Curso S1601002

Introducción á programación de autómatas

1 INTRODUCCIÓN 7

2 ELEMENTOS DUN AUTÓMATA PROGRAMABLE 7

2.1 UNIDADE CENTRAL DE PROCESO 8

2.2 MEMORIA 8

2.2.1 Memoria de só lectura (ROM) 8

2.2.2 Memoria de acceso aleatorio (RAM) 8

2.2.3 Memoria de só lectura borrable (EPROM) 8

2.2.4 Memoria de só lectura borrable electricamente (EEPROM) 8

2.2.5 Memoria FLASH 8

2.3 INTERFACES DE ENTRADA/SAÍDA 8

3 CICLO DE SCAN 9

4 SENSORES INDUSTRIAIS 10

4.1 CLASIFICACIÓN DE SENSORES SEGUNDO O PRINCIPIO DE FUNCIONAMENTO 10

4.1.1 Sensores activos 10

4.1.2 Sensores pasivos 11

4.2 CLASIFICACIÓN DOS SENSORES SEGUNDO O TIPO DE SINAL ELÉCTRICA QUE XENERAN 12

4.2.1 Sensores dixitais 12

4.2.2 Sensores analóxicos 13

4.3 CLASIFICACIÓN DOS SENSORES SEGUNDO A VARIABLE FÍSICA A MEDIR 13

4.4 SIMBOLOXÍA NORMALIZADA 14

4.4.1 Sensores de proximidade de dous fíos 16

4.4.2 Sensores de proximidade a tres fíos 17

4.4.3 Sensores de proximidade a catro fíos 19

5 INTERFACE DE CONEXIÓN CO PROCESO 19

5.1 INTERFACES DE APLICACIÓN XERAL 20

5.1.1 Interfaces de variables todo-nada 20

5.1.2 Interfaces de variables analóxicas 23

5.1.3 Interface de variables de entrada todo-nada sen illamento galvánico 24

5.1.4 Interface de variables de entrada todo-nada con illamento galvánico e alimentación en continua 24

5.1.5 Interface de variables de entrada todo-nada con illamento galvánico e alimentación en alterna 26

5.1.6 Interfaces de variables analóxicas de entrada 27

5.2 INTERFACES DE ENTRADA DE APLICACIÓN ESPECÍFICA 28

5.2.1 Módulos de medida de temperatura para PT100 28

5.2.2 Módulos de medida de temperatura para termopar 29

5.3 INTERFACES DE SAÍDA TODO-NADA 30

5.3.1 Interfaces de variables de saída todo-nada con relé 30

5.3.2 Interfaces de variables de saída todo-nada con transistor NPN 31

5.3.3 Interfaces de variables de saída todo-nada con transistor PNP *31*
5.3.4 Interfaces de variables de saída todo-nada con tiristor ou triac *32*
5.4 INTERFACES DE VARIABLES ANALÓXICAS DE SAÍDA 32
5.5 EXEMPLOS DE CONEXIONADO DE PULSADORES, INTERRUPTORES OU SENSORES DIXITAIS NAS ENTRADAS DUN AUTÓMATA. 33
5.5.1 Contacto simple empregando a fonte de alimentación própia *33*
5.5.2 Contacto simple empregando unha fonte de alimentación externa *34*
5.5.3 Conexionado dun transductor con saída a tres fíos NPN, alimentado pola própia fonte do autómata *34*
5.5.4 Conexionado dun transductor con saída a tres fíos PNP, alimentado pola própia fonte do autómata *35*
5.5.5 Conexionado dun encoder incremental con saídas A,B e Z (tipo NPN) alimentado pola propia fonte do autómata *35*
5.6 EXEMPLOS DE CONEXIONADO DE SAÍDAS DIXITAIS NUN AUTÓMATA PROGRAMABLE 36
5.6.1 Saídas a relé *36*
5.6.2 Saídas a transistor *36*
6 SISTEMA STEP7 DE PROGRAMACIÓN DE AUTÓMATAS. 37
6.1 CARACTERÍSTICAS XERAIS DE STEP7 37
6.1.1 Linguaxes literais *37*
6.1.2 Linguaxes gráficas *38*
6.2 TIPOS DE DATOS 38
7 INTRODUCCIÓN Á PROGRAMACIÓN DE AUTÓMATAS S7-1200 42
7.1 CONFIGURACIÓN DUN AUTÓMATA CON S7-1200 TOOL 42
7.2 CONFIGURACIÓN DUN PLC CON TIA PORTAL 44
7.3 CARACTERÍSTICAS DUN PROGRAMA 51
7.3.1 Linguaxes de programación en TIA Portal *52*
7.3.2 Acceso a datos nun PLC S7 1200 ..... *53*
7.4 ASPECTOS A TER EN CONTA Á HORA DE EMPREGAR O SIMULADOR DE PROCESOS VIRTUALMAKTCP 54
7.4.1 TIA Portal v12 ou anteriores e firmwares anteriores á versión 4 ..... *54*
7.4.2 TIA Portal v13 e firmware 4.0 ou posterior *54*
7.5 PRIMEIRO PROGRAMA 55
7.6 VISUALIZANDO VARIABLES 57
7.7 INSTRUCCIÓNS LÓXICAS CON BITS 58
7.7.1 Contactos e bobinas en instrucciónslóxicas con bits *58*
7.7.2 Operacións lóxicas *59*
7.7.2.1 Exercicio 1 63
7.7.3 Instrucciónslóxicas AND, OR e OR exclusiva en FUP e SCL *66*
7.7.4 Instrucción de negación NOT *67*
7.7.5 Instruccións "Activar saída" e "Desactivar saída" *67*
7.7.6 Báscula de activación/desactivación e de desactivación/activación *68*
7.7.6.1 Exercicio 2 70
7.7.7 Consulta de flancos ascendentes e descendentes *76*

7.7.7.1 Exercicio 3 77
7.8 TEMPORIZADORES 78
7.8.1.1 Exercicio 4 81
7.8.1.2 Exercicio 5 81
7.8.1.3 Exercicio 6 83
7.9 CONTADORES 86
7.9.1.1 Exercicio 7 88
7.9.1.2 Exercicio 8 91
7.9.1.3 Exercicio 9 93
7.10 INSTRUCCIÓNS DE COMPARACIÓN 96
7.10.1 Comparación *96*
7.10.2 Instruccións "Valor dentro do rango" e "Valor fóra do rango" *96*
7.11 FUNCIÓNS MATEMÁTICAS 97
7.11.1 Instrucción calcular *97*
7.11.2 Instruccións "sumar", "restar", "multiplicar" e "dividir" *98*
7.11.3 Instrucción "módulo" *98*
7.11.4 Instruccións "incrementar" e "decrementar" *99*
7.11.5 Instrucción "valor absoluto" *99*
7.11.6 Instruccións "mínimo" e "máximo" *99*
7.12 DESPRAZAMENTO 100
7.12.1 Instruccións "copiar valor" e "copiar bloque" *100*
7.12.2 Instrucción "cambiar orden" *101*
7.13 CONVERSIÓN 101
7.13.1 Instrucción CONV *101*
7.13.2 Instruccións "redondear" e "truncar" *102*
7.13.3 Instruccións CEIL e FLOOR *102*
7.13.4 Instruccións "escalar" e "normalizar" *103*
7.13.4.1 Exercicio 10 103
8 PROGRAMACIÓN DE SISTEMAS SECUENCIAIS MEDIANTE GRAFCET 106
8.1 ETAPAS 107
8.2 TRANSICIÓNS 108
8.3 TIPOS DE GRAFCET 108
8.3.1 GRAFCET de secuencia única *108*
8.3.2 GRAFCET de secuencias opcionais *109*
8.3.3 GRAFCET de secuencias simultáneas *109*
8.3.3.1 Exercicio 11 110
8.4 PROGRAMACIÓN DUN GRAFCET EN LINGUAXE DE CONTACTOS 112
8.4.1 Organización do GRAFCET *113*
8.4.1.1 Zona preliminar 113
8.4.1.2 Zona secuencial 113
8.4.1.3 Zona de accións 114
8.4.1.4 Exercicio 12 115

8.4.1.5 Exercicio 13 ... 124
8.4.1.6 Exercicio 14 ... 133
8.4.1.7 Exercicio 15 ... 137
8.4.1.8 Exercicio 16 ... 139

9 PARAMETRIZACIÓN DO PLC ... 142

9.1.1 Cambio de direccións de entradas/saídas ... 142
9.1.2 Parametrización das marcas de ciclo ... 143
9.1.3 Configuracións de módulos analóxicos ... 143
9.1.4 Configuración da saída de impulsos ... 144
9.1.5 Contadores rápidos ... 146
Exemplo ... 146

10 PROGRAMACIÓN DE SISTEMAS CONTÍNUOS. VARIABLES ANALÓXICAS ... 150

10.1.1.1 Exercicio 17 ... 152
10.1.1.2 Exercicio 18 ... 153
10.1.1.3 Exercicio 19 ... 155
10.1.1.4 Exercicio 20 ... 155
10.2 ESCALADO E DESESCALADO DE MAGNITUDES ANALÓXICAS ... 156
10.2.1.1 Exercicio 21 ... 157
10.2.1.2 Exercicio 22 ... 158
10.2.1.3 Exercicio 23 ... 159
10.2.1.4 Exercicio 24 ... 161

11 PROGRAMACIÓN ESTRUCTURADA ... 161

11.1.1.1 Exercicio 25 ... 162
11.1.1.2 Exercicio 26 ... 165

12 INTRODUCCIÓN Á PROGRAMACIÓN SCL ... 169

12.1 INSTRUCCIÓN IF-THEN ... 169
12.2 INSTRUCCIÓN CASE ... 169
12.3 INSTRUCCIÓN FOR ... 170
12.4 INSTRUCCIÓN WHILE ... 170
12.5 INSTRUCCIÓN REPEAT-UNTIL ... 171
12.5.1.1 Exercicio 27 ... 171
12.5.1.2 Exercicio 28 ... 174

13 INTRODUCCIÓN Á COMUNICACIÓN ENTRE AUTÓMATAS ... 175

13.1 COMUNICACIÓN ENTRE DOUS AUTÓMATAS ... 175
13.1.1.1 Exercicio 29 ... 175
13.1.1.2 Exercicio 30 ... 180

14 A NORMA IEC 61131-3 ... 183

14.1 LINGUAXES DA NORMA ... 183
14.1.1 Linguaxes literais ... 184

14.1.2 Linguaxes gráficos ... 184
14.2 UNIDADES DE ORGANIZACIÓN DE PROGRAMA (POU) ... 184
14.2.1 Programas ... 185
14.2.2 Funcións ... 185
14.2.3 Bloques funcionais ... 186
15 PROGRAMACIÓN DE PLCS CON CODESYS. SIMULACIÓN CON CONTROLWIN ... 186
15.1 PRIMEIROS PASOS ... 187
15.2 CONFIGURACIÓN DA COMUNICACIÓN ... 191
15.3 PRIMEIRO PROGRAMA ... 193
15.4 SIMULANDO O PROGRAMA ... 198
16 TRABALLO COS AUTÓMATAS DE SCHNEIDER ... 203
16.1 MÉTODOS DE ACCESO ... 203
16.1.1 Posibles problemas na conexión. ... 205
16.2 ENTORNO DE PROGRAMACIÓN ... 206
16.2.1 Configuración de hardware ... 208
16.3 CREANDO UN PROGRAMA EN GRAFCET ... 213
16.4 EXECUTANDO O PROGRAMA ... 220

1 Introducción

Nos anos 60 e 70 a industria comezou a decatarse da necesidade de incrementar os niveis de calidade e productividade, así como a necesesidade de realizar cambios con relativa frecuencia nos procesos productivos.

Naqueles intres, as liñas de producción incorporaban un número importante de elementos electromecánicos que ocupaban paneis de dimensións considerables, de costoso conexionado e pouco flexibles á hora de facer cambios.

Falábase dun sistema *Ladder* no que se interconectaban interruptores, sensores, motores, válvulas, relés, entre outros.

Foi o fabricante de automóbiles General Motors o primeiro que recoñeceu a necesidade de cambiar os sistemas cableados, e, aproveitando a auxe dos microprocesadores (daquela de 8 bits), pensaron en deseñar un dispositivo que substituíra en todo o posible aos sistemas electromecánicos. Así naceu o autómata programable (PLC).

2 Elementos dun autómata programable

A arquitectura dos PLCs é de tipo Harvard. Esto quere dicir que se dispón dunha memoria para almacenamento de programa e outra distinta para almacenamento de datos, a diferencia con outros dispositivos microprocesadores como os PCs, nos que se emprega a mesma memoria tanto para o programa como para os datos (von-Neumann).

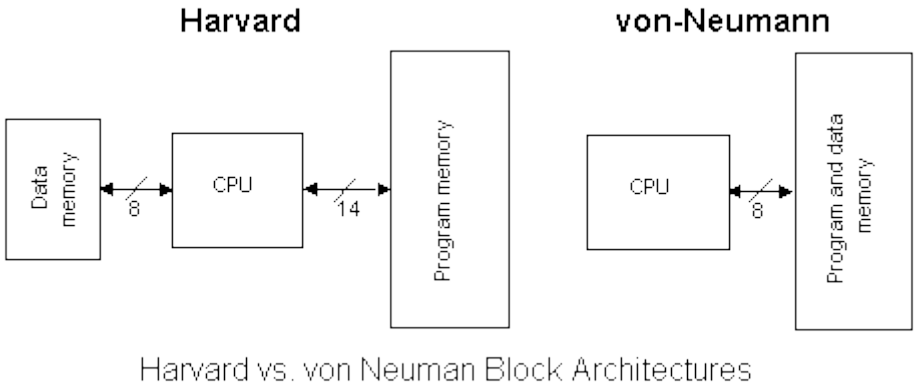


Fig: 1 Arquitecturas de sistemas microprocesador

Os elementos básicos dun autómata son:

2.1 Unidade central de proceso

Microprocesador que coordina as actividades do sistema PLC. Executa o programa, procesa os sinais de entrada/saída e comunícase cos dispositivos externos.

2.2 Memoria

Hai varios tipos de memoria nos PLCs. En todo caso aquí e onde se almacena o sistema operativo do autómata, o programa a executar e os datos que manexa o mesmo. Podemos falar dos seguines tipos de memoria:

2.2.1 Memoria de só lectura (ROM)

Memoria non-volátil, que só se pode programar unha vez.

2.2.2 Memoria de acceso aleatorio (RAM)

Comunmente empregada para almacenar os datos do programa. Unha vez cortada a alimentación, os datos almacenados neste tipo de memoria pérdense.

2.2.3 Memoria de só lectura borrable (EPROM)

Similar en comportamento á memoria ROM, pero os datos poden borrarse empregando luz ultravioleta.

2.2.4 Memoria de só lectura borrable electricamente (EEPROM)

Compórtase como unha ROM en canto ao almacenamento dos datos, xa que non se perden ao cortarlle a alimentación, pero pode borrarse e sobreescribirse por medios eléctricos un determinado número de veces.

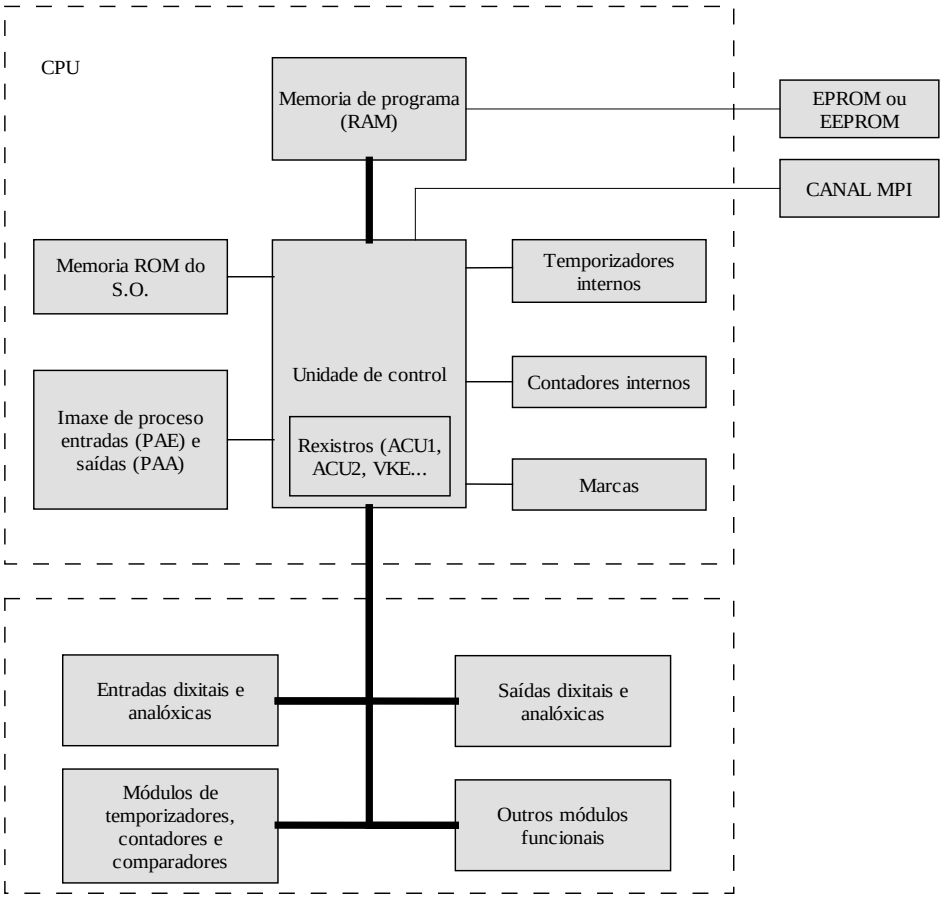
2.2.5 Memoria FLASH

Derivada da memoria EEPROM, pero con acceso de lectura e escritura moito máis rápido que as anteriores.

2.3 Interfaces de entrada/saída

Módulos electrónicos destinados a conectar o microprocesador cos elementos de entrada/saída do proceso. Existe na actualidade unha amplia gama de interfaces.

Na imaxe seguinte vemos un diagrama de bloques dun autómata programable da familia S7 de Siemens.



3 Ciclo de scan

O proceso de ler as entradas, executar o programa e actualizar as saídas denomínase **scan**. O tempo de **scan** é o tempo que tarda o PLC en levar a cabo as tarefas mencionadas. Este tempo é variable, e depende de varios factores. Sobre todo depende da lonxitude do programa, de como estea estructurado o mesmo, e das características das entradas/saídas do PLC (se son entradas/saídas locais ou remotas).

De forma predeterminada, todas as E/S dixitais e analóxicas locais actualízanse de xeito sincrónico co ciclo, utilizando unha área de memoria interna denominada memoria **imaxe de proceso**. A memoria imaxe de proceso contén unha instantánea das entradas e saídas físicas.

A CPU do autómata executa as seguintes tarefas:

Primeiro escribe as saídas desde a memoria imaxe de proceso das saídas nas saídas físicas.

A continuación lee as entradas físicas inmediatamente antes de executar o programa de usuario e almacena os valores de entrada na memoria imaxe de proceso das entradas. Así garántese que estes valores sexan coherentes durante a execución das instruccións programadas.

Por último executa a lóxica das instruccións programadas e actualiza os valores de saída na memoria imaxe de proceso das saídas, en vez de escribilos nas saídas físicas reais.

4 Sensores industriais

Para que un sistema electrónico poida controlar un proceso ou producto industrial é necesario que reciba información da evolución de determinadas variables físicas, como poden ser: temperatura, presión, nivel de líquido ou de sólido, forza, radiación luminosa, posición, velocidade, aceleración, desprazamento, entre outros.

Hai que dispoñer dos elementos que convirtan estas magnitudes físicas en sináis eléctricas. Estes dispositivos reciben varios nomes como **captador**, **detector**, **transductor**, **transmisor**, **sonda** ou **sensor**.

Esta útltima denominación (sensor) é a máis empregada polos fabricantes de sistemas de control.

Podemos definir **sensor** como o dispositivo que convirte unha variable física non eléctrica en outra eléctrica, que nalgún dos seus parámetros (tensión, corrente, frecuencia entre outros) contén información correspondente á primeira.

4.1 Clasificación de sensores segundo o principio de funcionamento

Podemos clasificalos en **activos** e **pasivos**.

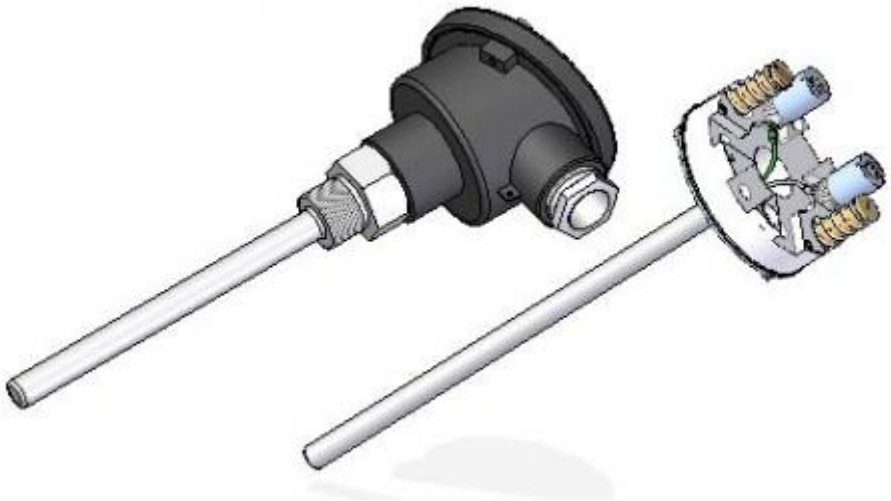
4.1.1 Sensores activos

Son aqueles nos que a magnitude física a medir proporciona a enerxía necesaria para a xeneración do sinal eléctrico de saída.

Como exemplos temos:

- Piezoeléctricos
- Fotoeléctricos ou optoeléctricos
    - Fotoemisivos

    - Fotovoltaicos
- Termoeléctricos (termopares)
- Magnetoeléctricos
    - Electromecánicos
    - Semiconductores
- Outros



4.1.2 Sensores pasivos

Son aqueles nos que a magnitude física a medir limítase a modificar algún dos parámetros eléctricos característicos, como resistencia ou capacidade, entre outros.

Estes sensores caracterízanse por necesitar unha tensión de alimentación externa.

Como exemplos temos:

- Resistivos (Resistencia variable)
    - Potenciómetros
    - Termorresistivos
    - Fotorresistivos
    - Extensiométricos
    - Magnetorresistivos

    - Electroquímicos
- Capacitivos (Capacidade variable)
- Inductivos (Inductancia variable)
    - Reluctancia variable
    - Magnetoestrictivos
    - Transformador variable
- Outros

4.2 Clasificación dos sensores segundo o tipo de sinal eléctrica que xeneran

4.2.1 Sensores dixitais

Xeneran sinais que só toman un número finito de valores. O caso máis característico son os binarios, que poden tomar dous valores (0 ou 1). Unha variable binaria recibe o nome de **bit**.

O criterio de asignación de valores aos estados é arbitrario, de xeito que, se asignamos o valor 1 ao nivel de tensión alto e 0 ao nivel de tensión baixo, estamos a falar de **lóxica positiva**. En caso contrario falariamos de **lóxica negativa**.

4.2.2 Sensores analóxicos

Este tipo de sensores xeneran un sinal analóxico, que pode tomar calquera valor dentro dun determinado rango.

Neste caso podemos falar de sinais **unipolares**, cando só poden ser positivas ou negativas con respecto a unha liña de referencia; ou **bipolares** cando poden ser tanto positivas como negativas con respecto á liña de referencia.

4.3 Clasificación dos sensores segundo a variable física a medir

As principais variables físicas que é ncesario medir en productos ou procesos industriais son:

- Presión
- Temperatura
- Humidade
- Forza

- Aceleración
- Velocidade
- Caudal
- Presencia e/ou posición de obxectos
- Nivel de sólidos ou líquidos
- Desprazamento de obxectos
- Químicos

Na seguinte táboa temos un resumo dos sensores a empregar segundo a aplicación.

		Variable física medida										
		Posición	Desprazamento	Velocidade	Aceleración	Tamaño	Nivel	Presión	Forza	Proximidade	Temperatura	Radiación luminosa
Principio de funcionamento	Microrruptores	X										
	Finais de carreira	X				X						
	Extensiométricos	X	X	X	X			X	X			
	Termorresistivos										X	
	Magnetorresistivos	X	X	X								
	Capacitivos	X	X		X		X	X	X	X		
	Inductivos	X	X	X	X			X	X	X		
	Optoelectrónicos	X	X	X						X		
	Piezoeléctricos		X	X	X			X	X			
	Fotovoltáicos											X
	Ultrasónicos	X					X					

4.4 Simboloxía normalizada

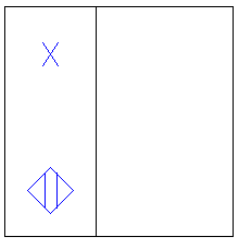
Os sensores de proximidade sin contacto comercialízanse con terminais ou con cables e por eso desenvolvéronse símbolos normalizados, incluídos na norma UNE EN 60947-5-2,

nos que, además do símbolo propiamente dito, establécese o tipo de sensor, a numeración dos terminais e a cor dos cables conectados aos distintos terminais.

Na táboa seguinte indícanse as cores que se empregan para identificar os cables conectados aos distintos terminais e a súa traducción ao inglés.

Cor	Abreviatura
Negro (Black)	BK
Marrón (Brown)	BN
Vermello (Red)	RD
Amarelo (Yellow)	YE
Verde (Green)	GN
Azul (Blue)	BU
Gris (Grey)	GY
Branco (White)	WH
Dourado (Gold)	GD
Verde/Amarelo (Green/Yellow)	GNYE

Na figura seguinte represéntase o símbolo normalizado básico, que consta de dúas partes. Na parte inferior esquerda colócase un rombo que indica que se trata dun sensor de proximidade e sobre el indícase, mediante unha letra X, o tipo de sensor do que se trata de acordo coa táboa seguinte.



Letra	Tipo de sensor
T	Emisor e receptor. Barrreira fotoélectrica (Sensor optoelectrónico de barreira de luz)
R	Fotoeléctrico réflex (Sensor optoelectrónico con reflector)
D	Emisor-receptor e obxecto. Sensor fotoeléctrico de detección directa. (Sensor optoelectrónico de reflexión directa ou de reflexión en obxecto)
I	Sensor inductivo
C	Sensor capacitivo
U	Sensor ultrasónico

Son numerosos os fabricantes que non cumpren la norna na súa totalidade. Por exemplo, é habitual indicar os inductivos mediante o símbolo dunha bobina en lugar da letra I e os capacitivos mediante o símbolo dun condensador en lugar da letra C. Debido á súa complexidade, os sensores optoelectrónicos de proximidade posúen ademáis algúns tipos de sinais especiais, como por exemplo de selección do funcionamento en luz ou en oscuridade, de inhibición entre outros.

Na parte dereita do símbolo represéntase, con letras (cor do cable) ou con números (número do terminal) e gráficamente, o tipo de saída normalmente aberta (NO) ou normalmente pechada (NC). Igualmente indícase mediante letras e números se o sensor se alimenta en alterna ou en contínua e neste caso, se é ou non polarizado. Se o sensor se alimenta en alterna, a fase represéntase mediante a letra L e o neutro mediante a letra N. Si se alimenta en contínua o positivo represéntase por L+ e o negativo por L- . Se a alimentación se pode realizar indistintamente en alterna ou en contínua, a fase de alterna indícase mediante L1.

4.4.1 Sensores de proximidade de dous fíos

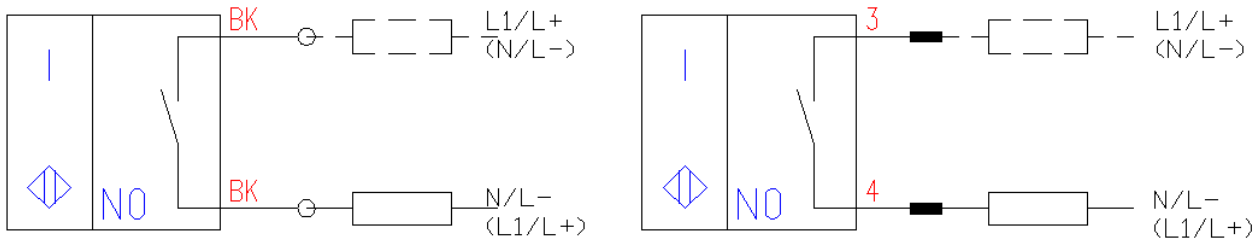
- Non polarizados

Pódense alimentar tanto con corrente contínua (DC) como con alterna (AC). Neste tipo de sensores as denominacións L1, N, L+ e L- colócanse, todas elas, nos dous terminais.

Se a saída do sensor é normalmente aberta (NO), aos terminais asígnanselle os números 3 e 4. Se a saída do sensor é normalmente pechada (NC), aos terminais asígnanselle os números 1 e 2.

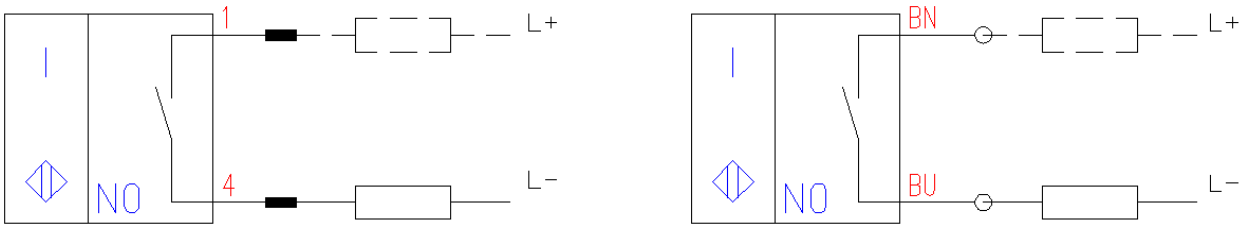
A cor dos cables pode ser calquera excepto o verde, o amarelo e o verde/amarelo, pero a norma aconsella que os dous sexan da mesma cor.

A figura seguinte amosa o símbolo normalizado dun sensor de proximidade inductivo de dous fíos non polarizado de saída equivalente a un contacto normalmente aberto, no que os terminais identifícanse con números. Na mesma figura indícase o mesmo tipo de sensor no que os dous fíos de conexión son negros (BK). Para indicar que a carga se pode conectar con calquera dos dous terminais, represéntase mediante dous rectángulos, un en trazo contínuo e o otro discontínuo.



- Polarizados

Aliméntanse en corrente contínua. O terminal ao que se conecta o positivo debe ser o cable marrón (BN) e o terminal ao que se conecta o negativo, o cable azul (BU). Ao primeiro asígnaselle sempre o número 1 e ao segundo o 2 ou o 4 segundo a saída sexa equivalente a un contacto normalmente pechado ou aberto, respectivamente. Na figura seguinte amósase o símbolo dun sensor de proximidade inductivo polarizado cunha saída equivalente a un contacto normalmente aberto, indicado mediante números e mediante letras. Ao igual que nos non polarizados, a carga represéntase mediante dous rectángulos. Un en trazo contínuo e o outro discontinuo, para indicar que pode estar conectada en cualquera dos dous terminais.

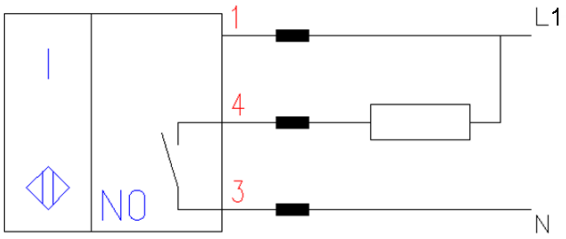


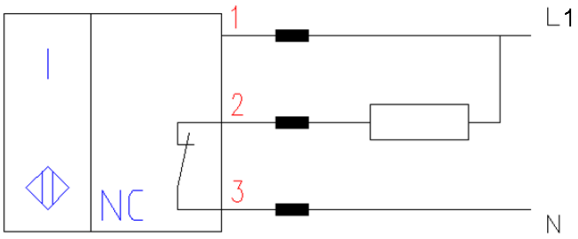
4.4.2 Sensores de proximidade a tres fíos

Poden ser polarizados ou non, e a carga externa conéctase entre dous dos tres fíos.

- Non polarizados

Tal como se indica na figura seguinte, a alimentación aplícase entre o terminal 1[cable marrón (BN)J e o 3 [cable azul (BU)]. A saída é sempre o fío negro (BK) e además se é NO asígnaselle ao terminal o número 4 e se é NC o número 2.

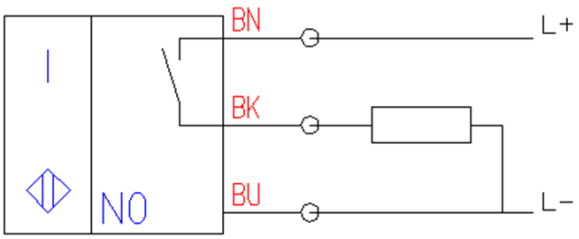


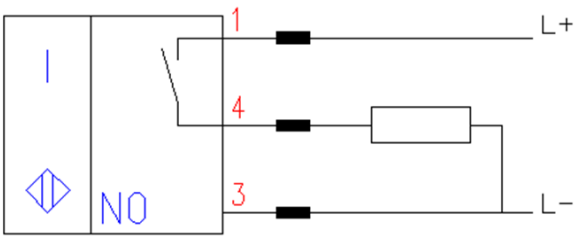


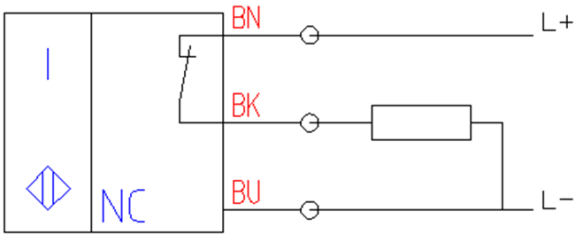
- <u>Polarizados</u>

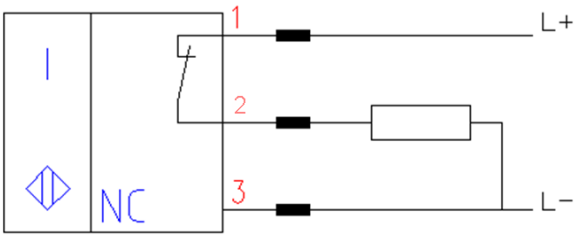
O positivo da alimentación aplícase ao terminal 1 (BN), e o negativo ao terminal 3 (BU). A saída asígnase ao terminal 4 ou 2 (BK).

Neste caso podemos atopar como transistores de saída un PNP ou un NPN. Nas figuras seguintes temos as diferentes situacións. Nas catro primeiras a variante con PNP.

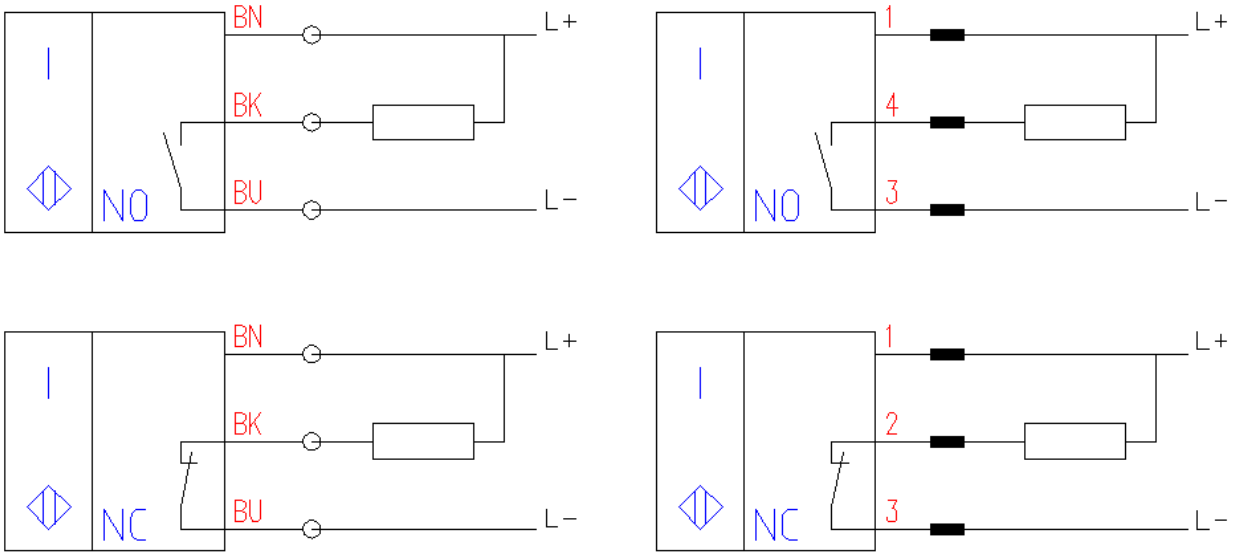






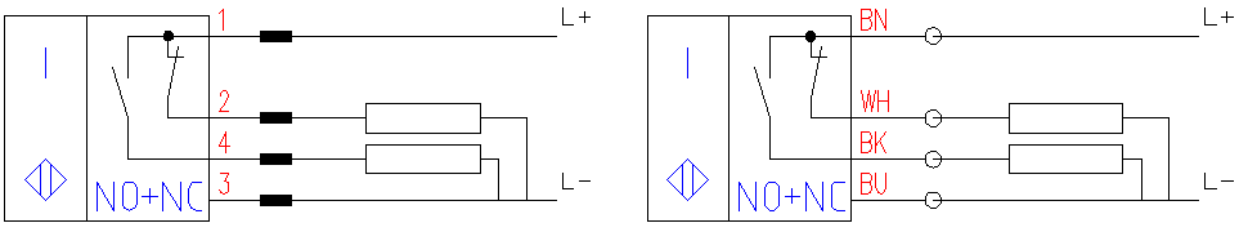


E a variante con NPN. Neste caso a carga conéctase entre a saída (fío negro (BK) ou terminal 4) e o positivo da alimentación, tal como vemos na figura seguinte.



4.4.3 Sensores de proximidade a catro fíos

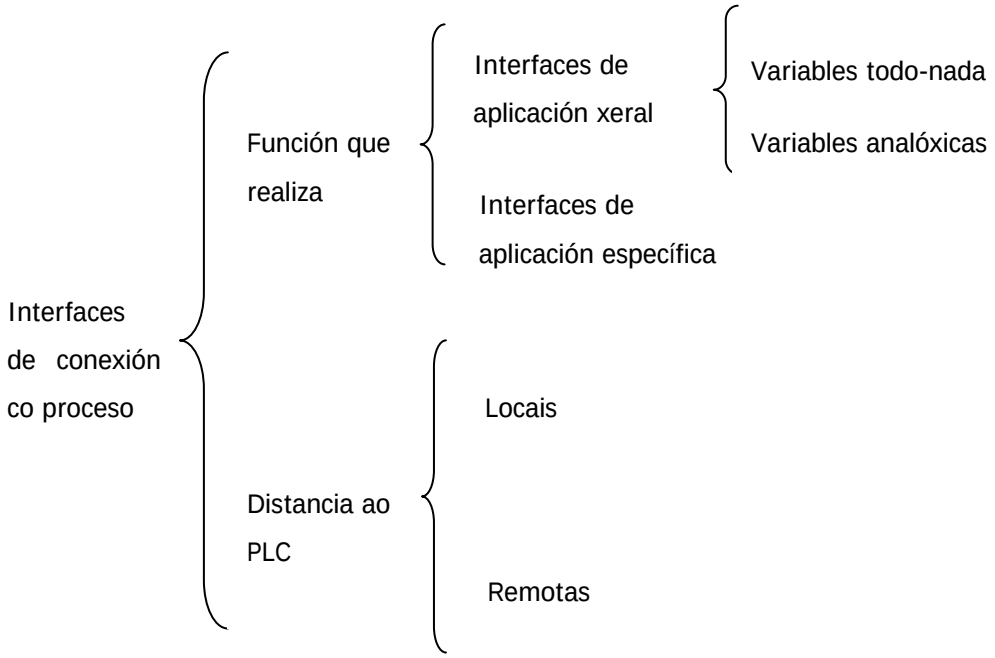
Levan incorporados un contacto normalmente aberto (NO) e un normalmente pechado (NC). O positivo da alimentación conéctase ao terminal 1 [cable marrón (BN)]. O negativo ao terminal 3 [cable azul (BU)]. A saída normalmente aberta (NO) vai ao terminal 4 [cable negro (BK)], e a normalmente pechada (NC) ao terminal 2 [cable branco (WH)]. Na figura seguinte vemos as dúas variantes.



5 Interface de conexión co proceso

O conxunto de circuítos electrónicos de acoplamento ou interfaces de entrada e de saída que emprega o PLC para relacionarse co proceso denomínase **interface de conexión co proceso**.

Podemos clasificar a interface de conexión co proceso como vemos na figura seguinte:



5.1 Interfaces de aplicación xeral

Teñen como misión acoplar ao PLC as variables do proceso, tanto dixitais (todo-nada) como analóxicas.

5.1.1 Interfaces de variables todo-nada

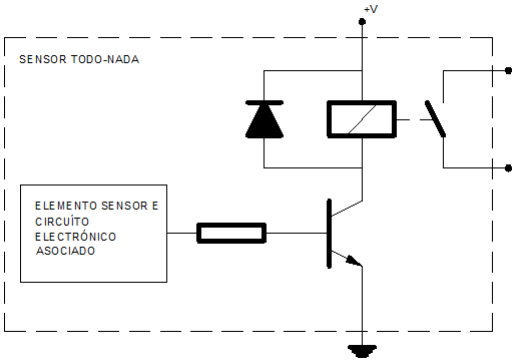
Son aquelas que só poden tomar dous valores.

Podemos clasificalas:

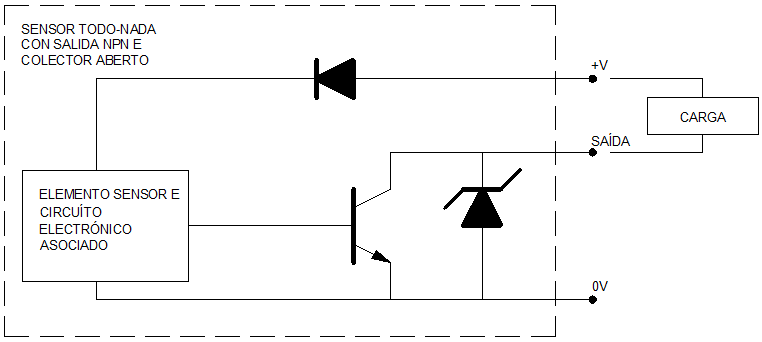
- Segundo o tipo de alimentación empregado (corrente contínua ou alterna).
- Segundo a maneira de realizar o acoplamento, que pode ser directo (sin illamanento galvánico) ou con illamento ou separación galvánica.

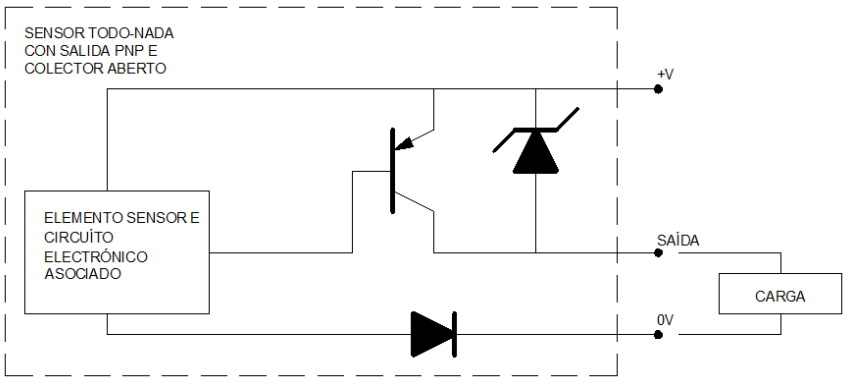
As variables de entrada todo-nada son xeradas por sensores todo-nada (como termostatos, presostatos, finais de carreira, pulsadores, interruptores, entre outros). Estes dispositivos actúan sobre:

- Un contacto libre de potencial.

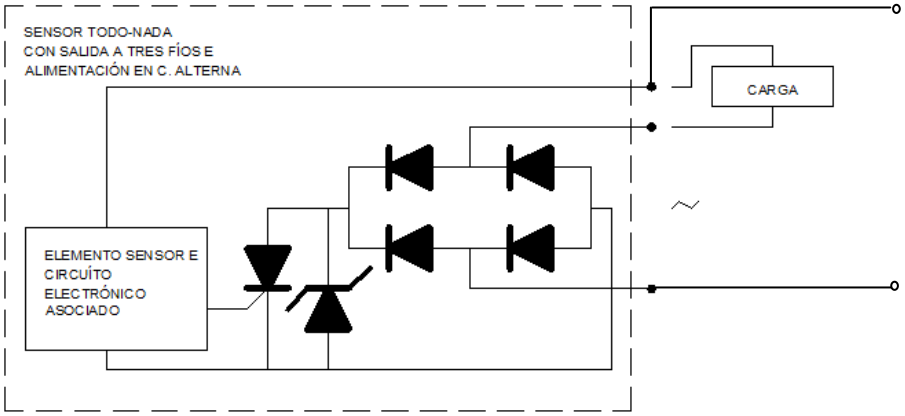


- Un transistor NPN ou PNP que bascula entre corte e saturación.





- Un tiristor ou triac que bascula entre corte e saturación.



A conexión dun sensor a un autómata programable a través dun circuíto de entrada pódese realizar de dúas maneiras diferentes:

- **Sen illamento galvánico**: Cando os dous dispositivos teñen polo menos dous puntos unidos eléctricamente, é dicir, están ao mesmo potencial eléctrico.
- **Con illamento galvánico**: Cando non existe conexión eléctrica entre o sensor e a entrada do PLC. Deste xeito, unha sobretensión ou unha sobrecorrente no circuíto de entrada non afecta ao PLC.

Na táboa seguinte temos unha clasificación das interfaces das variables de entrada todo-nada de acordo co tipo de alimentación e o tipo de saída do sensor que se lles pode conectar.

<table>
<tr><th colspan="3">INTERFACES DE ENTRADA TODO-NADA</th></tr>
<tr><td rowspan="4">Alimentación continua</td><td colspan="2">Sensores con saída tipo relé</td></tr>
<tr><td colspan="2">Sensores de dous fíos</td></tr>
<tr><td rowspan="2">Sensores de tres fíos</td><td>Saída transistor NPN</td></tr>
<tr><td>Saída transistor PNP</td></tr>
<tr><td rowspan="3">Alimentación alterna</td><td colspan="2">Sensores con saída tipo relé</td></tr>
<tr><td colspan="2">Sensores de dous fíos</td></tr>
<tr><td colspan="2">Sensores de tres fíos</td></tr>
</table>

As saídas todo-nada dos PLC están tamén deseñadas con illamento galvánico para evitar que as sobretensións ou sobrecorrentes que se poidan producir nos circuítos de saída poidan afectarlle.

Na táboa seguinte vemos unha clasificación dos tipos de saídas xunto co tipo de dispositivo que leva a cabo a conmutación do sinal.

INTERFACES DE SAÍDA TODO-NADA	
Alimentación en continua	Saída tipo relé
	Saída tipo transistor NPN
	Saída tipo transistor PNP
Alimentación en alterna	Saída tipo relé
	Saída tipo relé de estado sólido (tiristor ou triac)

5.1.2 Interfaces de variables analóxicas

Conectan o PLC con sensores que proporcionan múltiples valores dentro de un rango. Empréganse para tomar datos de magnitudes físicas como presión, temperatura, nivel, caudal, velocidade, posición entre outras.

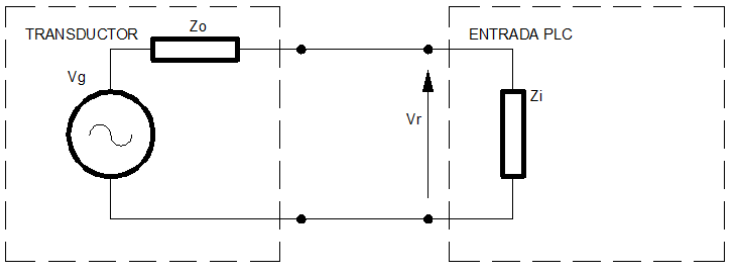
Poden sen de tensión ou de corrente.

- **Variables analóxicas de tensión**

O seu comportamento pódese representar mediante un circuíto como o da figura seguinte, no que o sensor é un xenerador de tensión ideal (Vg) en serie cunha impedancia de saída (Zo).

Se a impedancia da entrada analóxica do PLC (Zi) é moito maior que a de saída do sensor, entón a tensión de entrada coincide prácticamente coa do sensor.

$$Vr = \frac{Zi}{Zo + Zi} \cdot Vg$$

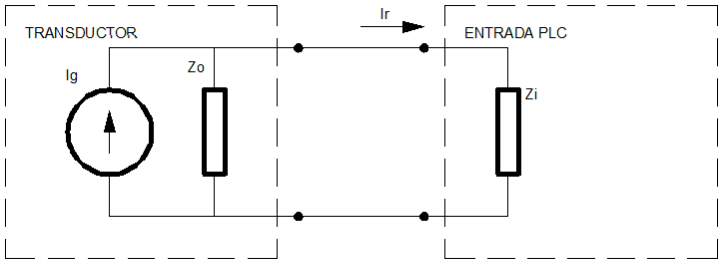


- **Variables analóxicas de corrente**

O seu comportamento pódese representar mediante un circuíto como o da figura seguinte, no que o sensor é un xenerador de corrente ideal (Ig) en paralelo cunha impedancia de saída (Zo).

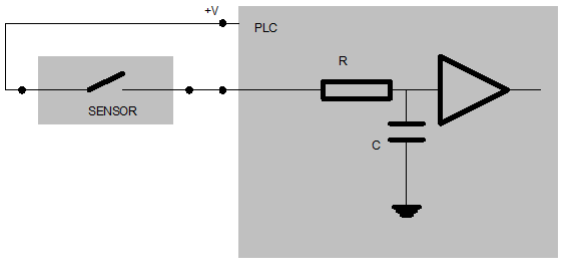
Se a impedancia de entrada do PLC é moito menor que a de saída do transductor, a corrente de entrada é prácticamente a mesma que a do transductor.

$$Ir = \frac{Zo}{Zo + Zi} \cdot Ig$$



5.1.3 Interface de variables de entrada todo-nada sen illamento galvánico

Este tipo de interfaces impleméntase só en corrente contínua. Este tipo de interfaces son axeitadas cando a lonxitude dos cables que conectan o contacto co PLC é reducida e a posibilidade de que aparezan sobretensións que poidan averiar o PLC é prácticamente nula.

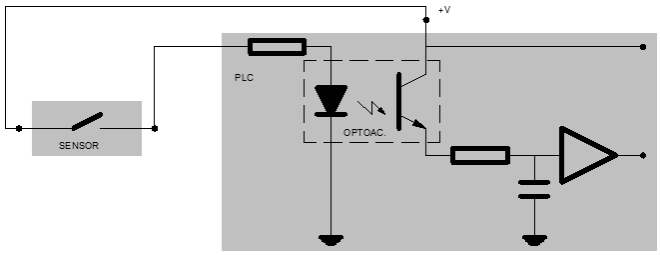


5.1.4 Interface de variables de entrada todo-nada con illamento galvánico e alimentación en continua.

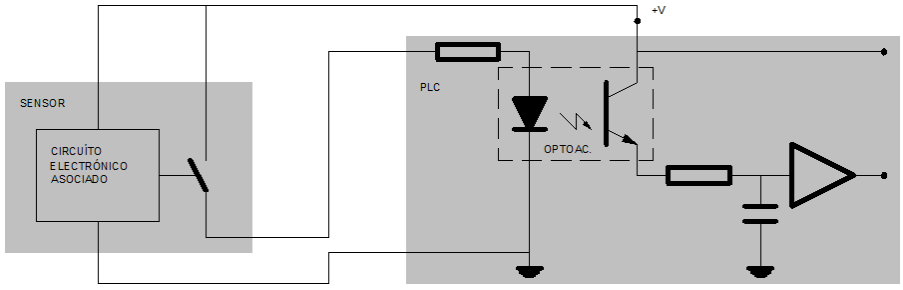
- Sensores todo-nada con saída tipo contacto

Para lograr o illamento galvánico, neste caso emprégase un dispositivo *optoacoplador*, formado por un diodo LED e un fototransistor.

Na imaxe seguinte vemos un esquema simplificado da interface de entrada para esta configuración, onde o sensor é un simple contacto.



Na imaxe seguinte vemos unha interface deste tipo, onde o sensor incorpora unha electrónica que manexa o contacto todo-nada e polo tanto necesita unha alimentación (xeralmente a 24V) que pode proceder do propio PLC ou dunha fonte de alimentación externa.



- Sensores todo-nada de tres fíos

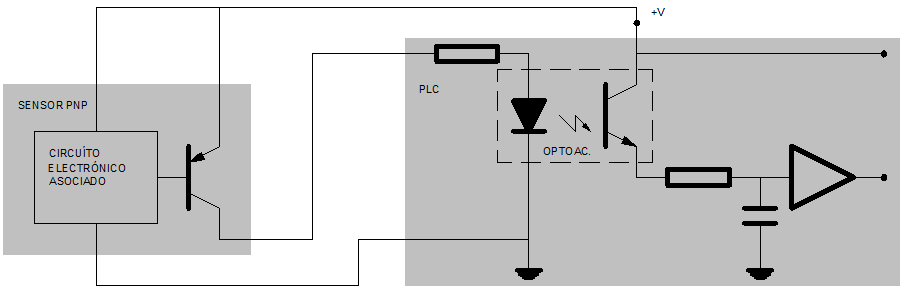
Este tipo de sensores aliméntanse a través dun terminal distinto ao de saída, polo que a corrente de alimentación non pasa a través do circuíto de entrada do PLC.

Existen as dúas variantes mencionadas anteriormente, con saída PNP e NPN.

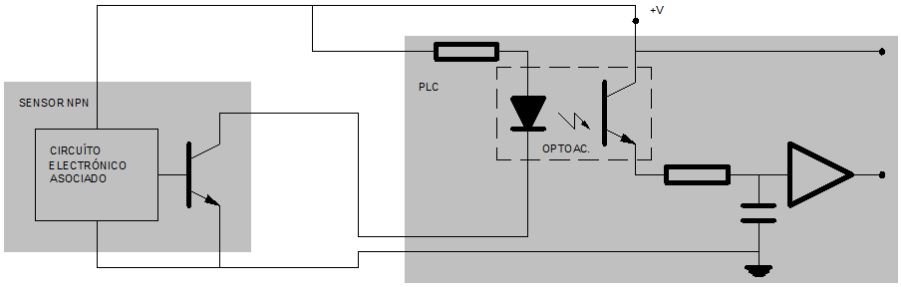
Os sensores con saída PNP caracterízanse por conmutar a corrente que sae do circuíto do módulo de entradas todo-nada do PLC, polo que o negativo da fonte de alimentación é común a todas as entradas.

Polo contrario, os sensores con saída NPN conmutan a corrente que entra no módulo de entradas todo-nada do PLC, polo que o positivo da fonte de alimentación é o común das entradas.

Na imaxe seguinte vemos unha conexión típica con saída PNP.



Na imaxe seguinte vemos unha conexión típica con saída NPN



Os sensores PNP e NPN diferéncianse polo seu comportamento ante un curtocircuíto entre a saída do sensor e o negativo da fonte de alimentación.

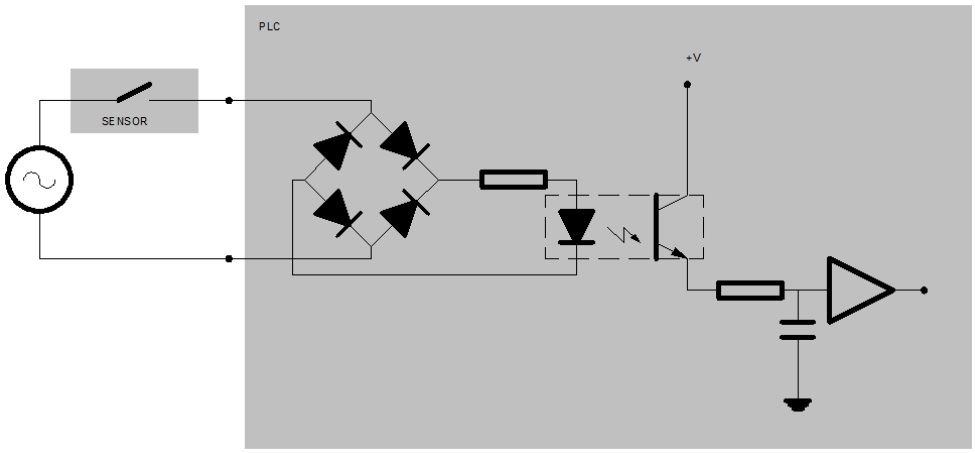
Nos esquemas anteriores é doado de ver que, si se produce un curtocircuíto entre a saída e masa no caso do NPN, a entrada do PLC detecta un contacto permanentemente pechado, ignorando o fallo.

No caso do PNP, cando se produce un curtocircuíto entre a saída e masa, no momento en que se sature o transistor, prodúcese un curtocircuíto real, sendo detectado pola fonte de alimentación.

Por este motivo, os transductores con saída PNP son máis empregados que os NPN.

5.1.5 Interface de variables de entrada todo-nada con illamento galvánico e alimentación en alterna

Mediante unha ponte rectificadora, pódese acoplar unha entrada en alterna tal como se ve na imaxe seguinte.



As tensións de alimentación habituais neste caso son 120V ou 230V.

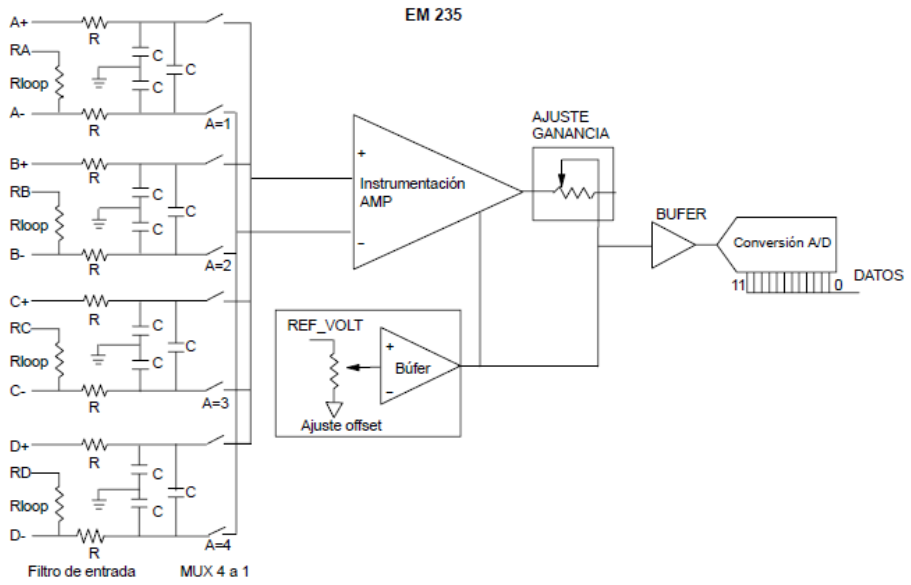
5.1.6 Interfaces de variables analóxicas de entrada

Moitos sensores subministran variables analóxicas en tensión ou en corrente. Por exemplo, o nivel de líquido nun depósito pode medirse mediante un sensor analóxico de nivel, que entregue unha tensión comprendida entre 0 e 10V cando o nivel se atope entre o 0 e o 100%.

Os fabricantes de PLCs comercializan interfaces para manexar estas variables con diferentes resolucións e rangos.

Os rangos de tensións máis habituais son ±1V, ±10V, 5V e 10V. Os rangos de corrente máis frecuentes son 0-20mA, ±20mA ou 4-20mA.

Na imaxe seguinte vemos un esquema dun módulo de catro entradas analóxicas, no que se observa un multiplexor, seguido dun aplificador de instrumentación en un conversor A/D.



Dependendo do número n de bits que incorpore o conversor A/D, a resolución será maior. Este número de bits varía entre 12 e 24 bits.

Por exemplo, se dispoñemos dun módulo de 12 bits ao que se lle aplica unha tensión de ±1V, a resolución sería:

$$r = \frac{2}{2^{12}} = \frac{2}{4096} = 0{,}58mV$$

É dicir, por cada 0,58mV que varíe o sinal de entrada, o dato binario resultado da conversión A/D varía nun bit.

5.2 Interfaces de entrada de aplicación específica

Empréganse cando se pretende traballar con determinadas magnitudes físicas de maneira máis eficiente (en xeral máis rápida) que con módulos de entradas analóxicas de aplicación xeral.

Os módulos máis característicos de aplicación específica son os que se conectan a sensores de temperatura (PT100 e termopar) ou os de contaxe.

5.2.1 Módulos de medida de temperatura para PT100

As PT100 caracterízanse por ser unha resistencia que a 0ºC ten un valor óhmico de 100 ohmios e varian este valor en función dun coeficiente, segundo a expresión:

$$R_t = R_0 \cdot (1 + \alpha \cdot t)$$

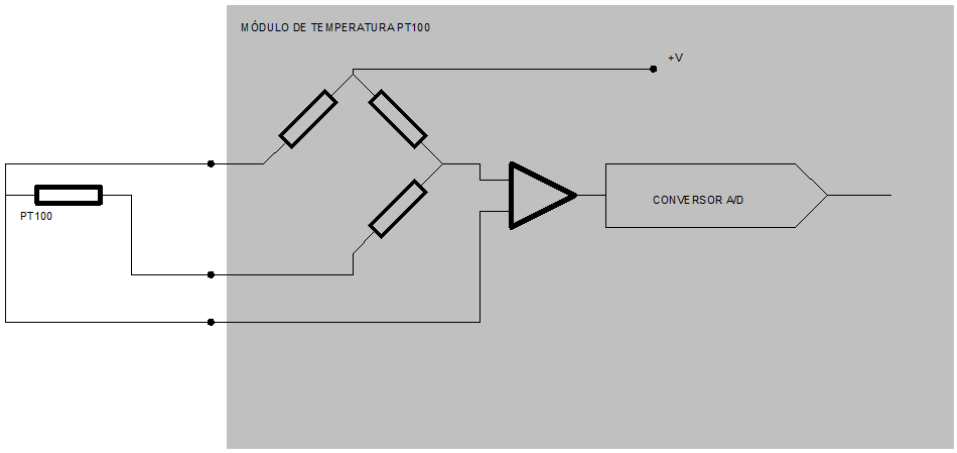
Onde R_0 é a resistencia a 0ºC, α é o coeficiente de temperatura e t a temperatura á que se atopa.

Os módulos que procesan estas variables constan de:

- Un circuíto básico de acondicionamento de sinal e un divisor de tensión ou ponte de medida.
- Un amplificador de instrumentación.
- UN conversor A/D.

En moitos casos empréganse conexións PT100 a tres fíos para minimizar o efecto das caídas de tensión nos cables que a conectan ao módulo.

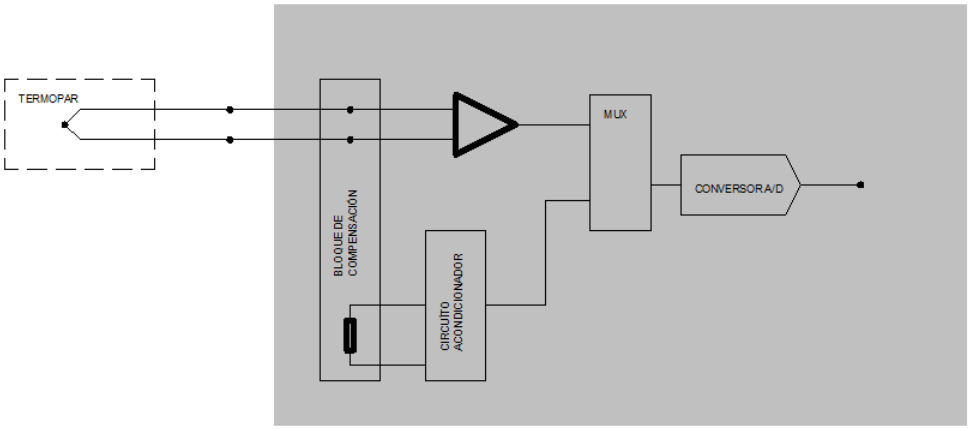
Na figura seguinte vemos un esquema típico deste tipo de conexión.



5.2.2 Módulos de medida de temperatura para termopar

Un termopar xenera unha pequena tensión (efecto Seebeck) da orde dun poucos microvoltios por cada ºC, denominada forma termoelectromotriz.

Para aproveitar esa pequena tensión, existen módulos como o da figura seguinte no que se acondiciona a mesma e se convirte a un valor binario co que pode traballar o PLC.



5.3 Interfaces de saída todo-nada

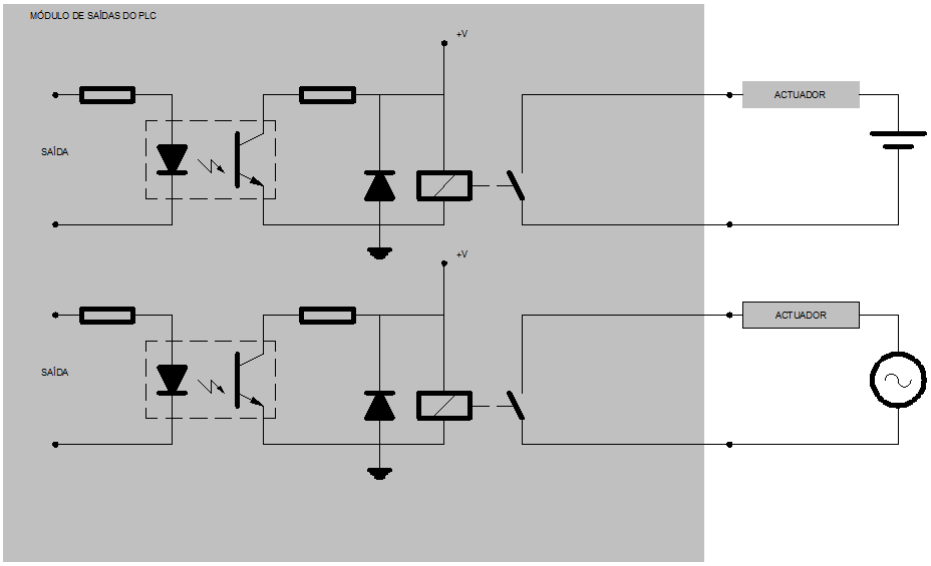
Son as encargadas de activar os correspondentes actuadores para controlar o proceso. O dispositivo de conmutación pode ser un relé ou un dispositivo electrónico.

As saídas de tipo relé conmutan correntes máis grandes que as que levan un dispositivo electrónico, pero a súa frecuencia de operación é menor.

5.3.1 Interfaces de variables de saída todo-nada con relé

Básicamente constan de un optoacoplador ao que se conecta un relé que pode ter diferentes combinacións de contactos. Na práctica, os fabricantes implementan un terminal común a varias saídas pra reducir o número de terminales e simplificar o conexionado.

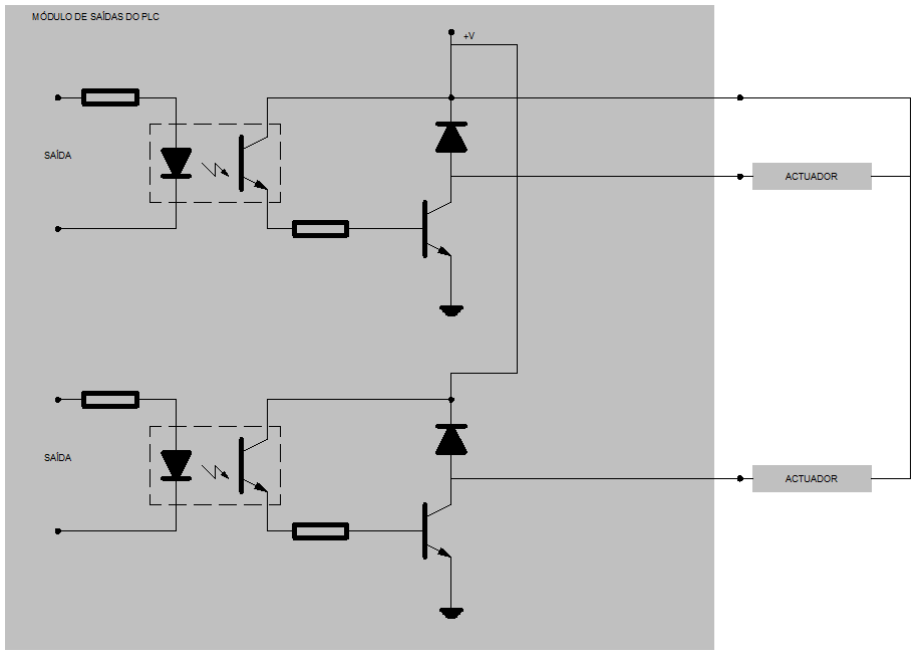
Na imaxe seguinte vemos en forma esquemática un módulo con dúas saídas a relé, e o seu conexionado con dous actuadores, o primerio alimentado en continua e o segundo en alterna.



5.3.2 Interfaces de variables de saída todo-nada con transistor NPN

Igual que no caso dos módulos de entradas NPN, neste caso temos un transistor que controla o actuador.

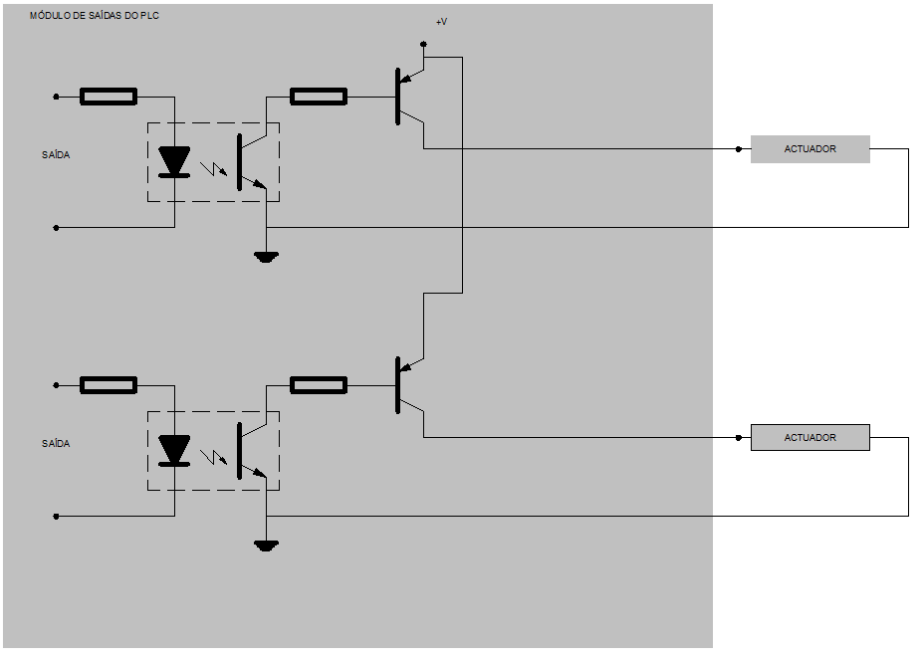
O actuador hai que conectalo como se ve na figura seguinte entre o colector do transistor e o positivo da alimentación.



5.3.3 Interfaces de variables de saída todo-nada con transistor PNP

Igual que no caso dos módulos de entradas PNP, neste caso temos un transistor que controla o actuador.

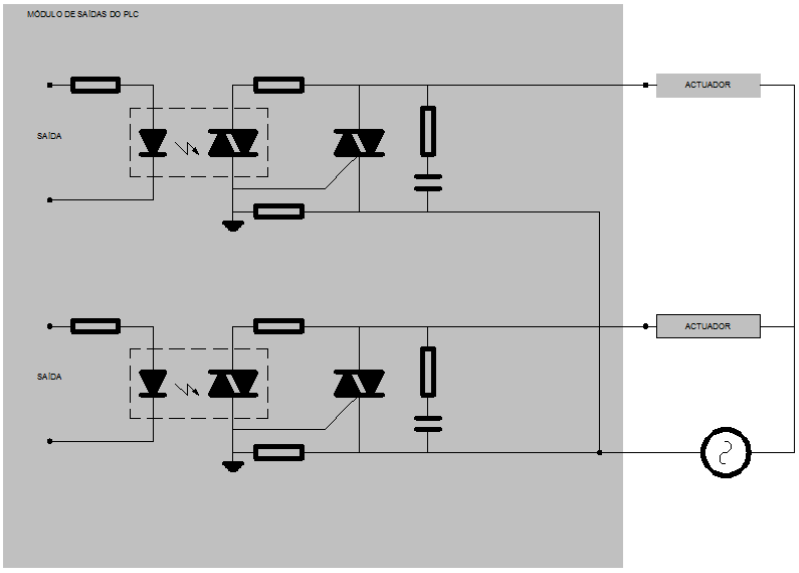
O actuador hai que conectalo como se ve na figura seguinte entre o colector do transistor e o negativo da alimentación.



5.3.4 Interfaces de variables de saída todo-nada con tiristor ou triac

Neste caso, o actuador aliméntase en corrente alterna.

Na figura seguinte vemos un exemplo.



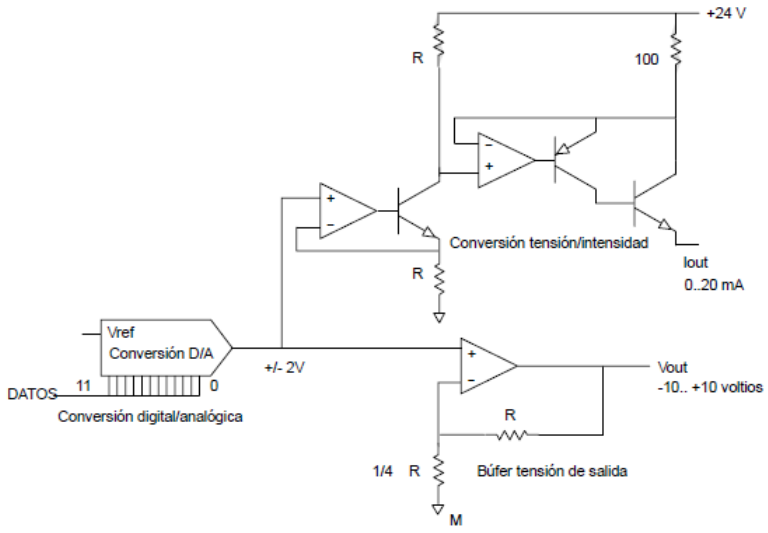
5.4 Interfaces de variables analóxicas de saída

Os PLCs incorporan interfaces de saída analóxicas para adaptar os valores dixitais que xeneran aos actuadores analóxicos existentes.

Normalmente empréganse conversores D/A con resolucións parecidas ás das entradas (a partir de 12 bits).

Cando hai que representar cantidades negativas, é frecuente utilizar o complemento a 2, deixando o último bit para o signo.

Na figura seguinte vemos un esquema dunha saída analóxica dun PLC de Siemens. Na mesma observamos o conversor D/A de 12 bits, seguido dos circuítos acondicionadores que proporcionan unha saída en tensión ou en corrente.

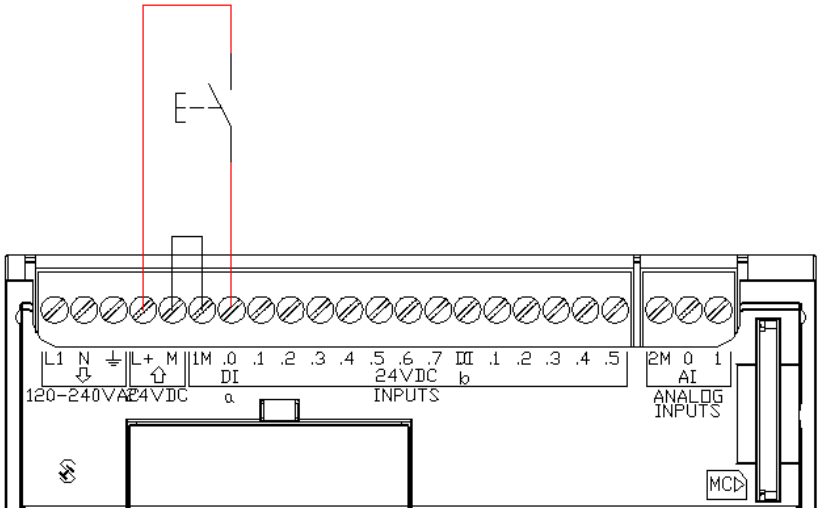


5.5 Exemplos de conexionado de pulsadores, interruptores ou sensores dixitais nas entradas dun autómata.

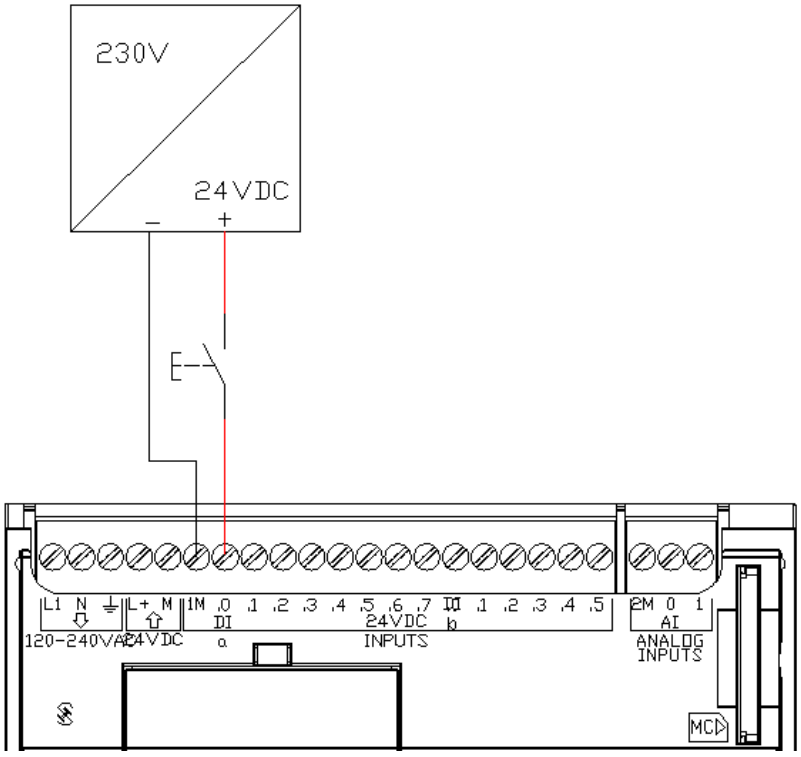
Hai que consultar a documentación do fabricante para facer correctamente o conexionado de pulsadores ou detectores nas entradas dos autómatas programables. Básicamente poden darse os casos seguintes.

5.5.1 Contacto simple empregando a fonte de alimentación própia

Para alimentación de sensores e pequenas cargas, os autómatas poden incorporar unha fonte de alimentación de 24V DC. Pódese empregar para alimentar un contacto simple.

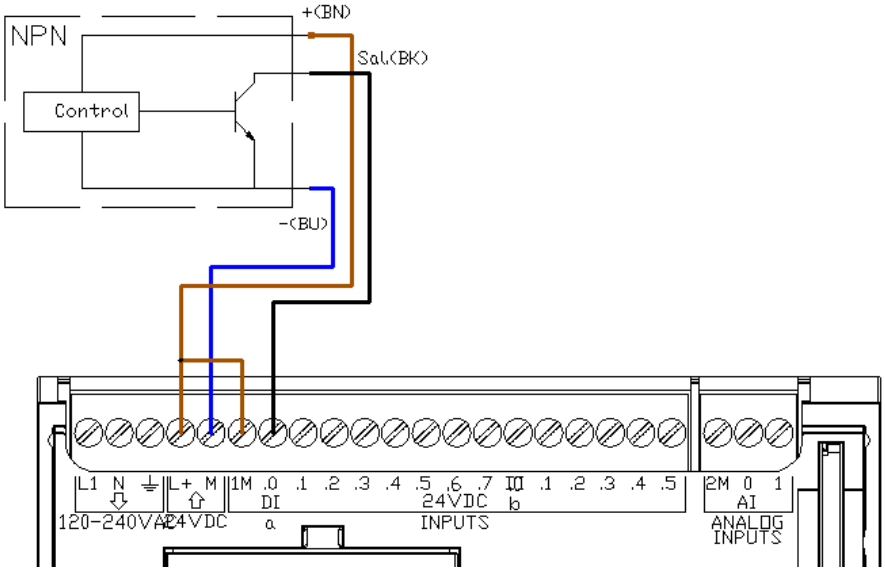


5.5.2 Contacto simple empregando unha fonte de alimentación externa



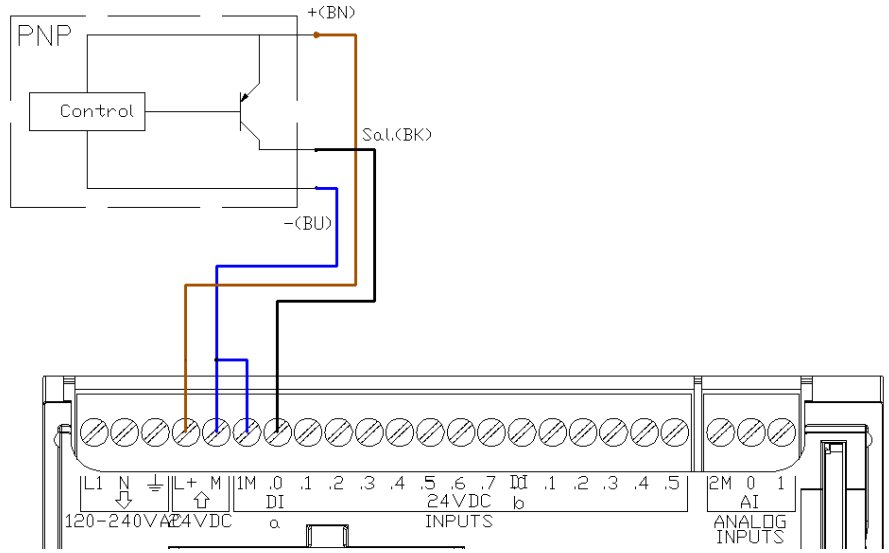
5.5.3 Conexionado dun transductor con saída a tres fíos NPN, alimentado pola própia fonte do autómata

Neste caso, o positivo da fonte de alimentación únese co común das entradas (1M).



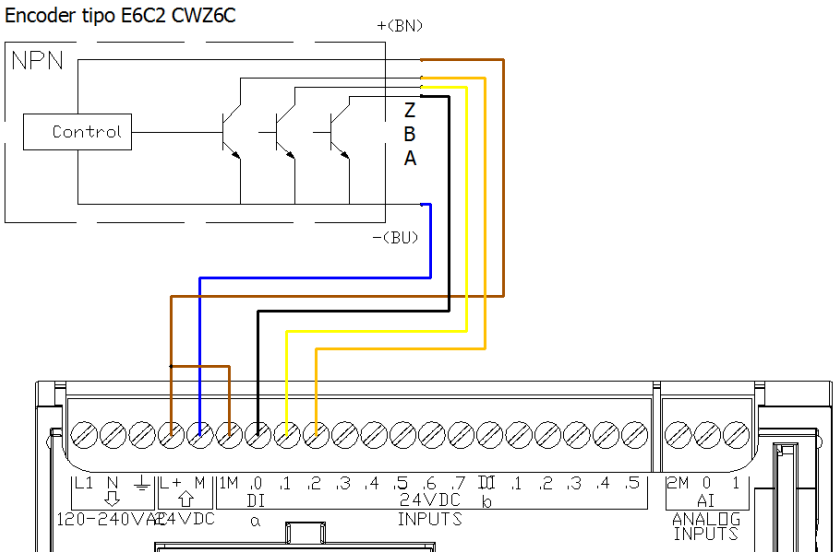
5.5.4 Conexionado dun transductor con saída a tres fíos PNP, alimentado pola própia fonte do autómata

Neste caso únense o negativo da fonte de alimentación co común das estradas (1M)



5.5.5 Conexionado dun encoder incremental con saídas A,B e Z (tipo NPN) alimentado pola propia fonte do autómata

Conéctase como si se tratara de tres detectores tipo NPN, cada un a unha entrada do autómata. As saídas A e B son as que van desfasadas entre sí un cuarto de período, para poder controlar o sentido de xiro, e a saída Z é a que marca o paso por cero.

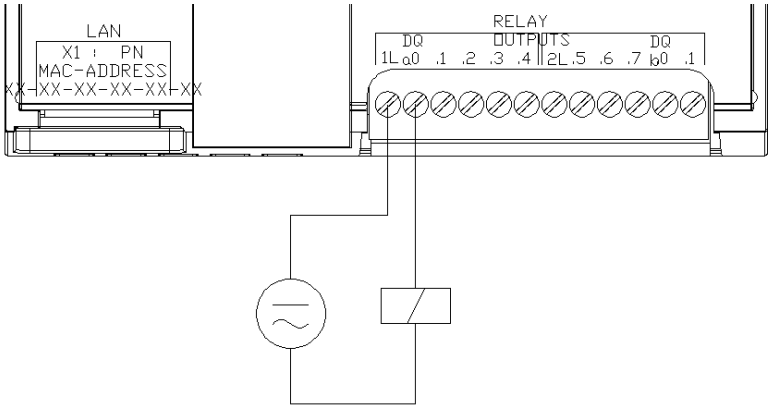


5.6 Exemplos de conexionado de saídas dixitais nun autómata programable

Distinguimos básicamente dous tipos de saídas, a relé ou a transistor.

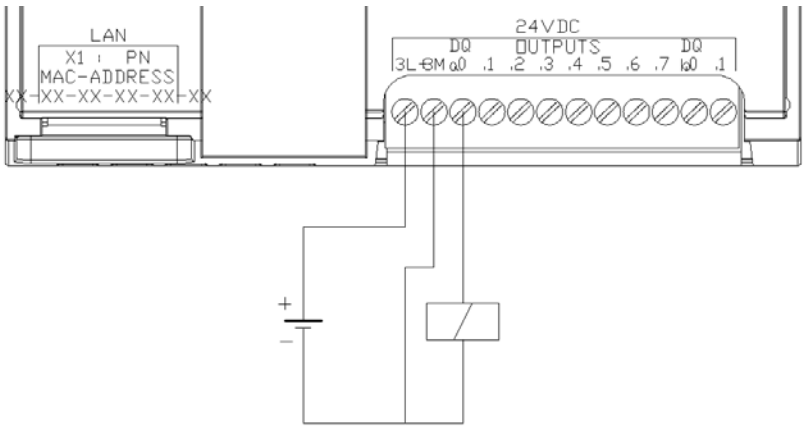
5.6.1 Saídas a relé

Nótese que se admite tanto alimentación do relé en corrente continua como en corrente alterna.



5.6.2 Saídas a transistor

Neste caso, a carga que se conecte na saída deberá funcionar en corrente contínua (habitualmente 24V), e o conexionado será como se vé na figura seguinte.



6 Sistema STEP7 de programación de autómatas.

Programar un autómata consiste en establecer unha secuencia ordenada de instruccións que resolven unha determinada tarefa de control.

Os fabricantes de autómatas programables desenvolveron diferentes linguaxes que constitúen un sistema de programación. O desenvolvemento de ditas linguaxes foi realizado por cada fabricante de forma independiente e aínda que todos eles teñen unha base común, son diferentes dun fabricante a outro e denomínanse linguaxes propietarias (Siemens empregou a norma **DIN 19239**).

A existencia de múltiples sistemas de programación propietarios diferentes e incompatibles entre sí, propiciou o desenvolvemento dun sistema de programación normalizado por parte da Comisión Electrotécnica Internacional (**Norma IEC 61131-3**).

Na actualidade coexisten o sistema de programación normalizado e os propietarios.

6.1 Características xerais de STEP7

Está formado por dous tipos de linguaxes diferentes: **linguaxes literais** e **linguaxes gráficos**.

