LỜI NÓI ĐẦU

Giáo trình *"Lập Trình PLC Theo Ngôn Ngữ Bậc Thang"* trước hết dành cho sinh viên ngành Công nghệ Kỹ thuật Cơ điện tử, khoa Cơ học kỹ thuật và Tự động hóa, Trường Đại học Công nghệ - ĐHQGHN với môn học lập trình PLC và thực tập chuyên ngành (năm thứ 4). Nội dung giáo trình cũng phù hợp cho công tác giảng dạy môn học Điều khiển tự động và Tự động hóa trong bậc đào tạo Đại học của trường. Ngoài ra giáo trình này cũng có thể phục vụ cho các sinh viên, kỹ sư và các Thầy Cô quan tâm đến vấn đề liên quan.

Trong giáo trình, nhóm tác giả đưa ra bức tranh về việc ứng dụng các bộ điều khiển logic khả trình và nhấn mạnh vai trò của nó trong quá trình Tự động hóa Công nghiệp. Các kiến thức lập trình và ứng dụng PLC được đưa ra từ cách nhìn của những người thiết kế hệ thống cũng người như những lập trình viên. Nội dung giáo trình đề cập một cách hệ thống những kiến thức cơ bản và hiện đại của kỹ thuật lập trình PLC theo ngôn ngữ bậc thang.

Đi cùng với cơ sở lý thuyết, giáo trình cũng trình bày nhiều ví dụ cụ thể. Các ví dụ được minh họa trên bộ PLC ED–4260 của hãng LS với công cụ phát triển và mô phỏng trong môi trường GMWIN.

Giáo trình này được chia làm 5 chương:

Chương 1: *Giới thiệu định nghĩa, lịch sử phát triển, cấu trúc, tiêu chuẩn kỹ thuật, phân loại và ứng dụng của PLC trong quá trình điều khiển.*

Chương 2: *Đề cập tới các hệ thống số được sử dụng thường xuyên và cách chuyển đổi giữa các hệ thống số.*

Chương 3: *Giới thiệu cấu tạo, nguyên lý làm việc của một số loại thiết bị vào/ra thường gặp trong hệ thống điều khiển sử dụng PLC.*

Chương 4: *Trình bày những khái niệm cơ bản về ngôn ngữ lập trình bậc thang, mô tả nguyên lý hoạt động và đưa ra các ví dụ minh hoạ cho các lệnh, các hàm, khối hàm được sử dụng nhiều trong quá trình thiết kế chương trình.*

Chương 5: *Trình bày các phương pháp phân tích, thiết kế chương trình cho hệ thống điều khiển, các phương pháp gỡ rối, sửa lỗi chương trình, các yêu cầu kiểm tra an toàn trước khi vận hành hệ thống.*

Trong quá trình biên soạn nhóm tác giả đã được các bạn đồng nghiệp góp nhiều ý kiến bổ ích. Ban chủ nhiệm Khoa Cơ học kỹ thuật và Tự động hoá và các phòng ban Trường Đại học Công nghệ đã tạo điều kiện tốt nhất để hoàn thành giáo trình này. Nhóm tác giả xin bày tỏ lời cảm ơn chân thành về sự giúp đỡ quý báu đó.

Mặc dù nhóm tác giả đã cố gắng thể hiện nội dung giáo trình một cách cơ bản, hiện đại và có hệ thống nhưng vì đây là lần đầu tiên giáo trình được xuất bản nên không tránh khỏi những thiếu sót, nhóm tác giả rất mong nhận được các ý kiến đóng góp của bạn đọc, đặc biệt là các đồng nghiệp và các em sinh viên để giáo trình được hoàn thiện hơn. Thư từ liên hệ xin gửi về địa chỉ: ***Khoa Cơ học Kỹ thuật và Tự động hoá – Trường Đại học Công nghệ – ĐHQGHN***.

Xin chân thành cảm ơn!

NHÓM TÁC GIẢ

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

PLC	Programmable Logic Controller	Bộ điều khiển logic khả trình
NEMA	National Electrical Manufacturers Association	Hiệp hội những nhà sản xuất điện quốc gia
IEC	International Electrotechnical Commission	Ủy ban kỹ thuật điện quốc tế
LD	Ladder Diagram	Sơ đồ bậc thang
FBD	Function Block Diagram	Sơ đồ khối hàm
IL	Instruction List	Liệt kê câu lệnh
ST	Structured Text	Ký tự có cấu trúc
SFC	Sequential Function Chart	Sơ đồ chức năng tuần tự
CPU	Central Processing Unit	Đơn vị xử lý trung tâm
DC	Direct Current	Dòng một chiều
AC	Alternating Current	Dòng xoay chiều
LED	Light Emitting Diode	Diode phát quang
CNC	Computerized Numeric Control	Sự điều khiển số máy tính hóa
ROM	Read–Only Memory	Bộ nhớ chỉ đọc
RAM	Random Access Memory	Bộ nhớ truy cập ngẫu nhiên
EPROM	Erasable Programmable Read–Only	Bộ nhớ chỉ đọc có thể xóa
EEPROM	Electrically Erasable Programmable Read–Only Memory	Bộ nhớ chỉ đọc có thể xóa bằng tín hiệu điện.
IC	Integrated Circuit	Mạch tích hợp

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

Chương 1
BỘ ĐIỀU KHIỂN LOGIC KHẢ TRÌNH – PLC

1.1. BỘ ĐIỀU KHIỂN LOGIC KHẢ TRÌNH

1.1.1. Định nghĩa

PLC là từ viết tắt của **Programmable Logic Controller** (Bộ điều khiển logic khả trình), được dùng để thay thế chức năng của các bộ rơle, bộ đếm hay bộ định thời trong các thiết bị điều khiển, đồng thời có thêm khả năng tính toán cơ bản giúp khả năng điều khiển dễ dàng được thực hiện.

Hiệp hội những nhà sản xuất điện quốc gia (NEMA) định nghĩa "PLC là thiết bị điện tử định hướng kĩ thuật số, sử dụng bộ nhớ có thể lập trình được để thực hiện những chức năng đặc biệt như logic, chuỗi, định thời, đếm và tính toán thông qua các mô-đun vào/ra số hoặc tương tự, có khả năng điều khiển các máy móc và các bộ xử lí khác nhau".

1.1.2. Lịch sử ra đời

Khái niệm PLC là ý tưởng của nhóm kỹ sư hãng General Motors vào năm 1968, với ý tưởng ban đầu là thiết kế và chế tạo một thiết bị với các chỉ tiêu kỹ thuật nhằm đáp ứng những yêu cầu điều khiển sau:

- Dễ lập trình và thay đổi chương trình điều khiển.
- Cấu trúc dạng mô-đun dễ mở rộng, dễ bảo trì và sửa chữa.
- Đảm bảo độ tin cậy hơn bộ điều khiển rơle.
- Đầu ra phải có khả năng kết nối tới các máy tính bậc cao hơn.
- Có hiệu quả kinh tế hơn so với bộ điều khiển rơle.
- Điện áp đầu vào sử dụng nguồn 115 VAC.
- Điện áp đầu ra 115 VAC, 2A.
- Trang bị bộ nhớ có khả năng lập trình được.
- Có khả năng mở rộng mà không cần phải thay đổi toàn bộ hệ thống.

Năm 1970, bộ điều khiển logic khả trình đầu tiên đã ra đời, đáp ứng được các thông số kỹ thuật cơ bản và mở ra sự phát triển cho một công nghệ điều khiển mới.

PLC có thể được coi là một tiến bộ mới với những chức năng giống như hệ điều khiển sử dụng role, thiết bị tương tự, hay các bộ xử lý logic khác. Theo thời gian, các chức năng của PLC ngày càng được cải thiện nhưng các tiêu chí thiết kế cũng như chi tiết kỹ thuật vẫn dựa trên những ý tưởng ban đầu là dễ sử dụng và có khả năng tái sử dụng.

Những tiến bộ về phần cứng:

- Dung lượng bộ nhớ lớn hơn.
- Số lượng ngõ vào/ra nhiều hơn.
- Nhiều loại mô-đun chuyên dụng hơn.
- Có khả năng điều khiển các ngõ vào/ra từ xa thông qua kỹ thuật truyền thông.
- Phát triển và hoàn chỉnh hơn về tốc độ xử lý cũng như hiệu suất làm việc bằng cách áp dụng những tiến bộ trong công nghệ điện tử và vi xử lý.
- Chi phí giá thành thấp.
- Giao diện điều khiển được cải thiện.

Về phần mềm cũng có sự phát triển cụ thể là:

- Lập trình hướng đối tượng đa ngôn ngữ dựa trên tiêu chuẩn IEC 1131-3. Nhưng ngôn ngữ được sử dụng nhiều và hiệu quả nhất là ngôn ngữ bậc thang.
- Ngôn ngữ lập trình bậc cao như C hay Passcal đã được sử dụng để lập trình cho các mô-đun của PLC, giúp tạo sự linh hoạt hơn khi giao tiếp với các thiết bị ngoại vi và thao tác dữ liệu.
- Các lệnh lập trình đơn giản nhờ có sự mở rộng của các khối chức năng.
- Hệ thống chuẩn đoán và phát hiện lỗi đã được mở rộng và đơn giản hóa, nhằm phát hiện lỗi trong điều khiển bao gồm chuẩn đoán máy, tìm lỗi trong quá trình điều khiển.
- Từ các lệnh logic đơn giản thì ngày nay các bộ PLC được hỗ trợ thêm các lệnh về tác vụ định thời, tác vụ đếm, sau đó là các lệnh về xử lý toán học, xử lý bảng dữ liệu, xử lý xung ở tốc độ cao, tính toán số thực 32-bit, xử lý thời gian thực, đọc mã vạch giúp PLC có khả năng thực hiện các yêu cầu phức tạp.

- Thao tác và xử lý dữ liệu được đơn giản hóa phù hợp với các yêu cầu điều khiển phức tạp.

Ngày nay, PLC cung cấp khả năng dự đoán cao. Chúng có thể giao tiếp với các hệ thống điều khiển khác, đưa ra các báo cáo sản xuất, lập kế hoạch sản xuất, và dự đoán lỗi của hệ thống trong quá trình hoạt động. Chính những tiến bộ đó đã góp phần làm cho PLC ngày càng đóng một vai trò quan trọng trong việc đáp ứng nhu cầu về chất lượng và năng suất công việc.

Bảng 1.1. Lịch sử ra đời PLC

Năm	Sự kiện
1968	Ra đời khái niệm về bộ điều khiển logic khả trình - PLC
1969	Giới thiệu bộ điều khiển logic khả trình PLC đầu tiên với bộ nhớ 1k và xử lý được 128 điểm vào/ra
1975	PLC với bộ điều khiển PID
1980	Các module vào/ra thông minh
1981	PLC nối mạng, 16-bit PLC, các màn hình CRT màu
1992	Chuẩn IEC 61131 ra đời
1996	PLC được thiết kế với các khe cắm để có thể mở rộng các mô-đun vào/ra
Ngày nay	Các PLC có thể kết nối với nhau tạo thành các hệ thống điều khiển phân tán

1.1.3. Tiêu chuẩn của PLC

a) Tiêu chuẩn IEC (Uỷ ban kỹ thuật điện quốc tế)

Ngày nay, nhiều người đã gặp những khó khăn nhất định với ngôn ngữ lập trình và truyền thông khi làm việc với PLC của các nhà sản xuất khác nhau. Để giải quyết vấn đề, IEC đã thống nhất và đưa ra tiêu chuẩn quốc tế IEC 1131. Tiêu chuẩn này bao gồm 5 phần.

Bảng 1.2. Tiêu chuẩn IEC 1131

Phần	Mô tả
1	Đặc điểm cơ bản của PLC và định nghĩa các thuật ngữ
2	Các chức năng cần thiết và các điều kiện thử nghiệm của các tính năng
3	Ngôn ngữ lập trình
4	Chú ý cho người sử dụng
5	Giao tiếp và mạng truyền thông

b) Một vài đặc điểm quan trọng được giới thiệu bởi IEC

- Hỗ trợ nhiều loại dữ liệu.

- Các thành phần như hàm, khối hàm, và chương trình có thể thực hiện theo thứ tự từ trên xuống (Top – Down), từ dưới lên (Bottom – Up) hay thực hiện theo kiểu cấu trúc (Structured).

- Chương trình người sử dụng có thể được xây dựng thành các thư viện để sử dụng trong các môi trường lập trình khác.

- Hỗ trợ đa ngôn ngữ, nhờ đó người sử dụng có thể lựa chọn ngôn ngữ hiệu quả tối ưu nhất cho việc sử dụng.

- Tiêu chuẩn ngôn ngữ lập trình PLC được đề xuất bởi IEC bao gồm:

Ngôn ngữ đồ hoạ (Graphic Language)

Sơ đồ bậc thang (LD): Đầu vào và đầu ra được kết nối thành một dạng chương trình thuộc loại biểu diễn trạng thái logic của rơle và giáo trình *"Lập trình PLC theo ngôn ngữ bậc thang"* sẽ trình bày theo ngôn ngữ lập trình này.

Sơ đồ khối hàm (FBD): Chương trình được biểu diễn dưới dạng kết nối các khối hàm.

Ngôn ngữ văn bản (Text-Based Language)

Liệt kê lệnh (IL): Đây là loại ngôn ngữ tương tự như ngôn ngữ Asembly.

Ngôn ngữ văn bản kiểu cấu trúc (ST): Ngôn ngữ bậc cao sử dụng trong các ứng dụng thời gian thực và dựa trên ngôn ngữ lập trình C, Passcal.

Sơ đồ chức năng tuần tự (SFC): Mô tả tiến trình và các điều kiện của quá trình sản xuất một cách liên tiếp với thời gian và sự kiện gì được biểu diễn như các khối điều khiển liên tiếp.

1.2. CẤU TRÚC VÀ ĐẶC ĐIỂM CỦA PLC

1.2.1. Cấu trúc phần cứng

Một hệ thống PLC bao gồm các thành phần chức năng cơ bản như: Bộ vi xử lý trung tâm, bộ nhớ, bộ cung cấp nguồn điện, giao diện đầu vào/đầu ra, giao diện truyền thông, và các thiết bị lập trình.

- Bộ xử lý trung tâm CPU đóng vai trò như là bộ não của PLC, CPU thực hiện và giải mã lần lượt từng chương trình được lưu trữ trong bộ nhớ. Bao gồm nhận dữ liệu ở ngõ vào, xử lý chương trình, nhớ chương trình, xử lý các kết quả

trung gian và xuất các kết quả xử lý tới các ngõ ra. Quá trình này được lặp đi lặp lại rất nhanh và mọi dữ liệu đều được xử lí dưới dạng mã nhị phân.

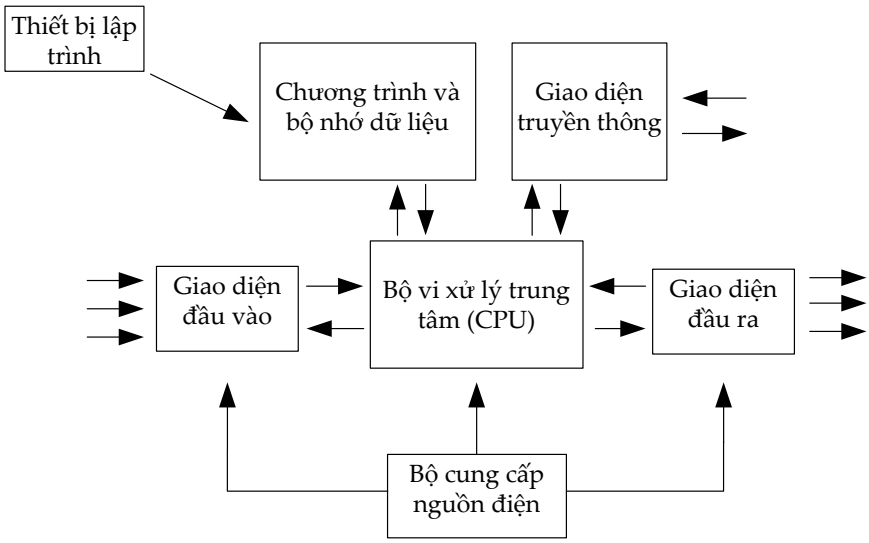
- Bộ cung cấp nguồn điện chuyển đổi nguồn điện AC thành nguồn điện áp DC cần thiết cho các bộ phận khác sử dụng.

- Các thiết bị lập trình được sử dụng để thiết kế và nạp chương trình điều khiển vào bộ nhớ của PLC.

- Bộ nhớ là nơi lưu trữ các hoạt động được thực hiện bởi bộ vi xử lý và là nơi lưu trữ dữ liệu được nhận từ đầu vào cũng như dữ liệu được xử lý ở đầu ra.

- Giao diện đầu vào/ra là bộ phận giao tiếp giữa hệ thống bên trong PLC với các thiết bị bên ngoài. Giao diện đầu vào có thể nhận tín hiệu từ các cảm biến hay các thiết bị đầu vào khác và đưa vào CPU. Giao diện đầu ra đưa tín hiệu điều khiển từ CPU tới cơ cấu chấp hành. Giao diện đầu vào/ra có thể được phân loại thành các các giao diện có tín hiệu rời rạc, số hay tương tự. Thiết bị cho tín hiệu rời rạc hoặc số là những thiết bị được sử dụng trong các ứng dụng sử dụng các tín hiệu OFF hoặc ON. Với tín hiệu rời rạc, mỗi một lần thay đổi tín hiệu sẽ cho một trạng thái tín hiệu rời rạc (không có điện áp hoặc có điện áp). Tín hiệu số có thể được xem như các tín hiệu rời rạc nhưng sẽ là một chuỗi các tín hiệu ON/OFF. Tín hiệu tương tự là tín hiệu có độ lớn các tín hiệu cần theo dõi tỷ lệ với nhau. Ví dụ, một bộ cảm biến nhiệt độ có thể cung cấp cho một điện áp tỉ lệ với nhiệt độ.

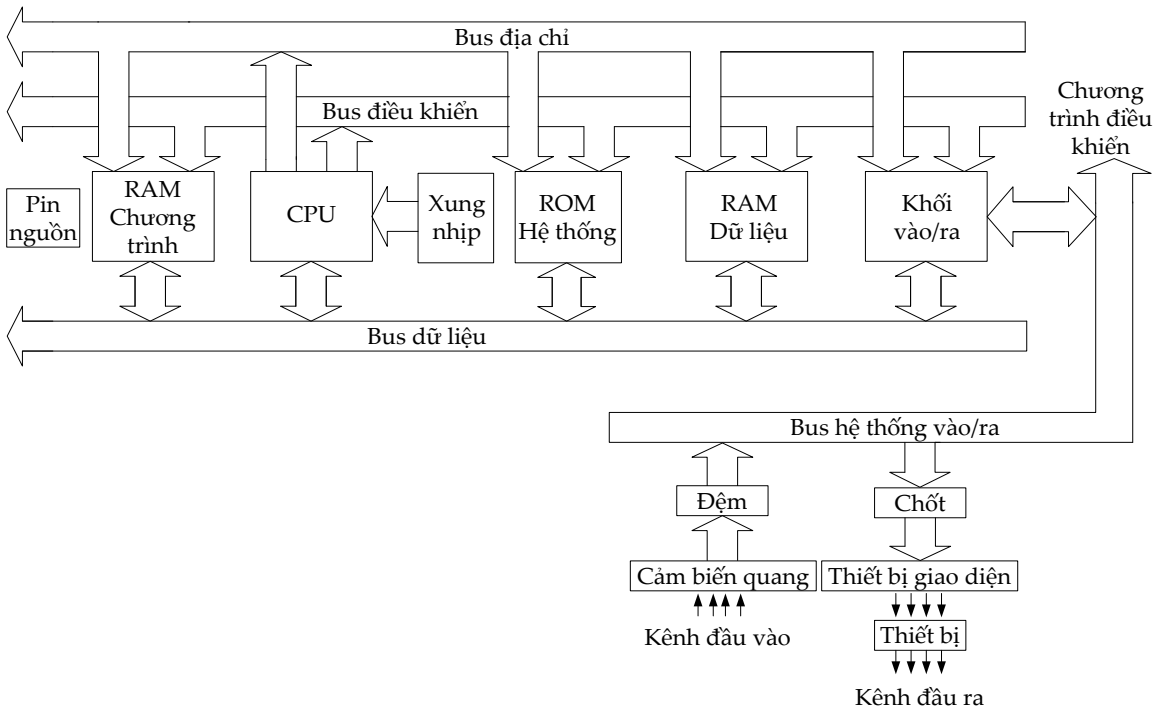
- Giao diện truyền thông được sử dụng để truyền và nhận dữ liệu giữa các PLC, PLC với PC hoặc thiết bị có khả năng giao tiếp khác, bao gồm xác minh thiết bị, thu thập dữ liệu, đồng bộ hóa giữa các ứng dụng của người sử dụng và quản lý kết nối.



Hình 1.1. Cấu trúc phần cứng PLC

1.2.2. Cấu trúc bên trong PLC

Hình 1.4 mô tả cấu trúc bên trong của PLC, bao gồm bộ xử lý trung tâm CPU, bộ nhớ, khối vào/ra. CPU điều khiển và xử lý tất cả các hoạt động của PLC, CPU được nối với bộ xung nhịp có tần số từ 1 tới 8 MHz, tần số này xác định tốc độ hoạt động của PLC, xung nhịp và sự đồng bộ hóa cho tất cả các yếu tố trong hệ thống. Thông tin trong PLC được lưu trữ và xử lý dưới dạng tín hiệu số, và được truyền đi theo một đường gọi là bus. Trong vật lý, bus là một dây dẫn truyền tín hiệu hay là đường kết nối trong một mạch in. CPU sử dụng bus dữ liệu để truyền dữ liệu giữa các mô-đun theo các bus địa chỉ để truyền dữ liệu và đưa ra các tín hiệu điều khiển. Hệ thống bus sử dụng cho truyền thông giữa các ngõ vào/ra và các khối vào/ra.



Hình 1.2. Cấu trúc bên trong PLC

a) Bộ điều khiển trung tâm CPU

Cấu trúc bên trong của CPU phụ thuộc vào bộ vi xử lý tương ứng nhưng nói chung nó sẽ bao gồm các yếu tố cơ bản sau đây:

- Khối logic và số học có chức năng thực hiện các phép tính logic và số học như cộng, trừ, nhân, chia, và logic (AND, OR, NOT).

- Bộ nhớ nằm bên trong CPU và là nơi lưu trữ thông tin liên quan đến việc thực hiện chương trình.

- Bộ điều khiển được sử dụng để kiểm soát thời gian và hoạt động.

b) Đường Bus

Bus là loại đường dẫn được sử dụng cho quá trình truyền thông trong PLC. Thông tin được truyền đi dưới dạng nhị phân.

Có các loại bus sau đây:

- Bus dữ liệu có chức năng truyền dữ liệu được xử lý bởi CPU.

- Bus địa chỉ có chức năng truyền địa chỉ của các vị trí trong bộ nhớ. Mỗi vị trí trong bộ nhớ có một địa chỉ xác định để CPU có thể truy vấn dữ liệu lưu tại vị trí này.

- Bus điều khiển có chức năng truyền các tín hiệu điều khiển bởi CPU, chẳng hạn như để thông báo cho các bộ nhớ về việc nhận dữ liệu từ một đầu vào hoặc dữ liệu đầu ra và thực hiện các tín hiệu xung nhịp để đồng bộ hóa hoạt động của PLC.

- Bus hệ thống vào/ra có chức năng truyền thông tin liên lạc giữa các đầu vào và đầu ra.

c) Bộ nhớ

Có chức năng lưu trữ dữ liệu với đơn vị nhỏ nhất là bit. Bộ nhớ là vùng chứa hệ điều hành (một phần mềm hệ thống giúp PLC có thể hoạt động được) và là nơi lưu trữ chương trình điều khiển của người sử dụng.

Bộ nhớ gồm các loại sau đây:

- ROM (Read–Only Memory) là loại bộ nhớ chỉ đọc. Đây là loại bộ nhớ có đặc điểm là nội dung bên trong nó không thể chỉnh sửa hoặc thay đổi được, do đó nó được dùng để lưu trữ các dữ liệu cố định. Nội dung lưu trong bộ nhớ này sẽ không bị mất hay thay đổi ngay cả khi mất nguồn. Mặt khác nó có thể đọc và ghi dữ liệu vào RAM bất cứ lúc nào.

- RAM (Random Access Memory) là loại bộ nhớ truy cập ngẫu nhiên có đặc điểm là dữ liệu chứa trong bộ nhớ này sẽ bị mất đi nếu như nguồn điện bị ngắt. Bộ nhớ của PLC là loại CMOSRAM, tiêu tốn năng lượng khá ít và được cấp pin dự phòng khi mất nguồn. Nhờ đó dữ liệu sẽ không bị mất.

- EPROM (Erasable Programmable Read–Only) là kiểu bộ nhớ ROM nhưng nội dung của nó có thể được ghi lại bằng cách chiếu vào nó tia cực tím sau khi đã tháo bỏ lớp bảo vệ.

- EEPROM (Electrically Erasable Programmable Read–Only Memory) là một kiểu EPROM có thể xóa bằng tín hiệu điện, tuy nhiên chỉ giới hạn một số lần

nhất định. Nó rất hữu dụng cho các thiết bị lưu trữ lâu dài mà không cần điện năng, lại cho phép ghi lại dữ liệu.

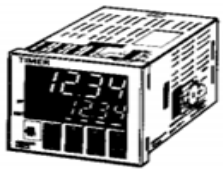
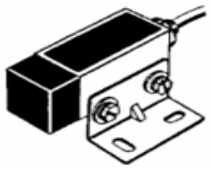
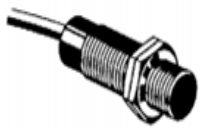
d) Khối vào/ra

Khối vào/ra đóng vai trò là mạch giao tiếp giữa hệ thống bên trong PLC với các thiết bị bên ngoài. Khối đầu vào nhận tín hiệu từ cảm biến, các thiết bị đầu vào và đưa vào CPU, khối đầu ra đưa tín hiệu điều khiển từ CPU ra cơ cầu chấp hành.

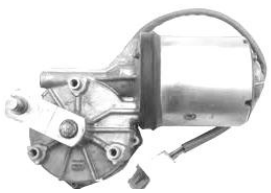
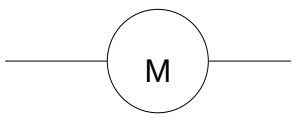
Khối vào/ra của PLC được kết nối trực tiếp với các thiết bị ngoại vi. Các mạch điện tử bên trong của PLC sử dụng dòng điện một chiều có điện áp phù hợp với mức TTL, tuy nhiên các khối vào/ra hoạt động với mức điện áp khác (24 VDC, 220 VAC), do đó ta phải chú ý khi giao tiếp giữa bên trong và bên ngoài PLC. Dưới đây là những yêu cầu cho đầu vào, đầu ra của PLC:

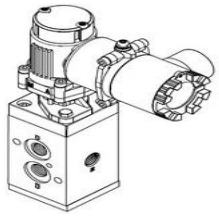
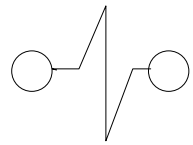
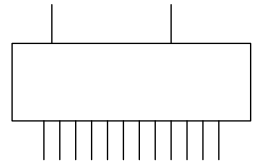
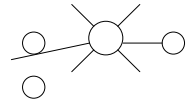
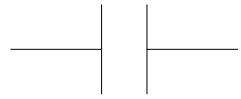
- Phải phù hợp với thiết bị bên ngoài về những thông số kỹ thuật điện.
- Nhiễu từ những thiết bị bên ngoài không làm ảnh hưởng đến CPU.
- Kết nối với thiết bị bên ngoài phải dễ dàng.
- Có thể theo dõi tình trạng của từng mối liên hệ giữa đầu vào và đầu ra (sử dụng đèn LED chỉ thị).

Bảng 1.3. Một số thiết bị vào

Công tắc giới hạn	Bộ đếm thời gian	Cảm biến ảnh	Bộ mã hoá quang học	Cảm biến tiệm cận
				

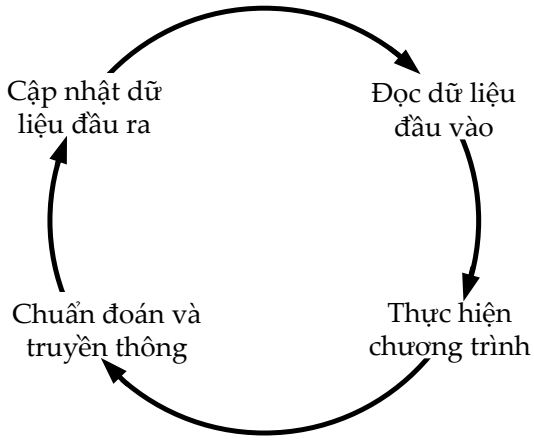
Bảng 1.4. Một số thiết bị ra

Hình ảnh	Tên thiết bị	Kí hiệu
	Động cơ	
	Bộ nung	

Hình ảnh	Tên thiết bị	Kí hiệu
	Van điện từ	
	LED chỉ thị	
	Bóng đèn	
	Khởi động từ	

e) Quá trình quét của CPU

Quá trình đọc các yếu tố đầu vào, thực hiện các chương trình và cập nhật các kết quả đầu ra được gọi là quá trình quét. Quá trình quét thường là một quá trình liên tục và tuần tự từ đọc trạng thái của đầu vào, đánh giá quá trình điều khiển logic cho đến việc cập nhật các kết quả đầu ra. Thời gian quét là yếu tố đặc trưng cho khả năng phản ứng với các yếu tố đầu vào và đánh giá quá trình điều khiển logic.



Hình 1.3. Vòng quét của CPU

Thời gian cần thiết của một vòng quét thay đổi tuỳ thuộc vào tốc độ xử lý của CPU và chiều dài của các chương trình người dùng. Việc sử dụng thêm các hệ vào/ra cũng làm tăng quá trình quét.

1.2.3. Ưu điểm của PLC

Có thể kể ra các ưu điểm của PLC như sau:

- Thời gian chuẩn bị hoạt động ngắn: Thiết kế kiểu mô-đun cho phép thích nghi nhanh với mọi chức năng điều khiển. Ngoài ra nó còn dễ dàng được sử dụng lại cho các ứng dụng khác.

- Độ tin cậy cao: Các linh kiện điện tử có tuổi thọ dài hơn các thiết bị cơ điện. Độ tin cậy của PLC ngày càng tăng, bảo dưỡng định kỳ thường không cần thiết còn với mạch rơle hay contactor thì việc bảo dưỡng định kỳ là cần thiết.

- Dễ dàng thay đổi chương trình: Việc thay đổi chương trình điều khiển được tiến hành đơn giản. Để thay đổi chương trình và các quy tắc điều khiển đang được sử dụng, người vận hành chỉ cần thay đổi các tập lệnh mà gần như không cần phải mắc nối lại dây (có thể vẫn phải nối lại nếu cần thiết). Nhờ đó hệ thống rất linh hoạt và hiệu quả.

- Khả năng tái tạo: Nếu dùng nhiều PLC với quy cách kỹ thuật giống nhau thì chi phí lao động sẽ giảm thấp hơn nhiều so với bộ điều khiển rơle, đó là do giảm được công lao động lắp ráp.

- Tiết kiệm không gian: PLC đòi hỏi ít không gian hơn so với bộ điều khiển rơle tương đương.

- Có nhiều chức năng: PLC có ưu điểm chính là có thể sử dụng cùng một thiết bị điều khiển cơ bản cho nhiều hệ thống điều khiển. Người ta thường dùng PLC cho các quá trình tự động vì thuận tiện trong tính toán, thay đổi chương trình và thay đổi các thông số.

Bảng 1.5. So sánh hệ điều khiển rơle và hệ điều khiển PLC

Hệ rơle	Hệ PLC
Nhiều bộ phận đã được chuẩn hoá	Thay đổi dễ dàng nhờ thiết kế thành các mô-đun
Kinh tế với các hệ thống nhỏ	Giá thành cao khi sử dụng với hệ thống nhỏ
Thời gian lắp đặt lâu	Lắp đặt đơn giản
Thay đổi khó khăn	Thay đổi nhanh quy trình điều khiển.
Kích thước lớn	Kích thước nhỏ gọn

Không có khả năng truyền thông	Có khả năng truyền thông giữa các PLC với nhau hoặc PLC với PC hay thiết bị có khả năng truyền thông khác

Như vậy có thể nói PLC là một bộ điều khiển rất thuận tiện để sử dụng và lắp đặt. Từ việc thay thế cách nối dây phức tạp như trước đây, các bộ điều khiển logic khả trình đã trở lên linh hoạt hơn. Sau khi cài đặt, có thể điều khiển chương trình bằng tay hoặc tự động, thay đổi để đáp ứng các yêu cầu về điều khiển. Kết nối vật lý giữa các thiết bị vào/ra cũng có thể dễ dàng thay đổi.

1.2.4. Phân loại và ứng dụng của PLC

Phân loại

Có thể phân loại PLC dựa trên các tiêu chí bao gồm: chức năng, số lượng đầu vào/ra, chi phí và kích thước vật lý. Trong số các tiêu chí này, số lượng đầu vào/ra là yếu tố chủ yếu.

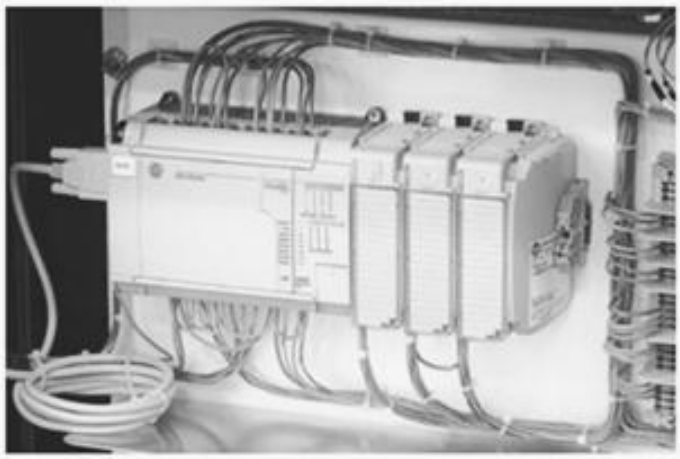
- PLC cỡ nhỏ có 128 đầu vào/ra và bộ nhớ 2 kbytes.
- PLC cỡ trung bình có 2048 đầu vào/ra và bộ nhớ 3 kbytes.
- PLC cỡ lớn có 8192 đầu vào/ra và bộ nhớ lên tới 75 kbytes.

Hình 1.4. Một số loại PLC

Khi lựa chọn PLC, cần phải lựa chọn cho phù hợp với yêu cầu và mục đích sử dụng. Tuy nhiên, để có thể mở rộng hay nâng cấp ứng dụng thì nên lựa chọn bộ PLC có kích thước lớn hơn kích thước nhu cầu.

PLC có 3 ứng dụng chính đó là: Ứng dụng thực hiện một nhiệm vụ duy nhất (Single – Ended), ứng dụng thực hiện chức năng đa vụ (Multitask) và ứng dụng quản lý điều khiển (Control Management). Hình 1.7 là ví dụ ứng dụng thực hiện

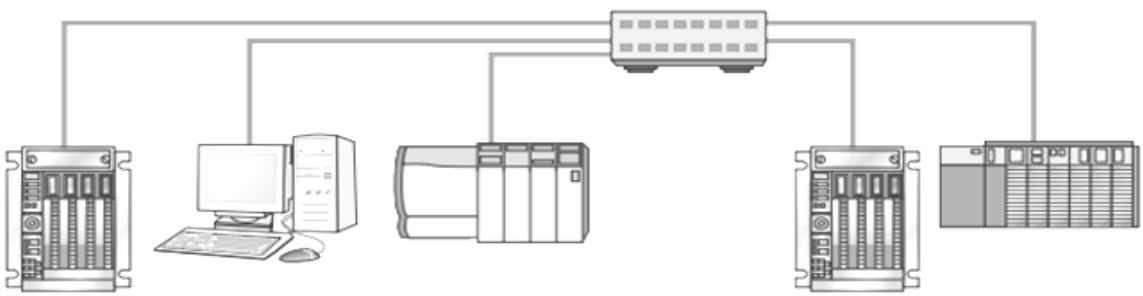
một nhiệm vụ duy nhất, tức chỉ điều khiển một quá trình và không sử dụng để giao tiếp với máy tính hoặc bộ PLC khác.



Hình 1.5. PLC thực hiện chức năng đơn vụ

Loại PLC thực hiện chức năng đa vụ sẽ thực hiện một số quá trình điều khiển. Số lượng ngõ vào/ra là yếu tố quan trọng để phân biệt loại này. Ngoài ra, loại PLC này là một hệ thống phụ trong một quá trình lớn hơn, và phải liên kết với một bộ PLC khác hay một máy tính trung tâm.

PLC ứng dụng trong quá trình quản lý điều khiển thực hiện chức năng quản lý và điều khiển một hệ thống nhiều PLC. Loại PLC này đòi hỏi một bộ xử lý trung tâm CPU có tốc độ lớn, có thể liên kết với các PLC và máy tính khác, có khả năng kết nối và điều khiển tất cả các PLC theo đúng địa chỉ và yêu cầu nhiệm vụ.



Hình 1.6. PLC thực hiện chức năng quản lý điều khiển

Các ứng dụng của PLC

Ngày nay chúng ta có thể thấy PLC trong hàng nghìn ứng dụng công nghiệp. Chúng được sử dụng trong công nghiệp hoá chất, công nghiệp chế biến dầu, công nghiệp thực phẩm, công nghiệp cơ khí, công nghiệp xử lý nước và chất thải, công nghiệp dược phẩm, công nghiệp dệt may, nhà máy điện hạt nhân, trong công nghiệp khai khoáng, trong giao thông vận tải, trong quân sự, trong các hệ

thống đảm bảo an toàn, trong các hệ thống vận chuyển tự động, điều khiển robot, điều khiển máy công cụ CNC.

Các PLC có thể được kết nối với các máy tính để truyền, thu thập và lưu trữ dữ liệu bao gồm cả quá trình điều khiển bằng thống kê, quá trình đảm bảo chất lượng, chẩn đoán sự cố trực tuyến, thay đổi chương trình điều khiển từ xa. Ngoài ra PLC còn được dùng trong hệ thống quản lý năng lượng nhằm giảm giá thành và cải thiện môi trường điều khiển trong các các hệ thống phục vụ sản xuất, trong các dịch vụ và các văn phòng công sở.

CÂU HỎI ÔN TẬP CHƯƠNG 1

<u>Chú ý:</u> T ký hiệu của TRUE và F ký hiệu của FALSE

1. PLC là từ viết tắt của:
   a. Personal Logic Computer
   b. Programmable Local Computer
   c. Personal Logic Controller
   d. Programmable Logic Controller
2. Đầu ra Transistor từ một PLC:
   (i) Chỉ được sử dụng cho chuyển đổi DC.
   (ii) Chỉ được cách ly với tải đầu ra nhờ sử dụng IC cách quang.
   Lựa chọn đáp án đúng:
   a. (i) T (ii) T
   b. (i) T (ii) F
   c. (i) F (ii) T
   d. (i) F (ii) F
3. Một đầu ra rơle trong PLC:
   (i) Chỉ được sử dụng để chuyển đổi DC.
   (ii) Có thể chịu được quá tải trong thời gian ngắn.
   Chọn câu trả lời đúng:
   a. (i) T (ii) T
   b. (i) T (ii) F
   c. (i) F (ii) T
   d. (i) F (ii) F
4. Một triac đầu ra từ PLC:
   (i) Chỉ được sử dụng cho đầu ra AC.
   (ii) Chỉ được cách ly với tải đầu ra nhờ sử dụng IC cách quang.
   Chọn đáp án đúng:
   a. (i) T (ii) T
   b. (i) T (ii) F
   c. (i) F (ii) T

d. (i) F (ii) F

5. Mức điện áp được cung cấp và sử dụng trong PLC là ?

   a. 5 V

   b. 24 V

   c. 110 V

   d. 240 V

6. Lý do sử dụng IC cách quang trên các mô-đun vào/ra:

   (i) Có chức năng giống như cầu trì có tác dụng ngắt mạch điện khi quá áp hoặc quá dòng.

   (ii) Cách ly CPU với điện áp và dòng điện lớn.

   Hãy chọn đáp án đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F

7. Vẽ sơ đồ khối của một PLC và giải thích chức năng của từng khối.

8. Nêu rõ đặc điểm chính của rơle, transistor và triac.

9. Bao nhiêu bit có thể được chứa trong 2K đơn vị bộ nhớ?

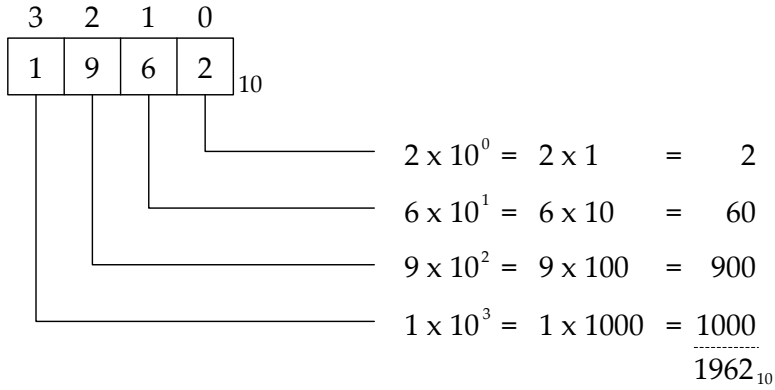
Chương 2
CÁC HỆ THỐNG SỐ

2.1. HỆ THẬP PHÂN

Hệ thập phân là một hệ đếm cơ số 10 và được con người sử dụng trong cuộc sống hàng ngày. Trong hệ thập phân, 10 ký tự khác nhau được dùng để biểu đạt 10 giá trị riêng biệt (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), tức là 10 con số.

Hệ thập phân là một hệ đếm dựa vào vị trí của các con số (*positional numeral system*). Vị trí của mỗi con số mô tả một phép nhân cơ số 10 với con số ở vị trí đó, và mỗi con số dịch về bên trái có giá trị gấp mười lần con số kế bên phải.

Trên Hình 2.1 mô tả cách tính giá trị của một số trong hệ thập phân tương ứng với từng giá trị tại vị trí tương ứng của nó. [1]

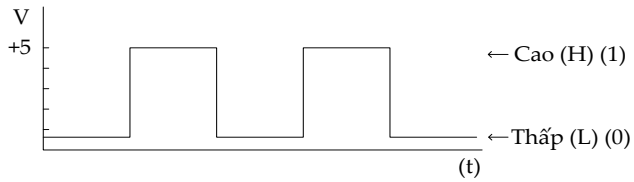


Hình 2.1. Trọng số của hệ thập phân

2.2. HỆ NHỊ PHÂN

Hệ nhị phân (hệ đếm cơ số 2) là một hệ đếm dùng hai ký tự 0 và 1 để biểu đạt một giá trị số (bằng tổng số các lũy thừa của 2). Hai ký tự này cũng có thể được dùng để biểu đạt hai giá trị hiệu điện thế tương ứng (có hiệu điện thế, hoặc hiệu điện thế cao là 1, hoặc thấp là 0). Do có ưu điểm tính toán đơn giản, dễ dàng thực hiện về mặt vật lý nên hệ nhị phân là một hệ cơ bản trong các máy tính đương thời và được áp dụng khá dễ dàng cho PLC. Bảng 2.1 mô tả cách

chuyển đổi giữa các hệ thống số cơ bản: thập phân (cơ số 10), bát phân (cơ số 8), hệ thập lục phân (cơ số 16) và nhị phân (cơ số 2). Lưu ý rằng tất cả các hệ thống số đánh số bắt đầu từ 0.



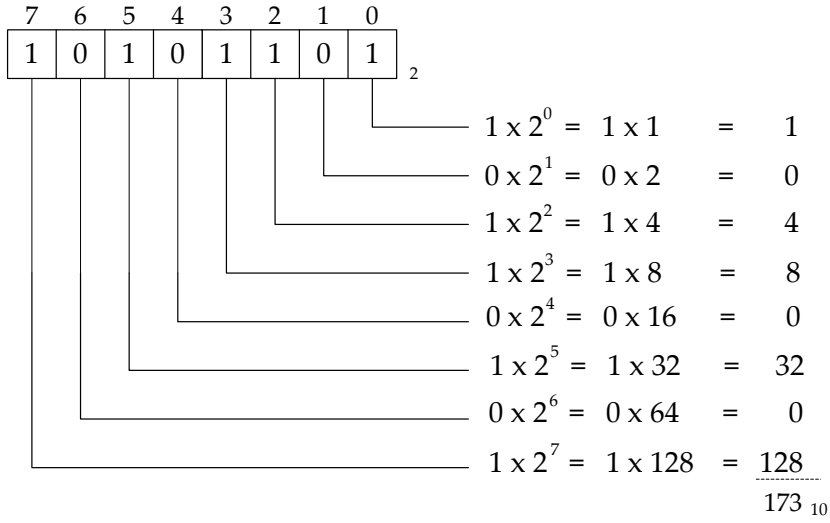
Hình 2.2. Tín hiệu số biểu diễn giá trị hiệu điện thế

Bảng 2.1. So sánh các hệ số

Hệ thập phân	Hệ bát phân	Hệ thập lục phân	Hệ nhị phân
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111

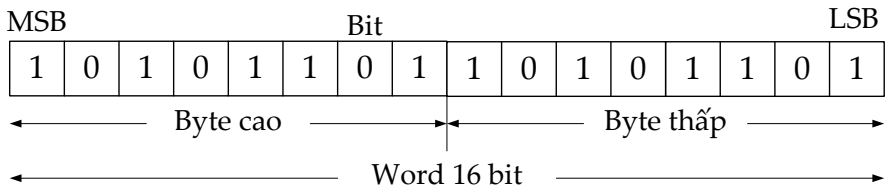
Để biểu diễn một số dạng nhị phân, chúng ta có thể sử dụng một số kiểu biểu diễn sau đây:

- 100101 binary
- 100101b (“b” - lấy chữ đầu của *binary* trong tiếng Anh)
- bin 100101 (“*bin*” cũng được lấy từ binary)
- 100101_2 (ký hiệu 2 viết nhỏ phía dưới ám chỉ gốc nhị phân)

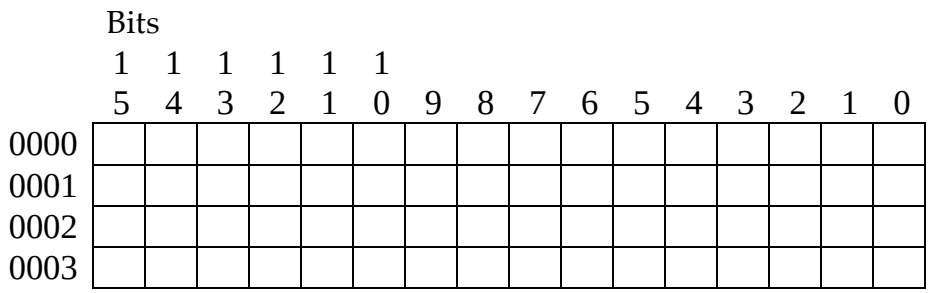


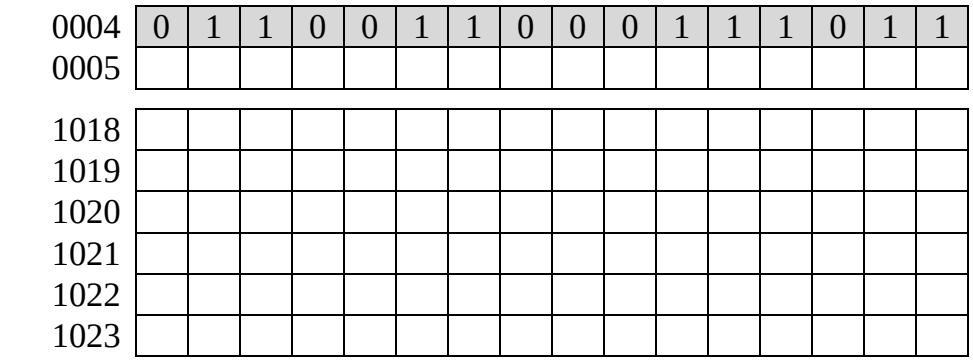
Hình 2.3. Chuyển đổi hệ nhị phân sang hệ thập phân

Mỗi chữ số của một số nhị phân tương ứng với 1 bit. Trong PLC yếu tố xử lý bộ nhớ bao gồm hàng trăm hoặc hàng ngàn địa chỉ khác nhau. Những địa chỉ này được gọi là "word". Mỗi word có khả năng lưu trữ dữ liệu dưới dạng nhị phân (các bit). Số bit của một word có thể lưu trữ phụ thuộc vào loại PLC được sử dụng. Phổ biến nhất là loại word gồm 16-bit và 32-bit. Một nhóm 8 bit tạo thành 1 byte và một nhóm của hai hoặc nhiều byte tạo thành 1 word. Hình 2.4 minh họa một word 16-bit của 2 byte. Mỗi bit trong một word có thể biểu thị một trong hai trạng thái là 1 (ON) hoặc 0 (OF). Bộ nhớ PLC là tổ hợp của nhiều bit, word đơn hoặc word kép. Nếu dung lượng của bộ nhớ là 1K word, nó có thể lưu trữ 1024 word hoặc 16.384 bit thông tin nếu sử dụng word 16-bit, hoặc 32.768 bit thông tin nếu sử dụng word 32-bit.



Hình 2.4. Một word 16 bit

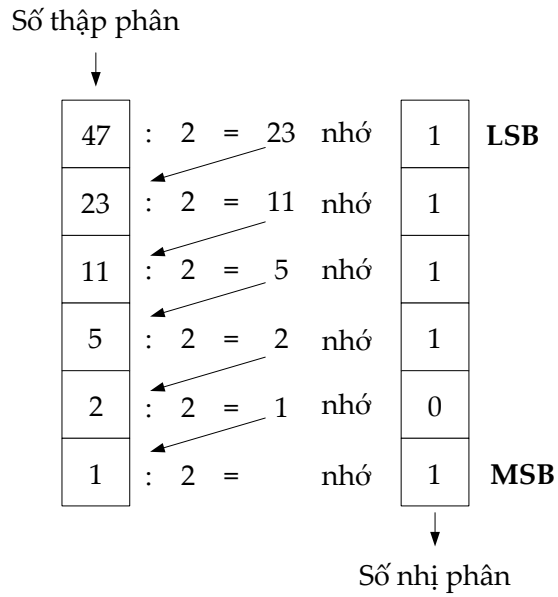




Hình 2.5. 1K word bộ nhớ

Phương pháp biến đổi một số nguyên ở hệ thập phân sang hệ nhị phân tương đương có thể được tiến hành bằng cách chia số này cho 2 và số dư được viết xuống hàng kết quả. Kết quả lại tiếp tục được chia cho 2 và số dư lại được viết xuống hàng chục. Phương thức này được thực hiện liên tục cho đến khi thương số của phép chia là 0.

Hình 2.6 biểu diễn cách chuyển đổi số 47 từ hệ thập phân sang hệ nhị phân.



Hình 2.6. Chuyển đổi số thập phân sang nhị phân

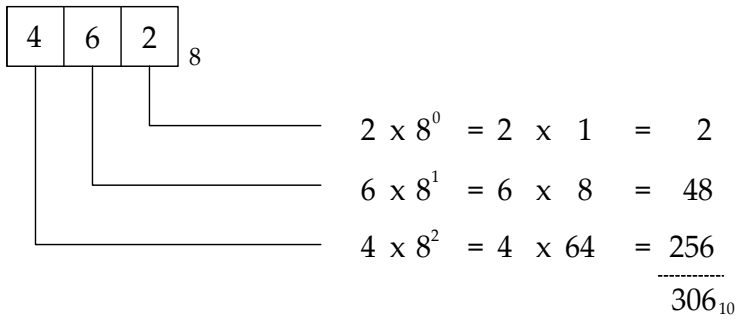
Hệ nhị phân tuy chỉ có hai chữ số nhưng nó có thể biểu diễn bất kỳ một đại lượng nào đó như ở hệ thập phân. Tất cả PLC đều làm việc với hệ nhị phân. Bộ vi xử lý là một thiết bị kỹ thuật số làm việc với hai số 0 và 1 (số nhị phân). [1]

2.3. HỆ BÁT PHÂN

Để biểu diễn một số trong hệ nhị phân đòi hỏi nhiều chữ số hơn trong hệ thập phân. Nếu quá nhiều chữ số nhị phân có thể trở nên khó khăn trong việc đọc hoặc viết. Để giải quyết vấn đề này, người ta sử dụng một hệ thống số khác tương đương nhưng cách viết lại đơn giản hơn.

Hệ bát phân hay hệ cơ số 8, là một hệ đếm khá phổ biến, gồm các chữ số từ 0 tới 7. Khi đó 8 bit dữ liệu được sử dụng để có thể tạo thành một byte thông tin. Hệ bát phân rất thuận tiện khi xử lý với số nhị phân lớn. Ví dụ trong Bảng 2.2, một chữ số bát phân có thể được sử dụng để thể hiện ba chữ số nhị phân.

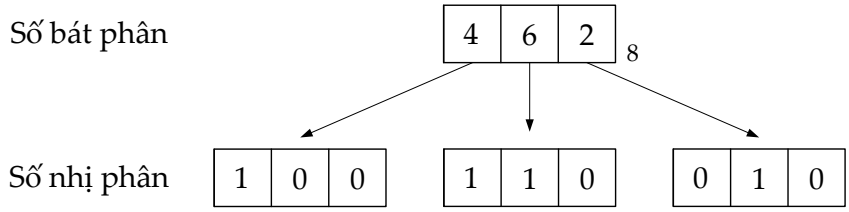
Hình 2.7 biểu diễn cách biến đổi số bát phân 462 tương đương với số thập phân của nó là 306. Cách chuyển đổi một số từ hệ bát phân sang hệ nhị phân cũng tương đối dễ dàng. Ví dụ, số bát phân 462 được chuyển đổi tương đương thành số nhị phân bằng cách ghép các nhóm 3-bit, như minh họa trong Hình 2.8. Ở đây số bát phân 462 dễ đọc và viết hơn nhiều so với sô nhị phân tương đương của nó. [1]



Hình 2.7. Chuyển đổi số bát phân sang thập phân

Bảng 2.2. Số nhị phân và bát phân tương ứng

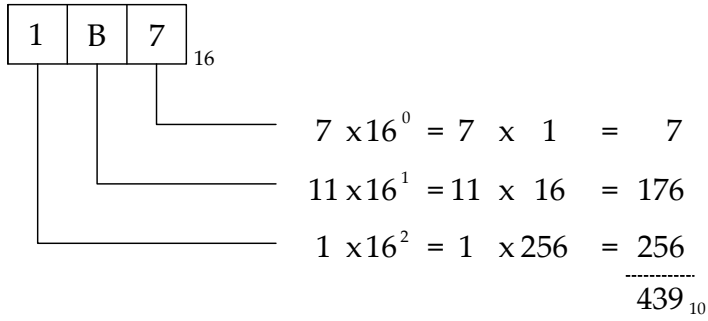
Nhị phân	Bát phân
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7



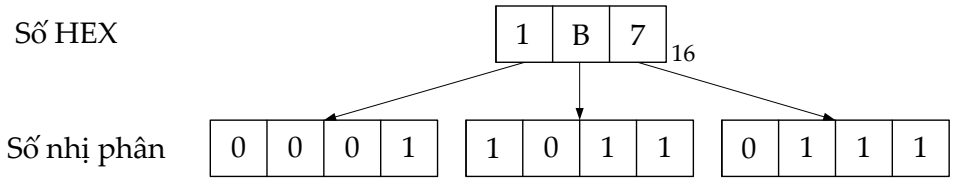
Hình 2.8. Chuyển đổi số bát phân sang nhị phân

2.4. HỆ THẬP LỤC PHÂN

Hệ thập lục phân (hex) là hệ đếm cơ số 16, bao gồm số từ 0 tới 9 và các chữ số từ A đến F (không phân biệt chữ hoa hay chữ thường). Hệ thập lục phân được sử dụng nhiều trong các bộ điều khiển lập trình vì một word dữ liệu bao gồm 16-bit dữ liệu, hoặc 2 byte 8-bit, do đó khi sử dụng hệ thập lục phân cũng như hệ bát phân, sẽ dễ dàng đọc và ghi hơn so với hệ nhị phân. Các hệ thập lục phân cho phép biểu diễn một số lượng lớn các bit nhị phân trong một không gian nhỏ, chẳng hạn như trên một màn hình máy tính hoặc thiết bị hiển thị PLC. Cách chuyển đổi hệ thập lục phân sang thập phân và ngược lại tương tự như cách chuyển đổi giữa hệ nhị phân và bát phân. Để chuyển đổi một số thập lục phân sang thập phân, các chữ số thập lục phân trong các cột được nhân với lũy thừa của 16 với số mũ là số thứ tự của chữ số đó. Hình 2.9 minh họa cách chuyển đổi số 1B7 (hex) sang hệ thập phân.



Hình 2.9. Chuyển đổi số HEX sang số thập phân



Hình 2.10. Chuyển đổi số HEX sang số nhị phân

Bảng 2.3. Bảng tương đương các hệ số

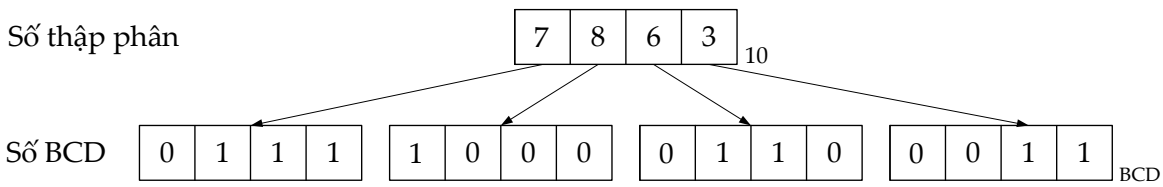
Thập lục phân	Nhị phân	Thập phân
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Cách chuyển đổi từ số thập lục phân sang nhị phân cũng tương đối dễ dàng bằng cách viết tương đương mỗi chữ số hex ứng với 4-bit nhị phân, như minh họa trong Hình 2.10. [1]

2.5. HỆ NHỊ PHÂN MÃ HOÁ THẬP PHÂN (BCD)

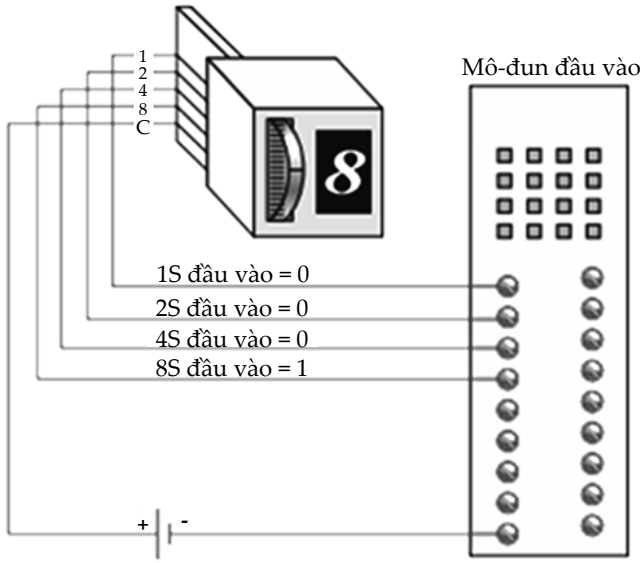
Hệ nhị phân mã hóa thập phân, hay gọi tắt là BCD là 1 hệ mã hóa thuận tiện để xử lý một số lượng lớn tín hiệu ở đầu vào hoặc đầu ra của PLC. BCD cung cấp một cách thức để chuyển đổi một mã dễ xử lý với con người (hệ thập phân) sang một mã dễ xử lý với các thiết bị (hệ nhị phân).

Hệ BCD sử dụng 4 bit để biểu diễn cho mỗi chữ số thập phân ở dạng nhị phân. Tên BCD được đặt ở bên phải của chữ số hàng đơn vị. Mã hóa BCD của số thập phân 7863 được thể hiện trong Hình 2.11.



Hình 2.11. Chuyển đổi một số nhị phân sang số hệ BCD.

Núm điều chỉnh bằng tay là một thiết bị đầu vào sử dụng mã BCD. Các chữ số trên bảng điều khiển được kết nối với thiết bị. Khi ấn các chữ số trên bảng điều khiển từ 0 đến 9, thông qua thiết bị chuyển đổi sẽ đưa ra kết quả 4 bit tương đương với dữ liệu BCD. Ở ví dụ này, khi ấn số 8, các bit đầu vào tương đương là 1000. [1]



Hình 2.12. Mã BCD trong giao tiếp của núm điều chỉnh bằng tay

2.6. MÃ GRAY

Mã Gray là một dạng của hệ thống số nhị phân, trong đó hai giá trị liên tiếp chỉ khác nhau một chữ số. Lúc đầu, mã Gray được phát minh với mục đích ngăn ngừa tín hiệu ngõ ra không chính xác của các bộ chuyển mạch cơ điện. Ngày nay, mã Gray được sử dụng rộng rãi để sửa lỗi trong những phương tiện liên lạc số, ví dụ như truyền hình kỹ thuật số mặt đất và một vài hệ thống truyền hình cáp.

Mã Gray được phát minh cùng với sự ra đời của điện thoại, khi mà sự chuyển đổi số điện thoại là liên tục. Mã Gray có lợi cho mỗi "count" (mỗi sự chuyển tiếp từ một số) chỉ có một số thay đổi. Bảng 2.4 cho thấy sự so sánh tương đương giữa mã Gray và mã nhị phân.

Trong hệ nhị phân, ví dụ quá trình chuyển đổi từ số nhị phân 0111_2 đến 1000_2 (số thập phân 7 đến 8) liên quan đến sự thay đổi tất cả bốn chữ số. Loại thay đổi này làm tăng khả năng xuất hiện lỗi trong các mạch kỹ thuật số nhất định. Vì lý do đó mã Gray được coi là một mã có thể làm giảm thiểu lỗi này bởi vì tại một thời

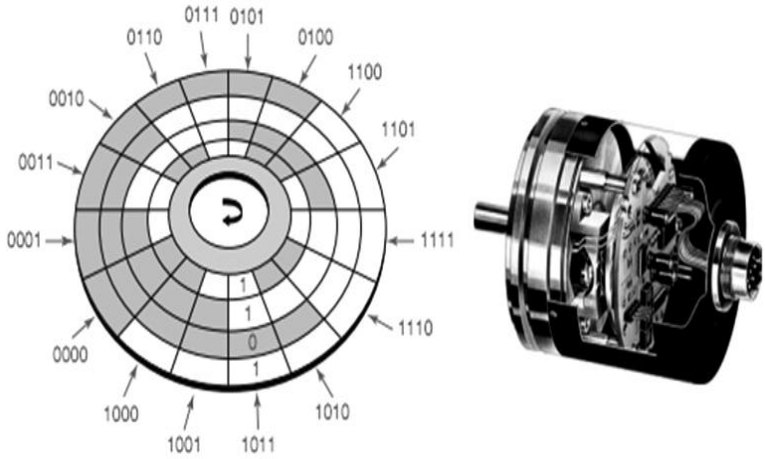
điểm, khi chuyển đổi giữa số nhị phân, tốc độ của quá trình chuyển đổi mã Gray là nhanh hơn đáng kể hơn so với các mã như BCD.

Bảng 2.4. Bảng so sánh hệ nhị phân và mã Gray

Mã Gray	Nhị phân
0000	0000
0001	0001
0011	0010
0010	0011
0110	0100
0111	0101
0101	0110
0100	0111

Mã Gray được sử dụng với các bộ mã hóa vị trí để kiểm soát chính xác chuyển động của robot, máy công cụ và các máy sử dụng động cơ servo.

Hình 2.13 mô tả một ổ đĩa mã hóa quang học sử dụng mã Gray 4-bit để phát hiện những thay đổi vị trí góc. Trong ví dụ này, ổ đĩa mã hóa được gắn liền với một trục quay và đầu ra là một tín hiệu mã Gray kỹ thuật số được sử dụng để xác định vị trí của trục. [1]



Hình 2.13. Đĩa mã hoá quang học

2.7. MÃ ASCII

Chuẩn mã trao đổi thông tin Hoa Kì hay mã ASCII (American Standard Code for Information Interchange) là bảng mã dựa trên bảng chữ cái La Tinh được dùng trong tiếng Anh hiện đại và các ngôn ngữ Tây Âu khác. Nó thường được dùng để

hiển thị văn bản trong máy tính và các thiết bị thông tin khác. Nó cũng được dùng bởi các thiết bị điều khiển làm việc với văn bản.

Các chữ cái in được theo thứ tự trong ASCII là:

s_p!"#$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ `abcdefghijklmnopqrstuvwxyz{|}~

(Với kí tự đầu tiên là khoảng trắng).

Các kí tự từ 0 đến 32 theo hệ thập phân không thể in ra màn hình. Các kí tự đó chỉ có thể in được trong môi trường DOS gồm một số hình như trái tim, mặt cười, hình tam giác,... Một số ký tự đặc biệt khi in ra màn hình sẽ thực hiện lệnh như: kêu tiếng bip với kí tự BEL, xuống hàng với kí tự LF,...

Trong bảng mã ASCII chuẩn có 128 kí tự. Trong bảng mã ASCII mở rộng có 255 kí tự bao gồm cả 128 kí tự trong mã ASCII chuẩn. Các kí tự sau là các phép toán, các chữ có dấu và các kí tự để trang trí.

Cũng như các mã máy tính biểu diễn kí tự khác, mã ASCII quy định mối tương quan giữa kiểu bit số với kí hiệu/biểu tượng trong ngôn ngữ viết, vì vậy cho phép các thiết bị số liên lạc với nhau và xử lí, lưu trữ, trao đổi thông tin hướng kí tự. Bảng mã kí tự ASCII hoặc các mở rộng tương thích được dùng trong hầu hết các máy tính thông thường, đặc biệt là máy tính cá nhân và máy trạm làm việc.

ASCII chính xác là mã 7-bit, tức là dùng kiểu biểu diễn với 7 số nhị phân (thập phân từ 0 đến 127) để biểu diễn thông tin về kí tự. Khi mã ASCII được giới thiệu, nhiều máy tính dùng nhóm 8-bit (byte hay bộ tám) làm đơn vị thông tin nhỏ nhất; bit thứ 8 thường được dùng làm bit chẵn-lẻ (parity) để kiểm tra lỗi trên các đường thông tin hoặc kiểm tra chức năng đặc hiệu theo thiết bị. Các máy không dùng chẵn-lẻ thường thiết lập bit thứ 8 là zero, nhưng một số thiết bị như máy PRIME chạy PRIMOS thiết lập bit thứ 8 có giá trị là 1.

ASCII được công bố làm tiêu chuẩn lần đầu vào năm 1963 bởi Hiệp hội tiêu chuẩn Hoa Kì (American Standards Association, ASA), sau này đổi thành ANSI. Có nhiều biến thể của ASCII, hiện tại phổ biến nhất là ANSI X3.4-1986, cũng được tiêu chuẩn hoá bởi Hiệp hội những nhà sản xuất máy tính châu Âu (European Computer Manufacturers Association) ECMA-6, ISO/IEC 646:1991 Phiên bản tham khảo quốc tế, ITU-T Khuyến cáo T.50 (09/92), và RFC 20 (Request for Comments). ASCII được xem là tiêu chuẩn phần mềm thành công nhất từng được công bố từ trước tới nay. [1]

2.8. CÁC PHÉP TÍNH TRONG HỆ NHỊ PHÂN

Các phép tính là một công việc quan trọng của CPU. Các phép tính bao gồm cộng, trừ, nhân và chia. Đối với hệ nhị phân, cũng tương tự như các phép tính được áp dụng trong các hệ khác.

– **Phép cộng**

Phép tính đơn giản nhất trong hệ nhị phân là tính cộng. Cộng hai đơn vị trong hệ nhị phân được làm như sau:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 (nhớ 1 lên hàng thứ 2)

Cộng hai số "1" với nhau tạo nên giá trị "10", tương đương với giá trị 2 trong hệ thập phân. Điều này xảy ra tương tự trong hệ thập phân khi hai số đơn vị được cộng vào với nhau. Nếu kết quả bằng hoặc cao hơn giá trị cơ số (10), giá trị của con số ở hàng tiếp theo được cộng thêm:

5 + 5 = 10

7 + 9 = 16

Hiện tượng này được gọi là "nhớ" hoặc "mang sang", khi tổng số lớn hơn cơ số của hệ số, bằng cách "nhớ" 1 sang vị trí bên trái. Phương thức "nhớ" cũng hoạt động tương tự trong hệ nhị phân:

```
    1   1   1   1   1   Nhớ
        0   1   1   0   1
+       1   0   1   1   1
---------------------------
    1   0   0   1   0   0
```

Trong ví dụ trên, hai số được cộng với nhau: 01101_2 (13 thập phân) và 10111_2 (23 thập phân). Hàng trên cùng biểu đạt những giá trị nhớ, hoặc mang sang. Bắt đầu bằng cột cuối cùng bên phải, $1 + 1 = 10_2$. Giá trị 1 được mang sang cột bên trái, và 0 được viết vào hàng tổng phía dưới (cột cuối cùng bên phải). Cột thứ hai tính từ bên phải được cộng tiếp theo: $1 + 0 + 1 = 10_2$; số 1 lại được nhớ và mang sang, số 0 được viết xuống dưới cùng. Cột thứ ba: $1 + 1 + 1 = 11_2$. Lần này 1 được nhớ và mang sang cột bên cạnh và 1 được viết xuống hàng dưới cùng. Tiếp tục khai triển theo quy luật trên cho chúng ta đáp án cuối cùng là 100100_2.

– **Phép trừ**

Phép trừ theo quy tắc tương tự:

$0 - 0 = 0$

$0 - 1 = -1$ (mượn)

$1 - 0 = 1$

$1 - 1 = 0$

Một đơn vị nhị phân được trừ với một đơn vị nhị phân khác như sau:

```
        *       *   *   *   Hình sao đánh dấu các cột phải mượn
    1   1   0   1   1   1   0
-           1   0   1   1   1
-----------------------------------------------
    1   0   1   0   1   1   1
```

– **Phép nhân**

Phép nhân trong hệ nhị phân cũng tương tự như phương pháp làm trong hệ thập phân. Hai số A và B được nhân với nhau bằng cách nhân B với số một con số trong A và viết xuống một hàng mới, mỗi hàng mới phải chuyển dịch 1 vị trí sang bên trái. Tổng của các tích đơn lẻ này cho ta kết quả tích số cuối cùng. Trong đó: 1x A= A; 0xA=0

Ví dụ, hai số nhị phân 1011_2 và 1010_2 được nhân với nhau như sau:

```
                1   0   1   1   (A)
            x   1   0   1   0   (B)
 ---------------------------
                0   0   0   0
+           1   0   1   1
+       0   0   0   0
+   1   0   1   1
 -----------------------------------------------
=   1   1   0   1   1   1   0
```

– **Phép chia**

Phép chia nhị phân cũng tương tự như phép chia trong hệ thập phân. Ở đây ta có số bị chia là 11011_2 (27 trong số thập phân) và số chia là 101_2 (5 trong số thập phân). Cách làm tương tự với cách làm trong số thập phân. Ta lấy 3 số đầu của số

bị chia là 110_2 để chia với số chia, tức là 101_2, được 1, viết lên trên hàng kẻ. Kết quả này được nhân với số chia, và tích số được trừ với 3 số đầu của số bị chia. Số tiếp theo là một con số 1 được hạ xuống để tạo nên một dãy số có 3 con số, tương tự với số lượng các con số của số chia:

```
    1
  _____________
  1 1 0 1 1 | 1 0 1
- 1 0 1
  -------
  0 0 1
```

Quy luật trên được lặp lại với những hàng số mới, tiếp tục cho đến khi tất cả các con số trong số bị chia đã được dùng hết:

```
    1 0 1
  _____________
  1 1 0 1 1 | 1 0 1
- 1 0 1
  ------
  0 1 1
- 0 0 0
  ------
  1 1 1
- 1 0 1
  ------
    1 0
```

Kết quả của phép chia 11011_2 chia cho 101_2 là 101_2, như liệt kê phía trên đường kẻ, trong khi số dư còn lại được viết ở hàng cuối là 10_2. Trong hệ thập phân, 27 chia cho 5 được 5, dư 2.

– **Phép toán thao tác bit trong hệ nhị phân**

Trong hệ nhị phân cũng sử dụng những thao tác như: Toán tử AND (cả hai đơn vị thực hiện phép toán phải là 1 thì mới cho kết quả 1), toán tử OR (một trong hai đơn vị thực hiện phép toán là 1 thì cho kết quả là 1), và toán tử XOR (nếu 2 bit thực hiện phép toán mà khác nhau thì kết quả bằng 1, giống nhau thì bằng 0) có thể được thi hành với từng cặp bit tương đồng trong một cặp số của hai số nhị phân. Thao tác của toán tử lôgic NOT (phép đổi ngược, 0 thành 1 và ngược lại) có

thể được thi hành trên từng bit một trong một con số nhị phân. Lấy ví dụ, loại bỏ bit cuối cùng bên phải trong một số nhị phân (còn được gọi là phép toán chuyển vị nhị phân – binary shifting) tương đương với phép chia 2 trong hệ thập phân, vì khi thực hiện phép tín này giá trị của số bị chia sẽ giảm một nửa.

– **Phép so sánh trong hệ nhị phân**

Phép so sánh là so sánh mức độ tương đối của hai đại lượng. Bộ phận so sánh trong PLC được sử dụng để so sánh các dữ liệu được lưu trữ trong hai word (hoặc hai số được nhập vào). Đôi khi, các dữ liệu này cần phải được kiểm soát khi chúng là bé hơn, bằng hoặc lớn hơn so với các giá trị hay dữ liệu khác đã được thiết lập sử dụng trong các ứng dụng, chẳng hạn như bộ đếm thời gian và giá trị bộ đếm. [1]

CÂU HỎI ÔN TẬP CHƯƠNG 2

1. Chuyển đổi sang số thập phân các số nhị phân: 000011; 111111; 001101.
2. Chuyển đổi sang số nhị phân các số: 100; 146; 255.
3. Chuyển sang hệ thập phân các số bát phân: 9F; D53; 67C.
4. Chuyển đổi sang số thập lục phân các số thập phân: 14; 81; 2562.
5. Chuyển đổi sang số nhị phân các số thập lục phân: E; 1D; A65.
6. Chuyển đổi sang số bát phân các số thập phân: 372; 14; 2540.
7. Chuyển đổi sang số thập phân các số bát phân: 20; 265; 400.
8. Chuyển đổi sang số nhị phân các số bát phân: 270; 102; 673.
9. Chuyển đổi sang số BCD các số thập phân: 20; 35; 92.
10. Chuyển đổi các số thập phân sang các số nhị phân ở dạng 8-bit: -1; -35; -125.

Chương 3
THIẾT BỊ VÀO/RA

3.1. THIẾT BỊ ĐẦU VÀO

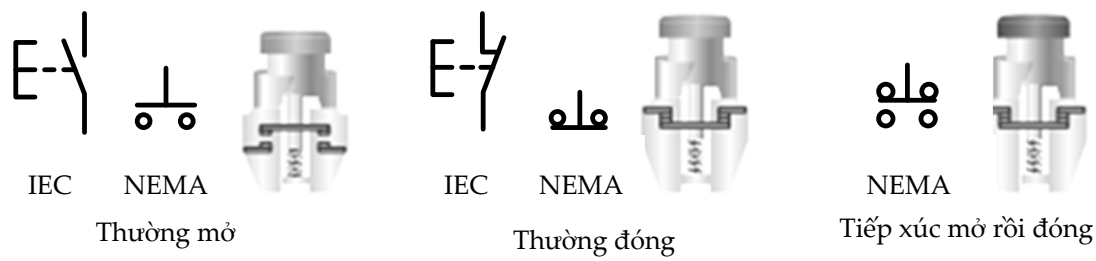
3.1.1. Nút nhấn

Các nút nhấn bao gồm các nút nhấn lật trạng thái, nút bấm, cầu dao, công tắc chọn lựa. Các nút nhấn là dạng thiết bị đầu vào thường gặp nhất trong thực tế. Khi hoạt động các nút nhấn sẽ đóng/mở các tiếp điểm. Hình 3.1 là hình ảnh của các loại nút nhấn thường thấy, bao gồm:

- Nút nhấn thường mở NO (Normally Open) sẽ khép kín mạch điện khi nó được nhấn và mạch điện hở khi nó được nhả.

- Nút nhấn thường đóng NC (Normally Close) hoạt động ngược lại so với nút nhấn thường mở.

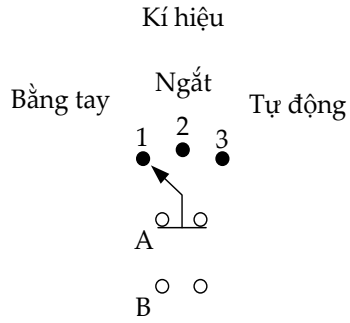
- Nút nhấn loại tiếp xúc ngắt rồi đóng (Break-Before-Make Pushbutton) là kiểu nút nhấn mà phía trên là tiếp điểm thường đóng và phía dưới là tiếp điểm thường mở. Khi được nhấn, tiếp điểm phía trên sẽ mở và tiếp điểm phía dưới sẽ đóng.



Hình 3.1. Hình dạng và ký hiệu các loại nút nhấn

Công tắc lựa chọn cũng là một loại công tắc thông dụng. Sự khác biệt chính giữa loại công tắc lựa chọn và nút nhấn chính là cấu tạo hoạt động. Hình 3.2 là hình ảnh của công tắc lựa chọn 3 vị trí. Các vị trí được lựa chọn bằng cách vặn núm điều chỉnh theo chiều phải hoặc trái.

Vị trí	Contactor	
	A	B
1	X	
2		
3		X



Hình 3.2. Công tắc lựa chọn 3 vị trí

3.1.2. Cảm biến

Các cảm biến được sử dụng để phát hiện hay để đo độ lớn của một đại lượng nào đó. Nguyên lý hoạt động của các cảm biến là chuyển đổi các đại lượng cơ học, từ, nhiệt, quang hay sự thay đổi hóa học thành các đại lượng điện áp hay dòng điện. Các cảm biến thường được phân loại theo các đại lượng mà chúng đo và đóng một vai trò quan trọng trong điều khiển quá trình sản xuất hiện nay.

3.1.2.1. Cảm biến tiệm cận

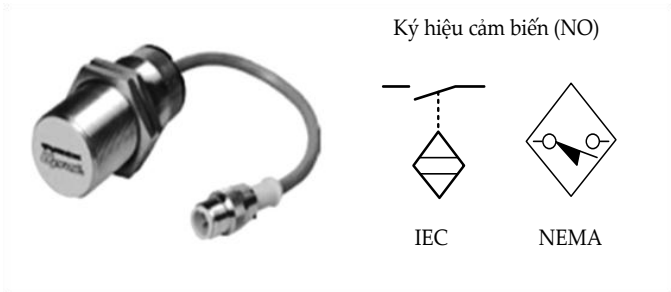
Hình 3.3 là hình ảnh của một loại cảm biến tiệm cận. Cảm biến này sẽ nhận biết sự xuất hiện của đối tượng mà không cần phải tiếp xúc. Các thiết bị bán dẫn này thường được sử dụng khi:

- Các đối tượng cần nhận biết quá nhỏ, nhẹ hoặc mềm và không thể sử dụng các chuyển mạch cơ khí.

- Yêu cầu đáp ứng và tốc độ chuyển mạch nhanh, ví dụ như các ứng dụng điều khiển đếm hoặc đẩy sản phẩm trong sản xuất công nghiệp.

- Nhận biết các đối tượng được bao bọc bởi các vật liệu như kính, nhựa và giấy các-tông.

- Đòi hỏi tuổi thọ lớn và vận hành an toàn.



Hình 3.3. Cảm biến tiệm cận

Các cảm biến tiệm cận hoạt động theo các nguyên lý khác nhau phụ thuộc vào đối tượng cần nhận biết. Khi cần nhận biết các vật liệu kim loại mà không cần tiếp xúc chúng ta có thể sử dụng cảm biến tiệm cận loại cảm kháng. Các cảm biến này được sử dụng để nhận biết cả kim loại đen (gồm cả thép) và kim loại màu (ví dụ như đồng, nhôm). Cảm biến tiệm cận loại cảm kháng hoạt động nhờ nguyên lý điện của điện cảm; nghĩa là dòng điện thăng giáng sẽ tạo ra sức điện động (emf) trong đối tượng cần đo. Sơ đồ khối của cảm biến tiệm cận loại cảm kháng được cho trên Hình 3.4 với nguyên tắc hoạt động như sau:

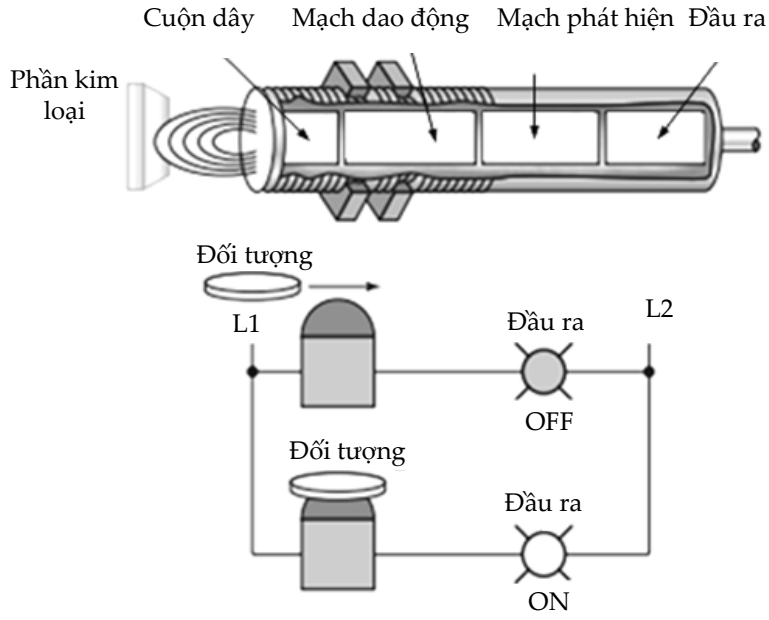
- Mạch dao động tạo ra trường điện từ với tần số cao và phát xạ từ từ đầu tới cuối của cảm biến.

- Khi đối tượng kim loại đi qua trường này, dòng điện phucô được tạo ra trên bề mặt của đối tượng.

- Dòng phucô trên đối tượng sẽ hấp thụ một phần năng lượng được phát xạ từ cảm biến và sẽ làm thay đổi cường độ dao động.

- Mạch phát hiện của cảm biến có vai trò giám sát cường độ của dao động và kích hoạt đầu ra bán dẫn tại một mức được định trước.

- Khi đối tượng ra khỏi vùng từ trường thì cường độ dao động sẽ trở lại giá trị ban đầu.

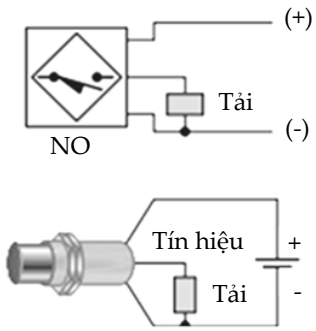


Hình 3.4. Cảm biến tiệm cận loại cảm kháng

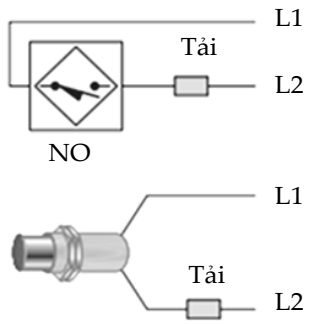
Hầu hết các ứng dụng cảm biến hoạt động tại mức điện áp hoặc là 24 VDC hoặc 120 VAC, vì vậy mà việc kết nối các cảm biến tiệm cận sẽ cần phải phù hợp với loại cảm biến và ứng dụng của nó. Hình 3.5 là một ví dụ kết nối cảm biến DC

loại 3 dây. Cảm biến tiệm cận DC loại 3 dây gồm có dây nguồn dương và nguồn âm. Khi cảm biến được kích hoạt thì dây tín hiệu sẽ được nối tới cực dương nếu hoạt động ở chế độ thường mở. Nếu hoạt động ở chế độ thường đóng thì mạch sẽ ngắt dây tín hiệu với cực dương của cảm biến.

Hình 3.6 là hình ảnh kết nối giữa cảm biến loại 2 dây nối tiếp với tải. Tại trạng thái tắt, trong mạch vẫn duy trì một dòng điện dò từ 1 tới 2 mA để duy trì hoạt động của cảm biến. Mạch điện sẽ khép kín khi cảm biến được kích hoạt.

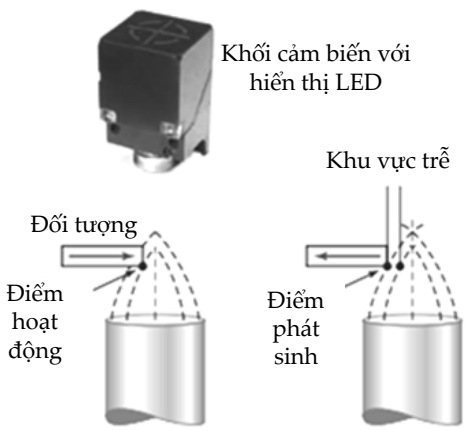


Hình 3.5. Kết nối cảm biến loại 3 dây

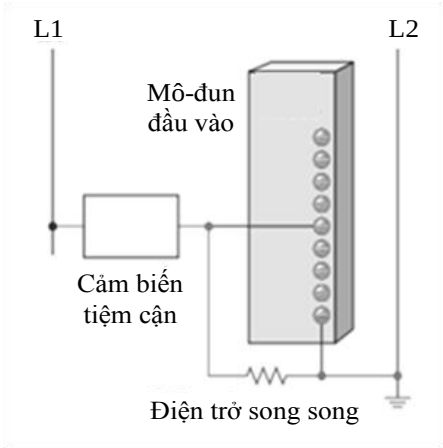


Hình 3.6. Kết nối cảm biến loại 2 dây nối tiếp với tải

Hình 3.7 là một ví dụ sử dụng cảm biến tiệm cận để nhận dạng trong một miền. Độ trễ là khoảng cách giữa điểm hoạt động khi đối tượng tiến lại gần bề mặt cảm biến và điểm dừng khi đối tượng rời xa bề mặt cảm biến. Đối tượng phải tiến lại đủ gần cảm biến để cảm biến có thể nhận biết và khi cảm biến được kích hoạt nó sẽ duy trì trạng thái cho tới khi đối tượng di chuyển tới điểm dừng. Khoảng cách cần được duy trì để các cảm biến tiệm cận có thể nhận biết đối tượng khi mà đối tượng bị dao động hoặc bị ảnh hưởng bởi các nhiễu điện từ hay sự thay đổi nhiệt độ. Hầu hết các cảm biến tiệm cận có các LED để biểu thị trạng thái đầu ra của cảm biến.



