HƯỚNG DẪN
LẬP TRÌNH ROBOT ABB IRB120

MỤC LỤC

Bài 1: Tổng quan về robot IRB120

I. *Giới thiệu*

Là loại robot nhỏ nhất của ABB, IRB 120 có đầy đủ tất cả các tính năng và rất thuần thục trong các loại robot của ABB ứng dụng trong các ngành đóng gói nhỏ. Trọng lượng được rút gọn chỉ có 25kg và thiết kế nhỏ gọn giúp cho robot này có thể được lắp ráp tại bất cứ đâu, ở trong một trạm làm việc hay ở trên một máy móc hạy cạnh các robot khác trong một dây chuyền sản xuất.

IRB120 được ứng dụng trong phạm vi rộng các ngành công nghiệp bao gồm điện tử, thực phẩm và đồ uống, máy móc, năng lượng mặt trời, ngành dược, y tế và lĩnh vực nghiên cứu, đây là thế hệ thứ 4 của công nghệ robot mới của ABB. Loại robot 6 trục có khả năng tải lên tới 3kg (4 kg khi cổ tay cúi xuống) và tầm với 580mm, robot này có thể thực hiện nhiều hoạt động khác nhau linh hoạt hơn các giải pháp tự động hoá cố định. IRB120 là hệ thống được xây dựng hoàn hảo để thiết kế cho các ứng dụng tiết kiệm chi phí – đặc biệt không gian sử dụng là tối thiểu.

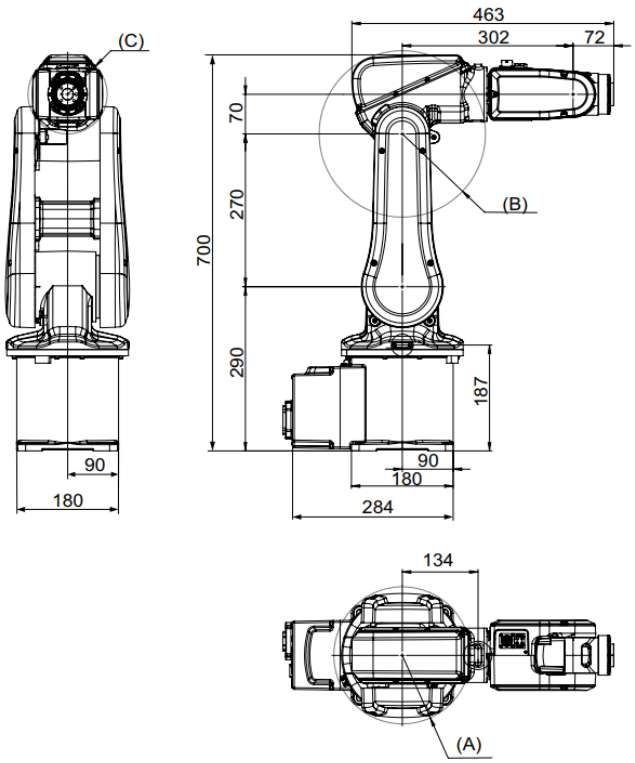
Bên cạnh tầm với theo phương ngang là 580mm, robot này là loại tốt nhất trong và có khả năng với 112mm phía dưới đế. Ngoài ra, IRB 120 còn có khả năng xoay tròn rất gọn nhờ cấu trúc đối xưng và không có độ dịch (offset) ở khớp thứ 2. Điều này đảm bảo robot có thể được lắp đặt cạnh các thiết bị khác và cổ tay mảnh giúp cánh tay robot có thể tiến tới gần các ứng dụng của nó hơn.

Được thiết kế với cấu trúc nhẹ và bằng nhôm, động cơ mạnh đảm bảo robot có khả năng gia tốc nhanh và có thể đạt được độ chính xác và linh hoạt trong bất cứ ứng dụng nào. Bộ điều khiển IRC5 tạo ra độ chính xác cao và điều khiển chuyển động cho các ứng dụng mà trước đó đã được lắp đặt rộng rãi. Bên cạnh lợi ích tiết kiệm không gian, bộ điều khiển mới này cũng dễ dàng đưa vào sử dụng thông qua nguồn điện đầu vào 1 pha, các đầu nối ngoài cho tất cả các tín hiệu và 16 đầu vào, 16 đầu ra, hệ thống I/O gắn liền có thể mở rộng. Lập trình offline RobotStudio giúp các nhà sản xuất mô phỏng trạm làm việc để tìm ra giải pháp tối ưu cho robot và cung cấp khả năng lập trình offline để giảm thời gian lãng phí và chậm trễ trong quá trình sản xuất

I.1. Robot IRB 120

- Thông số kỹ thuật:

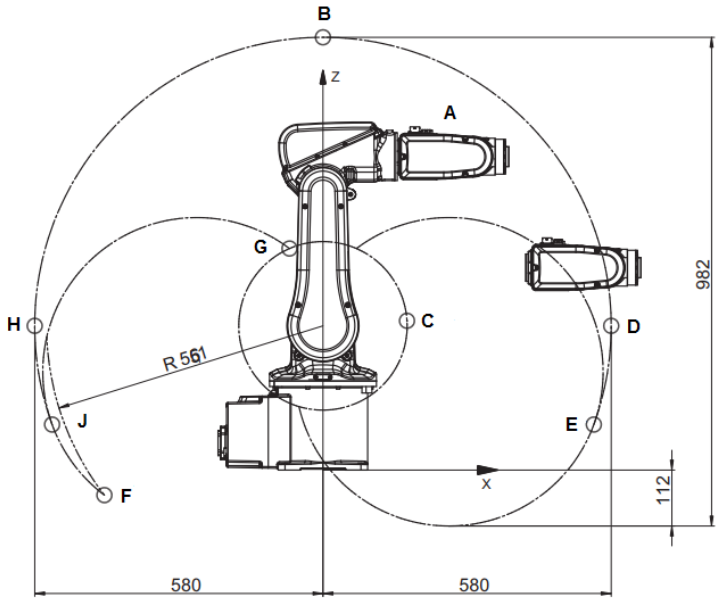
- Tải trọng tối đa: 3kg
- Tấm với: 580mm
- Số trục: loại robot 6 trục nhanh nhất của ABB
- Sai số vị trí: +/-0.01mm
- Trọng lượng: 25kg
- Nhiệt độ làm việc 50^0C tới 450^0C
- Độ ồn: <70dB
- Tiêu chuẩn chống nước: IP30
- Có thể gá đặt nhiều góc độ khác nhau
- Lập trình: lập trình trên màn hình cảm ứng được kế nối với bộ điều khiển



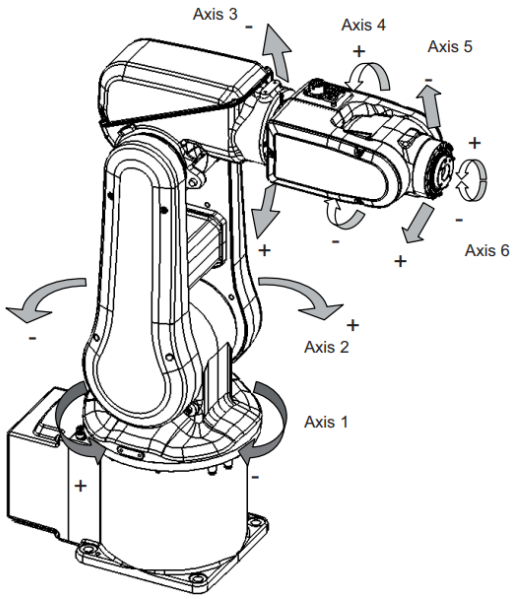
- R_A=121mm: bán kính quay tối thiểu của trục 1
- R_B=147mm: bán kính quay tối thiểu của trục 3
- RC=70mm: bán kính quay tối thiểu của trục 4

- Chuyển động:

- Tốc độ Tối đa: 6,2m/s
- Gia tốc tối đa: 28m/s^2

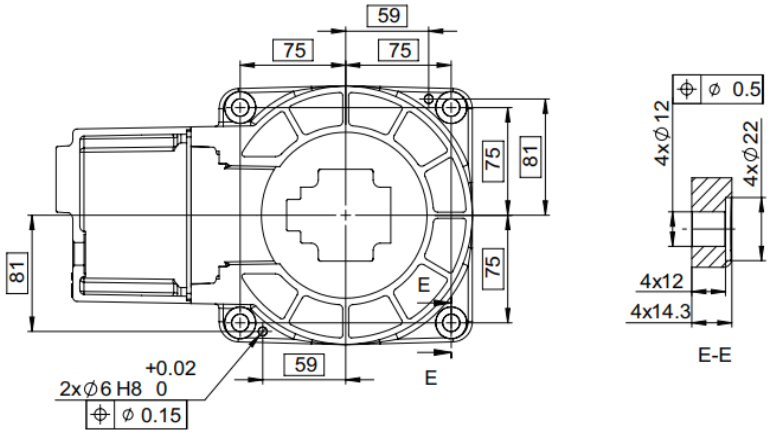


Điểm	Tọa độ điểm theo trục		Góc xoay	
	Trục X	Trục Z	Trục 2	Trục 3
A	302 mm	630 mm	0°	0°
B	0 mm	870 mm	0°	-77°
C	169 mm	300 mm	0°	+70°
D	580 mm	270 mm	+90°	-77°
E	545 mm	91 mm	+110°	-77°
F	-440 mm	-50 mm	-110°	-110°
G	-67 mm	445 mm	-110°	+70°
H	-580 mm	270 mm	-90°	-77°
J	-545 mm	91 mm	-110°	-77°

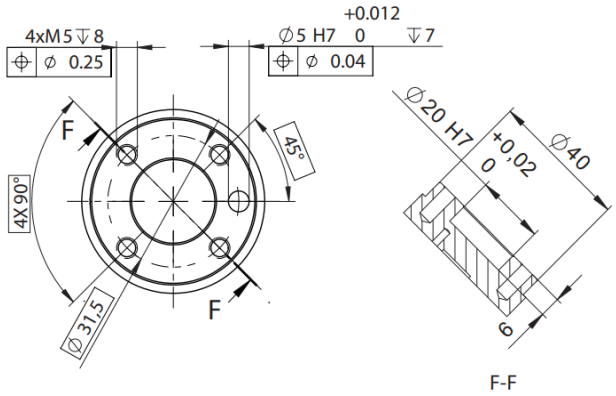


Trục di chuyển	Chuyển động trục	Giới hạn góc xoay	Tốc độ trục tối đa
Trục 1	Xoay tròn	+165° tới -165°	250 °/s
Trục 2	Cánh tay	+110° tới -110°	250 °/s
Trục 3	Cánh tay	+70° tới -90°	250 °/s
Trục 4	Cổ tay	+160° tới -160°	320 °/s
Trục 5	Uốn	+120° tới -120°	320 °/s
Trục 6	Đổi hướng	+400° tới -400°	420 °/s

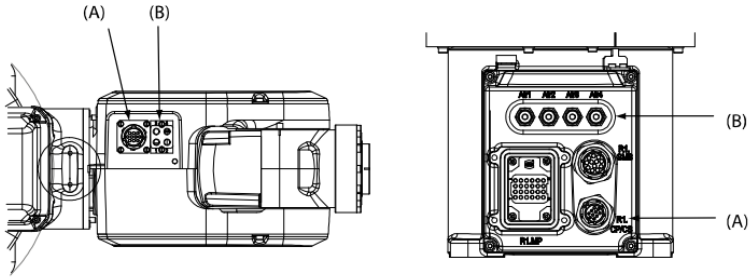
- Diện tích chân đế:



- Kích thước gá lắp tool trên trục 6



- Thiết bị tích hợp:



- Cổng I/O (A):
+ Số lượng cổng: 10
+ Điện áp tối đa: 49V, 500mA
+ Ngõ vào analog: 2
- Cổng cấp khí (B)
+ Số lượng cổng: 4
+ Kích thước ống khí: phi 4
+ Áp suất tối đa: 5 bar

I.2. Bộ điều khiển IRC5

- Nguồn cấp; 220-230VAC 50/60Hz
- Kích thước: 310x449x 42 mm
- Khối lượng: 30kg
- Nhiệt độ làm việc: 0-45^{0}C
- Tiêu chuẩn bảo vệ: IP20
- Cổng I/O:

+ Ngõ vào số: 16
+ Ngõ ra số: 16

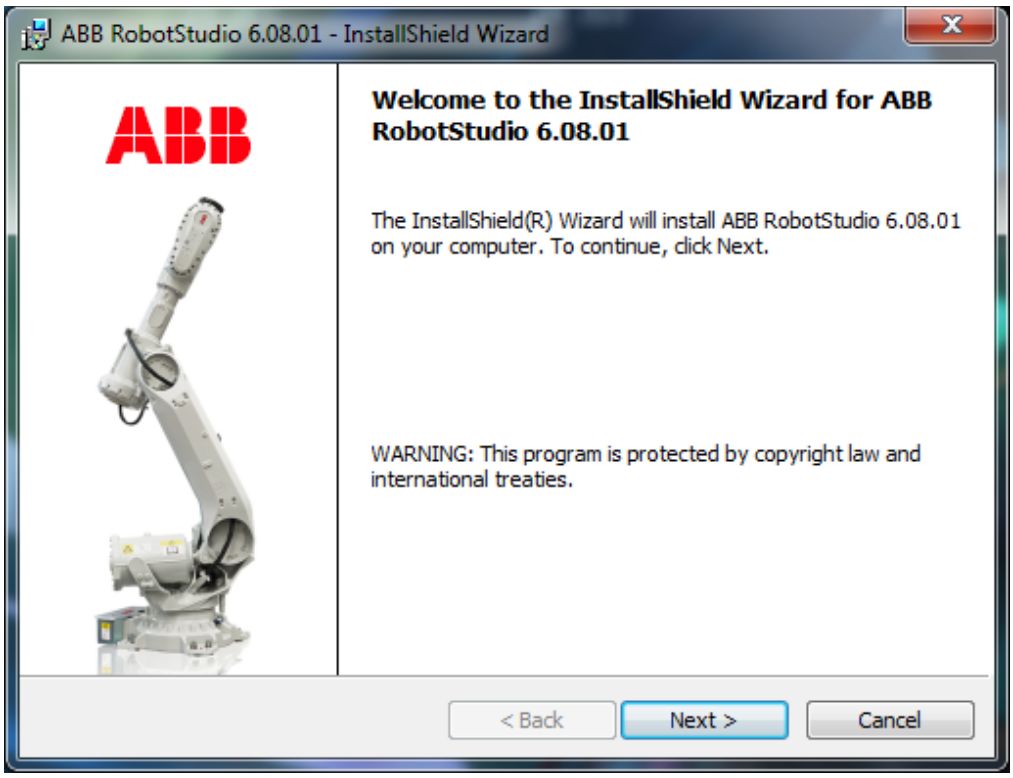
I.3. Tay dạy lập trình

- Phân loại IP: IP54
- Kích thước màn hình: 6,5 inch

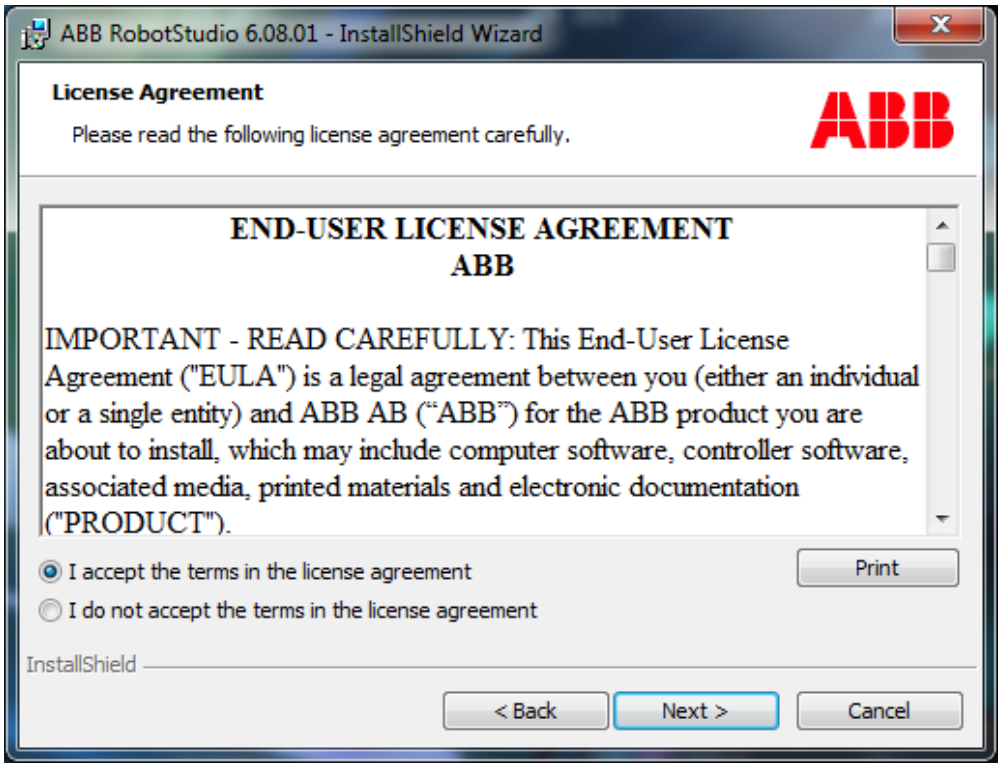
II. *Cài đặt phần mềm và mô phỏng robot ABB trên máy tính*

<u>Bước 1:</u> Vào trang https://new.abb.com/products/robotics/robotstudio/downloads để download phần mềm ***Robot Studio*** mới nhất.

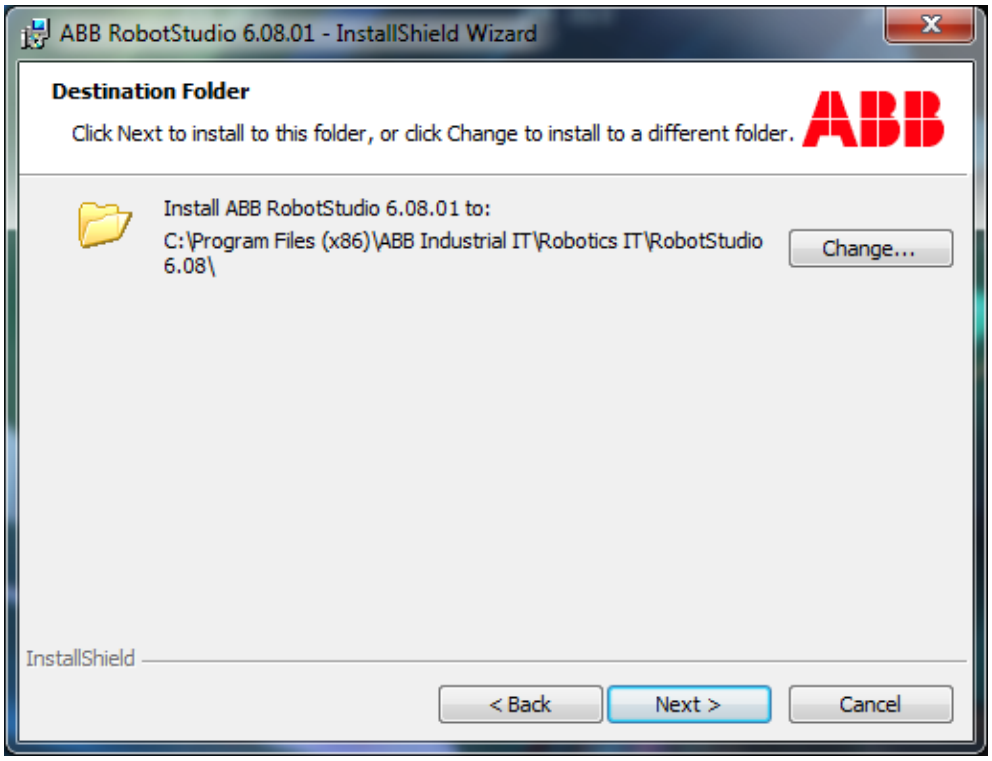
Bước 2: Chạy file ***ABB RobotStudio…exe*** để cài phần mềm. sau đó nhấn ***Next***



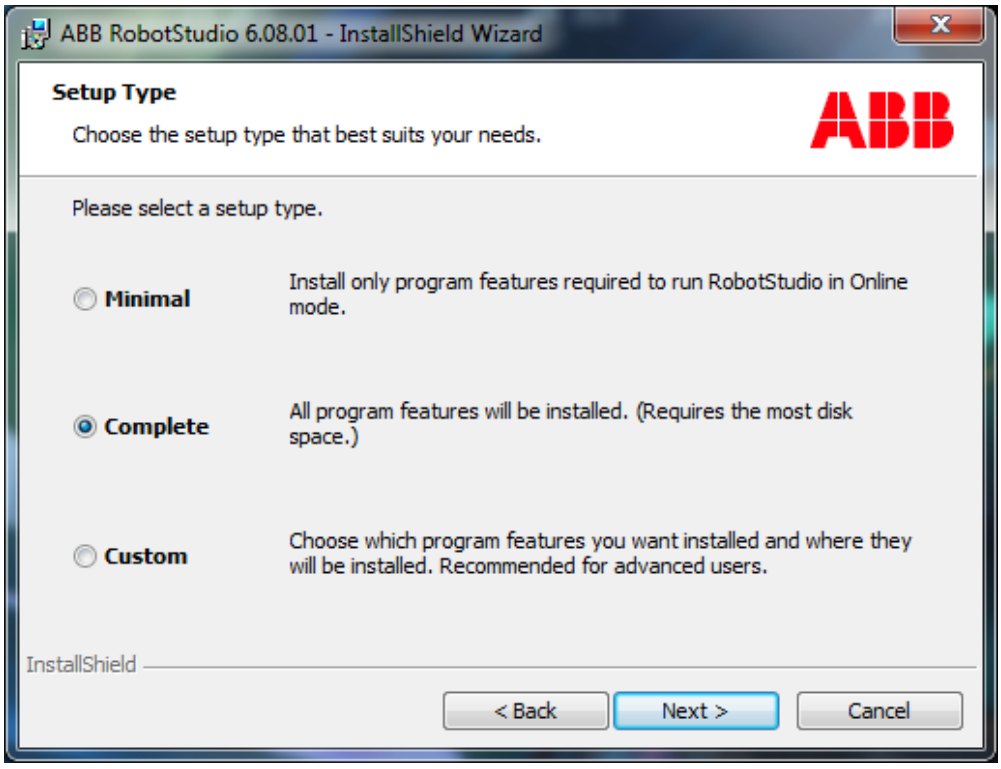
Bước 3: Nhấn ***I accept the terms in the license agreement***



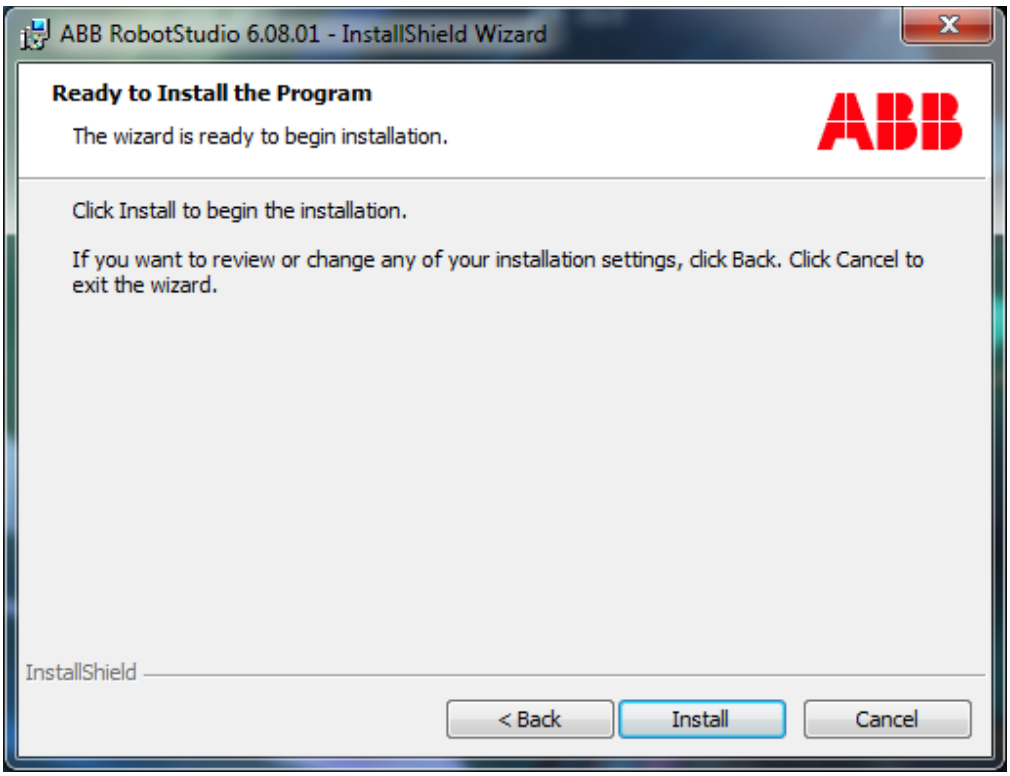
Bước 4: Chọn thư mục cài đặt rồi nhấn ***Next***



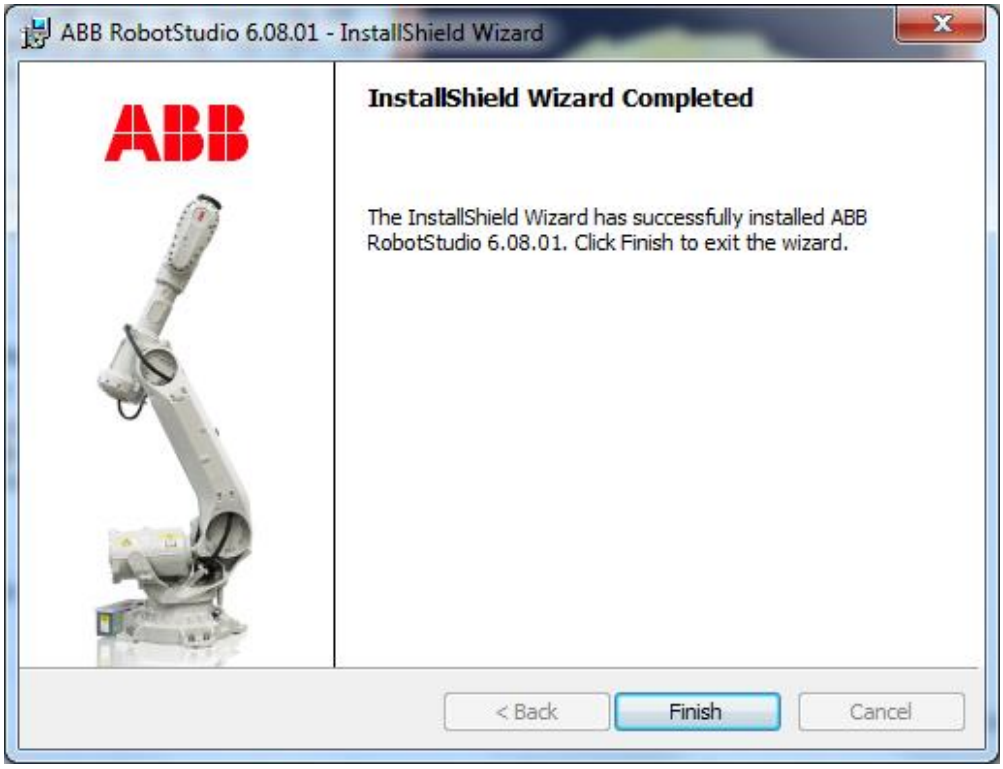
<u>Bước 5:</u> Chọn ***Complete*** và nhấn ***Next***



Bước 6: Nhấn ***Install***



Bước 7: Sau khi đợi cài đặt xong thì nhấn ***Finish***



Bài 2: Hướng dẫn khởi động robot IRB120

I. An toàn khi làm việc với robot

Tại sao phải để cập an toàn đối với robot IRB120 ?

⇨ Không giống như Universal Robot, robot ABB IRB120 không có tích hợp tính năng an toàn khi va chạm trên robot. Có nghĩa là khi bị va chạm mạnh, robot không dừng lại mà chỉ hiện thông báo và khóa phanh robot để robot không di chuyển được nữa. Để robot hoạt động lại cần di chuyển robot ra khỏi va chạm.

❖ Các tiêu chuẩn an toàn được đáp ứng từ bộ điều khiển IRC5

- Cấu tạo của robot được thiết kế đáp ứng theo tiêu chuẩn ISO 10218, vào tháng 1/1992, tiêu chuẩn về robot công nghiệp. Ngoài ra còn đáp ứng tiêu chuẩn ANSI/RIA 15.06-1999.
- Các tiêu chuẩn về chức năng an toàn:

+ **Dừng khẩn cấp** – IEC 204-1, 10.7
+ **Kích hoạt thiết bị** – ISO 11161, 3.4
+ **Bảo vệ** – ISO 10218 (EN 775), 6.4.3
+ **Giảm tốc** – ISO 10218 (EN 775), 3.2.17
+ **Khoá an toàn** – ISO 10218 (EN 775), 3.2.8
+ **Giữ để chạy** – ISO 10218 (EN 775), 3.2.7

❖ Để đối phó với những rủi ro trong các trường hợp trên ABB đưa ra các giải pháp an toàn sau:

- Dừng khẩn cấp

+ Thiết kế công tắc dừng khẩn cấp ở cả FlexPendant và Bảng điều khiển của robot

+ Có thể thiết kế thêm công tắc dừng khẩn cấp ở vị trí khác khi cần thiết.

- Chế độ làm việc

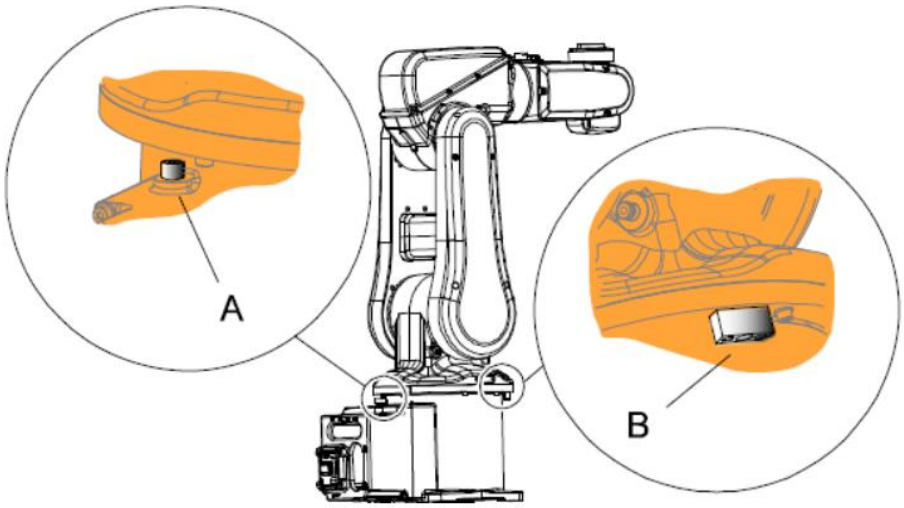
+ Chạy tự động với tốc độ không giới hạn
+ Chạy bằng tay với tốc độ < 250mm/s
+ Chạy bằng tay với tốc độ không giới hạn

- Kích hoạt thiết bị

+ Kích hoạt thiết bị là công tắc nhấn với 3 vị trí
+ Tất cả hoạt động của robot sẽ bị ngừng lại khi công tắc bị thả ra hoặc nhấn vào

- Giữ để chạy
+ Là một tính năng tuỳ chọn bằng cách thiết lập thông số
+ Sử dụng để chạy một đoạn hoặc từng câu lệnh chương trình ở chế độ tự chọn 100%
+ Công tắc kích hoạt và nút Giữ để chạy phải được nhấn đồng thời để kích hoạt động cơ

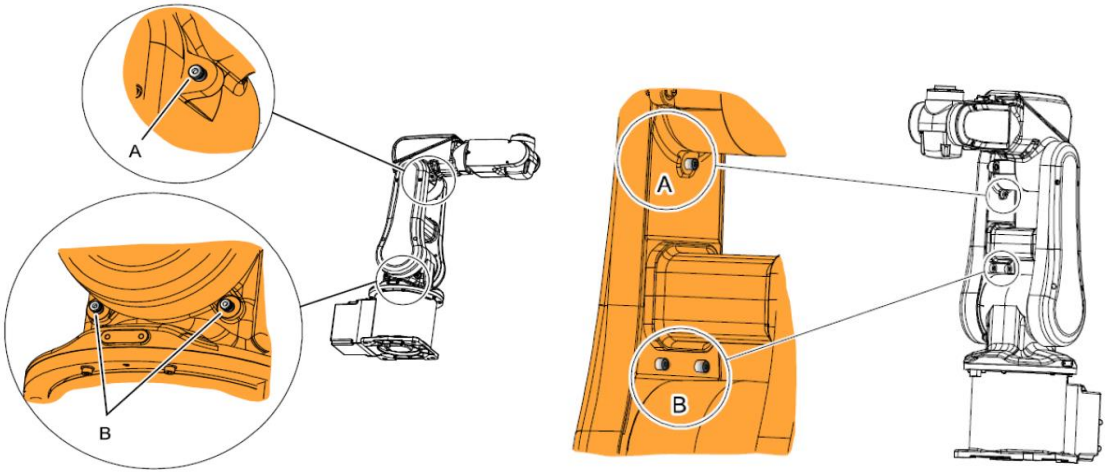
- An toàn bằng cách hạn chế khả năng hoạt động: Trên thân robot đã tích hợp sẵn các cử chặn cơ khí nhằm đảm bảo an toàn cho con người và robot
+ Cữ chặn trục quay 1



A : Cữ chặn cơ khí trục 1 (Đế)

B: Cữ chặn cơ khí trục 2 (tấm vòng cung)

+ Cữ chặn trục 2 và trục 3



A: Cữ chặn cơ kkhí trên trục 3 (Upper arm)

B : Cữ chặn cơ khí trên trục 2 (Swing housing)

❖ **Trường hợp khẩn cấp khi robot va chạm**

Khi robot bị va chạm, cần di chuyển ngay robot ra khỏi vùng va chạm bằng tay dạy. Chú ý lựa chọn di chuyển robot phù hợp (di chuyển thẳng/di chuyển theo trục, di chuyển theo tọa độ Base/Tool) để tránh va chạm mạnh hơn. Trong trường hợp không thể di chuyển được robot bằng tay dạy thì cần thực hiện thao tác sau (cần 2 người thao tác).

Người thứ 1: Tay giữ chặt robot để các khớp robot không di chuyển tự do khi nhả phanh tránh hư hỏng robot. Lưu ý sức nặng của robot.

Người thứ 2: Khi đã giữ robot thì nhấn vào nút nhả phanh trên tủ điều khiển robot. Cần có sự phối hợp ăn ý giữa 2 người.

Sau khi người thứ 1 nhấc robot ra khỏi vị trí va chạm thì ra hiệu để người thứ 2 thả nút nhả phanh ra.

II. Chuyển chế độ hoạt động của robot

Trên tủ điều khiển robot có công tắc chuyển chế độ hoạt động của robot bằng chìa khóa

Có 3 chế độ lựa chọn:

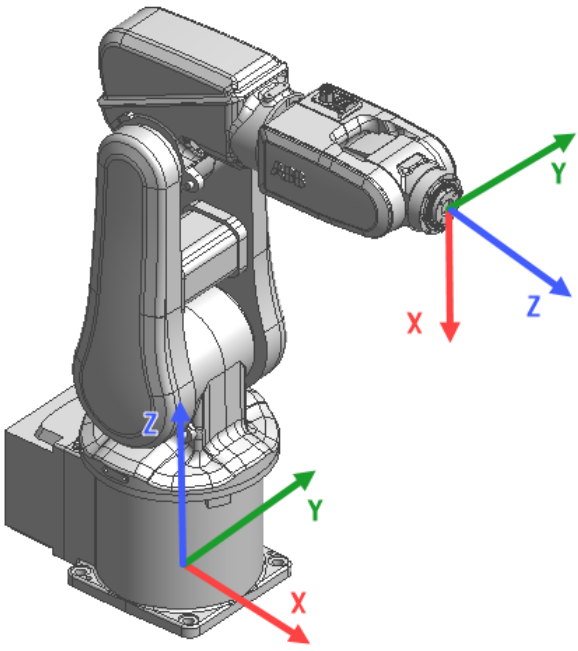
- Chuyển công tác sang bên trái: chế độ chạy tự động
- Chuyển công tắc sang vị trí giữa: chế độ bằng tay chạy với tốc độ an toàn mặc định của robot
- Chuyển công tắc sang bên phải: chế độ bằng tay chạy bằng tốc độ cài đặt

<u>Chú ý:</u>

- *Khuyến nghị nên chuyển công tắc sang vị trí giữa vì an toàn cho bạn và robot.*
- *Chỉ có thể tác động thay đổi lệnh chương trình robot khi chuyển sang chế độ bằng tay*

III. Hệ tọa độ làm việc của robot

Trong robot có 2 hệ tọa độ mặc định là hệ tọa độ Base và hệ tọa độ Tool. Tuy nhiên có thể tạo thêm mặt phẳng tọa độ hoặc TCP tool mới. Hệ tọa độ của robot được biểu diễn theo hình dưới

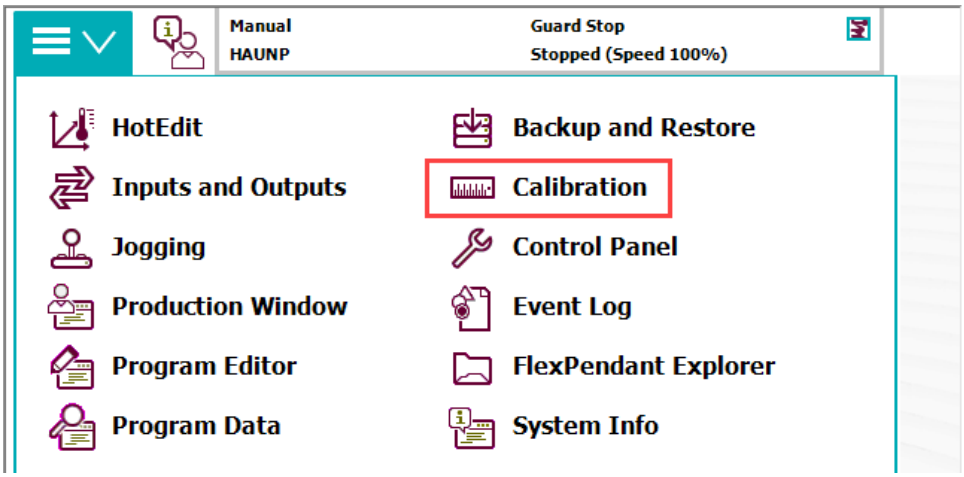


IV. Di chuyển robot

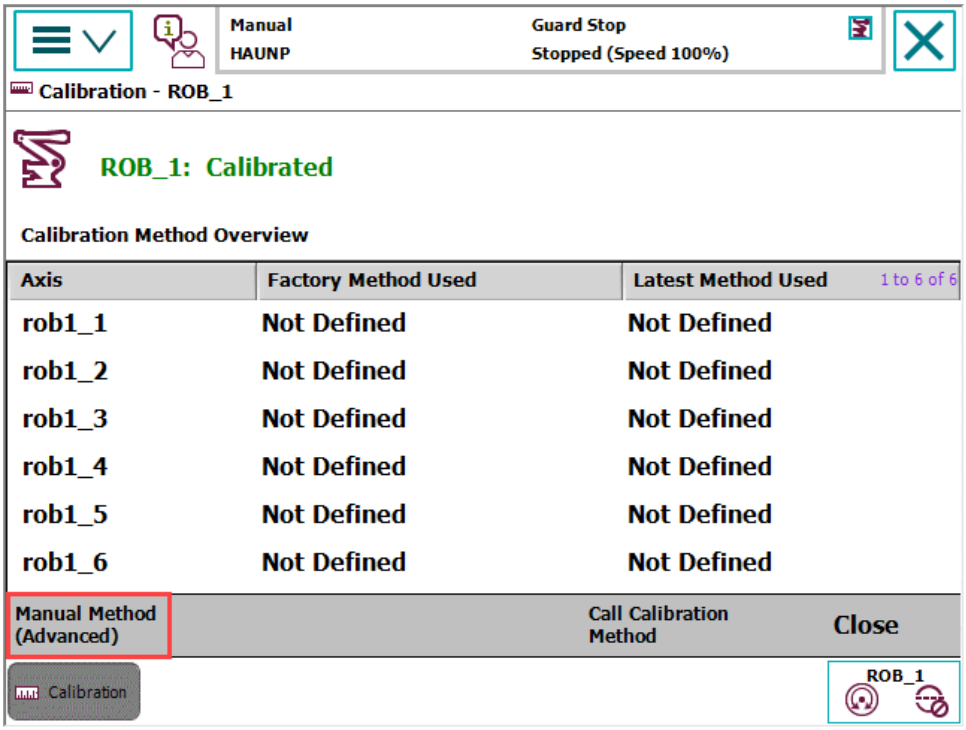
Khi khởi động robot lần đầu tiên, robot sẽ hiện thông báo robot chưa Calibration (chưa xác định tọa độ Base). Do vậy chỉ có thể di chuyển robot theo từng trục chứ không để di chuyển robot theo đường thẳng (trục X,trục Y,trục Z) và không thực hiện các thao tác tạo điểm tọa độ trong chương trình.

IV.1. Calibration robot

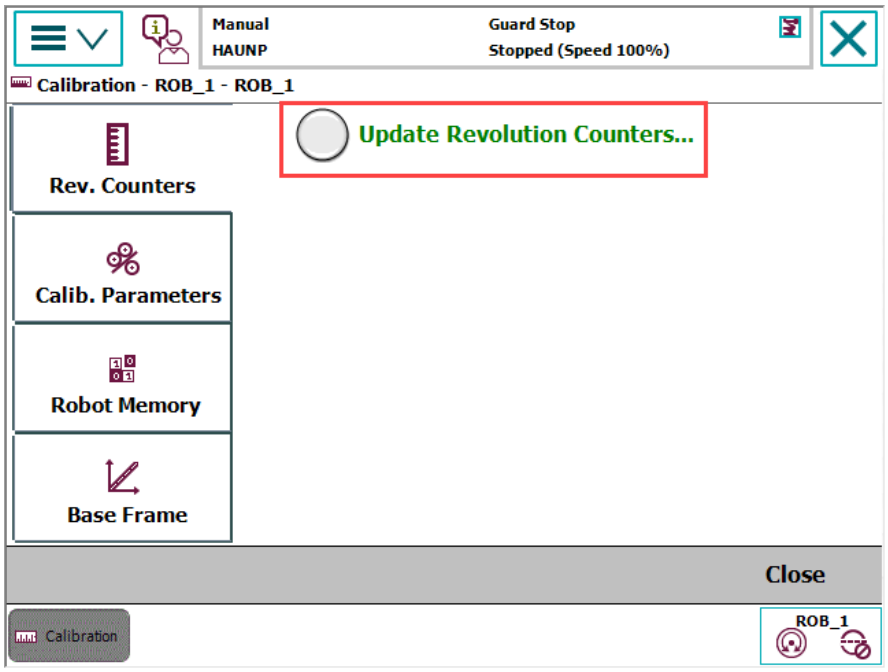
Bước 1: Chọn ***Calibration***



Bước 2: Chọn ***Manual Method (Advanced)***

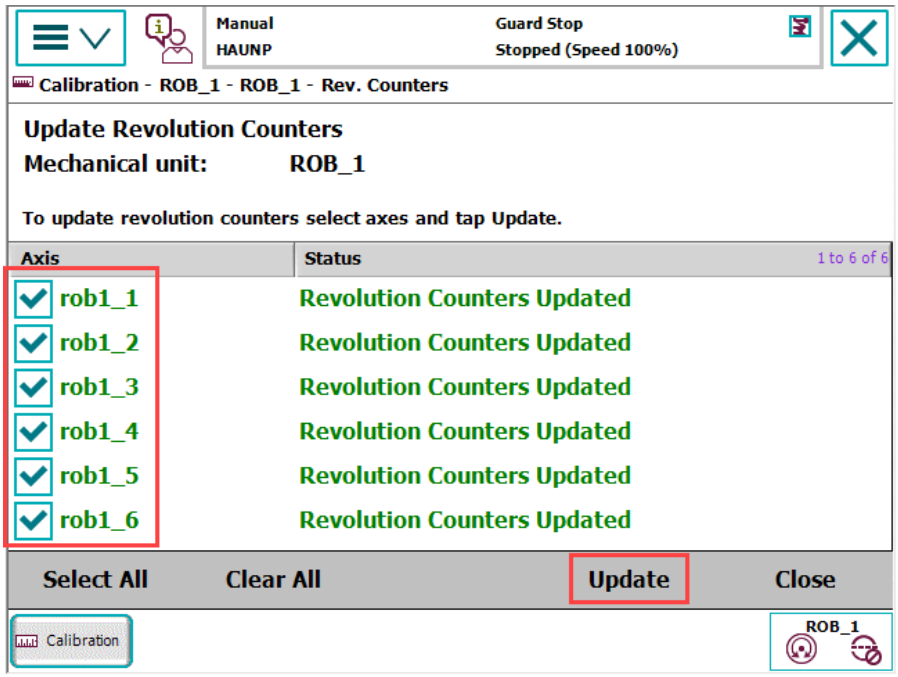


Bước 3: Chọn ***Update Revolution Counters..*** .Khi có thông báo hiện thì chọn ***Yes*** để xác nhận. sau đó chọn ***ROB_1*** và nhấn ***OK*** để calibrate.