6.1.1 Linguaxes literais

As instruccións nestas linguaxes están formadas por letras, números e símbolos especiais. Son linguaxes deste tipo:

- **AWL**: Abreviatura do alemán ("*Anweisungsliste*"), que significa precisamente lista de instruccións. É unha linguaxe propia de Siemens®. A esta linguaxe tamén se lle chama **STL** (*Statment List*).

- **SCL:** Linguaxe de texto estructurado (*Structured Control Language),* é unha linguaxe de alto nivel similar ao Pascal, que cumpre a norma IEC 61131-3. Aínda que empezou empregándose para tarefas de cálculo complexas, na actualidade, dados os coñecementos cada vez máis amplos neste tipo de programación por parte dos técnicos, emprégase como calquera outra linguaxe.

6.1.2 Linguaxes gráficas

As instruccións nestas linguaxes represéntanse mediante figuras xeométricas. Son linguaxes deste tipo:

- **LAD** ou **KOP:** Linguaxe de *esquema de contactos*. Toma os nomes do inglés (*Ladder Diagram*) ou do alemán (*Kontakts Plan*).
- **FBD** ou **FUP:** Linguaxe de *diagrama de funcións.* Toma os nomes do inglés (*Function Block Diagram*) ou do alemán (*Funktions Plan*).
- **SFC:** *Diagrama funcional de secuencias (Sequential Funtion Chart*), que en STEP7 denomínase S7-GRAPH, e que ten por principal antecedente á linguaxe GRAFCET (Grafo de control etapa-transición) desenvolvido pola Asociación Francesa para a Cibernética Económica e Técnica (AFCET).
- **CFC:** *Diagrama de transición de estados* S7-HiGraph, tamén chamada linguaxe de *conexión de bloques (Continuous Function Chart)* similar ao diagrama de funcións, no que cada bloque é a súa vez un programa.

Non todos os autómatas programables poden ser programados empregando todas as linguaxes anteriores. Por exemplo, a gama S7-200 só admite AWL, KOP e FUP, en cambio a linguaxe AWL na gama S7-1200 non se pode empregar, pero sí se pode SCL.

6.2 Tipos de datos

As instruccións de programa, sexa na linguaxe que sexa, manexan datos cos cales obteñen resultados (que son máis datos). Existe unha ampla variedade de datos. Os máis característicos son:

- Bit e secuencia de bits.

  Podemos falar de datos de 1 bit (Bool), de 8 bits (Byte), de 16 bits (Word) e de 32 bits (Dword).

Tipo de datos	Tamaño en bits	Tipo de número	Rango numérico	Ejemplos de constante	Ejemplos de dirección
Bool	1	Booleano	FALSE o TRUE	TRUE, 1,	I1.0 Q0.1 M50.7 DB1.DBX2.3 Nombre_variable
		Binario	0 ó 1	0, 2#0	
		Octal	8#0 ó 8#1	8#1	
		Hexadecimal	16#0 ó 16#1	16#1	
Byte	8	Binario	2#0 a 2#11111111	2#00001111	IB2 MB10 DB1.DBB4 Nombre_variable
		Entero sin signo	0 a 255	15	
		Octal	8#0 a 8#377	8#17	
		Hexadecimal	B#16#0 a B#16#FF	B#16#F, 16#F	
Word	16	Binario	2#0 a 2#1111111111111111	2#1111000011110000	MW10 DB1.DBW2 Nombre_variable
		Entero sin signo	0 a 65535	61680	
		Octal	8#0 a 8#177777	8#170360	
		Hexadecimal	W#16#0 a W#16#FFFF, 16#0 a 16#FFFF	W#16#F0F0, 16#F0F0	
DWord	32	Binario	2#0 a 2#11111111111111111111111111111111	2#11110000111111110000111111	MD10 DB1.DBD8 Nombre_variable
		Entero sin signo	0 a 4294967295	15793935	
		Octal	8#0 a 8#37777777777	8#74177417	
		Hexadecimal	DW#16#0000_0000 a DW#16#FFFF_FFFF, 16#0000_0000 a 16#FFFF_FFFF	DW#16#F0FF0F, 16#F0FF0F	

- <u>Números enteros.</u>

  Varios tipos:

  - Enteiro pequeno sen signo, de 8 bits (USInt).
  - Enteiro pequeno con signo, de 8 bits (SInt).
  - Enteiro sen signo, de 16 bits (UInt).
  - Enteiro con signo, de 16 bits (Int).
  - Enteiro dobre sen signo, de 32 bits (UDInt).
  - Enteiro dobre con signo, de 32 bits (DInt).

Tipo de datos	Tamaño en bits	Rango numérico	Ejemplos de constante	Dirección Ejemplos
USInt	8	0 a 255	78, 2#01001110	MB0, DB1.DBB4, Nombre_variable
SInt	8	-128 a 127	+50, 16#50	
UInt	16	0 a 65.535	65295, 0	MW2, DB1.DBW2, Nombre_variable
Int	16	-32.768 a 32.767	30000, +30000	
UDInt	32	0 a 4.294.967.295	4042322160	MD6, DB1.DBD8, Nombre_variable
DInt	32	-2.147.483.648 a 2.147.483.647	-2131754992	

- Números reais

  Tamén chamados números en coma flotante. Hai dous tipos: de simple precisión e de dobre precisión. Os dous seguen a norma IEE 754 de representación de números en coma flotante.

  - Número real de simple precisión, de 32 bits (Real).
  - Número real de dobre precisión, de 64 bits(LReal).

Tipo de datos	Tamaño en bits	Rango numérico	Ejemplos de constante	Ejemplos de dirección
Real	32	-3.402823e+38 a -1.175 495e-38, ±0, +1.175 495e-38 a +3.402823e+38	123.456, -3.4, 1.0e-5	MD100, DB1.DBD8, Nombre_variable
LReal	64	-1,7976931348623158e+308 a -2,2250738585072014e-308, ±0, +2,2250738585072014e-308 a +1,7976931348623158e+308	12345,123456789e40, 1.2E+40	Nombre_DB.nombre_var Reglas: • No se soporta el direccionamiento directo • Se puede asignar en una tabla de interfaz de OB, FB o FC

- Data e hora
  - Tempo expresado en días, horas, minutos, segundos e milisegundos (Time).
  - Data (Date).
  - Tempo transcurrido desde o comenzo do día (TOD).
  - Data e hora longo (DTL).

Tipo de datos	Tamaño	Rango	Ejemplos de entrada de constantes
Time	32 bits	T#-24d_20h_31m_23s_648ms a T#24d_20h_31m_23s_647ms Almacenado como: -2.147.483.648 ms hasta +2.147.483.647 ms	T#5m_30s T#1d_2h_15m_30s_45ms TIME#10d20h30m20s630ms 500h10000ms 10d20h30m20s630ms
Date	16 bits	D#1990-1-1 a D#2168-12-31	D#2009-12-31 DATE#2009-12-31 2009-12-31
Hora	32 bits	TOD#0:0:0.0 a TOD#23:59:59.999	TOD#10:20:30.400 TIME_OF_DAY#10:20:30.400 23:10:1
DTL (Fecha y hora largo)	12 bytes	Mín.: DTL#1970-01-01-00:00:00.0 Máx.: DTL#2554-12-31-23:59:59.999 999 999	DTL#2008-12-16-20:30:20.250

- Caracter e cadea

  - Dato de tipo caracter (Char).
  - Dato de tipo cadea de lonxitude máxima 254 caracteres (String).

Tipo de datos	Tamaño	Rango	Ejemplos de entrada de constantes
Char	8 bits	Códigos de caracteres ASCII: 16#00 a 16#FF	'A', 't', '@'
String	n+ 2 bytes	n = (0 a 254 bytes de caracteres)	'ABC'

- Matrices

  Poden crearse matrices con datos dos tipos anteriores, tendo en conta unha serie de normas que aparecen nos manuais do fabricante.

<table>
<tr><th>Tipo de datos</th><th>Sintaxis de una matriz</th></tr>
<tr><td>ARRAY</td><td>Nombre [index1_min..index1_max, index2_min..index2_max] de <tipo de datos>
• Todos los parámetros de la matriz deben tener el mismo tipo de datos.
• El índice puede ser negativo, pero el límite inferior debe ser inferior o igual que el límite superior.
• Las matrices pueden tener entre una y seis dimensiones.
• Las declaraciones multidimensionales mín..máx están separadas por caracteres coma.
• No se permiten matrices anidadas ni matrices de matrices.
• El tamaño de memoria de una matriz = (tamaño de un elemento * número total de elementos de una matriz)
<table>
<tr><th>Índice de matriz</th><th>Tipos de datos índice válidos</th><th>Reglas para índice de matriz</th></tr>
<tr><td>Constante o variable</td><td>USInt, SInt, UInt, Int, UDInt, DInt</td><td>• Límites de valores: -32768 a +32767
• Válido: Constantes y variables mezcladas
• Válido: Expresiones constantes
• No válido: Expresiones variables</td></tr>
</table>
</td></tr>
</table>

Na seguinte táboa vemos exemplos de declaración de matrices.

ARRAY[1..20] of Real	Matriz de unha dimensión, de números reais e de 20 elementos.
ARRAY[-5..5] of Int	Matriz de unha dimensión, de números enteiros e de 11 elementos.
ARRAY[1..2,1..2] of Char	Matriz de dúas dimensións, de caracteres e de 4 elementos en total.

7 Introducción á programación de autómatas S7-1200

Estes PLCs sitúanse nun punto intermedio entre a gama S7-200 e a S7-300 en canto a prestacións.

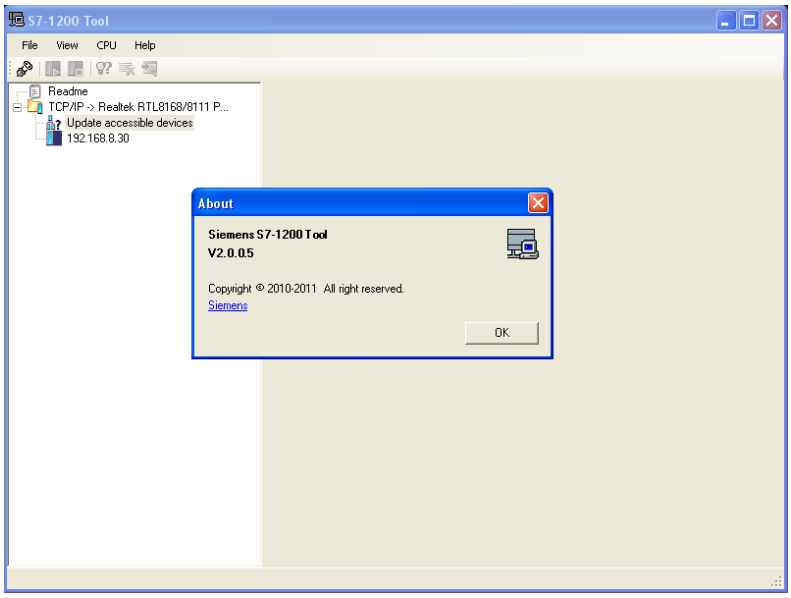
A maneira de programalos parécese máis á empregada nos S7-300.

Empregaremos o software de Siemens **TIA Portal V12** para configuralos e programalos.

A primeira diferencia que observamos con respecto aos S7-200 é que a interface de comunicación co PLC pasa de ser serie a Ethernet, polo que os novos 1200 terán asignada unha dirección IP dentro da rede de comunicacións na que se atopen.

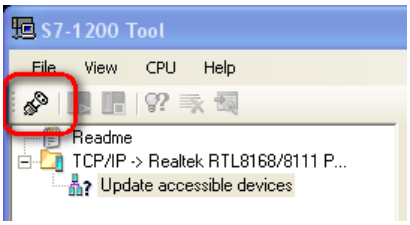
7.1 Configuración dun autómata con S7-1200 Tool

Existen algúns programas á marxe do **TIA Portal** que permiten asignar direccións IP a estes autómatas. Neste exemplo empregaremos **S7-1200 Tool** (gratuíto, dispoñible en Internet e válido só para Windows 23 bits).

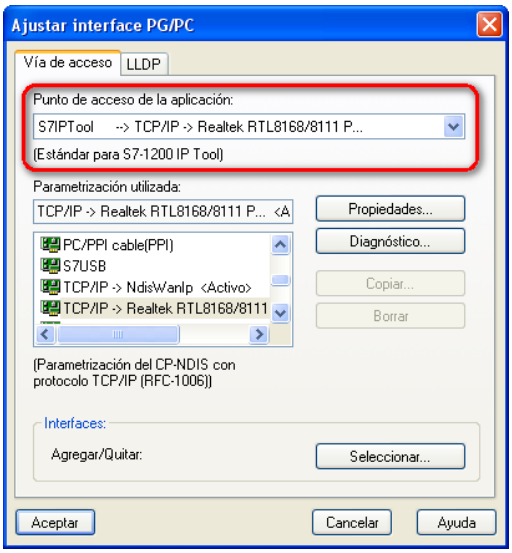


Conectamos os PLC e, ao arracar o programa, aparecen na parte esquerda os PLCs detectados.

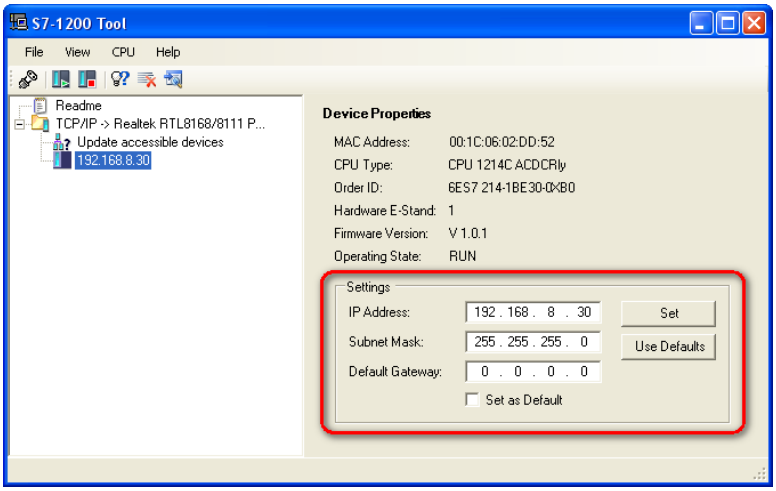
En caso de que non apareceran, teriamos que comprobar se temos ben configurado o punto de acceso á aplicación, pulsando no primeiro icono da esquerda.



Asegurámonos de que temos o punto de acceso conectado coa tarxeta de rede á que queremos conectar os PLCs.



Para cambiar a dirección IP, pulsamos sobre o PLC desexado e facemolo na parte da dereita.



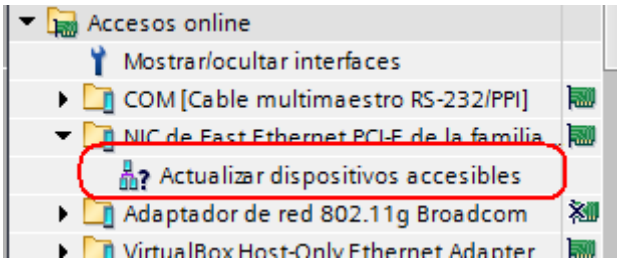
7.2 Configuración dun PLC con TIA Portal

En caso de traballar cun sistema operativo de 64 bits non podemos empregar o programa anterior. Vexamos como proceder neste caso.

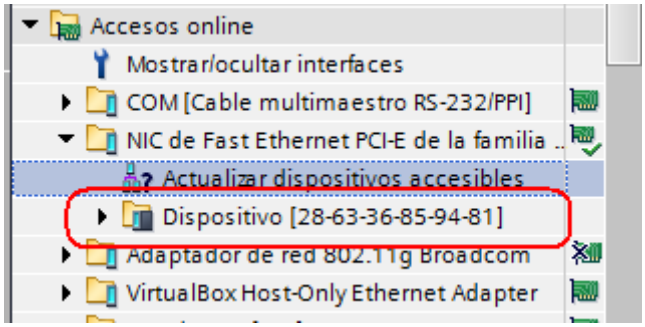
Supoñamos que temos o autómata como ven de fábrica. Neste caso non ten dirección IP asignada. Só podemos identificalo pola súa **dirección MAC**.

Arrancamos TIA Portal e creamos un proxecto en branco.

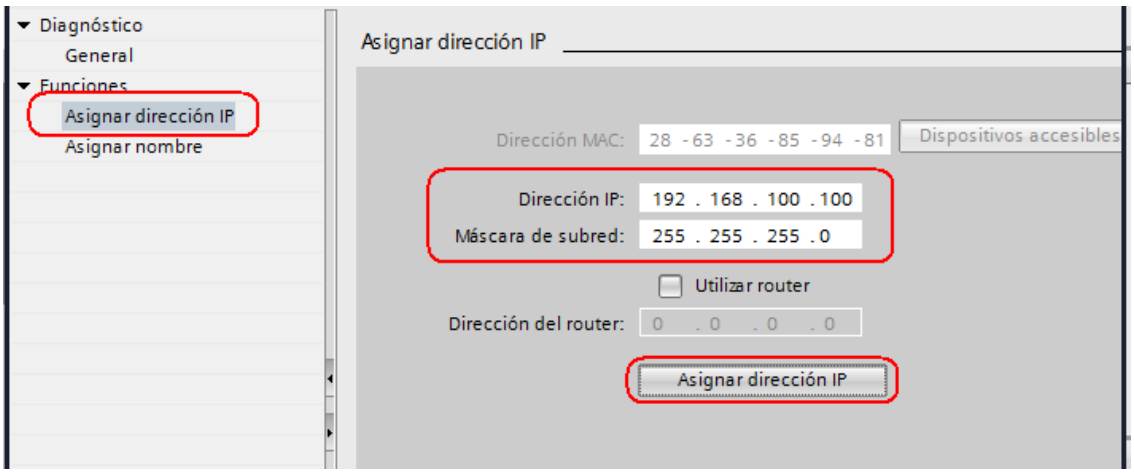
No apartado **Accesos online** localizamos a interface de rede que conecta co autómata e pulsamos sobre a opción **Actualizar dispositivos accesibles**.



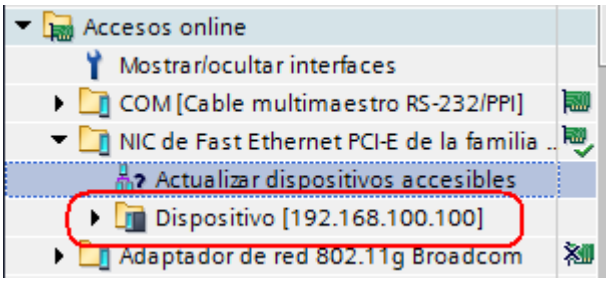
Ao cabo duns intres aparecerá o dispositivo que temos conectado coa súa dirección MAC.



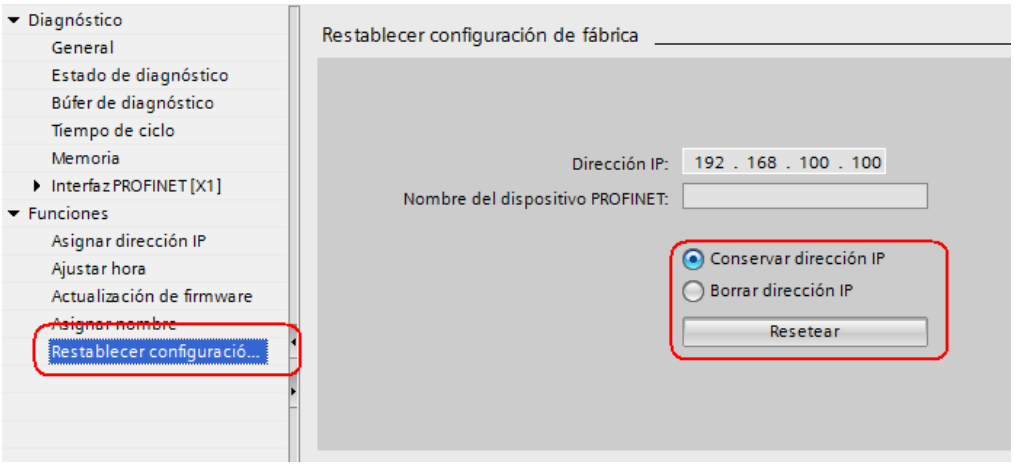
Abrimos o mesmo e pulsamos sobre **Online y diagnóstico**. Na pantalla que aparece, podemos asignarlle unha IP dentro da subrede na que nos atopamos.



Se agora facemos click de novo sobre **Accesos online->Actualizar dispositivos accesibles** vemos que xa aparece o autómata coa IP asignada.



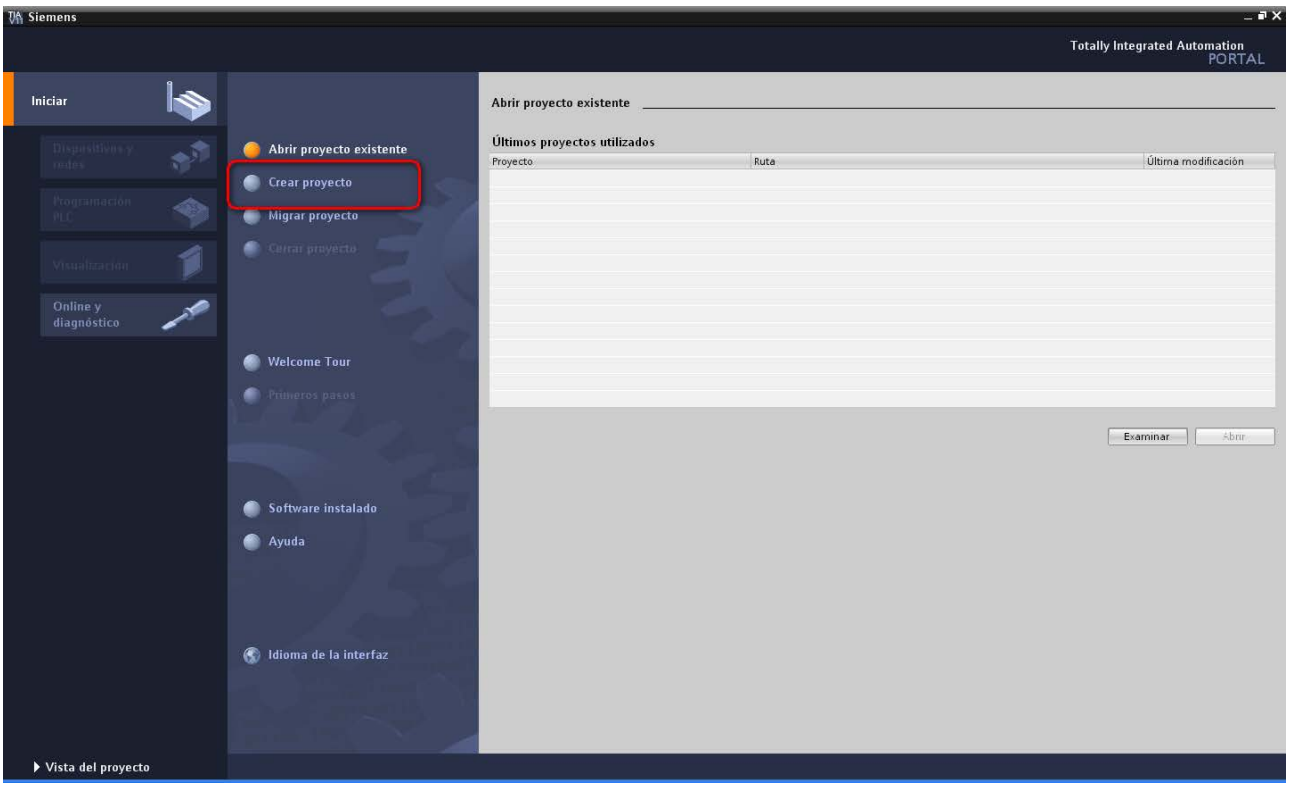
No mesmo menú podemos voltar á configuración de fábrica, unha vez que lle temos asignada a IP ao autómata.



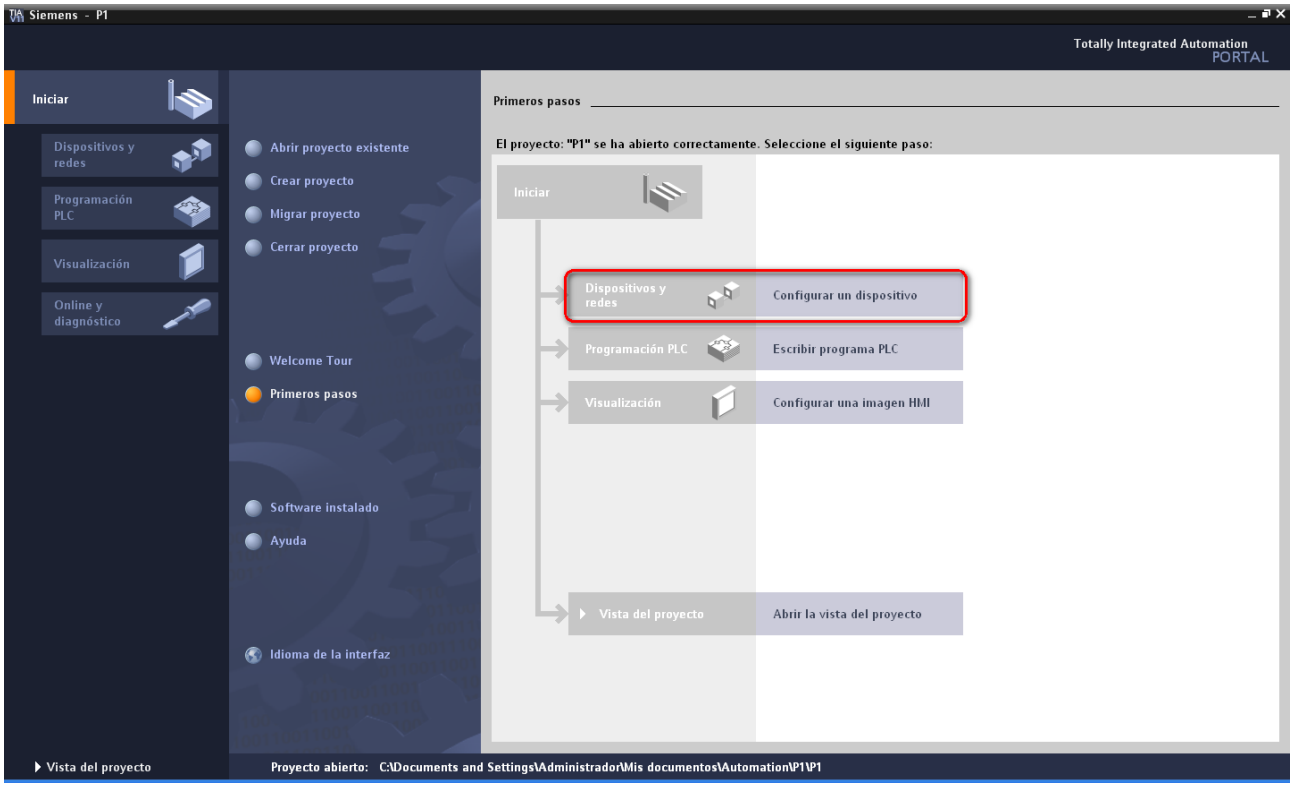
Unha vez que temos o PLC coa IP axeitada, podemos empezar a traballar co mesmo.

Empezamos creando un proxecto, como vemos na imaxe seguinte.

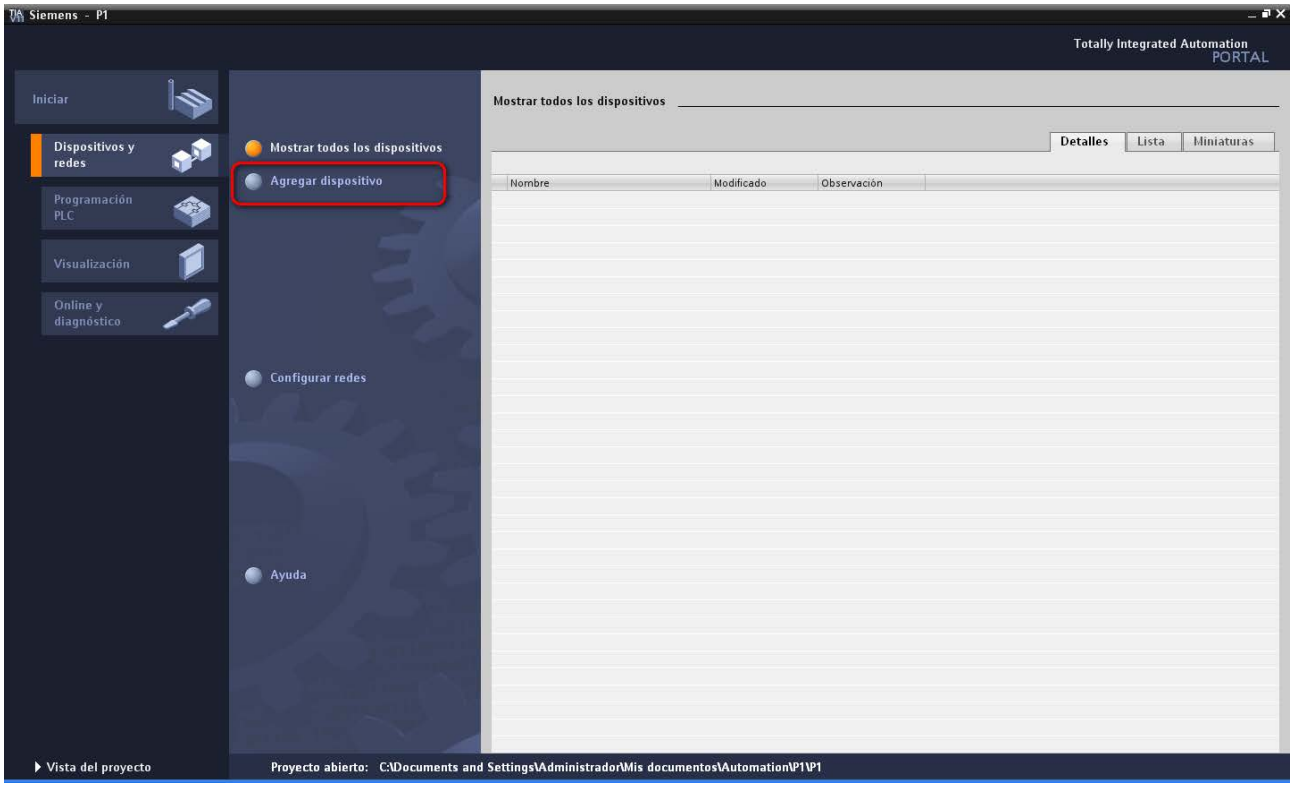
Dámoslle un nome e a ubicación onde queremos gardalo.



A continuación escollemos a opción **Configurar un dispositivo**.

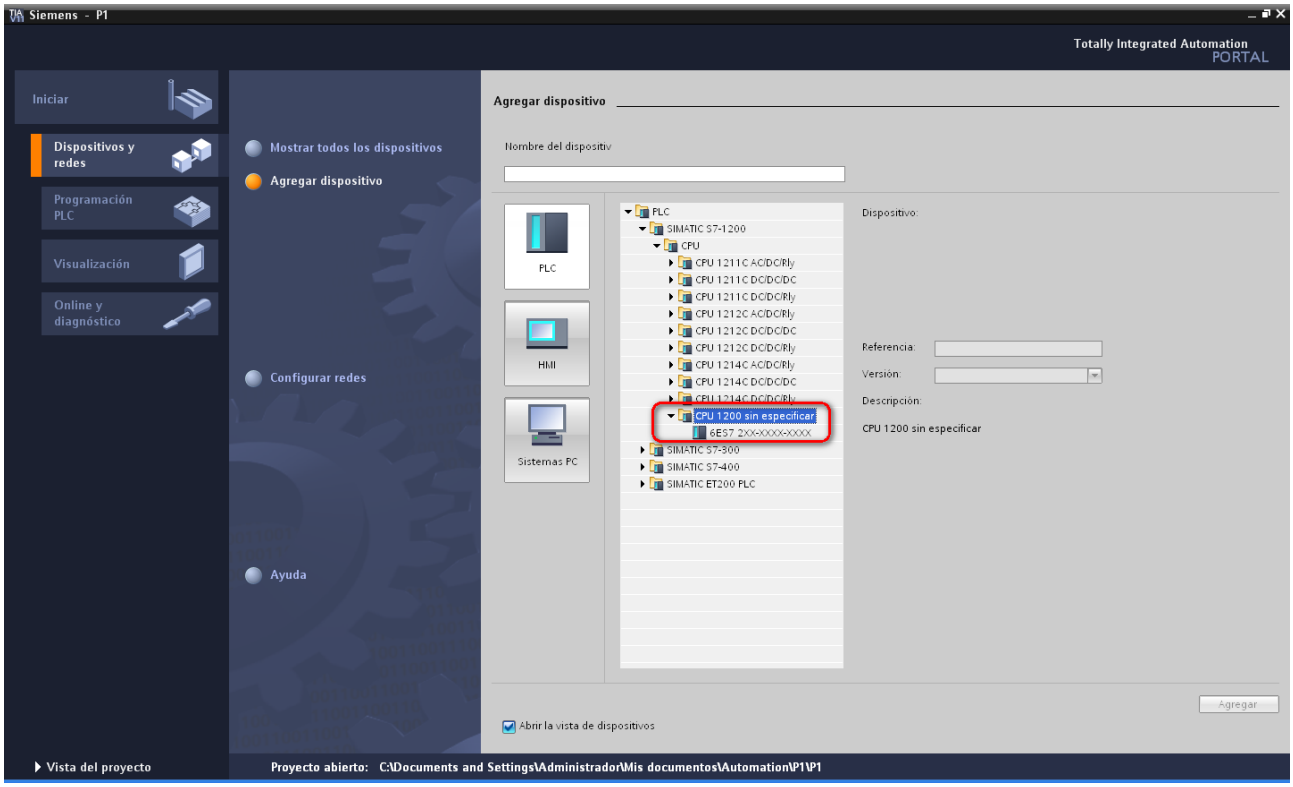


Como non temos aínda ningún dispositivo no proxecto, na seguinte pantalla escollemos a opción **Agregar dispositivo**.



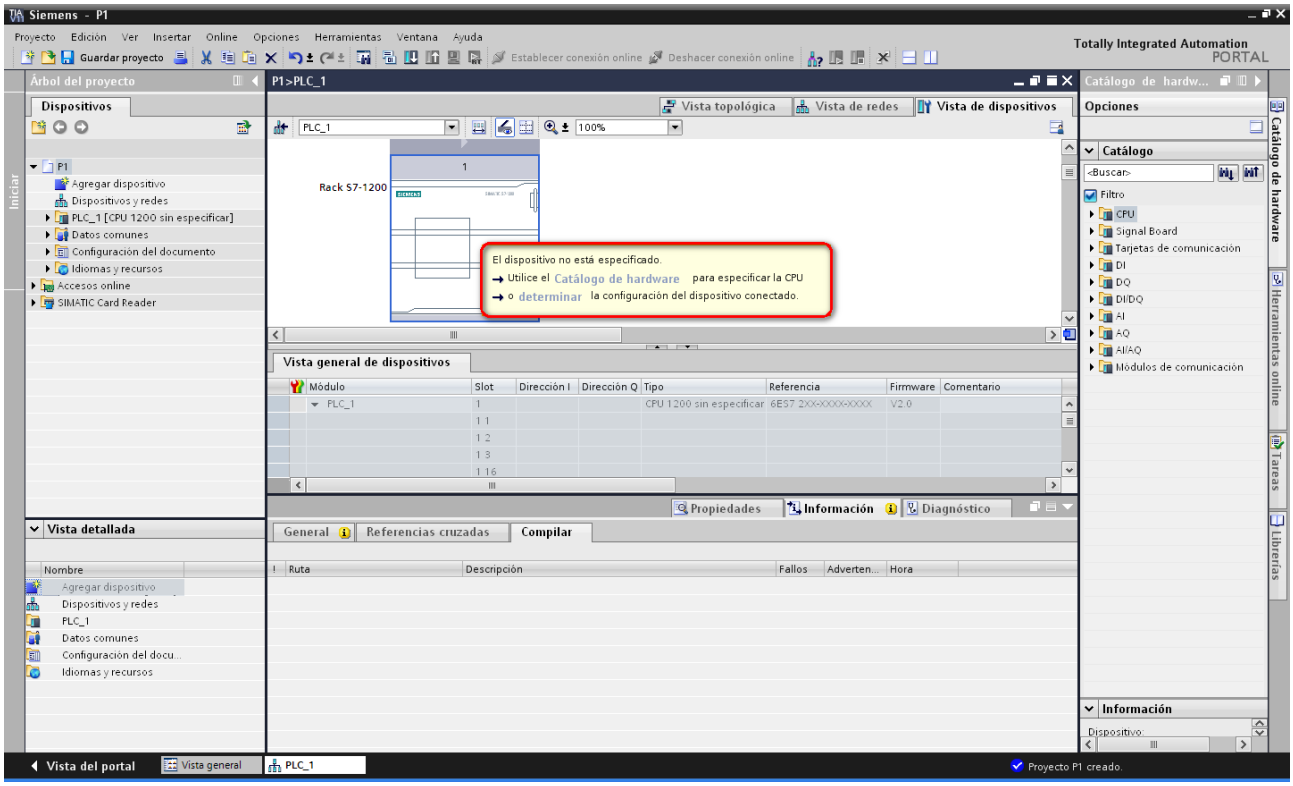
Na seguinte pantalla podemos proceder de dúas maneiras, dependendo de se coñecemos ben o hardware do PLC que estamos a configurar, ou non.

Suporemos que non temos claro de que tipo de S7-1200 se trata, polo que escolleremos un xenérico e deixaremos que o configure o **TIA Portal.**



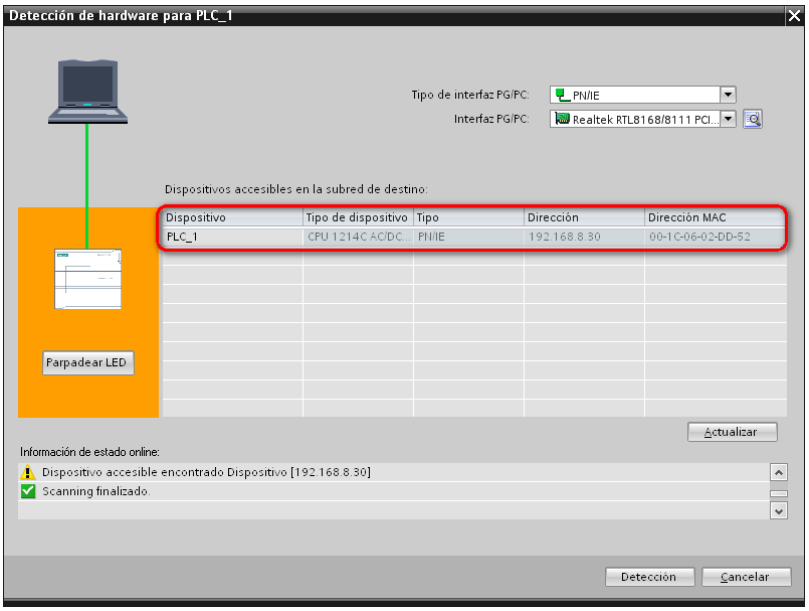
Escollemos a referencia **6ES7-2xx-xxxx-xxxx** e pulsamos o botón **Agregar**.

Ábrese o **TIA Portal** e aparece un PLC xenérico, cun aviso de que o dispositivo non está especificado, e decíndonos se queremos empregar o catálogo de hardware que incorpora o programa ou que o detecte automaticamente.



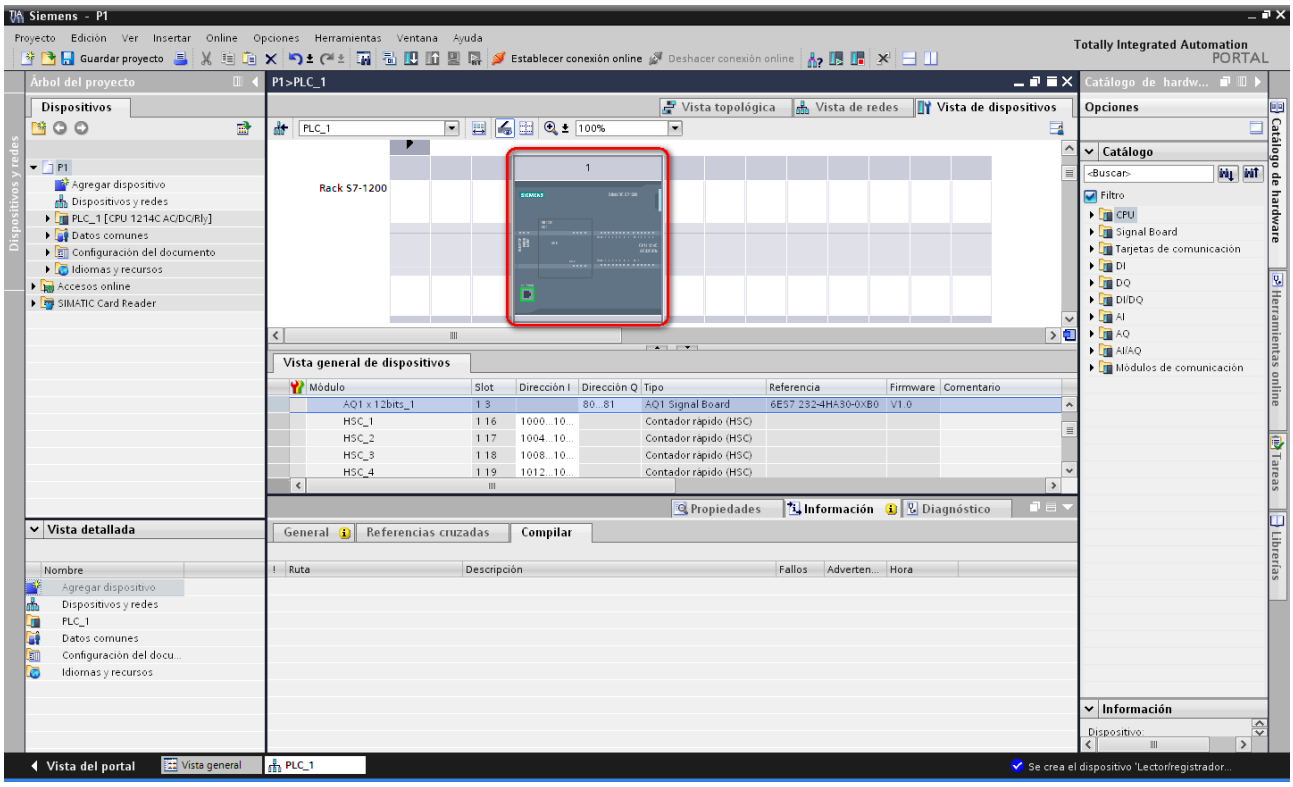
Escollemos esta segunda opción.

Despois de facer un escaneado da rede, amósase o PLC atopado.

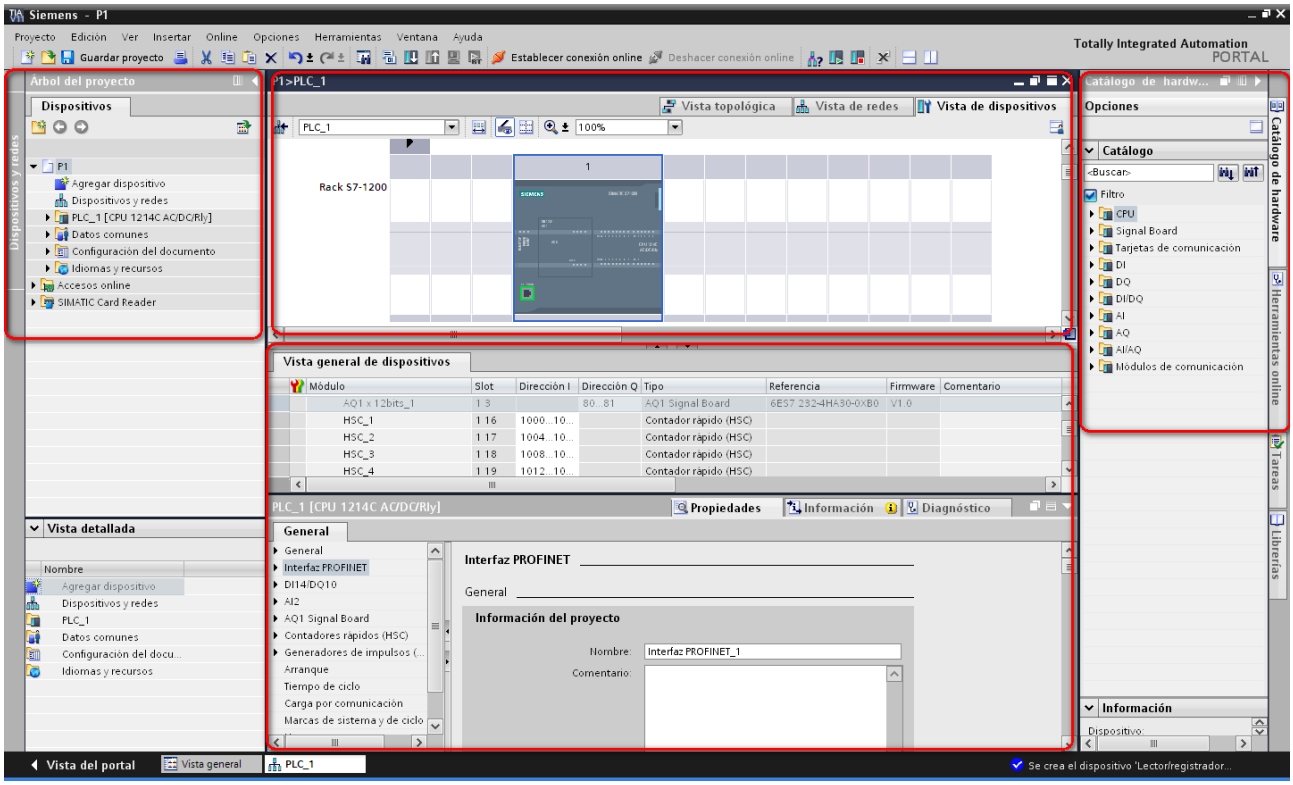


Se pulsamos sobre o botón **Parpadear LED**, veremos como hai comunicación co mesmo.

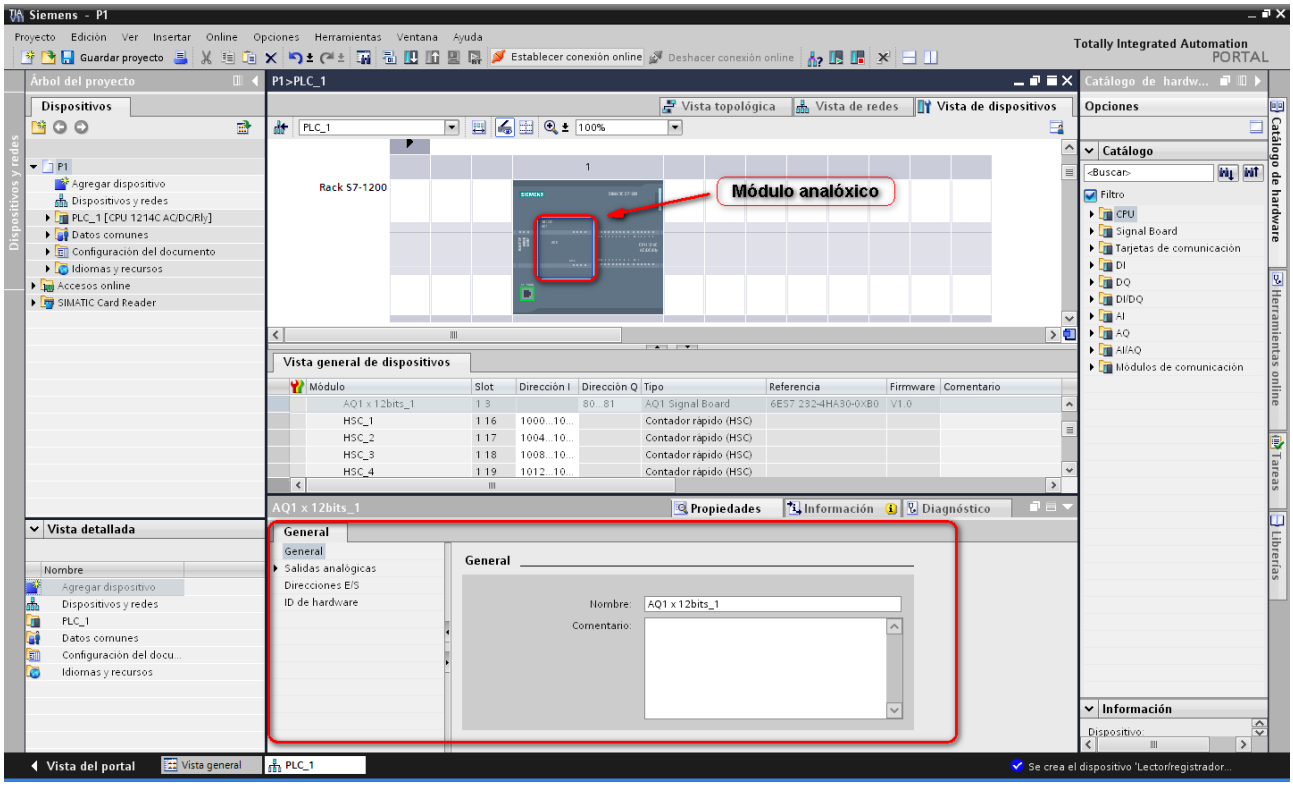
Pulsamos o botón **Detección** e aparece na ventana do proxecto o PLC configurado (incluso cos módulos adicionais que poida ter).



Como vemos, o **TIA Portal** ten o típico aspecto do software de programación actual, coa pantalla dividida en zonas. Podemos apreciar a parte da árbore do proxecto, o catálogo de hardware, a ventana de propiedades e a vista de dispositivos entre outros.

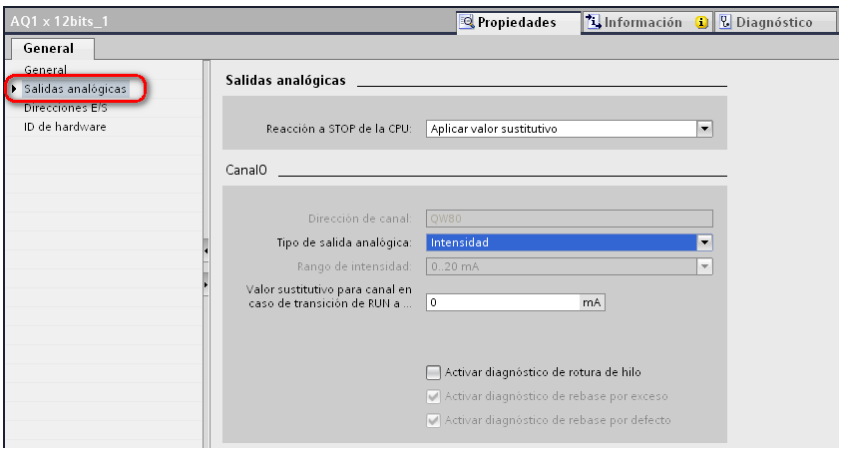


Pulsando na vista de dispositivos sobre algunha parte do hardware, aparecen as propiedades do mesmo. Por exemplo, se temos un módulo integrado cunha saída analóxica (como é o caso) e pulsamos sobre a mesma, aparecen as súas propiedades na parte inferior e podemos modificalas.



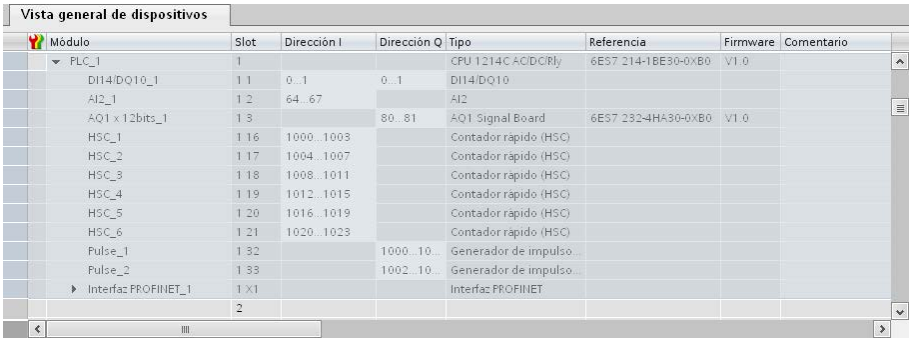
Un aspecto importante a ter en conta é que, a configuración dos módulos nestes PLCs non se fai como nos 200, a base de interruptores, senón por software.

Por exemplo, se queremos configurar a saída analóxica en corrente 0-20mA, escollemos a opción **Salidas analógicas** e facémolo.



Outra facilidade que proporcionan estes PLCs é que podemos modificar a dirección das entradas e saídas segundo nos interese. Nos PLCs 200 estabamos condicionados pola posición que ocupaba o módulo no sistema.

Podemos apreciar na **Vista general de dispositivos** as direccións de entrada/saída do PLC así como os contadores rápidos e saídas de pulsos.



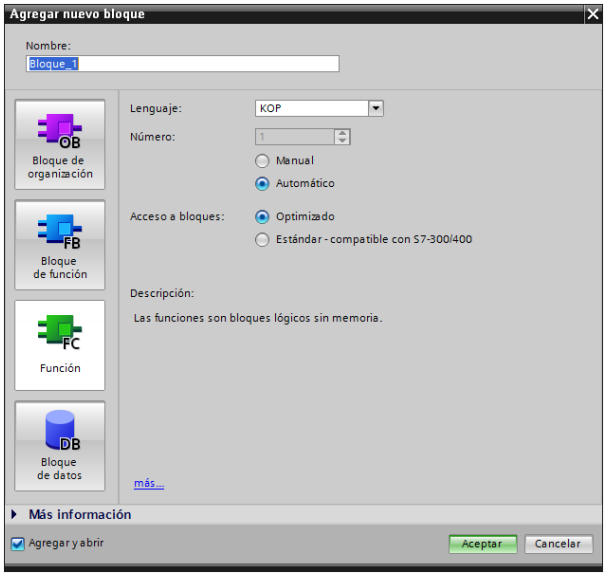
7.3 Características dun programa

O PLC soporta os seguintes tipos de bloques lóxicos que permiten estructurar o programa:

● Os bloques de organización (OBs) definen a estructura do programa. Algúns OBs teñen reaccións e eventos de arranque predefinidos. Tamén se poden crear OBs con eventos de arranque personalizados.

● As funcións (FCs) e os bloques de función (FBs) conteñen o código de programa correspondente a tarefas específicas ou combinacións de parámetros. Cada FC ou FB dispón de parámetros de entrada e saída para compartir datos co bloque que a chama. Un FB utiliza tamén un bloque de datos asociado (denominado DB de instancia) para conservar o estado de valores durante a execución que poden empregar outros bloques do programa. Os números válidos para FC e FB van de 1 ata 65535.

● Os bloques de datos (DBs) almacenan datos que poden ser empregados polos bloques do programa. Os números válidos para DB van de 1 ata 65535.

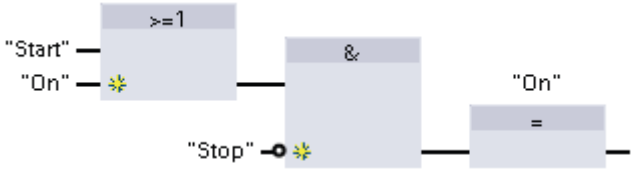


7.3.1 Linguaxes de programación en TIA Portal

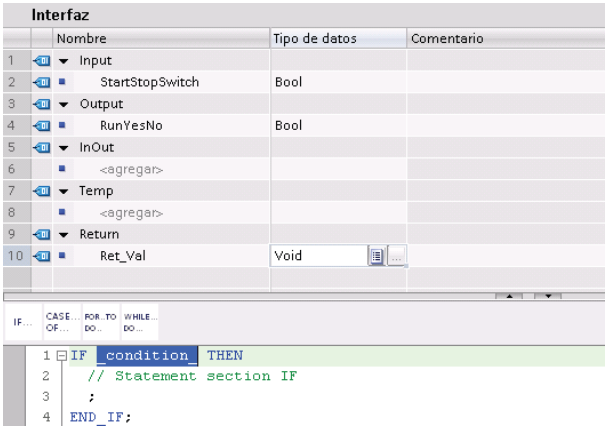
STEP 7 ofrece as linguaxes de programación estándar seguintes para S7-1200:

● KOP (esquema de contactos) é unha linguaxe de programación gráfica. A súa representación baséase en esquemas de circuitos.

● FUP (diagrama de funcións) é unha linguaxe de programación que se basea nos símbolos lóxicos gráficos empregados na álxebra booleanas.



● SCL (structured control language) é unha linguaxe de programación de alto nivel baseada en texto (norma IEC 61131 baseada nunha mistura de Pascal e Basic).



7.3.2 Acceso a datos nun PLC S7 1200

STEP 7 facilita a programación simbólica. Créanse nomes simbólicos ou "variables" para os enderezos dos datos, ben sexa como variables PLC asignadas a enderezos de memoria e E/S ou como variables locais empregadas dentro dun bloque lóxico. Para utilizar estas variables no programa de usuario basta con introducir o nome da variable no parámetro da instrucción.

O PLC ofrece varias opciónss para almacenar datos durante a execución do programa de usuario:

● **Memoria global**: O PLC ofrece distintas áreas de memoria, incluíndo entradas (I), saídas (Q) e marcas (M). Todos os bloques lóxicos poden acceder sin restricción algunha a esta memoria.

● **Tabla de variables PLC**: Pódense especificar nomes simbólicos na táboa de variables PLC de STEP 7 para posicións de memoria específicas. Esas variables son globais dentro do programa STEP 7 e permiten a programación con nomes significativos para a aplicación.

● **Bloque de datos (DB)**: É posible incluír DBs no programa de usuario para almacenar os datos dos bloques lóxicos. Os datos almacenados consérvanse cando finaliza a execución do bloque lóxico asociado. Un DB "global" almacena datos que poden ser utilizados por todos os bloques lóxicos, mentres que un DB de instancia almacena datos para un bloque de función (FB) específico e está estructurado segundo os parámetros do FB.

● **Memoria temporal**: Cada vez que se chama un bloque lóxico, o sistema operativo da CPU asigna a memoria temporal ou local (L) que debe utilizarse durante a execución do

bloque. Cando finaliza a execución do bloque lóxico, a CPU reasigna a memoria local para a execución doutros bloques lóxicos.

Dependendo do tipo de memoria que empreguemos, teremos características como forzado permanente ou remanencia.

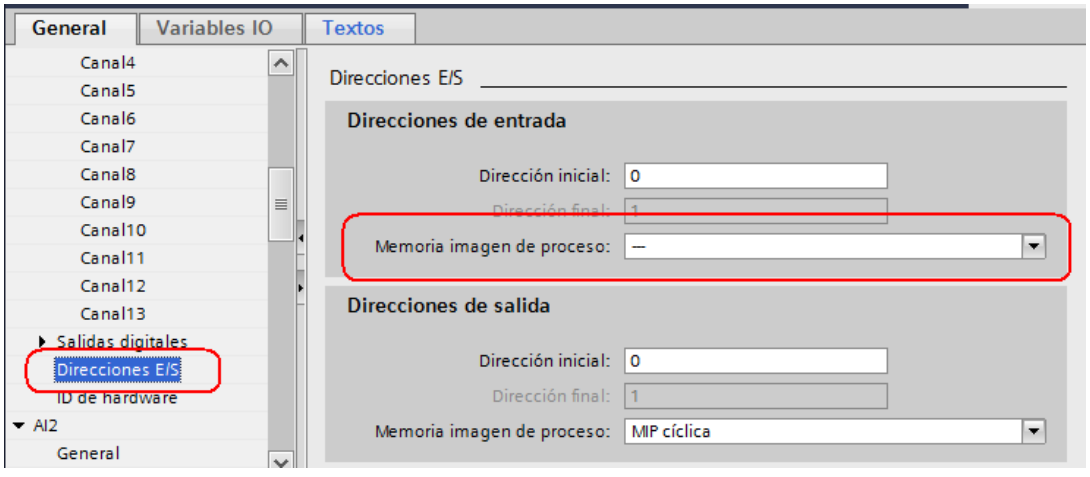
Toda posición de memoria diferente ten un enderezo unívoco. O programa de usuario emprega estes enderezos para acceder á información da posición de memoria. As referencias ás áreas de memoria de entrada (I) ou saída (Q), como I0.3 o Q1.7, acceden á memoria imaxen do proceso. Para acceder inmediatamente á entrada ou saída física é necesario engadir ":P" ao enderezo (por exemplo I0.3:P, Q1.7:P ou "Stop:P").

7.4 Aspectos a ter en conta á hora de empregar o simulador de procesos virtualmakTCP

Segundo as diferentes versións de TIA Portal que empreguemos, temos que configurar algunhas cousas para que os programas funcionen correctamente co simulador de procesos virtualmakTCP.

7.4.1 TIA Portal v12 ou anteriores e firmwares anteriores á versión 4

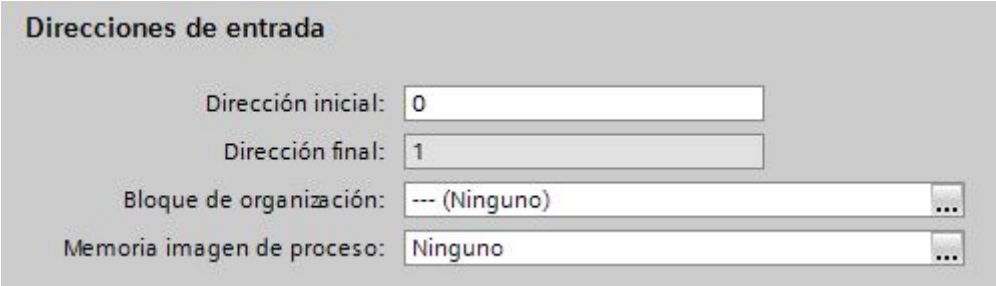
Hai que configurar as entradas dixitais e as analóxicas para que non se actualicen permanentemente desde a periferia. A forma de facelo vémola na seguinte imaxe.



Anulamos nas entradas, na opción **Memoria imagen de proceso** o valor **MIP cíclica**.

7.4.2 TIA Portal v13 e firmware 4.0 ou posterior

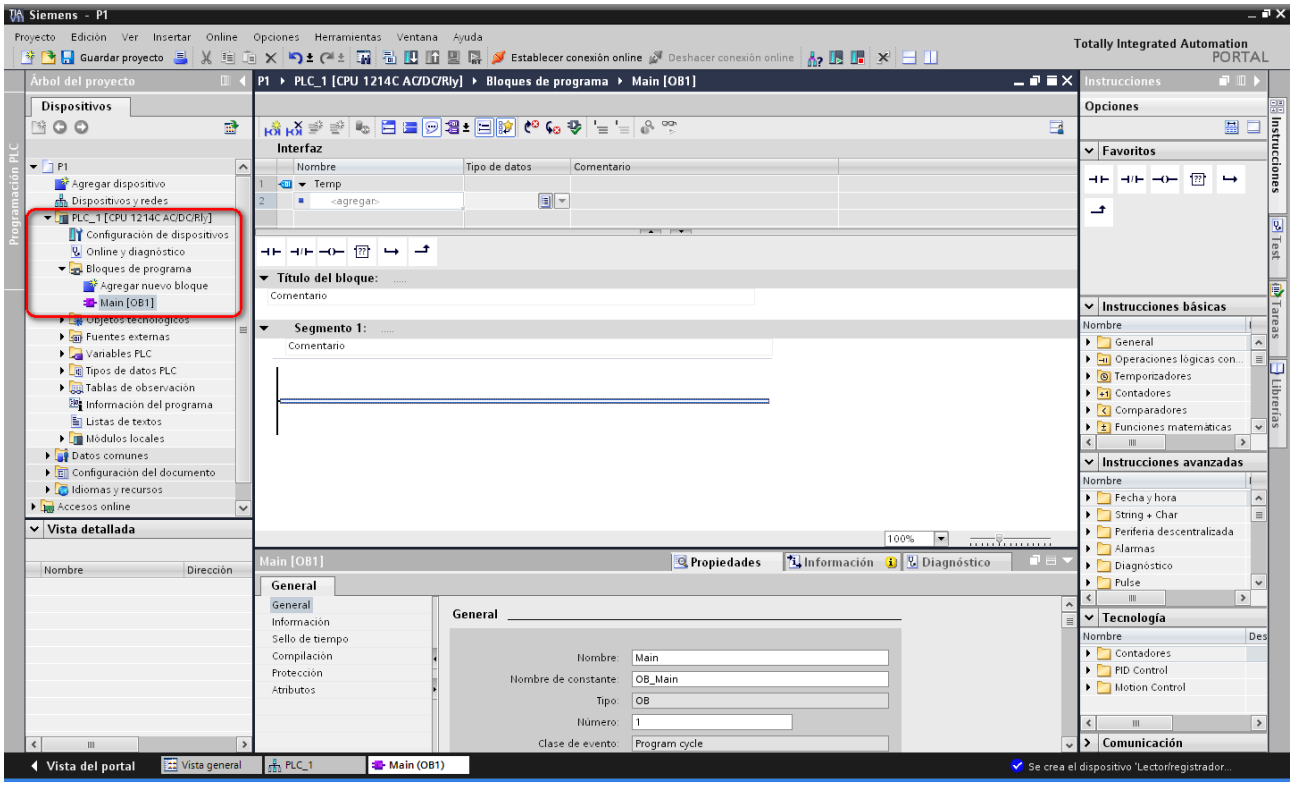
Temos que facer o mesmo do apartado anterior e poñer a opción **Ninguno** en **Memoria imagen de proceso**, como vemos na imaxe seguinte.



Ademais nesta versión de TIA Portal e firmware hai que activar **Permitir acceso vía comunicación PUT/GET del interlocutor remoto** na propiedade **Protección** do PLC.

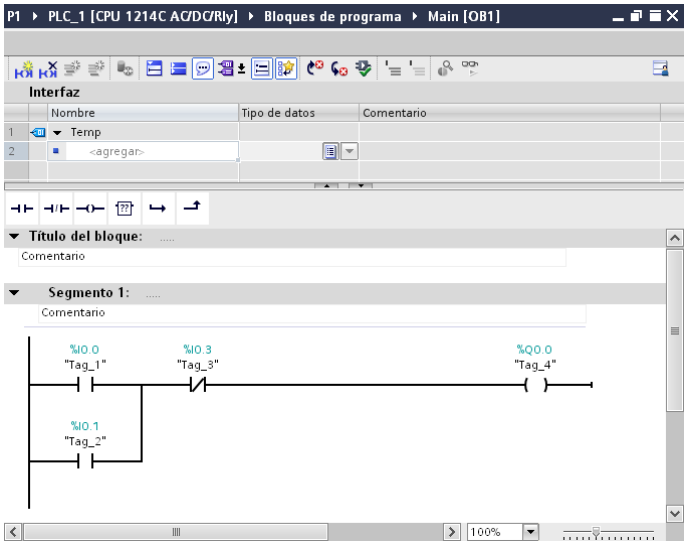
7.5 Primeiro programa

Abrimos a árbore do proxecto e seleccionamos o PLC desexado. Abrimos a opción **Bloques de programa** e aparece o bloque principal (**OB1**). Se facemos dobre click sobre o mesmo, aparece a ventana de programación.

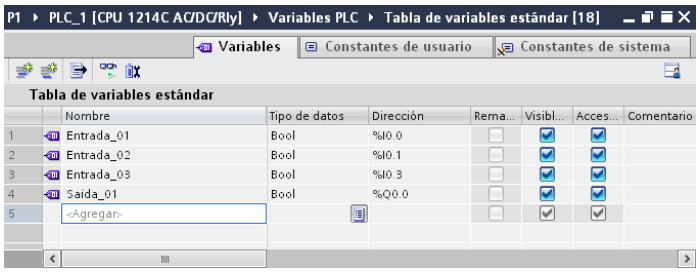


Ao seleccionar o bloque de programa vemos na parte dereita as ferramentas de programación (contactos, bloques e funcións entre outros).

A nomenclatura que se emprega parécese á da norma IEC-61131, pero non é necesario colocar o símbolo % diante dos nomes dos contactos, porque xa o fai automaticamente.



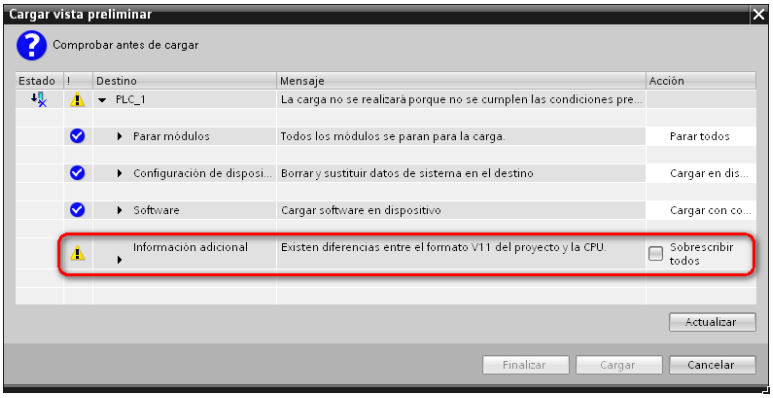
Por cada elemento que imos colocando, aparece debaixo un nome xenérico "**Tag_x**" que podemos cambiar e darlle algún valor máis ilustrativo. Para cambiar os nomes podemos ir á árbore do proxecto, escoller a opción **Variables PLC. Tabla de variables estándar** .



Despois de gardar o proxecto, podemos enviarllo ao PLC.

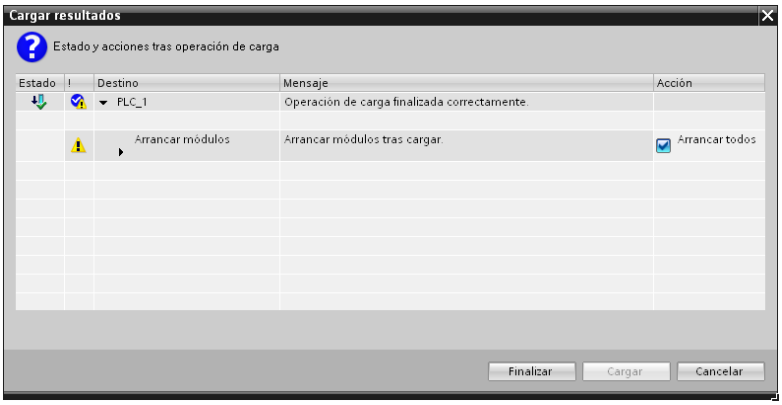
Marcamos o PLC que desexamos programar na árbore do proxecto e pulsamos o botón **Cargar en dispositivo**.

Aparece a ventana **Cargar vista preliminar** que nos informa de posibles problemas antes de proceder á carga do programa. Por exemplo, na imaxen seguinte estásenos a indicar que hai unha diferencia entre o formato da aplicación (V11) e o autómata (das primeiras versións e, polo tanto cun firmware antiguo).



Solucionamos os problemas si se pode (neste caso marcando a casilla **Sobreescribir todos**) e pulsamos sobre o botón **Cargar**.

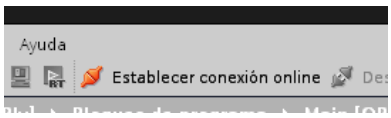
Cando remata de cargar, o PLC está en **Stop**. Pídenos que confirmemos se queremos poñelo a **Run**.



Pulsamos o botón **Finalizar**.

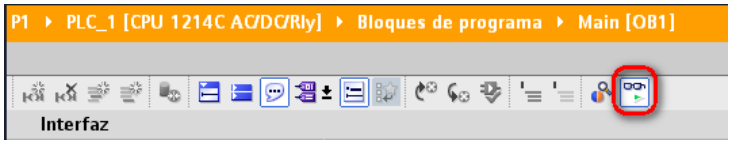
7.6 Visualizando variables

Se queremos visualizar o funcionamento do programa podemos escoller a opción do menú **Establecer conexión online** que aparece na parte superior.

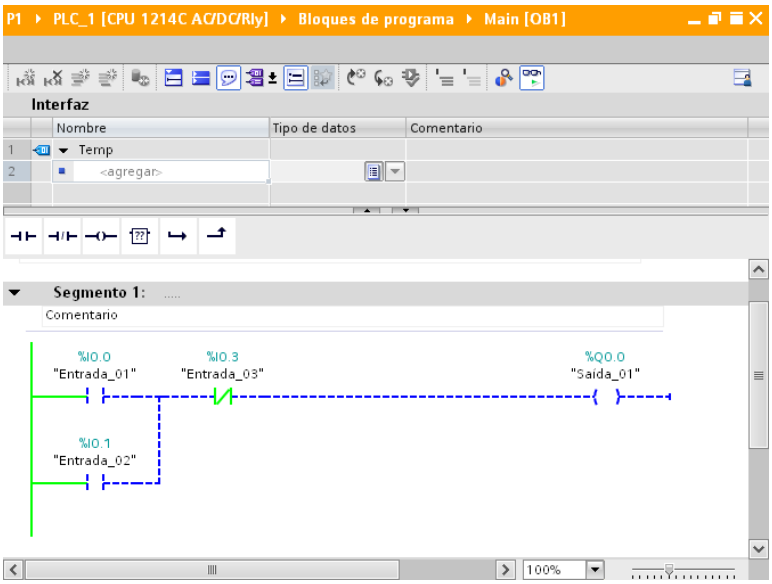


Para indicar que estamos **online**, os encabezados superiores das ventanas cambian a cor laranxa.

A continuación pulsamos sobre o botón **Activar/Desactivar observación**



O aspecto da ventana de programa cambia, para indicar as distintas partes cos niveis lóxicos correspondentes.



Outra maneira de visualizar variables consiste en crear unha táboa de observación. Para facelo, escollemos a opción do menú **Tablas de observación. Agregar nueva tabla de observación**.

Colocamos as variables que desexemos e pulsamos sobre o botón **Observar todo**.

Para saír do modo **online** pulsamos o botón **Deshacer conexión online**.

7.7 Instruccións lóxicas con bits

7.7.1 Contactos e bobinas en instruccións lóxicas con bits

As linguaxes KOP e FUP resultan moi efectivas para procesar lóxica booleana. A linguaxe SCL é particularmente efectiva para procesos matemáticos complexos, aínda que se pode empregar sen problema para resolver problemas de lóxica booleana.

Os elementos booleanos de entrada máis simples son o **contacto normalmente aberto** e o **contacto normalmente pechado**.

KOP	SCL	Descripción

"IN"	IF in THEN
	Statement;
	ELSE
	Statement;
	END_IF;
		Contactos normalmente abiertos y normalmente cerrados: Los contactos se pueden conectar a otros contactos, creando así una lógica combinacional propia. Si el bit de entrada indicado utiliza el identificador de memoria I (entrada) o Q (salida), el valor de bit se lee de la memoria imagen de proceso. Las señales de los contactos físicos del proceso controlado se cablean con los bornes de entrada del PLC. La CPU consulta las señales de entrada cableadas y actualiza continuamente los valores de estado correspondientes en la memoria imagen de proceso de las entradas.
		La lectura inmediata de una entrada física se indica introduciendo ":P" después del offset I (p. ej. "%I3.4:P"). En una lectura inmediata, los valores de datos de bit se leen directamente de la entrada física y no de la memoria imagen de proceso. La lectura inmediata no actualiza la memoria imagen de proceso.

"IN"	IF NOT (in) THEN
	Statement;
	ELSE
	Statement;
	END_IF;	

Na táboa anterior temos a representación en KOP e en SCL dun contacto normalmente aberto e dun normalmente pechado.

Os elementos de saída booleanos máis simples son as bobinas.

KOP	FUP	SCL	Descripción
"OUT"	"OUT" =	out := <Expresión booleana>;	En la programación FUP, las bobinas KOP se transforman en cuadros de asignación (= y /=), en los que se indica una dirección de bit para la salida del cuadro. Es posible conectar las entradas y salidas del cuadro con otros cuadros lógicos, o bien introducir una dirección de bit. La escritura inmediata en una salida física se indica introduciendo ":P" después del offset Q (p. ej. "%Q3.4:P"). En una escritura inmediata, los valores de datos de bit se escriben en la memoria imagen de proceso de las salidas y directamente en la salida física.
"OUT"	"OUT" /=	out := NOT <Expresión booleana>;	
	"OUT" /=		

Na táboa anterior temos a representación en KOP, FUP e SCL da bobina normalmente desconectada e a normalmente conectada.

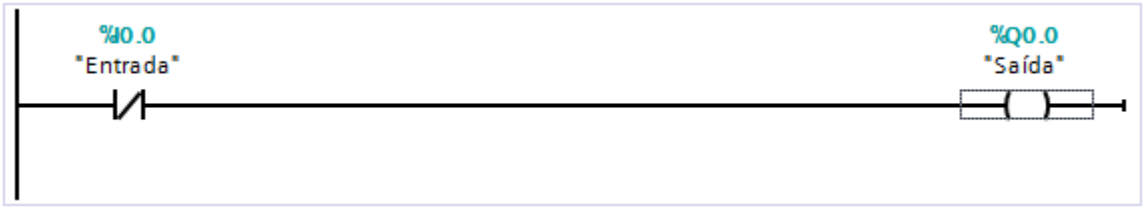
7.7.2 Operacións lóxicas

- Operación de selección dunha variable de entrada directa

Represéntase por un contacto normalmente aberto, que acciona unha variable de saída.

- Operación de selección dunha variable de entrada invertida

Represéntase mediante un contacto normalmente pechado, que acciona unha variable de saída.



- Operación lóxica OR (ou)

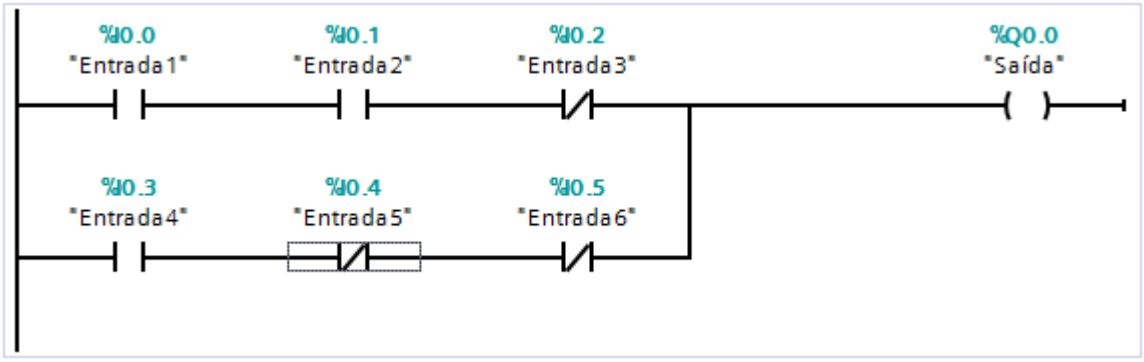
Represéntase mediante a montaxe de contactos en paralelo, que poden ser normalmente abertos, normalmente pechados ou combinacións dos dous tipos.

- Operación lóxica AND (e)

Represéntase mediante a montaxe en serie de contactos.

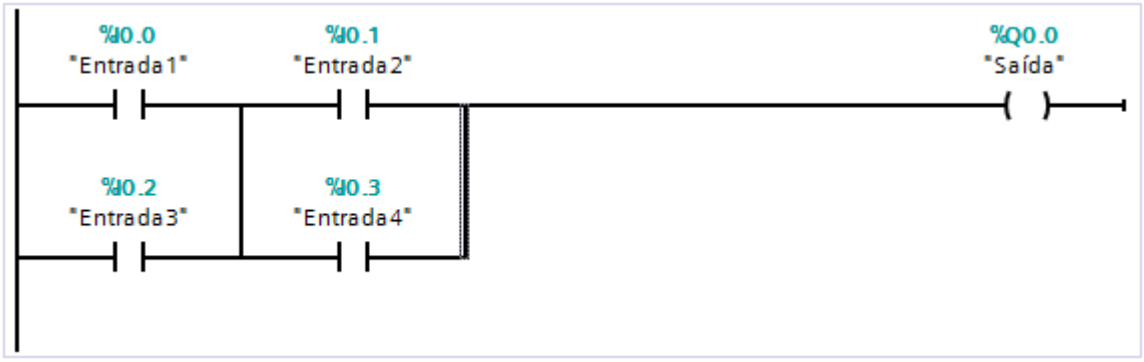
- Operación lóxica OR de operacións AND

Combinación en paralelo de contactos conectados en serie.

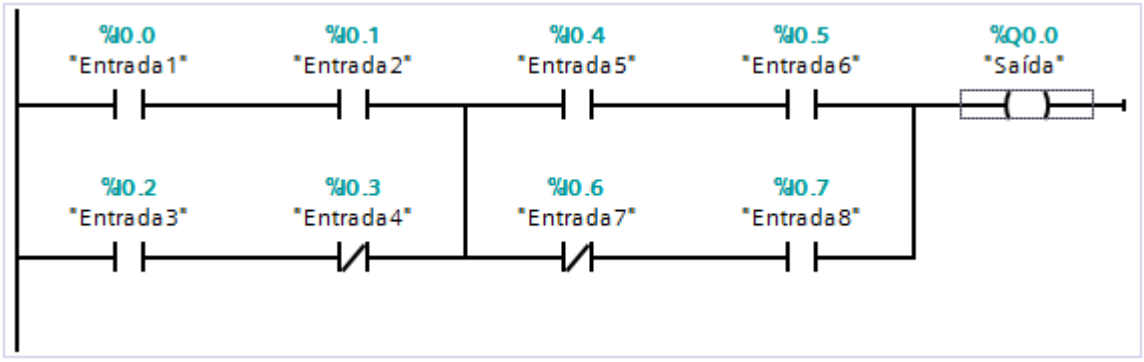


- Combinación lóxica AND de contactos OR

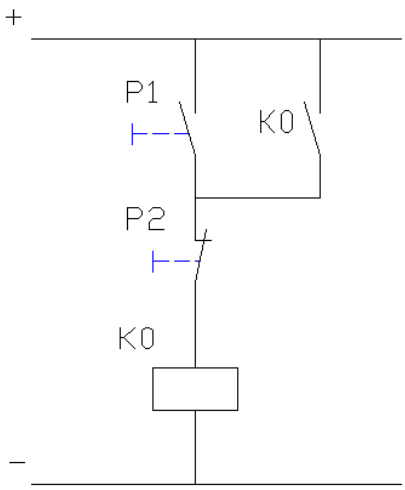
Represéntase mediante a combinación en serie de contactos conectados en paralelo.



Empregando as dúas combinacións anteriores pódense obter operacións complexas como a representada na figura seguinte.



Coas funcións lóxicas OR e AND pódese facer o típico circuíto de retención, no que activamos unha saída mediante un pulsador (P1) e mantémola activada ata que accionamos un segundo pulsador (P2).

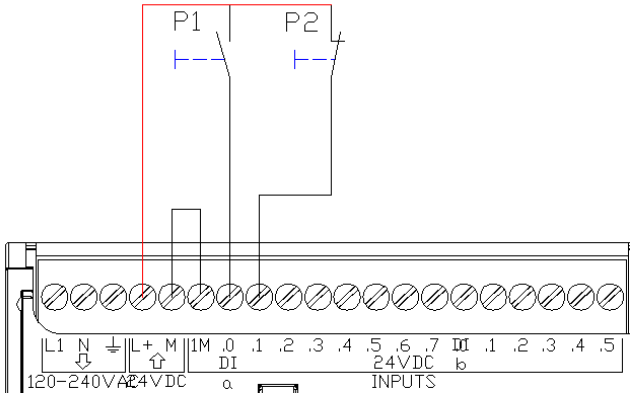


A táboa de entradas e saídas sería:

Entradas	Descrición
I0.0	Pulsador de marcha P1 (NO)
I0.1	Pulsador de paro P2 (NC)
Saídas	Descrición
Q0.0	Contactor K0

Agora temos un dos dilemas máis habituais cando un se inicia na programación de autómatas. A dúbida consiste en saber se o pulsador de paro P2 (NC) hai que representalo como un contacto aberto ou pechado. O primeiro que se nos ocorre é un esquema como o seguinte.

Teriamos un cableado como o que aparece na figura seguinte.



O pulsador P2 está a dar un nivel de sinal alto (1) cando non se pulsa, logo se o metemos como un contacto negado tal como aparece no esquema de contactos anterior, en condicións de funcionamento normais non dará sinal, e a bobina non se activa.

A solución correcta consiste en consideralo como un contacto normalmente aberto no esquema. Tal como vemos na figura seguinte.

Por outra banda, observamos como se pode consultar o estado dunha saída como se dunha entrada se tratara.

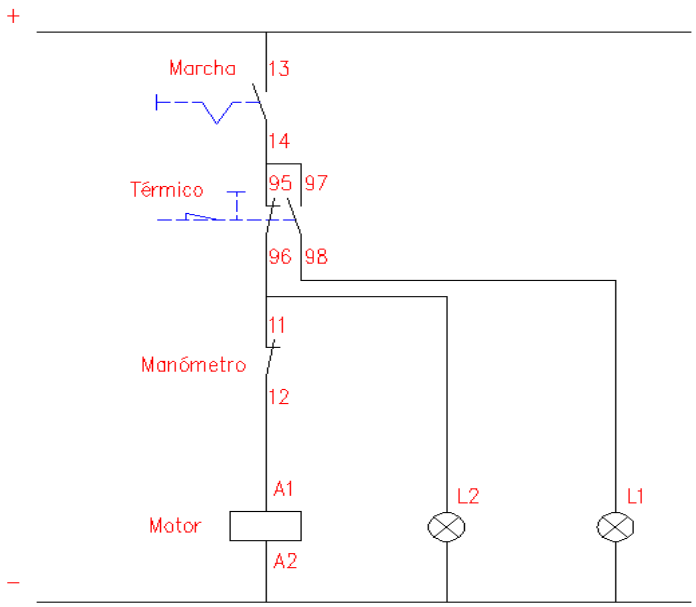
7.7.2.1 Exercicio 1

Dispoñemos dun compresor de aire que contén os seguintes elementos:

- Un interruptor de marcha que conecta o contactor do motor.
- Un relé de sobreintensidade do motor que dispón dun contacto auxiliar normalmente pechado, que se abre cando a intensidade que circula polo motor supera un valor prefixado.

- Un manómetro que posúe un contacto normalmente pechado, que se abre cando a presión supera un valor prefixado e provoca a parada do motor. Dito contacto péchase de novo cando a presión baixa por debaixo doutro valor prefixado.
- Unha lámpada de sinalización de alarma que se activa cando de dispara a protección térmica (L1).
- Unha lámpada de sinalización de servicio que se activa cando está pechado o interruptor de marcha e non se disparou a protección térmica, independentemente do estado do contacto auxiliar do manómetro (L2).

Segundo as especificacións, o esquema sería algo como o seguinte.



Asignamos variables de entrada/saída segundo a táboa seguinte.

