เอกสารอ้างอิง

1. Anilam Electronics Corporation. CRUSADER PROGRAMMING MANUAL. November 1984.
2. Anilam Electronics Europe Ltd. OEM DESIGN,INSTALLATION AND SERVICE MANUAL. England, August 1985.
3. Autodesk Inc. The AutoCADTM Drafting Package: User Guide. April 1985.
4. Pressman, Roger S., and Williams, John E. Numerical Control and Computer Aided Manufacturing. John Wiley & Sons, Inc., New York, 1977.
5. Paul, Richard P. ROBOT MANIPULATORS: MATHEMATICS,PROGRAMMING, AND CONTROL. MIT Press., USA, 1981.
6. Koren, Yoram. Computer Control of Manufacturing Systems. Chong Moh Offset Printing Pte. Ltd., Singapore, 1985.
7. Baumeisiter, T., Avallone A., and Baumeister III, T. Marks' Standard Handbook for Mechanical Engineers. McGraw-Hill Book Co., New York, USA, 1978.

ภาคผนวก ก

โปรแกรมวาดแบบ

ออโตแคท (AutoCAD) เป็นโปรแกรมสำเร็จรูปโปรแกรมหนึ่งที่ช่วยในการเขียนแบบหรือภาพได้อย่างมีประสิทธิภาพ เราสามารถจะทำการแก้ไขส่วนใดส่วนหนึ่งของแบบที่เขียนขึ้นได้โดยไม่ต้องทำการเขียนขึ้นใหม่ทั้งหมด และยังสามารถนำแบบที่เขียนขึ้นเป็นส่วนๆ มาประกอบกันได้ง่าย

<u>คำสั่งของโปรแกรม ออโตแคท ที่ใช้ในการเขียนแบบสำหรับเครื่อง ซีเอ็นซี</u>

คำสั่งที่ใช้ในโปรแกรมสำเร็จรูป ออโตแคท นั้น เราสามารถที่จะแบ่งคำสั่งพื้นฐานที่จำเป็นในการเขียนแบบ เพื่อเป็นข้อมูลสำหรับเครื่อง ซีเอ็นซี ได้เป็น 4 หมวดใหญ่ๆ ดังนี้

1. การเขียนภาพ (Draw)
2. การแสดงภาพ (Display)
3. การแก้ไขภาพ (Edit)
4. ยูทิลิตี้ (Utility)

1. <u>การเขียนภาพ</u> (Draw)

ชุดคำสั่งสำหรับการเขียนภาพจะประกอบด้วยคำสั่งที่ใช้ในการลากเส้น เขียนส่วนโค้ง เพื่อให้เกิดภาพ หรือ รูปเขียนแบบ ที่มีขนาดตามต้องการ

1.1 <u>คำสั่ง LINE</u> เป็นคำสั่งที่ใช้ในการเขียนเส้นตรง โดยกำหนดจุดเริ่มต้นและจุดสิ้นสุดของเส้นตรงนั้นๆ

1.2 <u>คำสั่ง CIRCLE</u> เป็นคำสั่งที่ใช้ในการเขียนวงกลม โดยกำหนดจุดศูนย์กลางและความยาวของรัศมี หรือ ความยาวของเส้นผ่านศูนย์กลางของวงกลม

1.3 <u>คำสั่ง ARC</u> เป็นคำสั่งที่ใช้ในการเขียนส่วนโค้งของวงกลม มีวิธีการกำหนดการเขียนส่วนโค้งได้หลายวิธี ดังนี้

1.3.1 กำหนดจุด 3 จุดบนเส้นโค้ง

1.3.2 กำหนดจุดเริ่มต้น จุดศูนย์กลาง และจุดสิ้นสุดของส่วนโค้ง

1.3.3 กำหนดจุดเริ่มต้น จุดศูนย์กลาง และมุมรองรับส่วนโค้ง

1.3.4 กำหนดจุดเริ่มต้น จุดศูนย์กลาง และความยาวของคอร์ด

1.3.5 กำหนดจุดเริ่มต้น จุดสิ้นสุด และรัศมีของส่วนโค้ง

1.3.6 กำหนดจุดเริ่มต้น จุดสิ้นสุด และมุมรองรับส่วนโค้ง

1.3.7 กำหนดจุดเริ่มต้น จุดสิ้นสุด และมุมของเส้นโค้งที่ออกจากจุดเริ่มต้น

1.4 คำสั่ง PLINE เป็นคำสั่งที่ใช้ในการเขียนเส้นตรงหรือเส้นโค้งต่อกันไป คอมพิวเตอร์จะถือว่ารูปเขียนแบบที่เขียนขึ้นด้วยคำสั่งนี้เป็นเพียงรูปเดียว การกำหนดตำแหน่งสำหรับการเขียนเส้น polyline มีลักษณะเดียวกันกับคำสั่ง LINE และ ARC ข้อดีประการหนึ่งของการเขียนรูปด้วยคำสั่งนี้ ก็คือ เราสามารถทำ curve fitting

2. การแสดงภาพ (Display)

เราสามารถควบคุมการย่อ ขยายภาพหรือเลื่อนภาพไปยังบริเวณที่ต้องการได้ โดยที่ขนาดและมิติของภาพหรือแบบนั้นไม่มีการเปลี่ยนแปลง คำสั่งที่เกี่ยวกับการแสดงภาพมีดังนี้

2.1 คำสั่ง ZOOM เป็นคำสั่งที่ใช้ในการย่อ ขยายภาพ เหมือนกับเลนซ์ซูมของกล้องถ่ายรูป เราสามารถเลือกการย่อ ขยายภาพได้ดังนี้

2.1.1 ขยายภาพทั้งหมด

2.1.2 ขยายภาพที่มีอยู่ทั้งหมดให้บรรจุเต็มจอภาพ

2.1.3 ย่อ ขยายภาพโดยกำหนดจำนวนเท่าที่ต้องการย่อ ขยายภาพ

2.1.4 ย่อ ขยายภาพโดยกำหนดกรอบภาพที่ต้องการ

2.1.5 ย่อ ขยายภาพโดยกำหนดจุดศูนย์กลางของภาพและความสูงจากจุดศูนย์กลางของภาพที่กำหนด

2.2 คำสั่ง PAN เป็นคำสั่งที่ใช้ในการเลื่อนภาพให้ปรากฏบนจอภาพตามส่วนที่ต้องการดู โดยกำหนดจุดอ้างอิง และระยะทางที่เราต้องการเลื่อนภาพไป หรืออาจจะกำหนดเป็นค่าพิกัดแทนก็ได้

2.3 คำสั่ง REDRAW เป็นคำสั่งที่ให้เขียนภาพบนจอใหม่ พร้อมกับลบสิ่งที่ไม่ต้องการบนจอภาพออก

2.4 คำสั่ง LAYER เป็นคำสั่งที่ใช้ในการกำหนด ชั้น สี และลักษณะของเส้นที่จะใช้กับภาพที่เขียนขึ้น

3. การแก้ไขภาพ (Edit)

เราสามารถทำการแก้ไขหรือคัดลอกภาพหรือแบบที่ต้องการได้ โดยไม่ต้องทำการเขียนใหม่ คำสั่งที่ใช้ในการแก้ไขภาพมีดังนี้

3.1 คำสั่ง ERASE เป็นคำสั่งที่ใช้ในการลบส่วนของภาพที่ไม่ต้องการออกไป โดยที่เราสามารถจะเลือกลบทีละส่วนหรือหลายส่วนพร้อมๆ กันก็ได้

3.2 คำสั่ง OOPS เป็นคำสั่งที่ใช้ในการเรียกส่วนของภาพที่เพิ่งจะทำการลบทิ้งไปกลับคืนมา คำสั่งนี้จะใช้เรียกส่วนของภาพที่ลบไปกลับคืนมาก่อนหน้าการลบครั้งหลังสุดไม่ได้

3.3 คำสั่ง MOVE เป็นคำสั่งที่ใช้ในการเคลื่อนย้ายส่วนของภาพจากตำแหน่งเดิมไปยังตำแหน่งใหม่ โดยกำหนดส่วนของภาพที่ต้องการเคลื่อนย้าย ซึ่งอาจจะเป็นเพียงส่วนเดียวหรือหลายส่วนของภาพก็ได้ จากนั้นจึงกำหนดจุดอ้างอิงและระยะทางที่ต้องการเคลื่อนย้ายไป หรืออาจจะกำหนดเป็นค่าพิกัดแทนก็ได้เช่นกัน ค่าพิกัดของภาพที่ได้ภายหลังการใช้คำสั่งนี้จะมีการเปลี่ยนแปลงไปจากภาพเดิม

3.4 คำสั่ง COPY เป็นคำสั่งที่ใช้ในการคัดลอกส่วนของภาพ วิธีการใช้คำสั่งนี้เหมือนกับการใช้คำสั่ง MOVE แต่รูปหรือภาพที่เป็นต้นแบบไม่ได้หายไปเหมือนกับคำสั่ง MOVE

3.5 คำสั่ง MIRROR เป็นคำสั่งที่ใช้ในการคัดลอกส่วนของภาพเช่นเดียวกับคำสั่ง COPY แต่แตกต่างกันตรงที่ ภาพที่ได้จากการคัดลอกด้วยคำสั่ง MIRROR จะได้ภาพเหมือนกับมองในกระจกเงา

3.6 คำสั่ง CHANGE เป็นคำสั่งที่ใช้ในการเปลี่ยนทิศทางและตำแหน่งของภาพ หรืออาจจะเปลี่ยนเลเยอร์ (Layer) ซึ่งเป็นชั้นที่ภาพนั้นอยู่ ไปยังอีกเลเยอร์หนึ่งก็ได้

3.7 คำสั่ง BREAK เป็นคำสั่งที่ใช้ในการลบและแยกส่วนส่วนของภาพที่ไม่ต้องการออกไป เช่น การแยกเส้นตรงออกเป็นสองส่วน หรือ การเปลี่ยนวงกลมให้เป็นส่วนโค้ง เราสามารถทำได้โดยเลือกภาพที่ต้องการลบ แล้วกำหนดจุดเริ่มต้นและจุดสิ้นสุดในกรณีที่เป็นเส้นโค้ง การลบของภาพจะกระทำในทิศทางทวนเข็มนาฬิกา

3.8 คำสั่ง FILLET เป็นคำสั่งที่ใช้ในการเชื่อมต่อเส้นที่ต่อกันเป็นมุมด้วยส่วนโค้ง โดยจะสร้างส่วนโค้งให้สัมผัสกับเส้นทั้งสองและปรับระยะปลายเส้นให้พอดี

3.9 คำสั่ง CHAMFER เป็นคำสั่งที่ใช้ในการตัดมุมของเส้นสองเส้นที่ต่อกัน โดยกำหนดระยะห่างจากมุมที่ต้องการในแต่ละเส้น แล้วจึงทำการเลือกเส้นทั้งสอง

3.10 คำสั่ง PEDIT เป็นคำสั่งที่ใช้ในการแก้ไขเส้นที่เขียนขึ้นด้วยคำสั่ง PLINE เราสามารถแก้ไขเส้นดังกล่าวได้หลายวิธีดังนี้

3.10.1 แยกเส้น polyline ออกเป็น 2 ส่วน

3.10.2 ต่อเส้น polyline สองเส้นให้เป็นเส้นเดียวกัน

3.10.3 เอาส่วนโค้งอยู่ระหว่างเส้นตรงหรือเส้นโค้งออก

3.10.4 ทำให้เส้นที่ต่อกันเป็นมุมกลายเป็นเส้นที่ต่อกันเป็นเส้นโค้ง

3.10.5 เปลี่ยนขนาดความหนาของเส้น

4. ยูทิลิตี้ (Utility)

ยูทิลิตี้ เป็นชุดคำสั่งที่จะช่วยให้การดำเนินงานเขียนภาพหรือแบบกระทำได้อย่างมีประสิทธิภาพมากยิ่งขึ้น ชุดคำสั่งที่ใช้มีดังนี้

4.1 คำสั่ง END เป็นคำสั่งที่จะทำการบันทึกรูปเขียนแบบที่เขียนขึ้นลงในแผ่นเก็บข้อมูลก่อนที่จะกลับเข้าสู่รายการหลัก (Main Menu) ในกรณีที่มีชื่อไฟล์ (File) นั้นอยู่แล้ว เครื่องก็จะทำการเปลี่ยนชื่อไฟล์ที่มีอยู่เดิมให้มีชนิดของไฟล์ (extension file) ที่ลงท้ายด้วย '.BAK' ถ้าชื่อของไฟล์นั้นมีชนิดของไฟล์ที่ลงท้ายด้วย '.BAK' อยู่แล้ว เครื่องก็จะทำการลบไฟล์เดิมนั้นก่อน

4.2 คำสั่ง QUIT เป็นคำสั่งที่ใช้สำหรับการยกเลิกการทำงานของโปรแกรมเขียนภาพ โดยไม่ทำการบันทึกข้อมูลที่ได้ทำการแก้ไขลงในแผ่นเก็บข้อมูล

4.3 คำสั่ง SAVE เป็นคำสั่งที่ใช้สำหรับการบันทึกข้อมูลของภาพที่ปรากฏบนจอภาพลงบนแผ่นเก็บข้อมูล และยังสามารถทำการแก้ไขต่อไปได้ ชื่อไฟล์ที่ใช้สำหรับการบันทึกลงบนแผ่นเก็บข้อมูลอาจจะเป็นชื่อเดิมหรือชื่อใหม่ก็ได้

4.4 คำสั่ง DXFOUT เป็นคำสั่งที่ใช้ในการบันทึกข้อมูลของภาพลงบนแผ่นเก็บข้อมูล โดยให้เก็บข้อมูลอยู่ในรูปแบบที่เป็นมาตรฐาน

4.5 คำสั่ง BLOCK เป็นคำสั่งที่ใช้ในการเก็บภาพที่ต้องการไว้ในหน่วยความจำเพื่อรอการเรียกใช้ต่อไป คำสั่ง BLOCK จะต้องใช้คู่กับคำสั่ง INSERT เสมอ เมื่อต้องการนำภาพที่เก็บไว้ด้วยคำสั่ง BLOCK มาใช้งาน เราสามารถทำการย่อ ขยายขนาดได้ หรือนำภาพที่เขียนไว้เป็นส่วนๆ มาประกอบกันได้ แต่ข้อจำกัดของการใช้ภาพที่ได้จากคำสั่ง BLOCK ก็คือ เราไม่สามารถทำการแก้ไขภาพหรือแบบที่อยู่ในแต่ละบล็อคได้ เพราะเครื่องจะถือว่าเป็นภาพหรือวัตถุเดียว

4.6 คำสั่ง WBLOCK เป็นคำสั่งที่ใช้ในการบันทึกภาพหรือแบบที่เก็บด้วยคำสั่ง BLOCK ลงบนแผ่นเก็บข้อมูลเพื่อสามารถนำภาพหรือแบบนั้นไปใช้กับไฟล์เขียนแบบอื่นๆ

4.7 คำสั่ง OSNAP เป็นคำสั่งที่จะช่วยในการเขียนภาพหรือแบบให้มีความถูกต้อง ผู้ใช้ไม่ต้องทำการคำนวนเพื่อค้นหาตำแหน่งที่ต้องการ โดยมีเงื่อนไขว่าภาพหรือแบบที่ต้องการค้นหานั้นจะต้องอยู่ในกรอบสี่เหลี่ยมเล็กๆ ที่เกิดขึ้นมาเมื่อใช้คำสั่งนี้ เราสามารถใช้หลายๆ คำสั่งได้พร้อมๆ กัน โดยคำสั่งที่อยู่ข้างหน้าจะมีความสำคัญกว่าคำสั่งที่ตามหลังมา

4.7.1 คำสั่ง NEA (NEAREST) เป็นคำสั่งที่ใช้ในการค้นหาตำแหน่งของภาพที่อยู่ใกล้ที่สุด

4.7.2 คำสั่ง END (ENDPOINT) เป็นคำสั่งที่ใช้ในการค้นหาตำแหน่งจุดปลายของภาพ

4.7.3 คำสั่ง MID (MIDPOINT) เป็นคำสั่งที่ใช้ในการค้นหาตำแหน่งที่เป็นจุดกึ่งกลางของเส้น

4.7.4 คำสั่ง CEN (CENTER) เป็นคำสั่งที่ใช้ในการหาจุดศูนย์กลางของส่วนโค้งหรือวงกลม

4.7.5 คำสั่ง NOD (NODE) เป็นคำสั่งที่จะค้นหาจุดแหลมของส่วนของภาพ

4.7.6 คำสั่ง QUA (QUADRANT) เป็นคำสั่งที่ใช้ในการค้นหาควอดซ์แรนท์ที่ใกล้ที่สุดของส่วนโค้งหรือวงกลม

4.7.7 คำสั่ง INT (INTERSECTION) เป็นคำสั่งที่ใช้ในการค้นหาจุดตัดของส่วนของภาพ

4.7.8 คำสั่ง INS (INSERT) เป็นคำสั่งที่ใช้ในการค้นหาตำแหน่งอ้างอิงของภาพที่ได้จากการใช้คำสั่ง INSERT

4.7.9 คำสั่ง PER (PERPENDICULAR) เป็นคำสั่งที่ใช้ในการค้นหาตำแหน่งที่จะลากเส้นตรงมาตั้งฉากเส้นตรงหรือส่วนโค้งที่ต้องการ

4.7.10 คำสั่ง TAN (TANGENT) เป็นคำสั่งที่ใช้ในการลากเส้นที่ต้องการให้สัมผัสกับส่วนโค้งหรือวงกลมที่กำหนด

จากชุดคำสั่งที่ได้กล่าวมานี้เป็นเพียงบางส่วนที่เพียงพอจะใช้ในการเขียนแบบเพื่อสั่งงานให้กับเครื่อง ซีเอ็นซี

