Tổng Quan Về Encoder.

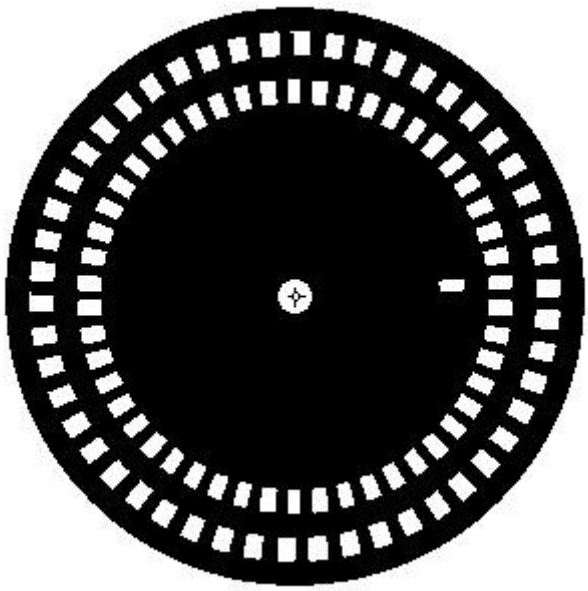
Encoder mục đích dùng để quản lý vị trí góc của một đĩa quay, đĩa quay có thể là bánh xe, trục động cơ, hoặc bất kỳ thiết bị quay nào cần xác định vị trí góc.

Encoder được chia làm 2 loại, absolute **encoder** và incremental **encoder**. Tạm dịch là **encoder** tuyệt đối và **encoder** tương đối. Chữ **encoder** tuyệt đối dịch theo nguyên văn, nhưng vì tiếng Việt mình cái gì có 2 loại, thì loại còn lại được dịch ngược lại với loại kia. Cho nen dịch là **encoder** tương đối cho incremental **encoder**.

Nếu dịch sát nghĩa, khi ta đọc absolute **encoder**, có nghĩa là **encoder** tuyệt đối, tức là tín hiệu ta nhận được, chỉ rõ ràng vị trí của **encoder**, chúng ta không cần xử lý gì thêm, cũng biết chính xác vị trí của **encoder**.

Còn incremental **encoder**, là loại **encoder** chỉ có 1, 2, hoặc tối đa là 3 vòng lỗ. Các bạn hình dung thế này, nếu bây giờ các bạn đục một lỗ trên một cái đĩa quay, thì cứ mỗi lần đĩa quay 1 vòng, các bạn sẽ nhận được tín hiệu, và các bạn đã biết đĩa quay một vòng. Nếu bây giờ các bạn có nhiều lỗ hơn, các bạn sẽ có được thông tin chi tiết hơn, có nghĩa là đĩa quay 1/4 vong, 1/8 vòng, hoặc 1/n vòng, tùy theo số lỗ nằm trên incremental **encoder**.

Cứ mỗi lần đi qua một lỗ, chúng ta phải lập trình để thiết bị đo đếm lên 1. Do vậy, **encoder** loại này có tên incremental **encoder** (**encoder** tăng lên 1 đơn vị).



Nguyên lý hoạt động cơ bản của encoder, LED và lỗ

Nguyên lý cơ bản của **encoder**, đó là một đĩa tròn xoay, quay quanh trục. Trên đĩa

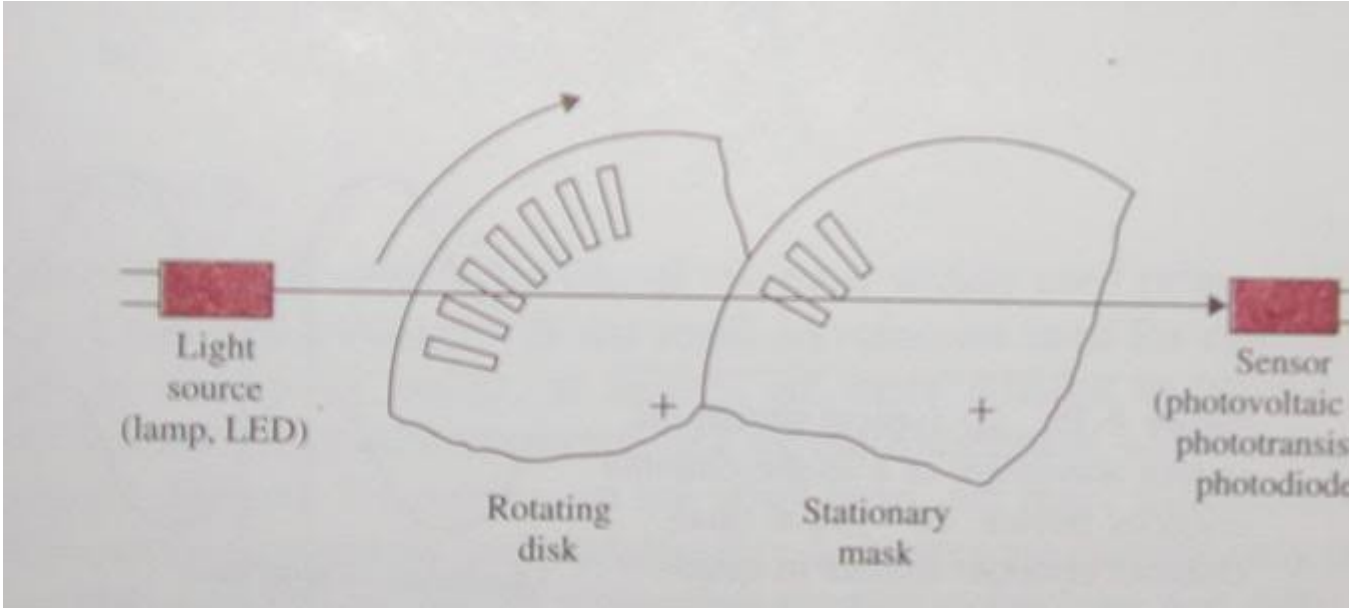
có các lỗ (rãnh). Người ta dùng một đèn led để chiếu lên mặt đĩa. Khi đĩa quay, chỗ không có lỗ (rãnh), đèn led không chiếu xuyên qua được, chỗ có lỗ (rãnh), đèn led sẽ chiếu xuyên qua. Khi đó, phía mặt bên kia của đĩa, người ta đặt một con mắt thu. Với các tín hiệu có, hoặc không có ánh sáng chiếu qua, người ta ghi nhận được đèn led có chiếu qua lỗ hay không.

Khi trục quay, giả sử trên đĩa chỉ có một lỗ duy nhất, cứ mỗi lần con mắt thu nhận được tín hiệu đèn led, thì có nghĩa là đĩa đã quay được một vòng.

Đây là nguyên lý rất cơ bản của **encoder**.

Tuy nhiên, những vấn đề được đặt ra là, làm sao để xác định chính xác hơn vị trí của đĩa quay (mịn hơn) và làm thế nào để xác định được đĩa đang quay theo chiều nào? Đó chính là vấn đề để chúng ta tìm hiểu về **encoder**.

Hình sau sẽ minh họa nguyên lý cơ bản của hoạt động **encoder**.



Các bạn thấy trong hình, có một đĩa mask, không quay, đó là đĩa cố định, thực ra là để che khe hẹp ánh sáng đi qua, giúp cho việc đọc **encoder** được chính xác hơn mà thôi. Chúng tôi không để cập đến đĩa mặt nạ này ở đây.