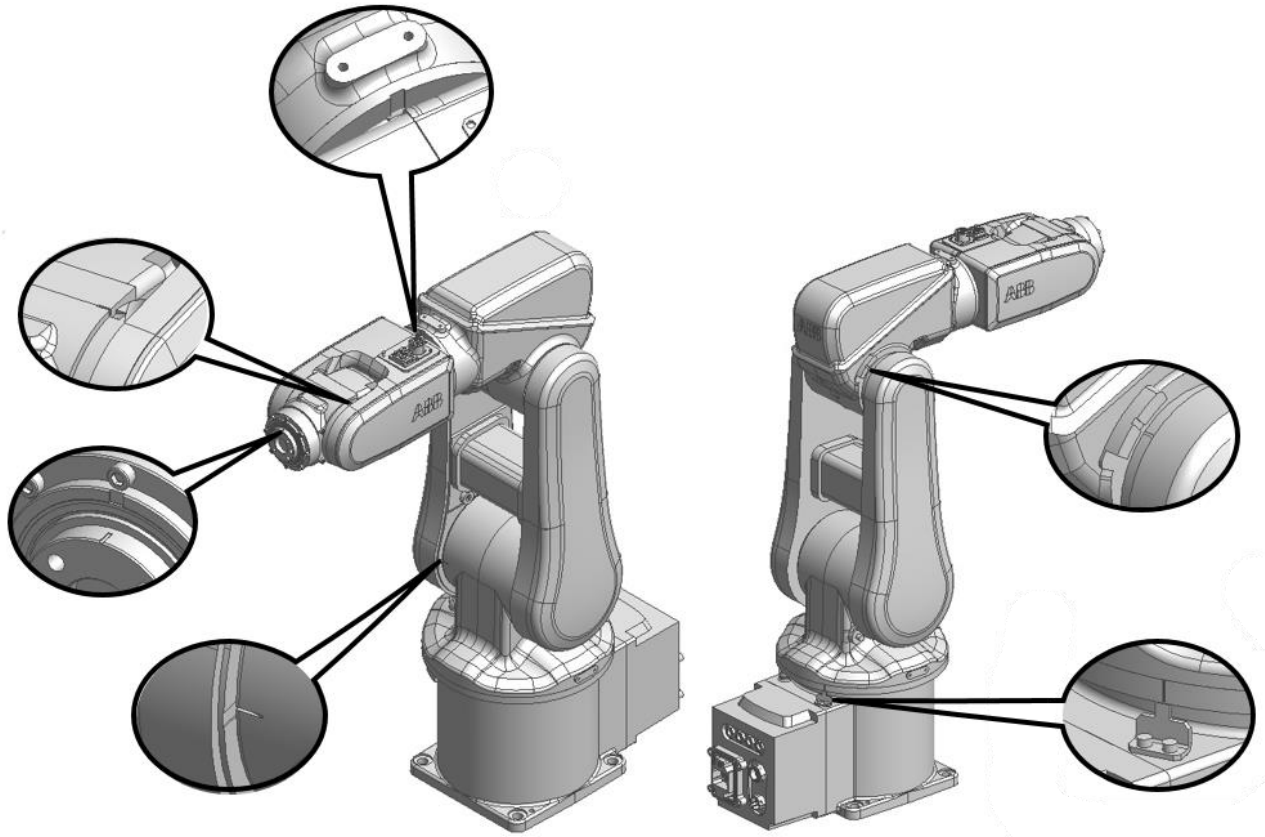


<u>Chú ý:</u> ***Không được Calibration robot bằng các tùy chọn nào khác nếu không có chuyên gia hướng dẫn. Nếu làm sai sẽ gây hư hỏng robot***

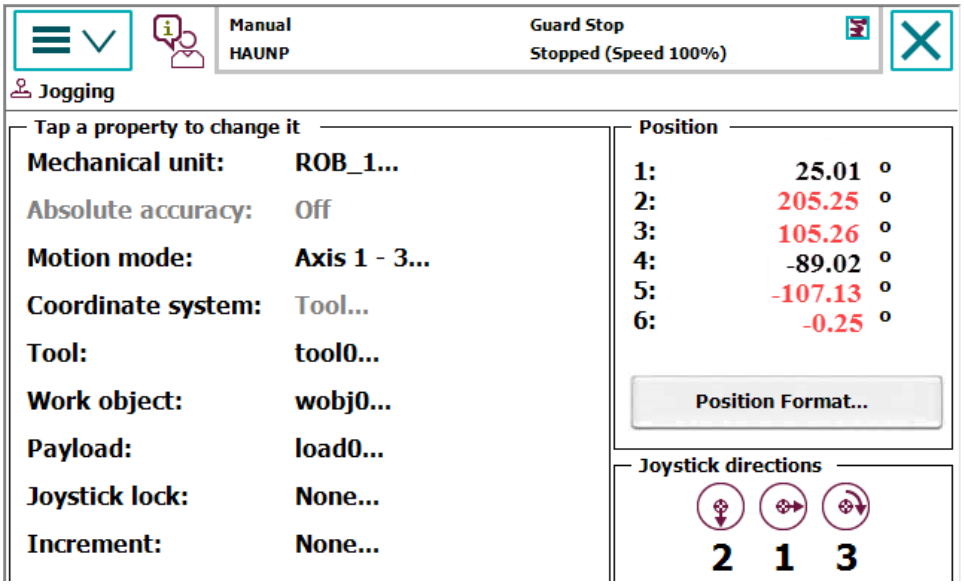
<u>Bước 4:</u> Chọn các trục của robot cần calibrate rồi nhấn ***Update.*** Có thể ***Update*** từng trục một nếu vùng hoạt động của robot quá nhỏ.



Các chuẩn calibrate của các trục được thể hiện như hình dưới. Di chuyển robot để tâm các vạch ở các trục liền kề trùng nhau. Các chuẩn này chỉ mang tính tương đối vì mỗi người có một góc nhìn khác nhau. Và sau khi calibrate xong thì đây chính là gốc của robot.



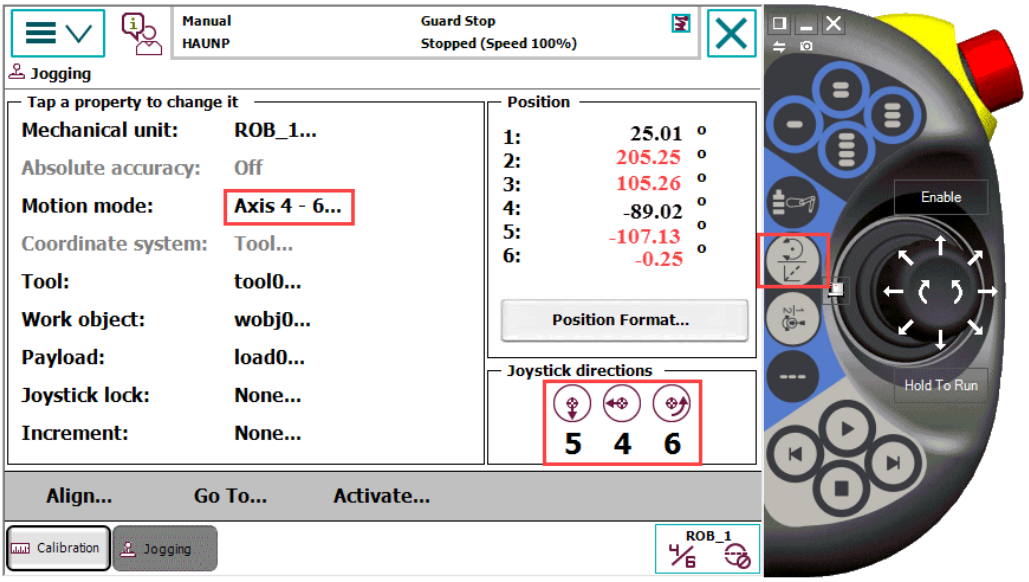
<u>Lưu ý:</u> *Khi Calibration trục robot, các vị trí trục chỉ mang tính tương đối. Do vậy khi robot bị mất gốc tọa độ (Calibration) thì cần phải lấy lại đúng vị trí robot đã* ***Calibration*** *trước đó để vị trí các điểm đã tạo trong chương trình không bị xê dịch.*



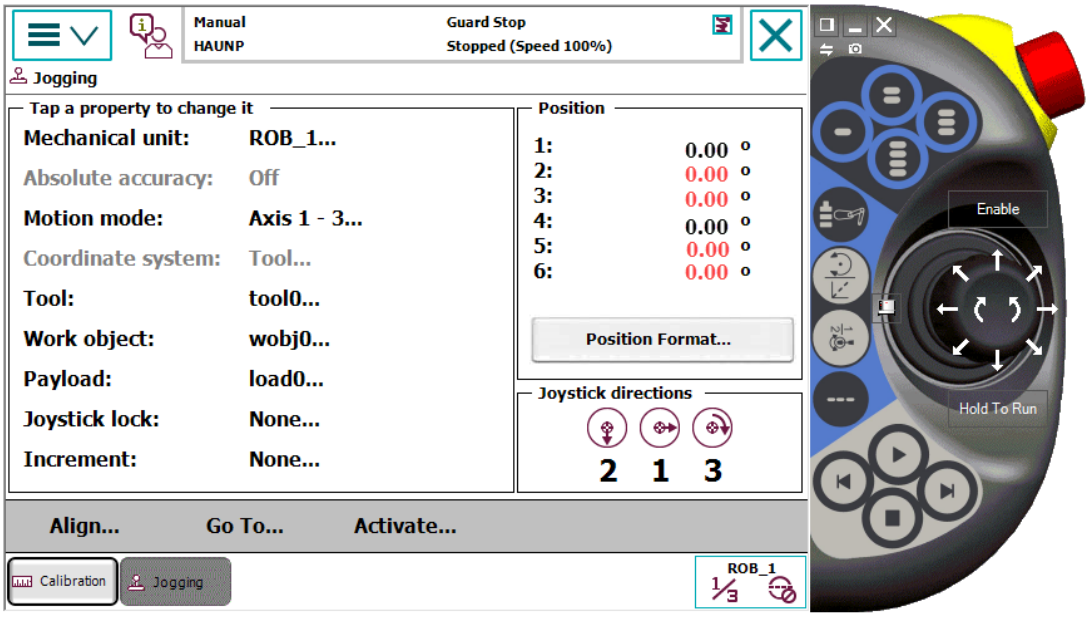
Như hình trên thì các trục 2, 3, 5, 6 bị mất gốc và sẽ hiển thị màu đỏ. Để calibration lại robot đúng như lần calibration trước đây thì thực hiện các thao tác sau:

- **Yêu cầu không được nhấn *Update Revolution Counters* nếu chưa thực hiện các thao tác dưới đây. Nếu không sẽ không lấy lại được gốc tọa độ của robot trước đây.**

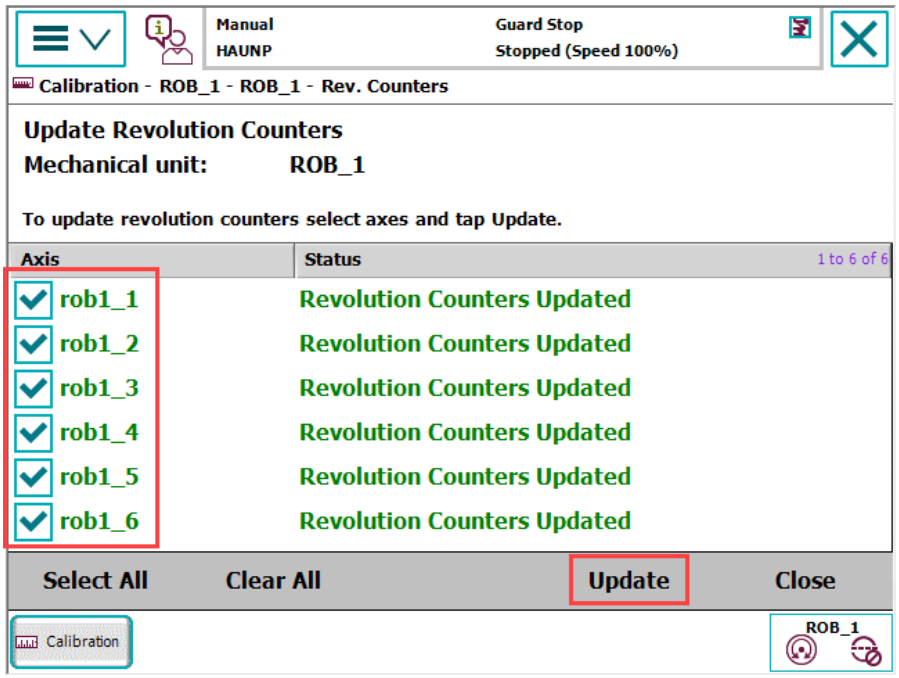
<u>Bước 1:</u> Chọn Jogging và chuyển sang chế độ di chuyển robot theo từng trục (khi mất gốc, robot không thể di chuyển theo đường thẳng vì không biết hệ tọa độ ở đâu)



<u>Bước 2:</u> Di chuyển từng trục về trạng thái **0.00^0**. Bắt buộc phải di chuyển tất cả các trục về 0 mới được ***Update Revolution Counters.***

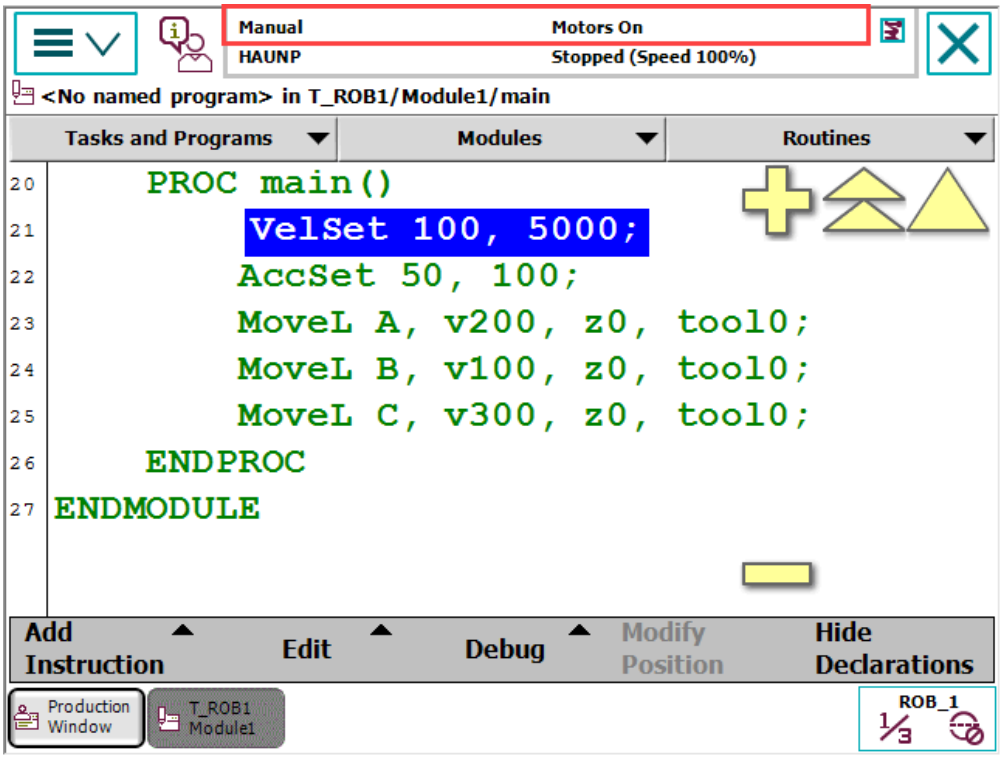


<u>Bước 3:</u> Vào ***Calibration*** để Update lại tọa độ của 6 trục robot trùng với tọa độ trước đây đã calibration



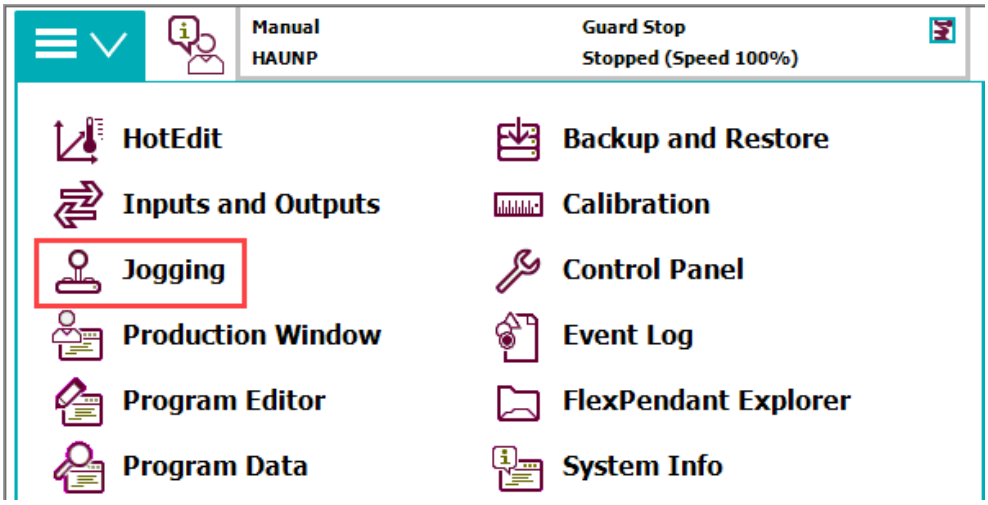
IV.2. Di chuyển robot bằng tay dạy lập trình

Để chạy robot bằng tay, cần nhấn giữ nút ***công tắc kích hoạt*** trên tay dạy lập trình (xem bài 2) khi nào trên màn hình thay đổi **Motor On** thì lúc này mới di chuyển được robot. Trong suốt quá trình di chuyển phải ấn giữ ***công tắc kích hoạt*** nếu thả ra robot sẽ dừng lại.

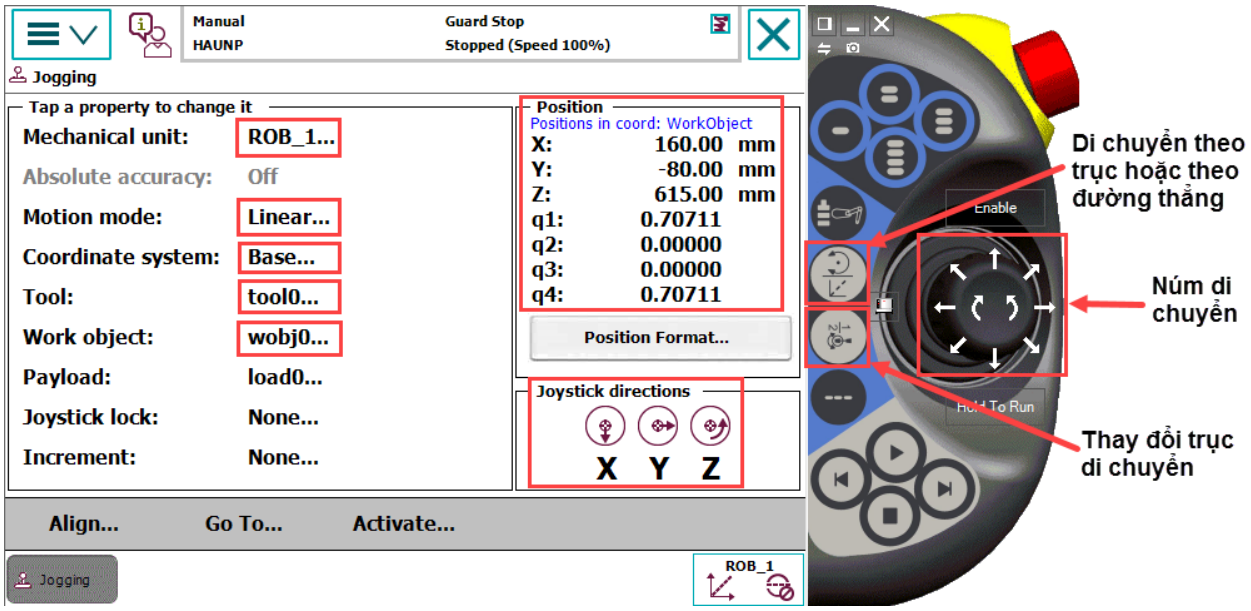


Bước 1: Chuyển chế độ bằng tay trên tủ điều khiển robot

Bước 2: Vào ***Jogging***



Bước 3: Lựa chọn các di chuyển theo hình dưới



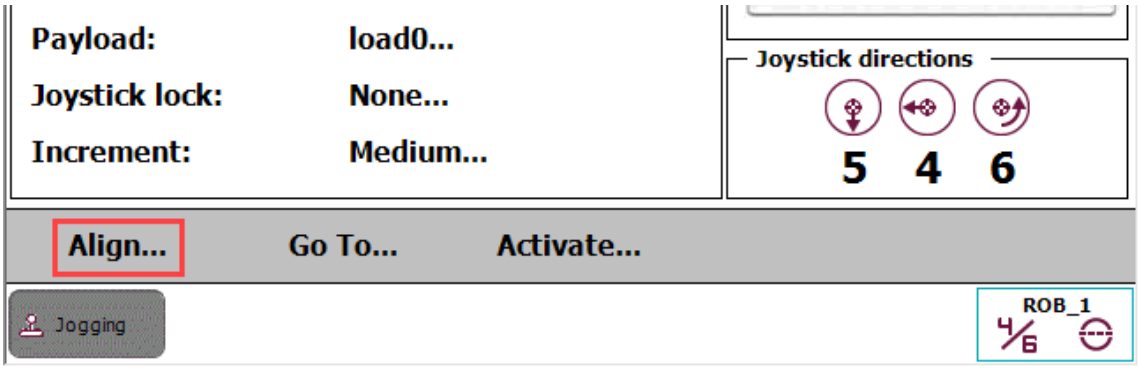
- ***Mechanical unit:*** lựa chọn robot cần di chuyển (nếu có 2 robot trở lên)
- ***Motion mode:*** di chuyển theo từng trục robot hay theo đường thẳng (trục X, trục Y ,trục Z)
- ***Coordinate system:*** chọn kiểu hệ tọa độ di chuyển (Base, Tool, Work object..)
- ***Tool:*** lựa chọn tool di chuyển (tool0-mặc định, tool tự tạo…)
- ***Work object:*** hệ tọa độ làm việc (wobj0-Base, wobj tự tạo..)
- ***Position:*** hiển thị tọa độ, góc xoay của robot
- ***Joystick directions:*** hiển thị lựa chọn di chuyển và hướng vặn của núm di chuyển

⇨ Nhấn vào biểu tượng phím tắt  để thay đổi lựa chọn di chuyển (linear, axis, reorient…)

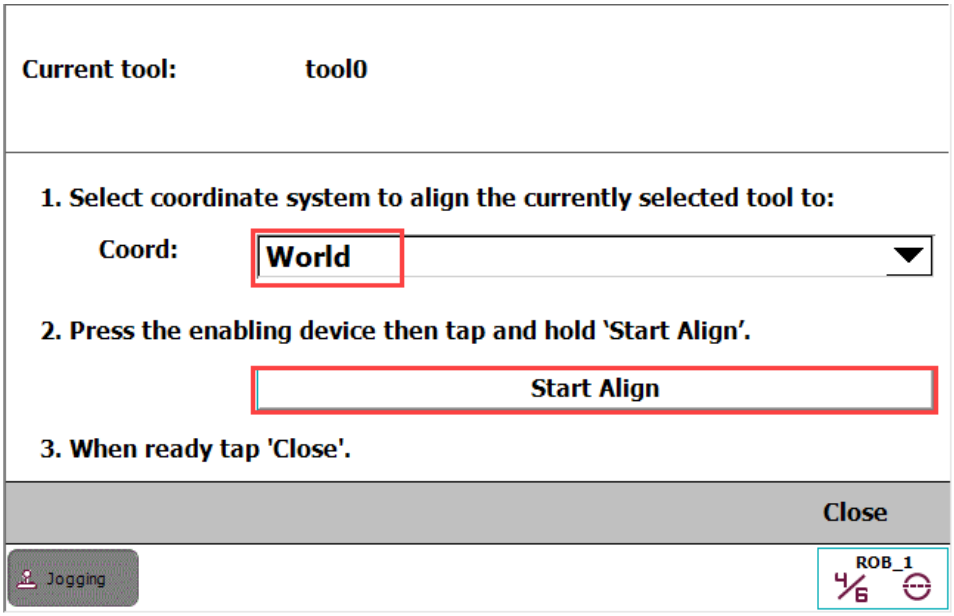
⇨ Nhấn vào biểu tượng phíp tắt  để thay đổi trục robot khi chọn di chuyển theo trục.

❖ **Di chuyển trục Z của TCP tool robot vuông góc với một mặt phẳng**

<u>Bước 1:</u> Nhấn vào ***Align***



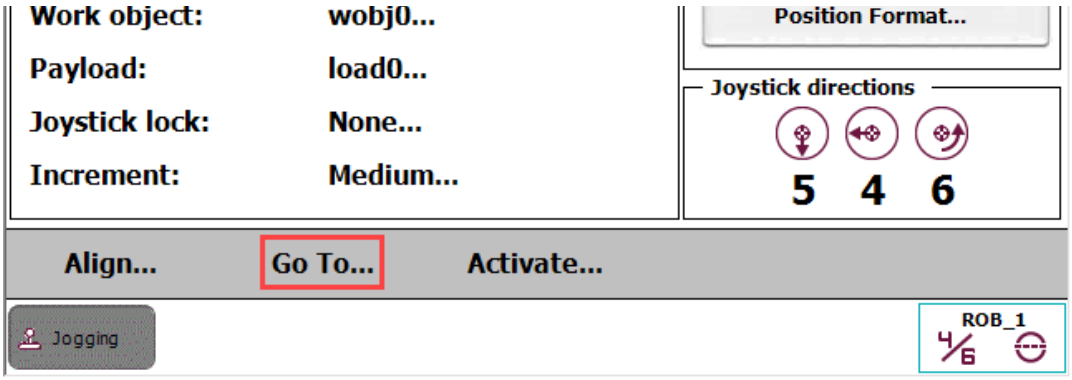
<u>Bước 2:</u> Lựa chọn di chuyển ở ô ***Coord.*** Sau đó nhấn giữ ***Start Align*** để robot tự động di chuyển cho đến khi robot dừng lại rồi nhấn ***Close***



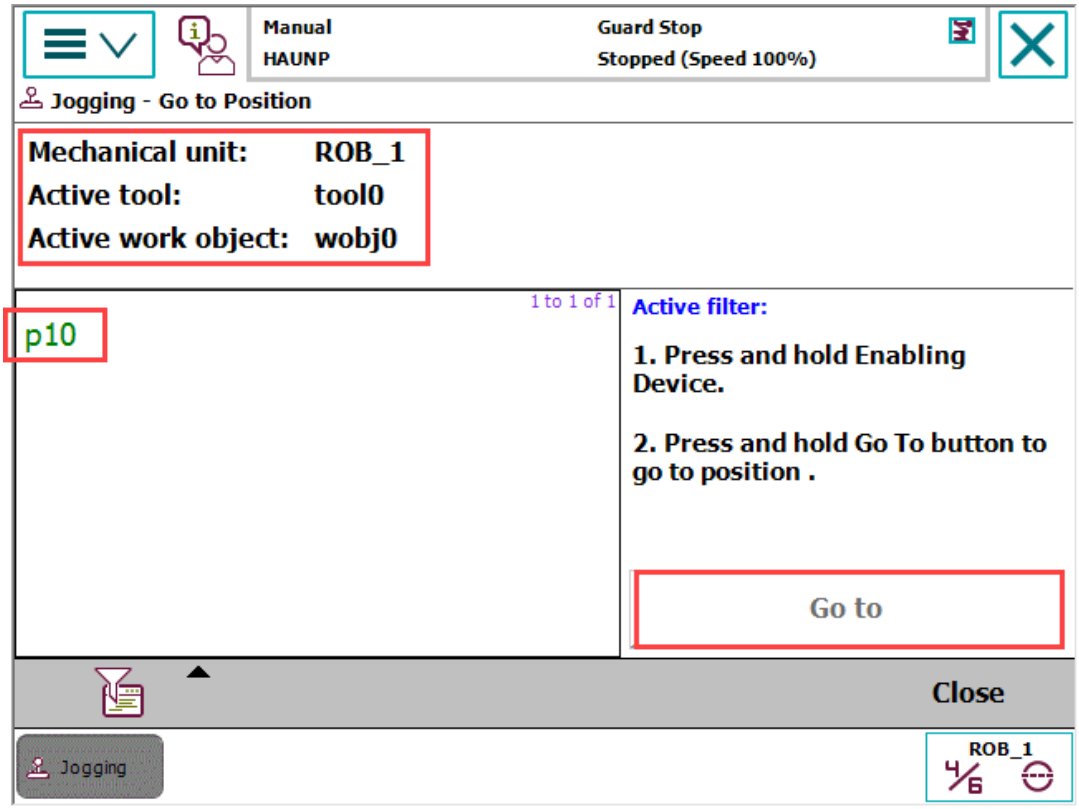
- ***World:*** Di chuyển trục Z của tool vuông góc với mặt phẳng gần nhất
- ***Base:*** Di chuyển trục Z của tool vuông góc với mặt phẳng base
- ***Work object:*** Di chuyển trục Z của tool vuông góc với mặt phẳng tự tạo (lựa chọn work object ở tab ***Jogging***)

❖ **Di chuyển robot đến một điểm đã tạo**

Bước 1: Nhấn vào ***Go to***



Bước 2: Chọn điểm cần di chuyển đến sau đó nhấn và giữ nút ***Go to*** cho đến khi robot di chuyển đến vị trí đó và dừng lại rồi nhấn ***Close***

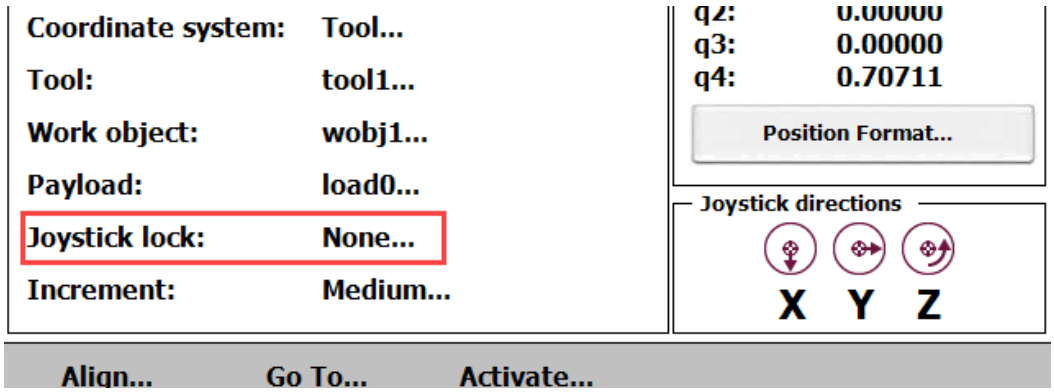


- ***Mechanical unit:*** robot di chuyển
- ***Active tool:*** chọn tool di chuyển
- ***Active work object:*** hệ tọa độ di chuyển

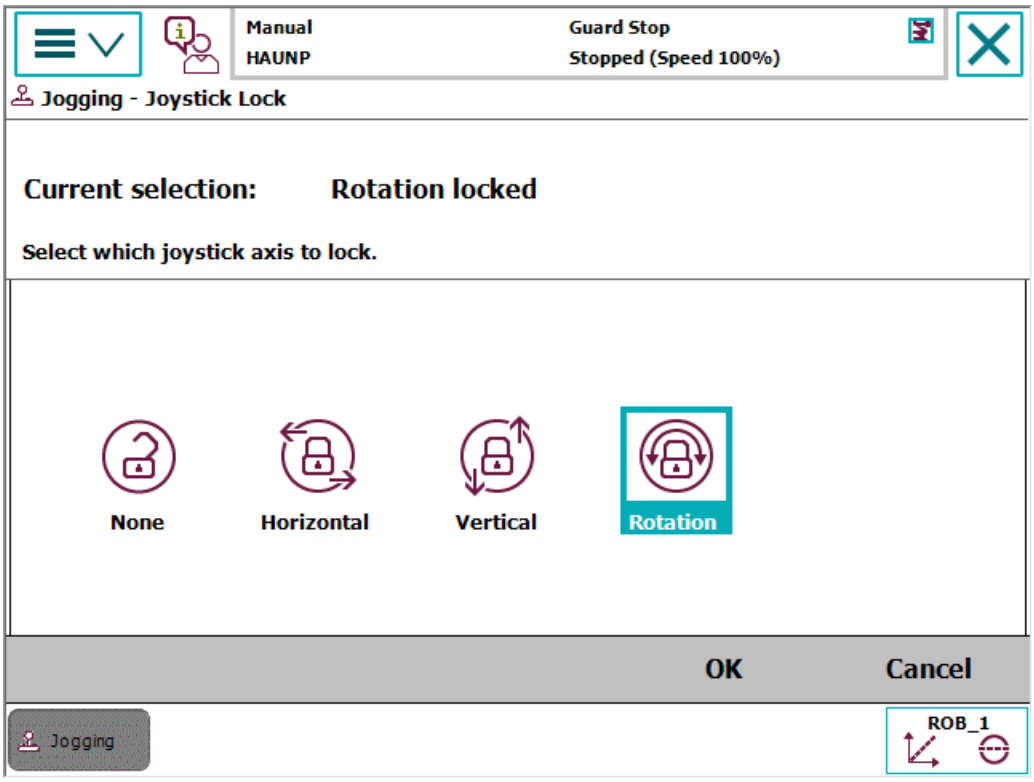
⇨ Đây chỉ là hiển thị di chuyển, nếu muốn thay đổi tool, work object thì thay đổi ở tab ***Jogging***

❖ **Khóa trục robot**

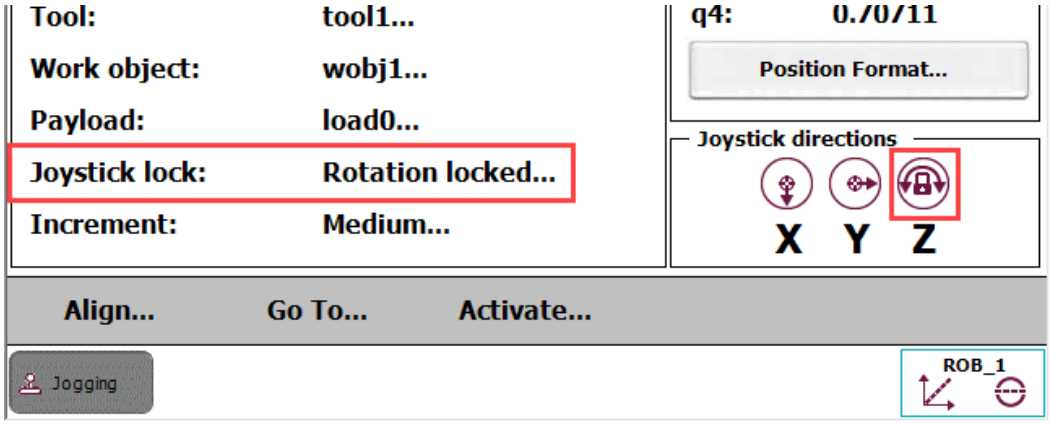
Bước 1: Chọn ***Joystick lock***



Bước 2: Lựa chọn hướng di chuyển cần khóa. Có thể chọn khóa nhiều hướng di chuyển



Bước 3: Các lựa chọn khóa di chuyển sẽ bao gồm di chuyển theo hệ tọa độ X,Y,Z và các trục của robot

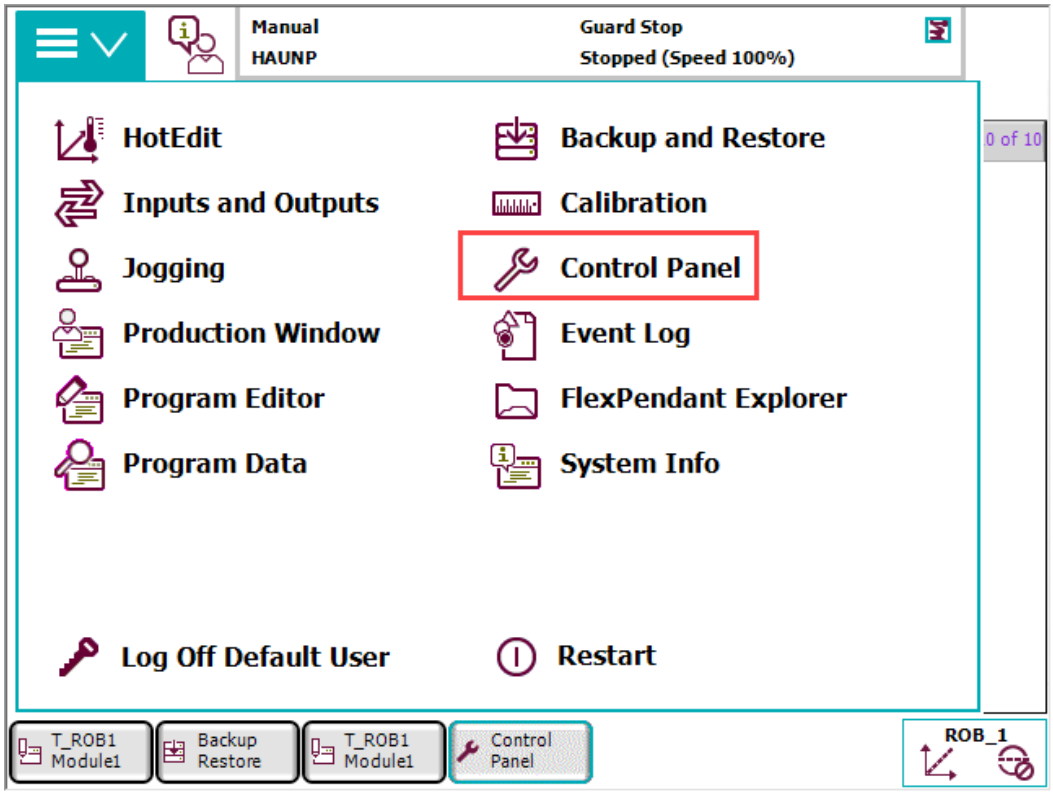


V. Cài đặt các thông số robot

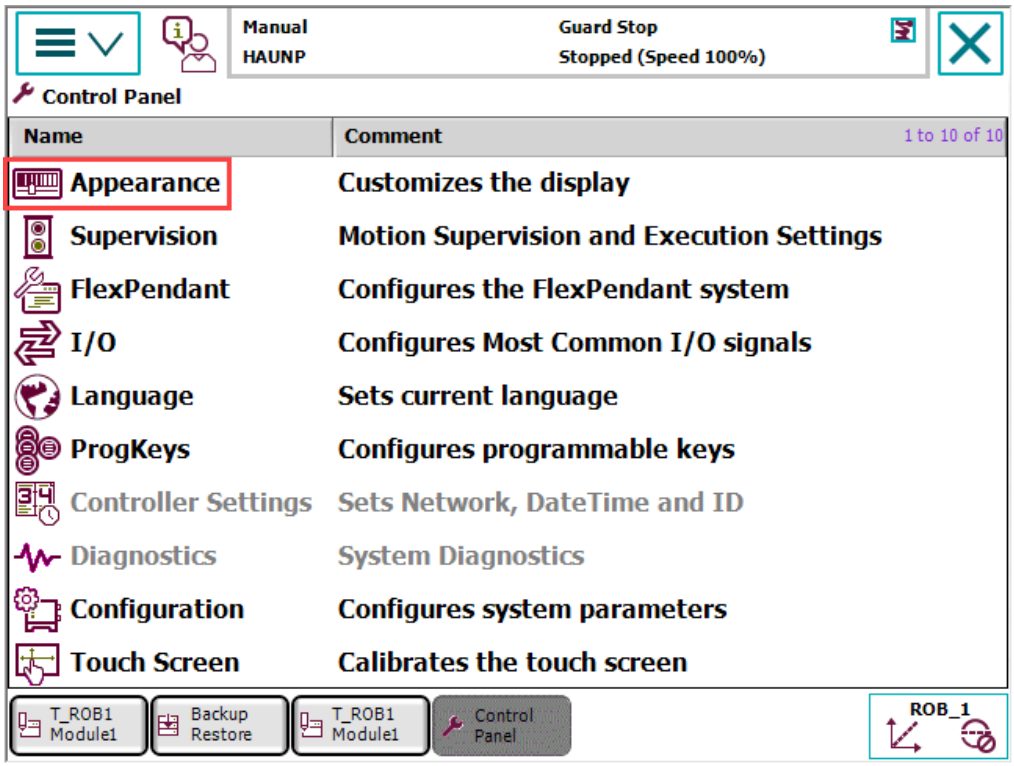
Lưu ý: Các thông số bên trong tab ***Control Panel*** chỉ được thay đổi/thiết lập mới khi hiểu biết về robot. Không được tự ý thay đổi/thiết lập khi không có người hướng dẫn.

V.1. Thay đổi độ sáng màn hình

Bước 1: Vào ***Control Panel***



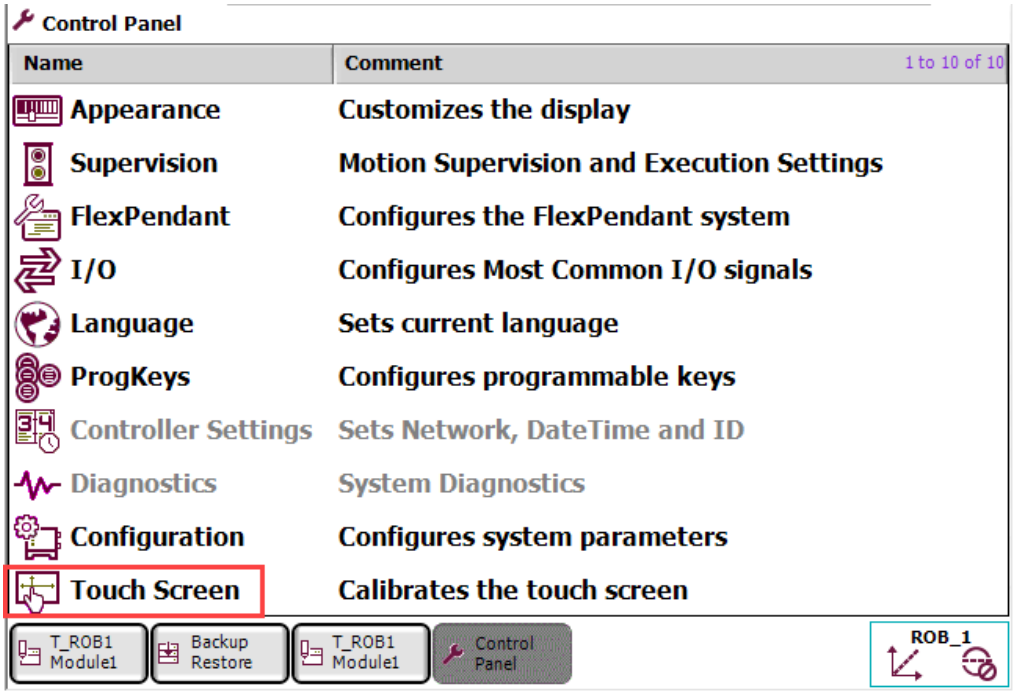
Bước 2: Chọn ***Appearance*** để chỉnh độ sáng phù hợp và nhấn ***OK***



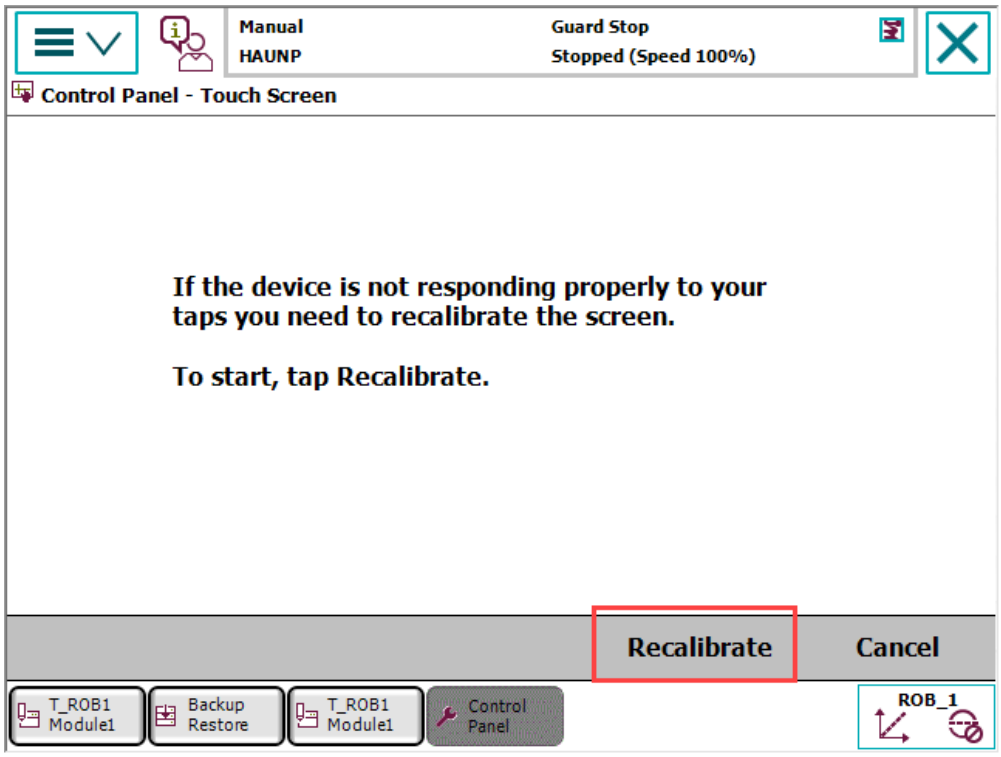
V.2. Calibration màn hình cảm ứng

Bước 1: Vào ***Control Panel***

Bước 2: Chọn ***Touch Screen***



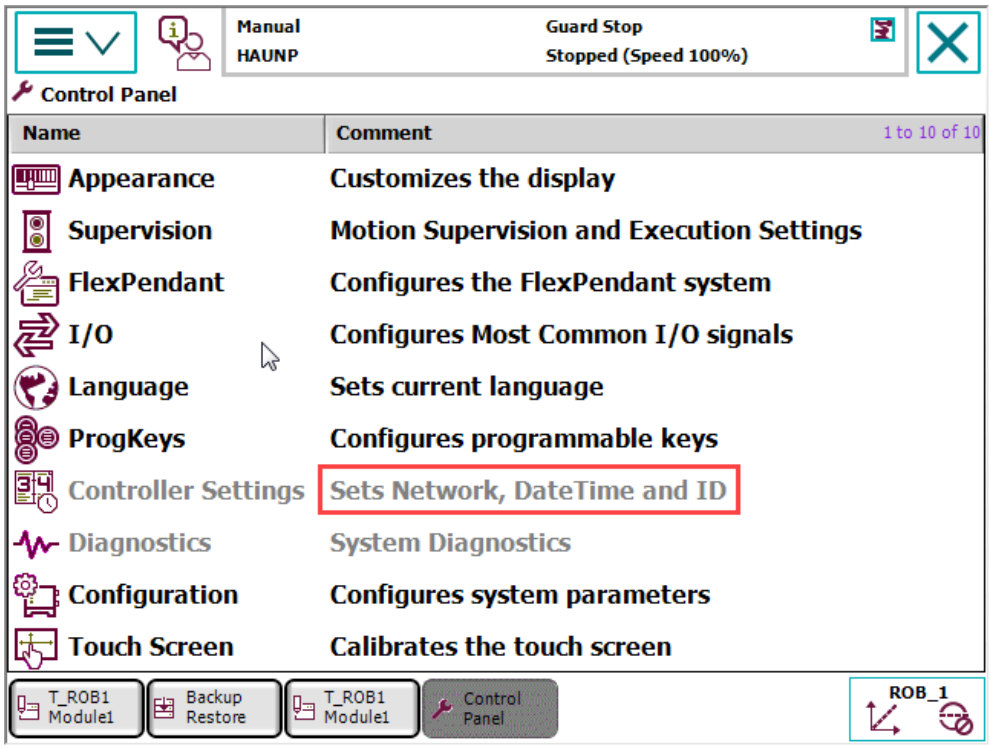
Bước 3: Nhấn ***Recalibrate*** và nhấn vào các điểm yêu cầu trên màn hình. Sau đó nhấn ***OK*** để khơi động lại robot.



