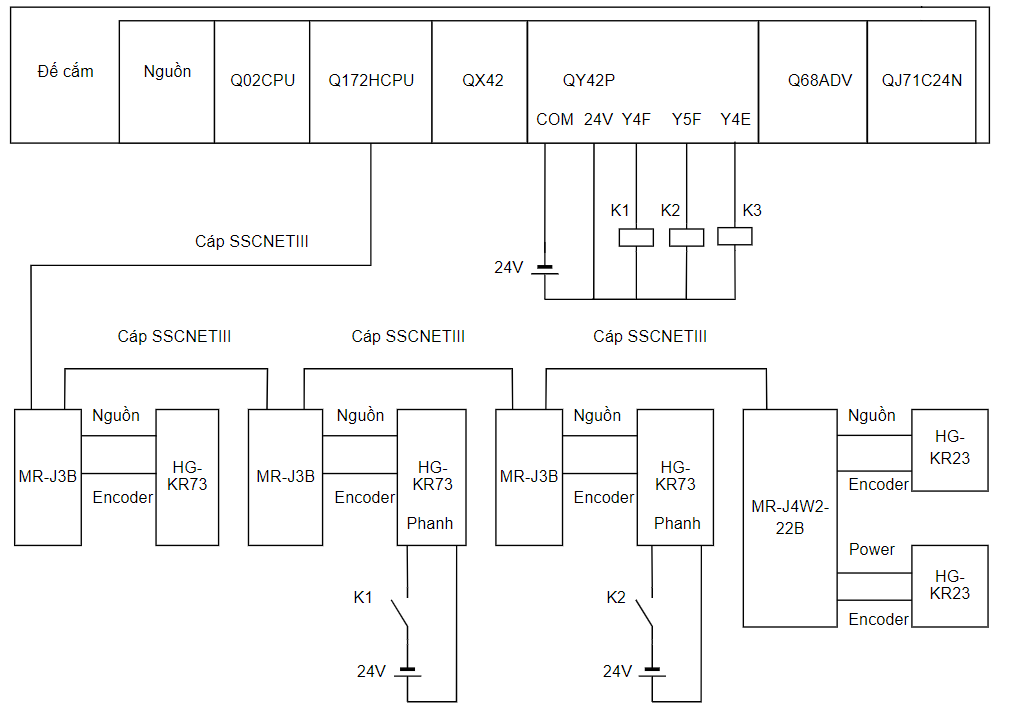
**Chương 4: Điều khiển cánh tay robot 5 bậc tự do sử dụng PLC Mishubishi**

PLC (Programmable Logic Controller) là một thiết bị không thể thiếu trong ngành công nghiệp hiện đại nhờ vào tính năng ổn định, linh hoạt và khả năng chịu đựng môi trường khắc nghiệt. Việc sử dụng PLC để điều khiển robot mang lại nhiều lợi ích vượt trội. Thứ nhất, PLC có khả năng xử lý nhanh chóng và chính xác các tín hiệu điều khiển, đảm bảo hoạt động của robot được thực hiện một cách hiệu quả và chính xác. Thứ hai, PLC dễ dàng lập trình và tái cấu trúc, giúp cho việc thay đổi và cải tiến hệ thống điều khiển robot trở nên đơn giản và tiết kiệm chi phí. Cuối cùng, với khả năng tích hợp cao, PLC có thể kết nối và điều khiển nhiều thiết bị khác nhau trong một hệ thống tự động hóa, tạo nên một giải pháp điều khiển toàn diện và tối ưu cho robot trong các dây chuyền sản xuất công nghiệp.

1. **Sơ lược về hệ thống điều khiển trong PLC**
   1. **Mô hình chung cho hệ thống điều khiển của PLC**



Ta có danh sách của các bộ phận thuộc hệ thống điều khiển như sau:

**- Base Q35B:**

Module nguồn cấp cho họ PLC Q gồm 5 chỗ cho các module PLC và 1 chỗ cho module nguồn.



**- PLC Q02HCPU**:

* **Vai trò:**

PLC này hỗ trợ đa dạng các kiểu module chức năng (truyền thông, Analog, phát xung, I/O,…). Với khả năng hỗ trợ số lượng đầu vào I/O tối đa ở mức khá lớn và chức năng ngắt thì Q02HCPU đáp ứng hầu hết các ứng dụng điều khiển phân tán quy mô vừa cần sử dụng ngắt ngoài. Ngoài ra, việc mở rộng thêm các Remote I/O cũng như kết nối mạng truyền thông công nghiệp rất thuận tiện với các module chức năng có sẵn.

* **Chức năng:**

Trong hệ thống điều khiển, module Q02HCPU có vai trò như một kênh giao tiếp giữa các module với nhau để đưa ra những tín hiệu điều khiển cho động cơ để thực thi. Ngoài ra, module Q02HCPU còn có khả năng liên kết với C# để có thể giúp người dùng tương tác với giao diện một cách linh hoạt.

**- Motion Controller Q172HCPU:**

* **Vai trò:**

Motion Controller Q172HCPU của Mitsubishi là bộ điều khiển đa trục, lý tưởng cho các ứng dụng điều khiển chuyển động phức tạp. Với khả năng điều khiển chính xác nhiều động cơ servo, hỗ trợ đa dạng tín hiệu và giao diện kết nối như Ethernet và CC-Link, Q172HCPU đảm bảo hiệu suất cao và ổn định. Thiết bị này là lựa chọn tối ưu cho các hệ thống tự động hóa và điều khiển robot trong công nghiệp.

* **Chức năng:**

Trong hệ thống điều khiển, Q172HCPU có vai trò quan trọng trong việc đưa ra tín hiệu điều khiển đến cho động cơ AC servo, là bộ não quyết định việc thực thi cho động cơ.

**- Module input QX42:**

* **Vai trò:**

Module input QX42 của Mitsubishi thuộc dòng MELSEC-Q SERIES, là module đầu vào số DC với 64 kênh và điện áp định mức 24VDC. Module này chiếm 64 điểm I/O trên PLC và sử dụng kết nối kiểu connector. Với dòng tiêu thụ 0,075A và kích thước nhỏ gọn (27,4mm x 98mm x 90mm), QX42 dễ dàng tích hợp vào các hệ thống tự động hóa. Trọng lượng chỉ 0,18kg, QX42 là lựa chọn lý tưởng cho các ứng dụng yêu cầu thu thập tín hiệu nhanh và chính xác trong các môi trường công nghiệp.

* **Chức năng:**

Trong hệ thống điều khiển, module này được sử dụng để đọc những tín hiệu từ cảm biến tiệm cận cũng như công tắc được gắn trên băng tải để từ đó làm dữ kiện để những module khác thực thi những chức năng phù hợp.



**- Module output QY42:**

* **Vai trò:**

Module output QY42P của Mitsubishi thuộc dòng MELSEC-Q SERIES, là module đầu ra transistor (sink) với 64 kênh. Thiết bị này hoạt động với điện áp nguồn DC từ 12-24V và có logic đầu ra loại sink. QY42P chiếm 64 điểm I/O trên PLC, sử dụng kết nối dây kiểu screw, và có dòng tiêu thụ 0,15A. Kích thước nhỏ gọn (27,4mm x 98mm x 90mm) và trọng lượng nhẹ (0,17kg) giúp QY42P dễ dàng tích hợp vào các hệ thống tự động hóa, đảm bảo điều khiển các thiết bị ngoại vi một cách hiệu quả và chính xác.

* **Chức năng:**

Trong hệ thống điều khiển, module này dùng để xuất tín hiệu ngõ ra để điều khiển băng tải và kết nối đến servo module để bật tắt thắng điện từ cho động cơ AC trong những trường hợp khẩn cấp.



**- Servo amplifier MR-J3-70B:**

* **Vai trò:**

Servo amplifier MR-J3-70B thuộc dòng MELSERVO J3 series của Mitsubishi Electric, là một bộ điều khiển servo AC mạnh mẽ và linh hoạt. Với công suất định mức 0.75kW và dòng điện 5.8A, MR-J3-70B là lựa chọn lý tưởng cho các ứng dụng đòi hỏi độ chính xác và độ tin cậy cao. Bộ điều khiển này được thiết kế với nhiều tính năng bảo vệ như ngắn mạch, quá tải, và quá tốc độ, đảm bảo an toàn cho cả thiết bị và hệ thống. MR-J3-70B có kích thước nhỏ gọn (168mm x 60mm x 185mm) và tiêu thụ điện năng thấp (3.8A), dễ dàng tích hợp và vận hành trong các môi trường công nghiệp khắc nghiệt. Đặc biệt, với khả năng giao tiếp nhanh chóng qua chuẩn SSCNET III/H, MR-J3-70B giúp tối ưu hóa hiệu suất và linh hoạt trong các ứng dụng điều khiển chuyển động.

* **Chức năng:**

Trong hệ thống điều khiển, module servo này sẽ dùng để truyền tín hiệu điều khiển cho động cơ AC bằng việc giao tiếp với **module controller** với những quyết định từ phía người dùng.



**- Servo amplifier MR-J4W2-22B:**

* **Vai trò:**

Servo amplifier MR-J4W2-22B thuộc dòng MELSERVO J4 series của Mitsubishi Electric, là một bộ điều khiển servo AC đa chức năng và tiên tiến. Với công suất định mức 0.2kW và dòng điện 1.5A, MR-J4W2-22B đáp ứng nhu cầu điều khiển chuyển động chính xác và linh hoạt trong các ứng dụng công nghiệp. Bộ điều khiển này tích hợp nhiều tính năng bảo vệ và điều khiển như chức năng an toàn Safe Torque Off (STO), giao tiếp nhanh chóng qua chuẩn SSCNET III/H, và các chức năng tự động điều chỉnh như tự động điều chỉnh và tự động điều chỉnh bằng một nút chạm. Với kích thước nhỏ gọn (168mm x 60mm x 195mm), MR-J4W2-22B dễ dàng tích hợp và vận hành trong các môi trường công nghiệp đòi hỏi hiệu suất và độ tin cậy cao.

* **Chức năng:**

Trong hệ thống điều khiển, Servo là bộ phận quyết định cho việc di chuyển của robot, nhận tín hiệu điều khiển từ các module PLC để thực thi một cách chính xác những hành động, câu lệnh đã được xử lí một cách nhanh chóng, chính xác nhất.

|  |  |
| --- | --- |
| **Bộ phận** | **Link tham khảo** |
| Base Q35B | [Q35B - Main Base Mitsubishi - Codienhaiau.com](https://codienhaiau.com/product/mitsubishi-q35b/) |
| Q02HCPU | [Q02HCPU - PLC Mitsubishi - Nhà Phân phối Tự Động Hóa Toàn Cầu](https://plcmitsubishi.com/q02hcpu) |
| Q172HCPU |  |
| QX42 | [QX42 - Module ngõ vào Mitsubishi - Codienhaiau.com](https://codienhaiau.com/product/mitsubishi-qx42/) |
| QY42P | [QY42P - Module Output Mitsubishi - Codienhaiau.com](https://codienhaiau.com/product/mitsubishi-qy42p/) |
| MR-J3-70B | [MR-J3-70B - Bộ điều khiển Servo Mitsubishi 0.75kW (codienhaiau.com)](https://codienhaiau.com/product/servo-amplifier-mitsubishi-mr-j3-70b/) |
| MR-J4W2-22B | [MR-J4W2-22B - Bộ điều khiển Servo Misubishi 0.2kW (codienhaiau.com)](https://codienhaiau.com/product/servo-amplifier-mitsubishi-mr-j4w2-22b/) |

* 1. **Cách lắp đặt hệ thống điều khiển**
     1. **Lắp các module lên Base Q35B**

Ta sẽ tiến hành lắp nguồn vào trước trên Base Q35B, tiếp theo sẽ là Q02HCPU. Đối với những module còn lại, sẽ không có quy tắc nào quyết định về thứ tự. Phần mềm sẽ đọc được những parameters từ Base Q35B và nhận diện được những module thành phần tại từng vị trí với khai báo trước đó.

Nguồn sẽ nhận điện áp từ 100 – 240VAC, nên ta có thể cấp trực tiếp điện 220V vào cục nguồn.



Khi Q02HCPU đã được cấp nguồn, module này sẽ hiện error. Sẽ có 2 trường hợp mà CPU báo lỗi: CPU chưa có chương trình hoặc thứ tự cắm của các module khác với những gì đã khai báo trước đó trong CPU.

* + 1. **Q02HCPU và máy tính thông qua GX Work2**

1. **Q02HCPU**

**Kết nối:**

Để kết nối module PLC Q02HCPU với máy tính, sẽ có nhiều lựa chọn để kết nối: RS-232, Ethernet và USB.

Trong hệ thống điều khiển hiện tại, chúng ta sử dụng giao thức RS-232 để truyền nhận dữ liệu từ máy tính với Q02HCPU.

1. **Máy tính (GX Works2)**

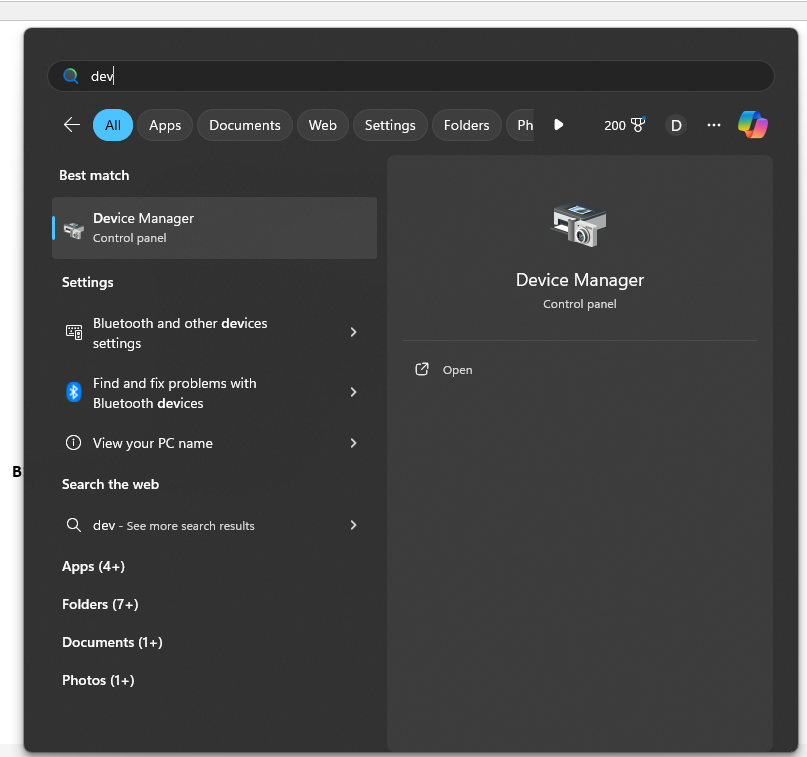
**Kiểm tra kết nối:**

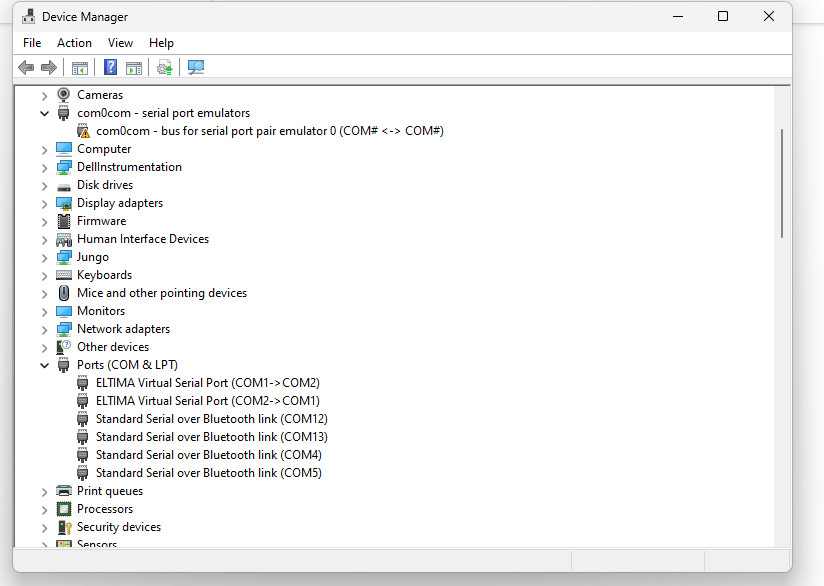
Để kiểm tra kết nối, ta tiến hành thực hiện những bước sau:

**B1: Mở device manager trên máy tính**

Nhấn vào biểu tượng Window trên bàn phím 🡪 Nhập Device manager.

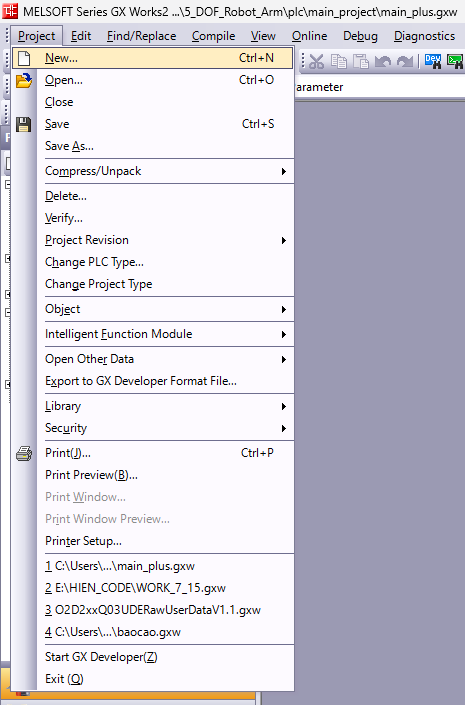
Sau khi mở Device manager 🡪 Đi đến tab Ports 🡪 Hãy thử rút dây kết nối ra và cắm lại. ta sẽ thấy được port sử dụng để kết nối giữa PLC và máy tính sẽ bật/tắt ở trong thư mục port.

****

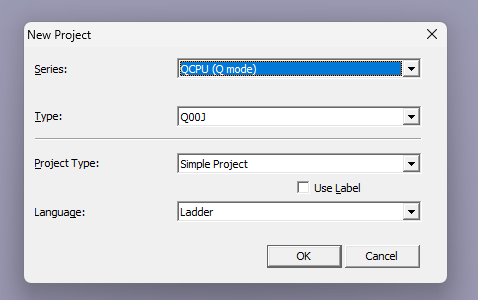


**B2: Kết nối với phần mềm GX Works2**

Sau khi cài đặt xong phần mềm GX Works2 trên máy tính 🡪 Ta có thể sử dụng chương trình có sẵn hoặc tạo mới một project 🡪 Ở trên tab Project trên thanh công cụ 🡪 Chọn New hoặc dùng tổ hợp phím Ctrl + N.