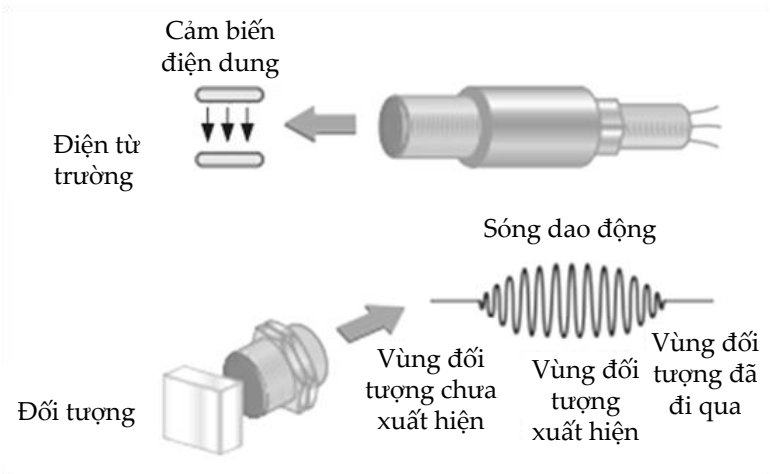
Hình 3.7. Cảm biến tiệm cận nhận dạng



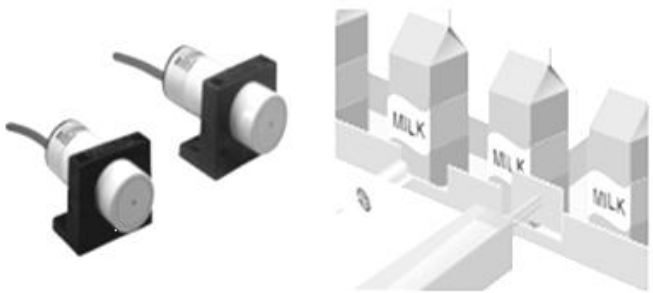
Hình 3.8. Điện trở được nối song song

Các chuyển mạch bán dẫn luôn có dòng điện rò nhỏ chạy qua cảm biến khi mà đầu ra của cảm biến được ngắt. Tương tự khi cảm biến được kích hoạt sẽ có một lượng điện áp nhỏ bị suy hao tại các cực của đầu ra. Các cảm biến tiệm cận hoạt động chính xác khi được cung cấp nguồn điện một cách liên tục. Hình 3. 8 là một ví dụ sử dụng điện trở được nối song song để có thể cung cấp đủ dòng cho cảm biến hoạt động nhưng không đủ để khởi động đầu vào của PLC.

Cảm biến tiệm cận loại điện dung tương tự như cảm biến tiệm cận loại cảm kháng. Điểm khác biệt chính giữa hai loại này đó là cảm biến tiệm cận loại điện dung tạo ra điện trường tĩnh thay vì từ trường và được kích động bởi cả các vật liệu dẫn điện và không dẫn điện. Nguyên lý hoạt động của loại cảm biến này được trình bày trên Hình 3.9. Cấu tạo của cảm biến bao gồm một bộ dao động tần số cao cùng với một bộ phận cảm nhận được tạo thành từ 2 điện cực kim loại. Khi đối tượng tiến gần tới bộ phận cảm nhận, nghĩa là nó đã đi vào miền điện trường tĩnh của cặp điện cực kim loại và làm thay đổi dung kháng của bộ dao động. Điều đó sẽ dẫn tới mạch dao động bắt đầu dao động và thay đổi trạng thái đầu ra của cảm biến khi nó đạt tới một giá trị cụ thể nào đó. Khi đối tượng di chuyển ra xa cảm biến, cường độ dao động sẽ giảm và cảm biến sẽ quay lại trạng thái ban đầu.



Hình 3.9. Cảm biến điện dung

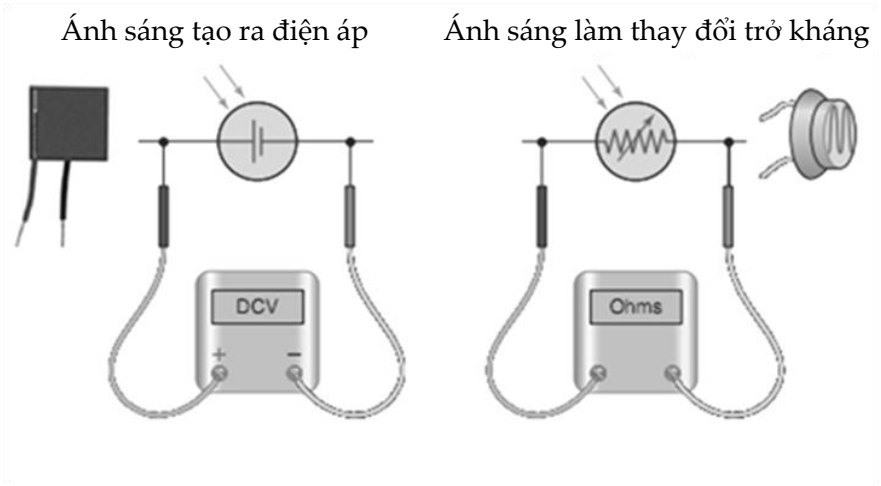


Hình 3.10. Ví dụ hoạt động cảm biến điện dung

Cảm biến tiệm cận điện dung sẽ cảm nhận được các vật bằng kim loại cũng như các vật liệu phi kim loại như giấy, thủy tinh, chất lỏng, và vải. Về cơ bản chúng có một miền nhận biết ngắn khoảng 1 inch bất chấp đối tượng là loại vật liệu nào. Các đối tượng có hằng số điện môi càng lớn thì cảm biến sẽ càng dễ phát hiện. Loại cảm biến này có khả năng phát hiện những vật liệu bên trong những bình chứa không phải kim loại như trên Hình 3.10. Trong ví dụ này, chất lỏng có hằng số điện môi cao hơn các bình chứa bằng các-tông vì vậy cảm biến có thể nhận biết được chất lỏng trong bình chứa. Trong quá trình xử lý, các bình chứa rỗng sẽ tự động được chuyển hướng thông qua cần đẩy. Cảm biến tiệm cận loại cảm kháng chỉ có thể nhận biết được các đối tượng bằng kim loại và không nhạy với môi trường có độ ẩm cũng như bụi nhiều. Ngược lại, các cảm biến tiệm cận loại điện dung có thể hoạt động trong môi trường bụi bẩn.

3.1.2.2. Cảm biến ánh sáng

Tế bào quang điện và tế bào quang dẫn là một ví dụ về cảm biến ánh sáng. Các tế bào quang điện khi tương tác với ánh sáng sẽ chuyển đổi trực tiếp năng lượng ánh sáng thành năng lượng điện còn đối với tế bào quang dẫn thì khi tiếp xúc với ánh sáng sẽ bị thay đổi điện trở.



Hình 3.11. Tế bào quang điện và tế bào quang dẫn

Cảm biến quang điện là một thiết bị điều khiển quang học hoạt động bằng cách phát hiện một chùm ánh sáng có thể nhìn thấy hoặc không nhìn thấy và đáp ứng với sự thay đổi về cường độ ánh sáng nhận được.

Các cảm biến quang học được cấu tạo bởi 2 bộ phận cơ bản: Nguồn phát và nguồn nhận như trên Hình 3.12. Hai bộ phận này có hoặc không cùng được đặt trong một khối duy nhất. Nguyên tắc hoạt động của cảm biến quang như sau:

- Bộ phát gồm một nguồn sáng (thường sử dụng LED) và một bộ dao động.

- Bộ dao động điều chế hoặc bật hoặc tắt LED với tốc độ cao.

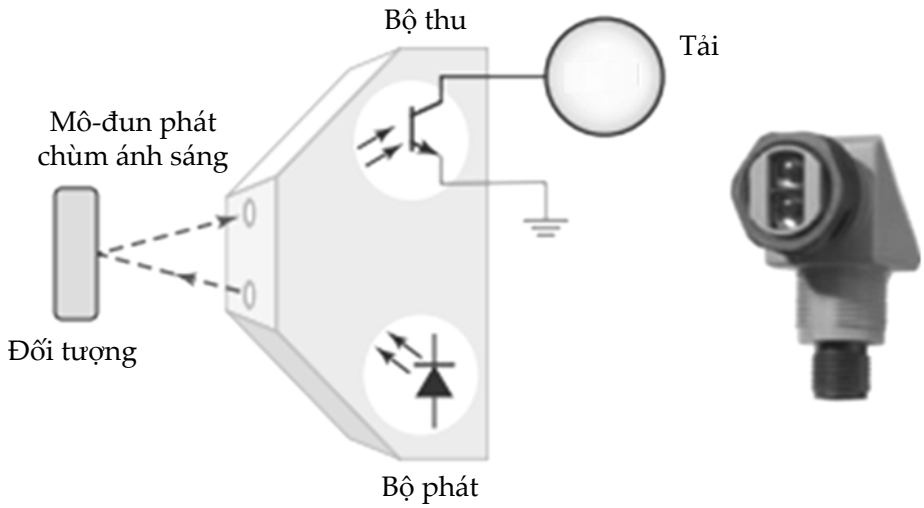
- Bộ phận gửi chùm sáng đã được điều chế này tới bộ phận nhận.

- Bộ nhận sẽ giải mã chùm ánh sáng nhận được và đóng/ngắt tải đầu ra.

- Bộ nhận sẽ điều chế tần số và chỉ khuếch đại tín hiệu ánh sáng tại một tần số cố định.

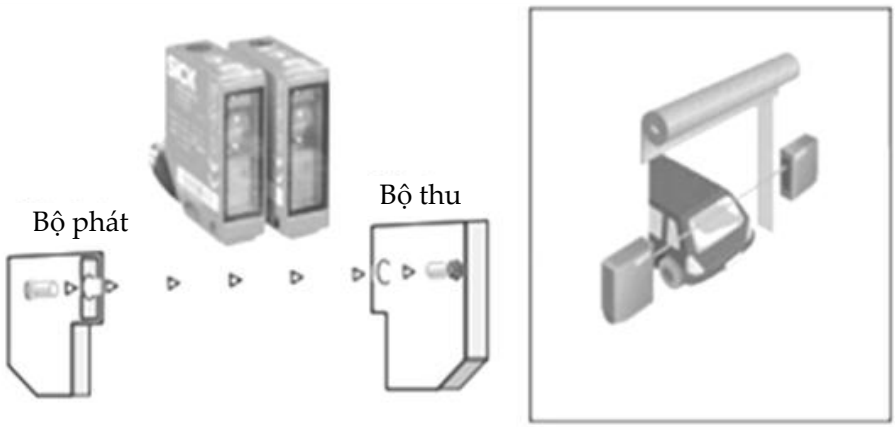
- Hầu hết cảm biến quang có thể điều chỉnh mức độ ánh sáng có thể làm thay đổi trạng thái của cảm biến.

- Thời gian đáp ứng của cảm biến liên quan tới tới tần số của các xung ánh sáng. Thời gian đáp ứng là quan trọng khi cảm biến được sử dụng trong ứng dụng nhận biết các đối tượng nhỏ, các đối tượng di chuyển với tốc độ cao hoặc cả hai.



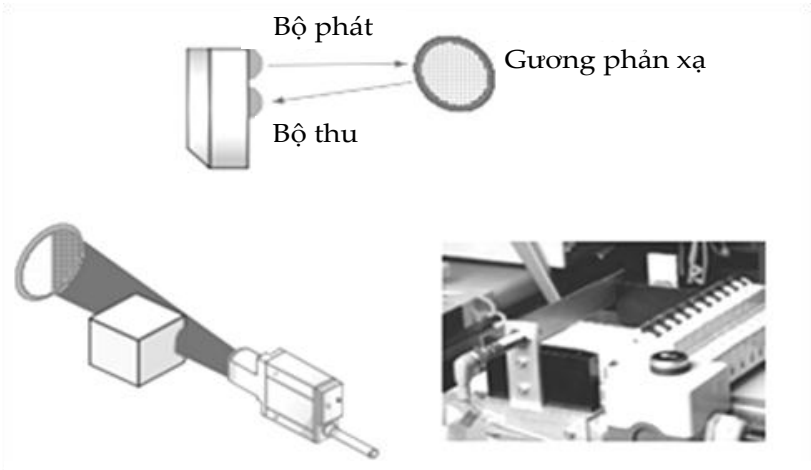
Hình 3.12. Cảm biến quang học

Kỹ thuật quét đề cập đến phương pháp được sử dụng bởi các cảm biến quang điện để phát hiện một đối tượng. Với kỹ thuật quét chùm (còn gọi là quét trực tiếp) thì bộ phận phát và bộ phận nhận được đặt đối diện với nhau như được minh họa trong Hình 3.13. Ví dụ, cơ cấu mở cửa nhà xe sẽ có một cảm biến quang học được gắn tại cửa ra vào của nhà xe và gần mặt sàn. Khi đó cảm biến sẽ nhận biết được những đối tượng đi quan cửa nhà xe.



Hình 3.13. Kỹ thuật quét chùm

Trong kỹ thuật quét phản xạ ngược trên Hình 3.14 gồm bộ phát và bộ nhận cùng được đặt trong cùng một vỏ bọc. Sự sắp xếp này yêu cầu sử dụng một bộ phản xạ riêng biệt để phản xạ ánh sáng lại bộ nhận. Kỹ thuật quét phản xạ được sử dụng cho các ứng dụng tầm trung.



Hình 3.14. Kỹ thuật quét phản xạ

3.1.2.3. Cảm biến siêu âm

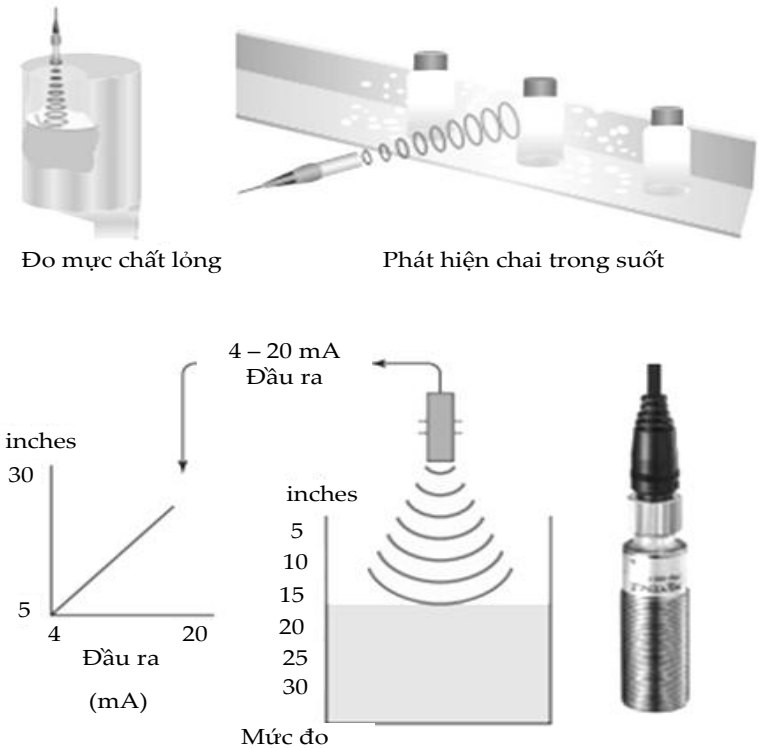
Cảm biến siêu âm hoạt động bằng cách gửi các sóng âm thanh tần số cao về phía mục tiêu và đo thời gian tín hiệu quay trở lại. Thời gian tín hiệu phản hồi tỷ lệ thuận với khoảng cách hoặc chiều cao của đối tượng bởi vì âm thanh có vận tốc không đổi. Hình 3.15 minh họa một ứng dụng thực tế trong đó các tín hiệu phản hồi được chuyển đổi thành dòng điện có giá trị từ 4 tới 20 mA. Ứng dụng này được sử dụng để giám sát lưu lượng của chất lỏng. Hoạt động của quá trình này được tóm tắt như sau:

- Dòng điện ra của cảm biến có giá trị từ 4 tới 20 mA.

- Giá trị 4 mA ứng với trường hợp khoảng cách lớn nhất từ cảm biến tới bề mặt chất lỏng.

- Giá trị 20 mA ứng với trường hợp khoảng cách nhỏ nhất từ cảm biến tới bề mặt chất lỏng.

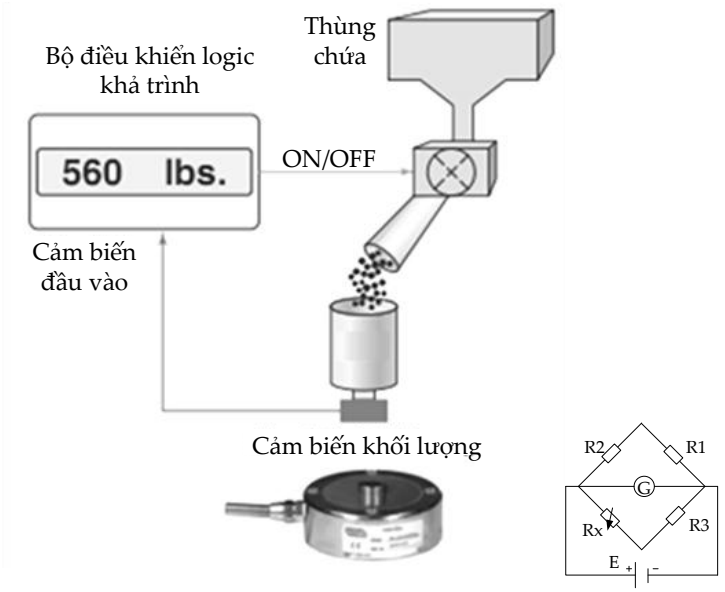
- Cảm biến siêu âm có thể phát hiện các chất rắn, chất lỏng, các đối tượng dạng hạt, và dạng sợi.



Hình 3.15. Cảm biến siêu âm

3.1.2.4. Cảm biến khối lượng

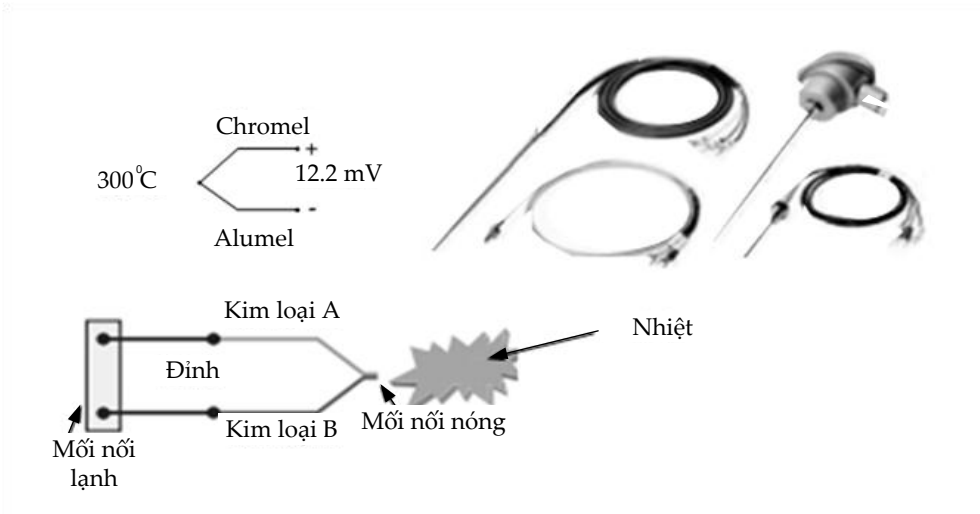
Cảm biến kiểu khối lượng kiểu cầu Wheatstone cân bằng là loại cảm biến chuyển đổi biến dạng cơ học thành tín hiệu điện. Nguyên lý hoạt động của cảm biến kiểu điện trở dựa trên nguyên tắc điện trở của dây dẫn thay đổi theo chiều dài và tiết diện ngang của dây dẫn. Khi có lực tác động lên cảm biến sẽ khiến cho cảm biến bị biến dạng. Sự biến dạng này làm thay đổi kích thước vật lý của cảm biến đồng thời làm thay đổi điện trở của nó. Loại cảm biến này thường được chế tạo từ thép và rất nhạy. Khi có tải thì bộ phận đo lực sẽ giãn ra hoặc nén lại một lượng rất nhỏ. Giá trị này sẽ được biến đổi thành tín hiệu điện áp. Hiện nay trên thị trường có nhiều loại cảm biến với nhiều kích thước, hình dạng và dải đo khác nhau. Các cảm biến này thường được dùng trong các ứng dụng cân công nghiệp như được minh họa trên Hình 3.16.



Hình 3.16. Cảm biến kiểu điện trở

3.1.2.5. Cảm biến nhiệt độ

Cặp nhiệt điện là cảm biến nhiệt độ được sử dụng rộng rãi nhất. Cặp nhiệt điện hoạt động dựa trên nguyên tắc kết hợp 2 loại vật liệu không đồng dạng với nhau, khi đó điện áp DC đầu ra tỷ lệ với sự chênh lệch nhiệt độ giữa mối nối nóng và mối nối lạnh. Mối nối nóng là điểm nối giữa 2 vật liệu và điểm nối này sẽ tiếp xúc trực tiếp với môi trường có nhiệt độ cần đo. Mối nối lạnh là mối nối giữa 2 vật liệu mà có nhiệt độ không đổi và tạo thành một điểm tham chiếu. Bởi độ nhám và dải đo của cặp nhiệt điện là rộng vì vậy mà cặp nhiệt điện thường được sử dụng trong công nghiệp để giám sát và điều khiển nhiệt độ của buồng sấy cũng như buồng đốt.



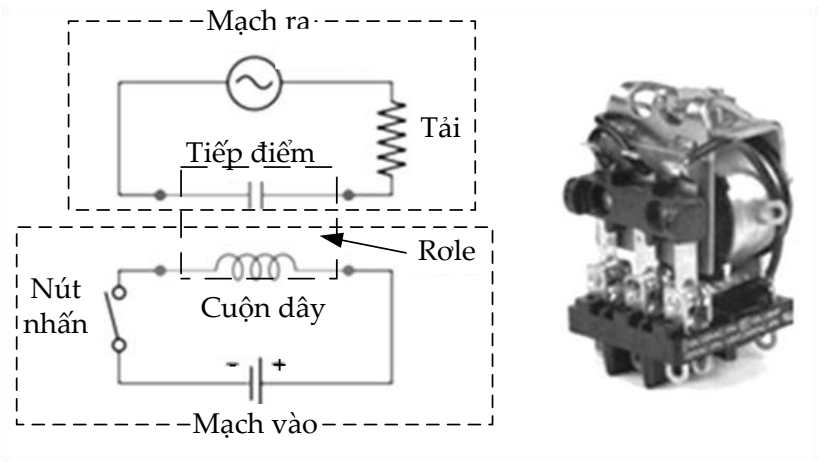
Hình 3.17. Cặp nhiệt điện

3.2. THIẾT BỊ ĐẦU RA

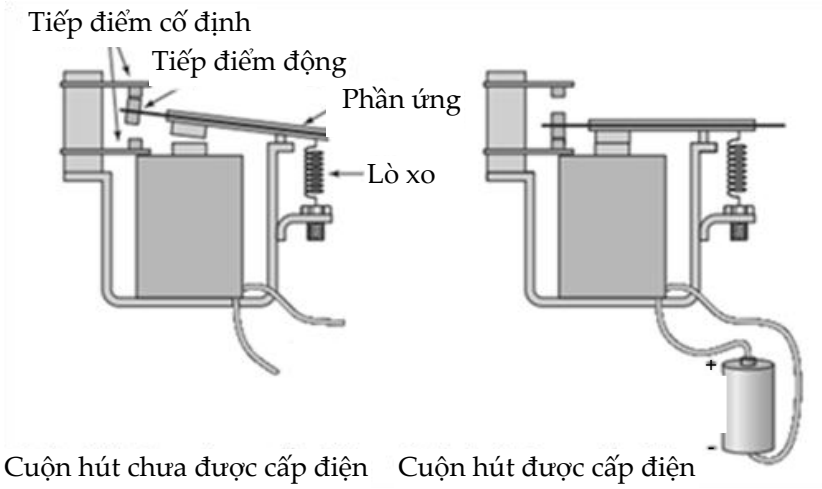
3.2.1. Rơle điện từ

PLC ra đời với mục đích ban đầu là để thay thế các rơle điện từ bằng một hệ thống chuyển mạch bán dẫn có thể lập trình được. Mặc dù, PLC đã thay thế nhiều loại rơle điều khiển logic nhưng rơle điện từ vẫn được sử dụng và có vai trò như các thiết bị phụ trợ để chuyển mạch các thiết bị vào/ra tại hiện trường. Các bộ điều khiển logic khả trình thiết kế để thay thế các rơle điều khiển, cụ thể để đưa ra các quyết định logic; tuy nhiên các rơle này lại không được thiết kế để chịu được dòng điện hoặc điện áp quá cao. Sự hiểu biết về hoạt động của các rơle điện từ và các thuật ngữ rất quan trọng để có thể chuyển đổi chính xác các sơ đồ nguyên lý mạch rơle sang các chương trình logic bậc thang. Rơle điện từ chính là một chuyển mạch từ và nó sử dụng điện từ để chuyển mạch các tiếp điểm.

Hình 3.18 giải thích hoạt động của một rơle điện từ cơ bản. Khi chưa có dòng điện chạy qua cuộn hút, phần ứng tách biệt so với lõi của phần cảm bởi sức căng lò xo. Khi phần cảm được cấp điện, nó sẽ tạo ra một từ trường trong các vòng dây và gây ra quá trình di chuyển vật lý của phần ứng. Chính sự di chuyển này sẽ làm cho các điểm tiếp xúc của rơle có thể mở hoặc đóng.



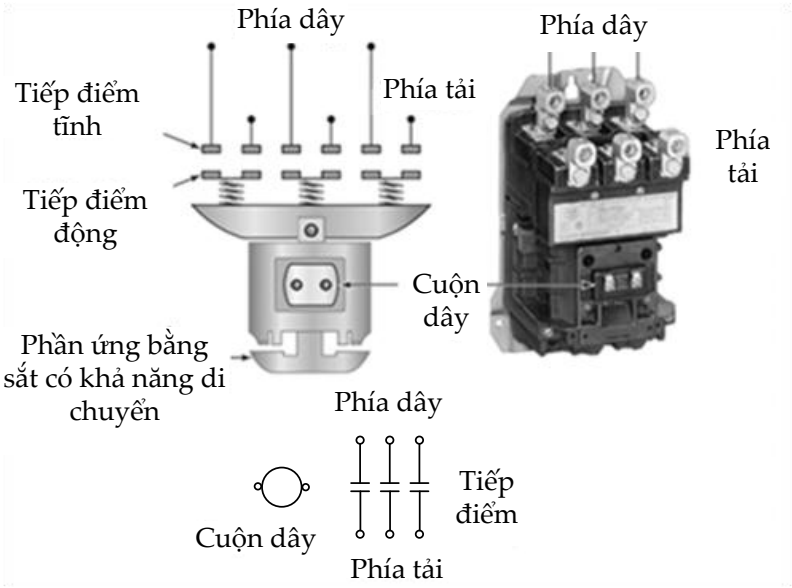
Hình 3.18. Rơle điều khiển điện từ



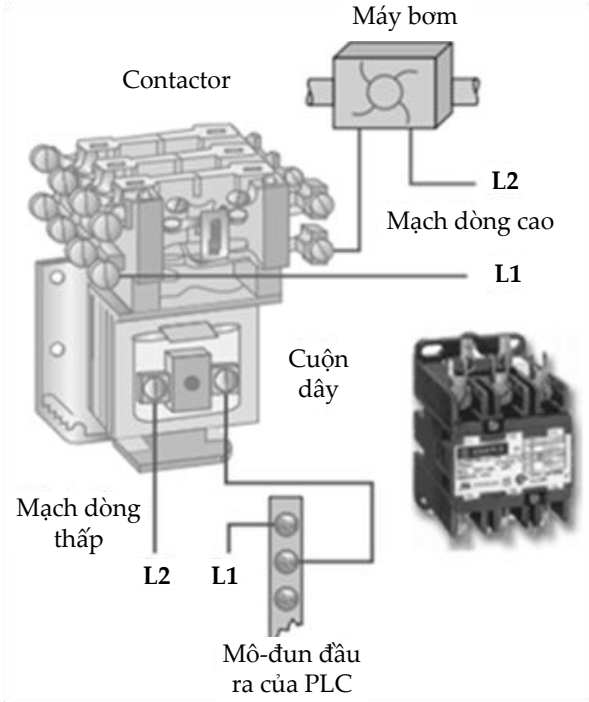
Hình 3.19. Nguyên lý hoạt động của Rơle

3.2.2. Contactor

Contactor là loại rơle đặc biệt được thiết kế để đóng cắt các loại tải có công suất lớn vượt quá khả năng chịu đựng của các rơle điện từ. Các tải này bao gồm đèn chiếu sáng, máy sưởi, máy biến thế và động cơ điện có bảo vệ quá tải được cung cấp riêng hoặc không cần thiết. Các bộ điều khiển logic khả trình thường có công suất đầu ra đủ để điều khiển cuộn hút của Contactor mà không cần phải tiếp xúc trực tiếp với tải có công suất lớn. Hình 3.21 là một sơ đồ ứng dụng PLC kết hợp với Contactor để đóng/ngắt máy bơm. Mô-đun đầu ra được nối tiếp với cuộn hút để tạo thành một mạch đóng/cắt dòng thấp. Các tiếp điểm của Contactor được nối tiếp với động cơ của máy bơm tạo thành mạch đóng/cắt có dòng lớn.



Hình 3.20. Contactor điện từ 3 cực



Hình 3.21. PLC kết hợp với Contactor

3.2.3. Bộ khởi động từ

Bộ khởi động động cơ được thiết kế để cung cấp điện cho quá trình khởi động động cơ. Bộ khởi động động cơ được tạo thành từ một Contactor kết với một rơle chống quá tải. Các chức năng chính của rơle chống quá tải có thể được tóm tắt như sau:

- Được thiết kế để bảo vệ cho các mạch điều khiển động cơ.
- Sẽ ngắt điện năng ra khỏi động cơ nếu xảy ra hiện tượng quá tải.
- Sau khi khắc phục hiện tượng quá tải ta có thể khởi động lại các rơle.

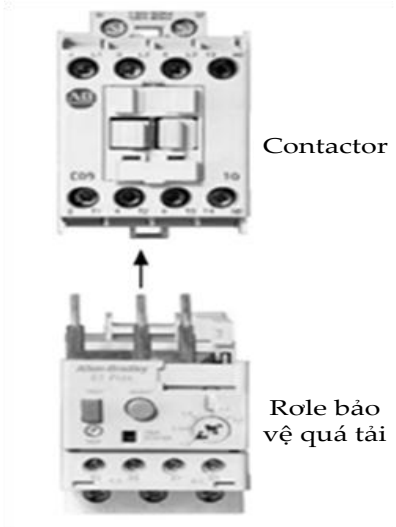
Hình 3.22 và Hình 3.23 là sơ đồ của bộ khởi động động cơ từ 3 pha thường thấy. Nguyên lý hoạt động của mạch như sau:

- Khi nút START được nhấn, cuộn hút M được cấp năng lượng và tạo thành mạch chốt, tất cả các tiếp điểm thường mở M tiếp điện.

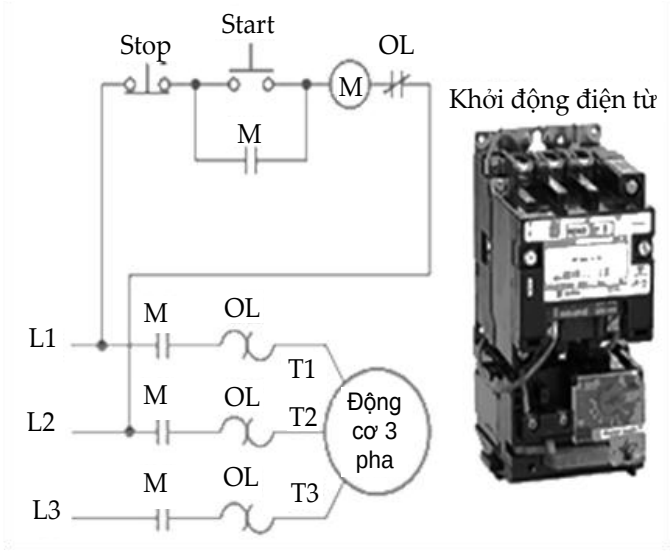
- Các tiếp điểm M nối tiếp với động cơ sẽ tạo ra một dòng điện khép kín trong động cơ. Các tiếp điểm này là một phần của mạch công suất và phải được thiết kế để xử lý hiện tượng quá tải của động cơ.

- Rơle bảo vệ quá tải (OL) được sử dụng để bảo vệ động cơ khỏi dòng điện ngược khi xảy ra hiện tượng quá tải. Rơle tiếp điểm thường đóng OL sẽ tự động

mở khi xảy ra hiện tượng quá tải (dòng quá tải) và đồng thời cuộn hút M sẽ được ngưng cấp điện (động cơ ngừng hoạt động).

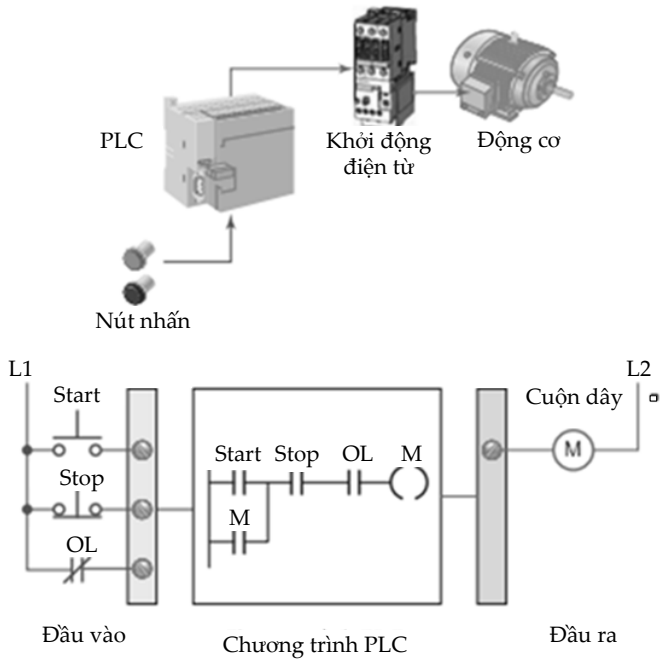


Hình 3.22. Bộ khởi động được kết hợp từ Contactor và rơle chống quá tải



Hình 3.23. Bộ khởi động động cơ từ 3 pha

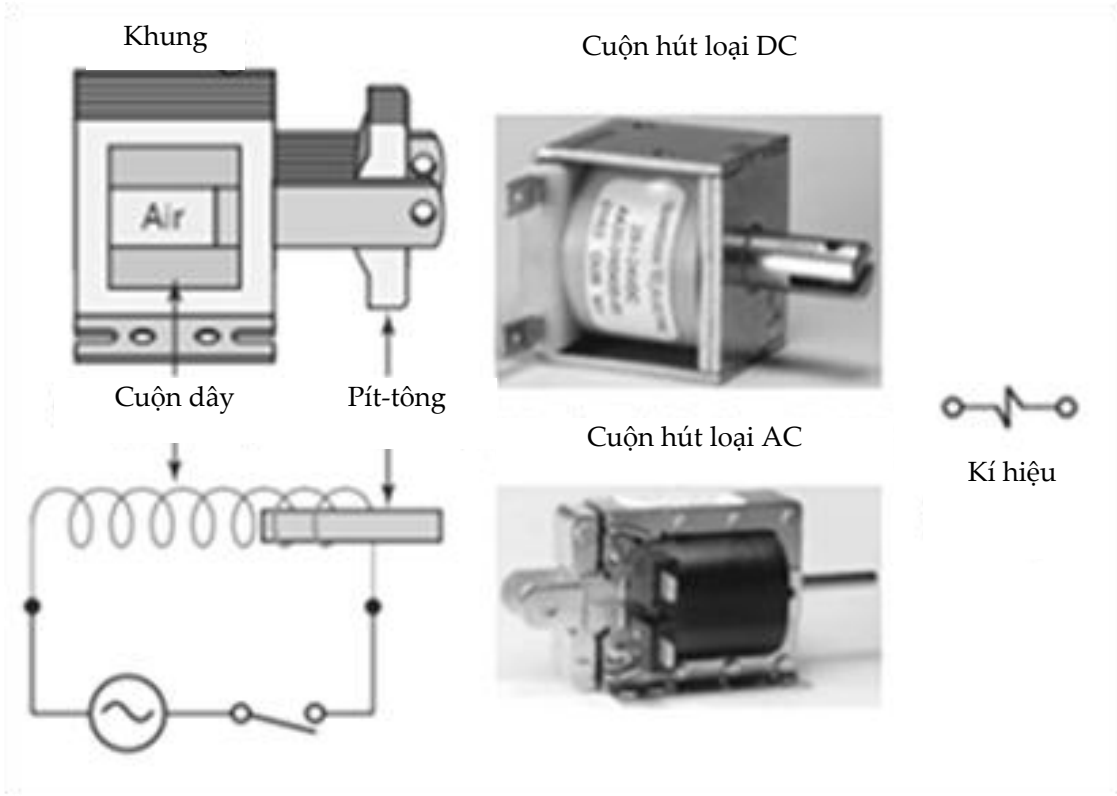
Khi sử dụng PLC để điều khiển động cơ có công suất lớn, chúng ta cần kết hợp PLC với một bộ khởi động động cơ như Hình 3.24. Công suất điều khiển cuộn hút của bộ khởi động động cơ phải nằm trong dải công suất mô-đun đầu ra của PLC.



Hình 3.24. Điều khiển động cơ bằng PLC

3.2.4. Van điện từ

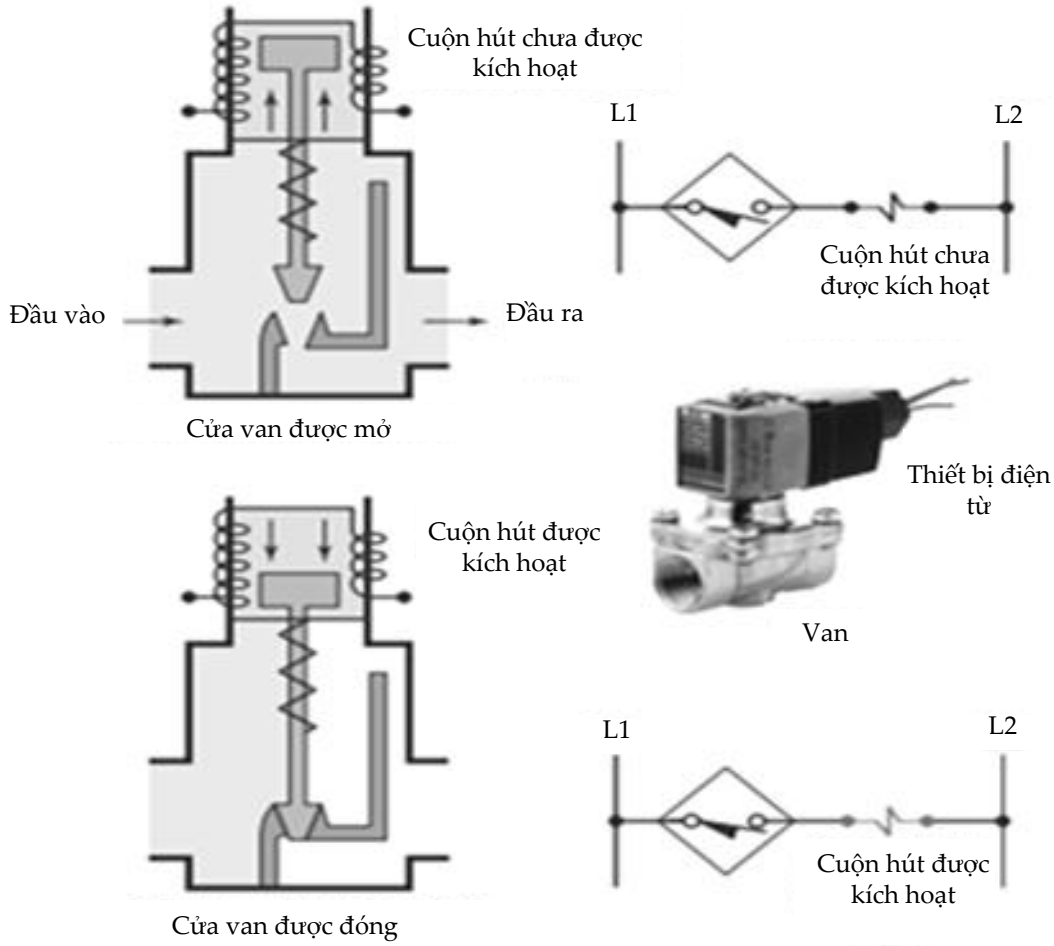
Van điện từ là loại thiết bị điện cơ hoạt động bằng cách cho dòng điện chạy qua cuộn hút làm thay đổi trạng thái của van. Cấu tạo của van thường có một bộ phận cơ khí (có thể là lò xo) để giữ cho van duy trì ở vị trí mặc định của nó. Van điện từ được sử dụng để điều khiển dòng chảy của chất lỏng, khí, hơi và các môi trường khác. Khi được cấp điện chúng sẽ mở, ngắt hoặc chuyển hướng dòng chảy của môi trường.



Hình 3.25. Cấu tạo và nguyên lý hoạt động của cuộn hút điện từ

Hình 3.26 mô tả cấu tạo và nguyên lý hoạt động của van điện từ được sử dụng trong môi trường chất lỏng:

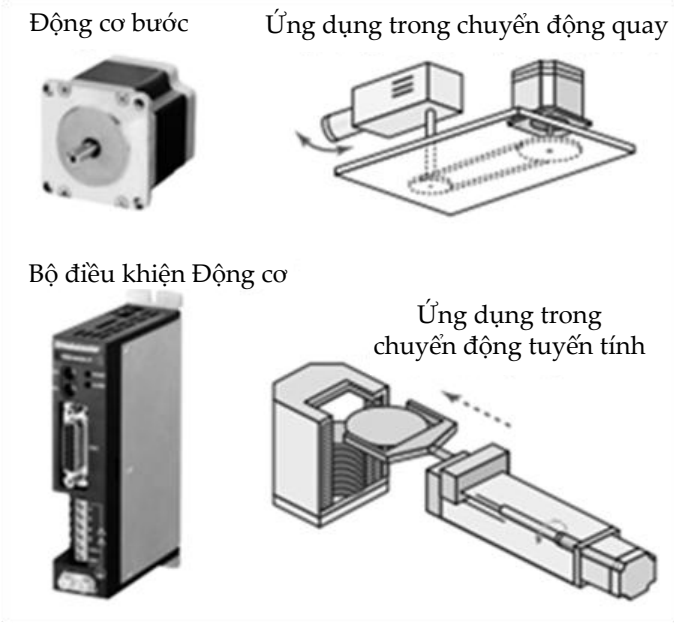
- Thân van có một bộ phận được định vị để cho phép hoặc cấm dòng chảy đi qua.
- Dòng chảy được phép hoặc bị cấm phụ thuộc vào trạng thái của lõi cuộn hút được cấp điện hay không.
- Khi cuộn hút được cấp điện thì van sẽ mở.
- Lực lò xo sẽ giúp cho van trở lại vị trị ban đầu khi lõi hút bị ngắt điện.
- Van phải được lắp đặt phù hợp với hướng của dòng chảy.



Hình 3.26. Cấu tạo và nguyên lý hoạt động của van điện từ

3.2.5. Động cơ bước

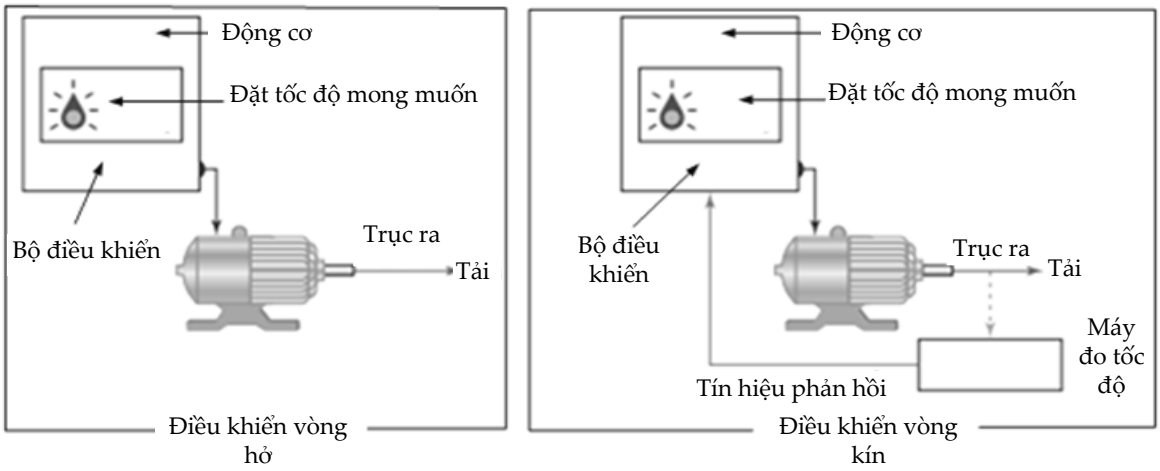
Nguyên lý hoạt động của động cơ bước khác so với các loại động cơ khác. Trục của động cơ bước quay một cách rời rạc khi được cấp các chuỗi xung điều khiển được một cách hợp lý. Một vòng quay bao gồm nhiều bước và mỗi bước quay của trục động cơ tương ứng với một chuỗi xung điện áp phù hợp. Số vòng quay tỷ lệ trực tiếp với số xung và tốc độ quay liên quan tới tần số của xung điều khiển. Động cơ bước với độ phân giải 1^0 trên một bước sẽ mất 360 bước để trục động cơ quay được 1 vòng. Các động cơ bước thường được sử dụng trong các hệ thống điều khiển vòng hở (open-loop) và điều khiển vị trí với công suất nhỏ. Sự chuyển động được tạo ra bởi mỗi xung là chính xác và có thể lặp lại, đó là lý do tại sao động cơ bước rất hiệu quả cho các ứng dụng định vị tải. Hình 3.27 là hình ảnh của động cơ bước, bộ điều khiển và một vài ứng dụng của nó.



Hình 3.27. Động cơ bước và bộ điều khiển

3.2.6. Động cơ servo

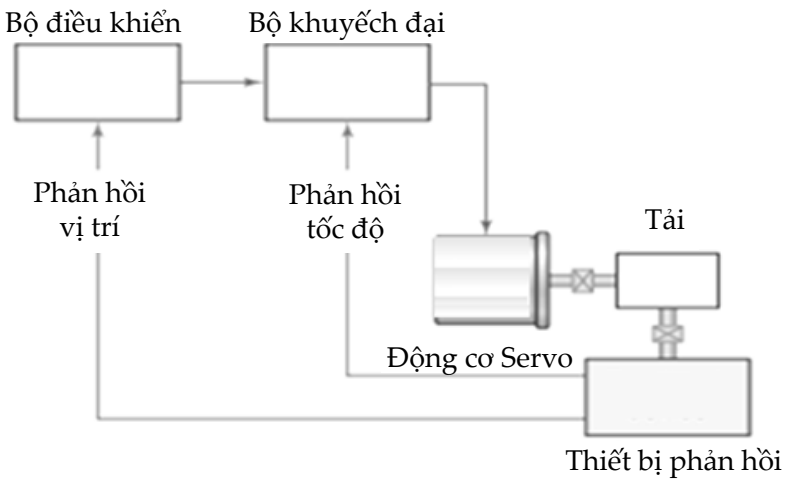
Tất cả các động cơ servo hoạt động ở chế độ vòng kín, trong khi hầu hết các động cơ bước hoạt động ở chế độ vòng hở. Nguyên lý hoạt động vòng hở và vòng kín được mô tả như trên Hình 3.28. Đối với chế độ vòng hở sẽ không có tín hiệu phản hồi. Bộ điều khiển chỉ điều khiển trục động cơ quay bao nhiêu bước và nhanh hay chậm nhưng nó sẽ không biết được vị trí của trục động cơ. Với chế hoạt động vòng kín thì vị trí hay tốc độ hiện tại của động cơ được phản hồi và so sánh với giá trị mong muốn để từ đó tính được sai số và đưa ra tín hiệu điều khiển để có được sai số nhỏ nhất.



Hình 3.28. Hệ thống điều khiển động cơ vòng hở và vòng kín

Hình 3.29 mô tả một hệ thống điều khiển vòng kín sử dụng động cơ servo. Bộ điều khiển trực tiếp điều khiển động cơ bằng cách gửi các tín hiệu điều khiển vị trí hay tốc độ tới bộ khuếch đại để điều khiển động cơ servo. Thiết bị phản hồi (bộ mã hóa quang học đối với điều khiển vị trí hay tốc độ kế đối với điều khiển tốc độ) có thể được gắn cùng với động cơ hoặc được định vị riêng biệt (thường được gắn cùng với động cơ). Thiết bị này sẽ cung cấp cho bộ điều khiển các giá trị tốc độ hay vị trí hiện tại của động cơ để từ đó bộ điều khiển tính toán sai số và đưa ra tín hiệu điều khiển.

Động cơ/bộ điều khiển



Hình 3.29. Hệ thống điều khiển vòng kín động cơ servo

CÂU HỎI ÔN TẬP CHƯƠNG 3

Chú ý: T ký hiệu cho TRUE và F ký hiệu cho FALSE

1. Nêu ứng dụng cho mỗi dạng mô-đun vào/ra đặc biệt sau:
    a) Mô-đun bộ đếm tốc độ cao
    b) Mô-đun điều chỉnh vặn tay
    c) Mô-đun TTL
    d) Mô-đun bộ đếm mã hoá quang học
    e) Mô-đun động cơ bước
    f) Mô-đun đầu ra BCD
2. Nêu ứng dụng cho mỗi dạng mô-đun vào/ra thông minh sau:
    a) Mô-đun PID
    b) Mô-đun điều khiển tốc độ và vị trí.
    c) Mô-đun truyền thông
3. Giải thích ngắn gọn cho mỗi khái niệm sau:
    a) Điện áp đầu vào danh định.
    b) Điện áp ngưỡng đầu vào.
    c) Dòng điện danh định đối với một đầu vào.
    d) Dải nhiệt độ môi trường.
    e) Trễ đầu ON/OFF đầu vào.
    f) Điện áp đầu ra.
    g) Dòng điện đầu ra.
    h) Dòng khởi động.
    i) Bảo vệ ngắn mạch.
    j) Dòng điện dò.
    k) Cách ly điện.
4. Giải thích ngắn gọn đặc điểm kỹ thuật cho mỗi dạng mô-đun vào/ra sau:
    a) Số kênh trên mỗi mô-đun.
    b) Dải áp/dòng đầu vào.

c) Dải áp/dòng đầu ra.

d) Bảo vệ đầu vào.

e) Độ phân giải.

f) Trở kháng và dung kháng đầu vào.

5. Giải thích ngắn gọn các thuật ngữ dưới đây áp dụng cho bộ nhớ của PLC:

a) Quá trình ghi.

b) Quá trình đọc.

c) Bits, bytes, word.

d) Địa chỉ trong bộ nhớ.

Chương 4
LẬP TRÌNH PLC THEO NGÔN NGỮ BẬC THANG

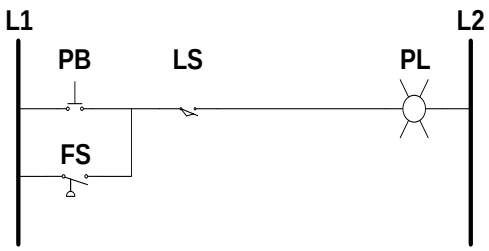
4.1. GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH BẬC THANG

Khi mà nền công nghiệp sản xuất PLC ngày càng phát triển và mở rộng hơn thì ngôn ngữ lập trình cũng phải được phát triển để tương thích với phần cứng. Các câu lệnh mới cung cấp một khả năng tính toán cao hơn. Vì vậy, PLC có thể truyền dữ liệu từ vị trí bộ nhớ này tới vị trí bộ nhớ khác trong khi vẫn có thể thực hiện các phép tính số học.

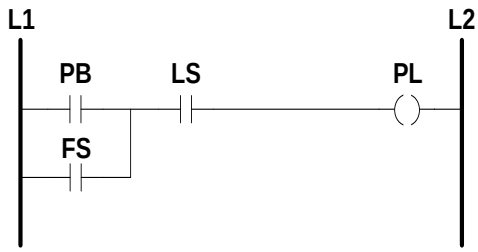
Ngoài ra sự phát triển mạnh mẽ các mô-đun vào/ra cũng đã làm thay đổi các câu lệnh lập trình hiện hành. Những thay đổi này bao gồm khả năng gửi và nhận dữ liệu từ các mô-đun. Ví dụ, PLC có khả năng đọc và ghi dữ liệu từ các mô-đun tương tự. Ngôn ngữ lập trình cho PLC cho phép lập trình dễ dàng hơn, nhỏ gọn hơn và chương trình hướng đối tượng hơn.

4.1.1. Ngôn ngữ lập trình bậc thang

Khi mới ra đời, PLC được sử dụng chủ yếu để thay thế các sơ đồ mạch điện phức tạp gồm các rơle, tiếp điểm, bộ định thời, mạch chốt và các phần tử điện trung gian khác làm nhiệm vụ của các mạch logic. Tuy nhiên khi dùng PLC, các phần tử logic trung gian này được thay thế hoàn toàn bằng các sơ đồ điện "ảo" do người thiết kế lập trình. Việc mô phỏng các sơ đồ mạch điện này được viết bằng một dạng ngôn ngữ điều khiển gọi là ngôn ngữ bậc thang (Ladder Language). Hình 4.1 và Hình 4.2 mô tả sự khác nhau giữa 2 loại sơ đồ của cùng một mạch điều khiển.



Hình 4.1. Sơ đồ đấu nối phần cứng



Hình 4.2. Sơ đồ bậc thang

Nhờ sự phát triển không ngừng mà ngôn ngữ lập trình bậc thang đã trở thành ngôn ngữ lập trình có tính ứng dụng cao. Các câu lệnh mới luôn được bổ sung để nâng cao các tính năng cơ bản của bộ lập trình. PLC thường được lập trình bằng một ngôn ngữ mô phỏng giống như sơ đồ điện gọi là sơ đồ bậc thang (Ladder Diagram). Mỗi phần tử của sơ đồ là một lệnh (Instruction). Các lệnh phức tạp thường có một mã lệnh (Code) riêng, và đều dựa trên nguyên tắc cơ bản giống như rơle, nhưng cho phép thực hiện các hoạt động phức tạp hơn rơle.

Ngôn ngữ bậc thang có sẵn trong PLC được chia thành 2 nhóm:

- Ngôn ngữ bậc thang cơ bản.
- Ngôn ngữ bậc thang nâng cao.

Trên thực tế, việc phân loại ngôn ngữ bậc thang dựa trên rất nhiều yếu tố. Ví dụ cách phân loại đơn giản nhất đó là: Ngôn ngữ mà sử dụng câu lệnh được gọi là ngôn ngữ bậc thang cơ bản, còn sử dụng khối hàm nâng cao được gọi là ngôn ngữ nâng cao. Ngoài ra, các câu lệnh trong ngôn ngữ lập trình bậc thang được nhóm thành các nhóm sau:

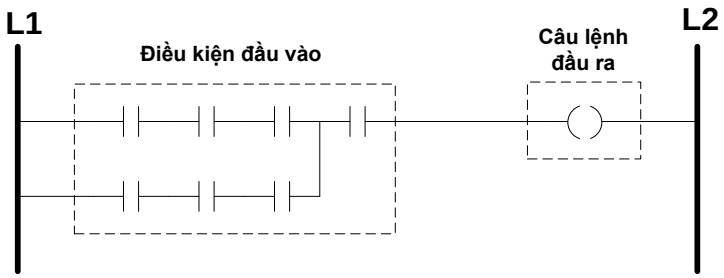
- Nhóm lệnh tiếp điểm đầu vào và cuộn hút đầu ra (lệnh rơle cơ bản).
- Nhóm lệnh định thời.
- Nhóm lệnh bộ đếm.
- Nhóm lệnh điều khiển chương trình.
- Nhóm lệnh số học.
- Nhóm lệnh xử lý dữ liệu.
- Nhóm lệnh sao chép dữ liệu.
- Nhóm lệnh đặc biệt.
- Nhóm lệnh giao tiếp truyền thông.

4.1.2. Định dạng sơ đồ bậc thang

Ngôn ngữ lập trình bậc thang là một tập các lệnh dạng ký hiệu được sử dụng để tạo ra các chương trình điều khiển PLC. Các ký hiệu câu lệnh hình thang có thể được sắp xếp để có thể điều khiển các mức logic mong muốn, sau đó sẽ được nạp vào bộ nhớ PLC.

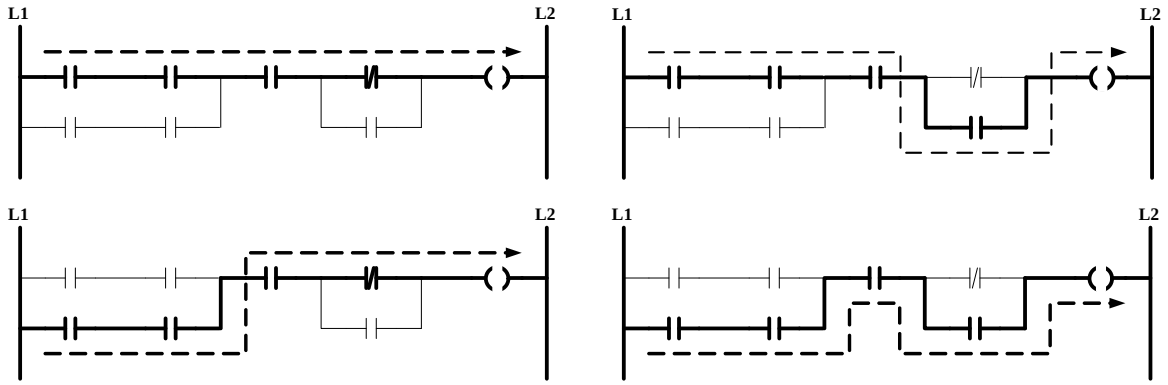
Khi có một sự hiểu biết sâu sắc về lập trình PLC bằng ngôn ngữ bậc thang sẽ rất có lợi bởi sơ đồ bậc thang rất dễ sử dụng và thực hiện. Ngôn ngữ lập trình bậc thang là một công cụ lập trình mạnh tuân theo chuẩn IEC 1131.

Chức năng chính của chương trình là điều khiển trạng thái của các cuộn hút đầu ra dựa trên các trạng thái của các tiếp điểm đầu vào và các yêu cầu hoạt động. Hình 4.3 mô tả cấu trúc cơ bản của một chương trình bậc thang. Mỗi bậc thang bao gồm một tập hợp các điều kiện đầu vào (biểu diễn bằng các câu lệnh tiếp điểm) và một câu lệnh đầu ra được bố trí ở cuối của bậc thang (biểu diễn bởi ký hiệu cuộn dây). Các câu lệnh tiếp điểm tại mỗi bậc thang có thể được coi là một điều kiện đầu vào hay logic điều khiển.



Hình 4.3. Cấu trúc một bậc thang

Đầu của một bậc thang có trạng thái logic là TRUE khi mà tính logic của nó là liên tục. Trạng thái logic của bậc thang là liên tục khi nguồn điện là liên tục từ trái qua phải. Trên một bậc, đường dây điện bên trái được ký hiệu là L1 và đường dây điện bên phải được ký hiệu là L2. Sự liên tục xảy ra khi tồn tại một đường dẫn liên tục giữa hai đường L1 và L2, cho phép dòng điện chạy từ trái sang phải như mô tả trên Hình 4.4.



Hình 4.4. Đường dẫn liên tục

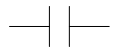
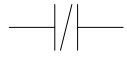
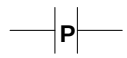
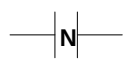
Khi chương trình có sử dụng các khối hàm thì các câu lệnh tiếp điểm cũng sẽ được sử dụng để biểu thị các điều kiện đầu vào để điều khiển (cho phép) hoạt động của khối hàm. Mỗi một khối hàm có thể có một hoặc nhiều đầu vào cho phép hoạt động đồng thời cũng có thể có một hoặc nhiều đầu ra.

4.2. CÁC LỆNH TIẾP ĐIỂM ĐẦU VÀO VÀ CUỘN HÚT ĐẦU RA

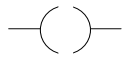
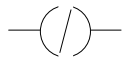
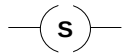
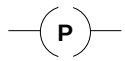
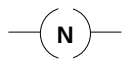
Tập lệnh rơle là tập các lệnh cơ bản nhất trong kỹ thuật lập trình PLC sử dụng ngôn ngữ bậc thang. Các lệnh này biểu diễn trạng thái ON/OFF của các tiếp điểm đầu vào cũng như các cuộn hút đầu ra. Các tiếp điểm đầu vào xác định các điều kiện đầu vào nhằm mục đích điều khiển; các cuộn hút đầu ra mô tả trạng thái đầu ra được điều khiển trên một bậc thang. Trong giáo trình này sẽ đề cập đến các lệnh lập trình PLC theo ngôn ngữ bậc thang với bộ điều khiển PLC dùng trong công tác giảng dạy PLC ED-4260.

Trong chương trình, mỗi một tiếp điểm và cuộn hút có một địa chỉ tham chiếu nhất định. Trên một bậc thang, các tiếp điểm có thể được nối song song, nối tiếp hoặc kết hợp song song/nối tiếp. Trạng thái cuộn hút đầu ra trên một bậc phụ thuộc vào trạng thái logic của các tiếp điểm đầu vào. Tại mỗi bậc thang, xét từ trái qua phải nếu tồn tại ít nhất một đường liên tục các giá trị logic TRUE của các tiếp điểm thì cuộn hút đầu ra sẽ được điều khiển. Ngược lại trạng thái của cuộn hút đầu ra sẽ không được điều khiển.

Bảng 4.1. Các loại tiếp điểm đầu vào

Tên tiếp điểm	Ký hiệu	Nguyên lý hoạt động
Tiếp điểm thường mở		Khi bit được gán cho tiếp điểm có trạng thái ON thì tiếp điểm sẽ tiếp điện hay mạch bên trái và bên phải tiếp điểm sẽ khép kín và ngược lại sẽ hở mạch.
Tiếp điểm thường đóng		Khi bit được gán cho tiếp điểm có trạng thái ON thì tiếp điểm sẽ tiếp điện hay mạch bên trái và bên phải tiếp điểm sẽ hở mạch và ngược lại sẽ khép kín.
Tiếp điểm phát hiện xung sườn trước		Khi tín hiệu đầu vào bên trái tiếp điểm chuyển từ trạng thái OFF sang ON thì đầu ra của tiếp điểm sẽ được ON trong một chu kỳ quét.
Tiếp điểm phát hiện xung sườn sau		Khi tín hiệu đầu vào bên trái tiếp điểm chuyển từ trạng thái ON sang OFF thì đầu ra của tiếp điểm sẽ được ON trong một chu kỳ quét.

Bảng 4.2. Các loại cuộn hút đầu ra

Loại cuộn hút đầu ra	Ký hiệu	Nguyên lý hoạt động
Cuộn hút thường mở		Khi các điều kiện đầu vào là TRUE thì trạng thái của bit gán cho cuộn hút thường mở cũng là TRUE và ngược lại.
Cuộn hút thường đóng		Khi các điều kiện đầu vào là TRUE thì trạng thái của bit gán cho cuộn hút thường đóng là FALSE và ngược lại.
Lệnh SET		Khi các điều kiện đầu vào là TRUE thì trạng thái của bit gán cho cuộn hút cũng là TRUE và sẽ chỉ trở về trạng thái FALSE khi được khởi động lại bởi lệnh RESET.
Lệnh RESET		Câu lệnh này được sử dụng để khởi động lại trạng thái của bit được thiết lập bởi câu lệnh SET. Câu lệnh chỉ hoạt động khi các điều kiện đầu vào là TRUE.
Cuộn hút hút đầu ra phát hiện xung sườn trước		Trạng thái của bit gán với câu lệnh chỉ ON trong một chu kỳ quét khi tín hiệu bên trái câu lệnh chuyển trạng thái từ FALSE sang TRUE.
Cuộn hút đầu ra phát hiện xung sườn sau		Trạng thái của bit gán với câu lệnh chỉ ON trong một chu kỳ quét khi tín hiệu bên trái câu lệnh chuyển trạng thái từ TRUE sang FALSE.

4.2.1. Đầu vào/ra cơ bản

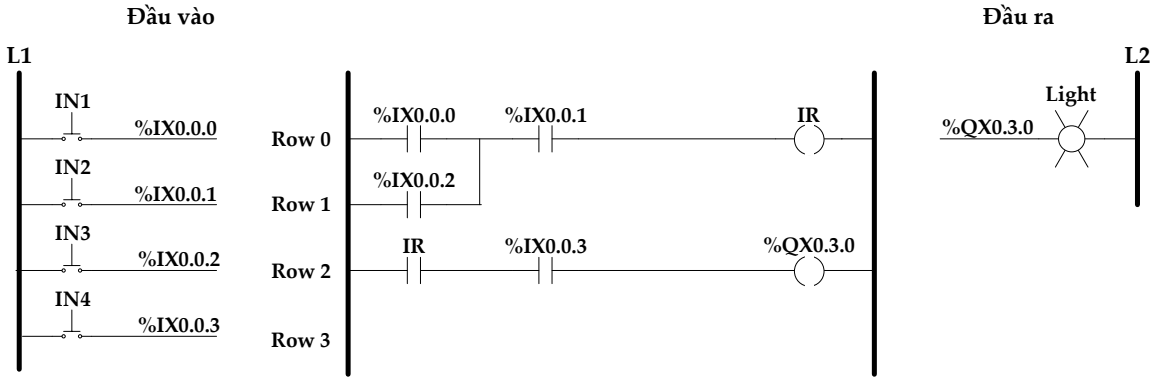
Để hiểu nguyên lý hoạt động của các loại tiếp điểm đầu vào và cuộn hút đầu ra, chúng ta hãy xét một số ví dụ sau:

Ví dụ 1:

- Tại bậc 0 và 1, khi cặp nút nhấn IN1/IN2 hoặc IN3/IN2 tiếp điện thì rơle nội IR được kích hoạt, đồng thời tiếp điểm thường mở IR tại bậc 2 cũng tiếp điện.

- Trạng thái đèn (Light) tại đầu ra phụ thuộc vào trạng thái của nút nhấn IN4. Nếu nút nhấn IN4 tiếp điện thì đèn sáng và ngược lại.

Chương trình và sơ đồ kết nối đầu vào/ra:



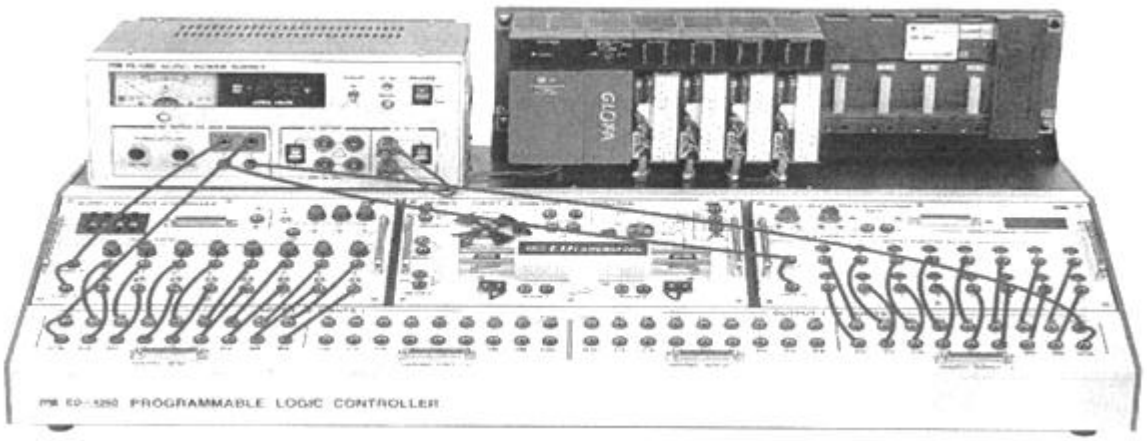
Hình 4.5. Chương trình bật-tắt đèn đơn giản

Ví dụ 2:

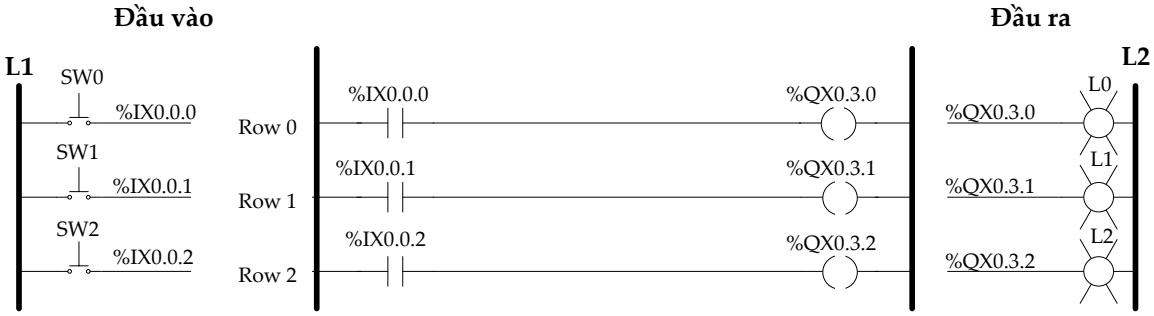
- Chương trình gồm 16 đầu vào tương ứng với 16 đầu ra. Các đầu vào được nối với các công tắc, các đầu ra được nối với các đèn chỉ thị.

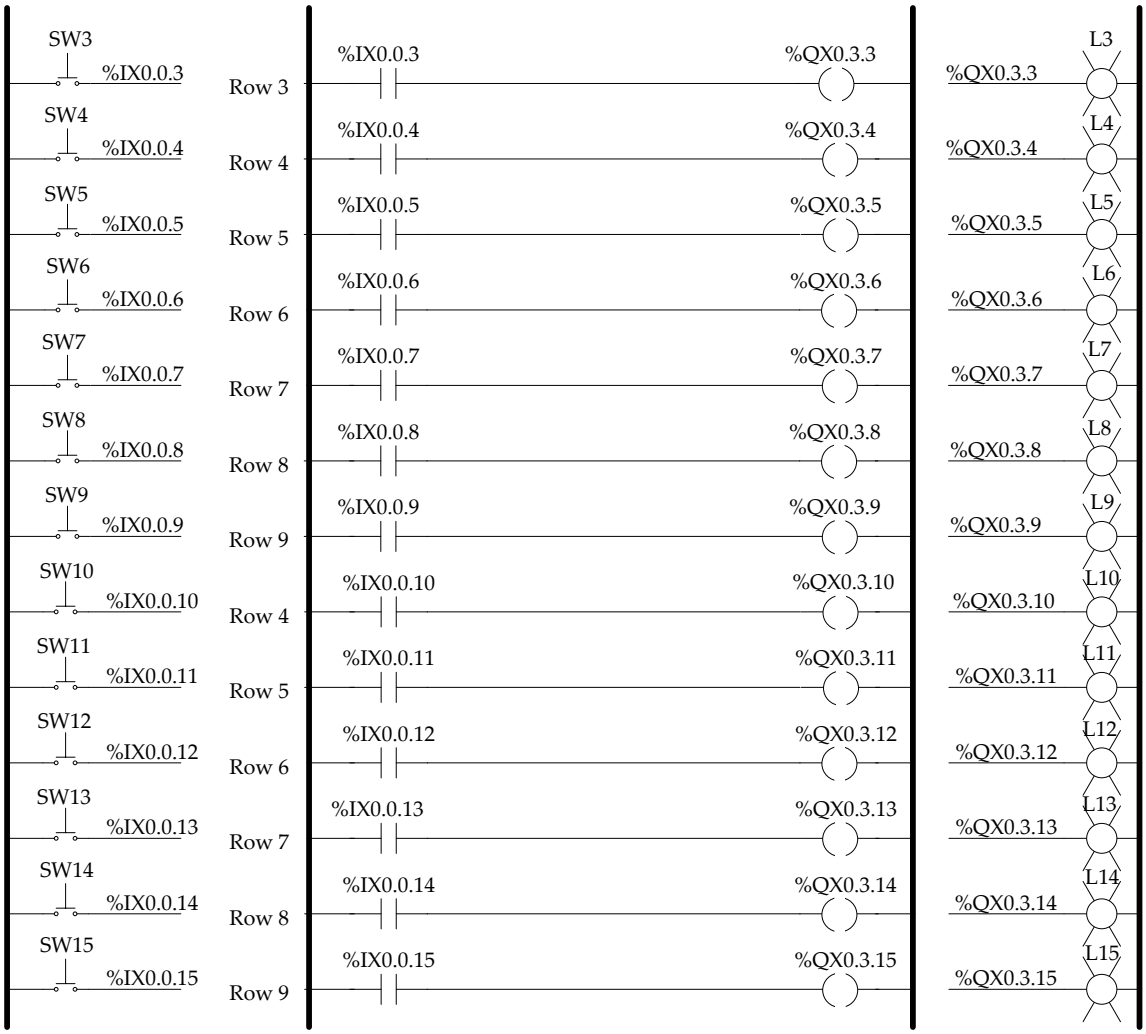
- Khi công tắc đầu vào được nhấn sẽ có đèn chỉ thị đầu ra tương ứng sáng.

Chương trình và sơ đồ kết nối đầu vào/ra:



Hình 4.6. Hình ảnh kết nối thiết bị





Hình 4.7. Chương trình với đầu vào-ra tương ứng

4.2.2. Mạch chốt

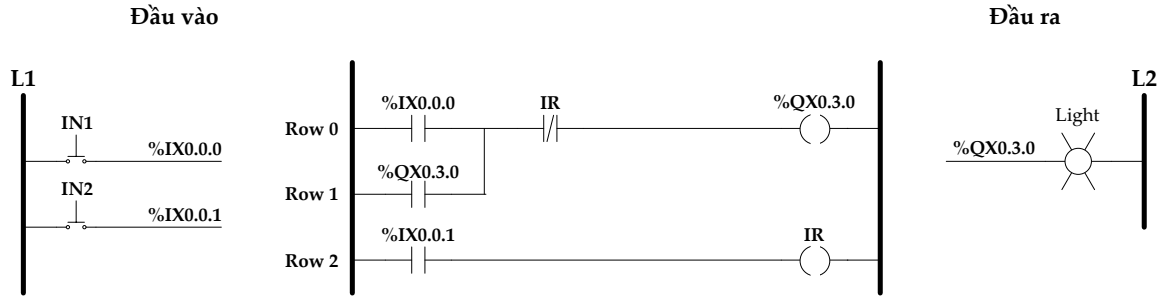
Mạch chốt là mạch được sử dụng để duy trì trạng thái của cuộn hút đầu ra. Chúng ta sẽ nghiên cứu mốt số thí dụ ứng dụng để hiểu về nguyên lý hoạt động của mạch chốt.

Ví dụ 3:

- Tại bậc 0, khi nút nhấn thường mở IN1 được nhấn, đầu ra %QX0.3.0 có giá trị logic là TRUE. Điều này sẽ khiến cho tiếp điểm thường mở %QX0.3.0 tại bậc 1 được tiếp điện và cuộn hút đầu ra %QX0.3.0 được duy trì trạng thái TRUE ngay cả trong trường hợp tiếp điểm IN1 có giá trị logic là FALSE, và đèn (Light) sáng.

- Tại bậc 2, khi đầu vào tiếp điểm IN2 tiếp điện, rơle nội IR được kích hoạt và tiếp điểm thường đóng IR tại bậc 0 hở mạch, cuộn hút đầu ra %QX0.3.0 có trạng thái OFF và đèn tắt.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.8. Mạch chốt trạng thái

Ví dụ 4:

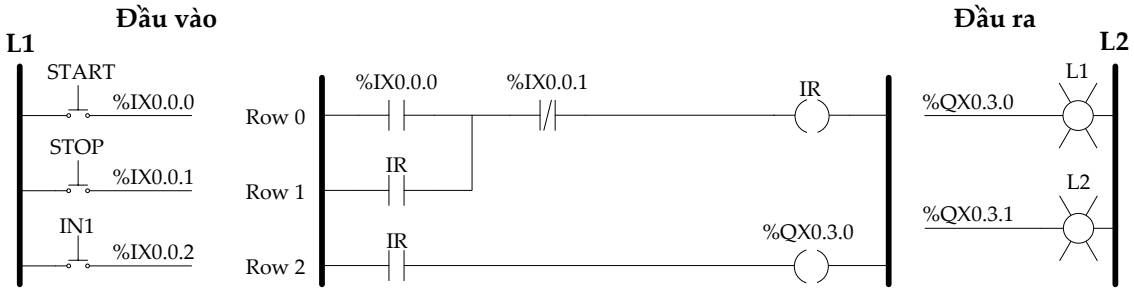
Giả sử một hệ thống gồm nhiều đầu ra có trạng thái ON/OFF phụ thuộc vào trạng thái của các đầu vào. Để điều khiển các đầu ra của thiết bị, chúng ta có thể viết chương trình điều khiển riêng biệt cho từng đầu ra. Tuy nhiên, để đơn giản hơn chúng ta có thể sử dụng phương pháp rơle nội.

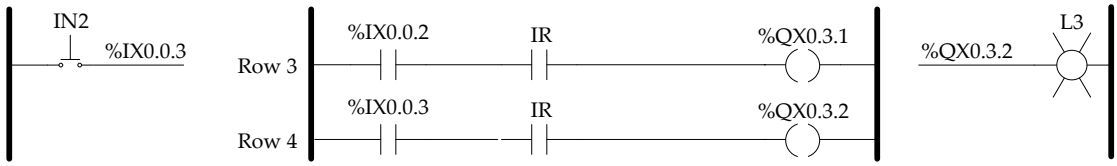
- Tại bậc 0 và 1, mạch chốt được sử dụng để duy trì trạng thái ON của rơle nội IR sau khi nút START được nhấn.

- Cuộn hút đầu ra %QX0.3.0 tại bậc 2 được cấp điện và chuyển sang trạng thái ON. Trạng thái các đầu ra %QX0.3.1 và %QX0.3.2 tương ứng tại bậc 3 và 4 cũng chuyển sang ON nếu các tiếp điểm thường mở đầu vào IN2, IN3 được tiếp điện.

- Nếu nút nhấn thường đóng STOP được nhấn thì tiếp điểm thường mở tại bậc 0 sẽ hở mạch và rơle nội IR chuyển sang trạng thái OFF và khi đó tất cả các đầu ra sẽ chuyển sang trạng thái OFF.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.9. Sử dụng rơle nội điều khiển nhiều đầu ra

4.2.3. Đầu vào/ra duy trì trạng thái hiện tại khi mất điện

Trong khi đang hoạt động nếu nguồn điện cung cấp cho PLC bị ngắt thì tất cả các cuộn hút nối với các thiết bị đầu ra và rơle nội sẽ có trạng thái OFF. Khi nguồn điện được khôi phục, trạng thái của các tiếp điểm đầu vào nối với các rơle sẽ được thiết lập khác nhau. Vì vậy, nếu PLC đang điều khiển một quá trình nào đó, nó sẽ tiếp tục điều khiển nhưng tại một điểm điều khiển khác trong chương trình. Để khắc phục vấn đề này, chúng ta có thể sử dụng một loại rơle nội có khả năng duy trì trạng thái để đảm bảo an toàn hoạt động của hệ thống trong trường hợp mất điện và do đó cho phép khởi động lại một cách thích hợp.

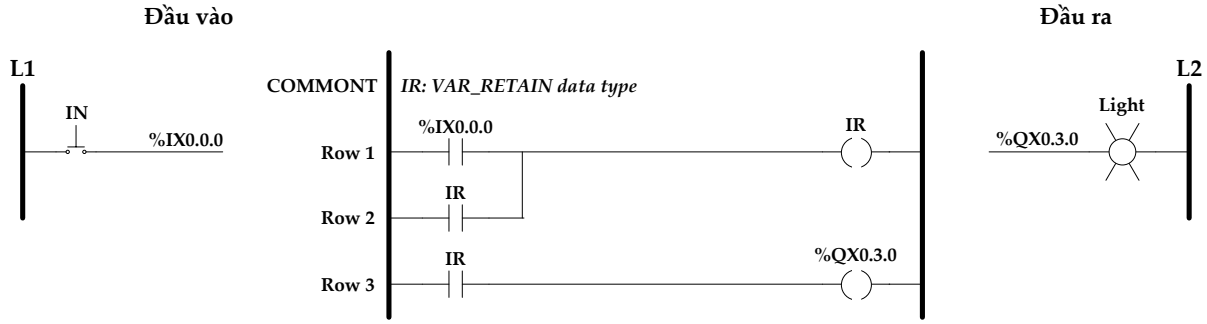
Ví dụ 5:

Hình 4.10 là chương trình được thiết kế điều khiển một hệ thống với khả năng duy trì trạng thái của cuộn hút đầu ra khi mất điện. Đầu ra IR là một rơle nội (kiểu dữ liệu là VAR_RETAIN) có khả năng lưu trạng thái khi xảy ra sự cố mất điện. Chương trình hoạt động của chương trình như sau:

- Bậc 1 và 2 tạo thành một mạch chốt, khi đầu vào IN được tiếp điện, rơle nội IR được kích hoạt và tiếp điểm đầu vào IR sẽ có trạng thái ON và duy trì trạng thái ON ngay cả khi đầu vào IN bị ngắt.

- Tại bậc 3, đầu ra %QX0.3.0 sẽ được kích hoạt và có trạng thái ON. Nếu xảy ra sự cố mất điện, IR vẫn duy trì trạng thái ON, trạng thái ON của đầu ra %QX0.3.0 vẫn được duy trì khi có điện trở lại.

<u>Chương trình và sơ đồ kết nối đầu vào/ra:</u>



Hình 4.10. Sử dụng biến lưu trạng thái khi mất điện

Ví dụ 6:

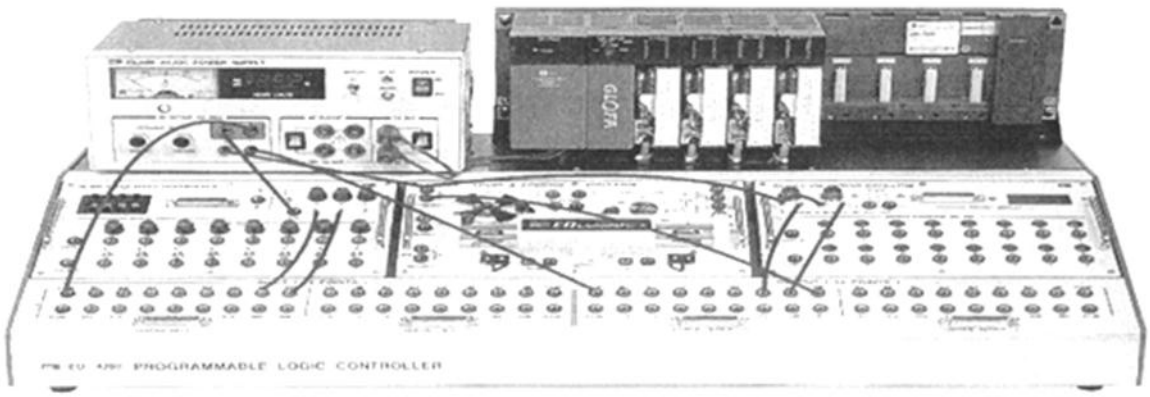
Hình 4.11 là chương trình được sử dụng để điều khiển động cơ DC. Hoạt động của chương trình có thể tóm tắt như sau:

- Đầu vào gồm 2 nút nhấn (nút khởi động và nút dừng hoạt động) và 3 đầu ra (điều khiển động cơ, đèn chỉ thị động cơ đang chạy và đèn chỉ động cơ dừng).

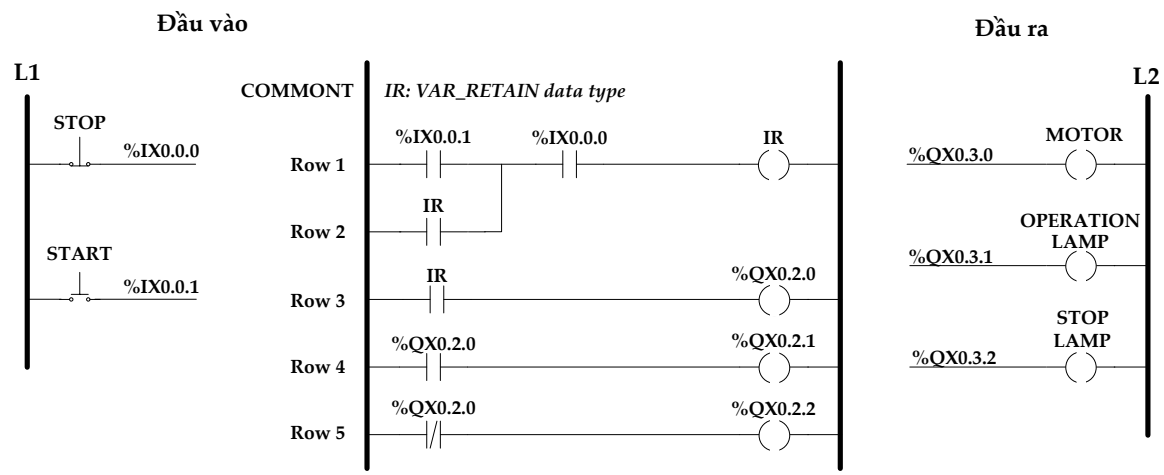
- Nếu START được nhấn, động cơ quay, đèn báo động cơ hoạt động sáng.

