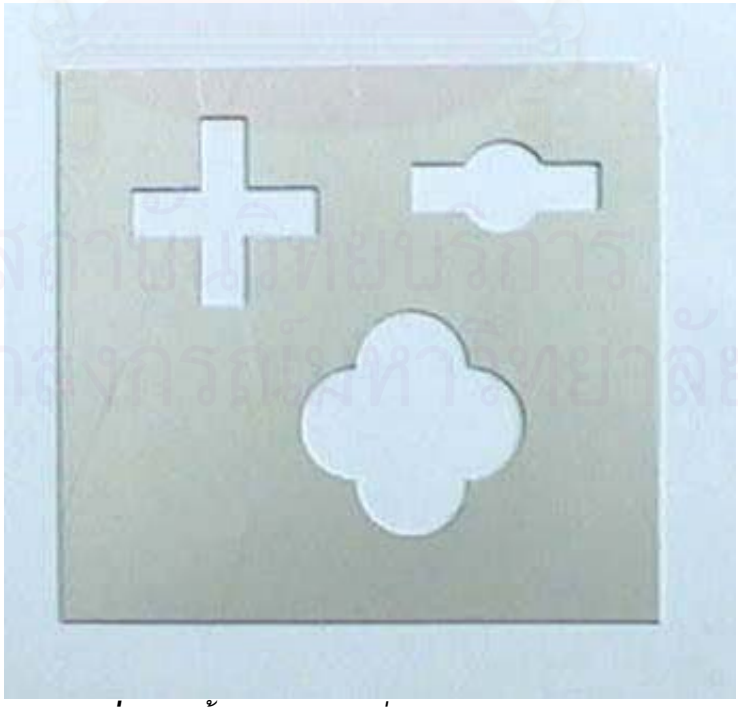
รูปที่ 6.22 *การติดตั้งหัวเจาะลงบนแท่นหมุน*

รูปที่ 6.23 *การติดตั้งแผ่นโลหะเข้ากับที่จับยึดของโต๊ะ*

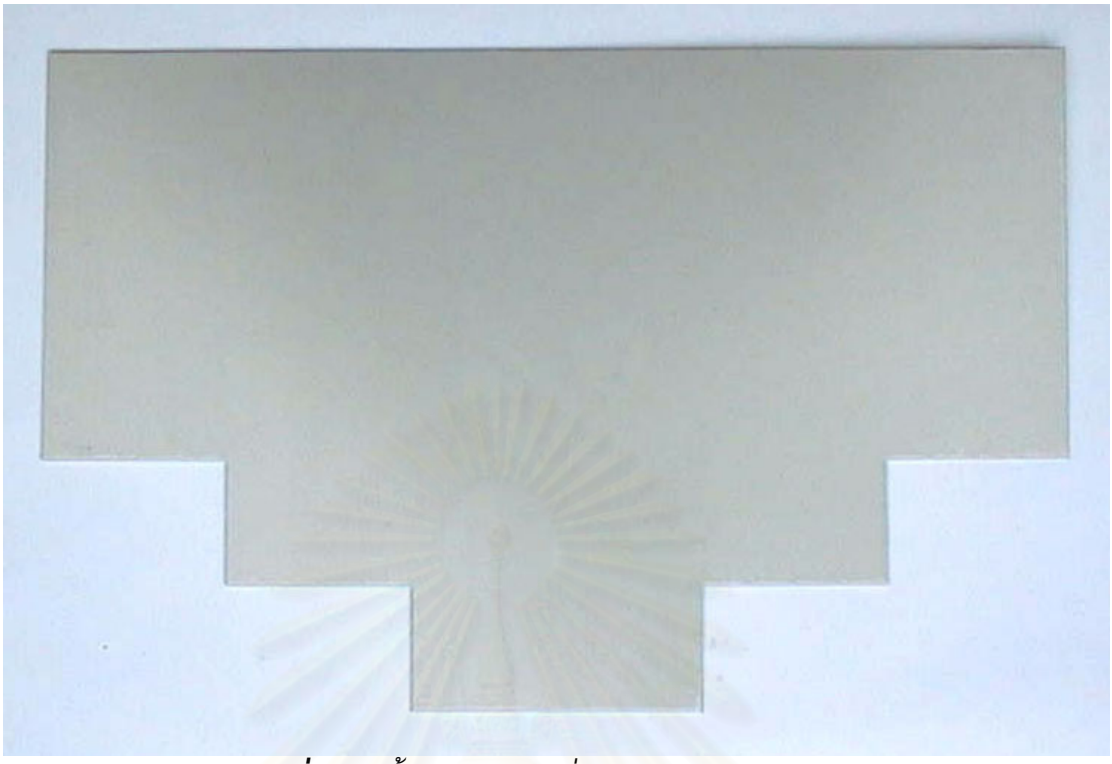
เมื่อป้อน G Code เข้าไป เครื่องจักร CNC จะมีจอสำหรับแสดงการจำลองการเจาะ ดังนั้นจึงนำ G Code ที่ได้ไปจำลองการเจาะก่อนที่จอควบคุมของเครื่องจักร CNC จากนั้นจึงเริ่มการทดลองเจาะจริง (รูปที่ 6.24) ผลของแผ่นโลหะทดสอบที่ 1 2 และ 3 ที่ถูกเจาะแล้วได้แสดงในรูปที่ 6.25 6.26 และ 6.27 ซึ่งจะเห็นว่าชิ้นงานที่ได้มีความถูกต้องตรงกับ Drawing ใน AutoCAD ที่ได้สร้างไว้ทั้ง 3 ตัวอย่างทดสอบ

รูปที่ 6.24 *ชิ้นงานทดสอบขณะถูกเจาะด้วยเครื่องจักร CNC*

รูปที่ 6.25 *ชิ้นงานทดสอบที่ 1 หลังจากเจาะเสร็จแล้ว*



รูปที่ 6.26 *ชิ้นงานทดสอบที่ 2 หลังจากเจาะเสร็จแล้ว*



รูปที่ 6.27 ชิ้นงานทดสอบที่ 3 หลังจากเจาะเสร็จแล้ว

6.4. สรุป

ในการทดสอบความถูกต้องของซอฟต์แวร์ในการช่วยสร้างรหัสโปรแกรม G Code จะทดสอบกับชิ้นงานที่สร้างขึ้นตามข้อกำหนดเพื่อให้เหมาะสมกับการใช้ซอฟต์แวร์ Object ที่สร้างขึ้นทั้งหมดภายในชิ้นงานทดสอบจะใช้ทดสอบการสร้างรหัสคำสั่ง G Code ทั้งหมดที่ซอฟต์แวร์จะสามารถสร้างได้ ซึ่งผลที่ได้จากซอฟต์แวร์คือรหัสคำสั่ง G Code ของการเจาะภายในและการเจาะภายนอก ซึ่งเมื่อนำมารวมกันแล้วป้อนเข้าเครื่องจักร CNC เพื่อทำการทดสอบการเจาะ ผลของแผ่นโลหะที่ได้จะมีความถูกต้องตรงกับ Drawing ที่สร้างไว้ใน AutoCAD

บทที่ 7

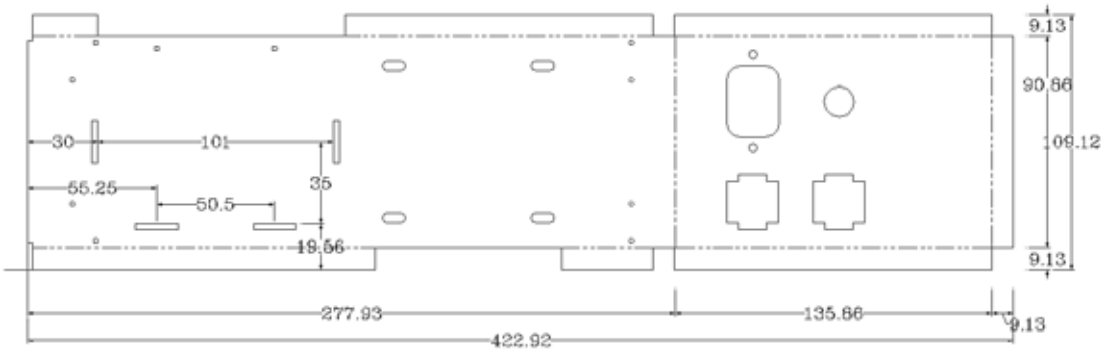
การเปรียบเทียบการวางแผนกระบวนการผลิตแบบเดิมและการนำซอฟต์แวร์เข้ามาช่วย

เนื้อหาในบทนี้กล่าวถึงการเปรียบเทียบการวางแผนกระบวนการผลิตแบบเดิมและการนำซอฟต์แวร์ที่สร้างขึ้นเข้ามาช่วยในส่วนการสร้างรหัสโปรแกรม G Code โดยใช้เวลาเป็นตัววัด รวมทั้งความพึงพอใจของผู้วางแผนกระบวนการผลิตที่มีต่อการใช้งานซอฟต์แวร์ดังกล่าว

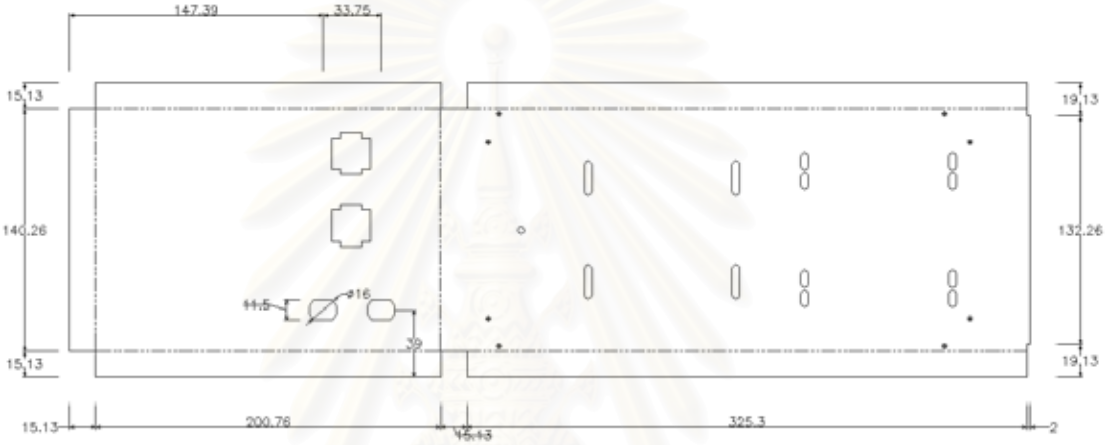
7.1. การเปรียบเทียบเวลาในการวางแผนกระบวนการผลิต

การนำซอฟต์แวร์เข้ามาช่วยในการวางแผนกระบวนการผลิตในส่วนของการสร้างรหัสโปรแกรม G Code ช่วยลดปัญหาต่าง ๆ ที่เกิดขึ้นในการวางแผนแบบเดิม เช่น การลดเวลาในการสร้างรหัสโปรแกรม G Code การลดความเมื่อยล้า และอื่น ๆ ในส่วนนี้เป็นการเปรียบเทียบเวลาที่ผู้วางแผนกระบวนการผลิตใช้ในการวางแผนเพื่อเจาะผลิตภัณฑ์ต้นแบบ โดยเปรียบเทียบเวลาที่ใช้ในขั้นตอนการวางแผนแบบเดิมกับเวลาที่ใช้วางแผนเมื่อใช้ซอฟต์แวร์เข้ามาช่วยเพื่อแสดงให้เห็นว่าซอฟต์แวร์ที่สร้างขึ้นเข้ามาช่วยลดภาระการทำงานของผู้วางแผนได้มากน้อยเพียงใด

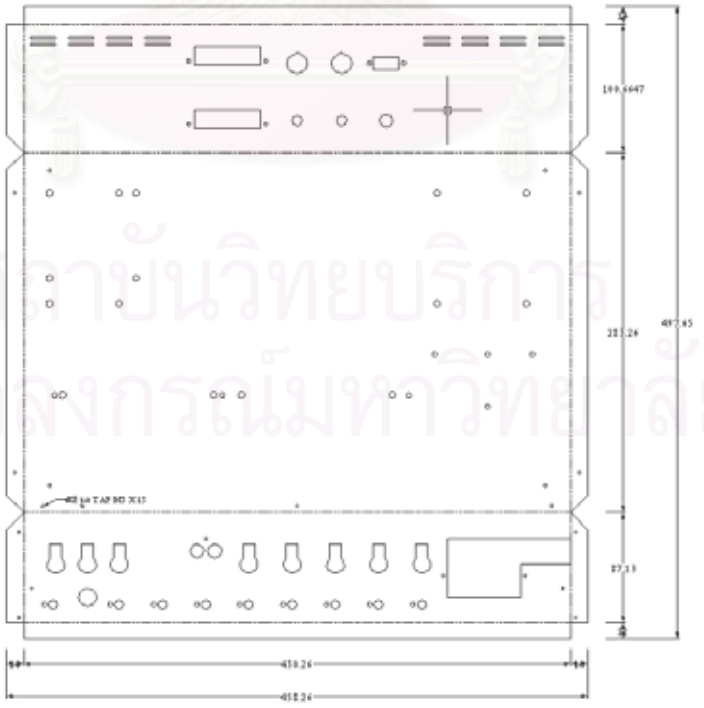
ในการเปรียบเทียบเวลานั้น จะทำโดยให้ผู้วางแผนกระบวนการผลิตคนหนึ่งซึ่งมีความชำนาญในการสร้างรหัสโปรแกรม G Code และการใช้ซอฟต์แวร์ ทำการวางแผนเพื่อเจาะชิ้นงานที่สร้างไว้แล้วในรูปแบบ Drawing ของ AutoCAD ที่ต่างกัน 3 ภาพ (รูปที่ 7.1 7.2 และ 7.3) การจับเวลาเริ่มจากการที่ได้แบบดังกล่าวและวางแผนการใช้หัวเจาะและติดตั้งหัวเจาะเสร็จแล้ว จนกระทั่งทำการเจาะสำเร็จเป็นชิ้นงานต้นแบบจริง ๆ ซึ่งขั้นตอนภายในช่วงเวลาดังกล่าวของการวางแผนแบบเดิมและแบบที่นำซอฟต์แวร์เข้ามาช่วยจะต่างกันเล็กน้อย ในการจับเวลาจะแบ่งขั้นตอนของทั้งสองแบบออกเป็นขั้นตอนย่อย ๆ แล้วนำเวลารวมมาเปรียบเทียบกัน



รูปที่ 7.1 ชิ้นงานทดสอบที่ 1



รูปที่ 7.2 ชิ้นงานทดสอบที่ 2



รูปที่ 7.3 ชิ้นงานทดสอบที่ 3

สำหรับเวลาที่ผู้วางแผนใช้ในแต่ละขั้นตอนของการวางแผนระหว่างการจับเวลาของการวางแผนแบบเดิมและแบบที่นำซอฟต์แวร์มาช่วยแสดงได้ในตารางที่ 7.1 และ 7.2 และรูปที่ 7.4

ตารางที่ 7.1 *เวลาที่ใช้ในขั้นตอนการวางแผนกระบวนการผลิตแบบเดิม (หน่วย นาที:วินาที)*

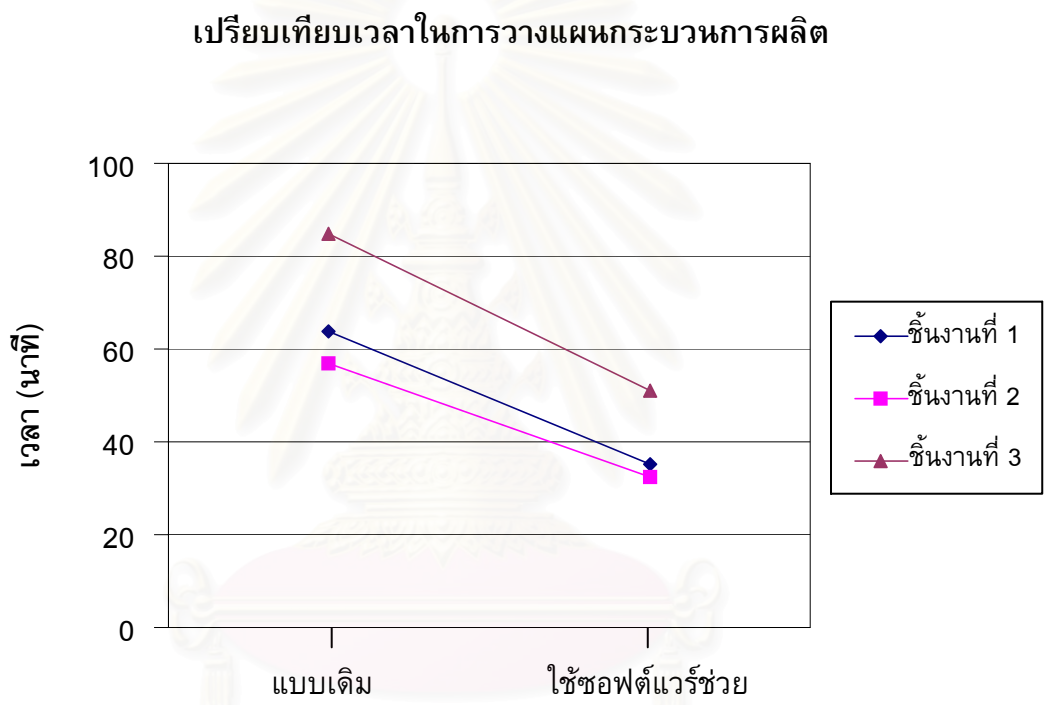
การวางแผนกระบวนการผลิตแบบเดิม			
ขั้นตอนระหว่างการจับเวลา	**ชิ้นงานที่ 1**	**ชิ้นงานที่ 2**	**ชิ้นงานที่ 3**
1. หลังจากได้ Drawing แล้ว พิมพ์ออกมาในให้อยู่ในกระดาษเขียนแบบ	00 : 14	00 : 14	00 : 14
2. สร้างรหัสโปรแกรม G Code โดยอ่านจากแบบที่พิมพ์ออกมา สำหรับ			
- การเจาะภายใน	20 : 08	18 : 25	45 : 32
- การเจาะภายนอก	26 : 45	25 : 10	17 : 05
3. ตรวจสอบความถูกต้องและแก้ไขรหัสคำสั่งหากเกิดความผิดพลาดโดยตรวจสอบกับการจำลองการเจาะที่ส่วนแสดงผลของเครื่องจักร CNC	14 : 32	10 : 45	19 : 06
4. ป้อนรหัส G Code ที่ถูกต้องและทดสอบเจาะจนได้ชิ้นงานต้นแบบสำเร็จ	02 : 15	02 : 10	03 : 47
เวลารวม	**63 : 54**	**56 : 44**	**84 : 44**

ตารางที่ 7.2 *เวลาที่ใช้ในขั้นตอนการวางแผนกระบวนการผลิตโดยนำซอฟต์แวร์เข้ามาช่วย (หน่วย นาที:วินาที)*

การวางแผนกระบวนการผลิตโดยนำซอฟต์แวร์เข้ามาช่วย			
ขั้นตอนระหว่างการจับเวลา	**ชิ้นงานที่ 1**	**ชิ้นงานที่ 2**	**ชิ้นงานที่ 3**
1. หลังจากได้ Drawing แล้ว สร้างและกำหนด Drawing ดังกล่าวให้มีรูปแบบเหมาะสมที่จะใช้กับซอฟต์แวร์	07 : 11	06 : 40	12 : 32
2. สร้างรหัสโปรแกรม G Code โดยใช้ซอฟต์แวร์สำหรับ			
- การเจาะภายใน	07 : 38	07 : 13	19 : 08
- การเจาะภายนอก	12 : 20	10 : 52	07 : 26
3. ตรวจสอบความถูกต้องและแก้ไขรหัสคำสั่งหากเกิดความผิดพลาดโดยตรวจสอบกับการจำลองการเจาะที่ส่วนแสดงผลของเครื่องจักร CNC	05 : 51	05 : 25	07 : 44

ตารางที่ 7.2 (ต่อ) *เวลาที่ใช้ในขั้นตอนการวางแผนกระบวนการผลิตโดยนำซอฟต์แวร์เข้ามาช่วย (หน่วย นาที:วินาที)*

การวางแผนกระบวนการผลิตโดยนำซอฟต์แวร์เข้ามาช่วย			
ขั้นตอนระหว่างการจับเวลา	**ชิ้นงานที่ 1**	**ชิ้นงานที่ 2**	**ชิ้นงานที่ 3**
4. ป้อนรหัส G Code ที่ถูกต้องและทดสอบเจาะจนได้ชิ้นงานต้นแบบสำเร็จ	02 : 19	02 : 16	04 : 03
เวลารวม	**35 : 19**	**32 : 26**	**50 : 53**



รูปที่ 7.4 *กราฟแสดงการเปรียบเทียบเวลาในการวางแผนกระบวนการผลิตแบบเดิมและแบบที่นำซอฟต์แวร์เข้ามาช่วย*

จากการเปรียบเทียบจะเห็นได้ว่า การวางแผนกระบวนการผลิตโดยใช้ซอฟต์แวร์เข้ามาช่วยนั้นใช้เวลารวมในการวางแผนน้อยกว่าการวางแผนกระบวนการผลิตแบบเดิม ซึ่งจากการวิเคราะห์การทดสอบที่เกิดขึ้น ในการวางแผนแบบเดิมใช้เวลามากกว่ามีสาเหตุเนื่องมาจาก

- การสร้าง G Code สำหรับเจาะภายในและภายนอกของแบบเดิมจะต้องใช้การวัดด้วยเครื่องมือวัดโดยใช้มือ ซึ่งเมื่อรายละเอียดของรูมีปริมาณมากทำให้ใช้เวลาค่อนข้างนานในการสร้างรหัสโปรแกรม
- ในขั้นตอนการตรวจสอบความถูกต้องและแก้ไขรหัสโปรแกรม G Code นั้นจะเห็นได้ชัดว่าเวลาที่ใช้ในแบบเดิมมากกว่าการใช้ซอฟต์แวร์ช่วยอยู่พอสมควร ทั้งนี้

เนื่องมาจากการสร้างรหัสโปรแกรม G Code แบบเดิมนั้นมีข้อผิดพลาดที่เกิดจากผู้วางแผนอยู่พอสมควรโดยเฉพาะข้อมูลด้านพิกัดที่จะต้องป้อน โดยที่ส่วนใหญ่ข้อผิดพลาดจะเกิดจากการสร้าง G Code ในส่วนของการกำหนดจุดอ้างอิงในคำสั่ง “MOV/” สำหรับคำสั่งที่เป็นรูปแบบในการเจาะต่าง ๆ ทำให้ต้องเสียเวลาในการแก้ไข ซึ่งต่างกับการใช้ซอฟต์แวร์ช่วย เพราะเป็นการสร้างรหัสโปรแกรมโดยใช้ฐานข้อมูลของ Drawing มาสร้าง ซึ่งจะมีความแน่นอนด้านพิกัดและระยะของรูป แต่อาจมีความผิดพลาดเกิดขึ้นบ้างเนื่องจากผู้วางแผนป้อนค่าในซอฟต์แวร์ไม่ถูกต้อง

- ในขั้นตอนของการเจาะชิ้นงานของเครื่องจักร CNC ใช้เวลาต่างกันเล็กน้อย เนื่องจากรหัสโปรแกรม G Code ที่สร้างขึ้นในแบบเดิมนั้น ผู้วางแผนอาจจะตัดสินใจใช้คำสั่งที่อยู่นอกเหนือขอบเขตของซอฟต์แวร์ ซึ่งรหัสโปรแกรมที่ได้จะมีรูปแบบต่างกันเล็กน้อย แต่ผลของชิ้นงานที่ได้มีความถูกต้องเหมือนกัน

7.2. ความพึงพอใจของผู้วางแผนกระบวนการผลิตที่มีต่อซอฟต์แวร์

ในการศึกษานี้ได้มีการสำรวจความคิดเห็นเกี่ยวกับความพึงพอใจที่เกิดขึ้นต่อการใช้ซอฟต์แวร์ช่วยในการสร้างรหัสโปรแกรม G Code โดยให้ผู้วางแผนกระบวนการผลิตทั้งหมด 3 คนได้ตอบแบบสอบถามซึ่งถูกรวบรวมอยู่ในภาคผนวก ฉ.

