Trabajo Fin de Máster
Máster en Ingeniería Industrial

Introducción al control remoto de servomotores industriales

Autor: Juan Carlos Fraile García
Tutor: Luis Fernando Castaño Castaño

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Máster
Máster en Ingeniería Industrial

Introducción al control remoto de servomotores industriales

Autor:
Juan Carlos Fraile García

Tutor:
Dr. Luis Fernando Castaño Castaño

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Máster: Introducción al control remoto de servomotores industriales

Autor: Juan Carlos Fraile García
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, 2021

El Secretario del Tribunal

Agradecimientos

A mis padres, por haber hecho lo imposible para que pudiera llegar hasta aquí y apoyarme en cada decisión. A mi hermano, por romperme mil barreras y enseñarme a ser quien soy. A Enrique y Carmen Lucía, por simplificar lo complicado.
A mi segunda Familia, por detener el tiempo para equilibrar la balanza y compartir vuestra felicidad y tristeza.
A Rafael, Sebastián, Miguel y Mario por hacer infinitamente más ameno estos dos años tan atípicos de Máster.
A Fernando por todo su trato y apoyo, por abrirme una nueva puerta en mi futuro profesional.

Juan Carlos Fraile García
Sevilla, 2021

Resumen

Los sistemas de control de movimiento son parte indispensable de una inmensa cantidad de aplicaciones industriales: sistemas de transporte, sistemas robóticos, sistemas de mecanizado (CNC), sistemas de alimentación de materia prima, sistemas de almacenamiento, etc.

Atendiendo a como se consigue controlar el motor que acciona el sistema se pueden distinguir tres grandes alternativas: Variadores de Velocidad o Frecuencia, Motores paso a paso y Servomotores. Desde el punto de vista del control, el más flexible, preciso y completo es el Servomotor por lo que, el desarrollo de la Industria, precisa cada vez más de su utilización.

Este proyecto se enfoca en realizar una Introducción completa al control de servomotores industriales desde una exposición teórica hasta un ejemplo de implementación real. Sin perdida de generalidad, todo el documento se desarrolla en el marco de equipos comerciales reales del fabricante *Schneider Electric*. Sin embargo, como se verá en el último capítulo, todo lo aquí aprendido podrá extrapolarse a los sistemas de control de movimiento propuestos por otros fabricantes (Siemens, ABB, etc).

Todo sistema de control de movimiento que cuente con ejes accionados por servomotores cuenta con unos equipos electrónicos denominados servo drives que se encargan del control en posición, velocidad y/o par. En el Capítulo 2 se aglutinan los conceptos teóricos asociados a estos dispositivos (conexionado, modos de funcionamiento, diccionario de parámetros, estructura del regulador, etc.)

Un servo accionamiento que se encarga de controlar un eje puede tener que integrarse con otros sistemas/dispositivos que también se dediquen o no al control de movimiento. Para ello, es necesario que exista un controlador de nivel superior (o remoto). En el Capítulo 1 se mencionan distintas alternativas. Todas ellas tienen en común una librería de funciones (*Motion Control*) que facilitan el manejo y monitorización de los servos.

El primer paso para la implantación de estos equipos en cualquier tipo de aplicación será realizar su configuración/parametrización inicial. Por regla general los fabricantes suelen poner a disposición del usuario un software para dicha puesta en marcha. En el Capítulo 1 se explora el ofrecido por *Schneider*.

En numerosas aplicaciones, se precisa que los servomotores sean capaces de trabajar de forma sincronizada entre ellos (o con otro tipo de motores) para realizar tareas como: Alimentar piezas, realizar cortes al vuelo, embotellar líquidos, etc. Por ello, se presenta en el Capítulo 1 una aplicación práctica de sincronización entre dos servomotores utilizando un PLC como controlador remoto. La implementación se describe desde el montaje físico de una maqueta de pruebas hasta la depuración de la aplicación.

En el Capítulo 5 se expone, de manera general, como se realiza el control remoto con otra familia de PLCs distinta (M238) a la utilizada en el Capítulo 1 (M340). El objetivo principal es salvar las diferencias entre el software de programación empleado en el M238 (SoMachine) y el utilizado en el M340 (Unity Pro)

Por último, en el Capítulo 1 se realizan diferentes propuestas para ampliar o profundizar lo visto en esta Introducción.

Abstract

Motion control systems are crucial for a wide range of industrial applications such as transport systems, robotic systems, machining systems (CNC), raw material feeding systems, storage systems...
Depending on how the motor that drives the system is controlled, three main alternatives can be distinguished: Variable Speed or Frequency Drives, Stepper Motors and Servomotors. From the control point of view, the one which provides more flexibility and accuracy is the servomotor. That is why the development of industry increasingly requires them.
This project focuses on a complete introduction to the control philosophy of industrial servomotors from the theory to a real implementation. Without loss of generality, the whole document is developed within the framework of real equipment from the manufacturer *Schneider Electric*. However, as it is stated in the last chapter, everything within this document can be extrapolated to motion control systems made by other manufacturers like *Siemens* or *ABB*.
Every servomotor has to be connected to an electronic equipment called servo drive which is responsible for its position, speed and torque control. Chapter 2 gathers essential basic theory associated with these devices (connections, operating modes, configurable parameters, controller structure and so on)
A servo drive that is in charge of control an axis may need to be integrated with other systems or devices which may or may not also be dedicated to motion control. This requires the existence of a higher-level (or remote) controller. Some commercial alternatives are mentioned in Chapter 3. All of them have in common a function library (*Motion Control*) that makes easier the handling and monitoring of the servos.
The first step for using these devices in any real application consists on carrying out an initial configuration of their parameters. As a rule, manufacturers usually provide their clients with a suitable software for this purpose. Chapter 4 explores the software provided by *Schneider* that is called *SOMove*.
In many applications, servomotors need to be able to work synchronously with other motors to perform some complex tasks such as feeding parts, flying saw or bottling liquids. For this reason, a practical application of synchronisation between two servomotors using a PLC as a remote controller, can be found on chapter 6. The full implementation is described. From the wiring to the application programming. An HMI has also been developed in order to make several tests.
Chapter 5 briefly describes how the remote control is performed with a different PLC family (M238) than the one used in Chapter 6 (M340). One of the main objectives is showing the differences between the programming software used in the M238 (*SoMachine*) and the one used in the M340 (*Unity Pro*). Finally, some ideas to keep going into details about servodrives and their applications can be found in Chapter 7.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Los motores son parte indispensable de una inmensa cantidad de aplicaciones industriales: sistemas de transporte, sistemas robóticos, sistemas de mecanizado (CNC), sistemas de alimentación de materia prima , sistemas de almacenamiento, etc.

Todos los motores cuentan con un accionamiento que será utilizado, generalmente, para llevar a cabo un control del eje en posición, velocidad y/o par que les permita cumplir adecuadamente con su función. Un accionamiento puede ser:

- **Neumático**: para aplicaciones de bajo costo y precisión.
- **Hidráulico**: para aplicaciones donde sea necesario ejercer un par elevado manteniendo una precisa regulación de la velocidad.
- **Eléctrico**: el más empleado debido a su facilidad para ser controlado y su mayor precisión respecto a los tipos anteriores.

En el contexto de este proyecto, nos centraremos en los eléctricos. Existen una gran variedad de motores eléctricos que podríamos agrupar básicamente en:

- **Motores de corriente continua (DC):** Serie, compound, shunt, sin escobillas…
- **Motores de corriente alterna monofásica o trifásica (AC):**
  - **Motores síncronos**: Brushless, Lineales, de accionamiento directo...
  - **Motores asíncronos** o de Inducción.

Sin embargo, si se atiende solamente a como consiguen el control de par, velocidad y/o posición existen 3 grupos principalmente:

Motores controlados mediante variadores de velocidad: los variadores de velocidad o VSD son unos

dispositivos que permiten, como su propio nombre indica, controlar la velocidad de giro del motor. En general, estos dispositivos trabajan en bucle abierto, aunque pueden aplicarse lazos cerrados de control si se requiere mayor precisión.

Son ampliamente utilizados en aplicaciones que hacen uso de un motor AC asíncrono. En dicho caso se conocen como variadores de frecuencia (VFD).

Los variadores de frecuencia con motores AC de inducción consiguen altas potencias a bajos costos y son especialmente útiles en aquellas aplicaciones de movimiento continuo donde no es necesario realizar un control del posicionamiento del motor. Por lo tanto, se suelen utilizar variadores de frecuencia en el control de sistemas donde las cargas son elevadas y variables como en cintas transportadoras.

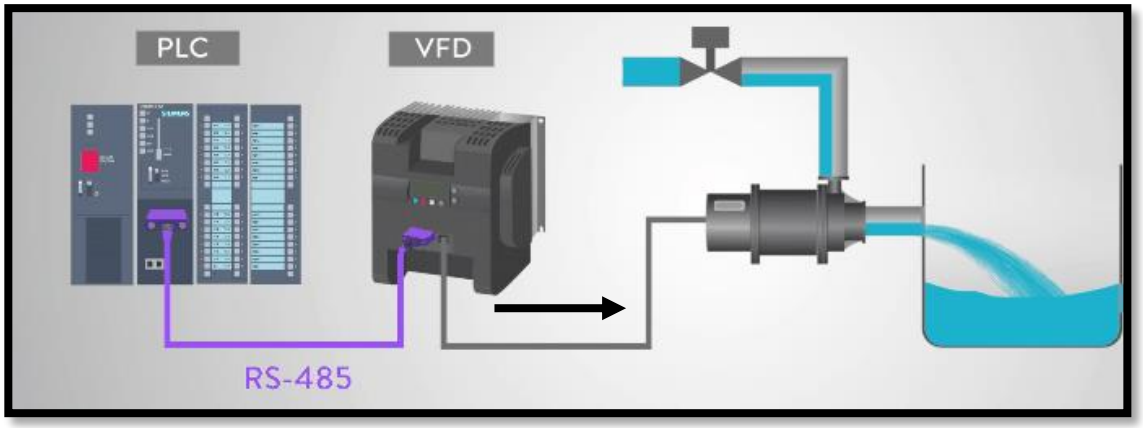
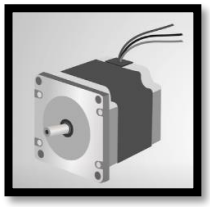


Figura 1-1: Lazo de control (abierto) de motor AC controlado con un VFD [1].

Motores paso a paso: Los motores paso a paso trabajan convirtiendo una serie de impulsos eléctricos (tren de pulsos) en desplazamientos angulares discretos. A diferencia de los motores controlados mediante variadores de velocidad, los motores paso a paso si se orientan a obtener un control en posición. También es posible conseguir un control en velocidad ya que es proporcional a la frecuencia del tren de pulsos con el que se alimente el motor.



Por otro lado, en general, el control de los motores paso a paso se realiza en lazo abierto por lo que no existe ningún tipo de realimentación que nos permita determinar si se alcanza la referencia antes de lo previsto o ha ocurrido algún error en el proceso. El controlador alterna la energización de unas bobinas u otras para hacer rotar el eje (Figura 1-2).

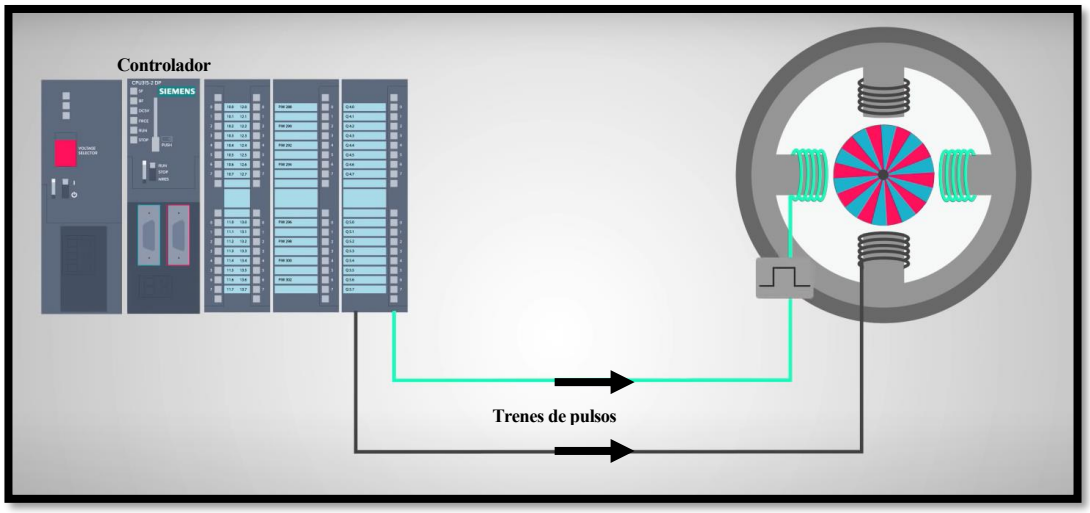
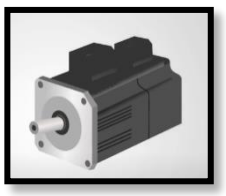


Figura 1-2: Lazo de control (abierto) de un motor paso a paso [2].

Servomotores: Los servomotores son motores síncronos que trabajan en bucle cerrado dentro de un sistema que estará conformado principalmente por: un dispositivo de control externo (opcional, por ejemplo, un PLC), el controlador electrónico (servo drive), un encoder y el servomotor. El servo drive es el dispositivo que se encargará del control del motor en posición, velocidad y/o par atendiendo a la señal del encoder y a lo comandado por el dispositivo de control externo o, en su defecto, a lo programado internamente.



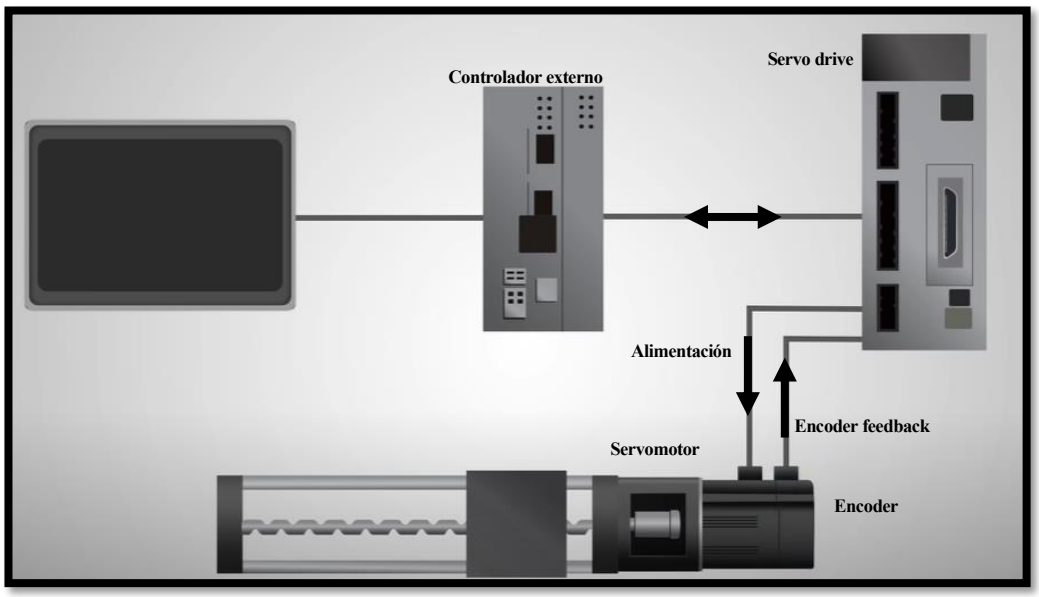


Figura 1-3: Lazo de control (cerrado) de un servomotor [3].

Comparando los motores paso a paso con los servomotores se pueden extraer las siguientes conclusiones [4]:

- A igual tamaño, el par otorgado por un motor paso a paso es mayor que el del servomotor a velocidades bajas. Sin embargo, el par entregado por el motor paso a paso decrece rápidamente conforme aumenta la velocidad. Por lo tanto, atendiendo a lo anterior, **a velocidades bajas será más adecuado un motor paso a paso y a velocidades altas un servomotor**.

- **El movimiento de los servomotores es, en general, más suave** que el de los motores PaP (excepto cuando estos cuentan con la opción de usar micro pasos). Los motores PaP también generan un mayor ruido durante su operación.

- **Los motores PaP** tienen menor relación par/inercia con lo que **no pueden acelerarse igual de rápidos que los servomotores**. Por lo tanto, se utilizarán preferentemente estos últimos en aplicaciones que necesiten de movimientos muy rápidos con grandes aceleraciones y deceleraciones.

- **Los motores PaP no incluyen un lazo de realimentación con lo que la fiabilidad y precisión de su control es menor al ser vulnerables a errores**.

- El lazo de realimentación permite que los servomotores dispongan de algoritmos de control más sofisticados que mejoren su respuesta. Sin embargo, esto supone al mismo tiempo un **incremento en la complejidad y tiempo de puesta en marcha de los servomotores frente a la facilidad de instalación y uso de los motores PaP**.

- Dado que son sistemas electromecánicos más simples, **los motores PaP son más baratos**.

La elección de un tipo de motor u otro dependerá fundamentalmente de las características de la aplicación en la que vayan a implementarse. Los servomotores son ampliamente utilizados en diferentes tipos de industrias (automotriz, textil, alimenticia, logística, etc.) con diferentes propósitos:

- Máquinas CNC
- Líneas de ensamblaje, embalaje y empaquetado
- Robótica (como los SCARA o cualquier otro brazo manipulador)
- Movimientos de prensado, sujeción o empuje que requieren de un control preciso del par aplicado

En definitiva, estos motores se hacen indispensables en todas aquellas aplicaciones que requieren de rapidez y alta precisión en posición, velocidad y/o par aplicado.
El crecimiento de ese tipo de aplicaciones ha generado la necesidad de encontrar profesionales capacitados en el montaje, puesta en marcha y manejo de estos dispositivos electromecánicos. Existe una gran cantidad de información disponible en libros y manuales de fabricantes. Sin embargo, esta información se encuentra en la mayoría de las ocasiones, dispersa, resultando en una extensa curva de aprendizaje.

1.1 Objetivos del Proyecto

El objetivo principal de este proyecto es la realización de un estudio general de los servomotores industriales. Se pretende realizar un recorrido por distintos conocimientos generales de forma que facilite la formación desde cero desde el punto de vista de control. Para ello, se desarrollan principalmente los siguientes aspectos:

- Conexionado y puesta en marcha
- Manejo (estados, modos de funcionamiento, canales de acceso…)
- Integración en sistemas más completos con un controlador remoto.

El proyecto se desarrolla particularizando los aspectos anteriores para un servomotor comercial concreto del proveedor *Schneider Electric (*LXM32M) y para controladores lógicos (PLC) de la misma firma. Con esto se pretende dar al contenido del documento un enfoque más práctico que puramente teórico.
La utilización de cualquier otro servomotor industrial y/o PLC solo precisará la extrapolación de los conceptos asimilados para los ejemplos comerciales aquí utilizados.

En línea con el enfoque buscado, tras un primer bloque de introducción teórica, se exponen algunos ejemplos de implementación práctica de servomotores controlados de forma remota. De esta forma, el lector conocerá los diferentes pasos esenciales a seguir para la implantación desde cero de este tipo de sistemas de control de movimiento. A partir de ahí, podrá profundizar y extrapolar lo aprendido a otras aplicaciones y/o equipos comerciales.

En particular, se hará un mayor énfasis sobre una aplicación orientada al control y manejo de dos servomotores que cuenten con la funcionalidad de sincronizarse entre sí. Para dicho caso, la implementación práctica se ha llevado a cabo fuera del papel sobre unos equipos disponibles en el laboratorio de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Como resultado se ha obtenido una maqueta de pruebas que puede ser utilizada con fines académicos, para exponer y ayudar al alumnado a familiarizarse con el funcionamiento y control de los servomotores así como a aplicar lo aprendido.

2 SERVO ACCIONAMIENTO

En este capítulo se realiza, en primer lugar, una descripción del servo accionamiento sobre el que se particularizan los distintos aspectos del presente proyecto. Se puede encontrar información más detallada y conceptos más avanzados en el manual de este [5]. Todo lo expuesto en el capítulo podrá ser asemejado posteriormente de forma sencilla a cualquier otro modelo comercial distinto.

2.1 Descripción física

Un servo accionamiento es un sistema conformado básicamente por un servomotor, un encoder en su interior y un servo drive. Estos componentes materializan un circuito realimentado para el control de movimiento del eje acoplado en posición, velocidad y/o par.

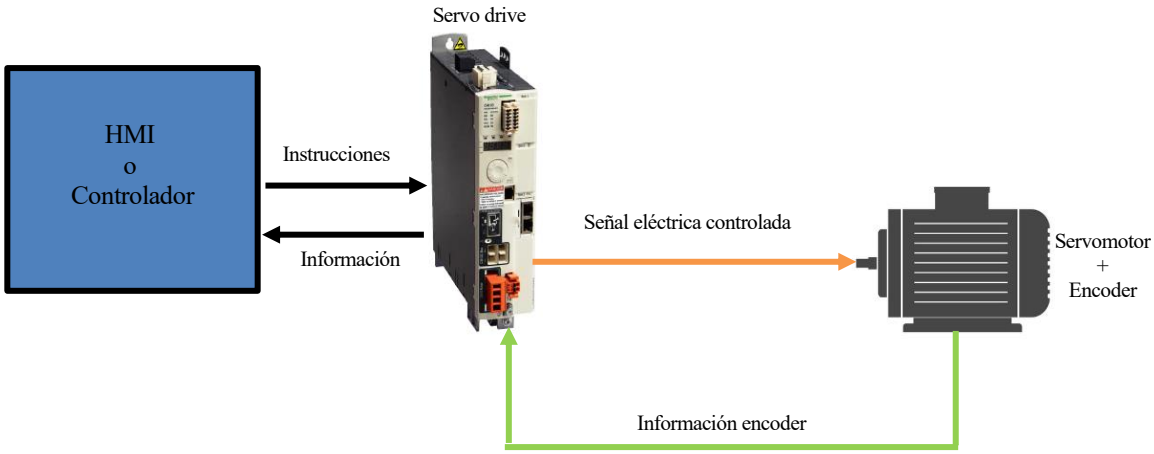


Figura 2-1: Esquema de servo accionamiento

Para que dicho control se realice de forma adecuada será necesario, además de una correcta configuración del servo drive, recibir las instrucciones o consignas de control adecuadas. Estas pueden proceder de distintas fuentes.

2.1.1 Servomotor

Existen servomotores de corriente continua y de corriente alterna. En el caso de los de corriente continua, cuanto mayor sea su tensión de alimentación, más velocidad a par constante podrá alcanzar. En corriente alterna no se presenta dicho problema. Entre los de corriente alterna monofásicos y los trifásicos son los segundos los que mayores potencias alcanzan. Para el presente proyecto se hace uso del modelo BSH0551T01A2A de *Schneider Electric*.

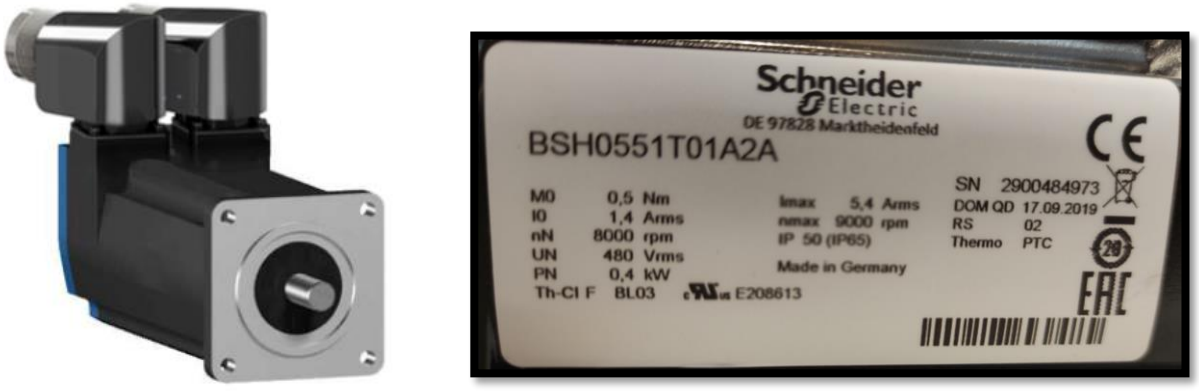


Figura 2-2: Servomotor BSH0551T01A2A. Placa de características

Se trata de un servomotor AC que puede alcanzar 9000 rpm. El servomotor viene con un encoder instalado en el eje que proporciona la información de la posición y velocidad del motor en todo momento. Se trata de un encoder angular HIPERFACE® Monovuelta SinCos (mezcla entre encoder absoluto e incremental). Este tipo de encoder utilizan señales senoidales/cosenoidales para alcanzar una alta resolución en la medida (± 0'0222º) [6].

2.1.2 Servo drive

El servo drive es el amplificador electrónico que se utiliza para alimentar al servomotor y controlarlo en posición, velocidad y/o par. Realiza este control actuando, en último lugar, sobre la corriente que debe ser inyectada a las bobinas del motor en cada instante (ver apartado 2.11). Para el presente proyecto se hace uso del modelo LXM32MU90M2 de *Schneider Electric*.

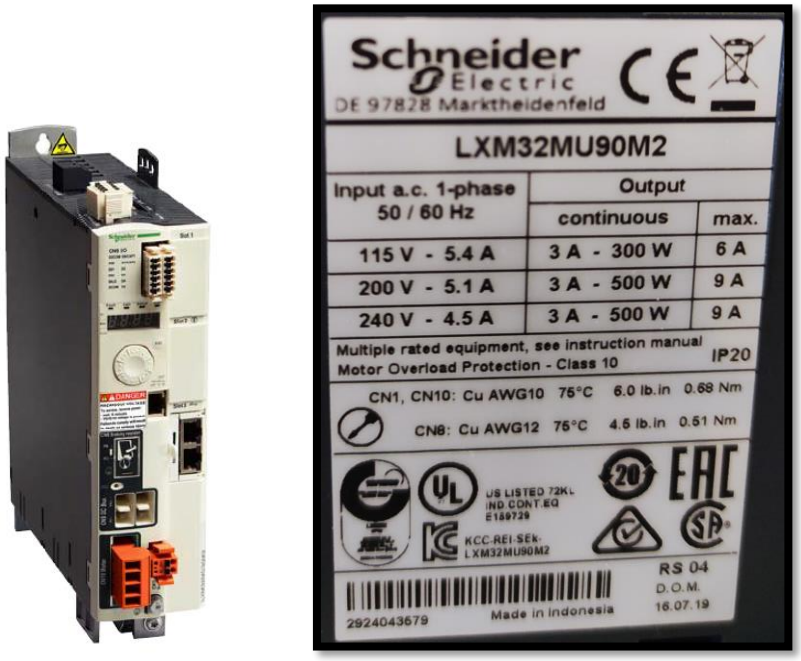


Figura 2-3: Servo drive LXM32MU90M2. Placa de características

El servo drive cuenta con un "diccionario" de parámetros cuyo ajuste permite configurar el funcionamiento del servo accionamiento. El presente capítulo hace referencia a ellos en numerosas ocasiones por lo que se ha decidido resaltarlos en *azul* para distinguirlos claramente del resto del texto. Para consultar todos los detalles de cualquiera de los parámetros (Descripción, longitud en bytes que ocupa, dirección de acceso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5].

2.1.3 Conexionado

A continuación, se muestra un típico de conexionado para un sistema de servo accionamiento con señales de entrada y salida digitales.

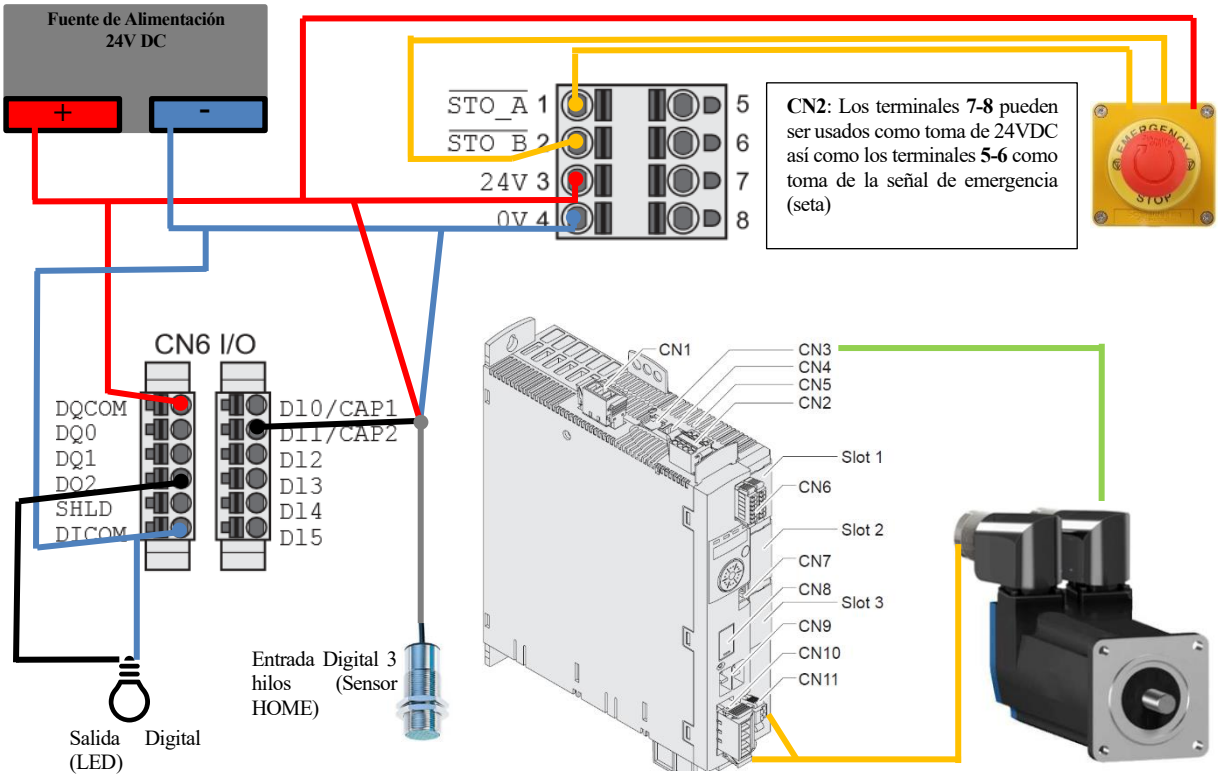


Figura 2-4: Diagrama de conexionado de servo accionamiento con entradas y salidas digitales

El servomotor tiene dos conectores: uno para la conexión del motor y otro para la conexión del encoder integrado. Para el caso del modelo de servomotor del que se hace uso en el proyecto (Figura 2-2), ambos conectores son del tipo M23 [7].

El encoder se conecta al servo drive directamente por el puerto CN3 a través de un cable específico con conectores M23 de 12 polos (Figura 2-5, derecha) - RJ45.

Por su parte, el motor se conecta al servo drive a través de un multicable con conector M23 de 8 polos (Figura 2-5, izquierda) por el extremo del motor y de cables abiertos por el otro extremo. Las fases W, V y U (terminales 1, 3 y 4) del motor van conectadas al puerto CN10 del servo drive mientras que la alimentación del freno de parada (terminales A (+) y B (-)) se conecta al puerto CN11.

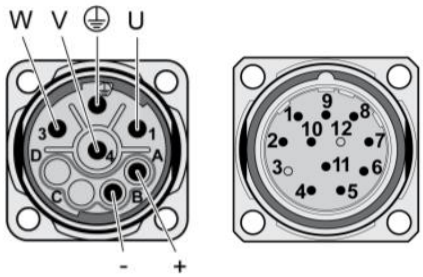


Figura 2-5: Conectores M23 del lado del servomotor

La etapa de potencia se alimenta de la red trifásica a través de un interruptor general de protección por el puerto CN1. La alimentación del control se toma de una fuente externa DC de 24V, protegida adicionalmente por un fusible, a través del puerto CN2.

Finalmente, cabe mencionar que se conectan las señales de entrada y salida digital a través del puerto CN6 y la seta de emergencia encargada de lanzar la función de seguridad STO[1] a través del puerto CN2.

[1] *Safe Torque Off*. Función que desconecta el par motor de forma segura a través de 2 entradas redundantes (STO_A y STO_B)

2.2 Estados de funcionamiento

Tras la conexión, el servo drive va pasando por una serie de estados operativos o de funcionamiento. En cada uno de ellos, el estado de la etapa de potencia y del modo de funcionamiento seleccionado podrá ser diferente. Por lo tanto, los estados de funcionamiento permiten al operador o al controlador que supervise la operación, conocer esa información para tomar acciones cuando sea necesario.
En el caso del servo drive LXM32M, se distinguen los siguientes estados de funcionamiento:

1. **Start** → se inicializa la electrónica.
2. **Not Ready To Switch On** → la etapa de potencia no está lista para la conexión.
3. **Switch On Disabled** → no se puede activar la etapa de potencia.
4. **Ready To Switch On** → la etapa de potencia esta lista para la conexión.
5. **Switched On** → se conecta la etapa de potencia.
6. **Operation Enabled** → la etapa de potencia esta activada. El modo de funcionamiento ajustado está activo.
7. **Quick Stop Active** → "Quick Stop" ejecutándose.
8. **Fault Reaction Active** → se ejecuta la reacción de error (ver apartado 2.2.1).
9. **Fault** → reacción de error finalizada. Etapa de potencia desactivada.

Las relaciones entre los estados de funcionamiento y las transiciones que llevan de unos a otros se ilustran en el siguiente diagrama de estado:

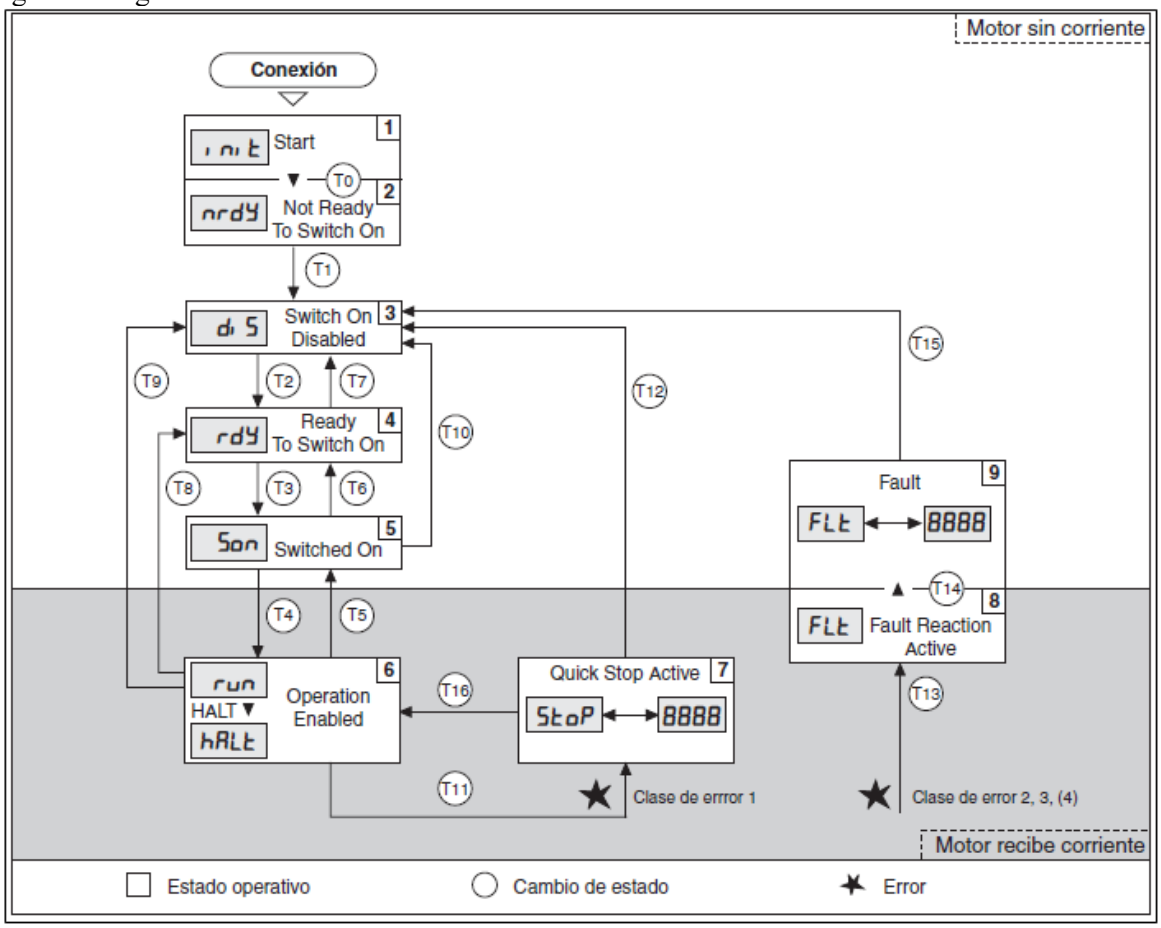


Figura 2-6: Diagrama de estados de funcionamiento [5].

Para el cambio de un estado de funcionamiento a otro (Tx[2]) basta con que se cumpla una de las condiciones establecidas:

- Ejecución de **comandos determinados mediante el control por bus de campo**. Todas las transiciones poseen uno, excepto T0, T1, T13 y T14. Los comandos se ejecutan a través del parámetro *DCOMcontrol*. La ejecución de un comando u otro dependerá del valor del parámetro bit a bit según lo mostrado en la Tabla 2-1. La "X" indica que el valor de dicho bit es indiferente para la transición.

[2] Con X → [1,16]

Comando	Transiciones	Bit 7 Fault Reset	Bit 3 Enable Operation	Bit 2 Quick Stop	Bit 1 Enable Voltage	Bit 0 Switch On
Shutdown	T2, T6, T8	0	X	1	1	0
Switch On	T3	0	0	1	1	1
Disable Voltage	T7, T9, T10, T12	0	X	X	0	X
Quick Stop	T7, T10, T11	0	X	0	1	X
Disable Operation	T5	0	0	1	1	1
Enable Operation	T4, T16	0	1	1	1	1
Fault Reset	T15, T16	0→1	X	X	X	X

Tabla 2-1: Combinaciones de bits del parámetro DCOMcontrol para transiciones entre estados de funcionamiento.

Además, cabe mencionar que en el modo de control por bus de campo (ver apartado 2.3) puede utilizarse el parámetro `_DCOMstatus` para obtener información acerca del estado operativo.

Es importante destacar que, tal como ocurre con `DCOMcontrol`, para algunos parámetros del servo nos interesa modificar el valor de cada uno de sus bits por separado. Sin embargo, se almacenarán como enteros con lo que será necesario realizar la conversión binario-decimal para configurarlos y decimal-binario para monitorizarlos.

- **Eventos de validación** (por ejemplo, para T0: "Sistema electrónico del equipo inicializado con éxito"). Todas las transiciones presentan uno, excepto T5, T6, T8.

- Por último, cuando el servo drive está en funcionamiento (estado 6, "Operation enabled"), es posible interrumpir el movimiento con una parada (deceleración) a través de un *Halt*. Puede activarse haciendo uso de una señal de entrada digital (ver apartado 2.3) o a través del bus de campo activando el bit 8 del parámetro `DCOMcontrol`.

2.2.1 Errores de funcionamiento

Los errores de funcionamiento que pueden darse (por ejemplo, error de temperatura de funcionamiento) se clasifican en **4 clases de error** cada una de ellas caracterizadas por la reacción que el servo drive experimenta.

- **Clase 1** → el movimiento se cancela con un "Quick Stop[3]".
- **Clase 2** → el movimiento se cancela con un "Quick Stop", se acciona el freno de parada[4] y se desactiva la etapa de potencia.
- **Clase 3** → la etapa de potencia se desactiva inmediatamente sin parar previamente el motor.
- **Clase 4** → la etapa de potencia se desactiva inmediatamente sin parar previamente el motor. Además, el error solo puede resolverse desconectando el servo drive de su fuente de alimentación.

Cuando se produce un error, el servo drive cancelará el movimiento en curso y ejecutará la reacción que corresponda según la clase de error que se haya producido. Tras finalizar la reacción, se debe reiniciar el mensaje de error (T15, T16) a través del bus de campo ("Fault Reset", Tabla 2-1) o a través de una entrada digital.

[3] "Quick Stop" hace referencia a una parada del motor mediante una rampa de decremento de velocidad o par.
[4] El freno de parada del motor tiene la función de mantener la posición del motor con la etapa de potencia desactivada incluso aunque se ejerzan fuerzas externas.

2.3 Canales de acceso y modos de control

El LXM32M dispone de 4 canales de acceso distinto a través de los cuales puede ser controlado/configurado. Estos canales son:

- **HMI integrada**: brinda la oportunidad de configurar los distintos parámetros, iniciar el modo de funcionamiento manual (apartado 2.5), realizar un diagnóstico básico (consultar valores de ciertos parámetros o códigos de error) y realizar un Auto ajuste del regulador (apartado 2.11.2)

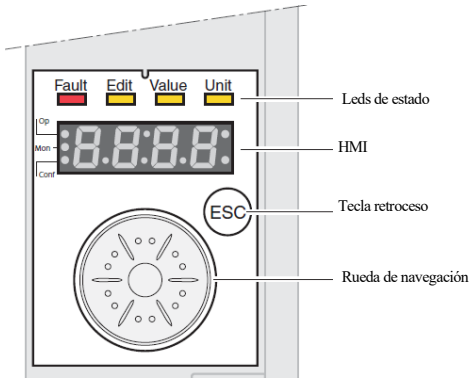


Figura 2-7: HMI Integrada del LXM32M.

- **Software de puesta en marcha**: Ver Capítulo 1.

- **Señales de entrada digitales**: el servo accionamiento permite configurar distintas funciones para cada una de las entradas digitales. Algunas de ellas son:
  - **LIMP, LIMN y REF**: Señales de final de carrera e interruptor de referencia.
  - **Enable**: Activar el modo de funcionamiento configurado.
  - **Fault Reset**: reiniciar mensaje de error.
  - **Halt**: Interrupción de movimiento con parada.
  - **Operating Mode Switch**: Alternar entre dos modos de funcionamento preestablecidos.

  La configuración de las funciones de entrada de señal para cada una de las entradas se realiza a través de los parámetros *IOfunct_DIX* donde X es el número de la entrada.

- **Señales de entrada analógica**: permiten la configuración de ciertos parámetros dentro de algunos de los modos de funcionamiento disponibles como el *Electronic Gear*.

- **Bus de campo:** permite configurar los distintos parámetros, cambiar entre estados de funcionamiento, activar los distintos modos de operación, realizar un diagnóstico y ajustar el regulador.

A través de los parámetros *HMILocked* y *AcessLocked* se puede bloquear el acceso a través de la HMI integrada y otorgar acceso exclusivo a través del bus de campo respectivamente. En cualquier caso, las funciones de entrada de señal *Halt, LIMP, LIMN, REF y Fault Reset* tendrán acceso siempre.
Cabe mencionar que, al igual que existen funciones de entrada de señal, el servo drive permite configurar una serie de funciones de salida de señal que permiten monitorizar la operación del motor (estado activo, seguimiento de la posición, seguimiento de la velocidad, etc.). Su configuración se realiza a través de los parámetros *IOfunct_DQX* donde X es el número de la salida.

Por otro lado, existen dos modos de control:

- **Modo de control local**: Los cambios entre los distintos modos de funcionamiento (apartado 2.4) se realizan a través de las entradas de señales digitales. La configuración de los parámetros necesarios se realizará, típicamente, a través de la HMI integrada o a través del Software de puesta en marcha.

- **Modo de control bus de campo**: Los cambios entre los distintos estados de funcionamiento (apartado 2.2) y modos de funcionamiento (apartado 2.4) y la configuración de los distintos parámetros, se realizan a través de un bus de campo desde controlador electrónico remoto. En cualquier caso, las entradas físicas pueden seguir siendo utilizadas (si no se bloquean con *AcessLocked*)

2.4 Modos de funcionamiento

El servo drive posee 8 modos de funcionamiento distintos:

- JOG
- Electronic Gear
- Profile Torque
- Profile Velocity
- Profile Position
- Interpolated Position
- Homing
- Motion Sequence

La inicialización y cambio entre estos modos de funcionamiento u operación se puede llevar a cabo a través del bus de campo mediante el parámetro `DCOMopmode` y algunos bits del parámetro `DCOMcontrol` (pág. 59, [8]). Llegados a este punto, es interesante mencionar que el servo drive procesa la información de posición y velocidad del servomotor con la información que le llega del encoder.

- La **posición**, se traduce a unas unidades de usuario *usr_p* mediante un factor de escala que se define mediante los parámetros `ScalePOSnum` y `ScalePOSdenom`.

$$\frac{ScalePOSnum}{ScalePOSdenom} = \frac{Revoluciones\ del\ motor}{Unidades\ de\ usuario\ (usr_p)}$$

- En el caso de la **velocidad**, las unidades de usuario *usr_v* son por defecto iguales a las rpm del motor, aunque esto puede cambiarse con `ScaleVELnum` *y* `ScaleVELdenom`.

2.5 Modo JOG

2.5.1 Descripción

El modo de funcionamiento JOG es el más básico de todos. Consiste en un manejo manual del servomotor en el que se pueden establecer las siguientes condiciones de movimiento:

- Dirección
- Velocidad (rápida o lenta, ambas parametrizables)
- Modo de accionamiento (continuo o paso a paso)

En el **modo continuo,** mientras la señal de dirección esté activada, el servomotor se mantendrá en movimiento en la dirección establecida por dicha señal (Figura 2-8).