<u>รูปแบบมาตรฐานของข้อมูลที่ได้จากโปรแกรม ออโตแคท</u>

นอกจากโปรแกรม ออโตแคท จะเก็บข้อมูลเป็นไบนารี่โคด (binary code) แล้วออโตแคท ยังสามารถเก็บข้อมูลดังกล่าวให้อยู่ในรูปแบบที่เป็นมาตรฐานด้วยคำสั่ง DXFOUT ข้อมูลที่ได้จากคำสั่งนี้จะเป็นไปตามรหัสอัสกี (ASCII code) โครงสร้างของรูปแบบมาตรฐานที่ได้จากโปรแกม ออโตแคท แบ่งออกได้เป็น 5 ส่วน คือ

1. <u>ส่วนของข้อมูลทั่วไป</u> (HEADER section)

ในส่วนนี้เป็นส่วนที่ใช้กำหนดค่าพารามิเตอร์ต่างๆ ให้กับตัวแปรของโปรแกรม เช่น กำหนดขนาดของหัวลูกศรที่ใช้บอกมิติ ชื่อของแบบตัวอักษรที่ใช้ เป็นต้น

2. ส่วนของตารางรายชื่อ (TABLES section)

ในส่วนนี้เป็นส่วนที่เก็บรายการชื่อต่างๆ ซึ่งถูกกำหนดขึ้นโดยผู้ใช้ แบ่งออกเป็น 4 ส่วน

2.1 ตารางชื่อของลักษณะเส้น
2.2 ตารางชื่อของชั้น
2.3 ตารางชื่อของแบบตัวอักษร
2.4 ตารางชื่อส่วนของภาพที่จะเรียกกลับมาดู

3. ส่วนของภาพชุด (BLOCKS section)

ส่วนของภาพชุดเป็นส่วนของภาพที่เขียนขึ้นมาแล้วเก็บไว้ในหน่วยความจำ เพื่อรอการเรียกใช้

4. ส่วนขององค์ประกอบภาพ (ENTITIES section)

ส่วนขององค์ประกอบเป็นส่วนของการวาดแบบที่เขียนขึ้น เพื่อใช้ในการกำหนดรูปลักษณะของแบบ

5. ส่วนของการจบไฟล์ (EOF section)

เป็นส่วนท้ายสุดที่บอกถึงการสิ้นสุดของไฟล์ข้อมูลของรูปแบบมาตรฐานที่ได้จากโปรแกรม ออโตแคท

ข้อมูลของรูปแบบมาตรฐานที่ใช้กำหนดค่าให้กับตัวแปรของโปรแกรม ออโตแคท ประกอบด้วย รหัส 2 บรรทัด บรรทัดแรกเป็นกลุ่มรหัส (group code) ที่แสดงถึงชนิดของค่าที่ตามหลังมา ส่วนบรรทัดที่สองเป็น กลุ่มกำหนดค่า (group value) สำหรับการแบ่งกลุ่มรหัสนั้น ออโตแคท ได้แบ่งออกเป็น 3 กลุ่ม

ช่วงของกลุ่มรหัส	กลุ่มกำหนดค่า
0 9	อักขระต่างๆ
10 59	เลขทศนิยม
60 79	เลขจำนวนเต็ม

นอกจากกลุ่มรหัสจะแสดงถึงชนิดของกลุ่มกำหนดค่าที่จะตามมาแล้ว กลุ่มรหัสยังแสดงถึงความหมายของกลุ่มกำหนดค่าที่ตามมา กลุ่มรหัสตามตาราง ก.1 เหล่านี้จะให้ความหมายของกลุ่มกำหนดค่าเหมือนกันไม่ว่าจะอยู่ในส่วนใดของโปรแกรม ยกเว้นกลุ่มรหัสที่มีคำว่า "Fixed" ซึ่งจะนำไปใช้กับส่วนอื่นๆ ไม่ได้นอกจากที่กำหนด

ตารางที่ ก.1 กลุ่มรหัสและกลุ่มกำหนดค่าที่ใช้ในข้อมูลมาตรฐาน

Group code	Group value
0	Identifies the start of an entity, table entry, entry, or file separator. The text value that follows indicates which.
1	The primary text value for an entity.
2	A name; attribute tag, block name, etc.
3 5	Other textual or name values.
6	Line type name (fixed).
7	Text style name (fixed).
8	Layer name (fixed).
9	Variable name identifier (used only in HEADER section of the DXF file).
10	Primary X coordinate (start point of a line or Text entity, center of a Circle, etc.).
11 18	Other X coordinates.
20	Primary Y coordinate. 2n values always correspond to 1n values and immediately follow them in the file.
21 28	Other Y coordinates.
30	Primary Z coordinate. 3n values always correspond to 1n and 2n values and immediately follow them in the file.
31 36	Other Z coordinates (future).
38	This entity's elevation, if nonzero (fixed).
39	This entity's thickness, if nonzero (fixed).
40 48	Floating point values (text height, scale factors, etc.).
49	Repeated value multiple 49 groups may appear in one entity form variable length tables (such as the dash lengths in the LTYPE table). A 7x group always appears before the first 49 group to specify the table length.
50 58	Angles.
62	Color number (fixed)
66	"Entities follow" flag (fixed).
70 78	Integer values, such as repeat counts, flag bits, or modes.

สำหรับงานวิจัยเพื่อหาองค์ประกอบของแบบที่วาดขึ้นเพื่อใช้เขียน จีโคด ให้กับเครื่อง ซีเอ็นซี นั้น จะเกี่ยวข้องกับส่วนของภาพชุด และส่วนขององค์ประกอบภาพ การจัดเรียงข้อมูลในส่วนของชุดจะมีการจัดเรียงข้อมูลเหมือนกับส่วนขององค์ประกอบภาพ กลุ่มรหัสที่ใช้สำหรับวาดแบบ เพื่อใช้เขียน จีโคด มีดังนี้

1. LINE

10 พิกัดของจุดเริ่มต้นในทิศทาง X
20 พิกัดของจุดเริ่มต้นในทิศทาง Y
11 พิกัดของจุดสิ้นสุดในทิศทาง X
21 พิกัดของจุดสิ้นสุดในทิศทาง Y

2. CIRCLE

10 พิกัดของจุดศูนย์กลางในทิศทาง X
20 พิกัดของจุดศูนย์กลางในทิศทาง Y
40 รัศมีของวงกลม

3. ARC

10 พิกัดของจุดศูนย์กลางในทิศทาง X
20 พิกัดของจุดศูนย์กลางในทิศทาง Y
40 รัศมีของส่วนโค้ง
50 มุมเริ่มต้นของส่วนโค้งที่กระทำกับแนวระดับ
51 มุมสิ้นสุดของส่วนโค้งที่กระทำกับแนวระดับ

4. BLOCK

2 ชื่อส่วนของภาพชุด
70 block type flags
10 จุดอ้างอิงของภาพชุดในทิศทาง X
20 จุดอ้างอิงของภาพชุดในทิศทาง Y

5. INSERT

66 "Attributes follow" flag

2 ชื่อส่วนของภาพชุด

10 พิกัดที่นำส่วนของภาพชุดเข้ามาในทิศทาง X

20 พิกัดที่นำส่วนของภาพชุดเข้ามาในทิศทาง Y

41 ขนาดของการย่อขยายในทิศทาง X

42 ขนาดของการย่อขยายในทิศทาง Y

50 มุมที่หมุนรอบจุดที่นำส่วนของภาพชุดเข้ามา

6. POLYLINE

70 polyline flags ถ้าค่าที่ตามหลังมามีค่าเป็น 1 แสดงว่า polyline นั้นเป็นรูปปิด และถ้าค่าเป็น 2 แสดงว่ามีการทำ curve fitting

40 ความกว้างที่ตำแหน่งเริ่มต้นของเส้น polyline

41 ความกว้างที่ตำแหน่งสิ้นสุดของเส้น polyline

7. VERTEX

10 พิกัดในทิศทาง X

20 พิกัดในทิศทาง Y

40 ความกว้างที่ตำแหน่งเริ่มต้นของเส้น polyline

41 ความกว้างที่ตำแหน่งสิ้นสุดของเส้น polyline

42 ค่า tangent ของ หนึ่งในสี่ของมุมที่รองรับส่วนโค้ง ถ้ามีค่าเป็นลบ แสดงว่า มีทิศทางตามเข็มนาฬิกา ถ้ามีค่าเป็น 0 แสดงว่า เป็นเส้นตรง ถ้ามีค่าเป็น 1 แสดงว่าเป็นครึ่งวงกลม

70 vertex flags ถ้าค่าที่ตามหลังมามีค่าเป็น 1 แสดงว่า มีการเพิ่มจุด vertex เนื่องจากการทำ curve fitting และถ้ามีค่าเป็น 2 แสดงว่า เป็นการ curve fitting tangent defined

ตาราง ก-2 ลำดับโครงสร้างของข้อมูลมาตรฐาน

```
  0       (Begin HEADER section)
SECTION
  2
HEADER
          <<< Header variable items go here >>>
  0       (End HEADER section)
ENDSEC
  0       (Bengin TABLES section)
SECTION
  2
TABLES
  0
TABLE
  2       (Begin LTYPE table)
LTYPE
 70
          (Layer table maximum item count)
          <<< Layer table items go here >>>
  0       (End LTYPE table)
ENDTAB
  0
TABLE
  2       (Beging TEXT STYLE table)
STYLE
 70
          (Text style table maximum item count)
          <<< Text style table items go here >>>
  0       (End TEXT STYLE table)
ENDTAB
  0
TABLE
  2       (Begin VIEW table)
VIEW
 70
          (View table maximum item count)
          <<< View table items go here >>>
  0       (End VIEW table)
ENDTAB
```

ตาราง ก.2 ลำดับโครงสร้างของข้อมูลมาตรฐาน (ต่อ)

```
  0          (End TABLES section)
ENDSEC
  0          (Begin BLOCKS section)
SECTION
  2
BLOCKS
             <<< Block definition entities go here >>>
  0          (END BLOCKS section)
ENDSEC
  0          (Begin ENTITIES section)
SECTION
  2
ENTITIES
                  <<< Drawing entities go here >>>
  0
ENDSEC       (End ENTITIES section)
  0
EOF          (end of file)
```

ภาคผนวก ข

รายละเอียดของโปรแกรมที่พัฒนาขึ้นเพื่อใช้ในการสร้างรหัส จีโคด

โปรแกรมที่พัฒนาขึ้นนี้ถูกเขียนด้วยโปรแกรมภาษา ซี รายละเอียดของโปรแกรมแบ่งออกได้เป็น 2 ส่วนใหญ่ คือ

1. ค่าคงที่ที่ได้กำหนดขึ้นเพื่อใช้สำหรับโปรแกรมนี้
2. โปรแกรมที่ใช้ในการคำนวณหาทางเดินของจุดศูนย์กลางของคัทเตอร์ และการเขียนรหัส จีโคด

1. ค่าคงที่ที่ได้กำหนดขึ้นเพื่อใช้สำหรับโปรแกรมนี้

```
#define START_END "%"     /* Code for starting and ending transfer */
#define STATE "N"         /* Statement No. */
#define RAPID "G0"        /* Rapid Positioning */
#define LINEAR "G1"       /* Linear Feed */
#define CIR_CLOCK "G2"    /* Circular Motion Clockwise */
#define CIR_COUNTER "G3"  /* Circular Motion Counterclockwise */
#define STOP "G4"         /* Program Stop */
#define RAPIDs "G10"      /* Rapid polar move with spindle at polar
                             center */
#define FEEDs "G11"       /* Feed polar move with spindle at polar
                             center */
#define RAPIDc "G12"      /* Rapid polar move with center specified */
#define FEEDc "G13"       /* Feed polar move with center specified */
#define SPEC29 "G29"      /* Special Function */
#define COMPENd "G40"     /* Deactivates Cutter Dia. Compensation */
#define COMPENl "G41"     /* Activates Cutter Dia. Compensation Left */
#define COMPENr "G42"     /* Activates Cutter Dia. Compensation Right
                             */
#define POLARr "G51"      /* Polar Rotation */
#define POLARd "G52"      /* Deactivates Polar Rotation */
#define SCALE "G53"       /* Activates Scaling */
#define SCALEd "G54"      /* Deactivates Scaling */
#define DIMENin "G70"     /* Dimension in Inch */
#define DIMENmm "G71"     /* Dimension in Millimeter */
#define HOLEm "G76"       /* Hole Milling */
#define POCKm "G77"       /* Circular Pocket Milling */
#define POCKr "G78"       /* Rectangular Pocket Milling */
#define Bolt "G79"        /* Bolt Hole Circle Drilling */
```

```
#define CAN80 "G80"       /* Deactivates Canned Cycles */
#define CAN81 "G81"       /* (Basic Drilling Cycle) Feeds in to depth
                             rapids out */
#define CAN82 "G82"       /* (Counter-boring of Spot-facing Cycle)
                             Feeds in to depth, dwells for programmed
                             time, the rapid out */
#define CAN83 "G83"       /* (Peck Drilling Cycle) Feed in, rapids
                             out,rapids in, feeds in, rapids out,
                             rapid return final depth is reached */
#define CAN85 "G85"       /* (Boring Cycle) Feeds in to depth and
                             feeds out */
#define CAN86 "G86"       /* (Boring in One Direction) Feeds in to
                             depth,program stop, rapids out, program
                             stop */
#define CAN87 "G87"       /* (Chip Breaking Cycle) Feeds in, retracts
                             .050, feeds in, retracts .050 until final
                             depth is reached */
#define CAN89 "G89"       /* (Flat Bottom Boring Cycle) Feeds in to
                             depth, dwells for a programmed time then
                             feeds out */
#define ABSOLUTE "G90"    /* Absolute Input */
#define INCREMENT "G91"   /* Incremental Input */
#define VAR01 "V01"       /* Sets X axis low software limit */
#define VAR02 "V02"       /* Sets X axis high software limit */
#define VAR03 "V03"       /* Sets Y axis low software limit */
#define VAR04 "V04"       /* Sets Y axis high software limit */
#define VAR05 "V05"       /* Sets Z axis low software limit */
#define VAR06 "V06"       /* Sets Z axis high software limit */
#define VAR11 "V11"       /* Polar center in the X axis for polar
                             moves, bolt circle, or scaling */
#define VAR12 "V12"       /* Polar center in the Y axis for polar
                             moves, bolt circle, or scaling */
#define VAR13 "V13"       /* Index angle used with polar rotation.
                             More detail look in "CRUSADER PROGRAMMING
                             MANUAL" in page 65 */
#define VAR14 "V14"       /* Radius for polar moves */
#define VAR15 "V15"       /* Angle for polar moves on angle of the
                             first hole in a bolt circle */
#define VAR16 "V16"       /* Angle of the last hole in a bolt circle */
#define VAR17 "V17"       /* Number of holes to be drilled in a bolt
                             circle */
#define VAR18 "V18"       /* Diameter of bolt circle, or hole mill
                             cycle */
#define VAR20 "V20"       /* Specifies feedrate for Z-axis moves in
                             Drilling Canned Cycles */
#define VAR21 "V21"       /* Specifies the height above the work piece
                             at which the tool begins and ends after
                             each hole is machined. This height must
                             be absolute */
#define VAR22 "V22"       /* Specifies dwell time when using G82 or
                             G89 */
#define VAR23 "V23"       /* Specifies the maximum amount of each
                             peck in Pecking cycles. This distance
                             must be given as a positive incremental
                             dimension */
```

```
#define VAR40 "V40"       /* Starting height in the Z axis for pocket
                             milling */
#define VAR41 "V41"       /* Length of the pocket in the X axis */
#define VAR42 "V42"       /* Width of the pocket in the Y axis */
#define VAR43 "V43"       /* Depth of the pocket in the Z axis */
#define VAR44 "V44"       /* Pocket corner radius */
#define VAR45 "V45"       /* Step over in the X and Y axis */
#define VAR46 "V46"       /* Maximum depth of cut in Z axis */
#define VAR47 "V47"       /* Stock left for a finish cut */
#define VAR48 "V48"       /* Finish pass feed rate */
#define VAR49 "V49"       /* Tool diameter */
#define CEN_X "I"         /* Arc Center Data X-Dimension */
#define CEN_Y "J"         /* Arc Center Data Y-Dimension */
#define CEN_Z "K"         /* Arc Center Data Z-Dimension */
#define DimX "X"          /* Coordinate in direction X */
#define DimY "Y"          /* Coordinate in direction Y */
#define DimZ "Z"          /* Coordinate in direction Z */
#define FEED "F"          /* Feed rate */
#define MISCELL "M"       /* Miscellaneous Command */
#define SPINDLE "S"       /* Spindle Speed */
#define TOOL "T"          /* Tool Command */
#define LOAD "L"          /* Load command */
#define  heading  "Version 1.0  (C) Copyright Control  lab.,Mech.
Dept.,Chula.1987."
#define  THESIS  "Application of Microcomputer Graphics for Control
of CNC Machine"
#define Pi 3.141592654
#define len 6
#define len1 7
#define len2 3
#define step 5
#define LimCha 15
#define MaxCha 21
#define Limit1 1E-9
#define Limit2 1E-5
#define Limit3 1E-3
#define Limit4 1E-4
#define nummax 200
#define max1 50
#define numb 30
#define l_num 100
#define c_num 100
#define a_num 100
#define p_num 100
#define b_num 200
#define maxlayer 50
#define space1 '\040'
#define LT '\311'
#define LB '\310'
#define HOR '\315'
#define VER '\272'
#define RT '\273'
#define RB '\274'
#define error1 "err1 : Drive x: and filename cannot exceed 10
characters."
```

```
#define error2 "err2 : Filename misses semicolon [:] at the second
character."
#define error3 "err3 : Select A:,B:,C: or D: for disk drive only."
#define error4 "err4 : No have filename for reading."
#define error5 "err5 : The 1st character must be alpha or nummeric."
#define error6 "err6 : Filename must be alpha, numeric, '-' or '_'."
#define error10 "err10 : There are some errors in your data!"
#define error11 "err11 : Divide by zero."
#define error12 "Cutter path begin at the nearest point from the
origin."
#define error13 "err13 : There is no any data for calculating a
cutter path."
#define error14 "err14 : There are some imaginary data."
#define error15 "err15 : Can't find the intersect point between two
arc."
#define nut 50
```