ผลที่ได้รับ จะเห็นว่า ในส่วนของความง่ายในการกระบวนการสร้างรหัส G Code โดยใช้ซอฟต์แวร์และการลดภาระที่เกิดขึ้นจากการวางแผนกระบวนการผลิตเดิมอยู่ในระดับที่ดี ในขณะที่ความเหมาะสมกับผลิตภัณฑ์ที่ลูกค้าสั่งและส่วนของความสามารถของซอฟต์แวร์อยู่ในระดับปานกลาง

7.3. สรุป

การนำซอฟต์แวร์เข้ามาช่วยสร้างรหัสโปรแกรม G Code จะช่วยให้การวางแผนกระบวนการผลิตใช้เวลาในการวางแผนลดลงและทำให้ลดความผิดพลาดที่เกิดขึ้นในการวางแผนกระบวนการผลิตได้มาก ในขณะที่ทำให้ผู้วางแผนมีความพึงพอใจที่ได้ใช้เนื่องจากลดภาระการทำงานที่ซ้ำซากและสร้างความถูกต้องในการทำงานได้อย่างน่าพอใจ

บทที่ 8

บทสรุปและข้อเสนอแนะ

8.1. สรุปผลงานวิจัย

กิจกรรมการวางแผนกระบวนการผลิตเป็นงานที่มีความสำคัญและมีผลกระทบโดยตรงต่อต้นทุนการผลิต ประสิทธิภาพการผลิต และคุณภาพของผลิตภัณฑ์ การวางแผนกระบวนการผลิตจะเกี่ยวข้องกับกิจกรรมต่าง ๆ มากมาย ทำให้การทำงานโดยอาศัยมนุษย์เกิดความไม่แน่นอนขึ้นซึ่งส่งผลทำให้เกิดต้นทุนที่สูงเกินความจำเป็น ใช้เวลาในการวางแผนนาน แผนกระบวนการผลิตเกิดความผิดพลาด และเกิดความเมื่อยล้ากับผู้วางแผนที่จะต้องทำงานซ้ำซาก ซึ่งมีรายละเอียดสูง การพัฒนาระบบคอมพิวเตอร์เพื่อเข้ามาช่วยในการวางแผนกระบวนการผลิตจึงมีความจำเป็น ระบบคอมพิวเตอร์ดังกล่าวจะถูกนำมาเชื่อมต่อระหว่างระบบคอมพิวเตอร์ช่วยในการออกแบบและระบบคอมพิวเตอร์ช่วยในการผลิตเพื่อให้การผลิตผลิตภัณฑ์เกิดความถูกต้องแม่นยำตรงตามที่ได้ออกแบบไว้

ในกระบวนการเจาะแผ่นโลหะโดยใช้เครื่อง CNC กิจกรรมวางแผนกระบวนการผลิตในส่วนของการสร้างรหัสโปรแกรม G Code ที่ผู้วางแผนได้สร้างขึ้นเป็นงานที่มีรายละเอียดสูง การวางแผนด้วยมนุษย์แบบเดิมจะทำให้เกิดความผิดพลาดได้ง่าย ซอฟต์แวร์ ZAmpPunch.arx จึงถูกพัฒนาขึ้นเพื่อช่วยให้ผู้วางแผนสามารถสร้างรหัสโปรแกรม G Code ให้เป็นไปตามรูปแบบคำสั่งที่เครื่องจักร CNC เข้าใจได้และมีความถูกต้องแม่นยำในการผลิต โดยซอฟต์แวร์ที่เข้ามาช่วยนี้เสมือนเป็นตัวเชื่อมระหว่างระบบคอมพิวเตอร์ช่วยในการออกแบบ ได้แก่ โปรแกรม AutoCAD R14 และระบบคอมพิวเตอร์ช่วยในการผลิต อันได้แก่ ระบบควบคุมแบบ CNC

ซอฟต์แวร์ ZAmpPunch.arx มีหน้าที่หลักในการสร้างรหัสคำสั่ง G Code สำหรับ Drawing ของแผ่นโลหะในรูปแบบ 2 มิติที่อยู่ในโปรแกรม AutoCAD R14 Drawing ดังกล่าวจะต้องมีรูปแบบที่เป็นไปตามข้อกำหนดที่กำหนดไว้เพื่อให้ซอฟต์แวร์สร้างรหัสโปรแกรม G Code ที่มีความถูกต้องที่สุด การทำงานของซอฟต์แวร์มีหลักการคือ ให้ผู้ใช้เลือก Object ที่เป็นรูปร่างใน Drawing ที่ต้องการเจาะแล้วป้อนค่าที่จำเป็นของคำสั่ง G Code ที่ต้องการ แล้วซอฟต์แวร์จะคำนวณข้อมูลต่าง ๆ แล้วเขียนรหัส G Code ลงใน Dat File ที่ได้สร้างขึ้น

ในการนำซอฟต์แวร์เพื่อช่วยในกระบวนการสร้างรหัสโปรแกรม G Code ไปให้ผู้วางแผนใช้ทดสอบผลิตชิ้นงานทดสอบโดยเครื่องจักร CNC ผลของแผ่นโลหะที่ได้มีความถูกต้องตรงตามแบบที่ได้ออกแบบไว้ใน AutoCAD และเมื่อเปรียบเทียบเวลาของการวางแผนแบบเดิมและการนำซอฟต์แวร์เข้ามาช่วย โดยให้ผู้วางแผนที่มีความสามารถในการสร้าง G Code และมีความชำนาญในการใช้งานซอฟต์แวร์ทดสอบสร้างรหัสโปรแกรม G Code ผลที่ได้คือซอฟต์แวร์ได้ช่วยลดเวลาในการสร้างรหัสโปรแกรม G Code ได้อย่างมาก และยังช่วยลดความผิดพลาดของรหัสโปรแกรม G Code ได้ ซึ่งทำให้ผู้วางแผนไม่ต้องเสียเวลาในการแก้ไขรหัสโปรแกรมที่ผิดพลาดและลดความเมื่อยล้าในการทำงาน ด้วยเหตุนี้จึงส่งผลทำให้เกิดความพึงพอใจต่อผู้วางแผนสำหรับการใช้ซอฟต์แวร์ช่วยสร้างรหัสโปรแกรม G Code นี้

8.2. ข้อเสนอแนะ

ซอฟต์แวร์ที่สร้างขึ้นเป็นระบบคอมพิวเตอร์ต้นแบบที่สามารถสร้างรหัสโปรแกรม G Code เพื่อช่วยในการวางแผนกระบวนการเจาะแผ่นโลหะโดยเครื่องจักร CNC ซึ่งมีข้อเสนอแนะเพื่อการศึกษาเพิ่มเติมดังนี้

1. ในการขั้นตอนของการสร้าง Drawing ของผลิตภัณฑ์ที่ได้มาจากลูกค้าในตอนแรกเพื่อเก็บไว้เป็น Drawing ของทางโรงงาน ควรจะสร้างด้วยรูปแบบที่เหมาะสมกับการใช้งานซอฟต์แวร์เลย เพื่อทำให้ลดเวลาในการสร้าง Drawing ที่เหมาะสมกับซอฟต์แวร์ขึ้นใหม่
2. การใช้ซอฟต์แวร์โดยการป้อนคำสั่งผ่าน AutoCAD Command Prompt ของซอฟต์แวร์ อาจทำให้มันมีความยากในการใช้งาน ผู้วางแผนที่จะใช้ซอฟต์แวร์จึงควรที่จะมีความชำนาญในการใช้โปรแกรม AutoCAD อยู่แล้วและได้รับการฝึกฝนในการใช้งานซอฟต์แวร์ที่สร้างขึ้นพอสมควร สำหรับการใช้งานที่ง่ายขึ้นอาจสร้างซอฟต์แวร์ให้มีรูปแบบการใช้ไดอะลอกบอกซ์ (Dialog Box) เข้ามาทดแทนการป้อนข้อมูลผ่าน AutoCAD Command Prompt ในบางส่วนของซอฟต์แวร์
3. เนื่องจากซอฟต์แวร์ที่พัฒนาขึ้นใช้ระบบการโปรแกรมมิ่งแบบ ARX โดยใช้ภาษา C++ ซึ่งเป็นภาษาที่มีความสามารถมากมายเป็นที่ยอมรับในการสร้างซอฟต์แวร์ที่ซับซ้อนและได้รับการสนับสนุนในการพัฒนาซอฟต์แวร์สำหรับโปรแกรม AutoCAD จากบริษัท Autodesk จำกัด ดังนั้นซอฟต์แวร์จะสามารถเพิ่มขีดความสามารถได้อีกมากมายเพื่อให้สามารถเจาะชิ้นงานที่มีรูปร่างหลากหลายและสามารถใช้รูปแบบคำสั่ง G Code ที่ซับซ้อนขึ้นได้
4. ในขั้นตอนของการนำรหัสโปรแกรม G Code ที่สร้างเสร็จแล้วไปจำลองการเจาะที่ส่วนของจอแสดงผลการเจาะของเครื่องจักร CNC นั้น เป็นขั้นตอนที่สามารถ

ตรวจสอบได้ว่ารหัสที่ป้อนเข้ามีความถูกต้องเพียงใด ซึ่งในส่วนนี้หากเพิ่มความสามารถให้กับซอฟต์แวร์ในการจำลองการเจาะได้ก็จะมีประโยชน์อย่างมาก เนื่องจากไม่ต้องเคลื่อนย้ายรหัสคำสั่งไปยังเครื่องจักร CNC ทุกครั้งเมื่อต้องการตรวจสอบความถูกต้องในการเจาะ

5. การสร้างรหัสโปรแกรม G Code ในส่วนของข้อมูลของหัวเจาะ หากซอฟต์แวร์ได้เพิ่มความสามารถโดยการนำฐานข้อมูลของหัวเจาะซึ่งจะมีข้อมูลเกี่ยวกับรูปร่างของหัวเจาะ ขนาดของหัวเจาะ และตำแหน่งของหัวเจาะที่ติดตั้งอยู่บนแท่นหมุนมาใช้ร่วมกับซอฟต์แวร์ มันจะทำให้ช่วยลดขั้นตอนการป้อนข้อมูลของซอฟต์แวร์ได้ เนื่องจากคุณสมบัติของหัวเจาะจะถูกเรียกมาใช้โดยตรงจากฐานข้อมูลของหัวเจาะ
6. การนำ Drawing ของผลิตภัณฑ์ที่ได้รับจากลูกค้าเพื่อมาใช้เป็น Drawing ที่เหมาะสมในการใช้กับซอฟต์แวร์ หากซอฟต์แวร์มีความสามารถด้าน Image Processingในการแปลง Drawing ที่ได้รับมาซึ่งจะมีรูปแบบที่หลากหลาย เช่น Drawing ของผลิตภัณฑ์อาจถูกจัดเก็บเป็นแฟ้มข้อมูลแบบ DWG หรือ DXF เป็นต้น ให้อยู่ในรูปแบบที่เหมาะสมกับการใช้งานกับซอฟต์แวร์ ก็จะทำให้สามารถลดเวลาในการวางแผนกระบวนการผลิตลงได้อีก

รายการอ้างอิง

ภาษาไทย

กสานต์ ปิ่นเวหาส์. การพัฒนาซอฟต์แวร์คอมพิวเตอร์ช่วยในการวางแผนกระบวนการผลิตสำหรับกระบวนการสร้างรูและกระบวนการกัด. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมอุตสาหการ สาขาวิชาวิศวกรรมอุตสาหการ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2541.

กอบเกียรติ สระอุบล. Advanced AutoCAD การเขียนโปรแกรมไดอะล๊อกบ๊อกซ์และ AutoLISP. กรุงเทพมหานคร: ซีเอ็ดยูเคชั่น, 2542.

สาโรช พรวิจิตรจินดา. การประยุกต์ใช้ไมโครคอมพิวเตอร์กราฟฟิกส์กับการควบคุมการทำงานของเครื่อง ซี เอ็น ซี. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2531.

ภาษาอังกฤษ

Amirouche, F. M. L. Computer-Aided Design and Manufacturing. New Jersey: Prentice-Hall, Inc., 1993.

Ashraf, M. A. Development of An Open Architecture Control for a CNC Milling machine. Master'sThesis. AIT. Thailand. 1999.

Chang, T. C. Expert Process Planning for Manufacturing. Massachusetts: Addison-Wesley, 1990.

Chapman, D. Sams Teach Yourself Visual C++ 6 in 21 Days. Indiana: Sams Publishing, 1998.

Chopra, M. An Object-Oriented Process Planning System for Rotational Components. Master'sThesis. AIT. Thailand. 1993.

Gerner, R.; Middlebrook, M.; and Tanzillo, T. Maximizing AutoCAD R13. Autodesk Press, 1997.

Halevi, G.; and Weill, R. D. Principle of Process Planning: A Logical Approach. London: Chapman & Hall, 1995.

Liberty, J. Sams Teach Yourself C++ in 21 Days. Indiana: Sams Publishing, 1999.

Linardakis, N.; and Mileham, A. R. (1994). A Strategy for Extracting Manufacturing Features for Prismatic Components from CAD. In Advances in Manufacturing Tech VIII: Proceeding of the Tenth National conference on Manufacturing Research. Case, K.; and Newman, S. (eds.). Taylor & Francis: 299-303.

Murata Wiedemann, Operator's Manual CNC Turret Punch Press Centrum 2500.

Ransen, O. AutoCAD Programming in C/C++. West Sussex: John Wiley & sons, 1997.

Rumana, N. An Object-Oriented Process Planning System for Rotational Components. Master's Thesis. AIT. Thailand. 1995.

Vallaponanthan, K. An Object-Oriented Computer-Aided Process Planning System for Prismatic Components. Master'sThesis. AIT. Thailand. 1995.

ภาคผนวก ก.

ข้อกำหนด (Specification) ของเครื่องจักร CNC Turret Punch Press MURATA C2500

ก.1. **Machine**

Capacity	196 kN (20 tons)
Sheet size, Max.	1250 × 1000 mm Without repositioning
Turret stations	22
Crankshaft SPM	420 SPM
Ram stroke	30 mm
Die/stripper clearance	21.3 mm
Press motor	3.7 kW
Turret drive motor	FANUC
Turret speed	25 rpm
Max. Table positioning speed	50 m/min
Table speed override	2 step High, Low
X axis servo motor MODEL 6L	FANUC AC
Y axis servo motor MODEL 7L	FANUC AC
C axis servo motor MODEL 5L	FANUC AC
Punching accuracy	± 0.1 mm
Sheet weight, Max.	75 kg
Turret diameter	860 mm
Sheet thickness, Max.	6.35 mm
Punch diameter, Max.	76 mm dia.
Tool style	114 and 112

Station Range	Diameter
A	~ ϕ 12.7
B	~ ϕ 25
C	~ ϕ 38
D	~ ϕ 51
E	~ ϕ 64

F	~ ϕ 76
Die holders	
A – D	quick change type
E – F	self aligning type
Floor space	2700 x 4280 mm
Frame height	1900 mm
Table height	980 mm
Hydraulic Unit	
Clutch oil pressure	60 kg / cm^2
Lubrication oil pressure	3 kg / cm^2
Machine weight	9500 kg
Air supply	
Pressure	4.5 kg / cm^2
Flow	0.2 m^3 / min
Electrical Input	200 / 220 V. 50 / 60 Hz 15
KVA	

ก.2. **Numerical Control Specifications**

Model	FANUC
Axes control	Simultaneous, 3 axes (X, Y, T or C)
Tape code	EIA
Position command (decimal point	metric (mm), Designation) Incremental
absolute	

	Commands Wiedepiont input
Least command increment mm	X, Y axes : 0.01
	T axis station
number	
	C axis 0.01
degrees	
Axes drives motor/AC servo	Digital servo
	Motor
Servo motor MODEL 6L	X axis: AC
	Y axis: AC
MODEL 7L	
	T axis: AC MODEL 5M
	C axis: AC
MODEL 5L	
Tape reader reader	Photo-electric tape
	Without reel.
Other functions	CRT display unit
	9" (15 x 64) Character Display
	Keyboard type MDI input
	Sequence number display and search
	TH, TV check
	Immediate error display
function	
	Tool change mode
	Optional block skip
	Turret off

Punch off function
Machine lock
Memory storage capacity 160 m of tape
Length (Standard) Option 320
m

Stored stroke limit
Coordinate command/Local
coordinate

Command
Repositioning
Table speed override
Single block
Cycle timer/Sheet counter
Data I/O interface
Punch counter

ก.3. **Index Tool Specification**

Stations	2
Punch range	D Station
RPM	28 rpm
Rotation range	359.99 °
Command form	Absolute
Tool style	112
Least command increment	0.01 deg.

ภาคผนวก ข.

หัวเจาะทั้งหมดของเครื่องจักร CNC Turret Punch Press MURATA C2500

ตารางที่ ข.1 *หัวเจาะทั้งหมดของเครื่องจักร CNC Turret Punch Press MURATA C2500 ที่โรงงานมีอยู่*

STATION A			STATION B		
TYPE*	**ขนาด**	**รูปร่าง**	**TYPE***	**ขนาด**	**รูปร่าง**
C	1.40	○	C	7.00	○
C	1.80	○	C	8.00	○
C	2.00	○	C	12.00	○
C	2.50	○	C	13.00	○
C	2.60	○	C	14.00	○
C	3.00	○	C	15.00	○
C	3.30	○	C	16.00	○
C	3.20	○	C	17.50	○
C	3.50	○	C	20.00	○
C	3.70	○	C	20.50	○
C	4.00	○	R	2.0 x11.0	▭
C	4.20	○	R	2.5 x 2.5	□
C	4.50	○	R	2.5 X 25.0	▭
C	5.00	○	R	3.0 x 3.0	□
C	5.40	○	R	3.0 X 7.0	▭
C	5.50	○	R	4.0 x 4.0	◇
C	6.00	○	R	4.5 x 4.5	□
C	6.20	○	R	4.0 x 32.0	▭
C	6.50	○	R	4.2 x 65.0	▭
C	7.00	○	R	4.5 x 25.0	▭
C	7.33	○	R	5.0 x 5.0	□
C	7.50	○	R	5.0 x 10.0	▭
C	8.00	○	R	5.4 x 2.4	▭
C	8.50	○	R	7.0 x 7.0	□
C	9.00	○	R	10.0 x 10.0	□
C	9.50	○	R	12.5 x 12.5	□
C	10.00	○	R	9.5 x 9.5	□
C	10.50	○	OB	3.0 x 20.0	⬭
C	11.00	○	OB	2.0 x 7.0	⬭

STATION A		
TYPE*	**ขนาด**	**รูปร่าง**
C	11.50	
C	12.00	
C	12.65	
STATION C		
TYPE*	**ขนาด**	**รูปร่าง**
C	22.00	
C	24.00	
C	25.00	
C	25.20	
C	30.20	
C	31.00	
C	38.00	
R	2.0 x 20.5	
R	3 X 30.5	
R	20 X 20 45	
OB	2 X 20.5	
OB	3 X 30.5	
OB	3 X 20	
OB	3 X 30.5	
STATION E		
TYPE*	**ขนาด**	**รูปร่าง**
R	50 X 10	
R	70 X 30	
STATION F		
TYPE*	**ขนาด**	**รูปร่าง**
C	60.00	
R	50 X 20	

STATION B		
TYPE*	**ขนาด**	**รูปร่าง**
OB	3.0 x 30.0	
OB	3.0 x 8.0	
OB	4.5 x 10.0	
OB	6.5 x 16.5	
DD	5.4 x 15	
DD	11.8 x 10.8	
DD	15.9 X 14.0	
DD	10.5 X 12.0	
DD	10.0 X 11.0	
CUP	3.00	
DE	2.0X15.0X2.05	
D6	6.0 X 6.0	
D4R	4-R 3.0	
STATION D		
TYPE	**ขนาด**	**รูปร่าง**
C	47.00	
R	5 X 75	
R	24 X 24	
R	30 X 30	
R**	40 X 3	
R**	40 X 5	
TRI**	30.00 X 15.00	
TRI**	40.00 X 20.00	
M4	TAP	

หมายเหตุ

* TYPE เป็นรหัสย่อที่โรงงานใช้เรียกหัวเจาะ

** เป็นหัวเจาะที่หมุนได้

ภาคผนวก ค.

รหัสคำสั่งของซอฟต์แวร์

ภาคผนวก ค.จะแสดงรหัสคำสั่งสำหรับ 2 ส่วนด้วยกัน ได้แก่ ส่วนของซอฟต์แวร์ ZAmpPunch.arx และ ส่วนของ Help File

ค.1 ซอฟต์แวร์ ZAmpPunch.arx

การสร้างซอฟต์แวร์ ZAmpPunch.arx สร้างขึ้นโดยใช้ภาษา C++ โดยใช้สภาพแวดล้อมของ Visual C++ 6.0 ซึ่งการสร้างซอฟต์แวร์ขึ้นจะต้องมีส่วนประกอบที่จำเป็นต่าง ๆ ภายในโปรเจกต์ (Project) ของสภาพแวดล้อม Visual C++ 6.0 ส่วนประกอบมีรายละเอียดและรหัสคำสั่งดังนี้

ค.1.1. Source Files

ประกอบไปด้วยแฟ้มข้อมูลย่อย ๆ 3 แฟ้มได้แก่

ค.1.1.1. arxtemplate.cpp

เป็นส่วนที่เก็บรหัสภาษา C++ เพื่อใช้ในการติดต่อสื่อสารกับโปรแกรม AutoCAD R14 และเป็นส่วนที่เพิ่มคำสั่งของซอฟต์แวร์ที่เขียนขึ้น มีรหัสคำสั่งดังต่อไปนี้

```
#include <aced.h>
#include <adslib.h>

// Function prototypes
extern void InPunch();
extern void OutPunch();
void initApp();
void unloadApp();
extern "C" AcRx::AppRetCode
acrxEntryPoint( AcRx::AppMsgCode msg, void* pkt);

// End of fn prototyypes


//////////////////////////////////////////////////////////
//
// Rx interface
//
//////////////////////////////////////////////////////////

void initApp()
{
      acedRegCmds->addCommand("ZIP_IN",
                              "ZIP_INS",
                              "inpunch",
                              ACRX_CMD_MODAL,
                              &InPunch);
```

```
      acedRegCmds->addCommand("ZIP_OUT",
                              "ZIP_OUTS",
                              "outpunch",
                              ACRX_CMD_MODAL,
                              &OutPunch);

    ads_printf("\n-------------------------------------------------
----");
      ads_printf("\n   Type INPUNCH or OUTPUNCH to start ZAmpPunch
program");
      ads_printf("\n-------------------------------------------------
------");
}

void unloadApp()
{
        acedRegCmds->removeGroup("ZIP_TEST");
}

//////////////////////////////////////////////////////////
//
// Entry point
//
//////////////////////////////////////////////////////////

extern "C" AcRx::AppRetCode
acrxEntryPoint( AcRx::AppMsgCode msg, void* pkt)
{
  switch( msg )
  {
    case AcRx::kInitAppMsg:
      acrxDynamicLinker->unlockApplication(pkt);
      initApp();
      break;
    case AcRx::kUnloadAppMsg:
      unloadApp();
      break;
    default:
      break;
  }
  return AcRx::kRetOK;
}
```

ค.1.1.2. arxtemplate.def

เป็นแฟ้มข้อมูลที่ทำหน้าที่เป็นตัวกลางในการเชื่อมโยง (Link) ระหว่างแฟ้มข้อมูลทั้งหมดในการสร้างซอฟต์แวร์ ZAmpPunch.arx มีรหัสคำสั่งดังต่อไปนี้

```
DESCRIPTION 'Narathip's ZAmpPunch application version 1.0'
LIBRARY     ZAmpPunch.arx

EXPORTS     acrxEntryPoint
            _SetacrxPtp
            acrxGetApiVersion
```