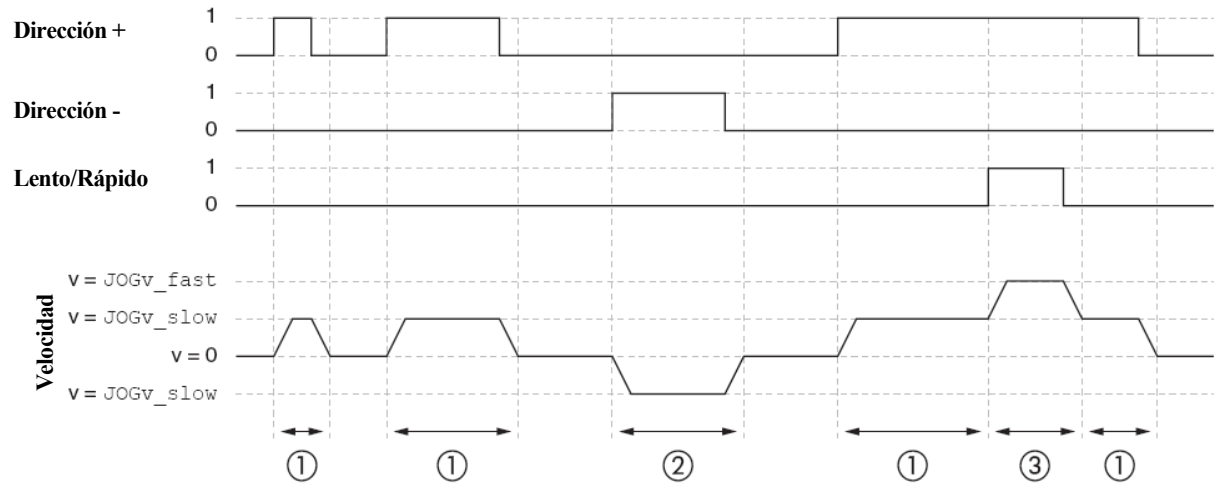


Figura 2-8: Ejemplo movimiento continuo [5]

1. Movimiento lento en dirección positiva
2. Movimiento lento en dirección negativa
3. Movimiento rápido en dirección positiva

En el **modo paso a paso,** si la señal de dirección se activa brevemente, el servomotor efectuará un movimiento de x pasos. Si la señal se mantiene activada tras ese movimiento, después de una breve pausa, el servomotor iniciará un movimiento continuo en la dirección establecida hasta que la señal de dirección conmute de nuevo (Figura 2-9).

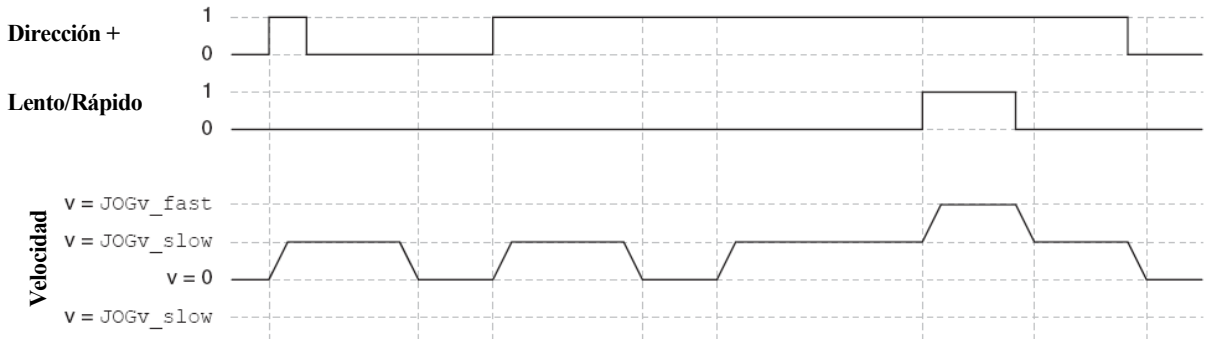


Figura 2-9: Ejemplo movimiento paso a paso [5]

1. Movimiento de x pasos tras una activación breve de la señal de dirección.
2. Tras eso, el motor se detiene a la espera de otra activación.
3. Se activa de nuevo la señal de dirección (mediante *JOGactivate*) con lo que:
   3.1. Se realiza un movimiento de x pasos (*JOGstep*)
   3.2. Como la señal sigue activada, el motor detiene el movimiento un tiempo establecido (*JOGtime*)
   3.3. Se inicia el modo de movimiento continuo hasta que se desactiva la señal de dirección.

2.5.2 Configuración de parámetros

En este apartado se pretende dar una idea básica sobre las opciones de configuración disponibles en este modo de funcionamiento. A través del modo de control bus de campo están disponibles los siguientes parámetros (Figura 2-10):

- *JOGactivate*: señal activación del movimiento que configura la dirección (positiva o negativa) y la velocidad (rápida o lenta).
- *JOGv_slow* / *JOGv_fast*: configuran los valores numéricos de las velocidades "lenta" y "rápida" respectivamente.
- *JOGmethod*: selecciona el modo de accionamiento deseado.
- *JOGstep*: el número x de pasos (en unidades de usuario) efectuados en el modo paso a paso.
- *JOGtime*: tiempo de espera tras movimiento paso a paso para comenzar movimiento continuo en el modo paso a paso.
- *RAMP_v_acc*, *RAMP_v_dec* y *RAMP_v_max*: para configurar el perfil de velocidad en el movimiento.

Para consultar todos los detalles de cualquiera de los parámetros (descripción, longitud en bytes que ocupa, dirección de acceso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

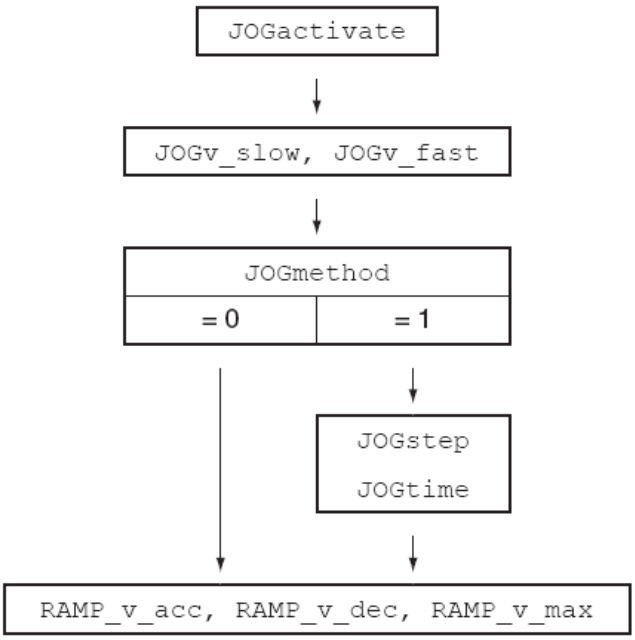


Figura 2-10: Parámetros modo JOG [5]

2.6 Modo Electronic Gear

2.6.1 Descripción

En este modo de funcionamiento, el servomotor actúa como un motor paso a paso manejado por dos señales rápidas externas (trenes de pulsos). Estas señales rápidas (conectadas al puerto PTI del servo drive) pueden ser de distinto tipo[5], pero en esencia, cada pulso marcará un movimiento de x pasos en una dirección de movimiento determinada.
Cada pulso de las señales externas se traducirá en un número de incrementos del motor mediante una **relación de transmisión** configurable.

$$Relación\ de\ transmisión\ (Gear\ Factor) = \frac{Incrementos\ de\ motor}{Incrementos\ de\ la\ señal\ de\ referencia}$$

El movimiento se puede llevar a cabo con **sincronización de posición o de velocidad** con respecto a las señales externas.
En el modo de sincronización de posición estará disponible, adicionalmente, un **movimiento de offset**: Cuando se dé la señal de activación oportuna (independiente de las señales externas), se iniciará un movimiento único con una cantidad parametrizable de incrementos.
Si las señales rápidas provienen del encoder de otro servomotor este modo de funcionamiento actúa, como su nombre indica, como un "engranaje electrónico" que relaciona directamente el movimiento de un servomotor con el movimiento del otro (sincronismo).

2.6.2 Configuración de parámetros

En este apartado se pretende dar una idea básica sobre las opciones de configuración disponibles en este modo de funcionamiento. A través del modo de control bus de campo están disponibles los siguientes parámetros (Figura 2-11):

- *GEARselect*, *GEARratio*, *GEARnum*, *GEARdenom*, *GEARnum2*, *GEARdenom2*: permiten configurar la relación de transmisión.
- *GEARreference* y *GEARposChgMode*: configuran el modo de sincronización (posición o velocidad).

[5] Las señales pueden ser A/B (cuatro estados), P/D (pulsos/dirección), CW/CCW (dir +/dir -).

- *OFS_xxxx:* parámetros que sirven para configurar los movimientos de offset disponibles en el modo de sincronización de posición.
- *GEARpos_v_max:* limitación de velocidad en el modo de sincronización de posición.
- *GEARdir_enabl:* limitación de dirección de movimiento única.
- *RAMP_v_xxxx:* parámetros que sirven para configurar el perfil de velocidad (aceleración/deceleración) en el modo de sincronización de velocidad.

Para consultar todos los detalles de cualquiera de los parámetros (descripción, longitud en bytes que ocupa, dirección de acceso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

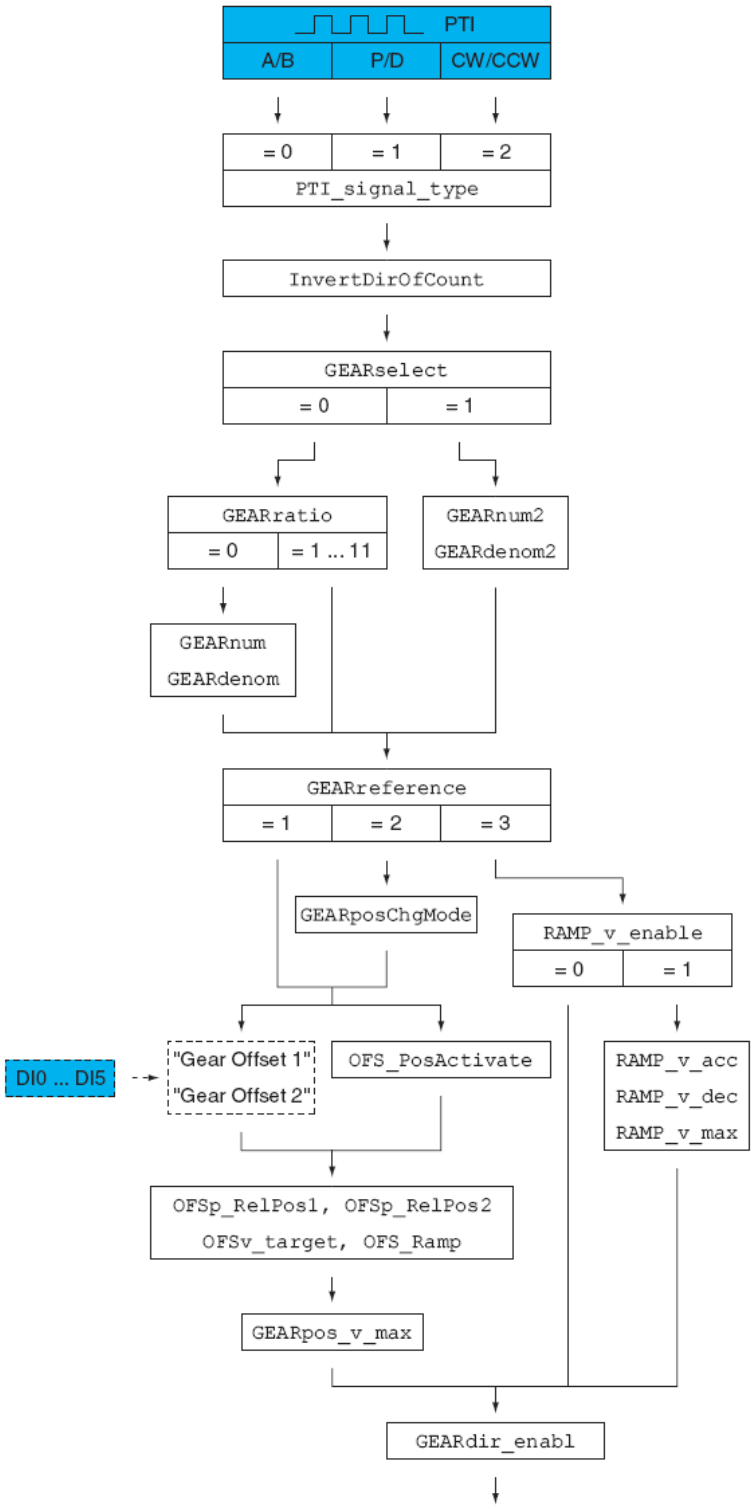


Figura 2-11: Parámetros modo Electronic Gear [1]

2.7 Modo Profile Torque, Velocity, Position

2.7.1 Descripción

En estos modos de funcionamiento se ejecuta un movimiento con un par, una velocidad o una posición de referencia determinada.
En el modo **Profile Position** cabe diferenciar 2 métodos diferentes:

- **Movimiento relativo**: La posición de destino establecida está referenciada a la posición en la que se encuentre en ese momento el motor.
- **Movimiento absoluto**: La posición de destino establecida está referenciada al punto cero, el cual debe estar previamente establecido por el modo Homing (apartado 2.9).

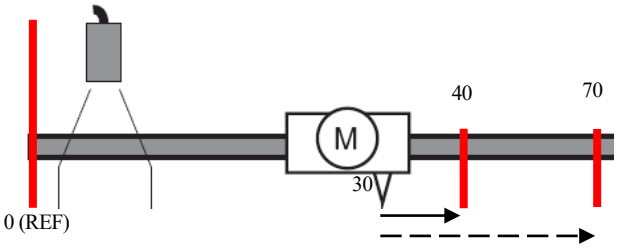


Figura 2-12: Diferencia entre mov. relativo (discontinua) y mov. absoluto (continua) ante una referencia de 40.

En el **modo Profile Torque y Profile Velocity**, el par o velocidad de consigna se puede establecer a través de las señales analógicas disponibles en el servo drive o a través de un parámetro interno.

2.7.2 Configuración de parámetros

En este apartado se pretende dar una idea básica sobre las opciones de configuración disponibles en estos modos de funcionamiento A través del modo de control bus de campo están disponibles los siguientes parámetros (Figura 2-13, Figura 2-14 y Figura 2-15):

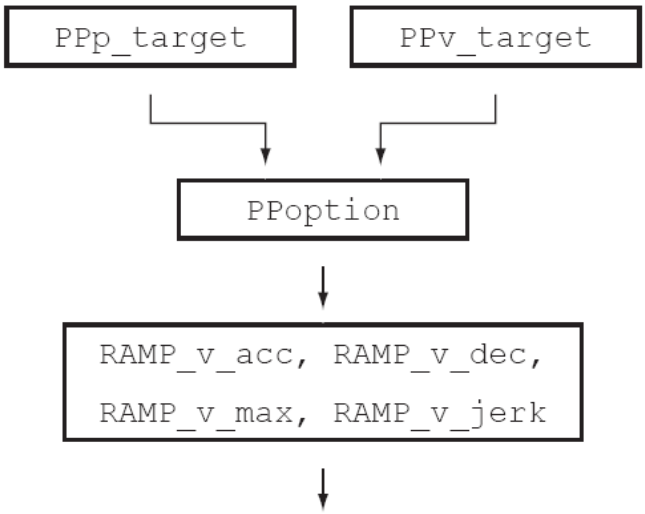


Figura 2-13: Parámetros modo Profile Position [1]

- *PPp_target* y *PPv_target*: valores de posición de destino y velocidad de movimiento.
- *PPoption*: permite elegir entre movimiento relativo y absoluto.
- *RAMP_v_xxx*: permiten configurar el perfil rampa de velocidad para el movimiento.

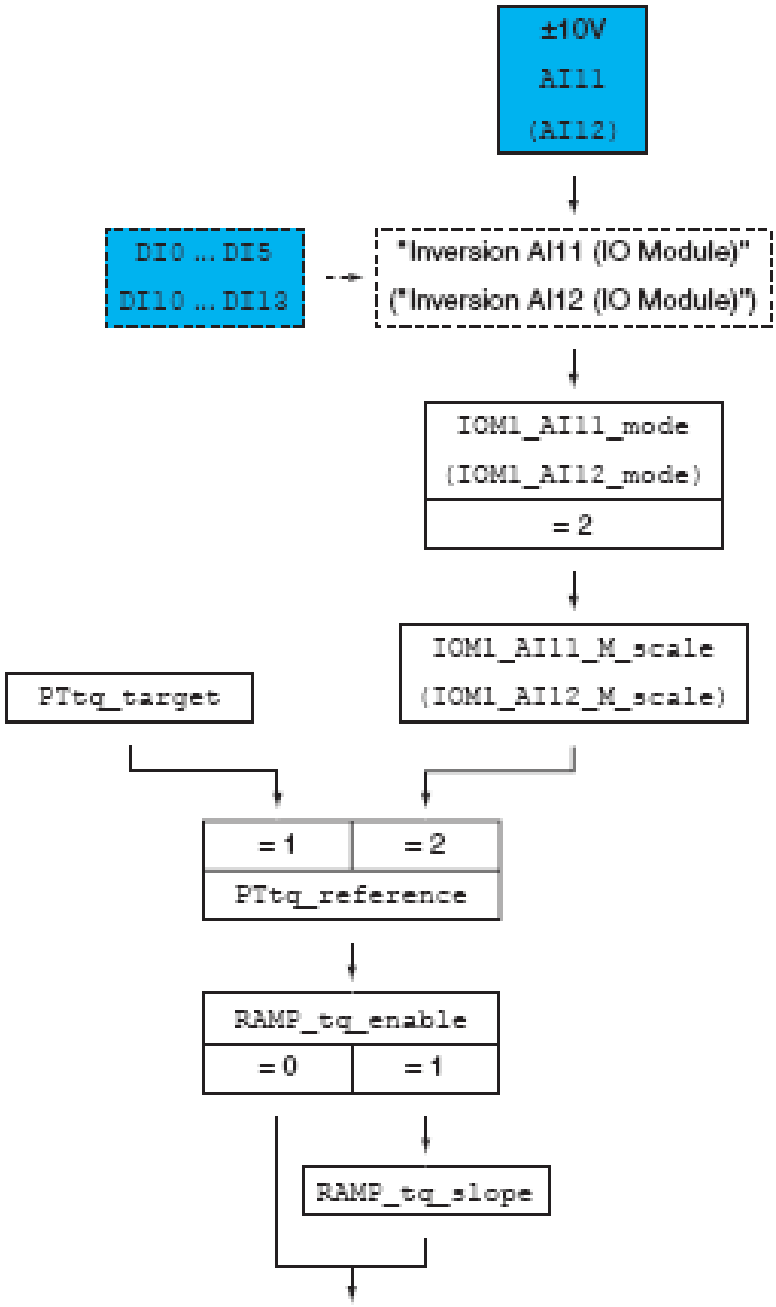


Figura 2-14: Parámetros modo Profile Torque [5]

- *IOM1_AI1X_xxxx*: permiten la configuración de las señales analógicas de entrada y su traducción al valor de consigna.
- *PTtq_target*: valor consignado de par (par objetivo).
- *PTtq_reference*: escoge la fuente de la consigna de par.
- *RAMP_tq_xxx*: permiten configurar un perfil rampa de par para el movimiento (se alcanza el par destino progresivamente).

Para consultar todos los detalles de cualquiera de los parámetros (descripción, longitud en bytes que ocupa, dirección de acesso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

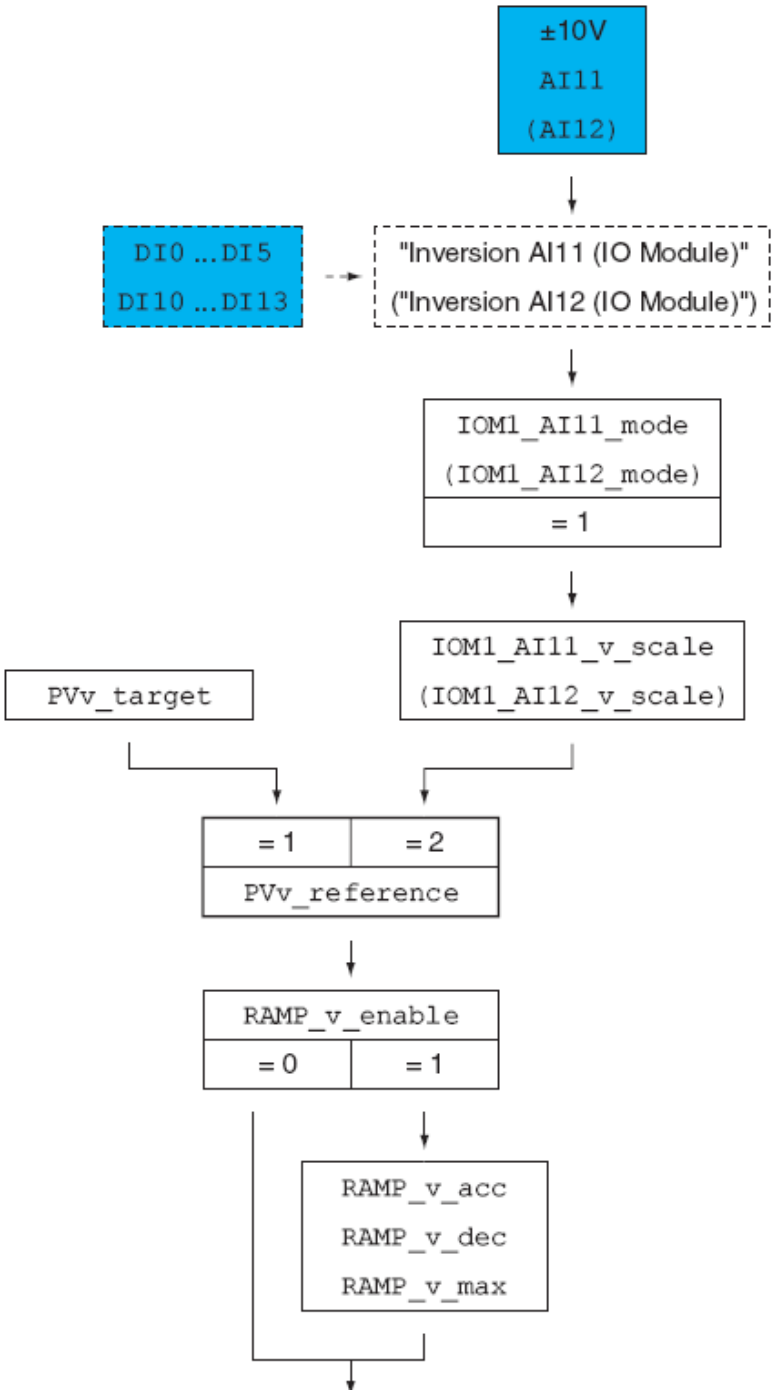


Figura 2-15: Parámetros modo Profile Velocity [1]

- *IOM1_AI1X_xxxx*: permiten la configuración de las señales analógicas de entrada y su traducción al valor de consigna.
- *PVv_target*: valor consignado de velocidad (velocidad objetivo).
- *PVv_reference*: escoge la fuente de consigna de velocidad.
- *RAMP_v_xxxx*: permiten configurar un perfil rampa de velocidad para el movimiento (se alcanza la velocidad destino progresivamente).

Para consultar todos los detalles de cualquiera de los parámetros (descripción, longitud en bytes que ocupa, dirección de acesso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

2.8 Modo Interpolated position

2.8.1 Descripción

En este modo de funcionamiento, el servo drive recibe a través del bus de campo una serie de posiciones de referencia que debe alcanzar cíclicamente. El protocolo de comunicaciones empleado debe ser CANOpen (Anexo A). De forma esquemática, el funcionamiento completo sería el siguiente:

1. El servo drive recibe, a través del bus (con un PDO) una posición de referencia a alcanzar ((1) en Figura 2-16).
2. La señal de cíclica de sincronismo del bus (SYNC, mandada por un controlador remoto) marcará el instante de inicio del movimiento hacia la posición de referencia recibida anteriormente.
3. El servo drive ajustará el movimiento (realizando una interpolación fina con un paso de 250 µs) para que se alcance dicha posición de referencia en un ciclo de la señal de sincronismo (2)
4. Mientras ocurre este movimiento, el servo drive podrá recibir otra posición de referencia hacia la cual se iniciará un movimiento en el próximo ciclo de la señal de sincronismo (una vez alcanza la posición de referencia recibida antes que esta última) (3).

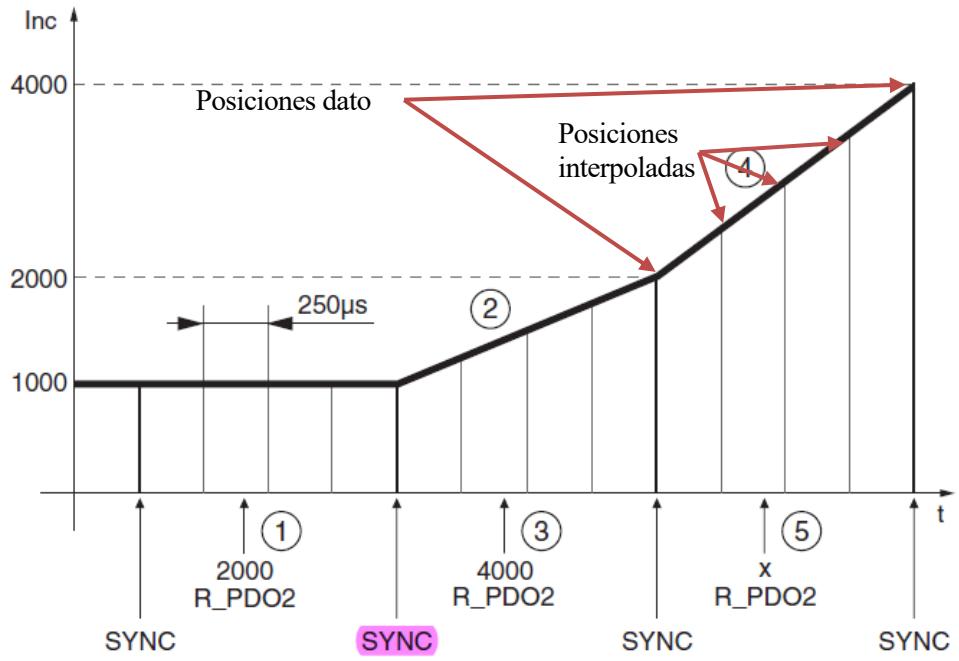


Figura 2-16: Ejemplo funcionamiento Modo Interpolated Position

Dado que el ciclo de la señal de sincronismo puede oscilar entre 1 y 20 ms, si la diferencia entre la posición real del motor y la posición referenciada es muy grande, el servo drive indicará error de seguimiento.
Este modo de funcionamiento puede utilizarse para coordinar/sincronizar varios ejes. Basta con activarlo simultáneamente en todos ellos (a través de un controlador remoto). De esta forma, todos los ejes generarán la trayectoria que les toca al mismo tiempo para resultar así en el movimiento coordinado deseado. Sin embargo, en este modo no hay una relación *Maestro*/*Esclavo* estricta entre los ejes (no hay un eje cuya posición **real** dependa de la posición de otro eje).

2.8.2 Configuración de parámetros

En este apartado se pretende dar una idea básica sobre las opciones de configuración disponibles en este modo de funcionamiento. A través del modo de control bus de campo (CAN) están disponibles los siguientes parámetros:

- `SyncMechStart:` debe establecerse a "2" para activar el mecanismo de sincronismo CANOpen necesario para este modo.
- `SyncMechTol:` establece la tolerancia de sincronización (la sincronización del movimiento con la señal SYNC no tiene que ser exacta)

- *IP_IntTimPerVal y IP_IntTimInd:* establecen la duración del ciclo de la señal de sincronismo (entre 1 y 20 ms). Deberá tenerse en cuenta la velocidad y carga de transmisión del bus. La misma duración debe estar configurada en el controlador externo que se utilice.
- *IPp_Target:* parámetro por el que se van recibiendo las posiciones de referencia objetivo.

Para consultar todos los detalles de cualquiera de los parámetros (descripción, longitud en bytes que ocupa, dirección de acesso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

2.9 Modo Homing

2.9.1 Descripción

Con este modo de funcionamiento se establece, mediante un movimiento de referencia (Home), el "punto cero". El "punto cero" relaciona una posición mecánica del motor con la posición numérica "0" del mismo y sirve de punto de partida para la localización de las distintas posiciones absolutas del motor.
Existen 4 métodos distintos para la realización del movimiento de referencia:

- **Final de carrera**
  En este método se realiza un movimiento desde la posición real actual, en dirección positiva o negativa, hasta llegar a uno de los finales de carrera establecidos (paso 1, Figura 2-17). Al activarse el final de carrera, el motor se detiene y se produce un movimiento de retorno en la dirección contraria hasta que el mismo final de carrera se desactive (paso 2). Desde ese punto, el motor se mueve una determinada distancia parametrizable o hasta el siguiente pulso índice[6] (paso 3). Ese último punto alcanzado es el "punto cero".

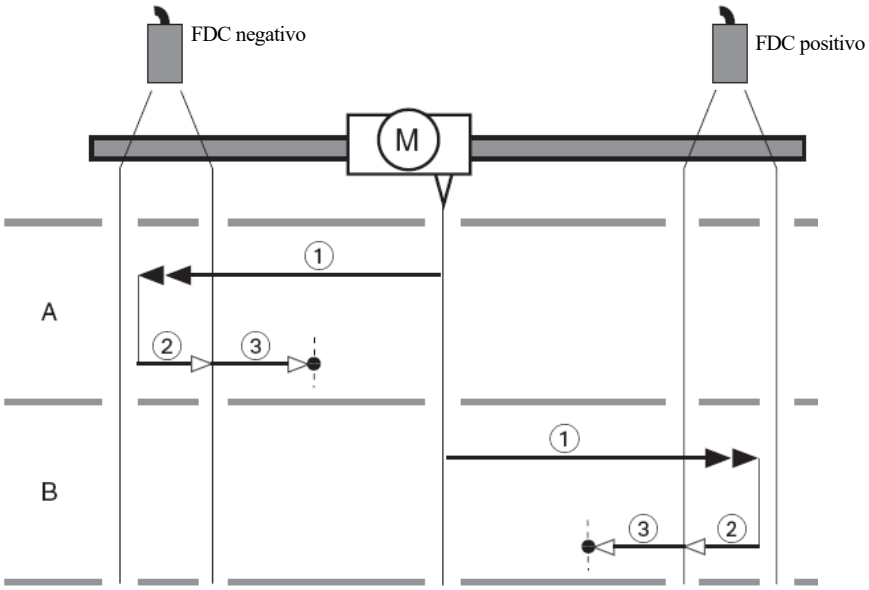


Figura 2-17: Ejemplo con finales de carrera [5]

- **Interruptor de referencia**
  En este método se realiza un movimiento desde la posición real actual hasta el interruptor de referencia establecido (paso 1, Figura 2-18). Al activarse dicho interruptor, el motor se detiene y se produce un segundo movimiento hasta uno de los puntos de conmutación del interruptor (paso 2). Desde ese punto, el motor se mueve una determinada distancia parametrizable o hasta el siguiente pulso índice (paso 3). Ese último punto alcanzado es el "punto cero".

[6] El pulso índice es un pulso determinado y único del encoder. (p.e establecido para que algo instalado sobre el motor este en una orientación deseada)

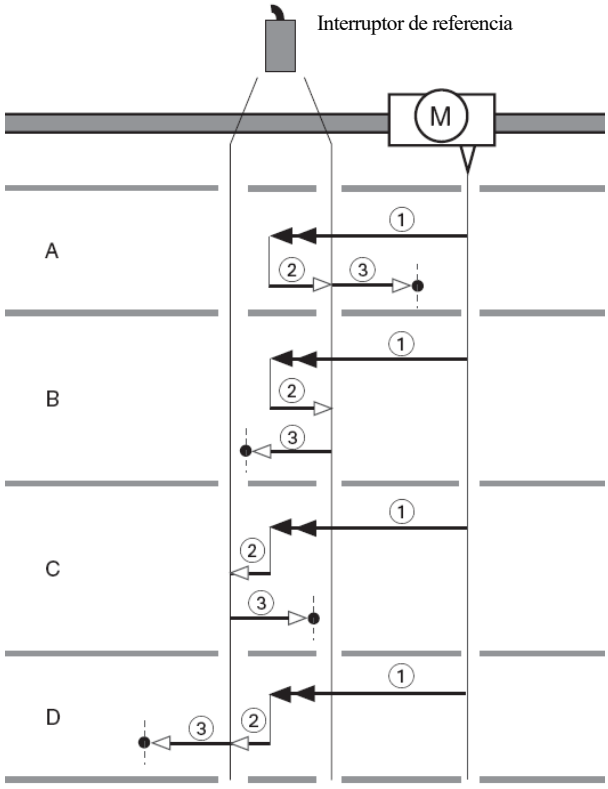


Figura 2-18 Ejemplo con interruptor de referencia [5]

En el caso del interruptor de referencia, al no tratarse de un límite mecánico de movimiento en ninguna dirección (p.ej el final de carrera negativo indica que el motor no puede moverse en dirección negativa más allá), se generan 4 variantes de movimiento de referencia (A, B, C y D)

- **Pulso Índice**
  En este método, se realiza un movimiento desde la posición real hasta el siguiente pulso índice que se convertirá en el "punto cero".

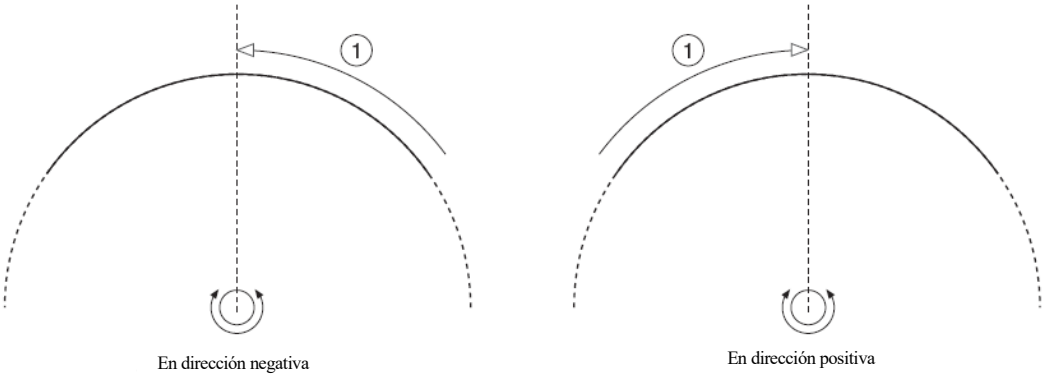


Figura 2-19: Movimientos de referencia al pulso índice [5]

- **Establecimiento de medida**
  Con el establecimiento de medida, la posición real actual se convierte en una posición real determinada de forma que quede establecida la posición real del "punto cero" en base a ello.

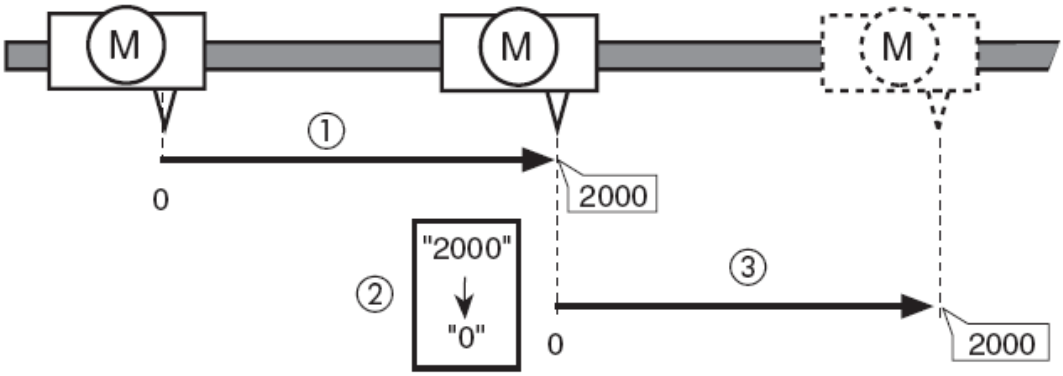


Figura 2-20:Ejemplo de establecimiento de medida [5]

En el ejemplo de la Figura 2-20**,** en el paso (2) se realiza un establecimiento de medida de forma que la posición del motor en ese instante (2000) se convierte en el nuevo "punto cero".
Es importante destacar que para poder llevar a cabo un establecimiento de medida el servomotor debe estar completamente detenido. El establecimiento de medida puede combinarse con cualquiera de los otros tres métodos anteriores.

Sea cual sea el método empleado, el movimiento de referencia debe completarse sin ser interrumpido de ninguna manera para el correcto establecimiento del "punto cero".

2.9.2 Configuración de parámetros

En este apartado se pretende dar una idea básica sobre las opciones de configuración disponibles en este modo de funcionamiento. A través del modo de control bus de campo están disponibles los siguientes parámetros:

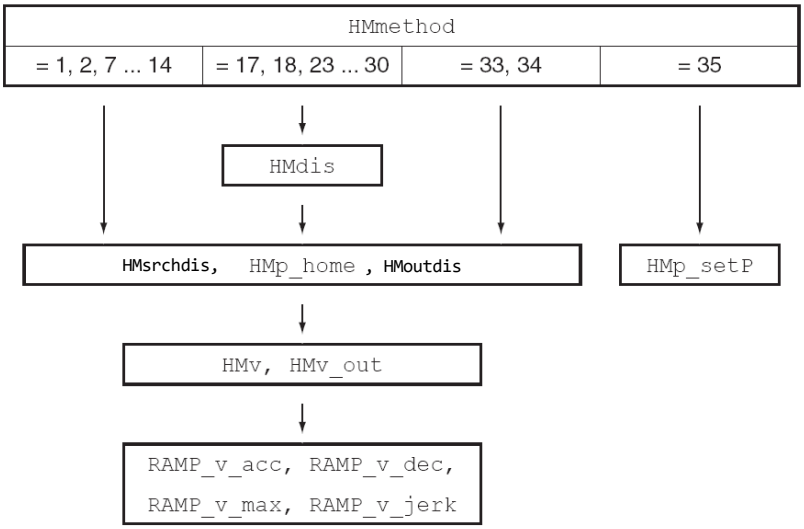


Figura 2-21: Parámetros modo Homing [5]

- *HMmethod*, *HMprefmethod*: método de referenciado. Contempla las diferentes variantes expuestas en el apartado anterior.
- *HMdis*: distancia al punto de conmutación para el paso 3 del movimiento de referenciado en los casos de final de carrera e interruptor de referencia.
- *HMp_home*: valor de posición real otorgado a la posición mecánica final alcanzada con el movimiento de referencia seleccionado (establecimiento de medida).
- *HMoutdis*: distancia máxima que se recorrerá para buscar el punto de conmutación antes de generar un error.
- *HMsrchdis*: distancia máxima que se recorrerá tras sobrepasar el interruptor antes de generar un error.

- *HMv y HMv_out*: ajuste de velocidades para el movimiento de búsqueda (paso 1) y para el movimiento de retorno (paso 2) respectivamente.
- *RAMP_v_xxxx*: permiten configurar un perfil rampa de velocidad para el movimiento de referencia seleccionado.
- *HMp_setP*: posición real otorgada a la posición mecánica real del motor en el momento del establecimiento de medida (solo para dicho modo).

Para consultar todos los detalles de cualquiera de los parámetros (sescripción, longitud en bytes que ocupa, dirección de acesso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

2.10 Modo Motion Sequence

2.10.1 Descripción

El modo Motion Sequence realiza movimientos basados en unos registros de datos parametrizables. Estos registros de datos contienen información sobre el tipo de movimiento y sobre valores consignados de par, velocidad y/o posición.
Existen 6 tipos de registros de datos:

- Movimiento a una posición consignada (movimiento relativo o absoluto).
- Movimiento con una velocidad consignada.
- Homing.
- Repetición de otros registros de datos (para generar secuencias).
- Movimiento Electronic Gear.
- Modificar valor de parámetro.

Para cada tipo existirán un total de hasta cuatro posibles ajustes (aceleración, velocidad, posición destino, tipo de movimiento…)
Los movimientos configurados en estos registros de datos pueden sucederse entre ellos (formando una secuencia) de diferentes formas: cancelando el que se esté ejecutando e iniciando el siguiente, el siguiente se ejecuta tras finalizar el movimiento actual, adaptando la velocidad del movimiento actual a la velocidad del siguiente...
Para lo anterior, se establecen unas condiciones de transición determinadas que den paso de unos movimientos a otros: esperar un cierto tiempo, recibir el flanco de una señal…
Los registros de datos pueden configurarse de forma fácil e intuitiva con el software de puesta en marcha (ver capítulo 1) o bien desde un controlador remoto a través de un bus de campo.
Para ilustrar mejor la funcionalidad de este modo se aconseja consultar el ejemplo práctico expuesto en [9].

Campo	Significado		Valor	Valor
Todos los registros de datos				
Registro de datos 0				
Tipo de registro de datos			Move Absolute	
Ajuste A	Acceleration		300	Deactivate
Ajuste B	Velocidad		60	
Ajuste C	Absolute position		2000	Positive
Ajuste D	Decelaration		300	
Tipo de transición			Buffer And Start Next	
Siguiente registro de datos			1	
Condición de transición 1			Wait Time	
Valor para condición de transición 1			2000	Rising edge
Conexión lógica			Logical AND	
Condición de transición 2			Start Request Edge	
Valor para condición de transición 2			0	Rising edge

Figura 2-22: Ejemplo de registro de datos

2.10.2 Configuración de parámetros

En este apartado se pretende dar una idea básica sobre las opciones de configuración disponibles en este modo de funcionamiento. A través del modo de control bus de campo están disponibles los siguientes parámetros:

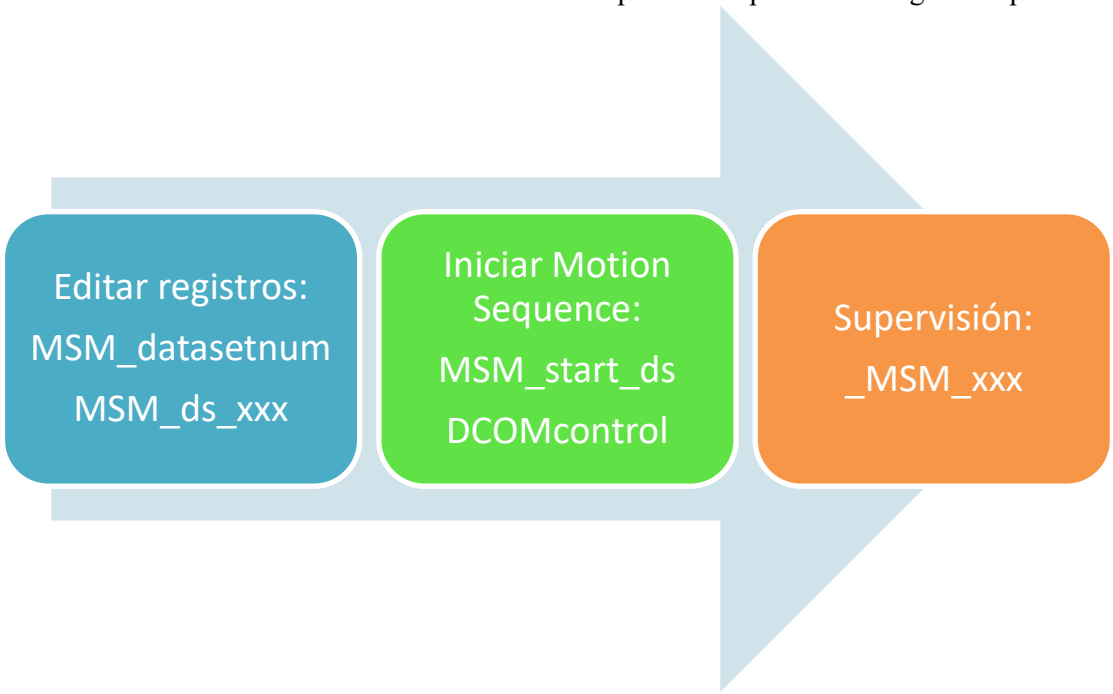


Figura 2-23: Parámetros modo Motion Sequence

- *MSM_datasetnum*: selección del registro de datos deseado para leer o escribir.
- *MSM_ds_xxxx*: permiten la edición de los diferentes campos de los registros de datos (*MSM_ds_setA*, *MSM_ds_transiti*, *MSM_ds_trancon1*, etc.)
- *MSM_start_ds*: Selección del registro de datos con el que se iniciará el movimiento.
- *DCOMcontrol:* algunos de sus bits se emplean para configurar la ejecución del modo de funcionamiento. El resto, como ya se mencionó anteriormente (apartado 2.2), son empleados para controlar el estado de funcionamiento del servo drive.
- *_MSM_xxx*: permiten obtener información acerca de errores y registros de datos utilizados durante el movimiento.

Bit	Valor
4	0→1: Inicio del registro seleccionado
5	0: Ejecución de un solo registro 1: Ejecución de secuencia de registros
6	1: Usar el registro seleccionado en *MSM_start_ds*

Tabla 2-2: Bits de *DCOMcontrol* usados para el modo Motion Sequence (Pag 70 en [8])

Para consultar todos los detalles de cualquiera de los parámetros (descripción, longitud en bytes que ocupa, dirección de acesso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

2.11 Regulador de control

2.11.1 Estructura del regulador

El servo accionamiento incluye una estructura de regulación en cascada con 3 controladores que se ajustan en el siguiente orden:

1. **Controlador de corriente:** de tipo Proporcional-Integral
2. **Controlador de velocidad:** de tipo Proporcional-Integral
3. **Controlador de posición:** de tipo Proporcional

La ejecución de los controladores se realiza en sentido inverso al de ajuste tal como se muestra en la Figura 2-24. Para un adecuado control de posición será necesario un buen ajuste del control de velocidad.