2. โปรแกรมที่ใช้ในการคำนวณหาทางเดินของจุดศูนย์กลางของคัทเตอร์ และการเขียนรหัส จีโคด

```
/*************************************************************/
/* This is a program for controlling Crusader II CNC Machine */
/*************************************************************/

#include <stdio.h>
#include <a:const.h>


FILE *rfpt,*sfpt,*wfpt;
int ncode;
int err7,err8,err9;
double datax[nummax],datay[nummax],dataxc[nummax],datayc[nummax];
char datac[nummax][len1],datad[nummax][len1],figure[numb][len];
double xcode;
double lxs[l_num],lys[l_num],lxe[l_num],lye[l_num];
double cirx[c_num],ciry[c_num],cirr[c_num],cirz[c_num];
double arcxc[a_num],arcyc[a_num],arcxs[a_num],arcys[a_num],
       arcxe[a_num],arcye[a_num];
double plx[p_num],ply[p_num],plxc[p_num],plyc[p_num];
double bxs[b_num],bys[b_num],bxc[b_num],byc[b_num],bxe[b_num],
       bye[b_num];
double binsx[b_num],binsy[b_num];
double xend,yend,xs_final,ys_final,xc_final,yc_final,xe_final,
       ye_final;
char t_final[len],d_final[len];
int polys[numb],polye[numb],datas[numb],datae[numb],binss[numb],
    binse[numb];
int lin,cin,ain,pin,pinl,bin,binl;
int ilin,icin,iain,ipin;
char arcd[a_num][len],pld[p_num][len],bd[b_num][len1],
     btype[b_num][len],btypel[b_num][len1];
char layer[maxlayer],ccode[maxlayer],bname[50][maxlayer];

main()
 {
 extern FILE *rfpt,*sfpt,*wfpt,*fopen();
 extern double atof(),xend,yend;
 extern int datas[],datae[],err9;
 extern char figure[][len];
 double m_diameter,m_depth,m_rate,tx,ty,tz,rx,ry,rz,d_rate,
        height,d_depth,scale,p_rate,XYstep,MZdep,Sfc,Ffc,
        lpxx[nut],lpyy[nut],rp[nut],pxc[nut],pyc[nut];
 char rfile[LimCha],wfile[LimCha],sfile[LimCha],direct[MaxCha],
      inout[MaxCha],dimension[LimCha],START_END[2],same[LimCha],
      compen[len],polar[len],can[len],Pock[LimCha],Tpock[LimCha];
 char data1[MaxCha],data2[MaxCha];
 int i,ii,num,index,no,result1,result2,Page,loop1,loop2,lin2,ain2;
 int begin,pnind;
```

```
loop1=1; loop2=1; Page=0;
strcpy(sfile,"STANDARD.STD");
ReadStandardFile(sfile,rfile,wfile,layer,direct,inout,compen,polar,
    dimension,can,&m_diameter,&m_depth,&m_rate,&tx,&ty,&tz,&rx,&ry,
    &rz,&d_rate,&height,same,&d_depth,&scale,&p_rate,&XYstep,
    &MZdep,&Sfc,&Ffc,Pock,Tpock);
crt_cls();  Frame(Page);  Menu(Page,0);
PrintAllData(rfile,sfile,wfile,layer,direct,inout,compen,polar,
    dimension,can,m_diameter,m_depth,m_rate,tx,ty,tz,rx,ry,rz,
    d_rate,height,same,d_depth,scale,Pock);
while (loop1 == 1) {
 while (loop2 == 1)
  RecieveData(&loop2,rfile,sfile,wfile,layer,direct,inout,compen,
      polar,dimension,can,&m_diameter,&m_depth,&m_rate,&tx,&ty,&tz,
      &rx,&ry,&rz,&d_rate,&height,same,&d_depth,&scale,&p_rate,
      &XYstep,&MZdep,&Sfc,&Ffc,Pock,Tpock);
 if (loop2 == -1) loop1=0;
 if (loop2 == 0) {
  if ((rfpt=fopen(rfile,"r")) == NULL) {
   crt_srcp(21,13,Page);
   printf("Couldn't open filename : %s",rfile); loop2=1;
 } if ((rfpt=fopen(rfile,"r")) != NULL) {
   num=0; index=0; no=step; result1=1; result2=1; loop2=1;
   lin=0; cin=0; ain=0; pin=0; pin1=0; bin=0; bin1=0;
   ilin=0; icin=0; iain=0; ipin=0; err9=0; begin=0;
   wfpt=fopen(wfile,"w"); crt_srcp(21,13,Page);
   printf("Reads data from given filename.");
   FindBeginningOfData(); StoreDataFromDiskToMemory();
   fclose(rfpt); ChangePolyLineDataintoLineOrArcData();
   if (cin > 0) {
    datas[index]=num; Drill1(same,d_depth);
    DrillData(&num,same); datae[index]=num;
    strcpy(figure[index],"NONE");
    ++index;  result2=0; ++begin;
  } lin2=lin; ain2=ain;
   if (lin > 0 || ain > 0) {
    while (result1 == 1) {
     LineDataNearOrigin(); ArcDataNearOrigin();
     datas[index]=num;
     FindDataNerarestOrigin(&num,figure[index],direct);
     if (err9 == 0) {
      SequenceData(&num); datae[index]=num;  ++index;
      AnotherData();
      if (lin <= 0 && ain <= 0)  result1=0;
} } } if (err9 == 0) {
    if (result1 != 1) result2=0;
    if (result2 != 1) {
     if(strncmp(Pock,"YES",strlen(Pock)) == 0)
      RectanPocketData(begin,index,&poind,lpxx,lpyy,rp,pxc,pyc);
     else
      NewCoordinate(dimension,begin,index,&num,direct,inout,
          m_diameter);
     if (err9 == 0) {
      lin=lin2; ain=ain2;
      strcpy(START_END,"%"); Space(21,13,Page,64);
```

```
        crt_srcp(21,13,Page); Scaling(num,rx,ry,scale);
        printf("Writes G code commands into diskette.");
        fprintf(wfpt,"%s\n",START_END);
        WriteGcode(index,poind,&no,compen,polar,dimension,can,d_rate,
         height,m_rate,m_depth,tx,ty,tz,rx,ry,rz,Pock,p_rate,XYstep,
         MZdep,Sfc,Ffc,m_diameter,Tpock,lpxx,lpyy,rp,pxc,pyc,d_depth);
        fprintf(wfpt,"%s\n",START_END);
        fclose(wfpt);
        Space(21,13,Page,64);
 } } } } } }
 if (err? == 1) fclose(wfpt);
 crt_cls(); printf("\n***** Good Luck ****\n\n");
 }

/***************** RecieveData() ******************/
/*                                                */
/* Changes some initial condition of CNC machine */
/* by receieving command from keyboard.          */
/*                                                */
/**************************************************/
RecieveData(loop,rfile,sfile,wfile,layer,direct,inout,compen,polar,
    dimen,can,dia,depth,rate,tx,ty,tz,rx,ry,rz,rate1,height,same,
    depth1,scale,Prate,XYstep,MZdep,Sfc,Ffc,Pock,Tpock)
char sfile[LimCha],rfile[LimCha],wfile[LimCha],layer[maxlayer],
     direct[LimCha],inout[LimCha],compen[len],polar[len],
     dimen[LimCha],can[len],same[LimCha],Pock[LimCha],Tpock[LimCha];
double *dia,*depth,*rate,*tx,*ty,*tz,*rx,*ry,*rz,*rate1,*height,
       *depth1,*scale,*Prate,*XYstep,*MZdep,*Sfc,*Ffc;
int *loop;
 {
 int pass,ind1,Page,Number; int loop3 = 0;
 char data1[MaxCha],data2[MaxCha],data3[MaxCha],data4[MaxCha],
      filevar[MaxCha],tfile[MaxCha];
 strcpy(data1,'\0'); strcpy(data2,'\0'); pass=0;
 Space(23,3,0,75);  crt_srcp(23,3,0);  printf("command :");
 InputData(data1,data2,data3,data4,0);
 ind1=strlen(data1);  Space(21,13,0,64);
 if (ind1 < 2) ind1=2;
 if (strncmp(data1,"NONE",ind1) != 0) {
  if (strncmp(data1,"SOURCE",ind1) == 0) {
   File(data1,data2,rfile); Print1Screen(6,3,25,0,rfile);
   pass=1;
} if (pass == 0) {
   if (strncmp(data1,"STANDARD",ind1) == 0) {
    strcpy(tfile,sfile); File(data1,data2,sfile);
    ReadStandardFile(sfile,rfile,wfile,layer,direct,inout,compen,
        polar,dimen,can,dia,depth,rate,tx,ty,tz,rx,ry,rz,rate1,
        height,same,depth1,scale,Prate,XYstep,MZdep,Sfc,Ffc,Pock,
        Tpock);
   if (strncmp(tfile,sfile,strlen(sfile)) != 0)
     PrintAllData(rfile,sfile,wfile,layer,direct,inout,compen,polar,
         dimen,can,*dia,*depth,*rate,*tx,*ty,*tz,*rx,*ry,*rz,*rate1,
         *height,same,*depth1,*scale,Pock);
 } } if (strncmp(data1,"OUTPUT",ind1) == 0) {
   File(data1,data2,wfile);  Print1Screen(6,54,76,0,wfile);
```

```
} if (strncmp(data1,"LAYER",ind1) == 0) {
  LayerFunction(data2,layer);  Print2Screen(11,17,38,0,layer);
  }
 if (strncmp(data1,"DIRECTION",ind1) == 0) {
  pass=1; DirectionFunction(data2,direct);
  Print2Screen(12,23,38,0,direct);
} if (strncmp(data1,"PATH",ind1) == 0) {
  In_Outside(data2,inout); Print2Screen(13,18,38,0,inout);
} if (strncmp(data1,"COMPENSATE",ind1) == 0) {
  Code("COMPENSATE",data2,compen);
  Print2Screen(17,46,48,0,compen);
} if (strncmp(data1,"POLAR",ind1) == 0) {
  Code("POLAR",data2,polar); Print2Screen(17,56,58,0,polar);
  pass = 1;
} if (pass == 0) {
  if (strncmp(data1,"DIMENSION",ind1) == 0) {
   Dimension(data2,dimen); Print2Screen(14,19,38,0,dimen);
 } } if (strncmp(data1,"CANNED",ind1) == 0) {
  Code("CANNED",data2,can); Print2Screen(17,66,68,0,can);
} if (strncmp(data1,"CUTTER",ind1) == 0) {
  AskFloat("Diameter of cutter",data2,dia);
  Print3Screen(15,22,38,0,*dia);
} if (strncmp(data1,"DEPTH",ind1) == 0) {
  AskFloat("Depth of cutter",data2,depth);
  Print3Screen(16,20,38,0,*depth);
} if (strncmp(data1,"FEED",ind1) == 0) {
  AskFloat("Feed rate of cutter",data2,rate);
  Print3Screen(17,16,38,0,*rate);
} if (strncmp(data1,"TOOL",ind1) == 0) {
  ToolChange(data2,data3,data4,tx,ty,tz);
  Print4Screen(18,20,38,0,*tx,*ty,*tz);
} if (strncmp(data1,"REFERENCE",ind1) == 0) {
  ReferencePosition(data2,data3,data4,rx,ry,rz);
  Print4Screen(19,20,38,0,*rx,*ry,*rz);
} if (strncmp(data1,"RATE",ind1) == 0) {
  AskFloat("Drilling feed rate",data2,rate1);
  Print3Screen(11,64,76,0,*rate1);
} if (strncmp(data1,"HEIGHT",ind1) == 0) {
  AskFloat("Height over object",data2,height);
  Print3Screen(12,64,76,0,*height);
} if (strncmp(data1,"SAME",ind1) == 0) {
  SameDepth(data2,same);  Print2Screen(13,65,76,0,same);  pass=1;
} if (pass == 0) {
  if (strncmp(data1,"DRILLING",ind1) == 0) {
   AskFloat("Drilling depth",data2,depth1);
   Print3Screen(14,60,76,0,*depth1);
} }  if (pass == 0) {
if (strncmp(data1,"POW",ind1) == 0) {
 DoPocket(data2,Pock);
 if (strncmp(Pock,"NO",strlen(Pock)) == 0)
  Print2Screen(15,63,76,0,Pock);
 if(strncmp(Pock,"YES",strlen(Pock)) == 0) {
  crt_cls(); Page = 0; loop3 = 1; Number = 1; Frame(Page);
  Menu(Page,Number);
```

```
    Print_PIData(rfile,sfile,wfile,*dia,*height,*Prate,*XYstep,
        *MZdep,*Sfc,*Ffc,*depth1,Tpock);
    while(loop3 == 1) {
     PagelRecieveData(dia,height,Prate,XYstep,MZdep,Sfc,Ffc,data1,
         data2,data3,data4,&ind1,depth1,Tpock);
     loop3 = 1;

     if(strncmp(data1,"RETURN",ind1) == 0) {
      crt_cls(); Page = 0; Number = 0;
      Frame(Page); Menu(Page,Number);
      PrintAllData(rfile,sfile,wfile,layer,direct,inout,compen,polar,
          dimen,can,*dia,*depth,*rate,*tx,*ty,*tz,*rx,*ry,*rz,*ratel,
          *height,same,*depth1,*scale,Pock);
      loop3 = 0;
    } if(strncmp(data1,"EXECUTE",ind1) == 0) {
     crt_cls(); Page = 0; Number = 0; Frame(Page); Menu(Page,Number);
       PrintAllData(rfile,sfile,wfile,layer,direct,inout,compen,polar,
          dimen,can,*dia,*depth,*rate,*tx,*ty,*tz,*rx,*ry,*rz,*ratel,
          *height,same,*depth1,*scale,Pock);
      loop3 = 0;
    } if(strncmp(data1,"QUIT",ind1)==0) loop3 = 0;
  } } pass = 0; }
 } if (pass == 0) {
    if (strncmp(data1,"SAVE",ind1) == 0)
     WriteStandardFile(sfile,rfile,wfile,layer,direct,inout,compen,
         polar,dimen,can,*dia,*depth,*rate,*tx,*ty,*tz,*rx,*ry,*rz,
         *ratel,*height,same,*depth1,*scale,*Prate,*XYstep,*MZdep,
         *Sfc,*Ffc,Pock,Tpock);
  } if (strncmp(data1,"SCALE",ind1) == 0) {
    AskFloat("Scale",data2,scale);
    Print3Screen(19,46,50,0,*scale);
  } if (strncmp(data1,"RUN",ind1) == 0) {
    system("xtalk.exe"); crt_cls(); Frame(0); Menu(0,0);
    PrintAllData(rfile,sfile,wfile,layer,direct,inout,compen,polar,
        dimen,can,*dia,*depth,*rate,*tx,*ty,*tz,*rx,*ry,*rz,*ratel,
        *height,same,*depth1,*scale,Pock);
  } if (strncmp(data1,"EXECUTE",ind1) == 0) *loop=0;
    if (strncmp(data1,"QUIT",ind1) == 0) *loop=-1;
 } }

/********************* Menu() ************************/
/*                                                   */
/* Shows the conditions of CNC machine on monitor */
/*                                                   */
/*****************************************************/
Menu(Page,Number)
int Page,Number;
 {
 crt_srcp(1,0,Page);    puts(THESIS);
 crt_srcp(3,0,Page);  puts(heading);
 crt_srcp(5,6,Page);  printf(" SOurce filename ");
 crt_srcp(5,30,Page);  printf(" STandard filename ");
 crt_srcp(5,57,Page);  printf(" OUtput filename ");
 if(Number == 0) {
  crt_srcp(7,25,Page);  printf(" Milling & Drilling  condition ");
```

```
  crt_srcp(11,4,Page);  printf("LAyer name :");
  crt_srcp(11,43,Page);  printf("drilling feed RAte :");
  crt_srcp(12,4,Page);  printf("cutter DIrection :");
  crt_srcp(12,43,Page);  printf("HEight over object :");
  crt_srcp(13,4,Page);  printf("cutter PAth :");
  crt_srcp(13,43,Page);  printf("drilling SAme depth :");
  crt_srcp(14,4,Page);  printf("DIMension in :");
  crt_srcp(14,43,Page);  printf("DRilling depth :");
  crt_srcp(15,4,Page);  printf("CUtter diameter :");
  crt_srcp(15,43,Page); printf("Pocket Making(POM):");
  crt_srcp(16,4,Page);  printf("cutting DEpth :");
  crt_srcp(16,43,Page);  printf("COmpensate");
  crt_srcp(16,55,Page);  printf("POlar");
  crt_srcp(16,62,Page);  printf("CAnned-cycle");
  crt_srcp(17,4,Page);  printf("FEed rate :");
  crt_srcp(18,4,Page);  printf("TOol position :");
  crt_srcp(18,43,Page);  printf("SCaling");
  crt_srcp(19,4,Page);  printf("REference pt. :");
  crt_srcp(19,44,Page); printf("1:");
 } if (Number == 1) {
  crt_srcp(9,25,Page); printf("Circular & rectang.Pocket Cond.");
  crt_srcp(11,4,Page); printf("CIRcu. or recTANg.(Tpock):");
  crt_srcp(12,4,Page); printf("roughing feed rate(PFE) :");
  crt_srcp(13,4,Page); printf("starting height(HE) :");
  crt_srcp(14,4,Page); printf("Max. XYstep over(XY) :");
  crt_srcp(15,4,Page); printf("Max. Zdepth per pass(MZ):");
  crt_srcp(16,4,Page); printf("Stock left Finish Cut(SFC):");
  crt_srcp(17,4,Page); printf("Fe.rate for Finish Cut(FFC):");
  crt_srcp(18,4,Page); printf("Tool Diameter(CU) :");
  crt_srcp(19,4,Page);printf("Pocket DepthZ(DR):");
  crt_srcp(19,44,Page); printf("RETurn to the root menu(RET):");
 } crt_srcp(21,4,Page);  printf("inform :");
 }

/****************** PrintAllData() ****************/
/*                                                */
/* Prints all data of initial condition of CNC */
/* machine on the monitor.                        */
/*                                                */
/**************************************************/
PrintAllData(file1,file2,file3,layer,direct1,inout,compen,polar,
    dimen,can,dia,depth,rate,tx,ty,tz,rx,ry,rz,rate1,height,same,
    depth1,scale,Pock)
char file1[LimCha],file2[LimCha],file3[LimCha],direct1[LimCha],
     inout[LimCha],compen[len],polar[len],dimen[LimCha],can[len],
     layer[maxlayer],same[LimCha],Pock[LimCha];
double dia,depth,rate,tx,ty,tz,rx,ry,rz,rate1,height,depth1,scale;
 {
 Print1Screen(6,3,25,0,file1);
 Print1Screen(6,28,51,0,file2);
 Print1Screen(6,51,76,0,file3);
 Print2Screen(11,17,38,0,layer);
 Print3Screen(11,64,76,0,rate1);
 Print2Screen(12,23,38,0,direct1);
 Print3Screen(12,64,76,0,height);
```

```
 Print2Screen(13,18,38,0,inout);
 Print2Screen(13,65,76,0,same);