ค.1.1.3. ZAmpPunch.cpp

เป็นแฟ้มข้อมูลที่เก็บรหัสคำสั่งทั้งหมดของ ZAmpPunch.arx ซึ่งเป็นกลไกหลักในการทำงานของซอฟต์แวร์ มีรหัสคำสั่งดังต่อไปนี้

```
//////////////////////////////////////////////////////////////////////
//////////////////
#include "zglobal.h"

#include <string.h>
#include <stdlib.h>

#include <aced.h>
#include <adslib.h>
#include <adsdlg.h>
#include <math.h>
#include <dbents.h>
#include <dbpl.h>
#include <dbsymtb.h>
#include <dbmain.h>
#include <geassign.h>
//#include <rxregsvc.h>

// Function prototypes
      // Inside punch fn prototypes
void InPunch();
void ZInMain(ZPunch &zObj);
void ZInCreateF(ZPunch &zObj);
void ZGetTool(ZPunch &zObj);
void ZInGetObject(ZPunch &zObj);
void ZInGetTypeCir(ZPunch &zObj);
void ZKB(ZPunch &zObj);
void ZWriteG(ZPunch &zObj);
void ZLoopAsk(ZPunch &zObj);
void ZInAskToolDia(ZPunch &zObj);
void ZAskToolDim(ZPunch &zObj);
void ZInGetTypeRec(ZPunch &zObj);
void ZInGetRecRef(ZPunch &zObj);
void ZInAskCornRad(ZPunch &zObj);
void ZAskArcInOut(ZPunch &zObj);
      // End of inside punch fn prototypes
      //---------------------------------//
      // Outside punch fn prototypes
void OutPunch();
void ZOutMain(ZPunch &zObj);
void ZOutCreateF(ZPunch &zObj);
void ZOutGetObject(ZPunch &zObj);
void ZOutGetLineRef(ZPunch &zObj);
void ZOutGetLineDirec(ZPunch &zObj);
void ZOutAskForJoint(ZPunch &zObj);
void ZCreateJoint(ZPunch &zObj);
      // End of outside punch fn prototypes
// End of fn prototyypes

//////////////////////////////////////////////////////////////////////
//////////////////

// INSIDE PUNCH         INSIDE PUNCH      INSIDE PUNCH

//////////////////////////////////////////////////////////////////////
//////////////////
```

```
void InPunch() // "punch()" func just create object of ZPunch for
using other operation.
{
      ZPunch zObj;

      ZInMain(zObj);
      return;
}

void ZInMain(ZPunch &zObj)
{
      int rc;
      char kw[20];

      ads_printf("\n->ZAmpPunch>>Do  you  want  to  proceed  inside-
punching...");
      ads_initget(0,"Yes No");
      rc = ads_getkword("(<Yes>/No)? :", kw);

      // start mechanism of user input //
      if (rc==RTNONE) // for [enter] or space input
      {
            strcpy(kw,"Yes");
            rc = RTNORM;
      }
      if (rc != RTNORM) // in case of ads_getkword return RTERROR
      {
            ads_printf("\n->ZAmpPunch>>Error");
            return;
      }

      rc = strcmp(kw, "No"); // if kw = No strcmp return 0
      if (rc == 0)
      {
            ads_printf("\n->ZAmpPunch>>User canceled operation");
            return;
      }

      rc = strcmp(kw, "Yes"); // if kw = Yes strcmp return 0
      if (rc == 0)
      {
            ads_printf("\n->ZAmpPunch>>Creating dat file for storing
g-code...");
            ZInCreateF(zObj); // create file and write a header on it
            ads_printf("\n->ZAmpPunch>>Start  generating  g-code  from
the drawing");

            ////////////////////
            //  Start of MAIN LOOPs
            ////////////////////

            strcpy(zObj.loopKw,"tk");
            while (strcmp(zObj.loopKw,"Exit")!=0)
            {
                  ZGetTool(zObj);
                  strcpy(zObj.loopKw,"Continue");
                  while     ((strcmp(zObj.loopKw,"Changetool")     &&
strcmp(zObj.loopKw,"Exit")) != 0)
                  {
                        ZInGetObject(zObj);
                        ZKB(zObj);
```

```
                        ZWriteG(zObj);
                        ads_printf("\n->ZAmpPunch>>End of generating
G-code for this object.");
                        ZLoopAsk(zObj);
                  }

            }

            // End of MAIN LOOPs

            stptr = fopen(zObj.nameFile->resval.rstring,"a");
            fprintf(stptr,"\n/End of inside-punching file.");
            fclose (stptr);
      }
      return;
}

void ZInCreateF(ZPunch &zObj) // fn built to create file and open it
for write
{
      int res;
      zObj.nameFile = ads_newrb(RTSTR);
      res = ads_getfiled ("Please type new inside-punching file
name... ", // title
                                    "C:\\", // no suggestion
                                    "dat", // extension
                                    NEWFILE, // flags
                                    zObj.nameFile) ;        //
where to put result
      if (res == RTNORM)
      {
      // Check which type of result buffer has been returned
            if (zObj.nameFile->restype == RTSTR)
            {
                  ads_printf ("\n->ZAmpPunch>>The user selected
<%s>",
                                                zObj.nameFile-
>resval.rstring) ;
            }
            else if (zObj.nameFile->restype == RTSHORT)
            {
                  ads_printf ("\n->ZAmpPunch>>Request to type at the"
                                                "       command
line.") ;
            }
            else
            {
                  ads_printf ("\n->ZAmpPunch>>Strange return
type:%d",

zObj.nameFile->restype) ;
            }
      }
      else
      {
            ads_printf ("\n->ZAmpPunch>>User abandoned dialog box") ;
            fclose (stptr);
            return;
      }

      if((stptr = fopen(zObj.nameFile->resval.rstring,"w"))!=NULL)
```

```
      {
            fprintf (stptr,"/Start inside-punching G-Code");
            fclose (stptr);
      }
      return;
}

void ZGetTool(ZPunch &zObj)
{
      int rc;

      ads_printf("\n->ZAmpPunch>>Enter tool number on the turret to
be used ");
      ads_initget(0,"01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
17 18 19 20 21 22");
      rc = ads_getkword("(<01> to 22) :",zObj.kwTool);

            if ((rc != RTNORM) && (rc != RTNONE))
            {
            ads_printf("\n->ZAmpPunch>>Nothing selected.");
            }

            if ((rc == RTNONE) || (strcmp(zObj.kwTool, "01")) == 0)
            {
                  strcpy(zObj.kwTool,"01");
            }
            else     if     ((strcmp(zObj.kwTool,     "02"))     ||
(strcmp(zObj.kwTool, "03")) ||
                 (strcmp(zObj.kwTool, "04")) || (strcmp(zObj.kwTool,
"05")) ||
                  (strcmp(zObj.kwTool,          "06"))          ||
(strcmp(zObj.kwTool, "07")) ||
                  (strcmp(zObj.kwTool,          "08"))          ||
(strcmp(zObj.kwTool, "09")) ||
                  (strcmp(zObj.kwTool,          "10"))          ||
(strcmp(zObj.kwTool, "11")) ||
                  (strcmp(zObj.kwTool,          "12"))          ||
(strcmp(zObj.kwTool, "13")) ||
                  (strcmp(zObj.kwTool,          "14"))          ||
(strcmp(zObj.kwTool, "15")) ||
                  (strcmp(zObj.kwTool,          "16"))          ||
(strcmp(zObj.kwTool, "17")) ||
                  (strcmp(zObj.kwTool,          "18"))          ||
(strcmp(zObj.kwTool, "19")) ||
                  (strcmp(zObj.kwTool,          "20"))          ||
(strcmp(zObj.kwTool, "21")) ||
                  (strcmp(zObj.kwTool, "22")) == 0)
            {
            }
     ads_printf("\n->ZAmpPunch>>You have chosen Tool number T%s",
zObj.kwTool);
      return;
}

void ZInGetObject(ZPunch &zObj)
{
    ads_name eName;
      ads_point pt;
      AcDbObjectId eId;
      AcDbEntity* ePtr;
      AcDbCircle* cPtr;
```

```
      AcDbPolyline* plPtr;
      AcDbArc* aPtr;
      int rc;

      //RECEIVE ENTITY TO KEEP AT name "eName"
      rc = ads_entsel("\n->ZAmpPunch>>Select object: ", eName, pt);

      if (rc != RTNORM)
      {
        ads_printf("\n-->->ZAmpPunch>>Nothing selected.\n", rc);
            return;
    }

      //RECEIVE Id OF ENTITY FROM eName
    Acad::ErrorStatus es = acdbGetObjectId(eId, eName);

      if (es != Acad::eOk)
      {
            ads_printf("\n->ZAmpPunch>>Error getting object id.");
            return;
      }

      //RECEIVE Oject OF ENTITY FROM eId
    es = acdbOpenObject(ePtr, eId, AcDb::kForWrite); // for write
'cos change color

      if (es != Acad::eOk)
      {
            ads_printf("\n->ZAmpPunch>>Error   opening   object   for
write.");
            return; // for exiting if clause when this function
return void
      }

      ////////////////////////////////////////////////////////
      // Start checking what entity is and keep its         //
      // attributes in the global variables defined in zglobal.h//
      ////////////////////////////////////////////////////////

      /////////////////
      // Circle      //
      /////////////////

      if(AcDbCircle::cast(ePtr)!=NULL)
      {
            ads_printf("\n->ZAmpPunch>>This object is Circle");

            // cast ePtr (acdbentity) to cPtr (acdbcircle)
            cPtr = AcDbCircle::cast(ePtr);

            zObj.eType = 1;

            // change color of object
            ads_printf("\n->ZAmpPunch>>Changing color..");
            cPtr->setColorIndex(4);

            // Get attributes
            zObj.cCenter = cPtr->center();
            zObj.cRadius = cPtr->radius();

            cPtr->close();
```

```
            ePtr->close();

            ZInGetTypeCir(zObj);
            return;
      }

      ////////////////
      // Polyline    //
      ////////////////
      else if(AcDbPolyline::cast(ePtr)!=NULL)
      {
            AcGePoint3d dum;
            double bulge0,bulge1,bulge2,bulge3;
            unsigned int numVer;

            ads_printf("\n->ZAmpPunch>>This object is Polyline");
            plPtr = AcDbPolyline::cast(ePtr);

            // start checking what type PLINE is by checking bulge
and vertex
            if ((plPtr->isClosed())!=Adesk::kTrue)
            {
                  ads_printf("\n->ZAmpPunch>>Object is not closed");
                  plPtr->close();
                  ePtr->close();
                  return;
            }

            // change color of object
            ads_printf("\n->ZAmpPunch>>Changing color..");
            plPtr->setColorIndex(4);

            numVer = plPtr->numVerts();

            if (numVer!= 4)
            {
                  ads_printf("\n->ZAmpPunch>>Object's  vertex  is  not
suitable.");
                  plPtr->close();
                  ePtr->close();
                  return;
            }

//          ads_printf("\n->ZAmpPunch>>Number of vertex in pline is
%d",numVer);

            // determine point of vertex and value of bulge
            plPtr->getPointAt(0,zObj.recPt0);
            plPtr->getPointAt(1,zObj.recPt1);
            plPtr->getPointAt(2,zObj.recPt2);
            plPtr->getPointAt(3,zObj.recPt3);

            plPtr->getBulgeAt(0,bulge0);
            plPtr->getBulgeAt(1,bulge1);
            plPtr->getBulgeAt(2,bulge2);
            plPtr->getBulgeAt(3,bulge3);

            // start checking type of polyline, here supports just
rectangle

            ////////////////////
```

```
            // Polyline - RECTANGLE
            ////////////////////
            if
((bulge0==0.00)&&(bulge1==0.00)&&(bulge2==0.00)&&(bulge3==0.00))
            {
                  // convert point to standard form
                  if    ((zObj.recPt0.x    <    zObj.recPt2.x)    &&
(zObj.recPt0.y < zObj.recPt2.y))
                  {
                        zObj.recPt0 = zObj.recPt0;
                        zObj.recPt1 = zObj.recPt1;
                        zObj.recPt2 = zObj.recPt2;
                        zObj.recPt3 = zObj.recPt3;
                  }
                  else   if   ((zObj.recPt0.x   <   zObj.recPt2.x)   &&
(zObj.recPt0.y > zObj.recPt2.y))
                  {
                        dum = zObj.recPt3;
                        zObj.recPt3 = zObj.recPt0;
                        zObj.recPt0 = dum;

                        dum = zObj.recPt2;
                        zObj.recPt2 = zObj.recPt1;
                        zObj.recPt1 = dum;
                  }
                  else   if   ((zObj.recPt0.x   >   zObj.recPt2.x)   &&
(zObj.recPt0.y > zObj.recPt2.y))
                  {
                        dum = zObj.recPt2;
                        zObj.recPt2 = zObj.recPt0;
                        zObj.recPt0 = dum;

                        dum = zObj.recPt3;
                        zObj.recPt3 = zObj.recPt1;
                        zObj.recPt1 = dum;
                  }
                  else
                  {
                        dum = zObj.recPt1;
                        zObj.recPt1 = zObj.recPt0;
                        zObj.recPt0 = dum;

                        dum = zObj.recPt2;
                        zObj.recPt2 = zObj.recPt3;
                        zObj.recPt3 = dum;
                  }
                  // check shape of rectangle
                  if
((zObj.recPt0.x!=zObj.recPt3.x)||(zObj.recPt0.y!=zObj.recPt1.y)||(zOb
j.recPt1.x!=zObj.recPt2.x)||(zObj.recPt2.y!=zObj.recPt3.y))
                  {
                        ads_printf("\n->ZAmpPunch>>Cannot    generate
code for this kind of rectangle");
                        plPtr->close();
                        ePtr->close();
                        return;
                  }

                  zObj.eType = 2;
                  ads_printf("\n->ZAmpPunch>>This      object      is
Rectangle");
```

```
                plPtr->close();
                ePtr->close();

                ZInGetTypeRec(zObj);
                ZInGetRecRef(zObj);
                return;
            }
            // end of rectangle of polyline
            ////////////////////
            // Polyline - OBLONG
            ////////////////////

            else if (( (bulge0==0)&&
                        (     ((bulge1-1)<0.0000001)&&((bulge1-1)>-
0.0000001) )&&
                        (bulge2==0)&&
                        (     ((bulge3-1)<0.0000001)&&((bulge3-1)>-
0.0000001) ) )
                        ||
                       ( (bulge1==0)&&
                        (     ((bulge0-1)<0.0000001)&&((bulge0-1)>-
0.0000001) )&&
                        (bulge3==0)&&
                        (     ((bulge2-1)<0.0000001)&&((bulge2-1)>-
0.0000001) ) )
                        ||
                       ( (bulge0==0)&&
                        (     ((bulge1+1)<0.0000001)&&((bulge1+1)>-
0.0000001) )&&
                        (bulge2==0)&&
                        (     ((bulge3+1)<0.0000001)&&((bulge3+1)>-
0.0000001) ) )
                        ||
                       ( (bulge1==0)&&
                        (     ((bulge0+1)<0.0000001)&&((bulge0+1)>-
0.0000001) )&&
                        (bulge3==0)&&
                        (     ((bulge2+1)<0.0000001)&&((bulge2+1)>-
0.0000001) ) )   )
            {
                // convert point 0 - 3 to standard pattern
                if ((bulge0 + bulge1 + bulge2 + bulge3) > 0) // CCW
drawing
                {
                      if
((zObj.recPt0.x<zObj.recPt2.x)&&(zObj.recPt0.y<zObj.recPt2.y))
                      {
                            zObj.recPt0 = zObj.recPt0;
                            zObj.recPt1 = zObj.recPt1;
                            zObj.recPt2 = zObj.recPt2;
                            zObj.recPt3 = zObj.recPt3;
                      }
                      else                                          if
((zObj.recPt0.x>zObj.recPt2.x)&&(zObj.recPt0.y<zObj.recPt2.y))
                      {
                            dum = zObj.recPt2;
                            zObj.recPt2 = zObj.recPt1;
                            zObj.recPt1 = zObj.recPt0;
                            zObj.recPt0 = zObj.recPt3;
                            zObj.recPt3 = dum;
```

```
                        }
                        else                                    if
((zObj.recPt0.x>zObj.recPt2.x)&&(zObj.recPt0.y>zObj.recPt2.y))
                        {
                              dum = zObj.recPt2;
                              zObj.recPt2 = zObj.recPt0;
                              zObj.recPt0 = dum;

                              dum = zObj.recPt1;
                              zObj.recPt1 = zObj.recPt3;
                              zObj.recPt3 = dum;
                        }
                        else                                    if
((zObj.recPt0.x<zObj.recPt2.x)&&(zObj.recPt0.y>zObj.recPt2.y))
                        {
                              dum = zObj.recPt0;
                              zObj.recPt0 = zObj.recPt1;
                              zObj.recPt1 = zObj.recPt2;
                              zObj.recPt2 = zObj.recPt3;
                              zObj.recPt3 = dum;
                        }
                  }
                  else if ((bulge0 + bulge1 + bulge2 + bulge3) < 0)
// CW Drawing
                  {
                        if
((zObj.recPt0.x<zObj.recPt2.x)&&(zObj.recPt0.y<zObj.recPt2.y))
                        {
                              zObj.recPt0 = zObj.recPt0;
                              dum = zObj.recPt1;
                              zObj.recPt1 = zObj.recPt3;
                              zObj.recPt3 = dum;
                              zObj.recPt2 = zObj.recPt2;
                        }
                        else                                    if
((zObj.recPt0.x<zObj.recPt2.x)&&(zObj.recPt0.y>zObj.recPt2.y))
                        {
                              dum = zObj.recPt0;
                              zObj.recPt0 = zObj.recPt3;
                              zObj.recPt3 = dum;

                              dum = zObj.recPt1;
                              zObj.recPt1 = zObj.recPt2;
                              zObj.recPt2 = dum;
                        }
                        else                                    if
((zObj.recPt0.x>zObj.recPt2.x)&&(zObj.recPt0.y>zObj.recPt2.y))
                        {
                              dum = zObj.recPt0;
                              zObj.recPt0 = zObj.recPt2;
                              zObj.recPt2 = dum;
                              zObj.recPt1 = zObj.recPt1;
                              zObj.recPt3 = zObj.recPt3;
                        }
                        else                                    if
((zObj.recPt0.x>zObj.recPt2.x)&&(zObj.recPt0.y<zObj.recPt2.y))
                        {
                              dum = zObj.recPt0;
                              zObj.recPt0 = zObj.recPt1;
                              zObj.recPt1 = dum;
```

```
                              dum = zObj.recPt2;
                              zObj.recPt2  = zObj.recPt3;
                              zObj.recPt3 = dum;
                        }
                  }
                  // End of changing point0 - 3 to standard pattern

                  // check shape of oblong
                  if
((zObj.recPt0.x!=zObj.recPt3.x)||(zObj.recPt0.y!=zObj.recPt1.y)||(zOb
j.recPt1.x!=zObj.recPt2.x)||(zObj.recPt2.y!=zObj.recPt3.y))
                  {
                        ads_printf("\n->ZAmpPunch>>Cannot    generate
code for this kind of oblong");
                        plPtr->close();
                        ePtr->close();
                        return;
                  }

                  ads_printf("\n->ZAmpPunch>>This object is Oblong");
                  zObj.eType = 6;

                  plPtr->close();
                  ePtr->close();
                  return;
            }
            else
            {
                  ads_printf("\n->ZAmpPunch>>Cannot   do   with   this
polyline");

                  plPtr->close();
                  ePtr->close();
                  return;
            }
            // end of polyline
      }

      /////////////////
      // Arc         //
      /////////////////

      else if (AcDbArc::cast(ePtr)!=NULL)
      {
            ads_printf("\n->ZAmpPunch>>This Object is Arc");

            aPtr = AcDbArc::cast(ePtr);
            // change color of object
            ads_printf("\n->ZAmpPunch>>Changing color..");
            aPtr->setColorIndex(4);

            zObj.arcCenter = aPtr->center();
            zObj.aRad = aPtr->radius();
            zObj.aStartAng = 180/pi*(aPtr->startAngle());
            zObj.aEndAng = 180/pi*(aPtr->endAngle());

            zObj.eType = 3;

            aPtr->close();
            ePtr->close();
```

```
            ZAskArcInOut(zObj);
            return;
      }

      else
      {
            ads_printf("\n->ZAmpPunch>>This object is out of scope");
      }
      return;
}

void ZInGetTypeCir(ZPunch &zObj)
{
      int rc;

      ads_printf("\n->ZAmpPunch>>Enter circle punching type");
      ads_initget(0,"1 2 3");
      rc = ads_getkword("(<1>/2/3) :",zObj.kwTypeCir);

      if ((rc != RTNORM) && (rc != RTNONE))
      {
            ads_printf("\n->ZAmpPunch>>Nothing selected.");
    }

      if ((rc == RTNONE) || (strcmp(zObj.kwTypeCir, "1")) == 0)
      {
            strcpy(zObj.kwTypeCir,"1");
            ads_printf("\n->ZAmpPunch>>You    have    chosen    circle
punching type %s",zObj.kwTypeCir);
            return;
      }
      else if ((strcmp(zObj.kwTypeCir, "2"))||(strcmp(zObj.kwTypeCir,
"3")) == 0)
      {
            ads_printf("\n->ZAmpPunch>>You    have    chosen    circle
punching type %s",zObj.kwTypeCir);
            return;
      }
}

void ZKB(ZPunch &zObj)
{
      switch (zObj.eType)
      {
      case 1:
            // eType = 1 means Circle
            {
                  if (strcmp(zObj.kwTypeCir,"1")==0)
                  {
                        zObj.cCenter.x = zObj.cCenter.x;
                        zObj.cCenter.y = zObj.cCenter.y;
                        zObj.cDia = zObj.cRadius * 2;
                        return;
                  }
                  else if (strcmp(zObj.kwTypeCir,"2")==0)
                  {
                        zObj.cCenter.x = zObj.cCenter.x;
                        zObj.cCenter.y = zObj.cCenter.y;
                        zObj.cDia = zObj.cRadius * 2;
                        ZInAskToolDia(zObj);
                        return;
```

```
                  }
                  else if (strcmp(zObj.kwTypeCir,"3")==0)
                  {
                        zObj.cCenter.x = zObj.cCenter.x;
                        zObj.cCenter.y = zObj.cCenter.y;
                        zObj.cDia = zObj.cRadius * 2;
                        ZInAskToolDia(zObj);
                        return;
                  }
            };
            break;
      case 2:
            // eType = 2 means Rectangle
            {
                  if (strcmp(zObj.kwTypeRec,"1")==0)
                  {
                        if (zObj.recPt0==zObj.recRfPt) // ref at pt0
                        {
                              strcpy(zObj.recDirectionLR,"R");
                              zObj.mcLength1   =   fabs(zObj.recPt1.x-
zObj.recPt0.x);
                              ZAskToolDim(zObj);
                              zObj.pitchX = zObj.toolDimx-1.0;
                        }
                        else if(zObj.recPt1==zObj.recRfPt) // ref at
pt1
                        {
                              strcpy(zObj.recDirectionLR,"L");
                              zObj.mcLength1   =   fabs(zObj.recPt1.x-
zObj.recPt0.x);
                              ZAskToolDim(zObj);
                              zObj.pitchX = zObj.toolDimx-1.0;
                        }
                        return;
                  }

                  else if (strcmp(zObj.kwTypeRec,"2")==0)
                  {
                        if (zObj.recPt0==zObj.recRfPt) // ref at pt0
                        {
                              strcpy(zObj.recDirectionUD,"U");
                              zObj.mcLength1   =   fabs(zObj.recPt3.y-
zObj.recPt0.y);
                              ZAskToolDim(zObj);
                              zObj.pitchY = zObj.toolDimy-1.0;
                        }
                        else if (zObj.recPt3==zObj.recRfPt) // ref at
pt3
                        {
                              strcpy(zObj.recDirectionUD,"D");
                              zObj.mcLength1   =   fabs(zObj.recPt3.y-
zObj.recPt0.y);
                              ZAskToolDim(zObj);
                              zObj.pitchY = zObj.toolDimy-1.0;
                        }
                        return;
                  }

                  else     if     ((strcmp(zObj.kwTypeRec,"4")     ==
0)||(strcmp(zObj.kwTypeRec,"3")==0))
                  {
```

```
                        if (zObj.recPt0==zObj.recRfPt) // ref at pt0
                        {
                              zObj.refAt = 0;
                              strcpy(zObj.recDirectionLR,"R");
                              zObj.mcLength1   =   fabs(zObj.recPt1.x-
zObj.recPt0.x);
                              ZAskToolDim(zObj);
                              zObj.pitchX = zObj.toolDimx-1.0;
                              strcpy(zObj.recDirectionUD,"U");
                              zObj.mcLength2   =   fabs(zObj.recPt2.y-
zObj.recPt1.y);
                              zObj.pitchY = zObj.toolDimy-1.0;
                        }
                        else if (zObj.recPt1==zObj.recRfPt) // ref at
pt1
                        {
                              zObj.refAt = 1;
                              strcpy(zObj.recDirectionUD,"U");
                              zObj.mcLength1   =   fabs(zObj.recPt2.y-
zObj.recPt1.y);
                              ZAskToolDim(zObj);
                              zObj.pitchY = zObj.toolDimy-1.0;
                              strcpy(zObj.recDirectionLR,"L");
                              zObj.mcLength2   =   fabs(zObj.recPt2.x-
zObj.recPt3.x);
                              zObj.pitchX = zObj.toolDimx-1.0;
                        }
                        else if (zObj.recPt2==zObj.recRfPt) // ref at
pt2
                        {
                              zObj.refAt = 2;
                              strcpy(zObj.recDirectionLR,"L");
                              zObj.mcLength1   =   fabs(zObj.recPt2.x-
zObj.recPt3.x);
                              ZAskToolDim(zObj);
                              zObj.pitchX = zObj.toolDimx-1.0;
                              strcpy(zObj.recDirectionUD,"D");
                              zObj.mcLength2   =   fabs(zObj.recPt3.y-
zObj.recPt0.y);
                              zObj.pitchY = zObj.toolDimy-1.0;
                        }
                        else if (zObj.recPt3==zObj.recRfPt) // ref at
pt3
                        {
                              zObj.refAt = 3;
                              strcpy(zObj.recDirectionUD,"D");
                              zObj.mcLength1   =   fabs(zObj.recPt3.y-
zObj.recPt0.y);
                              ZAskToolDim(zObj);
                              zObj.pitchY = zObj.toolDimy-1.0;
                              strcpy(zObj.recDirectionLR,"R");
                              zObj.mcLength2   =   fabs(zObj.recPt1.x-
zObj.recPt0.x);
                              zObj.pitchX = zObj.toolDimx-1.0;
                        }
                        return;
                  }
                  else if (strcmp(zObj.kwTypeRec,"5")==0)
                  {
                        zObj.mcLength1      =      fabs(zObj.recPt1.x-
zObj.recPt0.x);
```

```
                        ZAskToolDim(zObj);
                        zObj.pitchX = zObj.toolDimx-1.0;
                        zObj.mcLength2     =     fabs(zObj.recPt3.y-
zObj.recPt0.y);
                        zObj.pitchY = zObj.toolDimy-1.0;
                        ZInAskCornRad(zObj);
                        return;
                  }
                  else if (strcmp(zObj.kwTypeRec,"6")==0)
                  {
                        zObj.recCenPt1.x=zObj.recRfPt.x;
                        zObj.recCenPt1.y=zObj.recRfPt.y;
                  }
            };
            break;
      case 3:
            // eType = 3 means Arc
            {
                  zObj.aRad = zObj.aRad;
                  ZInAskToolDia(zObj);
                  zObj.mcStartAng = zObj.aStartAng;
                  zObj.mcTravAng = fabs(zObj.aEndAng-zObj.aStartAng);
                  zObj.aPitch = 2.0;
            };
            break;
      case 4:
            // eType = 4 means Line
            {
                  ZOutGetLineDirec(zObj);
                  ZAskToolDim(zObj);
                  ZOutGetLineRef(zObj);
                  ZOutAskForJoint(zObj);

/*1.1*/             if (strcmp(zObj.jointYN,"Yes")==0) // have joint
                  {
                        ZCreateJoint(zObj);
                        zObj.jointVal = 0.3;
/*2.1*/                 if (zObj.horVer==0) // vertical
                        {
                              zObj.mcLength1=(fabs(zObj.lStartPt.y-
zObj.lEndPt.y)-2*zObj.jointVal);
                              zObj.toolDimy = zObj.toolDimy;
                              zObj.pitchY=zObj.toolDimy-1.0;
                              zObj.toolDimx = zObj.toolDimx;
/*3.1*/                       if (strcmp(zObj.lineDirecLR,"L")==0) //
left side of line
                              {
/*4.1*/                             if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.x=zObj.lineRefPt.x-zObj.toolDimx;

      zObj.lineRefPt.y=zObj.lineRefPt.y-zObj.jointVal;

      strcpy(zObj.recDirectionUD,"D");
                                    }
/*4.2*/                             else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                    {
```

```
      zObj.lineRefPt.x=zObj.lineRefPt.x-zObj.toolDimx;

      zObj.lineRefPt.y=zObj.lineRefPt.y+zObj.jointVal;

