CHƯƠNG 6: CÁC CHỨC NĂNG CHUYÊN DÙNG TRÊN PLC S7-200

6.1. Đo lường và giám sát nhiệt độ với module EM235 nhận cảm biến nhiệt điện trở Pt100:

Yêu cầu phần cứng:

1 S7-200 CPU
1 Pt100 Temperature Sensor
1 TD200 Operator Interface
1 EM235 Analog Expansion Module

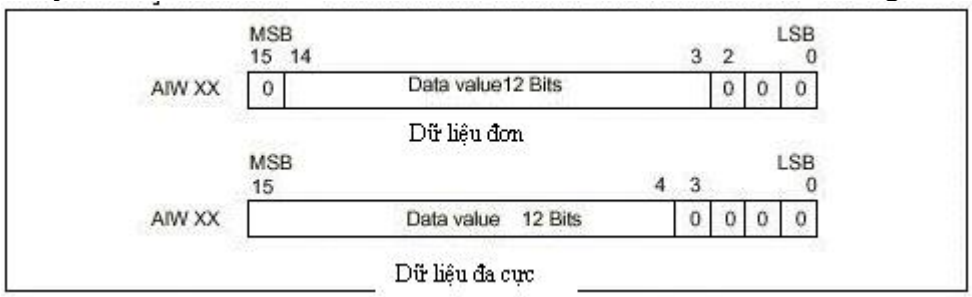
Chọn dãy điện áp trong giới hạn 0V÷10V cho EM235, bật các công tắc trên module theo các vị trí đã được ấn định tương ứng với từng dãy điện áp đầu và độ phân dải của tín hiệu vào theo bảng dưới đây:

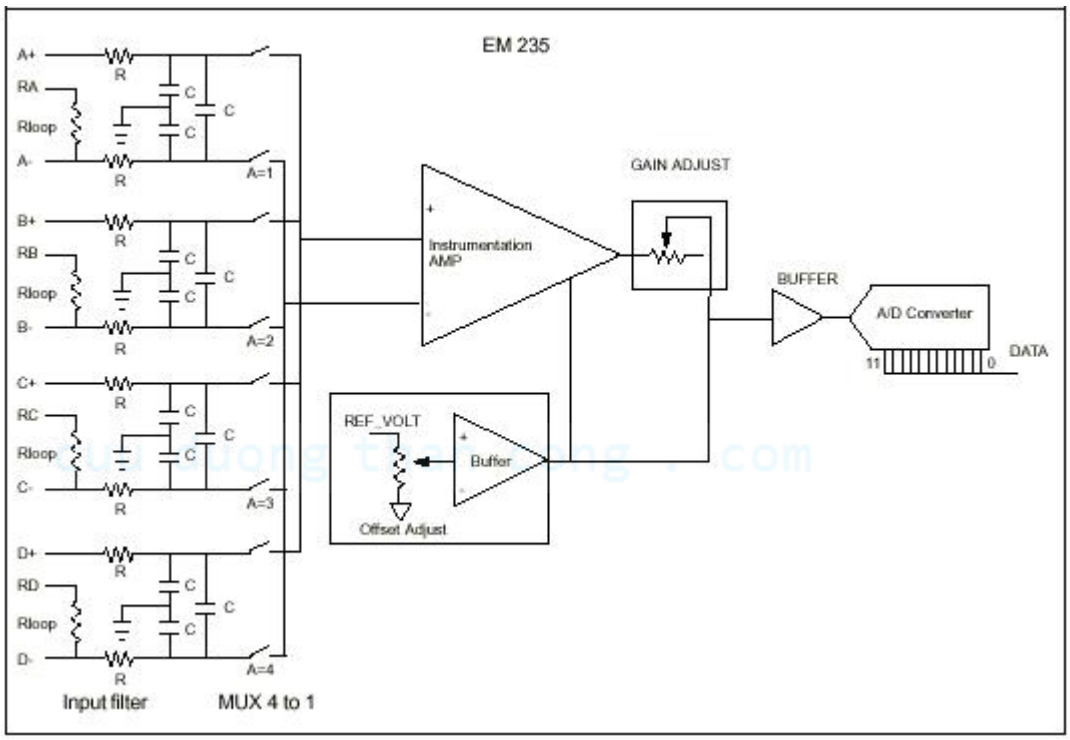
Không đảo dấu						Giới hạn dãy điện áp đầu vào	Độ phân dải
SW1	SW2	SW3	SW4	SW5	SW6		
ON	OFF	OFF	ON	OFF	ON	0 ÷ 50 mV	12.5 µV
OFF	ON	OFF	ON	OFF	ON	0 ÷ 100 mV	25 µV
ON	OFF	OFF	OFF	ON	ON	0 ÷ 500 mV	125 µV
OFF	ON	OFF	OFF	ON	ON	0 ÷ 1 V	250 µV
ON	OFF	OFF	OFF	OFF	ON	0 ÷ 5 V	12.5 mV
ON	OFF	OFF	OFF	OFF	ON	0 ÷ 20 mA	5 µA
OFF	ON	OFF	OFF	OFF	ON	0 ÷ 10 V	2.5 mV
Đảo dấu						Giới hạn dãy điện áp đầu vào	Độ phân dải
SW1	SW2	SW3	SW4	SW5	SW6		
ON	OFF	OFF	ON	OFF	OFF	±25 mV	12.5 µV
OFF	ON	OFF	ON	OFF	OFF	± 50mV	25 µV
OFF	OFF	ON	ON	OFF	OFF	± 100mV	50 µV
ON	OFF	OFF	OFF	ON	OFF	± 250 mV	125 µV
OFF	ON	OFF	OFF	ON	OFF	± 500 mV	250 µV
OFF	OFF	ON	OFF	ON	OFF	± 1V	500 µV
ON	OFF	OFF	OFF	OFF	OFF	± 2.5V	12.5 mV
OFF	ON	OFF	OFF	OFF	OFF	± 5V	25 mV
OFF	OFF	ON	OFF	OFF	OFF	± 10V	50 mV

SW6: chọn điện áp và dòng vào có dấu hoặc không dấu; SW4, SW5: chọn hệ số khuyếch đại; SW3,2,1: chọn hệ số suy giảm.

EM 235 Configuration Switches						Unipolar/Bipolar Select	Gain Select	Attenuation Select
SW1	SW2	SW3	SW4	SW5	SW6			
					ON	Unipolar		
					OFF	Bipolar		
			OFF	OFF			x1	
			OFF	ON			x10	
			ON	OFF			x100	
			ON	ON			invalid	
ON	OFF	OFF						0.8
OFF	ON	OFF						0.4
OFF	OFF	ON						0.2

Giá trị chuyển đổi ADC 12 bit của từ đơn đối với tín hiệu vào có/không có dấu:





Hình 6.1: Cấu trúc của module EM235

Tuỳ thuộc vào số kênh sử dụng trên module analog EM235 tương ứng với địa chỉ đầu vào (từ đơn) phải sử dụng trong quá trình lập trình: AWI0_cho channel 1, AWI2_cho channel 2, AWI4_cho channel 3.

Sau đây là chương trình gợi mở cho người sử dụng trong quá trình đo lường và giám sát nhiệt độ dựa trên hệ thống 1 module CPU, 1 module EM235, 1 cảm biến nhiệt điện Pt100 và 1 TD200 (Text Display).

Module tiến hành đọc giá trị nhiệt điện trở được biến thành giá trị điện áp theo bậc. Đầu đầu ra analog được sử dụng như hằng số của nguồn dòng. Dòng cung cấp cho Pt100 là 12.5 mA nguồn dòng.

Với mạch này đầu vào là tuyến tính của 5mV/1°C. Giá trị analog của đầu vào được số hoá qua hệ thống biến đổi ADC và được đọc đều đặn theo chu kỳ. Từ giá trị này, chương trình sẽ thực hiện tính toán và chuyển đổi theo công thức sau:

T[°C] = (Digital value - 0°C offset)/ 1°C value

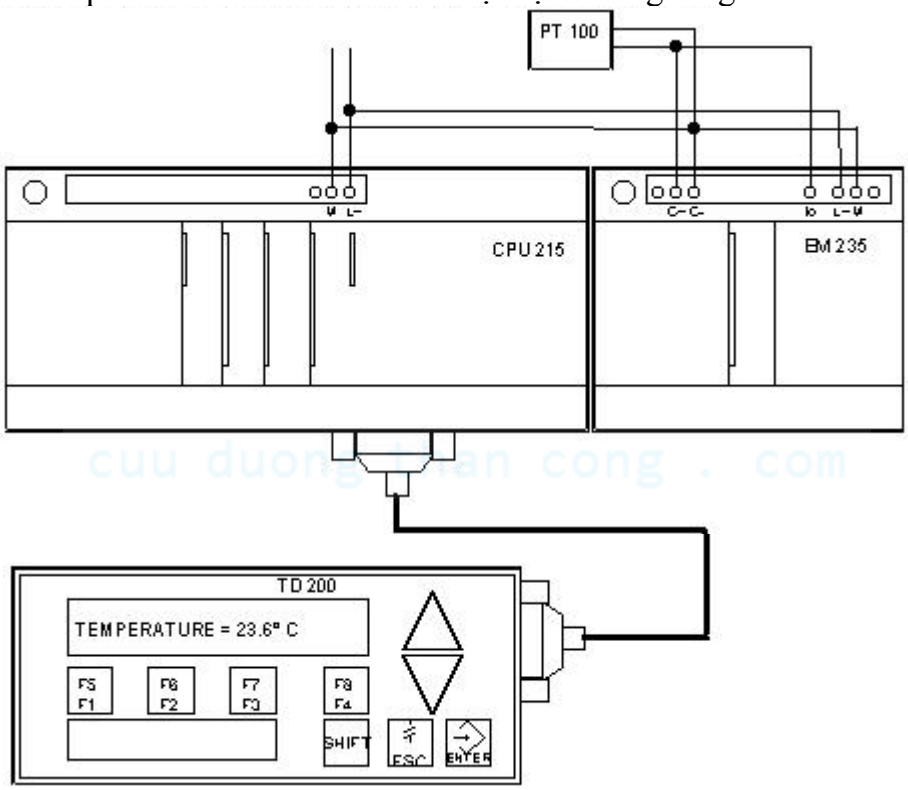
Digital value: giá trị đầu vào analog đã được chuyển đổi.

0°C offset: giá trị số, được đo ở 0°C; trong ví dụ này giá trị offset là 4000.

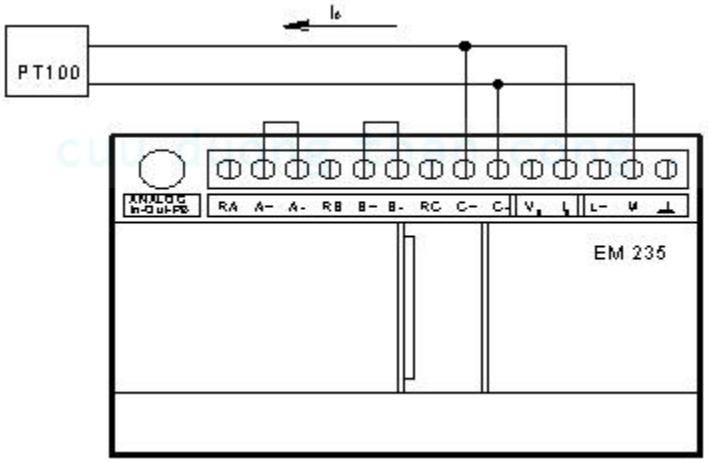
1°C value: giá trị tương ứng với 1°C, trong ví dụ này thì 1°C =16.

