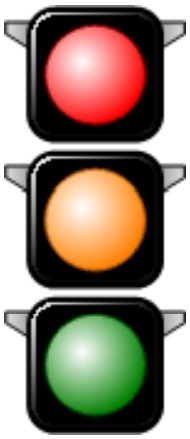
Ch 4: Kỹ thuật lập trình điều khiển trình tự

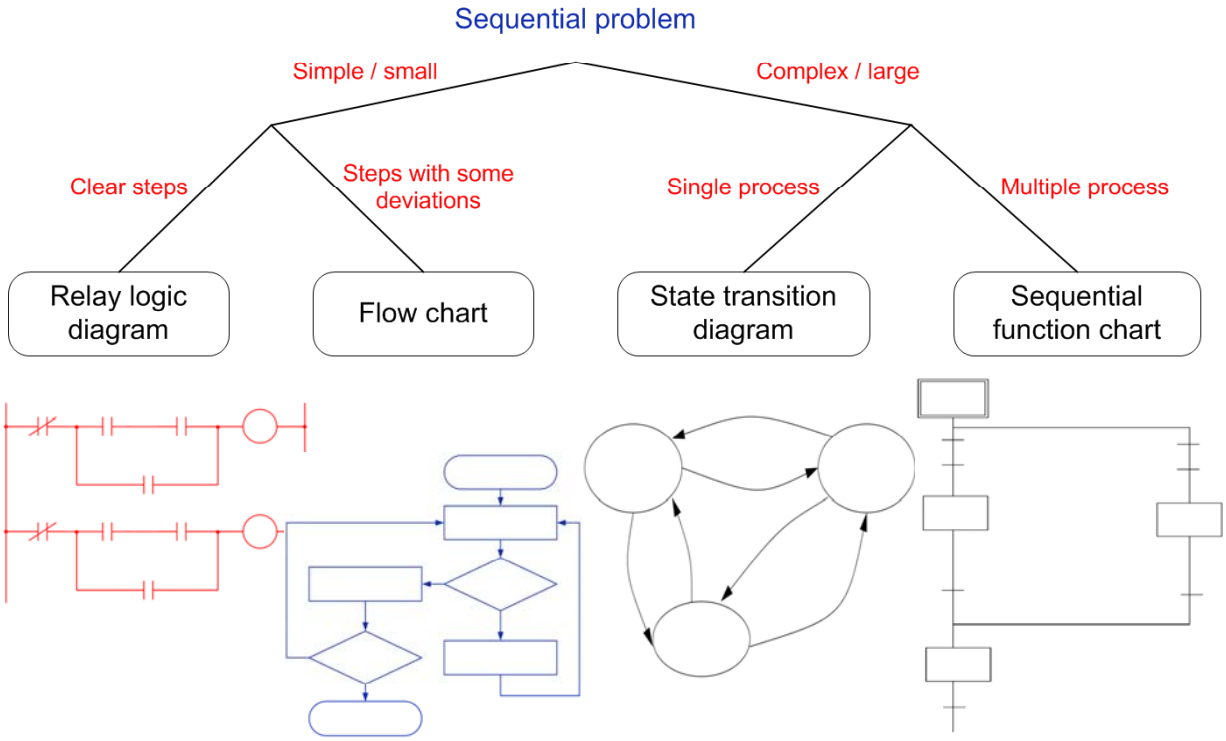
• Nếu chương trình bậc thang được viết theo kiểu: suy nghĩ về quá trình và thực hiện viết chương trình thì điều này luôn dẫn đến tốn rất nhiều công sức để hiệu chỉnh chương trình. Do đó, việc ứng dụng các phương pháp thiết kế giúp cho việc thiết kế chương trình được dễ dàng hơn.



• Hầu hết các hệ thống điều khiển có tính tuần tự. Có nhiều phương pháp thiết kế hệ thống điều khiển trình tự được phát triển, đáp ứng cho các hệ thống trình tự có mức độ phức tạp khác nhau.

• Các bước thiết kế chương trình trình tự cho PLC như sau:
  - Diễn đạt bằng lời quá trình điều khiển
  - Biểu diễn các diễn đạt sang dạng lưu đồ / biểu đồ
  - Xác định các điều kiện logic cho từng trạng thái của quá trình
  - Chuyển các biểu thức logic sang chương trình dạng bậc thang

Ch 4: Kỹ thuật lập trình điều khiển trình tự



4.1. Phương pháp logic rơle

Xét thí dụ thiết kế chương trình điều khiển màn hình đèn chiếu.

- Mô tả quá trình
  - Màn hình sẽ được cuốn lên khi nút nhấn lên được nhấn tức thời
  - Màn hình sẽ được hạ xuống khi nút nhấn xuống được nhấn tức thời
  - Có bố trí hai công tắc hành trình để nhận biết giới hạn trên cùng và dưới cùng
  - Khi bộ điều khiển bắt đầu làm việc, màn luôn được cuốn lên và dừng ở trên cùng

- Nhận diện các bước / trạng thái
  - Step 1: màn hình được cuốn lên cho đến khi cảm biến giới hạn trên bị tác động.
  - Step 2: màn hình đứng yên ở trên và chờ cho đến khi nút nhấn xuống được nhấn.
  - Step 3: màn hình được hạ xuống cho đến khi cảm biến giới hạn dưới bị tác động.
  - Step 4: màn hình đứng yên ở dưới và chờ cho đến khi nút nhấn lên được nhấn.

  (sau đó quay trở lại Step 1)

4.1. Phương pháp logic rơle

- Thiết lập biểu thức logic cho từng bước

Bước 1 tồn tại khi bộ điều khiển bắt đầu hoạt động hoặc khi đang ở bước 4 và có nhấn nút lên. Bước 1 sẽ kết thúc khi bước 2 tồn tại. Vì các nút nhấn là tức thời nên cần có yếu tố duy trì. Do đó, biểu thức logic của bước 1 là:

$$Step\_1 = (First\_scan + Step\_4.Up\_B + Step\_1).\overline{Step\_2}$$

Tương tự, các biểu thức logic cho 3 bước còn lại là:

$$Step\_2 = (Step\_1.Up\_LS + Step\_2).\overline{Step\_3}$$

$$Step\_3 = (Step\_2.Down\_B + Step\_3).\overline{Step\_4}$$

$$Step\_4 = (Step\_3.Down\_LS + Step\_4).\overline{Step\_1}$$

$$Up\_motor = Step\_1$$

$$Down\_motor = Step\_3$$

4.1. Phương pháp logic rơle

- Chuyển sang dạng bậc thang

Device	PC device	Description
Up_B	X000	Up button
Down_B	X001	Down button
Up_LS	X002	Screen is at top position
Down_LS	X003	Screen is at bottom position
Screen motor	Y000	Raise projector screen
	Y001	Lower projector screen

$$Step\_1 = (First\_scan + Step\_4.Up\_B + Step\_1).\overline{Step\_2}$$

4.1. Phương pháp logic rơle

$$Step\_2 = (Step\_1.Up\_LS + Step\_2).\overline{Step\_3}$$

$$Step\_3 = (Step\_2.Down\_B + Step\_3).\overline{Step\_4}$$

4.1. Phương pháp logic rơle

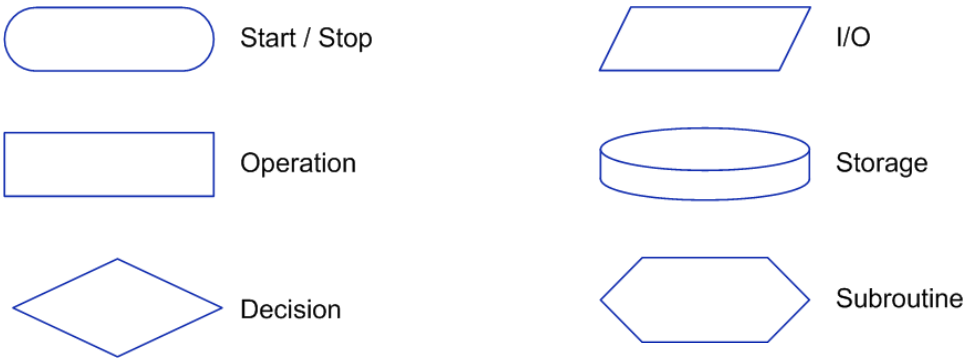
$$Step\_4 = (Step\_3.Down\_LS + Step\_4).\overline{Step\_1}$$

Và với biểu thức ngõ ra: $Up\_motor = Step\_1$ $Down\_motor = Step\_3$

4.2. Phương pháp biểu diễn lưu đồ

Phương pháp biểu diễn lưu đồ rất thường dùng khi thiết kế phần mềm máy tính, đồng thời nó cũng rất phổ biến để biểu diễn trình tự hoạt động của một hệ thống điều khiển. Lưu đồ có quan hệ trực tiếp đến sự mô tả bằng lời hệ thống điều khiển, chỉ ra từng điều kiện cần kiểm tra ở từng bước và các xử lý trong bước đó theo chuỗi trình tự.

Các khối biểu tượng được sử dụng để xây dựng lưu đồ:

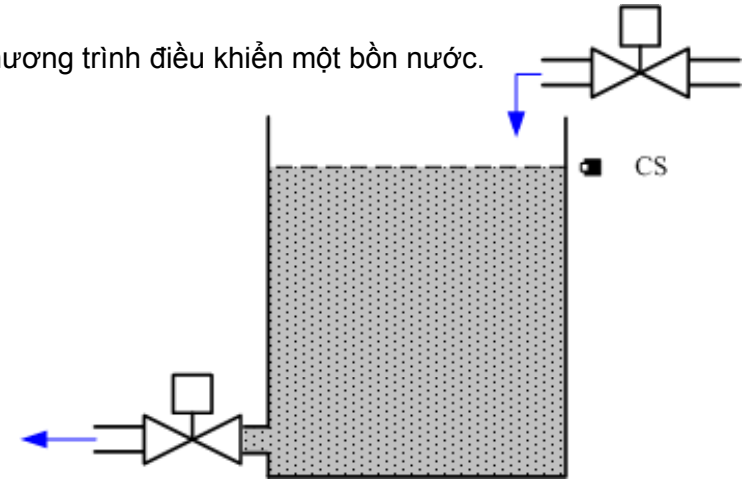


4.2. Phương pháp biểu diễn lưu đồ

Các bước thông thường để xây dựng lưu đồ là:

- Tìm hiểu quá trình
- Xác định các bước / hành động chính và biểu diễn chúng thành các khối
- Xác định trình tự của các bước (khối) thông qua các mũi tên.
- Sử dụng khối quyết định để thực hiện phân nhánh.