 Print2Screen(14,19,38,0,dimen);
 Print3Screen(14,60,76,0,depth1);
 Print3Screen(15,22,38,0,dia);
 Print2Screen(15,63,76,0,Pock);
 Print3Screen(16,20,38,0,depth);
 Print3Screen(17,16,38,0,rate);
 Print2Screen(17,46,48,0,compen);
 Print2Screen(17,56,58,0,polar);
 Print2Screen(17,66,68,0,can);
 Print4Screen(18,20,38,0,tx,ty,tz);
 Print4Screen(19,20,38,0,rx,ry,rz);
 Print3Screen(19,46,50,0,scale);
 }

/************************ InputData() ************************/
/*                                                           */
/* Receives data from keyboard for selecting sub-program.    */
/*                                                           */
/*************************************************************/
InputData(data1,data2,data3,data4,Page)
int Page;
char data1[MaxCha],data2[MaxCha],data3[MaxCha],data4[MaxCha];
 {
 extern char *upper();
 int loop1,ind1,ind2,result1;
 char data[40];

 crt_srcp(23,13,Page);  gets(data,39);  ind1=0;  ind2=strlen(data);
 DefineData(&result1,data,data1,&ind1,ind2);
 if (result1 == 0) strcpy(data1,"NONE");
 DefineData(&result1,data,data2,&ind1,ind2);
 if (result1 == 0) strcpy(data2,"NONE");
 DefineData(&result1,data,data3,&ind1,ind2);
 if (result1 == 0) strcpy(data3,"NONE");
 DefineData(&result1,data,data4,&ind1,ind2);
 if (result1 == 0) strcpy(data4,"NONE");
 strcpy(data1,upper(data1));  strcpy(data2,upper(data2));
 }

/********** DefineData() ***********/
/*                                 */
/* Split data by checking space.   */
/*                                 */
/***********************************/
DefineData(result1,data,data1,ind1,ind2)
char data[],data1[];
int *result1,*ind1,ind2;
 {
 char ch1;
 int i,j;
 j=0;
 ch1=*(data+*ind1);
 if (ch1 == '\0') *result1=0;
```

```
 if (ch1 != '\0') {
  for (i=*ind1;i<=ind2;++i) {
   ch1=*(data+i);
   if (ch1 != '\040') {
    *ind1=i; i=ind2+1;
 } }

  for (i=*ind1;i<=ind2;++i) {
   ch1=*(data+i);
   if (ch1 == '\0' || ch1 == '\040') {
    *(data1+j)='\0';  *ind1=i;  i=ind2+1;
 } if (ch1 != '\0' && ch1 != '\040') {
    *(data1+j)=ch1;  *result1=1;  ++j;
  } } } }

/************************** Frame() ***************************/
/*                                                            */
/* Call sub program for printing rectangular line on monitor. */
/*                                                            */
/**************************************************************/
Frame(Page)
int Page;
 {
 Retangular(2,0,22,79,Page); Retangular(22,1,24,78,Page);
 Retangular(0,6,4,73,Page); Retangular(0,6,2,73,Page);
 Retangular(5,2,7,26,Page); Retangular(5,27,7,52,Page);
 Retangular(5,53,7,77,Page); Retangular(8,2,20,39,Page);
 Retangular(8,41,20,77,Page); Retangular(20,2,22,77,Page);
 Retangular(8,24,10,56,Page);
 }

/************* Retangular() *************/
/*                                      */
/* Prints rectangular-line on monitor. */
/*                                      */
/****************************************/
Retangular(ROWs,COLs,ROWe,COLe,Page)
int ROWs,COLs,ROWe,COLe,Page;
 {
 int i;

 crt_srcp(ROWs,COLs,Page); printf("%c",LT);
 crt_srcp(ROWs,COLs+1,Page);
 for (i=COLs+1;i<=COLe-1;++i) printf("%c",HOR);
 crt_srcp(ROWs,COLe,Page); printf("%c",RT);
 for (i=ROWs+1;i<=ROWe-1;++i) {
  crt_srcp(i,COLs,Page); printf("%c",VER);
  crt_srcp(i,COLe,Page); printf("%c",VER);
 } crt_srcp(ROWe,COLs,Page); printf("%c",LB);
 crt_srcp(ROWe,COLs+1,Page);
 for (i=COLs+1;i<=COLe-1;++i)
  printf("%c",HOR);
 crt_srcp(ROWe,COLe,Page); printf("%c",RB);
 }
```

```
/*************** Space() *************/
/*                                   */
/* Use for earsing or changing some */
/* messages on monitor.              */
/*                                   */
/*************************************/
Space(row,col,Page,number)
int row,col,Page,number;
 {
 int i;

 crt_srcp(row,col,Page);
 for (i=1;i<=number;++i) printf("%c",space1);
 }

/********** Print1Screen() **********/
/*                                   */
/* Prints filename data on monitor. */
/*                                   */
/*************************************/
Print1Screen(row,cols,cole,Page,file1)
int row,cols,cole,Page;
char file1[LimCha];
 {
 int col,col1,ind1;

 col=(cols+cole)/2; col1=cole-cols;
 Space(row,cols,Page,col1); ind1=strlen(file1)/2;
 crt_srcp(row,col-ind1,Page);  printf("%s",file1);
 }

/********** Print2Screen() *********/
/*                                  */
/* Prints string data on monitor. */
/*                                  */
/************************************/
Print2Screen(row,cols,cole,Page,data4)
int row,cols,cole,Page;
char data4[MaxCha];
 {
 int col1;

 col1=cole-cols; Space(row,cols,Page,col1);
 crt_srcp(row,cols,Page); printf("%s",data4);
 }

/************** Print3Screen() *************/
/*                                          */
/* Prints doubleing point data on monitor. */
/*                                          */
/********************************************/
Print3Screen(row,cols,cole,Page,rec1)
int row,cols,cole,Page;
double rec1;
 {
```

```
 int col1;

 col1=cole-cols; Space(row,cols,Page,col1);
 crt_srcp(row,cols,Page); printf("%.2f",rec1);
 }

/************ Print4Screen() ************/
/*                                      */
/* Prints coordinate data on monitor.  */
/*                                      */
/****************************************/
Print4Screen(row,cols,cole,Page,rec1,rec2,rec3)
int row,cols,cole,Page;
double rec1,rec2,rec3;
 {
 int col1;

 col1=cole-cols; Space(row,cols,Page,col1);
 crt_srcp(row,cols,Page); printf("%.1f %.1f %.1f",rec1,rec2,rec3);
 }

/************************ LayerFunction() ************************/
/*                                                               */
/* Gets layer name, storing the entities of the drawing, from */
/* the keyboard.  The default is the last given layer name.   */
/*                                                               */
/*****************************************************************/
LayerFunction(data2,layer)
char data2[MaxCha],layer[MaxCha];
 {
 AskString("Layer name",data2,layer);
 if (strlen(data2) != 0)
  strcpy(layer,data2);
 }


/************************** AskString() ****************************/
/*                                                                 */
/* AskString(), receives a string data from a call function, will */
/* print out the string data on the monitor and get information   */
/* from the  keyboard. It will change the string to a upper case. */
/*                                                                 */
/*******************************************************************/
AskString(data1,data2,data3)
char data1[maxlayer],data2[MaxCha],data3[maxlayer];
 {
 extern char *upper();

 if (strncmp(data2,"NONE",strlen(data2)) == 0)
  {
  crt_srcp(23,3,0);  printf("%s  <%s> : ",data1,data3);
  gets (data2,MaxCha-1);
  }
 strcpy(data2,upper(data2));
 }
```

```
/***************************** AskFloat() ****************************/
/*                                                                   */
/* AskFloat(),receives a string data from a call function, will      */
/* print out the string data on the monitor and get information      */
/* from the keyboard.  It will change the information to a float-    */
/* ing point.                                                        */
/*                                                                   */
/*********************************************************************/
AskFloat(data1,data2,record)
char data1[maxlayer],data2[MaxCha];
double *record;
 {
 extern double atof();

 if (strncmp(data2,"NONE",strlen(data2)) == 0) {
  crt_srcp(23,3,0);  printf("%s  <%.4f> : ",data1,*record);
  gets (data2,MaxCha-1);
 } if (strlen(data2) != 0) *record=atof(data2);
   if (*record < 0) *record=0;
 }

/********************** File() ***********************/
/*                                                   */
/* This function accepts a filename from keyboard.   */
/* Source file   : "xxxxxxxx.DXF" (drawing data)     */
/* Standard file : "xxxxxxxx.DTA" (Working condition)*/
/* Output file   : "xxxxxxxx.CNC" (Gcode data)       */
/*                                                   */
/* ************************************************* */
File(data1,data2,file)
char data1[MaxCha],data2[MaxCha],file[LimCha];
 {
 extern char *upper();
 char file1[MaxCha],file2[MaxCha];
 int chk,chk1,ind1,num1,i,err;

 err=1; ind1=-1; chk=1; Space(23,3,0,70); crt_srcp(23,3,0);
 if (strncmp(data2,"NONE",strlen(data2)) == 0) {
  if (strncmp(data1,"SOURCE",strlen(data1)) == 0)
   { printf("Source filename : "); chk=0; }
  if (strncmp(data1,"STANDARD",strlen(data1)) == 0 && chk == 1)
   printf("Standard filename : ");
  if (strncmp(data1,"OUTPUT",strlen(data1)) == 0)
   printf("Output filename : "); gets(data2,MaxCha);
 } chk=1;  num1=strlen(data2); strcpy(data2,upper(data2));
 strcpy(file1,data2);
 /*****************************************************/
 /* Find a number of characters before '.' (period). */
 /*****************************************************/
 for (i=0;i<=strlen(file1);++i) {
  if (file1[i] == '.') ind1=i;
 } if (ind1 != -1) {
  strncpy(file2,file1,ind1); num1=ind1;
 } if (ind1 == -1) strcpy(file2,file1);
   CheckFile(file2,num1,&err);
```

```
 if (err == 0) {
  strcpy(file,file1);
  if (strncmp(data1,"SOURCE",strlen(data1)) == 0 && ind1 == -1)
   { strcat(file,".DXF"); chk=0; }
  if (strncmp(data1,"STANDARD",strlen(data1)) == 0 && ind1 == -1
      && chk == 1) strcat(file,".STD");
  if (strncmp(data1,"OUTPUT",strlen(data1)) == 0 && ind1 == -1)
   strcat(file,".CNC");
 } }

/********************* CheckFile() ********************/
/*                                                    */
/* CheckFile() recieves the filename from File()      */
/* function and sends it to ErrorFi1() and ErrorFi2() */
/* function for checking errors.                      */
/*                                                    */
/******************************************************/
CheckFile(file,numchar,status)
char file[MaxCha];
int numchar,*status;
 {
 int ind1,ok1,ok2;

 ErrorFi1(file,numchar,&ind1,&ok1);
 if (ok1 == 1) ErrorFi2(file,numchar,ind1,&ok2);
 if (ok1 == 1 && ok2 == 1) *status=0;
 }

/********************* ErrorFi1() *********************/
/*                                                    */
/* ErrorFi1() uses for checking a number of characters */
/* and the second character of the filename.          */
/* The error messages are:                            */
/*  error 1 : Drive x: and filename cannot exceed than */
/*            14 characters.                          */
/*  error 2 : Filename misses semicolon (:) at the    */
/*            second character.                       */
/*  error 3 : Select A:, B:, C: or D: for disk drive  */
/*            only.                                   */
/*  error4 : No have filename for reading.            */
/*                                                    */
/******************************************************/
ErrorFi1(file,numchar,ind,status)
char file[];
int numchar,*ind,*status;

 {
 *status=1;
 if (numchar > 10)
  { crt_srcp(21,13,0); puts (error1); *status=0; }
 if (*status == 1)
  if (numchar == 0)
   { crt_srcp(21,13,0); puts (error4); *status=0; }
 if (*status == 1)
  if ((numchar == 9 || numchar == 10) && file[1] != ':')
```

```
   { crt_srcp(21,13,0); puts (error2); *status=0; }
 if (*status == 1)
  if (file[1] == ':') {
   if (file[0] < 'A' || file[0] > 'D')
    { crt_srcp(21,13,0); puts (error3); *status=0; }
   if (file[2] == '\0')
    { crt_srcp(21,13,0); puts (error4); *status=0; }
 } if (*status == 1 && file[1] == ':') *ind=2;
 if (*status == 1)
  if (file[1] != ':') *ind=0;
 }

/********************* ErrorFi2() ************************/
/*                                                       */
/* ErrorFi2() use for checking each character of the     */
/* given filename.  It must be letters or digits. If     */
/* the second character is a semicolon (:), ErrorFi()    */
/* will start for checking at the third character.       */
/* The error message is :                                */
/*  error6 : Filename must be letters or digits.".       */
/*                                                       */
/*********************************************************/
ErrorFi2(file,numchar,ind,ok2)
char file[];
int numchar,ind,*ok2;
 {
 *ok2=1;
 if (isalnum(file[ind]) == 0)
  { crt_srcp(21,13,0); puts (error5); *ok2=0; }
 if (*ok2 == 1) {
  for (++ind;ind<=numchar-1;ind++) {
   if (iscsym(file[ind]) == 0) {
    crt_srcp(21,13,0); puts (error6); *ok2=0; ind=numchar+1;
 } } } }

/***************** TooChange() ***************/
/*                                           */
/* Uses for define the tool change position. */
/* The default is (0,0,0).                   */
/*                                           */
/*********************************************/
ToolChange(data2,data3,data4,tx,ty,tz)
char data2[MaxCha],data3[MaxCha],data4[MaxCha];
double *tx,*ty,*tz;
 {
 extern double atof();

 if (strncmp(data2,"NONE",strlen(data2)) != 0) {
  *tx=atof(data2); *ty=atof(data3); *tz=atof(data4);
 } if (strncmp(data2,"NONE",strlen(data2)) == 0) {
  crt_srcp(23,3,0);
  printf("Tool change position %.2f %.2f %.2f : ",*tx,*ty,*tz);
  gets(data2,MaxCha-1); SplitData(data2,tx,ty,tz);
 } }
```

```
/******************* ReferencePosition() ********************/
/*                                                          */
/* Uses for define the reference position.  The default is */
/* (0,0,0).                                                 */
/*                                                          */
/************************************************************/
ReferencePosition(data2,data3,data4,rx,ry,rz)
char data2[MaxCha],data3[MaxCha],data4[MaxCha];
double *rx,*ry,*rz;
 {
 extern double atof();

 if (strncmp(data2,"NONE",strlen(data2)) != 0) {
  *rx=atof(data2); *ry=atof(data3); *rz=atof(data4);
 } if (strncmp(data2,"NONE",strlen(data2)) == 0) {
  crt_srcp(23,3,0);
  printf("Reference position %.2f %.2f %.2f : ",*rx,*ry,*rz);