Chương trình tính toán giá trị thập phân và ghi kết quả vào biến của message1: "Temperature = xxx.x°C" giá trị này được hiển thị trên TD200. Trước khi khởi tạo chương trình này, phải xác định được giới hạn nhiệt độ thấp nhất và nhiệt cao nhất.

Nếu nhiệt độ vượt quá ngưỡng thì sẽ xuất hiện dòng cảnh báo trên TD200. Xuất hiện dòng thông báo Message 2: "Temperature > xxx.x°C" nếu nhiệt độ vượt quá ngưỡng. Message 3: "Temperature < xxx.x°C" nếu nhiệt độ dưới ngưỡng.



Hình 6.2: Cách lắp TD200 với CPU và module EM235



Hình 6.3: Cách lắp ghép cảm biến với module EM235

Chương trình viết trên Step 7 bằng ngôn ngữ STL:

Network 1: Set the High and Low Temperature Limits

```
LD    First_Scan_On:SM0.1      // In the first scan cycle,
MOVD  +0, VD196                // clear VW196 and VW198.
MOVW  +16, VW250               // Load 1° C = 16 in VW250
MOVW  +4000, VW252             // Set the 0° C offset = 4000.
MOVW  +300, VW260              // Set the high temperature
                               // limit = 30° C.
MOVW  +200, VW262              // Set the low temperature
                               // limit = 20° C.
MOVW  +20000, AQW0             // Initialize a 12.5 mA current
                               // at analog output word AQW0.
```

Network 2: Calculate the Value and Enable Message 1

```
LD    Always_On:SM0.0          // Every scan cycle,
MOVW  AIW4, VW200              // move the value in analog
                               // input word AIW4 to VW200.
-I    VW252, VW200             // Subtract the 0° C offset.
DIV   VW250, VD198             // Divide the result by the 1° C
                               // value.
MUL   +10, VD196               // Multiply the remainder by 10.
DIV   VW250, VD196             // Divide the value in variable
                               // double word VD196 (remainder x 10)
                               // by the 1° C value.
MOVW  VW198, VW160             // Shift the quotient by 1 decimal
                               // point to the left.
MOVW  +0, VW198                // Clear VW198.
MUL   +10, VD198               // Multiply the temperature value
                               // by 10.
+I    VW160, VW200             // Add the result of temperature
                               // value x 10 with the value that
                               // is stored as the digit following
                               // the decimal point.
MOVW  VW200, VW116             // Transfer the result to VW116
                               // (embedded value on the TD 200)
                               // for display.
S     V12.7, 1                 // Enable message 1 for display
                               // on the TD 200.
```

Network 3: If Temperature Exceeds High Limit, Enable Message 2 and Turn Off Furnace

```
LDW>= VW200, VW260             // If the temperature value >=
                               // the high temperature limit
                               // stored in VW260,
=     V12.6                    // enable message 2 on the TD 200.
R     Q0.0, 1                  // Turn off the furnace.
```

```
MOVW   VW260, VW136          // Move the high temperature limit
                             // value to VW136 (embedded value
                             // on the TD 200) for display
                             // in message 2.
```

Network 4: If Temperature Drops Below Low Limit, Enable Message 3 and Turn On Furnace

```
LDW<=  VW200, VW262          // If the temperature value <=
                             // the low temperature limit
                             // stored in VW262,
=      V12.5                 // enable message 3 on the TD 200.
S      Q0.0, 1               // Turn on the furnace.
MOVW   VW262, VW156          // Move the low temperature limit
                             // value to VW156 (embedded value
                             // on the TD 200) for display in
                             // message 3.
```

Network 5: Main Program End

6.2. Đo lường và giám sát nhiệt độ với module EM235 nhận cảm biến truyến tính nhiệt điện Pt100:

Yêu cầu phần cứng:

1 S7-200 CPU
1 Pt100 Temperature Detector
1 TD200 Operator Interface
1 EM235 Analog Expansion Module

Đây là chương trình gợi mở làm thế nào để có thể đo và giám sát trong phạm vi giới hạn theo danh nghĩa lý thuyết sử dụng module mở rộng analog EM235. Nhờ đó đầu dò nhiệt độ Pt100 là được kết nối tới kênh vào analog của module.

Qúa trình chuyển đổi điện trở trên Pt100 thành nhiệt độ dựa trên sự chuyển đổi điện áp. Nguồn nôi Pt100 được sử dụng như 1 nguồn dòng. Tín hiệu cung cấp có dòng ổn định ở mức 2.5mA cho đầu dò Pt100. Với mạch điện này, điện áp đầu vào thay đổi tuyến tính của 1mV/°C.

EM235 chuyển đổi giá trị analog (áp) thành digital được thực hiện tuần tự theo chu kỳ. Chương trình tính toán nhiệt độ dựa tren công thức sau:

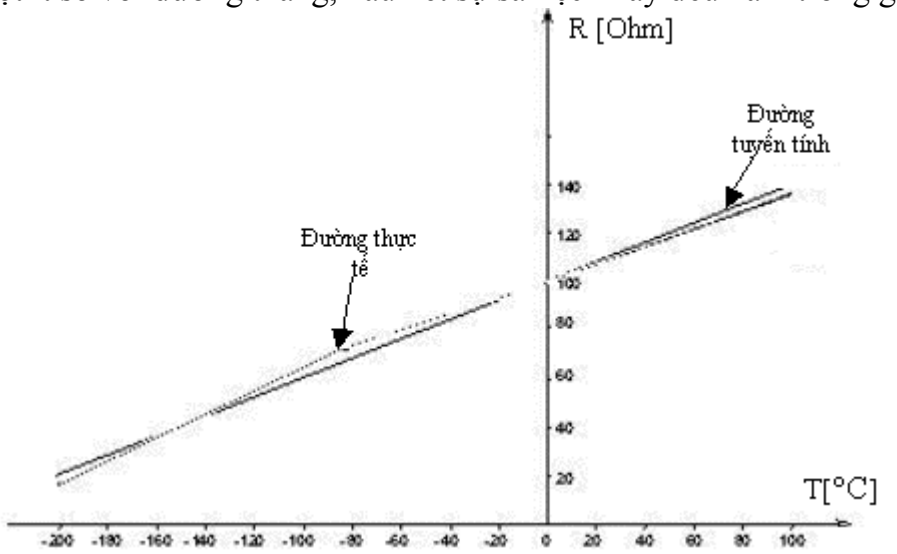
$$T[°C] = (t_e - t_o)/t_l$$

t_e : giá trị số đọc trực tiếp từ kênh đầu vào AWIx(x = 0,2,4)
t_o : giá trị số, đo ở 0°C (°C offset)
t_l : số nguyên tương ứng với 1°C

Chương trình tính toán giá trị thập phân và ghi kết quả vào biến nhớ của Message 1: "Temperature xxx.x°C" kết quả này được hiển thị trên TD200.

Trong quá trình khởi tạo, phải chỉ định rõ vùng giới hạn (giá trị thấp nhất và giá trị cao nhất). Ngoài ra trên TD200 còn xem được cảnh báo nếu nhiệt độ vượt quá giới hạn ấn định trước. Cách lắp TD200 với CPU và module EM23 xem hình 2.

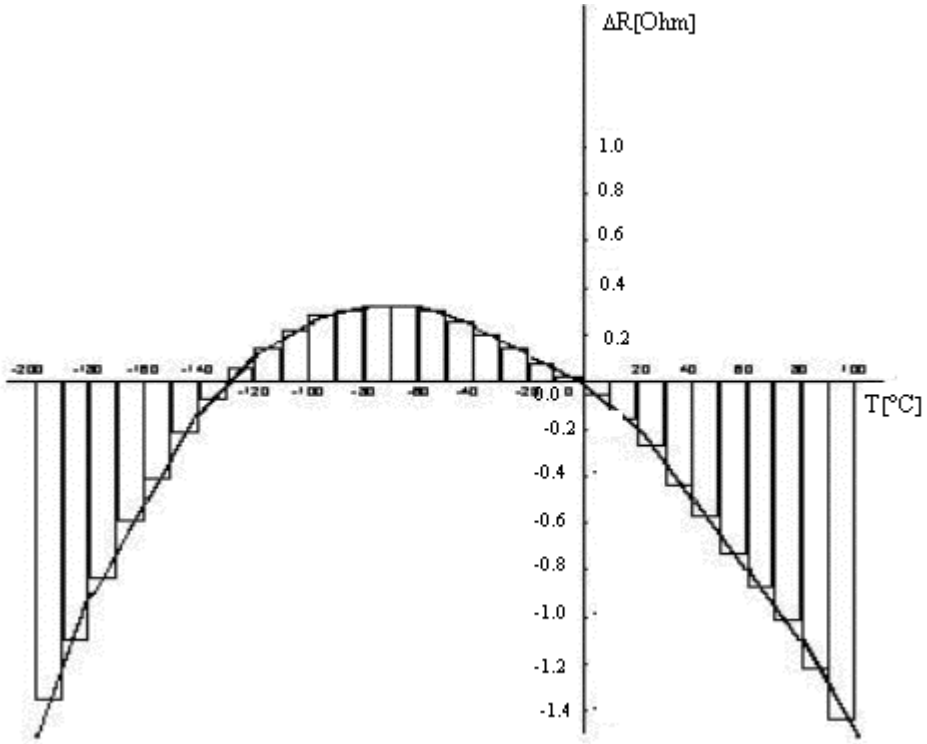
Đo điện trở shunt của Pt100 sử dụng ở ví dụ này là phù hợp trong giới hạn nhiệt độ từ -200°C÷100°C. Đường đặc tính của Pt100 xem bên dưới, nó không hoàn toàn tuyến tính. Sai khác một ít so với đường thẳng, hầu hết sự sai lệch này đều nằm trong giới hạn.



Hình 6.4: Đường đặc tính nhiệt điện trở của Pt100

Nhiệt độ trong giới hạn từ -200°C ÷ -130°C và từ 0°C ÷ 100°C.Nhiệt độ đo được ít hơn giá trị thực tế một ít và phải bù thêm.

Sự sai lệch về điện trở phụ thuộc vào nhiệt độ, xem hình bên dưới. Trong trường hợp này ta phân dãy nhiệt độ ra làm 30 đoạn, 10°C cho mỗi đoạn. Nhờ đó ta mới tìm được giá trị độ lệch trung bình cho từng đoạn. Kết quả độ lệch trong 30 đoạn này sẽ được sử dụng trong suốt quá trình "tuyến tính hoá" của chương trình bằng cách bù giá trị nhiệt độ tương ứng cho từng đoạn.



