ระบบช่วยสอนสำหรับหุ่นยนต์

TEACHING SYSTEM FOR ROBOT

โดย

นาย คมสัน อึ้งบุริบูรณ์ไพศาล

นาย พงษ์สิทธิ หิรัญญนิธิวัฒนา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2536

ปริญญานิพนธ์ ปีการศึกษา 2536

ภาควิชา วิศวกรรมระบบควบคุม

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง ระบบช่วยสอนสำหรับหุ่นยนต์

TEACHING SYSTEM FOR ROBOT

ผู้จัดทำ นาย คมสัน อึ้งบริบูรณ์ไพศาล 33100041

นาย พงษ์สิทธิ หิรัญญนิธิวัฒนา 33100234

(...สุเชียร...)

รศ.สุเชียร เกียรติสุนทร

อาจารย์ที่ปรึกษา

ระบบช่วยสอนสำหรับหุ่นยนต์

TEACHING SYSTEM FOR ROBOT

โดย นาย คมสัน อึ้งบริบูรณ์ไพศาล
นาย พงษ์สิทธิ หิรัญญนิธิวัฒนา

อาจารย์ที่ปรึกษา รศ.สุเชียร เกียรติสุนทร

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาการใช้งานในระบบการช่วยสอนอีกรูปแบบหนึ่ง ซึ่งมีการแสดงผลการทำงานของแขนกลจำลองแบบ Real time บนหน้าจอคอมพิวเตอร์ ทำให้ผู้ควบคุมสามารถตรวจสอบการเคลื่อนที่ของแขนกลโดยไม่ต้องใช้แขนกลจริงในการเคลื่อนที่ ผู้ควบคุมสามารถทราบทิศทางการเคลื่อนที่ของแขนกลจริงได้จากการเคลื่อนที่ของแขนกลจำลองบนหน้าจอคอมพิวเตอร์ ผู้ควบคุมสามารถตรวจสอบทิศทางให้แขนกลเคลื่อนที่ได้ก่อนที่จะให้แขนกลจริงทำการเคลื่อนที่จริง และสามารถทราบทิศทาง รวมทั้งตำแหน่งของการเคลื่อนที่ของแขนกลที่แน่นอน

ABSTRACT

THIS PROJECT IS DEVELOPMENT TO TEACHING THE ROBOTIC SYSTEM , THAT PRESENT THE OPERATING OF IMMITAGE ROBOT-ARM ON THE COMPUTER'S SCREEN IN THE REAL-TIME TYPE. THE MOVEMENT OF IMMITAGE ROBOT-ARM CAN INSPECT BY THE REAL ROBOT-ARM DOES NOT MOVE , THAT CAN CHECK THE ROBOT-WAY BEFORE THE REAL ROBOT-ARM MOVES. IT MEANS YOU CAN TEST MOVEMENT AND DIRECTION OF IMMITAGE ROBOT-ARM ON COMPUTER.

กิตติกรรมประกาศ

โครงงานระบบช่วยสอนหุ่นยนต์นี้สามารถสำเร็จลุล่วงไปด้วยดี เนื่องด้วยเพราะได้รับการเอาใจใส่ และให้คำปรึกษาในการแก้ปัญหาต่างๆที่เกิดขึ้นจาก บุคคลต่อไปนี้

อาจารย์ที่ปรึกษา รศ.สุเธียร เกียรติสุนทร

รศ.ดร.จงกล งามวิวิทย์ ที่คอยดูแลห่วงใยถึงโครงงานที่ทำอยู่เสมอ

อาจารย์เกียรติวรรณ,พี่เทพจิตต์ ที่คอยให้การสนับสนุนและดูแลอุปกรณ์เครื่องมือต่างๆ ทั้งในห้อง PROCESS ,ห้อง COMPUTER และ ห้องPROJECT ปี 4 ทุกคน รวมทั้งการให้การสนับสนุนทางด้านกำลังกาย,กำลังใจ และทุนทรัพย์ จากบิดา มารดาบังเกิดเกล้าทั้ง 2 ครอบครัว สุดท้ายขอขอบคุณกำลังใจจาก กุ๊ก,หนุ่ย,ปิ๊ง,เล็ก,ยัง ที่ให้การสนับสนุนที่พักอาศัย และปัจจัยอำนวยความสะดวกต่างๆจนทำให้ โครงงานนี้สำเร็จลุล่วงไปด้วยดี

สารบัญ

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

บทที่ 1

บทนำ

ในโลกแห่งยุคโลกานุวัตรนี้เทคโนโลยีต่างๆได้ก้าวหน้าไปไกลมากขึ้น ทำให้มนุษย์มีอุปกรณ์ที่ทันสมัยใช้กันมากขึ้น ไม่ว่าจะเป็นเครื่องมือเครื่องใช้ในการอุปโภค บริโภค หรือเครื่องจักรกลต่างๆ ที่ใช้ในอุตสาหกรรม และเกษตรกรรม เมื่อมีเครื่องมือที่เป็นเทคโนโลยีที่ทันสมัยมากขึ้น มนุษย์ก็ยิ่งต้องการความสะดวกสบายมากขึ้น แต่ก็ต้องคำนึงถึงความปลอดภัยในการใช้งานด้วยเช่นกัน ดังนั้น เครื่องจักรบางชนิดที่มีการควบคุมที่ยุ่งยาก ก็ต้องมีการดัดแปลงการควบคุมเครื่องจักรให้มีการใช้งานที่ง่ายต่อการควบคุม และสามารถทดสอบผลของการปฏิบัติงานที่จะเกิดขึ้นได้ ก่อนที่เครื่องจักรจะมีการปฏิบัติงานจริง ซึ่งจะทำให้ผู้ควบคุมสามารถตรวจสอบและวางแผนการปฏิบัติงานที่จะเกิดขึ้นจริงได้แม่นยำยิ่งขึ้น ทำให้ผู้ใช้งานมีความปลอดภัยมากขึ้นตามไปด้วย

บทที่ 2

การส่งข้อมูลด้วยพอร์ตอนุกรม

พอร์ตอนุกรม (serial port) เป็นอุปกรณ์ที่ใช้รับส่งข้อมูลที่สร้างปัญหายุ่งยากให้แก่ผู้เขียนโปรแกรมมากที่สุดอุปกรณ์หนึ่ง ซึ่งต่างจากพอร์ตแบบขนาน (parallel port) ที่มีการทำงานและให้งานที่ง่ายกว่ามาก แต่ถึงกระนั้นพอร์ตแบบอนุกรมก็ยังถูกนำมาใช้งานอย่างมาก เพราะค่าใช้จ่ายที่ถูกกว่าเมื่อเทียบกับพอร์ตแบบขนาน

ในบทนี้ จะได้อธิบายถึงพื้นฐานของการทำงานของพอร์ตแบบอนุกรมรวมทั้งการเตรียมสถานะเริ่มต้นของอุปกรณ์ (initialization) การส่งข้อมูล (transmition) การรับข้อมูล (reception) และความผิดพลาดทั่ว ๆ ไปของการรับส่งข้อมูล หลังจากนั้นจะพัฒนาโปรแกรมแอพพลิเคชั่นที่ใช้งานพอร์ตแบบอนุกรม 2 โปรแกรม ซึ่งได้แก่ โปรแกรมโอนย้ายไฟล์ (file transfer) ที่สามารถโอนย้ายไฟล์ระหว่างคอมพิวเตอร์ 2 เครื่องได้ทุกชนิดและอีกโปรกแกรมหนึ่งคือ ระบบแลน (Local Area Network: LAN) ซึ่งประกอบด้วย โปรแกรมที่ใช้ในศูนย์กลางบริการการใช้ไฟล์ หรือไฟล์เซิร์ฟเวอร์ (file server) และคำสั่งอีก 2 คำสั่งที่ใช้สำหรับเรียกใช้ไฟล์และส่งไฟล์ไปเก็บที่ไฟล์เซิร์ฟเวอร์

โปรแกรมตัวอย่างในนี้ ต้องการเครื่อง IBM PC,XT,AT หรือ PS/2 ที่ใช้ระบบปฏิบัติการของดอส

1. การรับส่งข้อมูลแบบอนุกรมอะซิงโครนัส

ก่อนที่จะได้เรียนรู้ถึงการทำงานของพอร์ตแบบอนุกรมนั้น ควรจะได้ทำความเข้าใจกับการสื่อสารแบบอะซิงโครนัส (asynchronous) เสียก่อน ในการสื่อสร้าแบบอะซิงโครนัสนั้นข้อมูลจะส่งผ่านพอร์ตแบบอนุกรมครั้งละ 1 บิต ซึ่งแตกต่างจากการส่งแบบขนานที่จะส่งครั้งหละ 1 ไบต์ และระยะเวลาที่ใช้ในการส่งข้อมูลแบบอะซิงโครนัสแต่ละไบต์นั้น ไม่จำเป็นต้องเท่ากันจึงได้ชื่อว่าการรับส่งข้อมูลแบบอะซิงโครนัส

ในการส่งข้อมูลผ่านพอร์ตแบบอนุกรมนั้น ข้อมูลแต่ละไบต์จะประกอบด้วย

1. บิตเริ่มต้น (start bit) 1 บิต
2. บิตข้อมูล (data bit) 7 หรือ 8 บิต
3. พาริตี้บิต (parity bit) (จะมีหรือไม่มีก็ได้)
4. บิตสิ้นสุด (stop bit) 1 หรือ 2 บิต

สถานะของสายส่งในขณะที่ไม่มีข้อมูลจะมีสถานะเป็นสูง(สถานะทางดิจิตอลมี 2 สถานะ คือ สูง (high) และต่ำ (low) ข้อมูลบิตใดมีค่า 0 จะทำให้สายส่งมีสถานะต่ำ ข้อมูลบิตใดมีค่า 1 ก็จะทำให้สายส่งมีสถานะสูงอยู่เช่นเดิม บิตเริ่มต้นใช้สำหรับบอกจุดเริ่มต้นของไบต์ของข้อมูล โดยการทำให้สถานะของสายส่งมีค่าต่ำ เป็นเวลา 1 รอบ (cycle) จากนั้นจะเป็นบิตของข้อมูล ตามด้วยพาริตี้บิต ซึ่งจะมีหรือไม่ก็ได้ สุดท้ายค่า บิตสิ้นสุด ซึ่งจะมี 1 หรือ 2 บิตก็ได้ ขึ้นกับว่าจะใช้เท่าใด

พาริตี้บิต ถ้าหากมีในไบต์ข้อมูล ก็จะทำหน้าที่ตรวจเช็คความผิดพลาดของข้อมูล พาริตี้มีค่า 2 อย่างคือ เป็น คู่ หรือ คี่ (even or odd) ถ้าเป็นคู่ หมายความว่า เมื่อรวมพาริตี้บิตแล้วจำนวนของบิตข้อมูลที่มีค่า 1 จะเป็นจำนวนคู่ และถ้าพาริตี้บิตเป็นคี่ หมายความว่าเมื่อรวมพาริตี้บิตแล้วจำนวนของบิตข้อมูลที่มีค่าเป็น 1 จะเป็นจำนวนคี่

อัตราการส่งข้อมูลมีหน่วยเป็น baud (bit per second) ค่า baud rate ที่ต่ำที่สุดที่มีใช้กันคือ 300 baud ซึ่งจะใช้กับโมเด็มรุ่นเก่า (โมเด็มรุ่นใหม่มักจะใช้ 1200-2400 baud) ส่วนเครื่องคอมพิวเตอร์ระดับ IBM PC สามารถใช้ค่า baud rate ได้สูงถึง 9600 baud

2. มาตรฐาน RS-232

การที่จะเข้าใจว่าปัญหามากมายที่เกิดกับพอร์ตแบบอนุกรมนั้นเกิดขึ้นได้อย่างไร และทำไมถึงเกิดขึ้นได้ จะต้องเข้าใจมาตรฐานของการสื่อสารแบบอนุกรมอะซิงโครนัสของ RS-232 มากพอสมควร ถึงแม้ว่าจะไม่ลงไปในรายละเอียดมากนักก็ตาม

พอร์ตแบบอนุกรมส่วนใหญ่ จะมีรูปร่างขึ้นอยู่กับมาตรฐานของ RS-232 คือ มีขา 25 ขา ที่คอนเน็กเตอร์แต่ละปลายสายส่ง (แต่ IBM AT จะมีเพียง 9 ขา) แต่ถึงแม้ว่าจะมีขาจำนวนเท่ากัน แต่พอร์ตส่วนใหญ่จะมีสัญญาณที่ไม่เหมือนสัญญาณของ RS-232 ทั้งหมด เพราะว่าบางขาสัญญาณออกไปเพื่อลดค่าใช้จ่าย สัญญาณพื้นฐานของ RS-232 แสดงดังตารางที่ 1

ตารางที่ 1

สัญญาณ	ชื่อย่อ	หมายเลขขา
REQUEST TO SEND	RTS	4
CLEAR TO SEND	CTS	5
DATA SET READY	DSR	6
DATA TERMINAL READY	DTR	20
TRANSMIT DATA	TxD	2
RECEIVE DATA	RxD	3
GROUND	GRD	7

สัญญาณทั้งหมดมีมากกว่านี้ เพราะว่าแรกเริ่มนั้น พอร์ตอนุกรมถูกออกแบบมาเพื่อให้งานร่วมกับโมเด็ม ดังนั้นเมื่อนำไปใช้กับอุปกรณ์อื่น บางสัญญาณจึงไม่จำเป็น เพราะสัญญาณเหล่านี้มีเพื่อใช้เป็นข้อตกลงระหว่างโมเด็มและคอมพิวเตอร์ว่า

1. คอมพิวเตอร์จะไม่ส่งข้อมูลให้แก่โมเด็ม ก่อนที่โมเด็มจะพร้อมส่งข้อมูล
2. คอมพิวเตอร์จะไม่อ่านข้อมูลจากโมเด็ม ก่อนที่โมเด็มจะพร้อม

ความผิดพลาดของกรอบข้อมูล

ความผิดพลาดของกรอบข้อมูล (framing error) คือ ความผิดพลาดของการส่งข้อ

มูล ที่เกิดจากสัญญาณนาฬิกา (clock) ที่ควบคุมการทำงานของอุปกรณ์ทั้ง 2 ด้านมีค่าไม่เท่ากัน เพราะว่าจากการทำงานของพอร์ตอนุกรม เมื่อพอร์ตได้รับบิตเริ่มต้นก็จะสุ่มอ่านค่าจากส่วนรับข้อมูล 1 ครั้งต่อ 1 รอบ เพื่ออ่านบิตต่อไป ซึ่งระยะเวลาในการสุ่มอ่านแต่ละรอบกำหนดได้จาก baud rate ถ้าหากว่าคอมพิวเตอร์ทั้ง 2 เครื่อง มีสัญญาณนาฬิกาไม่ตรงกัน คอมพิวเตอร์ด้านรับ ก็จะอ่านข้อมูลจากส่วนรับข้อมูลของตน ช้าเกินไปหรือเร็วเกินไป ก่อนที่ข้อมูลจะถูกส่งมาจากคอมพิวเตอร์ด้านส่ง ทำให้เกิดเฟรมมิงเออร์เรอร์ขึ้น

ฮาร์ดแวร์แฮนด์เชคกิ้ง

ฮาร์ดแวร์แฮนด์เชคกิ้ง (hardware handshaking) คือ วิธีที่ใช้ในการรับส่งข้อมูลแบบอนุกรมผ่านพอร์ตอนุกรม โดยจะต้องตรวจสอบดูสถานะของคอมพิวเตอร์ด้านของการรับข้อมูลว่าพร้อมจะรับข้อมูลหรือไม่ เมื่อคอมพิวเตอร์ด้านส่งพร้อมจะส่งข้อมูลให้ดังนั้นข้อมูลจะต้องไม่ถูกส่งออกไปจนกว่าสัญญาณพร้อมรับข้อมูลจะถูกส่งกลับมาจากคอมพิวเตอร์ด้านรับ สัญญาณพร้อมรับข้อมูลของคอมพิวเตอร์ด้านรับคือ clear to sen (CTS)

จะมีการตรวจสถานะ CTS ตลอดเวลาว่า พร้อมจะรับข้อมูลหรือไม่ ถ้าไม่พร้อม ก็จะรอไปเรื่อย ๆ แต่ถ้าพร้อม ก็จะส่งข้อมูลไปให้ จะทำเช่นนี้ไปเรื่อยๆ ตลอดที่ยังมีข้อมูลที่จะต้องส่งอยู่

3. ปัญหาการสื่อสาร

เพื่อที่จะให้การสื่อสารผ่านโมเด็มเป็นไปอย่างถูกต้อง สัญญาณหลาย ๆ สัญญาณถูกใช้เพื่อตรวจดูว่า ข้อมูลจะพร้อมเมื่อใดหรือข้อมูลไบต์ต่อไปจะถูกส่งมาเมื่อใด แต่ต่อมาเมื่อมีการสื่อสารผ่านคอมพิวเตอร์ สัญญาณบางสัญญาณได้ถูกตัดทิ้งไป เพื่อว่าสายสัญญาณจะได้ลดน้อยลง และลดค่าใช้จ่ายเกี่ยวกับสายส่ง สัญญาณจึงเหลือเพียง GRD, TxD และ RxD ซึ่งในทางทฤษฎีแล้ว เมื่อคอมพิวเตอร์เครื่องหนึ่งพร้อมที่จะส่งข้อมูล คอมพิวเตอร์อีกเครื่องจะต้องพร้อมที่จะรับข้อมูล แต่เมื่อลดสายสัญญาณลงแล้ว จะทำให้เกิดปัญหาตามมามากมาย ที่ยุ่งยากที่สุดปัญหาหนึ่งก็คือ ปัญหาของข้อมูลถูกเขียนทับ (overrun error)

ปัญหาข้อมูลถูกเขียนทับ

เมื่อการติดต่อผ่านพอร์ตอนุกรมใช้สายส่งเพียง 2 สายนั้น จะต้องให้กรรมวิธีพิเศษเล็กน้อย เพื่อให้พอร์ตตัวส่งเข้าใจว่าพอร์ตด้านรับพร้อมที่จะรับข้อมูลเสมอ โดยการต่อขา 6, 8 และขา 20 ของคอนเน็กเตอร์เข้าด้วยกัน วิธีการเช่นนี้เป็นการตัดฮาร์ดแวร์แฮนด์เชคกิ้งไปนั่นเอง

แต่การทำเช่นนี้จะทำให้เกิดปัญหาข้อมูลถูกเขียนทับได้ง่าย ซึ่งก็คือความผิดพลาดในการรับส่งข้อมูลอย่างหนึ่งที่เกิดขึ้นจากการที่คอมพิวเตอร์เครื่องส่ง ส่งข้อมูลใหม่มาให้คอมพิวเตอร์ด้านรับในขณะที่คอมพิวเตอร์ด้านรับยังไม่พร้อมจะรับข้อมูล เนื่องจากในขณะนั้นข้อมูลเดิมยังไม่ได้ถูกอ่านเข้าไปเก็บ แต่ข้อมูลใหม่ก็ถูกส่งมาแล้วข้อมูลเดิมจึงถูกเขียนทับไป จึงเกิดเป็นความผิดพลาดของข้อมูลขึ้น

4. การใช้งานพอร์ตอนุกรมผ่าน BIOS

การใช้งานพอร์ตอนุกรมสามารถทำได้ 3 วิธีคือ ผ่านทางดอส ผ่านทาง BIOS และสุดท้ายคือ การเขียนโปรแกรมควบคุมพอร์ตอนุกรมโดยตรง

การเรียกใช้พอร์ตอนุกรมผ่านทางดอสนั้น ไม่เหมาะสมเท่าใดนัก เพราะว่าดอสไม่มีวิธีที่จะตรวจดูสถานะของการรับส่งและตัวพอร์ตอนุกรมว่า การรับส่งข้อมูลถูกต้องหรือไม่เพียงใด

การเขียนโปรแกรมควบคุมการทำงานของพอร์ตอนุกรมโดยตรงนั้นคงไม่จำเป็น เพราะมีวิธีที่ดีกว่าและง่ายกว่าคือ การเรียกใช้ผ่าน BIOS อินเตอร์รัพต์หมายเลข 14

การเตรียมสถานะเริ่มต้นของพอร์ต

ในการเตรียมสถานะของพอร์ตอนุกรมเราสามารถทำได้ โดยผ่านอินเตอร์รัพต์หมายเลข 14 ฟังก์ชั่นหมายเลข 0 โดยมีรีจิสเตอร์ AH เป็นตัวผ่านค่าหมายเลขของฟังก์ชั่น (ในที่นี้คือ 0) รีจีสเตอร์ AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ตอนุกรม โดยมีความหมายของแต่ละบิตดังตารางที่ 2

ตารางที่ 2

หมายเลขประจำบิต	ความหมาย
7,6,5	อัตรารับส่งข้อมูล 000 = 110 baud 001 = 150 baud 010 = 300 baud 011 = 600 baud 100 = 1200 baud 101 = 2400 baud 110 = 4800 baud 111 = 9600 baud
4,3	พาริตี้ 00 หรือ 10 =ไม่มีพาริตี้ 01 = พาริตี้คี่ 11 = พาริตี้คู่
2	จำนวนของบิตที่สิ้นสุด 0 = 1 บิตสิ้นสุด 1 = 2 บิตสิ้นสุด
1,0	จำนวนบิตในข้อมูล1ไบต์ 10 = 7 บิต 11 = 8 บิต

ตัวอย่างการใช้รหัสเพื่อเตรียมสถานะของพอร์ตอนุกรมเช่น ถ้าต้องการตั้งให้พอร์ตมีอัตรา 9600 baud มีพาริตี้คู่ มี 1 บิตสิ้นสุด และใช้ 8 บิตต่อ 1 ไบต์ข้อมูล จะได้รูปแบบของรหัสดังตารางที่ 3

ตารางที่ 3

บิตที่	7	6	5	4	3	2	1	0
รหัส	1	1	1	1	1	0	1	1

เครื่องคอมพิวเตอร์โดยทั่วไปจะมีพอร์ตอนุกรมได้มากถึง 7 พอร์ตโดยหมายเลขพอร์ตที่จะใช้สามารถกำหนดผ่าน รีจิสเตอร์ DX พอร์ตแรกมีหมายเลข 0 พอร์ตต่อไปหมายเลข 1 เช่นนี้ต่อไปเรื่อย ๆ ฟังก์ชั่นที่แสดงต่อไปนี้มีชื่อว่า ini_port มีหน้าที่เตรียมสถานะของพอร์ตในระบบ

การส่งข้อมูลผ่านพอร์ตอนุกรม 1 ไบต์

อินเตอร์รัพต์หมายเลข 14 ฟังก์ชั่นหมายเลข 1 ของ BIOS ทำหน้าที่ส่งข้อมูล 1 ไบต์ออกทางพอร์ตอนุกรม หมายเลขของพอร์ตอนุกรมที่จะทำการส่งข้อมูล สามารถกำหนดได้ผ่านรีจิสเตอร์ DX และข้อมูลที่ต้องการส่งจะอยู่ในรีจิสเตอร์ AL เมื่อทำการส่งเสร็จเรียบร้อยแล้ว สถานะของการส่งข้อมูลจะปรากฏในรีจิสเตอร์ AH เพื่อใช้สำหรับตรวจดูว่า การส่งข้อมูลถูกต้องหรือไม่

ถ้าบิต 7 ของรีจิสเตอร์ AH มีค่า 1 เมื่อส่งข้อมูลเสร็จสิ้น แสดงว่าเกิดความผิดพลาดในการส่งข้อมูลขึ้น จะต้องตรวจดูสถานะของพอร์ตเพื่อดูว่าเกิดความผิดพลาดอะไร ดังในหัวข้อต่อไป

การตรวจสอบสถานะของพอร์ตอนุกรม

ฟังก์ชั่นหมายเลข 3 ของอินเตอร์รัพต์หมายเลข 14 ของ BIOS ใช้ตรวจดูสถานะของพอร์ตอนุกรม รีจิสเตอร์ DX ใช้สำหรับหมายเลขพอร์ตที่จะตรวจสอบสถานะของพอร์ตสามารถ

แปลความหมายได้จากรหัสที่ปรากฏในรีจิสเตอร์ AH และ AL ดังตารางที่ 4

ตารางที่ 4

สถานะของสายส่ง (AH)	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Data Ready	0
Overrun Error	1
Parity Error	2
Framing Error	3
Break-Detect Error	4
Transfer hold-register empty	5
Transfer shift-register empty	6
Time-out Error	7
สถานะของโมเด็ม (AL)	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Change in clear-to-send	0
Change in data-set-ready	1
Trailing-edge ring detector	2
Change in line signal	3
Clear-to-send	4
Data-set-ready	5
Ring indicator	6
Line signal detector	7

จะเห็นว่าสัญญาณสถานะต่าง ๆ ส่วนใหญ่จะใช้งานกับโมเด็ม ดังนั้นเมื่อนำมาใช้กับอุปกรณ์อื่นจึงลดความสำคัญลง แต่ก็ยังมีสัญญาณอีกสัญญาณหนึ่งที่มีความสำคัญมากก็คือ Data Ready ซึ่งจะเป็นตัวบอกว่า เมื่อไหร่ที่ข้อมูลถูกรับเข้ามา และพร้อมที่จะถูกอ่านเข้าไปเก็บ ฟังก์ชั่นในหัวข้อต่อไปมีหน้าที่อ่านข้อมูลจากพอร์ต โดยจะใช้สถานะ Data Ready นี้เป็นตัวบอกว่า ข้อมูลพร้อมที่จะถูกอ่านหรือยัง

การรับข้อมูลผ่านพอร์ตอนุกรม 1 ไบต์

การรับข้อมูลจากพอร์ตอนุกรม สามารถทำได้โดยเรียกผ่าน BIOS อินเตอร์รัพหมายเลข 14 ฟังก์ชั่นหมายเลข 2 รีจิสเตอร์ DX ใช้สำหรับกำหนดหมายเลขพอร์ต ข้อมูลที่อ่านได้จะเก็บในรีจิสเตอร์ AL เมื่ออ่านข้อมูลเสร็จแล้ว สถานะของข้อมูลและพอร์ตจะสามารถตรวจดูได้ที่บิต 7 ของรีจิสเตอร์ AH

การทำงานของฟังก์ชั่นนี้คือจะรอจนกระทั่งข้อมูลถูกรับมาผ่านพอร์ตอนุกรม แล้วส่งค่าอักษรกลับมา แต่ว่าการทำงานเช่นนี้อาจทำให้โปรแกรมไม่หลุดจากลูปการทำงาน เช่น เมื่อสายส่งของพอร์ตเกิดเสีย ดังนั้นจึงต้องตรวจสถานะของพอร์ตอนุกรมเสียก่อน ดังที่ได้อธิบายในหัวข้อที่ผ่านมา ฟังก์ชั่นใช้ตรวจสอบการกดคีย์ ถ้าหากไม่มีข้อมูลผู้ใช้ก็สามารถกดคีย์ใดคีย์หนึ่งเพื่อออกจากลูปได้ แต่ถ้ามีข้อมูลรับเข้ามา ฟังก์ชั่นก็จะผ่านไปเรียกอินเตอร์รัพต์เพื่ออ่านข้อมูลเข้ามา และเช่นเดียวกับการส่งข้อมูลบิต 7 ของรีจิสเตอร์ AH ใช้บอกว่า การอ่านข้อมูลมีข้อผิดพลาดหรือไม่

5. การโอนย้ายไฟล์ระหว่างคอมพิวเตอร์

ในปัจจุบันมักจะพบว่า ในสำนักงานหนึ่ง ๆ หรือผู้เขียนโปรแกรมที่มีคอมพิวเตอร์มากกว่า 1 เครื่อง และแต่ละเครื่องมักจะเป็นคอมพิวเตอร์คนละรุ่น ซึ่งใช้ดิสก์ขนาดต่างกัน เช่น เครื่อง PS/2 ของ IBM ใช้ดิสก์ขนาด 3 นิ้วครึ่ง ที่ต่างจากดิสก์ขนาด 5 1/4 นิ้ว ของเครื่องรุ่น PC XT และ AT การโอนย้ายข้อมูลระหว่างคอมพิวเตอร์เหล่านี้ผ่านทางพอร์ตอนุกรม เพื่อใช้ข้อมูลร่วมกันจึงสะดวกกว่า