Xét thí dụ thiết kế chương trình điều khiển một bồn nước.

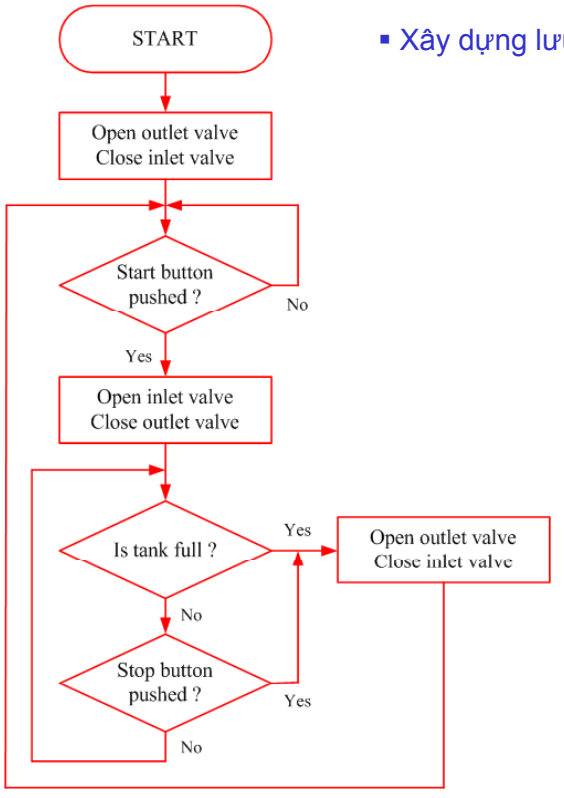


4.2. Phương pháp biểu diễn lưu đồ

▪ Mô tả quá trình

- Khi nút nhấn Start được nhấn, van ngõ vào mở ra cho nước chảy vào bồn và van ngõ ra đóng lại.
- Khi bồn đầy (dựa vào tín hiệu từ cảm biến) hoặc nút Stop được nhấn, van ngõ vào đóng lại và van ngõ ra mở ra cho nước chảy ra khỏi bồn.
- Khi bộ điều bắt đầu làm việc, van ngõ ra luôn được mở và van ngõ vào luôn bị đóng.

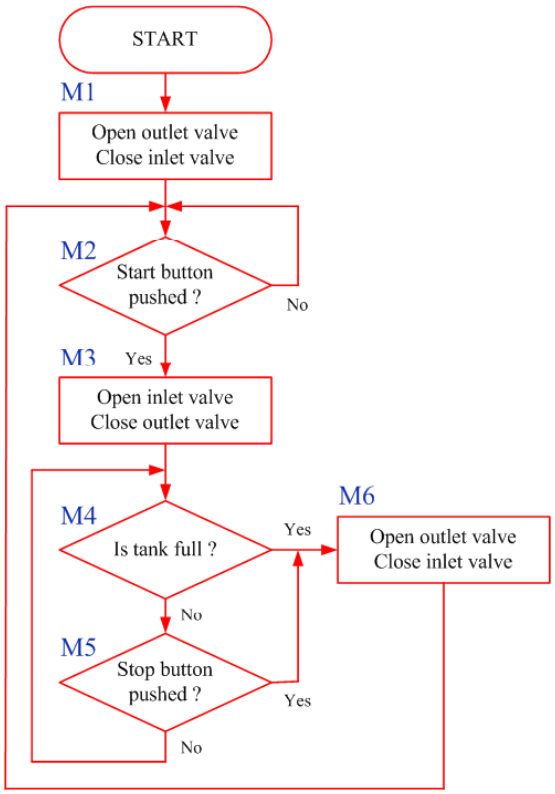
▪ Xây dựng lưu đồ



4.2. Phương pháp biểu diễn lưu đồ - block logic

- Phương pháp 1: Xây dựng logic bậc thang dùng khối lệnh (sử dụng cặp lệnh MC và MCR) .

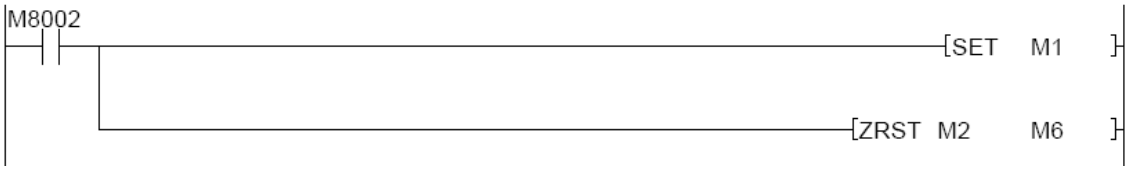
- Bước 1: Gán mỗi nhãn vào từng khối trên lưu đồ



4.2. Phương pháp biểu diễn lưu đồ - block logic

- Bước 2: Vẽ nhánh logic, chuyển chương trình PLC nhảy vào khối thứ nhất khi bộ điều khiển bắt đầu làm việc

Device	PC device	Description
CS	X000	Level sensor
Start	X001	Start button
Stop	X002	Stop button
IV	Y000	Inlet valve
OV	Y001	Outlet valve



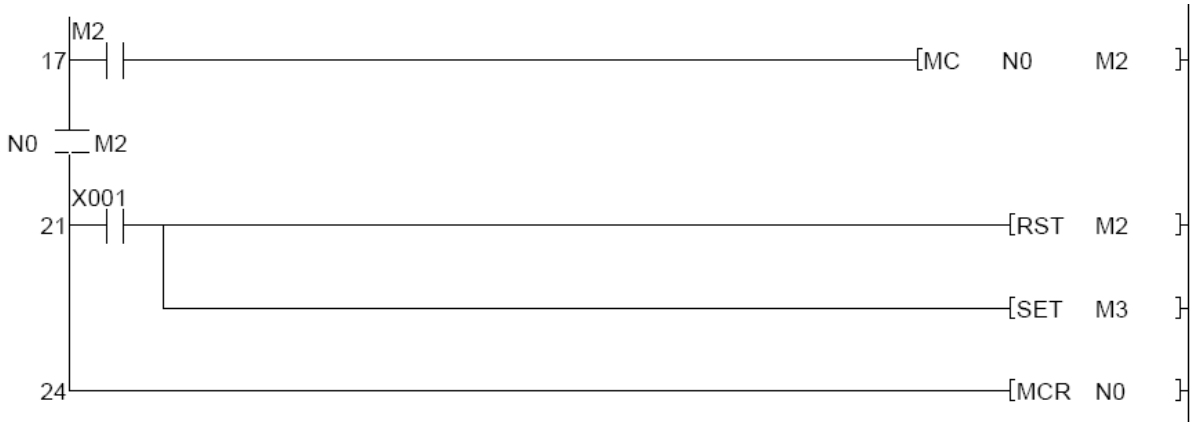
4.2. Phương pháp biểu diễn lưu đồ - block logic

• Bước 3: Vẽ nhánh logic cho mỗi khối trong lưu đồ

Khối thứ nhất khóa van ngõ vào và mở van ngõ ra, sau đó sẽ chuyển sang khối thứ hai, do đó nhánh logic của khối này có thể được biểu diễn như sau:

4.2. Phương pháp biểu diễn lưu đồ - block logic

Đối với khối thứ hai:

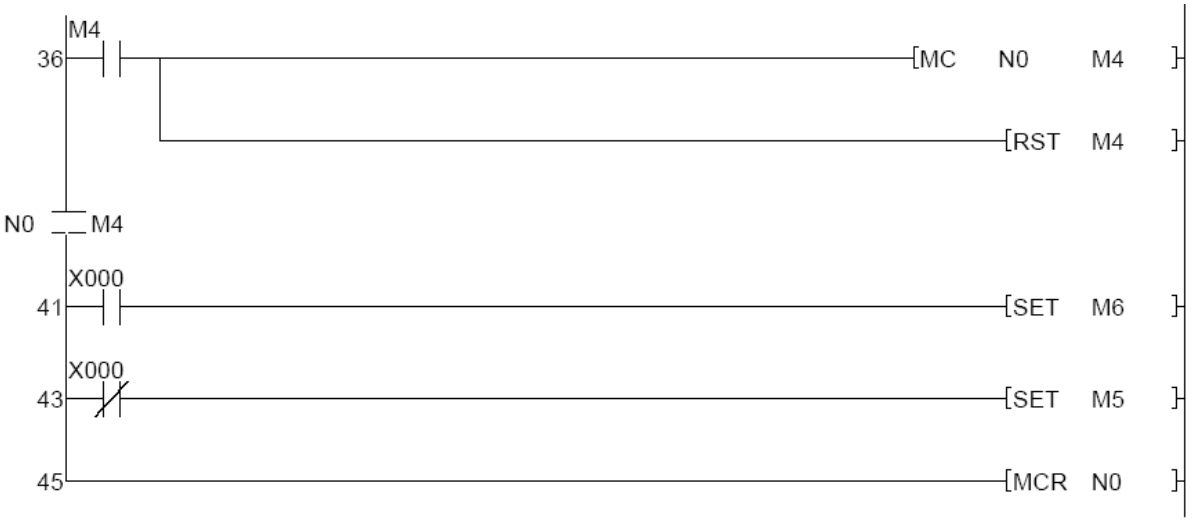


4.2. Phương pháp biểu diễn lưu đồ - block logic

Đối với khối thứ ba:

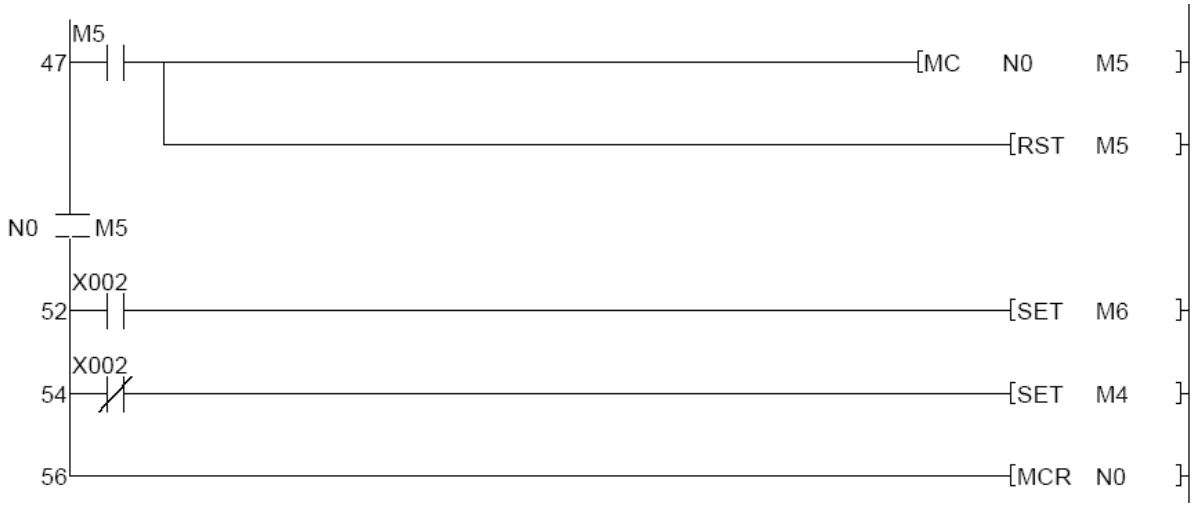
4.2. Phương pháp biểu diễn lưu đồ - block logic

Đối với khối thứ tư:



4.2. Phương pháp biểu diễn lưu đồ - block logic

Đối với khối thứ năm:



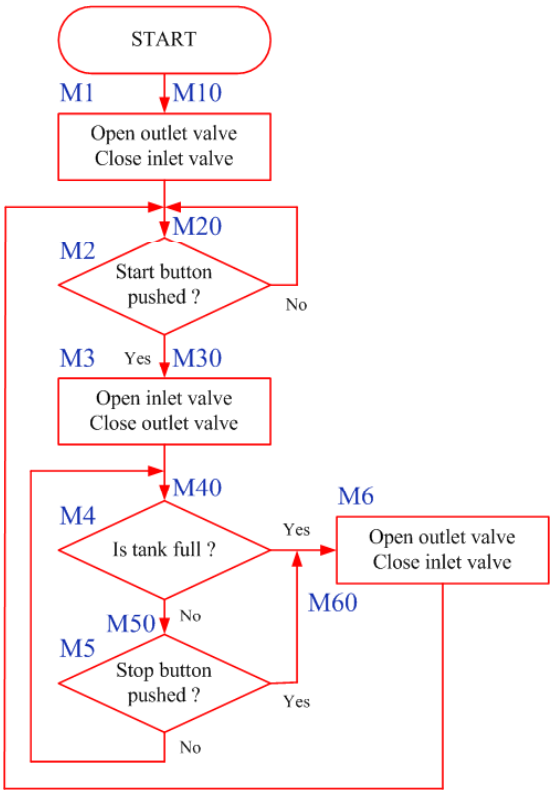
4.2. Phương pháp biểu diễn lưu đồ - block logic

Đối với khối thứ sáu:

4.2. Phương pháp biểu diễn lưu đồ - sequence bits

- Phương pháp 2: Xây dựng logic bậc thang dùng chuỗi các bit.

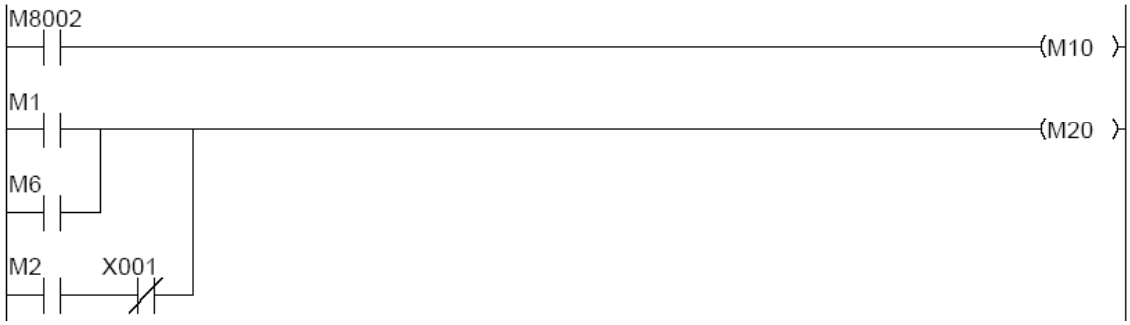
- Bước 1: Gán mỗi nhãn vào từng khối và vào từng mũi tên chuyển tiếp đến từng khối trên lưu đồ



4.2. Phương pháp biểu diễn lưu đồ - sequence bits

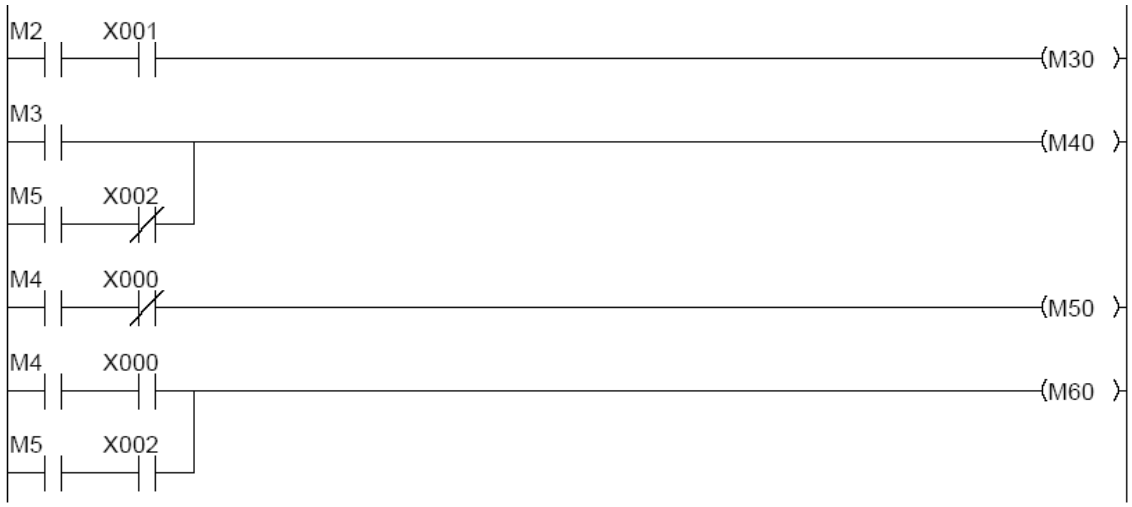
- Bước 2: Xây dựng đoạn chương trình xác định điều khiển chuyển tiếp giữa các khối. Các điều kiện này nên viết chung với nhau và được đặt vào phần đầu của chương trình PLC (trước các đoạn thực hiện logic cho các khối)

Điều kiện chuyển tiếp cho khối thứ nhất đúng khi bộ điều khiển bắt đầu hoạt động. Đối với khối thứ hai, điều kiện chuyển tiếp là từ ba khối (khối thứ nhất, thứ hai và thứ sáu).



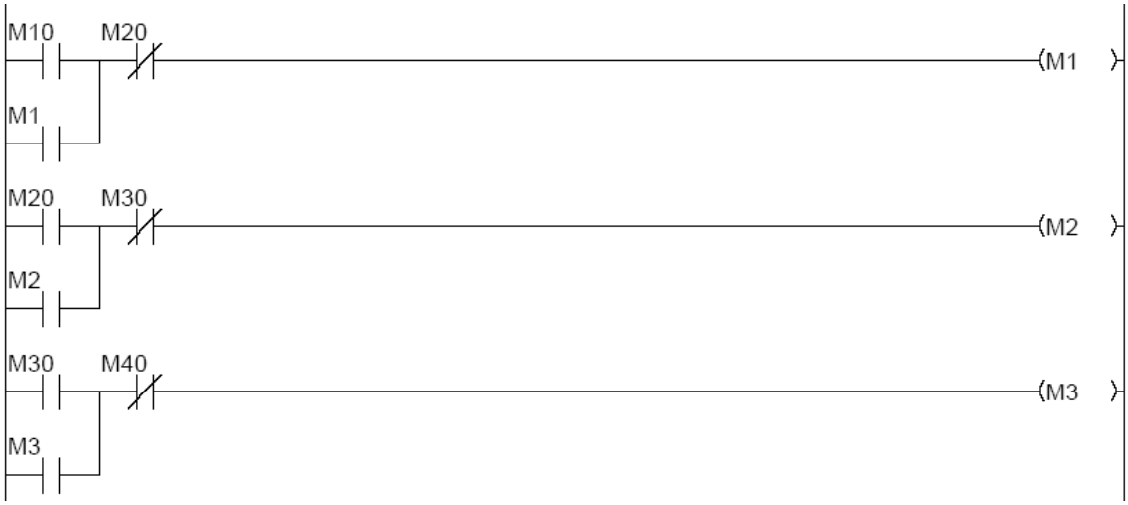
4.2. Phương pháp biểu diễn lưu đồ - sequence bits

Xét tương tự cho các điều kiện chuyển tiếp của bốn khối còn lại.