Entradas	Descrición
I0.0	Interruptor de marcha
I0.1	Contacto auxiliar (NC) da protección térmica
I0.2	Contacto auxiliar (NC) do manómetro
Saídas	Descrición
Q0.0	Contactor do motor
Q0.1	Alarma de protección térmica (L1)
Q0.2	Sinalización de servicio (L2)

De acordo coas especificacións, a saída Q0.0 activarase cando se accione o interruptor de marcha e non teña saltado o térmico nin o manómetro por sobrepresión (nótese que os

contactos do térmico e do manómetro son NC polo que en condicións de funcionamento normal están dando sinal).

$$Q0.0 = I0.0 \cdot I0.1 \cdot I0.2$$

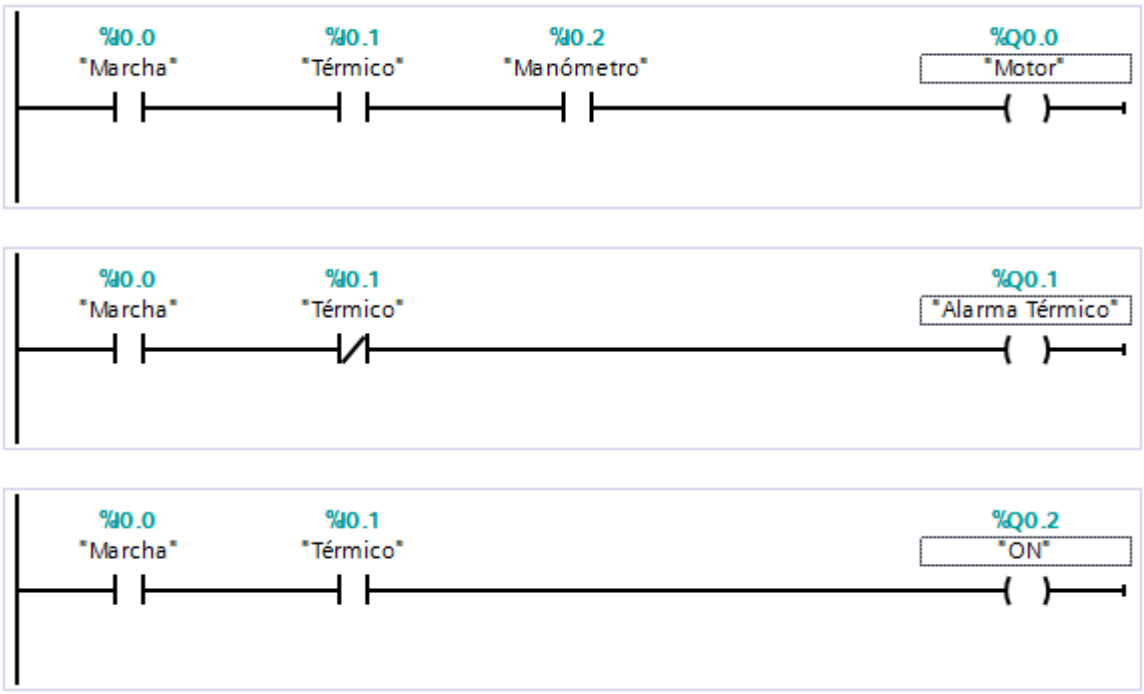
A saída Q0.1 activarase co interruptor de marcha e se saltou a protección térmica.

$$Q0.1 = I0.0 \cdot \overline{I0.1}$$

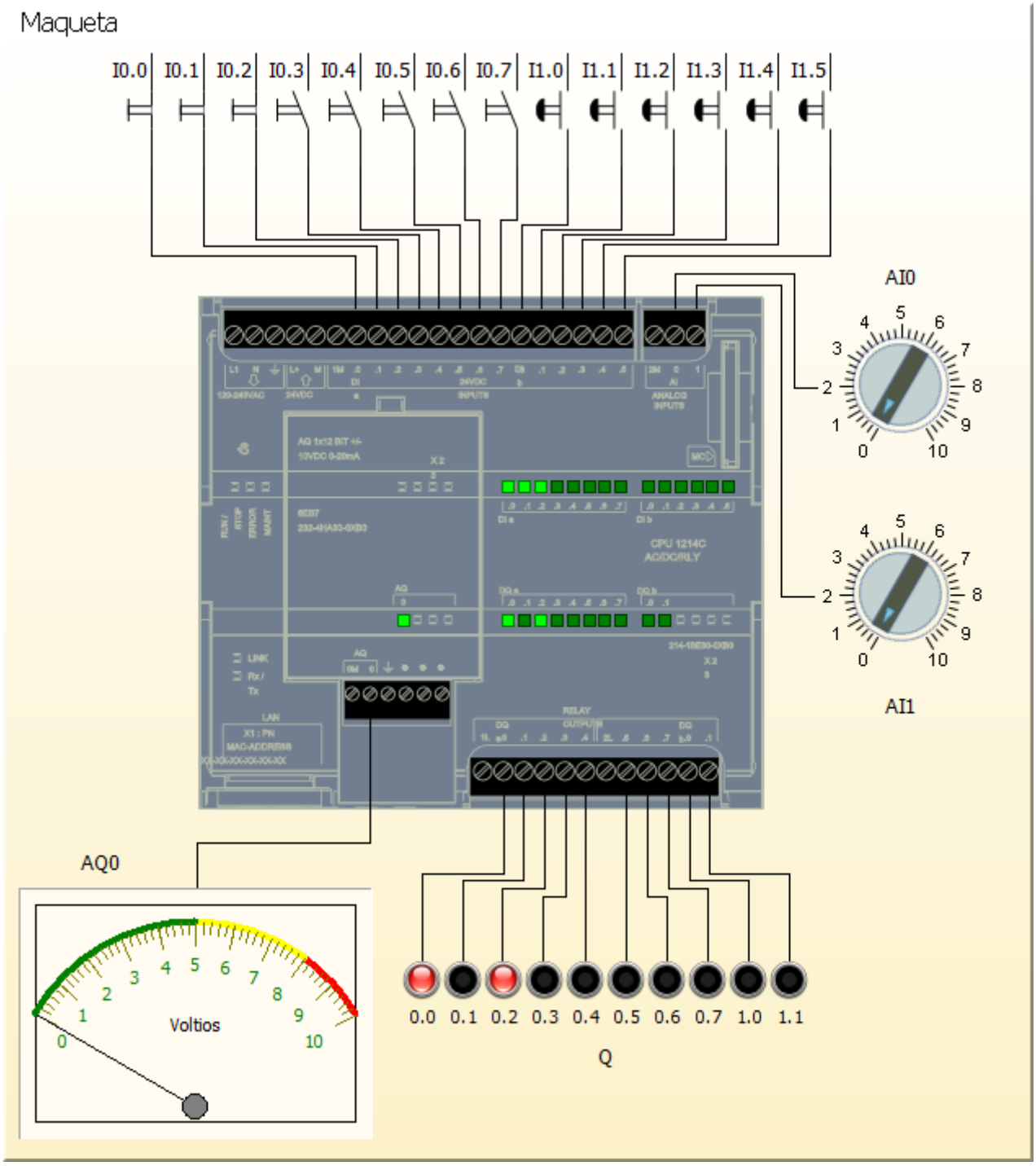
E a saída Q0.2 activarase co interruptor de marcha e se non saltou a protección térmica.

$$Q0.2 = I0.0 \cdot I0.1$$

Quedaríanos o programa como o seguinte.



Podemos comprobar no simulador **virtualmakTCP** o comportamento do sistema.



7.7.3 Instruccións lóxicas AND, OR e OR exclusiva en FUP e SCL

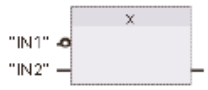
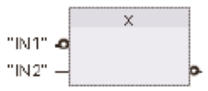
En programación FUP, a combinación serie e paralelo de contactos (AND, OR) convírtense en cadros nos que se indican as entradas e saídas. En linguaxe SCL tamén teñen a súa equivalencia.

FUP	SCL[1]	Descripción
	`out := in1 AND in2;`	Todas las entradas de un cuadro Y tienen que cumplirse para que la salida sea TRUE (verdadera).
	`out := in1 OR in2;`	Una entrada cualquiera de un cuadro O tiene que cumplirse para que la salida sea TRUE (verdadera).
	`out := in1 XOR in2;`	Un número impar de entradas de un cuadro O-exclusiva tiene que cumplirse para que la salida sea TRUE (verdadera).

[1] En SCL: El resultado de la operación debe asignarse a una variable para que pueda usarse en otra instrucción.

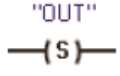
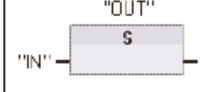
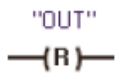
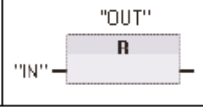
7.7.4 Instrucción de negación NOT

Tamén existe a posibilidade de utilizar un sinal negado empregando NOT nas tres linguaxes de programación (KOP, FUP e SCL).

KOP	FUP	SCL	Descripción
	 	`NOT`	En la programación FUP es posible arrastrar la función "Negar valor binario" desde la barra de herramientas "Favoritos" o desde el árbol de instrucciones y soltarla en una entrada o salida para crear un inversor lógico en ese conector del cuadro. El contacto NOT KOP invierte el estado lógico de la entrada de flujo de corriente. • Si no fluye corriente al contacto NOT, hay flujo de corriente en la salida. • Si fluye corriente al contacto NOT, no hay flujo de corriente en la salida.

7.7.5 Instruccións “Activar saída” e “Desactivar saída”

As bobinas vistas anteriormente actívanse e desactívanse segundo o sinal que se lle aplique. Se queremos activar unha bobina e deixala activada aínda que desapareza o sinal que a activou, empregaremos “Activar saída”. Neste caso necesitaremos dalgún xeito desactivar a saída ou quedará activada permanentemente.

KOP	FUP	SCL	Descripción
		No disponible	Si se activa S (Set) el valor de datos de la dirección OUT se pone a 1. Si S no es está activado, OUT no cambia.
		No disponible	Si se activa R (Reset), el valor de datos de la dirección de salida OUT se pone a 0. Si no se activa R, no se modifica OUT.

[1] En KOP y FUP: Estas instrucciones pueden disponerse en cualquier posición del segmento.

[2] En SCL: Es necesario escribir código para duplicar esta función en la aplicación.

Se queremos activar ou desactivar dunha sola vez un número “n” de bits empregaremos **SET_BF**.

KOP[1]	FUP	SCL	Descripción
"OUT" —(SET_BF)— "n"	"OUT" SET_BF EN N	No disponible	Cuando se activa SET_BF, el valor de datos 1 se asigna a "n" bits, comenzando en la dirección OUT. Si SET_BF no se activa, OUT no cambia.
"OUT" —(RESET_BF)— "n"	"OUT" RESET_BF EN N	No disponible	RESET_BF escribe el valor de datos 0 en "n" bits, comenzando en la dirección OUT. Si RESET_BF no se activa, OUT no cambia.

1 En KOP y FUP: Estas instrucciones sólo se pueden disponer en el extremo derecho de una rama.

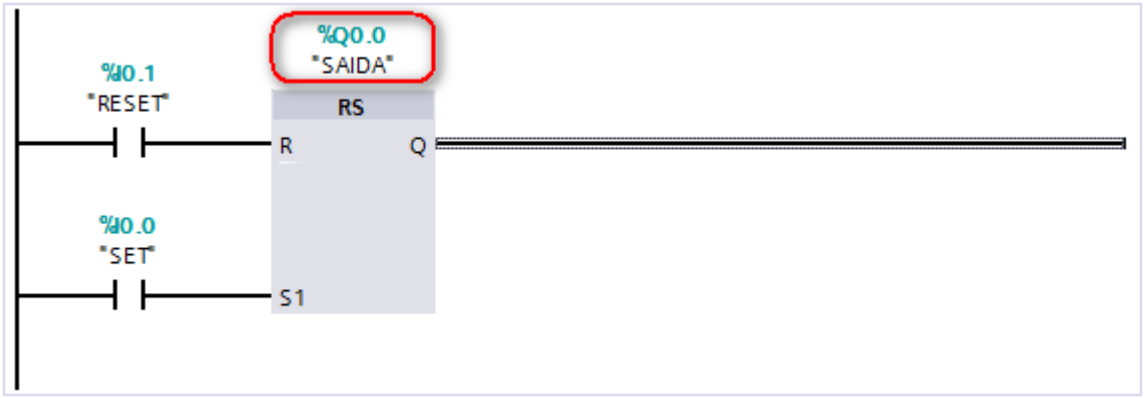
2 En SCL: Es necesario escribir código para duplicar esta función en la aplicación.

7.7.6 Báscula de activación/desactivación e de desactivación/activación

Existen funcións que permiten activar ou desactivar unha saída, aplicando dous sináis na entrada (o de activación e o de desactivación). Segundo desexemos prioridade á activación ou á desactivación empregaremos unha ou outra.

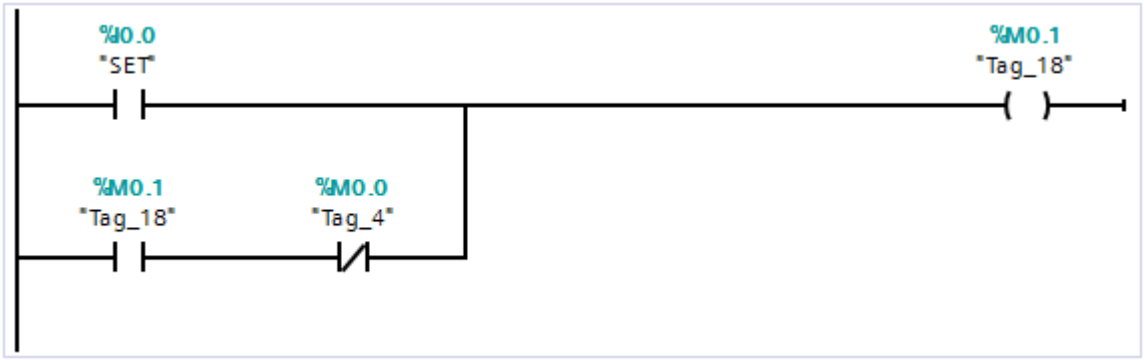
KOP / FUP	SCL	Descripción
"OUT" RS R Q S1	No disponible	RS es un flipflop en el que domina la activación. Si las señales de activación (S1) y desactivación (R) son verdaderas, la dirección de salida OUT se pone a 1.
"OUT" SR S Q R1	No disponible	SR es un flipflop en el que domina la desactivación. Si las señales de activación (S) y desactivación (R1) son verdaderas, la dirección de salida OUT se pone a 0.

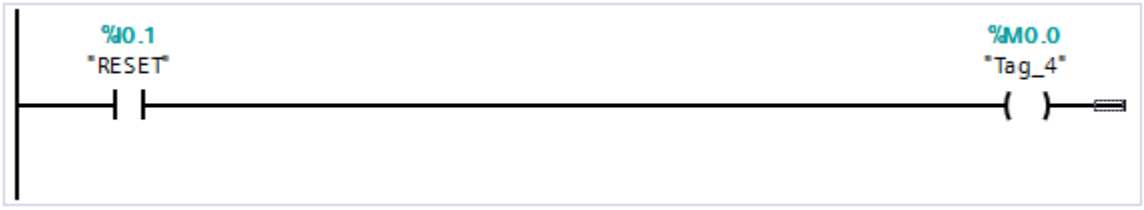
Nas funcións anteriores podemos colocar na parte superior a saída que pretendemos controlar.

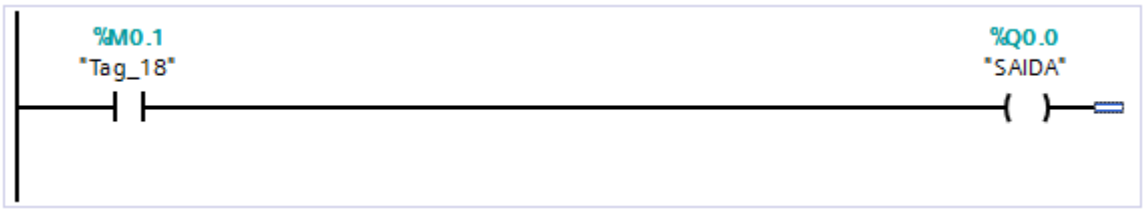


Na figura anterior activamos e desactivamos a saída Q0.0 coas entradas I0.0 e I0.1 respectivamente.

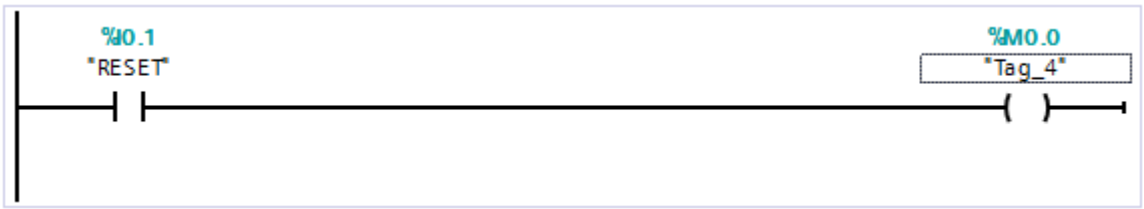
Un esquema equivalente dunha báscula RS sería o seguinte.

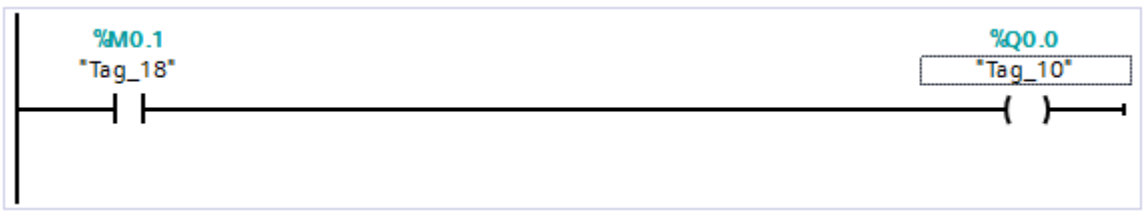






Un esquema equivalente dunha báscula SR sería o seguinte.





7.7.6.1 Exercicio 2

Dispoñemos dun depósito cos seguintes elementos de control:

- Un interruptor que detecta o nivel mínimo de líquido no depósito.
- Un interruptor que detecta o nivel máximo de líquido no depósito.
- Unha bomba que subministra líquido ao depósito.
- Un panel de mando con tres posicións. Manual, automático e fóra de servicio.
- Un relé térmico de protección da bomba.

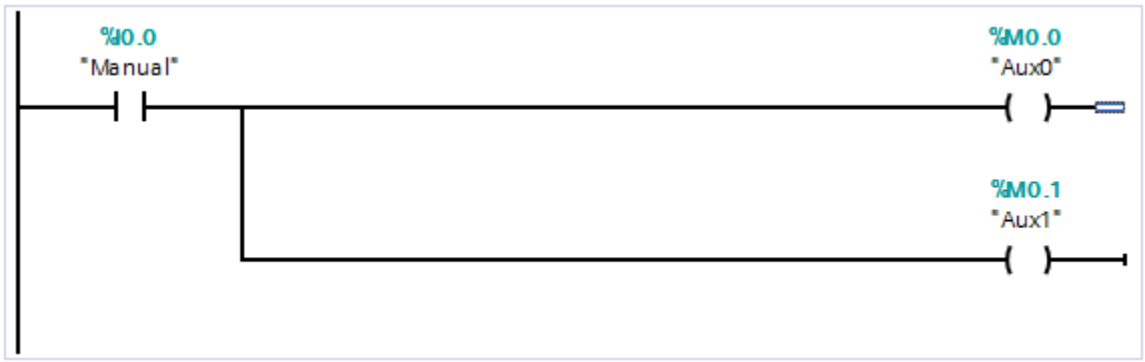
Deseñaremos un programa de acordo coas especificacións seguintes:

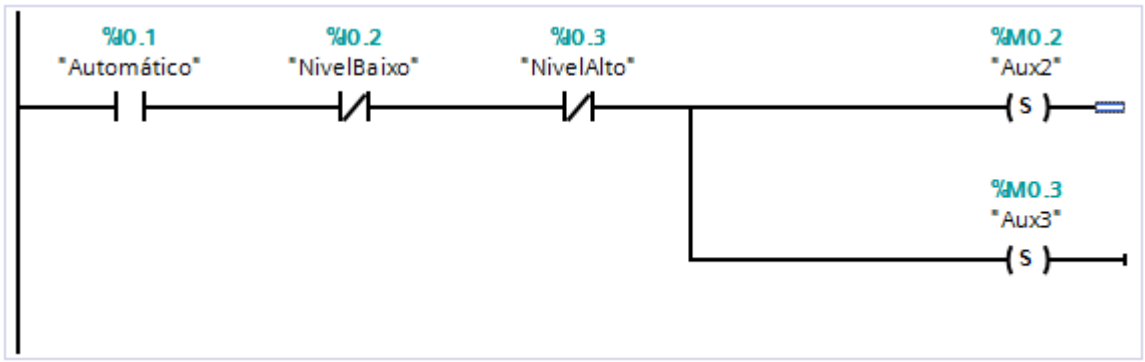
- Na posición Manual, a bomba funcionará permanentemente, independente dos valores dos interruptores de mínimo e máximo.
- Na posición Automático, a bomba procurará que o nivel de líquido estea entre os valores mínimo e máximo, para o cal conectarase cando se active o interruptor de mínimo e desactivarase cando se active o de máximo.
- Na posición fóra de servicio, a bomba non funciona.
- O relé térmico parará a bomba cando detecte unha temperatura elevada na bomba, en calquera das dúas posicións (manual, automático). Además acenderase unha lámpada de indicación de fallo.
- Cando a bomba está en marcha, débese iluminar a lámpada “Marcha”.

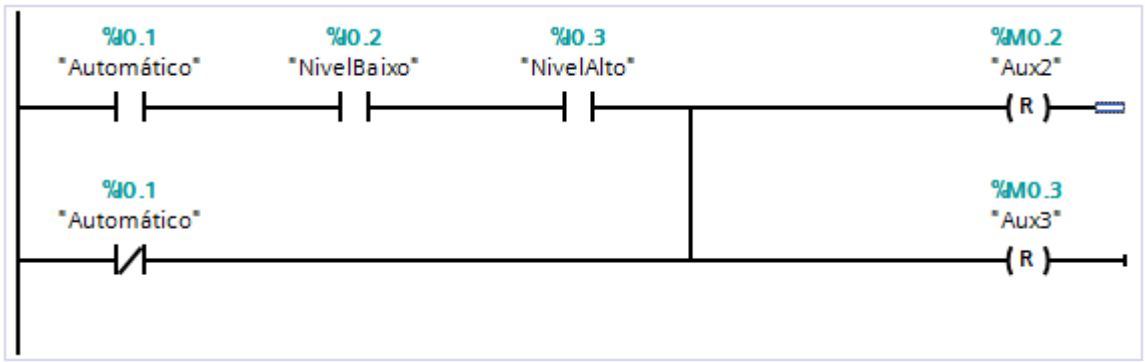
Elaboramos a táboa de entradas/saídas

Entradas	Descrición
I0.0	Interruptor en modo manual
I0.1	Interruptor en modo automático
I0.2	Detector nivel baixo
I0.3	Detector nivel alto
I0.7	Contacto auxiliar NC do relé térmico
Saídas	Descrición
Q0.0	Contactor da bomba
Q0.1	Sinalización de marcha
Q0.7	Sinalización de disparo térmico

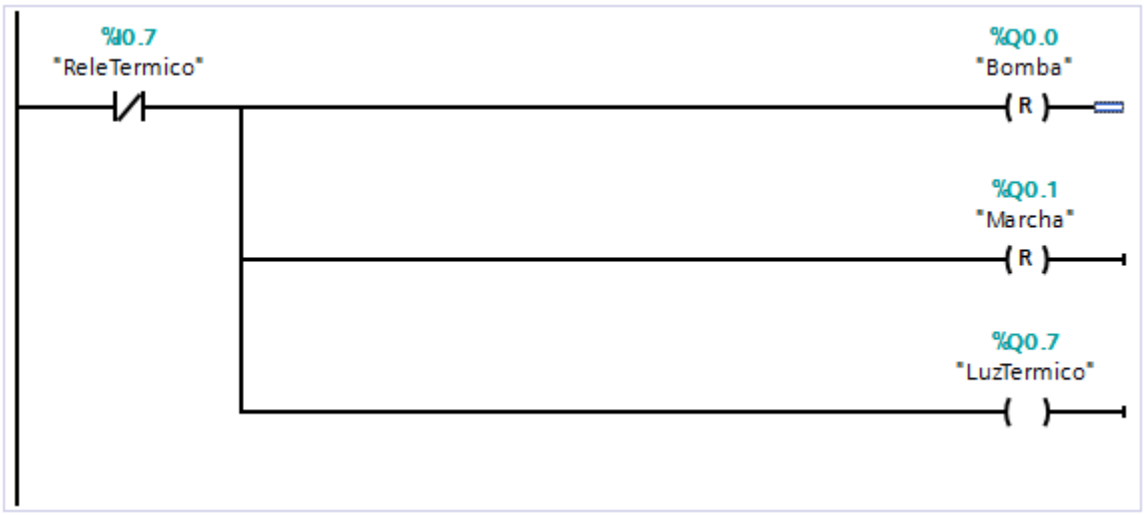
O programa podería ser:


%I0.0
"Manual"
%M0.0
"Aux0"
%M0.1
"Aux1"


%I0.1
"Automático"
%I0.2
"NivelBaixo"
%I0.3
"NivelAlto"
%M0.2
"Aux2"
S
%M0.3
"Aux3"
S


%I0.1
"Automático"
%I0.2
"NivelBaixo"
%I0.3
"NivelAlto"
%M0.2
"Aux2"
R
%I0.1
"Automático"
%M0.3
"Aux3"
R


%M0.0
"Aux0"
%Q0.0
"Bomba"
%M0.2
"Aux2"



Empezamos activando dúas marcas auxiliares (M0.0 e M0.1) coas que poñeremos en marcha a bomba e acenderemos a luz de marcha estando en modo manual. Empregamos a entrada I0.0 (Manual) para activalas.

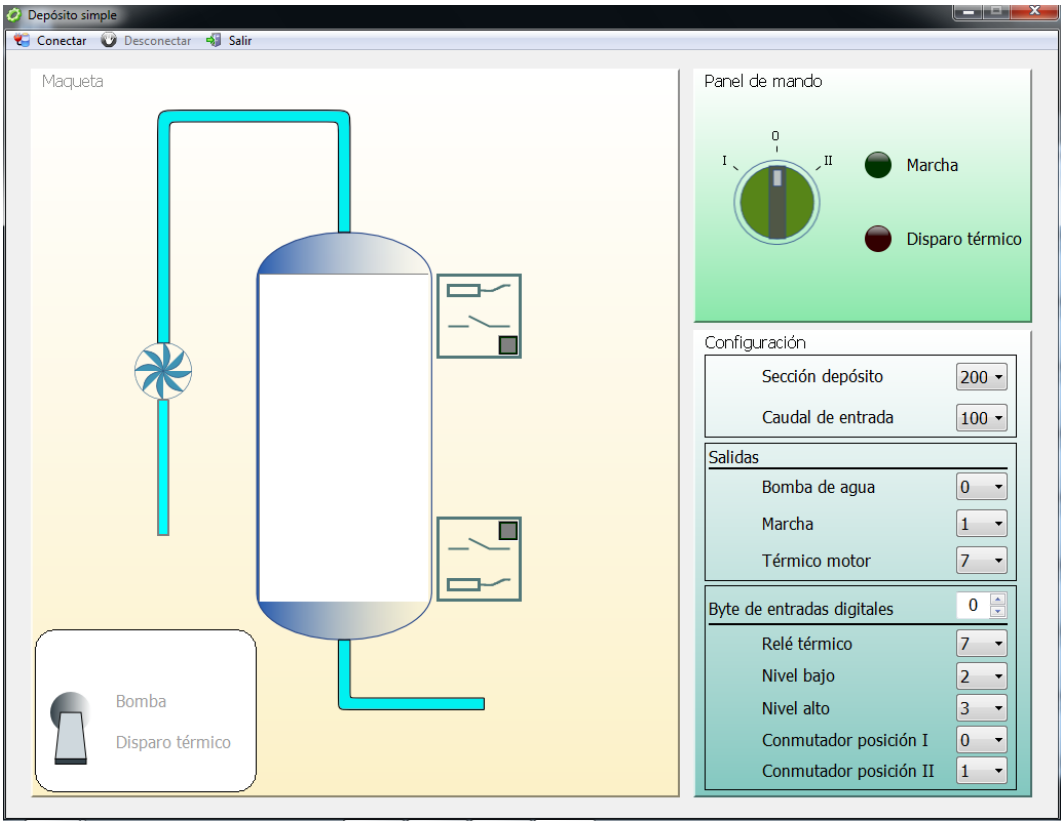
Utilizaremos outras dúas marcas auxiliares (M0.2 e M0.3) para o mesmo propósito pero traballando en modo automático. Por eso van en paralelo coas marcas anteriores para activar a bomba e a luz de marcha.

Estas marcas póñense a nivel alto cando se cumpra a condición de que o o selector de modo está en posición Automático (I0.1) e estea baleiro o depósito (condicións iniciais).

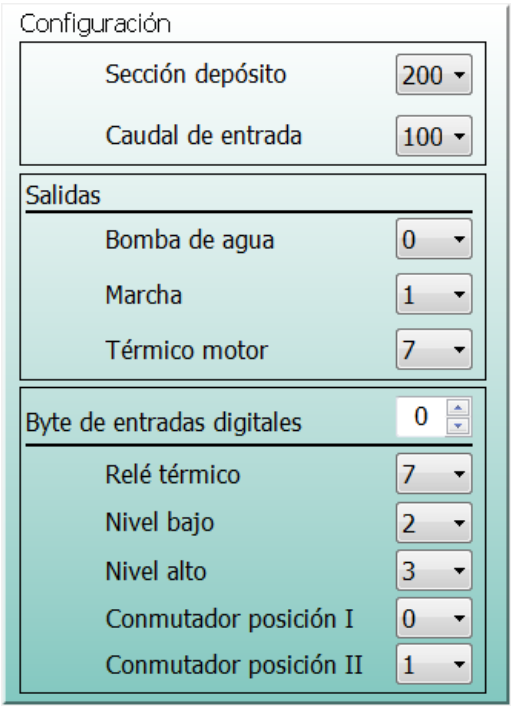
As mesmas marcas póñense a nivel baixo en dúas situacións. Por unha banda, estando en automático, se o depósito está cheo (I0.2 e I0.3 activos), ou ben cando saímos de modo automático.

Por último, en caso de que actúe a protección térmica do motor (I0.7), desactivamos o motor (Q0.0), a lámpada de marcha (Q0.1) e activamos a sinalización de disparo térmico (Q0.7).

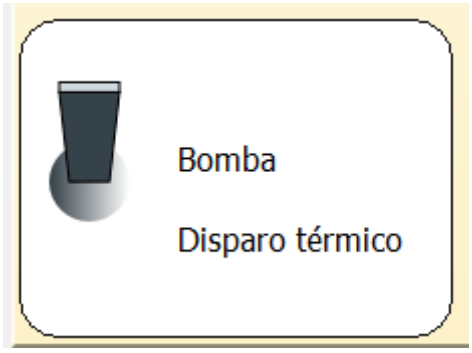
Podemos empregar **virtualmakTCP** coa maqueta do depósito simple para comprobar o funcionamento do programa.



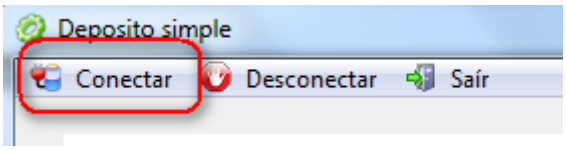
Para o correcto funcionamento, configuramos as entradas e saídas como nas especificacións do enunciado.



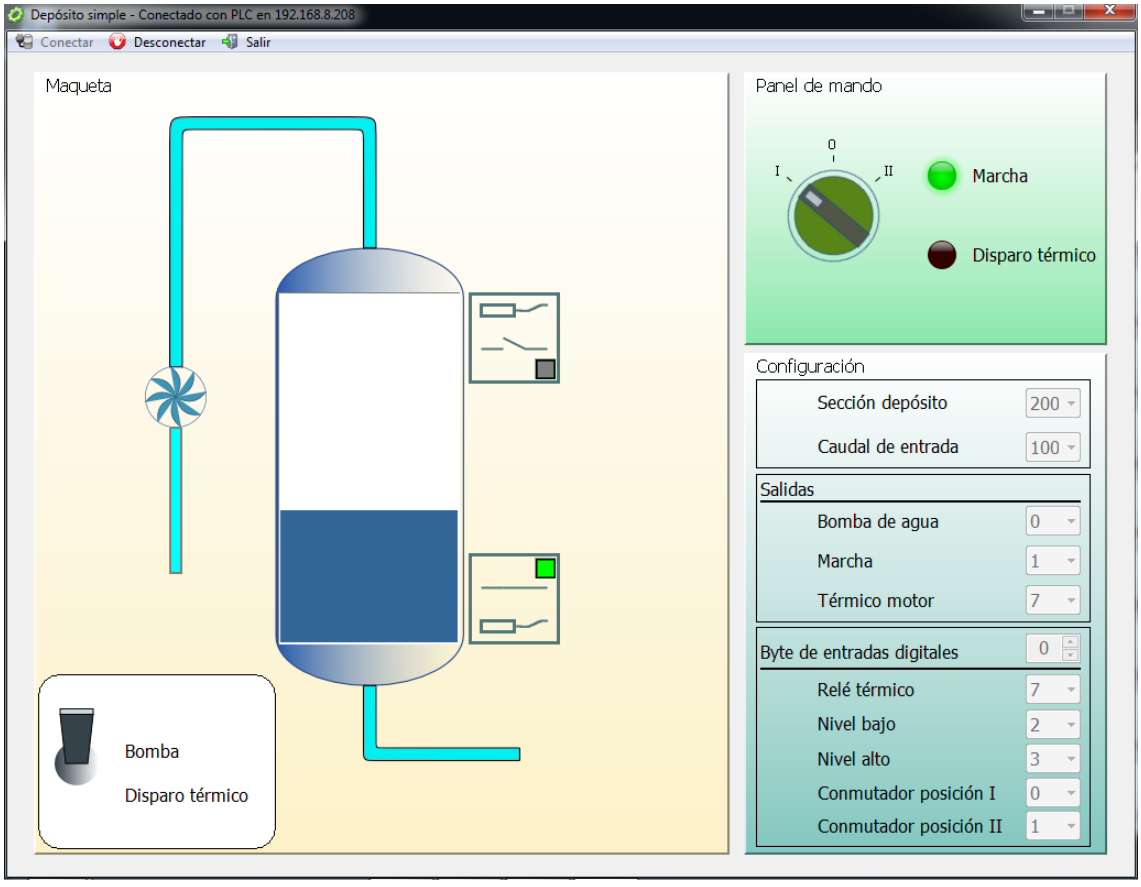
A continuación colocamos o Disparo térmico da bomba en posición de funcionamento.



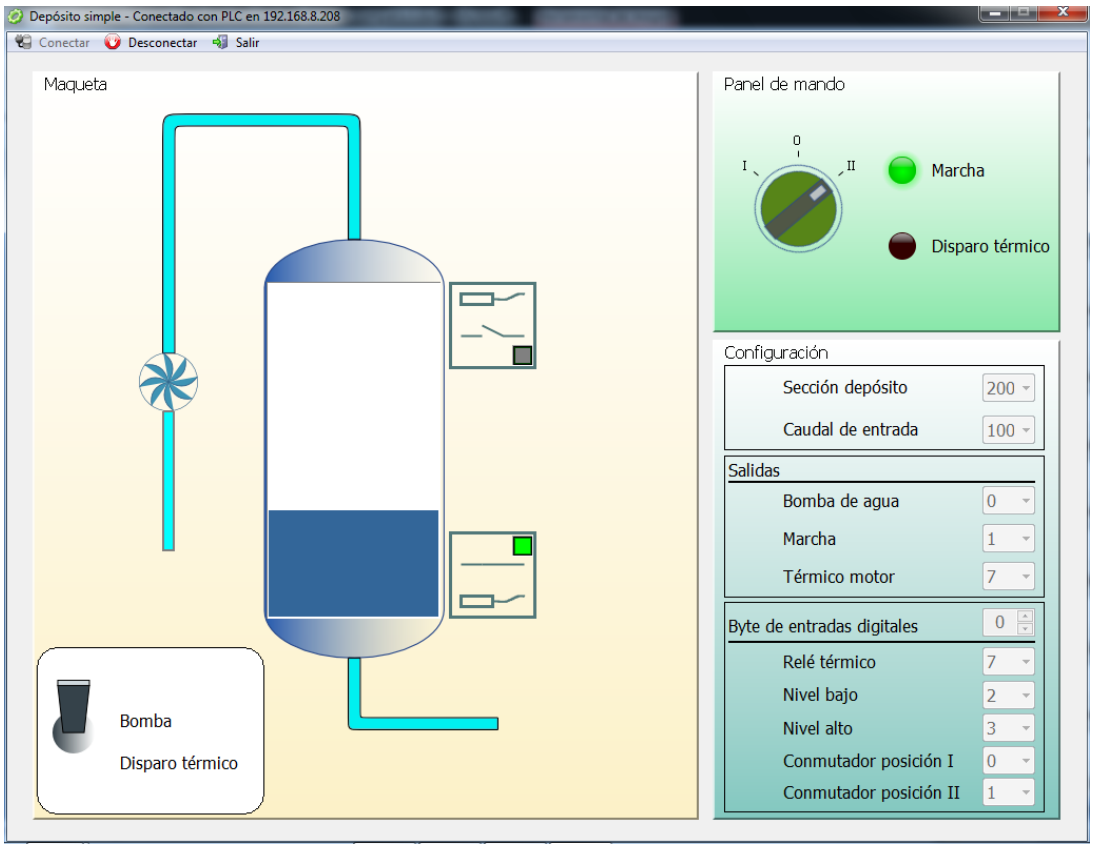
E conectamos co PLC.



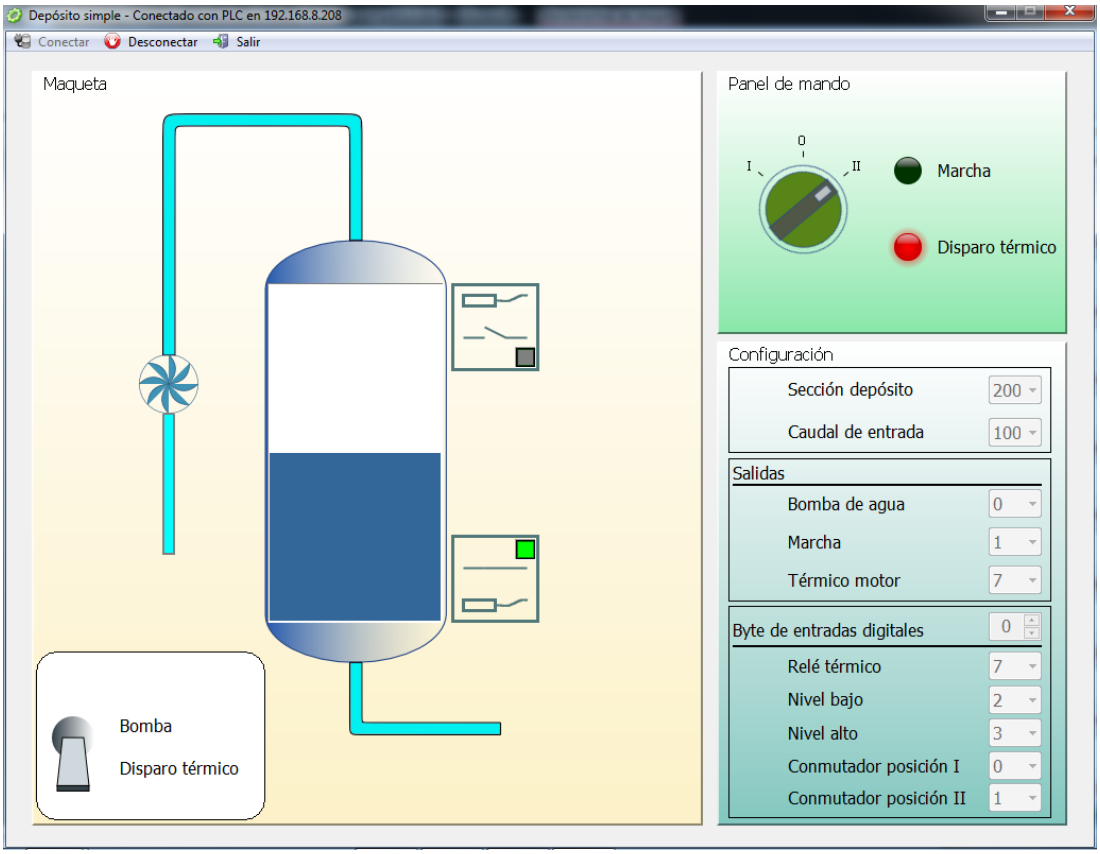
Probamos o modo de funcionamento Manual.



E o modo de funcionamento Automático.



Tamén podemos probar o disparo térmico.



7.7.7 Consulta de flancos ascendentes e descendentes

Nalgúns casos necesitamos que se detecte o cambio de estado dunha variable ben sexa de entrada ou de saída, de xeito que se produza un pulso no momento en que pasa de valer 0 a 1 (ou viceversa), pero non necesitamos que se estea detectando permanentemente o sinal. Para esto empregamos detección de flancos positivos (transición de 0 a 1) ou negativos (transición de 1 a 0).

KOP	FUP	SCL	Descripción
"IN" —|P|— "M_BIT"	"IN" P "M_BIT"	No disponible	KOP: El estado de este contacto es TRUE cuando se detecta un flanco ascendente (OFF a ON) en el bit "IN" asignado. El estado lógico del contacto se combina entonces con el estado de entrada del flujo de corriente para activar el estado de salida del flujo de corriente. El contacto P puede disponerse en cualquier posición del segmento, excepto al final de una rama. FUP: El estado lógico de la salida es TRUE (verdadero) cuando se detecta un flanco ascendente (OFF a ON) en el bit de entrada asignado. El cuadro P sólo se puede disponer al comienzo de una rama.
"IN" —|N|— "M_BIT"	"IN" N "M_BIT"	No disponible	KOP: El estado de este contacto es TRUE (verdadero) cuando se detecta un flanco descendente (ON a OFF) en el bit de entrada asignado. El estado lógico del contacto se combina entonces con el estado de entrada del flujo de corriente para activar el estado de salida del flujo de corriente. El contacto N puede disponerse en cualquier posición del segmento, excepto al final de una rama. FUP: El estado lógico de la salida es TRUE (verdadero) cuando se detecta un flanco descendente (ON a OFF) en el bit de entrada asignado. El cuadro N sólo se puede disponer al comienzo de una rama.
"OUT" —(P)— "M_BIT"	"OUT" P= "M_BIT"	No disponible	KOP: El bit asignado "OUT" es TRUE (verdadero) cuando se detecta un flanco ascendente (OFF a ON) en el flujo de corriente que entra a la bobina. El estado de entrada del flujo de corriente atraviesa la bobina como el estado de salida del flujo de corriente. La bobina P puede disponerse en cualquier posición del segmento. FUP: El bit asignado "OUT" es TRUE (verdadero) cuando se detecta un flanco ascendente (OFF a ON) en el estado lógico de la conexión de entrada del cuadro, o bien en la asignación del bit de entrada si el cuadro está ubicado al comienzo de una rama. El estado lógico de la entrada atraviesa el cuadro como el estado lógico de la salida. El cuadro P= puede disponerse en cualquier posición de la rama.
"OUT" —(N)— "M_BIT"	"OUT" N= "M_BIT"	No disponible	KOP: El bit asignado "OUT" es TRUE (verdadero) cuando se detecta un flanco descendente (ON a OFF) en el flujo de corriente que entra a la bobina. El estado de entrada del flujo de corriente atraviesa la bobina como el estado de salida del flujo de corriente. La bobina N puede disponerse en cualquier posición del segmento. FUP: El bit asignado "OUT" es TRUE cuando se detecta un flanco descendente (ON a OFF) en el estado lógico de la conexión de entrada del cuadro, o bien en la asignación del bit de entrada si el cuadro está ubicado al comienzo de una rama. El estado lógico de la entrada atraviesa el cuadro como el estado lógico de la salida. El cuadro N= puede disponerse en cualquier posición de la rama.

Observamos que todas as instruccións de detección de flanco levan asociada un bit de marcas (M_BIT) para almacenar o estado anterior do sinal que estamos a controlar. Os flancos detéctanse comparando o estado da entrada/saída co valor da marca auxiliar. Se a

situación indica un cambio da entrada/saída con respecto á marca, no sentido axeitado, notifícase o flanco activando a saída.

Hai que ter en conta que no primeiro ciclo de programa xa se fai unha comparación entre o sinal de entrada/saída e a marca, polo que deberemos inicializar correctamente os mesmos para evitar unha detección de flanco non desexada.

Por outra banda, o bit de marcas que empreguemos como auxiliar non debe ser empregado para outros fins no programa se non queremos falsear a lectura dos flancos.

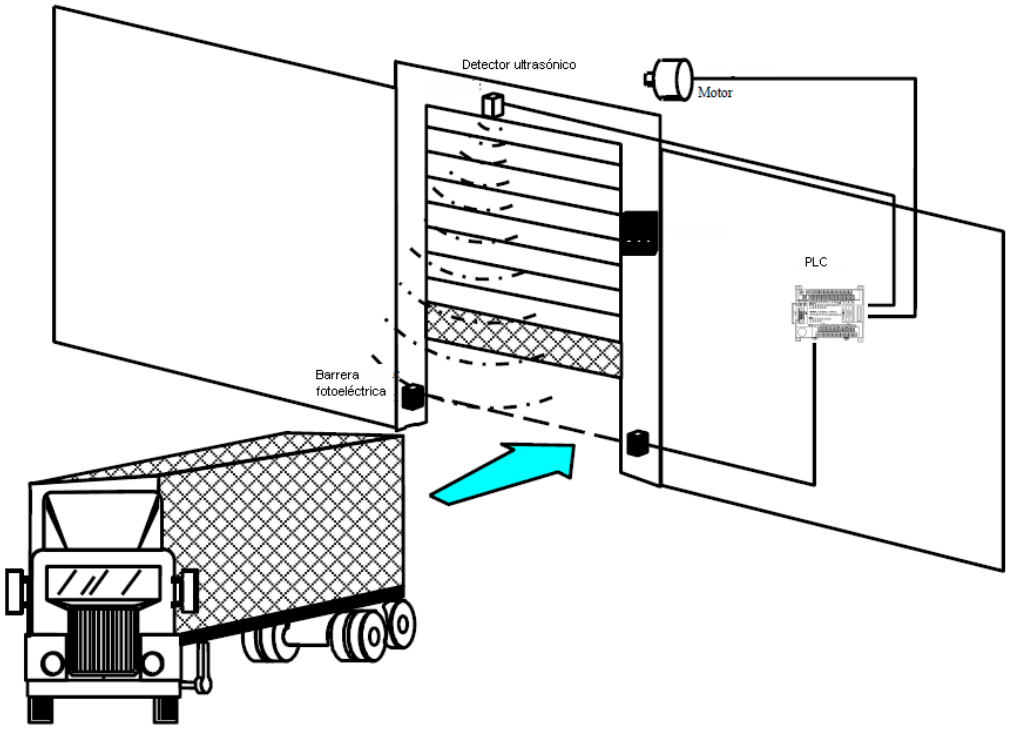
7.7.7.1 Exercicio 3

Preténdese automatizar a porta de entrada dun almacén ao que acceden vehículos.

No momento en que un camión se acerca é detectado por un detector de ultrasonidos, que inicia o proceso de apertura e peche da porta.

A continuación, o camión interrumpe unha barrera fotoeléctrica ata que acaba de entrar.

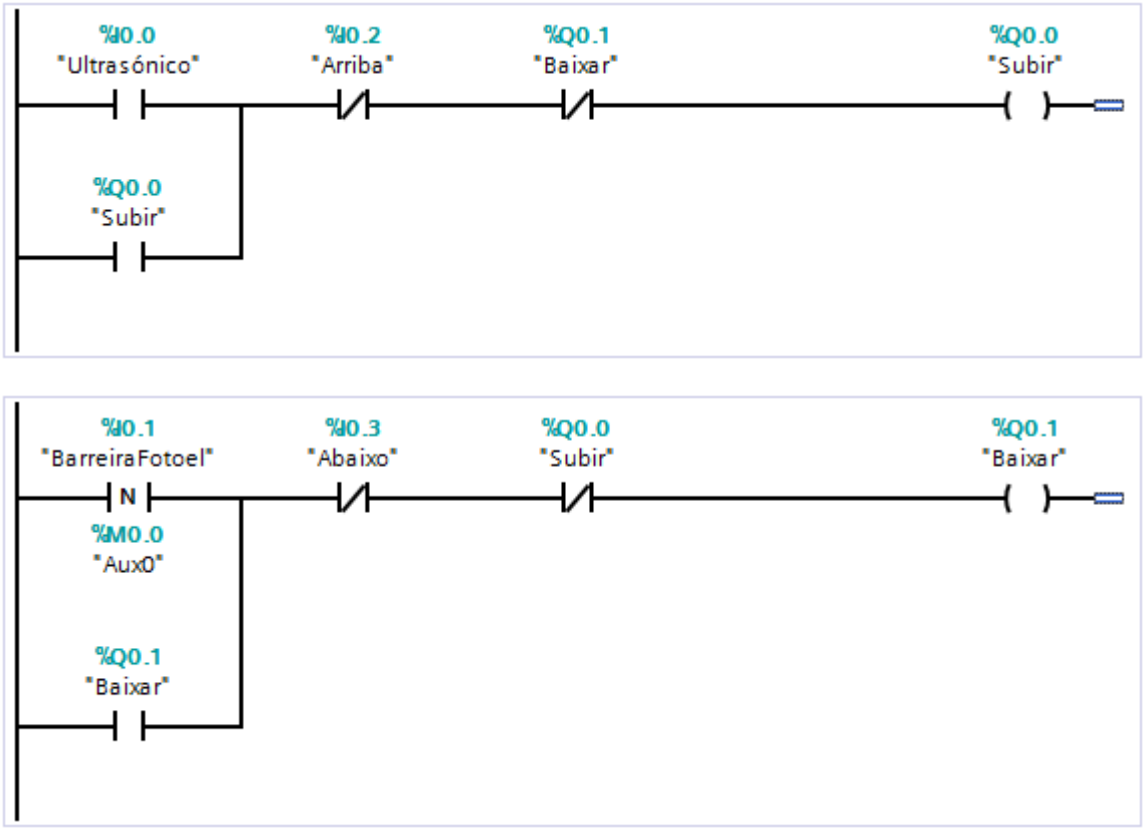
Unha vez que se detecta o flanco negativo na barreira fotoeléctrica, a porta péchase.



A táboa de entradas/saídas sería:

Entradas	Descrición
I0.0	Detector ultrasónico
I0.1	Barreira fotoeléctrica
I0.2	Detector porta arriba
I0.3	Detector porta abaixo
Saídas	Descrición
Q0.0	Contactor abrir porta
Q0.1	Contactor pechar porta

O programa sería:



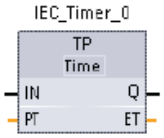
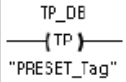
7.8 Temporizadores

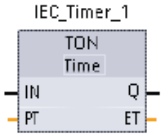
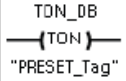
As instrucciónsde temporización empréganse para crear retardos no programa. Nos autómatas S7-1200 poden empregarse tantas instrucciónsdeste tipo como permita a memoria da CPU.

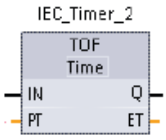
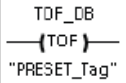
Cada temporizador crea un DB no que se gardan os datos de funcionamento do mesmo. Os DBs créanse automaticamente ao colocar o temporizador.

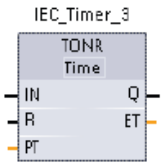
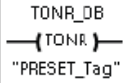
Basicamente os tipos de temporizadores son os catro que aparecen na seguinte táboa.

Cuadros KOP / FUP	Bobinas KOP	SCL	Descripción

		`"IEC_Timer_0_DB".TP(`
		`IN:=_bool_in_,`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			El temporizador TP genera un impulso con una duración predeterminada.

		`"IEC_Timer_0_DB".TON (`
		`IN:=_bool_in_,`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			El temporizador TON pone la salida Q a ON tras un tiempo de retardo predeterminado.

		`"IEC_Timer_0_DB".TOF (`
		`IN:=_bool_in_,`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			El temporizador TOF pone la salida Q a OFF tras un tiempo de retardo predeterminado.

		`"IEC_Timer_0_DB".TONR (`
		`IN:=_bool_in_,`
		`R:=_bool_in_`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			El temporizador TONR pone la salida Q a ON tras un tiempo de retardo predeterminado. El tiempo transcurrido se acumula a lo largo de varios periodos de temporización hasta que la entrada R inicializa el tiempo transcurrido.

O parámetro de entrada IN é o sinal para activar o temporizador.

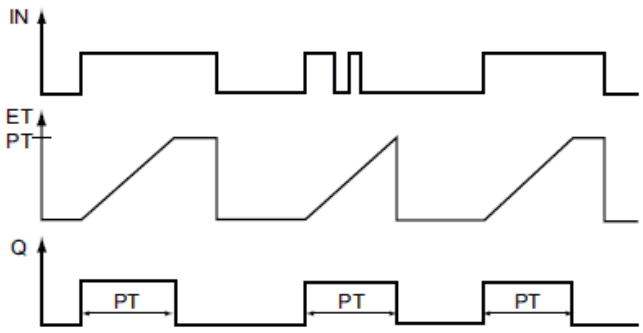
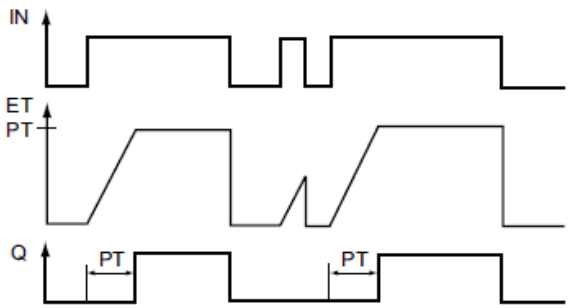
No parámetro PT escribimos o tempo desexado no formato IEC Time. Por exemplo, para representar unha temporización de 20 segundos escribiriamos T#20s.

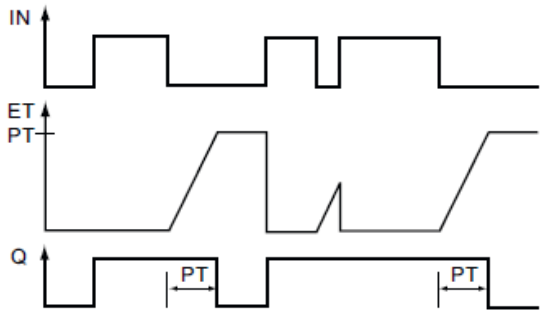
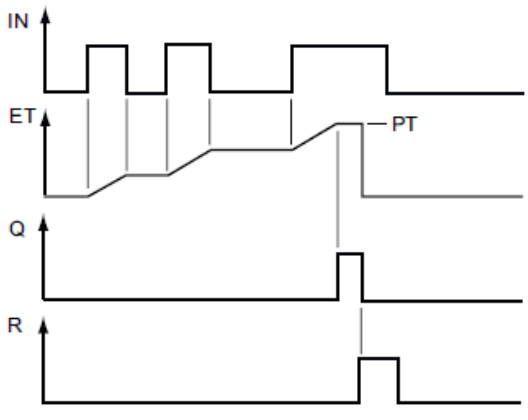
Como o tempo gárdase nun enteiro dobre con signo (32 bits) o tempo máximo que se pode utilizar para programar un temporizador é T#24d_20h_31m_23s_647ms que equivale a 2.147.483.647 ms.

É interesante ter en conta como se comportan as entradas IN e PT segundo o tipo de temporizador. Na seguinte táboa vemos un resumo.

Temporizador	Cambios en los parámetros de cuadro PT e IN y en los parámetros de bobina correspondientes
TP	• Un cambio de PT no tiene efecto alguno durante el funcionamiento del temporizador. • Un cambio de IN no tiene efecto alguno durante el funcionamiento del temporizador.
TON	• Un cambio de PT no tiene efecto alguno durante el funcionamiento del temporizador. • Si IN cambia a FALSE durante el funcionamiento del temporizador, éste se inicializará y se detendrá.
TOF	• Un cambio de PT no tiene efecto alguno durante el funcionamiento del temporizador. • Si IN cambia a TRUE durante el funcionamiento del temporizador, éste se inicializará y se detendrá.
TONR	• Un cambio de PT no tiene efecto alguno durante el funcionamiento del temporizador, pero sí cuando reanuda el contaje. • Si IN cambia a FALSE durante el funcionamiento del temporizador, éste se detendrá pero no se inicializará. Si IN vuelve a cambiar a TRUE, el temporizador comenzará a contar desde el valor de tiempo acumulado.

Na seguinte táboa temos os cronogramas de funcionamento dos catro tipos de temporizadores.

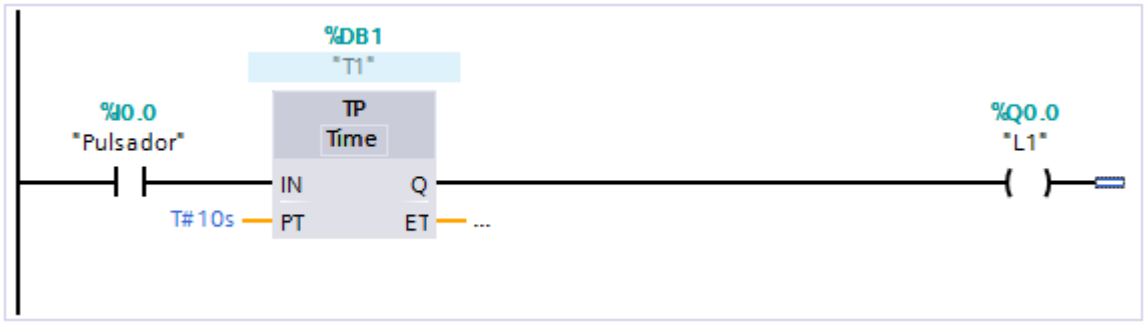
Temporizador	Cronograma
TP: Temporizador como impulso El temporizador TP genera un impulso con una duración predeterminada.	
TON: Temporizador como retardo a la conexión El temporizador TON pone la salida Q a ON tras un tiempo de retardo predeterminado.	

Temporizador	Cronograma
TOF: Temporizador como retardo a la desconexión El temporizador TOF pone la salida Q a OFF tras un tiempo de retardo predeterminado.	
TONR: Temporizador como retado a la conexión El temporizador TONR pone la salida Q a ON tras un tiempo de retardo predeterminado. El tiempo transcurrido se acumula a lo largo de varios periodos de temporización hasta que la entrada R inicializa el tiempo transcurrido.	

7.8.1.1 Exercicio 4

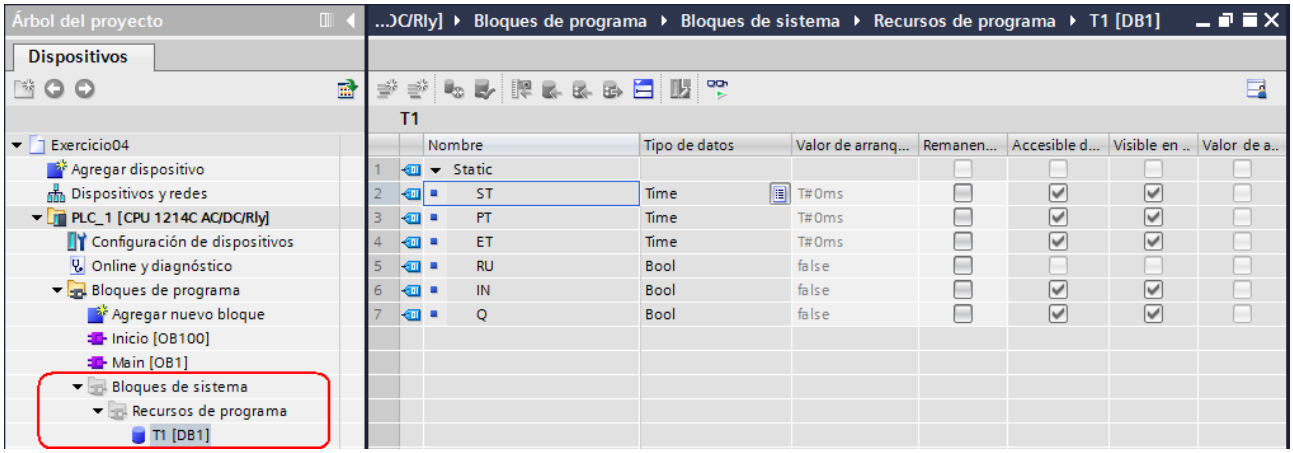
Acender unha lámpada durante 10s cando se pulse un pulsador. A lámpada apagarase soa.

O programa sería:



Nótese que se crea un DB asociado ao T1 no que se gardan os datos do temporizador. Este DB atópase en **Bloques de sistema. Recursos de programa**.

Facendo dobre click sobre o mesmo vemos a estructura que contén.

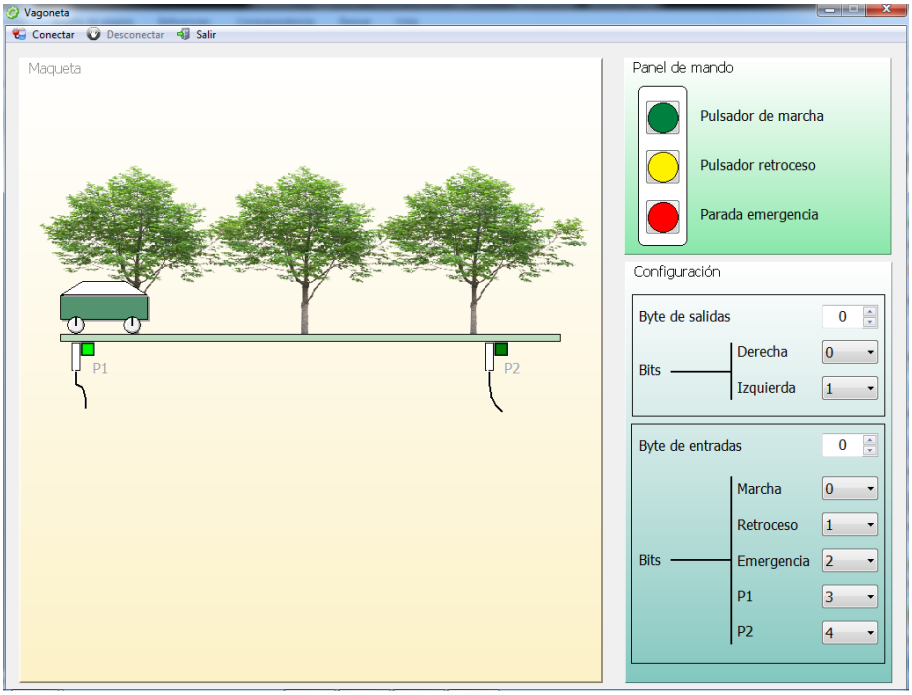


7.8.1.2 Exercicio 5

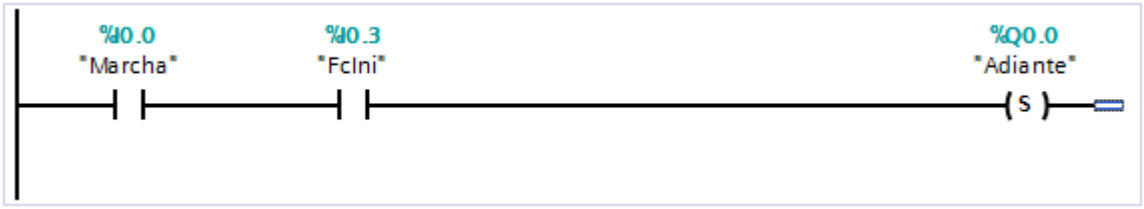
Unha vagoneta acciónase mediante un pulsador que a pon en marcha, avanzando ata unha posición final. Nesta posición final agardará 15 segundos ata que se lle cargue un material e voltará á posición de partida.

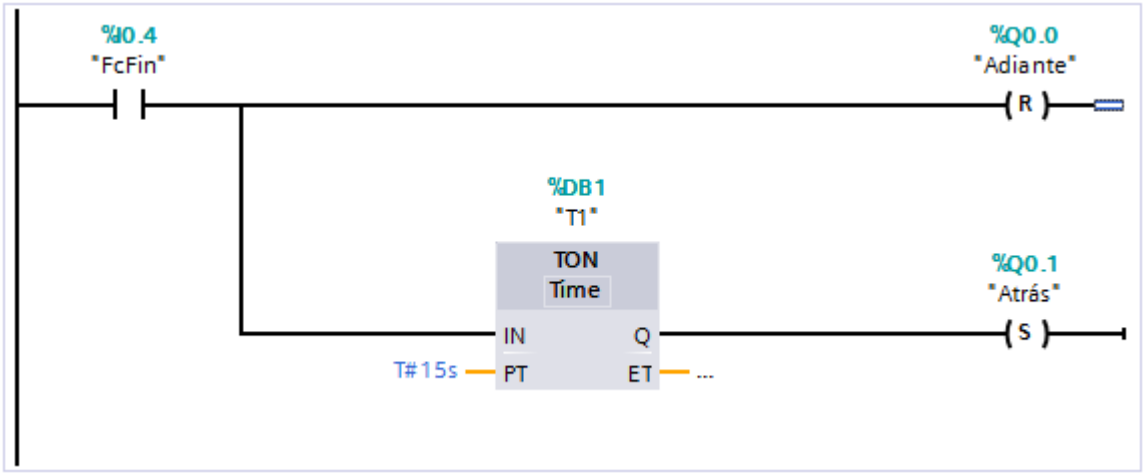
Entradas	Descrición
I0.0	Pulsador de marcha
I0.3	Detector posición inicial
I0.4	Detector posición final
Saídas	Descrición
Q0.0	Motor vagoneta dereita
Q0.1	Motor vagoneta esquerda

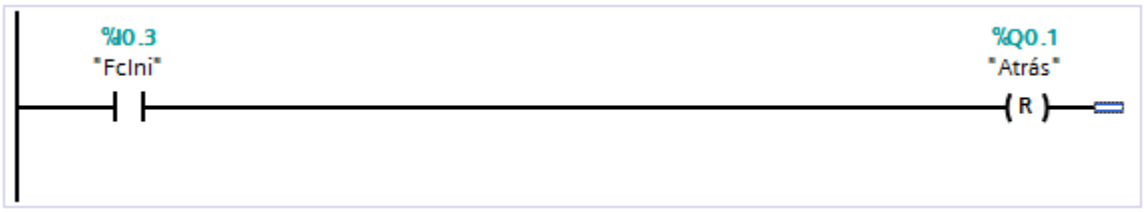
Podemos empregar virtualmakTCP para visualizar o proceso.



Un programa sinxelo empregando un temporizador á conexión (TON) sería o seguinte:







Exercicio 5_2: Repetir este exercicio coa condición de que o ciclo se ejecute continuamente ata que se accione o pulsador amarelo. Nese momento rematará o ciclo que se estea a executar, a vagoneta voltará á posición inicial e quedará ahí.

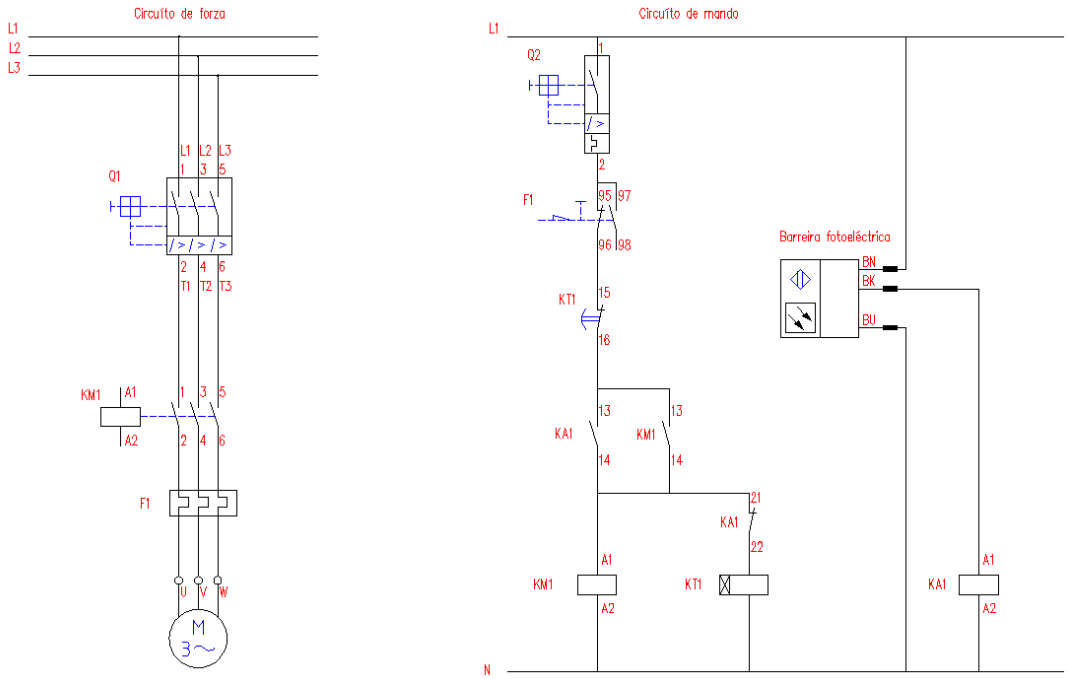
Exercicio 5_3: Repetir o exercicio anterior, engadíndolle además a posibilidade de que, ao accionar o botón vermello, estea onde estea o proceso, retorne á posición inicial.

7.8.1.3 Exercicio 6

Deseñar o programa para automatizar unha escaleira mecánica que se acciona mediante un motor trifásico asíncrono. O automatismo deberá cumplir as seguintes condicións:

- A escaleira deberá funcionar só cando sexa empregada por unha ou máis persoas.
- Debe funcionar o tempo suficiente para que faga o percorrido completo ata que sexa abandonada pola última persoa, de xeito que, se unha persoa se atopa subindo e outra empeza a subir, o temporizador reinicarase para que poida subir esa segunda persoa.
- O motor estará protexido de curtocircuítos e sobrecargas.

Un esquema de forza e mando convencional sería o seguinte.



No esquema anterior, no circuíto de forza, observamos que o motor está accionado por un interruptor (Q1). Dispoñemos dun contactor que pon en marcha ou para o mesmo (KM1) e un relé térmico de protección (F1).

No circuíto de mando temos un magnetotérmico unipolar (Q2), os contactos do térmico do motor (F1) e tres bobinas. A do contactor principal (KM1) a dun contactor auxiliar (KA1) e a do temporizador (KT1). Ademáis observamos un detector fotoeléctrico que acciona o contactor auxiliar KA1.

O funcionamento é o seguinte: Cando unha persoa se acerca á barreira fotoeléctrica, acciónase KA1, facendo que se abra primeiro o contacto auxiliar (21-22) e pechando o contacto auxiliar (13-14). Deste xeito empeza a andar o motor (KM1) e a persoa empeza a subir, abandonando a fotocélula e desactivando polo tanto KA1.

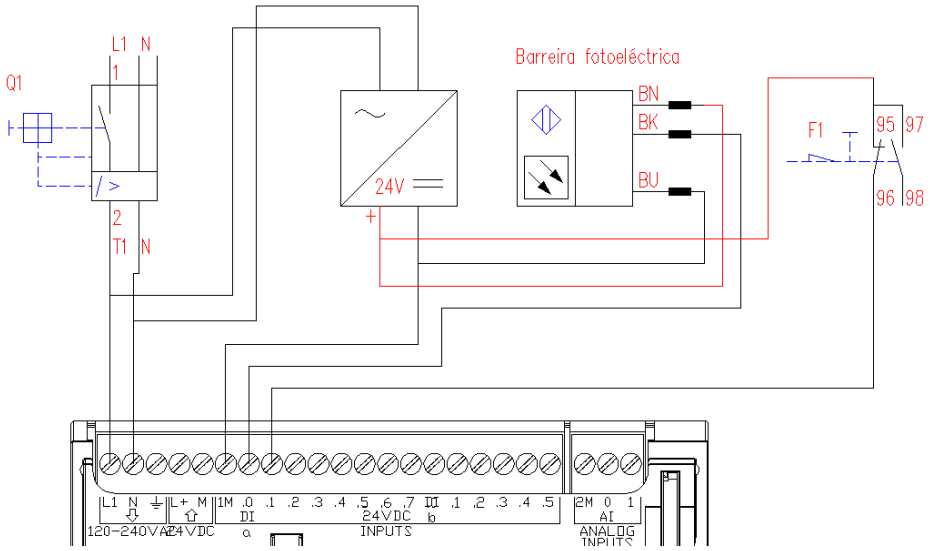
O motor seguirá funcionando porque hai un contacto auxiliar (13-14) que o mantén alimentado.

Por outra banda, despois de que a persoa abandona a barreira fotoeléctrica, e o contactor KA1 volve ao reposo, o contacto (21-22) do mesmo péchase e empeza a contar o tempo necesario para que a persoa chegue ao final da escaleira.

Ao chegar ao final do tempo establecido por KT1, ábrese o contacto (15-16) e para o sistema.

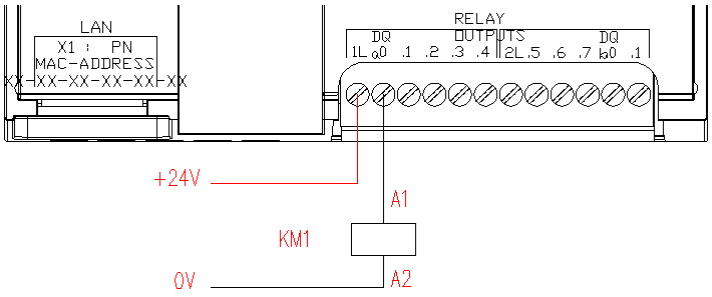
Se, mentres está a subir unha persoa, chega outra á barreira, acciona de novo KA1, abrindo o contacto (21-22) do mesmo e reseteando, polo tanto KT1.

Empregando un autómata para controlar o sistema teriamos nas entradas un esquema como o seguinte.



No esquema anterior temos un interruptor que alimenta o autómata. Unha fonte de alimentación de 230V CA a 24V CC para alimentar a barreira fotoeléctrica e o contacto do relé térmico do motor. A barreira fotoeléctrica e o contacto do térmico.

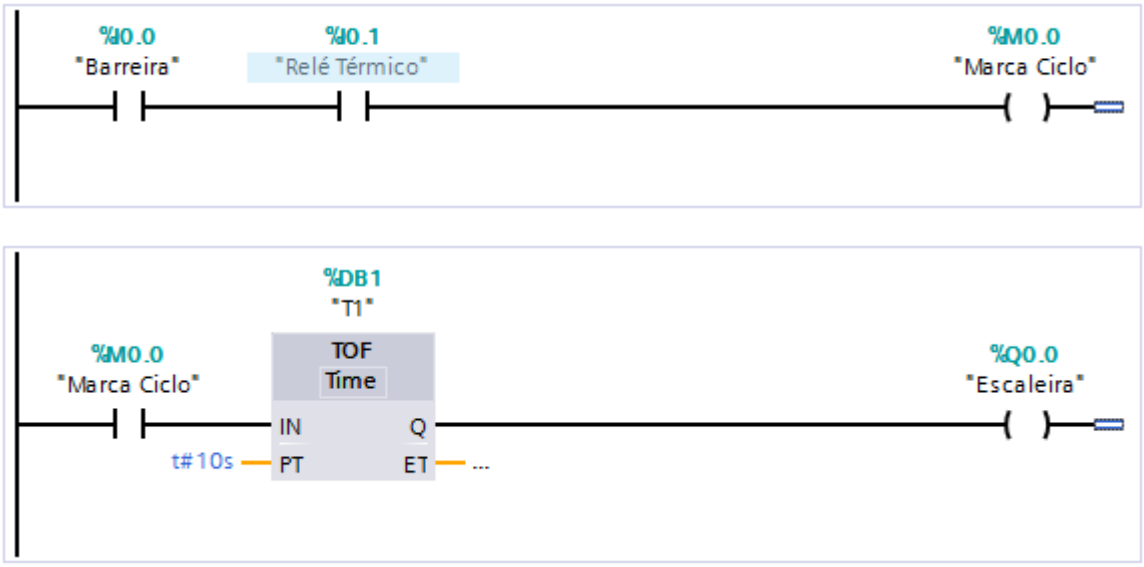
Nas saídas teriamos un esquema como o seguinte.



A táboa de entradas/saías sería:

Entradas	Descrición
I0.0	Barreira fotoeléctrica
I0.1	Contacto relé térmico (NC)
Saídas	Descrición
Q0.0	Motor escaleira mecánica

Unha posible traducción do automatismo clásico a un programa de autómata sería o seguinte:



7.9 Contadores

Empréganse para contar eventos tanto internos como externos no autómata. Existen os contadores ascendentes, descendentes e bidireccionais.

Na táboa seguinte vemos os tres.

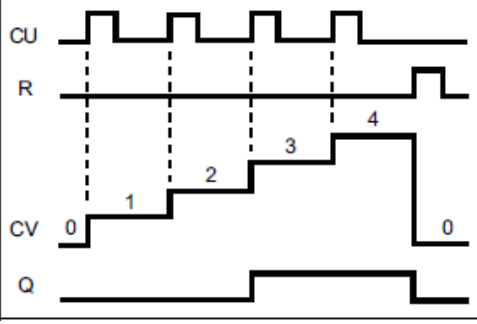
KOP / FUP	SCL	Descripción
"Counter name" CTU SInt CU Q R CV PV	"IEC_Counter_0_DB".CTU(CU:=_bool_in, R:=_bool_in, PV:=_int_in, Q=>_bool_out, CV=>_int_out);	Las instrucciones con contadores se utilizan para contar eventos del programa internos y eventos del proceso externos. Todo contador utiliza una estructura almacenada en un bloque de datos para conservar sus datos. El bloque de datos se asigna al colocar la instrucción de contaje en el editor. • CTU es un contador ascendente • CTD es un contador descendente • CTUD es un contador ascendente/descendente
"Counter name" CTD SInt CD Q LOAD CV PV	"IEC_Counter_0_DB".CTD(CD:=_bool_in, LD:=_bool_in, PV:=_int_in, Q=>_bool_out, CV=> int out);	
"Counter name" CTUD SInt CU QU CD QD R CV LOAD PV	"IEC_Counter_0_DB".CTUD(CU:=_bool_in, CD:=_bool_in, R:=_bool_in, LD:=_bool_in, PV:=_int_in, QU=>_bool_out, QD=>_bool_out, CV=> int out);	

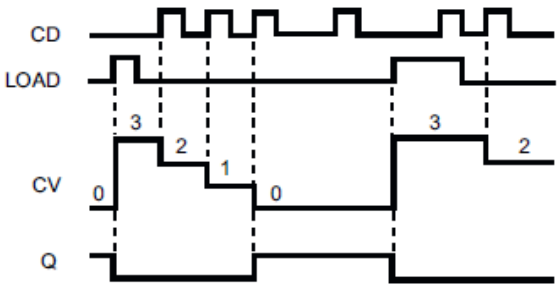
O significado das entradas e saídas dos contadores son os seguintes:

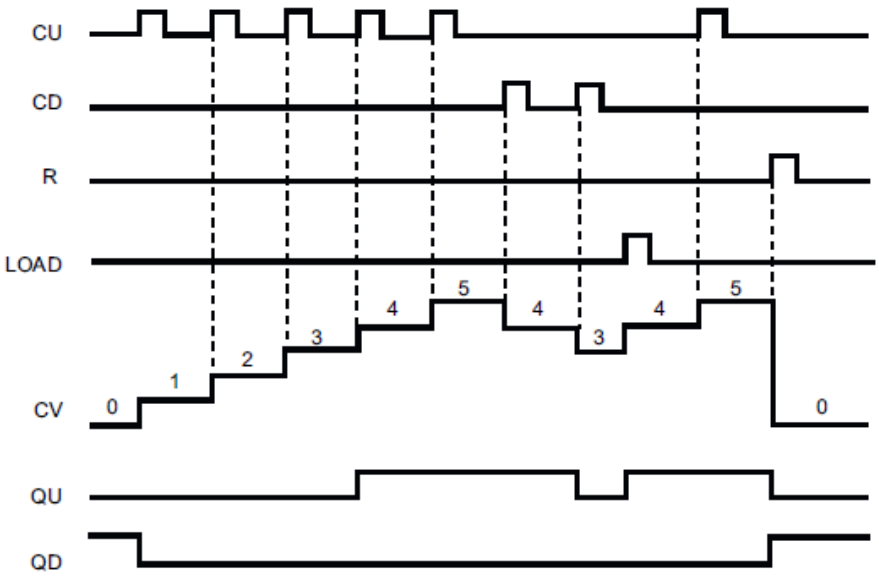
Parámetro	Tipo de datos	Descrición
CU, CD	Bool	Contaxe ascendente ou descendente en incrementos de un en un.
R (CTU, CTUD)	Bool	Entrada para poñer a cero o valor do contador.
LOAD (CTD, CTUD)	Bool	Entrada para cargar o valor predeterminado no contador.
PV	SInt, Int, DInt, USInt, UInt, UDInt	Valor de contaxe predeterminado.
Q, QU	Bool	Toma o valor verdadeiro cando o valor do contador (CV) é maior ou igual que o valor predeterminado (PV).
QD	Bool	Toma o valor verdadeiro se o valor do contador (CV) é menor ou igual que cero.
CV	SInt, Int, DInt, USInt, UInt, UDInt	Valor actual do contador

O rango numérico de valores de contaxe depende do tipo de datos seleccionado. Se o valor de contaxe é un enteiro sen signo, é posible contar cara atrás ata cero ou cara adiante ata o límite do rango. Se o valor de contaxe é un enteiro con signo, é posible contar cara atrás ata o límite de enteiro negativo e contar cara adiante ata o límite do enteiro positivo.

Nas seguinte táboas vemos cronogramas dos tres tipos de contadores.

Contador	Operación
El contador CTU incrementa en 1 cuando el valor del parámetro CU cambia de 0 a 1. El cronograma de CTU muestra el manejo con un valor de contaje de entero sin signo (donde PV = 3). • Si el valor del parámetro CV (valor de contaje actual) es superior o igual que el del parámetro PV (valor de contaje predeterminado), el parámetro de salida del contador Q = 1. • Si el valor del parámetro de desactivación R cambia de 0 a 1, el valor de contaje actual se pone a 0.	

Contador	Operación
El contador CTD decrementa en 1 cuando el valor del parámetro CD cambia de 0 a 1. El cronograma de CTD muestra el manejo con un valor de contaje de entero sin signo (donde PV = 3). • Si el valor del parámetro CV (valor de contaje actual) es inferior o igual a 0, el parámetro de salida del contador Q = 1. • Si el valor del parámetro LOAD cambia de 0 a 1, el valor del parámetro PV (valor predeterminado) se carga en el contador como nuevo CV (valor de contaje actual).	

Contador	Operación
El contador CTUD incrementa o decrementa en 1 en una transición de 0 a 1 de las entradas de contaje ascendente (CU) o descendente (CD). El cronograma muestra el funcionamiento de un contador CTUD con un valor de contaje de entero sin signo (donde PV = 4). • Si el valor del parámetro CV es superior o igual que el del parámetro PV, el parámetro de salida del contador QU = 1. • Si el valor del parámetro CV es inferior o igual a 0, el parámetro de salida del contador QD = 1. • Si el valor del parámetro LOAD cambia de 0 a 1, el valor del parámetro PV se carga en el contador como nuevo CV. • Si el valor del parámetro de reset R cambia de 0 a 1, el valor de contaje actual se pone a 0.	

7.9.1.1 Exercicio 7

Repetir o exercicio 5 (vagoneta) de xeito que o ciclo de carga e descarga da vagoneta se repita 4 veces e logo pare.

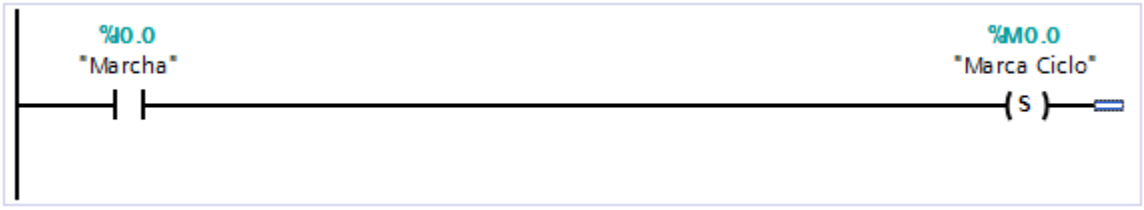
Empregaremos un contador para levar conta do número de veces que avanzou a vagoneta.

A táboa de entradas/saídas sería a mesma do exercicio 5.

Entradas	Descrición
I0.0	Pulsador de marcha
I0.3	Detector posición inicial
I0.4	Detector posición final
Saídas	Descrición
Q0.0	Motor vagoneta dereita
Q0.1	Motor vagoneta esquerda

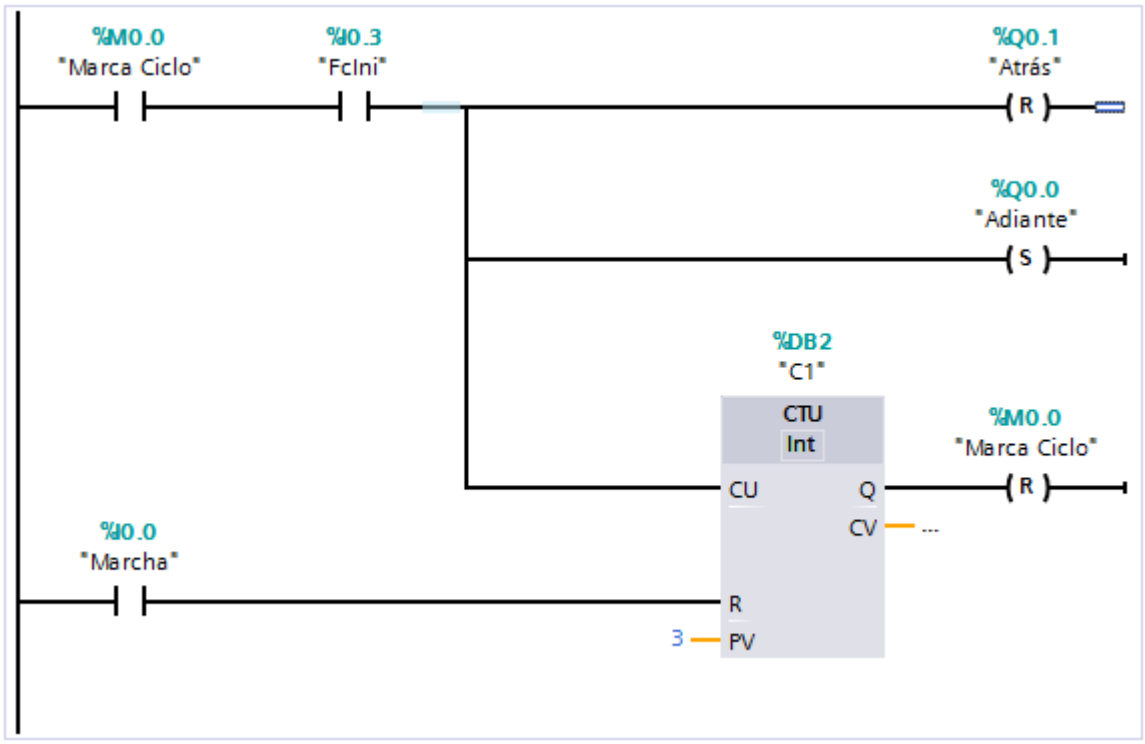
Unha posible solución axudándonos das marcas sería:

Co pulsador de marcha activamos unha marca de ciclo (M0.0). Mentres esta marca estea activa, estarase a repetir o ciclo.