  gets(data2,MaxCha-1); SplitData(data2,rx,ry,rz);
 } }

/********************* DirectionFunction() ******************/
/*                                                          */
/* Selects a direction, clockwise (CW) or counterclockwise */
/* (CCW), of a cutter for milling. The default is the last  */
/* given direction.                                         */
/*                                                          */
/************************************************************/
DirectionFunction(data2,direct1)
char data2[MaxCha],direct1[len];
 {
 AskString("Direction of cutter CW or CCW",data2,direct1);
 if (strlen(data2) != 0) {
  if (strncmp(data2,"CCW",strlen(data2)) == 0)
   strcpy(direct1,"CCW");
  if (strncmp(data2,"CW",strlen(data2)) == 0)
   strcpy(direct1,"CW");
 } }

/******************** In_Outside() **********************/
/*                                                      */
/* Selects a path, inside or outside path, of cutter    */
/* for milling.  The default is the last give path.     */
/*                                                      */
/********************************************************/
In_Outside(data2,inout)
char data2[MaxCha],inout[MaxCha];
 {
 AskString("Inside or Outside path",data2,inout);
 if (strlen(data2) != 0) {
  if (strncmp(data2,"OUTSIDE",strlen(data2)) == 0)
   strcpy(inout,"OUTSIDE");
  if (strncmp(data2,"INSIDE",strlen(data2)) == 0)
   strcpy(inout,"INSIDE");
  } }
```

```
/******************* Dimension() ********************/
/*                                                  */
/* Selects millimeter or inch unit for a working    */
/* condition. The default is the last given unit.   */
/*                                                  */
/****************************************************/
Dimension(data2,dimen)
char data2[MaxCha],dimen[MaxCha];
 {
 AskString("Dimension in Inch or Millimeter",data2,dimen);
 if (strlen(data2) != 0) {
  if (strncmp(data2,"MILLIMETER",strlen(data2)) == 0)
   strcpy(dimen,"MILLIMETER");
  if (strncmp(data2,"INCH",strlen(data2)) == 0)
   strcpy(dimen,"INCH");
  } }

/************************** Code() **************************/
/*                                                          */
/* This function, accept string data from a call function,  */
/* will print out the string data on the monitor and get    */
/* information from the keyboard.  The default is the last  */
/* given information ( for setting a working condition).    */
/*                                                          */
/************************************************************/
Code(data1,data2,data3)
char data1[maxlayer],data2[MaxCha],data3[MaxCha];
 {
 AskString(data1,data2,data3);
 if (strlen(data2) != 0) strncpy(data3,data2,len2);
 }


/****************** SplitData() *******************/
/*                                                */
/* Uses for split the string data, from a call    */
/* function and converts them to doubleing points. */
/*                                                */
/**************************************************/
SplitData(data,x,y,z)
char data[MaxCha];
double *x,*y,*z;
 {
 extern double atof();
 int ind1,ind2,result1;
 char data3[MaxCha];

 ind1=0; ind2=strlen(data);
 DefineData(&result1,data,data3,&ind1,ind2);
 if (result1 == 0) *x=0;
 if (result1 != 0) *x=atof(data3);
 DefineData(&result1,data,data3,&ind1,ind2);
 if (result1 == 0) *y=0;
 if (result1 != 0) *y=atof(data3);
 DefineData(&result1,data,data3,&ind1,ind2);
```

```
 if (result1 == 0) *z=0;
 if (result1 != 0) *z=atof(data3);
 }

/******************** SameDepth() ***********************/
/*                                                      */
/* Uses for define the equal depth for all drilling.    */
/* The default is the last answer.                      */
/*                                                      */
/******************************************************/
SameDepth(data2,same)
char data2[MaxCha],same[LimCha];
 {
 AskString("All drilling same depth Yes or No",data2,same);
 if (strlen(data2) != 0) {
  if (strncmp(data2,"NO",strlen(data2)) == 0) strcpy(same,"NO");
  if (strncmp(data2,"YES",strlen(data2)) == 0) strcpy(same,"YES");
 } }

/****************** FindBeginningOfData() *****************/
/*                                                        */
/* Finds the beginning of data from the given filename.   */
/*                                                        */
/**********************************************************/
FindBeginningOfData()
 {
 extern int ncode,err9;
 extern char ccode[];
 int chk,skip;

 chk=1; err9=0; skip=0;
 while (chk == 1) {
  if (skip != 1) Read_data();
  if (skip == 1) skip=0;
  if (strncmp(ccode,"EOF",strlen(ccode)) == 0) { chk=0; err9=1; }
  if (err9 == 0) {
   if (strncmp(ccode,"BLOCKS",strlen(ccode))== 0 && ncode == 2) {
     BlockData(); skip=1;
 } if (strncmp(ccode,"ENTITIES",strlen(ccode))== 0 && ncode == 2)
    chk=0;
 } } }

/************************ Read_data() *************************/
/*                                                            */
/* Uses for reading integer & string data from given Filename. */
/*                                                            */
/**************************************************************/
Read_data()
 {
 extern FILE *rfpt;
 extern int ncode;
 extern char ccode[];

 fscanf(rfpt,"%d",&ncode); fscanf(rfpt,"%s",ccode);
 }
```

```
/********************** Skip_data() ***************************/
/*                                                            */
/* Uses for skip integer & string data from given Filename. */
/*                                                            */
/**************************************************************/
Skip_data()
 {
 extern FILE *rfpt;
 extern int ncode;
 extern char ccode[];

 fscanf(rfpt,"%d",&ncode); fscanf(rfpt,"%s",ccode);
 }

/****** StoreDataFromDiskToMemory() ******/
/*                                        */
/* Stores data from diskette to memory. */
/*                                        */
/******************************************/
StoreDataFromDiskToMemory()
 {
 extern int ncode,err8;
 extern char ccode[];
 extern double lxs[],lys[],lxe[],lye[];
 extern double cirx[],ciry[],cirr[];
 extern double arcxc[],arcyc[],arcxs[],arcys[],arcxe[],arcye[];
 extern double plx[],ply[],plxc[],plyc[];
 extern char arcd[][len],pld[][len];
 extern int polys[],polye[],lin,cin,ain,pin,pin1;
 int loop1,right,skip;
 char code1[6];

 err8=0; skip=1; lin=0; cin=0; ain=0; pin=0; pin1=0;
 loop1=strncmp(ccode,"EOF",strlen(ccode));
 while (loop1 != 0) {
  if (skip != 1)  Read_data();
  if (skip == 1)  skip=0;
  if (ncode == 0 && strncmp(ccode,"LINE",strlen(ccode)) == 0) {
   Read_data();  Check_layer(&right);
   if (right == 0) {
    Line(&lxs[lin],&lys[lin],&lxe[lin],&lye[lin]);
    if (err8 == 0)  ++lin;
 } } if (ncode == 0 && strncmp(ccode,"CIRCLE",strlen(ccode)) == 0) {
   Read_data();  Check_layer(&right);
   if (right == 0) {
    Circle(&cirx[cin],&ciry[cin],&cirr[cin]);
    if (err8 == 0)  ++cin;
 } } if (ncode == 0 && strncmp(ccode,"ARC",strlen(ccode)) == 0) {
   Read_data();  Check_layer(&right);
   if (right == 0) {
    Arc(&arcxc[ain],&arcyc[ain],&arcxs[ain],&arcys[ain],&arcxe[ain],
       &arcye[ain],&arcd[ain]);
    if (err8 == 0) ++ain;
 } } if (ncode == 0 && strncmp(ccode,"POLYLINE",strlen(ccode)) == 0)
 { strcpy(code1,"POLY");  polys[pin1]=pin;
```

```
   Polyline(&pin,code1,plx,ply,plxc,plyc,pld);
   polye[pinl]=pin-1;  ++pinl;
 } if (ncode == 0 && strncmp(ccode,"INSERT",strlen(ccode)) == 0) {
   Read_data();  Check_layer(&right);
   if (right == 0) {
    SeparateBlock();  skip=1;
 } } loop1=strncmp(ccode,"EOF",strlen(ccode));
  if (err8 == 1) {
   Space(21,13,0,64); crt_srcp(21,13,0);  puts(error10);  loop1=0;
 } } }

/***************** Line() ****************/
/*                                       */
/* Reads LINE data from given filename. */
/*                                       */
/*****************************************/
Line(x1,y1,x2,y2)
double *x1,*y1,*x2,*y2;
 {
 extern double atof();
 extern int ncode,err8;
 extern char ccode[];

 err8=1;  Read_data();
 if (ncode == 10) { *x1=atof(ccode);  Read_data();
  if (ncode == 20) { *y1=atof(ccode);  Read_data();
   if (ncode == 11) { *x2=atof(ccode);  Read_data();
    if (ncode == 21) { *y2=atof(ccode);  err8=0;
 } } } } }

/***************** Circle() ****************/
/*                                         */
/* Reads CIRCLE data from given filename. */
/*                                         */
/*******************************************/
Circle(cirx,ciry,cirr)
double *cirx,*ciry,*cirr;
 {
 extern double atof();
 extern int ncode,err8;
 extern char ccode[];

 err8=1; Read_data();
 if (ncode == 10) { *cirx=atof(ccode);  Read_data();
  if (ncode == 20) { *ciry=atof(ccode);  Read_data();
   if (ncode == 40) { *cirr=atof(ccode);  err8=0;
 } } } }

/***************** Arc() ****************/
/*                                      */
/* Reads ARC data from given filename. */
/*                                      */
/****************************************/
Arc(xc,yc,xs,ys,xe,ye,d)
double *xc,*yc,*xs,*ys,*xe,*ye;
```

```
char d[4];
 {
 extern double sin(),cos(),atof();
 extern int ncode,err8;
 extern double vcode;
 double arcx,arcy,arcr,arcs,arce;

 err8=1; Read_data();
 if (ncode == 10) { arcx=atof(ccode);  Read_data();
  if (ncode == 20) { arcy=atof(ccode);  Read_data();
   if (ncode == 40) { arcr=atof(ccode);  Read_data();
    if (ncode == 50) { arcs=atof(ccode);  Read_data();
     if (ncode == 51) { arce=atof(ccode); err8=0;
/*************************************************/
/* Changes ARC data from AutoCAD to G code data.*/
/*************************************************/
      *xc=arcx;  *yc=arcy;
      *xs=arcx+arcr*cos(arcs*Pi/180);
      *ys=arcy+arcr*sin(arcs*Pi/180);
      *xe=arcx+arcr*cos(arce*Pi/180);
      *ye=arcy+arcr*sin(arce*Pi/180);
      strcpy(d,"CCW");
 } } } } } }

/***************** Polyline() *****************/
/*                                            */
/* Reads POLYLINE data from given filename. */
/*                                            */
/**********************************************/
Polyline(ind1,name,plx,ply,plxc,plyc,pld)
int *ind1;
char name[6],pld[][len];
double plx[],ply[],plxc[],plyc[];
 {
 extern double atof();
 extern int atoi();
 extern int ncode,err8;
 extern char ccode[];
 double plx1[numb],ply1[numb],pla1[numb];
 int right,loop,ind2,ind3,int1,skip;
 char type[6];

 err8=1; ind2=0; right=0;
 Read_data();
 if (strncmp(ccode,"SEQEND",strlen(ccode)) == 0 && ncode == 0)
  right=1;
 if (strncmp(ccode,"SEQEND",strlen(ccode)) != 0 || ncode != 0) {
  if (strncmp(name,"BLOCK",strlen(name)) == 0 && ncode == 8)
   right=0;
  if (strncmp(name,"POLY",strlen(name)) == 0 && ncode == 8)
   Check_layer(&right);
 } if (right == 0) {
  right=1; Read_data();
  if (ncode == 66 && atoi(ccode) == 1) {
   Read_data();
```

```
  if (ncode != 70 || atoi(ccode) != 1) strcpy(type,"OPEN");
  if (ncode == 70 && atoi(ccode) == 1) strcpy(type,"CLOSE");
  if (ncode == 70) Read_data();
  if (strncmp(ccode,"VERTEX",strlen(ccode)) == 0 && ncode == 0) {
   loop=1; skip=0;
   while (loop == 1) {
     Read_data();  right=1;
    if (strncmp(ccode,"SEQEND",strlen(ccode)) == 0 && ncode == 0) {
     right=1;  loop=0;
  } if (strncmp(ccode,"SEQEND",strlen(ccode)) != 0 || ncode != 0) {
     if (strncmp(name,"BLOCK",strlen(name)) == 0 && ncode == 8)
      right=0;
     if (strncmp(name,"POLY",strlen(name)) == 0 && ncode == 8)
      Check_layer(&right);
  } if (right == 0) { Read_data();
     if (ncode == 10) { plx1[ind2]=atof(ccode);  Read_data();
      if (ncode == 20) {
       ply1[ind2]=atof(ccode);  err8=0;  Read_data();
       if (ncode == 42) pla1[ind2]=atof(ccode);
       if (ncode == 0) { pla1[ind2]=9999;
        if (strncmp(ccode,"SEQEND",strlen(ccode)) == 0 ) loop=0;
     } ++ind2;
} } } } } } }
for (ind3=0;ind3<ind2-1;++ind3) {
 if (int1=pla1[*ind1] == 9999) {
  plx[*ind1]=plx1[ind3]; ply[*ind1]=ply1[ind3];
  plxc[*ind1]=9999; plyc[*ind1]=9999;
  strcpy(pld[*ind1],"NONE");
} if (int1=pla1[*ind1] != 9999) {
  if (pla1[*ind1] < 0) strcpy(pld[*ind1],"CW");
  if (pla1[*ind1] >= 0) strcpy(pld[*ind1],"CCW");
  plx[*ind1]=plx1[ind3]; ply[*ind1]=ply1[ind3];
  plx[*ind1+1]=plx1[ind3+1]; ply[*ind1+1]=ply1[ind3+1];
  Center_poly(&plx[*ind1],&ply[*ind1],&plxc[*ind1],&plyc[*ind1],
      &pla1[*ind1],&plx[*ind1+1],&ply[*ind1+1]);
} ++*ind1;
} if (strncmp(type,"CLOSE",strlen(type)) == 0) {
 if (int1=pla1[*ind1] == 9999) {
  plx[*ind1]=plx1[ind2-1]; ply[*ind1]=ply1[ind2-1];
  plxc[*ind1]=9999; plyc[*ind1]=9999;
  strcpy(pld[*ind1],"NONE");
} if (int1=pla1[*ind1] != 9999) {
  if (pla1[*ind1] < 0) strcpy(pld[*ind1],"CW");
  if (pla1[*ind1] >= 0) strcpy(pld[*ind1],"CCW");
  plx[*ind1]=plx1[ind2-1]; ply[*ind1]=ply1[ind2-1];
  plx[*ind1+1]=plx1[0]; ply[*ind1+1]=ply1[0];
  Center_poly(&plx[*ind1],&ply[*ind1],&plxc[*ind1],&plyc[*ind1],
      &pla1[*ind1],&plx[0],&ply[0]);
} ++*ind1; plx[*ind1]=plx1[0]; ply[*ind1]=ply1[0];
 strcpy(pld[*ind1],"NONE"); ++*ind1;
} if (strncmp(type,"OPEN",strlen(type)) == 0) {
 plx[*ind1]=plx1[ind2-1]; ply[*ind1]=ply1[ind2-1];
 strcpy(pld[*ind1],"NONE"); ++*ind1;
} }
```

```
/*************** BlockData() ***************/
/*                                         */
/* Reads BLOCK data from given filename. */
/*                                         */
/*******************************************/
BlockData()
 {
 extern double atof();
 extern int atoi();
 extern double bxs[],bys[],bxc[],byc[],bxe[],bye[];
 extern double binsx[],binsy[];
 extern char ccode[],bname[][maxlayer],btype[][6],btype1[][7],
             bd[][len];
 extern int binss[],binse[];
 extern int ncode,err8,bin,bin1;
 double bplx[b_num],bply[b_num],bplxc[b_num],bplyc[b_num];
 char bpld[b_num][len],code1[6];
 int loop1,loop2,loop3,blks,skip;

 loop1=1; bin=0; bin1=0; skip=0;
 while (loop1 == 1) {
  Read_data();
  if (strncmp(ccode,"ENDSEC",strlen(ccode)) == 0 && ncode == 0) {
   loop1=0; loop2=0;
 } if (strncmp(ccode,"BLOCK",strlen(ccode)) == 0 && ncode == 0)