Hình 6.5: Bù giá trị nhiệt tương ứng cho từng đoạn

Điện trở thay đổi 0.4 Ω tương ứng với nhiệt độ thay đổi 1°C. Giá trị bù có thể chuyển đổi sang °C và có thể đưa trực tiếp vào chương trình tính toán nhiệt độ. Giá trị bù được liệt kê theo bảng sau:

TempRange[°C]	-200 to -190	-190 to -180	-180 to -170	-170 to -160	-160 to -150
CompValue [°C]	3.5	2.6	2.0	1.5	1.0
TempRange[°C]	-150 to -140	-140 to -130	-130 to -120	-120 to -110	-110 to -100
CompValue [°C]	0.6	0.1	-0.2	-0.4	-0.5
TempRange[°C]	-100 to -90	-90 to -80	-80 to -70	-70 to -60	-60 to -50
CompValue [°C]	-0.6	-0.7	-0.8	-0.8	-0.7
TempRange[°C]	-50 to -40	-40 to -30	-30 to -20	-20 to -10	-10 to 0
CompValue [°C]	-0.6	-0.5	-0.4	-0.2	-0.1
TempRange[°C]	0 to 10	10 to 20	20 to 30	30 to 40	40 to 50
CompValue [°C]	0.1	0.3	0.6	1.0	1.4
TempRange[°C]	50 to 60	60 to 70	70 to 80	80 to 90	90 to 100
CompValue [°C]	1.6	2.1	2.5	3.0	3.5

Trong suốt quá trình thiết lập, giá trị hiệu chỉnh được lưu lại trong vùng nhớ đệm và sau đó chương trình sẽ thêm vào trong giá trị nhiệt độ đo được. Giá trị điện trở đo được của Pt100 ở 0°C là 100Ω. Điện trở thay đổi tuyến tính theo nhiệt độ theo hệ số 0.4Ω/°C. Nguồn nuôi cung cấp cho cảm biến phải là nguồn dòng ổn định 2.5mA.

Dãy điện áp lựa chọn từ 0V ÷ 1V, trong đó độ phân dải là 10μA/đơn vị. Như vậy 2.5mA được quy đổi thành 250 đơn vị. Chọn giá trị ngưỡng thấp tương ứng với 2.5mA là 4000, từ phương trình biến đổi sau: (32000*2.5mA)/20mA = 4000.

Lựa chọn điện áp trong giới hạn từ 0V ÷ 1V bằng cách lựa chọn các công tắc theo các chế độ như sau:

Switch: 1 3 5 7 9 11
ON OFF ON OFF ON OFF

Cách lắp ghép cảm biến với module EM235 xem hình 3.

Chương trình viết trên Step 7 bằng ngôn ngữ STL:

Network 1: Initialize the Current for the Pt100

```
LD    First_Scan_On:SM0.1      // In the first scan cycle,
MOVW  +4000, AQW0              // move 4000 into analog output
                               // word AQW0 to initialize a
                               // 2.5 mA current for the Pt100.
```

Network 2: Load the Measured Value and Calculate the Temperature

```
MOVW  AIW4, VW200              // load measured value from AIW4
                               // in VW200.
-I    VW252, VW200             // Subtract the 0º C offset from
                               // the temperature value.
DIV   VW250, VW198             // Divide the result by ºC.
MUL   +10, VW196               // Multiply the remainder by 10. . .
DIV   VW250, VW196             // Divide the result by the ºC
                               // value and add the resulting
                               // value to the first
                               // position after decimal.
MOVW  VW198, VW160             // Move VW198 to temporary
                               // location VW160.
MOVW  +0, VW198                // Clear VW198.
MUL   +10, VD198               // Multiply the temperature value
                               // by 10.
+I    VW160, VW200             // Add the temperature value and
                               // the value in the first position
                               // after the decimal to determine
                               // the exact temperature.
```

Network 3: Enable Message 2 On the TD 200

```
LDW>= VW200, VW260             // If the temperature value measured
                               // >= the high limit,
R     V12.5, 3                 // reset all three TD 200 messages.
=     V12.6                    // Enable the TD 200 message,
                               // "Temperature>".
MOVW  VW260, VW136             // Move the high limit into the
                               // TD 200 embedded value display.
JMP   1                        // Jump to Label 1.
```

Network 4: Enable Message 3 On the TD 200

```
LDW<= VW200, VW262             // If the temperature value measured
                               // <= the low limit,
R     V12.5, 3                 // reset all three TD 200 messages.
=     V12.5                    // Enable the TD 200 message,
```

```
                                    // "Temperature<".
MOVW   VW262, VW156          // Move the low limit into the
                                    // TD 200 embedded value display.
JMP    1                            // Jump to Label 1.
```

Network5: Find the Compensation Value and Display the Temperature

```
LD     Always_On:SM0.0              // Every scan cycle,
MOVD   +0, AC1                      // load the starting address for
                                    // the temperature table B
                                    // into accumulator AC1.
FND>   VW398, VW200, AC1            // Begin searching table B at
                                    // VW398 until the value stored
                                    // in VW200 is found.
                                    // Then, place the index value
                                    // in accumulator AC1.
MOVD   &VB300, AC2           // Load the starting address of
                                    // table A into AC2.
MUL    +2, AC1               // Multiply the index by 2.
+D     AC1, AC2                     // Add the index to the starting
                                    // address.
MOVW   *AC2, VW116           // Move the adjustment value into
                                    // VW116.
+I     VW200, VW116          // Add the adjustment value to
                                    // the measured temperature to
                                    // get the true value.
S      V12.7, 1                     // Enable the first TD 200 message,
                                    // "Temperature=".
```

Network 6: Label One

```
LBL    1             // This is the destination for
                     // the Jump to Label instruction
                     // in Network 3 and Network 4.
```

Network 7: Main Program End

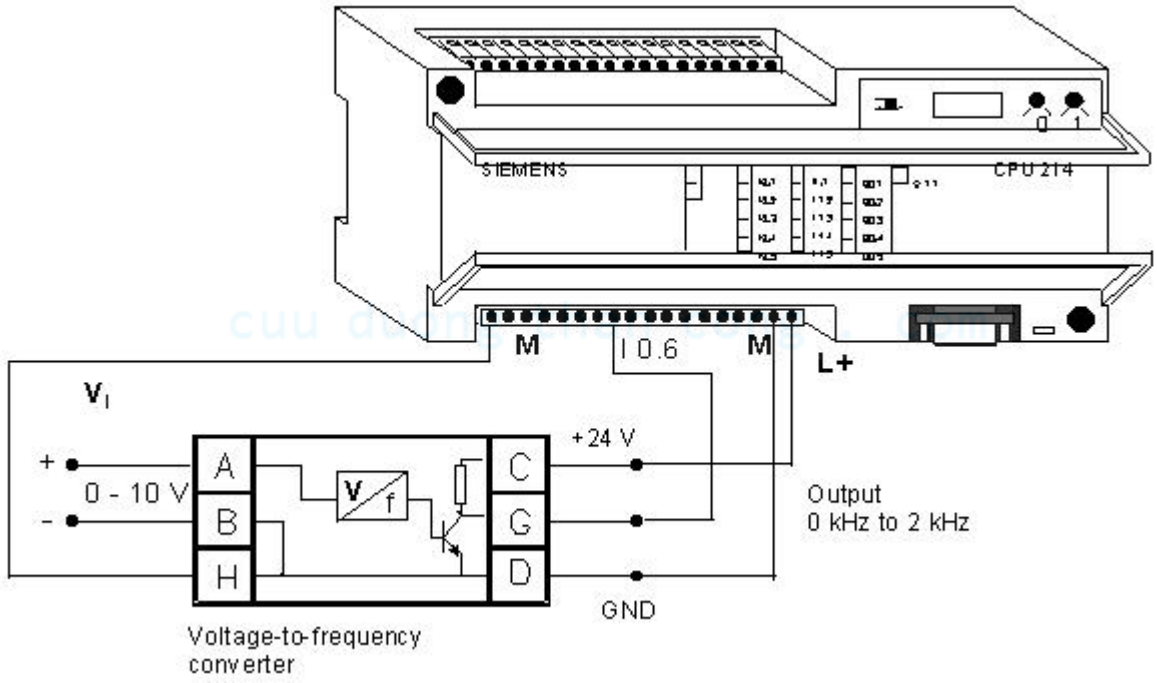
6.3. Cách sử dụng bộ đêm tốc độ cao để ghi lại giá trị analog bằng cách chuyển đổi giá trị analog sang tần số:

Yêu cầu phần cứng:

- Trong phần này có sử dụng đầu ra xung để phục vụ cho mục đích điều khiển nên phải sử dụng PLC DC/DC/DC CPU loại 214, 215, 216, 221, 222, 224, 224XP, 226, 226XM.
- Bộ chuyển đổi điện áp sang tần số loại SFW01 (Trnker Commpany), có tiêu chuẩn kỹ thuật như sau:

+ Nguồn cung cấp: 24VDC
+ Áp vào: 0VDC ÷ 10VDC
+ Đầu ra: Sóng xung vuông, 24VDC-GND
+ Giới hạn đo: 0VDC ÷10VDC ≈ 0Hz ÷ 2000Hz
+ Ratio (độ tăng tuyến tính): 200Hz/V

Mô tả: Với sự trợ giúp của bộ chuyển đổi điện áp sang tần số, bộ đếm tốc độ cao (HSC) của PLC 214 được sử dụng để ghi lại giá trị điện áp này. Bộ chuyển đổi sử dụng điện áp vào từ 0V ÷ 10V. Giá trị này được chuyển đổi sang dãy xung vuông có tần số tương ứng 0Hz ÷ 2000Hz. Tín hiệu này được đưa vào bộ đếm tốc độ cao của CPU 214. Dãy xung này sẽ được đếm, sau khoảng thời gian định trước, lượng xung sẽ được ghi và giá trị điện áp được tính toán.



Hình 6.6: Cách lắp bộ biến đổi điện áp sang tần số với đầu vào của bộ đếm tốc độ cao

Chương trình viết trên Step 7 bằng ngôn ngữ STL:

Main Program (OB1):

Network 1: *Call Subroutine SBR0*

```
LD     First_Scan_On:SM0.1        // Load SM0.1.
CALL   SBR_0:SBR0                 // Call SBR0.
```

Network 2: *Main Program End*

Subroutine Program (SBR0):

Network 1: *Subroutine SBR0*

Network 2: *Initialize High-Speed Counter and Enable Timed Interrupt*

```
LD     Always_On:SM0.0            // Load SM0.0.
MOVB   16#FC, HSC1_Ctrl:SMB47     // Load control bits for HSC1.
```

```
HDEF   1, 0                    // Assign mode 0 to HSC1.
MOVD   +0, HSC1_CV:SMD48       // Set the new current value of
                               // HSC1.
MOVD   16#0000FFFF, HSC1_PV:SMD52  // Set the new preset value of
                               // HSC1 (not used in this example).
MOVB   100, Time_0_Intrvl:SMB34    // Set the time interval for
                               // timed INT0 = 100 ms.
ATCH   INT_0:INT0, 10          // Attach interrupt event 10 to
                               // INT0.
ENI                            // Enable all interrupt events.
HSC    1                       // Start HSC1.
```

Network 3: *End of Subroutine SBR0*

Interupt Program (INT0):

Network 1: *Interrupt Routine INT0*

Network 2: *Evaluate High-Speed Counter HSC1*

```
LD     Always_On:SM0.0         // Load SM0.0.
MOVD   HC1, VD100              // Move the value in HSC1 to
                               // VD100 to store the current
                               // count.
MOVD   +0, HSC1_CV:SMD48       // Reset the current value (CV)
                               // of HSC1 = 0.
MOVB   16#C0, HSC1_Ctrl:SMB47      // Enable HSC1 and update current
                               // value (CV).
HSC    1                       // Start HSC1.
SRD    VD100, 1                // Divide the count stored in
                               // VD100 in half.
MOVB   VB103, Display_Voltage:QB0  // Display the result at QB0.
                               // (10 times the voltage).
```

