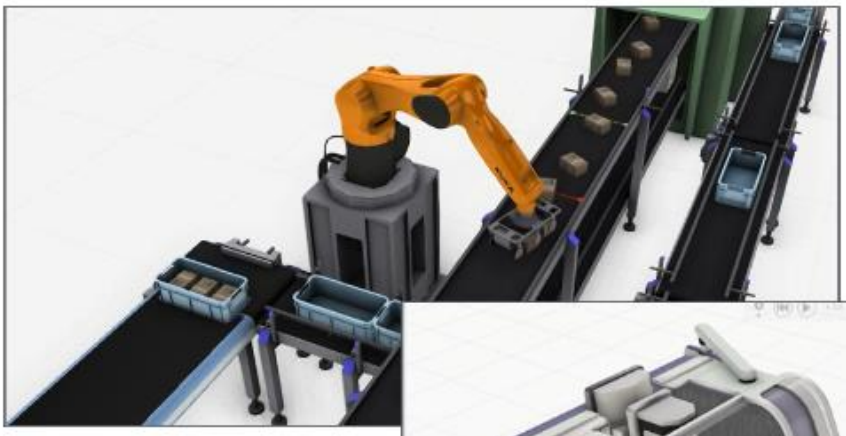
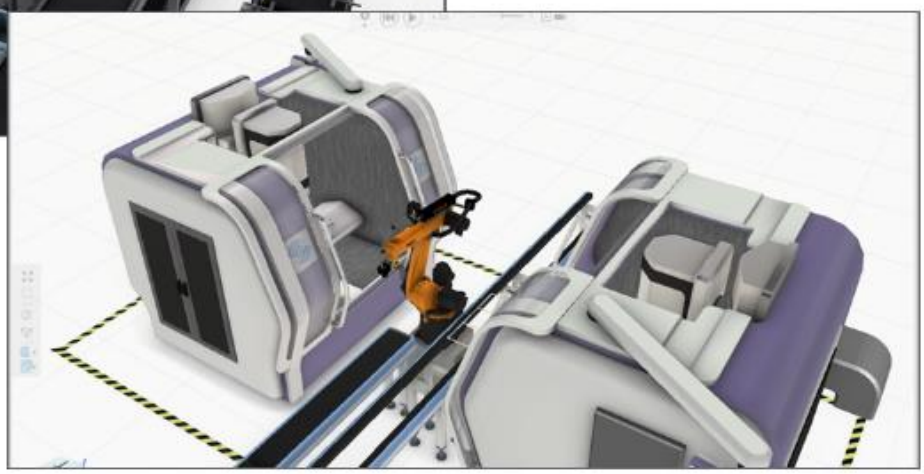
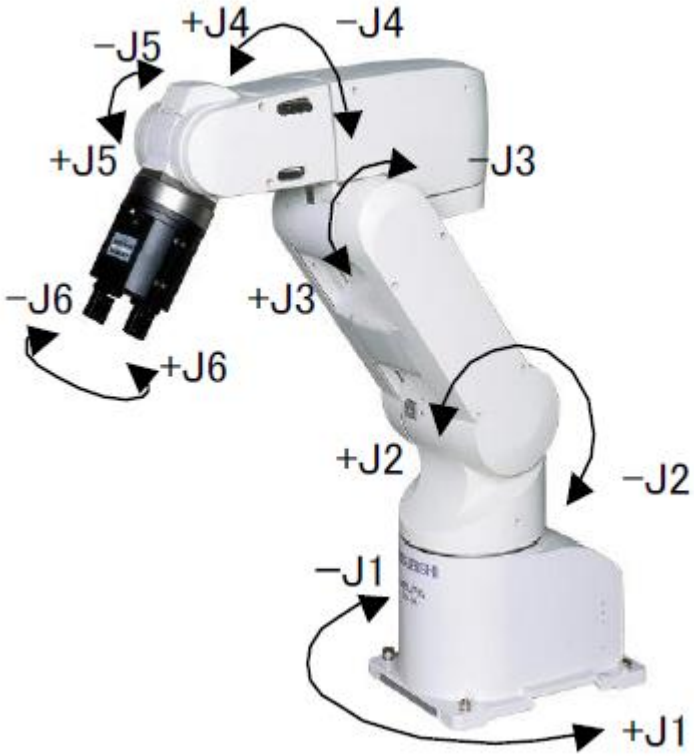
เทคโนโลยีหุ่นยนต์อุตสาหกรรม

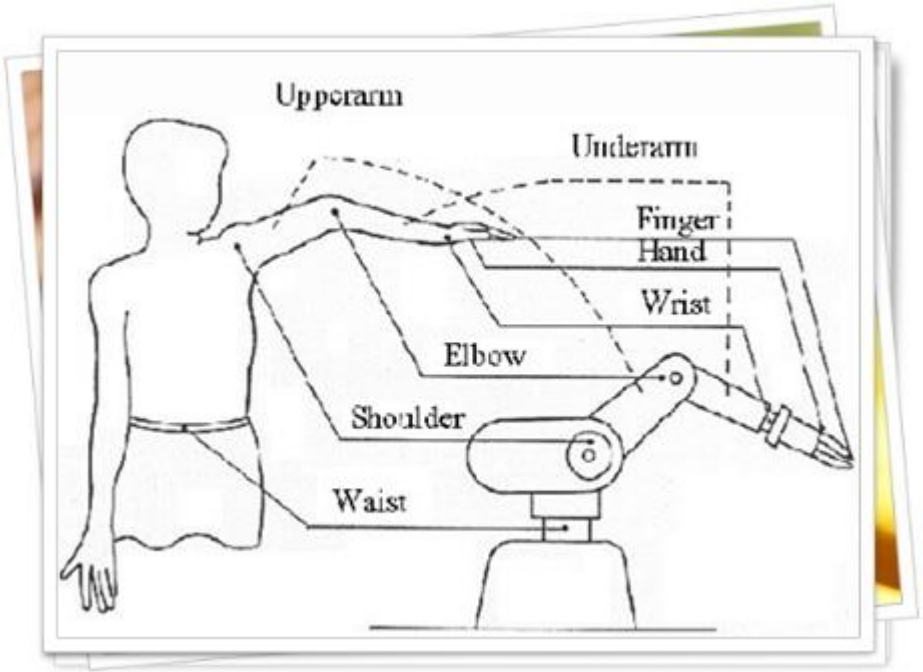






อุตสาหกรรมในประเทศไทยจะเห็นได้ว่ามีการนำเทคโนโลยีระบบอัตโนมัติ (Automation Technology) เข้ามาใช้งานเพื่อให้สินค้าสามารถแข่งขันในตลาดโลกได้ ทั้งในเรื่องราคา และคุณภาพ โดยเฉพาะในเรื่องคุณภาพ อาจเนื่องจากมีการเปลี่ยนรุ่นผลิตภัณฑ์อยู่บ่อยๆ ต้องใช้เวลาในการ Set Up ปัจจุบันจึงมีการนำเทคโนโลยีต่างๆ เข้ามาใช้ หนึ่งในเทคโนโลยีที่มีความยืดหยุ่นสูง ได้แก่ หุ่นยนต์อุตสาหกรรม เนื่องจากการเปลี่ยนการทำงานสามารถทำได้โดยการเปลี่ยนโปรแกรม นอกจากนี้คุณภาพของผลิตภัณฑ์ที่ได้มีความสม่ำเสมอเป็นมาตรฐานเดียวกัน

การทำงานของหุ่นยนต์อุตสาหกรรมจะเลียนแบบร่างกายของมนุษย์ โดยจะเลียนแบบเฉพาะส่วนของร่างกายที่จะนำไปใช้ประโยชน์ในอุตสาหกรรมเท่านั้น นั่นคือช่วงแขนของมนุษย์ ดังนั้น บางคนอาจจะได้ยินคำว่า “แขนกล” ซึ่งก็หมายถึงหุ่นยนต์อุตสาหกรรม การทำงานของหุ่นยนต์อุตสาหกรรมเปรียบเทียบกับแขนมนุษย์ แสดงดังรูป



ปัจจุบันและในอนาคตหุ่นยนต์อุตสาหกรรมจะเข้ามามีบทบาทในอุตสาหกรรมมากขึ้น โดยจะทำงานแทนมนุษย์ในงานต่างๆ เหล่านี้งานที่อันตราย เช่น งานยกเหล็กเข้าเตาหลอม งานที่เกี่ยวข้องกับสารเคมี งานซ้ำซากน่าเบื่อ เช่น งานยกสินค้าจากสายงานการผลิต งานประกอบ งานบรรจุผลิตภัณฑ์งานที่ต้องการคุณภาพมาตรฐานเดียวกัน เช่น งานเชื่อม งานตัด งานที่ต้องใช้ทักษะความชำนาญสูง เช่น งานเชื่อมแนว เชื่อมเลเซอร์ งานที่ต้องใช้ความละเอียดประณีต เช่น งานประกอบชิ้นส่วนอิเล็กทรอนิกส์ งานตรวจสอบ (Inspection) ฯลฯ

1.1 หุ่นยนต์คืออะไร

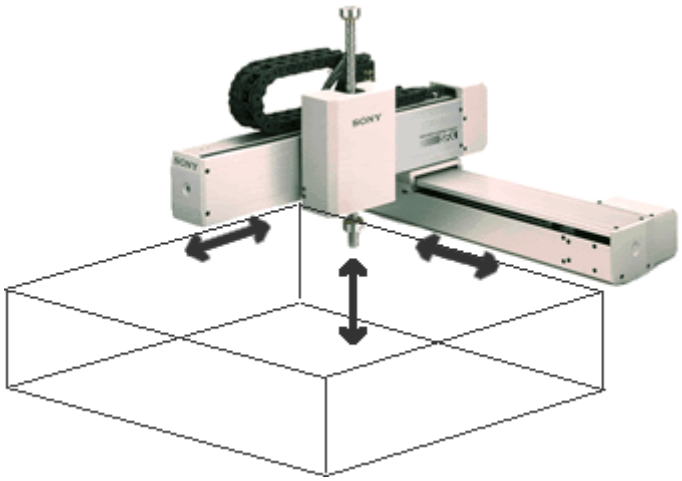
หุ่นยนต์ คือ เครื่องจักรที่ถูกควบคุมอัตโนมัติ สามารถเขียนโปรแกรมใหม่ได้ ใช้งานเอนกประสงค์ โปรแกรมการเคลื่อนที่จะต้องสามารถโปรแกรมให้เคลื่อนที่ได้อย่างน้อย 3 แกนหรือมากกว่า หุ่นยนต์อาจจะยึดอยู่กับที่หรือย้ายตำแหน่ง (Mobile) เพื่อใช้ในงานอุตสาหกรรม

1.2 การแบ่งชนิดของหุ่นยนต์

โดยทั่วไปการแบ่งชนิดของหุ่นยนต์จะแบ่งตามลักษณะรูปทรงของพื้นที่ทำงาน (Envelope Geometric) แต่ก่อนจะอธิบายชนิดของหุ่นยนต์ขออธิบายการทำงานของจุดต่อ (Joint) ของหุ่นยนต์อุตสาหกรรมซึ่งในขั้นพื้นฐานมี 2 ชนิดด้วยกัน ดังนี้จุดต่อ (Joint) ทั้งสองแบบเมื่อนำมาต่อเข้าด้วยกันอย่างน้อย 3 แกนหลักจะได้พื้นที่ทำงาน (Work envelope) ที่มีลักษณะแตกต่างกันไป ซึ่งสามารถนำมาแบ่งชนิดของหุ่นยนต์ได้ดังต่อไปนี้

1. Cartesian (Gantry) Robot

แกนทั้ง 3 ของหุ่นยนต์จะเคลื่อนที่เป็นแบบเชิงเส้น (Prismatic) ถ้าโครงสร้างมีลักษณะคล้าย Overhead Crane จะเรียกว่าเป็นหุ่นยนต์ชนิด Gantry แต่ถ้าหุ่นยนต์ไม่มีขาตั้งหรือขาเป็นแบบอื่น เรียกว่า ชนิด Cartesian



Cartesian Robot Work Envelop Of Cartesian Robot

ข้อดี

1. เคลื่อนที่เป็นแนวเส้นตรงทั้ง 3 มิติ
2. การเคลื่อนที่สามารถทำความเข้าใจง่าย

3. มีส่วนประกอบง่ายๆ
4. โครงสร้างแข็งแรงตลอดการเคลื่อนที่

ข้อเสีย

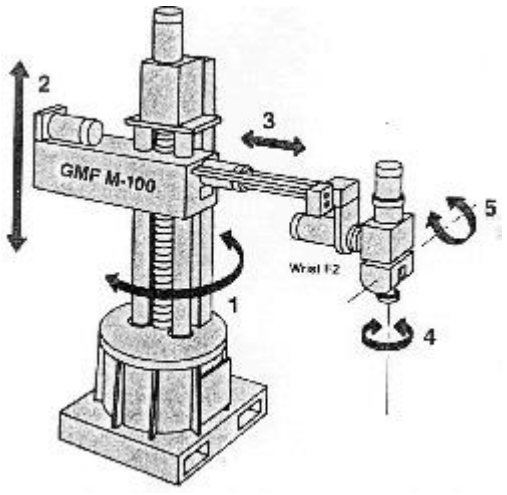
1. ต้องการพื้นที่ติดตั้งมาก
2. บริเวณที่หุ่นยนต์เข้าไปทำงานได้ จะเล็กกว่าขนาดของตัวหุ่นยนต์
3. ไม่สามารถเข้าถึงวัตถุจากทิศทางข้างใต้ได้
4. แกนแบบเชิงเส้นจะ Seal เพื่อป้องกันฝุ่นและของเหลวได้ยาก

การประยุกต์ใช้งาน

เนื่องจากโครงสร้างมีความแข็งแรงตลอดแนวการเคลื่อนที่ ดังนั้นจึงเหมาะกับงานเคลื่อนย้ายของหนักๆ หรือเรียกว่างาน Pick-and-Place เช่น ใช้โหลดชิ้นงานเข้าเครื่องจักร (Machine loading) ใช้จัดเก็บชิ้นงาน (Stacking) นอกจากนี้ยังสามารถใช้ในงานประกอบ (Assembly) ที่ไม่ต้องการเข้าถึงในลักษณะที่มีมุมหมุน เช่น ประกอบอุปกรณ์อิเล็กทรอนิกส์ และงาน Test ต่างๆ

2. Cylindrical Robot

หุ่นยนต์ประเภทนี้จะมีแกนที่ 2 (ไหล่) และแกนที่ 3 (ข้อศอก) เป็นแบบ Prismatic ส่วนแกนที่ 1 (เอว) จะเป็นแบบหมุน (Revolute) ทำให้การเคลื่อนที่ได้พื้นที่การทำงานเป็นรูปทรงกระบอก ดังรูป



Cylindrical Robot Work Envelop Of Cylindrical Robot

ข้อดี

1. มีส่วนประกอบไม่ซับซ้อน
2. การเคลื่อนที่สามารถเข้าใจได้ง่าย
3. สามารถเข้าถึงเครื่องจักรที่มีการเปิด – ปิด หรือเข้าไปในบริเวณที่เป็นช่องหรือโพรงได้ง่าย (Loading) เช่น การโหลดชิ้นงานเข้าเครื่อง CNC

ข้อเสีย

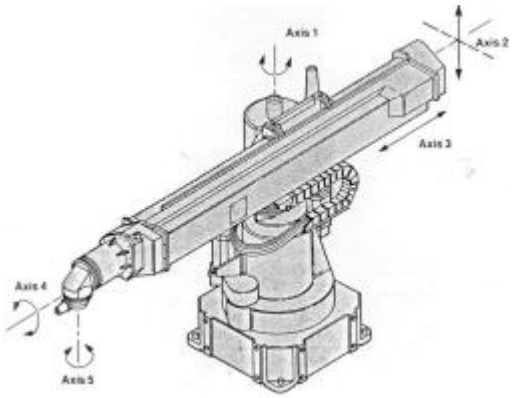
1. มีพื้นที่ทำงานจำกัด
2. แกนที่เป็นเชิงเส้นมีความยุ่งยากในการ Seal เพื่อป้องกันฝุ่นและของเหลว

การประยุกต์ใช้งาน

โดยทั่วไปจะใช้ในการหยิบยกชิ้นงาน (Pick-and-Place) หรือป้อนชิ้นงานเข้าเครื่องจักร เพราะสามารถเคลื่อนที่เข้าออกบริเวณที่เป็นช่องโพรงเล็กๆ ได้สะดวก

3. Spherical Robot (Polar)

มีสองแกนที่เคลื่อนในลักษณะการหมุน (Revolute Joint) คือแกนที่ 1 (เอว) และแกนที่ 2 (ไหล่) ส่วนแกนที่ 3 (ข้อศอก) จะเป็นลักษณะของการเคลื่อนที่แนวเส้นตรง ดังรูป



Spherical Robot Work Envelop Of Spherical Robot

ข้อดี

1. มีปริมาตรการทำงานมากขึ้นเนื่องจากการหมุนของแกนที่ 2 (ไหล่)
2. สามารถที่จะก้มลงมาจับชิ้นงานบนพื้นได้สะดวก

ข้อเสีย

1. มีระบบพิกัด (Coordinate) และส่วนประกอบ ที่ซับซ้อน
2. การเคลื่อนที่และระบบควบคุมมีความซับซ้อนขึ้น

การประยุกต์ใช้งาน

ใช้ในงานที่มีการเคลื่อนที่ในแนวตั้ง (Vertical) เพียงเล็กน้อย เช่น การโหลดชิ้นงานเข้าออกจากเครื่องปั้ม (Press) หรืออาจจะใช้งานเชื่อมจุด (Spot Welding)

4. SCARA Robot

หุ่นยนต์ SCARA (Selective Compliance Assembly Robot Arm) จะมีลักษณะแกนที่ 1 (เอว) และแกนที่ 3 (ข้อศอก) หมุนรอบแกนแนวตั้ง และแกนที่ 2 จะเป็นลักษณะการเคลื่อนที่ขึ้นลง (Prismatic) ดังรูป หุ่นยนต์ SCARA จะเคลื่อนที่ได้รวดเร็วในแนวระนาบ และมีความแม่นยำสูง

SCARA Robot Work Envelop Of SCARA Robot

ข้อดี

1. สามารถเคลื่อนที่ในแนวระนาบ และขึ้นลงได้รวดเร็ว
2. มีความแม่นยำสูง

ข้อเสีย

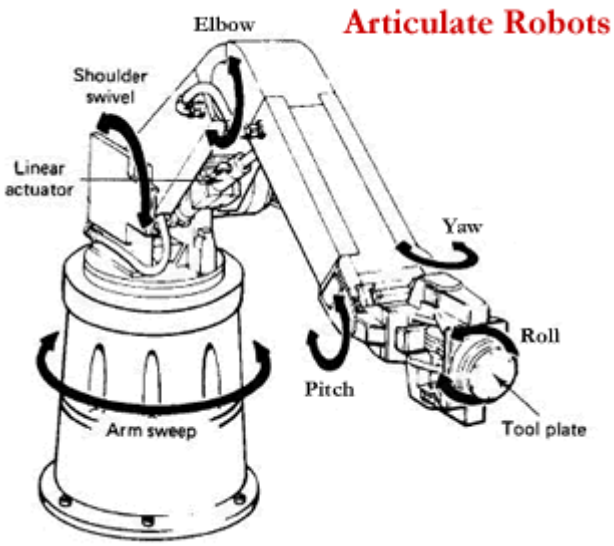
1. มีพื้นที่ทำงานจำกัด
2. ไม่สามารถหมุน (rotation)ในลักษณะมุมต่างๆได้
3. สามารถยกน้ำหนัก (Payload) ได้ไม่มากนัก

การประยุกต์ใช้งาน

เนื่องจากการเคลื่อนที่ในแนวระนาบและขึ้นลงได้รวดเร็วจึงเหมาะกับงานประกอบชิ้นส่วนทางอิเล็กทรอนิกส์ ซึ่งต้องการความรวดเร็วและการเคลื่อนที่ก็ไม่ต้องการการหมุนมากนัก แต่จะไม่เหมาะกับงานประกอบชิ้นส่วนทางกล (Mechanical Part) ซึ่งส่วนใหญ่การประกอบจะอาศัยการหมุน (Rotation)ในลักษณะมุมต่างๆ นอกจากนี้ SCARA Robot ยังเหมาะกับงานตรวจสอบ (Inspection) งานบรรจุภัณฑ์ (Packaging)

5. Articulated Arm (Revolute)

ทุกแกนการเคลื่อนที่จะเป็นแบบหมุน (Revolute) รูปแบบการเคลื่อนที่จะคล้ายกับแขนคน ซึ่งจะประกอบด้วยช่วงเอว ท่อนแขนบน ท่อนแขนล่าง ข้อมือ การเคลื่อนที่ทำให้ได้พื้นที่การทำงาน ดังรูป



Articulated Arm Robot Work Envelop Of Articulated Robot

ข้อดี

1. เนื่องจากทุกแกนจะเคลื่อนที่ในลักษณะ ของการหมุนทำให้มีความยืดหยุ่นสูงในการเข้าไปยังจุดต่างๆ
2. บริเวณข้อต่อ (Joint) สามารถ Seal เพื่อป้องกันฝุ่น ความชื้น หรือน้ำ ได้ง่าย
3. มีพื้นที่การทำงานมาก
4. สามารถเข้าถึงชิ้นงานทั้งจากด้านบน ด้านล่าง
5. เหมาะกับการใช้มอเตอร์ไฟฟ้า เป็นชุดขับเคลื่อน

ข้อเสีย

1. มีระบบพิกัด (Coordinate) ที่ซับซ้อน
2. การเคลื่อนที่และระบบควบคุมทำความ เข้าใจได้ยากขึ้น
3. ควบคุมให้เคลื่อนที่ในแนวเส้นตรง (Linear) ได้ยาก
4.โครงสร้างไม่มั่นคงตลอดช่วงการเคลื่อนที่ เพราะบริเวณขอบ Work Envelope ปลายแขนจะ
5. มีการสั่น ทำให้ความแม่ยำลดลง

การประยุกต์ใช้งาน

หุ่นยนต์ชนิดนี้สามารถใช้งานได้กว้างขวางเพราะสามารถเข้าถึงตำแหน่งต่างๆ ได้ดี เช่น งานเชื่อม Spot Welding, Path Welding, งานยกของ, งานตัด, งานทากาว, งานที่มีการเคลื่อนที่ยากๆ เช่น งานพ่นสี งาน Sealing ฯลฯ

การเลือกหุ่นยนต์ชนิดต่างๆ มาใช้งาน ควรพิจารณาให้เหมาะสมกับงานที่ต้องการให้หุ่นยนต์ทำดังที่อธิบายไว้ตอนต้น

2.0 **บทนำ**

ซอฟต์แวร์ COSIMIR เป็นซอฟต์แวร์จำลองการทำงานของระบบแบบสามมิติที่ทำงานบนพีซี สำหรับวินโดว์ 95, 98, NT หรือ 2000 ใช้สำหรับ**ออกแบบ** workcell **เขียนโปรแกรมควบคุมหุ่นยนต์** และ**จำลองการทำงาน**ของ workcell

การเคลื่อนไหวและหยิบจับชิ้นงานทุกขั้นตอนจะสามารถถูกจำลองเพื่อตรวจสอบและหลีกเลี่ยงการชนของหุ่นยนต์กับวัตถุรอบข้างและเพื่อลดเวลาในแต่ละรอบการทำงานให้น้อยที่สุด ซอฟต์แวร์ COSIMIR รองรับการออกแบบ workcell ด้วยไลบรารี่ที่มีเครื่องจักร หุ่นยนต์ เครื่องมือ สายพานลำเลียง อุปกรณ์ป้อนชิ้นงานและอื่นๆ นอกจากนี้ผู้ใช้ยังสามารถออกแบบชิ้นงานหรืออุปกรณ์อื่นๆ ในระบบ CAD แล้วนำเข้าไปในซอฟต์แวร์ COSIMIR ได้

2.1 COSIMIR family

- COSIMIR Industrial

  ความสามารถ:

  1. สามารถ**ออกแบบ** *หุ่นยนต์ใน workcell* **เขียนโปรแกรมควบคุม**และ**จำลองการทำงาน**ของหุ่นยนต์
  2. สามารถสื่อสารกับ คอนโทรลเลอร์ของหุ่นยนต์ Mitsubishi ได้

  ข้อจำกัด:

  1. จำลองการทำงานของหุ่นยนต์ของ MITSUBISHI เท่านั้น
  2. จำลองการทำงานของหุ่นยนต์ได้เพียงหนึ่งตัวเท่านั้น

- COSIMIR Professional

  ความสามารถ:

  1. สามารถ**ออกแบบ** *workcell ทั้งหมด* **เขียนโปรแกรมควบคุม**และ**จำลองการทำงาน** ของ workcell
  2. จำลองการทำงานของหุ่นยนต์มากกว่าหนึ่งตัวใน workcell เดียวกัน
  3. สามารถสื่อสารกับ คอนโทรลเลอร์ของหุ่นยนต์ Mitsubishi Kuka และ ABB ได้

- COSIMIR Educational

  เนื่องจากความซับซ้อนของระบบอัตโนมัติมากขึ้น ทำให้ COSIMIR Educational เข้ามามีบทบาททำให้ผู้เรียนเขียนโปรแกรมเพื่อจำลองการเคลื่อนที่ของหุ่นยนต์ก่อนที่จะไปใช้กับของจริง

  ความสามารถ:

  1. สามารถ**ออกแบบ** *หุ่นยนต์ใน workcell* แต่ไม่สามารถบันทึกได้ **เขียนโปรแกรมควบคุม**และ**จำลองการทำงานได้**
  2. สามารถดาวน์โหลดมาใช้ได้โดยไม่มีค่าใช้จ่ายที่ www.festo.com/didactic
     แล้วเลือกที่ Service > Software > Software Demos > COSIMIR Demo

  ข้อจำกัด:

  1. ไม่สามารถสื่อสารกับคอนโทรลเลอร์ของหุ่นยนต์ได้
  2. จำลองการทำงานของหุ่นยนต์ได้เพียงหนึ่งตัว

ในการติดตั้งซอฟต์แวร์ COSIMIR **ซอฟต์แวร์ต้องการคอมพิวเตอร์ที่มีสเปกอย่างต่ำดังต่อไปนี้**

หน่วยประมวลผล (Processor) : Pentium II 300 MHz Processor หรือสูงกว่า

หน่วยความจำ (Memory) : 128 MB RAM

พื้นที่ว่างในฮาร์ดดิสก์ : มากกว่า 200 MB

หน่วยปฏิบัติการ (Operating system): Window 95™/98™/NT™/2000™

Graphic Adapter : Adapter with 3D acceleration and OpenGL support

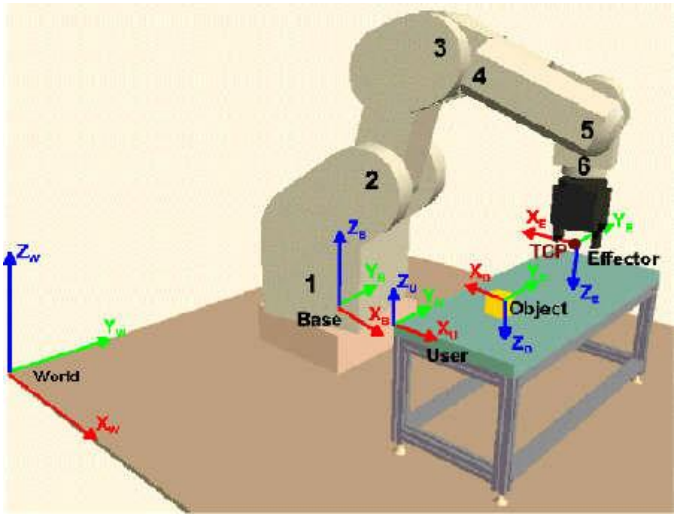
2.2 ระบบพิกัดจุด (Coordinate System)

2.2.1 ระบบพิกัดจุด

- Cartesian coordinate system

  คือ ระบบพิกัดจุดประกอบด้วยแกนที่ตั้งฉากกันและมีจุดตัดกันที่เรียกว่า จุดกำเนิด (origin) แกนสามแกนในระบบพิกัดจุดนี้ คือแกน X, Y และ Z ดังนั้นจุดใดๆ จะถูกระบุด้วยตำแหน่ง (X, Y, Z) คือ ระยะทางระหว่างจุดใดๆ กับจุดอ้างอิงของแกน X, Y และ Z

รูป 2.2.1: ระบบพิกัดจุด



ในวิชา robotics ระบบพิกัดจุดในระบบนี้ถูกใช้ดังต่อไปนี้ (ดูรูป 2.2.1 ประกอบ)

- World coordinate system ซึ่งโดยทั่วไปแล้วจะอยู่ที่มุมล่างสุดของพื้นที่ที่พิจารณา
- Base coordinate system ตั้งอยู่ที่ฐานของหุ่นยนต์
- Effector or tool coordinate systems ตั้งอยู่ที่จุดศูนย์กลางของกริปเปอร์ (Tool center point)
- Object coordinate systems ตั้งอยู่ที่ชิ้นงานแต่ละชิ้น
- User coordinate system ส่วนใหญ่ตั้งอยู่ที่วัตถุที่ไม่เคลื่อนไหวใน workcell

- Joint coordinate system (หรือ Robot coordinate system)

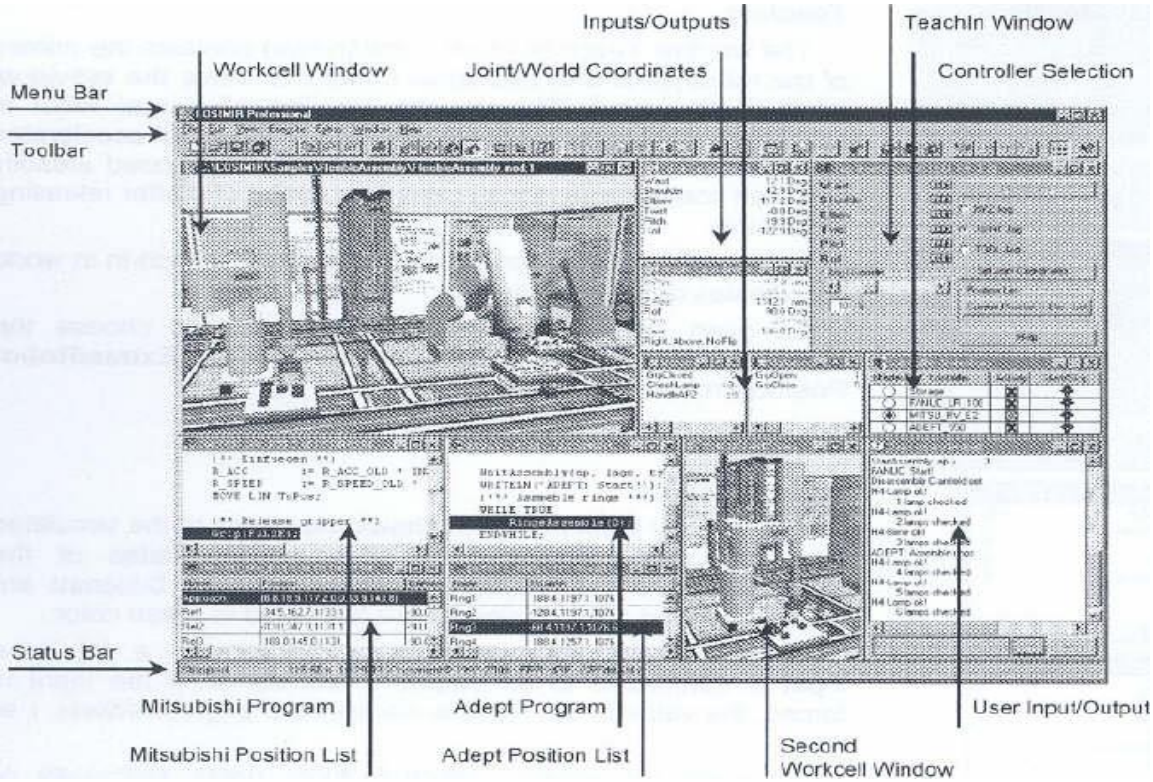
  ถูกนำมาใช้นอกเหนือจากระบบพิกัดฉาก Cartesian เป็นระบบพิกัดจุด (องศา) ที่เกิดจากตำแหน่งของข้อต่อทุกข้อต่อ จะระบุตำแหน่งของจุดโดยใช้ค่ามุมในการหมุนของข้อต่อทั้งหกข้อต่อ ซึ่งเป็นค่าที่หุ่นยนต์ต้องรักษาไว้เพื่อให้ไปถึงตำแหน่งที่กำหนดด้วยกริปเปอร์

- Tool coordinate system

  เป็นระบบพิกัดฉากที่อ้างอิงกับแกนที่หน้าแปลน (flange) ของหุ่นยนต์โดยในระบบพิกัดนี้ ผู้เรียนสามารถควบคุมทิศทางของกริปเปอร์ได้ตามทิศทางของแกน ZE (ดูรูป 2.2.1) ทำให้สะดวกในการกำหนดทางเดินของงาน เช่นในงาน pick and place เป็นต้น

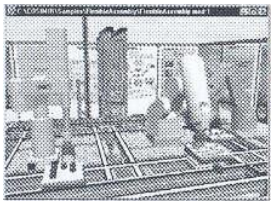
2.3 ส่วนประกอบของซอฟต์แวร์และคำสั่งต่าง ๆ

2.3.1 User Interface



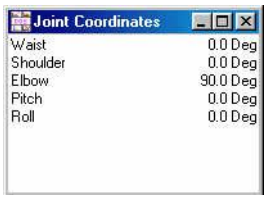
รูป 2.3.1: User interface

2.3.2 หน้าต่างย่อย



*Workcell window (**หน้าต่างแสดง** workcell)*

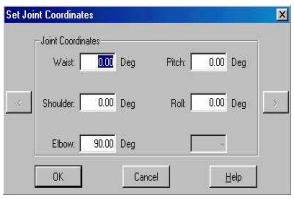
Workcell จะถูกแสดงในรูปกราฟฟิกในหน้าต่างนี้ ผู้ใช้สามารถเปิดหน้าต่าง workcell เพิ่มเติมได้อีกโดยการเลือกที่คำสั่ง View > New



Joint Coordinates

หน้าต่าง Joint Coordinates แสดงตำแหน่งของข้อต่อหุ่นยนต์แต่ละข้อต่อ หน่วยเป็นองศาสำหรับข้อต่อที่หมุนได้ และเป็นมิลลิเมตรสำหรับข้อต่อที่เคลื่อนที่เป็นเส้นตรง

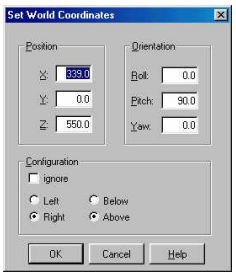
สามารถเปิดหน้าต่างนี้โดยเลือกคำสั่ง Extras > Robot Position > Show Joint Coordinates หรือ กด F7



เมื่อ ดับเบิลคลิ๊ก ลงบนหน้าต่างนี้จะปรากฏ dialog box “Set Joint Coordinates” ผู้ใช้สามารถระบุองศาในการหมุนของข้อต่อแล้วดูผลที่เกิดขึ้นกับหุ่นยนต์ได้

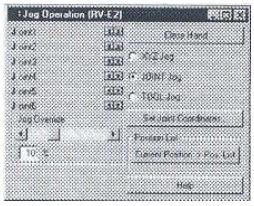
World Coordinates

หน้าต่างนี้แสดงตำแหน่งและทิศทางของ จุดศูนย์กลางของกริปเปอร์ (Tool Center Points) ใน world coordinates นอกจากนี้ ยังมีการระบุตำแหน่ง ทิศทาง และ configuration ของหุ่นยนต์ในบรรทัดล่างสุดของหน้าต่าง



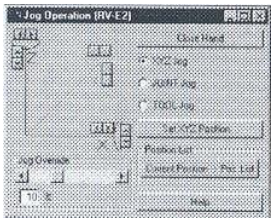
สามารถเปิดหน้าต่างนี้โดยเลือกคำสั่ง Extras > Robot Position > Show World Coordinates หรือ กด Shift+F7

เมื่อ ดับเบิลคลิ๊ก ลงบนหน้าต่างนี้จะปรากฏ dialog box “Set World Coordinates” ผู้ใช้สามารถระบุตำแหน่งในแกน X, Y และ Z และทิศทางของข้อต่อแล้วดูผลที่เกิดขึ้นกับหุ่นยนต์ได้



Teach-In

หน้าต่าง Teach-In ในโหมด Joint coordinates จะแสดงแถบเลื่อนของข้อต่อและมีปุ่มสองปุ่มในแต่ละข้อต่อเพื่อเคลื่อนที่ข้อต่อแต่ละข้อซึ่งเมื่อกดปุ่มเหล่านี้ หุ่นยนต์จะเคลื่อนที่ตามที่เรากำหนด ถ้าปุ่มถูกกดค้างเอาไว้ หุ่นยนต์จะเร่งความเร็วจนถึงระดับความเร็วที่ปรับเอาไว้ (override) แล้วรักษาความเร็วที่ระดับนั้นและหลังจากนั้นค่อยๆ ลดความเร็วลงจนเหลือศูนย์หลังจากที่ปล่อยปุ่ม



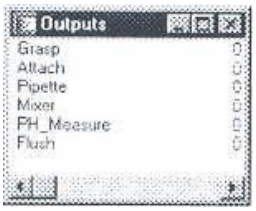
ผู้ใช้สามารถเปลี่ยนโหมดจาก Joint coordinate เป็น World coordinate หรือ Tool coordinates

สามารถเปิดหน้าต่างนี้โดยเลือกคำสั่ง Extras > Teach-In หรือกด F8

Inputs/Outputs

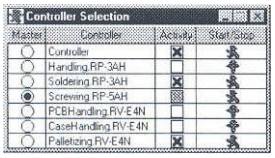
หน้าต่างนี้แสดงสถานะของการอินพุท/เอาท์พุทของคอนโทรลเลอร์

สถานะปัจจุบันของอินพุท/เอาท์พุทถูกแสดงถัดจากชื่อ สัญญาณ 0 ถูกแสดงด้วยสีแดง สัญญาณ 1 ถูกแสดงด้วยสีเขียว ค่าของอินพุทถูกแสดงในวงเล็บ [ ] เช่น [1] หมายถึงอินพุทถูกต่อกับเอาท์พุท ถ้าสัญญาณอินพุทถูกเปลี่ยนสถานะ ค่าของอินพุทจะถูกแสดงในวงเล็บ <>



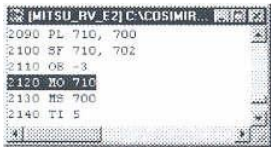
สามารถเปิดหน้าต่างนี้โดยเลือกคำสั่ง Extras > Inputs/Outputs > Show Inputs/Show Outputs หรือ กด F9 เพื่อเรียกหน้าต่างอินพุท กด shift+F9 เพื่อเรียกหน้าต่างเอาท์พุท

Controller Selection



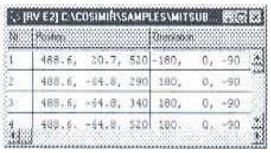
แสดงสถานะของคอนโทรลเลอร์ทุกตัวใน workcell ผู้ใช้สามารถเลือกคอนโทรลเลอร์หลักและสังเกตการทำงานของคอนโทรลเลอร์อื่น ตำแหน่งของหุ่นยนต์ อินพุท เอาท์พุท และ Teach-In ที่ปรากฏสำหรับคอนโทรลเลอร์หลัก
สามารถเปิดหน้าต่างนี้โดยเลือกคำสั่ง Execute > Controller Selection

Robot Program



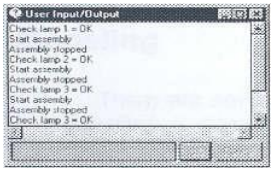
หน้าต่างนี้จะแสดงโปรแกรมสำหรับหุ่นยนต์ซึ่งจะปรากฏชื่อหุ่นยนต์บนหน้าต่าง
สามารถเปิดหน้าต่างนี้โดยเลือกคำสั่ง Menu > Open

Position List



หน้าต่างนี้จะแสดงรายการตำแหน่งต่างๆ ของหุ่นยนต์ หัวข้อบนหน้าต่างแสดงชื่อหุ่นยนต์เจ้าของตำแหน่งต่างๆ ในที่นี้ คือ RV-E2
สามารถเปิดหน้าต่างนี้โดยเลือกคำสั่ง Menu > Open

User Input/Output



หน้าต่างนี้จะปรากฏขึ้นโดยอัตโนมัติถ้าในโปรแกรมมีคำสั่งอ่านหรือส่งข้อมูลผ่าน Serial interface เนื่องจากเป็นการจำลอง ข้อมูลจะไม่ถูกส่งผ่านไปยังอุปกรณ์จริงผ่าน serial interface แต่จะถูกส่งไปยังหน้าต่างนี้เมื่อข้อมูลถูกแสดง

2.3.3 **คำสั่งบน** tool bar **ที่จำเป็น**

- Move 

คำสั่งนี้ใช้ในการ*เคลื่อนที่* มุมมองของวัตถุ (View and reference point) จะใช้ได้กับหน้าต่างที่กำลังทำงานอยู่ (active) ซึ่งมีมุมมองของวัตถุอยู่ เครื่องหมายจะแสดงแกนที่มุมมองสามารถเคลื่อนที่ตามแกนนั้นได้ ลูกศรที่เป็นเส้นประหมายถึง ไม่สามารถเคลื่อนที่ตามแกนนั้นได้

วิธีการเรียกคำสั่ง Move **และวิธีใช้งาน**

1. เรียกคำสั่ง Move ด้วยการเลือกที่ไอคอนบน tool bar หรือ View > Move หรือ คลิ๊กขวาที่วินโดวส์ของworkcellแล้วเลือกคำสั่ง Move
2. เมื่อเลือกคำสั่ง Move แล้ว เลื่อนเมาส์มาที่วินโดวส์workcellจะปรากฏสัญลักษณ์  หมายความว่า คำสั่ง Move พร้อมใช้งาน
3. คลิ๊กซ้ายเพื่อเคลื่อนที่แนวแกน X และ Z หรือคลิ๊กขวาเพื่อเคลื่อนที่แนวแกน X และ Y

- Rotate 

คำสั่งนี้ใช้ในการ*หมุน* มุมมองของวัตถุ จะใช้ได้กับหน้าต่างที่กำลังทำงานอยู่ ซึ่งมีมุมมองของวัตถุอยู่ เครื่องหมายจะแสดงแกนที่มุมมองสามารถหมุนตามแกนนั้นได้ ลูกศรที่เป็นเส้นประหมายถึง ไม่สามารถหมุนตามแกนนั้นได้

วิธีการเรียกคำสั่ง Rotate **และวิธีใช้งาน**

1. เรียกคำสั่ง Rotate ด้วยการเลือกที่ไอคอนบน tool bar หรือ View > Rotate หรือ คลิ๊กขวาที่วินโดวส์ของworkcellแล้วเลือกคำสั่ง Rotate
2. เมื่อเลือกคำสั่ง Rotate แล้ว เลื่อนเมาส์มาที่วินโดวส์ workcell จะปรากฏสัญลักษณ์  หมายความว่า คำสั่ง Rotate พร้อมใช้งาน
3. คลิ๊กซ้ายเพื่อเคลื่อนที่แนวแกน X และ Z หรือคลิ๊กขวาเพื่อเคลื่อนที่แนวแกน X และ Y

- Zoom 

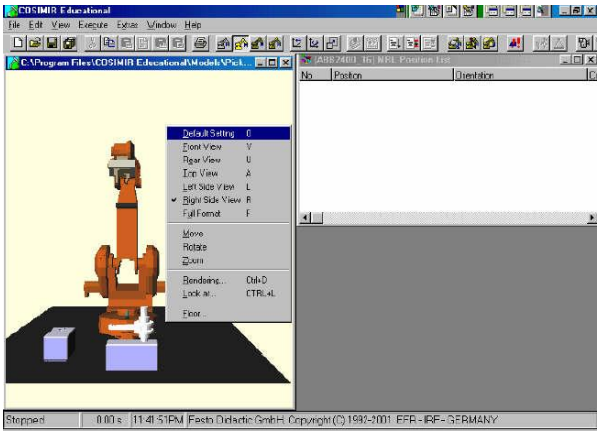
คำสั่งนี้ใช้ในการ*ขยายหรือลด* มุมมองของวัตถุ จะใช้ได้กับหน้าต่างที่กำลังทำงานอยู่ ซึ่งมีมุมมองของวัตถุอยู่

วิธีการเรียกคำสั่ง Zoom **และวิธีใช้งาน**

1. เรียกคำสั่ง Zoom ด้วยการเลือกที่ไอคอนบน tool bar หรือ View > Zoom หรือ คลิ๊กขวาที่วินโดวส์ของ workcell แล้วเลือกคำสั่ง Zoom
2. เมื่อเลือกคำสั่ง Zoom แล้ว เลื่อนเมาส์มาที่วินโดวส์ workcell จะปรากฏสัญลักษณ์  หมายความว่า คำสั่ง Zoom พร้อมใช้งาน

3. คลิ๊กซ้ายแล้วเลื่อนเมาส์ตามลูกศรในสัญลักษณ์เพื่อขยายมุมมองหรือคลิ๊กซ้ายแล้วเลื่อนเมาส์ในทิศทางตรงกันข้ามกับลูกศรในสัญลักษณ์เพื่อลดมุมมอง หรือถ้าต้องการขยายเฉพาะส่วน ให้นำเมาส์คลิ๊กในบริเวณที่ต้องการ หลังจากนั้นกด CTRL+Shift พร้อมกันแล้วคลิ๊กขวาพร้อมกับลากเมาส์จะปรากฏกรอบสีดำเพื่อเลือกบริเวณที่ต้องการขยาย หลังจากนั้นปล่อยเมาส์ บริเวณที่เลือกไว้จะถูกขยายขึ้น
นอกจากนี้สามารถคลิ๊กขวาที่รูปแล้วเลือกมุมมองเป็น Front view, Rear view, Top view, Left side view, และ Right side view ดังรูป 2.3.3a

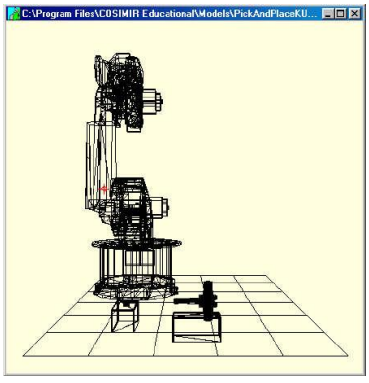
รูป 2.3.3a



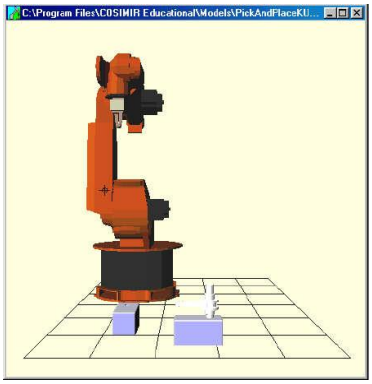
หมายเหตุ:
Full format: **ใช้คำสั่ง** full format **เพื่อดู** workcell **แบบเต็มหน้าต่างจากมุมมองปัจจุบัน**

Rendering: คำสั่ง rendering เปลี่ยนลักษณะภายนอกของวัตถุทุกชิ้นในหน้าต่างworkcell โดยลักษณะภายนอกมี 3 แบบ คือ