- Nếu động cơ đang chạy mà nút STOP được nhấn, động cơ sẽ dừng và đèn báo dừng sẽ sáng.

Chương trình và sơ đồ kết nối vào/ra:



Hình 4.11. Hình ảnh kết nối thiết bị



Hình 4.12. Chương trình điều khiển động cơ DC

Ví dụ 7:

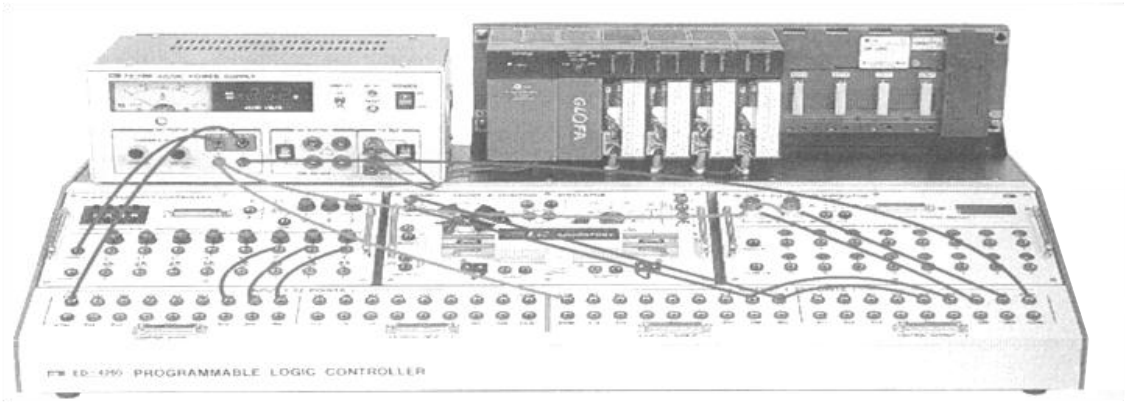
Hình 4.13 là chương trình được sử dụng để điều khiển thuận/nghịch động cơ. Hệ thống được xây dựng như sau:

- Đầu vào gồm 2 nút nhấn chọn chiều thuận/nghịch và 1 nút nhấn để dừng hoạt động của động cơ. Các đầu ra điều khiển các động cơ và các đèn chỉ thị.

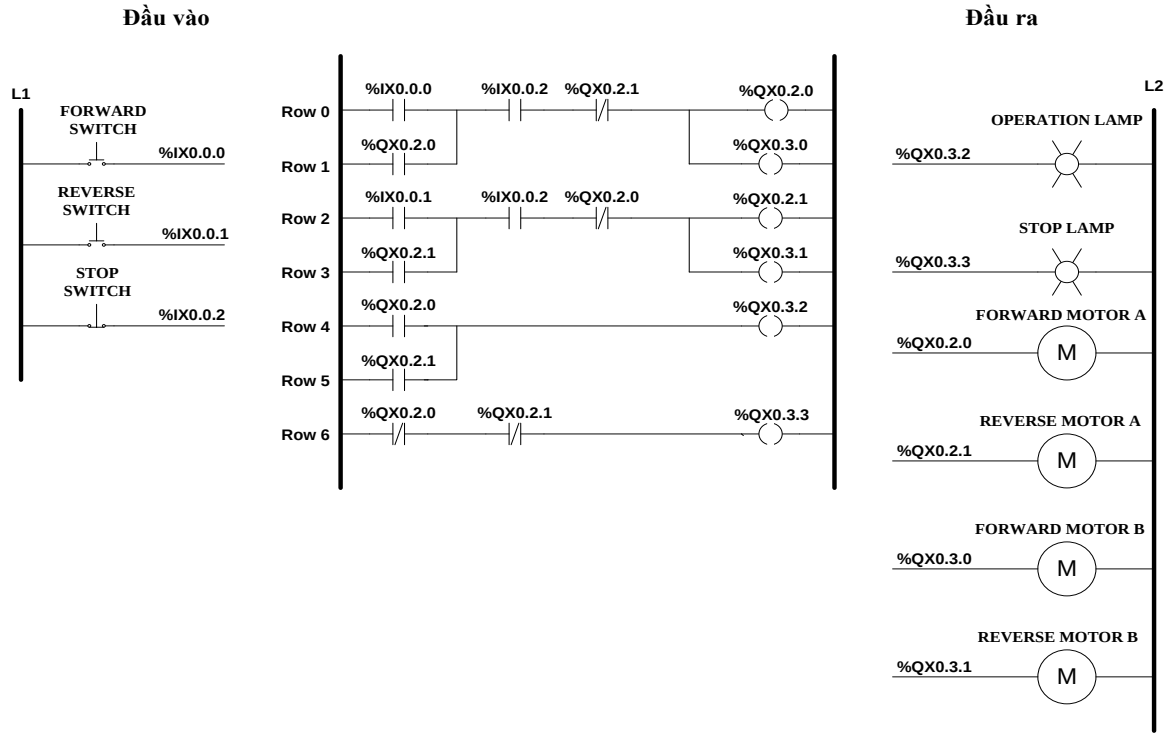
- Khi nhấn nút chọn chế độ quay theo chiều thuận thì động cơ bắt đầu quay thuận. Tương tự đối với nút nhấn chọn chế độ quay nghịch. Đèn OPERATION LAMP sáng khi động cơ hoạt động và đèn STOP LAMP sáng khi động cơ dừng hoạt động.

- Khi nút STOP SWITCH được nhấn thì mọi hoạt động của động cơ sẽ bị dừng và đèn báo STOP LAMP sẽ tắt. Nên dừng động cơ khi muốn đổi chiều quay.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.13. Hình ảnh kết nối thiết bị



Hình 4.14. Điều khiển thuận – nghịch động cơ DC

4.2.4. Câu lệnh hoạt động trong một chu kỳ quét

Một trong các lệnh được các nhà sản xuất PLC cung cấp đó là lệnh có khả năng bị kích hoạt chỉ trong 1 chu kỳ quét (xung đơn). Vì thế câu lệnh sẽ cung cấp một xung có thời gian cố định. Câu lệnh này thường được gọi là "One-Shot". Chúng ta cũng có thể dễ dàng phát triển một đoạn chương trình có chức năng như vậy bằng cách sử dụng kết hợp một vài lệnh đơn giản.

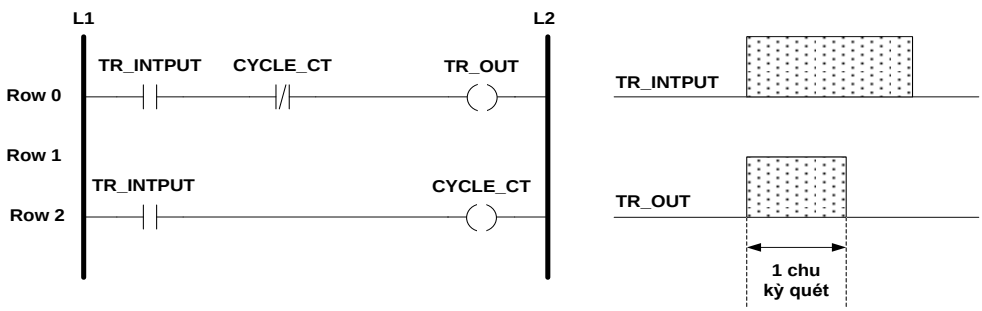
Ví dụ 8:

Hình 4.15 là chương trình có chức năng tạo ra một xung đơn với nguyên lý hoạt động như sau:

- Tại bậc 0, khi tiếp điểm đầu vào TR_INPUT tiếp điện, cuộn hút đầu ra TR_OUT được kích hoạt, tại bậc 2 cuộn hút CYCLE_CL cũng được kích hoạt.

- Do các câu lệnh tại bậc 2 được thực hiện sau khi các câu lệnh tại bậc 0 được thực hiện xong nên khi đầu ra CYCLE_CT trên bậc 2 có mức logic là TRUE sẽ làm hở mạch tiếp điểm thường đóng CYCLE_CT trên bậc 0 và trạng thái logic của cuộn hút đầu ra TR_OUT lúc này sẽ là FLASE. Như vậy đầu ra TR_OUT chỉ được ON trong một chu kỳ quét.

<u>Chương trình và giản đồ xung:</u>



Hình 4.15. Chương trình tạo ra xung đơn

Tuy nhiên, hiện nay một số phần mềm lập trình tích hợp sẵn câu lệnh có chức năng này. Lệnh One-Shot gồm hai loại: lệnh được kích hoạt với xung sườn trước và lệnh được kích hoạt với xung sườn sau .

Hình 4.16. Lệnh tạo xung đơn

Ví dụ 9:

Chúng ta hay nghiên cứu chương trình được cho trên Hình 4.17. Chương trình này được sử dụng để điều khiển động cơ với nguyên lý hoạt động như sau:

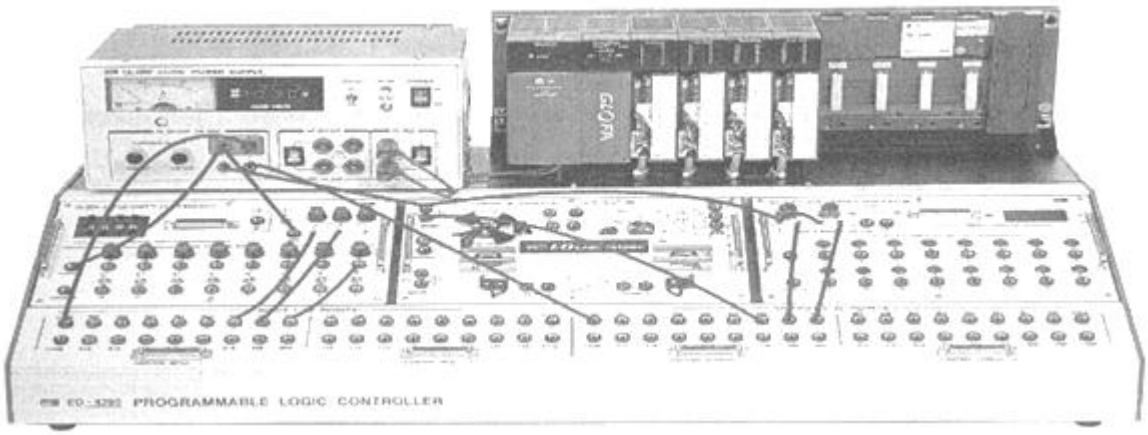
- Mạch được xây dựng với 3 đầu vào và 3 đầu ra.

- Khi nút nhấn thường mở START 1 được nhấn sẽ tạo ra một tín hiệu chuyển từ OFF san ON và cuộn hút được kích hoạt bởi xung sườn trước (POSITIVE) sẽ được kích hoạt và động cơ bắt đầu chạy đồng thời đèn OPERATION LAMP sáng.

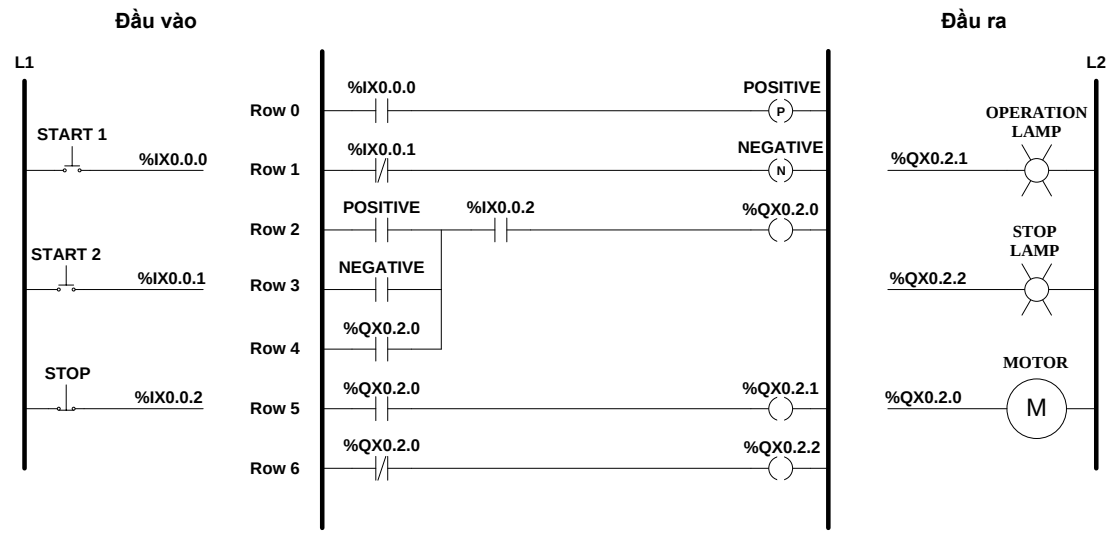
- Khi nút nhấn thường mở START 2 được nhấn sẽ tạo ra một tín hiệu chuyển từ ON sang OFF và cuộn hút được kích hoạt bởi xung sườn sau (NEGATIVE) sẽ được kích hoạt và động cơ bắt đầu chạy đồng thời đèn OPERATION LAMP sáng.

- Khi nhấn nút nhấn thường đóng STOP sẽ làm dừng hoạt động của động cơ và đèn báo STOP LAMP sáng và đèn OPERATION LAMP sẽ tắt.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.17. Hình ảnh kết nối thiết bị



Hình 4.18. Chương trình sử dụng xung đơn điều khiển động cơ

4.2.5. Lệnh SET và RESET

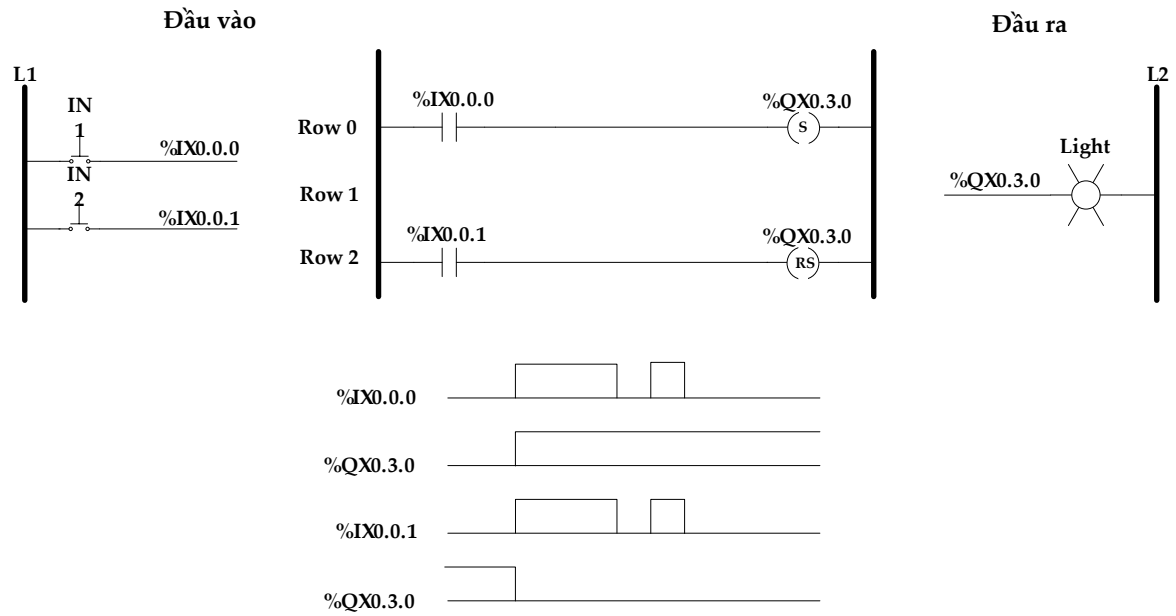
Một trong các lệnh được sử dụng thường xuyên trong quá trình lập trình đó là lệnh thiết lập trạng thái (SET) và lệnh khởi động lại trạng thái (RESET) cho cuộn hút đầu ra khi được thiết lập bởi câu lệnh SET.

Ví dụ 10:

- Với câu lệnh SET, trạng thái cuộn hút đầu ra %QX0.3.0 là ON khi đầu vào %IX0.0.0 có giá trị logic là TRUE và duy trì trạng thái ON cho dù đầu vào %IX0.0.0 chuyển sang trạng thái OFF và chỉ được khởi động lại bằng câu lệnh RESET.

- Trạng thái cuộn hút đầu ra %QX0.3.0 được thiết lập bởi lệnh SET sẽ được khởi động lại tới trạng thái OFF bởi câu lệnh RESET khi tiếp điểm %IX0.0.1 tiếp điện và duy trì trạng thái hiện tại tới khi được thiết lập bởi câu lệnh SET.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.19. Nguyên lý hoạt động của lệnh SET và RESET

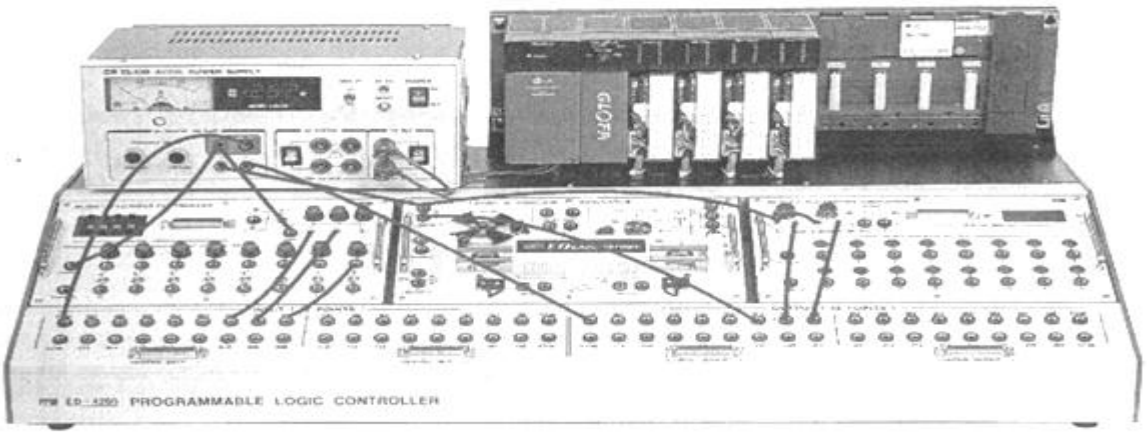
Ví dụ 11:

Hình 4.20 là chương trình sử dụng lệnh SET và RESET để điều khiển động cơ với nguyên lý hoat động như sau:

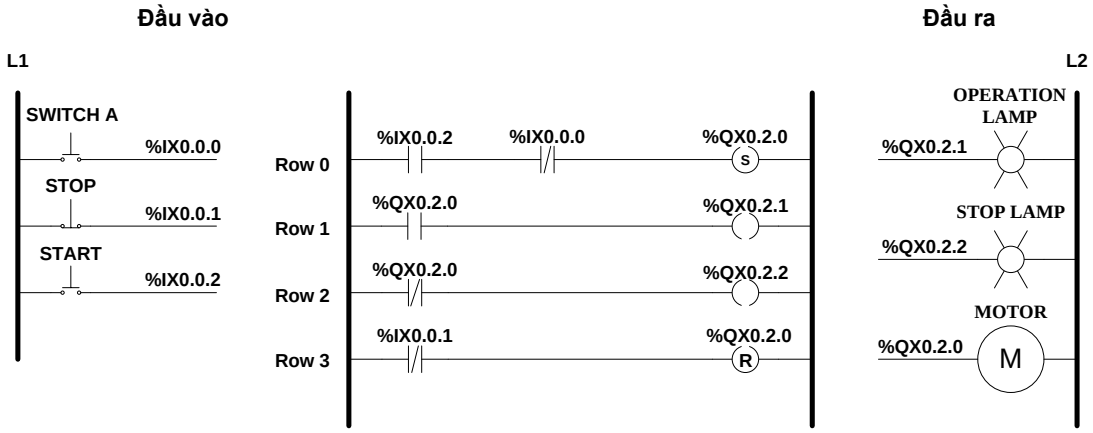
- Nếu nút nhấn thường mở START được nhấn, trạng thái ON của cuộn hút đầu ra %QX0.2.0 được thiết lập bởi câu lệnh SET vì vậy trạng thái ON của nó sẽ được duy trì cho dù nút nhấn thường mở SWITCH_A được nhấn và đèn chỉ thị OPERATION LAMP sáng.

- Nút nhấn thường đóng STOP được nhấn trong khi động cơ đang hoạt động thì trạng thái ON của cuộn hút đầu ra %QX0.2.0 được khởi động lại bởi câu lệnh RESET, động cơ dừng hoạt động và đèn STOP LAMP được bật.

Chương trình và sơ đồ kết nối đầu vào/ra:



Hình 4.20. Hình ảnh kết nối thiết bị



Hình 4.21. Sử dụng lệnh SET và RESET để điều khiển động cơ

4.2.6. Cặp lệnh điều khiển MCS và MCSCLR

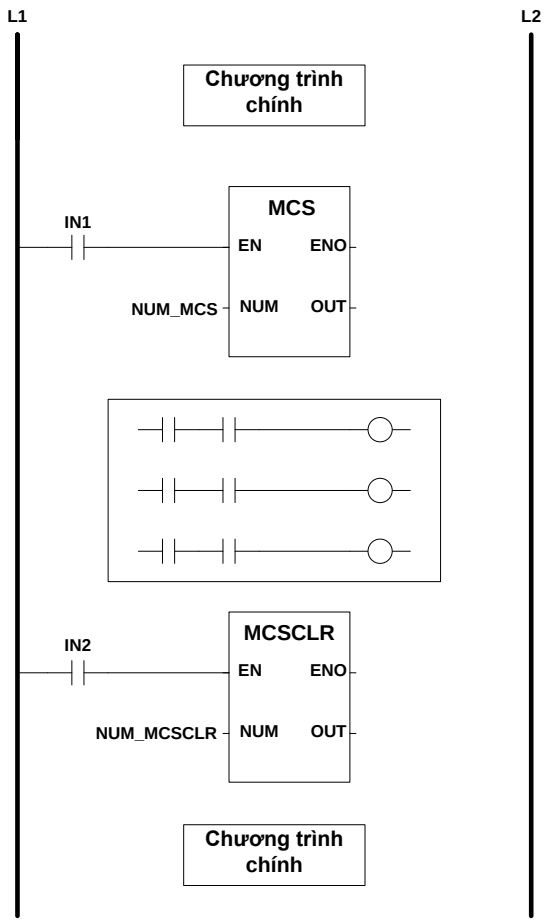
Chúng ta có thể sử dụng kết hợp hai câu lệnh MCS (Master Control) và MCSCLR (Master Control Clear) khi cần điều khiển một nhóm các bậc thang cùng phụ thuộc vào 1 điều kiện nào đó.

- Khi đầu vào cho phép câu lệnh hoạt động EN có mức logic là TRUE sẽ cho phép tất cả các bậc thang nằm giữa hai câu lệnh MCS và MCSCLR được phép điều khiển bởi các điều kiện đầu vào tương ứng. Chúng ta có thể nối trực tiếp đầu vào EN của câu lệnh MCS và MCSCLR vào đường L1.

- Đầu vào NUM (tối đa là 15) của câu lệnh MCS là số lệnh MCS được thực hiện trước khi câu lệnh MCS hiện tại được thực hiện tính từ lệnh MCS đầu tiên (các câu lệnh MCS lồng nhau). Chú ý, nếu NUM là 0 thì tất cả các câu lệnh MCS trước phải được thực hiện thì câu lệnh MCS hiện tại mới hoạt động. Tương tự đối với câu lệnh MCSCLR, đầu vào NUM (tối đa là 15) là số câu lệnh MCS được khởi động lại.

- Đối với cả hai câu lệnh, đầu ra ENO sẽ có giá trị là một nếu câu lệnh được thực hiện thành công và ngược lại.

- Đối với câu lệnh MCS và MCSCLR, đầu ra OUT phải được gán tới 1 biến hình thức (DUMMY).



Hình 4.22. Nguyên lý hoạt động của câu lệnh MCS và MCSCLR

Ví dụ 12:

Để hiểu thêm về nguyên tắc hoạt động của câu lệnh MCS và MCSCLR, chúng ta hãy xét một ví dụ như Hình 4.23 với nguyên lý hoạt động như sau:

- Nếu nút nhấn SW0, SW1 được nhấn, đầu ra %QX3.0.0 có trạng thái ON, đèn Light 1 sáng.

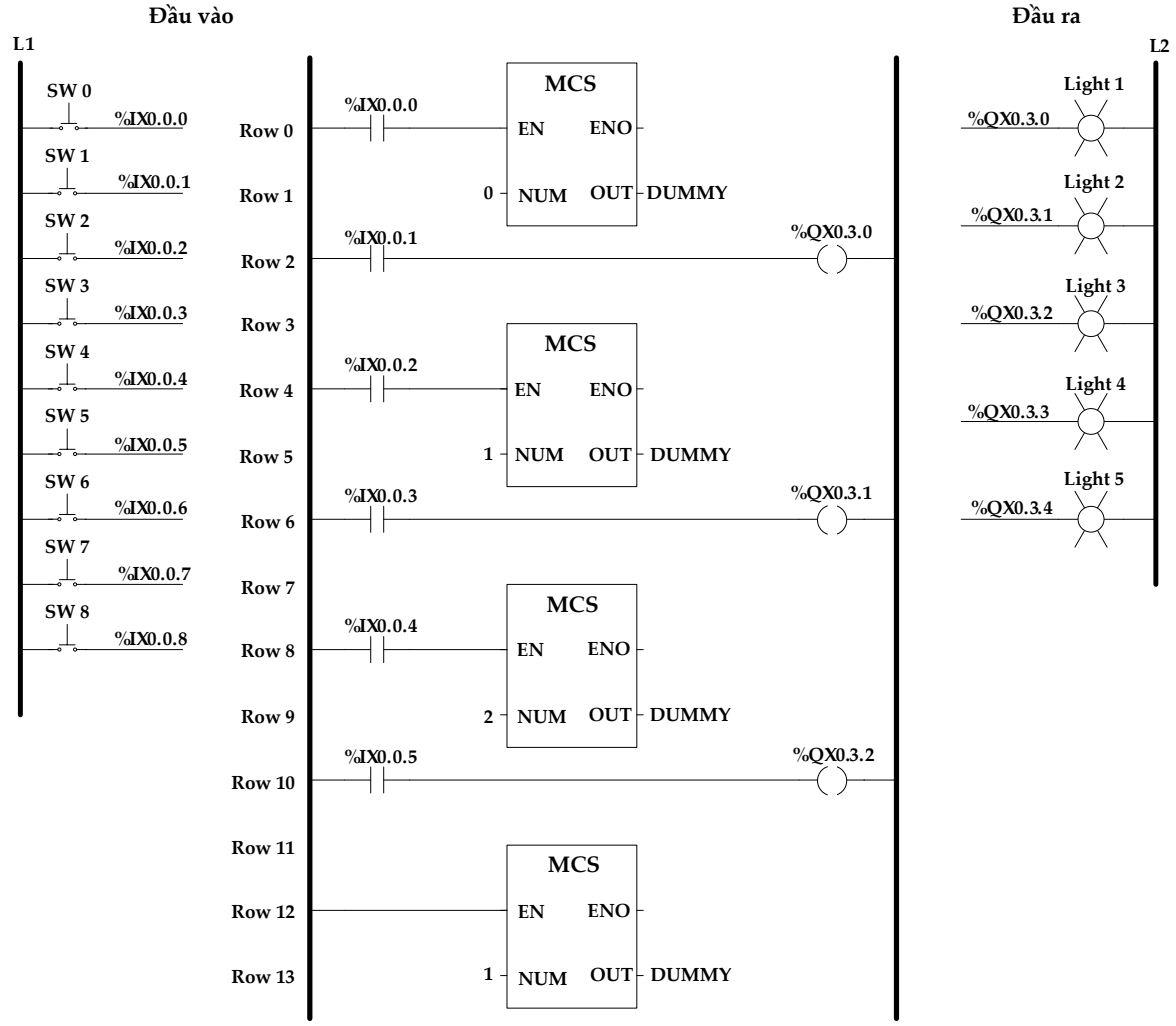
- Nếu nút nhấn SW0, SW2, SW3 được nhấn, đầu ra %QX3.0.1 có trạng thái ON, đèn Light 2 sáng.

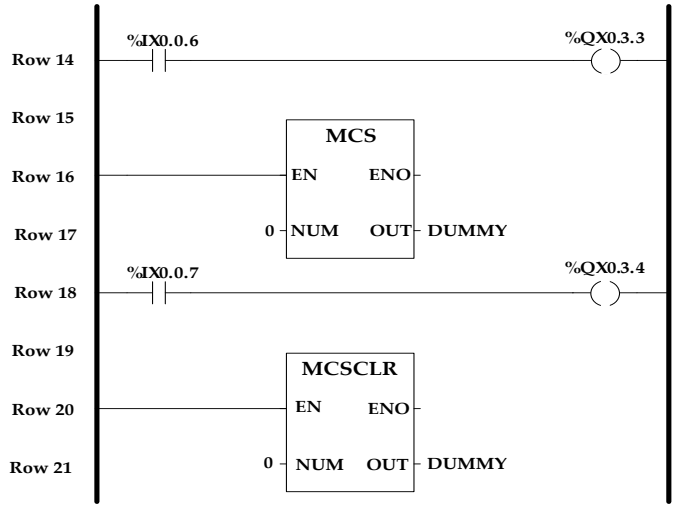
- Nếu nút nhấn SW0, SW2, SW4, SW5 được nhấn, đầu ra %QX3.0.2 có trạng thái ON, đèn Light 3 sáng.

- Nếu nút nhấn SW0, SW6 được nhấn, đầu ra %QX3.0.3 có trạng thái ON, đèn Light 4 sáng.

- Nếu nút nhấn SW0, SW2, SW4, SW7 được nhấn, đầu ra %QX3.0.4 có trạng thái ON, đèn Light 5 sáng.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.23. Ví dụ sử dụng câu lệnh MCS và MCSCLR

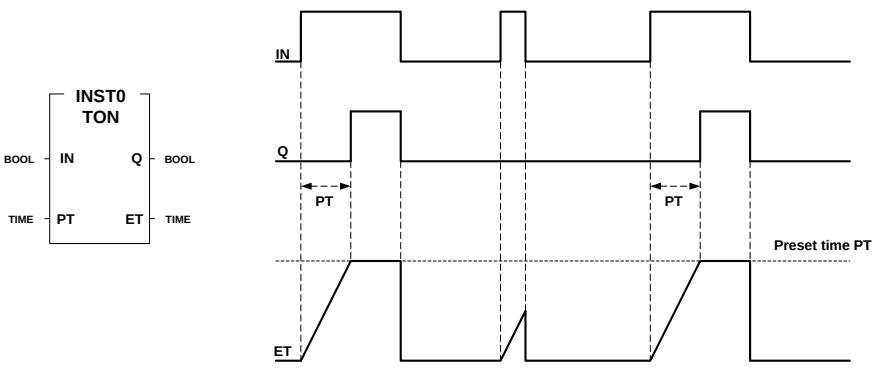
4.3. CÁC BỘ ĐỊNH THỜI

Trong nhiều ứng dụng chúng ta cần phải kiểm soát thời gian hoạt động của thiết bị. Ví dụ, chúng ta cần phải kiểm soát hoạt động của động cơ trong một khoảng thời gian cụ thể nào đó. Vì vậy, các thiết bị PLC thường được tích hợp sẵn các bộ định thời (Timer). Các bộ định thời hoạt động dựa trên xung nhịp của CPU.

Mỗi loại PLC khác nhau có thể có các loại bộ định thời khác nhau nhưng về cơ bản có thể phân thành ba loại sau: bộ định thời tạo trễ (on-delay timer), bộ định thời ngắt trễ (off-delay timer), bộ định thời tạo xung (pusle timer).

4.3.1. Bộ định thời tạo trễ

Với bộ định thời tạo trễ (TON), khi đầu vào IN chuyển từ trạng thái OFF sang ON thì bộ định thời sẽ bắt đầu đếm và được thể hiện qua thành phần đầu ra ET. Khi giá trị ET đạt tới giá trị đặt trước (PT) thì đầu ra Q của bộ định thời chuyển sang trạng thái ON.



Hình 4.24. Giản đồ xung của bộ định thời tạo trễ TON

Ví dụ 13:

Hình 4.30 là chương trình điều khiển động cơ 3 pha với nguyên lý hoạt động như sau:

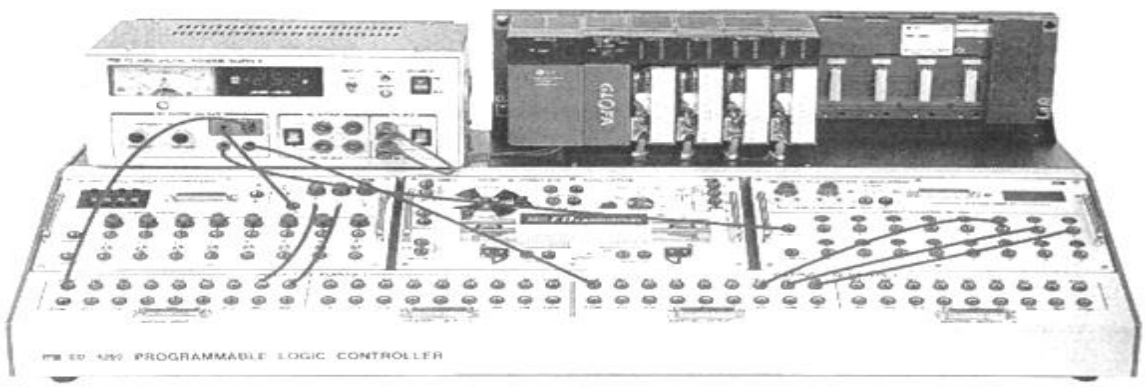
- Chương trình được thiết kế với 2 đầu vào và 3 đầu ra. Các đầu vào được sử dụng để khởi động và dừng hoạt động của động cơ. Các đầu ra được sử dụng để điều khiển đèn chỉ thị.

- Khi nút nhấn thường mở START được nhấn, chế độ Y_OPERATION được kích hoạt, bộ định thời T1 (giá trị đặt trước 5s) cũng được kích hoạt, đèn chỉ thị cũng sẽ sáng.

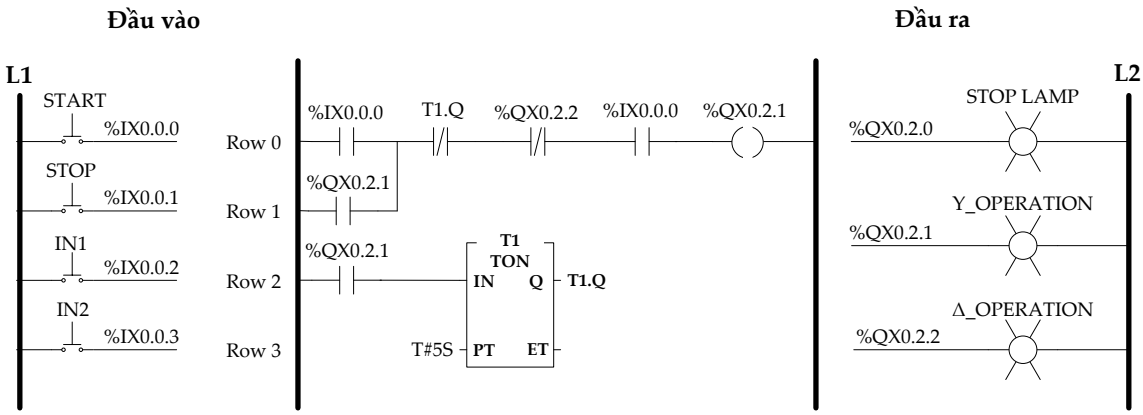
- Sau khoảng thời gian 5s, trạng thái đầu ra Q của T1 là ON và chế độ Y_OPERTION kết thúc chuyển sang chế độ Δ_OPERATION, đèn chỉ thị cho chế độ cũng sẽ sáng.

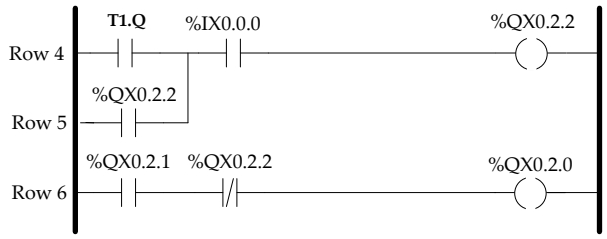
- Trong quá trình hoạt động, nếu nút nhấn thường đóng STOP được nhấn thì mọi quá trình hoạt động của động cơ sẽ bị ngắt và đèn chỉ thị tương ứng sẽ sáng.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.25. Hình ảnh kết nối thiết bị





Hình 4.26. Chương trình điều khiển động cơ 3 pha

4.3.1.1. Kết hợp các bộ định thời để điều khiển các sự kiện theo chuỗi

Có thể sử dụng kết hợp các bộ định thời tạo trễ TON để điều khiển các thiết bị một cách liên tiếp sau một khoảng thời gian nào đó.

Ví dụ 14:

Hình 4.32 là một chương trình được sử dụng để điều khiển 3 động cơ liên tiếp với nguyên lý hoạt động như sau:

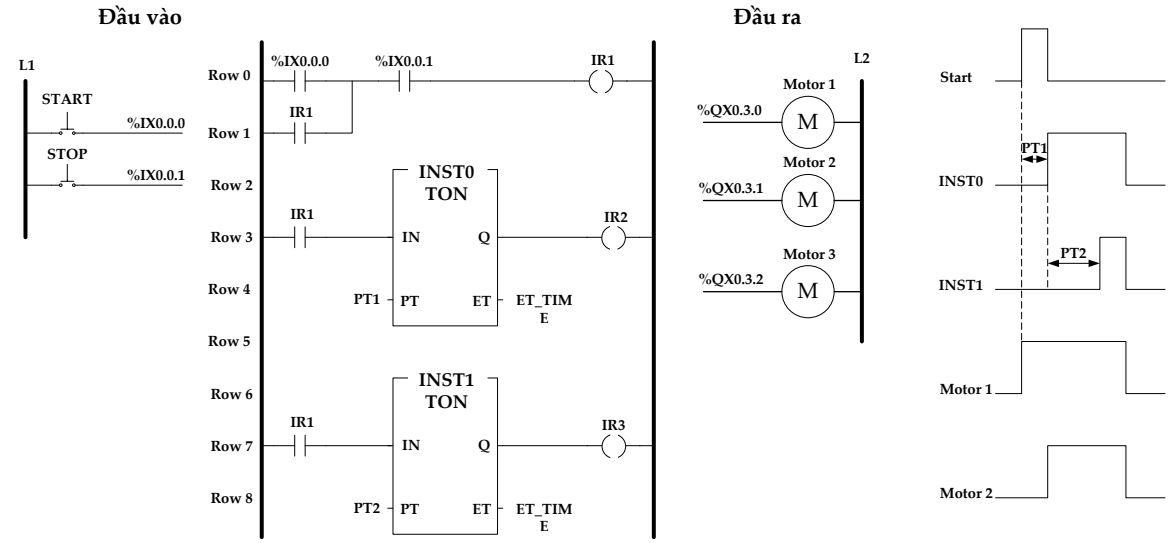
- Khi nút nhấn khởi động START được nhấn, đầu ra IR1 được chuyển sang trạng thái ON và được duy trì ở trạng thái đó nhờ mạch chốt và đồng thời khởi động bộ định thời T1, T2 và Motor 1.

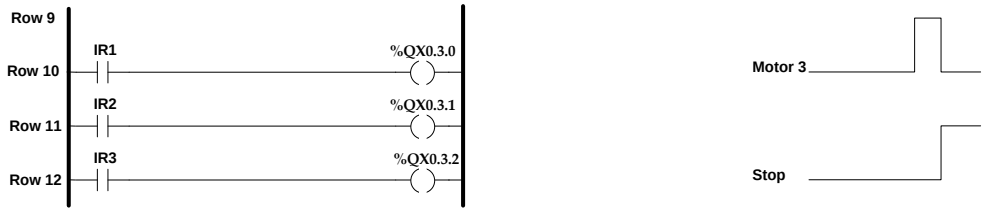
- Khi bộ định thời T1 đạt tới giá trị đặt trước PT1, đầu ra IR2 chuyển sang trạng thái ON, khởi động Motor 2.

- Khi bộ định thời T2 đạt tới giá trị đặt trước PT2, đầu ra IR3 chuyển sang trạng thái ON, Motor 3 được kích hoạt.

- Cả ba động cơ sẽ được dừng đồng thời khi nút dừng STOP được nhấn.

<u>Chương trình và kết nối các đầu vào/ra:</u>





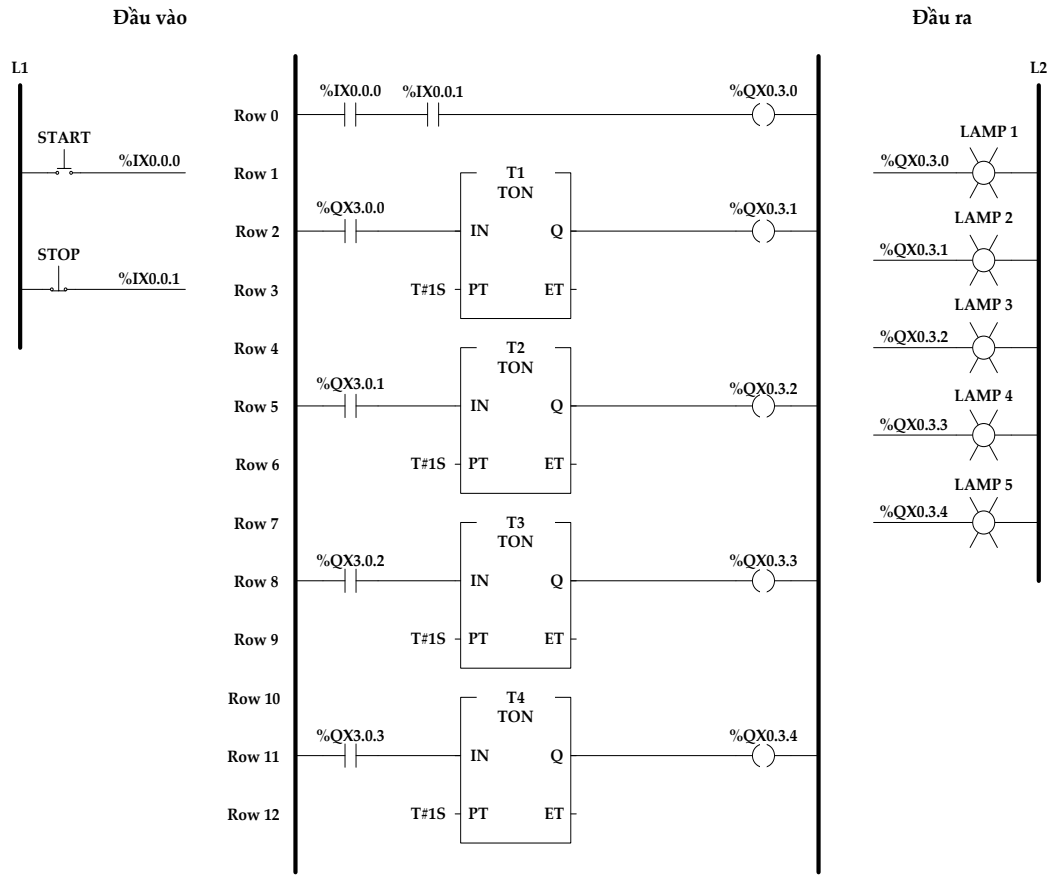
Hình 4.27. Chương trình điều khiển các động cơ hoạt động liên tiếp

Ví dụ 15:

Hình 4.33 là một ví dụ được sử dụng để bật/tắt các đèn theo thứ tự sau một khoảng thời gian định trước. Nguyên lý hoạt động của chương trình được tóm tắt như sau:

- Khi nút nhấn thường mở START được nhấn, các đèn sẽ được bật lần lượt sau khoảng thời gian 1s.
- Các đèn sẽ được tắt khi nút nhấn thường đóng STOP được nhấn.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.28. Chương trình bật-tắt các đèn liên tiếp

4.3.1.2. Kết hợp các bộ định thời để tạo trễ với thời gian lớn

Trong quá trình lập trình, chúng ta có thể kết hợp nhiều bộ định thời với nhau để tạo ra một khoảng thời gian trễ lớn.

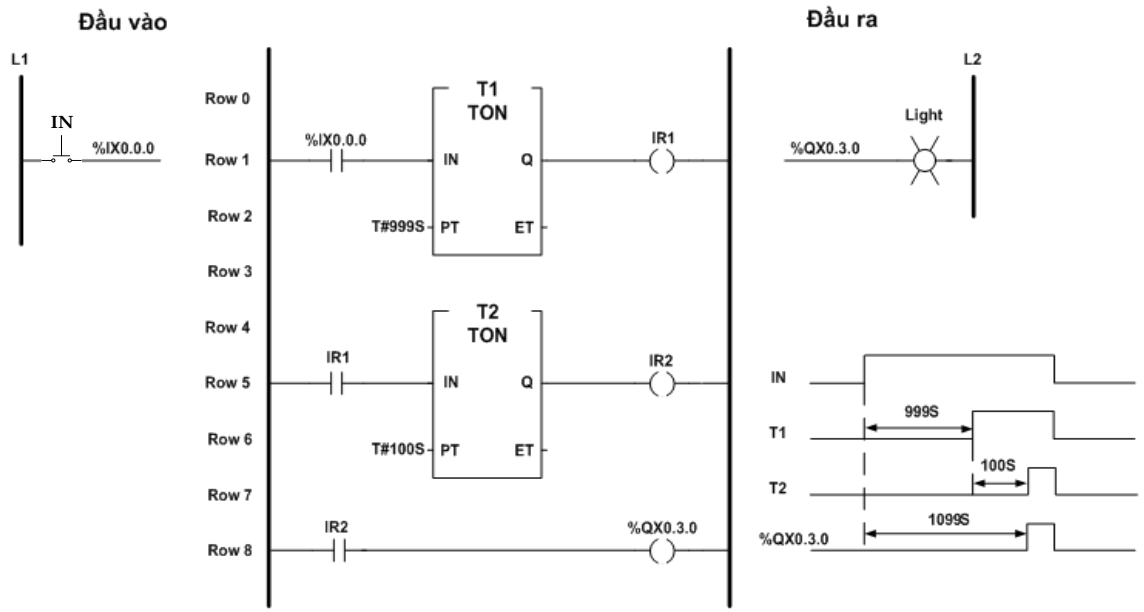
Ví dụ 16:

Hình 4.34 là một ví dụ kết hợp hai bộ định thời với nhau để tạo ra một khoảng thời gian trễ 1099s. Nguyên lý hoạt động của chương trình như sau:

- Bộ định thời T1 tạo ra một khoảng thời gian trễ là 999s. Bộ định thời này bắt đầu đếm khi nút nhấn thường mở đầu vào IN được nhấn. Khi giá trị đếm của bộ định thời đạt tới giá trị đặt trước 999s thì trạng thái đầu ra Q của T1 là ON, đồng thời bộ định thời T2 (bộ định thời tạo thời gian trễ là 100s) cũng được kích hoạt.

- Khi đầu ra Q của T2 chuyển sang trạng thái ON (sau 100s) thì cuộn hút đầu ra %QX0.3.0 tại bậc 8 sẽ được cấp điện và có giá trị logic là TRUE. Như vậy cuộn hút đầu ra %QX0.3.0 sẽ được kích hoạt sau một khoảng thời gian 1099s kể từ khi nút nhấn đầu vào IN được nhấn.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.29. Kết hợp các bộ định thời để tạo thời gian trễ lớn

4.3.1.3. Kết hợp bộ định thời tạo tín hiệu đóng/ngắt theo chu kỳ

Trong quá trình điều khiển đôi khi chúng ta cần phải sử dụng các xung tín hiệu có chu kỳ đóng/ngắt cố định nào đó. Để tạo ra các xung tín hiệu này, chúng

ta có thể sử dụng phương pháp đơn giản đó là kết hợp hai bộ định thời TON với nhau.

Ví dụ 17:

Hình 4.35 là một đoạn chương trình sử dụng hai bộ định thời tạo trễ (TON) để tạo ra các xung tín hiệu có chu kỳ đóng/ngắt là 5s. Hoạt động của chương trình như sau:

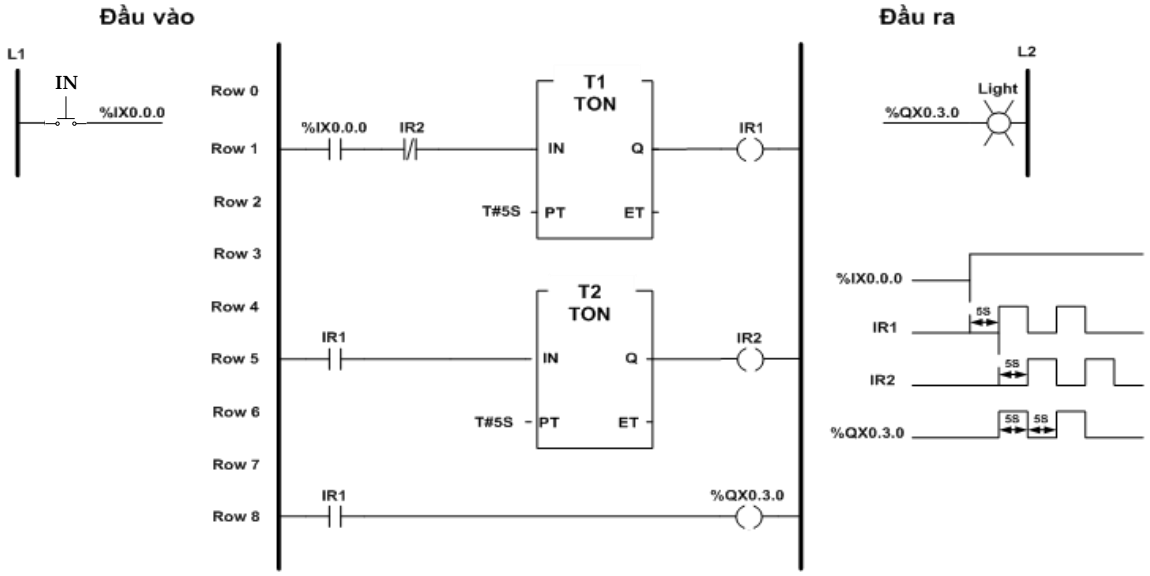
- Bộ định thời có chức năng tạo ra các xung có thời gian ON là 5s và thời gian OFF là 5s.

- Khi nút nhấn thường mở đầu vào IN được nhấn, bộ định thời T1 (giá trị đặt trước là 5s) bắt đầu hoạt động. Sau thời gian 5s, đầu ra Q của T1 có mức logic là TRUE và do đó sẽ kích hoạt bộ định thời T2 (giá trị đặt trước 5s) và cuộn hút đầu ra %QX0.3.0.

- Sau 5s, đầu ra Q của T2 có giá trị là TRUE và tiếp điểm thường đóng IR2 tại bậc 1 sẽ bị hở mạch. Kết quả là đầu ra Q của T1 chuyển sang trạng thái OFF, đầu ra Q của T2 tại bậc 5 cũng có trạng thái OFF, bộ định thời T1 được kích hoạt và quá trình sẽ được lặp lại.

- Kết quả là đèn (Light) sẽ sáng/tắt sau 5s.

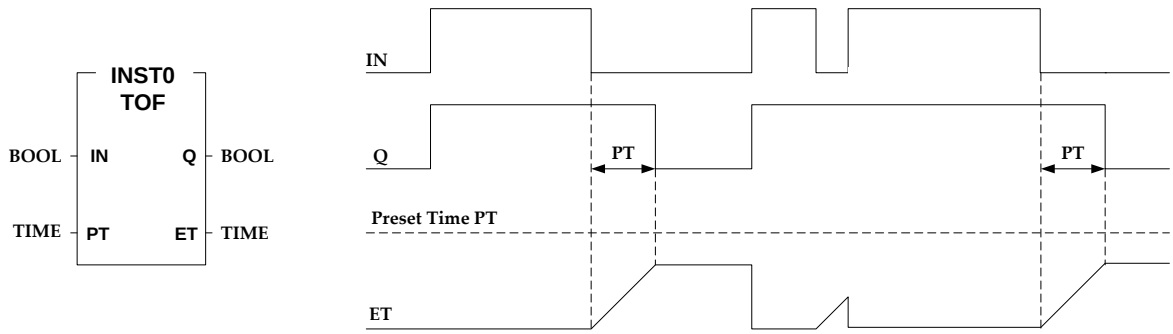
<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.30. Kết hợp các bộ định thời TON tạo tín hiệu đóng-ngắt theo chu kỳ

4.3.2. Bộ định thời tạo trễ ngắt

Với bộ định thời tạo trễ ngắt TOF, khi giá trị logic của đầu vào IN là TRUE thì giá trị logic đầu ra Q cũng là TRUE. Bộ định thời bắt đầu đếm khi đầu vào IN được đưa về trạng thái logic FALSE. Khi thời gian đếm hiện tại của bộ định thời ET đạt tới giá trị đặt trước PT thì giá trị logic của đầu ra Q trở về mức logic FALSE. Nếu đầu vào IN chuyển sang trạng thái logic TRUE trước khi giá trị ET đạt tới giá trị PT thì giá trị ET được đưa về giá trị 0.



Hình 4.31. Giản đồ xung của bộ định thời tạo ngắt TOF

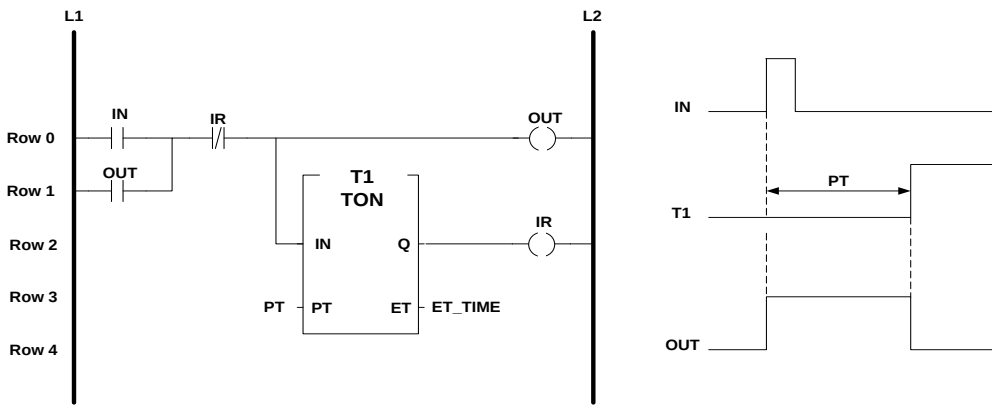
Ví dụ 18:

Hình 4.37 là chương trình sử dụng bộ định thời TON để tạo ngắt trễ.

- Khi tiếp điểm IN có giá trị logic TRUE, đầu ra OUT chuyển sang trạng thái ON, bộ định thời T1 được kích hoạt. Trạng thái hiện tại của đầu ra OUT được duy trì ngay cả khi đầu vào IN chuyển sang trạng thái OFF nhờ mạch chốt.

- Khi giá trị hiện tại ET đạt tới giá trị đặt trước PT thì đầu ra Q của T1 có mức logic TRUE. Điều này sẽ khiến cho tiếp điểm thường đóng IR tại bậc 0 bị hở mạch và đầu ra OUT chuyển sang trạng thái OFF.

<u>Chương trình và giản đồ xung:</u>



Hình 4.32. Sử dụng bộ định thời TON tạo trễ ngắt

Hiện nay, hầu hết các loại PLC có tích hợp sẵn các bộ định thời trễ ngắt TOF. Vì vậy sẽ là rất tiện lợi cho những người lập trình PLC vì chương trình sẽ đơn giản và ngắn gọn hơn so với việc sử dụng bộ định thời TON để tạo trễ ngắt.

Ví dụ 19:

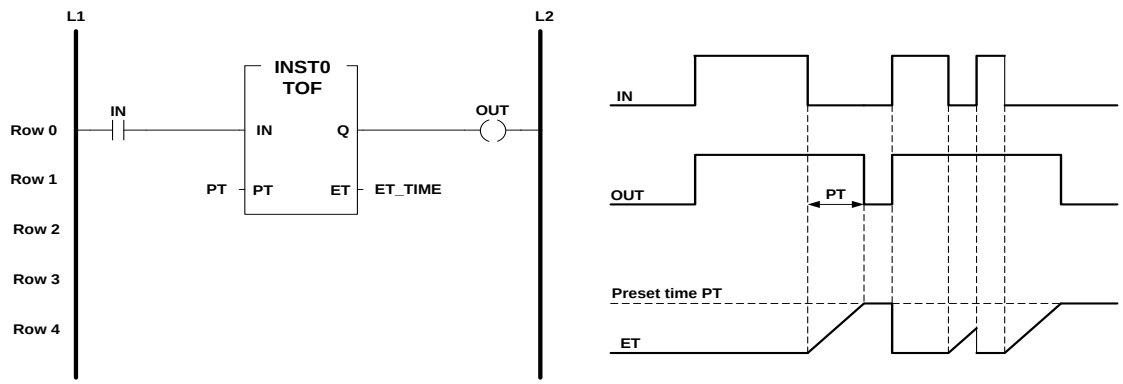
Hình 4.38 là một ví dụ sử dụng bộ định thời ngắt trễ TOF với nguyên tắc hoạt động như sau:

- Khi đầu vào IN có giá trị logic là TRUE thì đầu ra Q cũng có giá trị logic là TRUE.

- Bộ định thời chỉ bắt đầu đếm khi đầu vào IN chuyển từ mức logic TRUE sang mức logic FALSE. Khi giá trị đếm ET bằng với giá trị đặt trước PT thì đầu ra Q của bộ định thời chuyển về trạng thái có mức logic FALSE.

- Trong quá trình đếm nếu đầu vào IN được đưa trở lại mức logic TRUE trước khi ET đạt tới giá trị PT thì giá trị đếm ET sẽ được xoá về 0.

<u>Chương trình và giản đồ xung:</u>



Hình 4.33. Nguyên lý hoạt động của bộ định thời tạo trễ ngắt

Ví dụ 20:

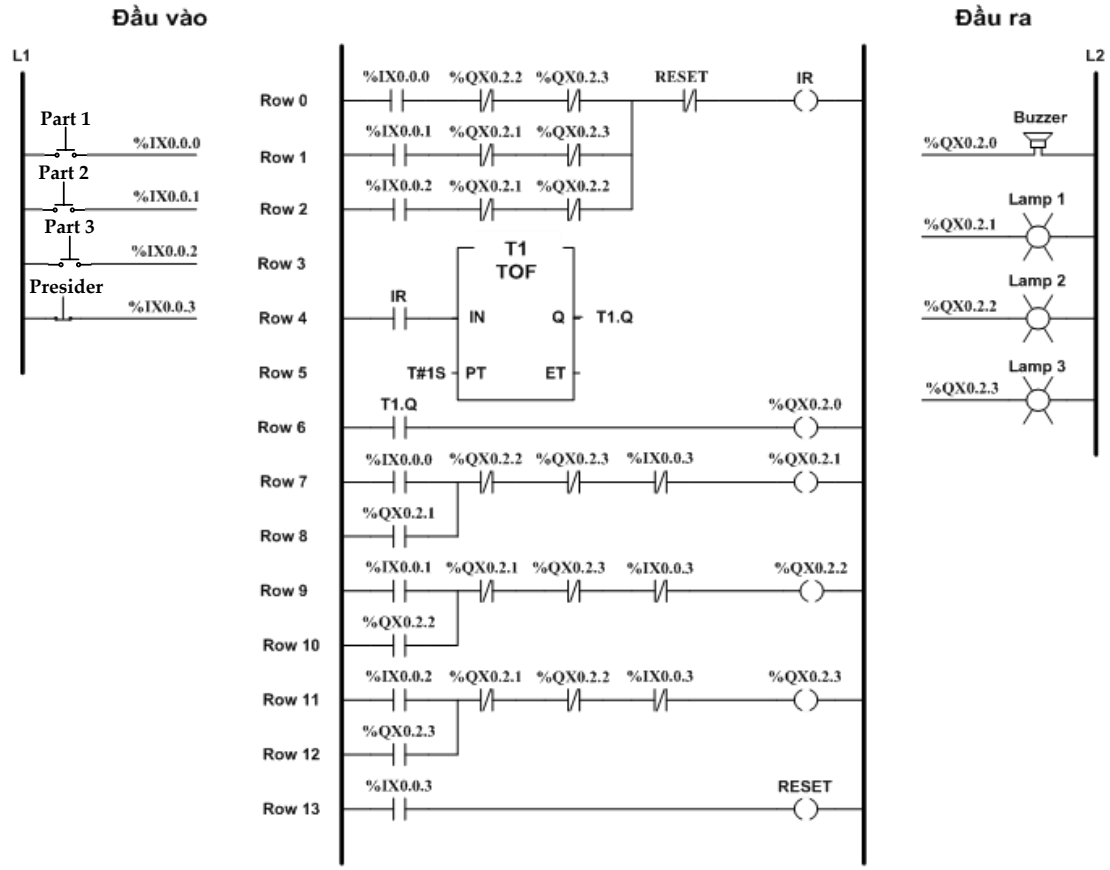
Hình 4.39 là một chương trình ví dụ sử dụng bộ định thời tạo trễ ngắt TOF. Trong chương trình này, cả người dẫn chương trình và người chơi đều tham gia. Nguyên tắc hoạt động của chương trình như sau:

- Người dẫn chương trình đưa ra câu hỏi cho người chơi. Quyền ưu tiên thuộc về người nào nhấn chuông trước.

- Khi người đầu tiên nhấn chuông, đèn ứng với người đó sẽ sáng và chuông kêu trong 1s.

- Đèn do người đầu tiên nhấn sẽ sáng cho đến khi người dẫn chương trình nhấn nút khởi động lại.

Chương trình và sơ đồ kết nối các đầu vào/ra:



Hình 4.34. Chương trình trò chơi

4.3.3. Bộ định thời tạo xung

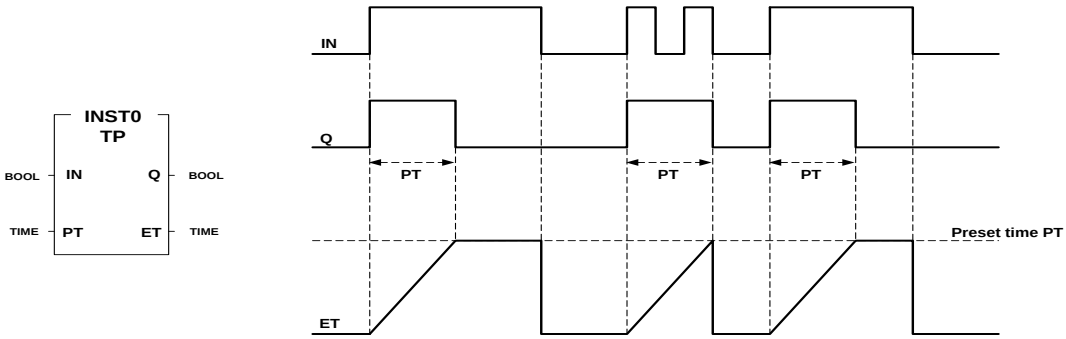
Các bộ định thời tạo xung (TP) được sử dụng để tạo ra các xung đầu ra có độ rộng xung (thời gian ON) cố định. Nguyên lý hoạt động động của câu lệnh TP được mô tả trên Hình 4.40 dưới đây:

- Khi đầu vào IN chuyển từ trạng thái OFF sang ON, bộ định thời bắt đầu đếm (ET tăng dần).

- Khi giá trị ET đạt tới giá trị đặt trước (PT) thì giá trị ET sẽ được duy trì ở giá trị của PT cho tới khi đầu vào IN được đưa về trạng thái OFF.

- Trong khi trạng thái của đầu vào IN là ON thì trạng thái đầu ra Q cũng là ON và trạng thái của Q chỉ được đưa về OFF khi giá trị ET bằng PT.

- Khi bộ định thời bắt đầu hoạt động (ET tăng dần), trạng thái của đầu vào IN không làm thay đổi quá trình đếm của ET.



Hình 4.35. Nguyên lý hoạt động của bộ định thời tạo xung

Ví dụ 21:

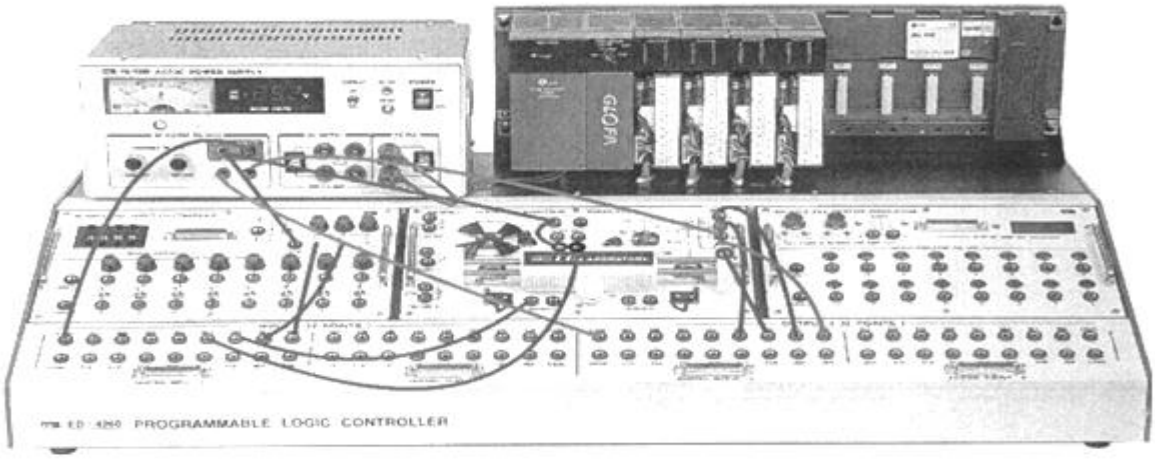
Hình 4.41 là một chương trình được sử dụng để điều khiển động cơ bước kết hợp giữa bộ định thời TP và bộ đếm tiến CTU. Nguyên lý hoạt động của chương trình như sau:

- Chương trình được thiết kế với 3 đầu vào và 4 đầu ra. Các đầu vào được sử dụng để khởi động, dừng hoạt động của động cơ cũng như thu nhận tín hiệu từ cảm biến nhận biết động cơ quay hết 1 vòng. Các đầu ra được sử dụng để điều khiển các cuộn dây của đông cơ.

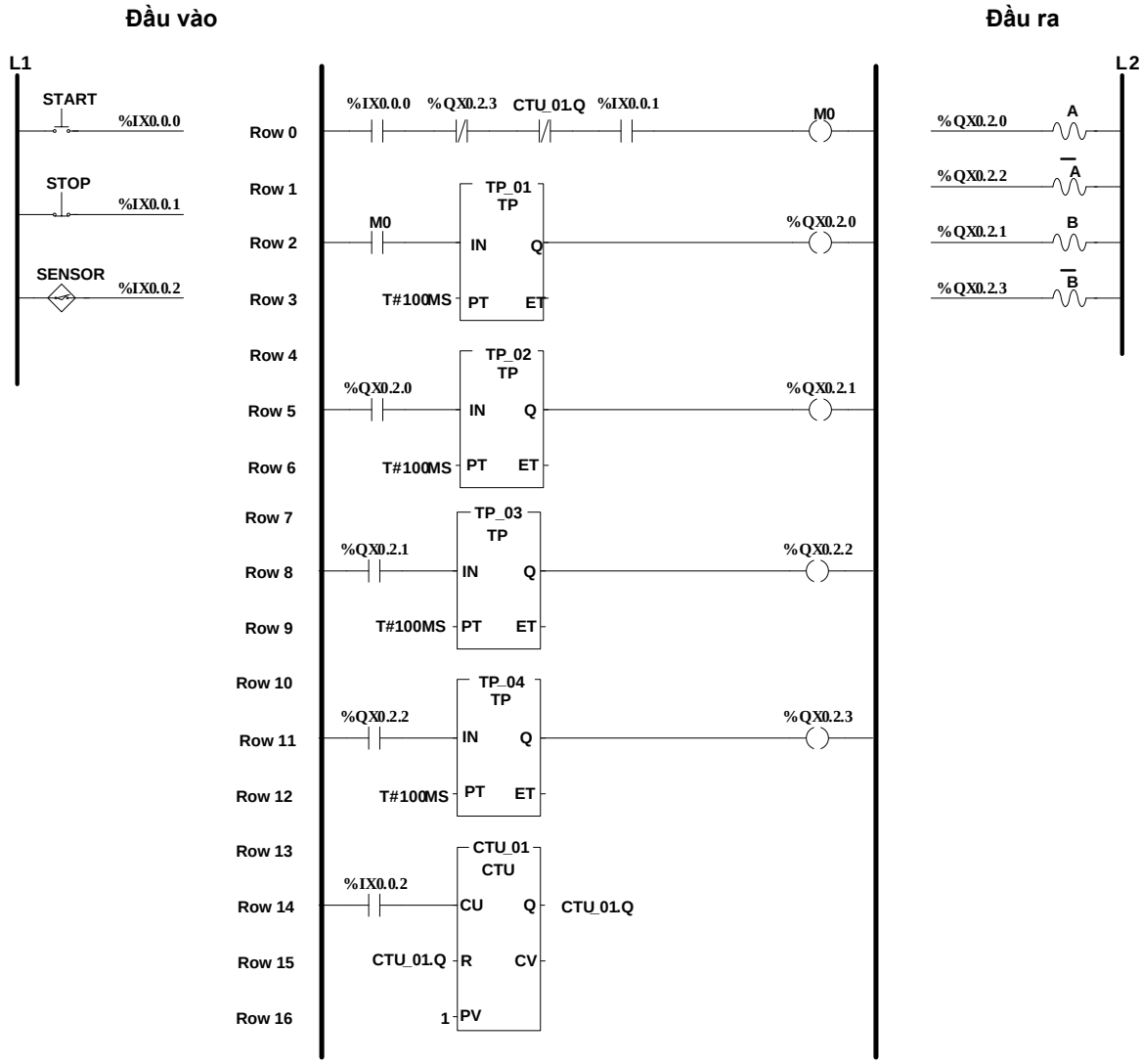
- Khi nút khởi động thường mở START được nhấn, động cơ bắt đầu chạy và dừng lại khi quay hết 1 vòng (cảm biến nhận được tín hiệu).

- Nếu nút nhấn thường đóng STOP được nhấn trong quá trình đông cơ hoạt động thì động cơ sẽ dừng lai.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4. 36. Hình ảnh kết nối thiết bị



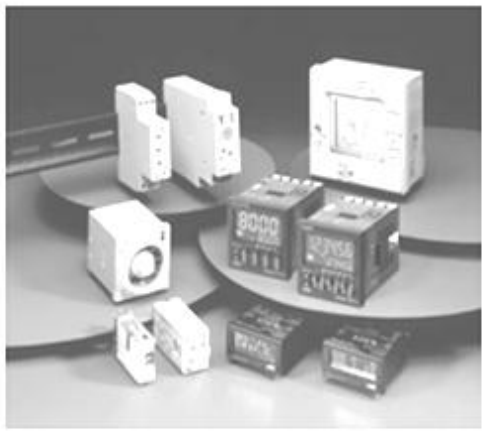
Hình 4.37. Chương trình điều khiển động cơ bước

4.4. CÁC BỘ ĐẾM LẬP TRÌNH ĐƯỢC

Các bộ đếm có khả năng lập trình được có chức năng giống như các bộ đếm cơ học. Hình 4.49 là hình ảnh của một bộ đếm cơ khí đơn giản. Mỗi khi cần gạt bị tác động, giá trị bộ đếm sẽ tăng thêm 1 giá trị và sau đó cần gạt sẽ tự động trở lại vị trí ban đầu. Bộ đếm có nút nhấn khởi động lại được sử dụng để để đưa giá trị đếm về giá trị 0.

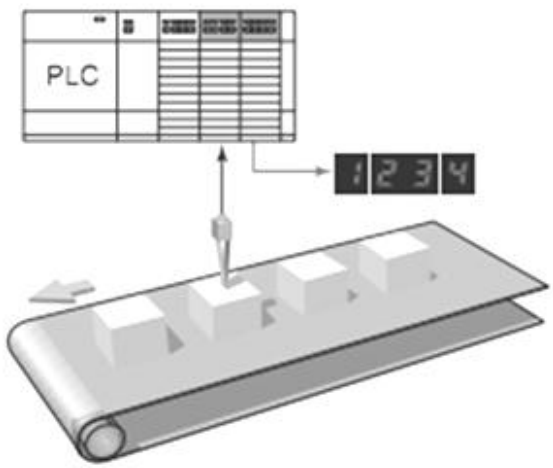
Các bộ đếm điện tử trên Hình 4.50 có khả năng đếm tiến, đếm lùi hoặc cả hai chức năng tiến/lùi. Trong các ứng dụng công nghiệp người ta thường sử dụng các bộ đếm tiến.

Hình 4.38. Bộ đếm cơ khí



Hình 4.39. Bộ đếm điện tử

Trong kỹ thuật lập trình PLC, các nhà sản xuất PLC đã tích hợp các lệnh có khả năng thực hiện chức năng đếm các sự kiện xảy ra. Một trong các ứng dụng phổ biến nhất của bộ đếm là đếm số lượng sản phẩm như được minh họa trong Hình 4.51 dưới đây:

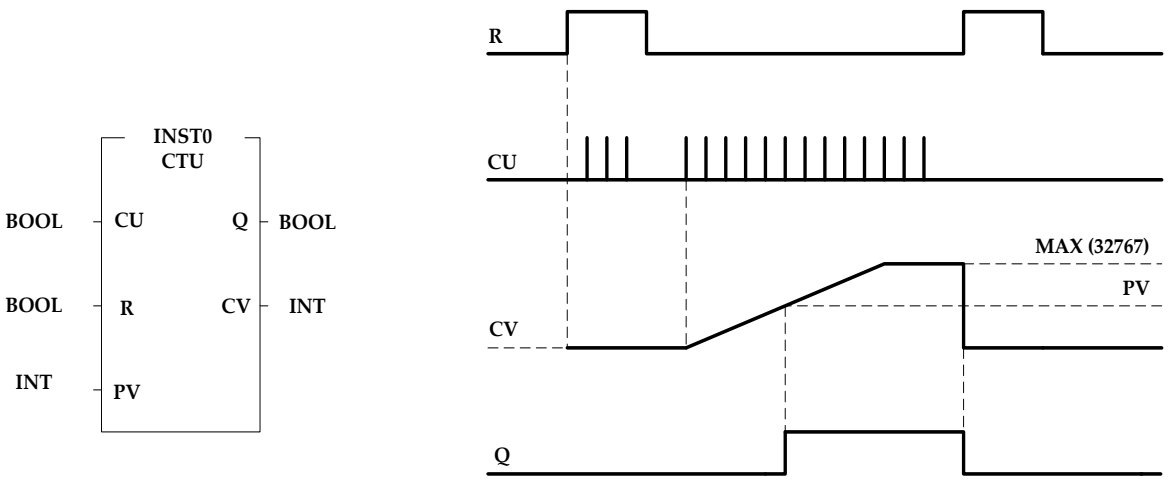


Hình 4.40. Đếm số lượng sản phẩm trên dây truyền sản xuất

Hoạt động của bộ đếm không phụ thuộc vào xung đồng hồ nội của PLC mà phụ thuộc vào sự kiện được đếm. Các bộ đếm phải có đầu vào để đặt giá trị khởi tạo cho bộ đếm. Khi được khởi động lại, giá trị của bộ tiến sẽ được xóa về 0, còn bộ đếm lùi được đưa về giá trị đặt trước. Giá trị đếm hiện tại của bộ đếm tiến hoặc lùi sẽ tăng hoặc giảm đi 1 giá trị mỗi khi có một tín hiệu chuyển từ trạng thái OFF sang ON đặt tới đầu vào. Giá trị đặt trước của bộ đếm có thể là cố định hoặc là kết quả của một đoạn chương trình nào đó.

4.4.1. Bộ đếm tiến

Mỗi khi đầu vào CU chuyển trạng thái từ OFF sang ON, giá trị hiện tại CV của bộ đếm sẽ tăng thêm 1 giá trị (chỉ tăng khi giá tri CV nhỏ hơn 32767). Khi giá trị CV bằng hoặc lớn hơn giá trị đặt trước PV thì đầu ra sẽ có mức logic là TRUE. Giá trị hiện tại CV của bộ đếm sẽ được khởi động lại (đưa về giá trị 0) khi đầu vào R được đưa lên mức 1. Vì vậy, bộ đếm tiến có thể được sử dụng để đếm các sự kiện và sau đó kích hoạt một quá trình nào đó khi mà giá trị đếm hiện tại CV đạt tới giá trị đặt trước PV.



Hình 4.41. Bộ đếm tiến và giản đồ xung

Ví dụ 22:

Hình 4.53 là chương trình ví dụ sử dụng bộ đếm tiến. Ứng dụng được thiết kế để điều khiển bật đèn màu đỏ và tắt đèn màu xanh sau khi giá trị của bộ đếm bằng 10. Quá trình hoạt động của hệ thống như sau:

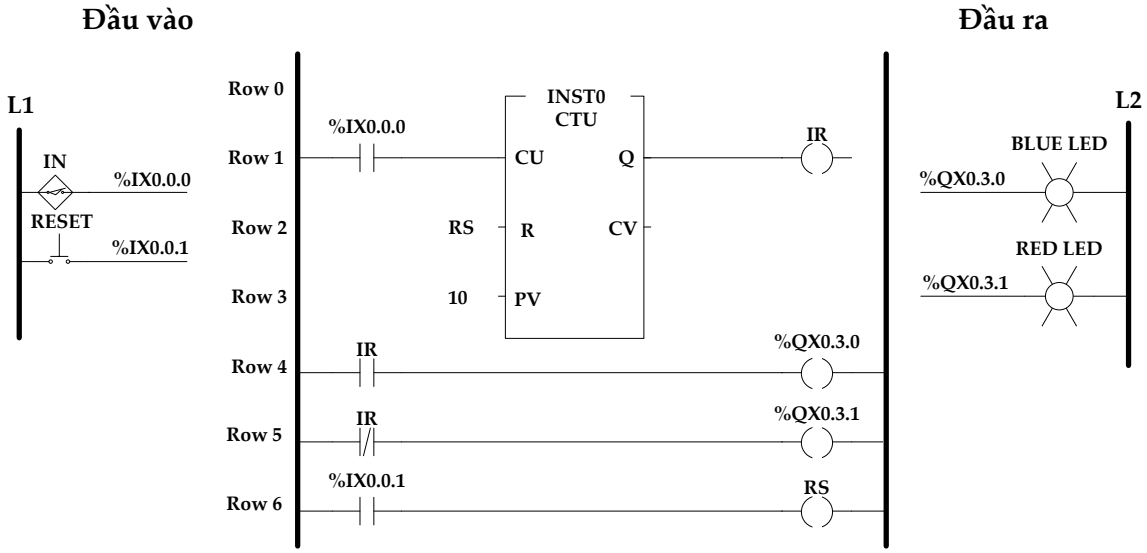
- Khi bắt đầu hoạt động, trạng thái đầu ra của bộ đếm là FALSE, đèn xanh sáng, đèn đỏ tắt.

- Mỗi khi đầu vào %IX0.0.0 được nhấn sẽ tạo ra một xung có trạng thái chuyển từ OFF sang ON, giá trị bộ đếm CT1 tăng lên 1 đơn vị.

- Khi giá trị đếm được CV bằng giá trị đặt trước PV, trạng thái đầu ra Q của bộ đếm là ON, trạng thái của IR cũng là ON. Tại bậc 5 tiếp điểm thường mở IR tiếp điện, đèn đỏ sáng. Tại bậc 6, tiếp điểm thường đóng IR lúc này hở mạch, đèn xanh tắt.

- Khi tiếp điểm %IX0.0.1 tiếp điện, trạng thái đầu ra RS là ON sẽ xóa giá trị CV của bộ đếm về 0. Để quá trình được thực hiện lại thì đầu ra RS phải được chuyển về trạng thái OFF.

Chương trình và sơ đồ kết nối các đầu vào/ra:



Hình 4.42. Chương trình sử dụng bộ đếm tiến

Ví dụ 23:

Hình 4.54 là chương trình điều khiển một hệ thống đóng gói sản phẩm có sử dụng kết hợp 3 bộ đếm tiến. Hoạt động của chương trình có thể được tóm tắt như sau:

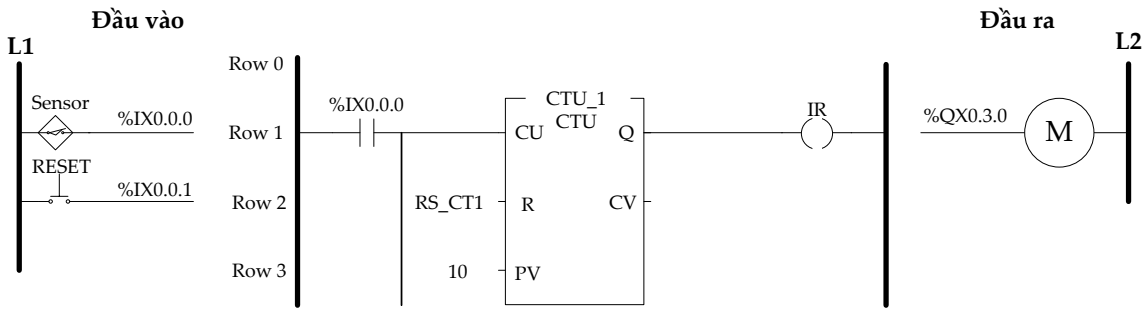
- Bộ đếm tiến CTU_1 có giá trị đặt trước là 10, được sử dụng để đếm số sản phẩm đóng gói vào mỗi thùng (10 sản phẩm).

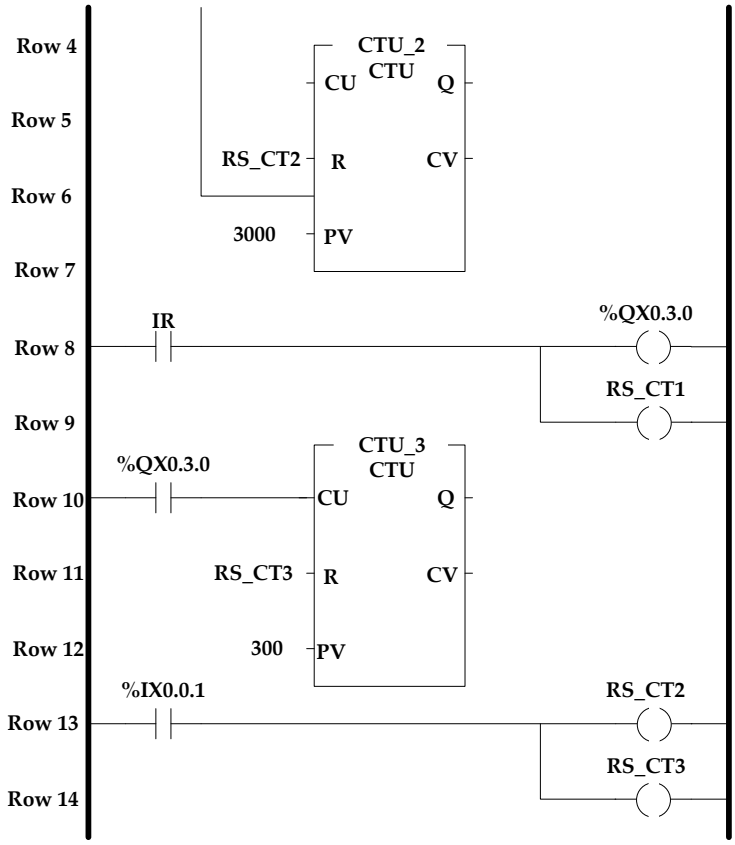
- Bộ đếm tiến CTU_2 có giá trị đặt trước là 3000, sử dụng để đếm tổng số sản phẩm đã được đóng gói.

- Bộ đếm tiến CTU_3 có giá trị đặt trước là 300, có vai trò đếm số thùng được đóng gói.

- Nút nhấn được kết nối tới đầu vào %IX0.0.1 được sử dụng để khởi động lại các bộ đếm CTU_2 và CTU_3.

Chương trình và sơ đồ kết nối các đầu vào/ra:

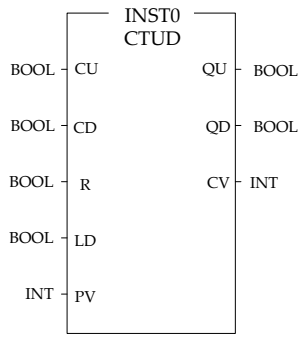




Hình 4.43. Chương trình điều khiển hệ thống đóng gói sản phẩm

4.4.2. Bộ đếm tiến – lùi

Nguyên lý hoạt động của bộ đếm lùi tương tự như bộ đếm tiến, giá trị đếm của nó sẽ giảm đi 1 mỗi khi xảy ra một sự kiện đếm. Tuy nhiên, trong các ứng dụng thực tế chúng ta thường sử dụng bộ đếm tiến/lùi CTUD:



- Giá trị hiện thời (CV) của bộ đếm sẽ tăng hoặc giảm mỗi khi nhận được tín hiệu sườn trước tương ứng ở đầu vào CU hoặc CD. Giá trị hiện thời CV có giá trị từ -32768 tới 32767.

- Giá trị đặt trước PV được sao chép vào CV (PV=CV) khi đầu vào LD có mức logic là TRUE.

- Khi đầu vào R có mức logic là TRUE thì giá trị CV sẽ được xóa về 0.

- Đầu ra QU sẽ có mức logic là TRUE nếu giá trị CV lớn hơn hoặc bằng PV và đầu ra QD có mức logic là TRUE khi CV nhỏ hơn hoặc bằng 0.

- Chú ý rằng thứ tự ưu tiên cho các tín hiệu đầu vào là R > LD > CU > CD.