Network 3: *End of Interrupt Routine INT0*

6.4. Cách đo mức từ đầu vào analog:

Yêu cầu phần cứng:

- EM235 mudule, vị trí các DIP Switching như sau:

1	3	5	7	9
off	off	on	off	off

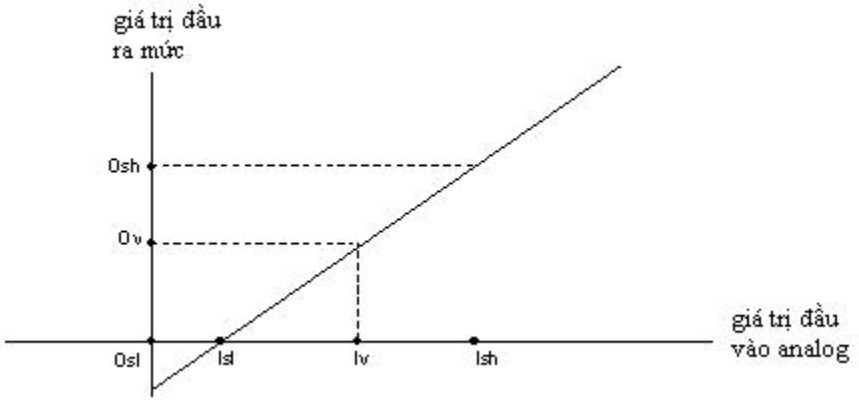
- TD200 (Text Display)

Đây là chương trình đọc giá trị analog từ kênh vào của S7-200 và cung cấp cho giá trị đầu ra mức. Tất cả các giá trị yêu cầu phải được cung cấp với giá trị được thiết lập trong chương trình. Các biến sau đây được đưa vào vị trí thích hợp trong công thức:

Ov : giá trị đầu ra mức
Iv : giá trị đầu vào analog
Osh : giới hạn mức cao cho giá trị đầu ra
Osl : giới hạn mức thấp cho giá trị đầu ra

Ish : giới hạn trên của giá trị dầu vào analog
Isl : giới hạn dưới của giá trị dầu vào analog

Sự quan hệ giữa giá trị đầu vào analog và giá trị đầu ra mức được thể hiện theo đồ thị sau:

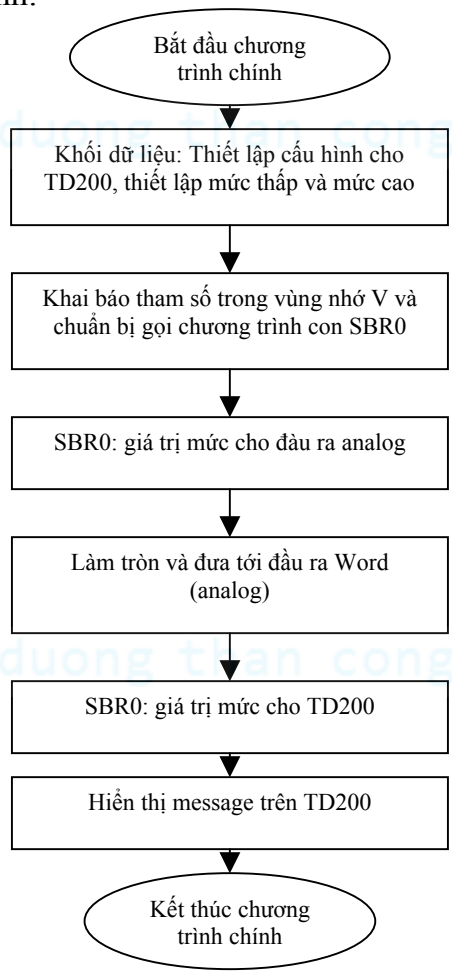


Hình 6.7: Đồ thị biểu diễn quan hệ giữa đầu vào analog và đầu ra mức

Công thức sau tính toán giá trị mức có thể say ra từ đồ thị:

Ov = [(Osh - Osl)*(Iv - Isl)/(Ish - Isl)] + Osl

Thuật toán của chương trình:



Chương trình viết trên Step 7 bằng ngôn ngữ STL:
Main Program (OB1):
Network 1: *Read the Analog Input Value and Convert to Real*

```
LD    Always_On:SM0.0          // Load SM0.0.
MOVD  +0, AC1                  // Move 0 to AC1 to clear AC1.
MOVW  AIW0, AC1                // Move the value in AIW0 into AC1.
DTR   AC1, VD500               // Convert the value in AC1 from
                               // decimal to real and store the
                               // real number in VD500.
```

Network 2: *Store the Maximum and Minumum Scaling Values*

```
LD    Always_On:SM0.0          // Load SM0.0.
MOVD  VD500, VD1000            // Move the value in VD500 into
                               // VD1000.
MOVD  VD200, VD1004            // Move the value in VD200 into
                               // VD1004.
MOVD  VD204, VD1008            // Move the value in VD204 into
                               // VD1008.
MOVW  VW208, VW1012            // Move the value in VW208 into
                               // VW1012.
MOVD  VD212, VD1016            // Move the value in VD212 into
                               // VD1016.
CALL  SBR_0:SBR0               // Call SBR0.
```

Network 3: *Truncate the Value Received from Subroutine SBR0*

```
LD    Always_On:SM0.0          // Load SM0.0.
TRUNC VD2000, AC1              // Truncate the value in VD2000
                               // and store the result in AC1.
MOVW  AC1, AQW0                // Move the value in AC1 to
                               // AQW0.
```

Network 4: *Store the TD 200 Maximum and Minimum Scale Values*

```
LD    Always_On:SM0.0          // Load SM0.0.
MOVD  VD216, VD1012            // Move the value in VD216 into
                               // VD1012.
MOVD  VD220, VD1016            // Move the value in VD220 into
                               // VD1016.
CALL  SBR_0:SBR0               // Call SBR0.
```

Network 5: *Enable the TD 200 Message to Display the Liquid Level Value*

```
LD    Always_On:SM0.0          // Load SM0.0.
MOVR  VD2000, AC1              // Move the value in VD2000 into
                               // accumulator AC1.
*R    100.0, AC1               // Multiply the value in AC1
                               // by 100.
TRUNC AC1, AC1                 // Truncate value in AC1.
MOVW  AC1, VW116               // Move the value in AC1 into
```

```
                                    // VW116 for TD 200 display.
=     V12.7                         // Enable 'Liquid Level = ' message
                                    // on the TD 200.
```

Network 6: *Enable Max. Level Reached Message on the TD 200*

```
LDR>=  VD2000, VD224                // If VD2000 >= VD224,
=     V12.6                         // enable 'Max. Level Reached'
                                    // message on the TD 200.
```

Network 7: Enable Min. Level Reached Message on the TD 200

```
LDR<=  VD2000, VD228                // If VD2000 <= VD228,
=     V12.5                         // enable 'Min. Level Reached'
                                    // message on the TD 200.
```

Network 8: Open the Inlet Valve

```
LDN    V12.6                        // Load variable memory bit V12.6
                                    // as a Normally Closed contact.
                                    // If V12.6 is not set
A      I0.0                         // and input I0.0 is set
AN     Q0.1                         // and output Q0.1 is not set,
=      Q0.0                         // set output Q0.0.
```

Network 9: Open the Outlet Valve

```
LDN    V12.5                        // Load variable memory bit V12.5
                                    // as a Normally Closed contact.
                                    // If V12.5 is not set
A      I0.1                         // and input I0.1 is set
AN     Q0.0                         // and output Q0.0 is not set,
=      Q0.1                         // set output Q0.1.
```

Network 10: *End of Main Program*

Subroutine Program (SBR0):

Network 1: *Subroutine SBR0*

Network 2: *Subtract Minimum Scale Values from Maximum Scale Values*

```
LD     Always_On:SM0.0              // Load SM0.0.
MOVR   VD1012, AC1                  // Move the value in VD1012 into
                                    // AC1.
-R     VD1016, AC1                  // Subtract the value in VD1016
                                    // from the value in AC1.
MOVR   VD1004, AC2                  // Move the value in VD1004 into
                                    // AC2.
-R     VD1008, AC2                  // Subtract the value in VD1008
                                    // from the value in AC2.
MOVR   VD1000, AC3                  // Move the value in VD1000 into
                                    // AC3.
-R     VD1008, AC3                  // Subtract the value in VD1008
```

```
                                    // from the value in AC3.
Network 3: Perform Final Mathematical Calculations
LD    Always_On:SM0.0          // Load SM0.0.
/R    AC2, AC3                  // Divide the value in AC3 by
                                // the value in AC2.
*R    AC1, AC3                  // Multiply the value in AC1
                                // by the value in AC3.
MOVR  AC3, VD2000               // Add the value in AC3 to
                                // the value in VD1016
+R    VD1016, VD2000            // and place the sum in VD2000.
```

Network 4: *End of Subroutine SBR0*

6.5. Module điều khiển vị trí một trục:

Yêu cầu phần cứng:

- Một CPU-221 or 222 or 224 or 224XP or 226 or 226XM
- Bởi vì đầu ra xung được sử dụng trong phần gợi ý này nên CPU loại DC/DC/DC được lựa chọn.
- Một cáp PC/PPI.
- Một bộ lập trình (PG) hoặc máy tính (PC).
- Một motor bước loại SIMOSTEP với độ tăng trưởng moment là 2Nm, độ tăng trưởng dòng điện là 1.8A. Để tra thông số kỹ thuật của các loại động cơ bước này dựa vào trang web sau: ht//www.ad.siemens.de/.
- Một module FM STEPDRIVE.
- Một đoạn cáp cho motor khoảng chừng 10m.
- Điện trở hoặc CALEX module 8502.
- Một bộ cáp cho tín hiệu điều khiển tới nguồn nuôi.
- Một bộ mô phỏng cho S7 200.

CPU 221 sử dụng trong ví dụ này sử dụng hai đầu ra phát xung tốc độ cao để điều khiển motor (có thể phát tới tần số 20kHz), nên dùng chức năng ramp up hoặc ramp down của các đời CPU 221 trở lên. Sử dụng bộ nguồn đặc biệt FM STEPDRIVE để chuyển đổi xung điều khiển thành nguồn dòng để cung cấp cho các cuộn dây của motor. Từ trường quay của motor có thể chuyển đổi sang vị trí, cụ thể là số bước tương ứng với góc đo α.A do xung điều khiển tạo ra một cách tuần tự. Dãy xung tuần tự tương ứng với tần số của những bước giống nhau (xung đồng bộ). Nếu tần số không đủ cao thì sẽ xảy ra hiện tượng chuyển động step-to-step của trục động cơ sẽ chuyển thành chuyển động quay liên tục (điều này có thể gây ra mất bước).

Trong ví dụ mẫu này sử dụng đầu ra phát xung Q0.0 cho motor; I0.0 tín hiệu điều khiển motor; việc điều khiển đọc ra số xung vuông được ấn định như là việc đọc số bước của motor; đầu vào I0.1 là công tắc off của motor; đầu vào I0.5 để lựa chọn hướng quay của motor.