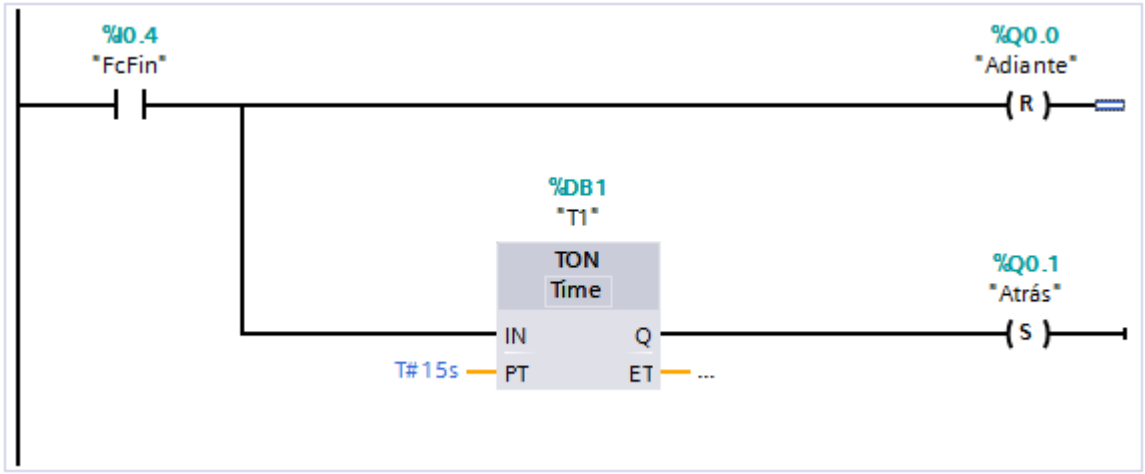


Coa marca de ciclo activada e co final de carreira do inicio resetamos a marcha atrás (Q0.1) e activamos a marcha adiante (Q0.0). Tamén incrementamos o contador, pero hai que observar que o contador non se activa na primeira pulsación do pulsador de marcha (I0.0) porque tamén o estamos a empregar para resetealo, logo no primeiro ciclo de programa non lle dará tempo a incrementar o contador.

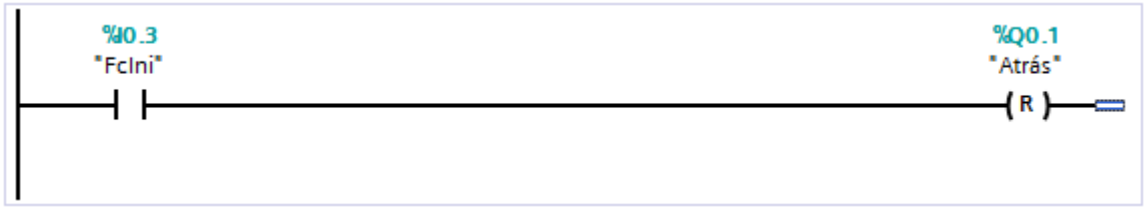
Por outra banda, o contador fixarase nun valor PV a 3 e non a 4 como poderiamos pensar. Esto é porque cando o contador conte tres ciclos (a vagoneta terá retornado 3 veces ao inicio), poñerá a reset a marca de ciclo (M0.0) pero xa teremos activado Adiante (Q0.0) para que faga o último percorrido.



Seguimos a ter o temporizador como no exercicio anterior.



E por último, paramos a vagoneta no último retroceso, cando xa a marca de ciclo non está activa.

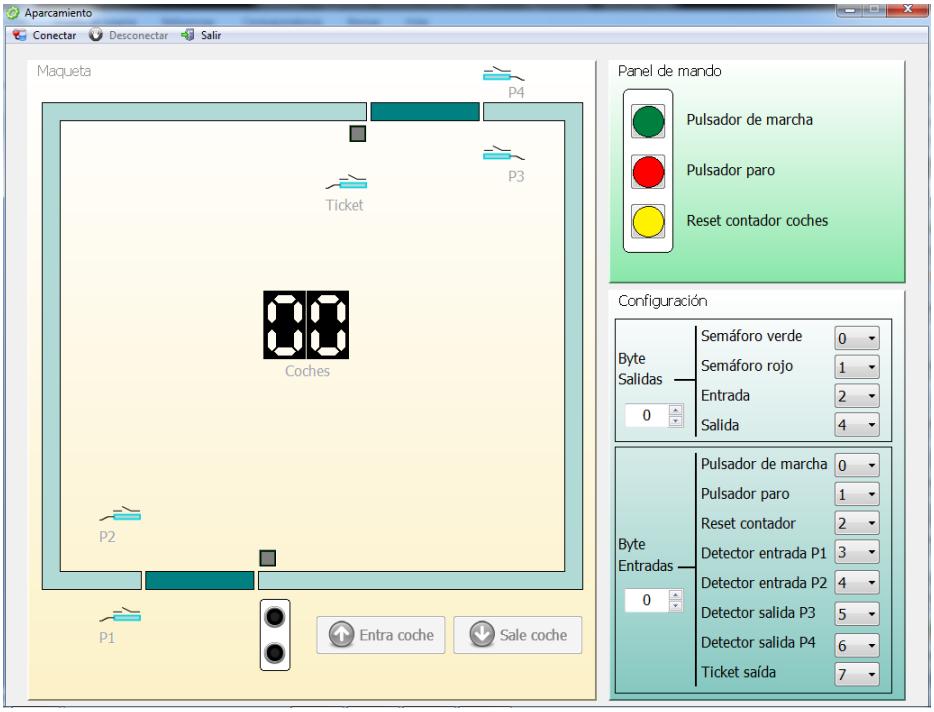


7.9.1.2 Exercicio 8

Preténdese automatizar o acceso a un aparcamento que dispón dun número determinado de prazas.

Na entrada do mesmo haberá un semáforo que estará verde cando dentro existan prazas libres, e poñerase vermello cando estean todas ocupadas.

Empregaremos de novo virtualmakTCP para simular o exercicio.

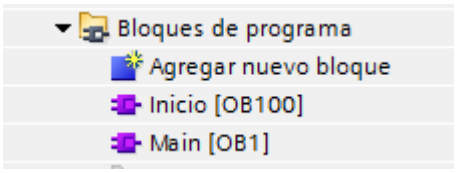


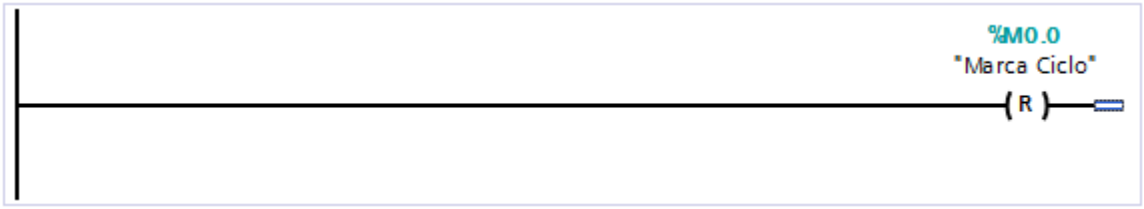
Como só queremos controlar o semáforo de entrada para indicar se hai prazas libres (verde) ou se están todas ocupadas (vermello), empregaremos as entradas e saídas seguintes:

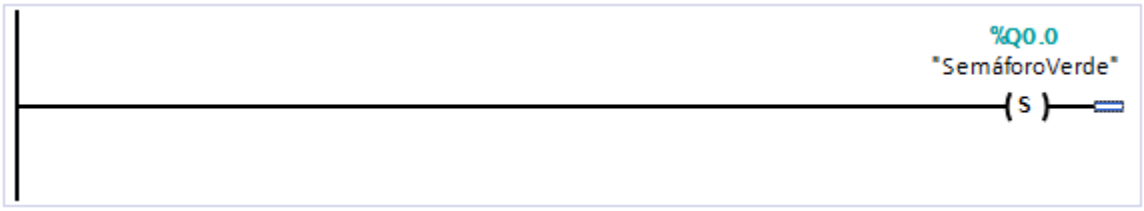
Entradas	Descrición
I0.2	Reset contador de coches
I0.3	Detector P1 de coche entrando
I0.4	Detector P2 de coche dentro
Saídas	Descrición
Q0.0	Semáforo verde
Q0.1	Semáforo vermello
Q0.2	Abrir porta entrada

O programa podería ser como o seguinte:

Primeiro programamos unha subrutina de inicio na que reseteamos unha marca auxiliar que empregaremos para detectar un flanco de baixada e poñemos o semáforo en verde (Inicio OB100).

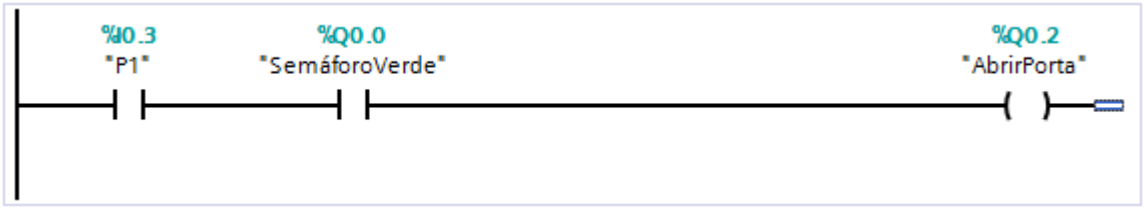




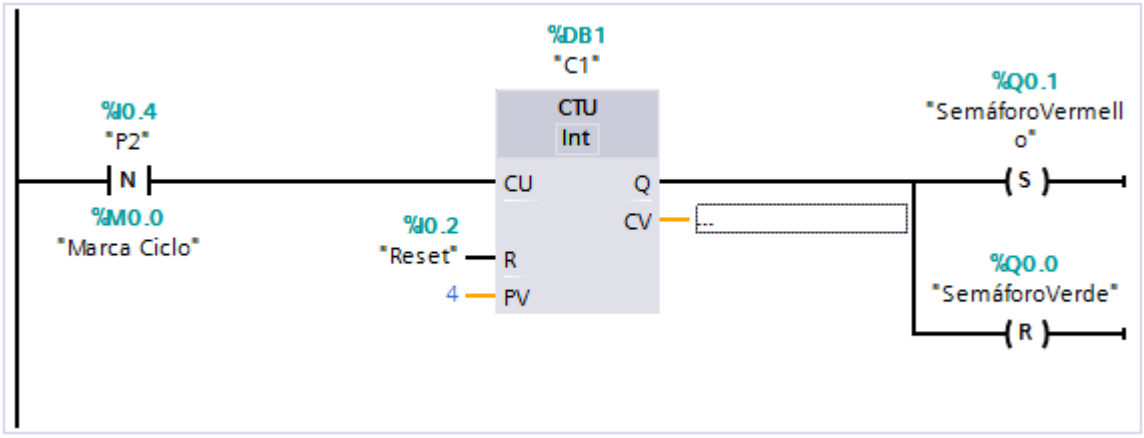


No programa principal (Main OB1) temos:

Cando un coche é detectado polo sensor de proximidade P1 e o semáforo verde indica que hai prazas libres, dase un pulso para que a porta de entrada abra (Q0.2).

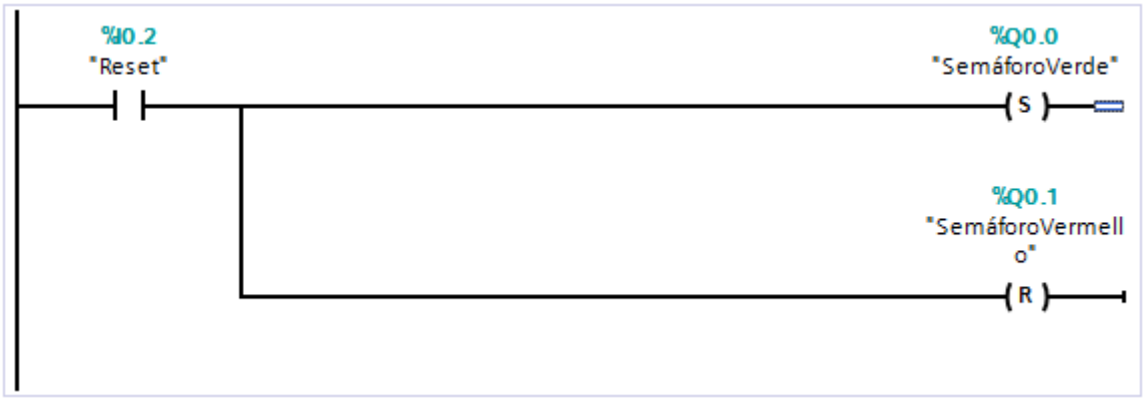


Comenza o proceso de entrada, no que un sensor de proximidade P2 detecta o coche. Cando este sensor deixa de detectalo (flanco negativo en I0.4) consideramos que o coche entrou, e polo tanto incrementamos o contador.



No segmento anterior vemos o contador para catro coches e o pulsador de reset (I0.2), para reinicialo.

Por último, o pulsador de reset tamén reinicia o semáforo verde e apaga o vermello.



7.9.1.3 Exercicio 9

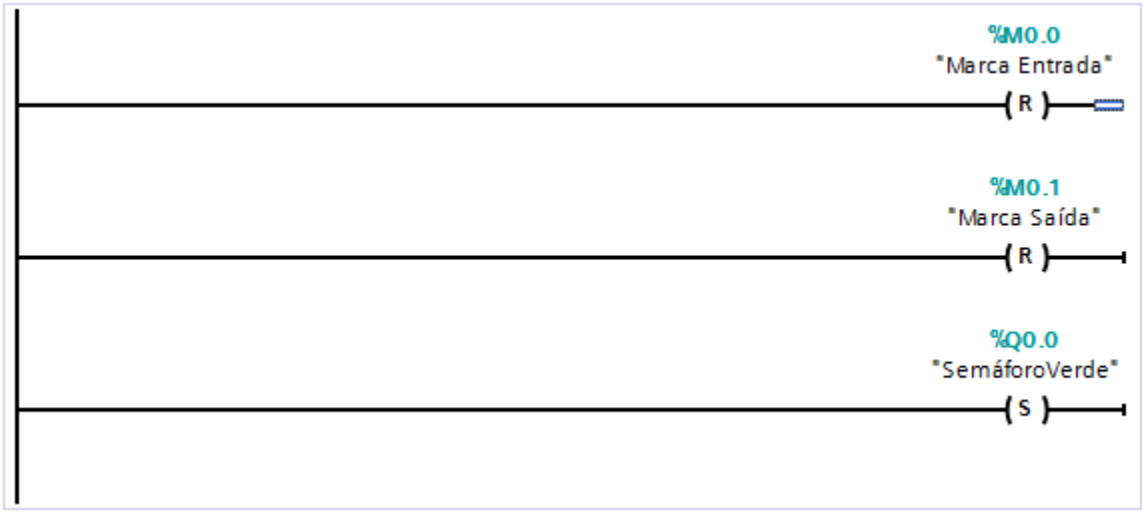
Completar o exercicio anterior, considerando os coches que entran e os que saen. Empregando un contador ascendente/descendente para facelo.

Agora temos a táboa de entradas/saídas seguinte:

Entradas	Descrición
I0.2	Reset contador de coches
I0.3	Detector P1 de coche entrando
I0.4	Detector P2 de coche dentro
I0.5	Detector P3 de coche saíndo
I0.6	Detector P4 de coche fóra
I0.7	Detector de tiket de saída
Saídas	Descrición
Q0.0	Semáforo verde
Q0.1	Semáforo vermello
Q0.2	Abrir porta entrada
Q0.3	Abrir porta saída

Unha posible solución sería:

Nunha subrutina de inicio (OB100) reseteamos marcas e poñemos o semáforo en verde.

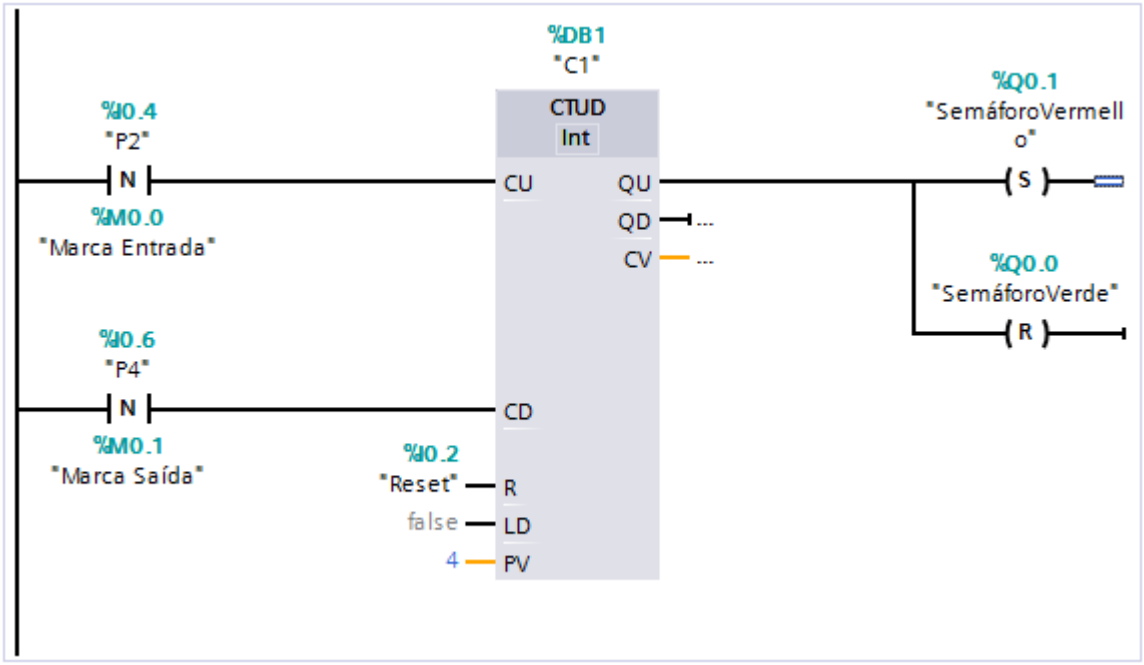


No programa principal Main (OB1) empezamos abrindo a porta de entrada si se cumpren as condicións para abrila, que son, que o sensor P1 detecte un coche e o semáforo estea verde.

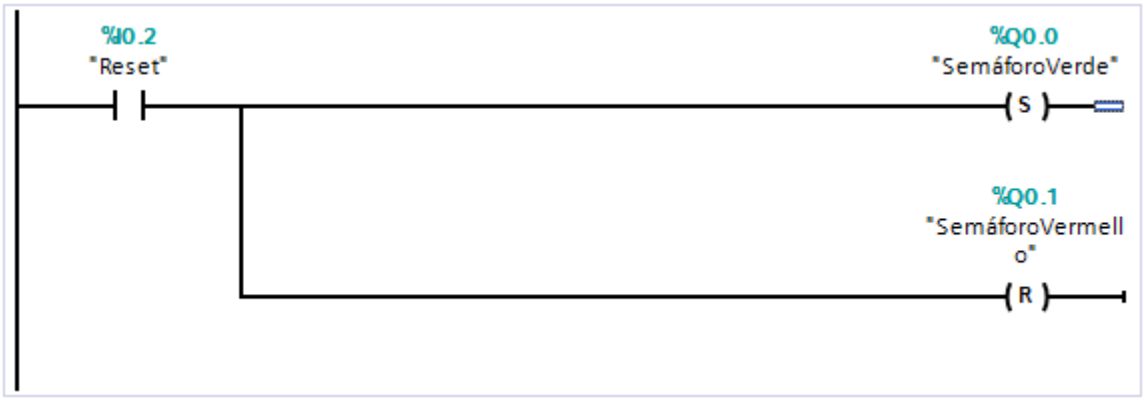
Incrementamos ou decrementamos o contador cando entren (flanco negativo en P2) ou cando saian (flanco negativo en P4) coches respectivamente.

Se o contador chega ao número de coches estipulado, o semáforo cambia a vermello, indicando que está cheo.

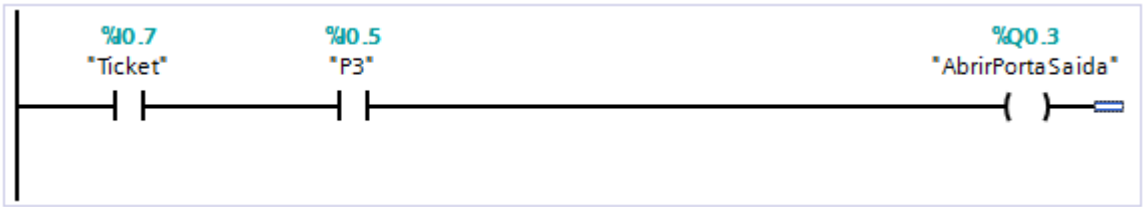
Por outra banda, podemos resetar o contador coa entrada I0.2



Ademais de resetar o contador, coa entrada I0.2 cambiamos o semáforo a verde.



No seguinte segmento, abrimos a porta de saída cando hai un ticket válido (Ticket I0.7) e hai un coche esperando para saír (P3 I0.5) .



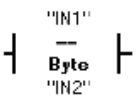
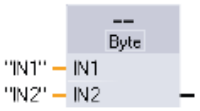
NOTA: Modifíquese o programa para que siga a funcionar cando se chega ao límite de coches e sae un coche.

7.10 Instruccións de comparación

7.10.1 Comparación

Compara dous datos e, dependendo do tipo de relación, activa ou desactiva unha saída booleana.

KOP	FUP	SCL	Descripción

		`out := in1 = in2;`
		or
		`IF in1 = in2`
		`THEN out := 1;`
		`ELSE out := 0;`
		`END_IF;`
			Compara varios elementos del mismo tipo de datos. Si la comparación de contactos KOP es TRUE (verdadera), se activa el contacto. Si la comparación de cuadros FUP es TRUE (verdadera), la salida del cuadro es TRUE.

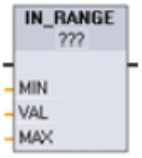
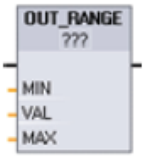
Os tipos de datos IN1 e IN2 poden ser: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, String, Char, Time.

Os tipos de relación entre os datos poden ser calquera dos da seguinte táboa.

Tipo de relación	La comparación se cumple si ...
=	IN1 es igual a IN2
<>	IN1 es diferente de IN2
>=	IN1 es mayor o igual a IN2
<=	IN1 es menor o igual a IN2
>	IN1 es mayor que IN2
<	IN1 es menor que IN2

7.10.2 Instruccións “Valor dentro do rango” e “Valor fóra do rango”

Son útiles para comprobar se un valor está dentro dun rango ou se está fóra do mesmo.

KOP / FUP	SCL	Descripción
	`out := IN_RANGE(min, val, max);`	Comprueba si un valor de entrada está dentro o fuera de un rango de valores específico. Si la comparación es TRUE (verdadera), la salida del cuadro es TRUE.
	`out := OUT_RANGE(min, val, max);`	

Os valores para MIN, MAX e VAL poden ser: SInt, Int, DInt, USInt, UInt, UDInt, Real, Lreal.

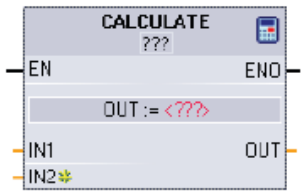
No caso da instrucción IN_RANGE os intervalos son pechados, é dicir, a saída actívase se MIN <= VAL <= MAX.

No caso da instrucción OUT_RANGE os intervalos son abertos, é dicir, a saída actívase se Val < MIN ou VAL > MAX

7.11 Funcións matemáticas

7.11.1 Instrucción calcular

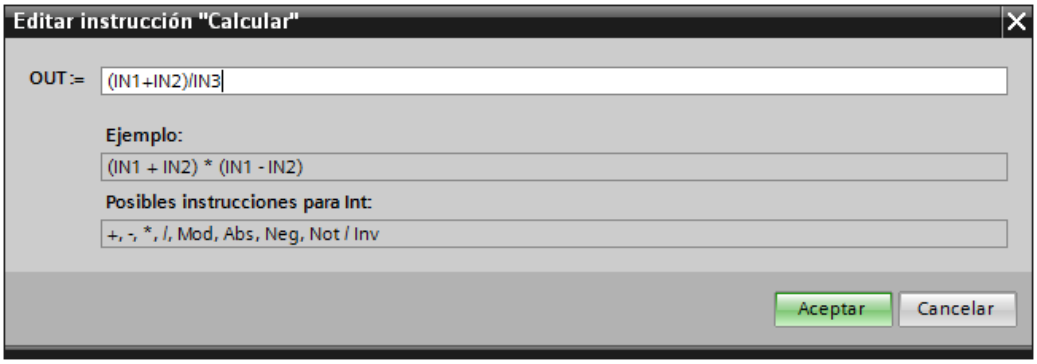
Permite construír unha expresión matemática con entradas e xenera unha saída segundo a expresión definida.

KOP / FUP	SCL	Descripción
	Utilice las expresiones matemáticas SCL estándar para crear la ecuación.	La instrucción CALCULATE permite crear una función matemática que funciona con entradas (IN1, IN2, ... INn) y genera el resultado en OUT, según la ecuación definida. • En primer lugar, seleccione un tipo de datos. Todas las entradas y la salida deben tener un mismo tipo de datos. • Para agregar otra entrada, haga clic en el icono de la última entrada.

Os tipos de datos para as entradas e a saída poden ser: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, Dword.

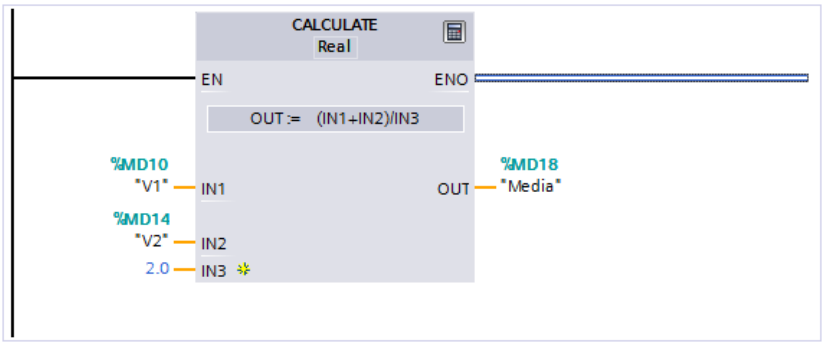
Os tipos de datos teñen que gardar certa coherencia. Por exemplo, se empregamos datos de tipo enteiro nas entradas e a operación matemática leva algunha división, a saída ten que ser un número real.

Para definir a expresión matemática primeiro seleccionamos o tipo de datos a empregar e logo faremos click no icono da calculadora que aparece na parte superior dereita.



Non de poden introducir directamente constantes na expresión. Por exemplo, se queremos calcular a media aritmética do dous valores, non podemos escribir a expresión como

(IN1+IN2)/2. Temos que definir outra entrada para a constante (IN3) e colocar o valor nesa entrada, tal como vemos na figura seguinte.



7.11.2 Instruccións “sumar”, “restar”, “multiplicar” e “dividir”

KOP / FUP	SCL	Descripción

	`out := in1 + in2;`
	`out := in1 - in2;`
	`out := in1 * in2;`
	`out := in1 / in2;`
		• ADD: Sumar (IN1 + IN2 = OUT)
		• SUB: Restar (IN1 - IN2 = OUT)
		• MUL: Multiplicar (IN1 * IN2 = OUT)
		• DIV: Dividir (IN1 / IN2 = OUT)
		Una operación de división de enteros trunca la parte fraccionaria del cociente y produce un valor de salida entero.

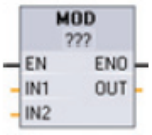
Os tipos de datos que se permiten para as entradas IN1 e IN2 son:

Parámetro	Tipo de datos[1]	Descripción
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, constante	Entradas de la operación matemática
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Salida de la operación matemática

Pódense inserir máis entradas cando as operación son “sumar” ou “multiplicar”. Para eso facemos click co botón dereito enriba da entrada IN2 e escollemos “Insertar entrada”.

7.11.3 Instrucción “módulo”

Obtén o resto dunha división de números enteiros.

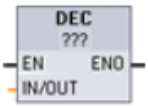
KOP / FUP	SCL	Descripción
	`out := in1 MOD in2;`	La instrucción MOD se puede utilizar para obtener el resto de una operación de división de enteros. El valor de la entrada IN1 se divide por el valor de la entrada IN2 y el producto se deposita en la salida OUT.

Os tipos de datos que se admiten para as entradas son:

Parámetro	Tipo de datos[1]	Descripción
IN1 y IN2	SInt, Int, DInt, USInt, UInt, UDInt, constante	Entradas modulo
OUT	SInt, Int, DInt, USInt, UInt, UDInt	Salida modulo

7.11.4 Instruccións “incrementar” e “decrementar”

Incrementan ou decrementan nunha unidade a entrada.

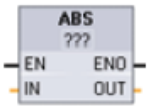
KOP / FUP	SCL	Descripción
	`in_out := in_out + 1;`	Incrementa un valor de número entero con o sin signo: Valor IN_OUT +1 = valor IN_OUT
	`in_out := in_out - 1;`	Decrementa un valor de número entero con o sin signo: Valor IN_OUT - 1 = valor IN_OUT

Os tipos de datos que se permiten para a entrada son:

Parámetro	Tipo de datos	Descripción
IN/OUT	SInt, Int, DInt, USInt, UInt, UDInt	Entrada/salida de la operación matemática

7.11.5 Instrucción “valor absoluto”

Calcula o valor absoluto da entrada.

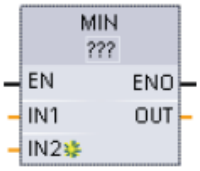
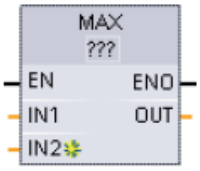
KOP / FUP	SCL	Descripción
	`out := ABS(in);`	Calcula el valor absoluto de un entero con signo o número real indicado en el parámetro IN y deposita el resultado en el parámetro OUT.

Os tipos de datos que se permiten para a entrada son:

Parámetro	Tipo de datos[1]	Descripción
IN	SInt, Int, DInt, Real, LReal	Entrada de la operación matemática
OUT	SInt, Int, DInt, Real, LReal	Salida de la operación matemática

7.11.6 Instruccións “mínimo” e “máximo”

Devolve o valor mínimo ou máximo dunha serie de valores de entrada.

KOP / FUP	SCL	Descripción
	out:= MIN(in1:=_variant_in_, in2:=_variant_in_ [,...in32]);	La instrucción MIN compara el valor de dos parámetros IN1 y IN2 y asigna el valor mínimo (menor) al parámetro OUT.
	out:= MAX(in1:=_variant_in_, in2:=_variant_in_ [,...in32]);	La instrucción MAX compara el valor de dos parámetros IN1 y IN2 y asigna el valor máximo (mayor) al parámetro OUT.

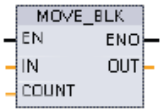
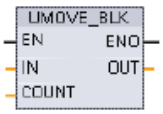
Os tipos de datos permitidos para os parámetros son:

Parámetro	Tipo de datos[1]	Descripción
IN1, IN2 [...IN32]	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, constante	Entradas de la operación matemática (hasta 32 entadas)
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Salida de la operación matemática

7.12 Desprazamento

7.12.1 Instruccións “copiar valor” e “copiar bloque”

Copian valores dunhas posicións de memoria a outras. Os valores orixinais non sofren variación.

KOP / FUP	SCL	Descripción
	out1 := in;	Copia un elemento de datos almacenado en una dirección indicada en una o varias direcciones diferentes. [1]
	MOVE_BLK(in:=_variant_in, count:=_uint_in, out=>_variant_out);	Desplazamiento con interrupciones que copia un bloque de elementos de datos en otra dirección.
	UMOVE_BLK(in:=_variant_in, count:=_uint_in, out=>_variant_out);	Desplazamiento sin interrupciones que copia un bloque de elementos de datos en otra dirección.

A instrucción MOVE copia un dato individual (IN) a unha ou varias posicións (OUT1...).

As instruccións MOVE_BLK e UMOVE_BLK copian bloques de datos especificados polo parámetro COUNT. A diferencia básica entre a primeira e a segunda e que na primeira vanse copiando os valores e van estando dispoñíbles no destino a medida que se van copiando, mentres que na segunda ata que non se remata o proceso de volcado de todos os datos do bloque, non están dispoñíbles para o seu proceso.

Os tipos de datos que se admiten para a instrucción MOVE son:

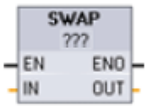
Parámetro	Tipo de datos	Descripción
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	Dirección de origen
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	Dirección de destino

Os tipos de datos que se admiten para as instruccións MOVE_BLK e UMOVE_BLK son:

Parámetro	Tipo de datos	Descripción
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal Byte, Word, DWord	Dirección de origen inicial
COUNT	UInt	Número de elementos de datos que deben copiarse
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord	Dirección de destino inicial

7.12.2 Instrucción “cambiar orden”

Inverte a orde dos bytes en datos de dous ou catro bytes.

KOP / FUP	SCL	Descripción
	`out := SWAP(in);`	Invierte el orden de los bytes para elementos de datos de dos bytes y cuatro bytes. El orden de los bits no se modifica dentro de los distintos bytes. ENO es siempre TRUE (verdadero) tras ejecutarse la instrucción SWAP.

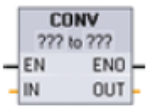
Os tipos de datos que se admiten para esta instrucción son:

Parámetro	Tipo de datos	Descripción
IN	Word, DWord	Bytes de datos ordenados en IN
OUT	Word, DWord	Bytes de datos en orden inverso en OUT

7.13 Conversión

7.13.1 Instrucción CONV

Converte un tipo de datos en outro.

KOP / FUP	SCL	Descripción
	`out := <data type in>_TO_<data type out>(in);`	Convierte un elemento de datos de un tipo de datos a otro tipo de datos.

Os tipos de datos que se admiten nesta instrucción son:

Parámetro	Tipo de datos	Descripción
IN	Cadena de bits[1], SInt, USInt, Int, UInt, DInt, UDInt, Real, LReal, BCD16, BCD32	Valor de entrada
OUT	Cadena de bits[1], SInt, USInt, Int, UInt, DInt, UDInt, Real, LReal, BCD16, BCE32	Valor de entrada convertido a un nuevo tipo de datos

7.13.2 Instruccións “redondear” e “truncar”

Redondean ao enteiro máis próximo (ROUND) ou truncan a parte fraccionaria (TRUNC) de números reais.

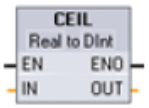
KOP / FUP	SCL	Descripción
	`out := ROUND (in);`	Convierte un número real en un entero. El tipo de datos predeterminado es DINT. Cuando la salida es un tipo de datos válido distinto de DINT, debe declararse de forma explícita; por ejemplo, ROUND_REAL o ROUND_LREAL. La fracción del número real se redondea al número entero más cercano (IEEE - redondear al número más cercano). Si el número se encuentra exactamente entre dos enteros (p. ej. 10,5), el número se redondeará al entero par. Ejemplo: • ROUND (10.5) = 10 • ROUND (11.5) = 12
	`out := TRUNC(in);`	TRUNC convierte un número real en un entero. La parte fraccionaria del número real se trunca a cero (IEEE - redondear hacia cero).

Os tipos de datos que se admiten nestas instruccións son:

Parámetro	Tipo de datos	Descripción
IN	Real, LReal	Número en coma flotante en la entrada
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Salida redondeada o truncada

7.13.3 Instruccións CEIL e FLOOR

Convirten un número no seguinte (CEIL) ou non anterior (FLOOR) enteiro.

KOP / FUP	SCL	Descripción
	`out := CEIL(in);`	Convierte un número real (Real o LReal) en el siguiente entero mayor o igual a ese número real (IEEE - redondear hacia el infinito positivo).
	`out := FLOOR(in);`	Convierte un número real (Real o LReal) en el siguiente entero menor o igual a ese número real (IEEE - redondear hacia el infinito negativo).

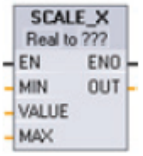
Os tipos de datos que se aceptan nestas instruccións son:

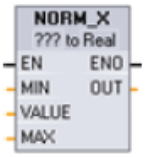
Parámetro	Tipo de datos	Descripción
IN	Real, LReal	Número en coma flotante en la entrada
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Salida convertida

7.13.4 Instruccións “escalar” e “normalizar”

A primeira escala un valor de entrada comprendido entre 0 e 1 nun rango de valores comprendidos entre un mínimo e un máximo. A segunda fai o contrario.

KOP / FUP	SCL	Descripción

	`out :=SCALE_X(min:=_in_,`
	`value:=_in_,`
	`max:=_in_);`
		Escala el parámetro VALUE real normalizado (donde 0,0 <= VALUE <= 1,0) al tipo de datos y rango de valores especificados por los parámetros MIN y MAX:
		OUT = VALUE (MAX - MIN) + MIN

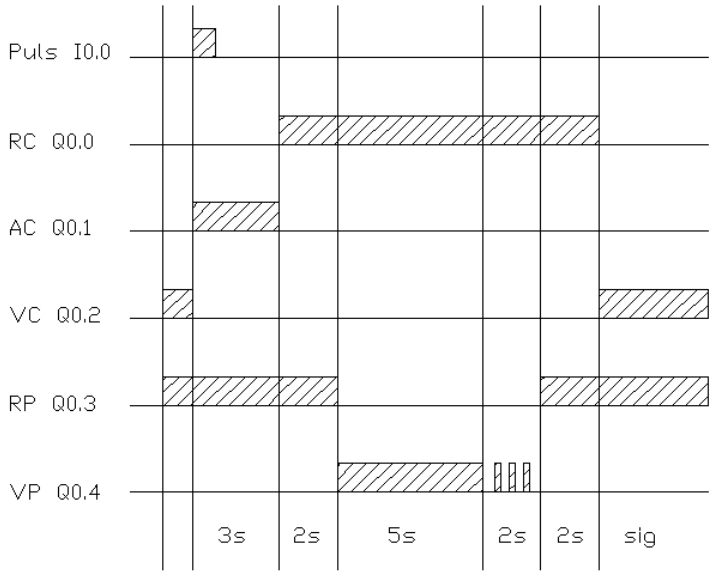
	`out :=NORM_X(min:=_in_,`
	`value:=_in_,`
	`max:=_in_);`
		Normaliza el parámetro VALUE dentro del rango de valores especificado por los parámetros MIN y MAX:
		OUT = (VALUE - MIN) / (MAX - MIN), donde (0,0 <= OUT <= 1,0)

Os tipos de datos que se permiten nestas instruccións son:

Parámetro	Tipo de datos[1]	Descripción
MIN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Entrada que indica el valor mínimo del rango
VALUE	SCALE_X: Real, LReal NORM_X: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Valor de entrada que se debe escalar o normalizar
MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Entrada que indica el valor máximo del rango
OUT	SCALE_X: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal NORM_X: Real, LReal	Valor de salida escalado o normalizado

7.13.4.1 Exercicio 10

Programa para un semáforo cun pulsador, no que o mesmo está verde para os coches ata que un peatón acciona o pulsador. Nese momento realizará o ciclo que se indica na figura seguinte:

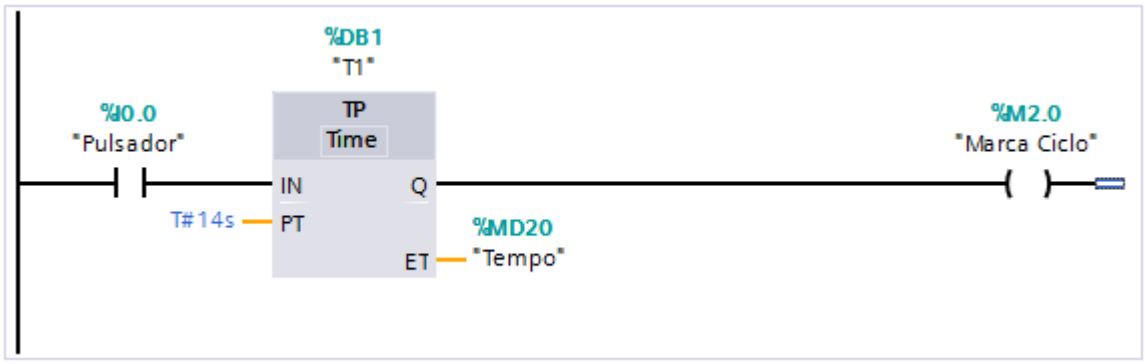


RC : Vermello coche	RP: Vermello peatón
AC: Amarelo coche	VP: Verde peatón
VC: Verde coche	

A táboa de variables sería:

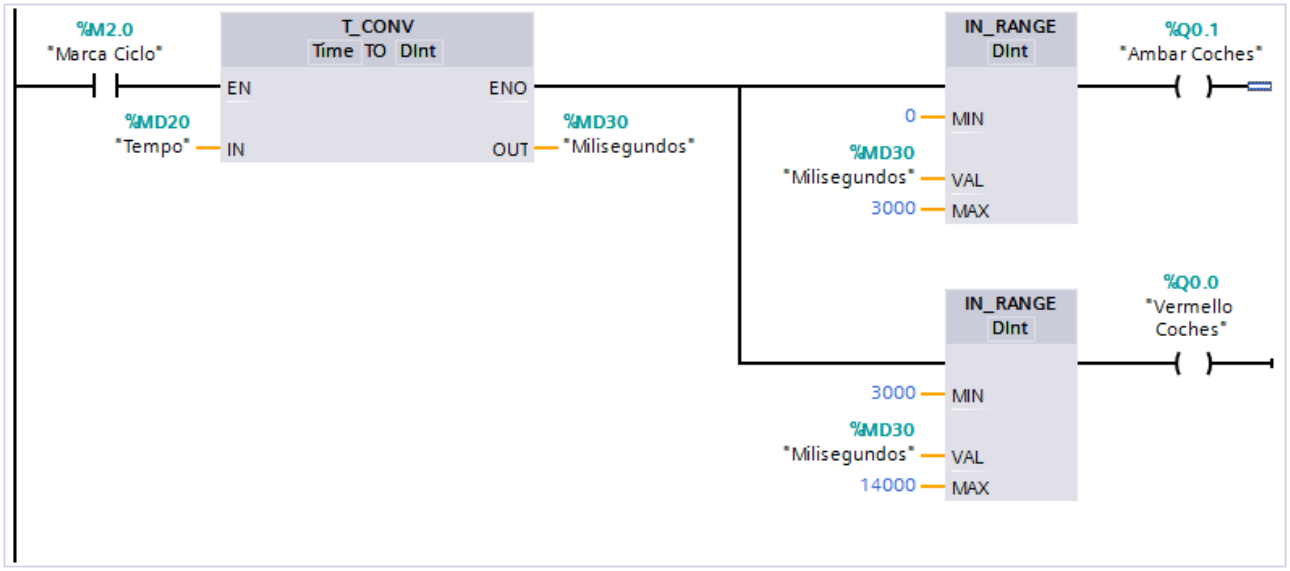
Entradas	Descrición
I0.0	Pulsador peóns
Saídas	Descrición
Q0.0	Vermello coches
Q0.1	Ámbar coches
Q0.2	Verde coches
Q0.3	Vermello peóns
Q0.4	Verde peóns

Primeiro, co pulsador poñemos en marcha un temporizador que establecerá o tempo de ciclo (T1). Tamén se activa unha Marca Ciclo (M2.0) que deixará de estar activa cando o ciclo remate.



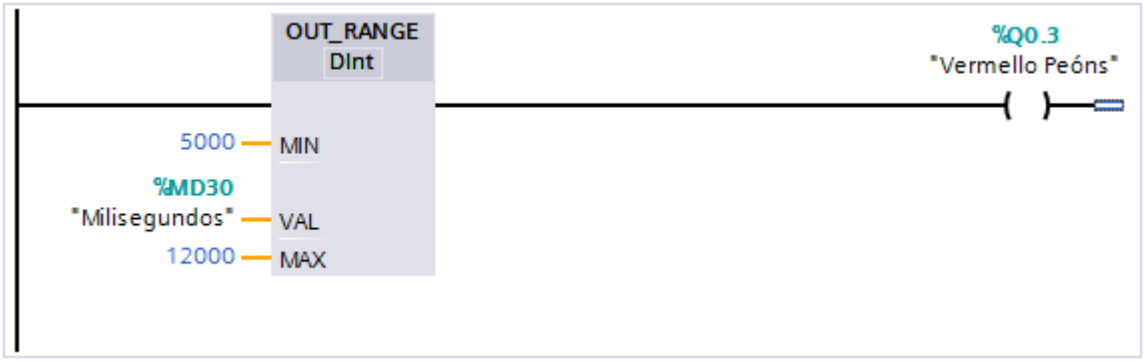
Coa Marca Ciclo convertemos o tempo de formato Time (MD20) a milisegundos (MD30) para traballar cómodamente con intervalos.

Se estamos entre o segundo 0 e o 3, poñemos ámbar para coches (Q0.2). No resto do ciclo, é dicir entre 3s e 14s poñemos vermello para coches (Q0.0).

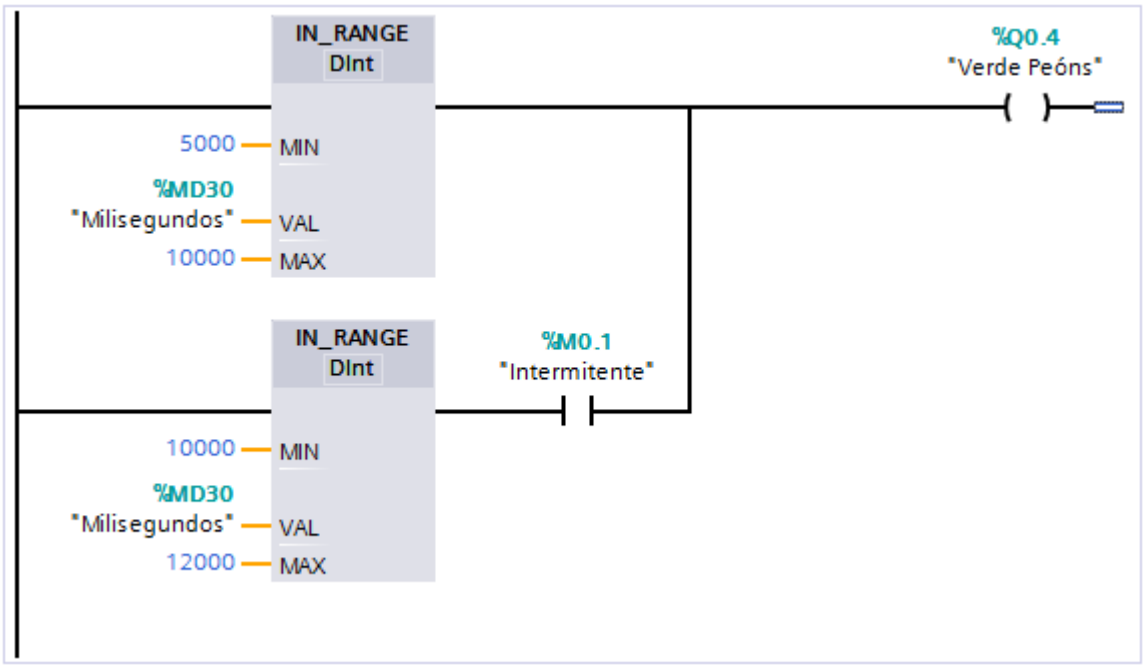


Cando non estea vermello nin ámbar para coches, poñemos verde para coches (Q0.4).

Se non estamos no intervalo de 5s a 12s poñemos vermello para peóns (Q1.0).



Entre 5s e 10s está verde para peóns (Q1.1). Entre 10s e 12s tamén está verde pero parpadeando a luz (M0.1).



Para que funcione o intermitente coa marca M0.1 hai que configurar os Bits de Marcas de Ciclo tal como se indica no punto 9.1.2

8 Programación de sistemas secuenciais mediante GRAFCET

O GRAFCET, tamén chamado SFC (apartado 6.1.2) , é unha forma gráfica de representar o funcionamento dun sistema secuencial.

Representa a secuencia de funcionamento da máquina e facilita a implementación a calquera linguaxe de programación de autómatas.

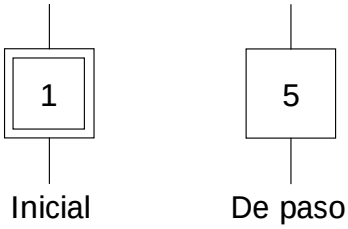
Está formado por un conxunto de elementos gráficos denominados etapas, transicións, etiquetas e liñas de dirección.

8.1 Etapas

Representan os diferentes estados do proceso secuencial.

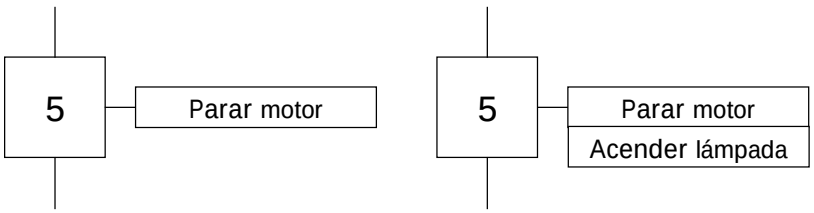
O seu símbolo é un cadrado cun número dentro.

Podemos distinguir dous tipos: Iniciais e de paso.



A etapa inicial é coa que comeza o proceso.

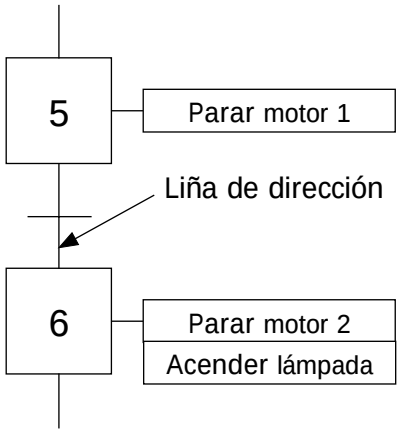
Das etapas colgan as chamadas etiquetas, nas que se indican as accións a realizar.



Cando a secuencia chega a unha etapa determinada, execútanse as accións que se indican nela.

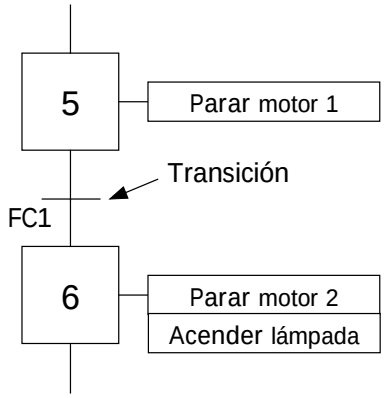
A non ser que se definan secuencias paralelas, nun GRAFCET só pode estar activa unha etapa nun momento determinado.

As etapas únense entre sí polas liñas de dirección.



8.2 Transicións

Son as condicións que permiten o paso dunha etapa á outra. Represéntanse mediante unha liña horizontal sobre a liña de dirección.



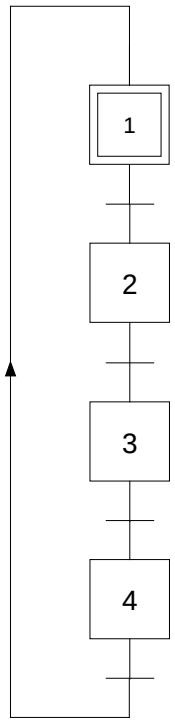
8.3 Tipos de GRAFCET

Segundo a evolución da secuencia, podemos representar tres tipos de GRAFCET: De secuencia única, de secuencias opcionais e de secuencias simultáneas.

8.3.1 GRAFCET de secuencia única

Neste caso as etapas e as transicións van conectadas en cascada. A evolución só segue un camiño.

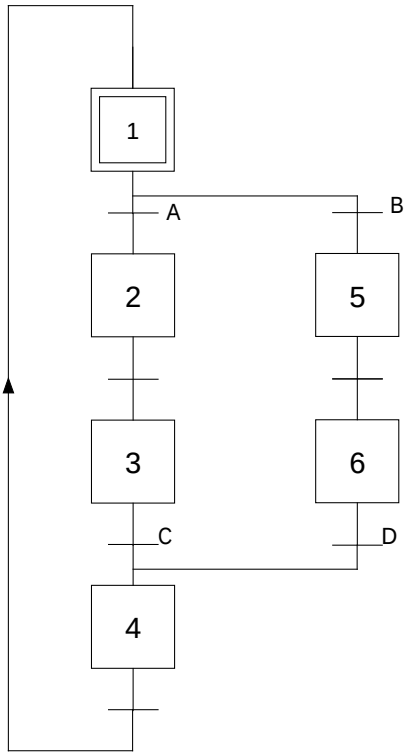
É a maneira máis sinxela de representar a secuencia dun proceso.



8.3.2 GRAFCET de secuencias opcionais

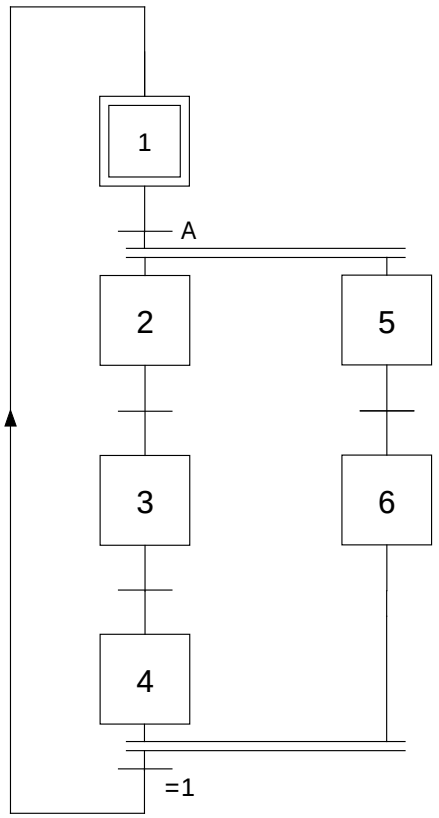
Neste caso, a secuencia pode optar por seguir varios camiños, pero só se executará un de cada vez.

Emprégase unha liña horizontal para representar as alternativas.



8.3.3 GRAFCET de secuencias simultáneas

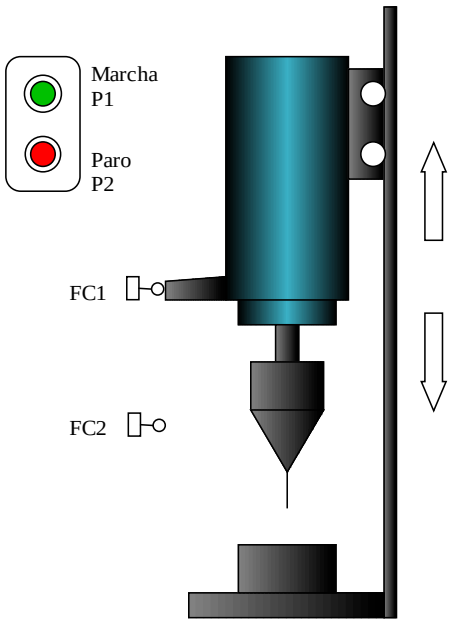
Este tipo de secuencia ten que evolucionar por varios camiños á vez, de xeito que ata que non se completan todos, non continúa o proceso.



Na figura anterior temos que coa transición A execútanse de xeito simultáneo as etapas 2 e 5. Só cando rematen as etapas 4 e 6 continuará a secuencia.

8.3.3.1 Exercicio 11

Preténdese representar a secuencia dun taladro, no que temos un pulsador de marcha (P1) e un pulsador de paro (P2). Tamén dispoñemos de dous finais de carreira (FC1 e FC2) que detectan se o taladro está en posición inicial ou está baixado.

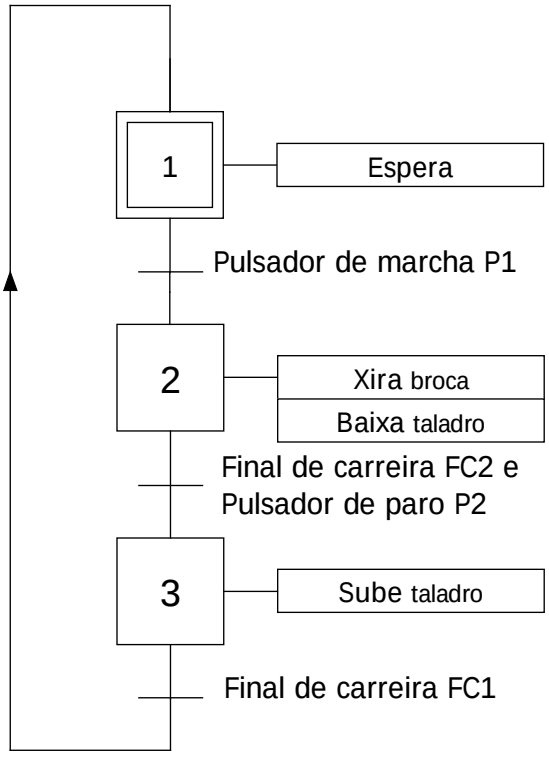


A secuencia do proceso sería:

1. O taladro está en espera.
2. Actuamos sobre o pulsador de marcha (P1).
3. O motor empeza a xirar e o taladro baixa.
4. O final de carreira de abaixo (FC2) detecta o taladro.
5. Actuamos sobre o pulsador de paro (P2).
6. O taladro sube coa broca xirando.
7. O final de carreira superior (FC1) detecta o taladro.
8. Paro total e volta ao estado de espera.

Na secuencia anterior hai etapas e transicións. En concreto, os puntos 1,3,6 e 8 representan etapas. Os puntos 2,4,5 e 7 son transicións.

O GRAFCET sería:



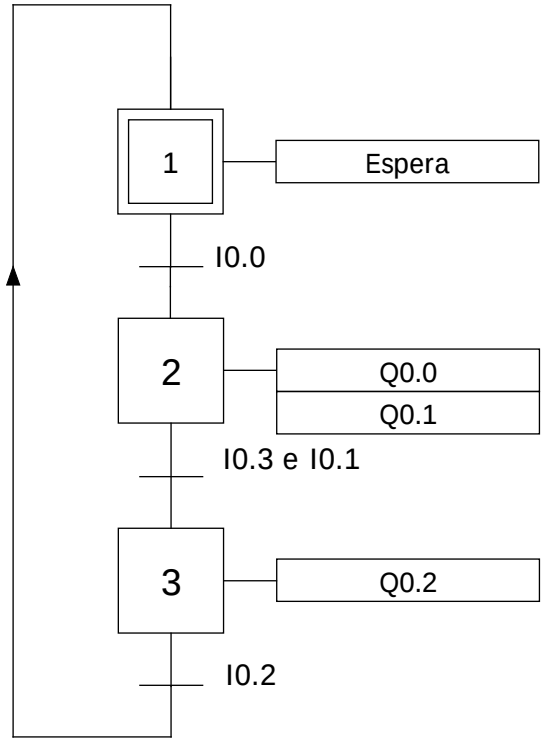
Este GRACFET chámase de **Primeiro Nivel** ou **Descriptivo**.

Nunha segunda estapa do deseño, aplícase ao GRAFCET a tecnoloxía a empregar., considerando as entradas e saídas do autómata.

Teriamos polo tanto, unha táboa de entradas/saídas como a seguinte.

Entradas	Descrición
I0.0	Pulsador marcha (P1)
I0.1	Pulsador paro (P2)
I0.2	Final de carreira arriba (FC1)
I0.3	Final de carreira abaixo (FC2)
Saídas	Descrición
Q0.0	Contactor do motor do taladro
Q0.1	Contactor de baixada
Q0.2	Contactor de subida

Obteriamos o GRAFCET de **Segundo Nivel** ou **Tecnolóxico**.



8.4 Programación dun GRAFCET en linguaxe de contactos

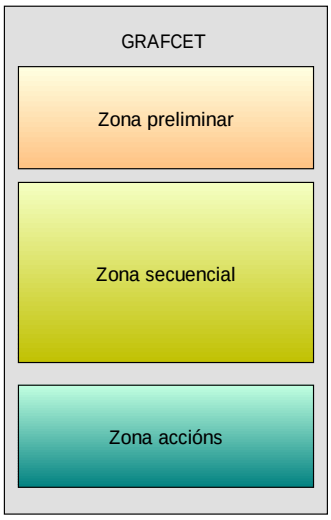
Aínda que algúns modelos de autómatas permiten a programación directa empregando GRAFCET, imos ver a secuencia de operacións a levar a cabo para convertelo a linguaxe de contactos.

Primeiro hai que ter en conta que a cada etapa asociaráselle unha marca interna do autómata, polo que no exemplo anterior teriamos.

Etapa	Marca	Simbólico
1	M10.1	ET1
2	M10.2	ET2
3	M10.3	ET3

8.4.1 Organización do GRAFCET

Para programar un GRAFCET distinguimos tres zonas: zona preliminar, zona secuencial e zona de accións.

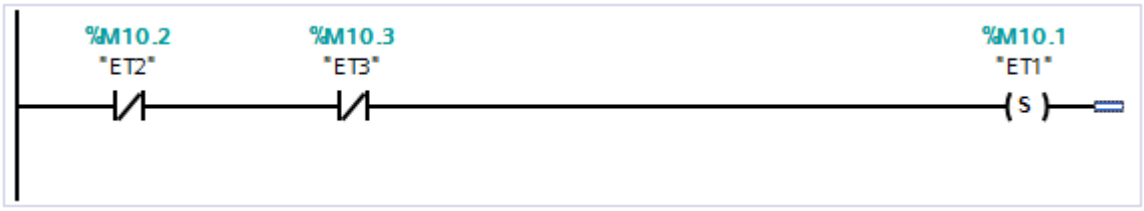


8.4.1.1 Zona preliminar

Nesta zona descríbese o programa que define a etapa incial pola que empeza a secuencia. Tamén se definen os modos de funcionamento manual e automático se os houbere e a parada de emerxencia.

A etapa inicial actívase poñendo a SET a bobina de etapa mediante unha serie de contactos pechados de todas as bobinas do resto das etapas.

No exemplo anterior sería:



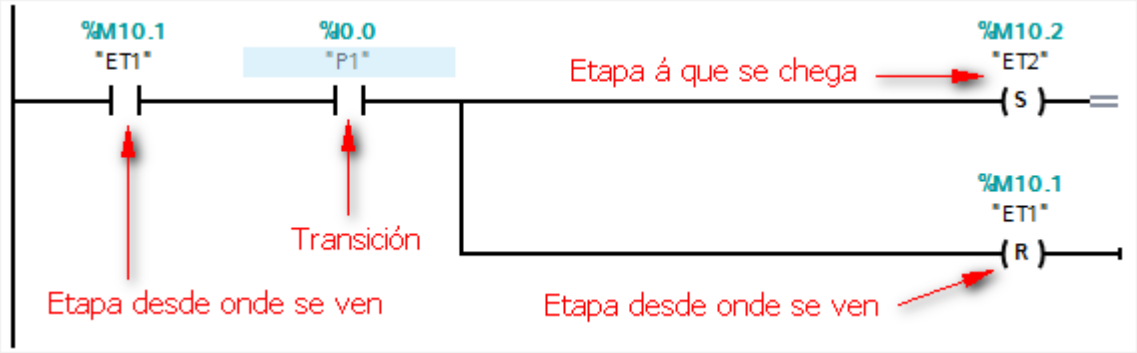
8.4.1.2 Zona secuencial

Define a secuencia que representa o encadenamento das etapas e transicións asociadas.

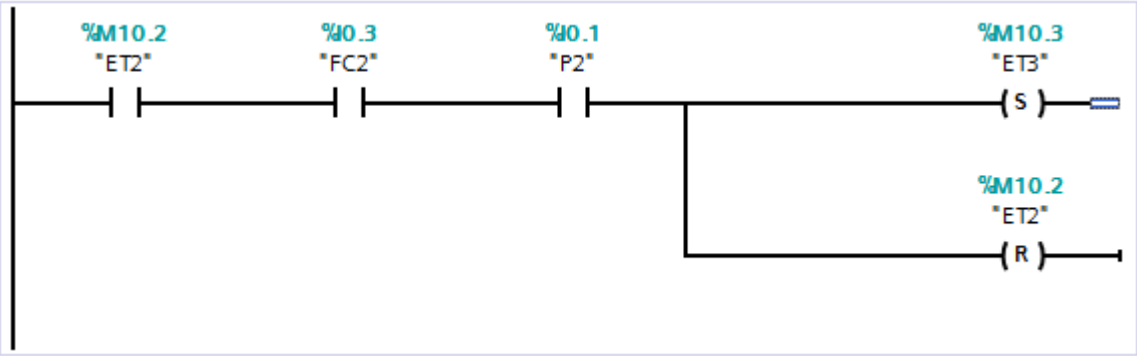
Cada etapa actívase mediante o SET dunha bobina, que se programa nunha rede de contactos na que se debe especificar o seguinte:

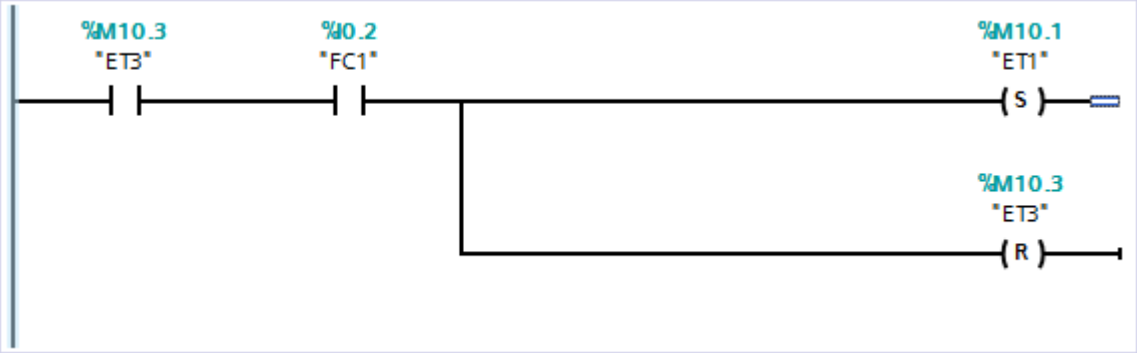
- Etapa á que se chega.
- Etapa desde a que se chega.

- Transición pola que se ven.



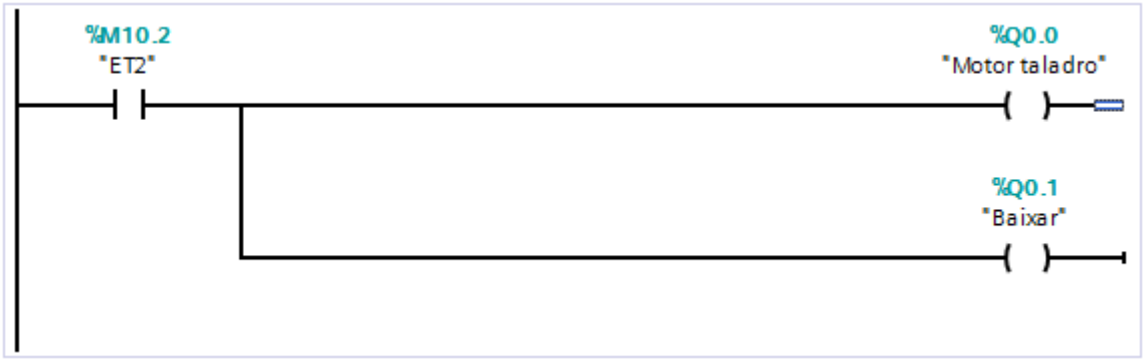
Así para todas as etapas.

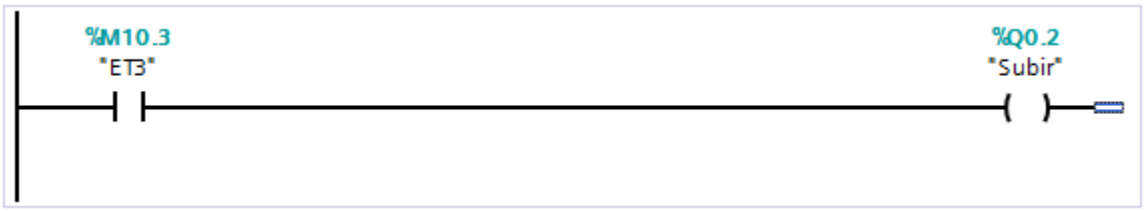




8.4.1.3 Zona de accións

Nesta parte activamos ou desactivamos as saídas do autómata en función da etapa na que nos atopemos. No exemplo anterior sería:

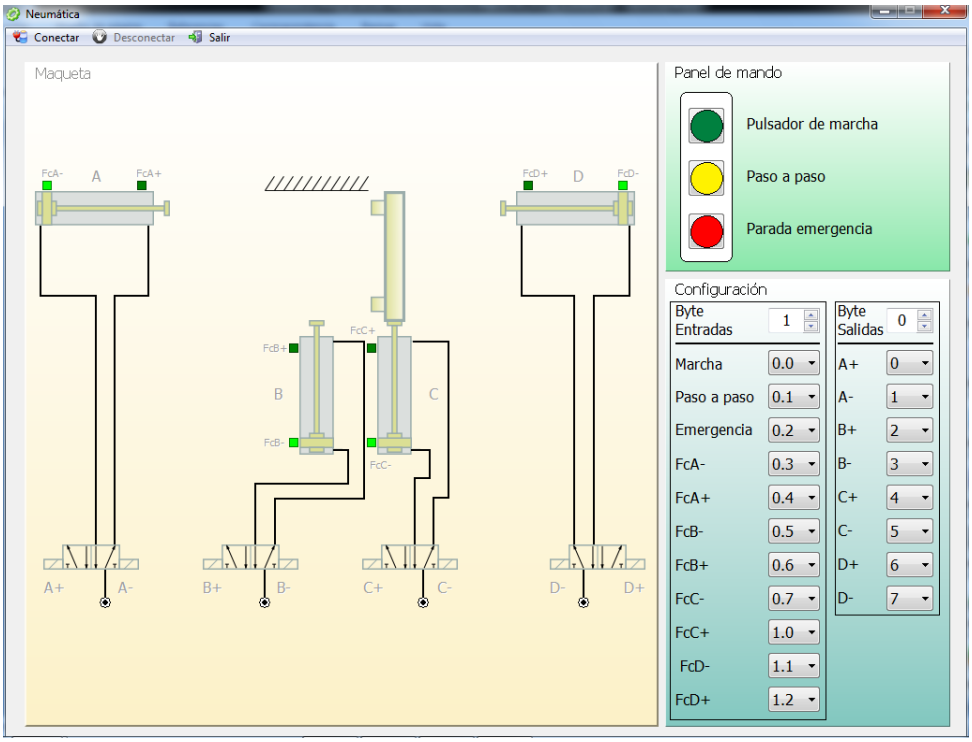




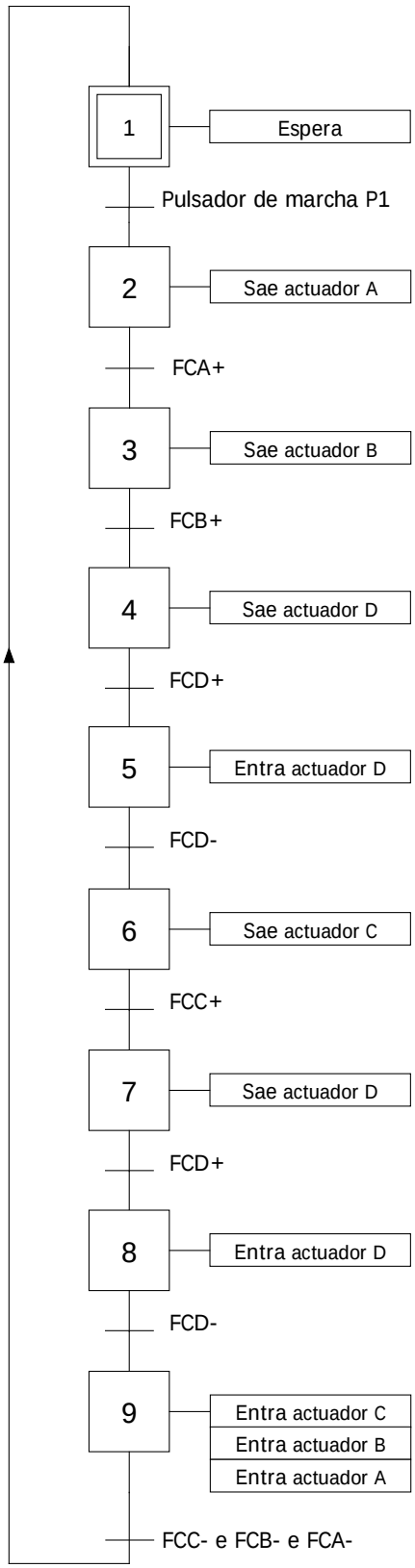
8.4.1.4 Exercicio 12

Preténdese programar a secuencia neumática dun sistema de estampado de pezas que consta de catro actuadores neumáticos. Un primeiro actuador (A) suxeita a peza pola parte posterior. Un segundo actuador (B) suxéitaa polo lateral. A continuación prodúcese o estampado dunha primeria marca mediante o actuador (D). Despois desprázase o actuador (C) e de novo se produce un segundo estampado pola acción do actuador (D).

Utilizaremos **virtualmakTCP** para simular o proceso.



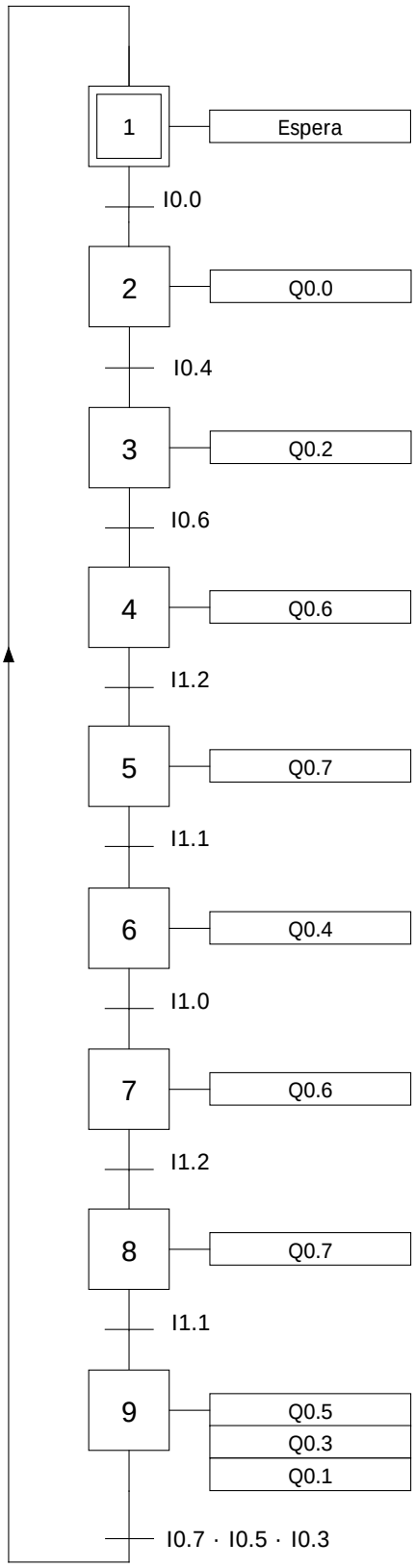
O GRAFCET de primeiro nivel sería:



A táboa de entradas/saídas sería a seguinte:

Entradas	Descrición
I0.0	Pulsador marcha (P1)
I0.3	FcA-
I0.4	FcA+
I0.5	FcB-
I0.6	FcB+
I0.7	FcC-
I1.0	FcC+
I1.1	FcD-
I1.2	FcD+
Saídas	Descrición
Q0.0	A+
Q0.1	A-
Q0.2	B+
Q0.3	B-
Q0.4	C+
Q0.5	C-
Q0.6	D+
Q0.7	D-

Coa táboa anterior temos o GRAFCET de segundo nivel.



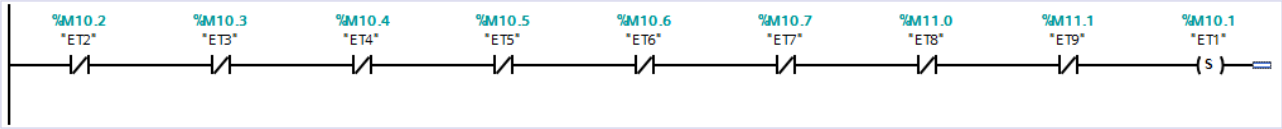
Temos unha táboa de etapas como a seguinte:

Etapa	Marca	Simbólico
1	M10.1	ET1
2	M10.2	ET2
3	M10.3	ET3
4	M10.4	ET4
5	M10.5	ET5
6	M10.6	ET6
7	M10.7	ET7
8	M11.0	ET8
9	M11.1	ET9

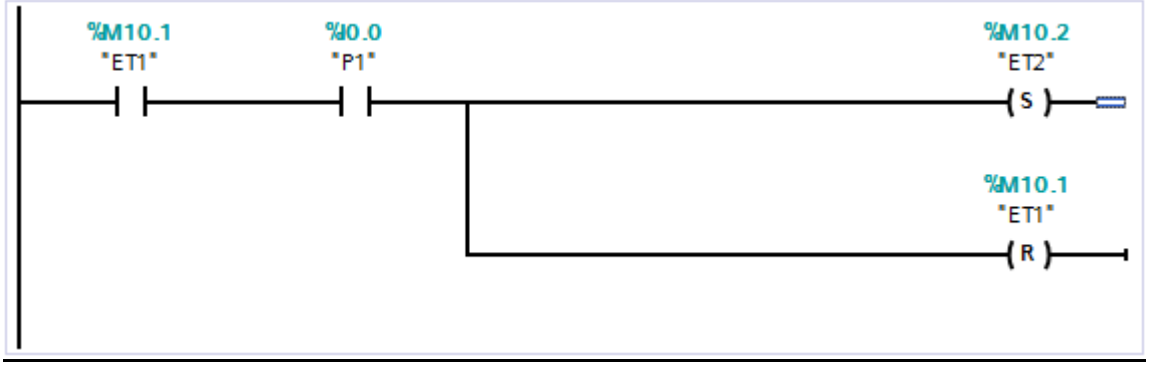
Agora programamos o autómata.

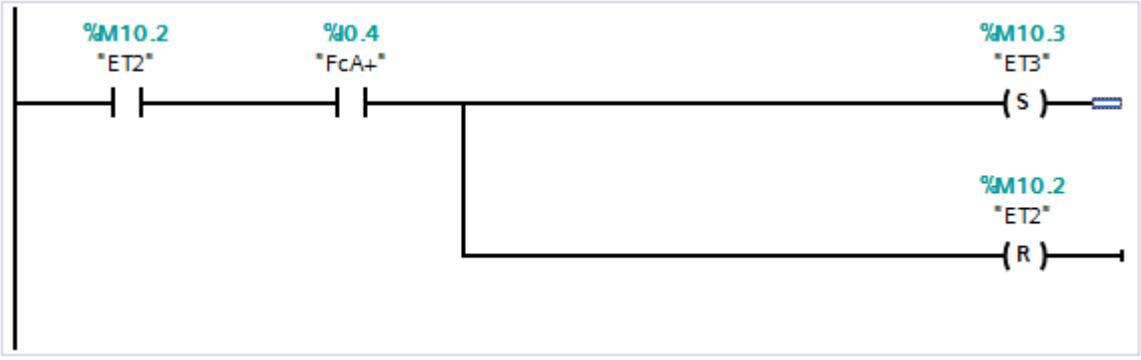
Zona preliminar

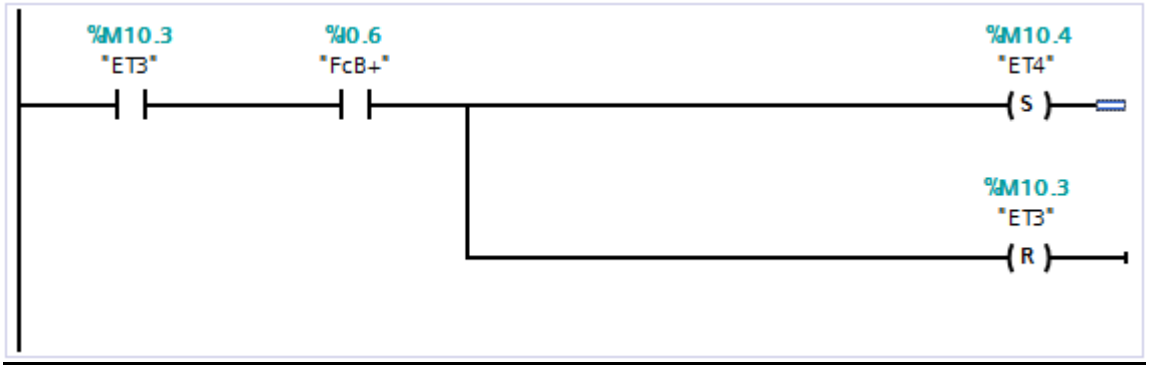
Definimos a etapa inical.