**B3:** **Thiết đặt cho new project:**

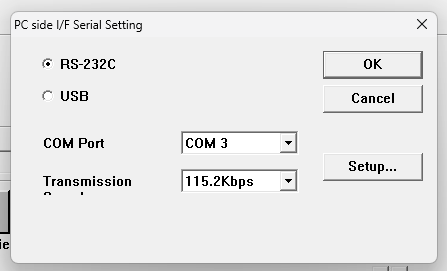


Sau khi có được một new project 🡪 Ta sẽ thử kết nối với Q02HCPU

**B4: Kiểm tra kết nối**

Ở bên tab Navigation 🡪 Ở mục All Connections 🡪 Chọn Connection hiện có.

Click vào icon  🡪 Chọn RS-232C 🡪 Chọn COM Port đang sử dụng theo Device Manager 🡪 Chọn Transmission Baud: 115.2Kbps 🡪 Nhấn OK 🡪 Nhấn Connection Test 🡪 Sẽ có cửa sổ thông báo install successfully nếu kết nối giữa PLC và máy tính được tạo.

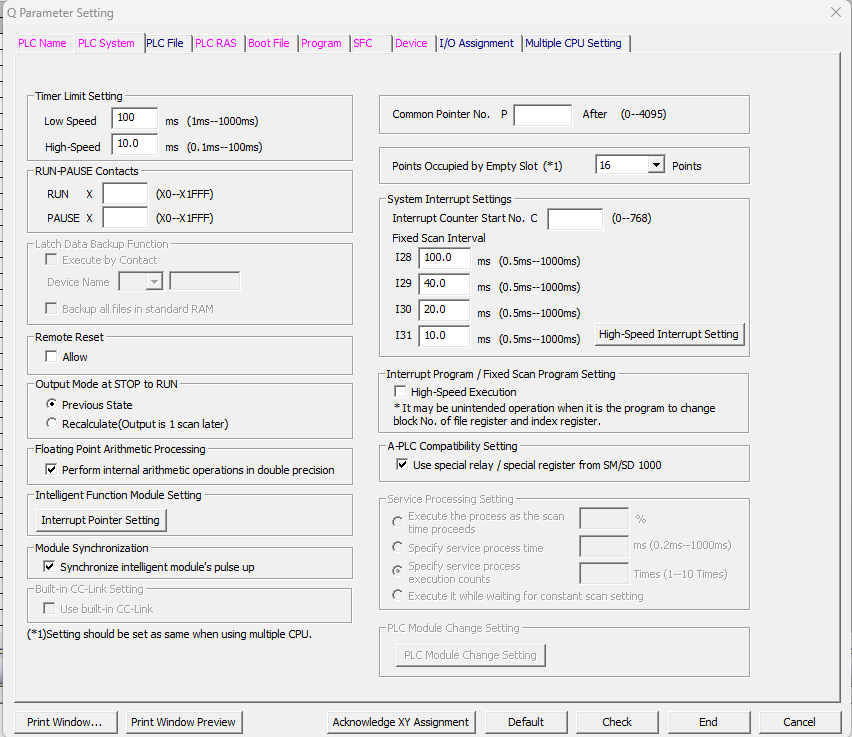


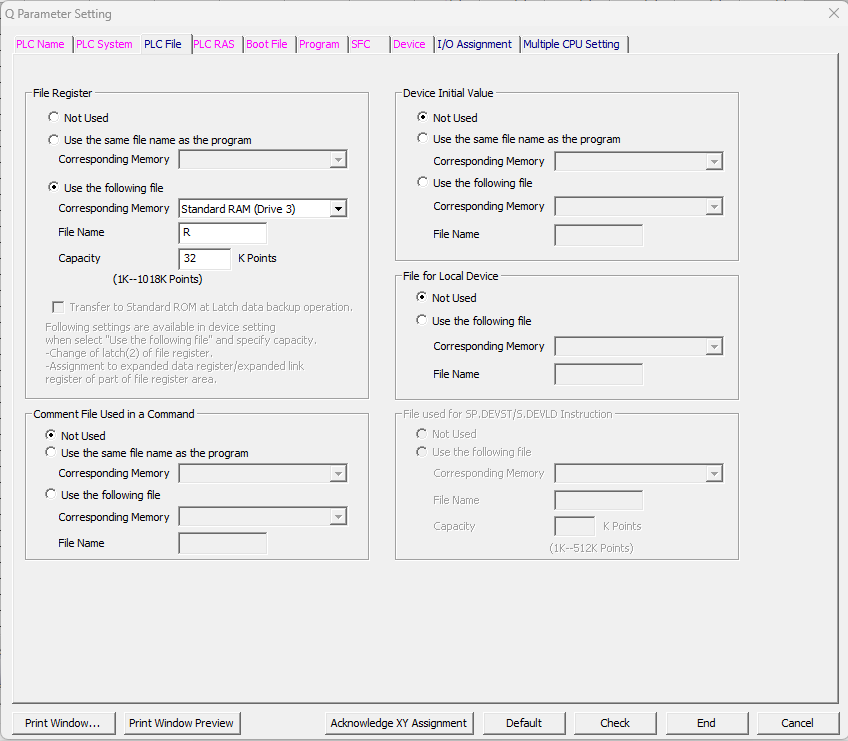


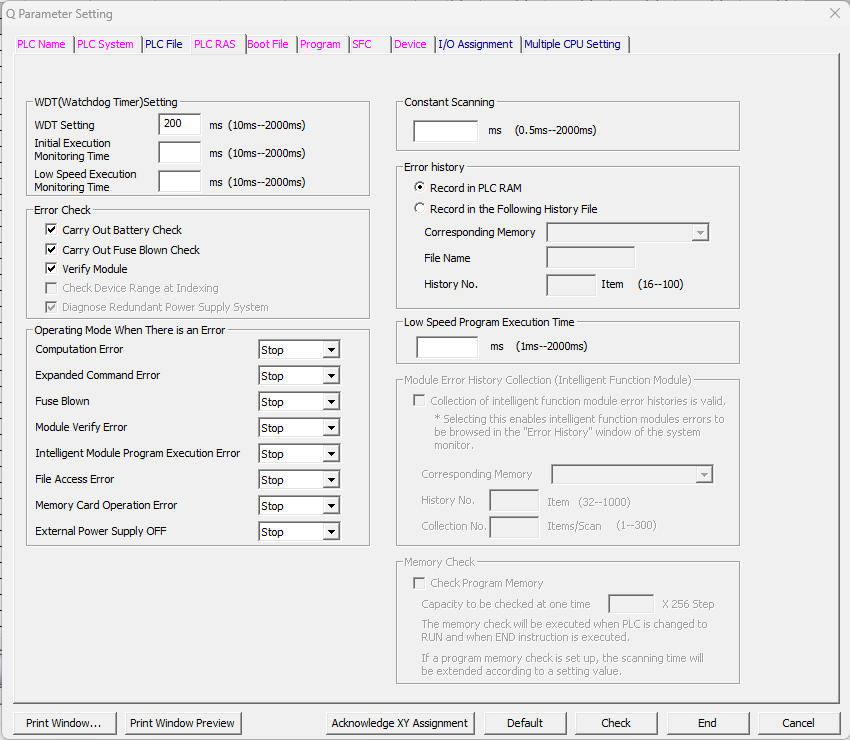
*Lưu ý: Sau khi kết nối thành công, phải nhấn nút OK thì chương trình mới lưu lại kết nối. không thì khi write chương trình xuống PLC sẽ thông báo là không kết nối được.*

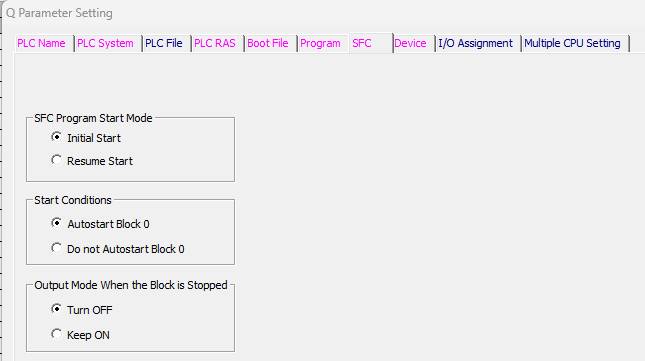
1. **Thiết đặt thông số cho các module trong hệ thống điều khiển trên GX Works2**

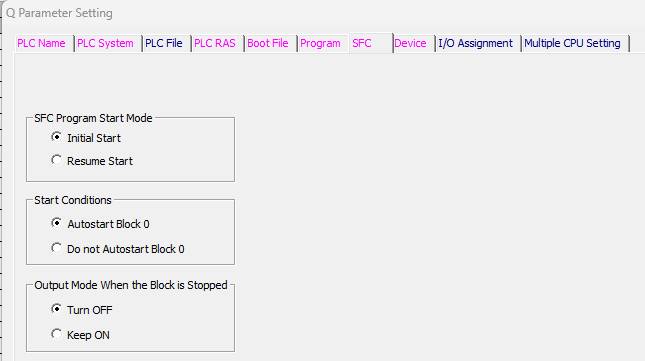
Để thiết đặt các thông số cho những module được lắp đặt trong hệ thống điều khiển, ta có thể tham khảo qua những datasheet do nhà sản xuất cung cấp. Đối với hệ thống điều khiển hiện tại, ta sẽ có những thông số cần setup như sau:

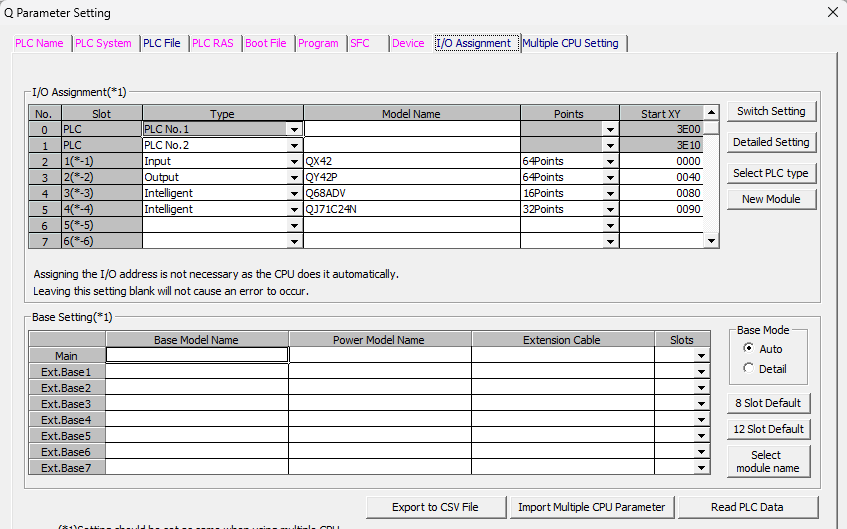


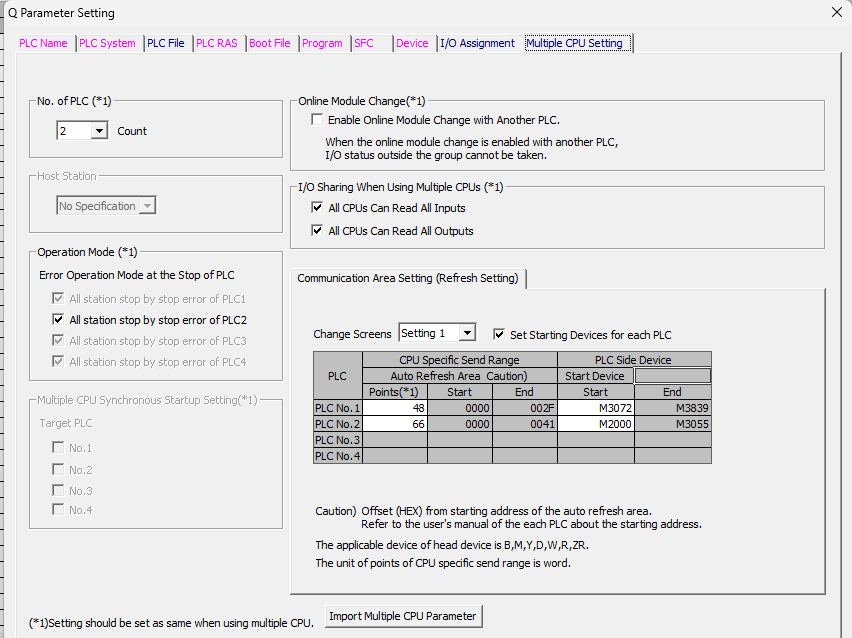












* + 1. **Q02HCPU và QY42P**

Thông qua module nguồn Base Q35B cũng như setting trên GX Works2, ta dễ dàng kết nối giữa 2 module lại với nhau từ đó có thể điều khiển được bật/tắt relays để kích hoạt thắng điện từ giúp đảm bảo tính an toàn cho robot.

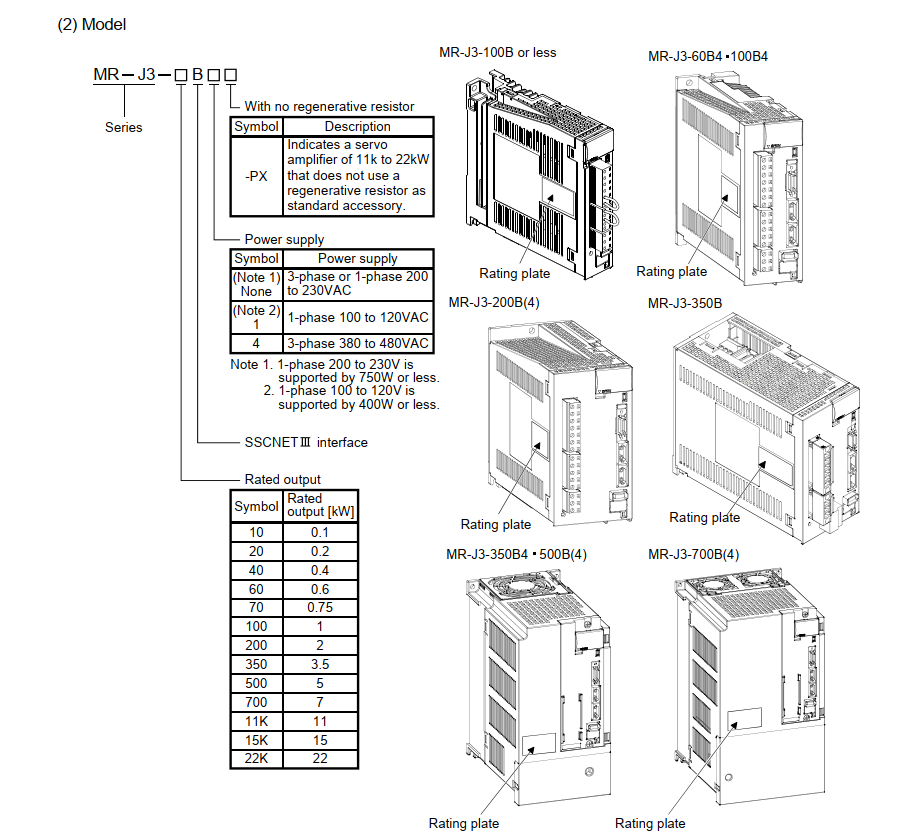
* 1. **Q172HCPU và những thành phần liên quan**
     1. **Q172HCPU và Q02HCPU**

Thông qua module nguồn Base Q35B cũng như setting trên GX Works2, ta dễ dàng kết nối giữa 2 module lại với nhau từ đó Q02HCPU có thể thông qua phần mềm GX Works2 để đưa ra được những tín hiệu điều khiển thông qua việc chia sẻ vùng nhớ cũng như những hàm đặc biệt như SFC SFCS… cho việc điều khiển robot.

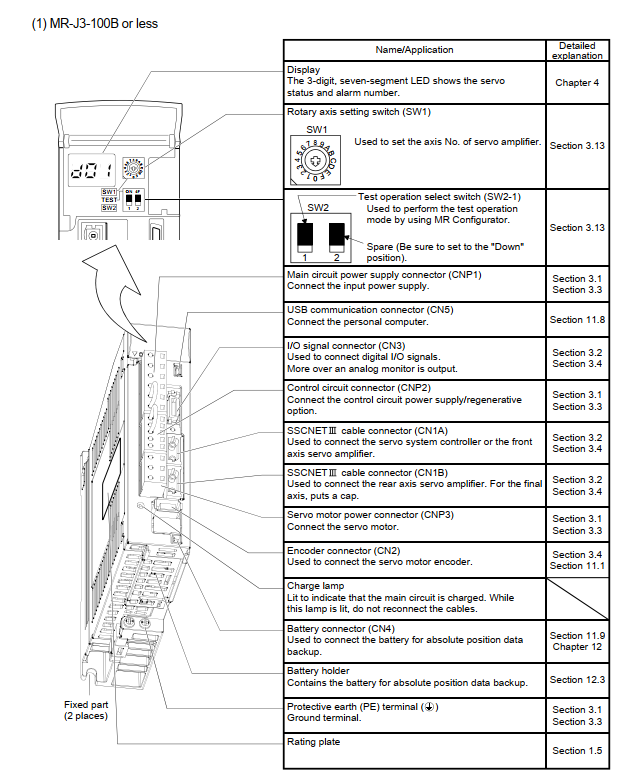
* + 1. **Q172HCPU và Servo motor**

1. **Lắp đặt Servo Amplifier**

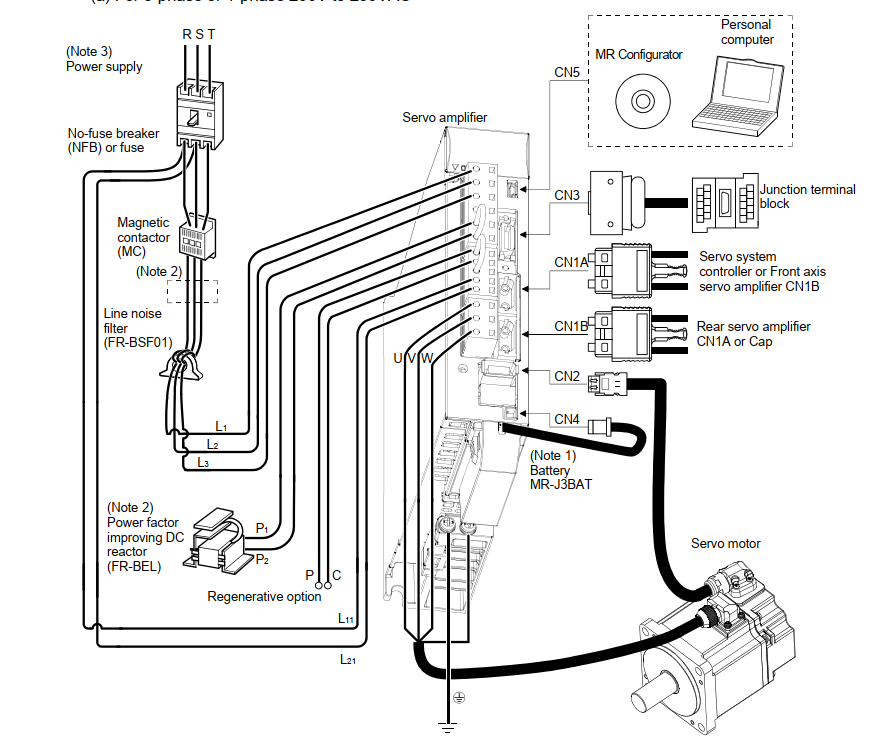
x: Rated input tương ứng với động cơ đang sử dụng