Để giảm thiểu lỗi trong quá trình điều khiển ở tần số cao, nên sử dụng đặc tính ramp lúc tăng hoặc giảm tốc điều này sẽ hiệu quả hơn rất nhiều. Đặc tính ramp này sẽ được giới thiệu ở phần sử dụng hai hàm phát xung tốc độ cao PTO và PWM.

+ **Module FM STEPDRIVE:**

Module này có thể điều khiển bằng tín hiệu clock ở mức cao đó là ưu điểm nổi bật. Mỗi một xung clock tương ứng với một bước của motor. Người ta có thể ấn định giá trị của dòng pha, số bước, độ suy giảm dòng bằng các lựa chọn trên các công tắc của module.

+ **Input Signals:**

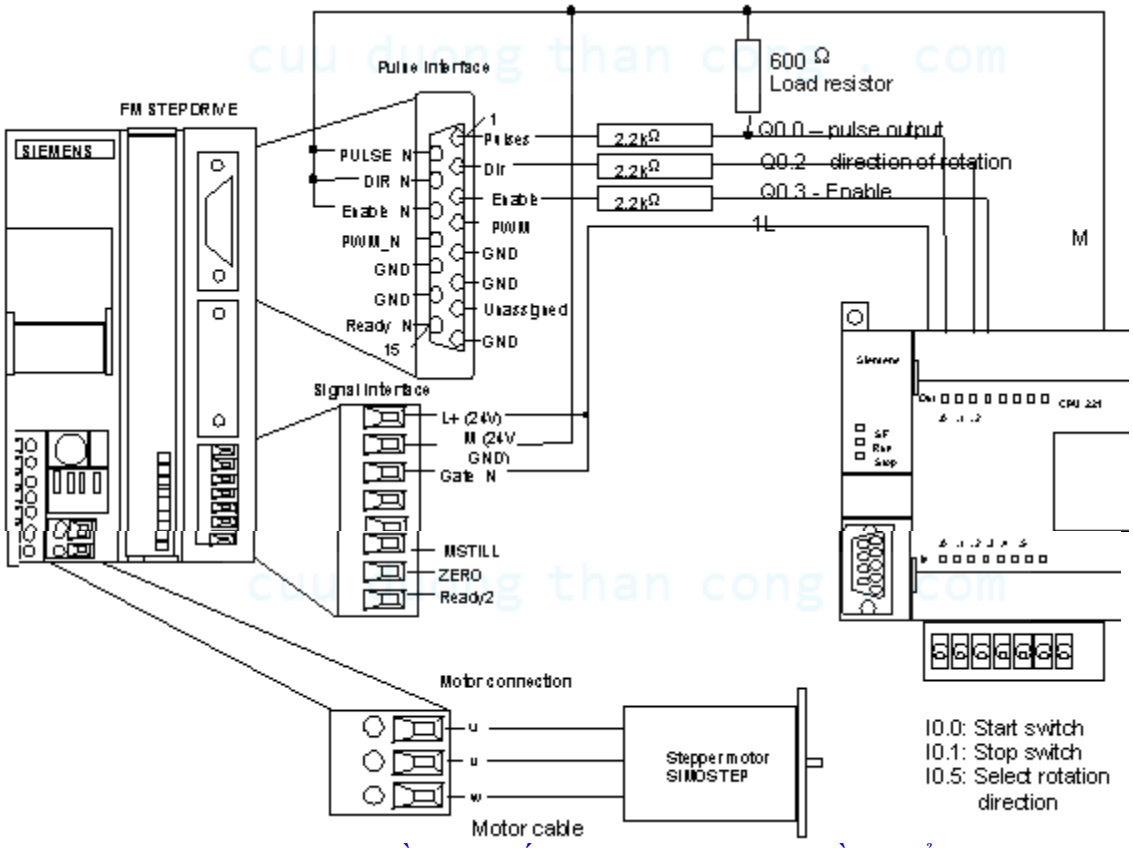
PULSE	Clock pulse	Mỗi sườn lên tạo nên một bước, điện áp 24VDC
DIR	Drection of rotation	Chọn chiều quay thuận ngược
ENABLE	Enable	Nếu có tín hiệu vào là cho phép thì bộ phận nguồn sẽ sẵn sàng cung cấp
PWM	Current Control	Mức dòng pha của motor được set lên, nó có thể thay đổi được bằng cách điều biến độ rộng xung.

+ **Out Signals:**

READY1_N Ready Status Sau khi đầu vào enable cho phép hoạt động, bộ phận nguồn sẽ có báo cáo sẵn sàng hoạt động cho đầu ra READY1_N.

+ **Tín hiệu giao tiếp:**

Tín hiệu của bộ điều khiển ở mức cao được cung cấp bằng xung điều khiển ở đầu vào 24VDC, có thể cho phép điều khiển motor ở đầu vào GATE_N.



Hình 6.8: Sơ đồ ghép nối step motor với bộ điều khiển

+ **Inputs:**

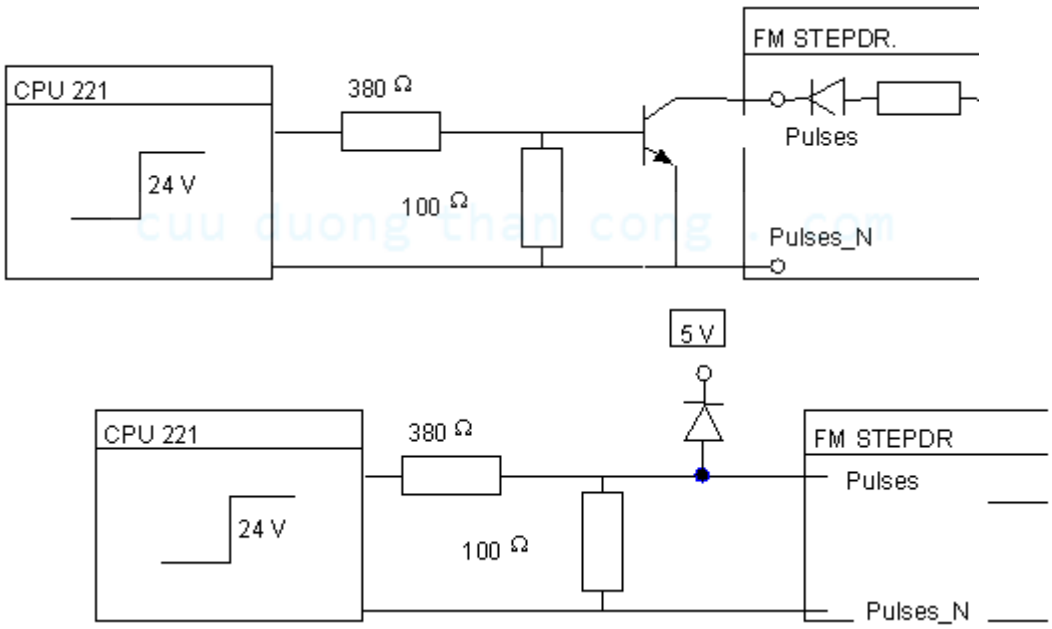
GATE_N Enable the clock pulse signal : Khi có 24V ở cổng vào GATE_N, tín hiệu đó đang chờ đợi cho việc điều khiển stepper motor. Nếu cho 0V, tín hiệu chờ đó bị huỷ bỏ.

+ **Outputs:**

ZERO	Zero signal right counter	Vị trí zero của bộ đếm vòng quay bên trong, điện áp 24V được cấp cho đầu ra zero này.
READY2	Ready status	Sau khi đầu vào cho phép hoạt động, bộ phận nguồn báo cáo là đọc được bởi đầu ra ready2.
MSTILL	Motor Stepped	Nếu tín hiệu clock bị huỷ bởi đầu vào GATE_N và motor dừng lại, sự dừng lại này là sự chấp nhận bởi tín hiệu MSTILL.

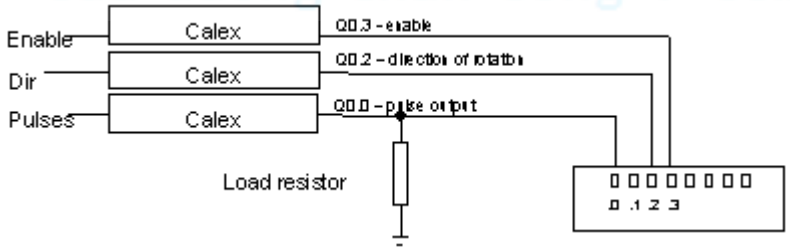
+ Bộ chuyển đổi điện áp cho bộ điều khiển Stepper Motor:

Xem hình sau đây bạn có thể dễ dàng tạo một mạch điện để kết nối bộ điều khiển lập trình tới bộ drive của stepper motor. Tất cả các đầu vào của bộ FM STEPDRIVE là 5V.



Hình 6.9: Sơ đồ ghép nối giữa đầu ra của PLC với module FM STEPDRIVE

Cũng có thể lựa chọn bộ chuyển đổi Callex (như là module 8502) để tạo ra nguồn tín hiệu 5V.



Hình 6.10: Sơ đồ ghép nối giữa đầu ra của PLC với Calex

Mô tả chương trình:

Trong vòng quét đầu tiên (SM0.1=1) các giá trị quan trọng cho việc tạo xung được đặt lại. Ở đây tốc độ hoạt động cũng như số bước theo danh nghĩa lý thuyết.

Cách lựa chọn hướng quay của motor:

Bạn có thể sử dụng công tắc I0.5 để lựa chọn hướng quay. Nếu đầu vào I0.5 = 1 đầu ra Q0.2 được set lên mức cao và chiều quay của động cơ lúc này là ngược chiều kim đồng hồ. Nếu đầu vào I0.5 = 0, đầu ra Q0.2 được reset xuống mức thấp và chiều quay của motor lúc này là cùng chiều kim đồng hồ. Trong trường hợp để tránh motor mất bước, hướng quay chỉ có thể thay đổi được khi bit nhớ trạng thái hoạt động của motor là off (M0.1 = 0).

Các bước khởi động motor:

1. Ấn nút Start, điều đó có nghĩa là có sườn lên truyền tới đầu vào I0.0.
2. Không có khoá liên động, điều đó có nghĩa là bit nhớ liên động đã bị reset.
3. Bộ điều khiển chuyển sang chế độ off, có nghĩa là bit nhớ hoạt động đã bị reset.

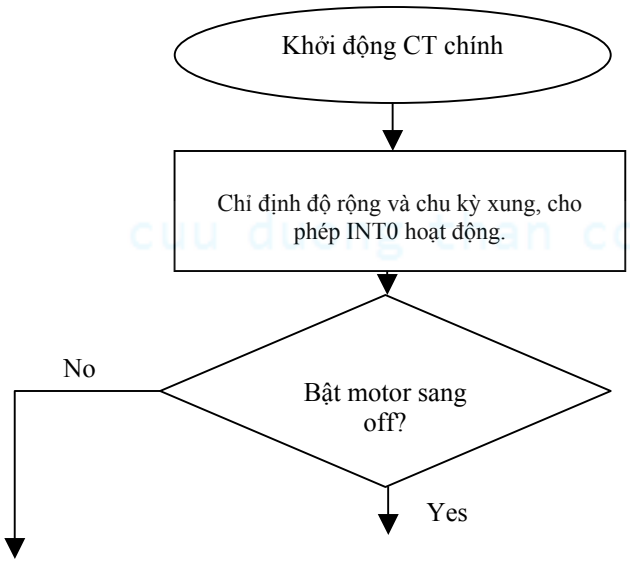
Nếu các yếu tố đã được hội tụ, bit nhớ M0.1 được reset và bộ điều khiển sử dụng lệnh PLS để khởi động việc phát ra dãy xung ở cổng Q0.0. Điều cần thiết cho việc phát xung là phải có dữ liệu được khai báo tương ứng trong vùng nhớ đặc biệt tương ứng với lệnh PTO/PWM và đầu ra Q0.3 được set.