Ví dụ 24:

Hình 4.55 là một chương trình sử dụng bộ đếm CTUD để điều khiển một hệ thống kiểm soát lượng xe vào/ra bãi đậu xe. Nguyên lý hoạt động của hệ thống như sau:

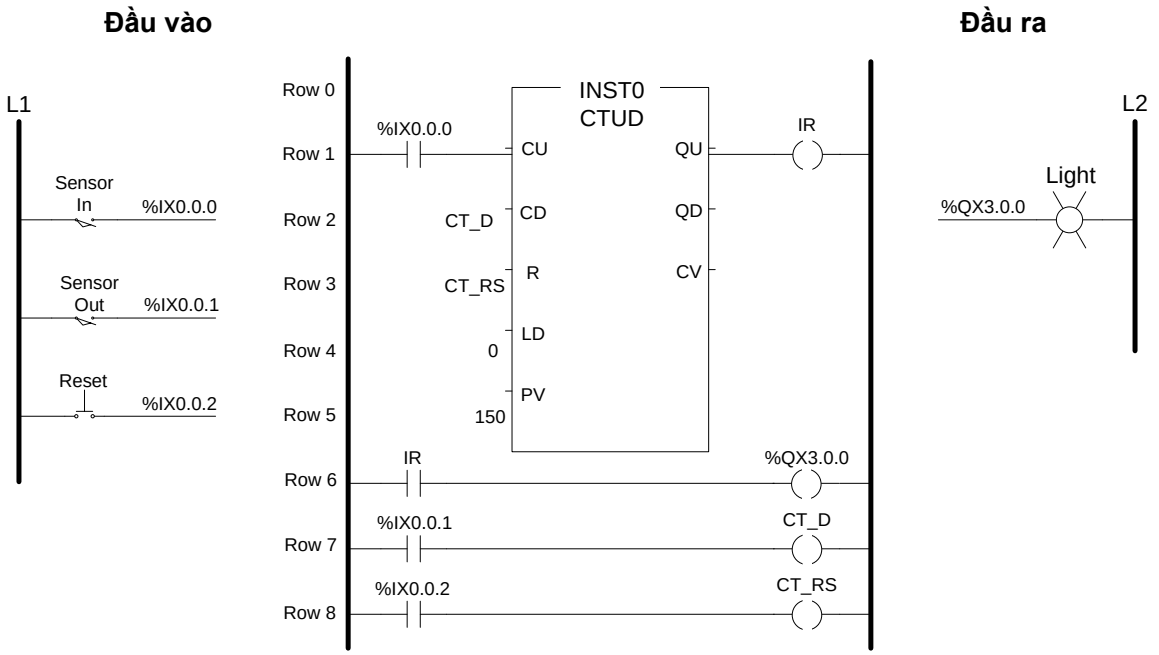
- Khi phát hiện có xe vào bến, đầu vào CU của bộ đếm được kích hoạt, giá trị của bộ đếm được tăng lên 1 giá trị.

- Khi phát hiện có xe rời bến, đầu vào CD của bộ đếm được kích hoạt, giá trị của bộ đếm được giảm đi 1 giá trị.

- Bất cứ khi nào trong bãi có 150 xe thì đầu ra chuyển sang trạng thái ON và đèn sáng báo hiệu bãi đậu xe đã đầy.

- Hệ thống có nút nhấn để khởi động lại giá trị bộ đếm.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.44. Chương trình kiểm soát số lượng ôtô trong gara

Bộ đếm có đếm chính xác hay không là phụ thuộc vào thời gian quét của chương trình. Để bộ đếm có thể hoạt động chính xác, xung tín hiệu đầu vào phải có thời gian ít nhất là bằng một chu kỳ quét. Nếu xung đầu vào ngắn hơn một chu kỳ quét, giá trị đếm có thể bị sai. Khi xảy ra trường hợp này, chúng ta cần phải sử dụng bộ đếm có tốc độ cao hơn hoặc sử dụng các mô-đun vào/ra được thiết kế cho các ứng dụng có tốc độ cao.

4.4.3. Kết hợp các bộ đếm

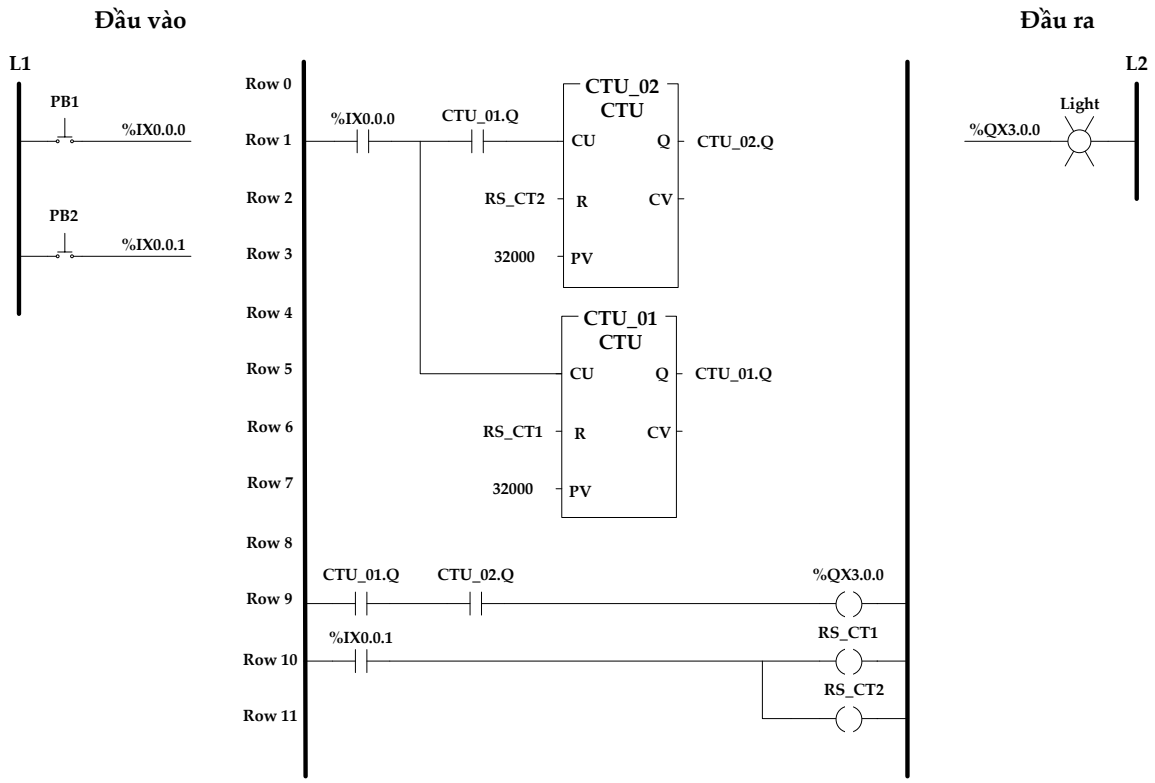
Trong thực tế có những hệ thống yêu cầu đếm các sự kiện với số lần xảy ra lớn hơn giá trị cho phép tối đa của bộ đếm. Để giải quyết vấn đề này, chúng ta có thể sử dụng kết hợp các bộ đếm với nhau để có thể đếm với giá trị lớn hơn.

Ví dụ 25:

Hình 4.56 là một ví dụ minh hoạ cho vấn đề này. Nguyên tắc hoạt động của chương trình như sau:

- Nút nhấn PB1 được sử dụng để tạo tín hiệu đầu vào cho bộ đếm CTU_01.
- Nút nhấn PB1 kết hợp với đầu ra của bộ đếm CTU_01 để tạo tín hiệu đầu vào cho bộ đếm CTU_02.
- Khi cả hai bộ đếm đều đạt đến giá trị đặt trước, đèn sẽ được bật sáng.
- Nút nhấn PB2 được sử dụng để khởi động lại các bộ đếm.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.45. Kết hợp các bộ đếm theo kiểu nối tiếp

Chương trình trên là một ứng dụng kết hợp hai bộ đếm theo kiểu nối tiếp. Nghĩa là sau khi giá trị bộ đếm thứ nhất đạt tới giá trị đặt trước thì bộ đếm thứ hai sẽ bắt đầu đếm. Như vậy, giá trị sau khi cả hai bộ đếm kết thúc quá trình đếm sẽ

bằng tổng giá trị đặt trước của hai bộ đếm. Để có thể đếm với giá trị lớn hơn, chúng ta có thể sử dụng phương pháp đếm theo vòng lặp (tương tự hai vòng for lồng vào nhau). Sau quá trình đếm kết thúc, hệ thống sẽ đếm được một giá trị bằng tích giá trị đặt trước của các bộ đếm.

Ví dụ 26:

Trên Hình 4.57 là một đoạn chương trình sử dụng phương pháp này. Hoạt động của chương trình có thể được tóm tắt như sau:

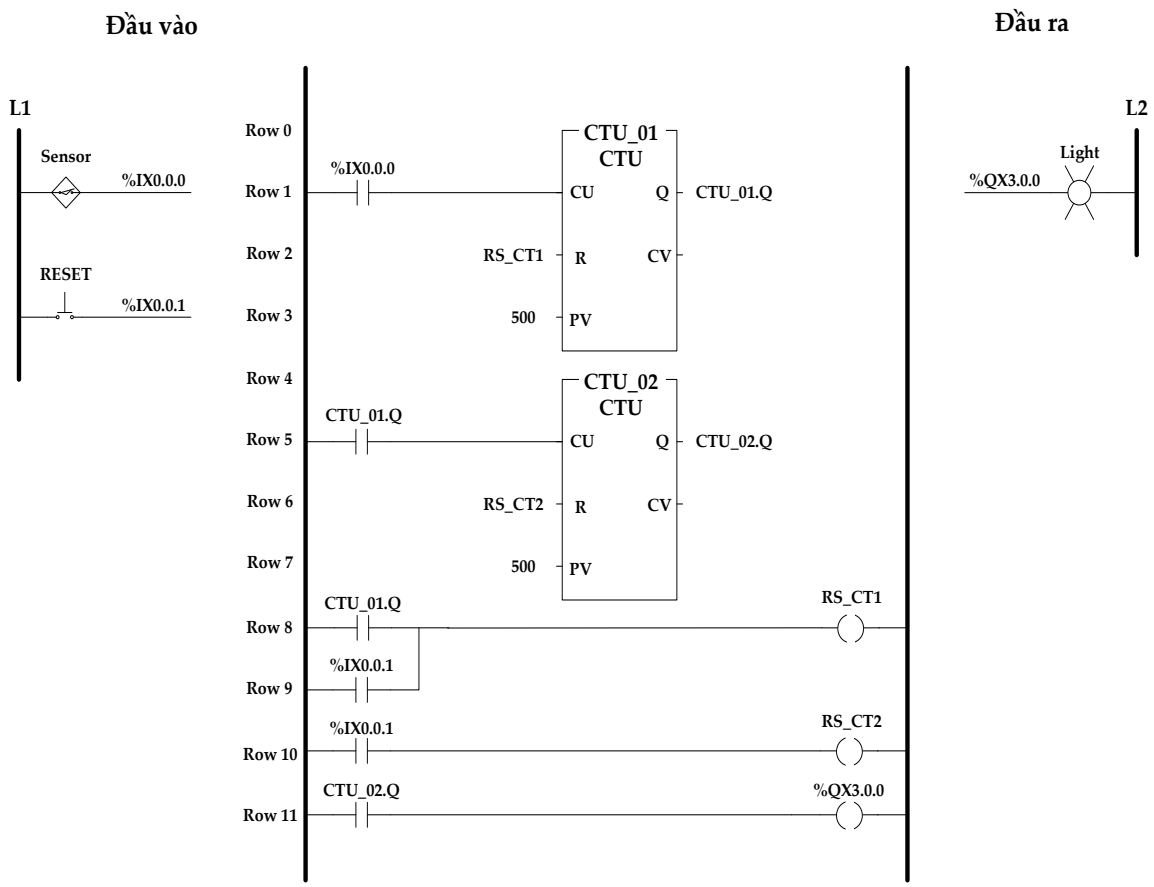
- Cả hai bộ đếm được đặt giá trị là 500.

- Bất cứ khi nào giá trị bộ đếm CTU_01 đạt tới giá trị đặt trước nó sẽ được khởi động lại và giá trị bộ đếm CTU_02 được tăng lên 1 giá trị.

- Khi giá trị của bộ đếm thứ hai đạt tới giá trị đặt trước thì đèn sẽ sáng. Như vậy, đèn sẽ sáng khi hệ thống đếm được 500 x 500 hay 250.000 sự kiện.

- Cả 2 bộ đếm được khởi động lại khi nút nhấn RESET được nhấn.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.46. Kết hợp các bộ đếm theo kiểu vòng lặp

4.4.4. Kết hợp bộ đếm với bộ định thời

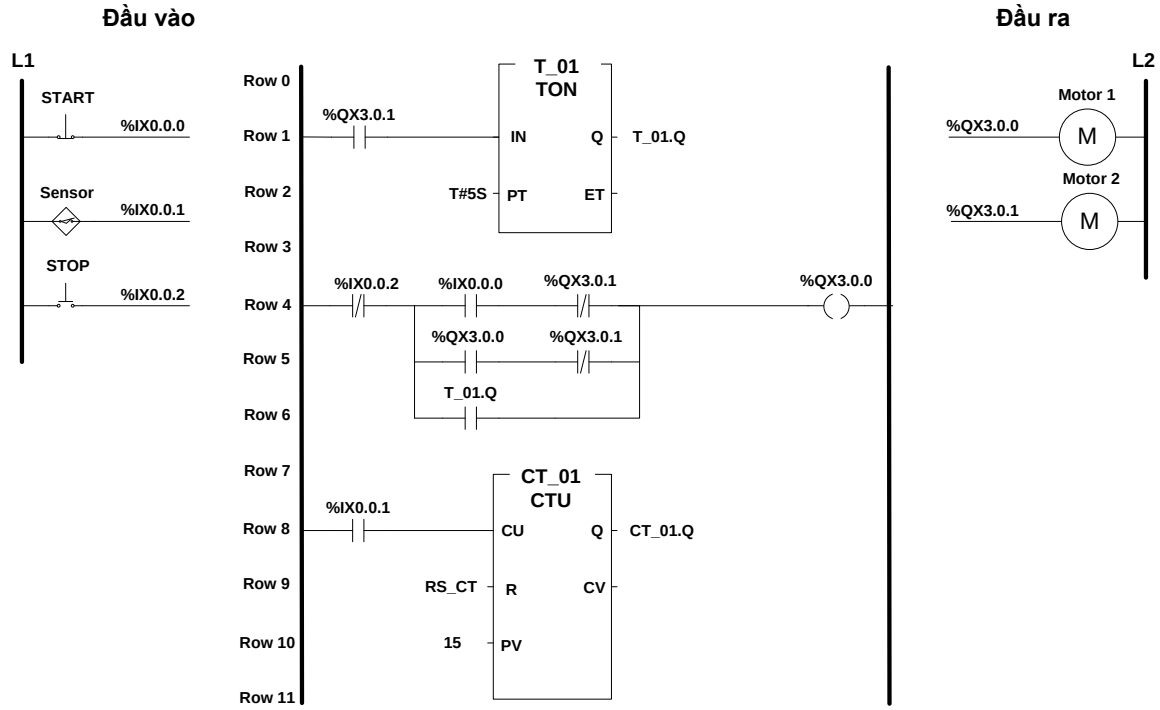
Trong quá trình điều khiển, nhiều ứng dụng đòi hỏi phải có sự kết hợp giữa các bộ định thời và bộ đếm.

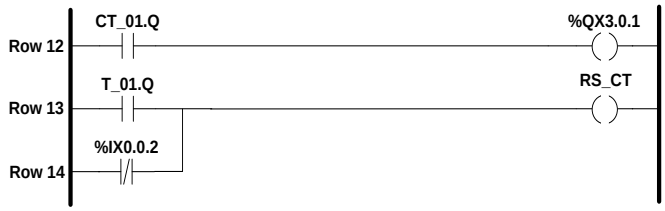
Ví dụ 27:

Hình 4.58 là ứng dụng được thiết kế để điều khiển một hệ thống xếp chồng các tấm kim loại. Trong hệ thống, băng tải M1 được sử dụng để xếp các tấm kim loại lên băng tải M2. Cảm biến quang cung cấp tín hiệu đầu vào cho bộ đếm mỗi khi có tấm kim loại chuyển từ M1 sang M2. Chương trình hoạt động có thể mô tả như sau:

- Khi nút khởi động START được nhấn, băng tải M1 bắt đầu hoạt động.
- Sau khi 15 tấm kim loại được xếp chồng lên nhau thì băng tải M2 bắt đầu hoạt động.
- Băng tải M2 sẽ ngừng hoạt động sau 5s và quá trình được lặp lại.
- Khi bộ định thời đạt đến giá trị đặt trước (5s) sẽ tạo ra ra tín hiệu để khởi động lại chính nó, bộ đếm và khởi động lại băng tải M1.
- Nút nhấn STOP có thể được dùng để khởi động lại hệ thống.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.47. Chương trình điều khiển hệ thống xếp sản phẩm

Ví dụ 28:

Chúng ta có thể kết hợp giữa bộ đếm và bộ định thời để tạo ra một hệ thống định thời với giá trị lớn hơn. Ví dụ, chúng ta cần bật sáng đèn sau khi nút nhấn S1 được nhấn 1000000s, chúng ta có thể làm như sau:

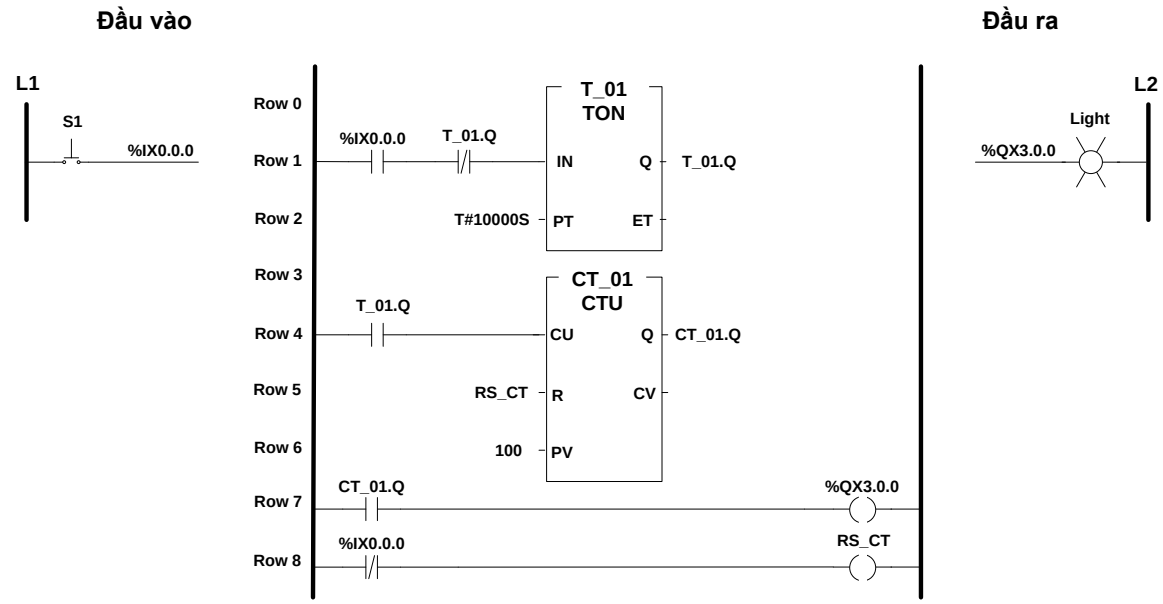
- Giá trị đặt trước của bộ định thời là 10000s và bộ đếm là 100.

- Mỗi khi giá trị bộ định thời đạt tới giá trị đặt trước 10000s nó sẽ được khởi động lại và giá trị của bộ đếm sẽ tăng thêm 1 giá trị.

- Khi giá trị của bộ đếm đạt tới giá trị 100, đèn sẽ được bật sáng.

- Như vậy, đèn sẽ được bật sáng sau 10000 x 100 hay 1000000s sau khi tiếp điểm S1 tiếp điện.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.48. Chương trình kết hợp bộ định thời và bộ đếm để tạo thời gian định thời lớn

Ví dụ 29:

Hình 4.60 là một ứng dụng sử dụng kết hợp bộ đếm và bộ định thời để điều khiển động cơ. Nguyên lý hoạt động của chương trình như sau:

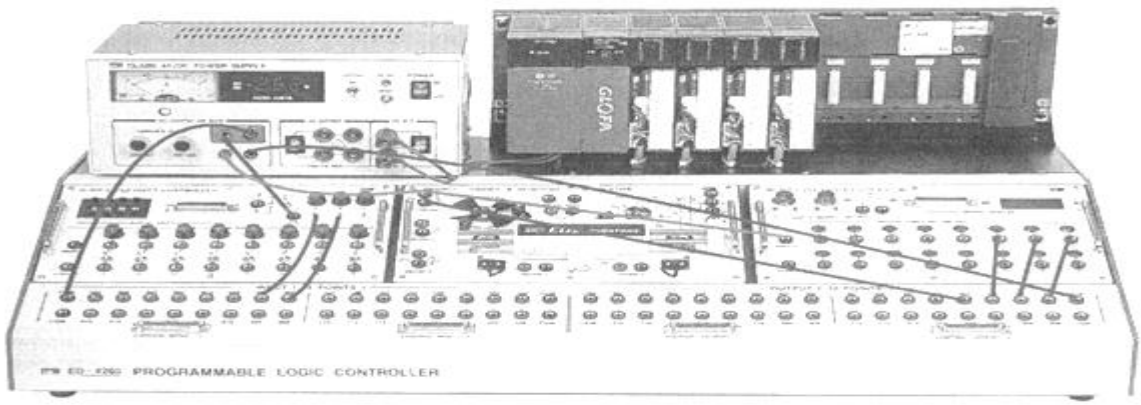
- Khi nút nhấn khởi động thường mở START được nhấn, đèn chỉ thị START LAMP sáng, đồng thời giá trị bộ đếm tăng lên 1 giá trị sau mỗi giây nhờ câu lệnh _T1S.

- Khi giá trị bộ đếm đạt tới giá trị đặt trước là 5, trạng thái đầu ra Q là ON đồng nghĩa với việc động cơ được khởi động và bộ định thời với giá trị đặt trước 5s cũng bắt đầu đếm, đèn chỉ thị OPERATION LAMP cũng sáng.

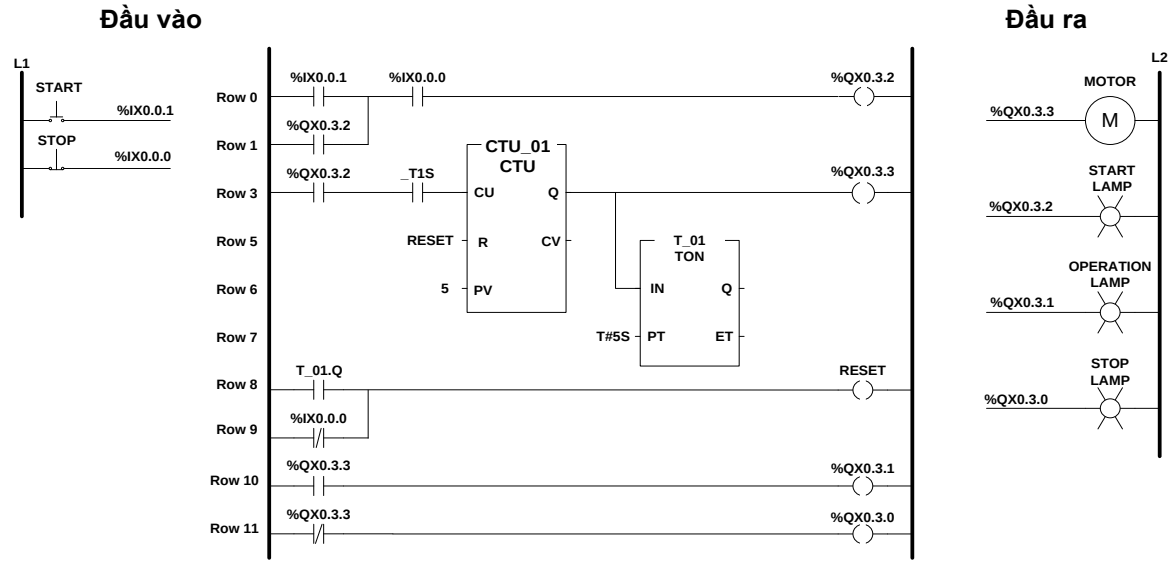
- Sau 5s, trạng thái đầu ra Q của bộ định thời là ON sẽ khởi động lại bộ đếm, động cơ dừng, đèn STOP LAMP sáng, đèn OPERATION LAMP tắt và quá trình được lặp lại.

- Trong quá trình hoạt động nếu nút nhấn thường đóng STOP được nhấn mọi hoạt động bị dừng và đèn STOP LAMP sẽ sáng.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.49. Hình ảnh kết nối thực tế



Hình 4.50. Chương trình kết hợp bộ đếm và bộ định thời điều khiển động cơ

Ví dụ 30:

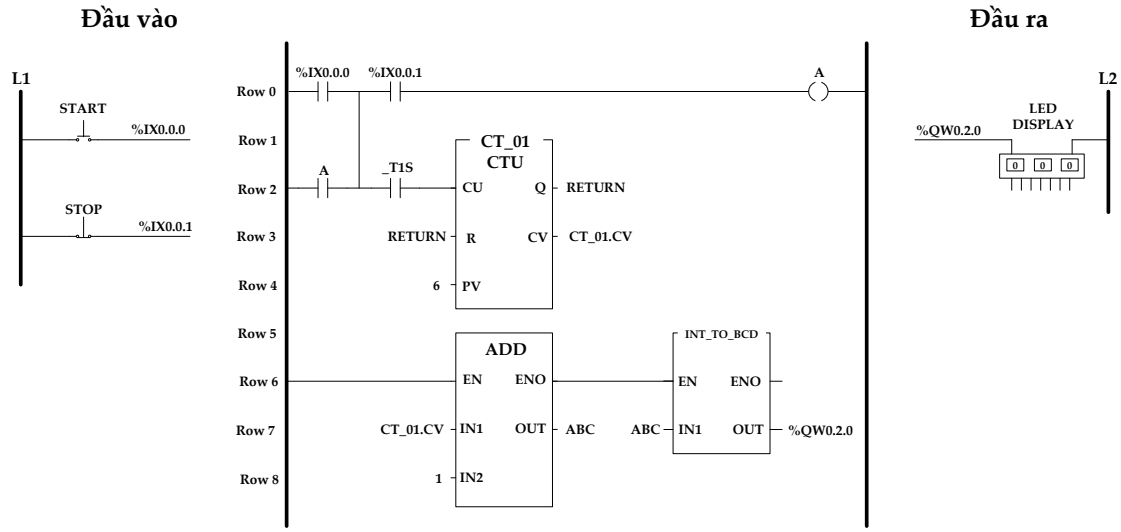
Hình 4.62 là chương trình ứng dụng hiển thị giá trị bộ đếm lên LED hiển thị. Hoạt động của chương trình như sau:

- Khi nút nhấn thường mở START được nhấn, giá trị hiện tại của bộ đếm tăng lên 1 giá trị sau mỗi giây và được hiển thị lên LED hiển thị.

- Khi giá trị đếm CV bằng với giá trị đặt trước PV thì bộ đếm được khởi động lại.

- Nút nhấn thường đóng STOP được sử dụng để khởi động lại bộ đếm.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.51. Chương trình hiển thị giá trị đếm

4.5. CÁC LỆNH ĐIỀU KHIỂN CHƯƠNG TRÌNH

Các câu lệnh điều khiển chương trình được sử dụng để thay đổi quá trình quét thông thường của chương trình. Sử dụng các câu lệnh điều khiển chương trình có thể rút ngắn thời gian thực hiện chương trình.

4.5.1. Lệnh nhảy

Trong kỹ thuật lập trình PLC, đôi khi chúng ta cần phải chuyển qua thực hiện một đoạn chương trình khác khi thỏa mãn một điều kiện nào đó. Lệnh JUMP có thể được sử dụng cho mục đích này. Lệnh JUMP thường được sử dụng để nhảy tới một vị trí nào đó trong chương trình mà người lập trình mong muốn.

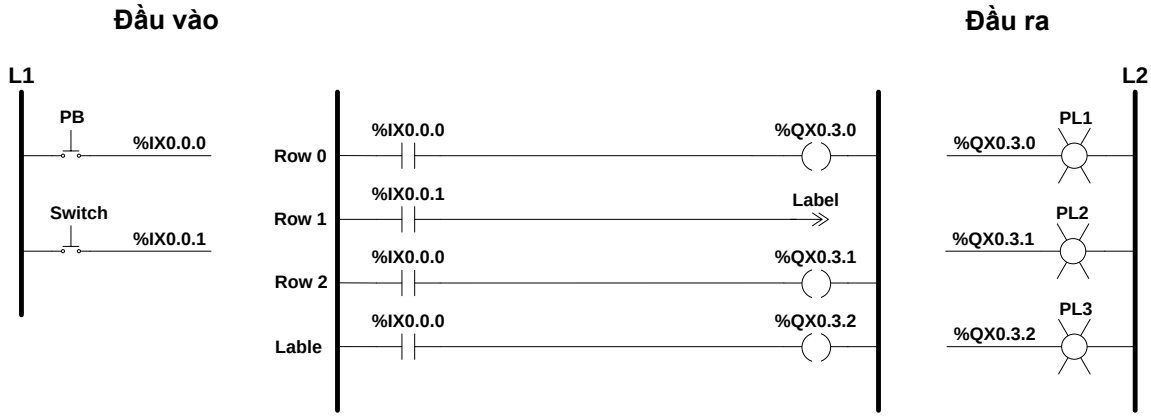
Ví dụ 31:

Chương trình trên Hình 4.68 là một ví dụ sử dụng câu lệnh JUMP. Khi cần nhảy đến vị trí nào chúng ta phải gán nhãn cho vị trí đó. Nguyên tắc hoạt động của chương trình được tóm tắt như sau:

- Khi Switch hở mạch, câu lệnh JUMP không có tác dụng, nút nhấn PB tiếp điện, cả 3 đèn báo sẽ sáng.

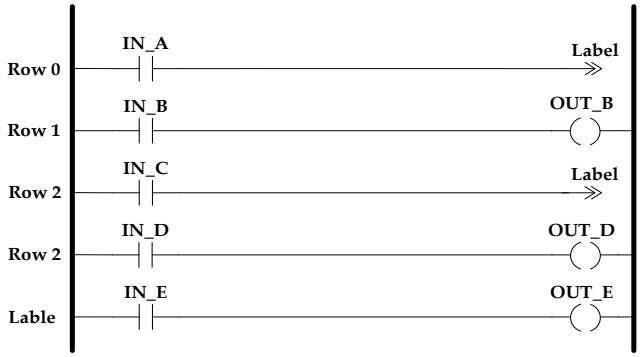
- Khi Switch tiếp điện, câu lệnh JUMP tác dụng, nút nhấn PB được nhấn, chỉ có đèn PL1, PL3 sáng. Các lệnh trên bậc thứ 3 được bỏ qua.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.52. Nguyên lý hoạt động của câu lệnh JUMP

Trong chương trình có thể sử dụng nhiều lệnh JUMP để nhảy tới cùng một nhãn như trong Hình 4.69. Trong chương trình này, có 2 lệnh JUMP cùng nhảy tới cùng một nhãn Lable. Lệnh JUMP nào được thực hiện là phụ thuộc vào các tiếp điểm đầu vào tương ứng. Lệnh JUMP có thể được sử dụng để nhảy ngược lại. Tuy nhiên không nên thực hiện quá nhiều lần. Chú ý khi quá trình quét xảy ra quá lâu trong vòng lặp. Bộ xử lý trung tâm có bộ định thời giám sát (watchdog timer) được sử dụng để thiết lập tổng thời gian quét của một chương trình. Nếu thời gian quét vượt quá thời gian này, bộ xử lý sẽ đưa ra thông báo lỗi và dừng hoạt động.



Hình 4.53. Chương trình sử dụng nhiều câu lệnh JUMP

Ví dụ 32:

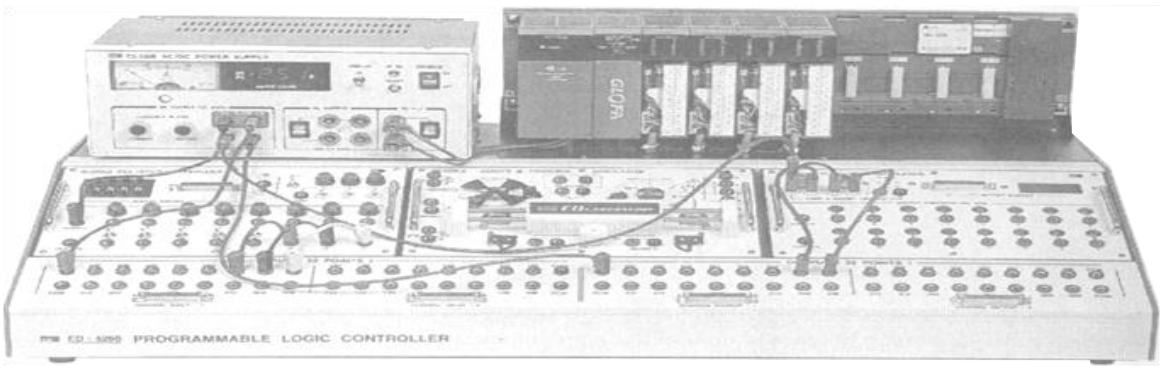
Dưới đây là môt ví dụ được thực hiện với bộ PLC ED-4260. Chương trình có sử dụng lệnh JUMP với nguyên tắc hoạt động như sau:

- Chương trình sử dụng 3 đầu vào nối với các nút nhấn thường mở và 2 đầu ra nối với các đèn chỉ thị.

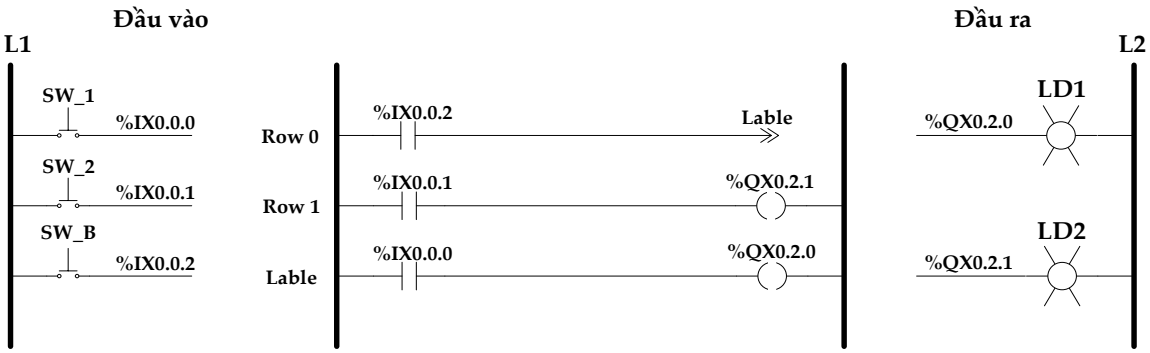
- Khi nút nhấn SW_B chưa được nhấn, các đèn chỉ thị sẽ sáng khi các nút nhấn đầu vào tương ứng được nhấn.

- Khi nút nhấn SW_B được nhấn, đèn chỉ thị LD2 sẽ không sáng cho dù nút nhấn SW_2 được nhấn mà chỉ có LD1 sáng khi nút nhấn SW_1 tiếp điện.

Chương trình và sơ đồ kết nối các đầu vào/ra:



Hình 4.54. Hình ảnh kết nối thực tế



Hình 4.55. Chương trình thực hiện câu lệnh JUMP

4.5.2. Lệnh gọi hàm con

Hàm con là một chương trình nhỏ được viết để thực hiện một nhiệm vụ cụ thể nào đó trong một chương trình lớn. Ưu điểm khi sử dụng chương trình con đó là chúng ta có thể sử dụng nó nhiều lần mà không cần phải viết lại. Chương trình con được viết và được gán với một tên nhãn. Để gọi một chương trình con chúng

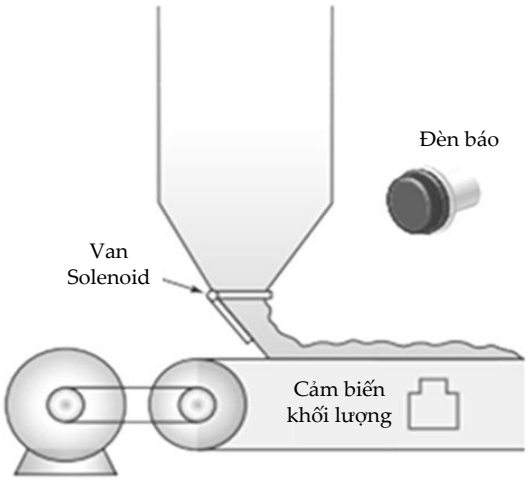
ta có thể sử dụng câu lệnh SCAL (Subroutine Call) kèm theo nhãn của chương trình con.

Ví dụ 33:

Hình 4.72 là một hệ thống băng tải được sử dụng để vận chuyển nguyên liệu, trong hệ thống có sử dụng đèn báo và được thiết kế để sử dụng hàm con. Hoạt động của chương trình có thể mô tả như sau:

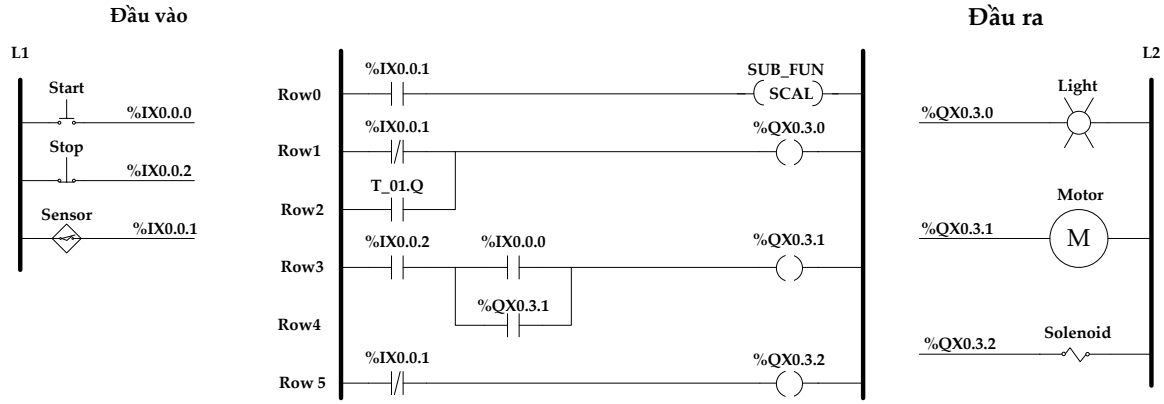
- Nếu khối lượng nguyên liệu trên băng tải lớn hơn giá trị được đặt trước, cảm biến khối lượng sẽ nhận được tín hiệu, van Solenoid đuọc ngắt điện, chương trình sẽ gọi một hàm con định thời và đèn báo PL1 sẽ nhấp nháy với chu kỳ 1s.

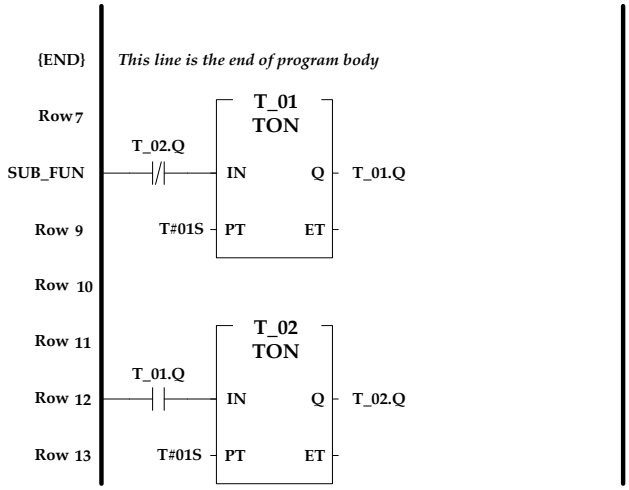
- Khi cảm biến khối lượng không còn nhận được tín hiệu, chương trình sẽ quay lại trạng thái quét thông thường, đèn báo PL1 lúc này sẽ không nhấp nháy nữa mà sẽ trở lại trạng thái sáng bình thường.



Hình 4.56. Hệ thống băng tải vận chuyển nguyên liệu

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.57. Chương trình điều khiển hệ thống băng tải

Ví dụ 34:

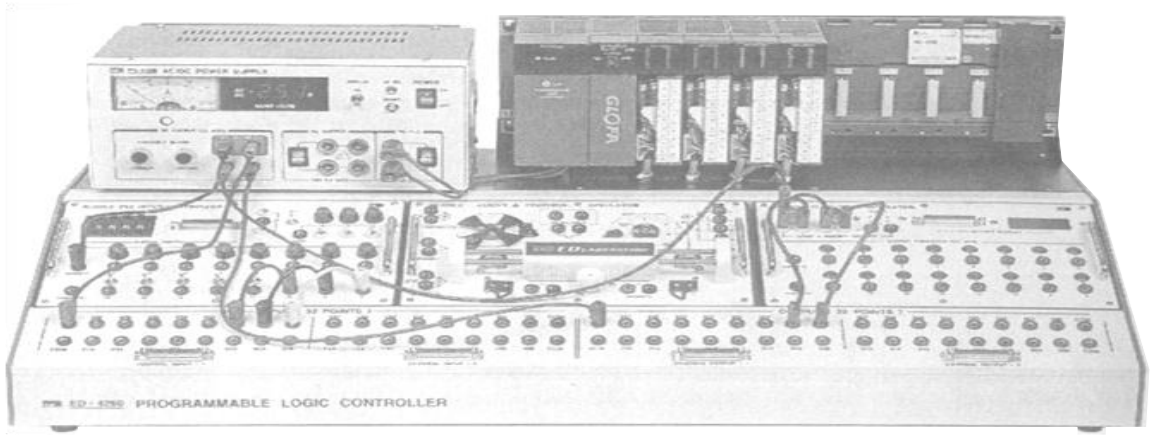
Hình 4.74 là ví dụ có sử dụng chương trình con với nguyên lý hoạt động như sau:

- Nếu nút nhấn MAIN_SW được nhấn, đèn MAIN LAMP sẽ sáng nhờ mạch chốt và đồng thời câu lệnh SCAL cũng được thực hiện. Lúc này nếu nút nhấn AU_SW được nhấn thì đèn AU LAMP sẽ sáng.

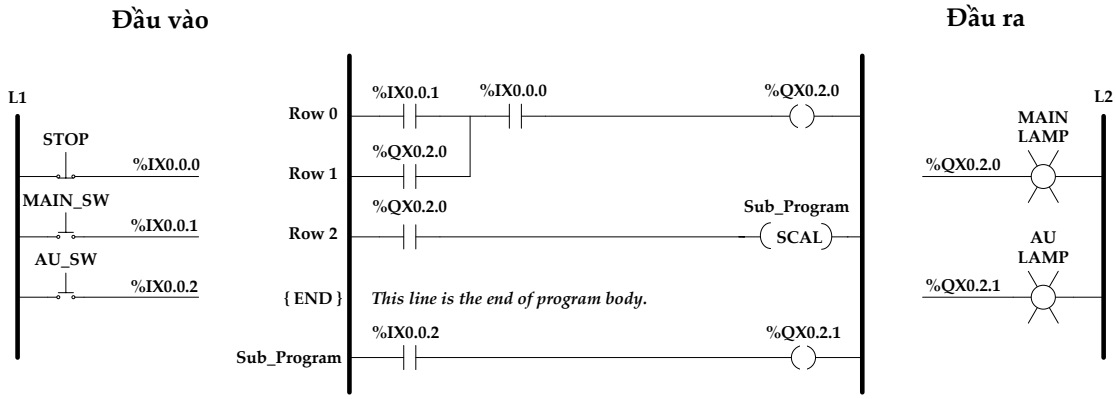
- Nếu nút nhấn MAIN_SW không được nhấn, đèn MAIN LAMP sẽ tắt và đồng thời câu lệnh SCAL cũng không được thực hiện. Lúc này nếu nút nhấn AU_SW được nhấn thì đèn AU LAMP vẫn không sáng.

- Nút nhấn thường đóng STOP được sử dụng để dừng mọi hoạt động của hệ thống.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.58. Hình ảnh kết nối thiết bị



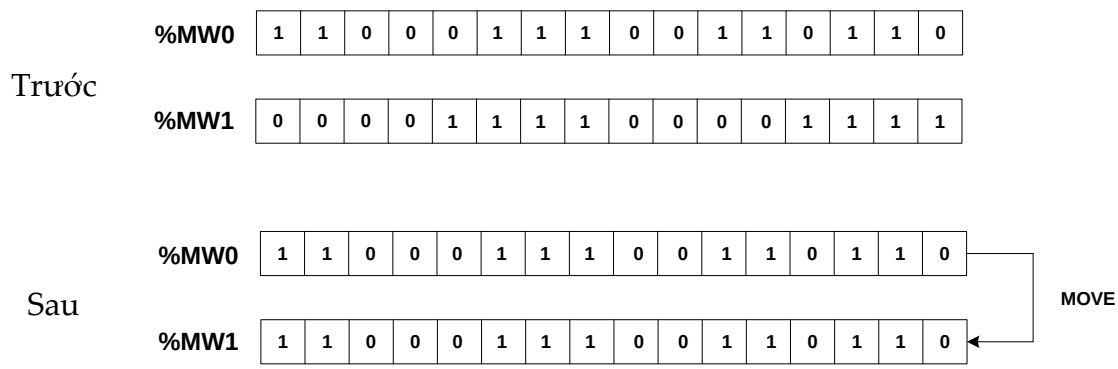
Hình 4.59. Chương trình con

4.6. CÁC LỆNH XỬ LÝ DỮ LIỆU

Xử lý dữ liệu là công việc liên quan đến quá trình sao chép dữ liệu hay các thao tác toán học (tính toán, chuyển đổi, so sánh dữ liệu hay các phép toán logic). Xử lý dữ liệu trên PLC cũng giống như quá trình trình xử lý dữ liệu của máy vi tính. Chúng ta sẽ tìm hiểu hoạt động của các câu lệnh qua các ví dụ cụ thể.

4.6.1. Lệnh sao chép dữ liệu

Câu lệnh sao chép dữ liệu (MOVE) đơn giản là thực hiện chức năng sao chép dữ liệu từ một địa chỉ nguồn (Source) tới một địa chỉ đích (Destination). Hình 4.81 biểu diễn sơ đồ tổ chức dữ liệu ban đầu và sau khi thực hiện câu lệnh MOVE tại 2 địa chỉ %MW0 và %MW1. Dữ liệu tại địa chỉ %MW1 lúc này sẽ được thay thế bởi dữ liệu tại địa chỉ %MW0, còn dữ liệu tại địa chỉ %MW0 giữ nguyên không thay đổi.



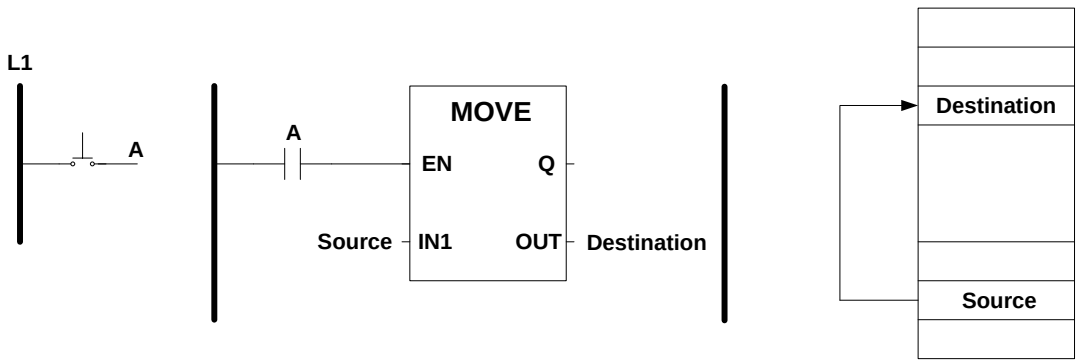
Hình 4.60. Sơ đồ tổ chức dữ liệu khi thực hiện lệnh MOVE

Nguyên tắc hoạt động của câu lệnh MOVE được mô tả trên Hình 4.82. Hoạt động của chương trình có thể được tóm tắt như sau:

- Khi tiếp điểm A được tiếp điện, giá trị tại địa chỉ nguồn (Source) được sao chép tới địa chỉ đích (Destination). Giá trị tại địa chỉ nguồn không bị thay đổi.

- Nguồn và đích phải có cùng loại dữ liệu.

- Đầu ra Q có giá trị logic là TRUE khi câu lệnh được thực hiện thành công.



Hình 4.61. Nguyên tắc hoạt động của lệnh MOVE

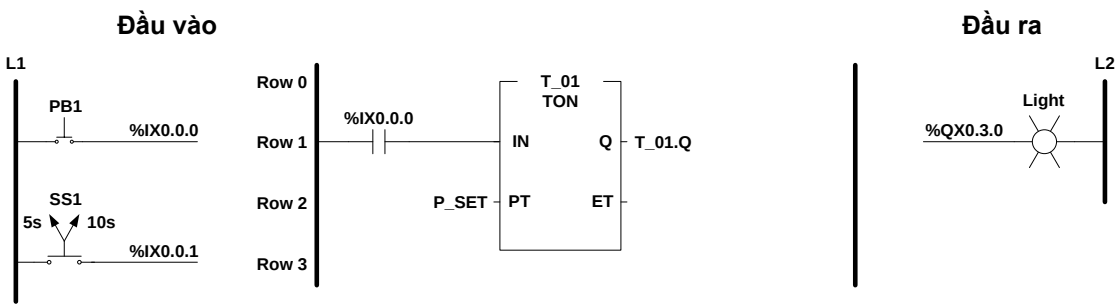
Ví dụ 35:

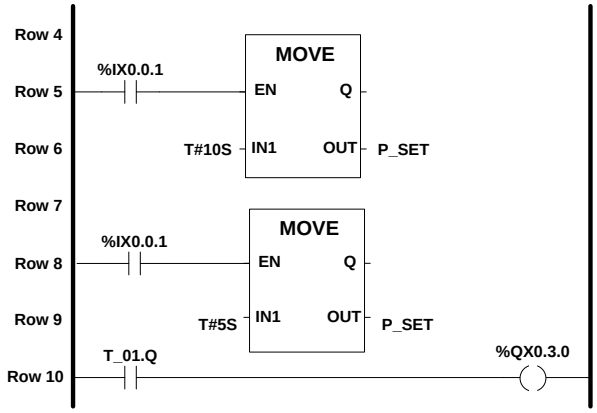
Hình 4.83 là chương trình sử dụng câu lệnh MOVE để tạo ra giá trị đặt trước cho bộ định thời. Nút nhấn hai trạng thái được sử dụng để lựa chọn giá trị định thời cho bộ định thời. Nguyên tắc hoạt động của chương trình như sau:

- Khi nút nhấn SS1 ở vị trí 10s, bậc 5 là liên tục, bậc 9 không liên tục, giá trị T#10S được sao chép sang biến P_SET. Giá trị đặt trước cho bộ định thời T_01 sẽ được thay đổi từ 0 tới 10. Khi nút nhấn PB1 tiếp điện, bộ định thời bắt đầu hoạt động và sau 10s trạng thái đầu ra Q của bộ định thời là ON và đèn báo PL1 trên bậc 12 sẽ sáng.

- Khi nút nhấn SS1 ở vị trí 5s, bậc 9 là liên tục, bậc 5 không liên tục, giá trị T#5S được sao chép sang biến P_SET. Giá trị đặt trước cho bộ định thời T_01 sẽ được thay đổi từ 0 tới 5. Khi nút nhấn PB1 tiếp điện, bộ định thời bắt đầu hoạt động và sau 5s trạng thái đầu ra Q của bộ định thời là ON và đèn báo PL1 trên bậc 12 sẽ sáng.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.62. Chương trình kết hợp lệnh MOVE và bộ định thời

Ví dụ 36:

Chương trình trên Hình 4.84 là một ví dụ sử dụng câu lệnh MOVE để tạo ra giá trị đặt trước cho bộ đếm. Chương trình hoạt động như sau:

- Cảm biến được nối tới đầu vào %IX0.0.0 của PLC để tạo thành đầu vào cho bộ đếm tiến CTU_01 (đếm số lượng sản phẩm trên băng tời).

- Trên băng tời có 3 loại sản phẩm.

- Giá đựng có thể chứa 300 hộp sản phẩm A hoặc 175 hộp sản phẩm B hoặc 50 hộp sản phẩm C.

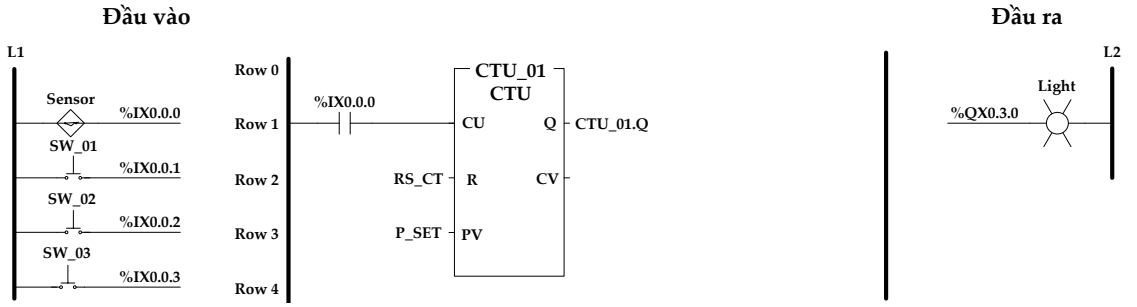
- 03 công tắc nhấn tự nhả SW_01, SW_02, SW_03 được sử dụng để lựa chọn giá trị đặt trước cho bộ đếm phụ thuộc vào dòng sản phẩm được sản xuất là A, B, hoặc C.

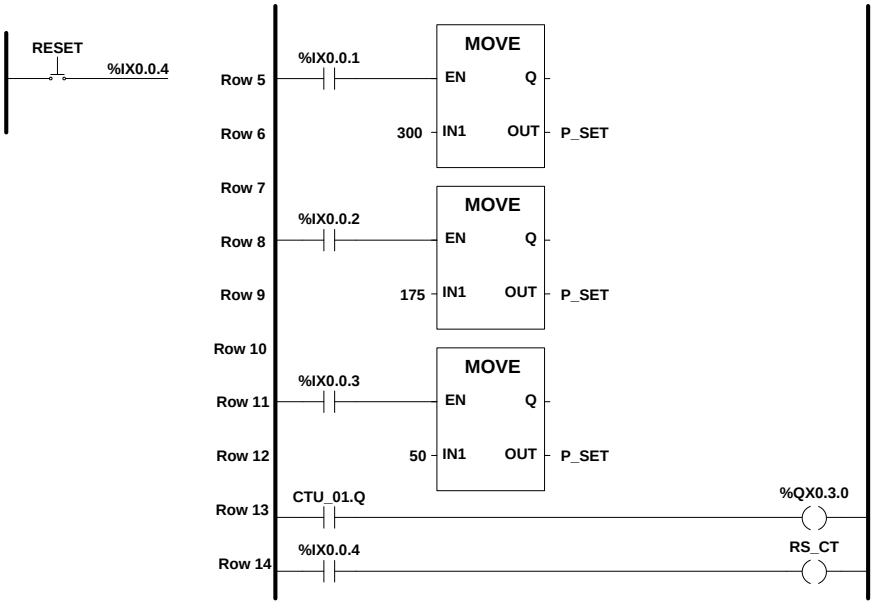
- Nút nhấn khởi động lại RESET được sử dụng để đặt lại giá trị cho bộ đếm.

- Đèn báo sẽ sáng khi giá đựng sản phẩm đã đầy.

- Chương trình được xây dựng để hoạt động ở chế độ chỉ một trong ba công tắc được chọn. Nếu có nhiều hơn một công tắc được chọn thì hệ thống sẽ vận hành với giá trị của khóa sau cùng được chọn.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.63. Chương trình kết hợp lệnh MOVE và bộ đếm

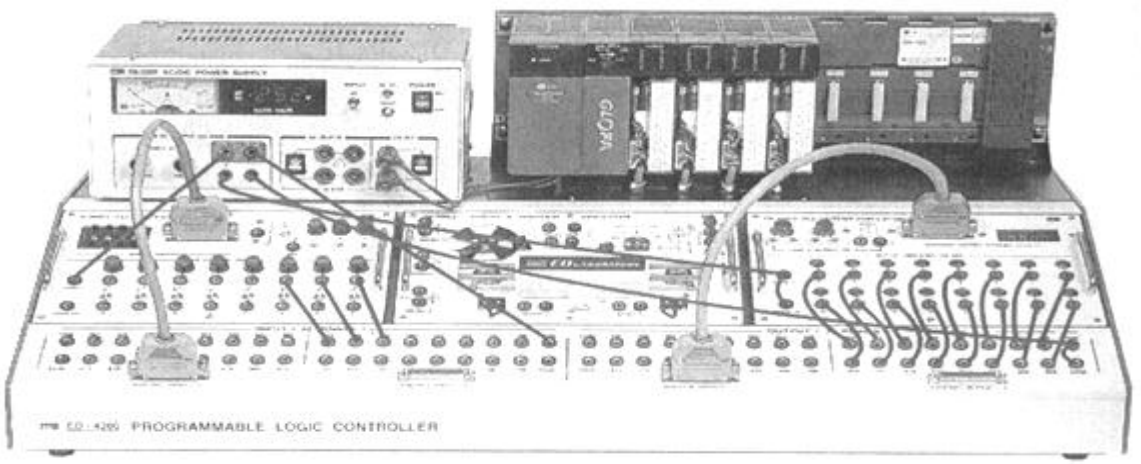
Ví dụ 37:

Hình 4.85 là ví dụ sử dụng câu lệnh MOVE để sao chép dữ liệu được nhập từ các nút nhấn tới các địa chỉ bộ nhớ để hiện thị lên LED hiển thị và làm sáng các đèn báo. Nguyên lý hoạt động của chương trình như sau:

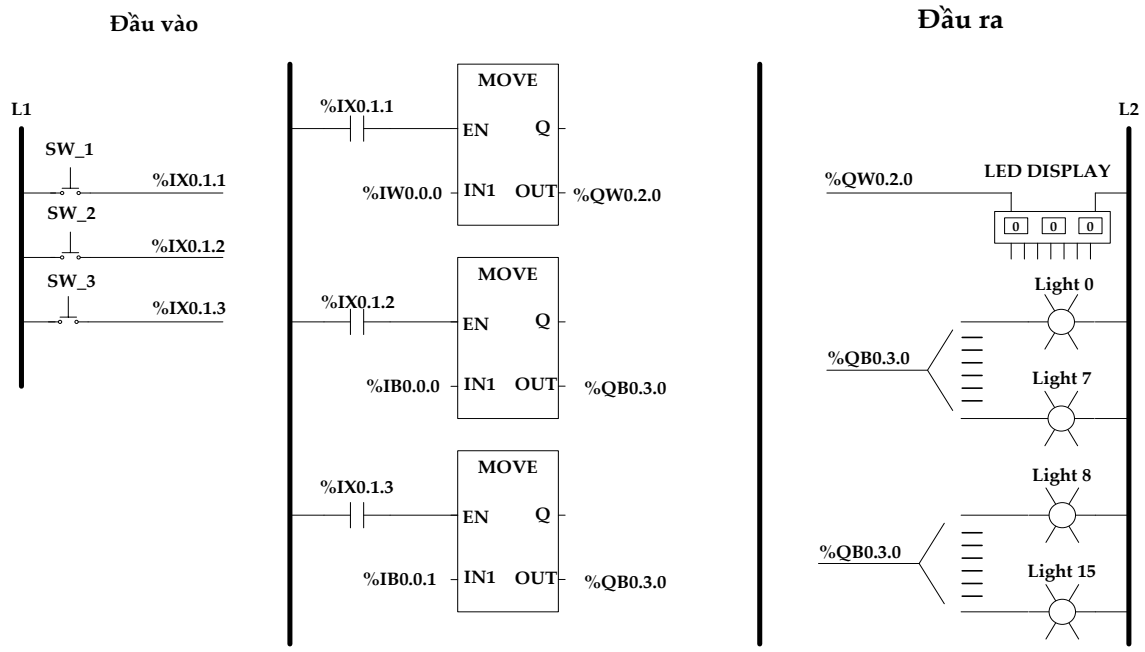
- Đầu vào số được chuyển tới địa chỉ %QX0.2.0 và cho hiển thị lên LED.

- Hệ thống sử dụng nút nhấn để tạo ra các giá trị và được sao chép tới các địa chỉ %QB0.3.0, %QB0.3.1. Sau đó các giá trị này được sử dụng để bật sáng các đèn tương ứng.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.64. Chương trình ví dụ sử dụng câu lệnh MOVE



Hình 4.65. Hình ảnh kết nối

4.6.2. Các câu lệnh so sánh

Chúng ta đã làm quen với định nghĩa so sánh dữ liệu khi sử dụng bộ định thời hay bộ đếm. Với cả hai câu lệnh này, đầu ra sẽ có trạng thái là ON hoặc OFF khi giá trị thu nhận đạt tới với giá trị đặt trước.

Các câu lệnh so sánh được sử dụng để so sánh giá trị số. Các câu lệnh so sánh được tổng hợp trong bảng dưới đây:

Bảng 4.3. Bảng tóm tắt các lệnh so sánh

Lệnh	Ký hiệu	Ứng dụng
So sánh lớn hơn	GT	Kiểm tra xem giá trị thứ nhất có lớn hơn giá trị thứ hai hay không.
So sánh lớn hơn hoặc bằng	GE	Kiểm tra xem giá trị thứ nhất lớn hơn hay bằng giá trị thứ hai.
So sánh nhỏ hơn	LT	Kiểm tra xem giá trị thứ nhất có nhỏ hơn giá trị thứ hai hay không.
So sánh nhỏ hơn hoặc bằng	LE	Kiểm tra xem giá trị thứ nhất nhỏ hơn hay bằng giá trị thứ hai.

So sánh bằng	EQ	Kiểm tra xem hai giá trị có bằng nhau không.
So sánh không bằng	NE	Kiểm tra xem giá trị thứ nhất là không bằng với giá trị thứ hai hay không.

Phép so sánh lớn hơn (GT)

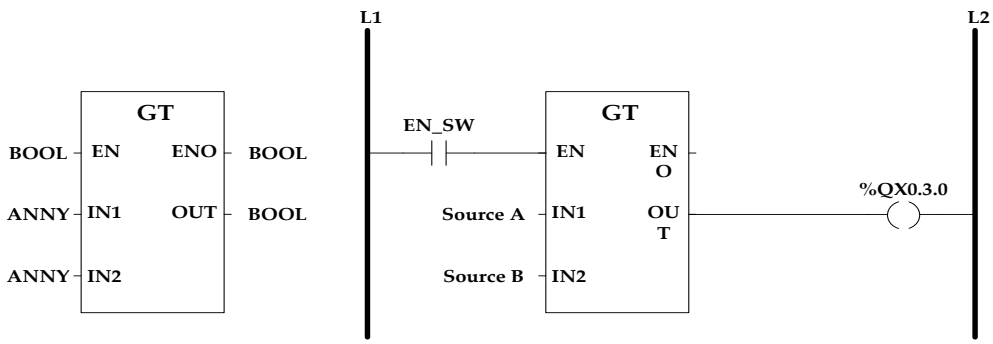
Phép so sánh lớn hơn được sử dụng để so sánh hai hay nhiều (tối đa là 8) giá trị số với nhau. Hình 4.87 là một ví dụ sử dụng câu lệnh GT để so sánh giá trị của hai nguồn là Soure A và Source B. Nguyên lý hoạt động của đoạn chương trình như sau:

- Khi tiếp điểm EN_SW tiếp điện, câu lệnh GT sẽ được thực hiện.

- Kết quả so sánh của câu lệnh GT là TRUE nếu giá trị Source A lớn hơn Source B, hoặc FALSE nếu ngược lại.

- Nếu câu lệnh được thực hiện thành công thì đầu ra ENO có mức logic là TRUE và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào IN có cùng loại dữ liệu.



Hình 4.66. Nguyên lý hoạt động của lệnh so sánh lớn hơn

Phép so sánh lớn hơn hoặc bằng (GE)

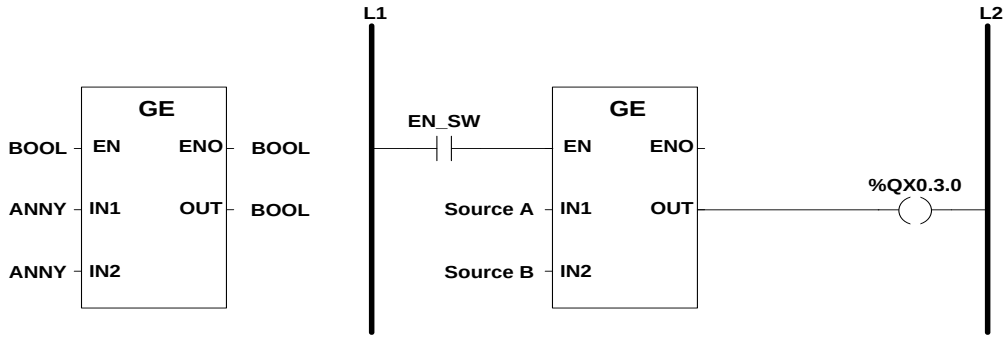
Phép so sánh lớn hơn hoặc bằng được sử dụng để so sánh hai hay nhiều (tối đa là 8) giá trị số với nhau. Để hiểu nguyên tắc hoạt động của câu lệnh GE, chúng ta hãy xem xét đoạn chương trình được cho trong Hình 4.88 dưới đây:

- Khi tiếp điểm EN_SW tiếp điện, câu lệnh GE sẽ được thực hiện.

- Kết quả so sánh của câu lệnh GE là TRUE nếu giá trị Source A lớn hơn hoặc bằng Source B, hoặc FALSE nếu ngược lại.

- Nếu câu lệnh được thực hiện thành công thì đầu ra ENO có mức logic là TRUE và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào IN có cùng loại dữ liệu.



Hình 4.67. Nguyên lý hoạt động của câu lệnh lớn hơn hoặc bằng

Phép so sánh nhỏ hơn (LT)

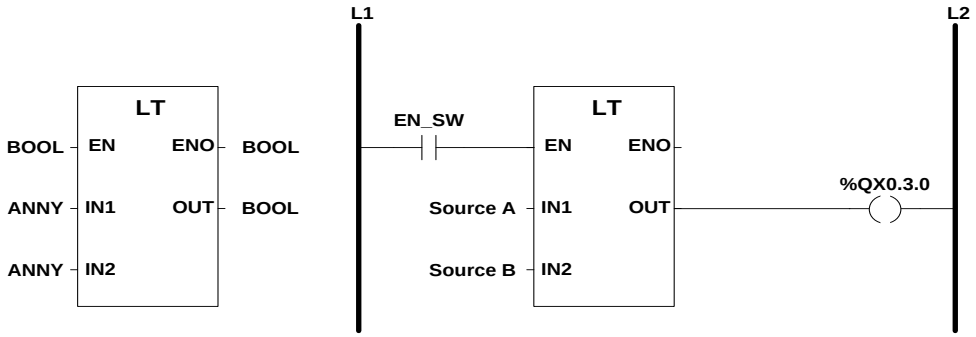
Phép so sánh nhỏ hơn được sử dụng để so sánh hai hoặc nhiều (tối đa là 8) giá trị số với nhau. Chương trình trên Hình 4.89 giải thích nguyên lý hoạt động của câu lệnh so sánh nhỏ hơn với hai đầu vào:

- Khi tiếp điểm EN_SW tiếp điện, câu lệnh LT sẽ được thực hiện.

- Kết quả so sánh của câu lệnh LT là TRUE nếu giá trị Source A nhỏ hơn Source B, hoặc FALSE nếu ngược lại.

- Nếu câu lệnh được thực hiện thành công thì đầu ra ENO có mức logic là TRUE và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào IN có cùng loại dữ liệu.



Hình 4.68. Nguyên lý hoạt động của lệnh so sánh nhỏ hơn

Phép so sánh nhỏ hơn hoặc bằng (LE)

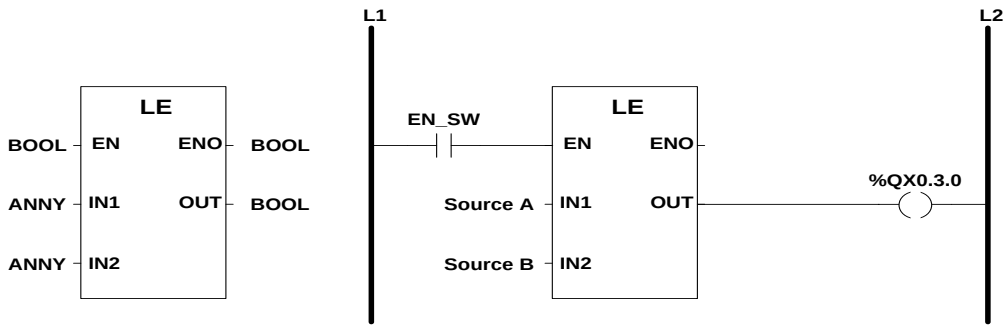
Phép so sánh nhỏ hơn hoặc bằng được sử dụng để so sánh hai hoặc nhiều (tối đa là 8) giá trị số với nhau. Nguyên lý hoạt động của câu lệnh so sánh nhỏ hơn hoặc bằng với hai đầu vào như sau:

- Khi tiếp điểm EN_SW tiếp điện, câu lệnh LE sẽ được thực hiện.

- Kết quả so sánh của câu lệnh LE là TRUE nếu giá trị Source A nhỏ hơn hoặc bằng Source B, hoặc FALSE nếu ngược lại.

- Nếu câu lệnh được thực hiện thành công thì đầu ra ENO có mức logic là TRUE và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào IN có cùng loại dữ liệu.



Hình 4.69. Nguyên lý hoạt động của lệnh so sánh nhỏ hơn hoặc bằng

Câu lệnh so sánh bằng (EQ)

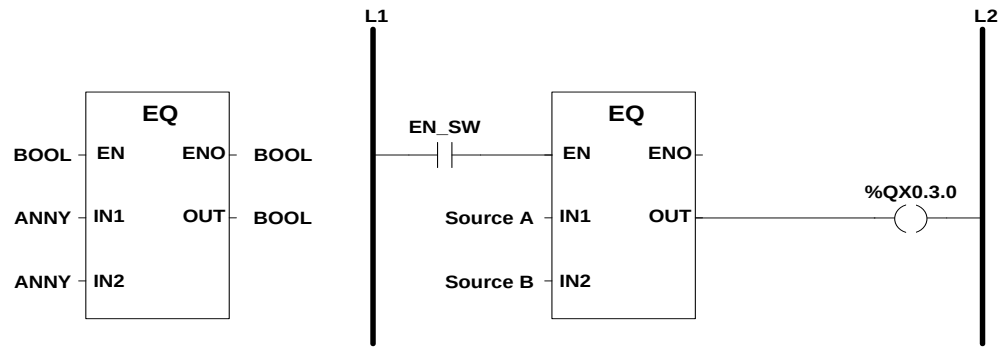
Phép so sánh bằng là được sử dụng để so sánh hai hay nhiều (tối đa là 8) giá trị số với nhau. Hình 4.91 là một ví dụ sử dụng câu lệnh EQ. Hoạt động của chương trình như sau:

- Khi tiếp điểm EN_SW tiếp điện, câu lệnh EQ sẽ được thực hiện.

- Kết quả so sánh của câu lệnh EQ là TRUE nếu giá trị Source A bằng Source B, hoặc FALSE nếu ngược lại.

- Nếu câu lệnh được thực hiện thành công thì đầu ra ENO có mức logic là TRUE và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào IN có cùng loại dữ liệu.



Hình 4.70. Nguyên lý hoạt động của lệnh so sánh bằng

Câu lệnh so sánh không bằng (NE)

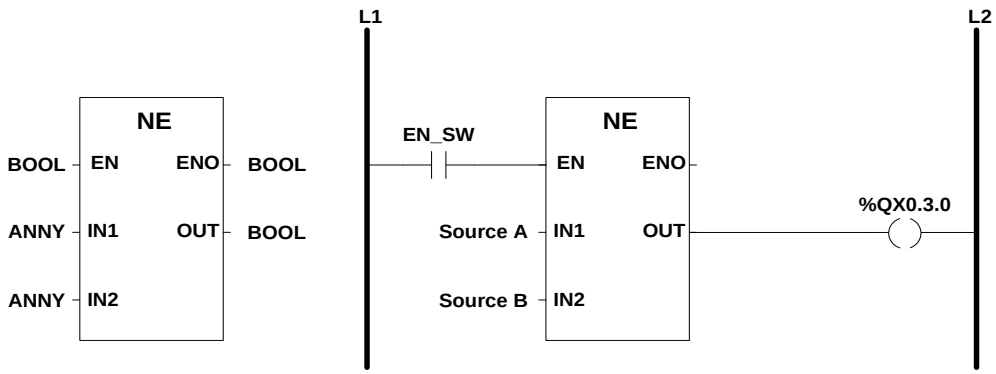
Phép so sánh không bằng được sử dụng để so sánh giá trị của 2 giá trị số với nhau. Hình 4.92 là một ví dụ dùng để giải thích nguyên tắc hoạt động của câu lệnh NE:

- Khi tiếp điểm EN_SW tiếp điện, câu lệnh NE sẽ được thực hiện.

- Kết quả so sánh của câu lệnh NE là TRUE nếu giá trị Source A không bằng Source B, hoặc FALSE nếu ngược lại.

- Nếu câu lệnh được thực hiện thành công thì đầu ra ENO có mức logic là TRUE và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào IN có cùng loại dữ liệu.



Hình 4.71. Nguyên lý hoạt động của lệnh so sánh không bằng

Ví dụ 38:

Chúng hãy thực hiện một chương trình kết hợp bộ định thời và các câu lệnh so sánh dữ liệu như Hình 4.93. Hoạt động của chương trình như sau:

- Nút nhấn thường đóng STOP dùng để dừng mọi hoạt động của hệ thống.

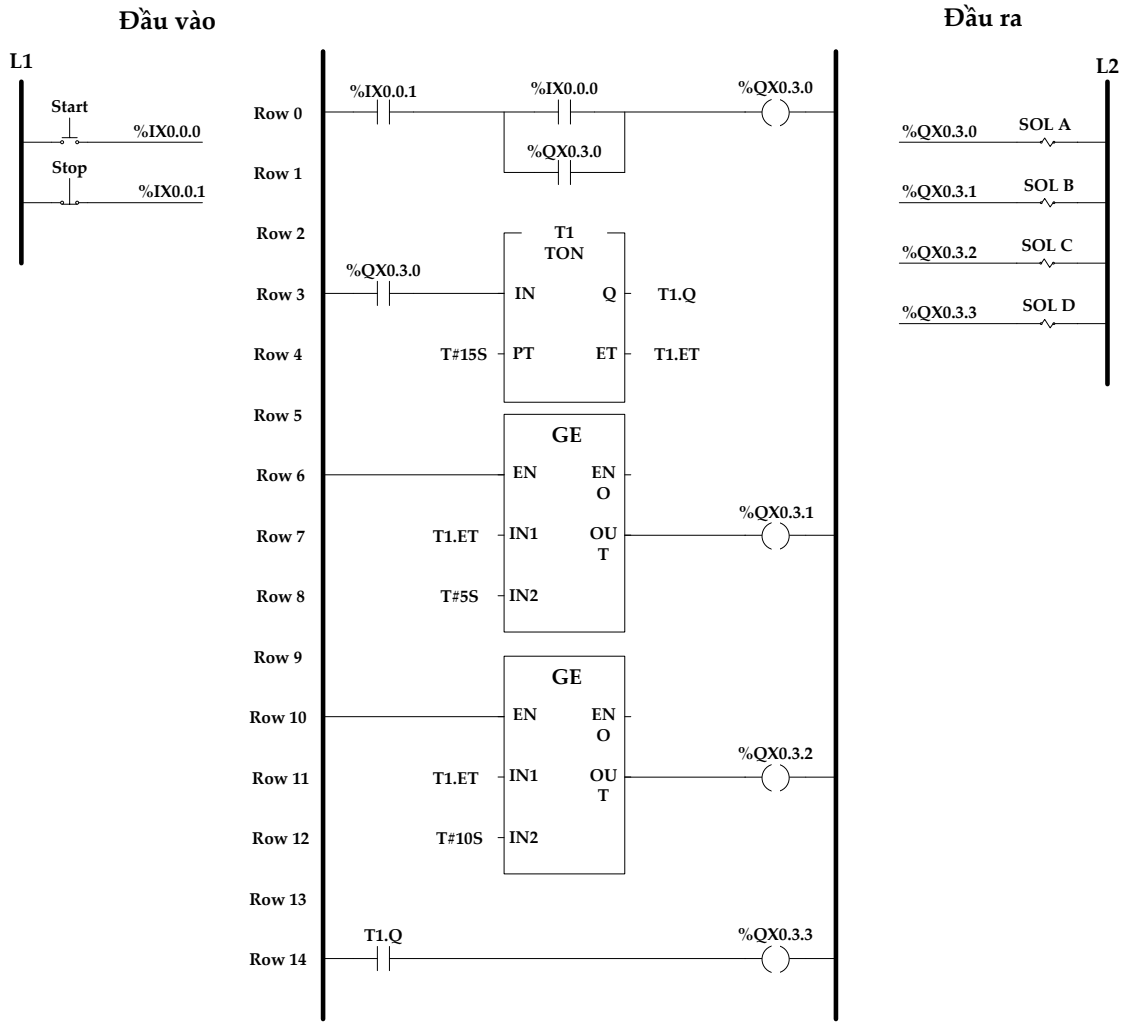
- Khi nút nhấn khởi động thường mở START được nhấn, van Solenoid A được cấp điện, bộ định thời T1 với giá trị đặt trước là 15s được kích hoạt hoạt và bắt đầu đếm.

- Van SOL D sẽ được cấp điện sau khi giá trị hiện tại của bộ định thời đạt tới giá trị đặt trước là 15s.

- Van SOL B sẽ được cấp điện khi giá trị hiện tại của bộ định thời lớn hơn hoặc bằng 5s.

- Van SOL C sẽ được cấp điện khi giá trị hiện tại của bộ định thời lớn hơn hoặc bằng 10s.

Chương trình và sơ đồ kết nối các đầu vào/ra:



Hình 4.72. Chương trình kết hợp bộ định thời và các bộ so sánh

Ví dụ 39:

Hình 4.94 là một chương trình sử dụng kết hợp giữa bộ đếm tiến và câu lệnh so sánh nhỏ hơn. Hoạt động của chương trình như sau:

- Giá trị của bộ đếm tiến CTU1 sẽ tăng 1 giá trị mỗi khi đầu vào nhận được một xung sườn trước.

- Câu lệnh so sánh nhỏ hơn với giá trị đầu vào IN1 được lấy từ giá trị đếm được của bộ đếm và IN2 là một số không đổi 20.

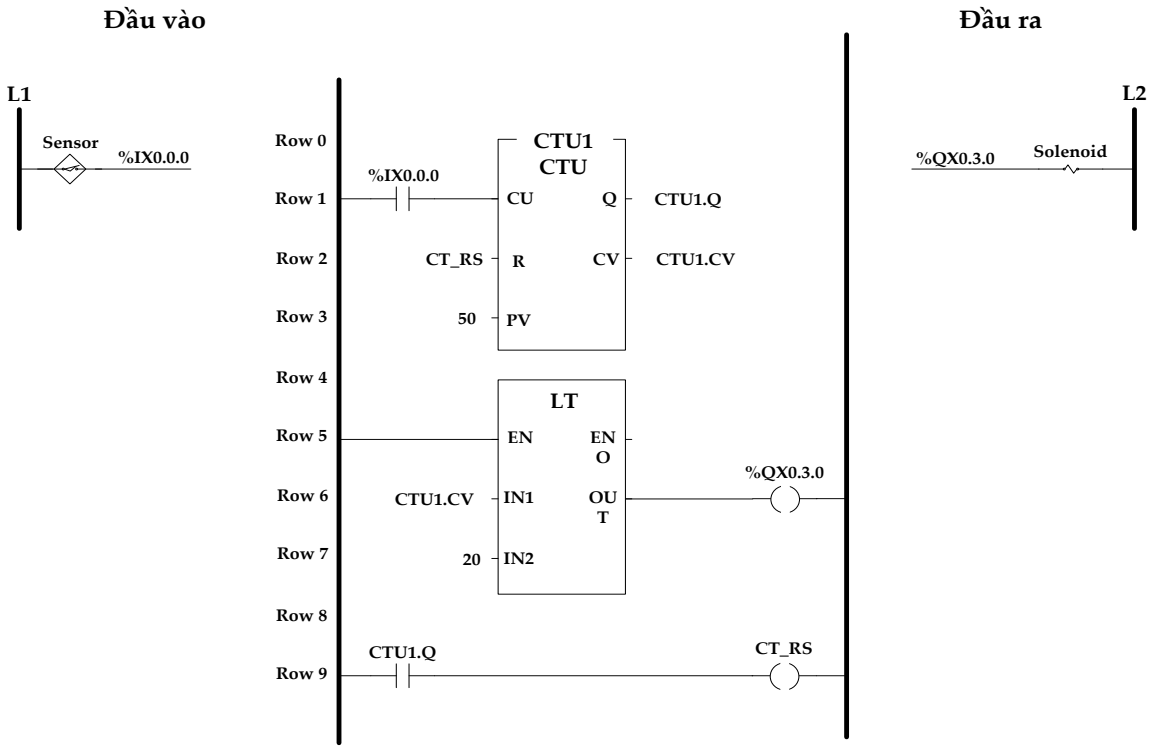
- Kết quả của phép so sánh sẽ là TRUE nếu giá trị của bộ đếm nhỏ hơn 20.

- Đầu ra %QX0.3.0 sẽ được cấp điện khi giá trị hiện tại của bộ đếm nằm trong khoảng từ 0 cho tới 19.

- Khi giá trị của bộ đếm đạt tới giá trị 20, kết quả của phép so sánh sẽ là FALSE, đầu ra %QX0.3.0 bị ngắt điện.

- Khi giá trị bộ đếm đạt tới giá trị đặt trước 50, bộ đếm sẽ được khởi động lại tới giá trị 0.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



Hình 4.73. Chương trình kết hợp bộ đếm và các lệnh so sánh

4.7. CÁC LỆNH TOÁN HỌC

Hầu hết các PLC có khả năng thực hiện các phép tính toán học. Các câu lệnh này được sử dụng để thực hiện các phép cộng, trừ, nhân và chia các giá trị số với nhau. PLC có khả năng thực hiện nhiều phép toán số học trong một chu kỳ quét. Phần này chúng ta sẽ làm quen với các câu lệnh toán học cơ bản và các ứng dụng của nó.

4.7.1. Lệnh ADD

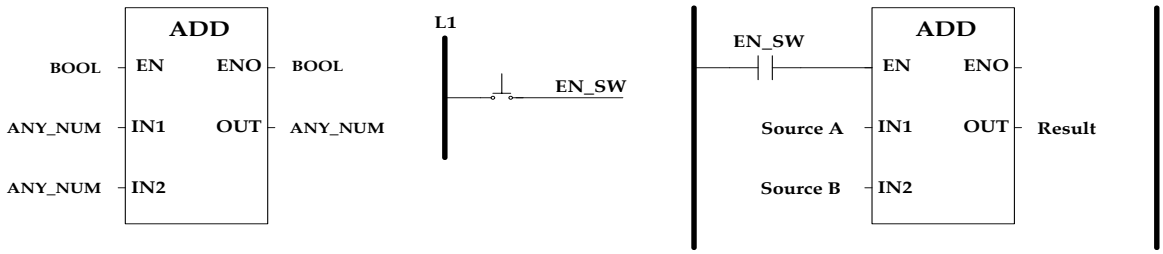
Lệnh ADD (phép cộng) có chức năng thực hiện phép tính cộng giữa 2 hay nhiều (tối đa là 8) giá trị số với nhau. Hình 4.103 là một ví dụ đơn giản được sử dụng để giải thích hoạt động của câu lệnh ADD:

- Câu lệnh được thực hiện khi khóa EN_SW tiếp điện.

- Giá trị Source A được cộng với giá trị Source B, kết quả được lưu vào biến Result.

- Đầu ra ENO có giá trị là TRUE nếu câu lệnh được thực hiện thành công và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào và ra có cùng kiểu dữ liệu số.



Hình 4.74. Nguyên lý hoạt động của lệnh ADD

Ví dụ 40:

Với chương trình được trình bày trên Hình 4.104, câu lệnh ADD có thể được sử dụng để cộng các giá trị đếm được của 2 bộ đếm tiến với nhau. Khi kết quả phép cộng lớn hơn hoặc bằng 350, đèn báo sẽ sáng. Nguyên tắc hoạt động của chương trình như sau:

- Giá trị đầu vào IN1 và IN 2 tương ứng được lấy từ giá trị đếm được của hai bộ đếm.

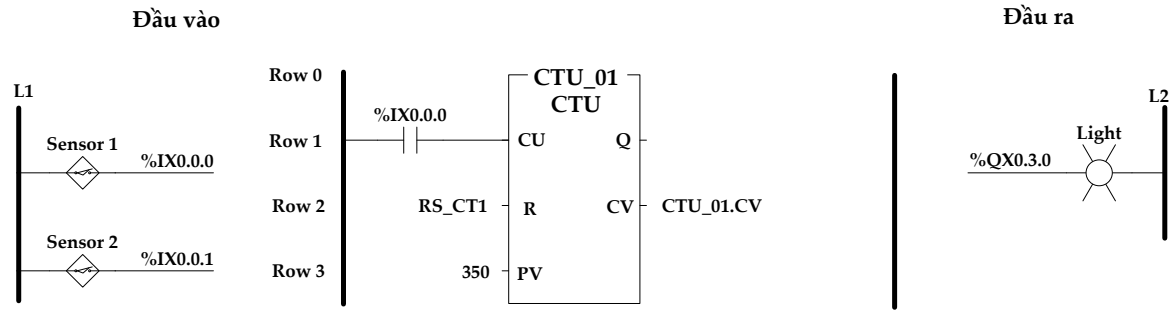
- Giá trị IN1 được cộng với giá trị IN2, kết quả được lưu vào biến RESULT.