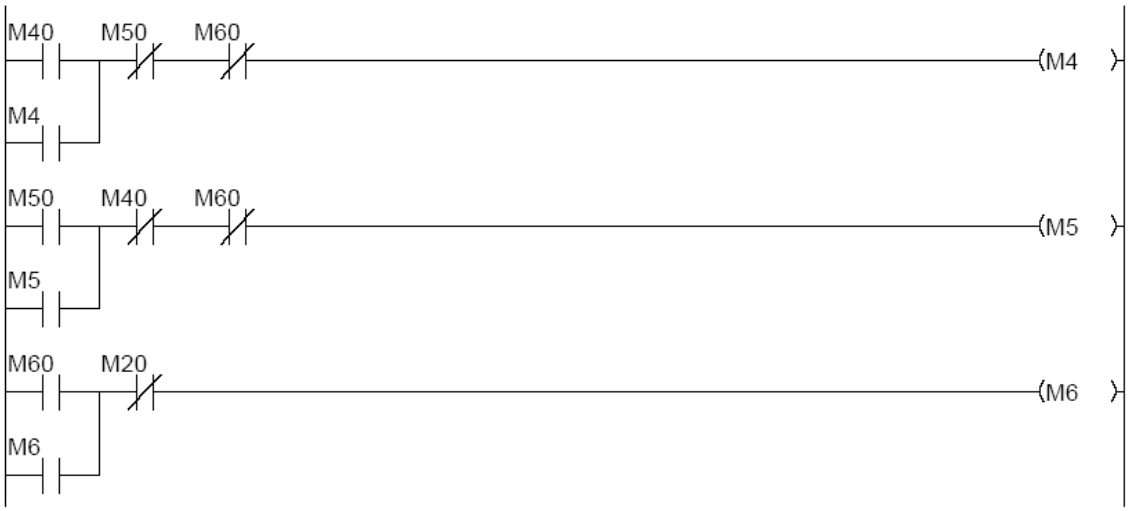


4.2. Phương pháp biểu diễn lưu đồ - sequence bits

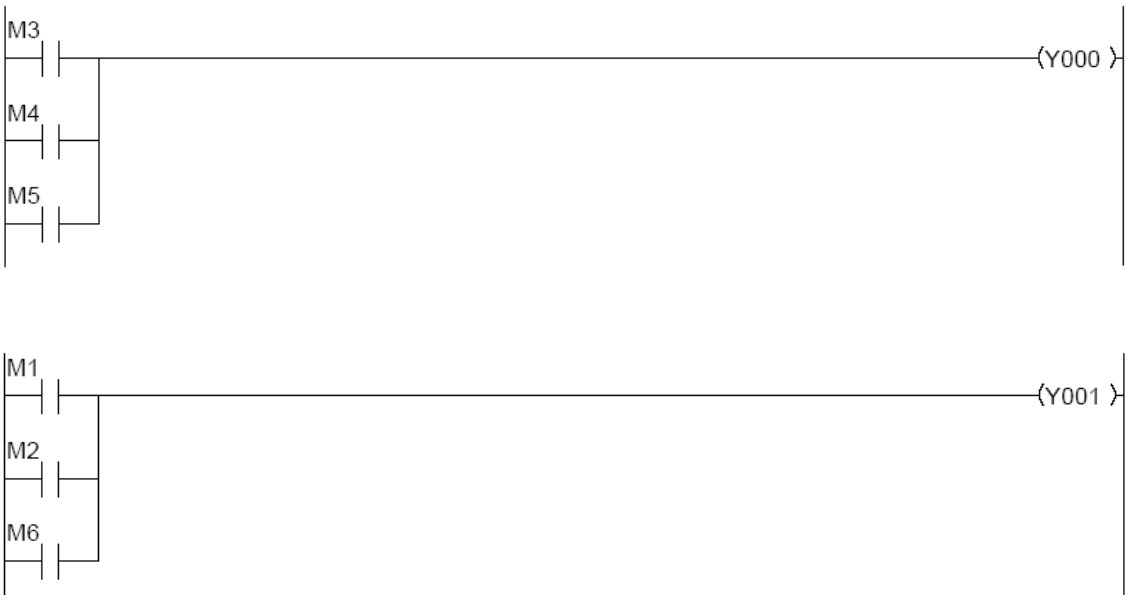
• Bước 3: Dùng các điều kiện chuyển tiếp ở bước hai để kích hoạt và kết thúc các khối. Theo đó, xây dựng nhánh logic cho các ngõ ra tương ứng.



4.2. Phương pháp biểu diễn lưu đồ - sequence bits

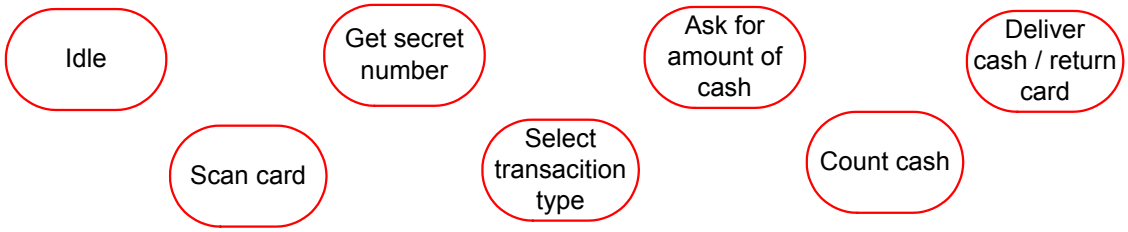


4.2. Phương pháp biểu diễn lưu đồ - sequence bits

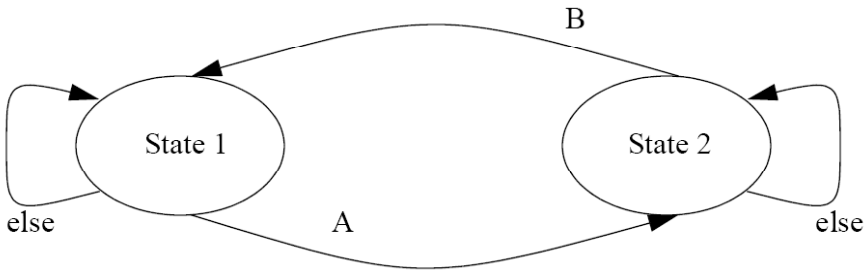


4.3. Phương pháp sơ đồ trạng thái

Một trạng thái hệ thống có thể được xem như một chế độ hoạt động của hệ thống điều khiển. Thí dụ: một máy ATM có thể được xem bao gồm các trạng thái như sau:

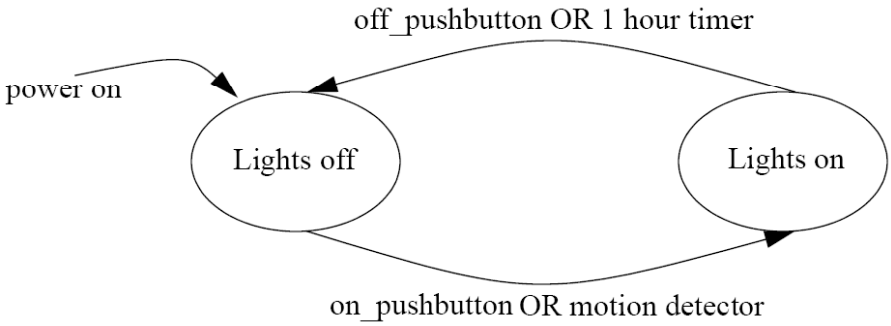


Một hệ thống điều khiển trình tự có thể được mô tả bởi các trạng thái hệ thống và các điều kiện chuyển tiếp giữa các trạng thái.



4.3. Phương pháp sơ đồ trạng thái

Thí dụ: một hệ thống điều khiển đèn tự động, được mô tả bởi sơ đồ trạng thái như sau:



Lưu ý:

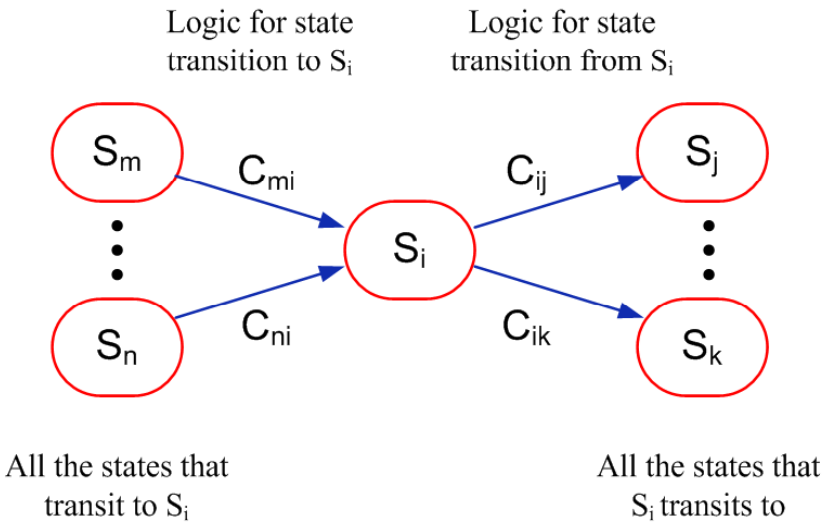
• Vòng lặp else ngụ ý rằng, trạng thái hiện tại vẫn tồn tại nếu điều kiện chuyển tiếp từ trạng thái đó đến các trạng thái không thỏa. Do đó, nó thường được bỏ qua khi biểu diễn sơ đồ trạng thái.

• Chỉ một trạng thái tồn tại ở một thời điểm, và các điều kiện chuyển tiếp là tức thời.

• Ngõ ra là hàm của các trạng thái.

4.3. Phương pháp sơ đồ trạng thái

Đối với một trạng thái S_i:

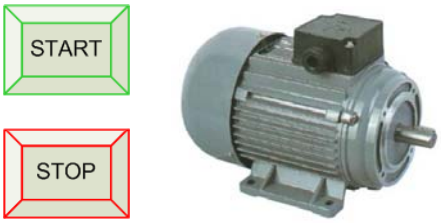


4.3. Phương pháp sơ đồ trạng thái

Các bước để thiết lập một sơ đồ trạng thái:

- Nhận diện các trạng thái
- Xác định các ngõ ra tại mỗi trạng thái
- Xác định các điều kiện chuyển tiếp giữa các trạng thái

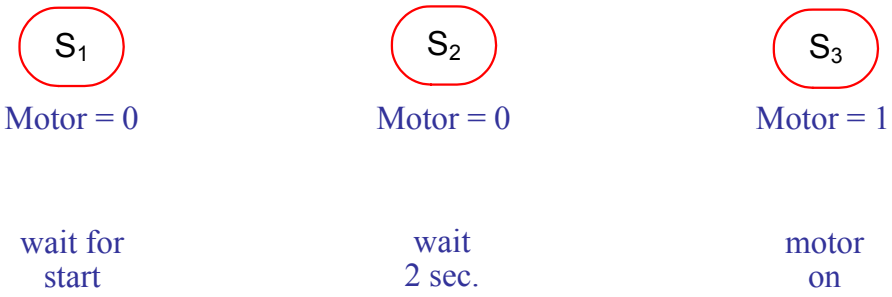
Xét thí dụ một hệ thống ON / OFF một động cơ



- Mô tả quá trình
  - Nếu nút START được nhấn (khi không nhấn nút STOP) thì động cơ sẽ làm việc sau hai giây.
  - Bất kỳ khi nào nút STOP được nhấn thì động cơ dừng lại.

