Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

# Librería de Funciones para el Control en Posición de un Pórtico

Autor: Adrián Rocha Íñigo
Tutor: Luis Fernando Castaño Castaño

**Dep. de Ingeniería de Sistemas y Automática**
**Escuela Técnica Superior de Ingeniería**
**Universidad de Sevilla**

Sevilla, 2018

Proyecto Fin de Carrera
Grado en Ingeniería de Tecnologías Industriales

# Librería de Funciones para el Control en Posición de un Pórtico

Autor:
Adrián Rocha Íñigo

Tutor:
Dr. Luis Fernando Castaño Castaño

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo Fin de Grado: Librería de Funciones para el Control en Posición de un Pórtico

Autor: Adrián Rocha Íñigo

Tutor: Luis Fernando Castaño Castaño

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

*A mi familia*

*A mis amigos*

*A mis maestros*

# Agradecimientos

Este trabajo de fin de grado pone fin a un camino de aprendizaje y esfuerzo, en el cual he sido acompañado de numerosas personas que me han enseñado, rectificado y motivado hasta culminar esta etapa.

Durante este tiempo, he tenido la oportunidad de aprovechar las competencias y experiencias de cada uno de los profesores que clase a clase me han brindado las herramientas necesarias para completar mi formación. A todos ellos, les muestro mi más amplio agradecimiento.

En especial, al Dr. Fernando Castaño Castaño, sin sus consejos, ayuda y paciencia este trabajo no sería posible.

Muchas gracias.

*Adrián Rocha Íñigo*

*Estudiante de Grado en Ingeniería de Tecnologías Industriales*

*Sevilla, 2018*

# Resumen

El presente proyecto propone una librería de funciones de bajo nivel para el posicionamiento y control de un pórtico instalado en una célula de fabricación.

Desde el punto de vista de la automatización de la célula de fabricación, el pórtico es una estación de trabajo que permite manipular *pallets* o piezas que son trasladadas por un sistema de transporte automatizado. Básicamente consta un sistema de posicionado de una pinza que es accionado por *servomotores*.

La librería ha sido desarrollada para ser implementada en un autómata programable, y debe servir como base para ser usada en programas de alto nivel a ejecutar por el mismo equipo.

Aparte de la librería, se han desarrollado los planos de conexionado entre los *servodrivers*, la electrónica auxiliar y las tarjetas específicas del autómata (tarjeta de contaje rápido y salidas de alta frecuencia).

Adicionalmente, se presenta la configuración necesaria de los equipos que intervienen.

De este modo, se propone una serie de recursos, a partir de los cuales se pueden desarrollar las futuras aplicaciones y trabajos de la máquina.

# Abstract

This project proposes a set of low-level functions for the positioning and management of a gantry installed in a manufacturing cell.

From the point of view of the automation of the manufacturing cell, the gantry is a workstation that allows to handle pallets or pieces that are moved by an automated transport system. In essence, it consists of a positioning system of a clamp that is driven by servomotors.

The set of functions has been developed to be implemented in a programmable logic controller. It must serve as a grounding to be used in high level programmes to be executed by the same system.

Apart from the set of functions, the connection diagrams between the servodrivers, the auxiliary electronics and the specific cards of the automaton have been developed (fast counting card and high frequency outputs).

In addition, the required configuration of the equipment involved is introduced.

By doing so, a range of resources is proposed, from which the future applications and tasks of the machine can be developed.

# Índice

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# 1 INTRODUCCIÓN

El control se puede definir como la acción de regulación, de forma manual o automática, de un sistema para que cumpla unos objetivos específicos, de tal forma que pueda mantener una consigna dada aunque existan perturbaciones que lo dificulten. Por sistema se entiende el conjunto de elementos que relacionados entre sí contribuyen a un determinado objeto.

Según lo anterior, se deduce que el control en posición es aquel que permite la situación física con precisión de las partes móviles de los sistemas mecánicos. Está presente en la mayoría de las máquinas automatizadas tales como máquinas-herramienta, puestos en células de fabricación autónomas, máquinas eléctricas rotativas, impresoras 3D y todo tipo de robots.

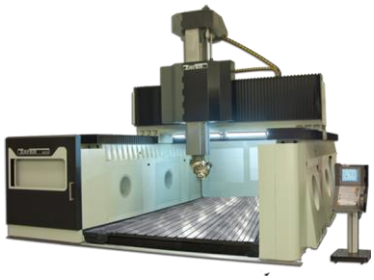


Figura 1-1. Máquina-herramienta. Fresadora Zayer-neos. [1]

Figura 1-2. Robot colaborativo. FANUC CR-7iA. [2]

Figura 1-3. Impresora 3D. Prusa i3 MK3. [3]

Para resolver este problema de control se hace uso, en general, de tres elementos básicos:

- **Actuadores**: Son necesarios para generar las fuerzas y pares que mueven la estructura, pueden ser electromecánicos, como por ejemplo servomotores, motores paso a paso, asíncronos o de corriente continua, o bien, sistemas hidráulicos.
- **Sensores**: Son dispositivos con la capacidad de medir desplazamientos lineales y giros, y representar la medida de forma adecuada para poder ser procesada. Estos mayoritariamente son *encoders* absolutos o incrementales, *resolvers* o potenciómetros.
- **Sistemas de control**: Permiten gestionar, controlar y supervisar a los anteriores, de tal forma que, a partir de los valores de las medidas de los sensores, determina en cada momento las señales de control sobre los actuadores para cumplir con el objetivo de control.

Como se puede apreciar, estos tres grupos corresponden a la estructura común de un sistema de control realimentado donde, el controlador, captando la magnitud física por medio de los sensores, influye sobre el medio mediante los actuadores para regular la variable bajo control. En la Figura 1-4 se puede apreciar el modo en el cual se interconectan los diferentes bloques.

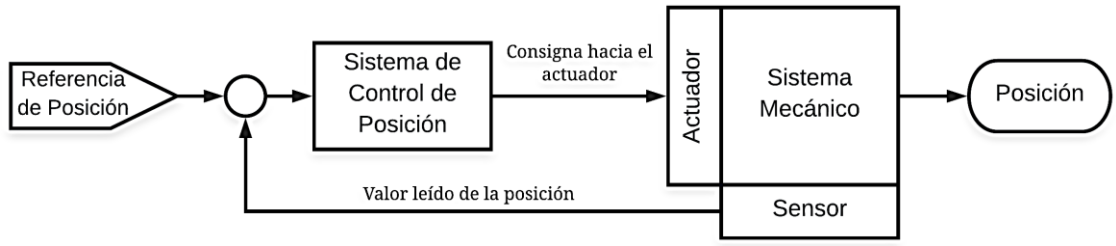



Figura 1-4. Sistema de control en posición de lazo cerrado.

No obstante, este esquema puede ser modificado eliminando la realimentación de la posición. Esto hace que el control a implementar sea más sencillo, pero como desventajas se obtiene un sistema menos robusto, menos tolerante a fallos y más sensible a perturbaciones, por lo que no es recomendado.

Para obtener un buen funcionamiento es de gran importancia la compatibilidad entre los diferentes elementos, es decir, la capacidad de poder conectarse entre sí y que las comunicaciones y el flujo de energía se desarrollen correctamente.

Si lo anterior no se cumple, es necesario incluir más dispositivos que permitan la adaptación de las señales de datos usando interfases electrónicas, y además, la adaptación de la alimentación eléctrica haciendo uso de diferentes tipos de convertidores.

Por último, tanto el desarrollo de un buen algoritmo de control, como la sintonización de este a partir del modelado del sistema mecánico, son sustanciales para obtener la operación deseada del sistema de control.

# 2 OBJETO Y JUSTIFICACIÓN DEL PROYECTO

Este proyecto se centra en el desarrollo de una librería básica para la automatización de un pórtico situado en los laboratorios del Departamento de Ingeniería de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de Sevilla. Este pórtico constituye una estación de trabajo en la célula de fabricación flexible a la que pertenece, por lo que se pretende completar su desarrollo para que adquiera las funcionalidades requeridas con el fin de que pueda desempeñar su papel en el proceso de fabricación.

La función principal de esta máquina será la del posicionamiento de piezas o *pallets,* que circulan por el sistema de transporte de la célula, mediante una pinza que se instalará como efector final.

En concreto, el pórtico es un sistema mecánico con tres articulaciones prismáticas que le confieren una morfología cartesiana con un volumen de trabajo en forma de ortoedro.

Está impulsado por *servomotores* en cada una de sus articulaciones, los cuales son gestionados a través de sus respectivos *servodrivers*. Estos ya disponen de la realimentación de la posición mediante *encoders* acoplados al propio eje del motor. La unidad de control que se usa para gestionar los *servodrivers* es en un PLC industrial, cuya programación es la que se va a desarrollar.

La máquina ya ha sido objeto de un trabajo previo, gracias al cual se ha solucionado el problema de las incompatibilidades entre la unidad de control utilizada y los *servodrivers*. El resultado fue el desarrollo de unas tarjetas electrónicas, a modo de interfaz de adaptación entre los diferentes dispositivos, que permiten la interconexión y la correcta transmisión de datos [4]. A pesar de que en un principio fueron realizadas pensando en un modo de operar diferente, han podido ser adaptadas a esta nueva configuración.

El presente proyecto es una continuación, cuyo objetivo es desarrollar la programación del *software* necesario para gestionar de forma eficiente el movimiento y la posición del pórtico. Por ello, se ha realizado un conjunto de funciones flexibles con capacidad de ser usadas en diferentes trabajos. Esta librería de funciones se basa principalmente en el uso de tarjetas de conteo rápido para recibir y generar las señales de gobierno del *servodriver.*

Las diferentes funciones se han elaborado siguiendo una programación por capas para conseguir el objetivo final, es decir, se ha partido desde las primitivas más básicas para luego utilizarlas como base de aquellas más generales con un mayor nivel de abstracción. Resulta evidente que la utilidad del sistema sería nula sin la programación del control.

La librería podrá ser integrada en todos aquellos proyectos que necesiten posicionar el efector final del pórtico en una posición definida y que, además, requieran gestionar los diferentes estados de funcionamiento, de error o de emergencia que precisen la parada programa o inmediata del dispositivo.

Por último, se presenta una pantalla de explotación, en la cual el usuario puede obtener información acerca del estado del sistema mecánico y gestionarlo de forma manual.

# 3 ESPECIFICACIONES DE LOS COMPONENTES

A continuación, se realiza una explicación detalla de los diferentes elementos, tanto *software* como *hardware,* que intervienen o han sido utilizados en el proyecto.

## 3.1. Pórtico

Es la estructura mecánica de aluminio que conforma la estación de trabajo dentro de la célula de fabricación. Básicamente está formada por cuatro pilares sobre los que se apoyan cuatro travesaños. Encima de estos últimos hay un par de rieles paralelos, y el espacio entre ambos está cubierto por una viga que aprovecha los carriles para poder desplazarse y que, además, permite el movimiento de un soporte a lo largo de ella. De esta forma, se puede alcanzar cualquier punto en un plano horizontal entre los pilares. Sobre este soporte móvil se sostendrá en el futuro un efector vertical para extender la zona de trabajo en esa dirección, obteniendo así un volumen con forma de tetraedro accesible por el actuador final.

Las dimensiones de la estructura son de 180 cm de alto, 100 cm de ancho y 118 cm de largo. Las distancias internas del espacio de trabajo son de 65 cm de alto, 91 cm de ancho y 102 cm de largo. En la Figura 3-1 se representa de forma esquemática la forma y dimensiones más características del pórtico.

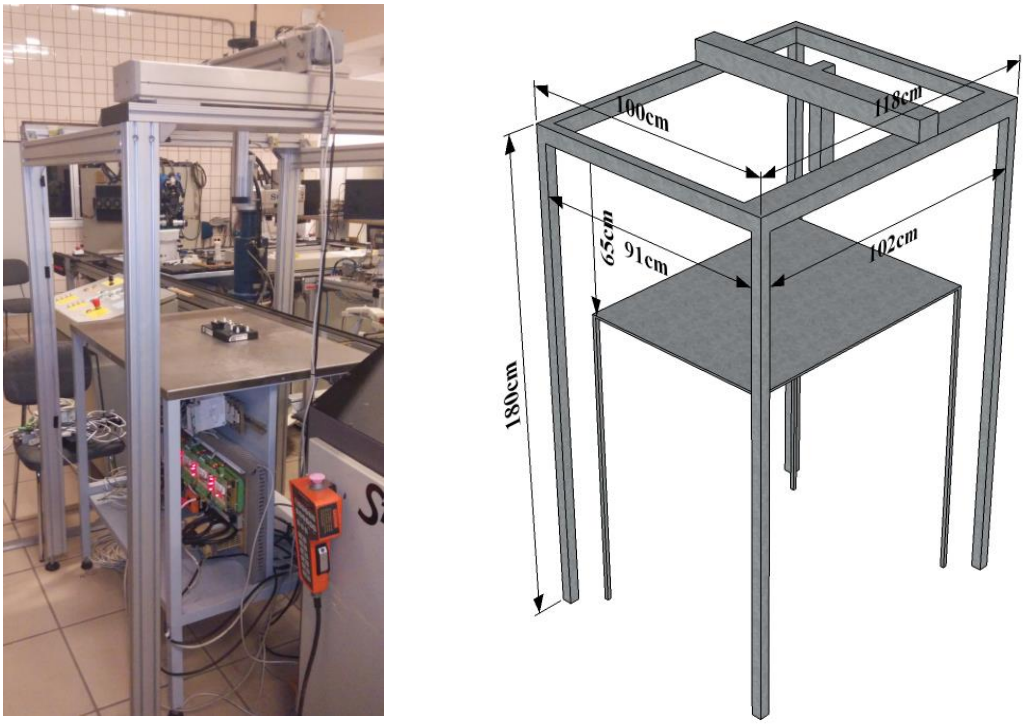



Figura 3-1. Pórtico

## 3.2. Sistema de Propulsión

Los dos ejes horizontales están motorizados con conjuntos de *servomotor* y *servodriver* de la serie OMNUC U del fabricante OMRON. En concreto, la viga que se desplaza entre los rieles paralelos está propulsada por el modelo R88M-U10030VA-S1 de 100 W sin freno junto al *servodriver* R88D-UP04V de la misma potencia, y el otro eje, por el par R88M-U05030VA-S1 de 50 W también sin freno y el *servodriver* R88D-UP03V de igual potencia. Serán nombrados *Motor 1* y Motor 2 respectivamente.

Figura 3-2. Conjunto *servomotor* y *servodriver* [5] [6]

Estos motores, tal y como es normal en los de su tipo, incluyen un sensor de posición, constituido por un *encoder* incremental, para que los *servodrivers* tengan en todo momento la información de la posición del eje y puedan realizar el control. El usuario puede elegir uno de los siguientes tipos de control:

- **Control de posición**: Permite controlar con precisión la posición y velocidad del *servomotor*. La referencia puede ser dada como pulsos y dirección, como pulsos de avance y pulsos de retroceso, o bien, mediante un par de señales diferenciales desfasadas 90º. Indiferentemente de cual sea la elección, se requiere tanto la propia señal de referencia como su negada, puesto que se trata de señales diferenciales RS-422. Además, se puede escoger entre lógica positiva o negativa. En la Figura 3-3 se muestra la forma de estas señales en lógica positiva con cuatro pulsos en cada sentido.

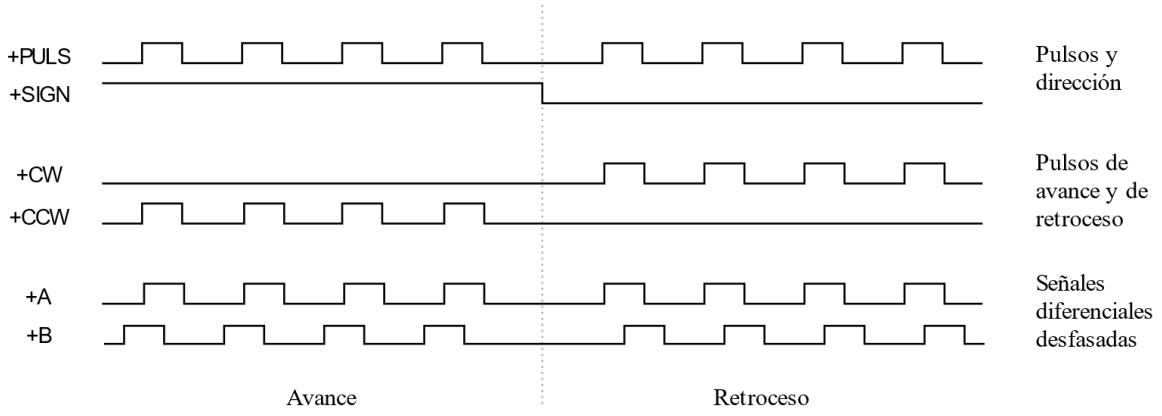



Figura 3-3. Tipos de referencias admitidas

- **Control de posición con entrada de parada de pulsos habilitada**: Si esta entrada se activa, se ignoran el resto de entradas de control durante el posicionamiento.
- **Control de velocidad según las predefinidas**: Permite predefinir hasta tres velocidades distintas, y luego, controlar el motor con alguna de ellas. El cambio entre unas y otras se realiza de forma progresiva. Útil para tareas de posicionamiento sencillas o de variación de velocidad.

- **Control de velocidad según las predefinidas y control en posición**: Este modo combina las características de ambos.

Las operaciones descritas son posibles gracias a los tres bucles de control realimentados en cascada, Figura 3-4, que integra el *servodriver*: el más interno es el lazo de corriente, seguido por el de velocidad y luego, el de posición. Para obtener el comportamiento deseado sin vibraciones ni movimientos extraños es necesario sintonizarlos, bien de forma manual, siguiendo el procedimiento descrito en la documentación, o bien, de forma automática, haciendo uso de la función que ofrece llamada "*auto-tuning"*.

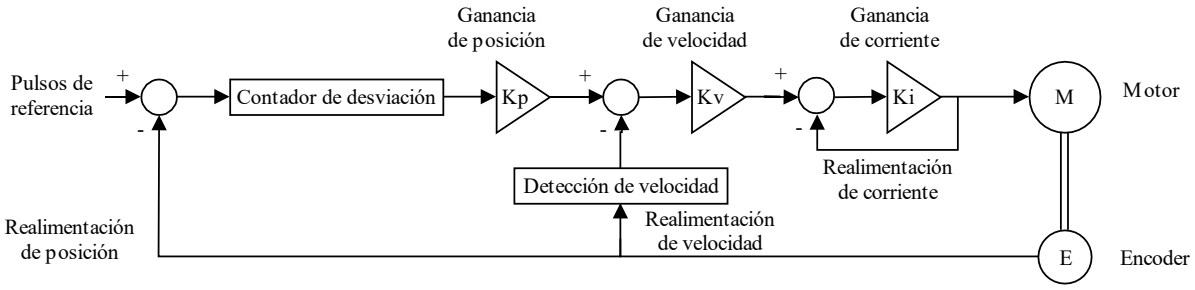



Figura 3-4. Bucle de control

El *servodriver* dispone de entradas y salidas para poder gobernar el sistema. Deben ser gestionadas externamente por el sistema de control, y todas ellas no siempre son necesarias. En la Tabla 3-1 se enumeran con una breve descripción.

Tabla 3-1. Conjunto de entradas y salidas CN1 del *servodriver*

| N.º Pin | Nombre | Función |
|---|---|---|
| | | **Entradas** |
| 01 | +PULS/CW/A | Pulsos de referencia / Pulsos de retroceso / Pulsos diferenciales (fase A) |
| 02 | -PULS/CW/A | |
| 03 | +SIGN/CCW/B | Señales de dirección / Pulsos de avance / Pulsos diferenciales (fase B) |
| 04 | -SIGN/CCW/B | |
| 05 | +ECRST | Señales de restablecimiento del contador de desviación |
| 06 | -ECRST | |
| 11 | PLC/SPD1 | Limitación de corriente en el avance / Entrada 1 de selección de velocidad |
| 12 | NCL/SPD2 | Limitación de corriente en el retroceso / Entrada 2 de selección de velocidad |
| 13 | +24VIN | Entrada de alimentación de 24V CC para control |
| 14 | RUN | Entrada de comando de energización de las bobinas |
| 15 | MING | Entrada de disminución de la ganancia de desaceleración |
| | IPG | Entrada de parada de pulso |
| | RDIR | Entrada de comando de dirección de rotación |
| 16 | POT | Si está a nivel bajo prohíbe el movimiento de avance |
| 17 | NOT | Si está a nivel bajo prohíbe el movimiento de retroceso |
| 18 | RESET | Restablecimiento de alarma |
| | | **Salidas** |
| 07 | BKIR | Freno engranado |
| 08 | INP | Posicionamiento completado |
| 09 | TGON | Detección de motor en movimiento |
| | CLIMT | Detección de corriente límite |
| 10 | OGND | Tierra común |
| 19 | EGND | Tierra del *encoder* |

| N.º Pin | Nombre | Función |
|---|---|---|
| 20 | +A | Fase A positiva del *encoder* |
| 21 | -A | Fase A negativa del *encoder* |
| 22 | -B | Fase B negativa del *encoder* |
| 23 | +B | Fase B positiva del *encoder* |
| 24 | +Z | Fase Z positiva del *encoder* |
| 25 | -Z | Fase Z negativa del *encoder* |
| 30 | ALO1 | Salida 1 de código de alarma |
| 31 | ALO2 | Salida 2 de código de alarma |
| 32 | ALO3 | Salida 3 de código de alarma |
| 33 | ALOCOM | Tierra común de código de alarma |
| 34 | NOT ALM | Salida negada de aviso de alarma |
| 35 | ALMCOM | Tierra común de aviso de alarma |
| 36 | FG | Tierra del bastidor |

Por otra parte, hay que añadir que el dispositivo incluye un amplio conjunto de funciones para adaptarlo a cada aplicación. Se exponen a continuación algunas relevantes para el proyecto:

- **Mecanismo de engranajes electrónico**: Permite variar la relación entre el número de pulsos de referencia usados para mover el motor y el número de pulsos que realmente ejecuta. Siendo este último igual al primero multiplicado por un cociente de dos parámetros conocido como relación de transmisión, Figura 3-5. El rango de este coeficiente está comprendido entre 0.01 y 100, y si es igual a la unidad, el motor gira una vuelta completa por cada 8192 pulsos de comando.
- **Resolución del *encoder* ajustable**: El número de pulsos por vueltas puede ser modificado desde 16 hasta 2048 pulsos/vuelta, Figura 3-6.
- **Arranque y parada suave**: Es posible programar tiempos de aceleración y desaceleración para evitar las perturbaciones introducidas por la inercia. Esto facilita la tarea de posicionamiento y elimina las oscilaciones. De esta forma, se consigue un perfil de velocidades trapezoidal, Figura 3-7.

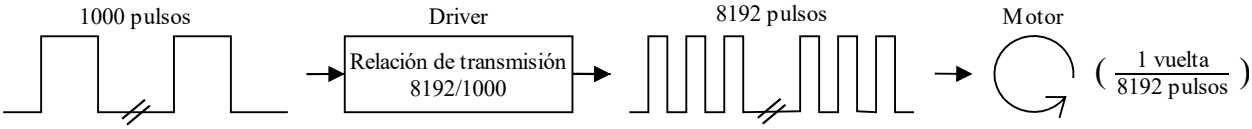



Figura 3-5. Engranajes electrónicos

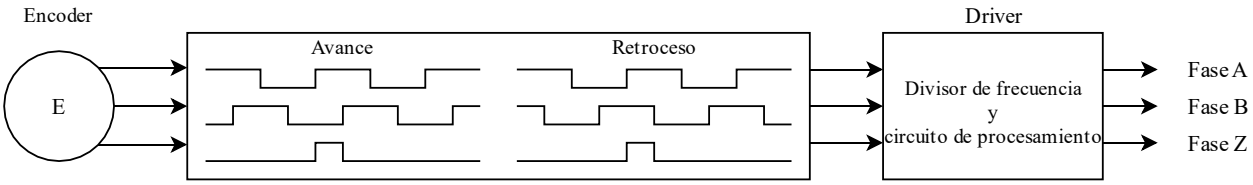



Figura 3-6. *Encoder* ajustable

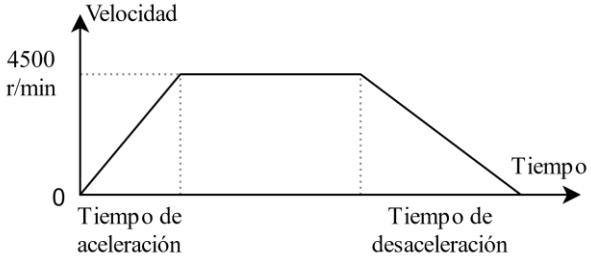



Figura 3-7. Trapecio de velocidades

- **Ejecución suave de órdenes**: Es posible programar tiempos de aceleración y desaceleración para ejecutar con suavidad trenes de pulsos de referencia de alta frecuencia.
- **Inversión de marcha**: Permite cambiar el sentido de giro sin modificar el cableado.
- **Secuencia de límite de recorrido**: Cuando se alcanza el límite del recorrido se puede seleccionar entre tres métodos de desaceleración: freno dinámico, desplazamiento libre o par de parada de emergencia.
- **Comunicación con ordenador**: Admite la conexión mediante un *software* especializado que permite modificar los parámetros de configuración, operar de forma manual, detectar errores y monitorizar el estado mecánico y electrónico.
- **Parada de emergencia por par**: Se pueden programar paradas en caso de colisión o sobrecarga para evitar dañar la maquinaria. Dicho de otra forma, esta función limita el *torque* máximo del motor.
- **Histórico de alarmas**: Almacena las últimas diez producidas y sus causas. Cuando se produce una, el circuito de potencia de alimentación del *servomotor* se desconecta automáticamente hasta que la alarma es reseteada. En la Tabla 3-2 se exponen las que se pueden dar.

Tabla 3-2. Listado de alarmas

| Código | Código de alarma | | | NOT ALM | Descripción |
|---|---|---|---|---|---|
| | ALO1 | ALO2 | ALO3 | | |
| a.02 | 0 | 0 | 0 | 0 | La suma de comprobación de los parámetros en memoria no coincide |
| a.04 | 0 | 0 | 0 | 0 | Parámetros de configuración incorrectos |
| a.10 | 1 | 0 | 0 | 0 | Detección de sobrecalentamiento o sobreintensidad |
| a.31 | 1 | 1 | 0 | 0 | Los pulsos restantes en el contador de desviación exceden el valor máximo seleccionado en Cn-1E |
| a.40 | 0 | 0 | 1 | 0 | Sobretensión en el circuito de CC principal |
| a.51 | 1 | 0 | 1 | 0 | Exceso de velocidad. Velocidad mayor que 4950 r/min |
| a.52 | 1 | 0 | 1 | 0 | Exceso de velocidad de referencia. Referencia mayor que 4700 r/min |
| a.70 | 1 | 1 | 1 | 0 | Sobrecarga: Par de salida superior al 120% del par nominal |
| a.71 | 1 | 1 | 1 | 0 | Sobrecarga: Par de salida superior al 135% del par nominal |
| a.72 | 1 | 1 | 1 | 0 | Sobrecarga: Par de salida entre el 120% y 135% del par nominal |
| a.c1 | 1 | 0 | 1 | 0 | Fallo en la alimentación o conexión con el *encoder* |
| a.c2 | 1 | 0 | 1 | 0 | *Encoder* mal cableado o desconectado |
| a.c3 | 1 | 0 | 1 | 0 | Fase A o B del *encoder* desconectada o cortocircuitada |
| a.c4 | 1 | 0 | 1 | 0 | Fase Z del *encoder* desconectada o cortocircuitada |
| a.f3 | 0 | 1 | 0 | 0 | Fallo momentáneo en la alimentación, ha sido restablecida sin periodo de retención |
| a.99 | 0 | 0 | 0 | 1 | Alimentación de restablecimiento de alarma activada |
| cpf00 | 0 | 0 | 0 | 0 | Los datos no han podido ser transmitidos después del encendido |
| cpf01 | --- | --- | --- | --- | Error de comunicación por sobretiempo |

La información ha sido recopilada de los manuales de usuario y fichas técnicas de estos equipos. Si se desea ampliar o concretar la información, se recomienda su lectura. [7] [8]

## 3.3. Controlador Lógico Programable

Un autómata programable es un tipo de computador de tiempo real especialmente diseñado para labores de automatización de procesos. Es ampliamente usado en industrias y en control de sistemas electromecánicos. Su arquitectura varía ligeramente respecto a un computador convencional, puesto que posee periféricos específicos con multitud de entradas y salidas para interactuar con el entorno. Además, está preparado para trabajar en ambientes hostiles con presencia de campos electromagnéticos, polvo, ruido, vibraciones y otros factores.

La tarea que desempeña este elemento en el presente proyecto es la de la gestión de los *servomotores,* y también, la comunicación con otros dispositivos o con el operador. El PLC proporciona las consignas de movimientos que se deben realizar en cada momento para llevar a cabo las diferentes operaciones asignadas al pórtico. Asimismo, para actuar en consecuencia, interpreta las señales provenientes de los *servodrivers* y del conjunto de sensores de contacto y de posicionamiento. A esto hay que añadir que es capaz de gestionar posibles estados de error y alarma para aportar seguridad y robustez. Con todo esto, se puede decir que conforma la lógica del proyecto.

El dispositivo es modular para poder adaptarlo a las necesidades de cada usuario. En este caso se hace uso de los elementos expuestos a continuación.

Figura 3-8. Controlador Lógico Programable instalado

Las características e imágenes de los módulos han sido extraídas de la página del fabricante, la cual se propone en caso de querer ampliarla. [9]

### 3.3.1 Bastidor y Fuente de Alimentación

El bastidor ejerce varias funciones, además de ser el soporte físico para el resto de elementos, posee un bus de comunicaciones interno para la transmisión de datos entre ellos. A esto hay que añadir que también distribuye la alimentación eléctrica para la lógica de cada módulo. Permite la desconexión y conexión en caliente de los módulos, a excepción de la alimentación y la CPU.

Se ha utilizado uno con 8 ranuras modelo BMX XBP 0800, Figura 3-9, compatible con la gama *Modicon*. Este incluye además dos *slots* que están reservado para un módulo de ampliación de bastidor y uno de alimentación.

La fuente de alimentación se encarga de suministrar la energía a los circuitos electrónicos del resto de elementos del PLC, para ello posee dos salidas CC de 24 V y 3,3 V. Del mismo modo, incluye una más auxiliar de 24V CC destinada a sensores externos. Está equipada con diferentes mecanismos de seguridad como fusibles internos, relé de alarma, led de estados y circuitos de protección contra sobrecarga, sobretensión y cortocircuito.

Se ha utilizado el modelo BMX CPS 2000, Figura 3-10, el cual se alimenta de la red eléctrica a 230 V AC y puede entregar hasta 20 W de potencia.

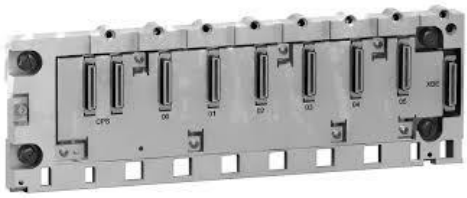


Figura 3-9. BMX XBP 0800

Figura 3-10. BMX CPS 2000

### 3.3.2 **Procesador**

El procesador es la pieza central del autómata programable, gestiona el resto de componentes y realiza las comunicaciones con otros equipos. Lleva integrado un reloj de tiempo real para cumplir las especificaciones temporales de la tarea que desempeña, el cual, sigue incluso contando tras retirar la alimentación. Gracias al panel de *leds* en la parte frontal es posible determinar su estado e identificar posibles errores. Existen diferentes modelos y sus principales diferencias son el tamaño de memoria, el número máximo de entradas y salidas que puede supervisar y las conexiones que integra.

Su programación y supervisión se realiza por medio de un ordenador con el *software* adecuado expuesto a continuación. Para ello se usa de la conexión USB o el protocolo TCP/IP.

El módulo elegido ha sido el BMX P34 2020, Figura 3-11, el cual incluye conexión Ethernet, USB y ModBus. Permite hasta 1024 conexiones binarias y 256 analógicas, y posee 256 Kb de memoria para los datos de programa. Su tiempo de ciclo y las características anteriores cumplen con holgura las necesidades del proyecto.

Figura 3-11. BMX P34 2020

### 3.3.3 **Módulo de Entradas y Salidas Digitales**

Gracias a este módulo es posible la interacción con el entorno. Las entradas permiten la adquisición de datos con señales del tipo todo o nada provenientes de sensores u otros dispositivos y, al mismo tiempo, las salidas envían señales o accionan preactuadores. Las conexiones están protegidas ante sobrecargas, cortocircuitos, inversiones de polaridad y sobretensiones inductivas. Existen tarjetas que funcionan en corriente continua o alterna y, en cualquier caso, estas deben ser alimentadas externamente por una fuente adecuada independiente.

Existe una gran variedad de módulos de este tipo con distinto número de puertos y diversas características. Se han usado dos del modelo BMX DDM 16022, Figura 3-12, este es un módulo binario que funciona con señales de 24V CC y que tiene 16 canales de lógica positiva, es decir, sus 8 entradas reciben corriente de los sensores (común positivo) y sus 8 salidas proporcionan corriente a los preactuadores (común negativo), Figura 3-14.