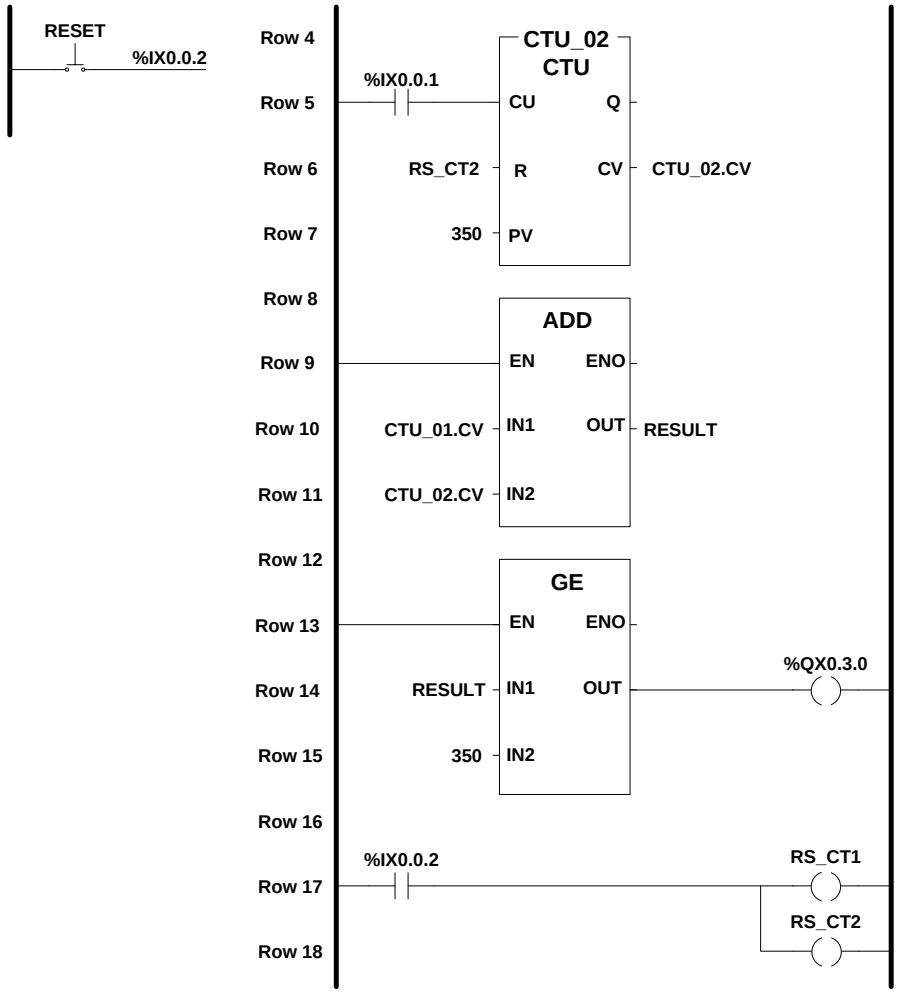
- Kết quả của phép cộng được so sánh với giá trị 350 nhờ phép so sánh lớn hơn hoặc bằng.

- Đèn PL1 sẽ sáng khi kết quả phép cộng lớn hơn hoặc bằng 350.

- Hệ thống có sử dụng nút nhấn khởi động lại RESET để đặt lại giá trị cho hai bộ đếm.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.75. Chương trình ví dụ sử dụng lệnh ADD

4.7.2. Lệnh SUB

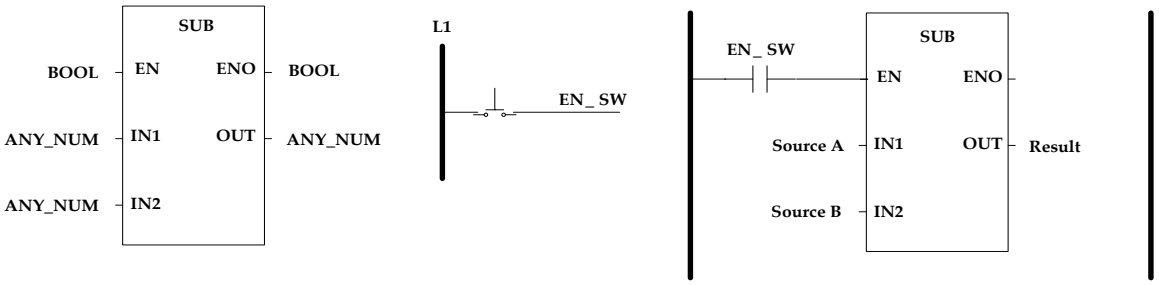
Lệnh SUB (phép trừ) được sử dụng để thực hiện phép tính trừ giữa 2 giá trị số với nhau. Hình 4.105 là một ví dụ đơn giản được sử dụng để giải thích nguyên lý hoạt động của câu lệnh SUB:

- Câu lệnh được thực hiện khi khóa EN_SW tiếp điện.

- Đầu ra Result của câu lệnh là kết quả thực hiện phép tính trừ giữa Source A và Source B.

- Đầu ra ENO có giá trị là TRUE nếu câu lệnh được thực hiện thành công và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào và ra có cùng kiểu dữ liệu số.



Hình 4.76. Nguyên lý hoạt động của lệnh SUB

Ví dụ 41:

Chương trình trên Hình 4.106 được sử dụng để điều khiển một hệ thống giám sát quá trình nạp nguyên liệu vào thùng chứa. Trong ứng dụng này, thùng chứa chỉ chứa được một lượng nguyên liệu tối đa là 500 lb. Nếu lượng nguyên liệu nạp vào lớn hơn giá trị tối đa của bình chứa từ 5 lb trở lên thì hệ thống sẽ phát ra âm thanh cảnh báo. Nguyên lý hoạt động của chương trình như sau:

- Khi nhấn nút START, cuộn Fill Solenoid được cấp điện, đèn báo quá trình nạp liệu Light Filling sẽ sáng, cho phép nguyên liệu được nạp vào thùng chứa.

- Khối lượng của thùng chứa luôn tục được giám sát bởi chương trình PLC trong quá trình nạp liệu (bậc 4).

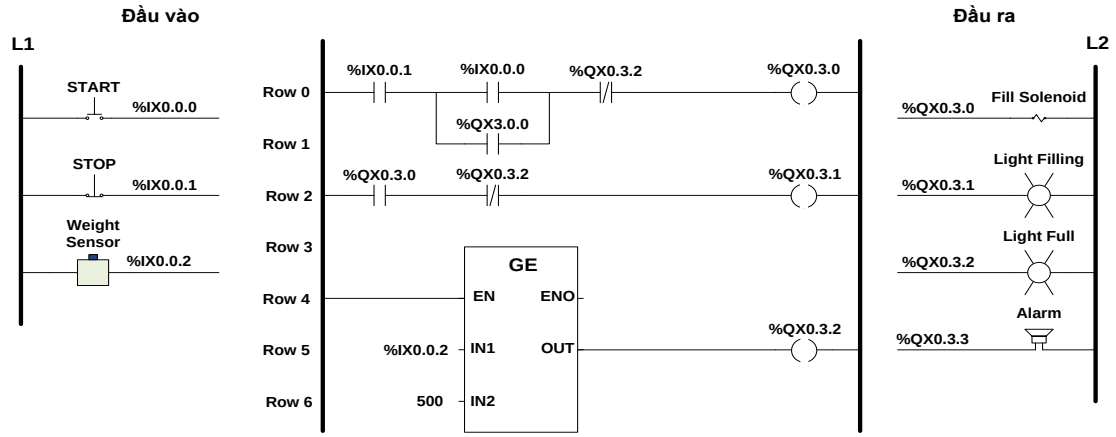
- Khi khối lượng thùng chứa lớn hơn hoặc bằng 500 lb, cuộn hút Fill Solenoid bị ngắt điện và ngừng nạp liệu vào thùng chứa.

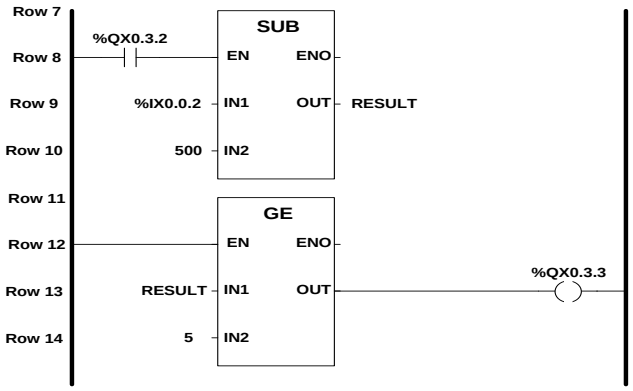
- Cùng thời điểm, đèn báo nạp liệu Light Filling tắt và đèn báo thùng chứa đã được nạp đầy Light Full được bật sáng.

- Nếu khối lượng thùng chứa lớn hơn 500 lb ít nhất là 5 lb thì hệ thống sẽ phát ra âm thanh báo động.

- Hoạt động của hệ thống sẽ bị dừng nếu nút STOP được nhấn.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>





Hình 4.77. Chương trình ví dụ sử dụng lệnh SUB

4.7.3. Lệnh MUL

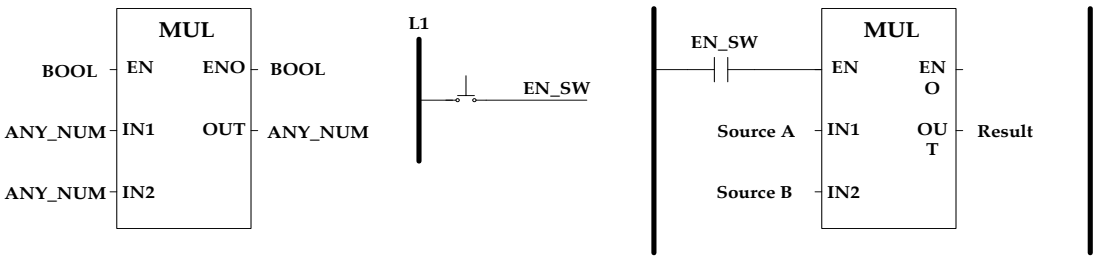
Lệnh MUL (phép nhân) được sử dụng để thực hiện phép tính nhân giữa hai hay nhiều (tối đa là 8) giá trị số với nhau. Hình 4.107 là một ví dụ đơn giản được sử dụng để giải thích nguyên lý hoạt động của câu lệnh MUL với hai đầu vào:

- Câu lệnh được thực hiện khi khóa EN_SW tiếp điện.

- Đầu ra Result của câu lệnh là kết quả thực hiện phép tính nhân giữa Source A và Source B.

- Đầu ra ENO có giá trị là TRUE nếu câu lệnh được thực hiện thành công và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào và ra có cùng kiểu dữ liệu số.



Hình 4.78. Nguyên lý hoạt động của lệnh MUL

Ví dụ 42:

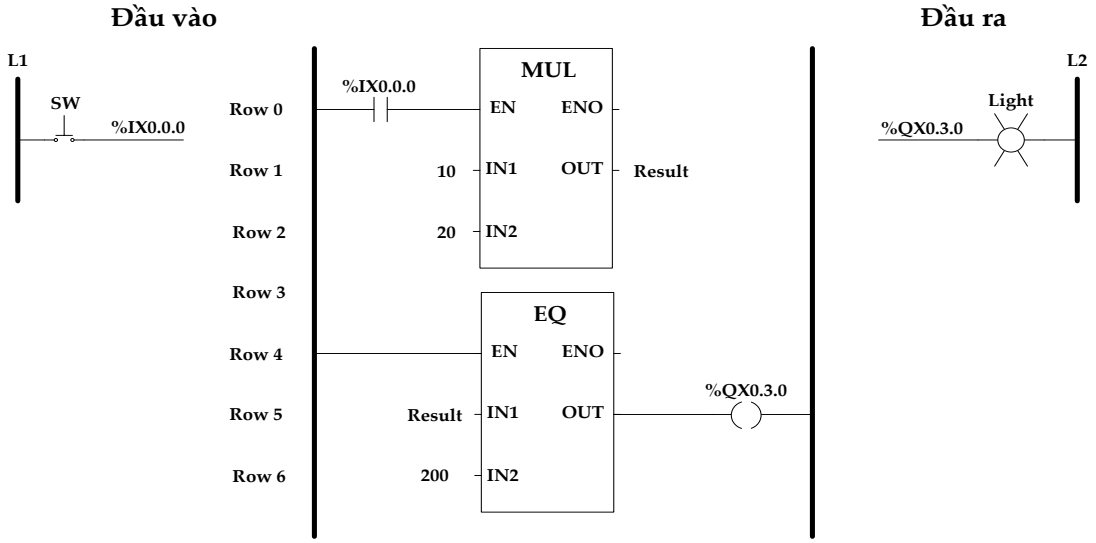
Chương trình được mô tả trên Hình 4.108 là một ví dụ sử dụng câu lệnh MUL để thực hiện phép tính nhân giữa 2 giá trị số với nhau. Nguyên lý hoạt động của chương trình có thể được tóm tắt như sau:

- Khi nút nhấn SW tiếp điện, câu lệnh MUL được thực hiện.

- Giá trị tại 2 đầu vào IN1 và IN2 được nhân với nhau và kết quả được lưu vào biến Result.

- Kết quả phép nhân được so sánh với giá trị 200 tại đầu vào IN2 của câu lệnh so sánh bằng và cho chúng ta kết quả là TRUE và đèn PL1 sẽ sáng.

Chương trình và sơ đồ kết nối các đầu vào/ra:



Hình 4.79. Chương trình ví dụ sử dụng lệnh MUL

4.7.4. Lệnh DIV

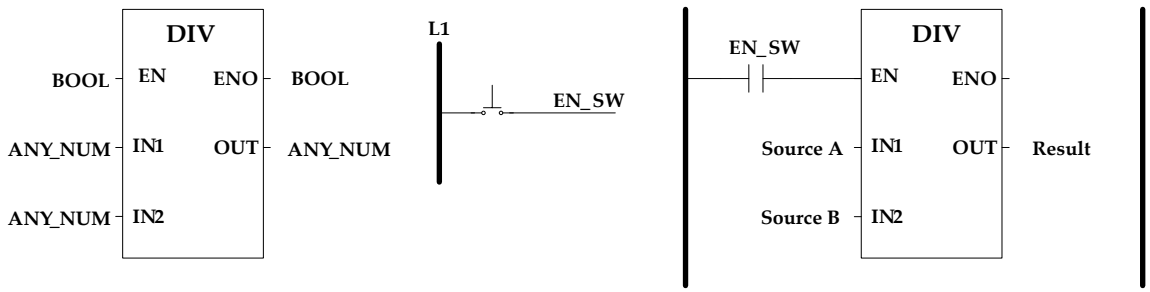
Trong kỹ thuật lập trình PLC, phép chia giữa hai giá trị số có thể được thực hiện bởi câu lệnh DIV. Nguyên tắc hoạt động của câu lệnh DIV có thể được giải thích tóm tắt như trên Hình 4.109 dưới đây:

- Câu lệnh DIV được thực hiện khi khóa SW_EN được tiếp điện.

- Dữ liệu tại địa chỉ Source A chia cho giá trị tại địa chỉ Source B. Kết quả được lưu vào biến Result.

- Đầu ra ENO có giá trị là TRUE nếu câu lệnh được thực hiện thành công và ngược lại sẽ có giá trị là FALSE.

- Các đầu vào và ra có cùng kiểu dữ liệu số.



Hình 4.80. Nguyên lý hoạt động của lệnh DIV

Ví dụ 43:

Hình 4.110 là một ví dụ được sử dụng với mục đích chuyển đổi nhiệt độ từ đơn vị Celsius sang Fahrenheit. Hoạt động của chương trình như sau:

- Núm điều chỉnh bằng tay được kết nối tới mô-đun đầu vào để biểu diễn nhiệt độ Celsius.

- Chương trình được thiết kế để chuyển đổi nhiệt độ Celsius sang nhiệt độ Fahrenheit để hiển thị với công thức chuyển đổi như sau:

$$F = \left(\frac{9}{5}.C\right) + 32$$

- Giả sử giá trị nhiệt độ là 60°C.

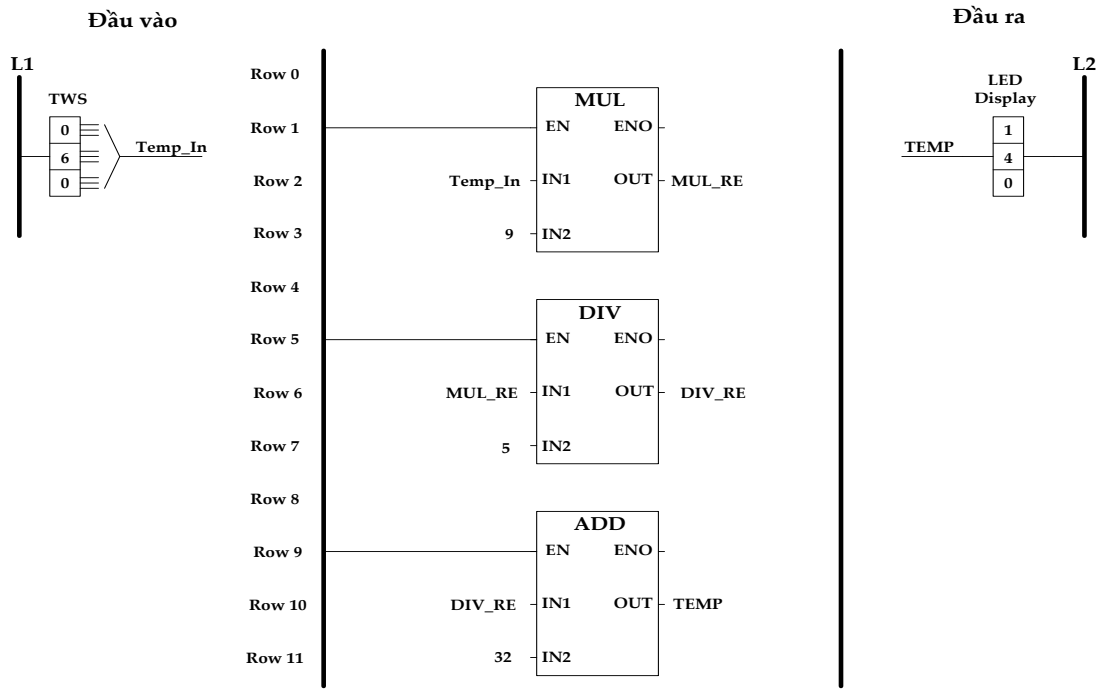
- Sử dụng lệnh MUL để thực hiện phép nhân giữa giá trị nhiệt độ (60°C) với giá trị không đổi (9) và lưu kết quả vào biến MUL_RE.

- Câu lệnh DIV thực hiện phép chia kết quả của phép nhân cho 5 và kết quả (108) được lưu vào biến DIV_RE.

- Sau cùng, lệnh ADD được sử dụng để thực hiện phép cộng giữa kết quả của phép chia với 32 để được giá trị sau cùng (140) và lưu vào biến TEMP.

- Nhiệt độ sau khi chuyển đổi là 140°F.

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>



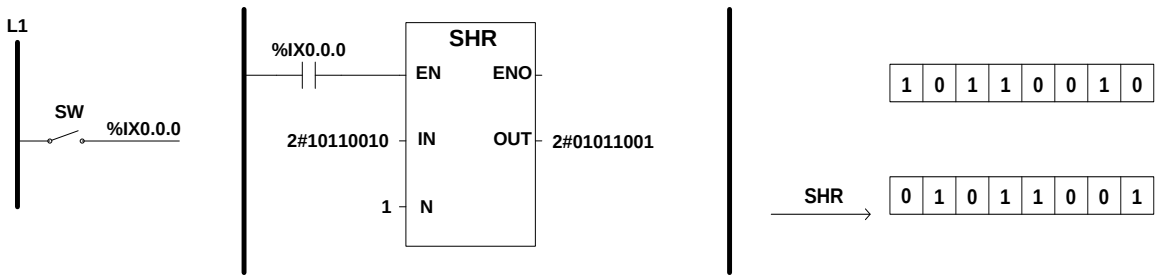
Hình 4.81. Chương trình ví dụ sử dụng lệnh DIV

4.8. THANH GHI DỊCH

Lệnh dịch bit của một thanh ghi được định địa chỉ bởi đầu vào IN đi N vị trí sang phải (SHR) hoặc sang trái (SHL), kết quả lưu vào biến được định địa chỉ bởi đầu ra OUT. Phép ghi dịch điền giá trị 0 vào vị trí các bit đã bị dịch. Như vậy nếu số lần dịch N lớn hơn độ dài của thanh ghi thì kết quả bằng 0. Để hiểu nguyên lý, chúng ta hãy xây dựng một ví dụ sử dụng lệnh dịch phải (SHR) với thanh ghi 8-bit:

- Khi trạng thái đầu vào EN chuyển từ OFF sang ON, câu sẽ lệnh được thực hiện. Khi đó, tất cả các bit được dịch sang bên phải 1 vị trí.

- Bit có vị trí thấp nhất (bit 0) sẽ bị dịch và mất đi, trong khi đó giá trị của bit có vị trí cao nhất (bit 7) được thay bởi giá trị 0.



Hình 4.82. Mô tả nguyên lý hoạt động của thanh ghi dịch phải

Ví dụ 44:

Hình 4.121 mô tả hoạt động của hệ thống khử đầu mỡ sử dụng PLC để điều khiển với nguyên tắc hoạt động của hệ thống như sau:

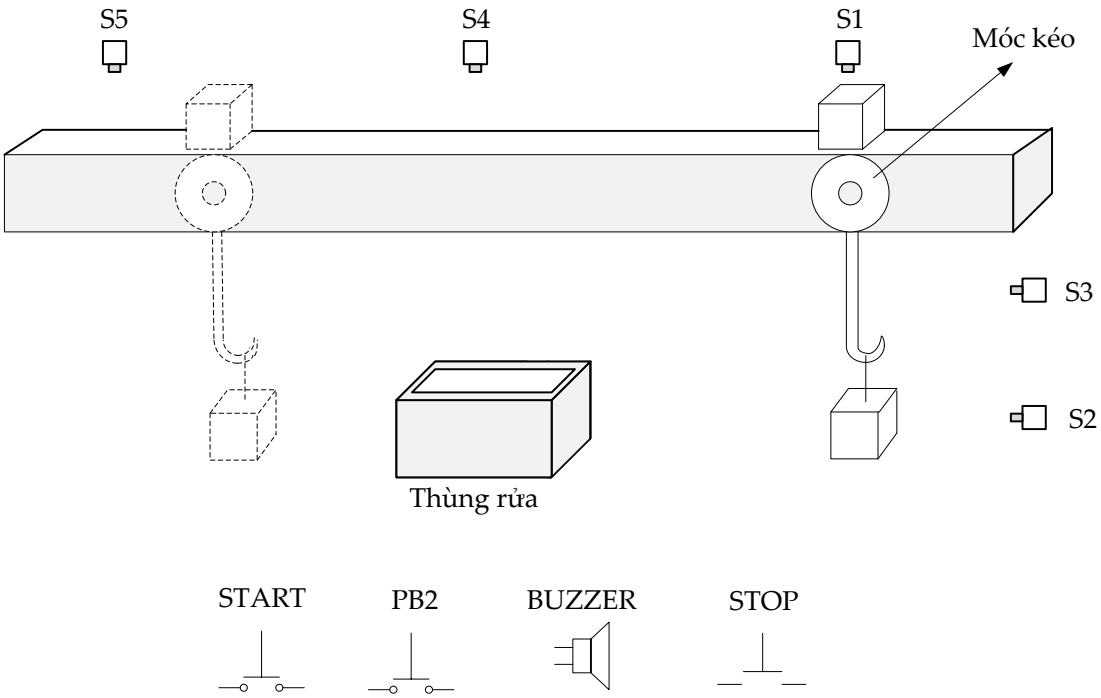
- Khi nút khởi động START được nhấn thì móc kéo sẽ được kéo lên tới khi cảm biến S3 nhận được tín hiệu.

- Móc kéo sẽ di chuyển sang trái cho tới khi cảm biến S4 nhận được tín hiệu. Lúc này nó sẽ được hạ thấp xuống tới vị trí của S2 và sản phẩm sẽ được ngâm trong thùng chứa 20s để làm sạch.

- Sau thời gian 20s, sản phẩm sẽ được nâng lên và di chuyển tiếp sang bên trái và chỉ dừng lại khi S5 nhận được tín hiệu và sẽ đi xuống tới khi nào tiếng còi được vang lên.

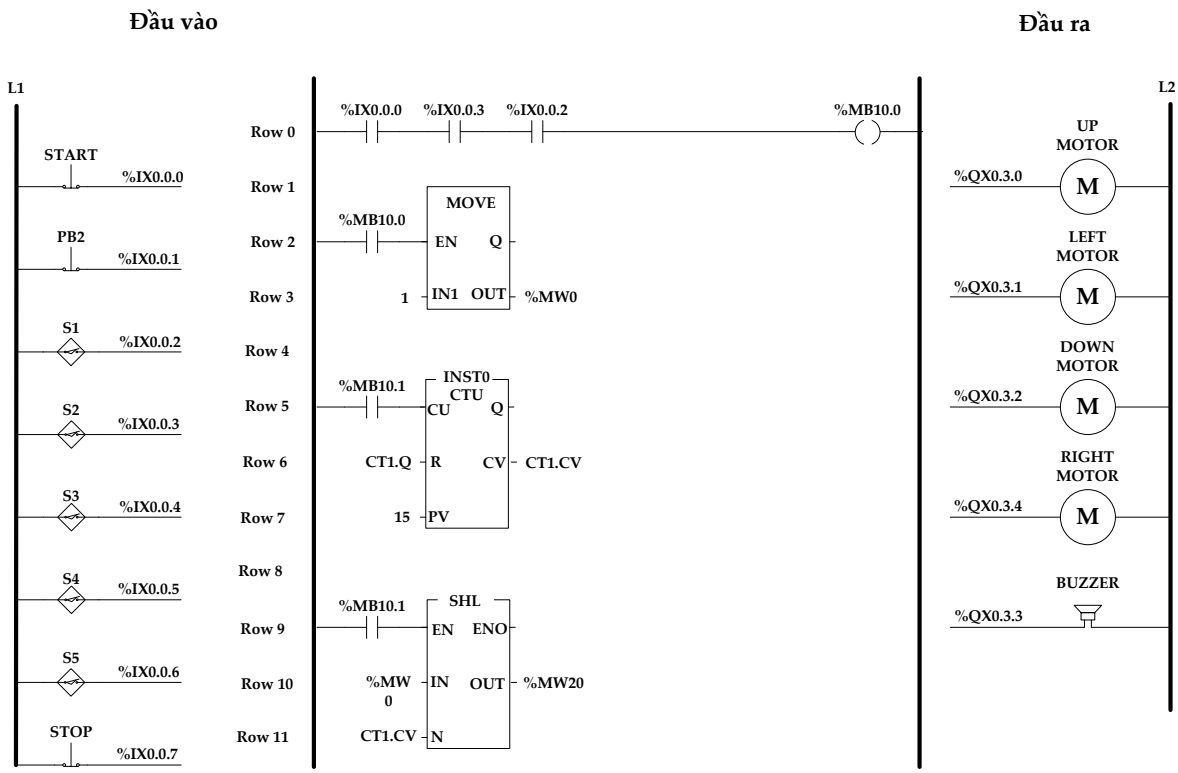
- Sau khi hoàn thành quá trình, nút PB2 được nhấn để móc di chuyển về vị trí ban đầu.

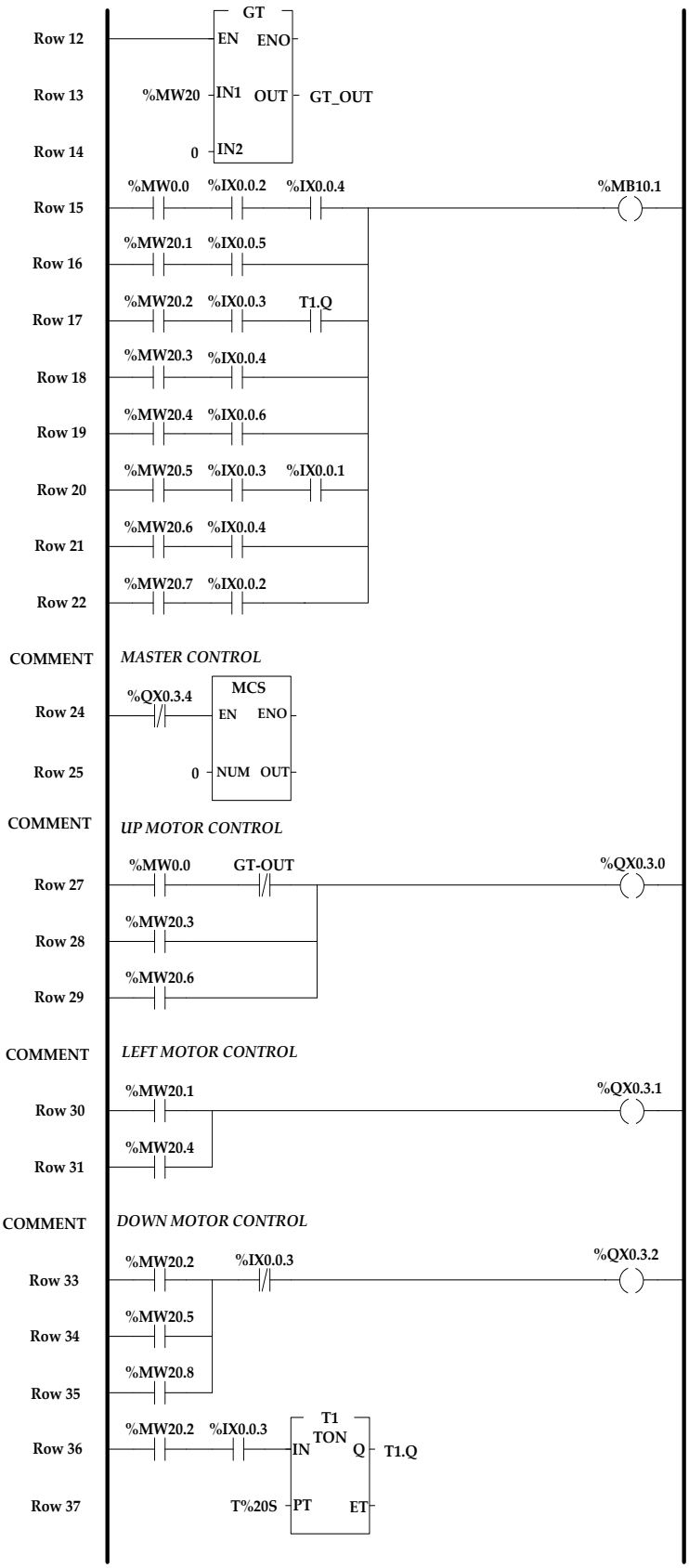
- Nút nhấn tạm dừng STOP được sử dụng để dừng sự di chuyển của móc kéo tại mọi thời điểm. Khi nút STOP được nhả ra thì móc kéo tiếp tục di chuyển theo hướng trước khi bị dừng.

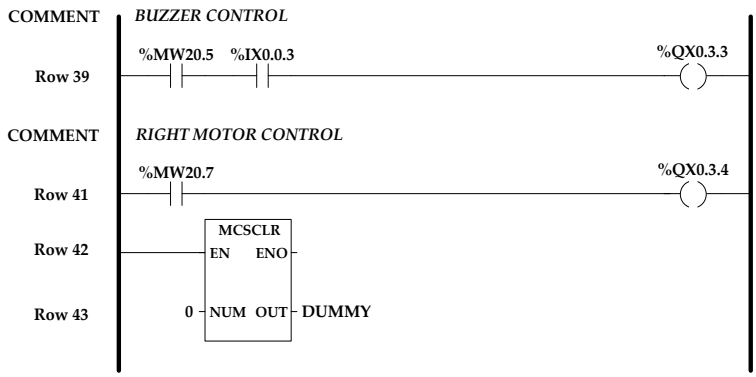


Hình 4.83. Mô hình hệ thống khử dầu

<u>Chương trình và sơ đồ kết nối các đầu vào/ra:</u>




Row 12
Row 13
Row 14
Row 15
Row 16
Row 17
Row 18
Row 19
Row 20
Row 21
Row 22
COMMENT
Row 24
Row 25
COMMENT
Row 27
Row 28
Row 29
COMMENT
Row 30
Row 31
COMMENT
Row 33
Row 34
Row 35
Row 36
Row 37
GT
EN
ENO
%MW20
IN1
OUT
GT_OUT
0
IN2
%MW0.0
%IX0.0.2
%IX0.0.4
%MB10.1
%MW20.1
%IX0.0.5
%MW20.2
%IX0.0.3
T1.Q
%MW20.3
%IX0.0.4
%MW20.4
%IX0.0.6
%MW20.5
%IX0.0.3
%IX0.0.1
%MW20.6
%IX0.0.4
%MW20.7
%IX0.0.2
MASTER CONTROL
%QX0.3.4
MCS
EN
ENO
0
NUM
OUT
UP MOTOR CONTROL
%MW0.0
GT-OUT
%QX0.3.0
%MW20.3
%MW20.6
LEFT MOTOR CONTROL
%MW20.1
%QX0.3.1
%MW20.4
DOWN MOTOR CONTROL
%MW20.2
%IX0.0.3
%QX0.3.2
%MW20.5
%MW20.8
%MW20.2
%IX0.0.3
T1
TON
IN
Q
T1.Q
T%20S
PT
ET



Hình 4.84. Chương trình điều khiển hệ thống khử dầu

CÂU HỎI ÔN TẬP CHƯƠNG 4

PHẦN 1: TIẾP ĐIỂM ĐẦU VÀO VÀ CUỘN HÚT ĐẦU RA

<u>Chú ý:</u> T ký hiệu cho TRUE và F ký hiệu cho FALSE

1. Cho sơ đồ như Hình 4.24. Đầu ra OUT có trạnh thái là ON khi:

   (i) Tiếp điểm đầu vào IN2 được tiếp điện và IN1 tiếp điện tức thời.

   (ii) Đầu vào IN1 hoặc IN2 tiếp điện.

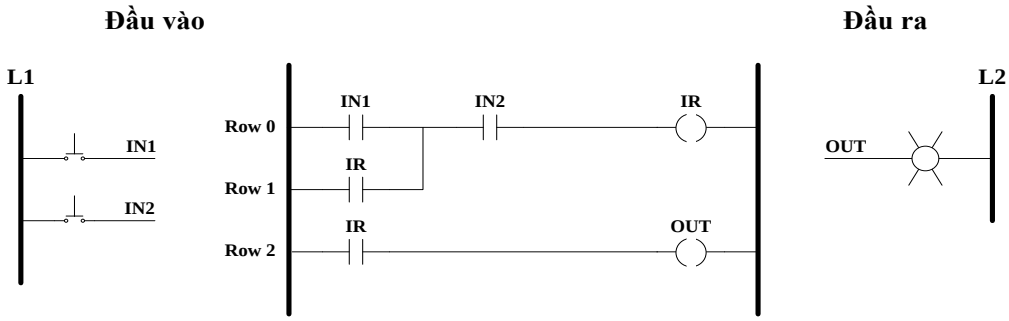
   Hãy chọn đáp án đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F



Hình 4.85. Hình dùng cho Bài 1

2. Cho một đoạn chương trình như Hình 4.25. Đầu ra OUT có trạng thái ON khi:

   (i) Khi các tiếp điểm IN1, IN2 và IN4 tiếp điện.

   (ii) Khi các tiếp điểm IN3 và IN 4 tiếp điện.

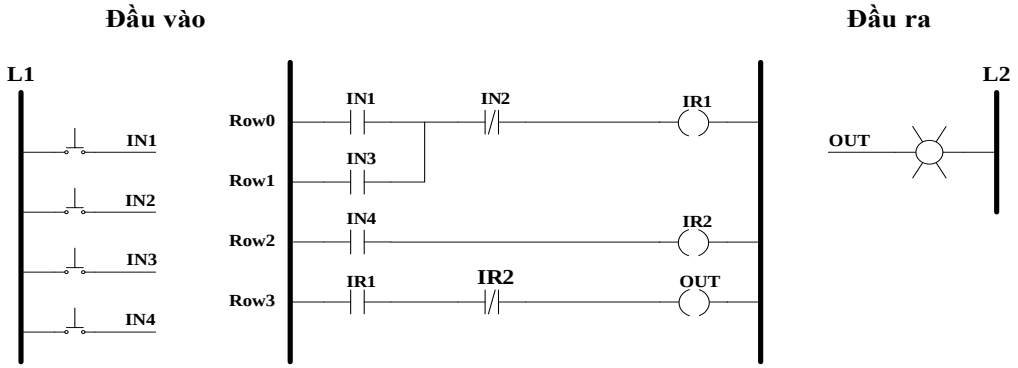
   Hãy chọn đáp án đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F



Hình 4.86. Hình dùng cho Bài 2

3. Cho một đoạn chương trình như Hình 4.26. Đầu ra OUT có trạng thái ON khi:

   (i) Có xung tức thời tại tiếp điểm đầu vào IN1.

   (ii) Tiếp điểm đầu vào IN2 không tiếp điện.

   Chọn câu trả lời đúng trong các câu trả lời dưới đây:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F

4. Cho sơ đồ bậc thang như trên Hình 4.26 với các mệnh đề sau:

   (i) Nếu tiếp điểm đầu vào IN1 tiếp điện, IN2 không tiếp điện thì trạng thái của cuộn hút đầu ra IR là ON và được duy trì ở trạng thái đó thậm trí cả khi đầu vào IN1 không còn tiếp điện nữa.

   (ii) Cuộn hút đầu ra IR là loại cuộn hút có trạng thái được duy trì kể cả khi mất điện vì vậy khi nó đã có trạng thái là ON thì nó vẫn tiếp tục duy trì cho tới khi IN2 có trạng thái là ON.

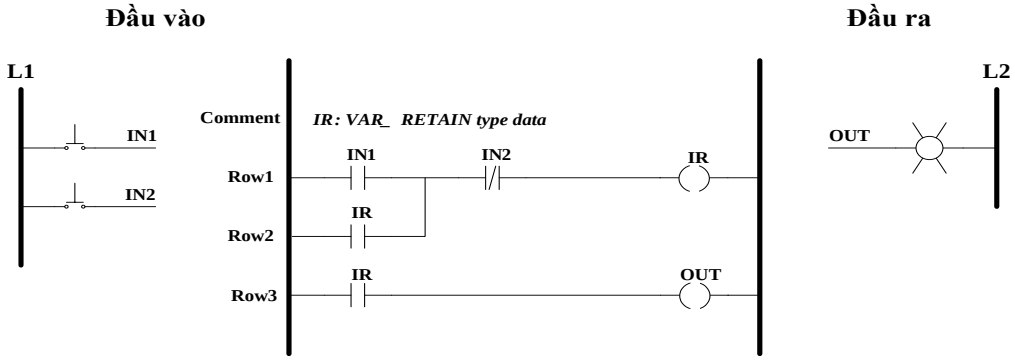
   Hãy chọn đáp án đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F



Hình 4.87. Hình dùng cho Bài 3 và Bài 4

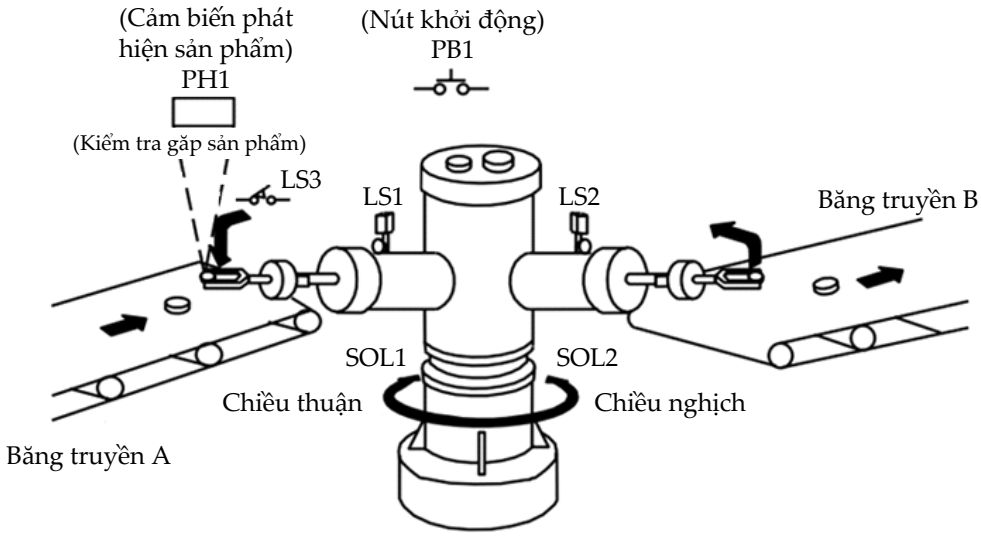
5. Viết các chương trình ứng với mỗi yêu cầu sau:

   a. Trạng thái đầu ra được duy trì khi tiếp điểm đầu vào không còn tiếp điện và khi xảy ra sự cố mất điện.

   b. Đầu ra sẽ được ON trong một chu kỳ quét khi tiếp điểm đầu vào được tiếp điện tức thời.

6. Hình 4.27 là một hệ thống gắp sản phẩm từ băng truyền A sang băng truyền B. Hãy thiết kế chương trình điều khiển cho hệ thống với các yêu cầu sau:

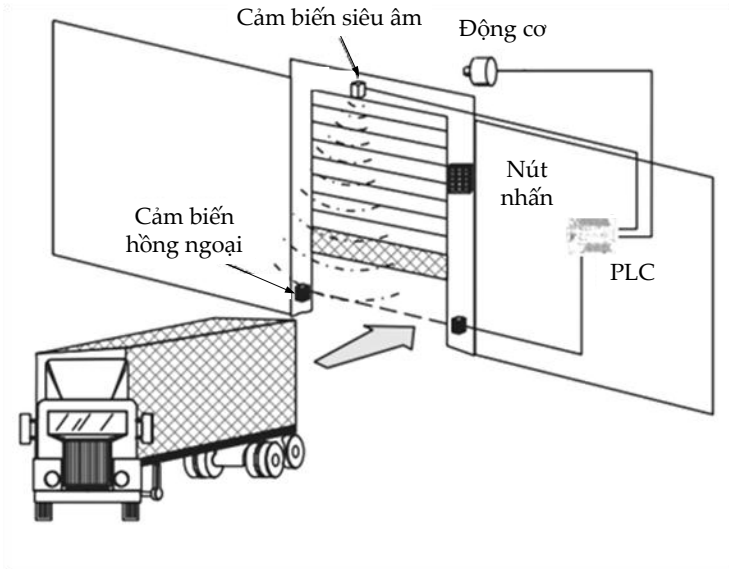
   - Khi nút khởi động START được nhấn, cánh tay robot sẽ quay theo chiều thuận kim đồng hồ.
   - Khi di chuyển tới và phát hiện sản phẩm trên bằng truyền A nó sẽ gắp sản phẩm.
   - Khi đã gắp được sản phẩm nó sẽ quay ngược chiều kim đồng hồ.
   - Khi quay tới vị trí băng truyền B nó sẽ nhả sản phẩm.



Hình 4.88. Hình dùng cho Bài 6

7. Hình 4.28 là một hệ thống điều khiển mở cửa gara ôtô. Thiết kế chương trình điều khiển với các yêu cầu hoạt động như sau:

   - Khi công tắc giới hạn dưới còn nhận được tín hiệu, cảm biến siêu âm phát hiện có ôtô tới, động cơ điều khiển cuốn cửa lên được kích hoạt cho tới khi công tắc giới hạn trên nhận được tín hiệu thi dừng.
   - Khi ôtô đi qua, cảm biến hồng ngoại không còn nhận được tín hiệu nữa, động cơ điều khiển cuốn cửa xuống được kích hoạt cho tới khi công tắc giới hạn dưới được kích hoạt thì dừng.



Hình 4.89. Hình dùng cho Bài 7

PHẦN 2: CÁC BỘ ĐỊNH THỜI

<u>Chú ý:</u> T ký hiệu cho TRUE và F ký hiệu cho FALSE

1. Cho chương trình như Hình 4.43. Khi tiếp điểm đầu vào IN được tiếp điện:

   (i) Bộ định thời TON bắt đầu hoạt động.

   (ii) Trạng thái đầu ra OUT là ON.

   Hãy chọn đáp án đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F

2. Cho sơ đồ bậc thang như Hình 4.3. Khi tiếp điểm đầu vào IN được tiếp điện thì trạng thái đầu ra OUT sẽ là:

   (i) ON trong khoảng thời gian bằng giá trị đặt trước (Pre_Time) cho bộ định thời.

   (ii) OFF trong khoảng thời gian bằng giá trị đặt trước (Pre_Time) cho bộ định thời.

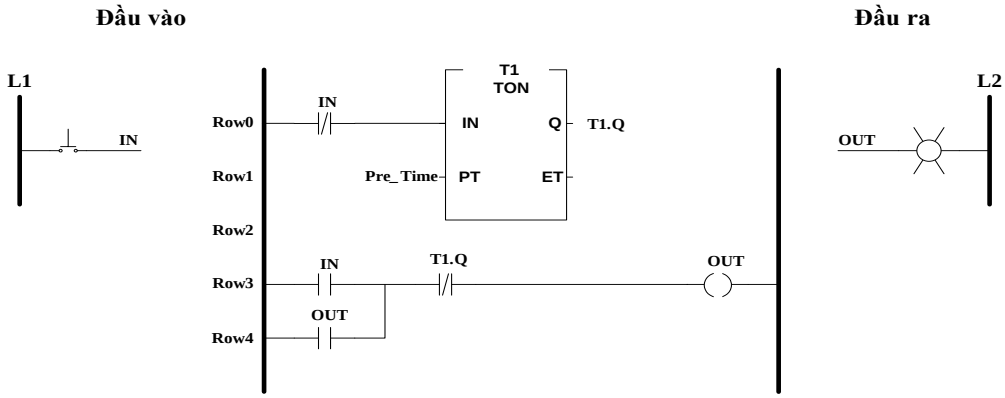
   Hãy chọn câu tra lời đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F



Hình 4.90. Chương trình dùng cho Bài 1 và Bài 2

3. Cho sơ đồ bậc thang như Hình 4.44. Khi tiếp điểm đầu vào IN được tiếp điện thì:

   (i) Bộ định thời TON bắt đầu hoạt động.

   (ii) Trạng thái đầu ra OUT2 là ON.

   Hãy chọn câu tra lời đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F

4. Cho sơ đồ bậc thang như Hình 4.44. Khi đầu vào IN1 tiếp điện, IN2 không tiếp điện thì trạng thái OUT2 sẽ là:

(i) ON ngay lập tức.

(ii) OFF sau một khoảng thời gian bằng giá trị đặt trước (Pre_Time) cho bộ định thời TON.

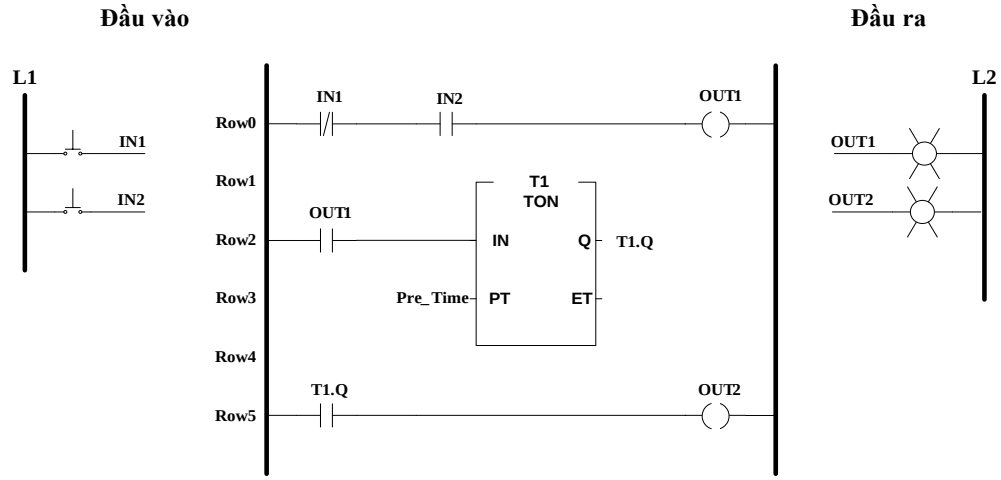
Hãy chọn đáp án đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F



Hình 4.91. Chương trình dùng cho Bài 3 và Bài 4

5. Cho sơ đồ bậc thang như Hình 4.45. Giá trị định trước (Pre_Time) cho bộ định thời TON là 5s, khi tiếp điểm đầu vào IN được tiếp điện:

(i) Trạng thái cuộn hút đầu ra OUT1 chuyển sang ON ngay lập tức.

(ii) Trạng thái cuộn hút đầu ra OUT2 chuyển sang ON sau 5s.

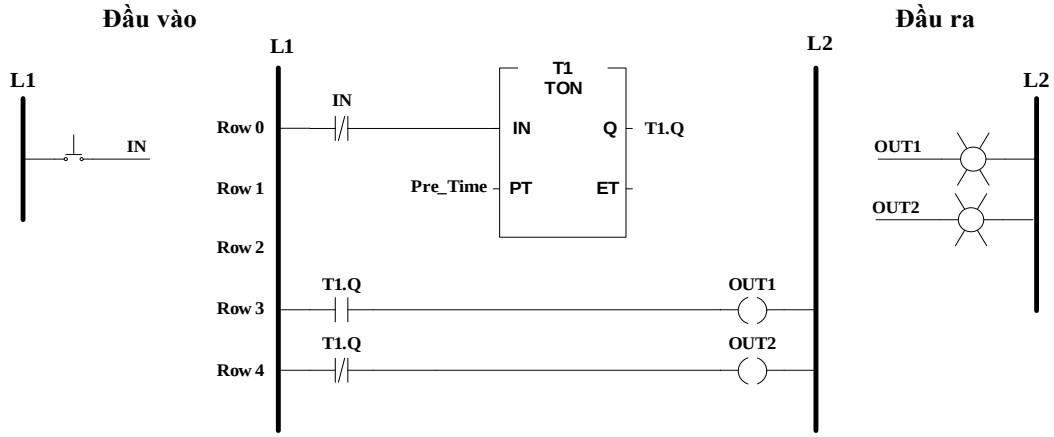
Hãy chọn câu trả lời đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F



Hình 4.92. Chương trình cho Bài 5

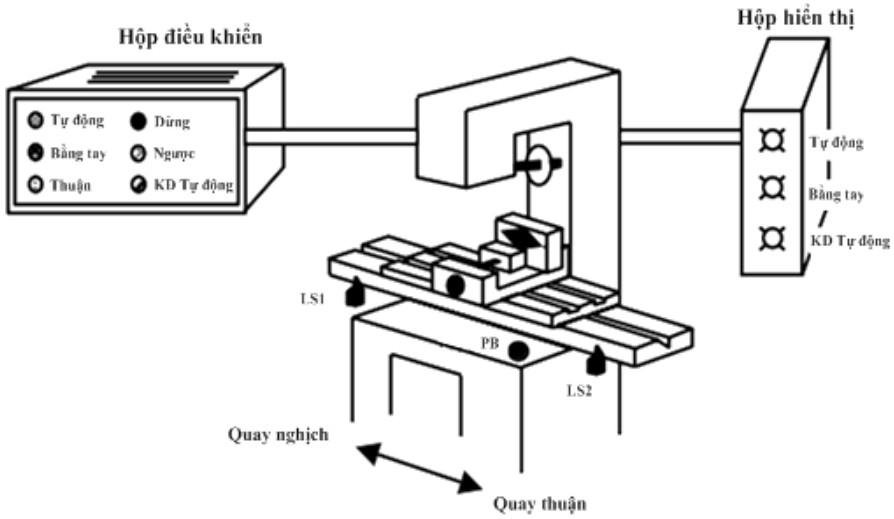
6. Hình 4.46 là một hệ thống khoan. Hãy viết chương trình điều khiển với hai chế độ hoạt động như sau:

A. Chế độ điều khiển bằng tay

- Khi nút nhấn điều khiển chạy thuận SW1 được nhấn, động cơ chạy thuận. nó có thể được dừng bằng nút nhấn dừng SW2. Khi bàn khoan chạm LS2 thì động cơ được dừng.
- Khi nút nhấn điều khiển chạy nghịch SW3 được nhấn, động cơ chạy nghịch. Nó có thể được dừng bằng nút nhấn dừng SW2. Khi bàn khoan chạm LS1 thì động cơ được dừng.

B. Chế độ tự động

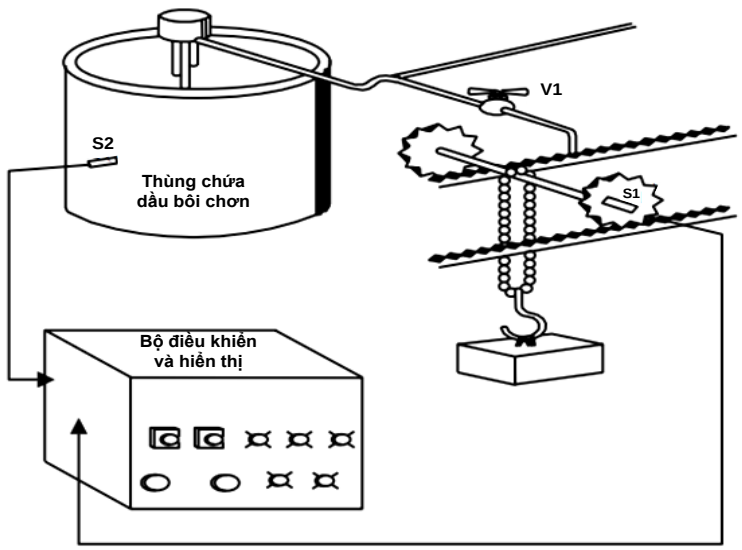
- Khi nút nhấn PB được nhấn và trạng thái LS1 là ON, động cơ sẽ quay thuận cho tới khi chạm LS2. Bộ định thời bắt đầu hoạt động đếm với giá trị định trước 2s.
- Khi kết thúc 2s, động cơ bắt đầu quay ngược tới khi chạm LS1 và chu kỳ hoạt động được lặp lại.



Hình 4.93. Hình dùng cho Bài 6

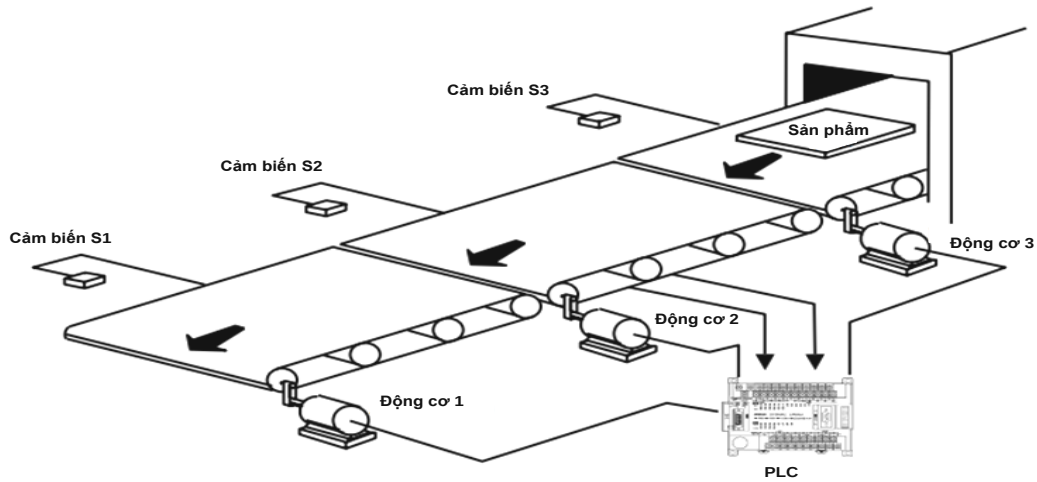
7. Hình 4.47 là hệ thống bôi trơn bộ truyền động. Hãy viết chương trình điều khiển với các yêu cầu sau:

   - Khi bộ truyền động di chuyển tới vị trí mà cảm biến S1 nhận được tín hiệu, hệ thống điều khiển mở van từ V1 để phun dầu bôi trơn lên cơ cấu trong một thời gian cụ thể.
   - Cảm biến S2 được sử dụng để giám sát mức dầu, khi mức dấu xuống thấp, hệ thống sẽ đưa ra cảnh báo nhờ đèn hiển thị.



Hình 4.94. Hình dùng cho Bài 7

8. Trong ứng dụng Hình 4.48, PLC được sử dụng để khởi động và dừng các động cơ điều khiển các phần băng tải riêng biệt. Điều này có nghĩa là chỉ cho phép đoạn băng tải nào có mang theo sản phẩm được di chuyển. Sản phẩm được phát hiện bởi bộ chuyển mạch gần được định vị trên các phần của băng tải. Khi sản phẩm được phát hiện thì động cơ sẽ được điều khiển và sau 2s sẽ được dừng. Hãy viết chương trình điều khiển cho hệ thống



Hình 4.95. Hình dùng cho Bài 8

PHẦN 3: CÁC BỘ ĐẾM LẬP TRÌNH ĐƯỢC

Chú ý: ***T ký hiệu cho TRUE và F ký hiệu cho FALSE***

1. Hình 4.63 là một đoạn chương trình. Bộ đếm được đặt trước giá trị là 5, đầu ra OUT có trạng thái là ON mỗi khi:

   (i) Tiếp điểm đầu vào IN1 được chuyển đổi trạng thái từ OFF sang ON là 5 lần

   (ii) Tiếp điểm đầu vào IN2 được chuyển đổi trạng thái từ OFF sang ON là 5 lần

   Hãy xác định câu trả lời đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F

2. Với chương trình trên Hình 4.63 và các mệnh đề sau:

(i) Đầu vào IN1 trên bậc 0 tạo ra tín hiệu để khởi động lại bộ đếm.

(ii) Đầu vào IN2 trên bậc 2 tạo ra những xung tín hiệu cần thiết để đếm.

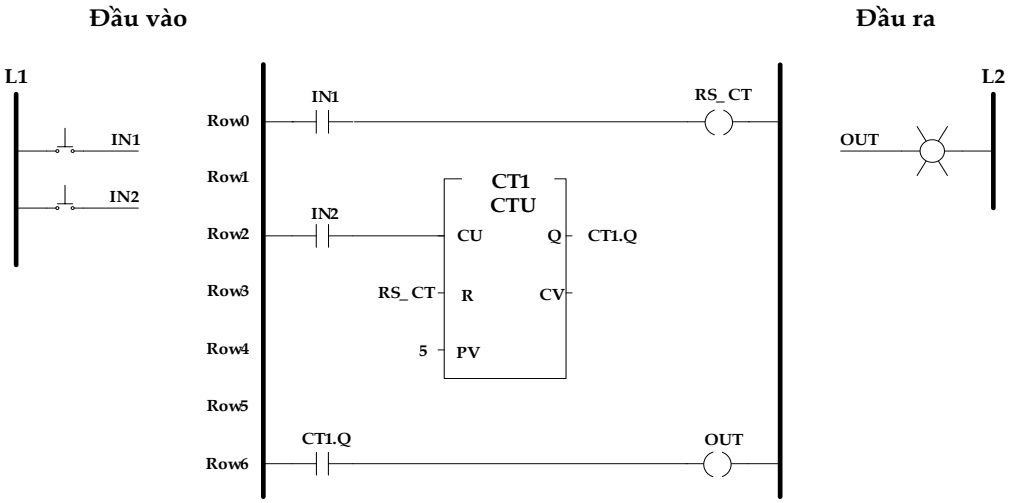
Hãy xác định câu trả lời đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F



Hình 4.96. Chương trình dùng cho Bài 1 và Bài 2

3. Ứng với chương trình trên Hình 4.64. Khi giá trị bộ bộ đếm nhỏ hơn 50 thì:

(i) Đầu ra QU có trạng thái là ON.

(ii) Đầu ra QD có trạng thái là ON.

Hãy chọn đáp án đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F

4. Với chương trình trên Hình 4.64 thì sự kiện nào sẽ xảy ra khi giá trị bộ đếm bằng 50 (giá trị đặt trước):

(i) Đầu ra QU có trạng thái là ON.

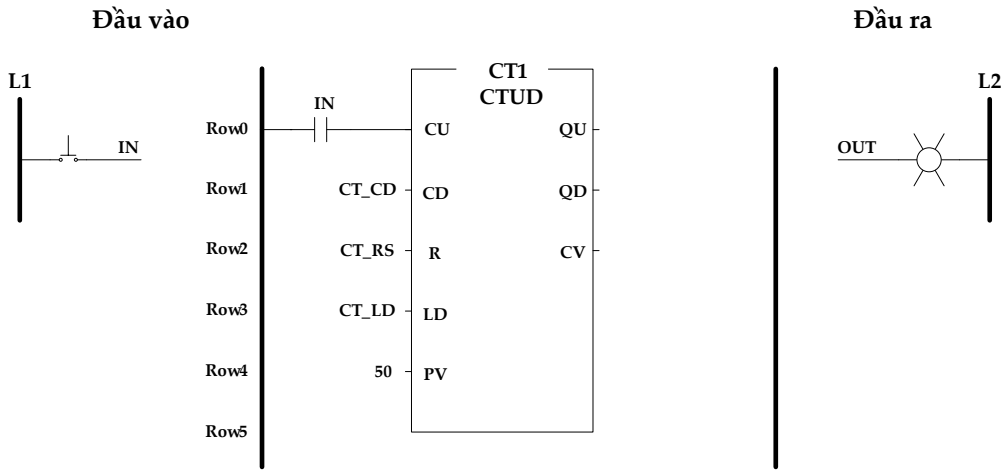
(ii) Đầu ra QD có trạng thái là ON.

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F



Hình 4.97. Chương trình dùng cho Bài 3 và Bài 4

5. Bộ đếm trong chương trình trên Hình 4.65 được khởi động lại khi:

   a. Giá trị đếm được bằng 5.

   b. Giá trị đếm được lớn hơn 5.

   c. Đầu vào IN1 được tiếp điện.

   d. Đầu vào IN2 được tiếp điện.

6. Đầu ra OUT trong chương trình trên Hình 4.65 sẽ có trạng thái ON khi:

   (i) Đầu vào IN1 tiếp điện.

   (ii) Đầu ra CT1.Q của bộ đếm có trạng thái là ON.

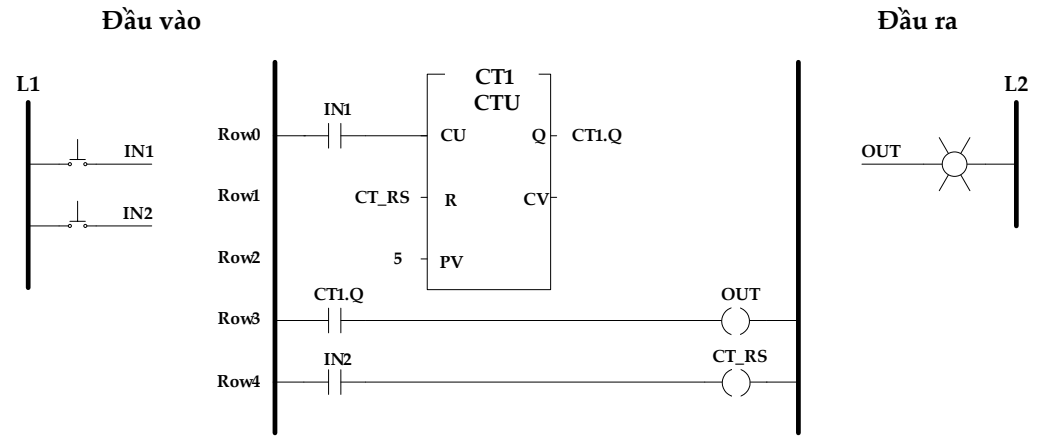
   Hãy chọn câu trả lời đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F



Hình 4.98. Chương trình dùng cho Bài 5 và Bài 6

7. Hãy viết các chương trình thực hiện các chức năng tương ứng sau:

   a. Đầu ra sẽ là ON khi cảm biến quang nhận được 10 xung tín hiệu (giả sử là 10 sản phẩm đi qua nó).

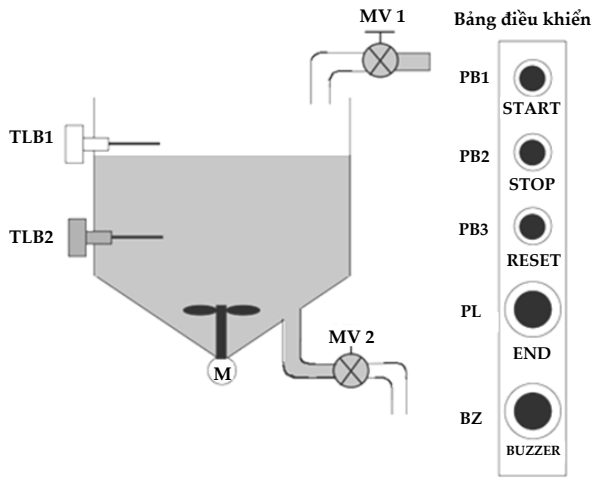
   b. Đầu ra là ON khi có 100 người trong siêu thị (luôn tục có người vào và người ra).

   c. Đèn đỏ sáng khi số lượng sản phẩm nhỏ hơn 5 và đèn xanh sáng khi số lượng sản phẩm lớn hơn hoặc bằng 5.

   d. Đếm số lượng sản phẩm trên băng truyền và đưa ra tín hiệu điều khiển khi số lượng đạt tới 100.

8. Hình 4.66 là một hệ thống nạp xả nhiên liệu. Hãy thiết kế chương trình điều khiển với các yêu cầu sau:

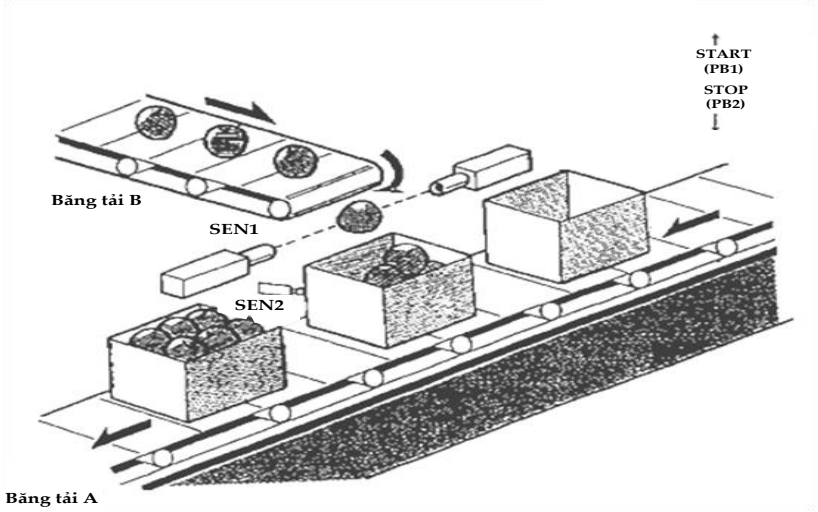
   - Khi nút khởi động PB1 được nhấn, van MV1 mở và nhiên liệu được nạp vào bình. Tại cùng thời điểm, động cơ khuấy M cũng bắt đầu hoạt động.
   - Khi mức nhiên liệu cao hơn TLB2 và đạt tới TLB1, van MV1 đóng và động cơ khuấy M cũng dừng.
   - Tiếp theo, van MV2 mở, nhiên liệu được xả. Khi mức nhiên liệu thấp hơn TLB2 thì van MV2 đóng lại.
   - Khi chu trình hoạt động được thực hiện 4 lần thì đèn chỉ thị END sáng và kết thúc bất chấp nút nhấn PB1 được nhấn.



Hình 4.99. Hệ thống nạp xả nhiên liệu

9. Hình 4.67 là một hệ thống đóng hộp sản phẩm. Hãy viết chương trình thực hiện quá trình điều khiển với các yêu cầu sau:

   - Khi nút khởi động START (PB1) được nhấn, băng tải vận chuyển hộp (Băng chuyền A) chuyển động.
   - Khi cảm biến phát hiện hộp SEN2, băng truyền A dừng và băng chuyền vận chuyển táo (Băng chuyền B) được khởi động.
   - Táo được đưa vào hộp và được đếm bởi cảm biến SEN1. Khi số lượng táo đạt tới giá trị 10 thì băng chuyền B dừng và băng chuyền A được khởi động và quá trình được lặp lại.
   - Bộ đếm được khởi động lại và quá trình được lặp lại cho tới khi nút nhấn dừng STOP (PB2) được nhấn.



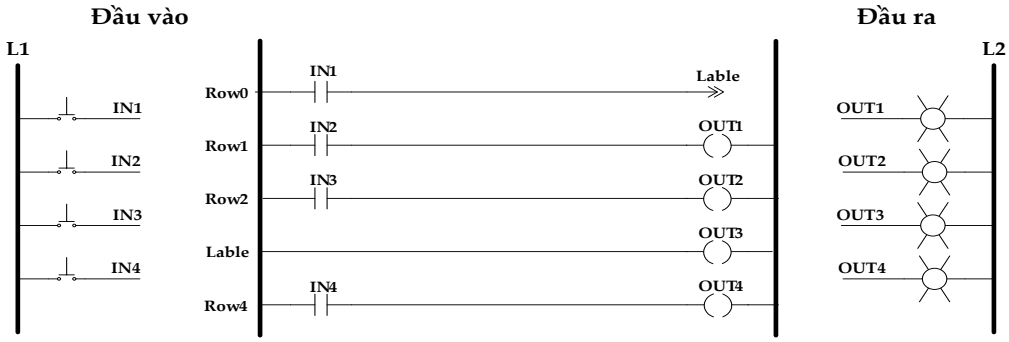
Hình 4.100. Hệ thống đóng hộp sản phẩm

PHẦN 4: CÁC LỆNH ĐIỀU KHIỂN CHƯƠNG TRÌNH

<u>Chú ý:</u> T ký hiệu cho TRUE và F ký hiệu cho FALSE

1. Cho chương trình như trên Hình 4.76. Trạng thái đầu ra OUT1 là ON khi:
   a. Chỉ cần đầu vào IN1 tiếp điện.
   b. Cả IN1 và IN2 tiếp điện.
   c. Đầu vào IN1 không tiếp điện, đầu vào IN2 tiếp điện.
   d. Cả đầu vào IN1 và IN2 không tiếp điện.
2. Chương trình trên Hình 4.76, đầu vào IN1 tiếp điện thì:
   (i) Đầu ra OUT1 có trạng thái là ON.
   (ii) Đầu ra OUT3 có trạng thái là ON.

   Hãy chọn câu trả lời đúng:
   a. (i) T (ii) T
   b. (i) T (ii) F
   c. (i) F (ii) T
   d. (i) F (ii) F

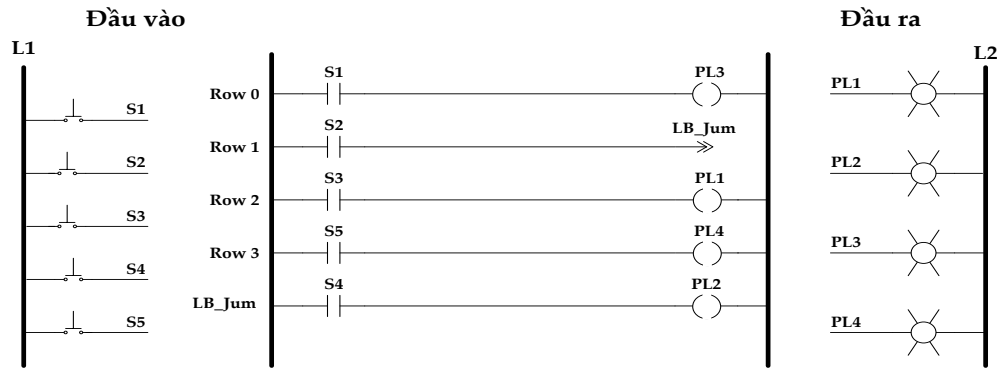


Hình 4.101. Chương trình dùng cho Bài 1 và Bài 2

3. Cho chương trình trên Hình 4.77. Giả sử các nút nhấn được nhả sau mỗi quá trình thực hiện.
   a. Nút nhấn S3 tiếp điện, trạng thái đầu ra PL1 thế nào và tại sao?
   b. Nút nhấn S2 tiếp điện sau đó là S5 cũng được tiếp điện, trạng thái đầu ra PL4 thế nào và tại sao?

c. Nút nhấn S3 tiếp điện, PL1 được kích hoạt. Tiếp theo, nút nhấn S2 được nhấn. Khi đó, trạng thái PL1 thế nào và tại sao?

d. Tất cả các nút nhấn được tiếp điện theo thứ tự S1, S2, S3, S5, và S4. Những đầu ra nào được kích hoạt?



Hình 4.102. Chương trình dùng cho Bài 3

4. Cho chương trình như Hình 4.78 với các phát biểu sau:

(i) Đầu vào IN1 tiếp điện, đầu ra OUT2 có trạng thái là ON.

(ii) Sau khi trạng thái đầu ra OUT3 chuyển sang ON, chương trình sẽ đợi IN2 được tiếp điện rồi mới xử lý.

Hãy chọn câu trả lời đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F

5. Cho chương trình như Hình 4.78 với các phát biểu sau:

(i) Đầu vào IN2 tiếp điện thì OUT1 và OUT2 cũng chuyển sang ON.

(ii) Đầu vào IN2 tiếp điện thì OUT3 chuyển sang trạng thái ON.

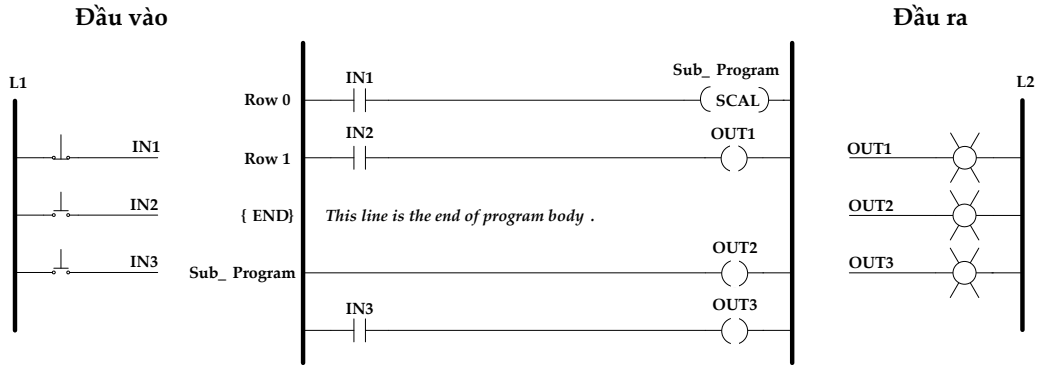
Hãy chọn đáp án đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F

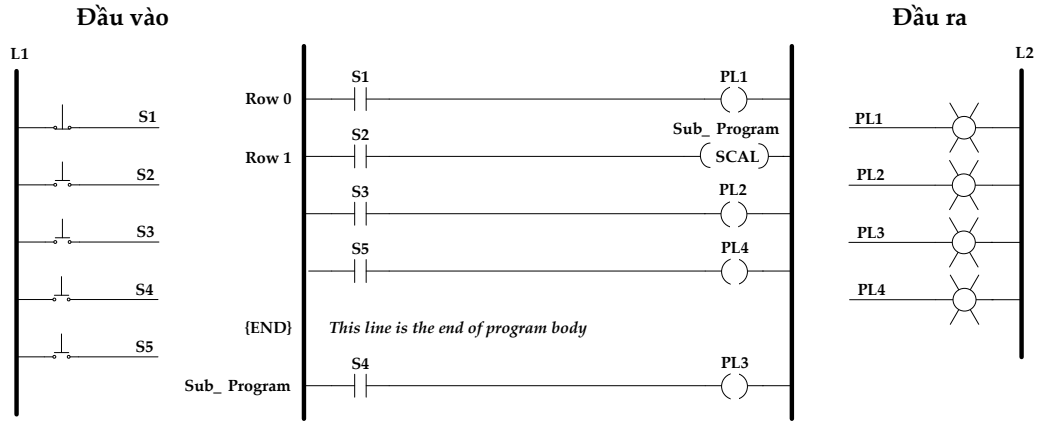


Hình 4.103. Chương trình dùng cho Bài 4 và Bài 5

6. Cho chương trình trên Hình 4.79. Giả sử các nút nhấn được nhả sau mỗi quá trình thực hiện.

   a. Khi các nút nhấn S1, S3, S4 và S5 cùng có trạng thái được tiếp điện thì đèn báo nào không được bật sáng và tại sao?

   b. Khi nút nhấn S2 được tiếp điện và sau đó là S4 thì đèn PL3 sẽ có trạng thái như thế nào và tại sao?



Hình 4.104. Chương trình dùng cho Bài 6

7. Cho chương trình trên Hình 4.80. Giả sử các nút nhấn được nhả sau mỗi quá trình thực hiện.

   a. Các nút nhấn được tiếp điện theo thứ tự S2, S12 và S5, trạng thái đầu ra PL5 thế nào và tại sao?

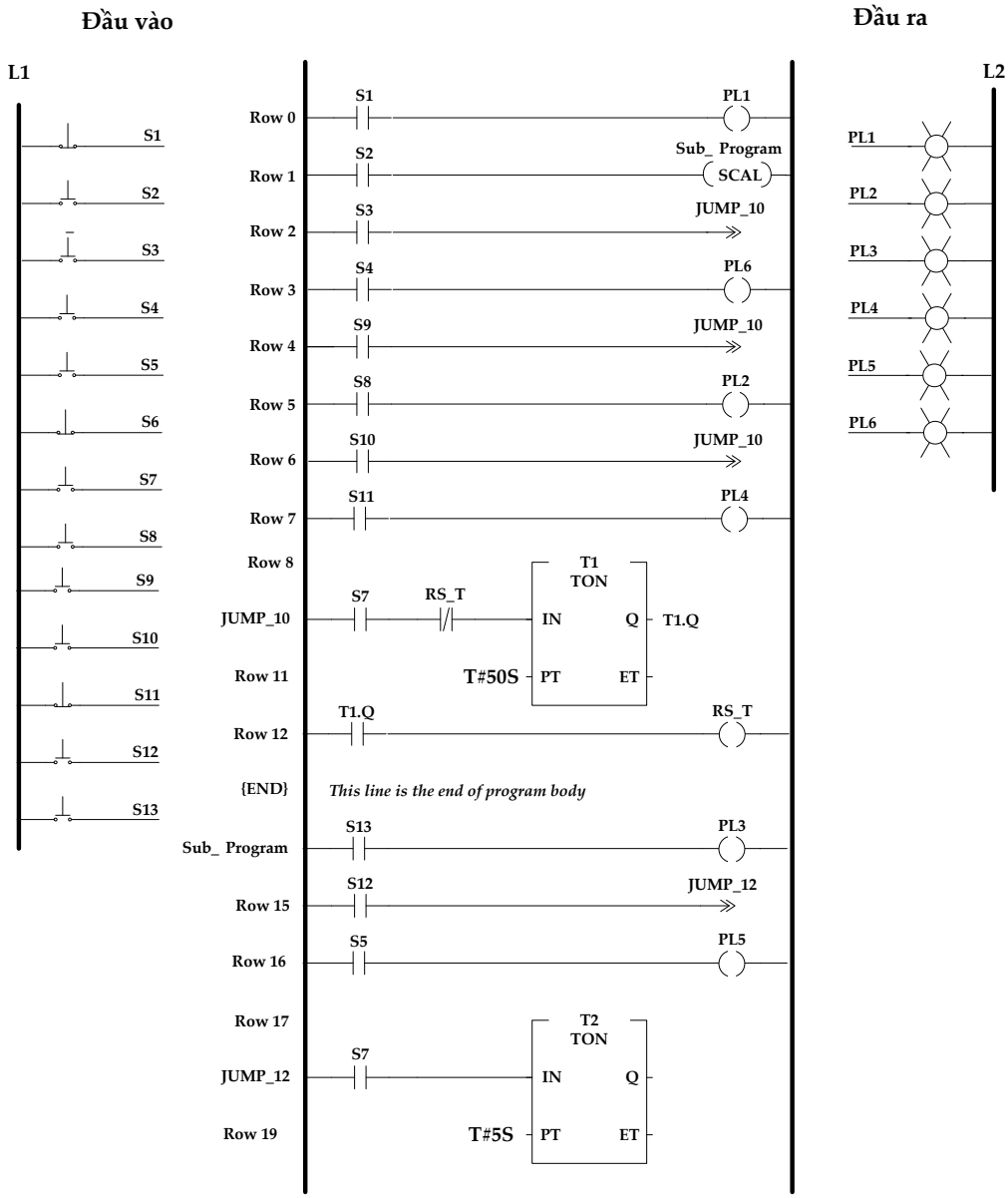
   b. Tất cả các nút nhấn không tiếp điện (ngoại trừ S7), trạng thái bộ định thời thế nào và tại sao?

   c. Các nút nhấn được tiếp điện theo thứ tự S3 và S8. Đèn PL2 có sáng không và tại sao?

   d. Bộ định thời T1 hoạt động khi nào?

e. Giả sử tất cả các nút nhấn tiếp điện, chương trình sẽ quét theo thứ tự như thế nào?

f. Giả sử tất cả các nút nhấn không tiếp điện, chương trình sẽ thực hiện như thế nào?



Hình 4.105. Chương trình dùng cho Bài 8

PHẦN 5: C ÁC LỆNH XỬ LÝ DỮ LIỆU

Chú ý: T ký hiệu cho TRUE và F ký hiệu cho FALSE

1. Cho chương trình như Hình 4.95 và các mệnh đề sau:

(i) Khi sao chép dữ liệu từ địa chỉ %MW100 sang %MW200, dữ liệu tại địa chỉ %MW100 bị mất.

(ii) Khi sao chép dữ liệu từ địa chỉ %MW100 sang %MW200, dữ liệu tại địa chỉ %MW100 giữ nguyên.

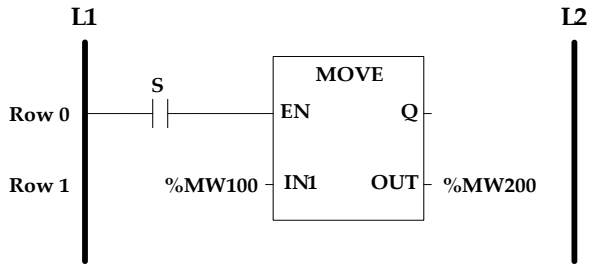
Hãy chọn đáp án đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F

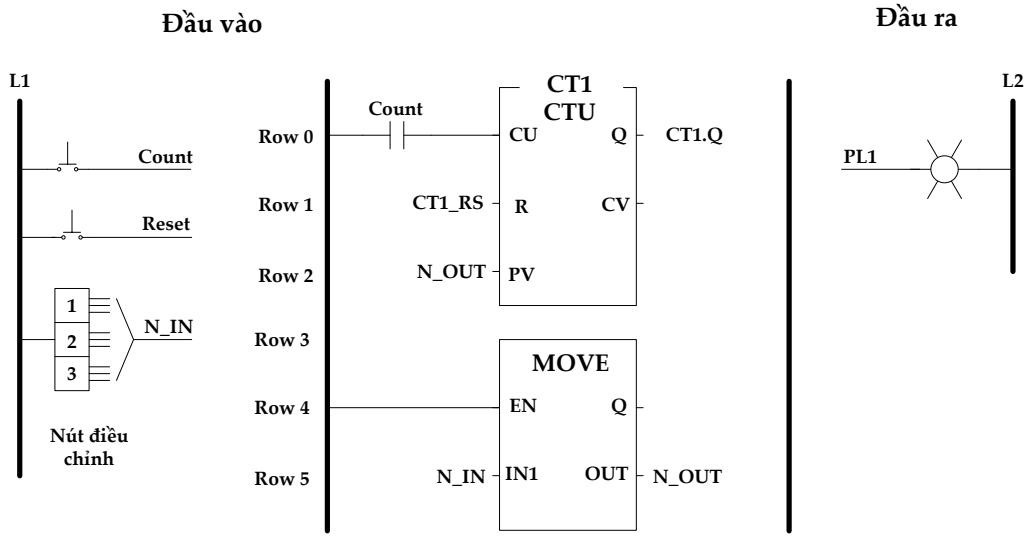


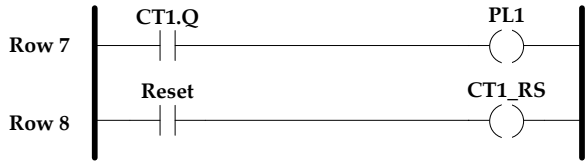
Hình 4.106. Chương trình dùng cho Bài 1

2. Cho chương trình trên Hình 4.96. Trả lời các câu hỏi sau đây:

a. Giá trị đặt trước cho bộ định thời là bao nhiêu?

b. Chỉ ra các bước thực hiện cần thiết để đèn PL1 sáng sau 25 lần có tín hiệu chuyển đổi trạng thái từ OFF sang ON xuất hiện tại đầu vào CU của bộ đếm?





Hình 4.107. Chương trình dùng cho Bài 2

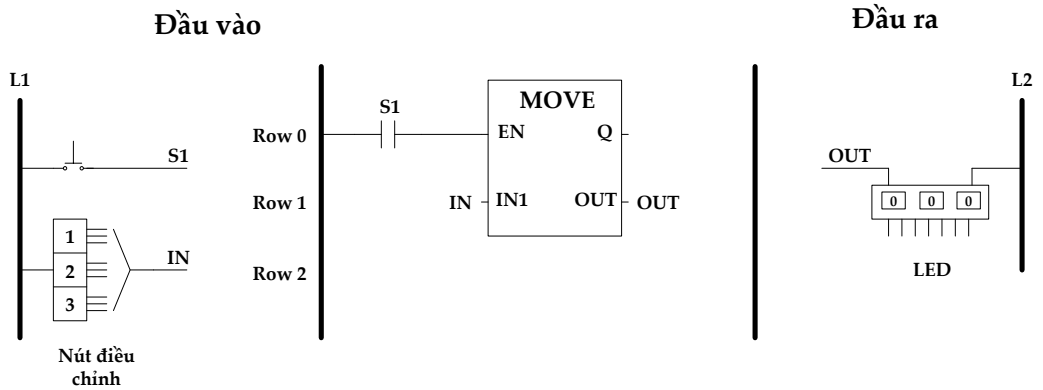
3. Cho chương trình trên Hình 4.97. Trả lời các câu hỏi sau đây:

   a. Khi nút nhấn S1 hở, dữ liệu được lưu vào biến OUT đúng hay sai?

   b. Khi nút nhấn S1 tiếp điện, dữ liệu được lưu vào biến IN đúng hay sai?

   c. Khi nút nhấn S1 tiếp điện, trạng thái của LED sẽ như thế nào?

   d. Cần phải thực hiện như thế nào để có số 216 hiển thị trên LED?