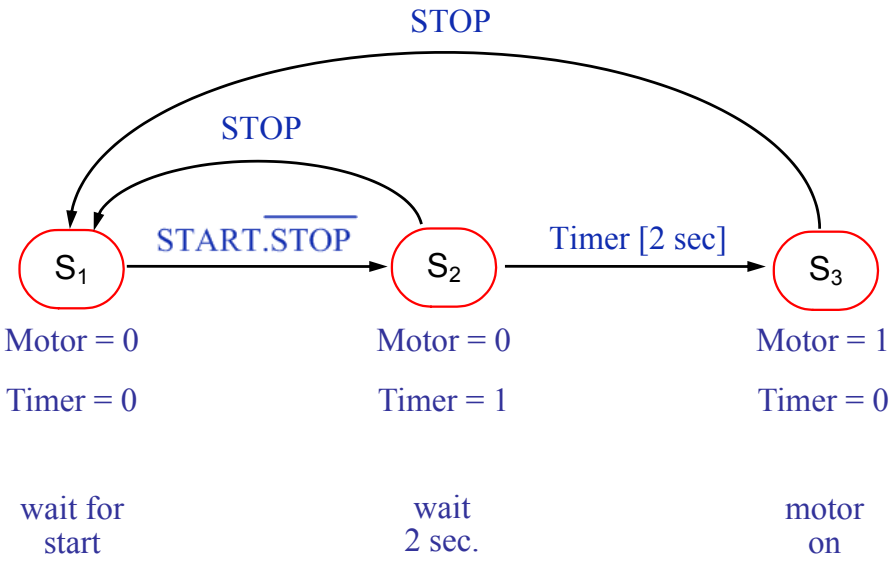
4.3. Phương pháp sơ đồ trạng thái

- Xây dựng sơ đồ trạng thái – nhận diện các trạng thái
- Xây dựng sơ đồ trạng thái – xác định các ngõ ra



4.3. Phương pháp sơ đồ trạng thái

- Xây dựng sơ đồ trạng thái – xác định các điều kiện chuyển tiếp



4.3. Phương pháp sơ đồ trạng thái

- Thiết lập biểu thức logic cho từng trạng thái và các ngõ ra tương ứng

Biểu thức logic tổng quát cho trạng thái thứ i là:

"memory" or "hold" for state S_i — all states that transit to S_i — all states that S_i transits to

$$S_i = (S_i + S_m.C_{mi} + \ldots + S_n.C_{ni}).\overline{S_j}.\ \ldots\ .\overline{S_k}$$

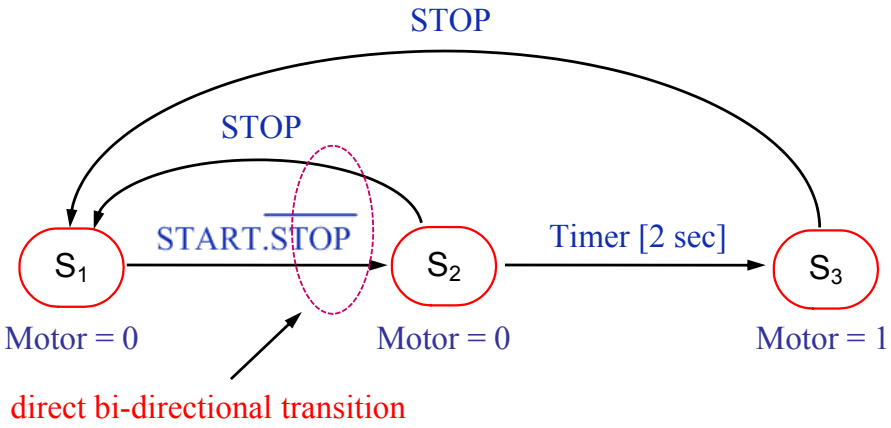
Biểu thức logic tổng quát cho ngõ ra O_i là:

$$O_i = S_p + S_q + \ldots$$

all states where O_i is true (on)

4.3. Phương pháp sơ đồ trạng thái

Lưu ý: Nếu giữa hai trạng thái vừa có chuyển tiếp tới và chuyển tiếp lui (direct bi-directional transition), khi đó cần phải thêm trạng thái giả (dummy state) vào sơ đồ trạng thái.

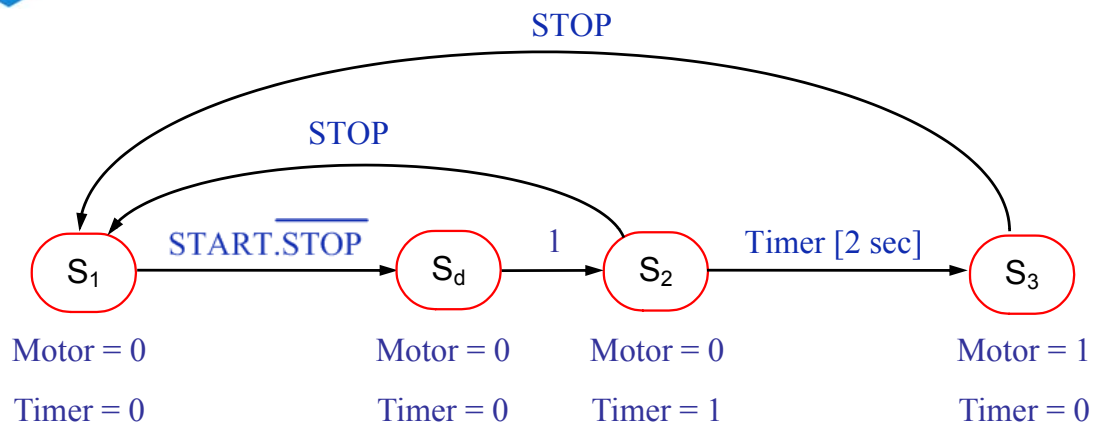


Ta có:

$$S_1 = (S_1 + S_2.STOP + S_3.STOP).\overline{S_2}$$ → Không thể trở về S_1 từ S_2

$$S_2 = (S_2 + S_1.START.\overline{STOP}).\overline{S_1}.\overline{S_3}$$

4.3. Phương pháp sơ đồ trạng thái



$S_1 = (S_1 + S_2.STOP + S_3.STOP).\overline{S_d}$

$S_d = (S_d + S_1.START.\overline{STOP}).\overline{S_2}$

$S_2 = (S_2 + S_d).\overline{S_1}.\overline{S_3}$

$S_3 = (S_3 + S_2.Timer\ [2\ sec]).\overline{S_1}$

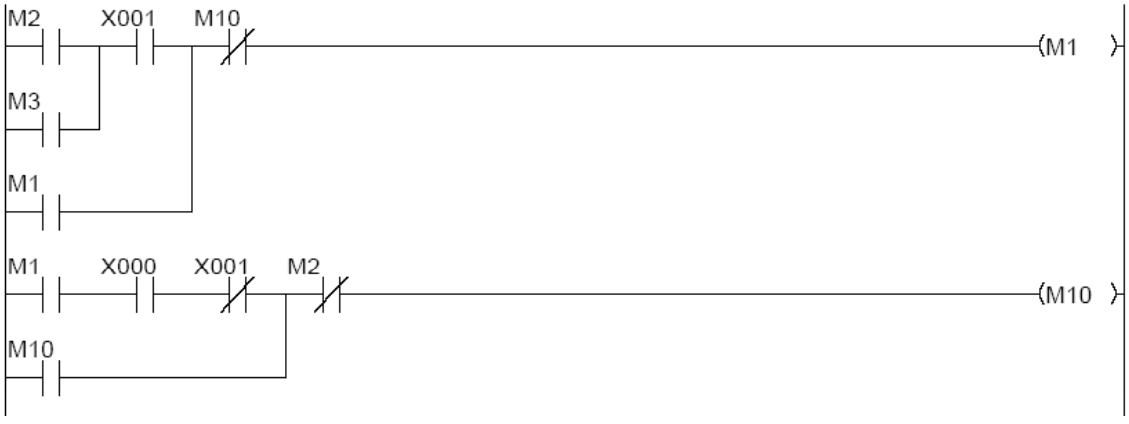
$Motor = S_3$

$Timer = S_2$

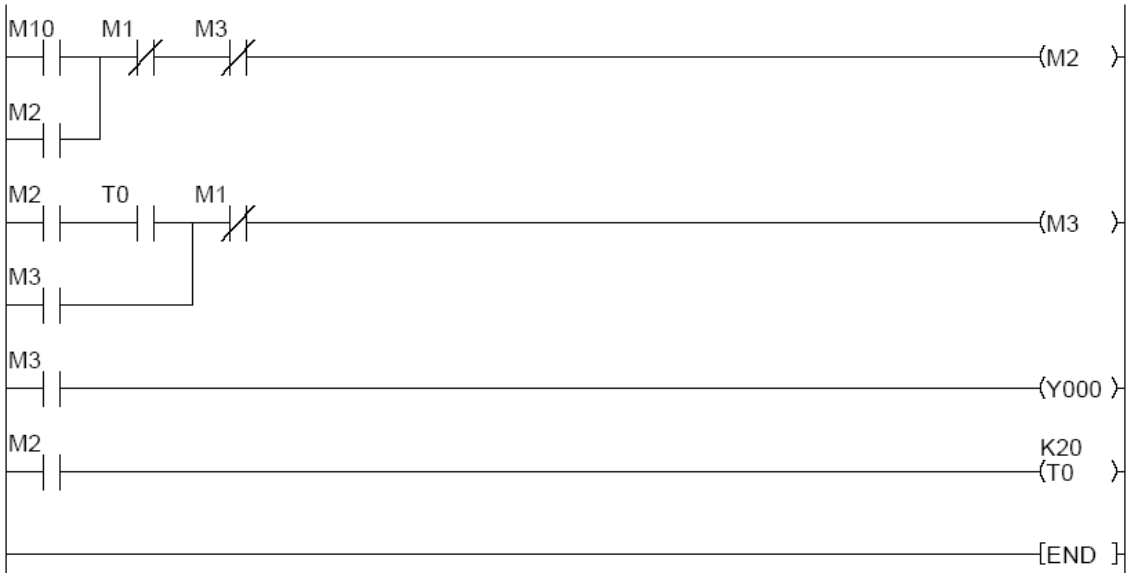
4.3. Phương pháp sơ đồ trạng thái

Với bảng I/O như hình bên, chương trình PLC được viết như sau:

Device	PC device	Description
START	X000	Start button
STOP	X001	Stop button
	Y000	Motor



4.3. Phương pháp sơ đồ trạng thái



4.4. Phương pháp biểu đồ chức năng

Các phương pháp vừa được trình bày ở trên thông thường chỉ phù hợp đối với những quá trình mà chỉ có một trạng thái / bước tồn tại tại một thời điểm. Do đó, chúng trở nên khó khăn khi biểu diễn một quá trình hoạt động phức tạp – bao gồm nhiều trình tự thực hiện đồng thời. Khi đó, phương pháp biểu đồ chức năng được sử dụng để giải quyết những trường hợp như thế.