ถึงแม้ว่าในปัจจุบัน จะมีโปรแกรมที่เกี่ยวกับการโอนย้ายไฟล์ที่เร็วกว่า และมีประสิทธิภาพมากกว่าโปรแกรมที่จะพัฒนาขึ้นในบทนี้ให้ใช้กันอย่างแพร่หลาย แต่โปรแกรมที่พัฒนาขึ้นนี้ก็มีลักษณะพิเศษหลายอย่าง เช่นสามารถใช้โอนย้ายไฟล์ได้ทุกชนิด และใช้ได้กับเครื่องคอมพิวเตอร์ทุกชนิด ถึงแม้ว่าจะมีความเร็วต่างกัน และที่พิเศษมากอย่างหนึ่งก็คือ โปรแกรมนี้ไม่ต้องใช้วิธีการฮาร์ดแวร์แฮนด์เชคกิ้ง คือการตรวจสอบสถานะทางด้านฮาร์ดแวร์แต่ใช้วิธีการซอฟต์แวร์แฮนด์เชคกิ้ง (software handshaking) ซึ่งเป็นการตรวจสถานะการส่งด้วยซอฟต์แวร์ ทำให้สามารถใช้สายส่งเพียง 3 เส้น ตามที่ได้กล่าวมาแล้ว

ซอฟต์แวร์แฮนด์เชคกิ้ง

เมื่อระบบการโอนย้ายไฟล์ ไม่มีระบบการตรวจสอบสถานะทางด้านฮาร์ดแวร์ หนทางเดียวที่จะสามารถรู้สถานะของการส่งข้อมูลว่าถูกต้องหรือไม่ ก็จะต้องทำโดยผ่านทางซอฟต์แวร์ซึ่งมีหลักการทำงานดังนี้คือ คอมพิวเตอร์ด้านส่ง เมื่อส่งข้อมูลไบต์แรกออกไปแล้ว ก็จะรอให้คอมพิวเตอร์ด้านรับ ส่งสัญญาณตอบรับกลับมาว่า รับข้อมูลเรียบร้อยแล้ว จึงจะส่งข้อมูลบิตต่อไปออกไป แล้วก็

รอรับสัญญาณตอบรับอีก เช่นนี้เรื่อยไปจนกว่าจะส่งข้อมูลจนหมด

ด้วยวิธีการนี้ การส่งข้อมูลจะไม่มีทางผิดพลาดเลย ถึงแม้ว่าคอมพิวเตอร์ทั้ง 2 เครื่อง จะมีความเร็วแตกต่างกันอย่างไรก็ตาม ข้อเสียเพียงข้อเดียวก็คือ อัตราการส่งข้อมูลจะลดลงครึ่งหนึ่ง เพราะว่าการส่งข้อมูลแต่ละไบต์ จะต้องใช้การส่งและรับอย่างละครั้ง เพื่อการตรวจสอบสถานะของคอมพิวเตอร์ด้านรับว่าพร้อมหรือไม่

ขนาดของข้อมูล 7 บิตกับ 8 บิต

หากว่าไฟล์ที่ต้องการโอนย้ายนั้นเป็นเพียงไฟล์ของตัวอักษร (text file) ไบต์ข้อมูลขนาด 7 บิตก็เพียงพอแล้ว เพราะว่าตัวอักษรและเครื่องหมายพิเศษอื่น ๆ นั้น ใช้เพียง 7 บิตต่อ 1 ตัวอักษร แต่ถ้าหากเป็นไฟล์แบบอื่นที่ไม่ใช่ไฟล์ของตัวอักษร เช่น ไฟล์ที่ทำงานได้ (executable file) ซึ่งจะใช้งานบิตที่ 8 ด้วย โปรแกรมการโอนย้ายไฟล์จะต้องส่งบิตที่ 8 ด้วย

ปัญหาที่ตามมาก็คือ รหัส EOF (end of file) ไม่สามารถนำมาเป็นตัวบอกจุดสิ้นสุดของไฟล์แบบไบนารีได้ ทางแก้ก็คือจะต้องหาขนาดของไฟล์ที่จะส่ง แล้วทำการส่งข้อมูลเป็นจำนวนตามที่คำนวณได้นั้น

บทที่ 3

การติดต่อกับเมาส์

อุปกรณ์ที่ทำหน้าที่รับข้อมูลของคอมพิวเตอร์ที่นิยม นอกเหนือจากคีย์บอร์ดก็คือ เมาส์ ซึ่งเป็นอุปกรณ์ที่หาได้ง่าย แต่ละแบบใช้เทคโนโลยีที่คล้ายคลึงกัน เช่น จะมีลูกบอลกลม ๆ อยู่ภายใน (ใช้รับรู้การเคลื่อนไหว) เมาส์ เริ่มนิยมใช้ตั้งแต่มีการพัฒนาเครื่องคอมพิวเตอร์ตระกูล Apple ให้ใช้เมาส์ได้ และมีระบบปฏิบัติการติดต่อกับ Icon จนกระทั่งกลายมาเป็นเครื่องตระกูล MacIntosh ซึ่งยังมีเมาส์และระบบปฏิบัติการที่พัฒนามาจากเครื่อง Apple

ส่วนเครื่องตระกูล IBM เดิมเมาส์เป็นเพียงอุปกรณ์เสริมสำหรับเครื่อง PC เท่านั้น จนกระทั่งมีการเปิดตัวเครื่อง IBM PS/2 ซึ่งมีพอร์ตสำหรับเมาส์และเมาส์มาพร้อมกับเครื่องด้วย หลังจากนั้นเมาส์ก็กลายเป็นอุปกรณ์ที่สำคัญอย่างหนึ่งบนเครื่อง PC

แต่อย่างไรก็ตามทุกคนก็ไม่ได้คิดว่าการใช้เมาส์นั้นดีที่สุด มีบางคนไม่เห็นด้วย เนื่องจากว่าผู้เขียนโปรแกรมหรือผู้ใช้บางคนไม่ชอบการสื่อสารกับคอมพิวเตอร์แบบ Icon User Interface และเนื่องจากว่าการสื่อสารแบบนี้ในยุคแรกต้องทำงานร่วมกับเมาส์ แต่อย่างไรก็ตามเมาส์สามารถใช้ประโยชน์ได้หลายด้าน โดยเฉพาะการทำงานร่วมกับโปรแกรมในโหมดกราฟฟิก ซึ่งทุกคนจะเห็นพ้องต้องกันว่าการใช้เมาส์กับโปรแกรมประเภทนี้เหมาะสมที่สุด

เมาส์ในปัจจุบันนี้มีหลายแบบ ซึ่งจะมีการทำงานแตกต่างกันออกไป โปรแกรมในบทนี้จะใช้กับเมาส์ 2 ปุ่มของไมโครซอฟต์ซึ่งมีการทำงานแต่ต่างจากเมาส์ 3 ปุ่มของ IBM PS/2 เล็กน้อย (แต่ก็สามารถดัดแปลงรูทีนให้ใช้กับเมาส์ 3 ปุ่มได้) ทางที่ดีควรมีหนังสือ Microsoft mouse Programmer's reference guide ซึ่งมีแผ่นดิสก์ติดมากับหนังสือ ในดิสก์จะมีไฟล์ที่เป็นไลบรารี่ (library) ชื่อ mouse.lib ซึ่งเป็นฟังก์ชั่นระดับต่ำที่ทำงานเกี่ยวกับเมาส์

1. พื้นฐานเกี่ยวกับเมาส์

ก่อนที่จะใช้เมาส์ได้นั้น ต้องติดตั้ง device driver ซึ่งสำหรับเมาส์ของไมโครซอฟต์

นั้น ส่วนใหญ่ทำการติดตั้งในไฟล์ config.sys โดยในไฟล์นี้จะมีบรรทัดหนึ่งเป็นข้อความดังนี้

device = mouse.sys

หมายเหตุ mousc.sys อาจเป็น msmouse.sys หรือไฟล์อื่นที่ทำหน้าที่เดียวกันก็ได้

สำหรับเมาส์แบบ 3 ปุ่มของ IBM นั้น เราจะต้องเรียกโปรแกรม mouse.com (หรือโปรแกรมอื่นที่ทำหน้าที่เดียวกัน) ก่อน โดยอาจใส่ไว้ในไฟล์ autoexec.bat ก็ได้

เมื่อติดตั้งไดรเวอร์ของเมาส์เรียบร้อยแล้ว เมื่อมีการเคลื่อนหรือกดปุ่มเมาส์ จะเกิดอินเทอร์รัพต์ 33H เกิดขึ้นและไดรเวอร์จะทำการตั้งค่าภายในต่าง ๆ และส่งออกมา ซึ่งเหตุการณ์นี้จะเกิดเมื่อมีการเคลื่อนไหวหรือกดปุ่มเมาส์เท่านั้น ซึ่งเมาส์ในอุดมคติจะต้องไม่ทำให้เกิดผลกระทบใด ๆ ต่อประสิทธิภาพการทำงานของคอมพิวเตอร์

และเช่นเดียวกับคีย์บอร์ดต้องมีเคอร์เซอร์ (บางครั้งเรียกว่าพอยน์เตอร์) เมาส์ก็เช่นกัน ในไลบรารี่ของเมาส์ของไมโครซอฟต์มีรูทีนที่ทำหน้าที่กำหนดลักษณะของเคอร์เซอร์ ซึ่งจะเป็นลูกศรในโหมดกราฟฟิกและเป็นเคอร์เซอร์ขนาดเท่าตัวอักษรในโหมดตัวอักษร เคอร์เซอร์จะบอกตำแหน่งของเมาส์บนจอภาพขณะนั้นเหมือนกับเคอร์เซอร์ของคีย์บอร์ด แต่เคอร์เซอร์ของเมาส์สามารถจะกำหนดให้ปรากฏหรือไม่ก็ได้ ซึ่งโดยปกติจะปรากฏให้เห็นเฉพาะตอนที่อยู่ในโปรแกรมส่วนที่ใช้เมาส์เท่านั้น เมื่ออยู่ในโปรแกรมส่วนอื่นก็จะไม่ปรากฏให้เห็น

ถึงแม้ลักษณะของกายภาพของเมาส์จะแตกต่างกัน แต่เมาส์ก็สามารถทำงานเชื่อมต่อกับจอภาพได้ เพราะไดรเวอร์ของเมาส์จะทำการนับว่าเมาส์เคลื่อนที่จากตำแหน่งเดิมไปเท่าไรโดยอัตโนมัติ ดังนั้นเมื่อเลื่อนเมาส์ไปทางใด เคอร์เซอร์บนจอภาพก็จะเคลื่อนที่ตามไปในทิศทางเดียวกัน

การวัดระยะทางของเมาส์ที่เคลื่อนที่ไปใช้ หน่วยเป็นมิกกี้ (mickey) ซึ่ง 1 มิกกี้เท่ากับ 1/200 นิ้ว แต่โดยทั่วไปแล้วไม่จำเป็นต้องทราบระยะทางที่เคลื่อนที่ไปจริง ๆ ว่าเป็นเท่าไร

2. จอภาพจริงกับจอภาพจำลอง

รูทีนของเมาส์ของไมโครซอฟต์ที่อยู่ในไลบรารี่จะทำงานกับจอภาพจำลอง (virtual screen) ซึ่งอาจจะมีจุดต่างๆแตกต่างกับลักษณะทางกายภาพของจอภาพจริง เมื่อมีการเคลื่อนเมาส์ ค่าตัวนับที่ทำหน้าที่เก็บตำแหน่งของเคอร์เซอร์จะเปลี่ยนแปลง การแสดงเคอร์เซอร์ ต้องนำเคอร์-เซอร์จากจอภาพจำลองย้ายตำแหน่ง (map) ไปปรากฏบนจอภาพจริง ในการแสดงผลโหมด 6,14,15 และ 16 เป็นการ map แบบ one-to-one สำหรับการแสดงผลในโหมด 4 และ 5 นั้น จะ map เฉพาะตำแหน่งในแนวนอนจากจอภาพจำลองไปจอภาพจริงเท่านั้น ข้อควรระวังคือ โปรแกรม PAINT ที่ดัดแปลงให้ใช้กับเมาส์ได้นั้นทำงานได้ในโหมดกราฟิกโหมด 4 เท่านั้น

3. ฟังก์ชั่นในไลบรารี่ของเมาส์

รูทีนภายใน mouse.lib จะทำงานโดยมีการเรียกใช้ฟังก์ชั่นตัวหนึ่ง การเรียกใช้ต้องส่งหมายเลขของฟังก์ชั่นของเมาส์ที่ต้องการใช้ (คล้ายกับการเรียกใช้ function call ของดอสหรืออินเทอร์รัพต์ 21H ที่ต้องส่งหมายเลขของฟังก์ชั่นที่ใช้งานให้ให้ด้วย)

ไมโครซอฟต์ได้กำหนดฟังก์ชั่นของเมาส์ไว้ถึง 30 ฟังก์ชั่น แต่สำหรับที่โปรแกรม PAINT ใช้มีเพียง 2-3 ฟังก์ชั่นเท่านั้น ซึ่งมีดังนี้

ฟังก์ชั่นเซตและแสดงสถานะ

ฟังก์ชั่นนี้เป็นฟังก์ชั่นหมายเลข 0 ทำหน้าที่รีเซตเมาส์ วางตำแหน่งของเคอร์เซอร์ไว้ที่กลางจอภาพและไม่แสดงเคอร์เซอร์ ค่าที่ส่งกลับจะเป็นจำนวนปุ่มของเมาส์ที่ใช้ขณะนั้นผ่านมาทาง arg2 ส่วนค่าที่ส่งผ่านตัวแปรที่ fnum ส่งกลับมาจะเป็น 0 ถ้าไม่ได้ติดตั้งเมาส์หรือโปรแกรม driver ถ้ามาการติดตั้งทั้งสองอย่างจะส่งค่า -1 กลับมา

ฟังก์ชั่นแสดงเคอร์เซอร์

ฟังก์ชั่น 1 มีหน้าที่ทำให้เคอร์เซอร์ของเมาส์ปรากฏ ไม่มีการส่งค่ากลับ

ฟังก์ชั่นปิดเคอร์เซอร์

ฟังก์ชั่น 2 ทำให้เคอร์เซอร์ไม่ปรากฏบนจอภาพ ไม่มีการส่งค่ากลับเช่นกัน

ฟังก์ชั่นอ่านสถานะของปุ่มและสถานะของเคอร์เซอร์

ฟังก์ชั่นหมายเลข 3 จะส่งค่าสถานะของปุ่มของเมาส์ซึ่งจะบอกว่าขณะนั้นปุ่มใดถูกกด ผ่านมาทาง arg2 ตำแหน่งของเคอร์เซอร์ในแกนนอนผ่านมาทาง ang3 ตำแหน่งในแกนตั้งใน arg4

สถานะของปุ่มที่ถูกกดหาได้จากค่าของ arg2 ที่บิตที่ 0 และบิตที่ 1 ถ้าบิตที่ 0 เป็น 1 หมายความว่าปุ่มซ้ายถูกกด และถ้าบิตที่ 1 เป็น 1 แสดงว่าปุ่มทางขวาถูกกด แต่ถ้าบิตทั้งสอบเป็น 0 แสดงว่าไม่มีปุ่มใดถูกกด

ฟังก์ชั่นกำหนดตำแหน่งของเคอร์เซอร์

ฟังก์ชั่นหมายเลข 4 จะวางเคอร์เซอร์ของเมาส์ตามตำแหน่งที่ต้องการ ค่าของตำแหน่งของเคอร์เซอร์ในแกนนอนจะส่งผ่านมาทางตัวแปร ang3 และตำแหน่งของแกนตั้งผ่านมาทาง ang4 ซึ่งตำแหน่งที่กำหนดนี้ต้องไม่เกินขอบเขตของจอภาพจำลองที่กำหนดไว้

ฟังก์ชั่นระบุการเคลื่อนที่

ฟังก์ชั่น 11 จะทำการส่งค่าที่แตกต่างของตำแหน่งของเคอร์เซอร์ของเมาส์ ระหว่างตำแหน่งของเคอร์เซอร์ที่มีการเรียกใช้ฟังก์ชั่นนี้ครั้งสุดท้ายกับตำแหน่งของเคอร์เซอร์ขณะนั้น เป็นค่าในแนวนอนและแนวตั้ง ค่าในแนวนอนจะส่งกลับผ่านมาทาง arg3 ส่วนค่าในแนวตั้งผ่านมาทาง ang4 ถ้าไม่มีการเปลี่ยนแปลงตำแหน่งค่าทั้งสองจะเท่ากับ 0 ถ้ามีการเลื่อนเมาส์ค่าทั้งสองจะไม่เท่ากับ 0 และนอกจากนั้นเมื่อมีการมาเรียกใช้ฟังก์ชั่นนี้จะมีการรีเซตค่าเคาน์เตอร์ภายในฟังก์ชั่นให้เป็น 0

ค่าในแนวตั้งเป็นค่าบวกแสดงว่าเมาส์ถูกเลื่อนลง ถ้าเป็นค่าลงแสดงว่าเมาส์ถูกเลื่อนขึ้น ส่วนค่าในแนวตั้งถ้าค่าเป็นค่าบวกแสดงเมาส์ถูกเลื่อนไปทางขวา ถ้าค่าเป็นค่าลงแสดงว่าเลื่อนไปทางซ้าย

บทที่ 4

หลักการทำงานของโปรแกรม

การทำงานในโปรแกรมหลัก

- เมื่อเริ่มต้นเข้าสู่โปรแกรม เราจะต้องทำการสอบสภาวะของจอภาพว่าเป็นจอภาพชนิดใด ซึ่งในโปรแกรมนี้ จะต้องใช้จอภาพชนิด EGA (HI)ขนาด 640*320 เพราะเนื่องจากว่าการกระทำการสร้างรูปของโปรแกรมทางกราฟฟิก เพื่อให้ภาพเคลื่อนไหวนิ่งมากที่สุด คุณสมบัติของจอภาพ EGA (HI) สามารถทำการใช้ page จอภาพได้ถึง 2 page

- ทำการสร้างสถานะของ Mouse ใช้พร้อมที่จะทำงาน ก่อนที่จะทำการสร้างสถานะของ Mouse จะต้องทำการเรียก Driver ของ Mouse ก่อน ซึ่งจะต้องเรียกใช้ Driver ของ Mouse ก่อนเข้าสู่โปรแกรม

- ทำการกำหนดตำแหน่ง Pointer ของ Mouse ให้อยู่กลางจอภาพ โดยตำแหน่งปัจจุบันของ Pointer Mouse ในจอภาพระนาบแกน x จะถูกเก็บไว้ในรีจีสเตอร์ Reg.CX เสมอ โดยตำแหน่งปัจจุบันของ Pointer Mouse ในจอภาพระนาบแกน y จะถูกเก็บไว้ในรีจีสเตอร์ Reg.DX เช่นกัน ดังนั้นจึงควรทำการเก็บค่ารีจีสเตอร์ Reg.CX และ Reg.DX ลงใน Xold,Yold ตามลำดับเพื่อใช้ในการคำนวณต่อไป

- ทำการ Set ค่าเริ่มต้นของตัวแปรที่จะใช้ในโปรแกรม มีดังนี้

ang คือ ตัวที่แทนค่ามุมที่หมุนแขนกลทั้งแขน((โดยมีค่า ang=0)

p คือ ตัวแปรที่แทนลำดับในการเก็บข้อมูลเพื่อแสดงการเคลื่อนที่ของแขนกล

a คือ ตัวแปรที่แทนค่ามุมในส่วน UPPER Arm ที่ทำกับระนาบ x ซึ่งต้องนำไปคำนวนร่วมกับ b

b คือ ตัวแปรที่แทนค่ามุมที่คำนวณได้จากการเคลื่อนที่ของแขน

orx คือ ตัวแปรที่แทนค่าตำแหน่งในระนาบแกน x ของแขนกล

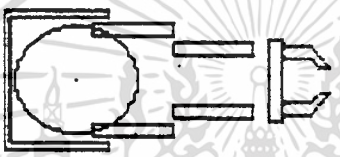
lx คือ ตัวแปรที่แทนขนาดความยาวของแขนในส่วน UPPER ARM ที่ฉายลงบนแกน x

tx คือ ตัวแปรที่แทนขนาดความยาวของแขนในส่วน UPPER ARM ที่ฉายลงบนแกน x และ FORE ARM

tx1 คือ ตัวแปรที่แทนขนาดความยาวของแขนในส่วน FORE ARM ที่ฉายลงบนแกน x

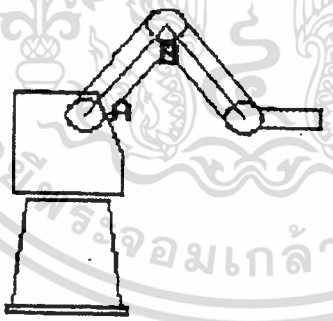
hag คือ ตัวแปรที่แทนค่ามุมที่หมุน Gripper

- ทำการสร้างภาพในระนาบ x,y ซึ่งแสดงภาพด้านบนของแขนกล



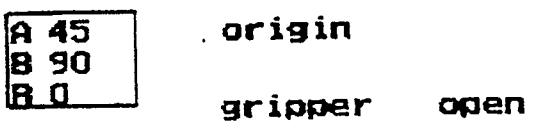
รูปที่ 1 แสดงการสร้างภาพด้าน TOP VIEW ของแขนกล

- ทำการสร้างภาพในระนาบ x,z ซึ่งแสดงภาพด้านข้างของแขนกล



รูปที่ 2 แสดงการสร้างภาพด้าน SIDE VIEW ของแขนกล

- ทำการแสดงค่าของ orx,a,b,ang



รูปที่ 3 แสดงค่าของ orx,a,b,ang

- เข้าสู่ While loop โดยการตรวจสอบค่ารีจีสเตอร์ Reg.BX (คือ กดปุ่ม Mouse)ถ้าทำการกดปุ่มทางด้านซ้ายของ Mouse จะทำการออกจากโปรแกรมที่ทำงานอยู่

- เมื่อโปรแกรมเริ่มทำงาน ทำการตรวจสอบสถานะของ Mouse ว่ามีการเปลี่ยนแปลงหรือไม่ และทำการตรวจสอบสถานะของ KEYBOARD ว่ามีการเปลี่ยนแปลงหรือไม่ ถ้ามีการเปลี่ยนแปลง ให้คำนวณผลจากการเปลี่ยนแปลงนั้น แล้วนำไปสร้างภาพ

- เมื่อต้องการเก็บข้อมูลจากการสร้างภาพที่ต้องการในการนำไปใช้งาน สามารถเก็บข้อมูลภาพและตำแหน่งได้โดยการกด KEYBOARD (ในโปรแกรมนี้สามารถเก็บข้อมูลได้ 20 ตำแหน่ง)

- เมื่อต้องการแสดงผลข้อมูลที่เก็บไว้มาแสดงเป็นภาพการเคลื่อนไหวที่เก็บได้ เพื่อนำไปใช้ในการขยับแขนกล โดยการทำทีละตำแหน่งที่เก็บตามลำดับ ด้วยการกด KEYBOARD (ก่อนที่จะใช้คำสั่งนี้ได้ต้องมีการเก็บข้อมูลเสียก่อน และต้องตรวจสอบก่อนว่าแขนกลอยู่ในสภาวะเริ่มต้น (HOME) หรือไม่

```
BEGIN
     SetG(GraphDriver,GraphMode);
     InitMouse(reg);
     Move_mou_prt(reg,getmaxx div 2, getmaxy div 2);
     xold :=reg.CX ;
     yold :=reg.DX ;
     ang := 0;
     p   := 0;
     a   := 45;
     b   := 0;
     orX := 320;
     lx  := 29;
     tx  := 58;
     tx1 := 29;
```

```
hag := 0;
grip:= 'open';
body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
display(orX,p,a,b,ang);
GoHome;
LOOP รับค่า MOUSE;
LOOP รับค่า KEYBOARD;
ทำการประมวลผลและทำการแสดงผล;
Closegraph;
END.
```

การทำงานในโปรแกรมย่อย

- โปรแกรมย่อยการรับค่า KEYBOARD

เริ่มต้นทำการตรวจสอบว่ามีการกดค่า KEYBOARD หรือไม่ ถ้ามีให้เริ่มทำการตรวจสอบว่า KEY ที่กดนั้นเป็น KEY อะไร โดยมีค่า KEY ดังต่อไปนี้

```
if keypressed then
begin   ch := readkey;
     if ch = #0 then ch := readkey;
```

1) KEY ที่ 0075 คือ การเปลี่ยนแปลงค่า KEY ที่แสดงผลการเคลื่อนที่ของแขนกลทั้งแขนไปทางซ้ายของ Slide way โดยการแทนค่าการเปลี่ยนแปลงที่ลดลงด้วยตัวแปร orx นำค่า orx ที่เปลี่ยนแปลงไปคำนวณหาค่า orx ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป (โดยค่า orx จะต้องไม่ขอบเขตความยาวขนาดของ Slide way ถ้าเกินให้หยุด)

```
if ch = #75 then
   begin
```

```
orX := orX - 4;
if (orX<224) then orX := 224;
ทำการคำนวณและสร้างภาพ;
end;
```

2) KEY ที่ 0077 คือ การเปลี่ยนแปลงค่า KEY ที่แสดงผลการเคลื่อนที่ของแขนกลทั้งแขนไปทางขวาของ Slide way โดยการแทนค่าการเปลี่ยนแปลงที่เพิ่มขึ้นด้วยตัวแปร orx นำค่า orx ที่เปลี่ยนแปลงไปคำนวณหาค่า orx ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป (โดยค่า orx จะต้องไม่ขอบเขตความยาวขนาดของ Slide way ถ้าเกินให้หยุด)

```
if ch = #77 then
   begin
   orX := orX + 4;
   if (orX>416) then orX := 416;
   ทำการคำนวณและสร้างภาพ;
   end;
```

3) KEY ที่ 0072 คือ การเปลี่ยนแปลงค่า KEY ที่แสดงผลการเคลื่อนที่ขึ้นในทิศทางกวาดเป็นวงกลมของแขนกล โดยการแทนค่าการเปลี่ยนแปลงที่เพิ่มขึ้นด้วยตัวแปร b นำค่า b ที่เปลี่ยนแปลงไปคำนวณหาค่า b ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป (โดยค่า b จะต้องไม่ของเขตความยาวขนาดของ Slide way ถ้าเกินให้หยุด)

```
if ch = #72 then
   begin
   b := b + 1;
   ทำการคำนวณและสร้างภาพ;
   end;
```

4) KEY ที่ 0080 คือ การเปลี่ยนแปลงค่า KEY ที่แสดงผลการเคลื่อนที่ลงในทิศทางกวาดเป็นวงกลมของแขนกล โดยการแทนค่าการเปลี่ยนแปลงที่ลดลงด้วยตัวแปร b นำค่า b ที่เปลี่ยนแปลงไปคำนวณหาค่า b ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป

```
if ch = #80 then
   begin
   b := b - 1;
   ทำการคำนวณและสร้างภาพ;
   end;
```

5) KEY ที่ 0073 คือ การเปลี่ยนแปลงค่า KEY ที่แสดงผลการหมุนขึ้น ของ Gripper โดยการแทนค่าการเปลี่ยนแปลงที่เพิ่มขึ้นด้วยตัวแปร hag โดยนำค่า hag ที่เปลี่ยนแปลงไปคำนวณหาค่า hag ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป

```
if ch = #73 then
   begin
   hag := hag + 90;
   if hag >180 then hag := 180;
   ทำการคำนวณและสร้างภาพ;
   end;
```

6) KEY ที่ 0081 คือ การเปลี่ยนแปลงค่า KEY ที่แสดงผลการหมุนลง ของ Gripper โดยการแทนค่าการเปลี่ยนแปลงที่ลดลงด้วยตัวแปร hag โดยนำค่า hag ที่เปลี่ยนแปลงไปคำนวณหาค่า hag ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป

```
if ch = #81 then
   begin
   hag := hag - 90;
   if hag < 0 then hag := 0;
```

```
ทำการคำนวณและสร้างภาพ;
end;
```

7) KEY ที่ 0032 {SPACE BAR} คือ KEY ที่ทำการเก็บข้อมูลรูปภาพและตำแหน่งที่ต้องการ โดยลำดับการเก็บข้อมูลจะถูกแสดงด้วยตัวแปร p โดยตัวแปร p จะเป็นตัวกำหนดลำดับที่เก็บข้อมูลในปัจจุบันและข้อมูลที่เก็บมีดังนี้คือ orx,lx,lz,tx,tx1,tz,xold,yold,ang,a,b,hag,grip ซึ่งจะเป็นค่าและตำแหน่งปัจจุบัน เมื่อทำการเก็บข้องมูลให้ทำการถ่ายค่าตัวแปรดังกล่าวนี้ลงใน Record ของ RPS และระบุลำดับของตำแหน่งและภาพนั้นด้วยตัวแปร p ส่วนตัวแปร f จะเป็นตัวแปรที่เก็บค่าลำดับของ p

```
if ch = #32 then    {space bar}
   begin
   p := p+1;
   ทำการเก็บข้อมูลรูปภาพและตำแหน่งดังนี้ คือ
   orx,lx,lz,tx,tx1,tz,xold,yold,ang,a,b,hag,grip;
   f := p;
   end;
```