      strcpy(zObj.recDirectionUD,"U");
                                    }
                               }
/*3.2*/                        else if
(strcmp(zObj.lineDirecLR,"R")==0) // right side of line
                               {
/*4.1*/                              if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.x=zObj.lineRefPt.x;

      zObj.lineRefPt.y=zObj.lineRefPt.y-zObj.jointVal;

      strcpy(zObj.recDirectionUD,"D");
                                    }
/*4.2*/                             else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                    {

      zObj.lineRefPt.x=zObj.lineRefPt.x;

      zObj.lineRefPt.y=zObj.lineRefPt.y+zObj.jointVal;

      strcpy(zObj.recDirectionUD,"U");
                                    }
                               }
                          }
/*2.2*/                   else if (zObj.horVer==1) // horizontal
                          {
                               zObj.mcLength1=(fabs(zObj.lEndPt.x-
zObj.lStartPt.x)-2*zObj.jointVal);
                               zObj.toolDimx=zObj.toolDimx;
                               zObj.pitchX=zObj.toolDimx-1.0;
                               zObj.toolDimy=zObj.toolDimy;
/*3.1*/                        if (strcmp(zObj.lineDirecUD,"U")==0) //
upside of line
                               {
/*4.1*/                              if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.y=zObj.lineRefPt.y;

      zObj.lineRefPt.x=zObj.lineRefPt.x+zObj.jointVal;

      strcpy(zObj.recDirectionLR,"R");
                                    }
/*4.2*/                             else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                    {

      zObj.lineRefPt.y=zObj.lineRefPt.y;

      zObj.lineRefPt.x=zObj.lineRefPt.x-zObj.jointVal;
```

```
      strcpy(zObj.recDirectionLR,"L");
                                    }
                               }
/*3.2*/                        else if
(strcmp(zObj.lineDirecUD,"D")==0) // downside of line
                               {
/*4.1*/                              if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.y=zObj.lineRefPt.y-zObj.toolDimy;

      zObj.lineRefPt.x=zObj.lineRefPt.x+zObj.jointVal;

      strcpy(zObj.recDirectionLR,"R");
                                    }
/*4.2*/                              else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                    {

      zObj.lineRefPt.y=zObj.lineRefPt.y-zObj.toolDimy;

      zObj.lineRefPt.x=zObj.lineRefPt.x-zObj.jointVal;

      strcpy(zObj.recDirectionLR,"L");
                                    }
                               }
                          }
                     }
/*1.2*/              else if (strcmp(zObj.jointYN,"No")==0) // have no
joint
                     {
/*2.1*/                     if (zObj.horVer==0) // vertical
                          {
                                    zObj.lineRefPt.y=zObj.lineRefPt.y;
                                    zObj.mcLength1=fabs(zObj.lStartPt.y-
zObj.lEndPt.y);
                                    zObj.toolDimy=zObj.toolDimy;
                                    zObj.pitchY=zObj.toolDimy-1.0;
                                    zObj.toolDimx=zObj.toolDimx;
/*3.1*/                             if (strcmp(zObj.lineDirecLR,"L")==0) //
leftside of line
                                    {
/*4.1*/                              if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.x=zObj.lineRefPt.x-zObj.toolDimx;

      strcpy(zObj.recDirectionUD,"D");
                                    }
/*4.2*/                              else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                    {

      zObj.lineRefPt.x=zObj.lineRefPt.x-zObj.toolDimx;

      strcpy(zObj.recDirectionUD,"U");
                                    }
                               }
```

```
/*3.2*/                         else if
(strcmp(zObj.lineDirecLR,"R")==0) // downside of line
                                {
/*4.1*/                               if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.x=zObj.lineRefPt.x;

      strcpy(zObj.recDirectionUD,"D");
                                    }
/*4.2*/                               else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                    {

      zObj.lineRefPt.x=zObj.lineRefPt.x;

      strcpy(zObj.recDirectionUD,"U");
                                    }
                                }
                           }
/*2.2*/                    else if (zObj.horVer==1) // horizontal
                           {
                                zObj.lineRefPt.x=zObj.lineRefPt.x;
                                zObj.mcLength1=fabs(zObj.lEndPt.x-
zObj.lStartPt.x);
                                zObj.toolDimx=zObj.toolDimx;
                                zObj.pitchX=zObj.toolDimx-1.0;
                                zObj.toolDimy=zObj.toolDimy;
/*3.1*/                         if (strcmp(zObj.lineDirecUD,"U")==0) //
upside of line
                                {
/*4.1*/                               if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.y=zObj.lineRefPt.y;

      strcpy(zObj.recDirectionLR,"R");
                                    }
/*4.2*/                               else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                    {

      zObj.lineRefPt.y=zObj.lineRefPt.y;

      strcpy(zObj.recDirectionLR,"L");
                                    }
                                }
/*3.2*/                         else if
(strcmp(zObj.lineDirecUD,"D")==0) // downside of line
                                {
/*4.1*/                               if
(zObj.lineRefPt==zObj.lStartPt) // ref at linestart
                                    {

      zObj.lineRefPt.y=zObj.lineRefPt.y-zObj.toolDimy;

      strcpy(zObj.recDirectionLR,"R");
                                    }
```

```
/*4.2*/                                 else if
(zObj.lineRefPt==zObj.lEndPt) // ref at lineend
                                        {

      zObj.lineRefPt.y=zObj.lineRefPt.y-zObj.toolDimy;

      strcpy(zObj.recDirectionLR,"L");
                                        }
                                  }
                            }
                      }
            };
            break;
      case 5:
            // eType = 5 means polyline fillet
            {
                  if (zObj.triOrCorn==0) // triangle
                  {
                        ZAskToolDim(zObj);
                        if(zObj.triStyle==1)
                        {
                              zObj.recCenPt1.x=zObj.fPt1.x-
zObj.toolDimx/2;
                              zObj.recCenPt1.y=zObj.fPt1.y;
                        }
                        else if(zObj.triStyle==2)
                        {

      zObj.recCenPt1.x=zObj.fPt1.x+zObj.toolDimx/2;
                              zObj.recCenPt1.y=zObj.fPt1.y;

                        }
                        else if(zObj.triStyle==3)
                        {
                              zObj.recCenPt1.x=zObj.fPt1.x;
                              zObj.recCenPt1.y=zObj.fPt1.y-
zObj.toolDimy/2;
                        }
                        else if(zObj.triStyle==4)
                        {
                              zObj.recCenPt1.x=zObj.fPt1.x;

      zObj.recCenPt1.y=zObj.fPt1.y+zObj.toolDimy/2;
                        }
                  }
                  else if(zObj.triOrCorn==1) // corner fillet
                  {
                        ZAskToolDim(zObj);
                        if      ((zObj.toolDimx>=fabs(zObj.fPt0.x-
zObj.fPt1.x))&&(zObj.toolDimy>=fabs(zObj.fPt2.y-zObj.fPt1.y)))
                        {
                              zObj.cfType=1;
                              // center punch at corner
                              ads_printf("\n->ZAmpPunch>>Center punch
at corner.");
                              if (zObj.cfStyle==1)
                              {
                                    zObj.recCenPt1.x=zObj.fPt1.x-
zObj.toolDimx/2;

      zObj.recCenPt1.y=zObj.fPt1.y+zObj.toolDimy/2;
```

```
                              }
                              else if (zObj.cfStyle==2)
                              {

      zObj.recCenPt1.x=zObj.fPt1.x+zObj.toolDimx/2;

      zObj.recCenPt1.y=zObj.fPt1.y+zObj.toolDimy/2;
                              }
                              else if (zObj.cfStyle==3)
                              {
                                    zObj.recCenPt1.x=zObj.fPt1.x-
zObj.toolDimx/2;
                                    zObj.recCenPt1.y=zObj.fPt1.y-
zObj.toolDimy/2;
                              }
                              else if (zObj.cfStyle==4)
                              {

      zObj.recCenPt1.x=zObj.fPt1.x+zObj.toolDimx/2;
                                    zObj.recCenPt1.y=zObj.fPt1.y-
zObj.toolDimy/2;
                              }
                        }
                        else   if   ((zObj.toolDimx>=fabs(zObj.fPt0.x-
zObj.fPt1.x))&&(zObj.toolDimy<fabs(zObj.fPt2.y-zObj.fPt1.y)))
                        {
                              zObj.cfType=2;
                              // rec punch up or down
                              ads_printf("\n->ZAmpPunch>>REC vertical
punch at corner.");
                              zObj.mcLength1=fabs(zObj.fPt1.y-
zObj.fPt2.y);
                              zObj.pitchY=zObj.toolDimy-1.0;
                              if
((zObj.cfStyle==1)||(zObj.cfStyle==3))
                              {
                                    zObj.fRefPt.x=zObj.fPt1.x-
zObj.toolDimx;
                                    zObj.fRefPt.y=zObj.fPt1.y;

                                    if (zObj.cfStyle==1)
                                    {

      strcpy(zObj.recDirectionUD,"U");
                                    }
                                    else if(zObj.cfStyle==3)
                                    {

      strcpy(zObj.recDirectionUD,"D");
                                    }
                              }
                              else if
((zObj.cfStyle==2)||(zObj.cfStyle==4))
                              {
                                    zObj.fRefPt.x=zObj.fPt1.x;
                                    zObj.fRefPt.y=zObj.fPt1.y;

                                    if (zObj.cfStyle==2)
                                    {

      strcpy(zObj.recDirectionUD,"U");
```

```
                                    }
                                    else if(zObj.cfStyle==4)
                                    {

      strcpy(zObj.recDirectionUD,"D");
                                    }
                              }
                        }
                        else if ((zObj.toolDimx<fabs(zObj.fPt0.x-
zObj.fPt1.x))&&(zObj.toolDimy>=fabs(zObj.fPt2.y-zObj.fPt1.y)))
                        {
                              // rec punch left or right
                              zObj.cfType=3;
                              ads_printf("\n->ZAmpPunch>>REC
horizontal punch at corner.");

                              zObj.mcLength1=fabs(zObj.fPt1.x-
zObj.fPt0.x);
                              zObj.pitchX=zObj.toolDimx-1.0;
                              if
((zObj.cfStyle==1)||(zObj.cfStyle==2))
                              {
                                    zObj.fRefPt.x=zObj.fPt1.x;
                                    zObj.fRefPt.y=zObj.fPt1.y;
                                    if (zObj.cfStyle==1)
                                    {

      strcpy(zObj.recDirectionLR,"L");
                                    }
                                    else if (zObj.cfStyle==2)
                                    {

      strcpy(zObj.recDirectionLR,"R");
                                    }

                              }
                              else if
((zObj.cfStyle==3)||(zObj.cfStyle==4))
                              {
                                    zObj.fRefPt.x=zObj.fPt1.x;
                                    zObj.fRefPt.y=zObj.fPt1.y-
zObj.toolDimy;
                                    if (zObj.cfStyle==3)
                                    {

      strcpy(zObj.recDirectionLR,"L");
                                    }
                                    else if (zObj.cfStyle==4)
                                    {

      strcpy(zObj.recDirectionLR,"R");
                                    }
                              }
                        }
                        else   if   ((zObj.toolDimx<fabs(zObj.fPt0.x-
zObj.fPt1.x))&&(zObj.toolDimy<fabs(zObj.fPt2.y-zObj.fPt1.y)))

                        {
                              zObj.cfType=4;
                              ads_printf("\n->ZAmpPunch>>REC
horizontal&vertical punch at corner.");
```

```
                                    // rec punch up or down and left or
right
                                    zObj.mcLength1=fabs(zObj.fPt0.x-
zObj.fPt1.x);
                                    zObj.pitchX=zObj.toolDimx-1.0;
                                    zObj.mcLength2=fabs(zObj.fPt2.y-
zObj.fPt1.y);
                                    zObj.pitchY=zObj.toolDimy-1.0;
                                    if
((zObj.cfStyle==1)||(zObj.cfStyle==2))
                                    {
                                          zObj.fRefPt.x=zObj.fPt0.x;
                                          zObj.fRefPt.y=zObj.fPt0.y;

                                          zObj.fRefPtSec.x=zObj.fPt1.x-
zObj.toolDimx;

                                          strcpy(zObj.recDirectionUD,"U");

                                          if (zObj.cfStyle==1)
                                          {

      zObj.fRefPtSec.x=zObj.fPt1.x-zObj.toolDimx;

      zObj.fRefPtSec.y=zObj.fPt1.y;


      strcpy(zObj.recDirectionLR,"R");
                                          }
                                          else if (zObj.cfStyle==2)
                                          {

      zObj.fRefPtSec.x=zObj.fPt1.x;

      zObj.fRefPtSec.y=zObj.fPt1.y;


      strcpy(zObj.recDirectionLR,"L");
                                          }
                                    }
                                    else                                    if
((zObj.cfStyle==3)||(zObj.cfStyle==4))
                                    {
                                          zObj.fRefPt.x=zObj.fPt0.x;
                                          zObj.fRefPt.y=zObj.fPt0.y-
zObj.toolDimy;
                                          strcpy(zObj.recDirectionUD,"D");

                                          if (zObj.cfStyle==3)
                                          {

      zObj.fRefPtSec.x=zObj.fPt1.x-zObj.toolDimx;

      zObj.fRefPtSec.y=zObj.fPt1.y;


      strcpy(zObj.recDirectionLR,"R");
                                          }
                                          else if (zObj.cfStyle==4)
                                          {
```

```
      zObj.fRefPtSec.x=zObj.fPt1.x;

      zObj.fRefPtSec.y=zObj.fPt1.y;


      strcpy(zObj.recDirectionLR,"L");
                                    }
                              }
                        }

                  }
            };
            break;
      case 6:
            // eType = 6 means polyline oblong
            {
                  zObj.recCenPt1.x=(zObj.recPt1.x+zObj.recPt0.x)/2;
                  zObj.recCenPt1.y=(zObj.recPt3.y+zObj.recPt0.y)/2;
                  ads_printf("\n->ZAmpPunch>>Oblong Reference pt. is
(%0.2f,%0.2f)",zObj.recCenPt1.x,zObj.recCenPt1.y);
            };
            break;
      }
      return;
}

void ZWriteG(ZPunch &zObj)
{
      stptr = fopen(zObj.nameFile->resval.rstring,"a");
      ads_printf("\n->ZAmpPunch>>...generating                    g-
code.....................");
      switch (zObj.eType)
      {
      case 1:
            // eType = 1 means Circle
            {
                  if (strcmp(zObj.kwTypeCir,"1")==0)
                  {

      fprintf(stptr,"\nX%0.2fY%0.2fT%s",zObj.cCenter.x,zObj.cCenter.y
,zObj.kwTool);
                  }
                  else if (strcmp(zObj.kwTypeCir,"2")==0)
                  {

      fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",zObj.cCenter.x,zObj.cCent
er.y,zObj.kwTool);
                        fprintf(stptr,"\nNBL/");
                        fprintf(stptr,"\nHOL/%0.2f              %0.2f
2.0",(zObj.cRadius * 2),zObj.toolDia);
                  }
                  else if (strcmp(zObj.kwTypeCir,"3")==0)
                  {

      fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",zObj.cCenter.x,zObj.cCent
er.y,zObj.kwTool);
                        fprintf(stptr,"\nNBL/");
                        fprintf(stptr,"\nOPN/%0.2f              %0.2f
2.0",(zObj.cRadius * 2),zObj.toolDia);
                  }
```

```
            };
            break;
      case 2:
            // eType = 2 means Rectangle
            {
                  if (strcmp(zObj.kwTypeRec,"1")==0)
                  {
                        fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.recRfPt.x, zObj.recRfPt.y, zObj.kwTool);
                        fprintf(stptr,"\nREC/%s   %0.2f   %0.2f   %0.2f
%0.2f",
                              zObj.recDirectionLR,zObj.mcLength1,
zObj.toolDimx, zObj.pitchX, zObj.toolDimy);
                  }
                  else if (strcmp(zObj.kwTypeRec,"2")==0)
                  {
                        fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.recRfPt.x, zObj.recRfPt.y, zObj.kwTool);
                        fprintf(stptr,"\nREC/%s   %0.2f   %0.2f   %0.2f
%0.2f",
                              zObj.recDirectionUD,zObj.mcLength1,
zObj.toolDimx, zObj.pitchY, zObj.toolDimy);
                  }
                  else if (strcmp(zObj.kwTypeRec,"3")==0)
                  {
                        fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.recRfPt.x, zObj.recRfPt.y, zObj.kwTool);
                        switch (zObj.refAt) // ref at 4 choices
                        {
                        case 0:
                              {
                                    fprintf(stptr,"\nREC/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionLR,
zObj.mcLength1, zObj.toolDimx, zObj.pitchX,
                                          zObj.recDirectionUD,
zObj.mcLength2, zObj.toolDimy, zObj.pitchY);
                              };
                              break;
                        case 1:
                              {
                                    fprintf(stptr,"\nREC/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionUD,
zObj.mcLength1, zObj.toolDimy, zObj.pitchY,
                                          zObj.recDirectionLR,
zObj.mcLength2, zObj.toolDimx, zObj.pitchX);
                              };
                              break;
                        case 2:
                              {
                                    fprintf(stptr,"\nREC/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionLR,
zObj.mcLength1, zObj.toolDimx, zObj.pitchX,
                                          zObj.recDirectionUD,
zObj.mcLength2, zObj.toolDimy, zObj.pitchY);
                              };
                              break;
                        case 3:
                              {
```

```
                                    fprintf(stptr,"\nREC/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionUD,
zObj.mcLength1, zObj.toolDimy, zObj.pitchY,
                                          zObj.recDirectionLR,
zObj.mcLength2, zObj.toolDimx, zObj.pitchX);
                              };
                              break;
                        }
                        //return;
                  }
                  else if (strcmp(zObj.kwTypeRec,"4")==0)
                  {
                        fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.recRfPt.x, zObj.recRfPt.y, zObj.kwTool);
                        switch (zObj.refAt) // ref at 4 choices
                        {
                        case 0:
                              {
                                    fprintf(stptr,"\nOBL/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionLR,
zObj.mcLength1, zObj.toolDimx, zObj.pitchX,
                                          zObj.recDirectionUD,
zObj.mcLength2, zObj.toolDimy, zObj.pitchY);
                              };
                              break;
                        case 1:
                              {
                                    fprintf(stptr,"\nOBL/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionUD,
zObj.mcLength1, zObj.toolDimy, zObj.pitchY,
                                          zObj.recDirectionLR,
zObj.mcLength2, zObj.toolDimx, zObj.pitchX);
                              };
                              break;
                        case 2:
                              {
                                    fprintf(stptr,"\nOBL/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionLR,
zObj.mcLength1, zObj.toolDimx, zObj.pitchX,
                                          zObj.recDirectionUD,
zObj.mcLength2, zObj.toolDimy, zObj.pitchY);
                              };
                              break;
                        case 3:
                              {
                                    fprintf(stptr,"\nOBL/%s     %0.2f
%0.2f %0.2f %s %0.2f %0.2f %0.2f",
                                          zObj.recDirectionUD,
zObj.mcLength1, zObj.toolDimy, zObj.pitchY,
                                          zObj.recDirectionLR,
zObj.mcLength2, zObj.toolDimx, zObj.pitchX);
                              };
                              break;
                        }
                  }
                  else if (strcmp(zObj.kwTypeRec,"5")==0)
                  {
```

```
                        fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.recRfPt.x, zObj.recRfPt.y, zObj.kwTool);
                        fprintf(stptr,"\nRRC/%0.2f %0.2f %0.2f %0.2f
%0.2f %0.2f %0.2f",
                              zObj.mcLength1,          zObj.toolDimx,
zObj.pitchX,
                              zObj.mcLength2,          zObj.toolDimy,
zObj.pitchY,
                              zObj.cornRad);
                  }
                  else if (strcmp(zObj.kwTypeRec,"6")==0)
                  {

      fprintf(stptr,"\nX%0.2fY%0.2fT%s",zObj.recCenPt1.x,zObj.recCenP
t1.y,zObj.kwTool);
                  }
            };
            break;
      case 3:
            {
                  fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.arcCenter.x, zObj.arcCenter.y, zObj.kwTool);
                  fprintf(stptr,"\nNBL/");
                  fprintf(stptr,"\nRAD/%s  %0.2f  %0.2f  %0.2f  %0.2f
%0.2f",
                        zObj.arcInOut, zObj.aRad, zObj.toolDia,
                        zObj.mcStartAng,              zObj.mcTravAng,
zObj.aPitch);
            };
            break;
      case 4:
            {
                  if (zObj.horVer==0)// vertical
                  {
                        fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.lineRefPt.x, zObj.lineRefPt.y, zObj.kwTool);
                        fprintf(stptr,"\nREC/%s  %0.2f  %0.2f  %0.2f
%0.2f",
                              zObj.recDirectionUD, zObj.mcLength1,
                              zObj.toolDimy,             zObj.pitchY,
zObj.toolDimx);
                  }
                  else if (zObj.horVer==1)// horizontal
                  {
                        fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.lineRefPt.x, zObj.lineRefPt.y, zObj.kwTool);
                        fprintf(stptr,"\nREC/%s  %0.2f  %0.2f  %0.2f
%0.2f",
                              zObj.recDirectionLR, zObj.mcLength1,
                              zObj.toolDimx,             zObj.pitchX,
zObj.toolDimy);
                  }
            };
            break;
      case 5:
            {
                  if (zObj.triOrCorn==0) // triangle
                  {

      fprintf(stptr,"\nX%0.2fY%0.2fT%s",zObj.recCenPt1.x,zObj.recCenP
t1.y,zObj.kwTool);
```

```
                }
                else if(zObj.triOrCorn==1) // corner fillet
                {
                        if (zObj.cfType==1)
                        {

      fprintf(stptr,"\nX%0.2fY%0.2fT%s",zObj.recCenPt1.x,zObj.recCenP
t1.y,zObj.kwTool);
                        }
                        else if (zObj.cfType==2)
                        {
                                fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.fRefPt.x, zObj.fRefPt.y, zObj.kwTool);
                                fprintf(stptr,"\nREC/%s   %0.2f   %0.2f
%0.2f %0.2f",

      zObj.recDirectionUD,zObj.mcLength1,

      zObj.toolDimy,zObj.pitchY,zObj.toolDimx);
                        }
                        else if (zObj.cfType==3)
                        {
                                fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.fRefPt.x, zObj.fRefPt.y, zObj.kwTool);
                                fprintf(stptr,"\nREC/%s   %0.2f   %0.2f
%0.2f %0.2f",

      zObj.recDirectionLR,zObj.mcLength1,

      zObj.toolDimx,zObj.pitchX,zObj.toolDimy);
                        }
                        else if (zObj.cfType==4)
                        {
                                fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.fRefPt.x, zObj.fRefPt.y, zObj.kwTool);
                                fprintf(stptr,"\nREC/%s   %0.2f   %0.2f
%0.2f %0.2f",

      zObj.recDirectionLR,zObj.mcLength1,

      zObj.toolDimx,zObj.pitchX,zObj.toolDimy);
                                fprintf(stptr,"\nMOV/X%0.2fY%0.2fT%s",
zObj.fRefPtSec.x, zObj.fRefPtSec.y, zObj.kwTool);
                                fprintf(stptr,"\nREC/%s   %0.2f   %0.2f
%0.2f %0.2f",

      zObj.recDirectionUD,zObj.mcLength2,

      zObj.toolDimy,zObj.pitchY,zObj.toolDimx);
                        }
                };
                break;
      case 6:
                {

      fprintf(stptr,"\nX%0.2fY%0.2fT%s",zObj.recCenPt1.x,zObj.recCenP
t1.y,zObj.kwTool);
                };
                break;
      }
```

```
	fclose (stptr);
	return;
}

void ZLoopAsk(ZPunch &zObj)
{
	int rc;

	ads_printf("\n->ZAmpPunch>>Do you want to ... ");
	ads_initget(0,"Continue Changetool Exit");
	rc	=	ads_getkword("(<Continue>/Changetool/Exit)?	:",
zObj.loopKw);

	if ((rc != RTNORM) && (rc != RTNONE))
	{
		ads_printf("\n->ZAmpPunch>>Nothing selected.");
	}

	else if ((rc == RTNONE) || (strcmp(zObj.loopKw,"Continue")) ==
0)
	{
		strcpy(zObj.loopKw,"Continue");
		return;
	}
	else if (strcmp(zObj.loopKw,"Changetool") == 0)
	{
		strcpy(zObj.loopKw,"Changetool");
		return;
	}
	else if (strcmp(zObj.loopKw,"Exit") == 0)
	{
		strcpy(zObj.loopKw,"Exit");
		ads_printf("\n->ZAmpPunch>>End of ZAmpPunch program.");
		return;
	}
	return;
}

void ZInAskToolDia(ZPunch &zObj)
{
	int res;
	ads_printf("\n->ZAmpPunch>>Enter diameter of the tool");
	ads_initget(1,"");
	res = ads_getreal("(1.4 to 60.0) :",&zObj.toolDia);
	if (res!=RTNORM)
	{
		ads_printf("\n->ZAmpPunch>>Error");
		return;
	}
	while ((zObj.toolDia < 1.4)||(zObj.toolDia > 60.0))
	{
		ads_printf("\n->ZAmpPunch>>Invalid input");
		ads_printf("\n->ZAmpPunch>>Enter diameter of the tool");
		ads_initget(1,"");
		res = ads_getreal("(1.4 to 60.0) :",&zObj.toolDia);
		if (res!=RTNORM)
		{
			ads_printf("\n->ZAmpPunch>>Error");
			return;
		}
```

```
	}
	ads_printf("\n->ZAmpPunch>>Tool diameter is %0.2f
mm.",zObj.toolDia);
	return;
}

void ZAskToolDim(ZPunch &zObj)
{
	int res;
	ads_printf("\n->ZAmpPunch>>Enter X-axis size of the tool");
	zObj.toolDimx = 10.0;
	ads_initget(1,"");
	res = ads_getreal("(2.0 to 70.0) :",&zObj.toolDimx);
	if (res!=RTNORM)
	{
		ads_printf("\n->ZAmpPunch>>Error");
		return;
	}

	while ((zObj.toolDimx < 2.0)||(zObj.toolDimx > 70.0))
	{
		ads_printf("\n->ZAmpPunch>>Invalid input");
		ads_printf("\n->ZAmpPunch>>Enter X-axis size of the
tool");
		ads_initget(1,"");
		res = ads_getreal("(2.0 to 70.0) :",&zObj.toolDimx);
		if (res!=RTNORM)
		{
			ads_printf("\n->ZAmpPunch>>Error");
			return;
		}
	}

	ads_printf("\n->ZAmpPunch>>Enter Y-axis size of the tool");
	zObj.toolDimy = 10.0;
	ads_initget(1,"");
	res = ads_getreal("(2.0 to 70.0) :",&zObj.toolDimy);
	if (res!=RTNORM)
	{
		ads_printf("\n->ZAmpPunch>>Error");
		return;
	}

	while ((zObj.toolDimy < 2.0)||(zObj.toolDimy > 70.0))
	{
		ads_printf("\n->ZAmpPunch>>Invalid input");
		ads_printf("\n->ZAmpPunch>>Enter Y-axis size of the
tool");
		ads_initget(1,"");
		res = ads_getreal("(2.0 to 70.0) :",&zObj.toolDimy);
		if (res!=RTNORM)
		{
			ads_printf("\n->ZAmpPunch>>Error");
			return;
		}
	}

	ads_printf("\n->ZAmpPunch>>Tool Dimension is %0.2fx%0.2f mm.x
mm.", zObj.toolDimx, zObj.toolDimy);
	return;
}
```

```
void ZInGetTypeRec(ZPunch &zObj)
{
      int rc;

      ads_printf("\n->ZAmpPunch>>Enter rectangle punching type");
      ads_initget(0,"1 2 3 4 5 6");
      rc = ads_getkword("(<1>/2/3/4/5/6) :",zObj.kwTypeRec);

      if ((rc != RTNORM) && (rc != RTNONE))
      {
            ads_printf("\n->ZAmpPunch>>Nothing selected.");
    }

      if ((rc == RTNONE) || (strcmp(zObj.kwTypeRec, "1")) == 0)
      {
            strcpy(zObj.kwTypeRec,"1");
            ads_printf("\n->ZAmpPunch>>You   have   chosen   rectangle