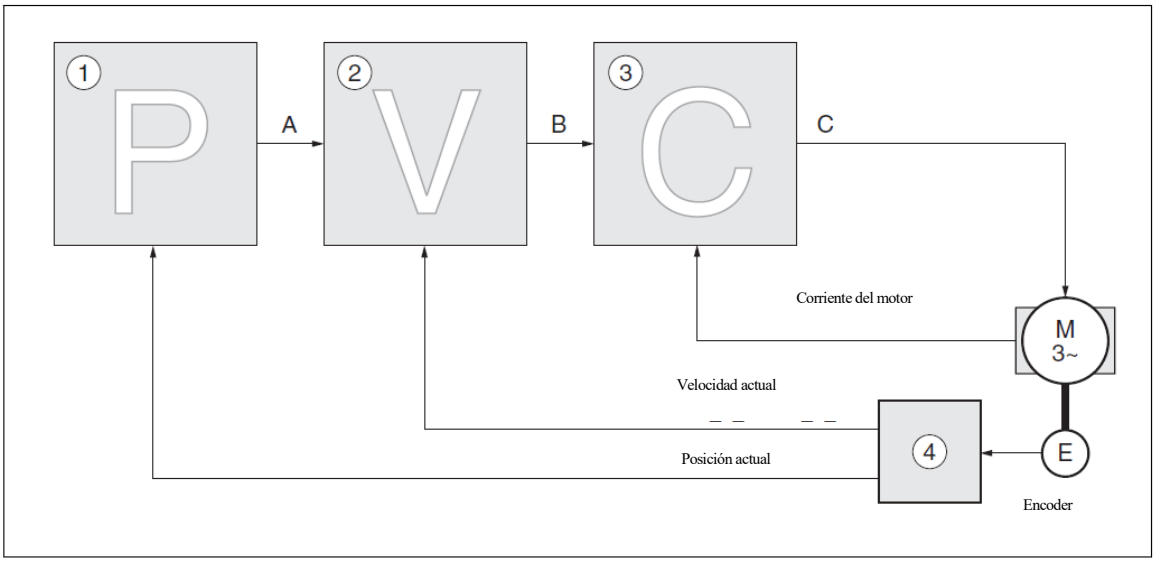


Figura 2-24: Estructura global del regulador

Cuando se establece en el motor un control en velocidad, el control en posición permanece desconectado. En el caso de establecer en el motor un control en par (corriente), el control en velocidad y posición permanecen desconectados. Por lo tanto, el servo drive realiza todo el control en posición, velocidad y par mediante el ajuste de la corriente que es inyectada a las bobinas del servomotor en todo momento.

2.11.2 Parámetros y métodos de ajuste.

El controlador de corriente se ajusta automáticamente utilizando los datos del motor conectado al servo drive.
Los controladores de velocidad (n) y posición (p) tienen principalmente 2 parámetros de control: Constante proporcional (*CTRLx_KPn*, *CTRLx_KPp*) y constante integral (*CTRLx_TNn*). Existen adicionalmente, otros parámetros para un ajuste avanzado de los controladores (*CTRLx_xxx*).
El **ajuste** de los controladores de velocidad y posición puede ser realizado de 3 formas distintas:

- **Automáticamente**: realizado completamente por el servo drive.

- **Semiautomáticamente**: realizado por el servo drive con ayuda del usuario el cual define una serie de parámetros (movimientos, mecánica…). Los parámetros asociados a este método de ajuste son los que comienzan de la forma *AT_xxx* (Figura 2-25).

- **Manual**: realizado completamente por el usuario. En los apartados 6.6.2 – 6.6.5 de [5] puede consultarse un procedimiento detallado para el ajuste manual de todos los parámetros.

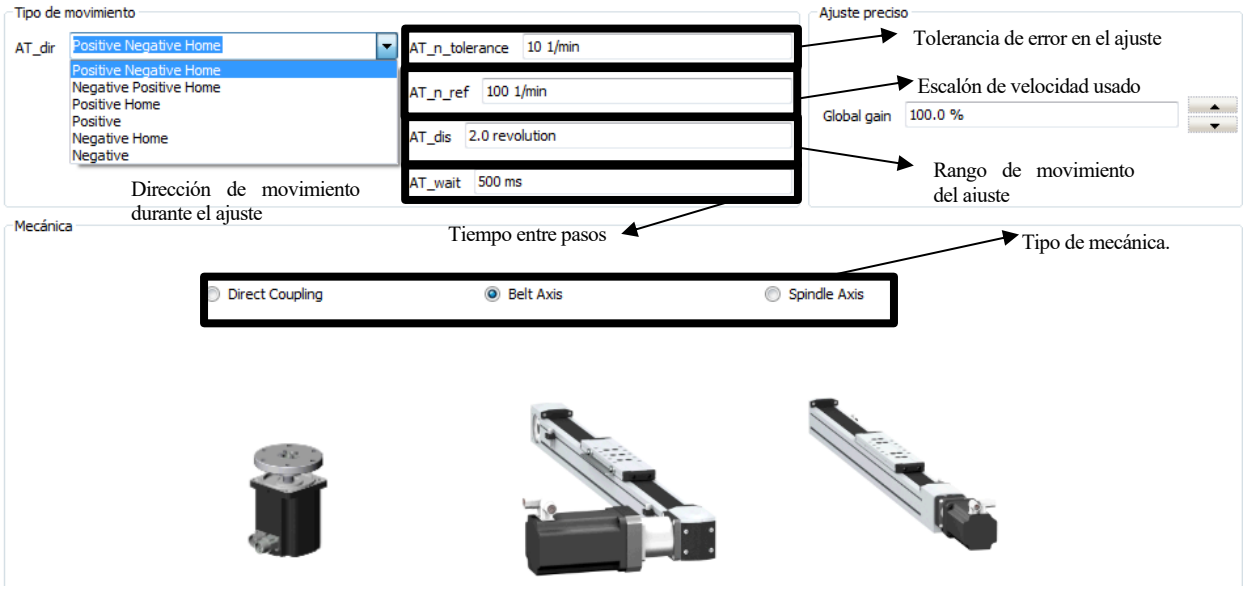


Figura 2-25: Configuración ajuste semiautomático en SoMove (Capítulo 1)

Es obligatorio realizar un ajuste del regulador la primera vez que se conecta el servo accionamiento o después de un reseteo de fábrica.
Independientemente del método de ajuste que se vaya a emplear, se recomienda llevarlo a cabo mediante el software de puesta en marcha.
Por último, también se recomienda al lector consultar [10] para una descripción más detallada sobre el ajuste del regulador.

3 ALTERNATIVAS DE CONTROL

Como ya se mencionaba en el apartado 2.3, los servomotores pueden ser programados y controlados **localmente** (a través de su propio servo drive). Sin embargo, en muchas ocasiones resulta necesario su integración en un sistema con otros sensores/actuadores resultando más interesante controlarlos **de forma remota**.
En particular, en numerosas aplicaciones industriales, se precisa que los servomotores sean capaces de trabajar de forma sincronizada entre ellos (o con otro tipo de motores) para realizar alguna tarea en concreto como:

- Alimentador de piezas
- Corte al vuelo (Figura 3-3)
- Selladoras
- Empacadoras
- Robots de posicionamiento (p. ej cartesianos)
- Embotellamiento de líquidos
- Máquinas CNC

Como controlador remoto se presentan aquí dos grandes alternativas:

- PLC
- Motion Controller

La elección de uno u otro dependerá de las necesidades de la aplicación, siendo necesario comprobar que el modelo escogido es capaz de cubrirlas con las funciones de control de movimiento que ofrece.
Por último, cabe mencionar que existen otros controladores electrónicos específicos que se encargan de controlar los servomotores que emplean las máquinas a las que están asociadas (p.ej para manipuladores robóticos o máquinas CNC).

3.1 Control remoto mediante Controlador de Movimiento

Un controlador de Movimiento o *Motion Controller* es un controlador especialmente diseñado para la coordinación, sincronización y creación de funciones de movimiento de varios ejes (servomotores). Además de conseguir tiempos de respuesta menores a los que conseguiríamos con un PLC, un *Motion Controller* posee un mayor abanico de posibilidades para el control de varios ejes. En particular, incluye la posibilidad de implementar perfiles CAM para la sincronización del movimiento de 2 o más ejes (*Electronic Camming*).

3.1.1 Perfiles CAM

Los perfiles CAM establecen una relación entre el movimiento de un eje o motor denominado *Maestro* y el movimiento de otro eje o servomotor denominado *Esclavo* o *Seguidor*. Dicho con otras palabras, un perfil CAM es el equivalente electrónico de una leva mecánica (Figura 3-1).

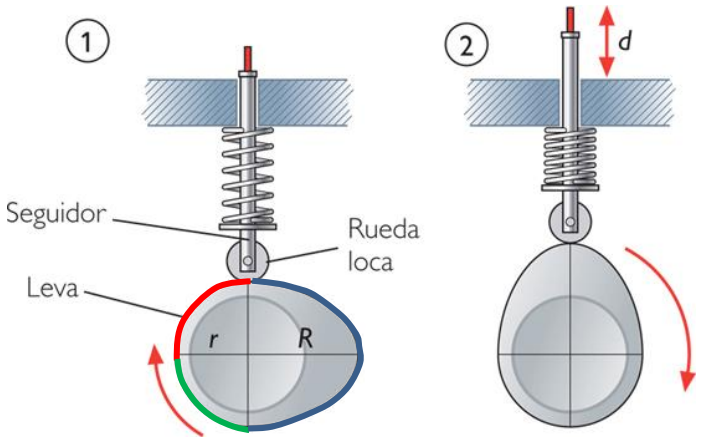


Figura 3-1: Ejemplo de leva mecánica[7].

Por un lado, se tiene un servomotor *Maestro* describiendo un movimiento determinado. Al mismo tiempo, cada posición del *Maestro* se traduce en una posición determinada del *Esclavo*. Por lo tanto, la velocidad de movimiento del *Maestro* determinará la velocidad del *Esclavo*.

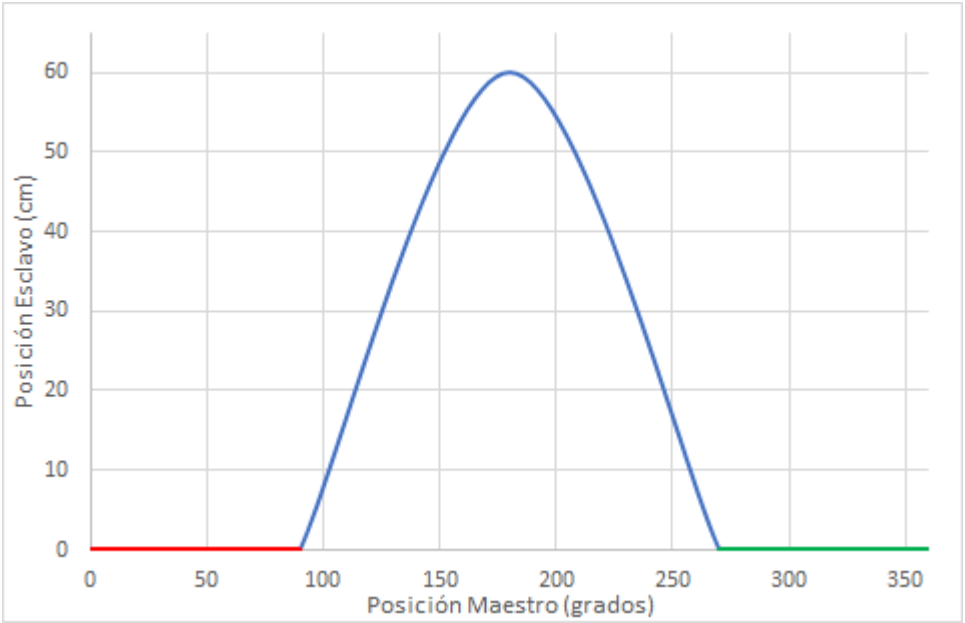


Figura 3-2: Perfil CAM de la leva de la Figura 3-1 considerando R=60cm.

Los softwares utilizados para la programación de los *Motion Controllers* (*EcoStruxure Machine Expert* para los controladores de *Schneider Electric*) incluyen la opción de diseñar estos perfiles CAM definiendo una serie de parámetros para cada uno de los puntos claves del perfil:

Variable	Descripción
Posición del *Maestro*	Eje X del Perfil
Posición del *Esclavo*	Eje Y del Perfil
Pendiente	Tangente al perfil dibujado X-Y. Representa la relación de la velocidad del *Maestro* con la velocidad del *Esclavo*
Interpolación	Interpolación utilizada entre los puntos X-Y definidos del perfil. El más típico es el polinómico de 5º grado

Tabla 3-1: Parámetros principales de un Perfil CAM.

[7] https://sites.google.com/site/mecanismos1oima03sap2/elementos-de-maquinas/leva (08/07/2021).
La leva representa el eje *Maestro* mientras que el Seguidor representa el eje *Esclavo*. El giro de la leva (grados) se traduce en un desplazamiento lineal del seguidor (cm).

Definido el perfil CAM, existirán distintas funciones (a nivel de programación del controlador de movimiento) para su implementación. Muchas de las aplicaciones industriales que requieren de servomotores sincronizados como embotelladoras, alimentadores de piezas o cortes al vuelo basan su funcionamiento en perfiles CAM.

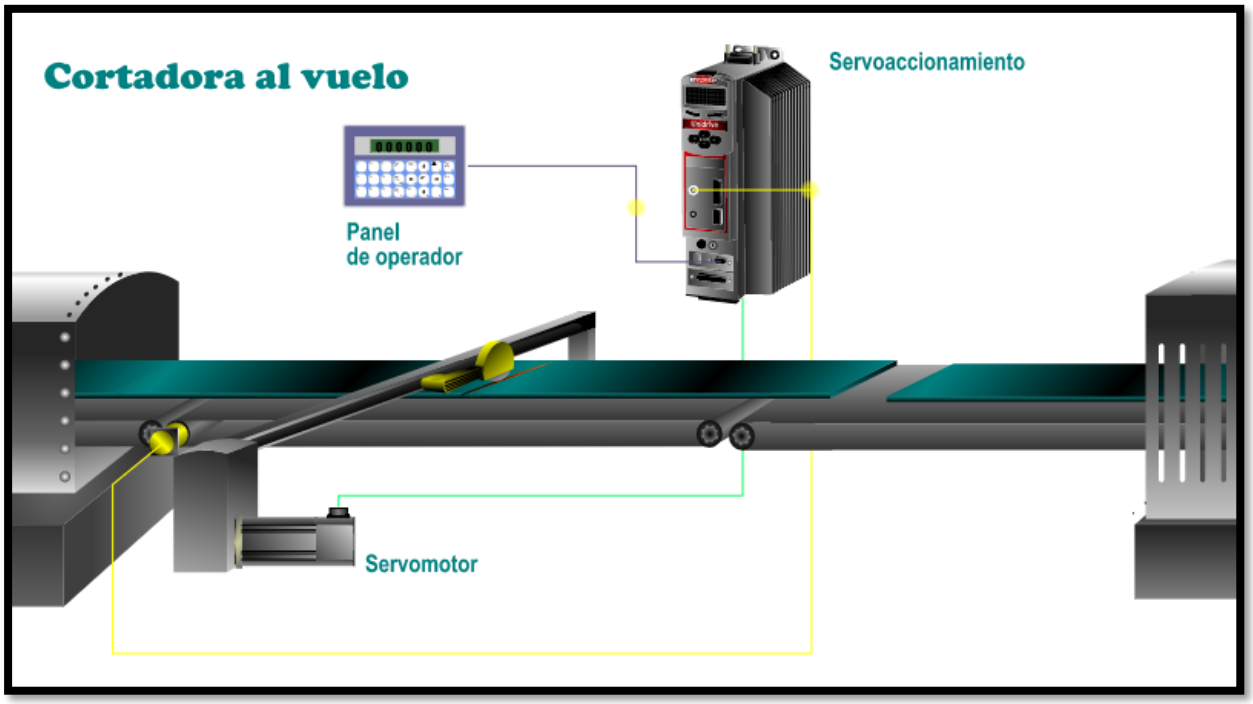


Figura 3-3: Corte al Vuelo. El motor de la cinta actúa de *Maestro* y el del eje de la cuchilla de *Esclavo.* Para conseguir un corte recto, el avance de la cuchilla debe depender en todo momento del avance de la cinta[8].

3.1.2 Ejemplo Comercial

Existe una gran variedad de soluciones en el mercado. Siguiendo la línea del mismo proveedor del servo accionamiento escogido en el capítulo 2, se nombra aquí el controlador M262 (Figura 3-4) capaz de sincronizar entre 4 y 16 ejes a la vez (según el modelo concreto).
Para la programación de los controladores de movimiento de *Schneider* es necesario la utilización del software *EcoStruxure Machine Expert* (*SoMachine*). El software incluye una librería para el control de movimiento (apartado 3.3) con diferentes funciones como la de implementar un perfil CAM (Figura 3-4).

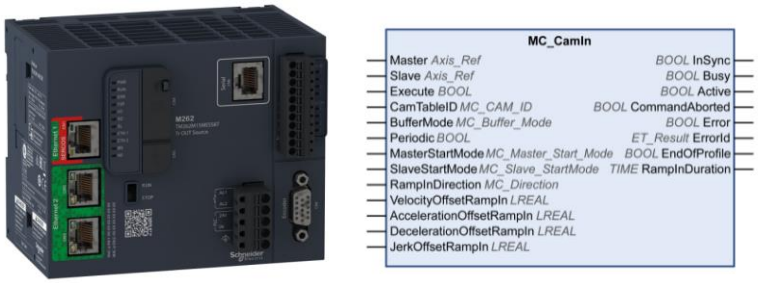


Figura 3-4: M262 Motion Controller (izquierda). Función para generación de perfiles CAM (derecha)[9]

3.2 Control remoto mediante PLC

También es posible el control de servomotores de forma remota haciendo uso de controladores lógicos programables (PLC). Estos controladores son más económicos que los controladores de movimiento. Sin embargo, sus prestaciones en tiempo de respuesta y en versatilidad son menores. Por ejemplo, con un PLC no podremos sincronizar de forma efectiva 2 o más ejes haciendo uso de un perfil CAM. En cualquier caso, si la aplicación a la que se destinan no es muy compleja puede cubrirse perfectamente con un PLC.

[8] https://www.automatizacion-industrial.es/images/2018/05/30/cortadora-al-vuelo.png [última consulta: Julio 2021]
[9] MC_CamIn (schneider-electric.com)

Para la programación de estos controladores de *Schneider* se hace uso de uno de sus softwares específicos: *EcoStruxure Control Expert* (*Unity Pro*) o *EcoStruxure Motion Expert (SoMachine)* dependiendo de la gama de PLC con la que se trabaje.
En cualquier caso, no hay que olvidar que cualquier PLC podría ser utilizado para el control remoto del servo drive. Sin embargo, si se utiliza un autómata del mismo fabricante, la puesta en marcha, manejo y supervisión del servomotor se convierte en una tarea más sencilla ya que, el fabricante pone a nuestra disposición herramientas útiles para ello (software, librería de funciones de alto nivel, etc.)

Figura 3-5: PLC M340 (izquierda). Ejemplo de función (LD) de la Liberia *Motion control* para el ajuste del Modo Home (derecha).

En el capítulo 1, se verá un ejemplo de sincronización básica entre dos servomotores que es posible realizar con un PLC M340 sirviéndose de *Unity Pro*. En el capítulo 1, se realiza una descripción básica del conexionado, configuración y programación necesaria para el control en remoto de un LXM32M haciendo uso de un M238 que, a diferencia del M340, trabaja sobre *SOMachine*.

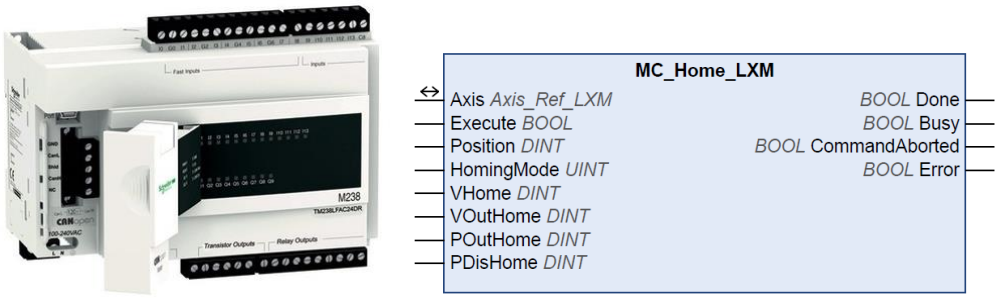


Figura 3-6: PLC M238 (izquierda). Ejemplo de función de la Librería Motion control para el ajuste del Modo Home (derecha)

3.3 Librería Motion Control

Ambos softwares presentados (*Unity Pro* y *SoMachine*) incluyen una librería de *Motion Control* [11] [12] que permite manejar fácilmente los servo accionamientos que se conecten al controlador mediante bus de campo (CANopen, Ethernet/IP o Modbus/TCP). Con las funciones de la librería, se pueden configurar los modos de funcionamiento disponibles (que dependen del modelo de servo drive) sin tener que acudir a la configuración individual de parámetros (aunque esto también es posible). También dispone de funciones de monitorización.
Las funciones que podrán ser utilizadas de esta librería dependerán de los modelos de servo drive y controlador utilizados en cada caso. Por ejemplo, la función MC_CamIn (Figura 3-4) no está disponible para la familia Lexium.
Casi todas las funciones de esta librería comienzan con el prefijo MC_. Estas funciones cumplen la especificación PLCopen[10] y con el estándar IEC 61131-3 de forma que, aunque aquí se han presentado para el fabricante *Schneider*, otros fabricantes (p.ej *Siemens*) contarán con estas funciones para la programación de aplicaciones de control de movimiento.

En cualquier caso, todas las funciones de la librería (*Schneider*) estarán destacadas en rojo en este documento para distinguirlas del resto del texto.

[10] https://www.plcopen.org/technical-activities/motion-control [último acceso: Agosto 2021]

4 SOFTWARE PARA LA PUESTA EN MARCHA

En este capítulo nos centraremos en realizar una descripción general del software de puesta en marcha que *Schneider Electric* pone a libre disposición. Se denomina ***SOMove*** y sirve para facilitar la configuración de los distintos dispositivos de control de motores, entre ellos, los servo drives de la familia Lexium.

4.1 Conexión e inicialización

SOMove permite trabajar estando conectado o no al servo drive. El servo drive se puede conectar a cualquier ordenador que disponga del software a través del puerto Modbus por el lado del servo drive (CN7, Figura 2-4) y a través de un puerto USB por el lado del ordenador utilizando para ello el cable adaptador correspondiente (Figura 4-1).

Figura 4-1: Cable adaptador Modbus to USB. TCSMCNAM3M002P

Una vez conectado, se inicializa *SOMove* y en el menú inicial que aparece se selecciona la opción "Conectar". El programa se encargará de detectar el modelo del servo accionamiento y generar el archivo de trabajo ("proyecto", de extensión .psx).
Por otro lado, también es posible crear un proyecto fuera de línea (sin conexión con el servo drive). Para ello se debe seleccionar la opción con dicho nombre en el menú inicial del programa. A continuación, se siguen los siguientes pasos:

1. Se selecciona el modelo (familia Lexium 32M) y el tipo de comunicación con el PC (Modbus TCP en nuestro caso)

Figura 4-2: Selección de familia. SoMove

2. Se selecciona el modelo concreto del servo drive (Figura 2-3) y del servomotor (Figura 2-2).

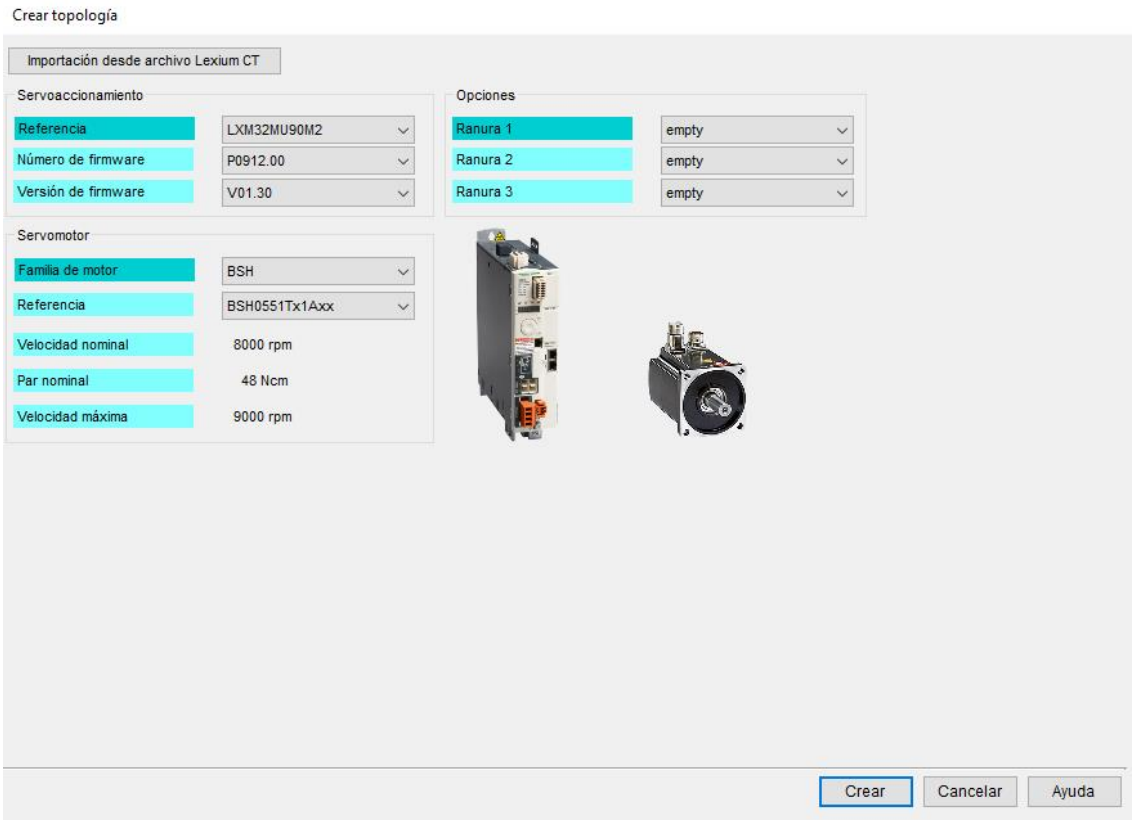


Figura 4-3: Selección del modelo. SoMove

3. Seleccionar "crear" y ya se tendrá un proyecto *offline* listo para poder realizar todos los cambios que quieran hacerse antes de la conexión con el servo.
4. Cuando el archivo esté listo, se guarda con extensión .psx
5. Tras conectar el servo drive al PC se abre el archivo guardado anteriormente y, si no lo realiza automáticamente, en el menú superior seleccionamos "Comunicación" → "Conectar a dispositivo".
6. Aparecerá una ventana emergente en la que seleccionaremos "Almacenar en dispositivo y conectar". La otra opción sobrescribiría la configuración guardada en el archivo con la que había almacenada en el servo drive.

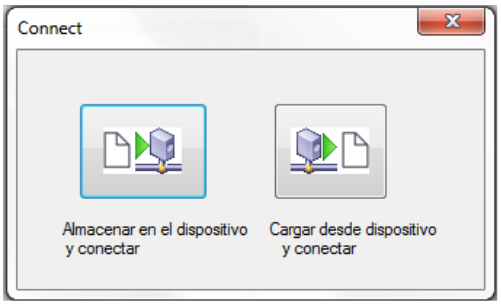


Figura 4-4: Opciones en la conexión del Servo drive a SoMove.

4.2 Funcionalidad

Dentro del archivo de trabajo o proyecto, *SOMove* nos permitirá realizar las siguientes tareas:

1. **Acceso a la lista de parámetros**: ver apartado 4.2.1.

2. **Acceso a la memoria de errores**: a través de ella se puede identificar la causa de cualquier problema o error que se pueda dar durante el funcionamiento o puesta en marcha.

Número	Texto	Estado del equipo
Last Warning	E0000	0:00:00:00 (dd:hh:mm:ss) : 0
Last Error	EA324 : Detectado error durante referenciado	15:06:54:38 (dd:hh:mm:ss) : 1 : EA32D : Detectado error en el final de carrera positivo (señal del interruptor activada brevemente
Error n-1	EA324 : Detectado error durante referenciado	15:06:54:20 (dd:hh:mm:ss) : 1 : EA32D : Detectado error en el final de carrera positivo (señal del interruptor activada brevemente
Error n-2	EA302 : Stop por final de carrera positivo	15:05:32:46 (dd:hh:mm:ss) : 1 : 0x0000 (0)
Error n-3	EA324 : Detectado error durante referenciado	15:03:19:11 (dd:hh:mm:ss) : 1 : EA32D : Detectado error en el final de carrera positivo (señal del interruptor activada brevemente
Error n-4	EA324 : Detectado error durante referenciado	15:06:54:30 (dd:hh:mm:ss) : 1 : EA32D : Detectado error en el final de carrera positivo (señal del interruptor activada brevemente
Error n-5	EA302 : Stop por final de carrera positivo	15:06:44:11 (dd:hh:mm:ss) : 1 : 0xA32D (41773)
Error n-6	EA302 : Stop por final de carrera positivo	15:03:19:12 (dd:hh:mm:ss) : 1 : 0x0000 (0)
Error n-7	EA324 : Detectado error durante referenciado	15:03:17:55 (dd:hh:mm:ss) : 1 : EA32D : Detectado error en el final de carrera positivo (señal del interruptor activada brevemente
Error n-8	EA324 : Detectado error durante referenciado	15:03:08:26 (dd:hh:mm:ss) : 1 : E0000

Figura 4-5: Ejemplo de historial de errores capturados en SOMove

3. **Ajuste del regulador (Tunning)**: es posible ajustar los distintos parámetros del regulador del servo drive de forma automática, semiautomática o manual (apartado 2.11.2). Es necesario llevarlo a cabo al menos la primera vez que se utiliza el servo drive, si se hace un reseteo de fábrica o si se modifica la mecánica de la instalación (p. ej cambiar un acoplamiento rígido (engranajes) por un acoplamiento elástico (correa)).

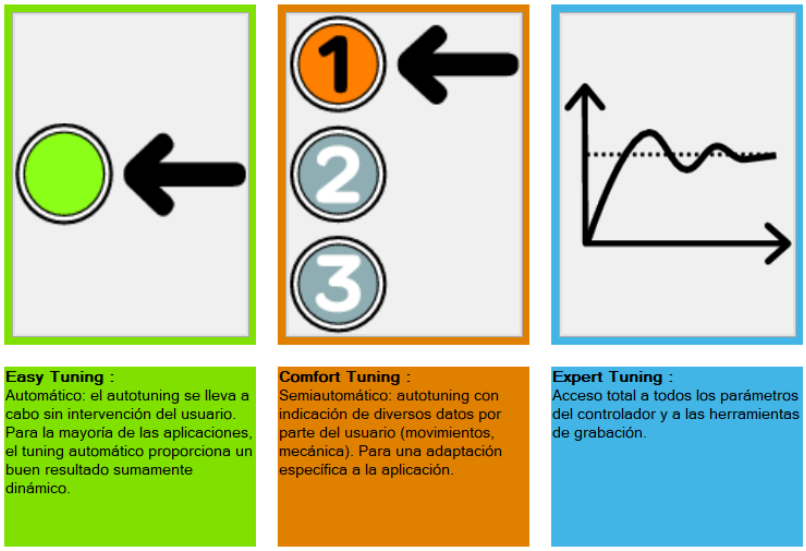


Figura 4-6: Opciones de ajuste del regulador en la interfaz de SoMove

4. **Acceso al registro de datos (Motion Sequence)**: es posible configurar los registros de datos que vayan a utilizarse en el modo Motion Sequence (Figura 2-22).

5. **Configuración de las funciones de entrada/salida de las señales digitales** (desde la lista de parámetros)**, consulta de su estado y opción de forzado en tiempo real** (desde la pestaña "Visualización")**.**

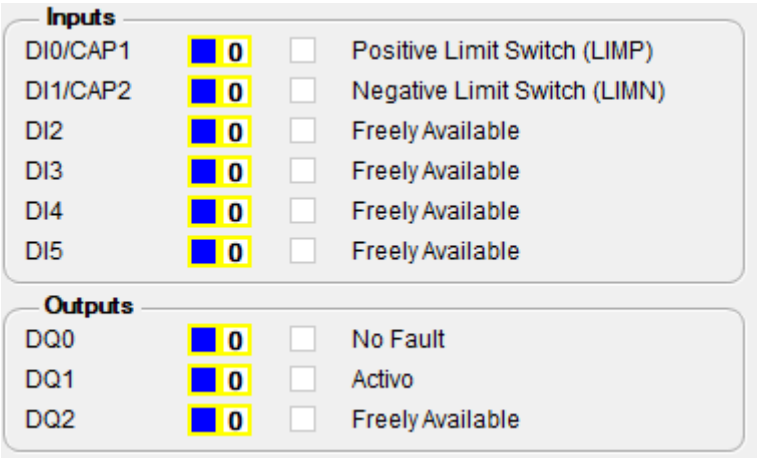


Figura 4-7: Interfaz para monitorización/forzado de entradas/salidas digitales. SoMove.

6. **Operación del equipo en modo local**: lo que incluye la posibilidad de cambiar entre los distintos modos de funcionamiento (salvo el "Interpolated Position") y ejecutarlos con la configuración deseada.

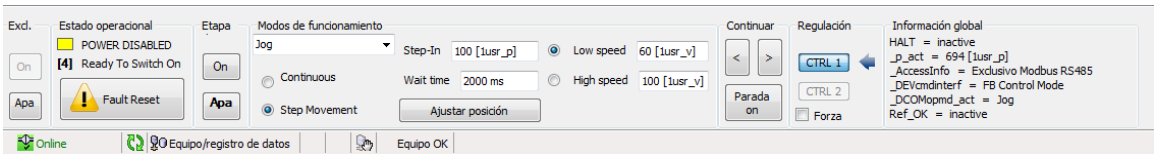


Figura 4-8: Panel de control para el manejo en modo local desde SoMove.

7. **Monitorización del servomotor:** es posible monitorizar el funcionamiento/estado del accionamiento (posición, velocidad, par, temperatura, carga, etc.)

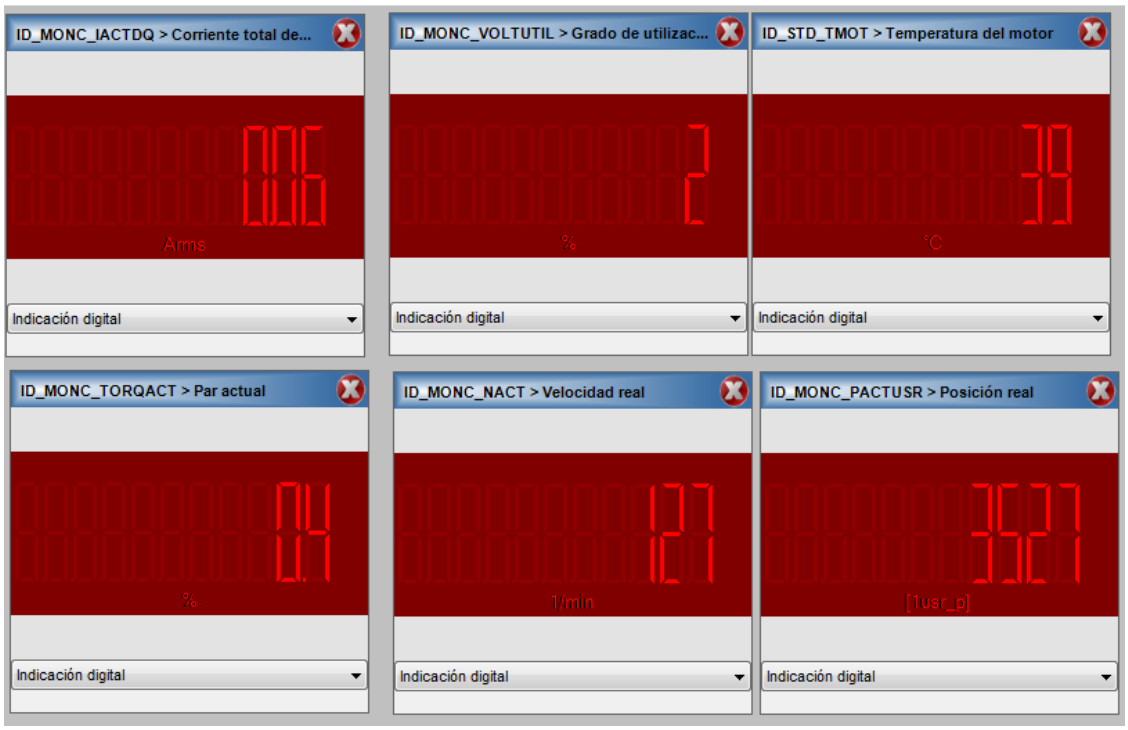


Figura 4-9: Ejemplos de displays disponibles para la monitorización. SoMove.

La monitorización de las distintas variables puede realizarse con los widgets de la pestaña "Visualización" (Figura 4-9) o bien en forma de gráficas temporales mediante la herramienta de grabación. En esta última es importante configurar adecuadamente la tasa de muestreo de la señal o Sampling *rate* (paso 3, Figura 4-10). Mientras mayor sea, mayor podrá ser la ventana de visualización (Sampling duration).

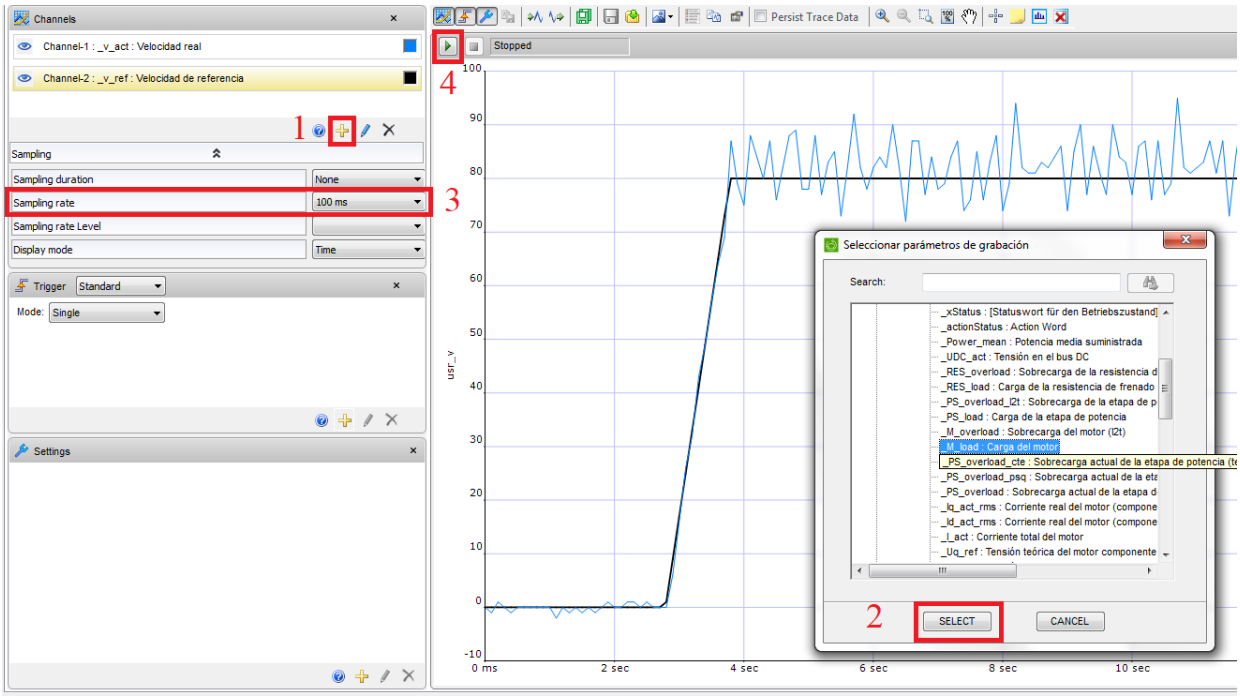
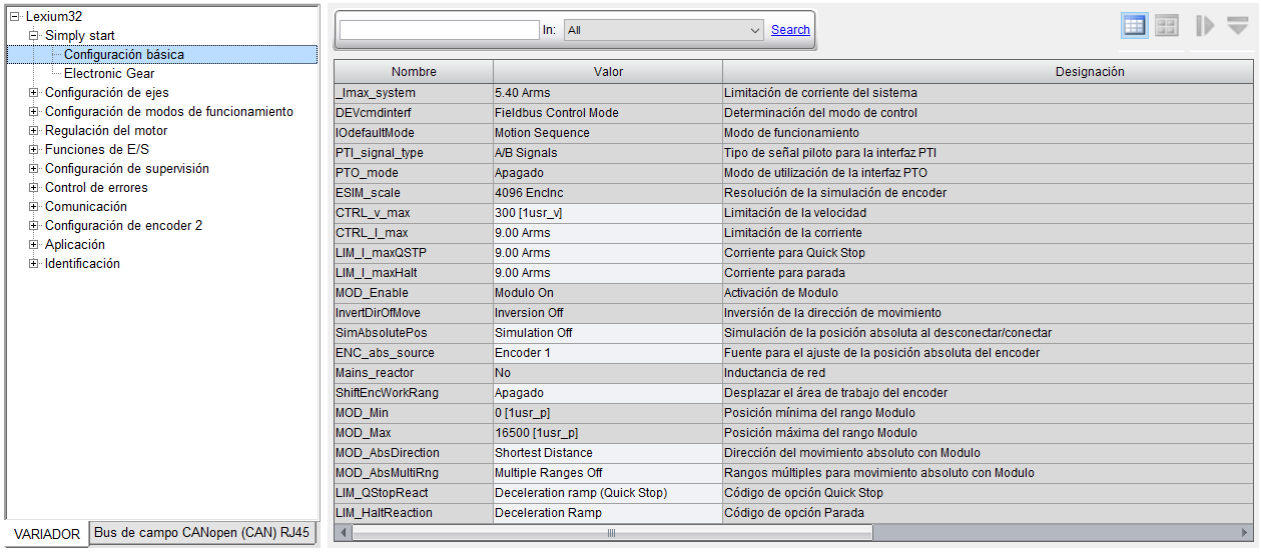


Figura 4-10: Herramienta de grabación de SoMove. Se indican los pasos a seguir para la grabación de un parámetro.

4.2.1 Acceso a la lista de parámetros.

A través de este listado se puede configurar el comportamiento del servo accionamiento de forma más rápida y sencilla que a través de la HMI incorporada en el servo drive, o que a través del bus de campo desde un controlador remoto.

Los parámetros que pueden ser modificados en cada momento, dependerá del estado de funcionamiento en el que se encuentre el servo drive. Podremos escribir cualquier parámetro mientras el servo drive se encuentre en el estado "Ready to Switch On" (apartado 2.2). Sin embargo, mientras está en funcionamiento (estado "run") muchos parámetros dejan de estar disponibles para su modificación. El software te lo indica sombreando su valor en gris.



Nombre	Valor	Designación
_Imax_system	5.40 Arms	Limitación de corriente del sistema
DEVcmdinterf	Fieldbus Control Mode	Determinación del modo de control
IOdefaultMode	Motion Sequence	Modo de funcionamiento
PTI_signal_type	A/B Signals	Tipo de señal piloto para la interfaz PTI
PTO_mode	Apagado	Modo de utilización de la interfaz PTO
ESIM_scale	4096 EncInc	Resolución de la simulación de encoder
CTRL_v_max	300 [1usr_v]	Limitación de la velocidad
CTRL_I_max	9.00 Arms	Limitación de la corriente
LIM_I_maxQSTP	9.00 Arms	Corriente para Quick Stop
LIM_I_maxHalt	9.00 Arms	Corriente para parada
MOD_Enable	Modulo On	Activación de Modulo
InvertDirOfMove	Inversion Off	Inversión de la dirección de movimiento
SimAbsolutePos	Simulation Off	Simulación de la posición absoluta al desconectar/conectar
ENC_abs_source	Encoder 1	Fuente para el ajuste de la posición absoluta del encoder
Mains_reactor	No	Inductancia de red
ShiftEncWorkRang	Apagado	Desplazar el área de trabajo del encoder
MOD_Min	0 [1usr_p]	Posición mínima del rango Modulo
MOD_Max	16500 [1usr_p]	Posición máxima del rango Modulo
MOD_AbsDirection	Shortest Distance	Dirección del movimiento absoluto con Modulo
MOD_AbsMultiRng	Multiple Ranges Off	Rangos múltiples para movimiento absoluto con Modulo
LIM_QStopReact	Deceleration ramp (Quick Stop)	Código de opción Quick Stop
LIM_HaltReaction	Deceleration Ramp	Código de opción Parada

Figura 4-11: Listad de parámetros en SoMove.

Tras realizar modificaciones sobre la lista debe guardarse en la memoria del servo drive (paso 1, Figura 4-12). Además, la modificación de algunos parámetros precisa de un posterior reinicio del servo drive (si tras la modificación del parámetro, se pasa al estado “Not Ready to Switch On”). Este reinicio puede realizarse desde el propio programa (pasos 2 y 3).

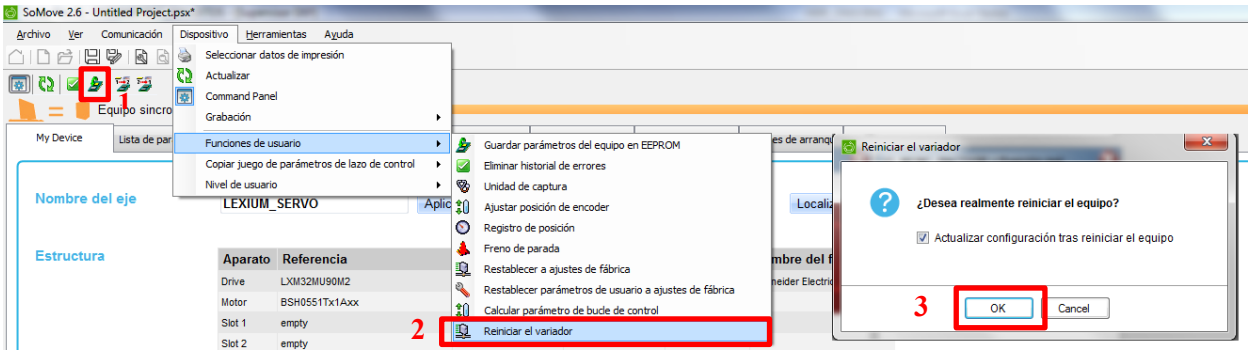


Figura 4-12: Guardar parámetros y reinicio del variador. SoMove.