Hoạt động của hai loại encoder này như thế nào?

1) Absolute encoder

Vấn đề chúng ta sẽ quan tâm ở đây, chính là vấn đề về độ mịn của **encoder**, có nghĩa là làm thế nào biết đĩa đã quay 1/2 vòng, 1/4 vòng, 1/8 vòng hay 1/n vòng, chứ không phải chỉ biết đĩa đã quay được một vòng.

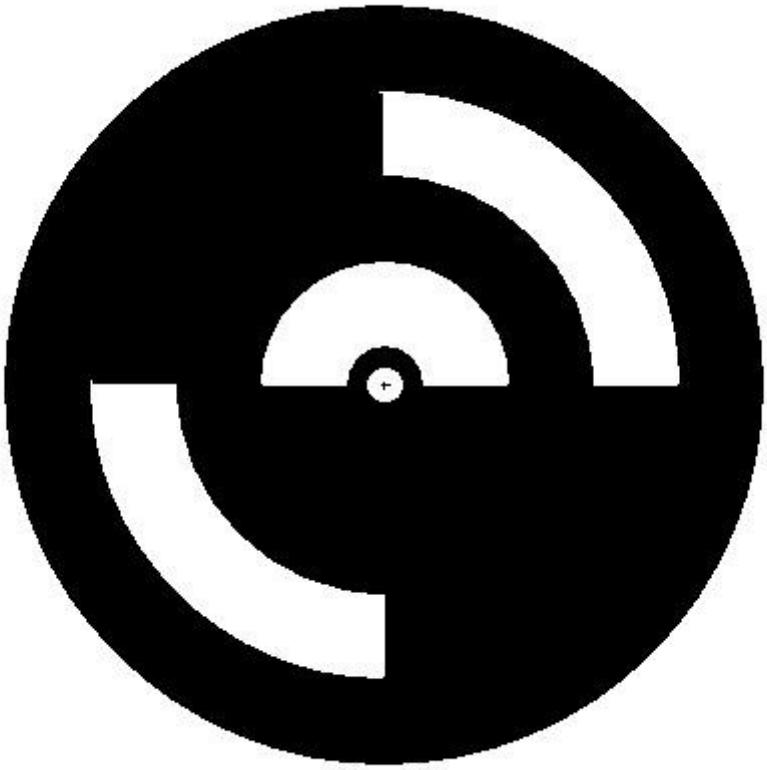
Quay lại bài toán cơ bản về bit và số bit, chúng ta xem xét vấn đề theo một cách hoàn toàn toán học nhé:

Với một số nhị phân có 2 chữ số, chúng ta sẽ có 00, 01, 10, 11, tức là 4 trạng thái. Điều đó có nghĩa là với 2 chữ số, chúng ta có thể chia đĩa **encoder** thành 4 phần bằng nhau. Và khi quay, chúng ta sẽ xác định được độ chính xác đến 1/4 vòng.

Tương tự như vậy, nếu với một số có n chữ số, chúng ta sẽ xác định được độ chính xác đến 1/(2^n) vòng.

Thế làm sao để xác định 2^n trạng thái này của đĩa **encoder**?

Các bạn xem hình sau:



Ở đây, tôi đưa ra ví dụ với đĩa **encoder** có 2 vòng đĩa. Các bạn sẽ thấy rằng, ở vòng trong cùng, có một rãnh rộng bằng 1/2 đĩa. Vòng phía ngoài, sẽ có 2 rãnh nằm đối diện nhau.

Như vậy, chúng ta cần 2 đèn led để phát xuyên qua 2 vòng lỗ, và 2 đèn thu.

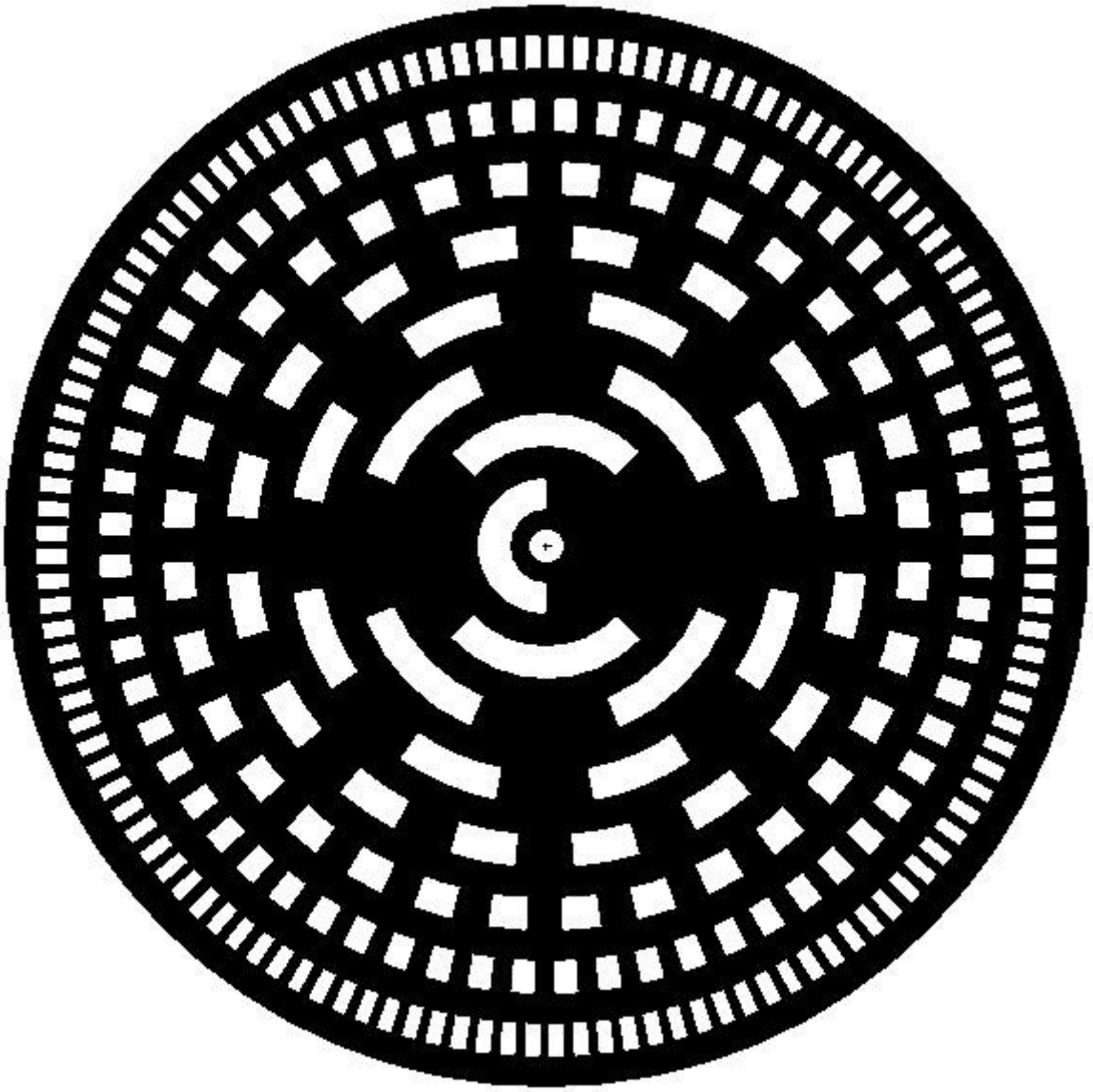
Giả sử ở vòng lỗ thứ nhất (trong cùng), đèn đọc đang nằm ở vị trí có lỗ hở, thì tín hiệu nhận được từ con mắt thu sẽ là 1. Và ở vòng lỗ thứ hai, thì chúng ta đang ở vị

trí không có lỗ, như vậy con mắt thu vòng 2 sẽ đọc được giá trị 0.