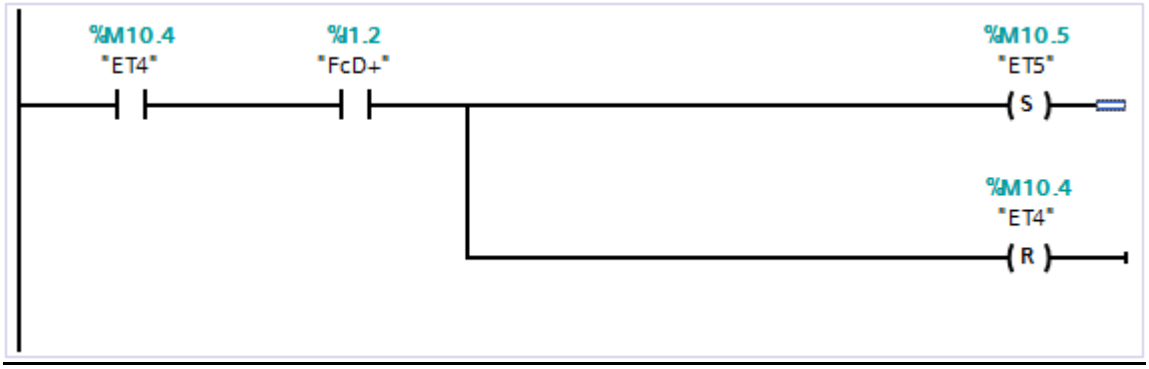


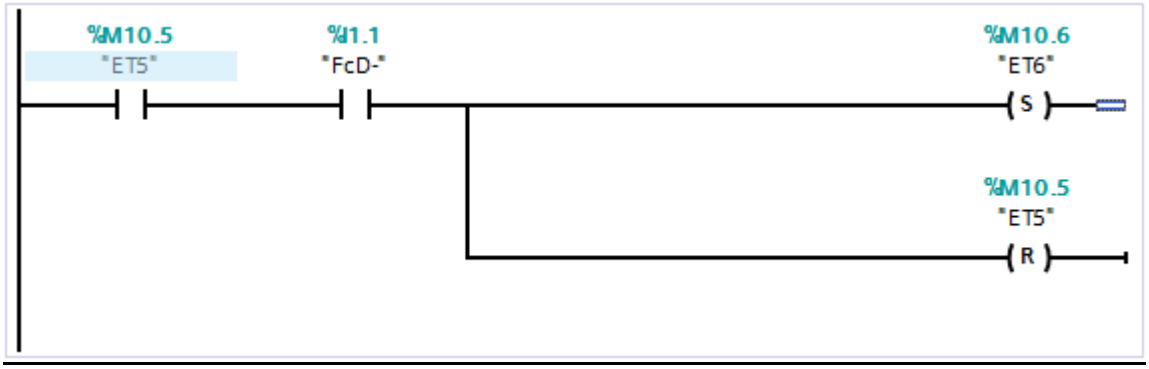
Zona secuencial

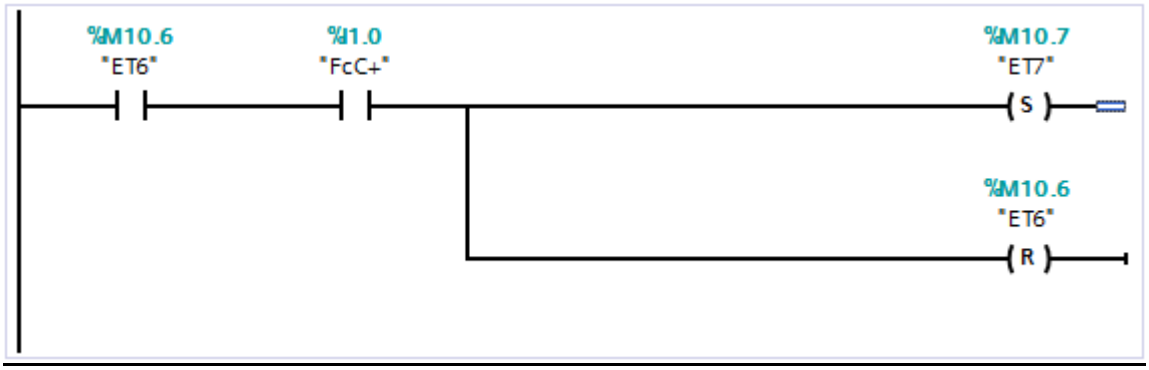


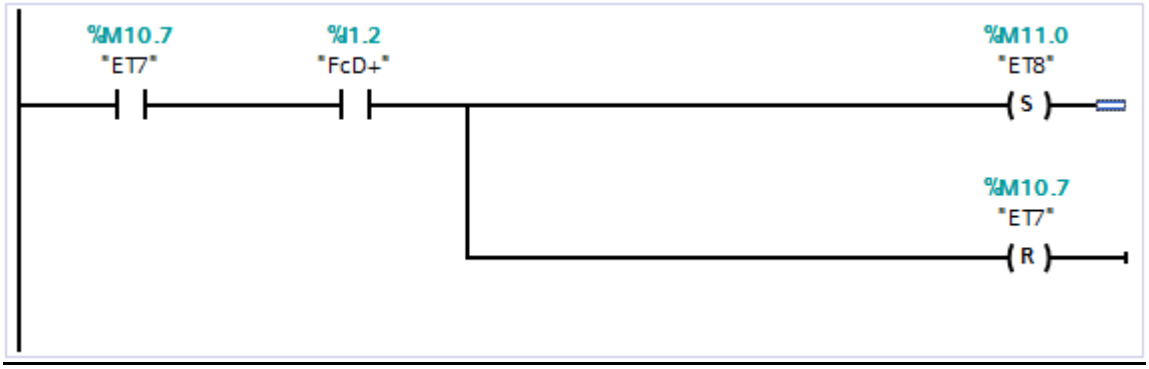


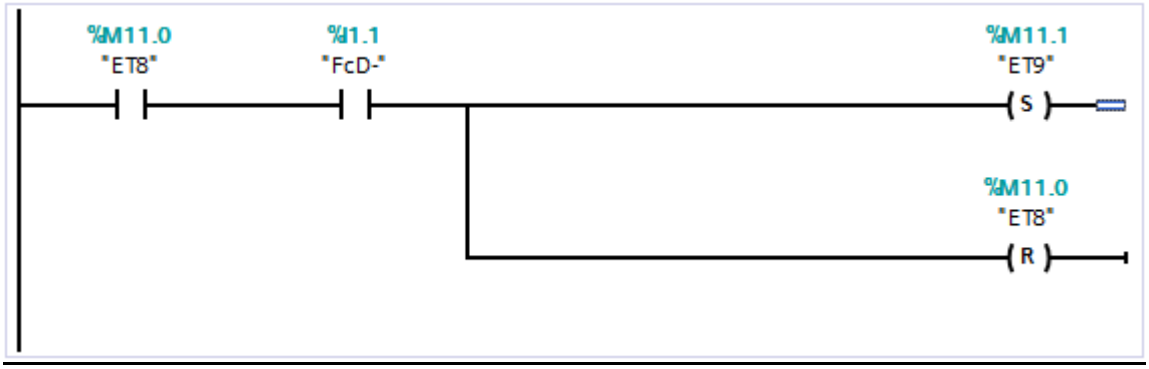

%M10.3
"ET3"
%I0.6
"FcB+"
%M10.4
"ET4"
S
%M10.3
"ET3"
R

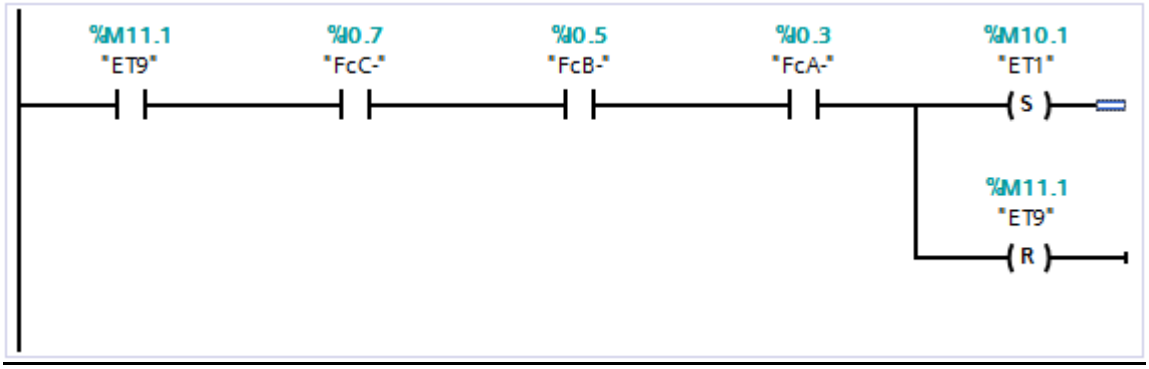

%M10.4
"ET4"
%I1.2
"FcD+"
%M10.5
"ET5"
S
%M10.4
"ET4"
R


%M10.5
"ET5"
%I1.1
"FcD-"
%M10.6
"ET6"
S
%M10.5
"ET5"
R

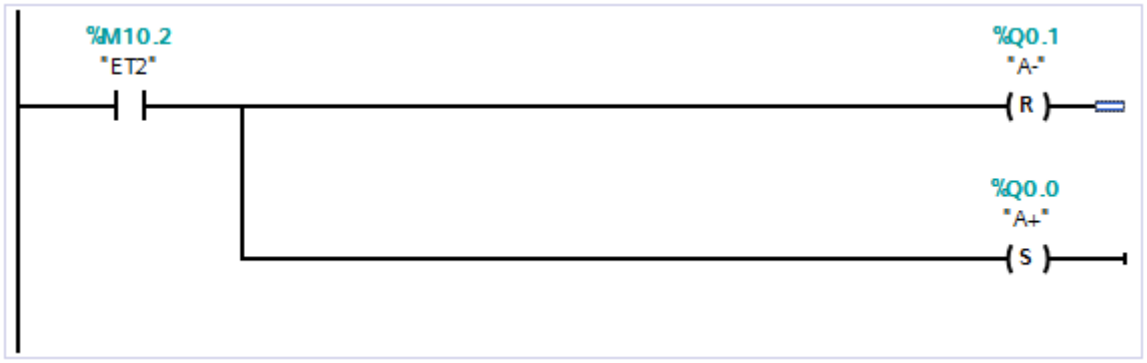

%M10.6
"ET6"
%I1.0
"FcC+"
%M10.7
"ET7"
S
%M10.6
"ET6"
R

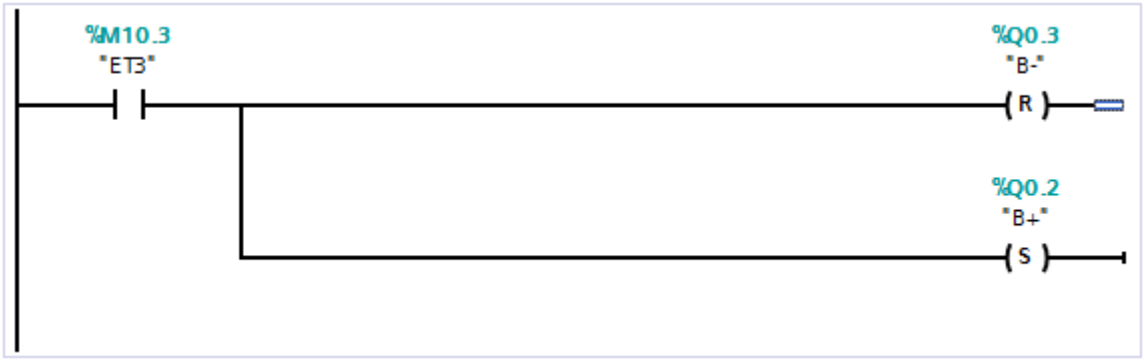


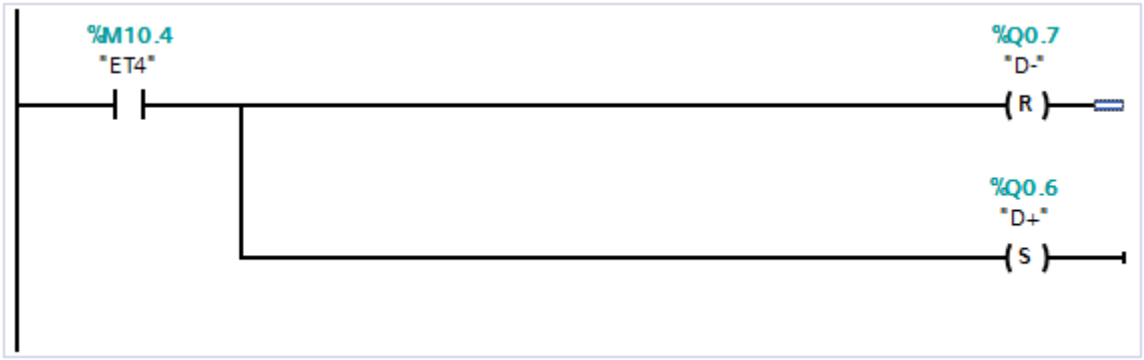


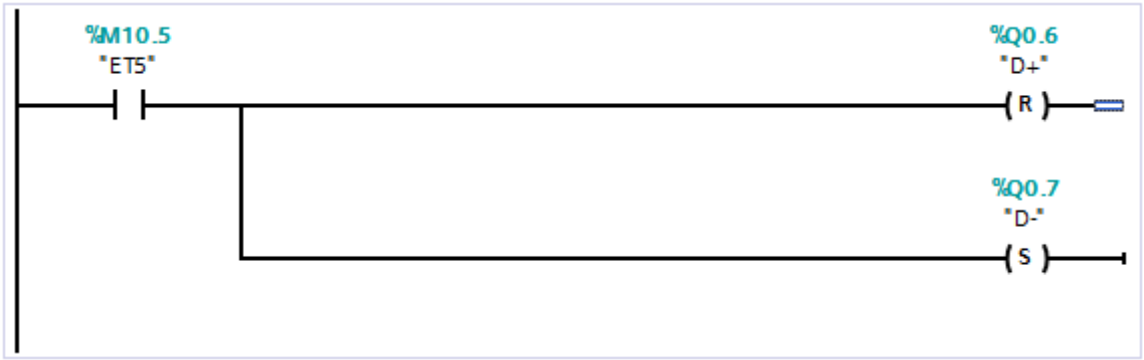


Zona de accións

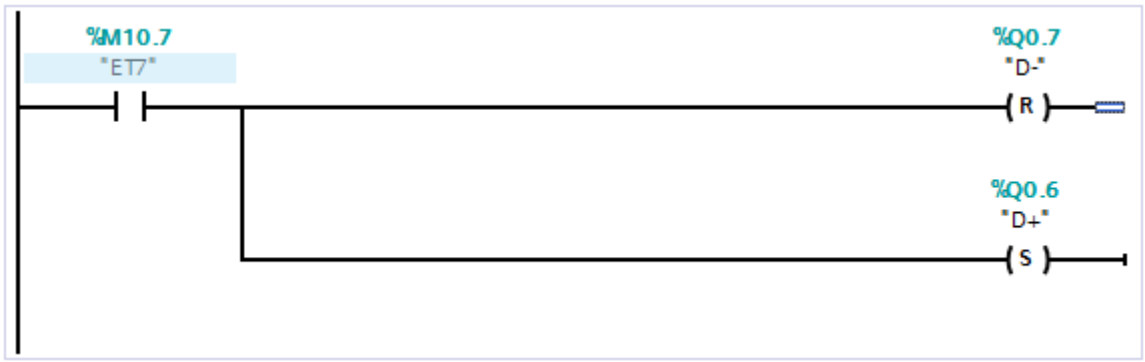


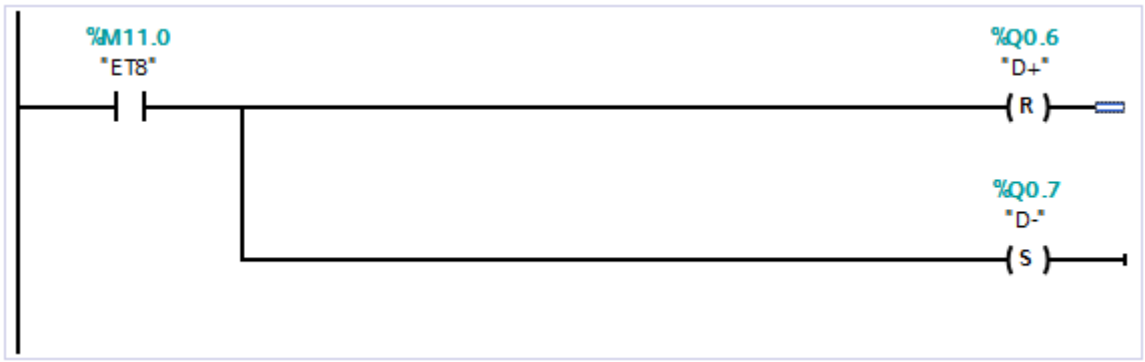

%M10.3
"ET3"
%Q0.3
"B-"
R
%Q0.2
"B+"
S

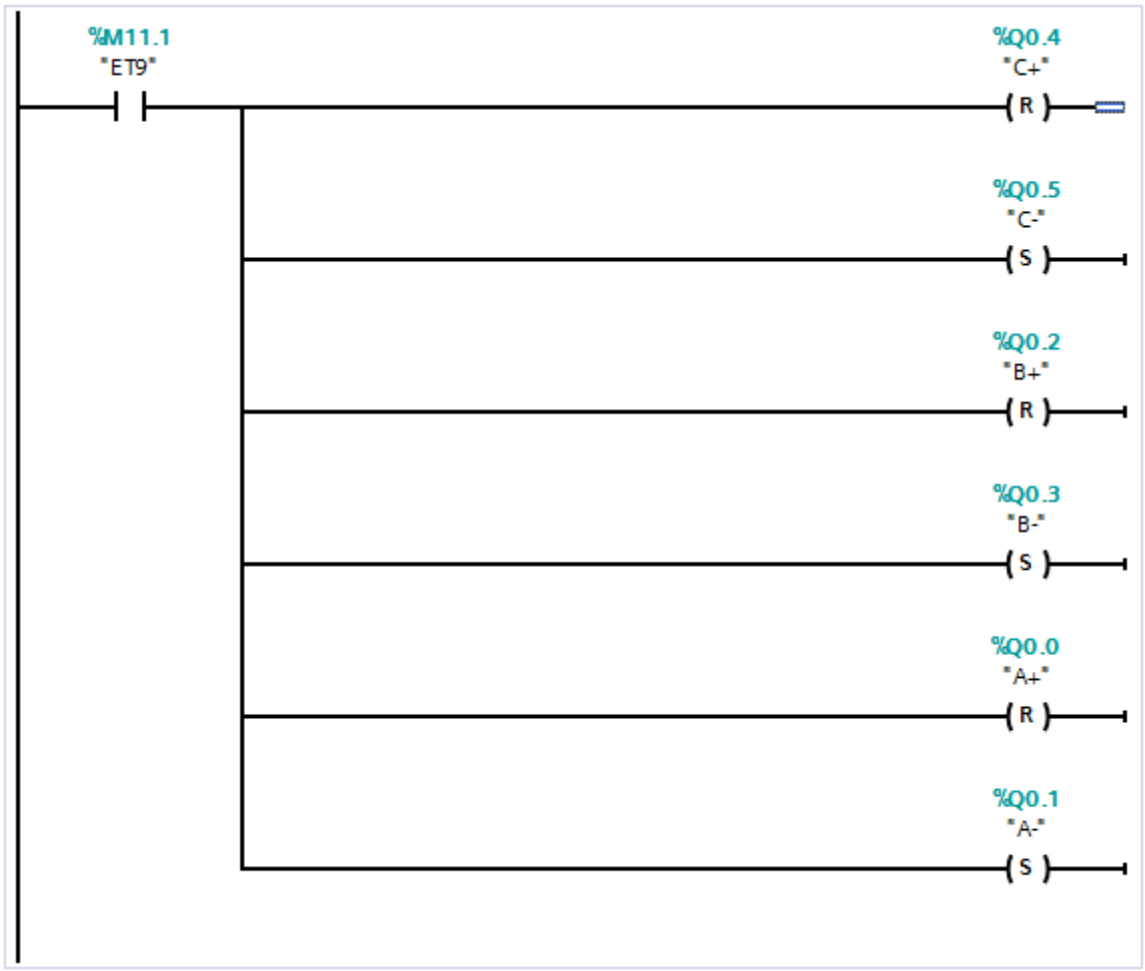

%M10.4
"ET4"
%Q0.7
"D-"
R
%Q0.6
"D+"
S


%M10.5
"ET5"
%Q0.6
"D+"
R
%Q0.7
"D-"
S


%M10.6
"ET6"
%Q0.5
"C-"
R
%Q0.4
"C+"
S


%M10.7
"ET7"
%Q0.7
"D-"
R
%Q0.6
"D+"
S

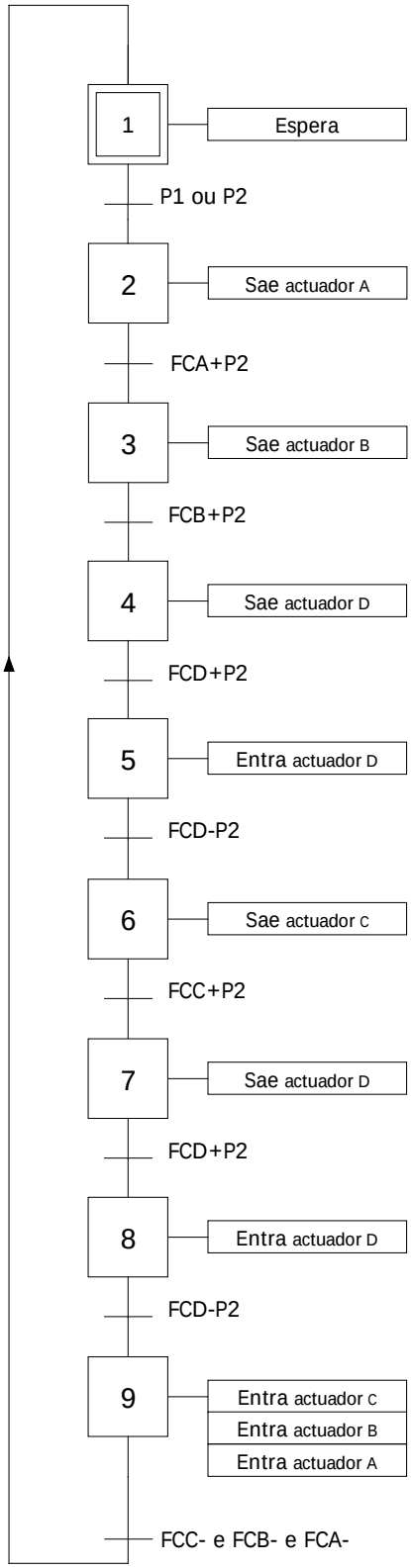

%M11.0
"ET8"
%Q0.6
"D+"
R
%Q0.7
"D-"
S


%M11.1
"ET9"
%Q0.4
"C+"
R
%Q0.5
"C-"
S
%Q0.2
"B+"
R
%Q0.3
"B-"
S
%Q0.0
"A+"
R
%Q0.1
"A-"
S

8.4.1.5 Exercicio 13

Sobre a base do exercicio 12, implementarase a posibilidade de ir executando o proceso paso a paso por etapas, cada vez que se accione o pulsador correspondente.

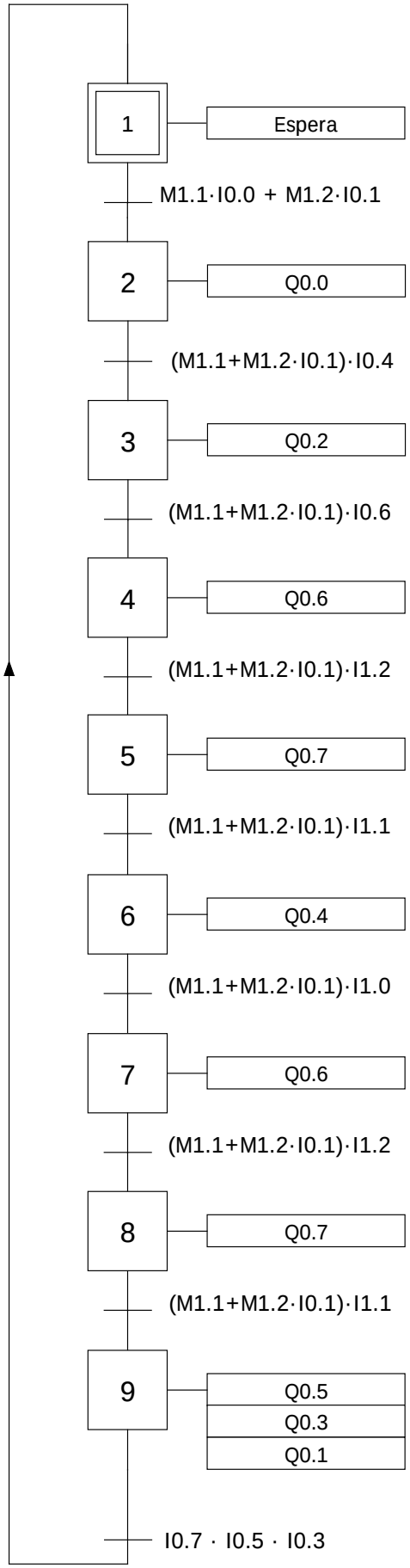
Partimos do mesmo GRAFCET e modificámolo para contemplar o paso a paso.



A táboa de entradas/saídas complétase co pulsador P2

Entradas	Descrición
I0.0	Pulsador marcha (P1)
I0.1	Pulsador paso a paso (P2)
I0.3	FcA-
I0.4	FcA+
I0.5	FcB-
I0.6	FcB+
I0.7	FcC-
I1.0	FcC+
I1.1	FcD-
I1.2	FcD+
Saídas	Descrición
Q0.0	A+
Q0.1	A-
Q0.2	B+
Q0.3	B-
Q0.4	C+
Q0.5	C-
Q0.6	D+
Q0.7	D-

E o GRAFCET de segundo nivel.


1
Espera
M1.1·I0.0 + M1.2·I0.1
2
Q0.0
(M1.1+M1.2·I0.1)·I0.4
3
Q0.2
(M1.1+M1.2·I0.1)·I0.6
4
Q0.6
(M1.1+M1.2·I0.1)·I1.2
5
Q0.7
(M1.1+M1.2·I0.1)·I1.1
6
Q0.4
(M1.1+M1.2·I0.1)·I1.0
7
Q0.6
(M1.1+M1.2·I0.1)·I1.2
8
Q0.7
(M1.1+M1.2·I0.1)·I1.1
9
Q0.5
Q0.3
Q0.1
I0.7 · I0.5 · I0.3

A táboa de etapas non varía con respecto ao exercicio anterior

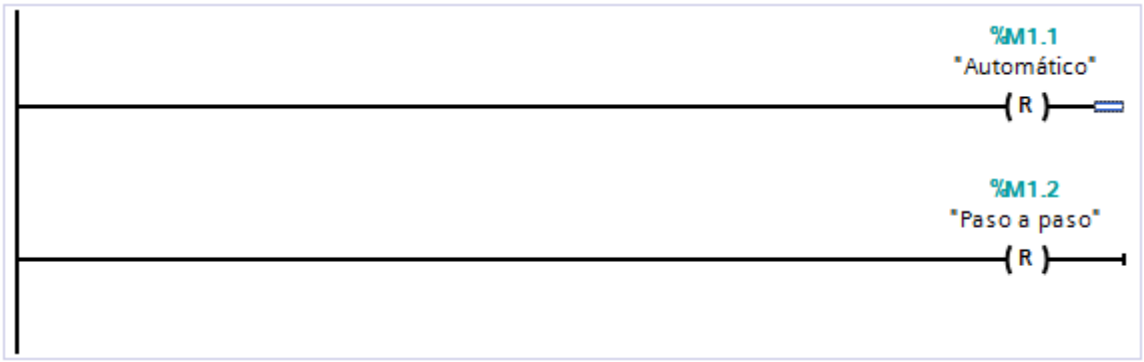
Etapa	Marca	Simbólico
1	M10.1	ET1
2	M10.2	ET2
3	M10.3	ET3
4	M10.4	ET4
5	M10.5	ET5
6	M10.6	ET6
7	M10.7	ET7
8	M11.0	ET8
9	M11.1	ET9

Neste caso ímonos axudar de dúas marcas internas que diferenciarán a posición de automático (M1.1) da de paso a paso (M1.2).

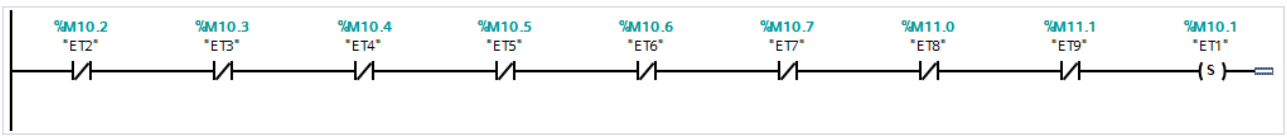
Faremos varias operacións previas. Primeiro nunha subrutina de inicialización (OB100) resetamos as marcas M1.1 e M1.2.

O programa quedaría:

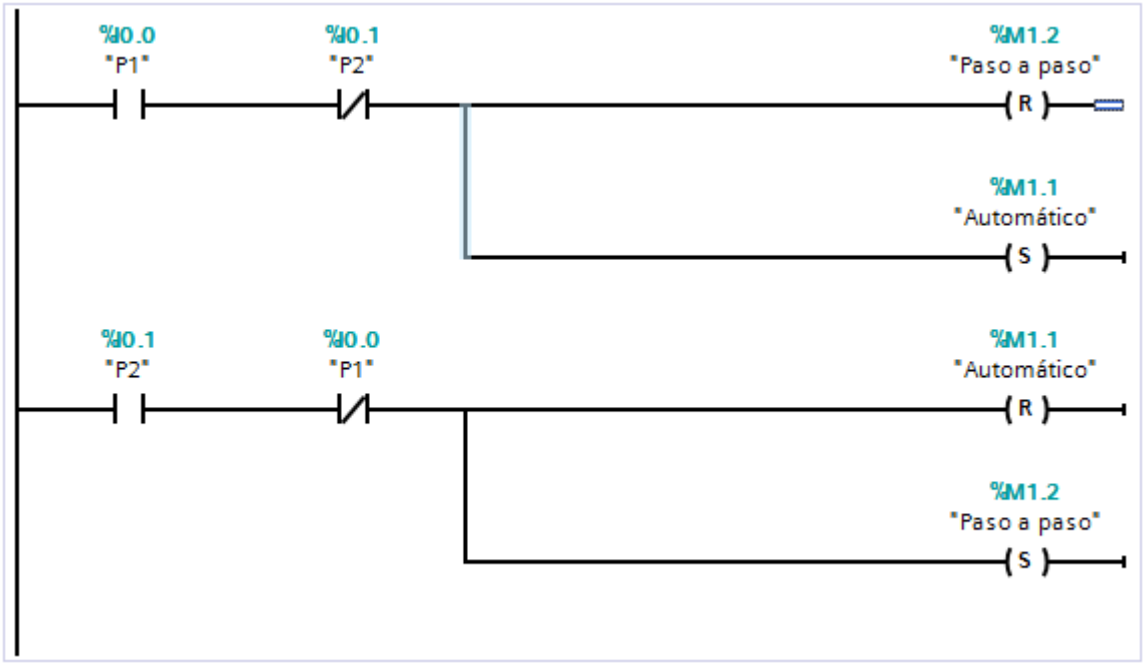
Incialización (OB100)



Zona preliminar

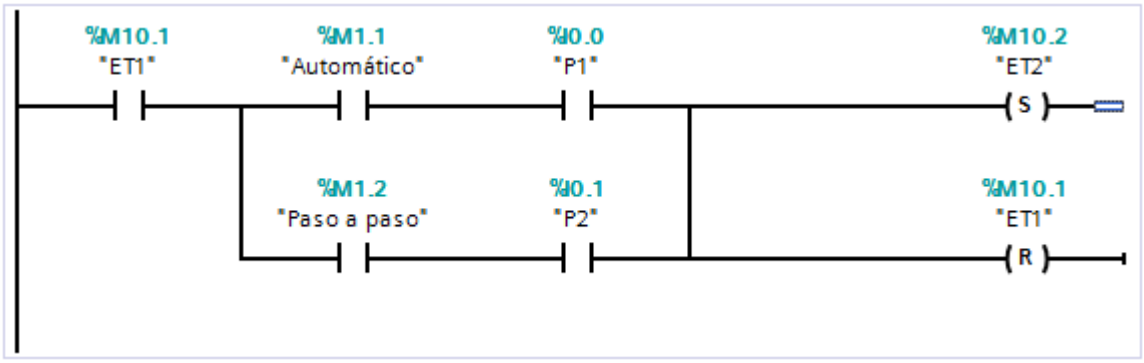


A diferencia co exercicio anterior, dependendo do pulsador que accionemos (metemos un enclavamento entre os dous para evitar simultaneidade), activamos a marca de automático (M1.1) ou paso a paso (M1.2).

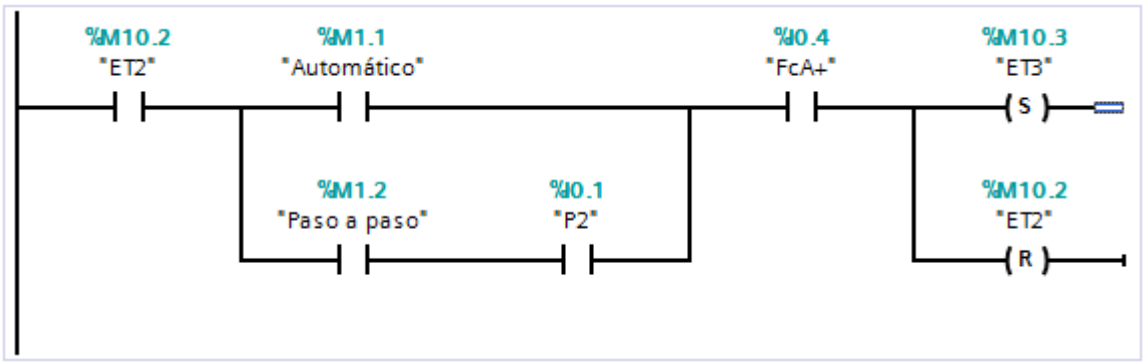


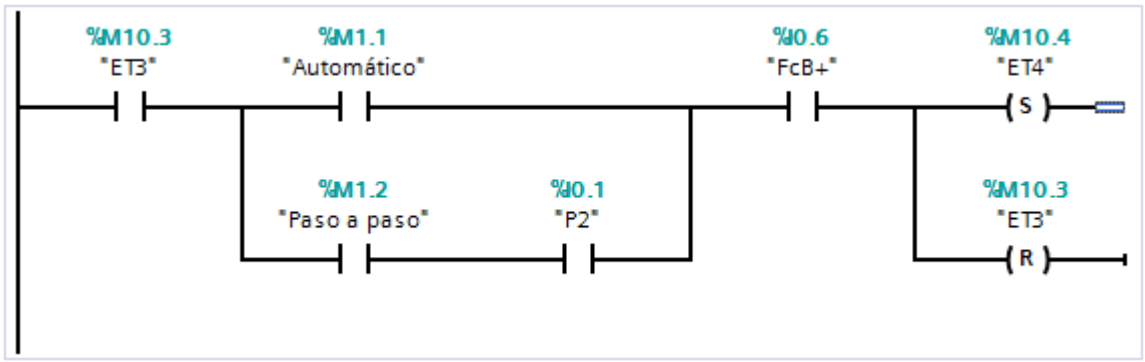
Zona secuencial

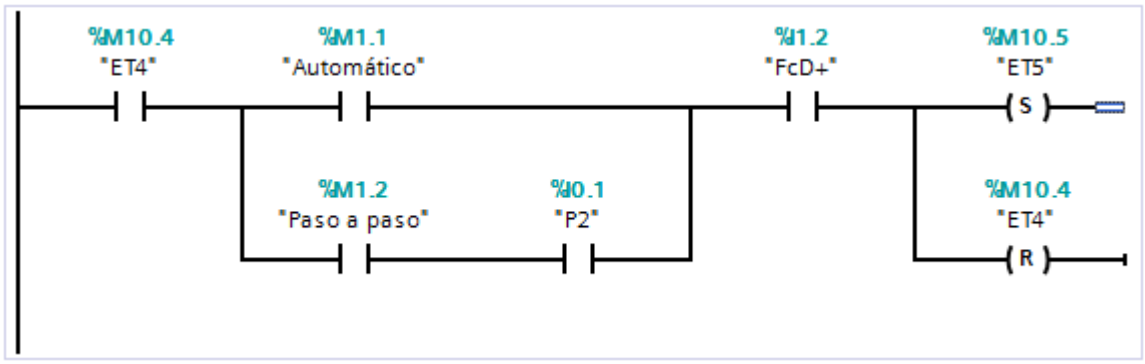
Como estamos na etapa inicial (ET1), en calquera caso pasamos á seguinte etapa.

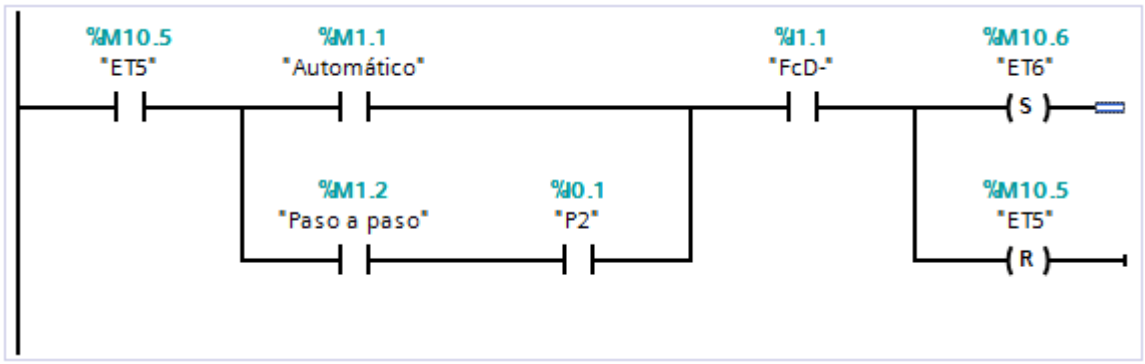


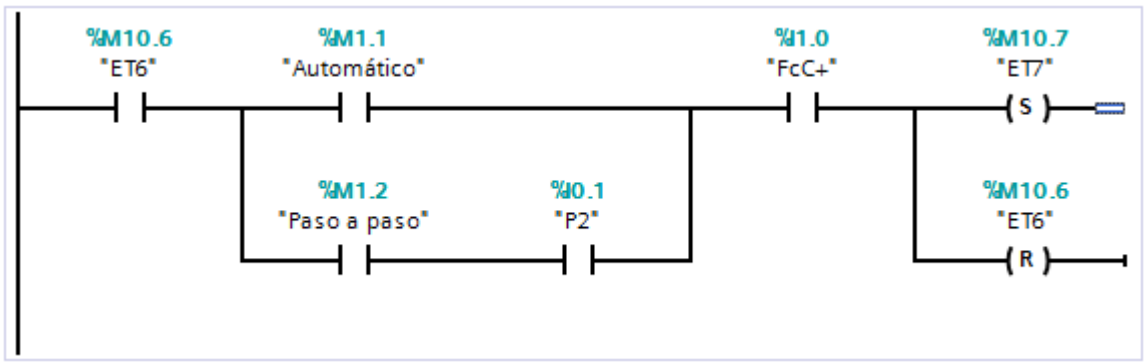
Agora, todas as transicións son similares. Coa etapa anterior (M10.2), se estamos en modo automático, evoluciona como no exercicio anterior, e se estamos en paso a paso, cada vez que pulsamos P2 (I0.1) imos tamén avanzando etapas.

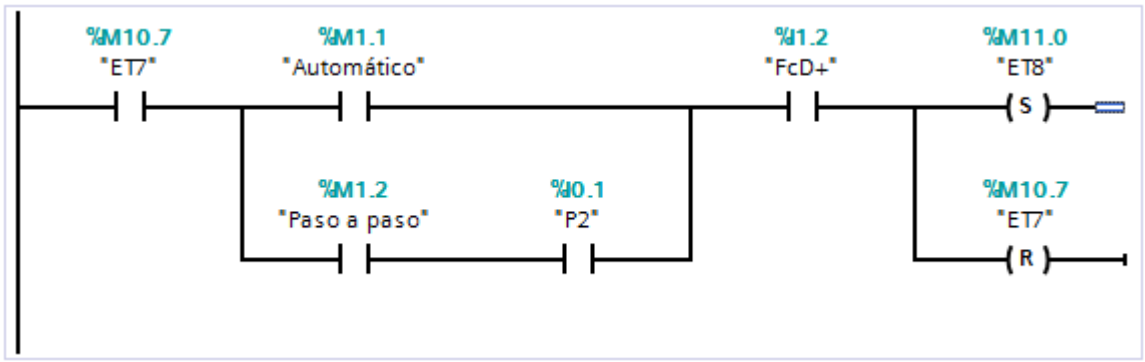


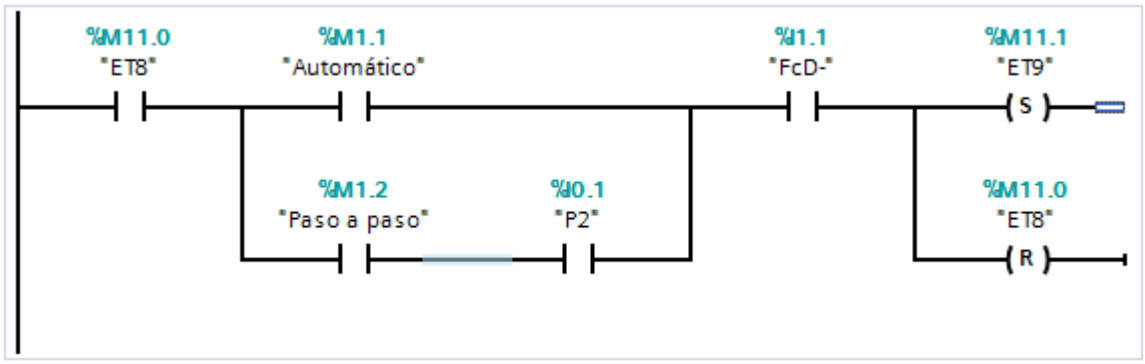

%M10.3
"ET3"
%M1.1
"Automático"
%I0.6
"FcB+"
%M10.4
"ET4"
S
%M1.2
"Paso a paso"
%I0.1
"P2"
%M10.3
"ET3"
R

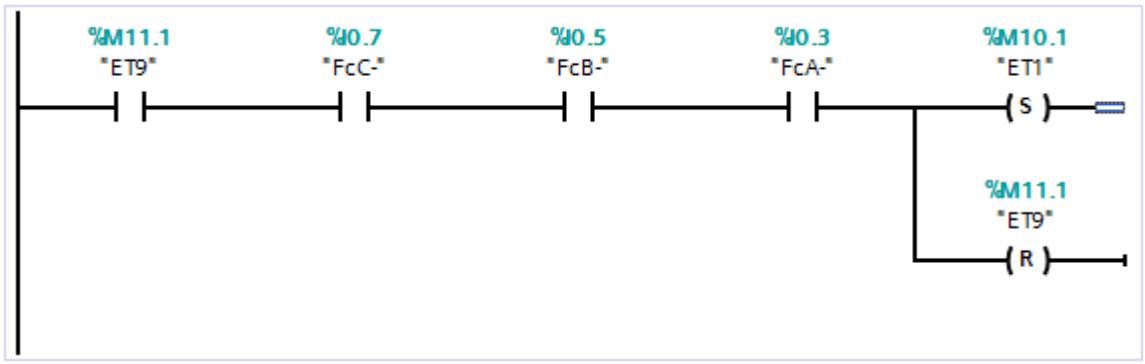

%M10.4
"ET4"
%M1.1
"Automático"
%I1.2
"FcD+"
%M10.5
"ET5"
S
%M1.2
"Paso a paso"
%I0.1
"P2"
%M10.4
"ET4"
R


%M10.5
"ET5"
%M1.1
"Automático"
%I1.1
"FcD-"
%M10.6
"ET6"
S
%M1.2
"Paso a paso"
%I0.1
"P2"
%M10.5
"ET5"
R


%M10.6
"ET6"
%M1.1
"Automático"
%I1.0
"FcC+"
%M10.7
"ET7"
S
%M1.2
"Paso a paso"
%I0.1
"P2"
%M10.6
"ET6"
R

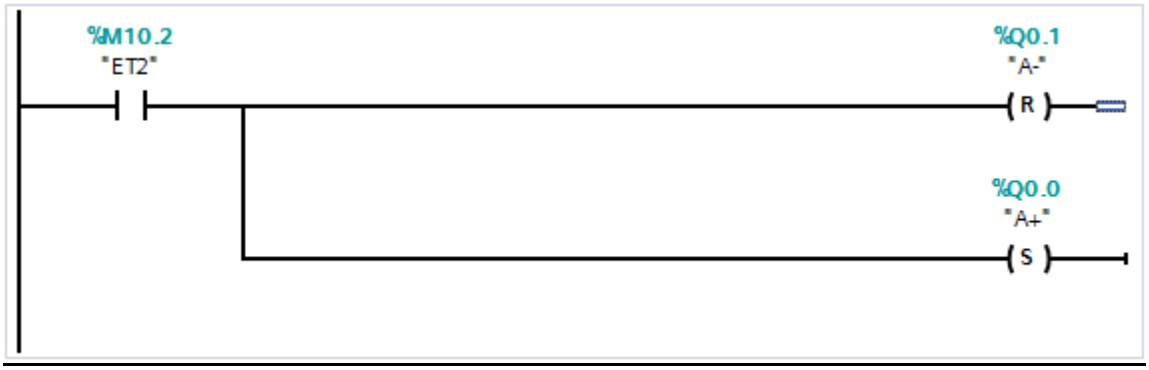


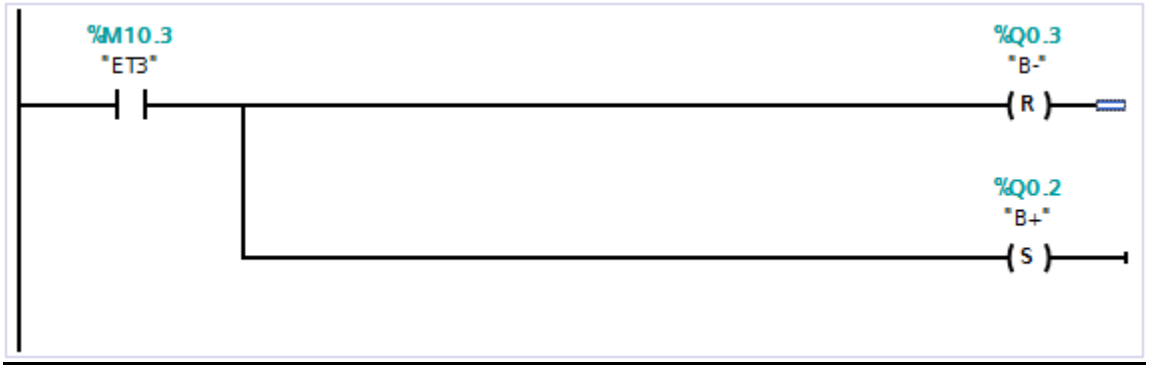


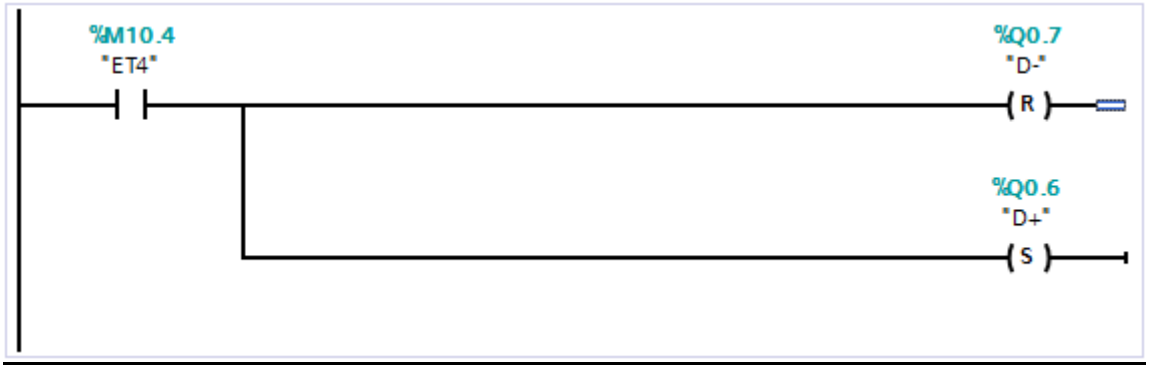


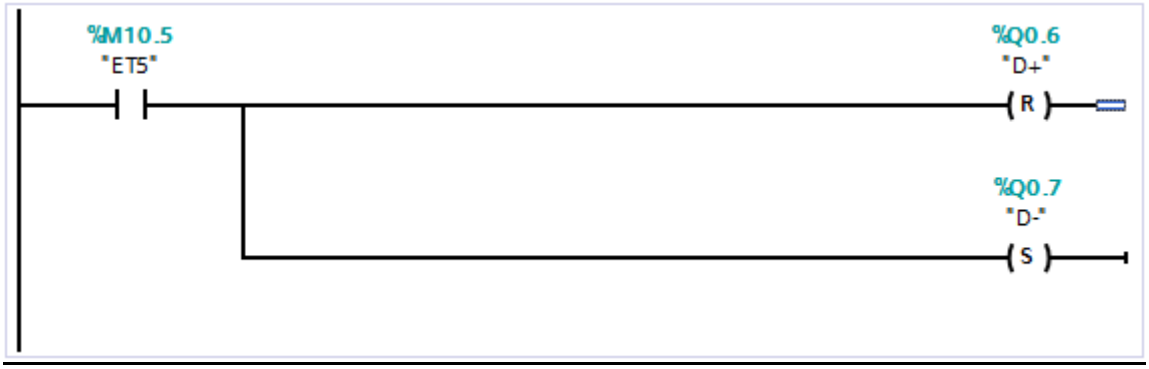
Zona de accións

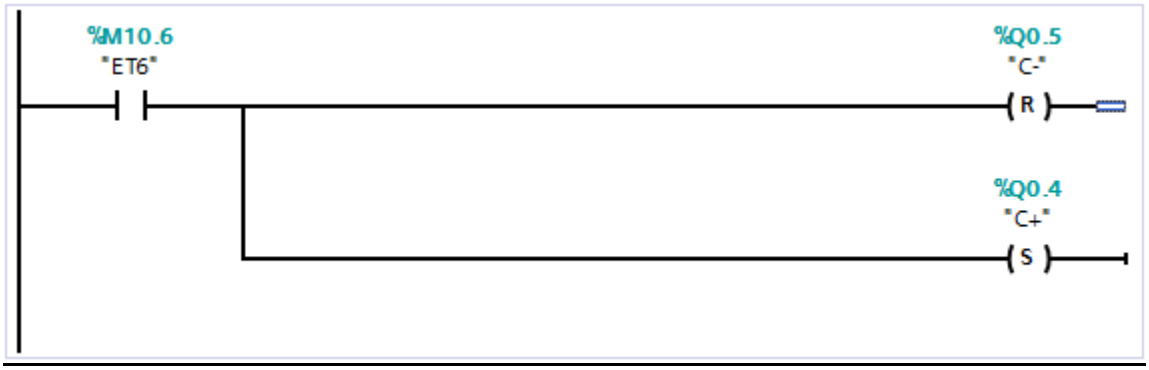
Na zona de accións non hai variación con respecto ao exercicio anterior

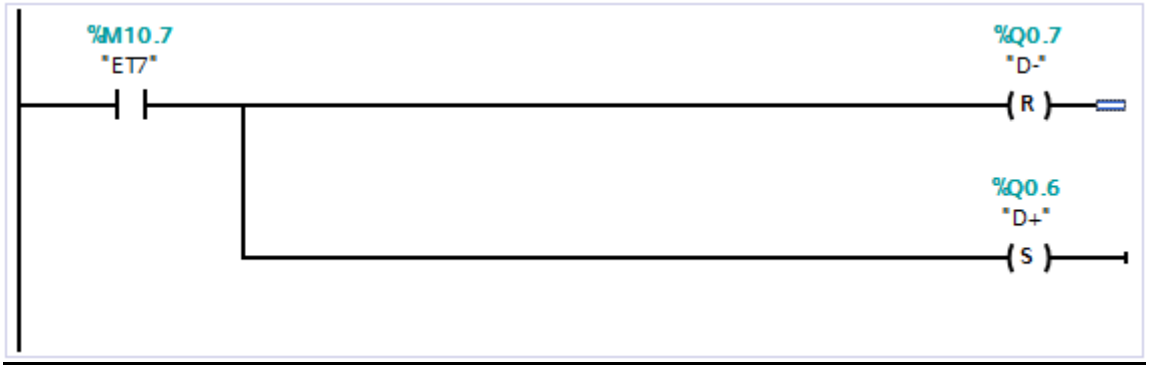


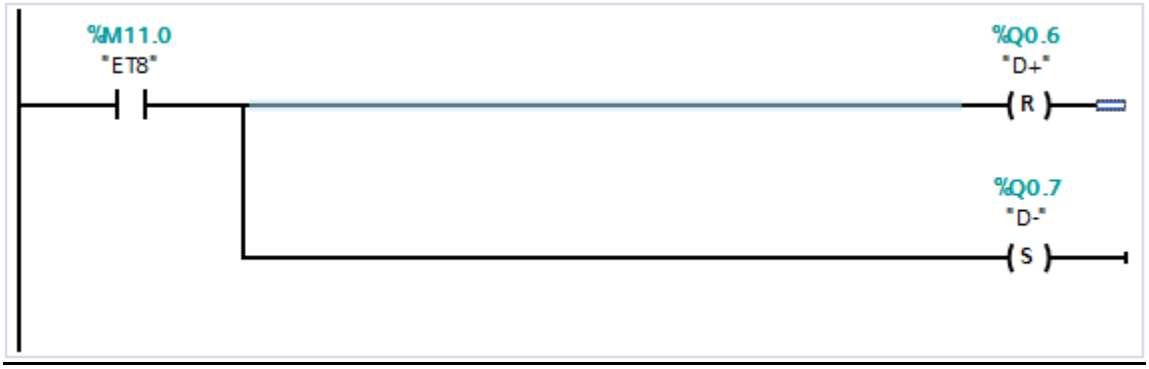

%M10.3
"ET3"
%Q0.3
"B-"
R
%Q0.2
"B+"
S

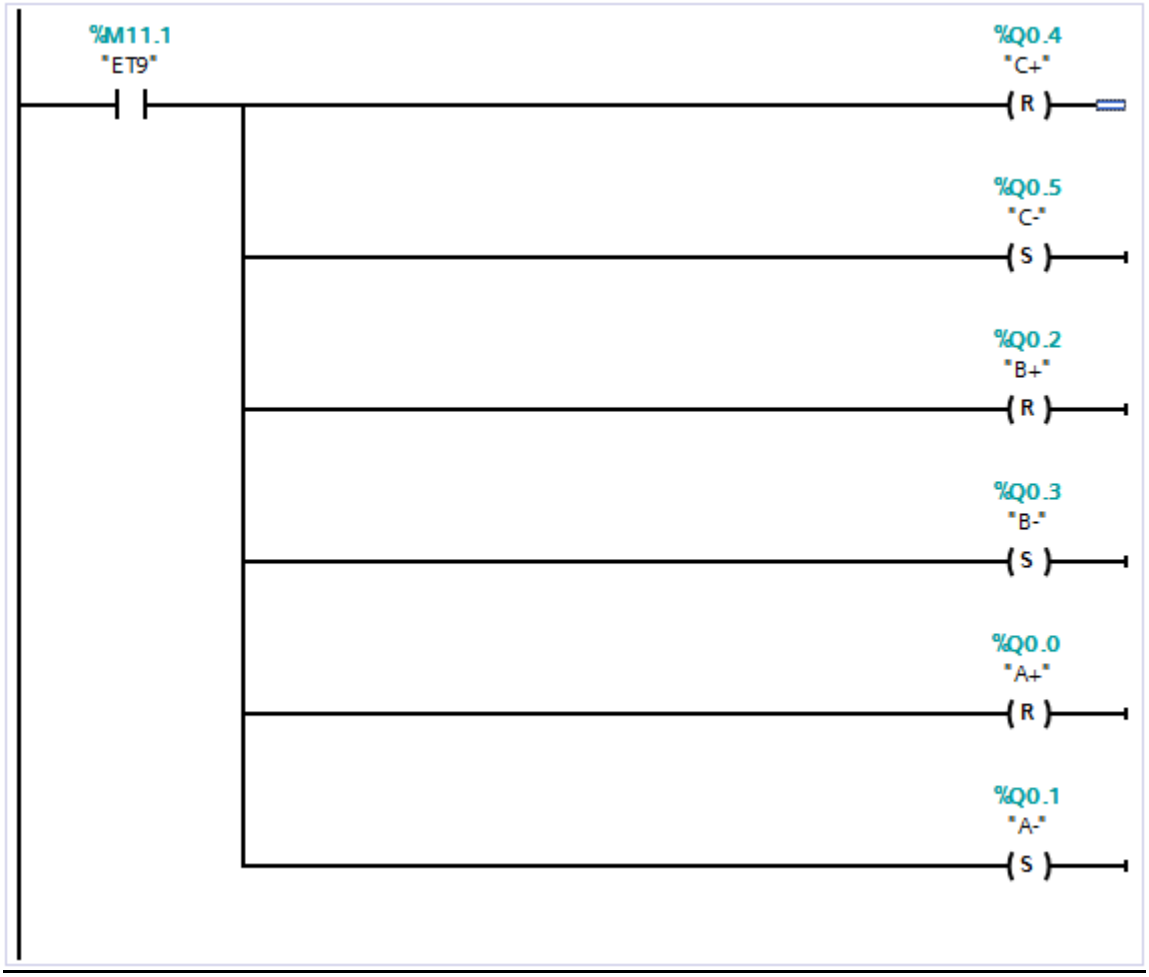

%M10.4
"ET4"
%Q0.7
"D-"
R
%Q0.6
"D+"
S


%M10.5
"ET5"
%Q0.6
"D+"
R
%Q0.7
"D-"
S


%M10.6
"ET6"
%Q0.5
"C-"
R
%Q0.4
"C+"
S


%M10.7
"ET7"
%Q0.7
"D-"
R
%Q0.6
"D+"
S

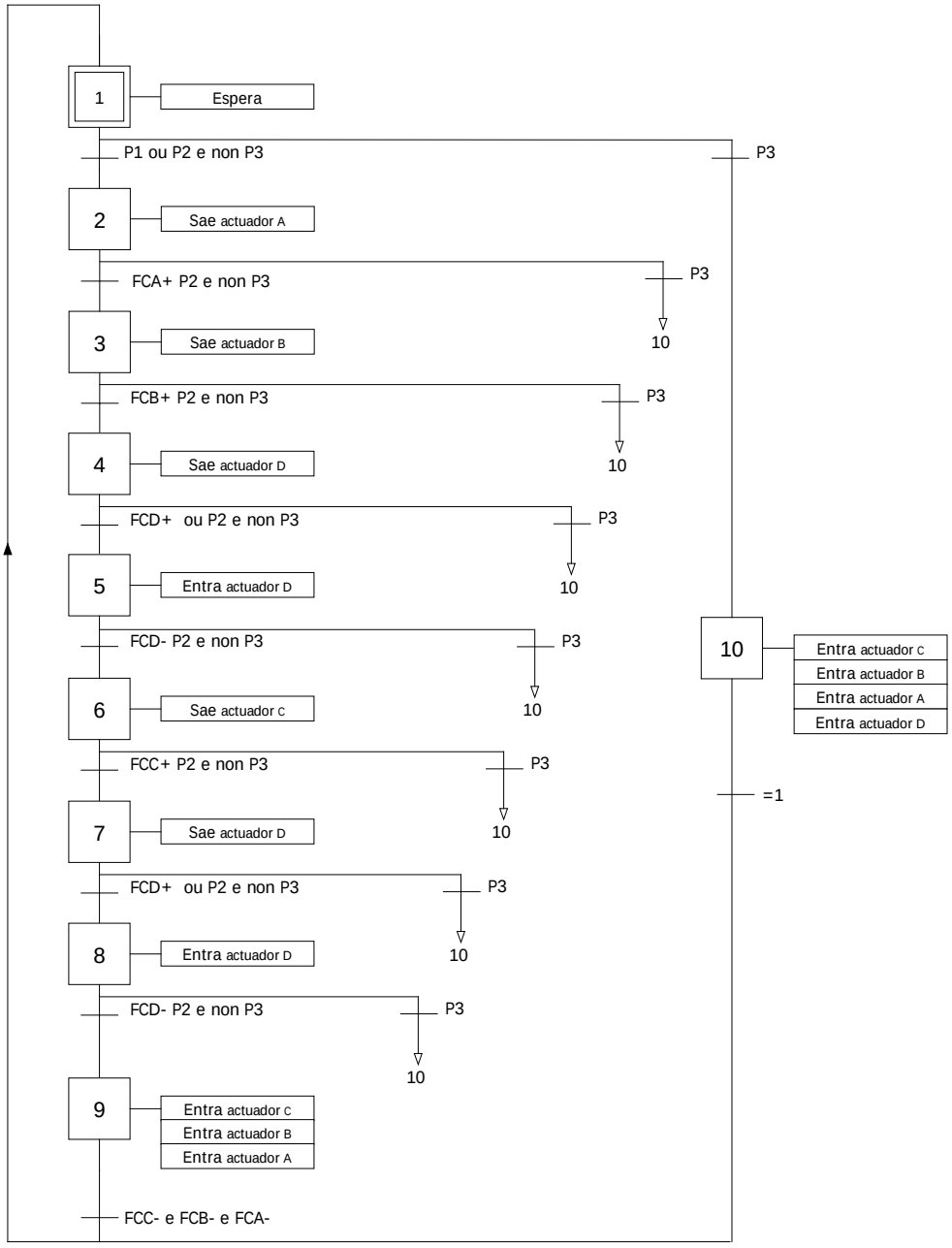

%M11.0
"ET8"
%Q0.6
"D+"
R
%Q0.7
"D-"
S


%M11.1
"ET9"
%Q0.4
"C+"
R
%Q0.5
"C-"
S
%Q0.2
"B+"
R
%Q0.3
"B-"
S
%Q0.0
"A+"
R
%Q0.1
"A-"
S

8.4.1.6 Exercicio 14

Repetiremos o exercicio anterior, incorporando unha parada de emerxencia na que, independentemente da etapa na que nos atopemos, retornaremos á etapa incial cos actuadores todos en posición de repouso.

Poderiamos prantexar o GRAFCET como o seguinte, onde P3 é o pulsador de emerxencia.

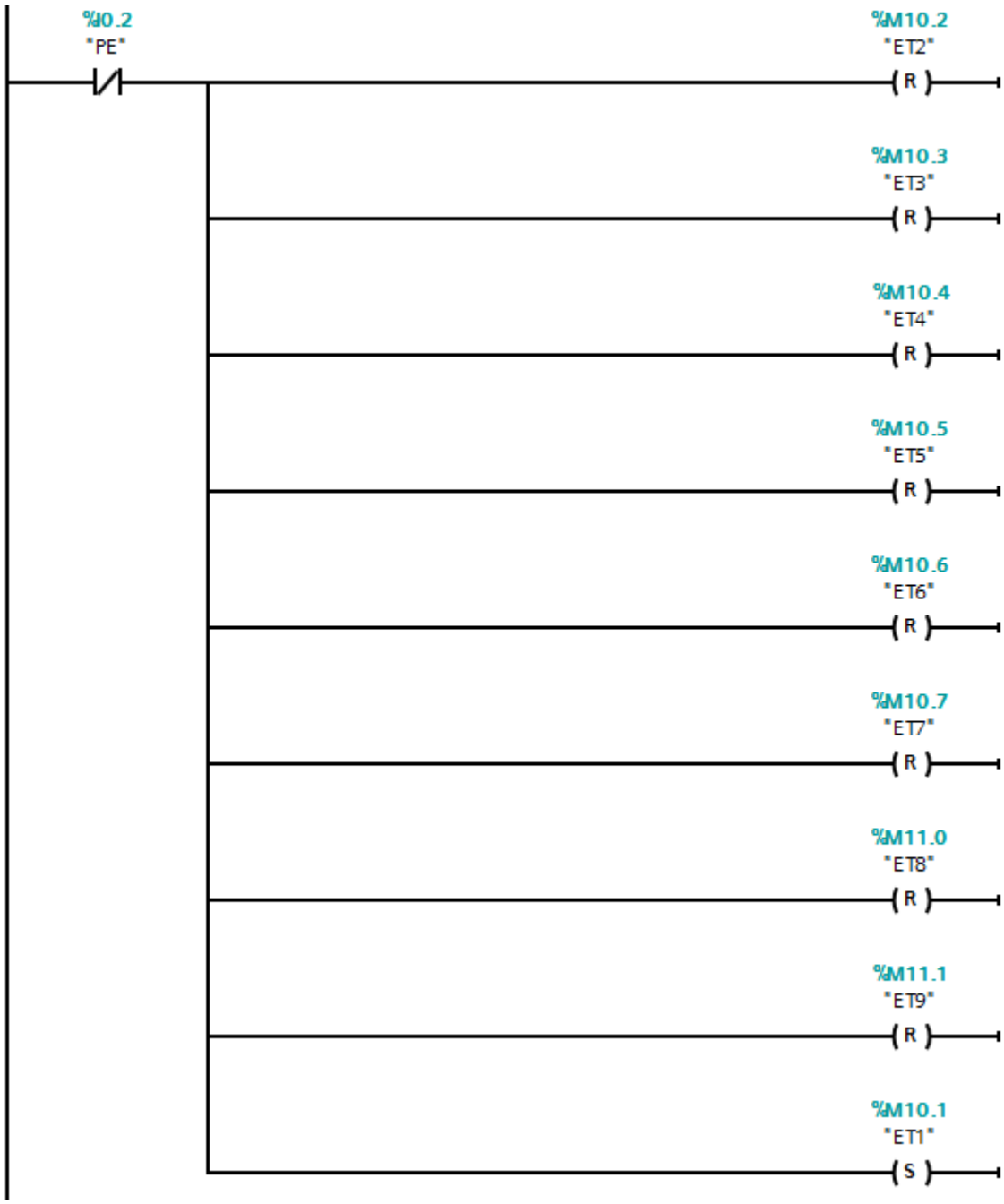


A táboa de entradas/saídas sería:

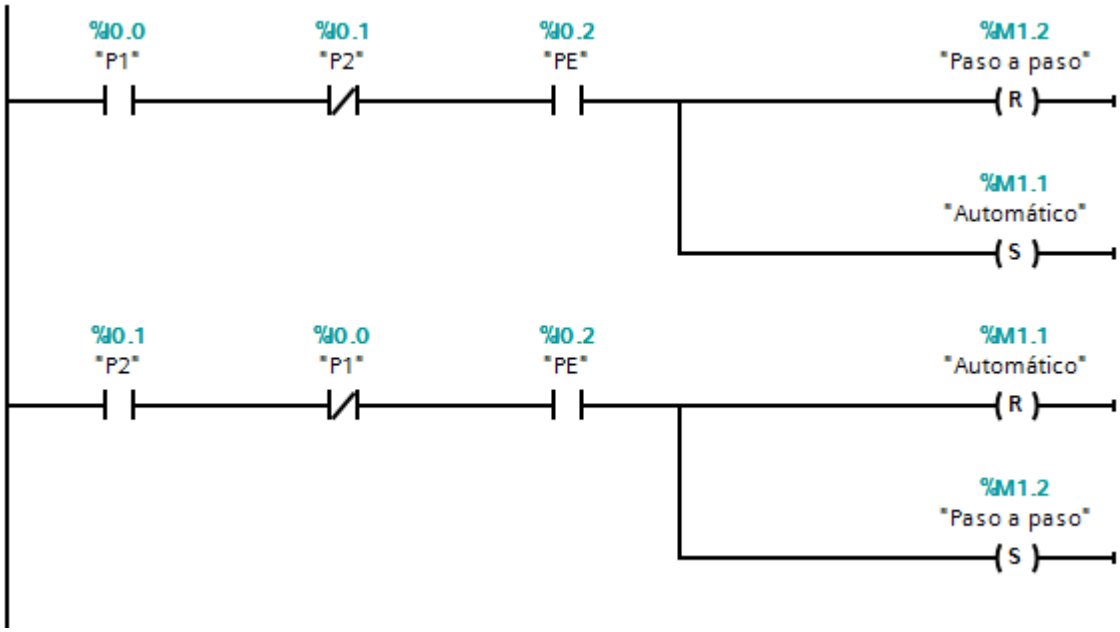
Entradas	Descrición
I0.0	Pulsador marcha NO(P1)
I0.1	Pulsador paso a paso NO(P2)
I0.2	Pulsador de emerxencia NC(P3)

I0.3	FcA-
I0.4	FcA+
I0.5	FcB-
I0.6	FcB+
I0.7	FcC-
I1.0	FcC+
I1.1	FcD-
I1.2	FcD+
Saídas	Descrición
Q0.0	A+
Q0.1	A-
Q0.2	B+
Q0.3	B-
Q0.4	C+
Q0.5	C-
Q0.6	D+
Q0.7	D-

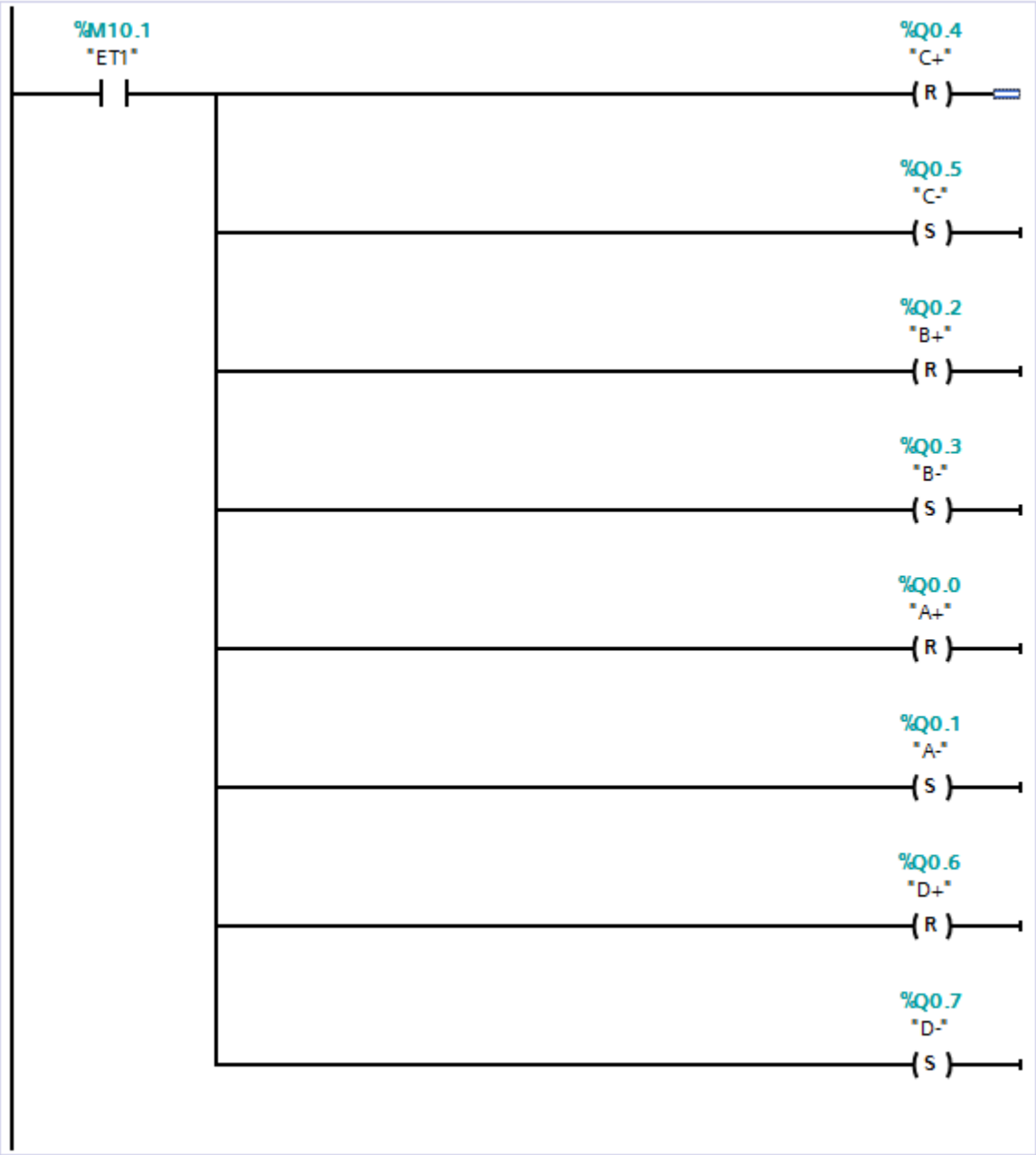
Na zona preliminar, ao principio poñemos as condicións de emerxencia, que consisten en resetear todas as etapas e activar a inicial.



Ademáis enclavamos os pulsadores de automático e paso a paso co pulsador de emerxencia.



Na zona de accionamentos, engadimos as condicións para a etapa inicial, que nos anteriores exercicios non tiñamos.

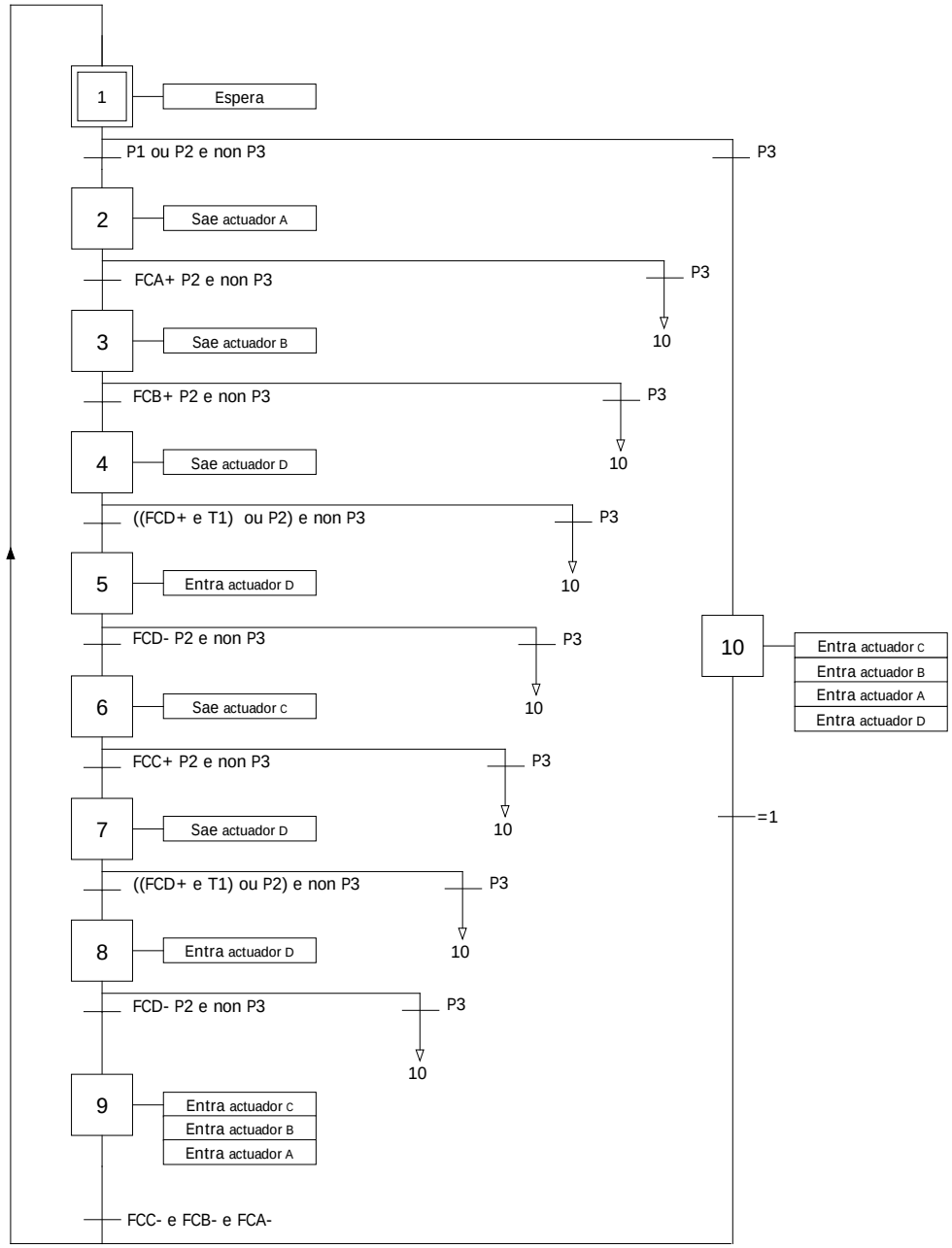


O resto do programa sería o mesmo do exercicio anterior.

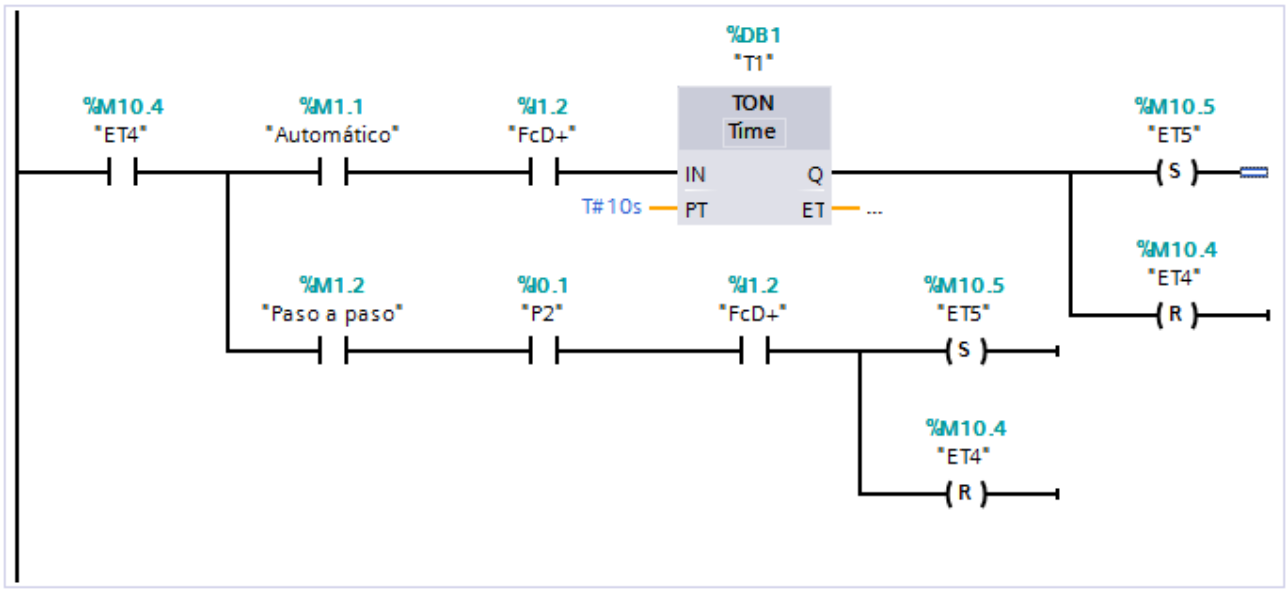
8.4.1.7 Exercicio 15

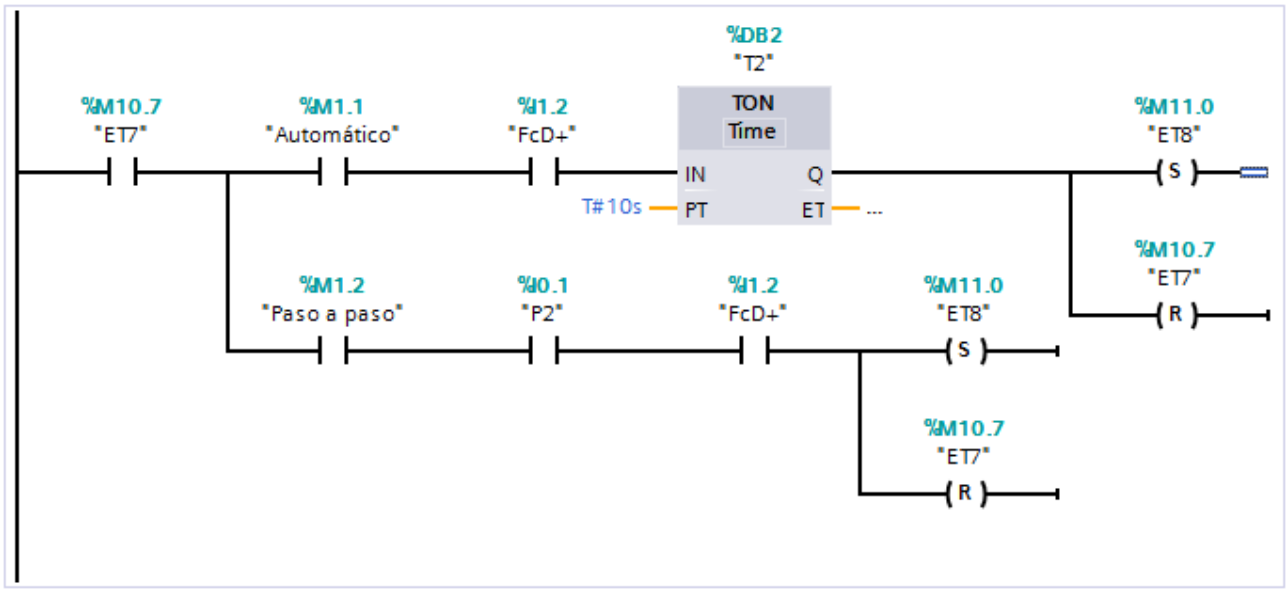
Sobre a base do exercicio anterior, introduciremos unha modificación de xeito que o actuador que fai a estampación (D) cada vez que saia estea 10 segundos accionado antes de voltar á posición inicial.

Para eso temos que incorporar un temporizador no GRAFCET (Transicións das etapas 4 e 7).



As modificacións consistirían en introducir un temporizador (só ten sentido en modo automático) e permitir igualmente o funcionamento paso a paso, polo que as transicións modificadas no GRAFCET (4 e 7) quedarían do seguinte xeito.





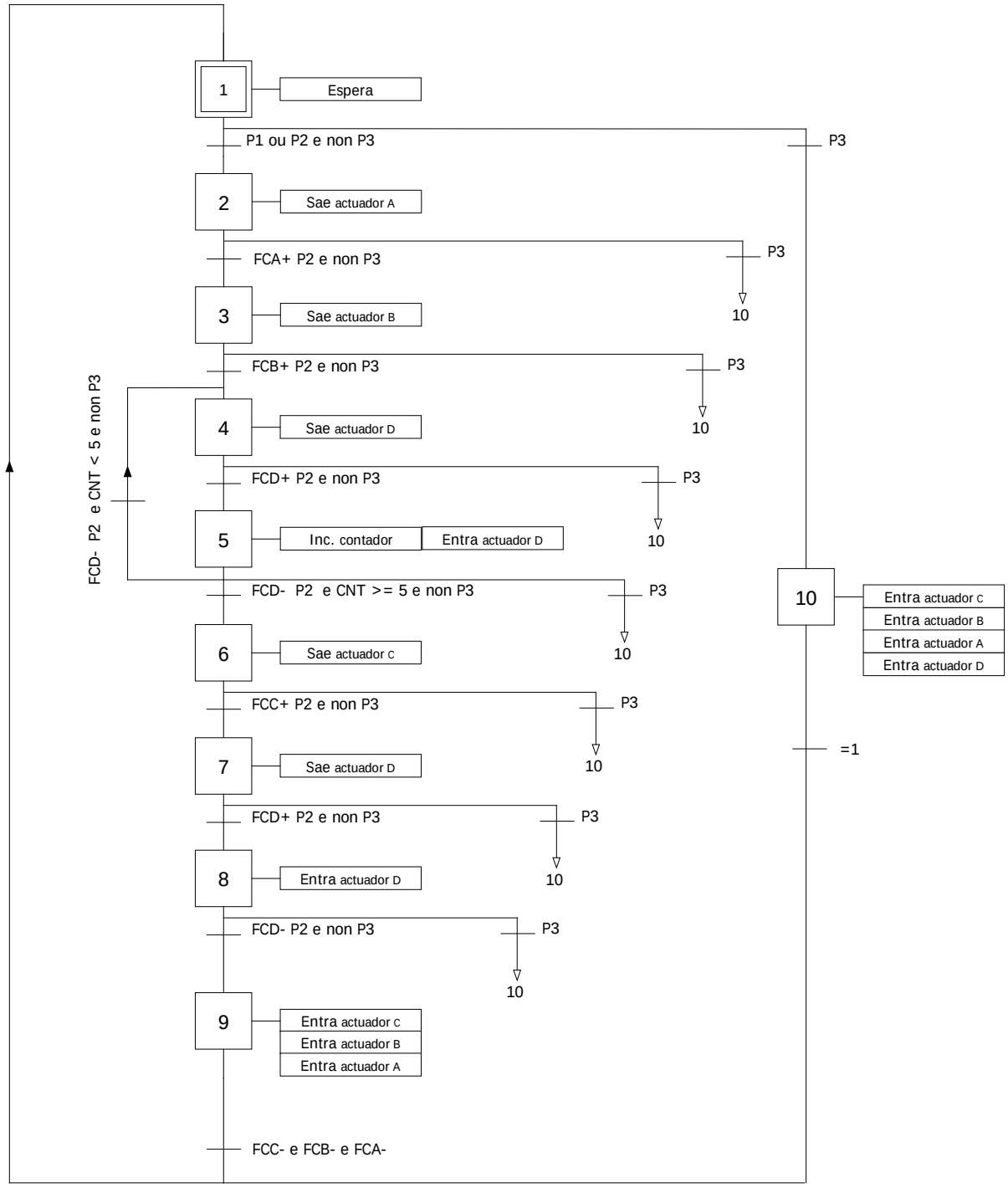
8.4.1.8 Exercicio 16

Sobre a base do exercicio 14 faremos as modificacións para que, a primeira vez que se accione o actuador D, faga 5 ciclos de saída/entrada (como 5 golpes) antes de continuar o ciclo do programa.

Neste caso, aparece no GRAFCET o concepto de salto condicional, que nos leva a unha etapa anterior ou posterior en función do contador.

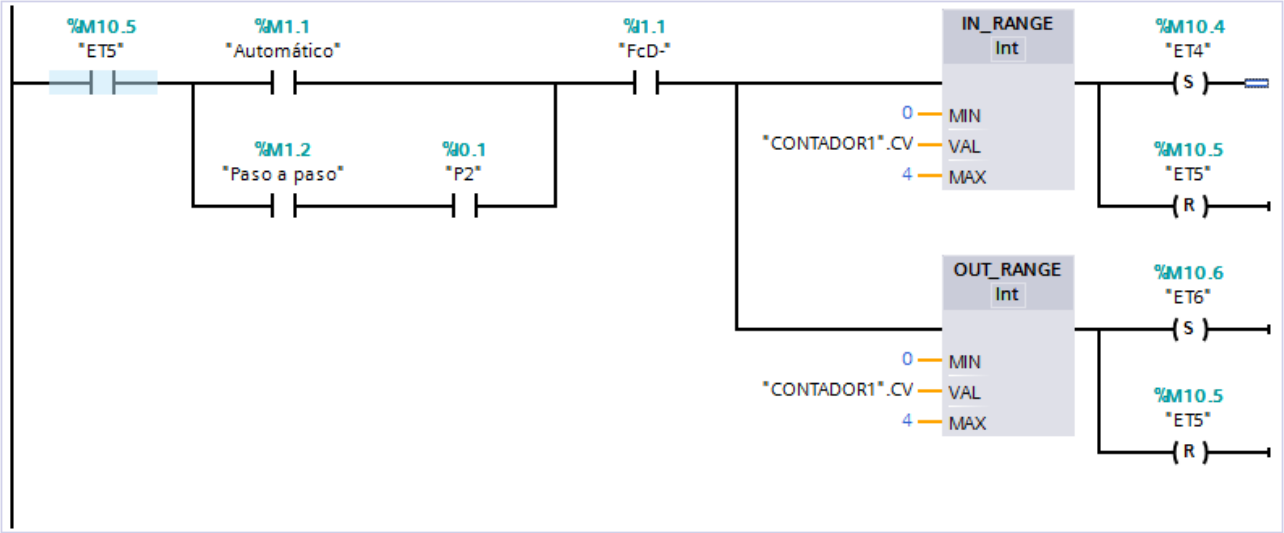
Ao chegar á etapa 5 por primeira vez (primeiro golpe) o programa terá que decidir se voltar á etapa 4 (voltar a saír o actuador) ou continuar coa etapa 6.

O GRAFCET sería algo como o seguinte:



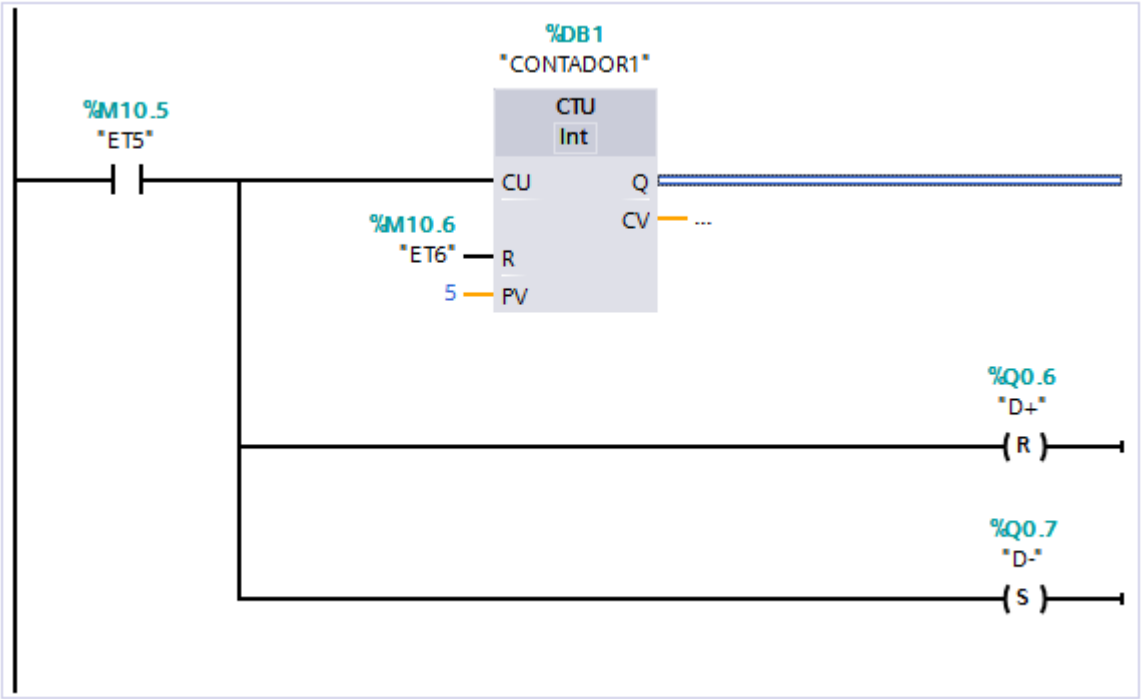
Os cambios a realizar con respecto ao exercicio 14 serían:

Na zona de transicións, na transición da etapa 5 introduciriamos o comparador que decide se voltar atrás ou seguir.



Na zona de accións, na etapa 5 incrementariamos o contador.

Observemos que empregamos a marca da seguinte etapa (M10.6) para resetear o contador. Se non o facemos, no seguinte ciclo de programa non funcionaría.

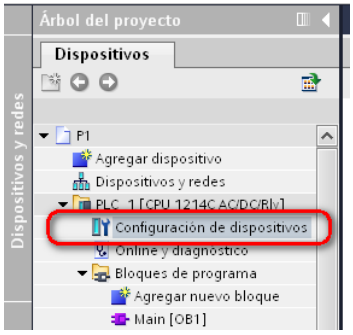


Exercicio 10_2: Como exercicio de aplicación de GRAFCET, repítase o exercicio 10 empregando este método.

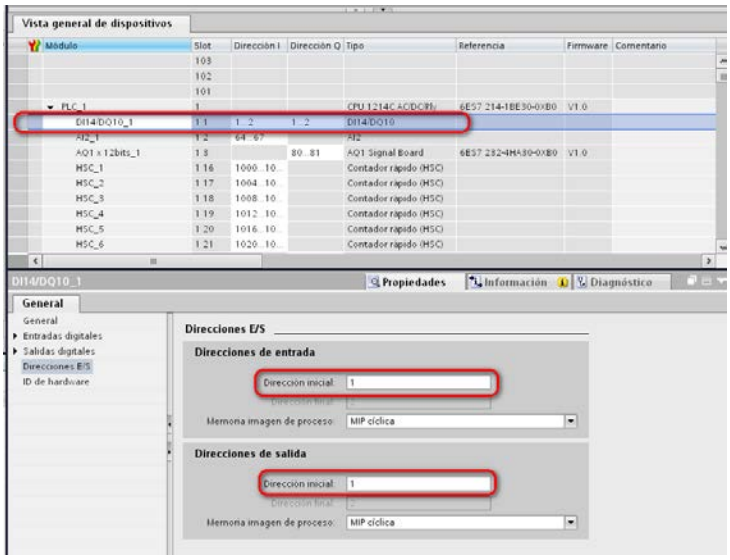
9 Parametrización do PLC

9.1.1 Cambio de direccións de entradas/saídas

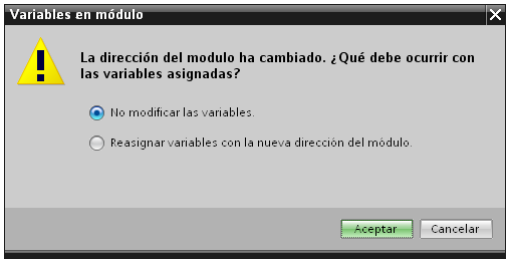
Como se dixo nun punto anterior, a diferencia do S7-200, podemos cambiar as direccións das entradas e/ou saídas. Para facelo seleccionamos a opción **Configuración de dispositivos** dentro do PLC que pretendemos cambiar.



Seleccionamos **DI14/DQ10** na vista xeral de dispositivos, e cambiamos as direccións na ventá de propiedades.



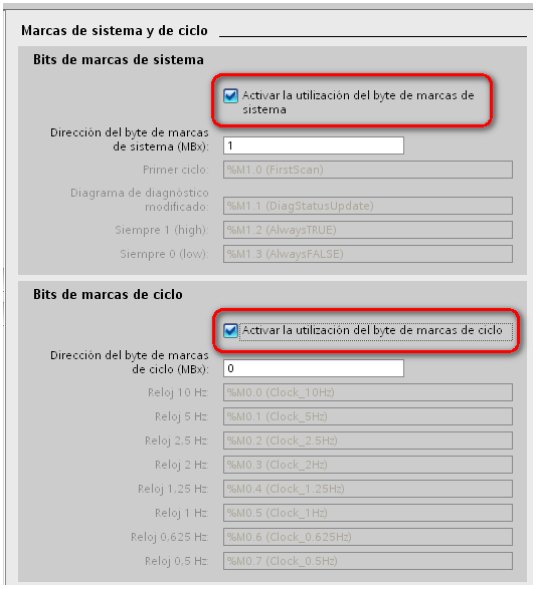
Se tiveramos xa un programa feito, preguntaríanos se queremos cambiar as asignacións no programa ou non.