Figura 3-12. BMX DDM 16022

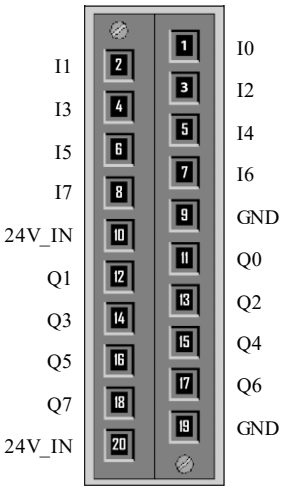



Figura 3-13. Diagrama de conexiones

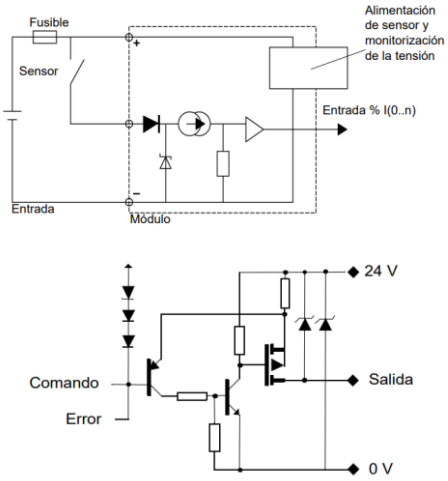



Figura 3-14. Circuitos de entrada y salida

## 3.3.4 Módulo de Conteo

Al igual que el anterior, este es un módulo de entradas y salidas binarias, con la principal diferencia de que puede tratar con señales de alta frecuencia. Esto hace que trabaje con cierta independencia del procesador, y por tanto, su funcionalidad debe ser programada previamente según su uso.

Se han utilizado tres tarjetas BMX EHC 0200. Esta tiene dos canales independientes y la frecuencia máxima admisible de las señales es de 60 kHz si son de entrada y 4 kHz si son de salida. Sus principales modos de funcionamiento son: conteo, conteo regresivo, conteo progresivo/regresivo, medición, medidor de frecuencias, generador de frecuencias y supervisión de ejes.

Para poder desempeñar estas tareas posee dos salidas por canal que pueden ser gestionadas bien desde la aplicación en la CPU, o bien, desde los bloques de funciones de salida que tiene el módulo. Estos bloques pueden actuar como comparadores entre umbrales definidos y valores internos de conteo, o bien, como generadores de pulsos. También se les puede configurar la polaridad y la recuperación ante fallos.

De igual forma, incluye por canal tres entradas rápidas, la primera para la medición o conteo progresivo simple, la segunda para poder realizar conteo diferencial o gestión de ejes, y la tercera de sincronización para marcar puntos iniciales. Y, por último, tres entradas lentas: la de habilitación del canal, la de toma de referencia de posición de inicio y la de captura de los registros internos. Todas ellas cuentan con filtros antirrebote programables.

### 3.3.4.1 Modalidad de Contador Grande Libre

Esta modalidad está diseñada especialmente para el etiquetado y la supervisión de ejes de los que se debe conocer su posición. El conteo progresivo y regresivo comienza tras el primer preestabecimiento del contador, y esta misma acción se puede realizar durante la operación para alterar la posición registrada. Admite múltiples configuraciones de conteo: pulsos de avance y de retroceso, pulsos y dirección, y cuadratura normal o inversa x1, x2 y x4.

La representación del valor almacenado se muestra con actualizaciones cada milisegundo. Si se desea obtener en un momento concreto con gran precisión, dispone de registros de captura activados por interrupción en la entrada física correspondiente. Asimismo, pueden ser utilizados los bloques de salida de comparación con este valor capturado.

El valor absoluto de los límites positivo y negativo del rango de números que puede contar es algo mayor de $2*10^9$, y en el caso de ser superado, se puede elegir entre saturar el valor o pasar al límite opuesto. También incluye funciones de histéresis para mitigar los errores introducidos por holguras en el eje mecánico al invertir el giro.

#### 3.3.4.2 Modulación de Ancho de Pulsos

En esta modalidad se aprovecha el reloj interno de la tarjeta para producir señales periódicas por medio del bloque de función de la salida Q0. La forma de onda es ajustable tanto en frecuencia, con variaciones de 0.1 Hz, como en el ciclo de servicio, con incrementos de 5%.

Gracias a la entrada de habilitación de canal es fácil activar o pausar la generación de la señal. También admite la coordinación de esta con otra, para ello, fuerza un flanco de subida cuando se activa la entrada de sincronización.

Figura 3-15. BMX EHC 0200

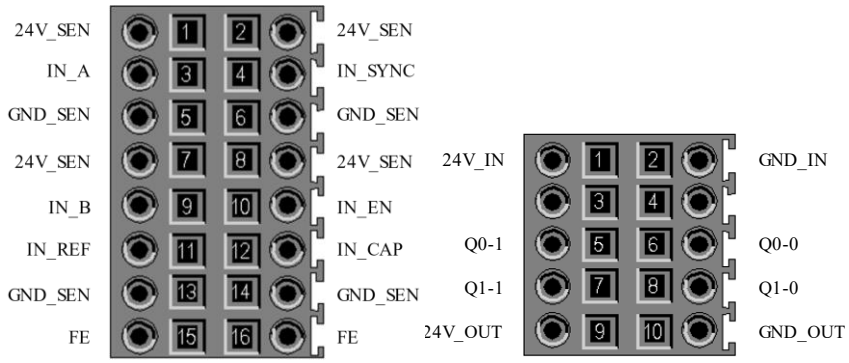



Figura 3-16. Conexiones por canal

## 3.4. Tarjetas de Adaptación de Señales

El proyecto previo sobre esta estación de la célula de fabricación consistió en la realización de una interfase electrónica que hace posible la comunicación entre el PLC y el *servodriver*. La unión de ambos equipos no está limitada por la funcionalidad de ellos, sino por el tipo de conexiones que presentan, ya que ni las familias lógicas ni los niveles de tensión de uno y otro son compatibles.

En primer lugar, se resolvió la alimentación eléctrica. Con este objetivo se desarrollaron dos tarjetas, una de ellas de distribución de corriente alterna hasta tres fuentes conmutadas, dos de 24 V - 1 A CC y la restante de 5 V - 2 A CC, y otra tarjeta de distribución de corriente continua para abastecer, a partir de las fuentes anteriores, el resto de tarjetas de la interfase, los módulos del autómata y también, en el caso de que hubiese, los frenos de los motores. Esto conforma los dos primeros elementos de la izquierda de la Figura 3-17.

Cabe destacar que estas fuentes exceden la potencia necesaria para alimentar todo el conjunto, por lo que en caso de alguna ampliación pueden seguir siendo utilizadas. A esto hay que añadir que una fuente de 24 V no está en uso al no haber aún ningún motor con freno.

Por otro lado, tener que redirigir todas las señales del *servodriver* y posteriormente transformarlas no era práctico ni necesario, ya que un buen número de ellas son prescindibles. Por este motivo, para realizar la interfaz, se seleccionaron las indispensables para tener un buen dominio sobre el motor. De las expuestas en la

Tabla 3-1 se seleccionaron: NOT ALM, ALMCOM, EGND, RESET, +CW, -CW, +CCW, -CCW, +24VIN, RUN, +A, -A, +B, -B, +Z y -Z.

El siguiente problema que se solventó fue el de la presencia de un único módulo de entradas y salidas digitales para gobernar ambos *servodrivers*. Al ser necesario redirigir las señales, se diseñó una tarjeta más con la capacidad de multiplexarlas y establecer la conexión con un motor u otro según se requiera en cada momento. Este módulo, en la zona central de la Figura 3-17, no se usa en el proyecto tal y como se manifestará posteriormente.

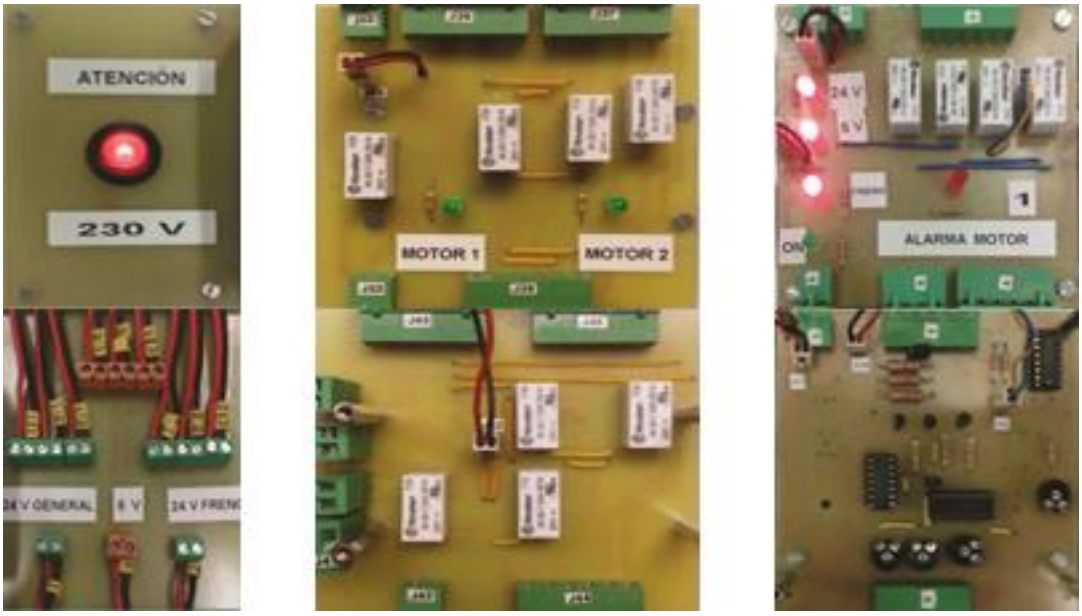



Figura 3-17. Interfaz de adaptación de señales [4]

Finalmente, el módulo más importante, el de adaptación de señales, a la derecha en la Figura 3-17. Para hacer posible las comunicaciones, aporta diferentes soluciones que transforman las conexiones del *servodriver* enumeradas anteriormente en las siguientes conexiones del PLC, Figura 3-18:

- **RUN**: Cuando esta salida del PLC es activada, el módulo permite que se propague hacia la entrada de mismo nombre del *servodriver* una señal compatible de 24 V, energizando así las bobinas del motor. También conmuta una salida de desbloqueo de freno, FRENO EXT, por si en un futuro se instalase.
- **PWM**: El módulo de conteo es capaz de generar un tren de pulsos de 24 V HTL, sin embargo, esta señal no es adecuada. Por este motivo, se usa un divisor de tensión para realizar un escalado a 5 V TTL y luego, un circuito integrado AM26LS31CD para hacer la conversión de RS-232 a RS-422. De esta forma, se consigue el par de pulsos diferenciales de avance y retroceso requeridos para la referencia.
- **INV GIRO**: Esta salida del PLC está conectada en la tarjeta de adaptación a la bobina de un relé y según su estado, el tren de pulsos de referencia diferencial generado previamente se redirige hacia las entradas +CW/-CW, para que sean interpretados de avance, o bien, hacia +CCW/-CCW si son de retroceso.
- **A / B / Z:** Las señales diferenciales provenientes del *encoder* son RS-422 a 5 V, y del mismo modo, son incompatibles con el autómata. Se usa el integrado AM26LS32CN, de funcionalidad complementaria al anterior, junto a potenciómetros de ecualización para convertirlas a RS-422. Posteriormente, con amplificadores inversores basados en un transistor, se eleva la tensión de las señales hasta 24 V HTL. Hay que señalar que también se usan puertas NOT, por lo que la polaridad de las entradas y las salidas coinciden.

- **ALA**: Cuando esta entrada conectada a NOT ALM es conmutada por el *servodriver,* el módulo propaga una señal análoga al PLC. La polaridad se invierte, por lo que ahora nivel alto significa que ha surgido alguna alarma.
- **RESET ALA**: Para poder reestablecer la operación tras una alarma es necesario restaurarla. Esto se realiza poniendo esta entrada a nivel alto desde el PLC, ya que conecta el pin RESET del *servodriver* a tierra.

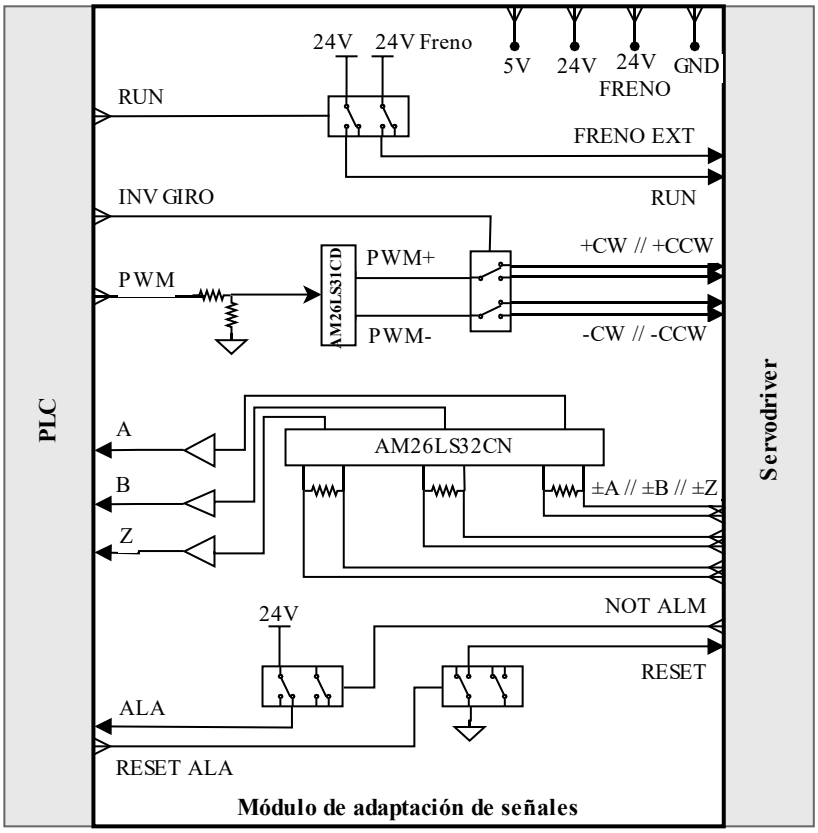



Figura 3-18. Esquema simplificado del módulo de adaptación de señales

## 3.5. **Unity Pro XL**

Para el desarrollo de las funciones se ha usado *Unity Pro XL* en la versión 12. Este es un *software* de Schneider Electric orientado a la programación, depuración, diagnóstico y operación para, entre otras, la gama *Modicom* de autómatas programables. Tiene un gran apoyo gráfico para facilitar su uso y acepta todos los lenguajes de programación recogidos en la norma IEC 61131-3.

El lenguaje utilizado para el desarrollo del código ha sido Ladder Diagramm, LD, el cual describe de forma gráfica un circuito de conmutadores de relé. Cada sección LD está compuesta de un raíl de alimentación izquierdo que actúa como alimentación, y otro raíl derecho a modo de conductor neutro. La programación consiste en la unión de ambos mediante diferentes tipos de objetos como contactos, que cierran o abren el circuito en función del valor de la variable a la que están asociados, o bobinas, que transfieren el estado de su conexión a una variable. Para realizar tareas más complejas también permite usar funciones y bloques funcionales, además de variables y expresiones no booleanas.

La librería está compuesta por un conjunto de bloques funcionales derivados, DFB. Se ha optado por este tipo debido a la necesidad de mantener estados internos en cada función, de esta forma, en cada ejecución el valor de las salidas puede ser diferente, aunque las entradas sean iguales. Este requisito es indispensable puesto que la respuesta depende en todo momento de la posición del efector final, la cual es una variable interna que es necesario mantener.

*Unity Pro* también es el *software* utilizado para la comunicación con el PLC, tanto en la programación, como en la supervisión durante la operación. Esto unido a las herramientas que incorpora para realizar Pantallas de Operador, hace posible la creación de interfaces HMI virtuales para la gestión y control del sistema. Aprovechando estas características se ha realizado la pantalla de explotación que se describirá posteriormente.

Además, cuenta con herramientas para la exportación/importación de DFB entre diferentes proyectos, y se hará uso de ellas para la distribución de la librería, facilitando así su posterior uso en diferentes aplicaciones. Al exportar el conjunto de funciones se genera un archivo con extensión *.XDB* que incluye todos los datos necesarios. Con las pantallas de explotación se procede de forma similar pero con formato *.XCR*.

## 3.6. SigmaWin - Yaskawa

Este es el *software* [10] libre utilizado para programar y configurar los *servodrivers*. A pesar de que los nuestros no son del mismo fabricante, puede ser utilizado, puesto que son compatibles. Requiere ser instalado en un PC y la conexión se realiza mediante puerto serie.

Está pensado para ser utilizado tanto por expertos como por principiantes debido a las facilidades que incluye para realizar las conexiones, tests de pruebas, sintonizaciones y más funciones como:

- Conversión y edición de parámetros.
- Notificación y liberación de las alarmas generadas incluyendo métodos de resolución.
- Exposición de datos del sistema tales como el estado de las entradas y salidas, el estado interno y datos del producto.
- Configuración de la unidad y de sus funciones.
- Representación gráfica de diversas variables como la velocidad y el par.
- Control manual del *servomotor*.

Para los motores usados existen dos tipos de parámetros que se pueden especificar. Por un lado, los parámetros de configuración, los cuales son realmente dos palabras de 16 bits, y según la combinación de estos, se obtiene un comportamiento u otro. Se detallan a continuación en la Tabla 3-3.

Por otro lado, están los parámetros de usuario, el cual es un conjunto más extenso donde cada uno de los elementos espera un valor numérico comprendido en un intervalo definido. La especificación de estos hace posible adaptar el *servomotor* a distintas aplicaciones. En la Tabla 3-4 se muestran junto a una breve explicación.

Es importante tener en cuenta que en ocasiones es necesario reiniciar el *servodriver* retirando la alimentación durante varios segundos para que los cambios se hagan efectivo. También incluye la posibilidad de exportar los parámetros elegidos con extensión *.YPM* para su posterior uso en la misma unidad u en otra similar.

Tabla 3-3. Parámetros de configuración [7]

| Bit | Por defecto | Función según valor |
|---|---|---|
| | | **Cn - 01** |
| 0 | 0 | 0 → El *servomotor* se energiza o se apaga mediante la entrada RUN<br>1 → El *servomotor* siempre está energizado |
| 1 | 0 | --- No se usa |
| 2 | 1 | 0 → Habilita la entrada de prohibición de movimiento de avance (POT)<br>1 → Siempre permite el movimiento de avance |
| 3 | 1 | 0 → Habilita la entrada de prohibición de movimiento de retroceso (NOT)<br>1 → Siempre permite el movimiento de retroceso |
| 4 | 0 | 0 → Señales TGON/CLIMT como salidas de detección de rotación<br>1 → Señales TGON/CLIMT como salidas de detección de corriente límite |

| Bit | Por defecto | | Función según valor |
|---|---|---|---|
| 5 | 1 | 0 → | La alarma se establece en el momento de recuperación de una parada momentánea |
| | | 1 → | La alarma se despejada sola en el momento de recuperación de una parada momentánea |
| 6 | 1 | 0 → | En paradas anormales el motor es detenido por frenado dinámico |
| | | 1 → | En paradas anormales el motor es detenido por desplazamiento libre |
| 7 | 1 | 0 → | En paradas anormales el freno dinámico permanece apagado tras la detención |
| | | 1 → | En paradas anormales el freno dinámico permanece encendido tras la detención |
| 8 | 0 | 0 → | El método de parada por exceso de recorrido depende del bit 6 |
| | | 1 → | La parada por exceso de recorrido se realiza con el par de parada de emergencia (Cn-06) |
| 9 | 0 | 0 → | Cuando se excede el recorrido, el motor es detenido y después apagado |
| | | 1 → | Cuando se excede el recorrido, el motor es detenido y después se bloquea su posición |
| A | 0 | 0 → | El contador de desviación es borrado mientras el motor está apagado |
| | | 1 → | El contador de desviación no es borrado mientras el motor está apagado |
| B | 1 | 0 → | Conmutación de controlador PI a P según los bits C y D |
| | | 1 → | Sin conmutación de controlador PI a P |
| D | 0 | 0,0 → | El valor del par motor se toma como condición para la conmutación (Cn-0C) |
| | | 0,1 → | El valor de la velocidad se toma como condición para la conmutación (Cn-0d) |
| C | 0 | 1,0 → | El valor de la aceleración se toma como condición para la conmutación (Cn-0E) |
| | | 1,1 → | El contador de pulsos de desviación se toma como condición para la conmutación (Cn-0F) |
| E | 0 | --- | No se usa |
| F | 0 | 0 → | Deshabilita la entrada de parada de pulsos. CN1-15 actúa como MING |
| | | 1 → | Habilita la entrada de parada de pulsos. CN1-15 actúa como IPG |
| **Cn - 02** | | | |
| 0 | 0 | 0 → | Rotación en sentido antihorario con referencias positivas |
| | | 1 → | Rotación en sentido horario con referencias positivas |
| 1 | 0 | --- | No se usa |
| 2 | 0 | 0 → | Control en posición con trenes de pulsos como referencia. CN1-11 y 12 como PCL y NCL |
| | | 1 → | Control en velocidad según las predefinidas. CN1-11 y 12 como SPD, y CN1-15 es RDI |
| 5 | 0 | 0,0,0 → | Referencia del tipo pulsos y dirección |
| | | 0,0,1 → | Referencia del tipo pulsos de avance y pulsos de retroceso |
| 4 | 0 | 0,1,0 → | Referencia del tipo par de señales diferenciales desfasadas 90º (x1) |
| 3 | 1 | 0,1,1 → | Referencia del tipo par de señales diferenciales desfasadas 90º (x2) |
| | | 1,0,0 → | Referencia del tipo par de señales diferenciales desfasadas 90º (x4) |
| 6 | 0 | --- | No se usa |
| 7 | 0 | --- | No se usa |
| 8 | 0 | --- | No se usa |
| 9 | 0 | --- | No se usa |
| A | 1 | 0 → | Borrar el contador de pulsos de desviación cuando la señal está a nivel alto (ECRST) |
| | | 1 → | Borrar el contador de pulsos de desviación cada flanco de subida de la señal (ECRST) |
| B | 0 | 0 → | Unidades de la constante de integración del bucle de velocidad igual a 1 ms |
| | | 1 → | Unidades de la constante de integración del bucle de velocidad igual a 0,1 ms |
| C | 0 | 0 → | Filtro primario con la constante de tiempo del filtro de la referencia de par |
| | | 1 → | Filtro secundario con la constante de tiempo del filtro de la referencia de par |
| D | 0 | 0 → | Lógica positiva para los pulsos de referencia |
| | | 1 → | Lógica negativa para los pulsos de referencia |
| E | 0 | 0 → | Unidades de la desviación de posición igual a 1 comando |
| | | 1 → | Unidades de la desviación de posición igual a 100 comandos |
| F | 0 | --- | No se usa |

Tabla 3-4. Parámetros de usuario [7]

| N.º | Parámetro | Por defecto | Rango | Unidades |
|---|---|---|---|---|
| Cn-00 | Modo de revisión del sistema | --- | --- | --- |
| Cn-01 | Parámetro de configuración número 1 | --- | --- | --- |
| Cn-02 | Parámetro de configuración número 2 | --- | --- | --- |
| Cn-04 | Ganancia del bucle de velocidad | 80 | [1 - 2000] | Hz |
| Cn-05 | Constante de integración del bucle de velocidad | 20 | [2 -10000] | ms |
| Cn-06 | Par motor durante la parada de emergencia | par máx. | [0 - par máx.] | % |
| Cn-07 | Arranque suave. Tiempo desde 0 a 4500 r/min. | 0 | [0 - 10000] | ms |
| Cn-08 | Par motor máximo para el movimiento de avance | par máx. | [0 - par máx.] | % |
| Cn-09 | Par motor máximo para el movimiento de retroceso | par máx. | [0 - par máx.] | % |
| Cn-0A | Pulsos del encoder por vuelta | 1000 | [16 - 2048] | pulsos/vuelta |
| Cn-0B | Umbral de detección de rotación del motor | 20 | [1 - 4500] | r/min |
| Cn-0C | Par motor máximo para pasar de un controlador PI a P | 200 | [0 - par máx.] | % |
| Cn-0D | Velocidad máxima para pasar de un controlador PI a P | 0 | [0 - 4500] | r/min |
| Cn-0E | Aceleración máxima para pasar de un controlador PI a P | 0 | [0 - 3000] | 10 (r/min)/s |
| Cn-0F | Desviación en pulsos máxima para pasar de PI a P | 10 | [0 - 1000] | De comando |
| Cn-10 | Velocidad de rotación durante el control manual | 500 | [0 - 4500] | r/min |
| Cn-11 | Número de pulsos por vuelta del *encoder* usado | 2048 | 2048 | pulsos/vuelta |
| Cn-12 | Retraso entre la orden de freno y la detención | 0 | [0 - 50] | 10 ms |
| Cn-15 | Velocidad para la activación de la orden de freno | 100 | [0 - 4500] | r/min |
| Cn-16 | Espera entre la detención y la señal de aviso de freno | 50 | [10 - 100] | 10 ms |
| Cn-17 | Constante de tiempo de filtrado para la referencia de par | 4 | [0 - 250] | 100 µs |
| Cn-18 | Par de avance para la corriente de referencia máxima | 100 | [0 - par máx.] | % |
| Cn-19 | Par de retroceso para la corriente de referencia máxima | 100 | [0 - par máx.] | % |
| Cn-1A | Ganancia del bucle de posición | 40 | [1 - 500] | 1/s |
| Cn-1B | Rango para la señalización de posición alcanzada | 3 | [0 - 250] | De comando |
| Cn-1C | Sesgo para el control en posición | 0 | [0 - 450] | r/min |
| Cn-1D | Compensación de avance para el control en posición | 0 | [0 - 100] | % |
| Cn-1E | Nivel de desbordamiento del contador de desviación | 1024 | [1 - 32767] | x256 comandos |
| Cn-1F | Velocidad de rotación número 1 | 100 | [0 - 4500] | r/min |
| Cn-20 | Velocidad de rotación número 2 | 200 | [0 - 4500] | r/min |
| Cn-21 | Velocidad de rotación número 3 | 300 | [0 - 4500] | r/min |
| Cn-23 | Para suave. Tiempo desde 4500 a 0 r/min. | 0 | [0 - 10000] | ms |
| Cn-24 | Numerador de la relación de transmisión | 4 | [1 - 65535] | --- |
| Cn-25 | Denominador de la relación de transmisión | 1 | [1 - 65535] | --- |
| Cn-26 | Constante de tiempo para la aceleración/desaceleración | 0 | [0 - 640] | x 0.1 ms |
| Cn-27 | Filtro para la compensación de avance | 0 | [0 - 640] | x 0.1 ms |
| Cn-28 | Compensación de la ganancia del control en posición | 0 | [0 - 100] | --- |
| Cn-29 | Número de la unidad para comunicaciones multiejes | 0 | [0 - 14] | --- |

# 4 IMPLEMENTACIÓN PROPUESTA

En esta sección se exponen las diversas partes en las que se descompone la solución adoptada. A grandes rasgos, esta se basa en el uso del *servodriver* para realizar el control del motor, y de varias tarjetas de conteo para poder leer la posición de este último y poder darle las consignas de gobierno en cada momento.

Para cada eje es necesario tres canales de conteo. El primero es el encargado de generar el tren de pulsos de referencia para gobernar el *servodriver,* por lo que está en modo de modulación de ancho de pulsos. Los dos restantes están en modo de contador libre de 32 bits: mientras que un canal realiza el conteo de los pulsos junto a su dirección generados por el anterior, el otro supervisa las fases del *encoder* con el objetivo de conocer su posición.

A priori, la presencia de los dos últimos parece innecesaria, ya que con solamente uno se puede determinar la posición. Sin embargo, la combinación de ambos permite un control más sencillo e incluso la detección de anomalías en el correcto funcionamiento.

El modo en el que se procede es el siguiente. Cuando se requiere el desplazamiento a una nueva posición, se calcula el número de pulsos de comando que debe ejecutar el motor para alcanzarla partiendo desde su posición actual. Sabiendo esta cantidad, se comienza a generar los pulsos de control al mismo tiempo que se van contando. En el momento en el cual se alcanza el número de comandos necesarios, el propio canal que los cuenta envía una señal rápida al generador para deshabilitarlo y terminar así el movimiento.

El uso de los comparadores internos de estas tarjetas para detectar cuando se han generado los pulsos oportunos tiene una clara ventaja, están preparados para trabajar con señales de alta frecuencia, por lo que su respuesta será inmediata ante tal evento. Si se hiciese mediante un programa en CPU, se produciría un retraso entre el evento, la detección y la respuesta, lo que supondría imprecisiones en el posicionamiento.

Durante el proceso descrito, la posición del eje se determina a partir de los pulsos de referencia impuestos. Para ello, se usa la relación existente en la programación del *servodriver* entre los comandos recibidos y los pasos efectuados por el *servomotor*. No obstante, aunque en condiciones normales de operación ofrezca buenos resultados, esta forma de proceder puede conducir a fallos. Durante largos periodos de funcionamiento es posible que el propio controlador que integra el motor acumule pequeños errores de posicionamiento, o bien, que con cargas pesadas se pierdan o ganen pulsos por la inercia del movimiento o vibraciones. A esto hay que sumarle que si las bobinas no están energizadas el sistema mecánico puede ser movido por cualquier otro objeto, por el operario o incluso por el peso de la carga. En cualquier caso, el PLC acabaría teniendo una posición inexacta del dispositivo.

Por este motivo es necesario el uso de *encoders*. Aunque no se empleen directamente para determinar las órdenes de gobierno, ya que complicaría la programación, sí son útiles para detectar los errores descritos anteriormente y además corregirlos. La solución consiste en recalcular el registro de la posición en pulsos del motor que se tiene a partir de la posición en pulsos del *encoder*. Esto es posible gracias a que dichos dispositivos detectan y cuantifican con gran precisión cualquier movimiento, incluso los involuntarios.

Por otro lado, también facilitan la obtención de una referencia exacta de posición inicial por medio de su fase Z. Esta propiedad es sustancial, ya que de esta forma se asignan siempre las mismas coordenadas a cada punto del espacio puesto que, de no ser así, la capacidad de la máquina para realizar múltiples tareas sería nula. Para esta tarea se requiere también de un final de carrera conectado a la entrada de referencia, para conseguir así que el pulso de la fase mencionada que se escoge como referencia siempre sea el mismo.

Asimismo, para aprovechar aún más las tarjetas y evitar tener que usar otras adicionales, se dedican las restantes entradas y salidas libres para gestionar el resto de señales usadas durante el control. En especial, la salida sobrante del canal generador de pulsos es empleada para la señal de inversión de giro, la del canal que cuenta los pulsos anteriores para la de energización de las bobinas, y la del canal supervisor del *encoder* para reiniciar el estado de alarma. Respecto a las entradas de este último canal, se dedica la de habilitación para recibir el estado de alarma del *servodriver* y la de captura para otro final de carrera.

Las dos tarjetas adicionales de entradas y salidas clásicas no se usan, pero se dejan en la instalación por si fuesen necesarias cuando se instale el efector final.

## 4.1. **Configuración de las Tarjetas de Conteo**

Es evidente que se necesita una tarjeta y media para cada motor. En este caso, al tener dos ejes motorizados, se han escogido las situadas en los extremos para generar lo pulsos y contarlos, y la tarjeta central para los *encoders.*

Tal y como se puede apreciar, son la base fundamental del proyecto, y por ello requieren ser programadas de forma correcta. A continuación, en la Tabla 4-1, se muestran los parámetros escogidos para cada uno de los canales que intervienen en cada eje.