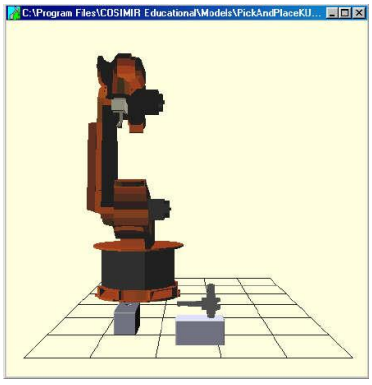
a) Wireframe:
b) Filled:
c) Flat-shaded:



รูป 2.3.3b: Wireframe



รูป 2.3.3c: Filled



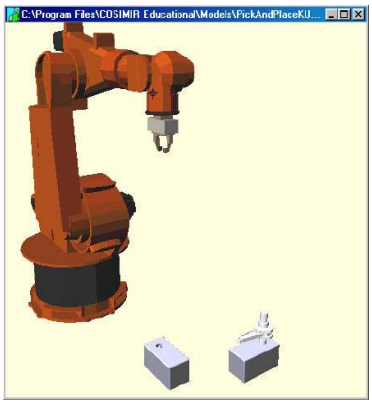
รูป 2.3.3d: Flat-shaded

Floor: คำสั่ง floor ทำหน้าที่เปลี่ยนลักษณะภายนอกของพื้นที่หุ่นยนต์ตั้งอยู่ โดยสามารถเปลี่ยนลักษณะ ขนาด และตำแหน่งของพื้นได้ สำหรับลักษณะของพื้นมี 3 ประเภท คือ

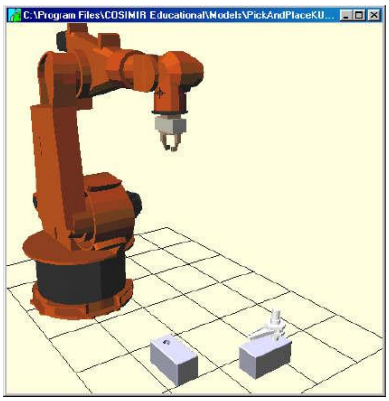
a) No Floor

b) Grid only

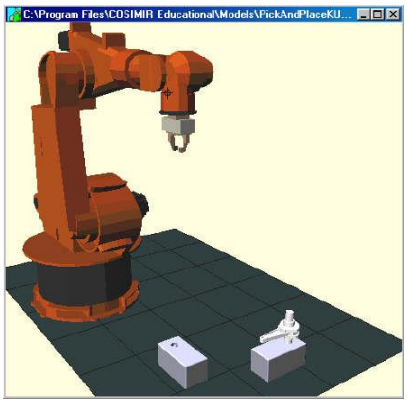
c) Filled Grid



รูป 2.3.3e: No floor



รูป 2.3.3f: Grid only



รูป 2.3.3g: Filled Grid

- Compile 

  ใช้คำสั่ง compile เพื่อให้ compiler แปลงโปรแกรมจากภาษาระดับสูง (High-level language) เช่น MRL และ Melfa Basic IV เป็นภาษาที่ระดับกลาง (Intermediate-level language)

  ผู้ใช้สามารถเรียกคำสั่ง Compile ได้ที่บน Tool bar และ Execute > Compile

- Compile + Link 

  ใช้คำสั่งนี้เพื่อให้ compiler แปลงโปรแกรมจากภาษาระดับสูงเป็นภาษาระดับกลาง และ linker จะแปลงโปรแกรมจากภาษาระดับกลางเป็น IRDATA code

  ผู้ใช้สามารถเรียกคำสั่ง Compile + Link ได้ที่บน Tool bar และ Execute > Compile + Link

- Start 

  ใช้คำสั่ง Start เพื่อเริ่มการอ่านโปรแกรมโดยหุ่นยนต์จะเคลื่อนที่ตามคำสั่งในโปรแกรม โดยจะอ่านโปรแกรม 1 รอบ

  ผู้ใช้สามารถเรียกคำสั่ง Start ได้ที่บน Tool bar และ Execute > Start

- Start Cycle

ใช้คำสั่ง Start cycle เพื่อเริ่มอ่านโปรแกรมโดยหุ่นยนต์จะเคลื่อนที่ตามคำสั่งในโปรแกรม โดยจะทำงานตามโปรแกรม เมื่อจบการอ่านโปรแกรม 1 รอบจะเริ่มเริ่มรอบใหม่โดยอัตโนมัติ จนกว่าผู้ใช้จะหยุดคำสั่ง Stop

ผู้ใช้สามารถเรียกคำสั่ง Start ได้ที่ Execute > Start Cycle

- Stop 

ใช้คำสั่ง Stop เพื่อหยุดการอ่านโปรแกรม คำสั่งนี้จะใช้ได้ก็ต่อเมื่อมีการอ่านโปรแกรมอยู่ ผู้ใช้สามารถเรียกคำสั่ง Stop ได้ที่บน Tool bar

- Next Step 

ใช้คำสั่ง Next Step เพื่ออ่านคำสั่งถัดไปในโปรแกรม โดยยกเว้นคำสั่งต่อไปนี้: Path smoothing, ตั้งค่าสัญญาณเอาท์พุท, อ่านค่าสัญญาณอินพุท, การเรียกโปรแกรมย่อย

2.3.4 การควบคุมหุ่นยนต์โดยใช้ Teach-In

- วิธีการเรียกคำสั่ง teach-in

ผู้ใช้สามารถเปิดหน้าต่าง Teach-In (ดูรูปในหัวข้อ 2.3.2 เรื่อง Teach-In ประกอบ) โดยเลือกคำสั่ง Extras > Teach-In หรือกด F8

- โหมดหรือรูปแบบในการกำหนดตำแหน่งในการเคลื่อนที่ให้หุ่นยนต์

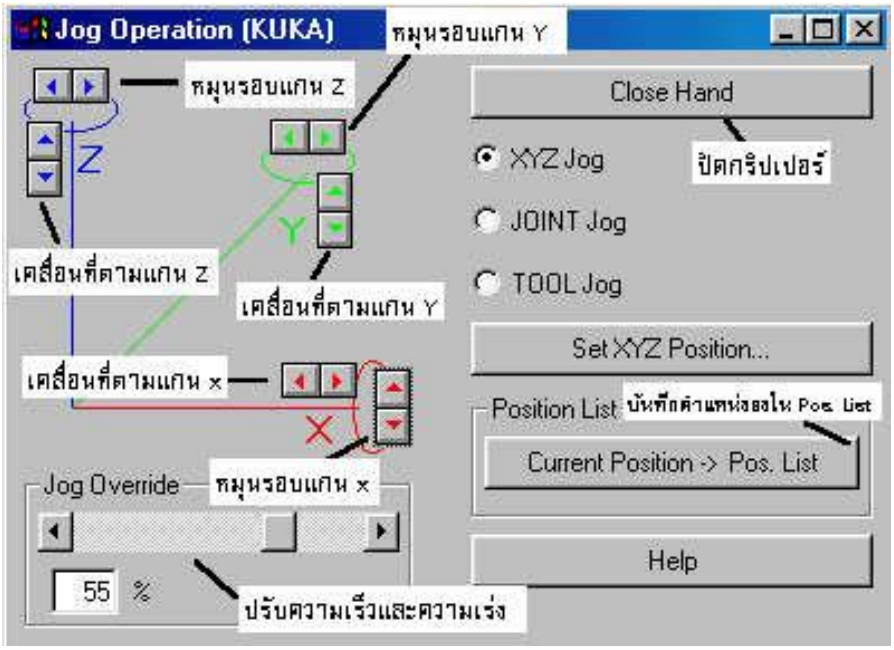
รูปแบบในการกำหนดตำแหน่งในการเคลื่อนที่มี 3 แบบ คือ แบบ XYZ Jog, แบบ Joint Jog และแบบ Tool Jog

- XYZ Jog

รูปแบบการกำหนดตำแหน่งในการเคลื่อนที่แบบ XYZ Jog อิงตามระบบพิกัดจุดแบบ Base coordinate system ซึ่งมีจุดกำเนิดตั้งอยู่ที่ฐานของหุ่นยนต์ (ดูรูป 2.2.1)

หน้าต่างของรูปแบบการกำหนดตำแหน่งในการเคลื่อนที่แบบ XYZ Jog จะปรากฏดังรูป 2.3.2a

รูป 2.3.4a: XYZ Jog



ปุ่มเคลื่อนที่ตามแกน X ทำหน้าที่ให้หุ่นยนต์เคลื่อนที่ในแกน X ทั้งทิศทางบวกและลบ

ปุ่มเคลื่อนที่ตามแกน Y ทำหน้าที่ให้หุ่นยนต์เคลื่อนที่ในแกน Y ทั้งทิศทางบวกและลบ

ปุ่มเคลื่อนที่ตามแกน Z ทำหน้าที่ให้หุ่นยนต์เคลื่อนที่ในแกน Z ทั้งทิศทางบวกและลบ

ปุ่มหมุนรอบแกน X ทำหน้าที่ให้หุ่นยนต์เคลื่อนที่หมุนรอบแกน X ทั้งทางบวกและลบ

ปุ่มหมุนรอบแกน Y ทำหน้าที่ให้หุ่นยนต์เคลื่อนที่หมุนรอบแกน Y ทั้งทางบวกและลบ

ปุ่มหมุนรอบแกน Z ทำหน้าที่ให้หุ่นยนต์เคลื่อนที่หมุนรอบแกน Z ทั้งทางบวกและลบ

เพื่อเคลื่อนที่จุดศูนย์กลางของกริปเปอร์ (TCP) ไปยังตำแหน่งที่ต้องการ

หมายเหตุ: ในการเคลื่อนที่ตามแกนหรือหมุนรอบแกนใดๆ นั้น หุ่นยนต์ต้องมีองศาอิสระ (degree of freedom) ในการเคลื่อนที่ด้วย ถ้าองศาอิสระไม่เพียงพอ จะปรากฏคำเตือนต่างๆ ขึ้นในระหว่างการเคลื่อนที่

ถ้าผู้ใช้กดปุ่มเหล่านี้ค้างไว้ หุ่นยนต์จะเร่งระดับความเร็วและความเร่งไปที่ระดับที่ตั้งไว้ และรักษาระดับความเร็วที่ระดับนั้นไว้และหลังจากนั้นจะลดระดับความเร็วลงจนเหลือศูนย์เมื่อปล่อยปุ่ม

ปุ่ม Jog Override ทำหน้าที่ปรับความเร็วและความเร่งในการเคลื่อนที่ โดยความเร็วและความเร่งจะแสดงเป็นตัวเลข (หน่วยเป็นเปอร์เซนต์) ที่ด้านล่างของปุ่ม

0% - ความเร็วและความเร่งต่ำ

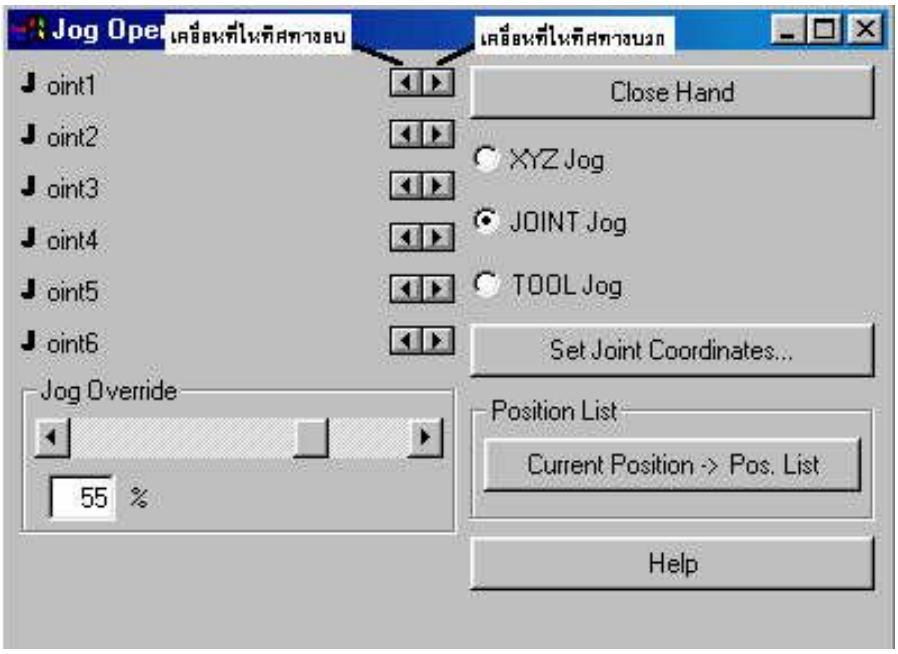
100% - ความเร็วและความเร่งสูง

ปุ่ม Close Hand ทำหน้าที่ปิดกริปเปอร์ เมื่อต้องการหยิบชิ้นงาน

ปุ่ม Current Position -> Pos. List ทำหน้าที่บันทึกตำแหน่งที่หุ่นยนต์อยู่ ณ ปัจจุบันลงใน Position list

รูปแบบการกำหนดตำแหน่งในการเคลื่อนที่ในลักษณะนี้เหมาะสมตำแหน่งที่หยิบจับและเข้าใกล้ชิ้นงาน

- Joint Jog



รูป 2.3.4b: Joint Jog

หน้าต่าง Teach-In ในโหมดนี้ได้บรรจุชื่อของข้อต่อแต่ละข้อต่อและมีปุ่มสองปุ่มในแต่ละข้อต่อเพื่อเคลื่อนที่ข้อต่อแต่ละข้อซึ่งปุ่มทางด้านซ้ายสำหรับเคลื่อนที่ในทิศทางลบ และด้านขวาสำหรับทิศทางบวก สำหรับปุ่มอื่นทำหน้าที่เหมือนดังรูป 2.3.4a เมื่อกดปุ่มเหล่านี้หุ่นยนต์จะเคลื่อนที่ตามที่กำหนด ถ้าปุ่มถูกกดค้างเอาไว้ หุ่นยนต์จะเร่งความเร็วจนถึงระดับความเร็วที่ปรับเอาไว้แล้วรักษาความเร็วที่ระดับนั้นและหลังจากนั้นค่อยๆ ลดความเร็วลงจนเหลือศูนย์หลังจากที่ปล่อยปุ่ม

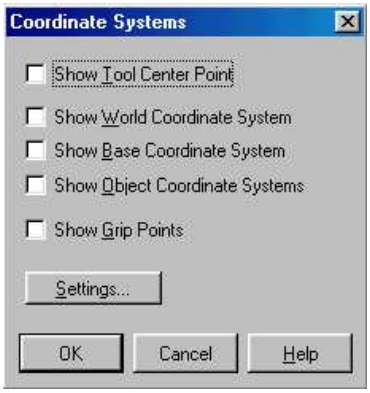
- Tool Jog

รูปแบบการกำหนดตำแหน่งในการเคลื่อนที่แบบ Tool Jog อิงตามระบบพิกัดจุดแบบ Tool coordinate system ซึ่งมีจุดกำเนิดตั้งอยู่ที่ศูนย์กลางของกริปเปอร์ของหุ่นยนต์ (ดูรูป 2.2.1 ประกอบ)

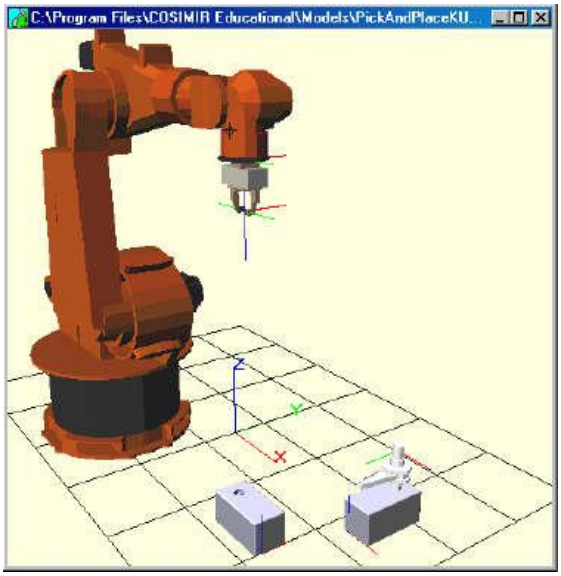
2.3.5 การแสดงระบบพิกัดจุด

ทำหน้าที่แสดงระบบพิกัดจุดต่างๆ บน workcell ผู้เรียนสามารถเรียก dialog box ของ ระบบพิกัดจุดให้ปรากฏโดยไปที่ Extras > Coordinate systems จะปรากฏ dialog box ดังรูป 2.3.5a ต่อจากนั้นทำการเลือกระบบพิกัดจุดที่ต้องการให้แสดง จะปรากฏระบบพิกัดจุดในworkcell ดังรูป 2.3.5b

หมายเหตุ: เมื่อเลือก Base coordinate system แล้วไม่เห็นระบบพิกัดจุดปรากฏ ให้ใช้คำสั่ง rendering เปลี่ยน ลักษณะภายนอก เป็น wireframe จะปรากฏ Base coordinate system อยู่ที่ฐานของหุ่นยนต์



รูป 2.3.5a: Coordinate system dialog box

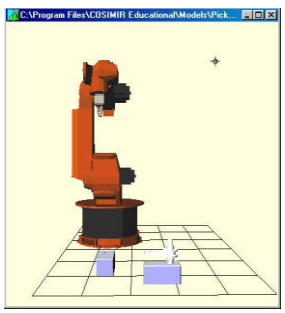


รูป 2.3.5b:

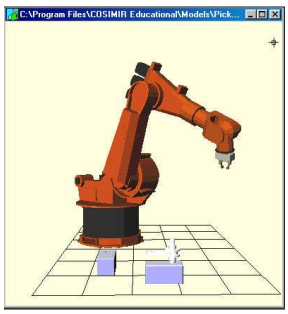
2.3.6 Reset workcell

คำสั่งนี้ใช้เพื่อกลับสู่สภาวะเริ่มต้นของ workcell โดยสภาวะของworkcellจะถูกรีเซ็ทกลับสู่สภาวะเดียวกับที่เพิ่งถูกโหลดจากไฟล์ สัญญาณเอาท์พุทจะถูกรีเซ็ทด้วยเช่นกัน โปรแกรมจะกลับสู่บรรทัดแรกและเวลาที่ใช้ในการรันโปรแกรมกลับเป็นศูนย์ ในกรณีที่ผู้เรียนทำการเปลี่ยนแปลงเลย์เอาท์ของ workcell เช่น เคลื่อนย้ายหรือหมุนวัตถุหรือเปลี่ยนสีของวัตถุ จะปรากฏ dialog box ถามว่าต้องการวัตถุอยู่ในสภาวะเดิม

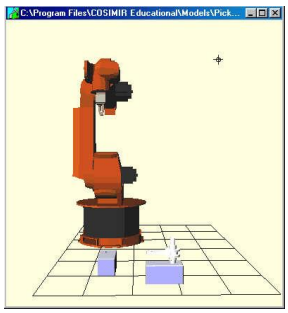
เรียกคำสั่ง Reset workcell ที่ Edit > Reset workcell



รูป 2.3.6a: สภาวะเริ่มต้น



รูป 2.3.6b:
ก่อนใช้คำสั่ง Reset workcell



รูป 2.3.6c:
หลังใช้คำสั่ง Reset workcell

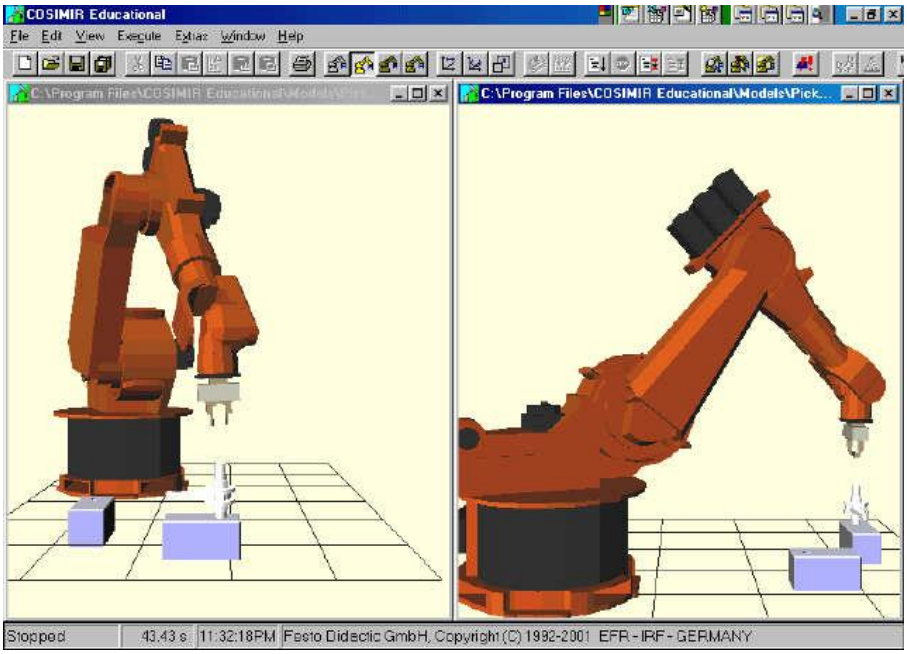
2.3.7 TCP tracking

ใช้คำสั่ง TCP tracking เพื่อแสดงเส้นทางการเคลื่อนที่ของจุดศูนย์กลางของกริปเปอร์ (TCP) โดยแสดงเป็นเส้นสีแดง วิธีการเรียกคำสั่ง TCP tracking คือ Extras > TCP tracking

2.3.8 New window

ใช้คำสั่ง View > New เพื่อเพิ่มหน้าต่าง workcell การใช้หน้าต่างหลายๆ หน้าต่างอย่างน้อย 2 หน้าต่าง คือ Right side view และ Front view โดยให้แต่ละหน้าต่างมีมุมมองที่ต่างกันไปมีประโยชน์มากในกรณีที่ต้องการควบคุมและกำหนดตำแหน่งในการเคลื่อนที่ในการจับชิ้นงาน

รูป 2.3.8a:



2.3.9 Help

นอกเหนือจากคำสั่งข้างต้นแล้วยังมีคำสั่งอื่นอีกมากมายที่ยังไม่ได้กล่าวถึงในหนังสือเล่มนี้ ดังนั้นถ้าผู้ที่สนใจคำสั่งอื่นๆ สามารถเปิดดูได้จาก Help ซึ่งจะมีคำอธิบาย ในหน้าต่าง Help (ดังรูป 2.3.9a) จะประกอบไปด้วยแท็บ 4 แท็บ คือ Contents, Index, Search และ Favorites

โดยที่แท็บ Content บรรจุเนื้อหาโดยเรียงตามหัวเรื่อง

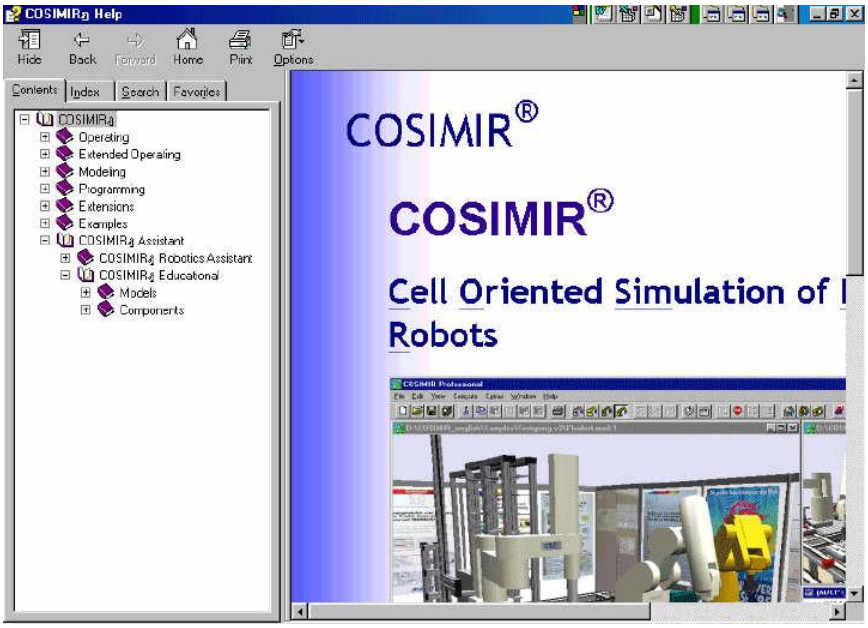
แท็บ Index บรรจุเนื้อหาโดยเรียงตามลำดับตัวอักษร

แท็บ Search สำหรับค้นหาคำสั่งที่ต้องการ

แท็บ Favorites สำหรับเก็บเนื้อหาที่ใช้บ่อยๆ

หมายเหตุ: Help จะปรากฏขึ้นพร้อมกับโปรแกรมสำหรับกรณีของ Cosimir educational

รูป 2.3.9a:



ตัวอย่างที่ 2.1: สาธิตวิธีการกำหนดตำแหน่ง

หลังจากที่รู้จักส่วนประกอบและคำสั่งในซอฟต์แวร์พอสมควรแล้ว ตัวอย่างนี้มีวัตถุประสงค์ให้ผู้อ่านทำความคุ้นเคยกับซอฟต์แวร์ ด้วยการกำหนดตำแหน่งในการเคลื่อนที่ทั้งหมด 3 จุด: จุดที่ 1 คือ จุดสภาวะเริ่มต้น จุดที่ 2 คือ จุดที่อยู่เหนือชิ้นงานและจุดที่ 3 คือ จุดที่หยิบชิ้นงาน โดยใช้โมเดล 'PickAndPlaceKuka'

ขั้นตอนที่ 1: เปิด โมเดล 'PickAndPlaceKuka'

การเปิดโมเดลทำได้โดยดูจากบทที่ 3 หัวข้อที่ 3.1 หน้าที่ 3-1 ถึง 3-10

ขั้นตอนที่ 2: เพิ่มหน้าต่างโมเดลโดยใช้คำสั่ง View > New

ขั้นตอนที่ 3: เปิดหน้าต่างสำหรับบันทึกตำแหน่ง (Position List)

การเปิดหน้าต่างทำได้โดยดูจากบทที่ 3 หัวข้อที่ 3.2 หน้าที่ 3-10 ถึง 3-13

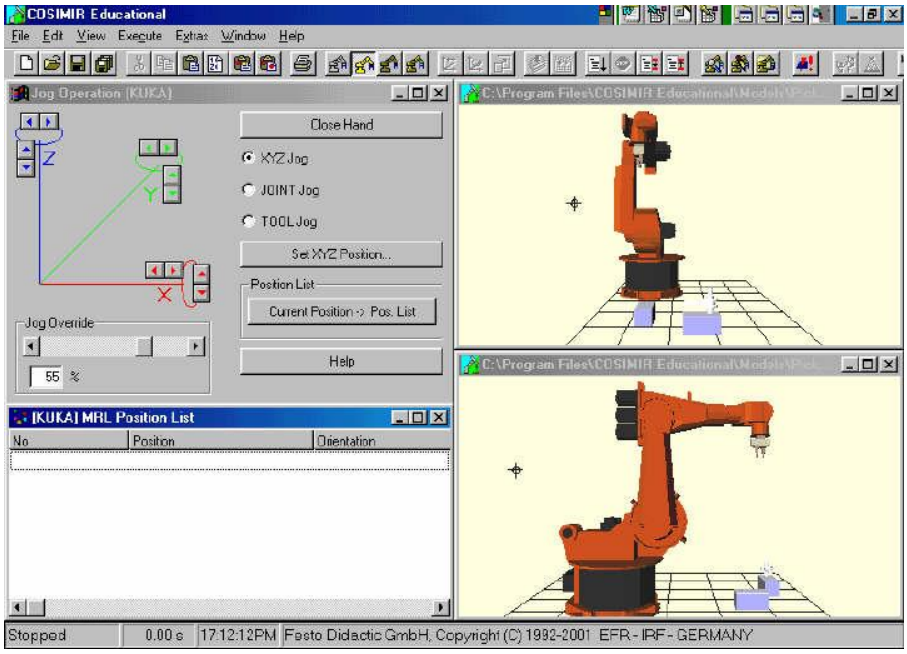
ขั้นตอนที่ 4: กด F8 เพื่อเปิดหน้าต่าง Teach-In

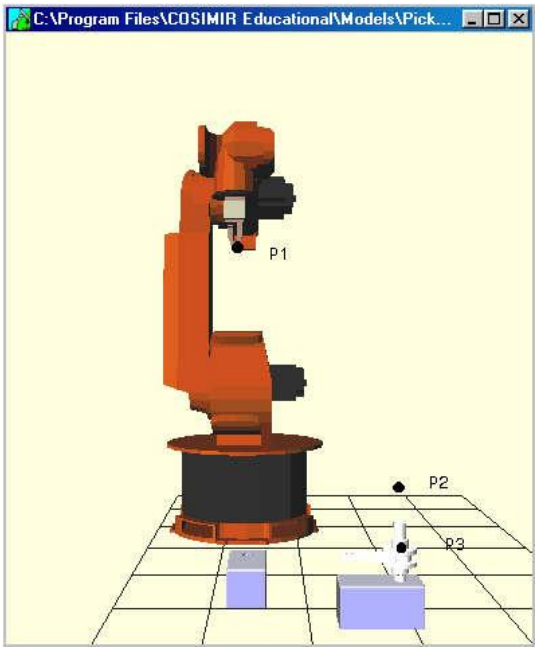
เมื่อปฏิบัติตามขั้นตอนที่ 1 ถึง 4 จะปรากฏหน้าต่างดังรูป 2.4.1a (หมายเหตุ: อาจมีหน้าต่างอื่นที่ไม่เหมือนกับสี่หน้าต่างนี้ ให้ปิดหน้าต่างเหล่านั้นไป)

ขั้นตอนที่ 5: กำหนดตำแหน่ง

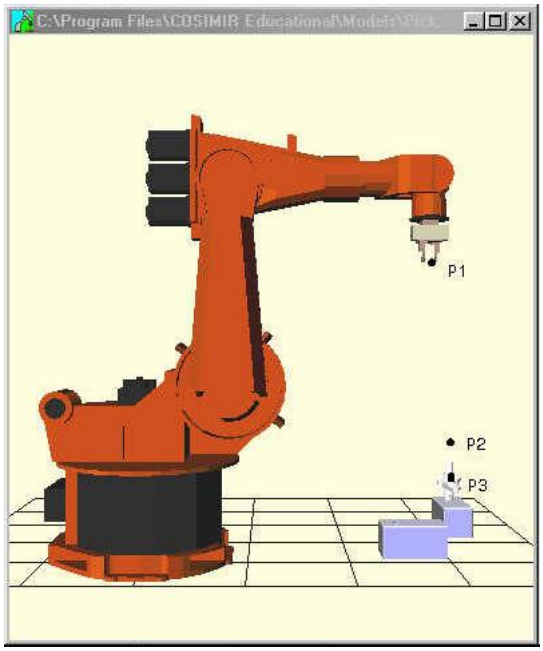
ก่อนการกำหนดตำแหน่งอาจวางแผนเกี่ยวกับตำแหน่งคร่าวๆ ในกระดาษดังเช่นรูป 2.4.1b และ c

รูป 2.4.1a:





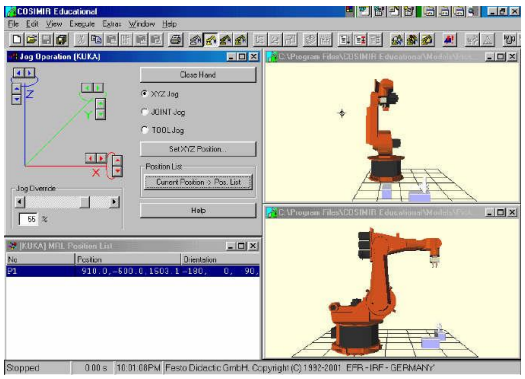
รูป 2.4.1b:



รูป 2.4.1c:

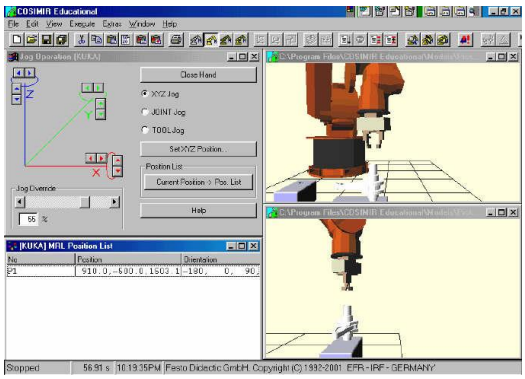
จุดที่ 1 คือ สภาวะเริ่มต้น ดังนั้นจุดใดๆ สามารถเป็นสภาวะเริ่มต้นได้ ตำแหน่งเริ่มต้นที่ได้มาจากการโหลดจากไฟล์สามารถเป็นตำแหน่งเริ่มต้นได้ และสามารถบันทึกตำแหน่งลงใน Position Lists ได้ทันที

รูป 2.4.1d:



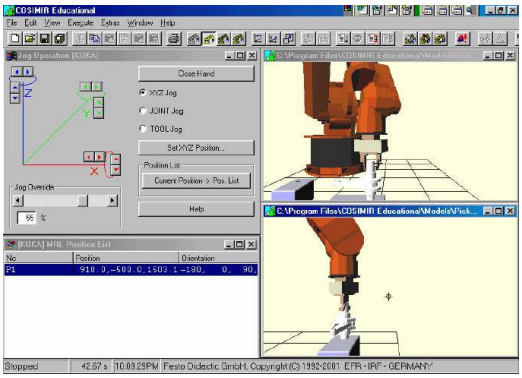
จุดที่ 2 คือ จุดที่กริปเปอร์อยู่เหนือชิ้นงาน จากจุดที่ 1 เคลื่อนที่ไปจุดที่ 2 ด้วยการควบคุมผ่านหน้าต่าง Teach-In โดยใช้รูปแบบการเคลื่อนที่แบบ Joint jog และ XYZ jog จนกระทั่งอยู่ในระยะที่อยู่เหนือชิ้นงานและกริปเปอร์ตั้งฉากกับชิ้นงานทั้งสองมุมมอง
หมายเหตุ: ยังไม่บันทึกจุดที่ 2 เนื่องจากเป็นการเห็นด้วยสายตาว่าตั้งฉากและยังไม่แน่นอนว่าจากจุดที่ 2 ไปจุดที่ 3 สามารถจับชิ้นงานได้หรือไม่

รูป 2.4.1e:

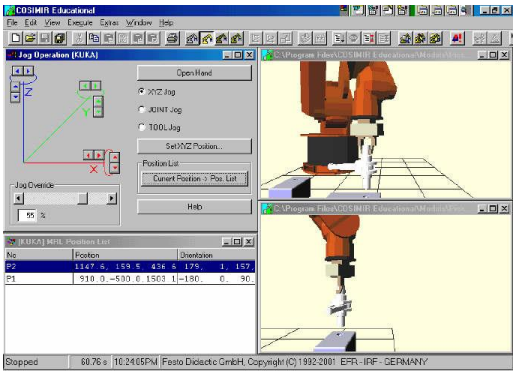


จุดที่ 3: จุดที่หยิบชิ้นงาน เคลื่อนที่จากจุดที่ 2 ไปจุดที่ 3 โดยใช้รูปแบบการเคลื่อนที่แบบ XYZ jog เริ่มเคลื่อนที่ในแนวแกน Z เพื่อทดสอบว่ากริปเปอร์สามารถจับชิ้นงานได้พอดีหรือไม่ ถ้าไม่พอดีสามารถขยับในแกน Y และ X ในระหว่างนี้สามารถซูมมาที่ชิ้นงานได้ เมื่อมั่นใจว่าสามารถจับชิ้นงานได้ให้กดปุ่ม Close Hand แล้วเคลื่อนที่ขึ้นในแนวแกน Z

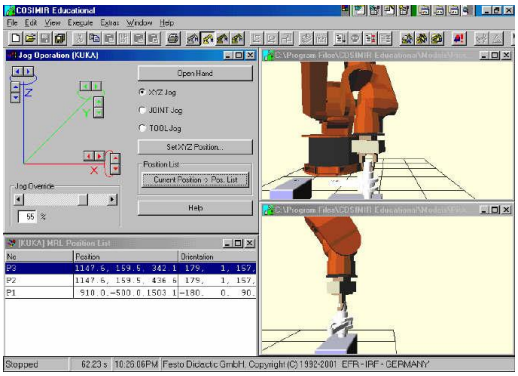
หลังจากนั้นบันทึกเป็นตำแหน่งที่ 2 เคลื่อนที่ลงนำชิ้นงานวางไว้ที่เดิม บันทึกเป็นตำแหน่งที่ 3



รูป 2.4.1f:



รูป 2.4.1g:

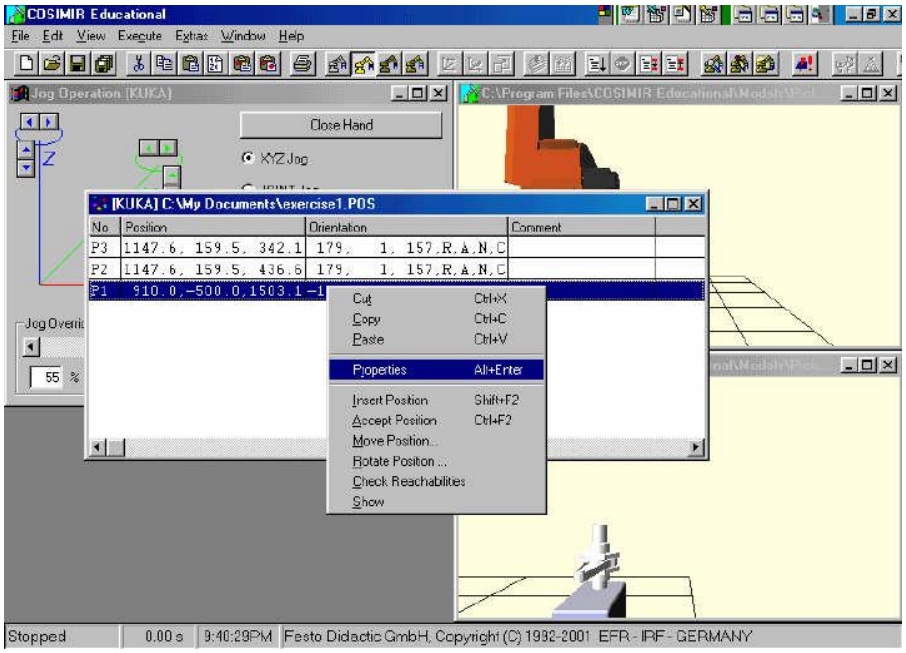


รูป 2.4.1h:

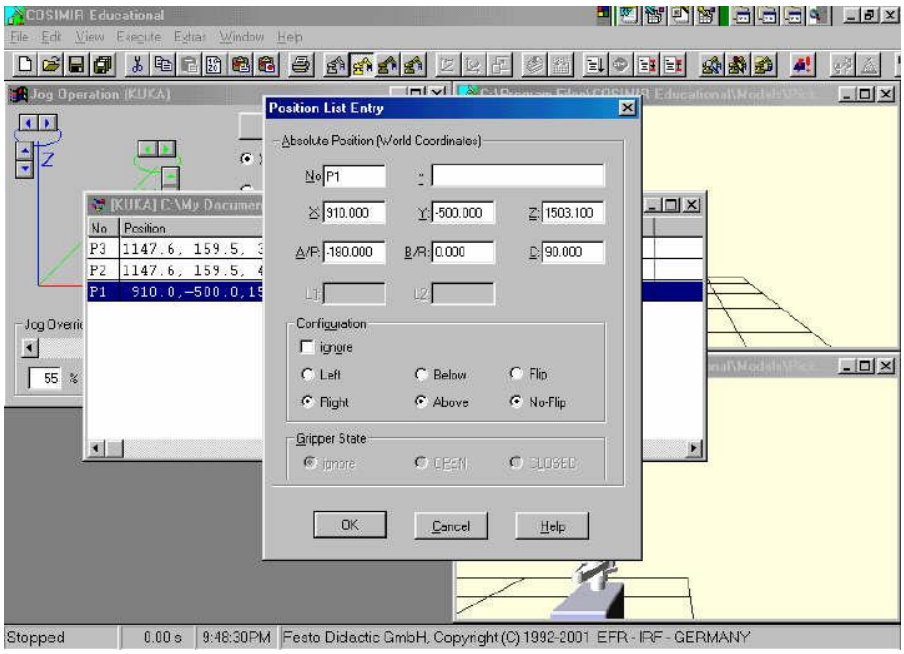
ขั้นตอนที่ 6: เมื่อได้ทั้งสามตำแหน่งที่ต้องการให้บันทึก Position List ใน C:/My Documents ชื่อ example 2_1.POS

หมายเหตุ: นอกจากจะบันทึกตำแหน่งแล้ว ยังสามารถใส่คำอธิบาย (Comment) ลงไปในตำแหน่งต่างโดยเลือกที่ตำแหน่งที่ต้องการใส่คำอธิบายแล้วคลิ๊กขวา เลือก Properties ดังรูป 2.4.1i หลังจากนั้นจะปรากฏหน้าต่าง dialog box ดังรูป 2.4.1j เพื่อใส่คำอธิบาย เมื่อใส่คำอธิบายเสร็จเรียบร้อยแล้วจะหน้าต่าง Position Lists จะปรากฏข้อความคำอธิบายดังรูป 2.4.1k

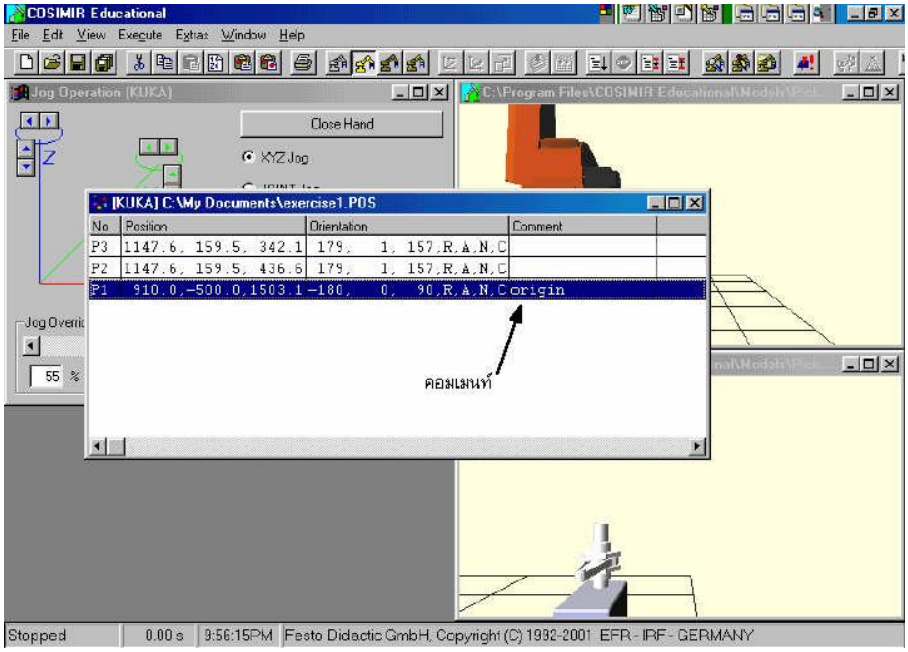
รูป 2.4.1i:



รูป 2.4.1j:

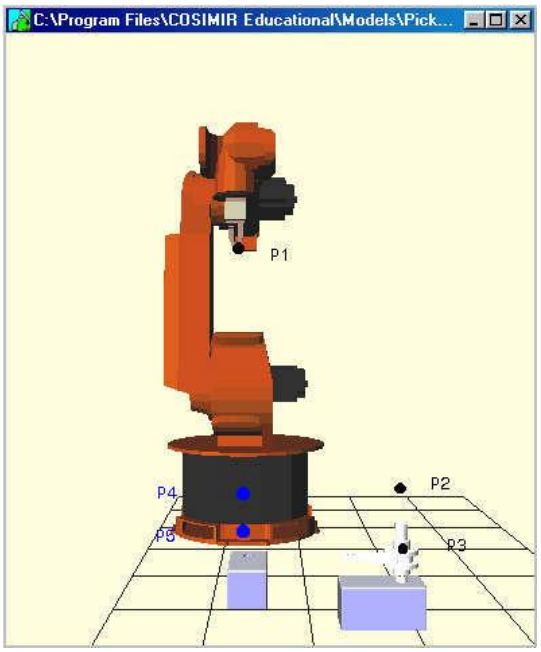


รูป 2.4.1k:

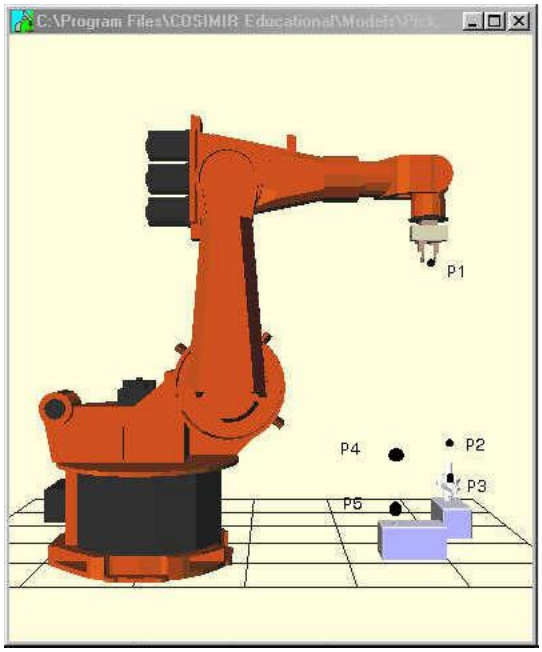


แบบฝึกหัดที่ 2.1

จงกำหนดตำแหน่งเพิ่มอีก 2 ตำแหน่ง คือ P4 และ P5 ดังรูป 2.5.1a และ b แล้วบันทึกใน C:/My Documents ชื่อ exercise 2_1.POS



รูป 2.5.1a:



รูป 2.5.1b:

2.4 คำสั่งต่าง ๆ ในภาษา Movemaster Command (MRL)

คำสั่งถูกแบ่งออกเป็น 7 หมวด โดยผู้เรียนต้องเลือกคำสั่งให้ถูกกับงาน

a) หมวดการควบคุมตำแหน่งและการเคลื่อนที่ (Position and motion control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับตำแหน่งและรวมไปถึง การกำหนดความเร็วในการเคลื่อนที่ การหน่วงเวลา รูปแบบในการเคลื่อนที่, tool, palette, เป็นต้น

b) หมวดการควบคุมโปรแกรม (Program control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับ การเลือกทำแบบมีเงื่อนไข (conditional branching) การทำซ้ำ (repetitive operation) การขัดจังหวะสัญญาณ (interrupting of signals) และ เคาน์เตอร์ (counter) เป็นต้น

c) หมวดการควบคุมกริปเปอร์ (Hand control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับการปิด/เปิดของกริปเปอร์

d) หมวดการควบคุมอินพุท/เอาท์พุท (I/O control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับการควบคุมอินพุท/เอาท์พุทจากสัญญาณภายนอก

e) หมวดการคำนวณทางคณิตศาสตร์(Operation, substitute, exchange command)
คำสั่งในหมวดนี้เกี่ยวข้องกับการคำนวณทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร เป็นต้น

f) หมวดการติดต่อสื่อสารผ่าน RS232C (Communication command: using RS232C)
คำสั่งในหมวดนี้เกี่ยวข้องกับการถ่ายโอนข้อมูลภายในหุ่นยนต์ เช่น เคาน์เตอร์ ตำแหน่ง รายชื่อโปรแกรม สถานะของสัญญาณอินพุท/เอาท์พุท และตัวแปรอื่นๆ ไปยังคอมพิวเตอร์

g) คำสั่งอื่นๆ
คำสั่งในหมวดนี้เกี่ยวข้องกับการตั้งค่าตัวแปร การเลือกโปรแกรม และการใส่คำอธิบายในโปรแกรม