5 CONTROL REMOTO CON M238

Como ya se mencionó anteriormente, este proyecto se centra en el control remoto del servo accionamiento mediante un autómata programable. De las dos grandes alternativas presentadas en el apartado 3.2, en este capítulo se describe el conexionado, configuración y programación necesaria para llevar a cabo el control remoto desde un M238. Por lo tanto, será necesario la utilización del software *SOMachine* [13].
Esta puesta en marcha básica del sistema PLC - Servo accionamiento se realiza sin tener en cuenta ninguna aplicación final concreta para el servomotor. Además, la intercomunicación de este sistema se lleva a cabo mediante un bus de campo con protocolo CANOpen (Anexo A).

Figura 5-1: Controlador Lógico M238.

5.1 Conexionado

El primer paso es la creación física de la red mediante el interconexionado de los equipos. El servo drive (LXM32M) debe contar con un módulo CANOpen instalado [8]. La conexión CANOpen se realiza atendiendo a [14]. La conexión a un PC que disponga del software *SOMachine* se realiza atendiendo a [15]. El esquema final de conexión es el mostrado en la Figura 5-2.

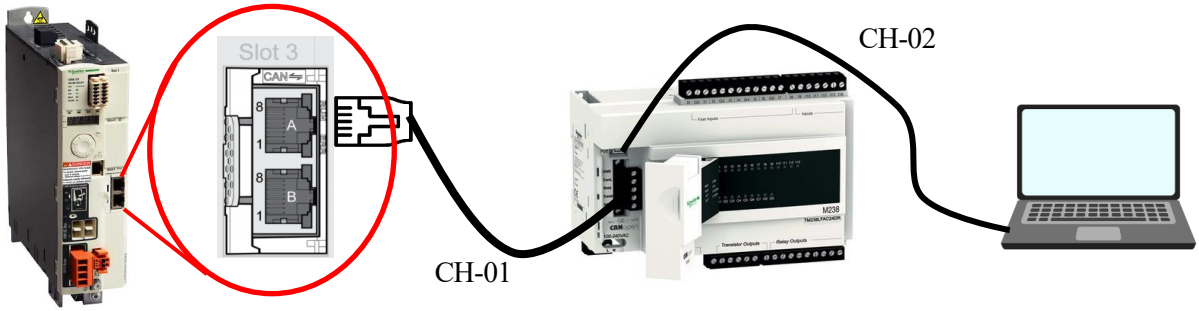


Figura 5-2: Esquema de Conexionado Servoaccionamiento – M238

CH-01: Cable comunicación CANOpen entre Servoaccionamiento y PLC

Pin LXM32M	Señal	Pin M238
1	CAN_H	4
2	CAN_L	2
3	CAN_0V / CAN_GD	1

Tabla 5-1: Cable comunicación CANOpen M238 – Servo

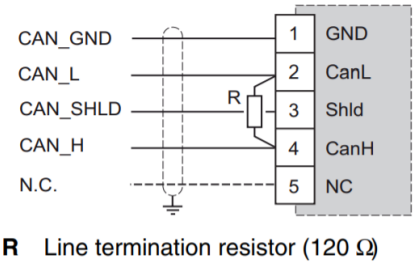


Figura 5-3: Pines CANOpen del M238

CH-02: Cable comunicación del PLC con PC para programación y puesta en marcha. Se trata de un cable adaptador Modbus RJ45 to USB (Figura 4-1)

5.2 Configuración en SoMachine

Una vez creada la red física, el siguiente paso sería la generación de un programa que transferir al autómata. En este apartado se describen los pasos necesarios para la generación de dicho programa a través de *SoMachine*[11].

1. En primer lugar, es aconsejable conectar el servo drive al software de puesta en marcha para realizar una configuración inicial de los parámetros del dispositivo. Dicha configuración inicial dependerá de la aplicación final a la que se destine el sistema. En este caso, basta con mencionar que se debe asignar un nodo al servo drive a través de su parámetro `CAN_adress`. En cuanto a la velocidad de transmisión (`CAN_Baud`), en ambos dispositivos puede mantenerse como viene por defecto (250kbaudio). Hay que recordar que, si es la primera vez que se utiliza el servo drive, también será necesario configurar los parámetros del regulador.

2. En segundo lugar, se debe abrir SoMachine y pinchar sobre el icono de "Nuevo Proyecto". Aparecerá una pantalla del flujo de trabajo necesario como la que se muestra en la Figura 5-4. Haciendo doble clic sobre "Configuración", se abrirá una ventana emergente donde deberemos seleccionar nuestro autómata (Figura 5-5). El modelo concreto de M238 puede encontrarse en el frontal del dispositivo (Figura 5-1).

3. A continuación, en la pantalla de flujo de trabajo, hacer doble clic sobre "Controlador". Se iniciará el programa para comenzar con la programación del PLC. Para seguir correctamente los siguientes pasos, nos aseguraremos de que la vista de la interfaz está configurada en modo "CODESYS Classic" (Figura 5-6).

4. En el menú izquierdo ("Dispositivos") se puede añadir una nueva sección de programación (*POU*) haciendo clic derecho sobre "Application" → "Agregar objeto" → POU.

[11] Los pasos a seguir pueden variar ligeramente en función de la versión del software que se utilice. Para este capítulo se hace uso de la versión 4.3.0.0

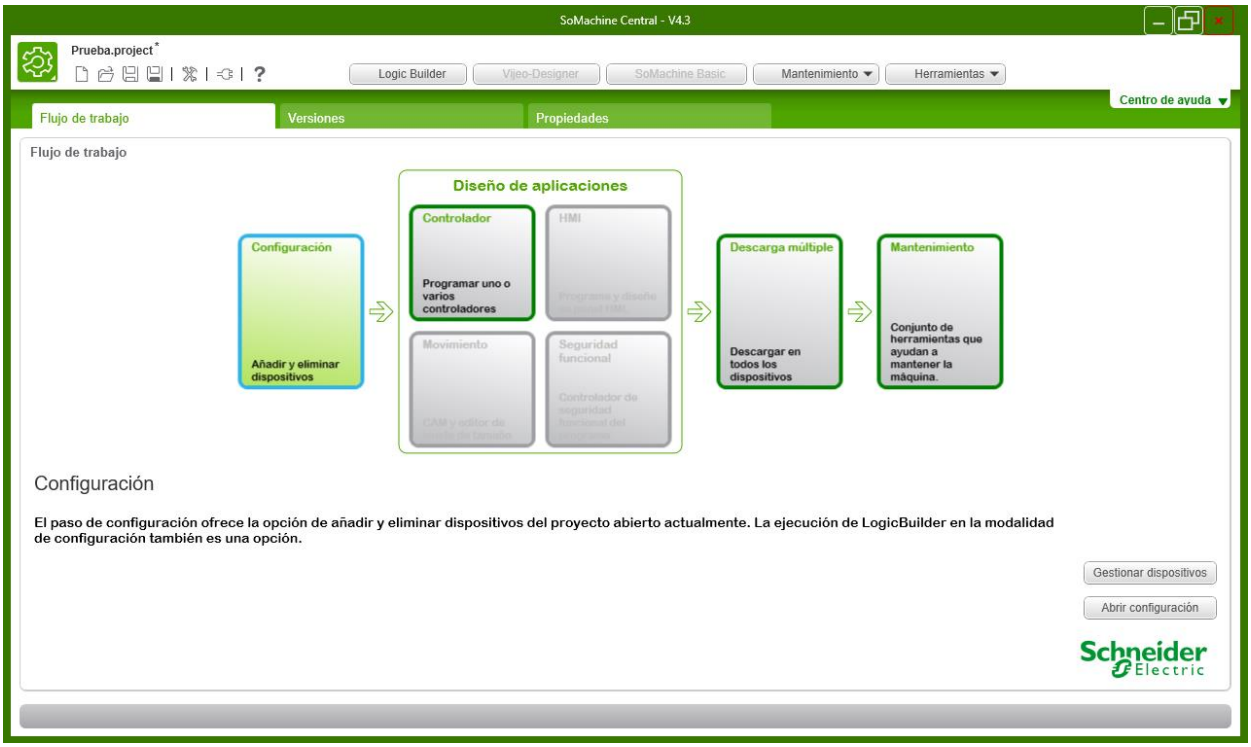


Figura 5-4: Configuración SoMachine. Paso 2.1

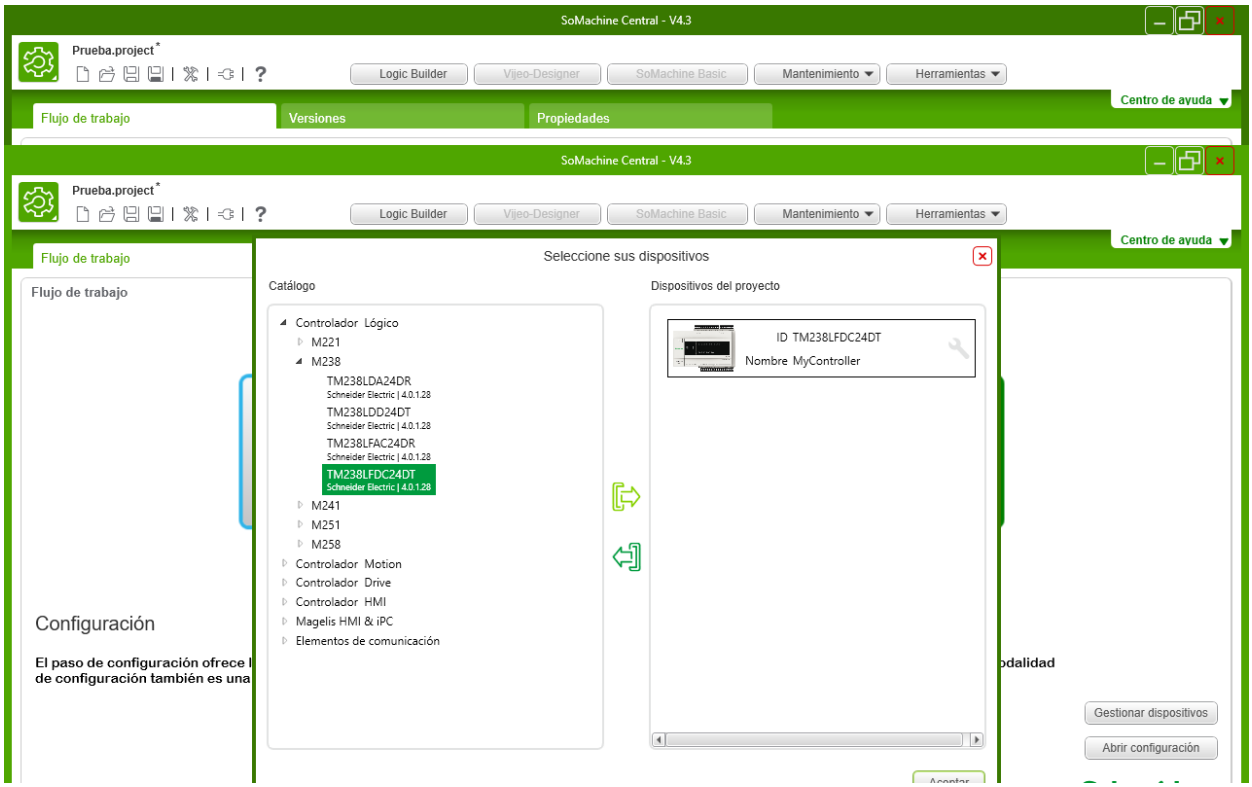


Figura 5-5: Configuración SoMachine. Paso 2.2

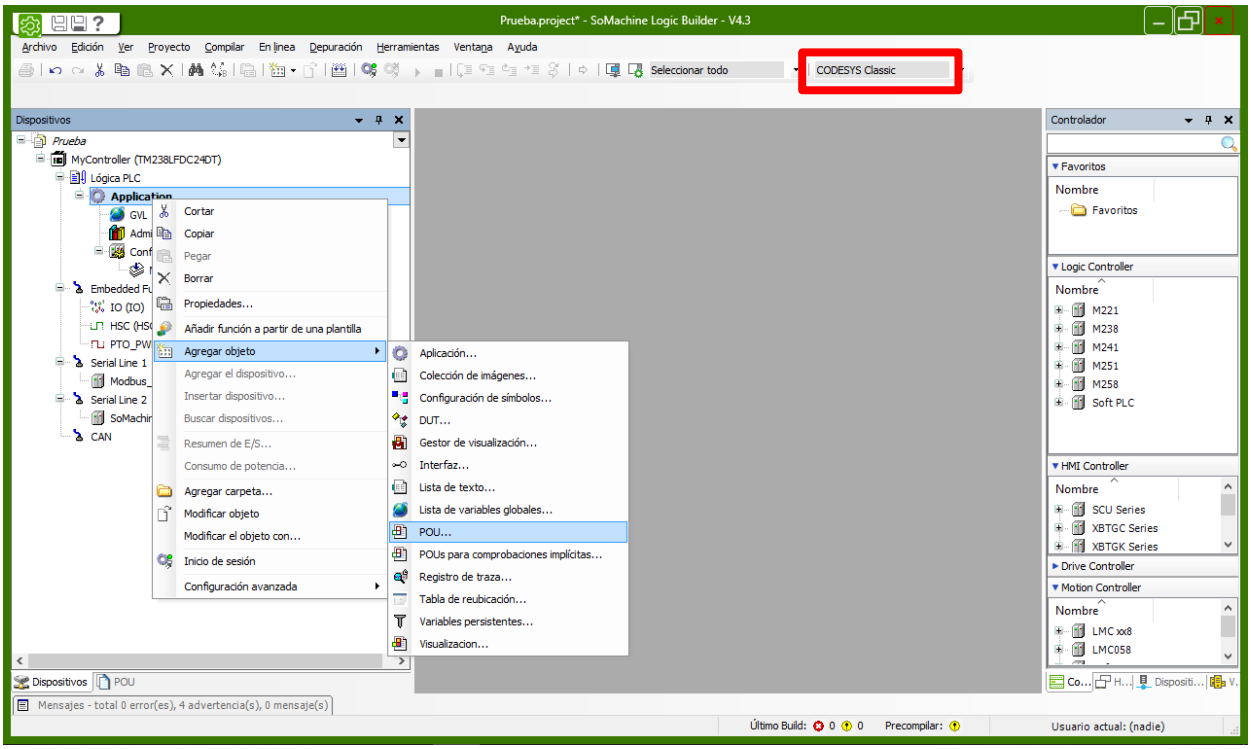


Figura 5-6: Configuración SoMachine. Paso 4

5. En la ventana emergente que se despliega (Figura 5-7), se selecciona la opción "Programa" y el lenguaje de implementación deseado. Para trabajar con la librería de funciones *Motion Control* se aconseja la utilización de FBD o LD como lenguaje de implementación. Al hacer clic sobre "Agregar" nos aparecerá la *POU* en el menú izquierdo dentro de "Application". Para que el PLC ejecute su código, debemos arrástralo dentro de "MAST" (Figura 5-8).

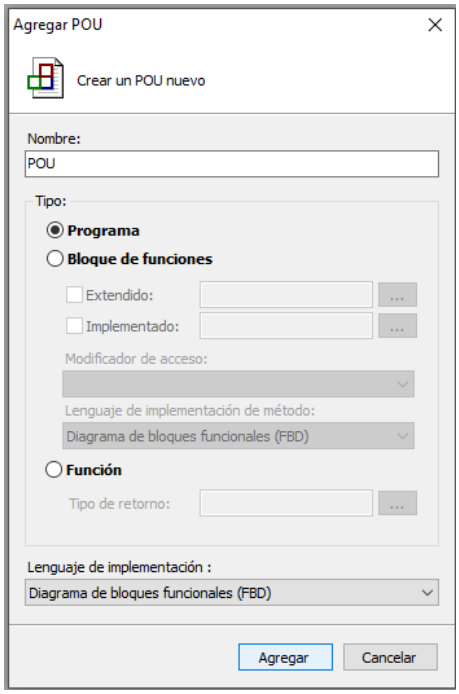


Figura 5-7: Configuración SoMachine. Paso 5.1

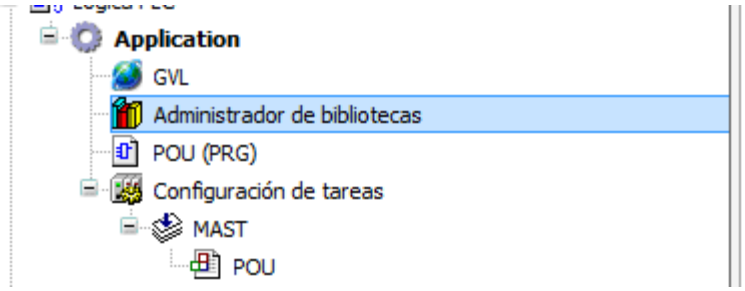


Figura 5-8: Configuración SoMachine. Paso 5.2

6. El siguiente paso es configurar la red CANOpen. Dentro del menú "Dispositivos", hacer clic izquierdo sobre "CAN" y seleccionar "Agregar dispositivo". En la ventana emergente desplegada, se selecciona "CANopen Optimized" y se hace clic en "Agregar el dispositivo". Una vez que este sea visible en el menú izquierdo, se puede cerrar la ventana emergente.

7. Para agregar el servo drive a la red CANOpen, hacer clic derecho sobre "CANOpen Optimized" y seleccionar "Agregar el dispositivo". Se busca nuestro servo drive (Lexium 32M), se le da un nombre y se hace clic en "Agregar el dispositivo". Una vez que este sea visible en el menú izquierdo, se puede cerrar la ventana emergente (Figura 5-9).

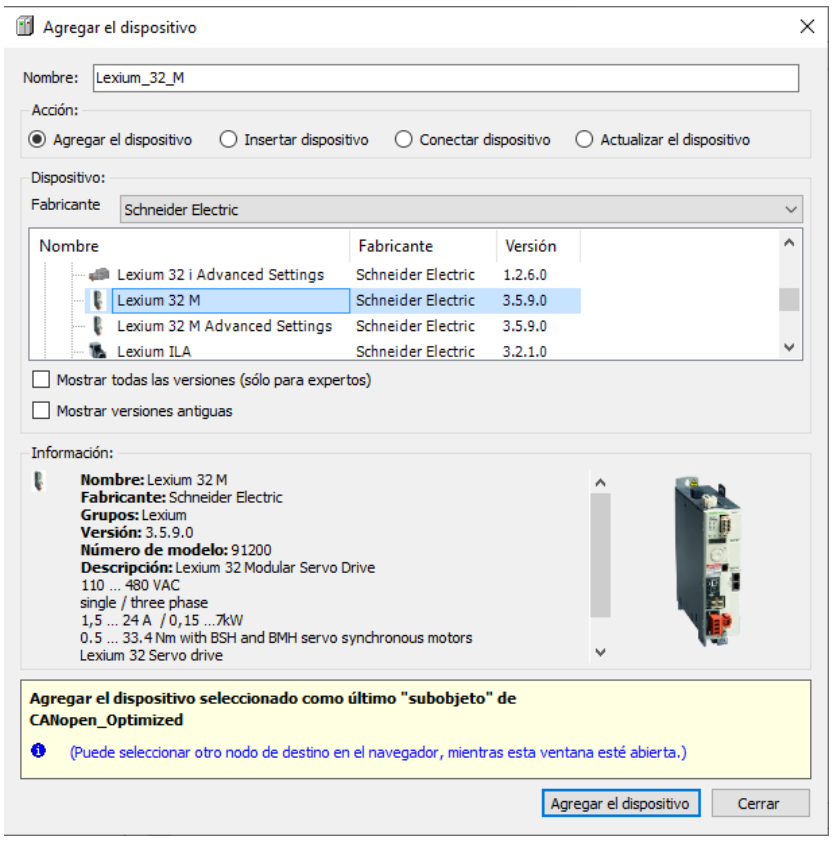


Figura 5-9: Configuración SoMachine. Paso 7

8. Para finalizar con la configuración de la red CAN, hacer doble clic sobre el servo drive añadido. Se nos abrirán una serie de pestañas en el área de trabajo que permiten configurar la comunicación. En este caso, basta con configurar el nodo asignado al servo drive (el establecido previamente en el parámetro `CAN_adress`).

9. Llegados a este punto, lo único que resta es realizar la programación con las herramientas del lenguaje escogido y transferir el programa resultante al PLC. Este paso se describe en el siguiente apartado.

5.3 Programación y transferencia de la aplicación

El entorno de programación es el mostrado en la Figura 5-10. Se pueden encontrar las herramientas de programación básicas en la parte superior de la pantalla. La primera gran diferencia con el entorno de programación de *Unity Pro* es que podremos encontrar accesos directos a operadores lógicos y matemáticos usuales (OR, AND, Suma, Mayor que...) así como a contadores y temporizadores.

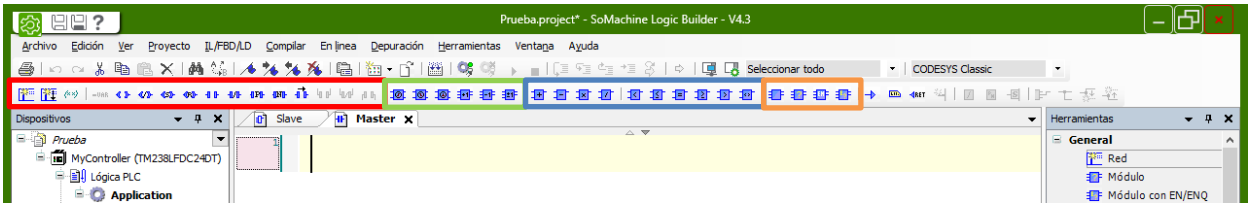


Figura 5-10: Herramientas de programación en SoMachine. En rojo Contactos y bobinas (Solo LD), en verde temporizadores y contadores, en azul operadores matemáticos y en naranja inserción de bloques funcionales.

Se puede consultar información detallada sobre todas las funciones de la librería *Motion Control* disponibles para nuestro servo drive en [12]. A continuación, se detallan los pasos para insertar dichas funciones.

1. Se hace doble clic sobre "Insertar módulo vacío" en la barra de herramientas.

2. Con ello, aparece un módulo vacío. Las interrogaciones indican cosas que faltan por definir. Al hacer clic sobre las internas (en negrita), se puede escribir la función de la librería que se quiera añadir. Por ejemplo: MC_POWER_LXM.

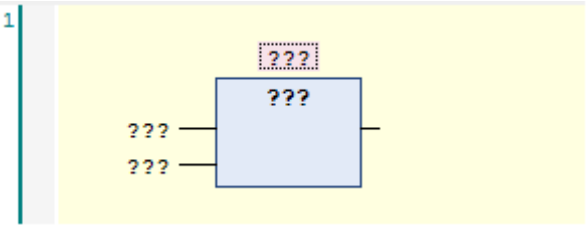


Figura 5-11: Módulo vacío en SoMachine.

3. Las interrogaciones superiores se sustituyen automáticamente por un nombre predefinido para la variable que representa el bloque funcional añadido. En este caso lo modificaremos por: "MC_POWER_Master". Al presionar la tecla Intro se despliega una ventana emergente que permite declarar dicha variable. Basta con hacer clic en "Aceptar".

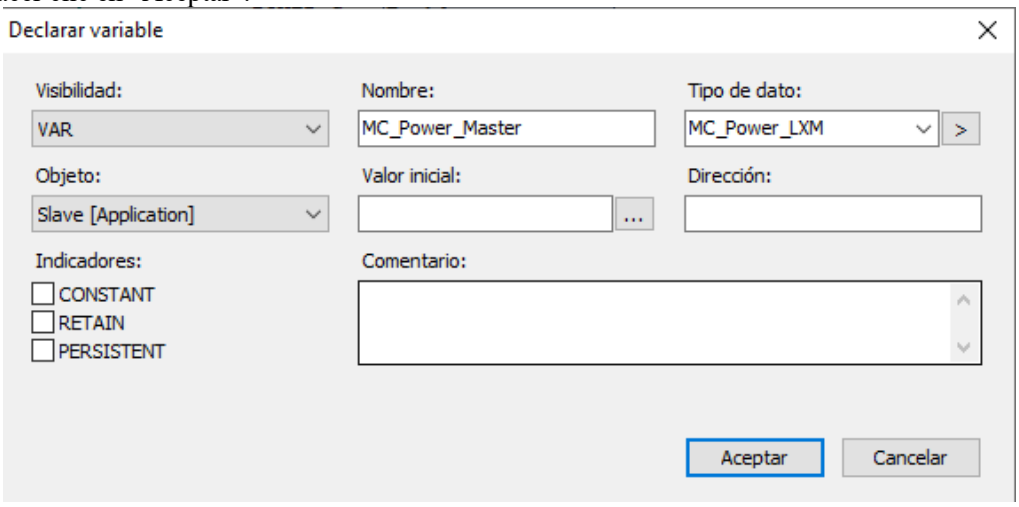


Figura 5-12: Ventana de declaración de variables. SoMachine

4. Por último, hay que definir las entradas y salidas de la función. Al seleccionar las interrogaciones que hay sobre ellas (Figura 5-13), podremos cambiarlas por el nombre de una variable predefinida o una variable no existente (en este caso se desplegará la ventana emergente para su declaración).

La entrada “Axis” indica el eje o dispositivo al que hace referencia la función. En dicha entrada se debe colocar el nombre otorgado al dispositivo en el paso 7.
Seleccionando el bloque de la función en sí (se destacará en rojo) y presionando la tecla F1 se desplegará información detallada de la función como la contenida en [12].

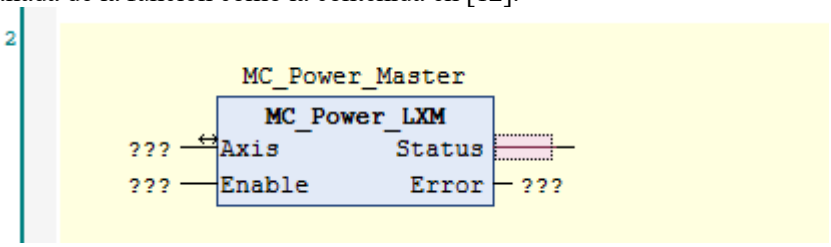


Figura 5-13: Asignación de variables en bloque MC_POWER

Una vez terminada la programación de la aplicación, presionaremos F11 para compilar y verificar si existe algún error. En caso negativo, el último paso es establecer la comunicación con el PLC y transferir el programa. Para ello hay que seguir los siguientes pasos:

1. Hacer doble clic sobre “MyController” en el menú “Dispositivos”. Se despliega una pestaña en el área de trabajo sobre la que habrá que hacer clic en su pestaña “Configuración de comunicación”

2. Si no existe un “Gateway-xxx” creado, hacer clic sobre “Agregar puerta de enlace”, renombrar si se considera necesario y clic en “Aceptar”

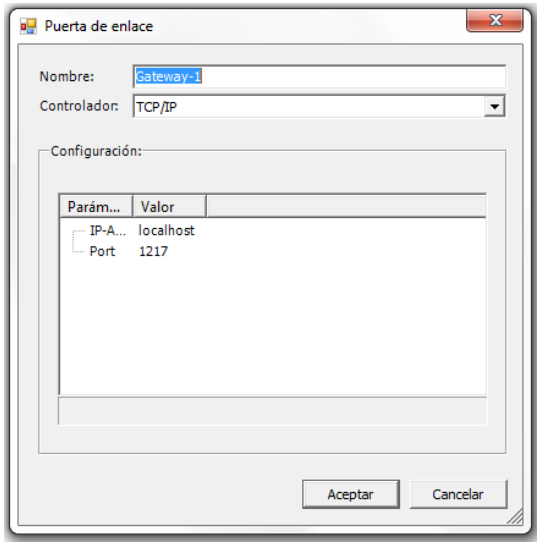


Figura 5-14: Añadir puerta de enlace para conexión con PLC en SoMachine.

3. Manteniendo seleccionada la puerta de enlace, hacer clic en “Examinar red”.

4. Si el cable CH-02 (Figura 5-2) está correctamente conectado, aparecerá el autómata (M238) bajo la puerta de enlace. Seleccionarlo y hacer clic en “Establecer una ruta activa”

5. Aparecerá una ventana de advertencia de seguridad que se debe confirmar presionando las teclas ALT+F. Tras ello, aparecerá indicada la ruta activa sobre el identificador del PLC detectado en el paso 3 (Figura 5-15).

Figura 5-15: Ruta activa de comunicación establecida con PLC

6. En el menú superior, seleccionar “En línea” → “Iniciar la sesión” para conectar el PC con el controlador.

7. Luego, sobre el mismo menú “En línea”, seleccionar “Descarga múltiple…” para transferir el programa. Se desplegará una ventana emergente. Escoger la tercera de las “Opciones de modificación en línea” y hacer clic en “Aceptar”

5.4 Depuración y monitorización de la aplicación

Una vez transferida la aplicación al PLC, basta con hacer clic sobre icono de Play en la barra de herramientas para iniciar la ejecución de esta.
Para poder monitorear/depurar la aplicación se pueden utilizar las listas de supervisión. Estas suelen aparecer por defecto en la parte inferior de la interfaz tras conectar el PLC con el PC. Si no puede visualizarlas, en el menú superior, hacer clic sobre “Ver” → “Supervisión” → “Supervisión x” (pueden usarse hasta 4 listas distintas) (Figura 5-16).

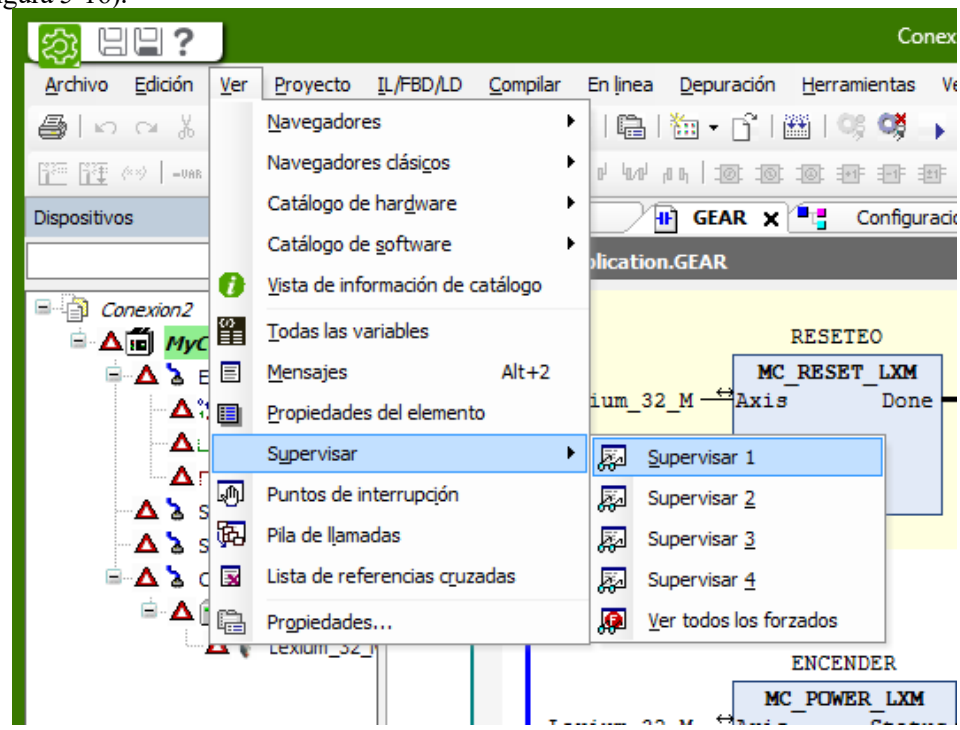


Figura 5-16: Listas de Supervisión. SoMachine

Para agregar cualquier variable a la lista de supervisión basta con hacer clic derecho sobre la misma allí donde aparezca (dentro de las POUs programadas) y seleccionar “Agregar a la lista de supervisión” (Figura 5-17). También se puede introducir haciendo clic sobre la primera fila vacía de la lista e introduciendo el nombre de la variable.

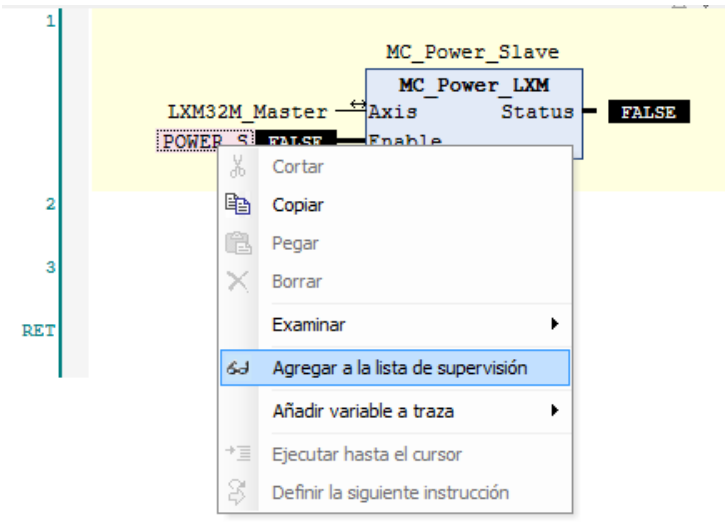


Figura 5-17: Agregar variable a lista de supervisión. SoMachine

Además de monitorearlas, podrá modificar/forzar el valor de las variables de entrada. Para ello hacer clic sobre "Valor preparado", seleccionar el valor deseado y presionar CTRL+W para modificar o CTRL+SHIFT+W para forzar.

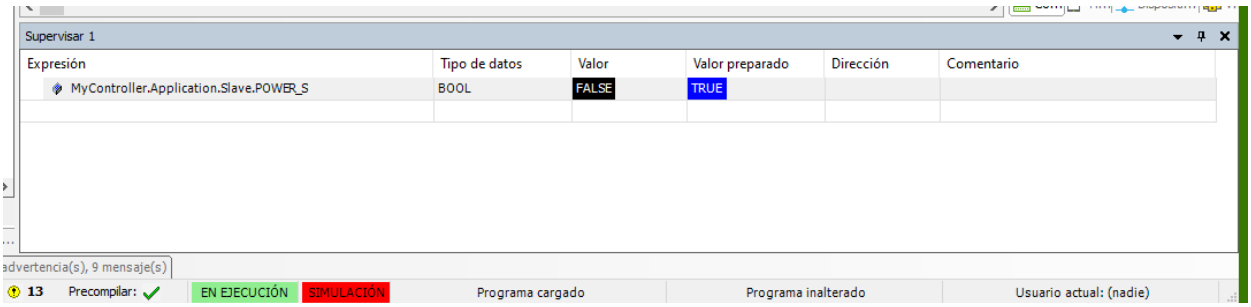


Figura 5-18: Lista de supervisión. SoMachine

6 SINCRONIZACIÓN DE DOS SERVOMOTORES

En el presente capítulo se describen todos los pasos necesarios para la implementación completa de un sistema conformado por dos servomotores que deben operar sincronizados entre sí. Por lo tanto, el objetivo principal de la aplicación es que funcionen como un conjunto *Maestro/Esclavo*. Sin embargo, se incluirá la posibilidad del manejo manual de ambos servo accionamientos con el objetivo de permitir la exploración de los modos de funcionamiento expuestos en el apartado 2.4. La funcionalidad completa propuesta para este sistema se describe en el apartado 6.4.

El control de ambos servomotores se realizará de forma remota desde un PLC (M340) haciendo uso de un bus de campo con protocolo CANOpen. En el Anexo A se describen las bases generales de dicho protocolo.

6.1 Maqueta de pruebas. Conexionado

6.1.1 Montaje mecánico

El sistema se ha integrado en una maqueta para servir con fines educativos en la Escuela Técnica Superior de Ingeniería de Sevilla (Figura 6-2).

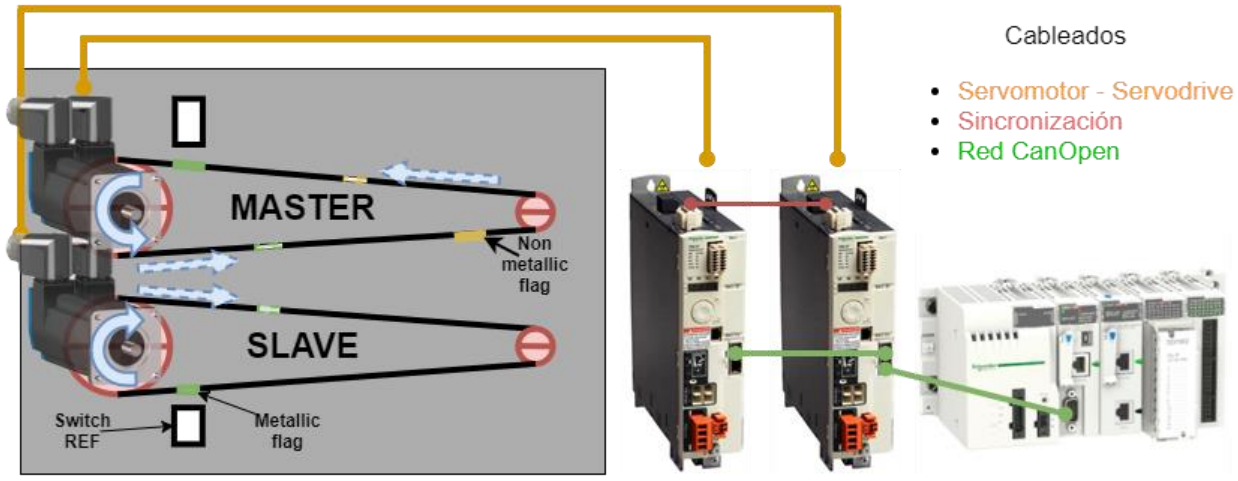


Figura 6-1: Croquis maqueta de pruebas

La maqueta está conformada principalmente por un panel metálico sobre el que se instalan dos subsistemas polea-correa. Cada subsistema está accionado por uno de los servomotores. Para el caso en el que trabajen en sincronismo, el sistema superior será el *Maestro* y el inferior el *Esclavo*.
Adicionalmente, en cada correa se han colocado una serie de trazas indicadoras que permiten visualizar la sincronización entre ambos servomotores. Cada correa lleva una de aluminio para que sea detectable por los interruptores de referencia (sensores inductivos instalados para el *Homing*).

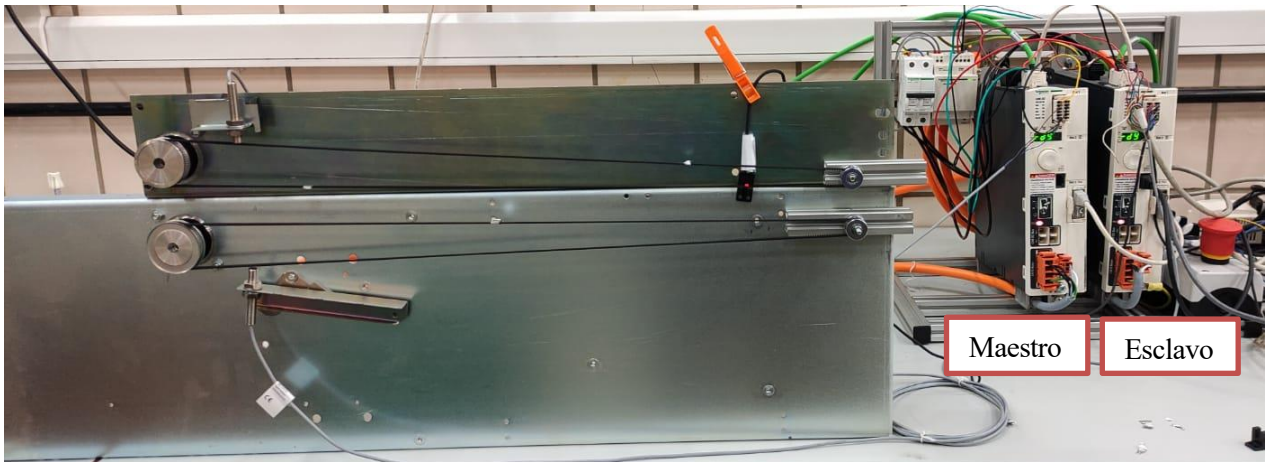


Figura 6-2: Maqueta de pruebas provisional.

6.1.2 Cableado servo drive

El conexionado de cada servo drive con el servomotor, la seta de emergencia, la alimentación y las entradas/salidas digitales se realiza prácticamente tal como se muestra en la Figura 2-4.
La única salvedad es que, al tratarse de dos servo drives, uno de ellos (en este caso el *Maestro*) se conecta indirectamente, a través del puerto CN2 del otro servo drive, a la alimentación de 24 VDC (terminales 7-8) y a la seta de emergencia (terminales 5-6).

6.1.3 Cableado red CANOpen

La comunicación de los servo accionamientos con el PLC (*Maestro* del sistema) se realizará a través de una red CANOpen como la mostrada en la Figura 6-3. Para ello, es imprescindible la instalación del módulo de comunicaciones CANOpen disponible para nuestro modelo de servo drive (VW3A3608) [8].

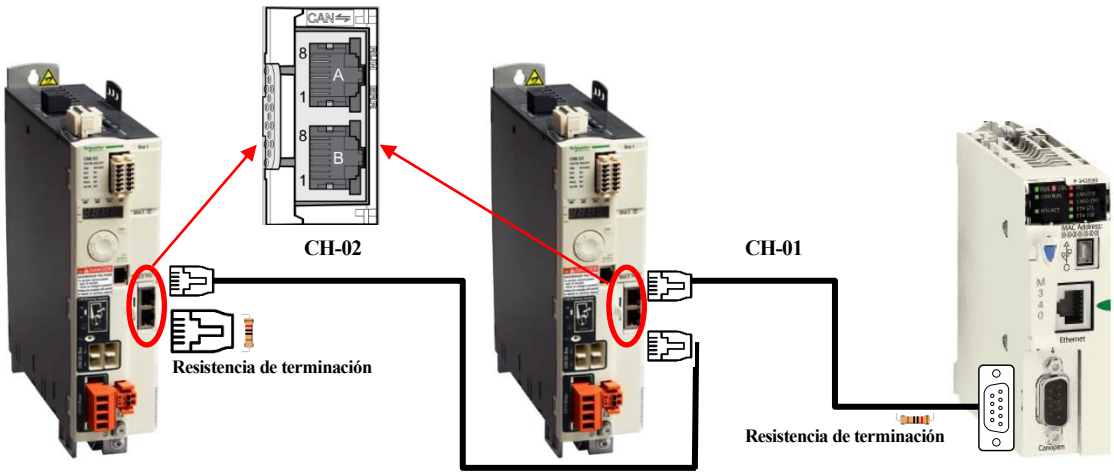


Figura 6-3: Cableado red CANOpen

El PLC se conecta al servo *Esclavo* mediante el cable CH-01. El servo *Maestro* se interconecta con el primero a través del cable CH-02. Las redes CANOpen constan de una línea de transmisión de 3 hilos (CAN_H (High), CAN_L (Low), CAN_GND (Ground)) que debe estar terminada en ambos extremos físicos con resistencias (LT) de 120 Ω tal como se muestra en la Figura 6-4.

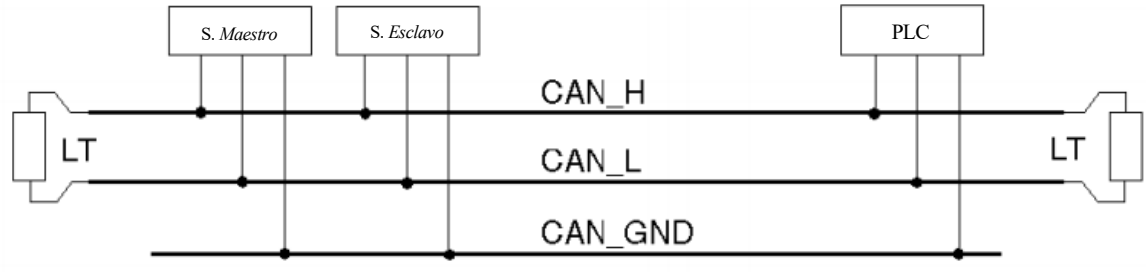


Figura 6-4: Línea de transmisión CANOpen

- **CH-01:** Por un lado, la conexión con el PLC se realiza a través del puerto SUB-D9. Por el otro lado, la conexión con el servo se realiza a través de uno de los puertos RJ45 del módulo CANOpen. En este cable se incluye una de las dos resistencias de terminación de la red. La asignación de pines se muestra en la Figura 6-5.

Pin LXM32M	Señal	Pin M340
1	CAN_H	7
2	CAN_L	2
3	CAN_0V / CAN_GD	3

Tabla 6-1: Asignación de pines conexión M340 – Servo (Cable CH-01)

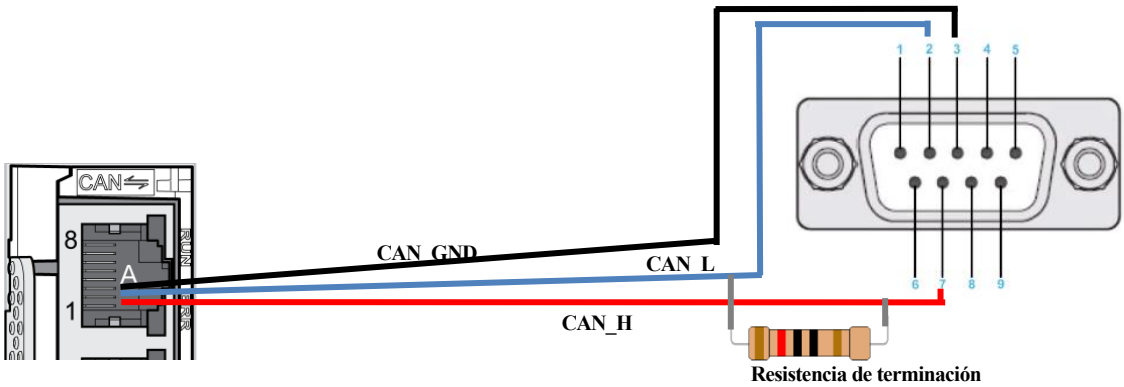


Figura 6-5: Cable CH-01. Conexión M340 – Servo.