   loop2=1;
  while (loop2 == 1) {
   Skip_data();  Read_data();
   if (ncode == 2) {
    strncpy(bname[bin],ccode,strlen(ccode));
    Read_data();
    if (ncode == 70 && atoi(ccode) == 1)
     strcpy(btype[bin],"CLOSE");
    if (ncode == 70 && atoi(ccode) == 0)
     strcpy(btype[bin],"OPEN");
    Read_data();
    if (ncode == 10) { binsx[bin]=atof(ccode);  Read_data();
     if (ncode == 20) {
      binsy[bin]=atof(ccode);  binss[bin]=bin1;  loop3=1;
      while (loop3 == 1) {
       if (skip != 1)  Read_data();
       if (skip == 1)  skip=0;
       if (strncmp(ccode,"ENDBLK",strlen(ccode)) == 0 &&
           ncode == 0) loop3=0;
       if (strncmp(ccode,"LINE",strlen(ccode)) == 0 && ncode == 0)
        { Skip_data();  strcpy(btype1[bin1],"LINE");
        Line(&bxs[bin1],&bys[bin1],&bxe[bin1],&bye[bin1]);
        if (err8 == 0)  ++bin1;
     } if (strncmp(ccode,"ARC",strlen(ccode)) == 0 && ncode ==0) {
        Skip_data();  strcpy(btype1[bin1],"ARC");
        Arc(&bxc[bin1],&byc[bin1],&bxs[bin1],&bys[bin1],&bxe[bin1],
            &bye[bin1],bd[bin1]);
        if (err8 == 0)  ++bin1;
      } if (strncmp(ccode,"CIRCLE",strlen(ccode)) == 0 &&
            ncode == 0) {
```

```
        Skip_data();  strcpy(btype1[bin1],"CIRCLE");
        Circle(&bxc[bin1],&byc[bin1],&bxs[bin1]);
        if (err8 == 0)  ++bin1;
      } if (strncmp(ccode,"POLYLINE",strlen(ccode)) == 0 &&
          ncode == 0) {
        strcpy(code1,"BLOCK");  blks=0;
        Polyline(&blks,code1,bplx,bply,bplxc,bplyc,bpld);
        ChangeBlockDataintoLineOrArcData(blks,bplx,bply,bplxc,bplyc,
          bpld);
      } if (strncmp(ccode,"INSERT",strlen(ccode)) == 0 &&
          ncode == 0) {
        Skip_data();  StoreBlock();  skip=1;
      } } binse[bin]=bin1-1;  ++bin;  loop2=0;
 } } } } } }

/********** Check_layer() ********/
/*                               */
/* Checks the given layer name. */
/*                               */
/*********************************/
Check_layer(right)
int *right;
 {
 extern int ncode;
 extern char ccode[],layer[];
 extern int err8;
 int result;

 *right=1;
 if (strncmp(ccode,layer,strlen(ccode)) == 0 && ncode == 8)
  { err8=1;  *right=0;
 } }

/******************** Center_poly() ********************/
/*                                                     */
/* Changes POLYLINE data from AutoCAD to G code data. */
/*                                                     */
/*******************************************************/
Center_poly(xs,ys,xc,yc,alpha,xe,ye)
double *xs,*ys,*xc,*yc,*alpha,*xe,*ye;
 {
 extern double fabs(),sqrt(),sin(),cos(),tan(),atan();
 double x1center,y1center,x2center,y2center;
 double xbar,ybar;
 double square1,chord1,beta,normal,alpha1;
 int result5,result6;

 xbar=(*xe+*xs)/2;  ybar=(*ye+*ys)/2;
 if (*alpha == 1 || *alpha == -1) { *xc=xbar;  *yc=ybar; }
 else {
  if (*alpha < 0) result5=-1;
  else result5=1;
  square1=((*xe-*xs)*(*xe-*xs)+(*ye-*ys)*(*ye-*ys))/4;
  chord1=sqrt(square1); beta=atan((*ye-*ys)/(*xe-*xs));
  alpha1=2*atan(*alpha); normal=fabs(chord1/tan(alpha1));
```

```
  x1center=xbar+normal*cos(beta+Pi/2);
  y1center=ybar+normal*sin(beta+Pi/2);
  x2center=xbar+normal*cos(beta-Pi/2);
  y2center=ybar+normal*sin(beta-Pi/2);
  CrossProduct(*xs,*ys,*xe,*ye,xbar,ybar,x1center,y1center,
      &result6);
  if (result5 == result6) { *xc=x1center;  *yc=y1center; }
  if (result5 != result6) {
   CrossProduct(*xs,*ys,*xe,*ye,xbar,ybar,x2center,y2center,
       &result6);
   if (result5 == result6) { *xc=x2center;  *yc=y2center; }
   else {
    Space(21,13,0,64);  crt_srcp(21,13,0);  puts(error10);
 } } } }

/******** CrossProduct() ********/
/*                              */
/* Finds sign of Cross Product. */
/*                              */
/********************************/
CrossProduct(xs1,ys1,xe1,ye1,xs2,ys2,xe2,ye2,result)
double xs1,ys1,xe1,ye1,xs2,ys2,xe2,ye2;
int *result;
 {
 double answer;

 answer=(xe1-xs1)*(ye2-ys2)-(xe2-xs2)*(ye1-ys1);
 if (answer == 0) {
  Space(21,13,0,64); crt_srcp(21,13,0); puts(error11); *result=0;
 } if (answer != 0) {
  if (answer < 0) *result=-1;
  else *result=1;
 } }

/********************* LineDataNearOrigin() *********************/
/*                                                              */
/* Finds a coordinat of LINE data which is the nearest origin. */
/*                                                              */
/****************************************************************/
LineDataNearOrigin()
 {
 extern double fabs();
 extern double lxs[],lys[],lxe[],lye[];
 extern int lin;
 double zero;
 int i,last;
 char nam1[5];

 if (lin >= 0) {
  strcpy(nam1,"LINE");
  for (i=0;i<lin;++i)
   Pro1_interchange(nam1,&lxs[i],&lys[i],&lxe[i],&lye[i],"NONE");
   zero=0; last=0;
  for (i=1;i<lin;++i) {
```

```
    if (fabs(lxs[0]-lxs[i]) <= Limit2 &&
         fabs(lys[0]-lys[i]) <= Limit2) last=i;
    Pro2_interchange(nam1,&lxs[0],&lys[0],&lxe[0],&lye[0],&zero,
        &zero,"NONE",&lxs[i],&lys[i],&lxe[i],&lye[i],&zero,&zero,
        "NONE");
 } if (last > 0) {
   i=last;
   Pro2_interchange(nam1,&lxs[1],&lys[1],&lxe[1],&lye[1],&zero,
       &zero,"NONE",&lxs[i],&lys[i],&lxe[i],&lye[i],&zero,&zero,
       "NONE");
 } } }

/******************** ArcDataNearOrigin() ***********************/
/*                                                              */
/* Finds a coordinat of ARC data whichis the nearest origin.    */
/*                                                              */
/**************************************************************/
ArcDataNearOrigin()
 {
 extern double fabs();
 extern double arcxc[],arcyc[],arcxs[],arcys[],arcxe[],arcye[];
 extern char arcd[][len];
 extern int ain;
 int i,last;
 char nam1[5];

 if (ain >= 0) {
  strcpy(nam1,"ARC");
  for (i=0;i<ain;++i)
   Pro1_interchange(nam1,&arcxs[i],&arcys[i],&arcxe[i],&arcye[i],
       &arcd[i]);
  last=0;
  for (i=1;i<ain;++i) {
   if (fabs(arcxs[0]-arcxs[i]) <= Limit2 &&
        fabs(arcys[0]-arcys[i]) <= Limit2) last=i;
   Pro2_interchange(nam1,&arcxs[0],&arcys[0],&arcxe[0],&arcye[0],
       &arcxc[0],&arcyc[0],&arcd[0],&arcxs[i],&arcys[i],&arcxe[i],
       &arcye[i],&arcxc[i],&arcyc[i],&arcd[i]);
 } if (last > 0) {
   i=last;
   Pro2_interchange(nam1,&arcxs[1],&arcys[1],&arcxe[1],&arcye[1],
       &arcxc[1],&arcyc[1],&arcd[1],&arcxs[i],&arcys[i],&arcxe[i],
       &arcye[i],&arcxc[i],&arcyc[i],&arcd[i]);
 } } }

/********** Pro1_interchange() **********/
/*                                      */
/* Interchanges values between starting */
/* and ending point in each data.       */
/*                                      */
/****************************************/
Pro1_interchange(name,x1,y1,x2,y2,direct1)
double *x1,*y1,*x2,*y2;
char name[5],direct1[5];
 {
```

```
 double xx,yy,dist1,dist2;
 int pass;

 pass=0; dist1=*x1**x1+*y1**y1;  dist2=*x2**x2+*y2**y2;
 if (dist1 > dist2) {
  xx=*x1;  yy=*y1;  *x1=*x2; *y1=*y2;  *x2=xx;  *y2=yy;
  if (strncmp(name,"ARC",strlen(name)) == 0) {
   if (strncmp(direct1,"CCW",strlen(direct1)) == 0) {
    strcpy(direct1,"CW");  pass=1;
 } if (pass == 0) {
    if (strncmp(direct1,"CW",strlen(direct1)) == 0)
     strcpy(direct1,"CCW");
 } } } }

/********** Pro2_interchange() **********/
/*                                      */
/* Interchanges values between LINE and */
/* LINE data or ARC and ARC data.       */
/*                                      */
/****************************************/
Pro2_interchange(name,xs1,ys1,xe1,ye1,xc1,yc1,d1,xs2,ys2,xe2,ye2,
    xc2,yc2,d2)
char name[5],d1[5],d2[5];
double *xs1,*ys1,*xe1,*ye1,*xc1,*yc1,*xs2,*ys2,*xe2,*ye2,*xc2,*yc2;
 {
 double xx1,yy1,xx2,yy2,dist1,dist2,cx,cy;
 char direct3[5];

 dist1=*xs1**xs1+*ys1**ys1;  dist2=*xs2**xs2+*ys2**ys2;
 if (dist1 > dist2) {
  xx1=*xs1;  yy1=*ys1;  xx2=*xe1;  yy2=*ye1;
  *xs1=*xs2; *ys1=*ys2; *xe1=*xe2; *ye1=*ye2;
  *xs2=xx1;  *ys2=yy1;  *xe2=xx2;  *ye2=yy2;
  if (strncmp(name,"ARC",strlen(name)) == 0) {
   cx=*xc1;  cy=*yc1;  *xc1=*xc2;  *yc1=*yc2;  *xc2=cx;  *yc2=cy;
   strcpy(direct3,d1);  strcpy(d1,d2);  strcpy(d2,direct3);
 } } }

/******************** IncreaseDeltaXY() ********************/
/*                                                         */
/* Increases a small value for finding cutter direction.   */
/*                                                         */
/***********************************************************/
IncreaseDeltaXY(xs,xe,delx,val)
double xs,xe,*delx,val;

 { if (xe-xs < 0)  *delx=-val;
 if (xe-xs > 0)  *delx=val;
 }

/********** ChangePolyLineDataIntoLineOrArcData() ***********/
/*                                                          */
/* Uses for change all POLYLINE data into LINE or ARC data. */
/*                                                          */
/************************************************************/
```

```
ChangePolyLineDataIntoLineOrArcData()
 {
 extern double lxs[],lys[],lxe[],lye[],plx[],ply[],plxc[],plyc[];
 extern double arcxs[],arcys[],arcxc[],arcyc[],arcxe[],arcye[];
 extern char arcd[][len],pld[][len];
 extern int lin,ain,pin1,polys[],polye[];
 int i,ii,k,l,int1,int2;

 if (pin1 >= 0) {
  for (ii=0;ii<pin1;++ii) {
   k=polys[ii]; l=polye[ii];
   for (i=k;i<l;++i) {
    if ((int1=plxc[i]) == 9999 && (int2=plyc[i]) == 9999) {
     lxs[lin]=plx[i];  lys[lin]=ply[i];
     lxe[lin]=plx[i+1];  lye[lin]=ply[i+1];  ++lin;
  } if ((int1=plxc[i]) != 9999 || (int2=plyc[i]) != 9999) {
     arcxs[ain]=plx[i];  arcys[ain]=ply[i];
     arcxc[ain]=plxc[i];  arcyc[ain]=plyc[i];
     strcpy(arcd[ain],pld[i]);
     arcxe[ain]=plx[i+1];  arcye[ain]=ply[i+1];  ++ain;
 } } } } }

/************************* NewCoordinate() *************************/
/*                                                                 */
/* 1. Defines value to the user's direction that uses to compare */
/*    the calculated direction from a sub-program.                 */
/* 2. Selects sub-programs for calculating the cutter positions  */
/*    from the edge of workpiece.                                  */
/* 3. Calls sub-program to find the intersected point of curves. */
/*                                                                 */
/*******************************************************************/
NewCoordinate(dim,begin,index,num,direct1,inout,dia1)
double dia1;
char direct1[],inout[],dim[];
int begin,index,*num;
 {
 extern double datax[],datay[],dataxc[],datayc[];
 extern char figure[][len],datad[][len];
 extern int datas[],datae[];
 double xc1,yc1,xc2,yc2,rx1,ry1,m1,c1,rx2,ry2,m2,c2;
 double x,y,xvar,yvar,xx,yy,x1,y1;
 int i,ii,j,inds,inde,result1,select1,select2;
 char d1[len],d2[len];

 if (dia1 > 0) {
  Space(21,13,0,64);  crt_srcp(21,13,0);
  printf("Calculates the cutter positions.");
  if (strncmp(direct1,"CCW",strlen(direct1)) == 0) {
   if (strncmp(inout,"OUTSIDE",strlen(inout)) == 0) result1=-1;
   if (strncmp(inout,"INSIDE",strlen(inout)) == 0) result1=1;
 } if (strncmp(direct1,"CW",strlen(direct1)) == 0) {
   if (strncmp(inout,"OUTSIDE",strlen(inout)) == 0) result1=1;
   if (strncmp(inout,"INSIDE",strlen(inout)) == 0) result1=-1;
 } for (ii=begin;ii<index;++ii) {
   i=datas[ii];  j=datae[ii]-1;  x=datax[j];  y=datay[j];
```

```
 xc1=dataxc[j];  yc1=datayc[j];  xvar=datax[i];  yvar=datay[i];
 xc2=dataxc[i];  yc2=datayc[i];  x1=datax[i+1];  y1=datay[i+1];
 strcpy(d1,datad[j]); strcpy(d2,datad[i]);
 Setcode(j,i,&select1,&select2);
 if (strncmp(figure[ii],"CLOSE",strlen(figure[ii])) == 0) {
  xx=xvar; yy=yvar;
  CalculateNewPoint(j,select1,x,y,xvar,yvar,dial,result1,&rx1,
      &ry1,&m1,&c1);
  CalculateNewPoint(i,select2,xvar,yvar,x1,y1,dial,result1,&rx2,
      &ry2,&m2,&c2);
  if (select1 == 1 && select2 == 1)
   LineLineIntersectPoint(i,x,xvar,x1,rx1,ry1,m1,c1,rx2,ry2,
       m2,c2);
  if ((select1 == 1 && select2 == 2) || (select1 == 2 &&
       select2 == 1))
   LineArcIntersectPoint(dim,num,ii,index,&i,&j,x,y,xvar,yvar,x1,
       y1,dial,result1,rx1,ry1,xc1,yc1,d1,m1,c1,rx2,ry2,xc2,yc2,
       d2,m2,c2);
  if (select1 == 2 && select2 == 2)
   ArcArcIntersectPoint(num,ii,index,&i,&j,x,y,xvar,yvar,x1,y1,
       dial,result1,rx1,ry1,dataxc[j],datayc[j],datad[j],rx2,ry2,
       dataxc[i],datayc[i],datad[i]);
 } else {
  CalculateNewPoint(i,select2,xvar,yvar,x1,y1,dial,result1,&rx2,
      &ry2,&m2,&c2);
  datax[i]=rx2; datay[i]=ry2;
 } inds=i+1; inde=j-1;
 for (i=inds;i<=inde;++i) {
  Setcode(i-1,i,&select1,&select2); x=xvar;  y=yvar;
  xvar=datax[i]; yvar=datay[i]; x1=datax[i+1]; y1=datay[i+1];
  xc1=dataxc[i-1]; yc1=datayc[i-1]; xc2=dataxc[i];
  yc2=datayc[i]; strcpy(d1,datad[i-1]);  strcpy(d2,datad[i]);
  CalculateNewPoint(i-1,select1,x,y,xvar,yvar,dial,result1,&rx1,
      &ry1,&m1,&c1);
  CalculateNewPoint(i,select2,xvar,yvar,x1,y1,dial,result1,&rx2,
      &ry2,&m2,&c2);
  if (select1 == 1 && select2 == 1)
   LineLineIntersectPoint(i,x,xvar,x1,rx1,ry1,m1,c1,rx2,ry2,m2,c2);
  if ((select1 == 1 && select2 == 2) || (select1 == 2 &&
       select2 == 1))
   LineArcIntersectPoint(dim,num,ii,index,&i,&inde,x,y,xvar,yvar,
       x1,y1,dial,result1,rx1,ry1,xc1,yc1,d1,m1,c1,rx2,ry2,xc2,
       yc2,d2,m2,c2);
  if (select1 == 2 && select2 == 2)
   ArcArcIntersectPoint(num,ii,index,&i,&inde,x,y,xvar,yvar,x1,
       y1,dial,result1,rx1,ry1,xc1,yc1,d1,rx2,ry2,xc2,yc2,d2);
 } j=inde+1;  Setcode(j-1,j,&select1,&select2);
 if (strncmp(figure[ii],"CLOSE",strlen(figure[ii])) == 0) {
  xc1=dataxc[j-1]; yc1=datayc[j-1]; xc2=dataxc[j];
  yc2=datayc[j]; strcpy(d1,datad[j-1]);  strcpy(d2,datad[j]);
  CalculateNewPoint(j-1,select1,xvar,yvar,x1,y1,dial,result1,
      &rx1,&ry1,&m1,&c1);
  CalculateNewPoint(j,select2,x1,y1,xx,yy,dial,result1,&rx2,
      &ry2,&m2,&c2);
  if (select1 == 1 && select2 == 1)
```

```
      LineLineIntersectPoint(j,xvar,x1,xx,rx1,ry1,m1,c1,rx2,ry2,
          m2,c2);
      if ((select1 == 1 && select2 == 2) || (select1 == 2 &&
          select2 == 1))
       LineArcIntersectPoint(dim,num,ii,index,&j,&j,xvar,yvar,x1,
          y1,xx,yy,dia1,result1,rx1,ry1,xc1,yc1,d1,m1,c1,rx2,ry2,
          xc2,yc2,d2,m2,c2);
      if (select1 == 2 && select2 == 2)
       ArcArcIntersectPoint(num,ii,index,&j,&j,xvar,yvar,x1,y1,xx,