Và như vậy, với số 10, chúng ta xác định được **encoder** đang nằm ở góc phần tư nào, cũng có nghĩa là chúng ta quản lý được độ chính xác của đĩa quay đến 1/4 vòng. Trong ví dụ trên, nếu đèn LED đọc được 10, thì vị trí của LED phải nằm trong góc phần tư thứ hai, phía trên, bên trái.

Kết quả, nếu đĩa **encoder** có đến 10 vòng lỗ, thì chúng ta sẽ quản lý được đến 1/(2^10) tức là đến 1/1024 vòng. Hay người ta nói là độ phân giải của **encoder** là 1024 xung trên vòng (pulse per revolution - ppr).

Sau đây là ví dụ abosulte **encoder** 8 vòng lỗ:



Vậy cách thiết kế absolute **encoder** như thế nào?

Các bạn luôn chú ý rằng, để thiết kế **encoder** tuyệt đối, người ta luôn vẽ sao cho bit thứ N (đối với **encoder** có N vòng lỗ) nằm ở trong cùng, có nghĩa là lỗ lớn nhất có góc rộng 180 độ, nằm trong cùng. Bởi vì chúng ta thấy rằng, bit0 (nếu xem là số nhị phân) sẽ thay đổi liên tục mỗi 1/2^N vòng quay, vì thế, chúng ta cần rất nhiều lỗ. Nếu đặt ở trong thì không thể nào vẽ được, vì ở trong bán kính nhỏ hơn. Ngoài ra, nếu đặt ở trong, thì về kết cấu cơ khí, nó quá gần trục, và quá nhiều lỗ, sẽ rất yếu. Vì hai điểm này, nên bit0 luôn đặt ở ngoài cùng, và bitN-1 luôn đặt trong cùng như hình trên.

Rất nhiều người thắc mắc về cách thực tế để vẽ **encoder** như thế nào. Tuy nhiên, kể từ khi có chương trình thiết kế **encoder** này, tôi cho rằng chúng ta không nên quan tâm đến vấn đề đó nữa. Chỉ cần hiểu nó hoạt động ra sao, rồi sau đó chúng ta dùng chương trình này để vẽ. **2) Incremental encoder**

Nhận thấy một điều rằng, **encoder** tuyệt đối rất có lợi cho những trường hợp khi góc quay là nhỏ, và động cơ không quay nhiều vòng. Khi đó, việc xử lý **encoder**tuyệt đối trở nên dễ dàng cho người dùng hơn, vì chỉ cần đọc giá trị là chúng ta biết ngay được vị trí góc của trục quay.

Tuy nhiên, nếu động cơ quay nhiều vòng, điều này không có lợi, bởi vì khi đó, chúng ta phải xử lý để đếm số vòng quay của trục.

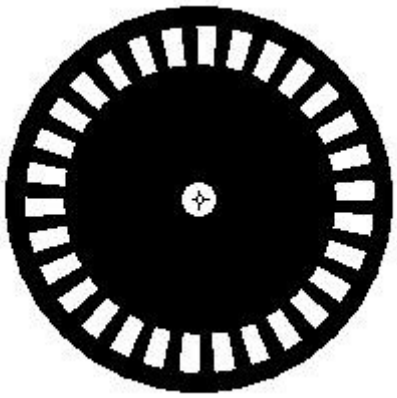
Ngoài ra, như các bạn thấy đó, nếu thiết kế **encoder** tuyệt đối, chúng ta cần quá nhiều vòng lỗ, và dẫn tới giới hạn về kích thước của **encoder**, bởi vì việc gia công chính xác các lỗ quá nhỏ là không thể thực hiện được. Chưa kể rằng việc thiết kế một dãy đèn led và con mắt thu cũng ảnh hưởng rất lớn đến kích thước giới hạn này.

Theo kinh nghiệm của cá nhân tôi, tôi thấy **encoder** 8 bit là đã rất chi tiết rồi, và ở trường DHBKHCM có loại **encoder** 12bit đã là loại tốt nhất mà tôi biết. Tôi chưa thấy loại **encoder** tuyệt đối nào 16 bit cả, và cũng không có ý định tìm nó trên internet.

Độ chính xác của **encoder** 12 bit đã là 1/4096 rồi.

Tuy nhiên, điều này được khắc phục bằng incremental **encoder** một cách khá đơn giản. Chính vì vậy, ngày nay, đa số người ta sử dụng incremental **encoder** trong những ứng dụng hiện đại. **Hoạt động của incremental encoder**
Thật đơn giản, incremental **encoder**, sẽ tăng 1 đơn vị khi một lần lên xuống của cạnh xung.

Các bạn xem hình **encoder** sau:



Các bạn thấy rằng, cứ mỗi lần quay qua một lỗ, thì **encoder** sẽ tăng một đơn vị trong biến đếm.

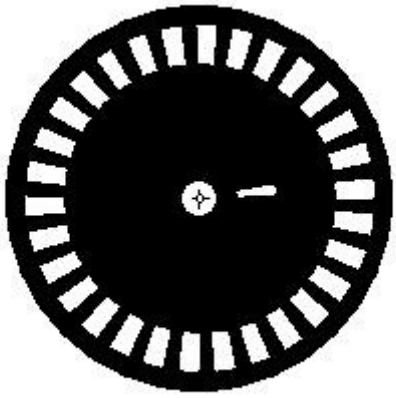
Tuy nhiên, một vấn đề là làm sao để biết được **encoder** quay hết một vòng? Nếu cứ đếm vô hạn như thế này, thì chúng ta không thể biết được khi nào nó quay hết một vòng. Nếu bây giờ các bạn đếm số lỗ **encoder** để biết nó đã quay một vòng, thì nếu với **encoder** 1000 lỗ chắc các bạn sẽ đếm đến sáng luôn.

Chưa kể, mỗi lần có những rung động nào đó mà ta không quản lý được, **encoder** sẽ bị sai một xung. Khi đó, nếu hoạt động lâu dài, sai số này sẽ tích lũy, ngày hôm nay sai một xung, ngày hôm sau sai một xung. Đến cuối cùng, có thể động cơ quay 2 vòng rồi các bạn mới đếm được 1 vòng.

Để tránh điều **tai** hại này xảy ra, người ta đưa vào thêm một lỗ định vị để đếm số vòng đã quay của **encoder**.

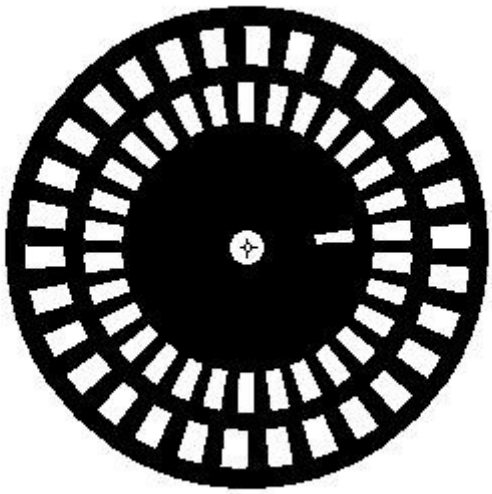
Như vậy, cho dù có lệch xung, mà chúng ta thấy rằng **encoder** đi ngang qua lỗ định vị này, thì chúng ta sẽ biết là **encoder** đã bị đếm sai ở đâu đó. Nếu vì một rung động nào đó, mà chúng ta không thấy **encoder** đi qua lỗ định vị, vậy thì từ số xung, và việc đi qua lỗ định vị, chúng ta sẽ biết rõ hiện tượng sai của **encoder**.