V.3. Cài đặt ngày, giờ và địa chỉ IP robot

Bước 1: Vào ***Control Panel***

Bước 2: Chọn ***Sets Network, Date Time and ID***

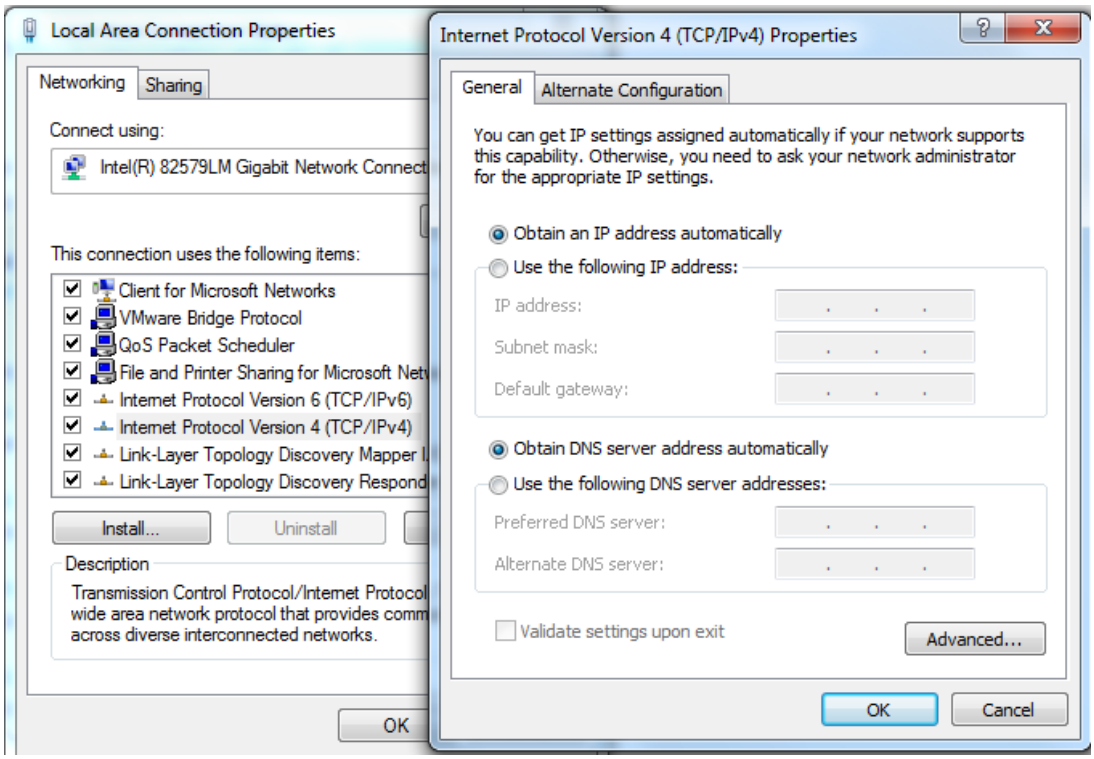


Bước 3: Điền địa chỉ IP và ngày giờ

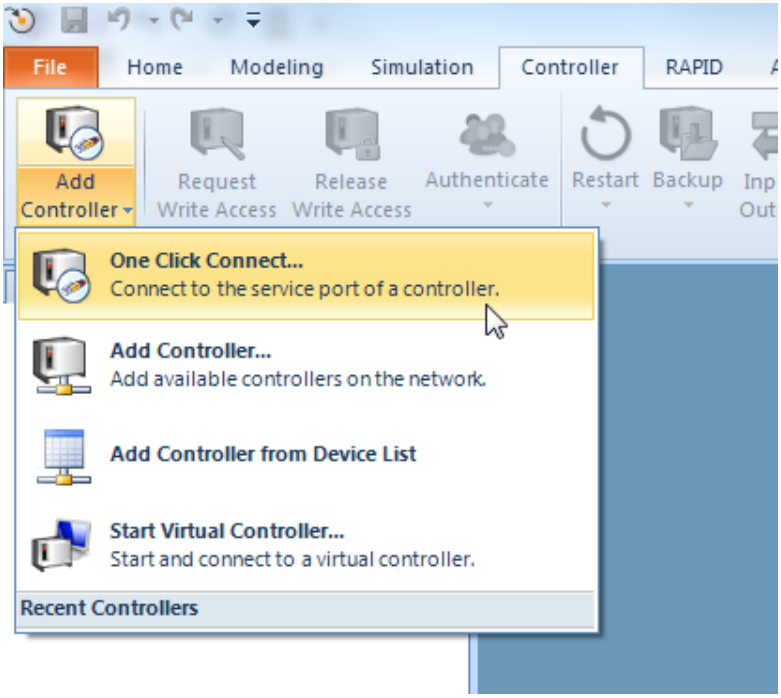
VI. Đồng bộ dữ liệu giữa máy tính và robot

Bước 1: Cắm dây mạng Ethernet (RJ45) vào cổng ***Services*** trong tủ điều khiển robot.

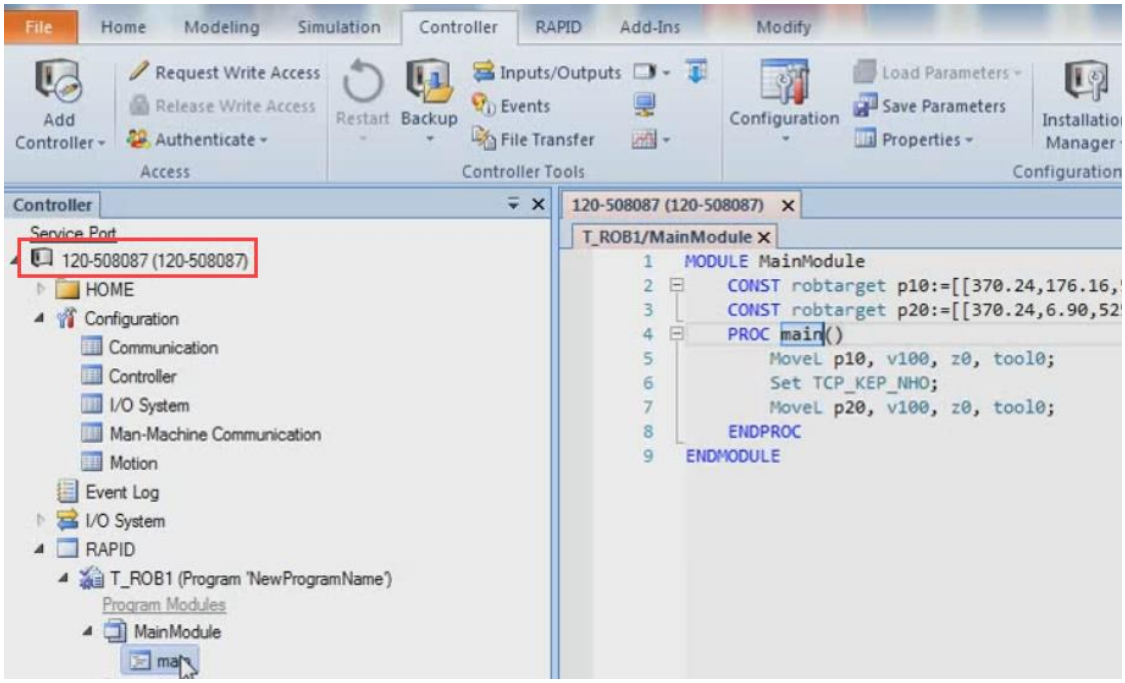
Bước 2: Xóa IP trên máy tính (chọn kết nối IP tự động)



Bước 3: Trên phần mềm ***Robot Studio*** nhấn vào ***Controller*** chọn ***One Click Connect*** để kết nối máy tính với robot.

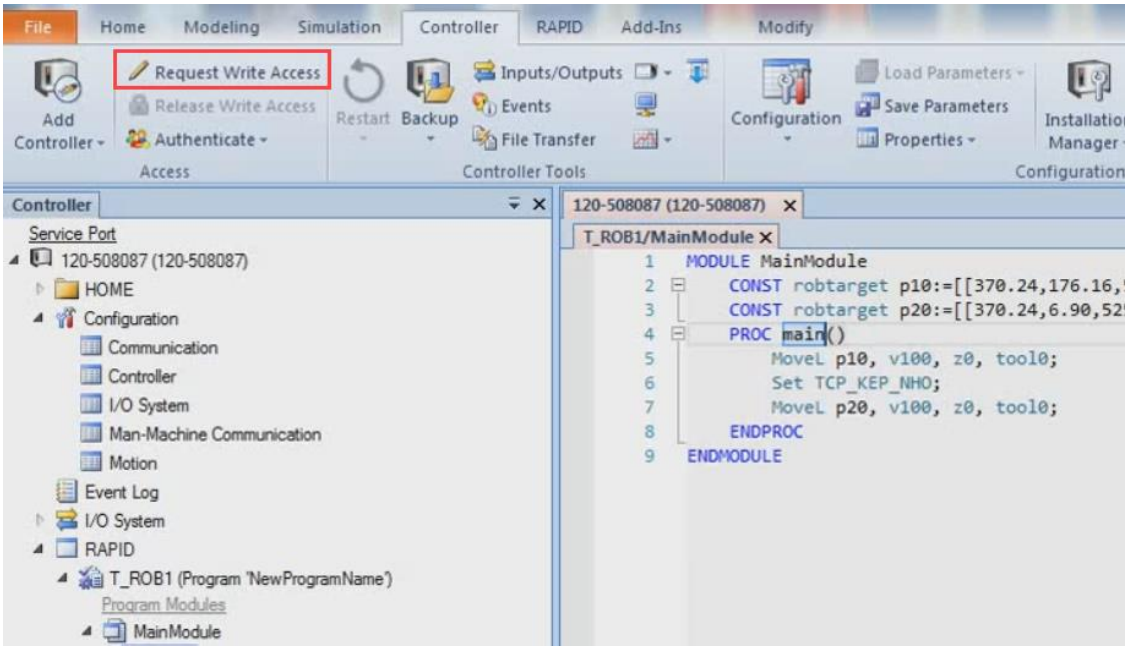


Bước 4: Sau khi kết nối thành công, trên phần mềm ***Robot Studio*** sẽ hiện thông tin robot kết nối và các dữ liệu trên robot đó

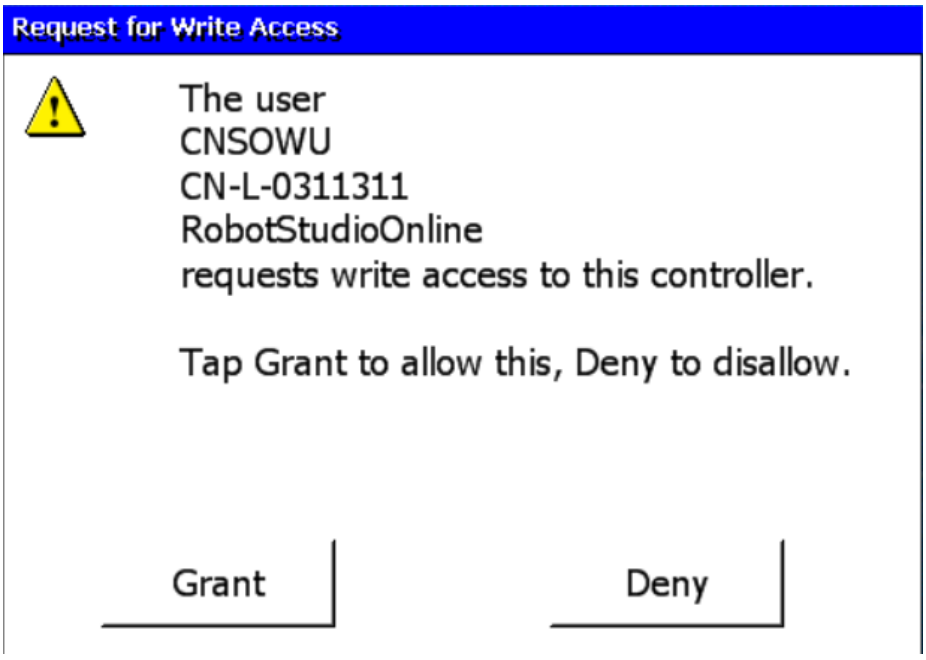


VI.1. Truy cập chỉnh sửa dữ liệu trên robot

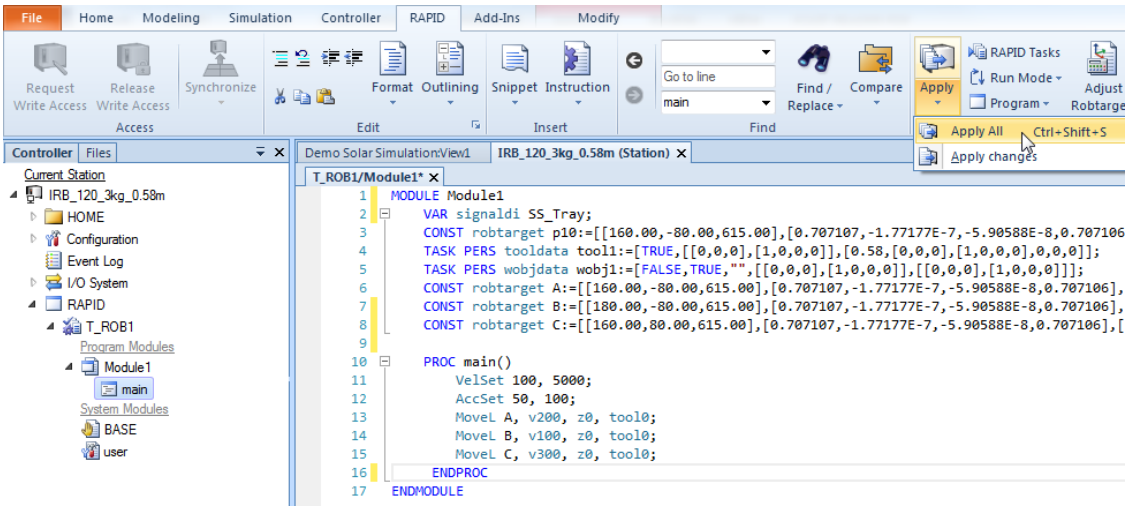
Bước 1: Nhấn vào ***Request Write Access*** để truy cập chỉnh sửa chương trình robot.



Khi đó trên màn hình tay dạy robot sẽ hiển thị yêu cầu truy cập. Chọn ***Grant*** để đồng ý.

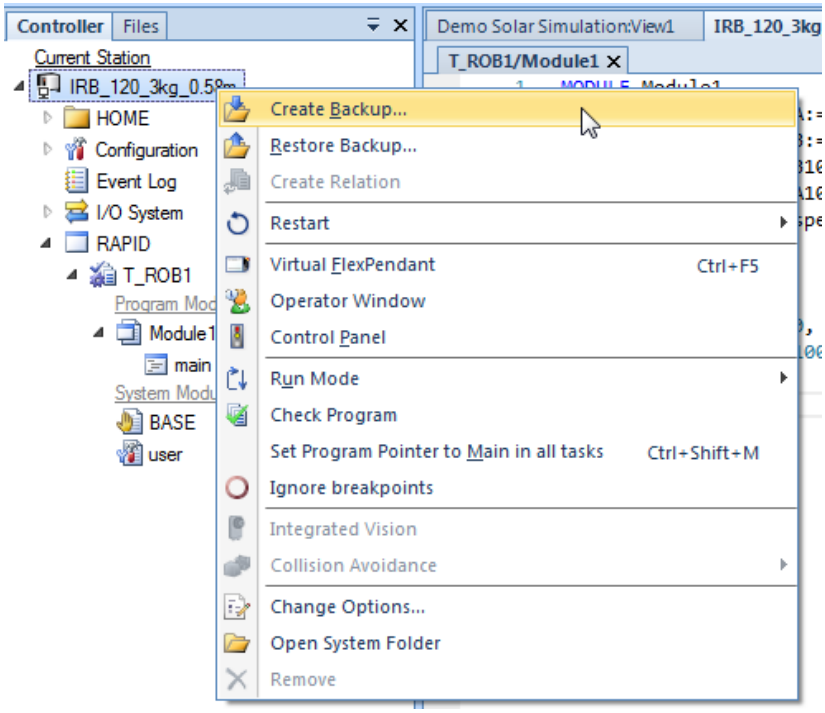


Bước 2: Sau khi thay đổi chương trình robot, chuyển qua tab ***RAPID*** trên phần mềm ***Robot Studio*** và chọn ***Apply All*** để đổ chương trình trên máy tính xuống robot.

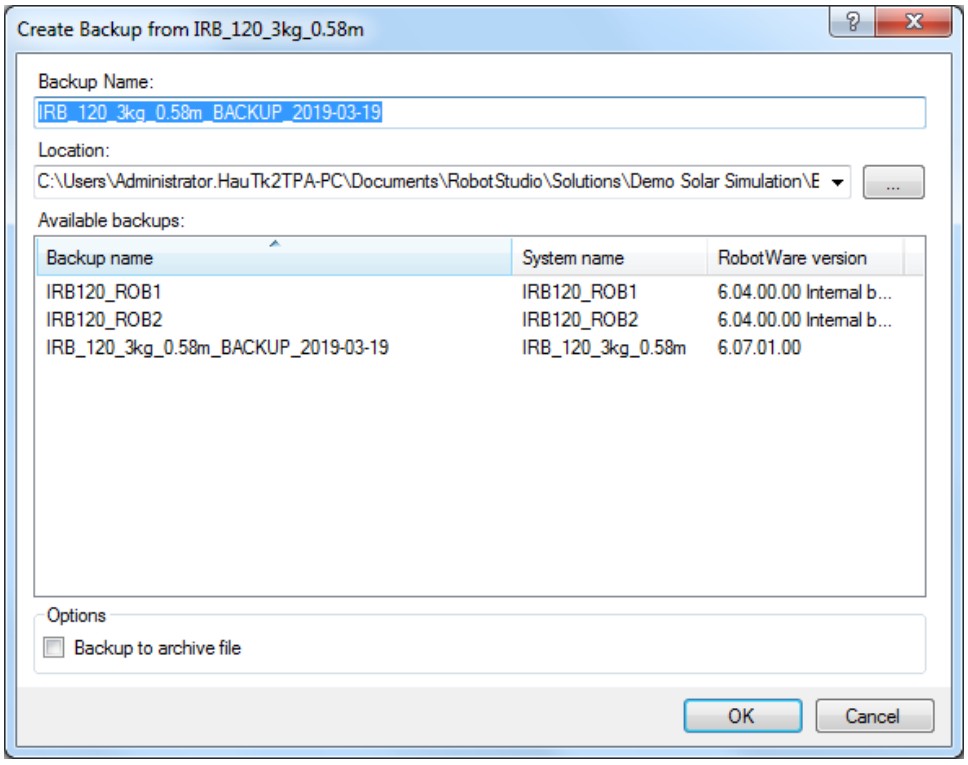


VI.2. Backup dữ liệu trên robot về máy tính

Bước 1: Click chuột phải vào robot đã kết nối và chọn ***Create Backup***

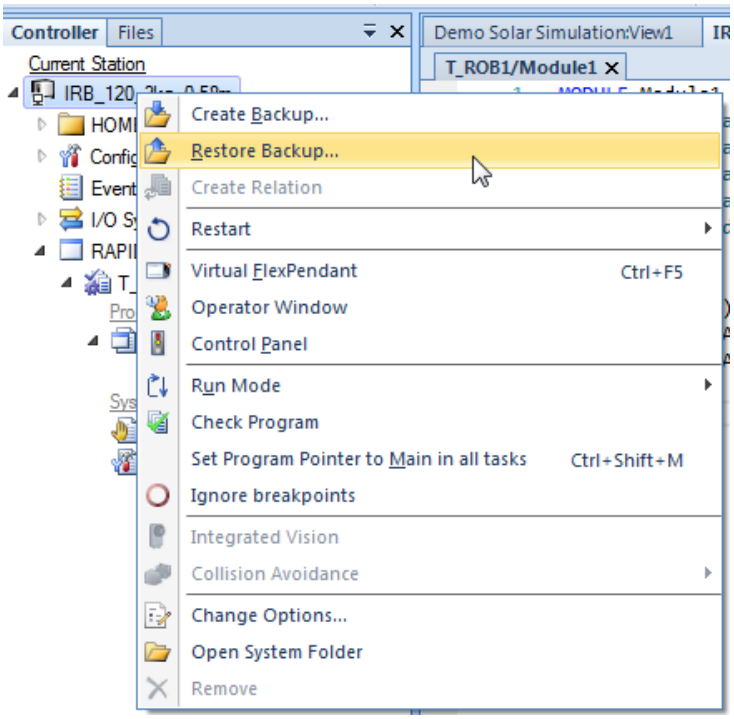


Bước 2: Chọn thư mục cần ***Backup*** rồi nhấn ***OK***

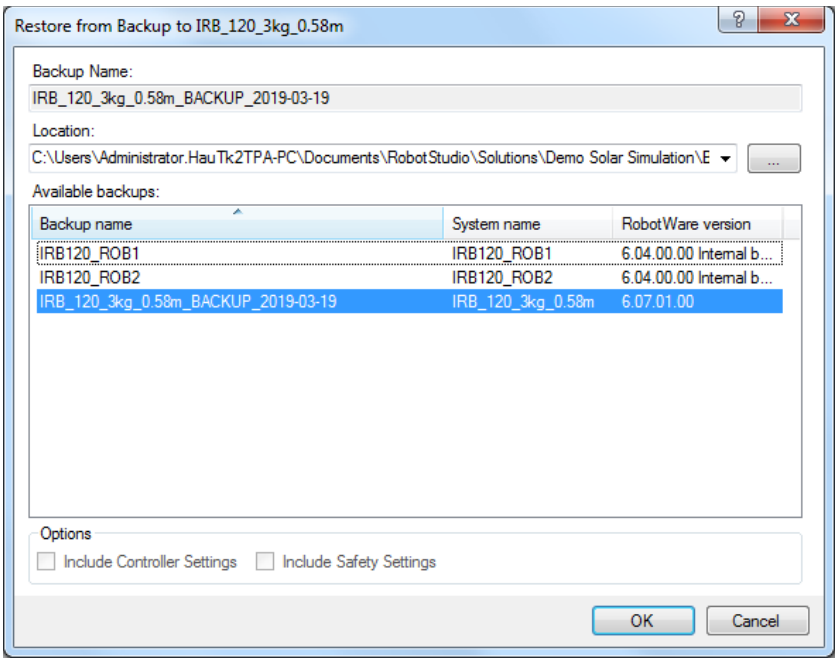


VI.3. Restore dữ liệu trên từ máy tính vào robot

<u>Bước 1:</u> Click chuột phải vào robot đã kết nối và chọn ***Restore Backup***

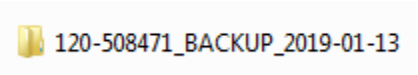


Bước 2: Chọn thư mục chứa dữ liệu cần ***Restore*** rồi nhấn ***OK.*** Lúc này robot sẽ khởi động lại.



Chú ý:

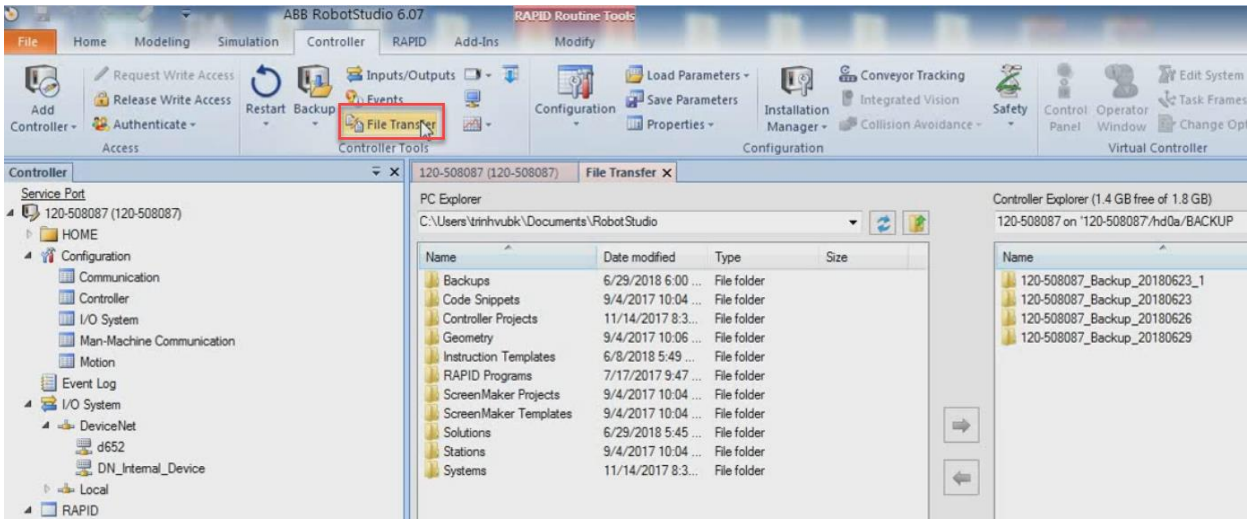
- Dữ liệu ***Restore*** vào robot phải là dữ liệu đã ***Backup*** trước đây của chính robot đó. Thông thường khi ***Backup***, robot sẽ tạo thư mục có tên mặc định gồm mã ***Seri*** của robot.



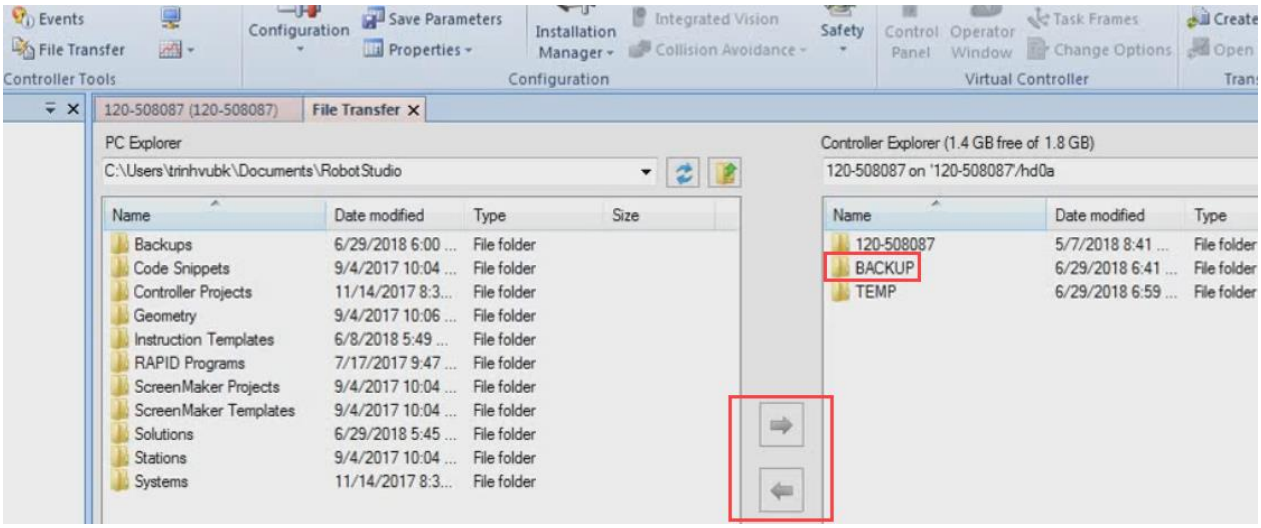
- Việc ***Restore*** lại file không phải của robot thì sau khi robot khởi động lại sẽ báo lỗi ***file CFG*** và robot bị mất gốc tọa độ. Đây chính là file hệ thống của robot và mỗi robot sẽ khác nhau. Trường hợp này bắt buộc phải ***Calibration*** các trục robot.
- Chỉ có thể lấy dữ liệu ***Code lập trình của robot này sang robot kia*** bằng cách ***Copy-Paste*** và đổ chương trình từ máy tính xuống robot. Không được dùng ***Restore***.

VI.4. Truyền dữ liệu giữa máy tính và bộ nhớ robot

Bước 1: Sau khi kết nối robot với máy tính, và yêu cầu truy cập vào bộ nhớ robot thì nhấn vào biểu tượng ***File Transfer*** trên phần mềm ***Robot Studio***

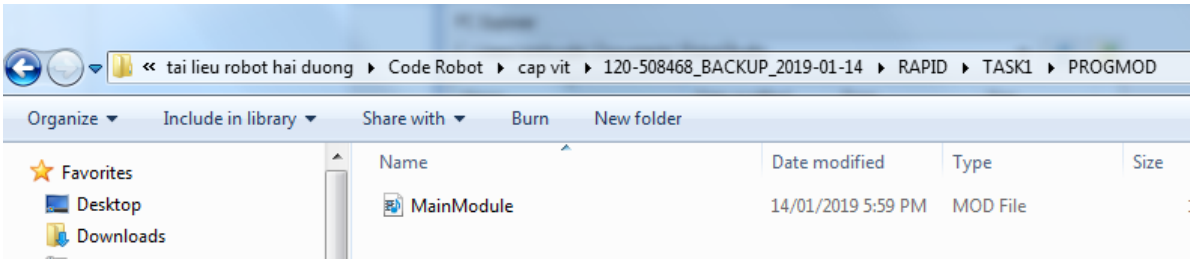


Bước 2: Chọn folder cần truyền-nhận và nhấn vào phím mũi tên. Chú ý dữ liệu robot được lưu trữ trong folder ***BACKUP***



VI.5. Xem và chỉnh sửa file chương trình offline

Bước 1: Tìm file MainModule theo đường dẫn **Folder→RAPID →TASK1→PROGMOD**

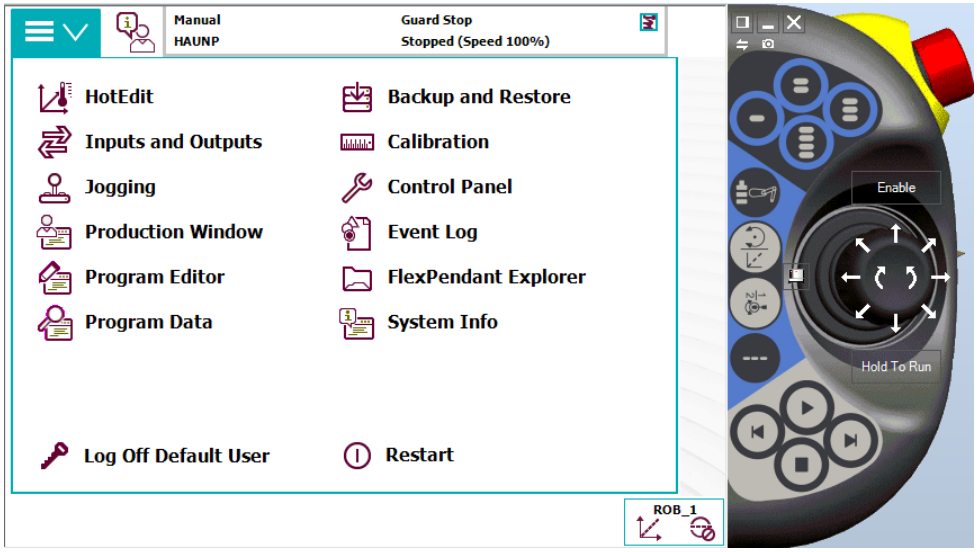


Bước 2: Mở file chương trình bằng phần mềm ***Robot Studio*** hoặc ***Notepad.*** Nếu có chỉnh sửa thì sau khi chỉnh sửa xong thì nhấn ***Strl+S*** để lưu lại.

Bài 3: Các lệnh lập trình di chuyển robot cơ bản

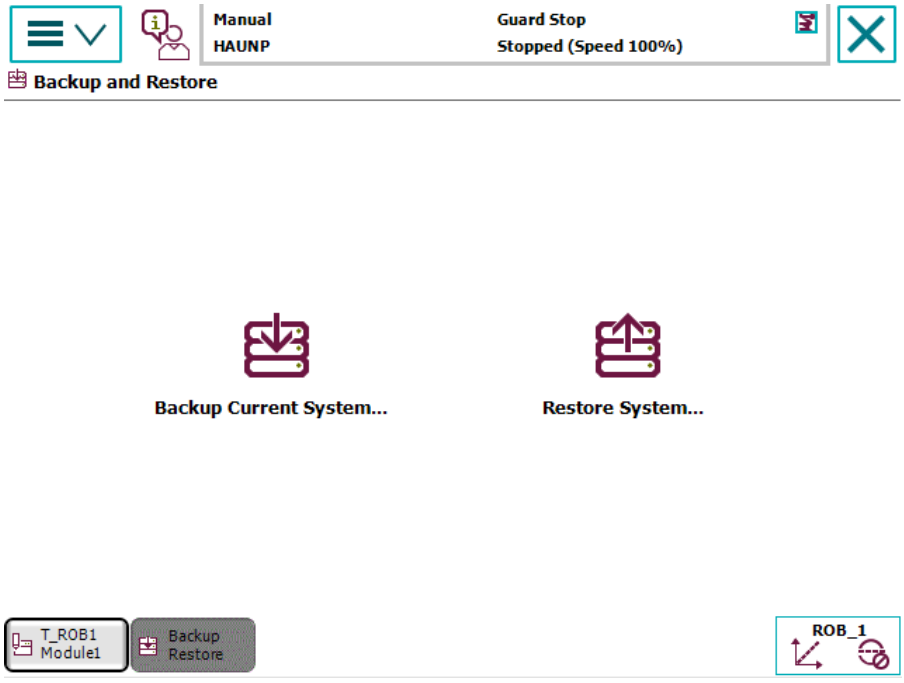
I. Tạo một chương trình lập trình mới

Ngoài màn hình giao diện chính, chọn ***Program Editor.***



II. Sao lưu và khôi phục lại chương trình robot

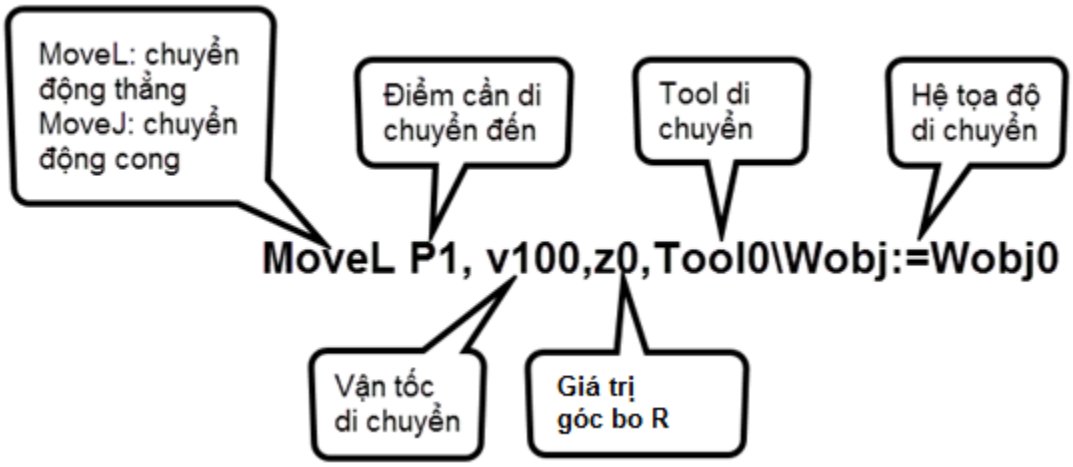
Chương trình mặc định của robot chính là chương trình đang có ở trong ***Program Editor.*** Robot sẽ tự động lưu lại chương trình đang viết ở trong ***Program Editor*** (lưu liên tục). Muốn lưu lại chương trình đang viết để viết chương trình mới thì chọn ***Backup***. Nếu muốn mở lại chương trình cũ thì chọn ***Restore.***



Đặt tên folder nếu muốn ***Backup***, chọn folder nếu muốn ***Restore.*** Sau đó nhấn và lựa chọn cần thực hiện.

III. Các lệnh di chuyển cơ bản

- ***MoveL:*** Di chuyển robot theo đường thẳng
- ***MoveJ:*** Di chuyển robot theo đường cong



Trong đó:

Z: tính bằng mm

V: tính bằng mm/s (Vmax=5000mm/s)

<u>Chú ý:</u> Khuyến nghị nên chạy với tốc độ <500mm/s

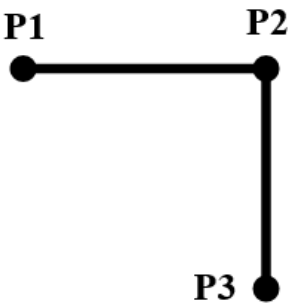
➢ Có thể bỏ dòng lệnh Wobj:=Wobj0 nếu sử dụng hệ tọa base để di chuyển robot

Robot đang ở điểm P1 cần di chuyển đến P2 với tool0 và hệ tọa với lệnh MoveL

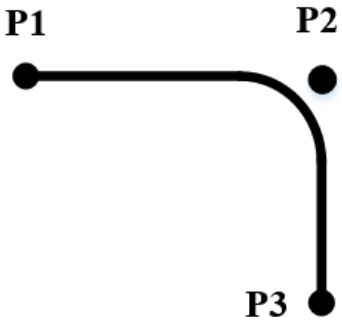
⇨ **MoveL P2, v20, z0,tool0;**

Robot đang ở điểm P1 cần di chuyển đến P2 sau đó di chuyển đến P3 với lệnh MoveL. Dưới đây là 2 cách di chuyển khi muốn bo cung R khi robot đến điểm P2

```
MoveL P1, v100, z0,tool0;
MoveL P2, v100, z0,tool0;
MoveL P3, v100, z0,tool0;
```

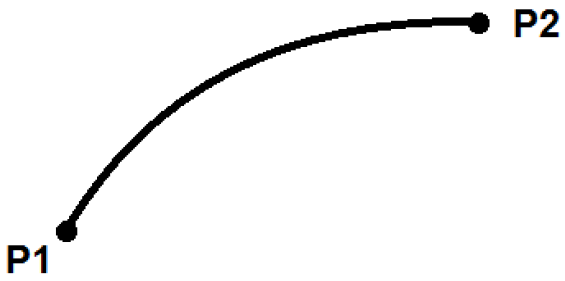



```
MoveL P1, v100, z0,tool0;
MoveL P2, v100, z50,tool0;
MoveL P3, v100, z0,tool0;
```



Robot đang ở điểm P1 cần di chuyển đến P2 với lệnh MoveJ

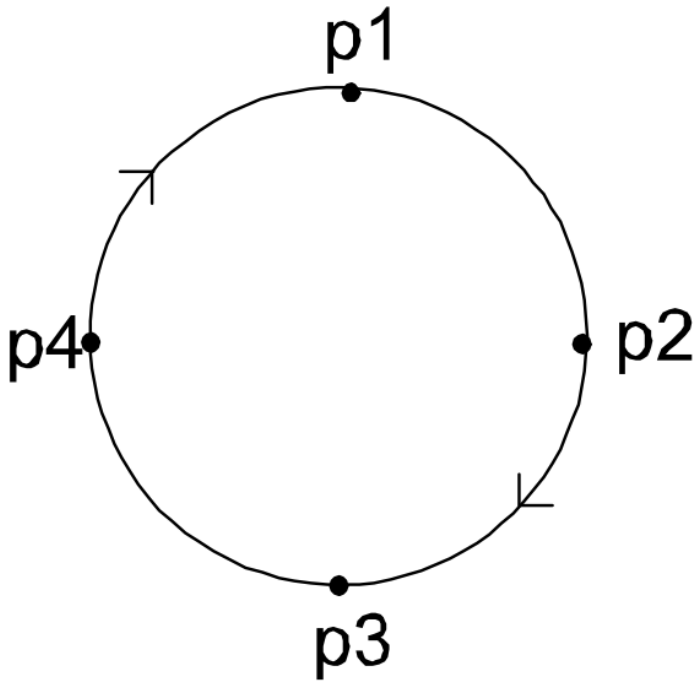
⇨ **MoveJ P2, v20, z0,tool0;**



- ***MoveC:*** Di chuyển robot theo cung tròn tạo từ hai điểm

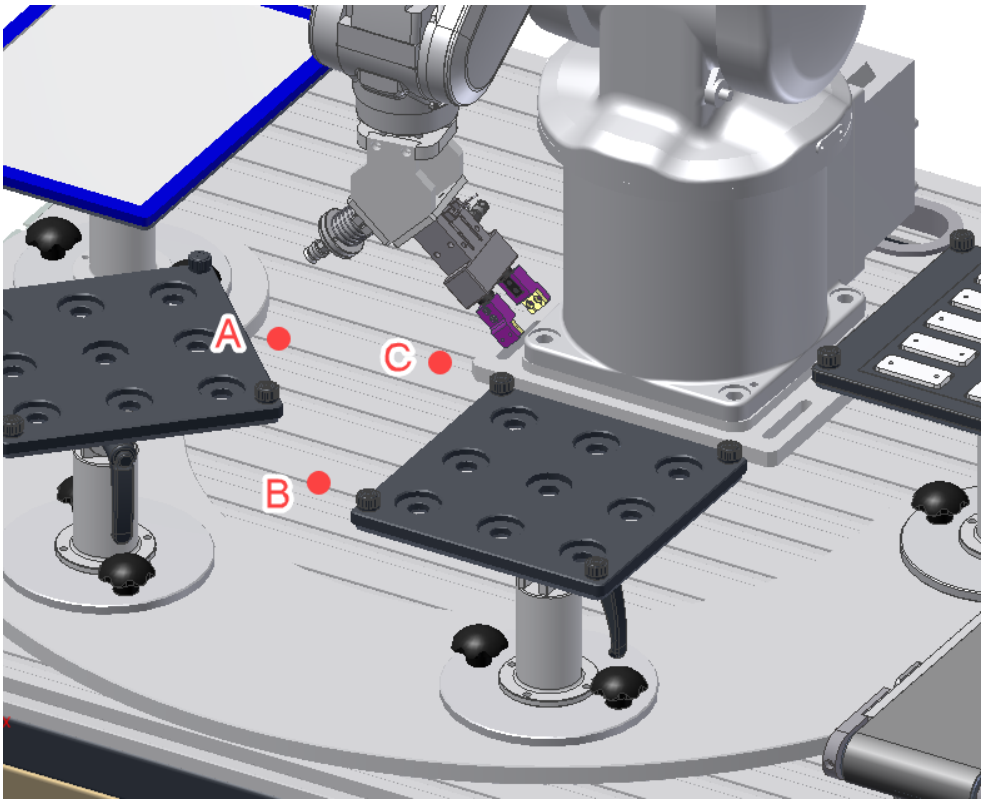
Robot đang ở điểm P1 cần di chuyển đến P2, P3, P4 rồi về P1 với lệnh MoveC

⇨ **MoveC P1, P2, v20, z0,tool0;**
MoveC P2, P3, v20, z0,tool0;
MoveC P3, P4, v20, z0,tool0;

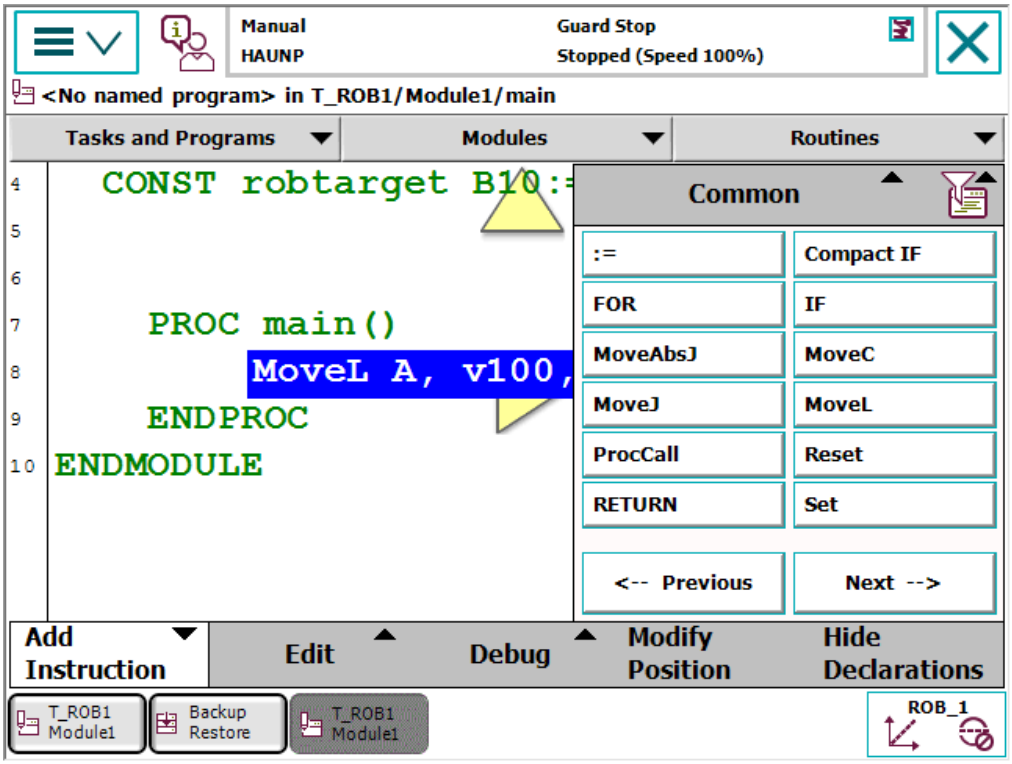


IV. Viết chương trình di chuyển robot.

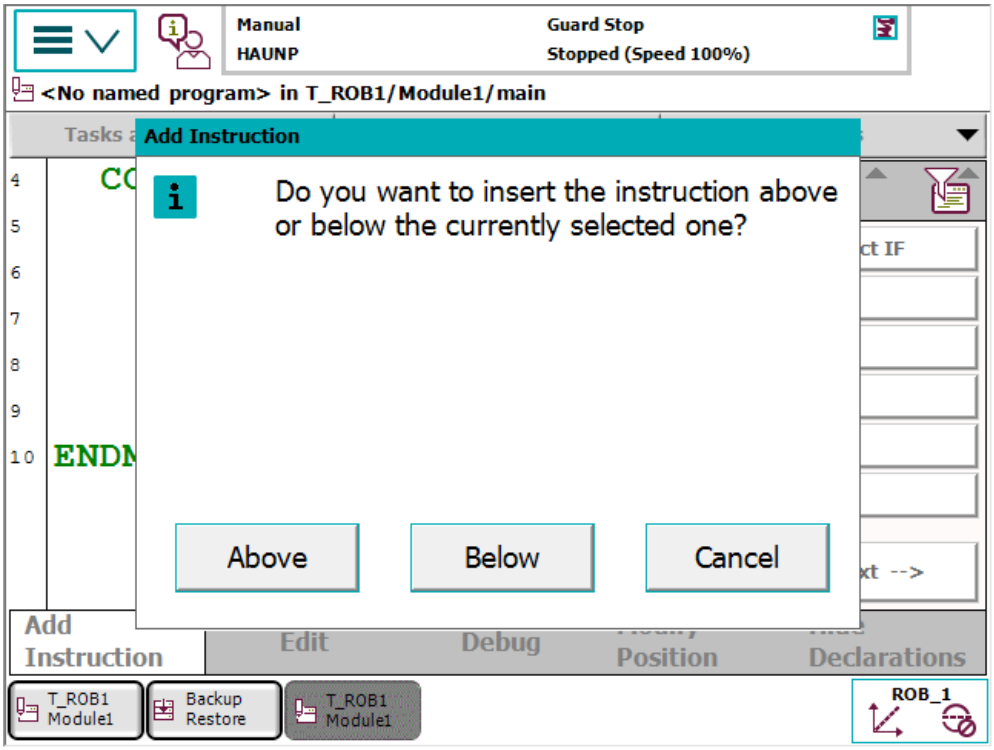
Viết chương trình di chuyển robot đi từ điểm A đến điểm B rồi đến điểm C và lặp lại chu trình trên theo các lệnh Move L, Move J



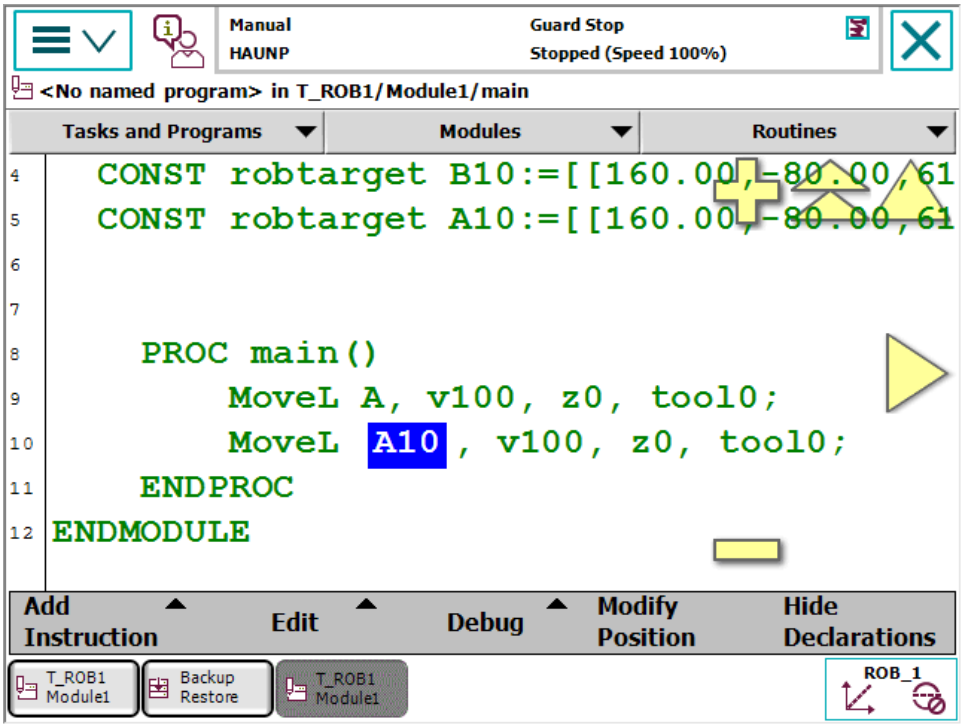
<u>Bước 1:</u> Chọn lệnh ***MoveL*** hoặc ***MoveJ*** trong tab ***Add Struction***