Đối với loại Servo chúng ta đang sử dụng là MR-J[]-[]B, ta có những thông số sau:



*Lưu ý: cần có nguồn điện 3 pha hoặc 1 pha từ 200VAC đến 230 VAC.*



*Nguồn điện từ pin được sử dụng cho hệ thống phát hiện vị trí tuyệt đối ở chế độ điều khiển vị trí*

*Reactor AC đang được được sử dụng. Trong trường hợp này, không thể sử dụng reactor DC. Khi không sử dụng cuộn kháng DC, ngắn mạch P1 và P2.*

*Có thể sử dụng nguồn điện 1 pha 200V đến 230VAC với bộ khuếch đại servo MR-J3-70B trở xuống*

*Đối với 1 pha 200V đến 230VAC, kết nối nguồn điện với L1 L2 và để L3 mở.*

1. **Lắp đặt 2 hoặc nhiều servo amplifier**

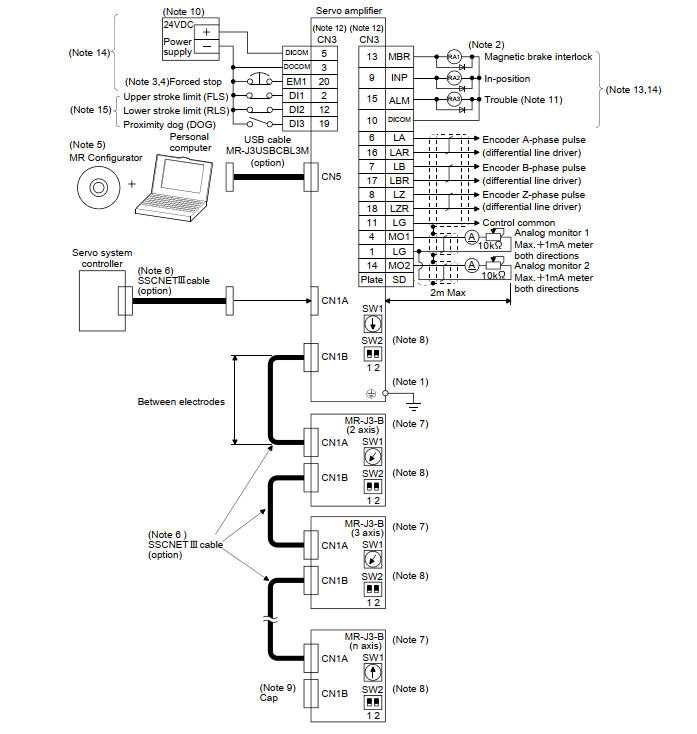
Chừa một khoảng trống lớn giữa mặt trên của bộ khuếch đại servo và bề mặt bên trong của hộp điều khiển, đồng thời lắp quạt làm mát để ngăn nhiệt độ bên trong hộp điều khiển vượt quá điều kiện môi trường.

Khi lắp đặt các bộ khuếch đại servo gần nhau, hãy chừa khoảng cách 1mm giữa các bộ khuếch đại servo liền kề để cân nhắc dung sai lắp đặt.

Trong trường hợp này, hãy điều chỉnh nhiệt độ môi trường trong khoảng từ 0o đến 45o (32o đến 113o) hoặc sử dụng ở tỷ lệ tải hiệu quả 75% hoặc nhỏ hơn.



1. **Cách kết nối tín hiệu I/O giữa 2 hoặc nhiều servo amplifier**





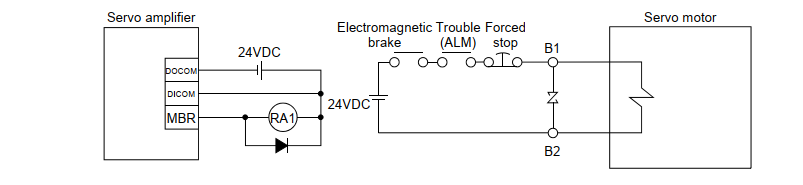
*Lưu ý:*

* *Để tránh bị điện giật, luôn kết nối đầu nối đất bảo vệ (PE) (đầu nối được đánh dấu ) của bộ khuếch đại servo với nối đất bảo vệ (PE) của hộp điều khiển*
* *Kết nối diode theo đúng hướng. Nếu được kết nối ngược lại, bộ khuếch đại servo sẽ bị lỗi và không phát ra tín hiệu, làm mất khả năng dừng cưỡng bức (EM1) và các mạch bảo vệ khác.*

1. **Servo motor với thắng điện từ**

Không dùng chung nguồn điện giao diện 24VDC giữa giao diện và phanh điện từ. Luôn sử dụng nguồn điện được thiết kế dành riêng cho phanh điện từ. Phanh sẽ hoạt động khi nguồn điện (24VDC) tắt. Tắt lệnh bật servo sau khi motor servo đã dừng.

**Sơ đồ kết nối:**



* + 1. **Q172HCPU và máy tính thông qua MT Developer2:**

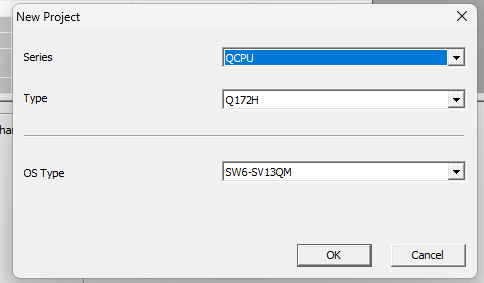
1. **Cách kiểm tra kết nối với PLC**

Giao thức truyền dữ liệu giữa Q172HCPU và máy tính sẽ thông qua USB, nên cơ bản về việc cài đặt phần mềm MT Developer2 sẽ có sẵn driver đã được thiết lập sẵn.

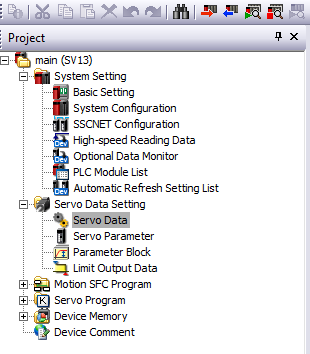
1. **Cài đặt thông số thông qua phần mềm MT Developer2**

Để thiết đặt các thông số cho những module được lắp đặt trong hệ thống điều khiển, ta có thể tham khảo qua những datasheet do nhà sản xuất cung cấp. Đối với hệ thống điều khiển hiện tại, ta sẽ có những thông số cần setup như sau:

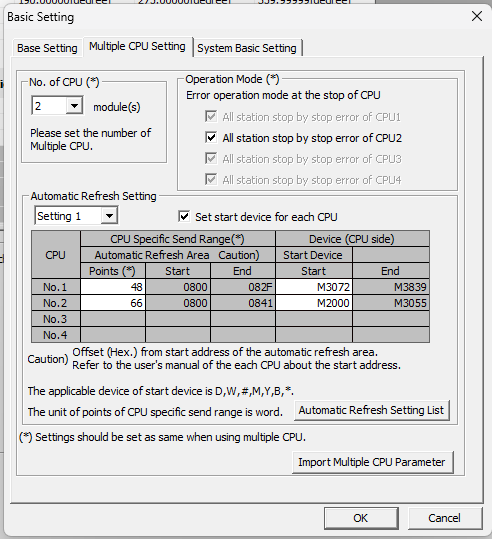
Tạo mới một project:

****

Khi tạo xong project, ta sẽ thấy bên cột **Project** bên trái ngoài cùng, tiến hành thiết đặt lần lượt các thông số sau:



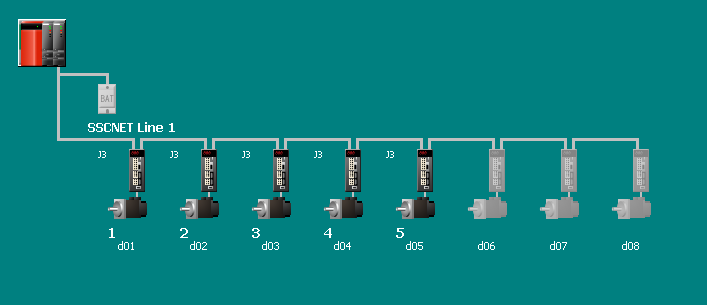
**Basic setting:**



**System configuration:**



**SSCNET Configuration:**

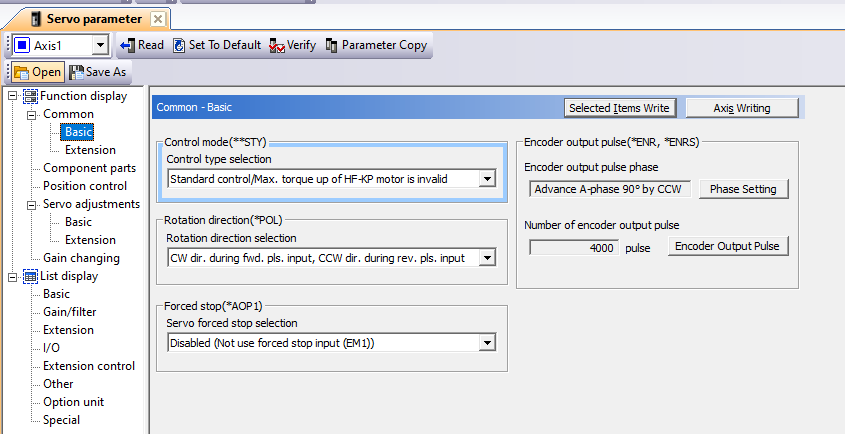


Nháy đúp chuột vào icon động cơ, cửa sổ Amplifier Setting sẽ hiện lên, tiến hành cài đặt thông số như hình:

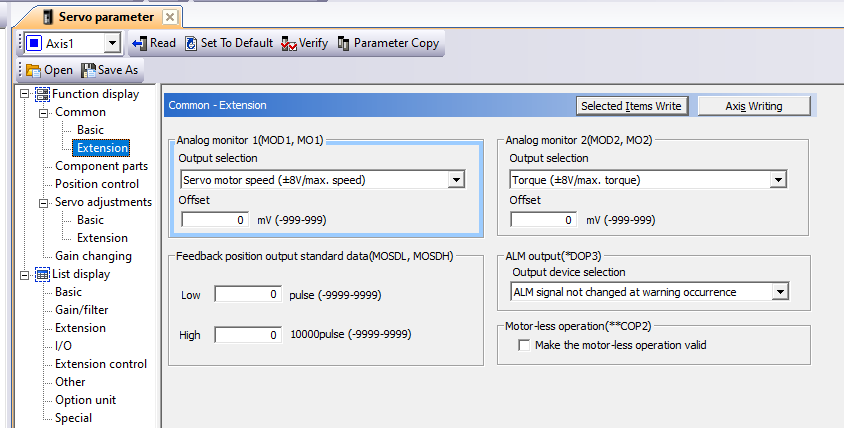


Tiếp tục nhấn vào *Servo Paramenter Setting,* 1 tab Servo parameter sẽ xuất hiện, tiến hành cài đặt với các thông số sau:

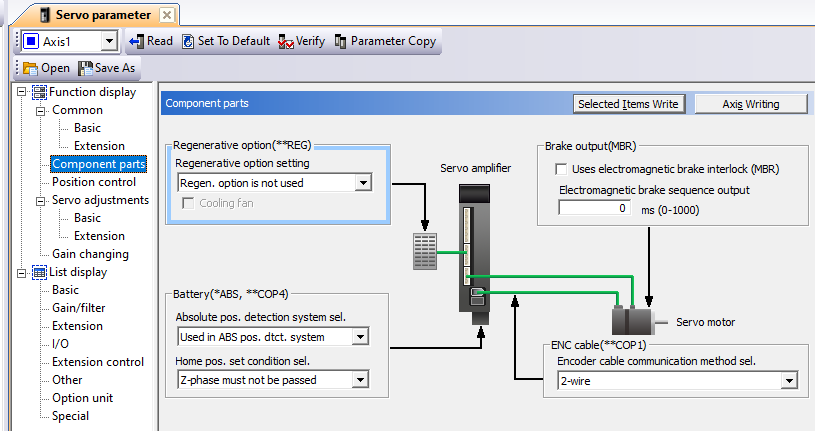
**Function display\Common\Basic:**



**Function display\Common\Extension:**

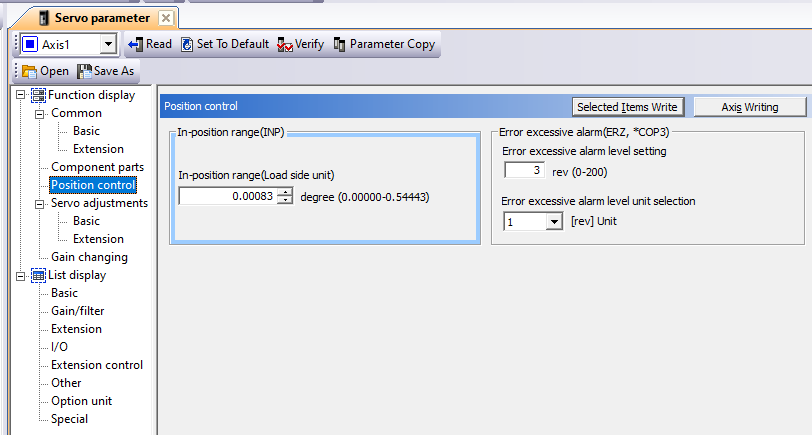


**Function display\Component parts:**

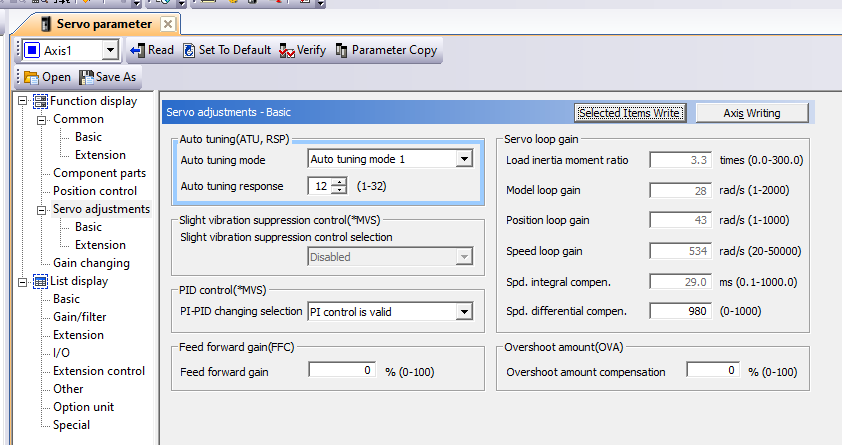


**Function display\Position control:**

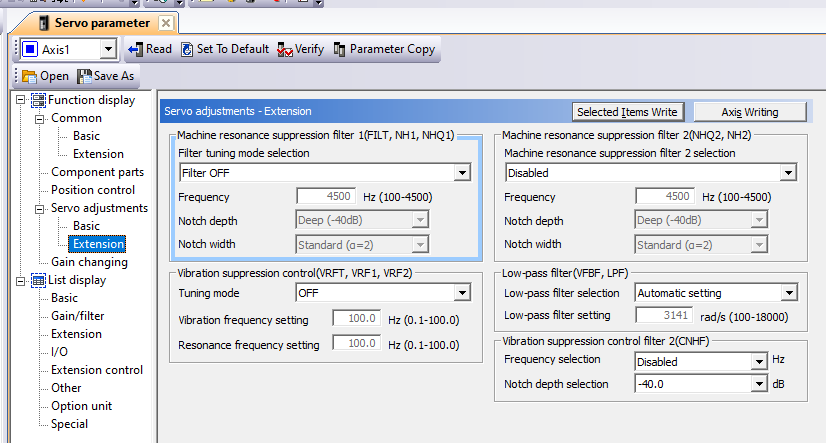
*Đây là phần thiết lập cho Axis 1, tương tự với 4 Axis còn lại, ta có thể dựa vào file tham khảo để thiết đặt theo cho đúng.*



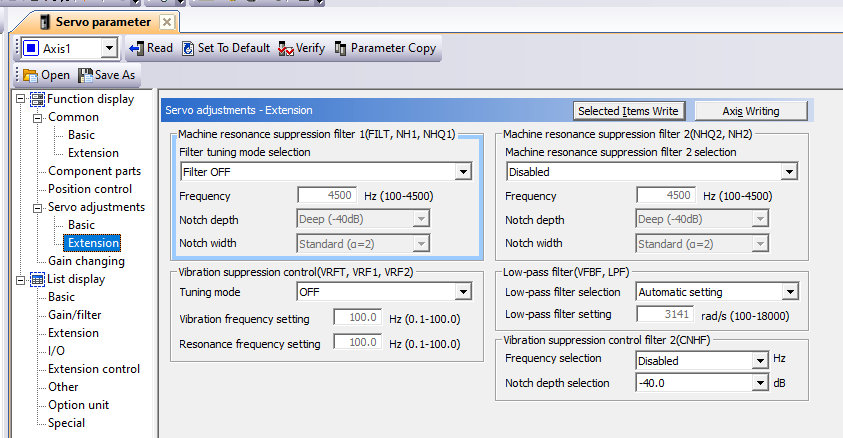
**Function display\Servo adjustments\Basic:**