Tabla 4-1. Configuración por eje de las tarjetas de conteo

| N.º | Etiqueta | Valor | |
|---|---|---|---|
| | **Contador libre de 32 bits:** | **Tarjeta 1 Canal 1** | **Tarjeta 2 Canal 0** |
| 00 | Filtro de entrada A | Sin | Sin |
| 01 | Filtro de entrada B | Sin | Sin |
| 02 | Filtro de entrada SYNC | Sin | Sin |
| 03 | Filtro de entrada EN | Sin | Sin |
| 04 | Filtro de entrada REF | Sin | Sin |
| 05 | Filtro de entrada CAP | Sin | Sin |
| 06 | Fallo en la alimentación de la entrada | Fallo de E/S general | Fallo de E/S general |
| 07 | Fallo en la alimentación de la salida | Fallo de E/S general | Fallo de E/S general |
| 08 | Interfase de conteo | A=impulso, B=dirección | Cuadratura normal X1 |
| 09 | Factor de escalado | 1 | 1 |
| 10 | Modo de preestablecimiento | Flanco ascendente en REF | 1er flanco ascendente en SYNC tras REF=1 |
| 11 | Comportamiento de conteo | Bloquear en los límites | Bloquear en los límites |
| 12 | Ajuste de la captura 0 | Preestablecer condición | Preestablecer condición |
| 13 | Bloque salida 0 | Contador en ventana | Des |
| 14 | Bloque salida 1 | Des | Des |
| 15 | Ancho pulso 0 | 10 ms | 10 ms |
| 16 | Ancho pulso 1 | 10 ms | 10 ms |
| 17 | Polaridad 0 | Polaridad - | Polaridad + |
| 18 | Polaridad 1 | Polaridad + | Polaridad + |
| 19 | Recuperación de fallos | Retención desactivada | Retención desactivada |
| 20 | Retorno 0 | Con | Con |
| 21 | Retorno 1 | Con | Con |
| 22 | Valor de retorno 0 | 0 | 0 |
| 23 | Valor de retorno 1 | 0 | 0 |
| 24 | Evento | Deshabilitar | Deshabilitar |
| 25 | Número de evento | --- | --- |

| | Modulación de ancho de pulso: | Tarjeta 1 Canal 0 |
|---|---|---|
| 00 | Filtro de entrada SYNC | Sin |
| 01 | Filtro de entrada EN | Sin |
| 02 | Sincronizar flanco | Flanco ascendente en SYNC |
| 03 | Fallo en la alimentación de entrada | Fallo de E/S general |
| 04 | Fallo en la alimentación de salida | Fallo de E/S general |
| 05 | Polaridad 1 | Polaridad + |
| 06 | Recuperación de fallos | Retención desactivada |
| 07 | Retorno 0 | Con |
| 08 | Retorno 1 | Con |
| 09 | Valor de retorno 0 | 0 |
| 10 | Valor de retorno 1 | 0 |

Los valores de algunos parámetros coinciden en todos los canales. En concreto, no se activan los filtros antirrebotes orientados a conmutadores mecánicos en ninguna entrada, y tampoco se definen estados específicos de recuperación después de fallos. Seguidamente se destacan algunos parámetros:

- **Contador de pulsos de referencia (T1 C1)**: Es importante seleccionar que la interfase de conteo es de tipo pulsos y dirección, dado que una entrada recibe los pulsos generados y la otra la señal de selección de dirección. Un ejemplo de este tipo de consigna se puede observar en la Figura 3-3. Otro parámetro importante es el de la selección del bloque de salida 0, debe ser contador en ventana, de tal forma que cuando se cuenten los pulsos requeridos y el valor registrado se encuentre entre los dos umbrales definidos, se activará la salida refleja asociada, que al ser de polaridad negativa por la elección del parámetro 17, desactivará el canal de generación de pulsos al que está conectada. La opción de contador en ventana tiene la ventaja de que cuando los dos umbrales de comparación coinciden no se define un intervalo, sino un único valor.
- **Supervisión del encoder (T2 C0)**: Destacar en este caso que la interfase de conteo debe ser cuadratura normal X1 por el modo en el que funcionan estos sensores, y que también se puede contemplar en la Figura 3-6. El modo de preestablecimiento, mediante el cual se altera el valor del contador, debe ser tal que la posición registrada se reinicialice cuando se alcance el límite del eje marcado por el final de carrera en la entrada REF y, además, se detecte la fase Z conectada en SYNC. Esto conduce a que el tipo adecuado es 1$^{er}$ flanco ascendente en SYNC tras REF=1.

## 4.2. Configuración de los Servomotores

De todos los parámetros ajustables mencionados anteriormente es necesario modificar algunos respecto a su valor por defecto con el objetivo de adaptarlos a esta aplicación.

Los parámetros Cn-01 y Cn-02 se han adecuado al funcionamiento requerido del motor, por lo que, tiene permitido el giro en ambos sentidos, las bobinas solo se energizan cuando se solicita, el sentido de giro antihorario es el positivo y el control es de posición mediante referencias de tipo pulsos de avance y de retroceso.

El resto cumplen varios objetivos. Por un lado, están los parámetros de sintonización del controlador para eliminar movimientos indeseados como vibraciones o sobreoscilaciones. Luego, aquellos que limitan el par máximo capaz de generar, valor importante de ajustar para evitar daños en la estructura en caso de exceder la carrera. Y, por último, los factores de la relación de transmisión electrónica. Estos últimos se han escogido procurando que la velocidad de desplazamiento sea razonablemente alta con la referencia de mayor frecuencia que se puede dar y, que además, ambos ejes se muevan aproximadamente a la misma celeridad con referencias iguales.

Se exponen en la Tabla 4-2 aquellos que se han alterado junto a su nuevo valor. Los que no aparecen quedan igual que en la Tabla 3-3 y Tabla 3-4.

Tabla 4-2. Configuración de los *servomotores*

| N.º | Parámetro | Motor 1 | Motor 2 |
|---|---|---|---|
| Cn-01 | Parámetro de configuración número 1 | 08EC (0011011000010000) | 08EC (0011011000010000) |
| Cn-02 | Parámetro de configuración número 2 | 0409 (0001000000100000) | 0408 (0001000000100000) |
| Cn-04 | Ganancia del bucle de velocidad | 250 | 80 |
| Cn-05 | Constante de integración del bucle de velocidad | 120 | 20 |
| Cn-06 | Par motor durante la parada de emergencia | 100 | 100 |
| Cn-07 | Arranque suave. Tiempo desde 0 a 4500 r/min. | 1000 | 1000 |
| Cn-08 | Par motor máximo para el movimiento de avance | 60 | 55 |
| Cn-09 | Par motor máximo para el movimiento de retroceso | 60 | 55 |
| Cn-17 | Constante de tiempo de filtrado para la referencia de par | 4 | 5 |
| Cn-1B | Rango para la señalización de posición alcanzada | 3 | 3 |
| Cn-23 | Para suave. Tiempo desde 4500 a 0 r/min. | 1000 | 1000 |
| Cn-24 | Numerador de la relación de transmisión | 8000 | 10000 |
| Cn-25 | Denominador de la relación de transmisión | 1000 | 1000 |

## 4.3. Conexiones

Para llevarlas a cabo se hace uso de la interfaz de adaptación expuesta con anterioridad. No se utiliza el módulo de selección de motor ya que con las tres tarjetas de conteo del PLC se tienen suficientes recursos para gestionar los dos ejes sin necesidad de redirigir las señales.

En la Tabla 4-3 se exponen las conexiones entre todos los elementos que intervienen en un eje junto al color del cable. Es fácil extrapolar las uniones para el resto de ejes, puesto que todos los conectores equivalentes tienen el mismo nombre y siguen la misma distribución de pines.

También se incluye la Figura 4-1 en la que aparecen representadas todas las conexiones del autómata con el resto de componentes presentes en los dos ejes que actualmente están motorizados. Estos son los cuatro finales de carrera, los módulos de adaptación de cada eje y las fuentes de alimentación. El orden de las tarjetas es 1, 2 y 3, de izquierda a derecha.

Tabla 4-3. Conexiones para un solo eje

| Señal | Pin | Color | Pin |
|---|---|---|---|
| ***Servodriver* ↔ Módulo de adaptación de señales** | | | |
| +CW | CN1 - 01 | Marrón | J06 - 1 |
| -CW | CN1 - 02 | Amarillo | J06 - 2 |
| +CCW | CN1 - 03 | Translucido | J06 - 3 |
| -CCW | CN1 - 04 | Gris claro | J06 - 4 |
| +24VIN | CN1 - 13 | Rojo | J06 - 5 |
| RUN | CN1 - 14 | Verde | J06 - 6 |
| RESET | CN1 - 18 | Blanco | J52 - 2 |
| EGND | CN1 - 19 | Negro | J52 - 1 |
| +A | CN1 - 20 | Crema | J09 - 1 |
| -A | CN1 - 21 | Celeste | J09 - 2 |
| -B | CN1 - 22 | Azul | J09 - 4 |
| +B | CN1 - 23 | Gris | J09 - 3 |
| +Z | CN1 - 24 | Verde claro | J09 - 5 |

| Señal | Pin | Color | Pin |
|---|---|---|---|
| -Z | CN1 - 25 | Violeta | J09 - 6 |
| NOT ALM | CN1 - 34 | Rosa | J51 - 1 |
| ALMCOM | CN1 - 35 | Naranja | J51 - 2 |
| **PLC tarjeta de conteo ↔ Módulo de adaptación de señales** | | | |
| RUN | T1 - AUX - 07 | Rojo | J03 - 3 |
| PWM | T1 - AUX - 06 | Gris | J08 - 1 |
| INV GIRO | T1 - AUX - 05 | Marrón | J03 - 2 |
| ALA | T2 - CH0 - 10 | Rosa | J03 - 5 |
| RESET ALA | T2 - AUX - 05 | Gris | J03 - 4 |
| A | T2 - CH0 - 03 | Verde | J08 - 3 |
| B | T2 - CH0 - 09 | Blanco | J08 - 4 |
| Z | T2 - CH0 - 04 | Amarillo | J08 - 5 |
| Malla | T2 - CH0 - 16 | --- | --- |
| GND | | | J03 - 1 |
| 24V | | | J03 - 6 |
| GND | | | J08 - 2 |
| **PLC tarjeta de conteo ↔ PLC tarjeta de conteo** | | | |
| PWM | T1 - AUX - 06 | Gris | T1 - CH1 - 03 |
| INV GIRO | T1 - AUX - 05 | Marrón | T1 - CH1 - 09 |
| Habilitación PWM | T1 - AUX - 08 | Verde | T1 - CH0 - 10 |
| +24V | T1 - AUX - 01 | Rojo | T1 - AUX - 09 |
| GND | T1 - AUX - 02 | Azul | T1 - AUX - 10 |
| +24V | T2 - AUX - 01 | Rojo | T2 - AUX - 09 |
| GND | T2 - AUX - 02 | Azul | T2 - AUX - 10 |
| +24V | T1 - AUX - 09 | Rojo | T2 - AUX - 01 |
| GND | T1 - AUX - 02 | Azul | T2 - AUX - 10 |
| **PLC tarjeta de conteo ↔ Finales de carrera** | | | |
| FC1H | T1 - CH0 - 11 | Rojo | --- |
| FC1H_24V | T1 - CH0 - 01 | Blanco | --- |
| FC1 | T1 - CH0 - 12 | Verde | --- |
| FC1_24V | T1 - CH0 - 01 | Marrón | --- |
| **Módulo de adaptación de señales ↔ Freno externo** | | | |
| 24V Freno | J04 - 1 | --- | --- |
| GND | J04 - 2 | --- | --- |
| **Módulo de distribución de CC ↔ Módulo de adaptación de señales** | | | |
| 24V General | J02 - 1 | Rojo | J01 - 1 |
| | | Rojo | J10 - 1 |
| | | Rojo | J13 - 1 |
| GND General | J02 - 2 | Negro | J01 - 2 |
| | | Negro | J10 - 2 |
| | | Negro | J13 - 2 |
| 5V | J10 - 1 | Rojo | J11 - 1 |
| | | Rojo | J07 - 1 |
| | | Rojo | J14 - 1 |
| GND | J10 - 2 | Negro | J11 - 2 |
| | | Negro | J07 - 2 |
| | | Negro | J14 - 2 |
| 24V Freno | J06 - 1 | Rojo | J02 - 1 |
| GND Freno | J06 - 2 | Negro | J02 - 2 |

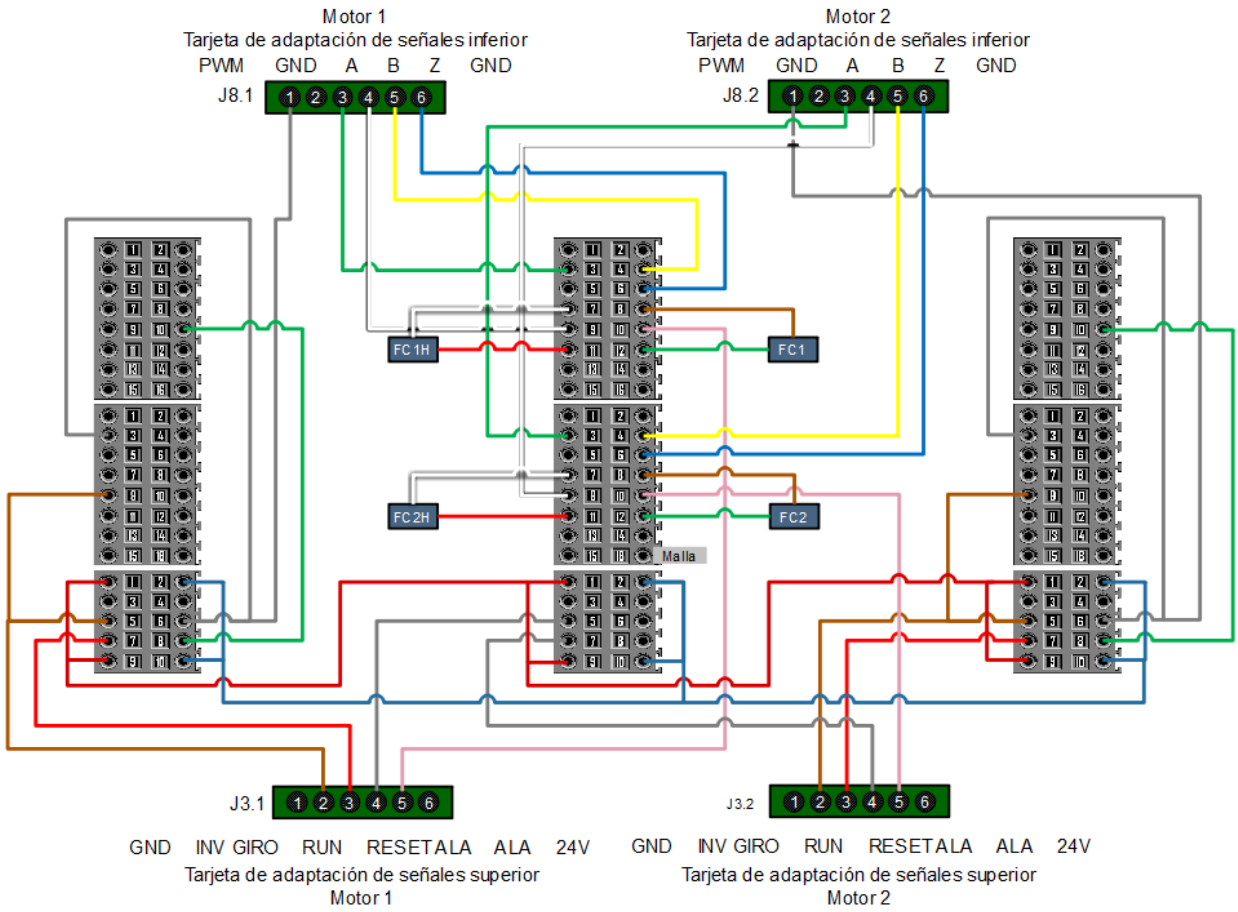



Figura 4-1. Conexiones del autómata

En la Figura 4-1 no se han representado las conexiones del 24 V y GND de las tarjetas de adaptación de señales debido a que pueden conectarse en cualquier punto con estas tensiones. Además, la alimentación de los finales de carrera también se puede tomar de la salida 1 del conector de 18 pines de cualquier canal.

## 4.4. **Cálculo de Parámetros**

Para la posterior implementación de la librería, se necesitan las relaciones de avance de los *encoders* y de los motores. Debido a que no existe documentación sobre el acoplamiento mecánico de los dispositivos, han tenido que ser calculados de forma experimental.

En ambos ejes se ha procedido de la siguiente forma. Se ha avanzado un número controlado de pulsos de comando, y se ha medido el incremento de la posición física de la articulación y el de la posición del *encoder*. Relacionando estos valores, y conociendo el número de comandos que necesita cada *servomotor* para completar una vuelta completa y el número de pulsos de *encoder* por vuelta, se han obtenido los resultados expuestos en la Tabla 4-4.

Tabla 4-4. Caracterización de los acoplamientos mecánicos

| Eje | Avance motor | Avance *encoder* | Carrera | Unidades |
|---|---|---|---|---|
| 1 | 200 | 100 | 6400000 | 0,00001 cm |
| 2 | 160 | 64 | 4600000 | 0,00001 cm |

# 5 LIBRERÍA DE FUNCIONES

La librería está compuesta por un conjunto de funciones que habilita la gestión, operación y supervisión del pórtico. Por consiguiente, al mismo tiempo que gobierna el *servodriver* mediante la generación de las señales adecuadas para la tarea en desarrollo, también detecta anomalías o errores en el correcto funcionamiento.

Está desarrollada siguiendo una estructura en capas en la cual cada función tiene un grado de abstracción diferente. Mientras que los niveles más internos se centran en producir o cuantificar el movimiento, los más externos aprovechan los resultados de los anteriores para desempeñar servicios más generales.

Esta forma de proceder tiene sus ventajas. Por ejemplo, admite la reutilización del código en otras máquinas en las que se requieren las mismas funcionalidades de los niveles más bajo, por estar basadas en el mismo sistema de propulsión, pero que a nivel más alto operan de forma diferente, o bien, en máquinas que operan igual pero la interfaz de comunicaciones con el motor es diferente, y por tanto, requieren reescribir los niveles más bajos.

Esto conlleva que a pesar de que la principal aplicación de la librería será este puesto de la célula de fabricación, no está limitada a ello. Su desarrollo no se ha cerrado específicamente al pórtico, tanto por lo expuesto previamente, como por no contener ningún dato específico del dispositivo y fundamentarse sobre variables modificables por el usuario que permiten la adaptación a otras situaciones.

Además, hay que añadir que se puede usar una función u otra según se necesite, lo cual es mejor opción que tener únicamente una y utilizarla siempre aunque incluya más opciones de las necesarias.

En concreto, el problema se ha dividido en los siguientes niveles, Figura 5-1:

- **Encoder**: Registrar la posición real del dispositivo por medio de este sensor.
- **Contador**: Registra la posición virtual del dispositivo por medio de los pulsos de referencia.
- **Motor**: Gestiona las señales del *servodriver* para generar movimientos definidos.
- **Acoplamiento**: Tiene en cuenta los mecanismos de transmisión del eje para asociar distancias reales a los desplazamientos.
- **Dispositivo**: Añade protección al dispositivo mediante el concepto de carrera y fija una posición de inicio inalterable.
- **Máquina**: Coordina todos los ejes involucrados en el movimiento.

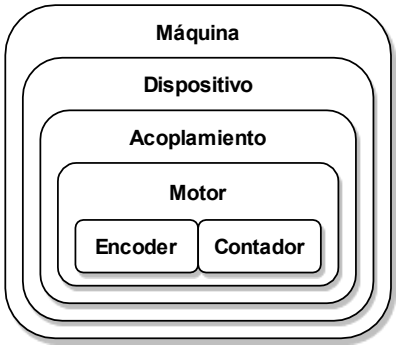



Figura 5-1. División por capas de la librería

# 5.1. Encoder

## 5.1.1 Descripción

El objetivo de la función es determinar la posición del *encoder*, y por tanto, la del eje, según la lectura de sus fases. Para ello, hace uso del canal en modo contador con interfase de conteo del tipo *"Cuadratura normal x1"*. El resultado será correcto siempre y cuando no se produzcan cortes en la alimentación.

Las unidades de la posición que proporciona son en pulsos de *encoder* y, en todo momento, su valor puede ser forzado a cualquiera indicado por el usuario. Esta acción de preestablecimiento se puede realizar de dos formas diferentes: el usuario lo solicita mediante un argumento de entrada dedicado a tal fin, o bien, se realiza en el primer flanco de subida de la fase Z del *encoder* tras la activación de la entrada *IN_REF*.

También permite la gestión de la bandera de sincronismo del propio canal, *SYNC_REF_FLAG*. Muestra su estado constantemente y, además, admite resetearla. Esto resultar útil para gestionar el sincronismo del eje, de modo que, el usuario puede usar la bandera para notificar problemas de calibración.

Esta variable es inicializada a nivel alto cuando la función es ejecutada, indicando así que el eje está sincronizado según la posición en la que se encontraba en el instante inicial. Tras ser reseteada, puede volver a ser activada realizando alguna de las dos operaciones de preestablecimiento.

## 5.1.2 Argumentos

- **Entradas/Salidas**
  1) **CanalEncoder - (T_SIGN_CPT_BMX):** Estructura del canal conectado a dicho sensor y que está configurado como contador grande libre.
- **Entradas**
  2) **ResetFlagSinc - (EBOOL):** Un flanco de subida en esta entrada resetea la bandera de sincronismo de la estructura de parámetros asociada al canal. Útil para notificar problemas de sincronismo.
  3) **Restaurar - (EBOOL):** Un flanco de subida en esta entrada cambia el valor actual del contador, o lo que es lo mismo, la posición del eje en pulsos de *encoder*, por la posición indicada en la entrada *NuevaPosPulsosEncoder*.
  4) **NuevaPosPulsosEncoder - (DINT):** Valor que se asigna al registro de posición en pulsos del *encoder* tras un flanco de subida en la entrada *Restaurar*. El rango admisible es [2147483648, +2147483647].
- **Salidas**
  2) **FlagSinc - (EBOOL):** Muestra el valor actual de la bandera de referencia de sincronismo, *SYNC_REF_FLAG*, de la estructura de parámetros asociada al canal en uso.
  3) **PosPulsosEncoder - (DINT):** Registro que almacena la posición del eje en pulsos de *encoder*. Su valor se actualiza conforme este sensor se mueve y los pulsos de sus fases son detectados por la tarjeta de conteo.

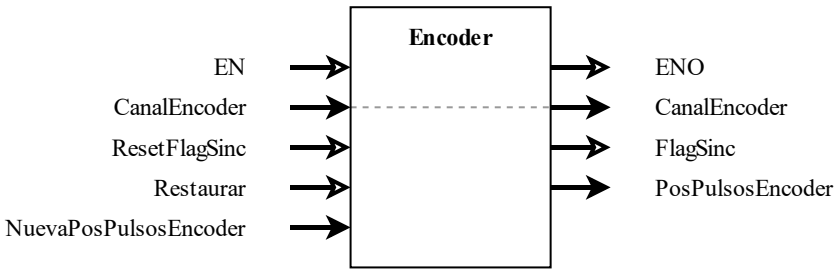



Figura 5-2. Bloque de la función *Encoder*

### 5.1.3 Ejemplo

Un ejemplo de funcionamiento se ilustra en el siguiente cronograma, Figura 5-3. En él se desarrolla la evolución de las salidas en función del movimiento del *encoder* y de los argumentos de entrada.

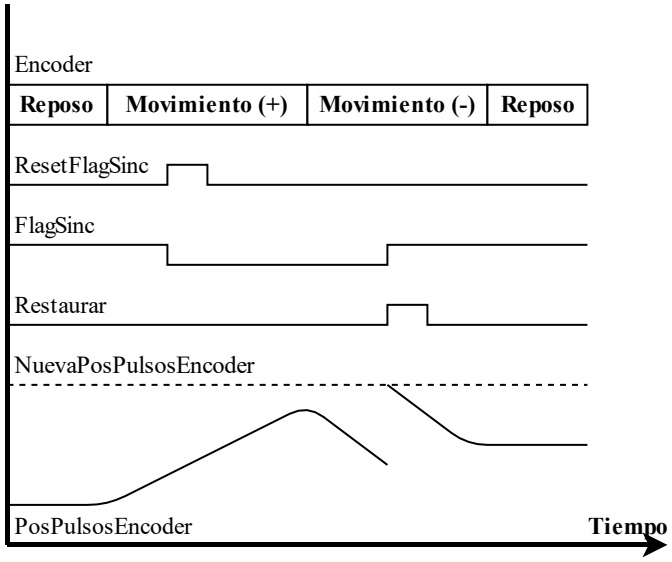



Figura 5-3. Cronograma de la función *Encoder*

## 5.2. ContadorPulsos

### 5.2.1 Descripción

El objetivo de la función es determinar la posición del eje según la lectura de los pulsos de referencia y de la señal de dirección generados. Para ello, hace uso del canal en modo contador con interfase de conteo del tipo *"A=impulso, B=dirección"*.

Las unidades de la posición que proporciona son en pulsos de motor y, en todo momento, su valor puede ser forzado a cualquiera indicado por el usuario. Esta acción de preestablecimiento se puede realizar únicamente cuando el usuario lo solicita a través de un argumento de entrada dedicado a tal fin.

A diferencia de la función anterior, que hace uso de un sensor especializado, la posición que esta ofrece es inconsistente. Su valor será correcto siempre que el eje no se someta a movimientos involuntarios por fuerzas externas, durante el movimiento o el reposo, que hagan que el *servomotor* avance o retroceda pulsos de referencia. Esto es así debido a que la función solo es capaz de registrar correctamente los desplazamientos ordenados llevados a cabo sin perturbaciones.

### 5.2.2 Argumentos

- **Entradas/Salidas**

  1) **CanalContador - (T_SIGN_CPT_BMX)**: Estructura del canal que recibe las señales de dirección y de referencia, y está configurado como contador grande libre.

- **Entradas**

  2) **Restaurar - (EBOOL)**: Un flanco de subida en esta entrada cambia el valor actual del contador, o lo que es lo mismo, la posición del eje en pulsos de motor, por la posición indicada en la entrada *NuevaPosPulsos.*

3) **NuevaPosPulsos - (DINT)**: Valor que se asigna al registro de posición en pulsos de motor tras un flanco de subida en la entrada *Restaurar*. El rango admisible es [2147483648, +2147483647].

- **Salidas**

2) **PosPulsos - (DINT)**: Registro que almacena la posición del eje en pulsos de motor. Su valor se actualiza conforme son generadas las consignas de movimiento y dirección, y la tarjeta de conteo las recibe.

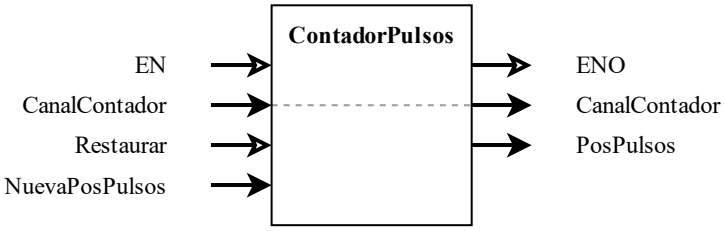



Figura 5-4. Bloque de la función *ContadorPulsos*

### 5.2.3 Ejemplo

Un ejemplo de funcionamiento se ilustra en el siguiente cronograma, Figura 5-5. En él se desarrolla la evolución de las salidas en función de la recepción de las señales de consigna y de los argumentos de entrada.

También se ha representado la posición real física del eje con el objetivo de mostrar que, si este es movido externamente, dicho desplazamiento no se refleja en el registro de la posición en pulsos de motor.

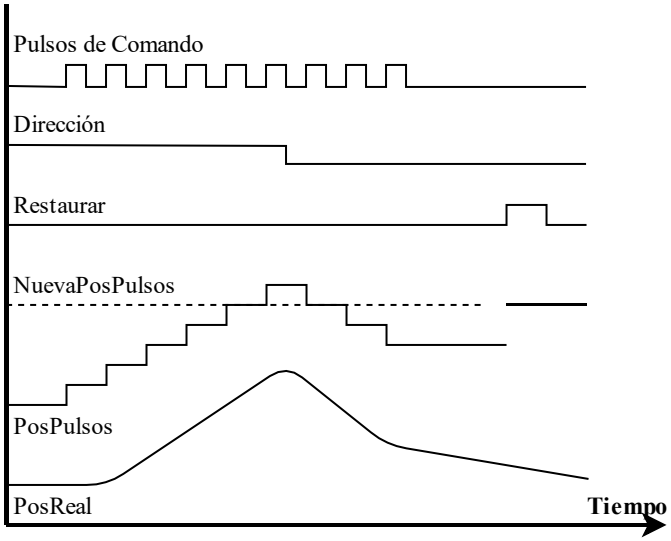



Figura 5-5. Cronograma de la función *ContadorPulsos*

## 5.3. MueveMotor

### 5.3.1 Descripción

Esta función habilita para usar el control en posición integrado en el *servodriver*. Gestiona las tarjetas de conteo para generar las señales de energización, dirección y pulsos, con el objetivo de poder alcanzar una posición concreta desde cualquier otro punto. Todas las posiciones están especificadas en unidades de pulsos de motor, puesto que se hace uso internamente de la función *ContadorPulsos*.

Cada vez que recibe una nueva posición objetivo, calcula el número de pulsos de diferencia con la posición actual, y los envía en forma de referencia al *servodriver*.

Además, debido a la integración de la función anterior, hereda características como la capacidad de mostrar la posición en cada momento según los movimientos que se van ordenando, y la posibilidad de forzar la posición a una deseada mediante la entrada correspondiente.

El usuario también tiene la capacidad de pausar el movimiento mediante un argumento de entrada de parada y luego, reanudarlo. También dispone de una entrada para gestionar el bloqueo del rotor del *servomotor* mediante la energización de sus bobinas durante las detenciones. Si se activa, el eje es inmovilizado cuando se alcanza la posición objetivo o el usuario solicita una detención con la entrada anterior. Este no es un freno mecánico, por lo que puede ser vencido si se aplica un par superior al que produce el *servodriver*.

La velocidad de desplazamiento se regula por medio de la frecuencia del tren de pulsos de comando. Este valor queda en manos del usuario, y si no es modificado, se mantiene constante sin aplicar un perfil trapezoidal durante todo el movimiento. El propio sistema de control que integra el motor ya realiza por sí solo arranques y paradas progresivas.

Esta función, al no tener realimentación de la posición, sufre las mismas desventajas que la anterior respecto a la inconsistencia de los valores de posición con los que trata. No obstante, este rasgo solo se pone de manifiesto si existen movimientos involuntarios, lo cual es excepcional y, por tanto, cumple con su cometido.

### 5.3.2 Argumentos

- **Entradas/Salidas**

  1) **CanalPWM - (T_UNSIGN_CPT_BMX)**: Estructura del canal que genera los pulsos de comando y está configurado en modo modulación de ancho de pulso.
  2) **CanalContador - (T_SIGN_CPT_BMX)**: Estructura del canal que recibe las señales de dirección y de referencia, y está configurado como contador grande libre.
  3) **CanalEncoder - (T_SIGN_CPT_BMX)**: Estructura del canal conectado a dicho sensor y que está configurado como contador grande libre.

- **Entradas**

  4) **RefPosPulsos - (DINT)**: Fija la posición en pulsos de motor que se desea alcanzar. El rango admisible es [-2147483648, +2147483647].
  5) **Hz - (UDINT)**: Fija la frecuencia en Hercios de los pulsos de comando generados, es decir, la velocidad de desplazamiento. El rango admisible es [1, 4000] y en caso de incumplirlo se utilizará el último valor válido.
  6) **STOP - (EBOOL)**: Si esta entrada se pone a nivel alto, se detiene la generación de pulsos de comando. De esta forma, se consigue detener el movimiento ordenado del *servomotor* inmediatamente. Cuando la entrada se desactiva, la función reanuda las consignas de movimiento por donde se ejecutaban antes de la interrupción. Su uso no altera el registro de la posición que se tiene del eje.
  7) **FrenoEnReposo - (EBOOL)**: Esta entrada solamente influye cuando el motor se encuentra en reposo, bien por coincidir la posición actual con la referencia, o bien, por estar la entrada *STOP*

activada. Si está a nivel alto y se cumple alguna condición anterior, forzará al *servodriver* a mantener las bobinas del *servomotor* energizadas para bloquear el rotor e impedir su movimiento.

8) **Restaurar - (EBOOL)**: Un flanco de subida en esta entrada cambia el valor actual del contador de pulsos de comando, o lo que es lo mismo, la posición del eje en pulsos de motor, por la posición indicada en la entrada *NuevaPosPulsos.*

9) **ResetAlarma - (EBOOL)**: Mientras esta entrada está activa, se envía una señal al *servodriver* para restaurar un posible estado de alarma y poder continuar con la operación.

10) **NuevaPosPulsos - (DINT)**: Valor que se asigna al registro de posición en pulsos de motor tras un flanco de subida en la entrada *Restaurar*. El rango admisible es [2147483648, +2147483647].

- **Salidas**

4) **ESTADO_ERROR - (INT)**: Informa mediante un código de cinco valores los diferentes estados del sistema:
   - -1. Alarma del *servodriver* activada.
   - 0. Motor en reposo y ausencia de errores.
   - 1. Motor en movimiento hacia la nueva posición.
   - 2. Motor forzado a detenerse por activación de la entrada *STOP.*
   - 3. Frecuencia designada para el tren de pulsos fuera de rango.

5) **PosPulsos - (DINT)**: Registro que almacena la posición del eje en pulsos de motor. Su valor se actualiza conforme son generadas las consignas de movimiento y dirección, y la tarjeta de conteo las recibe.