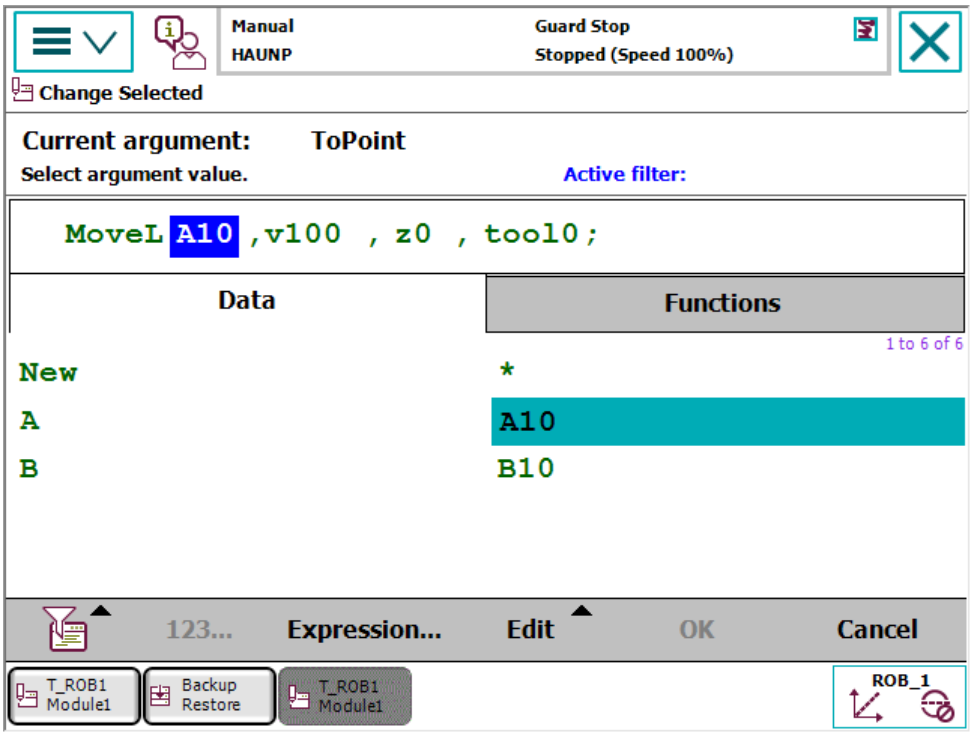
Thông báo chọn lệnh trên hoặc dưới dòng lệnh trước



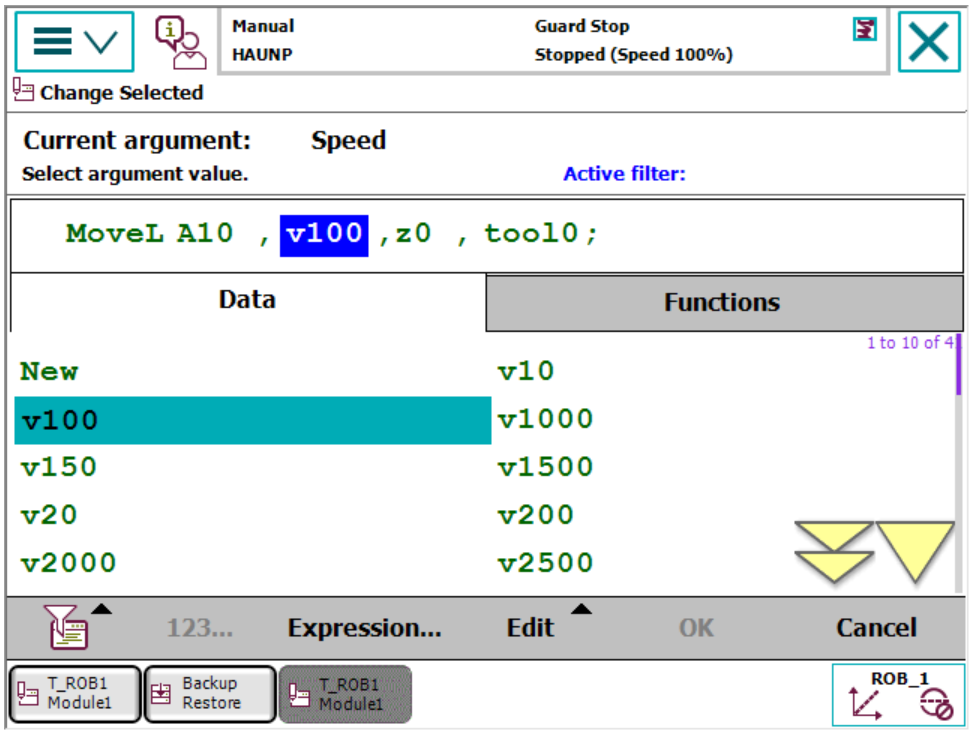
Bước 2: Nhấn 2 lần vị trí trên câu lệnh cần sửa



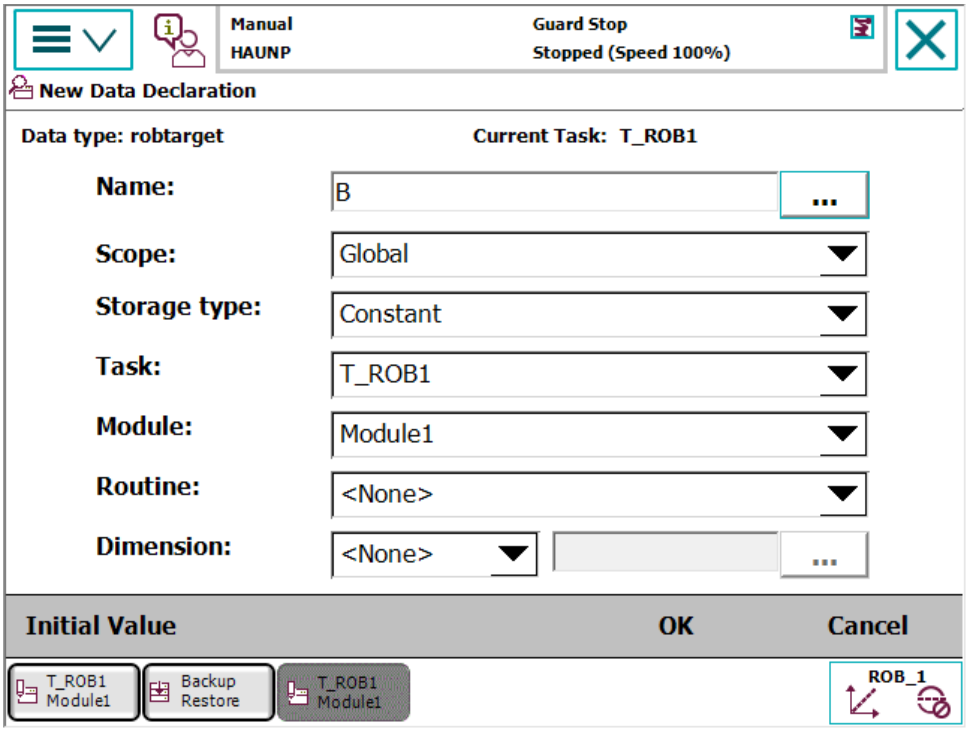
Bước 3: Chọn ***New*** để tạo điểm mới



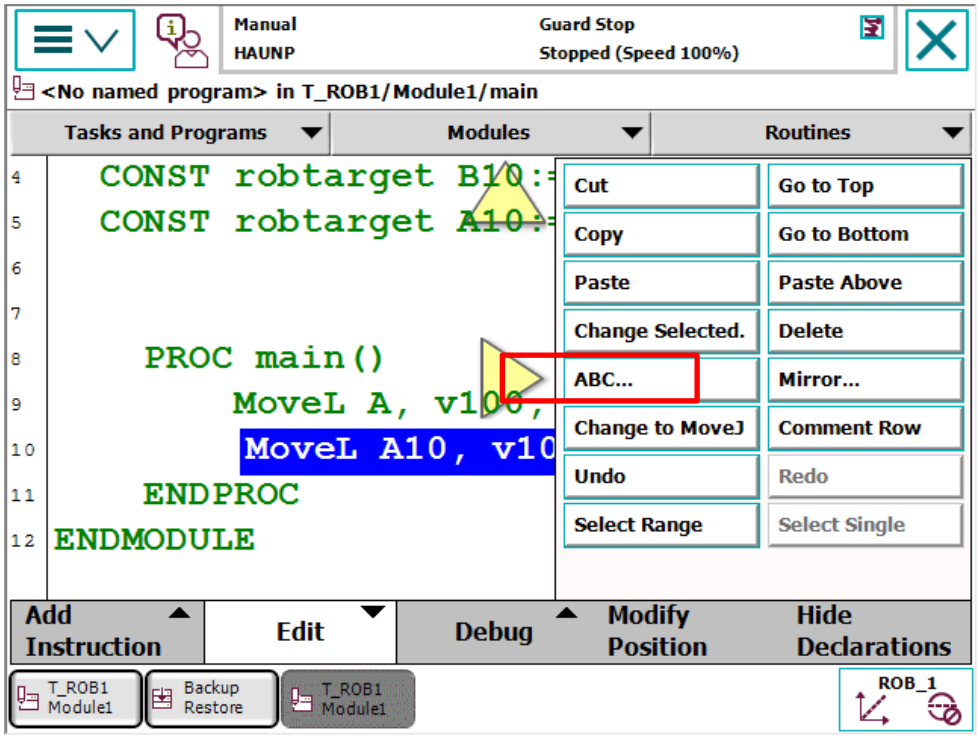
Có thể thay đổi các giá trị khác trong câu lệnh bằng cách nhấn vào giá trị đó



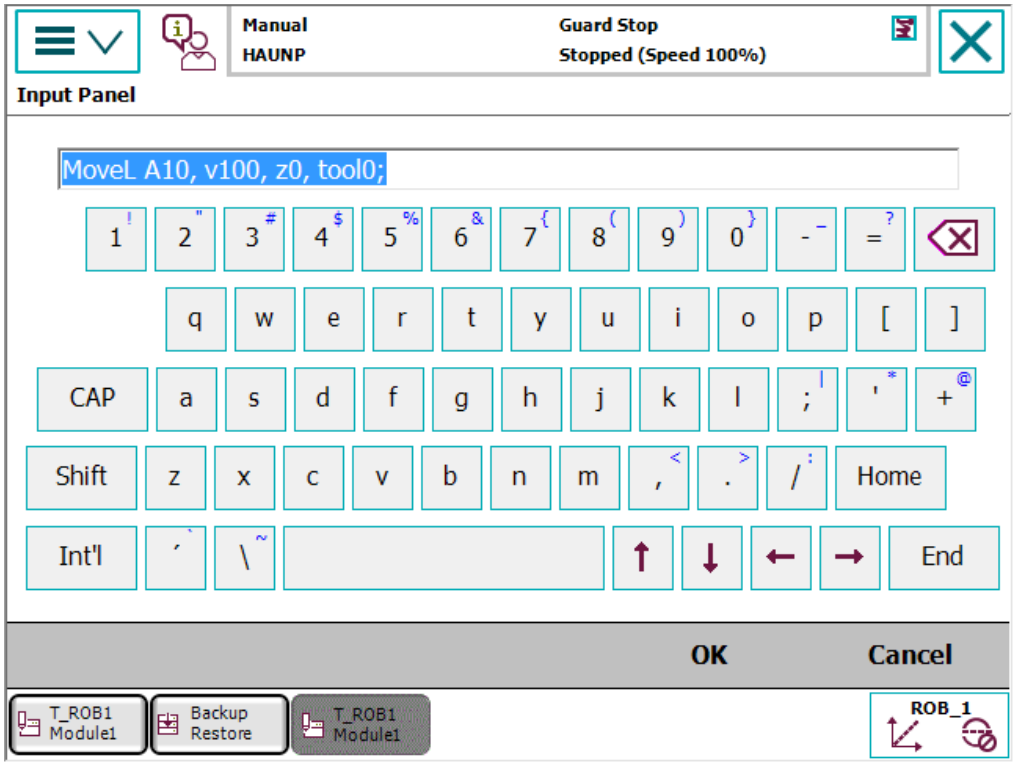
<u>Bước 4:</u> Điền tên cần lưu vào ô ***Name*** hoặc chọn giá trị cần thay đổi rồi nhấn ***OK***



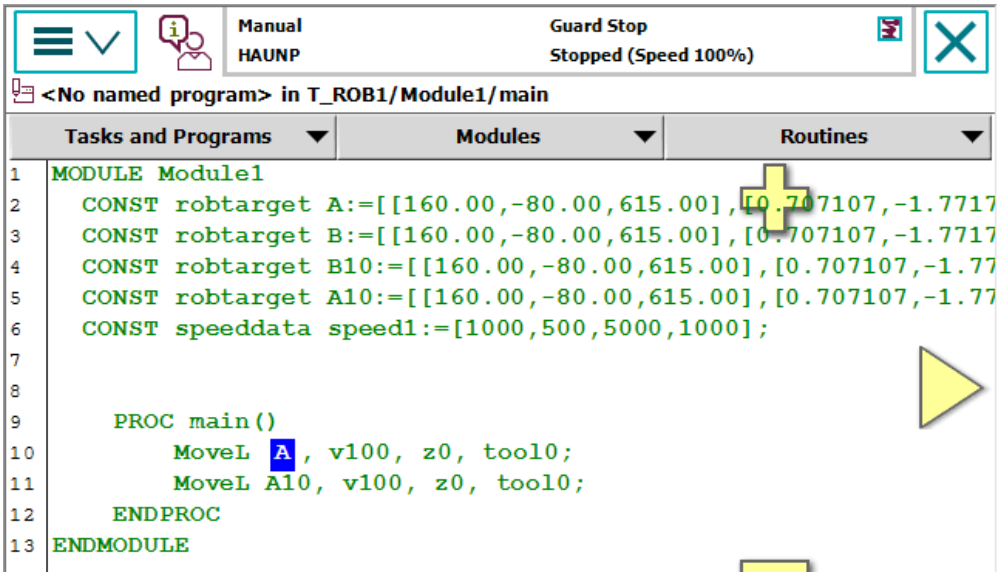
<u>Lưu ý:</u> Có thể thay đổi giá trị bằng cách chọn dòng lệnh cần sửa rồi chọn ***ABC..*** trong tab ***Edit***



Sửa giá trị trong câu lệnh rồi nhấn ***OK***



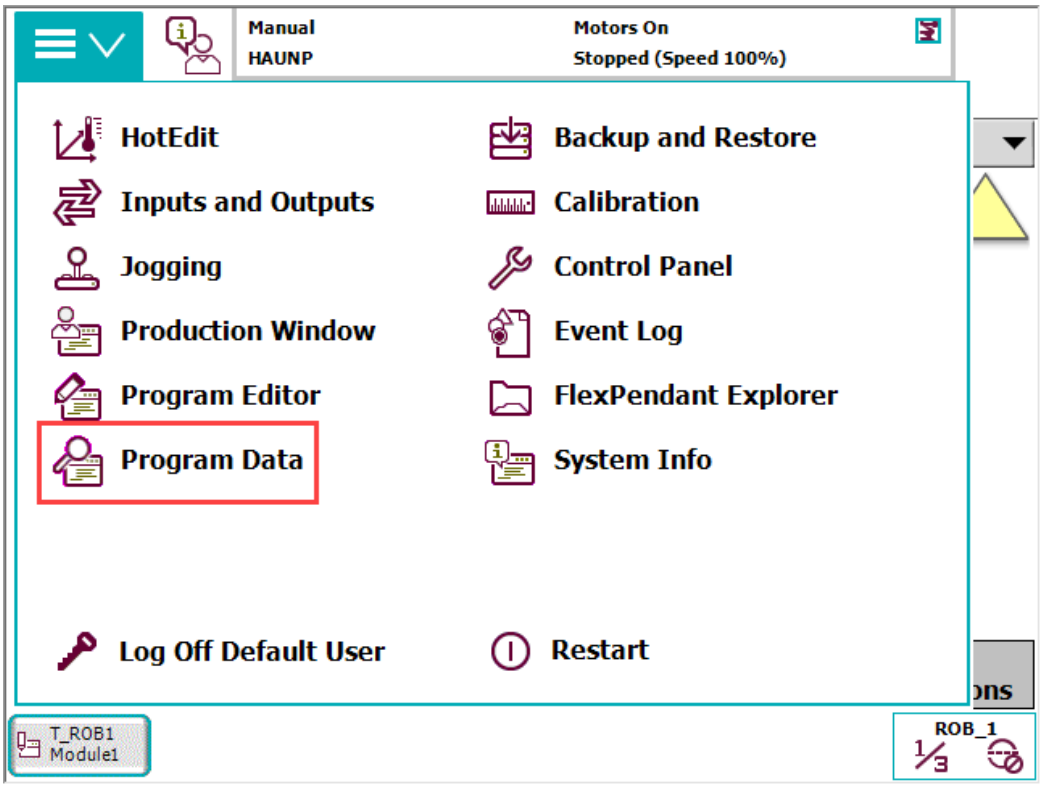
Có thể thay đổi tọa độ điểm bằng cách di chuyển robot đến vị trí điểm đó rồi chọn điểm cần thay đổi tọa độ và nhấn ***Modify Position***



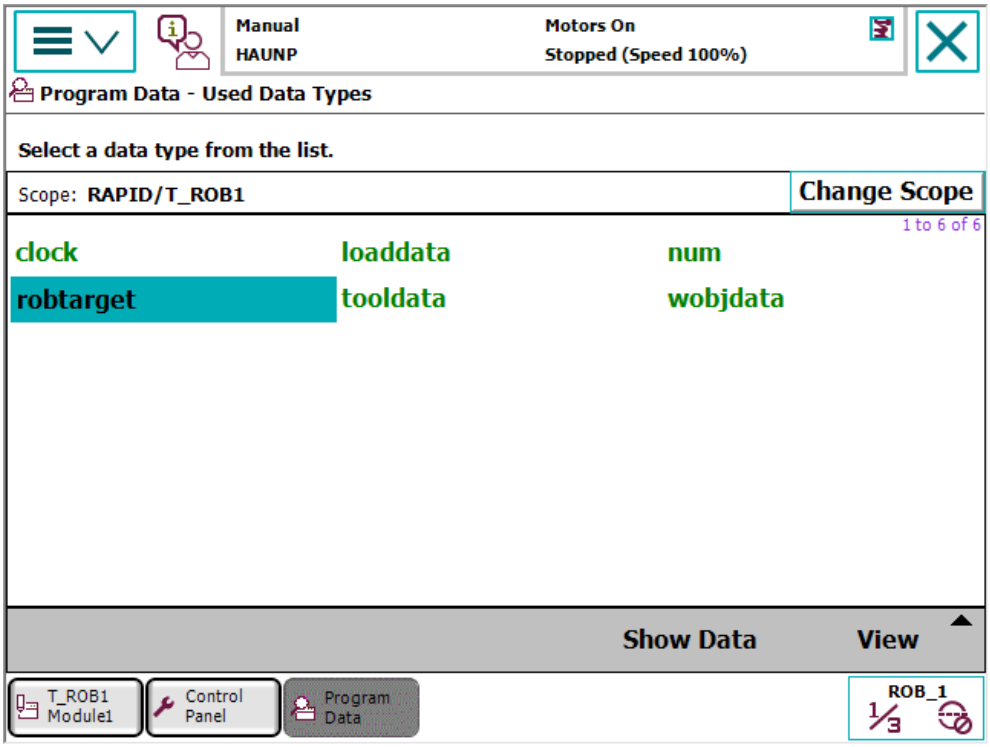
❖ **Quản lý các điểm tọa độ robot**

Trong cài đặt này có thể tạo điểm mới, sửa tọa độ điểm mà không cần phải dùng các lệnh ***Move***

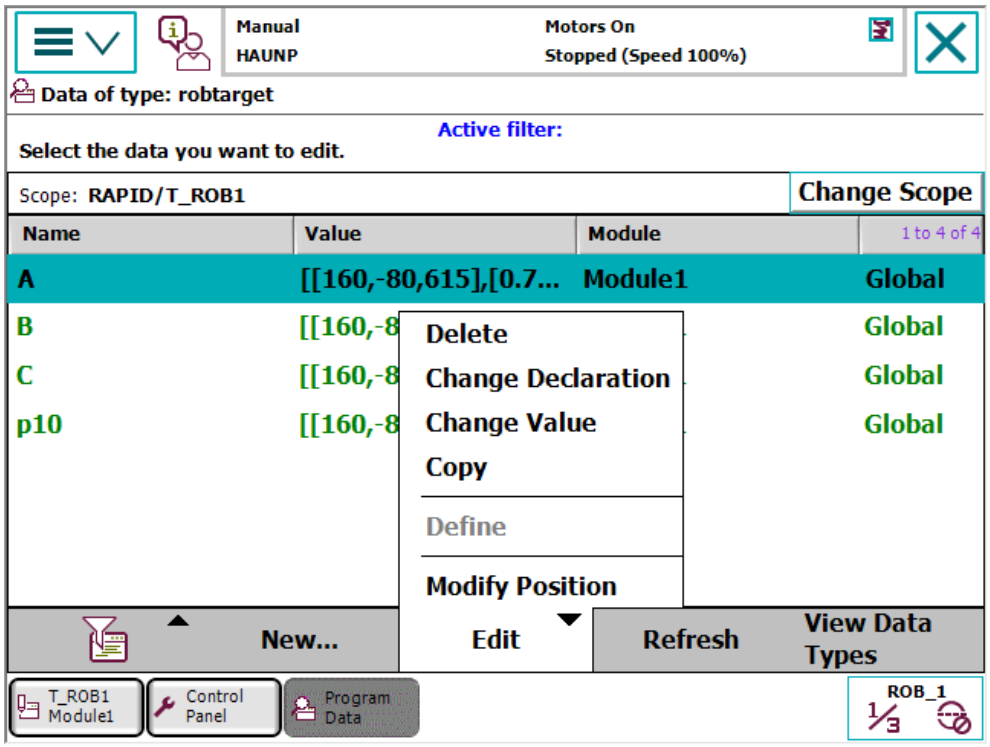
Bước 1: Chọn ***Program Data***



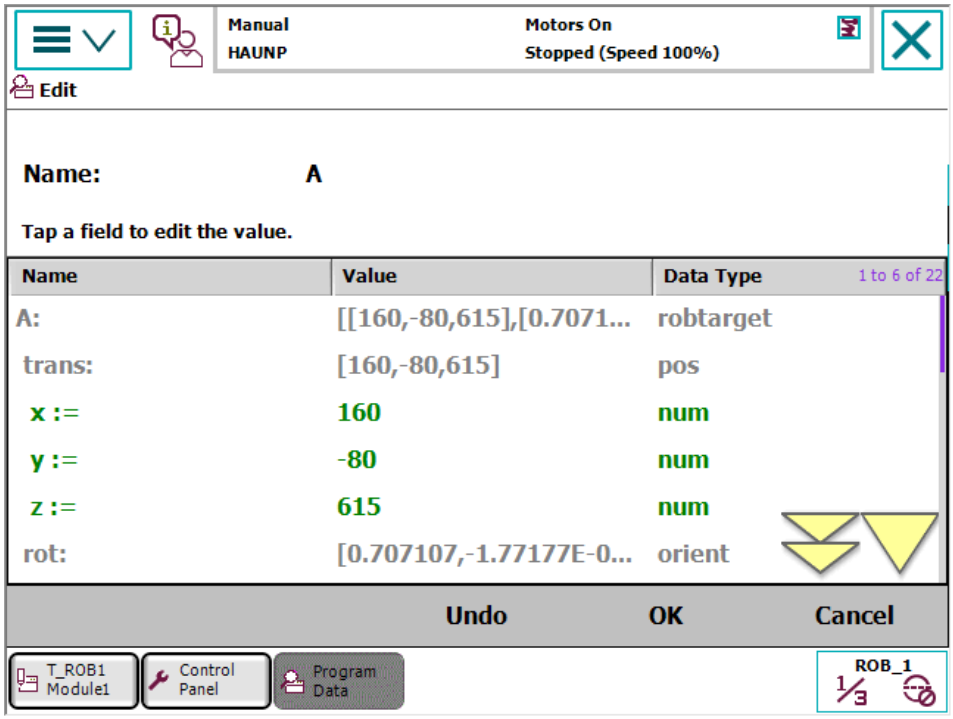
Bước 2: Chọn ***robtarget***. Nếu không có thì chọn ***View*** → ***All Data Types*** rồi lựa chọn ***robtarget***



Bước 3: Chọn ***New*** để tạo điểm mới. Tọa độ điểm này chính là vị trí robot hiện tại. Chọn vào điểm rồi chọn ***Edit*** để chỉnh sửa điểm



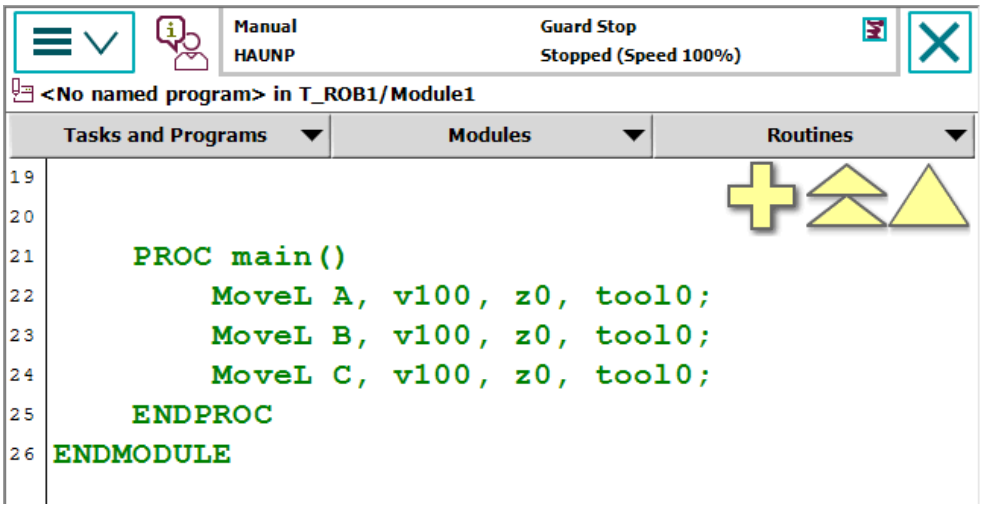
- ***Delete:*** xóa điểm
- ***Change Declaration:*** chỉnh sửa tên điểm, kiểu điểm (biến, hằng số..)
- ***Change Value:*** thay đổi giá trị điểm (tọa độ X, Y, Z)



- ***Modify Position:*** update tọa độ điểm ứng với tọa độ hiện tại của robot

Bước 4: nhấn ***OK*** để hoàn thành

❖ **Chương trình robot**



Nhấn nút  trên tay dạy để robot chạy theo chương trình đã lập trình

Nhấn nút  trên tay dạy để dừng robot.

Nhấn nút  trên tay dạy để chạy lại câu lệnh phía trước

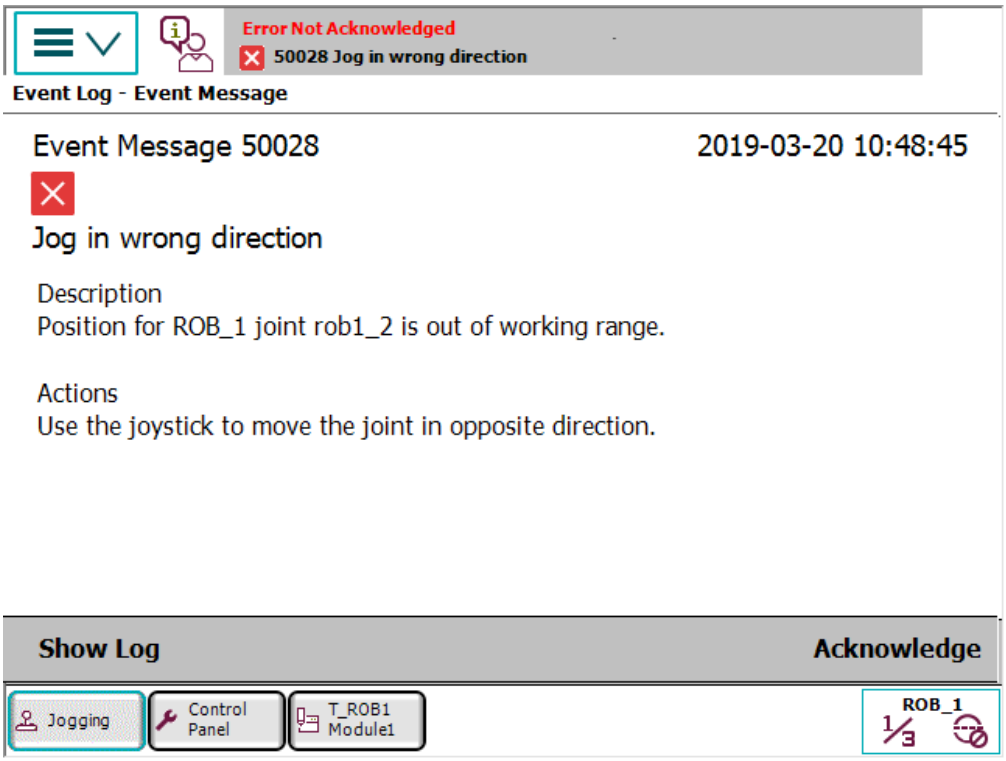
Nhấn nút  trên tay dạy để chạy câu lệnh tiếp theo.

Chú ý: Cần nhấn giữ ***công tắc kích hoạt*** trên tay dạy mới có thể di chuyển robot.

❖ **Thực hiện di chuyển robot đi từ điểm A đến điểm B rồi đến điểm C theo các lệnh Move J, Move C để biết được tính chất của các lệnh Move.**

V. Các lỗi thường gặp khi di chuyển, lập trình robot

Khi di chuyển hoặc lập trình thì có một số lỗi xảy ra. Nếu không điều khiển thuần thục hoặc lập trình không quen thì việc robot báo lỗi sẽ thường xuyên. Việc hiển thị thông báo lỗi sẽ có mã lỗi và hướng khắc phục.



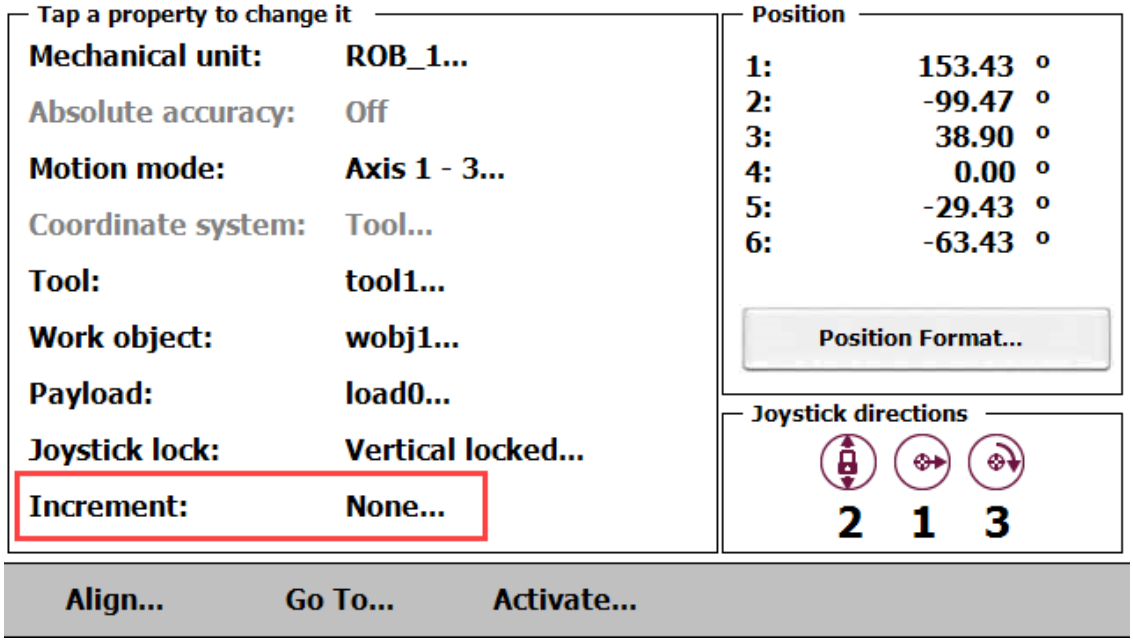
- Theo như thông báo trên, lỗi này do quá trục làm việc của 1 và 2 của robot. Giải pháp là di chuyển robot đến vị trí trong vùng làm việc. có thể tra mã lỗi 50028 trên wedsite để biết thêm nếu không hiểu hướng khắc phục trên thông báo

Bài 4: Thay đổi tốc độ di chuyển của robot

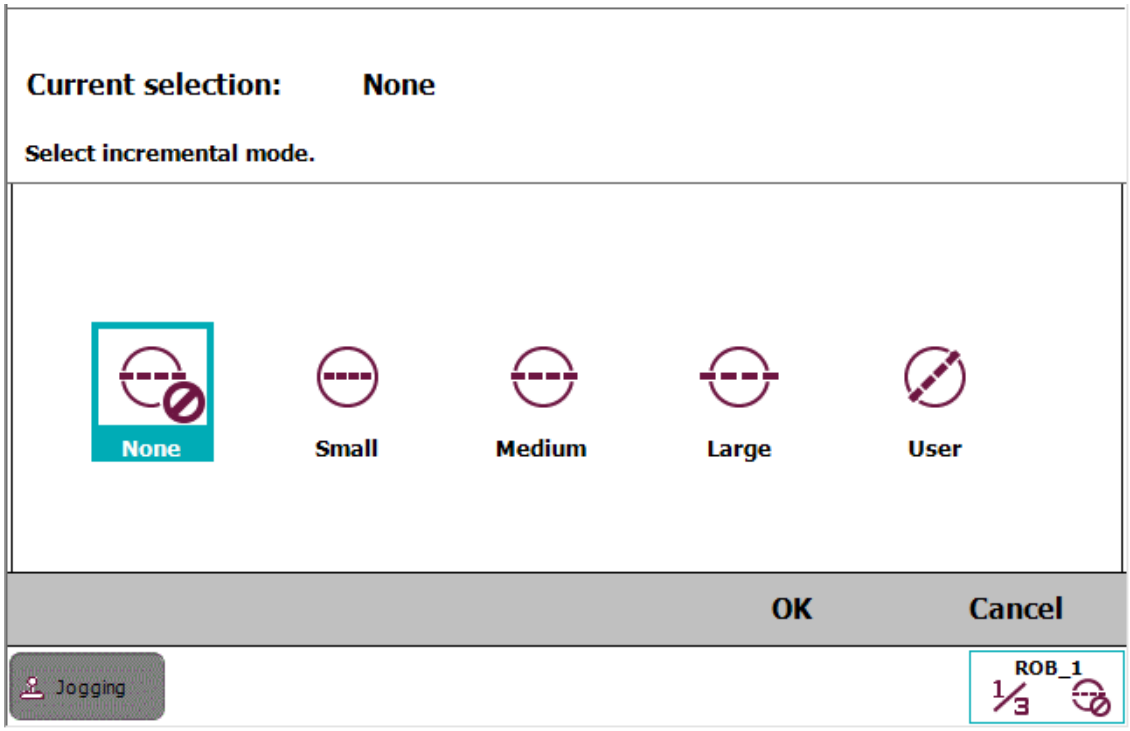
I. Thay đổi tốc độ di chuyển tối đa của robot

➢ Thay đổi tốc độ theo mức

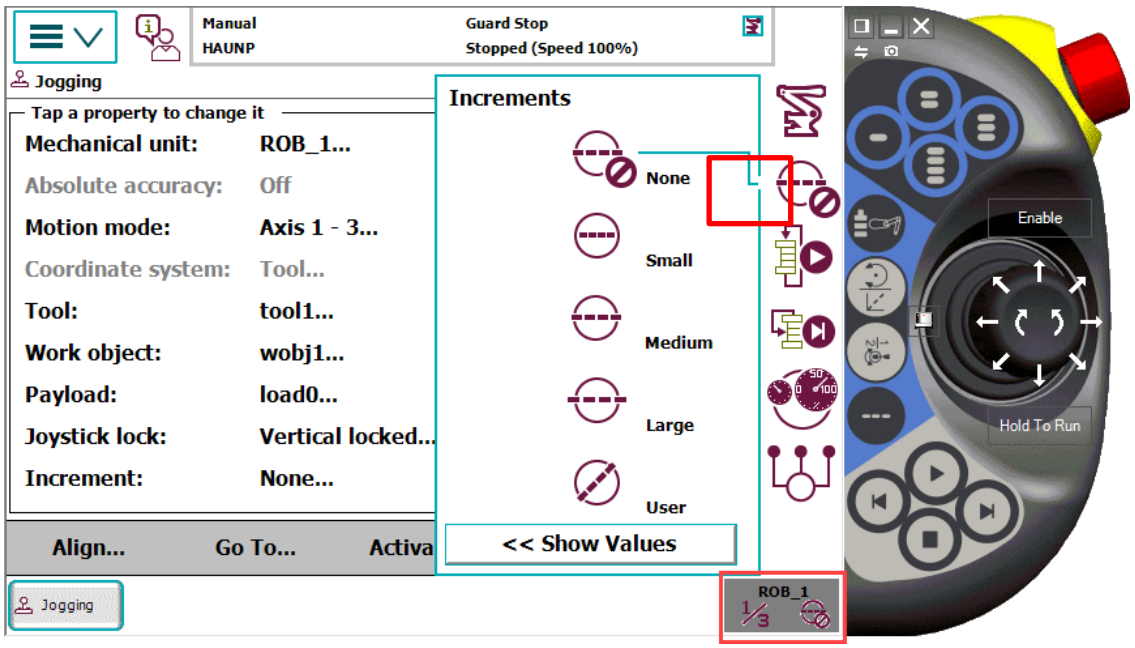
Bước 1: Chọn ***Increment***



Bước 2: Lựa chọn các mức di chuyển rồi nhấn ***OK***

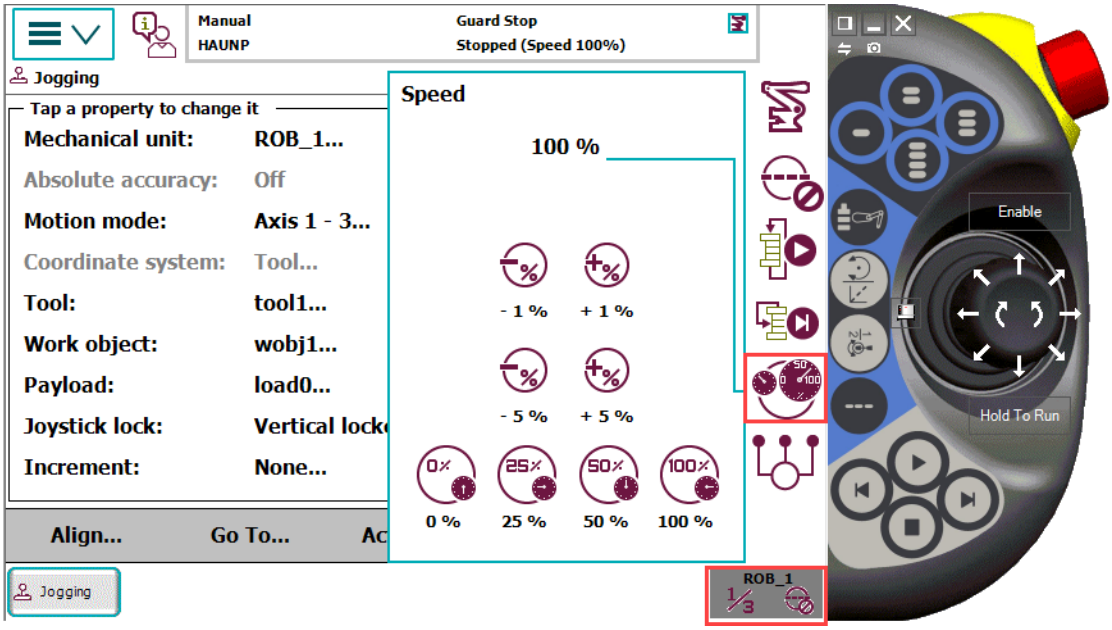


- Có thể truy cập nhanh bằng phím tắt



➢ Thay đổi tốc độ theo phần trăm của mỗi mức

Truy cập phím tắt và lựa chọn phần trăm di chuyển của mỗi mức (None, Small, Medium…)



II. Thay đổi tốc độ di chuyển toàn bộ câu lệnh trong chương trình

- Lệnh **VelSet**

VelSet dùng để tăng và giảm tốc độ robot sau vị trí câu lệnh này. Lệnh này còn dùng để tăng tốc độ lên tối đa.

VelSet 50, 5000;

Trong đó:

50: Tốc độ mong muốn bằng phần trăm của tốc độ từ chương trình.

5000: Tốc độ tối đa mm/s của chương trình.

Ví dụ:

VelSet 50, 800;

MoveL P1, v1000, z0, Tool0;

MoveL P2, v2000, z0, Tool0;

Trong câu lệnh trên, robot sẽ di chuyển đến điểm P1 với vận tốc 1000*50%=500mm/s và di chuyển từ P1 đến P2 với vận tốc 800mm/s (vì 2000*50%=1000>800).

- Lệnh **AccSet**

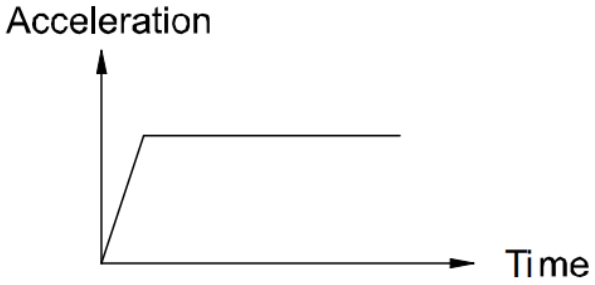
  AccSet 100, 100;

  Trong đó:

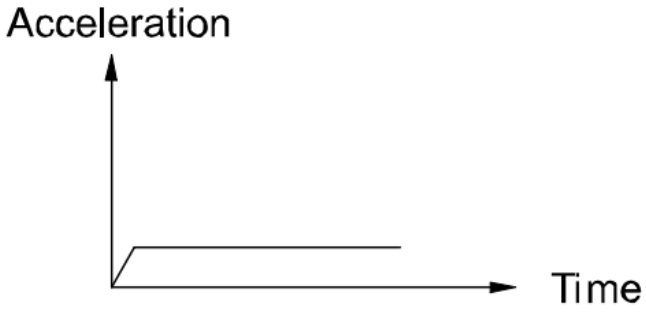
  100: Tăng tốc và giảm tốc theo phần trăm giá trị bình thường. Giá trị tối đa là 100% .

  100: Tốc độ tăng và giảm tốc

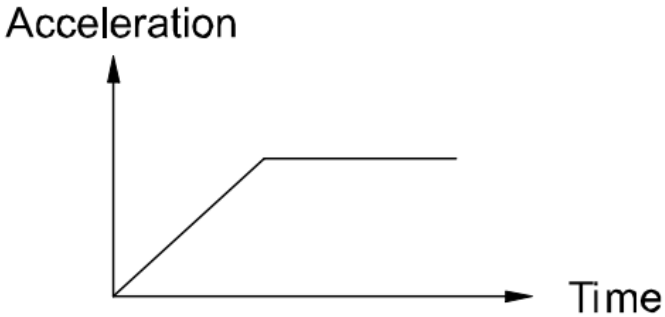
➢ AccSet 100, 100;



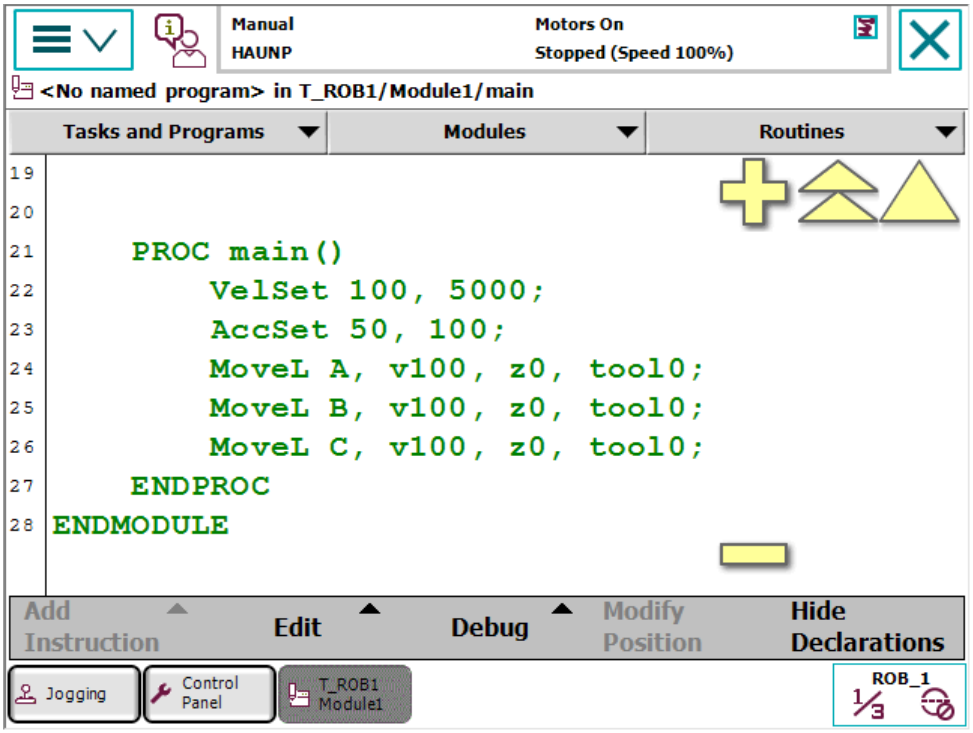
➢ AccSet 30, 100;



➢ AccSet 100, 30;

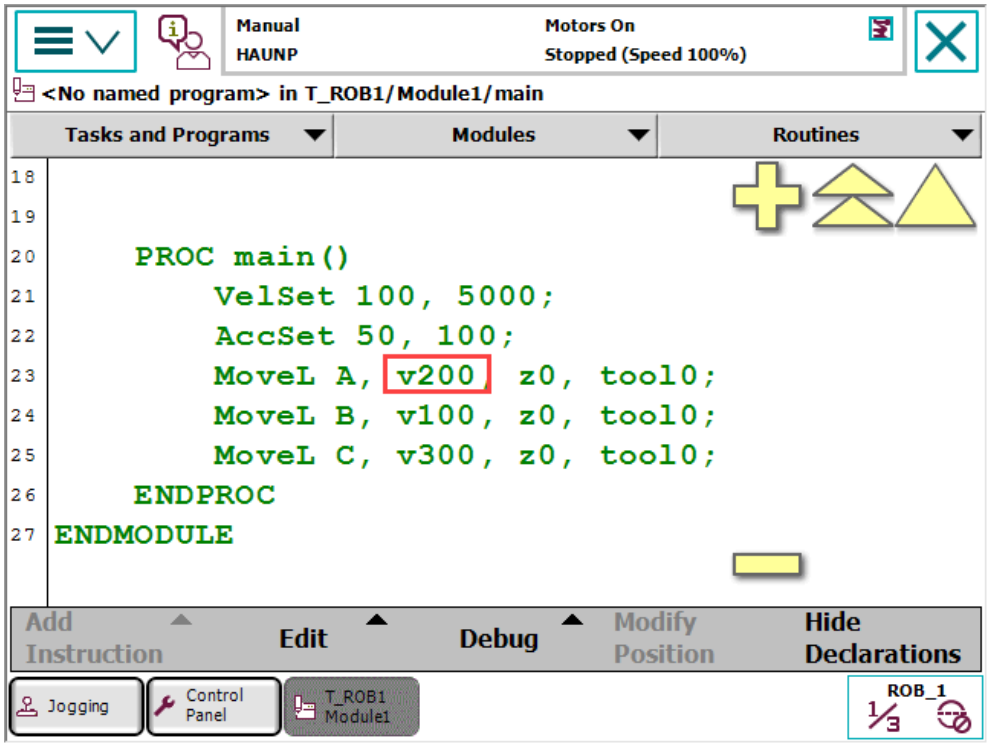


➢ **Chương trình robot**



III. Thay đổi tốc độ di chuyển theo từng điểm trong chương trình

Để thay đổi tốc độ theo từng lệnh Move, chỉ cần thay đổi giá trị vận tốc trong câu lệnh



<u>Chú ý:</u> Tốc độ của từng câu lệnh còn phụ thuộc vào giá trị ***Velset*** như đã đề cập ở ***mục II***

Bài 5: Khai báo I/O, khai báo biến và các lệnh điều kiện

I. Khai báo I/O robot

I.1. Các loại board I/O robot

Các I/O Board dùng trong Rôbốt được ký hiệu là : DSQC -DATA STANDARDS AND QUALITY CONTROL.