Đây là hình **encoder** có lỗ định vị:



Tuy nhiên, một vấn đề lớn nữa là, làm sao chúng ta biết **encoder** đang xoay theo chiều nào? Bởi vì cho dù xoay theo chiều nào, thì tín hiệu **encoder** cũng chỉ là các xung đơn lẻ và xoay theo hai chiều đều giống nhau.

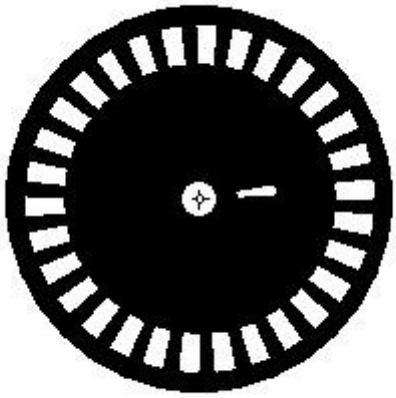
Chính vì vậy, người ta đặt thêm một vòng lỗ ở giữa vòng lỗ thứ 1 và lỗ định vị như hình sau:



Chú ý rằng, vị trí góc của các lỗ vòng 1 và các lỗ vòng 2 lệch nhau. Các cạnh của lỗ vòng 2 nằm ngay giữa các lỗ vòng 1 và ngược lại.

Chúng ta sẽ khảo sát tiếp vấn đề **encoder** trong phần tín hiệu xung để hiểu rõ hơn về **encoder**. Tuy nhiên, các bạn sẽ thấy một điều rằng, thay vì làm 2 vòng**encoder**, và dùng 2 đèn LED đặt thẳng hàng, thì người ta chỉ cần làm 1 vòng lỗ, và đặt hai đèn LED lệch nhau.

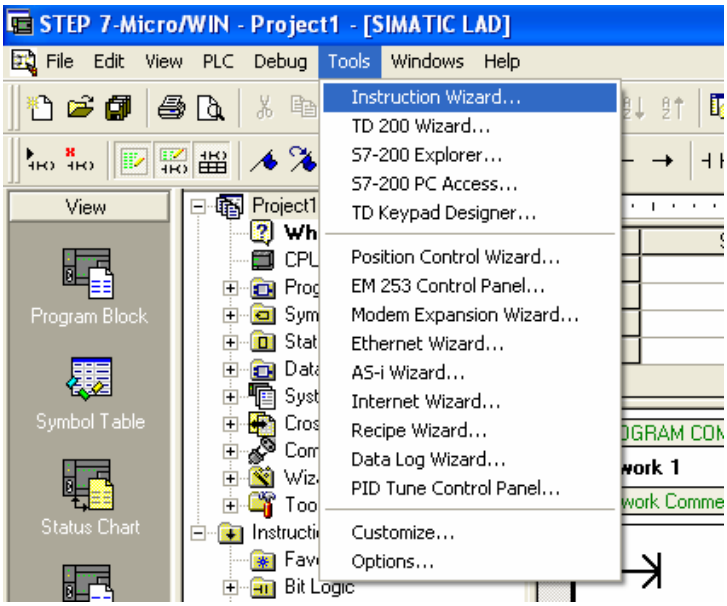
Kết quả, các bạn sẽ thường thấy các **encoder** có dạng như hình 2:



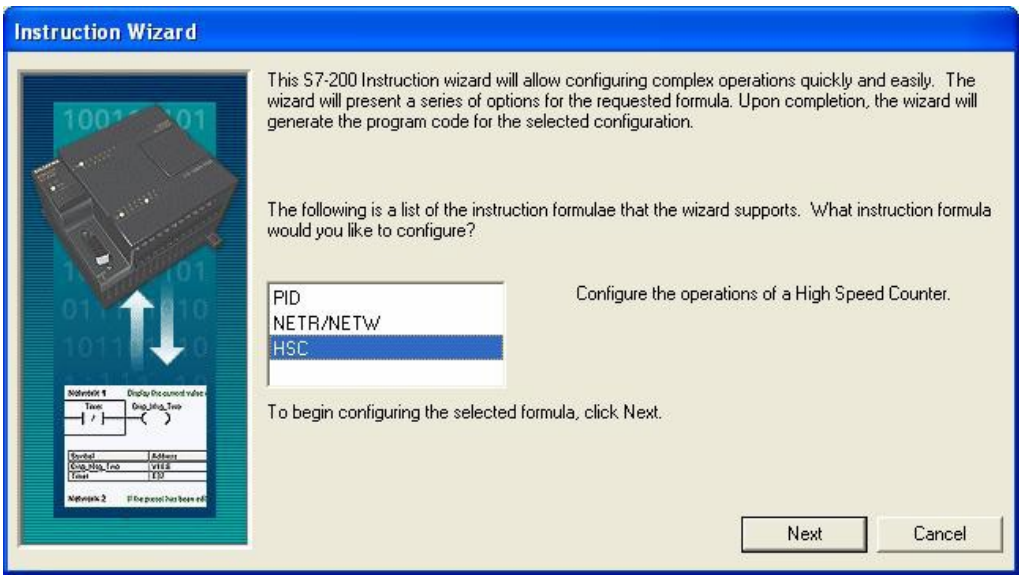
Đây là dạng **encoder** phổ biến nhất hiện nay