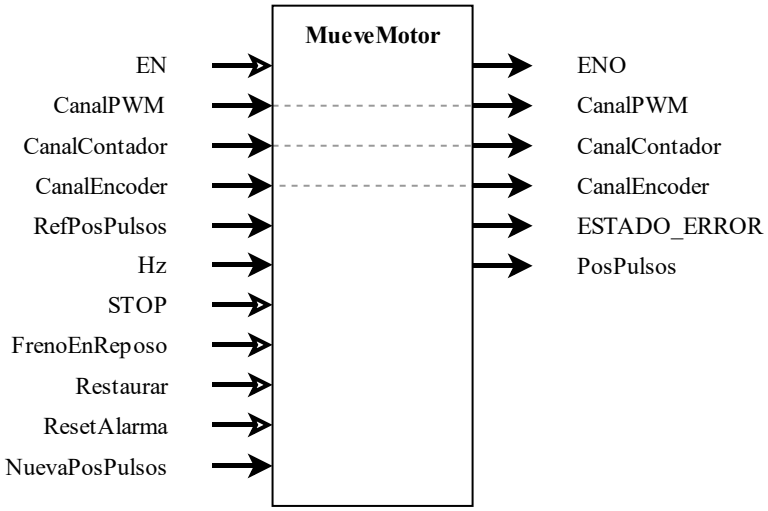



Figura 5-6. Bloque de la función *MueveMotor*

### 5.3.3 Ejemplo

Un ejemplo de funcionamiento se ilustra en el siguiente cronograma, Figura 5-7. En él se desarrolla la evolución de las salidas en función de la recepción de las señales de consigna y de los argumentos de entrada.

También se ha representado la posición real física del eje con el objetivo de mostrar que, si este es movido externamente, dicho desplazamiento no se refleja en el registro de la posición en pulsos de motor. Además, sirve para ilustrar el funcionamiento del freno, si se supone que sobre el eje se sostiene una carga elevada, comenzará a descender si no se activa la retención.

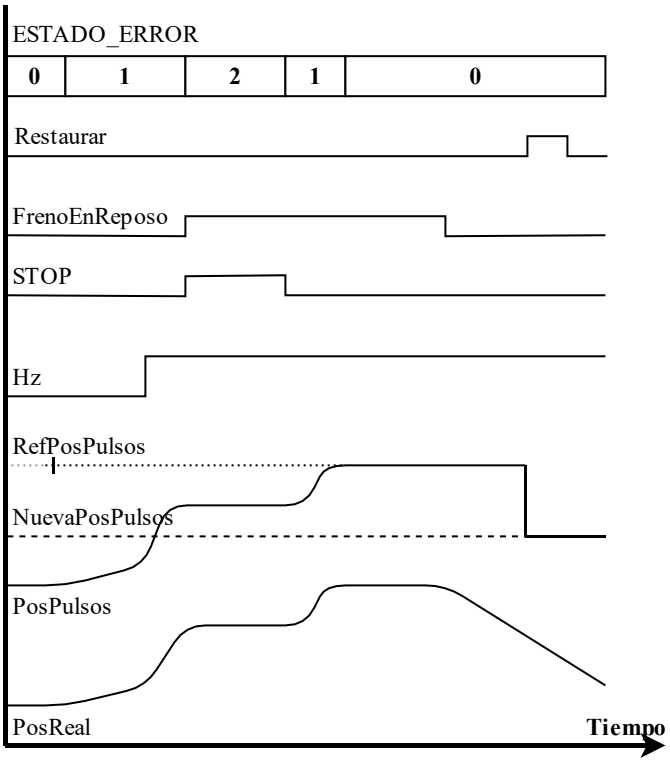



Figura 5-7. Cronograma de la función *MueveMotor*

## 5.4. **MueveAcoplamiento**

### 5.4.1 **Descripción**

Esta función añade las características necesarias para trabajar con las relaciones mecánicas de la transmisión. De esta forma, la referencia y la posición indicada ya no son en pulsos, sino en unidades de longitud lineal o angular.

Para conseguir este objetivo es necesario dos parámetros. Por un lado, el avance del *encoder*, que determina el desplazamiento del eje por cada pulso de este, y por otro lado, el avance del motor, que hace referencia a la longitud recorrida por cada pulso de comando que recibe y realiza el *servomotor*.

El hecho de ser argumentos de tipo entero obliga a modificarlos antes de ser introducidos para no perder precisión. Tienen que estar expresados en las mismas unidades de longitud y, además, hay que multiplicar ambos por la misma potencia de 10 que les haga perder todos los decimales. Las unidades resultantes fijarán las que tienen que ser usadas en el resto de argumentos de entrada y salida.

El uso del tipo entero tiene además una ventaja, aporta robustez al sistema gracias a que la representación interna de estos números es igual en cualquier computador, lo que facilita y refuerza la comunicación con otras máquinas que envíen los datos.

Hay que tener en cuenta que el eje no puede situarse físicamente en todas las posiciones, sino que solo lo puede hacer en aquellas que sean múltiplos del avance del motor. Esto es así debido a que el *servomotor* se desplaza una distancia constante por cada pulso de comando que recibe. Por ello, dada una referencia, la función mueve el conjunto a la indicada si es posible, o a la alcanzable inmediatamente anterior. De esta forma, si el avance del motor es 5 y la referencia 19, el motor se detiene en 15. A esto hay que añadir que, la posición devuelta se obtiene en función de la lectura del *encoder*, por lo que, si el avance de este fuese 4, la posición retornada sería 12.

A parte de mantener todas las características de las funciones anteriores, se apoya en la realimentación de la posición con el *encoder* para añadir más. Ahora es capaz de detectar movimientos involuntarios e intentar

corregirlos. La diferencia entre la posición virtual, calculada por el número de pulsos de comando generados, y la posición real es mostrada a modo de indicación del nivel de sincronismo. Si el usuario detecta que la diferencia es demasiado grande para ser admisible, puede solicitar su corrección. De esta forma, la función recalcula la posición virtual a partir de la real.

### 5.4.2 Argumentos

- **Entradas/Salidas**

1) **CanalPWM - (T_UNSIGN_CPT_BMX)**: Estructura del canal que genera los pulsos de comando y está configurado en modo modulación de ancho de pulso.
2) **CanalContador - (T_SIGN_CPT_BMX):** Estructura del canal que recibe las señales de dirección y de referencia, y está configurado como contador grande libre.
3) **CanalEncoder - (T_SIGN_CPT_BMX):** Estructura del canal conectado a dicho sensor y que está configurado como contador grande libre.

- **Entradas**

4) **RefPos - (DINT)**: Fija la posición, en las mismas unidades que los avances, que se desea alcanzar. El rango admisible es [-2147483648, +2147483647].
5) **AvanceMotor - (INT)**: Relación entre el desplazamiento del eje y los pulsos de comando ejecutados por el *servomotor*. Es necesario ingresar este valor como entero mediante la modificación de sus unidades, definiendo de esta forma las unidades que maneja la función en el resto de entradas y salidas. El rango admisible es [1, 32767].
6) **AvanceEncoder - (INT)**: Relación entre el desplazamiento del eje y los pulsos generados por *encoder*. Es necesario ingresar este valor como entero mediante la modificación de sus unidades, definiendo de esta forma las unidades que maneja la función en el resto de entradas y salidas. El rango admisible es [1, 32767].
7) **Hz - (UDINT)**: Fija la frecuencia en Hercios de los pulsos de comando generados, es decir, la velocidad de desplazamiento. El rango admisible es [1, 4000] y en caso de incumplirlo se utilizará el último valor válido.
8) **STOP - (EBOOL)**: Si esta entrada se pone a nivel alto, se detiene la generación de pulsos de comando. De esta forma se consigue detener el movimiento ordenado del *servomotor* inmediatamente. Cuando la entrada se desactiva, la función reanuda las consignas de movimiento por donde se ejecutaban antes de la interrupción. Su uso no altera el registro de la posición que se tiene del eje.
9) **FrenoEnReposo - (EBOOL)**: Esta entrada solamente influye cuando el motor se encuentra en reposo, bien por coincidir la posición actual con la referencia, o bien, por estar la entrada *STOP* activada. Si está a nivel alto y se cumple alguna condición anterior, forzará al *servodriver* a mantener las bobinas del *servomotor* energizadas para bloquear el rotor e impedir su movimiento.
10) **Rectificar - (EBOOL)**: Un flanco de subida en esta entrada recalcula el valor actual del contador de pulsos de comando, o lo que es lo mismo, la posición del eje en pulsos de motor, por medio de la posición real que otorga el *encoder.* Solventando así problemas de sincronismo por movimientos involuntarios.
11) **ResetFlagSinc - (EBOOL)**: Un flanco de subida en esta entrada resetea la bandera de sincronismo de la estructura de parámetros asociada al canal del *encoder*. Útil para notificar problemas de sincronismo.
12) **RestaurarEncoder - (EBOOL)**: Un flanco de subida en esta entrada cambia el valor actual de la posición real del eje por la posición indicada en la entrada *NuevaPosEncoder.*
13) **ResetAlarma - (EBOOL)**: Mientras esta entrada está activa, se envía una señal al *servodriver* para restaurar un posible estado de alarma y poder continuar con la operación.

14) **NuevaPosEncoder - (DINT)**: Valor de posición real que se utiliza para recalcular y sustituir el registro interno de posición en pulsos del *encoder* tras un flanco de subida en la entrada *Restaurar*. El rango admisible es [2147483648, +2147483647].

- **Salidas**

4) **ESTADO_ERROR - (INT)**: Añade cuatro posibles errores a los ya existentes:
   4. Entrada *AvanceMotor* fuera de rango.
   5. Entrada *AvanceEncoder* fuera de rango.
   6. El motor debería moverse y está detenido.
   7. El motor debería estar parado y se está moviendo.

5) **Sincronismo - (DINT)**: Valor que da a conocer la desincronización del eje mostrando la diferencia entre la posición real y la virtual. La cantidad es expresada en pulsos de comando.

6) **PosEncoder - (DINT)**: Registro que almacena la posición real del sistema en las mismas unidades que los avances. Su valor se actualiza según el movimiento de dicho sensor.

7) **FlagSincronismo - (EBOOL)**: Muestra el valor actual de la bandera de referencia de sincronismo, *SYNC_REF_FLAG*, de la estructura de parámetros asociada al canal del *encoder*.

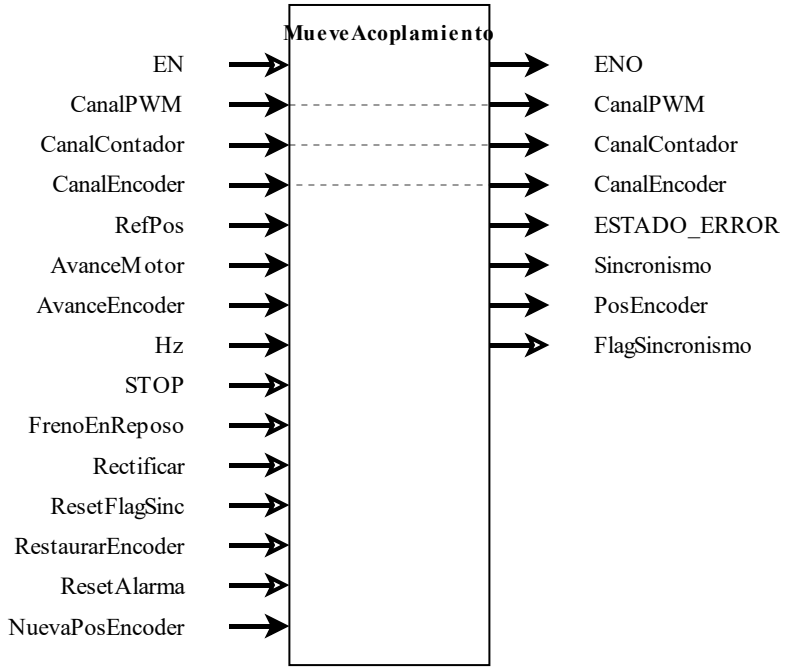



Figura 5-8. Bloque de la función *MueveAcoplamiento*

### 5.4.3 Ejemplo

Un conjunto mecánico posee un *encoder* de 2500 pulsos por vuelta, por lo que su avance es de 0,144 °/pulso. Por otro lado, el motor completa una vuelta tras recibir 100 pulsos de comando, resultando un avance de 0,36 º/C. La entrada *AvanceEncoder* deberá ser igual a 144 milésimas de grado por pulso, y la de *AvanceMotor* igual a 360 milésimas de grado por pulso.

Si se quisiera realizar un giro de media vuelta, es decir, alcanzar la posición 180°, partiendo desde una posición nula, 0°, el valor de la referencia debería ser: RefPos = 180000 milésimas de grado.

## 5.5. MueveDispositivo

### 5.5.1 Descripción

El nombre de esta función hace referencia a la integración de los elementos auxiliares del dispositivo, específicamente, de los finales de carrera en los extremos de cada eje.

El uso de estos por parte de la función aporta a la máquina dos cualidades muy importantes. Por un lado, otorga la posibilidad de definir la longitud de la zona de trabajo, conocida como carrera, elevando así la seguridad y protección del equipo. Por otro lado, habilita para definir una posición de origen inalterable denominada posición de *Home.* Esto permite que en cada uso del dispositivo se puedan asignar siempre las mismas posiciones a cada punto del espacio.

Solo se consideran admisibles aquellos movimientos que se realizan en el interior de la carrera. Si se fuerza una referencia fuera de rango, será notificado y el desplazamiento no se llevará a cabo. Además, si la función detecta mediante los finales de carrera que el eje ha excedido el espacio de trabajo permitido, provoca una parada inmediata por seguridad, ya que es indicativo de que la actividad no se está ejecutando correctamente.

Si esto ocurre, el usuario debe iniciar la búsqueda de la posición de *home* para poder mover de nuevo el dispositivo. El proceso para hallar dicho punto realiza la siguiente secuencia de movimientos:

1. Desplazamiento en el sentido positivo una décima parte de la carrera, o hasta alcanzar al sensor final de carrera, a la velocidad normal de operación.
2. Desplazamiento en el sentido negativo hasta alcanzar el sensor final de carrera designado para referenciar la posición de *Home* a la velocidad especificada para esta tarea.
3. Desplazamiento en el sentido positivo hasta detectar la fase Z del *encoder* a una décima parte de la velocidad de la etapa anterior.

Una vez terminado, lo notifica, resetea el fallo para permitir de nuevo el movimiento, e inicializa la posición. Cabe destacar que hasta que no se realice por primera vez, el sistema no se moverá para alcanzar la referencia, ya que se considera que la máquina no está calibrada tras ser encendida.

El resto de características heredadas de las funciones anteriores se mantienen.

### 5.5.2 Argumentos

- **Entradas/Salidas**

  1) **CanalPWM - (T_UNSIGN_CPT_BMX)**: Estructura del canal que genera los pulsos de comando y está configurado en modo modulación de ancho de pulso.
  2) **CanalContador - (T_SIGN_CPT_BMX)**: Estructura del canal que recibe las señales de dirección y de referencia, y está configurado como contador grande libre.
  3) **CanalEncoder - (T_SIGN_CPT_BMX)**: Estructura del canal conectado a dicho sensor y que está configurado como contador grande libre.

- **Entradas**

  4) **RefPos - (DINT)**: Fija la posición, en las mismas unidades que los avances, que se desea alcanzar. El rango admisible es [-2147483648, +2147483647].
  5) **AvanceMotor - (INT)**: Relación entre el desplazamiento del eje y los pulsos de comando ejecutados por el *servomotor*. Es necesario ingresar este valor como entero mediante la modificación de sus unidades, definiendo de esta forma las unidades que maneja la función en el resto de entradas y salidas. El rango admisible es [1, 32767].
  6) **AvanceEncoder - (INT)**: Relación entre el desplazamiento del eje y los pulsos generados por *encoder*. Es necesario ingresar este valor como entero mediante la modificación de sus unidades,

definiendo de esta forma las unidades que maneja la función en el resto de entradas y salidas. El rango admisible es [1, 32767].

7) **Hz - (UDINT)**: Fija la frecuencia en Hercios de los pulsos de comando generados, es decir, la velocidad de desplazamiento. El rango admisible es [1, 4000] y en caso de incumplirlo se utilizará el último valor válido.

8) **HzHome - (UDINT)**: Fija la velocidad de desplazamiento durante las dos últimas etapas de la búsqueda de la posición de *home.* El rango admisible es [1, 4000] y en caso de incumplirlo se utilizará la entrada *Hz.*

9) **STOP - (EBOOL)**: Si esta entrada se pone a nivel alto, se detiene la generación de pulsos de comando. De esta forma se consigue detener el movimiento ordenado del *servomotor* inmediatamente. Cuando la entrada se desactiva, la función reanuda las consignas de movimiento por donde se ejecutaban antes de la interrupción. Su uso no altera el registro de la posición que se tiene del eje.

10) **FrenoEnReposo - (EBOOL)**: Esta entrada solamente influye cuando el motor se encuentra en reposo, bien por coincidir la posición actual con la referencia, o bien, por estar la entrada *STOP* activada. Si está a nivel alto y se cumple alguna condición anterior, forzará al *servodriver* a mantener las bobinas del *servomotor* energizadas para bloquear el rotor e impedir su movimiento.

11) **Rectificar - (EBOOL)**: Un flanco de subida en esta entrada recalcula el valor actual del contador de pulsos de comando, o lo que es lo mismo, la posición del eje en pulsos de motor, por medio de la posición real que otorga el *encoder.* Solventando así problemas de sincronismo por movimientos involuntarios.

12) **Carrera - (DINT)**: Define la longitud del espacio de trabajo ubicado entre los finales de carrera en las mismas unidades que los avances. El rango admisible es (0, +2147483647].

13) **ResetAlarma - (EBOOL)**: Mientras esta entrada está activa, se envía una señal al *servodriver* para restaurar un posible estado de alarma y poder continuar con la operación.

14) **Home - (EBOOL)**: Un flanco de subida en esta entrada inicia el proceso de búsqueda de la posición *home.* Se puede solicitar en cualquier momento.

- **Salidas**

4) **ESTADO_ERROR - (INT)**: Añade dos posibles errores a los ya existentes:
    8. Dispositivo en fallo por la activación de algún final de carrera.
    9. Referencia de posición fuera de rango.

5) **Sincronismo - (DINT)**: Valor que da a conocer la desincronización del eje mostrando la diferencia entre la posición real y la virtual. La cantidad es expresada en pulsos de comando.

6) **PosEncoder - (DINT)**: Registro que almacena la posición real del sistema en las mismas unidades que los avances. Su valor se actualiza según el movimiento de dicho sensor.

7) **HomeOK - (EBOOL)**: Esta salida permanece a nivel bajo desde que se inicia la función hasta que se realiza la primera búsqueda de *home.* Tras esto, se mantendrá a nivel alto hasta que se solicite una nueva búsqueda: se desactivará para marcar el comienzo del proceso y se volverá a activar para indicar que ha sido completado.

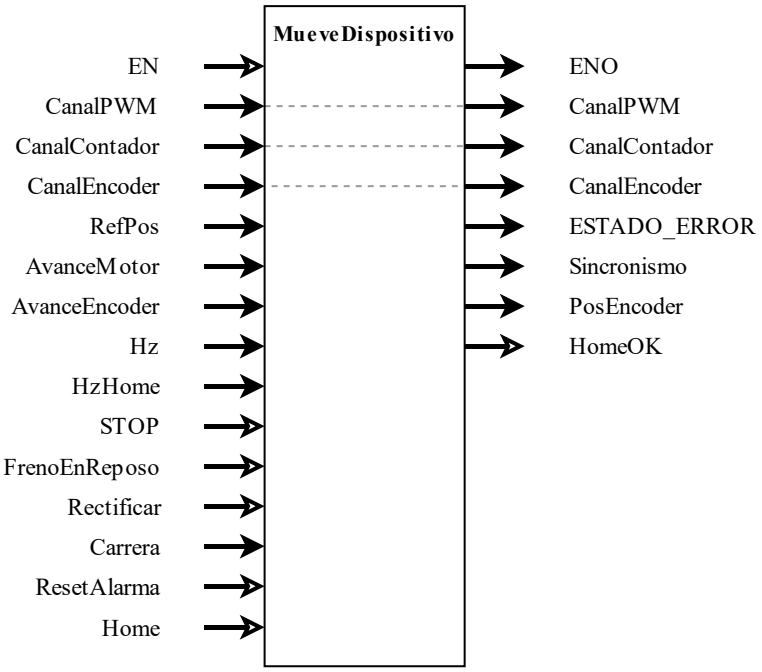



Figura 5-9. Bloque de la función *MueveDispositivo*

### 5.5.3 **Ejemplo**

Siguiendo el ejemplo anterior, si la zona de trabajo del dispositivo fuese de 720º, el valor a introducir en el argumento *carrera* sería 720000 milésimas de grado.

Un ejemplo de funcionamiento se ilustra en el siguiente cronograma, Figura 5-10. En él se desarrolla la evolución del sistema durante la búsqueda de la posición de *home*.

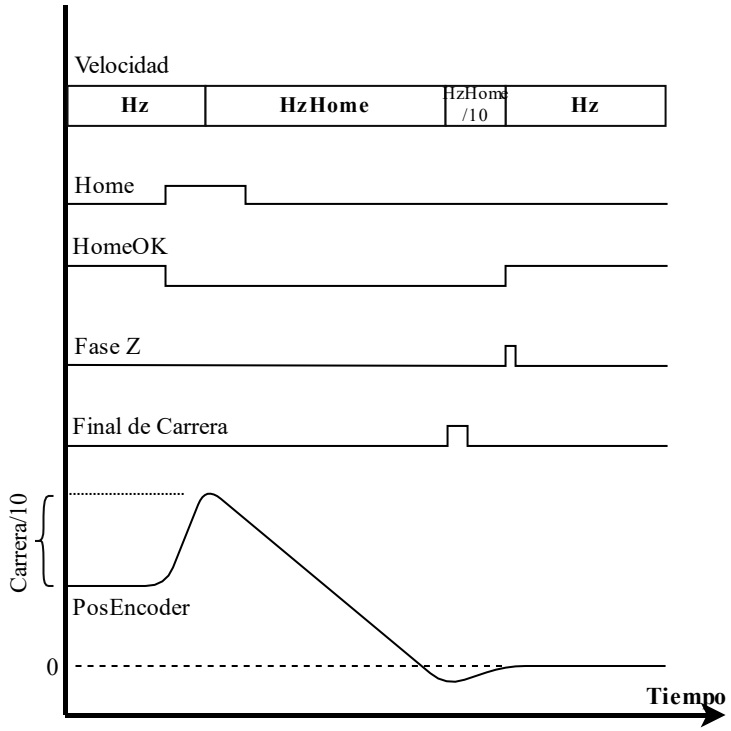



Figura 5-10. Cronograma de la función *MueveDispositivo*

## 5.6. MueveMaquina

### 5.6.1 Descripción

Esta es la función situada en la capa más externa de la librería, cuyo objetivo es sincronizar los ejes de la máquina para que puedan trabajar de forma conjunta.

El movimiento de los motores no es realizado de forma alternada, sino que, por cada nueva referencia o velocidad de desplazamiento introducida, se recalcula la frecuencia de los pulsos de comando de cada eje para que el punto de destino sea alcanzado al mismo tiempo por ambos. De esta forma, se consigue un movimiento bastante lineal y predecible. Asimismo, se dedica un nuevo argumento de salida para indicar que todas las articulaciones han alcanzado el punto de referencia individual.

Aquel eje que necesite más pulsos de comando para alcanzar la referencia funcionará con la frecuencia establecida en *Hz,* sin embargo, la frecuencia del otro será calculada a partir de la anterior y de la relación entre los incrementos de posición en cada dirección, y siempre será menor.

En el caso de que exista un retraso entre la introducción de una coordenada y la otra de la posición objetivo, se recomienda activar la entrada de detención durante ese intervalo de tiempo. Así se evita que sean consideradas como dos posiciones de referencia diferentes y el movimiento se realice en dos etapas.

Por otro lado, coordina la operación de búsqueda de *home.* Cada eje la realiza por separado y cuando termina uno comienza el otro, con el objetivo de eliminar posibles perturbaciones que afecten a la precisión de la operación.

Cabe destacar que las unidades de las medidas introducidas en los argumentos de un eje y otro también deben coincidir. El resto de funcionalidades anteriores se mantienen igual para cada eje.

### 5.6.2 Argumentos

- **Entradas/Salidas**

  1) **CanalPWM_Motor1 - (T_UNSIGN_CPT_BMX)**: Estructura del canal que genera los pulsos de comando para el *servomotor 1* y está configurado en modo modulación de ancho de pulso.
  2) **CanalContador_Motor1 - (T_SIGN_CPT_BMX)**: Estructura del canal que recibe las señales de dirección y de referencia dirigidas al *servomotor 1*, y está configurado como contador grande libre.
  3) **CanalEncoder_Motor1 - (T_SIGN_CPT_BMX)**: Estructura del canal conectado a dicho sensor del *servomotor 1,* y que está configurado como contador grande libre.
  4) **CanalPWM_Motor2 - (T_UNSIGN_CPT_BMX)**: Estructura del canal que genera los pulsos de comando para el *servomotor* 2 y está configurado en modo modulación de ancho de pulso.
  5) **CanalContador_Motor2 - (T_SIGN_CPT_BMX)**: Estructura del canal que recibe las señales de dirección y de referencia dirigidas al *servomotor 2*, y está configurado como contador grande libre.
  6) **CanalEncoder_Motor2 - (T_SIGN_CPT_BMX)**: Estructura del canal conectado a dicho sensor del *servomotor 2,* y que está configurado como contador grande libre.

- **Entradas**

  7) **RefPos - (ARRAY[0..1] OF DINT)**: Fija la posición, en las mismas unidades que los avances, que se desea alcanzar. El rango admisible es [-2147483648, +2147483647] para cada componente, y el elemento cero hace referencia al motor 1.
  8) **AvanceMotor - (ARRAY[0..1] OF INT)**: Relación entre el desplazamiento del eje y los pulsos de comando ejecutados por el *servomotor*. Es necesario ingresar este valor como entero mediante la modificación de sus unidades, definiendo de esta forma las unidades que maneja la función en el resto de entradas y salidas. El rango admisible es [1, 32767] para cada componente, y el elemento cero hace referencia al motor 1.

9) **AvanceEncoder - (ARRAY[0..1] OF INT)**: Relación entre el desplazamiento del eje y los pulsos generados por *encoder*. Es necesario ingresar este valor como entero mediante la modificación de sus unidades, definiendo de esta forma las unidades que maneja la función en el resto de entradas y salidas. El rango admisible es [1, 32767] para cada componente, y el elemento cero hace referencia al motor 1.

10) **Carrera - (ARRAY[0..1] OF DINT)**: Define la longitud del espacio de trabajo ubicado entre los finales de carrera en las mismas unidades que los avances. El rango admisible es (0, +2147483647] para cada componente, y el elemento cero hace referencia al motor 1.

11) **Hz - (UDINT)**: Fija la frecuencia en Hercios de los pulsos de comando generados para el eje que mayor distancia en pulsos tienen que recorrer. La velocidad del otro se calcula a partir de esta y siempre será menor. El rango admisible es [1, 4000] y en caso de incumplirlo se utilizará el último valor válido.

12) **HzHome - (UDINT)**: Fija la velocidad de desplazamiento durante las dos últimas etapas de la búsqueda de la posición de *home* en ambos ejes. El rango admisible es [1, 4000] y en caso de incumplirlo se utilizará la entrada *Hz.*

13) **STOP - (EBOOL)**: Si esta entrada se pone a nivel alto, se detiene la generación de pulsos de comando. De esta forma se consigue detener el movimiento ordenado de los *servomotores* inmediatamente. Cuando la entrada se desactiva, la función reanuda las consignas de movimiento por donde se ejecutaban antes de la interrupción. Su uso no altera los registros de las posiciones que se tiene de los ejes.

14) **FrenoEnReposo - (EBOOL)**: Esta entrada solamente influye en aquel motor que se encuentre en reposo, bien por coincidir la posición actual con su referencia, o bien, por estar la entrada *STOP* activada. Si está a nivel alto y se cumple alguna condición anterior, forzará al *servodriver* a mantener las bobinas del *servomotor* energizadas para bloquear el rotor e impedir su movimiento.

15) **Rectificar - (EBOOL)**: Un flanco de subida en esta entrada recalcula el valor actual del contador de pulsos de comando, o lo que es lo mismo, la posición del eje en pulsos de motor, por medio de la posición real que otorga el *encoder.* Solventando así problemas de sincronismo por movimientos involuntarios. Esta operación se ejecuta al unísono en ambas articulaciones y no se coordinan sus velocidades.

16) **Home - (EBOOL)**: Un flanco de subida en esta entrada inicia el proceso de búsqueda de la posición *home,* el primer eje que lo realiza es el correspondiente al *servomotor 1.* Se puede solicitar en cualquier momento.

17) **ResetAlarma - (EBOOL)**: Mientras esta entrada está activa, se envía una señal a los *servodrivers* para restaurar un posible estado de alarma y poder continuar con la operación.

- **Salidas**

7) **ESTADO_ERROR - (ARRAY[0..1] OF INT)**: Se mantiene el código de errores, para cada eje, de las funciones anteriores. El elemento cero hace referencia al motor 1.

8) **Sincronismo - (ARRAY[0..1] OF DINT)**: Vector que da a conocer la desincronización de los ejes mostrando las diferencias entre las posición reales y las virtuales. Las cantidades son expresadas en pulsos de comando y el elemento cero hace referencia al motor 1.

9) **PosEncoder - (ARRAY[0..1] OF DINT)**: Registro que almacena las posiciones reales del sistema en las mismas unidades que los avances. Sus valores se actualizan según el movimiento de dichos sensores, y el elemento cero hace referencia al motor 1.

10) **EnPosicion - (EBOOL)**: Cuando ambos ejes alcancen su posición de referencia objetivo, esta salida pasará a nivel alto. Útil para indicar el fin de la operación de posicionamiento coordinada.

11) **HomeOK - (EBOOL)**: Esta salida permanece a nivel bajo desde que se inicia la función hasta que se realiza la primera búsqueda de *home.* Tras esto, se mantendrá a nivel alto hasta que se solicite una nueva búsqueda: se desactivará para marcar el comienzo del proceso y se volverá a activar para indicar que ha sido completado en ambos ejes.

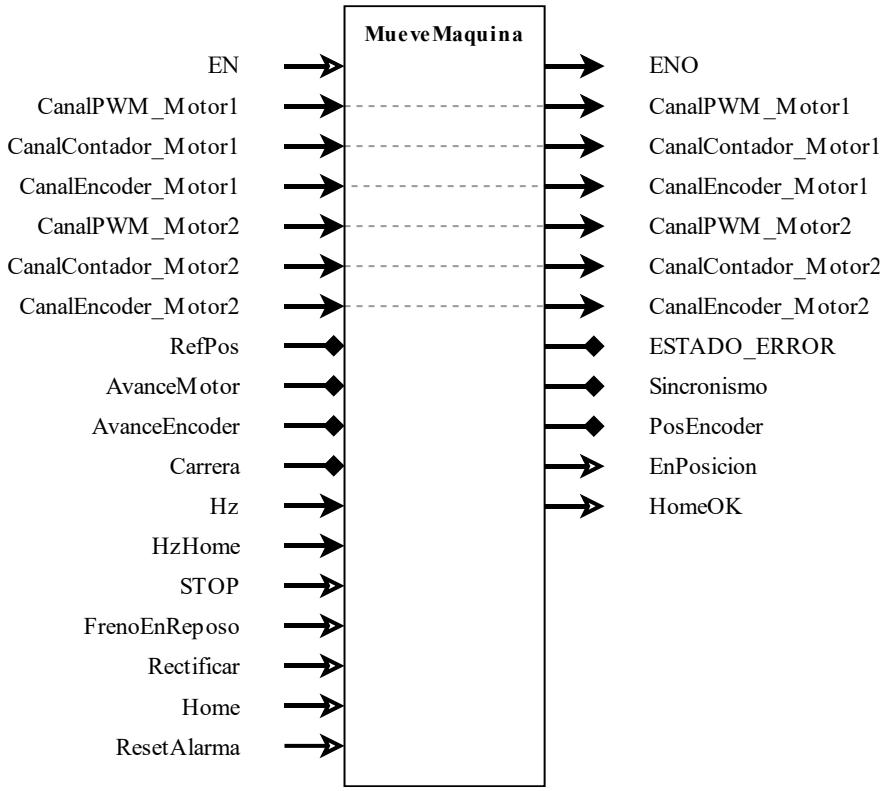



Figura 5-11. Bloque de la función *MueveMaquina*

### 5.6.3 Ejemplo

Si en un cierto momento, un motor tiene que ejecutar 400 pulsos de comando para alcanzar la posición de referencia, y el de la otra articulación solo 100 pulsos, el reparto de las velocidades quedará así: la frecuencia designada en *Hz* será aplicada al segundo motor por tener que ejecutar una mayor cantidad, y para el segundo se usará *Hz/4* con el objetivo de tardar el mismo tiempo.

### 5.6.4 Pantalla de Explotación

Para esta función se ha realizado una interfaz gráfica, Figura 5-12, que facilita su uso cuando se modifican las entradas de forma manual.

Está divida en tres partes: la central, con las señales que comparten ambos ejes, y los extremos derecho e izquierdo, con los argumentos específicos de cada uno. Los campos de textos claros son de escritura y los oscuros de lectura.

Los testigos luminosos existentes cambian de color según el estado de las variables asociadas.