9.1.2 Parametrización das marcas de ciclo

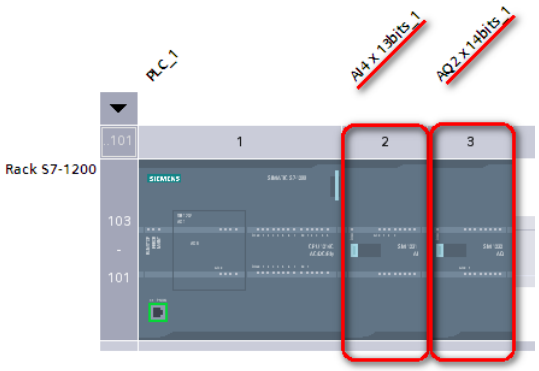
No S7-200 as marcas de ciclo estaban predefinidas e non se podían cambiar (SM0.0 e SM0.1 por exemplo). Nestes PLCs hai que parametrizalas para que poidan ser empregadas. Para facelo, escollemos a mesma opción do apartado anterior e, na ventá propiedades collemos **Marcas de sistema y de ciclo**.

Activamos segundo nos interesen as marcas de sistema (Primeiro ciclo, sempre a 1 entre outras) e/ou as marcas de ciclo coas frecuencias de reloxo que aparecen na pantalla.

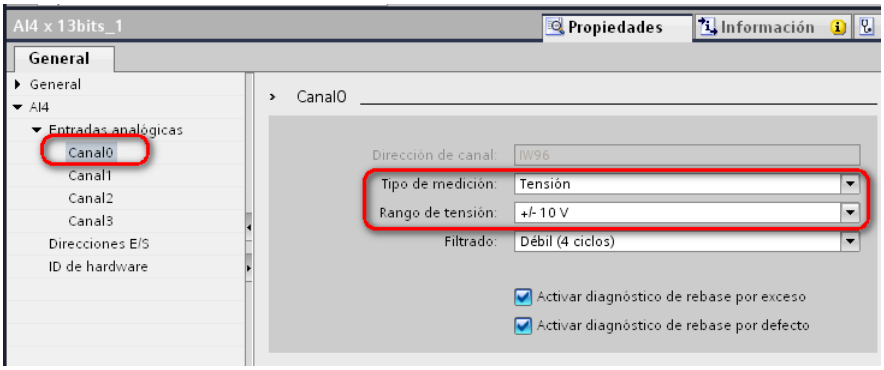


9.1.3 Configuracións de módulos analóxicos

O sistema de parametrización é sempre o mesmo. Supoñamos que dispoñemos dun módulo de 4 entradas analóxicas e outro de 2 saídas analóxicas, colocados a continuación da CPU.



Seleccionamos o módulo correspondente e, tal como fixemos nos apartados anteriores, cambiamos as configuracións.



Hai que ter en conta que, no caso das entradas analóxicas que veñen por defecto na propia CPU non se poden facer cambios na configuración das mesmas.

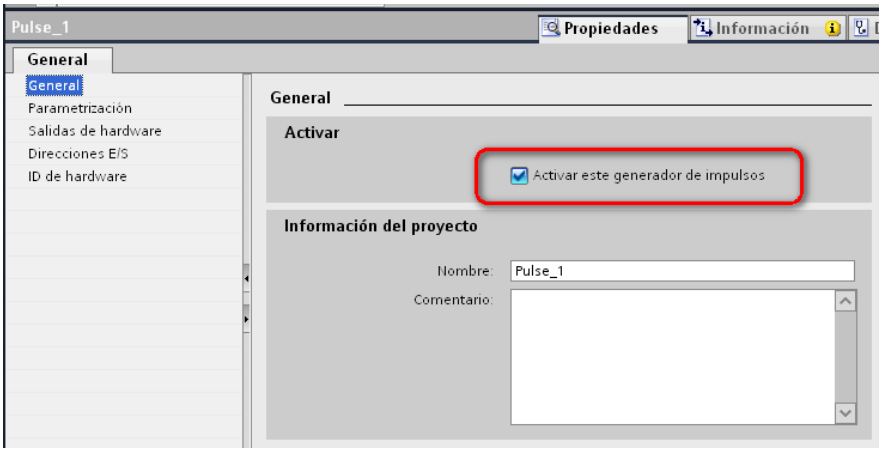
9.1.4 Configuración da saída de impulsos

Do mesmo xeito que nos S7-200, nestes PLCs incorpóranse dúas saídas de impulsos PTO/PWM. Mentres que nos primeiros había que configurar no programa o funcionamento destas dúas saídas, nos S7-1200, a configuración faise do mesmo xeito que fixemos as anteriores.

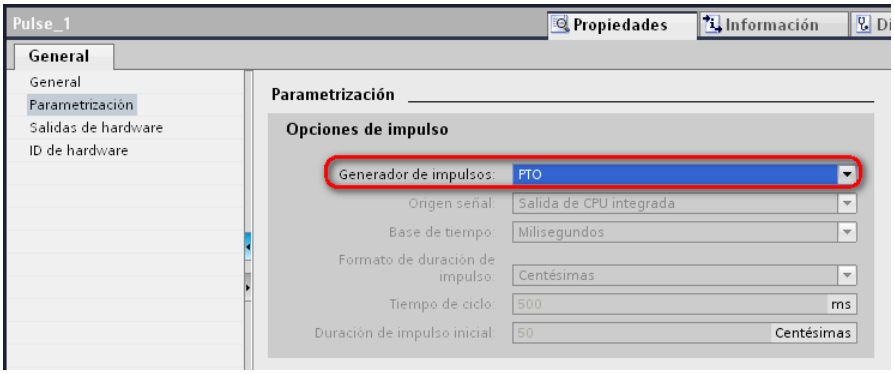
Supoñamos que queremos configurar a primeira saída para que execute un tren de 10 impulsos (PTO). O proceso sería o seguinte:

Na vista xeral de dispositivos escollemos a opción **Pulse_1** (podemos cambiarlle o nome segundo desexemos).

O primeiro é activar este xerador PTO, marcando **Activar este generador de impulsos**.

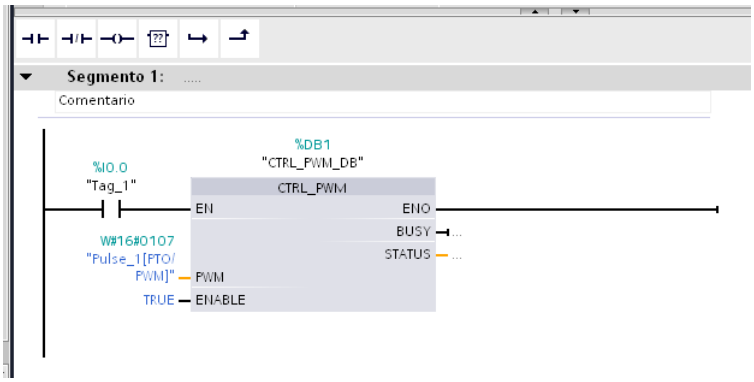


No apartado **Parametrización** escollemos a opción PTO (non me permite cambiar o tempo de ciclo para este formato de saída de impulsos).



Observamos que na seguinte pestana de **Salidas de hardware** non podemos modificar a saída física (Q0.0).

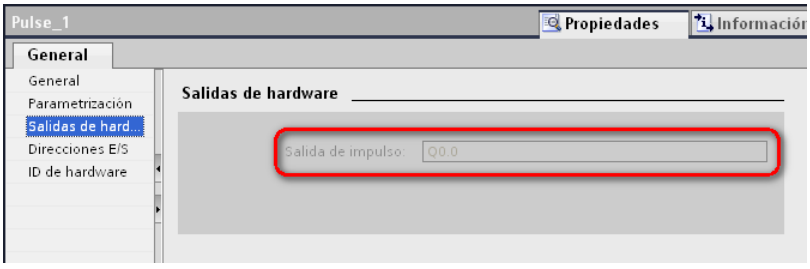
Unha vez configurado, podemos engadir no programa o bloque **CTRL_PWM**



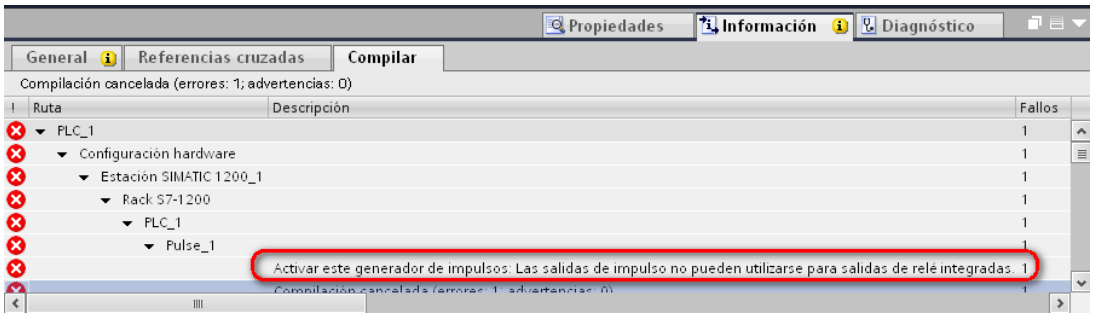
Na entrada PWM conectamos a configuración dun dos dous xeneradores de impulsos (por defecto Pulse_1 e Pulse_2). Activamos a entrada **Enable** se queremos inicialo automaticamente e xa funciona.

A diferencia cos S7-200, que sí o permitían, nestes autómatas non se pode dirixir a saída de impulsos a saídas a relé, polo que, se temos unha CPU AC/DC/RLY non podemos empregar estas funcións, a non ser que engadamos á mesma un módulo de ampliación con saídas dixitais a transistor.

De feito, cando non temos módulos de ampliación para traballar con saídas dixitáis, por defecto para o primeiro xenerador de impulsos asígnase a saída Q0.0.

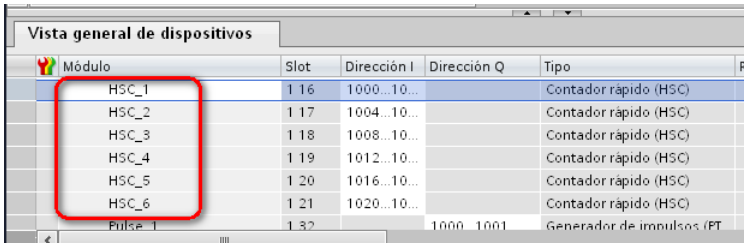


Pero, se facemos esto nunha CPU AC/DC/RLY, cando compilamos, aparece o segunte erro:

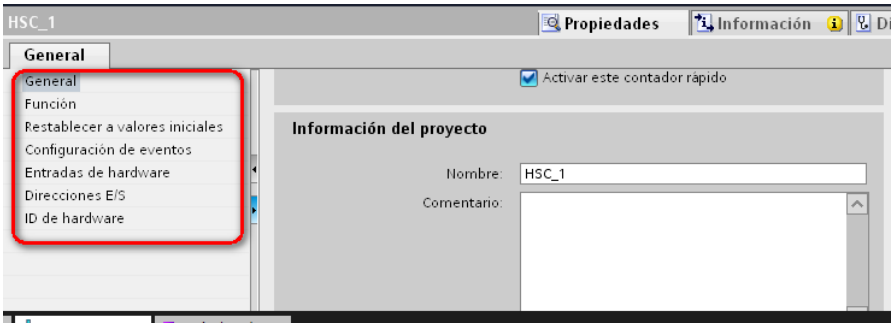


9.1.5 Contadores rápidos

Tamén se configuran na vista xeral de dispositivos como todo o anterior.



Na ventá **Propiedades** do contador, facemos a parametrización.



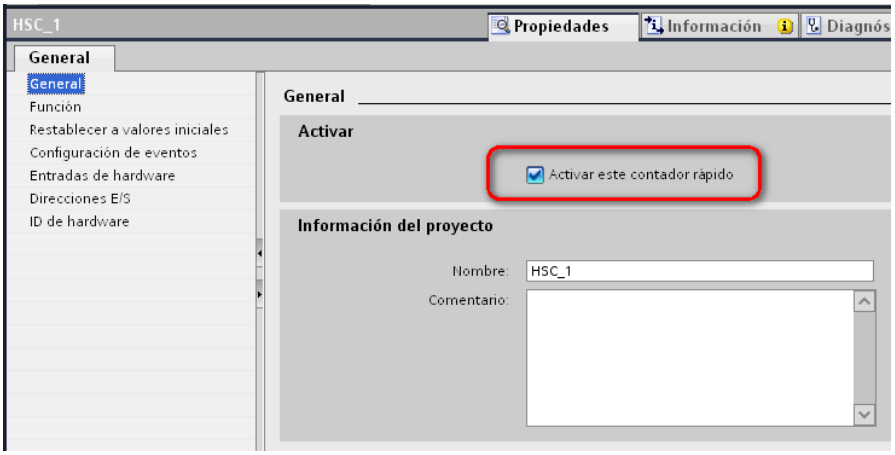
Primeiro activámolo. Despois no apartado **Función** configuramos o modo de traballo (equivalente ao modo nos S7-200), no apartado **Restablecer valores iniciales** programamos o valor inicial e o prefixado para o contador, e por último, configuramos se queremos asociar algún evento a unha interrupción cando cheguemos ao valor prefixado no apartado **Configuración de eventos.**

Exemplo

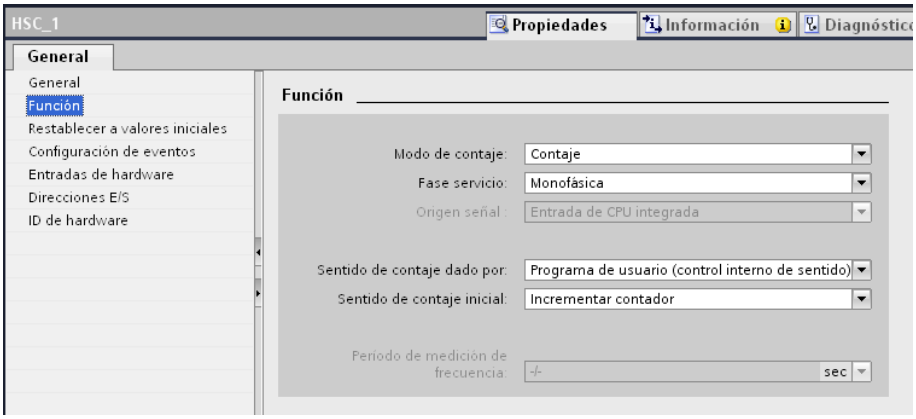
Programa para detectar pulsos nunha entrada dixital empregando un contador rápido, con reset automático ao chegar ao valor prefixado.

Configuramos o HSC1 tal como vemos nas seguintes imaxes.

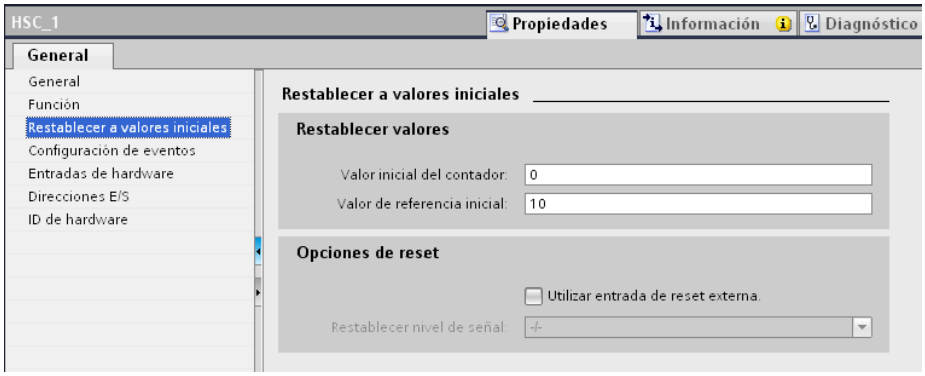
No apartado **General** activamos o contador.



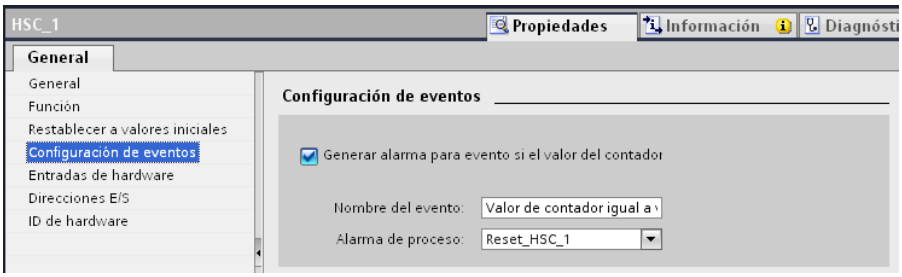
No apartado **Función** escollemos o modo de traballo desexado.



No apartado **Restablecer a valores iniciales** configuramos os valores iniciales actual e prefixado (CV e RV). No exemplo, empezará en 0 e rematará en 10.

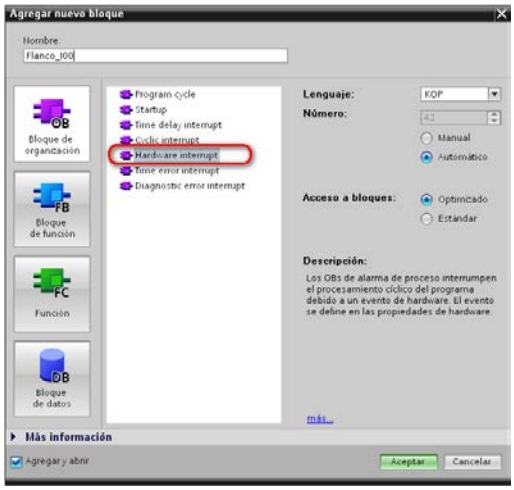


No apartado **Configuración de eventos** configuramos os eventos asociados a contadores rápidos. Neste caso configuramos o evento que queremos que se dispare cando o valor do contador acade o de referencia.

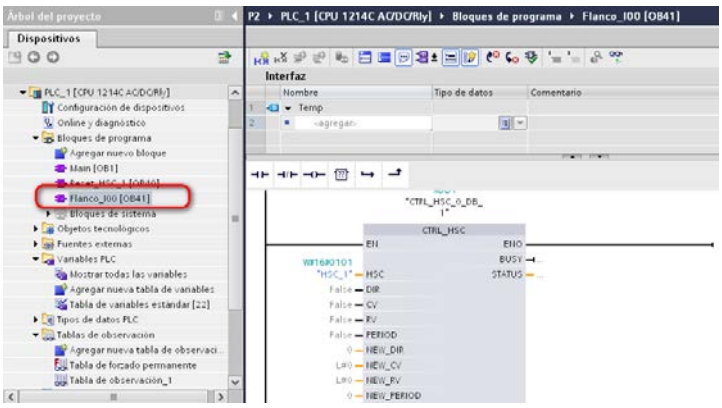


No apartado **Direcciones E/S** configuramos a dirección (en formato enteiro dobre) onde queremos ler o valor do contador. Por defecto propón a dirección %ID1000. Esta será a dirección que teremos que consultar para acceder ao valor do contador en calquera momemto.

A continuación creamos o **OB** que incrementará o contador. Como queremos controlar o flanco da entrada **%I0.0** programaremos un bloque de interrupción de hardware.

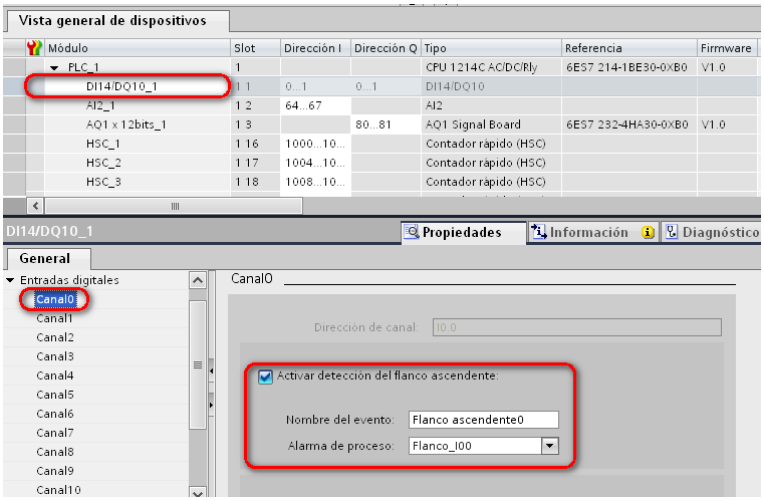


No bloque de programa incluímos a función **CTRL_HSC** facendo referencia ao contador **HSC_1** que configuramos anteriormente.

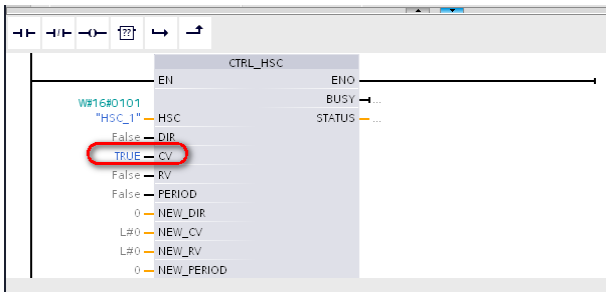


Agora temos que indicarlle que queremos activar a detección de flanco na entrada **%I0.0** e asocialo con este **OB**.

Esto facémolo de novo na configuración de dispositivos.

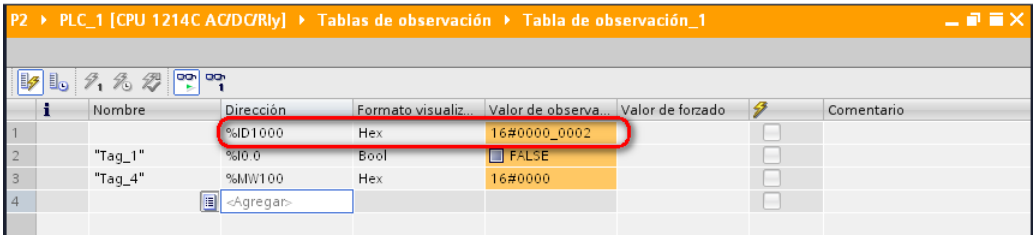


Como tamén queremos resetear o contador cando chegue ao valor prefixado, programaremos outro bloque que asociaremos ao evento correspondente (CV=RV). Nese bloque reseteamos o contador como vemos na figura seguinte.



Este bloque será o que asociemos nas propiedades do contador, no apartado **Configuración de eventos**.

Cargamos o programa, creamos unha táboa de observación e comprobamos na entrada **%ID1000** o valor do contador.



10 Programación de sistemas contínuos. Variables analóxicas

No punto 9.1.3 vimos como configurar as entradas e saídas analóxicas dun S7 1200. Haberá que distinguir entre as entradas incorporadas no propio PLC e os módulos de ampliación, xa que no caso das incorporadas non podemos configurar o rango do sinal de entrada (fixo 0-10V).

Os valores que se aceptan para os sinais de entrada son: ±10V, ±5V, ±2.5V ou 0-20mA.

No caso de entrada en corrente (0-20mA) o valor analóxico convírtese nun dixital comprendido entre 0 e 27648.

No caso das entradas en tensión, o valor analóxico convírtese nun dixital comprendido entre -27648 e +27648.

Os rangos de medida de entradas analóxicas configuradas en tensión son:

Sistema		Rango de medida de tensión						
Decimal	Hexadecimal	±10 V	±5 V	±2,5 V	±1,25V		De 0 a 10 V	
32767	7FFF	11,851 V	5,926 V	2,963 V	1,481 V	Rebase por exceso	11,851 V	Rebase por exceso
32512	7F00							
32511	7EFF	11,759 V	5,879 V	2,940 V	1,470 V	Rango de sobreimpulso	11,759 V	Rango de sobreimpulso
27649	6C01							
27648	6C00	10 V	5 V	2,5 V	1,250 V	Rango nominal	10 V	Rango nominal
20736	5100	7,5 V	3,75 V	1,875 V	0,938 V		7,5 V	
1	1	361,7 µV	180,8 µV	90,4 µV	45,2 µV		361,7 µV	
0	0	0 V	0 V	0 V	0 V		0 V	
-1	FFFF						Los valores negativos no se soportan	
-20736	AF00	-7,5 V	-3,75 V	-1,875 V	-0,938 V			
-27648	9400	-10 V	-5 V	-2,5 V	-1,250 V			
-27649	93FF					Rango de subimpulso		
-32512	8100	-11,759 V	-5,879 V	-2,940 V	-1,470 V			
-32513	80FF					Rebase por defecto		
-32768	8000	-11,851 V	-5,926 V	-2,963 V	-1,481 V			

Os rangos de medida de entradas analóxicas configuradas en corrente son:

Sistema		Rango de medida de intensidad		
Decimal	Hexadecimal	De 0 mA a 20 mA	De 4 mA a 20 mA	
32767	7FFF	23,70 mA	22,96 mA	Rebase por exceso
32512	7F00			
32511	7EFF	23,52 mA	22,81 mA	Rango de sobreimpulso
27649	6C01			
27648	6C00	20 mA	20 mA	Rango nominal
20736	5100	15 mA	16 mA	
1	1	723,4 nA	4 mA + 578,7 nA	
0	0	0 mA	4 mA	
-1	FFFF			Rango de subimpulso
-4864	ED00	-3,52 mA	1,185 mA	
-4865	ECFF			Rebase por defecto
-32768	8000			

Os rangos de medida de saídas analóxicas configuradas en tensión son:

Sistema		Rango de salida de tensión	
Decimal	Hexadecimal	±10 V	
32767	7FFF	V. nota 1	Rebase por exceso
32512	7F00	V. nota 1	
32511	7EFF	11,76 V	Rango de sobreimpulso
27649	6C01		
27648	6C00	10 V	Rango nominal
20736	5100	7,5 V	
1	1	361,7 µ V	
0	0	0 V	
-1	FFFF	-361,7 µ V	
-20736	AF00	-7,5 V	
-27648	9400	-10 V	
-27649	93FF		Rango de subimpulso
-32512	8100	-11,76 V	
-32513	80FF	V. nota 1	Rebase por defecto
-32768	8000	V. nota 1	

[1] En una condición de rebase por exceso o por defecto, la reacción de las salidas analógicas corresponderá a las propiedades ajustadas en la configuración de dispositivos para el módulo de señales analógico. En el parámetro "Reacción a STOP de la CPU", seleccione: "Aplicar valor sustitutivo" o "Mantener último valor".

Os rangos de medida de saídas analóxicas configuradas en corrente son:

Sistema		Rango de salida de intensidad	
Decimal	Hexadecimal	0 mA a 20 mA	
32767	7FFF	V. nota 1	Rebase por exceso
32512	7F00	V. nota 1	
32511	7EFF	23,52 mA	Rango de sobreimpulso
27649	6C01		
27648	6C00	20 mA	Rango nominal
20736	5100	15 mA	
1	1	723,4 nA	
0	0	0 mA	

[1] En una condición de rebase por exceso o por defecto, la reacción de las salidas analógicas corresponderá a las propiedades ajustadas en la configuración de dispositivos para el módulo de señales analógico. En el parámetro "Reacción a STOP de la CPU", seleccione: "Aplicar valor sustitutivo" o "Mantener último valor".

Os valores das entradas e saídas analóxicas gárdanse en datos de tipo palabra (IW e QW respectivamente). As direccións de entrada/saída pódense cambiar neste tipo de autómatas, nas propiedades do hardware.

Para todos os exercicios que se fagan neste documento, teremos en conta que as dúas entradas analóxicas atópanse en IW64 e IW66 e a saída analóxica está en QW80.

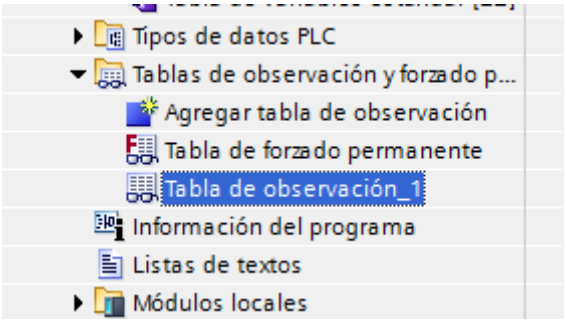
10.1.1.1 Exercicio 17

Visualizar na táboa de observación do TIA Portal o valor dixitalizado da entrada analóxica AI0 (IW64).

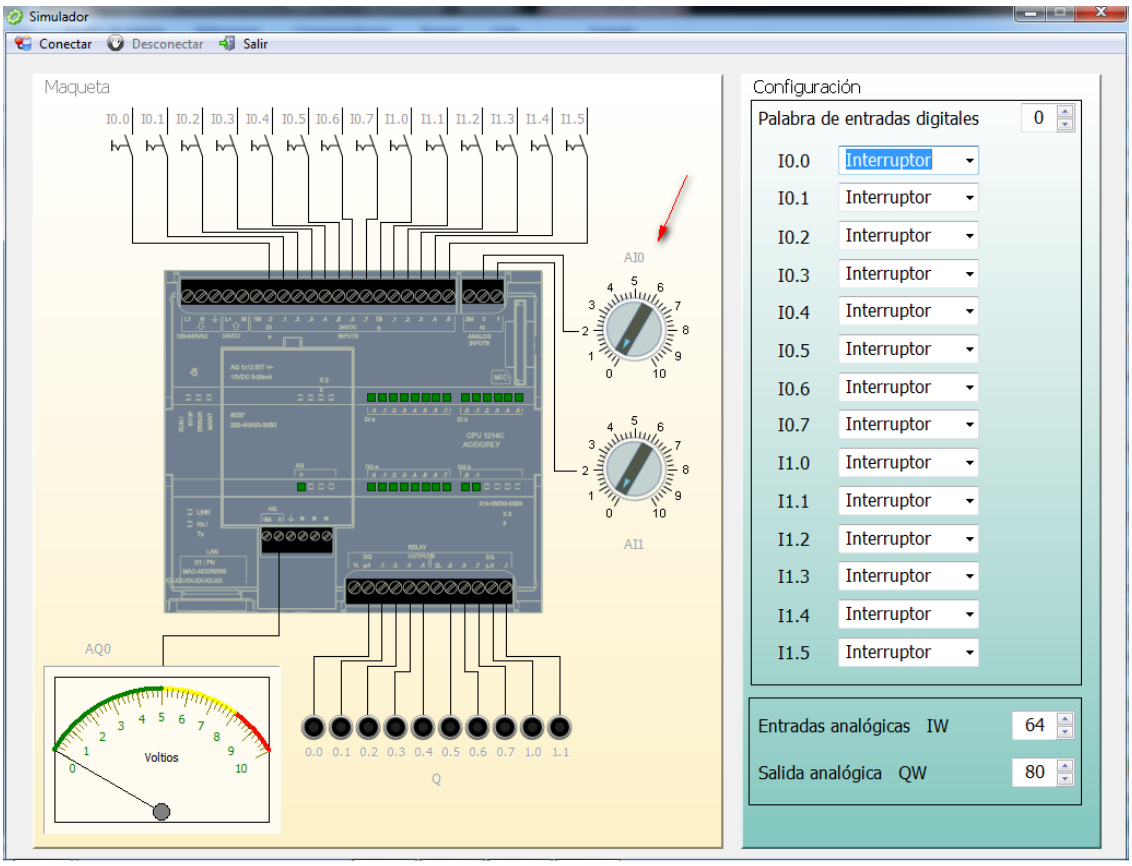
Utilizaremos **virtualmakTCP** para simular a entrada dun sinal 0-10V.

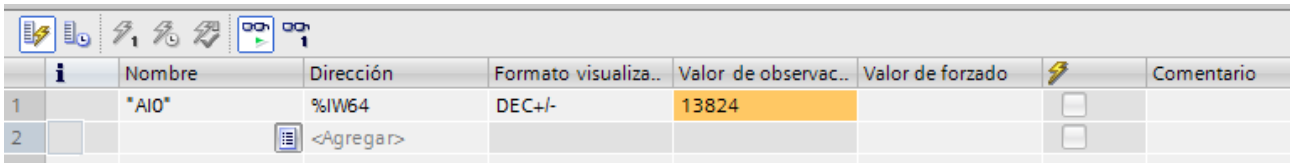
En realidade non temos que programar nada. Simplemente crear un proxecto e ver na táboa de observación o valor de IW64.

Executamos, abrimos a táboa de observación e comprobamos o funcionamento.



Temos que establecer conexión *on-line* e pulsar sobre o icono *Observar todo* para que se visualicen os valores.



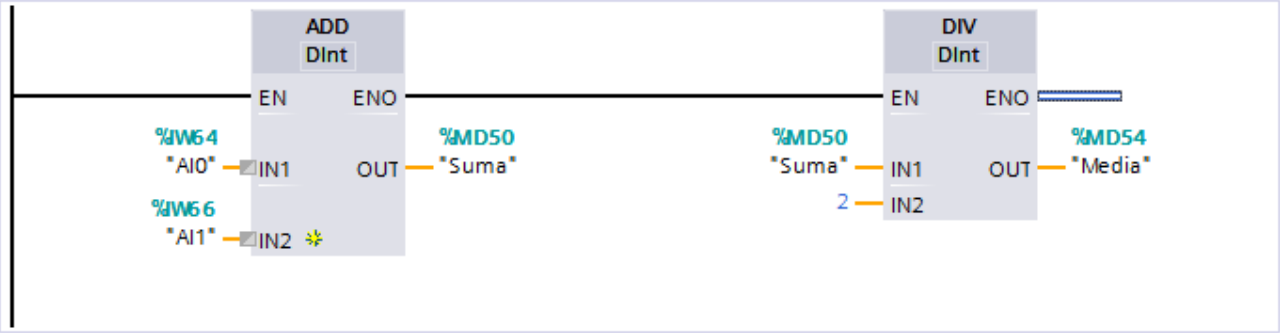


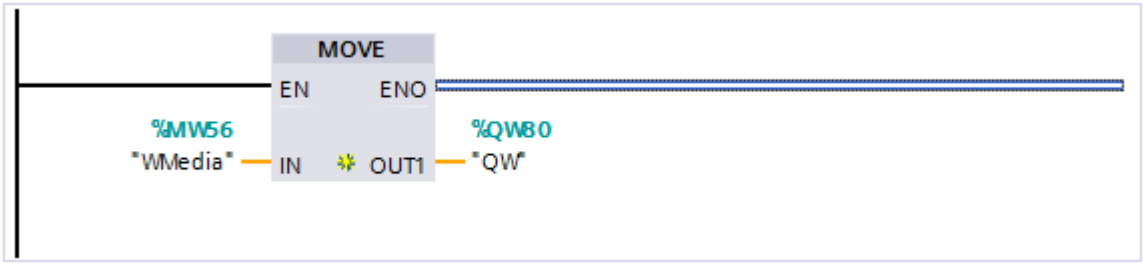
	Nombre	Dirección	Formato visualiza..	Valor de observac..	Valor de forzado	Comentario
1	"AI0"	%IW64	DEC+/-	13824		
2		<Agregar>				

10.1.1.2 Exercicio 18

Facer un programa que lea dúas entradas analóxicas e na saída analóxica nos dea a media aritmética das dúas.

O programa sería:





Neste programa hai que ter en conta algunhas cousas. Os datos das entradas analóxicas caben en 16 bits (palabra). Sabemos que se trata de valores comprendidos entre -27648 e 27648 para entradas bipolares (por exemplo ±10V). Tal como vimos no punto 6.2, os datos de tipo **Int** poden conter cantidades comprendidas entre -32767 e +32768.

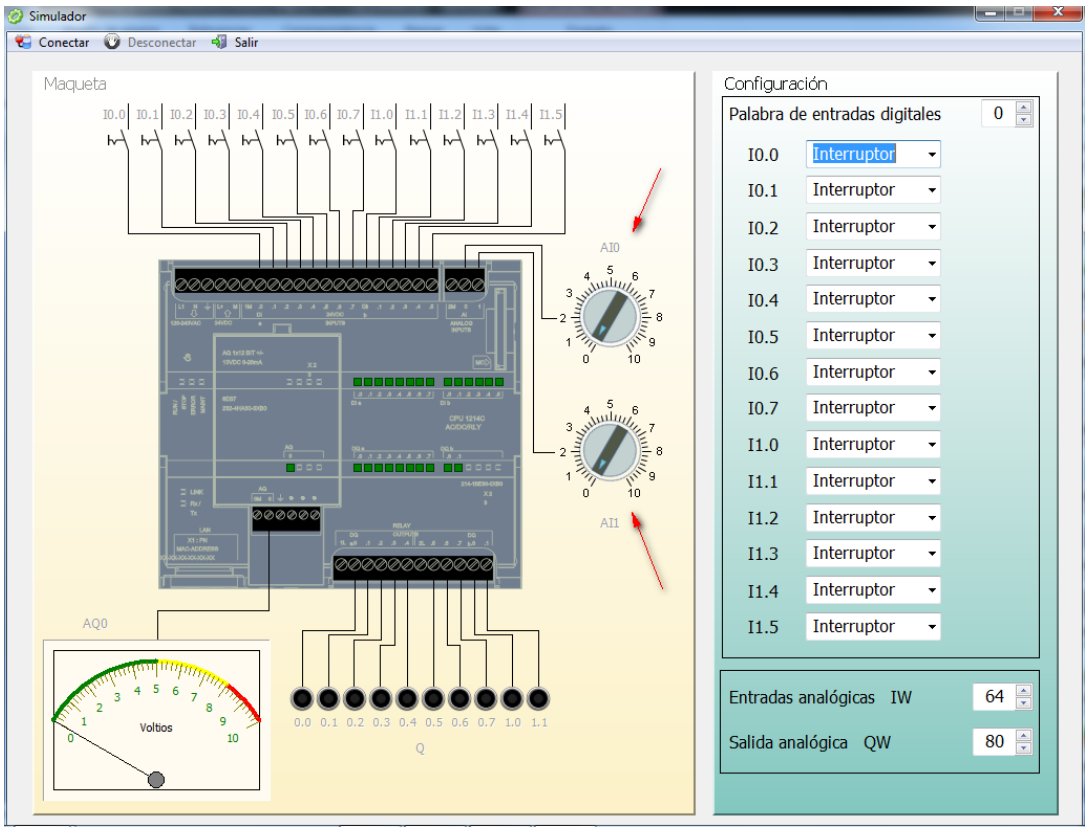
Esto xustifica que non empreguemos coa instrucción **ADD** o tipo de datos **Int** senón o **DInt**, que nos permite almacenar cantidades moito máis grandes (desde -2.147.483.648 ata 2.147.483.647).

O resultado gardámolo nunha posición de memoria na que cabe un **DInt** como pode ser %**MD50**.

Logo dividimos ese resultado entre 2 e gardamos a división noutra posición de memoria, neste caso %**MD54**.

A última cuestión é que para mover o dato á saída non podemos mover un **DInt**, porque nas saídas analóxicas só se aceptan datos en formato palabra. Neste caso movemos a palabra menos significativa de %**MD54** porque estamos seguros que a media de dous valores que non sobrepasan 27648 non vai ser superior a este valor. Movemos polo tanto %**MW56**.

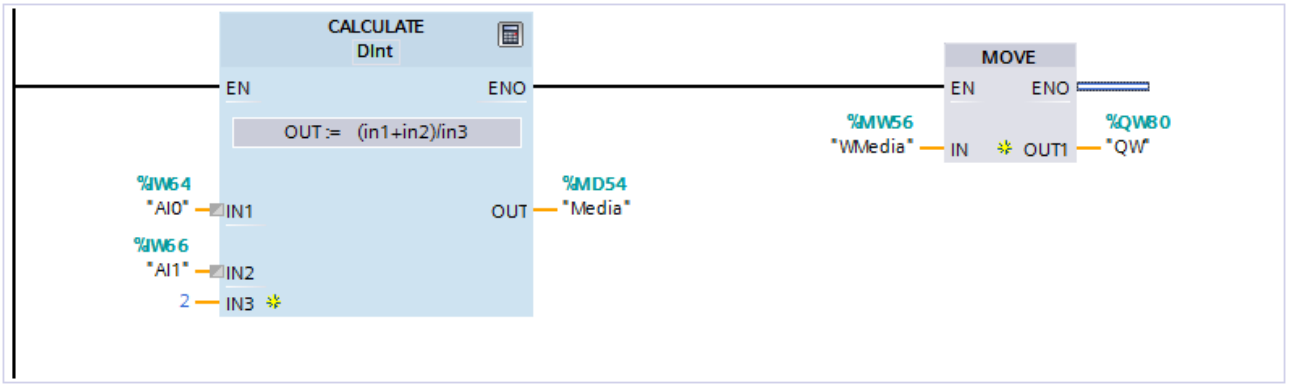
Comprobamos o funcionamento en **virtualmakTCP**.



10.1.1.3 Exercicio 19

Repetir o exercicio anterior empregando a instrucción **CALCULATE**.

Cambiariamos dous segmentos anteriores polo seguinte:



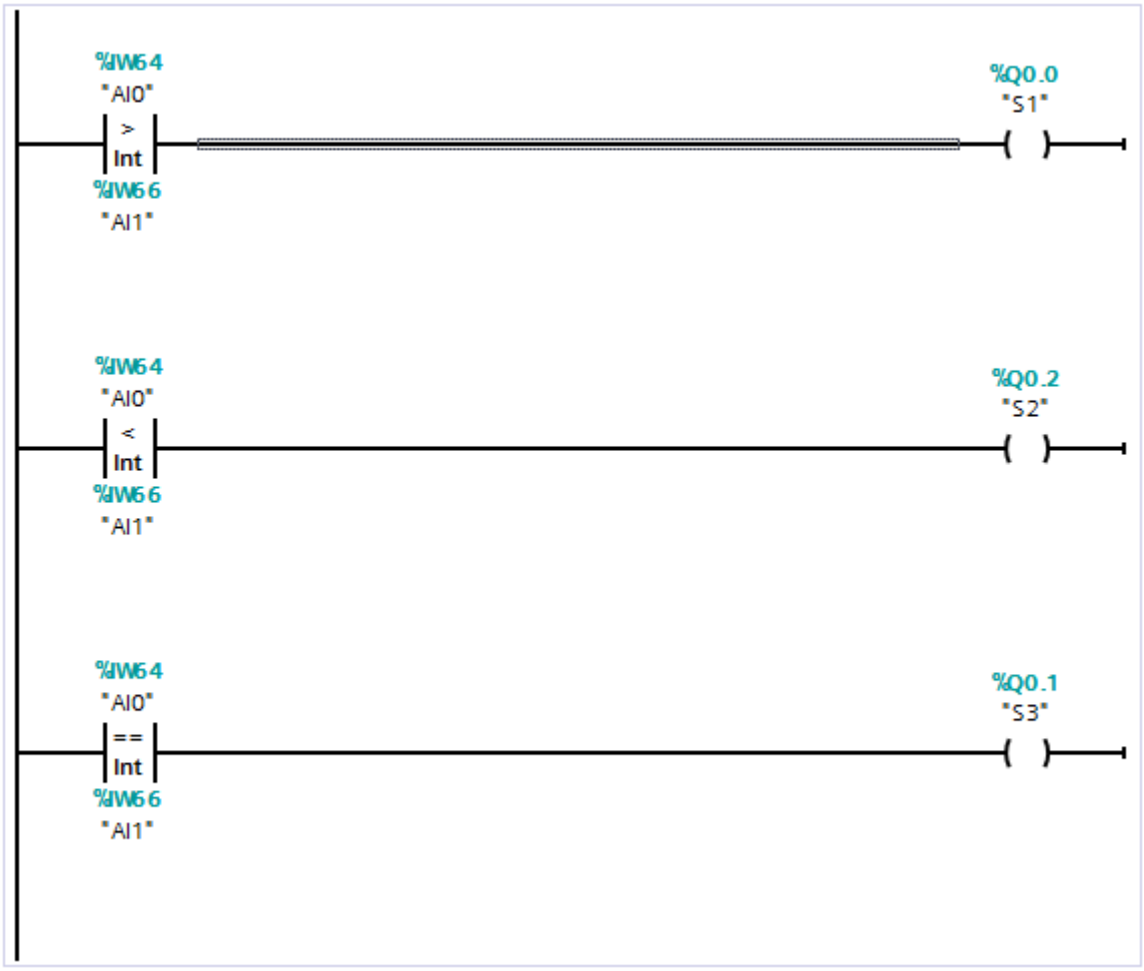
10.1.1.4 Exercicio 20

Facer un programa que compare dous datos procedentes de dúas entradas analóxicas, de xeito que, se o primeiro é maior que o segundo, actívase unha saída. Se o segundo é maior que o primeiro, actívase outra, e se os dous datos son iguais actívase unha terceira saída.

Tomaremos de novo %**IW64** e %**IW66**.

Condición	Saída activada
%**IW64** > %**IW66**	Q0.0
%**IW64** = %**IW66**	Q0.1
%**IW64** < %**IW66**	Q0.2

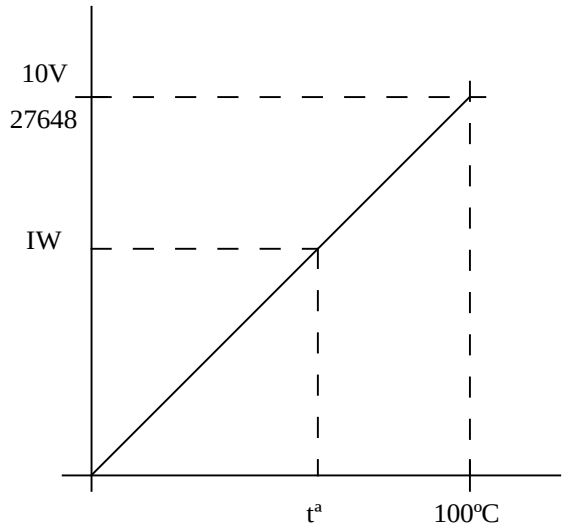
Despois de colocar os dous segmentos necesario para que funcione **virtualmakTCP** temos as comparacións:



10.2 Escalado e desescalado de magnitudes analóxicas

En aplicacións que manexan magnitudes analóxicas é habitual ter que escalar e desescalar as mesmas.

Por exemplo, supoñamos un transductor que está a detectar unha temperatura, e que unha vez conectado a unha entrada analóxica do autómata temos un sinal 0-10V para un rango de temperaturas de 0 a 100ºC.



En autómatas máis antiguos como os S7-200 para obter a relación entre a temperatura e o valor que estamos a ler na entrada analóxica había que prantexar a ecuación da recta correspondente. No exemplo sería:

$$t^a = \frac{100 \cdot IW}{27648}$$

Nos autómatas S7-1200 contamos coas funcións **NORM_X** e **SCALE_X** (ver 7.11.4), que nos permiten facer o mesmo.

10.2.1.1 Exercicio 21

Visualizar na táboa de observación o valor dunha tempertaura que está sendo tomada por un transductor conectado a un acondicionador de sinal que entrega 0-10V para unha temperatura de 0-100ºC.

O programa sería:

Empregando **virtualmakTCP** comprobamos como funcionan as instruccións anteriores.

	Nombre	Dirección	Formato visualización	Valor de observac..
1	"AI0"	%IW64	DEC+/-	13824
2	"Temperatura"	%MD24	Número en coma flotan...	50.0
3		<Agregar>		

A primeira instrucción (NORM_X) convirte o valor da entrada %IW64 a un valor normalizado entre 0 e 1.

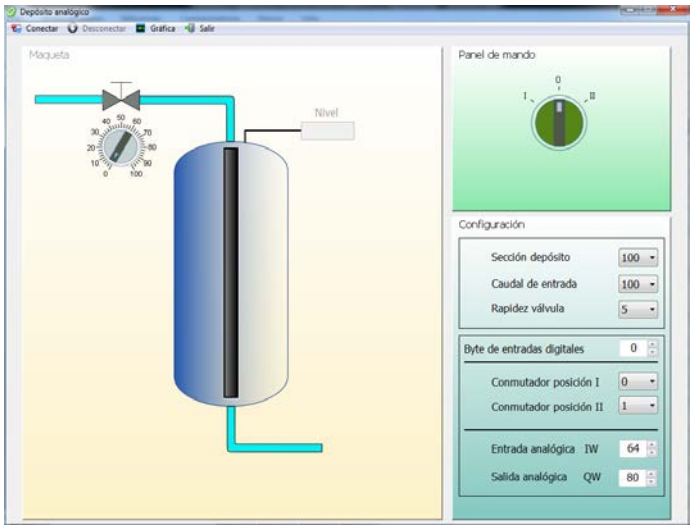
Ese valor pásaselle á segunda instrucción (SCALE_X) que o converte nun valor escalado entre 0 e 100.

10.2.1.2 Exercicio 22

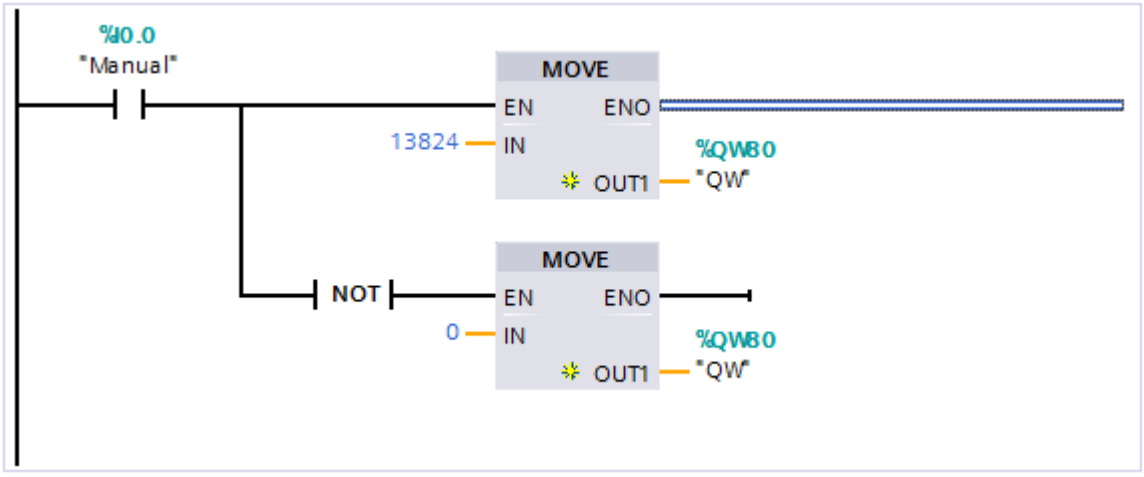
Dispoñemos dun depósito que ten un orificio de saída polo que está a saír auga continuamente. A entrada de auga ao mesmo regúlase mediante unha válvula proporcional que abre ao 100% cando recibe 10V e pecha cando recibe 0V.

Faremos unha aplicación na que abriremos a válvula nunha porcentaxe e comprobaremos ata onde sube o nivel.

Empregamos **virtualmakTCP** para simular o proceso.



Faremos simplemente que, cando o conmutador estea na posición I se abra a válvula á metade, e cando retorne á posición incial, pecharase a válvula. O programa sería:

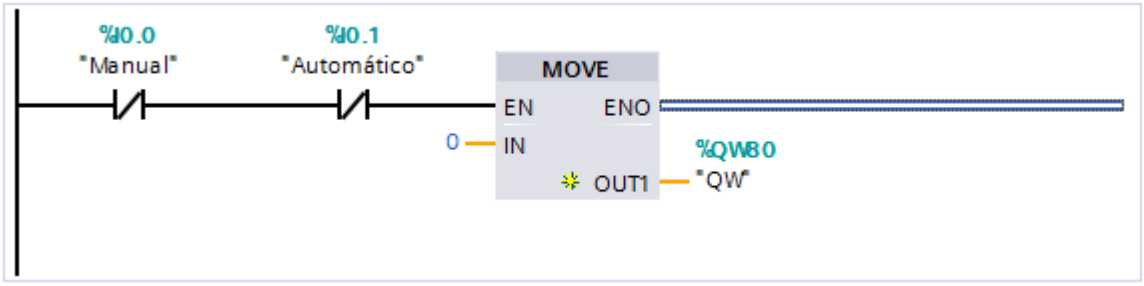


10.2.1.3 Exercicio 23

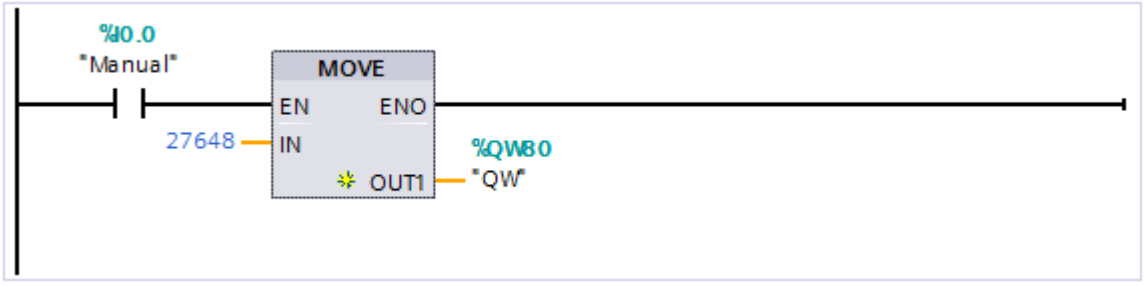
No mesmo depósito do exercicio anterior faremos unha regulación Todo-Nada. Se o conmutador está na posición I, a válvula abrirá ata o 100% e se está na posición II abrirá ao 100% ata chegar a un valor que prefixaremos (consigna), a partir do cal pecharase.

O programa sería:

En reposo movemos un cero á saída analóxica.

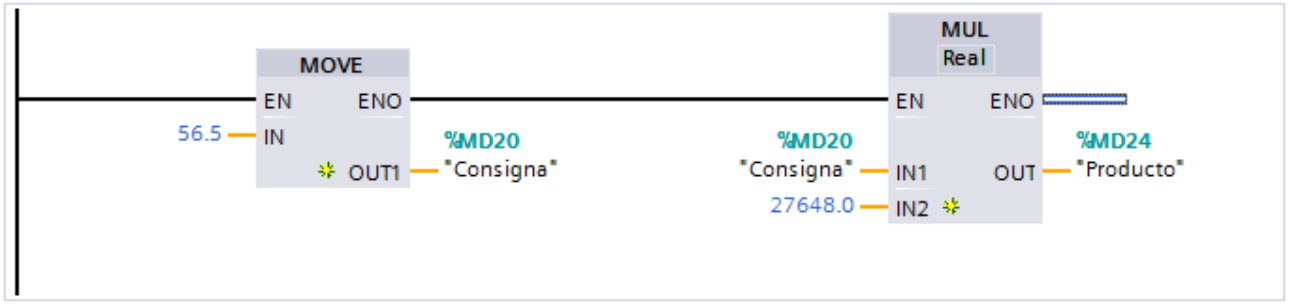


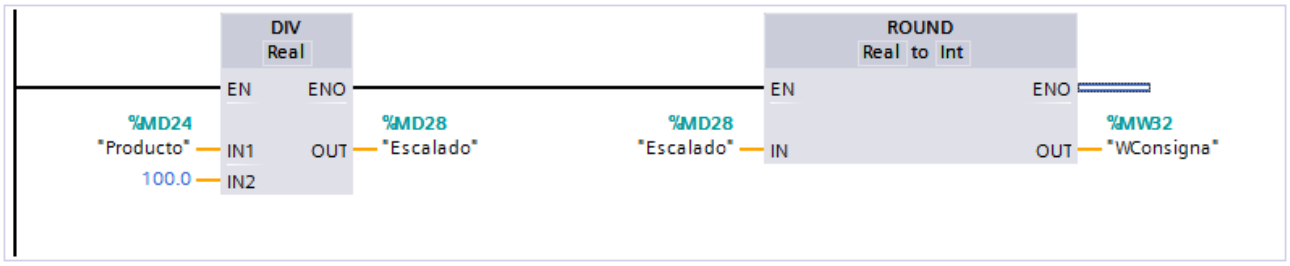
En modo manual abrimos a válvula ao 100%.



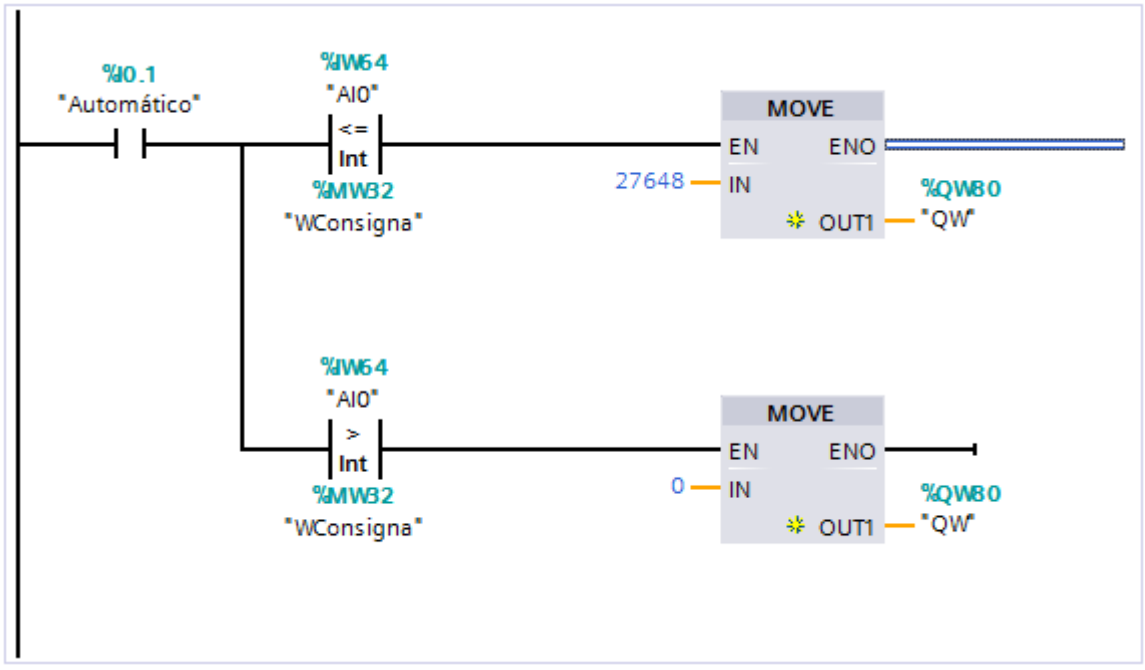
En modo automático, e como queremos introducir a temperatura co seu valor real, facemos a ecuación vista no punto 10.2

$$WConsigna = \frac{t^{a}.27648}{100}$$

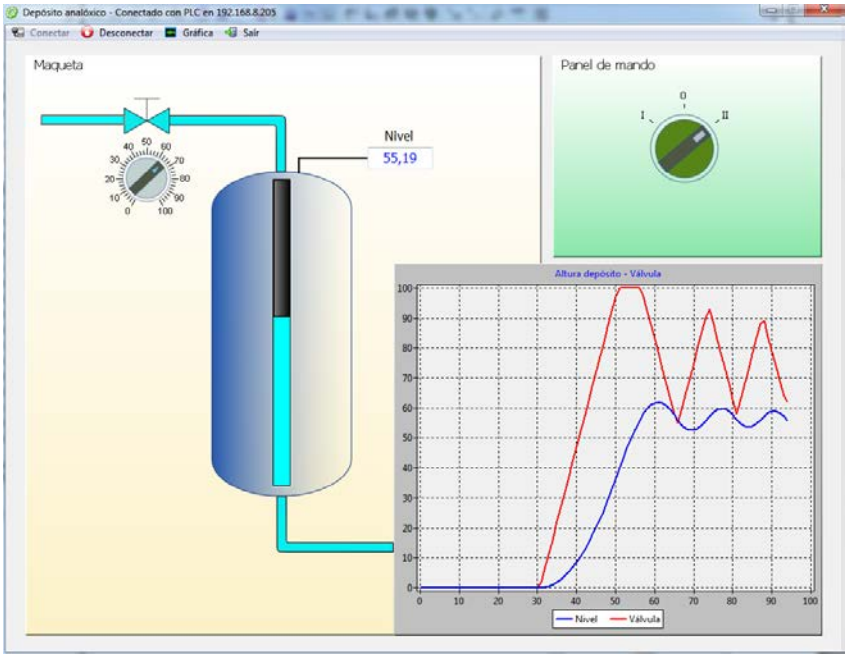




Comparamos o valor de %MW32 (Temperatura convertida a formato palabra), co valor do nivel do depósito.



Empregamos **virtualmakTCP** para visualizar o proceso.



10.2.1.4 Exercicio 24

Modificar o exercicio anterior empregando as instruccións vistas en 7.11.4 NORM_X e SCALE_X

O programa sería igual que o anterior, subtituíndo os dous segmentos do escalado polo seguinte.

11 Programación estructurada

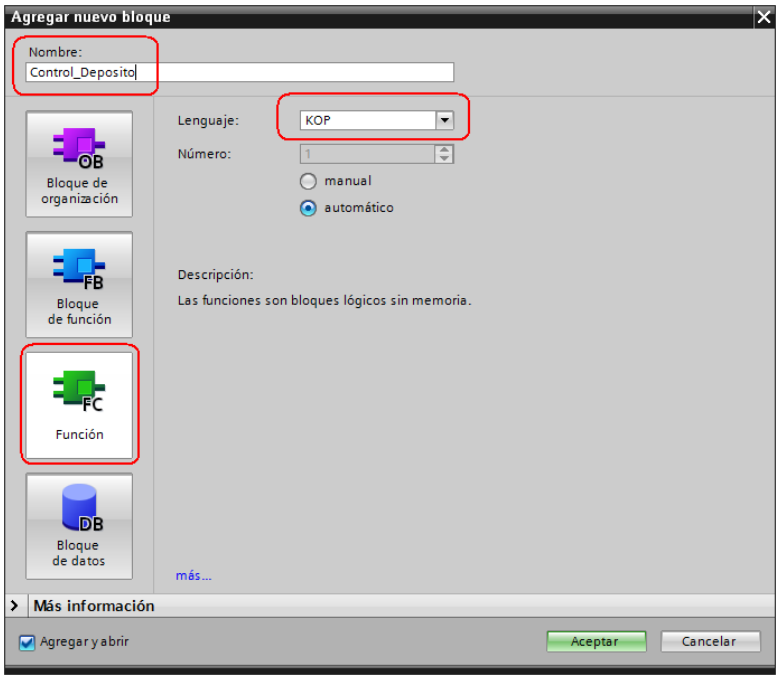
Como se indicou no punto 7.2, para facer programación estructurada nos S7-1200 dispoñemos de OBs, FBs e FCs, que nos permiten darlle modularidade aos programas actuais.

Para ver como é o método de traballo a seguir en caso de querer facer programación modular, imos repetir o exercicio anterior, pero implementando o control da válvula de saída desde unha funcion (FC).

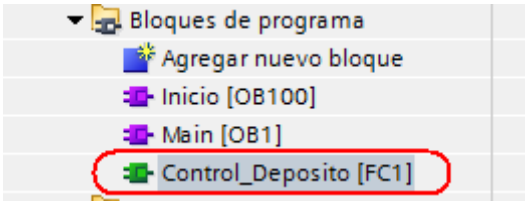
11.1.1.1 Exercicio 25

Empezamos programando a función FC.

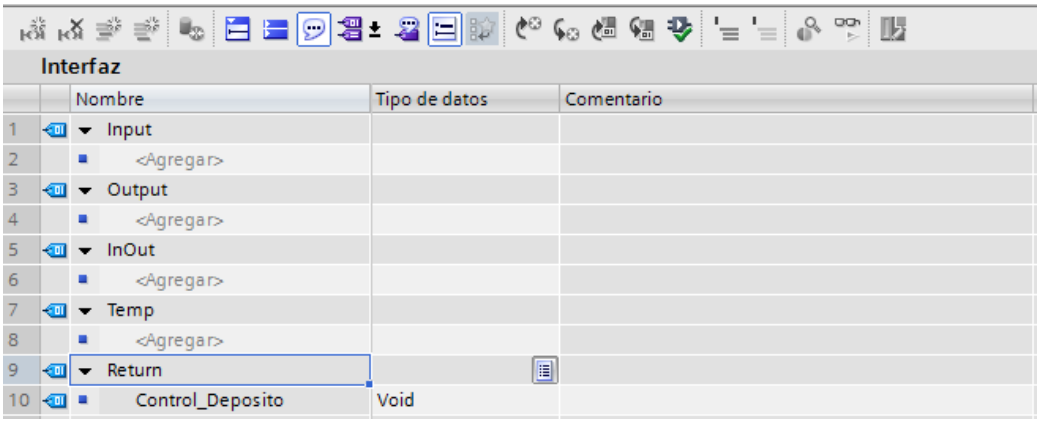
Pulsamos sobre **Agregar nuevo bloque** e creamos a FC, seleccionando a linguaxe **KOP**.



Aparece a nova FC na árbore do proxecto.



Cando programamos funcións é importante ter en conta a zona de variables que aparece na parte superior.

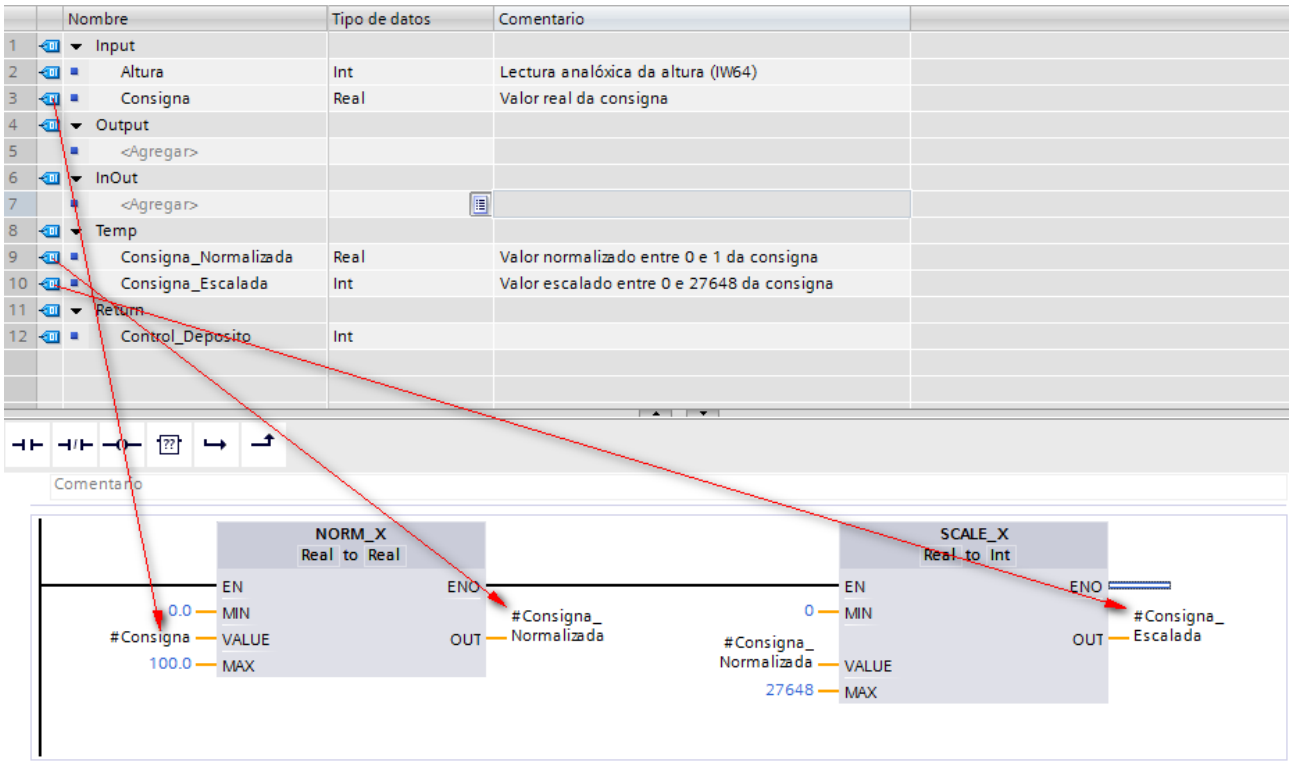


Aquí podemos declarar variables de entrada (Input), variables de saída (Output), variables bidireccionais, de entrada/saída (InOut), variables temporais, que se utilizan para cálculos dentro da propia función (Temp) e, por último, o valor de retorno da función (Return).

Declararemos as seguintes variables:

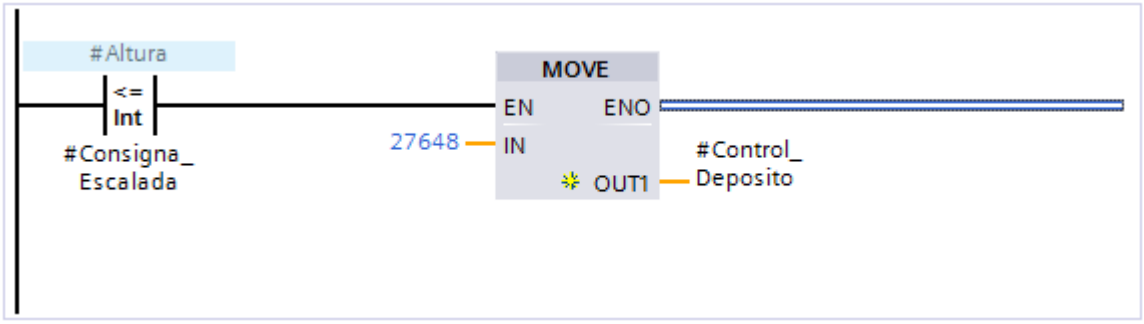
	Nombre	Tipo de datos	Comentario
1	Input		
2	Altura	Int	Lectura analóxica da altura (IW64)
3	Consigna	Real	Valor real da consigna
4	<Agregar>		
5	Output		
6	<Agregar>		
7	InOut		
8	<Agregar>		
9	Temp		
10	Consigna_Normalizada	Real	Valor normalizado entre 0 e 1 da consigna
11	Consigna_Escalada	Int	Valor escalado entre 0 e 27648 da consigna
12	<Agregar>		
13	Return		
14	Control_Deposito	Int	

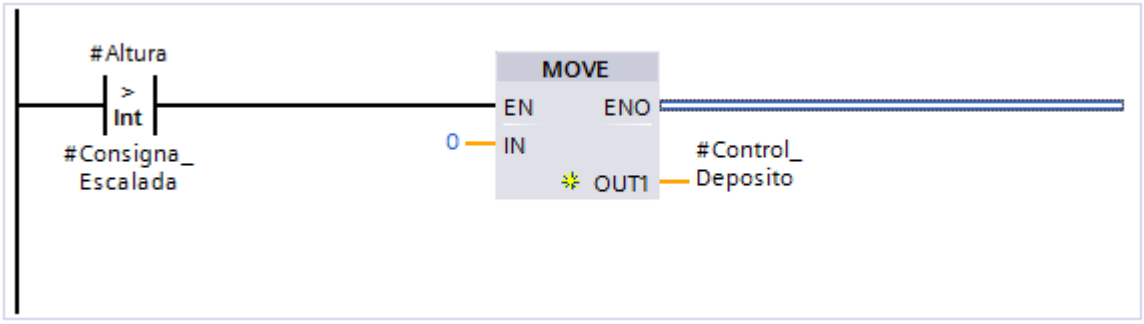
Agora escribimos o programa que vai dentro da función. Fixémonos que podemos arrastrar unha variable desde a táboa onde as temos definidas ata o programa.



As variables definidas amósanse no programa precedidas do símbolo #.

Escribimos os seguintes segmentos, seguindo o mesmo criterio.

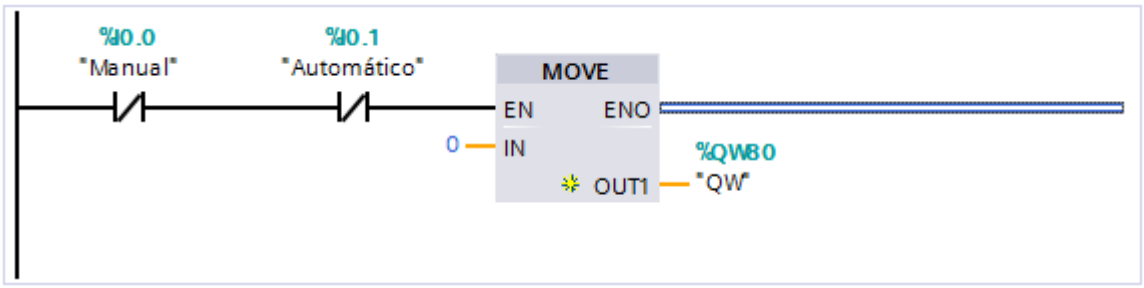


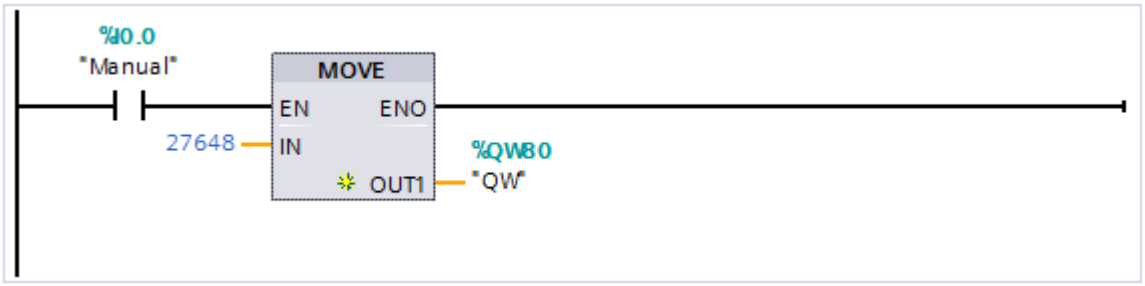


Fixémonos como movemos un 0 ou 27648 á saída da función, dependendo dos valores de comparación.

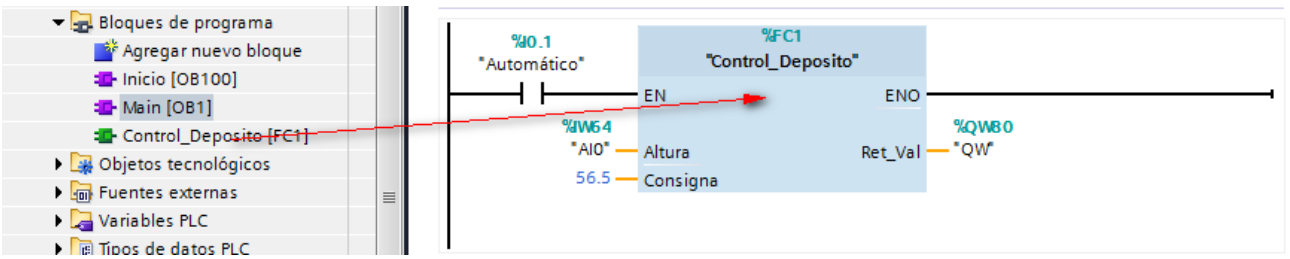
Agora, no programa principal (OB1), facemos a chamada á función.

Primerio programamos os modos desactivado e manual.

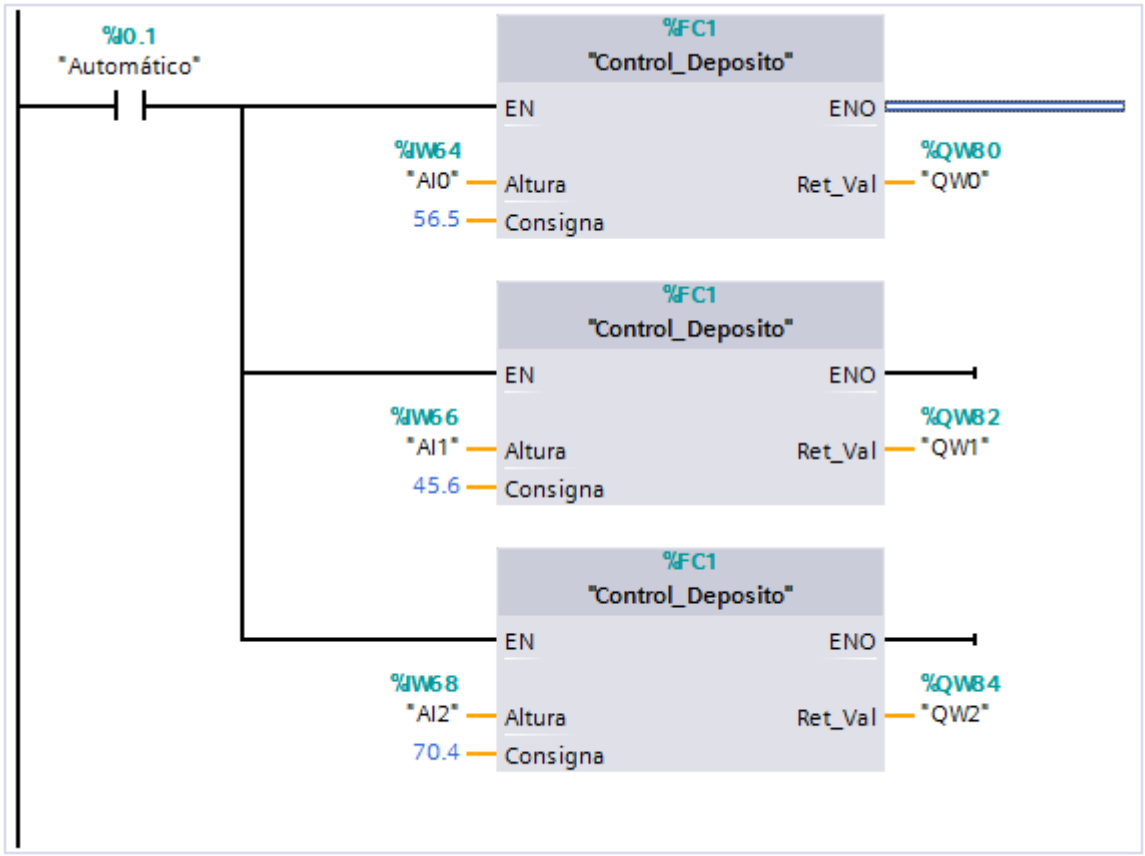




E por último á chamada á función, que a podemos facer arrastrando desde a árbore do proxecto o nome.



Este tipo de programación é útil cando temos varios elementos similares. Por exemplo, se tiveramos tres depósitos aos que lle quixeramos facer unha regulación similar pero con distintas consignas, só programariamos unha vez a función FC e chamariámola tres veces, unha con cada consigna, como vemos na figura seguinte.



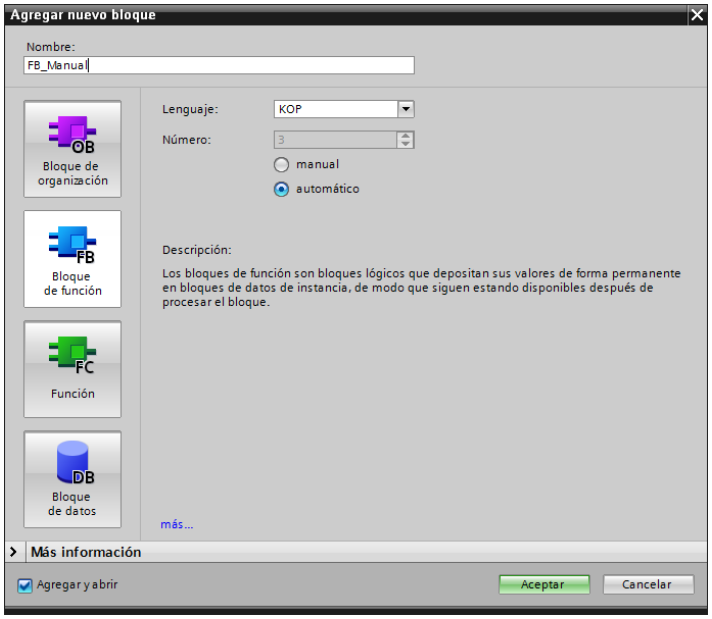
11.1.1.2 Exercicio 26

Neste caso imos implementar unha regulación similar á do exercicio anterior, empregando bloques de función (FBs).

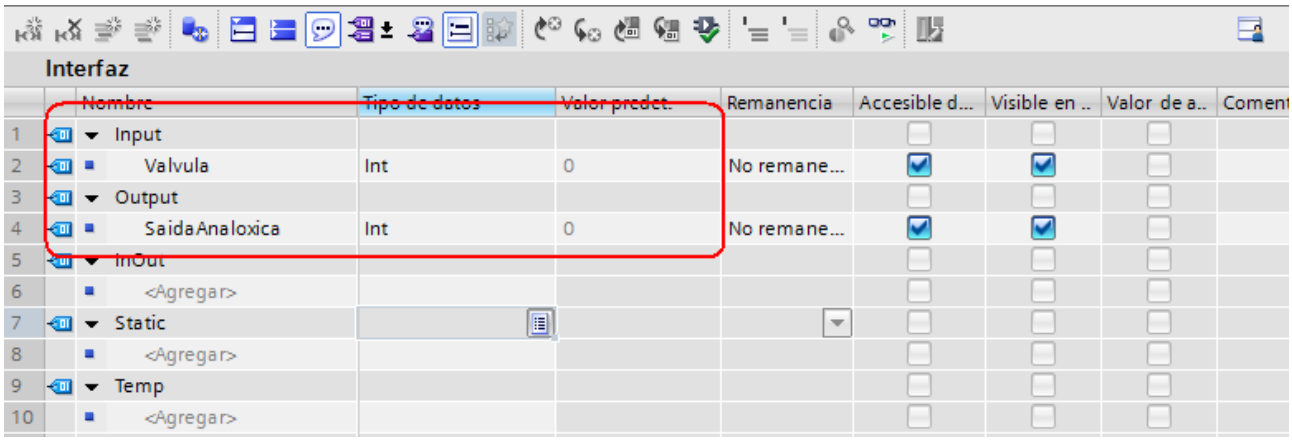
O procedemento é similar ao visto no exercicio anterior, con algún matiz. Por exemplo, agora dispoñemos de variables de tipo **Static** que conservan o valor cando se sae do bloque de función. Por outra banda, non temos a opción **Return** que vimos nas FCs, pero podemos colocar o valor de retorno nas variables de tipo **Output**.

No exercicio crearemos unha FB para o modo Manual e outra para o modo Automático.

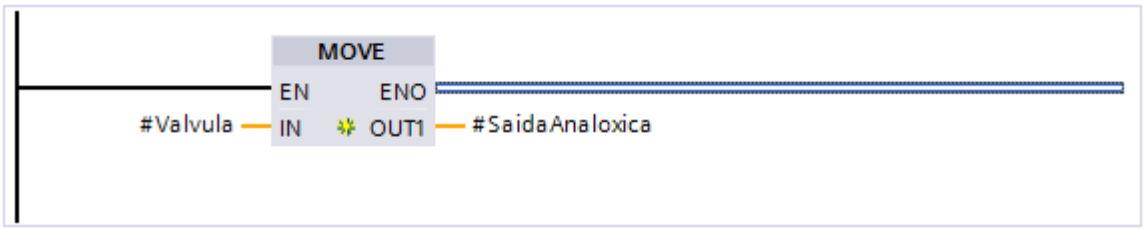
Empezamos creando a FB_Manual en **Agregar nuevo bloque**.



Declaramos as variables que imos empregar. Neste caso só moveremos á saída o valor que lle pasemos na chamada á función, polo que nos chega unha variable de entrada e unha de saída (#Valvula e #SaidaAnaloxica respectivamente.

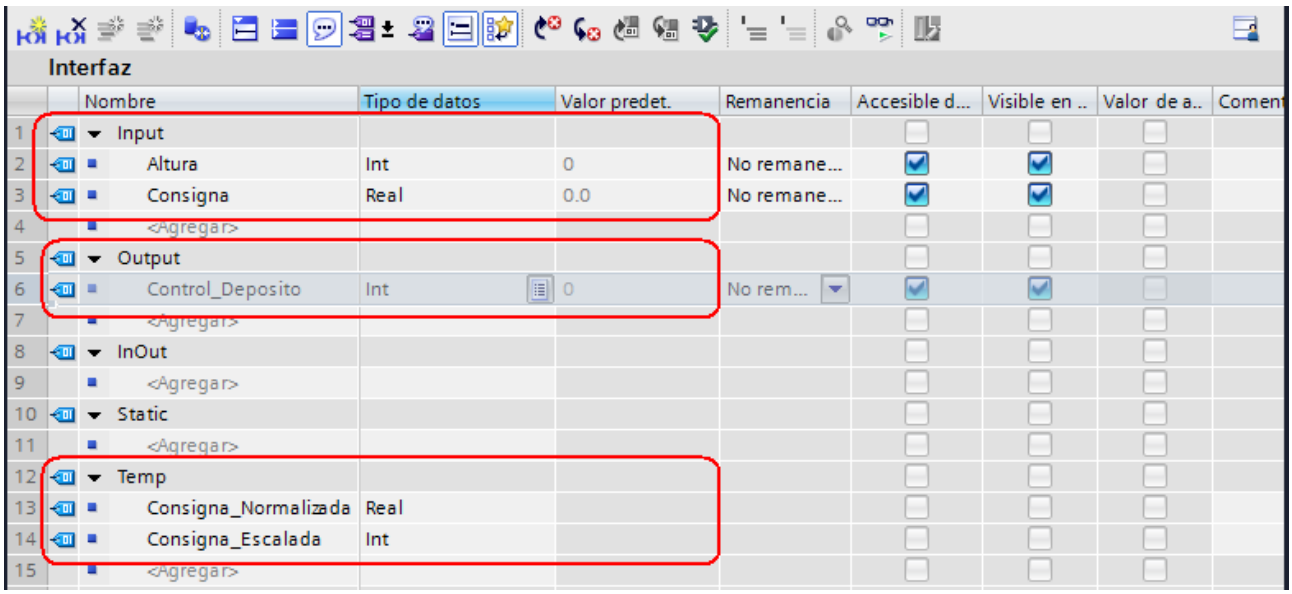


A continuación programamos o único segmento de que consta o FB.

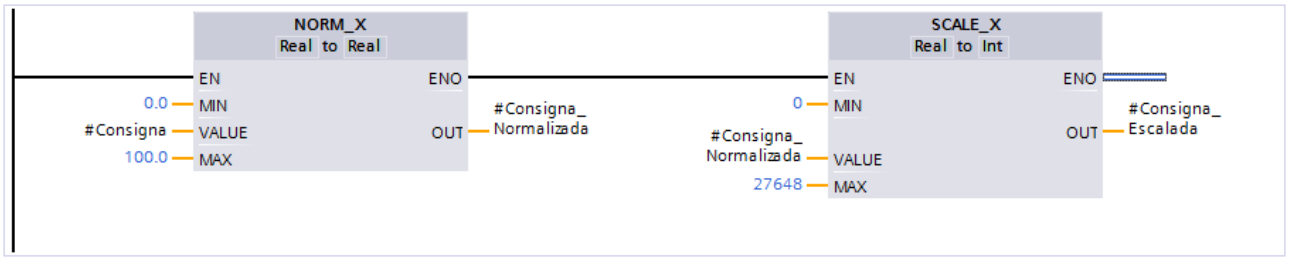


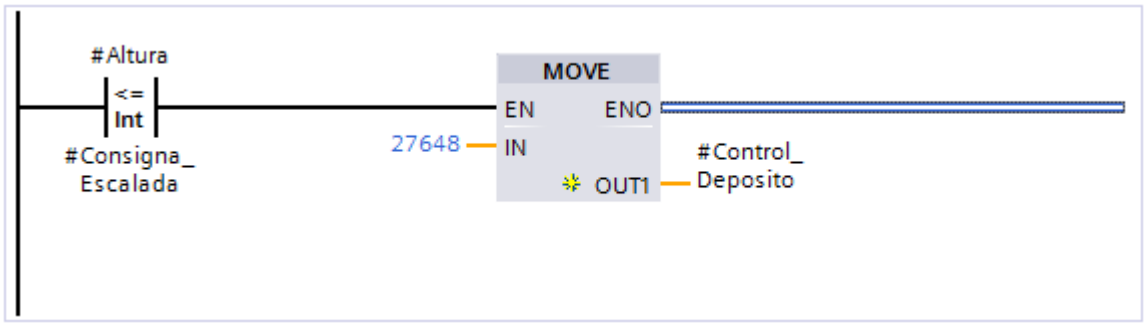
Facemos o mesmo coa FB_Automatico.