หมายเหตุ: ตัวแปรใน <> และ [] มีความหมายดังต่อไปนี้

< > หมายถึง ตัวแปรของคำสั่ง

[ ] หมายถึง ตัวแปรสามารถละเว้นได้ (จะใส่หรือไม่ใส่ก็ได้)

หมวดการควบคุมตำแหน่งและการเคลื่อนที่

คำสั่งที่อยู่ในหมวดนี้ที่ใช้บ่อยๆ ได้แก่

MO	Move
MS	Move Straight
OVR	Override
SP	Speed
TI	Timer

หมายเหตุ: คำสั่งทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Movemaster Command → Position and control command

คำสั่ง: MO (Move)

การทำงาน: เคลื่อนปลายสุดของกริปเปอร์ไปยังตำแหน่งที่ต้องการโดยเคลื่อนแบบคำนวณตามแกน

รูปแบบ: MO<position number>[,[<O/C>]]

คำศัพท์: *<position number>*

ระบุหมายเลขตำแหน่งปลายทางเป็นจำนวนเต็มตั้งแต่ 1 ถึง 999

<O/C>

ระบุสถานะปิด/เปิดของกริปเปอร์ ถ้าไม่ได้ระบุ สถานะของกริปเปอร์จะปรากฏ

O: กริปเปอร์เปิด

C: กริปเปอร์ปิด

คำอธิบาย:

(1) เคลื่อนที่ปลายสุดของกริปเปอร์ไปยังตำแหน่งที่ระบุไว้โดยการเคลื่อนที่แบบ การเคลื่อนที่แบบคำนวณตามแกน

(2) ถ้ามีการระบุสถานะของกริปเปอร์ หุ่นยนต์จะเคลื่อนที่หลังจากที่ปิด/เปิดกริปเปอร์แล้ว

(3) ถ้าไม่ได้ระบุสถานะของกริปเปอร์ หุ่นยนต์จะเคลื่อนที่ทันที

(4) จะมีเสียงร้องเตือนถ้าตำแหน่งที่ต้องการให้หุ่นยนต์เคลื่อนที่ไปไม่ได้กำหนดไว้ หรือ ตำแหน่งที่จะไปเกินขอบเขตการทำงานของหุ่นยนต์

ตัวอย่าง:

10 SP 10	'ตั้งค่าความเร็วที่ 10'
20 MO 20,C	'เคลื่อนที่ไปตำแหน่งที่ 20 กริปเปอร์ปิด'
30 MO 30, O	'เคลื่อนที่ไปตำแหน่งที่ 30 กริปเปอร์เปิด'

****

คำสั่ง: MS (Move Straight)

การทำงาน: เคลื่อนปลายสุดของกริปเปอร์ไปยังตำแหน่งที่ต้องการโดยเคลื่อนที่แบบเส้นตรง

รูปแบบ: MS <position number>[,[<O/C>]]

คำศัพท์: *<position number>*

ระบุหมายเลขตำแหน่งปลายทางเป็นจำนวนเต็มตั้งแต่ 1 ถึง 999

<O/C>

ระบุสถานะปิด/เปิดของกริปเปอร์ ถ้าไม่ได้ระบุ สถานะของกริปเปอร์จะปรากฏ

O: กริปเปอร์เปิด

C: กริปเปอร์ปิด

คำอธิบาย:

(1) เคลื่อนที่ปลายสุดของกริปเปอร์ไปยังตำแหน่งที่ระบุไว้โดยการเคลื่อนที่แบบ การเคลื่อนที่แบบเส้นตรง

(2) จะมีเสียงร้องเตือนถ้าตำแหน่งที่ต้องการให้หุ่นยนต์เคลื่อนที่ไปไม่ได้กำหนดไว้ หรือ ตำแหน่งที่จะไปเกินขอบเขตการทำงานของหุ่นยนต์

(3) ถ้ามีการระบุสถานะของกริปเปอร์ หุ่นยนต์จะเคลื่อนที่หลังจากที่ปิด/เปิดกริปเปอร์แล้ว

(4) ความเร็วในการเคลื่อนที่ถูกกำหนดโดยคำสั่ง SP หรือ SD

ตัวอย่าง 1: แสดงตัวอย่างการใช้คำสั่ง

10 SP 15 'ตั้งค่าความเร็วที่ 15'

20 MO 1 'เคลื่อนที่ไปตำแหน่งที่ 1 แบบ การเคลื่อนที่แบบคำนวณตามแกน'

30 MS 5 'เคลื่อนที่ไปตำแหน่งที่ 5 แบบ การเคลื่อนที่แบบเส้นตรง'

40 MS 6 'เคลื่อนที่ไปตำแหน่งที่ 6 แบบ การเคลื่อนที่แบบเส้นตรง'

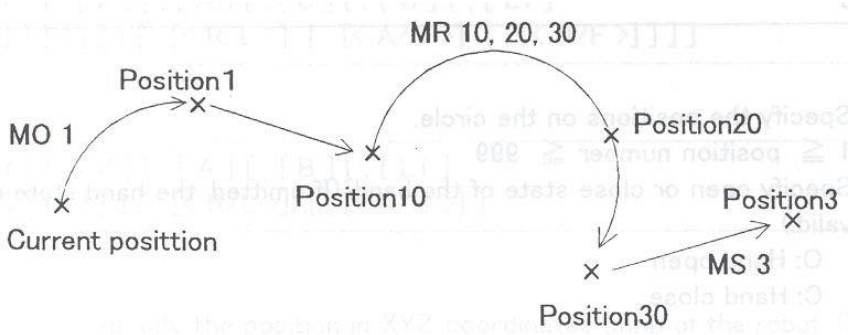
ตัวอย่าง 2: แสดงความแตกต่างของคำสั่ง MO และ MS

10 SP 8 'ตั้งค่าความเร็วที่ 8'

20 MO 1 'เคลื่อนที่ไปตำแหน่งที่ 1 โดยการเคลื่อนที่แบบคำนวณตามแกน'

30 MR 10,20,30 'เคลื่อนที่ไปตำแหน่งที่ 1 โดยการเคลื่อนที่แบบเส้นตรง หลังจากนั้น เคลื่อนที่ไปตำแหน่งที่ 30 โดยการเคลื่อนที่แบบวงกลมโดยคำนวณเส้นทางการเคลื่อนที่จากตำแหน่งที่ 10 20 และ 30'

40 MS 3 'เคลื่อนที่ไปตำแหน่งที่ 1 โดยการเคลื่อนที่แบบเส้นตรง'



****

คำสั่ง: OVR (Override)

การทำงาน: กำหนดค่าอัตราส่วนของความเร็วในการเคลื่อนที่ของหุ่นยนต์เป็นเปอร์เซนต์ซึ่งความเร็วนี้จะครอบคลุมการเคลื่อนที่ของหุ่นยนต์*ทั้งโปรแกรม*

รูปแบบ: OVR <specified override>

คำศัพท์: *<specified override>*

ค่าอัตราส่วนความเร็วเป็นเปอร์เซ็นต์ มีช่วงตั้งแต่ 1 ถึง 200

คำอธิบาย:

(1) กำหนดอัตราส่วนของความเร็วในการทำงานของหุ่นยนต์

(2) คำสั่งนี้มีผลกับการเคลื่อนที่ทุกประเภท เช่น การเคลื่อนที่แบบคำนวณตามแกน, การเคลื่อนที่แบบเส้นตรง และ การเคลื่อนที่แบบวงกลม

(3) ความเร็วที่แท้จริง ในกรณีต่างๆ

- การเคลื่อนที่แบบคำนวณตามแกน = playback override * ค่าอัตราส่วนความเร็วในคำสั่ง OVR * SP command setting value
- การเคลื่อนที่แบบเส้นตรง = playback override * ค่าอัตราส่วนความเร็วในคำสั่ง OVR * SP or SD command setting value

หมายเหตุ: Playback override คือค่าความเร็วที่ถูกกำหนดที่คอนโทรลเลอร์ หรือที่ teaching box
Program override คืออัตราความเร็วที่ถูกระบุอยู่ในคำสั่ง OVR ในโปรแกรม

(4) ค่าเริ่มต้นของ program override เท่ากับ 100%

(5) ค่าอัตราส่วนความเร็วที่ถูกกำหนดอยู่ในโปรแกรมจะมีผลอยู่จนกระทั่งเจอคำสั่ง OVR ใหม่หรือจบโปรแกรม

(6) จะมีเสียงเตือนดังขึ้นถ้าระบุค่าอัตราส่วนความเร็วเท่ากับศูนย์

ตัวอย่าง:

10 SP 30	'กำหนดระดับความเร็วที่ 30 ซึ่งเทียบเท่ากับอัตราส่วนความเร็ว 100% ในคำสั่ง OVR โดยดูจากตารางที่ 2.1'
20 OVR 80	'ตั้งค่าอัตราส่วนความเร็วเท่ากับ 80%'
30 MO 2	'เคลื่อนที่ไปตำแหน่งที่ 2 โดยเคลื่อนที่แบบคำนวณตามแกน '
40 ED	'จบโปรแกรม'

ถ้า playback override ในตัวอย่างด้านบนถูกกำหนดให้เท่ากับ 50%
ดังนั้น อัตราส่วนของความเร็วที่แท้จริงในการเคลื่อนที่แบบคำนวณตามแกน = 50*80*100(%) = 40%
หุ่นยนต์จะเคลื่อนที่ไปตำแหน่งที่ 2 ด้วยความเร็ว 40% ของค่าที่สูงที่สุด

****

คำสั่ง: SP (Speed)

การทำงาน: กำหนดระดับความเร็ว (Operation speed), ความเร่ง/ ความหน่วงและตั้งค่าการเคลื่อนที่อย่างต่อเนื่อง

รูปแบบ: SP <speed level> [,<H/L>[,<CNT setting>]]

คำศัพท์: *<speed level>*

กำหนระดับความเร็ว มีตั้งแต่ 0 ถึง 30

<H/L>

กำหนดระดับความเร่ง/ความหน่วง

H: ความเร่ง/ ความหน่วง สูง (มากที่สุด คือ 0.2 วินาที)

L: ความเร่ง/ ความหน่วง ต่ำ (มากที่สุด คือ 0.4 วินาที)

<CNT setting>

กำหนดสภาวะการทำงานของโหมดการเคลื่อนที่อย่างต่อเนื่อง (continuous path mode)

0: โหมดการเคลื่อนที่อย่างต่อเนื่องหยุดทำงาน

1: โหมดการเคลื่อนที่อย่างต่อเนื่องทำงาน

คำอธิบาย:

(1) กำหนดระดับความเร็วได้ 31 ค่า คือตั้งแต่ 0 ถึง 30 และกำหนดค่าเวลาความเร่ง/ ความหน่วง ได้ 2 ระดับคือ สูงและต่ำ

(2) ระดับความเร็วจะถูกกำหนดเป็นอัตราส่วน (%) กับความเร็วรอบที่มากที่สุดของแต่ละข้อสำหรับการเคลื่อนที่แบบคำนวณตามแกน และอัตราส่วนกับความเร็วที่มากที่สุดของปลายสุดของแขน (650 มม/ วินาที) สำหรับการเคลื่อนที่แบบเส้นตรง

(3) เวลาความเร่ง คือ เวลาที่มากที่สุดสำหรับหุ่นยนต์ที่จะมีความเร็วที่มากที่สุด เพราะฉะนั้น ถ้าความเร็วในการเคลื่อนที่ มีค่าไม่ถึงความเร็วสูงสุด ค่าเวลาความเร่งจะน้อยกว่าค่าที่ระบุเอาไว้

(4) ระยะทางที่เคลื่อนที่ในระหว่างที่มีความเร่ง/ ความหน่วง จะถูกกำหนดโดยความเร็วที่กำหนดและความเร็วที่กำหนดเอาไว้อาจจะไม่ถึงที่กำหนดเอาไว้ถ้าระยะทางในการเคลื่อนที่สั้น

(5) สำหรับการเคลื่อนที่แบบเส้นตรง ปลายสุดของแขนจะเคลื่อนที่ที่ความเร็วคงที่ ในบางกรณี อาจมีเสียงเตือนเนื่องจากข้อต่อใดข้อต่อหนึ่งมีค่าความเร็วมากที่สุดแล้ว

(6) เมื่อกำหนดค่าความเร็วและเวลาความเร่ง/ ความหน่วง แล้ว ค่านั้นจะยังคงอยู่จนกระทั่งมีคำสั่ง SP ที่มีระดับความเร็วต่างไปจากค่าเดิม ค่าตั้งต้นในโปรแกรม (default) คือ SP 12, H ค่าเวลาความเร่ง/ ความหน่วง ของครั้งล่าสุดยังคงอยู่เมื่อไม่ได้กำหนดค่าความเร่งอีกครั้ง

(7) เมื่อกำหนดให้โหมดการเคลื่อนที่อย่างต่อเนื่องทำงาน หุ่นยนต์จะเคลื่อนที่อย่างต่อเนื่องโดยปราศจากความเร่งและความหน่วง จนกระทั่งเจอคำสั่ง SD หรือ SP คำสั่งใหม่ซึ่งจะทำให้โหมดการเคลื่อนที่อย่างต่อเนื่องหยุดทำงาน

ตารางที่ 2.1: ความสัมพันธ์ระหว่างระดับความเร็วและความเร็วในการเคลื่อนที่

SP	**ความเร็วในการเคลื่อนที่ของการเคลื่อนที่แบบคำนวณตามแกน (%)**	**ความเร็วในการเคลื่อนที่ของการเคลื่อนที่แบบเส้นตรง (mm/s)**
0	0.1	0.2
1	0.4	2.7
2	0.6	3.8
3	0.8	5.3
4	1.1	7.3
5	1.5	9.8
6	2.0	13.3
7	2.7	17.8
8	3.7	23.8

SP	**ความเร็วในการเคลื่อนที่ของการเคลื่อนที่แบบคำนวณตามแกน (%)**	**ความเร็วในการเคลื่อนที่ของการเคลื่อนที่แบบเส้นตรง (mm/s)**
9	4.9	31.7
10	6.5	42.4
11	8.2	53.1
12	9.7	63.3
13	11.6	75.3
14	13.7	89.2
15	16.2	105.2
16	19.0	123.7
17	22.2	144.5
18	25.9	168.1
19	29.8	193.8
20	34.2	222.0
21	40.7	264.8
22	47.3	307.6
23	53.9	350.4
24	60.5	393.2
25	67.1	436.0
26	73.7	478.8
27	80.2	521.6
28	86.8	564.4
29	93.4	607.2
30	100.0	650.0

***ตัวอย่าง**:*

10 SP 8	‘กำหนดระดับความเร็วในการเคลื่อนที่เท่ากับ 8 ’
20 MO 5	‘เคลื่อนที่ไปตำแหน่งที่ 5 แบบ การเคลื่อนที่แบบคำนวณตามแกน’
30 SP 10	‘กำหนดระดับความเร็วในการเคลื่อนที่เท่ากับ 10’
40 MS 7	‘เคลื่อนที่ไปตำแหน่งที่ 5 แบบ การเคลื่อนที่แบบเส้นตรง’
50 ED	‘จบโปรแกรม’

****

คำสั่ง: TI (Timer)

การทำงาน: หยุดการเคลื่อนที่ตามช่วงเวลาที่กำหนด

รูปแบบ: TI <timer counter>

คำศัพท์: <*Timer counter*>

ตั้งระยะเวลาในการหน่วงซึ่งมีค่าตั้งแต่ 0 ถึง 32767 (หน่วยเป็น 0.1 วินาที)

คำอธิบาย:

(1) หุ่นยนต์จะหยุดทำงานตามช่วงระยะเวลาที่กำหนดซึ่งเท่ากับ timer counter*0.1 วินาที (ระยะเวลาในการหน่วงมากที่สุด เท่ากับ 3267.7 วินาที)

(2) คำสั่งจะถูกใช้ก่อนและหลังจากที่กริปเปอร์เปิดหรือปิดสำหรับการหยิบจับชิ้นงาน

(3) ค่าตั้งต้น (default) เท่ากับศูนย์

ตัวอย่าง:

10 MO 1,0	'เคลื่อนที่ไปตำแหน่งที่ 1'
20 TI 5	'หน่วงเวลา 0.5 วินาที'
30 GC	'กริปเปอร์ปิด'
40 TI 10	'หน่วงเวลา 1 วินาที'
50 MO 2	'เคลื่อนที่ไปตำแหน่งที่ 2'
60 ED	'จบโปรแกรม'

****

หมวดการควบคุมโปรแกรม

คำสั่งที่อยู่ในหมวดนี้ที่ใช้บ่อยๆ ได้แก่

ED	End
GT	Goto

หมายเหตุ: คำสั่งทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Movemaster Command → Program control command

คำสั่ง: ED (End)

การทำงาน: จบโปรแกรม

รูปแบบ: ED

คำอธิบาย:

(1) คำสั่ง ED จะอยู่ท้ายสุดของโปรแกรม

ตัวอย่าง:

100 SP 3	'กำหนดค่าความเร็วในการเคลื่อนที่เท่ากับ 3'
110 MO 3	'เคลื่อนที่ไปตำแหน่งที่ 3'
120 MO 5	'เคลื่อนที่ไปตำแหน่งที่ 5'

130 ED 'จบโปรแกรม'

****

คำสั่ง: GT (Go To)

การทำงาน: ข้ามไปบรรทัดที่กำหนดโดยไม่มีเงื่อนไข

รูปแบบ: GT <Line number>

คำศัพท์: *<Line number>*

กำหนดบรรทัดที่จะข้ามไป สามารถใส่จำนวนบรรทัดที่จะไปได้ตั้งแต่ 1 ถึง 9999

คำอธิบาย:

(1) คำสั่ง GT ทำให้โปรแกรมข้ามไปยังบรรทัดที่กำหนด

(2) จะมีเสียงเตือนดังขึ้นในขณะที่คอนโทรลเลอร์อ่านคำสั่ง GT ในกรณีที่ไม่มีบรรทัดที่กำหนดให้ข้ามไป

ตัวอย่าง:

10 MO 1 'เคลื่อนที่ไปตำแหน่งที่ 1'

20 GT 100 'กระโดดไปบรรทัดที่ 100 โดยไม่มีเงื่อนไข'

.
.

100 MO 12 'เคลื่อนที่ไปตำแหน่งที่ 12'

110 MO 15 'เคลื่อนที่ไปตำแหน่งที่ 15'

.
.

****

หมวดการควบคุมกริปเปอร์

คำสั่งที่อยู่ในหมวดนี้ที่ใช้บ่อยๆ ได้แก่

GC Grip Close

GO Grip Open

หมายเหตุ: คำสั่งทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Movemaster Command → Hand control command

คำสั่ง: GC (Grip Close)

การทำงาน: ปิดกริปเปอร์

รูปแบบ: GC [<hand number>]

คำศัพท์: *<Hand number>*

กำหนดหมายเลขกริปเปอร์

หมายเลขกริปเปอร์ คือ 1, 2, 3 และ 4

ค่า default คือ หมายเลข1

คำอธิบาย:

(1) กริปเปอร์ทำงานด้วยมอเตอร์ (Motor-operated hand) คือ กริปเปอร์หมายเลข 1เท่านั้น

(2) กริปเปอร์ทำงานด้วยลม (Pneumatic hand) ใช้โซลินอยด์วาล์วเพื่อปิด/เปิดกริปเปอร์

(3) หุ่นยนต์ต้องการช่วงระยะเวลาหนึ่งในการจัดตำแหน่งและหยิบชิ้นงาน เพราะฉะนั้น จำเป็นที่ต้องมีการหน่วงเวลาก่อนและหลังคำสั่ง GC

ตัวอย่าง:

10 MO 10,O	'เคลื่อนที่ไปตำแหน่งที่ 10 และกริปเปอร์เปิด'
20 TI 5	'หน่วงเวลา 0.5 วินาที'
30 GC	'กริปเปอร์ปิดเพื่อจับชิ้นงาน'
40 TI 5	'หน่วงเวลา 0.5 วินาที'
50 MO 15,C	'เคลื่อนที่ไปตำแหน่งที่ 15 และกริปเปอร์ปิด'

****

คำสั่ง: GO (Grip Open)

การทำงาน: เปิดกริปเปอร์

รูปแบบ: GO [<hand number>]

คำศัพท์: *<Hand number>*

กำหนดหมายเลขกริปเปอร์

หมายเลขกริปเปอร์ คือ 1, 2, 3 และ 4

ค่า default คือ หมายเลข1

คำอธิบาย:

(1) กริปเปอร์ทำงานด้วยมอเตอร์ (Motor-operated hand) คือ กริปเปอร์หมายเลข 1เท่านั้น

(2) กริปเปอร์ทำงานด้วยลม (Pneumatic hand) ใช้โซลินอยด์วาล์วเพื่อปิด/เปิดกริปเปอร์

(3) หุ่นยนต์ต้องการช่วงระยะเวลาหนึ่งในการจัดตำแหน่งและหยิบชิ้นงาน เพราะฉะนั้น จำเป็นที่ต้องมีการหน่วงเวลาก่อนและหลังคำสั่ง GO

ตัวอย่าง:

10 MO 10,C	'เคลื่อนที่ไปตำแหน่งที่ 10 และกริปเปอร์ปิด'
20 TI 5	'หน่วงเวลา 0.5 วินาที'
30 GO	'กริปเปอร์เปิดเพื่อปล่อยชิ้นงาน'
40 TI 5	'หน่วงเวลา 0.5 วินาที'
50 MO 15,O	'เคลื่อนที่ไปตำแหน่งที่ 15 และกริปเปอร์เปิด'

****

หมวดการควบคุมอินพุท/เอาท์พุท

คำสั่งที่อยู่ในหมวดนี้ที่ใช้บ่อยๆ ได้แก่

OB	Output Bit
TBD	Test Bit Direct

หมายเหตุ: คำสั่งทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Movemaster Command → I/O control command

คำสั่ง: OB (Output Bit)

การทำงาน: กำหนดสถานะเอาท์พุทของบิท (Bit) ที่กำหนดผ่านทางพอร์ทสัญญาณเอาท์พุทภายนอก (external output port)

รูปแบบ: OB [<+/->] <output bit number>

คำศัพท์:

<+/->

กำหนดสภาวะ ON หรือ OFF ของบิทที่กำหนด

+: ON

-: OFF

<Bit number>

กำหนดหมายเลขบิทของเอาท์พุท ซึ่งมีค่าตั้งแต่ 0 ถึง 32767

คำอธิบาย:

(1) + หมายถึงให้บิทนั้นทำงาน (ON) - หมายถึงให้บิทนั้นหยุดทำงาน (OFF)

(2) บิทอื่นนอกเหนือจากที่กำหนดไว้ไม่ได้รับผลกระทบใดจากคำสั่งนี้ สถานะของเอาท์พุทที่กำหนดจะยังคงอยู่ในสภาวะเดิมจนกระทั่งมีการกำหนดค่าใหม่ในบิทนั้นในคำสั่ง OB

(3) สำหรับกริปเปอร์ทำงานด้วยลม เราสามารถกำหนดสั่งให้กริปเปอร์ปิด/เปิดด้วยคำสั่ง OB แต่ไม่สามารถใช้กับกริปเปอร์ทำงานด้วยมอเตอร์

ตัวอย่าง:

```
10 OB -10          'สั่งให้สถานะของบิดที่ 10 ปิด'
20 ED              'จบโปรแกรม'
```

****

คำสั่ง: TBD (Test Bit Direct)

การทำงาน: กระโดดข้ามไปยังบรรทัดที่กำหนดไว้ ถ้าค่าของอินพุทบิทจากภายนอกตรงกับเงื่อนไข

รูปแบบ: TBD [<+/->] <input bit number>, <branching line>

คำศัพท์:

<+/->

กำหนดสภาวะบิทที่จะเป็นเงื่อนไข

+: ON

-: OFF

<Input bit number>

กำหนดหมายเลขบิทของอินพุท ซึ่งมีค่าตั้งแต่ 0 ถึง 32767

<Branching line number>

กำหนดหมายเลขบรรทัดที่จะกระโดดข้ามไป ซึ่งมีค่าตั้งแต่ 1 ถึง 9999

คำอธิบาย:

(1) กระโดดข้ามไปยังบรรทัดที่กำหนดไว้ ถ้าสถานะอินพุทจากภายนอกตรงกับเงื่อนไขที่กำหนดไว้

(2) โปรแกรมจะกระโดดข้ามไปยังบรรทัดที่กำหนดไว้ ถ้าสถานะของอินพุทตรงกับเงื่อนไข ถ้าไม่ตรงกับเงื่อนไข โปรแกรมจะอ่านคำสั่งในบรรทัดถัดไป

(3) จะมีเสียงเตือนดังขึ้น ถ้าไม่มีบรรทัดที่ระบุให้กระโดดไป

ตัวอย่าง:

10 TBD +19,100	'ถ้าสัญญาณอินพุทบิทที่ 19 ON ให้กระโดดไปยังบรรทัดที่ 100'
20 MS 1	'เคลื่อนที่ไปตำแหน่งที่ 1 โดย การเคลื่อนที่แบบเส้นตรง'
30 ED	'จบโปรแกรม'
.	
.	
100 MO 10	เคลื่อนที่ไปตำแหน่งที่ 10'

****

สำหรับหมวดการคำนวณทางคณิตศาสตร์ หมวดการติดต่อสื่อสารผ่าน RS232C และหมวดอื่นๆ ไม่มีการอ้างถึงในคู่มือการใช้งานเล่มนี้แต่ผู้เรียนสามารถค้นคว้าคำสั่งของทั้งสามหมวดนี้ใน COSIMIR Help

COSIMIR → Programming → Movemaster Command → Operation, substitute and exchange command/Communication command (Using RS323C)/other command

2.5 **คำสั่งต่าง ๆ ในภาษา** MELFA BASIC IV

เนื่องจาก Melfa Basic IV มีฟังก์ชั่นต่างๆ มากกว่า ทำให้สามารถสร้างโปรแกรมที่มีความซับซ้อนได้มากกว่าภาษา Movemaster command

คำสั่งถูกแบ่งออกเป็น 7 หมวด โดยผู้เรียนต้องเลือกคำสั่งให้ถูกกับงาน

a) หมวดการควบคุมตำแหน่งและการเคลื่อนที่ (Position and motion control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับตำแหน่งและรวมไปถึง การกำหนดความเร็วในการเคลื่อนที่ การหน่วงเวลา รูปแบบในการเคลื่อนที่, tool, palette, เป็นต้น

b) หมวดการควบคุมโปรแกรม (Program control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับ การเลือกทำแบบมีเงื่อนไข (conditional branching) การทำซ้ำ (repetitive operation) การขัดจังหวะสัญญาณ (interrupting of signals) และ เคาน์เตอร์ เป็นต้น

c) หมวดการควบคุมกริปเปอร์ (Hand control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับการปิด/เปิดของกริปเปอร์

d) หมวดการควบคุมอินพุท/เอาท์พุท (I/O control commands)
คำสั่งในหมวดนี้เกี่ยวข้องกับการควบคุมอินพุท/เอาท์พุทจากสัญญาณภายนอก

e) คำสั่งอื่นๆ
คำสั่งในหมวดนี้เกี่ยวข้องกับการตั้งค่าตัวแปร การเลือกโปรแกรม และการใส่คำอธิบายในโปรแกรม

หมวดการควบคุมตำแหน่งและการเคลื่อนที่

คำสั่งที่อยู่ในหมวดนี้ที่ใช้บ่อยๆ ได้แก่

DLY	Delay
MOV	Move
MVS	Move Straight
OVRD	Override
SPD	Speed

หมายเหตุ: คำสั่งทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Melfa-Basic IV Command → Melfa-Basic IV position and motion control command

คำสั่ง: DLY (Delay)

การทำงาน: หน่วงเวลาตามระยะเวลาที่กำหนด

รูปแบบ: DLY <time>

คำศัพท์: *<time>*

ระยะเวลาที่หน่วง หน่วยเป็นวินาที ช่วงระยะเวลาที่น้อยที่สุดที่ระบุได้ คือ 0.05 วินาที

คำอธิบาย:

(1) คำสั่ง DLY ถูกใช้เพื่อหน่วงเวลาในโปรแกรม

(2) ถ้าคอนโทรลเลอร์อ่านคำสั่ง END หรือคำสั่งในบรรทัดสุดท้ายของโปรแกรมหรือ ถ้ามีการกดปุ่มฉุกเฉิน (Emergency stop) เพื่อหยุดการอ่านโปรแกรมในระหว่างที่หน่วงเวลาไว้ สถานะของสัญญาณเอาท์พุทจะยังคงอยู่ในค่าเดิม

ตัวอย่าง:

10 DLY 30 'หน่วงเวลา 30 วินาที'

****

คำสั่ง: MOV (Move)

การทำงาน: เคลื่อนที่จากตำแหน่งปัจจุบันไปยังตำแหน่งปลายทางโดยเคลื่อนที่แบบ การเคลื่อนที่แบบคำนวณตามแกน

รูปแบบ: MOV <Target Position> [,[,<Close Distance>] [<Interpolation Type>]] [Appended conditions]

คำศัพท์:

<*Target Position*> คือ ตำแหน่งปลายทาง

<Close Distance>

ถ้ามีการกำหนดค่า close distance ตำแหน่งปลายทางจริงจะเป็นตำแหน่งที่ห่างจากตำแหน่งที่กำหนด (target position) ในแนวแกน Z (+/-) เป็นระยะทางเท่ากับ close distance

+ หมายถึง ตำแหน่งปลายทางจริงอยู่ต่ำกว่าชิ้นงาน

- หมายถึง ตำแหน่งปลายทางจริงอยู่สูงกว่าชิ้นงาน

<Interpolation type>

กำหนดชนิดของ interpolation ซึ่งสามารถระบุดังรูปแบบในบรรทัดถัดไป

Type <Numeric constant1>, <Numeric constant2>

โดย Numeric constant 1… Detour/short cut = 1/0

Numeric constant 2… Invalid

ค่า default คือ 1,0 (detour)

คำอธิบาย:

(1) Detour หมายถึง ลักษณะการเคลื่อนที่ที่ตรงตามกับที่กำหนดตำแหน่งในการเคลื่อนที่ (teach) ให้หุ่นยนต์ทุกประการ

(2) Short cut หมายถึง ลักษณะการเคลื่อนที่ในระหว่างเดินทางจากจุดเริ่มต้นไปจุดสุดท้ายโดยใช้การเคลื่อนที่ที่น้อยกว่าแบบ detour

(3) การกำหนดลักษณะการเคลื่อนที่แบบ detour และ short cut จะมีความแตกต่างกันมากในกรณีที่ต้องมีการเคลื่อนที่ในแกนใดแกนหนึ่งมากกว่า 180 องศา

(4) Numeric constant2 ไม่มีผลใดๆ กับการเคลื่อนที่แบบ การเคลื่อนที่แบบคำนวณตามแกน

ตัวอย่าง:

10 MOV P1 TYPE 1,0 'เคลื่อนที่ไปตำแหน่งที่ 1 โดยมีลักษณะการเคลื่อนที่แบบ detour'

****

คำสั่ง: MVS (Move Straight)

การทำงาน: เคลื่อนที่จากตำแหน่งปัจจุบันไปยังตำแหน่งปลายทางโดยเคลื่อนที่แบบ การเคลื่อนที่แบบเส้นตรง

รูปแบบที่ 1: MVS <Target Position> [,[,<Close Distance>] [<Interpolation Type>]] [Appended conditions]

รูปแบบที่ 2: MVS <Separation Distance> [<Interpolation Type>]

คำศัพท์:

<*Target Position*> คือ ตำแหน่งปลายทาง

<Close Distance>

ถ้ามีการกำหนดค่า close distance ตำแหน่งปลายทางจริงจะเป็นตำแหน่งที่ห่างจากตำแหน่งที่กำหนด (target position) ในแนวแกน Z (+/-) เป็นระยะทางเท่ากับ close distance

+ หมายถึง ตำแหน่งปลายทางจริงอยู่ต่ำกว่าชิ้นงาน

- หมายถึง ตำแหน่งปลายทางจริงอยู่สูงกว่าชิ้นงาน

<Interpolation type>

กำหนดชนิดของ interpolation ซึ่งสามารถระบุดังรูปแบบในบรรทัดถัดไป

Type <Numeric constant1>, <Numeric constant2>

โดย Numeric constant 1… Detour/short cut = 1/0

Numeric constant 2… 3-axis orthogonal/equivalent rotation = 1/0

ค่า default คือ 1,0 (detour, equivalent rotation)

<Separation Distance>

ถ้ามีการกำหนดค่า separation distance ตำแหน่งปลายทางจริงจะเป็นตำแหน่งที่ห่างจากตำแหน่งที่กำหนด (target position) ในแนวแกน Z (+/-) เป็นระยะทางเท่ากับ close distance

+ หมายถึง ตำแหน่งปลายทางจริงอยู่ต่ำกว่าชิ้นงาน

- หมายถึง ตำแหน่งปลายทางจริงอยู่สูงกว่าชิ้นงาน

คำอธิบาย:

(1) การเคลื่อนที่แบบเส้นตรง คือ ชนิดของการเคลื่อนที่ที่หุ่นยนต์เคลื่อนที่จากตำแหน่งปัจจุบันไปตำแหน่งปลายทางโดยกริปเปอร์เคลื่อนที่ผ่านจุดต่างๆ ในระหว่างจุดเริ่มต้นถึงจุดปลายทางเป็นเส้นตรง

ตัวอย่าง:

10 MVS P1 ‘เคลื่อนที่ไปตำแหน่งที่ 1 โดยเคลื่อนที่แบบ การเคลื่อนที่แบบเส้นตรง’

****

คำสั่ง: OVRD (Override)

การทำงาน: กำหนดอัตราส่วนความเร็วในการเคลื่อนที่ของหุ่นยนต์ซึ่ง*ครอบคลุมทั้งโปรแกรม*

รูปแบบ: OVRD <Designated override>

คำศัพท์:

<Designated override> คือ อัตราส่วนความเร็วเป็นจำนวนจริง หน่วยเป็นเปอร์เซ็นต์ (ช่วงตัวเลขที่ใส่ได้คือ 1 ถึง 100 ตัวเลข ตัวเลขนอกเหนือจากช่วงนี้จะทำให้เกิด error) นอกจากจะใส่เป็นตัวเลขแล้ว ยังสามารถใส่ค่าแบบที่มีการกระทำทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร เป็นต้น

คำอธิบาย:

(1) คำสั่ง OVRD สามารถใช้ได้กับทุกชนิดของการเคลื่อนที่ไม่ว่าจะเป็นการเคลื่อนที่แบบคำนวณตามแกนหรือแบบเส้นตรง

(2) อัตราส่วนของความเร็วที่แท้จริง ในกรณีต่างๆ

- การเคลื่อนที่แบบคำนวณตามแกน = playback override * อัตราส่วนความเร็วในคำสั่ง OVRD * อัตราส่วนความเร็วในคำสั่ง JOVRD
- การเคลื่อนที่แบบเส้นตรง = playback override * อัตราส่วนความเร็วในคำสั่ง OVRD * ความเร็วในคำสั่ง SPD

หมายเหตุ: Playback override คือค่าความเร็วที่ถูกกำหนดที่คอนโทรลเลอร์หรือ teaching box

Program override จะถูกระบุอยู่ในคำสั่ง OVRD ในโปรแกรม

JOVRD เป็นคำสั่งกำหนดค่าความเร็วในการเคลื่อนที่ของหุ่นยนต์แบบตามแกนเป็นหน่วยเปอร์เซนต์เทียบกับความเร็วสูงสุด ซึ่ง*ครอบคลุมเฉพาะการเคลื่อนที่แบบจากจุดไปจุดเท่านั้น*

ค่า default ของ JOVRD command อยู่ที่ 100%

ตัวอย่าง:

10 OVRD 50 'กำหนดอัตราส่วนความเร็วเท่ากับ 50%'

****

คำสั่ง: SPD (Speed)

การทำงาน: กำหนดความเร็วซึ่ง*ครอบคลุมการเคลื่อนที่แบบเส้นตรงหรือวงกลมเท่านั้น*

รูปแบบ: SPD <Designated speed>

คำศัพท์:

<Designated speed> คือ ค่าความเร็วเป็นจำนวนจริง หน่วยเป็นมิลลิเมตร/วินาที

คำอธิบาย:

(1) คำสั่ง SPD จะทำหน้าที่เปลี่ยนเฉพาะค่าความเร็วในการเคลื่อนที่แบบเส้นตรงและวงกลมเท่านั้น

(2) เมื่อกำหนดค่าความเร็วเป็น M_NSPD หุ่นยนต์จะเคลื่อนที่ที่ความเร็วสูงสุด

(3) ค่าความเร็วในคำสั่ง SPD จะยังคงอยู่จนกระทั่งคอนโทรลเลอร์อ่านคำสั่ง SPD ที่อยู่ถัดไป

(4) ค่าความเร็วในคำสั่ง SPD จะกลับสู่ค่า default เมื่อจบโปรแกรม

ตัวอย่าง:

10 SPD 100 'กำหนดค่าความเร็วในการเคลื่อนที่เป็นเส้นตรงหรือวงกลมเท่ากับ 100 มิลลิเมตร/วินาที'

ตารางที่ 2.2: **ตารางสรุปคำสั่งเกี่ยวกับการกำหนดความเร็ว**

คำสั่งเกี่ยวกับการกำหนดความเร็ว	คำอธิบาย
ACCEL	กำหนดความเร่งและความหน่วงระหว่างการเคลื่อนที่ มีหน่วยเป็นเปอร์เซนต์เมื่อเทียบกับความเร่งหรือความหน่วงสูงสุด
OVRD	กำหนดอัตราส่วนความเร็วในการเคลื่อนที่ของหุ่นยนต์ซึ่ง*ครอบคลุมทั้งโปรแกรม* หน่วยเป็นเปอร์เซนต์เมื่อเทียบกับความเร็วสูงสุด
JOVRD	กำหนดอัตราส่วนความเร็วในการเคลื่อนที่ของหุ่นยนต์*แบบคำนวณตามแกน* หน่วยเป็นเปอร์เซนต์เมื่อเทียบกับความเร็วสูงสุด
SPD	กำหนดความเร็วในการเคลื่อนที่ของหุ่นยนต์*ที่เป็นเส้นตรงและวงกลม* หน่วยเป็นมิลลิเมตรต่อวินาที

****

หมวดการควบคุมโปรแกรม

คำสั่งที่อยู่ในหมวดนี้ที่ใช้บ่อยๆ ได้แก่

END End

GOTO Go to

IF THEN ELSE If Then Else

หมายเหตุ: คำสั่งทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Melfa-Basic IV Command → Melfa-Basic IV program control command

คำสั่ง: END (End)

การทำงาน: จบโปรแกรม

รูปแบบ: END

คำอธิบาย:

(1) ในหนึ่งโปรแกรมสามารถมีคำสั่ง END ได้หลายครั้ง

(2) คำสั่ง END ไม่จำเป็นต้องอยู่ตำแหน่งสุดท้ายของโปรแกรมเสมอไป

(3) เมื่อคอนโทรลเลอร์อ่านคำสั่ง END ค่าตัวแปรในคำสั่ง SPD, ACCEL, OADL, JOVRD, OVRD, FINE, and CNT จะถูกรีเซ็ท

ตัวอย่าง:

100 END 'ตั้งค่าความเร็วในการเคลื่อนที่เป็นเส้นตรงหรือวงกลมเท่ากับ 100 มิลลิเมตร/วินาที'

****

คำสั่ง: GOTO (Go to)

การทำงาน: ข้ามไปยังบรรทัดที่กำหนดไว้โดยไม่มีเงื่อนไข

รูปแบบ: GOTO <Branch Destination>

คำศัพท์:

<Branch Destination> หมายถึง บรรทัดที่จะกระโดดข้ามไป'

****

คำสั่ง: IF THEN ELSE (If Then Else)

การทำงาน: การเปรียบเทียบเงื่อนไขและจะมีการทำงานขั้นตอนต่างๆ เมื่อเงื่อนไขเป็นจริงหรือเท็จ

รูปแบบ: IF<Expression> THEN<Process> [ELSE<Process>]

คำศัพท์:

<Expression>

เงื่อนไขทางคณิตศาสตร์หรือทางตรรกศาสตร์เพื่อเปรียบเทียบตัวแปรกับเป้าหมาย

ตัวอย่างตัวแปร Robot status variable เช่น

M_OUT ตัวแปรเกี่ยวกับเอาท์พุทบิท (General purpose output bit device)

M_IN ตัวแปรเกี่ยวกับอินพุทบิท (General purpose input bit device)

หมายเหตุ: ตัวแปรทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Melfa-Basic IV Command → Alphabetical overview of Melfa Basic IV Robot status variable

ตัวแปร: M_OUT

การทำงาน: สามารถอ่านค่าสัญญาณเอาท์พุทและเปลี่ยนแปลงค่าสัญญาณเอาท์พุท (Both reading and writing)

รูปแบบ: M_OUT (Bit Number) = bit signal output

หมายเหตุ: bit signal output 0=off, 1=on

ตัวอย่าง: M_OUT(1) = 0'เปลี่ยนค่าสัญญาณเอาท์พุทบิทที่ 1 ให้มีค่าเป็นศูนย์หรือปิด'

ตัวแปร: M_IN

การทำงาน: สามารถอ่านค่าสัญญาณอินพุทเท่านั้น

รูปแบบ: M_IN (Bit Number) = bit signal input

หมายเหตุ: bit signal input 0=off, 1=on

ตัวอย่าง: IF M_IN(1) = 0 THEN 100 'ถ้าค่าสัญญาณอินพุทบิทที่ 1 มีค่าเป็นศูนย์ ให้ข้ามไป บรรทัดที่ 100'

<Process>

ขั้นตอนการทำงานต่อจากคำสั่ง THEN เมื่อเงื่อนไขเป็นจริง

ขั้นตอนการทำงานต่อจากคำสั่ง ELSE เมื่อเงื่อนไขเป็นเท็จ

ตัวอย่าง:

```
100 IF M1>10 THEN 1000          'ถ้าเงื่อนไขตัวแปร M1 มีค่ามากกว่า 10 เป็นจริงให้ข้ามไปทำขั้นตอนใน บรรทัดที่ 1000'
110 IF M1>10 THEN GOTO 1000 ELSE GOTO 2000
                                'ถ้าเงื่อนไขตัวแปร M1 มีค่ามากกว่า 10 เป็นจริงให้ข้ามไปทำขั้นตอนในบรรทัดที่ 1000
                                 ถ้าเงื่อนไขตัวแปร M1 มีค่ามากกว่า 10 เป็นเท็จให้ข้ามไปทำขั้นตอนใน บรรทัดที่ 2000'
```

****

หมวดการควบคุมกริปเปอร์

คำสั่งที่อยู่ในหมวดนี้ที่ใช้บ่อยๆ ได้แก่

HCLOSE	Hand Close
HOPEN	Hand Open

หมายเหตุ: คำสั่งทั้งหมดที่อยู่ในหมวดนี้ อยู่ใน COSIMIR Help

COSIMIR → Programming → Melfa-Basic IV Command → Melfa-Basic IV hand control commands

คำสั่ง: HOPEN /HCLOSE (Hand Open/Close)

การทำงาน: สั่งกริปเปอร์ให้เปิดหรือปิด

รูปแบบ: HOPEN<Hand No.> [, <Starting grasp force>, <Holding grasp force>, <Starting grasp force holding time>]

HCLOSE<Hand No.>

คำศัพท์: *<Hand No.>*

กำหนดหมายเลขกริปเปอร์ ซึ่งมีให้เลือกตั้งแต่ 1 ถึง 8

<Starting grasp force>

ค่าตัวแปรนี้ใช้ได้กับกริปเปอร์มอเตอร์เท่านั้น กำหนดค่าของแรงในการจับชิ้นงาน (grasping force) เมื่อเริ่มต้นปิด/เปิดกริปเปอร์ ค่าของแรงสามารถกำหนดได้ตั้งแต่ 0 ถึง 63 (63 = 3.5 kgf) ค่า default เท่ากับ 63

<Holding grasp force>

ค่าตัวแปรนี้ใช้ได้กับกริปเปอร์มอเตอร์เท่านั้น กำหนดค่าของแรงในการจับชิ้นงานในการคงสภาพปิด / เปิดของกริปเปอร์ ค่าของแรงสามารถกำหนดได้ตั้งแต่ 0 ถึง 63 (63 = 3.5 kgf) ค่า default เท่ากับ 63

<Starting grasp force holding timer>

ค่าตัวแปรนี้ใช้ได้กับกริปเปอร์มอเตอร์เท่านั้น กำหนดช่วงเวลาที่ให้กริปเปอร์คงค่าของแรงในการเริ่มต้นปิด/เปิดกริปเปอร์ ซึ่งสามารถกำหนดช่วงเวลาได้ตั้งแต่ 0 วินาที ค่า default เท่ากับ 0.3 วินาที

***ตัวอย่าง**:*

```
10 HOPEN 1          'เปิดกริปเปอร์หมายเลข1'
20 HCLOSE 2         'ปิดกริปเปอร์หมายเลข2'
```

****

ในบทที่ 3 จะเกี่ยวกับขั้นตอนการสร้างโมเดล ขั้นตอนการกำหนดตำแหน่งในการเคลื่อนที่และขั้นตอนการเขียนโปรแกรม

3.0 ส่วนประกอบที่จำเป็นต่อการรันโปรแกรม

ส่วนประกอบที่จำเป็นต่อการรันโปรแกรมมีดังต่อไปนี้

1) **โมเดล** (model)
2) **ตำแหน่ง** (Position Lists)
3) **โปรแกรม**

3.1 ขั้นตอนการสร้างโมเดล

ขั้นตอนการสร้างโมเดลมีสองวิธี คือ การเรียกจากไฟล์ตัวอย่าง หรือการสร้างโมเดลเอง

3.1.1 ขั้นตอนการเรียกโมเดลจากไฟล์ตัวอย่าง

การเรียกไฟล์ตัวอย่างมีอยู่ 2 วิธี คือ

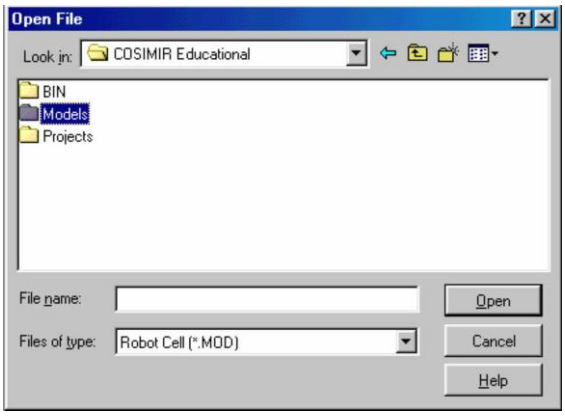
1) เปิดจากไดเร็กทอรี่ที่ลงซอฟต์แวร์ COSIMIR Educational ไว้ หรือ
2) เปิดจาก COSIMIR Help

วิธีการที่ 1: **เปิดจากไดเร็กทอรี่ที่มีซอฟต์แวร์** COSIMIR Educational **อยู่**

1. หลังจากเข้าโปรแกรม COSIMIR Education ให้ไปที่ File>Open เลือกไดเร็กทอรี่ที่ซอฟต์แวร์ COSIMIR Educational อยู่แล้วจะปรากฏหน้าต่าง (Window) ดังรูป 3.1.1a