punching type %s",zObj.kwTypeRec);
            return;
      }
      else if ((strcmp(zObj.kwTypeRec, "2"))||(strcmp(zObj.kwTypeRec,
"3")
            ||(strcmp(zObj.kwTypeRec,  "4"))||(strcmp(zObj.kwTypeRec,
"5"))||(strcmp(zObj.kwTypeRec, "6")) == 0))
      {
            ads_printf("\n->ZAmpPunch>>You   have   chosen   rectangle
punching type %s",zObj.kwTypeRec);
            return;
      }
      return;
}

void  ZInGetRecRef(ZPunch  &zObj)  //  get  reference  pt  for  every
kwTypeRec
{
      int res;
      ads_point dummy;

      if (strcmp(zObj.kwTypeRec,"1")==0)
      {
            ads_printf("\n->ZAmpPunch>>Pick reference point for this
rectangle");

            ads_initget(41,"");
            res = ads_getpoint(NULL," :",dummy);
            if (res!=RTNORM)
            {
                  ads_printf("\n->ZAmpPunch>>Error");
                  return;
            }
            // Convert ads_point to AcGePoint3d
            zObj.recRfPt = asPnt3d(dummy);

            while
((zObj.recRfPt!=zObj.recPt0)&&(zObj.recRfPt!=zObj.recPt1))
            {
                  ads_printf("\n->ZAmpPunch>>Invalid        reference
point");
                  ads_printf("\n->ZAmpPunch>>Pick reference point for
this rectangle");
```

```
                  ads_initget(41,"");
                  res = ads_getpoint(NULL," :",dummy);
                  if (res!=RTNORM)
                  {
                        ads_printf("\n->ZAmpPunch>>Error");
                        return;
                  }
                  zObj.recRfPt = asPnt3d(dummy);
            }
      }

      else if (strcmp(zObj.kwTypeRec,"2")==0)
      {
            ads_printf("\n->ZAmpPunch>>Pick reference point for this
rectangle");

            ads_initget(41,"");
            res = ads_getpoint(NULL," :",dummy);
            if (res!=RTNORM)
            {
                  ads_printf("\n->ZAmpPunch>>Error");
                  return;
            }
            // Convert ads_point to AcGePoint3d
            zObj.recRfPt = asPnt3d(dummy);

            while
((zObj.recRfPt!=zObj.recPt0)&&(zObj.recRfPt!=zObj.recPt3))
            {
                  ads_printf("\n->ZAmpPunch>>Invalid        reference
point");
                  ads_printf("\n->ZAmpPunch>>Pick reference point for
this rectangle");
                  ads_initget(41,"");
                  res = ads_getpoint(NULL," :",dummy);
                  if (res!=RTNORM)
                  {
                        ads_printf("\n->ZAmpPunch>>Error");
                        return;
                  }
                  zObj.recRfPt = asPnt3d(dummy);
            }
      }

      else if (strcmp(zObj.kwTypeRec,"3")==0)
      {
            ads_printf("\n->ZAmpPunch>>Pick reference point for this
rectangle");

            ads_initget(41,"");
            res = ads_getpoint(NULL," :",dummy);
            if (res!=RTNORM)
            {
                  ads_printf("\n->ZAmpPunch>>Error");
                  return;
            }
            // Convert ads_point to AcGePoint3d
            zObj.recRfPt = asPnt3d(dummy);

            while
((zObj.recRfPt!=zObj.recPt0)&&(zObj.recRfPt!=zObj.recPt1)
```

```
      &&(zObj.recRfPt!=zObj.recPt2)&&(zObj.recRfPt!=zObj.recPt3))
            {
                  ads_printf("\n->ZAmpPunch>>Invalid        reference
point");
                  ads_printf("\n->ZAmpPunch>>Pick reference point for
this rectangle");
                  ads_initget(41,"");
                  res = ads_getpoint(NULL," :",dummy);
                  if (res!=RTNORM)
                  {
                        ads_printf("\n->ZAmpPunch>>Error");
                        return;
                  }

                  zObj.recRfPt = asPnt3d(dummy);
            }
      }

      else if (strcmp(zObj.kwTypeRec,"4")==0)
      {
            ads_printf("\n->ZAmpPunch>>Pick reference point for this
rectangle");

            ads_initget(41,"");
            res = ads_getpoint(NULL," :",dummy);
            if (res!=RTNORM)
            {
                  ads_printf("\n->ZAmpPunch>>Error");
                  return;
            }
            // Convert ads_point to AcGePoint3d
            zObj.recRfPt = asPnt3d(dummy);

            while
((zObj.recRfPt!=zObj.recPt0)&&(zObj.recRfPt!=zObj.recPt1)

      &&(zObj.recRfPt!=zObj.recPt2)&&(zObj.recRfPt!=zObj.recPt3))
            {
                  ads_printf("\n->ZAmpPunch>>Invalid        reference
point");
                  ads_printf("\n->ZAmpPunch>>Pick reference point for
this rectangle");
                  ads_initget(41,"");
                  res = ads_getpoint(NULL," :",dummy);
                  if (res!=RTNORM)
                  {
                        ads_printf("\n->ZAmpPunch>>Error");
                        return;
                  }
                  zObj.recRfPt = asPnt3d(dummy);
            }
      }
      else if (strcmp(zObj.kwTypeRec,"5")==0)
      {
            zObj.recRfPt.x=(zObj.recPt0.x+zObj.recPt1.x)/2;
            zObj.recRfPt.y=(zObj.recPt0.y+zObj.recPt3.y)/2;
      }
      else if (strcmp(zObj.kwTypeRec,"6")==0)
      {
            zObj.recRfPt.x=(zObj.recPt0.x+zObj.recPt1.x)/2;
```

```
            zObj.recRfPt.y=(zObj.recPt0.y+zObj.recPt3.y)/2;
      }

      ads_printf("\n->ZAmpPunch>>Rectangle    Reference    pt.    is
(%0.2f,%0.2f)",zObj.recRfPt.x,zObj.recRfPt.y);
      return;
}

void ZInAskCornRad(ZPunch &zObj)
{
      int res;
      ads_printf("\n->ZAmpPunch>>Enter corner radius");
      zObj.cornRad = 3.0;
      ads_initget(1,"");
      res = ads_getreal("(1.4 to 60.0) :",&zObj.cornRad);
      if (res!=RTNORM)
      {
            ads_printf("\n->ZAmpPunch>>Error");
            return;
      }

      while ((zObj.cornRad < 1.4)||(zObj.cornRad > 60.0))
      {
            ads_printf("\n->ZAmpPunch>>Invalid input");
            ads_printf("\n->ZAmpPunch>>Enter corner radius");
            ads_initget(1,"");
            res = ads_getreal("(1.4 to 60.0) :",&zObj.cornRad);
            if (res!=RTNORM)
            {
                  ads_printf("\n->ZAmpPunch>>Error");
                  return;
            }
      }

      ads_printf("\n->ZAmpPunch>>Corner   radius   is   %0.2f   mm.",
zObj.cornRad);
      return;
}

void ZAskArcInOut(ZPunch &zObj)
{
      int rc;
      ads_printf("\n->ZAmpPunch>>Enter side of punching this arc");
      ads_initget(0,"In Out");
      rc = ads_getkword("(<In>/Out) :",zObj.arcInOut);

      if ((rc != RTNORM) && (rc != RTNONE))
      {
            ads_printf("\n->ZAmpPunch>>Nothing selected.");
      }

      if ((rc == RTNONE) || (strcmp(zObj.arcInOut, "In")) == 0)
      {
            strcpy(zObj.arcInOut,"I");
            ads_printf("\n->ZAmpPunch>>You  have  chosen  inside  arc
punching");
            return;
      }
      else if (strcmp(zObj.arcInOut, "Out") == 0)
      {
            strcpy(zObj.arcInOut,"O");
```

```
            ads_printf("\n->ZAmpPunch>>You have chosen outside arc
punching");
            return;
      }

      return;
}

////////////////////////////////////////////////////////////////////
//////////////////

// OUTSIDE PUNCH        OUTSIDE PUNCH           OUTSIDE PUNCH

////////////////////////////////////////////////////////////////////
//////////////////
void OutPunch()
{
      ZPunch zObj;

      ZOutMain(zObj);
      return;
}

void ZOutMain(ZPunch &zObj)
{
      int rc;
      char kw[20];

      ads_printf("\n->ZAmpPunch>>Do you want to proceed outside-
punching...");
      ads_initget(0,"Yes No");
      rc = ads_getkword("(<Yes>/No)? :",kw);

      if ((rc != RTNORM) && (rc != RTNONE))
      {
            ads_printf("\n->ZAmpPunch>>Error.");
      }

      if (strcmp(kw, "No") == 0)
      {
            ads_printf("\n->ZAmpPunch>>User canceled operation.");
            return;
      }

      if ((rc == RTNONE) || (strcmp(kw, "Yes")) == 0)
      {
            ads_printf("\n->ZAmpPunch>>Creating dat file for storing
g-code...");
            ZOutCreateF(zObj); // create file and write a header on
it
            ads_printf("\n->ZAmpPunch>>Start generating g-code from
the drawing");

            ////////////////////
            //  Start of MAIN LOOPs
            ////////////////////

            strcpy(zObj.loopKw,"tk");
            while (strcmp(zObj.loopKw,"Exit")!=0)
            {
                  ZGetTool(zObj);
```

```
                  strcpy(zObj.loopKw,"Continue");
                  while     ((strcmp(zObj.loopKw,"Changetool")     &&
strcmp(zObj.loopKw,"Exit")) != 0)
                  {
                        ZOutGetObject(zObj);
                        ZKB(zObj);
                        ZWriteG(zObj);
                        ads_printf("\n->ZAmpPunch>>End  of  generating
G-code for this object.");
                        ZLoopAsk(zObj);
                  }
            }

            // End of MAIN LOOPs
            stptr = fopen(zObj.nameFile->resval.rstring,"a");
            fprintf(stptr,"\n/End of outside-punching file.");
            fclose (stptr);
      }
      return;
}

void ZOutCreateF(ZPunch &zObj)
{
      int res;
      zObj.nameFile = ads_newrb(RTSTR);
      res  =  ads_getfiled  ("Please  type  new  outside-punching  file
name... ",  // title
                                          "C:\\",  // no suggestion
                                          "dat",   // extension
                                          NEWFILE, // flags
                                          zObj.nameFile)  ;        //
where to put result
      if (res == RTNORM)
      {
      // Check which type of result buffer has been returned
            if (zObj.nameFile->restype == RTSTR)
            {
                  ads_printf   ("\n->ZAmpPunch>>The   user   selected
<%s>",
                                                zObj.nameFile-
>resval.rstring) ;
            }
            else if (zObj.nameFile->restype == RTSHORT)
            {
                  ads_printf ("\n->ZAmpPunch>>Request to type at the"
                                                "       command
line.") ;
            }
            else
            {
                  ads_printf     ("\n->ZAmpPunch>>Strange     return
type:%d",

zObj.nameFile->restype) ;
            }
      }
      else
      {
            ads_printf ("\n->ZAmpPunch>>User abandoned dialog box") ;
            return;
      }
```

```
      if((stptr = fopen(zObj.nameFile->resval.rstring,"w"))!=NULL)
      {
            fprintf (stptr,"/Start Outside G-Code");
            fclose (stptr);
      }
      return;
}

void ZOutGetObject(ZPunch &zObj)
{
    ads_name eName;
      ads_point pt;
      AcDbObjectId eId;
      AcDbEntity* ePtr;
      AcDbPolyline* plPtr;
      AcDbLine* lPtr;
      int rc;

      //RECEIVE ENTITY TO KEEP AT name "eName"
      rc = ads_entsel("\n->ZAmpPunch>>Select object: ", eName, pt);

      if (rc != RTNORM)
      {
        ads_printf("\n->ZAmpPunch>>Nothing selected.\n", rc);
            return;
    }

      //RECEIVE Id OF ENTITY FROM eName
    Acad::ErrorStatus es = acdbGetObjectId(eId, eName);

      if (es != Acad::eOk)
      {
            ads_printf("\n->ZAmpPunch>>Error getting object id.");
            return;
      }

      //RECEIVE Oject OF ENTITY FROM eId
    es = acdbOpenObject(ePtr, eId, AcDb::kForWrite); // for write
'cos change color

      if (es != Acad::eOk)
      {
            ads_printf("\n->ZAmpPunch>>Error opening object for
write.");
            return; // for exiting if clause when this function
return void
      }

      ////////////////////////////////////////////////////////
      // Start checking what entity is and keep its         //
      // attributes in the global variables defined in zglobal.h//
      ////////////////////////////////////////////////////////

      ////////////////
      // Line       //
      ////////////////

      if (AcDbLine::cast(ePtr)!=NULL)
      {
            AcGePoint3d dummy;
```

```
            ads_printf("\n->ZAmpPunch>>Object is Line.");

            lPtr = AcDbLine::cast(ePtr);
            // change color of object
            ads_printf("\n->ZAmpPunch>>Changing color..");
            lPtr->setColorIndex(4);

            zObj.lStartPt = lPtr->startPoint();
            zObj.lEndPt = lPtr->endPoint();

            // check if "vertical or horizontal" and "Start or End"
and then convert it
            if (zObj.lStartPt.x == zObj.lEndPt.x) // vertical line
            {
                  zObj.horVer = 0; // 0 mean vertical

                  if (zObj.lStartPt.y > zObj.lEndPt.y) // start pt.
is at top
                  {
                        zObj.lStartPt = zObj.lStartPt;
                        zObj.lEndPt = zObj.lEndPt;
                  }
                  else if (zObj.lStartPt.y < zObj.lEndPt.y) // start
pt is at bottom
                  {
                        dummy = zObj.lStartPt;
                        zObj.lStartPt = zObj.lEndPt;
                        zObj.lEndPt = dummy;
                  }
            }
            else if (zObj.lStartPt.y == zObj.lEndPt.y) // horizontal
line
            {
                  zObj.horVer = 1;

                  if (zObj.lStartPt.x < zObj.lEndPt.x) // start pt.
is at left
                  {
                        zObj.lStartPt = zObj.lStartPt;
                        zObj.lEndPt = zObj.lEndPt;
                  }
                  else if (zObj.lStartPt.x > zObj.lEndPt.x) // start
pt is at right
                  {
                        dummy = zObj.lStartPt;
                        zObj.lStartPt = zObj.lEndPt;
                        zObj.lEndPt = dummy;
                  }
            }
            else
            {
                  ads_printf("\n->ZAmpPunch>>Cannot   do   with   this
line.");
                  return;
            }

            lPtr->close();
            ePtr->close();

            zObj.eType = 4;
```

```
            return;

      }

      /////////////////
      // Polyline    //
      /////////////////

      else if (AcDbPolyline::cast(ePtr)!=NULL)
      {
            AcGePoint3d fdummy;
            double bulge0,bulge1,bulge2;
            unsigned int numVer;

            ads_printf("\n->ZAmpPunch>>This Object is Polyline.");
            plPtr = AcDbPolyline::cast(ePtr);

            // start checking what type PLINE is by checking bulge
and vertex
            if ((plPtr->isClosed())==Adesk::kTrue)
            {
                  ads_printf("\n->ZAmpPunch>>Entity is closed, cannot
do this for outside punching.");
                  plPtr->close();
                  ePtr->close();
                  return;
            }

            // change color of object
            ads_printf("\n->ZAmpPunch>>Changing color..");
            plPtr->setColorIndex(4);

            // check bulge
            plPtr->getBulgeAt(0,bulge0);
            plPtr->getBulgeAt(1,bulge1);
            plPtr->getBulgeAt(2,bulge2);
            if ( (bulge0!=0)||(bulge1!=0)||(bulge2!=0) )
            {
                  ads_printf("\n->ZAmpPunch>>Cannot do this for this
polyline.");
                  return;
            }

            numVer = plPtr->numVerts();
            ads_printf("\n->ZAmpPunch>>Number of vertex in pline is
%d.",numVer);

            if (numVer==3) // corner fillet and triangle fillet
            {
                  zObj.eType = 5;
                  plPtr->getPointAt(0,zObj.fPt0);
                  plPtr->getPointAt(1,zObj.fPt1);
                  plPtr->getPointAt(2,zObj.fPt2);

                  // check if corner or triangle
                  if
(((zObj.fPt0.x!=zObj.fPt1.x)&&(zObj.fPt2.x!=zObj.fPt1.x))||

      ((zObj.fPt0.y!=zObj.fPt1.y)&&(zObj.fPt2.y!=zObj.fPt1.y)))
                  {
                        ////////////////////
```

```
                        // PLINE - triangle fillet
                        ////////////////////
                        AcGePoint3d tridummy;
                        zObj.triOrCorn = 0;
                        ads_printf("\n->ZAmpPunch>>This   object   is
polyline at edge.");
                        // check style
                        if(zObj.fPt0.x==zObj.fPt2.x)
                        {
                              if(zObj.fPt1.x>zObj.fPt0.x)
                              {
                                    zObj.triStyle=1;
                                    // convert to std.
                                    if(zObj.fPt0.y>zObj.fPt2.y)
                                    {
                                          zObj.fPt0=zObj.fPt0;
                                          zObj.fPt1=zObj.fPt1;
                                          zObj.fPt2=zObj.fPt2;
                                    }
                                    else
                                    {
                                          tridummy=zObj.fPt0;
                                          zObj.fPt0=zObj.fPt2;
                                          zObj.fPt2=tridummy;
                                    }
                              }
                              else if (zObj.fPt1.x<zObj.fPt0.x)
                              {
                                    zObj.triStyle=2;
                                    // convert to std.
                                    if(zObj.fPt0.y>zObj.fPt2.y)
                                    {
                                          zObj.fPt0=zObj.fPt0;
                                          zObj.fPt1=zObj.fPt1;
                                          zObj.fPt2=zObj.fPt2;
                                    }
                                    else
                                    {
                                          tridummy=zObj.fPt0;
                                          zObj.fPt0=zObj.fPt2;
                                          zObj.fPt2=tridummy;
                                    }
                              }
                        }
                        else if(zObj.fPt0.y==zObj.fPt2.y)
                        {
                              if(zObj.fPt1.y>zObj.fPt0.y)
                              {
                                    zObj.triStyle=3;
                                    // convert to std.
                                    if(zObj.fPt0.x<zObj.fPt2.x)
                                    {
                                          zObj.fPt0=zObj.fPt0;
                                          zObj.fPt1=zObj.fPt1;
                                          zObj.fPt2=zObj.fPt2;
                                    }
                                    else if(zObj.fPt0.x>zObj.fPt2.x)
                                    {
                                          tridummy=zObj.fPt0;
                                          zObj.fPt0=zObj.fPt2;
                                          zObj.fPt2=tridummy;
```

```
                              }
                        }
                        else if(zObj.fPt1.y<zObj.fPt0.y)
                        {
                              zObj.triStyle=4;
                              // convert to std.
                              if(zObj.fPt0.x<zObj.fPt2.x)
                              {
                                    zObj.fPt0=zObj.fPt0;
                                    zObj.fPt1=zObj.fPt1;
                                    zObj.fPt2=zObj.fPt2;
                              }
                              else if(zObj.fPt0.x>zObj.fPt2.x)
                              {
                                    tridummy=zObj.fPt0;
                                    zObj.fPt0=zObj.fPt2;
                                    zObj.fPt2=tridummy;
                              }
                        }
                  }
            }
            else
            {
                  //////////////////////
                  // PLINE -corner fillet
                  //////////////////////
                  zObj.triOrCorn = 1;
                  ads_printf("\n->ZAmpPunch>>This   object   is
polyline at corner.");
                  // check style of corner fillet
                  if
(((zObj.fPt0.x<zObj.fPt1.x)&&(zObj.fPt2.x==zObj.fPt1.x)&&(zObj.fPt0.y
==zObj.fPt1.y)&&(zObj.fPt2.y>zObj.fPt1.y))||

      ((zObj.fPt2.x<zObj.fPt1.x)&&(zObj.fPt0.x==zObj.fPt1.x)&&(zObj.f
Pt2.y==zObj.fPt1.y)&&(zObj.fPt0.y>zObj.fPt1.y)))
                  {
                        // Style1
                        zObj.cfStyle=1;
                        // make all pt in standard

      if((zObj.fPt0.x<zObj.fPt2.x)&&(zObj.fPt0.y<zObj.fPt2.y))
                        {
                              zObj.fPt0=zObj.fPt0;
                              zObj.fPt2=zObj.fPt2;
                        }
                        else
                        {
                              fdummy=zObj.fPt0;
                              zObj.fPt0=zObj.fPt2;
                              zObj.fPt2=fdummy;
                        }
                  }
                  else                                          if
(((zObj.fPt0.x==zObj.fPt1.x)&&(zObj.fPt2.x>zObj.fPt1.x)&&(zObj.fPt0.y
>zObj.fPt1.y)&&(zObj.fPt2.y==zObj.fPt1.y))||

      ((zObj.fPt2.x==zObj.fPt1.x)&&(zObj.fPt0.x>zObj.fPt1.x)&&(zObj.f
Pt2.y>zObj.fPt1.y)&&(zObj.fPt0.y==zObj.fPt1.y)))
                  {
                        // Style2
```

```
                              zObj.cfStyle=2;
                              // make all pt in standard

      if((zObj.fPt0.x>zObj.fPt2.x)&&(zObj.fPt0.y<zObj.fPt2.y))
                              {
                                    zObj.fPt0=zObj.fPt0;
                                    zObj.fPt2=zObj.fPt2;
                              }
                              else
                              {
                                    fdummy=zObj.fPt0;
                                    zObj.fPt0=zObj.fPt2;
                                    zObj.fPt2=fdummy;
                              }
                        }
                        else
if(((zObj.fPt0.x<zObj.fPt1.x)&&(zObj.fPt2.x==zObj.fPt1.x)&&(zObj.fPt0
.y==zObj.fPt1.y)&&(zObj.fPt2.y<zObj.fPt1.y))||

      ((zObj.fPt2.x<zObj.fPt1.x)&&(zObj.fPt0.x==zObj.fPt1.x)&&(zObj.f
Pt2.y==zObj.fPt1.y)&&(zObj.fPt0.y<zObj.fPt1.y)))
                        {
                              // Style3
                              zObj.cfStyle=3;
                              // make all pt in standard

      if((zObj.fPt0.x<zObj.fPt2.x)&&(zObj.fPt0.y>zObj.fPt2.y))
                              {
                                    zObj.fPt0=zObj.fPt0;
                                    zObj.fPt2=zObj.fPt2;
                              }
                              else
                              {
                                    fdummy=zObj.fPt0;
                                    zObj.fPt0=zObj.fPt2;
                                    zObj.fPt2=fdummy;
                              }
                        }
                        else                                          if
(((zObj.fPt0.x==zObj.fPt1.x)&&(zObj.fPt2.x>zObj.fPt1.x)&&(zObj.fPt0.y
<zObj.fPt1.y)&&(zObj.fPt2.y==zObj.fPt1.y))||

      ((zObj.fPt2.x==zObj.fPt1.x)&&(zObj.fPt0.x>zObj.fPt1.x)&&(zObj.f
Pt2.y<zObj.fPt1.y)&&(zObj.fPt0.y==zObj.fPt1.y)))
                        {
                              // Style4
                              zObj.cfStyle=4;
                              // make all pt in standard

      if((zObj.fPt0.x>zObj.fPt2.x)&&(zObj.fPt0.y>zObj.fPt2.y))
                              {
                                    zObj.fPt0=zObj.fPt0;
                                    zObj.fPt2=zObj.fPt2;
                              }
                              else
                              {
                                    fdummy=zObj.fPt0;
                                    zObj.fPt0=zObj.fPt2;
                                    zObj.fPt2=fdummy;
                              }
                        }
```

```
                  }

            plPtr->close();
            ePtr->close();
            }
            else
            {
                  ads_printf("\n->ZAmpPunch>>Cannot   do   this   for
outside punching.");
                  plPtr->close();
                  ePtr->close();
                  return;
            }

      } // end of polyline
      else
      {
            ads_printf("\n->ZAmpPunch>>This   object   is   out   of
scope.");
      }
      return;
}

void ZOutGetLineRef(ZPunch &zObj)
{
      int res;
      ads_point dummy;

      ads_printf("\n->ZAmpPunch>>Pick  start-punching  point  for  this
line");
      ads_initget(41,"");
      res = ads_getpoint(NULL," :",dummy);
      if (res!=RTNORM)
      {
            ads_printf("\n->ZAmpPunch>>Error");
            return;
      }
      // Convert ads_point to AcGePoint3d
      zObj.lineRefPt = asPnt3d(dummy);

      while  ((zObj.lineRefPt  !=  zObj.lStartPt)&&(zObj.lineRefPt  !=
zObj.lEndPt))
      {
            ads_printf("\n->ZAmpPunch>>Invalid      start-punching
point");
            ads_printf("\n->ZAmpPunch>>Pick start-punching point for
this line");
            ads_initget(41,"");
            res = ads_getpoint(NULL," :",dummy);
            if (res!=RTNORM)
            {
                  ads_printf("\n->ZAmpPunch>>Error");
                  return;
            }
            // Convert ads_point to AcGePoint3d
            zObj.lineRefPt = asPnt3d(dummy);
      }

      ads_printf("\n->ZAmpPunch>>You have chosen start-punching point
at (%0.2f,%0.2f).",
            zObj.lineRefPt.x, zObj.lineRefPt.y);
```

```
	return;
}

void ZOutGetLineDirec(ZPunch &zObj)
{
	int rc;
	ads_printf("\n->ZAmpPunch>>Enter side of punching this line");

	if (zObj.horVer == 0) // vertical
	{
		ads_initget(0,"Left Right");
		rc = ads_getkword("(<Left>/Right) :",zObj.lineDirecLR);

		if ((rc != RTNORM) && (rc != RTNONE))
		{
			ads_printf("\n->ZAmpPunch>>Nothing selected.");
		}

		if ((rc == RTNONE) || (strcmp(zObj.lineDirecLR, "Left"))
== 0)
		{
			strcpy(zObj.lineDirecLR,"L");
			ads_printf("\n->ZAmpPunch>>You  have  chosen  Left
side.");
			return;
		}
		else if (strcmp(zObj.lineDirecLR, "Right") == 0)
		{
			strcpy(zObj.lineDirecLR,"R");
			ads_printf("\n->ZAmpPunch>>You  have  chosen  Right
side.");
			return;
		}
	}
	else if (zObj.horVer == 1) // horizontal
	{
		ads_initget(0,"Up Down");
		rc = ads_getkword("(<Up>/Down):",zObj.lineDirecUD);

		if ((rc != RTNORM) && (rc != RTNONE))
		{
			ads_printf("\n->ZAmpPunch>>Nothing selected.");
		}

		if ((rc == RTNONE) || (strcmp(zObj.lineDirecUD, "Up")) ==
0)
		{
			strcpy(zObj.lineDirecUD,"U");
			ads_printf("\n->ZAmpPunch>>You  have  chosen  Up
side.");
			return;
		}
		else if (strcmp(zObj.lineDirecUD, "Down") == 0)
		{
			strcpy(zObj.lineDirecUD,"D");
			ads_printf("\n->ZAmpPunch>>You  have  chosen  Down
side.");
			return;
		}
	}
	return;
```

```
}

void ZOutAskForJoint(ZPunch &zObj)
{
      int rc;
      ads_printf("\n->ZAmpPunch>>Do you want joint on this line...");
      ads_initget(0,"Yes No");
      rc = ads_getkword("(<Yes>/No)? :",zObj.jointYN);

      if ((rc != RTNORM) && (rc != RTNONE))
      {
            ads_printf("\n->ZAmpPunch>>Nothing selected.");
      }

      if ((rc == RTNONE) || (strcmp(zObj.jointYN, "Yes")) == 0)
      {
            strcpy(zObj.jointYN,"Yes");
            ads_printf("\n->ZAmpPunch>>You want a microjoint on this
line.");
            return;
      }
      else if (strcmp(zObj.jointYN, "No") == 0)
      {
            strcpy(zObj.jointYN,"No");
            ads_printf("\n->ZAmpPunch>>You don't want a microjoint on
this line.");
            return;
      }
      return;
}


void ZCreateJoint(ZPunch &zObj)
{
      struct resbuf* rb;

      rb = ads_newrb(RTSTR);
      // set system variable "PDMODE"
      rb->resval.rint = 35;

      if (ads_getvar("pdmode", rb)!= RTNORM)
      {
            ads_printf("\n->ZAmpPunch>>Error getting PDMODE.");
            return;
      }

      rb->resval.rint = 35;

      if (ads_setvar("pdmode", rb)!= RTNORM)
      {
            ads_printf("\n->ZAmpPunch>>Error setting PDMODE.");
            return;
      }

      // start create point (start) to model space
      AcDbPoint* pPointS = new AcDbPoint(zObj.lStartPt);


      AcDbBlockTable *pBlockTable;
    acdbCurDwg()->getBlockTable(pBlockTable,
        AcDb::kForRead);
```

```
    AcDbBlockTableRecord *pBlockTableRecord;
    pBlockTable->getAt(ACDB_MODEL_SPACE, pBlockTableRecord,
        AcDb::kForWrite);
    pBlockTable->close();

    AcDbObjectId pointIdS;

    pBlockTableRecord->appendAcDbEntity(pointIdS, pPointS);

    pBlockTableRecord->close();

      pPointS->setColorIndex(4);
      pPointS->close();


      // end of creating point(start) to model space

      // start create point (end) to model space
      AcDbPoint* pPointE = new AcDbPoint(zObj.lEndPt);


//    AcDbBlockTable *pBlockTable;
    acdbCurDwg()->getBlockTable(pBlockTable,
        AcDb::kForRead);

//    AcDbBlockTableRecord *pBlockTableRecord;
    pBlockTable->getAt(ACDB_MODEL_SPACE, pBlockTableRecord,
        AcDb::kForWrite);
    pBlockTable->close();

    AcDbObjectId pointIdE;

    pBlockTableRecord->appendAcDbEntity(pointIdE, pPointE);

    pBlockTableRecord->close();

      pPointE->setColorIndex(4);
      pPointE->close();


      // end of creating point(end) to model space
      return;
}
```