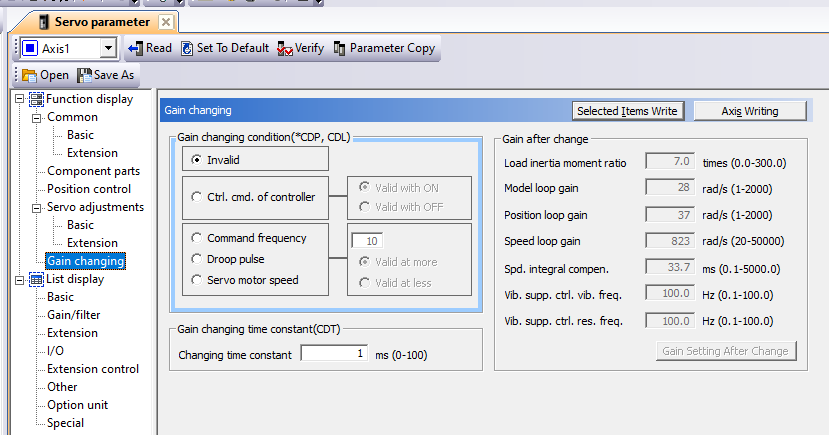
**Function display\Servo adjustments\Basic:**

****

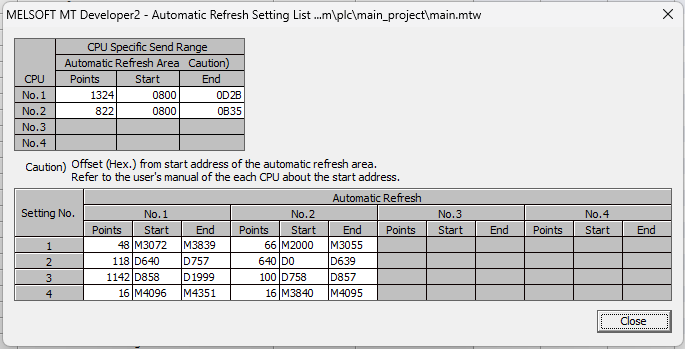
**Function display\Servo adjustments\Extension:**

****

**Function display\Gain changing:**

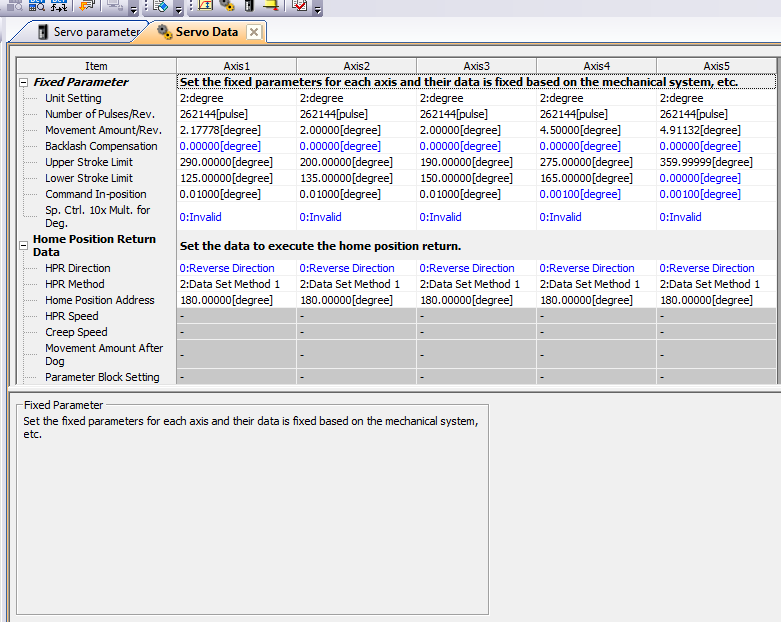
****

Sau khi cài đặt hết thông số, ta vô tab *Automatic Refresh Setting List* bên cây thư mục Project, thiết đặt vùng nhớ được chia sẻ với Q02HCPU như cấu hình sau:



Tất cả những bước nêu trên đã hoàn thành cho việt thiết đặt cấu hình cho *System Setting,* tiến hành cài đặt cấu hình cho phần *Servo Data Setting*

**Servo Data:**



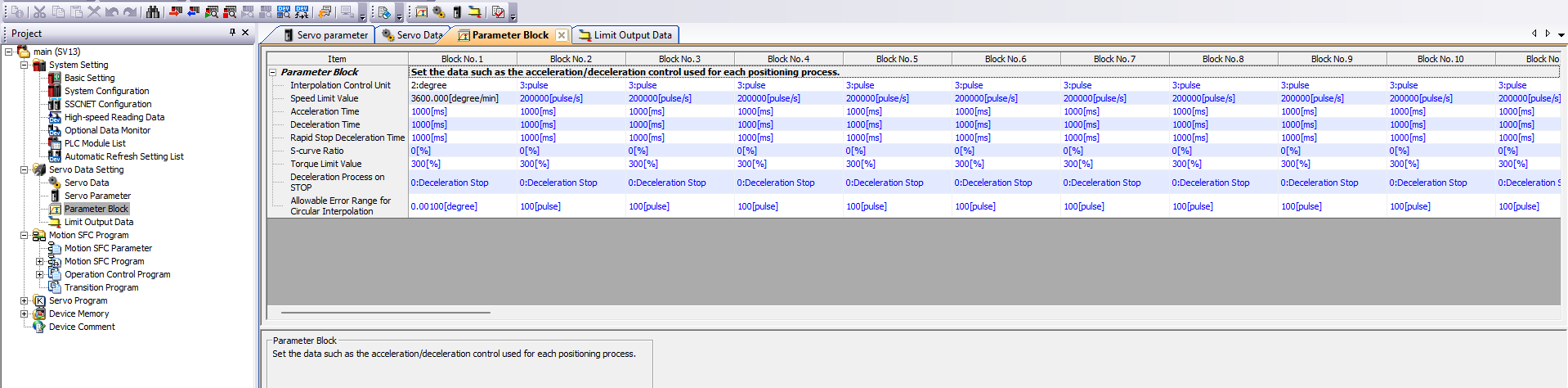
*Lưu ý:*

*Home Position Address* *đang được để là 180.00000 [degree] sẽ là vị trí ban đầu cho các khớp.*

*Unit Setting đang được thiết đặt là degree quyết định đơn vị mà ta sẽ điều khiển cho robot.*

**Parameter Block:**

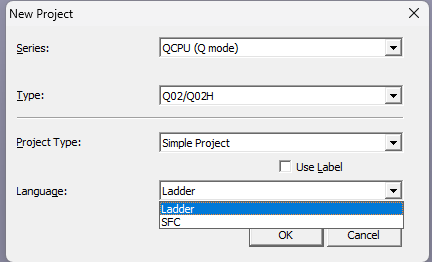
Thiết đặt các thông số liên quan đến tốc độ giới hạn và thời gian tăng tốc giảm tốc cho từng block, tiến hành cài đặt các thông số như hình bên dưới.



1. **Lập trình điều khiển cho robot thông qua phần mềm MT Developer2 và GX Work2**
   1. **Lập trình điều khiển robot trên GX Work2**

Tương tự như Tia Portal cho những dòng PLC Siemen, với mục đích chính để lập trình cho PLC hoạt động đúng với chức năng mong muốn từ phía người dùng. Dòng PLC Mitsubishi cũng được hãng đưa ra 1 phần mềm GX Work2 để có thể tiện lợi cho người sử dụng thiết lập các chương trình tải xuống PLC để thực thi những câu lệnh của người lập trình để điều khiển các thiết bị ngoại vi.

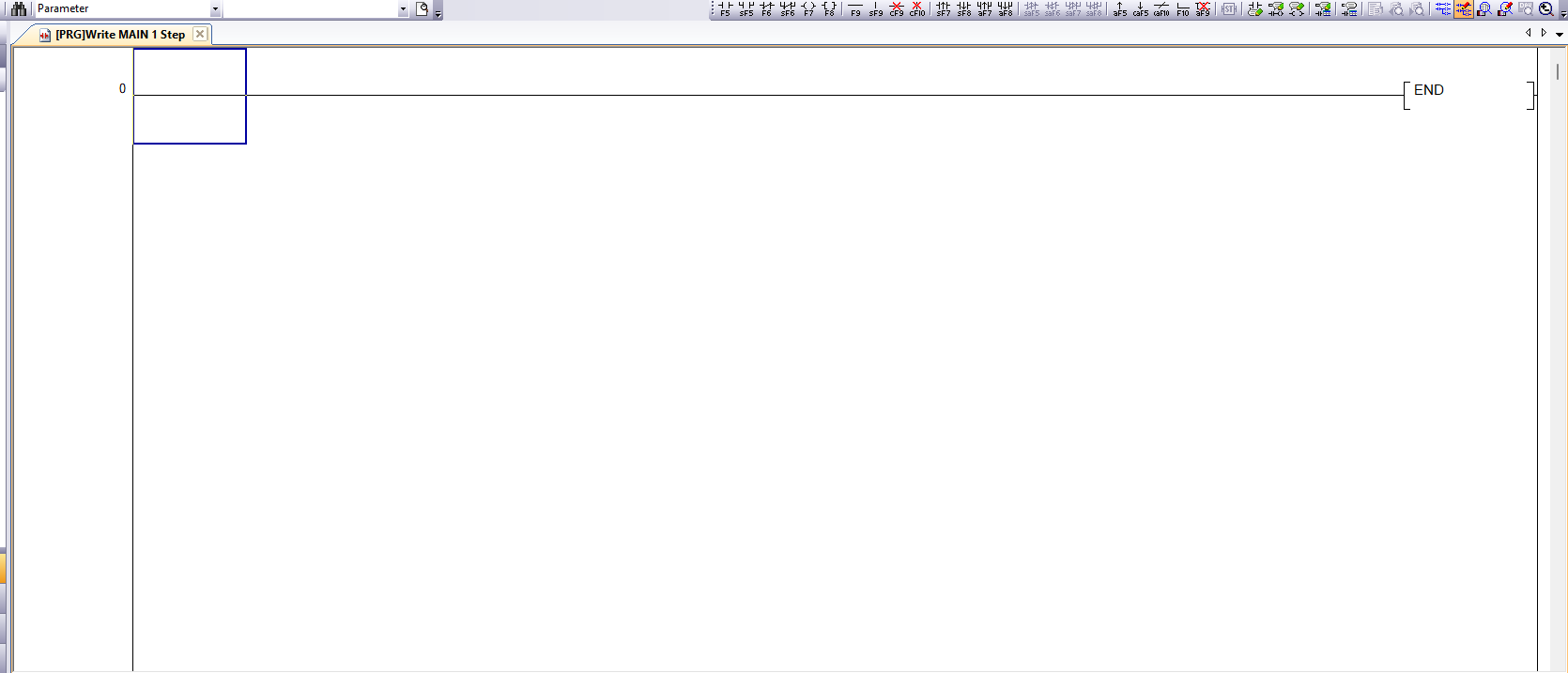
Ở lúc tạo new project, sẽ có 2 lựa chọn cho việc lập trình trong GX Work2: Ladder và SFC



**Ladder:** Đây là một ngôn ngữ lập trình đồ họa nhằm thể hiện các hoạt động logic thông qua các ký hiệu tượng trưng. Ladder Logic được cấu tạo từ các nấc thang logic, tạo thành một cấu trúc trông giống như cái thang, do đó có tên gọi là “Ladder”.

**SFC:** Viết tắt của cụm từ tiếng anh “Sequential Function Charts” tạm dịch là “Biểu đồ chức năng tuần tự”, bạn sẽ sử dụng các bước và quá trình chuyển đổi để đạt được kết quả cuối cùng của mình. SFC không phải là một ngôn ngữ, mà là một phương tiện đồ họa nhằm phân vùng mã và hiển thị trạng thái hoặc chế độ hoạt động một cách trực quan.

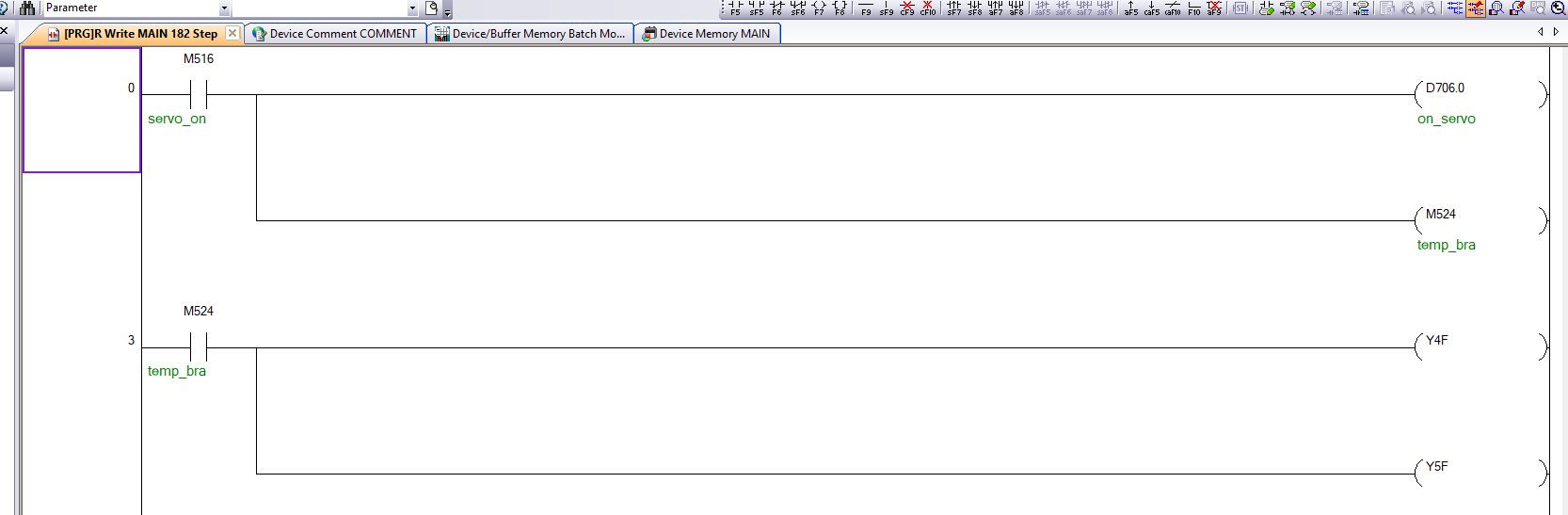
Trong chương trình hiện tại, ta chọn ngôn ngữ lập trình **Ladder** như là một phương thức quen thuộc để lập trình cho PLC.



* + 1. **Chương trình điều khiển robot**

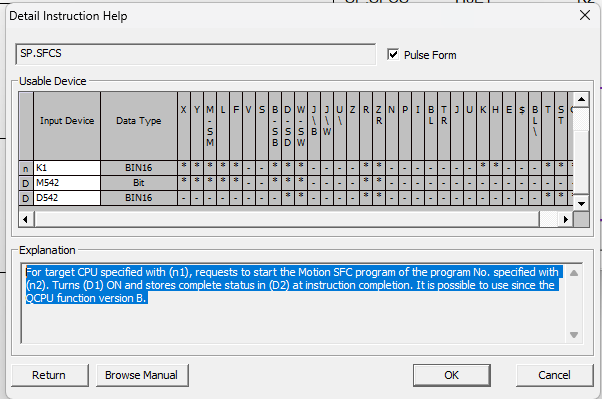
1. **Bật tắt Servo**

Ta sẽ có những vùng nhớ được chia sẻ giữa Q02HCPU và Q172HCPU. Vùng nhớ **D706.0** được thiết đặt cho việc bật tắt servo. Và khi đã bật Servo, đồng nghĩa với việc ta phải nhả phanh điện từ ra khỏi động cơ với 2 ngõ ra **Y4F** và **Y5F**.



1. **Các chương trình điều khiển cơ bản**

Khi đã hoàn tất cấu hình cho module Q02HCPU và Q172HCPU, ta có thể kết nối chúng bằng việc dùng chung các vùng dữ liệu với nhau. Đồng thời, GX Work2 cũng hỗ trợ hàm **SFCS** để có thể gọi chương trình bên MT Developer để thực hiện. Để rõ hơn thì ta có thể tra cứu trong datasheet do nhà sản xuất cung cấp



Cấu trúc của một hàm SP.SFCS được thể hiện như sau:



**H3E1:** Là vị trí của Q172HCPU được thể hiện dưới dạng mã hex.

**K[x]:** Là chương trình x được tạo ra trong phần mềm MT Developer2.

**M[x]:** Là relay ảo được bật lên lưu status vào vùng nhớ **D[x]** được cấu hình từ phía người dùng.

Tương ứng với từng chương trình, ta sẽ dùng lệnh này để gọi lên các chương trình SFC được tạo ở bên MT Developer2 để phục vụ cho từng mục đích khác nhau:

|  |  |
| --- | --- |
| **Định nghĩa** | **Tên chương trình** |
| Set home | K0 |
| Go home | K1 |
| MoveJ | K2 |
| Movepat | K3 |
| Adapt Control | K6 |

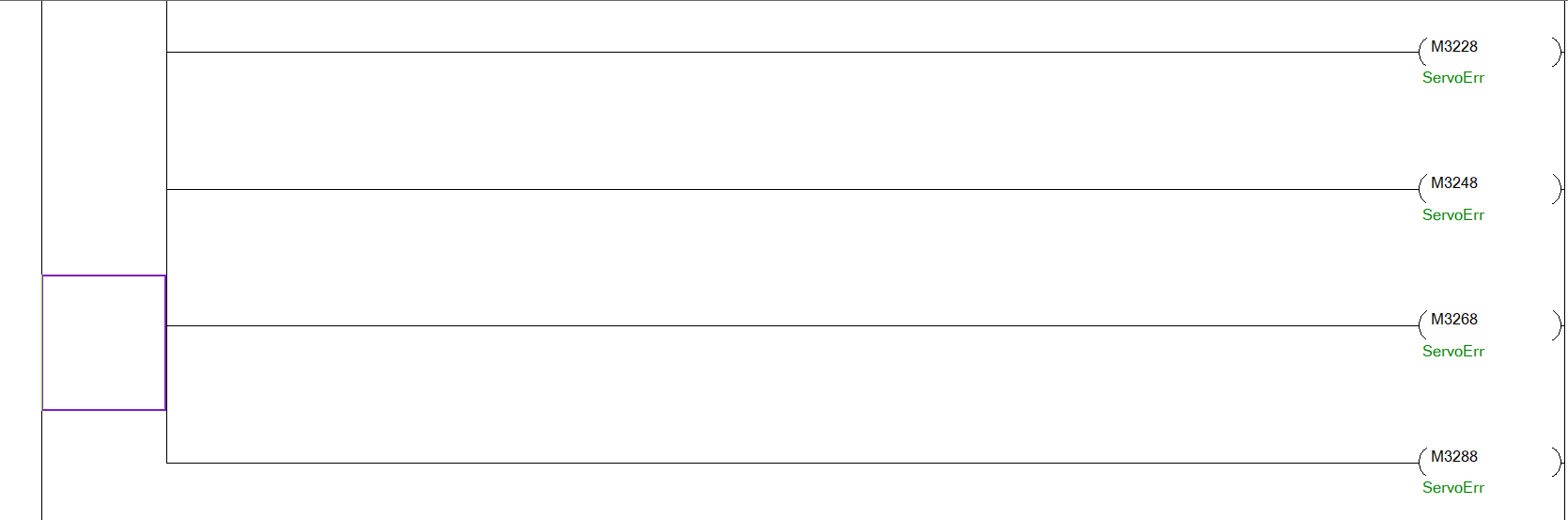
1. **Reset error**

Trong quá trình điều khiển, sẽ không thể tránh khỏi những lỗi như: ngoài vùng hoạt động, low battery… và nhiều lỗi khác nữa. Khi gặp lỗi, module sẽ tự động bật lên và ngắt khả năng điều khiển từ bên phía phần mềm. Do vậy, việc khởi tạo lại trạng thái ban đầu, nhà sản xuất đã cung cấp những relay đặc biệt để thực hiện chức năng đó.