          yy,dia1,result1,rx1,ry1,xc1,yc1,d1,rx2,ry2,xc2,yc2,d2);
    } else {
      CalculateNewPoint(j-1,select1,x1,y1,xvar,yvar,dia1,-result1,
          &rx1,&ry1,&m1,&c1);
      datax[j]=rx1; datay[j]=ry1;
    } } } }

/******************** Setcode() ********************/
/*                                                 */
/* Defines variable values for selecting function */
/* which calculates cutter positions.              */
/*                                                 */
/*************************************************/
Setcode(i,j,select1,select2)
int i,j,*select1,*select2;
 {
 extern char datac[][len];

 if (strncmp(datac[i],"LINE",strlen(datac[i])) == 0) *select1=1;
 if (strncmp(datac[i],"ARC",strlen(datac[i])) == 0) *select1=2;
 if (strncmp(datac[j],"LINE",strlen(datac[j])) == 0) *select2=1;
 if (strncmp(datac[j],"ARC",strlen(datac[j])) == 0) *select2=2;
 }

/******************** CalculateNewPoint() ********************/
/*                                                           */
/* Selects a function to calculates the cutter position.    */
/*                                                           */
/*************************************************************/
CalculateNewPoint(k,select,xvar,yvar,xvar1,yvar1,dia1,result1,rx,
    ry,m,c)
double xvar,yvar,xvar1,yvar1,dia1,*rx,*ry,*m,*c;
int k,result1,select;
 {
 extern double dataxc[],datayc[];
 extern char datad[][len];

 if (select == 1)
  LineInsideOrOutside(xvar,yvar,xvar1,yvar1,dia1,result1,rx,ry,m,c);
 if (select == 2)
  ArcInsideOrOutside(xvar,yvar,dataxc[k],datayc[k],datad[k],xvar1,
      yvar1,dia1,result1,rx,ry,m,c);
 }
```

```
/********** LineInsideOrOutside() **********/
/*                                          */
/* Finds a cutter position from LINE data. */
/*                                          */
/******************************************/
LineInsideOrOutside(xs,ys,xe,ye,dial,result1,rx,ry,m,c)
double xs,ys,xe,ye,dial,*rx,*ry,*m,*c;
int result1;
 {
 extern double fabs(),sqrt();
 double delx,dely,dela,delb,val1,a,b,a1,b1,s,sa,xi,yj,ai,bj,answer;
 int loop,result2;

 dela=1;  delb=1;  val1=Limit2;
 if (fabs(xe-xs) <= Limit2) {
  IncreaseDeltaXY(ys,ye,&dely,val1);
  a1=xs+dela;  b1=ys;  *m=0;  delx=0;
 } else {
  *m=(ye-ys)/(xe-xs); IncreaseDeltaXY(xs,xe,&delx,val1);
  dely=delx**m;  b1=ys+delb;  a1=xs-(b1-ys)**m;
 } s=sqrt(delx*delx+dely*dely); xi=delx/s;  yj=dely/s;
 a=a1-xs;  b=b1-ys; sa=sqrt(a*a+b*b); ai=a/sa;  bj=b/sa;
 answer=bj*xi-ai*yj;
 if (answer < 0) result2=-1;
 else result2=1;
 if (result1 != result2) { ai=-ai; bj=-bj; }
  *rx=xs+ai*dial/2;  *ry=ys+bj*dial/2;  *c=*ry-*rx**m;
 }

/********** ArcInsideOrOutside() **********/
/*                                         */
/* Finds a cutter position from ARC data. */
/*                                         */
/*****************************************/
ArcInsideOrOutside(xs,ys,xc,yc,direct1,xe,ye,dial,result1,rx,ry,m,c)
double xs,ys,xc,yc,xe,ye,dial,*rx,*ry,*m,*c;
char direct1[len1];
int result1;
 {
 extern double fabs(),sqrt();
 double delx,dely,dela,delb,val1,a,b,a1,b1,s,sa,x,y,xi,yj,ai,bj,
        answer;
 int loop,result2;

 ArcPath(xs,ys,xc,yc,xe,ye,direct1,&x,&y,Limit3,Limit3);
 val1=Limit2;
 if (fabs(xc-xs) <= Limit2) {
  IncreaseDeltaXY(ys,yc,&delb,val1); dela=0; *m=0;
 } else {
  *m=(yc-ys)/(xc-xs); IncreaseDeltaXY(xs,xc,&dela,val1);
  delb=dela**m;
 } a1=xs+dela;  b1=ys+delb; delx=x-xs;  dely=y-ys;
 s=sqrt(delx*delx+dely*dely); xi=delx/s;  yj=dely/s;
 a=a1-xs;  b=b1-ys; sa=sqrt(a*a+b*b);  ai=a/sa;  bj=b/sa;
 answer=bj*xi-ai*yj;
```

```
 if (answer < 0) result2=-1;
 else result2=1;
 if (result1 != result2) { ai=-ai;  bj=-bj; }
 *rx=xs+ai*dial/2;  *ry=ys+bj*dial/2;  *c=*ry-*rx**m;
 }

/*************** LineLineIntersectPoint() ***************/
/*                                                      */
/* Calculates a intersect point of two linear equation. */
/*                                                      */
/********************************************************/
LineLineIntersectPoint(i,x1,x2,x3,rx1,ry1,m1,c1,rx2,ry2,m2,c2)
double x1,x2,x3,rx1,ry1,m1,c1,rx2,ry2,m2,c2;
int i;
 {
 extern double fabs(),datax[],datay[];
 extern char datac[][len],datad[][len];
 double xn,yn;

 if (fabs(x1-x2) <= Limit2) {
  xn=rx1;  yn=m2*xn+c2;
 } if (fabs(x2-x3) <= Limit2) { xn=rx2;  yn=m1*xn+c1; }
 if (fabs(x1-x2) > Limit2 && fabs(x2-x3) > limit2) {
  xn=(c2-c1)/(m1-m2);  yn=m1*xn+c1;
 } datax[i]=xn; datay[i]=yn;
 }

/*************** LineArcIntersectPoint() ****************/
/*                                                      */
/* Calculates two intersect points of linear and circle */
/* equations.                                           */
/*                                                      */
/********************************************************/
LineArcIntersectPoint(dim,num,ii,index,i,inde,x1,y1,x2,y2,x3,y3,
    dial,result1,rx1,ry1,xc1,yc1,d1,m1,c1,rx2,ry2,xc2,yc2,d2,m2,c2)
double x1,y1,x2,y2,x3,y3,dial,rx1,ry1,xc1,yc1,m1,c1,rx2,ry2,xc2,
       yc2,m2,c2;
char d1[len],d2[len],dim[];
int *num,ii,index,*i,*inde,result1;
 {
 extern double fabs(),sqrt();
 extern double datax[],datay[];
 double r,var,xn1,yn1,xn2,yn2,dist1,dist2,term,term1,term2,term3,
        term4;
 int result2,change,intx,inty;
 char d[len];

 result2=1; change=0;
 if ((intx=xc1) != 9999 && (inty=yc1) != 9999) {
  var=x1;  x1=x3;   x3=var; var=y1; y1=y3; y3=var;
  var=rx1; rx1=rx2; rx2=var; var=ry1; ry1=ry2; ry2=var;
  var=xc1; xc1=xc2; xc2=var; var=yc1; yc1=yc2; yc2=var;
  var=m1;  m1=m2;   m2=var;  var=c1;  c1=c2;   c2=var;
  strcpy(d,d1); strcpy(d1,d2); strcpy(d2,d); change=1;
 } r=(rx2-xc2)*(rx2-xc2)+(ry2-yc2)*(ry2-yc2);
```

```
 if (fabs(x1-x2) <= Limit2) {
  xn1=rx1;  var=r-(xn1-xc2)*(xn1-xc2);
  if (fabs(var) <= Limit4)  var=0;
  if (var < 0) {
   result2=0;  yn1=y2;  xn2=rx2;  yn2=y2;
   NewOrderData(num,ii,index,*i);
   InsertLineArcFilletData(i,inde,x1,y1,x2,y2,x3,y3,dia1,result1,
       xn1,yn1,xc1,yc1,d1,xn2,yn2,xc2,yc2,d2,change);
  } if (var >= 0) {
   xn2=rx1;  var=sqrt(var);  yn1=yc2+var;  yn2=yc2-var;
  } } if (fabs(x1-x2) > Limit2) {
  term1=(xc2-m1*(c1-yc2))/(1+m1*m1); term2=r*(1+m1*m1);
  term3=(m1*xc2+c1-yc2)*(m1*xc2+c1-yc2);
  term4=(1+m1*m1)*(1+m1*m1); var=(term2-term3)/term4;
  if (fabs(var) <= Limit4)  var=0;
  if (var < 0) { result2=0;
   if (change == 0) {
    xn1=rx1+x2-x1;  yn1=ry1+y2-y1;  xn2=rx2;  yn2=ry2;
  } if (change == 1) {
    xn1=rx1;  yn1=ry1;  xn2=rx2;  yn2=ry2;
  } NewOrderData(num,ii,index,*i);
   InsertLineArcFilletData(i,inde,x1,y1,x2,y2,x3,y3,dia1,result1,
       xn1,yn1,xc1,yc1,d1,xn2,yn2,xc2,yc2,d2,change);
  } if (var >= 0) {
   term=sqrt(var); xn1=term1+term;  yn1=m1*xn1+c1;
   xn2=term1-term;  yn2=m1*xn2+c1;
  } } if (result2 == 1) {
  dist1=(x2-xn1)*(x2-xn1)+(y2-yn1)*(y2-yn1);
  dist2=(x2-xn2)*(x2-xn2)+(y2-yn2)*(y2-yn2);
  if (dist1 <= dist2) { datax[*i]=xn1;  datay[*i]=yn1; }
  if (dist1 > dist2) { datax[*i]=xn2;  datay[*i]=yn2; }
 } }

/***************** ArcArcIntersectPoint() ********************/
/*                                                           */
/* Calculates two intersect points of two circle equations. */
/*                                                           */
/***********************************************************/
ArcArcIntersectPoint(num,ii,index,i,inde,x1,y1,xx,yy,x2,y2,dia1,
    result1,xs1,ys1,xc1,yc1,d1,xs2,ys2,xc2,yc2,d2)
double x1,y1,xx,yy,x2,y2,xs1,ys1,xc1,yc1,xs2,ys2,xc2,yc2,dia1;
int *num,ii,index,*i,*inde,result1;
char d1[len],d2[len];
 {
 extern double fabs(),sin(),cos(),sqrt(),atan();
 extern double datax[],datay[];
 extern int err9;
 double r1,r2,rr,a,b,c,m1,var,xa,ya,xb,yb,zeta,beta,alpha,dist1,
        dist2;
 int chk1,chk2;

 r1=(xs1-xc1)*(xs1-xc1)+(ys1-yc1)*(ys1-yc1);
 r2=(xs2-xc2)*(xs2-xc2)+(ys2-yc2)*(ys2-yc2);
 if (fabs(xc1-xc2) <= Limit4 && fabs(yc1-yc2) <= Limit4)
  { datax[*i]=xs2; datay[*i]=ys2; }
```

```
 if (fabs(xc1-xc2) > Limit4 || fabs(yc1-yc2) > Limit4) {
  b=sqrt((xc1-xc2)*(xc1-xc2)+(yc1-yc2)*(yc1-yc2));
  var=sqrt(r1)+sqrt(r2)-b;
  if (fabs(var) <= Limit2)  var=0;
  if (var < 0) {
   NewOrderData(num,ii,index,*i);
   InsertArcArcFilletData(i,inde,x1,y1,xc1,yc1,xx,yy,xs1,ys1,d1,
       dial,result1,xs2,ys2,d2);
 } if (var >= 0) {
   a=(r1-r2+h*b)/(2*b);  c=sqrt(r1-a*a); alpha=atan(c/a);
   if (fabs(xc2-xc1) <= Limit2) beta=Pi/2;
   if (fabs(xc2-xc1) > Limit2) {
    m1=(yc2-yc1)/(xc2-xc1);  beta=atan(m1); }
   zeta=alpha+beta; chk1=1;
   xa=xc1+sqrt(r1)*cos(zeta); ya=yc1+sqrt(r1)*sin(zeta);
   rr=(xc2-xa)*(xc2-xa)+(yc2-ya)*(yc2-ya);
   if (fabs(rr-r2) <= Limit4)  chk1=0;
   if (fabs(rr-r2) > Limit4) {
    xa=xc1+sqrt(r1)*cos(Pi+zeta); ya=yc1+sqrt(r1)*sin(Pi+zeta);
    rr=(xc2-xa)*(xc2-xa)+(yc2-ya)*(yc2-ya);
    if (fabs(rr-r2) <= Limit4)  chk1=0;
  } zeta=-alpha+beta; chk2=1;
   xb=xc1+sqrt(r1)*cos(zeta); yb=yc1+sqrt(r1)*sin(zeta);
   rr=(xc2-xb)*(xc2-xb)+(yc2-yb)*(yc2-yb);
   if (fabs(rr-r2) <= Limit4)  chk2=0;
   if (fabs(rr-r2) > Limit4) {
    xb=xc1+sqrt(r1)*cos(Pi+zeta); yb=yc1+sqrt(r1)*sin(Pi+zeta);
    rr=(xc2-xb)*(xc2-xb)+(yc2-yb)*(yc2-yb);
    if (fabs(rr-r2) <= Limit4)  chk2=0;
  }
   if (chk1 == 0 && chk2 == 0) {
    dist1=(xa-xx)*(xa-xx)+(ya-yy)*(ya-yy);
    dist2=(xb-xx)*(xb-xx)+(yb-yy)*(yb-yy);
    if (dist1 <= dist2) { datax[*i]=xa; datay[*i]=ya; }
    if (dist1 > dist2) { datax[*i]=xb; datay[*i]=yb; }
  } if (chk1 == 1 || chk2 == 1) {
    Space(21,13,0,64);  crt_srcp(21,13,0);
    puts(error15); err9=1;
  } } } }

/********************* NewOrderData() ***********************/
/*                                                          */
/* Rearranges all data and prepares memory for a inserting */
/* data.                                                    */
/*                                                          */
/************************************************************/
NewOrderData(num,ii,index,i)
int *num,ii,index,i;
 {
 extern double datax[],datay[],dataxc[],datayc[];
 extern int datas[],datae[];
 extern char datac[][len],datad[][len];
 int k;
```

```
 datae[ii]=datae[ii]+1;
 for (k=ii+1;k<index;++k) {
  datas[k]=datas[k]+1; datae[k]=datae[k]+1;
 } for (k=*num;k>i;--k) {
  datax[k]=datax[k-1];   datay[k]=datay[k-1];
  dataxc[k]=dataxc[k-1]; datayc[k]=datayc[k-1];
  strcpy(datac[k],datac[k-1]); strcpy(datad[k],datad[k-1]);
 } ++*num; }

/************ InsertLineArcFilletData () ************/
/*                                                  */
/* Uses for inserting fillet data when linear and */
/* circle equations have no a intersect point.    */
/*                                                  */
/****************************************************/
InsertLineArcFilletData(i,inde,x1,y1,x2,y2,x3,y3,dia1,result1,rx1,
    ry1,rxc1,ryc1,d1,rx2,ry2,rxc2,ryc2,d2,change)
double x1,y1,x2,y2,x3,y3,dia1,rx1,ry1,rxc1,ryc1,rx2,ry2,rxc2,ryc2;
char d1[len],d2[len];
int *i,*inde,result1,change;
 {
 extern double fabs(),sqrt(),cos(),sin(),atan();
 extern double datax[],datay[],dataxc[],datayc[];
 extern int datas[],datae[];
 extern char datac[][len],datad[][len];
 double r,var,xn1,yn1,rxx,ryy,m,c,zeta;
 char d[len];
 int result2;

 r=(rx2-rxc2)*(rx2-rxc2)+(ry2-ryc2)*(ry2-ryc2);
 if (change == 1)
  {
  if (strncmp(d2,"CCW",strlen(d2)) == 0)
   strcpy(d,"CW");
  if (strncmp(d2,"CW",strlen(d2)) == 0)
   strcpy(d,"CCW");
  ArcInsideOrOutside(x2,y2,rxc2,ryc2,d,x3,y3,dia1,-result1,&rxx,
      &ryy,&m,&c);
  rx2=rxx; ry2=ryy;
 } if (fabs(x1-x2) <= Limit4)  {
  xn1=x2;  var=sqrt(r);  yn1=y2+var;
  if ((yn1 <= y2 && yn1 >= y1) || (yn1 <= y1 && yn1 >= y2))
   yn1=y2-var;
 } if (fabs(x1-x2) > Limit4) {
  m=(y2-y1)/(x2-x1);  zeta=atan(m);
  xn1=sqrt(r)*cos(zeta)+x2;  yn1=sqrt(r)*sin(zeta)+y2;
  if ((xn1 <= x1 && xn1 >= x2 && yn1 <= y1 && yn1 >= y2) ||
      (xn1 <= x2 && xn1 >= x1 && yn1 <= y2 && yn1 >= y1))
   {
   xn1=sqrt(r)*cos(Pi+zeta)+x2;  yn1=sqrt(r)*sin(Pi+zeta)+y2;
   }
  }
 CrossProduct(rx1,ry1,xn1,yn1,rx1,ry1,x2,y2,&result2);
 if (change == 0)
  {
```

```
  if (result2 == 1) strcpy(d,"CCW");
  if (result2 == -1) strcpy(d,"CW");
  datax[*i]=rx1;  datay[*i]=ry1;
  datax[*i+1]=rx2;  datay[*i+1]=ry2;
 } if (change == 1) {
  if (result2 == 1) strcpy(d,"CW");
  if (result2 == -1) strcpy(d,"CCW");
  datax[*i]=rx2;  datay[*i]=ry2;
  datax[*i+1]=rx1;  datay[*i+1]=ry1;
 } dataxc[*i]=x2;  datayc[*i]=y2; strcpy(datac[*i],"ARC");
 strcpy(datad[*i],d);  ++*i;  ++*inde;
 }

/********** InsertArcArcFilletData () **********/
/*                                              */
/* Uses for inserting fillet data when two      */
/* circle equations have no a intersect point.  */
/*                                              */
/************************************************/
InsertArcArcFilletData(i,inde,x1,y1,xc,yc,x2,y2,rx1,ry1,d1,dia1,
    result1,rx2,ry2,d2)
double x1,y1,xc,yc,x2,y2,rx1,ry1,rx2,ry2,dia1;
int *i,*inde,result1;
char d1[len],d2[len];
 {
 extern double datax[],datay[],dataxc[],datayc[];
 extern int datas[],datae[];
 extern char datac[][len],datad[][len];
 double rxx,ryy,m1,c1;
 char d3[len],d[len];

 if (strncmp(d1,"CCW",strlen(d1)) == 0) strcpy(d3,"CW");
 if (strncmp(d1,"CW",strlen(d1)) == 0) strcpy(d3,"CCW");
 ArcInsideOrOutside(x2,y2,xc,yc,d3,x1,y1,dia1,-result1,&rxx,&ryy,
     &m1,&c1);
 rx1=rxx; ry1=ryy;
 if (strncmp(d1,"CCW",strlen(d1)) ==0 && strncmp(d2,"CCW",
     strlen(d2)) ==0)  strcpy(d,"CW");
 if (strncmp(d1,"CW",strlen(d1)) ==0 && strncmp(d2,"CW",
     strlen(d2)) ==0)  strcpy(d,"CCW");
 datax[*i]=rx1;  datay[*i]=ry1;  dataxc[*i]=x2;  datayc[*i]=y2;
 strcpy(datac[*i],"ARC");  strcpy(datad[*i],d);
 datax[*i+1]=rx2;  datay[*i+1]=ry2;  ++*i;  ++*inde;
 }

/************ Drill1() ************/
/*                                */
/* Defines the depth of Drilling. */
/*                                */
/**********************************/
Drill1(same,depth)
double depth;
char same[LimCha];
 {
```

```
 extern double atof();
 extern double cirx[],ciry[],cirr[],cirz[];
 extern int cin;
 char data2[MaxCha];
 int i;

 if (strncmp(same,"YES",strlen(same)) == 0) {
  for (i=0;i<cin;++i)  cirz[i]=depth;
 } if (strncmp(same,"NO",strlen(same)) == 0) {