8) KEY ที่ 0013 {ENTER} คือ KEY ที่แสดงผลข้อมูลที่เก็บไว้ มาแสดงเป็นภาพการเคลื่อนไหวที่เก็บไว้ เพื่อนำไปใช้ในการเคลื่อนที่ของแขนกล โดยจะทำการเริ่มต้นทำงาน การแสดงภาพและตำแหน่งพร้อมทั้งทำการเคลื่อนที่แขนกลตามที่ได้เก็บข้อมูลไว้ตามลำดับเก็บข้อมูล

```
if ch = #13 then {enter}
   begin
   for p := 1 to f do
   Begin
if (p > 1) then move_it49(r(RPS[p].ang),r(RPS[p-1].ang))
          else move_it49(r(RPS[p].ang),0);
```

```
        if (p > 1) then move_it50(r(RPS[p].a),r(RPS[p-1].a))
                   else move_it50(r(RPS[p].a),45);
        if (p > 1) then move_it51(r(RPS[p].a),r(RPS[p-1].a))
                   else move_it51(r(RPS[p].a),45);
        if (p > 1) then move_it54(r(RPS[p].orX),r(RPS[p-1].orX))
                   else move_it54(r(RPS[p].orX),320);
        gripper(RPS[p].grip);
                End;
                end;
```

- โปรแกรมย่อยของการรับค่า Pointer ของ Mouse

เริ่มต้นทำการตรวจสอบ สถานะ Pointer ของ Mouse ว่ามีการเปลี่ยนแปลงหรือไม่ ถ้ามีการเปลี่ยนแปลง ให้ตรวจสอบการเปลี่ยนแปลงนั้นว่ามีแนวโน้มการเปลี่ยนแปลงในในทิศทางแนวใด

```
WHILE reg.BX<>1 DO
```

- ถ้ามีแนวโน้มในการเปลี่ยนแปลงในทิศทางแกน y มากกว่า แกน x ให้ทำการตรวจสอบว่า Pointer ของ Mouse มีการเปลี่ยนแปลงทิศทางในการเพิ่มขึ้นหรือลดลง

```
BEGIN
  IF(xold=reg.CX)OR(yold=reg.DX) THEN
  BEGIN
    REPEAT
    Get_ptr_posit_botton_status(reg);
    UNTIL (reg.CX<>xold)OR(reg.DX<>yold)OR(keypressed);
  END;
END;
```

- ถ้ามีการเปลี่ยนแปลงในทิศทางเพิ่มขึ้น หมายถึง การเคลื่อนที่ของหมุนแขนกลทั้งแขนไปในทิศทางทวนเข็มนาฬิกา โดยการเปลี่ยนแปลงที่เพิ่มขึ้นนั้นแทนค่าด้วยตัวแปร ang และนำค่า ang ที่เปลี่ยนแปลงไปคำนวณหาค่า ang ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป โดยค่า ang จะต้องกวาดเป็นมุมไม่เกิน 159 องศา

```
IF(xold<>reg.CX)OR(yold<>reg.DX) THEN
  BEGIN
  If( abs(yold-reg.DX) > abs(reg.CX-xold) ) then
    Begin
    if(yold>reg.DX) then
     begin
        ang := ang+4;
        if (ang>159) then ang := 159;
    end;  end;
  END;
```

- ถ้ามีการเปลี่ยนแปลงในทิศทางลดลง หมายถึง การเคลื่อนที่ของหมุนแขนกลทั้งแขนไปในทิศทางตามเข็มนาฬิกา โดยการเปลี่ยนแปลงที่ลดลงนั้นแทนค่าด้วยตัวแปร ang และนำค่า ang ที่เปลี่ยนแปลงไปคำนวณหาค่า ang ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป โดยค่า ang จะต้องกวาดเป็นมุมไม่เกิน 159 องศา

```
    if(yold<reg.DX) then
    begin
        ang := ang-4;
        if (ang<-159) then ang := -159;
    end;
```

- ถ้ามีแนวโน้มในการเปลี่ยนแปลงในทิศทางแกน x มากกว่าแกน y ให้ทำการตรวจสอบว่า Pointer ของ Mouse มีการเปลี่ยนแปลงทิศทางในการเพิ่มขึ้นหรือลดลง

- ถ้ามีการเปลี่ยนแปลงในทิศทางเพิ่มขึ้น หมายถึง การเคลื่อนที่ของแขนกลในทิศทางยืดแขนออกโดยการเปลี่ยนแปลงที่เพิ่มขึ้นนั้นแทนค่าด้วยด้วยแปร a และนำค่า a ที่เปลี่ยนแปลงไปคำนวณหาค่า a ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป

```
If( abs(yold-reg.DX) < abs(reg.CX-xold) ) then
  Begin
  if(xold>reg.CX) then
  begin
  a  := a+1.875;
     if a>75 then a:= 75;
  end;
```

- ถ้ามีการเปลี่ยนแปลงในทิศทางลดลง หมายถึง การเคลื่อนที่ของแขนกลในทิศทางหดแขนเข้าโดยการเปลี่ยนแปลงที่เพิ่มขึ้นนั้นแทนค่าด้วยด้วยแปร a และนำค่า a ที่เปลี่ยนแปลงไปคำนวณหาค่า a ณ ตำแหน่งใหม่ เพื่อนำไปทำการสร้างภาพต่อไป

```
  if(xold<reg.CX) then
  begin
  a  := a-1.875;
  (  if a<0  then a := 0;
  end;
```

<u>หมายเหตุ</u> การเคลื่อนที่ Pointer ของ Mouse นี้เมื่อสุดขอบจอภาพในแต่ละระนาบบนจอภาพให้ทำการกำหนด Pointer ของ Mouse ให้ไปอยู่ในตำแหน่งเริ่มต้น หรือตำแหน่งปลายของจอภาพ ซึ่งจะทำให้การเคลื่อนที่ของ Pointer ของ Mouse มีการเคลื่อนที่ที่ไม่สิ้นสุด คล้ายกับว่าจอภาพมีขนาดไม่จำกัด

```
if(reg.CX = 0) then Move_mou_prt(reg,getmaxx-1,yold);
if(reg.DX = 0) then Move_mou_prt(reg,xold,getmaxy-1);
if(reg.CX = getmaxx) then Move_mou_prt(reg,1,reg.DX);
if(reg.DX = getmaxy) then Move_mou_prt(reg,reg.CX,1);
```

- โปรแกรมย่อยการตรวจสอบจอภาพ (Set G)

- เริ่มต้นกำหนด Graph Driver ให้เป็น EGA และเลือก Graph Mode ให้เป็น EGA HI เพื่อการแสดงผลจะสามารถใช้ได้ 2 page ทำการกำหนดสภาวะเริ่มต้นของจอภาพ เพื่อให้พร้อมที่จะกระทำการทาง Graph Mode ด้วยคำสั่ง Init Graph โดยผ่านค่าตัวแปร Graph Driver และ Graph Mode ถ้ามีความผิดปกติทาง Graphic ไม่สามารถกำหนดสภาวะเริ่มต้นได้ก็จะออกจากโปรแกรม

```
Procedure SetG;
  begin
    GraphDriver := EGA;
    GraphMode := EGAHi;
    InitGraph(GraphDriver, GraphMode, '');
    if GraphResult <> grOk then Halt(1);
  end;
```

- โปรแกรมย่อยการกำหนดสภาวะเริ่มต้นทาง Graphic (Init Graph)

รูปแบบ InitGraph (VAR Graph Driver; Graph Mode: integer);

การทำงาน คำสั่ง Init Graph เริ่มต้นทำงานใน Mode Graphic โดยเปลี่ยนการทำงานของจอภาพ จากการทำงานใน Modeตัวอักษร ให้เป็นMode Graphic และกำหนดค่าต่าง ๆ ไว้ให้ด้วย เช่น Viewport, สีและลวดลายในการวาดภาพและการระบายพื้นผิว เป็นต้น ถ้าตัวแปร Graphic Driver=o (Detect) หมายความว่าให้โปรแกรมตรวจสอบฮาร์ดแวร์และเลือก Drive และ Mode การทำงานที่ดีที่สุดโดยอัตโนมัติ ถ้า Graph Driver มีค่าไม่เท่ากับ ศูนย์ จะถือว่าเป็นค่า Driver ที่ถูกเลือกใช้ ได้มีการกำหนดค่าคงที่ไว้สำหรับ Driver ให้เรียบร้อยแล้วในกราฟยูนิต (Unit Graph) ดังนี้

```
Const
Detect     = 0;          CGA        = 1;
MCGA       = 2;          EGA        = 3;
EGA64      = 4;          EGAMONO    = 5;
IBM8514    = 6;          HERCMONO   = 7;
ATT400     = 8;          VGA        = 9;
PC3270     =10;
```

ตัวแปร Graph Mode เป็นตัวแปร ที่กำหนดว่าจะให้จอภาพมีการทำงานใน Mode ใด จาก Driver ที่เลือกใช้ Driver บางประเภทจะมี Mode การทำงานทางด้าน Graphic ได้หลาย Mode

- โปรแกรมย่อยการกำหนดสภาวะเริ่มต้นของ Mouse

รูปแบบ InitMouse (VAR reg: registers);

การทำงาน การกำหนดสภาวะเริ่มต้นของ Mouse ด้วย FUNCTION 00H ค่ารีจีสเตอร์ reg AX=0 ซึ่งมีการอินเทอร์รัพของ DOS โดยการใช้อินเทอร์รัพที่ 33H หลังจากเรียกคำสั่ง Init Mouse แล้ว Pointer ของ Mouse จะอยู่ที่กำหนดกลางจอภาพที่ page ด้านหลังของจอภาพ ซึ่งเราไม่สามารถมองเห็นได้

```
procedure InitMouse(var reg : registers);
begin
        reg.AX:=0; {func 00H}
        intr($33,reg);
end;
```

- โปรแกรมย่อยการแสดง Pointer ของ Mouse

รูปแบบ Display_pointer (VAR reg: registers)

การทำงาน กำหนดค่ารีจีสเตอร์ reg AX ให้ทำงานใน FUNCTION 01H โดยใช้อินเตอร์รัพ 33 ของ DOS เพื่อแสดง Pointer ของ Mouse บนจอภาพ

```
procedure Display_pointer(var reg : registers);
begin
      reg.AX:=$01;
      intr($33,reg);
end;
```

- โปรแกรมย่อยการอ่านค่าตำแหน่ง Point ของ Mouse

รูปแบบ Get-ptr-posit-botton-status (VAR reg: register);

การทำงาน กำหนดค่ารีจีสเตอร์ reg.AX ให้ทำงานใน FUNCTION 03H โดยใช้อินเตอร์รัพ 33 ของ DOS เพื่ออ่านค่าตำแหน่ง Pointer Mouse ณ ตำแหน่งปัจจุบัน

```
procedure Get_ptr_posit_botton_status(var reg : registers);
begin
     reg.AX:=$03;
     intr($33,reg); { display coordinate to crt }
end;
```

- โปรแกรมการซ่อน Pointer Mouse

รูปแบบ hide-pointer (VAR reg: registers);

การทำงาน กำหนดค่ารีจีสเตอร์ reg.AX ให้ทำงานใน FUNCTION 02H โดยใช้อินเตอร์รัฟ 33 ของ DOS เพื่อทำการซ่อน Pointer Mouse

```
procedure hide_pointer(var reg : registers);
begin
      reg.AX:=$02;
      intr($33,reg);
end;
```

- โปรแกรมย่อยการกำหนดตำแหน่ง Pointer Mouse

รูปแบบ Move-mou-prt (VAR reg: registers: xx,yy: interger);

การทำงาน กำหนดค่ารีจีสเตอร์ reg.AX ให้ทำงานใน FUNCTION 04H ตามลำดับ โดยใช้อินเตอร์รัพ 33H ของ DOS

```
procedure Move_mou_prt(var reg : registers;  xx,yy : integer);
begin
     reg.AX:=$04;
     reg.CX:= xx;
     reg.DX:= yy;
     intr($33,reg);
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน TOP VIEW ลงบนระนาบแกน x

รูปแบบ

- Calx (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Cax (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Clx (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Calx1 (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Cax1 (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Clx1 (choose: integer; var ang: real; var lx,tx1: integer): integer;

การทำงาน คำนวณตำแหน่งของจุดที่เปลี่ยนแปลงในการสร้างภาพของแขนกลในระนาบแกน x (ด้าน TOP VIEW)

```
function calx(choose:integer; var ang : real;
              var lx,tx1: integer) : integer;
begin
ทำการหาตำแหน่งของข้อแขน upper arm ด้านขวาของแขนกลในระนาบแกน x
end;

Function cax(choose:integer; var ang : real;
             var lx,tx1: integer) : integer;
begin
ทำการหาตำแหน่งของข้อแขน fore arm ด้านขวาของแขนกลในระนาบแกน x
end;

Function clx(choose:integer; var ang : real;
             var lx,tx1: integer) : integer;
begin
ทำการหาตำแหน่งของข้อด้านในของตัว body ของแขนกลในระนาบแกน x
end;
```

```
Function calx1(choose:integer; var ang : real;
               var lx,tx1: integer) : integer;
begin
ทำการหาตำแหน่งของขอบแขน upper arm ด้านซ้ายของแขนกลในระนาบแกน x
end;

Function cax1(choose:integer; var ang : real;
              var lx,tx1: integer) : integer;
begin
ทำการหาตำแหน่งของขอบแขน fore arm ด้านซ้ายของแขนกลในระนาบแกน x
end;

Function clx1(choose:integer; var ang : real;
              var lx,tx1: integer) : integer;
begin
ทำการหาตำแหน่งของขอบด้านนอกของตัว body ของแขนกลในระนาบแกน x
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพการหมุน Gripper ลงบนระนาบแกน x (ด้าน TOP VIEW)

รูปแบบ

- ttx (choose: integer; var ang: real): real;
- tux (choose: integer; var ang: real): real;

การทำงาน คำนวณตำแหน่งของจุดที่เปลี่ยนแปลงในการสร้างภาพการหมุน Gripper ในระนาบแกน x

```
function ttx(choose : integer; var ang : real):real;
begin
ทำการหาตำแหน่งภาพด้านบนของ gripper ในระนาบแกน x
end;

function tux(choose : integer; var ang : real): real;
begin
ทำการหาตำแหน่งภาพด้านข้างของ gripper ในระนาบแกน x
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน TOP VIEW ลงบันระนาบแกน y

รูปแบบ

```
- Caly  (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Cay   (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Cly   (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Caly1 (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Cay1  (choose: integer; var ang: real; var lx,tx1: integer): integer;
- Clx1  (choose: integer; var ang: real; var lx,tx1: integer): integer;
```

การทำงาน คำนวณตำแหน่งของจุดที่เปลี่ยนแปลงในการสร้างภาพของแขนกลในระนาบแกน y (ด้าน TOP VIEW)

```
Function caly(choose:integer; var ang : real;
              var lx,tx1 : integer) : integer;
begin
ทำการหาตำแหน่งของขอบแขน upper arm ด้านขวาของแขนกลในระนาบแกน y
end;
```

```
Function cay(choose:integer; var ang : real;
             var lx,tx1 : integer) : integer;
begin
ทำการหาตำแหน่งของขอบแขน fore arm ด้านขวาของแขนกลในระนาบแกน y
end;

Function cly(choose:integer; var ang : real;
             var lx,tx1 : integer) : integer;
begin
ทำการหาตำแหน่งของขอบด้านในของตัว body ของแขนกลในระนาบแกน y
end;

Function caly1(choose:integer;var ang : real;
               var lx,tx1 : integer) : integer;
begin
ทำการหาตำแหน่งของขอบแขน upper arm ด้านซ้ายของแขนกลในระนาบแกน y
end;

Function cay1(choose:integer;var ang : real;
              var lx,tx1 : integer) : integer;
begin
ทำการหาตำแหน่งของขอบแขน fore arm ด้านซ้ายของแขนกลในระนาบแกน y
end;

Function cly1(choose:integer; var ang : real;
              var lx,tx1 : integer) : integer;
begin
ทำการหาตำแหน่งของขอบด้านนอกของตัว body ของแขนกลในระนาบแกน y
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพการหมุน Gripper ลงบนระนาบแกน y(ด้าน TOP VIEW)

รูปแบบ

- tty (choose: integer; var ang: real): real;
- tuy (choose: integer; var ang: real): real;

การทำงาน คำนวณตำแหน่งของจุดที่เปลี่ยนแปลงในการสร้างภาพการหมุน Gripper ในระนาบแกน y

```
function tty(choose : integer; var ang : real):real;
begin
ทำการหาตำแหน่งภาพด้านบนของ gripper ในระนาบแกน y
end;

function tuy(choose : integer; var  ang: real): real;
begin
ทำการหาตำแหน่งภาพด้านข้างของ gripper ในระนาบแกน y
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพการสร้างภาพ Gripper ลงบนระนาบแกน x (SIDE VIEW)

รูปแบบ

- hcx (choose: integer; var b: real): real;
- hdx (choose: integer; var b: real): real;
- hnx (choose: integer; var b: real): real;
- hmx (choose: integer; var b: real): real;

การทำงาน คำนวณตำแหน่งของจุดที่เปลี่ยนแปลงในการสร้างภาพ Gripper ในระนาบแกน x

```
function hcx(choose : integer; var b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนข้อมือด้านบนของ gripper ในระนาบแกน x
end;

function hdx(choose : integer; var b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนข้อมือด้านข้างของ gripper ในระนาบแกน x
end;

function hnx(choose : integer; var b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนปลายข้อต่อด้านบนของ gripper ในระนาบแกน x
end;

function hmx(choose : integer; var b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนปลายข้อต่อด้านข้างของ gripper ในระนาบแกน x
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพการสร้างภาพ Gripper ลงบนระนาบแกน z (SIDE VIEW)

รูปแบบ

- hcz (choose: integer; var b: real): real;
- hdz (choose: integer; var b: real): real;
- hnz (choose: integer; var b: real): real;
- hmz (choose: integer; var b: real): real;

การทำงาน คำนวณตำแหน่งของชุดที่เปลี่ยนแปลงในการสร้างภาพ Gripper ในระนาบแกน z

```
function hcz(choose : integer; var b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนข้อมือด้านบนของ gripper ในระนาบแกน z
end;

function hdz(choose : integer; var b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนข้อมือด้านข้างของ gripper ในระนาบแกน z
end;

function hnz(choose : integer; b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนปลายข้อต่อด้านบนของ gripper ในระนาบแกน z
end;

function hmx(choose : integer; var b : real): real;
begin
ทำการหาตำแหน่งภาพส่วนปลายข้อต่อด้านข้างของ gripper ในระนาบแกน z
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน SIDE VIEW ลงบนระนาบแกน x

รูปแบบ

- cax (choose: integer; var a,b: real): integer
- cax1 (choose: integer; var a,b: real): integer
- px1 (choose: integer; var a,b: real): integer

การทำงาน คำนวณตำแหน่งของจุดที่เปลี่ยนแปลงในในการสร้างภาพของแขนกลในระนาบแกน x

```
Function cax(choose : integer;var a,b : real) : integer;
begin
ทำการหาตำแหน่งของขอบล่างแขน upper arm ด้านข้างของแขนกลในระนาบแกน x
end;

Function cax1(choose : integer;var a,b : real) : integer;
begin
ทำการหาตำแหน่งของขอบบนแขน upper arm ด้านข้างของแขนกลในระนาบแกน x
end;

Function px1(choose : integer;var a,b : real) : integer;
begin
ทำการหาตำแหน่งของขอบแขน fore arm ด้านข้างของแขนกลในระนาบแกน x
end;
```

- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน SIDE VIEW ลงบนระนาบแกน z

รูปแบบ

- caz (choose: integer; var a,b: real): integer
- caz1 (choose: integer; var a,b: real): integer
- pz1 (choose: integer; var a,b: real): integer

การทำงาน คำนวณตำแหน่งของจุดที่เปลี่ยนแปลงในในการสร้างภาพของแขนกลในระนาบแกน z

```
Function caz(choose : integer;var a,b : real) : integer;
begin
ทำการหาตำแหน่งของขอบล่างแขน upper arm ด้านข้างของแขนกลในระนาบแกน z
end;

Function caz1(choose : integer;var a,b : real) : integer;
begin
ทำการหาตำแหน่งของขอบบนแขน upper arm ด้านข้างของแขนกลในระนาบแกน z
end;

Function pz1(choose : integer;var a,b : real) : integer;
begin
ทำการหาตำแหน่งของขอบแขน fore arm ด้านข้างของแขนกลในระนาบแกน z
end;
```

- โปรแกรมย่อยในการแสดงภาพการหมุน Gripper ทางด้าน TOP VIEW

รูปแบบ handt (var orx,hag,lx,tx1: integer;var ang: real);

การทำงาน คำนวณการเคลื่อนที่และการหมุนของ Gripper โดยการรับค่า Parameter ที่ผ่านมาจากการกดค่า KEYBOARD และการเคลื่อนที่ Mouse เพื่อทำการสร้างภาพต่อไป

```
procedure handt(var orX,hag,lx,tx1 : integer; var ang : real);
var ox : integer;
    oy : integer;
begin
    ทำการคำนวณหาค่า ox
    ทำการคำนวณหาค่า oy
    ถ้ามุม hag = 0 หรือ hag = 180
```

ให้ทำการสร้างภาพการเคลื่อนที่และการหมุนของ Gripper ที่มุม 0 หรือ 180 องศา

ถ้ามุม hag = 90

ให้ทำการสร้างภาพการเคลื่อนที่และการหมุนของ Gripper ที่มุม 90 องศา

ทำการสร้างสัญญาณเสียงและทำการหน่วงเวลา

End;

- โปรแกรมย่อยในการแสดงภาพการหมุน Gripper ทางด้าน SIDE VIEW

รูปแบบ handd (var orx,hag: integer; var a,b: real);

การทำงาน คำนวณการเคลื่อนที่และการหมุนของ Gripper โดยการรับค่า Parameter ที่ผ่านมาจากการกดค่า KEYBOARD และการเคลื่อนที่ Mouse เพื่อทำการสร้างภาพต่อไป

```
procedure handd(var orX,hag: integer; var a,b : real);
var hx,hz : integer;
begin
    ทำการคำนวณหาค่า hx
    ทำการคำนวณหาค่า hz
    ถ้ามุม hag = 0 หรือ hag = 180
    ให้ทำการสร้างภาพการเคลื่อนที่และการหมุนของ Gripper ที่มุม 0 หรือ 180 องศา
    ถ้ามุม hag = 90
    ให้ทำการสร้างภาพการเคลื่อนที่และการหมุนของ Gripper ที่มุม 90 องศา
END;
```

- โปรแกรมย่อยแสดงผลการเก็บข้อมูลการเคลื่อนที่ของแขนกลทั้งหมด

รูปแบบ displayrec (var orx,p: integer; var a,b,ang: real);

การทำงาน เป็นการแสดงค่าของข้อมูลบางตัวที่เก็บไว้ในแต่ละ RECORD ซึ่งจะแสดงตั้งแต่

RECORD เริ่มต้นจนถึง RECORD ลำดับที่ p ถ้าไม่เกิน 10 RECORD แรกจะถูกแสดงผลเรียงตามลำดับโดยชิดขอบทางซ้ายของจอภาพ และถ้าเกิน 10 RECORD จะแสดงผลเรียงตามลำดับ ถัดมาอีกแถวหนึ่ง ซึ่งในที่นี้กำหนด RECORD ที่สามารถมีได้ไม่เกิน 20 RECORD

```
procedure displayrec(var orX,p : integer; var a,b,ang : real);
type po = record
     a,b,ang : real;
    end;
var q,w   : integer;
    shift : integer;
    col   : integer;
    pco   : integer;
    s     : string;
    savep : integer;
    popp  : integer;
    re_p  : integer;
    RPS   : ARRAY[1..20] of po;
begin
      ทำการกำหนดแถวในการแสดงผลเป็น 2 แถว
      โดยแถวแรกด้านซ้ายสุด ให้เป็นการแสดงผลใน 10 เร็กคอร์ดแรก
      และแถวที่ 2 ถัดมา ให้เป็นการแสดงผลในเร็กคอร์ดที่ 11 ถึง 20
      โดยทำการแสดงผลค่า A,B,R ในกรอบสี่เหลี่ยม บนจอภาพด้านซ้าย
      ไปตามลำดับ
END;
```

- <u>โปรแกรมย่อยแสดงการสร้างภาพของแขนกลในด้าน TOP VIEW</u>

<u>รูปแบบ</u> body1 (var reg: registers; var xold,yold,lx,tx1,orx,hag: integer; var ang: real; var s: string);

การทำงาน ทำการแปลงค่ามุม ang ให้เป็น Radian เพื่อใช้ในการคำนวณ การคำนวณการสร้างภาพของแขนกลจะต้องมีการส่งค่าตัวแปรต่าง ๆ ตามรูปแบบที่กำหนด โดยมีการเรียกฟังก์ชั่นและโปรแกรมย่อยต่าง ๆ ดังนี้

- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน TOP VIEW ลงบนระนาบแกน x
- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน TOP VIEW ลงบนระนาบแกน y
- ฟังก์ชั่นการคำนวณการฉายภาพการหมุนของ Gripper ลงบนระนาบแกน x ด้าน TOP VIEW
- ฟังก์ชั่นการคำนวณการฉายภาพการหมุนของ Gripper ลงบนระนาบแกน y ด้าน TOP VIEW
- โปรแกรมย่อยในการแสดงภาพการหมุน Gripper ทางด้าน TOP VIEW

```
Procedure body1(var reg : registers;
                var xold,yold,lx,tx1,orX,hag : integer;
                var ang : real;
                var s   : string);
begin
      ทำการแปลงค่ามุม ang ให้เป็นมุม radian
      ทำการสร้างกรอบสี่เหลี่ยมที่เป็นขอบของจอภาพ
      ทำการสร้าง base และ slide way ของแขนกลทางด้าน top view
      ทำการสร้างภาพ gripper ของแขนกลและทำการหมุน gripper
      โดยการเรียกโปรแกรมย่อย handt(orX,hag,lx,tx1,ang);
      ทำการสร้างภาพ body ของแขนกล
      ทำการสร้างภาพ upper arm ของแขนกล
      ทำการสร้างภาพ fore arm ของแขนกล
      ถ้า orX = 320 จึงทำการเขียนคำว่า 'origin' บนจอภาพด้านขวา
      ทำการแสดงผลและค่าตำแหน่งของ reg.cx,reg.dx,xold,yold บนจอภาพด้านขวา
      (เพื่อแสดงค่าตำแหน่งของ mouse บนจอภาพในตำแหน่งปัจจุบันและก่อนหน้านั้น)
      ทำการแปลงค่ามุม ang ให้เป็นมุม degree
end.
```

- โปรแกรมย่อยแสดงการสร้างภาพของแขนกลในด้าน SIDE VIEW

รูปแบบ body 2 (var orx,lx,lz,tx,tx1,tz,hag: integer; var a,b,ang; real);

การทำงาน ทำการแปลงค่ามุม a,b ให้เป็น Radian เพื่อใช้ในการคำนวณ การคำนวณการสร้างภาพของแขนกลจะต้องมีการส่งค่าตัวแปรต่างๆ ตามรูปแบบที่กำหนด โดยมีการเรียกฟังก์ชั่นและโปรแกรมย่อยต่าง ๆ ดังนี้

- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน SIDE VIEW ลงบนระนาบแกน x
- ฟังก์ชั่นการคำนวณการฉายภาพของแขนกลด้าน SIDE VIEW ลงบนระนาบแกน z
- ฟังก์ชั่นการคำนวณการฉายภาพการหมุนของ Gripper ลงบนระนาบแกน x ทางด้าน SIDE VIEW
- ฟังก์ชั่นการคำนวณการฉายภาพการหมุนของ Gripper ลงบนระนาบแกน z ทางด้าน SIDE VIEW
- โปรแกรมย่อยในการแสดงภาพการหมุน Gripper ทางด้าน SIDE VIEW