Thí dụ như trong hệ thống điều khiển trạm trộn bê tông, có nhiều hoạt động được thực hiện đồng thời cung cấp nguyên vật liệu vào bồn trộn như:

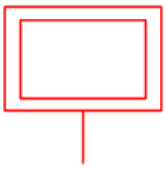
Định lượng và vận chuyển hỗn hợp cát đá	Định lượng và vận chuyển xi măng	Định lượng và vận chuyển nước, hóa chất ...

4.4. Phương pháp biểu đồ chức năng

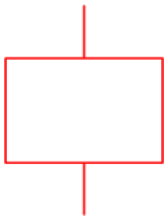
Các thành phần cơ bản hình thành nên biểu đồ chức năng:

- Dòng tín hiệu và điều kiện chuyển tiếp

- Bước đầu tiên (khởi tạo)

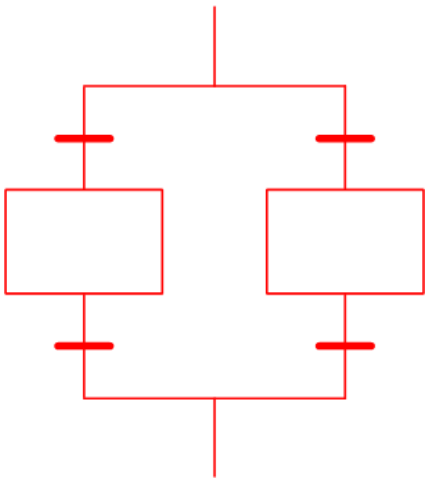


- Bước (trạng thái)

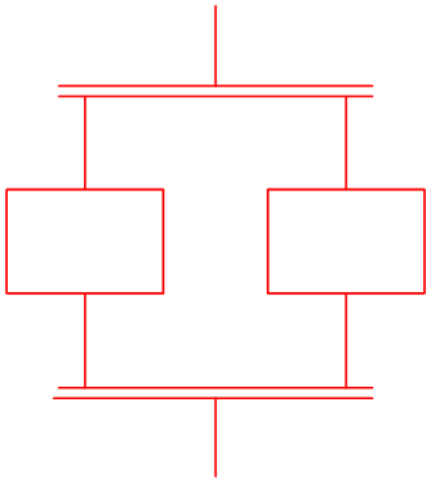


4.4. Phương pháp biểu đồ chức năng

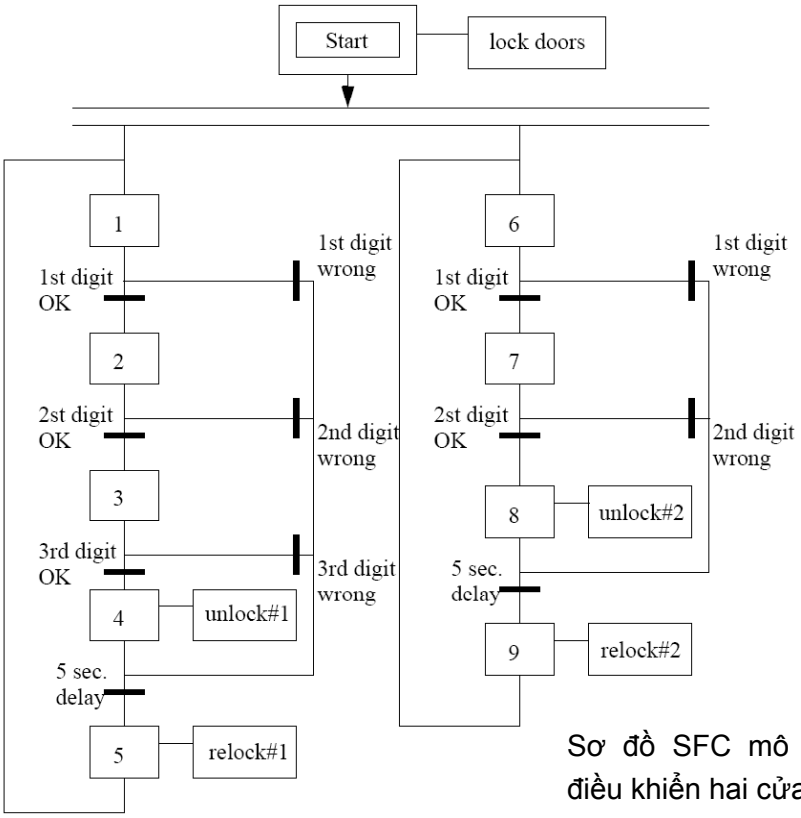
- Phân nhánh có lựa chọn



- Phân nhánh đồng thời



4.4. Phương pháp biểu đồ chức năng



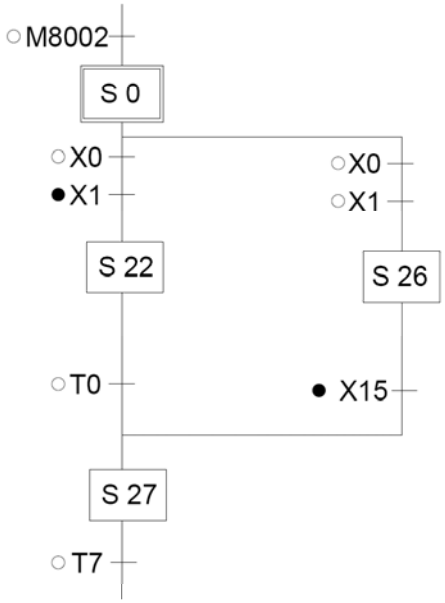
Sơ đồ SFC mô tả một hệ thống điều khiển hai cửa tự động.

4.4. Phương pháp biểu đồ chức năng

Kỹ thuật lập trình dùng STepLadder (STL) đối với PLC họ FX - Mitsubishi

Kỹ thuật lập trình STL này tương tự với sự biểu diễn của biểu đồ chức năng, và có những đặc điểm như sau:

- Khả năng giữ được trạng thái hiện hành nhờ dùng cờ có khả năng chốt.
- Tự động vô hiệu trạng thái đước đó khi chuyển vào trạng thái hiện hành.
- Dễ dàng phân nhánh song song (dạng OR / AND).
- Được hổ trợ thông qua cờ trạng thái S

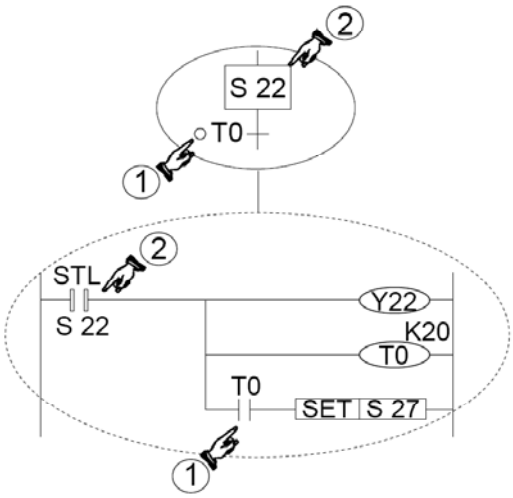


4.4. Phương pháp biểu đồ chức năng

Kỹ thuật lập trình STL được sử dụng khá phổ biến vì nó còn có thể được lập trình ở dạng bậc thang hoặc ở dạng lệnh thông thường.

Lệnh STL thể hiện dưới dạng công tắc STL cho phép đóng hay mở các trạng thái theo trình tự.

Mạch STL được phát triển từ cơ chế điều khiển MC và MCR. Nó cho phép trạng thái hoạt động trở thành trạng thái hiện hành, thực hiện tác vụ trong trạng thái đó và sang trạng thái khác khi thỏa điều kiện chuyển tiếp.



Lập trình STL

• Đoạn chương trình STL có thể được viết lồng bên trong một chương trình bậc thang thông thường.