- **CH-02:** Consiste en un cable de conexión directa (pin a pin) RJ45 – RJ45. La asignación de pines es la siguiente:
    - Pin 1: CAN_H
    - Pin 2: CAN_L
    - Pin 3: CAN_GND

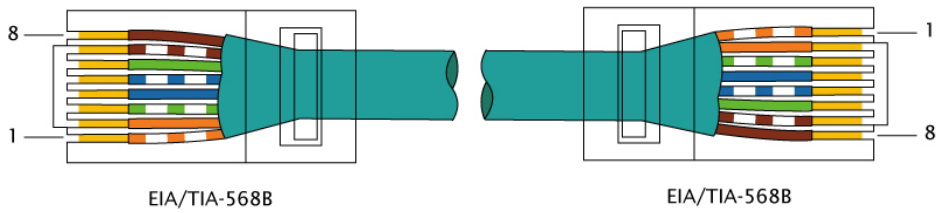


Figura 6-6: Cable CH-02- Conexión Servo – Servo.

- **La resistencia de terminación** del lado del servo drive (extremo izquierdo en Figura 6-3) se instala en uno de los puertos RJ45 del módulo CANOpen de dicho servo drive. Esta resistencia de 120 Ω deberá ir conectada a los pines 1 y 2 tal como aparece en la Figura 6-7.

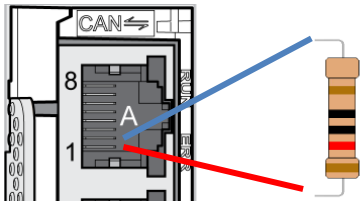


Figura 6-7: Resistencia de terminación del lado del servo drive

De cara a futuros proyectos, para redes CANOpen más complejas, en [16] se puede encontrar información más detallada para el diseño de estas (longitudes de cables, velocidad de transmisión, topología…)

6.1.4 Cableado de sincronización

Para la sincronización de los dos servomotores es necesario que el servo *Esclavo* reciba la señal de encoder del servo *Maestro*. La señal del encoder es una señal de pulsos rápidos por lo que, para transmitirla de un servo a otro, deben utilizarse los puertos PTI – PTO (pulse train input – pulse train output). Se debe utilizar un cable de conexión directa (pin a pin) RJ45 – RJ45 (como el utilizado para CH – 02). Se conectará al puerto PTO del *Maestro* y al puerto PTI del *Esclavo* (Figura 6-8).

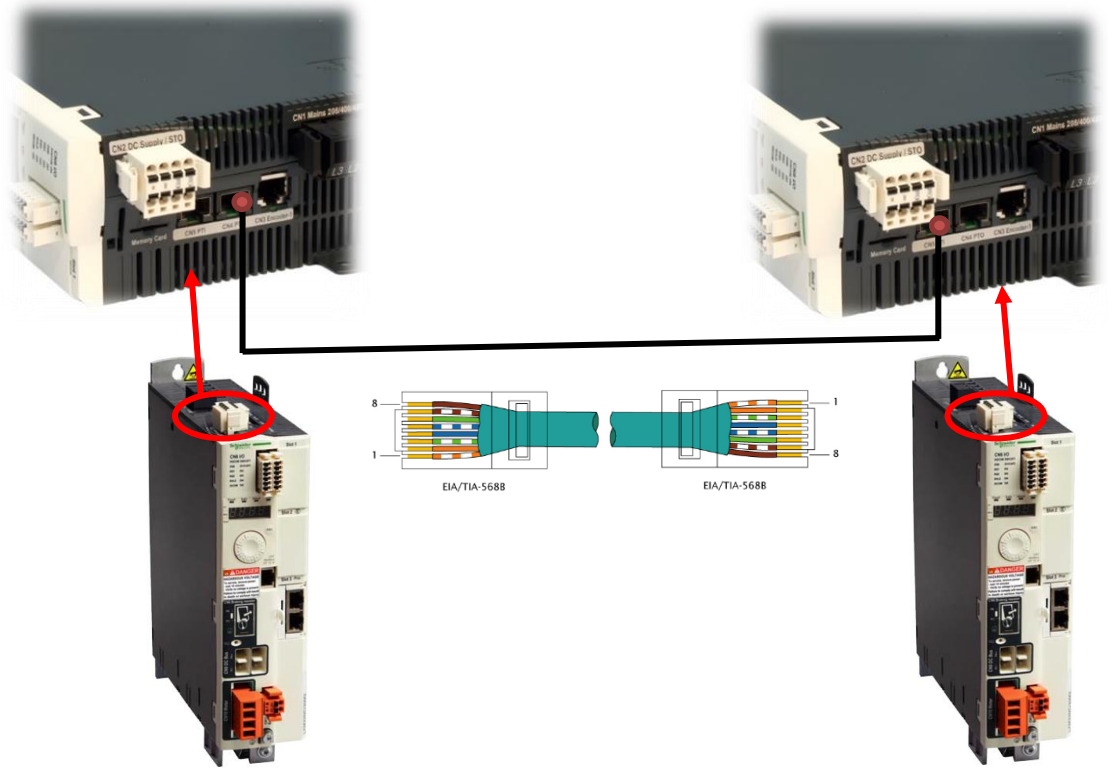


Figura 6-8: Cableado sincronización. Cable PTI – PTO.

6.2 Configuración Inicial del PLC

Para la programación del M340 debe hacerse uso del software *Unity Pro*. A continuación, se describe paso a paso la configuración inicial necesaria para poder programar la aplicación en dicho software.

1. En primer lugar, se abre *Unity Pro* y se genera un nuevo proyecto.

2. El programa pedirá que seleccione el PLC a programar. Se puede consultar el modelo concreto en la parte frontal superior del equipo tal como se muestra en la Figura 6-9. Aquí se emplea el modelo M340 BMX P34 20302.

Figura 6-9: Modelo del PLC

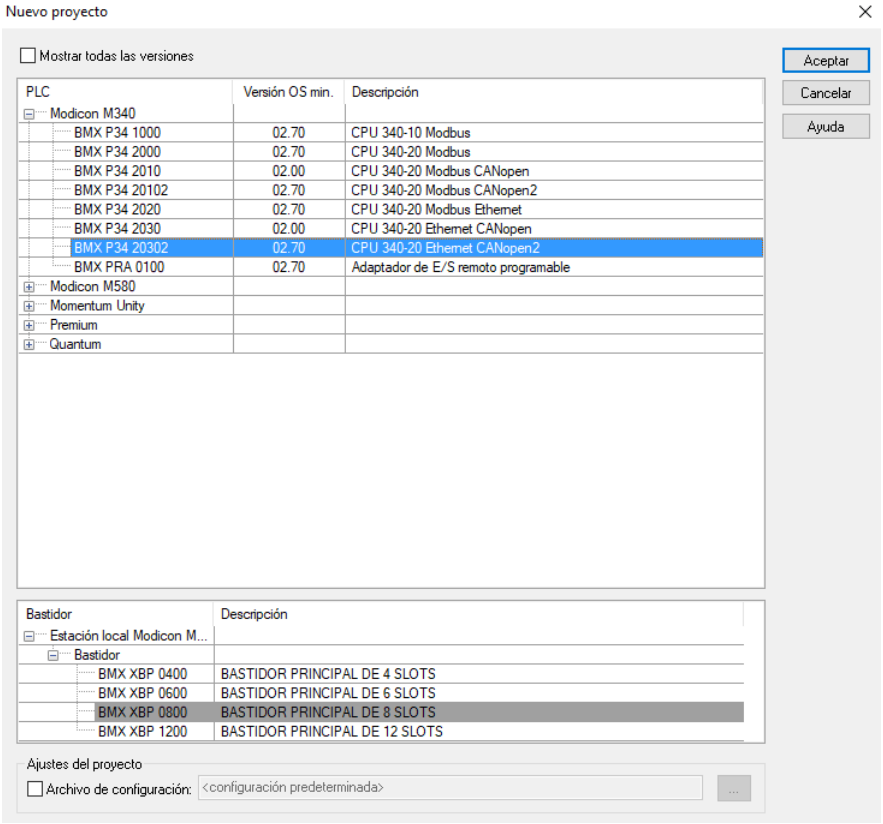


Figura 6-10: Selección modelo PLC. Unity Pro

3. A continuación, se tienen que añadir los servo drives a la red CANOpen del PLC. Para ello, en el navegador del proyecto hacer doble clic sobre "CANOpen". Se abrirá una ventana con un área rectangular en blanco (Figura 6-11). Al hacer doble clic sobre ella y aparecerá una ventana para añadir un nuevo dispositivo a la red.

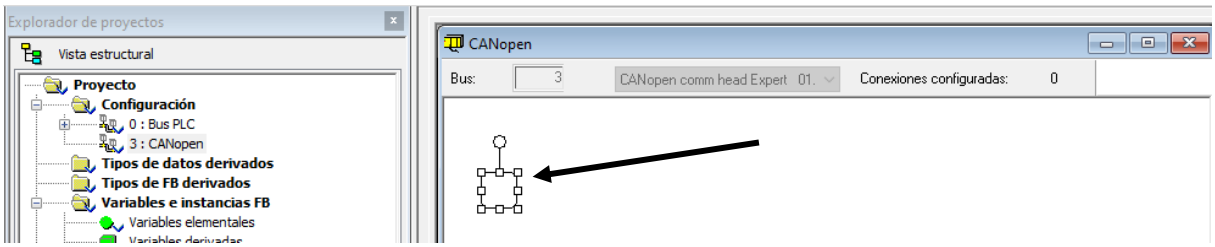


Figura 6-11: Red CANOpen del PLC. Unity Pro

4. Se busca nuestro servo drive para añadirlo. En el apartado "Movimiento y control" se selecciona "LXM32_MFB". Una vez añadido, aparecerá en la ventana de la red CAN mostrando el nodo asignado a dicho dispositivo (recuadrado en rojo en Figura 6-12). Haciendo doble clic sobre el nodo asignado puede modificarse.
   Repetir la acción para los dos servo drives y asignarle los nodos correspondientes. Estos deben coincidir con los asignados internamente (ver *CAN_Adress* Tabla 6-2).

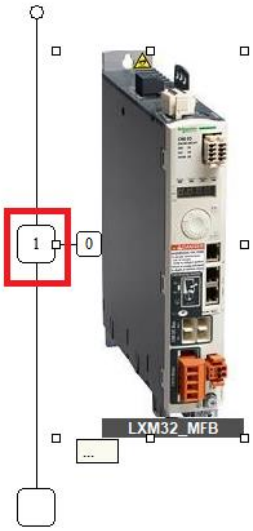


Figura 6-12: Dispositivo añadido a la red CANOpen del PLC. Unity Pro

5. Dentro del navegador del proyecto, hacer doble clic en "Bus PLC". Aparecerá el bastidor de nuestro PLC. Si se hace doble clic sobre el puerto CANOpen se abrirá la ventana de configuración de la red CAN donde es posible cambiar, entre otras cosas, la velocidad de transmisión de la red (Figura 6-14).

Figura 6-13: Puerto CANOpen en el bastidor del PLC. Unity Pro

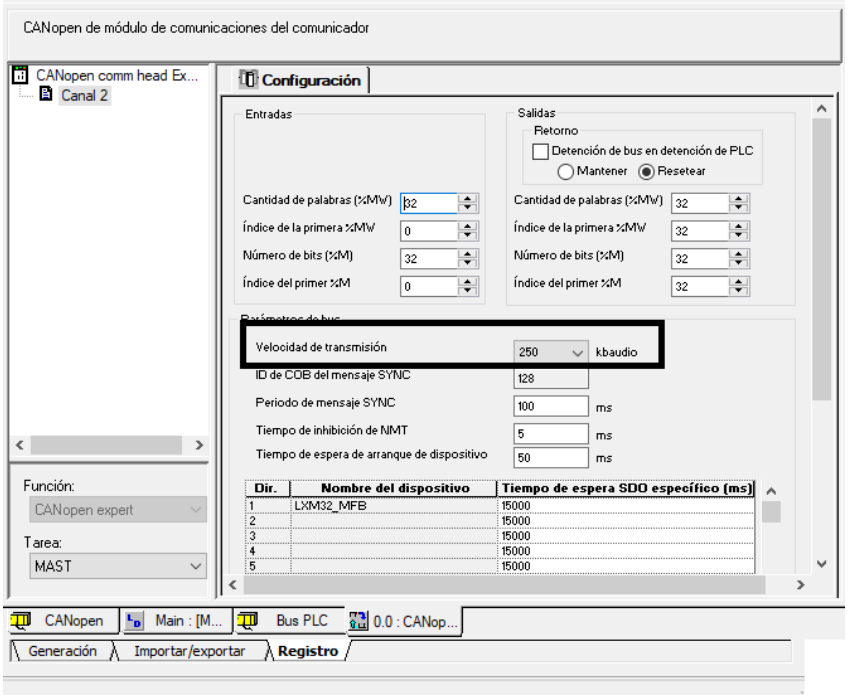


Figura 6-14: Ventana de configuración de la red CANOpen. Unity Pro

En principio, no es propósito de este proyecto entrar en configuraciones expertas por lo que se deja todo por defecto. Sin embargo, al igual que con el ID del nodo de cada servo drive, es importante asegurarse que la velocidad de transmisión establecida internamente en el servo drive es la misma que la establecida en el PLC (ver *CAN_Baud* en Tabla 6-2).

6. Por otro lado, también es necesario agregar el servo drive como eje para poder referenciarlo dentro de la programación del PLC. Para ello, en el explorador del proyecto hacer clic derecho sobre "Movimiento" y seleccionar "Eje nuevo". Se desplegará una ventana emergente que habrá que rellenar con los datos de cada servo drive.

   - En la lista de dispositivos disponibles seleccionar nuestro servo drive (Lexium 32)

   - En la lista de direcciones compatibles aparecerán los dispositivos de ese tipo que se tengan previamente configurados en la red CAN (paso 4). En la Figura 6-15 puede verse como se identifican dichas direcciones.

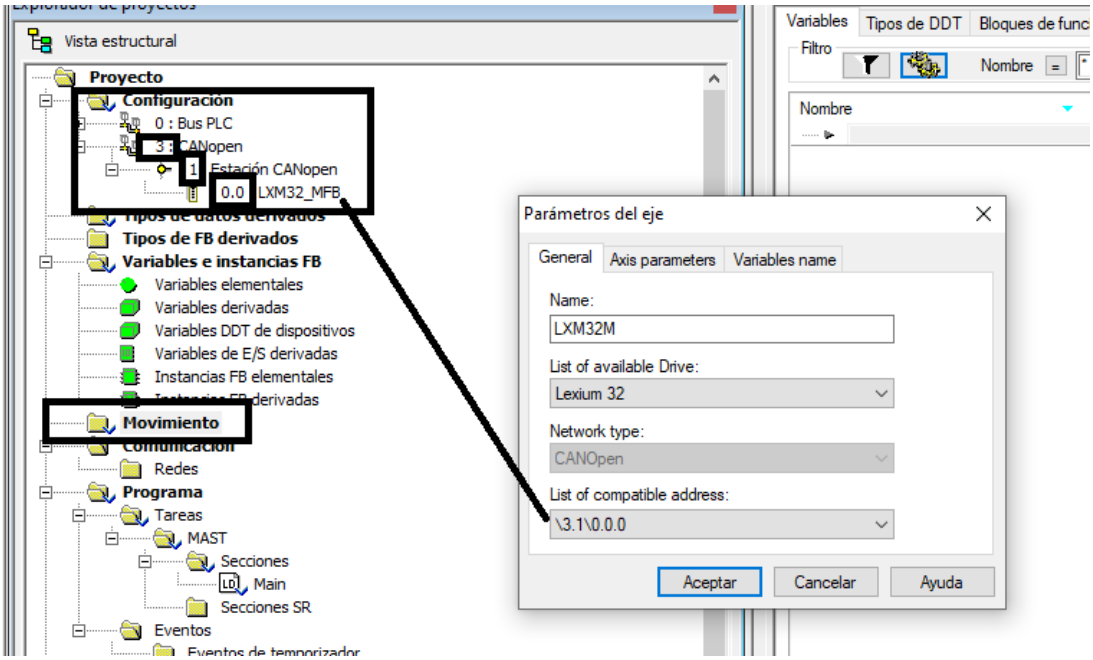


Figura 6-15: Dirección del dispositivo en los parámetros de un nuevo eje donde el "1" hace referencia al nodo asignado en la red CAN al dispositivo (ver *CAN_Adress* Tabla 6-2).

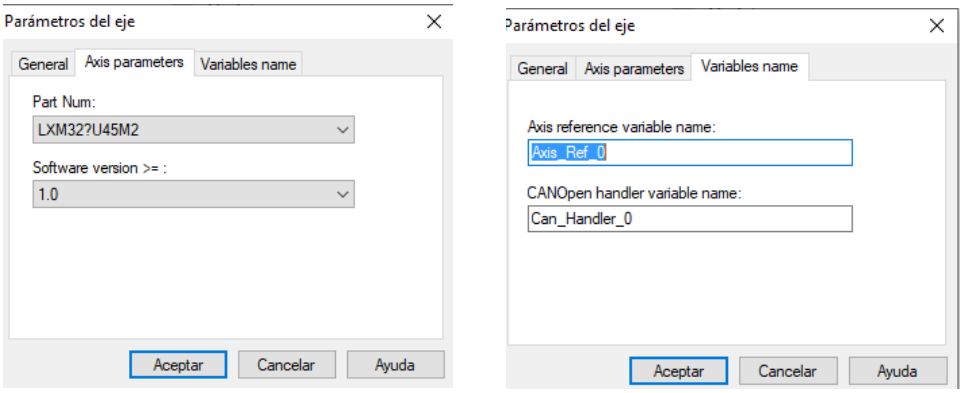


Figura 6-16: Parámetros de nuevo eje de movimiento. Unity Pro

   - En "Part Num" (Figura 6-16) se debe seleccionar el modelo concreto de nuestro Lexium32. Esto puede consultarse en el costado derecho del dispositivo o en la placa de características (Figura 2-3). En nuestro caso, se trata del **LXM32MU90M2**

   - Por último, en la pestaña de nombre de variables, se asigna el nombre que utilizaremos para hacer referencia al servo drive dentro de la programación del PLC (Axis reference variable name). También puede modificarse el nombre asignado a la variable CANOpen Handler.

Para profundizar en conceptos relacionados con la implementación de una red CANOpen en un PLC M340 puede consultarse [17].

6.3 Parametrización

Para la implementación de cualquier aplicación sobre servos es necesario establecer los valores de algunos de sus parámetros. **La parametrización inicial puede realizarse fácilmente a través del software de puesta en marcha (SoMove)** o bien a través del propio controlador remoto. En este caso, se ha optado por la primera opción al ser la que resulta más rápida e intuitiva y por lo tanto la más recomendada.

Sin embargo, si se quiere optar a la opción de usar **la configuración online desde el controlador remoto**, existen tres alternativas:

- Escribir cada parámetro que quiere configurarse a través de la función MC_WRITEPARAMETER de la librería *Motion Control* [11] (una llamada a la función por cada parámetro a actualizar). En este caso, basta con indicar para cada parámetro los siguientes datos:
  - Dirección en hexadecimal dentro del diccionario de parámetros (Índice y Subíndice)
  - Tamaño del parámetro en memoria (en bytes)
  - Valor del parámetro

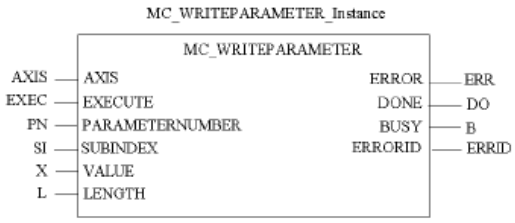


Figura 6-17: Bloque funcional MC_WRITEPARAMETER

Los dos primeros y el rango válido del tercero para cada parámetro pueden consultarse en el listado completo del LXM32M disponible en el apartado 10.2 de [5].

Esta alternativa es interesante cuando solo quieren modificarse unos pocos parámetros evitando que se tenga que conectar el servo drive con SoMove. Será utilizada en esta aplicación para configuraciones puntuales de algunos parámetros.

- Dentro de la ventana de configuración de cada servo drive en Unity pueden indicarse los valores de los parámetros que queramos establecer inicialmente (con la ejecución del programa del PLC). Los parámetros seleccionados, deben tener activa la casilla de configuración en la pestaña "Diccionario de objetos" para que el valor escrito en la pestaña "Configuración" sea transmitido al servo drive **durante el arranque** (Figura 6-18, Figura 6-19)

- Mediante la descarga de un juego de parámetros desde el PLC (apartado 6.3.3)

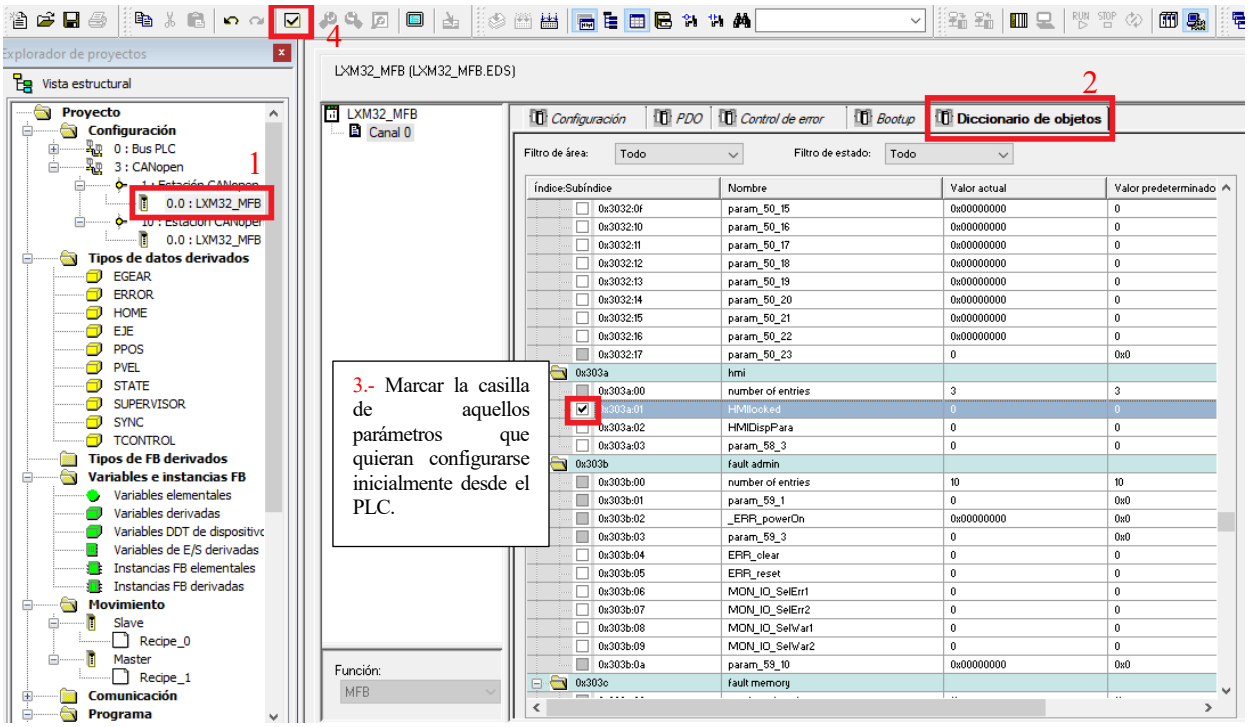


Figura 6-18: Parametrización inicial del servo drive desde la ventana de confg del PLC. Pasos 1-4

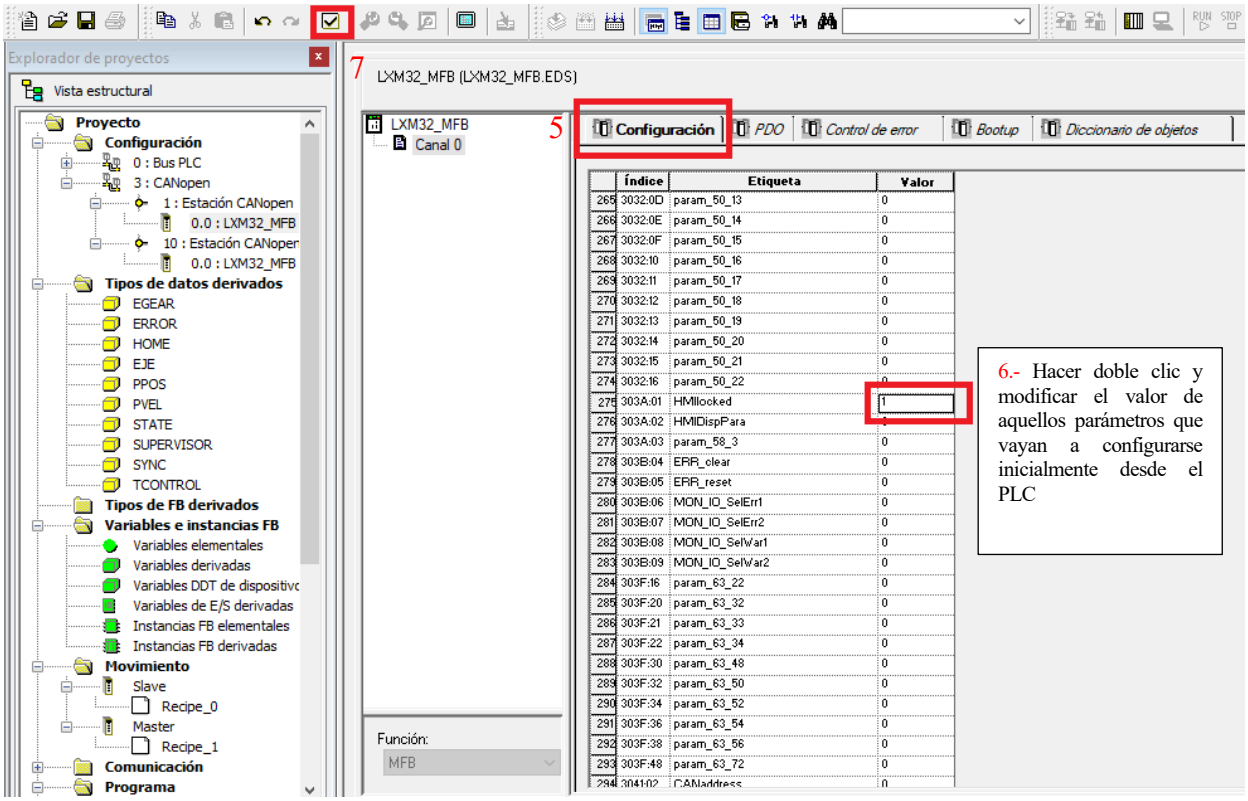


Figura 6-19: Parametrización inicial del servo drive desde la ventana de confg del PLC. Pasos 5-7

6.3.1 Parámetros iniciales

A continuación, se mencionan los parámetros que deben ser configurados al inicio antes de ejecutar la aplicación sobre el sistema real (maqueta).

- Dado que el servo drive puede controlarse en modo local o en modo bus de campo, es necesario indicar a través del parámetro *DEVcmdinterf* que el control es por bus de campo.

- Se debe establecer el nodo de cada servo drive en la red CAN a través de su parámetro *CAN_adress*. Por otro lado, se debe comprobar que la velocidad de transmisión *CAN_Baud* coincide con la configurada en la red CANOpen del PLC. En principio, en ambos dispositivos (PLC y servo drive) vendrá establecida a 250 kbaudio.

- Dado que nuestro sistema polea-correa es rotatorio, interesa establecer las posiciones de forma cíclica (en módulo) de manera que al llegar a un valor máximo vuelvan a 0. Para ello, se debe activar el modo módulo con el parámetro *MOD_enable* y establecer las posiciones que determinan el rango del módulo con *MOD_min* y *MOD_max*. Aquí se ha considerado el rango [0, 16500][12] usr_p. De esta forma, al rotar en sentido positivo, la posición aumentará de 0 hasta 16500 y volverá a 0 sucesivamente mientras siga en movimiento.

- Al trabajar con la posición en modo módulo es importante que el rango coincida con una vuelta completa de la correa. Para ello será necesario cambiar el escalado de usr_p a través de los parámetros *ScalePOSnum* y *ScalePOSdenom*.

- Como ya se comentó anteriormente, el *Maestro* debe transmitir su posición al *Esclavo* en todo momento. Para ello es necesario configurar la salida PTO a través del parámetro *PTO_mode*. En este caso, interesa transmitir la posición real dada por el encoder del *Maestro* (modo *Esim pAct Enc 1*). En la configuración del *Esclavo* nos aseguraremos que el tipo de señal de pulsos configurada en PTI (*PTI_signaltype*) es del tipo emitido por el *Maestro*. En este caso, se trata de la señal tipo A/D (cuatro estados) dada por el encoder instalado del servo.

- En función de cómo se haga la instalación, el sentido positivo de giro del servo será el deseado o no. Con el parámetro *InvertDirofMove* se puede cambiar el sentido positivo de giro (que será también el sentido positivo de crecimiento de la variable posición). En nuestro caso, el sentido positivo de giro será el definido con flechas en la Figura 6-2.

- Finalmente, hay que configurar las entradas/salidas digitales que sean necesarias a través de los parámetros *IOfunct_DIx* y *IOfunct_DQx* respectivamente. Para esta aplicación solo se ha considerado un interruptor de referencia (REF) para cada servo drive como señal de entrada.

Parámetro	Valor	Servo
DEVcmdinterf	2 (Fieldbus Control Mode)	Ambos
CAN_baud	250 (kBaud)	
MOD_enable	1 (Modulo On)	
MOD_min	0	
MOD_max	16500	
ScalePOSnum	100[13]	
ScalePOSdenom	162373[13]	
IOfunct_DI0	21 (Reference Switch)	
PTO_mode	1 (Esim pACt Enc 1)	*Maestro*
InvertDirofMove	1 (Inversion On)[13]	
CAN_adress	10	
PTI_signaltype	0 (type A/D)	*Esclavo*
InvertDirofMove	0 (Inversion Off)[13]	
CAN_adress	1	

Tabla 6-2: Parámetros Iniciales

[12] Se ha considerado este rango sin ningún motivo en especial. En principio, podría haberse escogido un valor con algún sentido concreto como, por ejemplo, que pudiera relacionarse fácilmente los cm recorridos con las unidades usr_p avanzadas.
[13] Valor obtenido para la maqueta descrita en el apartado 7.1.1

Para consultar todos los detalles de cualquiera de los parámetros (descripción, longitud en bytes que ocupa, dirección de acesso, rango de valores permitido y valor ajustado de fabrica) se deberá consultar el apartado 10.2 de [5]

6.3.2 Parámetros del regulador

Cuando el servo drive tiene establecido los ajustes de fábrica, es necesario realizar un ajuste inicial de los parámetros del regulador para asegurar un control adecuado del servo. Para dicho ajuste, en nuestros servo accionamientos se ha optado por un ajuste automático (*Easy Tunning*). A continuación, se muestran los resultados obtenidos con dicho ajuste.

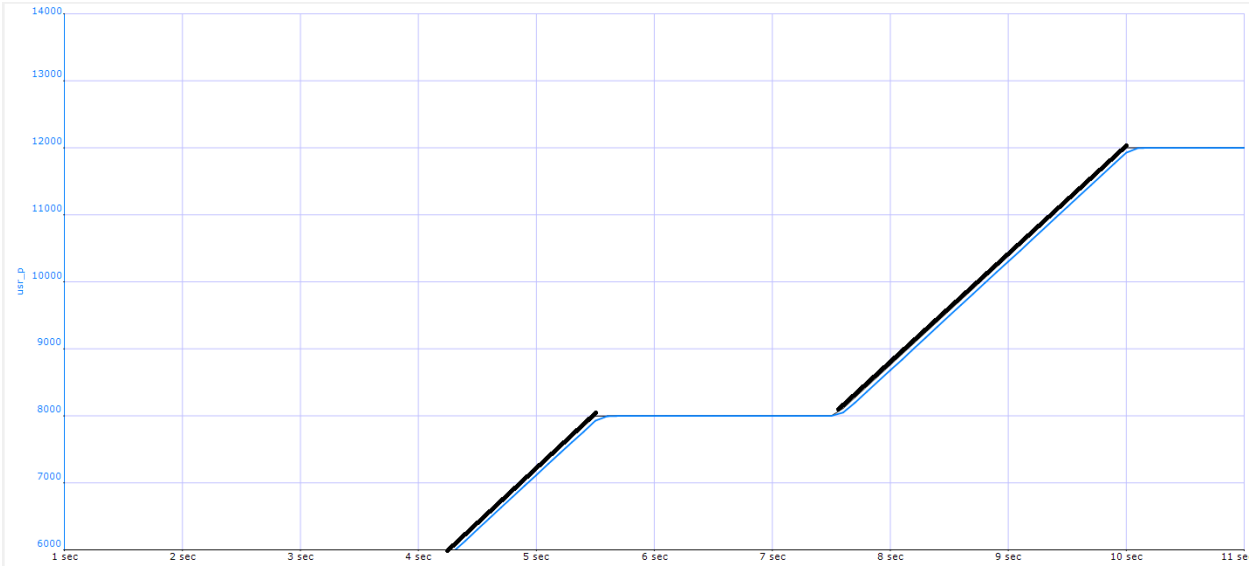


Figura 6-20: Control en posición. El error en régimen permanente es nulo mientras que en el transitorio mantiene un pequeño error respecto a la referencia (en negro)

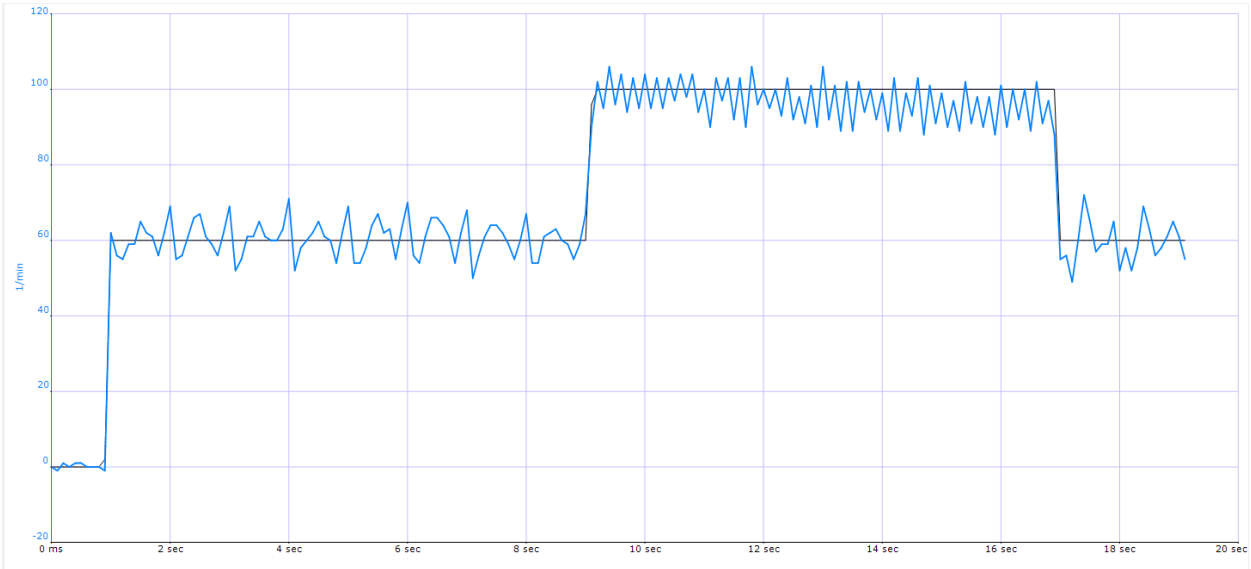


Figura 6-21: Control en velocidad. Referencia en negro.

Aunque no entre dentro del objetivo de esta implementación práctica, como puede observarse, el ajuste obtenido es bastante mejorable. Para ello podría mejorarse el montaje mecánico de la maqueta para reducir/evitar vibraciones o realizar un ajuste avanzado del controlador (*Expert Tunning*).

6.3.3 Cambio entre distintos juegos de parámetros

Cuando se agrega un servo drive como eje de movimiento (Paso 6, apartado 6.2) se generan con él las siguientes variables:

- **AxisParamDesc**: vector de enteros (ARRAY […] OF UINT) que incluye dirección y tamaño de cada parámetro del diccionario de parámetros del eje (Figura 6-22). Si se definen varios ejes del mismo tipo (mismo servo drive), Unity utilizará la misma variable para ambos. Esta variable de solo lectura servirá para la interpretación de las recetas de parámetros (Recipe)

Nombre	Tipo	Dirección	Valor	Comentario
AxisParamDesc_0	ARRAY[0..461] OF UINT			Type: 407, Ref: 14677, Vers: 1.0
AxisParamDesc_0[0]	UINT		407	AxisType
AxisParamDesc_0[1]	UINT		14677	AxisReference
AxisParamDesc_0[2]	UINT		1	MajorAxisSoftVersion
AxisParamDesc_0[3]	UINT		0	MinorAxisSoftVersion
AxisParamDesc_0[4]	UINT		152	NbParam
AxisParamDesc_0[5]	UINT		1	NetworkType
AxisParamDesc_0[6]	UINT		16#3005	MON_commutat
AxisParamDesc_0[7]	UINT		16#0005	
AxisParamDesc_0[8]	UINT		2	
AxisParamDesc_0[9]	UINT		16#3005	IO_AutoEnable
AxisParamDesc_0[10]	UINT		16#0006	
AxisParamDesc_0[11]	UINT		2	
AxisParamDesc_0[12]	UINT		16#3005	BRK_Add1_release

Figura 6-22: Vector *AxisParamDesc*. Si buscamos la información del parámetro `IO_AutoEnable` en [5] comprobamos que los valores remarcados se corresponden con su índice, subíndice y longitud en bytes

- **Recipe**: vector de bytes (ARRAY […] OF BYTE) que almacena los valores de los parámetros del eje. Se utilizan para actualizar/almacenar el diccionario de parámetros del servo. Se pueden crear varias recipes para un mismo eje haciendo clic derecho sobre su declaración en el apartado "Movimiento" del explorador de proyectos (paso 6, apartado 6.2)

La librería *Motion Control* (MFB) dispone de dos funciones: TE_UPLOADDRIVEPARAM y TE_DOWNLOADDRIVEPARAM que permiten:

- Cargar parámetros en un nuevo servo drive si uno de los instalados deja de funcionar adecuadamente.

- Cambiar el juego de parámetros del servo debido a un cambio en la operativa del mismo que lo requiera (si se contempla en la aplicación implementada)

Con TE_UPLOADDRIVEPARAM se puede realizar una copia de seguridad del diccionario de parámetros en una de las variables *Recipe* del eje correspondiente. La función necesita recibir como entrada la variable *AxisParamDesc* asociada al mismo eje.

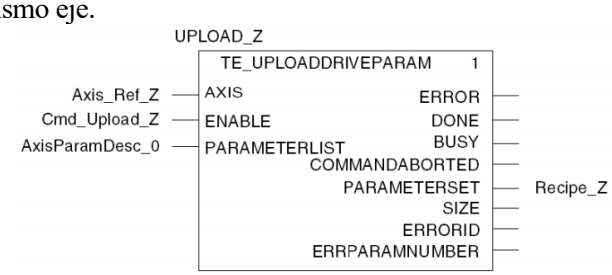


Figura 6-23: Función TE_UPLOADDRIVEPARAM.

Con TE_DOWNLOADDRIVEPARAM se pueden cargar en el servo los parámetros almacenados en una de sus recetas. La función necesita recibir como entrada la variable *AxisParamDesc* asociada al mismo eje.

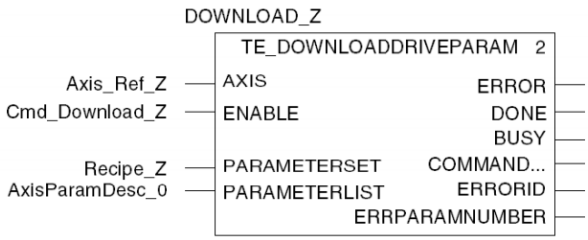


Figura 6-24: Función TE_DOWNLOADDRIVEPARAM

Con todo lo anterior, para realizar un cambio entre dos juegos de parámetros (A y B) desde el propio PLC habría que seguir los siguientes pasos.

1. Generar el juego de parámetros A con SoMove y guardarlo en la EEPROM (memoria) del servo drive.

2. Desde el PLC, manteniendo el estado del servo en "Ready to Switch On", activar la función TE_UPLOADDRIVEPARAM para el volcado sobre una receta del eje (*Recipe_A*). La edición directa de la *Recipe_A* nos ahorraría estos dos pasos, pero resultaría más tediosa y lenta.

3. Generar el juego de parámetros B con SoMove y guardarlo en la EEPROM (memoria) del servo drive.

4. Desde el PLC, manteniendo el estado del servo en "Ready to Switch On", activar la función TE_UPLOADDRIVEPARAM para el volcado sobre una receta del eje (*Recipe_B*).

5. En este caso, se empezaría a operar con el juego B

6. Cuando sea necesario el cambio de parámetros (al juego A), establecer el estado del servo en "Ready to Switch On" y activar la función TE_DOWNLOADDRIVEPARAM para cargar *Recipe_A*.

En un caso real en el que por fallo del equipo o mantenimiento hubiese que sustituirlo, el nuevo podría cargar la configuración del anterior siguiendo los pasos 1, 2 y 6.

Vista la utilidad de estas funciones, cabe mencionar que **no se contempla su utilización en esta aplicación práctica**.

6.4 Descripción funcional

Este apartado se centra en realizar una descripción funcional detallada de la aplicación.

- El sistema cuenta con dos modos: modo manual y modo sincronismo. Por defecto, el modo activo será el manual.

- En el **modo manual** el usuario maneja ambos servo accionamientos. Las acciones que puede llevar a cabo son las siguientes:
  - Lectura y escritura de los distintos parámetros del diccionario de objetos.
  - Configuración y activación de los siguientes modos de funcionamiento: Profile Position, Profile Velocity, Profile Torque, Home y Electronic Gear.
  - Activación de la etapa de potencia.
  - Reinicio de errores del servo drive.
  - Detención del movimiento en curso.

- En el **modo sincronismo** el servo que acciona el sistema polea-correa superior es el *Maestro* y el que acciona el sistema polea-correa inferior es el *Esclavo*. Cuenta con una versión básica y una versión avanzada. El usuario podrá volver en cualquier momento del modo sincronismo al modo manual en ambos ejes. Para ello, se interrumpirá el movimiento en curso (STOP).

- En la **versión básica del modo sincronismo**:
  - En primer lugar, ambos realizan un HOME para colocar su posición "0" de referencia bajo el correspondiente sensor inductivo (*Switch REF*) indicado en el croquis de la Figura 6-2. De esta forma, la posición marcada por el encoder de cada servo drive se corresponderá con la del indicador metálico instalado en sistema polea-correa que acciona.

  - Tras el HOME, el *Maestro* espera una señal de activación con la cual comienza a girar, por defecto, en sentido positivo a una velocidad predefinida por el usuario antes de la ejecución de la secuencia.

- Tras el HOME, el *Esclavo* espera a una señal (*ENGAGE*) que indica que debe sincronizarse con el *Maestro.* Para ello, se comienza a mover a una determinada velocidad en busca de igualar su posición con la del *Maestro* (con una pequeña holgura predefinida por el usuario). El sentido de giro lo determina el camino más corto para conseguir dicha sincronización.

- Cuando el *Esclavo* haya conseguido sincronizarse, cambiará su modo de funcionamiento a Electronic Gear para moverse en consonancia con el *Maestro* (haciendo uso de la señal que recibirá por el puerto PTI). Es decir, el *Esclavo* copiará el movimiento del *Maestro*.

- La sincronización es visible físicamente gracias a los indicadores colocados en ambas correas. Además, es necesario que el sentido de giro positivo de los servomotores sea el indicado en la misma Figura 6-2.

- El *Esclavo* puede sincronizarse/desincronizarse en cualquier momento mediante la señal antes mencionada (*ENGAGE*).

- En la **versión avanzada del modo sincronismo**:

  - En primer lugar, ambos realizan un HOME para colocar su posición "0" de referencia bajo el correspondiente sensor inductivo (*Switch REF*) indicado en el croquis de la Figura 6-2. De esta forma, la posición marcada por el encoder de cada servo drive se corresponderá con la del indicador metálico instalado en sistema polea-correa que acciona.

  - Tras el HOME, el *Maestro* espera una señal de activación con la cual comienza a girar, por defecto, en sentido positivo a una velocidad predefinida por el usuario antes de la ejecución de la secuencia.

  - Tras el HOME, el *Esclavo* espera a una señal (*ENGAGE*) que indica que debe sincronizarse con el *Maestro*. Para ello, se comienza a mover a una determinada velocidad. Con este movimiento el *Esclavo* busca que su indicador metálico se sincronice (con una pequeña holgura predefinida por el usuario) con el indicador del *Maestro* que se encuentre en ese momento en la parte inferior del sistema polea-correa (ya sea el metálico o el no metálico). El sentido de giro lo determina el camino más corto para conseguir dicha sincronización.

  - Cuando el indicador del *Esclavo* haya conseguido sincronizarse, cambiará su modo de funcionamiento a Electronic Gear para moverse en consonancia con el *Maestro* (haciendo uso de la señal que recibirá por el puerto PTI).

  - Cuando el indicador del *Maestro* con el que se encuentra sincronizado el *Esclavo* pasa a la parte superior del sistema polea-correa, el *Esclavo* invierte su sentido de giro para buscar sincronizar su indicador con el otro indicador del *Maestro* que ahora ha pasado a la parte inferior. De esta forma, el movimiento de sincronismo del *Esclavo* estará limitado a la parte superior de su sistema polea-correa.

  - La sincronización es visible físicamente gracias a los indicadores colocados en ambas correas. Además, es necesario que el sentido de giro positivo de los servomotores sea el indicado en la Figura 6-2.