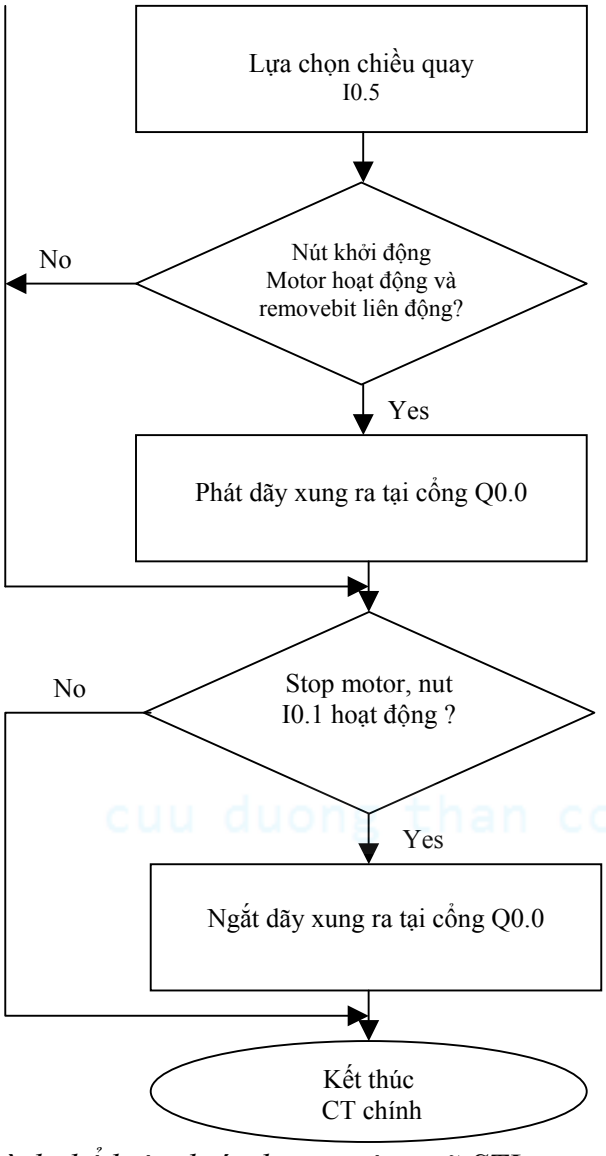
Cách dừng motor:

1. Ấn nút Stop, điều này tương ứng với việc truyền xung lên đến port I0.1
2. Bộ điều khiển bật lên on, điều đó tương ứng với M0.1 được set.

Nếu các yếu tố đã được hội tụ, bit nhớ M0.1 được reset. Sau đó xung ra tại port Q0.0 bị ngắt đi bởi vì quá trình điều biến độ rộng xung đã bị giải phóng kết nối với lệnh PLS0. Khi điều này xảy ra, độ rộng xung bị giảm xuống zero. Sau đó ngắt 0 được xử lý, bit nhớ M0.1 được reset lần nữa để chuẩn bị cho việc khởi động bộ điều khiển lần tiếp theo.

Cấu trúc chương trình điều khiển:





Chương trình thể hiện dưới dạng ngôn ngữ STL:

Network 1: ***MAIN PROGRAM*** Specify Pulse Width and Cycle Time

```
 LD    First_Scan_On          // Load SM0.1.
MOVW   +500, PLS0_Cycle       // Specify cycle time of 500
                              // microseconds for PWM.
MOVW   +0, PWM0_PW            // Specify pulse width of 0.
MOVD   +40000, PTO0_PC        // Read out 40,000 pulses.
S      Enable_Drive, 1        // Enable the FM STEPDRIVE.
ATCH   INT_0, 19              // Attach interrupt event 19 to
                              // INT0.
ENI                           // Enable interrupt.
```

Network 2: Enable Counterclockwise Rotation

```
LDN    Drive_ON               // Load M0.1 as a Normally Closed
                              // contact.
```

```
                                // If M0.1 is not set
A     Direction                 // and input I0.5 is set,
S     Q0.2, 1                   // set output Q0.2.
```
Network 3: Enable Clockwise Rotation
```
LDN   Drive_ON                  // Load M0.1 as a Normally Closed
                                // contact.
                                // If M0.1 is not set
AN    Direction                 // and input I1.5 is not set,
R     Q0.2, 1                   // reset output Q0.2.
```
Network 4: Activate Interlock
```
LD    Motor_STOP                // Load input I0.1.
                                // If input I0.1 is set,
S     Interlock, 1              // set memory bit M0.2.
```
Network 5: Cancel Interlock
```
LDN   Motor_START               // Load input I0.0 as a Normally
                                // Closed contact.
                                // If input I0.0 is not set
AN    Motor_STOP                // and input I0.1 is not set,
R     Interlock, 1              // reset memory bit M0.2.
```
Network 6: Set PWM/PTO Control for Output Q0.0 and Start Drive
```
LD    Motor_START               // Load input I0.0
EU                              // If there is a positive transition
                                // (Edge Up) at input I0.0
AN    Interlock                 // and memory bit M0.2 is not set
AN    Drive_ON                  // and memory bit M0.1 is not set,
MOVB  16#85, PLS0_Ctrl          // load the control bits for pulse
                                // train output at output Q0.0.
PLS   0                         // Enable pulse function at output
                                // Q0.0.
S     Drive_ON, 1               // Set memory bit M0.1.
```
Network 7: Stop Drive and Set PWM/PTO Control for Output Q0.0
```
LD    Motor_STOP                // Load input I1.1
EU                              // If there is a positive transition
                                // (Edge Up) at input I0.1
A     Drive_ON                  // and memory bit M0.1 is set,
R     Drive_ON, 1               // reset memory bit M0.1.
MOVB  16#CB, PLS0_Ctrl          // Load control bits for pulse width
                                // modulation at output Q0.0.
PLS   0                         // End pulse output at Q0.0.
```
Network 8: Kết thúc chương trình chính.

Network 1: Bắt đầu chương trình con (Interrupt Routine INT0)

Network 2: Reset Memory Bit M0.1 (drive ON)
```
LD    Always_On                 // Load SM0.0.
```

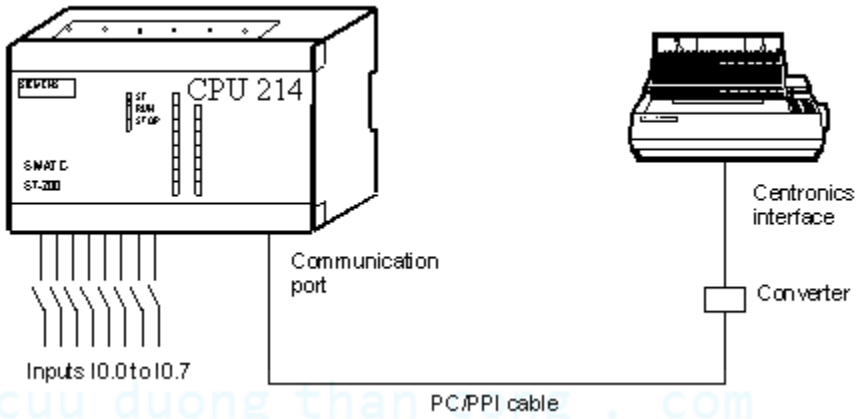
R Drive_ON, 1 // Reset memory bit M0.1.

Network 3: End of Interrupt Routine INT0.

RETI // End INT0.

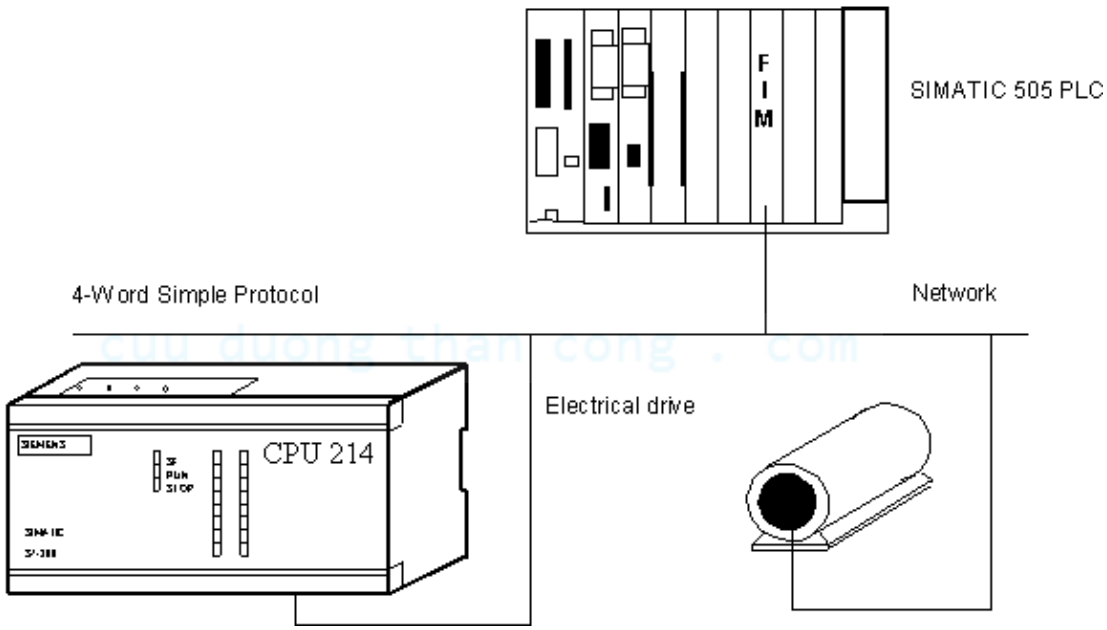
6.6. Các ứng dụng truyền thông trên Step 7-200:

6.6.1. Kết nối PLC với máy in qua cổng song trong chế độ truyền thông Freeport:



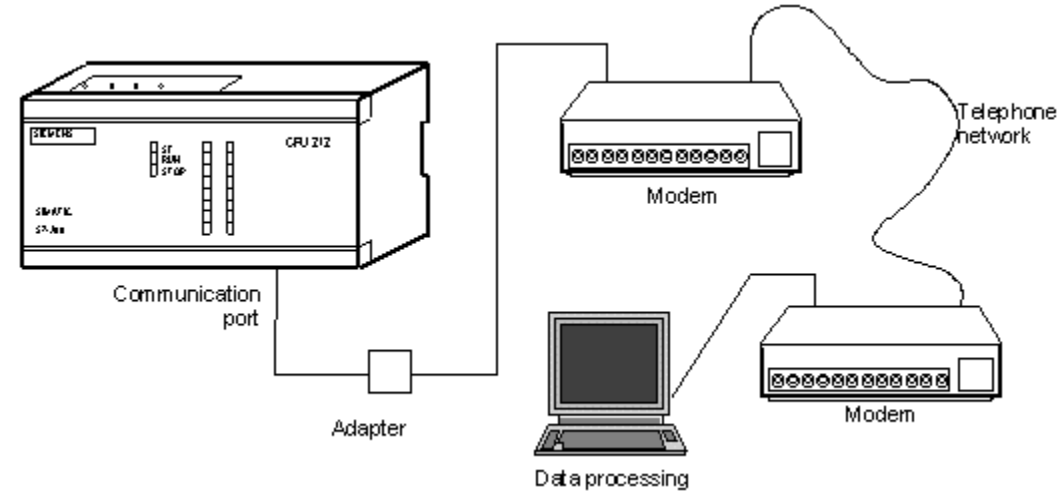
Hình 6.11: Kết nối PLC với máy in qua cổng song