```
procedure body2( var orX,lx,lz,tx,tx1,tz,hag : integer;
                 var a,b,ang : real;  var grip : string);
begin
      ทำการแปลงค่ามุม a ให้เป็นมุม radian
      ทำการแปลงค่ามุม b ให้เป็นมุม radian
      ทำการสร้างกรอบสี่เหลี่ยมที่เป็นขอบของจอภาพ
      ทำการสร้าง base,body และ slide way ของแขนกลทางด้าน side view
      ทำการสร้างภาพ gripper ของแขนกลและทำการหมุน gripper
      โดยการเรียกโปรแกรมย่อย handd(orx,hag,a,b);
      ทำการสร้างภาพ upper arm ของแขนกล
      ทำการสร้างภาพ fore arm ของแขนกล
      ทำการเขียนมุม 'A' ที่ตำแหน่ง joint ระหว่าง body กับ upper arm
      ทำการเขียนมุม 'B' ที่ตำแหน่ง joint ระหว่าง upper กับ fore arm
```

ทำการเขียน 'gripper' เมื่อ gripper ทำการ open หรือ close

ทำการแปลงค่ามุม a ให้เป็นมุม degree

ทำการแปลงค่ามุม b ให้เป็นมุม degree

end.

- โปรแกรมย่อยการสร้างรูปวงกลม

รูปแบบ Circle :(x,y: integer; readius: word);

การทำงาน คำสั่ง Circle วาดวงกลมที่มีจุดสุดที่ (x,y) มีรัศมีเท่ากับ Radius โดยสีของเส้นรอบวงสามารถกำหนดได้ด้วยคำสั่ง Setcolor การวาดวงกลมจะต้องใช้ค่า Aspect Ratio เพื่อให้รูปที่เกิดบนจอภาพนั้นกลม

- โปรแกรมย่อยการสร้างรูปสี่เหลี่ยม

รูปแบบ Rectangle (x1,y1,x2,y2: integer)

การทำงาน คำสั่ง Rectangle วาดกรอบสี่เหลี่ยมซึ่งกำหนดโดยจุด (x1,y1) และ (x2, y2) โดยสีและลวดลายของเส้นที่วาด สามารถกำหนดได้ด้วยคำสั่ง Setcolor และ Setline Style จุด (x1,y1) และ (x2,y2) เป็นจุดมุมซ้ายบนและล่างขวาของกรอบสี่เหลี่ยม

- โปรแกรมย่อยการสร้างเส้นตรง

รูปแบบ line (x1,y1,x2,y2: integer)

การทำงาน คำสั่ง line ลากเส้นตรงจากจุด (x1,y1) ไป (x2,y2) ความหนาลวดลายและสีของเส้นตรงที่ถูกวาดขึ้น สามารถกำหนดด้วยคำสั่ง Setline style และ Setcolor

- โปรแกรมย่อยการสร้างจุด

รูปแบบ Putpixel (x,y,: integer; Color: word);

การทำงาน คำสั่ง putpixel สร้างจุด ณ ตำแหน่ง (x,y) และเปลี่ยนสีตามที่ต้องการจากการกำหนดค่าตัวแปรที่ Color

- โปรแกรมย่อยการแสดงตัวอักษรในโหมดกราฟฟิก

รูปแบบ Outtext xy (x,y: integer; Text String: string);

การทำงาน คำสั่ง Outtext xy แสดงข้อความออกทางจอภาพ ณ ตำแหน่ง (x,y) โดยไม่มีผลต่อตำแหน่ง CP. (CORRENT POINTER) ลักษณะการวางข้อความสามารถกำหนดได้ด้วยคำสั่ง Set text Justify แบบและทิศทางตัวอักษรสามารถกำหนดได้ด้วยคำสั่ง Set Text Style ถ้าข้อความยาวเกินขอบของ Viewport ข้อความนั้นจะถูกคลิปออก

- โปรแกรมย่อยการกำหนดสี

รูปแบบ SetColor (Color: word);

การทำงาน คำสั่ง Set Color กำหนดสีสำหรับการวาดรูป โดยใช้สีที่กำหนดไว้ในตัวแปร Color

Color for setcolor:

BLACK	= 0	DARKGRAY	= 8
BLUE	= 1	LIGHT BLUE	= 9
GREEN	= 2	LIGHT GREEN	= 10
CYAN	= 3	LIGHTCYAN	= 11
RED	= 4	LIGHT RED	= 12

MAGENTA	= 5	LIGHT MAGENTA	= 13
BROWN	= 6	YELLOW	= 14
LIGHT GRAY	= 7	WHITE	= 15

- โปรแกรมย่อยการแสดงหน้าจอภาพให้เห็นได้

รูปแบบ SetVisualPage ()age Num: word);

การทำงาน คำสั่ง SetVisualPage แสดงภาพในเพจที่จำกนดจากค่า Page Num ให้เกิดขึ้นบนจอภาพ โดยภาพของเพจของก่อนหน้านั้นจะหายไปจากจอภาพการวาดภาพของคำสั่งต่างๆ จะเกิดขึ้นบนเพจที่กำหนดไว้โดยคำสั่ง SetActivePage

คำสั่ง SetVisualPage เป็นการเลือกที่จะมองภาพในเพจใด ภาพในเพจนั้นจะถูกแสดงออกบนจอภาพ ถ้า SetActivePage และ SetVisualPage เป็นคนละ Page กันเราจะไม่เห็นการวาดภาพบนจอภาพ การใช้คำสั่ง SetVisualPage เปลี่ยนเพจที่ต้องการเห็นและ SetActivePage เปลี่ยนเพจที่ต้องการวาด

- โปรแกรมย่อยที่ต้องการกระทำการเกี่ยวกับหน้าจอภาพ

รูปแบบ SetActivePage (Page Num: word)

การทำงาน คำสั่ง setActivepage กำหนดเพจที่ใช้สำหรับวาดรูป (เป็นเอาท์พุต) ของคำสั่งต่างๆ ในการทำงาน โหมดกราฟฟิกบางโหมด มีเพจ (Page) ที่ใช้สำหรับวาดรูปได้หลายเพจ และสามารถเลือกเพจต่าง ๆ โดยใช้โปรแกรมย่อยนี้

- โปรแกรมย่อยการล้างจอภาพ

รูปแบบ ClearDevice;

การทำงาน คำสั่ง ClearDevice ลบทุกสิ่งบนจอภาพออกทั้งจอภาพ โดยเปลี่ยนสีของจอภาพเป็นสี Blackground

- ฟังก์ชั่นการกำหนดค่าสูงสุดในระนาบแกน x

รูปแบบ GetMaxX: integer;

การทำงาน ค่าที่ส่งกลับโดยฟังก์ชั่น GetMaxX คือค่าโคออร์ดิเนตที่มากที่สุดทางแกน x ของ pixel บนจอภาพ ภายใต้ Driver และ Mode ที่กำลังทำอยู่

- ฟังก์ชั่นการกำหนดค่าสูงสุดในระนาบแกน y

รูปแบบ GetMaxY: integer;

การทำงาน ค่าที่ส่งกลับโดยฟังก์ชั่น GetMaxY คือค่าโคออร์ดิเนตที่มากที่สุดทางแกน Y ของ pixel บนจอภาพ ภายใต้ Driver และ Mode ที่กำลังทำอยู่

- โปรแกรมย่อยการแสดงการยกเลิกโหมดการทำงานทางด้านกราฟฟิก

รูปแบบ CloseGraph;

การทำงาน คำสั่ง CloseGraph เปลี่ยนการทำงานของจอภาพให้กลับไปอยู่ในโหมดเดิมก่อนเปลี่ยนการทำงานมาอยู่ในโหมดกราฟฟิก (โหมดตัวอักษร) และคืนหน่วยความจำทั้งหมดที่ถูกจองไว้ในโหมดกราฟฟิกให้กับระบบ

- โปรแกรมย่อยกำหนดสภาวะเริ่มต้นของพอร์ท

รูปแบบ ini_port (port,code: byte);

การทำงาน คำสั่ง ini_port จะทำการกำหนดพอร์ทว่าเป็นพอร์ท COM 1 หรือ COM 2 ด้วยรีจีสเตอร์ reg.dx ถ้าต้องการ initial port ที่ COM 1 ให้กำหนดค่ารีจีสเตอร์ reg.dx=0 ถ้าต้องการ initial port ที่ COM 2 ให้กำหนดค่ารีจีสเตอร์ reg.dx=1 กำหนดให้รีจีสเตอร์ reg.ah=0 เพื่อแสดงการกำหนดสภาวะเริ่มต้นของพอร์ทนั้น ส่วนการกำหนดรีจีสเตอร์ reg.al จะต้องกำหนดค่าให้สัมพันธ์กับ Device ที่จะติดต่อด้วย

หมายเลขประจำบิต	ความหมาย
7, 6, 5	อัตรารับส่งข้อมูล 111=9600 baud
4, 3	พาริตี้ 00 หรือ 10 = ไม่มีพาริตี้
2	จำนวนของบิตสิ้นสุด 1 = 2 บิตสิ้นสุด
1, 0	จำนวนบิตในข้อมูล 1 ไบต์ 11 = 8 บิต

เมื่อกำหนดค่าหมายเลขประจำบิตนี้ได้ทั้งหมด 8 บิตให้แปลงเป็นเลขฐาน 16 ซึ่งจะได้ค่า E7 H = 11100111_2 เมื่อกำหนดค่ารีจีสเตอร์ต่าง ๆ เหล่านี้แล้ว ให้ทำการอินเตอร์รัพหมายเลข 14 พร้อมทั้งส่งค่ารีจีสเตอร์เหล่านั้นออกไป

```
procedure ini_port(port,code:BYTE);
begin
     reg.dx := port ;
     reg.ah := 0 ; { init }
     reg.al := code ;
     intr($14,reg) ;
end;
```

- โปรแกรมย่อยเป็นการส่งข้อมูลออกพอร์ต

รูปแบบ send(port,data: byte);

การทำงาน คำสั่ง send จะทำการเลือกพอร์ทว่าเป็น COM 1 หรือ COM 2 ด้วยรีจีสเตอร์ reg.dx ถ้าต้องการส่งข้อมูลไปออก COM 1 ให้กำหนดรีจีสเตอร์ reg.dx=0 ถ้าต้องการส่งข้อมูลไปออก COM 2 ให้กำหนดรีจีสเตอร์ reg.dx=1 กำหนดให้รีจีสเตอร์ reg.ah=1 เพื่อแสดงถึงการส่งข้อมูลออกจากพอร์ตนั้น ส่วนรีจีสเตอร์ reg.al คือ ข้อมูลที่ต้องการจะส่งออกไป ให้ทำการอินเตอร์รัพหมายเลข 14 พร้อมส่งค่าข้อมูลเหล่านั้นออกไป

```
procedure send(port,data:byte);
begin
      reg.dx := port ;
      reg.ah := 1 ;   { sent }
      reg.al := data;  { data   }
      intr($14,reg) ;
end;
```

- โปรแกรมย่อยการส่งข้อมูลพร้อมทั้งการแสดงผลออกทางจอภาพ

รูปแบบ OutPort(p: po);

การทำงาน คำสั่ง OutPort คือ การส่งข้อมูลตามลำดับ พร้อมทั้งแสดงผลข้อมูลที่ส่งออกทางจอภาพ

```
procedure outport(p:po);
var s : string;
begin
     send(PT,p.choose); outtextxy(500,280,'choose motor: ') ;
     str(reg.al,s);     outtextxy(610,280,s);
     send(PT,p.m);
     send(PT,p.sign);   outtextxy(500,290,'sign   : ') ;
     str(reg.al,s);     outtextxy(600,290,s);
```

```
    send(PT,p.d1);      outtextxy(500,300,'data d1: ') ;
    str(reg.al,s);      outtextxy(600,300,s);
    send(PT,p.d2);      outtextxy(500,310,'data d2: ') ;
    str(reg.al,s);      outtextxy(600,310,s);
    send(PT,p.d3);      outtextxy(500,320,'data d3: ') ;
    str(reg.al,s);      outtextxy(600,320,s);
    send(PT,p.d4);      outtextxy(500,330,'data d4: ') ;
    str(reg.al,s);      outtextxy(600,330,s);
    send(PT,p.enter);
end;
```

- โปรแกรมย่อยการส่งข้อมูลแสดงสถานะของ Gripper

รูปแบบ OutportGrip (p: po);

การทำงาน คำสั่ง OutportGrip จะทำการส่งข้อมูลเพื่อแสดงว่า Gripper อยู่ในสถานะเปิดหรือปิดอยู่นั่นเอง

```
procedure outportGrip(p:po);
var s : string;
begin
    send(PT,p.choose);
    send(PT,p.m);
    send(PT,p.sign);
    send(PT,p.d1);
    send(PT,p.d2);
    send(PT,p.d3);
    send(PT,p.d4);
    send(PT,p.enter);
end;
```

- โปรแกรมย่อยกำหนดสถานะของแขนกลในตำแหน่ง Home

รูปแบบ gohome;

การทำงาน คำสั่ง gohome คือการกำหนดสภาวะเริ่มต้นของแขนกล

```
Procedure gohome;
var go_home : po;
begin
      ini_port(PT,$E7);
      send(PT,50);
      send(PT,77);
      send(PT,43);
      send(PT,48);
      send(PT,54);
      send(PT,53);
      send(PT,48);
      send(PT,13);
      delay(3000);
      send(PT,51);
      send(PT,77);
      send(PT,45);
      send(PT,48);
      send(PT,54);
      send(PT,48);
      send(PT,48);
      send(PT,13);
end;
```

- ฟังก์ชั่นตารางแปลงค่ารหัส ASCII เป็นค่าตำแหน่งฐานสิบ

รูปแบบ Table (dat: integer): integer;

การทำงาน คำสั่ง Tabel คือการแปลงรหัส ASCII ให้เป็นตำแหน่งเลขฐานสิบ

```
function  table(dat : integer): integer;
begin
case dat of
     0 : table := 48;
     1 : table := 49;
     2 : table := 50;
     3 : table := 51;
     4 : table := 52;
     5 : table := 53;
     6 : table := 54;
     7 : table := 55;
     8 : table := 56;
     9 : table := 57;
     end
end;
```

- โปรแกรมย่อยแสดงการขับมอเตอร์ในส่วน BODY

รูปแบบ Move_it 49 (data: integer; olddata: integer);

การทำงาน คำสั่ง move_it 49 คือทำการกำหนดค่าเริ่มต้นที่ต้องจะส่งคำสั่งไม่ขับมอเตอร์ ทำการคำนวณการเคลื่อนที่ที่เปลี่ยนแปลงของมอเตอร์ในส่วน BODY นำค่าที่เปลี่ยนแปลงไปเทียบกับตารางแปลงค่ารหัส ASCII เป็นค่าตำแหน่งเลขฐานสิบแล้วส่งข้อมูลเหล่านั้นออกทางพอร์ต

พร้อมทั้ง delay การทำงานของคอมพิวเตอร์เพื่อรอการทำงานของแขนกลให้เสร็จสิ้นเสียก่อน

```
procedure move_it49(data : integer; olddata : integer);
var orbit : integer;
    num   : integer;
    mov_e : integer;
begin
        ini_port(PT,$E7);     { com_ ,contr code }
        mov_e := data-olddata;
        orbit := round(Abs(mov_e*11.5));
        p.choose := 49;
        p.m := 77;
        if(data>olddata)then p.sign := 43;
        if(data<olddata)then p.sign := 45;
        p.d1 := trunc(orbit div 1000);
        p.d1 := table(p.d1);
        num  := trunc(orbit div 100);
        p.d2 := num mod 10;
        p.d2 := table(p.d2);
        num  := trunc(orbit div 10);
        p.d3 := num mod 10;
        p.d3 := table(p.d3);
        p.d4 := orbit mod 10;
        p.d4 := table(p.d4);
        p.enter := 13;
        outport(p);
        delay(orbit*4);
        outtextxy(5,350,'**END BYTE ') ;
end;
```

- โปรแกรมย่อยแสดงการขับมอเตอร์ในส่วน Slideway

รูปแบบ move_it 54 (data: integer; Olddata: integer);

การทำงาน คำสั่ง move_it 54 คือทำการกำหนดค่าเริ่มต้นที่ต้องจะส่งคำสั่งไม่ขับมอเตอร์ ทำการคำนวณการเคลื่อนที่ที่เปลี่ยนแปลงของมอเตอร์ในส่วน SLIDEWAY นำค่าที่เปลี่ยนแปลงไปเทียบกับตารางแปลงค่ารหัส ASCII เป็นค่าตำแหน่งเลขฐานสิบแล้วส่งข้อมูลเหล่านั้นออกทางพอร์ต พร้อมทั้ง delay การทำงานของคอมพิวเตอร์เพื่อรอการทำงานของแขนกลให้เสร็จสิ้นเสียก่อน โดยการส่งข้อมูลมีลักษณะเริ่มต้นเป็นการเลือกว่ามอเตอร์ตัวใดที่ต้องการขับให้เคลื่อนที่ คำสั่งต่อไปเป็นการส่งรหัสตัวอักษร M ซึ่งเท่ากับ 77 (เป็นตำแหน่งในรหัส ASCII) คำสั่งต่อไปอีก 1 บิต เป็นการแสดงทิศทางการหมุนของมอเตอร์ว่ามีทิศทางตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา คำสั่งต่อไปอีก 4 บิต แสดงถึงระยะที่ต้องการให้มอเตอร์เคลื่อนที่ และคำสั่งบิตสุดท้ายคือ ENTER=13 (เป็นตำแหน่งในรหัส ASCII) เป็นการสิ้นสุดคำสั่ง

```
procedure move_it54(lata : integer; olddata : integer);
var orbit : integer;
    num   : integer;
    mov_e : integer;
    s : string;
begin
     ini_port(PT,$E7);     { com_ ,contr code }
        Str(data,s);
        outtextxy(100,10,s);
        writeln('**START SEND**') ;
        mov_e := data-olddata;
        orbit := round(Abs(mov_e*15.625));
        p.choose := 54;
        p.m := 77;
        if(data<olddata)then p.sign := 45;
```

```
        if(data>olddata)then p.sign := 43;
        p.d1 := trunc(orbit div 1000);
        p.d1 := table(p.d1);
        num  := trunc(orbit div 100);
        p.d2 := num mod 10;
        p.d2 := table(p.d2);
        num  := trunc(orbit div 10);
        p.d3 := num mod 10;
        p.d3 := table(p.d3);
        p.d4 := orbit mod 10;
        p.d4 := table(p.d4);
        p.enter := 13;
        outport(p);
        delay(orbit*5);
        outtextxy(5,350,'**END BYTE ') ;
end;
```

- <u>โปรแกรมย่อยแสดงการขับมอเตอร์ที่อยู่ระหว่าง BODY กับ UPPER ARM</u>

<u>รูปแบบ</u> move_it 50 (data: integer; olddata: integer);

<u>การทำงาน</u> คล้ายกับ move_it 49 แต่เป็นการขับมอเตอร์ตัวที่อยู่ระหว่าง BODY กับ UPPER ARM

```
procedure move_it50(data : integer; olddata : integer);
var orbit : integer;
    num   : integer;
    mov_e : integer;
```

```
begin
     ini_port(PT,$E7);     { com_ ,contr code }
        mov_e := data-olddata;
        orbit := round(Abs(mov_e*8.46));
        p.choose := 50;
        p.m := 77;
        if(data<olddata)then p.sign := 43;
        if(data>olddata)then p.sign := 45;
        p.d1 := trunc(orbit div 1000);
        p.d1 := table(p.d1);
        num  := trunc(orbit div 100);
        p.d2 := num mod 10;
        p.d2 := table(p.d2);
        num  := trunc(orbit div 10);
        p.d3 := num mod 10;
        p.d3 := table(p.d3);
        p.d4 := orbit mod 10;
        p.d4 := table(p.d4);
        p.enter := 13;
        outport(p);
        delay(orbit*5);
        outtextxy(5,350,'**END BYTE ') ;
end;
```

- <u>โปรแกรมย่อยแสดงการขับมอเตอร์ที่อยู่ระหว่าง UPPER ARM กับ FORE ARM</u>

<u>รูปแบบ</u> move_it 51 (data: integer; olddata: integer);

<u>การทำงาน</u> คล้าย move_it 49 แต่เป็นการขับมอเตอร์ตัวที่อยู่ระหว่าง UPPERARM กับ

```
{FORE ARM

procedure move_it51(data : integer; olddata : integer);
 var orbit : integer;
     num   : integer;
     mov_e : integer;
 begin
      ini_port(PT,$E7);     { com_ ,contr code }
         mov_e := data-olddata;
         orbit := round(Abs(mov_e*8.46));
         p.choose := 51;
         p.m := 77;
         if(data>olddata)then p.sign := 45;
         if(data<olddata)then p.sign := 43;
         p.d1 := trunc(orbit div 1000);
         p.d1 := table(p.d1);
         num  := trunc(orbit div 100);
         p.d2 := num mod 10;
         p.d2 := table(p.d2);
         num  := trunc(orbit div 10);
         p.d3 := num mod 10;
         p.d3 := table(p.d3);
         p.d4 := orbit mod 10;
         p.d4 := table(p.d4);
         p.enter := 13;
         outport(p);
         delay(orbit*5);
         outtextxy(5,350,'**END BYTE ') ;
end;
```

- โปรแกรมย่อยแสดงการเก็บค่าสถานะของ Gripper

รูปแบบ gripper (grip: string);

การทำงาน คำสั่ง gripper ทำการรับค่า grip ว่ามีสถานะเปิดหรือปิดอยู่ พร้อมทั้งทำการตามคำสั่งนั้นออกทางพอร์ต