Hình 4.108. Chương trình dùng cho Bài 3

4. Cho chương trình như trên Hình 4.98. Đầu ra OUT chuyển sang trạng thái ON khi:

   (i) Giá trị đếm được của bộ định thời lớn hơn 400.

   (ii) Giá trị đếm được của bộ định thời nhỏ hơn hoặc bằng 400.

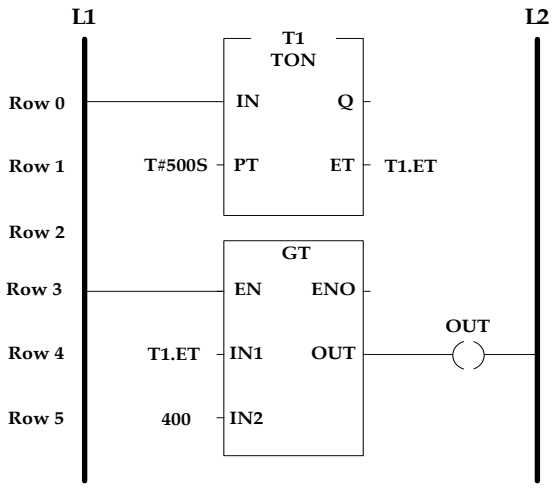
   Hãy chọn câu trả lời đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F



Hình 4.109. Chương trình dùng cho Bài 4

5. Cho chương trình như trên Hình 4.99. Khi đầu vào IN tiếp điện thì đầu ra OUT chuyển sang trạng thái ON khi:

   (i) Giá trị tại địa chỉ %MW100 bằng với giá trị tại địa chỉ %MW200.

   (ii) Giá trị tại địa chỉ %MW100 nhỏ hơn giá trị tại địa chỉ %MW200.

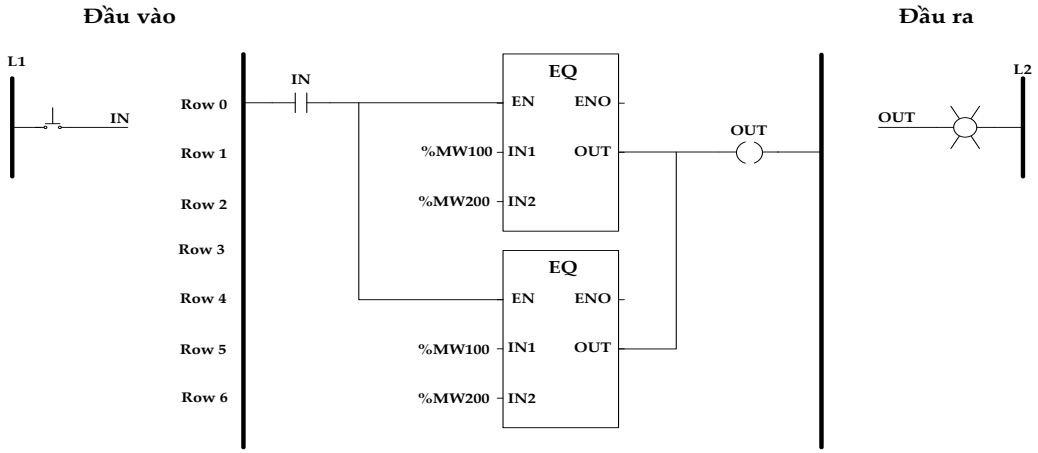
   Hãy chọn câu trả lời đúng:

   a. (i) T (ii) T

   b. (i) T (ii) F

   c. (i) F (ii) T

   d. (i) F (ii) F



Hình 4.110. Chương trình dùng cho bài 5

6. Cho chương trình như trên Hình 4.100. Khi đầu vào IN tiếp điện thì đầu ra OUT chuyển sang trạng thái ON khi:

(i) Giá trị tại địa chỉ %MW100 không bằng giá trị tại địa chỉ %MW200.

(ii) Giá trị tại địa chỉ %MW100 lớn hơn hoặc nhỏ hơn giá trị tại địa chỉ %MW200.

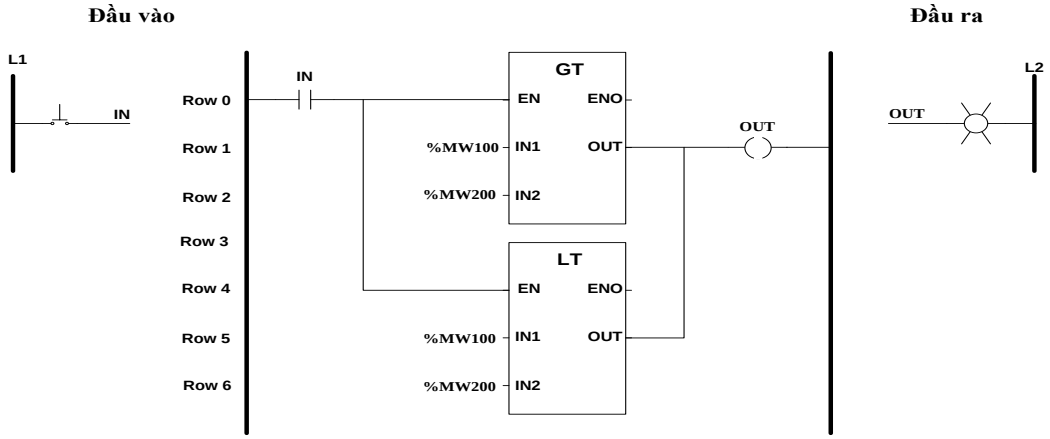
Hãy chọn câu trả lời đúng:

a. (i) T (ii) T

b. (i) T (ii) F

c. (i) F (ii) T

d. (i) F (ii) F



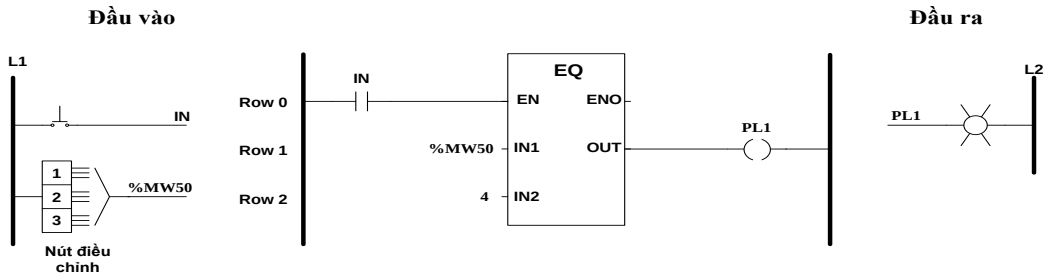
Hình 4.111. Chương trình dùng cho Bài 6

7. Nghiên cứu đoạn chương trình trên Hình 4.101 và trả lời các câu hỏi sau đây:

a. Đèn chỉ thị PL1 sẽ sáng bất cứ khi nào nút nhấn S1 được tiếp điện? Tại sao?

b. Khi nút nhấn S1 tiếp điện sẽ làm thay đổi giá trị được lưu tại địa chỉ %MW50?

c. Phải sử dụng nút điều chỉnh để đặt giá trị bằng bao nhiêu để đèn chỉ thị PL1 có thể sáng?



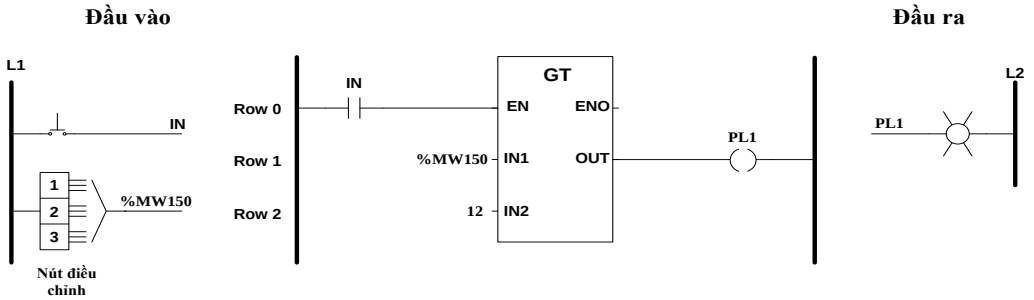
Hình 4.112. Chương trình dùng cho Bài 7

8. Nghiên cứu đoạn chương trình trên Hình 4.102 và trả lời các câu hỏi sau đây:

   a. Liệt kê các giá trị phải chỉnh bằng nút điều chỉnh để đèn PL1 sáng?

   b. Nếu giá trị đặt bởi nút điều chỉnh là 003 và S1 tiếp điện thì đèn PL1 sáng hay tắt và tại sao?

   c. Thay vì giá trị đầu vào IN2 là 12 thì giờ chúng ta thay bởi giá trị đếm được của bộ đếm. Như vậy chúng ta cần đặt giá trị cho đầu vào IN1 bằng bao nhiêu để khi tiếp điểm đầu vào S1 tiếp điện và giá tri bộ đếm đạt tới 150 thì đèn PL1 tắt?



Hình 4.113. Chương trình dùng cho Bài 8

9. Viết chương trình thực hiện các công việc sau đây:

   a. Bật sáng đèn PL1 khi giá trị đầu vào được đặt bằng nút điều chỉnh nhỏ hơn 4.

   b. Bật sáng đèn PL2 khi giá trị đầu vào đặt bằng nút điều chỉnh bằng 4.

   c. Bật sáng đèn PL3 khi giá trị đầu vào được đặt bằng nút điều chỉnh lớn hơn 4.

   d. Bật sáng đèn PL1 khi giá trị đầu vào được đặt bằng nút điều chỉnh nhỏ hơn hoặc bằng 4.

   e. Bật sáng đèn PL1 khi giá trị đầu vào được đặt bằng nút điều chỉnh lớn hơn hoặc bằng 4.

10. Viết chương trình điều khiển thực hiện các nhiệm vụ sau:

   a. Khởi động máy bơm khi mực nước trong thùng chứa lớn hơn 1.2m và tắt nó khi mực nước xuống dưới 1.0m.

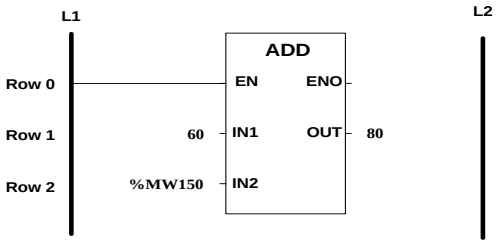
   b. Trước tiên là khởi động động cơ thứ nhất, sau 30s bật bộ phận nung và 100s bật tiếp động cơ thứ hai.

   c. Bật bộ phận nung khi nhiệt độ nhỏ hơn giá trị mong muốn.

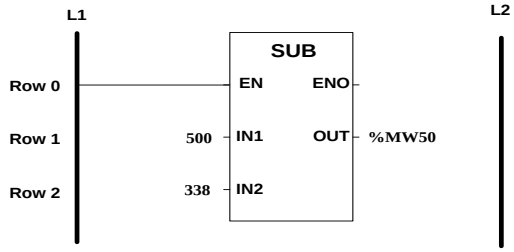
   d. Bật sáng đèn khi đầu vào dữ liệu không bằng 100.

PHẦN 6: CÁC LỆNH TOÁN HỌC

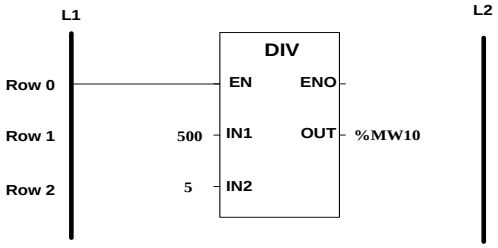
1. Cho đoạn chương trình như Hình 4.111. Giá trị của %MW150 là bao nhiêu?
2. Cho đoạn chương trình như Hình 4.112. Giá trị của %MW50 là bao nhiêu?
3. Cho đoạn chương trình như Hình 4.113. Giá trị của %MW10 là bao nhiêu?
4. Cho đoạn chương trình như Hình 4.114. Giá trị của %MW20 là bao nhiêu?



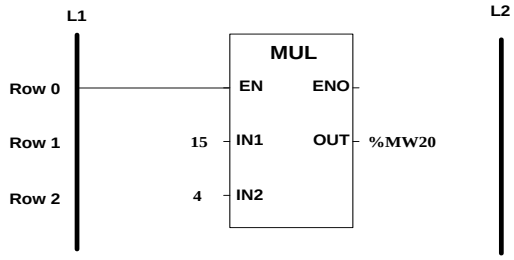
Hình 4.114. Hình dùng cho Bài 1



Hình 4.115. Hình dùng cho Bài 2

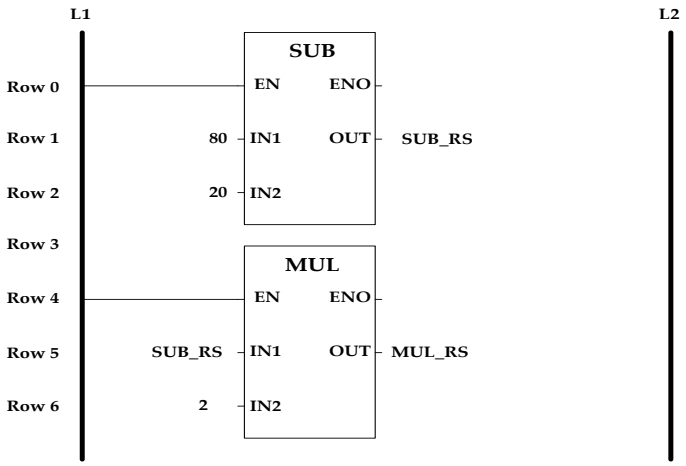


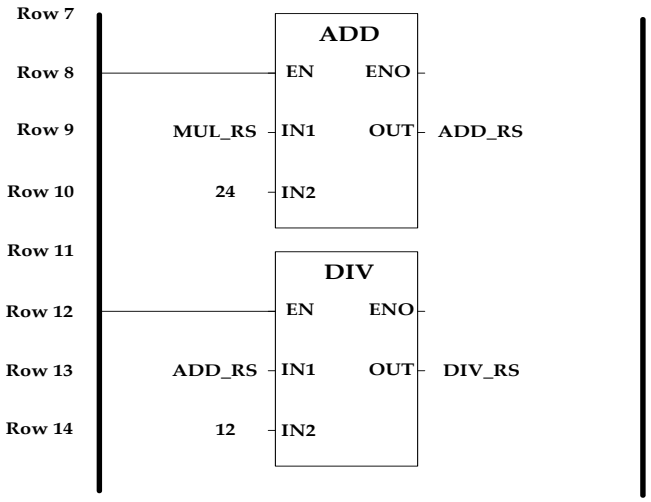
Hình 4.116. Hình dùng cho Bài 3



Hình 4.117. Hình dùng cho Bài 4

5. Cho chương trình trên Hình 4.115. Giá trị của các biến sau là bao nhiêu:
   a. Biến SUB_RS?
   b. Biến MUL_RS?
   c. Biến ADD_RS?
   d. Biến DIV_RS?





Hình 4.118. Chương trình dùng cho Bài 5

6. Cho chương trình như Hình 4.116. Trả lời những câu hỏi sau:

   a. Giả sử giá trị đếm được của các bộ đếm CT1 và CT2 tương ứng là 148 và 36. Khi đó giá trị các biến sau là bao nhiêu:

      (1) Đầu ra CT1.CV của bộ đếm CT1?

      (2) Đầu ra CT2.CV của bộ đếm CT2?

      (3) Đầu ra RESULT của câu lệnh ADD?

   b. Tại thời điểm này đèn PL1 có sáng hay không và tại sao?

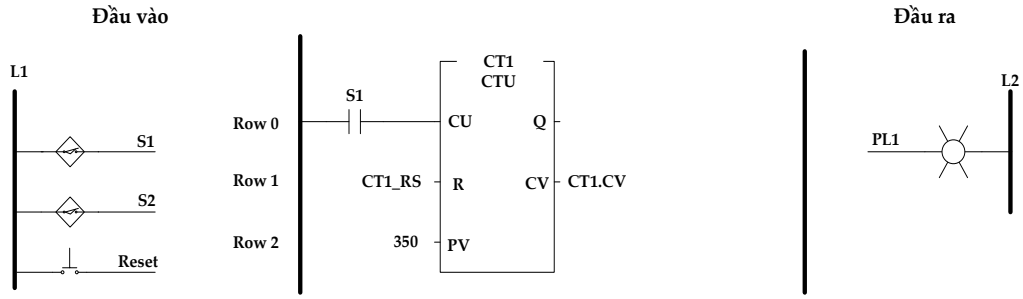
   c. Giả sử giá trị đếm được của các bộ đếm CT1 và CT2 tương ứng là 250 và 175. Khi đó giá trị các biến sau là bao nhiêu:

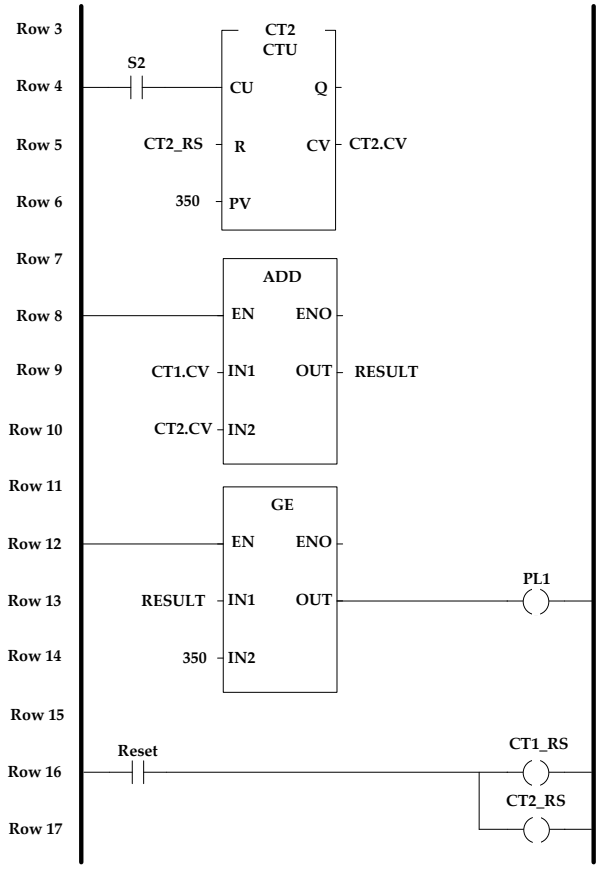
      (1) Đầu ra CT1.CV của bộ đếm CT1?

      (2) Đầu ra CT2.CV của bộ đếm CT2?

      (3) Đầu ra RESULT của câu lệnh ADD?

   d. Tại thời điểm này đèn PL1 có sáng hay không và tại sao?





Hình 4.119. Chương trình dùng cho Bài 6

7. Hình 4.117 là chương trình điều khiển hệ thống giám sát quá trình nạp nhiên liệu.

   a. Giả sử bình chứa được nạp nhiên liệu và đạt tới giá trị 300 lb. Trạng thái logic của mỗi bậc trong chương trình sẽ như thế nào?

   b. Giả sử bình chứa được nạp nhiên liệu và đạt tới giá trị 480 lb. Giá trị của các biến sau thế nào?

      (1) Biến Data?

      (2) Biến RESULT?

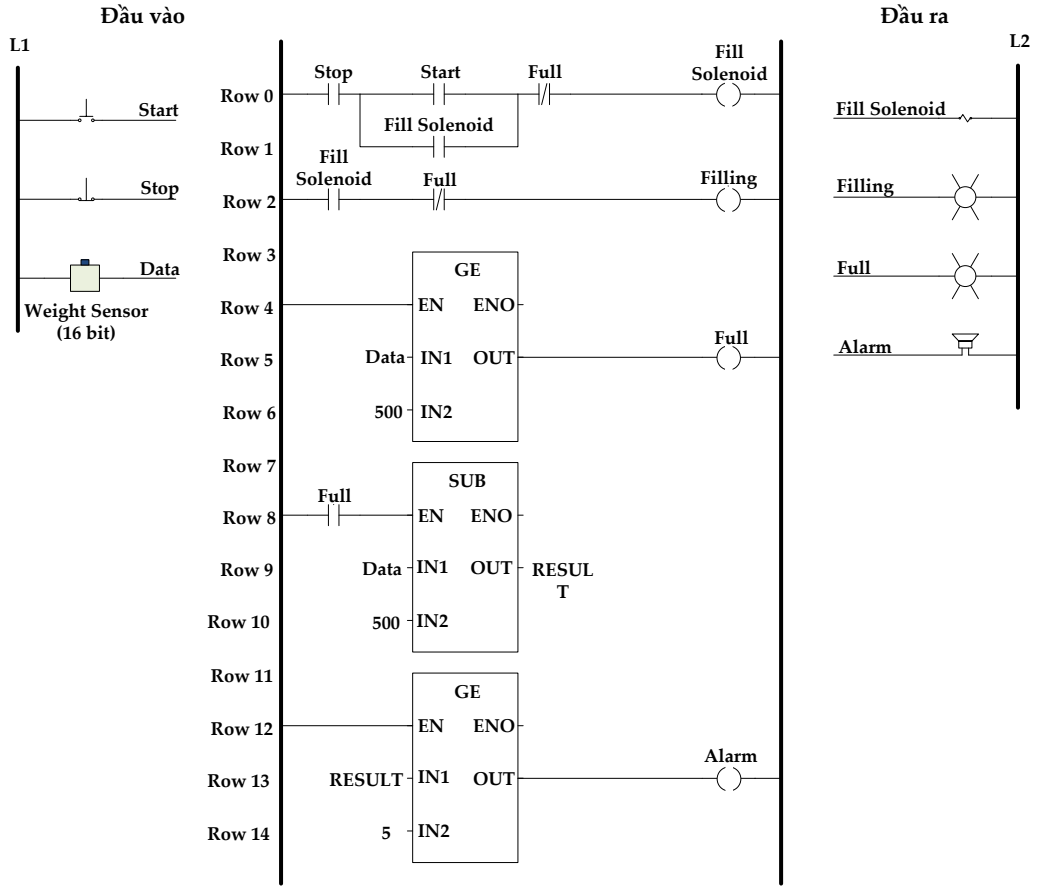
   c. Giả sử bình chứa được nạp nhiên liệu và đạt tới giá trị 502 lb. Trạng thái logic của mỗi bậc trong chương trình sẽ như thế nào?

   d. Giả sử bình chứa được nạp nhiên liệu và đạt tới giá trị 480 lb. Giá trị của các biến sau thế nào?

      (1) Biến Data?

      (2) Biến RESULT?

e. Giả sử bình chứa được nạp nhiên liệu và đạt tới giá trị 510 lb. Trạng thái logic của mỗi bậc trong chương trình sẽ như thế nào?



Hình 4. 120. Chương trình dùng cho Bài 7

8. Hình 4.118 là chương trình điều khiển và giám sát nhiệt độ. Hãy trả lời các câu hỏi sau đây:

   a. Giả sử nhiệt độ mong muốn (Setpoint) là 600^0F khi đó thiết bị nung nhiệt sẽ có trạng thái như thế nào (ON hay OFF)?

   b. Giả sử nhiệt độ mong muốn (Setpoint) là 600^0F và nhiệt độ đo được từ nhiệt điện trở (Data In) là 590^0F. Khi đó giá trị của các biến sau là bao nhiêu:

      (1) Biến Setpoint?

      (2) Biến Data In?

      (3) Biến DIV_RS?

      (4) Biến ADD_RS?

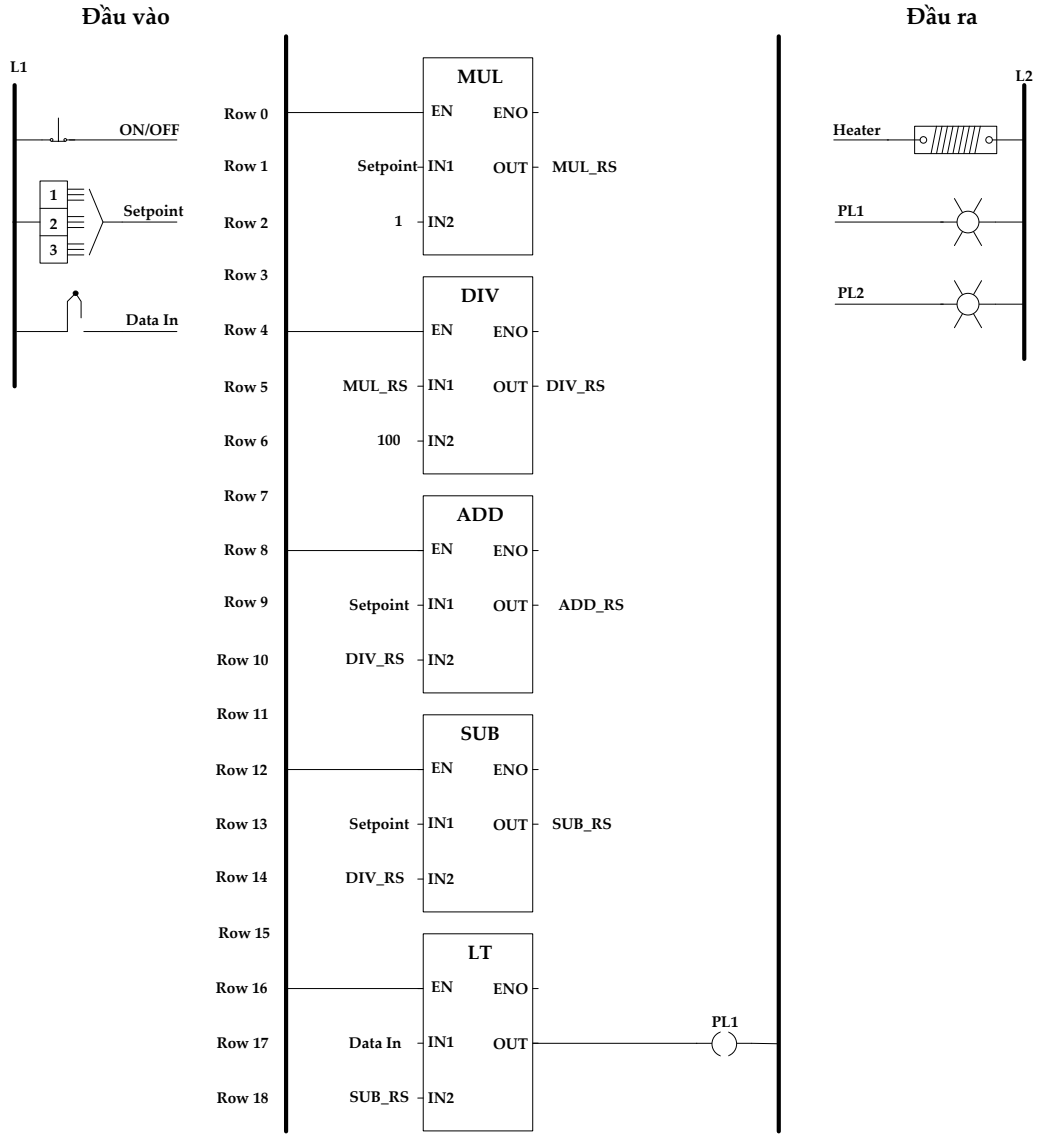
(5) Biến SUB_RS?

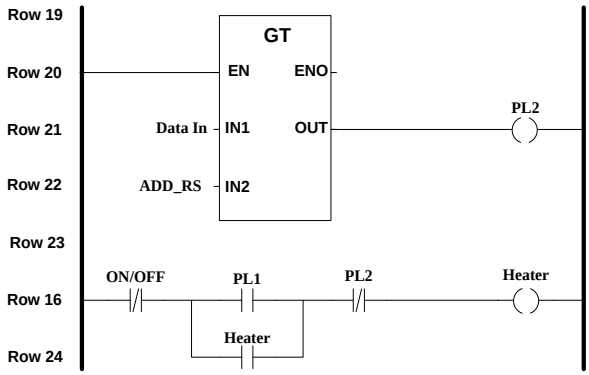
c. Giả sử nhiệt độ mong muốn (Setpoint) là 600^0F và nhiệt độ đo được từ nhiệt điện trở (Data In) là 608^0F. Trạng thái của các đầu ra sau đây như thế nào (ON hay OFF):

(1) Đầu ra PL1?

(2) Đầu ra PL2?

(3) Đầu ra Heater?

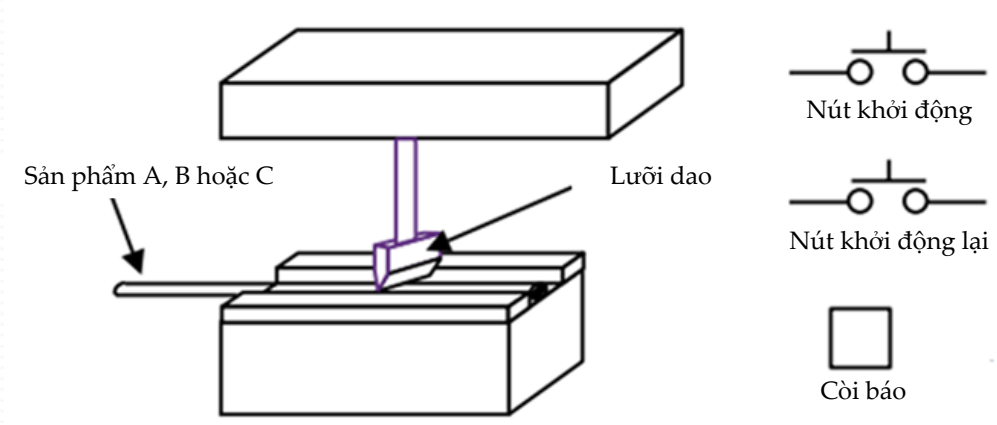




Hình 4.121. Chương trình dùng cho Bài 8

9. Hình 4.119 mô tả hoạt động của một máy cắt với nguyên lý hoạt động như sau:

   - Lưỡi dao được sử dụng để cắt 3 sản loại phẩm A, B, và C trong đó 1000 sản phẩm A, 500 sản phẩm B và 100 sản phẩm C, các sản phẩm được cắt một cách ngẫu nhiên.
   - Sử dụng 3 cảm biến khác nhau cho 3 loại sản phẩm và 1 cảm biến giám sát quá trình cắt hoàn thành.
   - Chuông sẽ kêu khi dao cắt được nhấc lên.
   - Khi nút khởi động được bật, máy bắt đầu hoạt động.



Hình 4.122. Hình dùng cho Bài 9

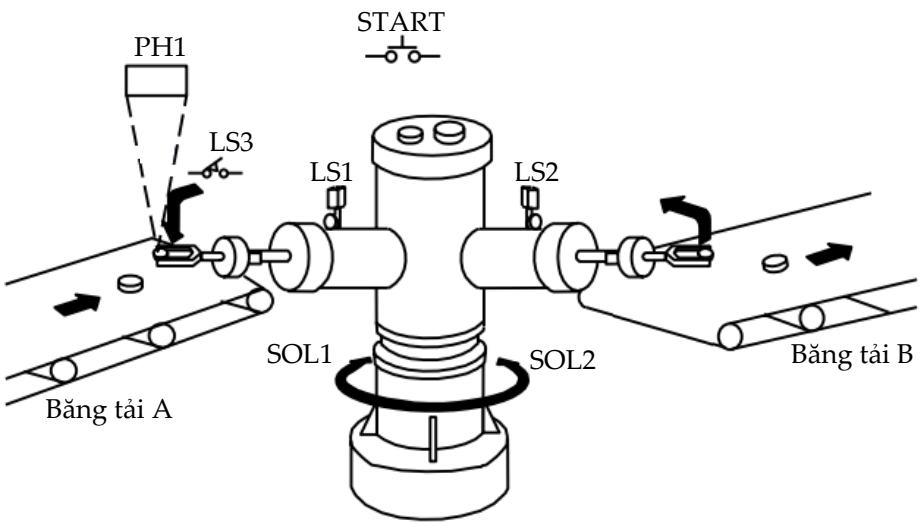
PHẦN 7: THANH GHI DỊCH

1. Hãy sử dụng thanh ghi dịch để viết chương trình để thực hiện các nhiệm vụ dưới đây:

   - Bật đầu ra thứ nhất khi sự kiện thứ nhất xảy ra và duy trì ở trạng thái ON.

- Bật đầu ra thứ hai khi sự kiện thứ hai xảy ra và duy trì ở trạng thái ON.
- Bật đầu ra thứ ba khi sự kiện thứ ba xảy ra và duy trì ở trạng thái ON.
- Bật đầu ra thứ tư khi sự kiện thứ tư xảy ra và duy trì ở trạng thái ON.
- Tất cả các đầu ra sẽ chuyển sang trạng thái OFF khi có một sự kiện đặc biệt xảy ra (chẳng hạn như nút STOP được nhấn).

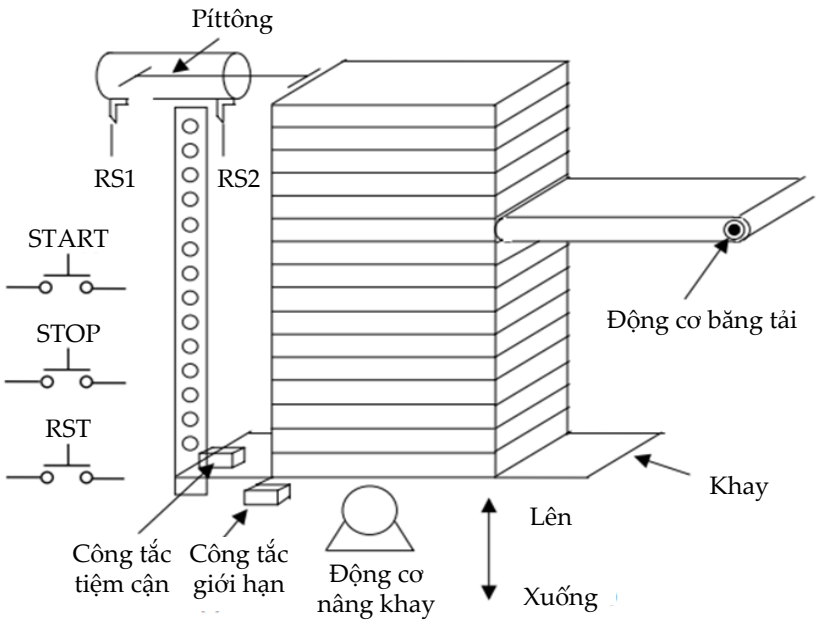
2. Sử dụng thanh ghi dịch để thiết kế chương trình điều khiển vòi phun sơn lên các sản phẩm được di chuyển qua vị trí của vòi phun nhờ một hệ thống băng tải trên cao với yêu cầu như sau:
   - Vòi phun sẽ được kích hoạt khi có sản phẩm xuất hiện tại vị trí nhận biết của vòi phun và bị ngắt khi không có sản phẩm nào.
   - Các sản phẩm được treo trên các móc treo (không phải móc treo nào cũng có sản phẩm).
3. Hình 4.123 là một hệ thống gắp sản phẩm từ băng truyền A sang băng truyền B sử dụng robot tự động. Thiết kế chương trình điều khiển (sử dụng thanh ghi dịch) cho hệ thống với các yêu cầu sau:
   - Khi nút khởi động START được nhấn, cánh tay robot sẽ quay theo chiều thuận kim đồng hồ.
   - Khi di chuyển và phát hiện sản phẩm trên băng truyền A nó sẽ dừng và gắp sản phẩm.
   - Khi đã gắp được sản phẩm nó sẽ quay ngược chiều kim đồng hồ.
   - Khi quay tới vị trí băng truyền B nó sẽ nhả sản phẩm.



Hình 4.123. Hệ thống gắp sản phẩm

4. Hình 4.124 là một hệ thống xếp các sản PCB lên băng tải. Bạn hãy sử dụng thanh ghi dịch để thiết kế chương trình điều khiển cho hoạt động của hệ thống với nguyên lý làm việc như sau:

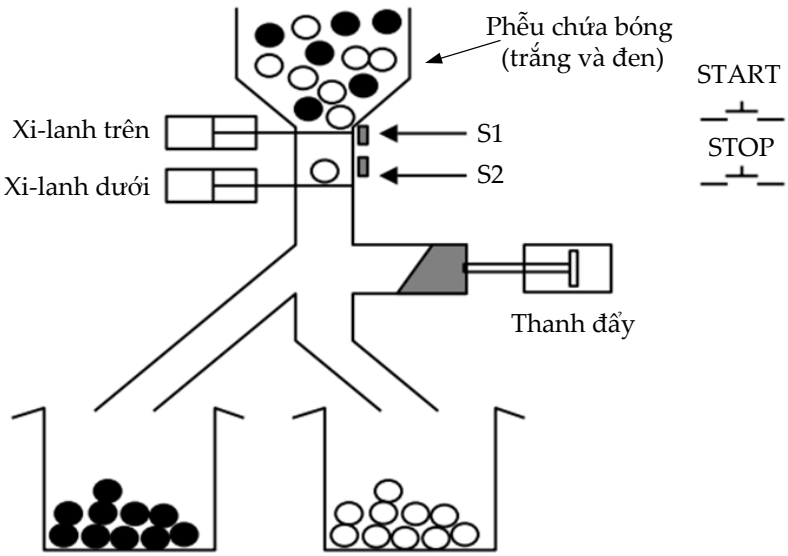
   - Khi nút khởi động START được nhấn, nếu công tắc giới hạn dưới nhận được tín hiệu thì động cơ băng tải và động cơ nâng khay được khởi động.
   - Khay chứa PCB được đẩy lên tới khi công tắc tiệm cận nhận được tín hiệu thì tạm dừng hoạt động của động cơ nâng khay.
   - Kiểm tra trạng thái của RS1 (ON/OFF). Nếu trạng thái là ON thì sẽ điều khiển Píttông dịch sang phải tới khi RS2 chuyển sang trạng thái ON (đẩy PCB lên băng tải).
   - Sau khi PCB được đẩy lên băng tải, Píttông sẽ chuyển động sang trái tới khi trạng thái của RS1 chuyển sang ON và động cơ nâng khay hoạt động trở lại. Toàn bộ quá trình được lặp lại như trên.
   - Khi toàn bộ PCB được đẩy lên băng chuyền, động cơ nâng khay được điều khiển để hạ khay xuống cho tới khi công tắc giới hạn nhận được tín hiệu.
   - Quá trình được bắt đầu lại khi nút khởi động START được nhấn lại lần nữa.



Hình 4.124. Hệ thống xếp sản phẩm PCB

5. Hình 4.127 là hệ thống phân loại bóng theo màu sắc. Bạn hãy thiết kế chương trình điều khiển sử dụng phương pháp thanh ghi dịch với nguyên lý hoạt động như sau:

- Hệ thống được khởi động khi nút START được nhấn.
- Hệ thống sử dụng cảm biến S1 để phát hiện sự xuất hiện của bóng và Xi-lanh trên sẽ mở. Cảm biến S2 để phân biệt màu của bóng để từ đó mở Xi-lanh dưới và di chuyển thanh đẩy một cách thích hợp.
- Hệ thống được dừng lại nhờ nút STOP.



Hình 4.125. Hệ thống phân loại bóng

Chương 5
THIẾT KẾ CHƯƠNG TRÌNH ĐIỀU KHIỂN

Khi lập trình với bất kỳ một ngôn ngữ nào thì việc tiếp cận vấn đề một cách có hệ thống là cần thiết để có thể cải thiện được hiệu quả của chương trình. Thông thường, khi thiết kế một hệ thống chúng ta phải trải qua các bước sau đây:

1. Xét xem hệ thống có những vấn đề gì cần giải quyết và liệt kê các đầu vào/ra cụ thể.
2. Xác định thuật toán sử dụng (xác định các bước giải quyết vấn đề).
3. Các chương trình điều khiển thường rất lớn nên chúng ta sẽ gặp khó khăn trong việc quản lý, tìm và sửa lỗi. Vì vậy, chúng ta nên chia chương trình thành các phần nhỏ.
4. Kiểm tra và gỡ rối chương trình.

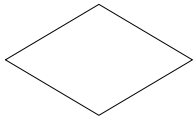
5.1. THIẾT KẾ CHƯƠNG TRÌNH

5.1.1. Thiết kế chương trình sử dụng lưu đồ thuật toán

5.1.1.1. Giới thiệu

Lưu đồ thuật toán là một ý tưởng giải quyết bài toán thông qua một chuỗi các bước xử lý và được thực hiện theo một trình tự nhất định. Những khối trong lưu đồ thuật toán được nối với nhau bằng các mũi tên để chỉ ra tính liên tục giữa các bước thực hiện. Các khối có hình dạng khác nhau dùng để biểu thị các hành động khác nhau. Mỗi chương trình luôn cần một khối bắt đầu (Start Block). Thông thường các chương trình PLC hiếm khi kết thúc vì vậy trong các chương trình sẽ không có khối kết thúc (Stop Block). Bên cạnh đó còn có các khối quan trọng khác đó là các khối xác định điều kiện và đưa ra các quyết định điều khiển.

Bảng 5.1. Một số ký hiệu sử dụng khi lập lưu đồ thuật toán

Ký hiệu	Diễn giải
	Bắt đầu/kết thúc chương trình
	Luồng xử lý
	Điều khiển lựa chọn
	Nhập/xuất dữ liệu
	Xử lý, tính toán hoặc gán
	Hàm con

Dưới đây là các bước xây dựng một lưu đồ thuật toán điều khiển:

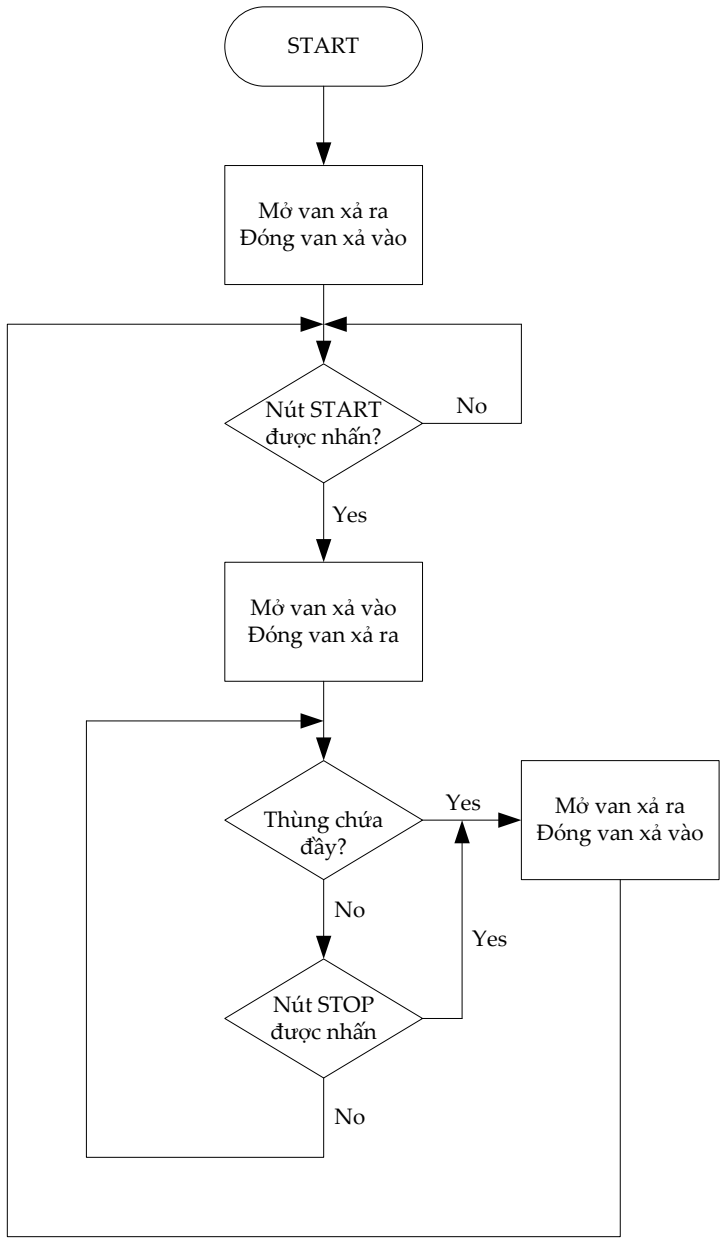
1. Hiểu được quá trình cần điều khiển.
2. Xác định được các hoạt động chính để vẽ các khối biểu diễn.
3. Xác định được chuỗi các bước thực hiện để vẽ chiều mũi tên.
4. Sử dụng khối kiểm tra khi cần rẽ nhánh chương trình.

Hình 5.1 là lưu đồ thuật toán điều khiển hệ thống bể chứa nước với nguyên lý hoạt động như sau:

- Khi khởi động, mở van xả ra, đóng van xả vào, và nước được bơm ra.
- Khi nút khởi động START được nhấn, đóng van xả ra, mở van xả vào, và nước được bơm vào bể chứa.
- Khi bể chứa đã đầy hoặc nút dừng STOP được nhấn sẽ mở van xả ra, đóng van xả vào.

Chương trình sẽ được thực hiện theo thứ tự từ trên xuống dưới. Khối biểu diễn hoạt động điều khiển được sử dụng để biểu diễn hoạt động đóng van xả ra và mở van xả vào. Tiếp theo, khối kiểm tra điều kiện để giám sát trạng thái nút

dừng STOP được nhấn. Khi nút này được nhấn, chương trình sẽ thực hiện theo nhánh "yes", van xả vào được đóng và van xả ra được mở. Chương trình sẽ đi vào thực hiện vòng lặp khi nút dừng STOP được nhấn hoặc thùng chứa đã đầy nước. Nếu xảy ra một trong hai trường hợp này thì van xả vào sẽ đóng và van xả ra sẽ mở. Hệ thống sẽ quay lại trạng thái đợi nút khởi động START được nhấn để quá trình điều khiển được lặp lại. Bộ điều khiển chỉ cần được khởi động một lần vì vậy trong lưu đồ ta thấy chỉ có một khối bắt đầu. Đối với những người mới làm quen sẽ thường bỏ qua không quan tâm tới việc kiểm tra trạng thái nút nhấn dừng quá trình STOP.

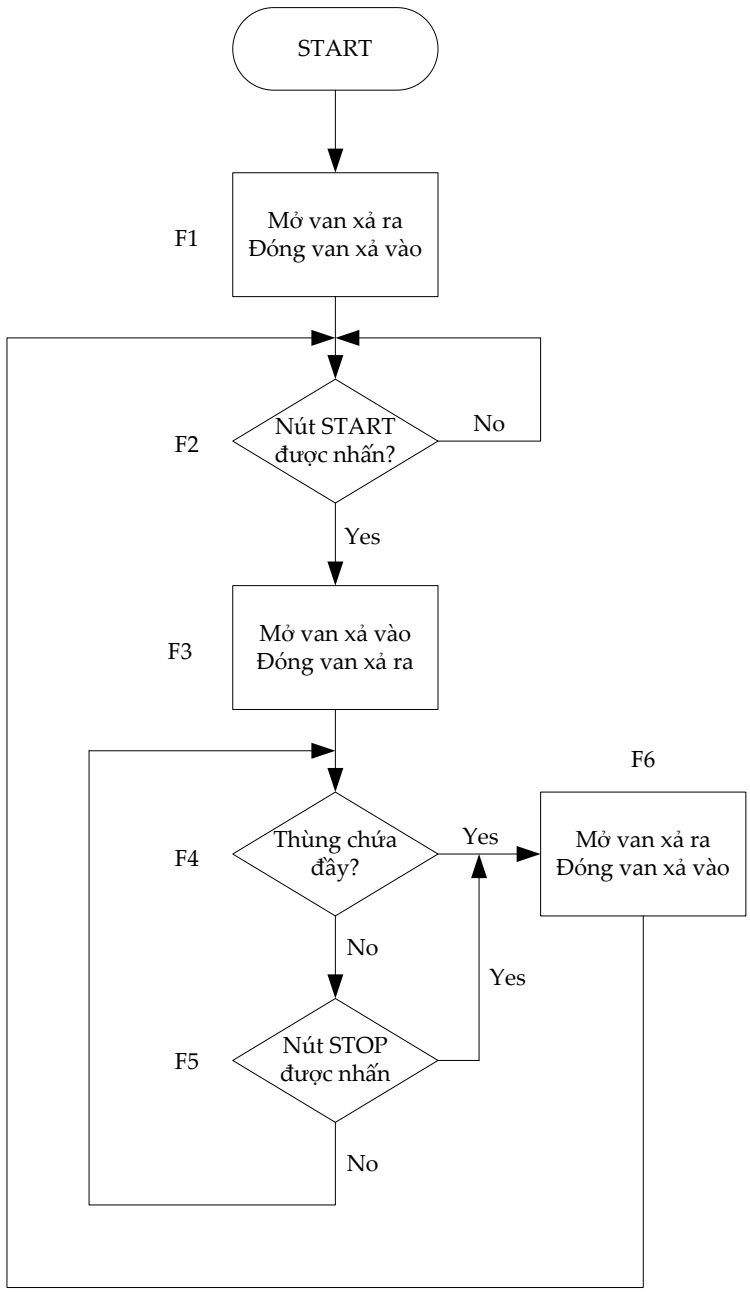


Hình 5.1. Lưu đồ thuật toán điều khiển bể chứa nước

5.1.1.2. Phương pháp chuyển lưu đồ thuật toán sang sơ đồ bậc thang sử dụng khối logic

Sau khi thiết kế được lưu đồ thuật toán điều khiển thì việc tiếp theo phải làm đó là chuyển lưu đồ thuật toán sang sơ đồ bậc thang. Để làm được điều đó việc đầu tiên chúng ta phải làm đó là đặt tên cho mỗi khối trong lưu đồ thuật toán và sau đó sẽ chuyển mỗi khối đó sang sơ đồ bậc thang.

Với lưu đồ thuật toán điều khiển bể chứa nước như Hình 5.1, chúng ta sẽ đặt tên cho mỗi khối như Hình 5.2.

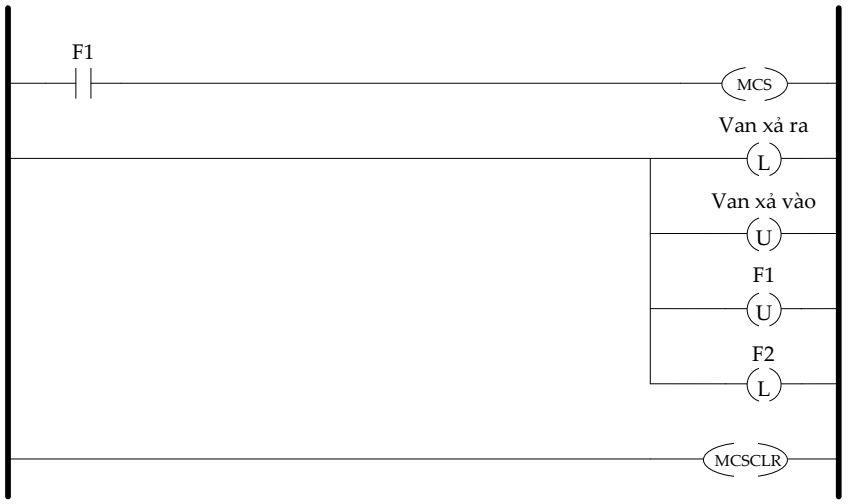


Hình 5.2. Đặt tên cho các khối trong lưu đồ thuật toán

Công việc tiếp theo chúng ta phải làm đó là thiết lập điều kiện ban đầu cho hệ thống như Hình 5.3. Trong đó, các đầu ra có ký hiệu L (Latched) là đầu ra được chốt, còn các đầu ra có ký hiệu U (Unlatched) là các đầu ra không chốt. Các giá trị khởi tạo này chỉ đúng trong lần quét đầu tiên của PLC.

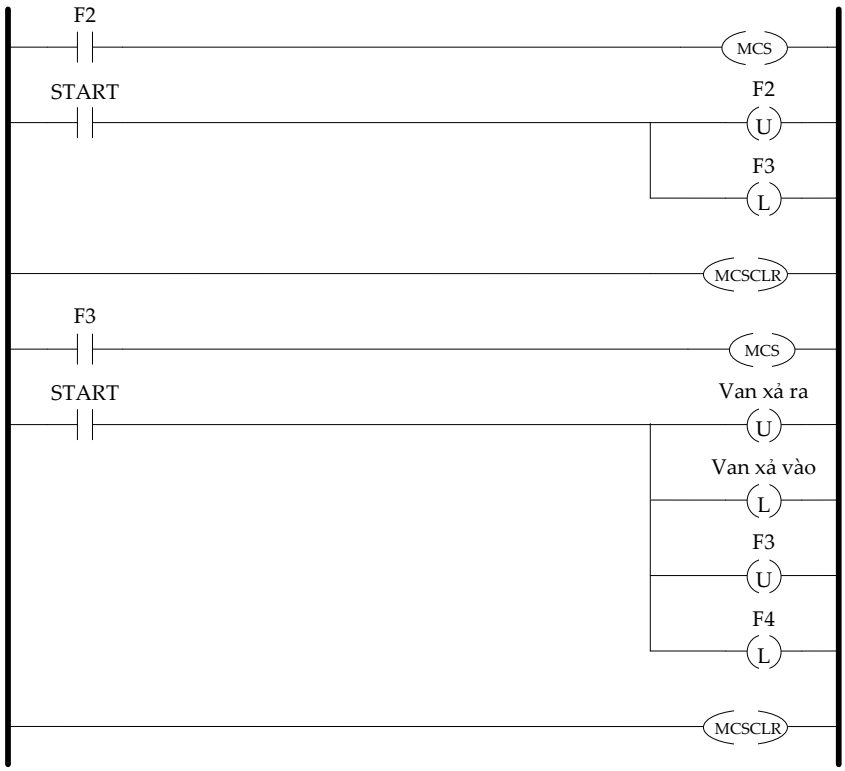
Hình 5.3. Khởi tạo trạng thái ban đầu cho các khối

Sau khi đã khởi tạo trạng thái ban đầu cho các khối, chúng ta sẽ chuyển đổi các khối sang sơ đồ bậc thang. Sơ đồ bậc thang cho khối F1 như Hình 5.4. Khi F1 có trạng thái logic là TRUE thì các bậc thang nằm giữa các lệnh MCS và MCSCLR được thực hiện, van xả ra được mở và đóng van xả vào, F1 được ngắt, và kích hoạt F2. Ngược lại, nếu F1 có trạng thái logic là FLASE thì các bậc thang này sẽ không được thực hiện.



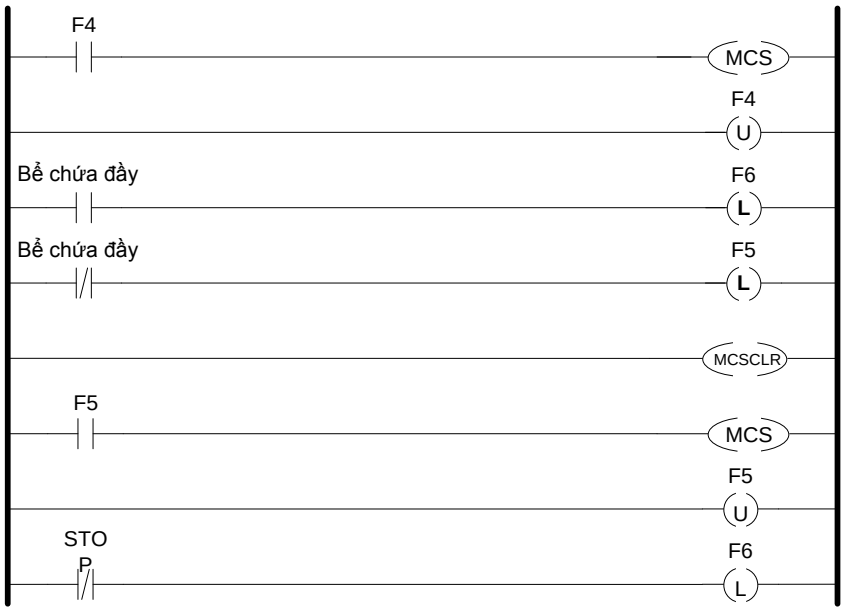
Hình 5.4. Sơ đồ bậc thang cho hoạt động của F1

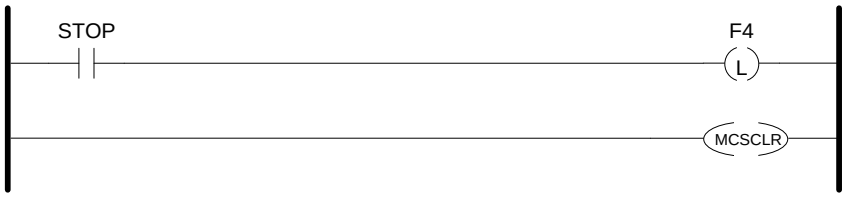
Sơ đồ bậc thang cho khối F2 khá đơn giản. Khi nút khởi động START được nhấn thì F2 được ngắt và F3 được kích hoạt. Sơ đồ bậc thang cho khối F3 sẽ là mở van xả vào, đóng van xả ra, và kích hoạt F4.



Hình 5.5. Sơ đồ bậc thang cho hoạt động của F2 và F3

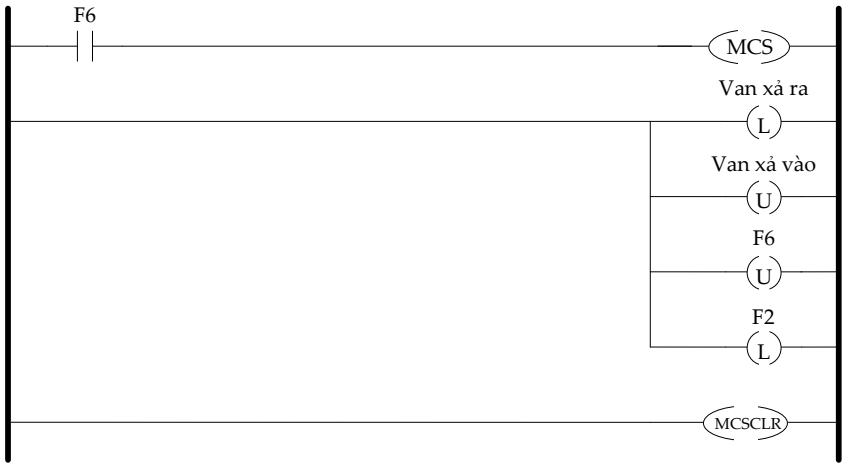
Sơ đồ bậc thang viết cho khối F4 sẽ là ngắt khối F4 và nếu như bể chứa đầy thì kích hoạt khối F6, ngược lại sẽ kích hoạt khối F5.





Hình 5.6. Sơ đồ bậc thang cho hoạt động của F4 và F5

Khối F6 thực hiện chức năng mở van xả ra, đóng van xả vào, kết thúc hoạt động của khối F6, và khởi động khối F2.



Hình 5.7. Sơ đồ bậc thang hoạt động của F6

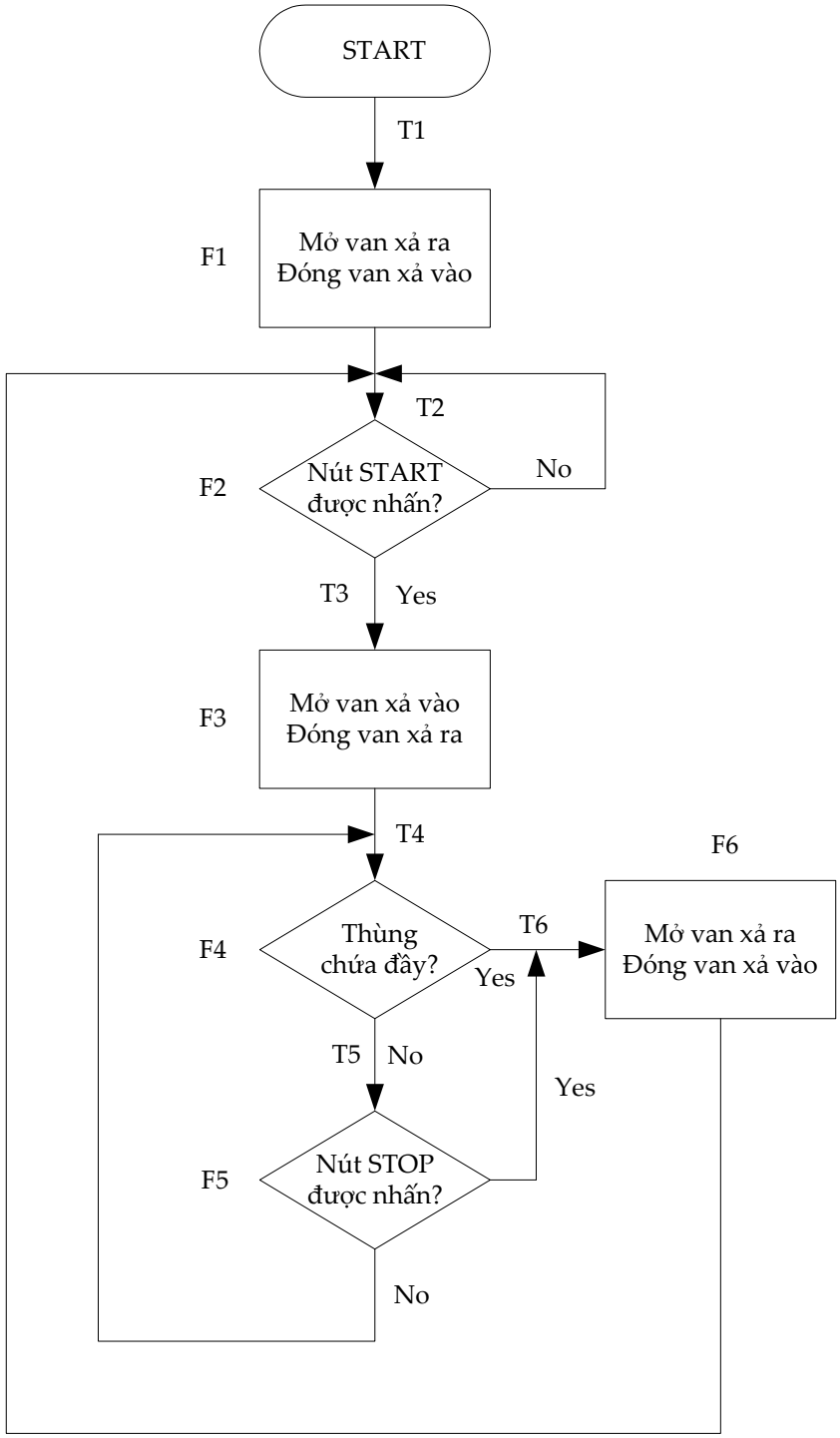
5.1.1.3. Phương pháp chuyển lưu đồ thuật toán sang sơ đồ bậc thang sử dụng bit tuần tự

Ngoài phương pháp sử dụng kết hợp hai câu lệnh MCS và MCSCLR, chúng ta có thể sử dụng phương pháp bit tuần tự như được miêu tả dưới đây để chuyển lưu đồ thuật toán sang sơ đồ bậc thang. Tương tự như phương pháp trước, chúng ta cũng sẽ đặt tên cho các khối trong lưu đồ thuật toán và điểm khác biệt đó là chúng ta cũng sẽ đặt tên cho các quá trình chuyển đổi (biểu thị bằng các mũi tên) từ khối này sang khối khác.

Sử dụng ví dụ điều khiển bể chứa nước với nguyên lý hoạt động như sau:

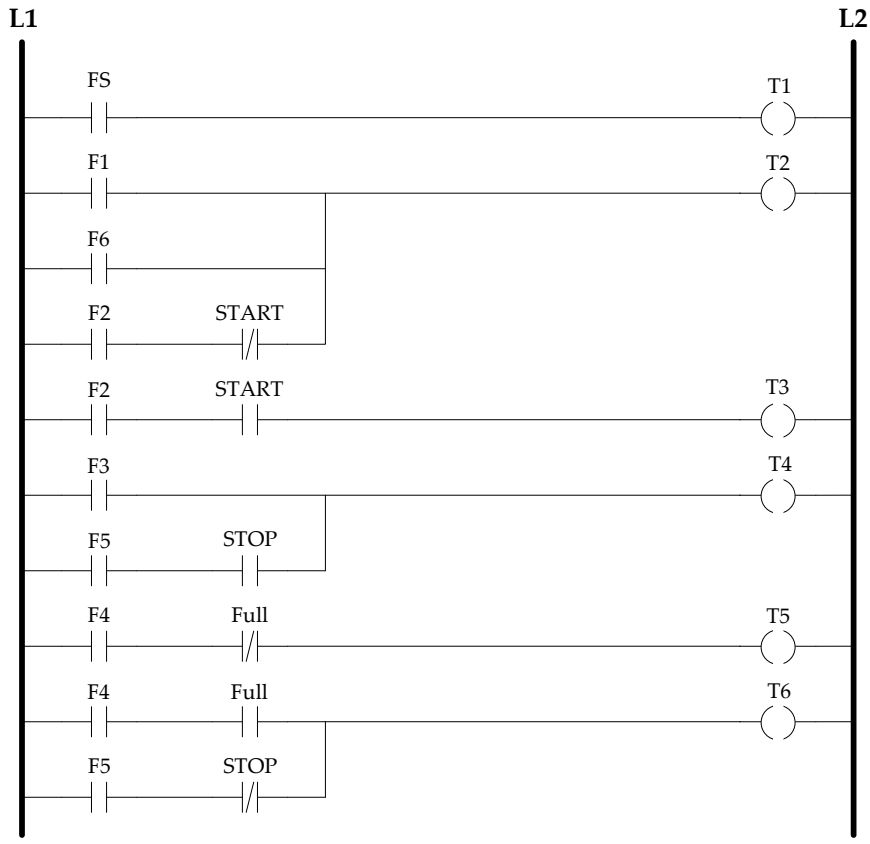
- Khi khởi động, mở van xả ra, đóng van xả vào, và nước được bơm ra.
- Khi nút khởi động START được nhấn, đóng van xả ra, mở van xả vào, và nước được bơm vào bể chứa.
- Khi bể chứa đã đầy hoặc nút dừng STOP được nhấn sẽ mở van xả ra, đóng van xả vào.

Hình 5.8 là lưu đồ thuật toán điều khiển sau khi đã đặt tên cho các khối và các quá trình chuyển đổi.

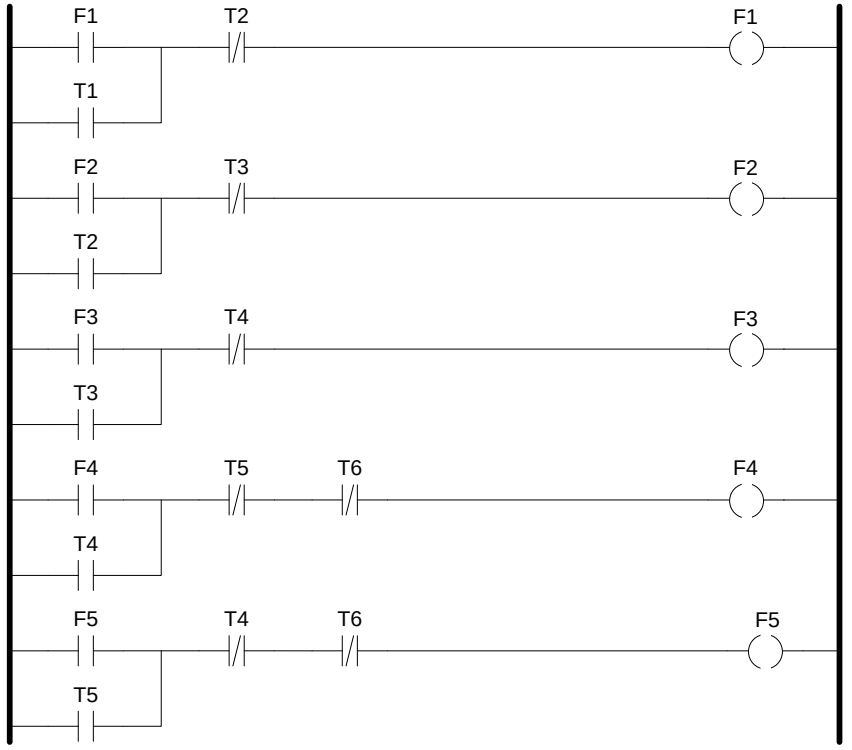


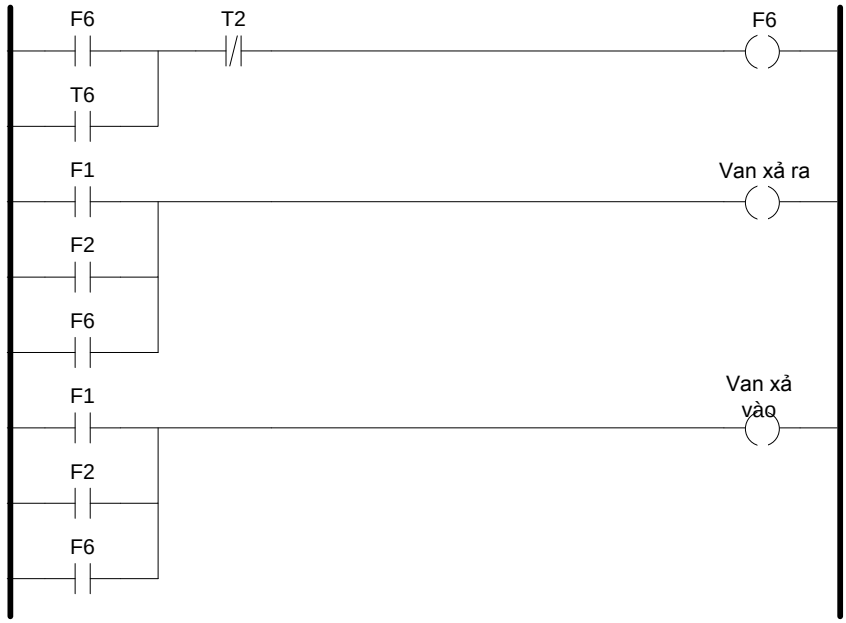
Hình 5.8. Đặt tên cho các khối và sự chuyển đổi trạng thái trong sơ đồ thuật toán

Hình 5.9 mô tả điều kiện xảy ra các chuyển đổi và Hình 5.10 là chương trình thực hiện các công việc điều khiển. Chúng ta nên thực hiện các phép chuyển đổi trạng thái trước khi các trạng thái được thực hiện.



Hình 5.9. Quá trình chuyển đổi trạng thái logic



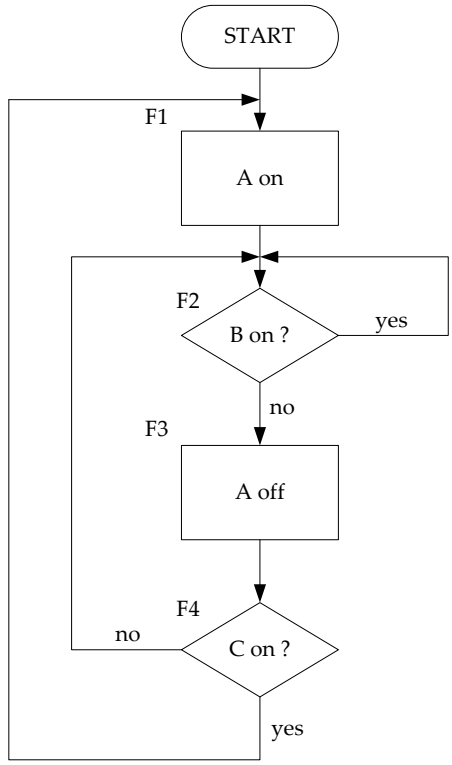


Hình 5.10. Thực hiện chức năng logic và các đầu ra

5.1.1.4. Một số ví dụ áp dụng

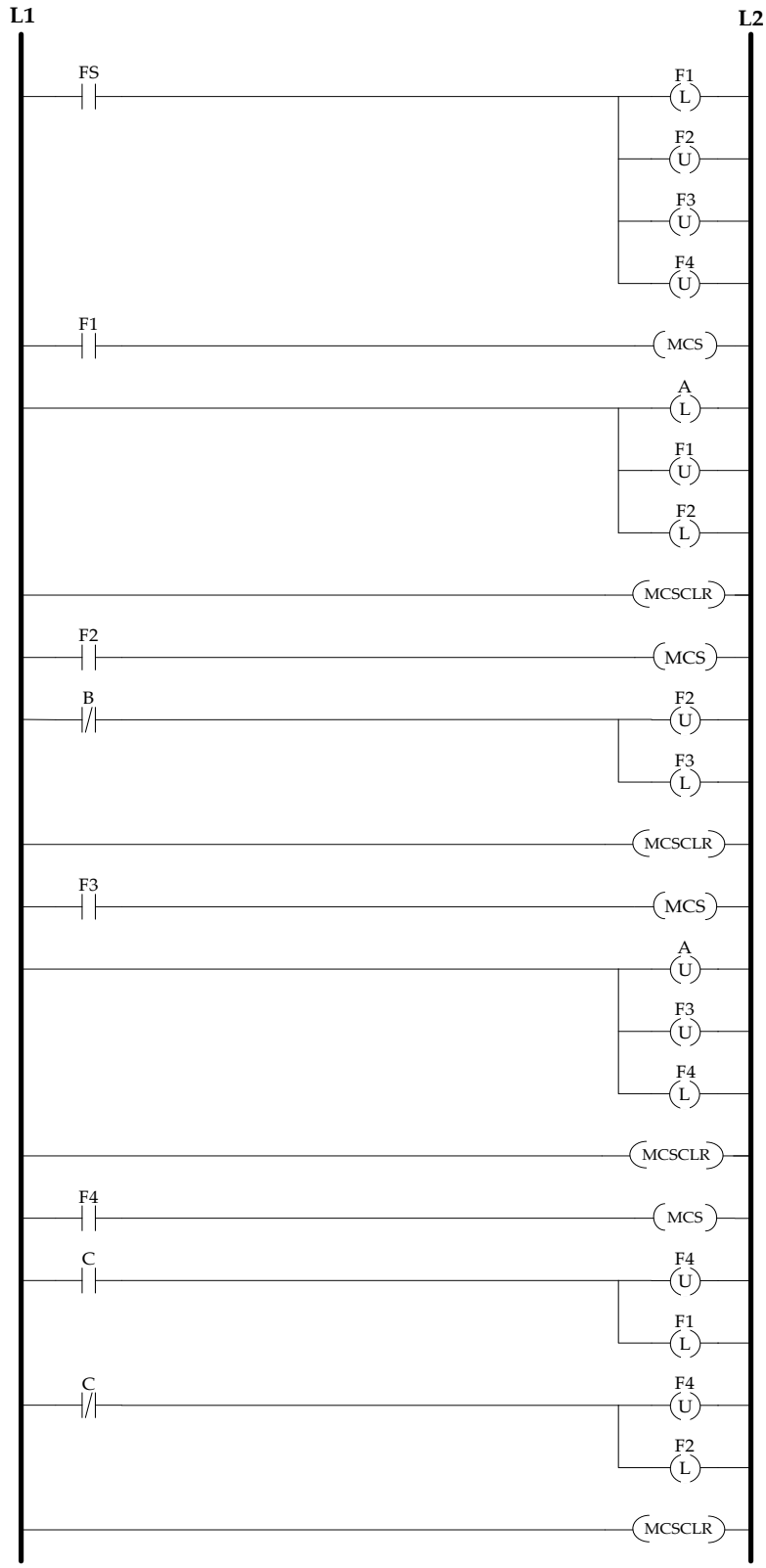
Ví dụ 1:

Chuyển đổi lưu đồ thuật toán sau sang sơ đồ bậc thang:



Hình 5.11. Lưu đồ thuật toán Ví dụ 1

Sơ đồ logic bậc thang:



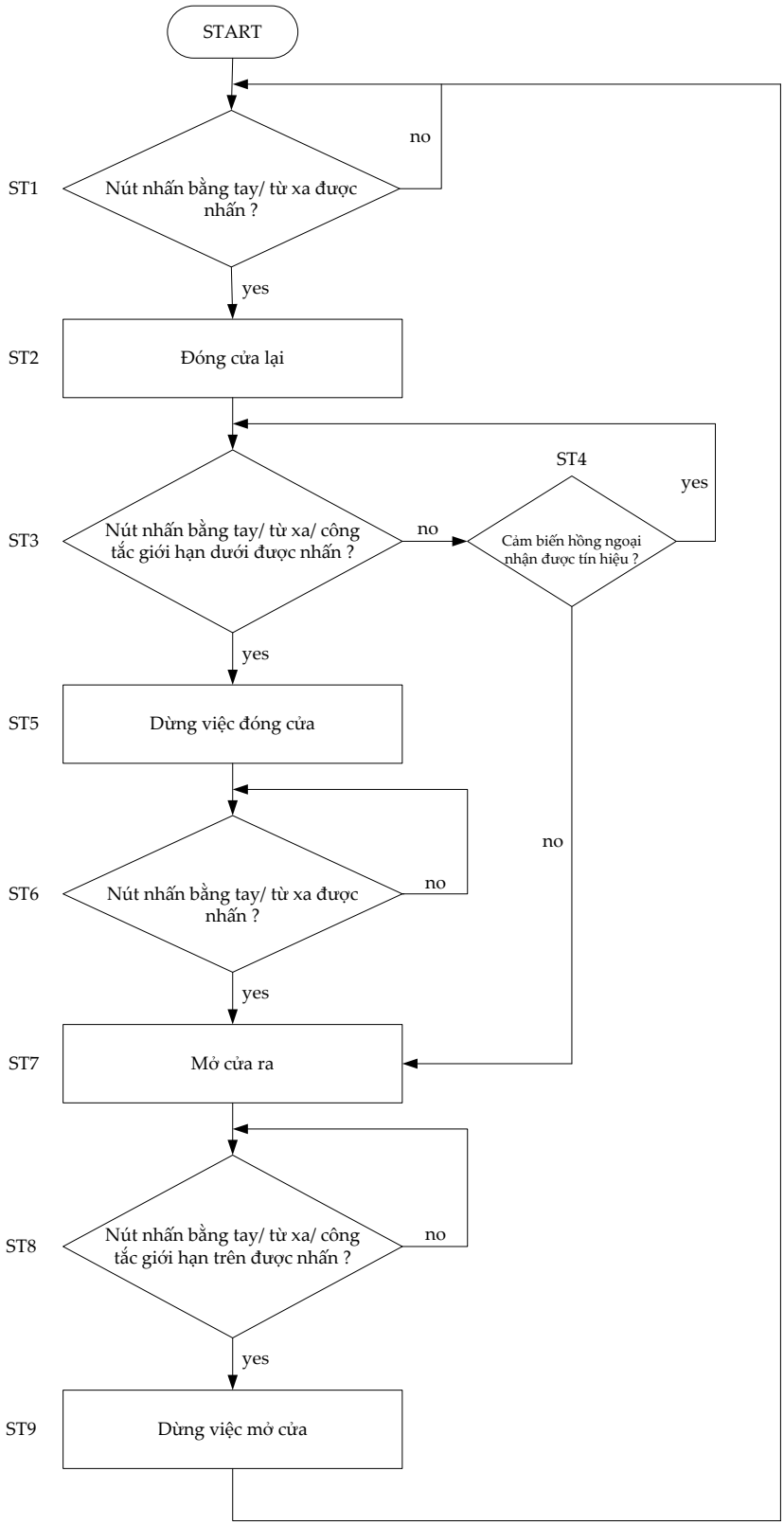
Hình 5.12. Chương trình cho Ví dụ 1

Ví dụ 2:

Xây dựng lưu đồ thuật toán và chuyển sang sơ đồ bậc thang cho bộ điều khiển đóng/mở cửa gara ôtô với hoạt động như sau:

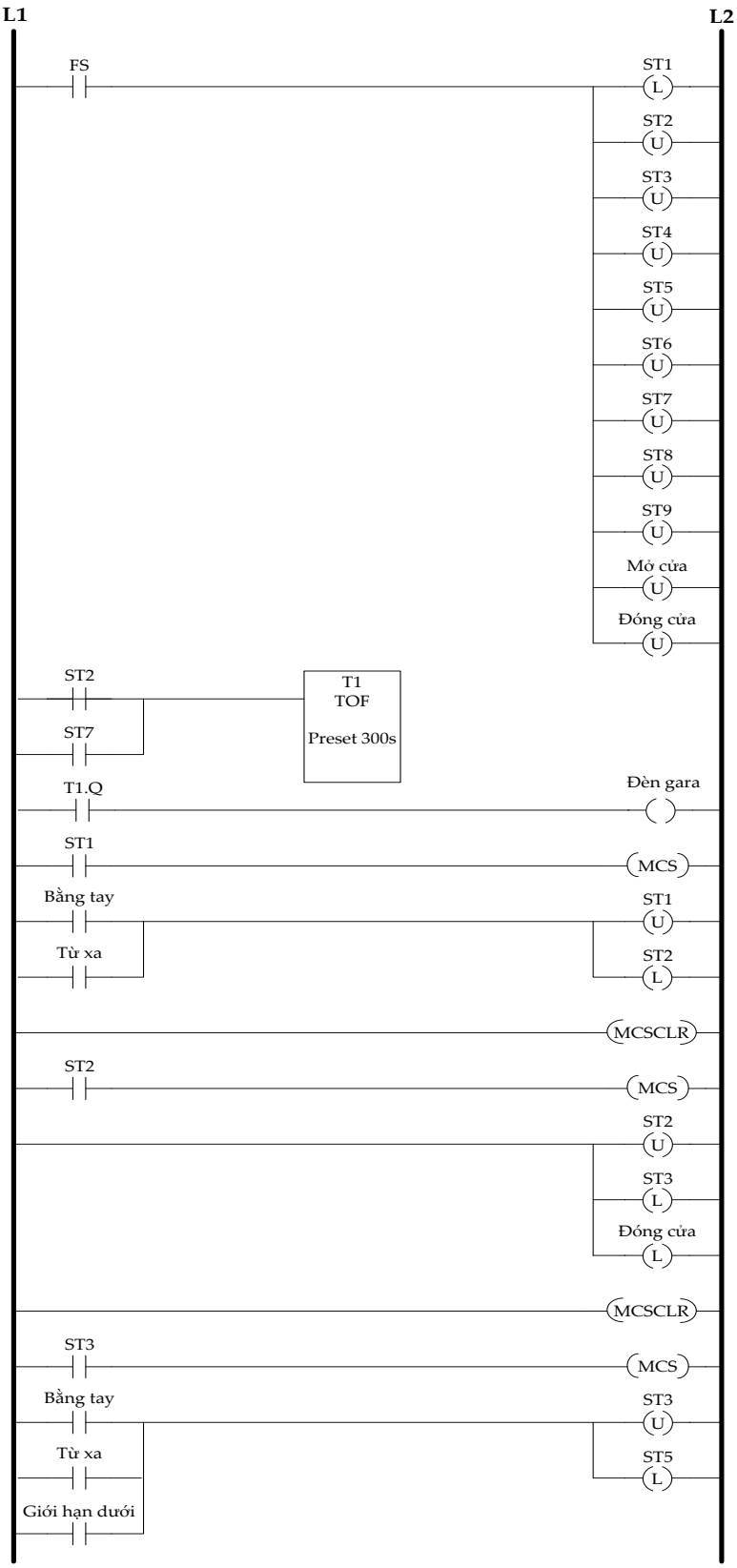
- Có một nút nhấn điều khiển bằng tay và một nút nhấn điều khiển từ xa.
- Khi nhấn nút, cửa sẽ di chuyển lên hoặc xuống.
- Khi cửa đang di chuyển mà nút nhấn được nhấn nó sẽ dừng lại và nếu được nhấn thêm lần nữa nó sẽ di chuyển theo chiều ngược lại.
- Có các công tắc giới hạn trên và dưới để dừng động cơ khi đạt tới giới hạn trên hoặc dưới.
- Tại vị trí cửa ra vào có gắn cảm biến hồng ngoại. Nếu chuỗi hồng ngoại bị ngắt trong khi cửa đang đóng lại thì cửa sẽ được dừng và di chuyển theo chiều ngược lại.
- Đèn báo sẽ sang trong 5 phút khi cửa hoạt động (đóng lại hoặc mở ra).