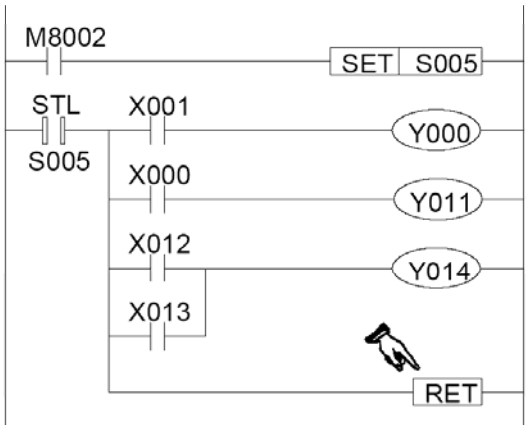
```
LD      X000
OUT     Y004
LD      X002
SET     S009
STL     S009
OUT     Y010
LDI     X003
OUT     Y006
RET
LD      X005
OUT     Y007
RST     M080
```

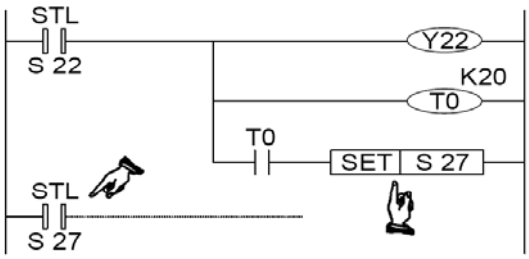
Normal Ladder Program

Embedded STL Program

Lập trình STL

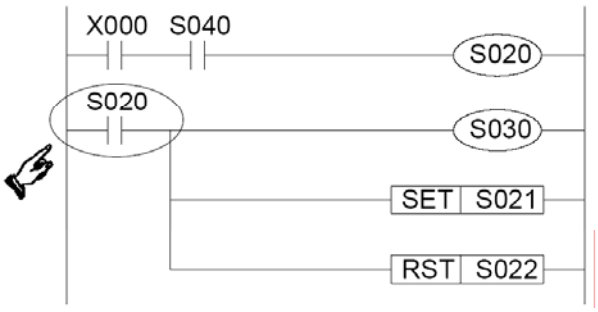
• Khởi tạo trạng thái / kích hoạt những trạng thái mới / kết thúc đoạn chương trình STL.

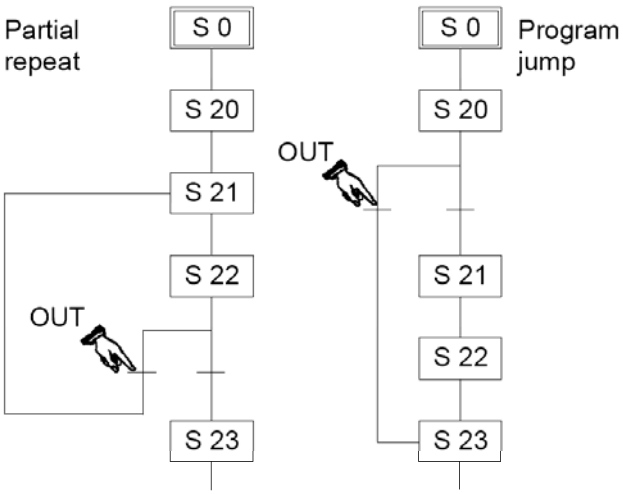




Lập trình STL

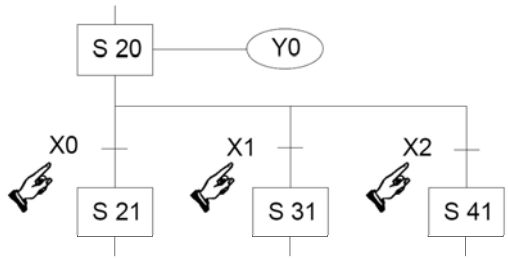
Lưu ý khi dùng lệnh SET / OUT để di chuyển giữa các bước:



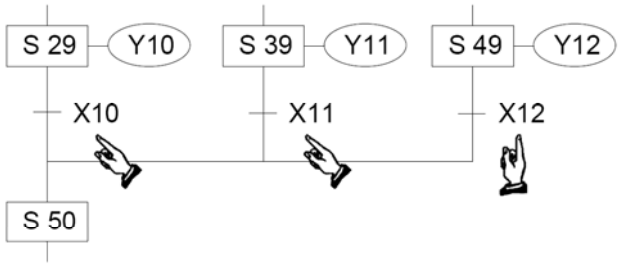


Lập trình STL

• Phân nhánh có lựa chọn



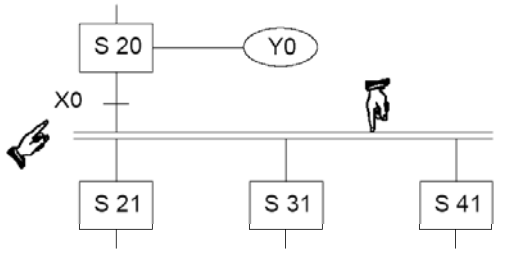

```
STL  S  20
OUT  Y   0
LD   X   0
SET  S  21
LD   X   1
SET  S  31
LD   X   2
SET  S  41
```



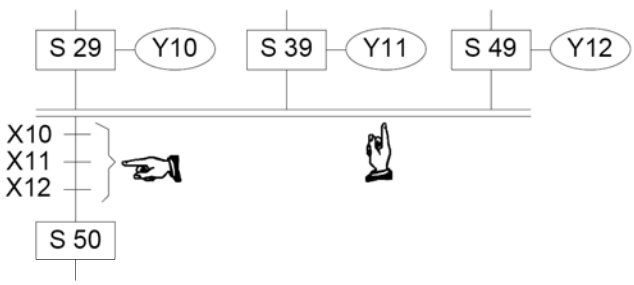

```
STL  S  29
OUT  Y  10
LD   X  10
SET  S  50
STL  S  39
OUT  Y  11
LD   X  11
SET  S  50
STL  S  49
OUT  Y  12
LD   X  12
SET  S  50
```


Lập trình STL

• Phân nhánh đồng thời

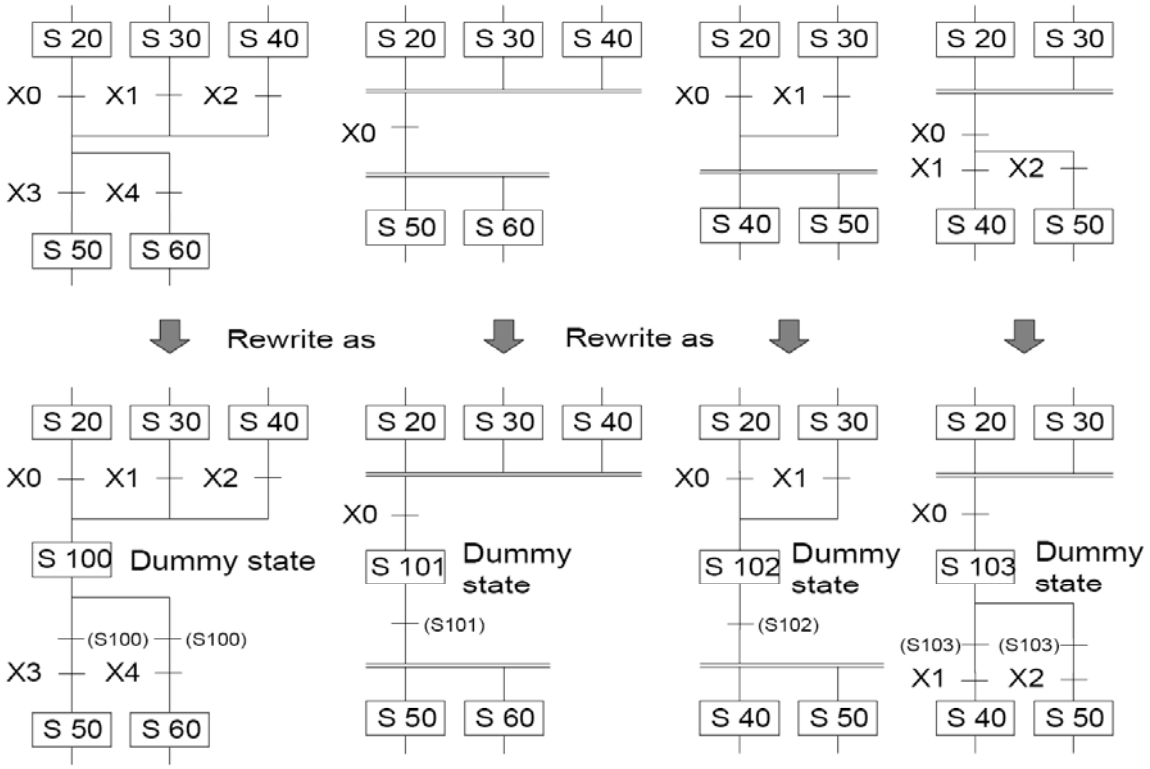



```
STL  S  20
OUT  Y   0
LD   X   0
SET  S  21
SET  S  31
SET  S  41
```



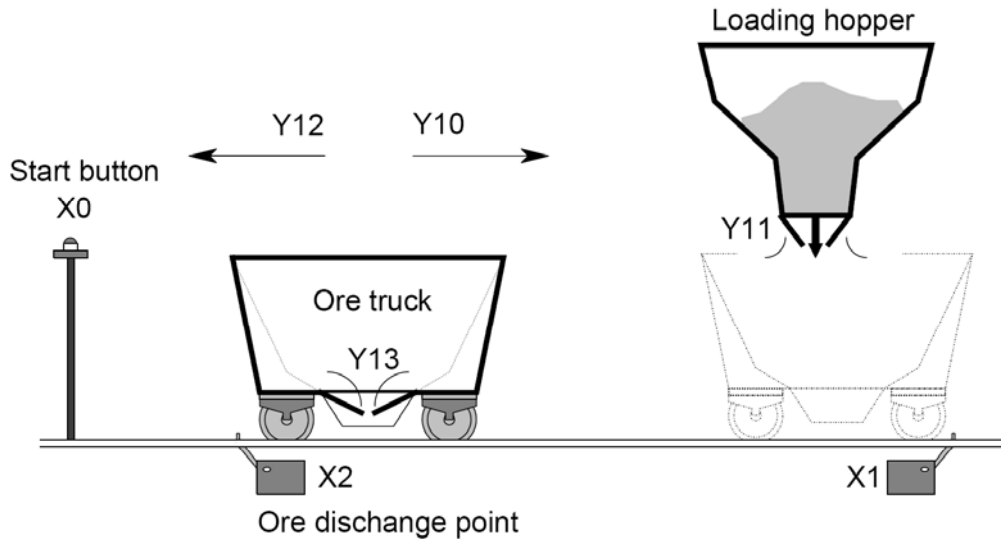

```
STL  S  29
OUT  Y  10

STL  S  39
OUT  Y  11

STL  S  49
OUT  Y  12

STL  S  29
STL  S  39
STL  S  49
LD   X  10
AND  X  11
AND  X  12
SET  S  50
```