```
Procedure  gripper(grip : string);
begin
ini_port(PT,$E7);     { com_ ,contr code }
     p.choose :=56;
     p.m      :=77;
   if(grip = 'close') then p.sign := 43;
   if(grip = 'open')  then p.sign := 45;
     p.d1     :=49;   (**1**)
     p.d2     :=48;   (**0**)
     p.d3     :=48;   (**0**)
     p.d4     :=48;   (**0**)
     p.enter  :=13;
outportGrip(p);
end;
```

- การติดต่อ Computer กับ Controller โดยผ่านทาง RS-232C

ตัว Controller Robot จะประกอบด้วยไมโครโปรเซสเซอร์ที่ต้องทำงานภายใต้การควบคุมจากคอมพิวเตอร์ภายนอก ซึ่งทั้ง 2 หน่วยจะต้องติดต่อกันโดยทาง port RS-232C การที่ SCORBOT-ER III จะสามารถเคลื่อนที่ได้จะต้องทำการกำหนดค่า port ที่จะส่งออกทาง port RS-232C ดังนี้

- Baud rate: 9600
- Data bits: 8
- Start bit: 1
- Stop bits: 2
- No parity

เมื่อเราใช้ Computer ติดต่อกับ SCORBOT-ER III ทาง port RS-232C ซึ่ง PIN ที่สำคัญของ RS-232C ที่สำคัญมี 3 PIN ได้แก่

- PIN 2: TRANSMISSION (การส่งข้อมูล)
- PIN 3: RECEPTION (การรับข้อมูล)
- PIN 7: SINAL GROUND (สายสัญญาณกราวด์)

โดยกระทำการต่อเชื่อม Computer กับ SCORBOT-ER III ทาง RS-232C ดังนี้

IBM PC COMPUTER (D 25 female Connector)	SCORBOT-ER III CONTROLLER (D 25 male Connector)
PIN 2	TO PIN 3
PIN 3	TO PIN 2
PIN 7	TO PIN 7
PIN 4 TO PIN 5	
PIN 6 TO PIN 8 AND 20	

ลักษณะการแสดงบนหน้าจอ

การกำหนดค่าตัวแปรต่างๆ บนจอภาพมีความหมายดังต่อไปนี้

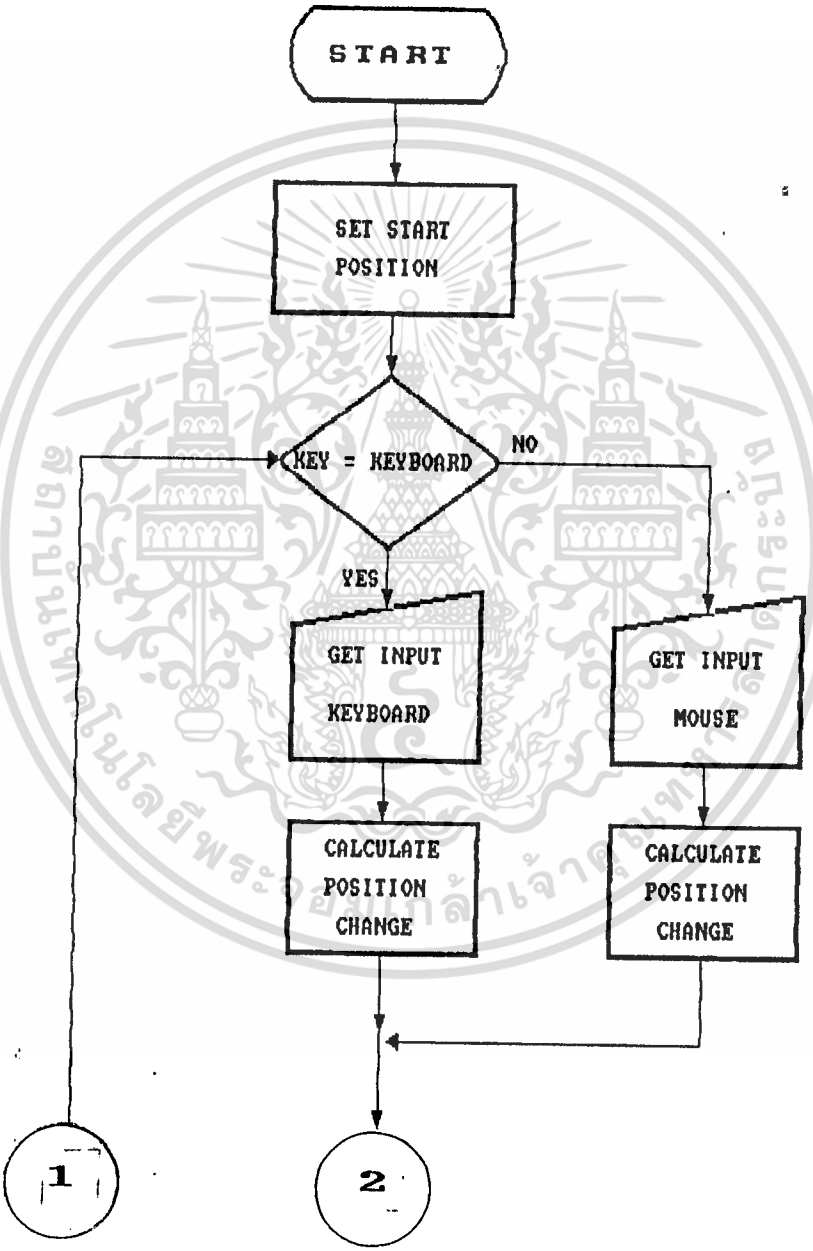
A หมายถึง ค่ามุมที่แขนกลในส่วน upperarm ทำมุมกับระนาบแนวนอนกับแขนกล

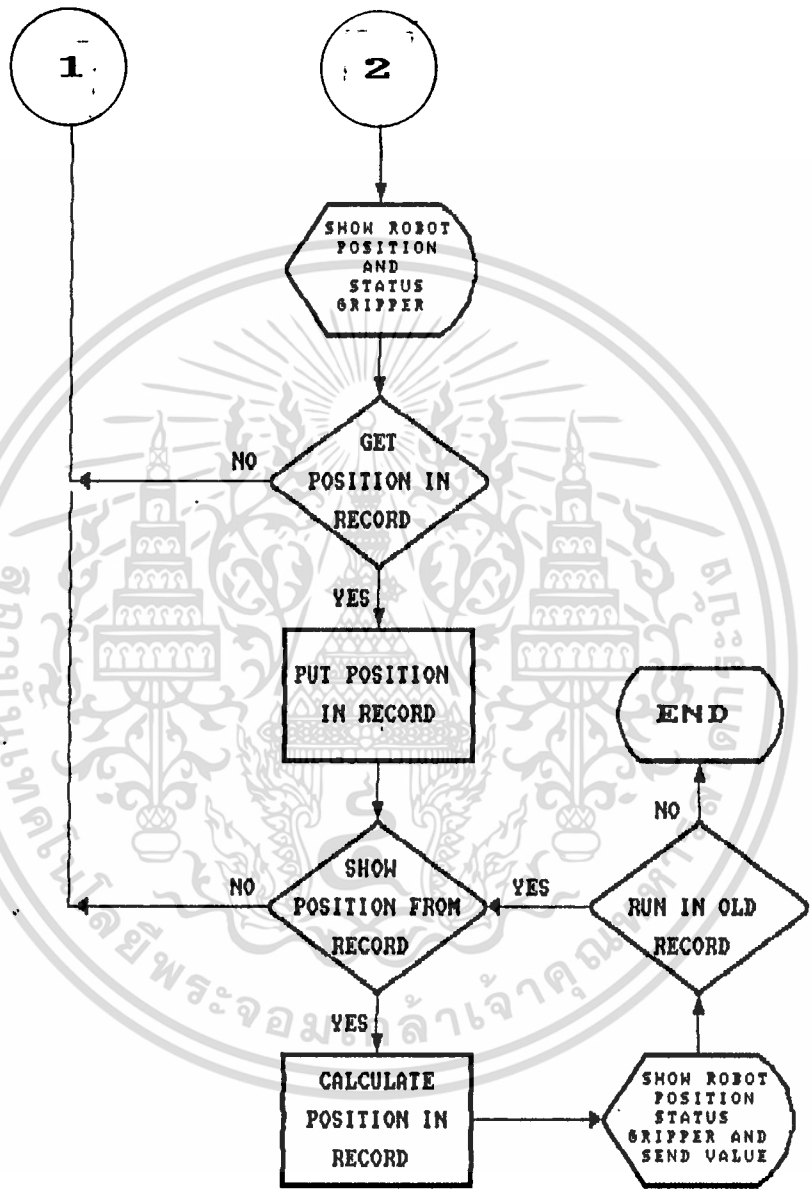
B หมายถึง ค่ามุมที่แขนกลในส่วน upperarm ทำมุมกับ forearm ของแขนกล

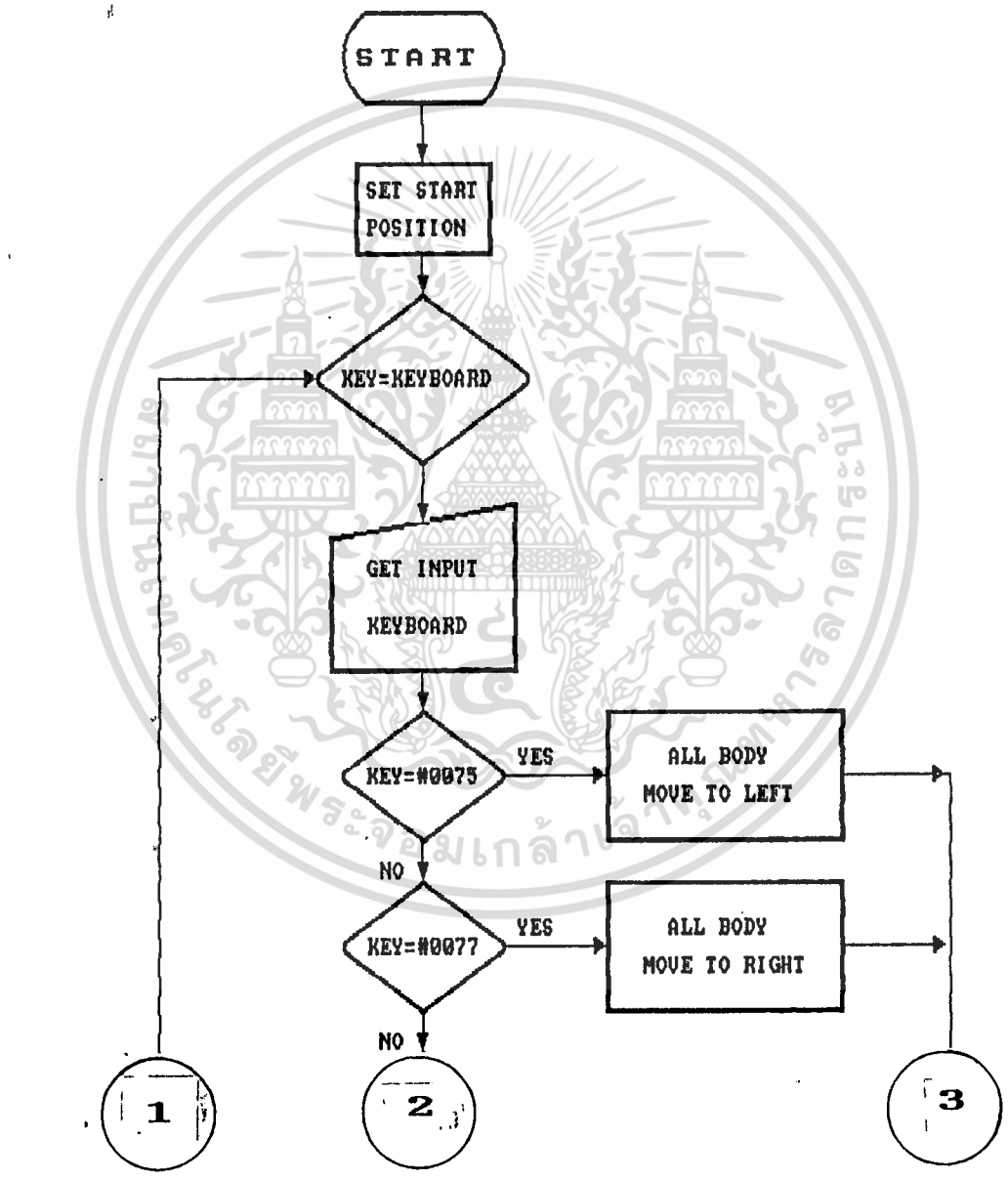
R หมายถึง ค่ามุมที่แขนกลทั้งแขน ทำมุมกับ slideway ในด้าน TOP VIEW

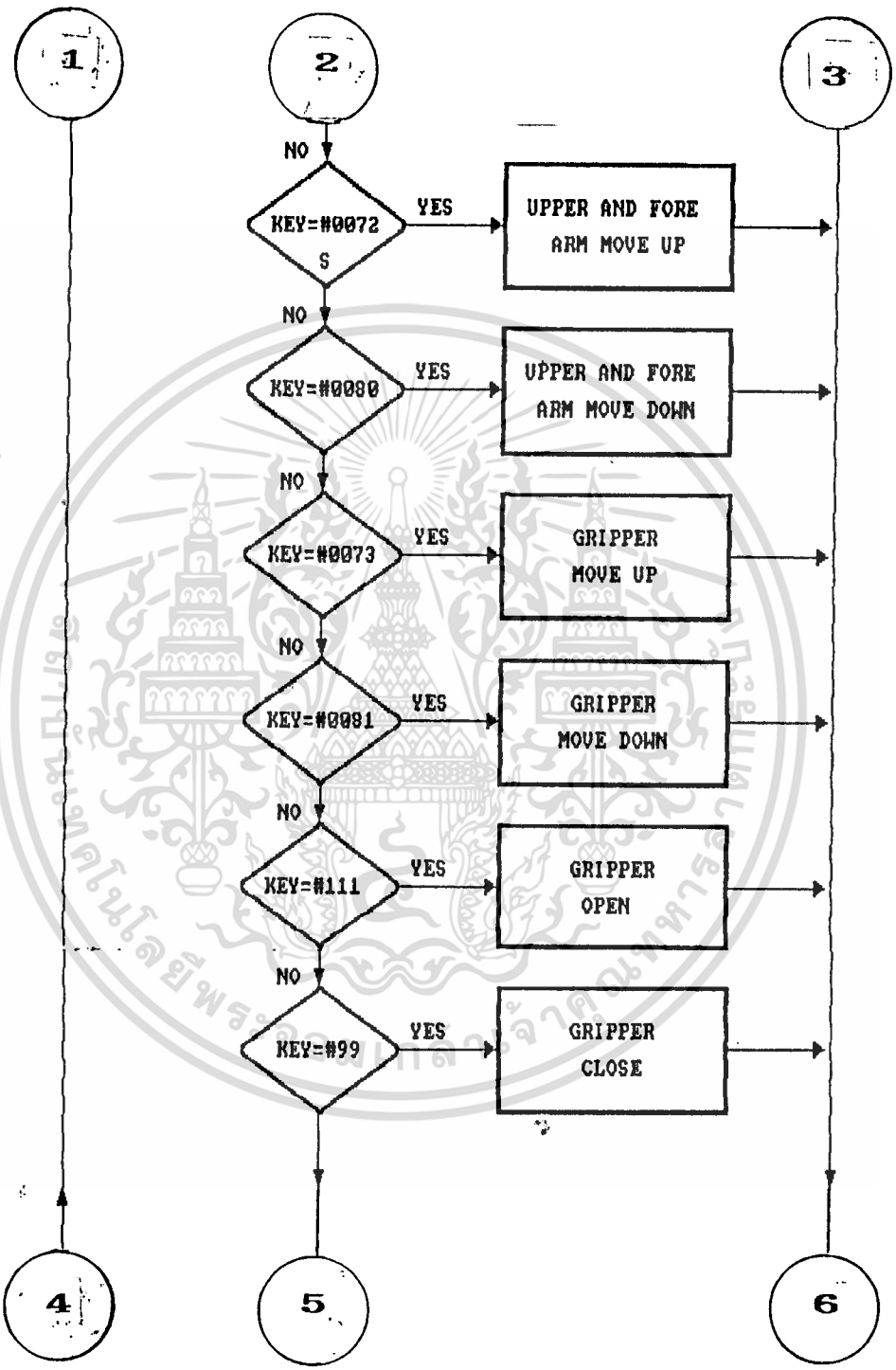
gripper open หมายถึง ปลายแขนที่ใช้จับยึดวัตถุทำการเปิดออก

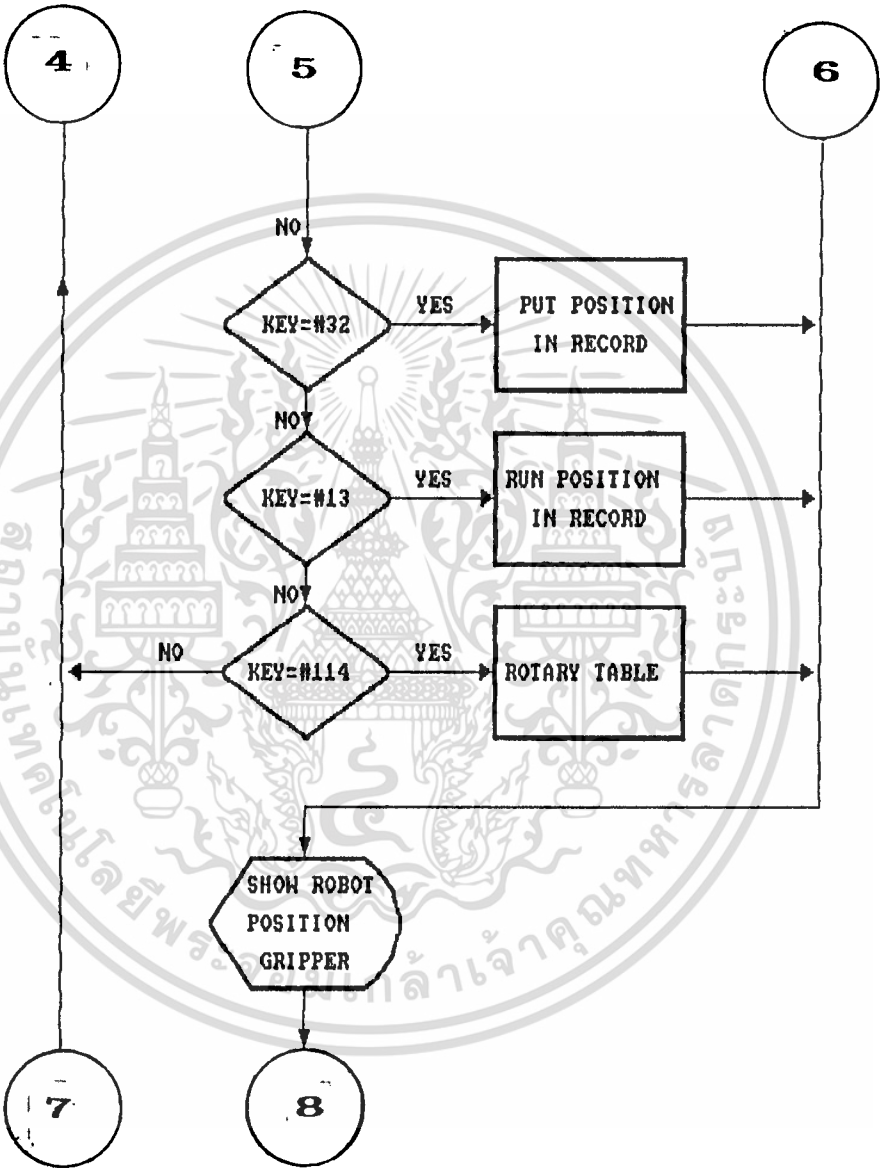
gripper close หมายถึง ปลายแขนที่ใช้จับยึดวัตถุทำการปิดเข้า

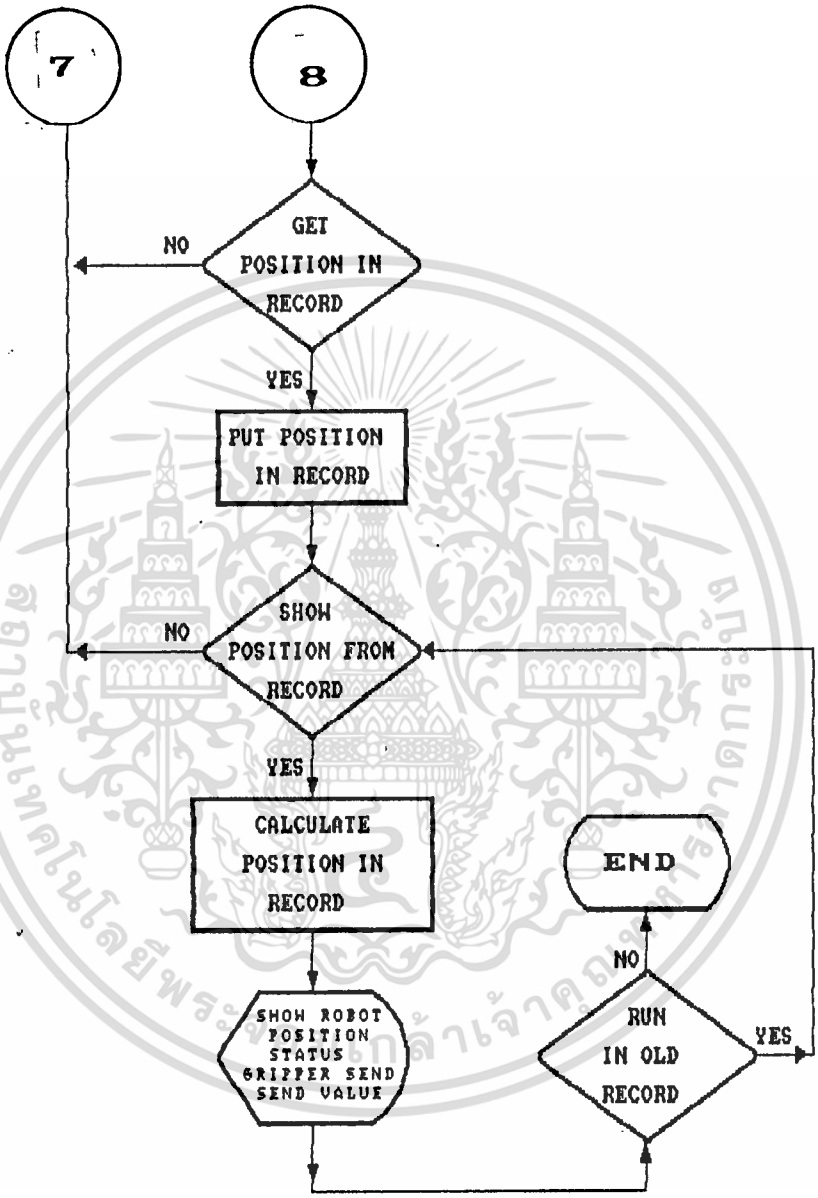


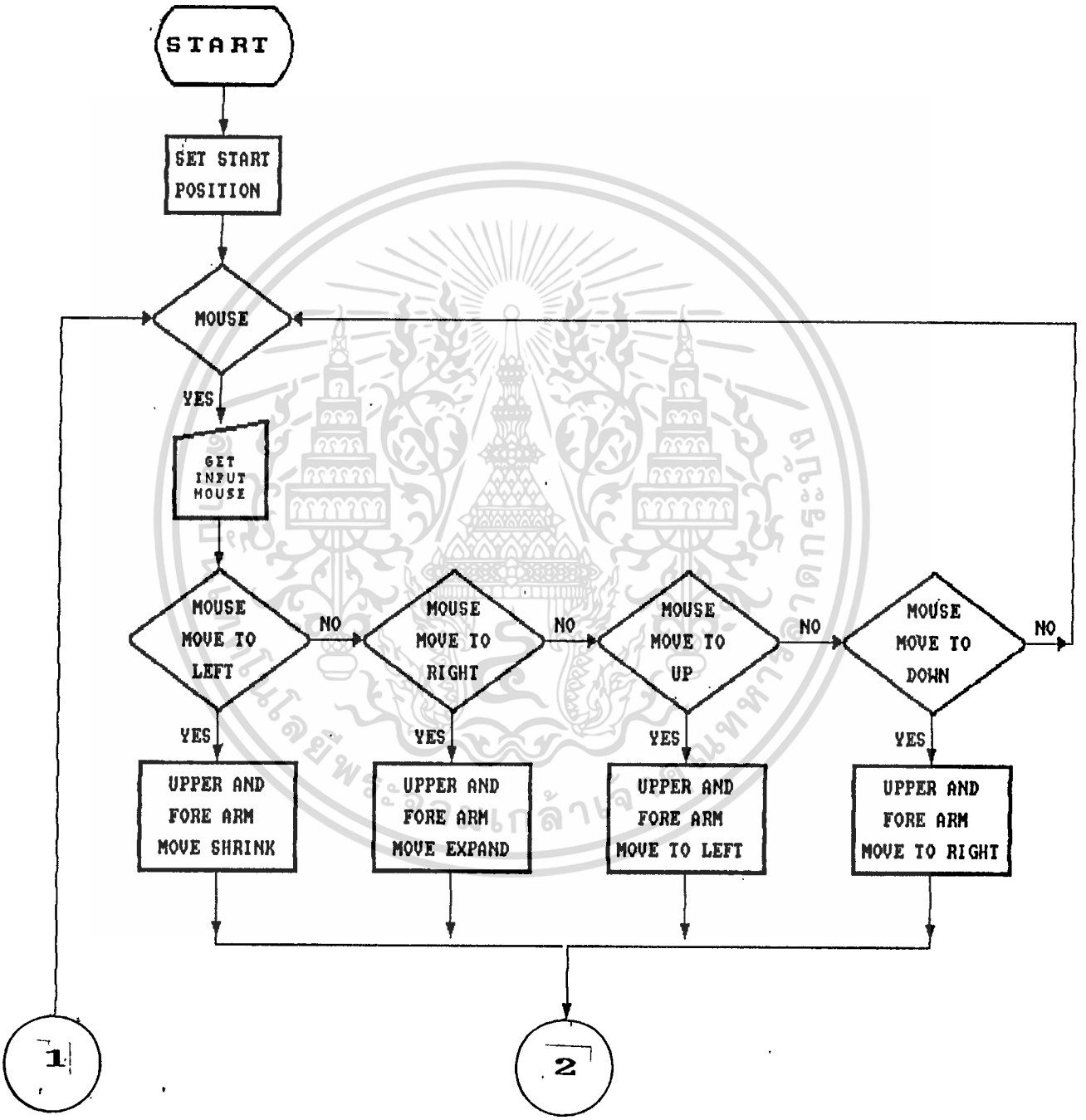


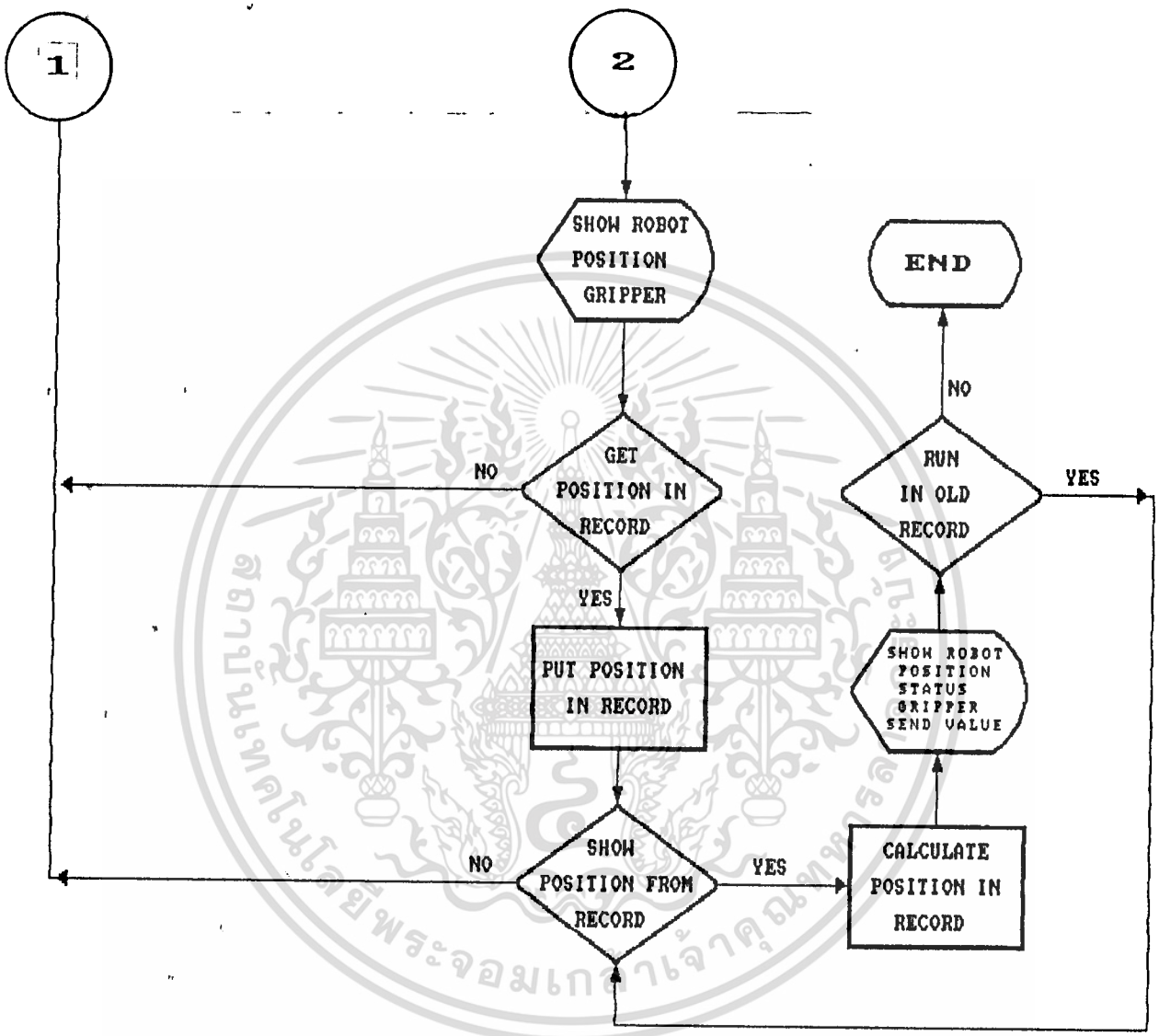












บทที่ 5

การทดลองการใช้งานโปรแกรม

จากโครงงานระบบช่วยสอนหุ่นยนต์สามารถเก็บเรคคอร์ดได้ถึง 100 เรคคอร์ด แต่จากการทดลองได้กำหนดให้ในโปรแกรมเก็บเรคคอร์ดได้ 20 เรคคอร์ด เป็นตัวอย่างในฉารทดลอง ซึ่งแสดงให้เห็นว่า สามารถเคลื่อนที่แขนกลจำลองบนจอภาพของคอมพิวเตอร์ไปในทิศทางที่ต้องการได้จากการเปลี่ยนแปลงค่า KEYBOARD และ MOUSE เมื่อได้ทิศทางและตำแหน่งที่ต้องการแล้วก็ทำการเก็บค่าเรคคอร์ดในตำแหน่งและทิศทางที่ต้องการไว้ เมื่อต้องการให้แขนกลจริงทำงานตามค่าที่เก็บค่าเรคคอร์ดไว้ในตอนต้นก็สั่งโปรแกรมให้ส่งค่าเรคคอร์ดผ่านทางพอร์ต RS-232C ไปยังตัว CONTROLLER ของ ROBOT-ARM ซึ่งในที่นี้ได้ทดลองใช้ ROBOT-ARM รุ่น SCORBOT ER-III โดยมี CONTROLLER 8031 เป็นตัว CONTROLLER ของ ROBOT-ARM

ROBOT-ARM รุ่น SCORBOT ER-III นี้ต้องใช้คำสั่ง UNI-DIRECTION ในการควบคุมการทำงานของมอเตอร์ทั้ง 8 ตัว โดยคำสั่งนี้สามารถสั่งมอเตอร์ให้ทำงานได้ทีละตัว ซึ่งในการทดลองนี้ได้ทำการทดลองให้แขนกลทำการเคลื่อนที่ใน 1 ระนาบ คือในระนาบแนวนอน โดยกำหนดให้ Slide way แทนทิศทางในการเคลื่อนที่ในระนาบแกน X และทิศทางการเคลื่อนที่ของแขนไปตามระนาบแกน XY จากการทดลองได้ทดสอบให้ระบบช่วยสอนหุ่นยนต์เคลื่อนที่ไปตามทิศทางที่ตั้งไว้ 5 ขั้นตอนโดยแต่ละขั้นตอนมีการเคลื่อนที่ดังต่อไปนี้

ขั้นตอนที่ 1 ทำการสอนหุ่นยนต์ให้เคลื่อนที่ไปทางด้านซ้าย

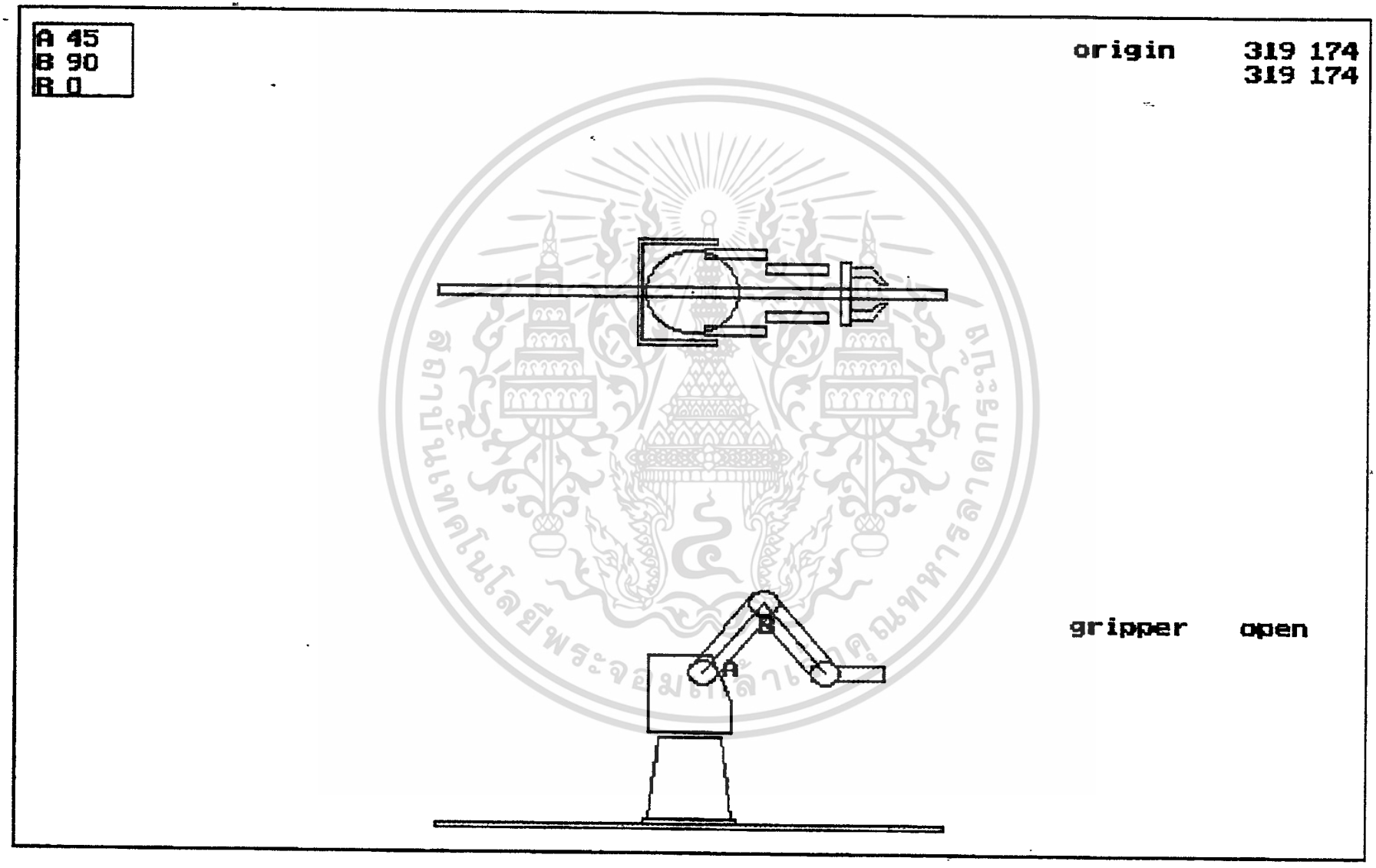
ขั้นตอนที่ 2 ทำการสอนหุ่นยนต์ให้หมุนแขนไปในทิศทางทวนเข็มนาฬิกา

ขั้นตอนที่ 3 ทำการสอนหุ่นยนต์ให้ยืดแขนในส่วน UPPER & FORE ARM ไปข้างหน้า และทำการหยิบของ(GRIPPER CLOSE)

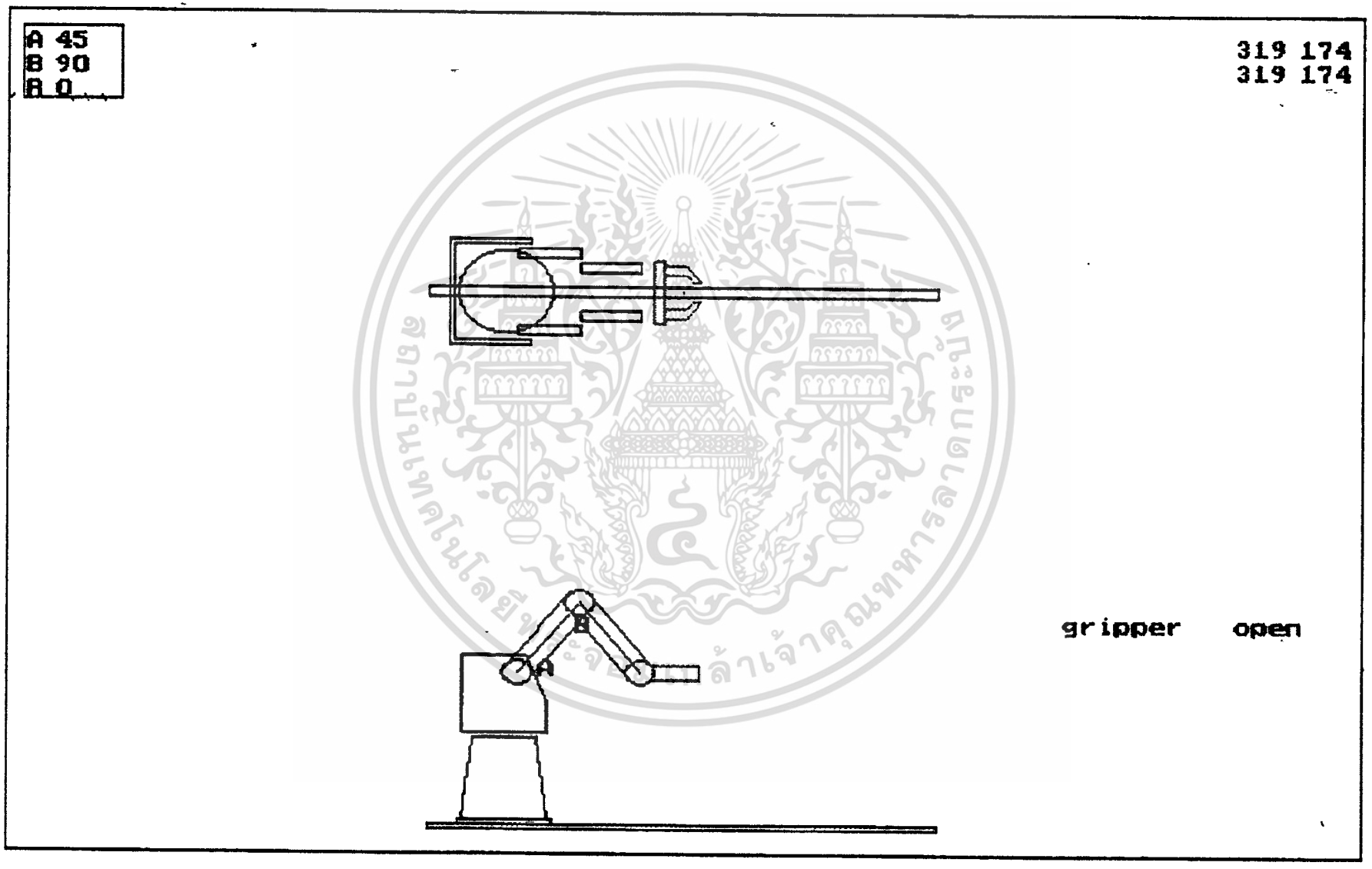
ขั้นตอนที่ 4 ทำการสอนหุ่นยนต์ให้หมุนแขนกลับในทิศทางตามเข็มนาฬิกา

ขั้นตอนที่ 5 ทำการสอนหุ่นยนต์ให้เคลื่อนที่ไปทางด้านขวาจนสุด Slide way หลังจากนั้นให้ทำการวางของลง(GRIPPER OPEN)

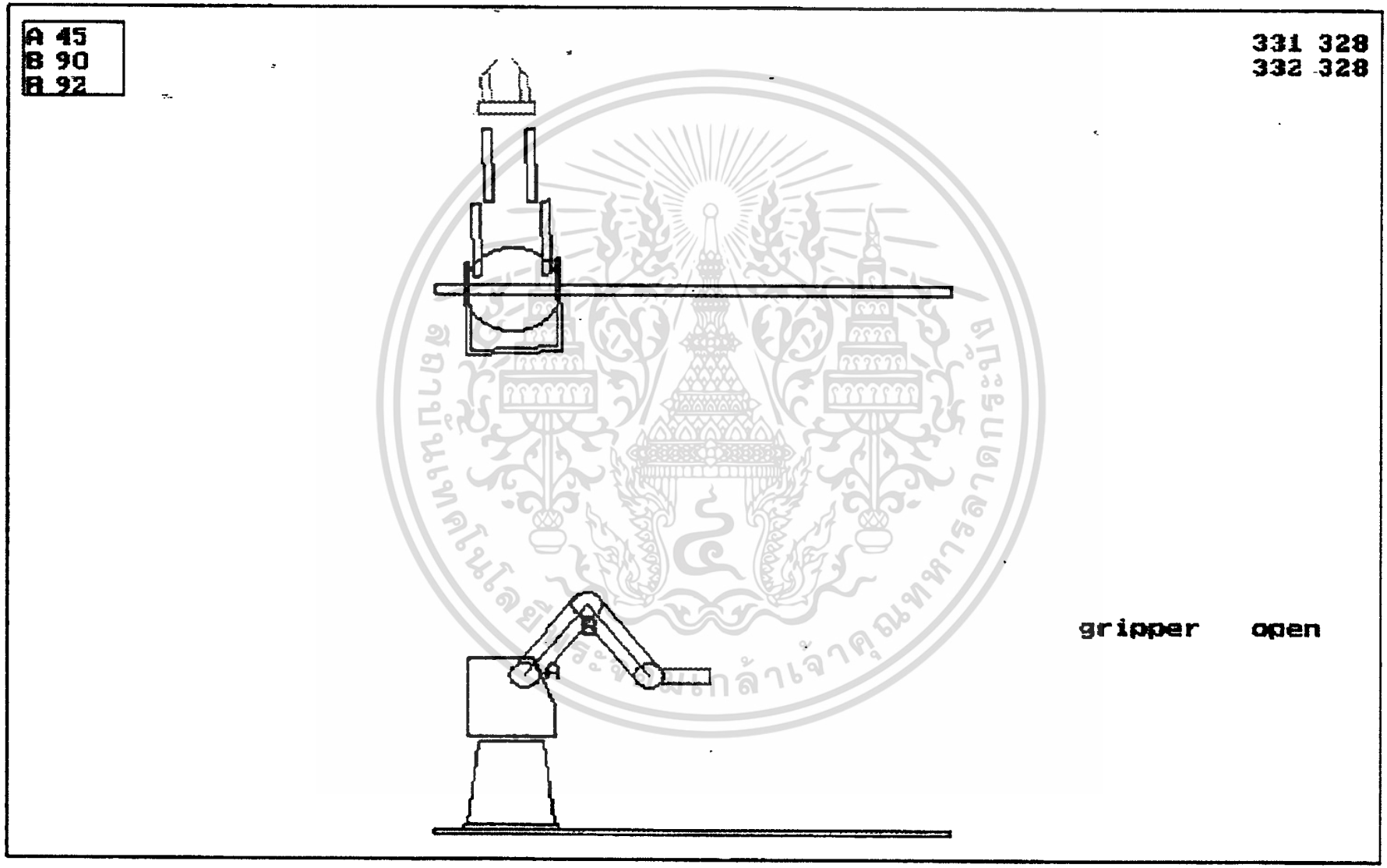
ถัดจากนี้ไปจะเป็นการแสดง FLOW CHARTS และ ทิศทางการเคลื่อนที่เป็น step ของระบบนี้



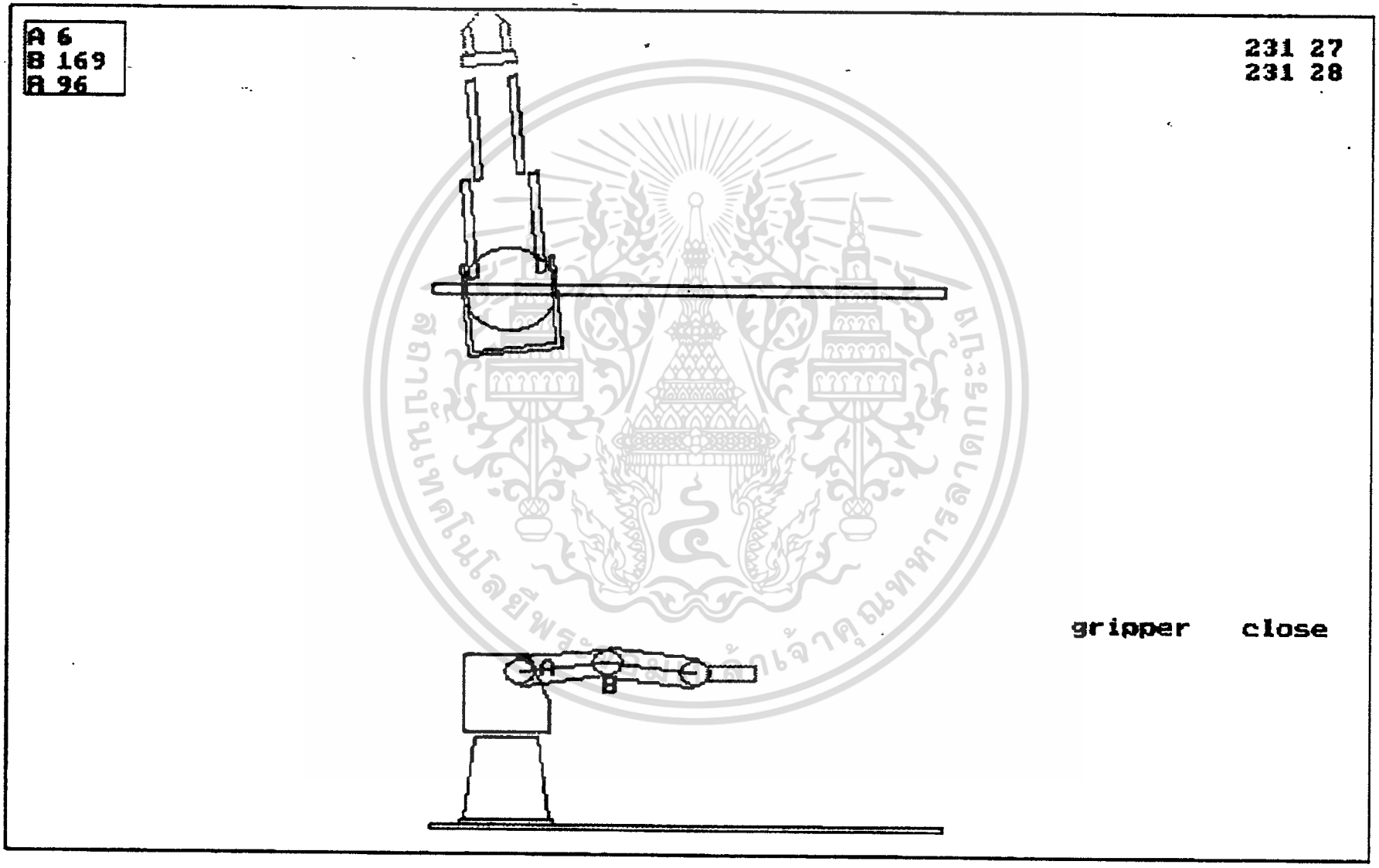
รูปที่ 4 แสดงการสร้างภาพของแขนกลตอนเริ่มต้น



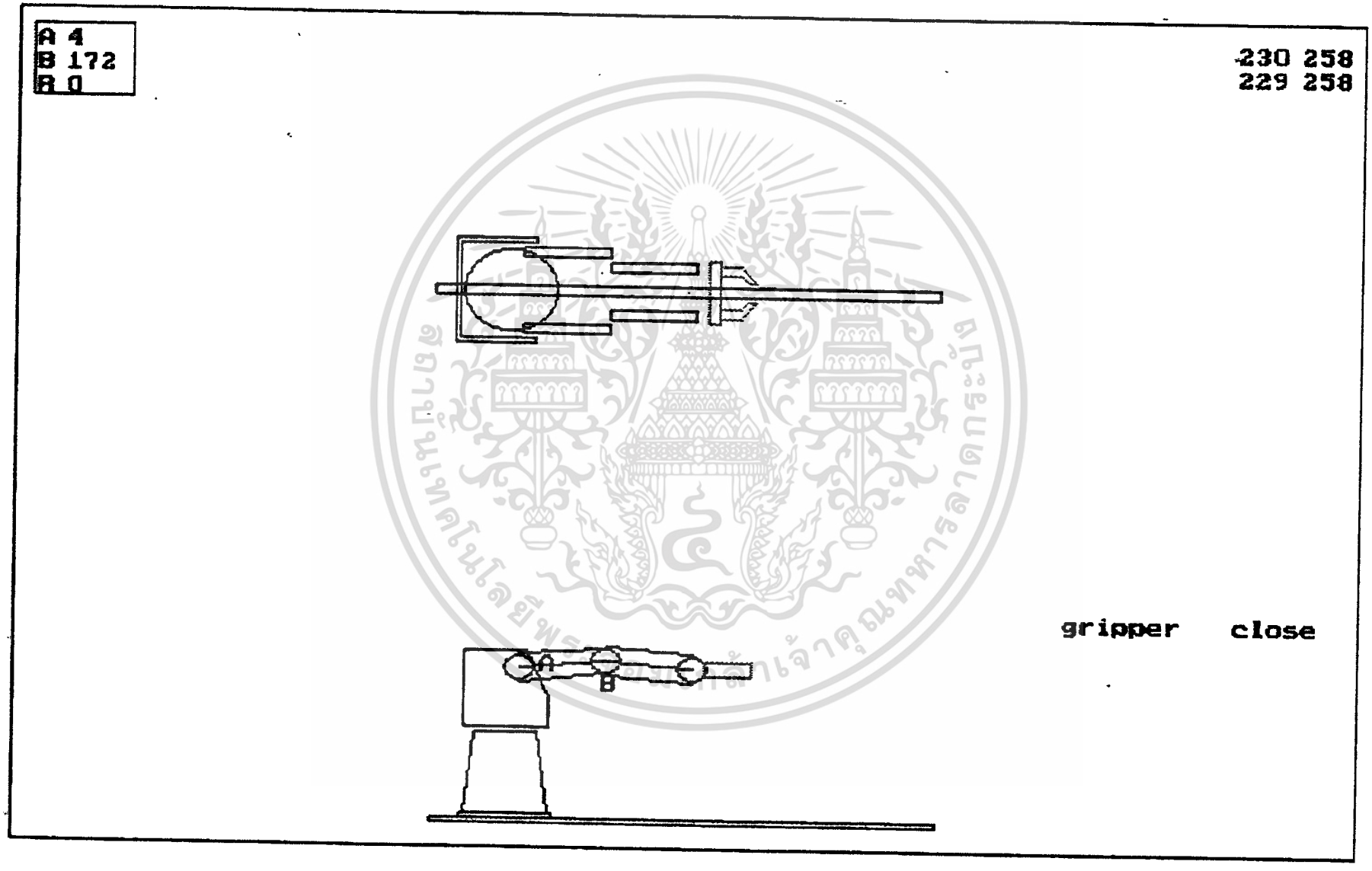
รูปที่ 5 แสดงการสร้างภาพของแขนกลขั้นตอนที่ 1



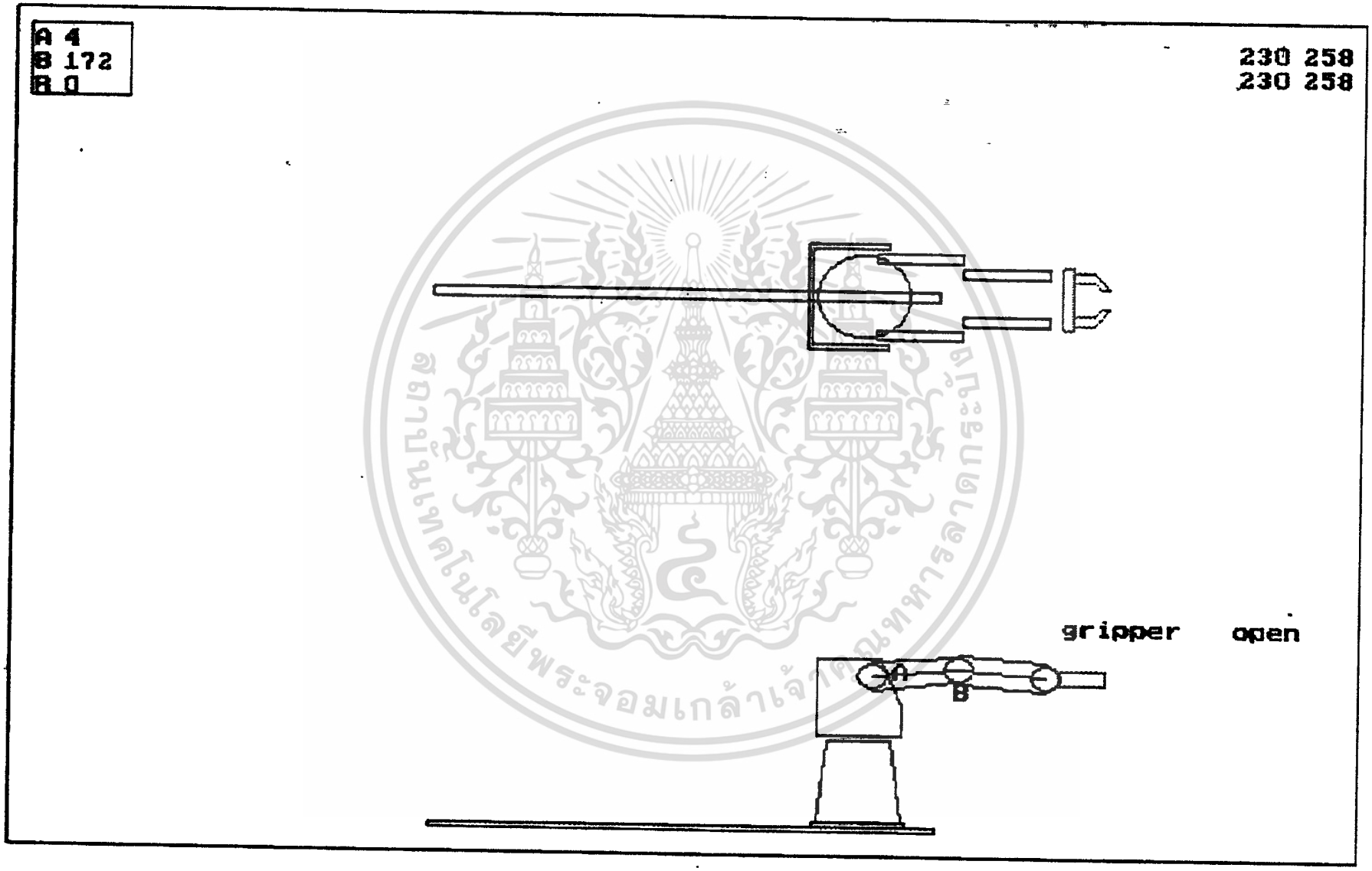
รูปที่ 6 แสดงการสร้างภาพของแขนกลขั้นตอนที่ 2



รูปที่ 7 แสดงการสร้างภาพของแขนกลขั้นตอนที่ 3



รูปที่ 8 แสดงการสร้างภาพของแขนกลขั้นตอนที่ 4



รูปที่ 9 แสดงการสร้างภาพของแขนกลขั้นตอนที่ 5

บทที่ 6

บทสรุปและวิจารณ์

จากโครงงานระบบช่วยสอนหุ่นยนต์นี้ ทำให้ผู้ควบคุมสามารถมองเห็นการเคลื่อนที่ของแขนกลจำลองได้ทุกตำแหน่ง ทุกมุม และสามารถทราบค่ามุม A , B และค่าระยะห่างจากจุดกึ่งกลาง Slideway จากหน้าจอภาพได้ จำนวนครั้งในการเก็บเรคคอร์ด ค่าที่เก็บเรคคอร์ดไว้ในแต่ละครั้ง สถานะการปิด-เปิดของ Gripper และการส่งข้อมูลออกจากพอร์ต จะแสดงให้เห็นบนหน้าจอภาพเช่นกัน จึงทำให้ผู้ควบคุมสามารถกำหนดทิศทางและตำแหน่งของแขนกลได้ตามที่ต้องการ ก่อนที่จะส่งค่าไปให้แขนกลจริงปฏิบัติตามแต่ละเรคคอร์ดที่เก็บไว้ และทำซ้ำเรื่อยๆจนผู้ควบคุมเห็นว่าการปฏิบัติงานของแขนกลสิ้นสุดแล้วจึงทำการออกจาก Loop การทำงานที่เก็บไว้เป็นเรคคอร์ดนั้นๆ หรือถ้าเรคคอร์ดที่เก็บไว้ไม่เป็นที่ต้องการก็สามารถเก็บเรคคอร์ดได้ใหม่อีกเช่นกัน หรือถ้าผู้ควบคุมทำการเก็บข้อมูลการปฏิบัติงานจากแขนกลจำลองบนจอภาพแล้วเกิดมีเหตุผิดปกติขึ้นบน Line การปฏิบัติงานบาง Line ก่อนการสั่งให้แขนกลจริงปฏิบัติงานตามข้อมูลที่เก็บไว้ ก็สามารถเปลี่ยนข้อมูลการปฏิบัติงานใหม่ได้ทันที เพื่อหลีกเลี่ยงอุบัติเหตุที่จะเกิดขึ้นเป็นต้น

หนังสืออ้างอิง

SCORBOT-ER III
USER'S MANUAL
6th EDITION
CATALOG NO. 100038
ESHED ROBOTEC 1982 LTD.

ภาคผนวก

ลักษณะการแสดงผลบนหน้าจอ

การกำหนดค่าตัวแปรต่างๆ บนจอภาพมีความหมายดังต่อไปนี้

A หมายถึง ค่ามุมที่แขนกลในส่วน upperarm ทำมุมกับ ระนาบแนวนอนของแขนกล

ถ้ามุม A มีค่าเป็น + จะหมายถึง upperarm ทำมุมเหนือแกนในระนาบแนวนอน

ถ้ามุม A มีค่าเป็น 0 จะหมายถึง upperarm ทำมุมอยู่ในระนาบแนวนอน

ถ้ามุม A มีค่าเป็น - จะหมายถึง upperarm ทำมุมใต้แกนในระนาบแนวนอน

B หมายถึง ค่ามุมที่แขนกลในส่วน upperarm ทำมุมกับ forearm ของแขนกล

R หมายถึง ค่ามุมที่แขนกลทั้งแขน ทำมุมกับ slideway ในด้าน TOP VIEW

ถ้ามุม R มีค่าเป็น + จะหมายถึง แขนกลทั้งแขน ทำมุมเหนือ slideway

ถ้ามุม R มีค่าเป็น 0 จะหมายถึง แขนกลทั้งแขน ทำมุมอยู่ใน slideway

ถ้ามุม R มีค่าเป็น - จะหมายถึง แขนกลทั้งแขน ทำมุมใต้ slideway

gripper open หมายถึง ปลายแขนที่ใช้จับยึดวัตถุทำการเปิดออก

gripper close หมายถึง ปลายแขนที่ใช้จับยึดวัตถุทำการปิดเข้า

origin หมายถึง ตำแหน่งที่ body ของแขนกลอยู่ตรงจุดกึ่งกลางของ slideway

ค่าตัวเลขทั้ง 4 ชุด หมายถึงค่าตำแหน่งของ mouse ในปัจจุบันและก่อนหน้านั้น