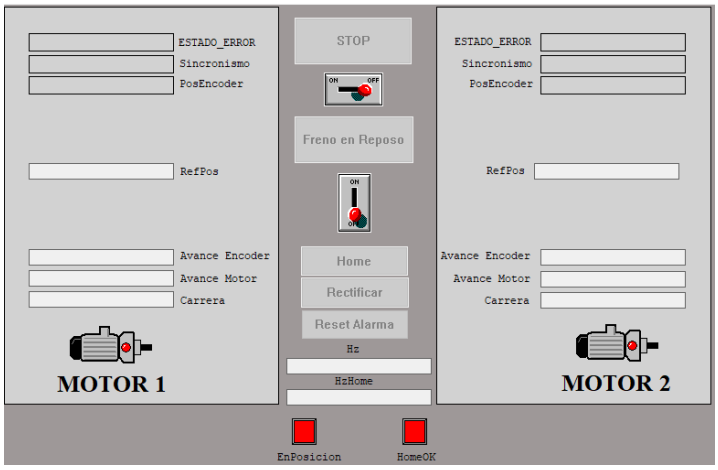



Figura 5-12. Pantalla de operador para *MueveMaquina*

# 6 PROTOCOLO DE PRUEBAS

A modo de verificación del correcto funcionamiento de la librería, y también, como exposición de los resultados, se muestra a continuación una serie de gráficas obtenidas directamente de la evolución de los argumentos de las funciones durante la ejecución en el pórtico.

## 6.1. MueveMotor

Se lleva a cabo una sucesión de operaciones para comprobar el correcto desempeño de la función.

Al comienzo, debido a que la frecuencia seleccionada es nula, se muestra un estado de error. Luego, tras introducir una correcta, se cambia la referencia de posición a 5000. Esto marca el inicio del movimiento, y tras finalizarse, el punto objetivo es alcanzado de forma correcta y precisa. En el transcurso de este desplazamiento aparece una breve pausa debido a la activación de la entrada STOP. Mientras se encuentra en posición, mantiene un estado de reposo.

Seguidamente, se prueba la operación de restauración de la posición del encoder. Para ello, se ha introducido la nueva posición, 2000, y se ha activado *Restaurar*. Se aprecia como el registro de posición es alterado instantáneamente con el nuevo valor, y acto seguido, la función trabaja para hacerlo coincidir con la referencia mediante el movimiento del eje.

En el último tramo, se elige una velocidad menor para realiza el mismo movimiento en sentido contrario. Como es de esperar, la pendiente de la evolución de la posición respecto al tiempo es menor.

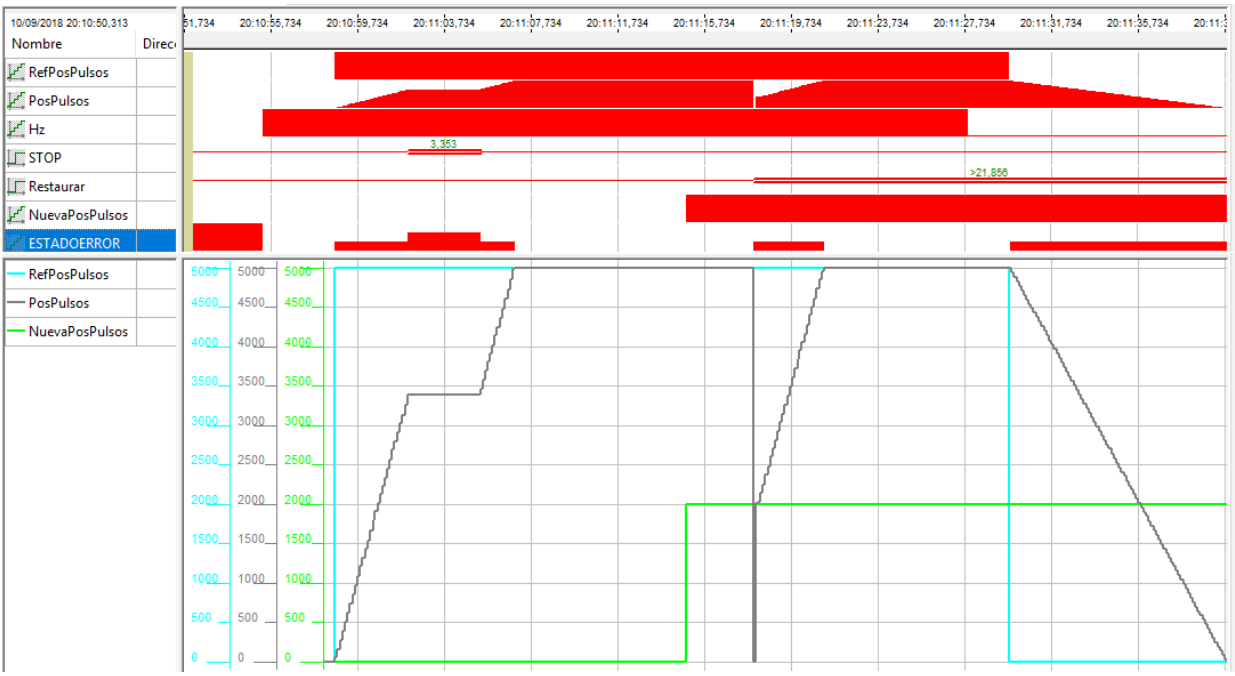



Figura 6-1. Gráfica de evolución de la ejecución de *MueveMotor*

## 6.2. MueveAcoplamiento

En este caso se presenta la evolución cuando se utilizan unidades de medidas de longitud para expresar las posiciones.

Usando valores de 2 y 1 milésimas de centímetro por pulso para los avances del motor y *encoder* respectivamente, se consiguen los siguientes resultados. En primer lugar, la posición 10 cm es alcanzada con bastante precisión, manteniendo muy bajo el valor de sincronismo. Luego, se realiza con las manos un movimiento no ordenado del eje, lo cual queda reflejado por el error número 7 y por el incremento de la salida de sincronismo.

Para corregir la desviación se hace uso de la entrada *Rectificar*. Con ella, se recalcula la posición virtual a partir de la real, y al realizarlo y advertir que no coincide con la referencia, el dispositivo se desplaza para alcanzarla de nuevo. Esta vez con un poco más de margen, debido a que las relaciones de avance están calculadas de forma experimental y no son todo lo precisas que deberían.

Finalmente, se propone un cambio de la posición del *encoder* mediante la entrada *NuevaPosEncoder*. Tras llevarse a cabo, el sistema comienza a moverse de nuevo, debido a que la posición de referencia ahora corresponde con otro punto del espacio que debe ser alcanzado.

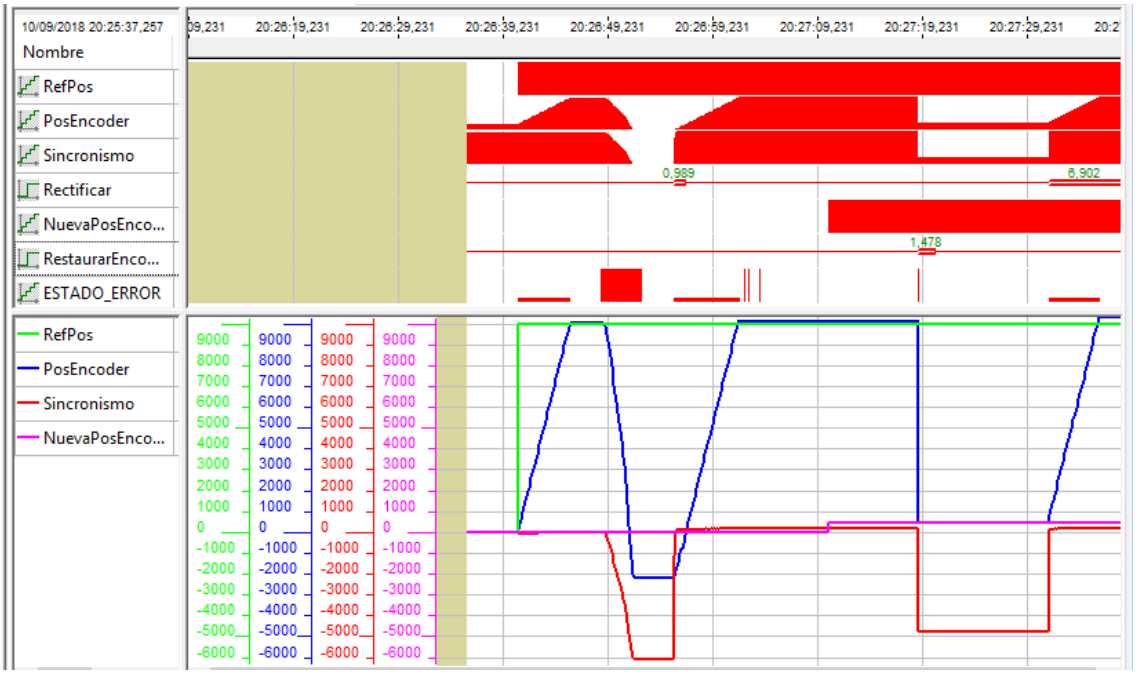



Figura 6-2. Gráfica de evolución de la ejecución de *MueveAcoplamiento*

## 6.3. MueveMaquina

En este caso se quiere poner de manifiesto la sincronización de los dos ejes de la máquina.

La primera operación consiste en una búsqueda de *home*. Es ordenada mediante la activación de la entrada *Home,* y acto seguido, un eje comienza a moverse para inicializarse. Tras acabar, lo hace el otro. Cuando el segundo también termina, *HomeOk* indica el fin del proceso y la máquina está calibrada respecto a este punto.

Luego, se introduce una posición de referencia, se hace con la entrada *STOP* activada para que el movimiento no se inicie hasta ser definidas las dos componentes. Tras habilitar el giro, es posible apreciar como la velocidad de cada eje es adaptada para que ambos concluyan el desplazamiento en el mismo tiempo.

Para acabar, se vuelve a realizar un movimiento a mano no ordenado, para posteriormente usar la entrada *Rectificar* y comprobar que ambas articulaciones vuelven a la posición previa.

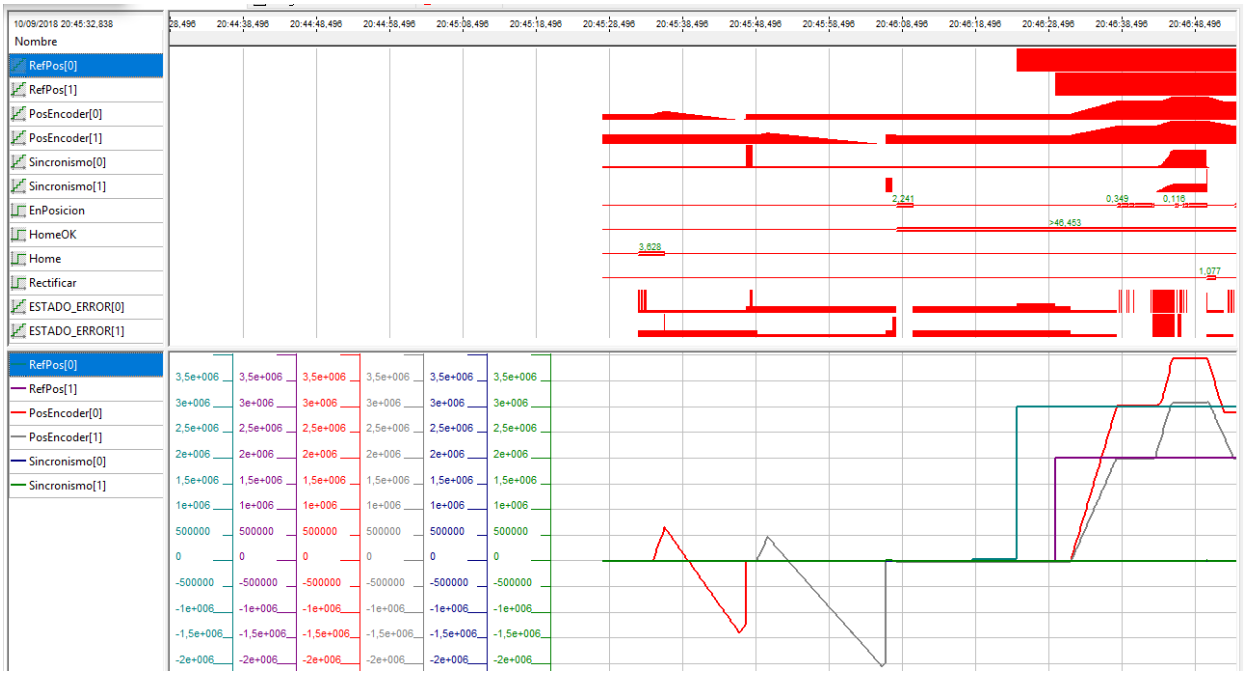



Figura 6-3. Gráfica de evolución de la ejecución de *MueveMaquina*

# 7 MANUAL DE USO

En esta sección se procede a explicar como realizar la puesta a punto de los diferentes elementos. Esto es un aspecto fundamental para poder hacer uso del pórtico o de sus mismas funciones en otros equipos equivalentes.

## 7.1. Configuración de los Módulos del Autómata

El primer paso es crear un nuevo proyecto, para ello, en la barra de tareas se debe seleccionar *"Fichero - Nuevo" (1)* y en la ventana que se abre buscar la CPU *(2)* y bastidor empleado *(3)*, BMX P34 2020 y BMX XBP 0800 *(2)*.

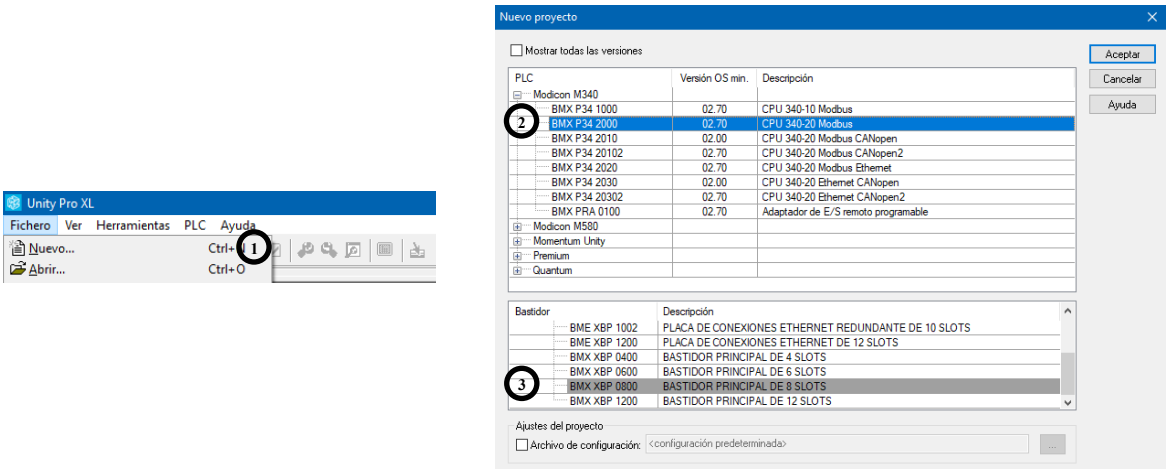



Figura 7-1. Configuración PLC 1

Una vez abierto el proyecto es necesario configurar el autómata. En el *Explorador de proyectos* accedemos a *"Configuración - BusPLC" (4)* y se abrirá una nueva venta. En esta se representa el bastidor con los *slots* disponibles para añadir tarjetas. Guiándonos por la instalación sobre el pórtico es necesario añadir la BMX EHC 0200 en los huecos 2, 3 y 4. Esto se consigue haciendo doble clic sobre el *slot* deseado *(5)* y acto seguido seleccionado el modelo adecuado dentro de la sección *"Conteo" (6).*

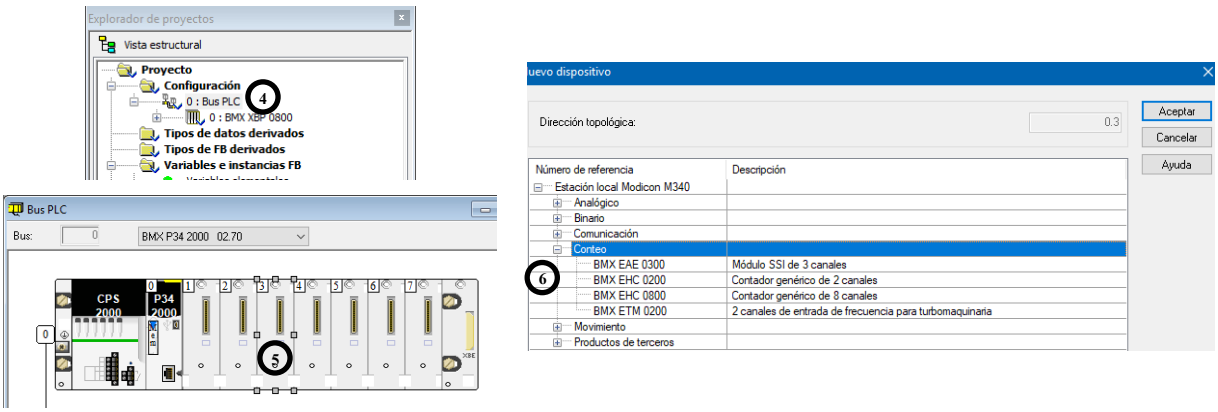



Figura 7-2. Configuración PLC 2

Tras ser agregadas hay que programarlas, para acceder a una basta con un doble clic sobre ella *(7)* de tal forma que se abre una ventana. Lo primero es asignar a cada canal un modo de funcionamiento, esto se hace seleccionando un canal en la vista estructural de la izquierda *(8)* y eligiéndolo en la lista desplegable llamada *"Función" (9).* Cual es cada uno se determina fácilmente a partir de la Figura 4-1. La tarjeta central en el *slot 3* tiene ambos canales en *"Modo contador libre de 32 bits"* y se configuran seleccionando los valores adecuados de la lista de parámetros que aparece a la derecha *(10)* según se exponen en la columna "*Tarjeta 2 Canal 0"* de la Tabla 4-1. Por otro lado, las dos tarjetas restantes poseen la misma configuración, el canal 0 en *"Modo modulación de ancho de pulsos"* y el canal 1 en *"Modo contador libre de 32 bits",* y son programados según las otras dos columnas correspondientes de la misma tabla anterior, Tabla 4-1.

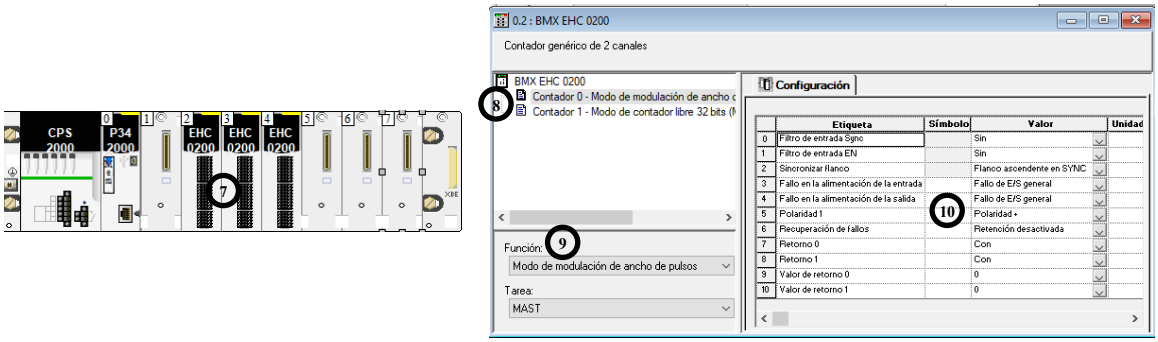



Figura 7-3. Configuración PLC 3

Seguidamente, se crean las estructuras que definen el estado interno de cada módulo, y que contienen las variables que permiten la interacción con estos dispositivos. Se realiza accediendo a *"BMX EHC 0200" (11)* en la vista estructural y a la pestaña *"Objetos de E/S" (12).* Se comienza marcando la casilla *"%CH" (13)* y pulsando *"Actualizar cuadrícula" (14).* Una vez activada la tabla de la derecha se selecciona la primera fila (15), esta corresponde con la estructura general de la tarjeta, y se le asigna un nombre pulsando *"crear" (17)* tras escribirlo en el cuadro de texto *"Prefijo para nombre" (16)*. Luego, se repiten los mismos pasos pero seleccionando la segunda fila *(18)* y, posteriormente, la tercera *(19)*. Estas corresponden al canal 0 y 1 respectivamente.

Se recomienda el uso de nombres intuitivos para cada estructura, por ejemplo: nombre de la tarjeta, "*Tarjeta_ENCODER"* o *"Tarjeta_MOTOR_n";* nombre del canal generado de pulsos, *"MOTOR_n_GENERADOR";* nombre del canal contador de pulsos, *"MOTOR_n_CONTADOR";* nombre del canal supervisor del *encoder*, *"MOTOR_n_ENCODER"*. La letra *n* representa un número particular de cada eje.

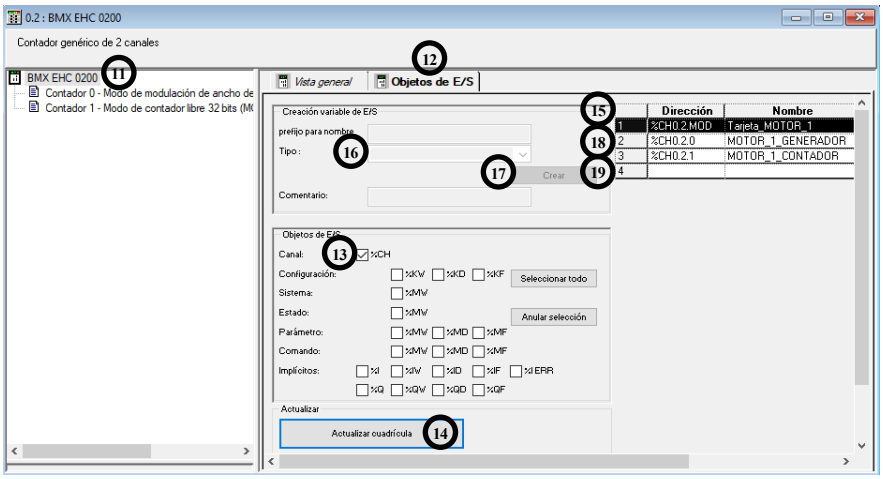



Figura 7-4. Configuración PLC 4

La configuración se realizaría del mismo modo si en un futuro se instalasen más tarjetas para controlar un tercer motor. Bastaría con seguir los pasos expuestos y elegir los valores adecuados según la misión que desempeñe cada canal extra añadido.

## 7.2. **Importación de la Librería**

Para añadir al proyecto el conjunto de bloques derivados funcionales es necesario la importación de los archivos que acompañan esta memoria.

Una vez el proyecto está abierto, se debe hacer clic derecho en *"Tipos de FB derivados" (1),* situado en el *Explorador de proyectos,* y luego se pulsa en *"Importar" (2).* Se abrirá una nueva ventana en la cual hay que buscar en el ordenador el archivo *"librería_pórtico.xdb" (3)* e importarlo *(4).*

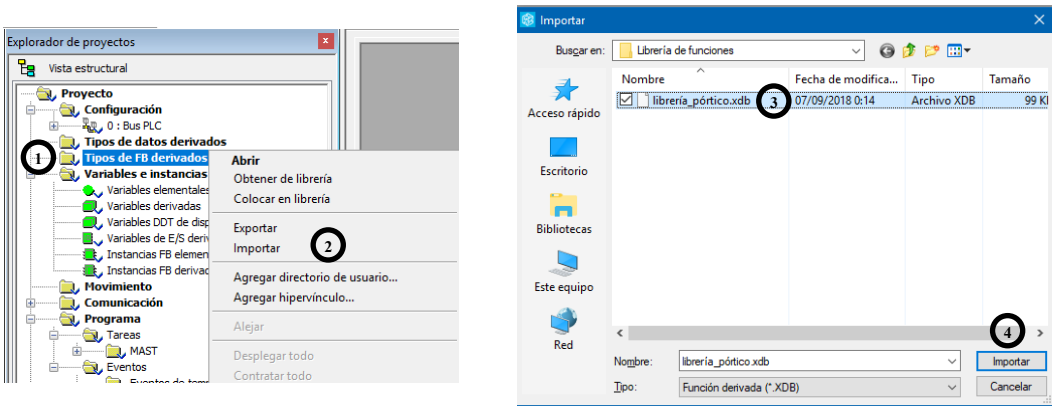



Figura 7-5. Importación de librería 1

Las funciones estarán disponibles *(5)* para ser usadas en cualquier sección del proyecto una vez se haya realizado esta operación. Para incluir alguna de ellas basta con abrir el *"Asistente de entrada FFB" (6)* y buscarla por su nombre *(7).*

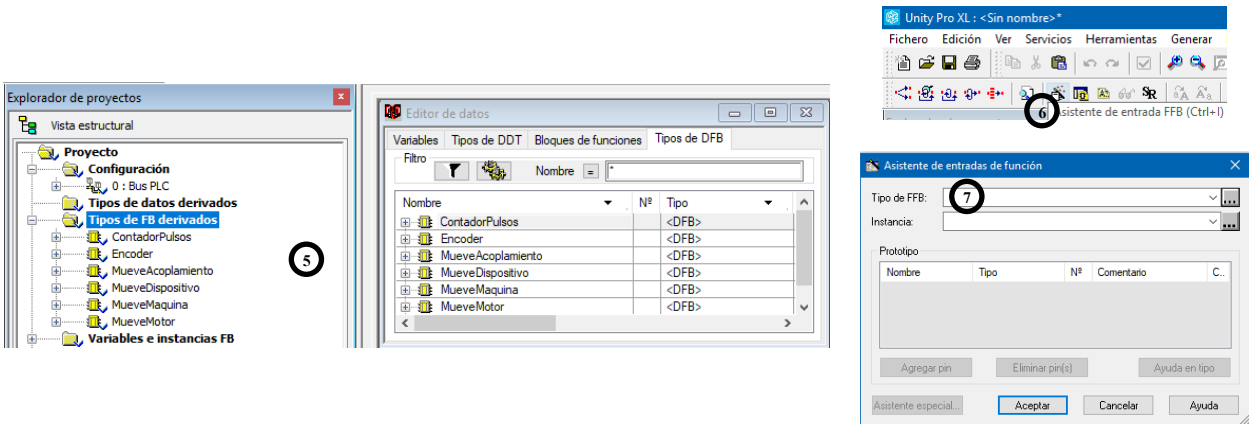



Figura 7-6. Importación de librería 2

Para la incorporación de la pantalla de explotación se procede del mismo modo. Esta vez se hace clic derecho sobre *"Pantallas de operador" (8)* y posteriormente en *"Importar"* (9). Se vuelve a abrir una ventana de exploración en la que se debe buscar el archivo *"hmi_pórtico.xcr" (10)* e importarlo *(11)*.

De este modo, la pantalla pasa a formar parte del proyecto *(12)* y puede ser utilizada para la supervisión y control manual del pórtico.

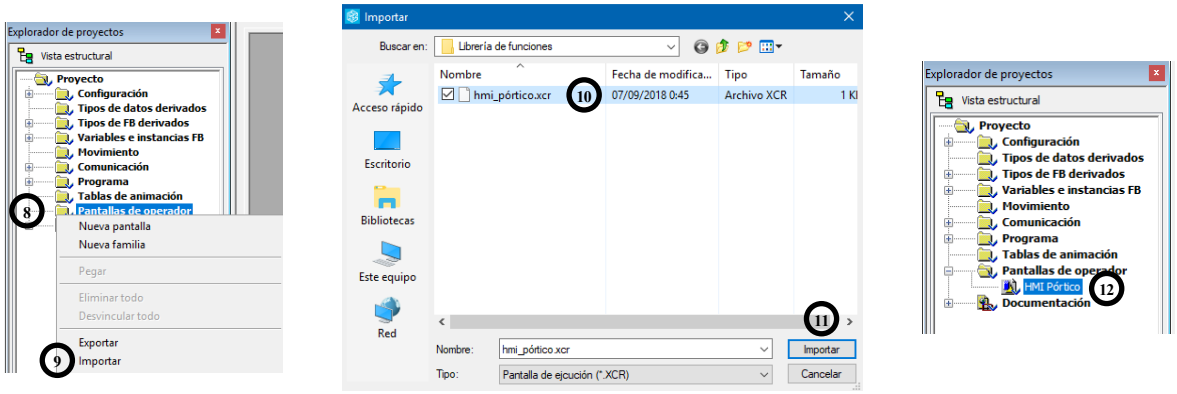



Figura 7-7. Importación de librería 3

## 7.3. **Configuración de los Motores**

Se realiza por medio del *software* SigmaWin expuesto previamente. En primer lugar, se debe realizar la conexión por medio de un cable serie con conectores RS-232 entre el PC y el conector CN3 del *servodriver*.

Tras ejecutar el programa aparecerá una ventana para la selección de la unidad conectada, se debe marcar la deseada *(1)* y aceptar *(2)*. Aparecerá la vista principal del programa con los valores actuales de los parámetros. Para modificarlos tan solo es necesario hacer doble clic sobre uno de ellos (3) e introducir el deseado. Una vez corregidos los requeridos, se procede a enviarlos al dispositivo usando *"SendAll" (4)*. Los cambios serán efectivos una vez se retire durante unos segundos la alimentación al *servodriver*.

Para realizar la conexión con la otra unidad es necesario cerrar la ya existente en el icono *(5)*, y tras conectar el cable, volver a buscar mediante el icono *(6)*.

Junto a la memoria se incluye un par de archivos de configuración, cada uno para un motor. De esta forma no es necesario la modificación a mano de los distintos valores. Para hacer uso de ellos, tras la conexión, se deben abrir seleccionando en la barra de herramientas *"File - Import Parameter File" (7).*

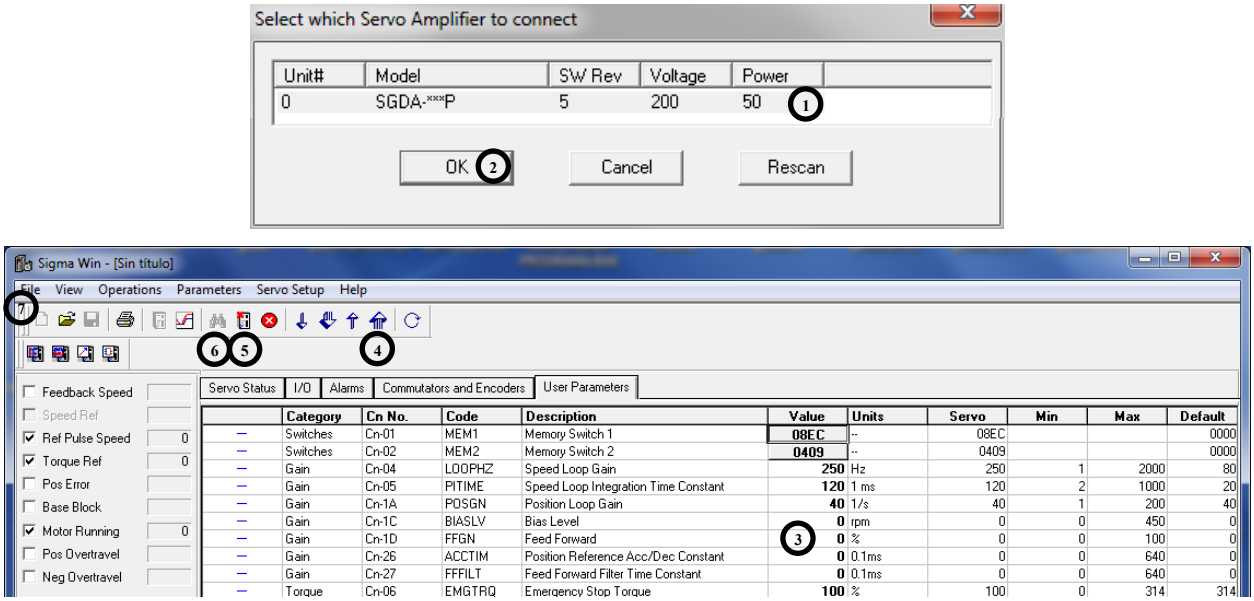



Figura 7-8. Configuración del *servodriver.*

# 8 LÍNEAS FUTURAS

El trabajo realizado sobre esta estación de la célula de fabricación puede ser ampliado siguiendo una multitud de caminos diferentes. Algunos de ellos basados en el trabajo realizado y expuesto en este proyecto, y otros abordando otras vías de estudio. A continuación se exponen algunos ejemplos.

Una buena opción sería implementar el tercer eje y dotar así a la máquina de su espacio de trabajo completo. Para el aspecto de control se puede usar la librería desarrollada, siempre y cuando el sistema de propulsión añadido sea similar al ya instalado. En el caso de usar otros equipos como motores paso a paso se necesitarían realizar modificaciones menores según la interfaz del *driver* que lo controle.

Asimismo, y para hacer el pórtico totalmente funcional, lo ideal sería instalar un efector final que pueda desarrollar tareas de manipulación. El mismo PLC usado para el control en posición podría ser utilizado, con las ampliaciones de *hardware* oportunas, para gobernar dicho manipulador.

Otra posible ampliación consistiría en usar las funciones realizadas como herramientas de bajo nivel para implementar un intérprete de lenguaje G, o algún otro lenguaje más simple de desarrollo propio específico para este dispositivo. De esta forma, se conseguiría una máquina de control numérico muy flexible y adaptable a una infinidad de trabajos.