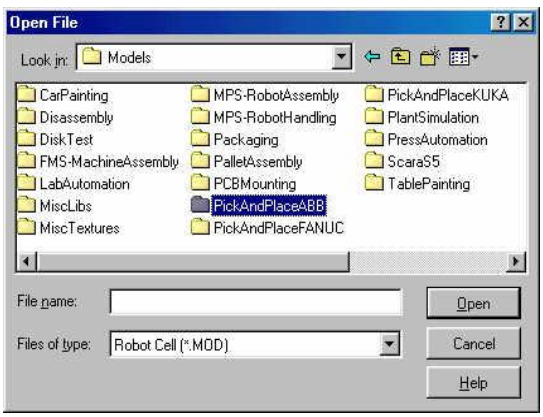
   หมายเหตุ: โดย default จะอยู่ใน C:/Program files/COSIMIR Educational

รูป 3.1.1a



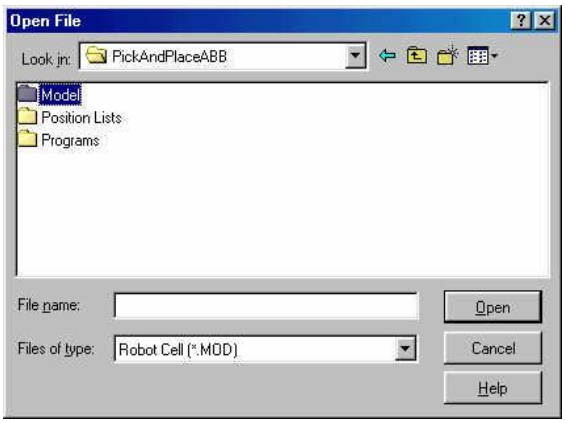
2. เลือกเข้าไปในโฟลเดอร์โมเดล จะปรากฏหน้าต่างซึ่งมี โมเดลให้เลือกมากมาย ดังรูป 3.1.1b

รูป 3.1.1b



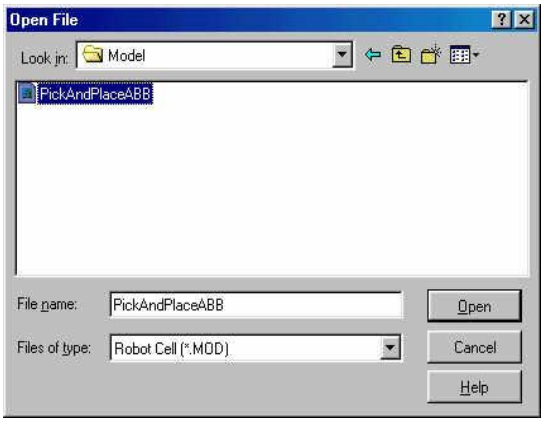
3. เลือกโมเดลที่ต้องการ ตัวอย่างเช่น PickAndPlaceABB ดังรูป 3.1.1c

รูป 3.1.1c



4. เลือกเข้าไปที่ โฟลเดอร์โมเดล ปรากฏไฟล์ชื่อ PickAndPlaceABB ซึ่งมีนามสกุล .MOD ดังรูป 3.1.1d

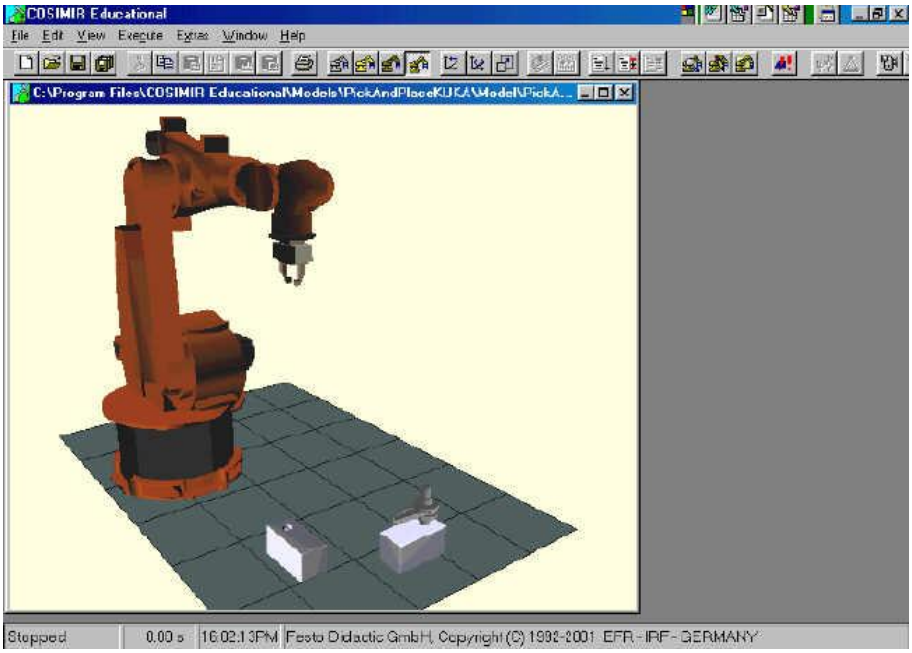
รูป : 3.1.1d



5. เมื่อดับเบิ้ลคลิ๊กที่ไฟล์ PickAndPlaceABB จะปรากฏหน้าต่างดังรูป 3.1.1e

หมายเหตุ: หน้าต่างที่ปรากฏอาจมีความแตกต่างจากหน้าต่างที่ปรากฏในรูป 3.1.1e คือ อาจมีหน้าต่างอื่นที่นอกเหนือจากหน้าต่างที่เป็น workcell

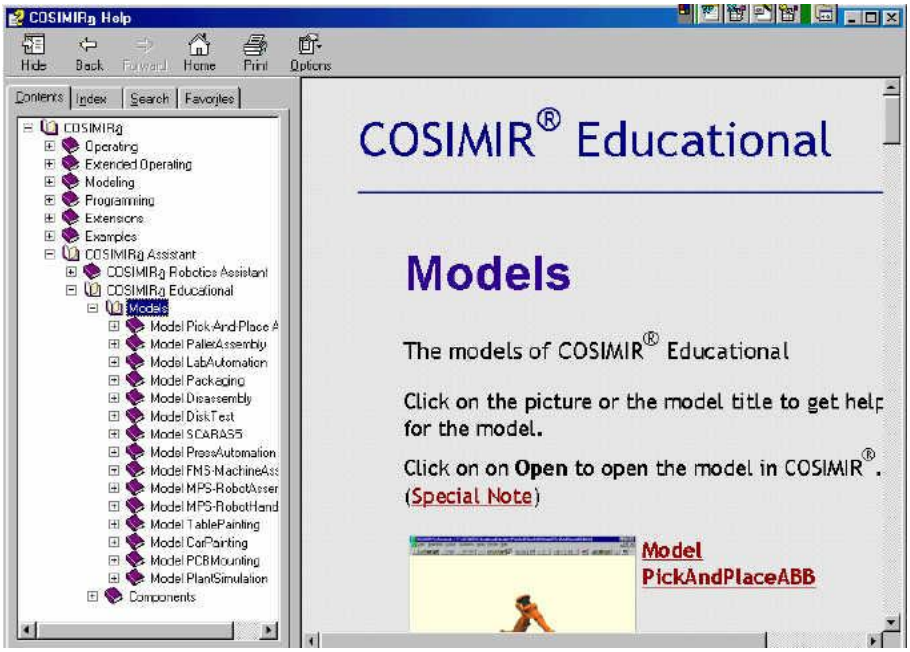
รูป: 3.1.1e



วิธีการที่ 2: **เปิดจาก** COSIMIR Help

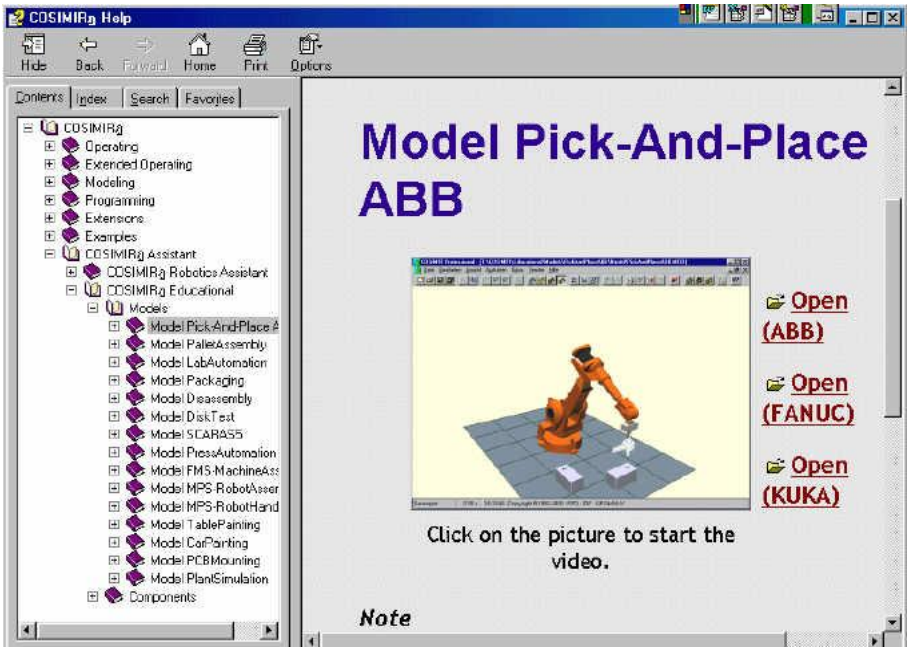
1. หลังจากเปิดโปรแกรม COSIMIR Educational แล้ว โปรแกรมจะปรากฏขึ้นพร้อมกับ COSIMIR Help ให้มาที่หน้าต่างของ COSIMIR Help แล้วคลิ๊กที่เครื่องหมายบวกด้านหน้าของ COSIMIR>COSIMIR Assistant> COSIMIR Educational> Model ดังรูป 3.1.1f ซึ่งมีโมเดลหุ่นยนต์ต่างๆ ให้เลือก

รูป: 3.1.1f



2. เลือกที่โมเดลที่ต้องการ ตัวอย่างเช่น PickAndPlaceABB จะปรากฏหน้าต่างดังรูป 3.1.1g หลังจากนั้นกดปุ่ม 

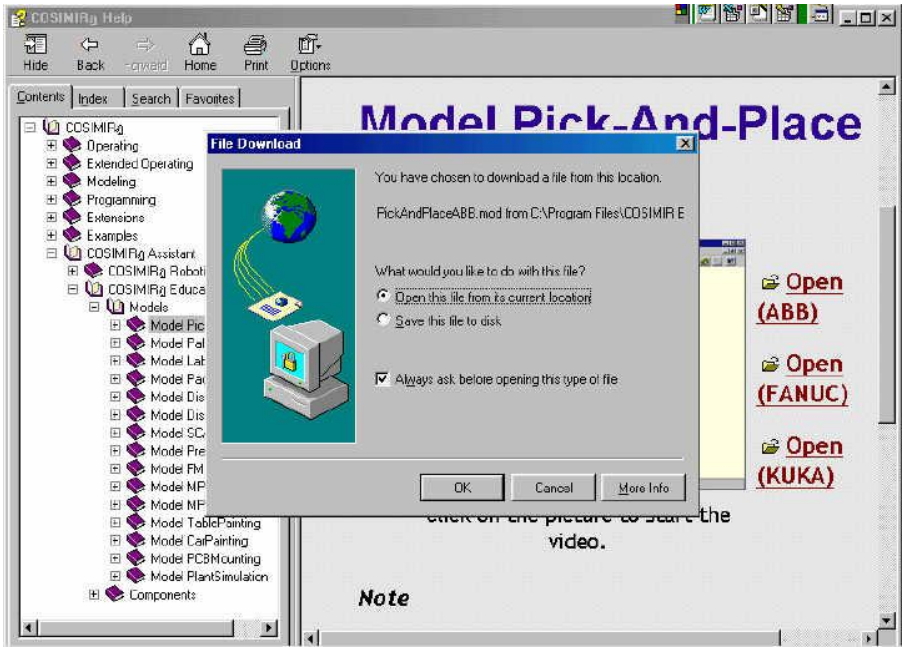
รูป: 3.1.1g



3. หลังจากนั้นจะปรากฏหน้าต่างดังรูป 3.1.1h ให้เลือก ‘Open this file from its current location’ จากนั้นคลิ๊ก OK เมื่อคลิ๊ก OK แล้วจะปรากฏหน้าต่างเช่นเดียวกับรูป 3.1.1e

หมายเหตุ: สาเหตุที่ไม่สามารถเลือก ‘Save this file to disk’ ได้เนื่องจากข้อจำกัดของซอฟต์แวร์ Cosimir Educational ที่ไม่สามารถบันทึกโมเดลได้

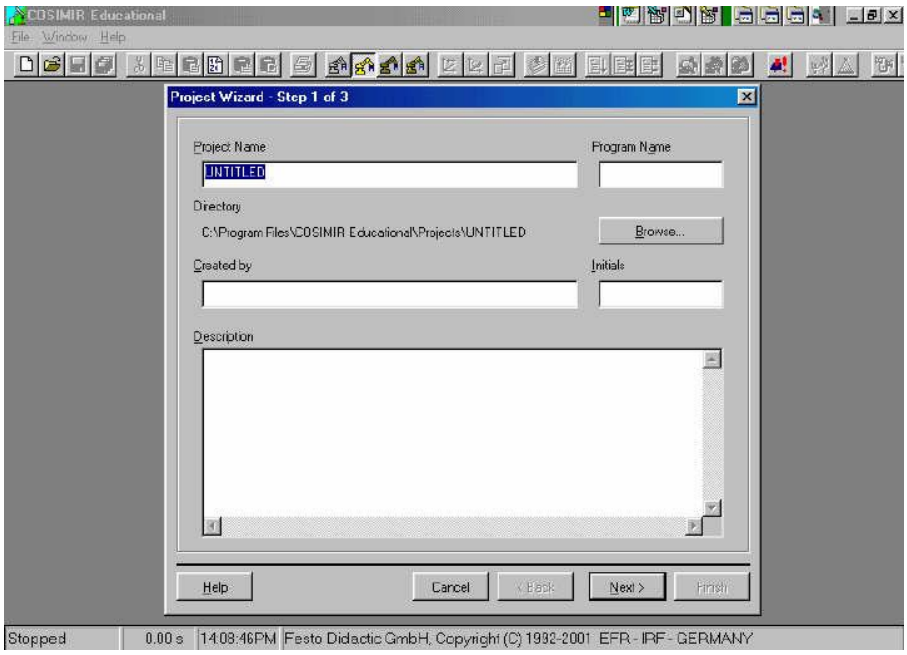
รูป: 3.1.1h



3.1.2 ขั้นตอนการสร้างโมเดลเอง

1. หลังจากเปิดโปรแกรม COSIMIR Educational แล้ว ใช้คำสั่ง File> Project Wizard จะปรากฏหน้าต่างดังรูป 3.1.2a

รูป: 3.1.2a



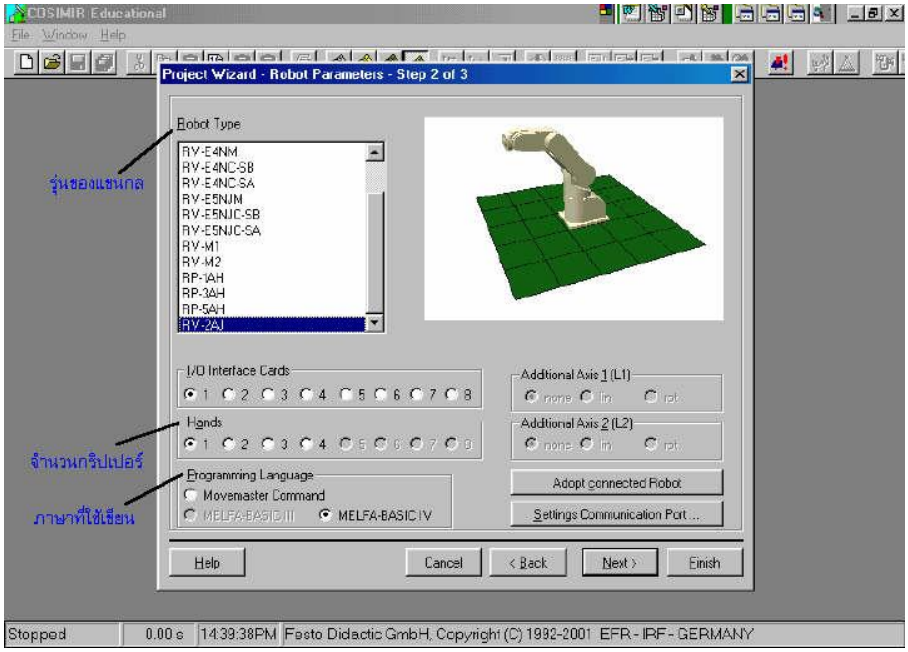
2. ใส่ชื่อ Project Name ซึ่งชื่อของ Project Name จะเป็นชื่อของโมเดล Position Lists และโปรแกรมโดยอัตโนมัติ ทั้งสามไฟล์จะมีชื่อไฟล์เหมือนกันแต่คนละนามสกุล

3. หลังจากนั้นกดปุ่ม Next แล้วจะปรากฏหน้าต่างดังรูป 3.1.2b

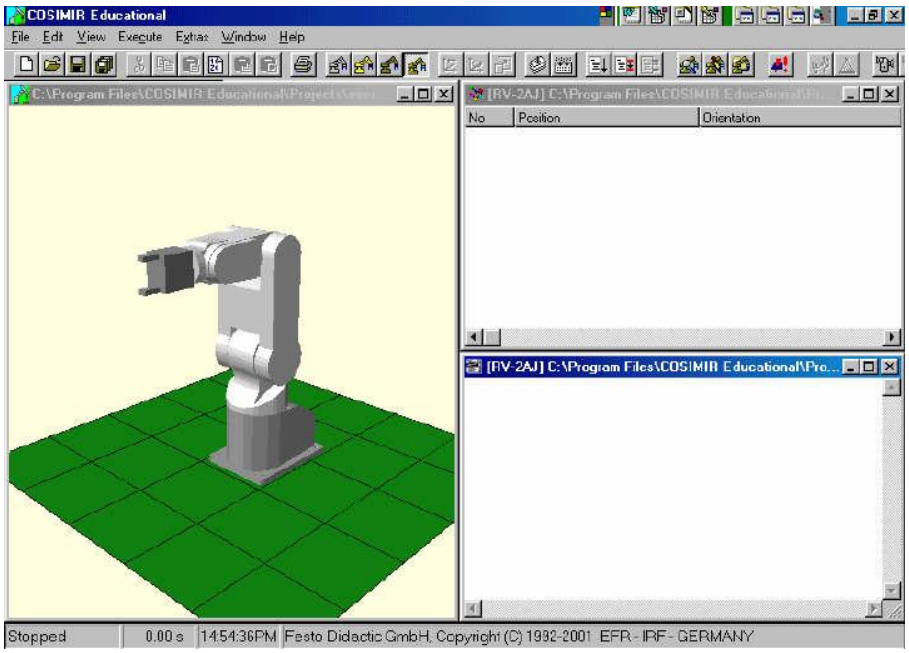
4. เลือกรุ่นของหุ่นยนต์, จำนวนการ์ดอินพุทเอาท์พุท (I/O interface card), จำนวนกริปเปอร์ และ ภาษาที่จะใช้ในการเขียนโปรแกรม

5. หลังจากนั้นกดปุ่ม Finish แล้วจะปรากฏหน้าต่างดังรูป 3.1.2c ซึ่งมีทั้งหน้าต่างโมเดล Position Lists และโปรแกรม โดยในที่นี้เลือกหุ่นยนต์ RV-2AJ

รูป: 3.1.2b



รูป: 3.1.2c



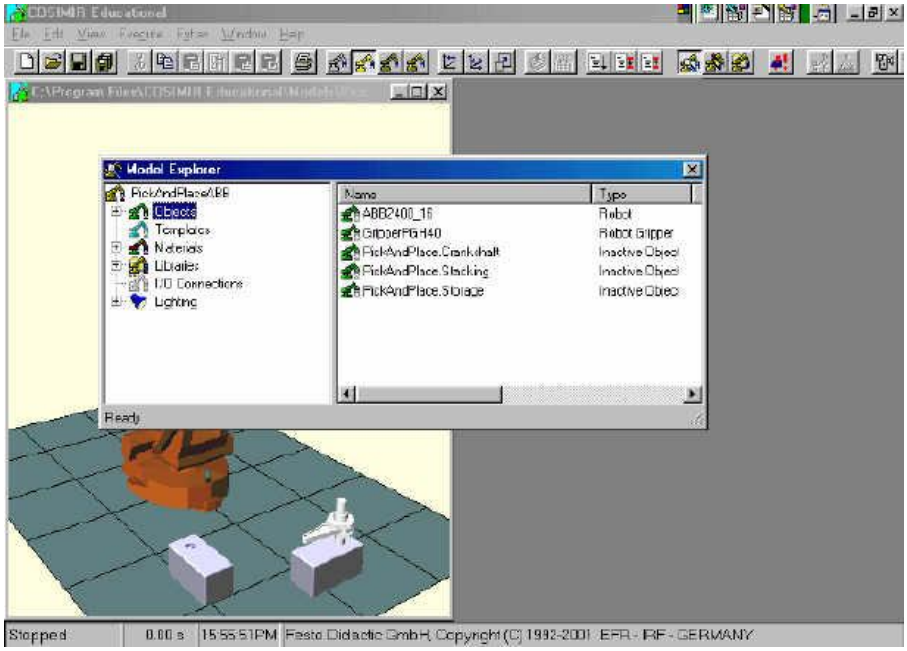
3.1.3 ขั้นตอนการเพิ่มวัตถุอื่น ๆ ในโมเดล

ขั้นตอนนี้เป็นขั้นตอนของการเลือกวัตถุอื่นๆ มาประกอบกับโมเดลเดิมที่มีอยู่ซึ่งมีวิธีการอยู่ 2 วิธี คือ การเรียกวัตถุอื่นจาก Model explorer และโดยการใช้คำสั่ง File > Import

วิธีที่ 1 คือ การเรียกวัตถุอื่น ๆ จาก Model explorer

1. ต้องมีโมเดลพร้อมอยู่แล้ว
2. กดปุ่ม  บนเมนูบาร์หรือใช้คำสั่ง Execute> Model Explorer จะปรากฏหน้าต่างดังรูป 3.1.3a

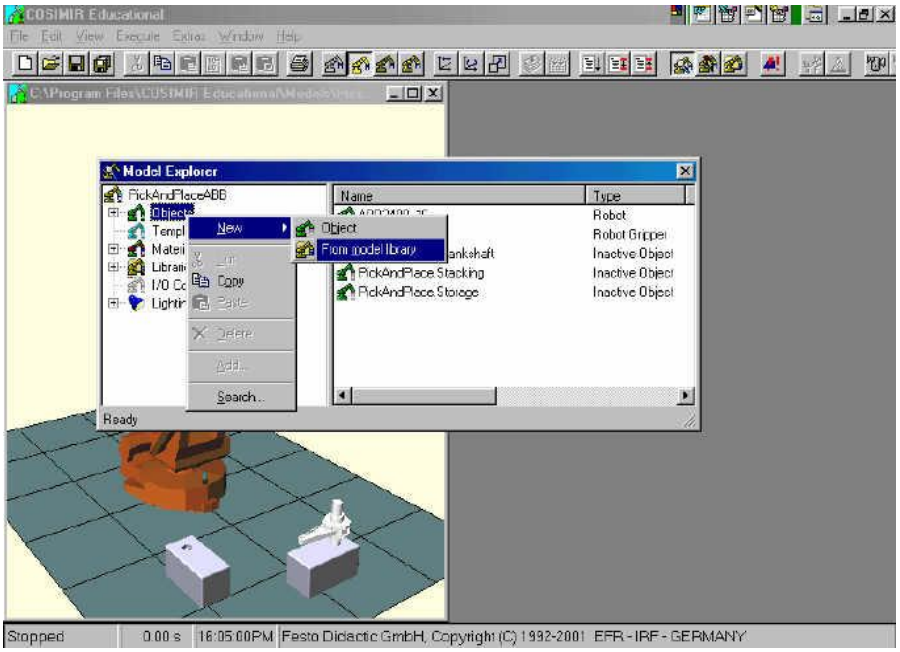
รูป: 3.1.3a



รูป 3.1.3a แสดงให้รายละเอียดภายใน workcell เช่น ขณะนี้ใน workcell ประกอบไปด้วยวัตถุ (object) 5 อย่าง คือ หุ่นยนต์ ABB, กริ๊ปเปอร์ (Gripper) PGH40, ชิ้นงาน (PickAndPlaceCrankshaft), PickAndPlaceStorage และ PickAndPlaceStacking ผู้เรียนสามารถตรวจสอบได้ว่า วัตถุชื่อนี้มีรูปร่างหน้าตาอย่างไรโดยการคลิ๊กที่ชื่อวัตถุนั้น เช่น ABB2400_16 จะปรากฏกรอบล้อมรอบวัตถุนั้นๆ ไว้

3. ทำการเพิ่มวัตถุใน workcell โดยคลิ๊กขวาที่ Object >New> From model library เพื่อเลือกวัตถุจาก model library ดังรูป 3.1.3b

รูป 3.1.3b

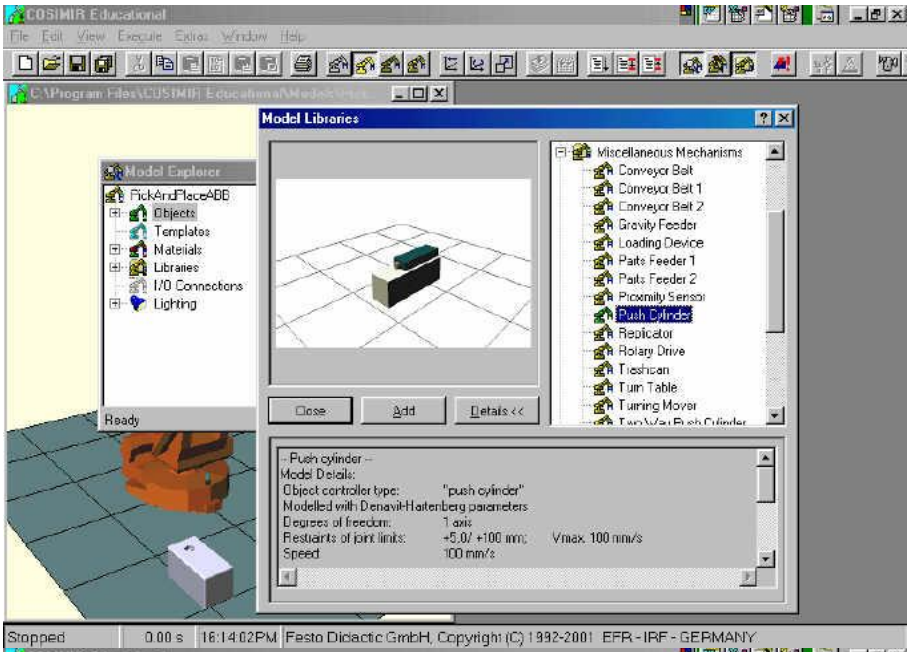


4. หลังจากนั้นจะปรากฏหน้าต่างดังรูป 3.1.3c ซึ่งมีวัตถุแยกตามประเภทต่างๆ ให้เพิ่มไปใน workcell

รูป 3.1.3c

5. เลือกวัตถุที่ต้องการเพิ่มลงไป ตัวอย่าง เช่น ต้องการเพิ่มกระบอกสูบ ให้คลิ๊กที่เครื่องหมายบวกหน้า Miscellaneous Mechanisms หลังจากนั้นเลือกที่ Push Cylinder กดปุ่ม Add ดังรูป 3.1.3d

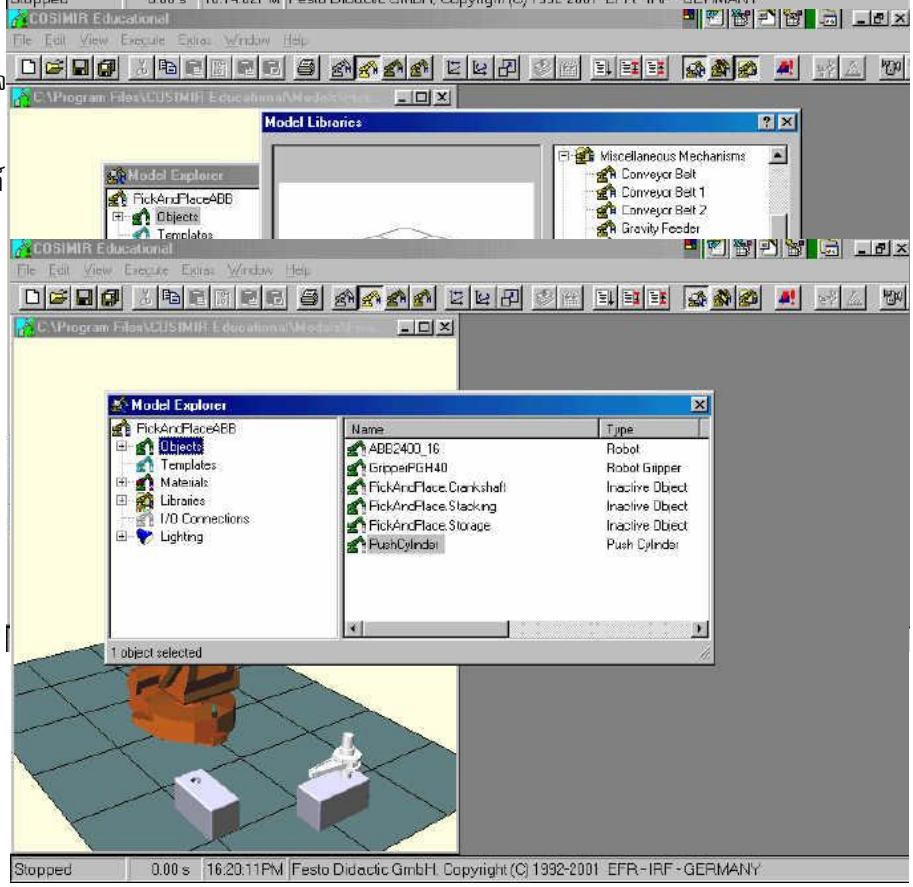
รูป: 3.1.3d



6. หลังจ

จากรูป

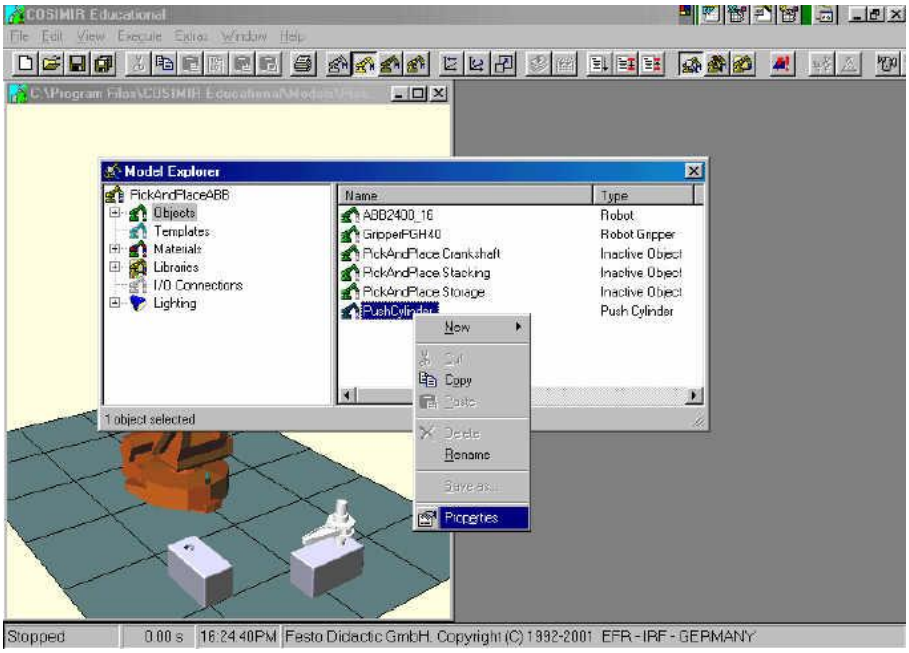
หุ่นยนต์



รูป: 3.1.3e

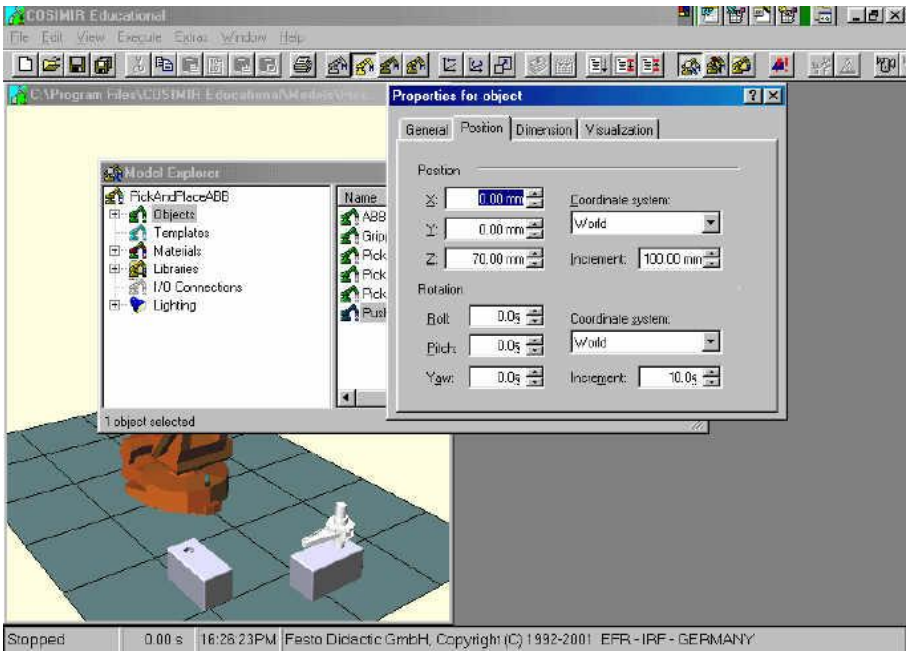
7. ดังนั้นจึงต้องเปลี่ยนตำแหน่งของ Push cylinder โดยคลิ๊กขวาที่ Push cylinder เลือก Properties ดังรูป 3.1.3f

รูป: 3.1.3f



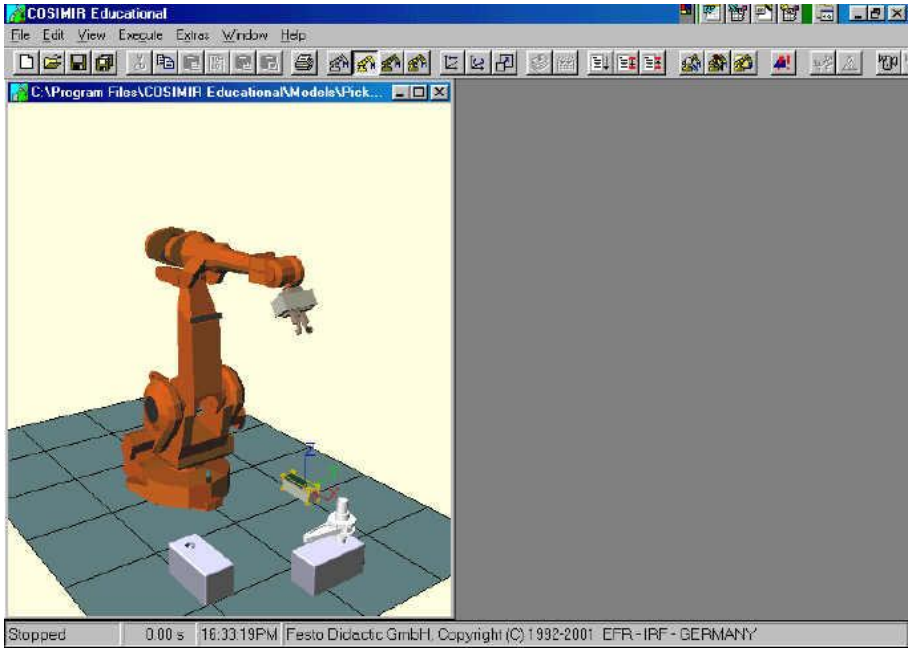
8. หลังจากนั้นจะปรากฏหน้าต่าง “Properties for Object” ซึ่งมีทั้งหมด 4 แท็บ ให้เลือกแท็บ ‘Position’ เพื่อกำหนดตำแหน่งวัตถุ ดังรูป 3.1.3g

รูป: 3.1.3g



9. กำหนดตำแหน่งในแกน x, y และ z เท่ากับ 500 mm, 500 mm และ 70 mm ตามลำดับ จะได้โมเดลดังรูป 3.1.3h

รูป: 3.1.3h



วิธีที่ 2 **คือ การเรียกวัตถุอื่นโดยใช้คำสั่ง** File > Import

1. ต้องมีโมเดลพร้อมอยู่แล้ว
2. ใช้คำสั่ง File > Import
3. หลังจากนั้นเลือกไดเร็กทอรี่ที่ซอฟต์แวร์ Cosimir ติดตั้งอยู่ ในที่นี้คือ C:\Program Files\ Cosimir Education\ Models\ MiscLibs
4. เลือกวัตถุที่จะนำมาเพิ่มในโมเดล หลังจากที่เลือกวัตถุแล้วจะปรากฏชื่อวัตถุใน Model explorer
5. ปรับตำแหน่งของวัตถุเช่นเดียวกันกับในแบบที่ 1 ข้อ 8

หมายเหตุ: วิธีแต่ละวิธีมีวัตถุให้เลือกต่างกัน

3.2 ขั้นตอนการกำหนดตำแหน่งในการเคลื่อนที่

เมื่อได้โมเดลเรียบร้อยแล้ว ขั้นตอนต่อไปคือการกำหนดตำแหน่งต่างๆ ให้กับหุ่นยนต์ซึ่งมีขั้นตอนดังต่อไปนี้ คือ

1. เปิดหน้าต่างสำหรับบันทึกตำแหน่งโดยใช้คำสั่ง File> New จะปรากฏหน้าต่าง “New” ให้เลือก MRL Position List หรือ Position Lists (COSIMIR position lists) ดังรูป 3.2.1a ถ้าผู้เรียนสร้างโมเดลเองให้ข้ามขั้นตอนที่ 1 และ 2 ได้เลย

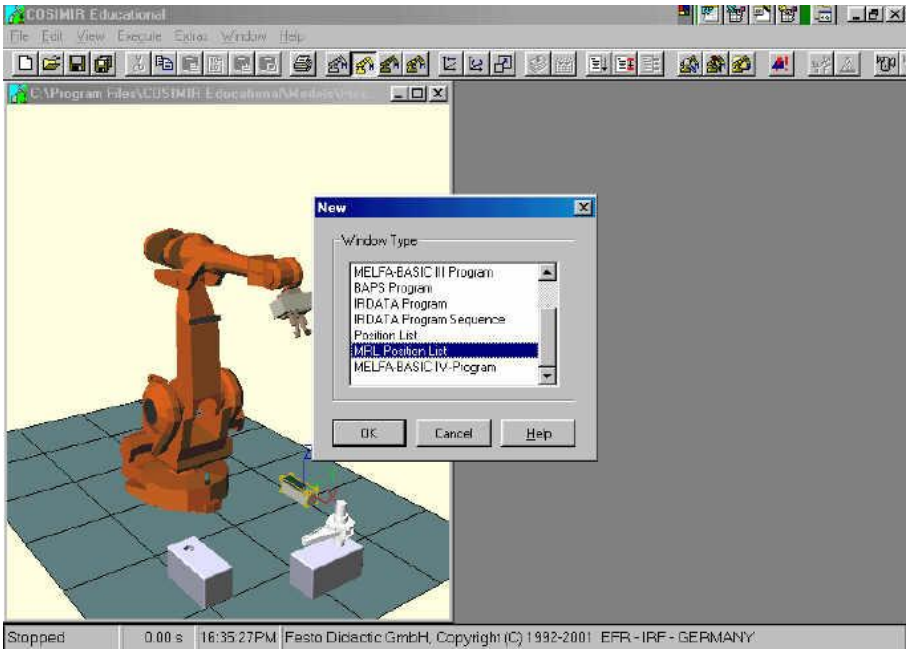
   หมายเหตุ: MRL Position Lists มีนามสกุลเป็น .POS และ Position Lists มีนามสกุลเป็น .PSL ซึ่งทั้งสองไฟล์สามารถบันทึกตำแหน่งได้แต่ในกรณีที่ผู้เรียนเลือกใช้ Position Lists ก่อนทำการคอมไพล์กับโปรแกรมต้องเปลี่ยนเป็น MRL Position lists เพื่อให้เข้ากับภาษา MRL และ Melfa Basic IV

   MRL Position Lists และ Position Lists มีคุณสมบัติต่างกันดังต่อไปนี้ คือ

a) **ชื่อตำแหน่งใน** Position Lists **สามารถเป็นตัวหนังสือหรือตัวเลขก็ได้ เช่น** P901 **หรือ** 901 **ในขณะที่ชื่อตำแหน่งใน** MRL Position Lists **ต้องเป็นตัวเลขเท่านั้น เช่น** 901 **เป็นต้น**

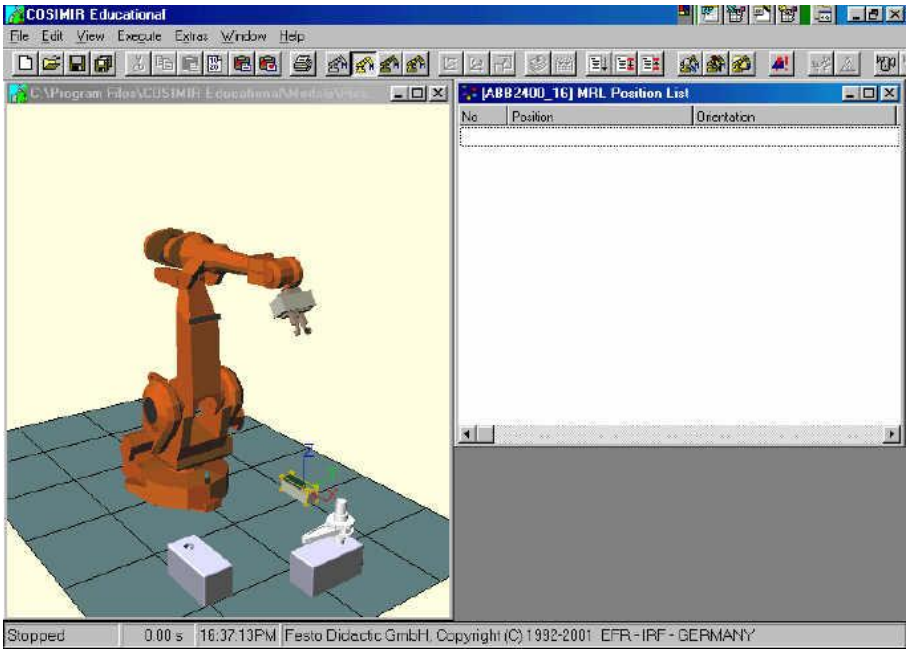
b) Position Lists สามารถระบุตำแหน่งโดยใช้ world coordinate joint coordinate หรือ MRL-5 joint coordinate นอกจากนี้ยังสามารถอ้างถึงตำแหน่งโดยการอิงจากวัตถุ (Relative to object หรือ object coordinate) ซึ่งคุณสมบัตินี้ทำให้เราสามารถปรับตำแหน่งใน Position Lists โดยอิงจากการเปลี่ยนแปลงของวัตถุนั้นๆ ใน workcell แต่ในส่วนของ MRL Position Lists ระบุตำแหน่งโดย world coordinate เท่านั้น โดยรวมแล้ว Position Lists เป็นเครื่องมือสำหรับการกำหนดตำแหน่งต่างๆ ซึ่งทำได้ง่ายกว่า MRL Position Lists เพราะมีคุณสมบัติอื่นๆ เพิ่มมากขึ้น อย่างไรก็ตาม เมื่อทดสอบตำแหน่งต่างๆ เรียบร้อยใน Position Lists เรียบร้อยแล้วให้ทำการเปลี่ยนเป็น MRL position lists

รูป 3.2.1a



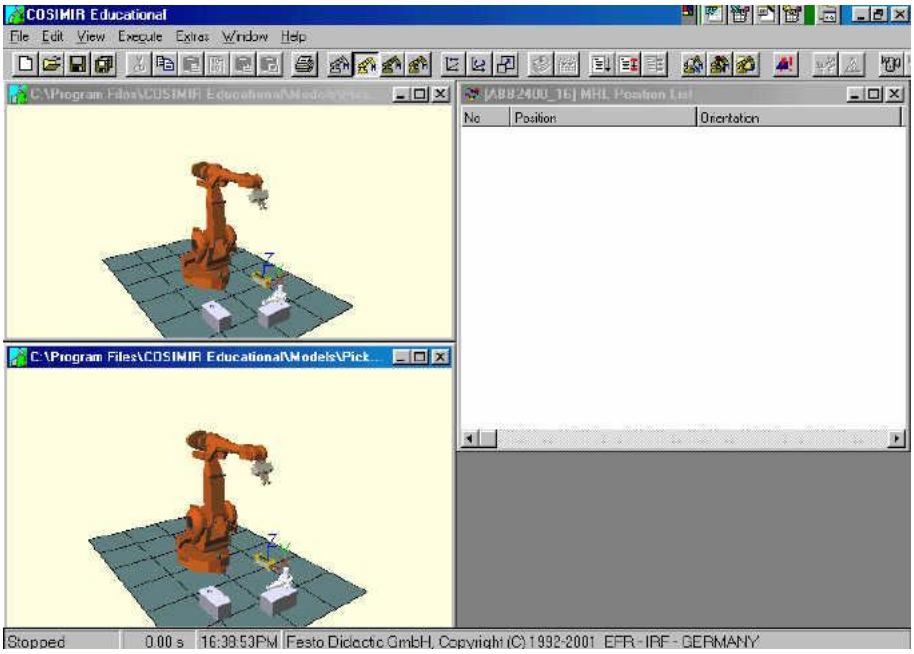
2. หลังจากกดปุ่ม OK จะปรากฏหน้าต่างดังรูป 3.2.1b

รูป 3.2.1b



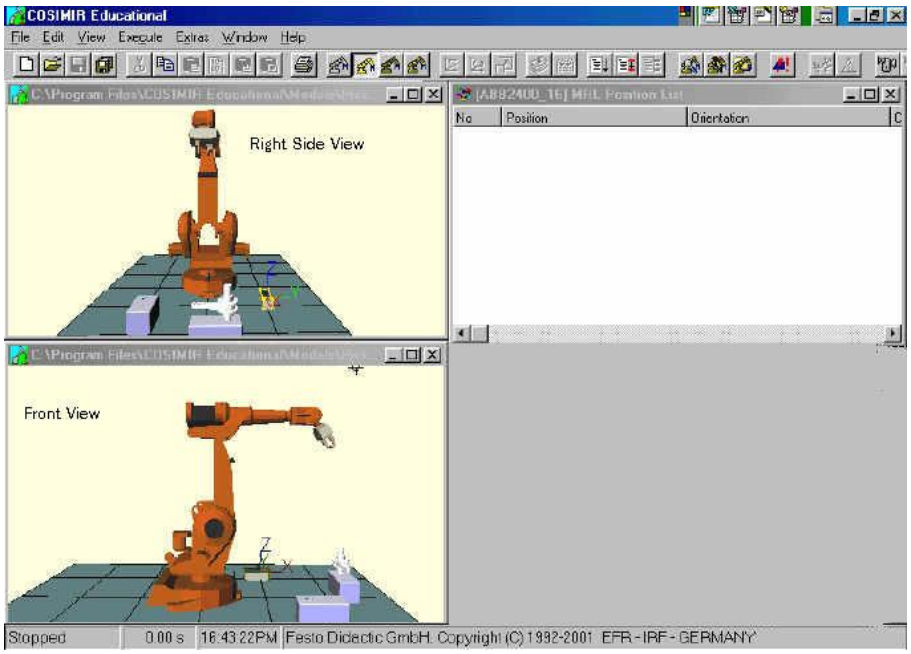
3. เนื่องจากการกำหนดตำแหน่งของหุ่นยนต์ในซอฟต์แวร์มีข้อจำกัด คือ อาจจะมีมิติความลึกที่ไม่เหมือนกับการมองหุ่นยนต์ของจริง ดังนั้นจึงขอแนะนำให้ใช้ workcell เพิ่มอีกหนึ่งหน้าต่างโดยใช้คำสั่ง View > New เพื่อให้เห็นตำแหน่งต่างๆ ได้ดีขึ้น โดยเฉพาะอย่างยิ่งในขณะที่หยิบชิ้นงาน ดังรูป 3.2.1c