Một số các I/O Board thông dụng:

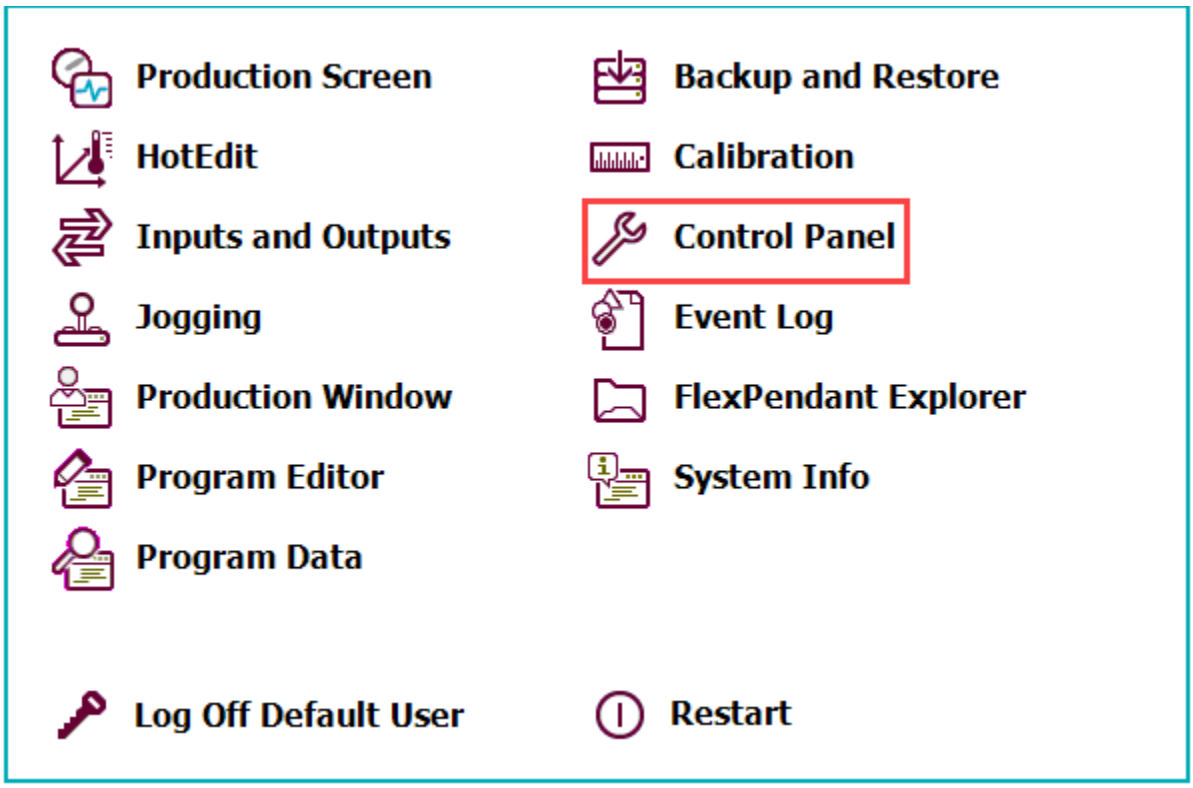
- Digital I/O – DSQC 328A : 24VADC, 16DI, 16DO
- Digital I/O – DSQC 652: 24 VDC, 16DI, 16DO
- Combi I/O – DSQC 327A: 24VADC, 16DI, 16DO, 2AI
- Relay I/O – DSQC 332A: 16DI, 16 Relay Output (thường hở, độc lập lẫn nhau)
- Analog I/O – DSQC 355A: 4AI, 4AO
- Remote I/O – DSQC 350A: số tín hiệu DI/DO có thể lựa chọn khi lập trình (32,64,96,128..)
- DSQC 352A - DeviceNet/PROFIBUS-DP: Điều khiển tín hiệu ra vào giữa hệ DeviceNet và Profibus DP gồm 128DI, 128DO

➢ Board DSQC 625 là board I/O mặc định đi kèm với bộ điều khiển robot. Các board khác là Options mua thêm

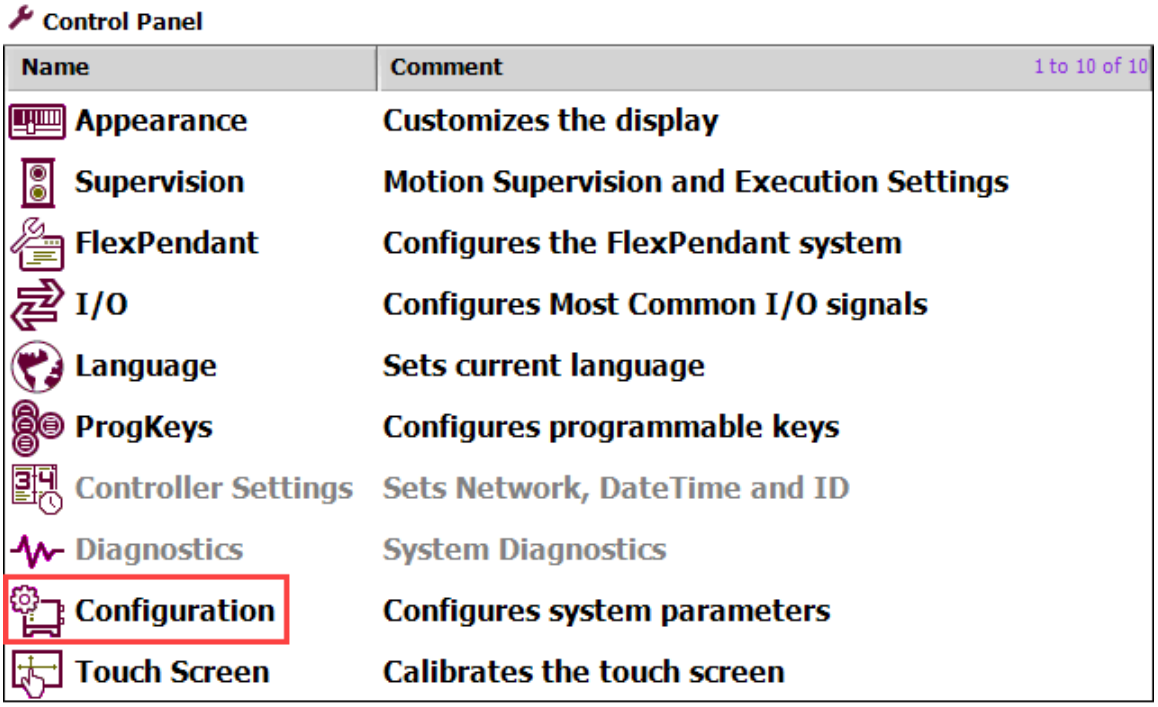
I.2. Cài đặt board I/O robot

❖ Cài đặt board I/O trên tay dạy lập trình

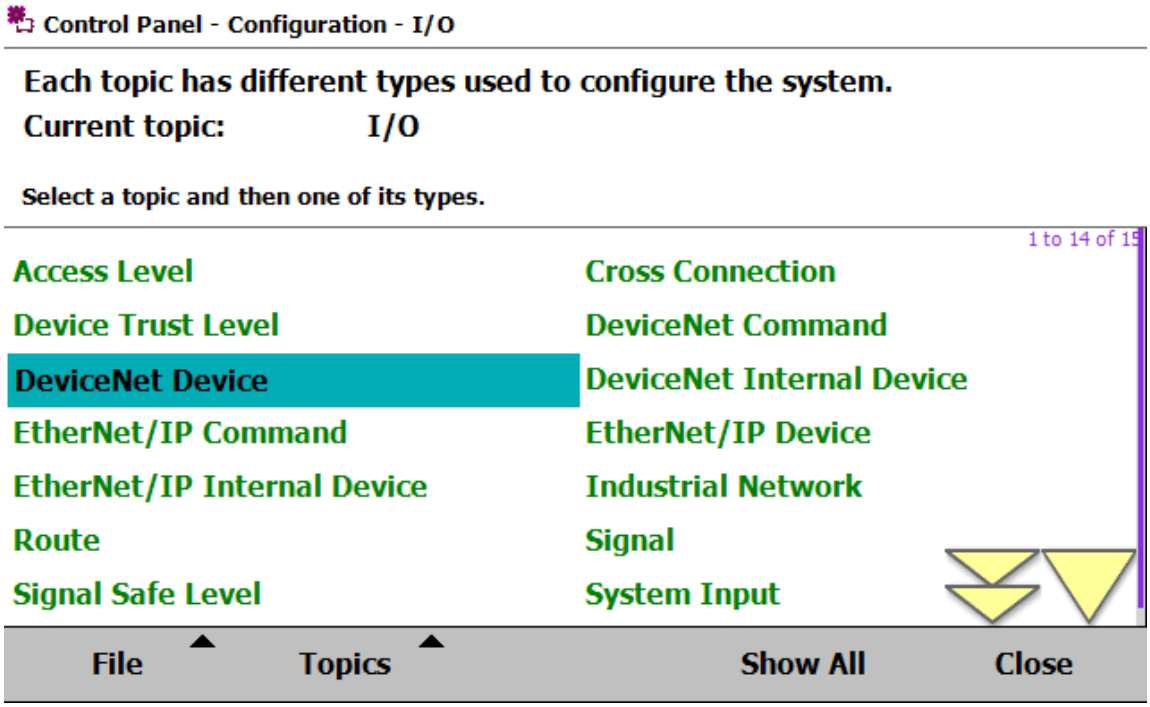
Bước 1: Chọn ***Control Panel***



Bước 2: Chọn ***Configuration***

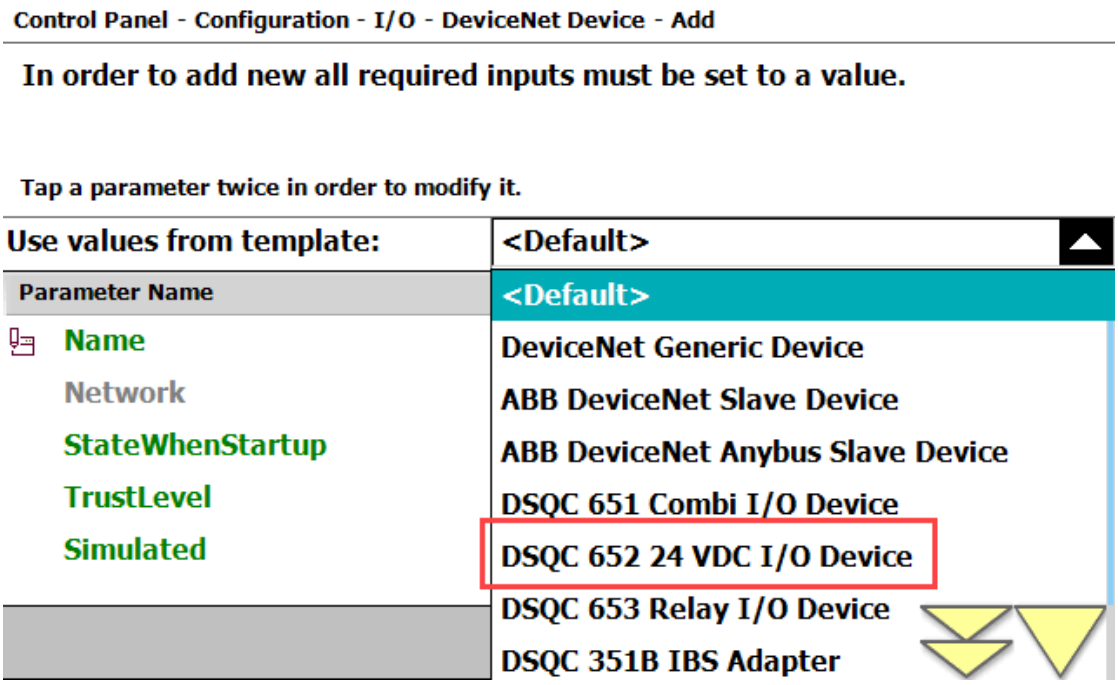


Bước 3: Chọn ***DeviceNet Device***

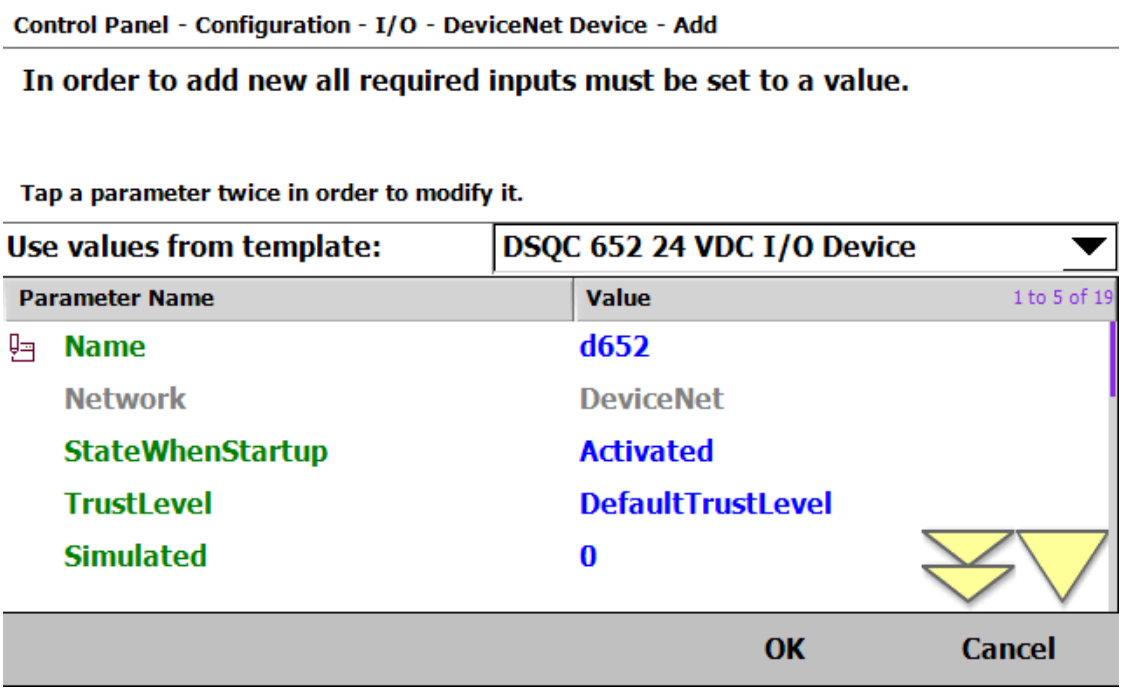


Bước 4: Nhấn ***Add*** để thêm board

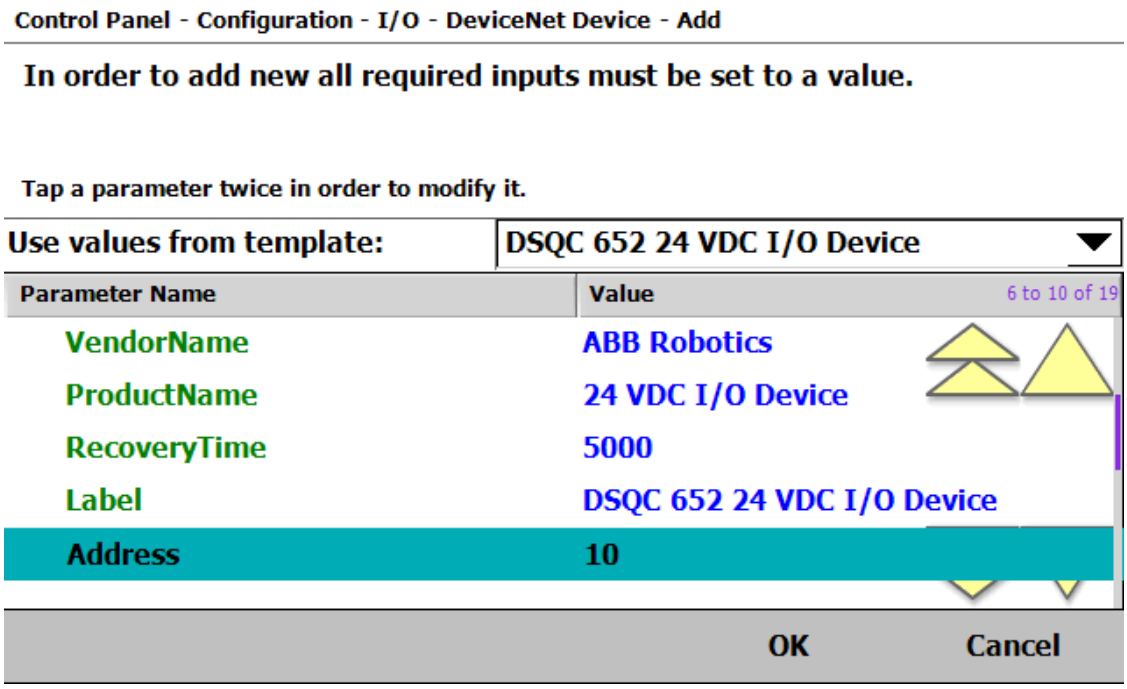
Bước 5: Chọn ***DSQC 652 24 VDC I/O Device*** ở mục ***Use values from template***



Các thuộc tính của board ***DSQC 652*** sẽ tự động điền vào

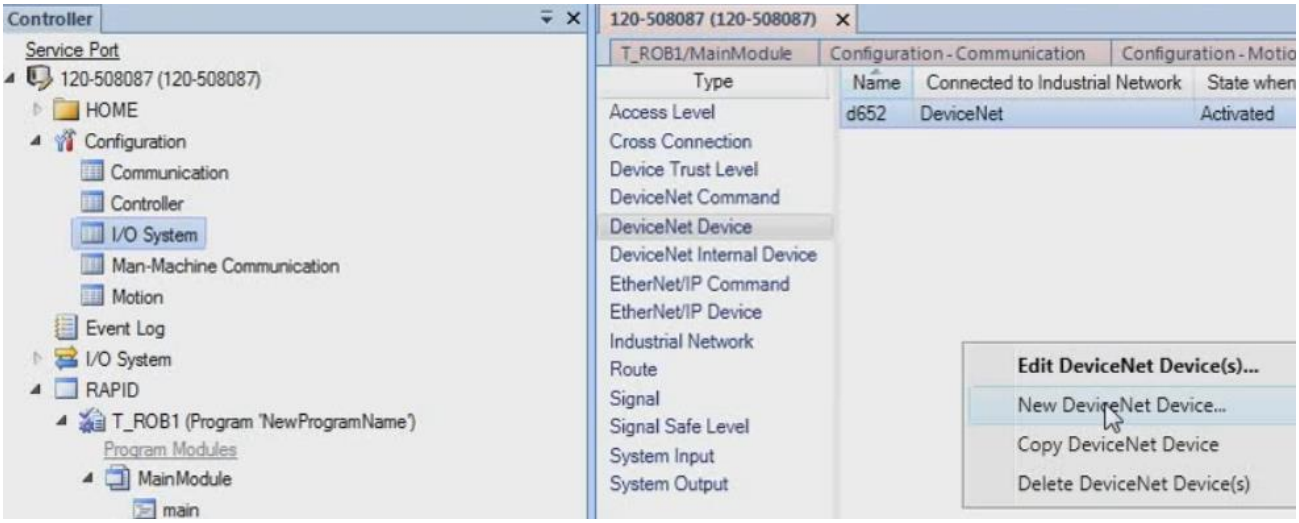


<u>Bước 6:</u> Điền địa chỉ ***Address***. Board ***DSQC D652*** của robot có địa chỉ là **10**



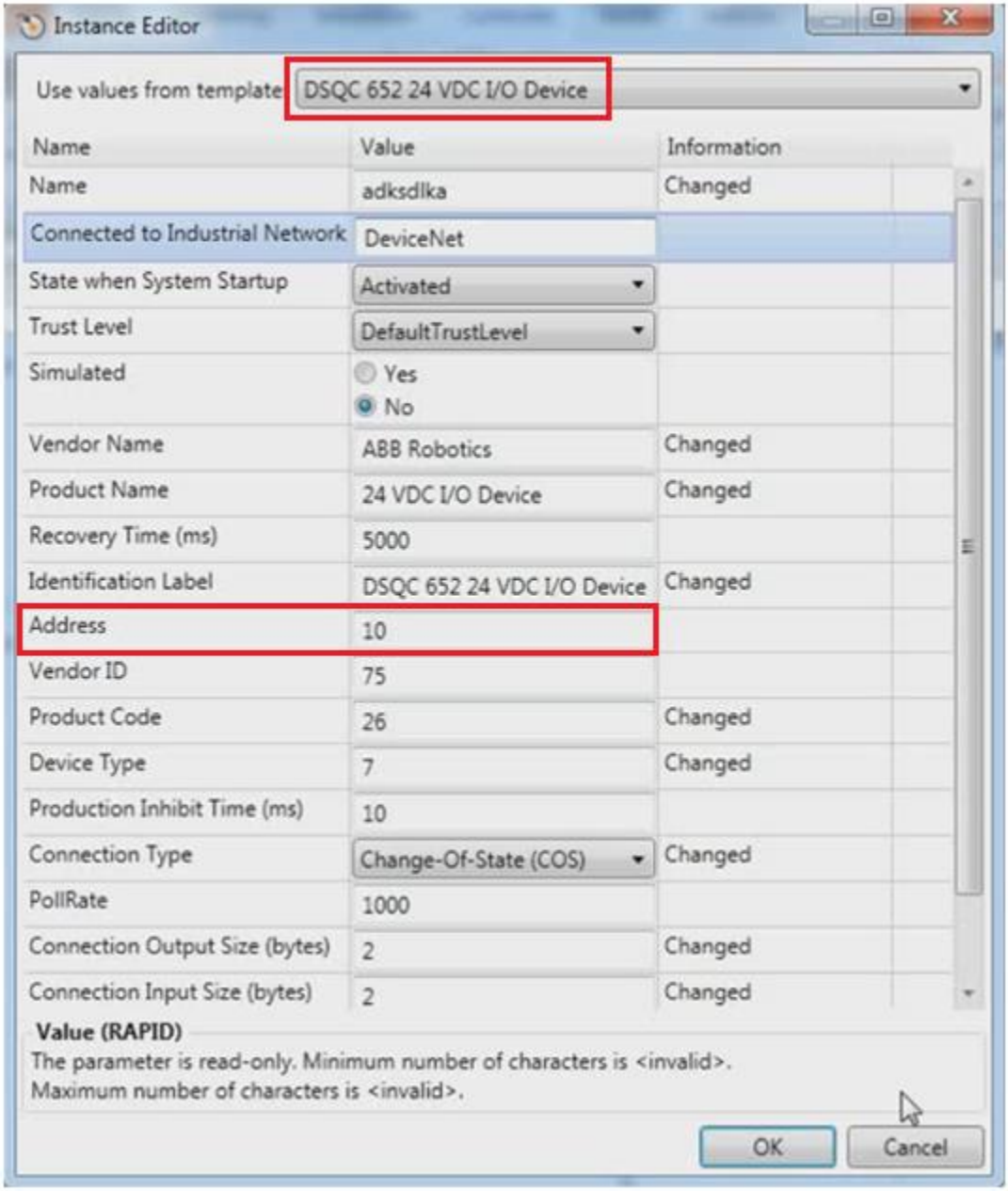
❖ Cài đặt board I/O trên máy tính đã kết nối với robot

Bước 1: Click vào ***I/O System*** và chọn ***DeviceNet Device***



Bước 2: Click chuột phải vào bảng địa chỉ board và chọn ***New DeviceNet Device.*** Sau đó điền các thuộc tín tương tự trên khai báo trên tay dạy lập trình vào bảng và nhấn ***OK.***

Lưu ý: Đối với board ***I/O DSQC 652 24 VDC I/O Device*** thì tại ô ***Address*** điền vào giá trị ***10***



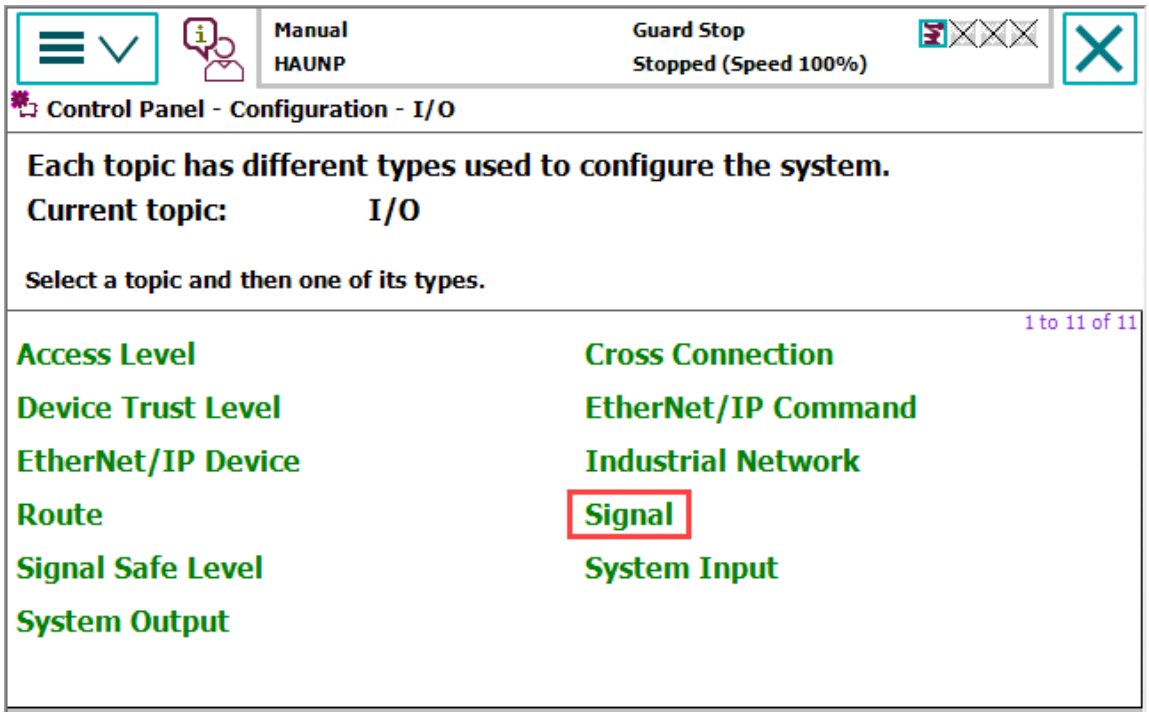
I.3. Cài đặt I/O robot trong chương trình

❖ Tạo I/O trên tay dạy lập trình

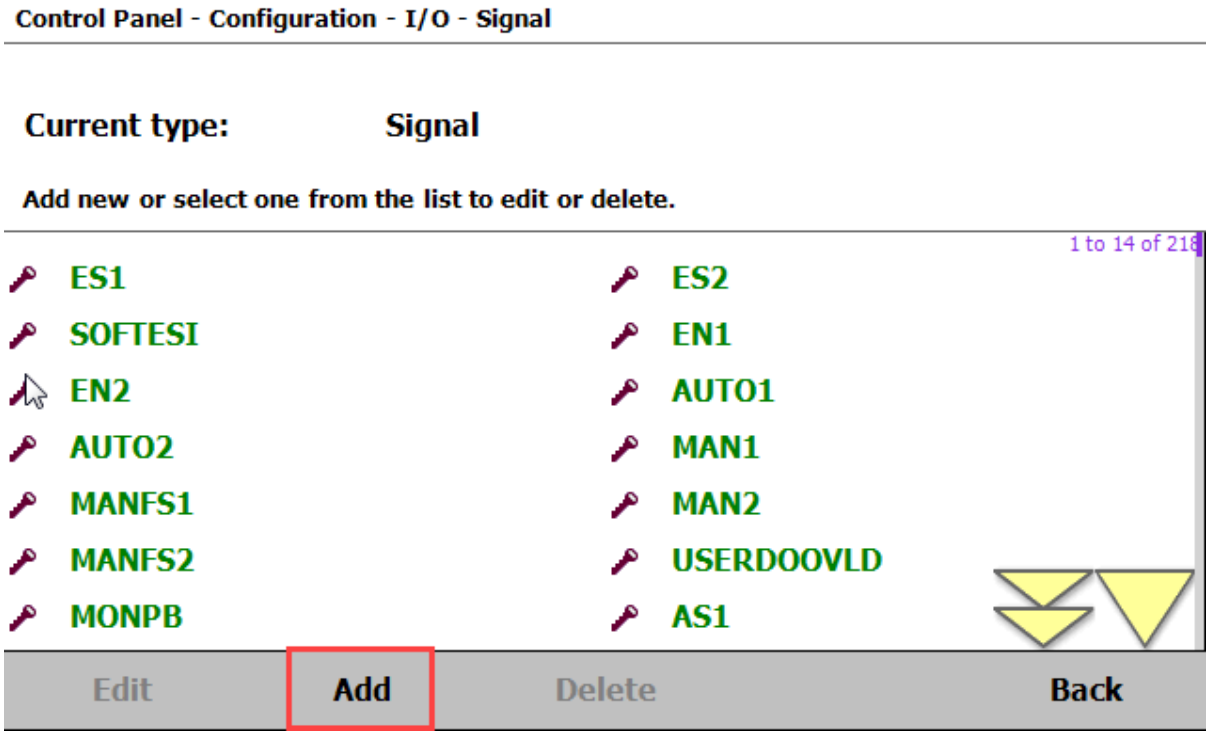
➢ Tạo tín hiệu I/O

<u>Bước 1:</u> Vào ***Control Panel***, chọn ***Configuration***

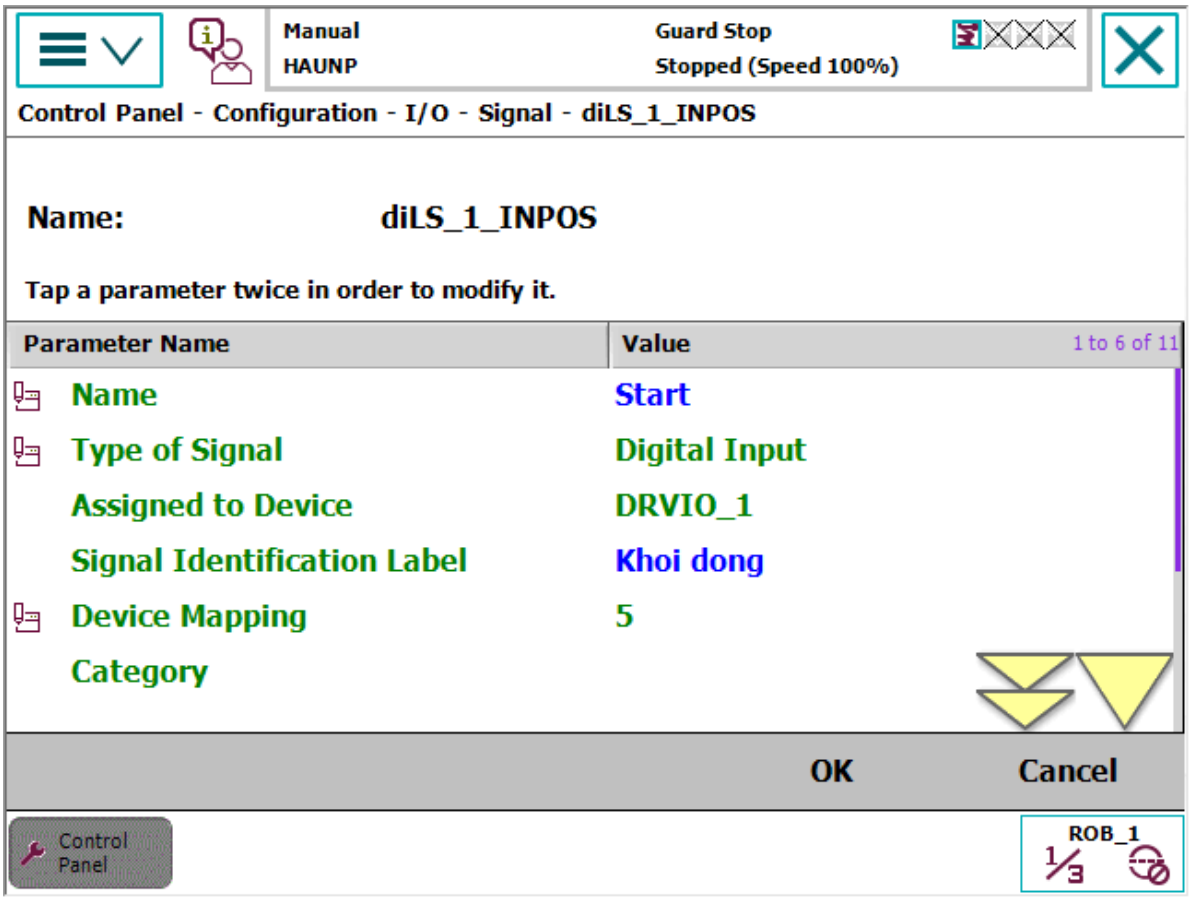
<u>Bước 2:</u> Chọn ***Signal*** để tạo tín hiệu I/O

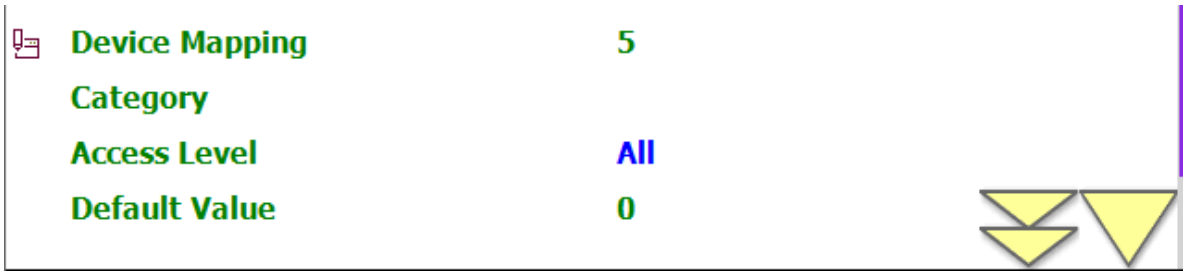


Bước 3: Nhấn ***Add*** để tạo tín hiệu. Lưu ý không được xóa tín hiệu, đặt trùng tên với tín hiệu hệ thống.



Bước 4: Điền các thuộc tính của tín hiệu và nhấn ***OK***, robot sẽ khởi động lại.



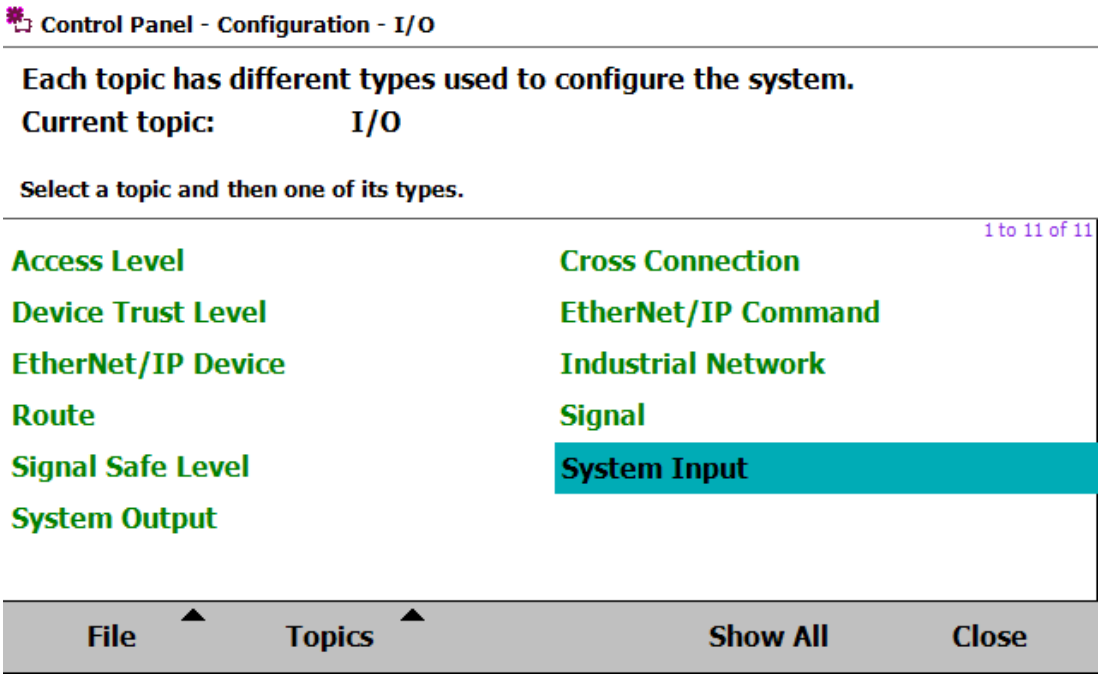


- ***Name:*** Tên tín hiệu
- ***Type of Signal:*** loại tín hiệu
- ***Assigned to Device:*** tên bus I/O
- ***Signal Identification Label:*** giải thích tên tín hiệu (có thể không ghi)
- ***Device Mapping:*** vị trí chân tín hiệu trên bus (0-15,31… tùy thuộc vào loại bus). Lưu ý không được trùng địa chỉ ***Device Mapping***
- ***Access Level***: chọn All

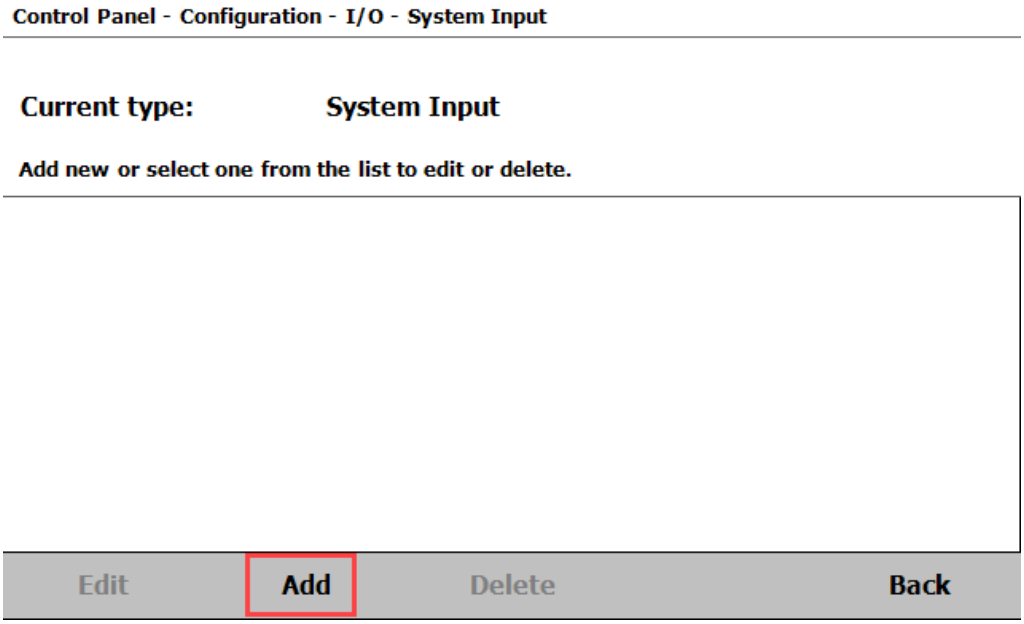
➢ Tạo tín hiệu hệ thống (Start, Stop, Reset)

Bước 1: Vào ***Control Panel***, chọn ***Configuration***

Bước 2: Chọn ***System Input*** để tạo tín hiệu vào hệ thống



Bước 3: Nhấn ***Add*** để thêm tín hiệu



Bước 4: Tạo tín hiệu theo lựa chọn sau và nhấn ***OK***

➢ Tín hiệu Start

Parameter Name	Value	1 to 3 of 3
Signal Name	Start	
Action	Motors On and Start	
Argument 1	Continuous	

➢ Tín hiệu Stop

Parameter Name	Value	1 to 2 of 2
Signal Name	Stop	
Action	Stop	

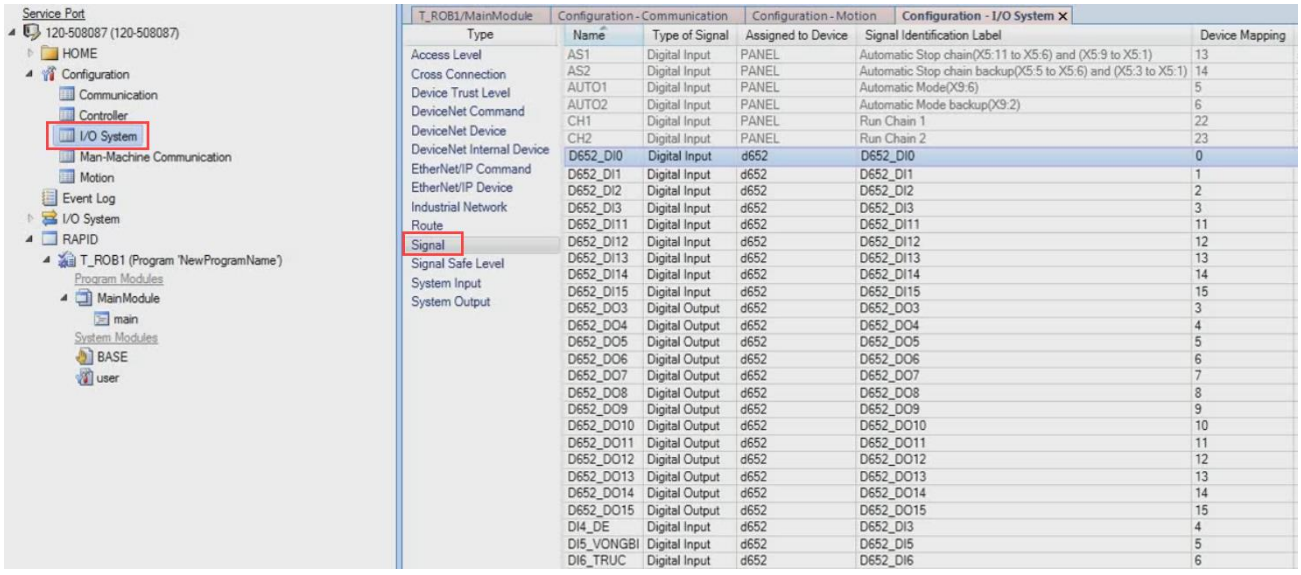
➢ Tin hiệu Reset

Parameter Name	Value	1 to 3 of 3
Signal Name	Reset	
Action	PP to Main	
Task Name	T_ROB1	

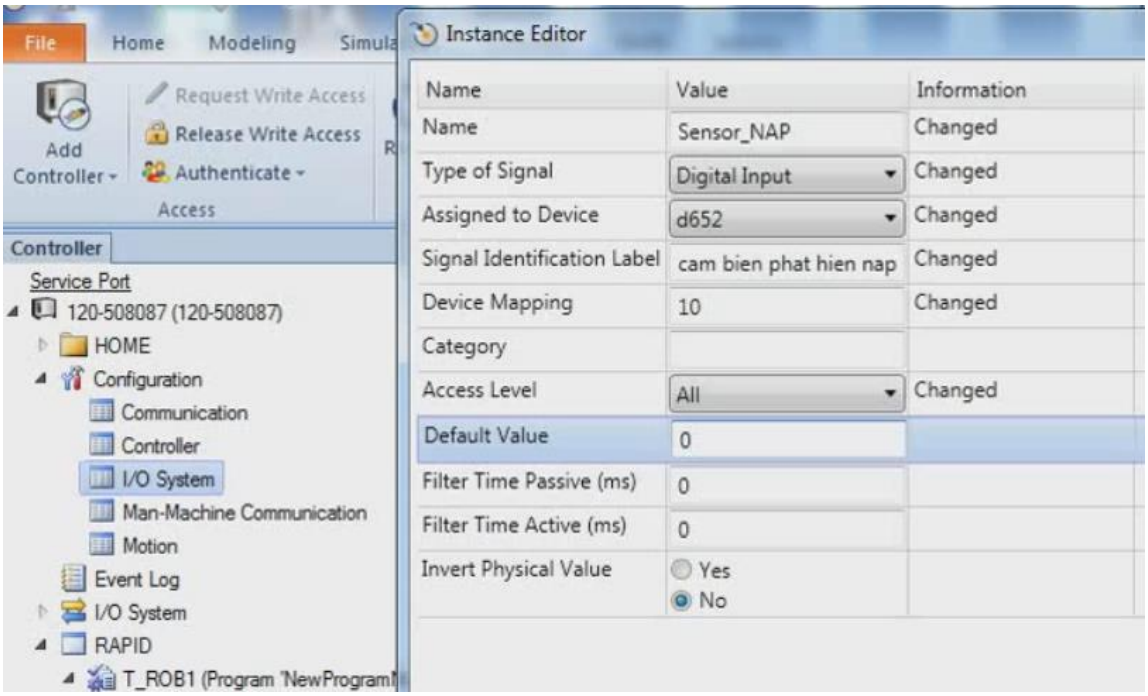
❖ Tạo I/O trên máy tính đã kết nối với robot