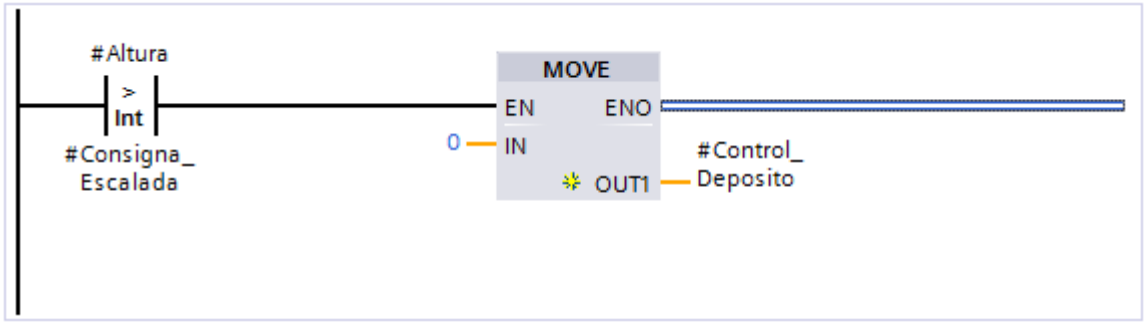
Agora temos as variables seguintes:



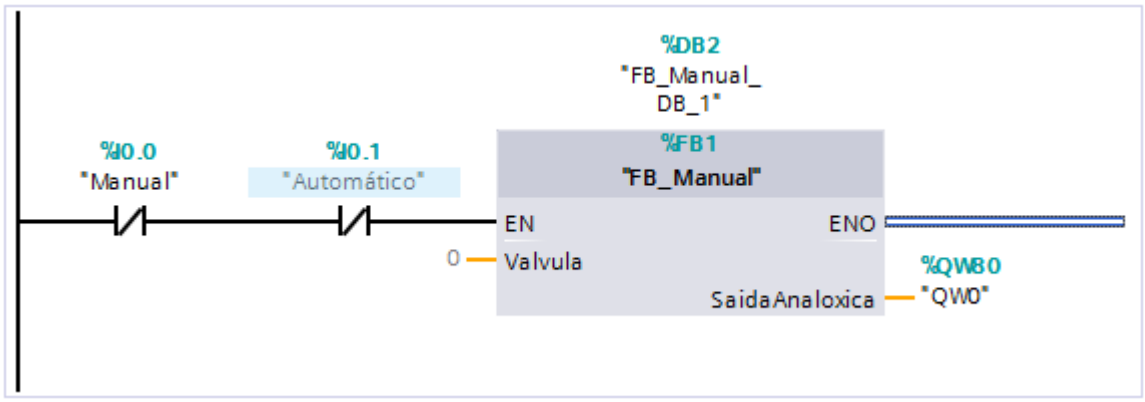
Programamos os segmentos igual que no exercicio anterior.

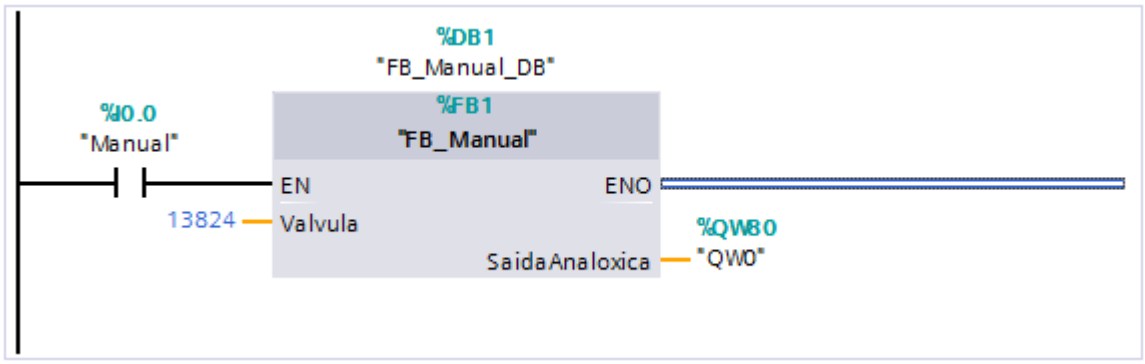


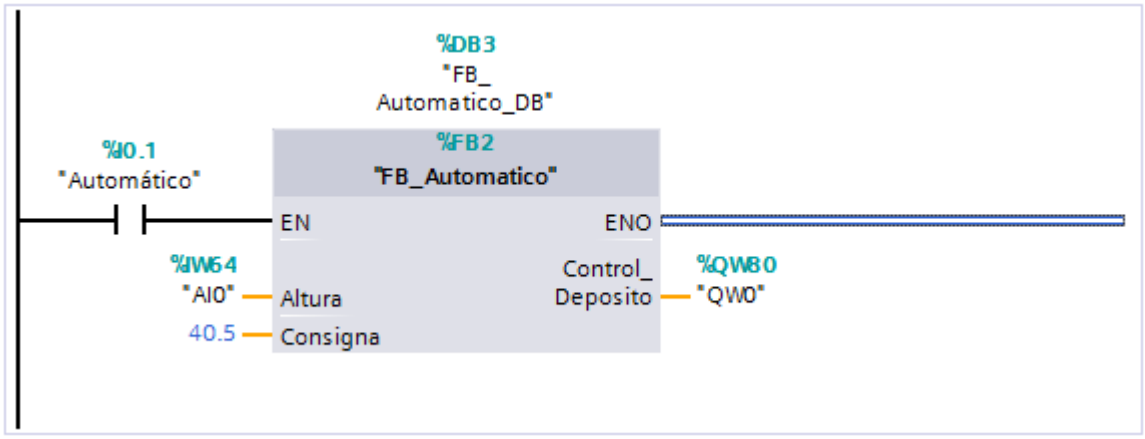




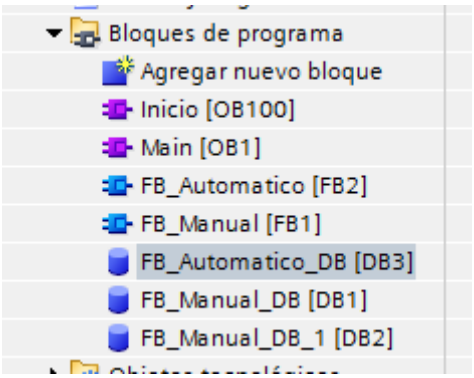
No programa principal (OB1) facemos as chamadas aos bloques de función cando o necesitemos. Neste caso aprovéitase a chamada á FB_Manual para pechar a válvula cando estemos en OFF.







Fixémonos que, cada vez que incorporamos unha chamada a unha FB, créase un bloque de datos (DB) diferente, onde podemos gardar os datos de cada chamada por separado.



Observamos que hai un DB3 que se corresponde coa chamada á FB_Automatico e dous DBs asociados a cada unha das chamadas á FB_Manual.

12 Introducción á programación SCL

Xa se mencionou no punto 6.1.1 a existencia da linguaxe de programación SCL (*Structured Control Language)*, como parte das linguaxes que pertencen a norma IEC-61131-3.

Aínda que os grandes fabricantes son reacios a abandonar as súas maneiras de facer programación de PLCs, cada vez máis van engadindo opcións que se acercan ao cumprimento de dita norma.

S7-SCL (Structured Control Language) é unha linguaxe de programación de alto nivel baseada en PASCAL e que permite facer unha programación estructurada.

A continuación repasamos as instruccións básicas desta linguaxe de programación.

12.1 Instrucción IF-THEN

A instrucción IF-THEN é unha instrucción condicional que controla o fluxo do programa executando unha serie de instruccións baseándose na avaliación dun valor lóxico dunha expresión. Tamén é posible utilizar paréntesis para anidar ou estructurar a execución de instrucciós IF-THEN múltiples.

SCL	Descripción

`IF "condición" THEN`
`  instrucción_A;`
`  instrucción_B;`
`  instrucción_C;`
`  ;`
	Si "condición" es TRUE o 1, entonces ejecuta las siguientes instrucciones hasta que aparezca la instrucción END_IF.
	Si "condición" es FALSE o 0, salta a la instrucción END_IF (a no ser que el programa incluya instrucciones ELSIF o ELSE adicionales).

`[ELSIF "condición-n" THEN`
`  instrucción_N;`
`  ;]`
	La condición ELSEIF[1] opcional aporta condiciones adicionales que deben evaluarse. Ejemplo: Si "condición" de la instrucción IF-THEN es FALSE, entonces el programa evalúa "condición-n". Si "condición-n" es TRUE, ejecuta "instrucción_N".

`[ELSE`
`  instrucción_X;`
`  ;]`
	La instrucción ELSE opcional aporta instrucciones que deben ejecutarse si la "condición" de la instrucción IF-THEN es FALSE.

`END_IF;`	La instrucción END_IF finaliza la instrucción IF-THEN.

12.2 Instrucción CASE

Esta instrucción executa unha serie de instruccións en función do valor dunha variable.

SCL	Descripción
CASE "Valor_test" OF "ListaValores": Instrucción[; Instrucción, ...] "ListaValores": Instrucción[; Instrucción, ...] [ELSE Instrucción Else[; Instrucción Else, ...]] END_CASE;	La instrucción CASE ejecuta uno de varios grupos de instrucciones en función del valor de una expresión.

O significado dos diferentes elementos da expresión CASE é o seguinte.

Parámetro	Descripción
"Valor_Test"	Requerida. Cualquier expresión numérica del tipo de datos Int
"ListaValores"	Requerida. Un valor único o una lista de valores o rangos de valores separados por coma. (Utilice dos periodos para definir un rango de valores: 2..8) El siguiente ejemplo ilustra las diferentes variantes de la lista de valores: 1: Instrucción_A; 2, 4: Instrucción _B; 3, 5..7,9: Instrucción _C;
Instrucción	Requerida. Se ejecutan una o más instrucciones cuando "Valor_test" coincide con cualquier valor de la lista de valores
Instrucción Else	Opcional. Una o más instrucciones que se ejecutan si no hay ninguna concordancia con un valor de "ListaValores"

12.3 Instrucción FOR

Emprégase para repetir un número de veces unha serie de instruccións.

SCL	Descripción
FOR "variable_control" := "inicio" TO "fin" [BY "incremento"] DO instrucción; ; END_FOR;	Una instrucción FOR se utiliza para repetir una secuencia de instrucciones mientras la variable de control se encuentre dentro del rango de valores especificado. La definición de un bucle con FOR incluye la especificación de un valor inicial y otro final. Ambos valores deben ser del mismo tipo de datos que la variable de control.

Onde os elementos da instrucción son os seguintes.

Parámetro	Descripción
"variable_control"	Requerida. Un entero (Int o DInt) que sirve como contador de bucles
"inicio"	Requerida. Expresión simple que especifica el valor inicial de las variables de control

Parámetro	Descripción
"fin"	Requerida. Expresión simple que determina el valor final de las variables de control
"Incremento"	Opcional. Cantidad con la que una "variable de control" incrementa después de cada bucle. El "incremento" debe tener el mismo tipo de datos que la "variable de control". Si el valor de "incremento" no está especificado, el valor de las variables de ejecución se incrementará en 1 después de cada bucle. No es posible cambiar el "incremento" mientras se ejecuta la instrucción FOR.

12.4 Instrucción WHILE

Repite unha serie de instrucción mentres se cumpra unha condición lóxica.

SCL	Descripción

`WHILE "condición" DO`
`  Instrucción;`
`  Instrucción;`
`  ...;`
`END_WHILE;`
	La instrucción WHILE realiza una serie de instrucciones hasta que una condición determinada es TRUE.
	Los bucles WHILE se pueden anidar. La instrucción END_WHILE se refiere a la última instrucción WHILE ejecutada.

Os parámetros son:

Parámetro	Descripción
"condición"	Requerida. Una expresión lógica que evalúa si el estado es TRUE o FALSE. (Una condición "null" se interpreta como FALSE.)
Instrucción	Opcional. Una o más instrucciones que se ejecutan hasta que la comprobación de la condición sea TRUE.

12.5 Instrucción REPEAT-UNTIL

Similar á anterior, repite unha serie de instrucción ata que deixe de ser válida unha condición lóxica. Á diferencia da anterior (WHILE), neste caso, polo menos unha vez execútanse as instruccións que van dentro do bucle.

SCL	Descripción

`REPEAT`
`  Instrucción;`
`  ;`
`UNTIL "condition"`
`END_REPEAT;`
	La instrucción REPEAT ejecuta una serie de instrucciones hasta que una condición determinada es TRUE.
	Los bucles REPEAT se pueden anidar. La instrucción END_REPEAT se refiere a la última instrucción REPEAT ejecutada.

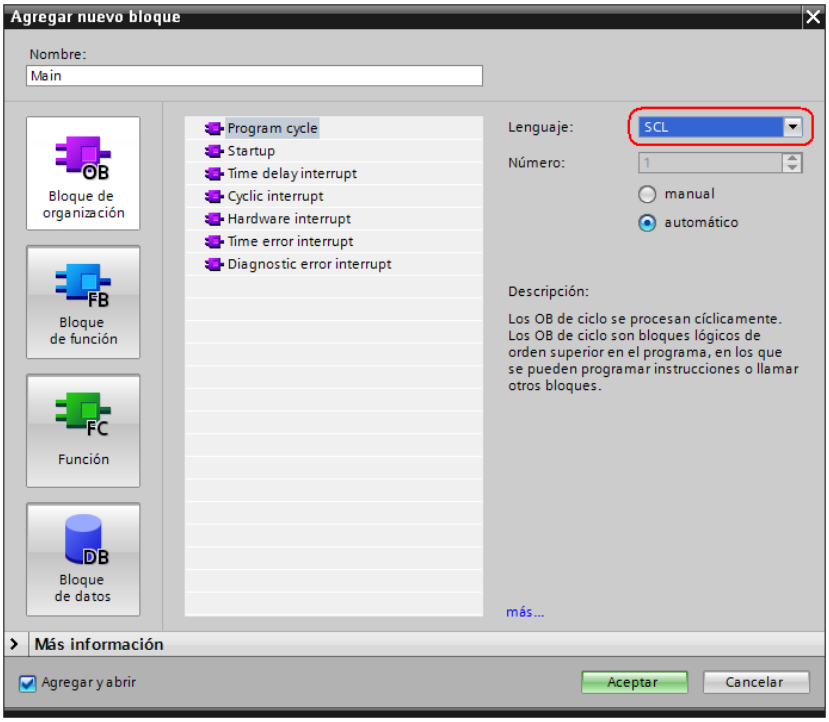
Os parámetros son:

Parámetro	Descripción
Instrucción	Opcional. Una o más instrucciones que se ejecutan hasta que la condición sea TRUE.
"condition"	Requerida. Una o más expresiones del siguiente modo: Una expresión numérica o de cadena que evalúa si el estado es TRUE o FALSE. Una condición "null" se interpreta como FALSE.

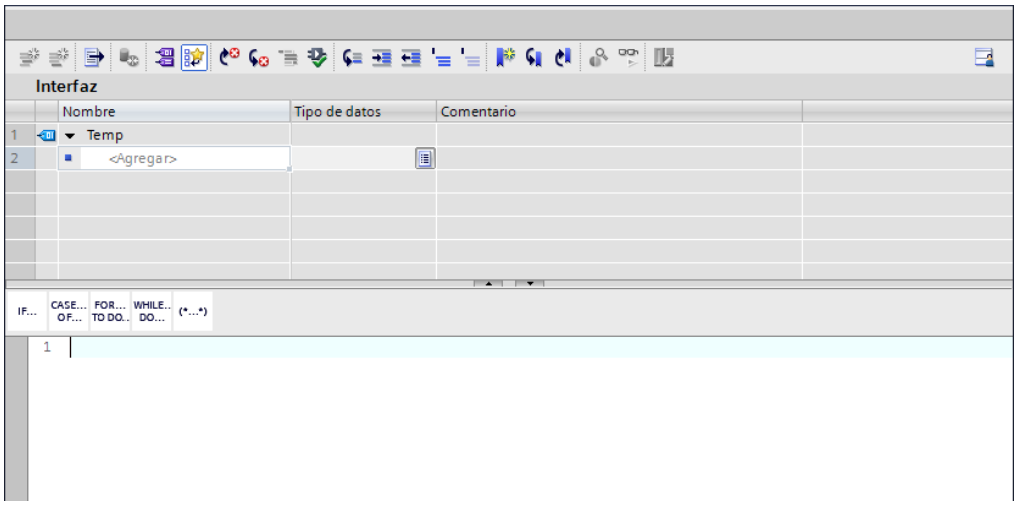
12.5.1.1 Exercicio 27

Imos repetir o exercicio 1 empregando linguaxe SCL.

O primeiro que temos que especificar é que o OB1 ten que ser programado nesta linguaxe.



Cando cambiamos a SCL, a ventá de programación toma o aspecto que vemos na seguinte imaxe.



Agora temos que escribir código como se dun programa informático se tratara, respetando a sintaxe de SCL (Pascal).

Se definimos o nome das variables antes de empezar a programar, cando as empreguemos no programa, cambiarase o nome xenérico polo asignado.

Tabla de variables estándar

	Nombre	Tipo de datos	Dirección	Rema...	Visibl...	Acces...
1	Marcha	Bool	%I0.0	☐	☑	☑
2	Térmico	Bool	%I0.1	☐	☑	☑
3	Manómetro	Bool	%I0.2	☐	☑	☑
4	Motor	Bool	%Q0.0	☐	☑	☑
5	Alarma Térmico	Bool	%Q0.1	☐	☑	☑
6	ON	Bool	%Q0.2	☐	☑	☑
7	MarcaEntrada	Byte	%MB100	☐	☑	☑
8	ByteEntrada	Byte	%IB0	☐	☑	☑
9	<Agregar>			☐	☑	☑

O programa quedaría como o seguinte:

```
 1  "ByteEntrada" := "MarcaEntrada";
 2
 3  IF ("Marcha" = TRUE) AND ("Térmico" = TRUE) AND ("Manómetro" = TRUE) THEN
 4    "Motor" := TRUE;
 5  ELSE
 6    "Motor" := FALSE;
 7  END_IF;
 8
 9  IF ("Marcha"=TRUE) AND ("Térmico" = FALSE) THEN
10    "Alarma Térmico":= TRUE;
11  ELSE
12    "Alarma Térmico" := FALSE;
13  END_IF;
14
15  IF ("Marcha" = TRUE) AND ("Térmico" = TRUE) THEN
16    "ON" := TRUE;
17  ELSE
18    "ON" := FALSE;
19  END_IF;
20
```

Nótese que hai que activar as saídas si se cumpre a condición lóxica e tamén hai que desactivalas se non se cumpre, porque se non o facemos, quedan activadas.

Cando programamos en SCL tamén podemos ver a evolución do programa se nos conectamos *on-line* .

```
1  "ByteEntrada" := "MarcaEntrada";
2
3  IF ("Marcha" = TRUE) AND ("Térmico" = TRUE) AND ("Manómetro" = TRUE) THEN
4    "Motor" := TRUE;
5  ELSE
6    "Motor" := FALSE;
7  END_IF;
8
9  IF ("Marcha"=TRUE) AND ("Térmico" = FALSE) THEN
10   "Alarma Térmico":= TRUE;
11 ELSE
12   "Alarma Térmico" := FALSE;
13 END_IF;
14
15 IF ("Marcha" = TRUE) AND ("Térmico" = TRUE) THEN
16   "ON" := TRUE;
17 ELSE
18   "ON" := FALSE;
19 END_IF;
20
21
```

"ByteEntrada"	16#1
Resultado	FALSE
"Motor"	
"Motor"	FALSE
Resultado	TRUE
"Alarma Térmico"	TRUE
"Alarma Térmico"	
Resultado	FALSE
"ON"	
"ON"	FALSE

Podemos incluso misturar programas empregando diferentes linguaxes de programación. Vexamos un exemplo.

12.5.1.2 Exercicio 28

Repetiremos o exercicio 25, empregando linguaxe KOP para o módulo principal (OB1) e linguaxe SCL para a función.

Non se repite o código KOP do módulo OB1 porque sería o mesmo que o do exercicio 25.

A programación da función sería:

```
1  #Consigna_Normalizada := #Consigna / 100;
2
3  #Consigna_Escalada := TRUNC(#Consigna_Normalizada * 27648);
4
5  IF  #Altura<= #Consigna_Escalada THEN
6    #Control_Deposito := 27648;
7  ELSE
8    #Control_Deposito := 0;
9  END_IF;
```

As liñas equivalentes ás instruccións NORM_X e SCALE_X da linguaxe KOP son agora as dúas primeiras.

13 Introducción á comunicación entre autómatas

Nos entornos de fabricación actuais existen conceptos como o de trazabilidade, nos que o manexo e intercambio de datos do proceso é necesario.

Ábrense, polo tanto, múltiples posibilidades de comunicación no contorno dos autómatas programables. Podemos comunicar dous autómatas entre sí, un autómata cunha pantalla táctil ou mesmo con un ordenador. Podemos facer que o tipo de comunicación sexa maestro-esclavo, onde un autómata ou PC funciona como maestro dirixindo a comunicación, e outro como esclavo, aceptando as peticións de información que lle faga o maestro.

13.1 Comunicación entre dous autómatas

Trátase do caso máis simple, no que dispoñemos de dous autómatas conectados entre sí e intercambiando información.

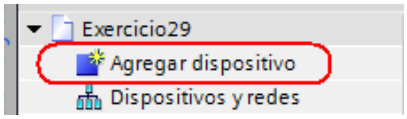
Nunha montaxe industrial, se queremos facer esta conexión, deberiamos empregar un switch de Profinet tipo CSM1277 como o da figura.

Para unha práctica a nivel docente, chéganos con empregar un swith dos utilizados en calquera rede Ethernet.

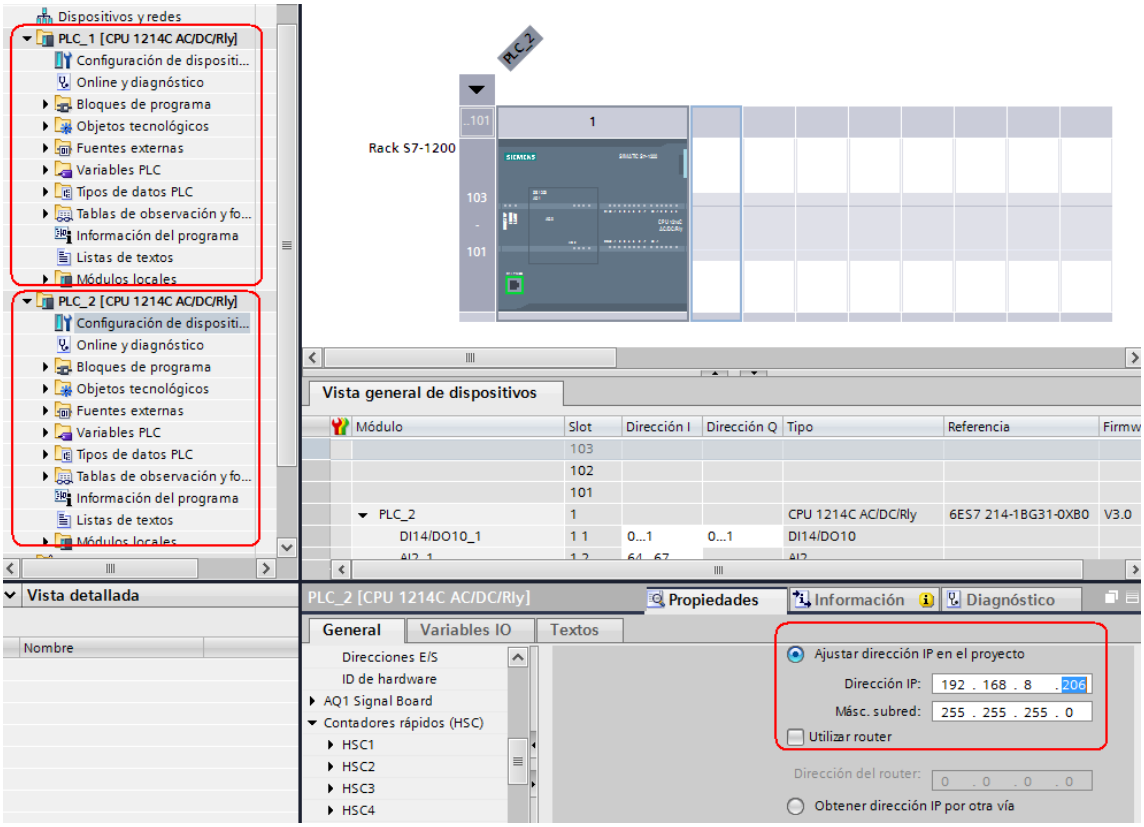
13.1.1.1 Exercicio 29

Repetiremos o exercicio 23 enviándolle os datos a un segundo autómata. Neste segundo autómata visualizaremos a apertura da válvula na táboa de observación de variables.

Abrimos o exercicio 23 e engadimos o segundo autómata facendo dobre click sobre **Agregar dispositivo**.

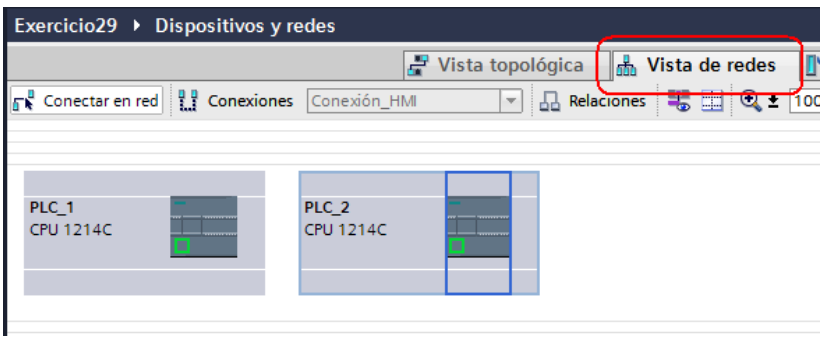


Seleccionamos o hardware do novo plc e asignámoslle un enderezo IP na mesma subrede que o primeiro autómata. No exemplo teremos IP: 192.168.8.206 e máscara de subrede 255.255.255.0

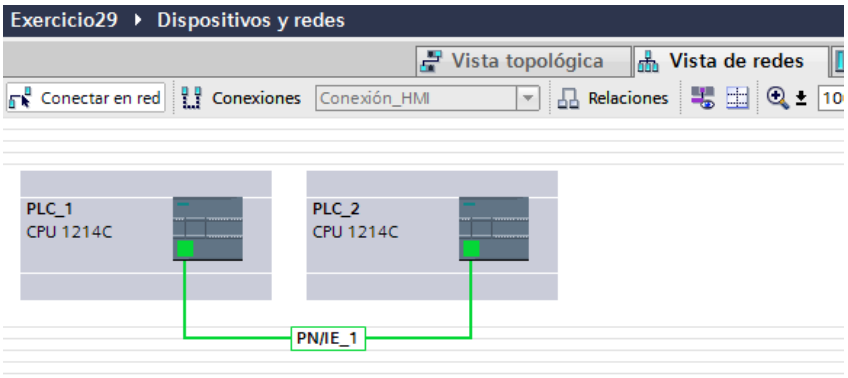


Agora temos dous autómatas (PLC_1 e PLC_2) con enderezos IP 192.168.8.205 e 192.168.8.206 respectivamente.

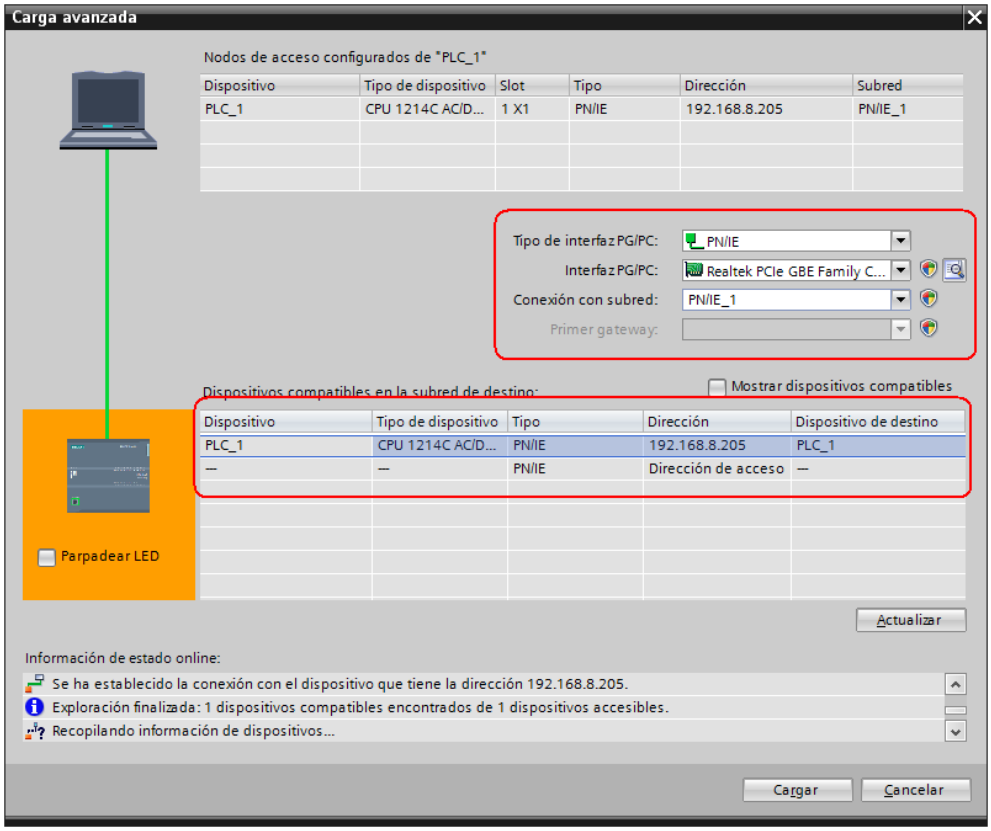
O seguinte paso consiste en conectar os dous autómatas en rede. Para facelo, cambiamos á **vista de redes**.

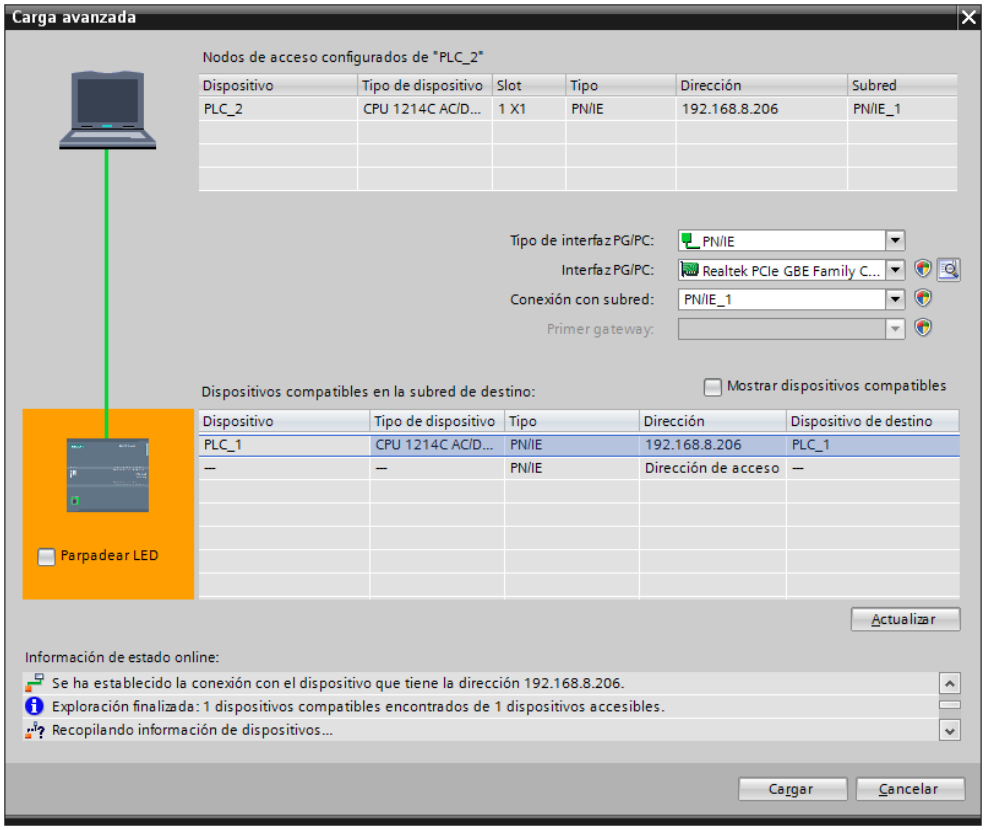


Arrastramos desde a conexión de rede dun autómata ata o outro, e créase a conexión entre os dous autómatas.



A continuación cargaremos en cada autómata a configuración.

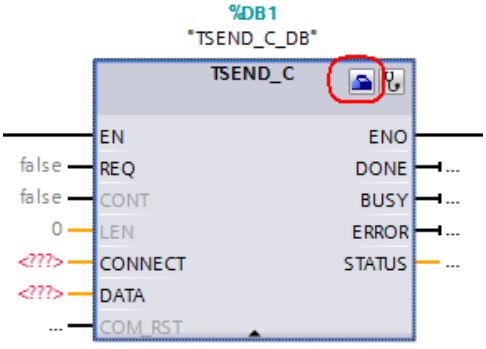




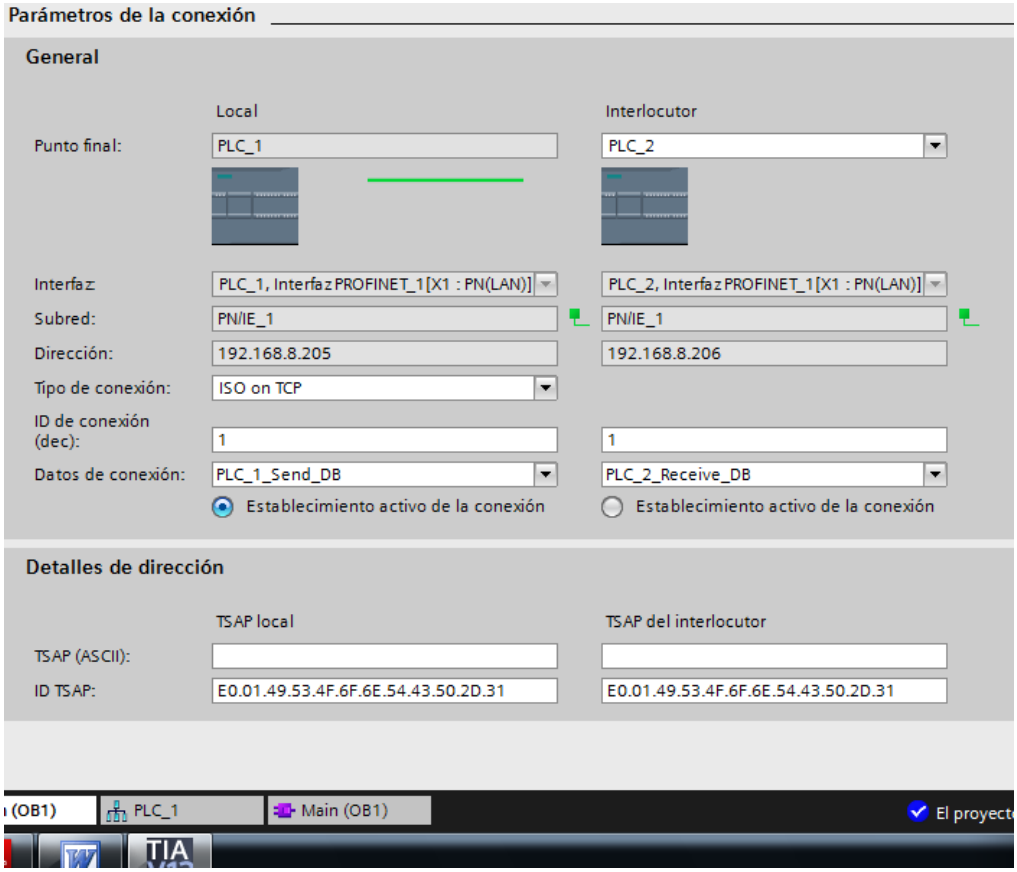
A continuación temos que incorporar aos programas dos autómatas as instruccións que lle van permitir comunicarse (TSEND_C e TRCV_C).

No OB1 do primeiro autómata incorporamos unha instrucción TSEND_C.

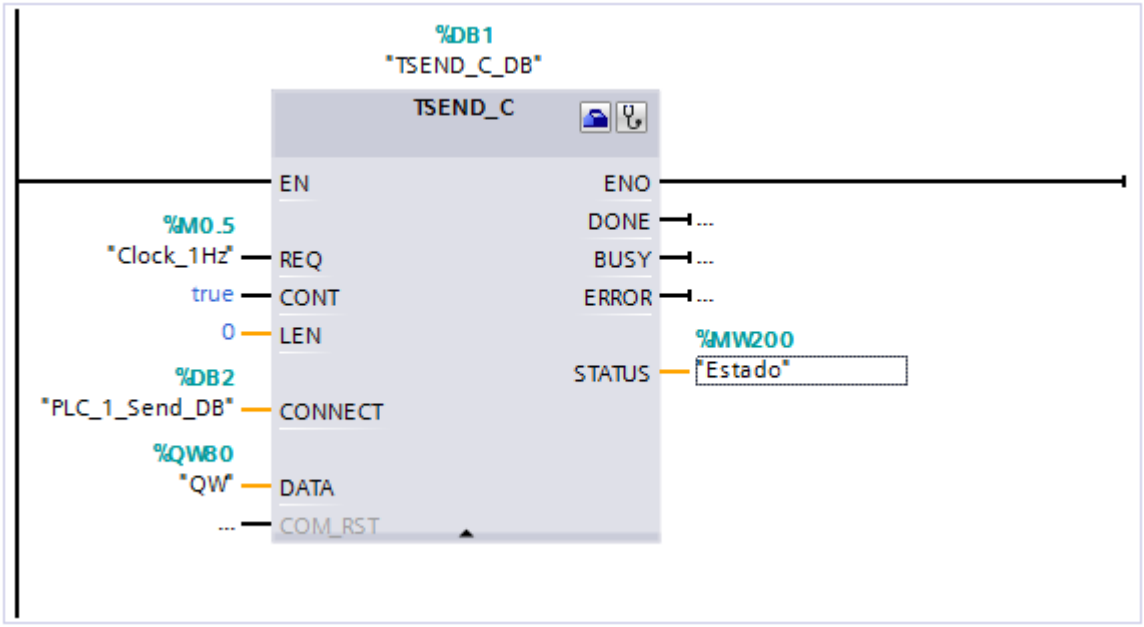
Para que a comunicación teña lugar, hai que configurar TSEND_C pulsando sobre o botón **Iniciar configuración**.



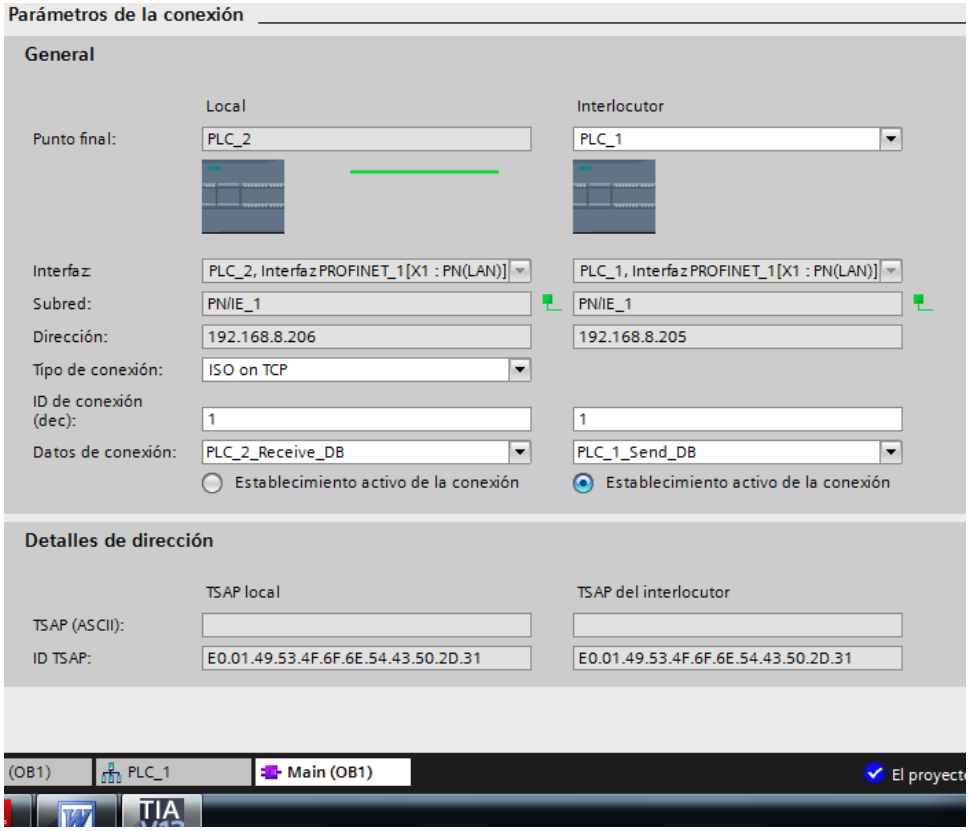
Na ventá de propiedades temos que completar todos os cadros que están en cor rosa.

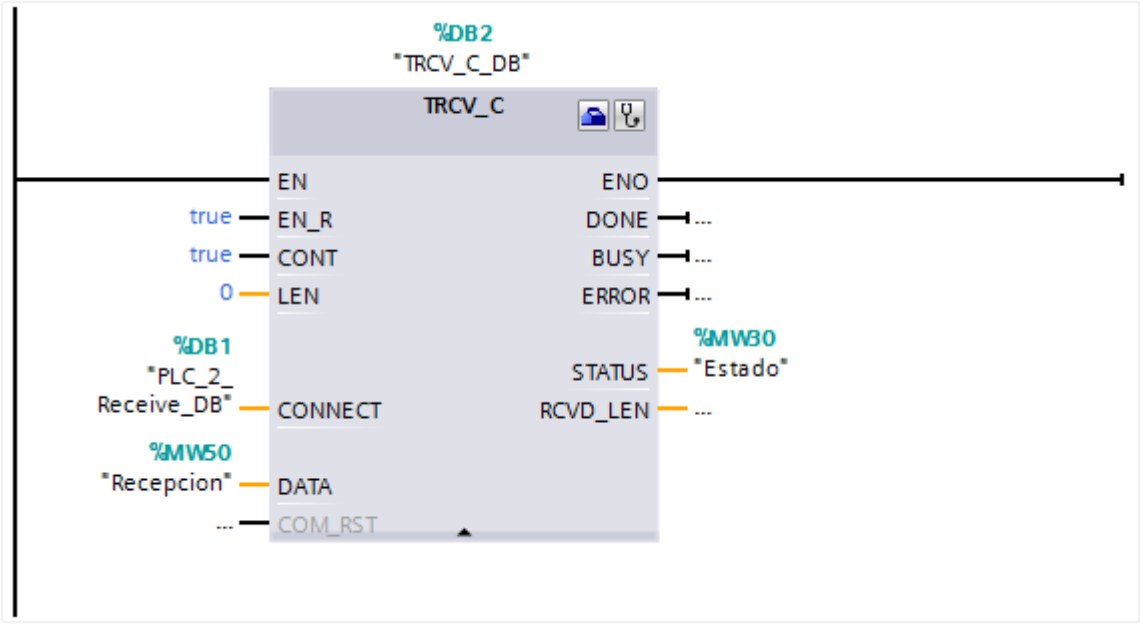


Na instrucción TSEND_C, a petición de transmisión (REQ) debe ser cíclica, polo que activamos as marcas de ciclo e colocamos unha en dita entrada.



Agora, no segundo autómata programamos a instrucción TRCV_C. Colocamos a instrucción no OB do segundo autómata e configuramos.





Xa podemos volcar os programas nos respectivos autómatas e probalos.

13.1.1.2 Exercicio 30

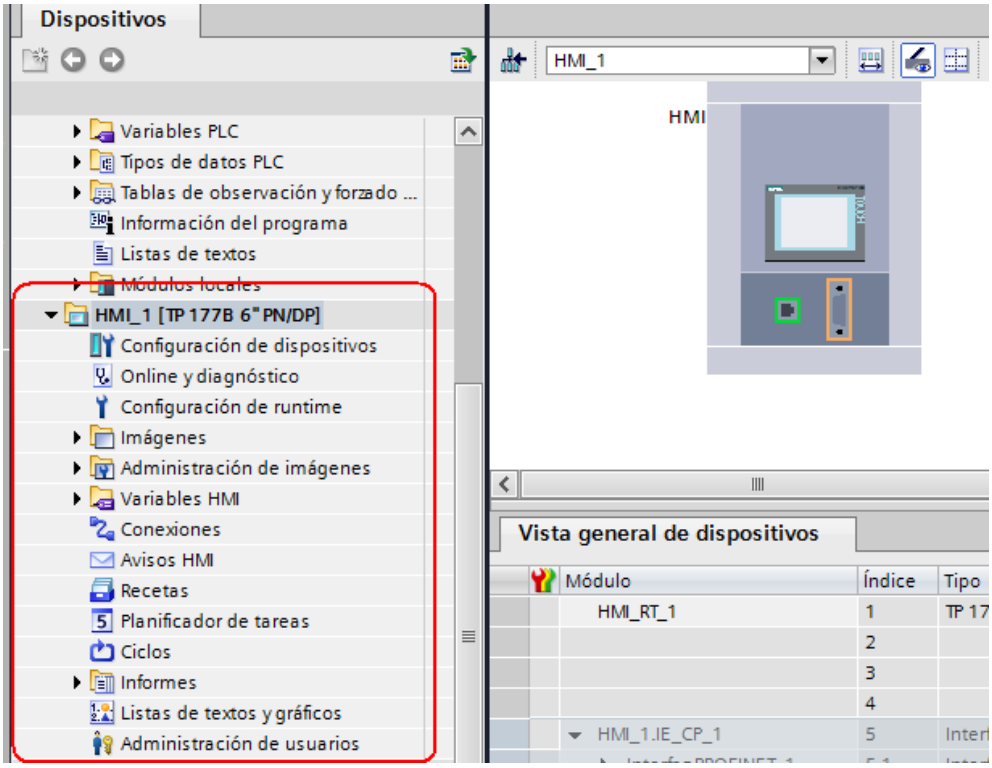
Neste caso imos facer unha comunicación entre un autómata e unha pantalla táctil. Sobre a base do mesmo exercicio 23.

Trátase dunha pantalla TP177B PNDP.

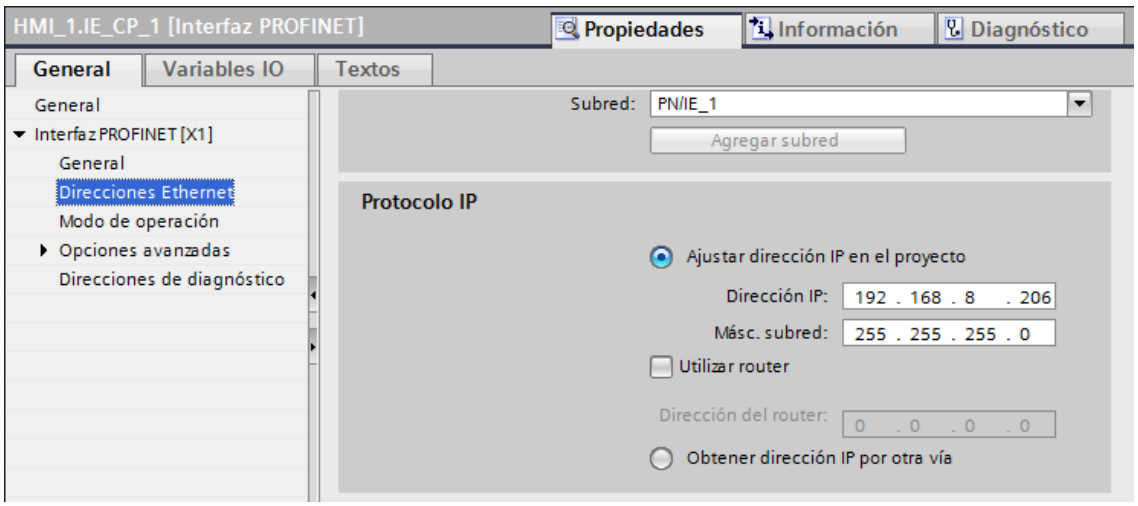
Para comunicar con ela, primeiro temos que configurarlle no panel de control da pantalla un enderezo IP na mesma subrede que a do autómata, por exemplo 192.168.8.206.

No proxecto de TIA Portal engadimos un novo dispositivo de tipo HMI e localizamos o modelo de pantalla.

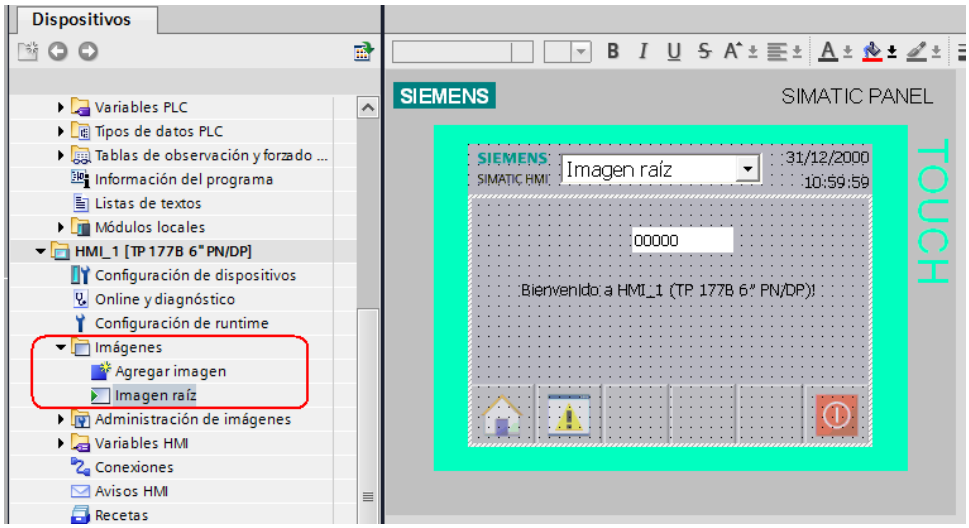
Despois de seguir un pequeno asistente onde se nos pregunta se queremos configurar unha serie de botóns por defecto, os avisos de alarmas, entre outros, xa temos a pantalla no proxecto.



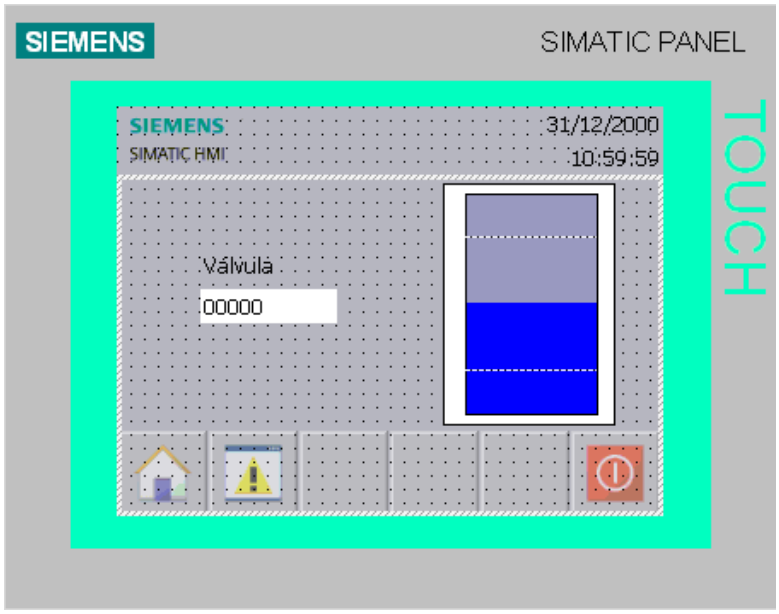
Configuraremos o enderezo IP como fixemos para os autómatas.



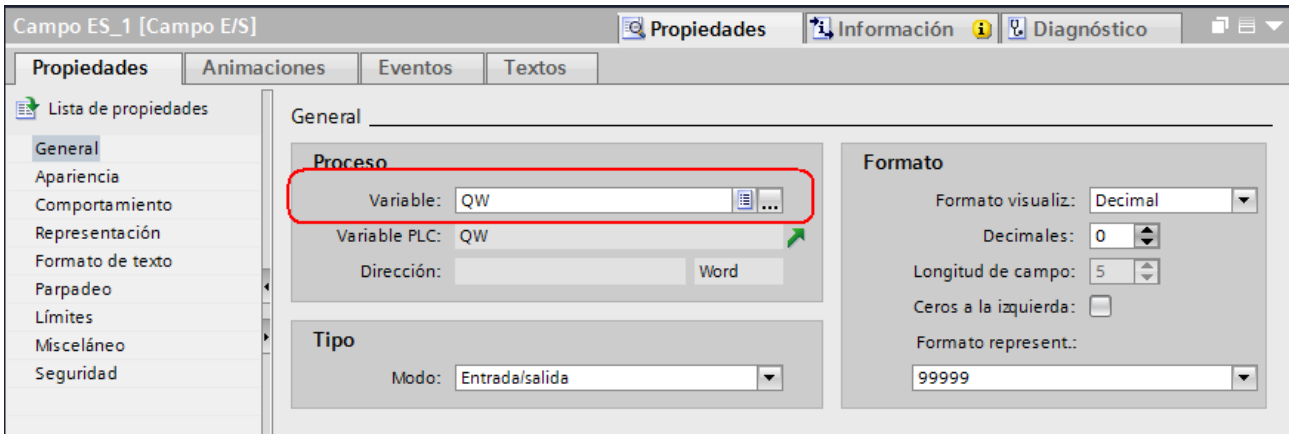
E deseñamos a pantalla. Esto faise no apartado **Imágenes** da mesma.

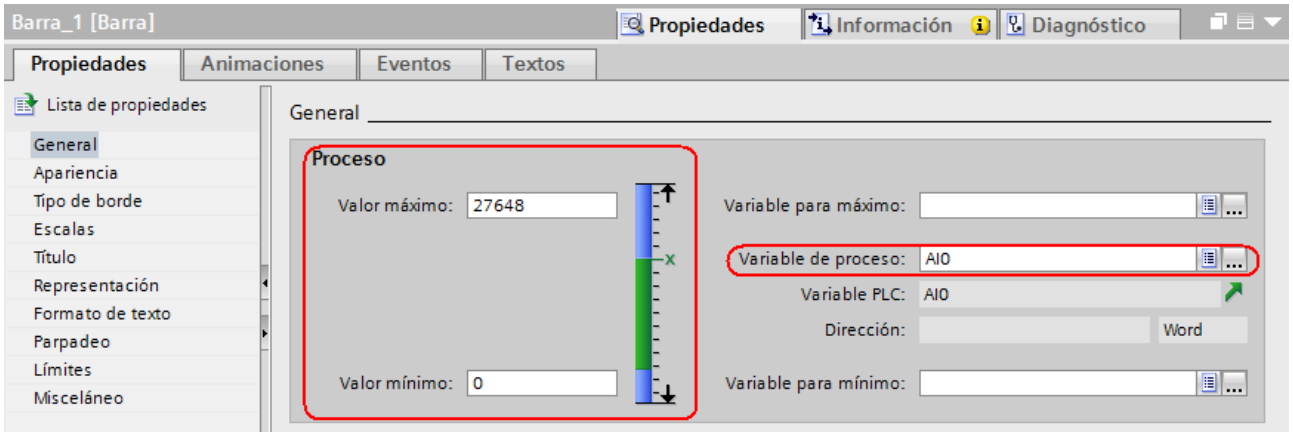


Por exemplo, poñemos un cadro no que se visualice o valor da válvula e un indicador no que se visualice o nivel do depósito.



Conectamos os obxectos coas variables do autómata.





Volcamos o programa na pantalla e comprobamos o funcionamento.

14 A norma IEC 61131-3

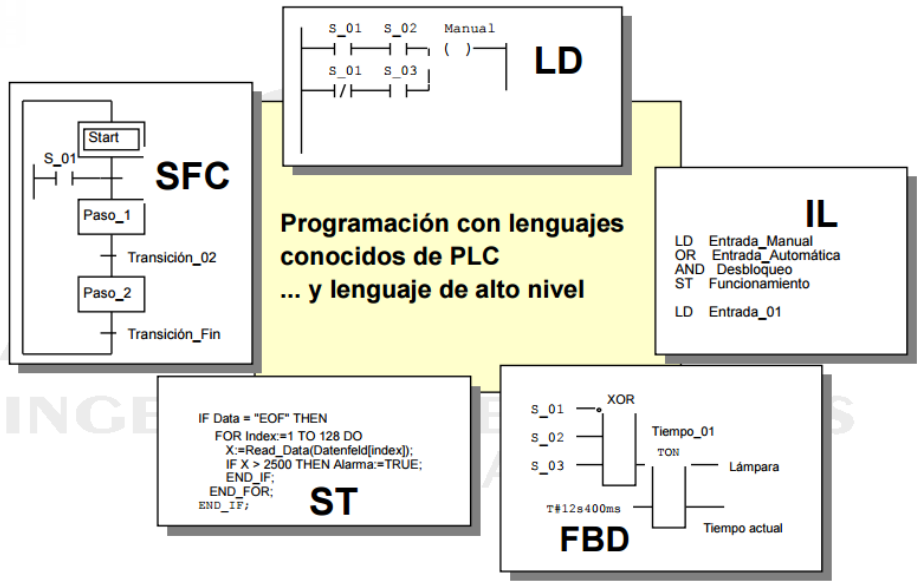
Como se dixo no capítulo 6 deste documento, diferentes fabricantes de PLCs decidiron unificar as linguaxes de programación a empregar e propuxeron á Comisión Electrotécnica Internacional, ao principio dos anos 90 a elaboración dunha norma a tal efecto. Naceu nese momento a **IEC 61131**, que no seu apartado 3 fala das diferentes linguaxes.

A día de hoxe, fabricantes como Schneider, Codesys, Bechkoff, Bosch, ABB, Phoenix Contact e outros traballan co este estándar. Cada un elaborou un software para a programación dos seus PLCs baseado na norma mencionada, polo que o cambio dun a outro é moi sinxelo.

Aínda que existen librerías de funcións propias dalgúns fabricantes, a meirande parte das linguaxes de programación son comúns a todos eles.

14.1 Linguaxes da norma

Basicamente temos cinco linguaxes fundamentais: LD, FBD, ST, IL e SFC



14.1.1 Linguaxes literais

- Lista de instruccións ("*Instruction List*" IL): Trátase dunha linguaxe parecida á que se emprega para programar microprocesadores (linguaxe ensambladora).
- Texto estructurado ("*Structured Text*" ST): Linguaxe de alto nivel, mistura entre Basic e Pascal.

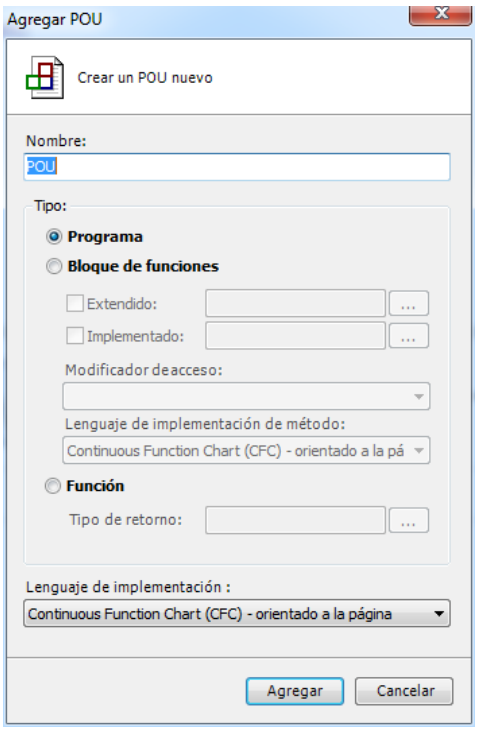
14.1.2 Linguaxes gráficos

- Diagrama en escaleira ("*Ladder Diagram*" LD): Linguaxe de contactos similar a un esquema eléctrico de contactos.
- Diagrama de Bloques Funcionais ("*Function Block Diagram*", FBD): Linguaxe que emprega bloques funcionais similiar ao utilizado en esquemas electrónicos.
- Gráfico Funcional Secuencial ("*Sequential Function Chart*", SFC): Aplicación da ferramenta GRAFCET na programación directa dos PLCs.

14.2 Unidades de Organización de Programa (POU)

A norma define tres formas distintas de presentar ou crear programas de control para PLCs, chamadas Unidades de Organizaciónde Programa (POU):

- Programas
- Funcións
- Bloques funcionais



14.2.1 Programas

A norma define un **programa** como o conxunto lóxico de todos os elementos e construccións que son necesarios para o tratamento de sinais que se requiren para o control dunha máquina ou proceso mediante un PLC.

É dicir, que un programa pode conter a declaración de tipos de datos, variables e instancias de bloques funcionais xunto coas instrucións (código ou programa propiamente dito) necesario para levar a cabo o control desexado do proceso ou máquina.

14.2.2 Funcións

Especifícanse funcións estándar e funcións definidas polo usuario. As estándar son, por exemplo, ADD(suma), ABS(valor absoluto), SQRT(raíz cuadrada), SIN(seno), entre outras.

As definidas polo usuario, unha vez implementadas poden ser usadas indefinidamente en calquera POU.

As funcións non poden conter ningunha información de estado interno, é dicir, que a invocación dunha función varias veces cos mesmos argumentos (parámetros de entrada) debe subministrar sempre o mesmo resultado (saída).

14.2.3 Bloques funcionais

Os bloques funcionais son os equivalentes dos circuítos integrados en electrónica e representas funcións de control especializadas. Os FBs conteñen tanto datos como instruccións, podendo gardas os valores de ditas variables entre sucesivas execucións (que é unha das diferencias coas funcións). Decimos, polo tanto que os FBs teñen memoria.

Presentan unha interface de entradas/saídas ben definido e un código interno oculto, como un circuíto integrado ou unha caixa negra. Un lazo de control de temperatura PID e un exemplo de bloque funcional. Unha vez definido, pode ser empregado unha e outra vez, no mesmo programa, en diferentes programas ou en diferentes proxectos.

Os bloques funcionais poden ser definidos polo usuario empregando algunha das linguaxes de programación da norma, pero tamén existen FBs estándar (biestables, detección de flancos, contadores, temporizadores, entre outros).

Outra das diferencias fundamentais con respecto ás funcións e que lles da unha grande potencia de uso é a posibilidade de crear tantas copias como se desexe dun mesmo FB. A cada copia chámaselle **instancia**. Cada instancia levará asociado un identificador e unha estrutura de datos que conteña as súas variables de entrada/saída e internas, separadas do resto das instancias.

15 Programación de PLCs con Codesys. Simulación con ControlWin

Unha das primeiras empresas que se decidiu pola norma mencionada no punto anterior foi a alamana 3S-Smart. Esta creou un software de programación na que se basearon outras posteriormente, chamado CODESYS.

Este software pódese descargar da páxina do fabricante: https://www.codesys.com . Ten partes coas que se pode traballar sen custo e outras polas que hai que pagar unha licenza.

Neste capítulo imos ver como traballar con CODESYS e virtualmakTCP para aprender a programar de acordo á norma mencionada.

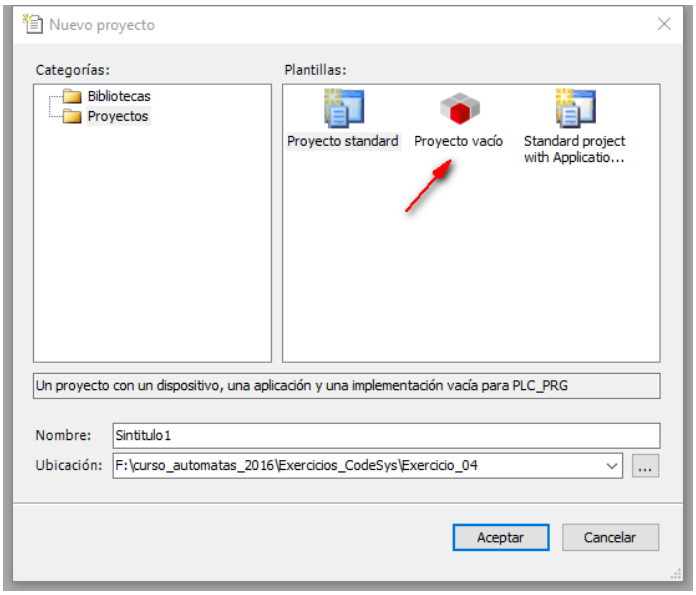
No capítulo seguinte veremos como aplicar este tipo de programación nos PLCs de Schneider.

Traballaremos con CODESYS V3.5 SP8 e co seu módulo ControlWin V3.

Hai que sinalar que o módulo que simula os PLCs (ControlWin V3) funciona durante dúas horas e despois para, sempre e cando non se merque unha licenza do mesmo.

15.1 Primeiros pasos

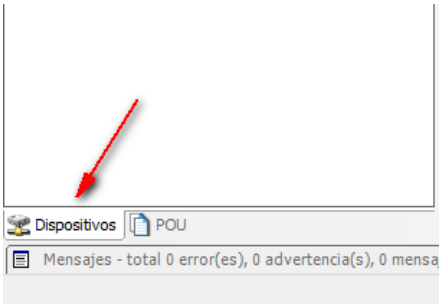
Unha vez instalado, arrancamos o programa, e empezamos creando un proxecto bacío.



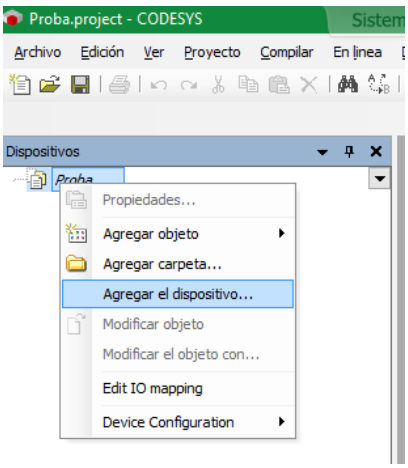
Dámoslle un nome e escollemos a ruta para gardalo.

O primeiro que faremos será escoller o *hardware* que utilizaremos.

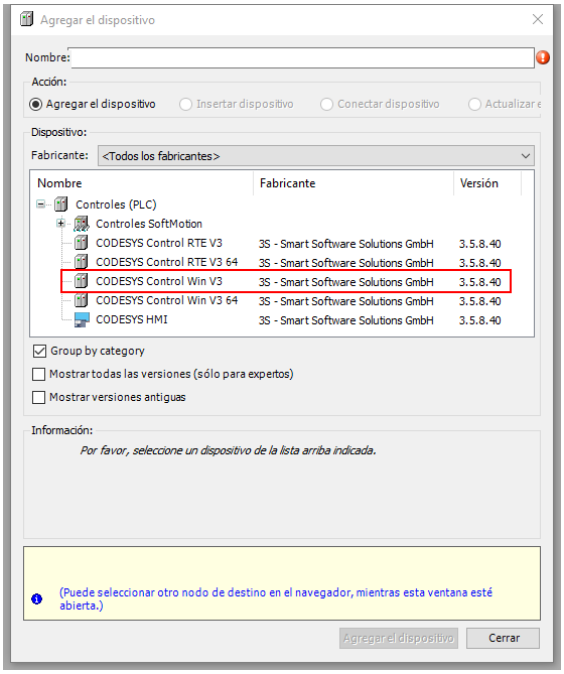
Asegurámonos de que na parte inferior esquerda da aplicación temos seleccionada a pestaña **Dispositivos**.



Na parte superior esquerda aparece o nome do proxecto co que estamos a traballar. Facendo click co botón dereito sobre o mesmo, escollemos a opción **Agregar el dispositivo...**



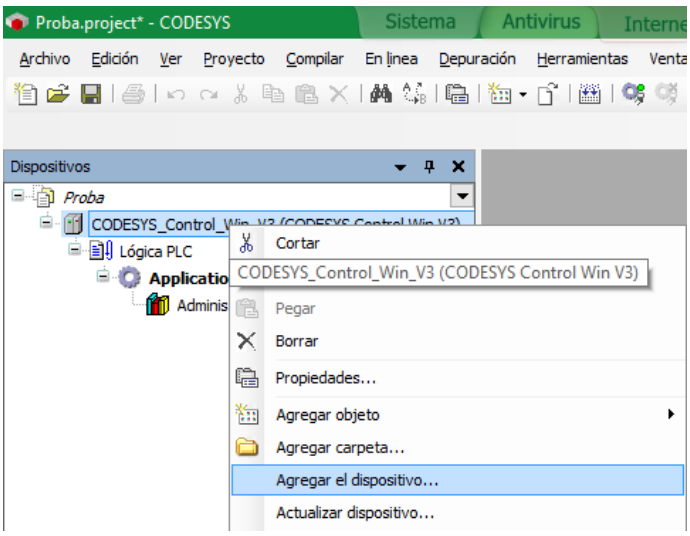
Escollemos na ventá seguinte o controlador **Codesys Control Win V3**.



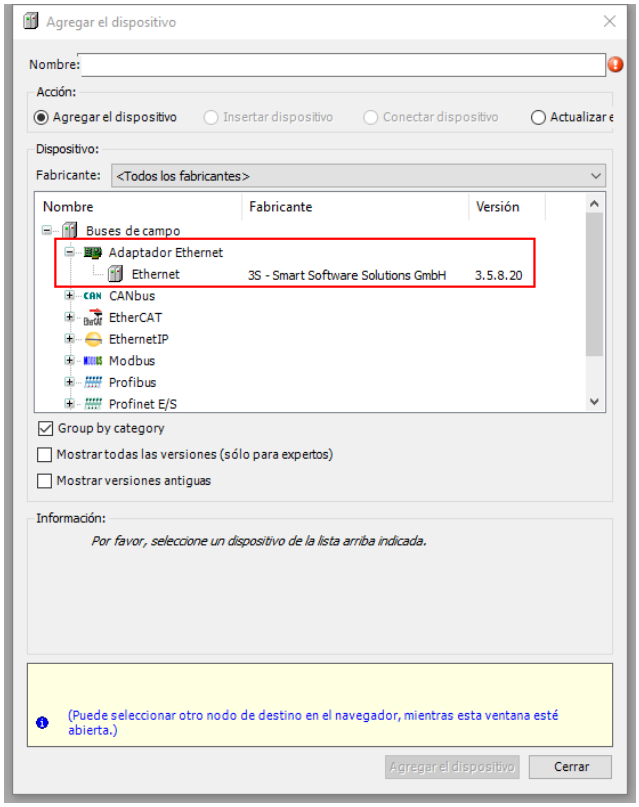
Pulsamos o botón **Agregar el dispositivo** para aceptar, e logo pulsamos o botón **Cerrar**.

Con esto xa poderiamos emprezar a programar o controlador, pero como imos comunicar este co virtualmakTCP, teremos que engadirlle un módulo Ethernet e un xestor de protocolo MODBUS.

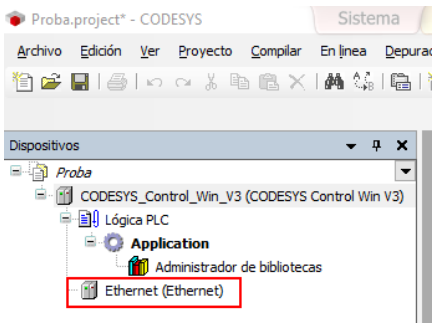
Para facelo, facemos click co botón dereito sobre o controlador na parte superior esquerda, e escollemos de novo **Agregar dispositivo...**



Na ventana que aparece escollemos **Adaprtador Ethernet** e dentro **Ethernet**.

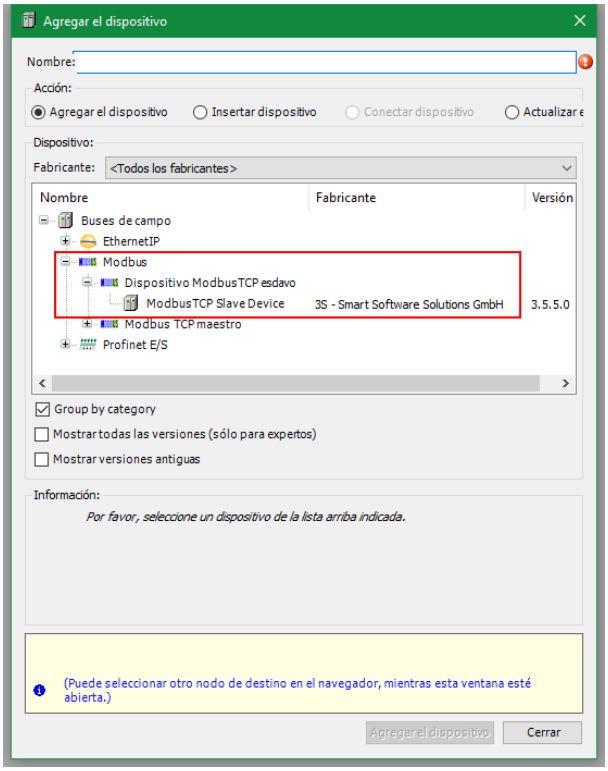


Aceptamos igual que no paso anterior e comprobamos como se engade un módulo de comunicacións ao Control Win.



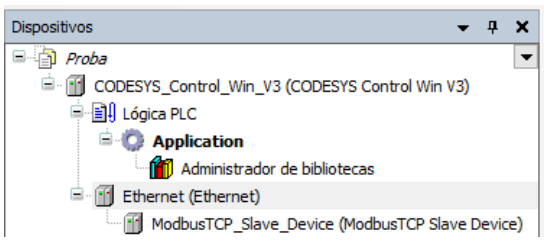
Agora, de novo facemos click co botón dereito sobre o módulo Ethernet e escollemos outra vez **Agregar dispositivo...**

Na ventá que se abre, escollemos **Modbus TCP Slave Device**.



Xa temos configurado o hardware do simulador Control Win V3.

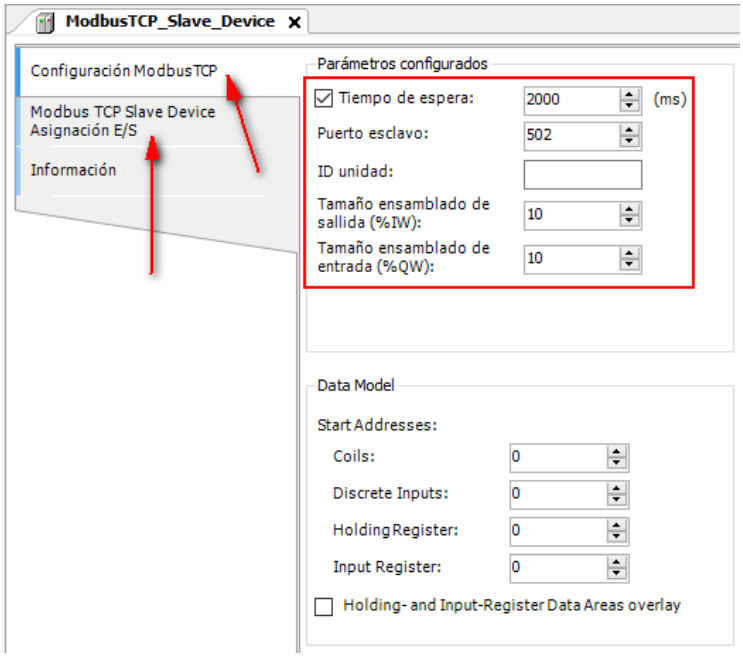
Deberiamos ver algo similar na estrutura do proxecto ao que estamos a ver na imaxe seguinte.



15.2 Configuración da comunicación

Este punto é importante para a correcta comunicación entre o simulador Control Win e virtualmakTCP.

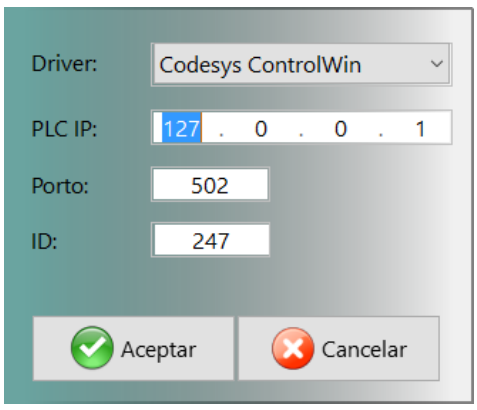
Unha vez configurado o hardware como se explica no punto anterior, facemos doble click sobre **ModbusTCP_Slave_Device** e aparece unha ventá como a seguinte.



Hai tres pestañas en vertical, das cales, de entrada faremos referencia ás dúas primeiras: **Configuración ModbusTC**, e **Modbus TCP SlaveDevice Asignación E/S**.

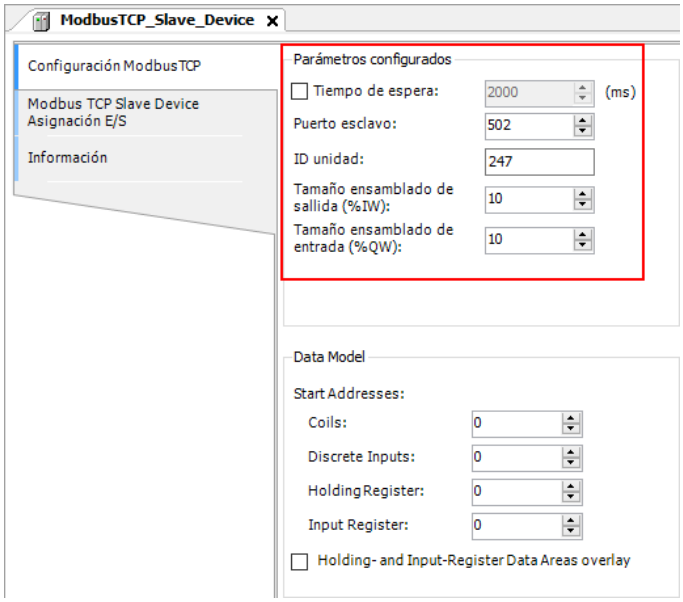
Na primeira, como se ve na figura anterior, teremos en conta o seguinte:

Desmarcaremos a opción **Tiempo de espera** para poder enviarlle os datos desde o virtualmakTCP. Deixaremos o valor 502 en **Puerto esclavo** (porto por defecto de Modbus TCP) e teremos que asignarlle un valor a **ID unidad** que terá que coincidir co que asignaremos en virtualmakTCP á hora de conectarnos.

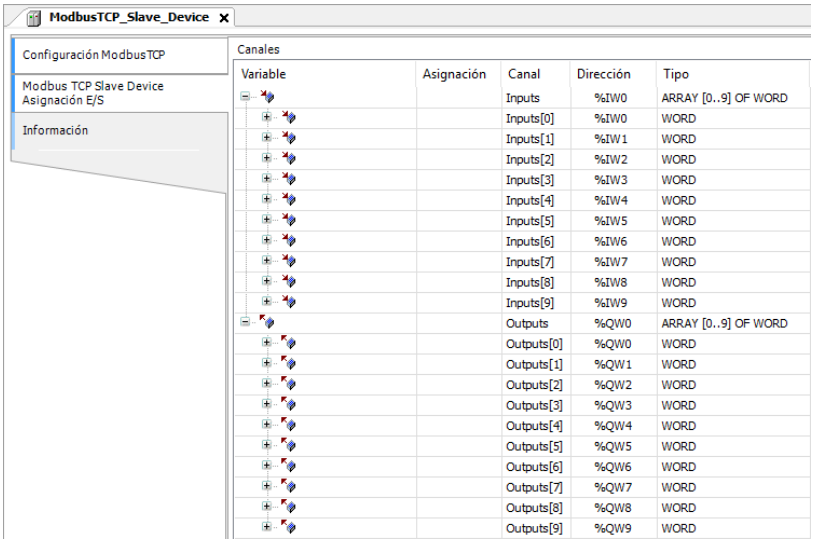


Os dous datos que veñen a continuación (**Tamaño ensamblado de salida** e **Tamaño ensamblado de entrada**) deixámolos por defecto en 10 palabras de entrada e 10 palabras de saída, que son máis que sificientes para as prácticas con virtualmakTCP.

Logo, quedaría a configuración como se ve na imaxe seguinte.



Na seguinte pestaña podemos ver o direccionamento de entradas e saídas do módulo. Serán as que teñamos que empregar nos programas que fagamos.

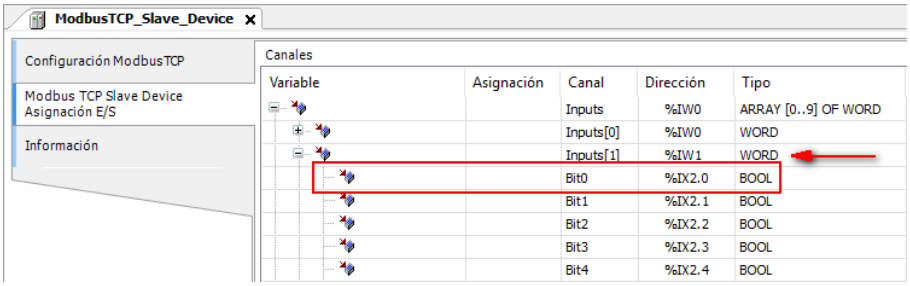


Variable	Asignación	Canal	Dirección	Tipo
		Inputs	%IW0	ARRAY [0..9] OF WORD
		Inputs[0]	%IW0	WORD
		Inputs[1]	%IW1	WORD
		Inputs[2]	%IW2	WORD
		Inputs[3]	%IW3	WORD
		Inputs[4]	%IW4	WORD
		Inputs[5]	%IW5	WORD
		Inputs[6]	%IW6	WORD
		Inputs[7]	%IW7	WORD
		Inputs[8]	%IW8	WORD
		Inputs[9]	%IW9	WORD
		Outputs	%QW0	ARRAY [0..9] OF WORD
		Outputs[0]	%QW0	WORD
		Outputs[1]	%QW1	WORD
		Outputs[2]	%QW2	WORD
		Outputs[3]	%QW3	WORD
		Outputs[4]	%QW4	WORD
		Outputs[5]	%QW5	WORD
		Outputs[6]	%QW6	WORD
		Outputs[7]	%QW7	WORD
		Outputs[8]	%QW8	WORD
		Outputs[9]	%QW9	WORD

Hai que ter coidado se procedemos da programación doutros fabricantes como Siemens, que traballa cos datos en formato palabra de dous en dous (IW0, IW2, IW4) e neste caso temos que traballar de unha en unha (IW0, IW1, IW2).

Tamén temos que acostumarnos a diferenciar entre o formato palabra (IW0) e formato bit (IX0.0). Se abrimos unha calquera das palabras da táboa anterior, entenderemos o que se está a dicir.

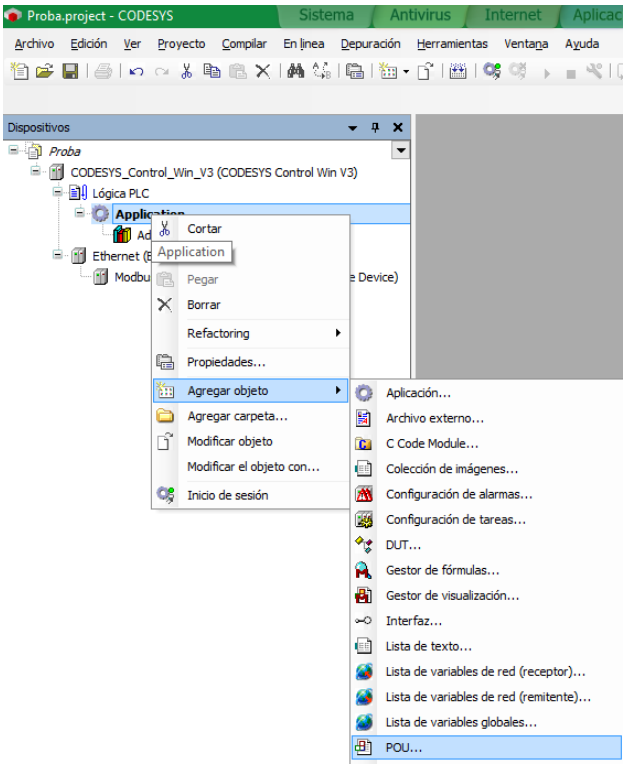
Por exemplo, o primeiro bit da palabra IW1 non é o IX1.0 senón o IX2.0



15.3 Primeiro programa

Empezaremos elaborando un programa básico para ver como é o proceso a seguir ata telo funcionando e interactuando con virtualmakTCP.

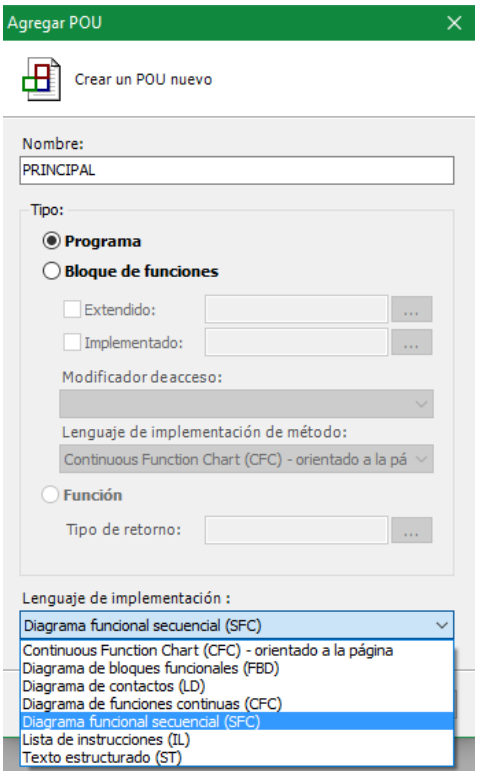
Na árbore do proxecto, facemos click co botón dereito sobre **Application** e escollemos **POU...**



Na ventá que aparece, dámoslle un nome e escollemos un tipo de POU (ver punto 14.2) dos que contempla a norma.

O lóxico será empezar por un programa.

Escollemos tamén a linguaxe de programación desexada, de entre as que contempla a norma (ver punto 14.1).

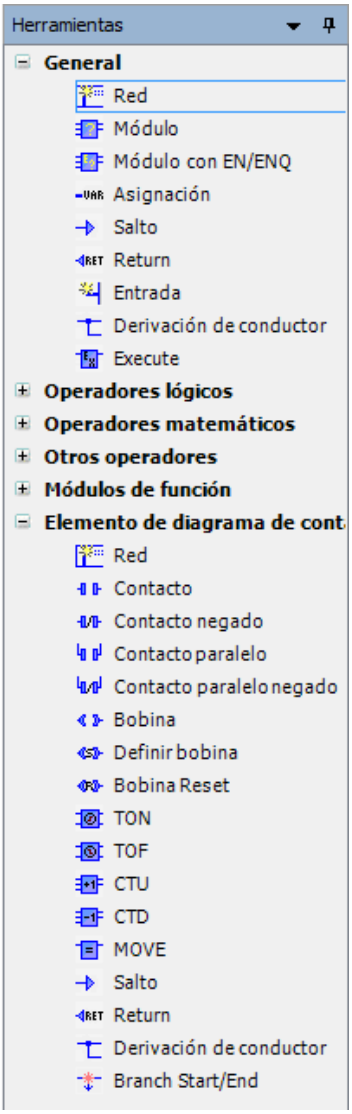


Para este primeiro exemplo traballaremos coa linguaxe de diagrama de contactos (LD) ou Ladder.

Unha vez seleccionado, aparece o editor de Ladder.

Na parte superior temos a zona de declaracións de variables (de acordo á norma IEC 61131-3), e a continuación temos o editor.

Na parte dereita vemos unhas pestañas con ferramentas para a programación LD.



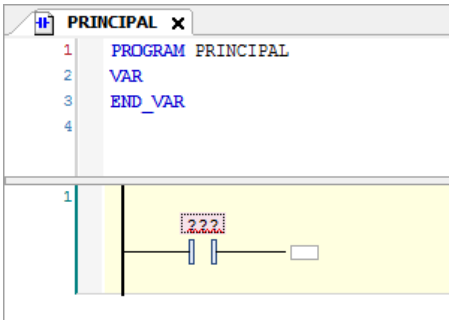
Segundo as preferencias do programador, agora podemos traballar de varias maneiras diferentes. Vexamos algunha.