ค.1.2. Header Files

ประกอบด้วยแฟ้มข้อมูล 1 แฟ้มได้แก่ zglobal.h ซึ่งเป็นแฟ้มที่เก็บข้อมูลที่ใช้ในการกำหนดตัวแปรที่เป็นแบบ Global ทั้งหมดและสร้าง Class ใหม่ที่เพิ่มเข้าไปที่ใช้ใน ZAmpPunch.cpp zglobal.h มีรหัสคำสั่งดังนี้

```
//
// zglobal.h
//
// here's where i define a predefine class and variable
```

```
#include <aced.h>
#include <stdio.h>
//////////////////////////////////////////////////////
// here's the class name "ZPunch"
//////////////////////////////////////////////////////

typedef struct
{
      struct resbuf* nameFile;
      ////////////////////////
      // tool attributes
      ////////////////////////
      char kwTool[5];
      char loopKw[20];

      double toolDia;
      double toolDimx;
      double toolDimy;

//  Type of entity (1-circle,2-rectangle,3-arc)
      int eType;

      ////////////////////////
      // Attribute of enttity
      ////////////////////////
//          1. circle

      AcGePoint3d cCenter;
      double cRadius,cDia;

      char kwTypeCir[5];

//          2. rectangle
      AcGePoint3d recPt0,recPt1,recPt2,recPt3,recRfPt;
      double mcLength1,mcLength2,pitchX,pitchY,cornRad;
      char recDirectionLR[2],recDirectionUD[2];
      int refAt; // variable for rectype3 which have 4 choice of ref
point
      AcGePoint3d  recCenPt1,recCenPt2,recCenPt3;  //  for  punching
center (used in fillet also)
      char kwTypeRec[5];

//          3. arc
      AcGePoint3d arcCenter;
      double aRad,aEndAng,aStartAng,mcStartAng,mcTravAng,aPitch;

      char arcInOut[5];

//          4. line
      AcGePoint3d lStartPt, lEndPt, lineRefPt, lJointPt;
      char lineDirecLR[7], lineDirecUD[7], jointYN[5];
      int horVer; // 0 means vertical
      double jointVal;

//          5. corner and triangle fillet
      AcGePoint3d  fPt0,  fPt1,  fPt2,  fRefPt,fRefPtSec;//fRefPtSec
means second ref pt.
      int triOrCorn; // 0 mean tri, 1 means corn
      int cfStyle; // store style of corner fillet
      int cfType; // store type of corner fillet
      //int bigFitSmall; // 0 mean big, 1 mean fit, 2 mean small
```

```
      int triStyle; // store style of triangle fillet

}ZPunch;

// end of class definition "ZPunch"
/////////////////////////////////////////////////////

// Flags of ads_filed function
#define NEWFILE    0x01
#define NOTYPEIT   0x02
#define ARBEXT     0x04
#define LIBSRCH    0x08

#define           MAX_DCL_CHARS     32

// here's a stream pointer for operating file
FILE *stptr;
// other constant..
const double pi = 3.1415926;
```

ค.1.3. Library Files

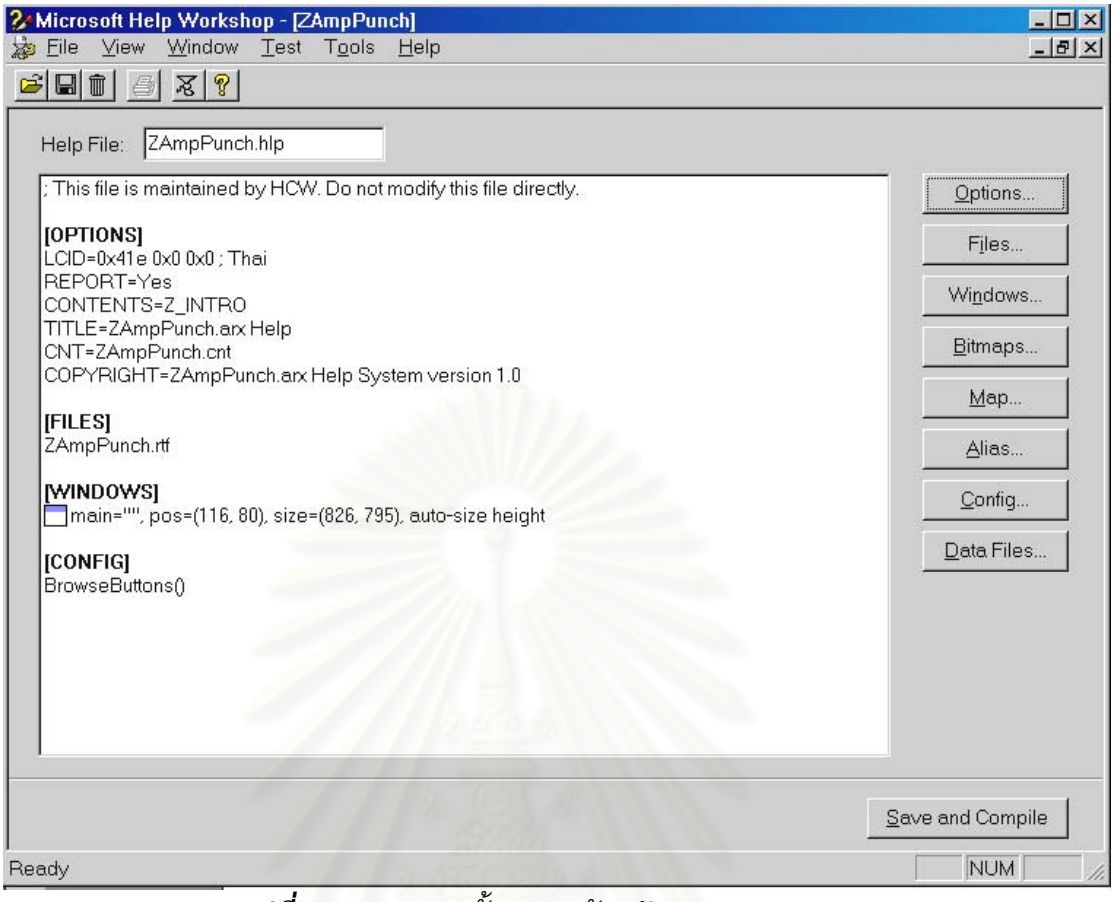
เป็นแฟ้มข้อมูล Libraries ซึ่งเก็บข้อมูลเกี่ยวกับการใช้งานและฐานข้อมูลของ AutoCAD R14 ทั้งหมดที่จำเป็นในการสร้าง ZAmpPunch.arx ซึ่งประกอบด้วย 5 แฟ้มข้อมูลได้แก่ Rxapi.lib Acedapi.lib Acgiapi.lib Libacge.lib และ Acad.lib แฟ้มข้อมูลเหล่านี้เป็นแฟ้มที่ทางบริษัท Autodesk จำกัด สร้างขึ้นเพื่อให้นำไปใช้ในการโปรแกรมมิ่งผ่าน AutoCAD R14 โดยระบบ ARX

ค.2. Help Files

ซอฟต์แวร์ ZAmpPunch.arx มีระบบช่วยเหลือ (Help) ซึ่งถูกสร้างขึ้นเพื่อเป็นประโยชน์ในการช่วยเหลือผู้ใช้สำหรับการใช้งานซอฟต์แวร์ โดยที่ผู้ใช้สามารถอ้างอิงไปพร้อมกับการใช้งาน ZAmpPunch.arx ชื่อของแฟ้มข้อมูลที่เป็นระบบช่วยเหลือคือ ZAmpPunch.hlp ซึ่งเกิดจากการสร้างโดยกระบวนการสร้าง On-line Help โดยโปรแกรม Microsoft Help Workshop ส่วนประกอบในการสร้าง ZAmpPunch.hlp ประกอบด้วย 3 ส่วนได้แก่ ZAmpPunch.hpj ZAmpPunch.cnt และ ZAmpPunch.rtf

ค.2.1. ZAmpPunch.hpj

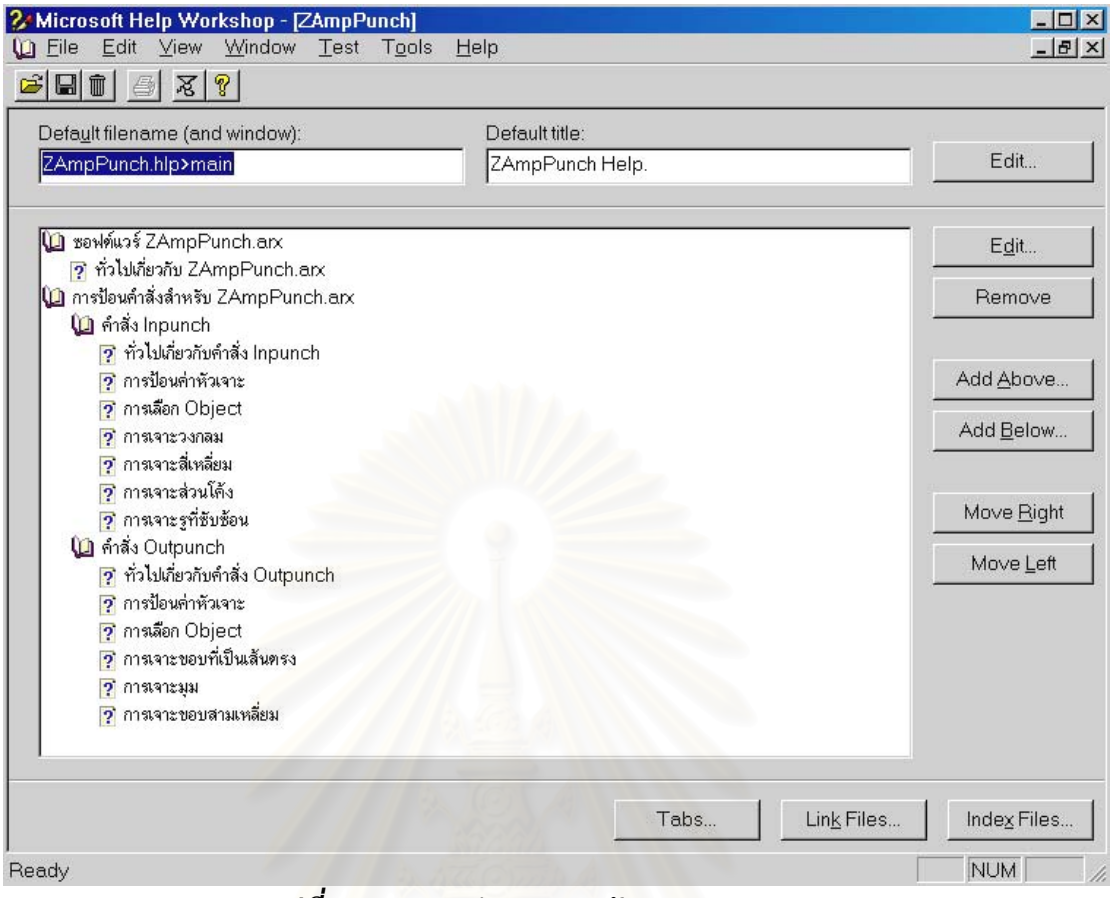
แฟ้มข้อมูล ZAmpPunch.hpj เป็นส่วนของ Project Files สำหรับสร้างสภาพแวดล้อมของระบบช่วยเหลือ โดยเป็นส่วนที่กำหนดรูปแบบต่าง ๆ ของการแสดงระบบช่วยเหลือ ค่าต่าง ๆ ถูกตั้งค่าไว้ดังรูปที่ ค.1



รูปที่ ค.1 แสดงการตั้งค่าสำหรับแฟ้ม ZAmpPunch.hpj

ค.2.1. ZAmpPunch.cnt

แฟ้มข้อมูล ZAmpPunch.hpj เป็นส่วนของ Content File ซึ่งเป็นส่วนของโครงสร้างของระบบช่วยเหลือที่จะแสดงให้ผู้ใช้ดู รูปแบบแสดงได้ในรูปที่ ค.2



รูปที่ ค.2 แสดงรูปแบบของแฟ้ม ZAmpPunch.cnt

ค.2.1. ZAmpPunch.rtf

แฟ้ม ZAmpPunch.rtf เป็นแฟ้มเอกสารในรูปแบบ Rich Text Format ซึ่งเป็นส่วนที่เก็บข้อมูลทั้งหมดที่จะแสดงให้ผู้ใช้ได้ดู โดยที่มันถูกสร้างขึ้นโดยรูปแบบที่กำหนดโดยใช้ Footnote เป็นตัวบอกถึงคุณสมบัติของส่วนต่าง ๆ ที่ประกอบอยู่ในเนื้อหาของระบบช่วยเหลือ

ภาคผนวก ง.

การติดตั้งและการใช้งานซอฟต์แวร์

ภาคผนวก ง.กล่าวถึงการติดตั้งและการใช้งานสำหรับซอฟต์แวร์ 2 ส่วนได้แก่ ส่วนของ ZAmpPunch.arx และส่วนของระบบช่วยเหลือ ZAmpPunch.hlp

ง.1. การติดตั้งและการใช้งาน ZAmpPunch.arx

ซอฟต์แวร์ช่วยวางแผนกระบวนการผลิตสำหรับกระบวนการเจาะรูของเครื่องจักร CNC Turret Punch Press เป็นแฟ้มข้อมูลที่เขียนขึ้นโดยระบบ ARX โดยภาษา C++ ซึ่งอยู่ในรูปแบบของแฟ้มข้อมูล Dynamic Link Libraries (DLLs) ประเภทหนึ่งซึ่งจะทำงานอยู่ภายใต้สภาพแวดล้อมของ AutoCAD R14 แฟ้มข้อมูลดังกล่าวมีชื่อว่า “ZAmpPunch.arx”

ง.1.1. การติดตั้ง ZAmpPunch.arx

การทำงานของซอฟต์แวร์จะทำงานอยู่ภายใต้สภาพแวดล้อม AutoCAD R14 ดังนั้นเมื่อต้องการใช้งาน ซอฟต์แวร์จะต้องถูกโหลดเข้าไปในหน่วยความจำของ AutoCAD R14 ทุกครั้ง วิธีการติดตั้งจะแบ่งออกได้เป็น 2 กรณี ดังนี้

- ต้องการให้ซอฟต์แวร์ ZAmpPunch.arx ถูกโหลดทุกครั้งเมื่อเริ่มเปิดใช้งานโปรแกรม AutoCAD R14 ซึ่งมีขั้นตอนในการติดตั้งดังนี้
    1. บันทึกซอฟต์แวร์ ZAmpPunch.arx ลงไปในตำแหน่งของ C:\Program Files\AutoCAD R14\SUPPORT\
    2. สร้างแฟ้มข้อมูลตัวอักษรชื่อ acad.rx โดยพิมพ์ชื่อและตำแหน่งของซอฟต์แวร์ที่มันอยู่ เช่น C:\Program Files\AutoCAD R14\SUPPORT\ ZAmpPunch.arx ไว้ในบรรทัดแรก และทำการบันทึกมันลงไปในตำแหน่งของ C:\Program Files\AutoCAD R14\SUPPORT\ แฟ้มข้อมูล acad.rx นี้เป็นส่วนที่ AutoCAD R14 จะเข้ามาตรวจสอบว่ามีซอฟต์แวร์ ARX ตัวใดที่ต้องโหลดเมื่อเริ่มต้นโปรแกรม

- ต้องการโหลดซอฟต์แวร์ ZAmpPunch.arx เมื่อกำลังเปิดใช้งานโปรแกรม AutoCAD R14 อยู่แล้ว มีขั้นตอนดังนี้
    1. บันทึกซอฟต์แวร์ ZAmpPunch.arx ลงไปในตำแหน่งใดก็ได้ในฮาร์ดดิสก์
    2. เลือก “Tools” บนเมนูของ AutoCAD R14
    3. เลือก “Load Application…”
    4. เลือก “Browse…” เพื่อเข้าไปหาซอฟต์แวร์ ZAmpPunch.arx ในตำแหน่งที่บันทึกไว้

5. เลือก “Load” เพื่อโหลดมันเข้าไปในหน่วยความจำ

เมื่อติดตั้งซอฟต์แวร์เรียบร้อยแล้ว จะสังเกตได้ว่าเมื่อซอฟต์แวร์ถูกโหลดเข้าไปในหน่วยความจำแล้ว จะมีข้อความขึ้นใน AutoCAD Command Prompt ดังนี้

Type INPUNCH or OUTPUNCH to start ZAmpPunch program

ซึ่งแสดงว่าพร้อมที่จะใช้งานซอฟต์แวร์ ZAmpPunch.arx เรียบร้อยแล้ว

ง.1.2. การใช้งาน ZAmpPunch.arx

การใช้งานซอฟต์แวร์เพื่อให้ได้รหัสโปรแกรม G Code ที่ถูกต้องที่สุดจะต้องใช้ซอฟต์แวร์กับ Drawing ที่เป็นไปตามข้อกำหนดที่เหมาะสมกับการใช้งาน ซึ่งเมื่อได้ Drawing ที่กำหนดแล้ว ทำการป้อนคำสั่งดังต่อไปนี้ที่ AutoCAD Command Prompt

- “inpunch” เมื่อต้องการเจาะรูภายใน
- “outpunch” เมื่อต้องการเจาะรูภายนอก

เมื่อป้อนคำสั่งดังกล่าวแล้วก็ทำงานตามข้อความที่ขึ้นมาให้ป้อนข้อมูล

ง.2. การติดตั้งและการใช้งานระบบช่วยเหลือ ZAmpPunch.hlp

ZAmpPunch.hlp เป็นระบบช่วยเหลือสำหรับ ZAmpPunch.arx ที่สามารถเรียกมาใช้งานได้บนการทำงานของโปรแกรม AutoCAD

ง.2.1. การติดตั้งระบบช่วยเหลือ ZAmpPunch.hlp

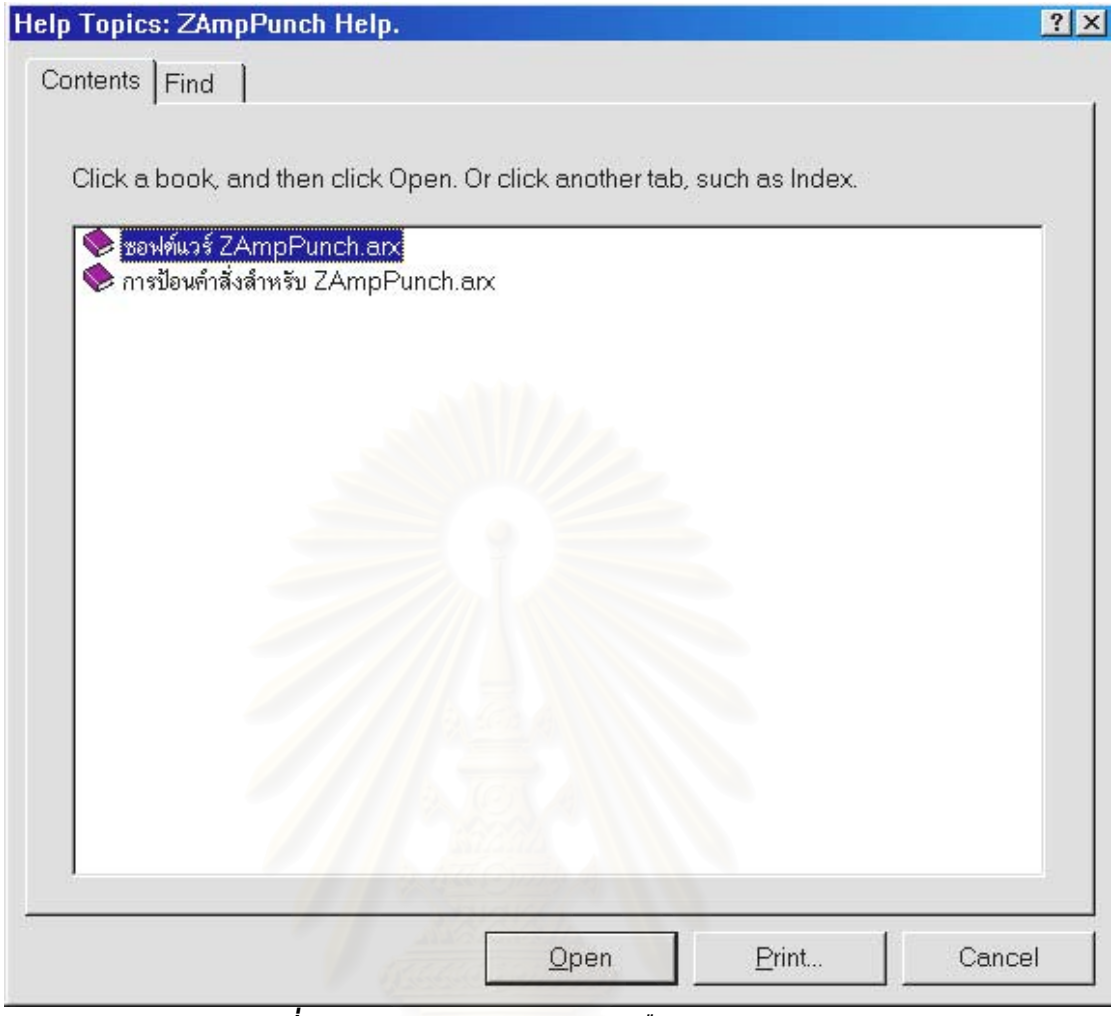
ในการติดตั้งระบบช่วยเหลือมีขั้นตอนดังนี้