El lenguaje de programación G se basa en una lista de instrucciones que indican qué hacer y cómo por medio de la definición de posiciones, trayectorias, velocidades de paso, zonas de activación de la herramienta, etc. Hoy en día es ampliamente usado desde máquinas-herramientas hasta impresoras 3D y existen programas que lo generan a partir de modelos simulados de la tarea a realizar.

También se podría tratar de centralizar el funcionamiento de toda la célula de fabricación mediante programas de supervisión, gestión y optimización, ejecutándose en un autómata central comunicado con cada uno de los autómatas de cada puesto. De esta manera, se podrían implementar tareas de producción que abarquen más de un puesto y, además, sincronizar todos ellos para que trabajen al unísono.

A la centralización anterior también se le podría dar un enfoque desde el lado de la seguridad y el mantenimiento. Con un PLC central conectado al resto se podría supervisar todos los sensores de la instalación para detectar comportamientos anómalos, o bien, problemas de seguridad que requieran una notificación o una parada de la operación.

# Anexo I: Código de las Funciones

A continuación, se expone el código de las funciones que componen la librería junto a las variables privadas utilizadas. La descripción de las entradas y salidas se puede encontrar en el Apartado 5 Librería de Funciones de esta memoria.

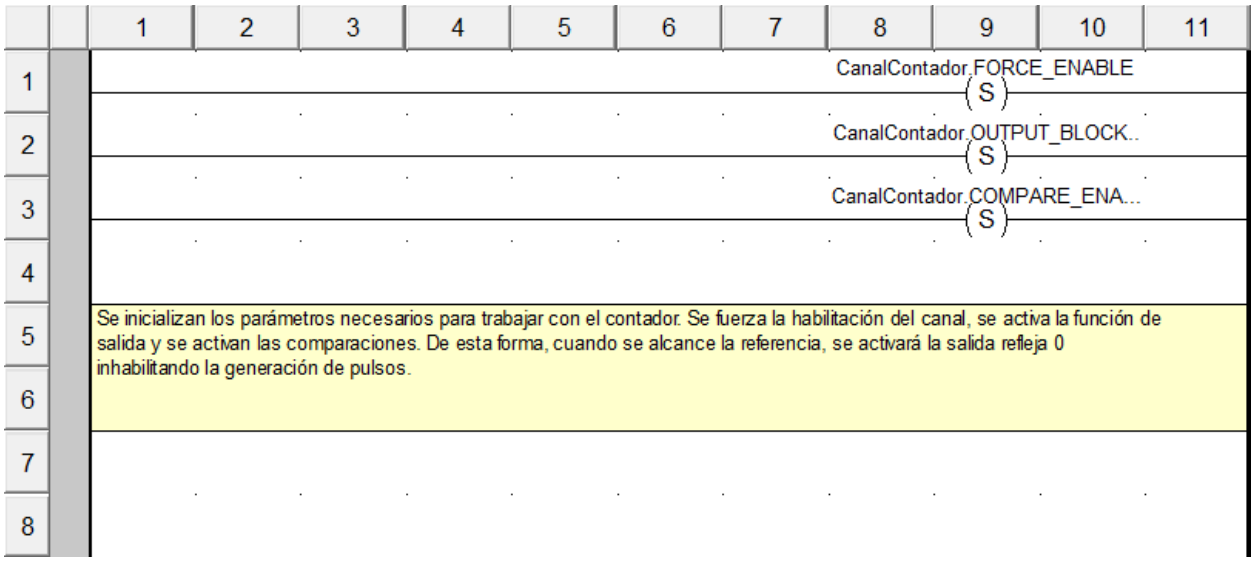
## 1. ContadorPulsos

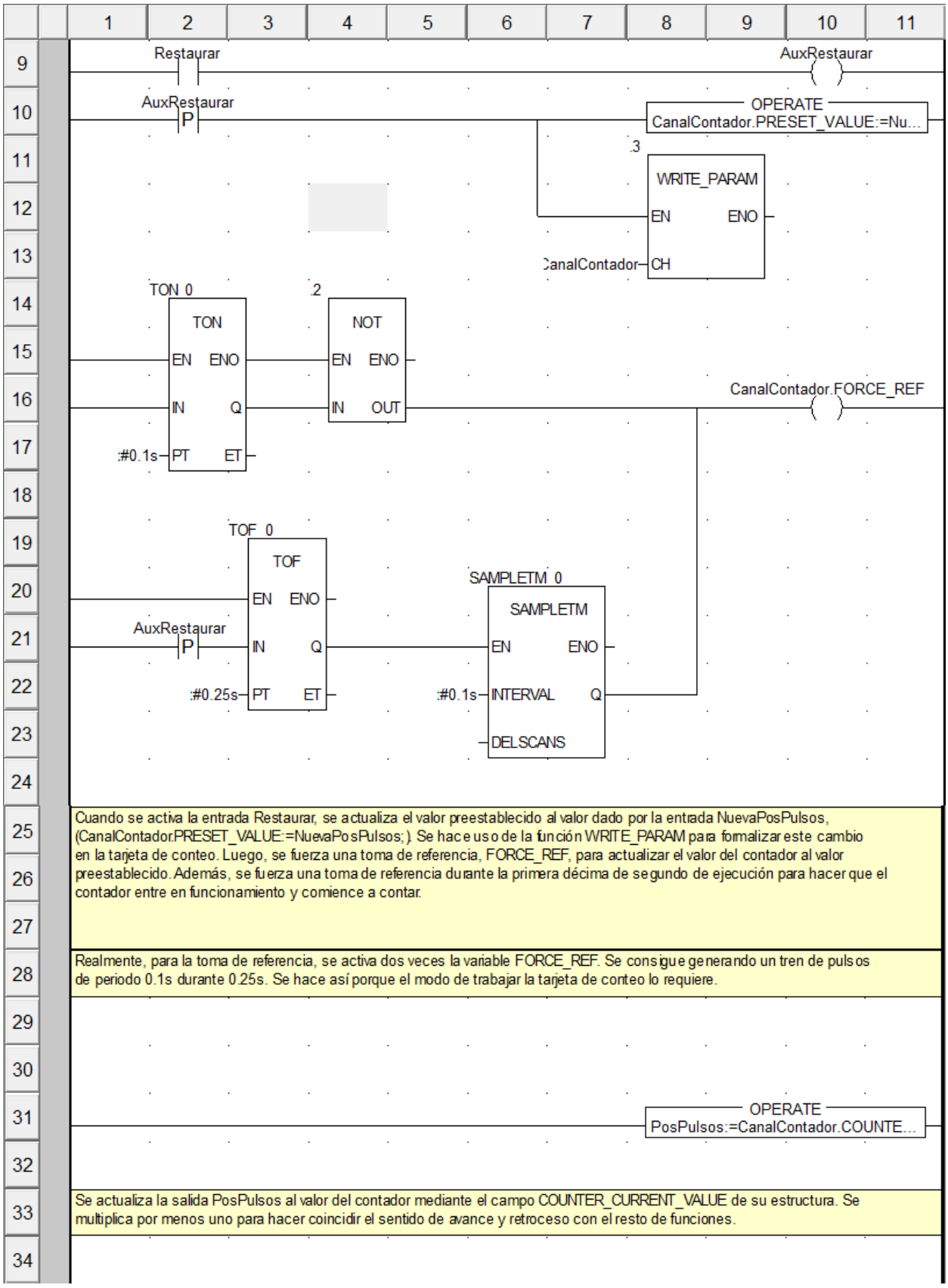
Tabla I-1. Variables privadas de *ContadorPulsos*

| Nombre | Tipo | Comentario |
|---|---|---|
| AuxRestaurar | EBOOL | Para la detección de flancos en la entrada *Restaurar* |
| TON_0 | TON | |
| TOF_0 | TOF | |
| SAMPLETM_0 | SAMPLETM | |

Tabla I-2. Etiquetas truncadas de *ContadorPulsos*

| Etiqueta | Posición |
|---|---|
| CanalContador.OUTPUT_BLOCK_0_ENABLE | (8, 2) |
| CanalContador.PRESET_VALUE:=NuevaPosPulsos*-1; | (8, 10) |
| PosPulsos:=CanalContador.COUNTER_CURRENT_VALUE*-1; | (8, 31) |




Restaurar
AuxRestaurar
AuxRestaurar
P
OPERATE
CanalContador.PRESET_VALUE:=Nu...
.3
WRITE_PARAM
EN
ENO
CanalContador
CH
TON 0
TON
EN
ENO
IN
Q
#0.1s
PT
ET
.2
NOT
EN
ENO
IN
OUT
CanalContador.FORCE_REF
TOF 0
TOF
EN
ENO
AuxRestaurar
P
IN
Q
#0.25s
PT
ET
SAMPLETM 0
SAMPLETM
EN
ENO
#0.1s
INTERVAL
Q
DELSCANS
Cuando se activa la entrada Restaurar, se actualiza el valor preestablecido al valor dado por la entrada NuevaPosPulsos, (CanalContador.PRESET_VALUE:=NuevaPosPulsos;). Se hace uso de la función WRITE_PARAM para formalizar este cambio en la tarjeta de conteo. Luego, se fuerza una toma de referencia, FORCE_REF, para actualizar el valor del contador al valor preestablecido. Además, se fuerza una toma de referencia durante la primera décima de segundo de ejecución para hacer que el contador entre en funcionamiento y comience a contar.
Realmente, para la toma de referencia, se activa dos veces la variable FORCE_REF. Se consigue generando un tren de pulsos de periodo 0.1s durante 0.25s. Se hace así porque el modo de trabajar la tarjeta de conteo lo requiere.
OPERATE
PosPulsos:=CanalContador.COUNTE...
Se actualiza la salida PosPulsos al valor del contador mediante el campo COUNTER_CURRENT_VALUE de su estructura. Se multiplica por menos uno para hacer coincidir el sentido de avance y retroceso con el resto de funciones.

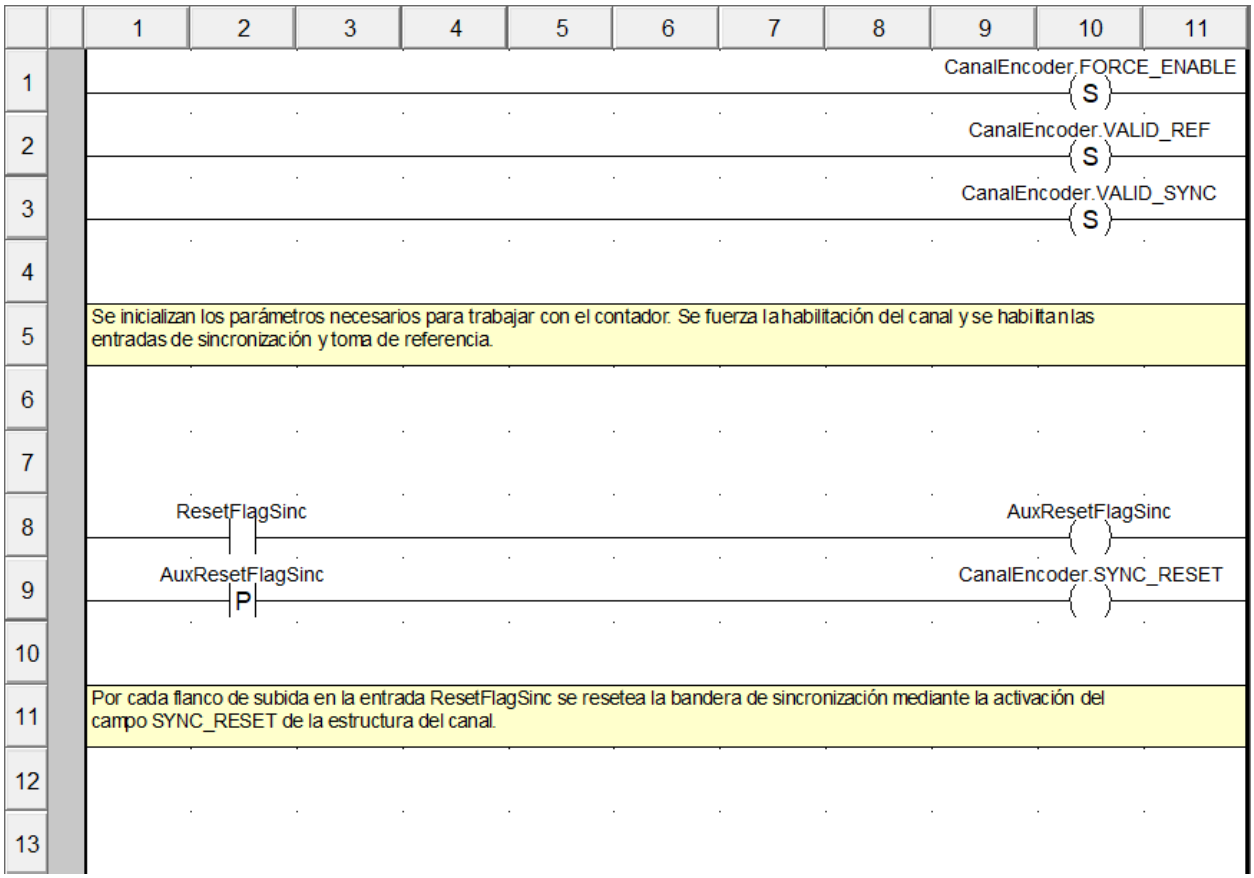
# 2. Encoder

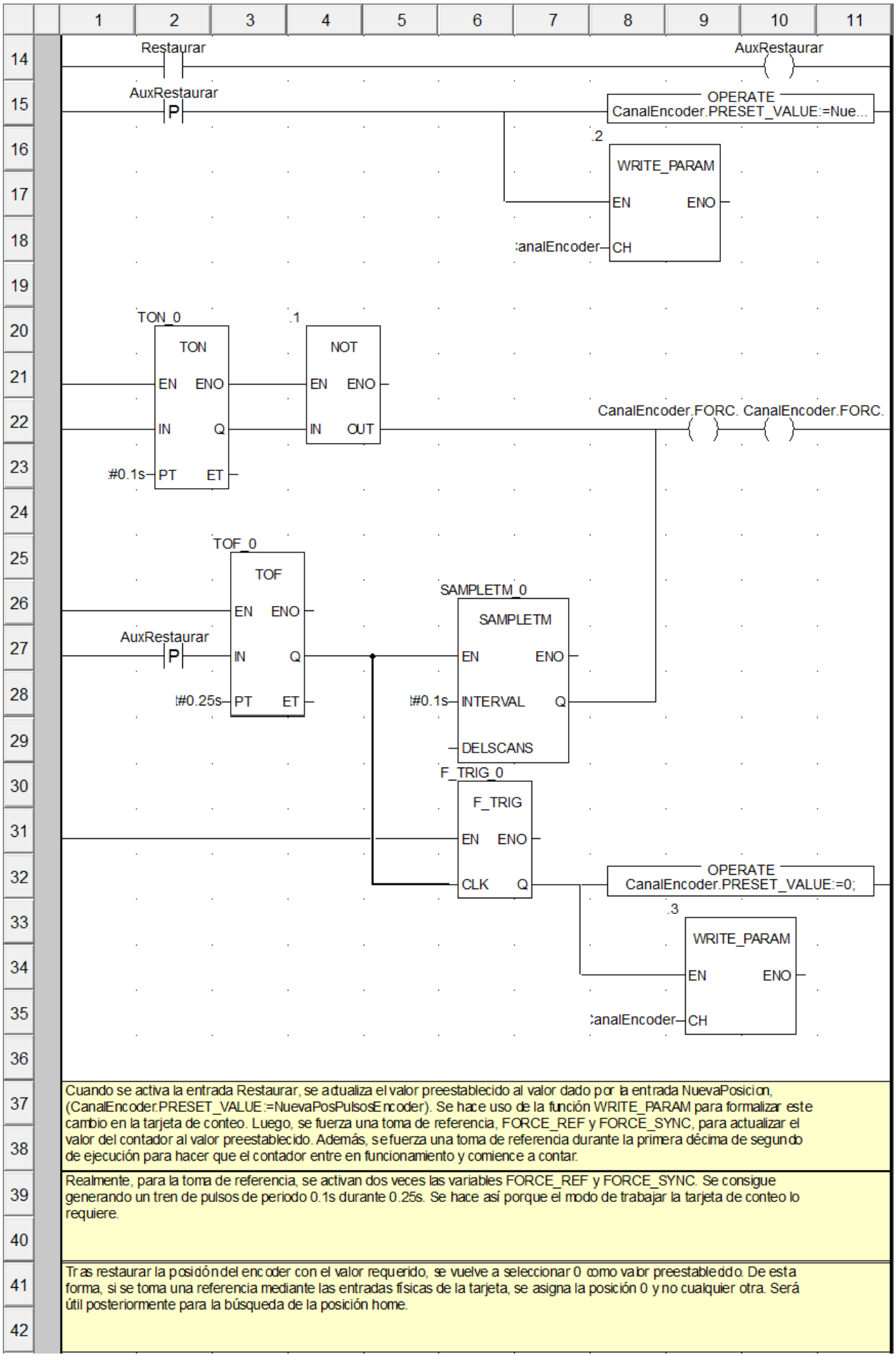
Tabla I-3. Variables privadas de *Encoder*

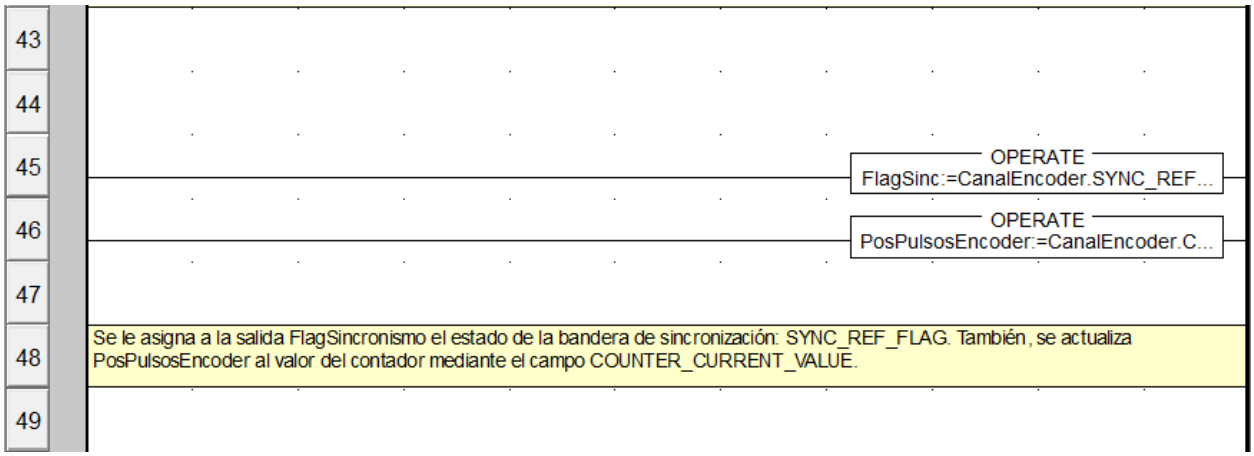
| Nombre | Tipo | Comentario |
|---|---|---|
| AuxResetFlagSinc | EBOOL | Para la detección de flancos en la entrada ResetFlagSinc |
| AuxRestaurar | EBOOL | Para la detección de flancos en la entrada Restaurar |
| TON_0 | TON | |
| TOF_0 | TOF | |
| SAMPLETM_0 | SAMPLETM | |
| F_TRIG_0 | F_TRIG | |

Tabla I-4. Etiquetas truncadas de *Encoder*

| Etiqueta | Posición |
|---|---|
| CanalEncoder.FORCE_REF | (10, 22) |
| CanalEncoder.FORCE_SYNC | (8, 22) |
| CanalEncoder.PRESET_VALUE:=NuevaPosPulsosEncoder; | (8, 15) |
| FlagSinc:=CanalEncoder.SYNC_REF_FLAG; | (8, 45) |
| PosPulsosEncoder:=CanalEncoder.COUNTER_CURRENT_VALUE; | (8, 46) |




Restaurar
AuxRestaurar
AuxRestaurar
P
OPERATE
CanalEncoder.PRESET_VALUE:=Nue...
.2
WRITE_PARAM
EN
ENO
analEncoder
CH
TON_0
TON
EN
ENO
IN
Q
#0.1s
PT
ET
.1
NOT
EN
ENO
IN
OUT
CanalEncoder.FORC.
CanalEncoder.FORC.
TOF_0
TOF
EN
ENO
AuxRestaurar
P
IN
Q
#0.25s
PT
ET
SAMPLETM_0
SAMPLETM
EN
ENO
#0.1s
INTERVAL
Q
DELSCANS
F_TRIG_0
F_TRIG
EN
ENO
CLK
Q
OPERATE
CanalEncoder.PRESET_VALUE:=0;
.3
WRITE_PARAM
EN
ENO
analEncoder
CH
Cuando se activa la entrada Restaurar, se actualiza el valor preestablecido al valor dado por la entrada NuevaPosicion, (CanalEncoder.PRESET_VALUE:=NuevaPosPulsosEncoder). Se hace uso de la función WRITE_PARAM para formalizar este cambio en la tarjeta de conteo. Luego, se fuerza una toma de referencia, FORCE_REF y FORCE_SYNC, para actualizar el valor del contador al valor preestablecido. Además, se fuerza una toma de referencia durante la primera décima de segundo de ejecución para hacer que el contador entre en funcionamiento y comience a contar.
Realmente, para la toma de referencia, se activan dos veces las variables FORCE_REF y FORCE_SYNC. Se consigue generando un tren de pulsos de periodo 0.1s durante 0.25s. Se hace así porque el modo de trabajar la tarjeta de conteo lo requiere.
Tras restaurar la posición del encoder con el valor requerido, se vuelve a seleccionar 0 como valor preestablecido. De esta forma, si se toma una referencia mediante las entradas físicas de la tarjeta, se asigna la posición 0 y no cualquier otra. Será útil posteriormente para la búsqueda de la posición home.


43
44
45
OPERATE
FlagSinc:=CanalEncoder.SYNC_REF...
46
OPERATE
PosPulsosEncoder:=CanalEncoder.C...
47
48
Se le asigna a la salida FlagSincronismo el estado de la bandera de sincronización: SYNC_REF_FLAG. También, se actualiza PosPulsosEncoder al valor del contador mediante el campo COUNTER_CURRENT_VALUE.
49

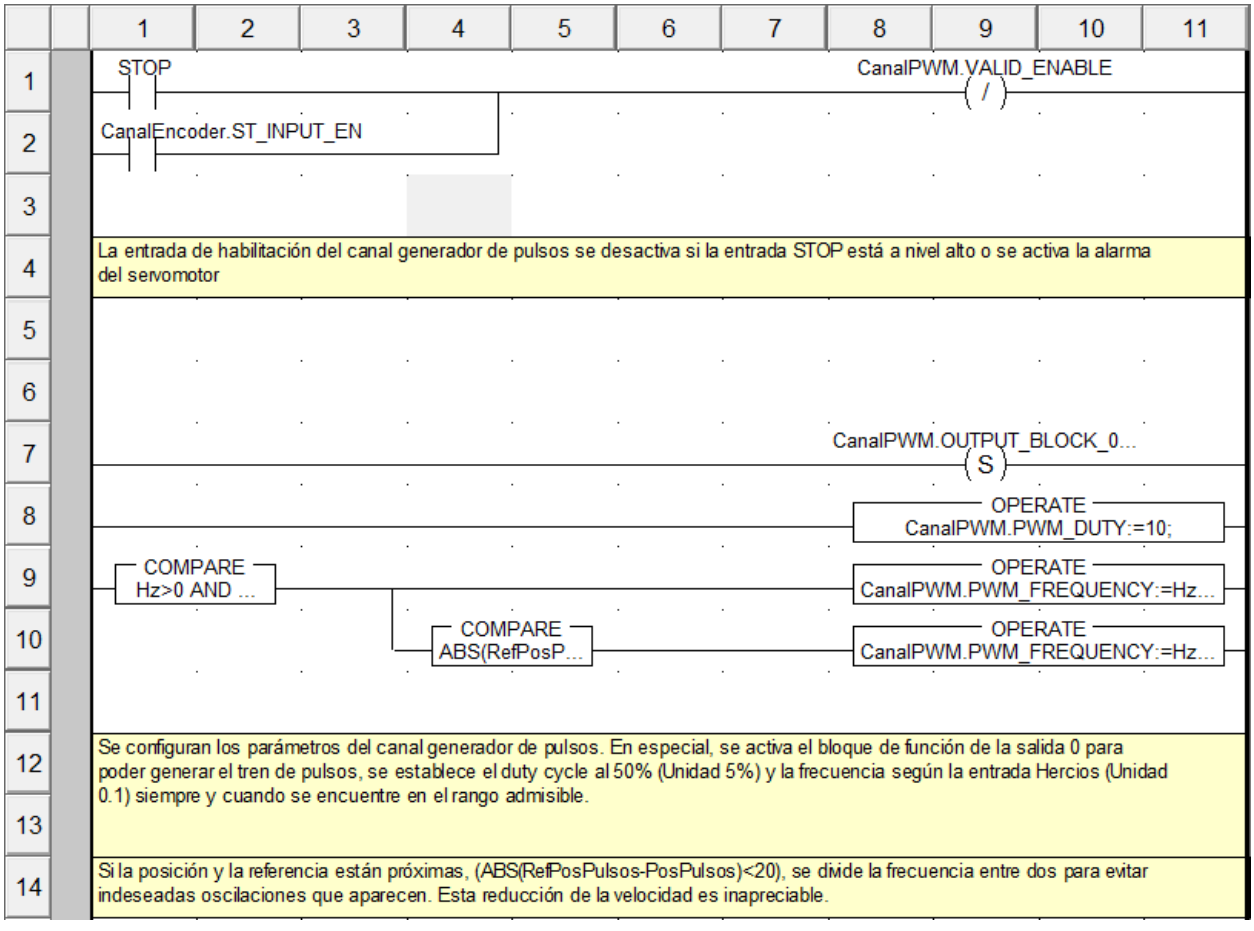
## 3. MueveMotor

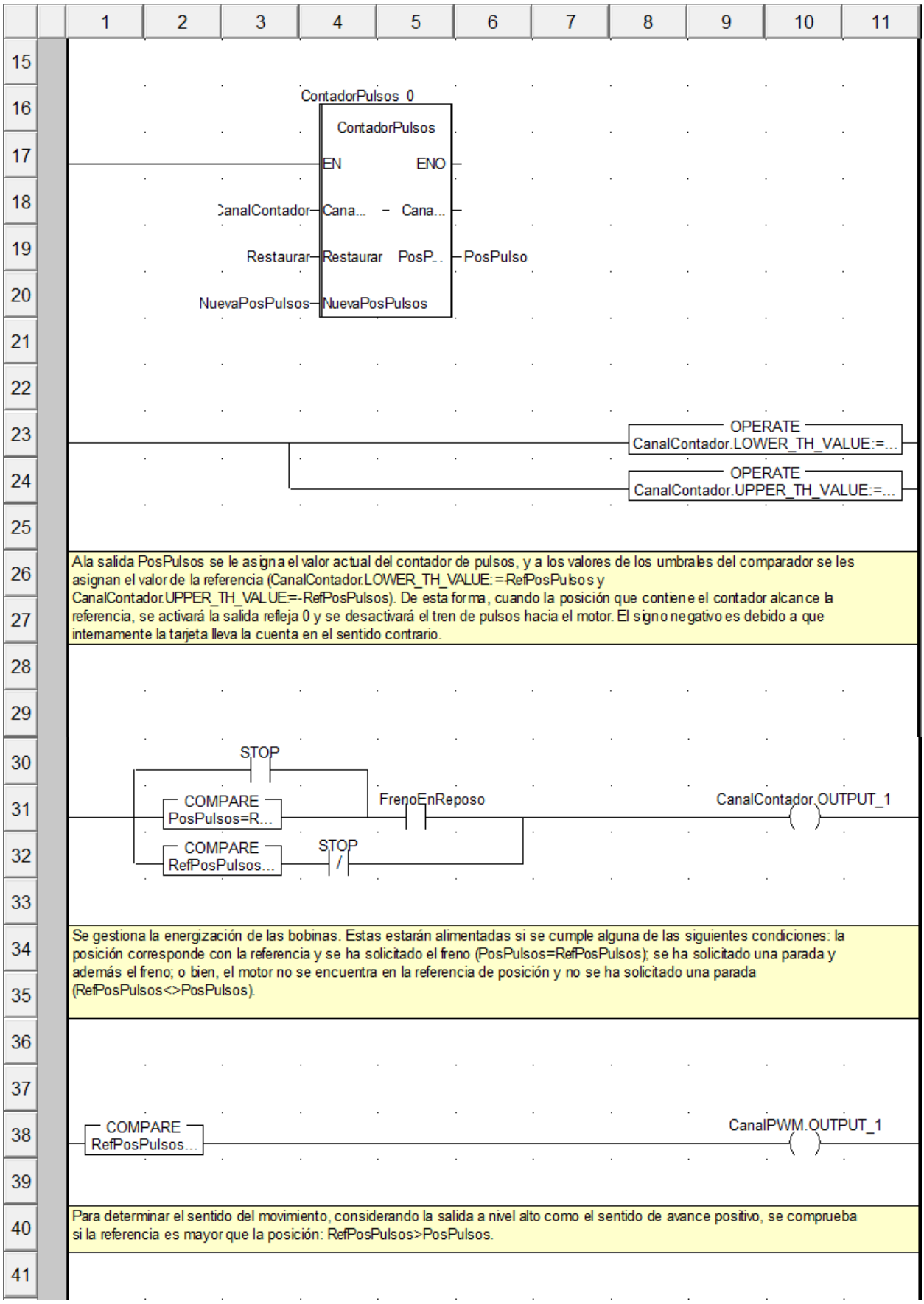
Tabla I-5. Variables privadas de *MueveMotor*

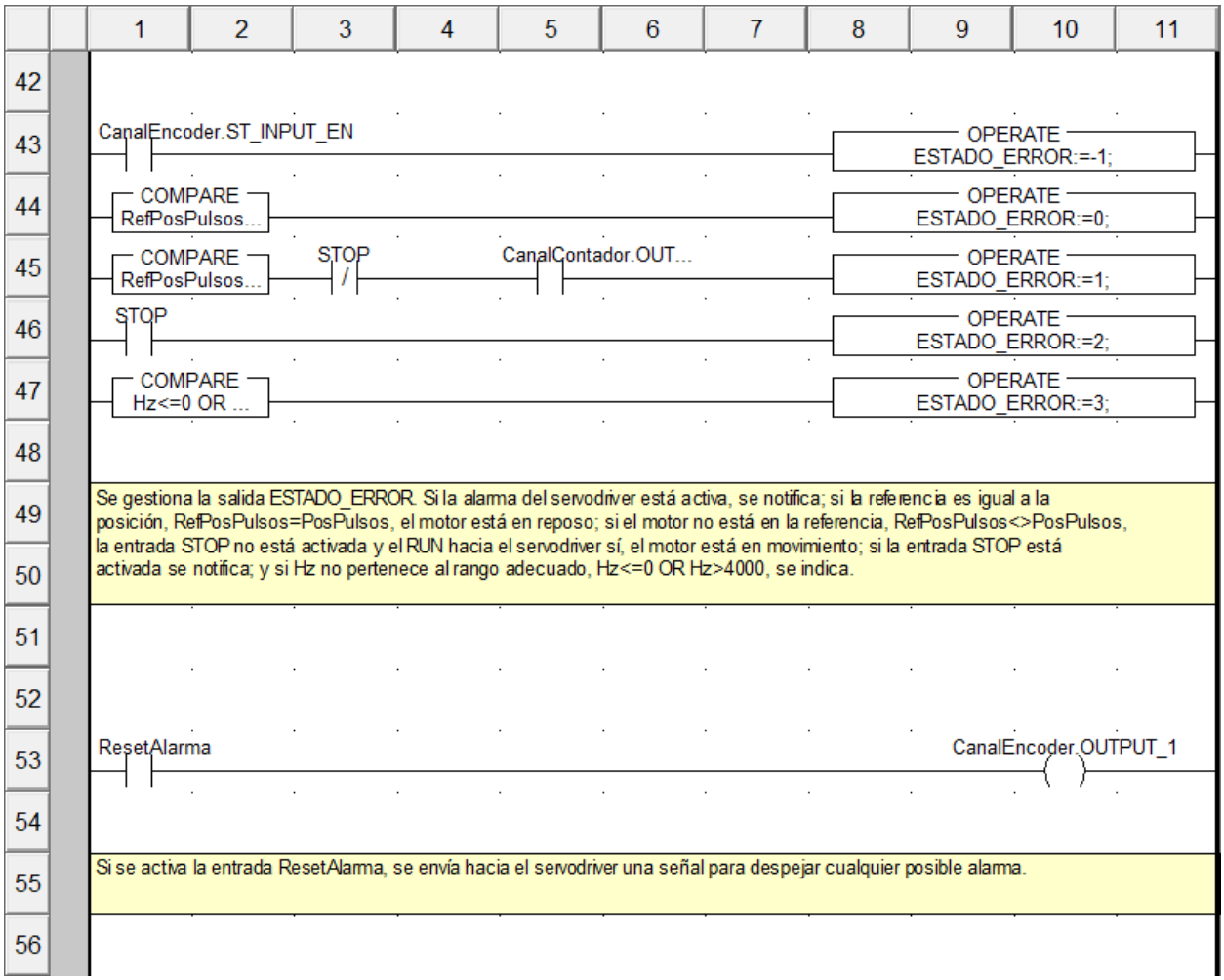
| Nombre | Tipo | Comentario |
|---|---|---|
| ContadorPulsos_0 | ContadorPulsos | |

Tabla I-6. Etiquetas truncadas de *MueveMotor*

| Etiqueta | Posición |
|---|---|
| ABS(RefPosPulsos-PosPulsos)<20 | (4, 10) |
| CanalContador.LOWER_TH_VALUE:=-RefPosPulsos; | (8, 23) |
| CanalContador.OUTPUT_1 | (5, 45) |
| CanalContador.UPPER_TH_VALUE:=-RefPosPulsos; | (8, 24) |
| CanalPWM.OUTPUT_BLOCK_0_ENABLE | (8, 7) |
| Hz<=0 OR Hz>4000 | (1, 47) |
| Hz>0 AND Hz<=4000 | (1, 9) |
| PosPulsos=RefPosPulsos | (2, 31) |
| RefPosPulsos<>PosPulsos | (2, 32) (1, 45) |
| RefPosPulsos=PosPulsos | (1, 44) |
| RefPosPulsos>PosPulsos | (1, 38) |




ContadorPulsos_0
ContadorPulsos
EN
ENO
CanalContador
Cana...
Cana...
Restaurar
Restaurar
PosP...
PosPulso
NuevaPosPulsos
NuevaPosPulsos
OPERATE
CanalContador.LOWER_TH_VALUE:=...
OPERATE
CanalContador.UPPER_TH_VALUE:=...
A la salida PosPulsos se le asigna el valor actual del contador de pulsos, y a los valores de los umbrales del comparador se les asignan el valor de la referencia (CanalContador.LOWER_TH_VALUE:=-RefPosPulsos y CanalContador.UPPER_TH_VALUE:=-RefPosPulsos). De esta forma, cuando la posición que contiene el contador alcance la referencia, se activará la salida refleja 0 y se desactivará el tren de pulsos hacia el motor. El signo negativo es debido a que internamente la tarjeta lleva la cuenta en el sentido contrario.
STOP
COMPARE
PosPulsos=R...
FrenoEnReposo
CanalContador.OUTPUT_1
COMPARE
RefPosPulsos...
STOP
Se gestiona la energización de las bobinas. Estas estarán alimentadas si se cumple alguna de las siguientes condiciones: la posición corresponde con la referencia y se ha solicitado el freno (PosPulsos=RefPosPulsos); se ha solicitado una parada y además el freno; o bien, el motor no se encuentra en la referencia de posición y no se ha solicitado una parada (RefPosPulsos<>PosPulsos).
COMPARE
RefPosPulsos...
CanalPWM.OUTPUT_1
Para determinar el sentido del movimiento, considerando la salida a nivel alto como el sentido de avance positivo, se comprueba si la referencia es mayor que la posición: RefPosPulsos>PosPulsos.