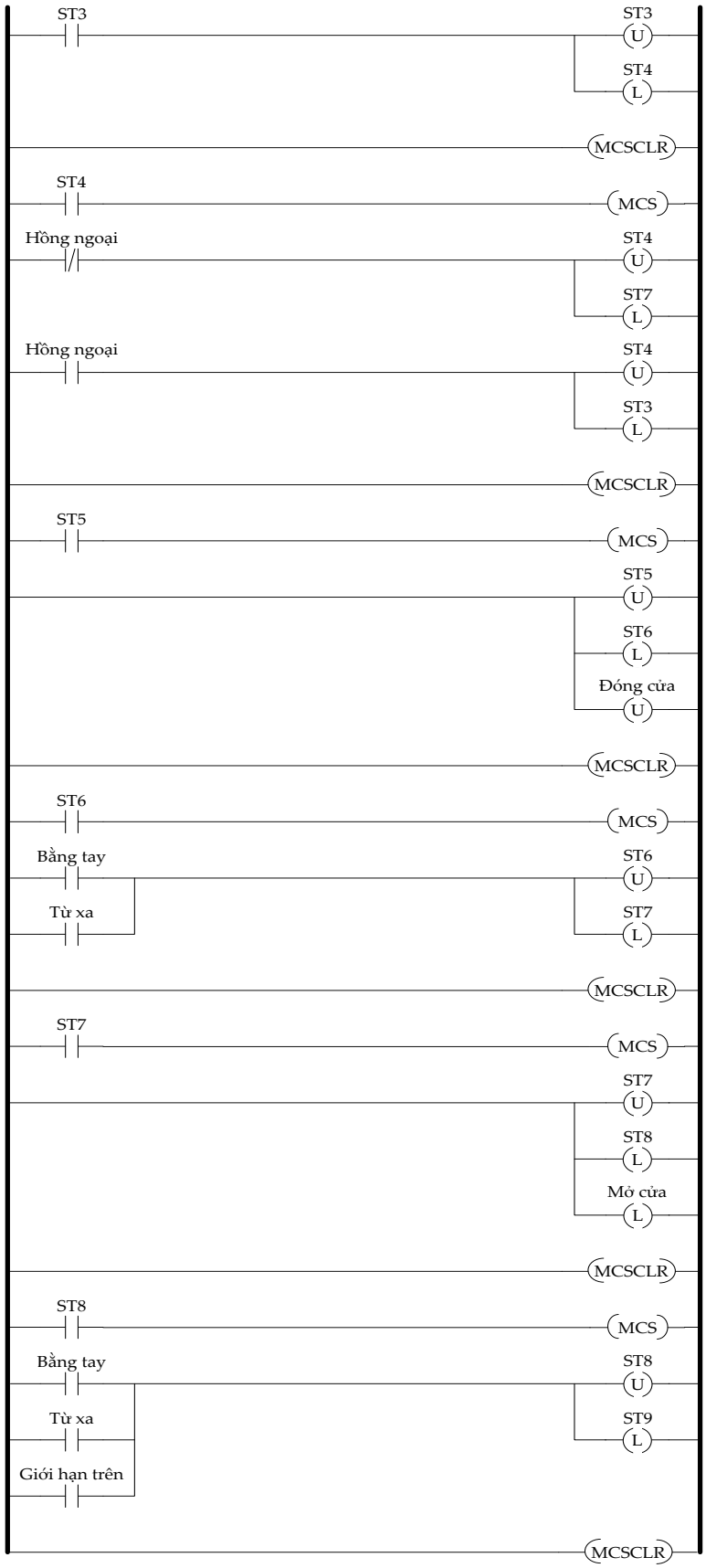
Lưu đồ thuật toán:

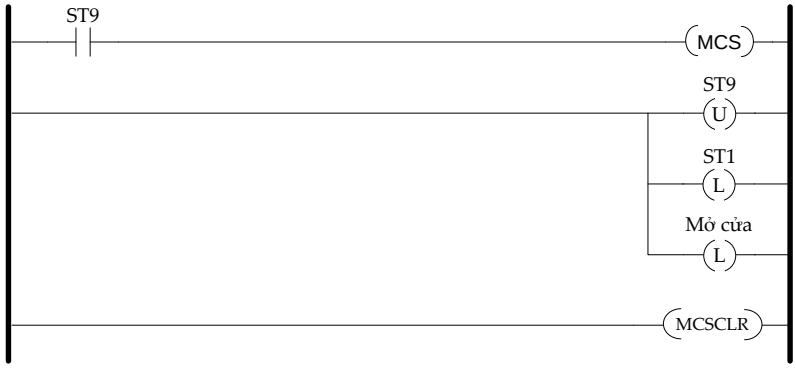


Hình 5.13. Lưu đồ thuật toán cho ví dụ 2

Sơ đồ bậc thang:




ST3
ST3
U
ST4
L
MCSCLR
ST4
MCS
Hồng ngoại
ST4
U
ST7
L
Hồng ngoại
ST4
U
ST3
L
MCSCLR
ST5
MCS
ST5
U
ST6
L
Đóng cửa
U
MCSCLR
ST6
MCS
Bằng tay
ST6
U
Từ xa
ST7
L
MCSCLR
ST7
MCS
ST7
U
ST8
L
Mở cửa
L
MCSCLR
ST8
MCS
Bằng tay
ST8
U
Từ xa
ST9
L
Giới hạn trên
MCSCLR



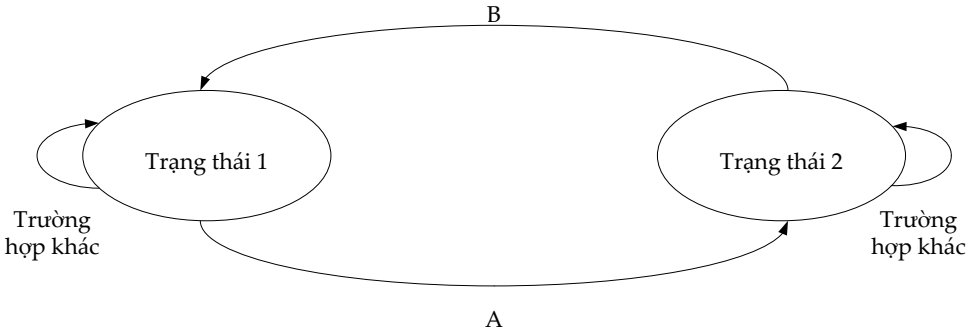
Hình 5.14. Sơ đồ bậc thang cho Ví dụ 2

5.1.2. Thiết kế chương trình sử dụng sơ đồ trạng thái

5.1.2.1. Giới thiệu

Mỗi một trạng thái là một chế độ hoạt động của hệ thống. Giả sử chúng ta xét hoạt động của một cây ATM được sử dụng để rút tiền. Thông thường các trạng thái hoạt động của máy có thể là trạng thái nghỉ, quét thẻ, xác nhận mã thẻ, xác nhận loại hình giao dịch, yêu cầu nhập số tiền, đếm số lượng tiền, trả tiền/trả thẻ và sau đó quay lại trạng thái nghỉ.

Thiết kế chương trình điều khiển dựa vào sơ đồ trạng thái có thể được mô tả cùng với các trạng thái hoạt động và quá trình chuyển đổi giữa các trạng thái của hệ thống. Ví dụ, xét sơ đồ trạng thái trên Hình 5.15 với 2 trạng thái hoạt động. Nếu hệ thống đang ở trạng thái 1 mà điều kiện A đúng thì hệ thống sẽ chuyển sang trạng thái 2; ngược lại hệ thống vẫn ở trạng thái 1. Tương tự nếu hệ thống đang ở trạng thái 2 mà điều kiện B đúng thì nó sẽ chuyển sang trạng thái 1; ngược lại nó vẫn sẽ ở trạng thái 2.



Hình 5.15. Sơ đồ trạng thái với hai trạng thái hoạt động

Để xây dựng được sơ đồ trạng thái cho hệ thống, chúng ta phải xem xét một số vấn đề sau đây:

– Hệ thống làm những công việc gì?

– Hệ thống có những thay đổi nào?

– Những tác động nào có thể làm thay đổi hệ thống?

Ví dụ, thiết kế chương trình điều khiển đơn giản cho máy bán Coffee tự động. Trước tiên, chúng ta cần nhận dạng được các trạng thái hoạt động của hệ thống. Trạng thái chủ đạo của máy chính là trạng thái nghỉ. Tiếp theo là trạng thái cho tiền vào máy. Khi tiền đã được cho đủ vào máy thì người sử dụng có thể lựa chọn thức uống. Sau khi đã chọn thức uống xong, máy bắt đầu pha chế theo yêu cầu. Và cuối cùng là các dịch vụ được yêu cầu (coffee, cups). Nếu xảy ra lỗi, hệ thống sẽ đưa ra thông báo.

Các trạng thái:

– Trạng thái nghỉ: Máy không có tiền và sẽ không làm gì.

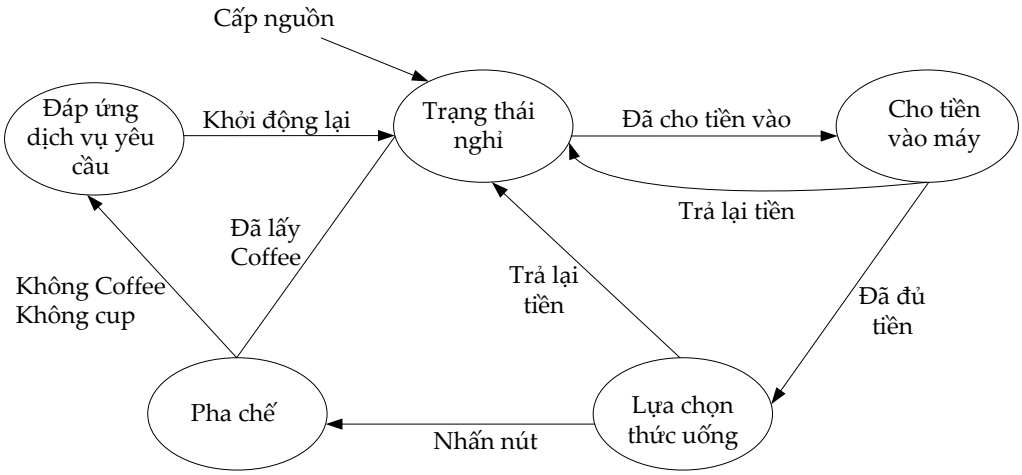
– Cho tiền vào máy: Tiền sẽ được cho vào máy.

– Lựa chọn thức uống: Khi đã cho đủ tiền vào máy thì người mua có thể chọn thức uống.

– Pha chế: Máy sẽ tự động pha chế thức uống đã được chọn.

– Đáp ứng dịch vụ yêu cầu: Coffee, cups hay thông báo lỗi.

Các trạng thái hoạt động được biểu diễn như Hình 5.16. Chúng ta có thể thấy rằng khi cấp nguồn, máy sẽ được đưa vào trạng thái nghỉ. Quá trình chuyển trạng thái chuyển phụ thuộc vào trạng thái của các đầu vào và các cảm biến trong máy.

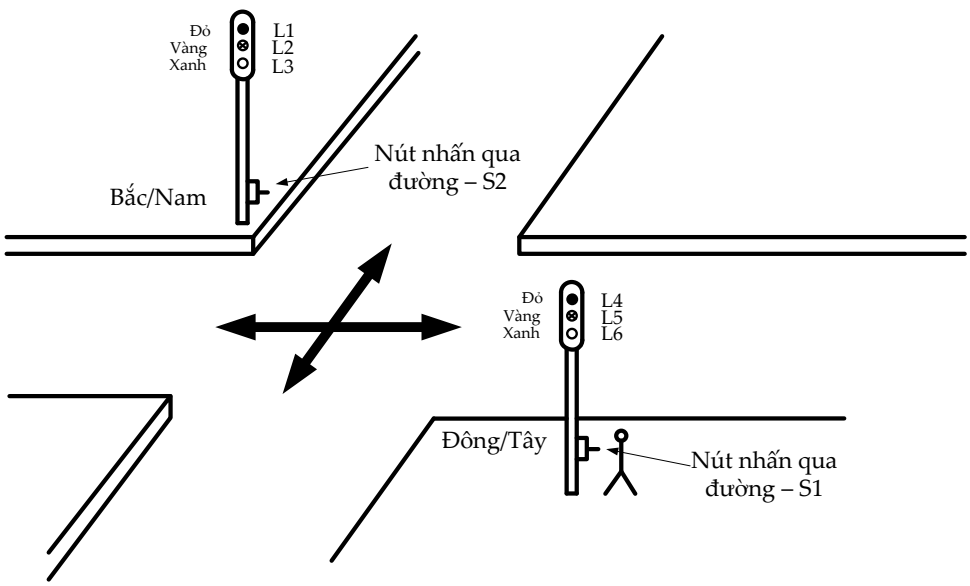


Hình 5.16. Sơ đồ trạng thái máy bán Coffee tự động

5.1.2.2. Thiết kế chương trình điều khiển sử dụng sơ đồ trạng thái

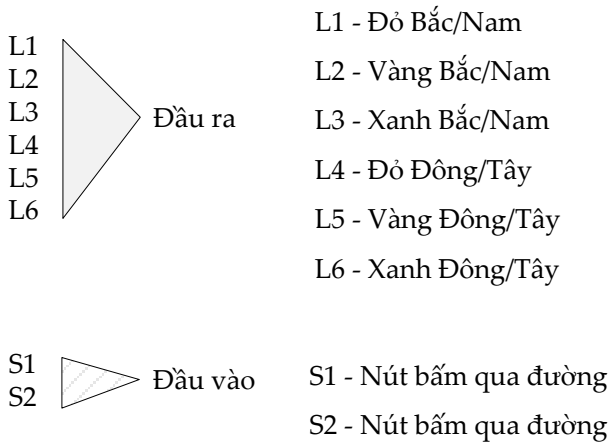
Ví dụ, xét một hệ thống đèn giao thông như Hình 5.17. Theo một hướng nào đó (Bắc/Nam hay Đông/Tây), trước tiên đèn xanh sẽ được bật trong một khoảng

thời gian (10s hoặc lâu hơn). Sau đó là đèn vàng (khoảng 4s). Tiếp theo đó là một chuỗi hoạt động giống như trên nhưng theo hướng còn lại. Như vậy, khi đèn xanh và đèn vàng tại một hướng đang sáng thì hướng còn lại sẽ là đèn đỏ. Trong hệ thống cũng có các nút nhấn dành cho người đi bộ khi muốn qua đường. Khi các nút này được nhấn thì đèn báo sẽ được bật và thời gian sáng của đèn xanh sẽ được tăng lên.



Hình 5.17. Hệ thống đèn giao thông

Trước tiên, chúng ta cần định nghĩa các biến đầu vào và đầu ra cho hệ thống như Hình 5.18. Các biến này thay đổi khi hệ thống chuyển từ trạng thái này sang trạng thái khác. Các biến đầu vào được sử dụng để định nghĩa các quá trình thay đổi trạng thái. Các biến đầu ra được sử dụng để định nghĩa trạng các thái hoạt động hệ thống.



Hình 5.18. Đầu vào/ra cho hệ thống điều khiển đèn giao thông

Chúng ta có thể sử dụng bảng trạng thái như Bảng 5.2 dưới đây để định nghĩa hệ thống.

Bảng 5.2. Bảng trạng thái cho hệ thống điều khiển đèn giao thông

Trạng thái của hệ thống						
L1	L2	L3	L4	L5	L6	
Trạng thái bit						0 – Đèn tắt
						1 – Đèn sáng

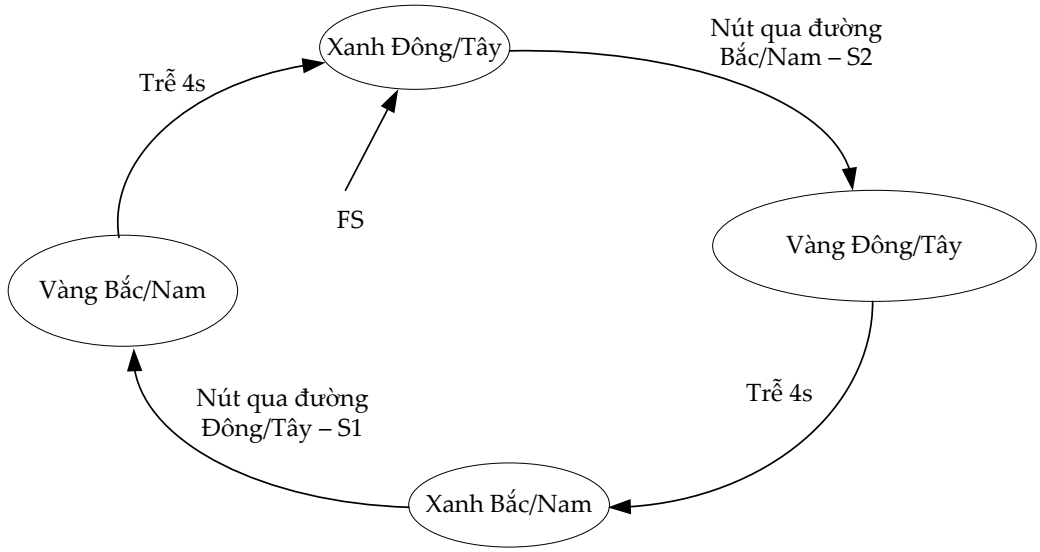
Bảng trạng thái							
Mô tả trạng thái	#	L1	L2	L3	L4	L5	L6
Xanh Đông/Tây	1	1	0	0	0	0	1
Vàng Đông/Tây	2	1	0	0	0	1	0
Xanh Bắc/Nam	3	0	0	1	1	0	0
Vàng Bắc/Nam	4	0	1	0	1	0	0

Tương tự, các quá trình biến đổi có thể được biểu diễn thành bảng trạng thái như Bảng 5.3. Quá trình chuyển đổi từ đèn xanh sang đèn vàng theo hướng Đông/Tây được ký hiệu là S1. Điều này có nghĩa là khi có người đi bộ muốn qua đường thì cần phải nhấn nút xin đường để kết thúc đèn xanh. Quá trình chuyển đổi từ đèn vàng theo hướng Đông/Tây sang đèn xanh theo hướng Bắc/Nam cần một khoảng thời gian trễ thường là 4s. Tương tự với nhóm đèn Bắc/Nam, khi có người đi bộ muốn qua đường thì cần phải nhấn nút xin đường S2. Trạng thái cuối cùng cũng cần một thời gian trễ là 4s trước khi hệ thống quay trở lại trạng thái đầu tiên trong bảng trạng thái.

Bảng 5.3. Bảng trạng thái với quá trình chuyển đổi các trạng thái

Bảng trạng thái								
Mô tả trạng thái	#	L1	L2	L3	L4	L5	L6	
Xanh Đông/Tây	1	1	0	0	0	0	1	S1
Vàng Đông/Tây	2	1	0	0	0	1	0	4s
Xanh Bắc/Nam	3	0	0	1	1	0	0	4s
Vàng Bắc/Nam	4	0	1	0	1	0	0	S2

Sơ đồ trạng thái biểu diễn quá trình hoạt động của hệ thống được biểu diễn như Hình 5.19 dưới đây.



Hình 5.19. Sơ đồ trạng thái cho hệ thống đèn giao thông

5.1.2.3. Chuyển đổi sơ đồ trạng thái sang sơ đồ bậc thang

Các sơ đồ trạng thái có thể được chuyển đổi trực tiếp sang sơ đồ bậc thang bằng cách sử dụng các khối logic. Kỹ thuật này sẽ tạo ra một chương trình lớn nhưng đó là một phương pháp đơn giản để hiểu và dễ dàng hiệu chỉnh lỗi. Xét ví dụ hệ thống đèn giao thông như được mô tả trong phần trước với các đầu vào và đầu ra được định nghĩa như Bảng 5.4.

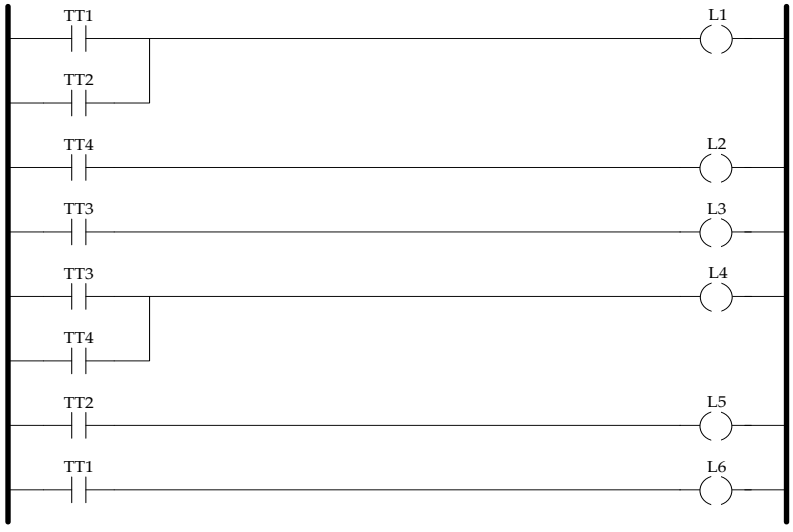
Bảng 5.4. Đầu vào/ra cho bộ điều khiển đèn giao thông

Trạng thái	Đầu ra	Đầu vào
TT1 – Xanh Đông/Tây	L1 – Đỏ Bắc/Nam	S1 – Nút nhấn qua đường
TT2 – Vàng Đông/Tây	L2 – Vàng Bắc/Nam	S2 – Nút nhấn qua đường
TT3 – Xanh Bắc/Nam	L3 – Xanh Bắc/Nam	FS – Lần quét đầu tiên
TT4 – Vàng Bắc/Nam	L4 – Đỏ Đông/Tây	
	L5 – Vàng Đông/Tây	
	L6 – Xanh Đông/Tây	

Trước tiên, chúng ta cần khởi tạo các giá trị ban đầu cho các trạng thái của hệ thống. Trong lần quét đầu tiên của PLC chỉ có TT1 được bật còn lại các trạng thái khác bị tắt.

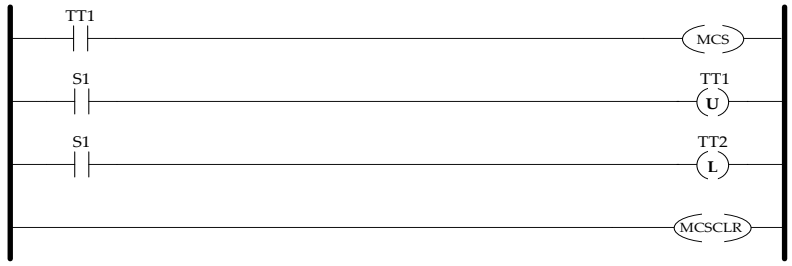
Hình 5.20. Khởi tạo các giá trị ban đầu cho bộ điều khiển đèn giao thông

Phần tiếp theo chỉ là công việc xử lý các đầu ra. Quan trọng chúng ta cần nhớ rằng các đầu ra phải được đặt ngoài các khối MCS và MCSCLR. Nếu nằm bên trong các khối này thì chúng chỉ có thể được kích hoạt khi khối MCS được kích hoạt; ngược lại chúng sẽ bị ngắt.



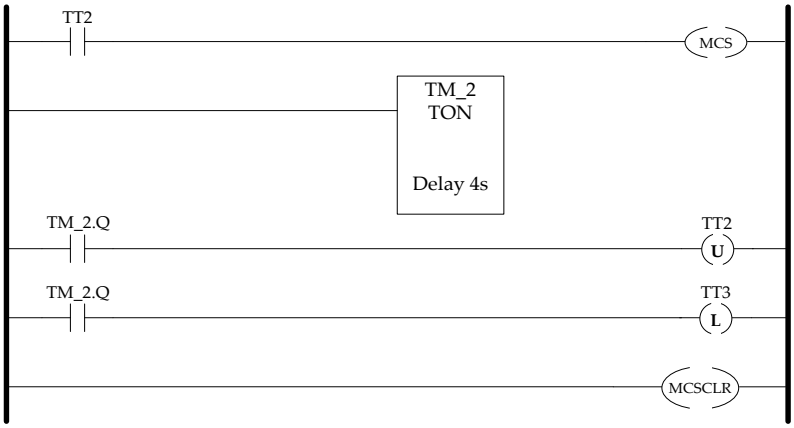
Hình 5.21. Sơ đồ bậc thang điều khiển các đầu ra chung

Trạng thái đầu tiên được thực hiện như Hình 5.22. Lệnh MCS được kích hoạt nếu TT1 được kích hoạt. Khi xảy ra quá trình chuyển đổi S1 sẽ kết thúc TT1 và khởi tạo TT2.



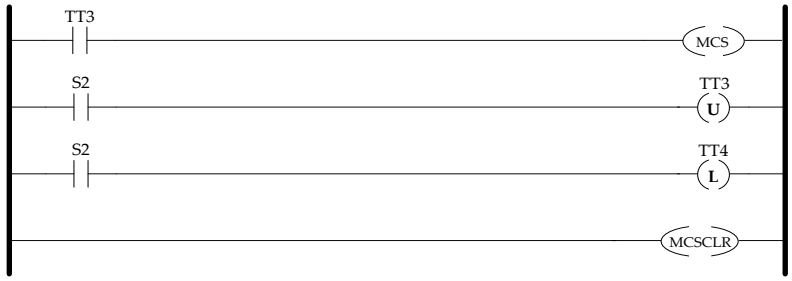
Hình 5.22. Sơ đồ bậc thang cho trạng thái thứ 1

Trạng thái thứ 2 được cho trên Hình 5.23. Khi TT2 tiếp điện thì bộ định thời tạo trễ TON bắt đầu hoạt động. Khi giá trị hiện tại của bộ định thời đạt tới giá trị đặt trước thì TT2 bị ngắt và TT3 được kích hoạt. Khi TT2 bị ngắt thì câu lệnh MCS sẽ bị ngắt đồng thời các câu lệnh nằm giữa MCS và MCSCLR cũng bị ngắt và TON được khởi động lại.

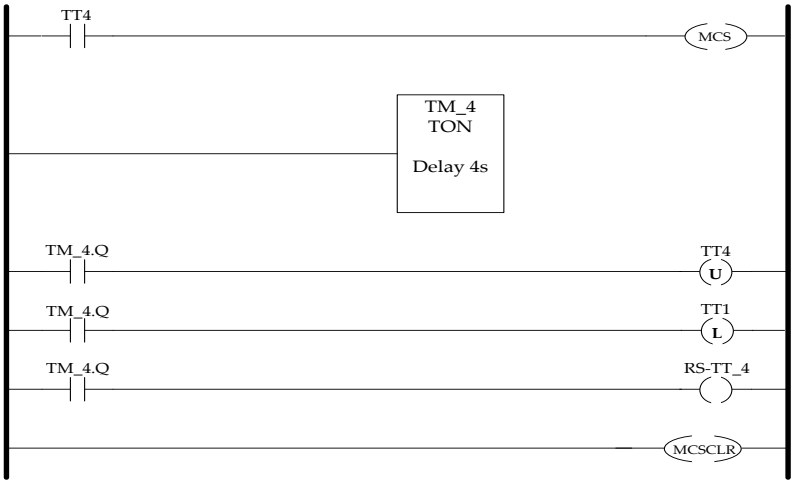


Hình 5.23. Sơ đồ bậc thang cho trạng thái thứ 2

Các trạng thái thứ 3 và thứ 4 được cho trong Hình 5.24 và Hình 5.25.

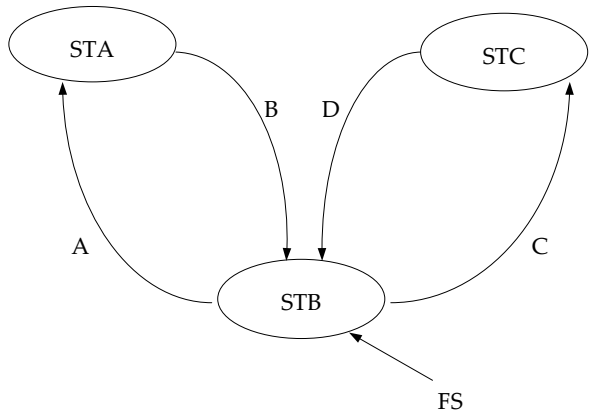


Hình 5.24. Sơ đồ bậc thang cho trạng thái thứ 3



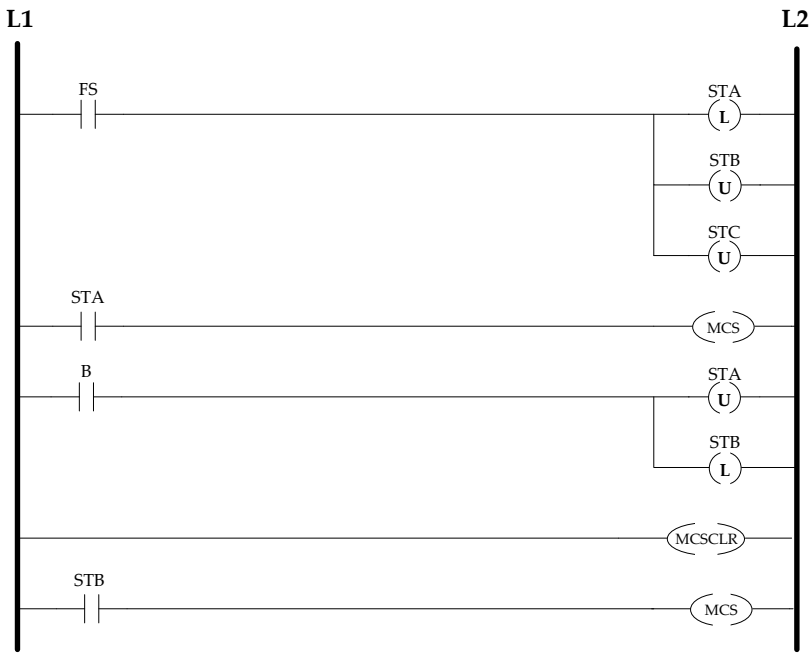
Hình 5.25. Sơ đồ bậc thang cho trạng thái thứ 4

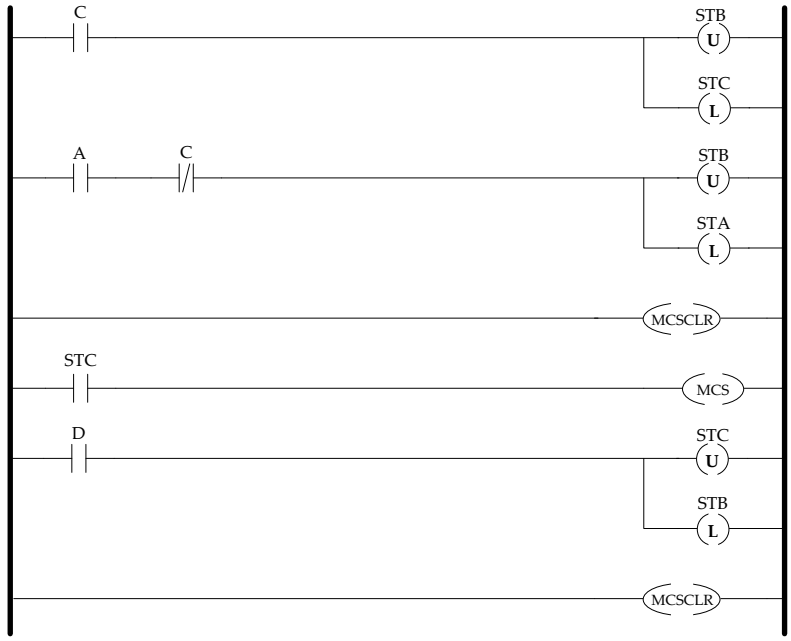
Ví dụ trước mới thực hiện ở mức đơn giản mà chưa tính tới trường hợp xấu có thể xảy ra. Sơ đồ trạng thái trên Hình 5.26 cho thấy 2 quá trình chuyển đổi trạng thái có thể xảy ra đồng thời. Chẳng hạn nếu hệ thống đang ở trạng thái STB mà xảy ra đồng thời 2 quá trình chuyển đổi A và C, thì khi đó hệ thống có thể rơi vào hoặc trạng thái STA hoặc STB hoặc cả 2 nếu chương trình được thiết kế không tốt. Để giải quyết vấn đề này chúng ta hãy gán mức ưu tiên cho một trong hai quá trình chuyển đổi.



Hình 5.26. Sơ đồ trạng thái với khả năng được ưu tiên

Sơ đồ bậc thang trên Hình 5.27 được thực hiện cùng với sơ đồ trạng thái trên Hình 5.26. Quá trình thực hiện này giống như quá trình đã được mô tả bên trên chỉ khác ở điểm đó là sẽ thêm một đoạn chương trình để ngăn cản quá trình chuyển đổi A nếu quá trình chuyển đổi C cũng được kích hoạt (ưu tiên quá trình C).





Hình 5.27. Sơ đồ bậc thang với trường hợp có ưu tiên

5.1.2.4. Phương trình trạng thái

Các sơ đồ trạng thái có thể được chuyển đổi sang các biểu thức Boolean và sau đó là sơ đồ bậc thang. Trước tiên chúng ta hãy tìm hiểu về các phương trình trạng thái. Các phương trình này gồm ba phần chính như được mô tả trên Hình 5.28. Một trạng thái là ON khi nó đang tồn tại ở trạng thái ON, hoặc nó được ON khi chuyển từ một trạng thái khác sang nhưng nó sẽ được OFF nếu nó chuyển sang trạng thái khác. Mỗi một trạng thái trong sơ đồ tương ứng với một phương trình trạng thái.

$$STATE_i = \left(STATE_i + \sum_{j=1}^{n} \left(T_{j,i} \,.\, STATE_j \right) \right) . \prod_{k=1}^{m} \overline{\left(T_{i,k} . STATE_i \right)}$$

Trong đó :

$STATE_i$ = *Trạng thái thứ i (biểu diễn trạng thái ON)*

n = *Số lượng chuyển đổi trạng thái tới trạng thái i .*

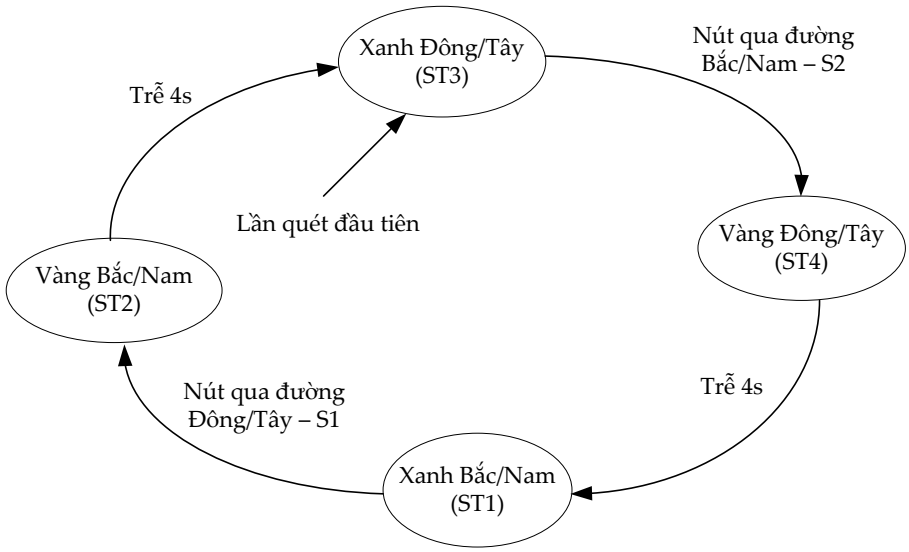
m = *Số lượng chuyển đổi trạng thái ra khỏi trạng thái i.*

$T_{j,i}$ = *Điều kiện logic để chuyển từ trạng thái j sang trạng thái i.*

$T_{i,k}$ = *Điều kiện logic để chuyển từ trạng thái i sang trạng thái k.*

Hình 5.28. Các phương trình trạng thái

Chúng ta sẽ áp dụng phương pháp này tới ví dụ điều khiển đèn giao thông với sơ đồ trạng thái như Hình 5.29.



Hình 5.29. Sơ đồ trạng thái hệ thống điều khiển đèn giao thông

Trước tiên chúng ta cần định nghĩa các biến. Tiếp theo là công việc phân tích sơ đồ trạng thái. Phương trình đầu tiên được viết cho trạng thái ST1 (Xanh Bắc/Nam). Trạng thái ST1 sẽ ON khi nó đang tồn tại ở trạng thái ON hoặc sau 4s khi ST4 được ON hoặc tại lần quét đầu tiên. Trạng thái ST1 là OFF nếu trạng thái hiện tại của nó là ON nhưng S1 được nhấn và S2 không được nhấn.

$$ST1 = Trạng\ thái\ 1 - Xanh\ Bắc/Nam$$

$$ST2 = Trạng\ thái\ 2 - Vàng\ Bắc/Nam$$

$$ST3 = Trạng\ thái\ 3 - Xanh\ Đông/Tây$$

$$ST4 = Trạng\ thái\ 4 - Vàng\ Đông/Tây$$

Phương trình biểu diễn mối liên quan các trạng thái:

$$ST1 = \left(ST1 + ST4\ .\ TON_2(ST4, 4s)\right)\ .\ \overline{ST1\ .\ S1\ .\ \overline{S2}} + \ FS$$

$$ST2 = (ST2 + ST1\ .\ S1\ .\ \overline{S2})\ .\ \overline{ST2\ .\ TON_1(ST2, 4s)}$$

$$ST3 = \left(ST3 + ST2\ .\ TON_1(ST2, 4s)\right)\ .\ \overline{ST3\ .\ \overline{S1}\ .\ S2}$$

$$ST4 = (ST4 + ST3\ .\ \overline{S1}\ .\ S2)\ .\ \overline{ST4\ .\ TON_2(ST4, 4s)}$$

Hình 5.30. Phương trình trạng thái cho ví dụ điều khiển đèn giao thông

Chú ý: Bộ định thời biểu diễn trong các phương trình này có dạng TONi(A, Delay) có nghĩa là: ***TON*** chỉ ra đây là bộ định thời tạo trễ, **A** là đầu vào IN của bộ định thời, ***Delay*** là giá trị đặt trước của bộ định thời và *i* là số thứ tự của bộ định thời.

Các phương trình trong Hình 5.30 không thể dùng để chuyển sang sơ đồ bậc thang. Vì vậy, chúng ta cần phải biến đổi các phương trình đó sang các phương trình có dạng như Hình 5.31.

$$ST1 = \left(ST1 + ST4\,.\,TON_2(ST4, 4s)\right).(\overline{ST1} + \overline{S1} + S2) + FS$$

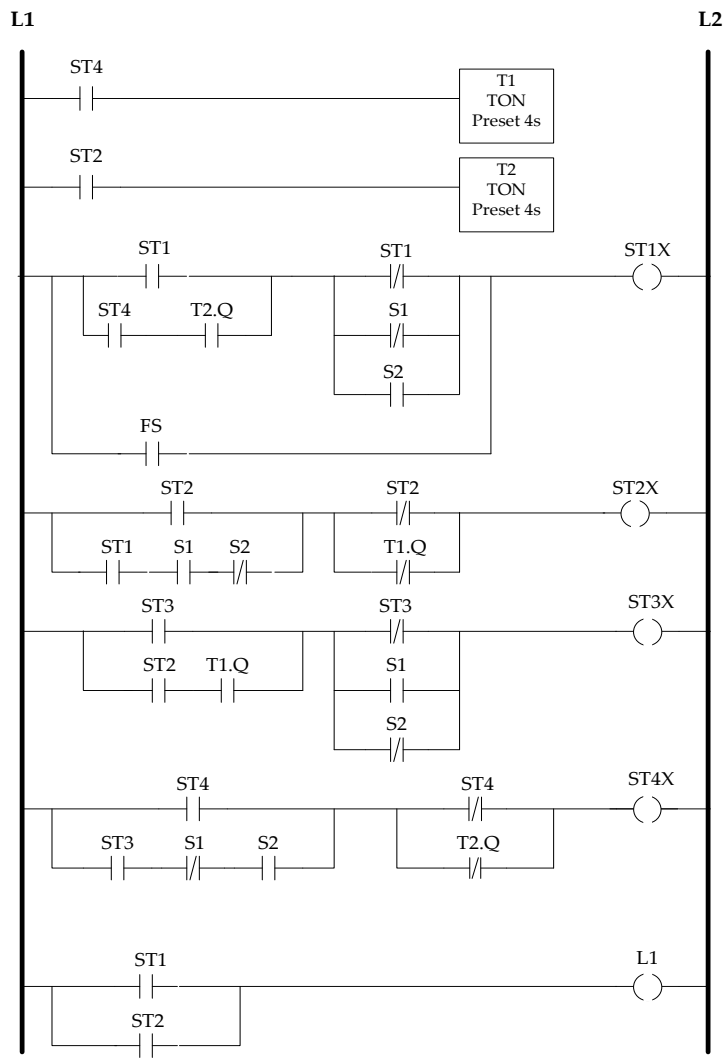
$$ST2 = (ST2 + ST1\,.\,S1\,.\,\overline{S2})\,.\,(\overline{ST2} + \overline{TON_1(ST2, 4s)})$$

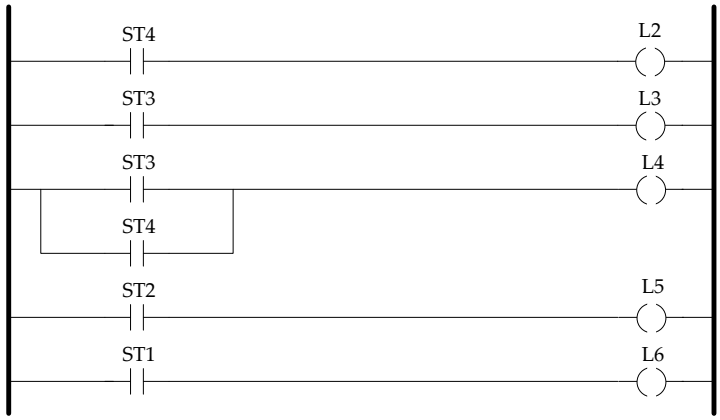
$$ST3 = \left(ST3 + ST2\,.\,TON_1(ST2, 4s)\right).(\overline{ST3} + S1 + \overline{S2})$$

$$ST4 = (ST4 + ST3\,.\,\overline{S1}\,.\,S2).(\overline{ST4} + \overline{TON_2(ST4, 4s)})$$

Hình 5.31. Các phương trình đại số Boolean

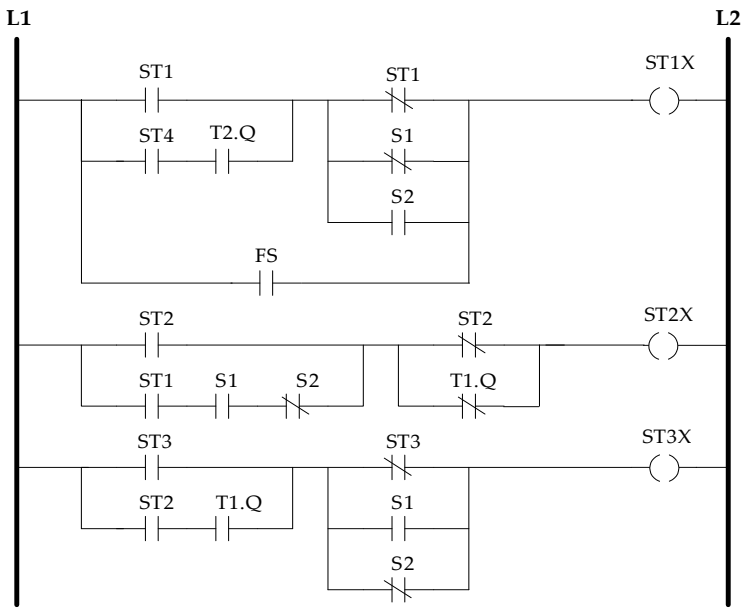
Các phương trình trạng thái này sau đó sẽ được chuyển sang sơ đồ bậc thang như Hình 5.32. Tại bậc 1 và 2 là các bộ định thời trễ. Tiếp theo là các phương trình trạng thái Boolean được chuyển đổi sang sơ đồ bậc thang. Sau đó là phần điều khiển các đèn báo hiệu.

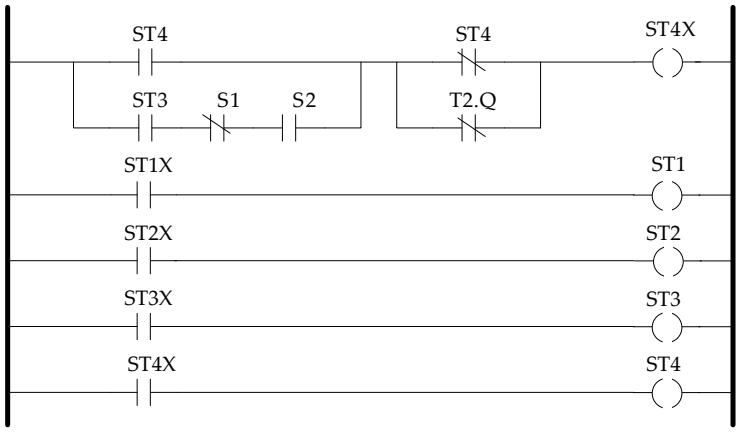




Hình 5.32. Sơ đồ bậc thang cho các phương trình trạng thái

Phương pháp này sẽ tạo ra mã lập trình với dung lượng nhỏ nhất trong các kỹ thuật lập trình. Tuy nhiên phương pháp cũng gặp phải những vấn đề tiềm ẩn. Xét ví dụ trên Hình 5.32, khi S1 được nhấn thì trạng thái của ST1 là OFF và ST2 là ON. Nhưng trên thực tế thì trạng thái của ST2 còn phụ thuộc vào trạng thái của ST1. Nếu trạng thái của ST1 là OFF ngay lập tức sau khi bậc thang này được quét thì trạng thái của tiếp điểm ST1 trên bậc thang có đầu ra là ST2 sẽ là OFF khi mà PLC quét tới bậc thang này (tốc độ quét của PLC chậm hơn thời gian ON của ST1). Điều này sẽ dẫn tới là trạng thái của ST2 là OFF. Vấn đề này bắt nguồn từ việc các phương trình trạng thái thường được tính song song nên các giá trị được cập nhật một cách đồng thời. Để giải quyết vấn đề này, chúng ta có thể sử dụng chương trình như được mô tả trên Hình 5.33. Trong chương trình này có sử dụng các biến tạm thời để lưu các giá trị trạng thái mới tạo ra. Sau khi tất cả các phương trình được thực hiện thì các trạng thái mới được cập nhật các giá trị mới.

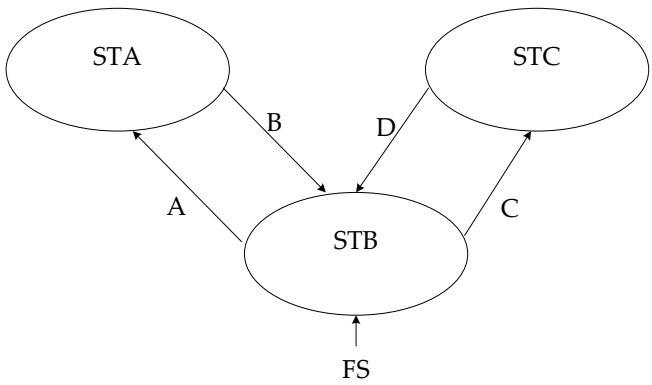




Hình 5.33. Cập nhật trạng thái

Khi có nhiều quá trình chuyển đổi diễn ra tại cùng một trạng thái chúng ta phải đặt quyền ưu tiên cho một quá trình nào đó. Khi lập trình chúng ta sẽ ưu tiên quá trình nào có mức ưu tiên cao hơn và loại bỏ quá trình nào có mức ưu tiên thấp hơn khi các quá trình chuyển đổi xảy ra đồng thời.

Ví dụ, sơ đồ trạng thái trên Hình 5.34 có hai quá trình chuyển đổi là A và C có thể xảy ra đồng thời. Các phương trình được viết sao cho chuyển đổi A có mức ưu tiên cao hơn. Khi các quá trình chuyển đổi xảy ra đồng thời thì A sẽ được ưu tiên và C sẽ bị cấm. Các phương trình này sau đó có thể được chuyển đổi sang sơ đồ bậc thang như Hình 5.35.



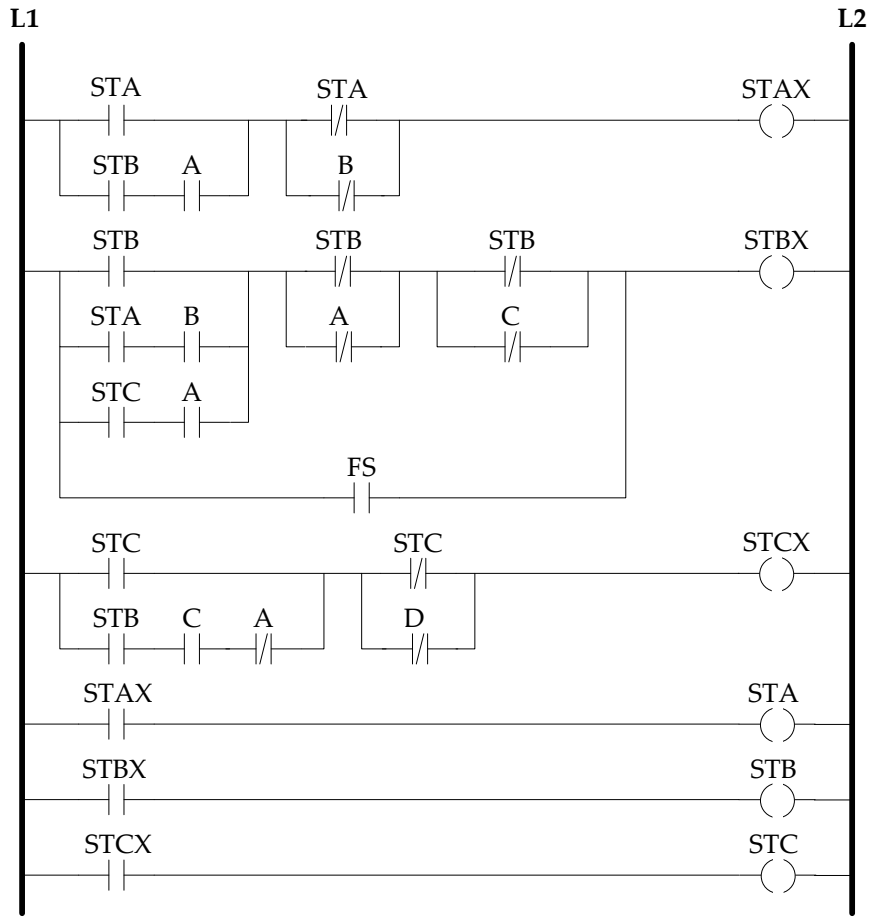
Hình 5.34. Sơ đồ trạng thái với mức ưu tiên khác nhau

Các phương trình trạng thái:

$STA = (STA + STB \cdot A) \cdot \overline{STA \cdot B}$

$STB = (STB + STA \cdot B + STC \cdot D) \cdot \overline{STB \cdot A} \cdot \overline{STB \cdot C} + FC$

$STC = (STC + STB \cdot C \cdot \overline{A}) \cdot \overline{STC \cdot D}$



Hình 5.35. Sơ đồ hình thạng với mức ưu tiên khác nhau

5.1.2.5. Phương trình chuyển đổi trạng thái

Có thể chuyển sơ đồ trạng thái sang các phương trình bằng cách viết phương trình cho mỗi trạng thái và mỗi quá trình chuyển đổi trạng thái. Mỗi trạng thái và quá trình chuyển đổi trạng thái được gán với một biến duy nhất. Các biến này sau đó sẽ được dùng để viết các phương trình cho sơ đồ. Các phương trình chuyển đổi được viết bằng cách xem xét mỗi trạng thái và xác định chuyển tiếp sẽ kết thúc trạng thái đó khi nào. Ví dụ, T1 sẽ có trạng thái là TRUE nếu trạng thái ST1 và S1 là TRUE còn S2 là FALSE. Phương trình viết cho quá trình chuyển đổi trạng thái tương tự như viết cho các trạng thái.

Định nghĩa trạng thái và các biến trạng thái

$ST1 - Xanh\ Bắc/Nam$	T1 = Chuyển đổi từ ST1 sang ST2
$ST2 = Vàng\ Bắc/Nam$	T2 = Chuyển đổi từ ST2 sang ST3
$ST3 = Xanh\ Đông/Tây$	T3 = Chuyển đổi từ ST3 sang ST4

$ST4 = V\grave{a}ng Đông\ ng/Tây$ T4 = Chuyển đổi từ ST4 sang ST5

T1 = Chuyển đổi tới ST1 cho cho lần quét đầu

Trạng thái và các phương trình trạng thái

T4 = ST4 . TON_2 (ST4, 4s) ST1 = (ST1 + $T4$ + T5) . $\overline{T1}$

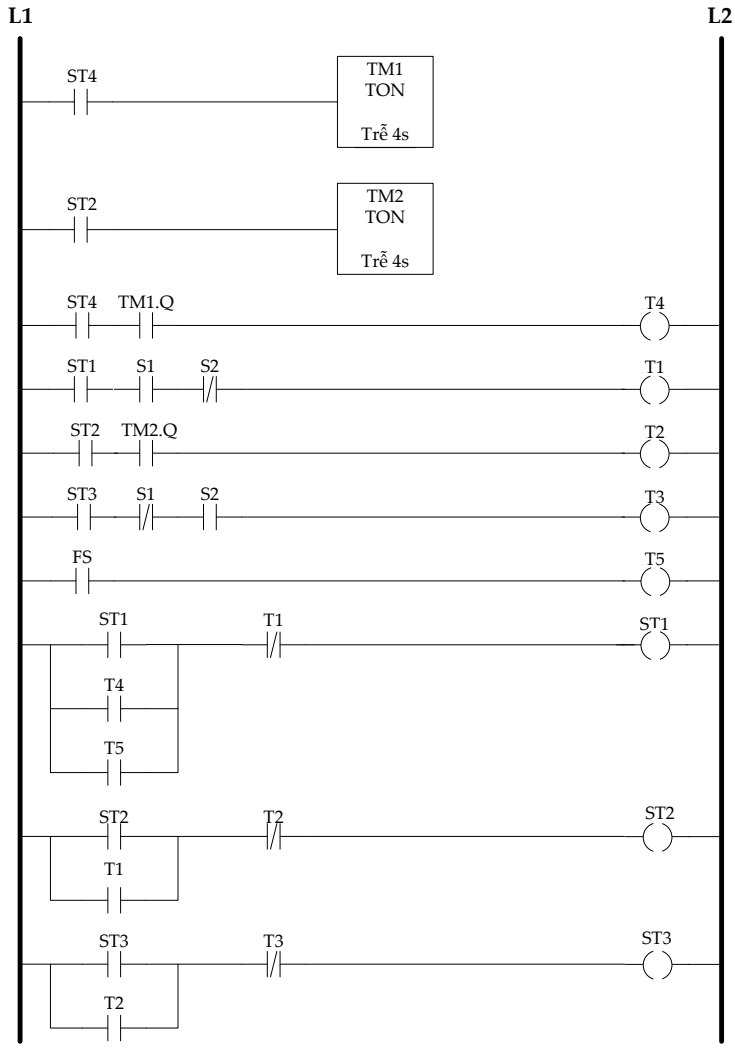
T1 = ST1 . S1 . $\overline{S2}$ ST2 = (ST2 + $T4$) . $\overline{T2}$

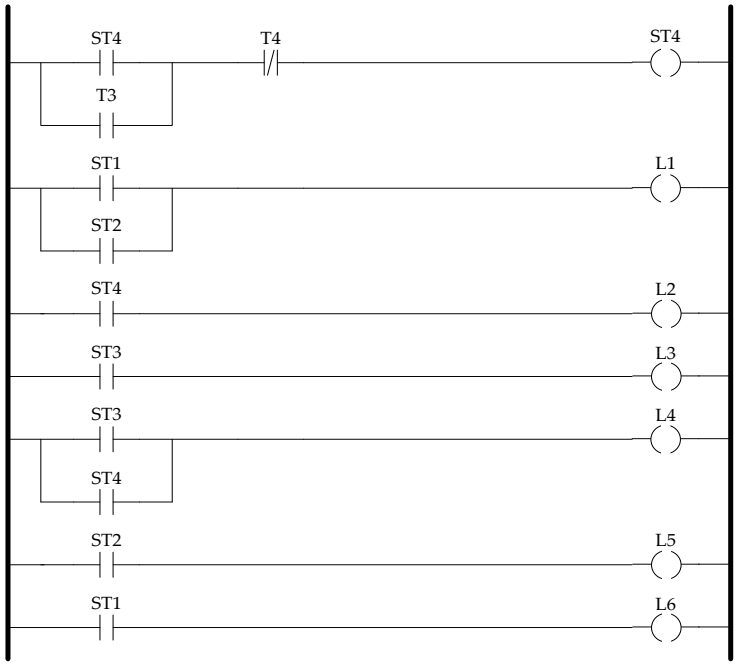
T2 = ST2 . TON_1 (ST2, 4s) ST3 = (ST3 + $T2$) . $\overline{T3}$

T3 = ST3 . S1 . $\overline{S2}$ ST4 = (ST4 + $T3$) . $\overline{T4}$

T5 = FS

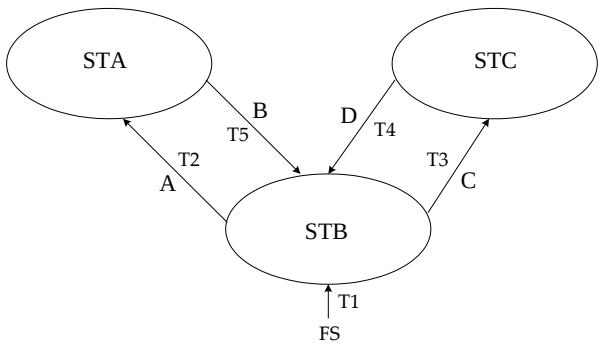
Các phương trình này có thể trực tiếp chuyển đổi sang bậc thang như Hình 5.36. Nếu các phương trình trạng thái được thực hiện trước phương trình chuyển tiếp sẽ gây ra một vấn đề bởi giá trị mới của các biến trạng thái sẽ bị bỏ qua. Vấn đề này đã được thảo luận trong phần phương trình trạng thái.





Hình 5.36. Sơ đồ logic bậc thang cho các phương trình chuyển đổi trạng thái

Để giải quyết vấn đề này chúng ta cũng sử dụng phương pháp gán mức ưu tiên cho các quá trình chuyển đổi. Trong ví dụ Hình 5.37, quá trình chuyển đổi T2 có mức ưu tiên cao hơn T3. Nếu đồng thời cùng xảy ra quá trình chuyển đổi T2 và T3 thì chương trình sẽ thực hiện theo hướng chuyển đổi T2 và T3 được bỏ qua.



Hình 5.37. Sơ đồ trạng thái với các mức ưu tiên khác nhau

Các phương trình trạng thái và phương trình chuyển đổi trạng thái:

T1 = FS

T2 = STB . A

T3 = STC . D

T4 = STC . D

T5 = STA . B

STA = (STA + T2) . $\overline{T5}$

STB = (STB + T5 + T4 + T1). $\overline{T2}$. $\overline{T3}$

STC = (STC + T3 . $\overline{T2}$) . $\overline{T4}$

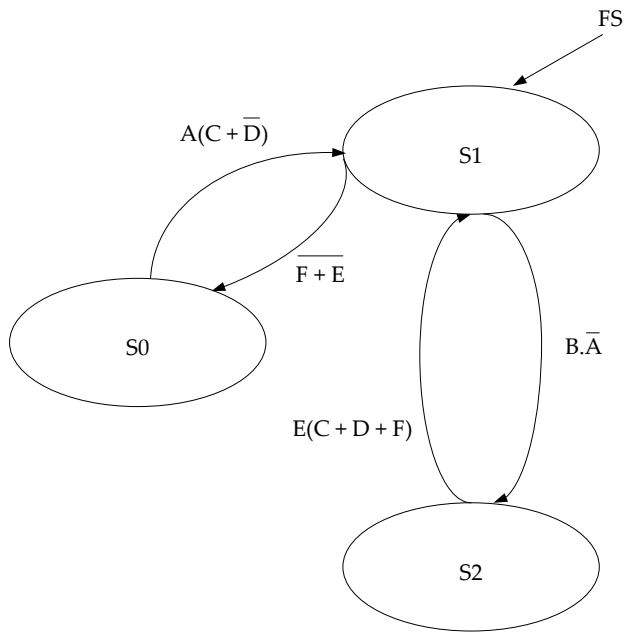
5.1.2.6. Một số ví dụ áp dụng

Ví dụ 1

Viết các phương trình trạng thái và chuyển đổi trạng thái cho sơ đồ trạng thái trên Hình 5.38.

Đầu vào	Đầu ra
A	P
B	Q
C	R
D	
E	
F	

TT	P	Q	R
S0	0	1	1
S1	1	0	1
S2	1	1	0



Hình 5.38. Hình dùng cho Ví dụ 1

<u>Các phương trình trạng thái:</u>

Định nghĩa các quá trình chuyển đổi trạng thái như sau:

T1 = Lần quét đầu tiên

T2 = Chuyển từ ST1 sang ST2

T3 = Chuyển từ ST2 sang ST1

T4 = Chuyển từ ST1 sang ST0

T5 = Chuyển từ ST0 sang ST1

Sau khi đã định nghĩa các quá trình chuyển đổi trạng thái, chúng ta có các phương trình trạng thái và chuyển đổi trạng thái như sau:

T1 = FS

T2 = S1 . (B . $\overline{A}$)

T3 = S2 . (E(C + D + F))

T4 = S1 . ($\overline{F + E}$)

T5 = S0 . (A . (C+ $\overline{D}$))

S1 = (S1 + T1 + T3 + T5) $\overline{T2}$. $\overline{T4}$

S2 = (S2 + T2) . $\overline{T3}$

S0 = (S0 + T4. $\overline{T2}$) . $\overline{T5}$

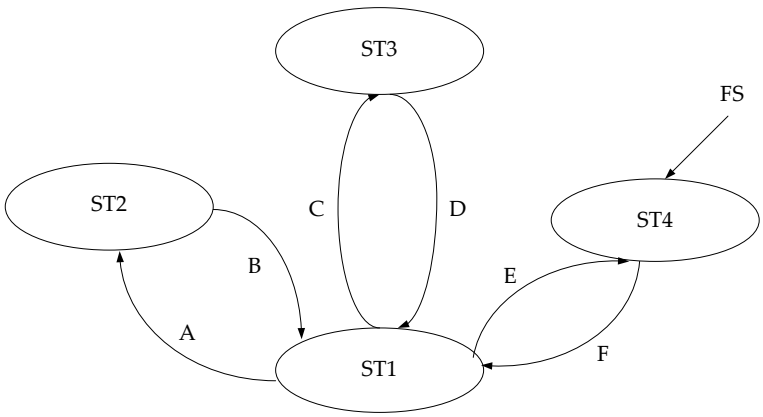
P = S1 + S2

Q = S0 + S2

R = S0 + S1

Ví dụ 2

Viết các phương trình trạng thái và sơ đồ bậc thang cho sơ đồ trạng thái hệ thống sau đây.



Hình 5.39. Hình dùng cho Ví dụ 3

Phương trình trạng thái:

Định nghĩa các quá trình chuyển đổi trạng thái như sau:

T1 = Chuyển từ ST1 sang ST2

T2 = Chuyển từ ST2 sang ST1

T3 = Chuyển từ ST1 sang ST3

T4 = Chuyển từ ST3 sang ST1

T5 = Chuyển từ ST1 sang ST4

T6 = Chuyển từ ST4 sang ST1

Sau khi đã định nghĩa các quá trình chuyển đổi trạng thái, chúng ta có các phương trình trạng thái và chuyển đổi trạng thái như sau:

T1 = ST1 . A

T2 = ST2 . B

T3 = ST1 . C

T4 = ST3 . D

T5 = ST1 . E

T6 = ST4 . F

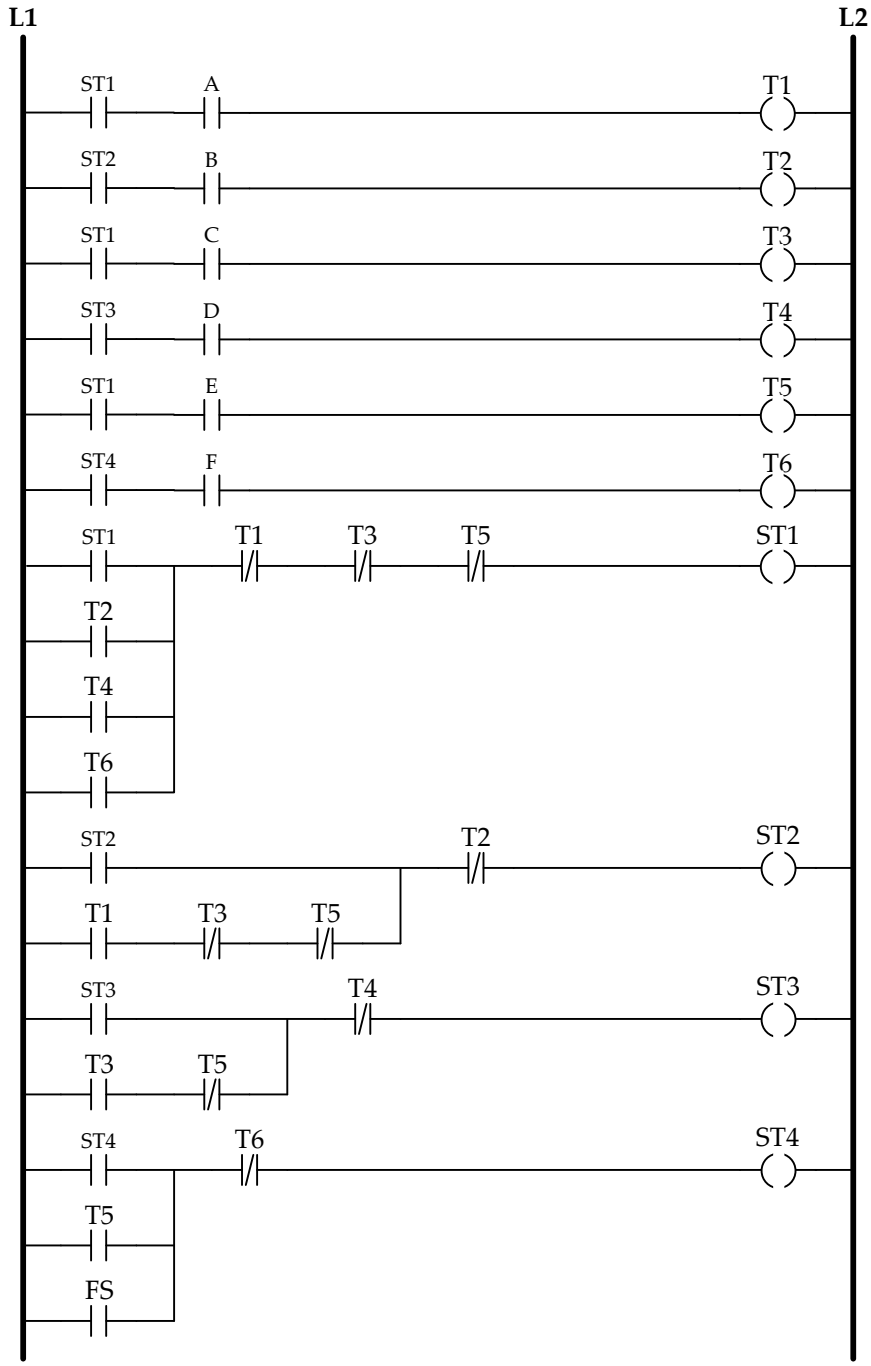
ST1 = (ST1 + T2 + T4 + T6).$\overline{T1}\ \overline{T3}\ \overline{T5}$

ST2 = (ST2 + T1 $\overline{T3}\ \overline{T5}$).$\overline{T2}$

ST3 = (ST3 + T3. $\overline{T5}$) . $\overline{T4}$

ST4 = (ST4 + T5 + FS) . $\overline{T6}$

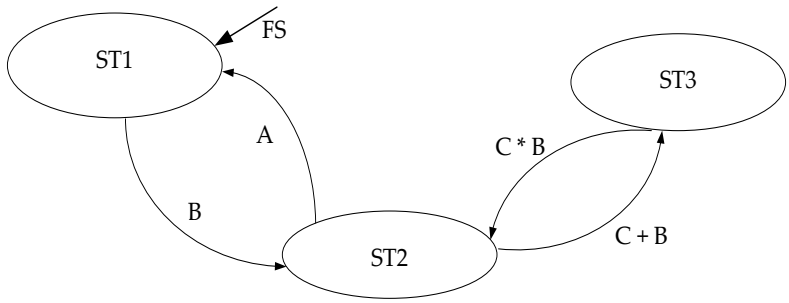
Sơ đồ logic bậc thang:



Hình 5.40. Sơ đồ bậc thang cho Ví dụ 2

Ví dụ 3

Cho sơ đồ trạng thái dưới đây, thiết lập các phương trình trạng thái và viết sơ đồ bậc thang.



Hình 5.41. Hình dùng cho Ví dụ 3

Phương trình trạng thái:

Định nghĩa các quá trình chuyển đổi trạng thái như sau:

T1 = Chuyển từ ST2 sang ST1 T3 = Chuyển từ ST3 sang ST2

T2 = Chuyển từ ST1 sang ST2 T4 = Chuyển từ ST2 sang ST3

Sau khi đã định nghĩa các quá trình chuyển đổi trạng thái, chúng ta có các phương trình trạng thái và chuyển đổi trạng thái như sau:

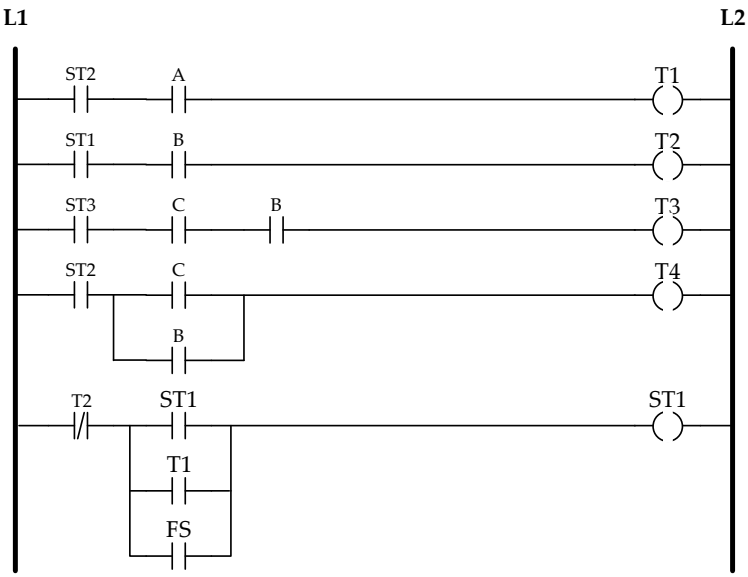
T1 = ST2 . A ST1 = (ST1 + T1) . $\overline{T2}$ + FS

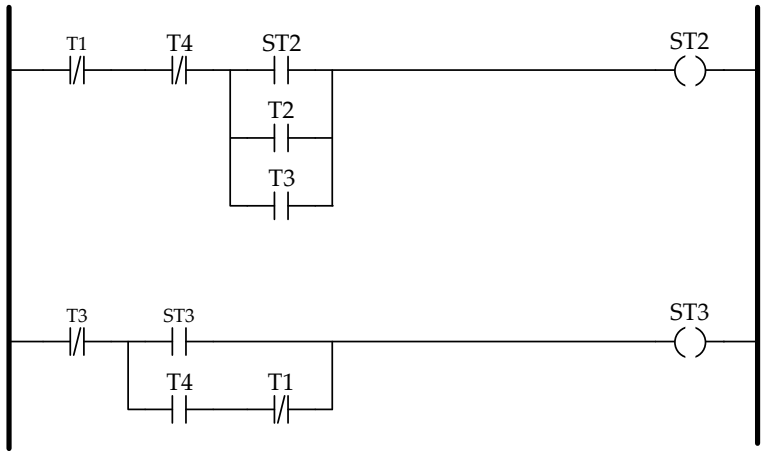
T2 = ST1 . B ST2 = (ST2 + T2 + T3) . $\overline{T1}$. $\overline{T4}$

T3 = ST3 . (C . B) ST3 = (ST3 + T4 . $\overline{T1}$) . $\overline{T3}$

T4 = ST2 . (C + B)

Sơ đồ logic bậc thang:

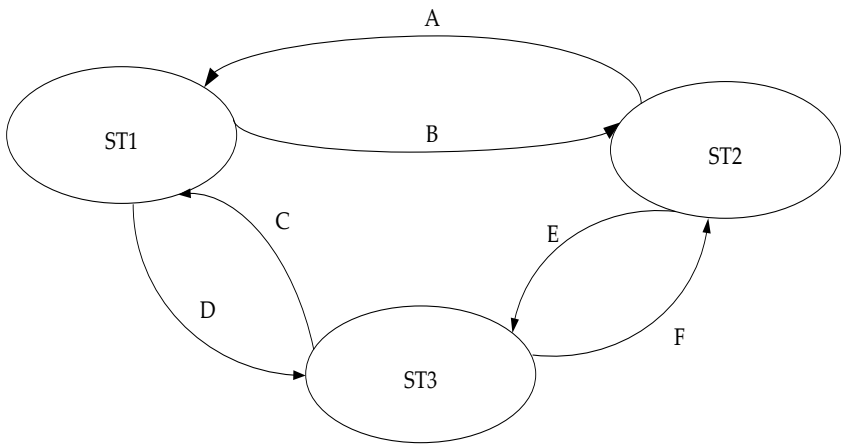




Hình 5.42. Sơ đồ bậc thang cho Ví dụ 3

Ví dụ 4:

Viết sơ đồ bậc thang cho sơ đồ trạng thái dưới đây bằng cách sử dụng các phương trình trạng thái.



Hình 5.43. Hình dùng cho Ví dụ 4

Phương trình trạng thái:

Định nghĩa các quá trình chuyển đổi trạng thái như sau:

TA = Chuyển từ ST2 sang ST1 TD = Chuyển từ ST1 sang ST3

TB = Chuyển từ ST1 sang ST2 TE = Chuyển từ ST2 sang ST3

TC = Chuyển từ ST3 sang ST1 TF = Chuyển từ ST3 sang ST2

Sau khi đã định nghĩa các quá trình chuyển đổi trạng thái, chúng ta có các phương trình trạng thái và chuyển đổi trạng thái như sau:

TA = ST2 . A

TB = ST1 . B

TC = ST3 . C

TD = ST1 . D + $\overline{B}$

TE = ST2 . E + $\overline{A}$

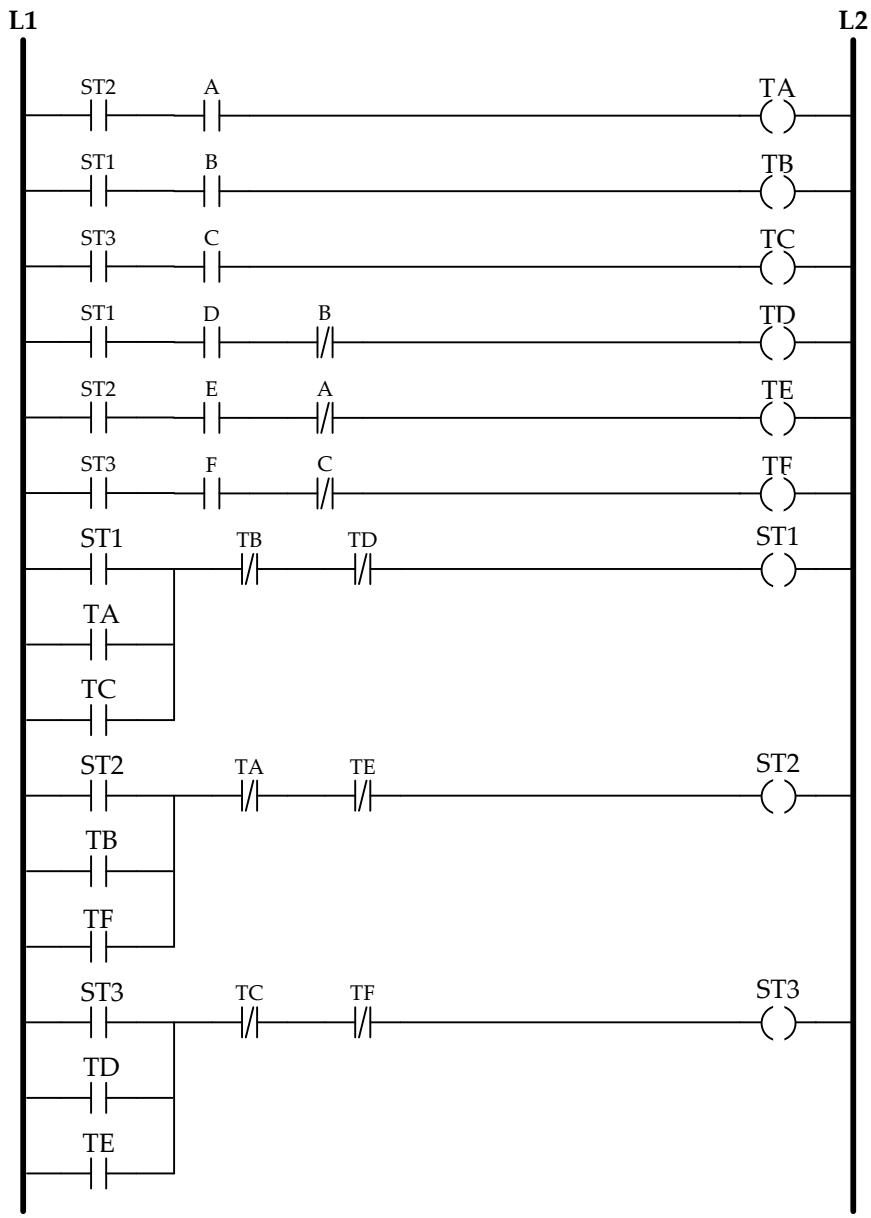
TF = ST3 . F + $\overline{C}$

ST1 = (ST1 + TA + TC) . $\overline{TB}$. $\overline{TD}$

ST2 = (ST2 + TB + TF) . $\overline{TA}$. $\overline{TE}$

ST3 = (ST3 + TD + TE) . $\overline{TC}$. $\overline{TF}$

<u>Sơ đồ logic bậc thang:</u>



Hình 5.44. Sơ đồ bậc thang cho Ví dụ 4

Ví dụ 5:

Một hệ thống điều khiển cửa gara với quy trình hoạt động như sau:

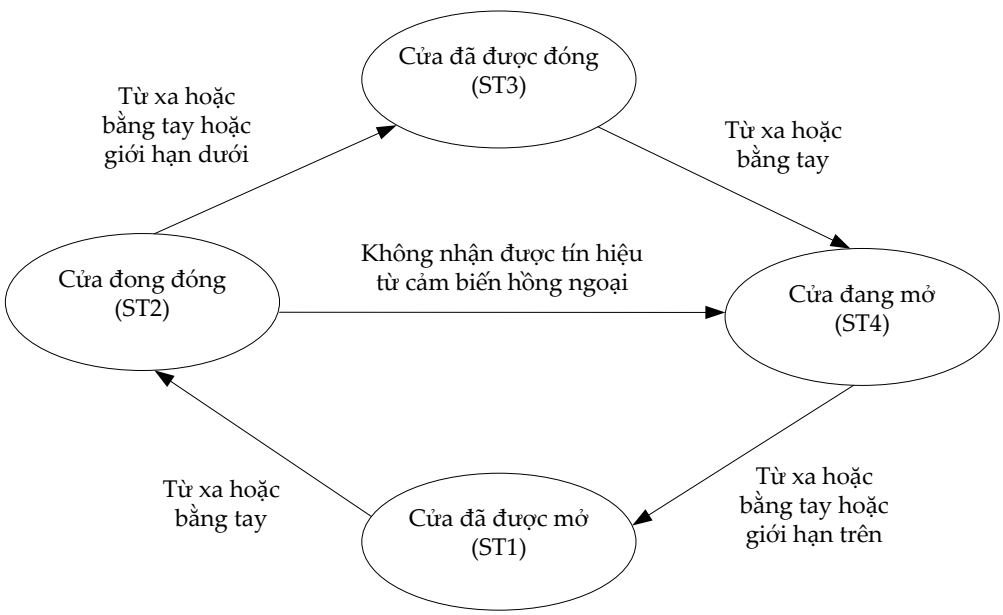
- Có 1 nút nhấn bằng tay trong gara và một nút điều khiển từ xa.
- Khi nút nhấn được nhấn thì cửa sẽ được kéo lên hoặc kéo xuống.
- Nếu nút nhấn được nhấn trong khi cửa đang di chuyển thì cửa sẽ được dừng lại, và nếu nhấn tiếp thì cửa sẽ bắt đầu di chuyển theo chiều ngược lại.
- Có các công tắc giới hạn trên và dưới để dừng di chuyển của cửa khi đạt tới giới hạn trên hoặc dưới.
- Có cảm biến hồng ngoại để phái hiện có sự di chuyển qua. Nếu cảm biến hồng ngoại nhận được tín hiệu trong khi cửa đang được đóng lại thì cửa sẽ được dừng lại và di chuyển theo chiều ngược lại.
- Có đèn báo sẽ sáng trong 5 phút khi cửa được mở hay đóng.