1. บันทึกแฟ้ม ZAmpPunch.hlp และ ZAmpPunch.cnt ลงไปยังตำแหน่งของ C:\Program Files\AutoCAD R14\SUPPORT\
2. ทำการสร้างปุ่มในโปรแกรม AutoCAD เพื่อเรียกระบบช่วยเหลือนี้ขึ้นมาใช้โดยเริ่มจากการเปิดโปรแกรม AutoCAD R14

3. คลิ๊กปุ่มขวาของเมาส์ที่ Toolbar ใด ๆ
4. กด “NEW…” เพื่อสร้าง Toolbar สำหรับ ZAmpPunch.hlp ขึ้นมา
5. ตั้งชื่อในช่อง “Toolbar Name” จากนั้นกด “OK”
6. เมื่อกด “OK” แล้วจะเห็นว่ามี Toolbar ส่วนใหม่ถูกสร้างขึ้นมา จากนั้นคลิ๊กปุ่มขวาของเมาส์ที่ Toolbar ที่สร้างขึ้นใหม่
7. กำหนดคุณสมบัติ (Properties) ต่าง ๆ ที่ต้องการ แล้วปิด Dialog Box นั้น
8. จากนั้นกด “Customize…” เพื่อเพิ่มปุ่มเข้าไปบน Toolbar ที่สร้างขึ้นใหม่
9. เลือกหัวข้อ “Custom” จาก Drop-down menu แล้วทำการลากปุ่มที่ว่างไปยัง Toolbar ที่สร้างขึ้น
10. ปิด Dialog Box ของ “Customize Toolbar”
11. จากนั้นคลิ๊กปุ่มขวาของเมาส์ที่ปุ่มที่สร้างขึ้นใหม่ ซึ่งจะเห็นว่ามี Dialog Box ขึ้นมา
12. ส่วนช่องของ “Name” ให้ตั้งชื่อปุ่ม
13. ส่วนช่องของ “Help” ให้ใส่ข้อความที่ต้องการแสดงให้ผู้ใช้เห็นเมื่อนำเมาส์เข้าไปวาง
14. ส่วนของ “Macro” ให้พิมพ์ว่า “**^C^C(Help “ZAmpPunch.hlp”)**”
15. ส่วนของ Icon ที่ต้องการอาจเลือกจากรายการที่มีให้หรืออาจสร้างรูปใหม่ก็ได้
16. จากนั้นปิด Dialog Box “Button Properties” ซึ่งถือว่าเสร็จเรียบร้อยแล้ว

ง.2.2. การใช้งานระบบช่วยเหลือ ZAmpPunch.hlp

การใช้งานทำได้โดยการกดปุ่มที่ได้สร้างขึ้นมา โดยระบบช่วยเหลือจะมีรูปแบบดังรูป ง.1

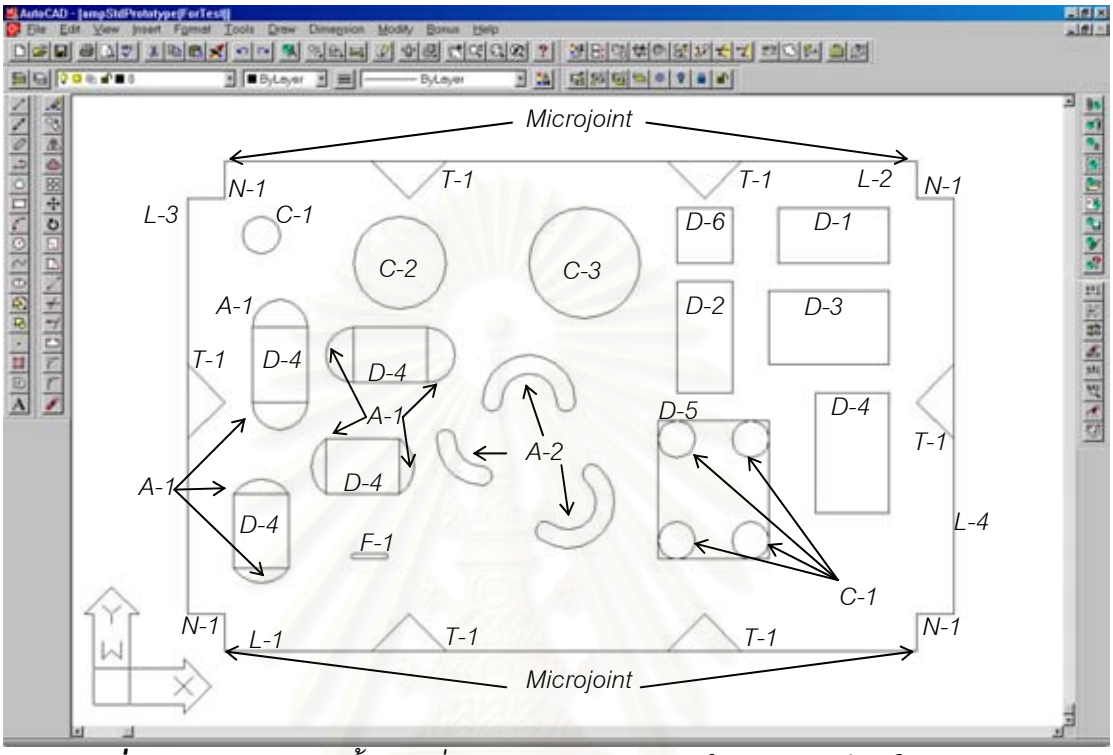


รูปที่ ง.1 *รูปแบบของระบบช่วยเหลือ ZAmpPunch.hlp*

ภาคผนวก จ.

ตัวอย่างการทดสอบความถูกต้องของซอฟต์แวร์เพื่อสร้างรหัสโปรแกรม G Code สำหรับเจาะชิ้นงาน

ชิ้นงานที่นำมาทำการทดสอบเป็น Drawing ของแผ่นโลหะซึ่งถูกสร้างขึ้นภายใน AutoCAD ตามข้อกำหนดทั้งหมดที่เหมาะสมกับการนำมาใช้กับซอฟต์แวร์ ชิ้นงานมีลักษณะดังรูปที่ จ.1 โดยมีรหัสของแต่ละ Object อยู่ภายในซึ่งจะใช้อธิบายภายหลัง



***รูปที่ จ.1** Drawing ของชิ้นงานที่เหมาะสมกับการนำมาใช้กับซอฟต์แวร์ใน AutoCAD*

รายละเอียดภายในของชิ้นงานสรุปได้ดังนี้

- ใช้หน่วยวัดเป็นมิลลิเมตรสำหรับสร้างชิ้นงานนี้
- รหัสที่ใช้มีรายละเอียดแสดงในรูปที่ จ.2 และในตารางที่ จ.1

***รูปที่ จ.2** แสดงรหัสของ Object ในชิ้นงานทดสอบ*

ตารางที่ จ.1 *รายละเอียดของรหัสของ Object ในชิ้นงานทดสอบ*

รหัสตัวหน้า	รหัสตัวหลัง	ความหมาย	หมายเหตุ
A	1	ส่วนโค้ง / เจาะภายใน	คำสั่ง “RAD/”
	2	ส่วนโค้ง / เจาะภายนอก	คำสั่ง “RAD/”
C	1	วงกลม / เจาะแบบที่ 1	เจาะครั้งเดียวตำแหน่งศก.หัวเจาะ
	2	วงกลม / เจาะแบบที่ 2	คำสั่ง “HOL/”
	3	วงกลม / เจาะแบบที่ 3	คำสั่ง “OPN/”
D	1	สี่เหลี่ยม / เจาะแบบที่ 1	คำสั่ง “REC/” แนวนอน
	2	สี่เหลี่ยม / เจาะแบบที่ 2	คำสั่ง “REC/” แนวตั้ง
	3	สี่เหลี่ยม / เจาะแบบที่ 3	คำสั่ง “REC/” แนวนอนและตั้ง
	4	สี่เหลี่ยม / เจาะแบบที่ 4	คำสั่ง “OBL/”
	5	สี่เหลี่ยม / เจาะแบบที่ 5	คำสั่ง “RRC/”
	6	สี่เหลี่ยม / เจาะแบบที่ 6	เจาะครั้งเดียวตำแหน่งศก.หัวเจาะ
F	1	รูป Oblong	เจาะครั้งเดียวตำแหน่งศก.หัวเจาะ
L	1	เส้นตรง / เจาะส่วนแรก	คำสั่ง “REC/” แนวนอน
	2	เส้นตรง / เจาะส่วนที่สอง	คำสั่ง “REC/” แนวนอน
	3	เส้นตรง / เจาะส่วนที่สาม	คำสั่ง “REC/” แนวตั้ง
	4	เส้นตรง / เจาะส่วนที่สี่	คำสั่ง “REC/” แนวตั้ง
N	1	ขอบมุม	คำสั่ง “REC/” แนวนอนและตั้ง
T	1	ขอบด้านข้างที่เป็นสามเหลี่ยม	เจาะครั้งเดียวตำแหน่งศก.หัวเจาะ

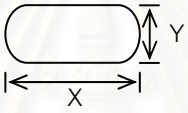
- มีส่วนของ Microjoint ที่ถูกกำหนดขึ้นเพื่อทำให้ชิ้นงานไม่ตกหล่นลงบนโต๊ะซึ่งกำหนดเอาไว้ที่ 4 จุด ซึ่งมีตำแหน่งในรูปที่ จ.1

จ.2. การสร้างรหัสโปรแกรม G Code โดยซอฟต์แวร์

เมื่อสร้างชิ้นงานที่เหมาะสมกับการใช้ซอฟต์แวร์แล้ว ขั้นตอนต่อไปผู้วางแผนจะต้องพิจารณาว่ารูที่มีอยู่ทั้งหมดจะต้องใช้หัวเจาะอะไรบ้าง แล้ววางแผนการติดตั้งหัวเจาะเหล่านั้นลงบนตำแหน่งหัวเจาะบนแท่นหมุน

ลักษณะการเจาะนั้นจะกระทำโดยใช้หัวเจาะตัวหนึ่งสำหรับ Object ต่าง ๆ ให้เสร็จก่อนแล้วจึงเปลี่ยนหัวเจาะ สำหรับชิ้นงานนี้การวางแผนด้านการใช้หัวเจาะแสดงในตารางที่ จ.2

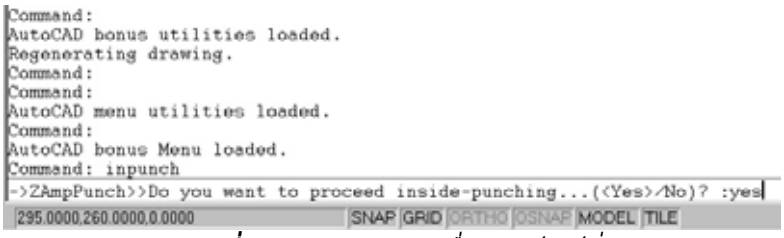
ตารางที่ จ.2 *การวางแผนการเจาะด้านการใช้หัวเจาะของชิ้นงานทดสอบ*

ลำดับการใช้หัวเจาะ	รูปร่างของหัวเจาะ	ขนาด	ตำแหน่งบนแท่นหมุน	Object ที่จะเจาะ (ตามลำดับ)
1	วงกลม	∅ 20.00 มม.	02	C-1, C-2, C-3 และ A-1
2	วงกลม	∅ 10.00 มม.	03	A-2
3	สี่เหลี่ยมจัตุรัส	30.00 x 30.00 มม.	04	D-1, D-2, D-3, D-4, D-5 และ D-6
4	Oblong	20.00 x 3.00 มม. (X x Y) 	06	F-1
5	สี่เหลี่ยมผืนผ้า	40.00 x 5.00 มม. (X x Y)	01 (หมุนได้)	L-1, L-2, L-3 และ L-4
6	สี่เหลี่ยม	12.50 x 12.50 มม.	07	N-1
7	สามเหลี่ยมมุมฉาก	ฐาน 40.00 สูง 20.00 มม.	12 (หมุนได้)	T-1

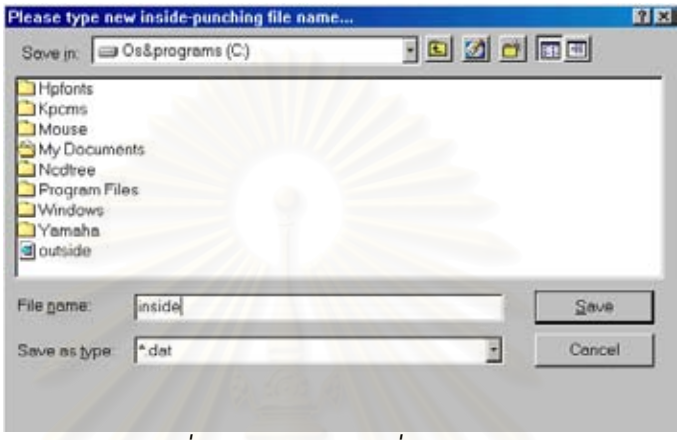
จากนั้นเริ่มใช้ซอฟต์แวร์เพื่อช่วยในการสร้างรหัสโปรแกรม G Code ซึ่งการเจาะชิ้นงานจะเริ่มจากการเจาะภายใน แล้วตามด้วยการเจาะภายนอก ขั้นตอนมีดังนี้

จ.2.1. การเจาะภายใน

- ป้อนคำสั่ง “inpunch” เพื่อเจาะ Object ภายใน ที่ AutoCAD Command Prompt
- ซอฟต์แวร์จะถามว่าต้องการจะทำต่อหรือไม่ ดังรูปที่ จ.3 ถ้าทำต่อ มันก็จะเข้าไปสร้างแฟ้มข้อมูลที่เป็นตัวอักษรขึ้นเพื่อพร้อมที่จะเขียน G Code ลงไป โดยที่ผู้ใช้ตั้งชื่อแฟ้มที่ต้องการจัดเก็บ ลงใน Dialog Box ซึ่งแฟ้มข้อมูลที่สร้างขึ้นมีนามสกุลเป็น “dat” ดังรูปที่ จ.4 และจากนั้นกด Save



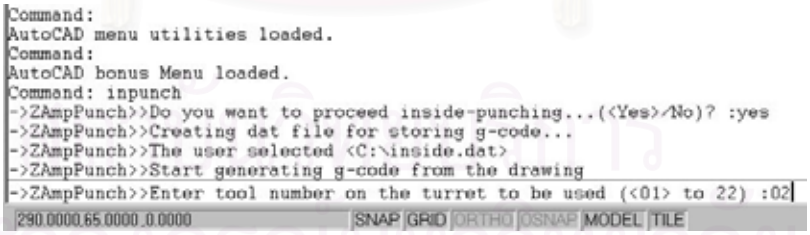
รูปที่ จ.3 แสดงข้อความเมื่อซอฟต์แวร์เริ่ม



รูปที่ จ.4 แสดงหน้าจอรับชื่อของแฟ้มข้อมูลที่จะเก็บ G Code ของการเจาะภายใน

- ลักษณะการเจาะนั้นจะกระทำโดยใช้หัวเจาะตัวหนึ่งสำหรับ Object ต่าง ๆ ให้เสร็จก่อนแล้วจึงเปลี่ยนหัวเจาะดังนั้นซอฟต์แวร์จะถามว่าต้องการจะใช้หัวเจาะหมายเลขใดในการเจาะก่อนที่จะเลือก Object ที่จะเจาะ ซึ่งเมื่อใช้หัวเจาะตัวนั้นเสร็จแล้วก็จะมีการเปลี่ยนหัวเจาะหมายเลขที่ป้อนจะเป็น 01 ถึง 22 ส่วนต่อไปจะอธิบายย่อยสำหรับหัวเจาะแต่ละหมายเลขในการเจาะ Object ของมัน ซึ่งเริ่มจากหัวเจาะหมายเลข 02 ดังรูปที่ จ.5

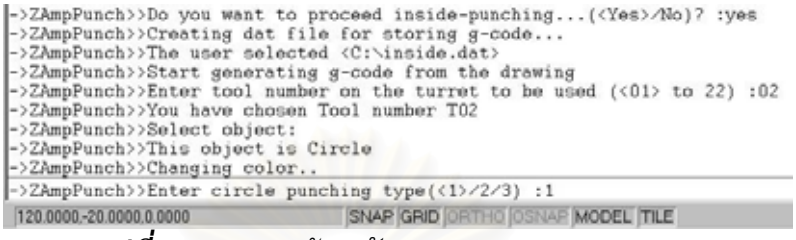
รูปที่ จ.5 แสดงหน้าจอรับค่าของตำแหน่งหัวเจาะที่จะใช้



การเจาะด้วยหัวเจาะหมายเลข 02

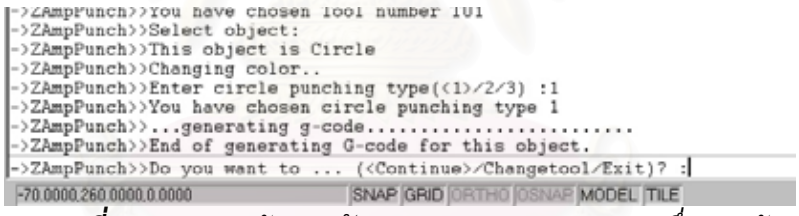
- ซอฟต์แวร์จะให้ผู้ใช้เลือก Object ที่ต้องการใช้หัวเจาะหมายเลข 02 ในการเจาะ ซอฟต์แวร์จะให้ผู้ใช้เป็นผู้กำหนดเส้นทางในการเจาะว่าต้องการจะเจาะ Object ใดก่อนหลัง ลำดับจะขึ้นอยู่กับการวางแผนในตารางที่ จ.2 ซึ่งจะเริ่มจากการเจาะวงกลม C-1

- เมื่อผู้ใช้เลือกวงกลม C-1 แล้ว ซอฟต์แวร์จะบอกว่า Object ที่เลือกเป็นวงกลม สีของ Object ที่เลือกจะเปลี่ยนเป็นสีฟ้าเช่นเดียวกับ Object ต่าง ๆ ที่จะถูกเลือก ต่อมามันจะถามว่าต้องการเจาะวงกลมในลักษณะใด สำหรับวงกลม C-1 จะเจาะแบบที่ 1 ซึ่งก็ป้อน 1 เข้าไป ดังรูปที่ จ.6



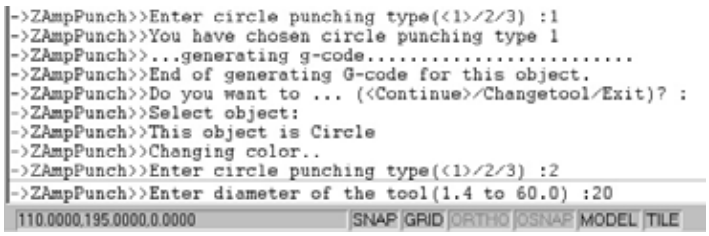
รูปที่ จ.6 *แสดงหน้าจอรับค่าแบบของการเจาะวงกลม*

- สำหรับ C-1 เป็นการเจาะครั้งเดียวในตำแหน่งจุดศก.ของหัวเจาะ ดังนั้นไม่ต้องป้อนค่าอะไรเพิ่มเติม ซอฟต์แวร์จะสร้าง G Code สำหรับ C-1 ไว้ในแฟ้มที่สร้างไว้เรียบร้อยแล้ว ซึ่งผลจากการสร้าง G Code สำหรับแต่ละ Object เสร็จทุกครั้งจะมีรูปแบบดังรูปที่ จ.7 ซึ่งมันจะถามต่อไปว่าต้องการจะใช้หัวเจาะนี้เจาะต่อ (ป้อน “C”) หรือต้องการจะเปลี่ยนหัวเจาะ (ป้อน “Ch”) หรือจะสิ้นสุดการใช้ซอฟต์แวร์ (ป้อน “E”)



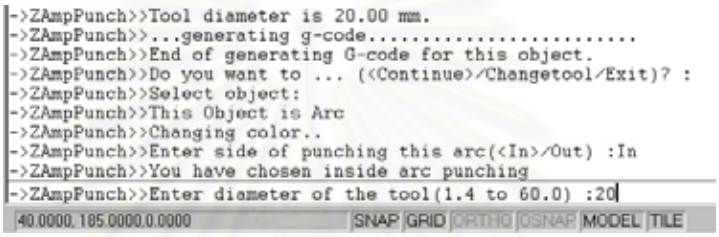
รูปที่ จ.7 *แสดงหน้าจอหลังจากจบการเจาะ Object หนึ่ง ๆ แล้ว*

- ป้อน “C” เนื่องจากต้องใช้หัวเจาะตัวนี้เจาะต่อ สำหรับ Object ต่อไปที่จะต้องเลือกคือวงกลม C-2 ซึ่งจะเจาะแบบที่ 2 (คำสั่ง “HOL/”) ซึ่งก็ป้อนค่า “2” เข้าไป ในการเจาะวงกลมแบบที่ 2 จะต้องมีการป้อนค่าของขนาดเส้นผ่านศก.ของหัวเจาะด้วย ดังรูปที่ จ.8 ซึ่งเมื่อป้อนแล้วก็จะเสร็จสำหรับ Object C-2



รูปที่ จ.8 *แสดงหน้าจอรับค่าของขนาดเส้นผ่านศก.หัวเจาะในการเจาะวงกลม*

- จากนั้นทำการเจาะวงกลม C-3 ซึ่งเป็นการเจาะแบบที่ 3 (คำสั่ง “OPN/”) ซึ่งคล้ายกับการเจาะ C-2 คือต้องป้อนค่าของขนาดเส้นผ่านศก.ของหัวเจาะด้วย จากนั้นถือว่าเสร็จสิ้นการสร้าง G Code ของ C-3
- Object ต่อไปสุดท้ายสำหรับหัวเจาะ 02 คือ ส่วนโค้ง A-1 (ใช้คำสั่ง “RAD/”) ซึ่งเป็นส่วนโค้งที่เจาะเพื่อที่จะสร้างรูป Oblong โดยการสร้างรวมกับรูปสี่เหลี่ยม D-4 การเจาะ A-1 ก็ทำการเลือกส่วนโค้ง A-1 แล้วซอฟต์แวร์จะถามว่าต้องการเจาะแบบภายใน (ป้อน “In”) หรือเจาะแบบภายนอก (ป้อน “Out”) สำหรับส่วนโค้ง A-1 จะเจาะภายใน และอีกส่วนที่ต้องป้อนเข้าได้แก่ขนาดเส้นผ่านศก.ของหัวเจาะด้วย ดังรูปที่ จ.9 จากนั้นถือว่าเสร็จการเจาะส่วนโค้ง A-1 และเสร็จการใช้หัวเจาะ 02



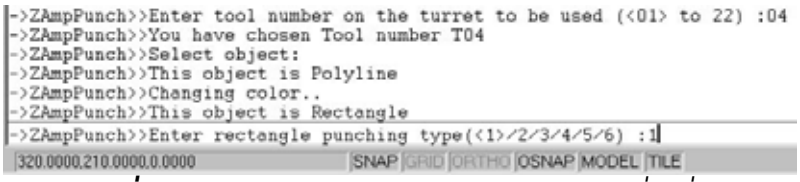
รูปที่ จ.9 *แสดงหน้าจอรับค่าของขนาดเส้นผ่านศก.หัวเจาะในการเจาะส่วนโค้ง*

การเจาะด้วยหัวเจาะหมายเลข 03

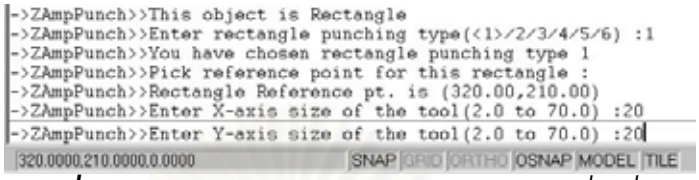
- เมื่อเจาะ Object ที่ผ่านมาเสร็จแล้ว ผู้ใช้ป้อน “Ch” เพื่อเปลี่ยนหัวเจาะในการเจาะ Object ต่อไป Object ที่เลือกต่อมาคือส่วนโค้ง A-3 (ใช้คำสั่ง “RAD/”) ซึ่งเป็นรูส่วนโค้งที่มีความกว้าง 10 มม. การเจาะจะเจาะภายนอกส่วนโค้งที่ได้เลือกทีละส่วนโค้ง ขั้นตอนการเจาะ Object ที่เป็นส่วนโค้งก็มีลักษณะเดียวกับส่วนโค้ง A-1 ที่ต้องป้อนว่าจะเจาะภายในหรือภายนอก และการป้อนค่าขนาดเส้นผ่านศก.ของหัวเจาะ จึงถือว่าเสร็จสิ้นสำหรับส่วนโค้ง A-2 และเสร็จสิ้นการใช้หัวเจาะหมายเลข 03

การเจาะด้วยหัวเจาะหมายเลข 04

- หัวเจาะหมายเลข 04 จะใช้เจาะ Object ที่เป็นสี่เหลี่ยมทั้งหมด ซึ่งจะเริ่มจากสี่เหลี่ยม D-1 ซึ่งเป็นการเจาะแบบที่ 1 (ใช้คำสั่ง “REC/” แนวนอน) เมื่อเริ่มเลือกสี่เหลี่ยม D-1 ซอฟต์แวร์จะถามว่าต้องการเจาะแบบใด ผู้ใช้ป้อน “1” ดังรูปที่ จ.10 โดยที่การเจาะแบบที่ 1 จะให้ผู้ใช้เลือกจุดอ้างอิงในการเริ่มเจาะซึ่งจุดที่เป็นจุดอ้างอิงจะต้องเป็นจุดมุมล่างด้านซ้ายหรือด้านขวาเท่านั้น จากนั้นต้องป้อนขนาดหัวเจาะในด้านแกน X และแกน Y ด้วยดังรูปที่ จ.11จากนั้นถือว่าเสร็จสิ้นการเจาะสี่เหลี่ยม D-1