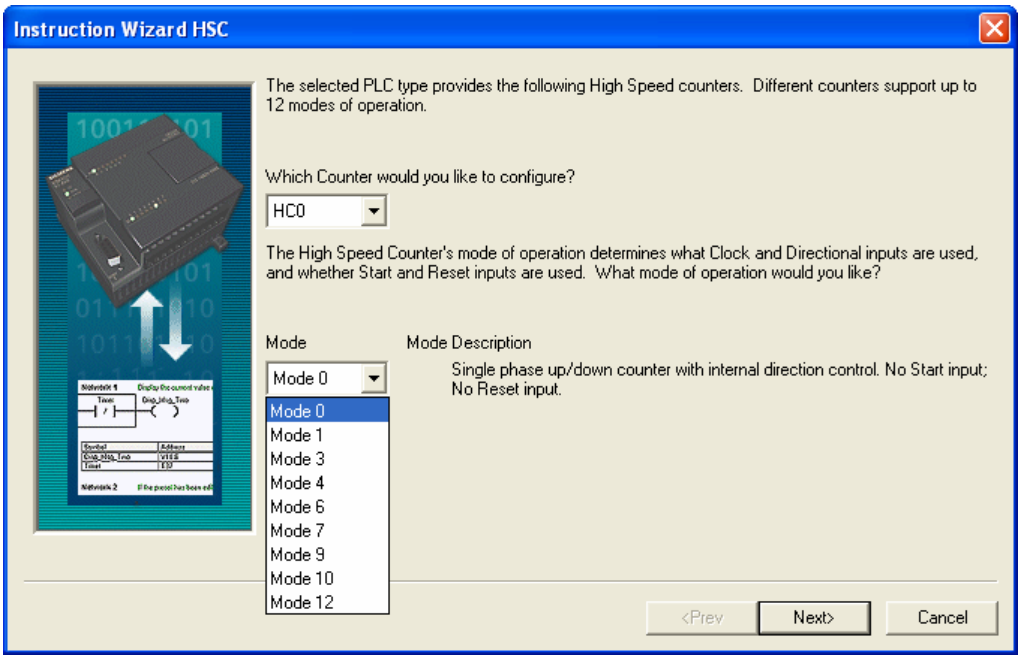
Đọc xung tốc độ cao HSC-PLC

Để đọc xung tốc độ cao, ta cần phải thực hiện cac bước cho vệc định dạng Wizard



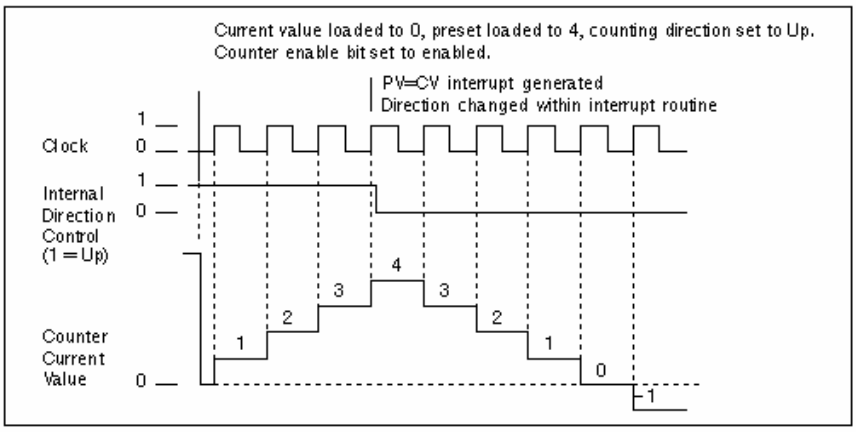
- Chọn Wizard đọc xung tốc độ cao High Speed Counter





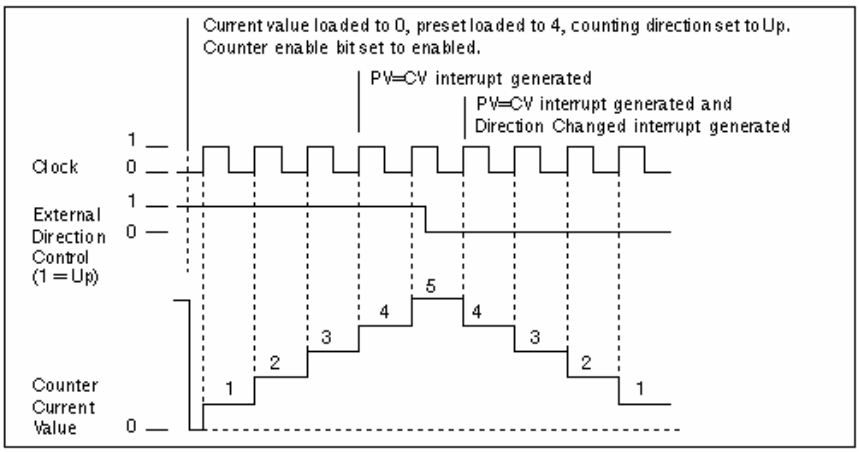
Chọn Mode đọc xung tốc độ cao và loại Counter n跻 (HC0,HC1…)
Tuỳ từng loại ứng dụng mà ta có thể chọn nhiều Mode đọc xung tốc độ cao kh
塊
nhau,có tất cả 12 Mode đọc xung tốc độ cao như sau

Operation Example of Modes 0, 1, and 2



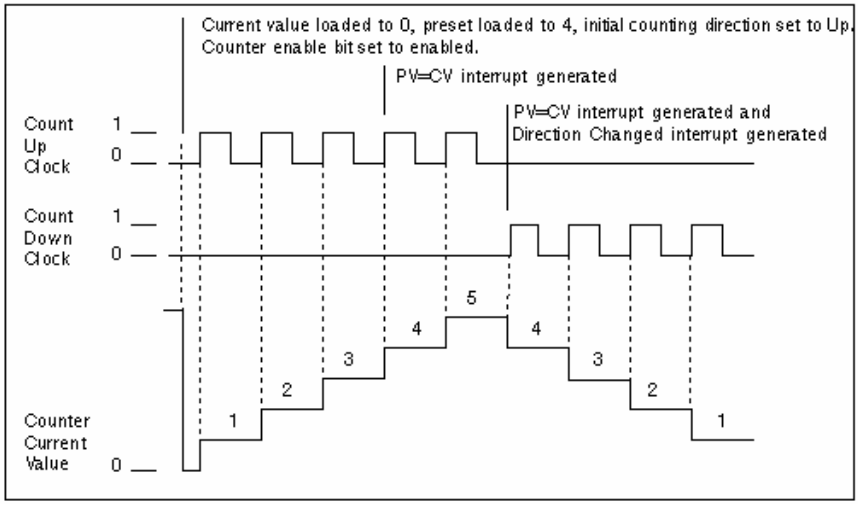
Mode 0,1,2 : D蠼g đếm 1 pha với hướng đếm được x墂 định bởi Bit nội
Mode 0: Chỉ đếm tăng hoặc giảm, kh獼g c?Bit Start cũng như bit Reset
Mode 1: Đếm tăng hoặc giảm, c?bit Reset nhưng kh 獼 g c?bit Start
Mode 2: Đếm tăng hoặc giảm, c?Bit Start cũng như bit Reset để cho ph齋 chọn bắt đầu đếm cũng như chọn thời điểm bắt đầu Reset. C墂 Bit Start cũng như Reset là
các ngõ Input chọn từ bên ngoài

Operation Example of Modes 3, 4, and 5



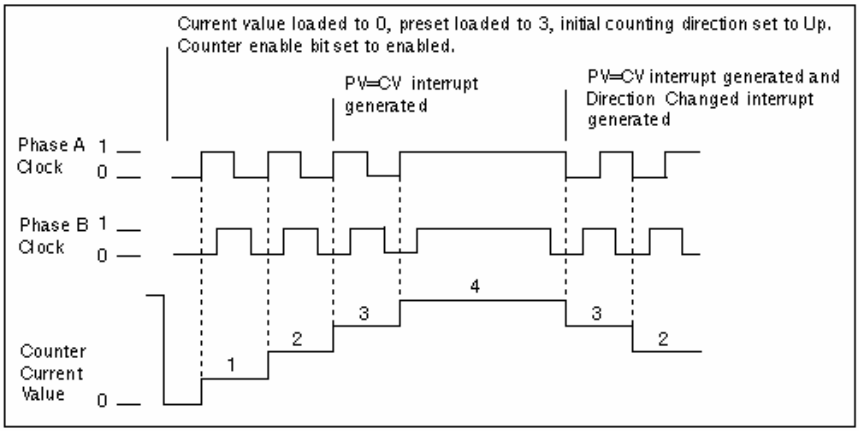
Mode 3,4,5: D蠼g đếm 1 pha với hướng đếm được x墂 định bởi Bit ngoại, tức là có
thể chọn từ ngõ vào input.
Mode 3: Chỉ đếm tăng hoặc giảm, kh獼g c?Bit Start cũng như bit Reset
Mode 4: Đếm tăng hoặc giảm, c?bit Reset nhưng kh獼g c?bit Start
Mode 5: Đếm tăng hoặc giảm, c?Bit Start cũng như bit Reset để cho ph齋 chọn bắt
đầu đếm cũng như chọn thời điểm bắt đầu Reset. C墂 Bit Start cũng như Reset l?c墂
ngõ Input chọn từ bên ngoài.