6.6.2. Truyền thông giữa S505 và S7 trong mạng qua module giao diện trường MIF:

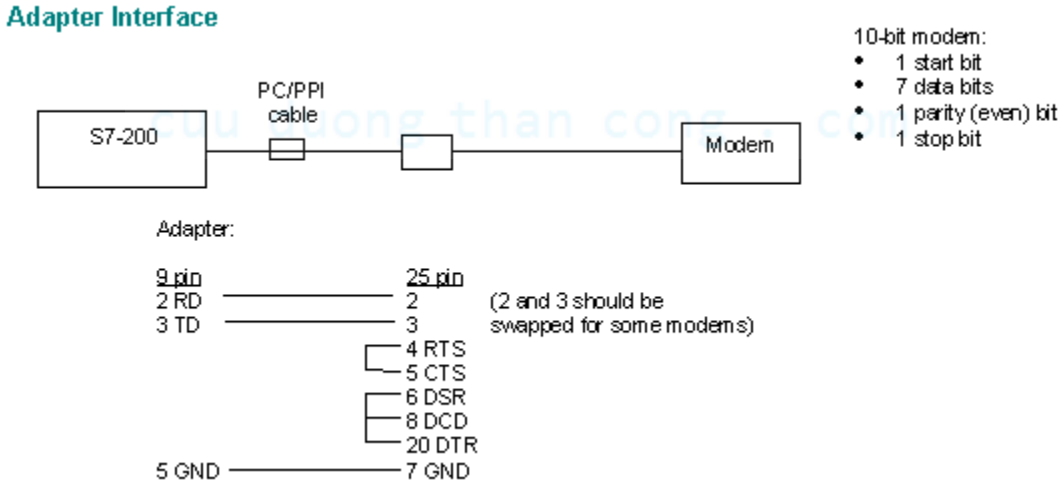


Hình 6.12: Kết nối S505 và S7 trong mạng qua module giao diện trường MIF

6.6.3. Truyền thông S7-200 ở chế độ Freeport sử dụng modem điện thoại telephone network:

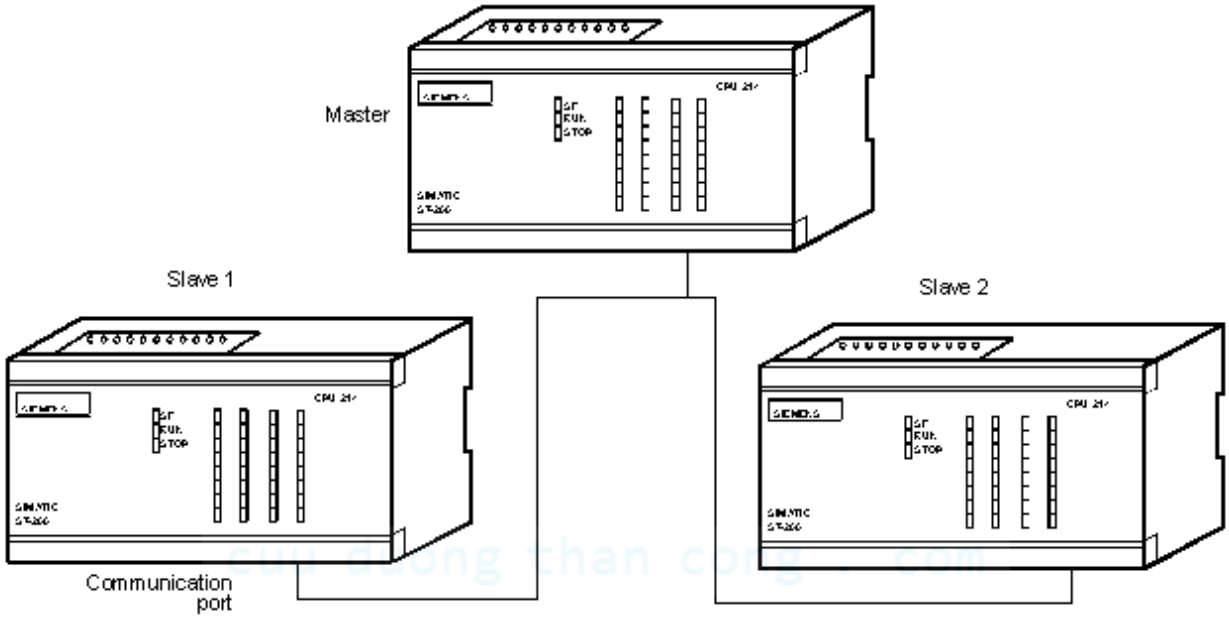


Hình 6.12: Kết nối S7-200 ở chế độ Freeport sử dụng modem điện thoại



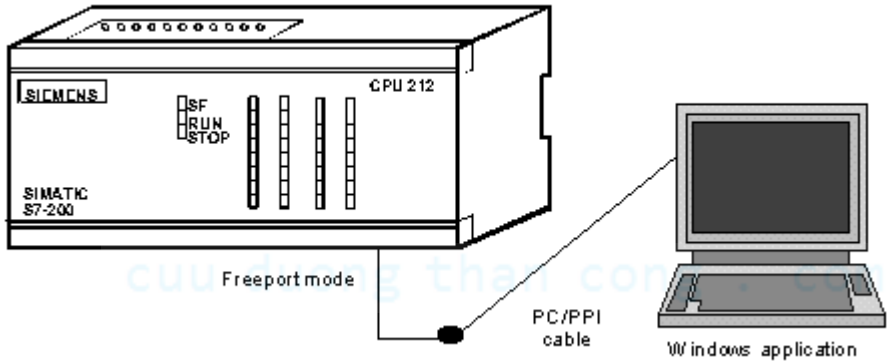
Hình 6.13: Kết nối port truyền thông giữa S7 và modem

6.6.4. Truyền thông Freeport để kết nối mạng vài S7-200 CPUs trong trường hợp I/O ở xa:



Hình 6.14: Truyền thông giữa các S7-200 với nhau

6.6.5. Sử dụng trình ứng dụng Hyper Terminal window kết nối giữa PC và PLC:

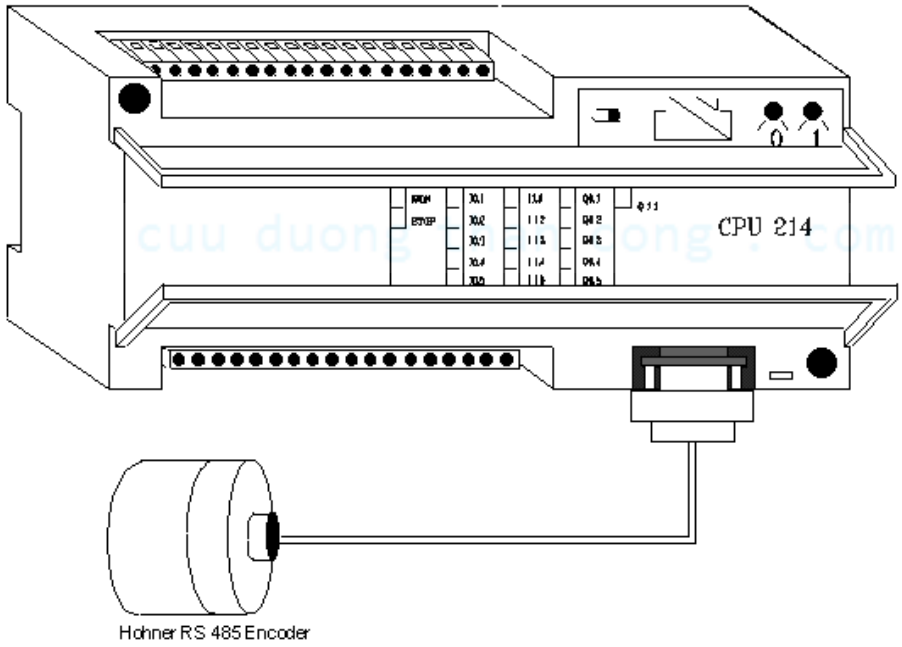


Hình 6.15: Truyền thông giữa S7 và PC

6.6.6. Kết nối giữa S7-200 với encoder sử dụng port truyền thông RS485:

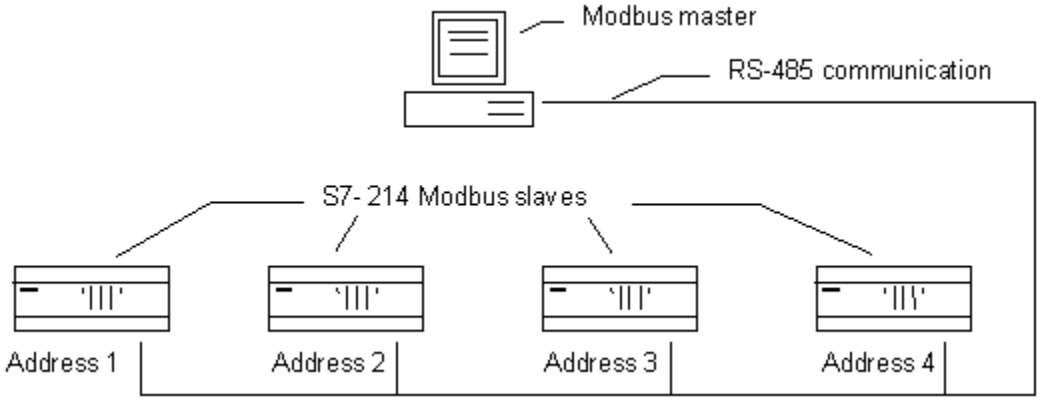
Cable Assignment:

S7-200 9-Pin Male Connector		Hohner RS 485 Encoder 12-Pin Male Circular Connector	
Pin 1	Gnd	Pin 1	Gnd
Pin 2	24V-	Pin 2	11 - 24V
Pin 3	T/R+	Pin 3	T/R+
Pin 4	Reserved	Pin 4	T/R-
Pin 5	Gnd	Pin 5	Ident 2E0
Pin 6	5V	Pin 6	Ident 2E1
Pin 7	24V+	Pin 7	Ident 2E2
Pin 8	T/R-	Pin 8-12	Free
Pin 9	Reserved		