1 2 3 4 5 6 7 8 9 10 11
42
43
CanalEncoder.ST_INPUT_EN
OPERATE
ESTADO_ERROR:=-1;
44
COMPARE
RefPosPulsos...
OPERATE
ESTADO_ERROR:=0;
45
COMPARE
RefPosPulsos...
STOP
/
CanalContador.OUT...
OPERATE
ESTADO_ERROR:=1;
46
STOP
OPERATE
ESTADO_ERROR:=2;
47
COMPARE
Hz<=0 OR ...
OPERATE
ESTADO_ERROR:=3;
48
49
Se gestiona la salida ESTADO_ERROR. Si la alarma del servodriver está activa, se notifica; si la referencia es igual a la
posición, RefPosPulsos=PosPulsos, el motor está en reposo; si el motor no está en la referencia, RefPosPulsos<>PosPulsos,
50
la entrada STOP no está activada y el RUN hacia el servodriver sí, el motor está en movimiento; si la entrada STOP está
activada se notifica; y si Hz no pertenece al rango adecuado, Hz<=0 OR Hz>4000, se indica.
51
52
53
ResetAlarma
CanalEncoder.OUTPUT_1
54
55
Si se activa la entrada ResetAlarma, se envía hacia el servodriver una señal para despejar cualquier posible alarma.
56

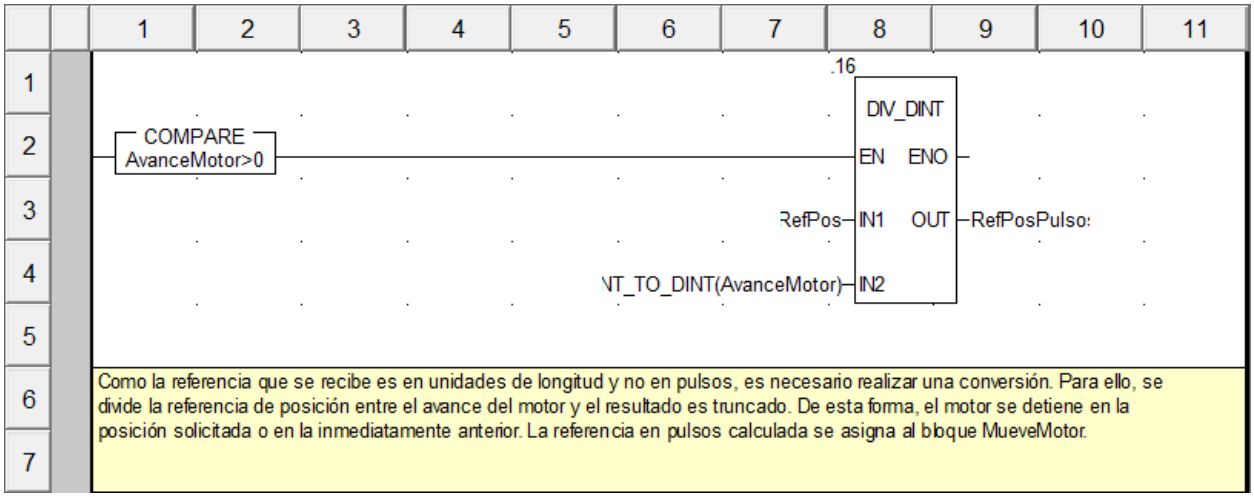
# 4. MueveAcoplamiento

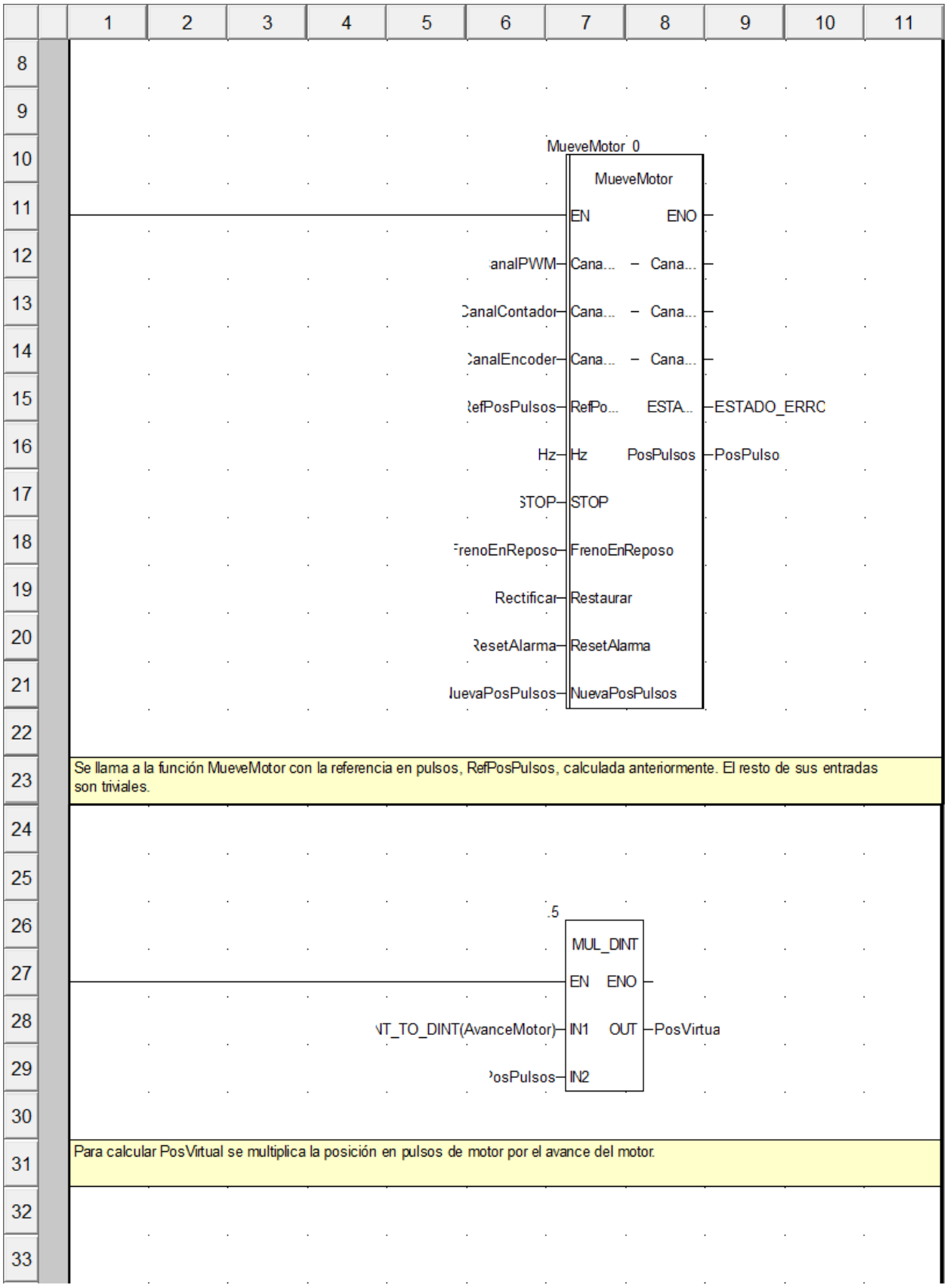
Tabla I-7. Variables privadas de *MueveAcoplamiento*

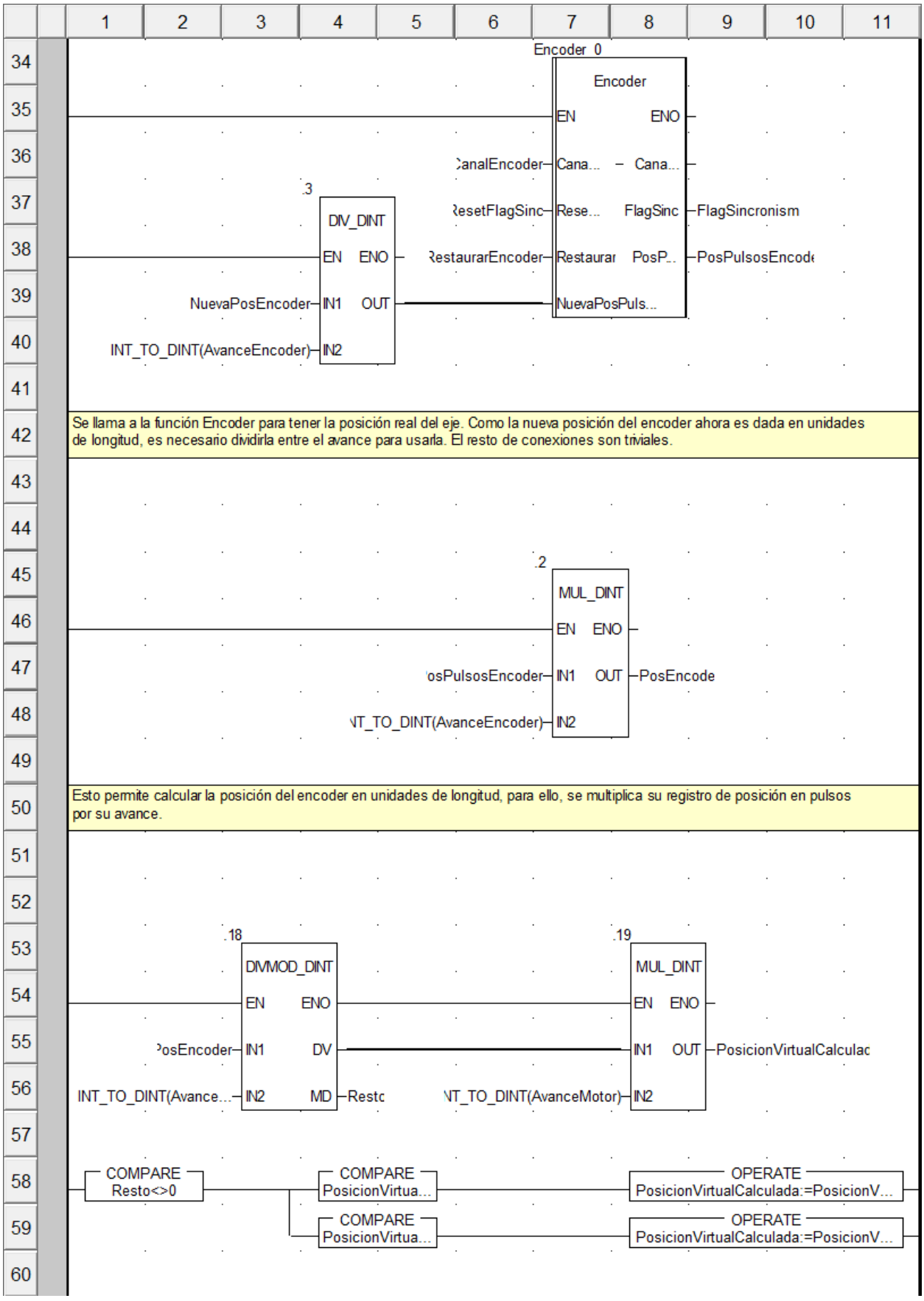
| Nombre | Tipo | Comentario |
|---|---|---|
| PosPulsos | DINT | Posición en pulsos de comando |
| PosVirtual | DINT | Posición en unidades de longitud a partir de *PosPulsos* |
| PosPulsosEncoder | DINT | Posición del *encoder* en pulsos |
| PosicionVirtualCalculada | DINT | Posición virtual calculada equivalente a PosEncoder |
| RefPosPulsos | DINT | Posición de referencia en pulsos de comando |
| Resto | DINT | Variable auxiliar para operaciones matemáticas |
| NuevaPosPulsos | DINT | Posición en pulsos de comando calculada para la rectificación |
| PosPulsosAnterior | DINT | Posición en pulsos de comando un segundo antes |
| PosPulsosEncoderAnterior | DINT | Posición en pulsos de *encoder* un segundo antes |
| EncoderMovimiento | BOOL | Variación en la posición del *encoder* |
| PulsosMovimiento | BOOL | Variación en la posición en pulsos de comando |
| MueveMotor_0 | MueveMotor | |
| Encoder_0 | Encoder | |
| SAMPLETM_0 | SAMPLETM | |

Tabla I-8. Etiquetas truncadas de *MueveAcoplamiento*

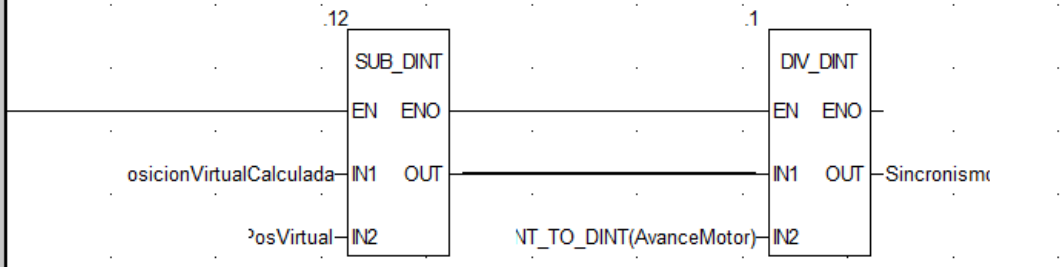
| Etiqueta | Posición |
|---|---|
| AvanceEncoder<=0 OR AvanceEncoder>32767 | (2, 84) |
| AvanceMotor<=0 OR AvanceMotor>32767 | (2, 83) |
| PosPulsosAnterior<>PosPulsos | (2, 94) |
| PosPulsosEncoderAnterior:=PosPulsosEncoder; | (8, 92) |
| PosPulsosEncoderAnterior<>PosPulsosEncoder | (2, 95) |
| PosicionVirtualCalculada:=PosicionVirtualCalculada+INT_TO_DINT(AvanceMotor); | (8, 59) |
| PosicionVirtualCalculada:=PosicionVirtualCalculada-INT_TO_DINT(AvanceMotor); | (8, 58) |
| PosicionVirtualCalculada<0 | (4, 58) |
| PosicionVirtualCalculada>0 | (4, 59) |



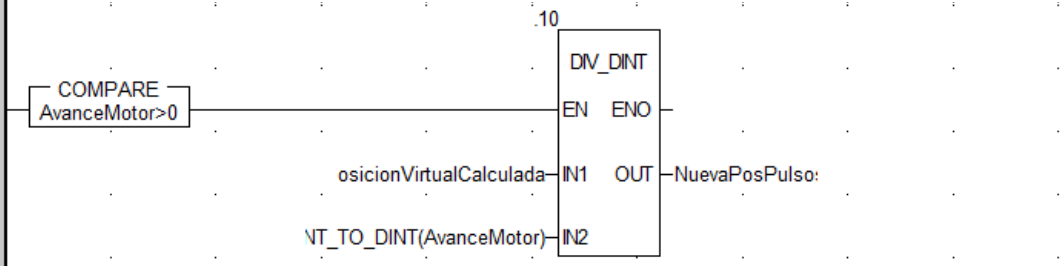

MueveMotor_0
MueveMotor
EN
ENO
analPWM
Cana...
Cana...
CanalContador
Cana...
Cana...
CanalEncoder
Cana...
Cana...
RefPosPulsos
RefPo...
ESTA...
ESTADO_ERRC
Hz
Hz
PosPulsos
PosPulso
STOP
STOP
FrenoEnReposo
FrenoEnReposo
Rectificar
Restaurar
ResetAlarma
ResetAlarma
NuevaPosPulsos
NuevaPosPulsos
Se llama a la función MueveMotor con la referencia en pulsos, RefPosPulsos, calculada anteriormente. El resto de sus entradas son triviales.
.5
MUL_DINT
EN
ENO
NT_TO_DINT(AvanceMotor)
IN1
OUT
PosVirtua
PosPulsos
IN2
Para calcular PosVirtual se multiplica la posición en pulsos de motor por el avance del motor.


Encoder_0
Encoder
EN
ENO
CanalEncoder
Cana...
Cana...
.3
DIV_DINT
ResetFlagSinc
Rese...
FlagSinc
FlagSincronism
EN
ENO
RestaurarEncoder
Restaurar
PosP...
PosPulsosEncode
NuevaPosEncoder
IN1
OUT
NuevaPosPuls...
INT_TO_DINT(AvanceEncoder)
IN2
Se llama a la función Encoder para tener la posición real del eje. Como la nueva posición del encoder ahora es dada en unidades de longitud, es necesario dividirla entre el avance para usarla. El resto de conexiones son triviales.
.2
MUL_DINT
EN
ENO
osPulsosEncoder
IN1
OUT
PosEncode
NT_TO_DINT(AvanceEncoder)
IN2
Esto permite calcular la posición del encoder en unidades de longitud, para ello, se multiplica su registro de posición en pulsos por su avance.
.18
DIVMOD_DINT
.19
MUL_DINT
EN
ENO
EN
ENO
osEncoder
IN1
DV
IN1
OUT
PosicionVirtualCalculac
INT_TO_DINT(Avance...
IN2
MD
Resto
NT_TO_DINT(AvanceMotor)
IN2
COMPARE
Resto<>0
COMPARE
PosicionVirtua...
COMPARE
PosicionVirtua...
OPERATE
PosicionVirtualCalculada:=PosicionV...
OPERATE
PosicionVirtualCalculada:=PosicionV...

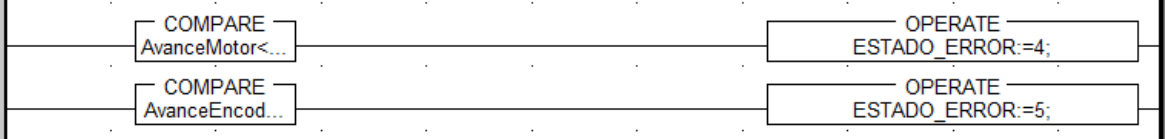
En estas líneas se convierte la posición real del encoder a su equivalente en la escala de posiciones virtuales del motor. Para ello, se divide la posición del encoder entre el avance del motor y nos quedamos con el cociente. El resultado lo volvemos a multiplicar por el avance del motor y obtenemos así la posición buscada. Además, hay que tener en cuenta el Resto, si este no es 0 quiere decir que a la posición obtenida hay que añadirle el desplazamiento correspondiente a un pulso, es decir, el avance del motor. En los comparadores se busca si la PosicionVirtualCalculada es positiva o negativa, ya que según eso hay que sumarle o restarle el avance del motor.



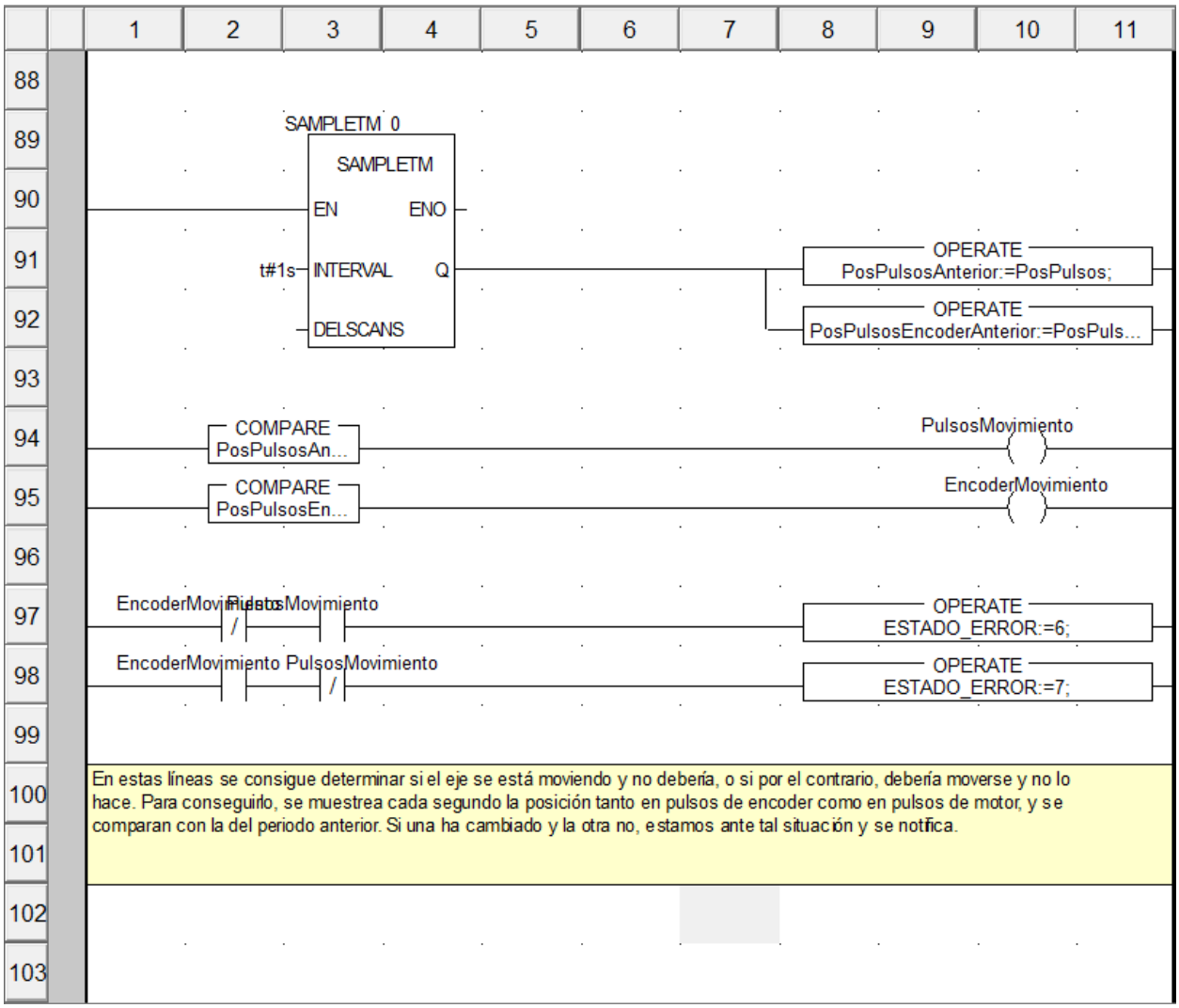

Teniendo la PosicionVirtualCalculada a partir del encoder, es fácil calcular el sincronismo. Tan solo se restan las posiciones medidas y calculadas, y se divide entre el avance del motor para obtener la salida en unidades de pulsos de motor.




Para llevar a cabo la operación de rectificación, es necesario recalcular la posición en pulsos de motor a partir de la posición del encoder. Se consigue dividiendo la posición virtual calculada por el avance del motor. Tan solo queda asignar este resultado a la entrada NuevaPosPulsos del bloque MueveMotor y activar Restaurar.




Si el avance del motor (AvanceMotor<=0 OR AvanceMotor>32767), o el avance del encoder (AvanceEncoder<=0 OR AvanceEncoder>32767), se encuentra fuera de rango es notificado al usuario.


SAMPLETM_0
SAMPLETM
EN
ENO
t#1s
INTERVAL
Q
DELSCANS
OPERATE
PosPulsosAnterior:=PosPulsos;
OPERATE
PosPulsosEncoderAnterior:=PosPuls...
COMPARE
PosPulsosAn...
PulsosMovimiento
COMPARE
PosPulsosEn...
EncoderMovimiento
EncoderMovimiento PulsosMovimiento
OPERATE
ESTADO_ERROR:=6;
EncoderMovimiento PulsosMovimiento
OPERATE
ESTADO_ERROR:=7;
En estas líneas se consigue determinar si el eje se está moviendo y no debería, o si por el contrario, debería moverse y no lo hace. Para conseguirlo, se muestrea cada segundo la posición tanto en pulsos de encoder como en pulsos de motor, y se comparan con la del periodo anterior. Si una ha cambiado y la otra no, estamos ante tal situación y se notifica.

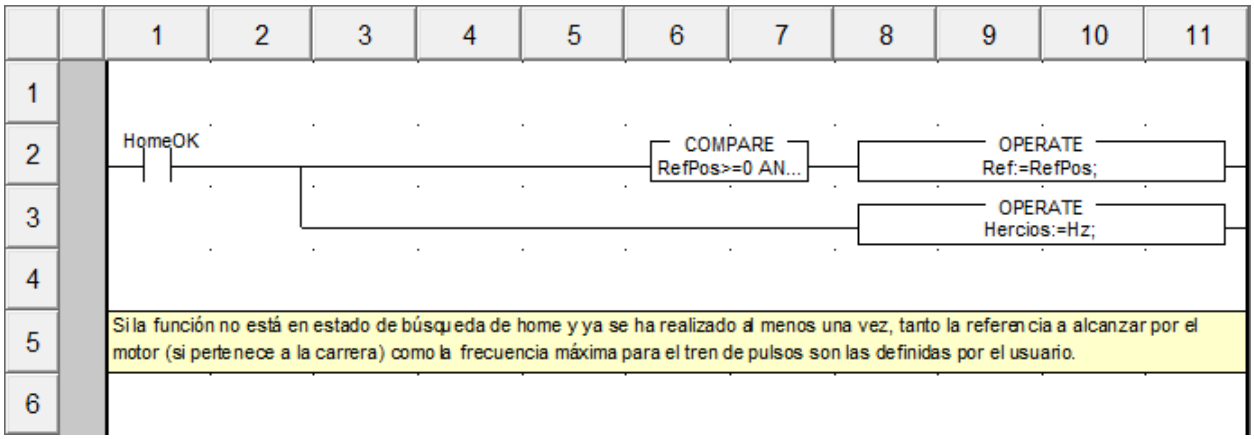
## 5. MueveDispositivo

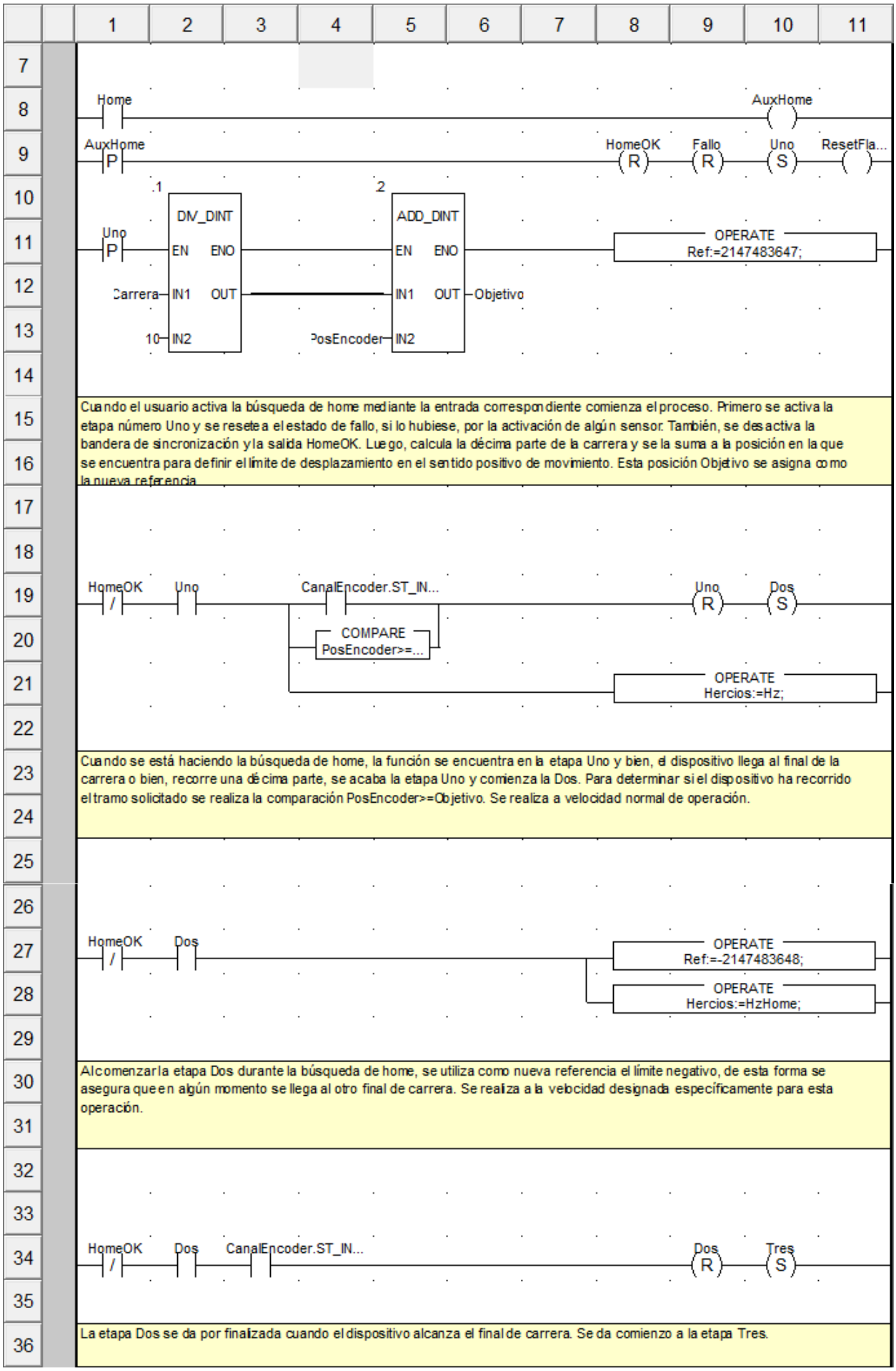
Tabla I-9. Variables privadas de *MueveDispositivo*