Operation Example of Modes 6, 7, and 8

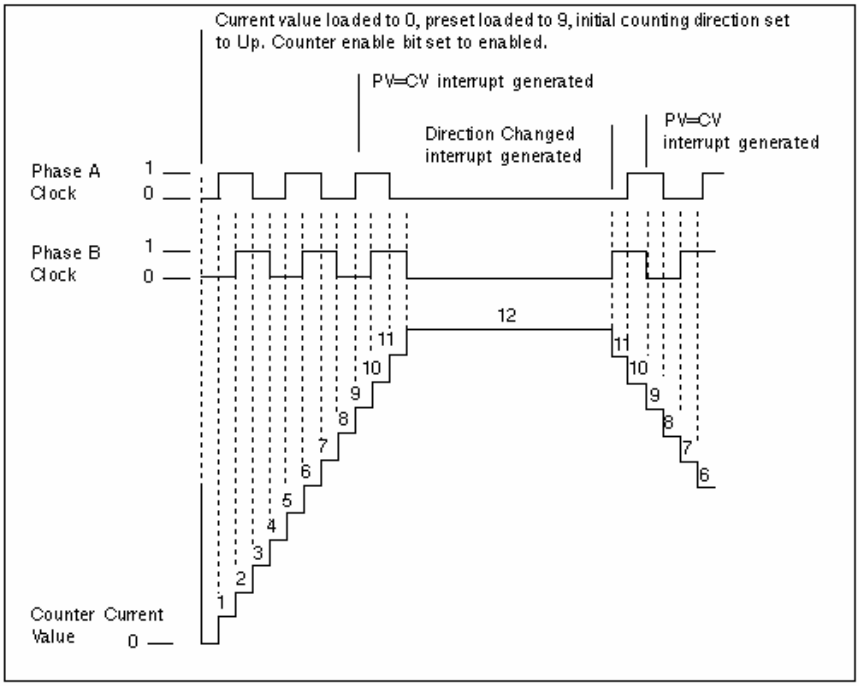


Mode 6,7,8: D蠼g đếm 2 pha với 2 xung v跻, 1 xung d蠼g để đếm tăng và một xung
đếm giảm
Mode 6: Chỉ đếm tăng giảm, kh獬g c?Bit Start cũng như bit Reset
Mode 7: Đếm tăng giảm, c?bit Reset nhưng kh獬g c?bit Start
Mode 8: Đếm tăng giảm, c?Bit Start cũng như bit Reset để cho ph廧 chọn bắt đầu
đếm cũng như chọn thời điểm bắt đầu Reset. C塶 Bit Start cũng như Reset l?c 塶 ngõ
Input chọn từ bên ngoài.

Operation Example of Modes 9,10, and 11 (Quadrature 1x Mode)



Operation Example of Modes 9,10, and 11 (Quadrature 4x Mode)



Mode 9,10,11 : D 蠼 g để đếm xung A/B của Encoder, c12 2 dạng
Dạng 1 (**Quadrature 1x mode**): Đếm tăng 1 khi c?xung A/B quay theo chiều thuận,
và giảm 1 khi c?xung A/B quay theo chiều ngược.
Dạng 2 (**Quadrature 4x mode**): Đếm tăng 4 khi c?xung A/B quay theo chiều thuận,
và giảm 4 khi c?xung A/B quay theo chiều ngược.
Mode 9: Chỉ đếm tăng giảm, kh獼g c?Bit Start cũng như bit Reset
Mode 10: Đếm tăng giảm, c?bit Reset nhưng kh獼g c?bit Start
Mode 11: Đếm tăng giảm, c?Bit Start cũng như bit Reset để cho ph嗇 chọn bắt đầu đếm cũng như chọn thời điểm bắt đầu Reset. C墝 Bit Start cũng như Reset l?c墝
ngõ Input chọn từ bên ngoài.
Mode 12: Chỉ 嫚 dụng với HSC0 v?HSC3, HSC0 d蠼g để đếm số xung ph嫢 ra từ
Q0.0 v?HSC3 đếm số xung từ Q0.1 (Được ph嫢 ra ở chế độ ph嫢 xung nhanh) mà
không cần đấu phần cứng, nghĩa là PLC tự kiểm tra từ bên trong

Mode	HSC0			HSC3	HSC4			HSC5
	I0.0	I0.1	I0.2	I0.1	I0.3	I0.4	I0.5	I0.4
0	Clk	-	-	Clk	Clk	-	-	Clk
1	Clk	-	Reset	-	Clk	-	Reset	-
2	-	-	-	-	-	-	-	-
3	Clk	Direction	-	-	Clk	Direction	-	-
4	Clk	Direction	Reset	-	Clk	Direction	Reset	-
5	-	-	-	-	-	-	-	-
6	Clk Up	Clk Down	-	-	Clk Up	Clk Down	-	-
7	Clk Up	Clk Down	Reset	-	Clk Up	Clk Down	Reset	-
8	-	-	-	-	-	-	-	-
9	Phase A	Phase B	-	-	Phase A	Phase B	-	-
10	Phase A	Phase B	Reset	-	Phase A	Phase B	Reset	-
11	-	-	-	-	-	-	-	-

Mode	HSC1				HSC2			
	I0.6	I0.7	I1.0	I1.1	I1.2	I1.3	I1.4	I1.5
0	Clk	-	-	-	Clk	-	-	-
1	Clk	-	Reset	-	Clk	-	Reset	-
2	Clk	-	Reset	Start	Clk	-	Reset	Start
3	Clk	Direction	-	-	Clk	Direction	-	-
4	Clk	Direction	Reset	-	Clk	Direction	Reset	-
5	Clk	Direction	Reset	Start	Clk	Direction	Reset	Start
6	Clk Up	Clk Down	-	-	Clk Up	Clk Down	-	-
7	Clk Up	Clk Down	Reset	-	Clk Up	Clk Down	Reset	-
8	Clk Up	Clk Down	Reset	Start	Clk Up	Clk Down	Reset	Start
9	Phase A	Phase B	-	-	Phase A	Phase B	-	-
10	Phase A	Phase B	Reset	-	Phase A	Phase B	Reset	-
11	Phase A	Phase B	Reset	Start	Phase A	Phase B	Reset	Start

Bảng M?tả chế độ đếm cũng như loại HSC, quy định địa chỉ vào.
Căn cứ vào bảng tr 瀍 để c?thể chọn loại HSC cho từng ứng dụng phù hợp