รูปที่ จ.10 *แสดงหน้าจอรับค่าแบบของการเจาะสี่เหลี่ยม*



รูปที่ จ.11 *แสดงหน้าจอรับค่าขนาดของหัวเจาะสี่เหลี่ยม*

- Object ต่อไปคือ D-2 ซึ่งเป็นการเจาะแบบที่ 2 (ใช้คำสั่ง “REC/” แนวตั้ง) ซึ่งเริ่มจากการเลือกสี่เหลี่ยมเช่นกันแล้วเลือกจุดอ้างอิง สำหรับแบบที่ 2 จุดอ้างอิงจะเป็นได้เฉพาะจุดด้านซ้ายบนหรือล่างของสี่เหลี่ยมเท่านั้น ซึ่งเมื่อเลือกเสร็จ การป้อนค่าอื่น ๆ จะเหมือนกับ Object D-1
- สำหรับ D-3 เป็นการเจาะแบบที่ 3 (ใช้คำสั่ง “REC/” ในแนวนอนและตั้ง) ซึ่งเป็นการเปิดรูสี่เหลี่ยมโดยที่ไม่มีการเอาเศษเหล็ก (ถ้ามี) ตรงกลางออก ซึ่งค่าที่ป้อนคือ 3 เมื่อซอฟต์แวร์ถามว่าจะให้เจาะแบบใด แล้วทำการเลือกจุดเริ่ม โดยที่จุดเริ่มจะต้องเป็นจุดมุมใดมุมหนึ่งของสี่เหลี่ยมเท่านั้น และต้องป้อนค่าขนาดของหัวเจาะเช่นเดียวกัน
- จากนั้นเจาะ D-4 ซึ่งเป็นการเจาะแบบที่ 4 (ใช้คำสั่ง “REC/” ในแนวนอนและตั้ง) ซึ่งเป็นการเปิดรูสี่เหลี่ยมโดยที่มีการเอาเศษเหล็ก (ถ้ามี) ตรงกลางออก ค่าที่ป้อนคือ 3 ลักษณะการป้อนค่าอื่น ๆ จะมีลักษณะเหมือนกับ D-3
- Object ต่อไปคือ D-5 ซึ่งเป็นลักษณะการเจาะสี่เหลี่ยมที่มีมุมเป็นส่วนโค้ง (คำสั่ง “RRC/”) ภาพที่สร้างขึ้นมีมุมซึ่งเป็น Object วงกลม C-1 ประกอบอยู่ การเจาะ D-5 เริ่มจากการป้อน 5 เมื่อถามว่าจะเจาะแบบใด จากนั้นซอฟต์แวร์จะถามขนาดของหัวเจาะ แต่ไม่ต้องกำหนดจุดอ้างอิง เพราะซอฟต์แวร์จะคำนวณจุดให้ และผู้ใช้ต้องป้อนค่าของขนาดรัศมีของมุมที่เป็นส่วนโค้งด้วย
- สุดท้ายสำหรับหัวเจาะหมายเลข 04 ได้แก่การเจาะ D-6 ซึ่งเป็นการเจาะรูสี่เหลี่ยมที่มีขนาดเดียวกับหัวเจาะ การเจาะก็จะเกิดขึ้นเพียงครั้งเดียวที่จุดศก.ของหัวเจาะ การป้อนคำสั่งเพียงป้อน 6 เมื่อถามว่าจะเจาะแบบใดเท่านั้น ตอนนี้ถือว่าสิ้นสุดการเจาะด้วยหัวเจาะหมายเลข 04

การเจาะด้วยหัวเจาะหมายเลข 06

- เป็นการเจาะ Object ที่มีลักษณะเป็น Oblong ซึ่งมีขนาดเดียวกันกับหัวเจาะ การเจาะจะเป็นการเจาะเพียงครั้งเดียวที่จุดศก.ของหัวเจาะ ผู้ใช้เพียงแค่เลือก Object ดังกล่าว ซอฟต์แวร์ก็จะสร้าง G Code ให้เลย เมื่อสิ้นสุดตรงนี้ถือว่าสิ้นสุดการเจาะภายในชิ้นงานแล้วให้ป้อน “Exit” เพื่อจบการเขียน G Code สำหรับการเจาะภายในต่อไปจะเจาะภายนอกชิ้นงาน

จ.2.2. การเจาะภายนอก

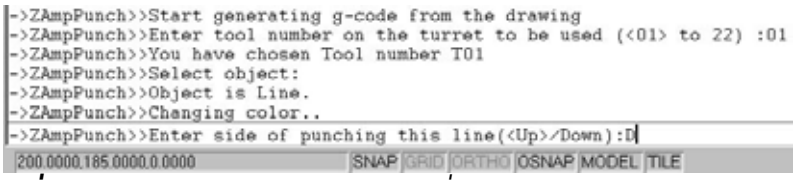
- ป้อนคำสั่ง “outpunch” เพื่อเจาะ Object ภายนอก ที่ AutoCAD Command Prompt
- คำสั่งนี้จะมีขั้นตอนเหมือนกันกับการเจาะภายในซึ่งจะให้ผู้ใช้สร้างแฟ้มข้อมูลที่เป็นตัวอักษรเก็บไว้ โดยลักษณะจะเหมือนกันกับรูปที่ จ.3 และรูปที่ จ.4
- การเจาะภายนอกจะเริ่มจากการใช้หัวเจาะหมายเลข 01 07 และ 12 ตามลำดับ ซึ่งมีรายละเอียดในการเจาะของแต่ละหัวเจาะดังนี้

การเจาะด้วยหัวเจาะหมายเลข 01

สำหรับหัวเจาะหมายเลข 01 เป็นหัวเจาะสี่เหลี่ยมที่มีขนาดด้านแกน X คือ 40 มม.และแกน Y คือ 5 มม. หัวเจาะหมายเลขนี้หมุนได้ ดังนั้นการติดตั้งหัวเจาะเริ่มต้นเป็นสิ่งสำคัญที่ต้องคำนึงถึง สำหรับชิ้นงานนี้หัวเจาะหมายเลข 01 ติดตั้งในลักษณะขนาดแนวแกน X เป็น 40 มม. และแกน Y คือ 5 มม. (ซึ่งเมื่อหัวเจาะหมุนไป 90 องศาจะทำให้ขนาดด้านแกน X เป็น 5 มม. และด้านแกน Y เป็น 40 มม. แทน)

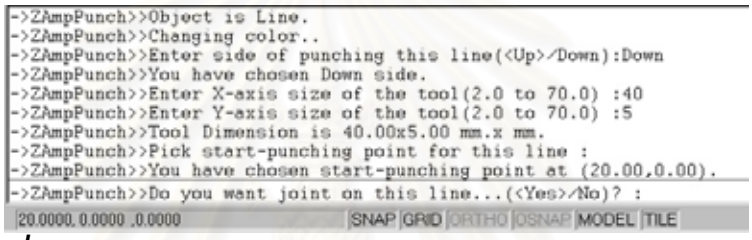
อย่างไรก็ตามการเจาะแบบใช้หัวเจาะหมุนได้ จะต้องเพิ่ม G Code ส่วนของ “C” โดยเพิ่มไว้หลัง Code “T” เช่น MOV/X10Y120T01C00 โดยที่ตัวเลขตามหลัง “C” คือองศาที่จะให้หมุนไป อาทิหากเป็น C90 แสดงว่าให้หมุนหัวเจาะไป 90 องศา G Code ในส่วนของ “C” เป็นส่วนที่ผู้วางแผนต้องเขียนเพิ่มลงไปเองในแฟ้มข้อมูลที่ซอฟต์แวร์สร้างขึ้น

- เริ่มด้วยการเจาะ Object ที่เป็นเส้น L-1 ซึ่งจะเจาะโดยใช้คำสั่ง REC (เหมือนการเจาะสี่เหลี่ยมในแนวนอน การเจาะเริ่มจากเลือกเส้น L-1 แล้วซอฟต์แวร์จะถามว่าจะเจาะเหนือเส้น (ป้อน “Up”) หรือใต้เส้น (ป้อน “Down”) ในที่นี้ป้อน “Down” ดังรูปที่ จ.12



รูปที่ จ.12 *แสดงหน้าจอรับค่าของด้านที่ต้องการเจาะของเส้นขอบ*

ป้อน "Down" ลงไปแล้วซอฟต์แวร์จะถามขนาดของหัวเจาะด้านแกน X และแกน Y ซึ่งก็ป้อน 40 และ 5 ลงไปตามลำดับจากนั้นให้เลือกจุดเริ่มในการเจาะขอบ L-1 ซึ่งเมื่อเลือกแล้ว ซอฟต์แวร์ถามว่าจะให้มี Microjoint ที่ปลายทั้งสองของขอบหรือไม่ ดังรูปที่ จ.13 เมื่อตอบต้องการซอฟต์แวร์จะแสดงตำแหน่ง Microjoint ที่ต้องการให้เห็นในรูป Drawing ซึ่งถือว่าเป็นการเสร็จสิ้นสำหรับเจาะ L-1



รูปที่ จ.13 *แสดงหน้าจอรับค่าของการเลือกให้ขอบมี Microjoint หรือไม่*

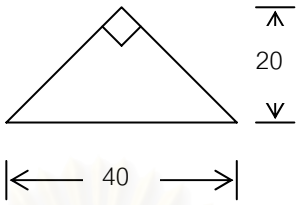
- ต่อไปจะเจาะ L-2 ซึ่งเป็นเส้นแนวนอนเหมือนกับ L-1 การเจาะจึงทำในลักษณะเดียวกันแต่ต่างกันตรงที่เป็นการเจาะเหนือเส้นที่เลือก
- การเจาะ L-3 จะเจาะโดยใช้การหมุนหัวเจาะไป 90 องศา ซึ่งเมื่อเลือกเส้น L-3 แล้วซอฟต์แวร์จะถามว่าจะให้เจาะด้านซ้าย (ป้อน "Left") หรือด้านขวา (ป้อน "Right") สำหรับ L-3 แล้วเป็นการเจาะด้านซ้ายของขอบ เมื่อป้อนด้านแล้วซอฟต์แวร์จะให้ป้อนค่าซึ่งมีขั้นตอนเหมือน L-1 โดยที่ค่าของขนาดหัวเจาะที่ต้องระบุจะเปลี่ยนไปเนื่องจากได้มีการหมุนหัวเจาะแล้ว ขนาดหัวเจาะที่ป้อนในแนวแกน X จะกลายเป็น 5 และแกน Y เป็น 40 สำหรับเส้นแนวตั้ง L-3 และ L-4 จะไม่ต้องมี Microjoint อยู่ เพราะได้กำหนดไปแล้วในส่วนของ L-1 และ L-2
- การเจาะ L-4 จะทำในลักษณะเดียวกับ L-3 แต่แตกต่างกันที่ด้านจะเป็นการเจาะในด้านขวาของขอบ จากนั้นถือว่าใช้หัวเจาะหมายเลข 01 เสร็จแล้ว

การเจาะด้วยหัวเจาะหมายเลข 07

- จะเป็นการเจาะมุมภายนอกของชิ้นงานสำหรับการพับซึ่งมีอยู่ 4 มุมด้วยกัน การเจาะ N-1 จะทำโดยการเลือกแต่ละมุม แล้วซอฟต์แวร์จะถามเพียงขนาดของหัวเจาะด้านแกน X และแกน Y เท่านั้น จากนั้นถือว่าเสร็จสำหรับการเจาะด้วยหัวเจาะหมายเลข 07

การเจาะด้วยหัวเจาะหมายเลข 12

หัวเจาะหมายเลข 12 เป็นหัวเจาะที่หมุนได้เช่นเดียวกับหมายเลข 01 ซึ่งในที่นี้ติดตั้งหัวเจาะรูปสามเหลี่ยมซึ่งมีลักษณะการติดตั้งตอนเริ่มต้นดังรูปที่ จ.14



รูปที่ จ.14 *ทิศทางการติดตั้งหัวเจาะสามเหลี่ยม*

- ลำดับการเจาะ T-1 จะเริ่มที่ขอบด้านล่าง ด้านขวา (หมุนหัวเจาะ 90 องศา) ด้านบน (หมุนหัวเจาะ 180 องศา) และด้านซ้าย (หมุนหัวเจาะ 270 องศา) ตามลำดับโดยการเจาะเป็นลักษณะเจาะเพียงครั้งเดียว (สำหรับแต่ละจุด) ในตำแหน่งจุดศก.ของหัวเจาะ ดังนั้นผู้ใช้เพียงเลือกขอบสามเหลี่ยมที่ต้องการแล้วซอฟต์แวร์จะสร้าง G Code ให้เลย ซึ่งถือว่าเป็นการสิ้นสุดการสร้าง G Code สำหรับการเจาะภายนอกชิ้นงาน

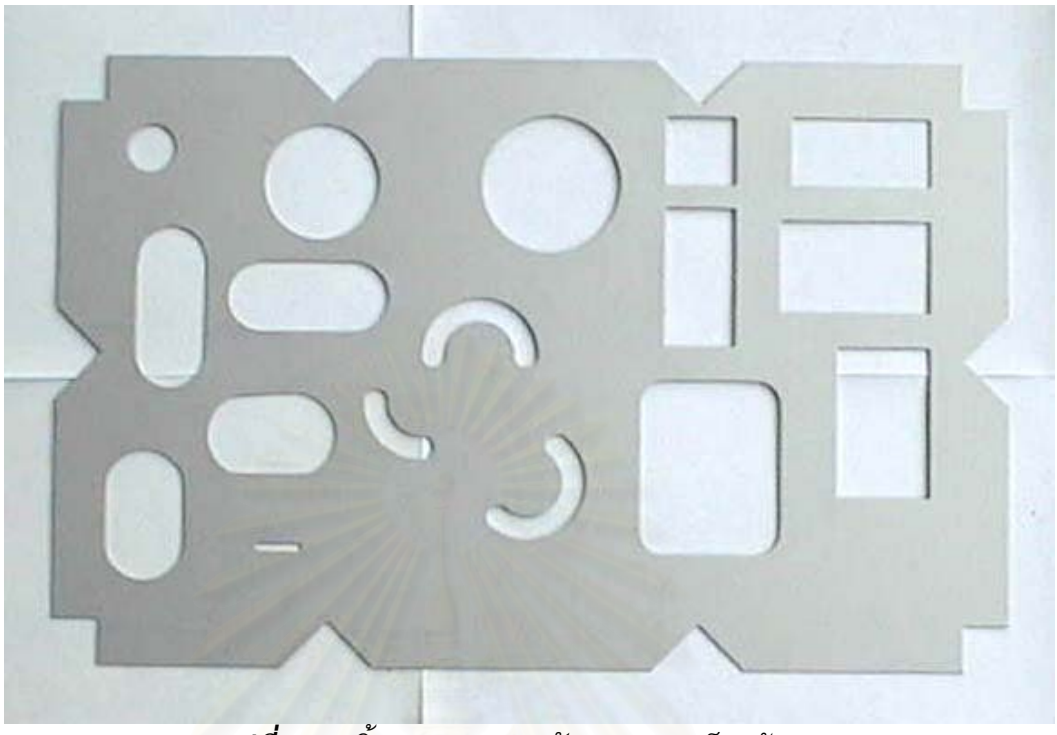
หลังจากที่ได้รหัสโปรแกรม G Code สำหรับการเจาะภายในและภายนอกชิ้นงานแล้ว (แสดงได้ในรูปที่ จ.15 และรูปที่ จ.16) ผู้วางแผนจะต้องนำเอารหัสโปรแกรมทั้งสองส่วนมารวมเข้าด้วยกันเนื่องจากมันถูกเก็บอยู่คนละแฟ้มข้อมูล แล้วบันทึกเก็บเอาไว้เป็นรหัสโปรแกรม G Code สำหรับผลิตภัณฑ์ต้นแบบ ซึ่งพร้อมที่จะนำไปทดสอบกับเครื่องจักร CNC

รูปที่ จ.15 *รหัสโปรแกรม G Code สำหรับการเจาะภายในที่เปิดด้วยโปรแกรม Notepad*

รูปที่ จ.16 รหัสโปรแกรม G Code สำหรับการเจาะภายนอกที่เปิดด้วยโปรแกรม Notepad

จ.3. การนำรหัสโปรแกรม G Code ป้อนเข้าเครื่องจักร CNC

หลังจากที่ได้รหัสคำสั่งโปรแกรม G Code สมบูรณ์พร้อมจะนำไปทดสอบแล้ว ผู้วางแผนกระบวนการผลิตทำการติดตั้งหัวเจาะที่ได้วางแผนไว้ แล้วนำแผ่นโลหะทดสอบมาติดตั้งที่โต๊ะ จากนั้นทำการเจาะ ผลของชิ้นงานที่ได้แสดงในรูปที่ จ.17



รูปที่ จ.17 *ชิ้นงานทดสอบหลังจากเจาะเสร็จแล้ว*

ภาคผนวก ฉ.

แบบสอบถาม

แบบสอบถาม 1/2

แบบสอบถาม (Questionnaire)

ข้อมูลผู้กรอกแบบสอบถาม:

ชื่อ นายอภิสิทธิ์ นามสกุล กองสุวรรณ ตำแหน่ง พนักงานออกแบบ/เขียนแบบ

หน่วยงาน / บริษัท เอ เอ็ม พี เมทัลเวอคส์ จำกัด

รายละเอียดของแบบสอบถาม:

~ แบบสอบถามนี้มี 2 หน้า ประกอบด้วยคำถามทั้งหมด 8 ข้อ

~ ในแต่ละคำถาม จะพิจารณาถึงระดับความพึงพอใจที่ได้รับจากการใช้โปรแกรม ZAmpPunch.arx โดยมีการให้ระดับคะแนนดังนี้

คะแนนระดับ	10	หมายถึง	ดีที่สุด
คะแนนระดับ	9	หมายถึง	ดีมาก
คะแนนระดับ	8	หมายถึง	ระหว่างดีมากถึงค่อนข้างดี
คะแนนระดับ	7	หมายถึง	ค่อนข้างดี
คะแนนระดับ	6	หมายถึง	ระหว่างค่อนข้างดีถึงพอใช้ได้
คะแนนระดับ	5	หมายถึง	พอใช้ได้
คะแนนระดับ	4	หมายถึง	ระหว่างพอใช้ได้ถึงต้องปรับปรุงค่อนข้างด่วน
คะแนนระดับ	3	หมายถึง	ต้องปรับปรุงแก้ไขค่อนข้างด่วน
คะแนนระดับ	2	หมายถึง	ระหว่างต้องปรับปรุงแก้ไขค่อนข้างด่วนถึงด่วนมาก
คะแนนระดับ	1	หมายถึง	ต้องปรับปรุงแก้ไขด่วนมาก

แบบสอบถาม 2/2

กรุณาทำเครื่องหมาย O ล้อมรอบคะแนนที่ท่านพิจารณาในแต่ละคำถามต่อไปนี้

คำถาม	ระดับความพึงพอใจที่ได้รับ
1 ความง่ายในการสร้าง Drawing ที่เป็นไปตามข้อกำหนดของซอฟต์แวร์	1 2 3 4 5 (6) 7 8 9 10
2 ความง่ายต่อการฝึกใช้ซอฟต์แวร์	1 2 3 4 5 6 7 (8) 9 10
3 ความง่ายในการใช้งานซอฟต์แวร์เพื่อช่วยสร้างรหัสโปรแกรม G Code	1 2 3 4 5 6 (7) 8 9 10
4 ความเหมาะสมกับผลิตภัณฑ์ที่ลูกค้าสั่งมา	1 2 3 4 5 6 (7) 8 9 10
5 ความสามารถและศักยภาพของซอฟต์แวร์	1 2 3 4 5 (6) 7 8 9 10
6 การลดภาระการทำงานเกี่ยวกับการสร้างรหัสโปรแกรม G Code	1 2 3 4 5 6 7 (8) 9 10
7 ความพอใจที่ได้ใช้ซอฟต์แวร์เพื่อนำเข้ามาช่วยทำงาน	1 2 3 4 5 6 (7) 8 9 10

ความเห็นอื่น ๆ ลดขั้นตอนการสร้างโปรแกรมและสร้างแบบให้น้อยลงอีกก็จะดี

แบบสอบถาม 1/2

แบบสอบถาม (Questionnaire)

ข้อมูลผู้กรอกแบบสอบถาม:

ชื่อ นายสราวุธ นามสกุล บุญวัฒน์ ตำแหน่ง พนักงานออกแบบ/เขียนแบบ

หน่วยงาน / บริษัท เอ เอ็ม พี เมทัลเวอคส์ จำกัด

รายละเอียดของแบบสอบถาม:

~ แบบสอบถามนี้มี 2 หน้า ประกอบด้วยคำถามทั้งหมด 8 ข้อ

~ ในแต่ละคำถาม จะพิจารณาถึงระดับความพึงพอใจที่ได้รับจากการใช้โปรแกรม ZAmpPunch.arx โดยมีการให้ระดับคะแนนดังนี้

คะแนนระดับ	10	หมายถึง	ดีที่สุด
คะแนนระดับ	9	หมายถึง	ดีมาก
คะแนนระดับ	8	หมายถึง	ระหว่างดีมากถึงค่อนข้างดี
คะแนนระดับ	7	หมายถึง	ค่อนข้างดี
คะแนนระดับ	6	หมายถึง	ระหว่างค่อนข้างดีถึงพอใช้ได้
คะแนนระดับ	5	หมายถึง	พอใช้ได้
คะแนนระดับ	4	หมายถึง	ระหว่างพอใช้ได้ถึงต้องปรับปรุงค่อนข้างด่วน
คะแนนระดับ	3	หมายถึง	ต้องปรับปรุงแก้ไขค่อนข้างด่วน
คะแนนระดับ	2	หมายถึง	ระหว่างต้องปรับปรุงแก้ไขค่อนข้างด่วนถึงด่วนมาก
คะแนนระดับ	1	หมายถึง	ต้องปรับปรุงแก้ไขด่วนมาก

แบบสอบถาม 2/2

กรุณาทำเครื่องหมาย O ล้อมรอบคะแนนที่ท่านพิจารณาในแต่ละคำถามต่อไปนี้

คำถาม	ระดับความพึงพอใจที่ได้รับ
1 ความง่ายในการสร้าง Drawing ที่เป็นไปตามข้อกำหนดของซอฟต์แวร์	1 2 3 4 5 6 (7) 8 9 10
2 ความง่ายต่อการฝึกใช้ซอฟต์แวร์	1 2 3 4 5 6 7 8 (9) 10
3 ความง่ายในการใช้งานซอฟต์แวร์เพื่อช่วยสร้างรหัสโปรแกรม G Code	1 2 3 4 5 6 7 8 9 (10)
4 ความเหมาะสมกับผลิตภัณฑ์ที่ลูกค้าสั่งมา	1 2 3 4 5 6 7 (8) 9 10
5 ความสามารถและศักยภายของซอฟต์แวร์	1 2 3 4 5 6 (7) 8 9 10
6 การลดภาระการทำงานเกี่ยวกับการสร้างรหัสโปรแกรม G Code	1 2 3 4 5 6 7 8 (9) 10
7 ความพอใจที่ได้ใช้ซอฟต์แวร์เพื่อนำเข้ามาช่วยทำงาน	1 2 3 4 5 6 7 8 (9) 10

ความเห็นอื่น ๆ _____

_____

_____

แบบสอบถาม 1/2

แบบสอบถาม (Questionnaire)

ข้อมูลผู้กรอกแบบสอบถาม:

ชื่อ นายศุภเลิศ นามสกุล แก้วเงิน ตำแหน่ง พนักงานออกแบบ/เขียนแบบ

หน่วยงาน / บริษัท เอ เอ็ม พี เมทัลเวอคส์ จำกัด

รายละเอียดของแบบสอบถาม:

- แบบสอบถามนี้มี 2 หน้า ประกอบด้วยคำถามทั้งหมด 8 ข้อ
- ในแต่ละคำถาม จะพิจารณาถึงระดับความพึงพอใจที่ได้รับจากการใช้โปรแกรม ZAmpPunch.arx โดยมีการให้ระดับคะแนนดังนี้

คะแนนระดับ	10	หมายถึง	ดีที่สุด
คะแนนระดับ	9	หมายถึง	ดีมาก
คะแนนระดับ	8	หมายถึง	ระหว่างดีมากถึงค่อนข้างดี
คะแนนระดับ	7	หมายถึง	ค่อนข้างดี
คะแนนระดับ	6	หมายถึง	ระหว่างค่อนข้างดีถึงพอใช้ได้
คะแนนระดับ	5	หมายถึง	พอใช้ได้
คะแนนระดับ	4	หมายถึง	ระหว่างพอใช้ได้ถึงต้องปรับปรุงค่อนข้างด่วน
คะแนนระดับ	3	หมายถึง	ต้องปรับปรุงแก้ไขค่อนข้างด่วน
คะแนนระดับ	2	หมายถึง	ระหว่างต้องปรับปรุงแก้ไขค่อนข้างด่วนถึงด่วนมาก
คะแนนระดับ	1	หมายถึง	ต้องปรับปรุงแก้ไขด่วนมาก

แบบสอบถาม 2/2

กรุณาทำเครื่องหมาย O ล้อมรอบคะแนนที่ท่านพิจารณาในแต่ละคำถามต่อไปนี้

คำถาม	ระดับความพึงพอใจที่ได้รับ
1 ความง่ายในการสร้าง Drawing ที่เป็นไปตามข้อกำหนดของซอฟต์แวร์	1 2 3 4 5 6 (7) 8 9 10
2 ความง่ายต่อการฝึกใช้ซอฟต์แวร์	1 2 3 4 5 6 7 (8) 9 10
3 ความง่ายในการใช้งานซอฟต์แวร์เพื่อช่วยสร้างรหัสโปรแกรม G Code	1 2 3 4 5 (6) 7 8 9 10
4 ความเหมาะสมกับผลิตภัณฑ์ที่ลูกค้าสั่งมา	1 2 3 4 5 6 (7) 8 9 10
5 ความสามารถและศักยภาพของซอฟต์แวร์	1 2 3 4 5 (6) 7 8 9 10
6 การลดภาระการทำงานเกี่ยวกับการสร้างรหัสโปรแกรม G Code	1 2 3 4 5 6 7 8 (9) 10
7 ความพอใจที่ได้ใช้ซอฟต์แวร์เพื่อนำเข้ามาช่วยทำงาน	1 2 3 4 5 6 7 (8) 9 10

ความเห็นอื่น ๆ _____

_____

_____

ประวัติผู้วิจัย

นายนราธิป วีระกิจพานิช เกิดวันที่ 24 ตุลาคม พ.ศ. 2520 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรีวิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมโลหการ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2540 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต ภาควิศวกรรมอุตสาหการ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2541