```
PROGRAM SIZE;
USES GRAPH,CRT,DOS,q1,q2,move;

type PO = record
     orX : integer;
     lx  : integer;
     lz  : integer;
     tx  : integer;
     tx1 : integer;
     tz  : integer;
     xold: integer;
     yold: integer;
     ang : real;
     a   : real;
     b   : real;
     hag : integer;
     grip: string;
     ccc : Boolean;
     END;

VAR
    GraphDriver   : Integer;
    GraphMode     : Integer;
    LowMode       : Integer;
    HighMode      : Integer;
    reg           : registers;
    ang,a,b       : real;
    Xold,yold,i,tx,lx,lz,orX,tx1,tz,p,f,hag : integer;
    q,w : integer;
    s,grip        : string;
    ch            : char;
    RPS           : array[1..20] of PO;
    ccc : Boolean;

BEGIN
     SetG(GraphDriver,GraphMode);
     InitMouse(reg);
     Move_mou_prt(reg,getmaxx div 2, getmaxy div 2);
     xold :=reg.CX ;
     yold :=reg.DX ;
     {Display_pointer(reg);}
     ang := 0;
     p   := 0;
     a   := 45;
     b   := 0;
     orX := 320;
     lx  := 29;
     tx  := 58;
     tx1 := 29;
     hag := 0;
     grip:= 'open';
{     gripper(grip);}
     body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
     body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
     display(orX,p,a,b,ang);
{     GoHome;                    }
WHILE reg.BX<>1 DO
BEGIN

  IF(xold=reg.CX)OR(yold=reg.DX) THEN
  BEGIN
    REPEAT
    Get_ptr_posit_botton_status(reg);
    UNTIL (reg.CX<>xold)OR(reg.DX<>yold)OR(keypressed);
```

```
if keypressed then
begin   ch := readkey;
     if ch = #0 then ch := readkey;
     if ch = #75 then
        begin
        orX := orX - 4;
        if (orX<224) then orX := 224;
        if i = 1 then i := 0 else i := 1;
        SetActivePage(i);
        clearviewport;
        setcolor(15);
        SetActivePage(i);
        body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
        SetActivePage(i);
        body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
        SetActivePage(i);
        display(orX,p,a,b,ang);
        SetVisualPage(i);
        end;

     if ch = #77 then
        begin
        orX := orX + 4;
        if (orX>416) then orX := 416;
        if i = 1 then i := 0 else i := 1;
        SetActivePage(i);
        clearviewport;
        setcolor(15);
        SetActivePage(i);
        body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
        SetActivePage(i);
        body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
        SetActivePage(i);
        display(orX,p,a,b,ang);
        SetVisualPage(i);
        end;

     if ch = #72 then
        begin
        b := b + 1;
        if i = 1 then i := 0 else i := 1;
        SetActivePage(i);
        clearviewport;
        setcolor(15);
        SetActivePage(i);
        body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
        SetActivePage(i);
        body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
        SetActivePage(i);
        display(orX,p,a,b,ang);
        SetVisualPage(i);
        end;

     if ch = #80 then
        begin
        b := b - 1;
        if i = 1 then i := 0 else i := 1;
        SetActivePage(i);
        clearviewport;
        setcolor(15);
        SetActivePage(i);
        body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
        SetActivePage(i);
        body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
```

```
                SetActivePage(i);
                display(orX,p,a,b,ang);
                SetVisualPage(i);
                end;

(********) if ch = #73 then
                begin
                hag := hag + 90;
                if hag >180 then hag := 180;
                if i = 1 then i := 0 else i := 1;
                SetActivePage(i);
                clearviewport;
                setcolor(15);
                SetActivePage(i);
                body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
                SetActivePage(i);
                body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
                SetActivePage(i);
                display(orX,p,a,b,ang);
                SetVisualPage(i);
                end;

(********) if ch = #81 then
                begin
                hag := hag - 90;
                if hag < 0 then hag := 0;
                if i = 1 then i := 0 else i := 1;
                SetActivePage(i);
                clearviewport;
                setcolor(15);
                SetActivePage(i);
                body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
                SetActivePage(i);
                body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
                SetActivePage(i);
                display(orX,p,a,b,ang);
                SetVisualPage(i);
                end;

              if ch = #111 then grip := 'open';

              if ch = #99  then grip := 'close';
              if ch = #114 then ccc  := true;
              if ch = #101 then ccc  := false;
              if ch = #103 then Gohome;

              if ch = #32 then    {space bar}
                begin
                p := p+1;
                str(p,s);
                setcolor(9);
                q := 0;
                w := 0;
                repeat
                setcolor(15);
                rectangle(540,40,600,60);
                rectangle(540+q,40+w,600-q,60-w);
                q := q+1;
                w := w+1;
                if w > 20 then w := 20;
                until (q=60)and(w=20);
                setcolor(red);
                outtextxy(545,50,'READ ');
                outtextxy(585,50,s);
                RPS[p].orx := orX;
```