Có 2 cách để xem relay phục vụ cho chức năng gì: Kiểm bằng MT Developer2 và kiểm tra thông qua datasheet do nhà sản xuất cung cấp.

Trong hệ thống điều khiển, những relay dưới đây sẽ thực hiện reset error một số lỗi như sau:

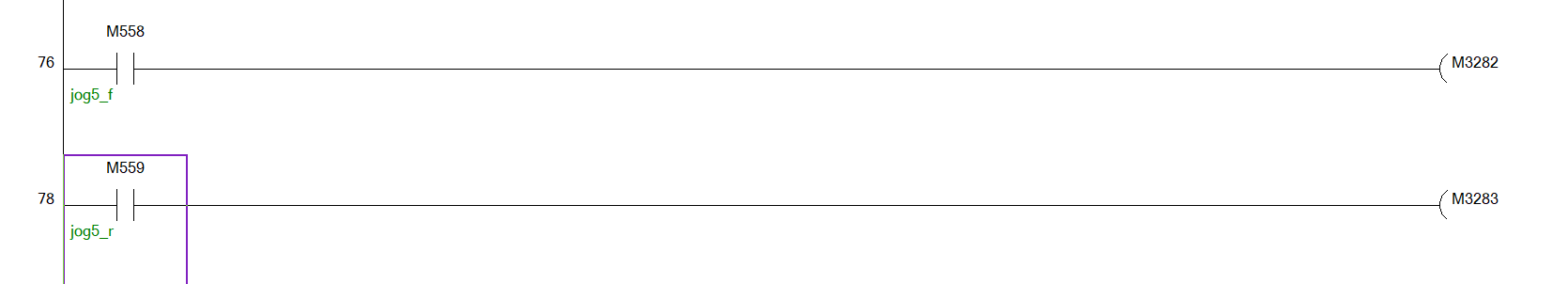




1. **Chạy jogging**

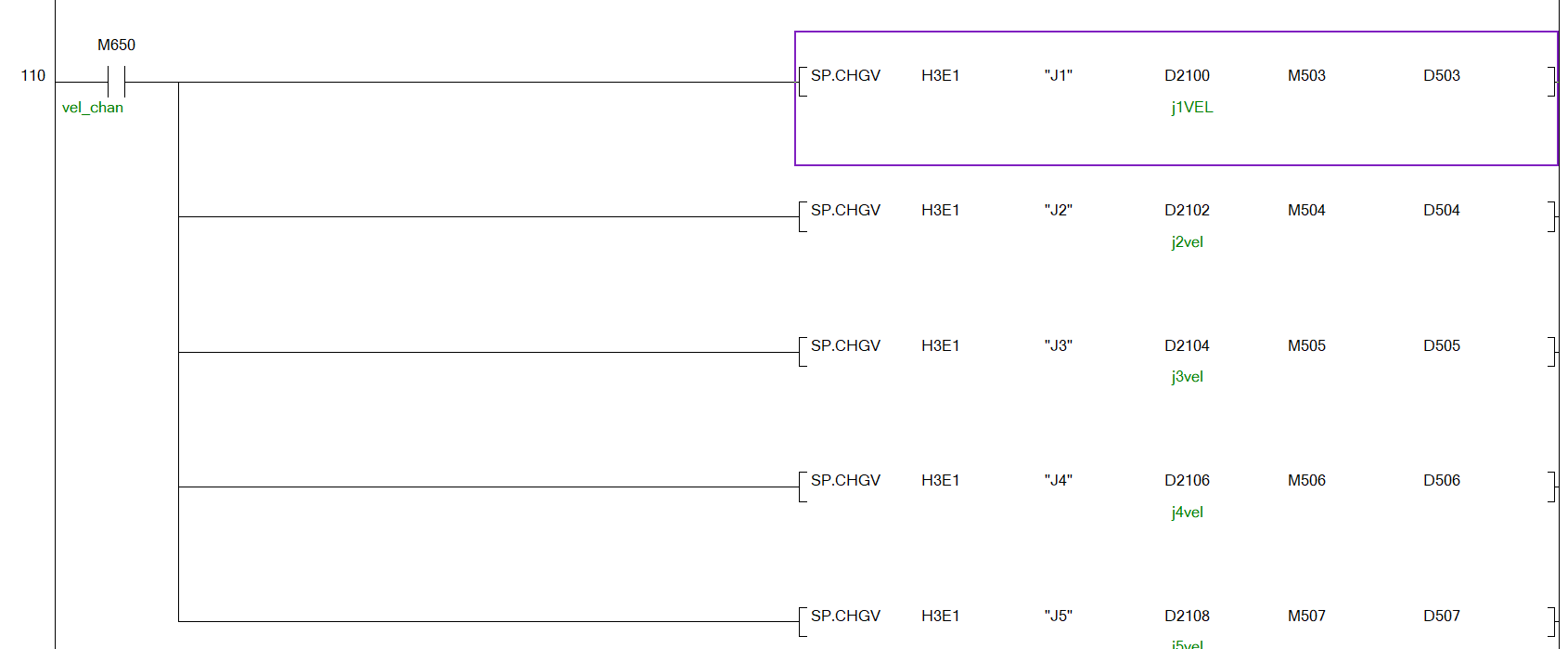
Ngoài ra, sẽ có những relay đặc biệt sẽ trigger thẳng trực tiếp đến servo motor để chạy jog cho từng động cơ.





1. **Thay đổi vận tốc tức thời**

Trong quá trình điều khiển cho một chương trình trong MT Developer2, ta sẽ phải định nghĩa từ đầu cho vận tốc của nó và thiết đặt vị trí cho từng khớp mong muốn và thực hiện bằng câu lệnh bằng SFCS để gọi hàm đó thực thi. Song, ta cũng có những vùng nhớ đặt biệt để thay đổi vận tốc của động cơ. Nhà sản xuất cũng cung cấp hàm **CHGV** để giúp thực thi điều đó:



1. **Quan sát hoàn thành chương trình**

Trong quá trình thực hiện 1 chương trình nào đó. Sẽ có những relay đặc biệt được bật lên để quan sát quá trình thực thi của động cơ, khi hoàn tất, những relays này sẽ được bật lên. Trong điều khiển, ta có thể sử dụng điều đó để kiểm tra xem rằng Robot đã thực hiện xong quy trình hay chưa:

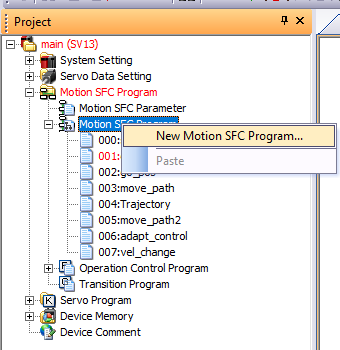
****

* 1. **Lập trình điều khiển robot trên MT Developer2**

Sau khi cấu hình xong cho module Q172HCPU, ta tiến hành lập trình cho các chương trình thực thi. Có 2 cách để tạo 1 chương trình: Tạo trực tiếp bằng **SFC program** hoặc tạo từng Servo Program và tổng hợp lại vào SFC program. Thường thì cách thứ nhất sẽ được sử dụng nhiều hơn do khả năng linh hoạt cũng như tiết kiệm thời gian.

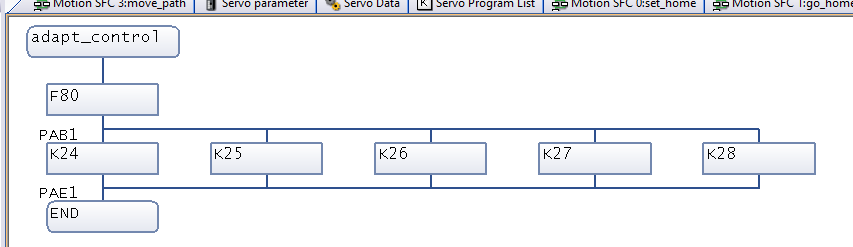
SFC Program đã được đề cập trên phần mềm GX Work2 ở phần các chương trình điều khiển cơ bản. Tương ứng với số chương trình trong phần mềm MT Developer2, sẽ là số chương trình nằm trong **K[x]** trong hàm SFCS của GX Work2.

**B1:** Tại tab **Motion SFC Program** ở thanh **Project,** ta xổ xuống thì sẽ thấy ở thư mục con **Motion SFC Program** 🡪 Chuột phải và chọn **New Motion SFC Program** như hình sau:



**B2:** Thiết lập cấu trúc chương trình

Về cấu trúc chương trình ta sẽ phải bắt đầu bằng 1 block **với tên được tạo** và kết thúc bằng 1 block **END.**



Về cấu trúc của 1 chương trình, ta sẽ thường dùng các block **F[x]** và **K[x]** để thực hiện các tác vụ:

**F[x]:** Là block để thực hiện các chức năng gán giá trị cho vùng nhớ.

**K[x]:** Là block dùng để thực hiện các hàm chức năng giúp trong việc di chuyển robot.

* + 1. **Các chương trình trong hệ thống điều khiển**

1. **set\_home**

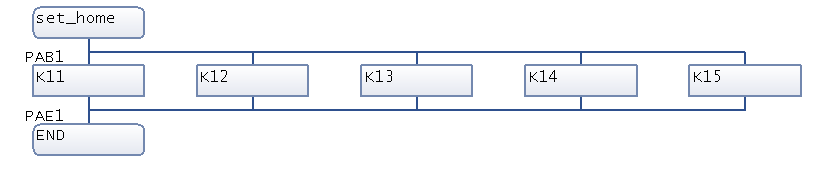
Trên thực tế, các dòng motor servo của các hãng Mitsubishi, ABB đều có khả năng lưu lại vị trí. Nhưng chỉ khi các servo module đều có nguồn dự phòng. Trong trường hợp có vấn đề với nguồn dự phòng, dữ liệu vị trí được lưu sẽ mất. Vì vậy, việc thiết đặt lại vị trí ban đầu cho robot là cần phải có trong hệ thống điều khiển.

MT Developer2 cho phép người dùng sử dụng lệnh để cài đặt lại vị trí ban đầu cho robot.



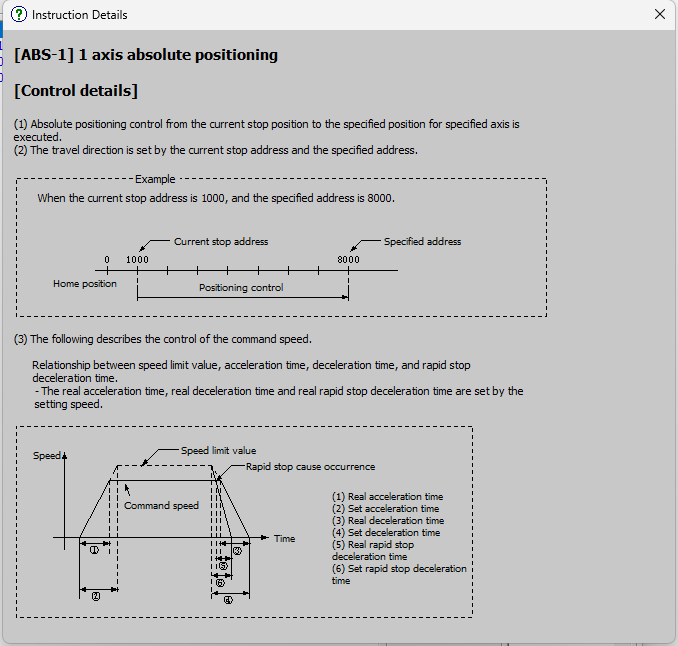
Lấy ví dụ như chương trình K11, ta dùng lệnh **ZERO** 🡪 **Axis [x]** với **x** là trục/khớp mà ta muốn thiết đặt vị trí ban đầu cho robot.

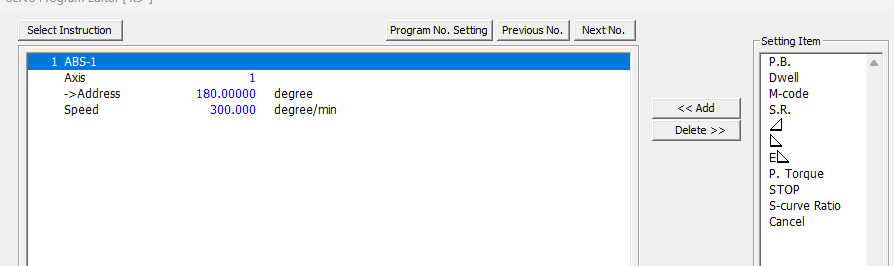
Thiết lập tương tự cho các Axis/khớp 2 🡪 5 tương ứng với K12 🡪 K15, ta có được một chương trình SFC hoàn chỉnh phục vụ cho việc Set home như sau:



1. **go\_home**

Ta dùng lệnh ABS-1 (Axis abosulte positioning) để điều khiển cho từng khớp đến vị trí tuyệt đối. Ta có thể tra trên datasheet của nhà sản xuất hoặc có thể xem trong tab Instruction Details.



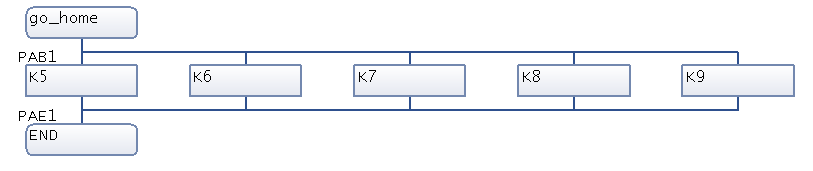


Hàm ABS-1 có rất nhiều thành phần cho phép điều khiển mà ta có thể thêm vào (Add) cũng như xóa (Delete) nằm ở bên ô **Setting Item.** Song để điều khiển được robot đến vị trí mong muốn, cơ bản ta chỉ cần yếu tố là **vị trí (Address)** và **vận tốc (Speed)**.

**Address:** Là vị trí mà ta mong muốn robot tới. Như đã nêu ở phần lưu ý trên phần thiết đặt, vị trí ban đầu của robot sẽ là 180.00000 degree.

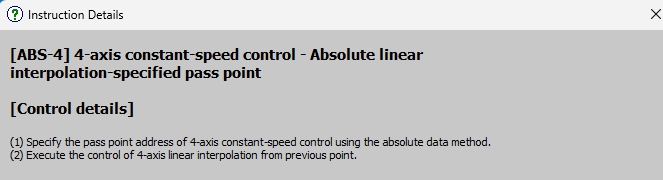
**Speed:** Với đơn vị là degree, thì ta cũng sẽ có thể chọn đơn vị vận tốc trong MT Developer2 là degree/min.

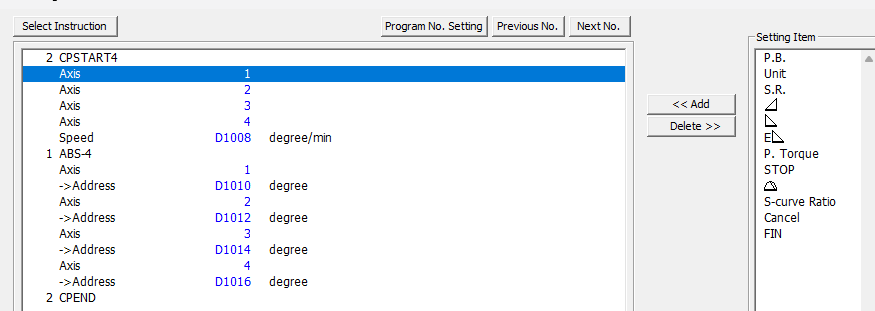
Tương ứng cho các trục/khớp từ 2 🡪 5 , ta thay đổi ở phần Axis và chọn vận tốc tương ứng với giá trị Address không đổi. Ta có 1 chương trình hoàn thiệt như sau:



1. **go\_pos**

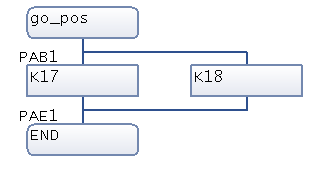
Ngoài ra với việc chia sẻ được vùng nhớ với Q02HCPU, ta có thể dựa vào đó mà đưa vào các giá trị góc mong muốn để có thể điều khiển được robot đến vị trí mình mong muốn bằng việc thay đổi giá trị **Address** trong các hàm bằng các vùng nhớ có thể thay đổi được vận tốc và vị trí với sự kết hợp sử dụng hàm ABS-4 (Axis constant speed control).





***Lưu ý:*** *Address sẽ cần 32 bit để lưu giá trị. Ta cần phải có 2 vùng nhớ 16 bit. Như hình trên, với việc khai báo D1010 thì mặc định hàm sẽ đọc từ D1010 -> D1011 để lưu vị trí vào module servo để điều khiển robot. Tương tự với các Axis còn lại*

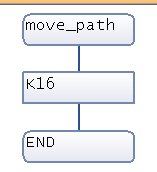
Nhưng với việc robot trong điều khiển đang có 5 bậc tự do, trong khi hàm ABS-4 chỉ hỗ trợ 4 trục. Ta có thể giải quyết vấn đề này bằng cách kết hợp thêm hàm ABS-1 để điều khiển cho khớp còn lại. Chương trình hoàn chỉnh như sau:



1. **move\_path**

Ngoài ra bằng việc kết hợp các chuỗi hàm ABS-4, ta có thể điều khiển được vị trí và vận tốc di chuyển tuân theo quỹ đạo mà ta quy định. Trong chương trình điều khiển, quỹ đạo đang được chia thành 100 điểm tải xuống robot để điều khiển.