➢ Tạo tín hiệu I/O

<u>Bước 1</u>: Click vào ***I/O System*** và chọn ***Signal***

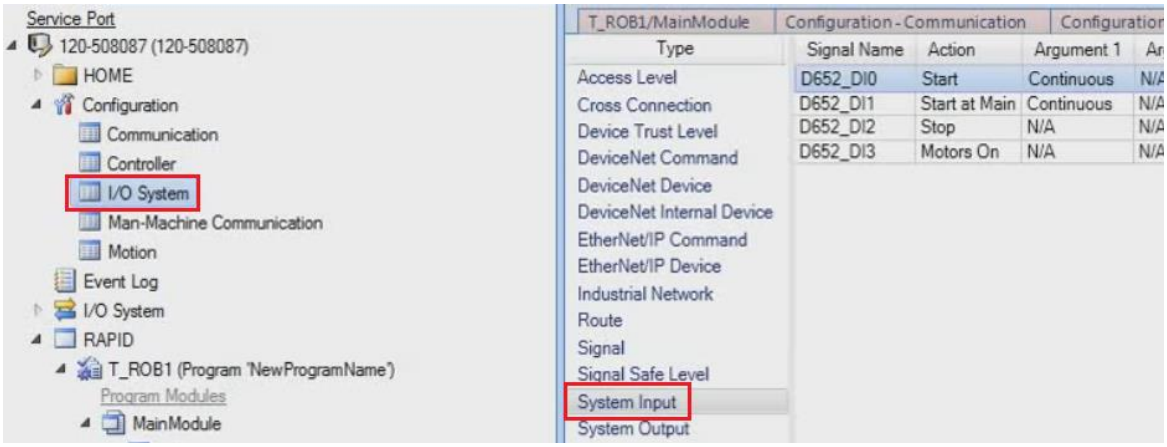


<u>Bước 2:</u> Click chuột phải vào bảng tín hiệu và chọn ***New Signal.*** Sau đó điền các thuộc tín tương tự trên khai báo trên tay dạy lập trình vào bảng và nhấn ***OK.*** Lưu ý không được trùng địa chỉ ***Device Mapping***

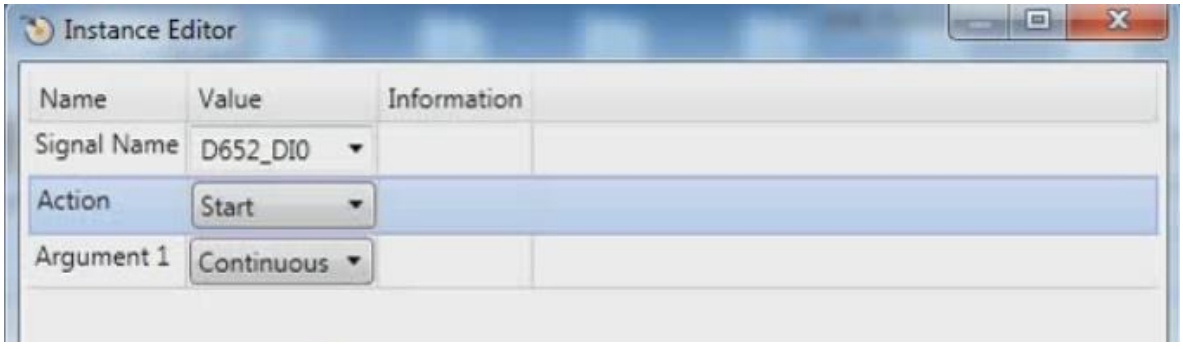


➢ Tạo tín hiệu I/O hệ thống (Start, Stop, Reset)

<u>Bước 1</u>: Click vào ***I/O System*** và chọn ***System Input***

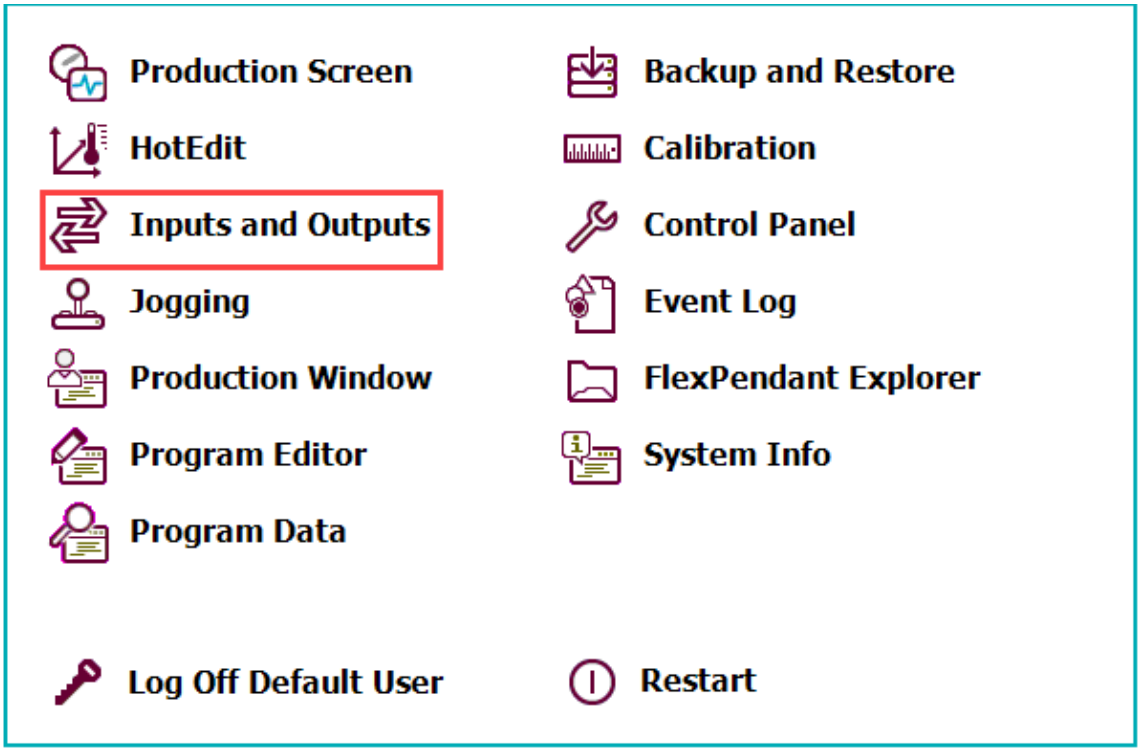


<u>Bước 2:</u> Click chuột phải vào bảng tín hiệu và chọn ***New System Input.*** Sau đó điền các thuộc tín tương tự trên khai báo trên tay dạy lập trình vào bảng và nhấn ***OK.***

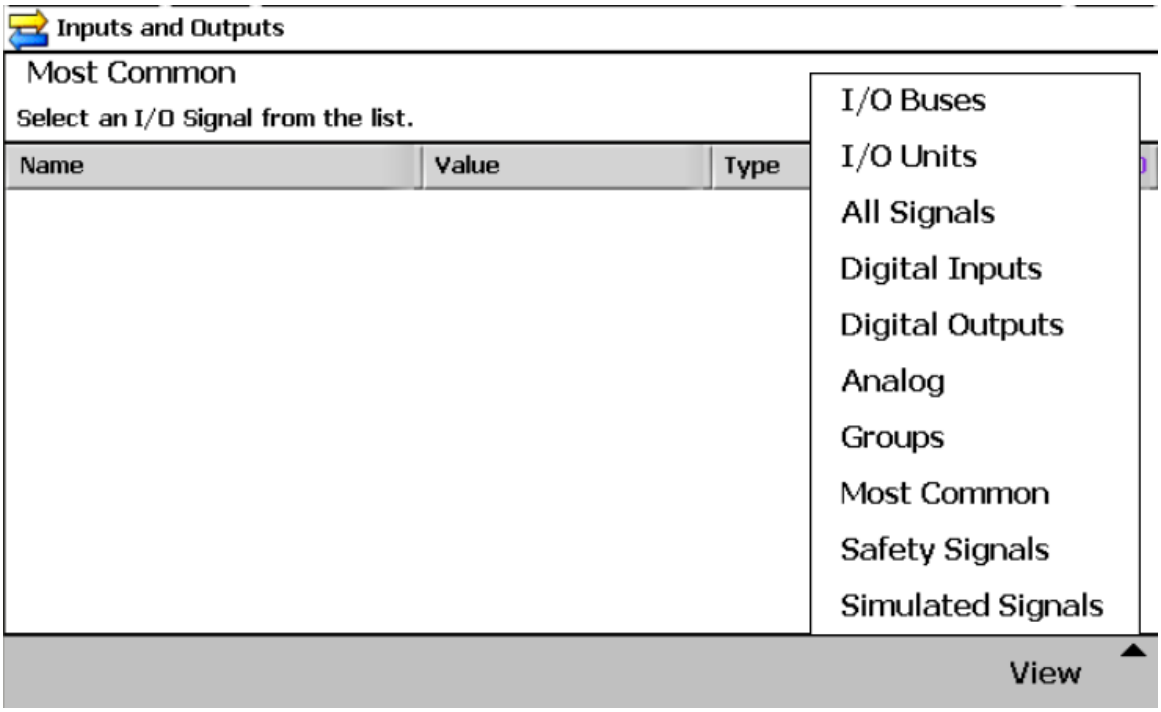


II. Xem tín hiệu I/O

Bước 1: Vào ***Inputs and Outputs***



Bước 2: Nhấn ***View*** và chọn I/O cần xem

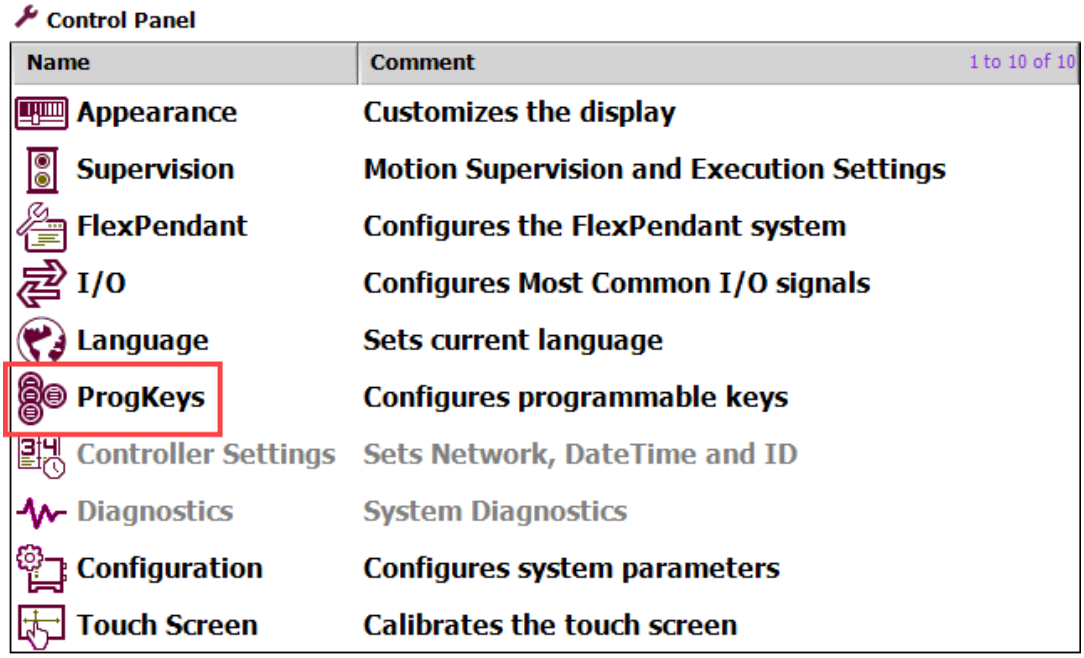


III. Phím tắt I/O trên tay dạy lập trình

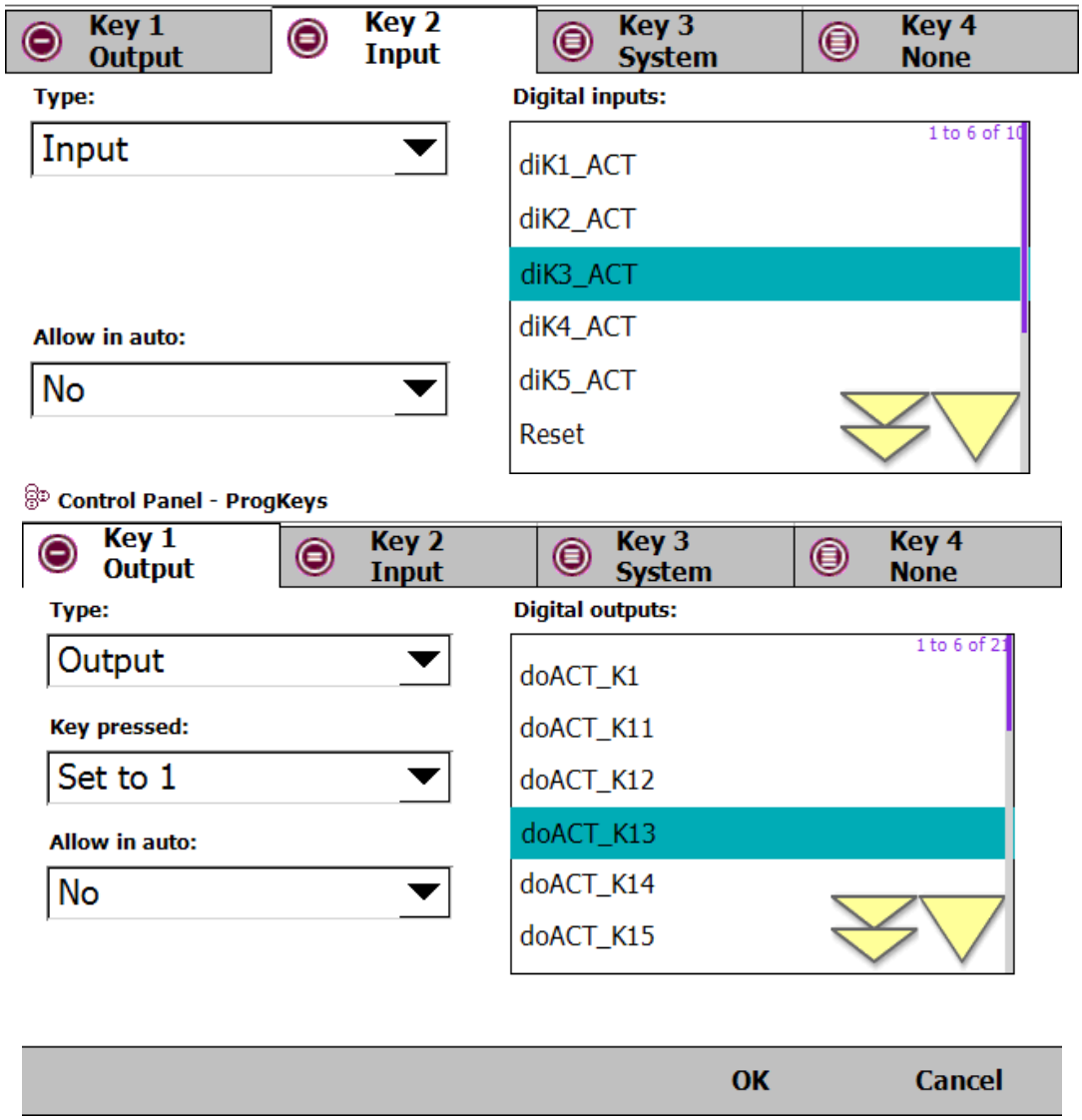
Trên tay dạy lập trình có 4 phím chức năng để tạo tín hiệu tắt cho robot khi lập trình. Thứ tự các phím được chỉ rõ trên tay dạy

Để cài đặt các tín hiệu tắt này cần làm theo các bước sau:

Bước 1: Vào ***Control Panel***, chọn ***ProgKeys***



Bước 2: lựa chọn các tín hiệu cần tạo phím tắt và nhấn ***OK***



IV. Dữ liệu, biến dữ liệu

IV.1. Kiểu dữ liệu

❖ **VAR**

Biến có thể gán giá trị mới trong quá trình chạy chương trình. Và sẽ quay lại giá trị ban đầu khi chạy lại chương trình từ đầu.

Ví dụ :

VAR num mint;

VAR num cherry := 100;

VAR bool TimeOut:=TRUE;

VAR string stBase:=”Welcome”;

❖ **CONST**

Biến có giá trị tĩnh và không thể gán giá trị mới khi chạy chương trình.

Ví dụ :

CONST string str:=”ABB Robotics…”;

CONST robtarget home0p:=[[1962.64, 334.13,-177.17],[0.420452,0.582636, -0.576927, 0.388473],[0,-1,-1,0],[9E+09, 9E+09,9E+09,9E+09,9E+09,9E+09]];

❖ **PERS**

Có thể thay đổi giá trị như biến khi chạy chương trình, tuy nhiên PERS sẽ vẫn lưu lại giá trị mới, cả khi chương trình chạy lại từ đầu

Ví dụ :

PERS bool dirty:=FALSE;

PERS num nCouther:=0;

IV.2. Kiểu biến

- ***Bool:*** giá trị ***TRUE/ FALSE***
- ***Num:*** giá trị số thực
- ***String:*** giá trị chuỗi
- ***Length:*** độ dài chuỗi

IV.3. Khai báo biến

Biến toàn cục: là kết quả của biến có thể dùng cho tất cả các chương trình trong robot (chương trình con, chương trình ngắt…) và phải được khai báo trước chương trình chính

Biến cục bộ: là kết quả của biến chỉ dùng trong một chương trình duy nhất (biến dùng trong chương trình con, biến dùng trong vòng lặp) và được khai báo trong đoạn chương trình đó,

➢ Cách khai báo biến:

Var num x:=0;

PERS num Vit2_x:=0;

PERS string X2_Data:=" ";

VAR bool Camera_data:=TRUE;

V. Các lệnh điều kiện Wait, If-Then, Test-Case, While-Do, For-To

III.1. Lệnh Wait

Chương trình sẽ dừng lại khi gặp lệnh này. Dừng một khoảng thời gian hay đợi một tín hiệu, một hàm,… nếu thỏa điều kiện thì sẽ chương trình sẽ chạy qua lệnh tiếp theo.

- WaitDI: đợi tí hiệu DI

  WaitDI <biến>,<giá trị>;

  VD: WaitDI SS_nap,1;

  WaitDI SS_nap,0;
- WaitDO: đợi tí hiệu DO
- WaitTime: đợi một khoảng thời gian (tính bằng giây)

  VD: WaitTime 0.5;

III.2. Lệnh If-Then

Lệnh kiểm tra điều kiện. Khi gặp lệnh này, chương trình sẽ kiểm tra các điều kiện (tín hiệu, biến, hàm..) nếu thỏa mãn thì sẽ thực hiện các lệnh bên trong nó.

IF <điều kiện đúng>THEN
    < các lệnh>
ELSEIF <điều kiện đúng-trừ các điều kiện đã có>
    <các lệnh>
ELSE (các điều kiện còn lại)
    <các lệnh>
ENDIF

III.3. Lệnh Test-Case

Tương tự như lệnh If-Then với nhiều điều kiện khác nhau. Khi đó có thể thay đổi giá trị Case. Nếu giá trị Case trùng với giá trị biến thì nó sẽ thực hiện chương trình trong Case đó.

TEST <biến>
CASE <giá trị 1 của biến >
    <câu lệnh>
……..
CASE <giá trị n của biến >
    <câu lệnh>
DEFAULT: (các giá trị Case còn lại)
<câu lệnh>
ENDTEST

III.4. Lệnh While-Do

Dùng để lặp lại một đoạn chương trình. Khi điều kiện sai thì sẽ thoát vòng lặp.

WHILE<điều kiện đúng>DO
<câu lệnh>
ENDWHILE

III.5. Lệnh For-To

Dùng để lặp lại một đoạn chương trình. Khi hết số lần lặp lại đã thiết lập thì sẽ thoát vòng lặp.

FOR<biến>FROM<giá trị đầu>TO<giá trị cuối>DO
<câu lệnh>
<biến>:=<biến> +1;
Hoặc <biến>:=<biến> -1;
ENDFOR

Chú ý: Biến trong vòng lặp For-To không cần phải khai báo, đây là biến cục bộ chỉ chạy trong vòng For-To

III.5. Các lệnh khác

- Lệnh ***Start***

Start được dùng để tiếp tục chương trình

Start;

- Lệnh ***Stop***

Stop được dùng để tạm dừng chương trình

Stop;

- Lệnh ***Exit***

Exit được dùng để dừng chương trình

Exit;

- Lệnh **TPWrite**

TPWrite được dùng để ghi dòng chữ lên màn hình tay dạy

TPWrite " string ";

Dòng chữ sẽ được ghi tối đa 80 ký tự

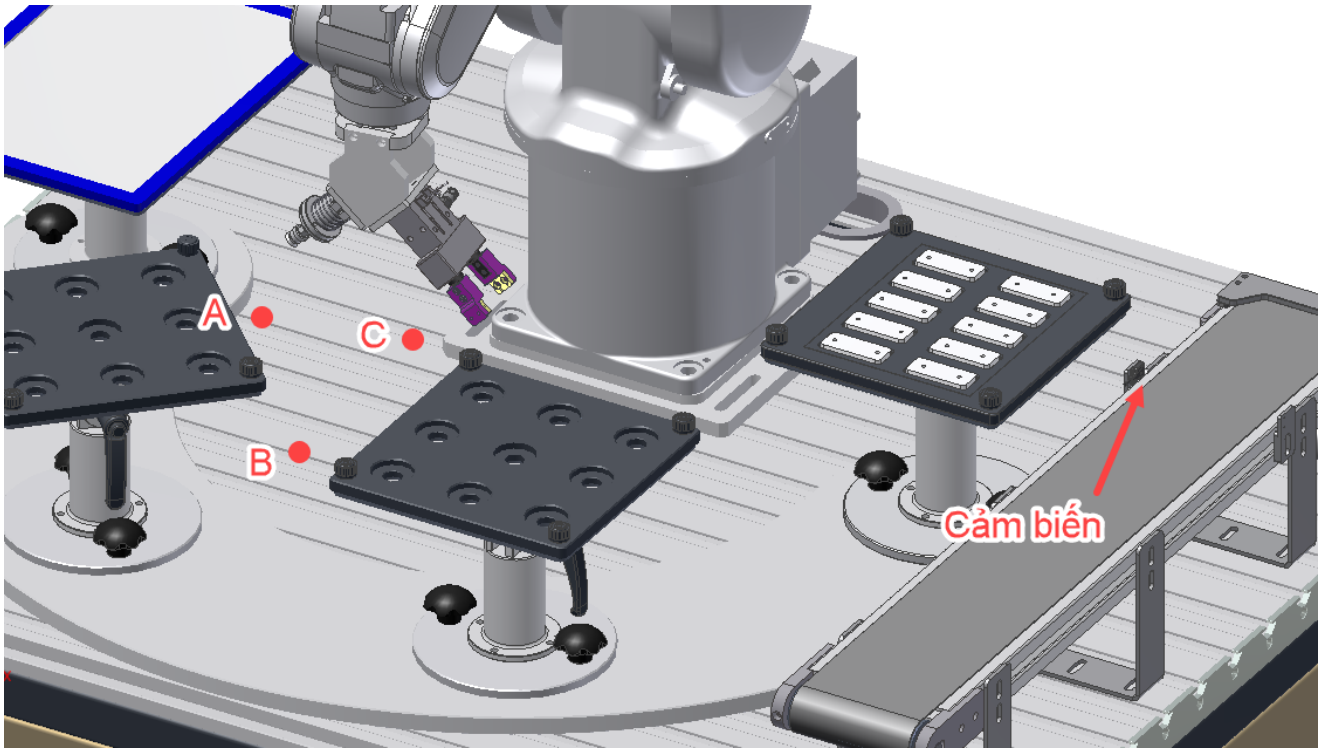
- Lệnh **TPErase**

TPErase được dùng để xóa toàn bộ dòng chữ lên màn hình tay dạy

TPErase;

VI. Viết chương trình di chuyển robot kèm điều kiện (cảm biến, Start, Stop..)

Viết chương trình khi nhấn **Start** robot sẽ di chuyển đi từ điểm A đến điểm B rồi đến điểm C. Robot chỉ lặp lại di chuyển trên khi có tín hiệu **Cảm Biến** mức cao. Khi nhấn **Stop** robot sẽ dừng lại, nhấn **Start** robot sẽ chạy tiếp. Nhấn **Stop** rồi nhấn **Reset** sau đó nhấn **Start** để robot chạy lại chương trình từ đầu.



❖ **Thiết lập khai báo I/O robot:** xem bản vẽ điện trạm robot để xác định đúng tín hiệu cảm biến, nút nhấn tương ứng với địa chỉ Input của robot

Ví dụ:

SS_Tray: tương ứng với cảm biến phát hiện

Các tín hiệu hệ thống ***Start, Stop, Reset*** được khai báo theo hướng dẫn mục I

❖ **Chương trình robot**

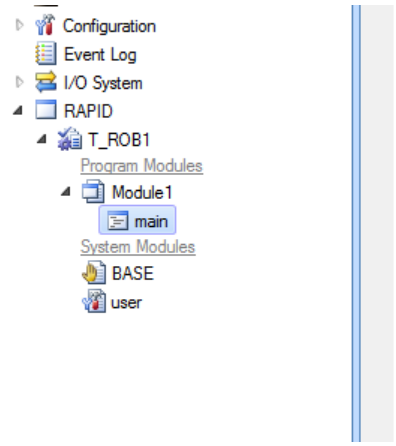
```
21      PROC main()
22          VelSet 100, 5000;
23          AccSet 50, 100;
24          MoveL A, v200, z0, tool0;
25          MoveL B, v100, z0, tool0;
26          MoveL C, v300, z0, tool0;
27          WaitDI SS_Tray, 1;
28      ENDPROC
29  ENDMODULE
```

❖ **Giải thích chương trình:**

Sau khi nhấn nút **Start** trên panel, robot sẽ di chuyển về điểm **Home** và chạy đến điểm A, B, C sau đó dừng lại đợi tín hiệu **SS_Tray**. Khi cảm biến **SS_Tray** lên mức **High (1)** thì robot qua lại điểm A rồi di chuyển đến B, C và dừng sau đó lại kiểm tra tín hiệu **SS_Tray**, nếu mức **High** thì tiếp tục chu trình lên, còn mức **Low (0)** thì sẽ dừng lại đợi tín hiệu lên mức **High**.

Nhấn **Stop** robot sẽ dừng lại ngay lập tức. Muốn robot chạy tiếp tục thì nhấn **Start**. Nếu muốn robot chạy lại từ đầu sau khi nhấn **Stop** thì nhấn **Reset** và nhấn lại **Start**, robot sẽ chạy lại chương trình bắt đầu việc trở về điểm **A**

❖ **Chương trình robot dùng lệnh While**




```
3       CONST robtarget p10:=[[160.00,-80.00,615.00],[
4       TASK PERS tooldata tool1:=[TRUE,[[0,0,0],[1,0,
5       TASK PERS wobjdata wobj1:=[FALSE,TRUE,"",[[0,0
6       CONST robtarget A:=[[160.00,-80.00,615.00],[0.
7       CONST robtarget B:=[[180.00,-80.00,615.00],[0.
8       CONST robtarget C:=[[160.00,80.00,615.00],[0.7
9
10      PROC main()
11          VelSet 100, 5000;
12          AccSet 50, 100;
13          MoveL A, v200, z0, tool0;
14          MoveL B, v100, z0, tool0;
15          MoveL C, v300, z0, tool0;
16          WHILE SS_Tray<>1 DO
17              WaitTime 0.2;
18          ENDWHILE
19      ENDPROC
20  ENDMODULE
```

<u>Chú ý:</u> Để thuận tiện cho việc lập trình, khuyến nghị nên lập trình trên máy tính sau đó đổ chương trình xuống robot. Trước khi viết chương trình trên máy tính cần yêu cầu truy cập vào robot.

Bài 6: Kết hợp tín hiệu I/O trong di chuyển robot

I. Lệnh Set

Để xuất một tín hiệu ra cổng Output của robot hoặc đặt giá trị biến trong chương trình lên mức 1. Tín hiệu Output này có thể là tín hiệu số, hàm, biến,…

Set <biến>; (mặc định biến này được đưa lên mức 1)

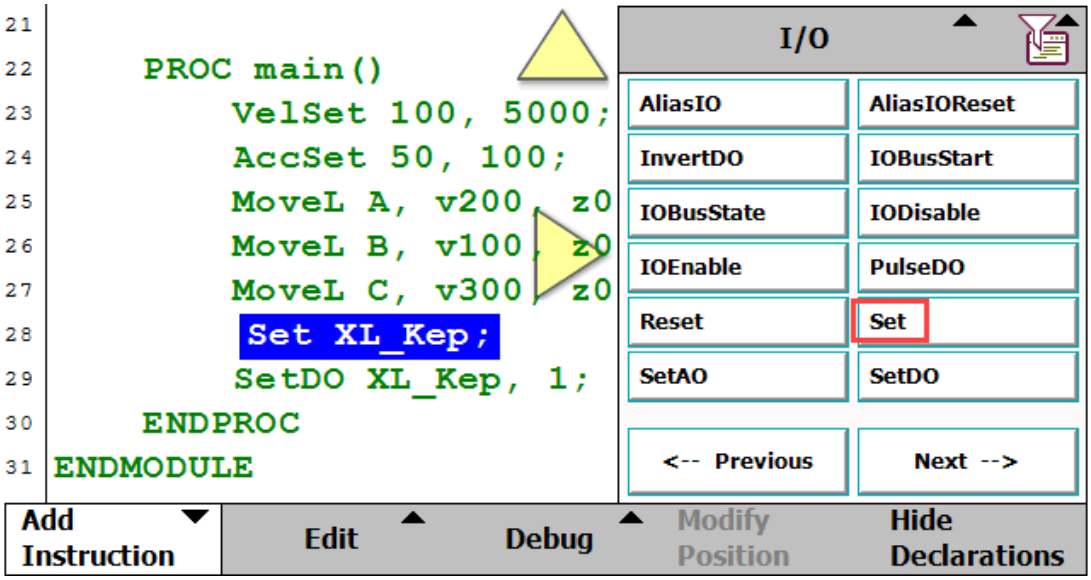
SetDO <Biến>,<Giá trị>;

Trong đó:

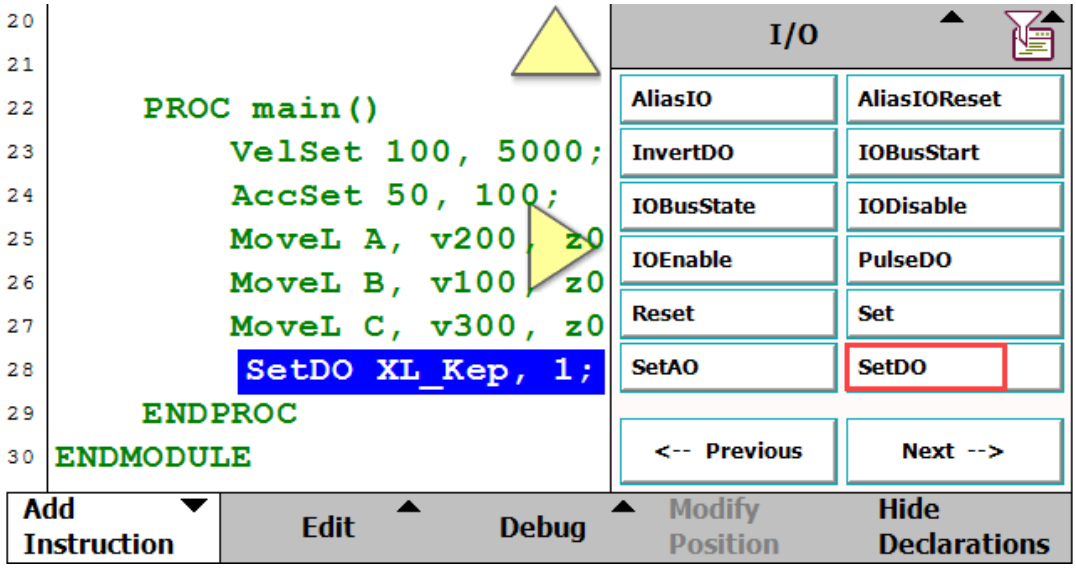
Biến: Có thể là tín hiệu output hoặc biến chương trình

Giá trị: 0 hoặc 1, có thể chọn High hoặc Low

Vào ***Add Instrucion*** chọn ***Set***



Vào ***Add Instrucion*** chọn ***SetDO***



Chú ý: Đối với trường hợp điều khiển xi lanh. Có 2 loại điều khiển tùy vào loại van điện từ điều khiển xi lanh.

- Van điện từ tác động đơn (phản hồi bằng lo xò):

  Set Xilanh_Kep; xi lanh kẹp

  SetDO Xilanh_Kep, 0; xi lanh nhả
- Van điện từ tác động kép (tác động 2 đầu):

  Set Xilanh_Kep; xi lanh kẹp

  SetDO Xilanh_Kep, 0; tắt tín hiệu xi lanh kẹp trên van điện từ

  SetDO Xilanh_Nha, 1; xi lanh nhả

  SetDO Xilanh_Nha, 0; tắt tín hiệu xi lanh nhả trên van điện từ

❖ **Không thể điều khiển van điện từ nếu tác động điện cả 2 đầu van**

II. Lệnh Reset

Để xuất một tín hiệu ra cổng Output của robot hoặc đặt giá trị biến trong chương trình xuống mức 0. Tín hiệu Output này có thể là tín hiệu số, hàm, biến,…

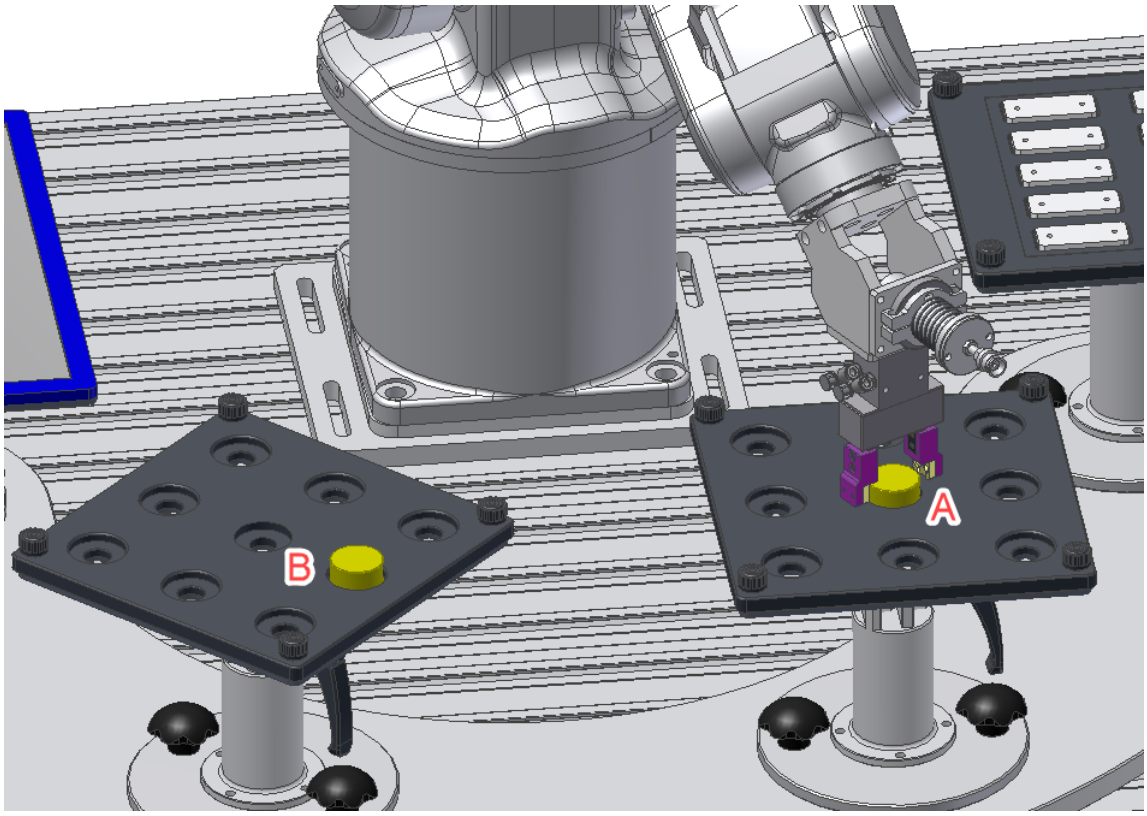
Lệnh ***Reset*** có thể thay thế cho lệnh SetDO xuống mức 0 của tín hiệu

Reset <biến>;

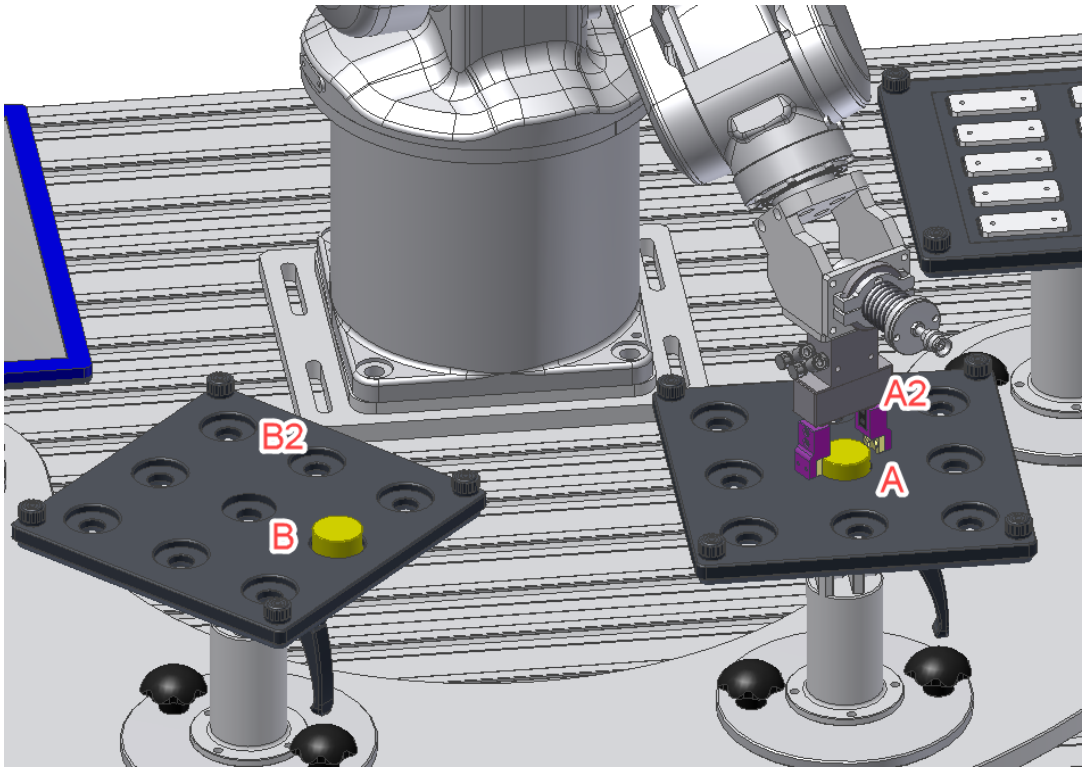
SetDO Xilanh_Nha, 0; hoặc Reset Xilanh_Nha;

III. Viết chương trình robot kẹp vật từ A và thả vào vị trí B

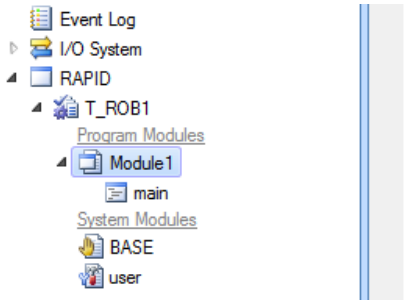
➢ **Yêu cầu viết chương trình cho robot sẽ kẹp vật từ vị trí A bỏ sang vị trí B và lặp lại liên tục**



❖ Trong chương trình này, cần thêm điểm A2, B2 để robot di chuyển không bị va chạm.
Điểm A2: điểm phía trên điểm A theo trục Z
Điểm B2: điểm phía trên điểm B theo trục Z



➢ **Chương trình robot**



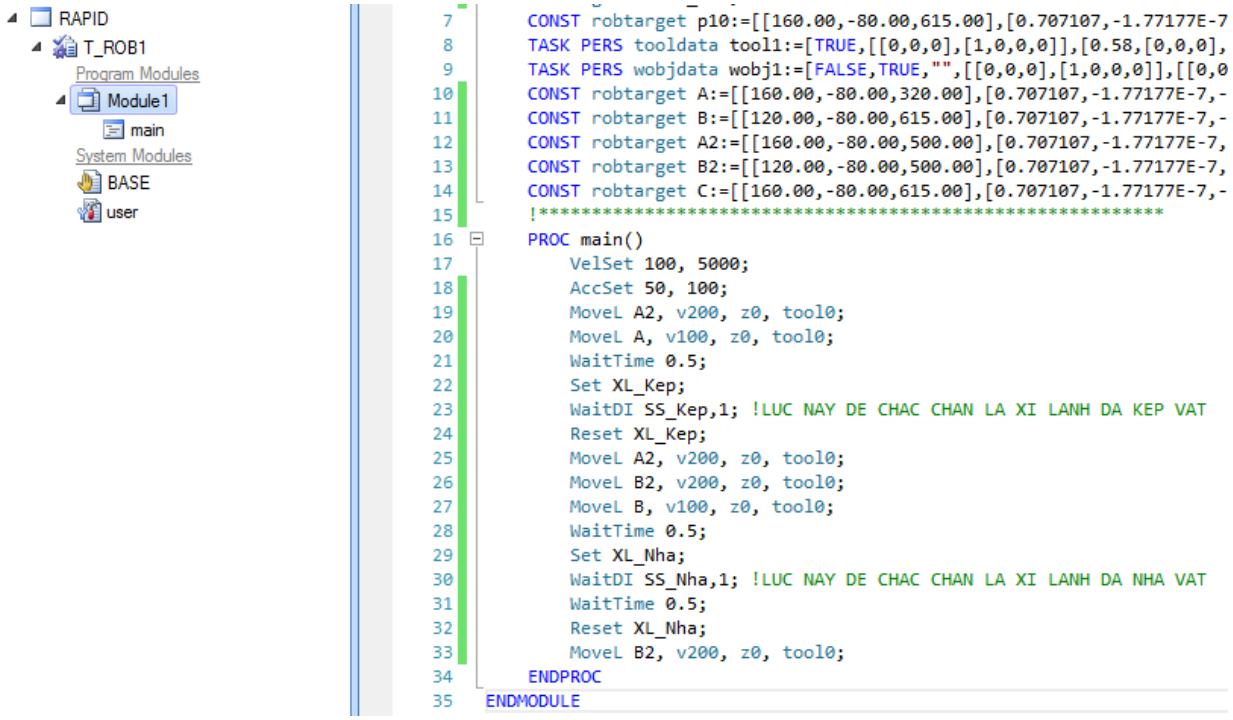

```
    CONST robtarget p10:=[[160.00,-80.00,615.00],[0.707107,-1.77
    TASK PERS tooldata tool1:=[TRUE,[[0,0,0],[1,0,0,0]],[0.58,[0
    TASK PERS wobjdata wobj1:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]]
    CONST robtarget A:=[[160.00,-80.00,320.00],[0.707107,-1.7717
    CONST robtarget B:=[[120.00,-80.00,615.00],[0.707107,-1.7717
    CONST robtarget A2:=[[160.00,-80.00,500.00],[0.707107,-1.771
    CONST robtarget B2:=[[120.00,-80.00,500.00],[0.707107,-1.771
    CONST robtarget C:=[[160.00,-80.00,615.00],[0.707107,-1.7717
    !*********************************************************
    PROC main()
        VelSet 100, 5000;
        AccSet 50, 100;
        MoveL A2, v200, z0, tool0;
        MoveL A, v100, z0, tool0;
        WaitTime 0.5;
        Set XL_Kep;
        WaitTime 0.5;
        Reset XL_Kep;
        MoveL A2, v200, z0, tool0;
        MoveL B2, v200, z0, tool0;
        MoveL B, v100, z0, tool0;
        WaitTime 0.5;
        Set XL_Nha;
        WaitTime 0.5;
        Reset XL_Nha;
        MoveL B2, v200, z0, tool0;
    ENDPROC
ENDMODULE
```

➢ **Giải thích chương trình**

Khi chạy chương trình, robot di chuyển đến vị trí **A2** và đến **A**. robot dừng lại **0.5s** để ổn định vị trí rồi tiến hành kẹp vật. Sau khi kẹp **0.5s** robot di chuyển lên lại điểm **A2** để tránh va chạm rồi di chuyển đến **B2** và **B**. sau **0.5s** robot thả vật xuống và di chuyển lại về vị trí **B2** để tránh va chạm rồi qua về vị trí **A2** tiếp tục lặp lại chu trình

❖ Có thể thay thế các lệnh **Wait 0.5s** đợi xi lanh kẹp, nhả bằng cách đợi tí hiệu cảm biến trên thân xi lanh.

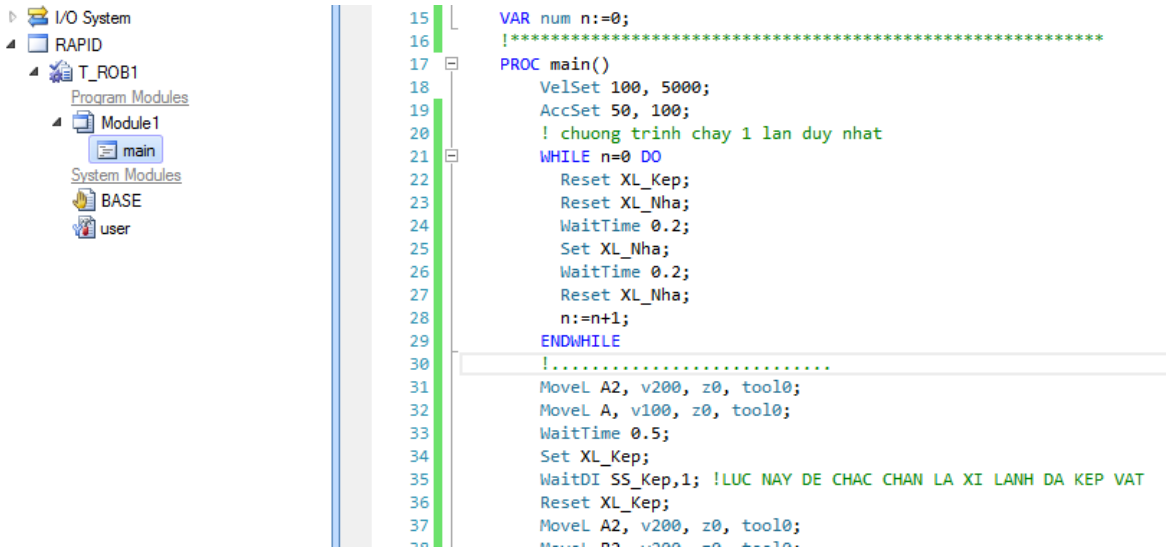
❖ Kiểm tra cảm biến trên thân xi lanh là yêu cầu bắt buộc trong công nghiệp. vì hệ thống chạy cần biết chính xác xi lanh đang ở trạng thái nào để thực hiện các lệnh tiếp theo. Lệnh **Wait** sẽ làm robot chạy sai chương trình do ảnh hưởng của van tiết lưu trên xi lanh.




```
7     CONST robtarget p10:=[[160.00,-80.00,615.00],[0.707107,-1.77177E-7
8     TASK PERS tooldata tool1:=[TRUE,[[0,0,0],[1,0,0,0]],[0.58,[0,0,0],
9     TASK PERS wobjdata wobj1:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]],[[0,0
10    CONST robtarget A:=[[160.00,-80.00,320.00],[0.707107,-1.77177E-7,-
11    CONST robtarget B:=[[120.00,-80.00,615.00],[0.707107,-1.77177E-7,-
12    CONST robtarget A2:=[[160.00,-80.00,500.00],[0.707107,-1.77177E-7,
13    CONST robtarget B2:=[[120.00,-80.00,500.00],[0.707107,-1.77177E-7,
14    CONST robtarget C:=[[160.00,-80.00,615.00],[0.707107,-1.77177E-7,-
15    !*********************************************************
16    PROC main()
17        VelSet 100, 5000;
18        AccSet 50, 100;
19        MoveL A2, v200, z0, tool0;
20        MoveL A, v100, z0, tool0;
21        WaitTime 0.5;
22        Set XL_Kep;
23        WaitDI SS_Kep,1; !LUC NAY DE CHAC CHAN LA XI LANH DA KEP VAT
24        Reset XL_Kep;
25        MoveL A2, v200, z0, tool0;
26        MoveL B2, v200, z0, tool0;
27        MoveL B, v100, z0, tool0;
28        WaitTime 0.5;
29        Set XL_Nha;
30        WaitDI SS_Nha,1; !LUC NAY DE CHAC CHAN LA XI LANH DA NHA VAT
31        WaitTime 0.5;
32        Reset XL_Nha;
33        MoveL B2, v200, z0, tool0;
34    ENDPROC
35  ENDMODULE
```

Đối với chương trình này, nếu không có khí cấp cho xi lanh thì hệ thống sẽ dừng lại vì tín hiệu cảm biến trên thân xi lanh khi kẹp và nhả không có. Vì vậy sẽ biết được lỗi hệ thống cấp khí.