รูป: 3.2.1c



4. ปรับหน้าต่างแรกให้เป็น Right side view และหน้าต่างที่สองให้เป็น Front view ดังรูป 3.2.1d

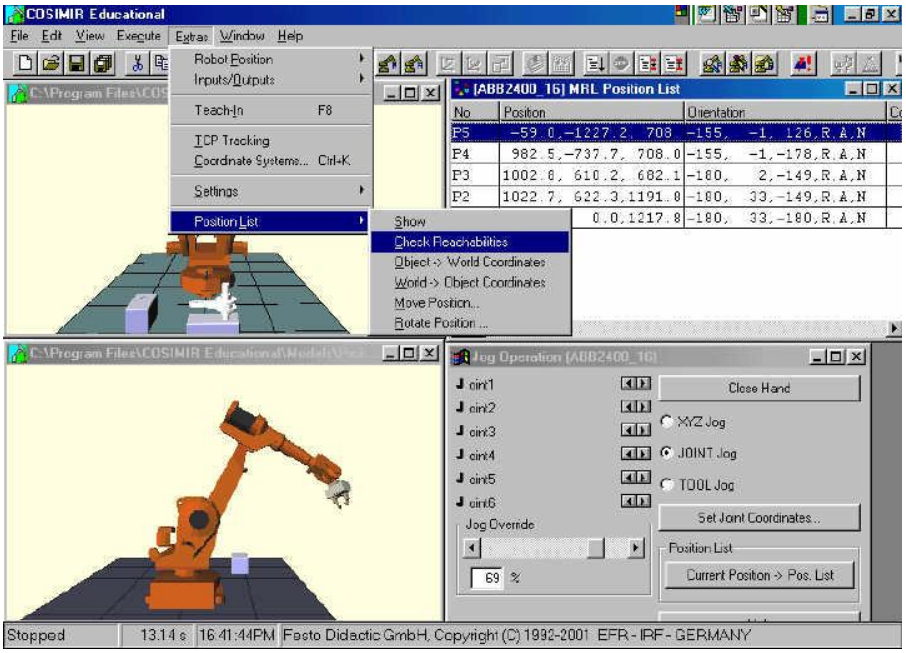
รูป: 3.2.1d



5. เรียก Teach-In Box ขึ้นมาเพื่อกำหนดตำแหน่ง (วิธีการกำหนดตำแหน่งให้ดูจากบทที่ 2)

6. หลังจากที่มีการโหลดและสร้างอุปกรณ์ต่างๆ ใน workcell และกำหนดตำแหน่งให้กับหุ่นยนต์แล้ว ดังนั้นก่อนที่จะทำการจำลองการทำงานควรที่จะมีการตรวจสอบว่าตำแหน่งที่กำหนดให้กับหุ่นยนต์นั้นอยู่ใน workspace ของหุ่นยนต์หรือไม่ เพื่อป้องกันความผิดพลาดในระหว่างจำลองการทำงาน การตรวจสอบว่าตำแหน่งใดๆ อยู่ใน workspace สามารถทำได้โดยเลือกที่ตำแหน่งที่ต้องการตรวจสอบ แล้วใช้คำสั่ง Extras > Position List > Check Reachability สามารถตรวจสอบได้ครั้งละหนึ่งตำแหน่ง ดังรูป 3.2.1e หลังจากการตรวจสอบ ตำแหน่งที่ไม่อยู่ใน workspace จะปรากฏเป็นสีเทา

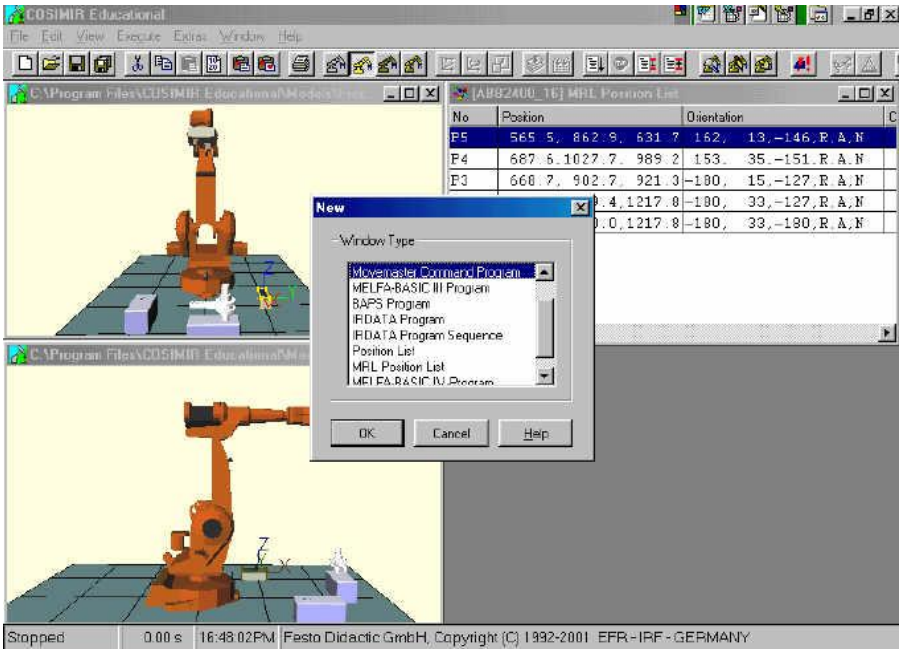
รูป: 3.2.1e



3.3 ขั้นตอนการเขียนโปรแกรม

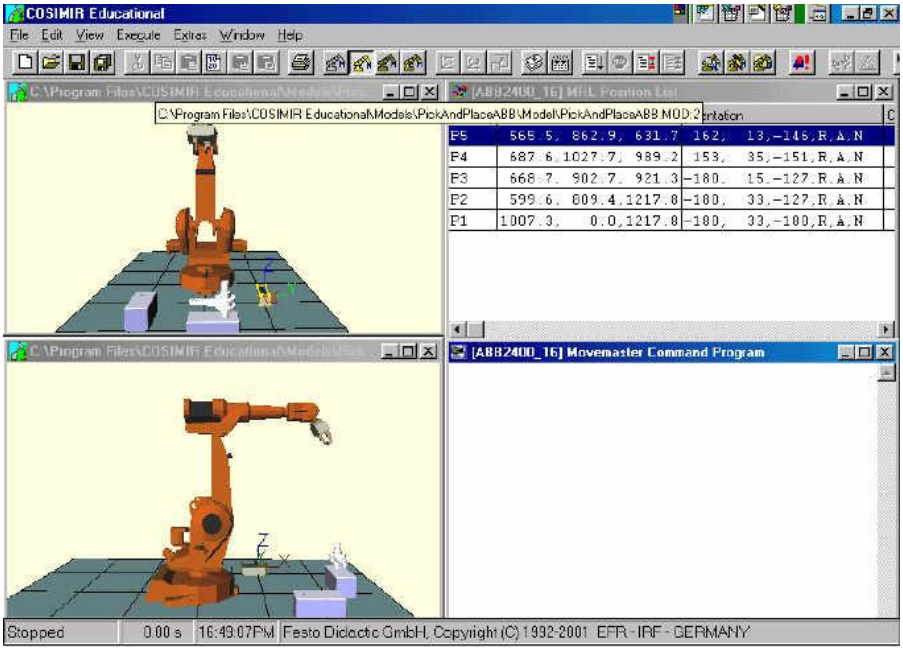
1. เปิดหน้าต่างสำหรับเขียนโปรแกรมโดยใช้คำสั่ง File > New จะปรากฏหน้าต่าง dialog box ดังรูป 3.3.1a เลือกภาษาที่จะใช้เขียนคือ Movemaster Command Program หรือ Melfa Basic IV

รูป: 3.3.1a



2. กดปุ่ม OK จะปรากฏหน้าต่างสำหรับเขียนโปรแกรม ดังรูป 3.3.1b

รูป: 3.3.1b



3. วางแผนในการเขียนโปรแกรม

วิธีการเขียนโปรแกรมอย่างเป็นระบบเริ่มจาก

1) เข้าใจ (โจทย์) ปัญหา
2) วิเคราะห์ปัญหา: ศึกษาปัญหาโดยแบ่งเป็นปัญหาย่อยๆ และหาข้อมูลขาเข้า (input) และข้อมูลขาออก (output) ของปัญหา
3) ออกแบบขั้นตอนการเขียนโปรแกรม (algorithm)
4) พัฒนาโปรแกรม: แปลงขั้นตอนการแก้ปัญหาเป็นภาษา MRL หรือ Melfa Basic IV
5) ทดสอบโปรแกรม: เมื่อทำการตรวจสอบความถูกต้องของโปรแกรมแล้วให้บันทึกโปรแกรม

ในกรณีที่เลือกใช้ภาษา MRL ใช้คำสั่ง compile+Link บน tool bar เพื่อแปลงโปรแกรมเป็น IRDATA Code หลังจากนั้นใช้คำสั่ง Start บน tool bar เพื่อรันโปรแกรม

สำหรับกรณีของภาษา Melfa Basic IV จะมีขั้นตอนของ Project Management ที่เพิ่มเติมขึ้นมา Project management ช่วยในเรื่องการจัดการไฟล์โปรแกรมซึ่งใช้ภาษาในการเขียนที่ต่างกันกล่าวคือ ในหนึ่งโปรเจกท์ผู้เรียนสามารถเลือก Melfa Basic IV ในการรันโปรแกรมในครั้งแรก และในครั้งถัดไปสามารถเลือกภาษาที่ต่างไปจาก Melfa Basic IV ได้ Project Management สามารถรองรับภาษา V+, RAPID, KRL และ Melfa Basic IV แต่ไม่รองรับภาษา MRL

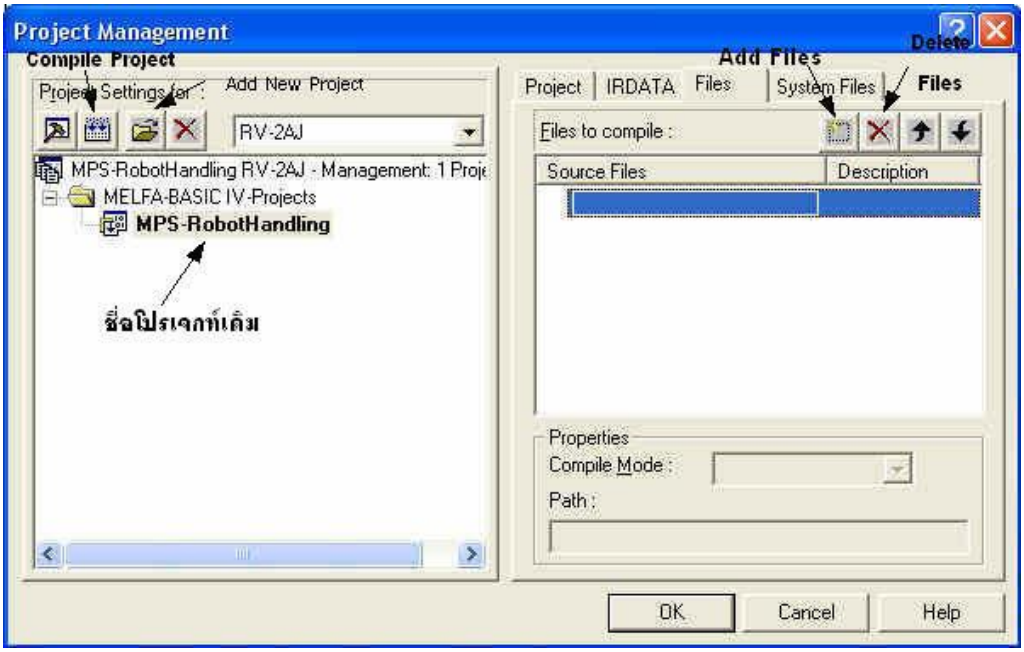
ขั้นตอนของ Project Management มีดังต่อไปนี้

a) ใช้คำสั่ง Execute > Project Management จะปรากฏหน้าจอดังตัวอย่างในรูป 3.3.1c
b) กดปุ่ม 'Add New Project' แล้วเลือกไดเร็กทอรี่ที่จะบันทึกโปรเจกท์ดังตัวอย่างในรูป 3.3.1d

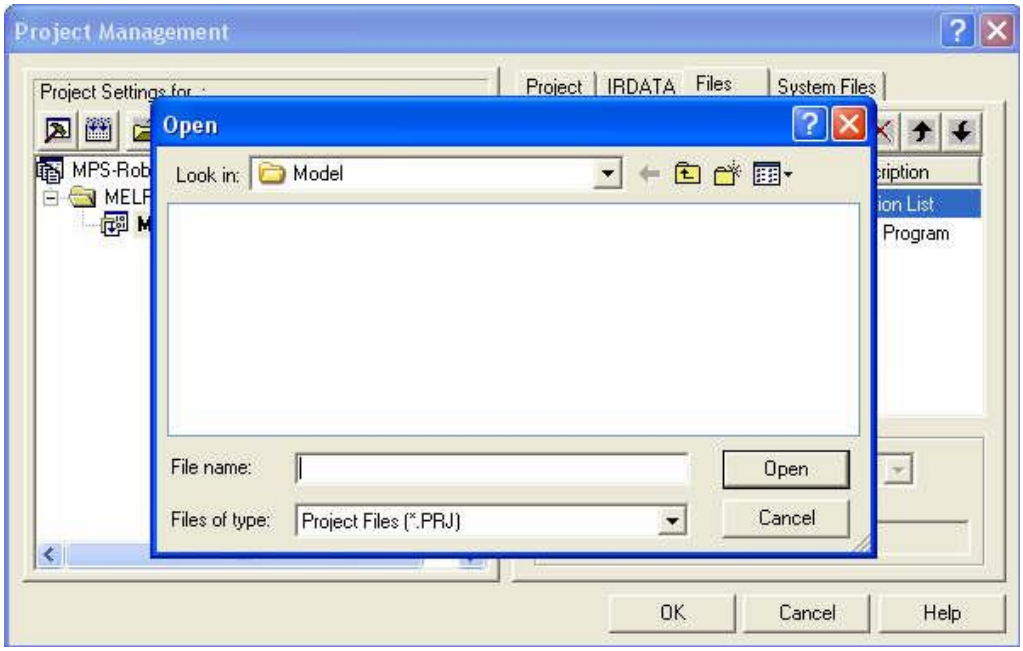
c) เลือกแท็บ ‘Files’ แล้วใส่ ‘Source Files’ คือ โปรแกรมและ Position Lists ให้กับโปรเจกท์ โดยใช้ปุ่ม ‘Add Files’ เพื่อเลือกไฟล์โปรแกรมและ Position Lists ที่ได้บันทึกไว้ เสร็จแล้วจะได้ดังตัวอย่างในรูป 3.3.1e และเลือกโปรแกรมจาก compile mode เป็น main program

d) ทำการคอมไพล์โปรเจกท์โดยใช้คำสั่ง Compile Project

e) รันโปรแกรมโดยใช้คำสั่ง Start บน tool bar

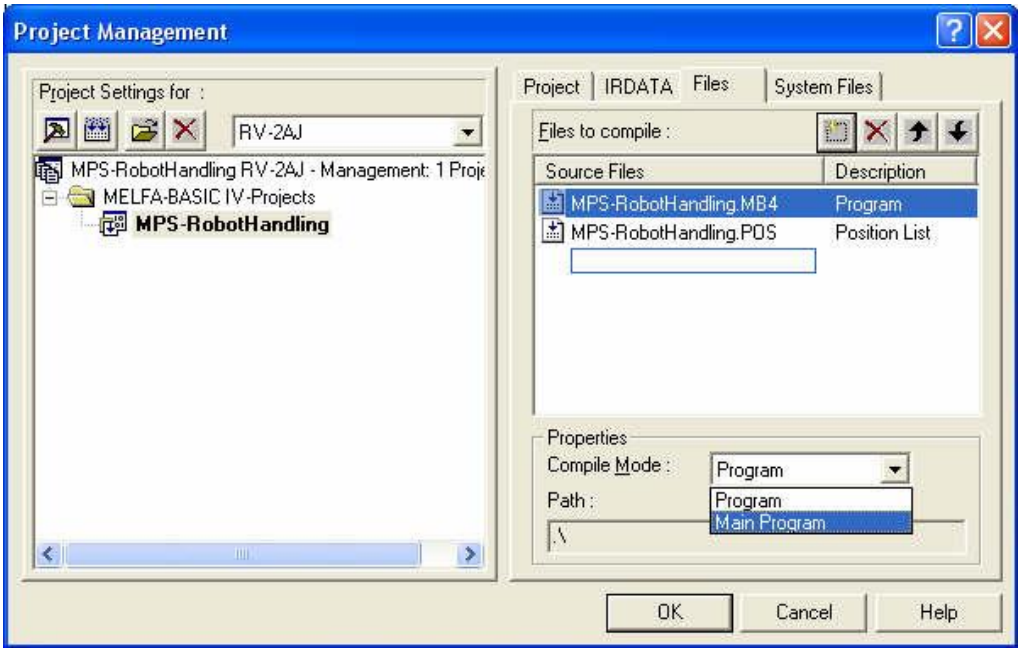


รูป 3.3.1c:
Project Management
Dialog Box



รูป: 3.3.1d

รูป: 3.3.1e



ตัวอย่างที่ 3.1: **สาธิตวิธีการเขียนโปรแกรม**

จงเขียนโปรแกรมควบคุมหุ่นยนต์โดยใช้ Movemaster Command Program ให้หุ่นยนต์เคลื่อนที่ไปหยิบชิ้นงานขึ้นมาจาก Storage หมายเหตุ: ใช้โมเดล PickAndPlaceKuka และใช้ตำแหน่งที่กำหนดในตัวอย่างที่ 2.1 (ชื่อไฟล์: exercise 2_1.POS)

ขั้นตอนที่ 1: เข้าใจปัญหา

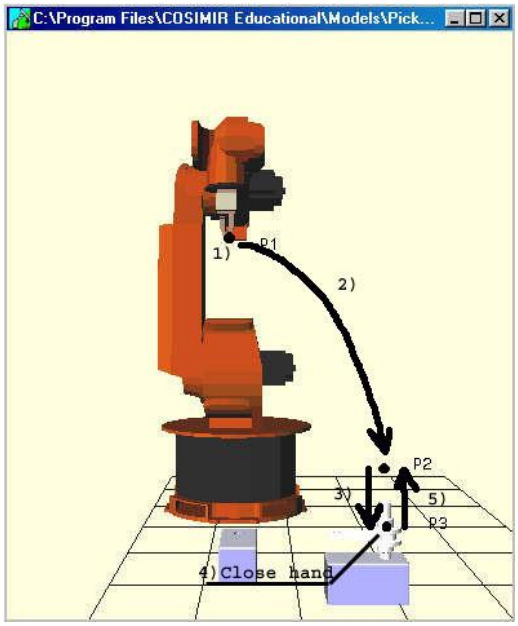
ขั้นตอนที่ 2: วิเคราะห์ปัญหา

เนื่องจากปัญหานี้ยังไม่มีความซับซ้อนเท่าใดนัก คือ หุ่นยนต์ทำงานตามลำดับขั้นตอน ดังนั้นจึงสามารถใช้เทคนิค sequential programming คือการทำงานแบบเป็นลำดับขั้นตอน (step-by-step execution) ซึ่งเป็นเทคนิคเบื้องต้นสำหรับผู้เริ่มต้น

ขั้นตอนที่ 3: ออกแบบ

ออกแบบขั้นตอน เส้นทางการเคลื่อนที่และการทำงานที่ตำแหน่งต่างๆ ซึ่งมีลำดับขั้นตอนหลักดังนี้

1) เคลื่อนที่ไปตำแหน่งที่ 1
2) เคลื่อนที่ไปตำแหน่งที่ 2 (ตามแนวแกน)
3) เคลื่อนที่ไปตำแหน่งที่ 3 (แบบเส้นตรง)
4) หยิบชิ้นงาน
5) เคลื่อนที่ไปตำแหน่งที่ 2 (แบบเส้นตรง)



รูป 3.3.2a: ตัวอย่างการออกแบบลำดับขั้นตอนการทำงานและเส้นทางการเคลื่อนที่

หลังจากเรียงลำดับขั้นตอนหลักแล้วก็พิจารณาว่าสมควรมีขั้นตอนย่อยหรือไม่ ในกรณีที่ขั้นตอนหลักอธิบายขั้นตอนการทำงานไว้ครบถ้วนก็ไม่จำเป็นต้องมีขั้นตอนย่อย ขั้นตอนที่ 1 และ 2 มีความชัดเจนอยู่แล้ว ขั้นตอนที่ต้องเพิ่มรายละเอียดมีดังต่อไปนี้

ลำดับที่ 3 เคลื่อนที่ไปตำแหน่งที่ 3

3.1 หน่วงเวลา 1 วินาที

ลำดับที่ 4 หยิบชิ้นงาน

4.1 หน่วงเวลา 1 วินาที

ดังนั้น ลำดับขั้นตอนทั้งหมด คือ

1. เคลื่อนที่ไปตำแหน่งที่ 1
2. เคลื่อนที่ไปตำแหน่งที่ 2 (ตามแนวแกน)
3. เคลื่อนที่ไปตำแหน่งที่ 3 (แบบเส้นตรง)

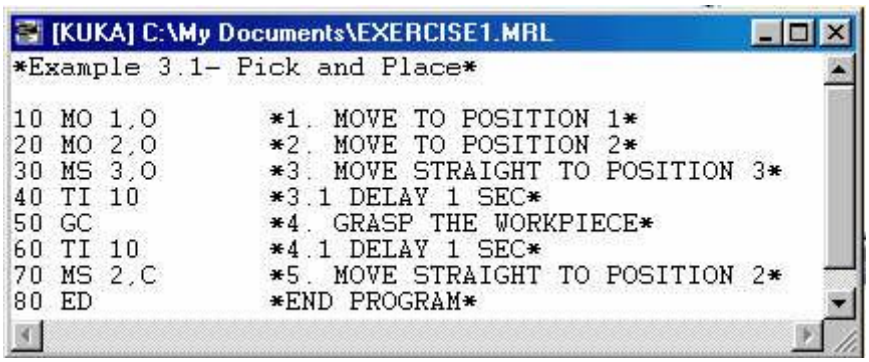
   3.1 หน่วงเวลา 1 วินาที
4. หยิบชิ้นงาน

   4.1 หน่วงเวลา 1 วินาที
5. เคลื่อนที่ไปตำแหน่งที่ 2 (แบบเส้นตรง)

ขั้นตอนที่ 4: พัฒนาโปรแกรม

ในขั้นตอนการพัฒนาโปรแกรม ให้นำขั้นตอนทั้งหมดมาเขียนเป็นภาษา Movemaster command program ดังรูป 3.3.2b

รูป: 3.3.2b



ขั้นตอนที่ 5: ทดสอบโปรแกรม

ตรวจสอบผลการทำงานของโปรแกรมโดยจำลองการทำงาน ถ้าผลการทำงานไม่ตรงตามต้องการให้ทบทวนแก้ไขโปรแกรมและตรวจสอบจนแน่ใจว่าการทำงานเป็นไปตามที่ปัญหากำหนดไว้

เมื่อจบตัวอย่างที่ 3.1 ผู้เรียนสามารถทำแบบฝึกหัดข้อ 3.1 ถึง 3.3 ในหน้าที่ 30 ได้

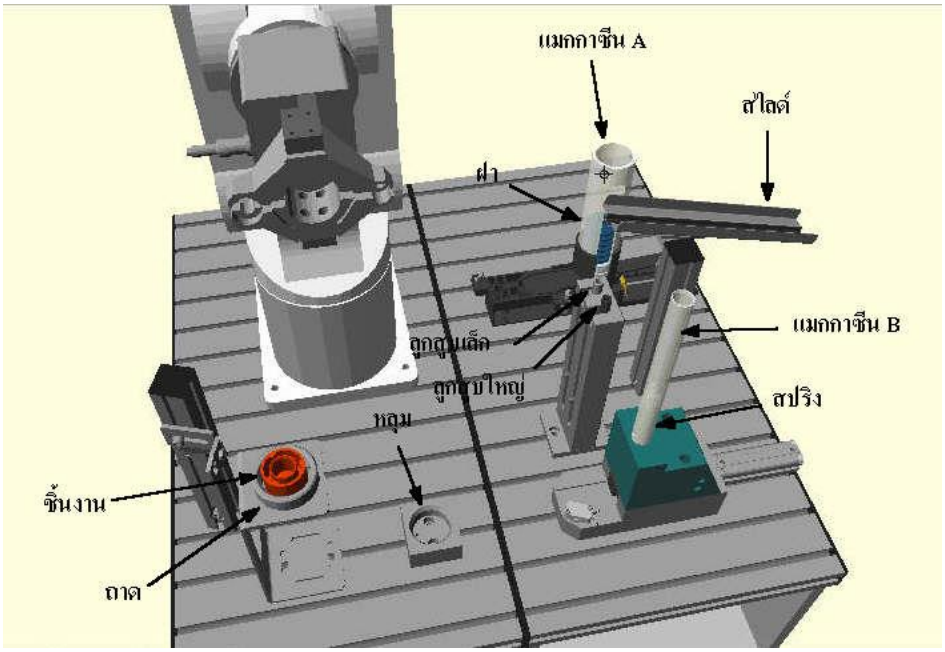
ตัวอย่างที่ 3.2: I/O connection **และ** structured programming

จงเขียนโปรแกรมควบคุมหุ่นยนต์โดยใช้ภาษา **Melfa Basic IV** ให้หุ่นยนต์ประกอบชิ้นงาน (Housing) เข้ากับฝา ใช้โมเดล MPS RobotAssembly ดังรูป 3.3.3a และใช้ตำแหน่งและอินพุทเอาท์พุท จากไฟล์ตัวอย่างดังตารางที่ 3.3.1 และ 3.3.2 ตามลำดับ

หมายเหตุ: เมื่อเรียกหน้าต่าง workcell ขึ้นมาแล้วจะพบว่าไม่มีชิ้นงาน ดังนั้นให้เรียกชิ้นงานในขณะที่หน้าต่าง workcell ทำงานอยู่ โดยใช้คำสั่ง File > Import หลังจากนั้นเลือก Program files > COSIMIR Educational > Model > MPS-RobotAssembly > Import มีชิ้นงานให้เลือกสามสี คือ HousingBlack, HousingRed และ HousingSilver เลือกสีใดสีหนึ่งเท่านั้น

ตารางที่ 3.3.1: ตารางแสดงตำแหน่งต่างๆ พร้อมคำอธิบาย

ตำแหน่ง	คำอธิบาย
901	ตำแหน่งเริ่มต้น (Home)
902	ตำแหน่งที่หยิบชิ้นงานในถาด (Tray)
903	ตำแหน่งที่วางชิ้นงานในหลุม (socket)
904	ตำแหน่งที่หยิบลูกสูบสีเทา (ขนาดเล็ก)
905	ตำแหน่งที่ห่างจากลูกสูบสีเทามาทางซ้ายมือ
906	ตำแหน่งที่วางลูกสูบสีเทาลงในชิ้นงาน
907	ตำแหน่งที่หยิบลูกสูบสีดำ (ขนาดใหญ่)
908	ตำแหน่งที่ห่างจากลูกสูบสีดำมาทางซ้ายมือ
909	ตำแหน่งที่วางลูกสูบสีดำลงในชิ้นงาน
910	ตำแหน่งที่หยิบสปริง
911	ตำแหน่งที่วางสปริงลงในชิ้นงาน
912	ตำแหน่งที่หยิบฝา
913	ตำแหน่งที่วางฝาลงบนชิ้นงาน
914	ตำแหน่งที่ประกอบฝากับชิ้นงาน
915	ตำแหน่งเหนือสไลด์ (slide)



รูป 3.3.3a

ตารางที่ 3.3.2: **ตารางแสดงอินพุทและเอาท์พุทของ** workcell **พร้อมคำอธิบาย**

อินพุท	คำอธิบาย	เอาท์พุท	คำอธิบาย
0	ตรวจสอบว่ากริปเปอร์อยู่ในสภาวะปิด	0	ปิดกริปเปอร์
1	ตรวจสอบว่ามีฝาอยู่ในแมกกาซีน A	1	ให้ฝาชิ้นถัดไปพร้อมที่จะถูกเลื่อนออก
2	ตรวจสอบว่ามีลูกสูบขนาดใหญ่	2	ให้กระบอกสูบเลื่อนฝาออกมา
3	ตรวจสอบว่ามีลูกสูบขนาดเล็ก	3	ให้ลูกสูบขนาดใหญ่ชิ้นถัดไปพร้อมที่จะถูกเลื่อนออก
4	ตรวจสอบว่ามีสปริงอยู่ในแมกกาซีน B	4	ให้ลูกสูบขนาดเล็กชิ้นถัดไปพร้อมที่จะถูกเลื่อนออก
5	ตรวจสอบว่ามีชิ้นงานเป็นสีดำ	5	ให้สปริงชิ้นถัดไปพร้อมที่จะถูกเลื่อนออก
6	ตรวจสอบว่ามีสปริง	6	ให้กระบอกสูบเลื่อนสปริงออกมา
7	ตรวจสอบว่าฝาถูกกระบอกสูบเลื่อนออกมาแล้ว		
8	ตรวจสอบว่าสปริงถูกกระบอกสูบเลื่อนออกมาแล้ว		
9	ตรวจสอบว่ามีชิ้นงานอยู่ในถาด		

ขั้นตอนที่ 1: เข้าใจปัญหา

ขั้นตอนที่ 2: วิเคราะห์ปัญหา

เนื่องจากปัญหานี้ค่อนข้างมีความซับซ้อน ดังนั้นเพื่อลดความซับซ้อนจึงนำเทคนิคการเขียนโปรแกรมแบบ Structured programming มาใช้เพื่อแบ่งเป็นปัญหาย่อยๆ คือ โปรแกรมหลัก (main program) 1 โปรแกรมเพื่อควบคุมลำดับการทำงานของโปรแกรมย่อย และโปรแกรมย่อย (subprogram) 3 โปรแกรม คือ โปรแกรมรีเซ็ทสู่สภาวะพร้อมทำงาน (initailize) โปรแกรมหยิบชิ้นงานจากถาดมาวางไว้ที่หลุม โปรแกรมประกอบฝาเข้ากับชิ้นงาน

ขั้นตอนที่ 3: ออกแบบ

โปรแกรมรีเซ็ทสู่สภาวะเริ่มต้น

ลำดับขั้นตอน:

1. กำหนดความเร็ว
2. เคลื่อนที่สู่ตำแหน่งเริ่มต้น
3. เปิดกริปเปอร์
4. กลับสู่โปรแกรมหลัก

โปรแกรมหยิบชิ้นงานจากถาดวางไว้ที่หลุม

1. ตรวจสอบว่ามีชิ้นงานที่ถาดหรือไม่
2. เคลื่อนที่มาตำแหน่งเหนือชิ้นงาน
3. เคลื่อนที่มาตำแหน่งที่หยิบชิ้นงาน
4. ปิดกริปเปอร์
5. เคลื่อนที่เหนือตำแหน่งที่หยิบชิ้นงาน
6. เคลื่อนที่เหนือตำแหน่งที่วางชิ้นงานในหลุม
7. เคลื่อนที่มาตำแหน่งที่วางชิ้นงาน
8. เปิดกริปเปอร์

9. เคลื่อนที่เหนือตำแหน่งที่วางชิ้นงาน
10. กลับสู่โปรแกรมหลัก

โปรแกรมประกอบฝาเข้ากับชิ้นงาน

1. เคลื่อนที่เหนือตำแหน่งที่หยิบฝา
2. ให้กระบอกสูบเลื่อนฝาออกมา
3. ตรวจสอบว่าฝาถูกเลื่อนออกแล้ว
4. ให้กระบอกสูบเคลื่อนที่เข้า
5. ให้ฝาชิ้นถัดไปพร้อมที่จะถูกเลื่อนออก
6. เคลื่อนที่มาตำแหน่งที่หยิบฝา
7. ปิดกริปเปอร์
8. เคลื่อนที่มาตำแหน่งเหนือฝา
9. เคลื่อนที่มาตำแหน่งที่เหนือชิ้นงาน
10. เคลื่อนที่มาตำแหน่งที่วางฝาบนชิ้นงาน
11. เคลื่อนที่หมุนฝาเพื่อประกอบเข้ากับชิ้นงาน
12. เปิดกริปเปอร์
13. เคลื่อนที่มาตำแหน่งที่เหนือชิ้นงาน
14. กลับสู่โปรแกรมหลัก

โปรแกรมหลัก

1. โปรแกรมย่อย 'รีเซ็ทสู่สภาวะเริ่มต้น'
2. โปรแกรมย่อย 'หยิบชิ้นงานจากถาดวางไว้ที่หลุม'
3. โปรแกรมย่อย 'ประกอบฝาเข้ากับชิ้นงาน'
4. เคลื่อนที่กลับสู่ตำแหน่งเริ่มต้น
5. จบโปรแกรม

ขั้นตอนที่ 4: พัฒนาโปรแกรม

ในขั้นตอนการพัฒนาโปรแกรม ให้นำขั้นตอนทั้งหมดมาเขียนเป็นภาษา Melfa Basic IV ดังรูป 3.3.3b

```
10 '*************************************************************************************************
20 '***EXAMPLE 3.2: I/O COMMUNICATION AND STRUCTURED PROGRAMMING***
30 '*************************************************************************************************

40 '*MAIN PROGRAM*
50 GOSUB 110                    'INITIALIZE
60 GOSUB 160                    'ASSEMBLE HOUSING
70 GOSUB 370                    'ASSEMBLE CAP
80 MOV P901                     'HOME POSITION
90 END                          'END PROGRAM
```

```
100 '*INITIALIZE*
110 OVRD 70                     'SPEED
120 MOV P901                    'HOME POSITION
130 M_OUT(0)=0                  'GRIPPER OPEN
140 RETURN                      'RETURN TO MAIN PROGRAM

150 '*ASSEMBLE HOUSING*
160 DLY 1
170 WAIT M_IN(9)=1              'WAIT UNTIL HOUSING IN TRAY
180 SPD 60
190 MOV P902,-50                'OVER TRAY
200 SPD 30
210 MVS P902                    'TO PICK
220 DLY 1
230 M_OUT(0)=1                  'GRIPPER CLOSE
240 DLY 1
250 MVS P902,-50                'MOVE UP
260 SPD 60
270 MOV P903,-250               'OVER SOCKET
280 SPD 45
290 MVS P903                    'TO PLACE
300 DLY 1
310 M_OUT(0)=0                  'GRIPPER OPEN
320 DLY 1
330 MVS P903,-50                'MOVE UP
340 RETURN                      'RETURN

350 '*ASSEMBLE CAP*
360 SPD 60
370 MOV P912,-250               'OVER PICK
380 M_OUT(2)=1                  'PUSHER OUT
390 WAIT M_IN(7)=1              'WAIT UNTIL CAP IS MOVED OUT
400 M_OUT(2) =0                 'PUSHER IN
```

```
410 M_OUT(1)=1                  'CAP NEXT PART
420 SPD 45
430 MVS P912                    'TO PICK
440 DLY 1
450 M_OUT(0)=1                  'GRIPPER CLOSE
460 DLY 1
470 MVS P912,-50                'MOVE UP
480 SPD 60
490 MOV P913,-250               'OVER PLACE
500 SPD 45
510 MVS P913                    'TO PLACE
520 MVS P914                    'ASSEMBLE
530 M_OUT(0)=0                  'GRIPPER OPEN
540 MVS P914,-50                'MOVE UP
550 RETURN                      'RETURN
```

รูป 3.3.3b: โปรแกรมประกอบชิ้นงานเข้ากับฝา

ขั้นตอนที่ 5: ทดสอบโปรแกรม

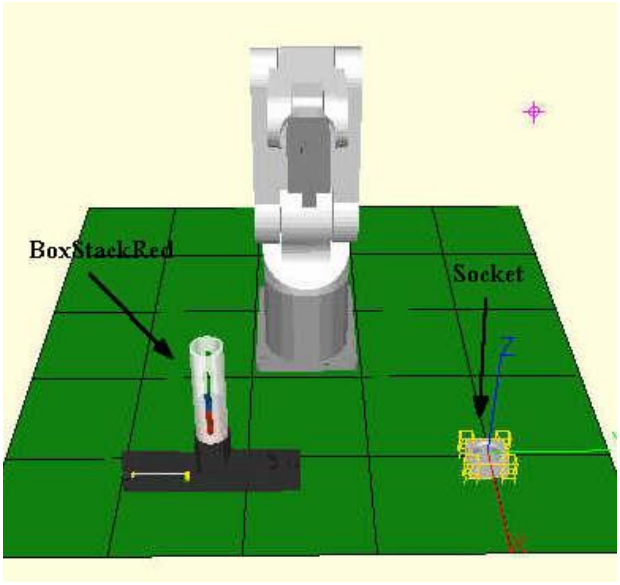
ตรวจสอบผลการทำงานของโปรแกรมโดยจำลองการทำงาน ถ้าผลการทำงานไม่ตรงตามต้องการให้ทบทวนแก้ไขโปรแกรม

เมื่อจบตัวอย่างที่ 3.2 ผู้เรียนสามารถทำแบบฝึกหัดข้อ 3.4 ในหน้าที่ 30 ได้

จากตัวอย่างที่ 3.2 เราใช้อินพุทและเอาท์พุทที่ไฟล์ตัวอย่างกำหนดไว้ให้ แต่ในตัวอย่างที่ 3.3 ผู้เรียนต้องสร้างอินพุทและเอาท์พุทและต้องกำหนดความสัมพันธ์ระหว่างกันด้วยตนเอง

ตัวอย่างที่ 3.3: **การสร้างโมเดลและ** I/O communication

จงเขียนโปรแกรมควบคุมหุ่นยนต์โดยใช้ภาษา Melfa Basic IV โดยใช้ หุ่นยนต์ Mitsubishi RV-2AJ และเพิ่มวัตถุสองชนิด คือ BoxStackedRed และ Socket ซึ่งอยู่ที่ตำแหน่ง (350, -150, 0) และหมุนในทิศทาง Roll –90 องศา และ (125, 290, 35) ตามลำดับ ดังรูป 3.3.4a และมีเงื่อนไขการทำงานดังนี้: เมื่อกดปุ่มสตาร์ทหุ่นยนต์จะเคลื่อนที่ไปหยิบชิ้นงานจาก BoxStackRed โดยที่ชิ้นงานจะถูกดันออกมาโดยกระบอกสูบ เมื่อหยิบชิ้นงานได้แล้วต้องรอจนกว่ากระบอกสูบเคลื่อนที่กลับแล้วจึงนำชิ้นงานไปวางในหลุม (Socket) โดยกำหนดให้ใช้ตำแหน่งดังรูป 3.3.4b



รูป 3.3.4a

ขั้นตอนการสร้างโมเดล: ดูหัวข้อ 3.1.2 ในหน้าที่ 4
ขั้นตอนการเพิ่มวัตถุในโมเดล: ดูหัวข้อ 3.1.3 วิธีที่ 2 ในหน้าที่ 10
ขั้นตอนการกำหนดตำแหน่งในการเคลื่อนที่: ดูหัวข้อ 3.2 ในหน้าที่ 10

รูป 3.3.4b

[RV-2AJ] C:\Program Files\COSIMIR Educational\Proj...