Lập trình STL

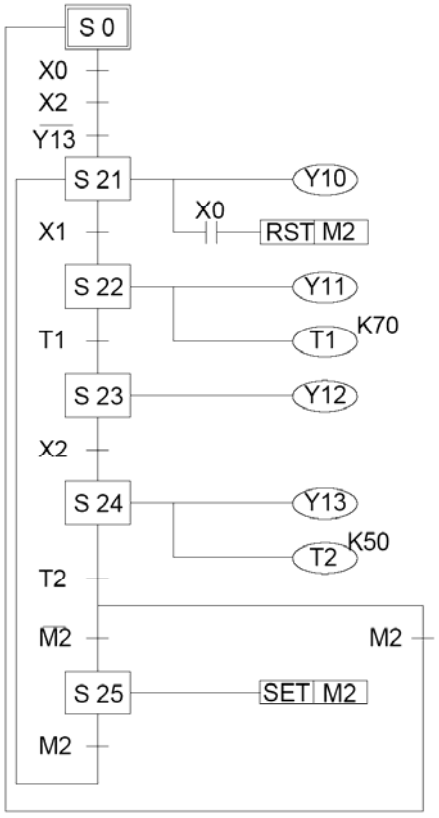


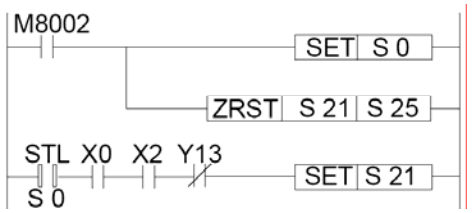
Lập trình STL

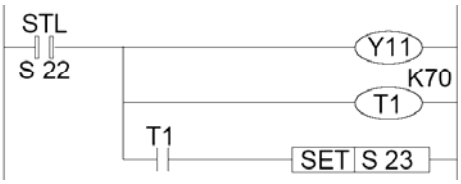
Thí dụ 1: a semi-automatic loading-unloading ore truck



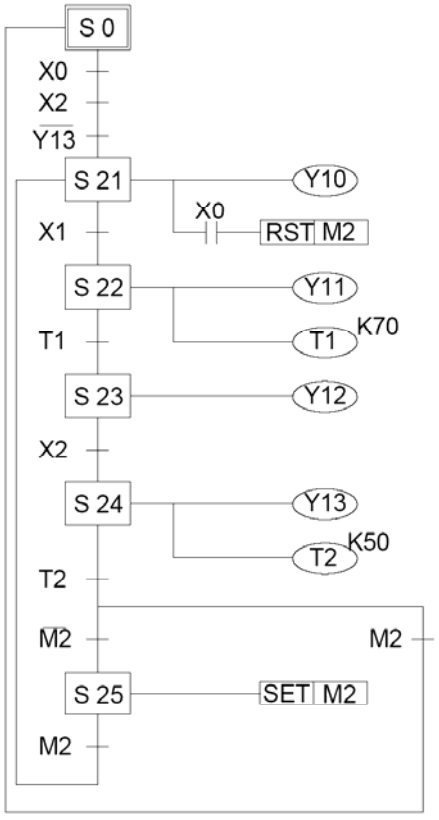
Lập trình STL

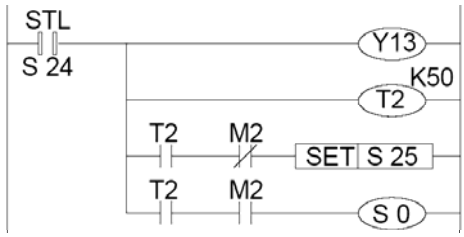


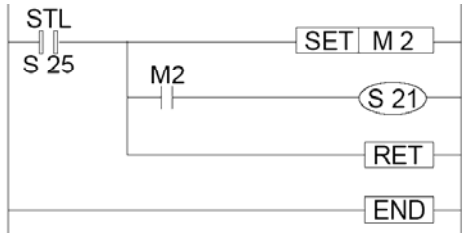




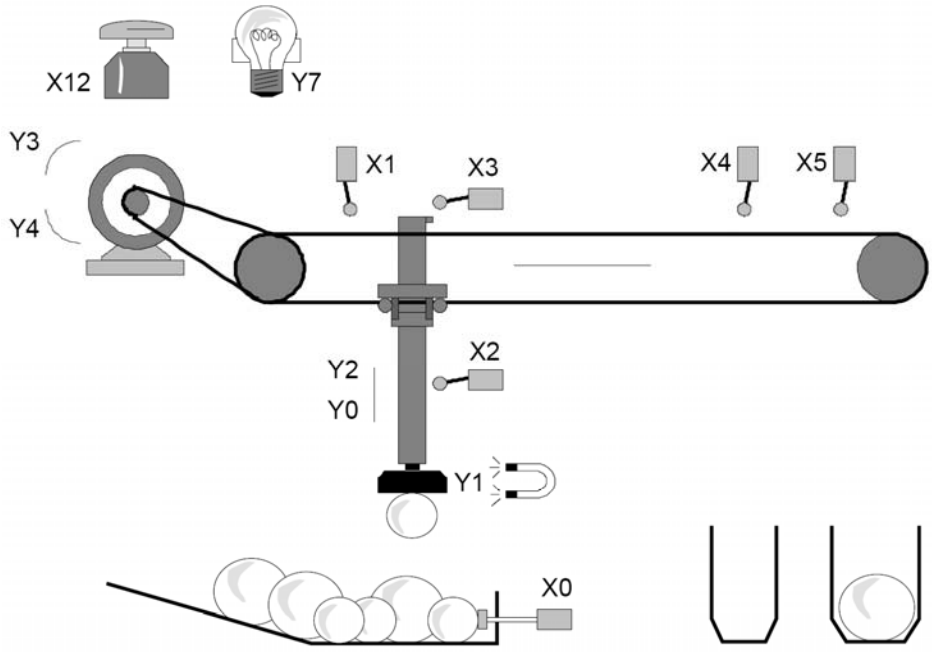
Lập trình STL





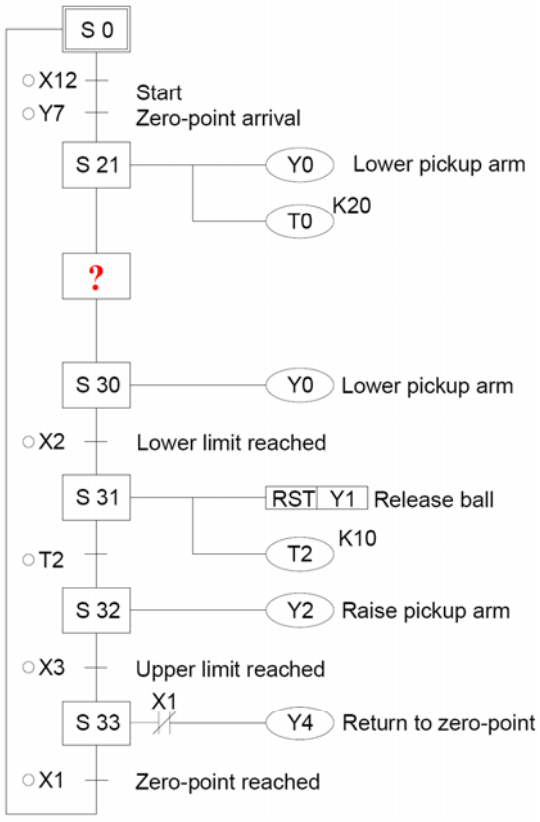


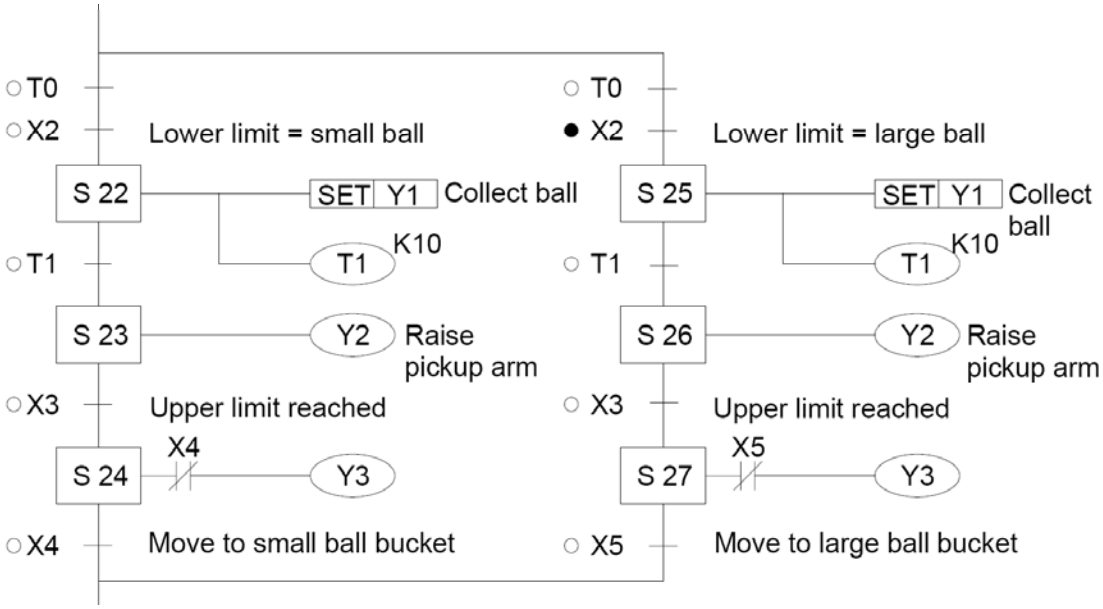
Thí dụ 2: an automatic sorting robot



This example uses the dot notation to identify normally open and normally closed contacts.

○ Normally open contacts
● Normally closed contacts




T0
X2
Lower limit = small ball
S 22
SET Y1
Collect ball
T1
K10
T1
S 23
Y2
Raise pickup arm
X3
Upper limit reached
X4
S 24
Y3
X4
Move to small ball bucket
T0
X2
Lower limit = large ball
S 25
SET Y1
Collect ball
T1
K10
T1
S 26
Y2
Raise pickup arm
X3
Upper limit reached
X5
S 27
Y3
X5
Move to large ball bucket