1 Số Bit được sử dụng để điều khiển các chế độ của HSC

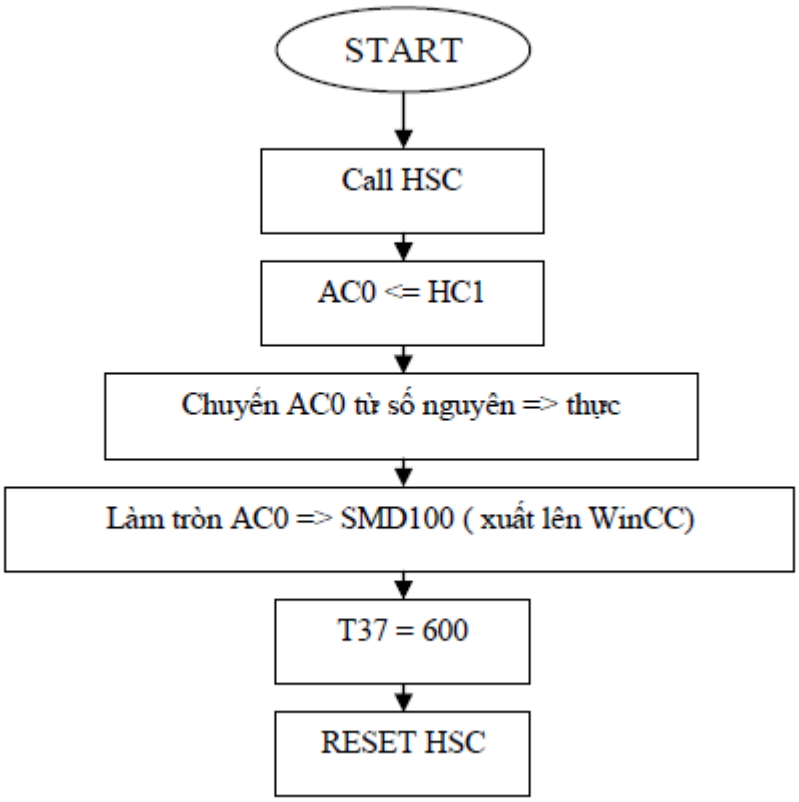
HSC0	HSC1	HSC2	HSC4	Description
SM37.0	SM47.0	SM57.0	SM147.0	Active level control bit for Reset**: 0 = Reset active high 1 = Reset active low
	SM47.1	SM57.1		Active level control bit for Start**: 0 = Start active high 1 = Start active low
SM37.2	SM47.2	SM57.2	SM147.2	Counting rate selection for Quadrature counters: 0 = 4x counting rate 1 = 1x counting rate

SM Control Bits for HSC Parameters

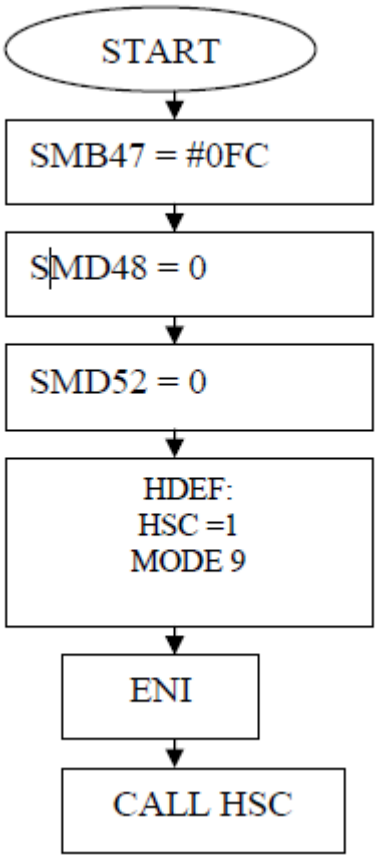
Ứng dụng: Đọc xung tốc độ cao từ Encoder

1. Lưu đồ thuật toán

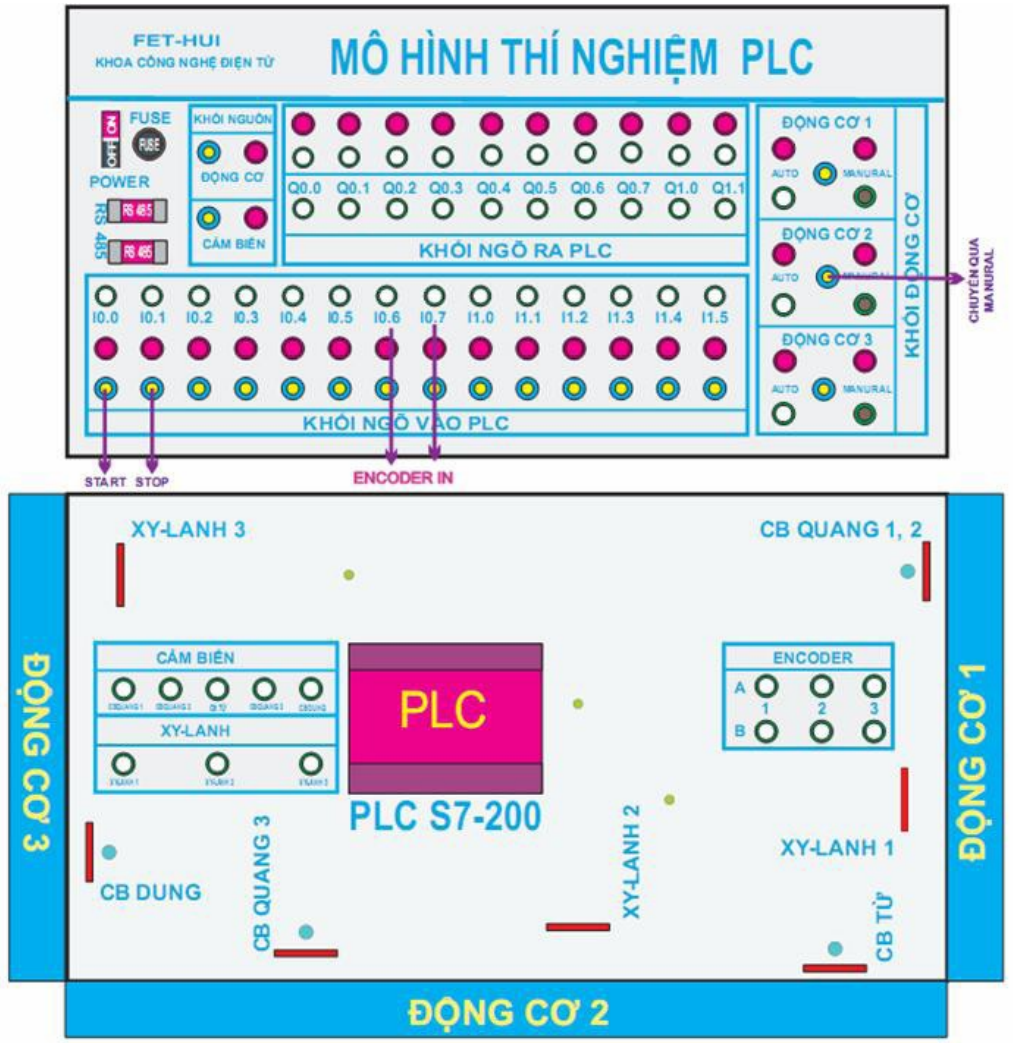
- **Lưu đồ giải thuật chương trình chính:**