No	Position	Orientation	Comment
P1	355.0, 0.0, 550.0	0, 90,R,A	Home
P2	350.0, -75.0, 33.5	2, 178,R,A	Pick
P3	125.0, 290.0, 30.0	17, 178,R,A	Place

ขั้นตอนการเขียนโปรแกรม

ขั้นตอนที่ 1: เข้าใจปัญหา

ขั้นตอนที่ 2: วิเคราะห์ปัญหา

เนื่องจากปัญหานี้ยังไม่มีความซับซ้อนเท่าใดนักจึงสามารถใช้เทคนิค sequential programming ได้ อย่างไรก็ตามตัวอย่างนี้มีการสร้างวัตถุเพิ่มในโมเดล และวัตถุเหล่านี้มีการติดต่อสื่อสารเพื่อแลกเปลี่ยนข้อมูลกับคอนโทรลเลอร์ของหุ่นยนต์ผ่านอินพุทและเอาท์พุท

1. ตรวจสอบว่าวัตถุที่เลือกมามีอินพุทและเอาท์พุทใดบ้าง

โมเดลนี้ประกอบไปด้วย หุ่นยนต์ Mitsubishi RV-2AJ, Socket และ BoxStackRed ซึ่งประกอบด้วย Push Cylinder จาก Model Explorer สามารถตรวจสอบอินพุท เอาท์พุทเดิมที่กระบอกสูบมีอยู่ ดังรูป 3.3.4c

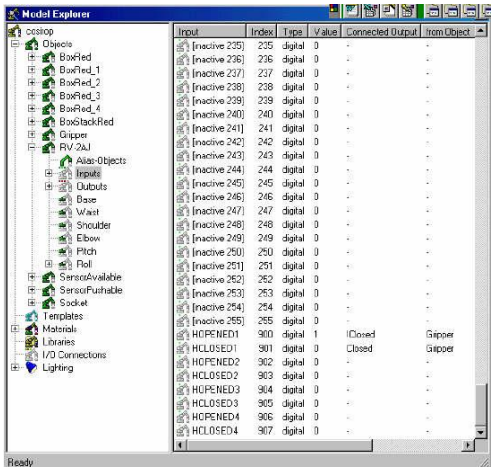
จากรูป 3.3.4c จะพบว่า RV-2AJ มีอินพุทที่ยังไม่ใช้งานตั้งแต่บิต 0 ถึง 255 บิตที่ 900 ถึง 907 เกี่ยวกับกริปเปอร์ เช่นเดียวกันกับเอาท์พุทในรูป 3.3.4d หมายเหตุ: ในที่นี้จะใช้อินพุทและเอาท์พุทบิตที่ 901 คือ HCLOSED1 และ HCLOSE1 ตามลำดับ

จากรูป 3.3.4e จะพบว่า PushCylinder มีอินพุทบิต 0 และ 1 คือ MoveOut และ MoveIn ตามลำดับ เอาท์พุทบิต 0 และ 1 คือ IsMovedOut และ IsMovedIn ตามลำดับ

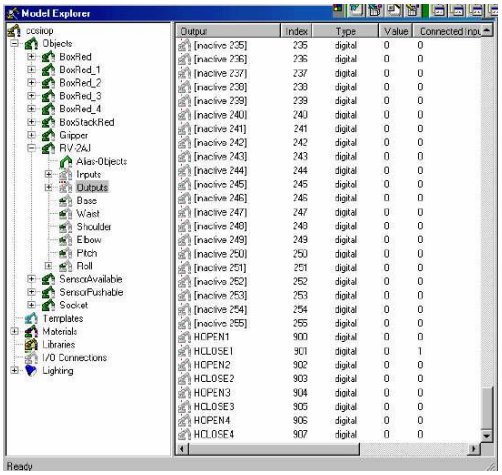
BoxStackRed มีอินพุทบิต 0 คือ NextPart และเอาท์พุทบิต 0 คือ PartAvail

Gripper มีอินพุทบิต 0 คือ Close และเอาท์พุทบิต 0 คือ Closed

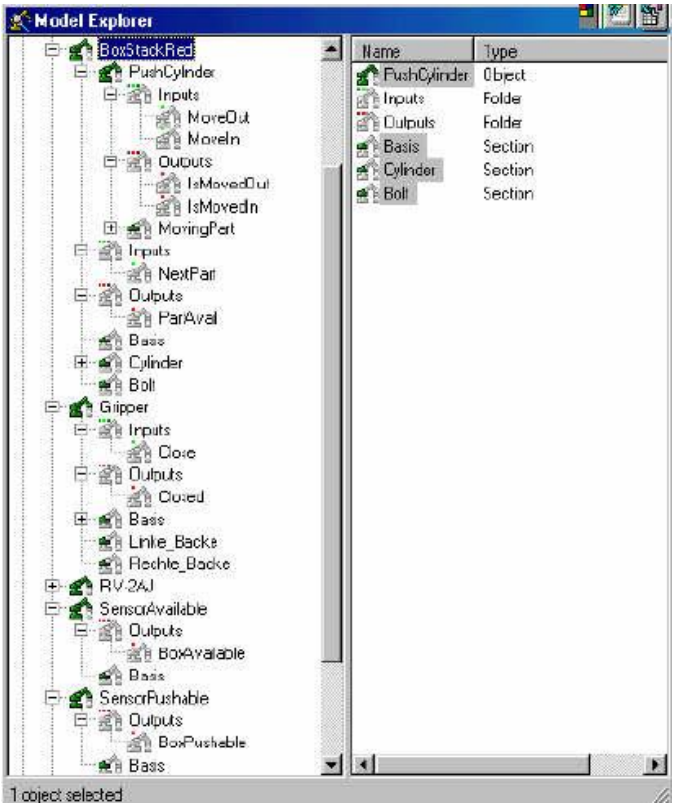
Socket ไม่มีอินพุทและเอาท์พุท



รูป 3.3.4c



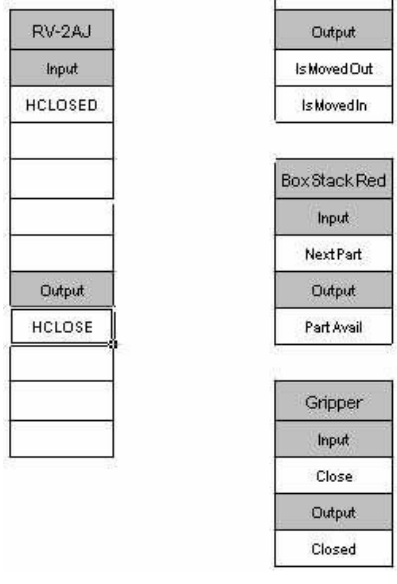
รูป 3.3.4d



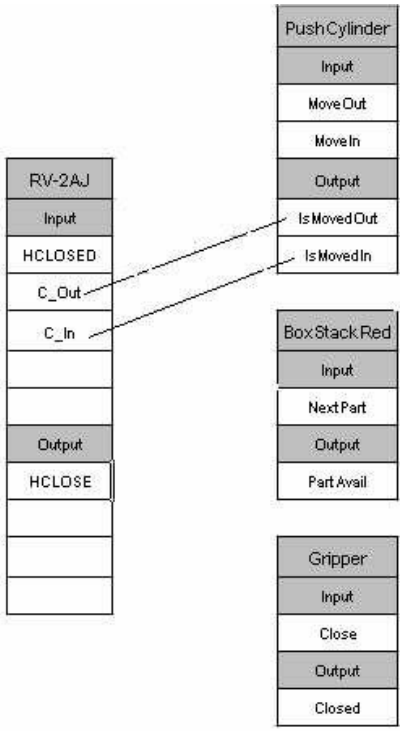
รูป 3.3.4e

2. สร้างตารางแสดงอินพุทและเอาท์พุทของวัตถุทุกชนิดในโมเดลรวมถึงหุ่นยนต์ และเพิ่มอินพุทและเอาท์พุทให้กับหุ่นยนต์และวัตถุอื่นและเชื่อมโยงความสัมพันธ์ระหว่างอินพุทและเอาท์พุท

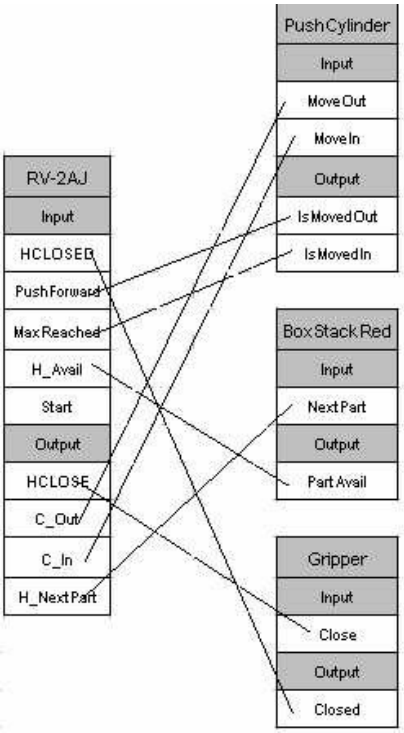
จากข้อ 1 สามารถสรุปเป็นตารางอินพุทและเอาท์พุทของวัตถุทุกชนิดในโมเดลดังรูป 3.3.4f พิจารณาจาก PushCylinder มี อินพุท MoveOut และ MoveIn ซึ่งเป็นสัญญาณที่ได้รับจากหุ่นยนต์ให้กระบอกสูบเคลื่อนที่ออกและเข้าตามลำดับ ดังนั้น หุ่นยนต์ต้องส่งสัญญาณเอาท์พุทให้อินพุทดังกล่าว ซึ่งผู้เรียนต้องกำหนดชื่อสัญญาณเอาท์พุทลงในตาราง ตัวอย่างเช่น กำหนดสัญญาณเอาท์พุทจากหุ่นยนต์ชื่อ C_Out และ C_In เป็นสัญญาณอินพุทของ PushCylinder ชื่อ MoveOut และ MoveIn ตามลำดับ แล้วลากลูกศรเชื่อมโยงระหว่างกัน ดังรูป 3.3.4g (หมายเหตุ: C_Out และ C_In ย่อมาจาก CylinderOut และ CylinderIn ตามลำดับ ในการกำหนดชื่อนั้น กำหนดให้สั้นและสื่อความหมายเพื่อให้ผู้อื่นสามารถเข้าใจได้) ทำจนครบทุกอินพุทและเอาท์พุทจะได้ตารางดังรูป 3.3.4h



รูป 3.3.4f



รูป 3.3.4g

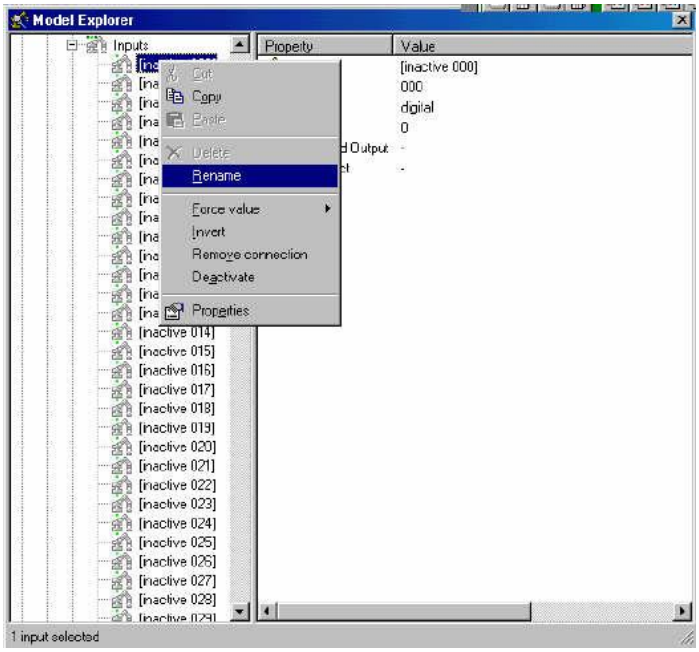


รูป 3.3.4h

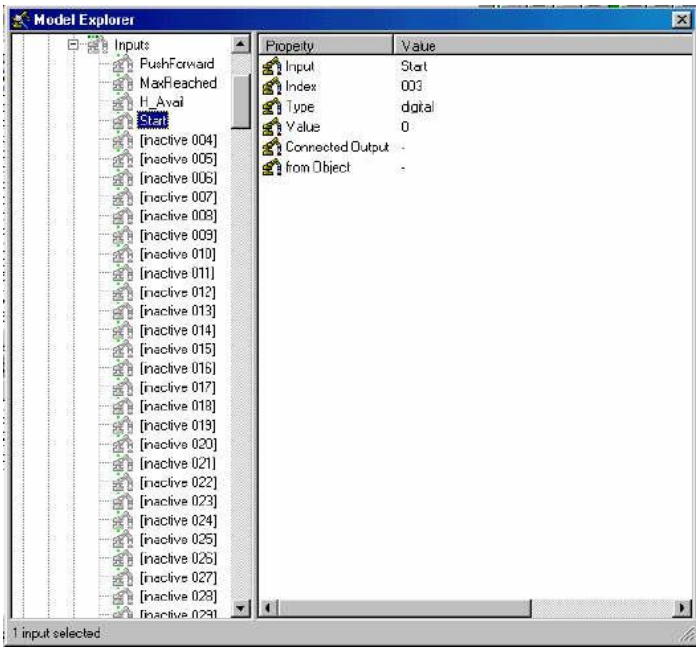
3. เชื่อมโยงความสัมพันธ์อินพุทและเอาท์พุทของหุ่นยนต์และวัตถุอื่น

หลังจากที่ได้ออกแบบความสัมพันธ์ระหว่างอินพุทและเอาท์พุทแล้ว เราจะใช้ข้อมูลเหล่านี้ไว้เชื่อมโยงความสัมพันธ์ระหว่างกันในซอฟต์แวร์ โดยไปที่ Model explorer แล้วเปลี่ยนชื่ออินพุทและเอาท์พุทของหุ่นยนต์

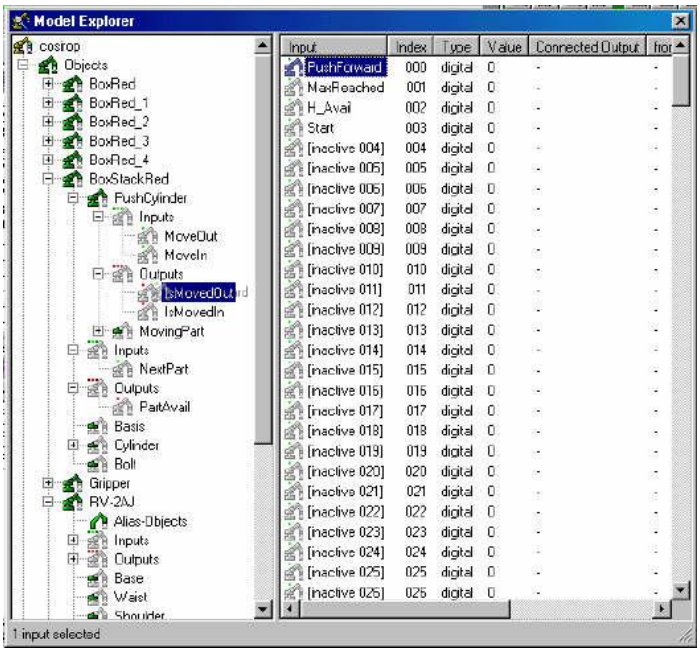
จากรูป 3.3.4i และ 3.3.4j พบว่าหุ่นยนต์มีอินพุทและเอาท์พุทที่ยังไม่ได้ถูกใช้งานอยู่ซึ่งจะปรากฏด้วยชื่อ 'inactive' เช่น inactive 000 ดังนั้นเราสามารถนำมาใช้ได้ด้วยการเปลี่ยนชื่อโดยคลิ๊กขวาที่อินพุทหรือเอาท์พุทที่ปรากฏชื่อ inactive แล้วใช้คำสั่ง Rename ดังรูป แล้วเปลี่ยนชื่อตามที่ได้กำหนดไว้ทั้งในส่วนของอินพุทและเอาท์พุทดังตัวอย่างในรูป 3.3.4i หลังจากนั้นเชื่อมโยงความสัมพันธ์ตามที่ได้ออกแบบไว้โดยจัดให้อินพุทของหุ่นยนต์ทั้งหมดอยู่ทางขวามือ และวัตถุอื่นอยู่ซ้ายมือ ดังรูป 3.3.4k แล้วเชื่อมโยงความสัมพันธ์กันโดย drag and drop เช่น ลากอินพุทของหุ่นยนต์ชื่อ PushForward มาใส่เอาท์พุทของ PushCylinder ชื่อ IsMovedOut แล้วตรวจสอบว่า วัตถุทั้งสองมีการเชื่อมต่อกันดังรูป อินพุท 'PushForward' ต่อกับเอาท์พุท 'IsMovedOut' จาก PushCylinder ทำจนครบทุกอินพุท ในส่วนของเอาท์พุทก็ทำเช่นเดียวกับอินพุทเช่น ลากเอาท์พุท 'C_Out' มาใส่อินพุท 'MoveOut' ของ PushCylinder แล้วตรวจสอบว่า วัตถุทั้งสองมีการเชื่อมต่อกันดังรูป 3.3.4l และ 3.3.4m



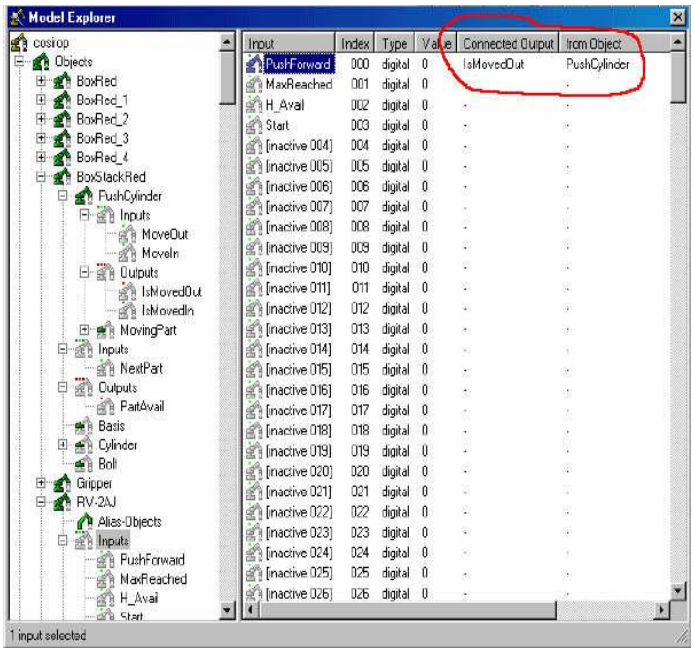
รูป 3.3.4i



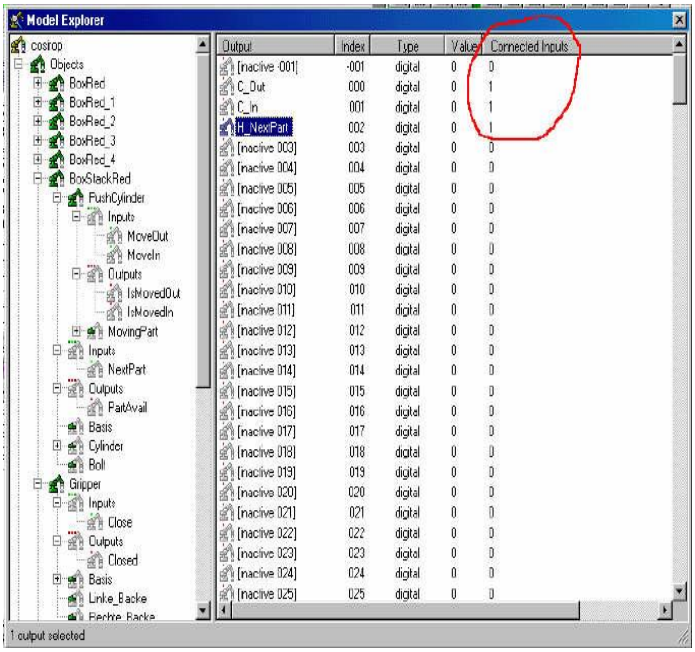
รูป 3.3.4j



รูป 3.3.4k



รูป 3.3.4l



รูป 3.3.4m

ขั้นตอนที่ 3: ออกแบบ

ลำดับขั้นตอนหลัก

1. กดปุ่มสตาร์ท
2. หุ่นยนต์เคลื่อนที่ไปตำแหน่งที่ 1
3. เปิดกริปเปอร์
4. เคลื่อนที่มาเหนือตำแหน่งที่ 2
5. กระบอกสูบเลื่อนชิ้นงานออก
6. กระบอกสูบเคลื่อนที่ออกสุด
7. หุ่นยนต์เคลื่อนที่มาตำแหน่งที่ 3
8. หยิบชิ้นงาน (ปิดกริปเปอร์)
9. กระบอกสูบเคลื่อนที่กลับ
10. รอจนกระบอกสูบเคลื่อนที่เข้าสุด
11. หุ่นยนต์เคลื่อนที่มาเหนือตำแหน่งที่ 2
12. หุ่นยนต์เคลื่อนที่มาเหนือตำแหน่งที่ 3
13. หุ่นยนต์เคลื่อนที่มาตำแหน่งที่ 3
14. ปล่อยชิ้นงาน
15. หุ่นยนต์เคลื่อนที่มาเหนือตำแหน่งที่ 3
16. หุ่นยนต์เคลื่อนที่ไปตำแหน่งที่ 1

ขั้นตอนที่ 4: พัฒนาโปรแกรม

ในขั้นตอนการพัฒนาโปรแกรม ให้นำขั้นตอนทั้งหมดมาเขียนเป็นภาษา Melfa Basic IV ดังรูป 3.3.4n

```
10 WAIT M_IN(3)=1              'WAIT FOR START SIGNAL
20 MOV P1                      'HOME POSITION
30 M_OUT(901)=0                'GRIPPER OPEN
40 MOV P2,-50                  'OVER PICK
50 M_OUT(0)=1                  'CYLINDER OUT
60 WAIT M_IN(0)=1              'WAIT UNTIL HOUSING AVAILABLE
70 MVS P2                      'TO PICK
80 DLY 1
90 M_OUT(901)=1                'GRIPPER CLOSE
100 DLY 1
110 M_OUT(0)=0                 'CANCEL CYLINDER IN
120 M_OUT(1)=1                 'CYLINDER IN
130 WAIT M_IN(1)=1             'WAIT UNTIL CYLINDER IN
140 MVS P2,-50                 'MOVE UP
150 MOV P3,-50                 'OVER PLACE
```

```
160 MVS P3                  'TO PLACE
170 DLY 1
180 M_OUT(901)=0            'GRIPPER OPEN
190 DLY 1
200 MVS P3,-50              'MOVE UP
210 MOV P1                  'HOME POSITION
220 END                     'END PROGRAM
```

รูป 3.3.4n: โปรแกรมแสดงการทำงานตัวอย่างที่ 3.3

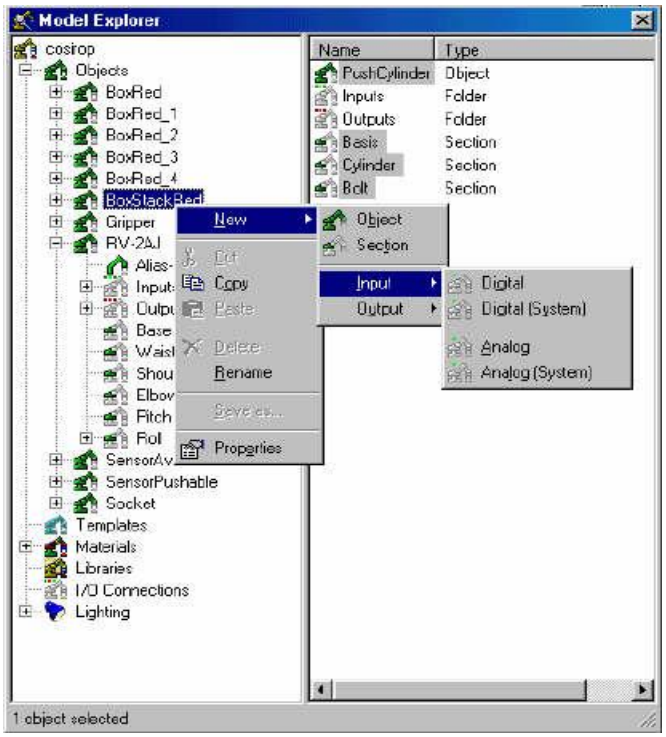
ขั้นตอนที่ 5: ทดสอบโปรแกรม

ตรวจสอบผลการทำงานของโปรแกรมโดยจำลองการทำงาน ถ้าผลการทำงานไม่ตรงตามต้องการให้ทบทวนแก้ไขโปรแกรม

หมายเหตุ: ในกรณีที่อินพุทหรือเอาท์พุทของวัตถุใดไม่พอให้ใช้งานต้องทำการเพิ่มอินพุทหรือเอาท์พุทสามารถทำได้โดยคลิ๊กขวาที่วัตถุนั้น New > Input หรือ output > Digital (System) ดังรูป 3.3.4o

ข้อแตกต่างระหว่าง Digital I/Os และ Digital (System) I/Os คือ Digital (System) I/Os จะถูกประมวลผลและจัดการโดยอุปกรณ์ภายในของระบบนั้น ซึ่งผู้เรียนสามารถดูสถานะของอินพุทและเอาท์พุทเหล่านี้ได้แต่ไม่สามารถบังคับค่าอินพุทหรือเอาท์พุทได้ เช่นในกรณีของ Push cylinder ระบบภายในจะควบคุม Push cylinder เคลื่อนที่เข้าออก

สำหรับ Digital I/Os จะถูกควบคุมโดยผู้ใช้หรือโปรแกรม อินพุทและเอาท์พุทประเภทนี้จะถูกเขียนและอ่านโดยโปรแกรม เช่น ในกรณีของ Push button ผู้เรียนกดปุ่มโดยเปลี่ยนค่าจาก 0 เป็น 1 ซึ่งผู้เรียนควบคุมค่าอินพุทเอาท์พุทได้



รูป 3.3.4o

แบบฝึกหัดที่ 3.1

จงเขียนโปรแกรมควบคุมหุ่นยนต์โดยใช้ภาษา Melfa Basic IV ให้หุ่นยนต์เคลื่อนที่ไปหยิบชิ้นงานขึ้นมาจาก Storage

หมายเหตุ: ใช้โมเดล PickAndPlaceKuka และใช้ตำแหน่งที่กำหนดในตัวอย่างที่ 2.1

แบบฝึกหัดที่ 3.2

จงเขียนโปรแกรมควบคุมหุ่นยนต์โดยใช้ภาษา Melfa Basic IV ให้หุ่นยนต์เคลื่อนที่ไปหยิบชิ้นงานขึ้นมาจาก Storage ไปวางที่ Stacking

หมายเหตุ: ใช้โมเดล PickAndPlaceKuka และใช้ตำแหน่งที่กำหนดในแบบฝึกหัดที่ 2.1

แบบฝึกหัดที่ 3.3: **ใช้คำสั่งวน** loop **การทำงาน**

จงเขียนโปรแกรมควบคุมหุ่นยนต์โดยใช้ภาษา Melfa Basic IV ให้หุ่นยนต์เคลื่อนที่จากตำแหน่งที่ 1 ถึงตำแหน่งที่ 5 แล้ววนรอบการทำงานเคลื่อนที่จากตำแหน่งที่ 1 ถึง 5 ซ้ำไปเรื่อยๆ

หมายเหตุ: สร้างโมเดล RV-2AJ และกำหนดตำแหน่งใดๆ 5 ตำแหน่ง

แบบฝึกหัดที่ 3.4: I/O communication and structured programming

จงเขียนโปรแกรมควบคุมหุ่นยนต์ต่อจากตัวอย่างที่ 3.2 โดยใช้ภาษา Melfa Basic IV ให้หุ่นยนต์ประกอบชิ้นงานเข้ากับลูกสูบสปริงและฝา

หมายเหตุ: โดยมีเงื่อนไขเพิ่มเติมดังต่อไปนี้นี้: ชิ้นงานสีดำจะประกอบเข้ากับลูกสูบขนาดเล็ก (สีเทา) ชิ้นงานที่เหลือ คือ สีเงินและสีแดงประกอบเข้ากับลูกสูบขนาดใหญ่ (สีดำ)

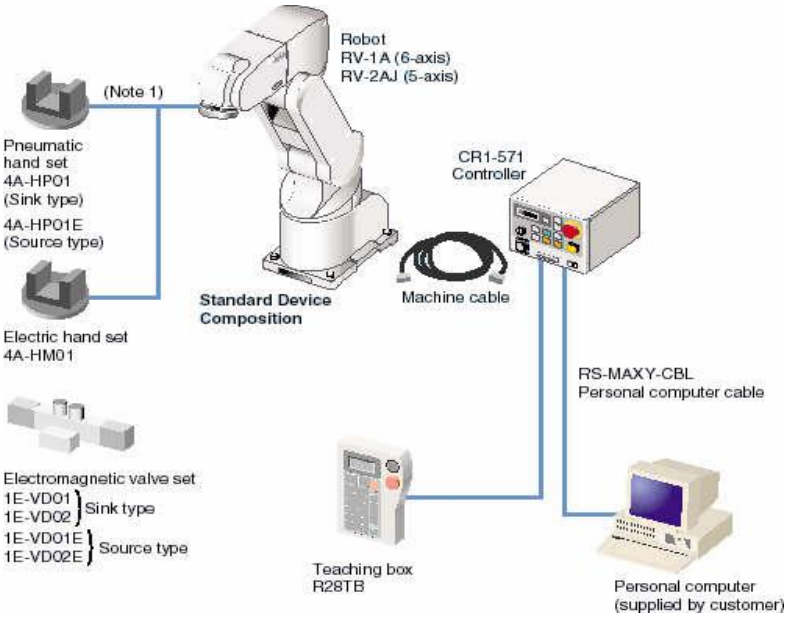
หลังจากที่ผู้เรียนมีความคุ้นเคยกับซอฟต์แวร์ COSIMIR Educational และสามารถเขียนโปรแกรมได้แล้ว ผู้เรียนมีความพร้อมที่จะทำงานกับหุ่นยนต์จริงและใช้ซอฟต์แวร์ COSIMIR Industrial ซึ่งมีฟังก์ชั่นเพิ่มจาก COSIMIR Educational คือสามารถติดต่อสื่อสารกับคอนโทรลเลอร์ของหุ่นยนต์ได้

4.0 ระบบโดยรวม: หุ่นยนต์ + คอนโทรลเลอร์ + Teaching Box (TB) + คอมพิวเตอร์

ประกอบไปด้วยส่วนประกอบหลักๆ คือ หุ่นยนต์ คอนโทรลเลอร์ TBและคอมพิวเตอร์ซึ่งมีซอฟต์แวร์ COSIMIR Industrial ติดตั้งอยู่

หุ่นยนต์และคอนโทรลเลอร์เชื่อมต่อกันผ่าน machine cable โดยที่คอนโทรลเลอร์ซึ่งเปรียบเสมือนตัวสั่งการของหุ่นยนต์เชื่อมต่อกับ TB และ คอมพิวเตอร์

รูปที่ 4.0.0a: รูปแสดงระบบการเรียนการสอนเรื่องหุ่นยนต์อุตสาหกรรม



4.0.1 หุ่นยนต์

หุ่นยนต์ที่ใช้อ้างอิงในคู่มือนี้เป็นหุ่นยนต์มิตซูบิชิ รุ่น RV-2AJ ซึ่งเป็นหุ่นยนต์อุตสาหกรรมถูกออกแบบมาให้ใช้ได้กับงานทุกประเภท (all-purpose industrial robot) RV-2AJ ถูกพัฒนาขึ้นมาจากรุ่น RV-M1 มีความรวดเร็วและแม่นยำในการทำงานเพิ่มมากขึ้น มีแกนทั้งหมด 5 แกน คือ แกนที่ 1, 2, 3, 5 และ 6แต่ละแกนเคลื่อนที่ด้วยเซอร์โวมอเตอร์ (AC servo motor)

หมายเหตุ: รายละเอียดของสเปกหรือคุณสมบัติอื่นๆ ของหุ่นยนต์ขอให้ผู้เรียนศึกษาเพิ่มเติมจากคู่มือที่มาพร้อมกับหุ่นยนต์

หุ่นยนต์รุ่น RV-2AJ ถูกนำไปใช้ในอุตสาหกรรมต่อไปนี้ อุตสาหกรรมอิเล็กทรอนิกส์ที่ต้องการเครื่องจักรที่มีความแม่นยำสูง อุตสาหกรรมอาหาร เครื่องสำอางและยา และการศึกษาและการวิจัย

ตัวอย่างการประยุกต์ใช้ในอุตสาหกรรมอิเล็กทรอนิกส์ เช่น ในงานขนย้ายซีดีรอมดังรูป 4.0.1a งานประกอบชิ้นส่วนอิเล็กทรอนิกส์ ดังรูป 4.0.1b งานประกอบชิ้นส่วนทั่วไป ดังรูป 4.0.1c งานตรวจสอบคุณภาพคีย์บอร์ดและชิ้นส่วนอิเล็กทรอนิกส์ดังรูป 4.0.1d และ 4.0.1e

รูปที่ 4.0.1a: งานขนย้ายซีดีรอม

รูปที่ 4.0.1b: งานประกอบชิ้นส่วนอิเล็กทรอนิกส์

รูปที่ 4.0.1c: งานประกอบชิ้นส่วนทั่วไปโดยหุ่นยนต์ทำหน้าที่หยอดกาว

รูปที่ 4.0.1d: งานตรวจสอบคีย์บอร์ด

รูปที่ 4.0.1e: งานตรวจสอบชิ้นส่วนอิเล็กทรอนิกส์

ตัวอย่างการประยุกต์ใช้ในอุตสาหกรรมอาหาร เครื่องสำอางและยา เช่น ในงานตรวจสอบและวิเคราะห์คุณภาพยา ดังรูป 4.0.1f และงานบรรจุอาหาร ดังรูป 4.0.1g

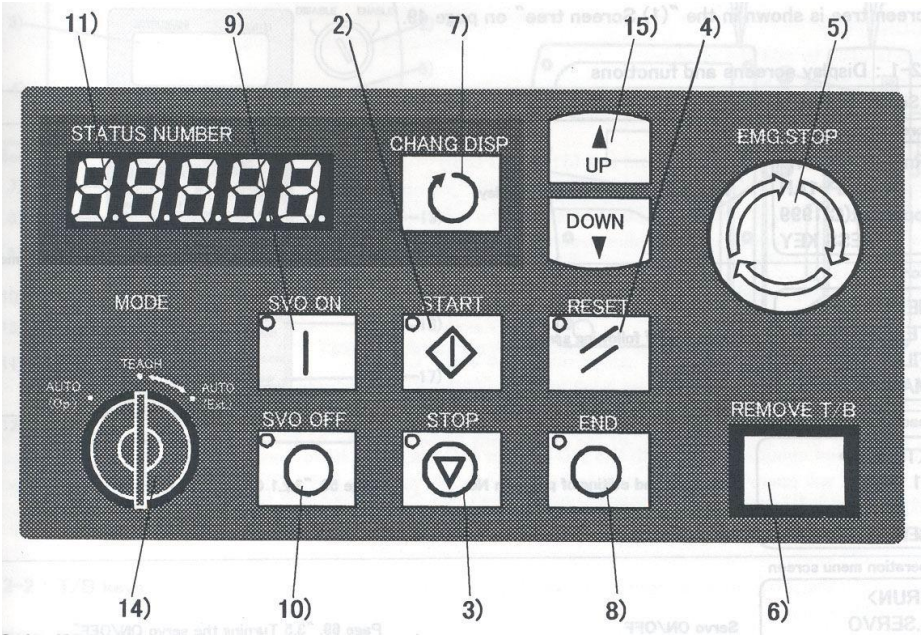
รูปที่ 4.0.1f: งานตรวจสอบและวิเคราะห์คุณภาพยา

รูปที่ 4.0.1g: งานบรรจุอาหาร

4.0.2 **คอนโทรลเลอร์**

คอนโทรลเลอร์ของหุ่นยนต์ RV-2AJ เป็นรุ่น CR2-571 ภาษาที่รองรับ คือ MELFA BASIC IV และ MOVEMASTER COMMAND สามารถบันทึกโปรแกรมได้ 88 โปรแกรม และสามารถบันทึกตำแหน่งได้ 2, 500 ตำแหน่ง

รูปที่ 4.0.2a: แผงด้านหน้าของคอนโทรลเลอร์



2) Start button…ทำหน้าที่รันโปรแกรมและสั่งให้หุ่นยนต์ทำงาน โปรแกรมจะรันอย่างต่อเนื่องโดยจะมีไฟ LED สีเขียวติดอยู่ตลอดที่ปุ่มนี้ทำงาน

3) Stop button…ทำหน้าที่หยุดการทำงานของหุ่นยนต์ทันที แต่ เซอร์โวมอเตอร์ ยังทำงานอยู่ ไฟ LED สีแดงติดอยู่ตลอดที่ปุ่มนี้ทำงาน

4) Reset button…ทำหน้าที่รีเซ็ท error และรีเซ็ทสภาวะโปรแกรมที่ถูกขัดจังหวะและรีเซ็ทโปรแกรมให้เริ่มที่บรรทัดแรกใหม่อีกครั้ง ไฟ LED สีแดงจะติดอยู่ตลอดเมื่อมี error เกิดขึ้น

5) Emergency stop button…ทำหน้าที่หยุดการทำงานของหุ่นยนต์โดยที่ เซอร์โวมอเตอร์ หยุดการทำงานด้วย

6) T/B Connection switch…ทำหน้าที่เชื่อมต่อหรือยกเลิกการเชื่อมต่อกับ TB โดยไม่ต้องทำปิดคอนโทรลเลอร์

7) CHANG DISP… เมื่อกดปุ่มนี้ หน้าจอจะเปลี่ยนรายละเอียดที่แสดง โดยเรียงตามลำดับมีดังนี้ คือ หมายเลขโปรแกรม →หมายเลขบรรทัดของโปรแกรม →ความเร็ว override นอกจากนี้ยังทำหน้าที่แสดง หมายเลข error ที่เกิดขึ้น

8) End button…ทำหน้าที่จบการอ่านโปรแกรมที่บรรทัดสุดท้ายหรือเมื่อเจอคำสั่ง END ไฟ LED สีแดงจะติดตลอดขณะระหว่างการทำงานหนึ่งรอบ

9) SVO.ON switch… เมื่อกดปุ่มนี้ เซอร์โวมอเตอร์ จะเริ่มต้นทำงานโดยมีไฟ LED สีเขียวติดตลอดเวลาที่ เซอร์โวมอเตอร์ทำงาน

10) SVO.OFF switch… เมื่อกดปุ่มนี้ เซอร์โวมอเตอร์ จะหยุดทำงานโดยมีไฟ LED สีแดงติดตลอดเวลาที่ เซอร์โวมอเตอร์หยุดการทำงาน

11) STATUS.NUMBER (display panel)…หน้าจอทำหน้าที่แสดงหมายเลข error หมายเลขโปรแกรม หมายเลขบรรทัด ค่าเปอร์เซ็นต์ความเร็ว override

14) MODE changeover switch…ทำหน้าที่เปลี่ยนโหมดการทำงานของหุ่นยนต์ โดยมี 3 โหมดดังต่อไปนี้

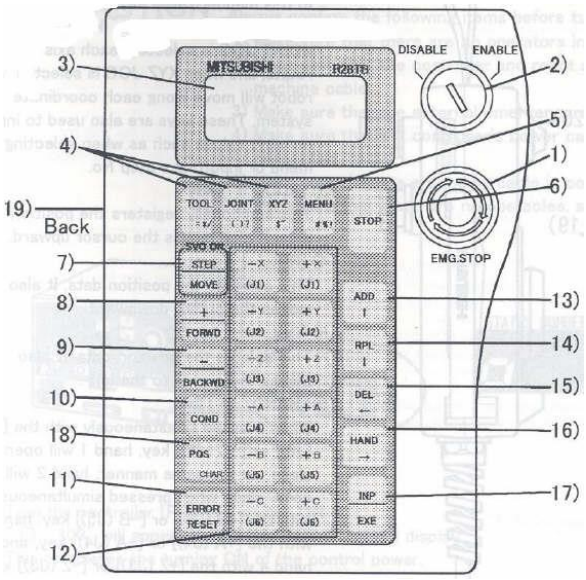
AUTO (Op.)… เป็นโหมดสำหรับการทำงานของคอนโทรลเลอร์เท่านั้น

TEACH…เป็นโหมดที่ใช้ร่วมกับการทำงานของ TB เท่านั้น

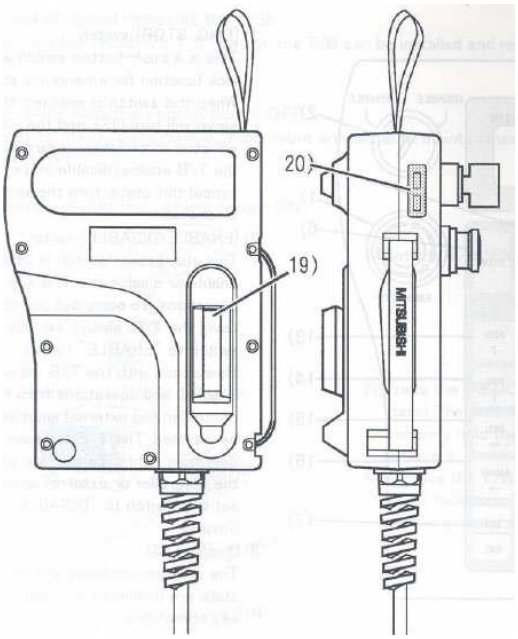
AUTO (Ext.)… เป็นโหมดสำหรับการสั่งงานคอนโทรลเลอร์จากอุปกรณ์ภายนอก (คอมพิวเตอร์) เท่านั้น

15) UP/ DOWN switch… ทำหน้าที่เปลี่ยนรายละเอียดที่แสดงบนหน้าจอขึ้นลง

4.0.3 Teaching Box (TB)



รูปที่ 4.0.3a: ภาพด้านหน้าของ TB



รูปที่ 4.0.3b: ภาพด้านข้างและด้านหลังของ TB

1) [EMG. STOP] switch…เมื่อกดปุ่มนี้ เซอร์โวมอเตอร์ จะหยุดการทำงานและหุ่นยนต์จะหยุดการทำงานทันทีไม่ว่า TB จะทำงานอยู่หรือไม่ ถ้าต้องการยกเลิกการทำงานของปุ่มนี้ให้หมุนปุ่มไปในทิศทางตามเข็มนาฬิกา
2) [ENABLE/ DISABLE] switch…สวิตช์นี้ทำหน้าที่ปิดหรือเปิด TB เมื่อต้องการใช้การ TB ให้หมุนสวิตช์ไปที่ ‘ENABLE’ ซึ่งจะหยุดการทำงานจากคอนโทรลเลอร์และอุปกรณ์ภายนอกทันที เมื่อไม่ต้องการใช้ TB แล้วให้หมุนสวิตช์ไปที่ ‘DISABLE’
3) Display LCD…โปรแกรมและสถานะของหุ่นยนต์จะปรากฏบนหน้าจอเมื่อสวิตช์ของ TB อยู่ที่ ‘ENABLE’
4) [TOOL] key…เคลื่อนที่แบบ tool jog
4) [JOINT] key…เคลื่อนที่แบบ joint jog
4) [XYZ] key…เคลื่อนที่แบบในแนวแกน XYZ
5) [MENU] key…กดปุ่ม ‘MENU’ เพื่อต้องการกลับสู่เมนูหลัก
6) [STOP] key…ทำหน้าที่หยุดการอ่านโปรแกรมและค่อยๆ ลดความเร็วของหุ่นยนต์จนกระทั่งหยุด ซึ่งเป็นคำสั่งเดียวกับ STOP button หน้าแผงคอนโทรลเลอร์และสามารถใช้ได้ในทุกกรณีแม้ว่า สวิตช์จะอยู่ที่ตำแหน่ง ‘DISABLE’
7) [STEP/MOVE] key…การเคลื่อนที่ของหุ่นยนต์สามารถกระทำได้ก็ต่อเมื่อกดปุ่มนี้ค้างไว้พร้อมกับปุ่ม [Jog operation] key ซึ่งการกดปุ่มนี้ทำให้ เซอร์โวมอเตอร์ ทำงาน นอกจากนี้ step jump ยังสามารถกระทำได้โดยกดปุ่มนี้ค้างไว้พร้อมกับปุ่ม [INP/EXE]
8) [+/FORWD] key…step feed สามารถกระทำได้เมื่อกดปุ่มนี้ค้างไว้พร้อมกับปุ่ม [INP/EXE] เมื่อกดปุ่มนี้พร้อมกับ [STEP/MOVE] ความเร็ว override จะเพิ่มขึ้น
9) [-/BACKWD] key…เมื่อกดปุ่มนี้พร้อมกับปุ่ม [STEP/MOVE] ความเร็ว override จะลดลง
10) [COND]…ทำหน้าที่แก้ไขโปรแกรม
11) [ERROR RESET] key…ทำหน้าที่รีเซ็ท error ที่เกิดขึ้น เมื่อกดปุ่มนี้พร้อมกับ [INP/EXE] โปรแกรมจะถูกรีเซ็ท
12) [Jog operation] key (ปุ่มจำนวน 12 ปุ่มเริ่มจาก [-X (J1) ถึง +C(J6)]…ปุ่มจำนวน 12 ปุ่มเหล่านี้เรียกว่าปุ่ม ‘jog operation’ เมื่อเลือกการเคลื่อนที่แบบ joint jog แกนแต่ละแกนจะหมุน และเมื่อเลือกการเคลื่อนที่ในแนวแกน XYZ หุ่นยนต์จะเคลื่อนที่ตามแกนแต่ละแกน นอกจากนี้ปุ่มเหล่านี้ยังถูกใช้ในการใส่ค่าตัวเลขเช่นในการเลือกเมนูหรือใส่หมายเลขตำแหน่ง
13) [ADD ↑] key…ทำหน้าที่เพิ่มตำแหน่งและนอกจากนี้ยังทำหน้าที่เคลื่อนที่เคอร์เซอร์ขึ้น (↑)
14) [RPL/ ↓] key…This corrects the position data. นอกจากนี้ยังทำหน้าที่เคลื่อนที่เคอร์เซอร์ลง (↓)
15) [DEL/ ←] key…ทำหน้าที่ลบตำแหน่ง นอกจากนี้ยังทำหน้าที่เคลื่อนที่เคอร์เซอร์ไปทางซ้าย (←)
16) [HAND/ →] key…

    เมื่อกดปุ่มนี้พร้อมกับปุ่ม [+C (J6)] หรือ [- C (J6)] กริปเปอร์หมายเลข 1 จะปิดหรือเปิด
    เมื่อกดปุ่มนี้พร้อมกับปุ่ม [+B (J5)] หรือ [- B (J5)] กริปเปอร์หมายเลข 2 จะปิดหรือเปิด
    เมื่อกดปุ่มนี้พร้อมกับปุ่ม [+A (J4)] หรือ [- A (J4)] กริปเปอร์หมายเลข 3 จะปิดหรือเปิด
    เมื่อกดปุ่มนี้พร้อมกับปุ่ม [+Z (J3)] หรือ [- Z (J3)] กริปเปอร์หมายเลข 4 จะปิดหรือเปิด
    นอกจากนี้ปุ่มนี้ยังทำหน้าที่เคลื่อนที่เคอร์เซอร์ไปทางขวา

17) [INP/ EXE] key…ทำหน้าที่ป้อนโปรแกรมและทำหน้าที่บันทึกตำแหน่ง

18) [POS CHAR] key…ทำหน้าที่เปลี่ยนลักษณะการป้อนข้อมูลคือ เปลี่ยนไปมาระหว่างตัวเลขและตัวอักษรในระหว่างการแก้ไขตำแหน่ง เป็นต้น

19) Deadman switch… เมื่อบิดสวิตช์มาที่ตำแหน่ง 'ENABLE' และกดปุ่มนี้ค้างไว้ เซอร์โวมอเตอร์ จะทำงาน ถ้าปล่อยปุ่มนี้ เซอร์โวมอเตอร์ หยุดทำงาน อย่างไรก็ตาม ในกรณีที่มีการกดปุ่ม [EMG. STOP] หรือ SVO.OFF switch บนแผงคอนโทรลเลอร์ทำงาน เซอร์โวมอเตอร์ จะหยุดการทำงาน และจะไม่กลับสู่สภาวะการทำงานแม้ว่าจะกดปุ่ม Deadman switch ไว้ก็ตาม ในกรณีนี้สามารถแก้ไขได้โดยกดปุ่ม SVO.ON switch

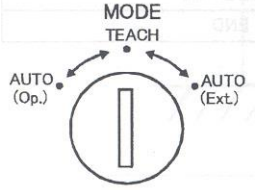
20) Contrast setting switch…ทำหน้าที่ปรับความสว่างของหน้าจอ กดปุ่มด้านบนเพื่อปรับหน้าจอให้เข้มขึ้น กดปุ่มด้านล่างเพื่อปรับหน้าจอให้สว่างขึ้น

4.1 วิธีการใช้ Teaching Box (TB)

4.1.1 การกำหนดตำแหน่งด้วย TB

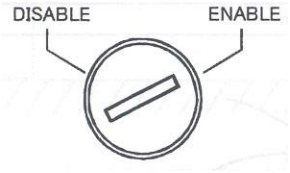
การกำหนดตำแหน่งทำได้สองวิธีคือ การกำหนดตำแหน่งด้วย TB ซึ่งจะกล่าวในหัวข้อนี้และการกำหนดตำแหน่งบนซอฟต์แวร์ซึ่งจะกล่าวในหัวข้อ 4.3.2 แม้ว่าเราสามารถกำหนดตำแหน่งบนซอฟต์แวร์ได้แต่ผู้เขียนขอแนะนำให้ใช้ TB เพราะการใช้ TB ทำให้ผู้เรียนต้องอยู่หน้างานจริงและกำหนดตำแหน่งได้ถูกต้องมากกว่า การกำหนดตำแหน่งบนซอฟต์แวร์นั้นสิ่งแวดล้อมไม่เหมือนกับของจริงและผู้เรียนอาจไม่ได้อยู่ใกล้กับหน้างานจึงมีส่วนให้การกำหนดตำแหน่งผิดพลาดได้ สำหรับขั้นตอนในการกำหนดตำแหน่งด้วย TB มีดังต่อไปนี้

1) บิดกุญแจที่คอนโทรลเลอร์มาที่ตำแหน่ง 'TEACH' ดังรูปที่ 4.1.1a



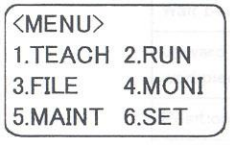
รูปที่ 4.1.1a: บิดสวิตช์หน้าแผงคอนโทรลเลอร์มาที่ตำแหน่ง 'TEACH'

2) บิดกุญแจที่ TB มาที่ตำแหน่ง 'ENABLE' ดังรูปที่ 4.1.1b



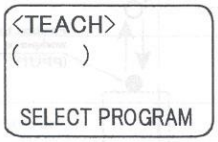
รูปที่ 4.1.1b: บิดสวิตช์ที่ TB มาที่ตำแหน่ง 'ENABLE'

3) กดปุ่ม [MENU] ที่ TB หน้าจอที่ TB จะปรากฏดังรูปที่ 4.1.1c ใช้ปุ่ม [ADD ↑], [RPL/ ↓], [DEL/ ←] และ [HAND/ →] เพื่อเลื่อนเคอร์เซอร์ไปยังฟังก์ชั่นที่ต้องการ



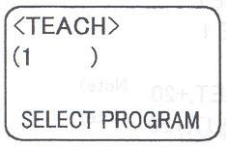
รูปที่ 4.1.1c: หน้าจอของ TB

4) เลือกหมายเลข 1 'TEACH' แล้วกดปุ่ม [INP/ EXE] เพื่อตอบตกลง หน้าจอที่ TB จะปรากฏดังรูปที่ 4.1.2d

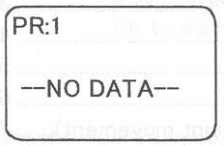


รูปที่ 4.1.1d: หน้าจอของ TB

5) ใส่หมายเลขโปรแกรมที่ต้องการ (เลือกได้ตั้งแต่ 1 ถึง 999) ดังตัวอย่างในรูป 4.1.2eหลังจากนั้นกดปุ่ม [INP/ EXE] เพื่อตอบตกลง หน้าจอจะเปลี่ยนไปดังรูปที่ 4.1.2f ซึ่งในขณะนี้โปรแกรมที่หนึ่งยังไม่มีข้อมูลหรือตำแหน่งถูกบันทึกอยู่



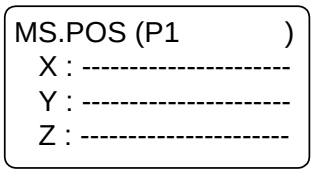
รูปที่ 4.1.1e: ระบุหมายเลขโปรแกรม



รูปที่ 4.1.1f: ระบุหมายเลขโปรแกรม

หมายเหตุ: ในการใช้งานจริงต้องไม่ใช้โปรแกรมหมายเลข 1 เป็นอันขาดเนื่องจากเป็นโปรแกรมที่สมบูรณ์สำหรับการทำงานของชุด MPS การเรียกใช้โปรแกรมหมายเลข 1 หมายถึงการบันทึกทับโปรแกรมเดิม

6) กดปุ่ม [POS CHAR] ดังรูป 4.1.2g



รูปที่ 4.1.1g: ระบุหมายเลข

7) เริ่มต้นการกำหนดตำแหน่ง โดยใช้มือซ้ายกดปุ่ม Deadman switch พร้อมกับปุ่ม [STEP/MOVE] ค้างไว้จนได้ยินเสียง เซอร์โวมอเตอร์ ทำงานและไฟ LED สีเขียวติดที่ปุ่ม SVO.ON switch ซึ่งหมายความว่า เซอร์โวมอเตอร์อยู่ในสภาพพร้อมทำงานแล้ว

และใช้มือขวากดปุ่ม [TOOL] หรือ [JOINT] หรือ [XYZ] เพื่อเลือกรูปแบบการเคลื่อนที่

8) โดยที่มือซ้ายยังกด Deadman switch พร้อมกับปุ่ม [STEP/MOVE] อยู่ ใช้มือขวากดปุ่ม [Jog operation] key จำนวน 12 ปุ่มเริ่มจาก [-X (J1) ถึง +C(J6)]

ในกรณีที่เลือกใช้การเคลื่อนที่แบบ XYZ mode ให้ใช้ปุ่มต่อไปนี้...