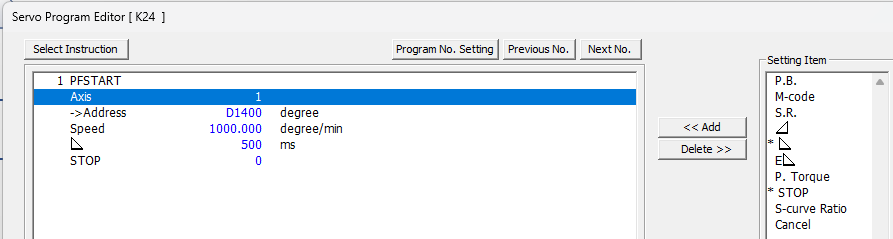
Ta có chương trình hoàn thiện như hình:





1. **adapt\_control**

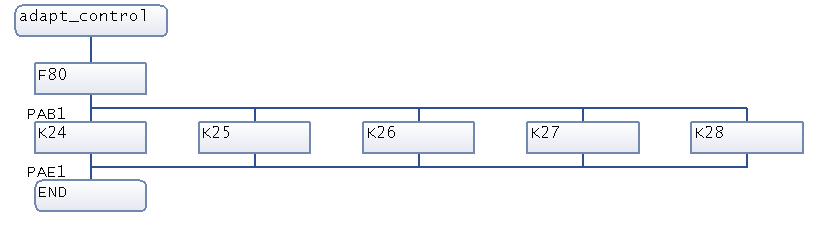
Ta cũng sẽ có những vùng nhớ đặc biệt giúp cập nhật vị trí muốn đến một cách tức thời cho robot. Bằng việc truy cập vào chúng, module servo sẽ có thể nhận diện và đáp ứng vị trí sau một khoảng thời gian mà ta cài đặt



Như trên hình, vùng nhớ D1400 là vùng nhớ đặc biệt dành cho trục 1, ta có thể cài đặt vận tốc mong muốn cũng như thời gian đáp ứng sao cho tương thích.

*Lưu ý: Chế độ adapt control sẽ được bật khi ta phải thiết đặt lại cờ PFSTART bằng việt kích hoạt relay ảo M3200 thì mới hoạt động được.*

Chương trình hoàn chỉnh của adapt control:



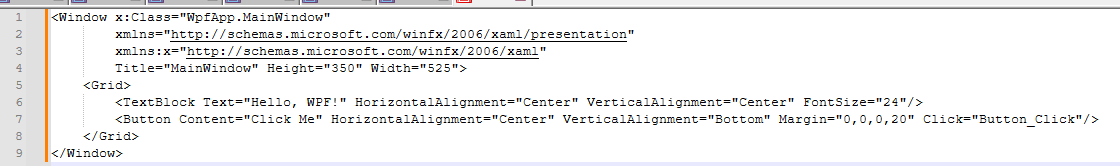
**Chương 8: Thiết kế, lập trình giao diện hệ thống với WPF**

WPF (Windows Presentation Foundation) là lựa chọn phù hợp cho việc điều khiển robot nhờ vào khả năng tạo ra giao diện người dùng hiện đại, trực quan và thân thiện. Trong công nghiệp, WPF đóng vai trò quan trọng do khả năng tích hợp dễ dàng với các hệ thống và công nghệ khác của Microsoft, hỗ trợ đồ họa mạnh mẽ 2D và 3D, giúp hiển thị chi tiết các mô hình và dữ liệu robot. Với khả năng vượt trội, WPF đảm bảo kết nối dữ liệu giữa giao diện và logic ứng dụng hiệu quả, hỗ trợ cập nhật thời gian thực và tách biệt rõ ràng giữa các thành phần. Ngoài ra, WPF cung cấp nhiều control mạnh mẽ như Charts, Graphs, DataGrids để hiển thị dữ liệu cảm biến và trạng thái robot, đồng thời tích hợp dễ dàng các giao thức truyền thông như TCP/IP, UDP, Serial Communication để giao tiếp với robot. Với tính tùy biến cao, WPF cho phép thiết kế giao diện phù hợp với yêu cầu cụ thể của ứng dụng công nghiệp, đảm bảo hiệu suất cao và khả năng mở rộng khi cần thiết.

1. **Cấu trúc của WPF**

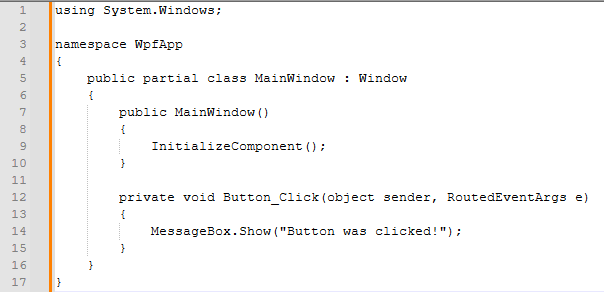
Một chương trình WPF cơ bản thường bao gồm hai phần chính: tệp XAML và tệp code-behind. Dưới đây là cấu trúc của một chương trình WPF:

**XAML (.xaml):** định nghĩa giao diện người dùng của ứng dụng. Đây là nơi bạn khai báo các thành phần UI như buttons, textboxes, labels…



* **Window**: Định nghĩa cửa sổ chính của ứng dụng WPF.
* **xmlns**: Khai báo namespace để sử dụng các thành phần WPF.
* **Grid**: Container layout để sắp xếp các thành phần UI.
* **TextBlock**: Hiển thị văn bản tĩnh.
* **Button**: Nút bấm, với sự kiện Click được liên kết tới phương thức Button\_Click trong file code-behind.

**Code-behind (.xaml.cs):** chứa logic ứng dụng và sự kiện xử lý cho giao diện người dùng được định nghĩa trong file XAML.



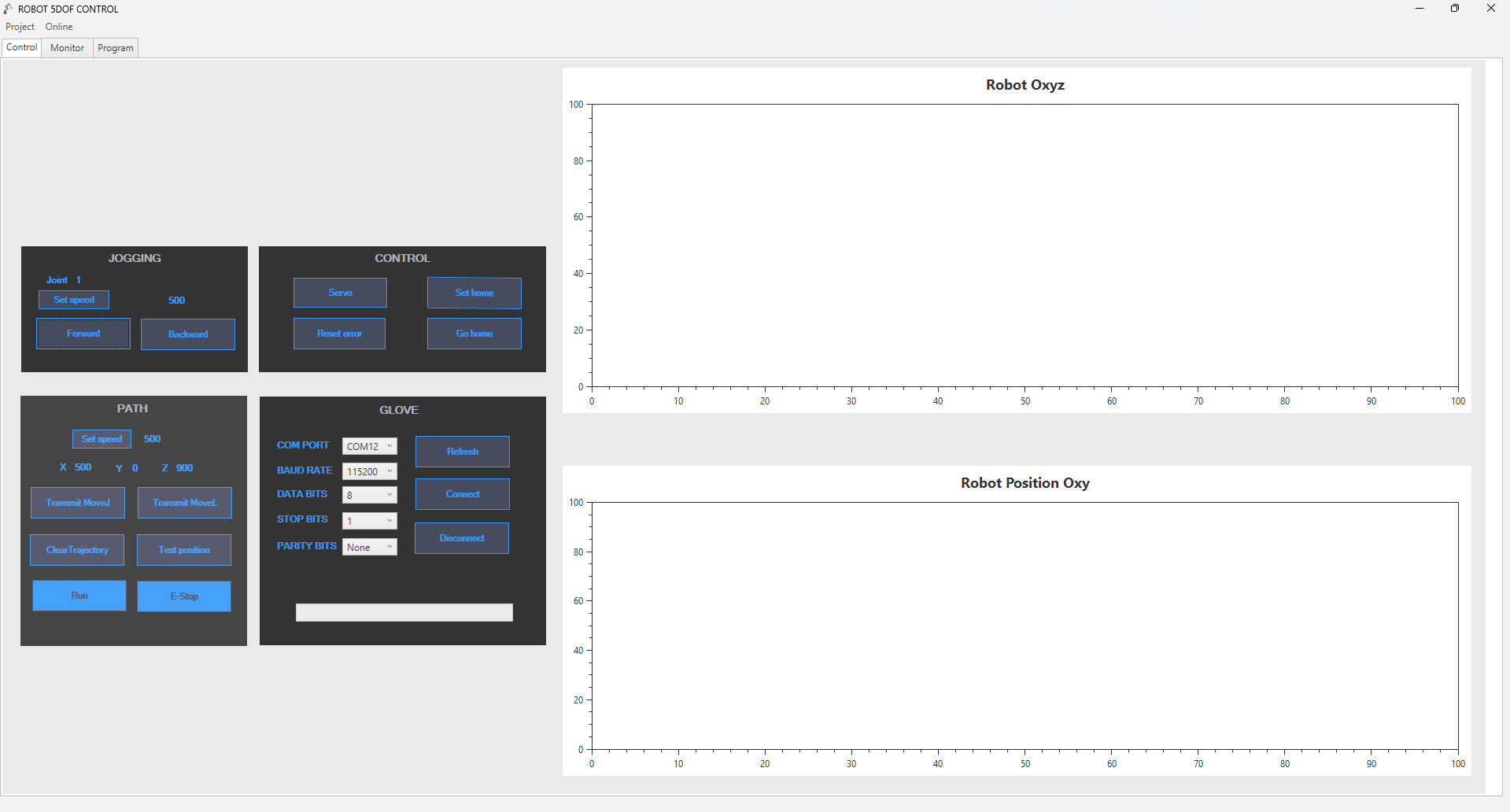
* **MainWindow**: Lớp đại diện cho cửa sổ chính, kế thừa từ Window.
* **InitializeComponent()**: Phương thức khởi tạo và load giao diện từ file XAML.
* **Button\_Click**: Phương thức xử lý sự kiện khi nút bấm được click, hiển thị một MessageBox.

1. **Giao diện điều khiển với WPF**

**Tổng quan về giao diện điều khiển:** Giao diện điều khiển robot sẽ được chia làm tab chính: Control, Monitor và Program. Tương ứng với từng vùng sẽ mang một chức năng khác nhau:

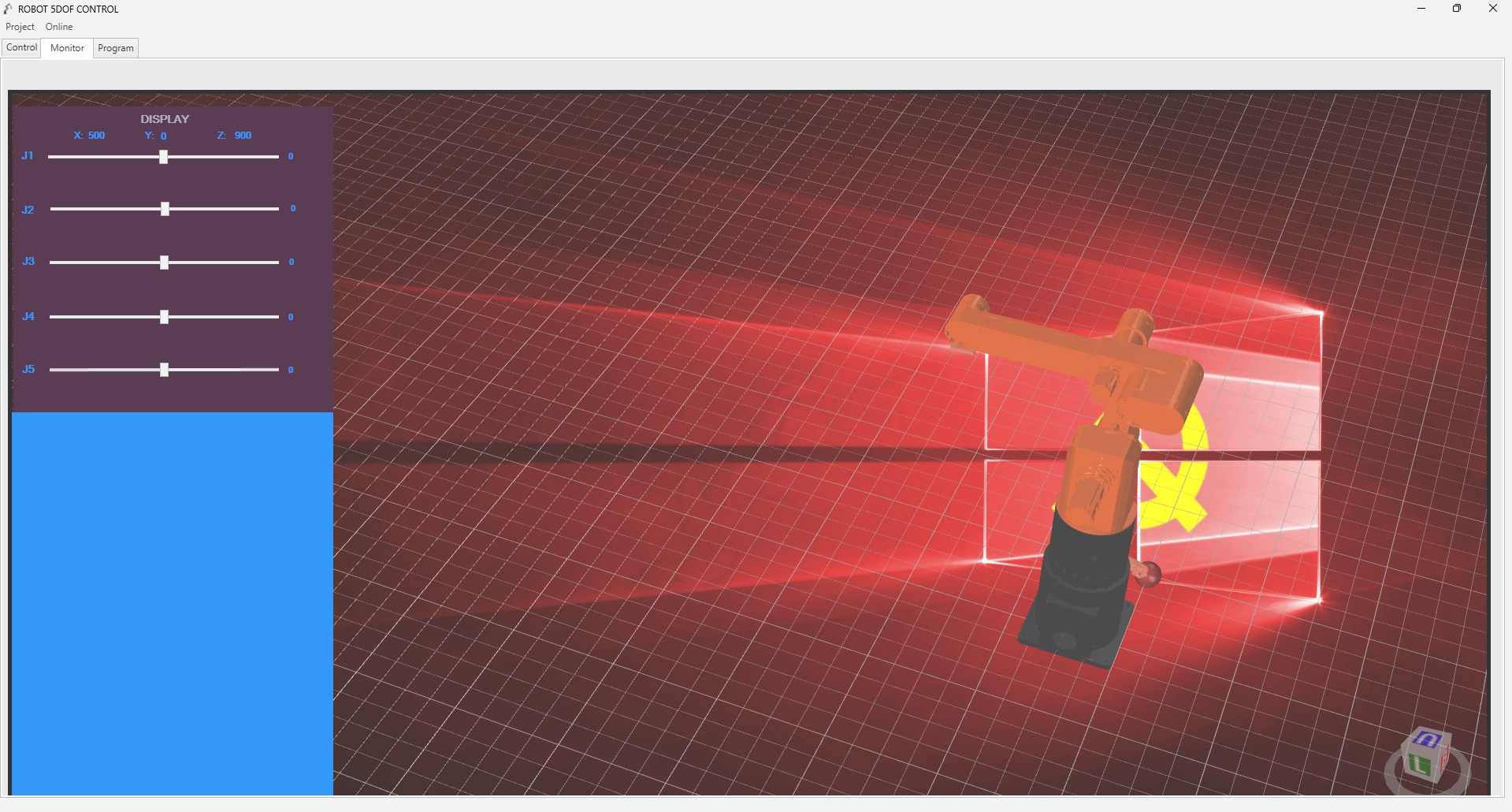
**Control:**

* Chạy Jog cho từng khớp của robot.
* Thực hiện những chức năng cơ bản như: Bật tắt Servo, Set home, Go home và Reset error.
* Thực hiện những lệnh chạy path như MoveJ, MoveL, Test Position và Clean Trajectory.
* Cổng kết nối UART để kết nối với thiết bị theo dõi chuyển động cánh tay người cũng như PLC Siemen.
* Đồ thị vận tốc và vị trí phục vụ cho việc theo dõi chuyển động.

****

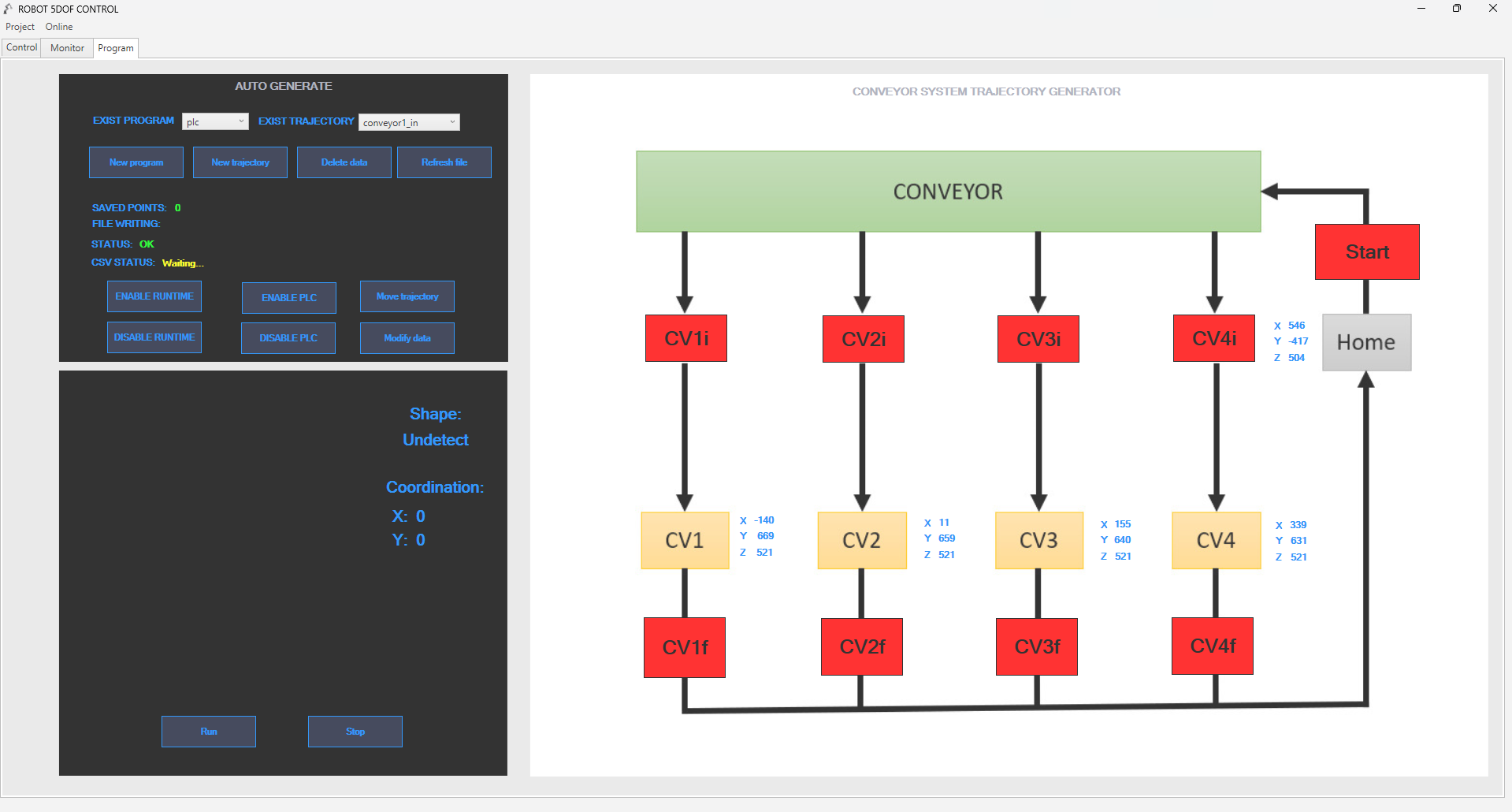
**Monitor:**

* Dùng để quan sát cũng như mô phỏng chuyển động của robot thực tế so với mô phỏng bằng robot 3D và Slider.
* Quan sát được trạng thái robot trong quá trình điều khiển thông qua textbox bao gồm như trạng thái nhấn các nút điều khiển, kiểm tra lỗi conversion động học nghịch robot cho các khớp từ hệ tọa độ…

****

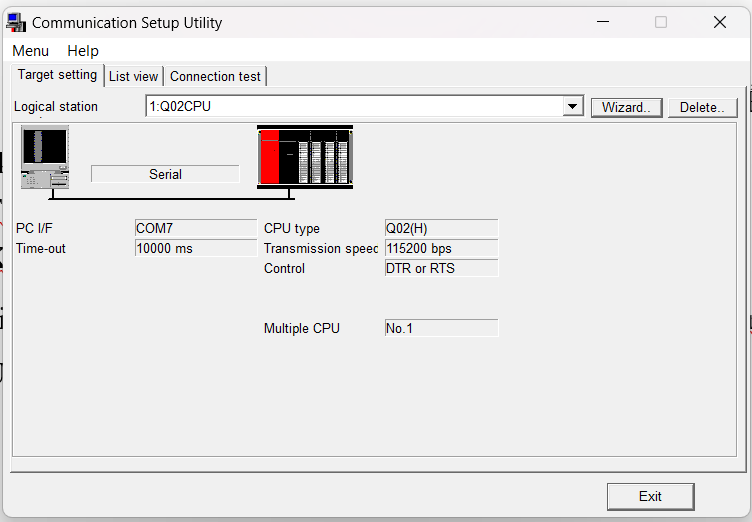
**Program:**

* Bật/Tắt chế độ điều khiển robot thông qua hệ thống điều khiển chuyển động và chức năng lưu quỹ đạo của robot tự động và bằng thủ công.
* Thể hiện được trạng thái số điểm được lưu lại vào tệp csv chứa quỹ đạo hiện tại.
* Tạo/Xóa tệp hiện có trong nơi chứa quỹ đạo của robot.
* Thể hiện được hình ảnh đã qua xử lý và trả ra kết quả nhận diện được từ camera về hình dạng cũng như trọng tâm từ camera.
* Bật/tắt chế độ điều khiển tự động hệ thống bốc xếp sản phẩm.