<u>Chú ý:</u> Cần viết chương trình reset tín hiệu Output, Biến dữ liệu trước khi robot chạy chương trình chính nhằm tránh lỗi chương trình




```
VAR num n:=0;
!***********************************************************
PROC main()
    VelSet 100, 5000;
    AccSet 50, 100;
    ! chuong trinh chay 1 lan duy nhat
    WHILE n=0 DO
      Reset XL_Kep;
      Reset XL_Nha;
      WaitTime 0.2;
      Set XL_Nha;
      WaitTime 0.2;
      Reset XL_Nha;
      n:=n+1;
    ENDWHILE
    !.........................
    MoveL A2, v200, z0, tool0;
    MoveL A, v100, z0, tool0;
    WaitTime 0.5;
    Set XL_Kep;
    WaitDI SS_Kep,1; !LUC NAY DE CHAC CHAN LA XI LANH DA KEP VAT
    Reset XL_Kep;
    MoveL A2, v200, z0, tool0;
```

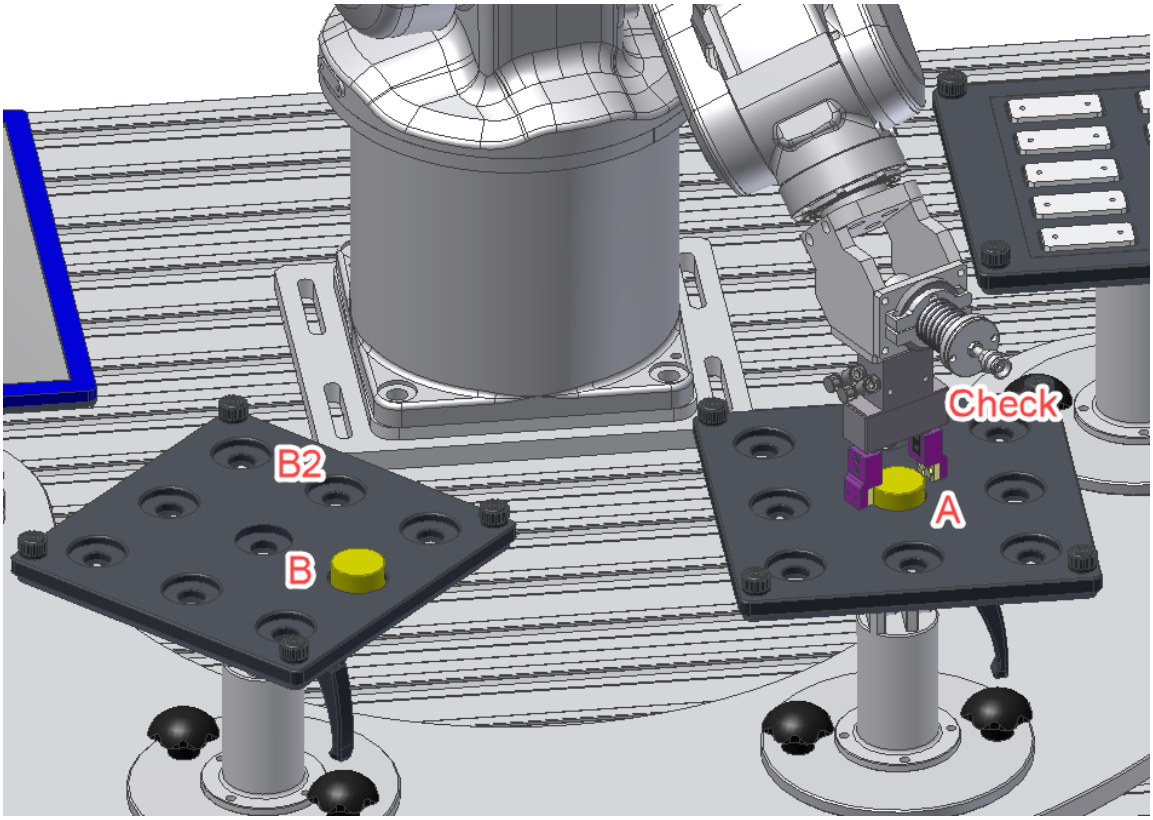
➢ **Giải thích chương trình**

Với đoạn chương trình này, có thể reset xi lanh về trạng thái đang nhả và tắt hết các tín hiệu điều khiển trên van điện từ. Do đó khi vào chương trình robot sẽ biết được trạng thái ban đầu của xi lanh

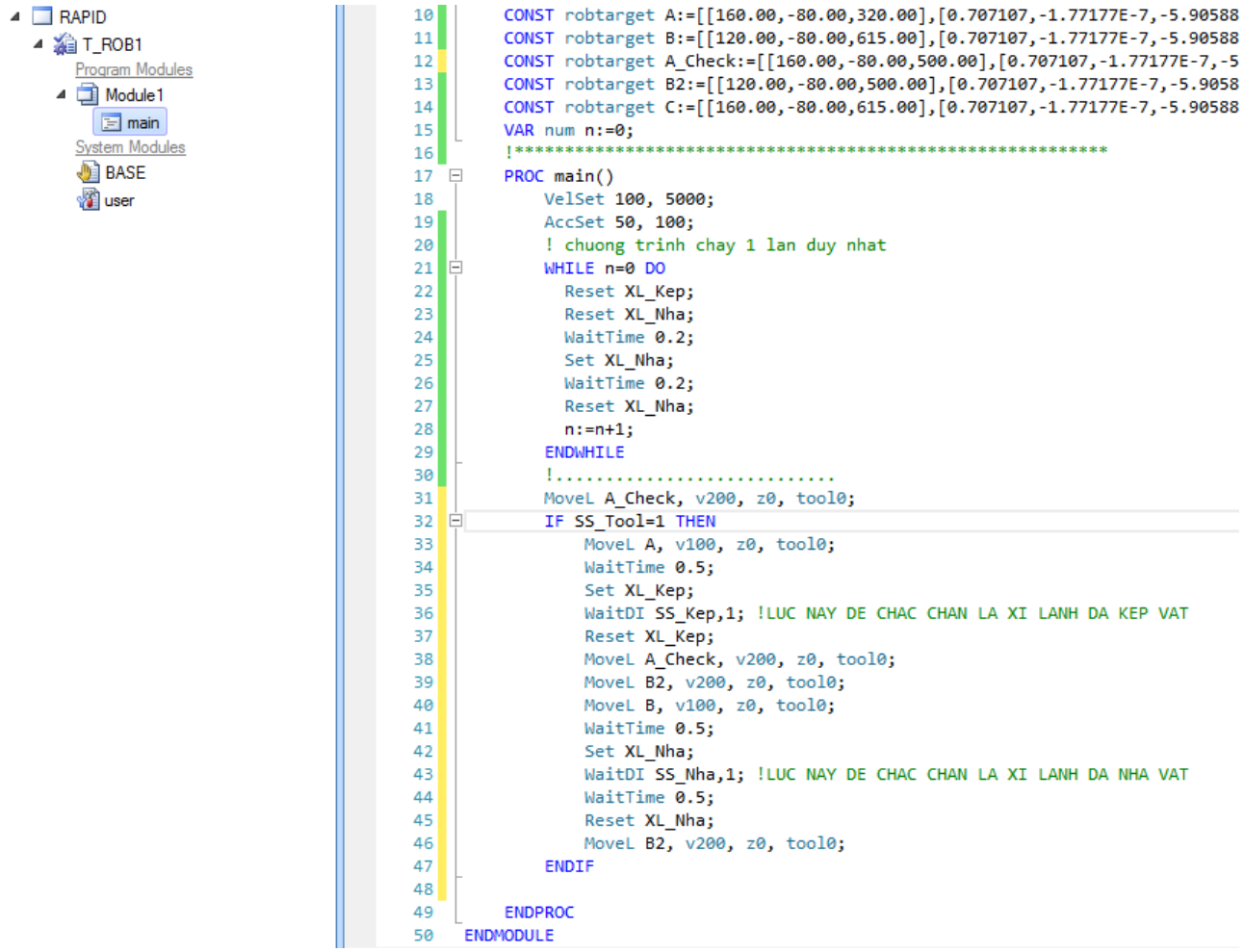
⇨ **Đối với chương trình mục II, robot sẽ di chuyển kẹp vật từ A bỏ sang B liên tục không cần biết ở vị trí A có vật hay không. Chương trình ở mục tiếp theo sẽ khắc phục lỗi trên**

IV. Viết chương trình robot kẹp vật từ A và thả vào vị trí B dựa vào cảm biến

➢ **Yêu cầu viết chương trình cho robot sẽ kẹp vật từ vị trí A bỏ sang vị trí B chỉ khi Ở vị trí A có vật.**



- Tương tự bài tập ***mục II*** nhưng trong bài tập này sẽ dùng cảm biến trên tool kẹp robot kiểm tra vật tại vị trí A. Cần thêm điểm kiểm tra vật dựa vào khoảng cách cảm biến trên tool robot.
- **Chương trình robot**




```
10    CONST robtarget A:=[[160.00,-80.00,320.00],[0.707107,-1.77177E-7,-5.90588
11    CONST robtarget B:=[[120.00,-80.00,615.00],[0.707107,-1.77177E-7,-5.90588
12    CONST robtarget A_Check:=[[160.00,-80.00,500.00],[0.707107,-1.77177E-7,-5
13    CONST robtarget B2:=[[120.00,-80.00,500.00],[0.707107,-1.77177E-7,-5.9058
14    CONST robtarget C:=[[160.00,-80.00,615.00],[0.707107,-1.77177E-7,-5.90588
15    VAR num n:=0;
16    !*********************************************************
17    PROC main()
18        VelSet 100, 5000;
19        AccSet 50, 100;
20        ! chuong trinh chay 1 lan duy nhat
21        WHILE n=0 DO
22          Reset XL_Kep;
23          Reset XL_Nha;
24          WaitTime 0.2;
25          Set XL_Nha;
26          WaitTime 0.2;
27          Reset XL_Nha;
28          n:=n+1;
29        ENDWHILE
30        !.........................
31        MoveL A_Check, v200, z0, tool0;
32        IF SS_Tool=1 THEN
33            MoveL A, v100, z0, tool0;
34            WaitTime 0.5;
35            Set XL_Kep;
36            WaitDI SS_Kep,1; !LUC NAY DE CHAC CHAN LA XI LANH DA KEP VAT
37            Reset XL_Kep;
38            MoveL A_Check, v200, z0, tool0;
39            MoveL B2, v200, z0, tool0;
40            MoveL B, v100, z0, tool0;
41            WaitTime 0.5;
42            Set XL_Nha;
43            WaitDI SS_Nha,1; !LUC NAY DE CHAC CHAN LA XI LANH DA NHA VAT
44            WaitTime 0.5;
45            Reset XL_Nha;
46            MoveL B2, v200, z0, tool0;
47        ENDIF
48
49    ENDPROC
50  ENDMODULE
```

- **Giải thích chương trình**

Khi chạy chương trình, robot di chuyển đến vị trí **A_Check** và kiểm tra cảm biến trên tool. Nếu có vật tại vị trí **A** thì robot sẽ di chuyển đến **A**. robot dừng lại **0.5s** để ổn định vị trí rồi tiến hành kẹp vật. Sau khi kẹp **0.5s** robot di chuyển lên lại điểm **A2** để tránh va chạm rồi di chuyển đến **B2** và **B**. sau **0.5s** robot thả vật xuống và di chuyển lại về vị trí **B2** để tránh va chạm rồi qua về vị trí **A_Check** tiếp tục kiểm tra vật tại **A.** Nếu không có vật thì robot sẽ đứng yêu tại đó đến khi có vật thì robot sẽ di chuyển gắp thả vật từ **A** sang **B**

Bài 7: Chương trình con trong robot

I. Tạo chương trình con

Copy đoạn chương trình chính rồi ***Paste*** vào vị trí muốn để chương trình con. Sau đó sửa lại tên chương trình ta được chương trình con. Chương trình con có thể để phía trên hoặc dưới chương trình chính

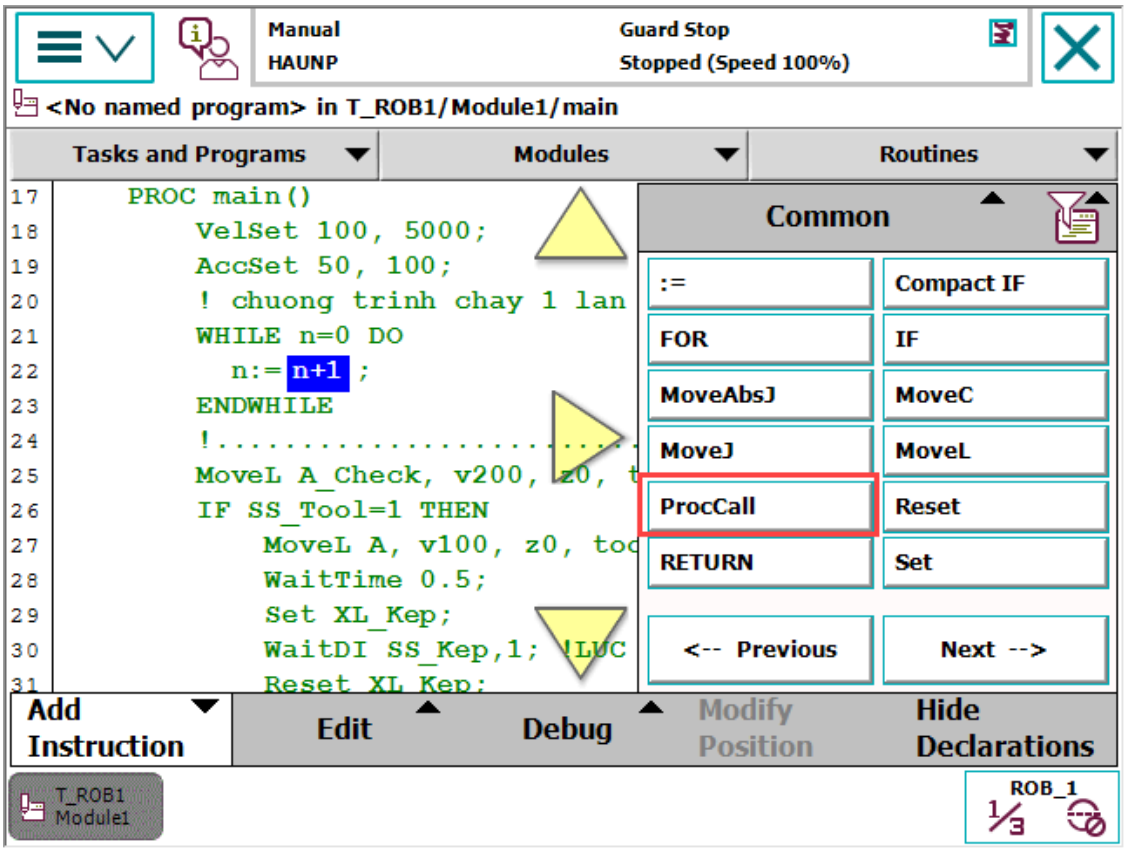
```
16      !**********************
17    + PROC main
50    - PROC main()
51
52    -     WHILE n=0 DO
53            Reset XL_Kep;
54            Reset XL_Nha;
55            WaitTime 0.2;
56            Set XL_Nha;
57            WaitTime 0.2;
58            Reset XL_Nha;
59            n:=n+1;
60          ENDWHILE
61      ENDPROC
62  ENDMODULE
```

```
16      !**********************
17    + PROC main
50    - PROC Reset_Output()
51            Reset XL_Kep;
52            Reset XL_Nha;
53            WaitTime 0.2;
54            Set XL_Nha;
55            WaitTime 0.2;
56            Reset XL_Nha;
57      ENDPROC
58  ENDMODULE
```

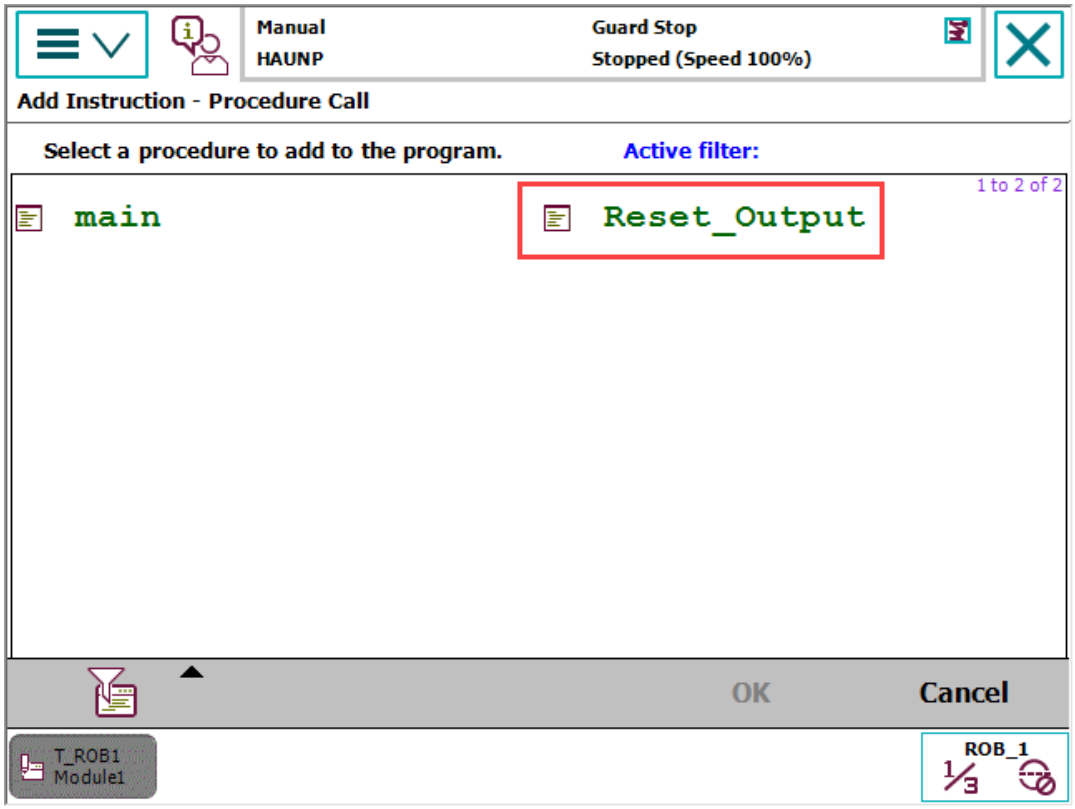
II. Gọi chương trình con

- Trên tay dạy màn hình

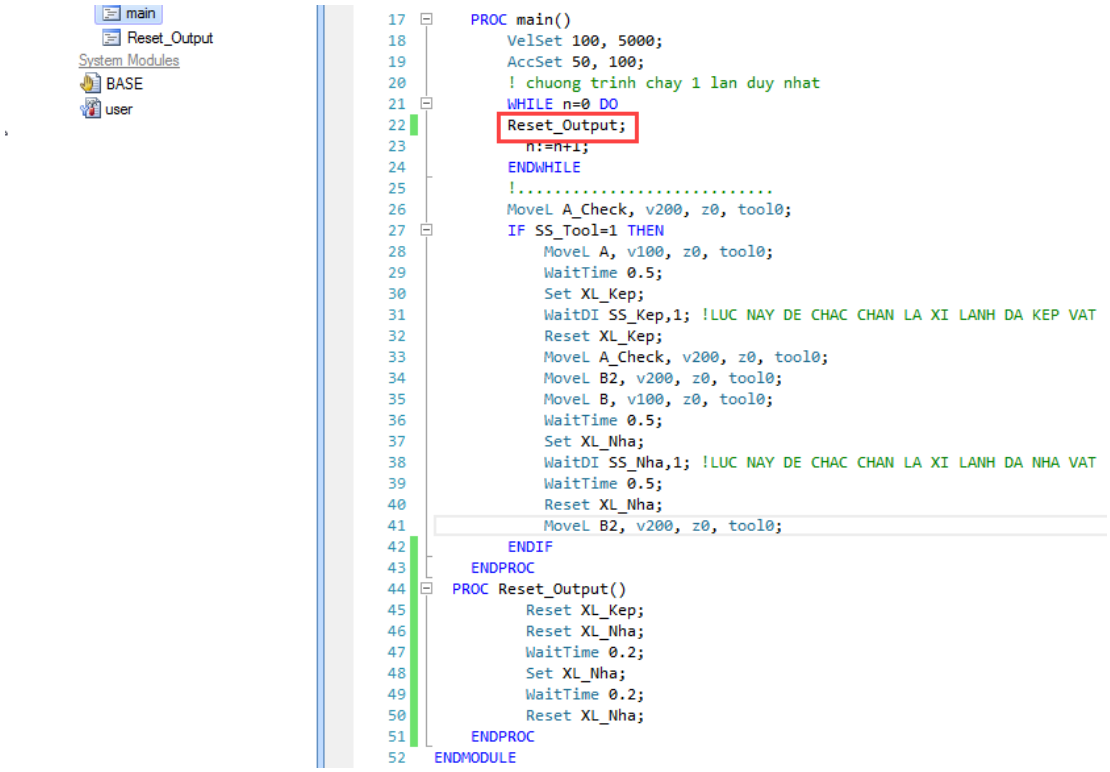
Bước 1: Vào ***Add Instruction*** chọn ***ProcCall***



Bước 2: Chọn chương trình con cần gọi và nhấn ***OK***. Thông báo hiện xác nhận lựa chọn chèn vào trên hoặc dưới dòng lệnh trước đó.



- Trên máy tính

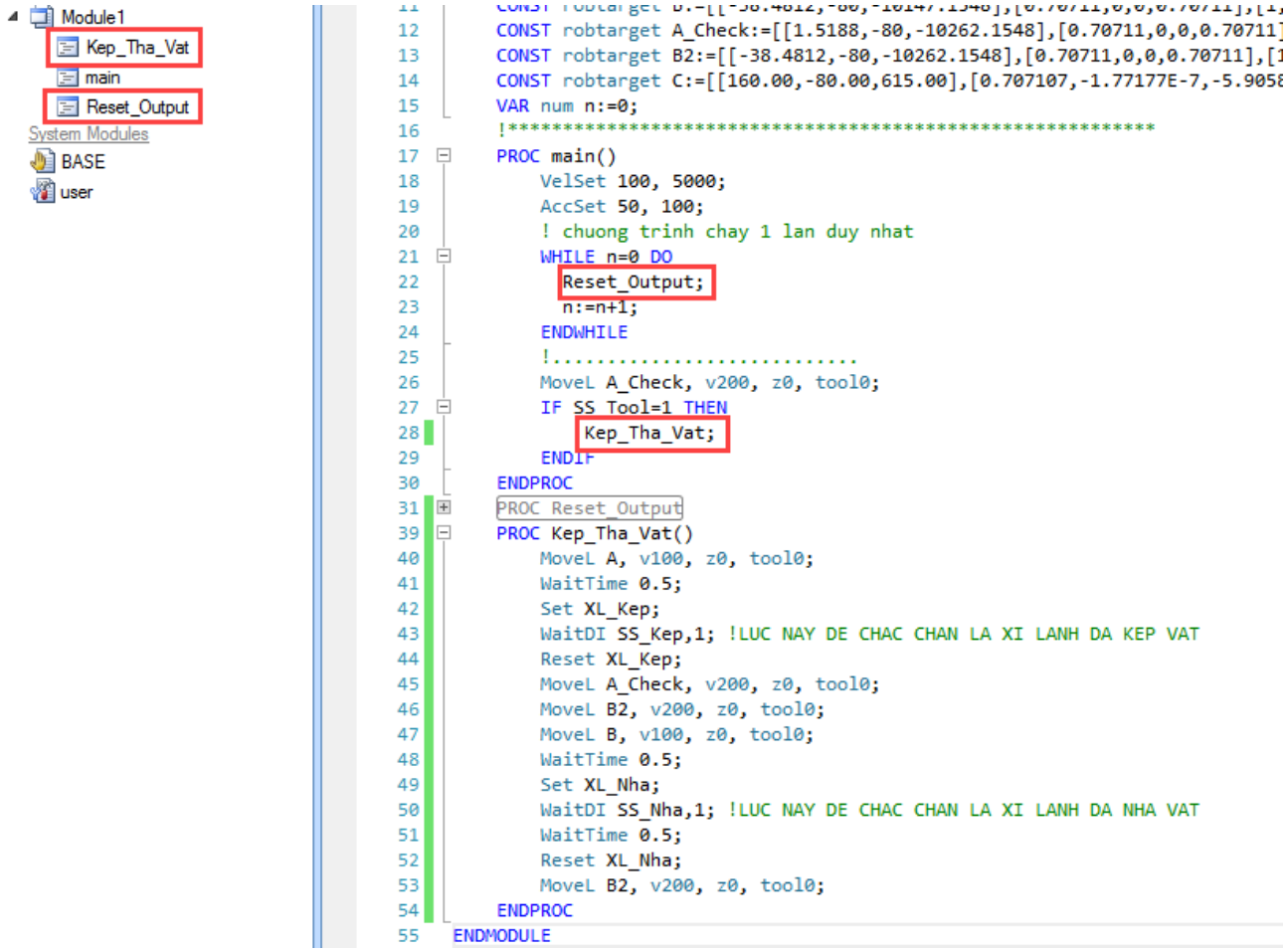


<u>Chú ý:</u> Có thể gọi nhiều chương trình con trong chương trình con. Các chương trình con có thể lồng vào nhau.

III. Viết chương trình robot dùng chương trình con

- **Yêu cầu viết chương trình con đoạn chương trình kẹp-thả vật cho robot theo yêu cầu của mục III Bài 6**

Chương trình robot



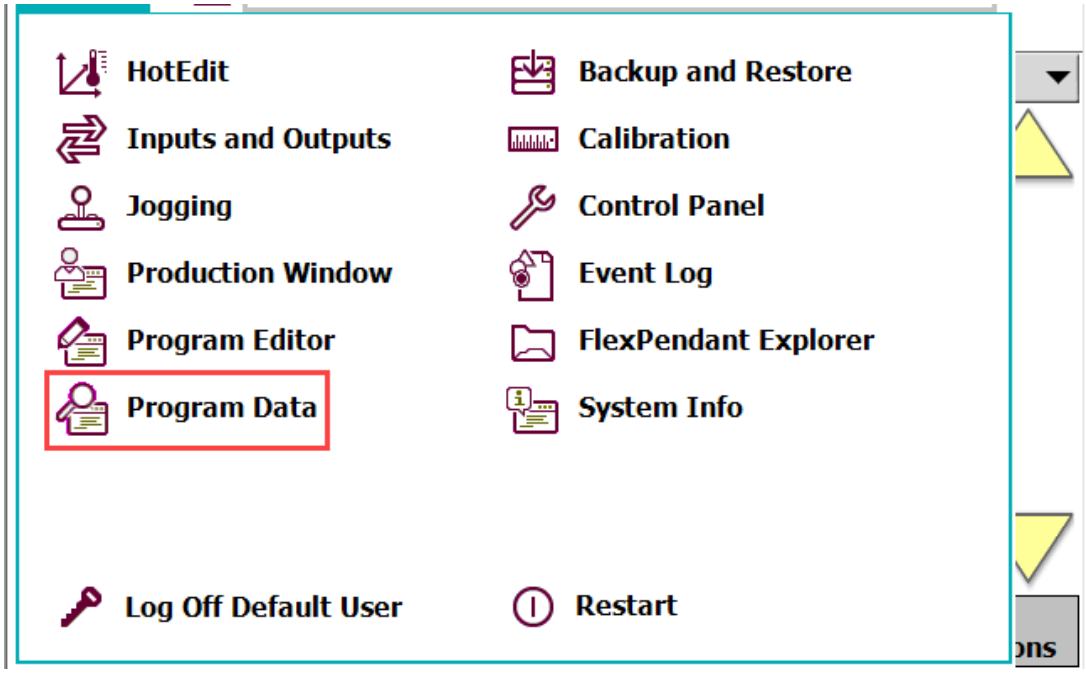
Sau khi nhấn ***Apply All*** để đổ chương trình xuống robot, trên cây thư mục ***Module1*** sẽ tự động thêm danh sách các chương trình con

Bài 8: Tạo mặt phẳng làm việc mới

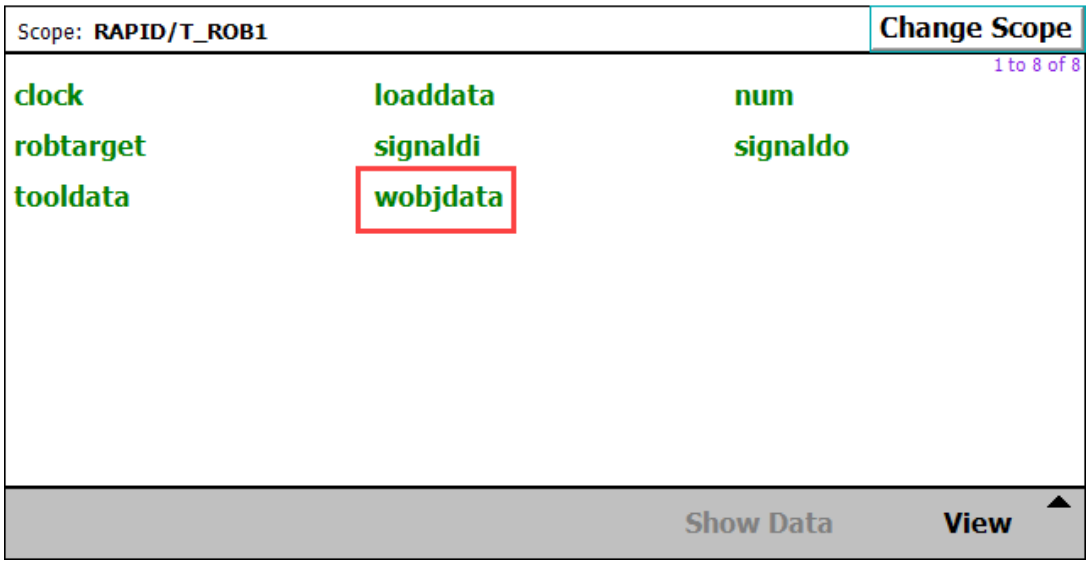
I. Tạo mặt phẳng mới

- Tạo khả năng có thể dịch chuyển khu vực làm việc (ví dụ như bộ gá lắp) mà không cần thay đổi chương trình
- Giúp dễ dàng sử dụng chương trình đã tạo. Khi robot khác làm việc với workplace khác, chỉ cần thay đổi workobject, không cần thay đổi chương trình.

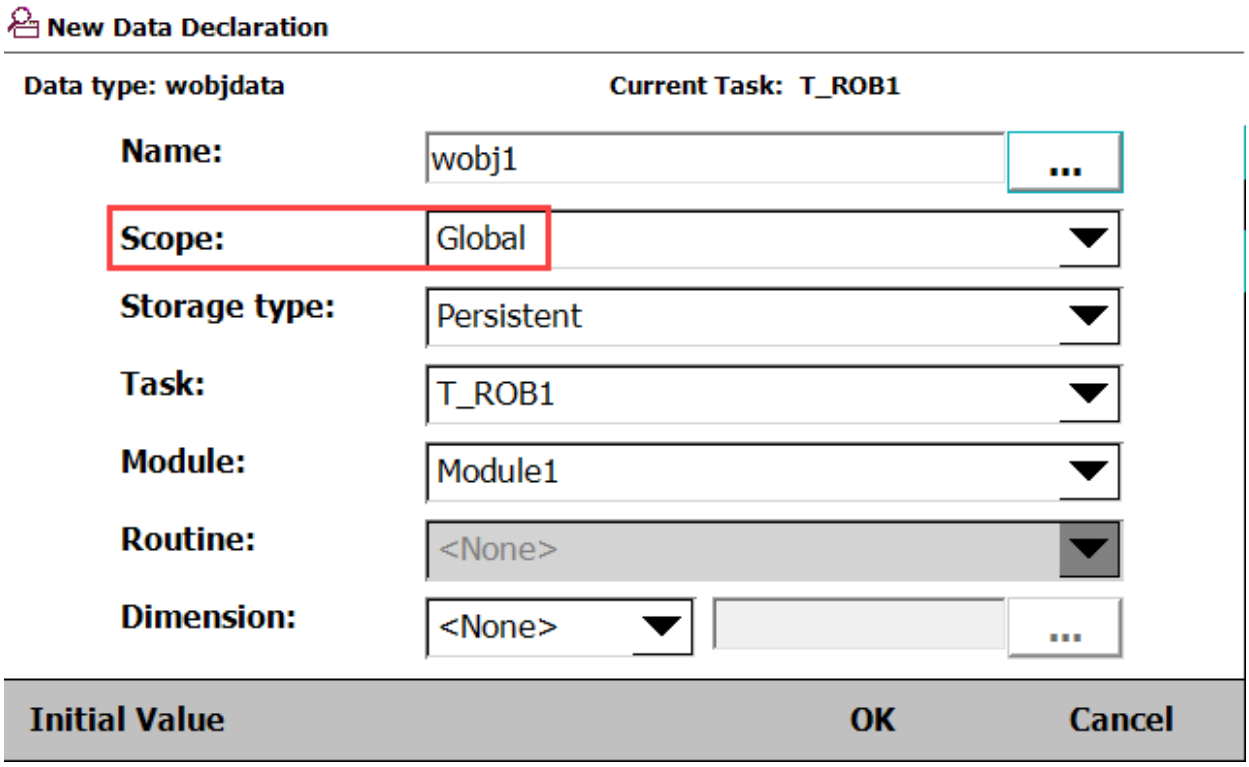
Bước 1: Vào ***Program Data***



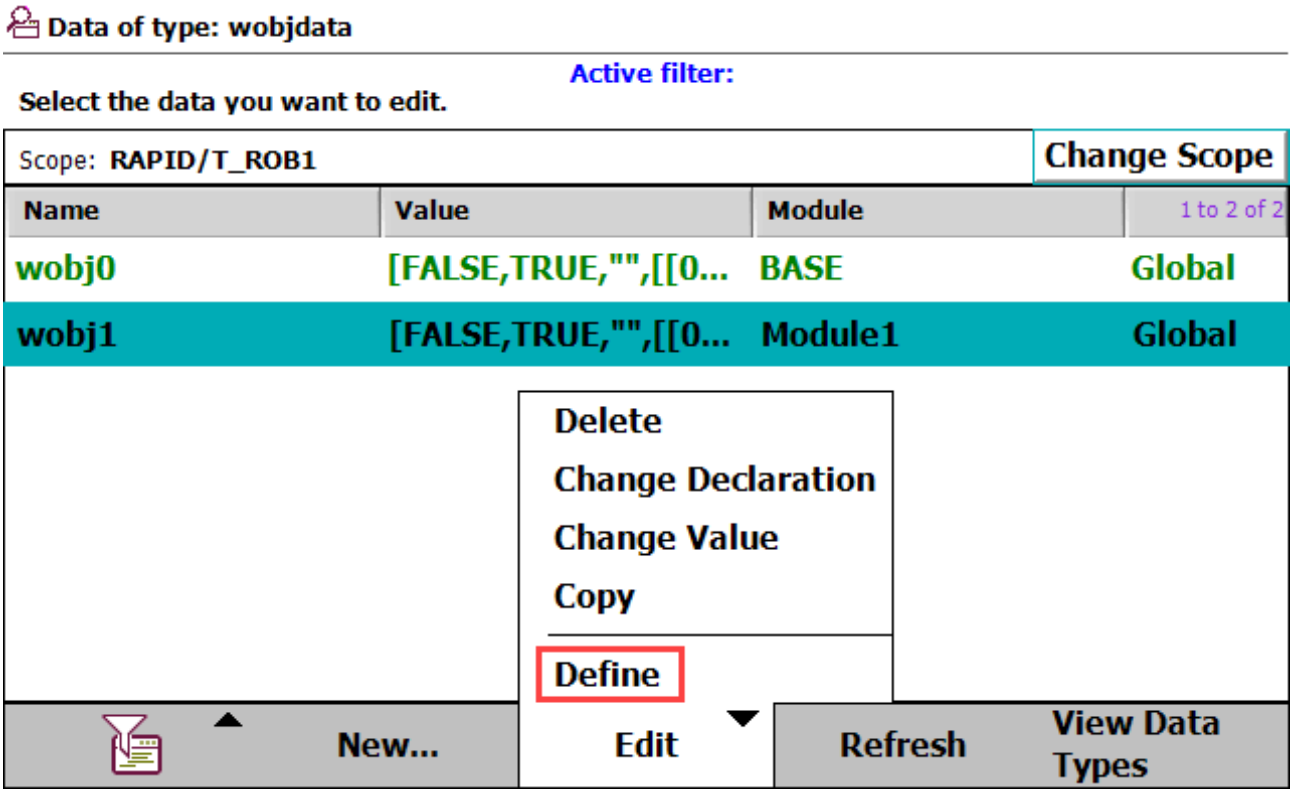
Bước 2: Chọn ***wobjdata***. Nếu không có thì chọn ***View*** → ***All Data Types*** rồi lựa chọn ***wobjdata***



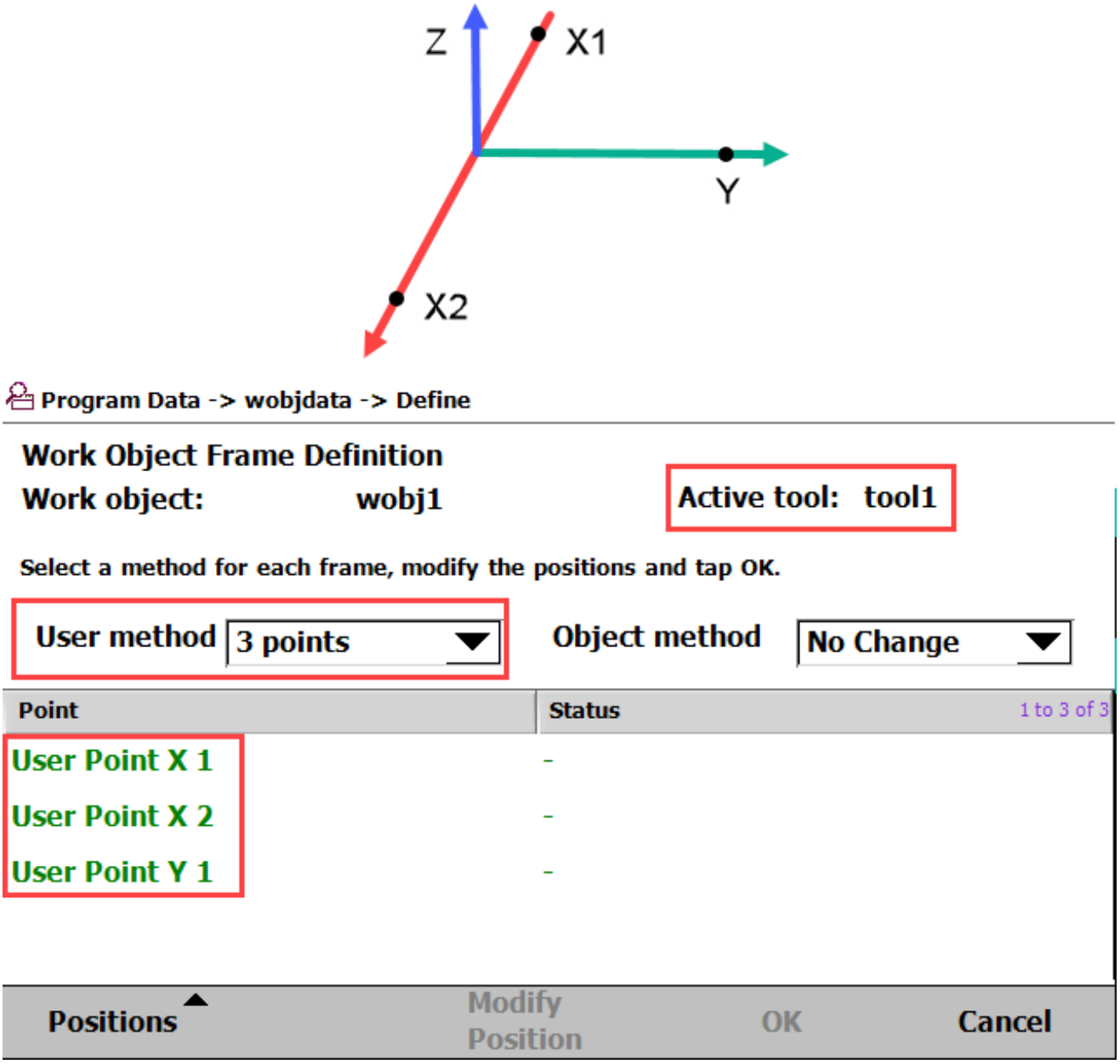
Bước 3: Đặt tên mặt phẳng ở ô ***Name***. Chọn ***Global*** ở mục ***Scope*** để khai báo toàn cục. Sau đó nhấn ***OK***



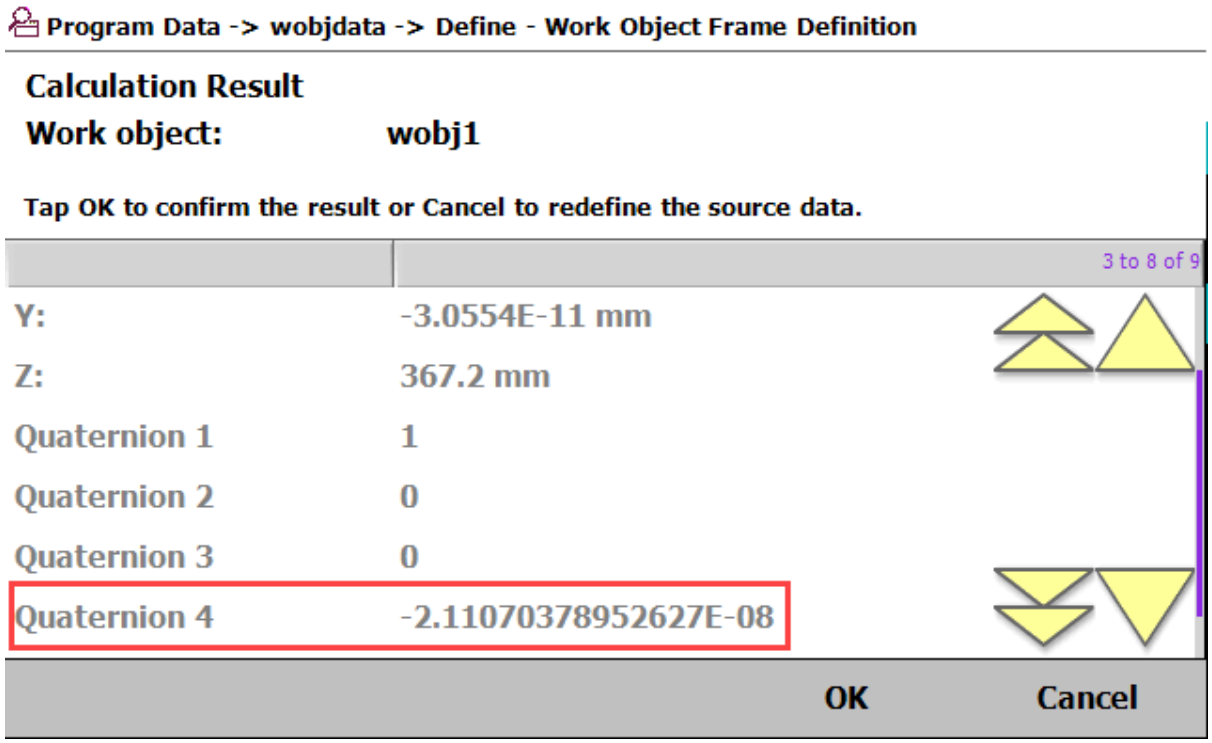
Bước 4: Chọn mặt phẳng và nhấn ***Define*** để tạo thông số cho mặt phẳng



Bước 5: Chọn 3 điểm cần tạo với 2 điểm trên trục X theo chiều dương từ ***X1*** đến ***X2*** và 1 điểm trên trục Y. Điểm ***X1*** có thể không trùng với gốc tạo độ của ***Work Object*** sau đó nhấn ***Modify Position*** tương ứng với mỗi điểm. Chú ý ***Active tool*** sẽ hiển thị tool tạo ***Work Object***. Lựa chọn tool ở tab ***Jogging.***



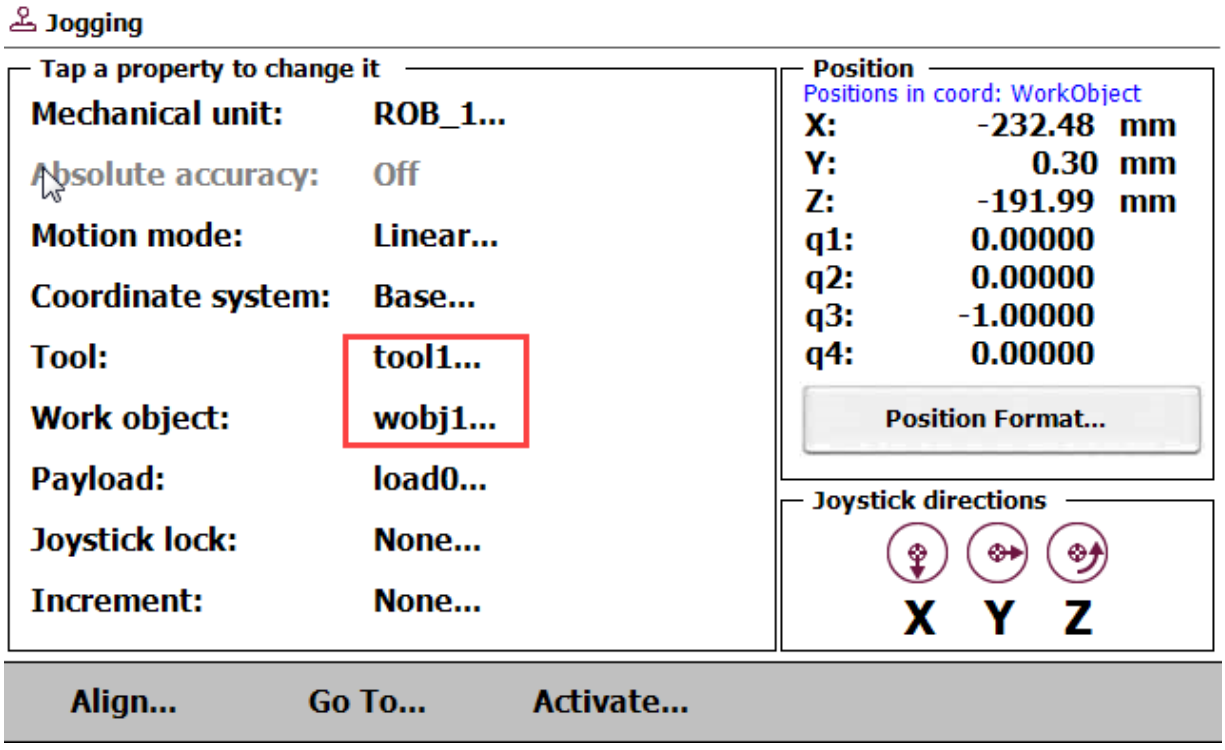
Bước 6: Sau khi tạo xong 3 điểm thì nhấn ***OK***. Robot sẽ hiển thị sai số tạo ***Work Object***



II. Thay đổi mặt phẳng làm việc của robot

➢ **Tạo điểm trên mặt phẳng làm việc mới**

Bước 1: Chọn Tool làm việc và mặt phẳng làm việc (***Work object***) của robot trong tab ***Jogging***



Bước 2: Di chuyển robot đến vị trí cần tạo điểm trên mặt phẳng làm việc

Bước 3: Vào ***Program Data*** chọn ***robtarget*** để tạo điểm. Tọa độ điểm của điểm này sẽ ứng với hệ tọa độ làm việc (***work object***) và ***tool*** làm việc đã chọn.