  crt_srcp(23,3,0); printf("Depth for drilling at");
  for (i=0;i<cin;++i) {
   Space(23,25,0,50); crt_srcp(23,25,0);
   printf("(%.3f,%.3f) is <%.3f> : ",cirx[i],ciry[i],depth);
   gets(data2,MaxCha-1);
   if (data2[0] == NULL)  cirz[i]=depth;
   if (data2[0] != NULL)  cirz[i]=atof(data2);
 } } }

/****************** DrillData() ******************/
/*                                               */
/* Stores data of drilling for writting G code.  */
/*                                               */
/*************************************************/
DrillData(num,same)
int *num;
char same[LimCha];
 {
 extern double datax[],datay[],dataxc[],datayc[];
 extern char datac[][len1];
 extern double cirx[],ciry[],cirr[],cirz[];
 extern int cin;
 int i,int1,int2;

 if (cin > 0) {
  for (i=0;i<cin;++i) {
   int1=cirx[i]; int2=ciry[i];
   if (int1 != 9999 && int2 != 9999) {
    dataxc[*num]=cirx[i]; datayc[*num]=ciry[i];
    datax[*num]=cirr[i]; datay[*num]=cirz[i];
    strcpy(datac[*num],"CIRCLE"); cirx[i]=9999; ciry[i]=9999;
    ++*num;
 } } } }

/********************* WriteGcode() *********************/
/*                                                      */
/* Writes initial conditions and drillings or millings  */
/* in G code format.                                    */
/*                                                      */
/********************************************************/
WriteGcode(ind1,poind,n,compen,polar,dimen,can,drate,height,mrate,
    mdepth,tx,ty,tz,rx,ry,rz,Pock,prate,XYstep,MZdep,Sfc,Ffc,dia,
    Tpock,lpxx,lpyy,rp,pxc,pyc,ddepth)
int ind1,poind,*n;
double drate,height,mrate,mdepth,tx,ty,tz,rx,ry,rz,prate,XYstep,
       MZdep,Sfc,Ffc,dia,ddepth;
```

```
double lpxx[nut],lpyy[nut],rp[nut],pxc[nut],pyc[nut];
char compen[len],polar[len],dimen[LimCha],can[len],Pock[LimCha],
     lpock[LimCha];
 {
 extern FILE *wfpt;
 extern int lin,ain,pin,cin;
 int tnum,ind,tdrill,tmill;
 char dim[len];

 tnum=1; ind=0;
 if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
  strcpy(dim,DIMENin);
 if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
  strcpy(dim,DIMENmm);
 fprintf(wfpt,"%s%d %s %s %s %s %s %s\n",STATE,*n,RAPID,COMPENd,
         POLARd,dim,CAN80,ABSOLUTE);
 *n+=step;
 if (cin > 0) {
  fprintf(wfpt,"%s%d %s%d\n",STATE,*n,TOOL,tnum);
  *n+=step; tdrill=tnum; tnum+=1;
  fprintf(wfpt,"%s%d\n",STATE,*n); *n+=step;
 } if (lin > 0 || ain > 0) {
  fprintf(wfpt,"%s%d %s%d\n",STATE,*n,TOOL,tnum); *n+=step;
  fprintf(wfpt,"%s%d\n",STATE,*n); *n+=step; tmill=tnum;
 } fprintf(wfpt,"%s%d %s0\n",STATE,*n,TOOL); *n+=step;
 if (strncmp(dimen,"INCH",strlen(dimen)) == 0) {
  fprintf(wfpt,"%s%d %s %s%.4f\n",STATE,*n,RAPID,DimZ,tz);
  *n+=step;
  fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,RAPID,DimX,tx,
          DimY,ty);
 } if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0) {
  fprintf(wfpt,"%s%d %s %s%.2f\n",STATE,*n,RAPID,DimZ,tz);
  *n+=step;
  fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,RAPID,DimX,tx,
          DimY,ty);
 } *n+=step;
 if (cin > 0) {
  if (strncmp(Pock,"NO",strlen(Pock)) == 0) {
   Drill_Gcode(&ind,n,can,drate,dimen,height,tdrill,rx,ry,rz);
   if (strncmp(dimen,"INCH",strlen(dimen)) == 0) {
    fprintf(wfpt,"%s%d %s %s%.4f\n",STATE,*n,RAPID,DimZ,tz);
    *n+=step;
    fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,RAPID,DimX,tx,
            DimY,ty);
   } if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0) {
    fprintf(wfpt,"%s%d %s %s%.2f\n",STATE,*n,RAPID,DimZ,tz);
    *n+=step;
    fprintf(wfpt,"%s%d %s %s%.2f %s%.4f\n",STATE,*n,RAPID,DimX,tx,
            DimY,ty);
   } *n+=step;
   if (lin > 0 || ain > 0) {
    fprintf(wfpt,"%s%d %s0\n",STATE,*n,TOOL); *n+=step;
    if (strncmp(dimen,"INCH",strlen(dimen)) == 0) {
     fprintf(wfpt,"%s%d %s %s%.4f\n",STATE,*n,RAPID,DimZ,tz);
     *n+=step;
```

```
      fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,RAPID,DimX,tx,
              DimY,ty);
    } if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0) {
      fprintf(wfpt,"%s%d %s %s%.2f\n",STATE,*n,RAPID,DimZ,tz);
      *n+=step;
      fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,RAPID,DimX,tx,
              DimY,ty);
    } *n+=step;
   } } if(strncmp(Pock,"YES",strlen(Pock))==0) {
    if(strncmp(Tpock,"CIR",strlen(Tpock))==0) {
     PocketCycle_Gcode(ind1,&ind,n,prate,dimen,height,XYstep,MZdep,
     rx,ry,rz,    Sfc,Ffc,dia,tz);            /*ind1=index*/
     if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
      fprintf (wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,RAPID,DimX,
               tx,DimY,ty);
     if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
      fprintf (wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,RAPID,DimX,tx,
               DimY,ty);
     *n+=step;
    } if(strncmp(Tpock,"TAN",strlen(Tpock))==0) {
     Space(21,13,0,64); crt_srcp(21,1,0);
     printf("error in type of pocket. Incorrect data (TAN) pocket");
   } } } if (lin > 0 || ain > 0) {
   if(strncmp(Pock,"YES",strlen(Pock))==0) {
    if(strncmp(Tpock,"TAN",strlen(Tpock))==0) {
     PockRectan_Gcode(poind,&ind,n,prate,dimen,height,lpxx,lpyy,rp,
         XYstep,MZdep,rx,ry,rz,Sfc,Ffc,dia,pxc,pyc,ddepth,tz);
    /*poind!=index*/
     if (strncmp(dimen,"INCH",strlen(dimen)) == 0).
      fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,RAPID,DimX,tx,
              DimY,ty);
     if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
      fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,RAPID,DimX,tx,
              DimY,ty);
     *n+=step;
    } if(strncmp(Tpock,"CIR",strlen(Tpock))==0) {
     Space(21,13,0,64); crt_srcp(21,13,0);
     printf("error in type of pocket. Incorrect data (CIR) pocket");
  } } if(strncmp(Pock,"NO",strlen(Pock))==0) {
    Mill_Gcode(ind1,&ind,n,drate,mrate,dimen,height,mdepth,tmill,rx,
        ry,rz,tz);
    if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
     fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,RAPID,DimX,tx,
             DimY,ty);
    if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
     fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,RAPID,DimX,tx,
             DimY,ty);
    *n+=step;
  } } }

/*************** Drill_Gcode() ***************/
/*                                           */
/* Writes drilling commands in G code format. */
/*                                           */
/*********************************************/
```

```
Drill_Gcode(ind,n,can,rate1,dimen,height,tdrill,rx,ry,rz)
int *ind,*n,tdrill;
double rate1,height,rx,ry,rz;
char can[LimCha],dimen[LimCha];
 {
 extern FILE *wfpt;
 extern double datax[],datay[],dataxc[],datayc[];
 extern int datas[],datae[];
 int i;

 fprintf(wfpt,"%s%d %s%d\n",STATE,*n,TOOL,tdrill);
 *n+=step;
 if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
  fprintf(wfpt,"%s%d %s %s%s=%.4f %s=%.4f\n",STATE,*n,SPEC29,LOAD,
          VAR20,rate1,VAR21,height);
 if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
  fprintf(wfpt,"%s%d %s %s%s=%.2f %s=%.2f\n",STATE,*n,SPEC29,LOAD,
          VAR20,rate1,VAR21,height);
 *n+=step;
 fprintf(wfpt,"%s%d %s\n",STATE,*n,can); *n+=step;
  for (i=datas[*ind];i<datae[*ind];++i) {
   if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.4f %s%.4f %s%.4f\n",STATE,*n,RAPID,
            DimX,dataxc[i]-rx,DimY,datayc[i]-ry,DimZ,-datay[i]-rz);
   if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.2f %s%.2f %s%.2f\n",STATE,*n,RAPID,
            DimX,dataxc[i]-rx,DimY,datayc[i]-ry,DimZ,-datay[i]-rz);
   *n+=step;
 } fprintf(wfpt,"%s%d %s\n",STATE,*n,CAN80); *n+=step; ++*ind;
 }

/********************* Mill_Gcode() ***************************/
/*                                                            */
/* Writes initial conditions for millings in G code format.   */
/*                                                            */
/************************************************************/
Mill_Gcode(ind1,ind,n,rate0,rate1,dimen,height,depth,tmill,rx,ry,
    rz,tz)
int ind1,*ind,*n,tmill;
double rate0,rate1,height,depth,rx,ry,rz,tz;
char dimen[LimCha];
 {
 extern FILE *wfpt;
 extern double datax[],datay[],dataxc[],datayc[];
 extern char datac[][len],datad[][len],figure[][6];
 extern int datas[],datae[];
 int i,inds,inde;
 char fig1[len1];

 fprintf(wfpt,"%s%d %s%d\n",STATE,*n,TOOL,tmill); *n+=step;
 for (i=*ind;i<ind1;++i) {
  inds=datas[i]; inde=datae[i]; strcpy(fig1,figure[i]);
  Mill1_Gcode(inds,inde,rate0,rate1,dimen,fig1,n,datac,datad,
       datax,datay,dataxc,datayc,height,depth,rx,ry,rz);
  if (strncmp(dimen,"INCH",strlen(dimen)) == 0) {
```

```
    fprintf(wfpt,"%s%d %s %s%.4f\n",STATE,*n,LINEAR,DimZ,height);
    *n+=step;
    fprintf(wfpt,"%s%d %s %s%.4f\n",STATE,*n,RAPID,DimZ,tz);
  } if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0) {
    fprintf(wfpt,"%s%d %s %s%.2f\n",STATE,*n,LINEAR,DimZ,height);
    *n+=step;
    fprintf(wfpt,"%s%d %s %s%.2f\n",STATE,*n,RAPID,DimZ,tz);
  } *n+=step; }
  }

/*************** Milll_Gcode() ****************/
/*                                            */
/* Writes milling commands in G code format. */
/*                                            */
/**********************************************/
Milll_Gcode(inds,inde,rate0,rate1,dimen,figl,n,c,d,x,y,xc,yc,height,
            depth,rx,ry,rz)
int inds,inde,*n;
double rate0,rate1,x[],y[],xc[],yc[],height,depth,rx,ry,rz;
char c[][len],d[][len],dimen[LimCha],figl[6];
 {
 extern FILE *wfpt;
 int i,j;
 char direct1[len];

 if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
  {
  fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,RAPID,DimX,
          x[inds]-rx,DimY,y[inds]-ry);
  *n+=step;
  fprintf(wfpt,"%s%d %s %s%.4f\n",STATE,*n,RAPID,DimZ,height);
  *n+=step;
  fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,LINEAR,DimZ,
          -depth-rz,FEED,rate0);
  *n+=step;
  fprintf(wfpt,"%s%d %s%.4f\n",STATE,*n,FEED,rate1);
 } if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0) {
  fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,RAPID,DimX,
          x[inds]-rx,DimY,y[inds]-ry);
  *n+=step;
  fprintf(wfpt,"%s%d %s %s%.2f\n",STATE,*n,RAPID,DimZ,height);
  *n+=step;
  fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,LINEAR,DimZ,
          -depth-rz,FEED,rate0);
  *n+=step;
  fprintf(wfpt,"%s%d %s%.2f\n",STATE,*n,FEED,rate1);
 } *n+=step;
 for (i=inds+1;i<inde;++i) {
  if (strncmp(c[i-1],"LINE",strlen(c[i-1])) == 0) {
   if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,LINEAR,DimX,
            x[i]-rx,DimY,y[i]-ry);
   if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,LINEAR,DimX,
            x[i]-rx,DimY,y[i]-ry);
```

```
 } if (strncmp(c[i-1],"ARC",strlen(c[i-1])) == 0) {
   if (strncmp(d[i-1],"CCW",strlen(d[i-1])) == 0)
    strcpy(direct1,CIR_COUNTER);
   if (strncmp(d[i-1],"CW",strlen(d[i-1])) == 0)
    strcpy(direct1,CIR_CLOCK);
   if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.4f %s%.4f %s%.4f %s%.4f\n",STATE,*n,
            direct1,DimX,x[i]-rx,DimY,y[i]-ry,CEN_X,xc[i-1]-rx,
            CEN_Y,yc[i-1]-ry);
   if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.2f %s%.2f %s%.2f %s%.2f\n",STATE,*n,
            direct1,DimX,x[i]-rx,DimY,y[i]-ry,CEN_X,xc[i-1]-rx,CEN_Y,
            yc[i-1]-ry);
  } *n+=step;
  }
 if (strncmp(fig1,"CLOSE",strlen(fig1)) == 0) {
  i=inds; j=inde-1; if (strncmp(c[j],"LINE",strlen(c[j])) == 0) {
   if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.4f %s%.4f\n",STATE,*n,LINEAR,DimX,
            x[i]-rx,DimY,y[i]-ry);
   if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.2f %s%.2f\n",STATE,*n,LINEAR,DimX,
            x[i]-rx,DimY,y[i]-ry);
 } if (strncmp(c[j],"ARC",strlen(c[j])) == 0) {
   if (strncmp(d[j],"CCW",strlen(d[j])) == 0)
    strcpy(direct1,CIR_COUNTER);
   if (strncmp(d[j],"CW",strlen(d[j])) == 0)
    strcpy(direct1,CIR_CLOCK);
   if (strncmp(dimen,"INCH",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.4f %s%.4f %s%.4f %s%.4f\n",STATE,*n,
            direct1,DimX,x[i]-rx,DimY,y[i]-ry,CEN_X,xc[j]-rx,CEN_Y,
            yc[j]-ry);
   if (strncmp(dimen,"MILLIMETER",strlen(dimen)) == 0)
    fprintf(wfpt,"%s%d %s %s%.2f %s%.2f %s%.2f %s%.2f\n",STATE,*n,
            direct1,DimX,x[i]-rx,DimY,y[i]-ry,CEN_X,xc[j]-rx,CEN_Y,
            yc[j]-ry);
 } *n+=step; } }

/*************  AnotherData() ***************/
/*                                           */
/* Uses for calling Another1Data() function. */
/*                                           */
/*********************************************/
AnotherData()
 {
 extern double lxs[],lys[],lxe[],lye[];
 extern double arcxs[],arcys[],arcxe[],arcye[],arcxc[],arcyc[];
 extern char arcd[][len];
 extern int lin,ain,pin;
 extern int ilin,iain,ipin;

 Another1Data(&lin,ilin,"LINE",lxs,lys,lxe,lye);
 Another1Data(&ain,iain,"ARC",arcxs,arcys,arcxe,arcye,arcxc,arcyc,
arcd);
 }
```

```
/******************  AnotherIData() ********************/
/*                                                     */
/* Deletes some LINE or ARC data which had been used */
/* and rearranges for the rest of them.                */
/*                                                     */
/*******************************************************/
AnotherIData(ind1,ind2,code1,xs,ys,xe,ye,xc,yc,d)
double xs[],ys[],xe[],ye[],xc[],yc[];
int *ind1,ind2;
char d[][len],code1[len];
 {
 int intx,inty,i,j,k;
 double xs1[nummax],ys1[nummax],xe1[nummax],ye1[nummax],
        xc1[nummax],yc1[nummax];
 char d1[nummax][len];

 j=0;
 for (i=0;i<*ind1;++i) {
  if ((intx=xs[i]) != 9999 && (inty=ys[i]) != 9999) {
   xs1[j]=xs[i]; ys1[j]=ys[i]; xe1[j]=xe[i]; ye1[j]=ye[i];
   if (strncmp(code1,"ARC",strlen(code1)) == 0) {
    xc1[j]=xc[i]; yc1[j]=yc[i]; strcpy(d1[j],d[i]);
 }  ++j; } }
 if (j > 0) {
  i=0;
  for (k=0;k<j;++k) {
   xs[i]=xs1[k]; ys[i]=ys1[k]; xe[i]=xe1[k]; ye[i]=ye1[k];
   if (strncmp(code1,"ARC",strlen(code1)) == 0) {
    xc[i]=xc1[k]; yc[i]=yc1[k]; strcpy(d[i],d1[k]);
 } ++i;
   } } *ind1=j; }
```

ประวัติผู้เขียน

นาย สาโรช พรวิจิตรจินดา เกิดเมื่อวันอังคารที่ 2 กุมภาพันธ์ พ.ศ. 2503 จบปริญญาตรีทางวิศวกรรมเครื่องกล จากสถาบันเทคโนโลยี พระจอมเกล้า วิทยาเขตธนบุรี เมื่อ พ.ศ. 2527