[-X (J1)] และ [+X (J1)] ทำหน้าที่เคลื่อนที่ไปในแกน X- และ X+ ตามลำดับ

[-Y (J2)] และ [+Y (J2)] ทำหน้าที่เคลื่อนที่ไปในแกน Y- และ Y+ ตามลำดับ

[-Z (J3)] และ [+Z (J3)] ทำหน้าที่เคลื่อนที่ไปในแกน Z- และ Z+ ตามลำดับ

ในกรณีที่เลือกใช้การเคลื่อนที่แบบ Joint mode ให้ใช้ปุ่มต่อไปนี้...

[-X (J1)] และ [+X (J1)] ทำหน้าที่เคลื่อนที่ตามแกนที่ 1 ไปในทิศทาง – และ+ตามลำดับ

[-Y (J2)] และ [+Y (J2)] ทำหน้าที่เคลื่อนที่ไปในแกนที่ 2 ไปในทิศทาง – และ+ตามลำดับ

[-Z (J3)] และ [+Z (J3)] ทำหน้าที่เคลื่อนที่ไปในแกนที่ 3 ไปในทิศทาง – และ+ตามลำดับ

[-A (J4)] และ [+A (J4)] ทำหน้าที่เคลื่อนที่ไปในแกนที่ 4 ไปในทิศทาง – และ+ตามลำดับ

[-B (J5)] และ [+B (J5)] ทำหน้าที่เคลื่อนที่ไปในแกนที่ 5 ไปในทิศทาง – และ+ตามลำดับ

[-C (J6)] และ [+C (J6)] ทำหน้าที่เคลื่อนที่ไปในแกนที่ 6 ไปในทิศทาง – และ+ตามลำดับ

หมายเหตุ: ในกรณีที่ยังไม่ได้ยินเสียง เซอร์โวมอเตอร์ ทำงานแล้วกดปุ่มใดปุ่มหนึ่งของ [Jog Operation] key จะเกิด error ขึ้น

ในกรณีที่ต้องการเปิดหรือปิดกริปเปอร์ให้กดปุ่ม [HAND/ →] ค้างไว้แล้วกด [-C (J6)] เพื่อปิดกริปเปอร์ หรือ [+C (J6)] เพื่อเปิดกริปเปอร์ โดยไม่ที่ต้องกดปุ่ม Deadman switch และ [STEP/MOVE]

9) เมื่อได้ตำแหน่งที่ต้องการ ให้บันทึกตำแหน่งโดยปล่อยปุ่ม Deadman switch และ [STEP/MOVE]

10) ใส่หมายเลขตำแหน่ง เช่น P1, P2, … โดยใช้ปุ่ม [POS CHAR] เพื่อป้อนหมายเลขและ ปุ่ม [ADD ↑], [RPL/ ↓], [DEL/ ←] และ [HAND/ →] เพื่อเลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ

11) เมื่อใส่หมายเลขตำแหน่งเรียบร้อยแล้วให้กดปุ่ม Deadman switch และ [STEP/MOVE] ค้างไว้ แล้วกดปุ่ม [ADD ↑] 2 ครั้งเพื่อบันทึกตำแหน่ง

12) ทำซ้ำขั้นตอนที่ 7) ถึง 11) ใหม่ทุกครั้งที่ต้องการเพิ่มตำแหน่ง

4.1.2 วิธีการแสดงตำแหน่ง

ในกรณีที่กำหนดตำแหน่งต่างๆ ครบเรียบร้อยแล้ว ต้องการตรวจสอบว่าตำแหน่งต่างๆ อยู่ ณ ที่ตำแหน่งใดบ้าง มีวิธีการตรวจสอบดังต่อไปนี้

1) บิดกุญแจที่คอนโทรลเลอร์มาที่ตำแหน่ง ‘TEACH’ ดังรูปที่ 4.1.1a

2) บิดกุญแจที่ TB มาที่ตำแหน่ง ‘ENABLE’ ดังรูปที่ 4.1.1b

3) กดปุ่ม [MENU] ที่ TB หน้าจอที่ TB จะปรากฏดังรูปที่ 4.1.1c ใช้ปุ่ม [ADD ↑], [RPL/ ↓], [DEL/ ←] และ [HAND/ →] เพื่อเลื่อนเคอร์เซอร์ไปยังฟังก์ชั่นที่ต้องการ

4) เลือกหมายเลข 1 ‘TEACH’ แล้วกดปุ่ม [INP/ EXE] เพื่อตอบตกลง หน้าจอที่ TB จะปรากฏดังรูปที่ 4.1.2d

5) ใส่หมายเลขโปรแกรมที่ต้องการตรวจสอบตำแหน่ง

6) กดปุ่ม [POS CHAR] จะปรากฏหน้าจอดังรูป 4.1.2g

7) ใส่หมายเลขตำแหน่งที่ต้องการตรวจสอบ เช่น P1, P2, …

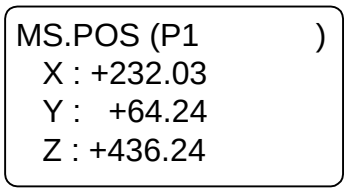
8) ใช้มือซ้ายกดปุ่ม Deadman switch พร้อมกับปุ่ม [STEP/MOVE] ค้างไว้จนได้ยินเสียง เซอร์โวมอเตอร์ ทำงาน และไฟ LED สีเขียวติดที่ปุ่ม SVO.ON switch ซึ่งหมายความว่า เซอร์โวมอเตอร์ อยู่ในสภาพพร้อมทำงานแล้ว แล้วใช้มือขวากดปุ่ม [INP/ EXE] ค้างไว้จนกว่าหุ่นยนต์จะเคลื่อนที่กลับสู่ตำแหน่งที่กำหนดไว้

หมายเหตุ:

- ในกรณีที่กดปุ่ม [INP/ EXE] ก่อนได้ยินเสียง เซอร์โวมอเตอร์ ทำงาน จะทำให้เกิด error
- ในบางกรณี หุ่นยนต์ไม่สามารถกลับสู่ตำแหน่งที่กำหนดให้ได้เนื่องจากข้อจำกัดขององศาการหมุนของแต่ละข้อซึ่งเป็นผลมาจากการออกแบบโครงสร้างหุ่นยนต์จากบริษัทผู้ผลิต

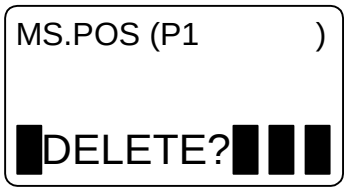
4.1.3 **วิธีการลบตำแหน่ง**

1) ให้หน้าจอปรากฏตำแหน่งที่ต้องการลบ โดยใส่ตำแหน่งที่ต้องการลบ แล้วกดปุ่ม [INP/ EXE] เช่น ดังในรูป 4.1.3a



รูปที่ 4.1.3a: หน้าจอ TB แสดงตำแหน่งที่ต้องการลบ

2) กดปุ่ม [STEP/MOVE] ค้างไว้แล้วกดปุ่ม [DEL/ ←] หนึ่งครั้งแล้วปล่อย จะได้ยินเสียงบี๊บ 3 ครั้ง เพื่อเป็นการเตือน และจะปรากฏข้อความเตือนดังรูป 4.1.3b



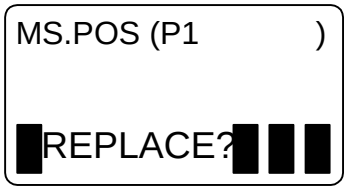
รูปที่ 4.1.3b: หน้าจอ TB แสดงข้อความเตือนการลบ

3) ยังคงกดปุ่ม [STEP/MOVE] ค้างไว้แล้วกดปุ่ม [DEL/ ←] หนึ่งครั้งเพื่อยืนยันการลบ หลังจากนั้นจะได้ยินเสียงบี๊บและที่ด้านล่างหน้าจอจะปรากฏข้อความ ‘EXECUTING’ เพื่อแจ้งว่าได้ทำการลบตำแหน่งเรียบร้อยแล้ว

4.1.4 **วิธีการบันทึกตำแหน่งทับตำแหน่งเดิม** (replace)

1) ให้หน้าจอปรากฏตำแหน่งที่ต้องการแทนที่ด้วยตำแหน่งอื่น โดยใส่ตำแหน่งที่ต้องการแทนที่ แล้วกดปุ่ม [INP/ EXE] เช่น ดังในรูป 4.1.3a

2) กดปุ่ม [STEP/MOVE] ค้างไว้แล้วกดปุ่ม [RPL/ ↓] หนึ่งครั้งแล้วปล่อย จะได้ยินเสียงบี๊บ 3 ครั้ง เพื่อเป็นการเตือน และจะปรากฏข้อความเตือนดังรูป 4.1.4a



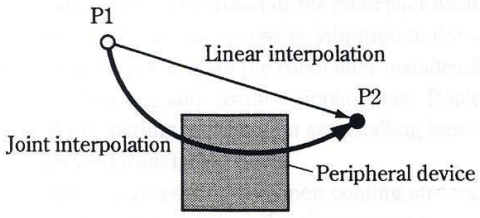
รูปที่ 4.1.4a: หน้าจอ TB แสดงข้อความเตือนการแทนที่

3) ยังคงกดปุ่ม [STEP/MOVE] ค้างไว้แล้วกดปุ่ม [RPL/ ↓] หนึ่งครั้งเพื่อยืนยันการแทนที่ หลังจากนั้นจะได้ยินเสียงบี๊บและที่ด้านล่างหน้าจอจะปรากฏข้อความ ‘EXECUTING’ เพื่อแจ้งว่าได้ทำการแทนที่ตำแหน่งเรียบร้อยแล้วโดยได้นำตำแหน่งที่อยู่ถัดไปมาเป็นตำแหน่งที่ถูกแทนที่ แล้ว โดยที่ตำแหน่งที่อยู่ถัดไปยังคงอยู่

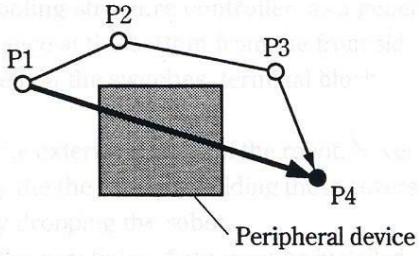
4.1.5 การปรับค่าความเร็วด้วย TB

1) ใช้มือซ้ายยังกด Deadman switch พร้อมกับปุ่ม [STEP/MOVE] อยู่ ใช้มือขวากดปุ่ม [+/FORWD] เพื่อปรับค่าความเร็ว override ให้สูงขึ้น หรือ [-/BACKWD] เพื่อปรับค่าความเร็ว override ให้ลดลง

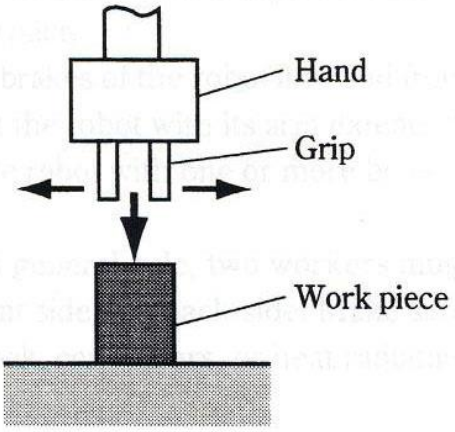
การกำหนดตำแหน่งโดยใช้ TB นั้นมีแนวทางเช่นเดียวกับแนวทางในซอฟต์แวร์ซึ่งต้องมีการวางแผนดังตัวอย่างที่ 2.1 อย่างไรก็ตาม ในการกำหนดตำแหน่งให้กับหุ่นยนต์จริงๆ นั้นมีข้อควรระวังเกี่ยวกับความปลอดภัยมากกว่าในซอฟต์แวร์ ซึ่งต้องทำการวางแผนให้รอบคอบก่อนปฏิบัติงานจริงทุกครั้ง รูป 4.1.6a, b และ c แสดงตัวอย่างให้เห็นความผิดพลาดที่เกิดขึ้นในระหว่างการกำหนดตำแหน่ง



รูปที่ 4.1.6a: แสดงการใช้การเคลื่อนที่ผิดประเภท การเคลื่อนที่จาก P1 ไป P2 สามารถทำได้ 2 วิธี คือ การเคลื่อนที่แบบคำนวณตามแกน และการเคลื่อนที่แบบเส้นตรง แต่การกำหนดตำแหน่งให้กับหุ่นยนต์ต้องคำนึงถึงอุปกรณ์ที่อยู่ในบริเวณรอบๆ หุ่นยนต์ด้วย ในที่นี้ ถ้าใช้การเคลื่อนที่แบบคำนวณตามแกน หุ่นยนต์จะเคลื่อนที่ชนอุปกรณ์



รูปที่ 4.1.6b: แสดงการใช้เส้นทางที่ไม่เหมาะสม การเคลื่อนที่จาก P1 ไป P4 มี 2 เส้นทาง คือ เคลื่อนที่จาก P1, P2, P3 และ P4 และ เคลื่อนที่จาก P1 ถึง P4 ซึ่งผู้เรียนต้องเลือกเส้นทางที่หุ่นยนต์จะไม่ชนกับอุปกรณ์อื่นในบริเวณรอบๆ หุ่นยนต์ ในที่นี้เส้นทางแรกเป็นเส้นทางที่เหมาะสมที่สุด

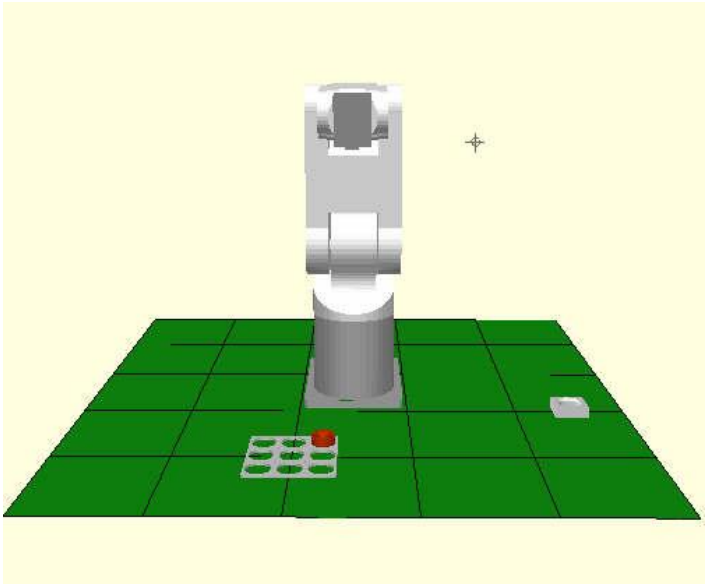


รูปที่ 4.1.6c: แสดงการไม่ตรวจสอบสภาวะของกริปเปอร์ก่อนหยิบชิ้นงาน ในที่นี้กริปเปอร์ปิดอยู่ แต่กำลังเคลื่อนที่เข้าหาชิ้นงาน ซึ่งจะทำให้มีการชนเกิดขึ้น

แบบฝึกหัดที่ 4.1a: จงกำหนดตำแหน่งใดๆ 5 ตำแหน่งให้กับหุ่นยนต์โดยใช้ TB

แบบฝึกหัดที่ 4.1b: จงกำหนดตำแหน่งให้กับหุ่นยนต์โดยใช้ TB และใช้สภาพแวดล้อมที่มีลักษณะเหมาะสำหรับการหยิบจับชิ้นงานจากสถานที่หนึ่งไปวางอีกสถานที่หนึ่ง (Pick and place) วัตถุประสงค์ของแบบฝึกหัดนี้ต้องการให้ผู้เรียนสามารถกำหนดตำแหน่งให้หุ่นยนต์เพื่อการจับชิ้นงานจากที่หนึ่งไปวางอีกที่หนึ่งได้

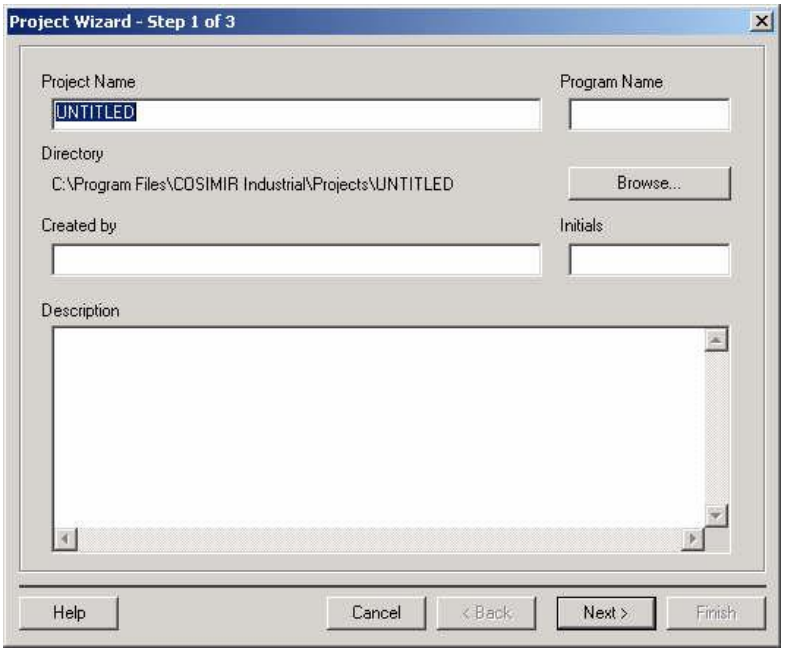
หมายเหตุ: การจัดสภาพแวดล้อมขึ้นอยู่กับชุดฝึกและอุปกรณ์ที่มีอยู่ ถ้าสภาพแวดล้อมของผู้เรียนมีสภาพที่เอื้ออำนวยต่อการหยิบจับชิ้นงานก็ไม่มีความจำเป็นต้องจัดสภาพแวดล้อมใหม่ อย่างไรก็ตาม ในกรณีที่สภาพแวดล้อมไม่เหมาะสม อาจจัดสภาพแวดล้อมดังรูปที่ 4.1.5a



รูปที่ 4.1.5a: ตัวอย่างสภาพแวดล้อมสำหรับแบบฝึกหัดที่ 4.1b

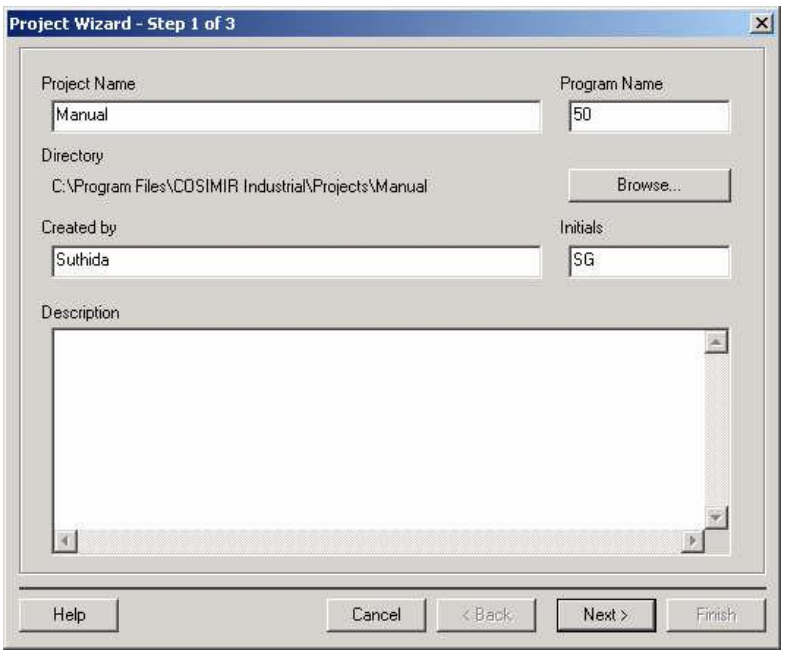
4.2 การเขียนโปรแกรม

1. หลังจากเปิดโปรแกรม COSIMIR Industrial แล้ว ใช้คำสั่ง File> Project Wizard จะปรากฏหน้าต่างดังรูปที่ 4.2.1a หมายเหตุ: เสียบ Hardlock ขณะใช้งานโปรแกรม COSIMIR Industrial ตลอดเวลา



รูปที่ 4.2.1a: หน้าต่าง Project Wizard

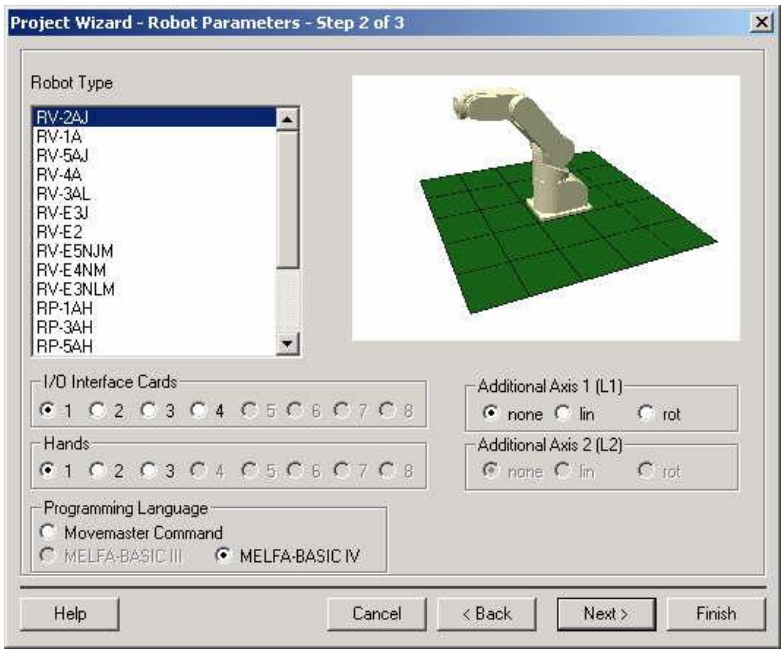
2. ใส่ชื่อ Project Name ซึ่งชื่อของ Project Name จะเป็นชื่อของโมเดล Position Lists และโปรแกรมโดยอัตโนมัติ ทั้งสามไฟล์จะมีชื่อไฟล์เหมือนกันแต่คนละนามสกุล ซึ่งทั้งสามไฟล์จะถูกเก็บไว้ที่ C:\Program Files\COSIMIR Industrial\Projects ซึ่งสามารถเปลี่ยนสถานที่เก็บไฟล์เหล่านี้ได้
ใส่ชื่อโปรแกรม ใส่ชื่อผู้เขียนโปรแกรมในช่อง 'Created by' และชื่อย่อผู้เขียนโปรแกรมในช่อง 'Initials' ดังตัวอย่างในรูป 4.2.1b



รูปที่ 4.2.1b: หน้าต่าง Project Wizard

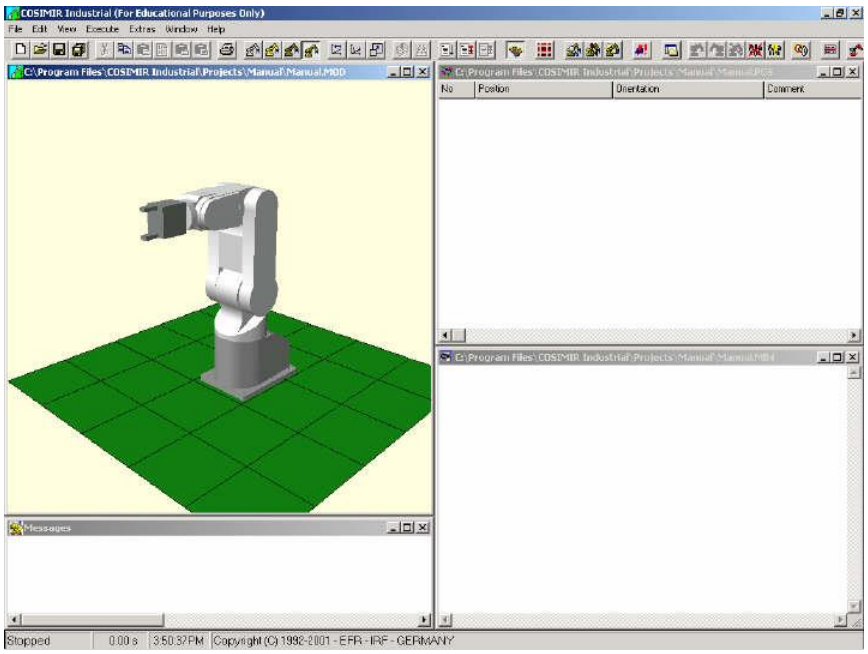
3. หลังจากนั้นกดปุ่ม Next แล้วเลือกรุ่นของหุ่นยนต์, จำนวนการ์ดอินพุทเอาท์พุท (I/O interface card), จำนวนกริปเปอร์ และ ภาษาที่จะใช้ในการเขียนโปรแกรม ซึ่งข้อมูลเหล่านี้จะต้องตรงกับฮาร์ดแวร์ที่มีอยู่ดังตัวอย่างในรูป 4.2.1c (หมายเหตุ: ในบทนี้จะกล่าวถึง Melfa Basic IV เท่านั้น)

รูปที่ 4.2.1c: หน้าต่าง Project Wizard



4. หลังจากนั้นกดปุ่ม Finish แล้วจะปรากฏหน้าต่างดังรูปที่ 4.2.1d ซึ่งจะปรากฏหน้าต่างทั้งหมด 4 หน้าต่างคือ หน้าต่างโมเดล Position Lists โปรแกรม และ message ซึ่งเป็นหน้าต่างรายงานผลการคอมไพล์และการติดต่อสื่อสารกับคอนโทรลเลอร์

รูปที่ 4.2.1d:



5. เขียนโปรแกรมโดยใช้หลักการวางแผนในการเขียนโปรแกรมเช่นเดียวกันกับหัวข้อ 3.3 เรื่องขั้นตอนใน การเขียนโปรแกรม เมื่อเขียนโปรแกรมเสร็จแล้วให้บันทึกโปรแกรมโดยคลิ๊กที่หน้าต่างโมเดลแล้วใช้คำสั่ง File > Save all ทั้งสามหน้าต่างคือ โมเดล ตำแหน่ง และโปรแกรมจะถูกบันทึกทั้งหมด

6. เมื่อเขียนโปรแกรมเสร็จแล้ว ให้อัพโหลดตำแหน่งขึ้นมาโดยใช้วิธีการในหัวข้อ 4.3.1

7. ใช้คำสั่ง File > Save all

8. ขั้นตอนต่อไปคือขั้นตอนของ Project Management โดยใช้คำสั่ง Execute > Project Management แล้วเลือกแท็บ Files เพื่อตรวจสอบว่ามีไฟล์โปรแกรมและตำแหน่งหรือไม่ หลังจากนั้นคอมไพล์โปรเจคท์

9. ดาวน์โหลดโปรแกรมโดยใช้วิธีการในหัวข้อ 4.3.1

10. รันโปรแกรมจากคอมพิวเตอร์โดยใช้คำสั่ง Execute > Program Start และหยุดการรันโปรแกรมโดยใช้คำสั่ง Execute > Program Stop

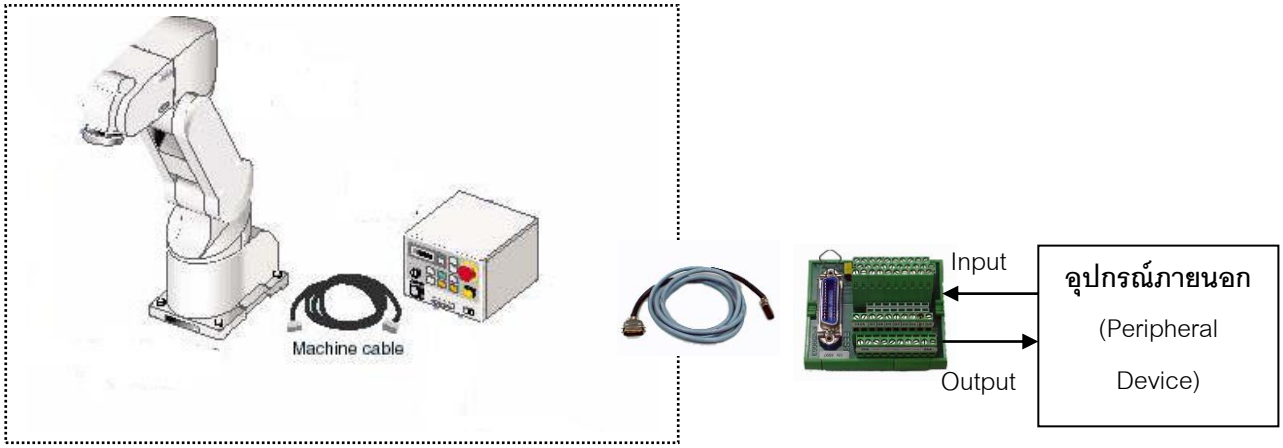
หมายเหตุ: การรันโปรแกรมอีกวิธีหนึ่ง คือ การรันจากคอนโทรลเลอร์ซึ่งอธิบายไว้ในหัวข้อ 4.4.1

11. เมื่อต้องการเปิด project ที่ชื่อ manual ขึ้นมาใหม่ให้ไปที่คำสั่ง File > Open เข้าไปในไดเร็กทอรี่ที่ COSIMIR Industrial ติดตั้งอยู่จะปรากฏหน้าต่างดังรูปที่ 4.2.1e ดับเบิ้ลคลิ๊กที่ไดเร็กทอรี่ ‘Projects’ ดังรูปที่ 4.2.1f จะพบว่ามี Project ต่างๆ ถูกบันทึกไว้ ดับเบิ้ลคลิ๊กที่ชื่อโปรเจคท์ที่ต้องการเปิด ในที่นี้คือ ‘Manual’ จะปรากฏไฟล์ Manual.MOD หลังจากนั้นจะปรากฏหน้าต่างโมเดล ตำแหน่งและโปรแกรมขึ้นพร้อมกัน

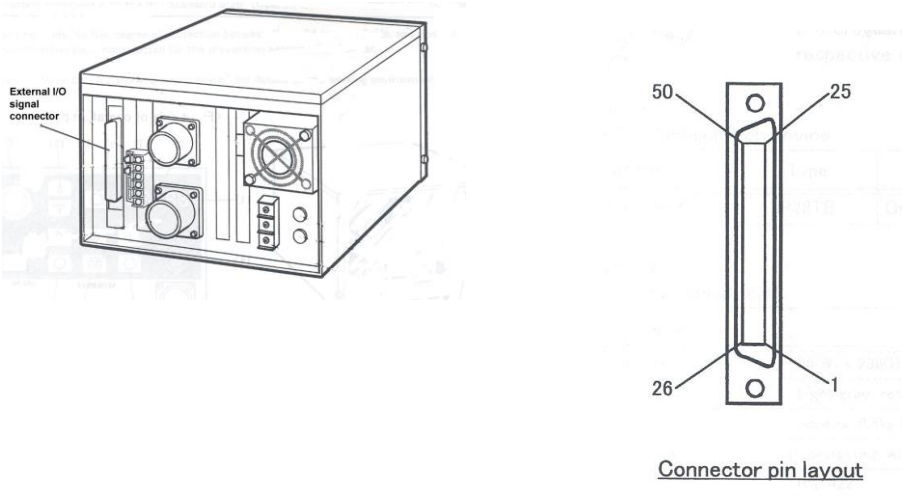
แบบฝึกหัดที่ 4.2a: จงเขียนโปรแกรมโดยใช้ตำแหน่งจากแบบฝึกหัดที่ 4.1a โดยให้หุ่นยนต์เคลื่อนที่จากจุดที่ 1 ไปจุดที่ 5

แบบฝึกหัดที่ 4.2b: จงเขียนโปรแกรมโดยใช้ตำแหน่งจากแบบฝึกหัดที่ 4.1b โดยให้หุ่นยนต์เคลื่อนที่หยิบชิ้นงานจากสถานที่หนึ่งไปวางอีกสถานที่หนึ่ง

ในบทที่ผ่านมาเราเรียนซอฟต์แวร์ COSIMIR Educational ซึ่งสามารถเขียนโปรแกรมที่มีการส่งข้อมูลไปมาระหว่างหุ่นยนต์และอุปกรณ์ภายนอก และการเชื่อมต่อระหว่างหุ่นยนต์และอุปกรณ์ภายนอกสามารถกระทำในซอฟต์แวร์ได้ ซึ่งซอฟต์แวร์ COSIMIR Industrial ก็มีคุณสมบัตินี้ อย่างไรก็ตาม ในบทนี้เราจะให้ความสำคัญกับการเชื่อมต่อระหว่างหุ่นยนต์และอุปกรณ์ภายนอกโดยใช้สายไฟเชื่อมต่อกันจริงๆ ซึ่งเราจำเป็นต้องทราบระบบที่เรามีอยู่ ซึ่งมีลักษณะดังรูป 4.2.2a ระบบประกอบไปด้วยคอนโทรลเลอร์และหุ่นยนต์ ซึ่งระบบนี้ติดต่อสื่อสารกับอุปกรณ์ภายนอกโดยผ่านสายสื่อสารแบบขนาน ต่อจาก External I/O signal connection ที่อยู่ด้านหลังของคอนโทรลเลอร์ซึ่งเป็นพอร์ทแบบ 50 พิน (ดูรูป 4.2.2b) ไปยังพอร์ทของ I/O terminalซึ่งมี 24 พิน และจากไป I/O terminal ไปยังอุปกรณ์ภายนอก I/O terminal เปรียบเสมือนตัวกลางระหว่างระบบและอุปกรณ์ภายนอกทำให้การเชื่อมต่อสายไฟทำได้ง่ายและสะดวกขึ้น ในกรณีที่ไม่มี I/O terminal อุปกรณ์ภายนอกจะต้องต่อกับ External I/O signal connection โดยตรงซึ่งเป็นพอร์ทแบบ 50 ขา ซึ่งการเชื่อมต่อแบบนี้ไม่เหมาะสมและไม่ปลอดภัย จึงต้องมี I/O terminal เพื่อเพิ่มความสะดวก



รูป 4.2.2a: รูปแสดงระบบการเชื่อมต่อหุ่นยนต์กับอุปกรณ์ภายนอกผ่าน I/O terminal



รูป 4.2.2b: รูปแสดงด้านหลังของคอนโทรลเลอร์ซึ่งมีพอร์ทแบบ 50 ขา

ตารางที่ 4.2.1: **ตารางแสดงความสัมพันธ์ระหว่างอินพุทและเอาท์พุทบิท หมายเลขบิทและพินของพอร์ท** I/O terminal

Input bit	bit number	Pin		Output bit	bit number	Pin
0	8	13		0	8	1
1	9	14		1	9	2
2	10	15		2	10	3
3	11	16		3	11	4
4	12	17		4	12	5
5	13	18		5	13	6
6	14	19		6	14	7
7	15	20		7	15	8

พินที่เหลือถูกใช้สำหรับการจ่ายไฟให้กับ I/O terminal ซึ่งรายละเอียดความสัมพันธ์ทั้งหมดอยู่ในตารางที่ 1 ใน ภาคผนวก หน้า A-15

ตารางที่ 4.2.2: **ตารางแสดงความสัมพันธ์ระหว่างอินพุทและเอาท์พุทบิท หมายเลขบิทและพินของพอร์ทของคอนโทรลเลอร์**

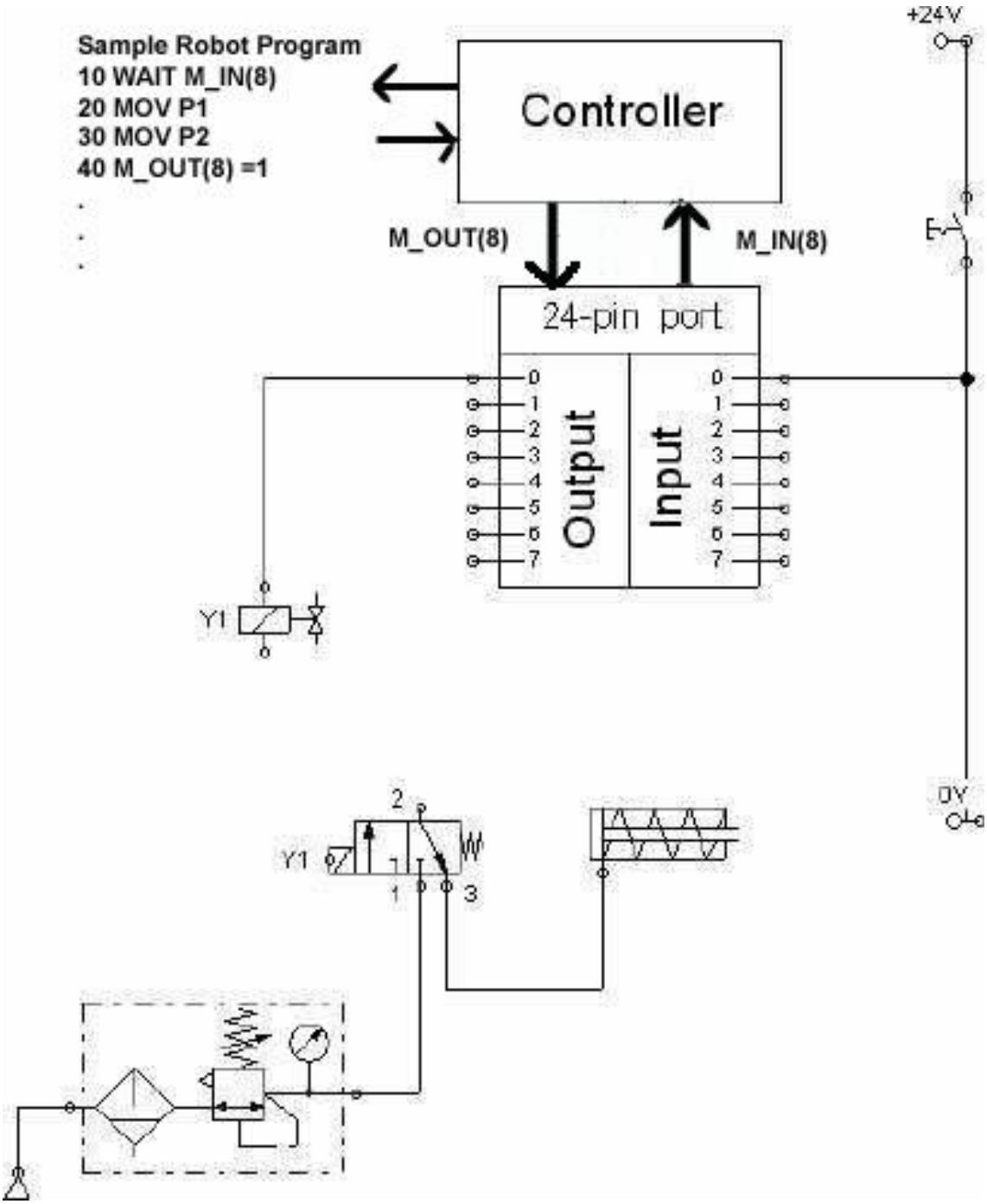
Input bit	bit number	Pin		Output bit	bit number	Pin
0	8	40		0	8	10
1	9	41		1	9	11
2	10	42		2	10	12
3	11	43		3	11	13
4	12	44		4	12	35
5	13	45		5	13	36
6	14	46		6	14	37
7	15	47		7	15	38

พินที่เหลือถูกใช้สำหรับการจ่ายไฟให้กับคอนโทรลเลอร์ ควบคุมเซอร์โวมอเตอร์ เป็นต้น ซึ่งรายละเอียดความสัมพันธ์ทั้งหมดอยู่ในตารางที่ 2 ใน ภาคผนวก หน้า A-16

หมายเหตุ: การที่เราจะทราบว่าอินพุทบิทไหนวิ่งผ่านพินไหนแล้วเป็นบิทหมายเลขใดในการเขียนโปรแกรมนั้นขึ้นอยู่กับการ์ด I/O ที่ติดตั้งในคอนโทรลเลอร์

รูป 4.2.2c แสดงตัวอย่างการเชื่อมต่อระหว่างระบบกับอุปกรณ์ภายนอก ระบบรับสัญญาณอินพุทจาก start button และส่งสัญญาณเอาท์พุทให้วาล์ว start button เป็นต่อเข้ากับอินพุทบิทที่ 0 ของ I/O terminal ซึ่งเป็นบิทหมายเลข 8 (M_IN(8)) เป็นหมายเลขสำหรับอ้างอิงในการเขียนโปรแกรม (ดูตาราง 4.2.1 ประกอบ) เมื่อกดปุ่มสตาร์ทจะมีสัญญาณค่า

'1' เข้ามาที่อินพุทบิทที่ 0 (M_IN(8) = 1) โดยสัญญาณจะวิ่งผ่านพินที่ 13 ของพอร์ท I/O terminal ผ่านสายสื่อสารแบบขนาน และเข้าสู่คอนโทรลเลอร์โดยที่คอนโทรลเลอร์รับสัญญาณ M_IN(8) = 1 ผ่านอินพุทบิทที่ 0 พินที่ 40 ของคอนโทรลเลอร์ (ดูตาราง 4.2.2 ประกอบ)



รูป 4.2.2c: รูปแสดงตัวอย่างการเชื่อมต่อระหว่างระบบกับอุปกรณ์ภายนอกผ่าน I/O terminal

โซลินอยด์วาล์วต่อเข้ากับเอาท์พุทบิทที่ 0 ของ I/O terminal ซึ่งเป็นบิทหมายเลข 8 (M_OUT(8)) เป็นหมายเลขสำหรับอ้างอิงในการเขียนโปรแกรม (ดูตาราง 4.2.2 ประกอบ) เมื่อคอนโทรลเลอร์ส่งสัญญาณค่า '1' สัญญาณจะวิ่งจากพินที่ 10 ของพอร์ทคอนโทรลเลอร์เข้าพินที่ 1 ของพอร์ท I/O terminal (ดูตาราง 4.2.1 ประกอบ) ผ่านทางเอาท์พุทบิทที่ 0 (M_OUT(8) = 1) โซลินอยด์วาล์วได้รับสัญญาณค่า '1' วาล์วก็เกิดการเปลี่ยนห้อง ลมอัดเคลื่อนที่เข้าสู่กระบอกสูบ เมื่อคอนโทรลเลอร์ส่งสัญญาณค่า '0' วาล์วและกระบอกสูบจะกลับสู่สภาวะปกติ

แบบฝึกหัดที่ 4.2c: จงเขียนโปรแกรมโดยใช้ตำแหน่งจากแบบฝึกหัดที่ 4.1b และใช้ start button และวาล์วและกระบอกสูบเป็นอินพุทและเอาท์พุทตามลำดับ โดยมีเงื่อนไขดังต่อไปนี้ เมื่อกดปุ่ม start button หุ่นยนต์เริ่มทำงานเคลื่อนที่ไปหยิบชิ้นงานจาก pallet ใส่ socket เมื่อวางชิ้นงานเสร็จเรียบร้อยแล้วกลับสู่ตำแหน่งเริ่มต้น แล้วกระบอกสูบเคลื่อนที่ออกและกลับ จบหนึ่งรอบการทำงาน

4.3 การแลกเปลี่ยนข้อมูลระหว่างคอนโทรลเลอร์และคอมพิวเตอร์

การแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์และคอนโทรลเลอร์ผ่านสาย personal computer cable การแลกเปลี่ยนข้อมูลทำได้สองทาง คือ จากคอมพิวเตอร์สู่คอนโทรลเลอร์ (download) และจากคอนโทรลเลอร์สู่คอมพิวเตอร์ (upload)

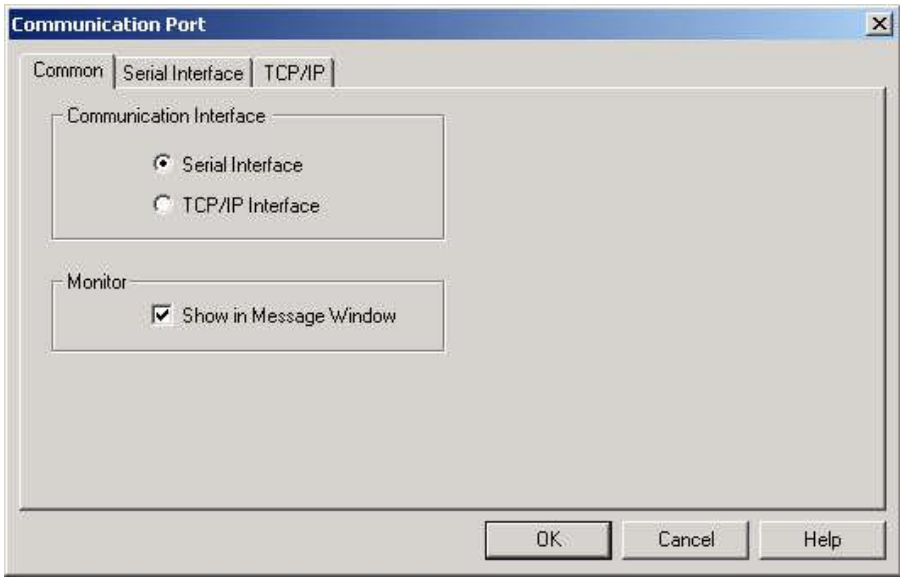
4.3.1 ขั้นตอนการอัพโหลดและดาวน์โหลด

1) เสียบสาย personal computer cable ที่ serial port ของคอมพิวเตอร์เข้ากับคอนโทรลเลอร์เพื่อการสื่อสารระหว่างคอมพิวเตอร์และคอนโทรลเลอร์

2) บิดกุญแจที่คอนโทรลเลอร์มาที่ตำแหน่ง 'AUTO (Ext)'

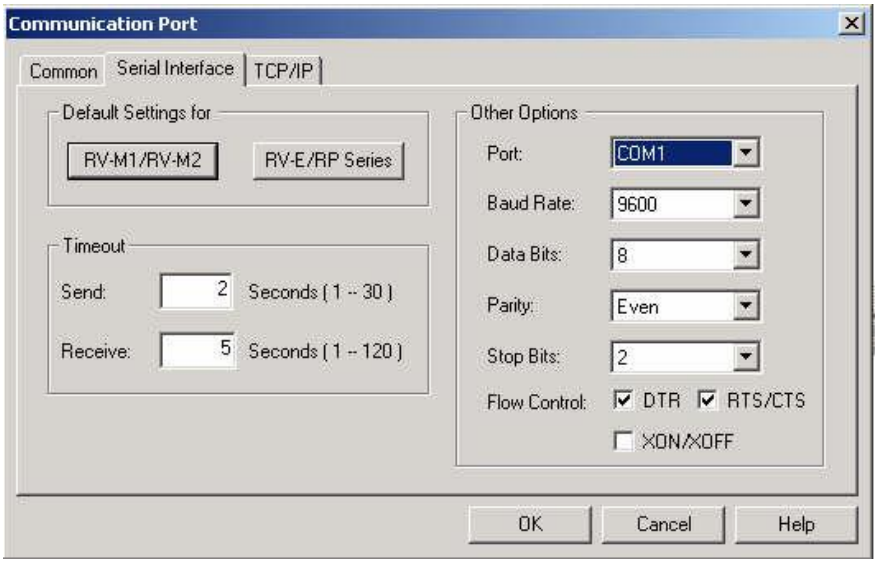
3) Extras > Settings > Communication Port รูป 4.3.1a