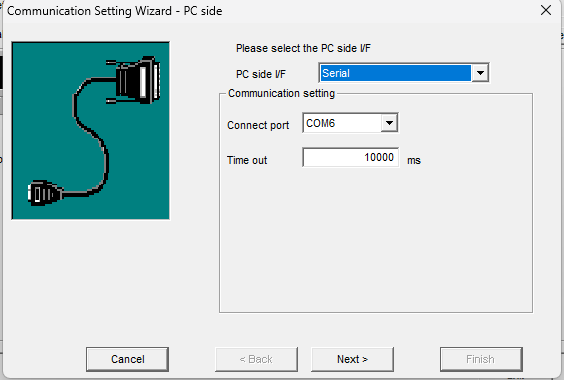
****

1. **Cách kết nối thư viện PLC Mitsubishi với máy tính**

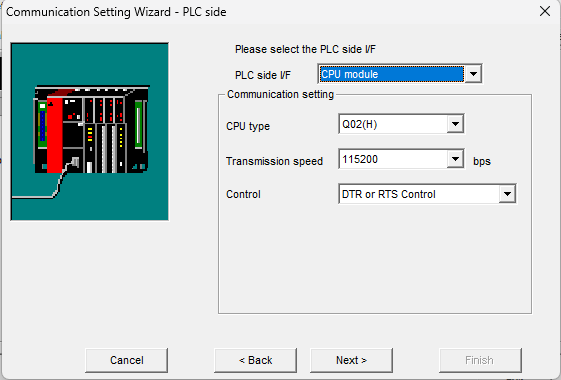
PLC Mitsubishi hỗ trợ thư viện **ActUtlTypeLib** thông qua phần mềm Communication Setup Utility để có thể kết nối được giữa PLC và C# thông qua serial port RS232.

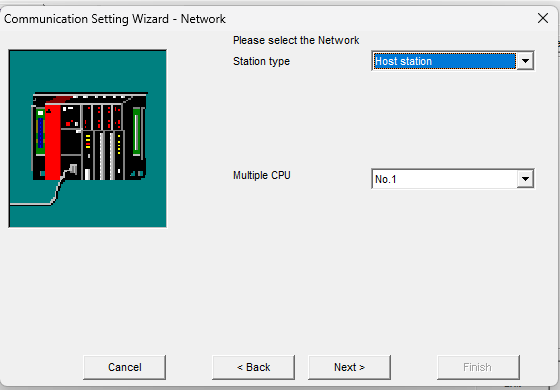


*Lưu ý: Khi mở phần mềm Communication Setupt Utility, phải mở bằng quyền Admin thì mới điều chỉnh được thông số trong phần mềm.*

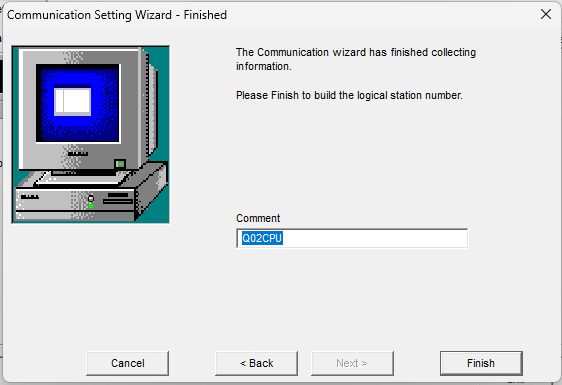
**

**Connect port** sẽ phụ thuộc vào cổng COM khi cắm dây RS232 vào máy tính. Ta có thể dễ dàng kiểm tra bằng Device Manager.



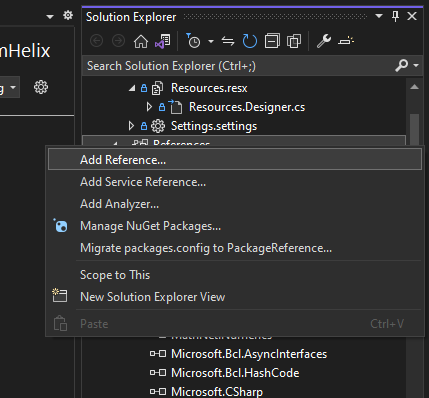


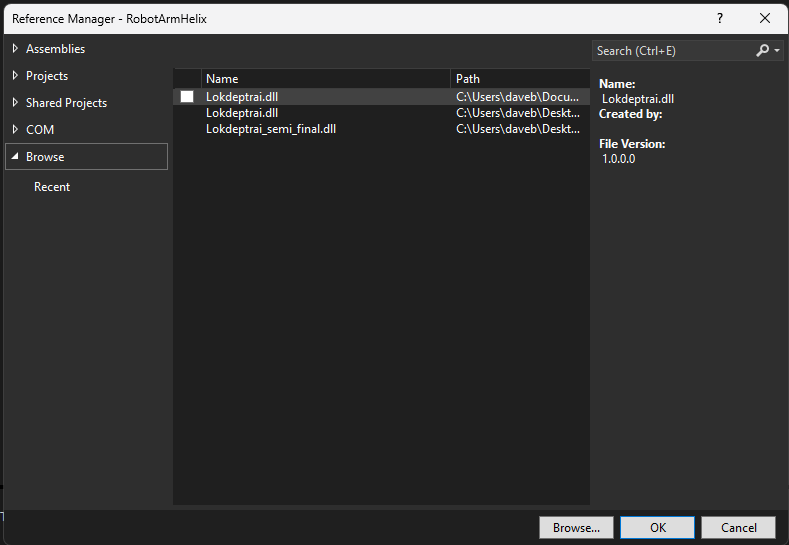
**Multiple CPU:** sẽ phụ thuộc vào vị trí lắp đặt module Q02HCPU 🡪 Hiện tại đang được đặt ở slot 1 🡪 **No.1**



Sau khi setup trên phần mềm, ta tiến hành cài đặt thư viện sử dụng trong WPF:

**B1: Add referrence vào trong chương trình:** Tại cửa sổ lập trình ở ô Solution Explorer góc bên phải 🡪 Chuột phải vào References 🡪 Add reference… 🡪 Tại cửa sổ Reference Manager 🡪 Chọn Browse 🡪 Ta chọn đường dẫn đến thư viện *ActUtlTypeLib.dll* 🡪 Nhấn OK và hoàn tất.





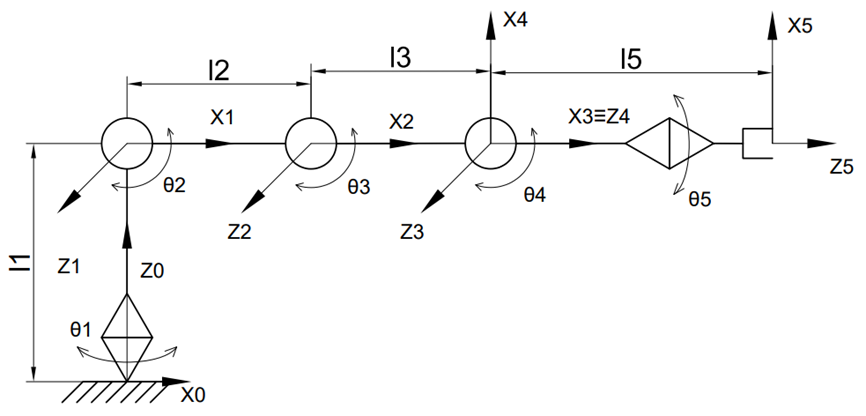
**B2:** ta khai báo cách kết nối trên C# bằng một số câu lệnh trong WPF ở tệp code-behind (.xaml.cs):

*using ActUtlTypeLib;*

Sau khi thiết đặt xong tất cả các cấu hình, ta đã hoàn tất việc kết nối giữa module Q02HCPU và có thể điều khiển relay ảo cũng như thay đổi các vùng nhớ thông qua phần mềm WPF.

1. **Chương trình điều khiển trong WPF trong hệ thống điều khiển robot.**
   1. **Lý thuyết cơ bản được áp dụng trong WPF trong hệ thống điều khiển robot**
      1. **Động học tay máy**
2. **Động học thuận**

Sơ đồ động Robot:

****

*Sơ đồ động Robot*

**Biểu diễn Denavit – Hartenberg:**

Mô hình hóa Denavit-Hartenberg (Viết tắt là phương pháp D - H) là cách biểu diễn đơn giản mô hình các khâu và khớp của robot và có thể sử dụng cho bất cứ cấu hình robot nào, kể cả bài toán phức tạp hay đơn giản.

Áp dụng phương pháp D – H, ta gắn một hệ trục tham chiếu tới mỗi khớp và sau đó xác định sự chuyển vị từ khớp này đến khớp kế tiếp, với mỗi khớp chúng ta sẽ gắn trục z và trục x lên chúng (hình 3.1).

Áp dụng phương pháp D - H cho khớp 1, tương tự với các khớp còn lại. Ta có:

* Đánh số trục z của khớp 1 là 0. Như vậy, trục z biểu diễn khớp 1 là z0. Trục x sẽ song song với đường vuông góc chung giữa các trục khớp của khâu, tương tự với các khớp còn lại.
* Với:
* l: biểu diễn khoảng cách dường vuông góc chung giữa trục z0­ và trục z1.
* θ1: góc quay từ trục x0 tới x1 xung quanh trục z­0.
* α1: góc quay của trục z0 tới z1 xung quanh trục x1.
* d1: khoảng cách từ gốc tọa độ thứ 0 tới giao điểm của của trục z0 và x1 dọc theo trục z0.

🡪 Từ đó, ta có thể lập ra bảng D – H với 5 khớp một cách tương ứng dựa theo hình 3.1:

Bảng D-H:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Khâu** | **θ** | **l** | **α** | **d** | **Biến** |
| **1** |  | **0** |  | **l1** |  |
| **2** |  |  | **0** | **0** |  |
| **3** |  |  | **0** | **0** |  |
| **4** |  | **0** |  | **0** |  |
| **5** |  | **0** | **0** | **l5** |  |

**Bảng Denavit–Hartenberg của robot**

Ở hệ trục thứ i sẽ mô tả ma trận vị trí và hướng so với hệ trục thứ i-1 như sau:

Ma trận chuyển trục Ai:

** (1)**

Với **.**

Từ đó, ta có ma trận chuyển trục của 5 khớp từ A1 đến A5 như sau:

** **

** **

****

Với kết quả trên, ta có ma trận chuyển vị tổng giữa nền Robot và cánh tay là:

****

**=**

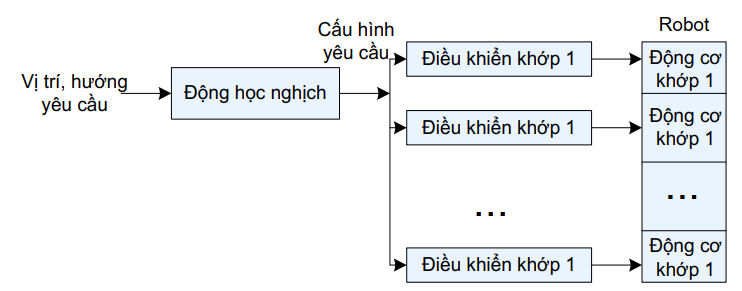
** (2)**

Theo động học thuận, tọa độ điểm P là:

****

1. **Động học nghịch**

Bài toán động học thuận gồm việc giải quyết vị trí và hướng của tay gắp hay cơ cấu chấp hành cuối khi biết tất cả các biến khớp. Bài toán động học nghịch yêu cầu tìm tập hay nhiều tập nghiệm của các khớp khi biết vị trí và hướng tay gắp.

****

Điều khiển vị trí của cánh tay robot

Với các giá trị đầu vào là Px, Py, P­z, φ, γ, Ta có:

****

****

**** (4)

****

Ta đặt:

****

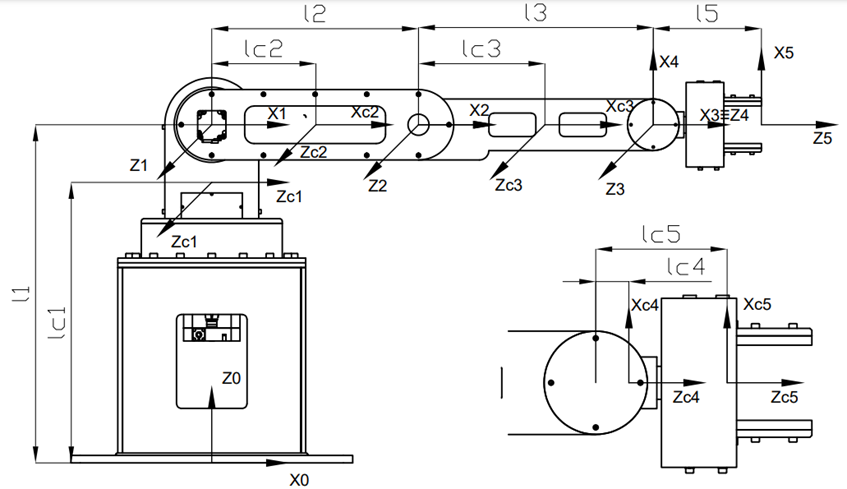
****

****

**** (6)

****

* + 1. **Động lực học tay máy**

****

*Sơ đồ tính động học robot*

Phương trình động lực học tổng quát

Hay

Với







là ma trận quán tính

là vector hướng tâm

là vector trọng lực

+Tính ma trận quán tính :

**Ta có:**



- Ma trận chỉ hướng Ri là ma trận lấy các phần tử của 3 hàng và 3 cột đầu của ma trận chuyển vị Ti (đã tính ở phần động học thuận)

- Ma trận moment quán tính Ii (là ma trận đặt trưng cho moment quán tính của các khâu)

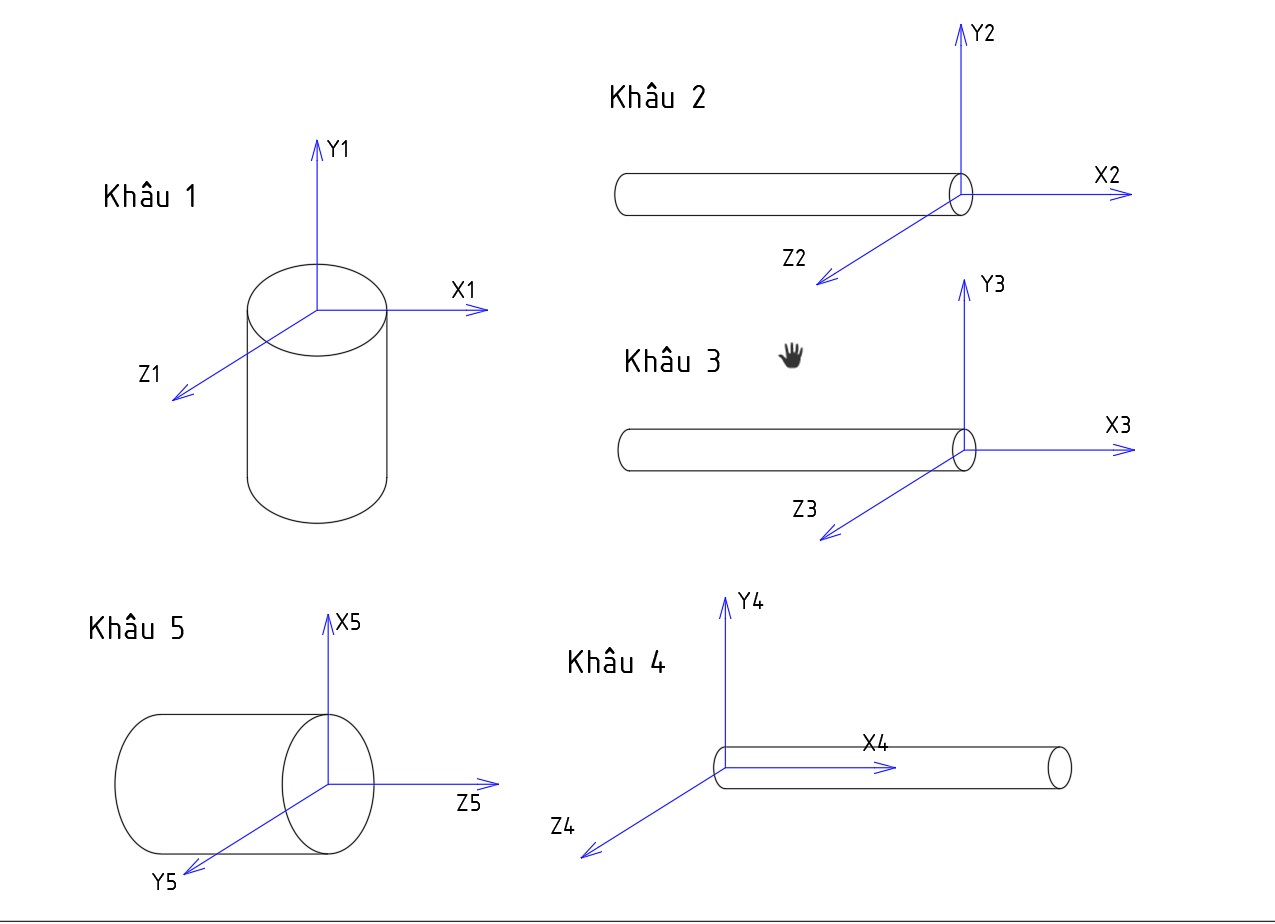
Để đơn giản hóa việc tính toán, giả sử các khâu đều đối xứng qua các trục xx, yy, zz

Ta có

Xét hình dáng của các khâu 2,3,4 là thanh thẳng có tiết diện ngnang không đáng kể.