  - El *Esclavo* puede sincronizarse/desincronizarse en cualquier momento mediante la señal antes mencionada (*ENGAGE*)

- El usuario dispone de una **ventana Supervisor** (apartado 6.6) que permite llevar a cabo todas las operaciones descritas anteriormente, así como la monitorización de ambos servo drives.

6.5 Implementación

Después de haber montado la maqueta de pruebas, haber realizado la configuración inicial de los servos y haber desarrollado la descripción funcional de la aplicación, el siguiente paso consiste en el desarrollo de la aplicación en sí mediante:

- Declaración de las variables necesarias.
- Programación de las tareas del PLC para el control de los servo accionamientos.
- Depuración de la aplicación.

Para una mejor compresión de este apartado se aconseja su lectura teniendo por delante el proyecto abierto en *Unity Pro*.

6.5.1 Declaración de variables

Para la programación y el manejo de la aplicación será necesario la declaración de una serie de variables. La inmensa mayoría de ellas se agrupan en estructuras de datos (Figura 6-25).

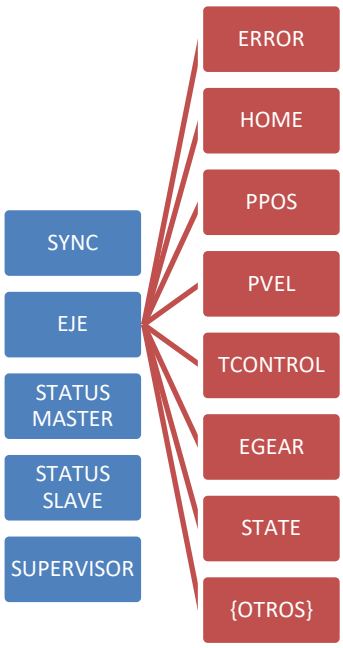


Figura 6-25: Estructuras o tipos de datos derivados.

- **SYNC**: Agrupa todas las variables relacionadas directamente con el modo sincronismo de la aplicación.
- **EJE:** Agrupa todas las variables que están asociadas con uno de los ejes en concreto. Por lo tanto, se definen dos estructuras tipo eje: *Esclavo* y *Maestro*.
  - **ERROR:** Variables que marcan los errores del servo drive durante la ejecución.
  - **HOME:** Variables asociadas a la configuración del modo Homing.
  - **PPOS**: Variables asociadas a la configuración del modo Profile Position.
  - **PVEL**: Variables asociadas a la configuración del modo Profile Velocity.
  - **TCONTROL**: Variables asociadas a la configuración del modo Profile Torque.
  - **EGEAR**: Variables asociadas a la configuración del modo Electronic Gear.
  - **STATE**: Variables que sirven para informar sobre el estado de los controles básicos del eje.
  - **{OTROS}**: La estructura eje incluye otras variables no agrupadas en subestructuras que sirven esencialmente como:
    - **Variables para ejecutar acciones**: Power, Reset, Stop, Escribir parámetro…
    - **Variables para consultar información**: Valor del parámetro leído, modo de operación activo…
- **STATUS MASTER**: Variables que sirven para informar sobre el estado del servomotor *Maestro*.
- **STATUS SLAVE**: Variables que sirven para informar sobre el estado del servomotor *Esclavo.*
- **SUPERVISOR:** Variables auxiliares para el funcionamiento del panel de operador.

Además de dichas variables, se encuentran las variables asociadas a la definición de cada servo drive como eje (*Can_Handler_x*, *Axis_Ref_x*, *AxisParamDesc_x*, *Recipe_x*).

A lo largo del resto del capítulo, se destacan los nombres de variables en verde para distinguirlas del resto del texto.

6.5.2 Programación de la aplicación

La programación de la aplicación está basada fundamentalmente en la utilización de las funciones de la librería *Motion Control* (MFB). La mayoría de los bloques funcionales tienen en común una serie de entradas y salidas básicas (Figura 6-26):

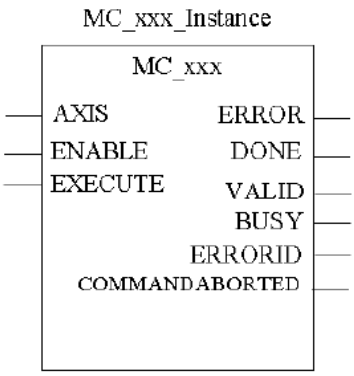


Figura 6-26: Entradas y Salidas básicas de los FB de la librería *Motion Control*.

- **AXIS:** Variable tipo *AXIS_REF* asociada al eje al cual va dirigida la función. En nuestro caso será *Axis_Ref_Master* o *Axis_Ref_Slave* dependiendo de si aplica al eje *Maestro* o al eje *Esclavo*.
- **ENABLE/EXECUTE:** Variable booleana para la habilitación/ejecución de la función. Con *ENABLE* el bloque funcional se ejecutará en cada ciclo de autómata mientras esta sea verdadera. Con *EXECUTE* el bloque funcional se ejecutará una sola vez cuando se produzca un flanco de subida.
- **ERROR:** Booleano que indica la aparición de un error en la ejecución de la función.
- **DONE/BUSY:** Booleano que indica la finalización/ejecución de la función.
- **ERRORID:** Variable entera que sirve como identificador de errores.

La aplicación se estructura en 4 secciones programadas en Ladder (LD)

- **Main:** Incluye todas las funciones requeridas por el modo manual.
- **Master Control:** Incluye la programación del modo sincronismo asociada al eje *Maestro*.
- **Slave Control Básico:** Incluye la programación del modo sincronismo básico asociada al eje *Esclavo*. Su ejecución está condicionada a la elección del modo básico de sincronismo mediante la variable *SYNC.basico*.
- **Slave Control Avanzado:** Incluye la programación del modo sincronismo avanzado asociada al eje *Esclavo*. Su ejecución está condicionada a la elección del modo avanzado de sincronismo mediante la variable *SYNC.avanzado*.

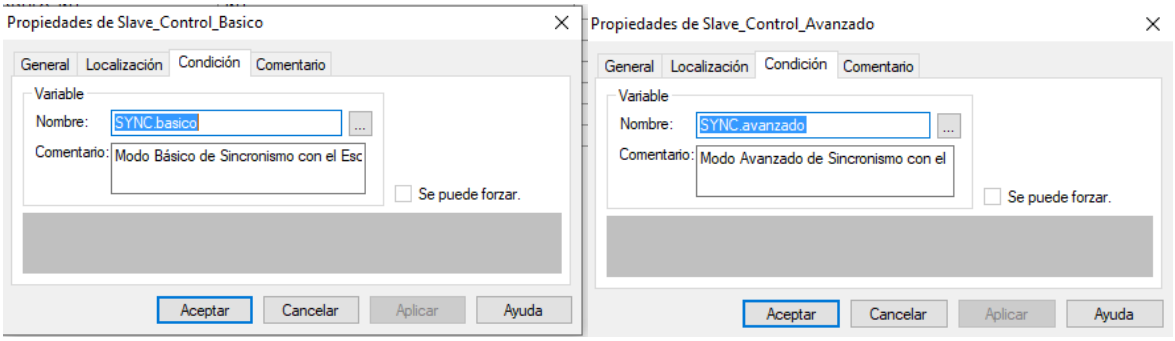


Figura 6-27: Condición de ejecución de los modos de control Básico y Avanzado del *Esclavo*.

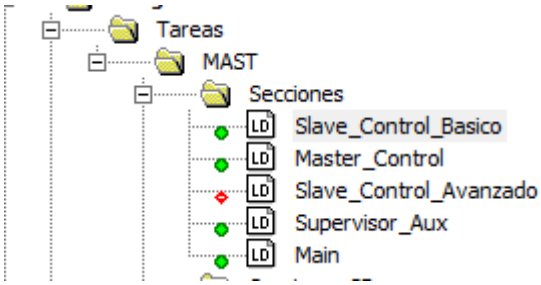


Figura 6-28: Ejemplo de ejecución con el modo básico activado. En el explorador del proyecto puede verse como el LD del control avanzado está desactivado.

6.5.2.1 Main

En este apartado se detallarán los distintos aspectos sobre la programación de la sección LD con el mismo nombre encargada de implementar todas las funciones requeridas para el funcionamiento del modo manual de la aplicación.

En primer lugar, para la utilización del resto de funciones de la librería *Motion Control* es necesario instanciar la función *CAN_HANDLER* para cada eje definido. Esta función se utiliza para comprobar que la comunicación CANOpen entre PLC y servo drive es correcta. El parámetro *NETWORKOPERATIONAL* debe asignarse a un bit que valide el correcto funcionamiento de la comunicación del eje a través de la red. Una posibilidad es utilizar la variable *SLAVE_ACTIV_X*, procedente del IODDT[14] del puerto CANOpen del M340. Dicha variable indica si el dispositivo asociado al nodo X está activo.

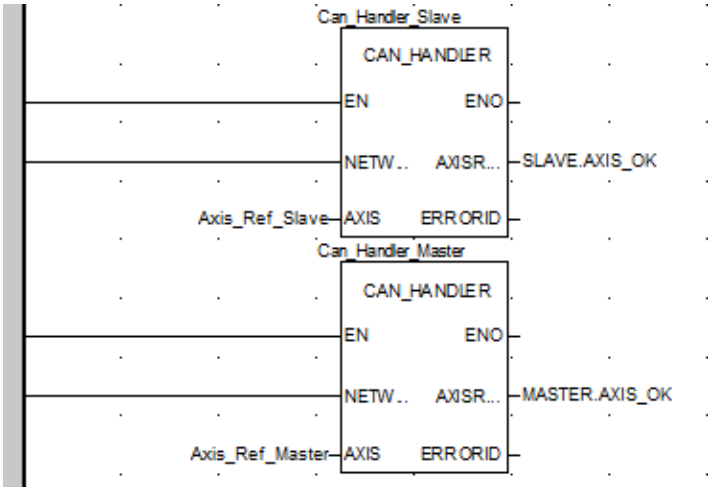


Figura 6-29: Instancias CAN_HANDLER

Configuración variable IODDT.

Para la configuración de la variable IODDT asociada al módulo CANOpen del M340 se deben seguir los siguientes pasos:

1. Hacemos clic sobre el puerto CANOpen que aparece en el navegador del proyecto.
2. En la ventana que se despliega, seleccionamos la pestaña "Objetos de E/S".
3. Marcamos la casilla %CH y hacemos clic sobre "Actualizar Cuadrícula". Nos aparecerá un canal disponible al que le podremos asociar la variable IODDT de tipo "T_COM_CO_BMX_EXPERT".
4. Le asignamos un nombre y hacemos clic en crear
5. Si hacemos clic en la sección "Variables e instancias FB" podremos consultar la variable IODDT creada y todas las variables simples asociadas, entre ellas, las variables *SLAVE_ACTIV_X*.

[14] IOODT: Tipo de datos derivados (estructura de variables simples) de entradas y salidas asociadas a un módulo físico que proporcionan información relacionada con el mismo.

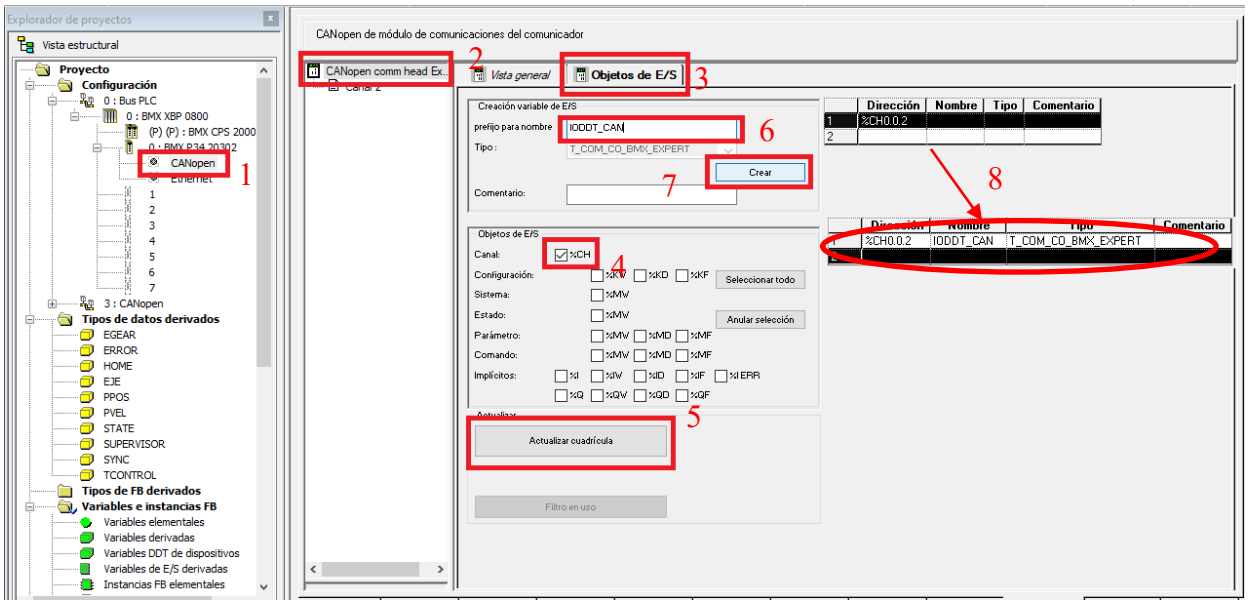


Figura 6-30: Configuración variable IODDT CANOpen en M340. Pasos 1-4 (desglosados en 8)

Tras las instancias de *CAN_HANDLER* para ambos servos, se incluyen las funciones MC_POWER, MC_RESET y MC_STOP que permiten la activación de la etapa de potencia, reinicio e interrupción del movimiento de los dos servo drive respectivamente. Como la ejecución del MC_RESET es instantánea, se ha añadido un bloque SAMPLE_TM para retrasar la desactivación de la señal *SLAVE/MASTER.STATE.RESET* que servirá para indicar la ejecución exitosa en el Supervisor.

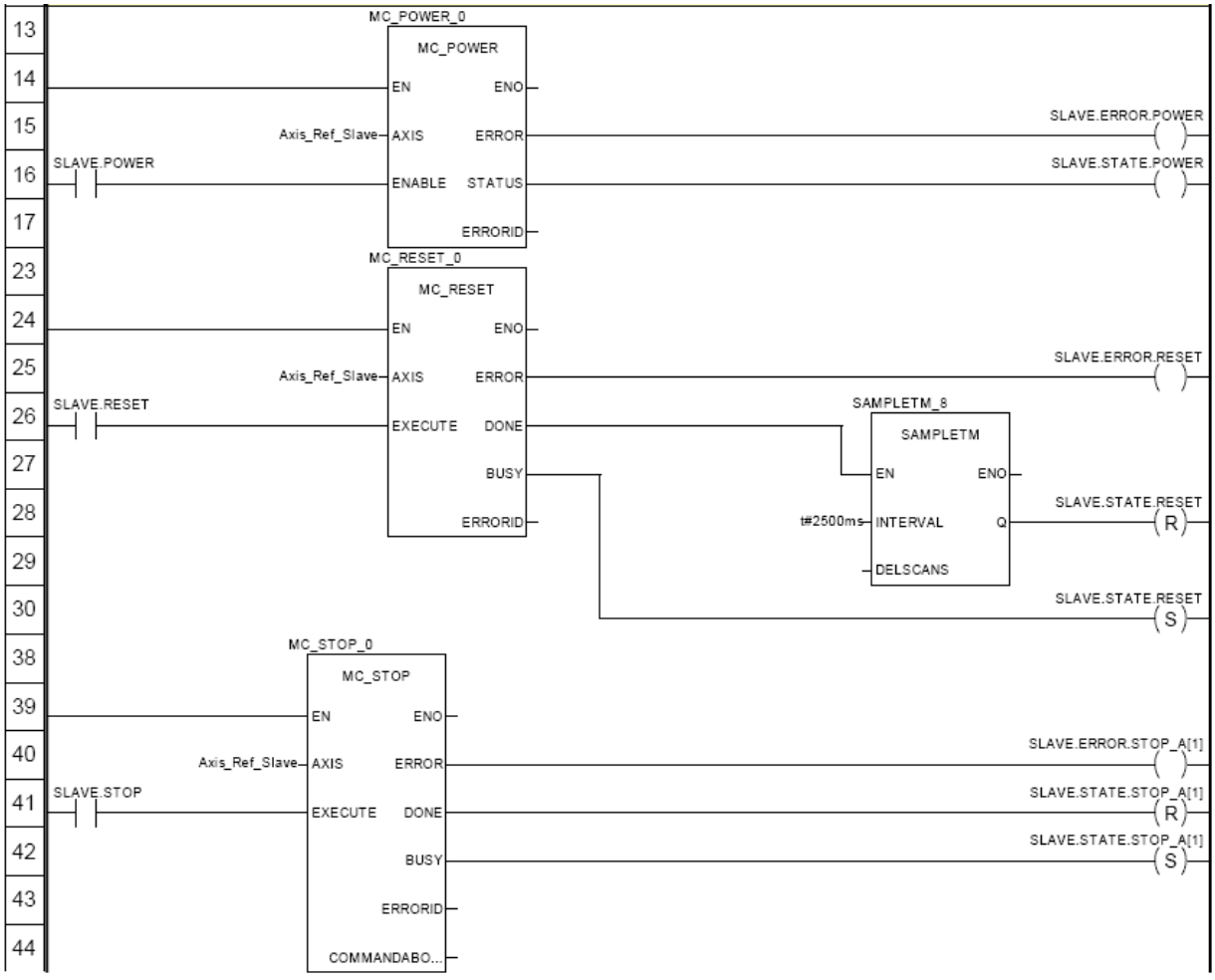


Figura 6-31: Funciones MC_POWER, MC_RESET, MC_STOP

Para la monitorización y la programación de ciertas funciones algunos de los parámetros de los servomotores deben ser leídos por el PLC de forma cíclica haciendo uso de la función MC_READPARAMETER (Figura 6-32) que hace una petición SDO (Anexo A) al servo drive objetivo. El bloque SAMPLETM se encarga de llamar cíclicamente a la función de lectura

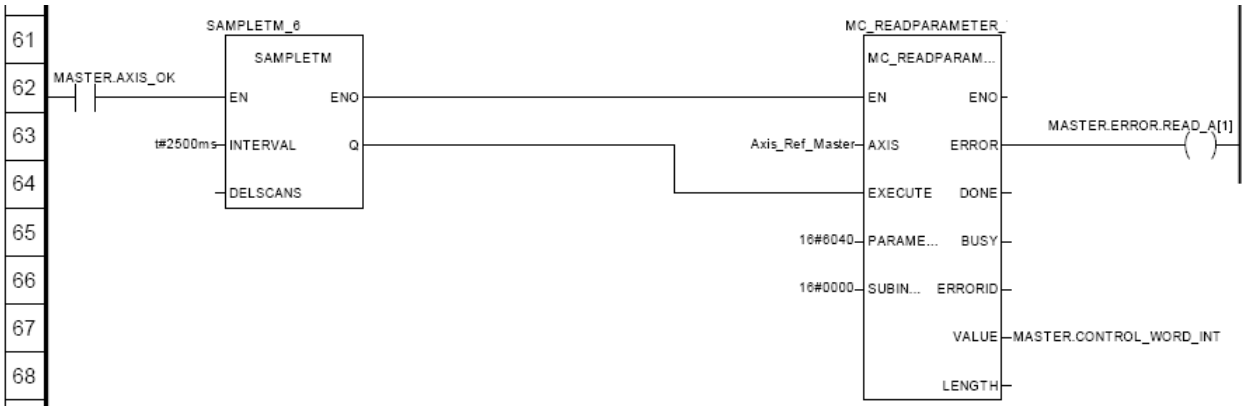


Figura 6-32: Ejemplo de lectura cíclica.

La dirección del parámetro al que se accede con la petición SDO debe expresarse en hexadecimal, tanto el índice (*PARAMETERNUMBER*) como el subíndice (*SUBINDEX*).
Para esta aplicación se ha decidido hacer la lectura cíclica de:

- `DCOMcontrol` (en Unity, *CONTROL_WORD*) entero que permite controlar el estado operativo del servo siendo modificado bit a bit.
- `_DCOMstatus` (en Unity, *STATUS_WORD*) entero que permite monitorizar el estado operativo del servo siendo analizado bit a bit.
- `_tq_act` (en Unity, *PAR*_ACTUAL) valor real del par en décimas de % frente al par de parada en continua (`_M_M_0` que para nuestro servo drive es de 50Ncm).
- `DCOMopmode` (en Unity, *OPMODE*) entero que indica el modo de funcionamiento activo en el servo drive.

Sin embargo, las dos variables más importantes que se precisan conocer en todo momento son la posición y la velocidad del eje. Se podrían obtener mediante SDOs como las variables anteriores o mediante recepción cíclica de los PDO emitido por el servo.
La recepción de los PDOs puede activarse desde la ventana de configuración de cada servo drive. El PDO 2 que transmite cada servo, lleva la información de velocidad y posición (32 bits cada una) y son almacenadas en palabras de memoria (16 bits) consecutivas (p.ej %MW4 y %MW5 para la posición del eje *Esclavo*) (Figura 6-33). Ambas palabras de memoria deben ser combinadas para obtener el valor completo de la variable (Figura 6-34).

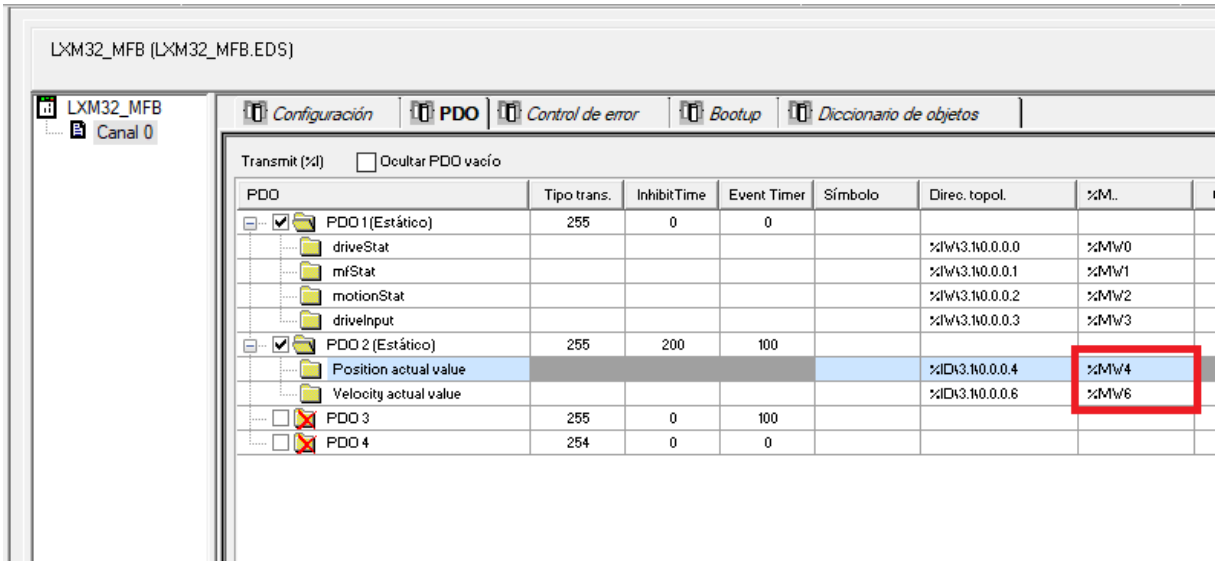


Figura 6-33: Pestaña PDO de la ventana de configuración del servo *Esclavo* (se accede desde el explorador de proyectos como en Figura 6-18)

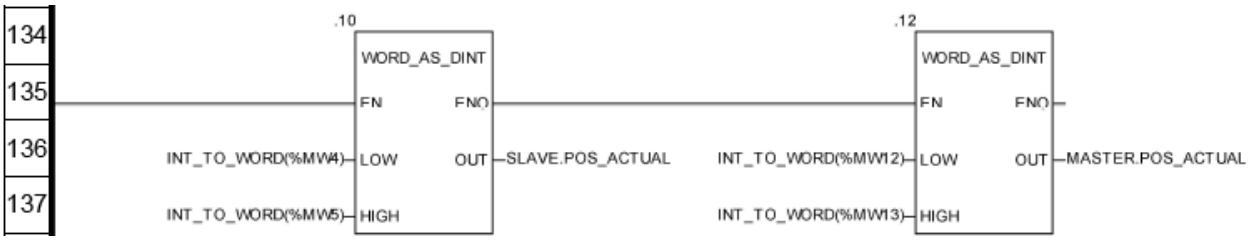


Figura 6-34: Combinación palabras de memoria (%MW) de los PDOs para obtener posición del servo.

A continuación, se descompone la *CONTROL_WORD* en bits para poder generar, de forma alternativa a las funciones MC_RESET y MC_STOP, reinicios de error (FAULT RESET) e interrupciones de movimiento (HALT). Para ello, será necesaria la modificación de los bits 7 y 8 tal como se muestra en la Tabla 6-3.

Orden de usuario	Bit 7	Bit 8
FAULT_RESET	1	0
HALT	0	1

Tabla 6-3: Gestión FAULT RESET y HALT desde variable *CONTROL_WORD*

La *CONTROL_WORD* se sobrescribe utilizando la función MC_WRITEPARAMETER. Para esta función, además de aportar la dirección del parámetro (`DCOMcontrol`) en hexadecimal (*PARAMETERNUMBER* y *SUBINDEX*), se debe proporcionar la longitud en bytes del parámetro a escribir (`DCOMcontrol` es de tipo UINT16 lo que supone una longitud de 2 bytes).

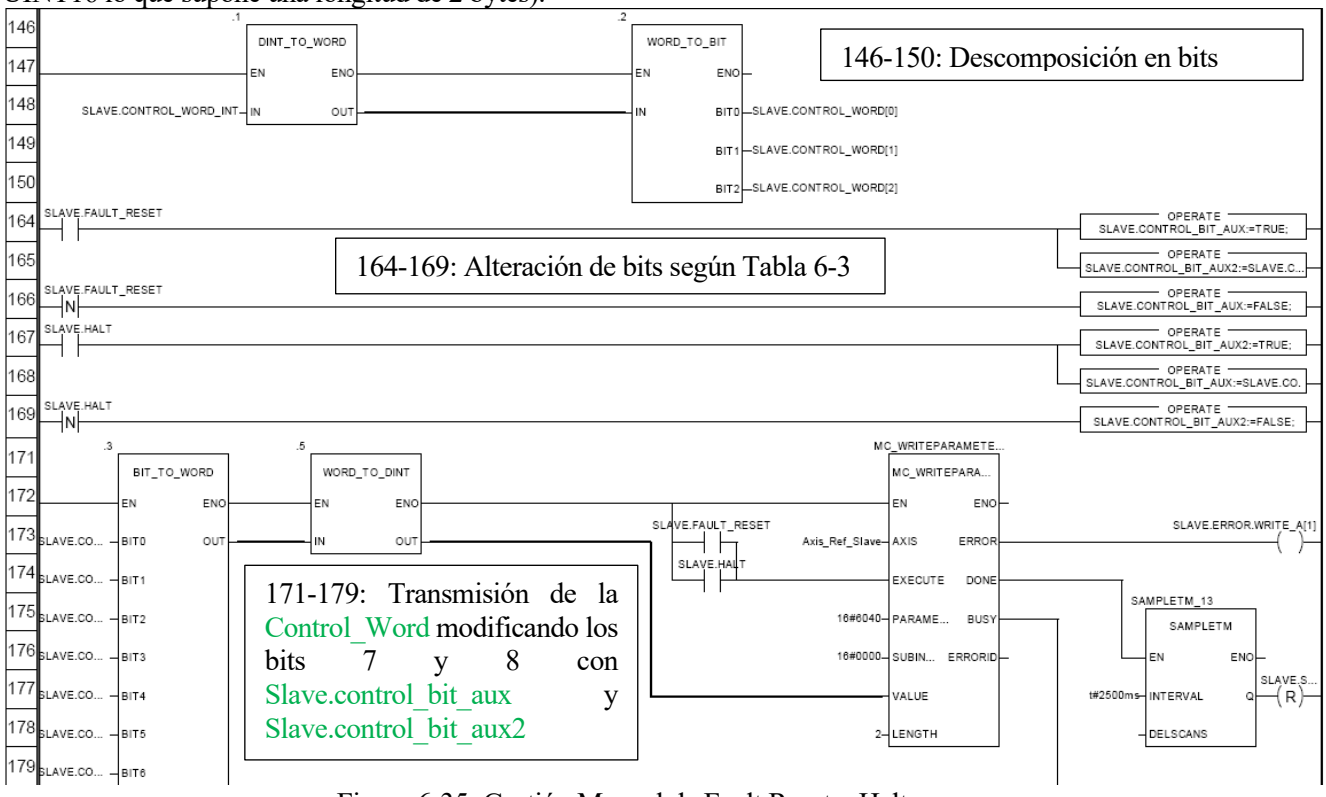


Figura 6-35: Gestión Manual de Fault Reset y Halt.

Esta sección también incluye una función genérica de lectura (MC_READPARAMETER) y otra de escritura (MC_WRITEPARAMETER) de parámetros para que el usuario pueda leer/escribir cualquier parámetro disponible en ambos servos.

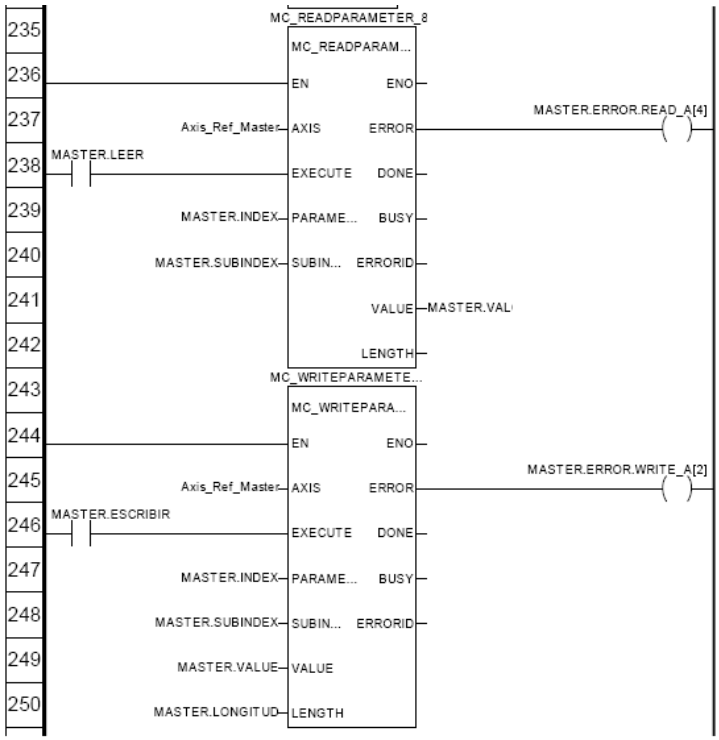


Figura 6-36: Implementación de lectura/escritura genérica de parámetros en *Unity Pro*.

Por último, se recogen la implementación de los distintos modos de funcionamiento.
HOMING: Utilizando la función MC_HOME y proporcionando como entrada las variables *MASTER.HOME* o *SLAVE.HOME* según corresponda al servo *Maestro* o al servo *Esclavo* respectivamente.

Variable de entrada	Parámetros que modifica	Descripción
HOME.HOME	`DCOMcontrol` (bit 4) `DCOMopmode`	Variable booleana de activación del modo Homing.
HOME.REF_POS	`HMp_home`	Valor numérico que queda marcado en la posición física final del movimiento Home. Por defecto será 0.
HOME.SPEED	`HMv`	Velocidad del eje durante el movimiento Home.
HOME.HTYPE	`HMmethod`	Tipo de Home a realizar.

Tabla 6-4: Variables MC_HOME

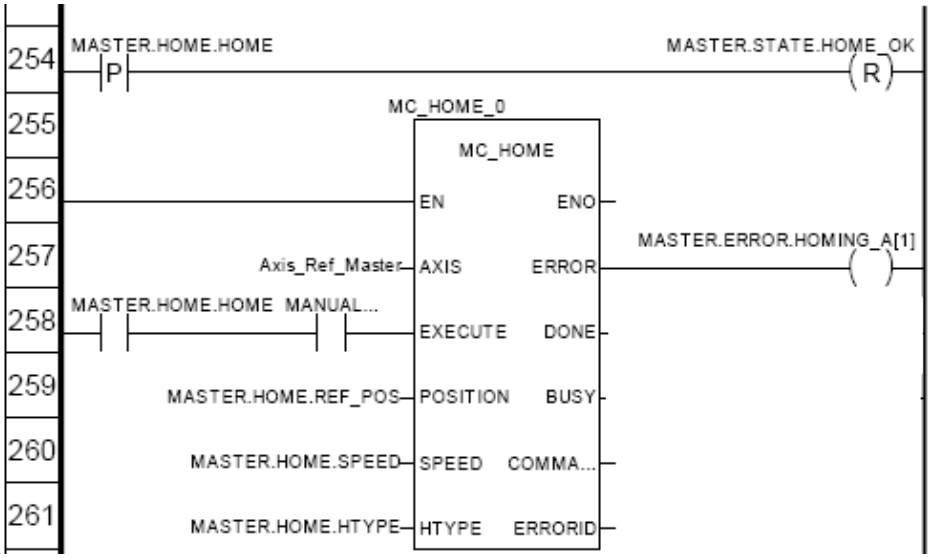


Figura 6-37: Implementación Homing en *Unity Pro*.

PROFILE POSITION: Utilizando las funciones MC_MOVEABSOLUTE y MC_MOVERELATIVE y proporcionando como entrada las variables *MASTER.PPOS* o *SLAVE.PPOS* según corresponda al servo *Maestro* o al servo *Esclavo* respectivamente.

Variable de entrada	Parámetros que modifica	Descripción
PPOS.POSICIONAR	`DCOMcontrol` (bit 4,5,6 y 9) `DCOMopmode`	Variable booleana de activación del modo Profile Position
PPOS.ABS	-	Variable booleana de elección entre movimiento absoluto y relativo
PPOS.REF	`PPp_target`	Posición objetivo
PPOS.VEL	`PVv_target`	Velocidad de movimiento
PPOS.ACCEL	`RAMP_v_acc`	Aceleración del movimiento (si es 0, se coge el valor por defecto guardado en el servo)
PPOS.DECCEL	`RAMP_v_dec`	Deceleración del movimiento (si es 0, se coge el valor por defecto guardado en el servo)

Tabla 6-5: Variables MC_MOVEABSOLUTE y MC_MOVERELATIVE

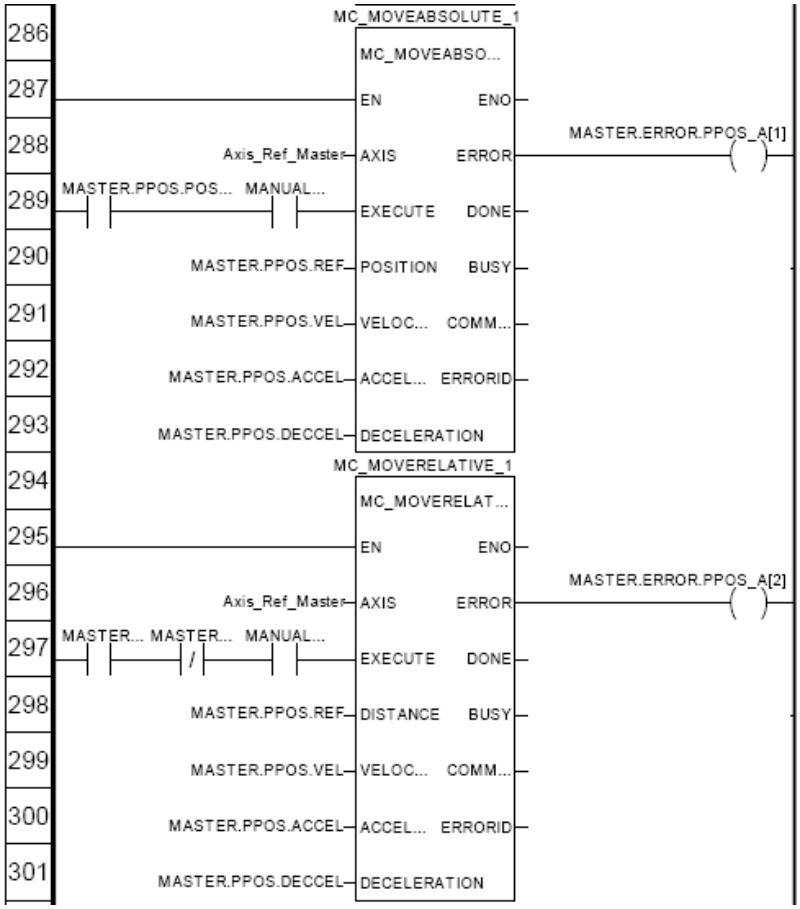


Figura 6-38: Implementación del Modo Profile Position en *Unity Pro*

PROFILE VELOCITY: Utilizando la función MC_MOVEVELOCITY y proporcionando como entrada las variables *MASTER.PVEL* o *SLAVE.PVEL* según corresponda al servo *Maestro* o al servo *Esclavo* respectivamente.

Variable de entrada	Parámetros que modifica	Descripción
PVEL.PVEL	*DCOMopmode* *PVv_reference*	Variable booleana de activación del modo Profile Velocity
PVEL.REF	*PVv_target*	Velocidad objetivo del movimiento
PVEL.INVERT	-	Variable booleana para la inversión del sentido de giro

Tabla 6-6: Variables MC_MOVEVELOCITY

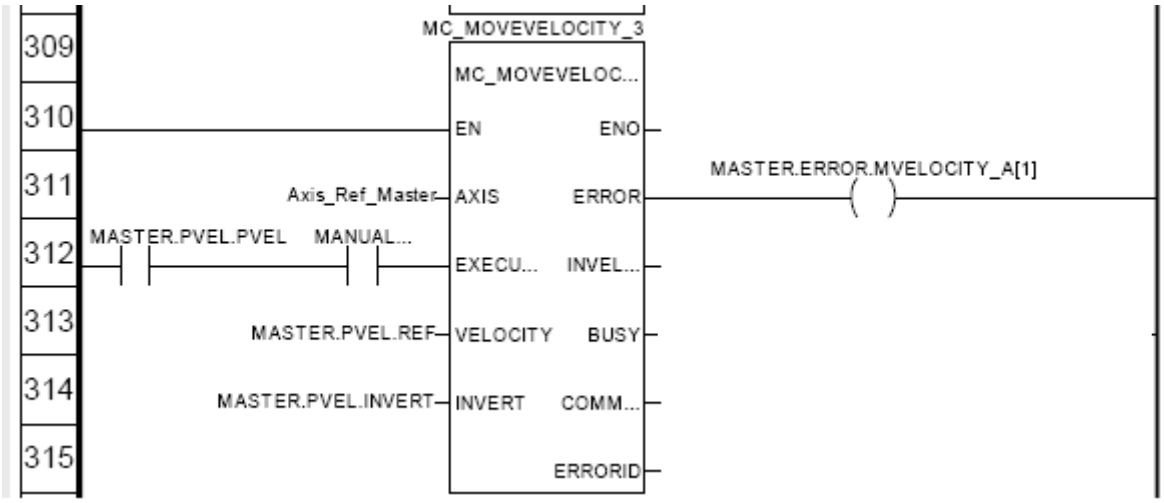


Figura 6-39: Implementación del Modo Profile Velocity en *Unity Pro*.

PROFILE TORQUE: Utilizando la función MC_TORQUECONTROL y proporcionando como entrada las variables *MASTER.TCONTROL* o *SLAVE.TCONTROL* según corresponda al servo *Maestro* o al servo *Esclavo* respectivamente.

Variable de entrada	Parámetros que modifica	Descripción
TCONTROL.ACTIVATE	*DCOMopmode* *PVv_reference*	Variable booleana de activación del modo Profile Torque
TCONTROL.TORQUE	*PTtq_target*	Par objetivo (en décimas de % frente al par de parada en continua (*_M_M_0*. Para nuestro servo drive es de 50Ncm)

Tabla 6-7: Variables MC_TORQUECONTROL

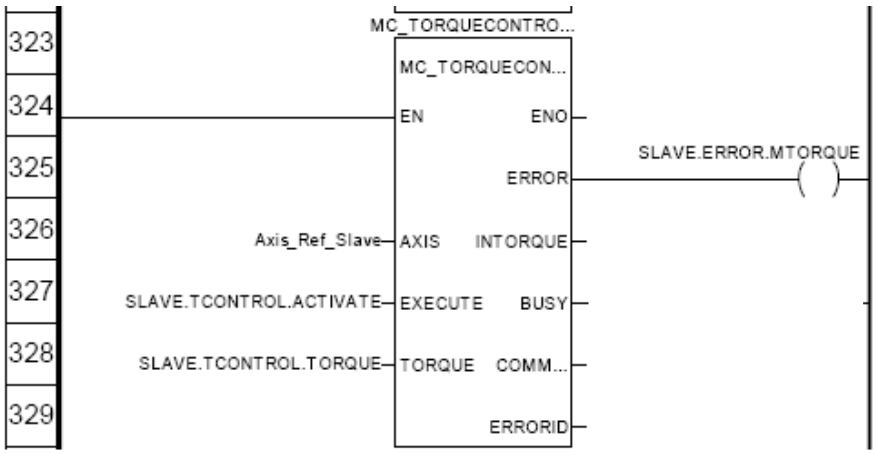


Figura 6-40: Implementación del Modo Profile Torque en *Unity Pro*

ELECTRONIC GEAR: Utilizando la función LXM_GEARPOSS y proporcionando como entrada las variables *MASTER.EGEAR* o *SLAVE.EGEAR* según corresponda al servo *Maestro* o al *Esclavo*, el PLC puede activar este modo de funcionamiento con su modo de sincronización en posición.
No existe una función específica de la librería *Motion Control* (en *Unity Pro*) para utilizar el modo de sincronización en velocidad por lo que aquí se implementa modificando manualmente los parámetros necesarios haciendo uso de la función MC_WRITEPARAMETER.

Variable de entrada	Parámetros que modifica	Descripción
EGEAR.GEAR	*DCOMopmode* *GEARreference*	Variable booleana de activación del modo Electronic Gear
EGEAR.NUM *EGEAR.NUMV*	*GEARnum*	Numerador de la relación de engranaje entre *Maestro* y *Esclavo*. (NUM para sincronización en posición y NUMV para sincronización en velocidad)
EGEAR.DENOM *EGEAR.DENOMV*	*GEARdenom*	Denominador de la relación de engranaje entre *Maestro* y *Esclavo*. (NUM para sincronización en posición y NUMV para sincronización en velocidad)
EGEAR.VEL	-	Variable booleana de elección entre el modo de sincronización en posición y en velocidad

Tabla 6-8: Variables para el modo Electronic Gear

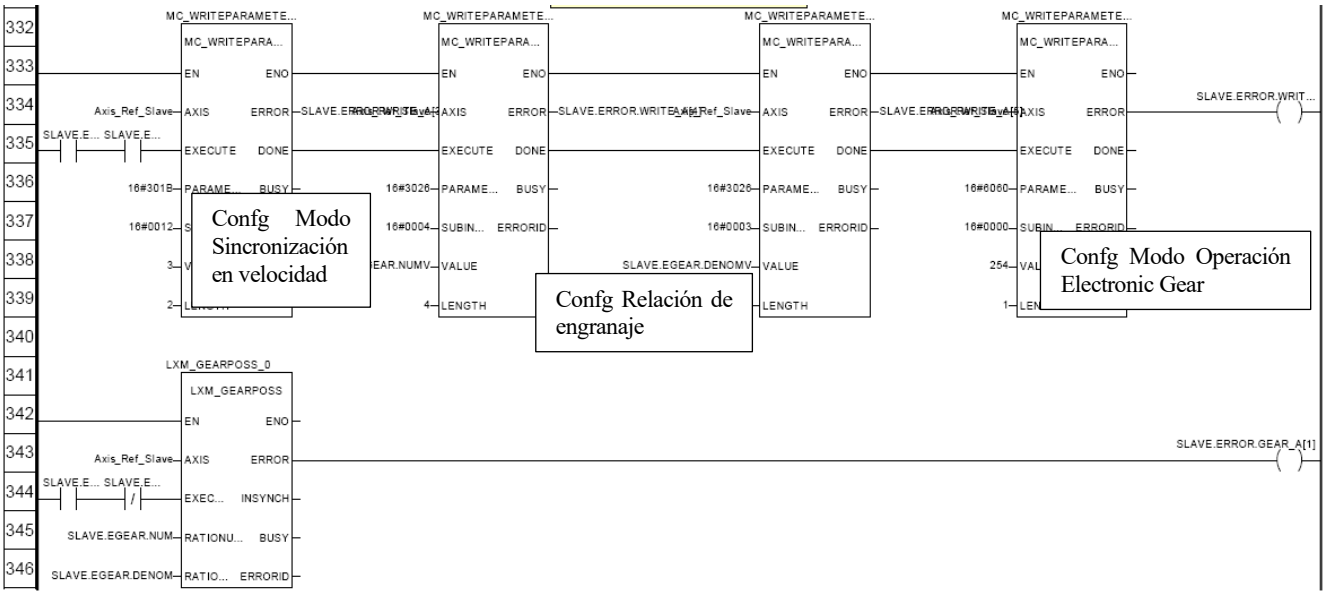


Figura 6-41: Implementación Modo Electronic Gear en *Unity Pro*.

6.5.2.2 Master Control

En esta sección LD se implementa el control del eje *Maestro* para el modo sincronismo. La variable *SYNC.MASTER_ESTADO* permite trabajar como una máquina de estados (Figura 6-42).

"HOMING" hace referencia a un movimiento de referenciado en el que la posición '0' se establece en bajo el interruptor de referencia. El "STOPPING" se realiza haciendo uso de la función MC_STOP y el "MOVE VELOCITY" se realiza haciendo uso de la función MC_MOVEVELOCITY.