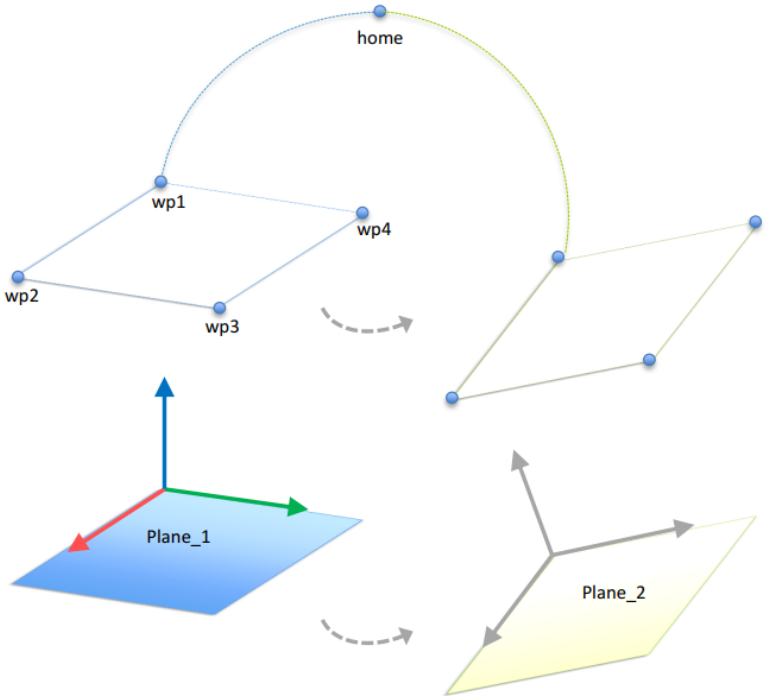
Scope: RAPID/T_ROB1 — Change Scope

Name	Value	Module	1 to 7 of 7
A	[[443.72,0,367.2],[...	Module1	Global
A_Check	[[-0.01,0.3,0.39],[2...	Module1	Global
B	[[443.72,213.8,367...	Module1	Global
B2	[[443.72,213.8,462...	Module1	Global
C	[[160,-80,615],[0.7...	Module1	Global
Home	[[301.99,0,558.01]...	Module1	Global
p10	[[160,-80,615],[0.7...	Module1	Global

New... | Edit | Refresh | View Data Types

Trong trường hợp này, điểm ***A_Check*** có tọa độ ***X=-0.01, Y=0.3, Z=0.39*** so với hệ tọa độ vừa tạo (wobj1) và tool1.

- **Thay đổi mặt phẳng làm việc mới mà không cần set lại điểm đã tạo trên mặt phẳng cũ.**
  Chương trình robot di chuyển 4 điểm trên mặt phẳng **Plane_1**. Để di chuyển lại vị trí 4 điểm mới tương ứng với 4 điểm trên mặt phẳng **Plane_1** thì thực hiện các bước sau:



<u>Bước 1:</u> Tạo một mặt phẳng làm việc mới **wobj2**

<u>Bước 2:</u> Thay đổi mặt phẳng di chuyển trong lệnh ***Move*** bằng cách thay đổi câu lệnh

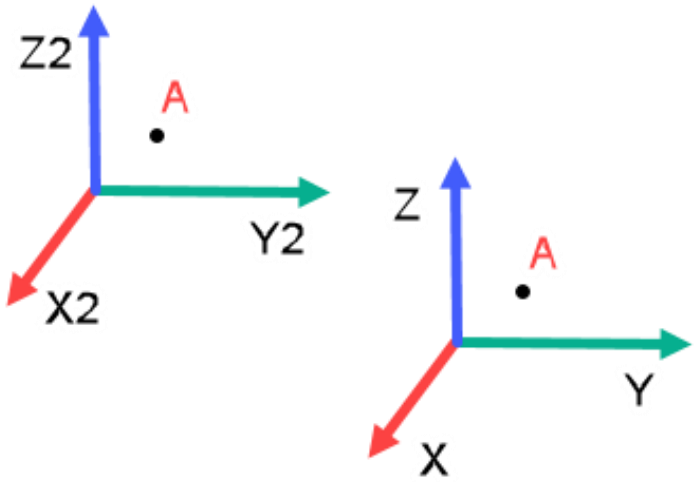
*MoveL A_Check ,v100, z0, tool1****wobj:=wobj2;***

<u>Bước 3:</u> Chạy kiểm tra chương trình robot theo mặt phẳng làm việc vừa tạo

<u>Chú ý:</u> Tọa độ của điểm khi thay đổi work object thì tọa độ mới sẽ ứng với work object đó

Ví dụ: Điểm ***A_Check*** có tọa độ ***X=-0.01, Y=0.3, Z=0.39*** so với hệ tọa độ ***wobj1***

Nếu dùng lệnh thay đổi ***work object*** thì tọa độ của điểm ***A_Check*** cũng sẽ có tọa độ ***X=-0.01, Y=0.3, Z=0.39*** nhưng so với ***wobj2 (***điểm ***A*** sẽ có vị trí mới theo hệ tọa độ ***X2, Y2, Z2*** của ***wobj2)***



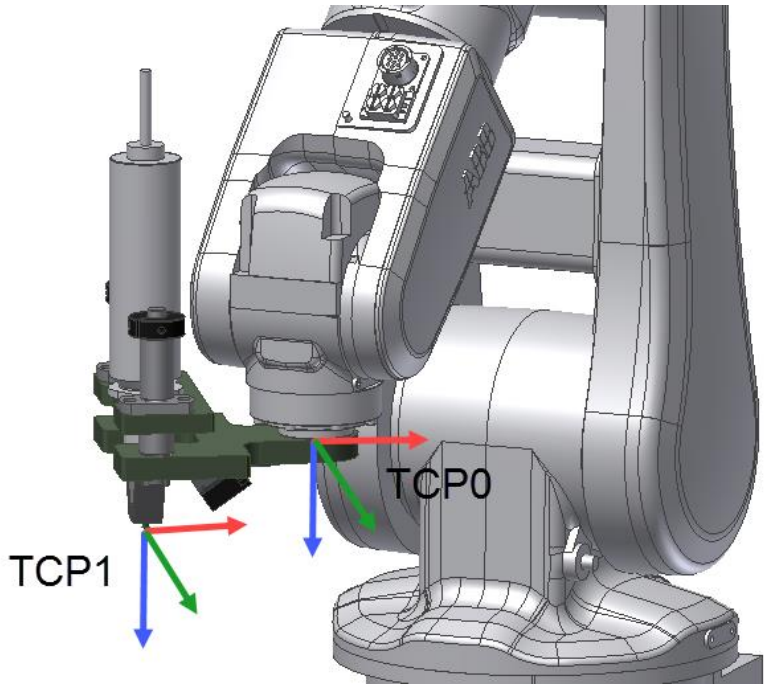
⇨ **Lưu ý khi thay đổi work object, phải chắc chắn rằng tọa độ của các điểm cũ ứng với work object mới phải nằm trong vùng hoạt động của robot.**

Bài 9: Tạo TCP robot

Khi hệ tọa độ tool cần làm việc không trùng với hệ tọa độ tool mặc định của robot, ta cần tạo TCP mới cho tool.

Tọa TCP nhằm giảm thời gian tạo lại điểm làm việc khi thay TCP mới khi tọa độ điểm làm việc không thay đổi.

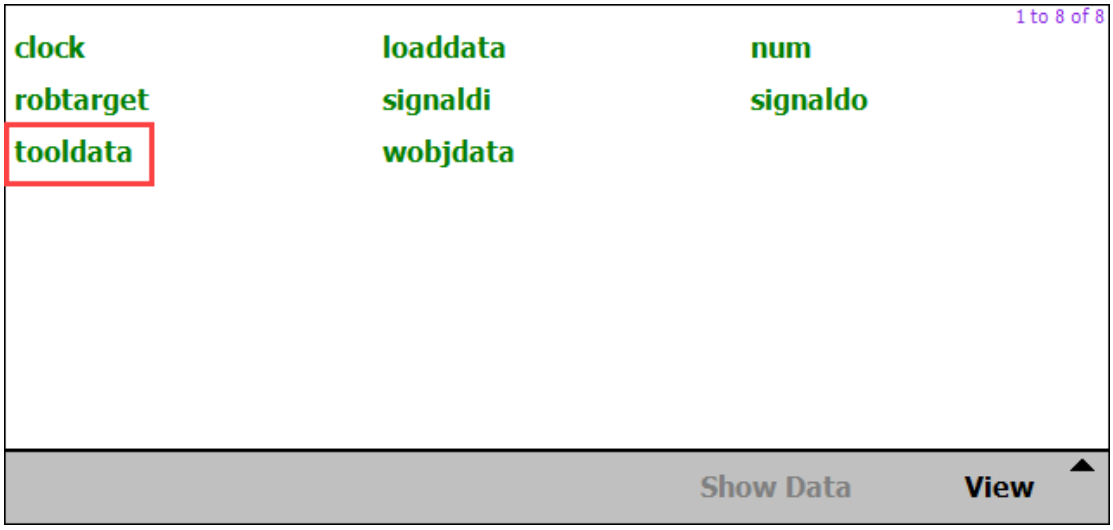
Tọa độ TCP có thể được thiết lập bằng cách di chuyển robot quanh một điểm với ít nhất 3 góc độ khác nhau. Sau khi tạo xong thì robot tự tính toán ra hệ tọa độ TCP mới cho robot.



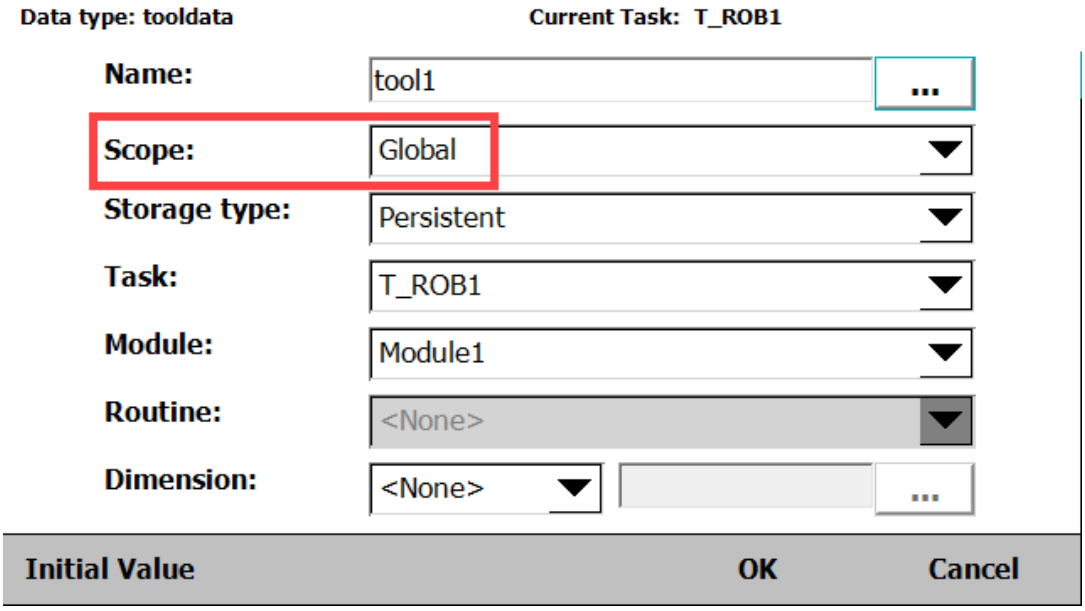
I. Tạo TCP mới

Bước 1: Vào ***Program Data***

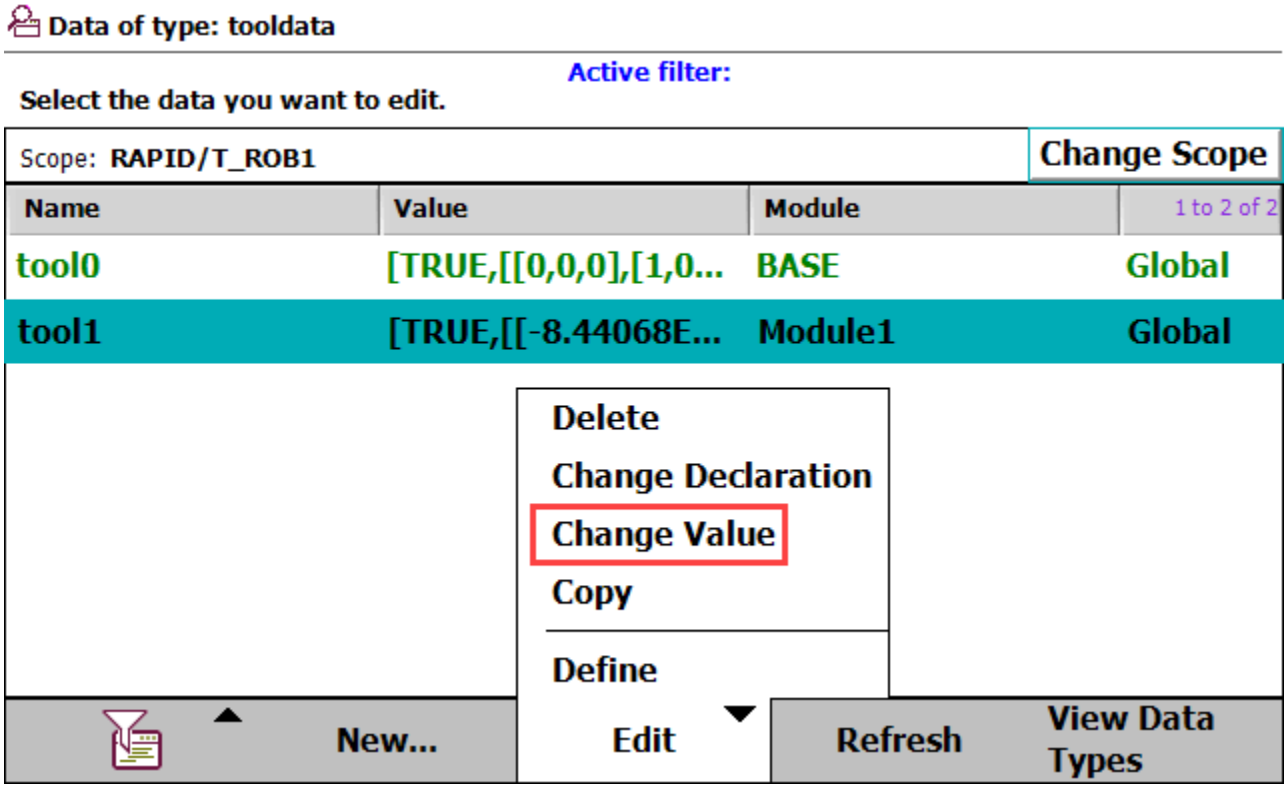
Bước 2: Chọn ***tooldata***. Nếu không có thì chọn ***View*** → ***All Data Types*** rồi lựa chọn ***tooldata***



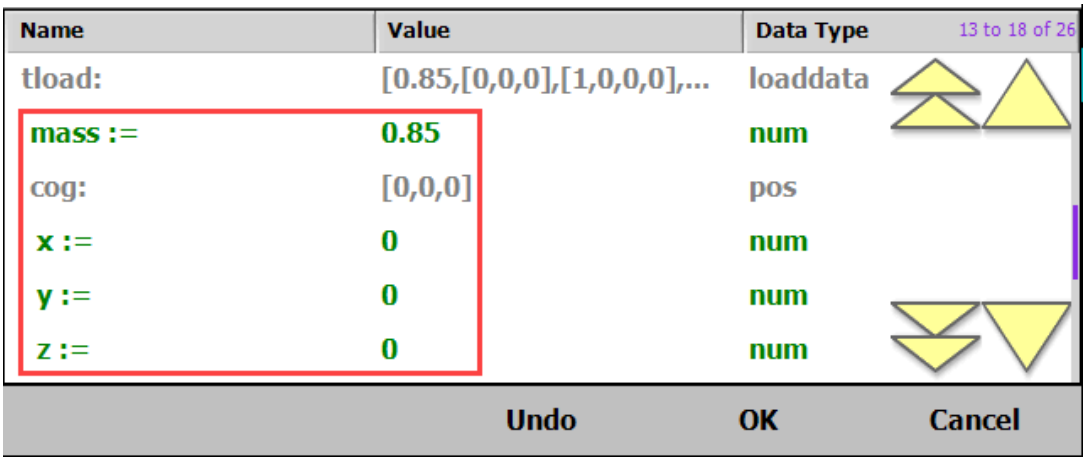
Bước 3: Đặt tên tool ở ô ***Name***. Chọn ***Global*** ở mục ***Scope*** để khai báo toàn cục. Sau đó nhấn ***OK***



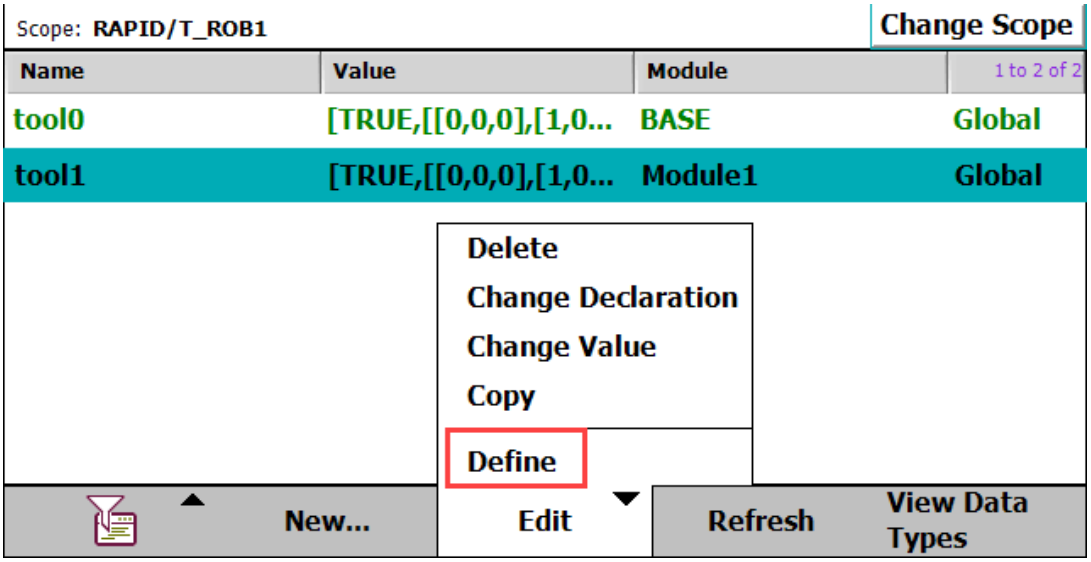
Bước 4: Chọn tool sau đó vào ***Edit*** chọn ***Chane Value***



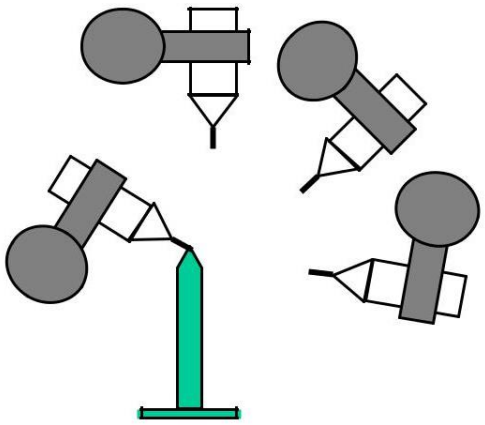
Bước 5: Điền trọng lượng tool vào ô ***mass*** và tâm trọng lượng tool vào giá trị x, y, z (tọa độ này tính trên phần mềm ***Robot Studio***) rồi nhấn ***OK.*** Tọa độ trọng tâm tool có tác dụng giúp robot biết và di chuyển đầu tool robot mượt hơn. Nếu không tính toán được thì không cần phải nhập thông số vào.

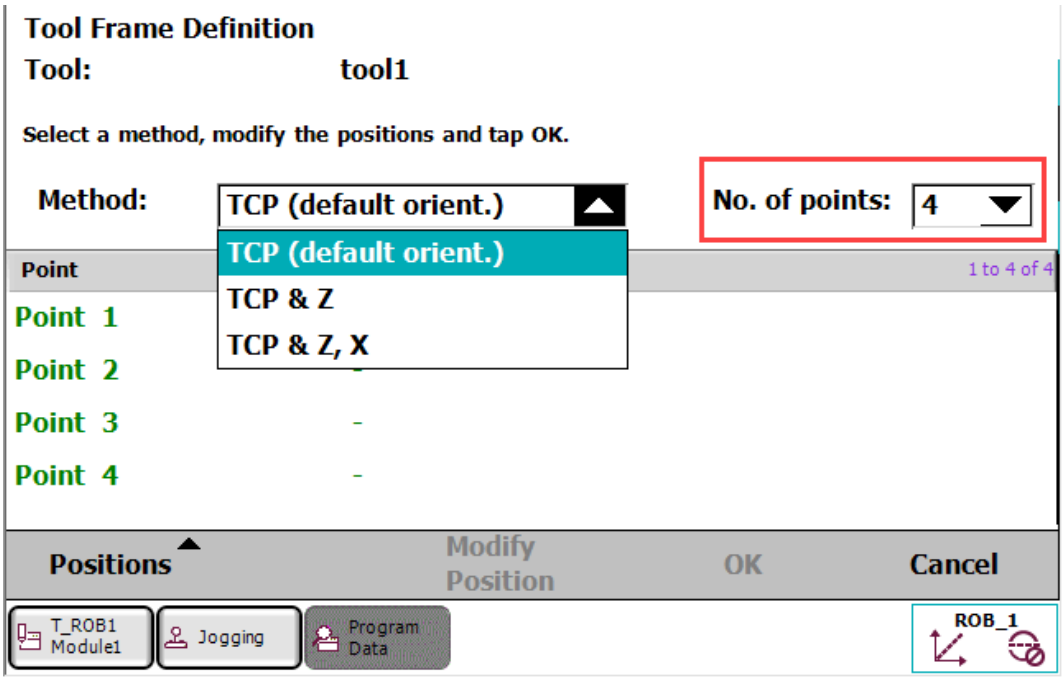


Bước 6: Chọn tool và vào ***Edit*** nhấn ***Define*** để tạo thông số cho tool



Bước 7: Tạo 3 hoặc 4 điểm đầu tool quanh một điểm bất kì với các góc nghiêng khác nhau. Sau đó nhấn ***Modify Position*** tương ứng với mỗi điểm.





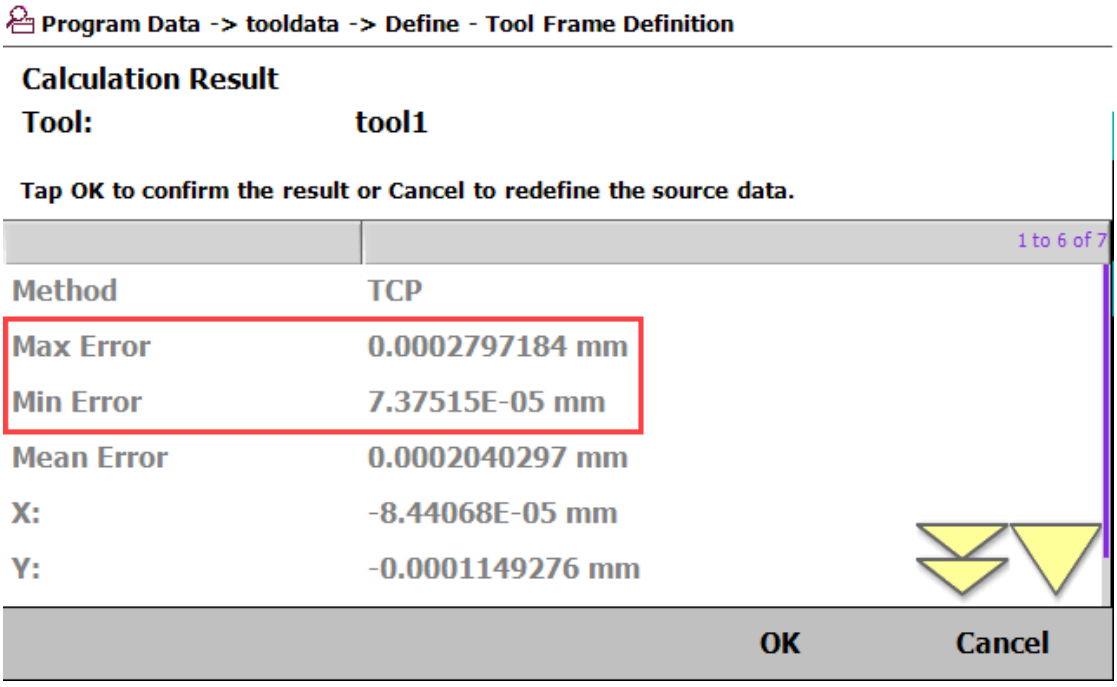
TCP: Tạo TCP với hướng của các trục X, Y ,Z mặc định

TCP & Z: Tạo TCP với hướng của trục Z tự chọn

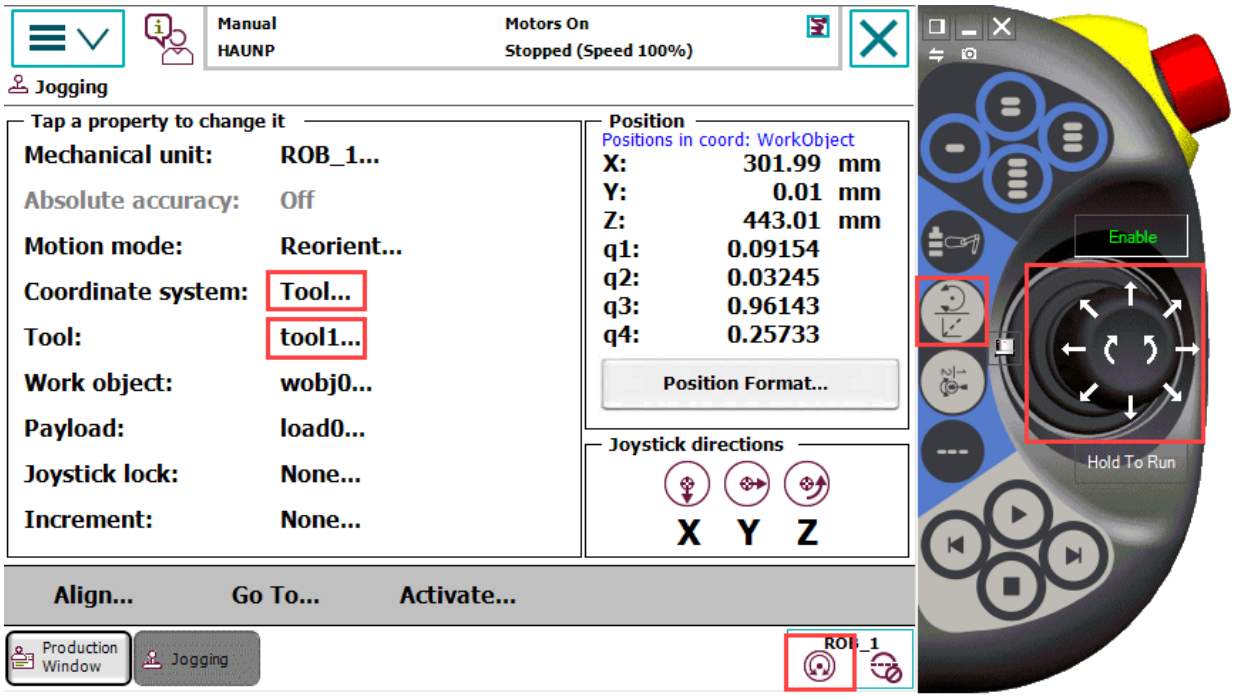
TCP & Z, X: Tạo TCP với hướng của trục Z, X tự chọn

Thay đổi số điểm tạo ở ***No. of points***

Bước 8: Sau khi tạo xong 3 điểm thì nhấn ***OK***. Robot sẽ hiển thị sai số tạo ***Tool. Max Error*** <0.05 là đạt yêu cầu



- Có thể kiểm tra lại tool bằng cách đưa đầu tool đến vị trí vật nhọn và quay tool robot theo tool vừa tạo bằng các thiết lập sau. Nếu xoay quanh đầu tool không bị lệch khỏi một điểm (hoặc lệch ít) thì tool đấy đạt yêu cầu, nếu không thì phải tạo lại tool.



- ***Coordinate system:*** chọn ***Tool***
- ***Tool:*** chọn ***tool1*** (tool vừa tạo)
- Chuyển chế độ di chuyển sang di chuyển xoay
- Di chuyển quay quanh đầu tool vừa tạo bằng tay cầm

II. Thay đổi TCP robot trên mặt phẳng làm việc

Khi trong thực tế, robot đang dùng TCP làm việc trên một mặt phẳng đã tạo. tuy nhiên vì lý do thao tác cần thay tool mới cho robot nhưng robot vẫn làm công việc như thế. Do vậy có thể tạo TCP mới cho robot thay thế cho TCP cũ mà không cần tạo lại điểm làm việc mới.

Bước 1: Tạo TCP mới

Bước 2: Thay đổi TCP mới trong lệnh ***Move***

MoveL A_Check ,v100, z0, ***tool1;***

Bước 3: Chạy kiểm tra chương trình robot theo TCP mới vừa tạo

Bài 10: Biến điểm, biến di chuyển robot

I. Biến di chuyển tọa độ

➢ Lệnh **Offset**

Dùng để di chuyển đến tọa độ mới các tọa độ của điểm đã chọn theo phương X, Y, Z của hệ tọa độ ***Base*** theo lựa chọn di chuyển (***MoveL*** hoặc ***MoveJ***)

MoveL Offs(<điểm>,<offset X>,<offset Y>, <offset Z>), v100,tool0\wobj:=wobj0;

Ví dụ: ***MoveL Offs(p1,100,50,0),v100,tool0;***

Di chuyển đến vị trí cách điểm p1 100 mm theo trục X, 50 mm theo trục Y và 0mm theo trục Z của ***wobj0***

➢ Lệnh **RelTool**

Dùng để di chuyển đến tọa độ mới các tọa độ của điểm đã chọn theo phương X, Y, Z của hệ tọa độ ***TCP*** đã chọn theo lựa chọn di chuyển (***MoveL*** hoặc ***MoveJ***)

MoveL RelTool(<điểm>,<offset X>,<offset Y>,<offset Z>\Rx:=<góc xoay>\Ry:=<góc xoay>\Rz:=<góc xoay>), v100, tool0\wobj:=wobj0;

Ví dụ: ***MoveL RelTool(p1,100,50,0\Rx:=0\Ry:=0 \Rz:=45)***

Di chuyển thẳng đến vị trí cách điểm P1 100mm theo trục X, 50mm theo trục Y, 0mm theo trục Z, đồng thời xoay quanh trục Z một góc 45 độ theo hệ tọa độ ***tool0***

II. Biến mặt phẳng

Có thể tạo mặt phẳng mới dựa trên mặt phẳng cũ theo các thông số offset trục X, Y, Z theo hệ tọa độ ***Base*** như sau:

<wobj>.oframe.trans.x:=<offset X>;

<wobj>.oframe.trans.y:=<offset Y>;

<wobj>.oframe.trans.z:=<offset Z>;

hoặc

<wobj>.oframe.trans:=[X,Y,Z];

Ví dụ:

Di chuyển robot đến điểm Home trên wobj0. Sau đó di chuyển đến điểm Home2 trên work object mới với yêu cầu work object mới cách wobj0 theo tạo độ X=100, Y=100,Z=100 của base.

➢ Chương trình robot

```
17    !*****************************************
18    PROC main()
19        VelSet 100, 5000;
20        AccSet 50, 100;
21            wobj0.oframe.trans.x:=0;
22            wobj0.oframe.trans.y:=0;
23            wobj0.oframe.trans.z:=0;
24            MoveL Home, v200, z0, tool0;
25            wobj0.oframe.trans:=[50,100,50];
26            MoveL Home, v200, z0, tool0;
27
28    ENDPROC
```

Bài 11: Chương trình quét mảng

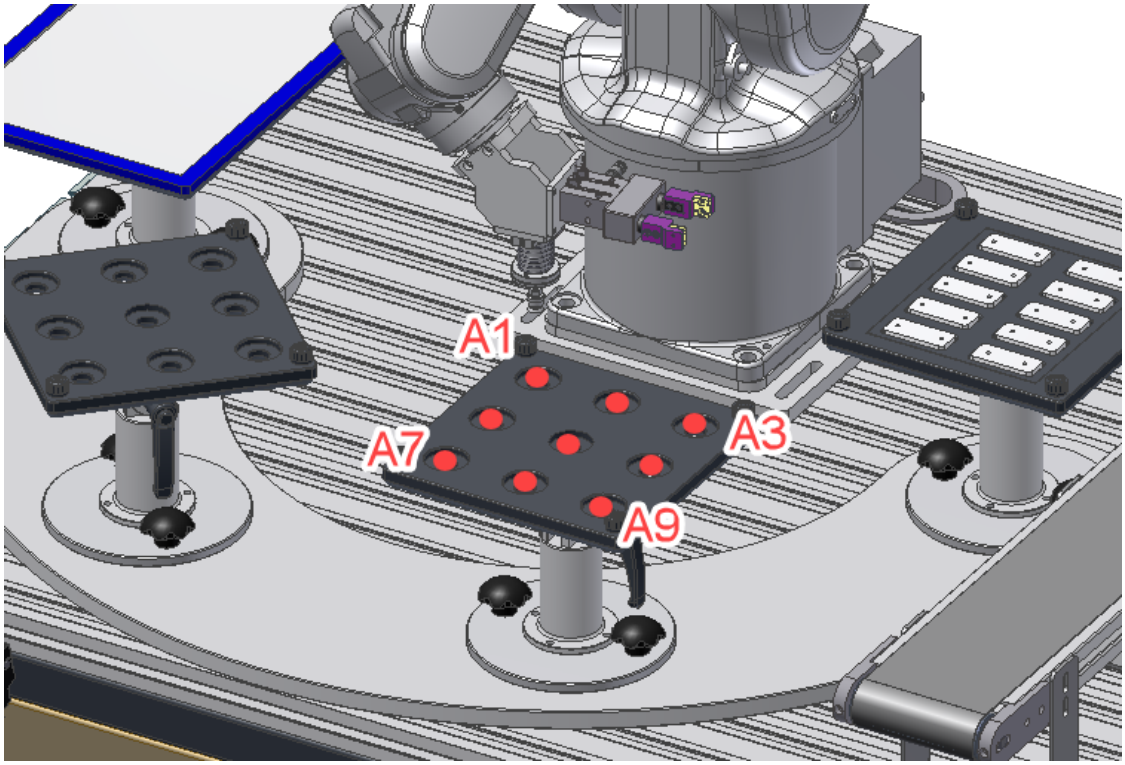
I. Chương trình quét mảng 2 chiều (Square) dùng vòng lặp

Chương trình này sẽ giúp robot chạy được pallettizing trên một mặt phẳng. Đây là code ứng dụng từ **lập trình C** để viết cho robot.Trong chương trình này có ứng dụng các loại biến điểm, biến di chuyển tọa độ…

II.1. Code lập trình C quét mảng 2 chiều có 3 hàng, 3 cột

```
y:=0;
  WHILE y<=2 DO
    x:=0;
    WHILE x<=2 DO
      <câu lệnh>
      x:=x+1;
    ENDWHILE
    y:=y+1;
  ENDWHILE
```

II.2. Viết chương trình di chuyển robot theo thứ tự các điểm A1, A2,…,A9 rồi lặp lại chương trình A1, A2…



❖ **Trường hợp 1:** Các điểm **A1, A2,…,A9** cách đều nhau 80mm hoặc vị trí làm việc của robot tại các điểm này không cần phải chính xác

⇨ Viết chương trình quét mảng 2 chiều chỉ cần tạo một điểm duy nhất

➢ **Chương trình robot**

Tasks and Programs | Modules | Routines

```
17    VAR num x:=0;
18    VAR num y:=0;
19    !***********************************************************
20    PROC main()
21        VelSet 100, 5000;
22        AccSet 50, 100;
23        y:=0;
24        WHILE y<=2 DO
25            x:=0;
26            WHILE x<=2 DO
27              MoveL Offs(A1,x * 80,y * 80,0),v100,z0,tool0;
28              WaitTime 0.2;
29              x:=x+1;
30            ENDWHILE
31            y:=y+1;
32        ENDWHILE
33    ENDPROC
34 ENDMODULE
```

Add Instruction | Edit | Debug | Modify Position | Hide Declarations

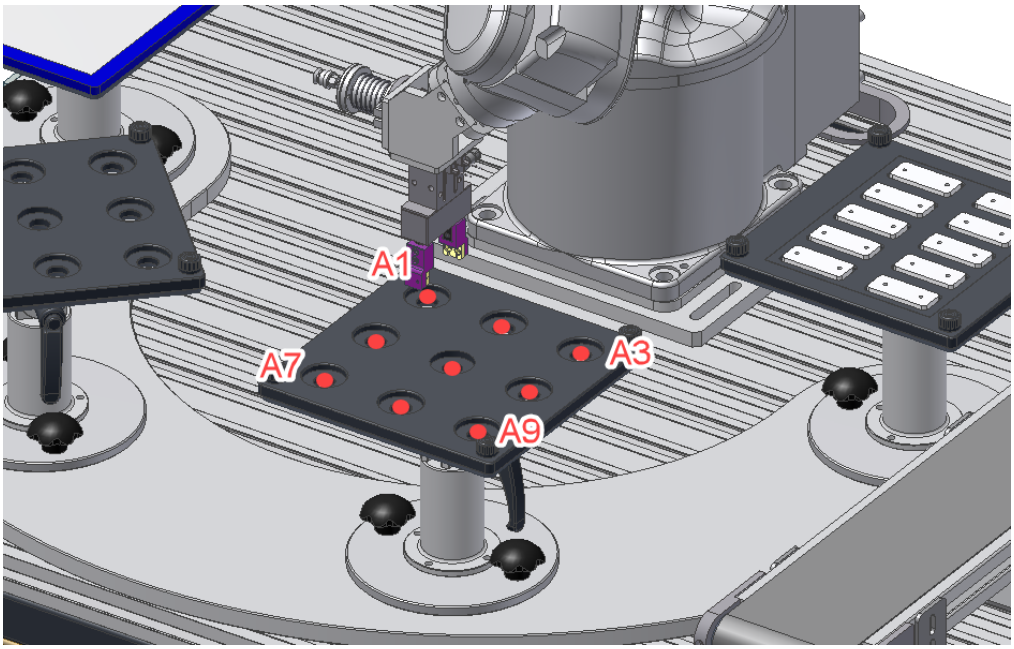
Jogging | Program Data | T_ROB1 Module1 | Production Window | ROB_1

➢ **Giải thích chương trình:**

Trong bài tập này, ta chỉ cần tạo điểm **A1**, sau đó chương trình robot sẽ tự động di chuyển đến các điểm **A2, A3,…,A9** bằng lệnh ***offset*** theo các trục.

MoveL Offs(A1,x*80,y*80,0), v100, z0, tool0;

❖ **Trường hợp 2:** Các điểm **A1, A2,…,A9** không theo quy luật nào **(List Point)** hoặc vị trí làm việc của robot tại các điểm này cần phải chính xác cao.



⇨ Viết chương trình tạo 9 điểm tương ứng 9 vị trí làm việc.

➢ **Chương trình robot theo trường hợp 2 (dùng Test-Case)**

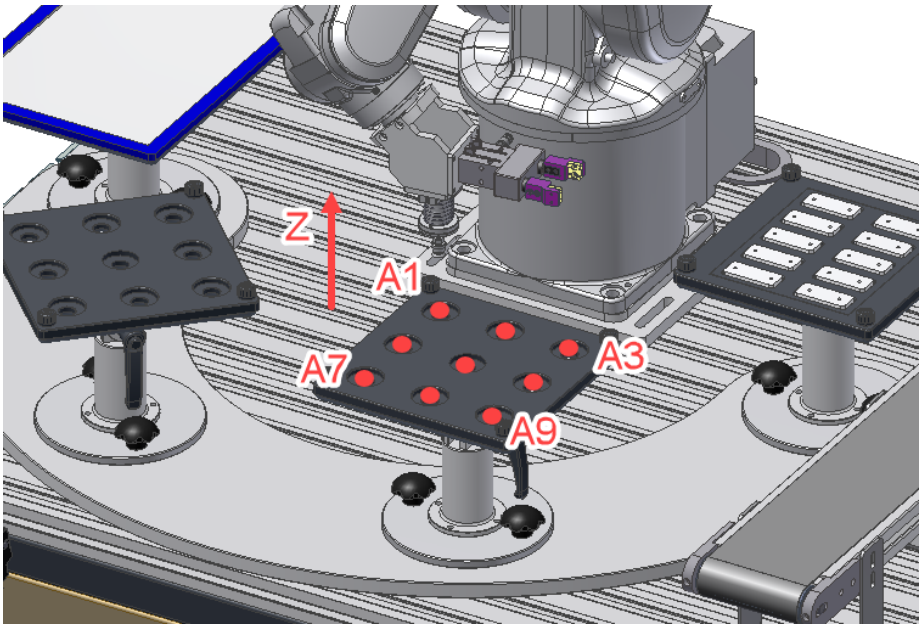
```
27    VAR num dem:=1;
28    !*********************************************************
29    PROC main()
30        VelSet 100, 5000;
31        AccSet 50, 100;
32        WaitTime 0.2;
33        dem:=dem+1;
34        IF dem>9 THEN
35            dem:=1;
36        ENDIF
37        TEST dem
38        CASE 1: MoveL A1,v100,z0,tool0;
39        CASE 2: MoveL A2,v100,z0,tool0;
40        CASE 3: MoveL A3,v100,z0,tool0;
41        CASE 4: MoveL A4,v100,z0,tool0;
42        CASE 5: MoveL A5,v100,z0,tool0;
43        CASE 6: MoveL A6,v100,z0,tool0;
44        CASE 7: MoveL A7,v100,z0,tool0;
45        CASE 8: MoveL A8,v100,z0,tool0;
46        CASE 9: MoveL A9,v100,z0,tool0;
47        DEFAULT:
48        ENDTEST
```

Add Instruction | Edit | Debug | Modify Position | Hide Declarations

II. Chương trình quét các mảng 3 chiều (Box) dùng vòng lặp

Tương tự như chương trình quét mảng 2 chiều (**Square**) dùng vòng lặp, tuy nhiên có thể ứng dụng thêm biến mặt phẳng để chạy palletizing tạo thành mảng 3 chiều (**Box**)

➢ Viết chương trình di chuyển các điểm A1, A2,…,A9 sau đó di chuyển các điểm A1', A2',…, A9' cách các điểm tương ứng A1, A2,…,A9 một đoạn theo trục Z 50mm và sau đó di chuyển các điểm A1'', A2'',…, A9'' cách các điểm tương ứng A1', A2',…,A9' một đoạn theo trục Z 50mm và quay trở lại điểm A1, A2,..,A9 tiếp tục chu trình mới.



- **Chương trình robot**

```
25    VAR num x:=0;
26    VAR num y:=0;
27    VAR num z:=0;
28    !*********************************************************
29    PROC main()
30        VelSet 100, 5000;
31        AccSet 50, 100;
32        z:=0;
33        WHILE z<=2 DO
34            y:=0;
35            WHILE y<=2 DO
36              x:=0;
37                WHILE x<=2 DO
38                    MoveL Offs(A1,x*80,y*80,z*50), v100, z0, tool0;
39                    WaitTime 0.2;
40                    x:=x+1;
41                ENDWHILE
42                y:=y+1;
43            ENDWHILE
44            z:=z+1;
45        ENDWHILE
46    ENDPROC
```

Add Instruction | Edit | Debug | Modify Position | Hide Decl

- **Giải thích chương trình:**

Trong bài tập này, ta chỉ cần tạo điểm **A1**, sau đó chương trình robot sẽ tự động di chuyển đến các điểm **A2, A3,…,A9 và các điểm offset A1', A2'…** bằng lệnh ***offset*** theo các trục.

MoveL Offs(A1,x*80,y*80,z*50), v100, z0, tool0;

Bài 12: Tạo mảng chứa dữ liệu

I. Tạo mảng chứa dữ liệu

PERS num {<số cột>,< số hàng >};

PERS num{<số cột>,< số hàng >}:=[[<giá trị 1>,...,<giá trị n>],[<giá trị 1>,...,<giá trị m>]];

Số dữ liệu trong mảng bắt đầu từ 1, kiểu biến có thể là VAR, PERS, CONTS.. theo yêu cầu bài toán

Ví dụ:

- Mảng một chiều:

  PERS num Vit1_dataX{7};

  PERS num Vit1_dataX{5}:=[2.15, 3.2, 4.15, 9.15, 9.48];
- Mảng hai chiều:

  PERS num Vit2_dataX{3,9};

  PERS num Vit2_dataX{2,3}:=[[2.15, 3.2, 4.15], [7.25, 9.15, 9.48]];

II. Dịch dữ liệu trong mảng một chiều

Trong chương trình này, tạo 2 mảng một chiều **Data_X**, **Data_Y** và tiến hành dịch dữ liệu trong mảng theo quy luật dồn dữ liệu:

- Dữ liệu ô nhớ 5 bằng dữ liệu ô nhớ 4
- Dữ liệu ô nhớ 4 bằng dữ liệu ô nhớ 3
- Dữ liệu ô nhớ 3 bằng dữ liệu ô nhớ 2
- Dữ liệu ô nhớ 2 bằng dữ liệu ô nhớ 1
- Dữ liệu ô nhớ 1 bằng dữ liệu ô nhớ 0

Sau đó sẽ in ra dữ liệu ô nhớ 5 sau mỗi lượt dịch dữ liệu.

- **Chương trình robot**

```
FOR i FROM 6 TO 2 STEP -1 DO
    Vit1_dataX{i} := Vit1_dataX{i-1};
    Vit1_dataY{i} := Vit1_dataY{i-1};
    Vit2_dataX{i} := Vit2_dataX{i-1};
    Vit2_dataY{i} := Vit2_dataY{i-1};
    TPWrite "gia tri cap vit X1="\num:= Vit1_dataX{6};
    TPWrite "gia tri cap vit Y1="\num:= Vit1_dataY{6};
    TPWrite "gia tri cap vit X2="\num:= Vit2_dataX{6};
    TPWrite "gia tri cap vit Y2="\num:= Vit2_dataY{6};
    WaitTime 0.2;
ENDFOR
```