Xét hình dáng của các khâu 1,5 là hình trụ tròn.

Ta có hình dáng tổng quát cùng vị trí đặt trục tọa độ như sau:



Hình 3.16: Hình dáng tổng quát để tính Moment quán tính của các khớp

Ta có:

I1: ,

I2: ,

I3: ,

I4: ,

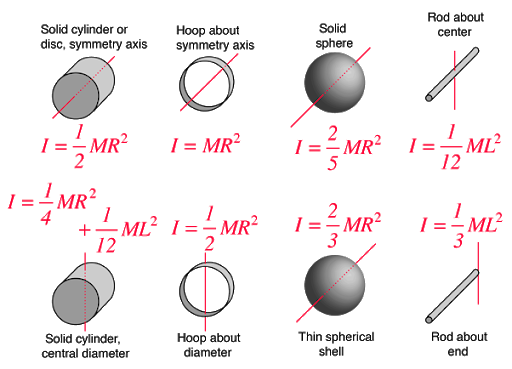
I5: ,

Tổng hợp lại ta có:

; ; ;

;

Sử dụng công thức tính moment quán tính cho các khối cơ bản ta xác định được giá trị của các phần tử trong ma trận I.



*Các công thức tính moment quán tính cho các khối cơ bản*

- Ma trận biểu diễn vi trí trọng tâm khâu so với hệ tọa độ gốc O0 (Các giá trị được tính bằng phương pháp hình học)

; ; ;

- Ma trận Jacobi khâu

Ma trận Jacobi vận tốc gốc :

0 với

Ta có:

; ; ; ;

0; 0; 0; 0;

0;

**Ma trận Jacobi vận tốc dài :**

***Kết quả:***

(22)

(23)

(24)

Với giá trị i chạy từ 1 đến 5, thay các giá trị vừa tính được vào biểu thức ban đầu, ta tìm được ma trận quán tính D(q).

- Tính Vector hướng tâm

với k chạy từ 1 đến 5 và i và j thể hiện vị trí phần tử của ma trận quán tính

- Tính Vector trọng lực

với

*P* là thế năng tổng cộng của cánh tay được tính bằng

* với

Mà chính là cột thứ k trong ma trận

- Tiến thành tính toán số liệu cụ thể

Ta có các thông số đã biết:

* m1=27,5 [kg]; m2=21 [kg]; m3=25,1 [kg]; m4=3 [kg]; m5=5,2 [kg];
* l1=690 [mm]; l2=440 [mm]; l3=500 [mm]; l5=230 [mm];
* lc1=660 [mm]; lc2=255 [mm]; lc3=143 [mm]; lc4=6 [mm]; lc5=143 [mm];
* g=9,81 [m/s2]
* Ixx1=461147 N.mm2; Iyy1=299920 N.mm2; Iyy2=803322 N.mm2; Iyy3=1604504 N.mm2 Iyy4=6328 N.mm2; Iyy5=24861 N.mm2; Izz5=14388 N.mm2

Vì khối lượng tính toán rất phức tạp nên ta sẽ sử dụng MATLAB để tăng độ chính xác và tin cậy cho quá trình tính toán. Ta sẽ thế giá trị trực tiếp và áp dụng thêm điều kiện ràng buộc để kết quả tính toán ngắn gọn nhất có thể.

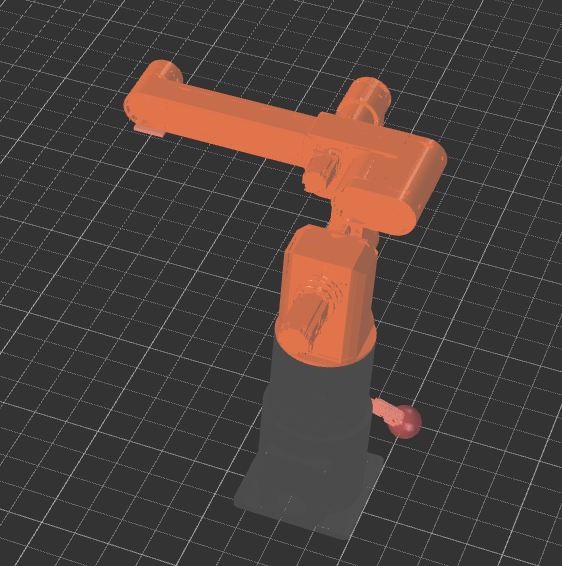
- Xác định tải tĩnh lớn nhất mà cánh tay robot chịu được:

Xét trường hợp cánh tay robot ở trạng thái tĩnh, khi đó các giá trị vận tốc và gia tốc sẽ bằng 0, từ đó dẫn đến . Phương trình động lực học khi đó chỉ còn phụ thuộc vào vector trọng lực :

= (49\*m4\*(l3\*cos(t2 + t3) + l2\*cos(t2)))/5 + (49\*m5\*(l3\*cos(t2 + t3) + l2\*cos(t2)))/5 + (49\*m3\*(lc3\*cos(t2 + t3) + l2\*cos(t2)))/5 + (49\*lc2\*m2\*cos(t2))/5

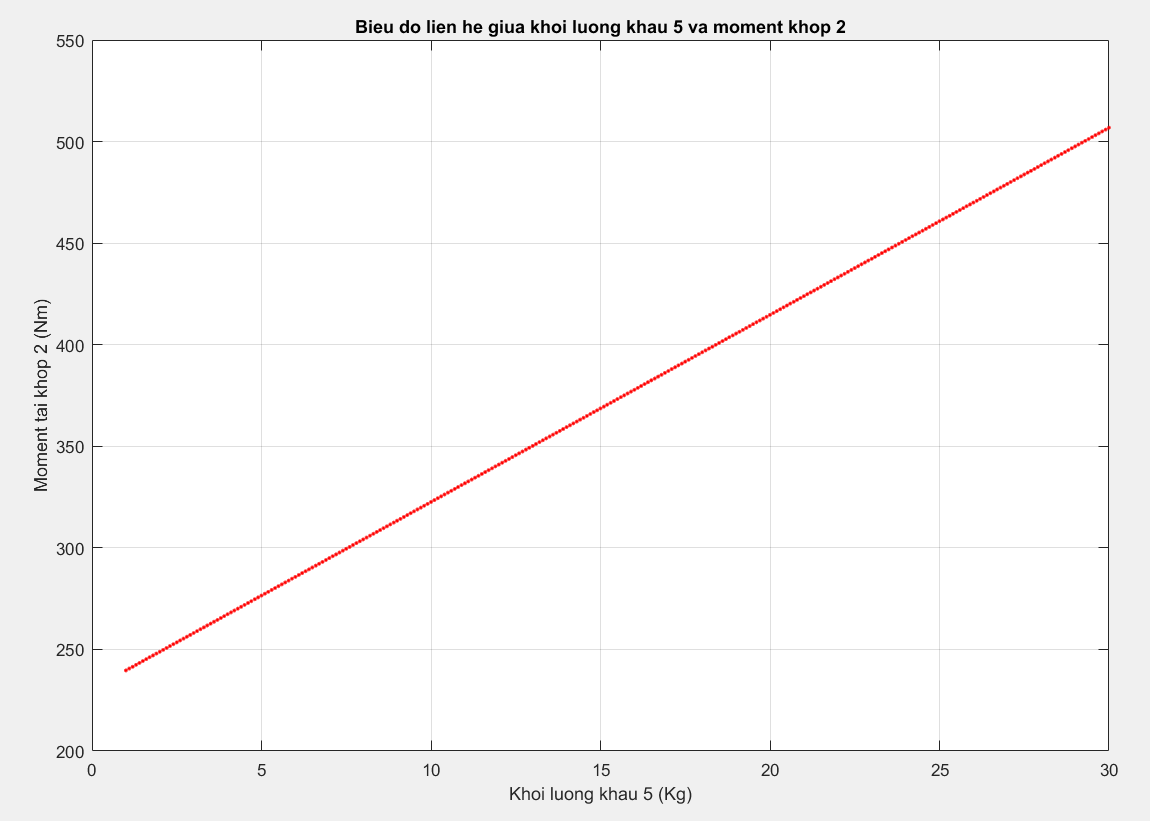
= (49\*cos(t2 + t3)\*(l3\*m4 + l3\*m5 + lc3\*m3))/5

Khi đó, ta xác định vị trí mà cánh tay robot chịu moment lớn nhất. Về nguyên tắc, moment được xác định bằng tích của lực và độ dài cánh tay đòn. Lực ở đây chính là trọng lực của vật thể cần gắp, cánh tay đòn là khoảng cách từ vật gắp đến gốc tọa độ ban đầu của robot. Ta xác định được cấu hình mà cánh tay chịu lực lớn nhất đó chính là khi cánh tay bị duỗi thẳng như hình:

****

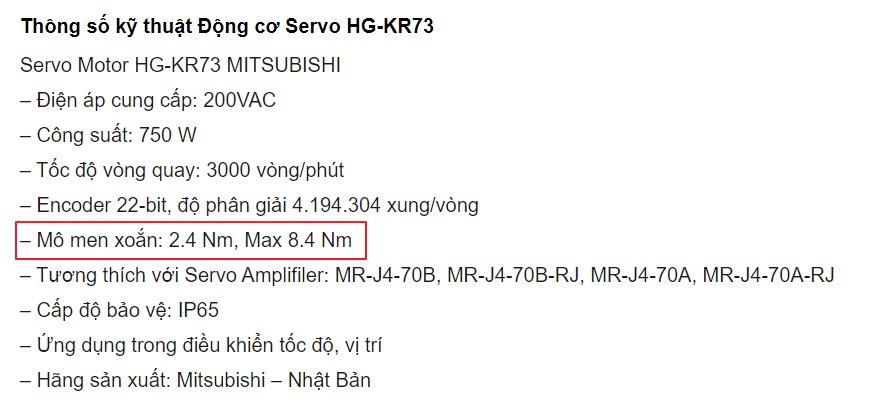
*Vị trí của robot mà động cơ chịu moment lớn nhất thõa mãn đầu tay gắp vuông góc với mặt đất*

Với khâu 5 gắn với vật cần gắp nên ta đặt là 1 biến số. Cho biến m5 tăng dần, ta thu được biểu đồ liên hệ giữa khối lượng vật càn gắp và moment phải chịu của các động cơ:

****

Mối liên hệ giữa khối lượng khâu 5 và moment phải chịu của khớp 2

Tra thông số động cơ tại khớp 2 của robot (HG-KR73) ta có:

****

Thông số động cơ tại khớp 2

Bộ truyền Hamornic có tỉ số truyền 1:180, cho hiệu suất bộ truyền, do đó moment mà khớp 2 chịu được sẽ là:

T2max = 2.4 \* 180 = 432 Nm

Suy ra khối lượng lớn nhất của khâu số 5 mà động cơ vẫn chịu được là 19Kg

Từ đó, vật lớn nhất mà robot gắp được sẽ khoảng 17kg.

**Ma trận Jacobi nghịch sử dụng phương pháp The Pseudo-Inverse:**

Từ ma trận Jacobi thuận Jv5 . Vì ma trận cho robot 5 bậc tự do là một ma trận không vuông, ta không thể tính Jacobi nghịch theo cách thông thường. Ta sẽ áp dụng công thức Pseudo-Inverse:



Với  và là vận tốc dài của robot trong tọa độ Oxyz, từ dó ta sẽ có được ma trận chứa vận tốc cho từng khớp của robot.

Từ những kiến thức nêu trên, ta có thể áp dụng đưa vào hệ thống điều khiển để tính toán cũng như đưa ra những giải thuật điều khiển robot về vị trí cũng như vận tốc, đảm bảo tính linh hoạt và hiệu quả trong quy trình hoạt động.

* 1. **Các phương pháp điều khiển quỹ đạo trong WPF**
     1. **Phương pháp điều khiển theo đường thẳng**

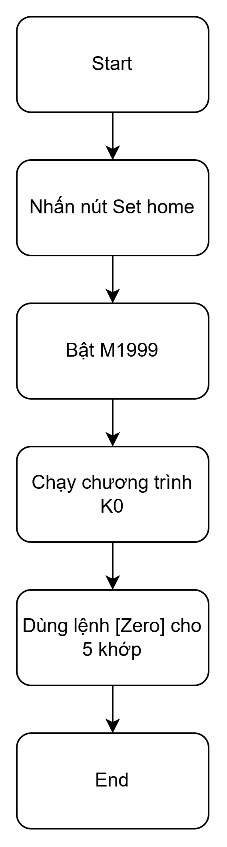
Từ dữ liệu đầu vào là 2 đầu của đoạn thẳng ta có thể xác định được phương trình và từ đó tìm ra đồ thị vị trí của 5 trục khi thực hiện quỹ đạo. Để có thể thực hiện chuyển động của robot bám theo đường thẳng mong muốn ta cần phải điều khiển vị trí của 5 trục theo đồ thị đã tính toán dựa trên quỹ đạo cần di chuyển. Ta tiến hành chia quỹ đạo thành các đoạn nhỏ, mỗi đoạn nhỏ sẽ sử dụng hàm Constant Speed để tiến hành di chuyển. Số điểm chia càng nhiều, quỹ đạo bám theo sẽ càng chính xác.

* + 1. **Phương pháp điều khiển theo hệ thống theo dõi chuyển động**

Trong PLC, nhà sản xuất cho phép ta can thiệp vào những vùng nhớ đặc biệt giúp thiết đặt các giá trị góc một cách tức thời cho động cơ, để điều khiển động cơ ngay tức thì. Dựa vào đó ta sẽ dùng hệ thống theo dõi chuyển động để cập nhật vị trí tức thời cho robot. Từ đó sẽ tạo ra 1 quỹ đạo với một đường cong bất kỳ bám theo chuyển động của tay người. Tăng khả năng linh hoạt và chính xác của robot.

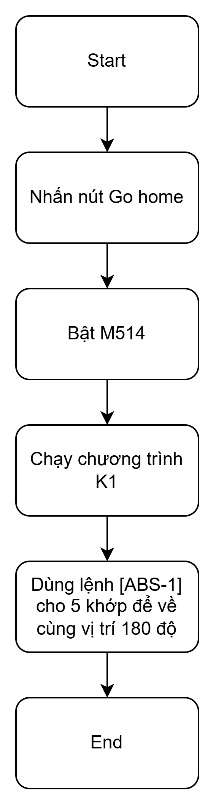
* + 1. **Giải thuật cho các câu lệnh điều khiển trên WPF**

**Set Home**



*Lưu đồ cho lệnh Set Home*

**Go Home**



*Lưu đồ cho lệnh Go Home*

**Chạy đơn điểm (MoveJ)**



*Lưu đồ cho lệnh MoveJ*

**Chạy đa điểm (MoveL)**



*Lưu đồ cho lệnh MoveL*

* 1. **WPF với thiết bị theo dõi chuyển động**

**Lưu đồ giải thuật:**

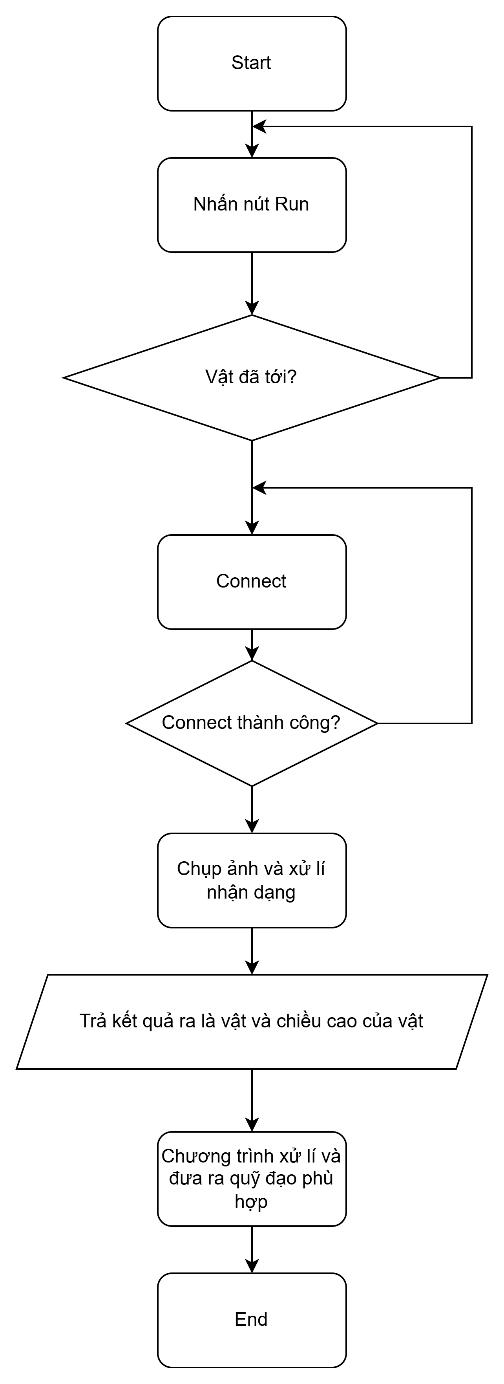


*Lưu đồ cho lệnh chạy Runtime*

Như đã trình bày ở chương trước, sau khi ta nhận được frame truyền từ thiết bị theo dõi chuyển động, ta tiến hành bóc tách và xử lý từng byte trong frame từ đó có được vị trí mong muốn và vận tốc của găng tay. Từ đó tiến hành tính toán động học nghịch và áp dụng Jacobi nghịch thì ta sẽ có được chuyển động của robot sẽ đáp ứng tức thời cả về vị trí và vận tốc của cánh tay robot.

* 1. **WPF với camera nhận dạng vật ảnh O2D222**

**Lưu đồ giải thuật:**



Bằng việc giao tiếp thông qua giao thức TCP/IP, ta có thể kết nối được với camera và dựa vào những thuật toán xử lí ảnh đã nêu ở chương trước. Ta có thể áp dụng vào hệ thống điều khiển phục vụ cho hệ thống điều khiển tự động.

* 1. **WPF với PLC Siemen**

**Lưu đồ hoạt động:**



Bằng việc giao tiếp với PLC Siemen với WPF thông qua UART, ta có thể sử dụng nó như một phương thức truyền thông giữa chúng để từ đó có thể thiết lập nên 1 hệ thống điều khiển tự động hình thành nên một quy trình điều khiển.