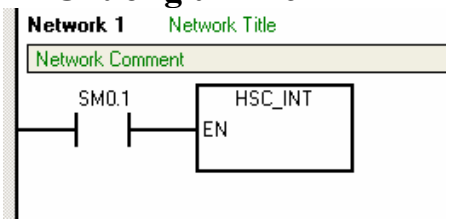
- **Lưu đồ giải thuật HSC**



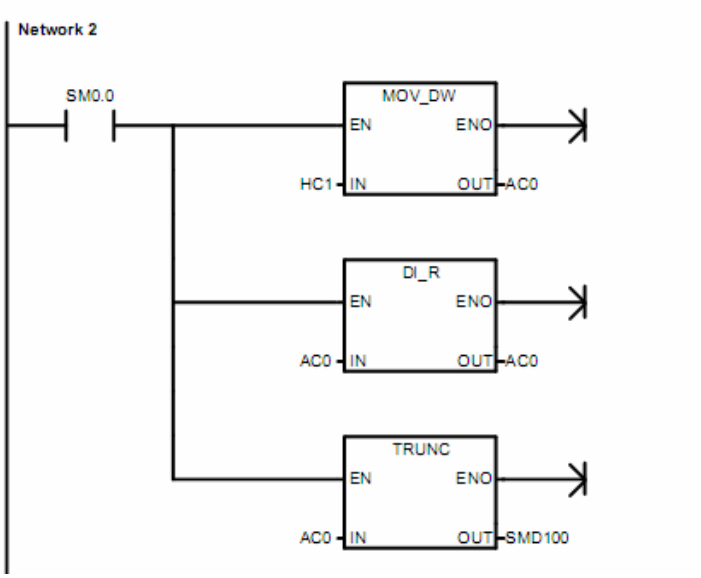
- **Sơ đồ kết nối:**



- **Chương trình chính**

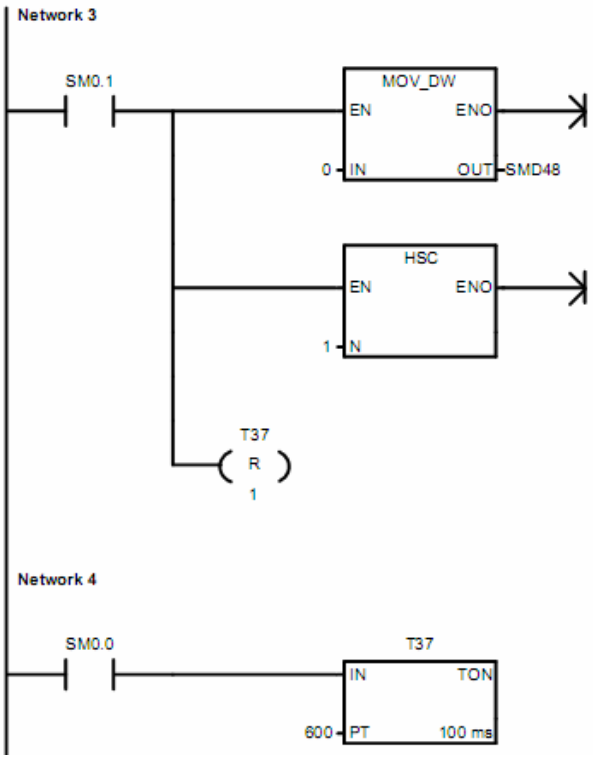


Chương trình con HSC được gọi ngay tại vòng quét đầu tiên của PLC

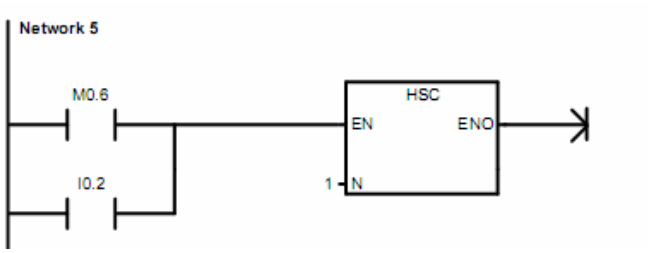


Dòng lệnh đầu là gán giá trị trong HSC (số xung đọc được) vào thanh ghi AC0, tiếp theo l?đổi số nguyên 32 bit sang số thực.

Câu lệnh thứ 3 là làm tròn số thực trong AC0 sang số DI và gán vào SMD100

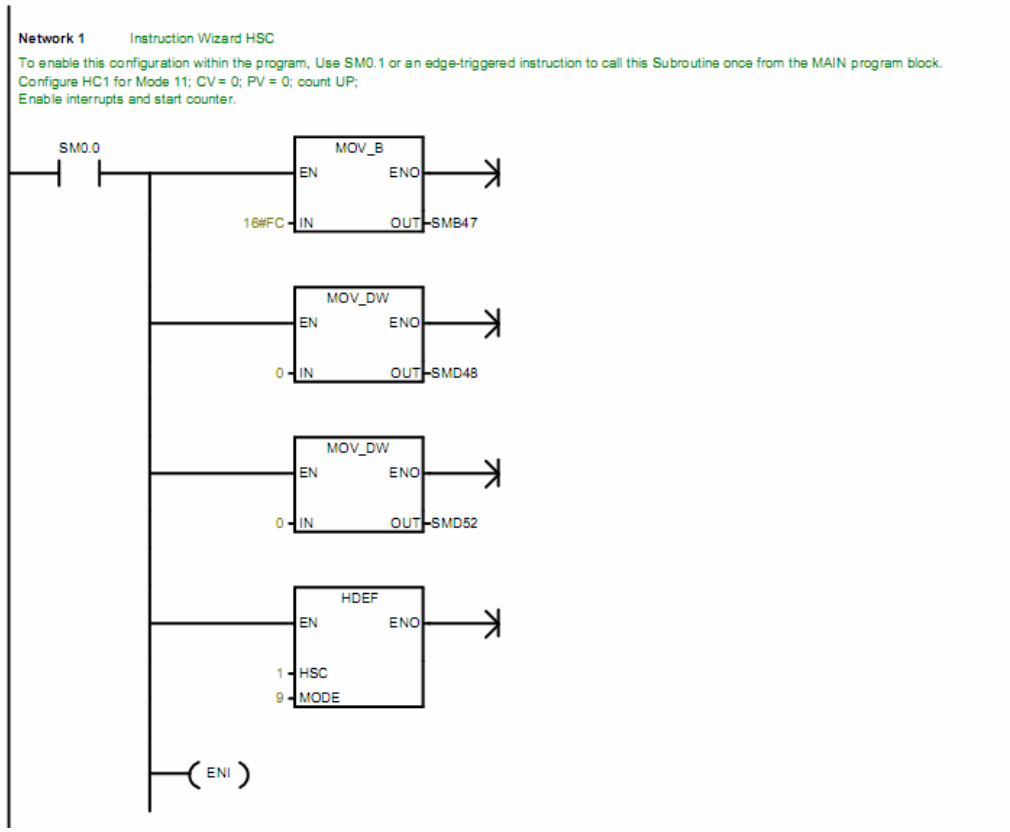


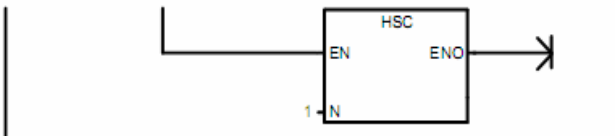
Nạp gi?trị ban đầu cho việc update và gọi chương trình HSC, đồng thời nu瀾 TIME để reset lại HSC



I0.2 nút nhấn RESET lại HSC

- **Chương trình con HSC**





Đ滶 l?c墂 dòng lệnh bắt buộc nhập c墂 gi?trị v跻 để c?thể đọc được gi?trị đưa vào từ encoder.
Nhập số #FC để chọn chế độ đọc là 1x hay 4x, nhập gi?trị 0 vào SMD48, SMD52
l?để nạp gi?trị hiện tại v?đặt cho việc update HSC, nạp gi?trị 1 v?9 v跻 HDEF để
chọn loại HSC và chế độ của n? C墂 gi?trị như thế nào xem kỹ trong gi嫘 trình

S7200.