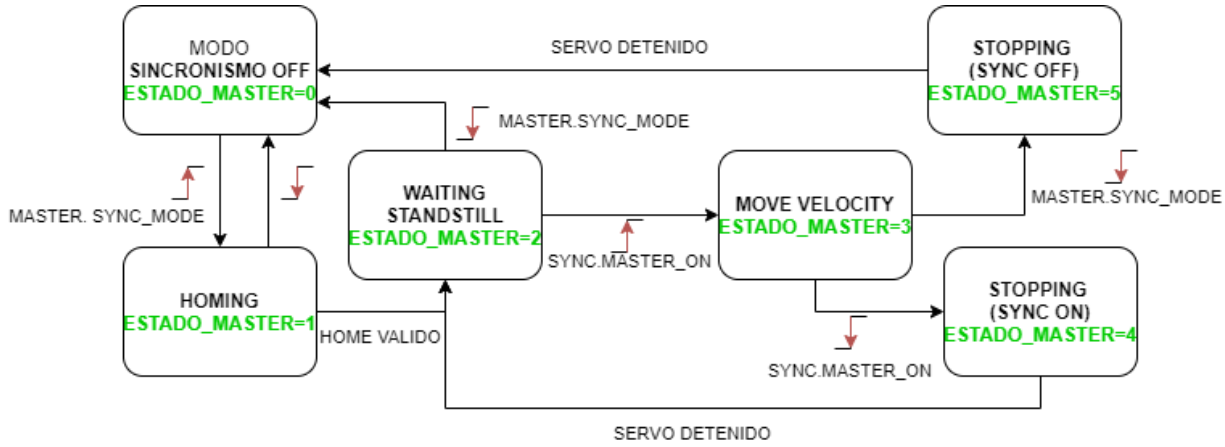


Figura 6-42: Máquina de estados para control del *Maestro* en modo sincronismo.

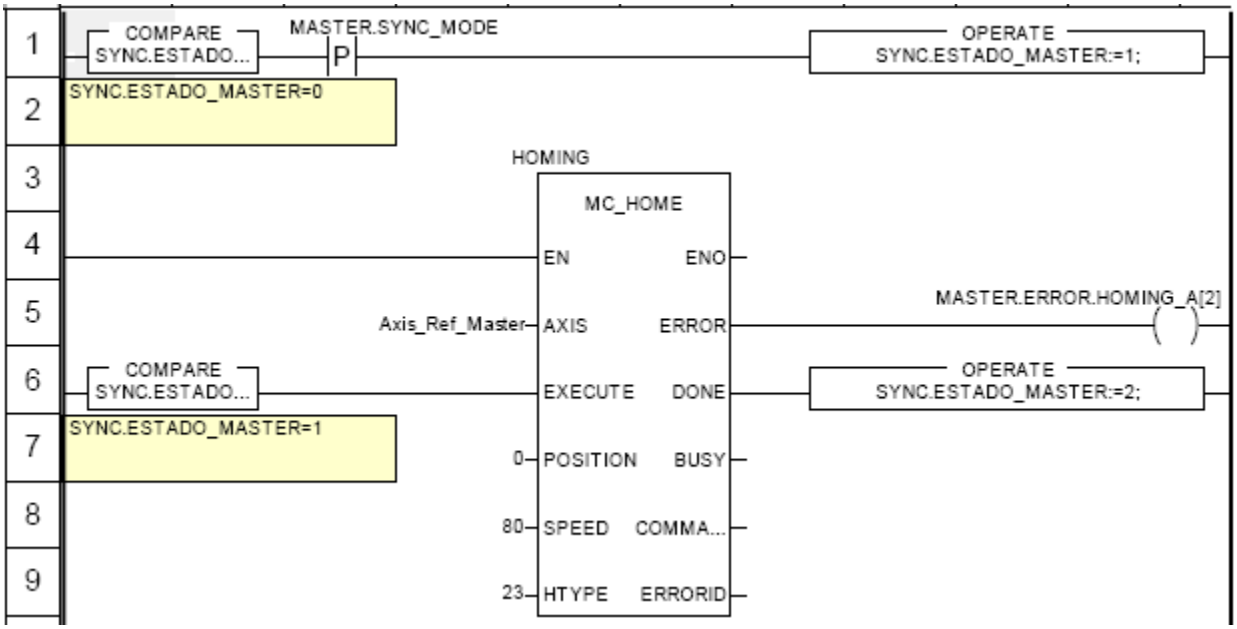


Figura 6-43: "HOMING" en el control del *Maestro.* Modo Sincronismo.

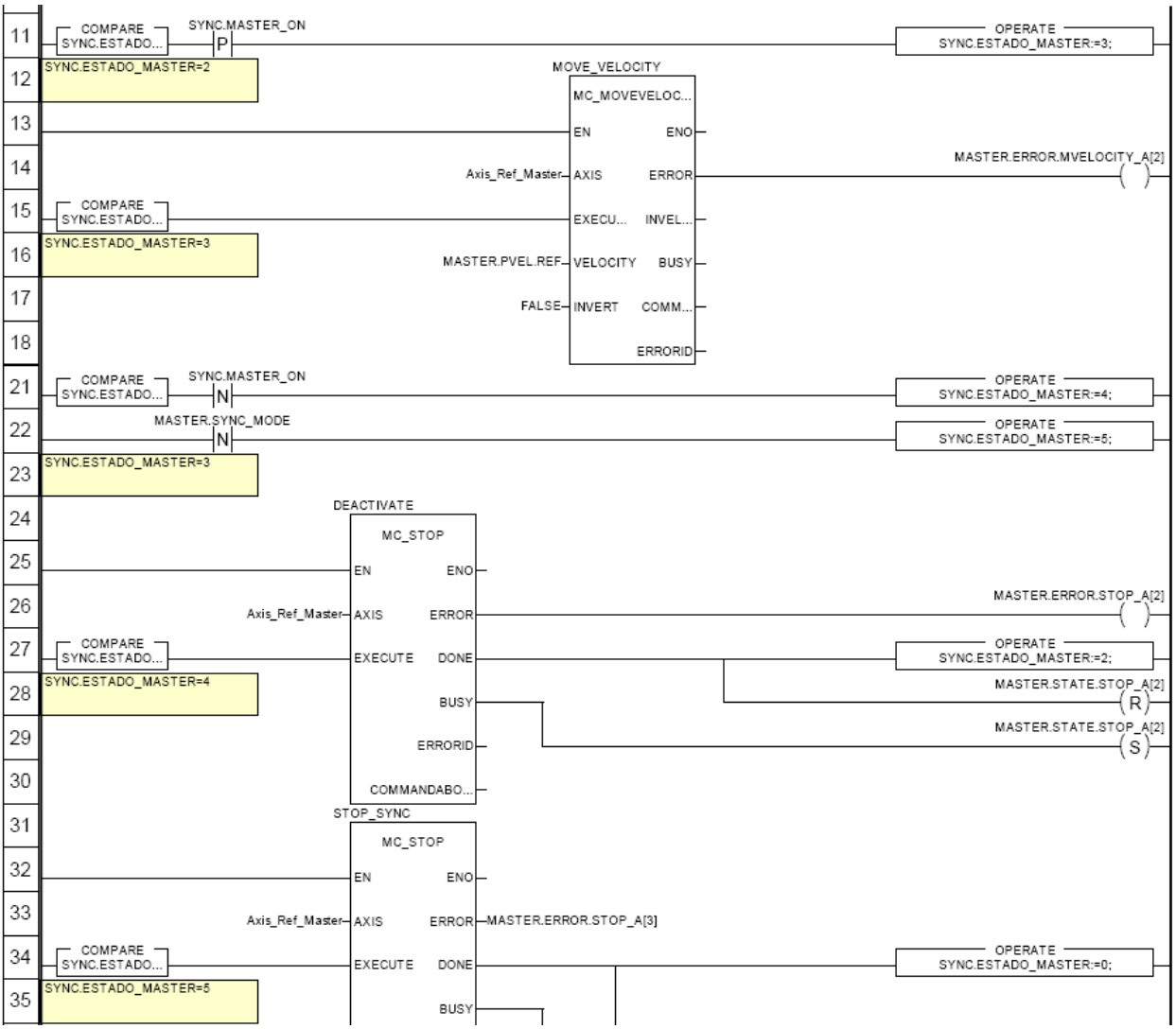


Figura 6-44: "MOVE VELOCITY" y "STOPPING" en el control del *Maestro*. Modo Sincronismo.

6.5.2.3 Slave Control Básico

En esta sección LD se implementa el control básico del eje *Esclavo* para el modo sincronismo. La variable *SYNC.ESTADO_SLAVE* permite trabajar como una máquina de estados (Figura 6-45).

"HOMING" hace referencia a un movimiento de referenciado en el que la posición '0' se establece en bajo el interruptor de referencia. El "STOPPING" se realiza haciendo uso de la función MC_STOP. Ambos estados se implementan de forma similar a como se hace en el control del *Maestro* (Figura 6-43 y Figura 6-44)

"ACTUALIZAR PERFIL DE VELOCIDAD" hace referencia al hecho de que, para garantizar una buena sincronización entre *Maestro* y *Esclavo*, es necesario configurar un perfil de aceleración/deceleración agresivo. Para ello se modifican los parámetros correspondientes haciendo uso de la función MC_WRITEPARAMETER (*RAMP_v_acc* y *RAMP_v_dec*).

"SYNCHRONISING" hace referencia al movimiento del *Esclavo* en búsqueda de igualar su posición con la del *Maestro*. La entrada "INVERT" de la función MC_MOVEVELOCITY, utilizada por este paso, se decide con unos bloques comparadores que utilizan la posición del *Maestro*, la del *Esclavo* y el valor del módulo configurado (*MOD_max*) para determinar el camino más corto para conseguir la sincronización (en el sentido positivo o negativo de giro).

El estado 6 hace referencia al movimiento del *Esclavo* una vez ambos ejes están sincronizados (la diferencia entre sus posiciones actuales es menor que la holgura configurada en la variable *SYNC.HOLGURA*). Mediante la variable *SYNC.GEAR_VEL* se decide si la sección empleará el modo de sincronización en velocidad o en posición. Se implementa de forma similar a como se realiza para el control manual (Figura 6-41)

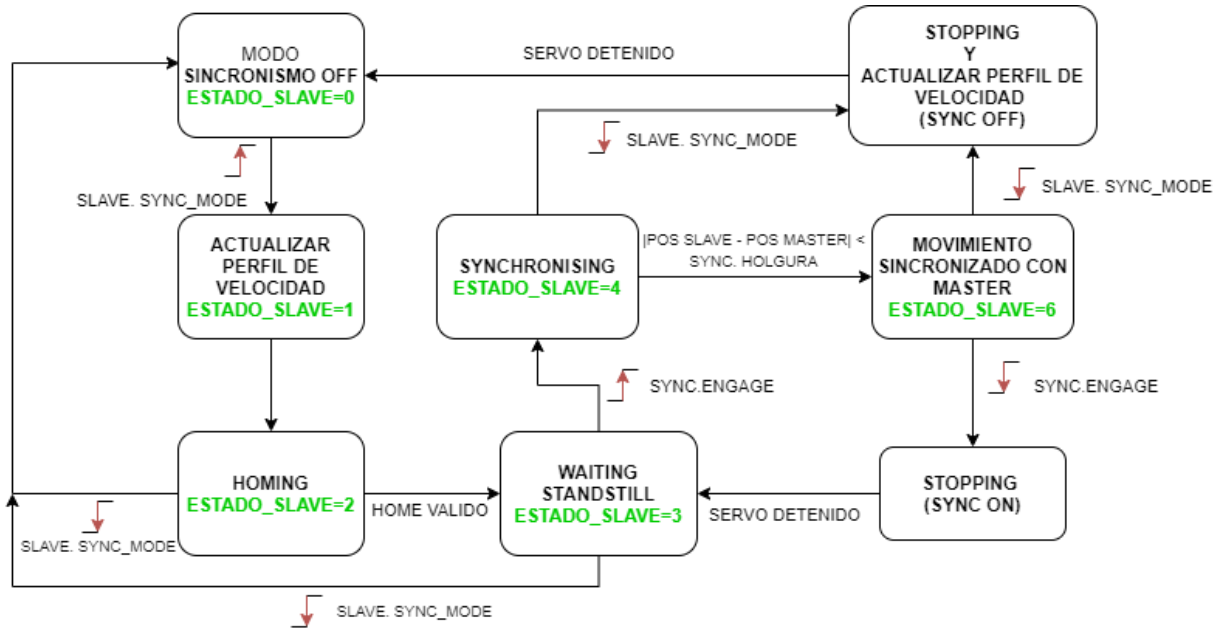


Figura 6-45: Máquina de estados para control básico del *Esclavo* en modo sincronismo.

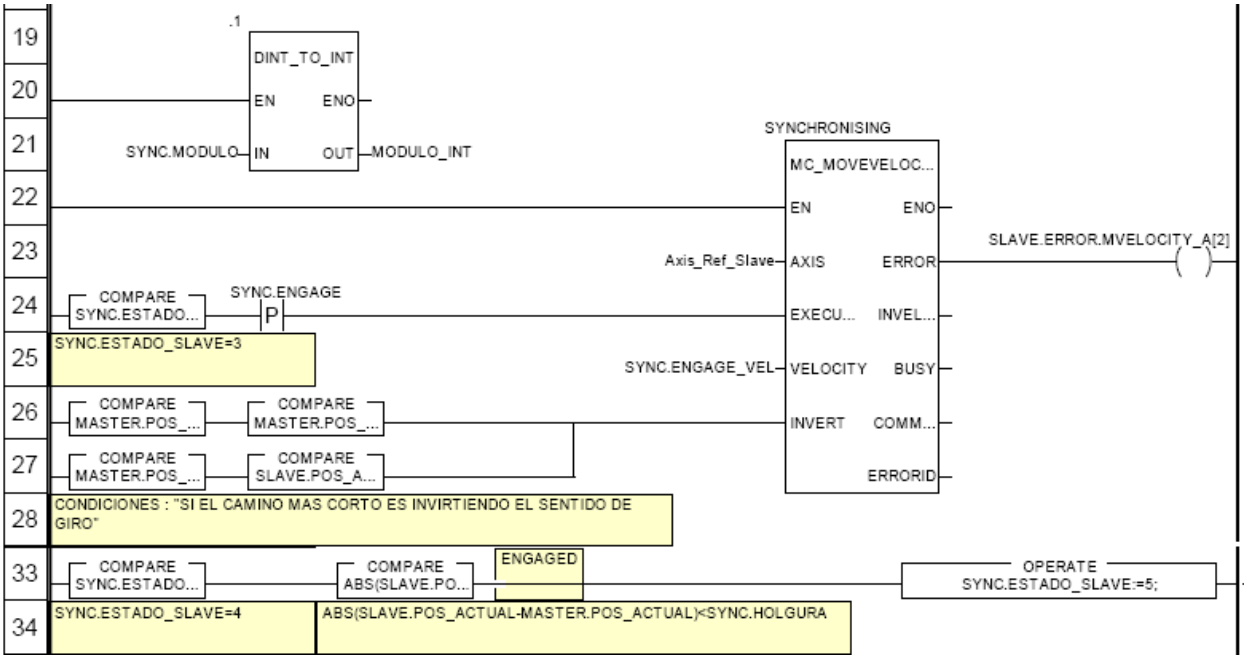


Figura 6-46: "SYNCHRONISING" (con comprobación de holgura) en control básico del *Esclavo.*

6.5.2.4 Slave Control Avanzado

En esta sección LD se implementa el control avanzado del eje *Esclavo* para el modo sincronismo. La variable *SYNC.ESTADO_SLAVE* permite trabajar como una máquina de estados ().

"HOMING", "STOPPING", "ACTUALIZAR PERFIL DE VELOCIDAD" se implementa de la misma forma que en el control básico (apartado 6.5.2.3)

Tal como se menciona en la descripción funcional (apartado 6.4) la sincronización se realiza de modo que el indicador visual del *Esclavo* esté siempre sincronizado con el indicador del *Maestro* que esté en la parte inferior de la correa. Por ello, es importante tener localizado sus dos indicadores en todo momento. La sección calcula las variables *SYNC.POS_1* (indicador metálico) y *SYNC.POS_2* (indicador no metálico) a partir de la posición medida por el encoder del eje *Maestro* (*MASTER.POS_ACTUAL*). La variable *SYNC.ENGAGE_SELECTOR* determina cuál de los dos indicadores del *Maestro* está en la parte inferior.

SYNCHRONISING hace referencia al movimiento del *Esclavo* en búsqueda de igualar su posición con la del indicador del *Maestro* que esté en la parte inferior. La entrada "INVERT" de la función MC_MOVEVELOCITY, utilizada por este paso, se decide con unos bloques comparadores que determinan el sentido de giro en función del camino más corto para conseguir la sincronización.

ELECTRONIC GEAR hace referencia al movimiento del *Esclavo* una vez ambos ejes están sincronizados. Mediante la variable *SYNC.GEAR_VEL* se decide si la sección empleará el modo de sincronización en velocidad o en posición (por defecto, en posición). Se implementa de forma similar a como se realiza para el control manual (Figura 6-41)

Estando sincronizados, cuando el indicador del *Esclavo* se sale de la mitad inferior de su correa indica que el indicador del *Maestro* que se encuentra en la parte inferior ha cambiado y por lo tanto el *Esclavo* debe resincronizarse (transición destacada en amarillo en)

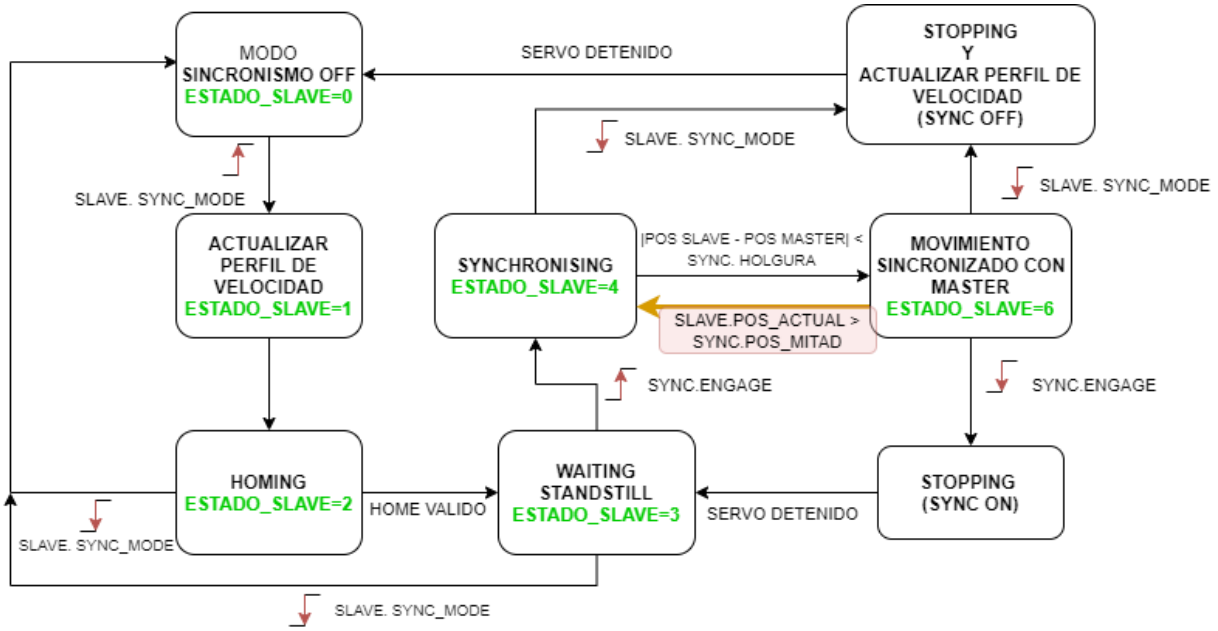


Figura 6-47: Máquina de estados para control avanzado del *Esclavo* en modo sincronismo.

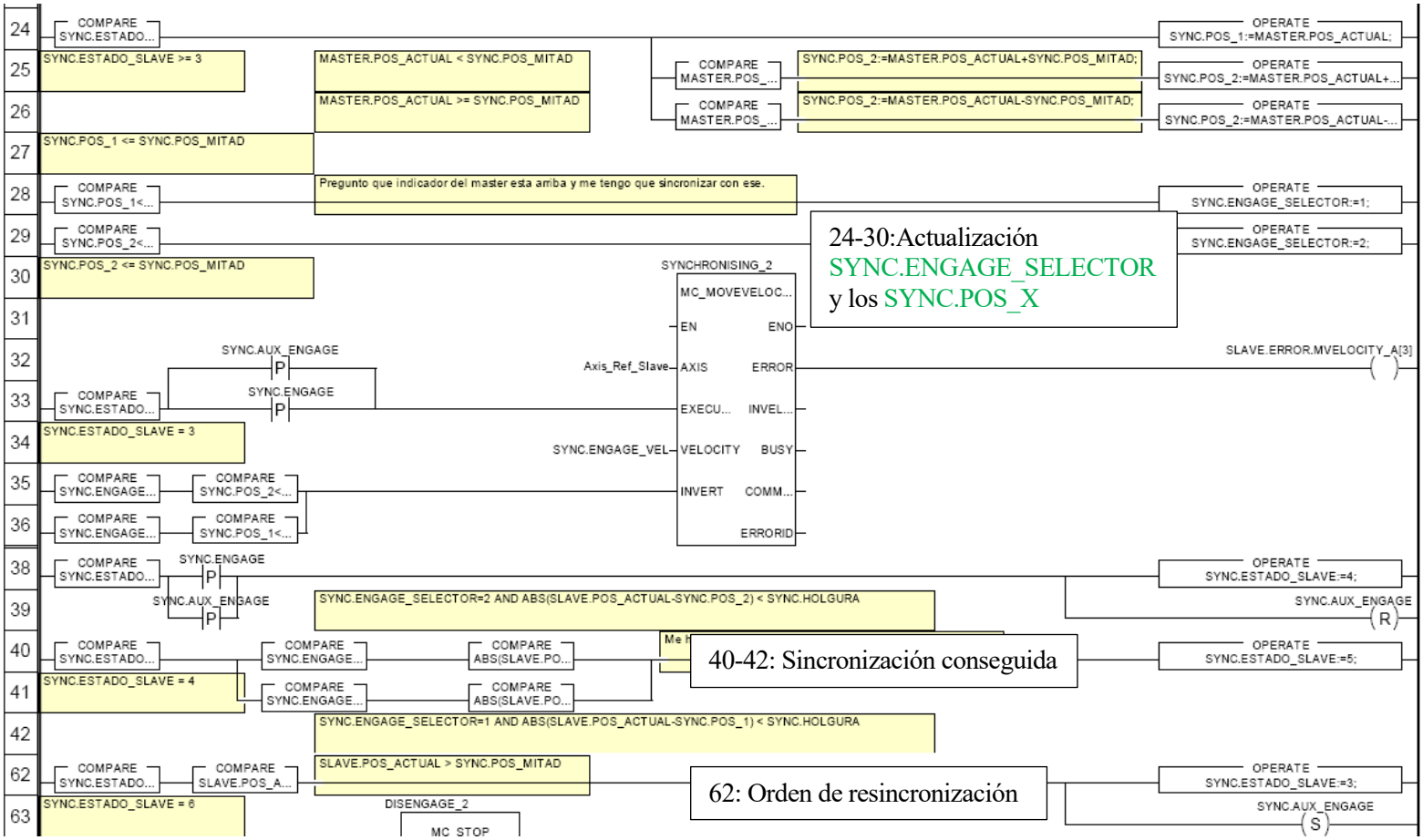


Figura 6-48: "SYNCHRONISING" (con comprobación de holgura, actualización de SYNC.ENGAGE_SELECTOR y los SYNC.POS_X y resincronización) en control avanzado del *Esclavo.*

6.6 Supervisor

Para facilitar el manejo de todas las funcionalidades de las que dispone la aplicación descrita anteriormente, se pone a disposición del usuario una pantalla de operador implementada sobre el mismo proyecto de *Unity Pro*.

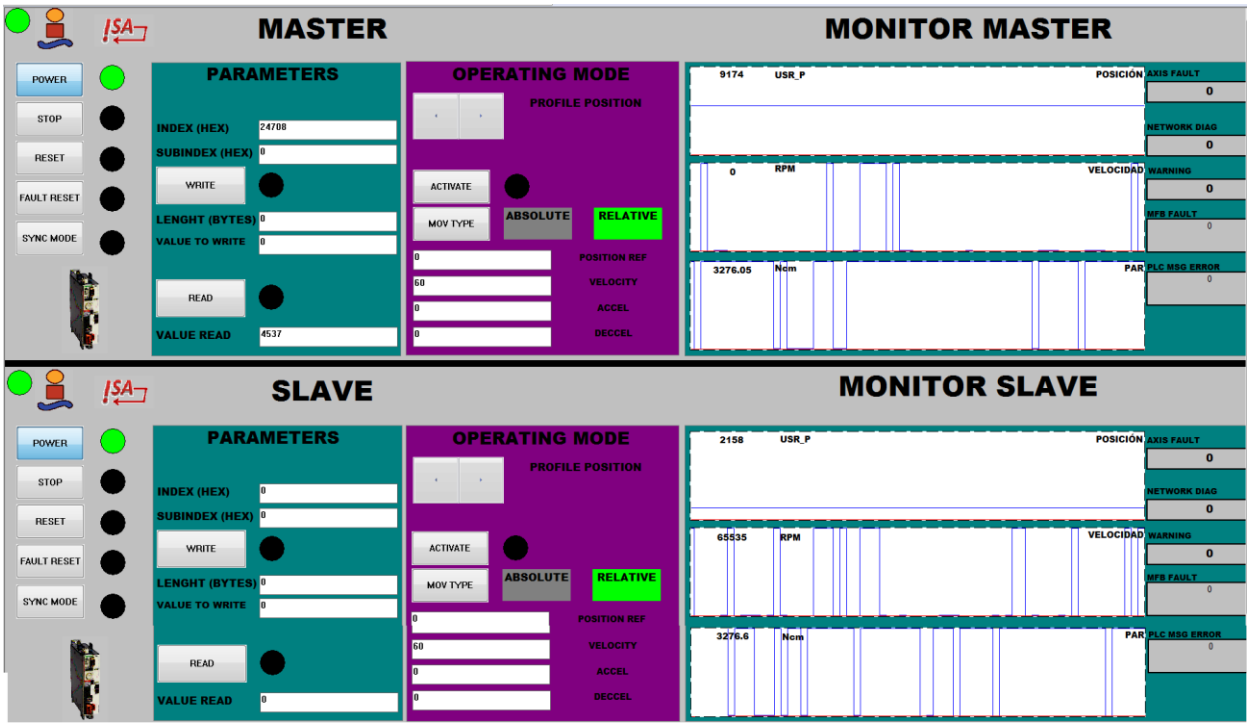


Figura 6-49: Pantalla de Operador para el manejo del sistema.

Para la generación de dicha pantalla se declara una estructura de datos auxiliar denominada "SUPERVISOR" así como una sección LD ("Supervisor_Aux") para la generación de señales de visualización (leds de estado, errores…). En particular, para la generación de las señales de los leds de estado se hace uso de la función MC_READSTATUS que nos permite conocer si el eje está detenido, si tiene activo algún modo de funcionamiento, si existe un punto de referencia (HOME) válido, si la etapa de potencia está activada o si existe algún error que haya detenido el movimiento en curso. Cada llamada a la función genera una estructura de variables con toda la información anterior, *STATUS_MASTER* y *STATUS_SLAVE*.

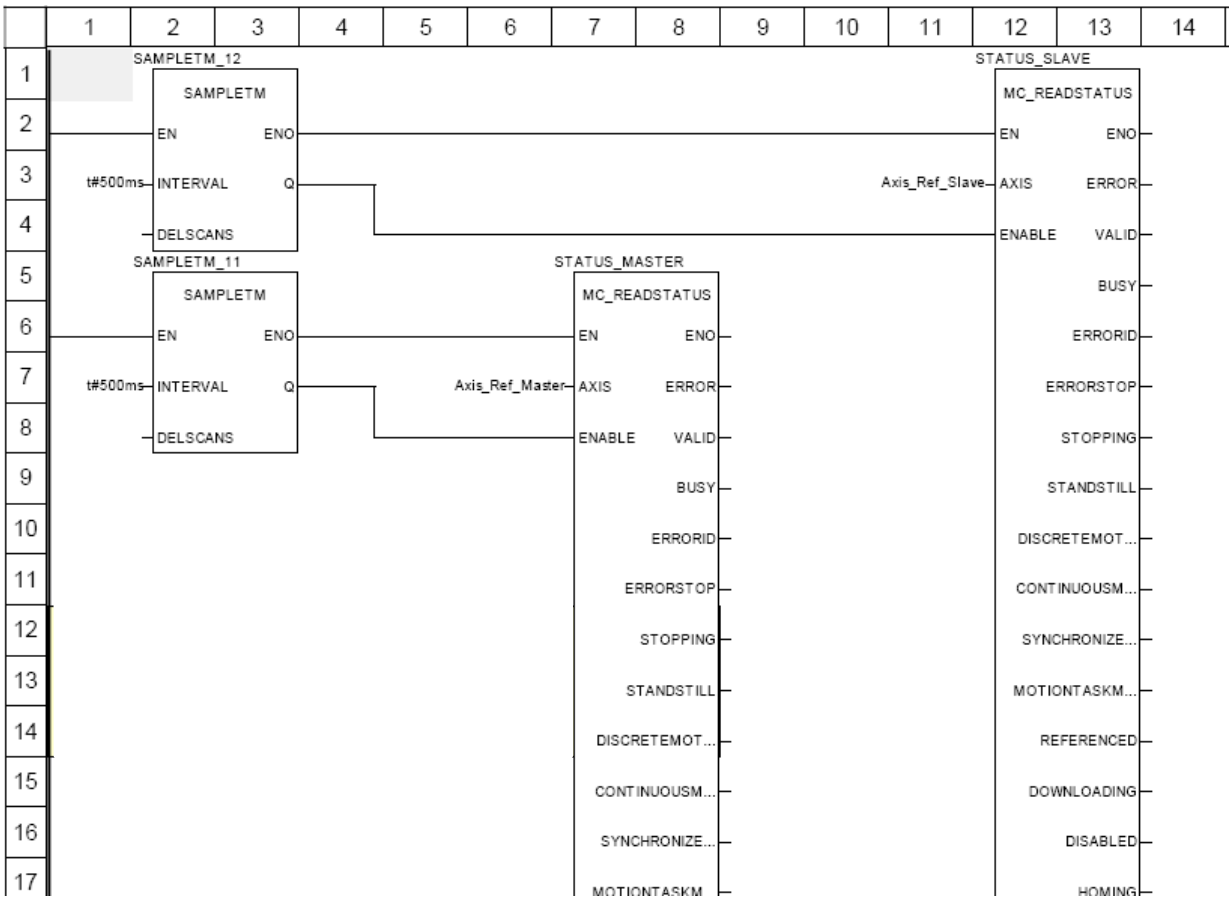


Figura 6-50: Función MC_READSTATUS para consultar información de los servos.

La pantalla se divide horizontalmente en dos mitades: una para el eje que actúa como *Maestro* y otra para el eje que actúa como *Esclavo*. Ambas mitades cuentan con una serie de elementos que pueden agruparse en las 4 secciones descritas a continuación.

6.6.1 Controles básicos

Esta sección comprende los elementos mostrados en la Figura 6-51 que son idénticos para ambos servos.

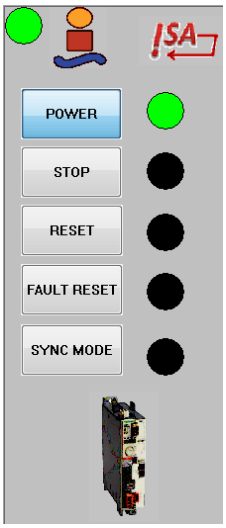


Figura 6-51: Sección de controles básicos. Pantalla de operador.

1. **Led estado comunicación CANOpen.** Iluminado en verde indica que la comunicación entre el PLC y el servo, a través del bus CANOpen, es correcta. Cabe recordar que esta información es proporcionada por la función *CAN_HANDLER* (Variable de salida *MASTER/SLAVE.AXIS_OK*).

2. **Botón y led de encendido.** Permite la activación de la etapa de potencia. Iluminado en verde, el led indica que la etapa de potencia está activa.

3. **Botón y led de detención.** Permite la detención del movimiento activo. Iluminado en verde, el led indica que la detención se ha realizado correctamente.

4. **Botones y leds de reinicio.** Permite el reinicio del servo tras un error (T15 en Figura 2-6). Con "Reset" se realiza a través de la función *MC_RESET*. Con "Fault Reset" se realiza mediante la modificación de los bits de la CONTROL_WORD (Tabla 6-3). Iluminados en verde, los leds indican que el reinicio de error se ha realizado correctamente.
   "Fault Reset" solo funcionará para el reinicio de errores del servodrive (FLT en HMI) y al accionarlo el equipo se ira al estado "Ready to Switch On". "Reset" reiniciará el funcionamiento del equipo, en cualquier caso (p. ej STOP en HMI) y volverá al estado previo al error.

5. **Botón y led del modo sincronismo.** Permite el cambio del modo manual al modo sincronismo. Iluminado en verde, el led indica que el modo sincronismo está activo.

6.6.2 Manejo de parámetros

Esta sección comprende los elementos necesarios para el manejo manual de los parámetros del servo drive.

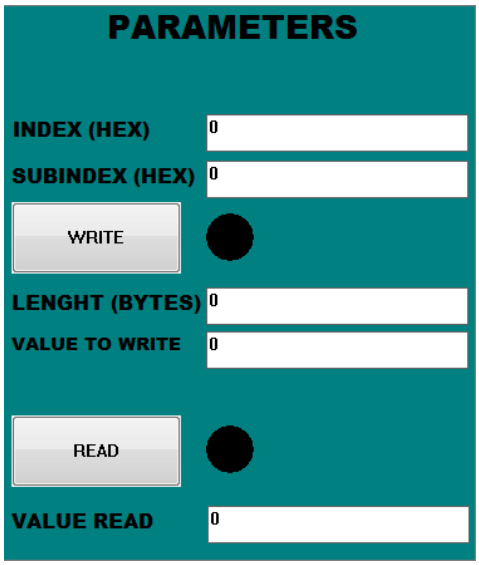


Figura 6-52: Sección de manejo manual de parámetros. Pantalla de operador.

1. **Dirección.** Campos de entrada donde deben indicarse tanto el índice como el subíndice del parámetro a leer o escribir. Deben introducirse en Hexadecimal (16#XXXX).

2. **Escritura.** Incluye: los campos de entrada para indicar la longitud (en bytes) del parámetro y el valor numérico a escribir, el botón de comando para ordenar la escritura y un led que se ilumina en rojo si ocurre algún error en el proceso de escritura.

3. **Lectura.** Incluye: el botón de comando para ordenar la lectura, el campo donde se muestra el valor leído y un led que se ilumina en rojo si ocurre algún error en el proceso de lectura.

6.6.3 Manejo de los modos de operación

Esta sección comprende los elementos necesarios para el manejo de los distintos modos de operación:

- **Modo Manual:** Profile Velocity, Profile Position, Profile Torque, Electronic Gear y Homing.
- **Modo Sincronismo**

En la parte superior se incluye un selector que permite desplazarse por las distintas subpantallas existentes (una por cada modo de operación).

A continuación, se describen los elementos de cada subpantalla:

Profile Position

1. **Botón de activación.** Permite ejecutar el movimiento.

2. **Led de estado.** Muestra el estado del modo de operación.

Color	Estado
Negro	Desactivado
Naranja	Stand-by (activado, pero sin movimiento)
Verde	Activado y en movimiento
Rojo	Error

Tabla 6-9: Pantalla de operador. Led de estado modo Profile Position

3. **Botón de selección de tipo (MOV TYPE).** Permite conmutar entre el modo de movimiento absoluto y relativo.

4. **Campos de entrada.** Permite configurar directamente los parámetros esenciales del modo de operación:
   a. **Position Ref.** Posición de referencia (`PPp_target`)
   b. **Velocity.** Velocidad objetivo de movimiento (`PPv_target`)
   c. **Accel/Deccel.** Aceleración y deceleración en el perfil de velocidad del movimiento. (`RAMP_v_acc/_dec`)

   Aparecen inicialmente con los valores establecidos por defecto.

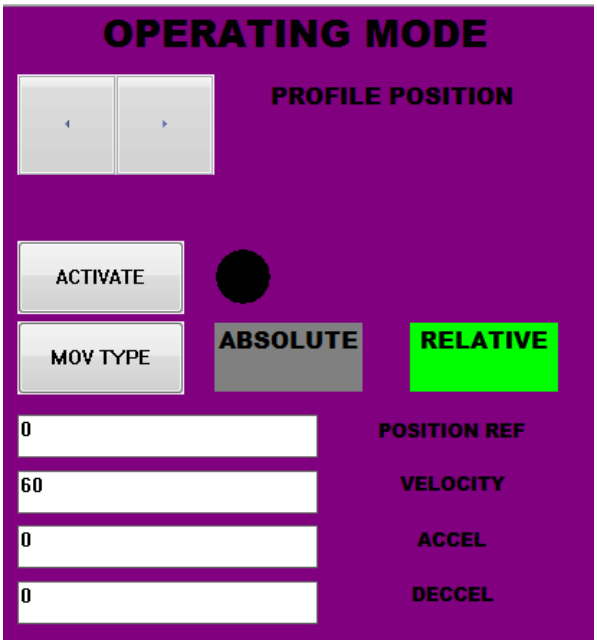


Figura 6-53: Sección manejo modos de operación. Modo Profile Position.

Profile Velocity

1. **Botón de activación.** Permite ejecutar el movimiento.

2. **Led de estado.** Muestra el estado del modo de operación (Tabla 6-9).

3. **Botón de dirección.** Permite seleccionar la dirección de movimiento.

4. **Velocity Ref.** Campo de entrada que permite configurar la velocidad objetivo (`PVv_target`).

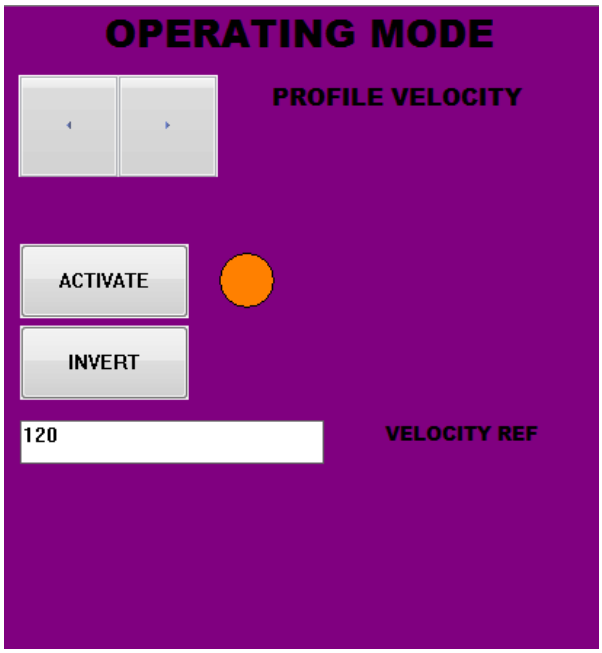


Figura 6-54: Sección manejo modos de operación. Modo Profile Velocity

Profile Torque

1. **Botón de activación.** Permite ejecutar el movimiento.

2. **Led de estado.** Muestra el estado del modo de operación (Tabla 6-9).

3. **Torque Ref.** Campo de entrada que permite configurar el par objetivo (*PTtq_target*).

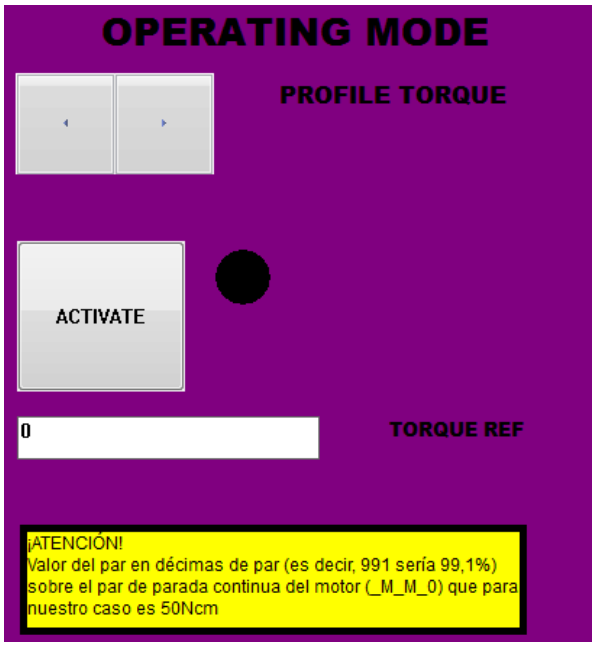


Figura 6-55: Sección manejo modos de operación. Modo Profile Torque

Electronic Gear

1. **Botón de activación.** Permite ejecutar el movimiento.

2. **Led de estado.** Muestra el estado del modo de operación (Tabla 6-10)

Color	Estado
Negro	Desactivado
Verde	Activado
Rojo	Error

Tabla 6-10: Pantalla de operador. Led de estado modo Electronic Gear

3. **Botón de tipo de sincronización.** Permite escoger entre el modo de sincronización de posición y el modo de sincronización de velocidad.

4. **Campos de entrada.** Permite configurar la relación (numerador/denominador) entre el giro del *Maestro* y el giro del *Esclavo* (`GEARnum` y `GEARdenom`)

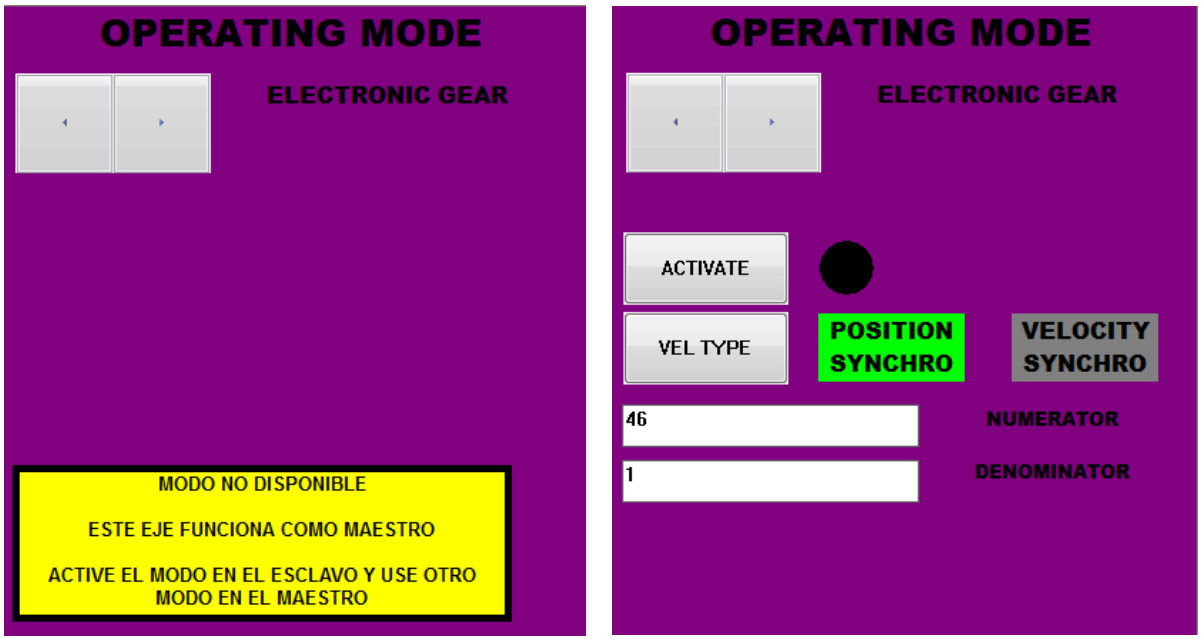


Figura 6-56: Sección manejo modos de operación. Modo Electronic Gear en eje *Maestro* (izquierda) y en eje *Esclavo* (derecha)

Homing

1. **Botón de activación.** Permite ejecutar el movimiento.

2. **Led de estado.** Muestra el estado del modo de operación (Tabla 6-11)

Color	Estado
Negro	Desactivado
Naranja	Homing en progreso
Verde	Punto de referencia (HOME) válido
Rojo	Error

Tabla 6-11: Pantalla de operador. Led de estado modo Homing

3. **Campos de entrada.** Permite configurar directamente los parámetros esenciales del modo de operación:
    a. **Home Type:** Tipo de Homing (*HMmethod*)
    b. **Velocity:** Velocidad de movimiento (*HMv*)
    c. **Referenced Position:** Valor de posición real otorgado a la posición mecánica final alcanzada con el movimiento (*HMp_home*)

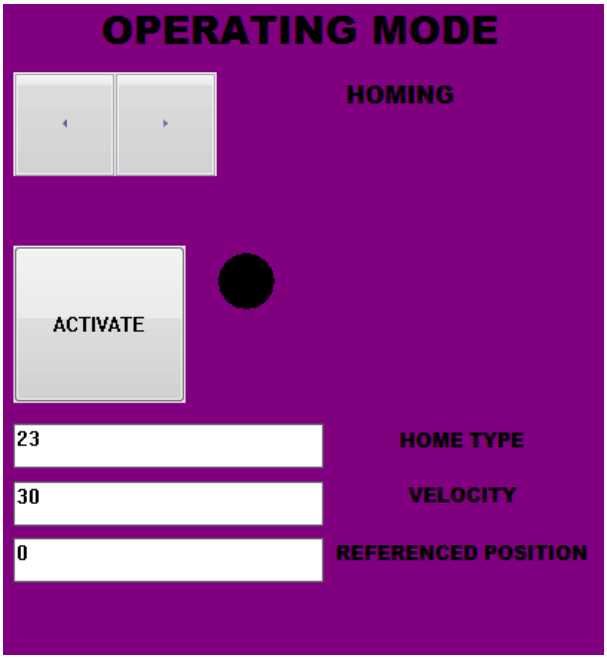


Figura 6-57: Sección manejo modos de operación. Modo Homing.