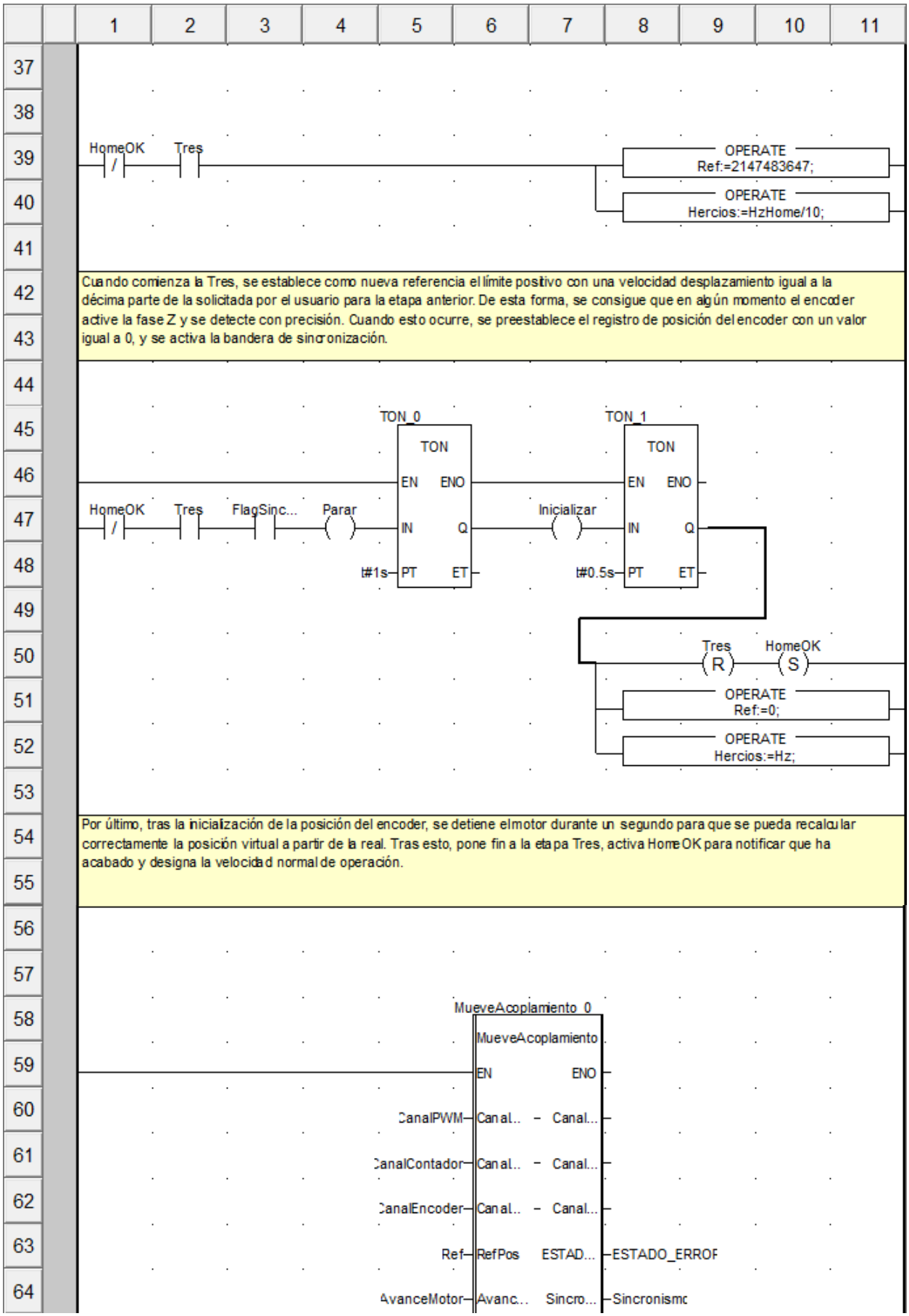
| Nombre | Tipo | Comentario |
|---|---|---|
| Ref | DINT | Define la posición a alcanzar en cada momento |
| Hercios | UDINT | Define la frecuencia de movimiento en cada momento |
| FlagSincronismo | BOOL | Bandera de sincronización del canal del *encoder* |
| AuxHome | EBOOL | Para la detección de flancos en la entrada *Home* |
| Fallo | BOOL | Variable de estado de fallo por activación de final de carrera |
| Uno | BOOL | Variable de etapa 1 |
| Dos | BOOL | Variable de etapa 2 |
| Tres | BOOL | Variable de etapa 3 |
| Objetivo | DINT | Almacena la décima parte de la carrera más la posición actual |
| ResetFlagSinc | BOOL | Resetea la bandera de sincronización del canal del *encoder* |
| Parar | BOOL | Variable auxiliar para la entrada *STOP* |
| Inicializar | BOOL | Variable auxiliar para reiniciar la posición |
| MueveAcoplamiento_0 | MueveAcoplamiento | |
| TON_0 | TON | |
| TON_1 | TON | |

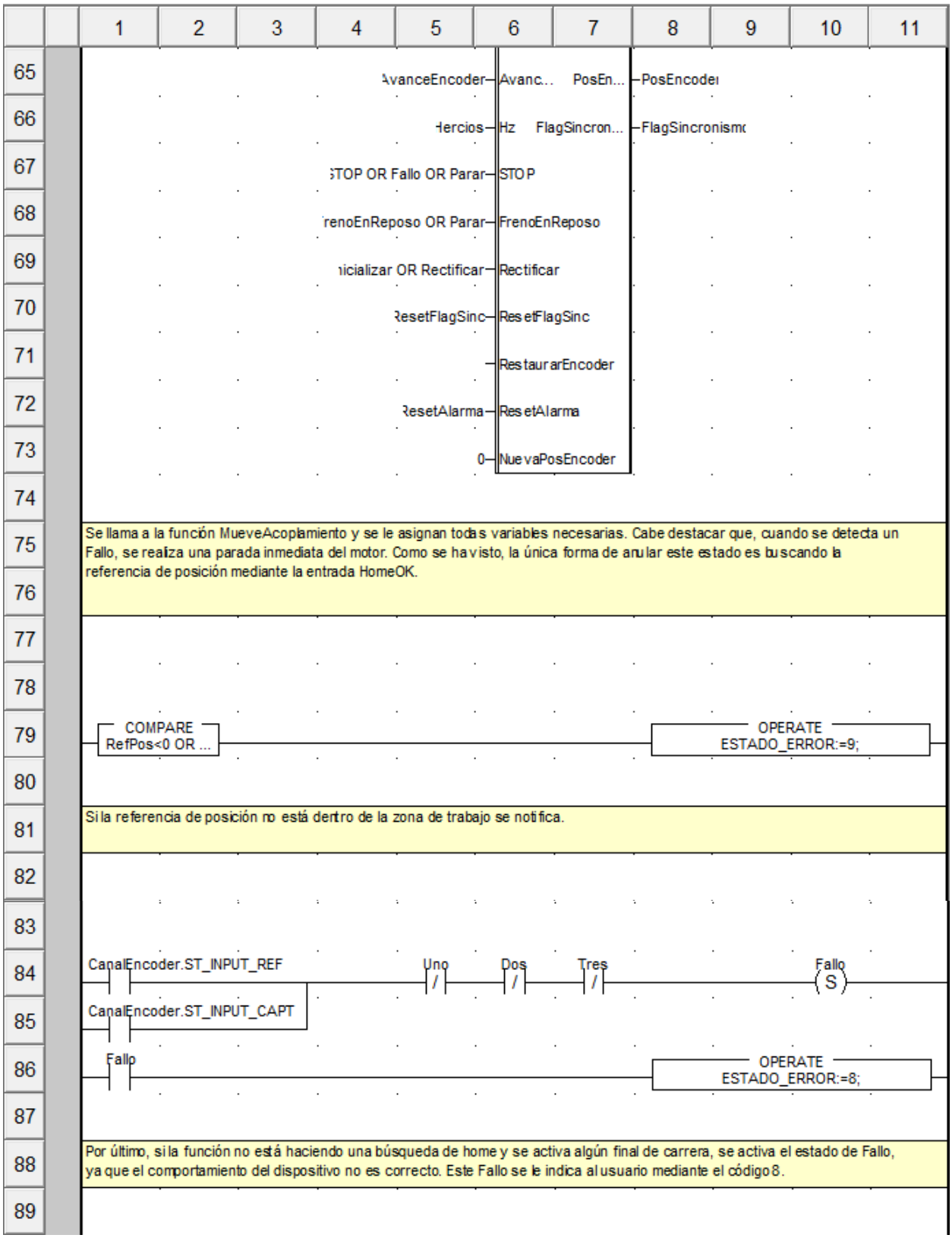
Tabla I-10. Etiquetas truncadas de *MueveDispositivo*

| Etiqueta | Posición |
|---|---|
| CanalEncoder.ST_INPUT_CAPT | (4, 19) |
| CanalEncoder.ST_INPUT_REF | (3, 34) |
| FlagSincronismo | (3, 47) |
| PosEncoder>=Objetivo | (4, 20) |
| RefPos<0 OR RefPos>Carrera | (1, 79) |
| RefPos>=0 AND RefPos<=Carrera | (6, 2) |
| ResetFlagSinc | (11, 9) |




Home
AuxHome
AuxHome
HomeOK
Fallo
Uno
ResetFla...
.1
DIV_DINT
EN
ENO
IN1
OUT
IN2
10
.2
ADD_DINT
EN
ENO
IN1
OUT
IN2
Objetivo
Uno
OPERATE
Ref:=2147483647;
Cuando el usuario activa la búsqueda de home mediante la entrada correspondiente comienza el proceso. Primero se activa la etapa número Uno y se resetea el estado de fallo, si lo hubiese, por la activación de algún sensor. También, se desactiva la bandera de sincronización y la salida HomeOK. Luego, calcula la décima parte de la carrera y se la suma a la posición en la que se encuentra para definir el límite de desplazamiento en el sentido positivo de movimiento. Esta posición Objetivo se asigna como la nueva referencia.
HomeOK
Uno
CanalEncoder.ST_IN...
Uno
Dos
COMPARE
PosEncoder>=...
OPERATE
Hercios:=Hz;
Cuando se está haciendo la búsqueda de home, la función se encuentra en la etapa Uno y bien, el dispositivo llega al final de la carrera o bien, recorre una décima parte, se acaba la etapa Uno y comienza la Dos. Para determinar si el dispositivo ha recorrido el tramo solicitado se realiza la comparación PosEncoder>=Objetivo. Se realiza a velocidad normal de operación.
HomeOK
Dos
OPERATE
Ref:=-2147483648;
OPERATE
Hercios:=HzHome;
Al comenzar la etapa Dos durante la búsqueda de home, se utiliza como nueva referencia el límite negativo, de esta forma se asegura que en algún momento se llega al otro final de carrera. Se realiza a la velocidad designada específicamente para esta operación.
HomeOK
Dos
CanalEncoder.ST_IN...
Dos
Tres
La etapa Dos se da por finalizada cuando el dispositivo alcanza el final de carrera. Se da comienzo a la etapa Tres.


HomeOK
Tres
OPERATE
Ref:=2147483647;
OPERATE
Hercios:=HzHome/10;
Cuando comienza la Tres, se establece como nueva referencia el límite positivo con una velocidad desplazamiento igual a la décima parte de la solicitada por el usuario para la etapa anterior. De esta forma, se consigue que en algún momento el encoder active la fase Z y se detecte con precisión. Cuando esto ocurre, se preestablece el registro de posición del encoder con un valor igual a 0, y se activa la bandera de sincronización.
TON_0
TON
EN
ENO
IN
Q
t#1s
PT
ET
TON_1
TON
EN
ENO
IN
Q
t#0.5s
PT
ET
HomeOK
Tres
FlagSinc...
Parar
Inicializar
Tres
R
HomeOK
S
OPERATE
Ref:=0;
OPERATE
Hercios:=Hz;
Por último, tras la inicialización de la posición del encoder, se detiene el motor durante un segundo para que se pueda recalcular correctamente la posición virtual a partir de la real. Tras esto, pone fin a la etapa Tres, activa HomeOK para notificar que ha acabado y designa la velocidad normal de operación.
MueveAcoplamiento_0
MueveAcoplamiento
EN
ENO
CanalPWM
Canal...
Canal...
CanalContador
Canal...
Canal...
CanalEncoder
Canal...
Canal...
Ref
RefPos
ESTAD...
ESTADO_ERROF
AvanceMotor
Avanc...
Sincro...
Sincronismc


AvanceEncoder
Avanc...
PosEn...
PosEncoder
Hercios
Hz
FlagSincron...
FlagSincronism
STOP OR Fallo OR Parar
STOP
FrenoEnReposo OR Parar
FrenoEnReposo
Inicializar OR Rectificar
Rectificar
ResetFlagSinc
ResetFlagSinc
RestaurarEncoder
ResetAlarma
ResetAlarma
0
NuevaPosEncoder
Se llama a la función MueveAcoplamiento y se le asignan todas variables necesarias. Cabe destacar que, cuando se detecta un Fallo, se realiza una parada inmediata del motor. Como se ha visto, la única forma de anular este estado es buscando la referencia de posición mediante la entrada HomeOK.
COMPARE
RefPos<0 OR ...
OPERATE
ESTADO_ERROR:=9;
Si la referencia de posición no está dentro de la zona de trabajo se notifica.
CanalEncoder.ST_INPUT_REF
Uno
Dos
Tres
Fallo
S
CanalEncoder.ST_INPUT_CAPT
Fallo
OPERATE
ESTADO_ERROR:=8;
Por último, si la función no está haciendo una búsqueda de home y se activa algún final de carrera, se activa el estado de Fallo, ya que el comportamiento del dispositivo no es correcto. Este Fallo se le indica al usuario mediante el código 8.

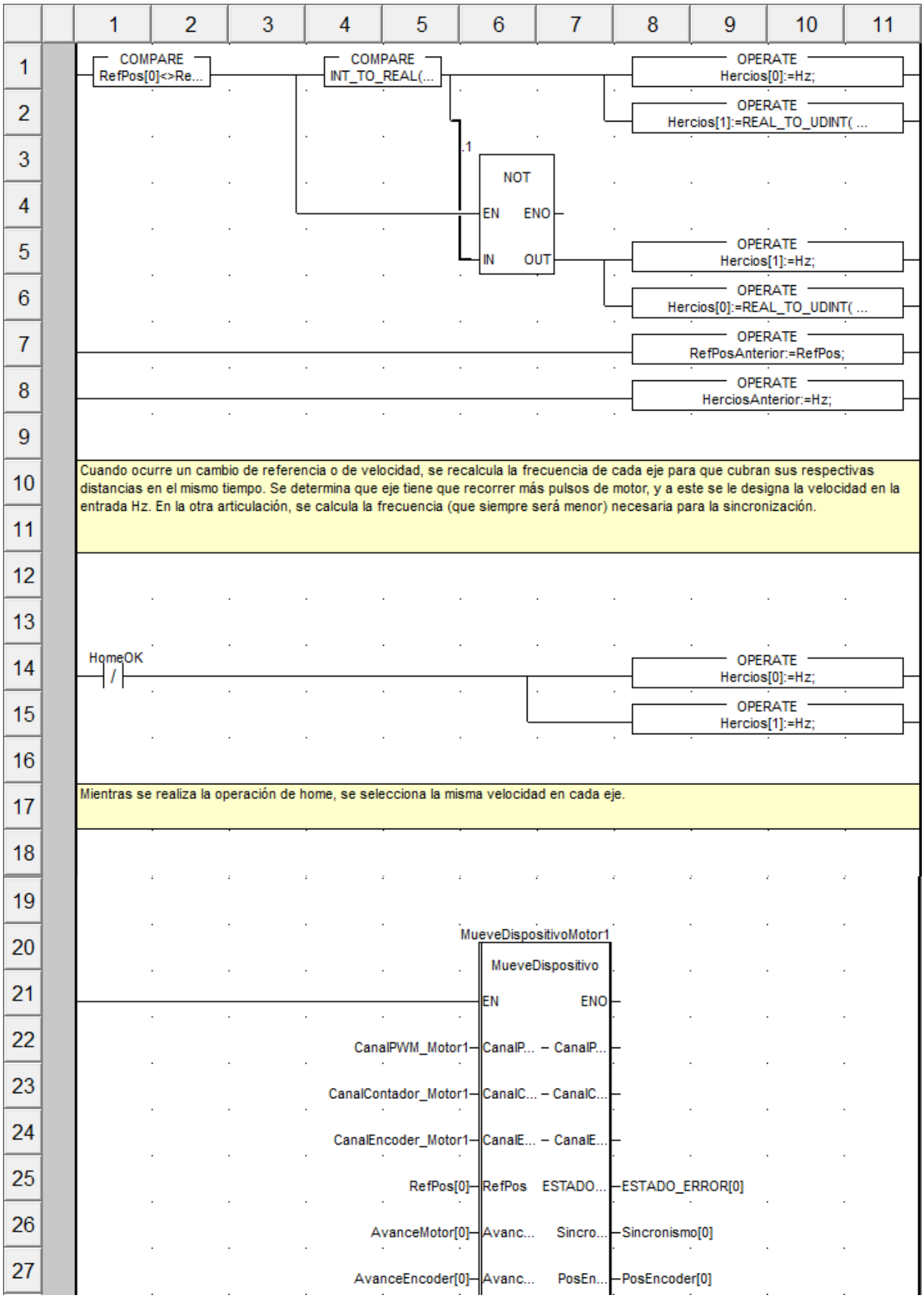
# 6. MueveMaquina

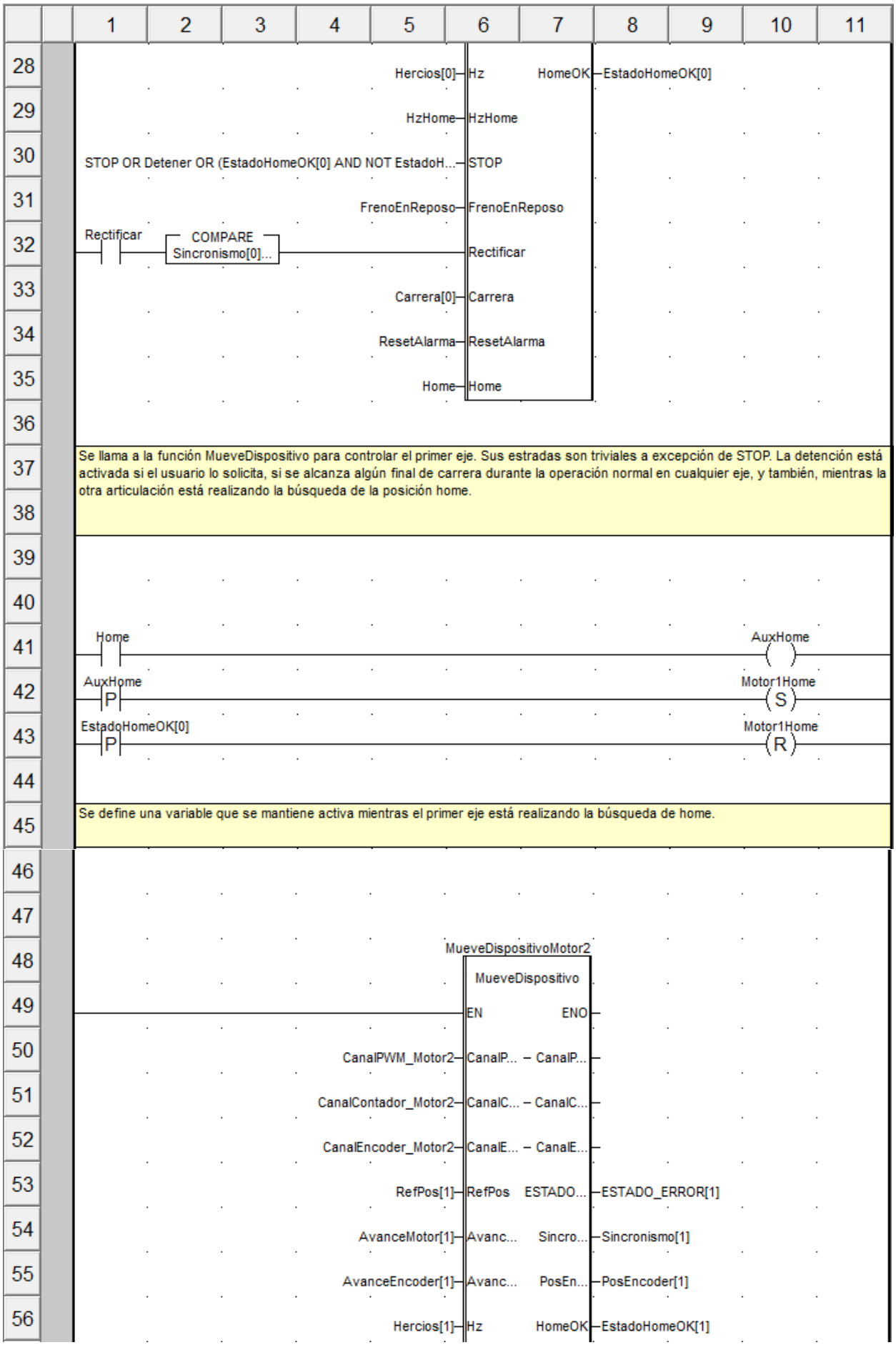
Tabla I-11. Variables privadas de *MueveMaquina*

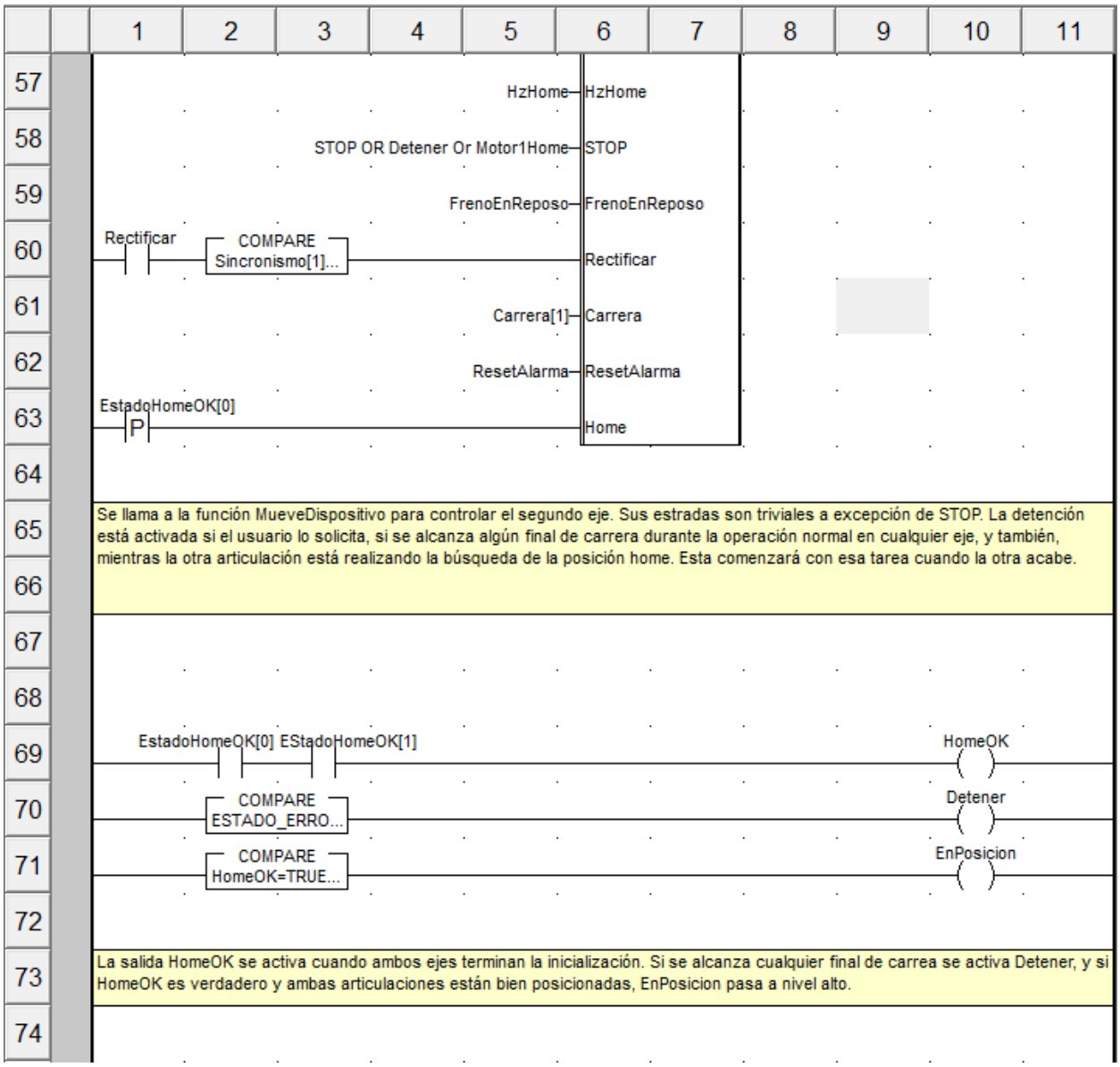
| Nombre | Tipo | Comentario |
|---|---|---|
| EstadoHomeOK | ARRAY[0..1] OF BOOL | Vector que almacena la salida *HomeOK* de cada eje |
| Hercios | ARRAY[0..1] OF UDINT | Vector que almacena la frecuencia a aplicar a cada eje |
| RefPosAnterior | ARRAY[0..1] OF DINT | Posición de referencia del ciclo anteior |
| HerciosAnterior | UDINT | Frecuencia designada en el ciclo anterior |
| Detener | BOOL | Detiene ambos ejes si se alcanza un final de carrera |
| Motor1Home | BOOL | Indica que el primer eje ya ha terminado de buscarlo |
| AuxHome | EBOOL | Para la detección de flancos en la entrada *Home* |
| MueveDispositivoMotor1 | MueveDispositivo | |
| MueveDispositivoMotor2 | MueveDispositivo | |

Tabla I-12. Etiquetas truncadas de *MueveMaquina*

| Etiqueta | Posición |
|---|---|
| ESTADO_ERROR[0]>=8 OR ESTADO_ERROR[1]>=8 | (2, 70) |
| Hercios[0]:=REAL_TO_UDINT( (UDINT_TO_REAL(Hz) * INT_TO_REAL(AvanceMotor[1]) * DINT_TO_REAL(ABS(RefPos[0]-PosEncoder[0])) ) / ( INT_TO_REAL(AvanceMotor[0]) * DINT_TO_REAL(ABS(RefPos[1]-PosEncoder[1])))); | (8, 6) |
| Hercios[1]:=REAL_TO_UDINT( (UDINT_TO_REAL(Hz) * INT_TO_REAL(AvanceMotor[0]) * DINT_TO_REAL(ABS(RefPos[1]-PosEncoder[1])) ) / ( INT_TO_REAL(AvanceMotor[1]) * DINT_TO_REAL(ABS(RefPos[0]-PosEncoder[0])))); | (8, 2) |
| HomeOK=TRUE AND ESTADO_ERROR[0]=0 AND ESTADO_ERROR[1]=0 | (2, 71) |
| INT_TO_REAL(AvanceMotor[1]) * DINT_TO_REAL(ABS(RefPos[0]-PosEncoder[0]))<=INT_TO_REAL(AvanceMotor[0]) * DINT_TO_REAL(ABS(RefPos[1]-PosEncoder[1])) | (4, 1) |
| RefPos[0]<>RefPosAnterior[0] OR RefPos[1]<>RefPosAnterior[1] OR HerciosAnterior<>Hz | (1, 1) |
| Sincronismo[0]<>0 | (2, 32) |
| Sincronismo[1]<>0 | (2, 60) |


COMPARE
RefPos[0]<>Re...
COMPARE
INT_TO_REAL(...
OPERATE
Hercios[0]:=Hz;
OPERATE
Hercios[1]:=REAL_TO_UDINT( ...
.1
NOT
EN
ENO
IN
OUT
OPERATE
Hercios[1]:=Hz;
OPERATE
Hercios[0]:=REAL_TO_UDINT( ...
OPERATE
RefPosAnterior:=RefPos;
OPERATE
HerciosAnterior:=Hz;
Cuando ocurre un cambio de referencia o de velocidad, se recalcula la frecuencia de cada eje para que cubran sus respectivas distancias en el mismo tiempo. Se determina que eje tiene que recorrer más pulsos de motor, y a este se le designa la velocidad en la entrada Hz. En la otra articulación, se calcula la frecuencia (que siempre será menor) necesaria para la sincronización.
HomeOK
OPERATE
Hercios[0]:=Hz;
OPERATE
Hercios[1]:=Hz;
Mientras se realiza la operación de home, se selecciona la misma velocidad en cada eje.
MueveDispositivoMotor1
MueveDispositivo
EN
ENO
CanalPWM_Motor1
CanalP...
CanalP...
CanalContador_Motor1
CanalC...
CanalC...
CanalEncoder_Motor1
CanalE...
CanalE...
RefPos[0]
RefPos
ESTADO...
ESTADO_ERROR[0]
AvanceMotor[0]
Avanc...
Sincro...
Sincronismo[0]
AvanceEncoder[0]
Avanc...
PosEn...
PosEncoder[0]


Hercios[0]
Hz
HomeOK
EstadoHomeOK[0]
HzHome
HzHome
STOP OR Detener OR (EstadoHomeOK[0] AND NOT EstadoH...
STOP
FrenoEnReposo
FrenoEnReposo
Rectificar
COMPARE
Sincronismo[0]...
Rectificar
Carrera[0]
Carrera
ResetAlarma
ResetAlarma
Home
Home
Se llama a la función MueveDispositivo para controlar el primer eje. Sus estradas son triviales a excepción de STOP. La detención está activada si el usuario lo solicita, si se alcanza algún final de carrera durante la operación normal en cualquier eje, y también, mientras la otra articulación está realizando la búsqueda de la posición home.
Home
AuxHome
AuxHome
P
Motor1Home
S
EstadoHomeOK[0]
P
Motor1Home
R
Se define una variable que se mantiene activa mientras el primer eje está realizando la búsqueda de home.
MueveDispositivoMotor2
MueveDispositivo
EN
ENO
CanalPWM_Motor2
CanalP...
CanalP...
CanalContador_Motor2
CanalC...
CanalC...
CanalEncoder_Motor2
CanalE...
CanalE...
RefPos[1]
RefPos
ESTADO...
ESTADO_ERROR[1]
AvanceMotor[1]
Avanc...
Sincro...
Sincronismo[1]
AvanceEncoder[1]
Avanc...
PosEn...
PosEncoder[1]
Hercios[1]
Hz
HomeOK
EstadoHomeOK[1]


HzHome
HzHome
STOP OR Detener Or Motor1Home
STOP
FrenoEnReposo
FrenoEnReposo
Rectificar
COMPARE
Sincronismo[1]...
Rectificar
Carrera[1]
Carrera
ResetAlarma
ResetAlarma
EstadoHomeOK[0]
P
Home
Se llama a la función MueveDispositivo para controlar el segundo eje. Sus estradas son triviales a excepción de STOP. La detención está activada si el usuario lo solicita, si se alcanza algún final de carrera durante la operación normal en cualquier eje, y también, mientras la otra articulación está realizando la búsqueda de la posición home. Esta comenzará con esa tarea cuando la otra acabe.
EstadoHomeOK[0] EStadoHomeOK[1]
HomeOK
COMPARE
ESTADO_ERRO...
Detener
COMPARE
HomeOK=TRUE...
EnPosicion
La salida HomeOK se activa cuando ambos ejes terminan la inicialización. Si se alcanza cualquier final de carrea se activa Detener, y si HomeOK es verdadero y ambas articulaciones están bien posicionadas, EnPosicion pasa a nivel alto.

# REFERENCIAS


[1] Advance Manufacturing Technologies, «Productos / Máquinas-herramientas / Zayer_neos,» [En línea]. Available: https://www.afm.es/es/productos/maquinas-herramienta/centros-mecanizado/centros-mecanizado-horizontales-alta-velocidad/fresadoras-gantry-zayer-neos. [Último acceso: Mayo 2018].

[2] FANUC, «Robots / Robots colaborativos,» [En línea]. Available: https://www.fanuc.eu/es/es/robots/p%C3%A1gina-filtro-robots/%D1%80obots-colaborativos/collaborative-cr7ial. [Último acceso: 1 Mayo 2018].

[3] Prusa 3D, «Impresoras / Prusa I3 MK3,» [En línea]. Available: https://www.prusa3d.es/original-prusa-i3-mk3-spa/. [Último acceso: 1 Mayo 2018].

[4] F. J. C. Solís, Control de Ejes mediante PLC. Aplicación en Célula de Fabricación Flexible., Sevilla, 2016.

[5] «Artisan Technology Group,» [En línea]. Available: https://www.artisantg.com/ViewImage.aspx?Image=Omron_R88D_UP04V_View1.JPG&Item=63320-11. [Último acceso: 12 09 2018].

[6] «Artisan Technology Group,» [En línea]. Available: https://www.artisantg.com/ViewImage.aspx?Image=OMRON_R88M-U10030VA_BS1_View1.jpg&Item=75113-26. [Último acceso: 12 09 2018].

[7] OMRON, "Omron eData," [Online]. Available: http://www.edata.omron.com.au/eData/Servos/I514-E1-2.pdf. [Accessed 12 09 2018].

[8] OMRON, «Omron eData,» [En línea]. Available: http://www.edata.omron.com.au/eData/Servos/I800-E1-05.pdf. [Último acceso: 12 09 2018].

[9] S. Electric, «Descarca de software y documentación de productos | Schneider Electric,» [En línea]. Available: https://www.schneider-electric.es/es/download/. [Último acceso: 09 12 2018].

[10] Yaskawa, «SigmaWin+ Ver.5 - Yaskawa,» [En línea]. Available: https://www.yaskawa.com/products/motion/sigma-5-servo-products/software-tools/sigmawinplus. [Último acceso: 12 09 2018].

# GLOSARIO