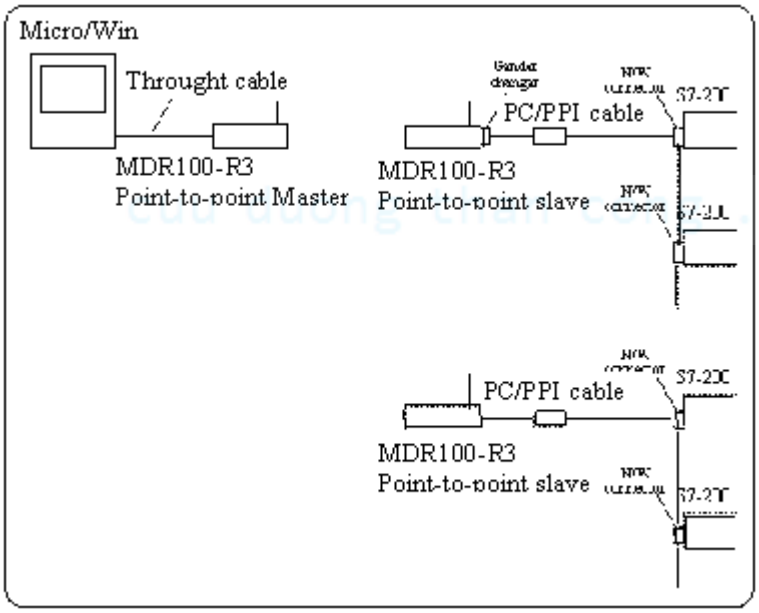
Hình 6.16: Kết nối Encoder vào port truyền thông

6.6.7. Truyền thông theo giao thức Mudbus để kết nối các S7-200 slave:



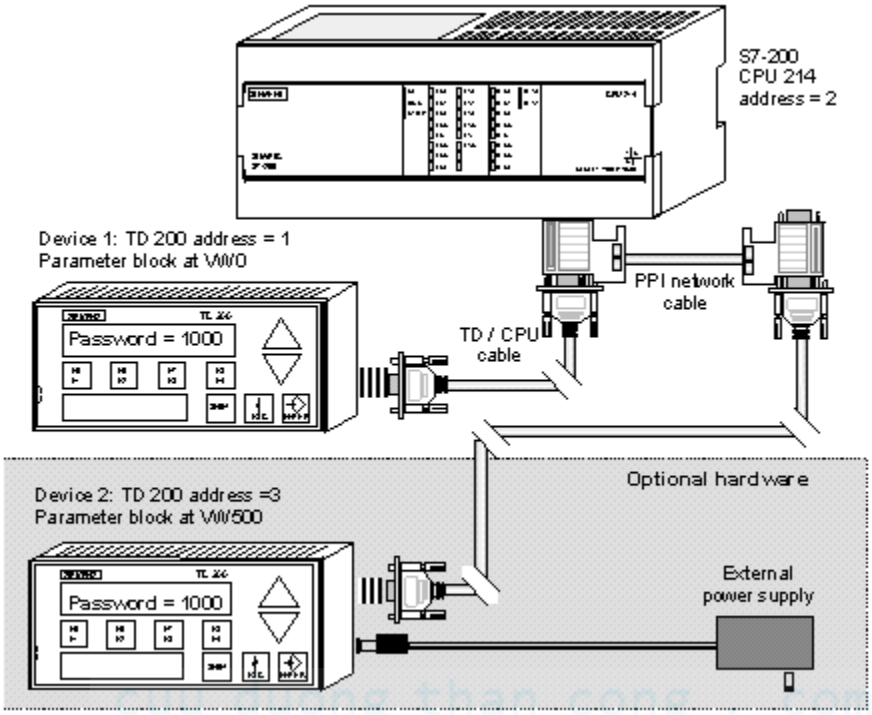
Hình 6.17: Kết nối truyền thông theo giao thức Mudbus

6.6.8. Sử dụng modem Radio để kết nối mạng S7-200:



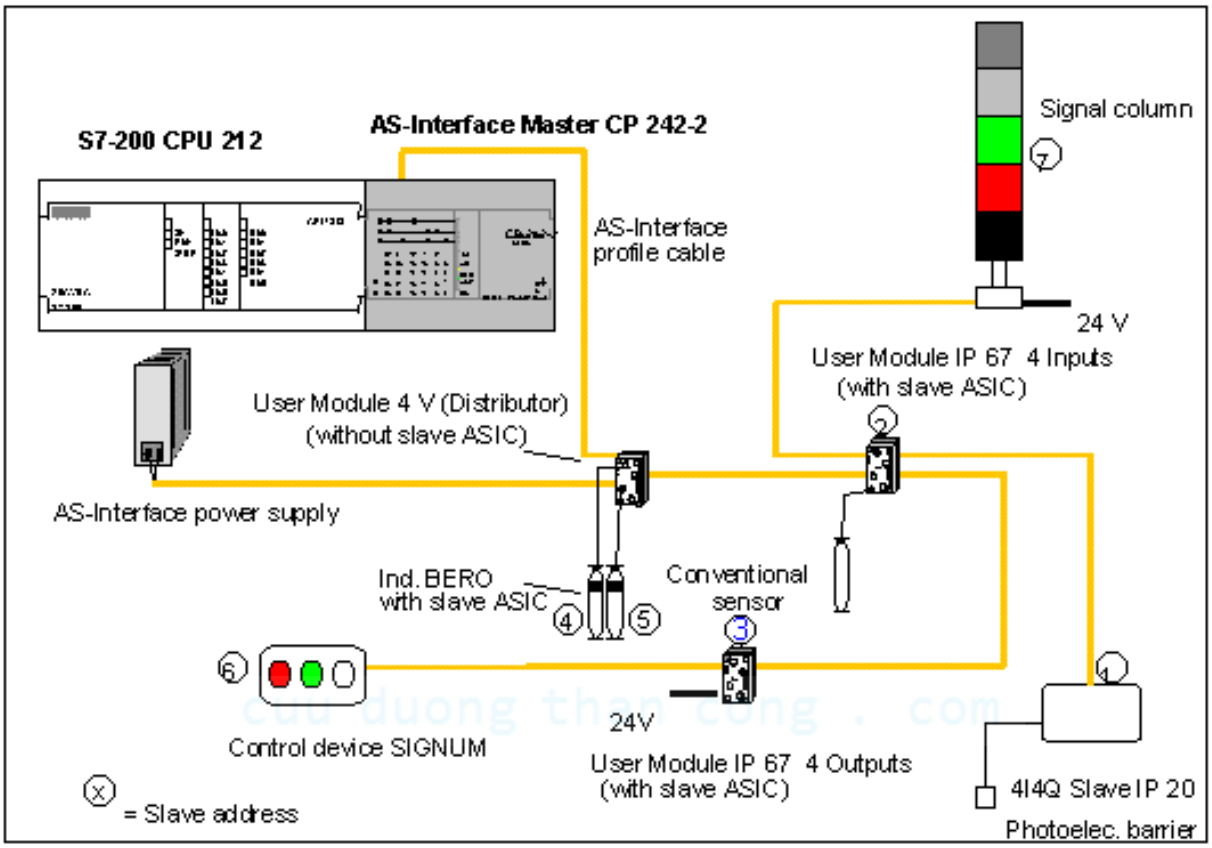
Hình 6.18: Kết nối truyền thông sử dụng Radio modem

6.6.9. Sử dụng TD-200 để điều khiển và giám sát S7-200:



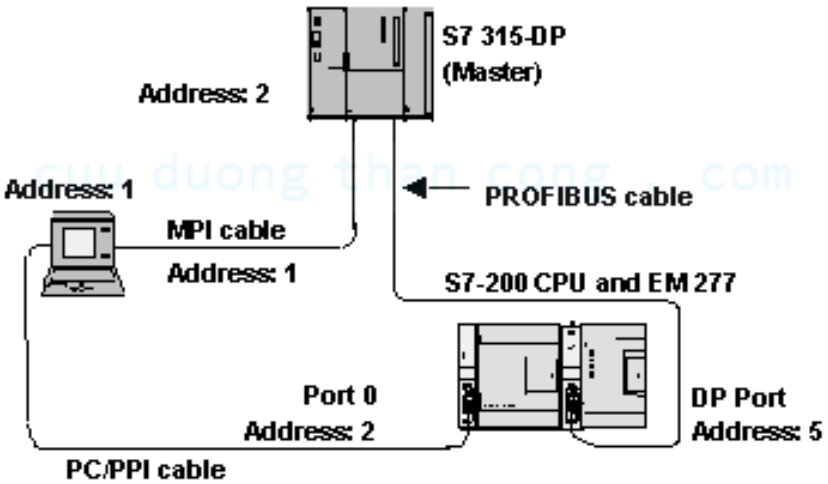
Hình 6.19: Kết nối truyền thông S7-200 và TD200

6.6.10. Tích hợp mạng AS-I với S7-200:



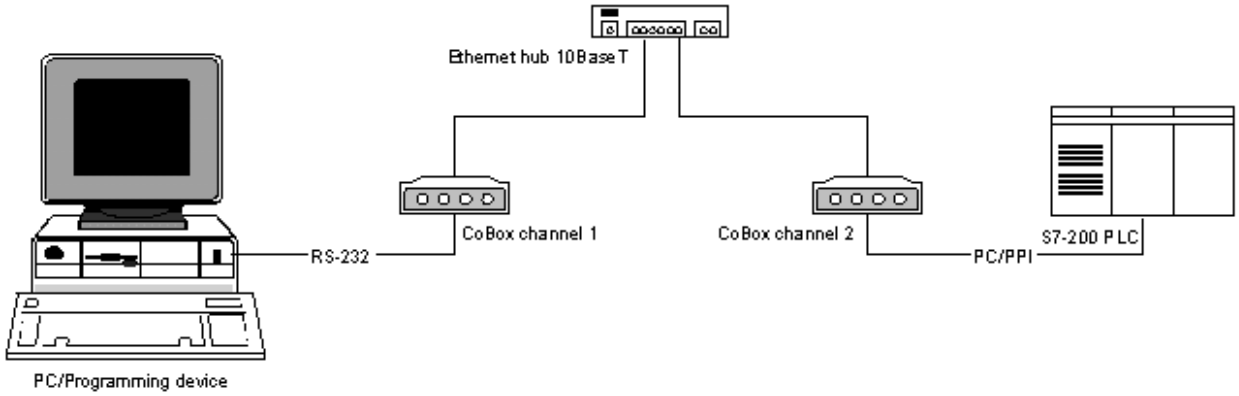
Hình 6.20: Kết nối truyền thông S7-200 với sensor và cư cấu chấp hành qua mạng AS-I

6.6.11. Kết nối S7-300 với S7-200 theo chuẩn Profibus và với máy lập trình:



Hình 6.21: Truyền thông S7-200 và S7-300 theo chuẩn Profibus

6.6.12. Kết nối S7-200 với mạng Ethernet:



Hình 6.22: Truyền thông S7-200 trong mạng Ethernet

External CoBox, Ethernet

(Front view)

Network interface:
- - RJ45 (10BaseT) and Attachment Unit Interface (AUI) Ethernet

Firmware: version 3.6

(Rear view)

Serial interface:
- - DB25 RS-232/RS-422/RS-485 (channel 1)
- - DB9 RS-232 (channel 2)

Note: Only the RS232 port was tested on this unit.

Standard (DIN) Rail CoBox

Network interface:
- - 10BaseT (RJ45) connection

Serial interface:
- - RS-232/RS-422/RS-485 by means of terminal block or serial interface with RJ45 port

Note: The RS-485 2-wire port did not function as of Feb. 2000. The RS232 port worked correctly.

Hình 6.23: Các modem kết nối S7-200 với mạng Ethernet