Modo sincronismo. *Maestro*

1. **Botón de activación.** Permite iniciar y mantener el movimiento del *Maestro* contemplado para este caso (modo Profile Velocity)

2. **Indicador de estado.** Tal como se menciona en el apartado 6.5.2.2 el comportamiento del *Maestro* se implementa como una máquina de estados. Aquí se indican los diferentes estados por los que va pasando (*SYNC.ESTADO_MASTER*).

Modo sincronismo. *Esclavo*

1. **Botón de activación.** Permite iniciar y mantener la sincronización del eje *Esclavo* con el eje *Maestro*.

2. **Led de estado.** Iluminado en verde indica que el movimiento del *Maestro* y del *Esclavo* están sincronizados.

3. **Indicador de estado.** El comportamiento del *Esclavo* también se implementa como una máquina de estados. Aquí se indican los diferentes estados por los que va pasando (*SYNC.ESTADO_SLAVE*).

4. **Botón modo avanzado.** Permite conmutar entre los modos de sincronización básico y avanzado descritos en los apartados 6.5.2.3 y 6.5.2.4.

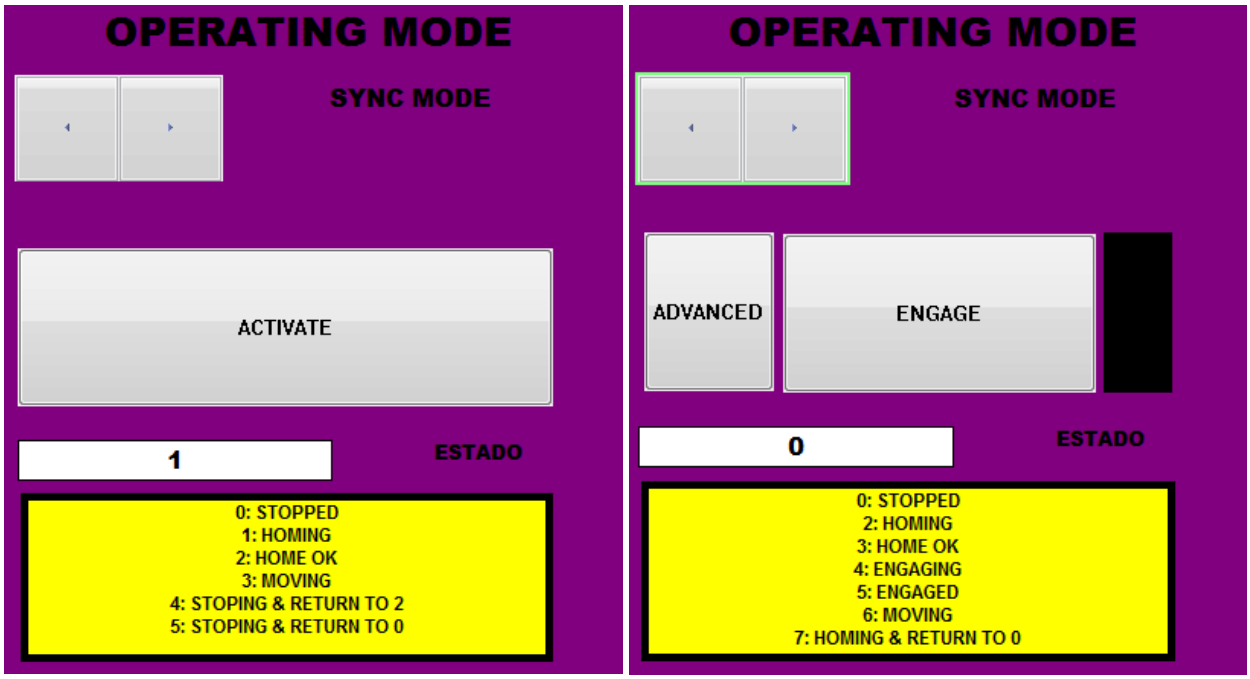


Figura 6-58: Sección manejo modos de operación. Modo Sincronismo en eje *Maestro* (izquierda) y en eje *Esclavo* (derecha)

6.6.4 Monitorización del servo

Esta sección comprende los elementos necesarios para la monitorización de la posición, velocidad y par aplicado al servomotor, así como para la identificación de errores.

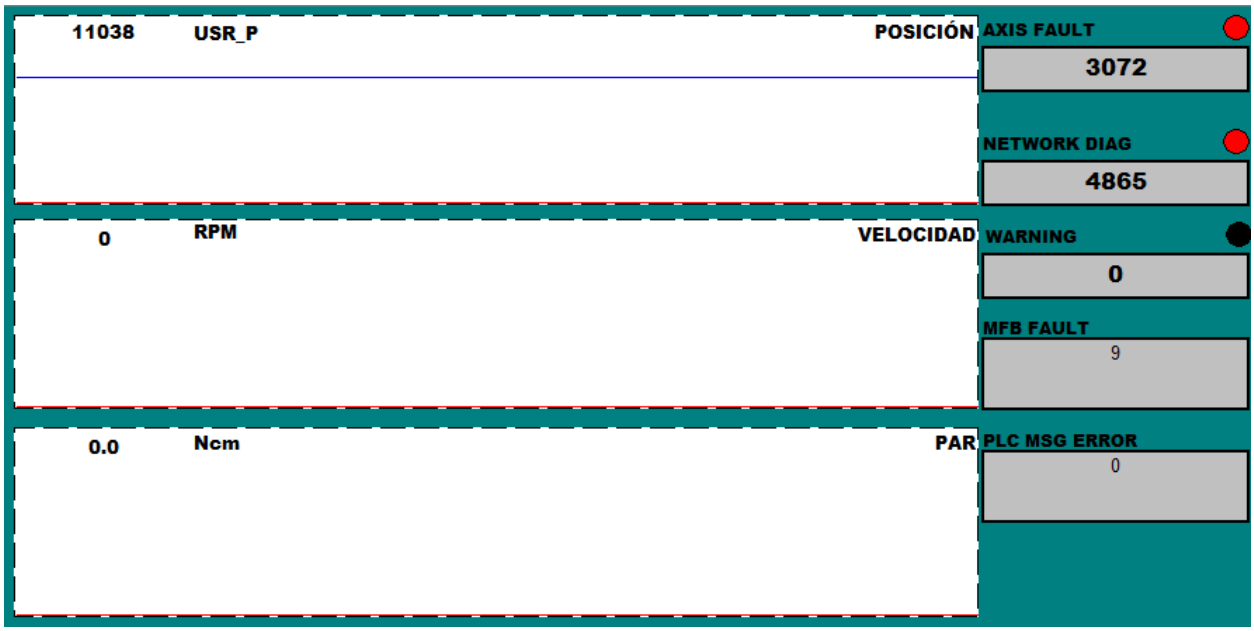


Figura 6-59: Sección monitorización del eje.

1. **Monitorización de la posición**. En esta subpantalla se mostrará: en azul la posición del servomotor en todo momento y en rojo la posición de referencia para el modo de operación activo.
    a. Para el *Maestro*, solo aparecerá la posición de referencia en el modo Profile Position.
    b. Para el *Esclavo*, además de en el modo Profile Position, también aparecerá como posición de referencia, la posición con la que debe sincronizarse en el modo sincronismo:
        i. Modo Básico: Posición de referencia = Posición actual del *Maestro*
        ii. Modo Avanzado: Posición de referencia = Posición del indicador que se encuentre en la parte inferior de la correa.

2. **Monitorización de la velocidad.** En esta subpantalla se mostrará: en azul la velocidad del servomotor en todo momento y en rojo la velocidad de referencia cuando está activo el modo de operación Profile Velocity.

3. **Monitorización del par aplicado.** En esta subpantalla se mostrará: en azul el par aplicado al servomotor en todo momento y en rojo el par de referencia cuando está activo el modo de operación Profile Torque.

4. **Identificación de errores.** Para la identificación de errores del sistema se utiliza la función *MC_READAXISERROR*. Cuando cualquiera de las funciones utilizadas de la librería *Motion Control* reporta un error (*MASTER.ERROR.XXX* / *SLAVE.ERROR.XXX*), la función *MC_READAXISERROR* identifica si se trata de un fallo o advertencia del eje, un fallo en la comunicación o un fallo del propio bloque funcional (*MC_XXXX*).
   Para la interpretación de los códigos de error consultar el parámetro `_SigLatched` para el "Axis Fault" (pasándolo de decimal a binario), `_WarnLatched` para el "Warning" (pasándolo de decimal a binario), Appendix A de [11] para "MFB Fault" y apartado 9.4 de [5] para "Network Diag" (pasándolo de decimal a hexadecimal).

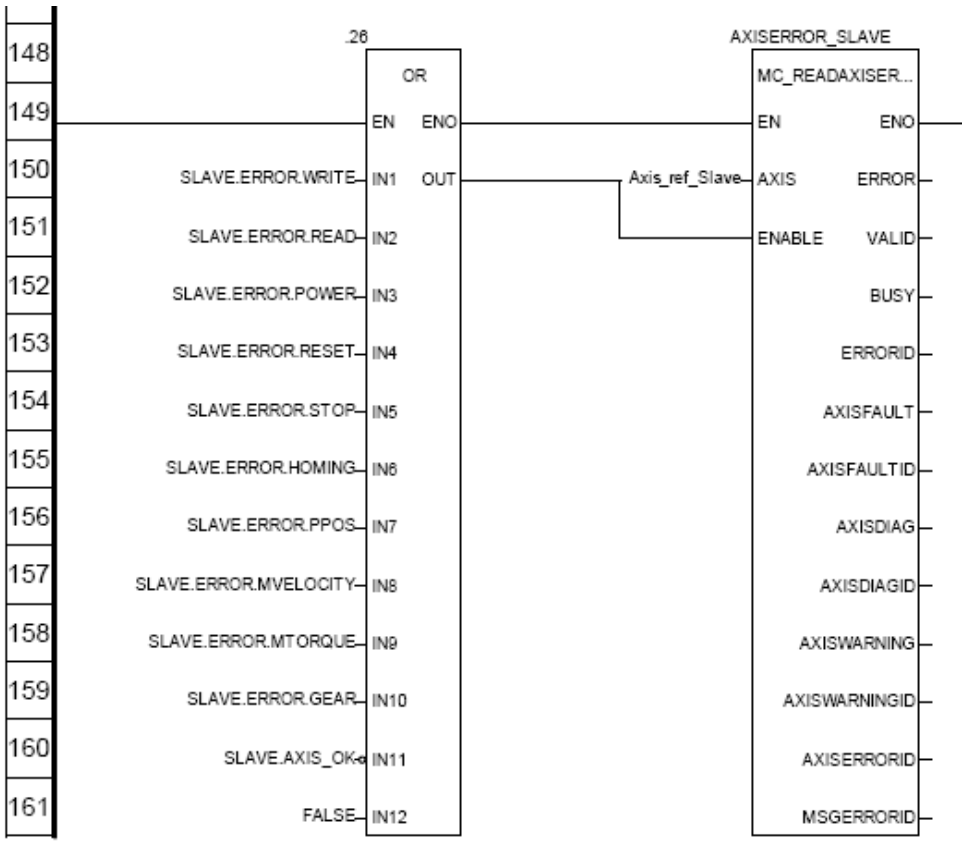


Figura 6-60: Identificación de errores en el eje *Esclavo* con MC_READAXISERROR

7 TRABAJO FUTURO

En este capítulo se proponen una serie de ideas que permitan:

- Desarrollar conceptos interesantes pero poco tratados durante el presente proyecto que permitan profundizar en el manejo, puesta en marcha e integración de servomotores.
- Complementar los conocimientos adquiridos.

El listado que procede a continuación busca cubrir el primero de los puntos mientras que los apartados 7.1 y 7.2 cubren el segundo.

- En el apartado 6.3 , se describían tres alternativas para **llevar a cabo la configuración de los servo drives desde su controlador remoto en lugar de utilizar el software de puesta en marcha**. Partiendo del proyecto en *Unity Pro* adjunto a este documento, se propone realizar la parametrización inicial descrita en 6.3.1 a partir de las dos últimas alternativas mencionadas:
  - Desde la ventana de configuración de cada servo drive.
  - Mediante la utilización de las funciones TE_UPLOADDRIVEPARAM y TE_DOWNLOADDRIVEPARAM (apartado 6.3.3).
- En el apartado 2.11.2 se describían los diferentes métodos de ajuste para el regulador del servo drive. Una práctica interesante podría ser **ajustar el regulador por los tres métodos distintos: Manual, Semiautomático y Automático**. De esa forma, también se conseguiría profundizar en el significado de todos los parámetros disponibles del regulador y no solo en los básicos que se mencionan en dicho apartado.
- Con el sistema planteado en el capítulo 1 y siguiendo los pasos allí descritos, **programar en *SOMachine* la aplicación propuesta en el apartado 6.4**. Para la depuración de la aplicación además de lo descrito en el apartado 5.4, también podría plantearse simular una HMI con el software *Vijeo Designer* [18].
- Para completar los conocimientos relativos a los modos de funcionamiento disponibles en el servo drive, podrían plantearse un par de prácticas en relación con los dos modos de funcionamiento no implementados en la aplicación del capítulo 1:
  - **Implementar el modo Interpolated position** para que uno de los dos servo pase por una serie de posiciones determinadas generando una trayectoria interpolada. Para ello será necesario que el PLC genere la señal SYNC con el periodo adecuado (Figura 7-1).
  - **Implementar un ejemplo de aplicación del modo Motion Sequence.** Se podría probar a implementar sobre un servo, haciendo uso de *SoMove*, el ejemplo propuesto en [9].
- **Ampliar los conocimientos en el diseño e implementación de perfiles CAM** (apartado 3.1.1) haciendo uso de la serie de videos explicativos propuestos en [19]. Si se dispone de un controlador de movimiento y servo drive compatible con estos perfiles, podría tratar de implementarse algunas de las aplicaciones propuestas en dichos videos (p. ej "Alimentación Sincrona de producto mediante usillo") haciendo uso de *SoMachine* y la librería *Motion Control* (MC_CamIn, MC_CamOut, MC_CamTableSelect…).

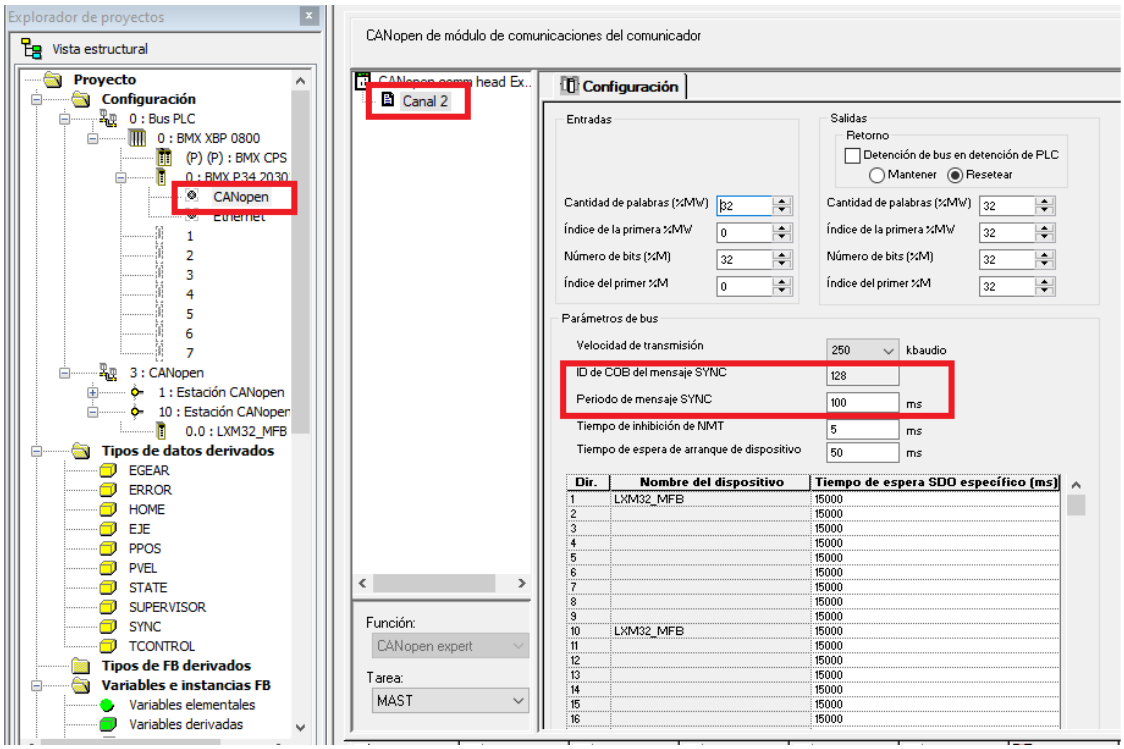


Figura 7-1: Configuración del periodo del mensaje SYNC desde e M340.

7.1 Aplicación a otros fabricantes comerciales

Aunque durante el desarrollo del proyecto se han mencionado únicamente productos de *Schneider Electric*, existen otros grandes suministradores de soluciones para aplicaciones de control de movimiento. En este apartado se ilustra como todo lo desarrollado en este documento puede trasladarse a otros fabricantes comerciales como *Siemens*.

Se aconseja revisar la información que se expone para hacer ese ejercicio de extrapolar todo lo visto con productos de *Schneider Electric* a productos de *Siemens* observando las diferencias y semejanzas.

7.1.1 Servo accionamientos

Siemens dispone de una familia de Servo drives (p. ej SINAMICS S210[15]) que pueden combinarse con sus servo motores (p. ej SIMOTICS S-1FK2). Sin embargo, en su caso, están previstos para su uso con un controlador remoto de nivel superior (PLC, p. ej SIMATICS S7-1500) conectados via PROFINET (Figura 7-2).

En [20] puede consultarse toda la información relativa al servo drive SINAMICS S210 (lista de parámetros, puesta en marcha, funciones de seguridad integradas, etc.)

7.1.2 Librería *Motion Control*

Para la programación del control de movimiento de los servo accionamientos, *Siemens* también integra una librería de instrucciones *Motion Control* que también se atienen a la especificación PLCopen con lo que serán muy similares a las vistas en los softwares de *Schneider*. Puede consultarse la información detallada de la librería en [21].

7.1.3 Software de Usuario

Siemens dispone de un entorno denominado *TiA Portal* que integra todas las herramientas que pone a disposición del usuario para la configuración y parametrización/programación de los distintos componentes del sistema de control de movimiento. En el marco de este proyecto cabe destacar dos:

- **STEP 7:** Permite realizar la configuración hardware y programación del PLC.
- **Startdrive:** Permite realizar la configuración hardware y parametrización (puesta en marcha) del servo drive.

[15] https://www.new.siemenes.com/global/en/products/drives/sinamics/low-voltage-converters/servo-converter.html [Consultado en: Julio de 2021]

Se aconseja la lectura de [22] donde se realiza una introducción a *TiA Portal* haciendo un previo repaso del hardware (SIMATIC y SINAMICS) y terminando con un ejemplo de aplicación.

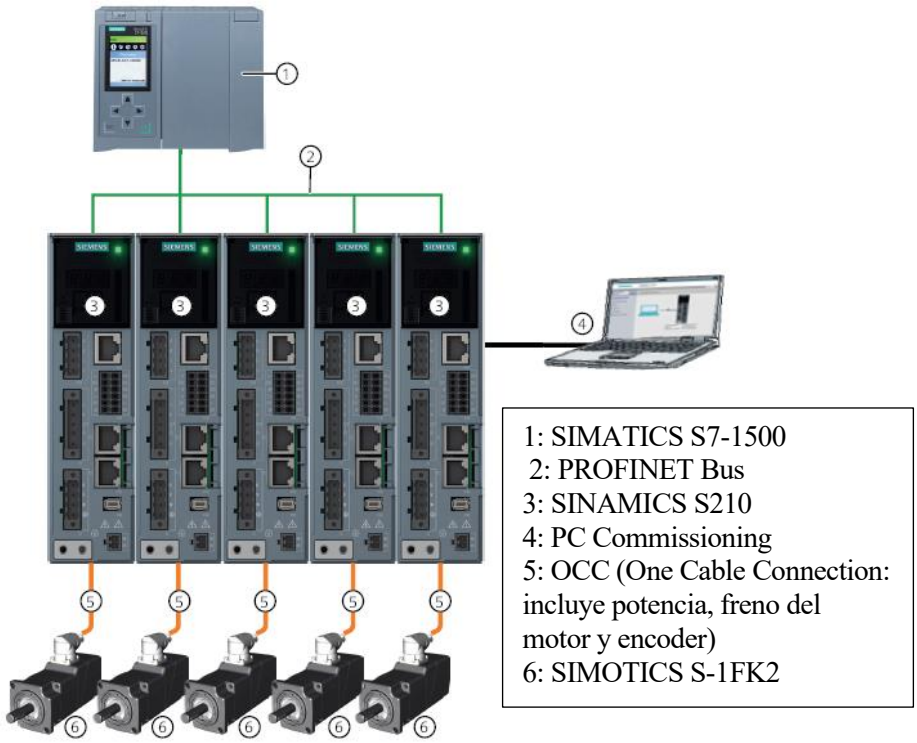


Figura 7-2: Ejemplo de Sistema de control de movimiento de *Siemens*.

7.2 Maqueta de pruebas para visualización del control de par

Con la implementación práctica del capítulo anterior, el usuario puede activar el modo de funcionamiento de control de par (Profile Torque) en los servo accionamientos. Sin embargo, más allá de visualizar por software (a través del supervisor (apartado 6.6.4)) que el par aplicado por el servomotor es igual al demandado, el usuario no puede comprobar que efectivamente el control en par está funcionando. Por ello, se propone la realización de otra maqueta de pruebas donde sea posible verificarlo **visualmente** acoplando a ella uno de los servo accionamientos disponibles.
La maqueta propuesta es la mostrada en la Figura 7-3. Estaría conformada por un sistema polea-correa que tendría acoplado el servomotor en la polea grande derecha y un motor paso a paso en la polea grande izquierda. Adicionalmente se instalarían dos muelles con una cierta tensión en las poleas pequeñas de la izquierda que servirían para visualizar el par aplicado.

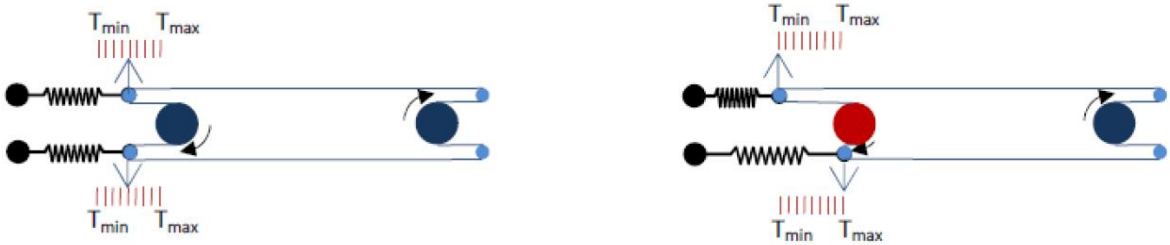


Figura 7-3: Maqueta para visualización de control de par

En el esquema de la derecha se muestra el sistema en funcionamiento. Se aplicaría un par determinado con el modo Profile Torque a través del servomotor a la vez que las bobinas del motor paso a paso están cortocircuitadas para que este actue como freno (pero sin llegar a detener el movimiento de la correa). De esta forma, la tensión en la parte baja de la correa aumenta (el servomotor tira de ella) y se refleja en muelle inferior. Mientras tanto, la tensión en la parte alta disminuye lo que se refleja en el muelle superior.
Es aconsejable colocar unos interruptores que sirvan para dar señal de parada al servo drive si la correa se tensa en exceso para evitar que esta se parta.

ANEXO A: FUNDAMENTOS PROTOCOLO CANOPEN

El presente Anexo pretende mostrar solo algunas nociones básicas del protocolo CANOpen. Una mayor información acerca del protocolo puede consultarse en [23].
CANOpen es un protocolo de comunicaciones de alto nivel basado en CANbus y desarrollado por la asociación CiA (CAN in Automation). Permite la interoperabilidad entre diferentes dispositivos (nodos) de un sistema, así como su configuración durante y después de su instalación. Hoy en día, es ampliamente utilizado para el control de la operativa de motores paso a paso y servomotores. Sin embargo, puede ser utilizado en otras muchas otras aplicaciones.

Figura A-1: Aplicaciones CANOpen

Según el modelo ISO-OSI, el cual estandariza las distintas funciones dentro de un protocolo de comunicación red y las agrupa en 7 posibles capas, la tecnología CANbus se corresponde con la implementación de las dos capas inferiores (capa física, a nivel de bit y capa de enlace de datos, a nivel de trama) representando el "medio de transporte" de los datos a transmitir entre los dispositivos. Por otro lado, el protocolo CANOpen implementa la capa superior (capa de aplicación, a nivel de dato). En los buses industriales o de campo, las capas intermedias del modelo OSI pierden su utilidad. El protocolo CANOpen concentra todos los servicios necesarios de las capas superiores solo en la última capa (Figura A-2).

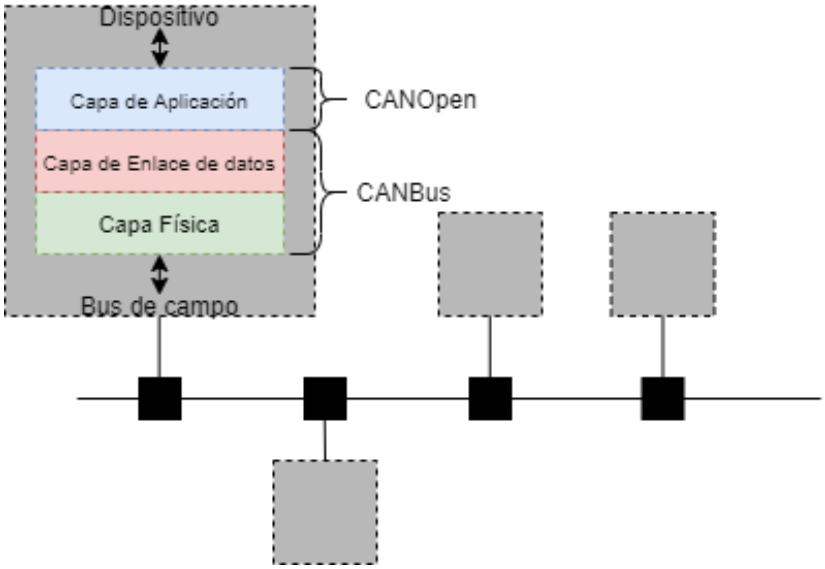


Figura A-2: Capas CAN en Modelo OSI

A.1 Diccionario de Objetos (OD)

Cada dispositivo de la red cuenta con un diccionario de objetos (OD) que contienen un grupo de parámetros. Estos parámetros describen como se establece su comunicación con el resto de los dispositivos de la red, su configuración interna y el control/ejecución de sus funciones.

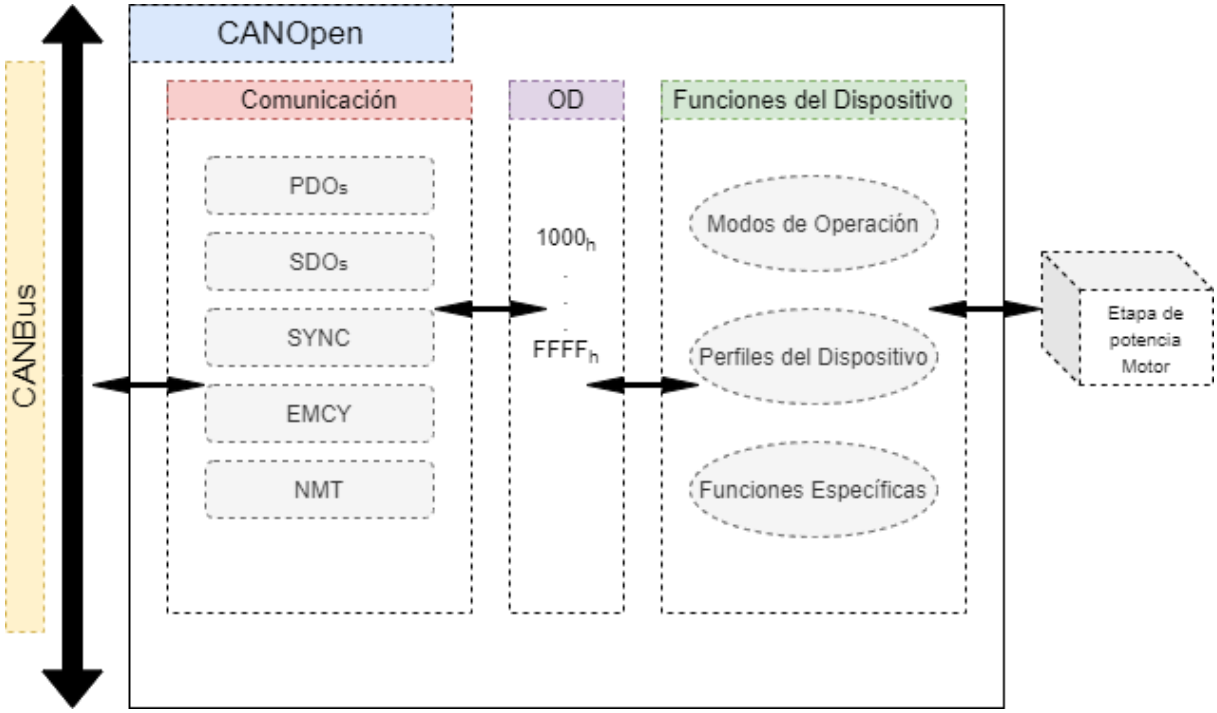


Figura A-3: Dispositivo de una red CAN

Los parámetros del OD de cualquier dispositivo de la red estarán agrupados en una serie de perfiles o conjuntos de objetos. Entre ellos cabe destacar:

- **Perfil de comunicación (DS301):** estos objetos se encargan de gestionar las funciones de comunicación del dispositivo con la red.
- **Perfil de dispositivo (DSP402):** estos objetos se encargan de gestionar las funciones de monitorización y parametrización del propio dispositivo.
- **Perfil del fabricante:** estos objetos recogen todas las funciones que el fabricante ofrece para la explotación del dispositivo.

Los diferentes parámetros se identifican y diferencian unos de otros haciendo uso de un índice (16 bits) y un subíndice (8 bits) (Tabla A-1).

Índice	Perfil
$1xxx_h$	Perfil de comunicación
$3xxx_h$	Perfil de dispositivo
$6xxx_h$	Perfil del fabricante

Tabla A-1: Perfiles de objetos en LXM32M

Por último, cabe mencionar que existe el concepto de ficheros EDS que no son más que los propios OD en un fichero interpretable por el usuario. Para cada parámetro se definen una serie de campos: nombre, valores límites, valor por defecto, tipo de acceso permitido (lectura, escritura), si admite ser mapeada en un PDO, tipo de dato, etc (Figura A-4).

Cuando el usuario para la configuración de un dispositivo edita su fichero EDS añadiendo valores a sus parámetros, pasa a denominarse DCF (device configuration file).

```
[1000]
ParameterName=DeviceType
ObjectType=0x7
DataType=0x7
AccessType=ro
PDOMapping=0
DefaultValue=0x00420192

[1001]
ParameterName=ErrorRegister
ObjectType=0x7
DataType=0x5
AccessType=ro
PDOMapping=0

[1018]
ParameterName=Identity Object
ObjectType=0x9
SubNumber=4

[1018sub0]
ParameterName=number of elements
ObjectType=0x7
DataType=0x5
LowLimit=1
HighLimit=4
AccessType=ro
DefaultValue=3
PDOMapping=0
```

Figura A-4: Ejemplo de contenido de fichero EDS.

A.2 Mensajes CANOpen

Los mensajes a través de los cuales se hace efectiva la comunicación en una red CAN están conformados por diferentes partes (Figura A-5):

- **COB-ID (11 bits):** Identifica el nodo que transmite/recibe los datos (7 bits) y que función de comunicación emplea.

- **RTR (1 bit):** Indica si el mensaje contiene un dato o contiene una petición procedente desde otro dispositivo.

- **Dato (68 bits):** Los 4 primeros bits indican la longitud del dato/petición y los 64 bits restantes son los disponibles para la transmisión del dato/petición en sí.

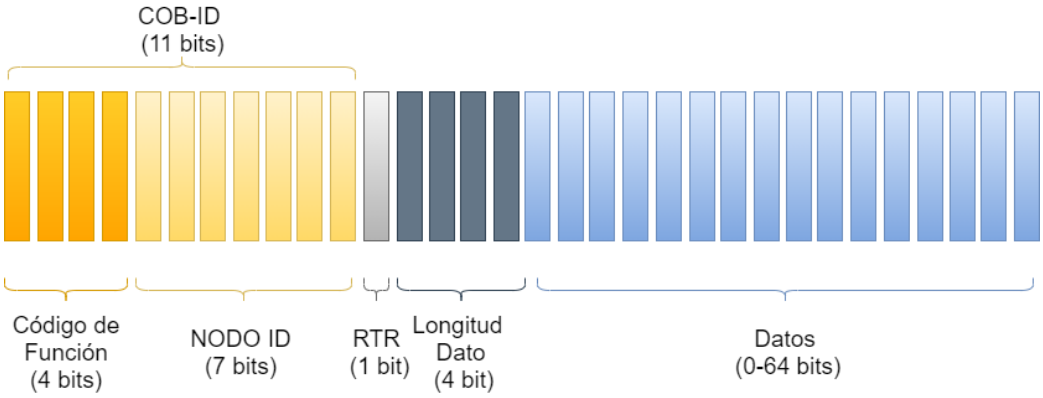


Figura A-5: Mensaje CANOpen

A.3 Funciones de comunicación

Las funciones que permiten la comunicación de un dispositivo con la red y que se encuentran incluidas dentro del perfil de comunicación del OD son las siguientes:

1. **NMT (Network Management)**: permite controlar el estado operacional de los dispositivos de la red mediante distintos comandos (start, stop, reset...)

2. **SYNC (Synchronization)**: permite la sincronización en la transmisión de datos o en la actuación de varios dispositivos de la red.

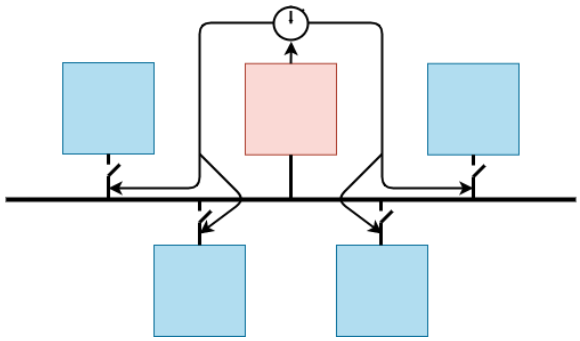


Figura A-6: Función SYNC

3. **EMCY (Emergency)**: permite notificar los eventos de error que puedan ocurrir en los dispositivos de la red.

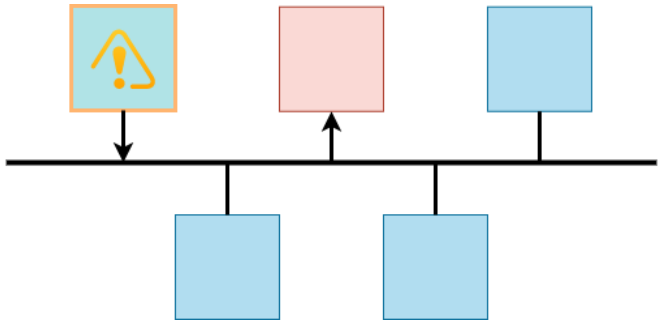


Figura A-7: Función EMCY

4. **TIME (Time Stamp)**: permite transmitir desde un dispositivo *Maestro* datos de tiempo a la red (tiempo global de la red).

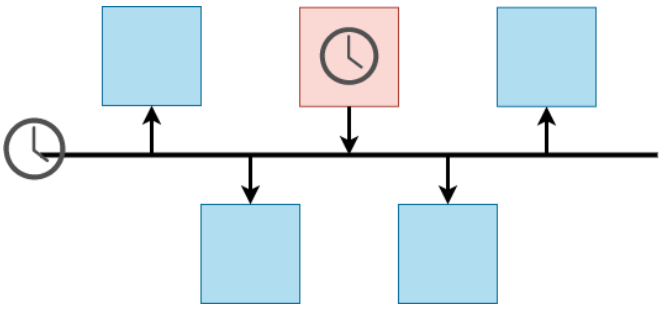


Figura A-8:Función TIME

5. **PDO (Process Data Object)**: permite la transmisión de datos entre dispositivos de la red.

6. **SDO (Service Data Object)**: permite leer/escribir valores de los objetos del OD de cualquier dispositivo de la red.

7. **Heartbeat (Node Monitoring)**: permite mandar un mensaje de confirmación de conexión del dispositivo a la red para que, en caso de error, el dispositivo *Maestro* actúe en consecuencia.

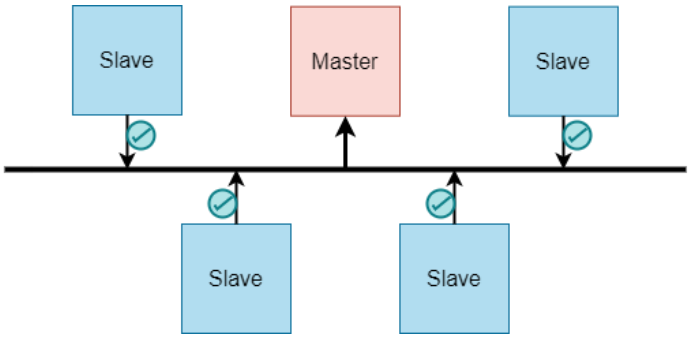


Figura A-9: Función Heartbeat

A.3.1 Transmisión de datos. PDOs y SDOs.

Tanto PDOs como SDOs nos permiten transmitir datos de un dispositivo de la red a otro. Sin embargo, existen notables diferencias entre ellos.

Los SDOs utilizan parte del espacio del mensaje reservado para el dato, para indicar si se trata de una petición de lectura o escritura y para incluir la información del índice y subíndice del parámetro en el OD del dispositivo al que se manda. Es decir, la función SDO emplea un modelo de comunicación entre nodos de tipo "cliente/servidor" (Figura A-10).

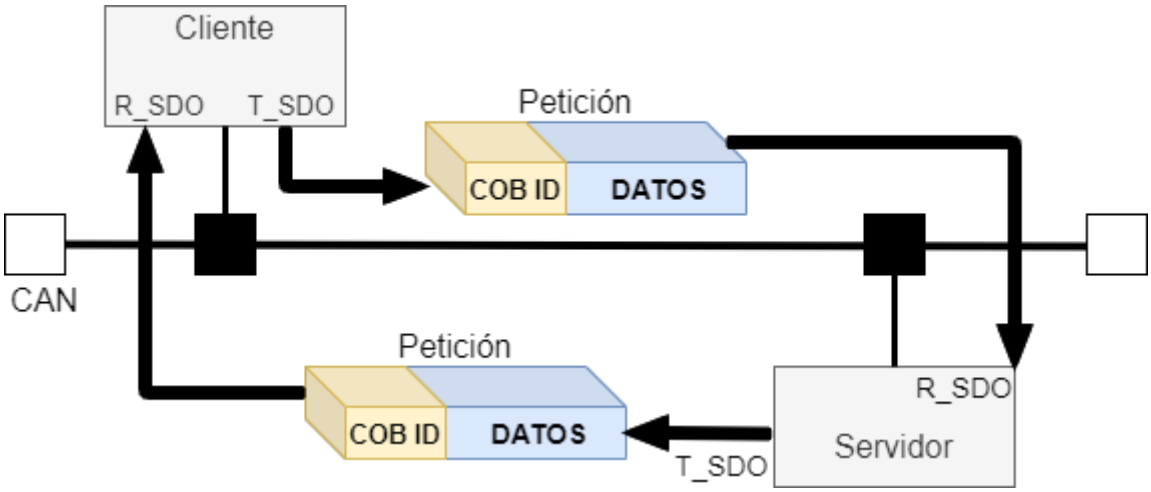


Figura A-10: Intercambio de datos con función SDO

Por otra parte, los PDOs no van provistos de esa cabecera por lo que dejan más espacio para la transmisión de datos efectivos entre los dispositivos que los SDOs. En el caso de los PDOs, habrá un dispositivo que transmitirá los datos a la red para que los reciban uno o varios de los dispositivos conectados a la misma (modelo de comunicación de tipo "productor/consumidor" (Figura A-11)). Por defecto, cada dispositivo solo tiene acceso a 4 PDOs.

Además, los receptores deberán conocer la disposición interna de la trama de los mensajes PDOs (PDO Mapping) ya que estos no contienen información para identificar al parámetro objetivo. Dependiendo del dispositivo, algunos PDOs pueden tener un mapeado fijo (el contenido que se envía/recibe por él no puede cambiarse) mientras que el resto permitirán un mapeado dinámico (el programador decide que parámetros se envían/reciben en cada PDO).

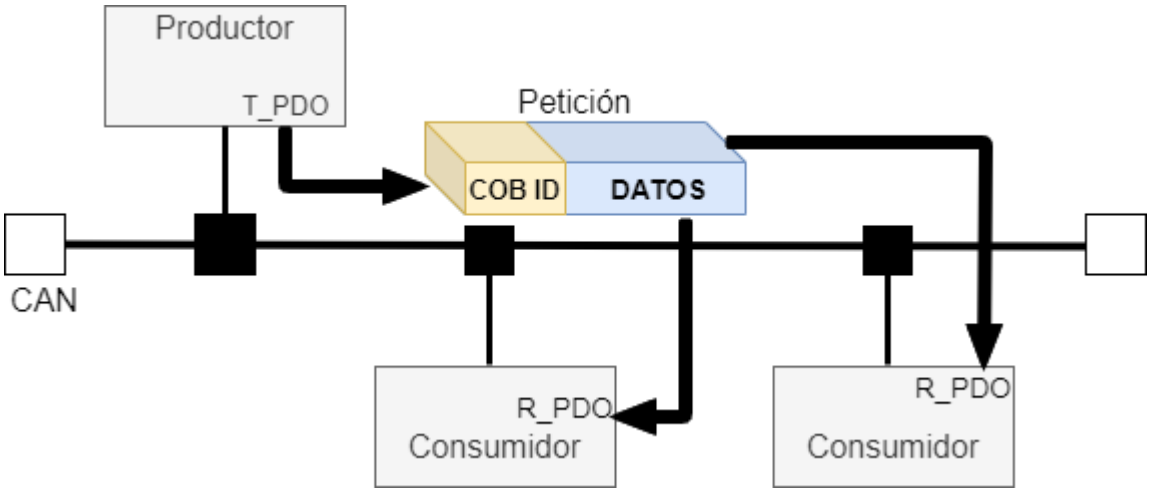


Figura A-11:Intercambio de datos con función PDO

REFERENCIAS

[1] K. Cope, «RealPars,» 29 Octubre 2018. [En línea]. Available: https://www.realpars.com/vfd. [Último acceso: Julio 2021].

[2] W. Gastreich, «RealPars,» 9 Julio 2018. [En línea]. Available: https://www.realpars.com/stepper-motor/. [Último acceso: Julio 2021].

[3] W. Gastreich, «RealPars,» 27 Agosto 2018. [En línea]. Available: https://www.realpars.com/servo-motor/. [Último acceso: Julio 2021].

[4] W. Gastreich, «RealPars,» 10 Septiembre 2018. [En línea]. Available: https://www.realpars.com/stepper-motors-advantages/. [Último acceso: Julio 2021].

[5] Schneider Electric, LXM32M Servo accionamiento AC: Manual del producto, 2014.

[6] Schneider Electric, «Schneider Electric,» [En línea]. Available: https://www.se.com/es/es/product/BSH0551T01A2A. [Último acceso: Julio 2021].

[7] Schneider Electric, BSH Servomotor: Manual del motor, 2017.

[8] Schneider Electric, LXM32M Servo accionamiento AC: Módulo CANOpen. Manual del bus de campo, 2018.

[9] Instituto Schneider Electric de Formación, «Práctica 14 - Motion Sequence,» de *Lexium 32: Manual de formación*, pp. 82-88.

[10] C. Pena, «infoPLC,» 23 Abril 2008. [En línea]. Available: https://www.infoplc.net/files/descargas/schneider/infoPLC_net_LXM32.pdf. [Último acceso: Julio 2021].

[11] Schneider Electric, EcoStruxure Control Expert: Motion Function Blocks, Block Library, 2018.

[12] Schneider Electric, Lexium Library: Function blocks Software Manual, 2012.

[13] Schneider Electric, SoMachine: Guía de programación, 2018.

[14] Schneider Electric, «CANopen Connection,» de *M238 Logic Controller: Hardware Guide*, 2011, pp. 97-100.

[15] Schneider Electric, «Connecting the Modicon M238 Logic Controller to a PC,» de *M238 Logic Controller: Hardware Guide*, 2011, pp. 89-92.

[16] Schneider Electric, «Topologías y Diseño de red CANopen,» de *CANOpen: Manual de configuración Hardware*, 2018, pp. 11-34.

[17] Schneider Electric, Modicon M340 - CANOpen: Manual de Usuario, 2018.

[18] S. Lozano, «Schneider Electric,» 24 Noviembre 2015. [En línea]. Available: https://www.se.com/ar/es/faqs/FA27568/. [Último acceso: Julio 2021].

[19] P. G. Isart, «infoPLC,» 08 Octubre 2017. [En línea]. Available: https://www.infoplc.net/documentacion/210-motion-control-servos-mecatronica/267-curso-motion-control-4-perfiles-cam. [Último acceso: Julio 2021].

[20] Siemens, SINAMICS S210 Operating Instructions Manual, 2021.

[21] Siemens, S7-1500 Motion Control V4.0 in TIA PORTAL V15. Function Manual, 2017.

[22] P. C. Pomareta, Introducción a TIA Portal con S7-1500, Madrid: Universidad Politécnica de Madrid, 2017.

[23] Schneider Electric, Manual de introducción al CANOpen, bus de campo para máquinas e instalaciones, 2018.