รูปที่ 4.3.1a:
หน้าต่าง Communication Port



4) เลือกแท็บ 'Serial Interface' รูป 4.3.1b พอร์ท 'COM1' กดปุ่ม OK

รูปที่ 4.3.1b:
หน้าต่าง Communication Port



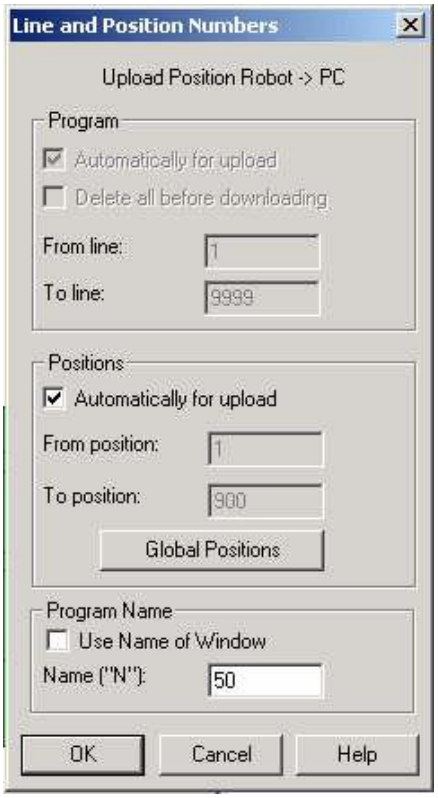
5) Execute > Init Connection เมื่อการติดต่อระหว่างคอนโทรลเลอร์และคอมพิวเตอร์สำเร็จจะปรากฏหน้าต่างดังรูป 4.3.1c ให้กดปุ่ม OK และจะมีเครื่องหมายถูกอยู่ด้านหน้าคำสั่ง Init Connection

รูปที่ 4.3.1c:

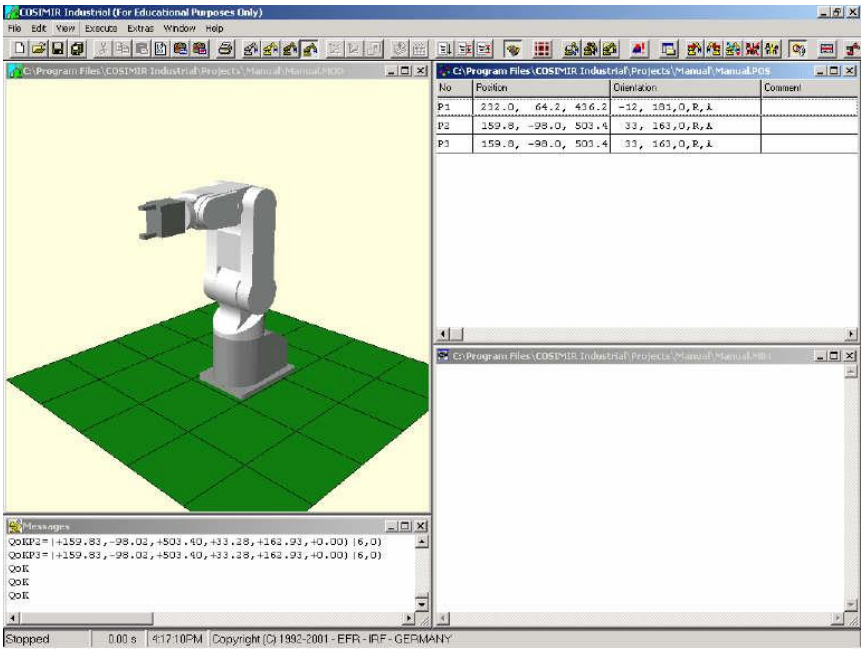
6) Activate หน้าต่างที่ต้องการดาวน์โหลดหรืออัพโหลด โดยคลิ๊กที่หน้าต่างนั้นๆ ในตัวอย่างนี้จะอัพโหลดตำแหน่ง

7) ใช้คำสั่ง Execute > Upload Robot -> PC เพื่อการอัพโหลด หรือ Execute > Download Pc-> Robot เพื่อการดาวน์โหลด จะปรากฏหน้าต่างดังรูป 4.3.1d แล้วระบุหมายเลขโปรแกรมของตำแหน่งที่กำหนดไว้ในช่อง Name(“N”)

หลังจากนั้นกดปุ่ม OK การถ่ายโอนข้อมูลจะเริ่มต้น และเมื่อการถ่ายโอนข้อมูลเสร็จสิ้นจะปรากฏตำแหน่งในหน้าต่าง Position list ดังรูป 4.3.1e



รูปที่ 4.3.1d:



รูปที่ 4.3.1e:

8) เมื่อต้องการยกเลิกการติดต่อสื่อสาร ให้ใช้คำสั่ง Execute > Init Connection ก่อนถอดสาย personal computer cable

4.3.2 การกำหนดตำแหน่งบนซอฟต์แวร์ด้วย Jog operation

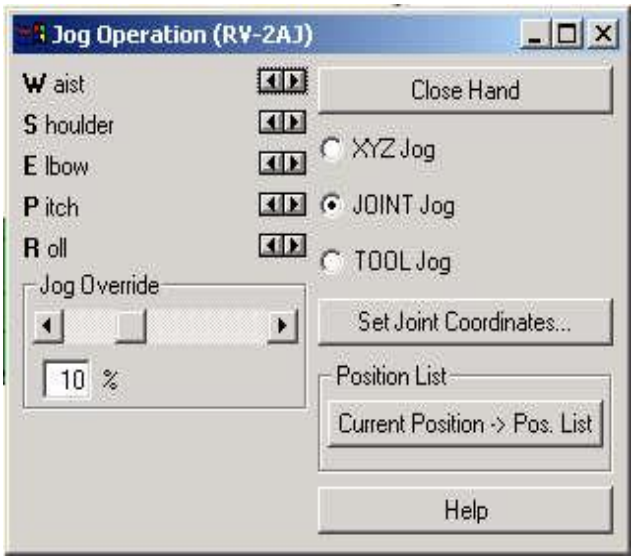
เป็นฟังก์ชั่นสำหรับการกำหนดตำแหน่งบนซอฟต์แวร์โดยที่หุ่นยนต์จริงจะเคลื่อนที่ไปด้วย ขั้นตอนการกำหนดตำแหน่งบนซอฟต์แวร์มีดังต่อไปนี้

ขั้นตอนที่ 1) – 5) เหมือนในหัวข้อ 4.3.1

6) ใช้คำสั่ง Execute > Jog operation จะปรากฏหน้าต่างดังรูป 4.3.2a

หมายเหตุ: รูป 4.3.2b คือ หน้าต่างการกำหนดตำแหน่งบนซอฟต์แวร์ที่เคยใช้กันใน COSIMIR Education ซึ่งหุ่นยนต์จริงจะไม่เคลื่อนที่ตามไปด้วย การเรียกหน้าต่างนี้ใช้คำสั่ง Extras > Teach-In แต่อย่างไรก็ตามฟังก์ชั่นการทำงานของปุ่มต่างๆ ยังคงเหมือนเดิม

รูปที่ 4.3.2a: หน้าต่างสำหรับกำหนดตำแหน่งบนซอฟต์แวร์โดยที่หุ่นยนต์เคลื่อนที่ตามไปด้วย



รูปที่ 4.3.2b: หน้าต่างสำหรับกำหนดตำแหน่งบนซอฟต์แวร์โดยที่หุ่นยนต์ไม่เคลื่อนที่ตามไปด้วย

ขั้นตอนการให้หุ่นยนต์แสดงตำแหน่งที่กำหนดบนซอฟต์แวร์มีดังต่อไปนี้

1) – 5) เหมือนในหัวข้อ 4.3.1

6) ดับเบิ้ลคลิ๊กที่ตำแหน่งที่ต้องการให้หุ่นยนต์แสดงผล

7) ใช้คำสั่ง Execute > COSIMIR Position -> Robot แล้วหุ่นยนต์จะเคลื่อนที่ไปยังตำแหน่งที่ระบุไว้

ขั้นตอนการให้โมเดลแสดงตำแหน่งของหุ่นยนต์มีดังต่อไปนี้

1) – 5) เหมือนในหัวข้อ 4.3.1

6) ใช้ Jog operation ดังรูป 4.3.2a เคลื่อนที่หุ่นยนต์จริง

7) ใช้คำสั่ง Execute > Robot Position -> COSIMIR แล้วโมเดลจะเคลื่อนที่ไปยังตำแหน่งของหุ่นยนต์จริง

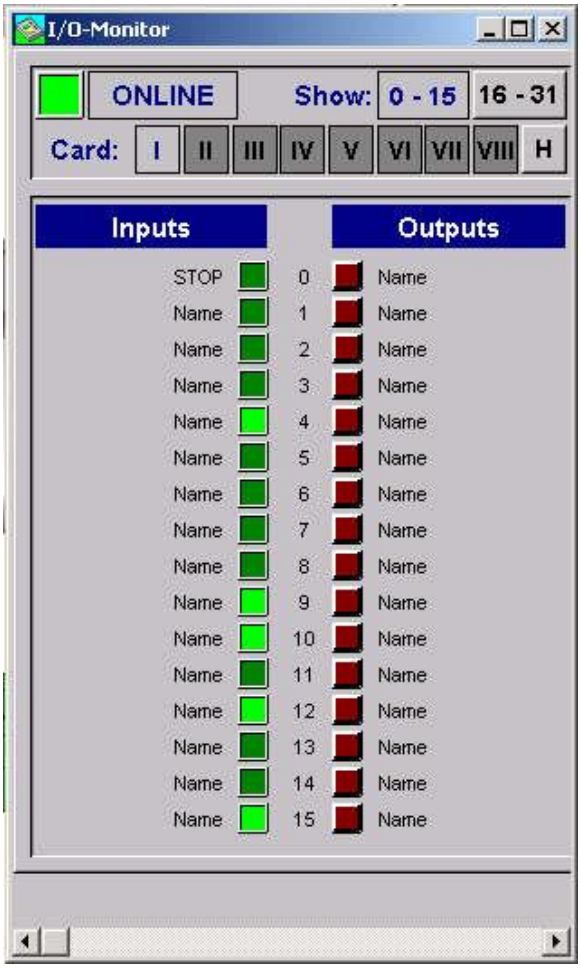
4.3.3 การดูสภาวะการทำงานของหุ่นยนต์

A) การดูสภาวะอินพุทและเอาท์พุท (I/O Monitor)

เป็นฟังก์ชั่นในการมอนิเตอร์ดูสภาวะอินพุทและเอาท์พุท ซึ่งมีขั้นตอนในการเรียกดูดังต่อไปนี้

ขั้นตอนที่ 1) ถึง 5) เหมือนในหัวข้อ 4.3.1

6) ใช้คำสั่ง Extras > I/O Monitor แล้วจะปรากฏหน้าต่าง I/O Monitor ดังรูป 4.3.3a



รูปที่ 4.3.3a: หน้าต่าง I/O Monitor สำหรับดูสภาวะอินพุทและเอาท์พุท

B) **การดูสภาวะการทำงานของหุ่นยนต์** (Monitor function)

การดูสภาวะการทำงานของหุ่นยนต์สามารถกระทำได้โดยใช้คำสั่ง Execute > Monitor Functions หน้าต่างดังรูป 4.3.3b จะปรากฏขึ้น โดยที่ 5 ขั้นตอนแรกเหมือนกับในหัวข้อ 4.3.1

รูปที่ 4.3.3b: หน้าต่าง Monitor สำหรับดูสภาวะการทำงานต่างๆ ของหุ่นยนต์

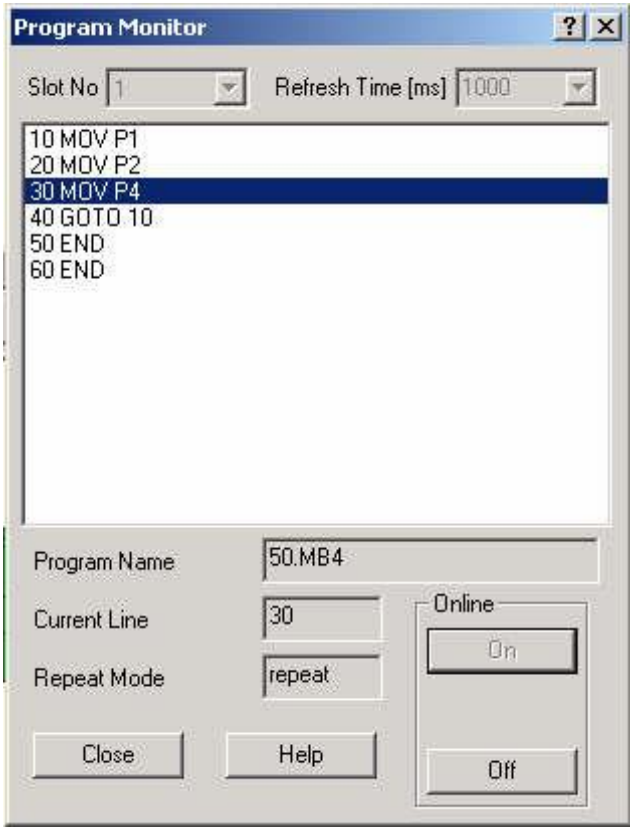
จากหน้าต่าง Monitor จะเห็นว่าเราสามารถดูสภาวะการทำงานของหุ่นยนต์ได้หลายแง่มุมหรือตัวแปร ดังตารางที่ 4.3.2

ตารางที่ 4.3.2: **ตารางแสดงการทำงานของปุ่มต่าง ๆ บนหน้าต่าง** Monitor

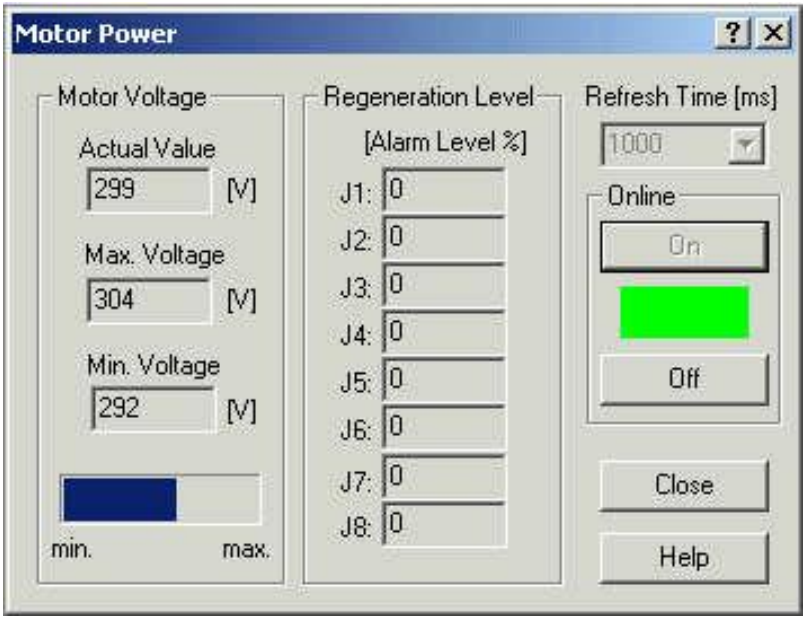
ปุ่ม	ชื่อ	หน้าที่
	Program-monitor	มอนิเตอร์สถานะของโปรแกรมที่รันอยู่
	Slot state monitor	
	Position monitor	มอนิเตอร์ค่าของตำแหน่งในเชิงมุมของข้อต่อและแนวแกนของหุ่นยนต์
	Motor speed	มอนิเตอร์ค่าความเร็วของเซอร์โวมอเตอร์
	Droop	
	Encoder	มอนิเตอร์ค่า encoder ของเซอร์โวมอเตอร์
	Motor load	มอนิเตอร์ค่า encoder ของเซอร์โวมอเตอร์
	Current1	มอนิเตอร์ค่ากระแสไฟฟ้าของเซอร์โวมอเตอร์
	Current2	มอนิเตอร์ค่ากระแสไฟฟ้าของเซอร์โวมอเตอร์
	Voltage	มอนิเตอร์ค่าแรงดันกระแสไฟฟ้าของเซอร์โวมอเตอร์

ในที่นี้จะขอยกตัวอย่างการใช้งานบางปุ่ม เช่น การมอนิเตอร์สถานะของโปรแกรมที่รันอยู่ หลังจากที่กดปุ่ม Program-monitor จะปรากฏหน้าต่าง Program Monitor ดังรูป 4.3.3c แล้วกดปุ่ม On เพื่อออนไลน์ดูสถานะของการรันโปรแกรม ในที่นี้โปรแกรม ชื่อ 50.MB4 รันถึงบรรทัดที่ 30 แล้ว Refresh time หมายถึง ช่วงเวลาในการอัพเดทข้อมูล ในที่นี้ Refresh time ถูกตั้งไว้ที่ 1000 ms ซึ่งหมายความว่าทุกๆ 1 วินาทีจะมีการอัพเดทข้อมูล การใช้งานนี้เหมาะสำหรับการมอนิเตอร์และหา error ในโปรแกรมแบบ multitask ได้ง่ายขึ้นซึ่งเป็นโปรแกรมที่ควบคุมการทำงานของอุปกรณ์หลายๆ ตัวพร้อมๆ กันในโรงงาน เช่น ควบคุมหุ่นยนต์และสายพาน การมอนิเตอร์ค่าตัวแปรอื่นก็สามารถกระทำได้เช่นเดียวกันกับการมอนิเตอร์สถานะของโปรแกรม เช่น การมอนิเตอร์ค่าแรงดันไฟฟ้าดังรูป 4.3.3d

.



รูปที่ 4.3.3c: หน้าต่าง Program Monitor สำหรับดูสภาวะการรันโปรแกรมของหุ่นยนต์



รูปที่ 4.3.3d: หน้าต่าง Motor Power สำหรับดูแรงดันไฟฟ้าของเซอร์โว

4.4 การใช้งานคอนโทรลเลอร์

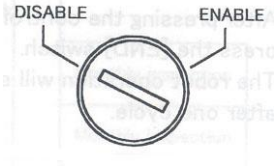
4.4.1 การเริ่มต้นสตาร์ทคอนโทรลเลอร์ในโหมดอัตโนมัติ

ก่อนเริ่มต้นสตาร์ทคอนโทรลเลอร์ในโหมด AUTO (Op) หรือโหมดอัตโนมัติ ควรทำการตรวจสอบสิ่งต่อไปนี้ เพื่อป้องกันความเสียหายต่อบุคคลและทรัพย์สิน

- ตรวจสอบว่าไม่มีบุคคลอยู่ในบริเวณพื้นที่ทำงานของหุ่นยนต์
- ตรวจสอบว่าไม่มีวัตถุใดๆ ที่ไม่จำเป็นภายในบริเวณพื้นที่ทำงานของหุ่นยนต์
- ตรวจสอบว่าชิ้นงานวางอยู่ในตำแหน่งที่กำหนดไว้
- ตรวจสอบให้แน่ใจว่าโปรแกรมทำงานถูกต้อง

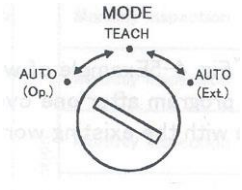
ขั้นตอนในการสตาร์ทคอนโทรลเลอร์ในโหมดอัตโนมัติ มีดังต่อไปนี้

1) บิดกุญแจที่ TB มาที่ตำแหน่ง ‘DISABLE’ ดังรูปที่ 4.4.1a



รูปที่ 4.4.1a: บิดกุญแจที่ TB มาที่ตำแหน่ง ‘DISABLE’

2) บิดกุญแจที่คอนโทรลเลอร์มาที่ตำแหน่ง ‘AUTO (Op)’ ดังรูปที่ 4.4.1b

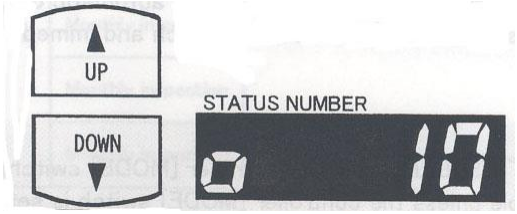


รูปที่ 4.4.1b: บิดกุญแจหน้าแผงคอนโทรลเลอร์มาที่ตำแหน่ง ‘DISABLE’

3) กดปุ่ม [CHANG DISP] ที่หน้าแผงคอนโทรลเลอร์สองครั้ง หน้าจอจะปรากฏความเร็ว override ดังรูป 4.4.1c

รูปที่ 4.4.1c: หน้าจอแสดงผลความเร็ว override

4) กดปุ่ม [UP] หรือ [DOWN] เพื่อเปลี่ยนค่าความเร็ว override ในตัวอย่างเปลี่ยนค่าความเร็วเป็น 10 ดังรูป 4.4.1d



รูปที่ 4.4.1d: ใช้ปุ่ม [UP] หรือ [DOWN] เพื่อเปลี่ยนค่าความเร็ว

5) กดปุ่ม [CHNG DISP] แล้วหน้าจอจะแสดง ‘หมายเลขโปรแกรม’

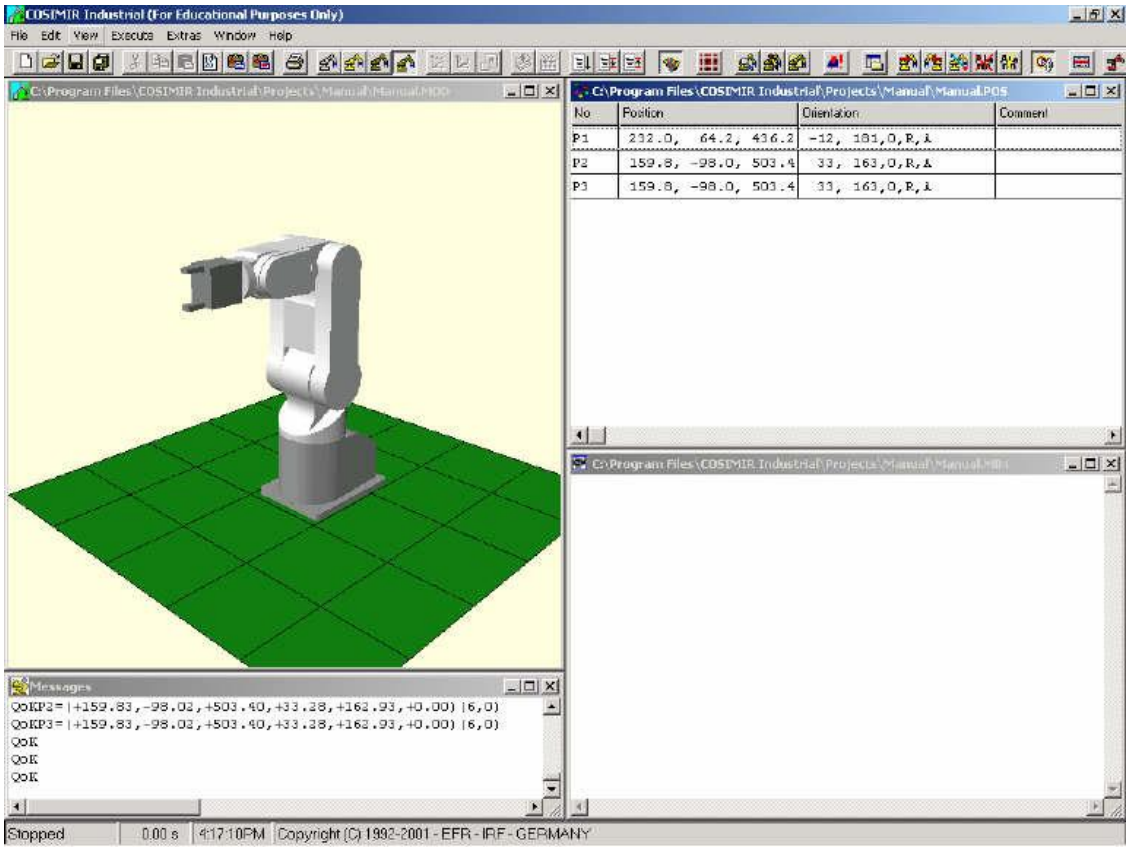
รูปที่ 4.4.1e: หน้าจอแสดงหมายเลขโปรแกรม

6) กดปุ่ม [UP] หรือ [DOWN] เพื่อเลือกโปรแกรมที่ต้องการ ในตัวอย่างเปลี่ยนเป็นโปรแกรมหมายเลข 1 ดังรูป 4.4.1f

รูปที่ 4.4.1f: ใช้ปุ่ม [UP] หรือ [DOWN] เพื่อเลือกโปรแกรมที่ต้องการ

7) กดปุ่ม [START] ที่หน้าแผงคอนโทรลเลอร์เพื่อเริ่มต้นการทำงาน และกดปุ่ม [STOP] เพื่อหยุดการทำงาน

Programming Language (MELFA Basic IV)



Software “COSIMIR Industrial” เป็น Software ลิขสิทธิ์ ที่ใช้สำหรับควบคุมการทำงานของ Robot อุตสาหกรรม โดยมีฟังก์ชันการทำงานที่หลากหลายและซับซ้อนมากกว่า Software “COSIMIR Educational” ที่ใช้เพื่อการศึกษาเท่านั้น ทั้งนี้ฟังก์ชันและความสามารถต่างๆ ภายใน Software COSIMIR Industrial มีดังนี้

- สามารถจำลองการทำงาน (Simulation) ของ Mitsubishi Robot ในรุ่นต่างๆ ได้
- สามารถสร้าง Work Cell ของ Robot ได้
- สามารถสร้าง Mechanical Work Cell เพื่อแสดงผลการจำลองการทำงานได้ เช่น ระบบ Conveyor, Sensors, Actuators, ฯลฯ
- สามารถทำการติดต่อสื่อสารกันระหว่าง Software บนคอมพิวเตอร์ และ Robot เพื่อจะได้เห็นถึงการทำงานจริง

- สามารถเขียนโปรแกรมควบคุมการทำงาน (Programming Language) เช่น *.MB4 และโปรแกรมควบคุมการเคลื่อนที่ (Position List) เช่น *.POS, *.MRL
- สามารถทำการ Download และ Upload โปรแกรมและ File การทำงานต่างๆ ได้

จะเห็นได้ว่า Software “COSIMIR Industrial” มีประโยชน์อย่างมากในการนำไปใช้เพื่อศึกษาถึงพฤติกรรมการทำงาน การเคลื่อนที่ของ Robot รวมถึงการนำไปประยุกต์ใช้งาน Robot จริงในงานอุตสาหกรรม

คำสั่งและการใช้คำสั่งในการเขียนโปรแกรมควบคุมแขนกล MELFA Basic IV

1. สัญลักษณ์ ‘

เมื่อปรากฏสัญลักษณ์ ’ หน้าอักษรใดๆ ในแต่ละบรรทัด หมายความว่าข้อความหลังสัญลักษณ์นั้นเป็นคำอธิบาย จะไม่เกี่ยวข้องกับโปรแกรม

ตัวอย่าง MOV P1 ‘Move to position P1

ข้อความหลังคำสั่งจะเป็นการอธิบายถึงคำสั่งข้างต้นเท่านั้น ไม่มีผลต่อการทำงานในโปรแกรม

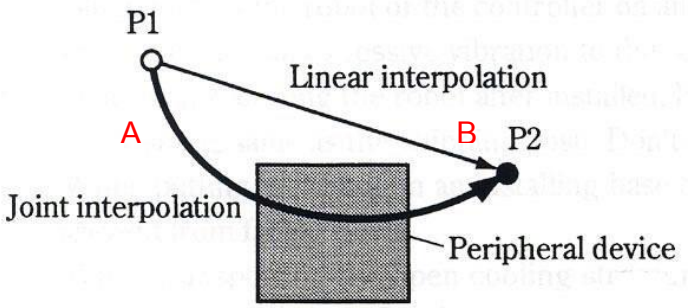
2. Position ในโปรแกรม

หลังจากทำการ Teach Position ที่จำเป็นให้กับ Robot แล้ว ภายในโปรแกรม MELFA Basic IV จะเรียกชื่อตำแหน่ง Position แต่ละ Position เป็น P1, P2, P3 … ตามลำดับไปเรื่อยๆ

3. คำสั่ง MOV… และ MVS… ในการเคลื่อนที่ของ Robot

คำสั่ง MOV เป็นคำสั่งให้ Robot ทำการเคลื่อนที่ไปยังตำแหน่งที่กำหนด โดยหน่วยประมวลผลภายในตัว Robot เองจะทำการคำนวณหาระยะทาง และทิศทางการเคลื่อนที่ที่เหมาะสมที่สุด โดยปกติจะเป็นการเคลื่อนที่ในแนวเส้นโค้ง (Joint interpolation)

คำสั่ง MVS เป็นคำสั่งให้ Robot ทำการเคลื่อนที่ไปยังตำแหน่งที่กำหนดเช่นกัน แต่การใช้คำสั่งเคลื่อนที่แบบ MVS นี้จะเป็นการสั่งให้ Robot เคลื่อนที่จากตำแหน่งปัจจุบันไปยังตำแหน่งที่กำหนดตามแนวเส้นตรง (Linear interpolation) ซึ่งการใช้คำสั่งการเคลื่อนที่เป็นแนวเส้นตรงแบบนี้จะมีประโยชน์อย่างมากในบาง Application เช่นการสั่งให้ Robot เคลื่อนที่ไปตามแกนแนวดิ่ง (Z)



ตัวอย่าง

MOV P2	Robot จะเคลื่อนที่ไปตามเส้นโค้ง A
MVS P2	Robot จะเคลื่อนที่ไปตามเส้นตรง B

4. คำสั่ง IF…THEN…

คำสั่ง IF…THEN เป็นคำสั่งที่ใช้สำหรับการสร้างเงื่อนไขให้โปรแกรมทำงานตามที่กำหนด เมื่อเงื่อนไขเป็นจริงให้ไปทำคำสั่งอย่างหนึ่ง และเมื่อเงื่อนไขเป็นเท็จก็ให้โปรแกรมกระโดดไปทำงานตามคำสั่งอีกอย่างหนึ่ง เช่น การใช้คำสั่ง IF…THEN ในการเช็ค Status Bit ว่าเมื่อไรก็ตามที่ Bit แสดงสถานะเป็น 1 ก็ให้โปรแกรม Robot เคลื่อนที่

ไปยังตำแหน่งต่อไป อย่างไรก็ตามถ้า Bit ยังไม่แสดงสถานะเป็น 1 (ยังตรวจเช็คได้ว่า สถานะเป็น 0) ก็ให้โปรแกรมคงวนลูปเช็คไปเรื่อยๆ

ตัวอย่าง

```
100    IF M_IN(9) = 1 THEN 120
110    GOTO 100
120    MOV P3
```

5. คำสั่ง GOTO…/GOSUB…RETURN

คำสั่ง GOTO เป็นคำสั่งให้โปรแกรมข้ามไปทำงานตามบรรทัด (Line Number) ในโปรแกรมที่กำหนด

คำสั่ง GOSUB เป็นคำสั่งที่สั่งให้โปรแกรมข้ามไปทำงานยังโปรแกรมย่อยที่กำหนด โดยในโปรแกรมย่อยแต่ละโปรแกรมจะต้องมีหัวของโปรแกรมเป็นตัวบ่งชี้ว่าเป็นโปรแกรมย่อยชื่ออะไร ทั้งนี้ในคำสั่ง MELFA Basic IV จะใช้สัญลักษณ์ * เป็นตัวแสดงว่าเป็นหัวของโปรแกรมย่อย

ตัวอย่าง

```
3900    GOSUB *PLACECYL
3910    GOSUB *GETPIST
3920    GOSUB *ASSPIST
  :
4100    *PLACECYL
4110    COVER% = 1
  :
4140    RETURN
```

6. คำสั่ง OVRD…

คำสั่ง OVRD เป็นคำสั่งที่ใช้ในการเพิ่มหรือลด Override Speed ของการเคลื่อนที่ของ Robot โดยการคำนวนการเพิ่มหรือลดความเร็วในการเคลื่อนที่จะคิดเป็นเปอร์เซ็นต์ (0% ถึง 100%) ของความเร็วที่ตั้งไว้ตั้งแต่ตอนแรกที่ตัว Controller ของ Robot

ตัวอย่าง 1) ถ้า Controller ถูกเซตค่า override speed ไว้ที่ 100%

MOV P1 Robot จะเคลื่อนที่ไปยังตำแหน่ง P1 ด้วยความเร็ว override 100%

OVRD 70 คำสั่งลดความเร็ว Override Speed

MOV P2 Robot จะเคลื่อนที่ไปยังตำแหน่ง P2 ด้วยความเร็ว override 100*70% = 70%

2) ถ้า Controller ถูกเซตค่า override speed ไว้ที่ 50%

MOV P1 Robot จะเคลื่อนที่ไปยังตำแหน่ง P1 ด้วยความเร็ว override 50%

OVRD 70 คำสั่งลดความเร็ว Override Speed

MOV P2 Robot จะเคลื่อนที่ไปยังตำแหน่ง P2 ด้วยความเร็ว override 50*70% = 35%

7. คำสั่ง HCLOSE1 และ HOPEN1

คำสั่ง HCLOSE1 และ HOPEN1 เป็นคำสั่งเฉพาะที่ใช้ในการสั่งการให้ Gripper ที่ติดตั้งอยู่ที่ตัว Robot ทำการเปิดหรือปิด เพื่อหยิบจับและปล่อยชิ้นงาน โดย

HCLOSE1 จะสั่งการให้ Robot Gripper Close

HOPEN1 จะสั่งการให้ Robot Gripper Open

8. **คำสั่ง** DLY…

คำสั่ง DLY เป็นคำสั่งเพื่อให้โปรแกรมทำการหน่วงเวลาทำงานเพื่อให้ Robot ทำงานทันกันกับคาบเวลาในโปรแกรม เช่นในการเคลื่อนที่ไปหยิบจับชิ้นงาน ณ ตำแหน่งๆ หนึ่งนั้น หลังจากที่ Robot เคลื่อนที่มาถึงตำแหน่งที่ต้องการแล้ว ต้องมีการ Delay หน่วงเวลาก่อนเพื่อให้แน่ใจว่า Robot เคลื่อนที่มาถึงตำแหน่งแล้วจริงๆ ก่อนที่จะทำการหยิบชิ้นงาน

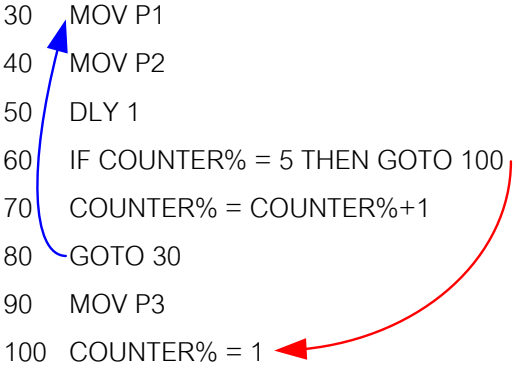
ตัวอย่าง	OVRD 10	คำสั่งลดความเร็ว Override Speed
	MVS P10	คำสั่งเคลื่อนที่ในแนวเส้นตรงไปยังตำแหน่ง P10
	DLY 1	คำสั่งหน่วงเวลาทำงาน 1 วินาที
	HCLOSE1	คำสั่งให้ Robot Gripper Close
	DLY 1	คำสั่งหน่วงเวลาทำงาน 1 วินาที
	MVS P9	คำสั่งเคลื่อนที่ในแนวเส้นตรงไปยังตำแหน่ง P9

9. **คำสั่ง** DEF INTE…

คำสั่ง DEF INTE เป็นคำสั่งที่ใช้ในการกำหนดตัวแปรที่จะนำไปใช้ในโปรแกรม โดยการกำหนดตัวแปรนั้นจะใช้อักษร (character) ไม่เกิน 8 ตัวอักษร และการนำไปใช้ในโปรแกรมก็สามารถใช้ชื่อที่กำหนดนั้นได้เลย แต่ต้องตามด้วยเครื่องหมาย %

ตัวอย่าง

```
10   DEF INTE COUNTER
20   COUNTER% = 1
30   MOV P1
40   MOV P2
50   DLY 1
60   IF COUNTER% = 5 THEN GOTO 100
70   COUNTER% = COUNTER%+1
80   GOTO 30
90   MOV P3
100  COUNTER% = 1
110  GOTO 30
```



10. **คำสั่ง** SPD…

คำสั่ง SPD จะเหมือนกับคำสั่ง OVRD คือเป็นการกำหนดค่าความเร็ว Override ของการเคลื่อนที่ของ Robot แต่คำสั่ง OVRD นั้นจะใช้งานง่ายกว่า คือไม่ต้องมีการกำหนดตัวแปรเพื่อนำมาใช้ในโปรแกรม เพียงแต่กำหนดตัวเลขแสดงความเร็วที่ต้องการเป็นเปอร์เซ็นต์เท่านั้น แต่คำสั่ง SPD นั้นจะต้องมีการกำหนดตัวแปรเป็น Character ขึ้นมาก่อน โดยคำสั่ง DEF INTE

ข้อแตกต่างระหว่างการกำหนดค่าความเร็ว Override ทั้งสองวิธีนี้ก็มีข้อดีข้อเสียแตกต่างกันไป คือ การใช้คำสั่ง OVRD นั้นสามารถกำหนดค่าความเร็วได้ทันทีโดยไม่ต้องมีการกำหนดค่าตัวแปรก่อน แต่ในขณะเดียวกัน การใช้คำสั่ง SPD ในการกำหนดความเร็วนั้น เมื่อมีการกำหนดตัวแปรขึ้นสำหรับใช้ในโปรแกรมแล้วก็สามารถใช้ตัวแปรนั้นๆ ได้ทุกตำแหน่งตลอดทั้งโปรแกรมโดยไม่ต้องกำหนดตัวแปรใหม่อีก ซึ่งทำให้มีความสะดวกในการเขียนโปรแกรม

ตัวอย่าง

การใช้คำสั่ง OVRD

```
10   MOV P1
20   OVRD 70 ←
30   MOV P2
40   OVRD 15 ←
50   MVS P3
60   DLY 1
70   HCLOSE1
80   DLY 1
90   MVS P2
100  OVRD 70 ←
110  MOV P1
:
```

การใช้คำสั่ง SPD

```
10   DEF INTE FST
20   FST% = 70
30   DEF INTE SLW
40   SLW% = 15
50   MOV P1
60   SPD FST% ←
70   MOV P2
80   SPD SLW% ←
90   MVS P3
100  DLY 1
110  HCLOSE1
120  DLY 1
130  MVS P2
140  SPD FST% ←
150 MOV P1
:
```

11. **คำสั่ง** DEF POS...

คำสั่ง DEF POS เป็นการกำหนดชื่อและค่าตำแหน่งของโปรแกรม โดยจะทำให้ Robot สามารถเคลื่อนที่ไปยังตำแหน่งต่างๆ ภายในโปรแกรมได้นอกเหนือไปจาก Position ที่ทำการ Teach ไว้แล้ว สำหรับการเคลื่อนที่ที่สำคัญจะแบ่งเป็น 4 แกนหลักๆ ด้วยกัน คือ แกน X, แกน Y, แกน Z, และแกนการหมุนของหัว Gripper (ทิศทางตามเข็มนาฬิกาเป็น + และทวนเข็มนาฬิกาเป็น -)

นอกจากนี้ยังมีคำสั่งในการกำหนดชื่อและตำแหน่งของโปรแกรมที่เป็นตำแหน่งช่วย (Auxiliary Position) เพื่อใช้เป็นตำแหน่งในการพักค่าการคำนวณการเคลื่อนที่ของ Robot ก่อนที่จะมีการเคลื่อนที่จริง สำหรับการกำหนดชื่อตำแหน่งช่วย สามารถทำได้โดยการใช้คำสั่ง “DEF POS AUXPOS”

ทั้งนี้การกำหนดตำแหน่ง Offset ในการให้ Robot เคลื่อนที่นั้น ก็สามารถทำได้โดยให้เคลื่อนที่เป็น Vector ผลรวมของการเคลื่อนที่ทั้งสองหรือสามแกนในเวลาเดียวกันได้ โดยเราสามารถกำหนดค่า Position ที่ต้องการเข้าไปในแต่ละแนวแกน จากนั้น Controller ภายในจะทำการคำนวณทิศทางการเคลื่อนที่ที่เหมาะสมให้เอง เช่นต้องการเคลื่อน Robot ให้ห่างจากตำแหน่ง P2 ไปตามแนวแกน +X 50 mm. แกน -Y 20 mm. สามารถทำได้ดังนี้

```
10      DEF POS AUXPOS
20      DEF POS VECXY
30      VECXY = (+50.00, -20.00,+0.00,+0.00,+0.00)
40      MOV P1
50      DLY 1
60      AUXPOS = P1 + VECXY
70      MOV AUXPOS
:
```

ตัวอย่าง ต้องการให้ Robot เคลื่อนที่ไปในตำแหน่ง Offset เหนือ Position P1 (+100.00, +150, +30.00, +0.00, +0.00) ไปตามแนวแกน Z เป็นระยะ 100 mm. แล้วทำการหมุนหัว Gripper ไปตามทิศทางตามเข็มนาฬิกาเป็นมุม 10 องศา หลังจากนั้นเคลื่อน Robot ไปยังตำแหน่ง P2 (+50.00, +120.00, +50.00, +90.00, +0.00) แต่ยังคงให้หัว Gripper ค้างอยู่ที่ตำแหน่งเดิมจากการหมุนตามเข็มนาฬิกาเป็นมุม 10 องศาที่ตำแหน่ง Offset P1

```
10    DEF POS AUX
20    DEF POS VECZ100
30    DEF POS VECDEG10
40    DEF POS VECXYZ
50    VECZ100 = (+0.00, +0.00, +100.00, +0.00, +0.00)
60    VECDEG10 = (+0.00, +0.00, +0.00, +10.00, +0.00)
70    VECXYZ = (-50.00, -30.00, +50.00, +0.00, +0.00)
80    MOV P1
90    DLY 1
100   AUXPOS = P1 + VECZ100
110   MOV AUXPOS
120   DLY 1
130   AUXPOS= AUXPOS + VECDEG10
140   MOV AUXPOS
150   DLY 1
160   AUXPOS = AUXPOS + VECXYZ
170   MOV AUXPOS
:
```

กำหนดชื่อตำแหน่งที่ใช้ในโปรแกรม

ระบุตำแหน่งที่ใช้ในการเคลื่อนที่ของ Robot

ผลต่างของตำแหน่งทางแกน X, Y และ Z ระหว่างตำแหน่ง P1 และ P2

12. คำสั่งกำหนด Input/Output

คำสั่งกำหนด Input/Output ที่เป็น bit เพื่อแสดงสถานะ on/off สามารถทำได้โดยใช้รูปแบบดังนี้

- กรณี INPUT ใช้ **"M_IN(Bit Number)"** เช่น

  Input bit number 3 มีสถานะเป็น 1 (ON) → M_IN(3) =1

- กรณี OUTPUT ใช้ **"M_OUT(Bit Number)"** เช่น

  Output bit number 7 มีสถานะเป็น 0 (OFF) → M_OUT(7) =0

นอกจากนี้ยังมีคำสั่งการ Define ตำแหน่ง bit ของ input และ output อีกวิธีหนึ่งเพื่อนำเงื่อนไงการแสดงผลสถานะ on/off โดยใช้รูปแบบการ Define คล้ายกับการ define ตำแหน่ง ดังนี้

"DEF IO (ชื่อ) = Bit, (Bit Number)"

ตัวอย่าง กำหนดเงื่อนไขให้สภาวะเริ่มต้นการทำงาน Lamp output Bit ที่ 8 มีสถานะเป็น ON (1) และเมื่อมีเงื่อนไขถ้า Input Bit ที่ 8 เป็น OFF (0) จะส่งผลให้ Lamp output Bit ที่ 8 ดังกล่าวเปลี่ยนสถานะเป็น OFF (0)

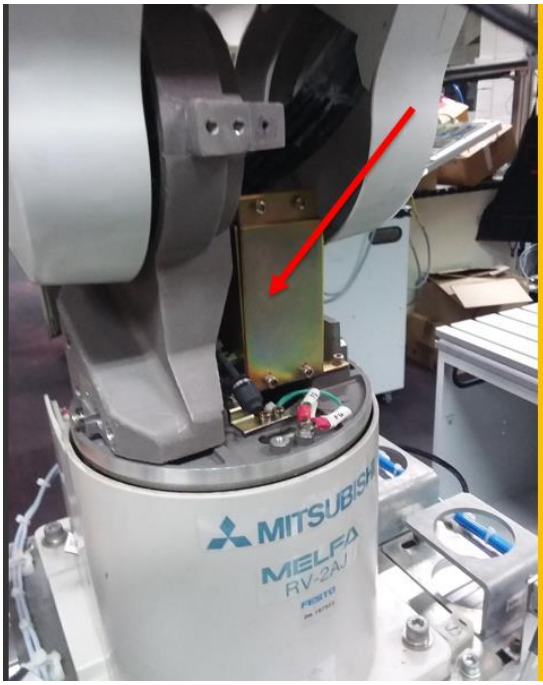
```
10      DEF IO MO1B1 = BIT,8
20      DEF IO SPRING = BIT,8
:
:
50      SPRING = 1          ชื่อแสดง output bit ที่ 8
60      IF MO1B1 = 0 THEN GOTO 60
70      DLY 1               ชื่อแสดง input bit ที่ 8
80      SPRING = 0
:
```

การถอดอุปกรณ์เพื่อเปลี่ยน แบตเตอร์รี่

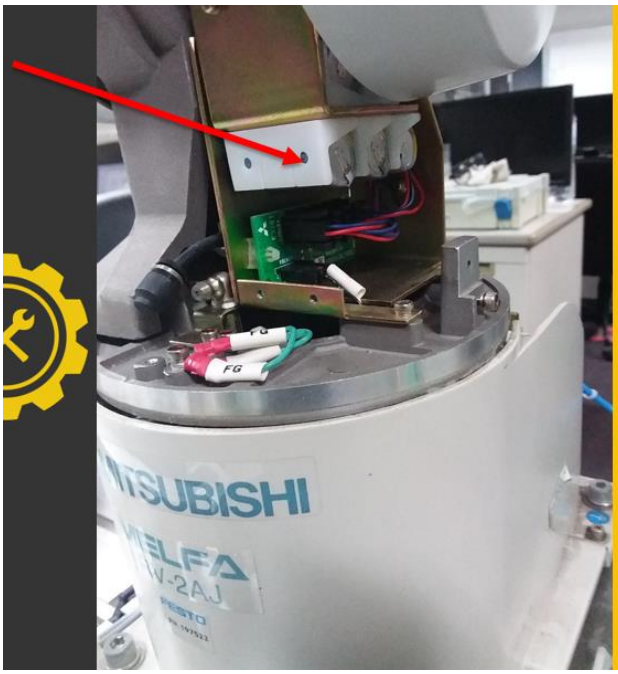
การบำรุงรักษา หรือการเปลี่ยนแบตเตอร์รี่สำหรับ หุ่นยนต์รุ่น MITSUBISHI MELFA RV-2AJ

******การเปลี่ยนแบตเตอร์รี่ ต้องทำไปอย่างรวดเร็วและถูกต้อง และจะต้องทำ ภายในระยะเวลา 15 นาที !!! ไม่เช่นนั้นแล้ว ตำแหน่งของ ROBOT จะหายไปและอาจจะทำให้ โปรแกรมใน ตัวหุ่นยนต์เสียหาย ได้ *******

ขั้นตอนมีดังต่อไปนี้



1.3 จะสังเกตเห็นในส่วนของ ฝาครอบแบตเตอร์รี่ ดังแสดงในรูป



1.4 จะสังเกตเห็นถึงตัวแบตเตอร์รี่ภายใน ตัวหุ่นยนต์ดังรูปภาพ

1.5ถอดสาย POWER SORUE ของตัว BATTERY ก่อน เพื่อความปลอดภัย

1.6 ถอดชุด ของแบตเตอร์รี่ออก เพื่อที่จะได้ทำการเปลี่ยนแบตเตอร์รี่ชุดใหม่เข้าไปทดแทน แบตเตอร์รี่ชุดเดิม