```
                RPS[p].lx  := lx;
                RPS[p].lz  := lz;
                RPS[p].tx  := tx;
                RPS[p].tx1 := tx1;
                RPS[p].tz  := tz;
                RPS[p].xold:= xold;
                RPS[p].yold:= yold;
                RPS[p].ang := ang;
                RPS[p].a   := a;
                RPS[p].b   := b;
                RPS[p].hag := hag;
                RPS[p].grip:= grip;
                RPS[p].ccc := ccc;
                f := p;
                end;

          if ch = #13 then   {enter}
     (*13*) begin

                 p := 1;
                while(p<>f+1) do
     (*1*f*)    Begin
                if i = 1 then i := 0 else i := 1;
                SetActivePage(i);
                clearviewport;
                setcolor(15);
                SetActivePage(i);
                body1(reg,RPS[p].xold,RPS[p].yold,RPS[p].lx
                ,RPS[p].tx1,RPS[p].orX,RPS[p].hag,RPS[p].ang,s);
                SetActivePage(i);
                body2(RPS[p].orX,RPS[p].lx,RPS[p].lz,RPS[p].tx,RPS[p].tx1
                ,RPS[p].tz,RPS[p].hag,RPS[p].a,RPS[p].b,RPS[p].ang
                ,RPS[p].grip);
                SetActivePage(i);
                displayrec(RPS[p].orX,p,RPS[p].a,RPS[p].b,RPS[p].ang);
                SetActivePage(i);
                setcolor(13);
                outtextxy(545,50,'RUN ');
                str(p,s);
                SetActivePage(i);
                outtextxy(585,50,s);
                SetVisualPage(i);

                          (**SEND**)

           if (p > 1) then move_it54(r(RPS[p].orX),r(RPS[p-1].orX))
                      else move_it54(r(RPS[p].orX),320);


                if i = 1 then i := 0 else i := 1;
     (*old*)    SetActivePage(i);
     (*picture*) clearviewport;
                setcolor(15);
                SetActivePage(i);
                body1(reg,RPS[p].xold,RPS[p].yold,RPS[p].lx
                ,RPS[p].tx1,RPS[p].orX,RPS[p].hag,RPS[p].ang,s);
                SetActivePage(i);
                body2(RPS[p].orX,RPS[p].lx,RPS[p].lz,RPS[p].tx
                ,RPS[p].tx1,RPS[p].tz,RPS[p].hag,RPS[p].a,RPS[p].b
                ,RPS[p].ang,RPS[p].grip);
                SetActivePage(i);
                displayrec(RPS[p].orX,p,RPS[p].a,RPS[p].b,RPS[p].ang);
                SetActivePage(i);
                setcolor(13);
                outtextxy(545,50,'RUN ');
```

```
                str(p,s);
                SetActivePage(i);
                outtextxy(585,50,s);
          (**) SetVisualPage(i);

          if (p > 1) then move_it49(r(RPS[p].ang),r(RPS[p-1].ang))
                     else move_it49(r(RPS[p].ang),0);
                if i = 1 then i := 0 else i := 1;
(*old*)         SetActivePage(i);
(*picture*)     clearviewport;
                setcolor(15);
                SetActivePage(i);
                body1(reg,RPS[p].xold,RPS[p].yold,RPS[p].lx
                ,RPS[p].tx1,RPS[p].orX,RPS[p].hag,RPS[p].ang,s);
                SetActivePage(i);
                body2(RPS[p].orX,RPS[p].lx,RPS[p].lz,RPS[p].tx
                ,RPS[p].tx1,RPS[p].tz,RPS[p].hag,RPS[p].a,RPS[p].b
                ,RPS[p].ang,RPS[p].grip);
                SetActivePage(i);
                displayrec(RPS[p].orX,p,RPS[p].a,RPS[p].b,RPS[p].ang);
                SetActivePage(i);
                setcolor(13);
                outtextxy(545,50,'RUN ');
                str(p,s);
                SetActivePage(i);
                outtextxy(585,50,s);
          (**) SetVisualPage(i);


          if (p > 1) then move_it50(r(RPS[p].a),r(RPS[p-1].a))
                     else move_it50(r(RPS[p].a),45);

                if i = 1 then i := 0 else i := 1;
(*old*)         SetActivePage(i);
(*picture*)     clearviewport;
                setcolor(15);
                SetActivePage(i);
                body1(reg,RPS[p].xold,RPS[p].yold,RPS[p].lx
                ,RPS[p].tx1,RPS[p].orX,RPS[p].hag,RPS[p].ang,s);
                SetActivePage(i);
                body2(RPS[p].orX,RPS[p].lx,RPS[p].lz,RPS[p].tx
                ,RPS[p].tx1,RPS[p].tz,RPS[p].hag;RPS[p].a,RPS[p].b
                ,RPS[p].ang,RPS[p].grip);
                SetActivePage(i);
                displayrec(RPS[p].orX,p,RPS[p].a,RPS[p].b,RPS[p].ang);
                SetActivePage(i);
                setcolor(13);
                outtextxy(545,50,'RUN ');
                str(p,s);
                SetActivePage(i);
                outtextxy(585,50,s);
          (**) SetVisualPage(i);

          if (p > 1) then move_it51(r(RPS[p].a),r(RPS[p-1].a))
                     else move_it51(r(RPS[p].a),45);

          if (RPS[p].grip <> RPS[p-1].grip) then gripper(RPS[p].grip);
          if (RPS[p].ccc = true) then rotate;
               p := p+1;
               if p = f+1 then p := 1;
     (*1*f*)End;
(*13*) end;
               end;

END;
```

```
IF(xold<>reg.CX)OR(yold<>reg.DX) THEN
  BEGIN
  If( abs(yold-reg.DX) > abs(reg.CX-xold) ) then
    Begin

    if(yold>reg.DX) then
    begin
        ang := ang+4;
        if (ang>159) then ang := 159;

    end;

    if(yold<reg.DX) then
    begin
        ang := ang-4;
        if (ang<-159) then ang := -159;
    end;

        if i = 1 then i := 0 else i := 1;
              SetActivePage(i);
              clearviewport;
              setcolor(15);
              SetActivePage(i);
              body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
              SetActivePage(i);
              body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
              SetActivePage(i);
              display(orX,p,a,b,ang);
              SetVisualPage(i);

  End;


  If( abs(yold-reg.DX) < abs(reg.CX-xold) ) then
    Begin
    if(xold>reg.CX) then
    begin
    a  := a+1.875;
       if a>75 then a:= 75;
    end;


    if(xold<reg.CX) then
    begin
    a  := a-1.875;
       if a<0  then a := 0;
    end;

    if(xold<reg.CX) then
        begin
              if i = 1 then i := 0 else i := 1;
              SetActivePage(i);
              clearviewport;
              setcolor(15);
              SetActivePage(i);
              body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
              SetActivePage(i);
              body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
              SetActivePage(i);
              display(orX,p,a,b,ang);
              SetVisualPage(i);
        end;
```

```
            if(xold>reg.CX) then
                begin
                    if i = 1 then i := 0 else i := 1;
                    SetActivePage(i);
                    clearviewport;
                    setcolor(15);
                    SetActivePage(i);
                    body1(reg,xold,yold,lx,tx1,orX,hag,ang,s);
                    SetActivePage(i);
                    body2(orX,lx,lz,tx,tx1,tz,hag,a,b,ang,grip);
                    SetActivePage(i);
                    display(orX,p,a,b,ang);
                    SetVisualPage(i);
                end;

          End;

        END;{check move}
      if(reg.CX = 0) then Move_mou_prt(reg,getmaxx-1,yold);
      if(reg.DX = 0) then Move_mou_prt(reg,xold,getmaxy-1);
      if(reg.CX = getmaxx) then Move_mou_prt(reg,1,reg.DX);
      if(reg.DX = getmaxy) then Move_mou_prt(reg,reg.CX,1);

       xold :=reg.CX ;
       yold :=reg.DX ;

END;
Closegraph;
END.
```

```
    unit q1;
 interface
 uses graph,crt,dos;
 function r(rea_1 : real) : integer;
 Procedure SetG(var GraphDriver,GraphMode : integer);
 procedure InitMouse(var reg : registers);
 procedure Display_pointer(var reg : registers);
 procedure Get_ptr_posit_botton_status(var reg : registers);
 procedure hide_pointer(var reg : registers);
 procedure Move_mou_prt(var reg : registers; xx,yy : integer);
 Procedure body1(var reg : registers;
                 var xold,yold,lx,tx1,orX,hag : integer;
                 var ang : real;
                 var s   : string);

 Implementation

 Procedure SetG;
   begin
     GraphDriver := EGA;
     GraphMode := EGAHi;
     InitGraph(GraphDriver, GraphMode, '');
     if GraphResult <> grOk then Halt(1);
   end;

 procedure InitMouse(var reg : registers);
 begin
       reg.AX:=0; {func 00H}
       intr($33,reg);
 end;

 procedure Display_pointer(var reg : registers);
 begin
       reg.AX:=$01;
       intr($33,reg);
 end;

 procedure Get_ptr_posit_botton_status(var reg : registers);
 begin
      reg.AX:=$03;
      intr($33,reg);
      { display coordinate to crt }

 end;

 procedure hide_pointer(var reg : registers);
 begin
       reg.AX:=$02;
       intr($33,reg);
 end;

 procedure Move_mou_prt(var reg : registers;  xx,yy : integer);
 begin
      reg.AX:=$04;
      reg.CX:= xx;
      reg.DX:= yy;
      intr($33,reg);
 end;

 function r(rea_1 : real) : integer;
 begin
      r := round(rea_1);
 end;

 function calx(choose:integer; var ang : real;
```

```
                var lx,tx1: integer) : integer;
begin
case choose of
1 : calx := r(6*cos(ang)-14*sin(ang));
2 : calx := r(6*cos(ang)-14*sin(ang)+lx*cos(ang));
3 : calx := r(6*cos(ang)-14*sin(ang)+lx*cos(ang)+2*sin(ang));
4 : calx := r(6*cos(ang)-14*sin(ang)+lx*cos(ang)+2*sin(ang)
           +tx1*cos(ang));
end
end;

Function cax(choose:integer; var ang : real;
             var lx,tx1: integer) : integer;
begin
case choose of
1 : cax := r(6*cos(ang)-18*sin(ang));
2 : cax := r(6*cos(ang)-18*sin(ang)+lx*cos(ang));
3 : cax := r(6*cos(ang)-18*sin(ang)+lx*cos(ang)+10*sin(ang));
4 : cax := r(6*cos(ang)-18*sin(ang)+lx*cos(ang)+10*sin(ang)
           +tx1*cos(ang));
end
end;

Function clx(choose:integer; var ang : real;
             var lx,tx1: integer) : integer;
begin
case choose of
1 : clx := r(12*cos(ang)-20*sin(ang));
2 : clx := r(12*cos(ang)-20*sin(ang)-36*cos(ang));
3 : clx := r(12*cos(ang)-20*sin(ang)-36*cos(ang)+40*sin(ang));
4 : clx := r(12*cos(ang)-20*sin(ang)-36*cos(ang)+40*sin(ang)
           +36*cos(ang));
end
end;

Function calx1(choose:integer; var ang : real;
               var lx,tx1: integer) : integer;
begin
case choose of
1 : calx1 := r(6*cos(ang)+14*sin(ang));
2 : calx1 := r(6*cos(ang)+14*sin(ang)+lx*cos(ang));
3 : calx1 := r(6*cos(ang)+14*sin(ang)+lx*cos(ang)-2*sin(ang));
4 : calx1 := r(6*cos(ang)+14*sin(ang)+lx*cos(ang)-2*sin(ang)
           +tx1*cos(ang));
end
end;

Function cax1(choose:integer; var ang : real;
              var lx,tx1: integer) : integer;
begin
case choose of
1 : cax1 := r(6*cos(ang)+18*sin(ang));
2 : cax1 := r(6*cos(ang)+18*sin(ang)+lx*cos(ang));
3 : cax1 := r(6*cos(ang)+18*sin(ang)+lx*cos(ang)-10*sin(ang));
4 : cax1 := r(6*cos(ang)+18*sin(ang)+lx*cos(ang)-10*sin(ang)
           +tx1*cos(ang));
end
end;

Function clx1(choose:integer; var ang : real;
              var lx,tx1: integer) : integer;
begin
case choose of
1 : clx1 := r(12*cos(ang)-22*sin(ang));
2 : clx1 := r(12*cos(ang)-22*sin(ang)-38*cos(ang));
```

```
3 : clx1 := r(12*cos(ang)-22*sin(ang)-38*cos(ang)+44*sin(ang));
4 : clx1 := r(12*cos(ang)-22*sin(ang)-38*cos(ang)+44*sin(ang)
           +38*cos(ang));
end
end;

Function caly(choose:integer; var ang : real;
              var lx,tx1 : integer) : integer;
begin
case choose of
1 : caly := r(-6*sin(ang)-14*cos(ang));
2 : caly := r(-6*sin(ang)-14*cos(ang)-lx*sin(ang));
3 : caly := r(-6*sin(ang)-14*cos(ang)-lx*sin(ang)+2*cos(ang));
4 : caly := r(-6*sin(ang)-14*cos(ang)-lx*sin(ang)+2*cos(ang)
           -tx1*sin(ang));
end
end;

Function cay(choose:integer; var ang : real;
             var lx,tx1 : integer) : integer;
begin
case choose of
1 : cay := r(-6*sin(ang)-18*cos(ang));
2 : cay := r(-6*sin(ang)-18*cos(ang)-lx*sin(ang));
3 : cay := r(-6*sin(ang)-18*cos(ang)-lx*sin(ang)+10*cos(ang));
4 : cay := r(-6*sin(ang)-18*cos(ang)-lx*sin(ang)+10*cos(ang)
           -tx1*sin(ang));
end
end;

Function cly(choose:integer; var ang : real;
             var lx,tx1 : integer) : integer;
begin
case choose of
1 : cly := r(-12*sin(ang)-20*cos(ang));
2 : cly := r(-12*sin(ang)-20*cos(ang)+36*sin(ang));
3 : cly := r(-12*sin(ang)-20*cos(ang)+36*sin(ang)+40*cos(ang));
4 : cly := r(-12*sin(ang)-20*cos(ang)+36*sin(ang)+40*cos(ang)
           -36*sin(ang));
end
end;

Function caly1(choose:integer;var ang : real;
               var lx,tx1 : integer) : integer;
begin
case choose of
1 : caly1 := r(-6*sin(ang)+14*cos(ang));
2 : caly1 := r(-6*sin(ang)+14*cos(ang)-lx*sin(ang));
3 : caly1 := r(-6*sin(ang)+14*cos(ang)-lx*sin(ang)-2*cos(ang));
4 : caly1 := r(-6*sin(ang)+14*cos(ang)-lx*sin(ang)-2*cos(ang)
           -tx1*sin(ang));
end
end;

Function cay1(choose:integer;var ang : real;
              var lx,tx1 : integer) : integer;
begin
case choose of
1 : cay1 := r(-6*sin(ang)+18*cos(ang));
2 : cay1 := r(-6*sin(ang)+18*cos(ang)-lx*sin(ang));
3 : cay1 := r(-6*sin(ang)+18*cos(ang)-lx*sin(ang)-10*cos(ang));
4 : cay1 := r(-6*sin(ang)+18*cos(ang)-lx*sin(ang)-10*cos(ang)
           -tx1*sin(ang));
end
end;
```

```
Function cly1(choose:integer; var ang : real;
              var lx,tx1 : integer) : integer;
begin
case choose of
1 : cly1 := r(-12*sin(ang)-22*cos(ang));
2 : cly1 := r(-12*sin(ang)-22*cos(ang)+38*sin(ang));
3 : cly1 := r(-12*sin(ang)-22*cos(ang)+38*sin(ang)+44*cos(ang));
4 : cly1 := r(-12*sin(ang)-22*cos(ang)+38*sin(ang)+44*cos(ang)
           -38*sin(ang));
end
end;


function ttx(choose : integer; var ang : real):real;
begin
case choose of
1 : ttx := 22*cos(ang);
2 : ttx := 13*sin(ang+pi);
3 : ttx := ttx(2,ang)+5*cos(ang);
4 : ttx := ttx(3,ang)-13*sin(ang+pi);
5 : ttx := ttx(3,ang)-2*sin(ang+pi);
6 : ttx := ttx(5,ang)+10*cos(ang);
7 : ttx := ttx(3,ang)-6*sin(ang+pi);
8 : ttx := ttx(7,ang)+10*cos(ang);
9 : ttx := ttx(6,ang)-7*sin(ang+pi)+7*cos(ang);
10: ttx := ttx(6,ang)-3*sin(ang+pi)+3*cos(ang);
end
end;



(**********************)

function hdx(choose : integer; var b : real): real;
begin
case choose of
1 : hdx := 22*cos(b);
2 : hdx := -13*sin(b+pi);
3 : hdx := hdx(2,b)+5*cos(b);
4 : hdx := hdx(3,b)+13*sin(b+pi);
5 : hdx := hdx(3,b)+2*sin(b+pi);
6 : hdx := hdx(5,b)+10*cos(b);
7 : hdx := hdx(3,b)+6*sin(b+pi);
8 : hdx := hdx(7,b)+10*cos(b);
9 : hdx := hdx(6,b)+7*sin(b+pi)+7*cos(b);
10: hdx := hdx(6,b)+3*sin(b+pi)+3*cos(b);
end
end;

function hdz(choose : integer; var  ang: real): real;
begin
case choose of
1 : hdz := 22*sin(ang);
2 : hdz := -13*cos(ang+pi);
3 : hdz := hdz(2,ang)-5*sin(ang);
4 : hdz := hdz(3,ang)+13*cos(ang+pi);
5 : hdz := hdz(3,ang)+2*cos(ang+pi);
6 : hdz := hdz(5,ang)-10*sin(ang);
7 : hdz := hdz(3,ang)+6*cos(ang+pi);
8 : hdz := hdz(7,ang)-10*sin(ang);
9 : hdz := hdz(6,ang)+7*sin(ang+pi)-7*cos(ang);
10: hdz := hdz(8,ang)+3*sin(ang+pi)-3*cos(ang);
end
end;
```

```
procedure handd(var orX: integer; var a,b : real);
var hx,hz : integer;
begin
    hx := round(orX+6+82*cos(a)*cos(b)+6*cos(b));
    hz := round(274-82*cos(a)*sin(b)-6*sin(b));
    setcolor(red);
{    line(hx,hz,hx+r(hcx(1,b)),hz-r(hcz(1,b)));

    line(hx,hz,hx+r(hcx(2,b)),hz+r(hcz(2,b)));
    line(hx+r(hcx(2,b)),hz+r(hcz(2,b)),hx+r(hcx(3,b)),hz+r(hcz(3,b)));
    line(hx+r(hcx(3,b)),hz+r(hcz(3,b)),hx+r(hcx(4,b)),hz+r(hcz(4,b)));
    line(hx+r(hcx(5,b)),hz+r(hcz(5,b)),hx+r(hcx(6,b)),hz+r(hcz(6,b)));
    line(hx+r(hcx(7,b)),hz+r(hcz(7,b)),hx+r(hcx(8,b)),hz+r(hcz(8,b)));
    line(hx+r(hcx(6,b)),hz+r(hcz(6,b)),hx+r(hcx(9,b)),hz+r(hcz(9,b)));
    line(hx+r(hcx(8,b)),hz+r(hcz(8,b)),hx+r(hcx(10,b)),hz+r(hcz(10,b)));
    line(hx+r(hcx(9,b)),hz+r(hcz(9,b)),hx+r(hcx(10,b)),hz+r(hcz(10,b)));

    line(hx,hz,hx+r(hdx(2,b)),hz+r(hdz(2,b)));
    line(hx+r(hdx(2,b)),hz+r(hdz(2,b)),hx+r(hdx(3,b)),hz+r(hdz(3,b)));
    line(hx+r(hdx(3,b)),hz+r(hdz(3,b)),hx+r(hdx(4,b)),hz+r(hdz(4,b)));
    line(hx+r(hdx(5,b)),hz+r(hdz(5,b)),hx+r(hdx(6,b)),hz+r(hdz(6,b)));
    line(hx+r(hdx(7,b)),hz+r(hdz(7,b)),hx+r(hdx(8,b)),hz+r(hdz(8,b)));
    line(hx+r(hdx(6,b)),hz+r(hdz(6,b)),hx+r(hdx(9,b)),hz+r(hdz(9,b)));
    line(hx+r(hdx(8,b)),hz+r(hdz(8,b)),hx+r(hdx(10,b)),hz+r(hdz(10,b)));
    line(hx+r(hdx(9,b)),hz+r(hdz(9,b)),hx+r(hdx(10,b)),hz+r(hdz(10,b)));
    }

end;


(**********************)

function tty(choose : integer; var ang : real):real;
begin
case choose of
1 : tty := 22*sin(ang);
2 : tty := 13*cos(ang+pi);
3 : tty := tty(2,ang)-5*sin(ang);
4 : tty := tty(3,ang)-13*cos(ang+pi);
5 : tty := tty(3,ang)-2*cos(ang+pi);
6 : tty := tty(5,ang)-10*sin(ang);
7 : tty := tty(3,ang)-6*cos(ang+pi);
8 : tty := tty(7,ang)-10*sin(ang);
9 : tty := tty(6,ang)+7*sin(ang+pi)+7*cos(ang);
10: tty := tty(8,ang)+3*sin(ang+pi)+3*cos(ang);

end
end;

function tux(choose : integer; var ang : real): real;
begin
case choose of
1 : tux := 22*cos(ang);
2 : tux := -13*sin(ang+pi);
3 : tux := tux(2,ang)+5*cos(ang);
4 : tux := tux(3,ang)+13*sin(ang+pi);
5 : tux := tux(3,ang)+2*sin(ang+pi);
6 : tux := tux(5,ang)+10*cos(ang);
7 : tux := tux(3,ang)+6*sin(ang+pi);
8 : tux := tux(7,ang)+10*cos(ang);
9 : tux := tux(6,ang)+7*sin(ang+pi)+7*cos(ang);
10: tux := tux(6,ang)+3*sin(ang+pi)+3*cos(ang);
end
```

```
end;

function tuy(choose : integer; var  ang: real): real;
begin
case choose of
1 : tuy := 22*sin(ang);
2 : tuy := -13*cos(ang+pi);
3 : tuy := tuy(2,ang)-5*sin(ang);
4 : tuy := tuy(3,ang)+13*cos(ang+pi);
5 : tuy := tuy(3,ang)+2*cos(ang+pi);
6 : tuy := tuy(5,ang)-10*sin(ang);
7 : tuy := tuy(3,ang)+6*cos(ang+pi);
8 : tuy := tuy(7,ang)-10*sin(ang);
9 : tuy := tuy(6,ang)+7*sin(ang+pi)-7*cos(ang);
10: tuy := tuy(8,ang)+3*sin(ang+pi)-3*cos(ang);
end
end;

function hnx(choose : integer; var ang : real): real;
begin
case choose of
1 : hnx := 3*cos(ang+pi/2);
2 : hnx := hnx(1,ang)-22*cos(ang);
3 : hnx := hnx(2,ang)+3*cos(ang+pi/2);
end
end;

function hnz(choose : integer; ang : real): real;
begin
case choose of
1 : hnz := 3*sin(ang+pi/2);
2 : hnz := hnz(1,ang)+22*sin(ang);
3 : hnz := hnz(2,ang)-3*sin(ang+pi/2);
end
end;

function hmx(choose : integer; var ang : real): real;
begin
case choose of
1 : hmx := 3*cos(ang+pi/2);
2 : hmx := hmx(1,ang)-22*cos(ang);
end
end;

function hmz(choose : integer; ang : real): real;
begin
case choose of
1 : hmz := -3*sin(ang+pi/2);
2 : hmz := hmz(1,ang)+22*sin(ang);
end
end;

procedure handt(var orX,hag,lx,tx1 : integer; var ang : real);
var ox : integer;
    oy : integer;
begin
     ox := orX+r((12+lx+tx1)*cos(ang));
     oy := 116 -r((12+lx+tx1)*sin(ang));
{line(ox,oy,ox+r(ttx(1,ang)),oy-r(tty(1,ang)));}

IF (hag = 0)OR(hag = 180) then
Begin
line(ox,oy,ox+r(ttx(2,ang)),oy+r(tty(2,ang)));
line(ox+r(ttx(2,ang)),oy+r(tty(2,ang)),ox+r(ttx(3,ang)),oy+r(tty(3,ang)));
line(ox+r(ttx(3,ang)),oy+r(tty(3,ang)),ox+r(ttx(4,ang)),oy+r(tty(4,ang)));
```

```
  line(ox+r(ttx(5,ang)),oy+r(tty(5,ang)),ox+r(ttx(6,ang)),oy+r(tty(6,ang)));
  line(ox+r(ttx(7,ang)),oy+r(tty(7,ang)),ox+r(ttx(8,ang)),oy+r(tty(8,ang)));
  line(ox+r(ttx(6,ang)),oy+r(tty(6,ang)),ox+r(ttx(9,ang)),oy+r(tty(9,ang)));
  line(ox+r(ttx(8,ang)),oy+r(tty(8,ang)),ox+r(ttx(10,ang)),oy+r(tty(10,ang)));
  line(ox+r(ttx(9,ang)),oy+r(tty(9,ang)),ox+r(ttx(10,ang)),oy+r(tty(10,ang)));

  line(ox,oy,ox+r(tux(2,ang)),oy+r(tuy(2,ang)));
  line(ox+r(tux(2,ang)),oy+r(tuy(2,ang)),ox+r(tux(3,ang)),oy+r(tuy(3,ang)));
  line(ox+r(tux(3,ang)),oy+r(tuy(3,ang)),ox+r(tux(4,ang)),oy+r(tuy(4,ang)));
  line(ox+r(tux(5,ang)),oy+r(tuy(5,ang)),ox+r(tux(6,ang)),oy+r(tuy(6,ang)));
  line(ox+r(tux(7,ang)),oy+r(tuy(7,ang)),ox+r(tux(8,ang)),oy+r(tuy(8,ang)));
  line(ox+r(tux(6,ang)),oy+r(tuy(6,ang)),ox+r(tux(9,ang)),oy+r(tuy(9,ang)));
  line(ox+r(tux(8,ang)),oy+r(tuy(8,ang)),ox+r(tux(10,ang)),oy+r(tuy(10,ang)));
  line(ox+r(tux(9,ang)),oy+r(tuy(9,ang)),ox+r(tux(10,ang)),oy+r(tuy(10,ang)));
  End;

  IF hag = 90 then
  Begin


     line(ox,oy,ox+r(hnx(1,ang)),oy-r(hnz(1,ang)));
     line(ox+r(hnx(1,ang)),oy-r(hnz(1,ang)),ox-r(hnx(2,ang))
     ,oy-r(hnz(2,ang)));
     line(ox-r(hnx(2,ang)),oy-r(hnz(2,ang)),ox-r(hnx(3,ang))
     ,oy-r(hnz(3,ang)));
   {  line(ox+r(hnx(1,ang)),oy+r(hnz(1,ang)),ox-r(hnx(2,ang))
    ,oy+r(hnz(2,ang)));}

     line(ox,oy,ox-r(hmx(1,ang)),oy-r(hmz(1,ang)));
     line(ox-r(hmx(1,ang)),oy-r(hmz(1,ang)),ox-r(hmx(2,ang))
     ,oy-r(hmz(2,ang)));
     line(ox-r(hnx(3,ang)),oy-r(hnz(3,ang)),ox-r(hmx(2,ang))
     ,oy-r(hmz(2,ang)));
   {   line(ox+r(hmx(2,ang)),oy+r(hnz(2,ang)),ox+r(hmx(3,ang))
      ,oy+r(hnz(3,ang)));}

      Sound(220); delay(220); nosound;
  End;

  end;


  Procedure body1(var reg : registers;
                  var xold,yold,lx,tx1,orX,hag : integer;
                  var ang : real;
                  var s   : string);
  begin

       ang := pi/180*ang;
       Rectangle(0,0,GETMAXX,GETMAXY);

       Circle(orX,116,22);
       Putpixel(320,116,12);
  (****){line(orX,116,orX+round(114*cos(ang)),116-round(114*sin(ang)) );}
       setcolor(3);
       rectangle(200,114,440,118);

       setcolor(red);
       handt(orX,hag,lx,tx1,ang);

       setcolor(12);
       line(orX+clx(1,ang,lx,tx1),116+cly(1,ang,lx,tx1)
       ,orX+clx(2,ang,lx,tx1),116+cly(2,ang,lx,tx1));
       line(orX+clx(3,ang,lx,tx1),116+cly(3,ang,lx,tx1)
       ,orX+clx(2,ang,lx,tx1),116+cly(2,ang,lx,tx1));
```