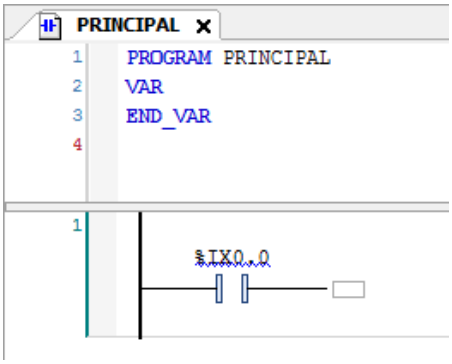
Supoñamos que queremos facer un programa simple cun par de contactos e unha bobina.

Podemos empezar colocando directamente os contactos e a bobina no diagrama.

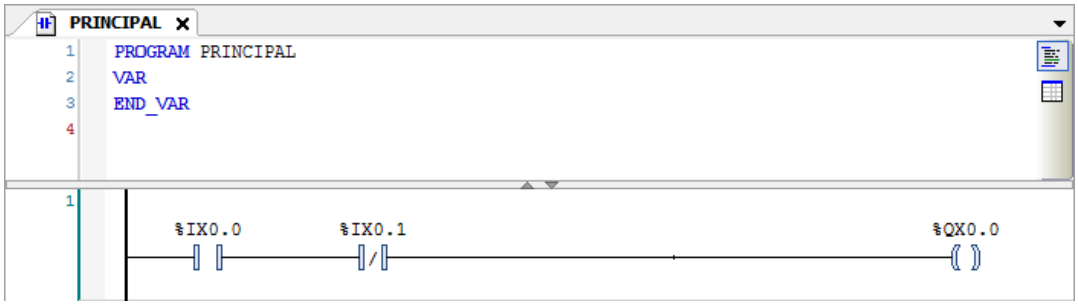
Neste programa é útil o botón dereito do rato para inserir elementos no programa.

Cando colocamos o primeiro contacto, aparecen ??? enriba do mesmo. Se editamos esas interrogacións e asignamos unha entrada, xa quedaría conectada coa mesma.





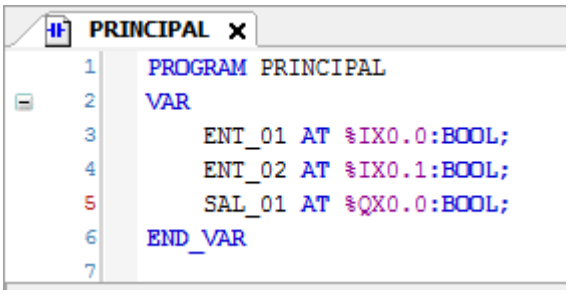
Deste xeito completamos o programa básico.



Fixémonos que, o feito de empezar por %IX ou %QX, indícalle ao PLC que se trata de sinais booleanas que proceden ou que van directamente ás entradas/saídas físicas do PLC. Esto será sempre así cando programemos de acordo á IEC 611131-3.

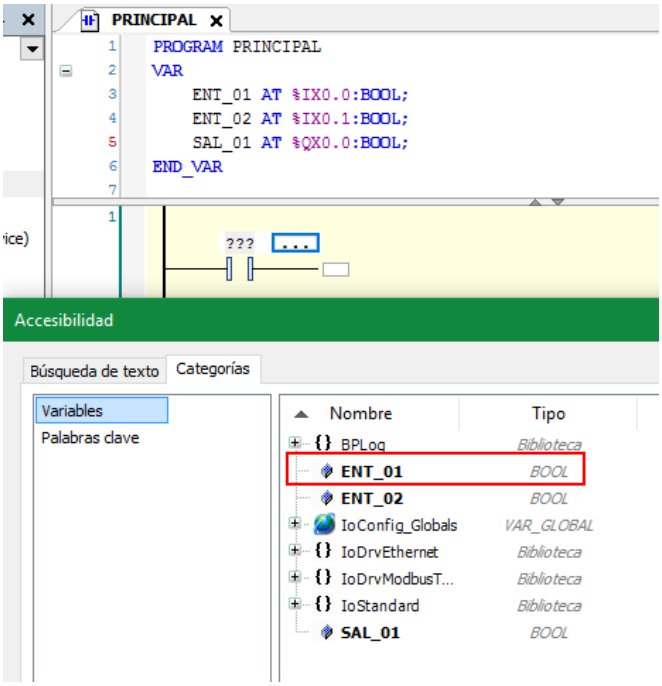
Outra maneira de traballar consiste en declarar primeiro na zona de variables as entradas e/ou saídas que queremos empregar e logo asignalas ao colocar os contactos.

Deste xeito empezariamos colocando as variables.

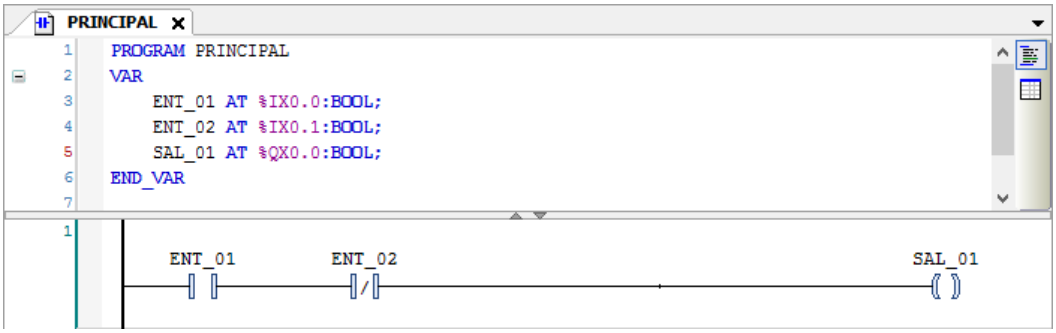


E agora, colocariamos os contactos.

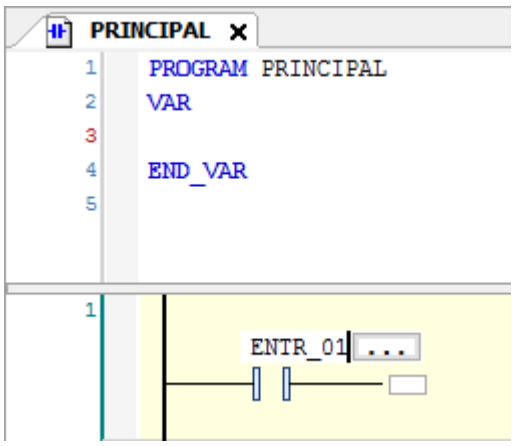
Neste caso, cando poñemos un contacto, en vez de escribir nas tres ???, pulsamos o botón con tres puntos que aparece á dereita do mesmo, e ábrese unha ventá que nos permite escoller entre as variables que temos declaradas.



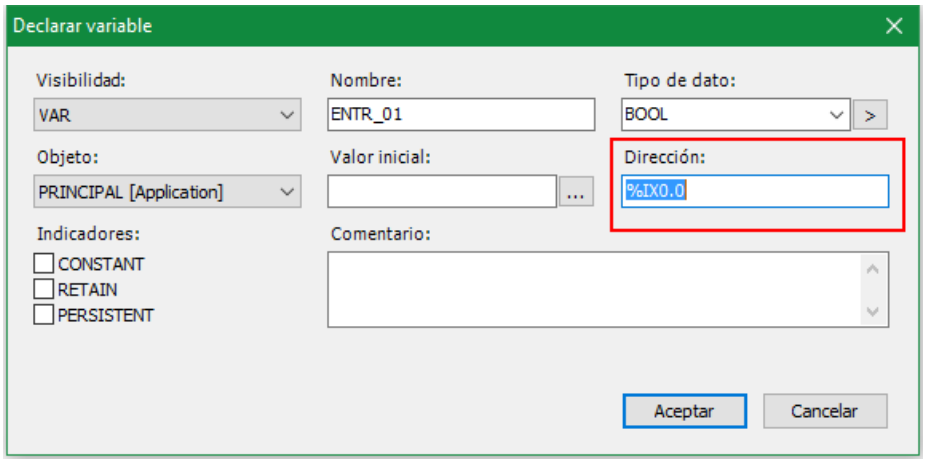
Completamos deste xeito o programa.



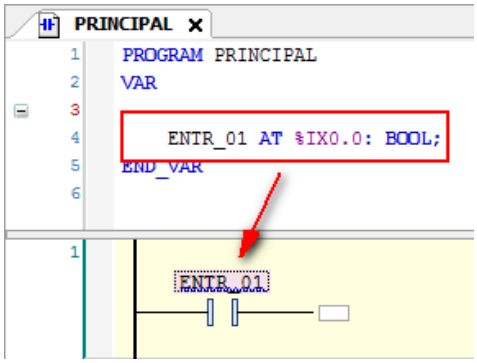
Unha terceira maneira de traballar consiste en ir colocando os contactos sen declarar as variables previamente, e en vez de asignarlle %IX0.0, darlle un nome. Neste caso o sistema pediranos con que entrada ou saída queremos conectar ese nome e creará directamente a variable.



Cando aceptamos, aparece a ventá na que conectamos ese nome cunha entrada física.



Especificamos a dirección e comprobamos como se engade na parte superior esa variable.



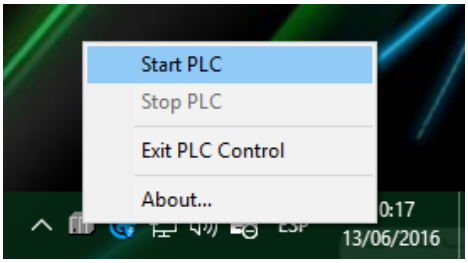
Facemos o mesmo para os demais elementos.

15.4 Simulando o programa

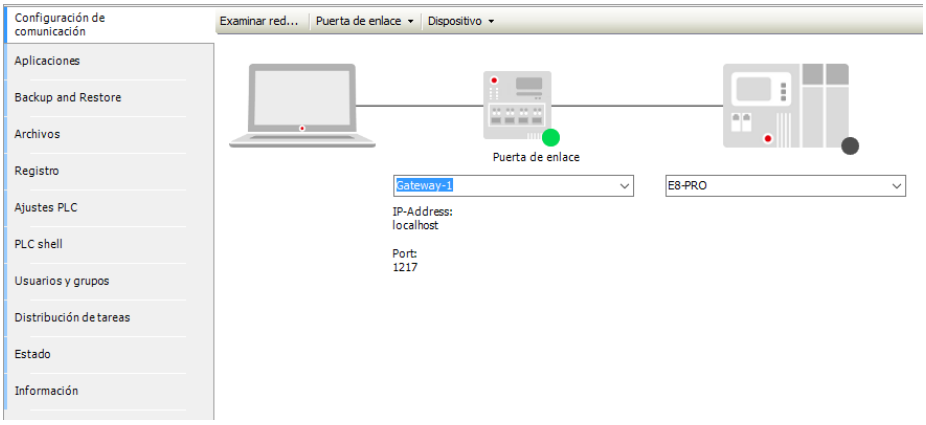
Cando instalamos CODESYS e o módulo Control Win, temos no PC un módulo chamado CODESYS Gateway V3 que é a pasarela de comunicacións que se instala e un PLC simulado (CODESYS Control Win V3).

Poñemos en marcha este último (normalmente a pasarela instálase como un servicio de Windows e estase a executar desde que arrancamos o ordenador).

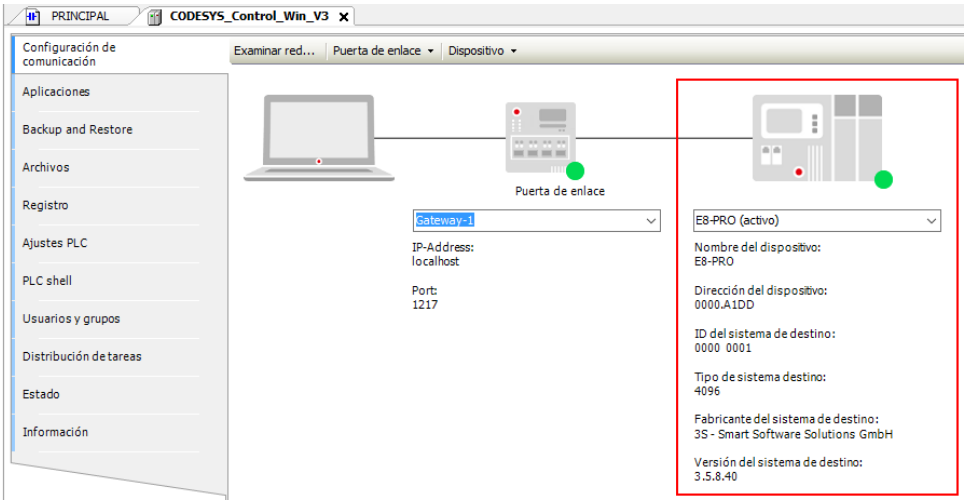
Arrancamos o PLC, facendo click sobre o mesmo (parte inferior dereita de Windows) e escollemos **Start PLC**.



Agora facemos doble click na árbore do proxecto sobre **CODESYS_Control_Win_V3** e aparece a ventá de comunicación.

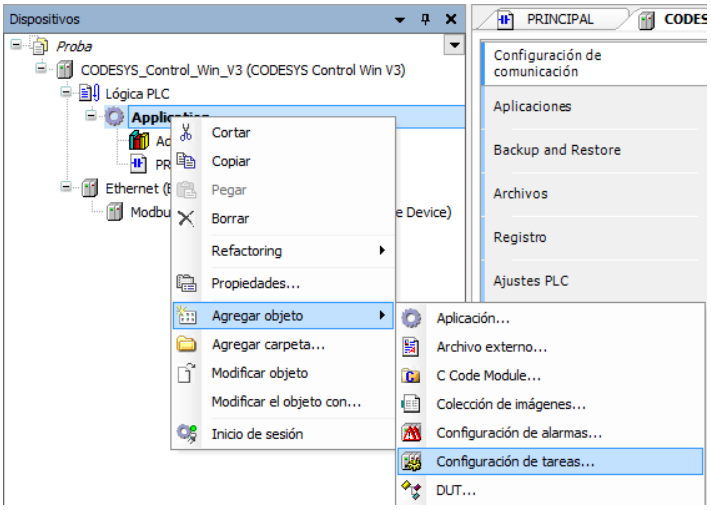


Escollemos, na parte dereita o PLC (normalmente aparecerá o nome do ordenador no que estamos a traballar), e se nos coloca como ruta activa.

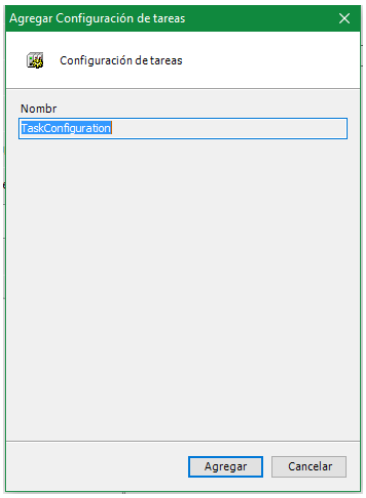


Antes de volcar o programa no PLC, temos que especificar que POU será o que se execute cando se poña en RUN. Como só temos un POU (PRINCIPAL), escolleremos este.

Para eso facemos click sobre **Application** co botón dereito e escollemos **Agregar objeto** e **Configuración de tareas...**

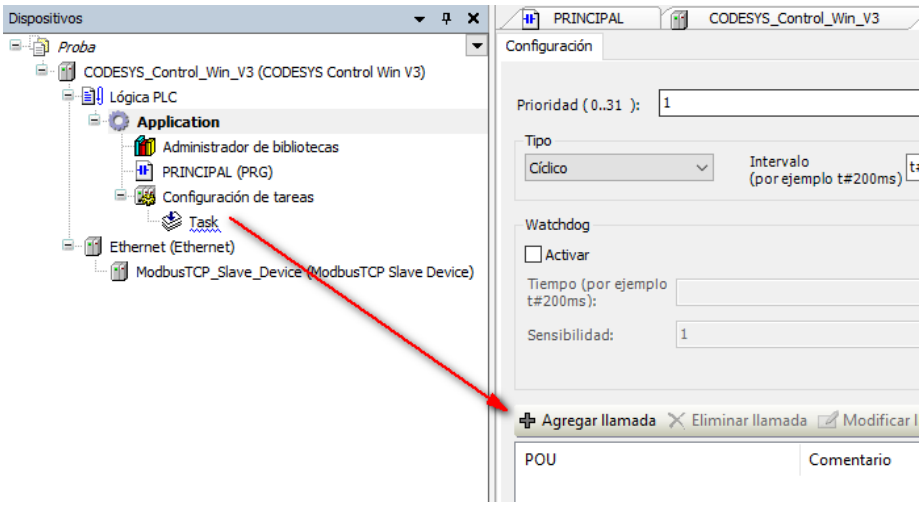


Aceptamos o nome por defecto ou cambiamos.

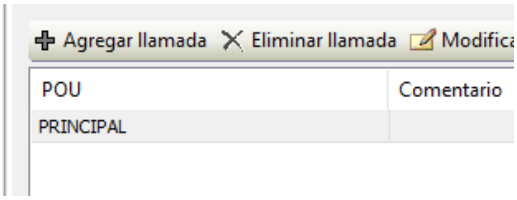


E, a continuación, asignamos o POU PRINCIPAL.

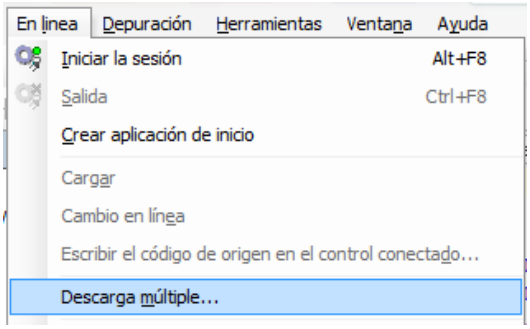
Pulsamos o botón **Agregar llamada**

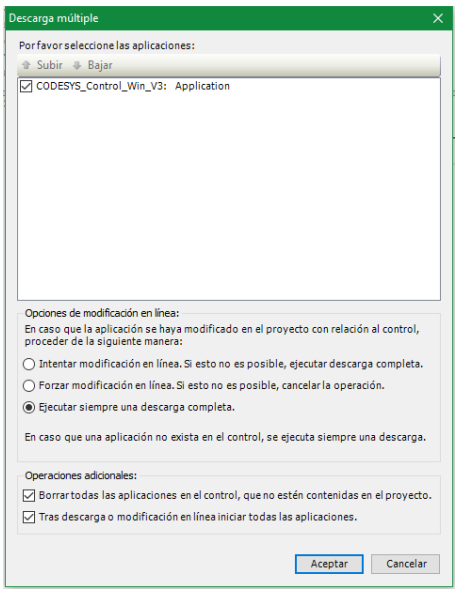


E escollemos PRINCIPAL

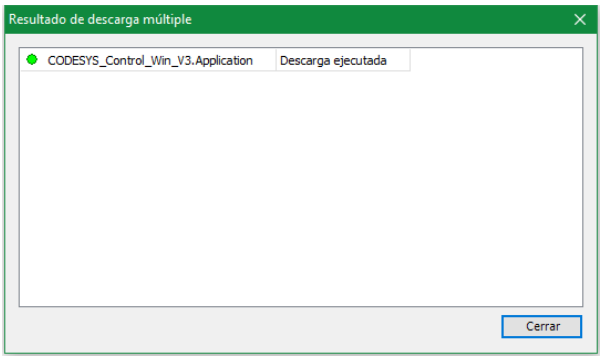


Agora podemos compilar, e volcar o programa ao PLC. Para facelo, escollemos a opción do menú **En línea**, **Descarga múltiple...**



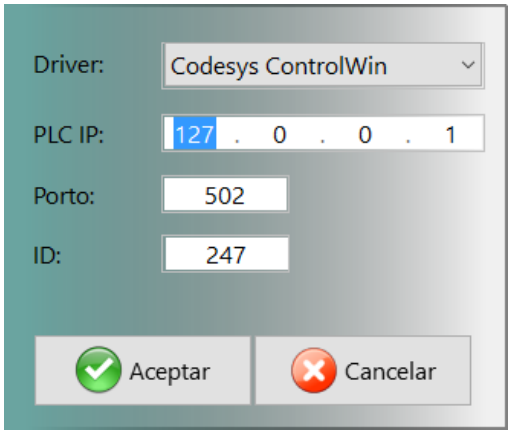


Se aparece unha ventá de resultado de descarga múltiple cun punto verde, quere dicir que o volcado se fixo correctamente.

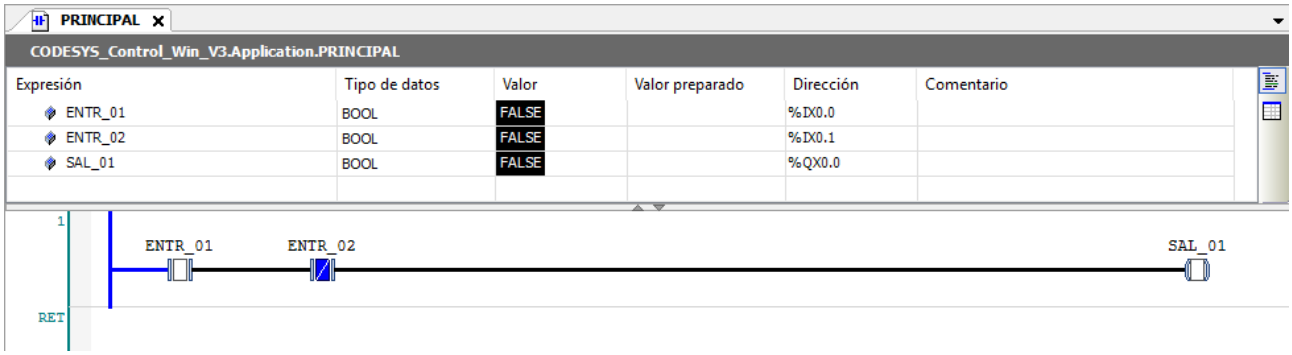


Agora xa podemos probar a conexión con virtualmakTCP.

Abrimos este e conectamos. Fixémonos que a IP será sempre a mesma (127.0.0.1). Os demais datos xa se comentaron nun punto anterior.



Podemos poñernos en liña no CODESYS coa opción do menú **En línea**, **Iniciar la sesión**.



Arrancamos unha maqueta no virtualmakTCP e comprobamos como interactuamos co mesmo.

16 Traballo cos autómatas de Schneider

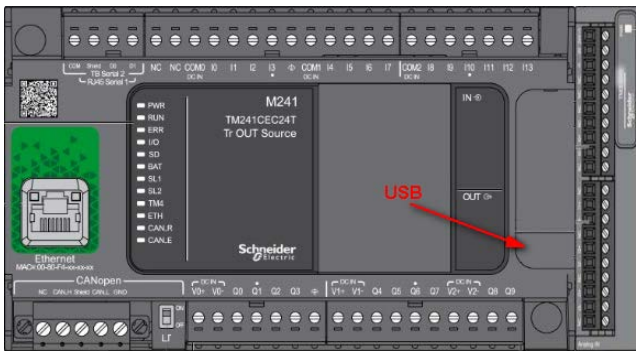
Os PLCs de Schneider veñen de fábrica cunha dirección IP na que os dous primeiros bytes son 10.10 e os dous seguintes dependen da dirección MAC.

16.1 Métodos de acceso

Estes PLCs veñen cunha conexión USB e un cable estándar polo que para conectarnos podemos empregar dous métodos: empregar dito cable ou ter en conta a IP que traen por defecto e colocar a tarxeta de rede do PC na mesma subrede.

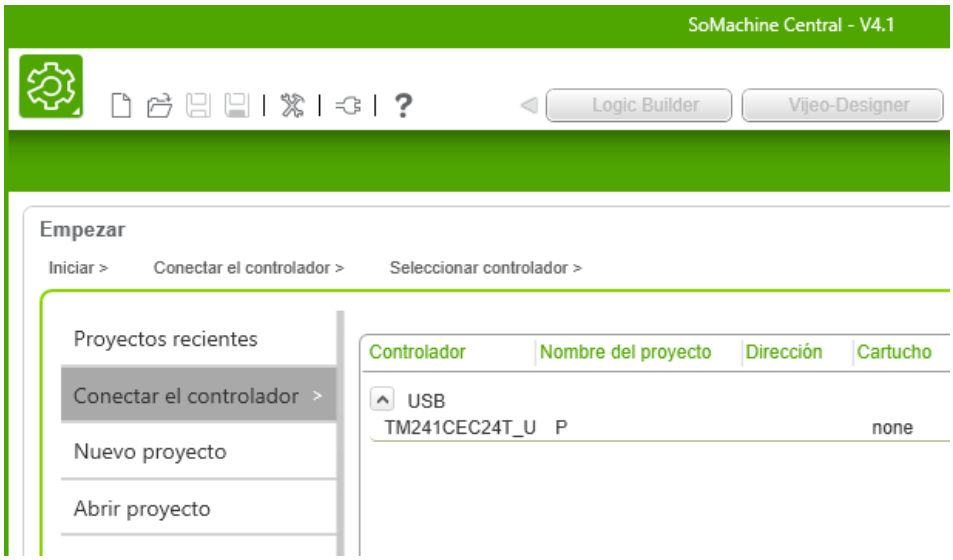
Supoñamos que conectamos por USB.

Conectamos o cable USB.



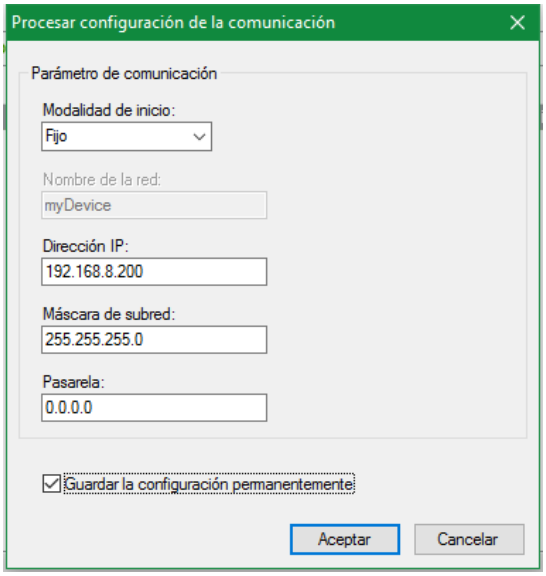
Abrimos SoMachine e escollemos a opción **Conectar el controlador**.

Aparece o TM241CEC24T_U que é o modelo conectado



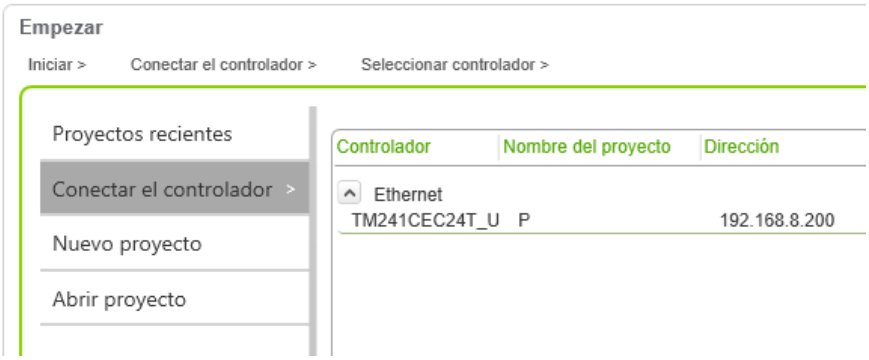
Se facemos click co botón dereito sobre o mesmo, aparece un menú que nos permite cambiar a IP escollendo a opción **Procesar configuración de la comunicación**...

Asignámoslle a IP, a máscara de subrede e aceptamos.



Aínda que podemos seguir a traballar por USB, como xa temos o PLC na subrede, agora desconectamos o cable e conectamos por TCP/IP.

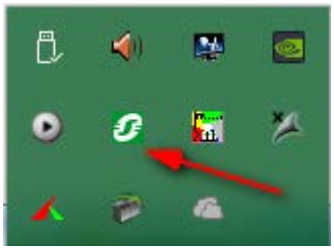
Agora comprobamos que aparece a conexión Ethernet e o mesmo PLC.



16.1.1 Posibles problemas na conexión.

En algún caso, non aparece o PLC cando pretendemos conectarnos a el. En ese caso non estará de máis facer as comprobacións seguintes:

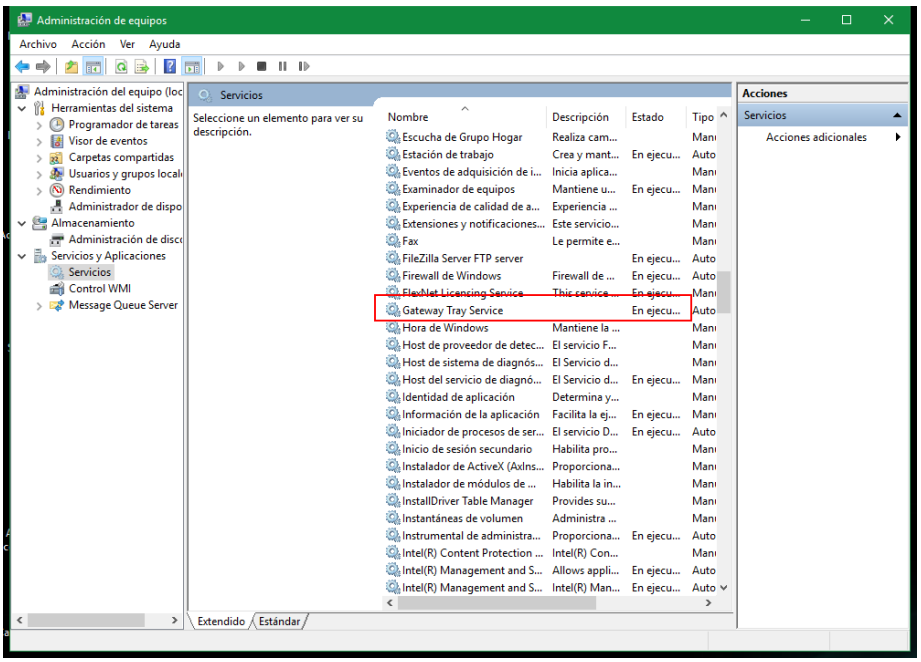
Na barra de tareras de Windows hai un icono da **Aplicación Gateway Tray** que é a pasarela que emprega SoMachine para comunicarse.



Haberá que comprobar que está iniciado o servicio de pasarela para que funcione correctamente a comunicación.

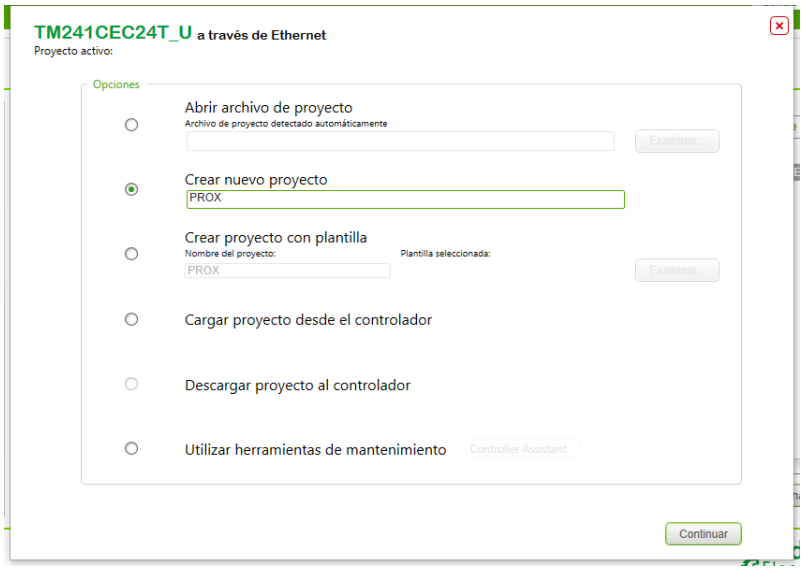
En caso de que apareza o icono anterior desactivado, quere decir que o servicio está inactivo (problema detectado en Windows 10).

Nese caso teremos que ir aos servicios de Windows e activar o **Gateway Tray Service**.

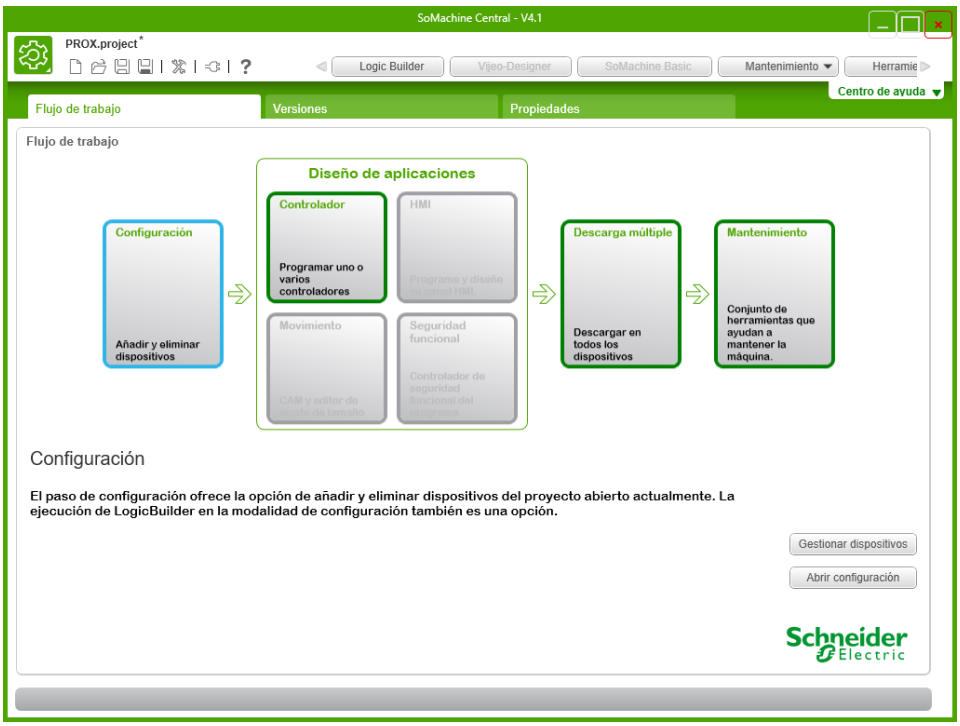


16.2 Entorno de programación

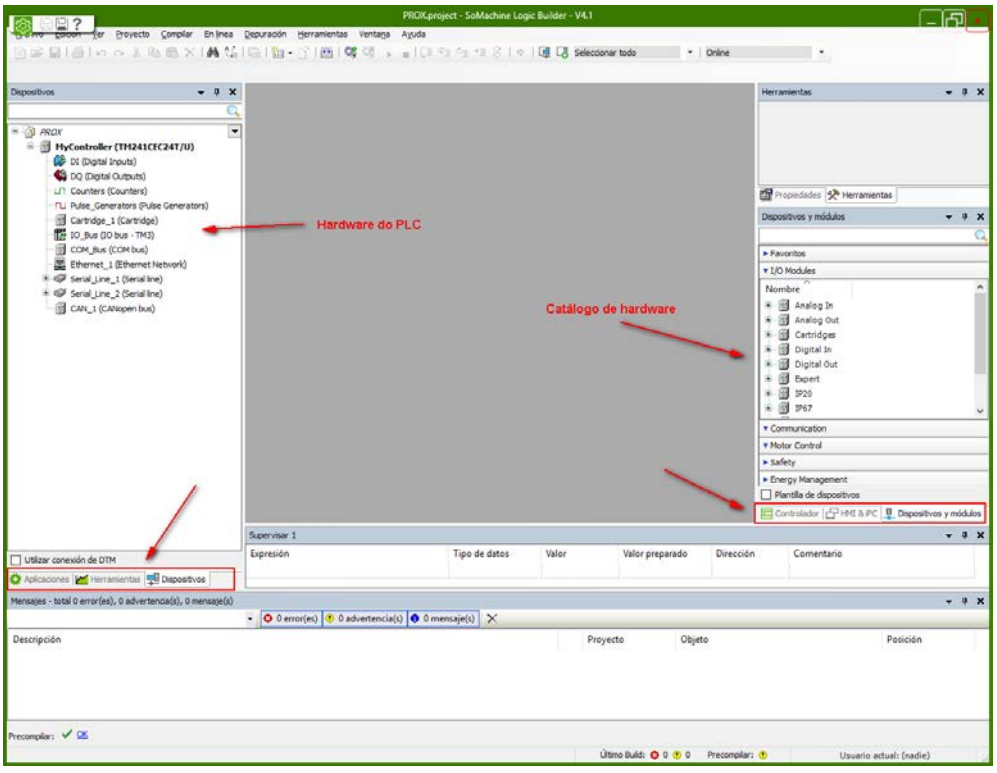
Unha vez que aparece o PLC conectámonos e creamos un novo proxecto.



Na seguinte pantalla pulsamos o botón **Abrir configuración**.



Entramos no entorno de programación no que temos na parte esquerda a árbore do proxecto e na parte dereita o catálogo de hardware.

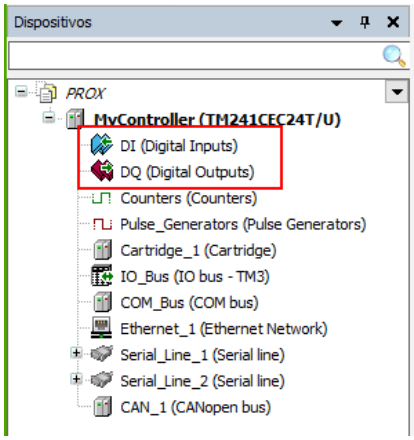


Na parte inferior esquerda temos tres pestañas: **Aplicaciones, Herramientas** e **Dispositivos**. Nestas pestañas clicaremos para cambiar entre a vista de hardware e o programa do PLC.

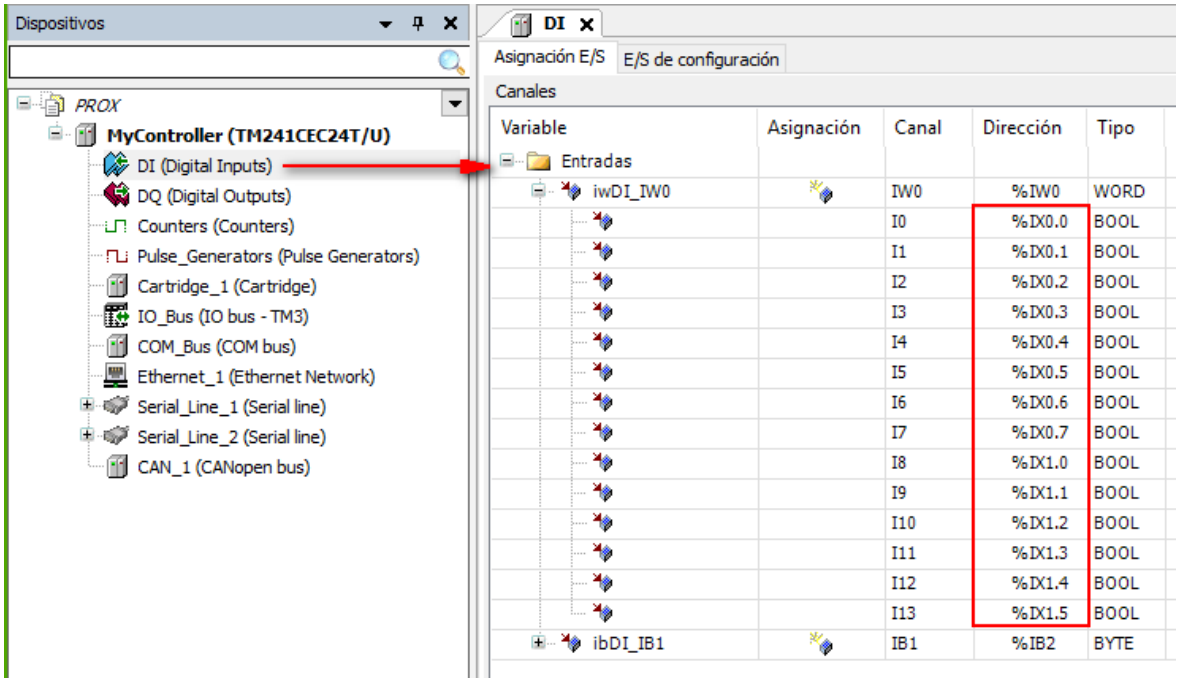
Na parte inferior dereita tamén hai tres pestañas: **Controlador, HMI** e **Dispositivos y módulos**. Nestas pestañas podemos seleccionar os módulos de ampliación de sinais analóxicas/dixitais entre outros.

16.2.1 Configuración de hardware

Dependendo do modelo de PLC teremos unha configuración de hardware determinada. No exemplo da figura hai un módulo de entradas dixitais e un módulo de saídas do mesmo tipo (incorporadas na CPU).

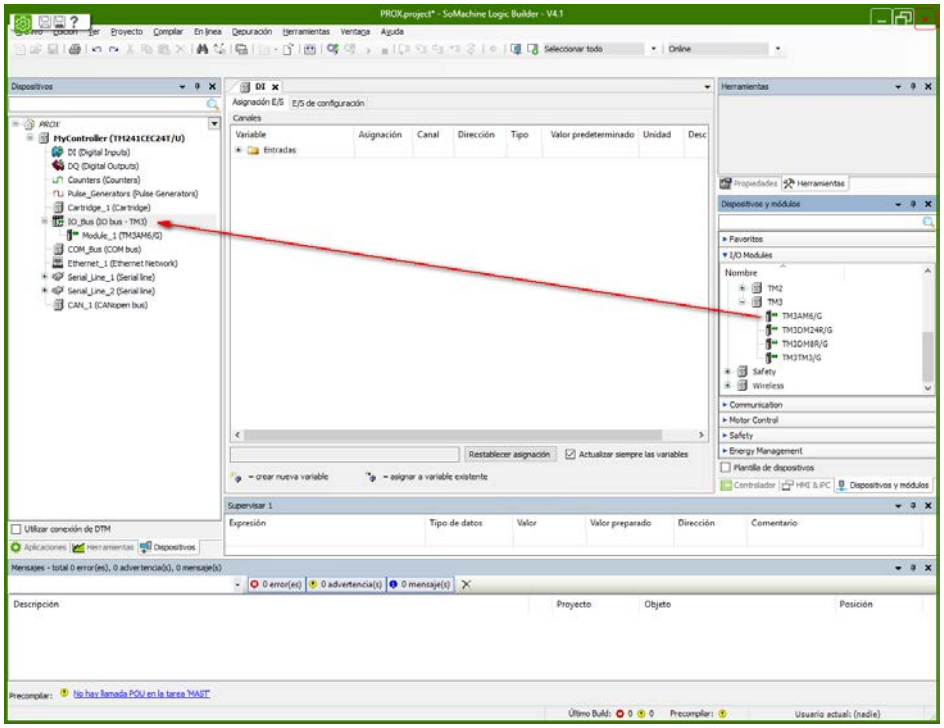


Se facemos doble click sobre unha delas, na parte central aparece o mapeado das mesmas. Na figura seguinte observamos a nomenclatura da norma IEC61131, polo que o primeiro bit de entradas é o **%IX0.0**.

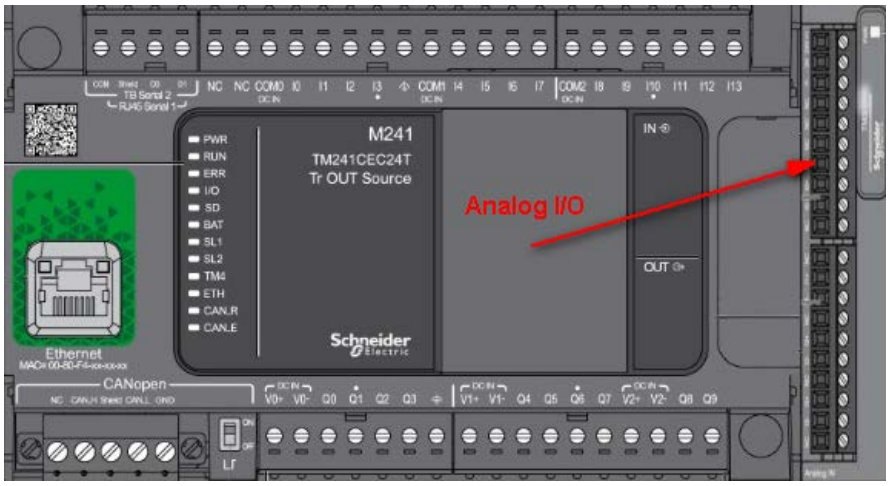


Como imos traballar con **virtualmakTCP** e podemos simular entradas e saídas analóxicas, engadirémoslle un módulo. En concreto un TM3AM6/G.

Para eso, localizamos o módulo na parte dereita e arrastrámolo ata a parte esquerda.

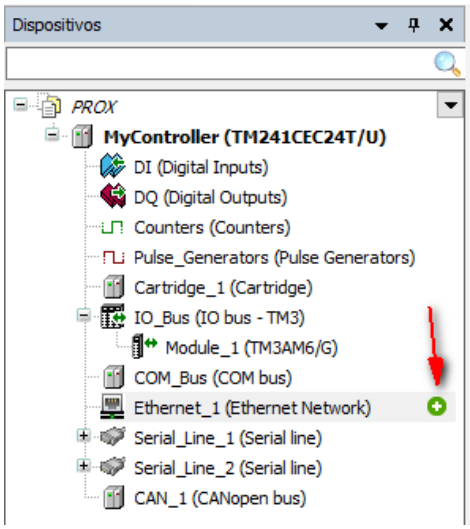


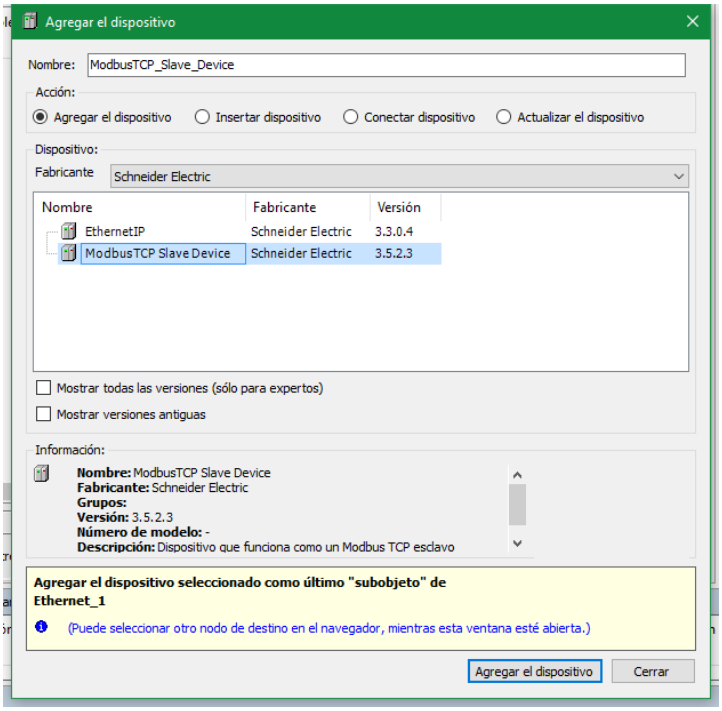
Este módulo hai que telo conectado á dereita da CPU e alimentado a 24V para non ter problemas ao volcar o proxecto desde SoMachine.



En realiadade esto sería suficiente para crear un proxecto e descargalo no PLC, pero unha vez máis, se imos traballar con **virtualmakTCP** é necesario engadir entradas virtuais e non empregar as físicas que trae o PLC.

Para eso engadimos un módulo **Modbus TCP Slave Device** debaixo de **Ethernet_1**. Pulsamos sobre o botón verde que aparece á dereita e seleccionamos o módulo mencionado.

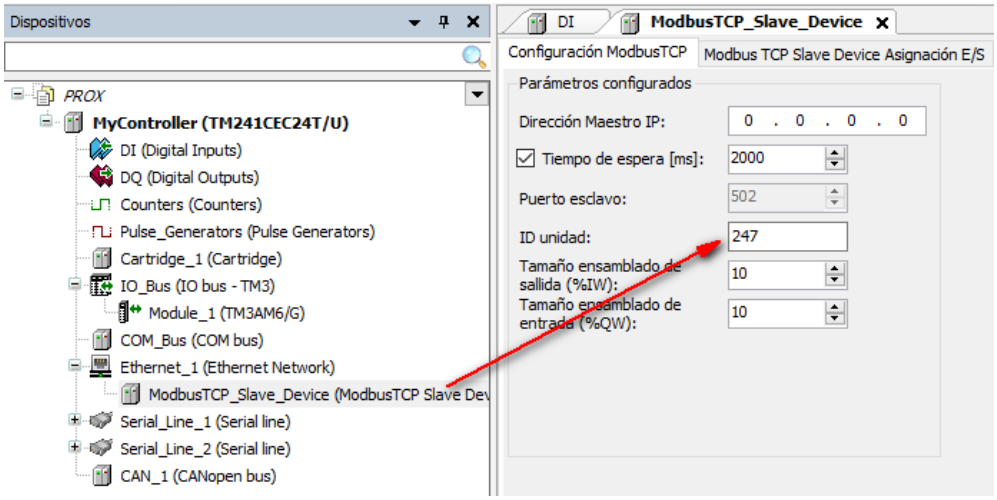


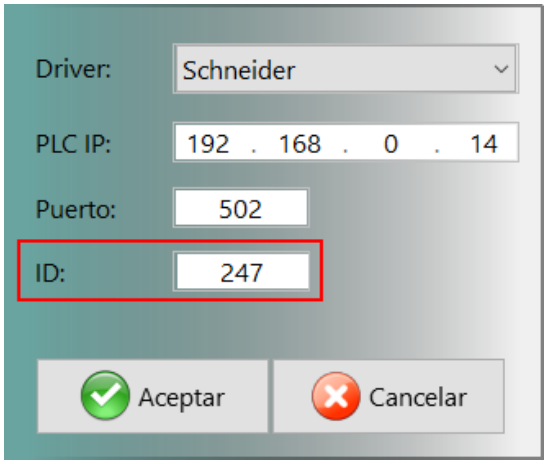


Agregamos e pechamos a ventana.

Se facemos doble click sobre o módulo, aparecen unhas pestañas na que temos datos interesantes.

Nunha primeira está a ID de unidade (por defecto 247) que temos que respetar no **virtualmakTCP**.



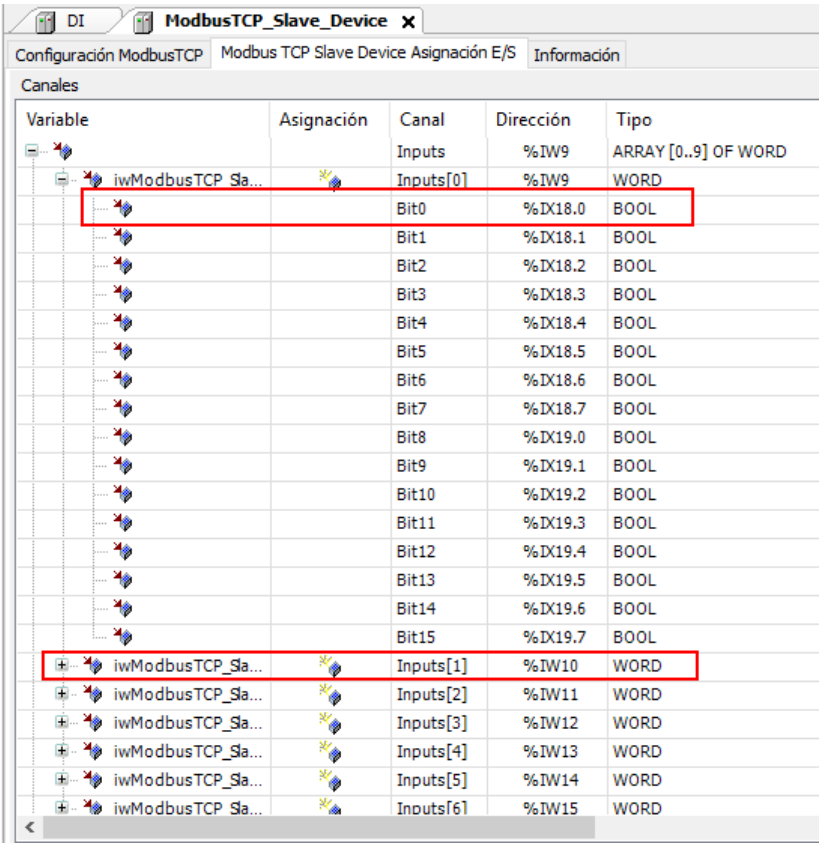


Na segunda pestaña atópase o direccionamento de entradas/saídas do módulo Modbus. A nós só nos interesan as entradas, porque para as saídas empregaremos as que veñen integradas na CPU.

Hai que ter coidado co direccionamento e utilizar a nomenclatura da norma mencionada anteriormente.

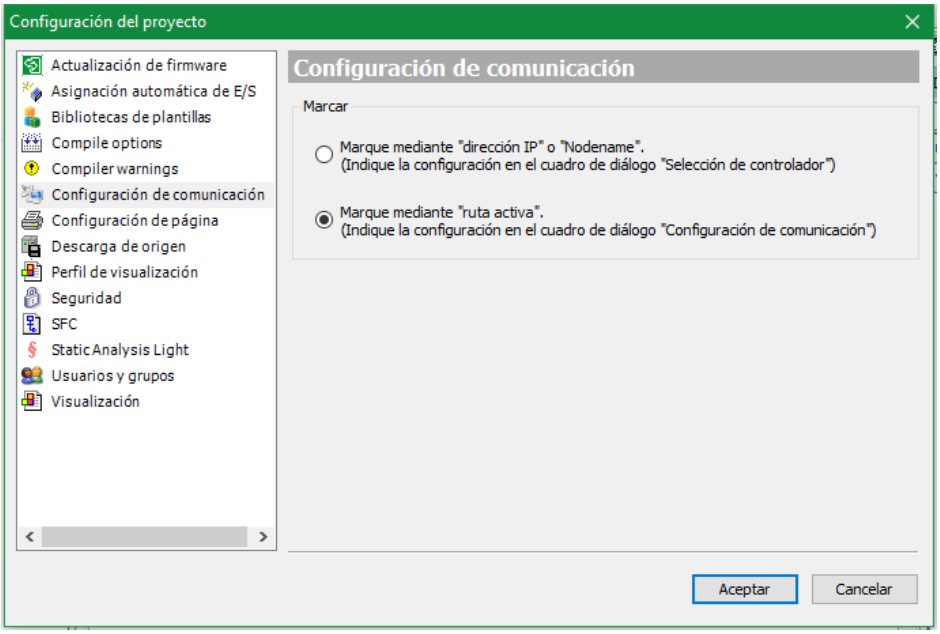
Así vemos que o primeiro bit das entradas dixitais sería **%IX18.0** en cambio, se imos tratar a posición de memoria en formato palabra, sería a **%IW9**, tal como vemos na seguinte imaxe.

Os programas que fagamos pensando en empregar o simulador, levarán ese direccionamento.

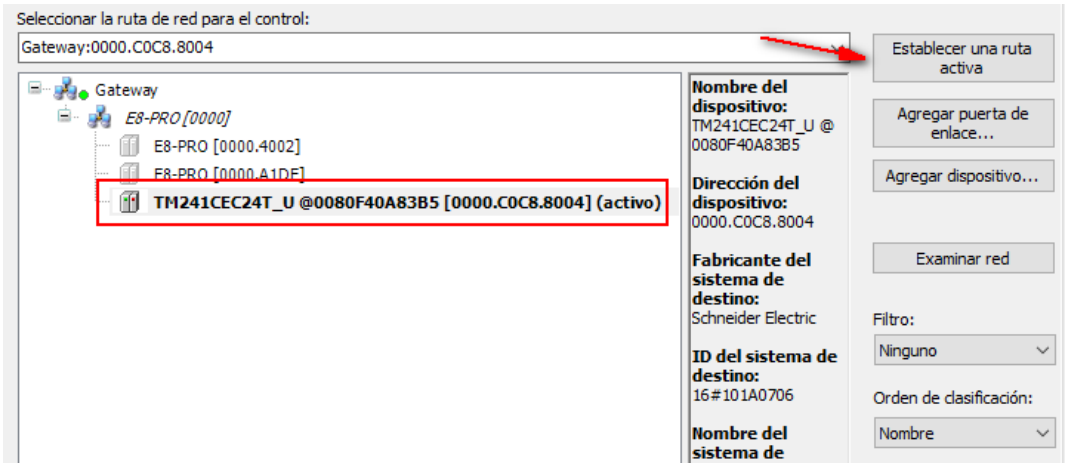


Para comprobar se temos comunicación efectiva co PLC agora podemos establecer o que se chama “ruta activa”.

Para eso imos a **Proyecto->Configuración del proyecto->Configuración de comunicación** e escollemos a opción “ruta activa”.

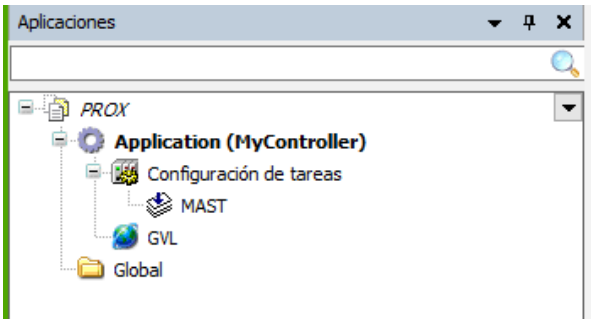


A parte central cambia e, despois de quitar a opción Filtro (poñer Ninguno) e pulsar o botón **Examinar red** ten que aparecer o PLC conectado. Marcámolo e pulsamos **Establecer una ruta activa**.



16.3 Creando un programa en Grafcet

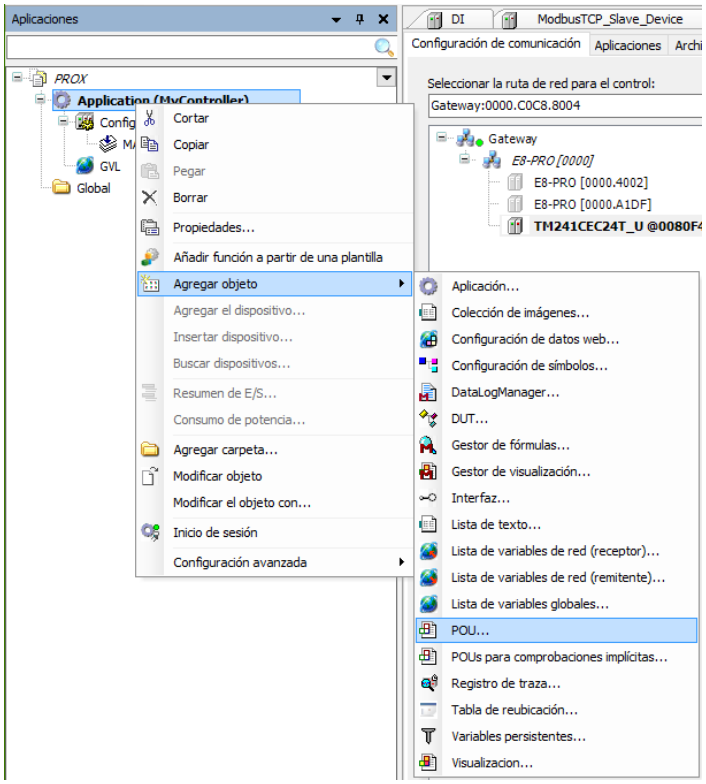
Nas pestañas da parte esquerda inferior, escollemos a de **Aplicaciones**.



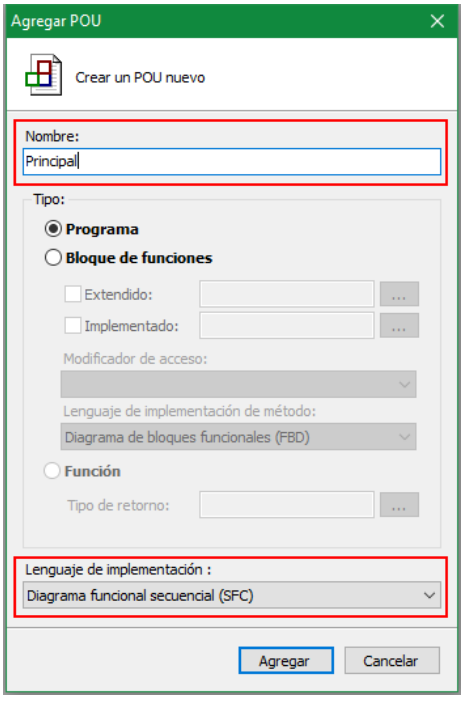
Vemos unha tarefa principal (MAST) que será a que se execute cíclicamente.

O primeiro que faremos será crear un novo POU, chamarémoslle **Principal** e escolleremos linguaxe SFC (Grafcet).

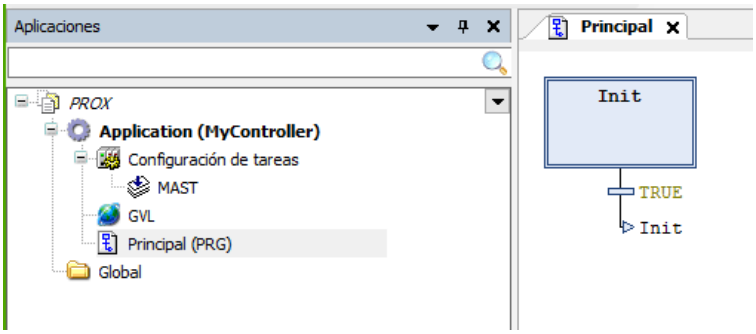
Para facelo, clicamos co botón dereito sobre **Application** e escollemos **Agregar objeto->POU...**



É importante cambiarlle o nome que propón por defecto (POU) senón dará problemas ao compilar.

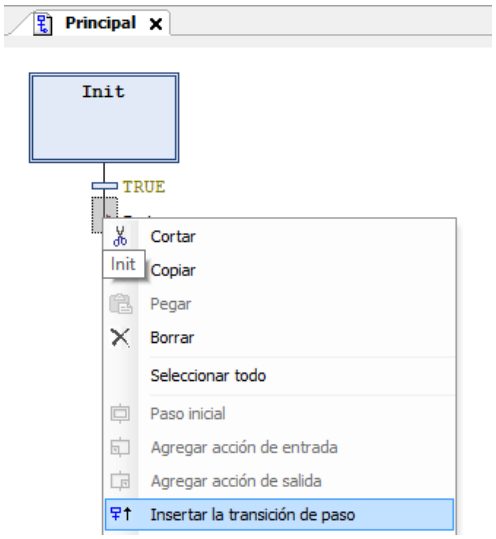


Aparece na parte central a interface para implementar o Grafcet.

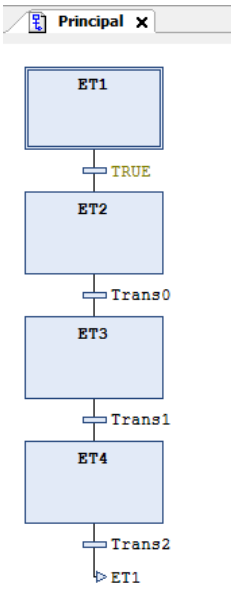


Imos facer unha secuencia simple onde sae primeiro un actuador (A+), logo outro (B+) e despois recóllense os dous (A-,B-).

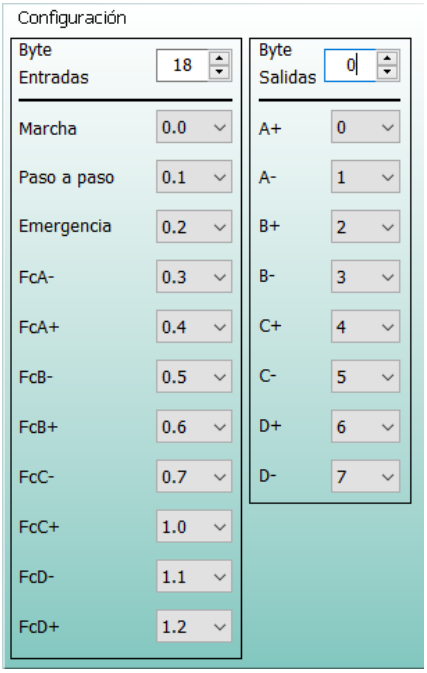
Colocamos as etapas e transicións pulsando o botón dereito ao final do Grafcet e escollendo **Insertar la transición de paso**.



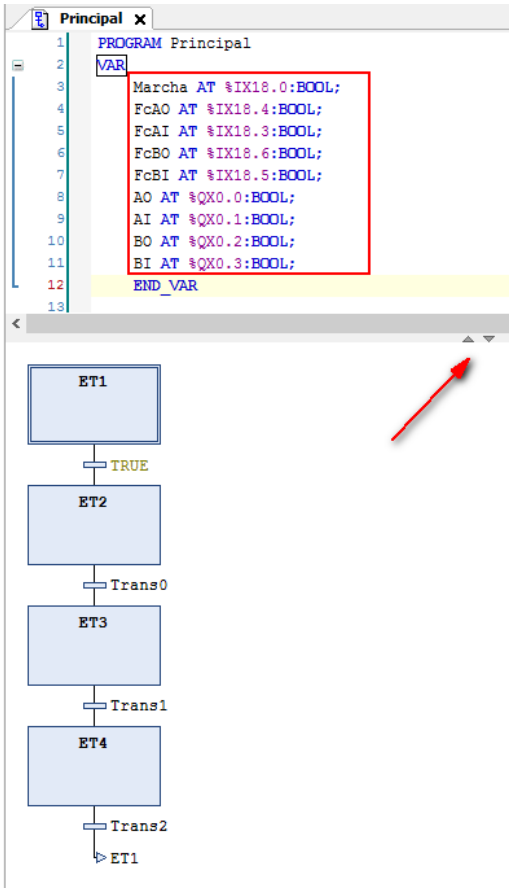
Renomeamos as etapas.



Cando se traballa con Grafcet é interesante definir as variables que imos empregar. Neste caso como imos traballar con **virtualmakTCP** temos a seguinte táboa.

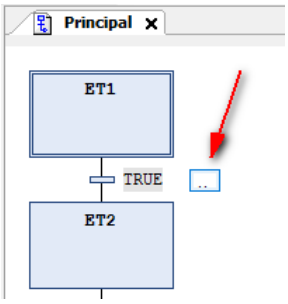


Logo definimos as variables no SoMachine. Para definilas, abrimos a parte superior do Grafcet pulsando sobre o trianguliño que se atopa no centro da pantalla.

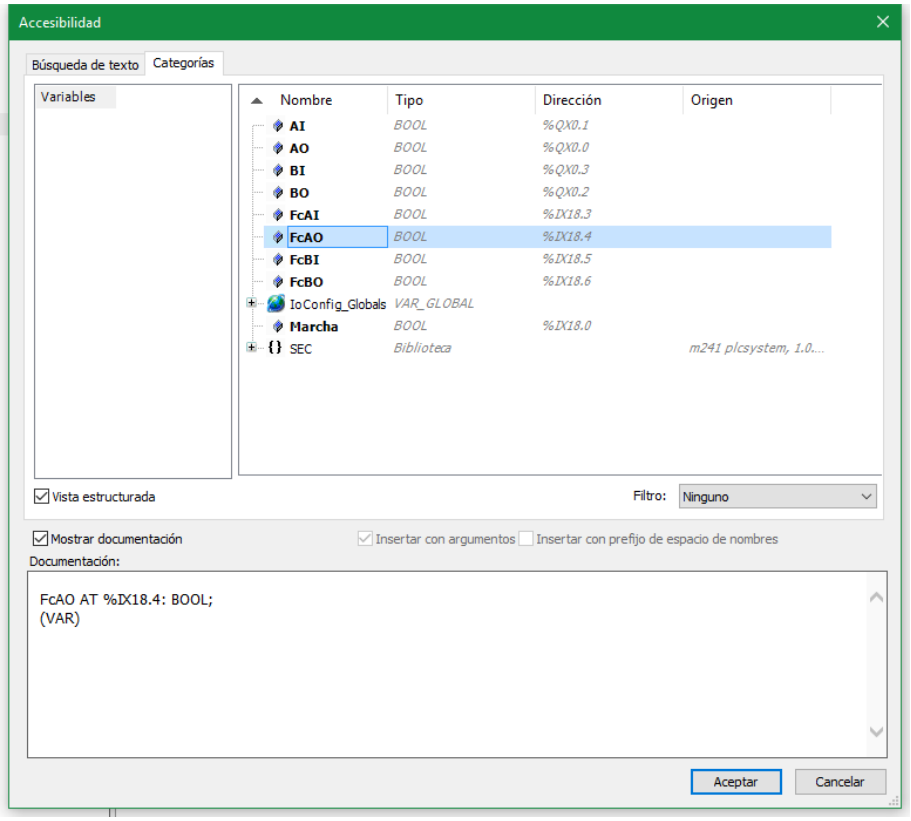


Agora podemos ir colocando transicións e accións nas etapas.

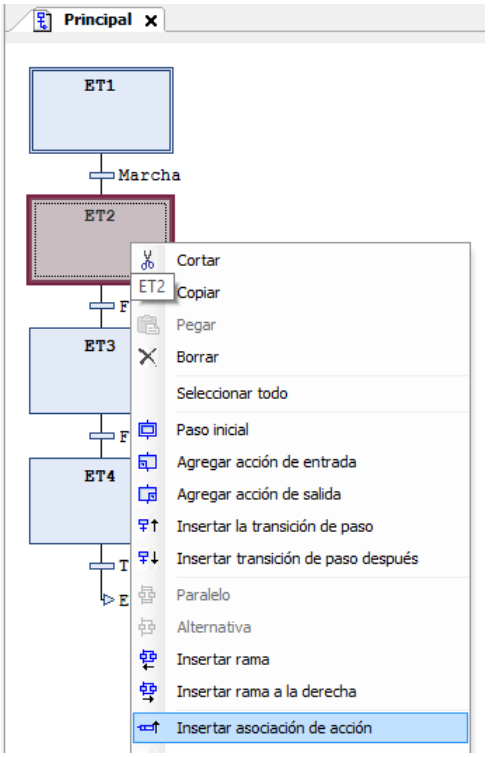
Para colocar unha transición simple, pulsamos sobre a mesma e logo pulsamos o botón con dous puntos que aparece á dereita.



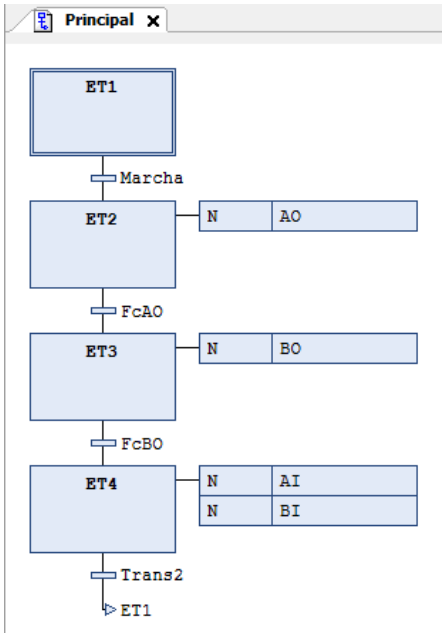
Na ventana de variables imos collendo as que necesitemos.



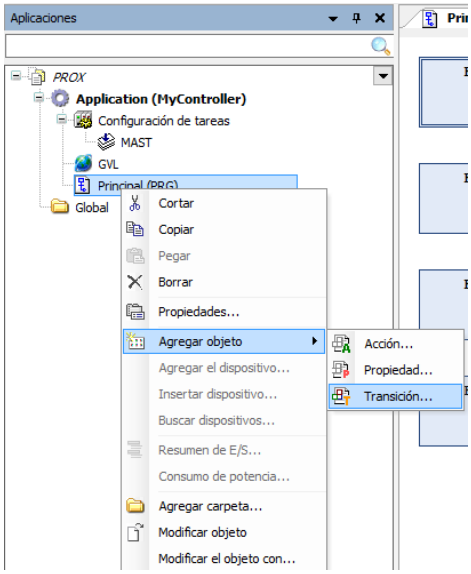
Para colocar as accións dentro dunha etapa pulsamos o botón dereito na mesma e escollemos **Insertar asociación de acción**.



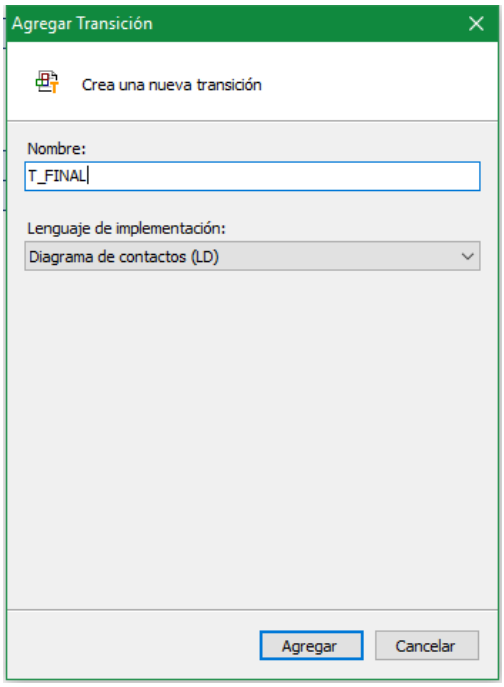
Podemos insertar as accións que necesitemos nunha etapa (N significa que a acción terá lugar mentres estea activa a etapa).



Agora queda a última transición, que está composta de dous contactos en serie. Para programala temos que poñernos enriba de **Principal** e co botón dereito escoller **Agregar objeto->Transición**.

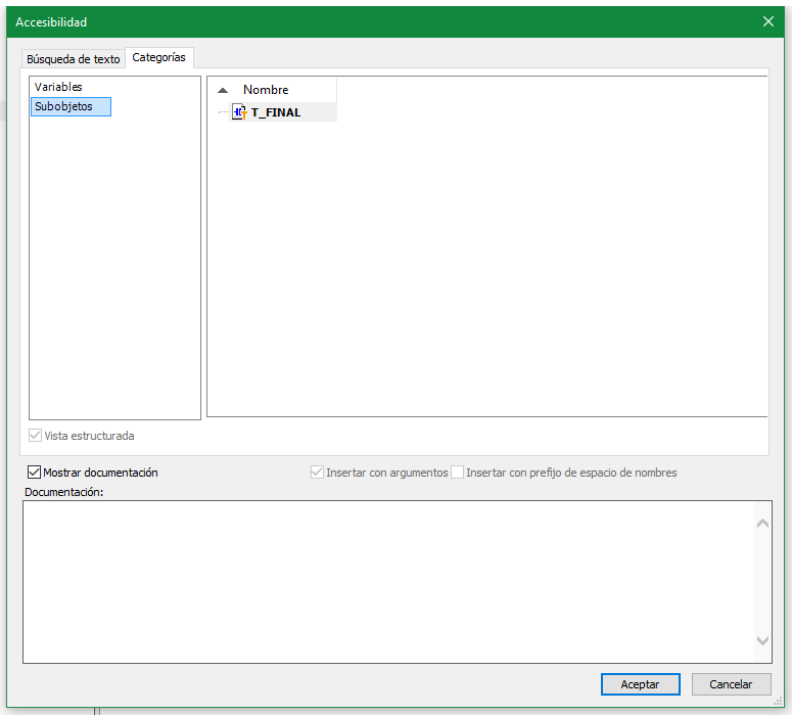


Dámoslle un nome e escollemos a linguaxe de programación a empregar.

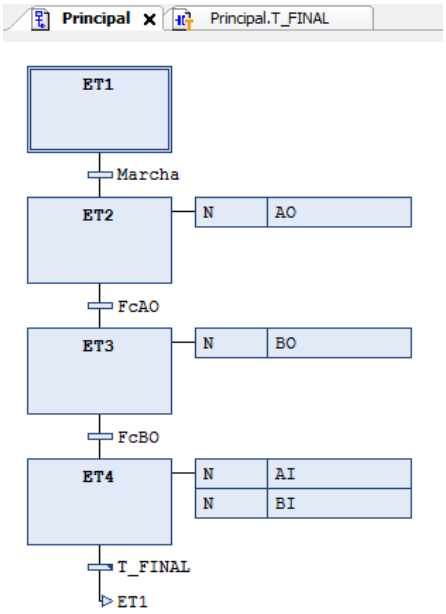


Programamos os contactos e unha bobina que representa que se cumpre a transición.

Agora imos de novo ao Grafcet e, na posición da transición, eleximos a mesma.



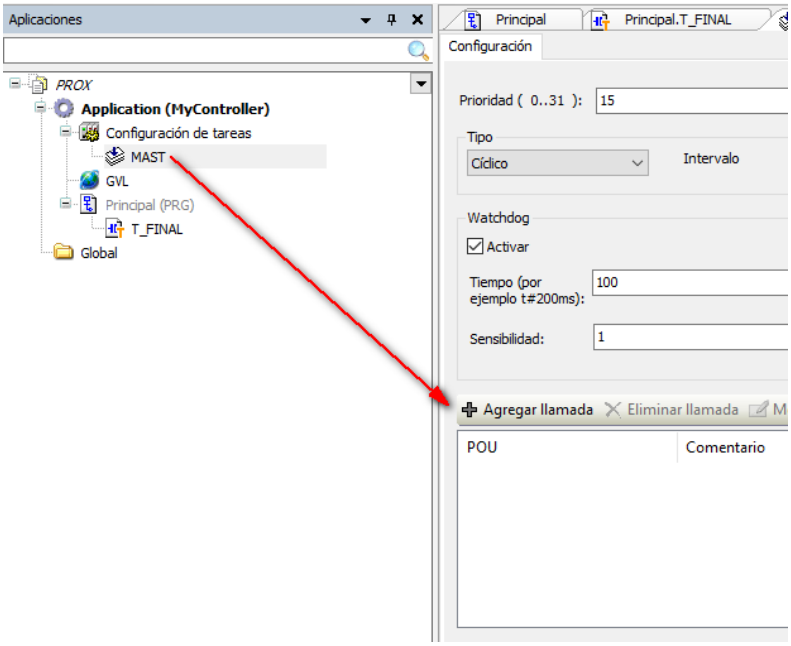
Quedaría o Grafcet completo, desrte xeito.

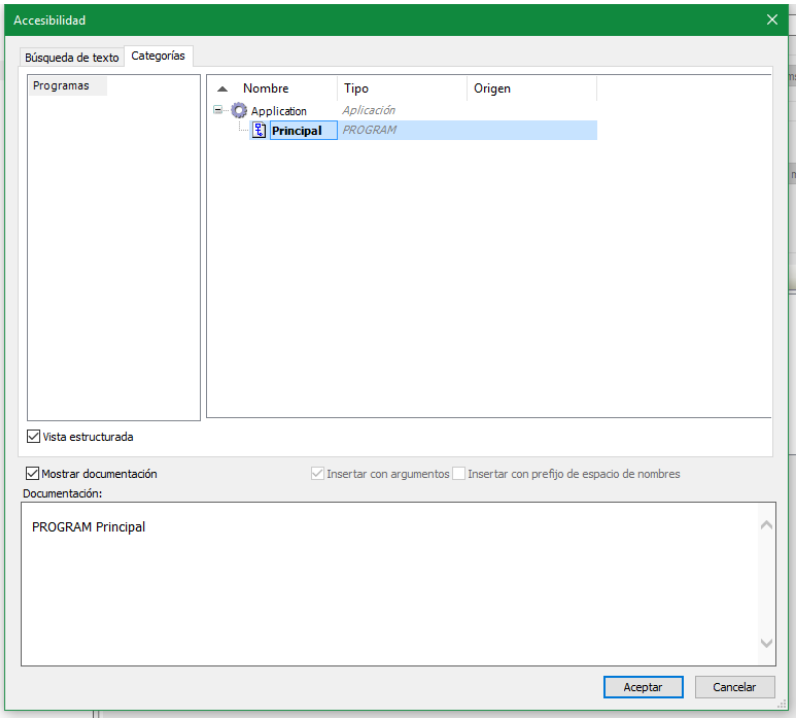


16.4 Executando o programa

Unha vez feito o Grafcet temos que indicarlle á tarefa principal (MAST) que execute este programa.

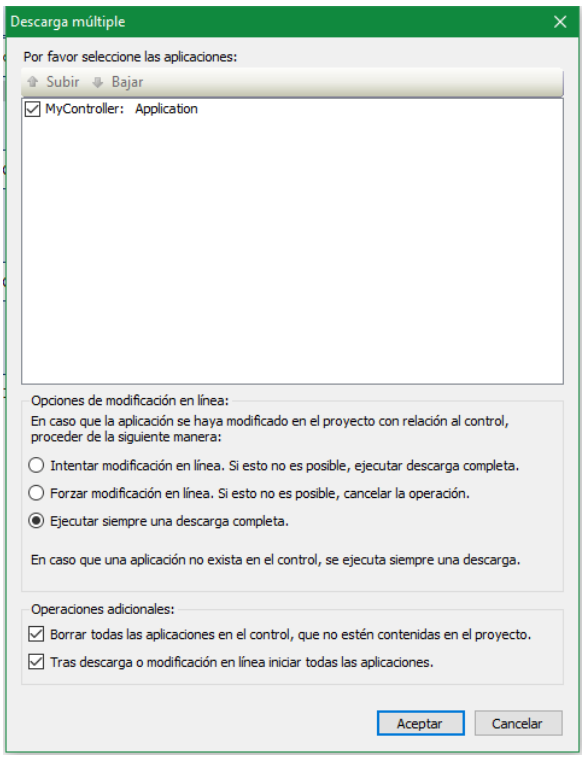
Facemos doble click sobre MAST e no apartado **Agregar llamada** escollemos este programa.





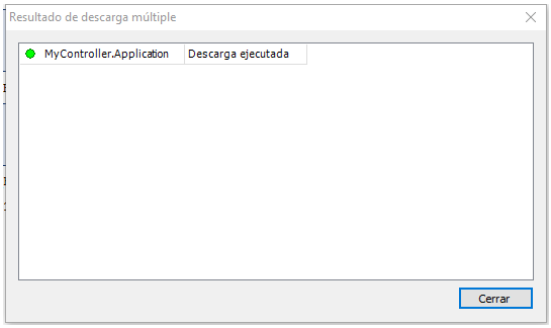
Agora xa podemos volcar o programa no PLC.

Para volcalo escollemos a opción do menú **En línea->Descarga múltiple**.

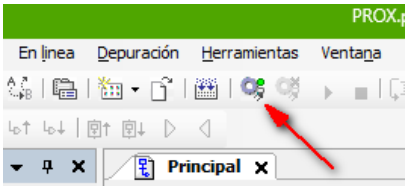


Teremos que pulsar varias veces a combinación de teclas **Alt+F** para aceptar.

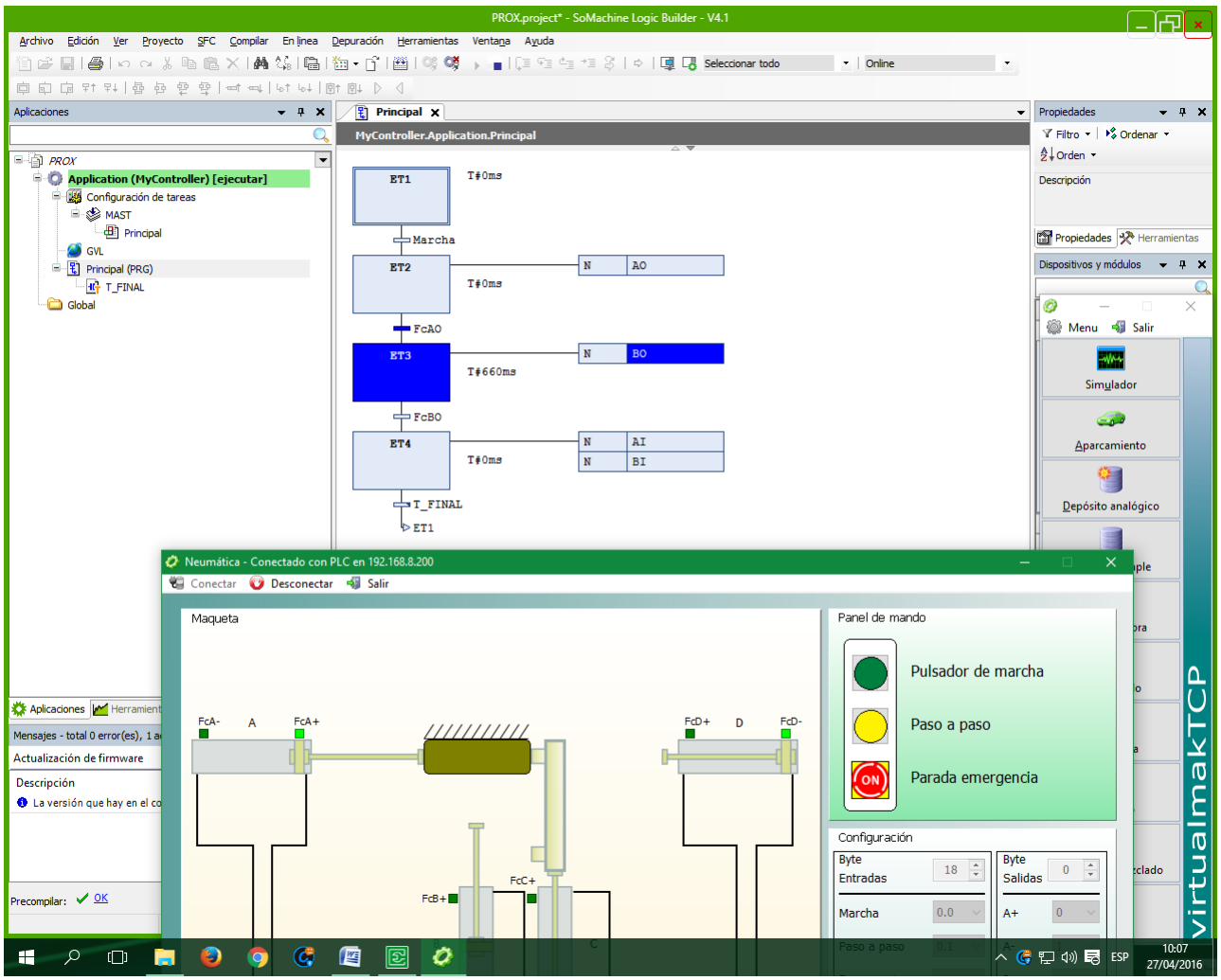
Unha vez efectuada a descarga se nos confirma mediante o cadro seguinte.



Podemos comprobar o funcionamento do programa mediante o icono de **Iniciar sesión**.



Agora arrancamos **virtualmakTCP** e comprobamos o funcionamento.


PROX.project* - SoMachine Logic Builder - V4.1
Archivo Edición Ver Proyecto SFC Compilar En línea Depuración Herramientas Ventana Ayuda
Aplicaciones
PROX
Application (MyController) [ejecutar]
Configuración de tareas
MAST
Principal
GVL
Principal (PRG)
T_FINAL
Global
Principal
MyController.Application.Principal
ET1
T#0ms
Marcha
ET2
T#0ms
N AO
FcAO
ET3
T#660ms
N BO
FcBO
ET4
T#0ms
N AI
N BI
T_FINAL
ET1
Propiedades
Filtro
Ordenar
Orden
Descripción
Propiedades
Herramientas
Dispositivos y módulos
Menu
Salir
Simulador
Aparcamiento
Depósito analógico
Neumática - Conectado con PLC en 192.168.8.200
Conectar
Desconectar
Salir
Maqueta
FcA- A FcA+
FcD+ D FcD-
FcB+
FcC+
Panel de mando
Pulsador de marcha
Paso a paso
Parada emergencia
ON
Configuración
Byte Entradas
18
Byte Salidas
0
Marcha
0.0
A+
0
Paso a paso
Aplicaciones
Mensajes - total 0 error(es), 1 a
Actualización de firmware
Descripción
Precompilar:
OK
virtualmakTCP
ESP
10:07
27/04/2016