Hãy thiết kế chương trình điều khiển bằng phương pháp:

- Sử dụng khối logic
- Sử dụng các phương trình trạng thái

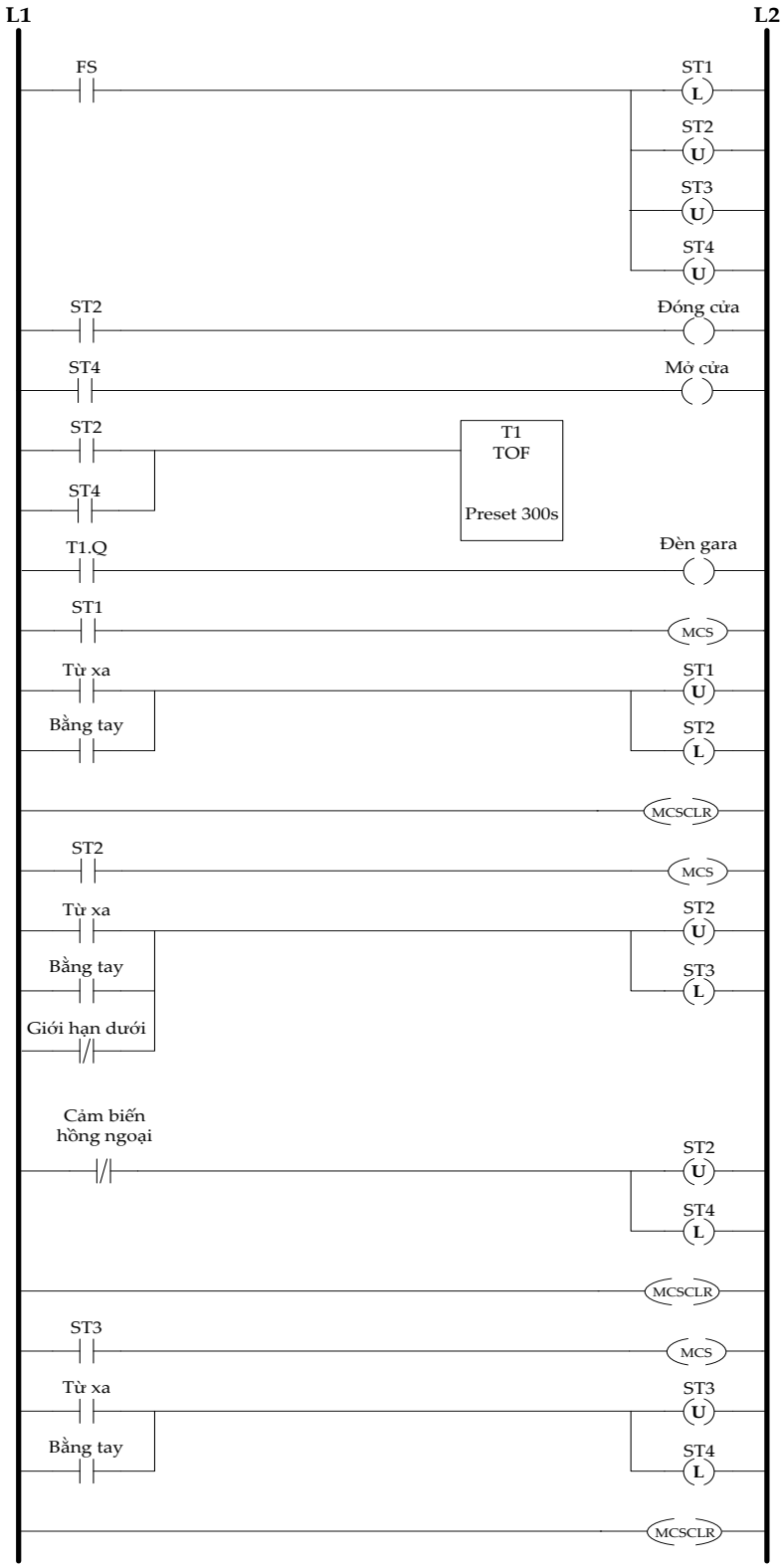
a. Phương pháp khối logic

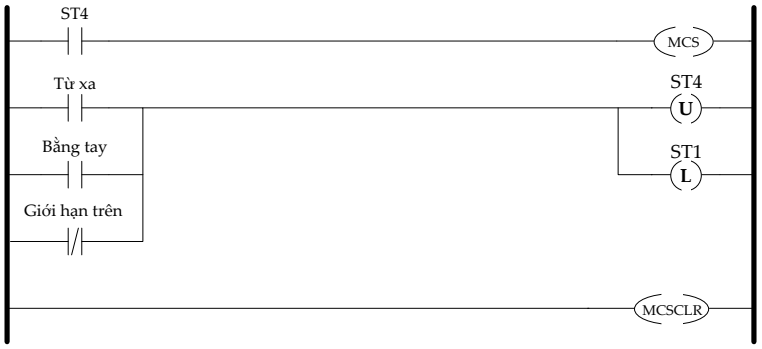
<u>Sơ đồ trạng thái của hệ thống:</u>



Hình 5.45. Sơ đồ trạng thái cho Ví dụ 5

Sơ đồ bậc thang:





Hình 5.46. Sơ đồ bậc thang cho Ví dụ 5

b. Phương pháp sử dụng các phương trình trạng thái

Các phương trình trạng thái:

Sử dụng sơ đồ trạng thái được trình bày với phương pháp khối logic và định nghĩa các quá trình chuyển đổi trạng thái như sau:

T1 = Chuyển từ ST1 sang ST2

T2 = Chuyển từ ST2 sang ST3

T3 = Chuyển từ ST2 sang ST4

T4 = Chuyển từ ST3 sang ST4

T5 = Chuyển từ ST4 sang ST1

Sau khi đã định nghĩa các quá trình chuyển đổi trạng thái, chúng ta có các phương trình trạng thái và chuyển đổi trạng thái như sau:

$ST1 = (ST1 + T5) . \overline{T1}$ $\quad$ $T1 = ST1 . (Từ\ xa + Bằng\ tay)$

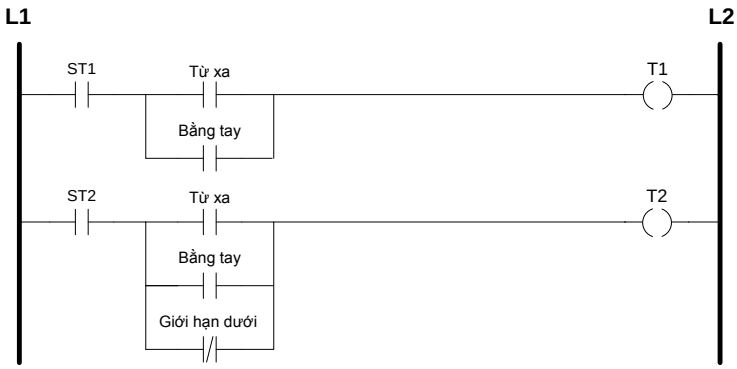
$ST2 = (ST2 + T1) . \overline{T2} . \overline{T3}$ $\quad$ $T2 = ST2 . (Từ\ xa + Bằng\ tay + Giới\ hạn\ dưới)$

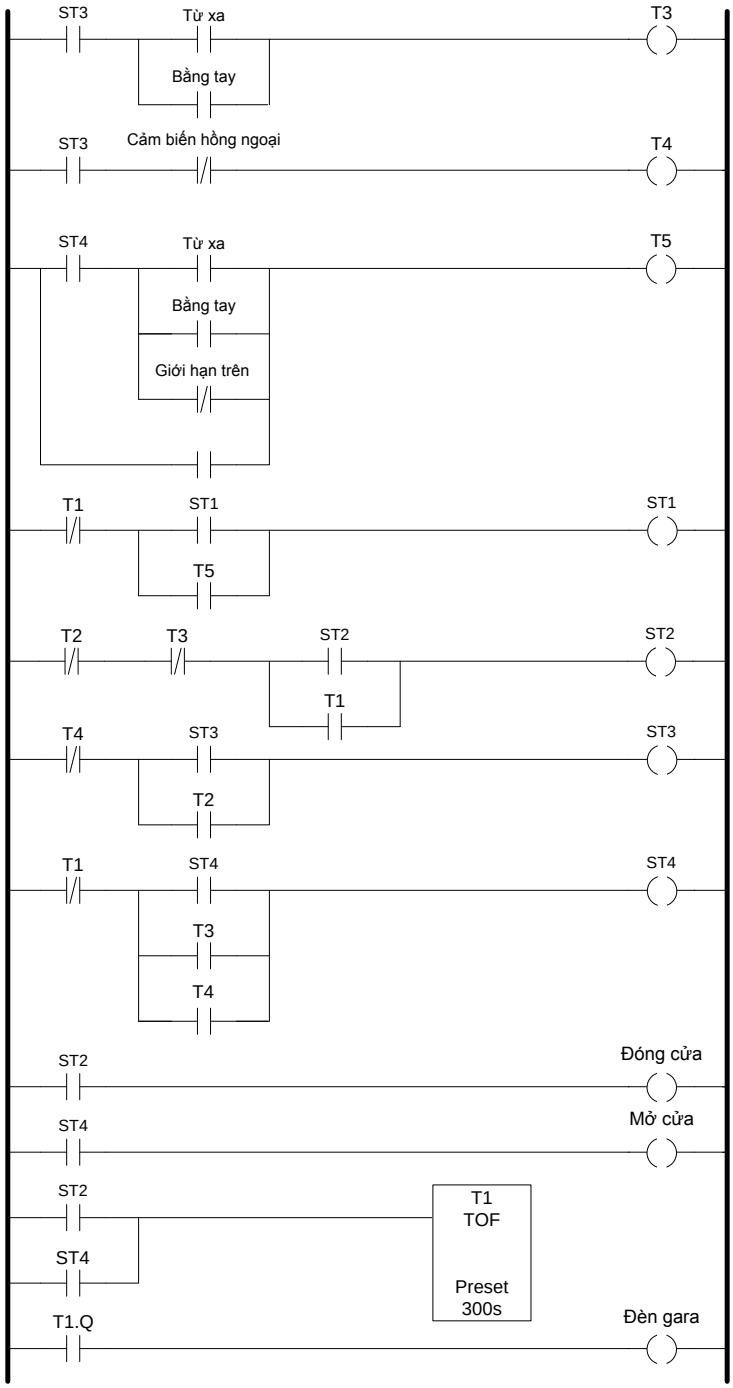
$ST3 = (ST3 + T2) . \overline{T4}$ $\quad$ $T3 = ST2 . (Từ\ xa + Bằng\ tay)$

$ST4 = (ST4 + T3 + T4) . \overline{T5}$ $\quad$ $T4 = ST3 . (Không\ nhận\ được\ tín\ hiệu\ từ\ cảm\ biến\ hồng\ ngoại)$

$T5 = ST4 . (Từ\ xa + Bằng\ tay + Giới\ hạn\ trên) + FS$

Sơ đồ bậc thang:





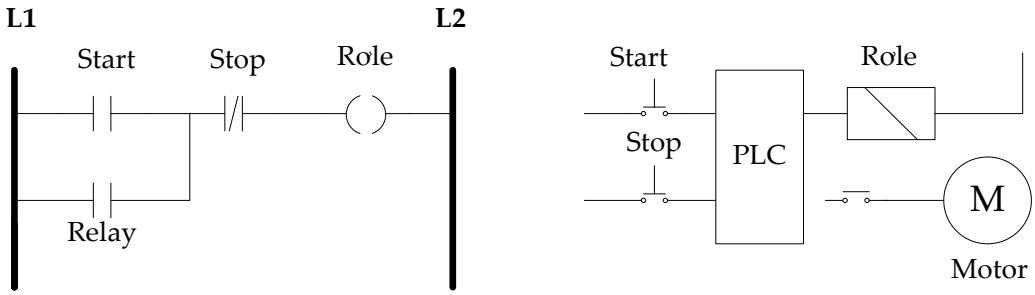
Hình 5.47. Sơ đồ bậc thang cho Ví dụ 6

5.2. AN TOÀN HỆ THỐNG

5.2.1. Hệ thống PLC với sự an toàn khi hoạt động

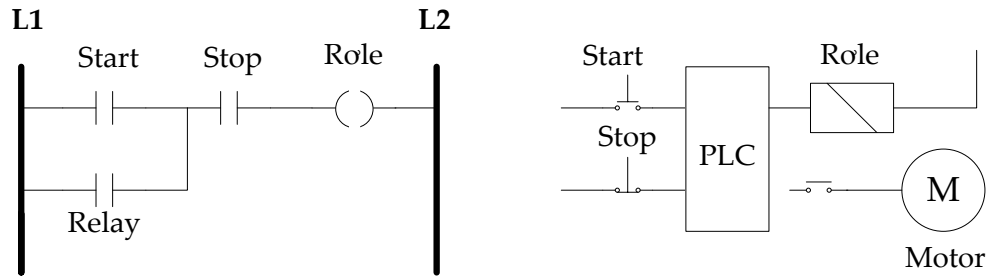
An toàn phải là sự ưu tiên hàng đầu khi thiết kế một hệ thống sử dụng PLC. Vì vậy, nút dừng hoạt động khẩn cấp và các chuyển mạch bảo vệ an toàn phải được thiết kế và không phụ thuộc vào quá trình thực hiện phần mềm điều khiển PLC. Do đó, trong trường hợp xảy ra sự cố với chương trình điều khiển chúng ta vẫn có thể dừng hoạt động của hệ thống một cách an toàn.

Hình 5.59 là chương trình sử dụng một nút nhấn để dừng hoạt động hệ thống PLC. Tuy nhiên sự bố trí như Hình 5.59 là không an toàn khi cần dừng hệ thống bởi nếu xảy ra lỗi chương trình thì tiếp điểm thường đóng Stop sẽ không thể hoạt động. Do đó sẽ không có tín hiệu dừng hệ thống và hệ thống sẽ không dừng lại được.



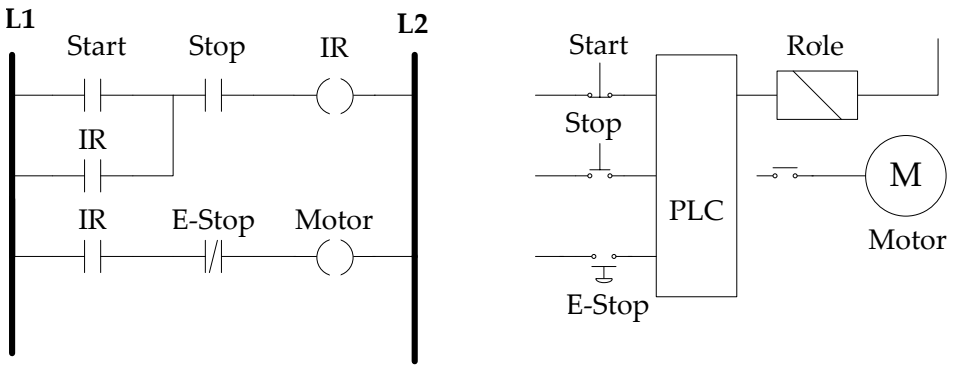
Hình 5.48. Hệ thống với quá trình dừng hoạt động không an toàn

Chúng ta có thể giải quyết vấn đề này khi bố trị lại chương trình như Hình 5.60. Trong chương trình này chúng ta sử dụng tiếp điểm thường mở Stop nhưng trong phần kết nối chúng ta sẽ dùng nút Stop loại thường đóng. Nếu xảy ra lỗi chương trình trong quá trình hoạt động chúng ta vẫn có thể dừng hoạt động của hệ thống bằng cách nhấn nút Stop bởi nút Stop là nút nhấn thường đóng.



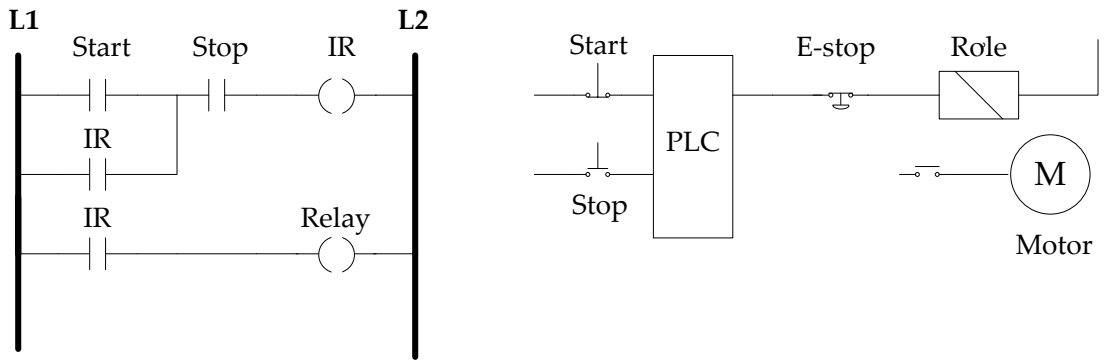
Hình 5.49. Hệ thống với quá trình dừng hoạt động một cách an toàn

Với mỗi một hệ thống PLC, chúng ta cần phải thiết kế một nút nhấn có chức năng dừng khẩn cấp (Emergency Stop) hoạt động độc lập với PLC để có thể dừng hệ thống một cách an toàn khi nút dừng (Stop) không hoạt động như Hình 5.61. Tuy nhiên cũng giống như trường hợp thiết kế cho nút dừng Stop. Chương trình này cũng không an toàn khi xảy ra sự cố về phần mềm.



Hình 5.50. Hệ thống không an toàn khi dừng hoạt động khẩn cấp

Để cải thiện mức độ an toàn cho hệ thống, thay vì nối nút nhấn dừng khẩn cấp (E-Stop) tại đầu vào chúng ta sẽ nối tại đầu ra như Hình 5.62 dưới đây.



Hình 5.51. Hệ thống an toàn với quá trình dừng hoạt động khẩn cấp

5.2.2. Bảo trì hệ thống

Mặc dù PLC được thiết kế để giảm thiểu tối đa công việc bảo trì, tuy nhiên cũng cần có một số biện pháp phòng ngừa cần được xem xét thường xuyên.

Nhiều hệ thống điều khiển dừng hoạt động trong một thời gian ngắn để thay đổi. Trong thời gian dừng đó có thể thực hiện một số công việc bảo trì dưới đây:

– Bụi bẩn tích lũy trên các bo mạch PLC cần phải vệ sinh sạch sẽ. Nếu bụi bẩn bám trên các bo mạch cũng như các bộ phận tản nhiệt có thể ngăn cản quá trình tản nhiệt và gây ra các trục trặc cho các bo mạch. Hơn thế nữa, nếu các hạt

dẫn điện bám trên các bo mạch điện tử có thể gây ra hiện tượng ngắn mạch gây phá hủy các bo mạch điện tử.

– Quá trình lắp ráp các mô-đun vào/ra nên được kiểm tra kỹ lưỡng trước khi lắp ráp để đảm bảo rằng tất cả các phích cắm, ổ cắm, các đầu nối, các mô-đun đã được kết nối đúng và an toàn. Kết nối không chắc chắn và đúng cách có thể dẫn đến hậu quả bộ điều khiển hoạt động không chính xác và có thể gây thiệt hại cho các thành phần khác của hệ thống.

– Kiểm tra tất cả các thiết bị vào/ra một cách kỹ lưỡng để đảm bảo rằng chúng lắp đặt chính xác. Các bo mạch nên được kiểm tra thường kỳ (khoảng 6 tháng 1 lần). Các thiết bị khác như cảm biến nên được kiểm tra hàng tháng. Các thiết bị hiện trường được sử dụng để chuyển các tín hiệu cơ học thành tín hiệu điện có thể bị bẩn hay hỏng hóc vì vậy sẽ hoạt động không chính xác.

– Nên đặt PLC cách xa các thiết bị phát nhiệt hay nhiễu lớn.

– Thường xuyên kiểm tra dung lượng pin để duy trì hoạt động của RAM. Nhiều loại CPU có đèn LED chỉ thị trạng thái điện áp của pin. Nếu cần thay thế pin thì phải thay chính xác loại pin đã sử dụng.

– Nên lưu lại chương trình điều khiển PLC đã được sử dụng.

5.3. VẬN HÀNH HỆ THỐNG

Trước khi vận hành hệ thống cần kiểm tra:

– Kiểm tra tất cả các dây nối giữa PLC và thiết bị một cách toàn diện, an toàn, các đặc kiểm kỹ thuật cần thiết và đáp ứng các tiêu chuẩn cục bộ.

– Kiểm tra nguồn điện phù hợp với mức điện áp hoạt động của PLC.

– Kiểm tra tất cả các thiết bị bảo vệ.

– Kiểm tra hoạt động của các nút nhấn dừng hệ thống khẩn cấp.

– Kiểm tra các thiết bị vào/ra được nối xem đã chính xác hay không.

– Nạp và chạy thử chương trình.

5.3.1. Kiểm tra các đầu vào/ra

Hoạt động của các thiết bị đầu vào (chẳng hạn như các chuyển mạch) có thể được chỉ thị bởi các LED được tích hợp trên các mô-đun đầu vào. Các LED sẽ sáng khi các đầu vào tiếp điện và sẽ tắt khi các đầu vào không được tiếp điện. LED không sáng có thể do thiết bị đầu vào hoạt động không chính xác, kết nối tới các mô-đun đầu vào chưa đúng, LED hay mô-đun đầu vào bị hỏng. Đối với các thiết

bị đầu ra có thể được khởi động một cách an toàn, mỗi đầu ra có thể sử dụng một nút ấn riêng biệt để có thể được kiểm tra.

5.3.2. Kiểm tra phần mềm điều khiển

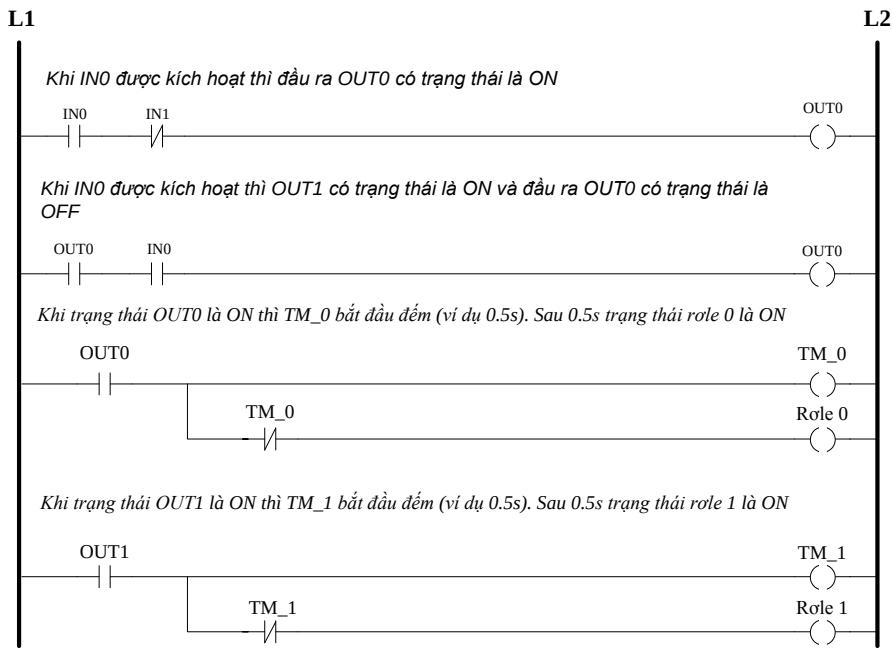
Hầu hết PLC có phần mềm kiểm tra chương trình. Phần mềm này sẽ kiểm tra chương trình xem địa chỉ thiết bị có chính xác hay không và cung cấp một danh sách các các đầu vào/ra được sử dụng, các giá trị cài đặt bộ đếm/bộ định thời và những lỗi tìm thấy. Ví dụ, PLC có thể đưa ra các thông báo rằng có nhiều đầu ra sử dụng cùng một địa chỉ hay bộ định thời (bộ đếm) được sử dụng mà chưa có giá trị định trước.

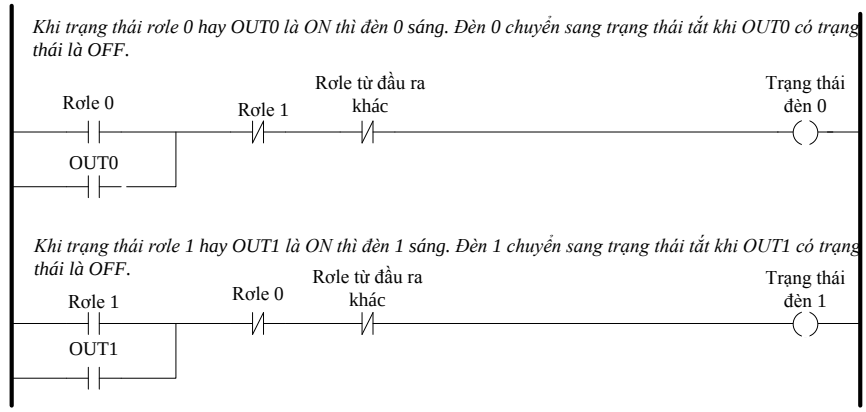
5.4. TÌM LỖI

Sau đây là một số kỹ thuật được sử dụng phổ biến để phát hiện lỗi:

– *Giám sát thời gian:* Bộ định thời Watchdog Timer được sử dụng để giám sát thời gian thực hiện chương trình. Giá trị đặt trước cho bộ định thời Watchdog lớn hơn giá trị thời gian quét thông thường của PLC. Nếu thời gian thực hiện vượt quá giá trị thời gian thông thường thì mặc định sẽ có 1 lỗi xảy ra và bộ định thời Watchdog ngắt và đưa ra cảnh báo hoặc kết thúc quá trình hoạt động của PLC.

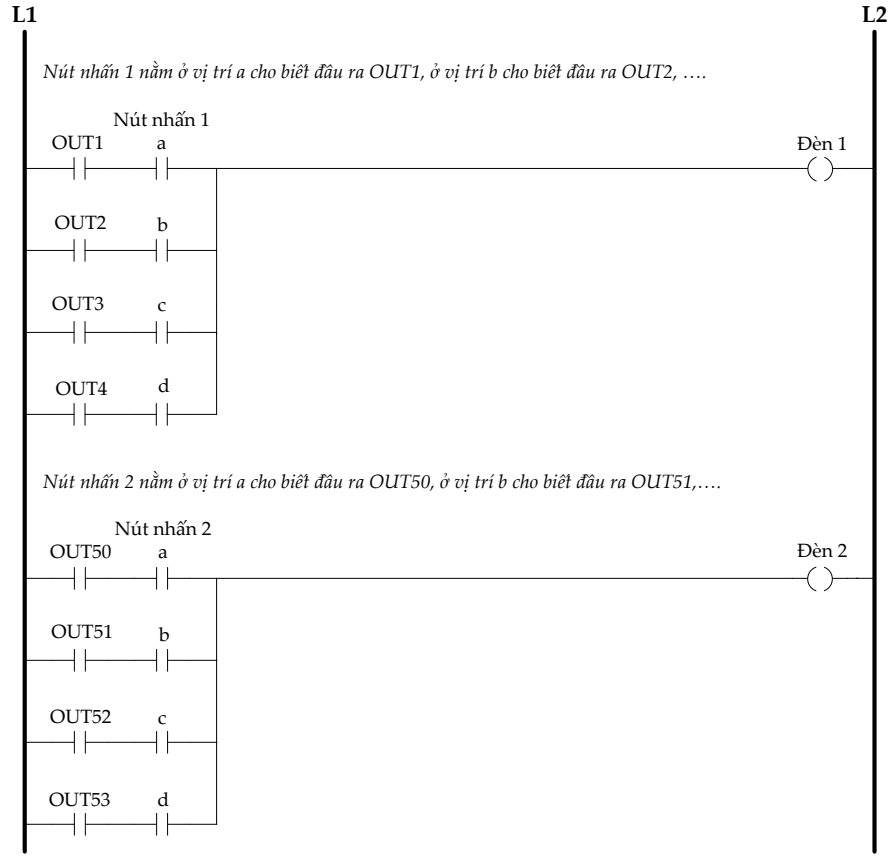
– *Giám sát đầu ra:* Kỹ thuật này sử dụng các đèn trạng thái để chỉ ra trạng thái sau cùng của các đầu ra. Mỗi đèn chỉ thị tương ứng với mỗi đầu ra. Khởi tạo chương trình sẽ được thiết kế để trạng thái của đèn là OFF và sau đó nếu trạng thái đầu ra là ON thì đèn báo mới sáng.





Hình 5.52. Chương trình phỏng đoán trạng thái đầu ra

Phương pháp này có thể gặp khó khăn trong các hệ thống lớn có nhiều đầu ra. Trong trường hợp đó, chúng ta có thể nhóm các đầu ra thành từng nhóm tương ứng và mỗi một nhóm sử dụng một đèn báo để chỉ thị trạng thái của nhóm.



Hình 5.53. Trạng thái của một nhóm đầu ra

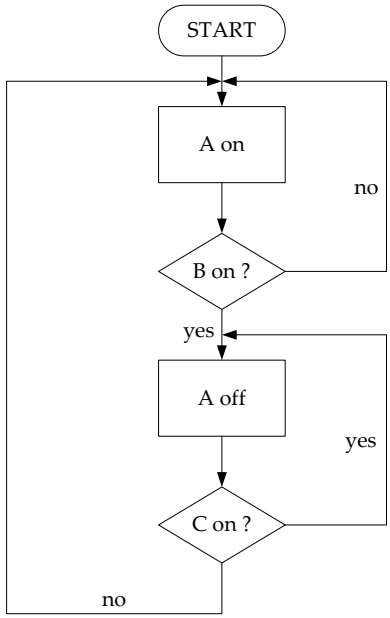
– *Lập lại hoạt động kiểm tra:* Thực hiện lập đi lập lại một quá trình hoạt động nào đó mà cho ra cùng một kết quả thì có thể giả thiết rằng không có lỗi nào xảy ra. Biện pháp này có thể tìm ra các sự cố quá độ. Một biện pháp tốn kém đó là có

thể dùng 2 hệ thông PLC thực hiện cùng một công việc và so sánh kết quả. Trong trường hợp không có lỗi thì hệ thống sẽ cho 2 kết quả như nhau.

– *Kiểm tra giá trị kỳ vọng:* Lỗi phần mềm có thể được phát hiện bằng cách kiểm tra giá trị thu được khi có một đầu vào cụ thể nào đó. Nếu giá trị thu được không như giá trị kỳ vọng thì đã có lỗi xảy ra.

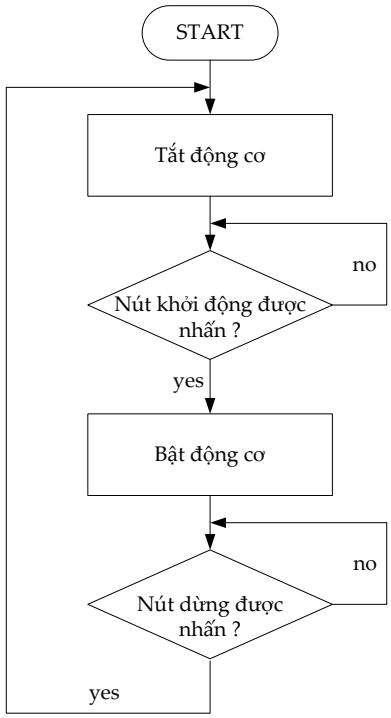
CÂU HỎI ÔN TẬP CHƯƠNG 5

1. Chuyển lưu đồ thuật toán cho trong Hình 5.48 sang sơ đồ bậc thang.



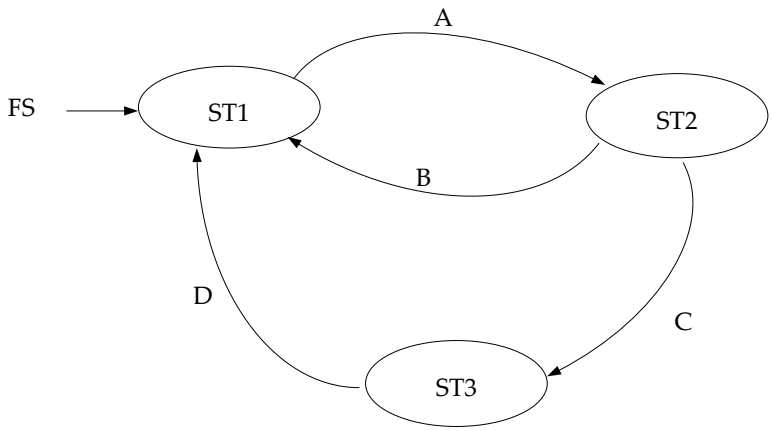
Hình 5.54. Hình dùng cho Bài 1

2. Chuyển lưu đồ thuật toán cho trong Hình 5.49 sang sơ đồ bậc thang.



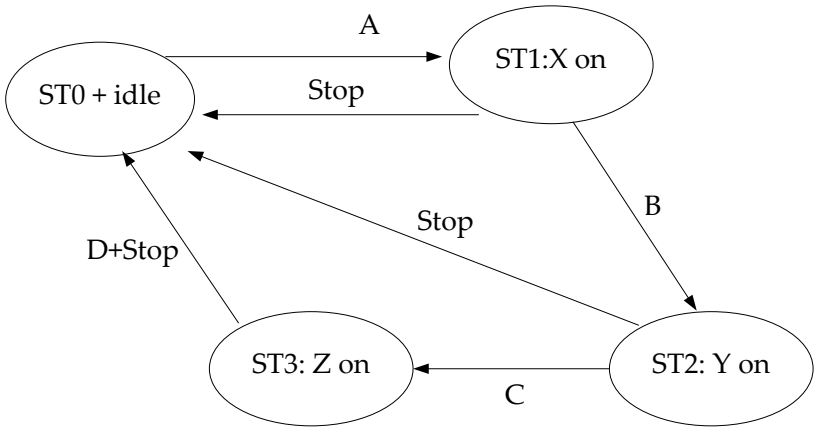
Hình 5.55. Hình dùng cho Bài 2

3. Hãy viết lưu đồ thuật toán điều khiển cho hệ thống tự động đóng gói sản phẩm chocolate hoạt động như sau:
   - Sự xuất hiện của hộp được phát hiện bởi cảm biến quang (P), sau khi phát hiện hệ thống sẽ dừng băng tải (C) và hộp được kẹp lại (H).
   - Máy đóng hộp (W) được khởi động trong 2s.
   - Thiết bị dán nhãn (S) được khởi động trong 1s để dán nhãn hàng lên hộp.
   - Kẹp (H) được ngắt và băng tải (C) hoạt động trở lại.
   - Sau khi hộp được đóng xong thì hệ thống trở lại trạng thái chờ.
4. Sử dụng phương pháp khối logic để viết sơ đồ bậc thang cho sơ đồ trạng thái cho trong Hình 5.50 dưới đây.



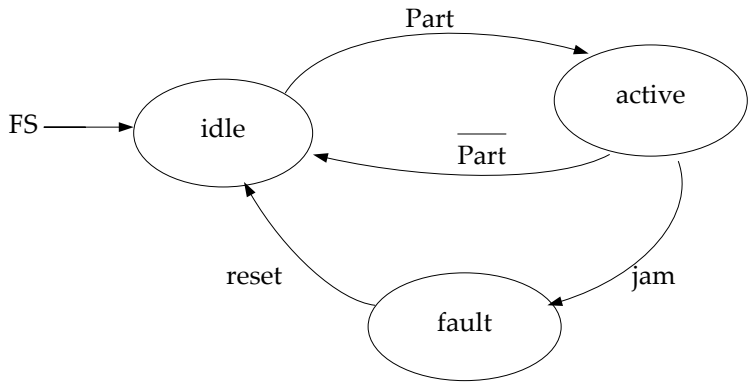
Hình 5.56. Hình dùng cho Bài 4

5. Chuyển đổi sơ đồ trạng thái cho trong Hình 5.51 sang sơ đồ bậc thang sử dụng phương pháp khối logic. Giả sử nút nhấn STOP có mức ưu tiên cao hơn.



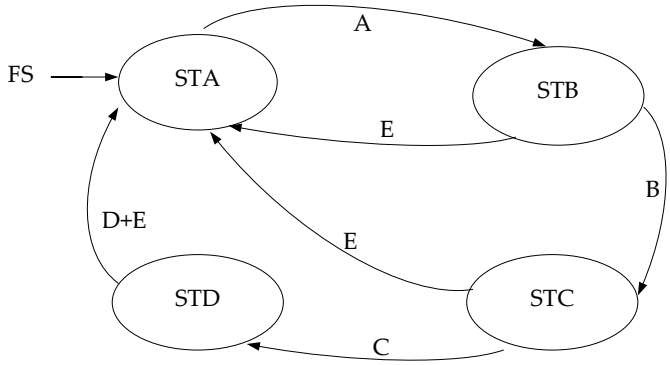
Hình 5.57. Hình dùng cho Bài 5

6. Chuyển đổi sơ đồ trạng thái cho trong Hình 5.52 sang sơ đồ bậc thang sử dụng phương pháp cập nhật trạng thái các đầu ra.



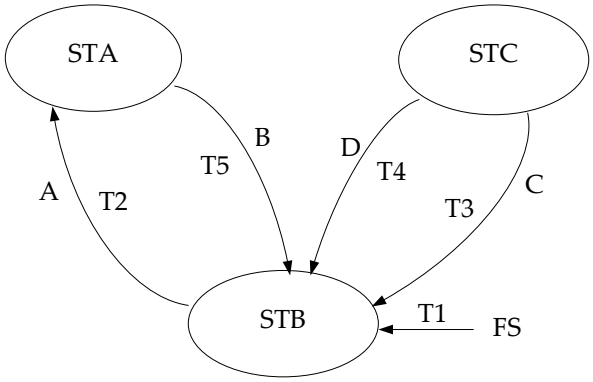
Hình 5.58. Hình dùng cho Bài 6

7. Sử dụng các phương trình trạng thái để viết sơ đồ bậc thang cho sơ đồ trạng thái dưới đây bằng cách sử dụng phương pháp cập nhật trạng thái các đầu ra. Chú ý các mức ưu tiên.



Hình 5.59. Hình dùng cho Bài 7

8. Cho sơ đồ trạng thái, các phương trình trạng thái, và các phương trình chuyển đổi trạng thái. Hãy chuyển sang sơ đồ bậc thang.



$T1 = FS$

$T2 = STB.A$

$T3 = STB.C$

$T4 = STC.D$

$T5 = STA.B$

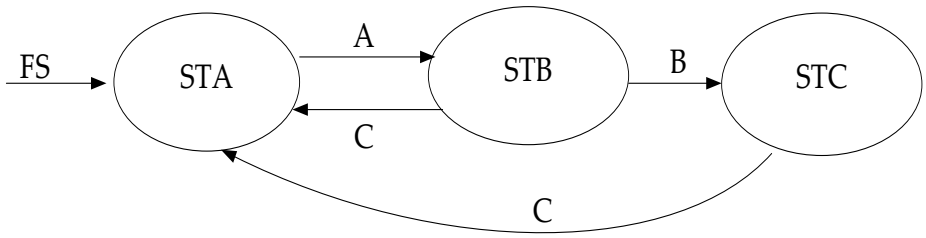
$STA = (STA + T2).\overline{T5}$

$STB = (STB + T5 + T4 + T1).\overline{T2}.\overline{T3}$

$STC = (STC + T3.\overline{T2}).\overline{T4}$

Hình 5.60. Hình dùng cho Bài 8

9. Viết sơ đồ bậc thang thực hiện sơ đồ trạng thái sau đây.

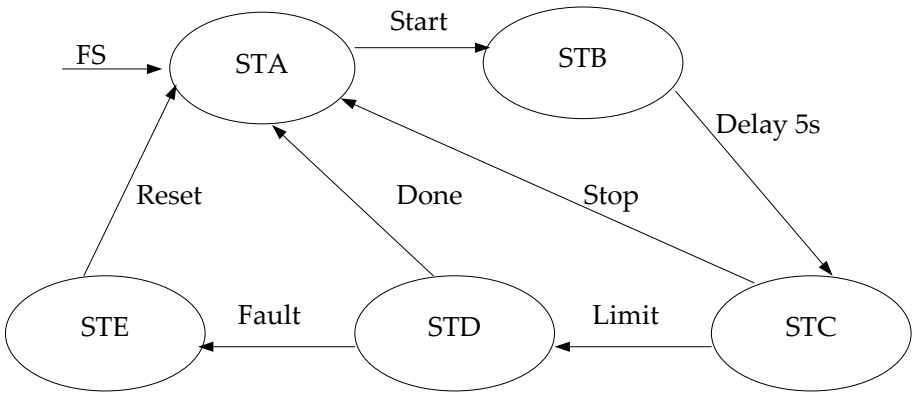


Hình 5.61. Hình dùng cho Bài 9

10. Chuyển đổi sơ đồ trạng thái sau đây sang sơ đồ bậc thang với các phương pháp sau:

    a. Sử dụng các phương trình trạng thái.

    b. Không sử dụng phương trình trạng thái.



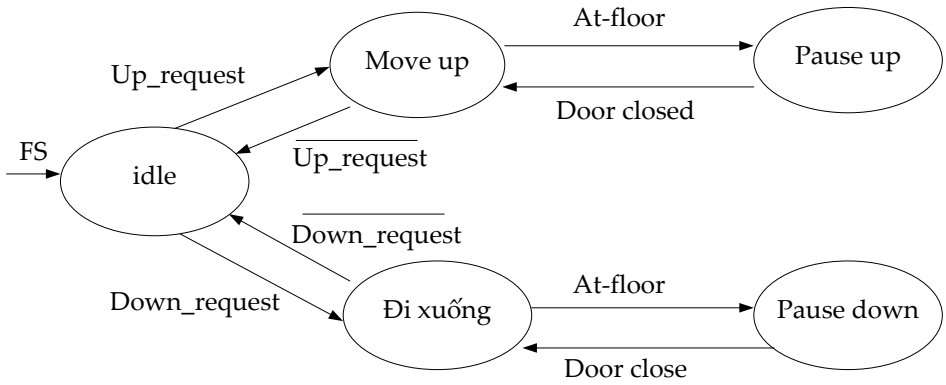
Hình 5.62. Hình dùng cho Bài 10

11. Cho sơ đồ trạng thái hoạt động của một bộ điều khiển máy nâng như dưới Hình 5.57 bên dưới. Bạn hãy:

    a. Sử dụng các phương trình đại số Boolean để viết chương trình điều khiển.

    b. Viết chương trình điều khiển sử dụng phương pháp khối logic.

c. Viết chương trình điều khiển sử dụng phương pháp cập nhật trạng thái các đầu ra.

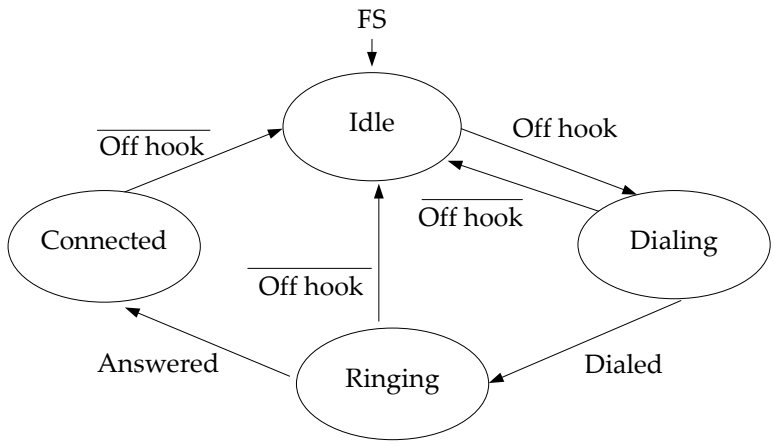


Hình 5.63. Hình dùng cho Bài 11

12. Viết sơ đồ bậc thang thực hiện sơ đồ trạng thái cho trong Hình 5.58 dưới đây bằng cách:

    a. Sử dụng các phương trình trạng thái.

    b. Không sử dụng phương trình trạng thái.



Hình 5.64. Hình dùng cho Bài 12

Phụ lục 1

THÔNG SỐ KỸ THUẬT BỘ PLC ED– 4260 TRAINER

1. Các thành phần cơ bản của PLC ED 4260 Trainer

- 01 khối CPU
- 01 khối nguồn (PS-4260)
- 01 mô-đun đầu vào (IM-4260-2)
- 01 mô-đun đầu ra (OM-4260-3)
- 01 mô-đun điều khiển vị trí (PM-4260-4)
- 01 mô-đun chuyển đổi tương tự - số (AD-4260-5)
- 01 mô-đun đo nhiệt độ sử dụng cặp nhiệt điện (SU-4260-9)
- 01 mô-đun còi và đèn báo (BL-4260-15)

2. Chi tiết về khối PLC

Phân loại	Chi tiết
Kiểu điều khiển	Lưu trữ chương trình, lặp lại việc tính toán, tính toán định kì, thực hiện việc ngắt.
Kiểu điều khiển vào ra	Xử lý quét đồng bộ (chức năng vào/ra trực tiếp)
Ngôn ngữ lập trình	LD, IL, SFC
Cấu trúc ngôn ngữ lập trình	Hoạt động : LD(13), IL(21) và 109 hàm cơ bản
	11 khối chức năng cơ bản, khối chức năng cho các mô đun đặc biệt

Phân loại	Chi tiết
Tốc độ tính toán	Với hàm cơ bản mất 0.2us/lệnh, 0.2us/bước với các khối hàm cơ bản (FB)
Bộ nhớ chương trình	128Kbytes (32k bước)
Số điểm vào/ra	512 điểm cho mô-đun 16 bit, 1024 điểm cho mô-đun 32 bit
Bộ nhớ dữ liệu	Biến trực tiếp : 2~16 kbytes
	Biến gián tiếp : 52 kbytes
Bộ định thời	Không giới hạn
	Giải thời gian : 0.001s ~ 4294967.295s (1.193hrs))
Bộ đếm sự kiện	Không giới hạn
	Giải giá trị đếm : -32768 ~ +32768
Chế độ hoạt động	RUN, STOP, PAUSE, DEBUG
Phục hồi dữ liệu khi nguồn bị sự cố	Kiểu dữ liệu tĩnh được duy trì
Khối lập trình	180 khối
Chương trình (tác vụ - Task)	Quét : Chương trình không đăng ký như một chương trình nhiệm vụ
	32 tác vụ định kì, 08 tác vụ liên hệ bên ngoài, 16 tác vụ liên hệ bên trong
	03 tác vụ khởi tạo (_INIT, _H, _INIT, _ERR, _SYS)
Tự phỏng đoán	Bộ định thời Watch dog, lỗi bộ nhớ, lỗi vào/ra, lỗi nguồn,...

Phân loại	Chi tiết
Chế độ khởi động lại	Khi lạnh, ấm và nóng
Mạng truyền thông	Truyền thông với các mô-đun FMM, FSM
Khe cắm cơ bản	08 khe (ngoại trừ nguồn và mô đun CPU)
Môi trường hoạt động	Nhiệt đô: 0 ~40 ^{0}C, RH: 20 ~ 80%
Điện áp đầu vào	AC 220V, 60 Hz / DC 24V
Kích thước khối PLC	438 (W) * 159(H) * 140(D)mm

Phân loại	Đặc điểm kỹ thuật
Đầu ra DC	Điện áp: 0 ~ 24V, 24 V (cố định)
	Dòng: 2A (lớn nhất)
	Độ gợn sóng: 0.02% + 2mV
	Vôn kế : DVM 3 chữ số
	Ampe kế: AMM
	Bảo vệ đầu ra DC: Bảo vệ dòng
Đầu ra AC	Điện áp: 220V (cố định)
	Dòng 1A (lớn nhất)
	Đầu nối: Loại đầu nối an toàn
	Mạch bảo vệ: Ngắtắt đi khi điện quá tải
	Vôn kế : DVM 3 số

Phân loại	Đặc điểm kỹ thuật
Tín hiệu điều khiển đầu vào (Với mô-đun đầu vào)	16 đầu vào chuyển mạch có thể chọn
	02 nút nhấn
	01 nút nhấn 2 trạng thái
	04 công tắc số, cài đặt kết nối 25 pin
	02 xung HSC và 1 xung điều khiển động cơ bước.
Thiết bị đầu ra (Với mô-đun đầu ra và mô-đun mô phỏng)	16 đèn báo DCm, 24V, Φ8
	02 đèn báo DC 24V, Φ16
	01 bộ đếm nhị phân 4 số
	01 còi báo DC 24V
	01 động cơ bước
	02 động cơ DC
Đầu nối vào/ra	Đầu vào 32 bit, 08 COM
	Đầu ra 32 bit, 8 COM
Điện áp đầu vào	AC 220V 50/60 Hz
Số mô-đun	03 mô-đun cơ bản
Kích thước mô-đun	250(W) * (H) * 166(D)mm
Kĩch thước hệ thống	760(W) * 352(H) * 437(D)mm
Khối lượng (hệ thống)	34Kg

Phụ lục 2

ĐỊNH DẠNG DỮ LIỆU TRONG GMWIN

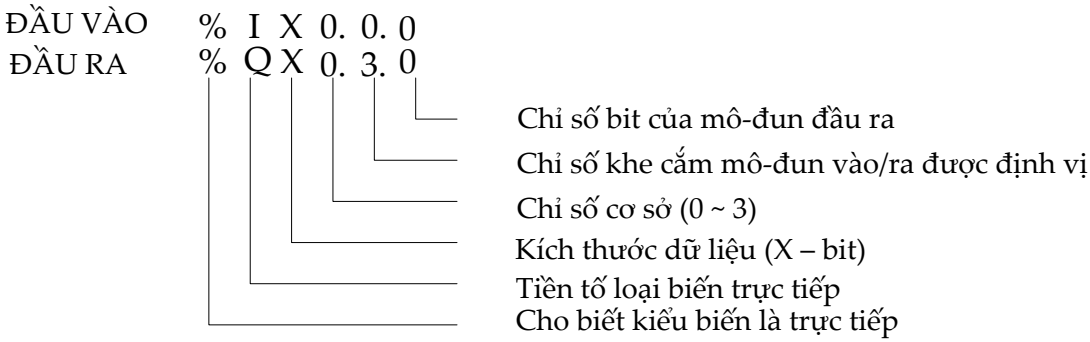
1. Khai báo biến trực tiếp

TT	Tiền tố	Ý nghĩa
1	%I	Khai báo biến đầu vào
2	%Q	Khai báo biến đầu ra
3	%M	Khai báo biến nội

2. Kích thước tiền tố

TT	Tiền tố	Kích thước
1	X	1 bit
2	B	1 byte (8 bit)
3	W	1 word (16 bit)
4	D	1 double word (32 bit)
5	L	1 long word (64 bit)

Ví dụ:

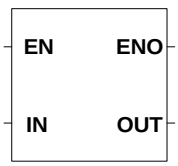


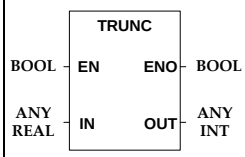
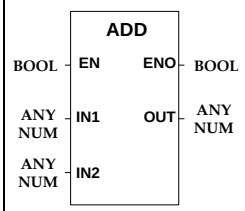
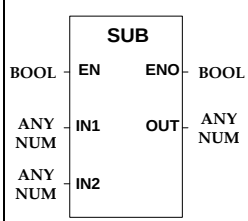
3. Loại biến

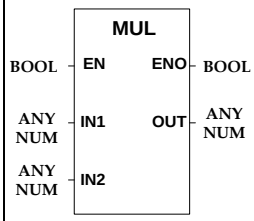
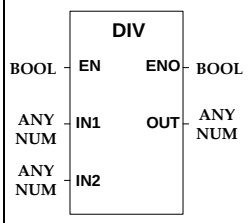
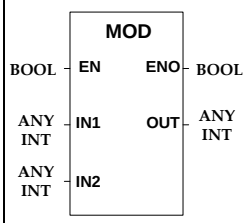
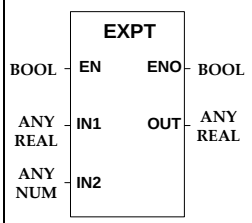
TT	Loại biến	Ý nghĩa
1	VAR	Loại chung cho đọc và ghi
2	VAR_RETAIN	Giá trị của biến vẫn giữ được ngay cả trong trường hợp mất nguồn
3	VAR_CONSTANT	Các biến chỉ đọc
4	VAR_EXTERNAL	Biến gán cho biến ngoài (VAR_GLOBAL)

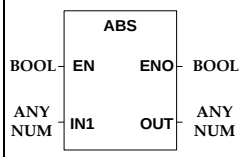
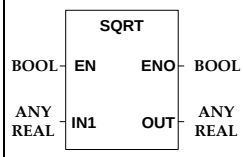
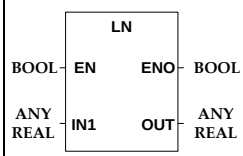
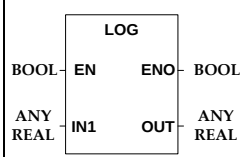
Phụ lục 3

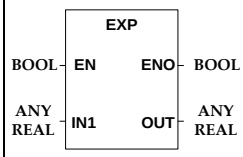
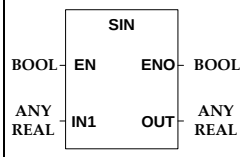
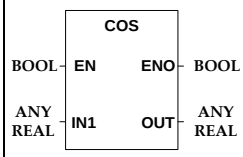
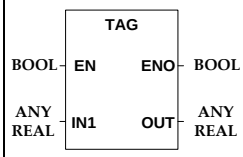
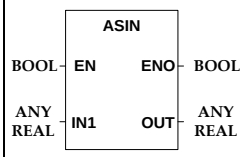
DANH SÁCH MỘT SỐ HÀM HAY SỬ DỤNG

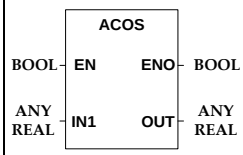
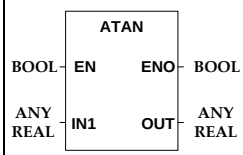
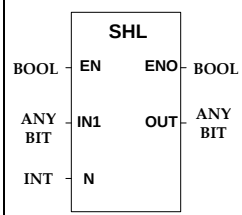
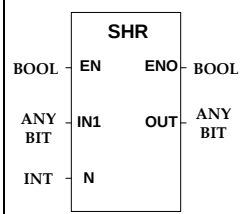
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
Hàm chuyển đổi dữ liệu	***TO***		Hàm chuyển đổi dữ liệu **EN:** Hàm thực hiện khi EN =1 **IN:** Dữ liệu vào **OUT:** Dữ liệu ra **ENO:** Bằng 1 khi không xảy ra lỗi **Các loại hàm chuyển đổi dữ liệu:** SINT_TO_INT INT_TO_SINT DINT_TO_SINT LINT_TO_SINT USINT_TO_SINT UINT_TO_SINT UDINT_TO_SINT ULINT_TO_SINT BYTE_TO_SINT WORD_TO_SINT DWORD_TO_SINT LWORD_TO_SINT BCD_TO_SINT

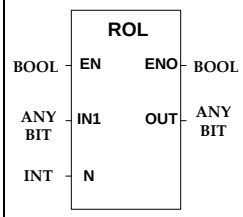
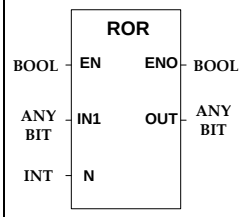
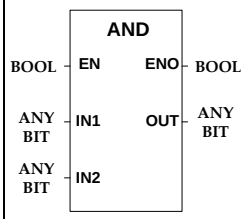
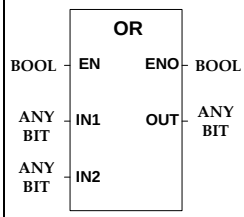
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			REAL_TO_SINT LREAL_TO_SINT STRING_TO_SINT NUM_TO_STRING TIME_TO_UDINT DATE_TO_UINT TOD_TO_UDINT DT_TO_DATE
	TRUNC		Chuyển đổi từ số thực sang số nguyên **EN:** Hàm thực hiện khi EN =1 **IN:** Dữ liệu vào **OUT:** Dữ liệu ra **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm thực hiện các phép toán học	ADD		Hàm thực hiện phép tính cộng giữa các số nguyên **EN:** Hàm thực hiện khi EN =1 **IN1~IN8:** Các toán hạng **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	SUB		Hàm thực hiện phép tính trừ giữa hai số nguyên **EN:** Hàm thực hiện khi EN =1 **IN1:** Số bị trừ **IN2:** Số trừ **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi

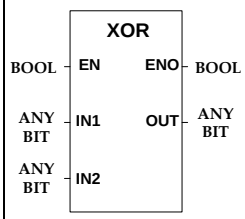
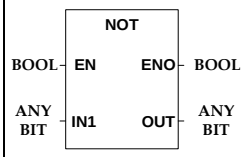
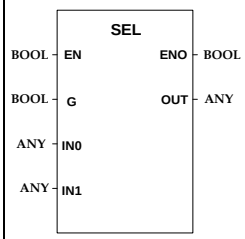
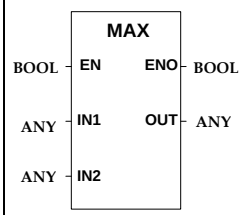
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	MUL		Hàm thực hiện phép tính nhân giữa hai số nguyên **EN:** Hàm thực hiện khi EN =1 **IN1~IN8:** Các toán hạng **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	DIV		Hàm thực hiện phép tính chia giữa hai số nguyên **EN:** Hàm thực hiện khi EN =1 **IN1:** Số bị chia **IN2:** Số chia **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	MOD		Hàm thực hiện chia lấy phần dư giữa hai số nguyên **EN:** Hàm thực hiện khi EN =1 **IN1:** Số bị chia **IN2:** Số chia **OUT:** Số dư **ENO:** Bằng 1 khi không xảy ra lỗi
	EXPT		Hàm mũ với mọi số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Cơ số **IN2:** Số mũ **OUT:** Kết quả

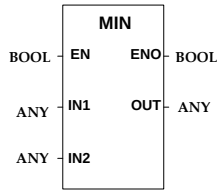
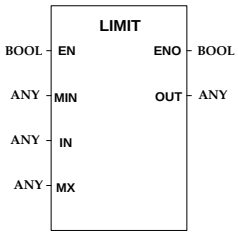
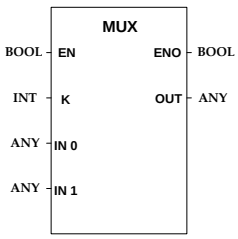
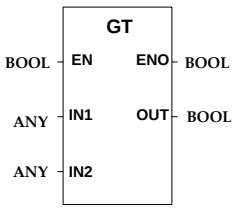
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			ENO: Bằng 1 khi không xảy ra lỗi
	ABS		Hàm lấy giá trị tuyệt đối một số nguyên **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	SQRT		Hàm lấy căn bậc hai của mọi số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	LN		Hàm lấy log Nepe của một số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	LOG		Hàm lấy lôgarit cơ số 10 của một số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi

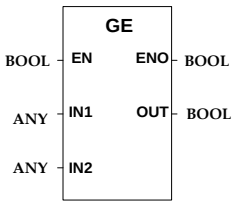
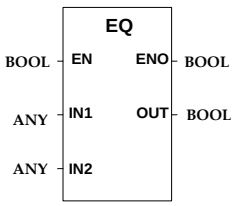
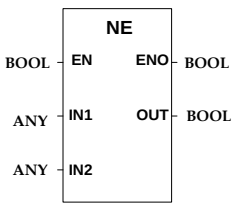
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	EXP		Hàm e mũ tự nhiên **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm lượng giác	SIN		Hàm lấy giá trị SIN của một số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	COS		Hàm lấy giá trị COSIN của một số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	TAN		Hàm lấy giá trị TANGENT của một số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	ASIN		Hàm lấy giá trị ARCH SIN của một số thực **EN:** Hàm thực hiện khi EN =1

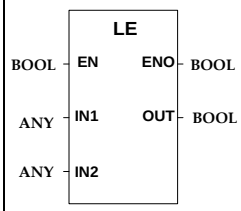
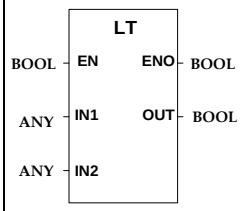
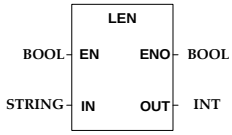
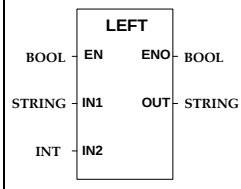
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			IN1: Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	ACOS		Hàm lấy giá trị ARCH COSIN của một số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả ENO: Bằng 1 khi không xảy ra lỗi
	ATAN		Hàm lấy giá trị ARCH TANGENT của một số thực **EN:** Hàm thực hiện khi EN =1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm dịch bit	SHL		Hàm dịch các bit sang trái N vị trí. **EN:** Hàm thực hiện khi EN =1 **IN:** Dữ liệu đầu vào **N**: Số vị trí cần dịch **OUT:** Dữ liệu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
	SHR		Hàm dịch các bit sang phải N vị trí. **EN:** Hàm thực hiện khi EN =1 **IN:** Dữ liệu đầu vào **N**: Số vị trí cần dịch

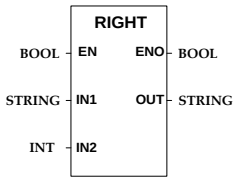
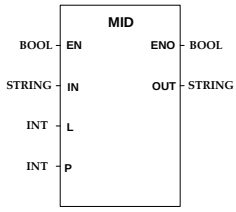
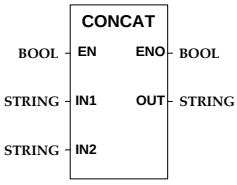
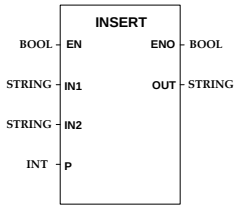
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			OUT: Dữ liệu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm quay bit	ROL		Hàm quay trái N bit **EN:** Hàm thực hiện khi EN = 1 **IN:** Dữ liệu đầu vào **N:** Số lượng vị trí cần quay **OUT:** Dữ liệu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
	ROR		Hàm quay phải N bit **EN:** Hàm thực hiện khi EN = 1 **IN:** Dữ liệu đầu vào **N:** Số lượng vị trí cần quay **OUT:** Dữ liệu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm thực hiện phép tính logic	AND		Hàm thực hiện phép "và" **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	OR		Hàm thực hiện phép "hoặc" **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Giá trị đầu vào **OUT:** Kết quả **ENO: Bằng 1 khi không xảy ra lỗi**

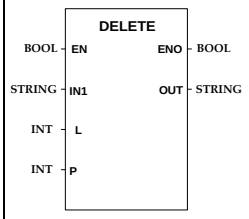
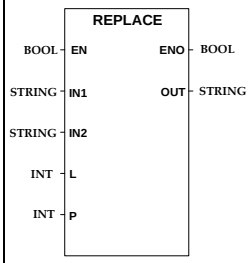
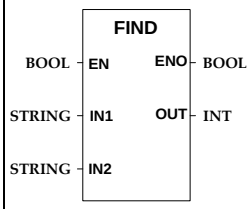
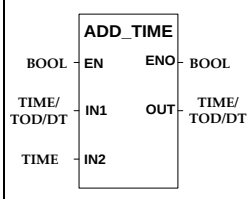
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	XOR		Hàm thực hiện phép "hoặc loại trừ" **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	NOT		Hàm thực hiện phép "phủ định" **EN:** Hàm thực hiện khi EN = 1 **IN1:** Giá trị đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm lựa chọn	SEL		Hàm chọn 1 trong 2 giá trị đầu vào đối với mọi kiểu dữ liệu **EN:** Hàm thực hiện khi EN = 1 **G:** Chọn đầu ra (BOOL) **IN0:** Được chọn khi G = 0 **IN1:** Được chọn khi G = 1 **OUT:** Giá trị ra **ENO:** Bằng 1 khi không xảy ra lỗi
	MAX		Hàm lấy ra giá trị lớn nhất trong số các giá trị nguyên đầu vào **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Các giá trị đầu vào **OUT:** Giá trị lớn nhất **ENO:** Bằng 1 khi không xảy ra lỗi

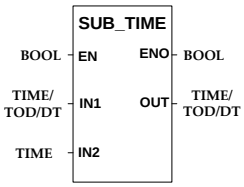
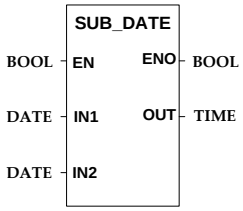
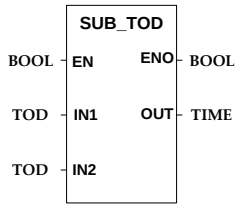
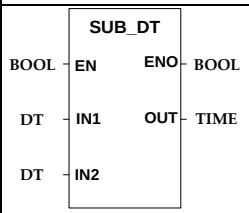
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	MIN		Hàm lấy ra giá trị nhỏ nhất trong số các giá trị nguyên đầu vào **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Các giá trị đầu vào **OUT:** Giá trị lớn nhất **ENO:** Bằng 1 khi không xảy ra lỗi
	LIMIT		Hàm lấy giới hạn trên và dưới đối với một dãy số nguyên **EN:** Hàm thực hiện khi EN = 1 **MN:** Giới hạn trên **IN:** Dữ liệu đầu vào **MX**: Giới hạn dưới **OUT:** Giá trị đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
	MUX		Hàm lựa chọn đầu ra từ các giá trị đầu vào (tối đa là 7 đầu vào) đối với mọi kiểu dữ liệu ứng với giá trị của K (từ 0 tới 6). **EN:** Hàm thực hiện khi EN = 1 **K:** Giá trị chọn đầu ra **IN0~IN6:** Dữ liệu đầu vào **OUT:** Dữ liệu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm so sánh	GT(>)		Phép so sánh lớn hơn. Nếu IN1 > IN2 > … > IN8 thì đầu ra OUT có trạng thái là ON. **EN:** Hàm thực hiện khi EN = 1

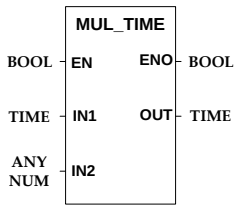
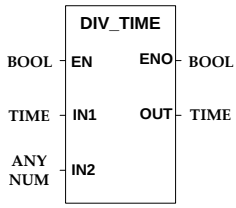
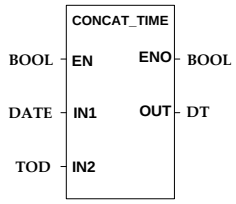
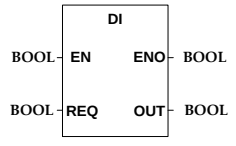
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			IN1~IN8: Dữ liệu đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	GE(>=)		Phép so sánh lớn hơn hoặc bằng. Nếu IN1 ≥ IN2 ≥ … ≥ IN8 thì đầu ra OUT có trạng thái là ON. **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Dữ liệu đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	EQ(=)		Phép so sánh bằng. Nếu IN1 = IN2 = … = IN8 thì đầu ra OUT có trạng thái là ON. **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Dữ liệu đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	NE(#)		Phép so sánh không bằng với mọi kiểu dữ liệu. Nếu IN1 # IN2 thì trạng thái đầu ra OUT là ON. **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi

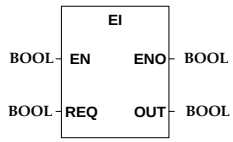
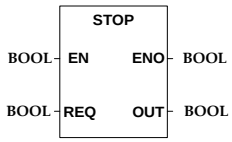
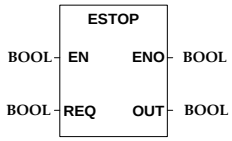
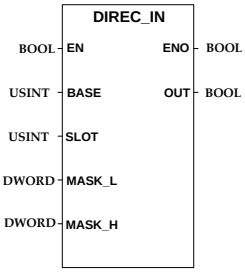
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	LE(=<)		Phép so sánh nhỏ hơn hoặc bằng. Nếu IN1 ≤ IN2 ≤ … ≤ IN8 thì đầu ra OUT có trạng thái là ON. **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Dữ liệu đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
	LT(<)		Phép so sánh nhỏ hơn. Nếu IN1 < IN2 < … < IN8 thì đầu ra OUT có trạng thái là ON. **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:a** Dữ liệu đầu vào **OUT:** Kết quả **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm liên quan tới chuỗi ký tự	LEN		Hàm tìm độ dài của một chuỗi ký tự. **EN:** Hàm thực hiện khi EN = 1 **IN1:** Xâu đầu vào. **OUT:** Chiều dài của xâu. **ENO:** Bằng 1 khi không xảy ra lỗi
	LEFT		Hàm lấy xâu con bên trái **EN:** Hàm thực hiện khi EN = 1 **IN1:** Xâu đầu vào. **L:** Chiều dài xâu con **OUT:** Kết quả. **ENO:** Bằng 1 khi không xảy ra lỗi

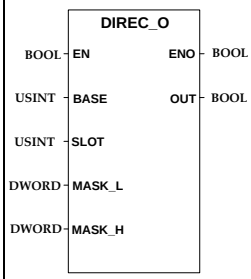
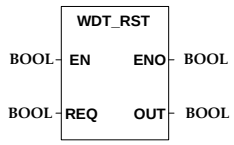
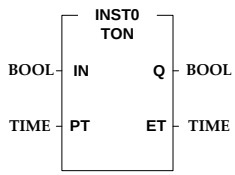
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	RIGHT		Hàm lấy xâu con bên phải **EN:** Hàm thực hiện khi EN = 1 **IN1:** Xâu đầu vào. **L:** Chiều dài xâu con **OUT:** Xâu con. **ENO:** Bằng 1 khi không xảy ra lỗi
	MID		Hàm lấy xâu con từ giữa **EN:** Hàm thực hiện khi EN = 1 **IN1:** Chuỗi ký tự đầu vào. **L:** Chiều dài xâu con **P:** Vị trí bắt đầu của xâu con **OUT:** Chiều dài của chuỗi. **ENO:** Bằng 1 khi không xảy ra lỗi
	CONCAT		Hàm nối các xâu với nhau **EN:** Hàm thực hiện khi EN = 1 **IN1~IN8:** Các xâu đầu vào **OUT:** Xâu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
	INSERT		Hàm chèn xâu **EN:** Hàm thực hiện khi EN = 1 **IN1:** Xâu được chèn **IN2:** Xâu cần chèn **P:** Vị trí cần chèn **OUT:** Xâu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi

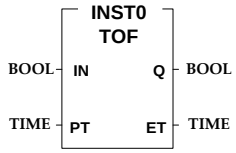
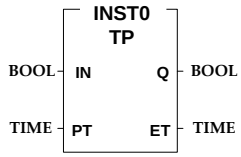
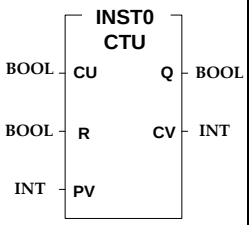
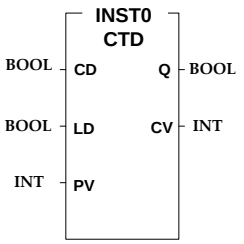
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	DELETE		Hàm xóa một xâu con **EN:** Hàm thực hiện khi EN = 1 **IN1:** Xâu đầu vào **L:** Chiều dài xâu cần xóa **P:** Vị trí bắt đầu của xâu cần xoá **OUT:** Xâu đầu ra **ENO:** Bằng 1 khi không xảy ra lỗi
	REPLACE		Hàm thay thế xâu **EN:** Hàm thực hiện khi EN = 1 **IN1:** Xâu đầu vào **IN2:** Xâu thay **P**: Vị trí bắt đầu xâu muốn thay **L:** Chiều dài xâu muốn thay **OUT**: Xâu ra **ENO:** Bằng 1 khi không xảy ra lỗi
	FIND		Hàm tìm xâu **EN:** Hàm thực hiện khi EN = 1 **IN1:** Xâu đầu vào **IN2**: Xâu cần tìm **OUT:** Xâu ra **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm liên quan tới thời gian và ngày tháng	ADD_TIME		Hàm thực hiện phép cộng thời gian. **EN:** Hàm thực hiện khi EN = 1 **IN1:** Đầu vào 1 (TIME, TOD, DT). **IN2:** Đầu vào 2 (TIME)

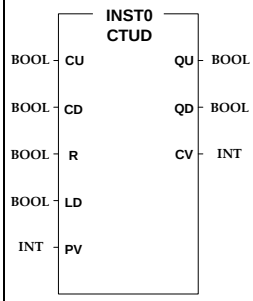
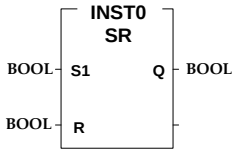
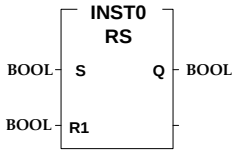
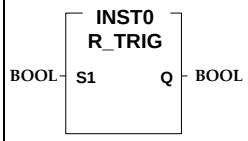
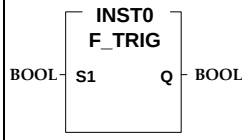
Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			OUT: Đầu ra (TIME, TOD, DT) **ENO:** Bằng 1 khi không xảy ra lỗi
	SUB_TIME		Hàm thực hiện phép trừ thời gian **EN:** Hàm thực hiện khi EN = 1 **IN1:** Số bị trừ (TIME, TOD, DT). **IN2:** Số trừ (TIME) **OUT:** Đầu ra (TIME, TOD, DT) **ENO:** Bằng 1 khi không xảy ra lỗi
	SUB_DATE		Hàm thực hiện phép trừ ngày **EN:** Hàm thực hiện khi EN = 1 **IN1:** Số bị trừ (DATE). **IN2:** Số trừ (DATE) **OUT:** Đầu ra (TIME) **ENO:** Bằng 1 khi không xảy ra lỗi
	SUB_TOD		Hàm thực hiện phép trừ thời gian của ngày. **EN:** Hàm thực hiện khi EN = 1 **IN1:** Số bị trừ (TOD) **IN2:** Số trừ (TOD) **OUT:** Đầu ra (TIME) **ENO:** Bằng 1 khi không xảy ra lỗi
	SUB_DT		Hàm thực hiện phép trừ thời gian và ngày **EN:** Hàm thực hiện khi EN = 1 **IN1:** Số bị trừ (DATE & TIME)

Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			IN2: Số trừ (DATE & TIME) **OUT:** Đầu ra (TIME) **ENO:** Bằng 1 khi không xảy ra lỗi
	MUL_TIME		Hàm thực hiện phép nhân thời gian. **EN:** Hàm thực hiện khi EN = 1 **IN1:** Đầu vào (TIME) **IN2:** Hệ số nhân (INT) **OUT:** Kết quả (TIME) **ENO:** Bằng 1 khi không xảy ra lỗi
	DIV_TIME		Hàm thực hiện phép chia thời gian **EN:** Hàm thực hiện khi EN = 1 **IN1:** Đầu vào (TIME) **IN2:** Hệ số chia (INT) **OUT:** Kết quả (TIME) **ENO:** Bằng 1 khi không xảy ra lỗi
	CONCAT_TIME		Hàm thực hiện phép nối ngày và thời gian **EN:** Hàm thực hiện khi EN = 1 **IN1:** Ngày (DATE) **IN2:** Thời gian trong ngày (TOD) **OUT:** Kết quả (DT) **ENO:** Bằng 1 khi không xảy ra lỗi
Hàm điều khiển hệ thống	DI		Khối yêu cầu ngắt **EN:** Hàm thực hiện khi EN = 1 **REQ:** Yêu cầu thực hiện tác vụ

Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			OUT: Xác nhận hàm đã thực hiện **ENO:** Bằng 1 khi không xảy ra lỗi
	EI		Khối cho phép ngắt **EN:** Hàm thực hiện khi EN = 1 **REQ:** Yêu cầu thực hiện ngắt **OUT:** Xác nhận hàm đã thực hiện **ENO:** Bằng 1 khi không xảy ra lỗi
	STOP		Khối yêu cầu dừng hoạt động của PLC **EN:** Hàm thực hiện khi EN = 1 **REQ:** Yêu cầu thực hiện hàm **OUT:** Xác nhận hàm đã thực hiện **ENO:** Bằng 1 khi không xảy ra lỗi
	ESTOP		Khối yêu cầu dừng PLC khẩn cấp **EN:** Hàm thực hiện khi EN = 1 **REQ:** Yêu cầu thực hiện hàm **OUT:** Xác nhận hàm đã thực hiện **ENO:** Bằng 1 khi không xảy ra lỗi
	DIREC_IN		Cập nhật dữ liệu đầu vào tức thời **EN:** Hàm thực hiện khi EN = 1 **BASE:** Số chỉ của mô-đun đầu vào **SLOT:** Số chỉ khe cắm mô-đun đầu vào **MASK_L:** Chỉ định bit không được cập nhập của 32 bit vị trí thấp trong DWORD. **MASK_H:** Chỉ định bit không được cập nhập của 32 bit vị trí cao trong

Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
			DWORD. **OUT:** Bằng 1 nếu cập nhật thành công. **ENO:** Bằng 1 khi không xảy ra lỗi
	DIREC_O		Cập nhật dữ liệu đầu ra tức thời **EN:** Hàm thực hiện khi EN = 1 **BASE:** Số chỉ của mô-đun đầu vào **SLOT:** Số chỉ khe cắm mô-đun đầu vào **MASK_L:** Chỉ định bit không được cập nhập của 32 bit vị trí thấp trong DWORD. **MASK_H:** Chỉ định bit không được cập nhập của 32 bit vị trí cao trong DWORD. **OUT:** Bằng 1 nếu cập nhật thành công. **ENO:** Bằng 1 khi không xảy ra lỗi
	WDT_RST		Khởi tạo bộ định thời Watch Dog Timer. **EN:** Hàm thực hiện khi EN = 1 **REQ:** Yêu cầu khởi tạo **OUT:** Bằng 1 nếu khởi tạo thành công. **ENO:** Bằng 1 khi không xảy ra lỗi
Bộ định thời	TON		Bộ định thời tạo trễ **IN:** Hàm thực hiện khi EN = 1 **PT:** Thời gian đặt trước **Q:** Đầu ra **ET:** Giá trị đếm hiện tại

Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	TOF		Bộ định thời tạo trễ ngắt **IN:** Hàm thực hiện khi EN = 1 **PT:** Thời gian đặt trước **Q:** Đầu ra **ET:** Giá trị đếm hiện tại
	TP		Bộ định thời tạo xung **IN:** Hàm thực hiện khi EN = 1 **PT:** Thời gian đặt trước **Q:** Đầu ra **ET:** Giá trị đếm hiện tại
Bộ đếm sự kiện	CTU		Bộ đếm tiến **CU:** Đầu vào tín hiệu **R:** Đầu vào khởi động lại bộ đếm. **PV:** Giá trị đặt trước **Q:** Đầu ra **CV:** Giá trị đếm hiện thời
	CTD		Bộ đếm lùi **CD:** Đầu vào tín hiệu **LD:** Đọc giá trị đặt trước **PV:** Giá trị định trước **Q:** Đầu ra **CV:** Giá trị đếm hiện thời

Loại hàm	Lệnh	Kí hiệu	Mô tả chức năng
	CTUD		Bộ đếm tiến – lùi **CU:** Đầu vào đếm tiến **CD:** Đầu vào đếm lùi **R:** Đầu vào khởi động lại bộ đếm **LD:** Đọc giá trị đặt trước **PV:** Giá trị đặt trước **QU:** Đầu ra bộ đếm tiến **QD:** Đầu ra đếm lùi **CV:** Giá trị đếm hiện thời
Khối chức năng	SR		Khối ưu tiên đầu vào SET **S1:** Tín hiệu SET **R:** Tín hiệu RESET **Q1:** Đầu ra
	RS		Khối ưu tiên đầu vào RESET **S:** Tín hiệu SET **R1:** Tín hiệu RESET **Q:** Đầu ra
	R_TRIG		Khối dò xung sườn trước **CLK:** Xung đầu vào **Q:** Đầu ra
	F_TRIG		Khối dò xung sườn sau **CLK:** Đầu vào **Q:** Đầu ra

Phụ lục 4
BÀI TẬP THỰC HÀNH

Đối với những bài tập dưới đây, yêu cầu sử dụng phần mêm GMWIN để lập trình PLC theo sơ đồ bậc thang và chạy mô phỏng trên bộ PLC ED-4260 Trainer.

Bài 1: Điều khiển động cơ chuyển động tuyến tính

1. *Chương trình được hoạt động theo nguyên tắc*

   a) Đầu vào gồm 2 nút (nút khởi động và nút dừng), 2 cảm biến nhận biết vị trí của động cơ. Đầu ra của PLC được kết nối tới động cơ.

   b) Khi nhấn nút khởi động, động cơ bắt đầu hoạt động và di chuyển xuống. Khi nhận được tín hiệu từ cảm biến giới hạn dưới thì động cơ sẽ di chuyển lên cho tới khi cảm biến nhận biết vị trí giới hạn trên nhận được tín hiệu thì nó lại di chuyển ngược lại và cứ như thế quá trình được lặp lại.

   c) Trong quá trình di chuyển nếu nút dừng được nhấn thì động cơ sẽ dừng hoạt động.

2. *Vẽ sơ đồ kết nối các đầu vào/ra với PLC.*

3. *Viết chương trình sử dụng phần mêm GMWIN và vẽ lại sơ đồ vào bảng sau.*

SƠ ĐỒ BẬC THANG	
TÊN CHƯƠNG TRÌNH :	NGÀY VIẾT:
Dòng 1	
Dòng 2	
Dòng 3	
Dòng 4	
Dòng 5	
Dòng 6	
Dòng 7	
Dòng 8	
Dòng 9	
Dòng 10	
Dòng 11	
Dòng 12	
Dòng 13	
Dòng 14	
Dòng 15	
Dòng 16	
Dòng 17	
Dòng 18	
Dòng 19	
Dòng 20	

Bài 2. Thiết kế chương trình mô tả hoạt động của đồng hồ điện tử

1. *Chương trình được hoạt động theo nguyên tắc*

   a) Đầu vào gồm 2 nút nhấn (nút khởi động và nút dừng), đầu ra gồm khối hiển thị và khối đầu ra.

   b) Khi nhấn nút khởi động thì bộ định thời bắt đầu đếm và giá trị thời gian được hiển thị ra khối hiển thị.

   c) Khi nút dừng được nhấn thì bộ định thời ngừng hoạt động.

2. *Vẽ sơ đồ kết nối giữa các đầu vào/ra với PLC*

3. *Viết chương trình sử dụng GMWIN và vẽ lại sơ đồ vào bảng sau*

SƠ ĐỒ BẬC THANG	
TÊN CHƯƠNG TRÌNH :	NGÀY VIẾT:
Dòng 1	
Dòng 2	
Dòng 3	
Dòng 4	
Dòng 5	
Dòng 6	
Dòng 7	
Dòng 8	
Dòng 9	
Dòng 10	
Dòng 11	
Dòng 12	
Dòng 13	
Dòng 14	
Dòng 15	
Dòng 16	
Dòng 17	
Dòng 18	
Dòng 19	
Dòng 20	

Bài 3. Chương trình dùng bộ định thời ngòai

1. Chương trình được hoạt động theo nguyên tắc

a) Đầu vào gồm một công tắc đầu vào số và một nút nhấn khởi động. Đầu ra điều khiển đèn và khối hiển thị.

b) Giá trị đặt trước cho bộ định thời (PV) được đặt bởi công tắc đầu vào số. Khi nút khởi động được nhấn, bộ định thời bắt đầu đếm và được hiển thị trên khối hiển thị.

c) Đèn sẽ sang khi giá trị đếm hiện tại (ET) của bộ định thời đạt đến giá trị đặt trước.

d) Gợi ý đầu vào/ra của PLC được cho như bảng

Tên biến	Kiểu biến	Kiểu dữ liệu	Địa chỉ bộ nhớ
C1	VAR	FB Instance	
Data	VAR	INT	
Data 1	VAR	UDINT	
Data 2	VAR	UDINT	
Digital Switch	VAR	WORD	AT %IW0.1.0
Display	VAR	WORD	AT%QW0.2.0
ET – Data	VAR	TIME	
ET– Data1	VAR	UDINT	
ET– Data2	VAR	UDINT	
ET– Data3	VAR	INT	
LAMP	VAR	BOOL	AT %QX0.3.0
RESET	VAR	BOOL	
START	VAR	BOOL	AT %IX0.0.0
STOP	VAR	BOOL	AT%IX0.0.1
T1	VAR	FB Instance	

T1	VAR	FB Instance	
TIME DATA	VAR	Time	

2. *Vẽ sơ đồ kết nối giữa các đầu vào/ra với PLC*
3. *Viết chương trình sử dụng GMWIN và vẽ lại sơ đồ vào bảng sau*

SƠ ĐỒ BẬC THANG	
TÊN CHƯƠNG TRÌNH :	NGÀY VIẾT:
Dòng 1	
Dòng 2	
Dòng 3	
Dòng 4	
Dòng 5	
Dòng 6	
Dòng 7	
Dòng 8	
Dòng 9	
Dòng 10	
Dòng 11	
Dòng 12	
Dòng 13	
Dòng 14	
Dòng 15	
Dòng 16	
Dòng 17	
Dòng 18	
Dòng 19	
Dòng 20	

Bài 4. Chương trình dùng bộ định thời ngòai

1. Chương trình được hoạt động theo nguyên tắc

a) Đầu vào gồm một công tắc đầu vào số, một nút nhấn khởi động và một nút nhấn dừng. Đầu ra điều khiển đèn và khối hiển thị.

b) Giá trị đặt trước cho bộ định thời (TP) được đặt bởi công tắc đầu vào số. Khi nút khởi động được nhấn, bộ định thời bắt đầu đếm và được hiển thị trên khối hiển thị (khối hiển thị có thể hiện thỉ tới 7 chữ số). Đèn sẽ sáng khi giá trị đếm hiện tại (ET) của bộ định thời đạt đến giá trị đặt trước.

c) Quá trình được khởi tạo khi nhấn nút dừng.

d) Gợi ý đầu vào/ra của PLC được cho như bảng

Thành phần	Tên biến	Địa chỉ bộ nhớ	Chú thích
Đầu vào	START	%IX0.0.0	Push_Switch S-2
	PT	%IW0.1.0	Digital Switch
	STOP	%IX0.0.1	Push_Switch S-3
Đầu ra	LAMP	%QX0.3.0	Lamp L -1
	DISPLAY	%QW0.2.0	Display Out

Tên biến	Kiểu biến	Kiểu dữ liệu	Địa chỉ bộ nhớ
C1	VAR	FB Instance	
Data	VAR	INT	
Data 1	VAR	UDINT	
Data 2	VAR	UDINT	
Digital Switch	VAR	WORD	AT %IW0.1.0
Display	VAR	WORD	AT%QW0.2.0
ET – Data	VAR	TIME	
ET– Data1	VAR	UDINT	
ET– Data2	VAR	UDINT	
ET– Data3	VAR	INT	
LAMP	VAR	BOOL	AT %QX0.3.0
RESET	VAR	BOOL	
START	VAR	BOOL	AT %IX0.0.0
STOP	VAR	BOOL	AT%IX0.0.1
T1	VAR	FB Instance	

Tên biến	Kiểu biến	Kiểu dữ liệu	Địa chỉ bộ nhớ
T1	VAR	FB Instance	
TIME DATA	VAR	Time	

2. *Vẽ sơ đồ kết nối giữa các đầu vào/ra với PLC*
3. *Viết chương trình sử dụng GMWIN và vẽ lại sơ đồ vào bảng sau*

SƠ ĐỒ BẬC THANG	
TÊN CHƯƠNG TRÌNH :	NGÀY VIẾT:
Dòng 1	
Dòng 2	
Dòng 3	
Dòng 4	
Dòng 5	
Dòng 6	
Dòng 7	
Dòng 8	
Dòng 9	
Dòng 10	
Dòng 11	
Dòng 12	
Dòng 13	
Dòng 14	
Dòng 15	
Dòng 16	
Dòng 17	
Dòng 18	
Dòng 19	
Dòng 20	

Bài 5. Chương trình dùng các lệnh MOVE, COUTER

1. *Chương trình được hoạt động theo nguyên tắc*

   a) Đầu vào gồm một nút nhấn khởi động, một nút nhấn dừng. Đầu ra là khối hiển thị.

   b) Chương trình sử dụng counter (TON), hàm chức năng chuyển đổi (INT_TO_BCD), và cờ lệnh.

   c) Khi nhấn nút khởi động, COUTER bắt đầu đếm, va khi đạt tới giá trị được cài sẵn (PT) thì bắt đầu đếm lại.

   d) Bộ đếm COUTER sẽ dừng, khi nút dừng được nhấn, và khi đó giá của COUTER được hiển thị trên khối hiển thị OUTPUT DISPLAY.

   e) Thiết kế lại chương trình theo cách mà khi nút khởi động được nhấn lần nữa thì COUTER mới bắt đầu đếm lại.

2. *Vẽ sơ đồ kết nối các đầu vào/ra với PLC*

3. *Viết chương trình sử dụng GMWIN và vẽ lại sơ đồ vào bảng sau*

SƠ ĐỒ BẬC THANG	
TÊN CHƯƠNG TRÌNH :	NGÀY VIẾT:
Dòng 1	
Dòng 2	
Dòng 3	
Dòng 4	
Dòng 5	
Dòng 6	
Dòng 7	
Dòng 8	
Dòng 9	
Dòng 10	
Dòng 11	
Dòng 12	
Dòng 13	
Dòng 14	
Dòng 15	
Dòng 16	
Dòng 17	
Dòng 18	
Dòng 19	
Dòng 20	

Bài 6. Chương trình sử dụng bộ định thời TP

1. *Chương trình được hoạt động theo nguyên tắc*

   a) Đầu vào là nút nhấn và đầu ra là đèn hiển thị.

   b) Chương trình sử dụng TP Timer, một cuộn cảm biến có điểm tiếp xúc theo hướng thuận, và một cuộn cảm biến ngược.

   c) Khi nút nhấn được ấn, đèn hiển thị sáng. Đèn hiển thị tự tắt sau 3s, sau đó tự sáng lại.

2. *Vẽ sơ đồ kết nối các đầu vào/ra với PLC*

3. *Viết chương trình sử dụng GMWIN và vẽ lại sơ đồ vào bảng sau*

SƠ ĐỒ BẬC THANG	
TÊN CHƯƠNG TRÌNH :	NGÀY VIẾT:
Dòng 1	
Dòng 2	
Dòng 3	
Dòng 4	
Dòng 5	
Dòng 6	
Dòng 7	
Dòng 8	
Dòng 9	
Dòng 10	
Dòng 11	
Dòng 12	
Dòng 13	
Dòng 14	
Dòng 15	
Dòng 16	
Dòng 17	
Dòng 18	
Dòng 19	
Dòng 20	

TÀI LIỆU THAM KHẢO

1. Nguyễn Kim Giao, *Kỹ thuật điện tử số*. NXB Đại học QGHN, MS: 1K – 44 ĐH2006, 2006.
2. Nguyễn Văn Hoà, Bùi Đăng Thảnh, Hoàng Sỹ Hồng, *Giáo trình đo lường điện và cảm biến đo lường,* NXB Giaos dục, MS: 7B612M5-DAI, 2005.
3. ED Corporation, Korea, *Programmable logic controller ED - 4260*.
4. E. A. Parr, *Programmable Controllers, An Engineer's Guide*. 3rd edition, Newnes, ISBN: 978-0750657570, 2003.
5. Frank D.Petruzella. *Programmable logic controllers*, 4th edition, McGraw-Hill Companies, Inc., ISBN: 978-0-07-351088-0, 2011.
6. Gary A. Dunning, *Introduction to Programmable Logic Controllers*, 3rd edition, Thomson/Delmar Learning, ISBN: 978-1401884260, 2005.
7. Gary Kirckof, *Cascading Logic: A Machine Control Methodology for Programmable Logic Controllers*, The Instrumentation, Systems, and Automation Society, ISBN: 978-1556178146, 2002.
8. George L., Jr. Batten, *Programmable Controllers: Hardware, Software, and Applications*, 2nd edition, Mcgraw-Hill, ISBN: 978-0070042148, 1994.
9. Hans Berger. *Automation with STEP 7 in LAD and FBD*, 4th edition, Publicis Coporate Publishing, ISBN: 978-3 89578 297-8, 2008.
10. James A. Rehg , Glenn J. Sartori, *Programmable Logic Controllers*, Pearson Custom Publishing, ISBN: 978-0558082628, 2007.
11. John Ridley, *Mitsubishi FX Programmable Logic Controllers, Applications and Programming*. 2nd edition, Newnes, ISBN 0 7506 56794, 2004.
12. Max Rabiee, *Programmable Logic Controllers: Hardware and Programming - Laboratory Manual,* 3rd edition, Goodheart-Willcox, ISBN: 978-1605259482, 2012.
13. L.A.Bryan, E.A.Bryan. *Programmable Controllers theory and implemention*,2nd edition, Amer Technical Pub, ISBN-13: 978-0826913005, 1997.