```
    line(orX+clx(3,ang,lx,tx1),116+cly(3,ang,lx,tx1)
    ,orX+clx(4,ang,lx,tx1),116+cly(4,ang,lx,tx1));

    line(orX+clx1(1,ang,lx,tx1),116+cly1(1,ang,lx,tx1)
    ,orX+clx1(2,ang,lx,tx1),116+cly1(2,ang,lx,tx1));
    line(orX+clx1(3,ang,lx,tx1),116+cly1(3,ang,lx,tx1)
    ,orX+clx1(2,ang,lx,tx1),116+cly1(2,ang,lx,tx1));
    line(orX+clx1(3,ang,lx,tx1),116+cly1(3,ang,lx,tx1)
    ,orX+clx1(4,ang,lx,tx1),116+cly1(4,ang,lx,tx1));

    line(orX+clx1(1,ang,lx,tx1),116+cly1(1,ang,lx,tx1)
    ,orX+clx(1,ang,lx,tx1),116+cly(1,ang,lx,tx1));
    line(orX+clx(4,ang,lx,tx1),116+cly(4,ang,lx,tx1)
    ,orX+clx1(4,ang,lx,tx1),116+cly1(4,ang,lx,tx1));


    setcolor(13);
    line(orX+calx1(1,ang,lx,tx1),116+caly1(1,ang,lx,tx1)
    ,orX+calx1(2,ang,lx,tx1),116+caly1(2,ang,lx,tx1));
    line(orX+calx(1,ang,lx,tx1),116+caly(1,ang,lx,tx1)
    ,orX+calx(2,ang,lx,tx1),116+caly(2,ang,lx,tx1));

    line(orX+cax1(1,ang,lx,tx1),116+cay1(1,ang,lx,tx1)
    ,orX+cax1(2,ang,lx,tx1),116+cay1(2,ang,lx,tx1));
    line(orX+cax(1,ang,lx,tx1),116+cay(1,ang,lx,tx1)
    ,orX+cax(2,ang,lx,tx1),116+cay(2,ang,lx,tx1));

    line(orX+calx1(1,ang,lx,tx1),116+caly1(1,ang,lx,tx1)
    ,orX+cax1(1,ang,lx,tx1),116+cay1(1,ang,lx,tx1));
    line(orX+calx(1,ang,lx,tx1),116+caly(1,ang,lx,tx1)
    ,orX+cax(1,ang,lx,tx1),116+cay(1,ang,lx,tx1));

    line(orX+calx1(2,ang,lx,tx1),116+caly1(2,ang,lx,tx1)
    ,orX+cax1(2,ang,lx,tx1),116+cay1(2,ang,lx,tx1));
    line(orX+calx(2,ang,lx,tx1),116+caly(2,ang,lx,tx1)
    ,orX+cax(2,ang,lx,tx1),116+cay(2,ang,lx,tx1));

    setcolor(14);
    line(orX+calx1(3,ang,lx,tx1),116+caly1(3,ang,lx,tx1)
    ,orX+calx1(4,ang,lx,tx1),116+caly1(4,ang,lx,tx1));
    line(orX+calx(3,ang,lx,tx1),116+caly(3,ang,lx,tx1)
    ,orX+calx(4,ang,lx,tx1),116+caly(4,ang,lx,tx1));

    line(orX+cax1(3,ang,lx,tx1),116+cay1(3,ang,lx,tx1)
    ,orX+cax1(4,ang,lx,tx1),116+cay1(4,ang,lx,tx1));
    line(orX+cax(3,ang,lx,tx1),116+cay(3,ang,lx,tx1)
    ,orX+cax(4,ang,lx,tx1),116+cay(4,ang,lx,tx1));

    line(orX+calx1(3,ang,lx,tx1),116+caly1(3,ang,lx,tx1)
    ,orX+cax1(3,ang,lx,tx1),116+cay1(3,ang,lx,tx1));
    line(orX+calx(3,ang,lx,tx1),116+caly(3,ang,lx,tx1)
    ,orX+cax(3,ang,lx,tx1),116+cay(3,ang,lx,tx1));

    line(orX+calx1(4,ang,lx,tx1),116+caly1(4,ang,lx,tx1)
    ,orX+cax1(4,ang,lx,tx1),116+cay1(4,ang,lx,tx1));
    line(orX+calx(4,ang,lx,tx1),116+caly(4,ang,lx,tx1)
    ,orX+cax(4,ang,lx,tx1),116+cay(4,ang,lx,tx1));

            (*****************************)


    if orX = 320 then

    outtextxy(500,10,'origin');

    Setcolor(2);
```

```
            Str(reg.CX,s);
            Outtextxy(580,10,s);
            Setcolor(3);
            Str(reg.DX,s);
            Outtextxy(610,10,s);

            Setcolor(2);
            Str(xold,s);
            Outtextxy(580,20,s);
            Setcolor(3);
            Str(yold,s);
            Outtextxy(610,20,s);

            ang := 180*ang/pi;

        end;
    end.
```

```
unit q2;
interface
uses graph,crt,dos;
procedure body2( var orX,lx,lz,tx,tx1,tz,hag  : integer;
                 var a,b,ang : real; var grip : string);
procedure display(var orX,p : integer; var a,b,ang : real);
procedure displayrec(var orX,p : integer; var a,b,ang : real);

Implementation



Function cax(choose : integer;var a,b : real) : integer;
begin
case choose of
1 : cax := round(6*sin(a+b));
2 : cax := round(6*sin(a+b)+41*cos(a+b)-sin(a+b));
end
end;

Function caz(choose : integer;var a,b : real) : integer;
begin
case choose of
1 : caz := round(6*cos(a+b));
2 : caz := round(6*cos(a+b)-41*sin(a+b)-cos(a+b));
end
end;

Function cax1(choose : integer;var a,b : real) : integer;
begin
case choose of
1 : cax1 := round(-6*sin(a+b));
2 : cax1 := round(-6*sin(a+b)+41*cos(a+b)-sin(a+b));
end
end;

Function caz1(choose : integer;var a,b : real) : integer;
begin
case choose of
1 : caz1 := round(-6*cos(a+b));
2 : caz1 := round(-6*cos(a+b)-41*sin(a+b)+cos(a+b));
end
end;

Function px1(choose : integer;var a,b : real) : integer;
begin
case choose of
1 : px1 := round(-6*sin(a-b));
end
end;

Function pz1(choose : integer;var a,b : real) : integer;
begin
case choose of
1 : pz1 := round(6*cos(a-b));
end
end;

function r(rea_1 : real) : integer;
begin
     r := round(rea_1);
end;

function hcx(choose : integer; var b : real): real;
```

```
begin
case choose of
1 : hcx := 22*cos(b);
2 : hcx := 13*sin(b+pi);
3 : hcx := hcx(2,b)+5*cos(b);
4 : hcx := hcx(3,b)-13*sin(b+pi);
5 : hcx := hcx(3,b)-2*sin(b+pi);
6 : hcx := hcx(5,b)+10*cos(b);
7 : hcx := hcx(3,b)-6*sin(b+pi);
8 : hcx := hcx(7,b)+10*cos(b);
9 : hcx := hcx(6,b)-7*sin(b+pi)+7*cos(b);
10: hcx := hcx(6,b)-3*sin(b+pi)+3*cos(b);
end
end;

function hcz(choose : integer; var b : real): real;
begin
case choose of
1 : hcz := 22*sin(b);
2 : hcz := 13*cos(b+pi);
3 : hcz := hcz(2,b)-5*sin(b);
4 : hcz := hcz(3,b)-13*cos(b+pi);
5 : hcz := hcz(3,b)-2*cos(b+pi);
6 : hcz := hcz(5,b)-10*sin(b);
7 : hcz := hcz(3,b)-6*cos(b+pi);
8 : hcz := hcz(7,b)-10*sin(b);
9 : hcz := hcz(6,b)+7*sin(b+pi)+7*cos(b);
10: hcz := hcz(8,b)+3*sin(b+pi)+3*cos(b);
end
end;

function hdx(choose : integer; var b : real): real;
begin
case choose of
1 : hdx := 22*cos(b);
2 : hdx := -13*sin(b+pi);
3 : hdx := hdx(2,b)+5*cos(b);
4 : hdx := hdx(3,b)+13*sin(b+pi);
5 : hdx := hdx(3,b)+2*sin(b+pi);
6 : hdx := hdx(5,b)+10*cos(b);
7 : hdx := hdx(3,b)+6*sin(b+pi);
8 : hdx := hdx(7,b)+10*cos(b);
9 : hdx := hdx(6,b)+7*sin(b+pi)+7*cos(b);
10: hdx := hdx(6,b)+3*sin(b+pi)+3*cos(b);
end
end;

function hdz(choose : integer; var b : real): real;
begin
case choose of
1 : hdz := 22*sin(b);
2 : hdz := -13*cos(b+pi);
3 : hdz := hdz(2,b)-5*sin(b);
4 : hdz := hdz(3,b)+13*cos(b+pi);
5 : hdz := hdz(3,b)+2*cos(b+pi);
6 : hdz := hdz(5,b)-10*sin(b);
7 : hdz := hdz(3,b)+6*cos(b+pi);
8 : hdz := hdz(7,b)-10*sin(b);
9 : hdz := hdz(6,b)+7*sin(b+pi)-7*cos(b);
10: hdz := hdz(8,b)+3*sin(b+pi)-3*cos(b);
end
end;

function hnx(choose : integer; var b : real): real;
begin
```

```
case choose of
1 : hnx := 3*cos(b+pi/2);
2 : hnx := hnx(1,b)-22*cos(b);
3 : hnx := hnx(2,b)+3*cos(b+pi/2);
end
end;

function hnz(choose : integer; b : real): real;
begin
case choose of
1 : hnz := 3*sin(b+pi/2);
2 : hnz := hnz(1,b)+22*sin(b);
3 : hnz := hnz(2,b)-3*sin(b+pi/2);
end
end;

function hmx(choose : integer; var b : real): real;
begin
case choose of
1 : hmx := 3*cos(b+pi/2);
2 : hmx := hmx(1,b)-22*cos(b);
end
end;

function hmz(choose : integer; b : real): real;
begin
case choose of
1 : hmz := -3*sin(b+pi/2);
2 : hmz := hmz(1,b)+22*sin(b);
end
end;

procedure handd(var orX,hag: integer; var a,b : real);
var hx,hz : integer;
begin
    hx := round(orX+6+82*cos(a)*cos(b)+6*cos(b));
    hz := round(274-82*cos(a)*sin(b)-6*sin(b));
    setcolor(red);
{    line(hx,hz,hx+r(hcx(1,b)),hz-r(hcz(1,b)));}

IF (hag = 90) then
Begin
    line(hx,hz,hx+r(hcx(2,b)),hz+r(hcz(2,b)));
    line(hx+r(hcx(2,b)),hz+r(hcz(2,b)),hx+r(hcx(3,b)),hz+r(hcz(3,b)));
    line(hx+r(hcx(3,b)),hz+r(hcz(3,b)),hx+r(hcx(4,b)),hz+r(hcz(4,b)));
    line(hx+r(hcx(5,b)),hz+r(hcz(5,b)),hx+r(hcx(6,b)),hz+r(hcz(6,b)));
    line(hx+r(hcx(7,b)),hz+r(hcz(7,b)),hx+r(hcx(8,b)),hz+r(hcz(8,b)));
    line(hx+r(hcx(6,b)),hz+r(hcz(6,b)),hx+r(hcx(9,b)),hz+r(hcz(9,b)));
    line(hx+r(hcx(8,b)),hz+r(hcz(8,b)),hx+r(hcx(10,b)),hz+r(hcz(10,b)));
    line(hx+r(hcx(9,b)),hz+r(hcz(9,b)),hx+r(hcx(10,b)),hz+r(hcz(10,b)));

    line(hx,hz,hx+r(hdx(2,b)),hz+r(hdz(2,b)));
    line(hx+r(hdx(2,b)),hz+r(hdz(2,b)),hx+r(hdx(3,b)),hz+r(hdz(3,b)));
    line(hx+r(hdx(3,b)),hz+r(hdz(3,b)),hx+r(hdx(4,b)),hz+r(hdz(4,b)));
    line(hx+r(hdx(5,b)),hz+r(hdz(5,b)),hx+r(hdx(6,b)),hz+r(hdz(6,b)));
    line(hx+r(hdx(7,b)),hz+r(hdz(7,b)),hx+r(hdx(8,b)),hz+r(hdz(8,b)));
    line(hx+r(hdx(6,b)),hz+r(hdz(6,b)),hx+r(hdx(9,b)),hz+r(hdz(9,b)));
    line(hx+r(hdx(8,b)),hz+r(hdz(8,b)),hx+r(hdx(10,b)),hz+r(hdz(10,b)));
    line(hx+r(hdx(9,b)),hz+r(hdz(9,b)),hx+r(hdx(10,b)),hz+r(hdz(10,b)));
End;

If (hag=0)OR(hag=180) then
Begin
    line(hx,hz,hx+r(hnx(1,b)),hz-r(hnz(1,b)));
    line(hx+r(hnx(1,b)),hz-r(hnz(1,b)),hx-r(hnx(2,b)),hz-r(hnz(2,b)));
```

```
    line(hx-r(hnx(2,b)),hz-r(hnz(2,b)),hx-r(hnx(3,b)),hz-r(hnz(3,b)));
  {   line(hx+r(hnx(1,b)),hz+r(hnz(1,b)),hx-r(hnx(2,b)),hz+r(hnz(2,b)));}

    line(hx,hz,hx-r(hmx(1,b)),hz-r(hmz(1,b)));
    line(hx-r(hmx(1,b)),hz-r(hmz(1,b)),hx-r(hmx(2,b)),hz-r(hmz(2,b)));
    line(hx-r(hnx(3,b)),hz-r(hnz(3,b)),hx-r(hmx(2,b)),hz-r(hmz(2,b)));
{    line(hx+r(hmx(2,b)),hz+r(hnz(2,b)),hx+r(hmx(3,b)),hz+r(hnz(3,b)));}
End;
end;


procedure display(var orX,p : integer; var a,b,ang : real);
var q,w : integer;
    s   : string;
begin
      setcolor(9);
      rectangle(7,7,54,37);
      setcolor(3);
      str(round(a+b),s);
      outtextxy(9,10,'A');
      outtextxy(23,10,s);
      str(round(180-2*a),s);
      outtextxy(9,20,'B');
      outtextxy(23,20,s);
      str(round(ang),s);
      outtextxy(9,30,'R');
      outtextxy(23,30,s);
end;

{procedure chooseblock(var p : integer);
begin
     case p of
     1 : rectangle(7+q,7+ w,54-q,37-w);
     2 : rectangle(7+q,40+w,54-q,70-w);
     3 : rectangle(7+q,73+w,54-q,103-w);
     end
end;}

procedure displayrec(var orX,p : integer; var a,b,ang : real);
type po = record
     a,b,ang : real;
    end;
var q,w    : integer;
    shift  : integer;
    col    : integer;
    pco    : integer;
    s      : string;
    savep  : integer;
    popp   : integer;
    re_p   : integer;
    RPS    : ARRAY[1..20] of po;
begin

     if p>10  then
        begin
        col := 50;
        pco := p mod 10;
        if p = 20 then pco := 10;
        end;

     if p<=10 then
        begin
        col := 0;
        pco := p;
        end;
```

```
        shift := 33*(pco-1);
        savep := 0;
        popp  := 1;
        setcolor(9);
        q := 0;
        w := 0;
        repeat
             setcolor(15);
             rectangle(7+col+q,7+shift+w,54+col-q,37+shift-w);
             q := q+1;
             w := w+1;
             if w > 30 then w := 30;
      until (q=47)and(w=30);
        RPS[p].a  := a;
        RPS[p].b  := b ;
        RPS[p].ang:= ang;

        setcolor(3);
        str(round(RPS[p].a+RPS[p].b),s);
        outtextxy(9+col,10+shift,'A');
        outtextxy(23+col,10+shift,s);
        str(round(180-2*RPS[p].a),s);
        outtextxy(9+col,20+shift,'B');
        outtextxy(23+col,20+shift,s);
        str(round(RPS[p].ang),s);
        outtextxy(9+col,30+shift,'R');
        outtextxy(23+col,30+shift,s);

    {  WHILE savep<>p DO
       BEGIN
        If p<= 10 then
        Begin
        while(savep<>p) do
        begin
            if savep<1 then savep :=1;
            shift := 33*(savep-1);
            if (savep mod 5) = 0 then setcolor(12)
                                 else setcolor(9);
            rectangle(7+col,7+shift,54+col,37+shift);
            setcolor(3);
            str(round(RPS[savep].a+RPS[savep].b),s);
            outtextxy(9+col,10+shift,'A');
            outtextxy(23+col,10+shift,s);
            str(round(180-2*RPS[savep].a),s);
            outtextxy(9+col,20+shift,'B');
            outtextxy(23+col,20+shift,s);
            str(round(RPS[savep].ang),s);
            outtextxy(9+col,30+shift,'R');
            outtextxy(23+col,30+shift,s);
            if p = 1 then savep := 0;
            savep := savep+1;
        end;
        End;

        If p >= 11  then
        Begin
        while(savep<>p) do
        begin
            savep := savep mod 10;
            shift := 33*(savep-1);
            if (savep mod 5) = 4 then setcolor(12)
                                 else setcolor(9);
            rectangle(7+col,40+shift,54+col,70+shift);
            savep := savep+10;
```

```
            setcolor(3);
            str(round(RPS[savep].a+RPS[savep].b),s);
            outtextxy(9+col,10+shift,'A');
            outtextxy(23+col,10+shift,s);
            str(round(180-2*RPS[savep].a),s);
            outtextxy(9+col,20+shift,'B');
            outtextxy(23+col,20+shift,s);
            str(round(RPS[savep].ang),s);
            outtextxy(9+col,30+shift,'R');
            outtextxy(23+col,30+shift,s);
            savep := savep+1;
                              re_p := 11;
                              If re_p = 11 then
                              Begin
                              while(popp<>re_p) do
                              begin
                              shift := 33*(popp-1);
                              if (popp mod 5)=0 then setcolor(12)
                                                else setcolor(9);
                              rectangle(7,7+shift,54,37+shift);
                              setcolor(3);
                              str(round(RPS[popp].a+RPS[popp].b),s);
                              outtextxy(9,10+shift,'A');
                              outtextxy(23,10+shift,s);
                              str(round(180-2*RPS[popp].a),s);
                              outtextxy(9,20+shift,'B');
                              outtextxy(23,20+shift,s);
                              str(round(RPS[popp].ang),s);
                              outtextxy(9,30+shift,'R');
                              outtextxy(23,30+shift,s);
                              popp := popp+1;
                              end;
                              End;


      end;
      End;
  END;}

end;

procedure body2( var orX,lx,lz,tx,tx1,tz,hag : integer;
                 var a,b,ang : real;  var grip : string);
begin

      a := pi/180*a;
      b := pi/180*b;

      setcolor(15);
      rectangle(orX-22,335,orX+22,337);
      line(orX-20,299,orX+20,299);
      line(orX-15,301,orX-20,335);
      line(orX+15,301,orX+20,335);
      line(orX-15,301,orX+15,301);
      line(orX-20,267,orX+10,267);
      line(orX-20,267,orX-20,299);
      line(orX+20,299,orX+20,287);
      line(orX+20,287,orX+10,267);
      circle(orX+6,274,7);

      lx := round(41*cos(a+b));
      lz := round(41*sin(a+b));
      tx := round(82*cos(a)*cos(b));
      tx1:= round(82*cos(a)*cos(b))-lx;
      tz := round(82*cos(a)*sin(b));
```

```
        setcolor(13);
        line(orX+6+cax(1,a,b),274+caz(1,a,b),orX+6+cax(2,a,b)
        ,274+caz(2,a,b));
        line(orX+6+cax1(1,a,b),274+caz1(1,a,b)
        ,orX+6+cax1(2,a,b),274+caz1(2,a,b));
        line(orX+6,274,orX+lx+6,274-lz);
        circle(orX+lx+6,274-lz,7);

        setcolor(14);
         line(orX+lx+6+px1(1,a,b),274-lz+pz1(1,a,b)
         ,orX+tx+6+px1(1,a,b),274-tz+pz1(1,a,b));
         line(orX+lx+6-px1(1,a,b),274-lz-pz1(1,a,b)
         ,orX+tx+6-px1(1,a,b),274-tz-pz1(1,a,b));
         line(orX+lx+6,274-lz,orX+tx+6,274-tz);
         circle(orX+tx+6,274-tz,7);

(****)handd(orx,hag,a,b);

        setcolor(3);
        rectangle(200,337,440,339);

        a := a*180/pi;
        b := b*180/pi;

        outtextxy(orX+16,269,'A');
        outtextxy(orX+4+lx,280-lz,'B');
        outtextxy(500,250,'gripper');
        outtextxy(580,250,grip);

{       display(orx,a,b,ang);}


end;
end.
```

```
unit move;
interface
Procedure GoHome;
procedure ini_port(port,code:BYTE);
procedure send(port,data:byte);
Procedure move_it49(data: integer; olddata : integer);
Procedure move_it50(data: integer; olddata : integer);
Procedure move_it51(data: integer; olddata : integer);
Procedure move_it54(data: integer; olddata : integer);
Procedure gripper(grip : string);
Procedure rotate;

Implementation


uses dos,crt,graph;
type po = record
     choose : byte;
     m      : byte;
     sign   : byte;
     d1     : byte;
     d2     : byte;
     d3     : byte;
     d4     : byte;
     enter  : byte;
     end;

const PT=1;      {com1 := 0 , com2 := 1}

var reg:registers;
    i:integer;
    p:po;
    RPS : array[1..10] of po;
    ch:char;

procedure ini_port(port,code:BYTE);
begin
     reg.dx := port ;
     reg.ah := 0 ; { init }
     reg.al := code ;
     intr($14,reg) ;
end;

procedure send(port,data:byte);
begin
      reg.dx := port ;
      reg.ah := 1 ;   { sent }
      reg.al := data;  { data   }
      intr($14,reg) ;
end;

procedure butport(p:po);
var s : string;
begin

     send(PT,p.choose); outtextxy(500,280,'choose motor: ') ;
     str(reg.al,s);     outtextxy(610,280,s);

     send(PT,p.m);

     send(PT,p.sign);   outtextxy(500,290,'sign    : ') ;
     str(reg.al,s);     outtextxy(600,290,s);

     send(PT,p.d1);     outtextxy(500,300,'data d1: ') ;
     str(reg.al,s);     outtextxy(600,300,s);
```

```
        send(PT,p.d2);      outtextxy(500,310,'data d2: ') ;
        str(reg.al,s);      outtextxy(600,310,s);

        send(PT,p.d3);      outtextxy(500,320,'data d3: ') ;
        str(reg.al,s);      outtextxy(600,320,s);

        send(PT,p.d4);      outtextxy(500,330,'data d4: ') ;
        str(reg.al,s);      outtextxy(600,330,s);

        send(PT,p.enter);
    end;

    Procedure gohome;
    var go_home : po;
    begin
         ini_port(PT,$E7);
         send(PT,50);
         send(PT,77);
         send(PT,43);
         send(PT,48);
         send(PT,54);
         send(PT,53);
         send(PT,48);
         send(PT,13);

         delay(3000);

         send(PT,51);
         send(PT,77);
         send(PT,45);
         send(PT,48);
         send(PT,54);
         send(PT,48);
         send(PT,48);
         send(PT,13);

         delay(3000);

         send(PT,52);
         send(PT,77);
         send(PT,43);
         send(PT,48);
         send(PT,49);
         send(PT,51);
         send(PT,48);
         send(PT,13);

       delay(1500);

         send(PT,53);
         send(PT,77);
         send(PT,45);
         send(PT,48);
         send(PT,49);
         send(PT,51);
         send(PT,48);
         send(PT,13);

    end;

    function  table(dat : integer): integer;
    begin
    case dat of
```

```
     0 : table := 48;
     1 : table := 49;
     2 : table := 50;
     3 : table := 51;
     4 : table := 52;
     5 : table := 53;
     6 : table := 54;
     7 : table := 55;
     8 : table := 56;
     9 : table := 57;
     end
end;

procedure move_it49(data : integer; olddata : integer);
var orbit : integer;
    num   : integer;
    mov_e : integer;
begin
     ini_port(PT,$E7);    { com_ ,contr code }

        mov_e := data-olddata;
        orbit := round(Abs(mov_e*11.5));
        p.choose := 49;
        p.m := 77;
        if(data>olddata)then p.sign := 43;
        if(data<olddata)then p.sign := 45;

        p.d1 := trunc(orbit div 1000);
        p.d1 := table(p.d1);

        num  := trunc(orbit div 100);
        p.d2 := num mod 10;
        p.d2 := table(p.d2);

        num  := trunc(orbit div 10);
        p.d3 := num mod 10;
        p.d3 := table(p.d3);

        p.d4 := orbit mod 10;
        p.d4 := table(p.d4);

        p.enter := 13;

        outport(p);
        delay(orbit*4);
        outtextxy(5,350,'**END BYTE ') ;
end;


procedure move_it54(data : integer; olddata : integer);
var orbit : integer;
    num   : integer;
    mov_e : integer;
    s : string;
begin
     ini_port(PT,$E7);    { com_ ,contr code }
        Str(data,s);
        outtextxy(100,10,s);
        writeln('**START SEND**') ;
        mov_e := data-olddata;
        orbit := round(Abs(mov_e*15.625));
        p.choose := 54;
        p.m := 77;
        if(data<olddata)then p.sign := 45;
        if(data>olddata)then p.sign := 43;
```

```
        p.d1 := trunc(orbit div 1000);
        p.d1 := table(p.d1);

        num  := trunc(orbit div 100);
        p.d2 := num mod 10;
        p.d2 := table(p.d2);

        num  := trunc(orbit div 10);
        p.d3 := num mod 10;
        p.d3 := table(p.d3);

        p.d4 := orbit mod 10;
        p.d4 := table(p.d4);

        p.enter := 13;

        outport(p);
        delay(orbit*5);
        outtextxy(5,350,'**END BYTE ') ;
end;


procedure move_it50(data : integer; olddata : integer);
var orbit : integer;
    num   : integer;
    mov_e : integer;
begin
     ini_port(PT,$E7);    { com_ ,contr code }

        mov_e := data-olddata;
        orbit := round(Abs(mov_e*8.46));
        p.choose := 50;
        p.m := 77;
        if(data<olddata)then p.sign := 43;
        if(data>olddata)then p.sign := 45;

        p.d1 := trunc(orbit div 1000);
        p.d1 := table(p.d1);

        num  := trunc(orbit div 100);
        p.d2 := num mod 10;
        p.d2 := table(p.d2);

        num  := trunc(orbit div 10);
        p.d3 := num mod 10;
        p.d3 := table(p.d3);

        p.d4 := orbit mod 10;
        p.d4 := table(p.d4);

        p.enter := 13;

        outport(p);
        delay(orbit*5);
        outtextxy(5,350,'**END BYTE ') ;
end;

procedure move_it51(data : integer; olddata : integer);
var orbit : integer;
    num   : integer;
    mov_e : integer;
begin
     ini_port(PT,$E7);    { com_ ,contr code }
```

```
        mov_e := data-olddata;
        orbit := round(Abs(mov_e*8.46));
        p.choose := 51;
        p.m := 77;
        if(data>olddata)then p.sign := 45;
        if(data<olddata)then p.sign := 43;

        p.d1 := trunc(orbit div 1000);
        p.d1 := table(p.d1);

        num  := trunc(orbit div 100);
        p.d2 := num mod 10;
        p.d2 := table(p.d2);

        num  := trunc(orbit div 10);
        p.d3 := num mod 10;
        p.d3 := table(p.d3);

        p.d4 := orbit mod 10;
        p.d4 := table(p.d4);

        p.enter := 13;

        outport(p);
        delay(orbit*5);
        outtextxy(5,350,'**END BYTE ') ;
end;

Procedure  gripper(grip : string);
var sign : byte;
begin
ini_port(PT,$E7);    { com_ ,contr code }

     send(PT,56);
     send(PT,77);
     if(grip = 'close') then sign := 43;
     if(grip = 'open')  then sign := 45;
     send(PT,sign);
     send(PT,50);
     send(PT,48);
     send(PT,48);
     send(PT,48);
     send(PT,13);
     delay(3500);

end;

Procedure rotate;
begin
      ini_port(PT,$E7);

      send(PT,55); {circle motor}
      send(PT,77);
      send(PT,45);
      send(PT,48);
      send(PT,51);
      send(PT,50);
      send(PT,48);
      send(PT,13);
      delay(2000);

end;

end.
```