PRÁCTICA DE CONTROL DE MOTORES

Servocontrol de motores de corriente continua y motores paso a paso

RECURSOS UTILIZADOS

Motores Maxon de Corriente continua junto con driver. Motores Paso a Paso junto con Driver. Control desde Matlab. Control desde PLC

Ángel Gaspar González Rodríguez
Automatización y Control

1 Objetivos y conceptos fundamentales

En esta práctica se pretende mostrar la posibilidad de control tanto de motores de corriente continua con escobillas como motores paso a paso.

El primero de ellos será accionado mediante un driver específico, que se conecta a un dispositivo digital encargado de transmitir las órdenes de posicionamiento o velocidad. Dichas órdenes pueden ser enviadas desde una aplicación creada al efecto o, como será este caso, desde Matlab.

La segunda parte de la práctica corresponde a una secuencia de movimientos almacenados como array de posiciones y velocidades en un bloque de datos del PLC. Un programa se encargará de recorrer todas las posiciones y de enviar el tren de pulsos correspondiente a las tarjetas de potencia.

2 Control de motores de corriente continua

2.1 Componentes

En la Ilustración 1 se muestra el aspecto y composición de un servomotor de corriente continua, de características similares al utilizado en la práctica.

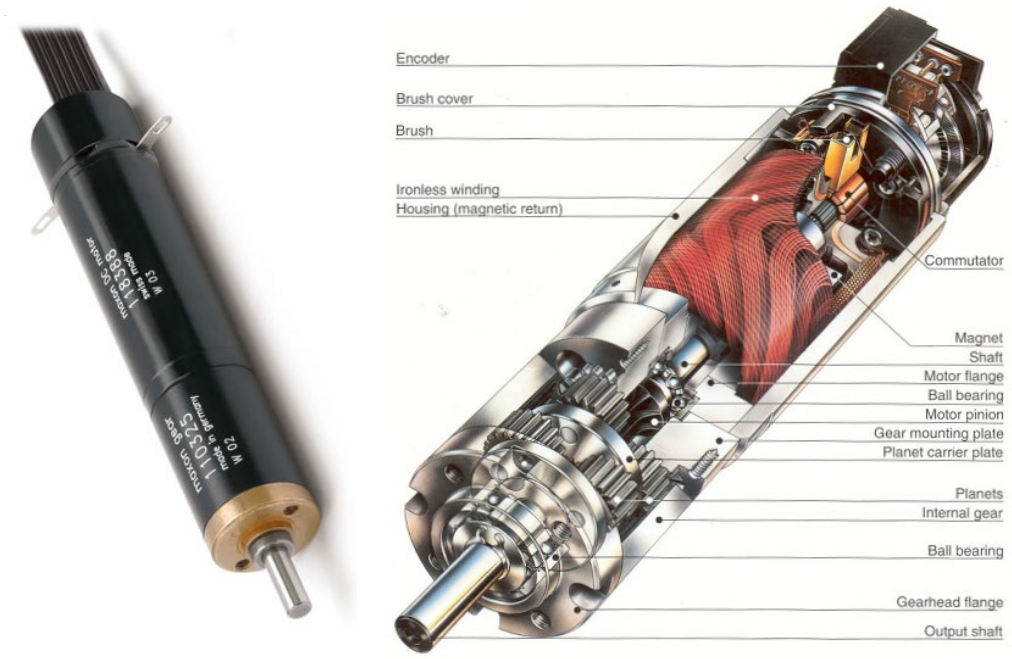

Ilustración 1. Aspecto y componentes de un motor de corriente continua

Como se muestra, el servomotor Maxon utilizado consta de los siguientes elementos, que son ensamblados en fábrica:

- Motor de corriente continua propiamente dicho, que es el cuerpo intermedio.
- Elemento de sensorización, que en este caso es un encóder incremental
- Reductor de velocidades de engranajes planetarios

En la plataforma de Docencia Virtual puede encontrarse manuales de estos componentes.

Para alimentar el servomotor se utiliza una tarjeta EPOS 24/5 también de Maxon (ver Ilustración 2 para una tarjeta similar). Dicho fabricante proporciona el software y librerías para comunicar el PC con dicha tarjeta a través de un cable USB (también es posible mediante una red CAN).

Para poder utilizar este software es preciso realizar una parametrización partiendo de las hojas de características de cada uno de los componentes (principalmente motor y encóder).

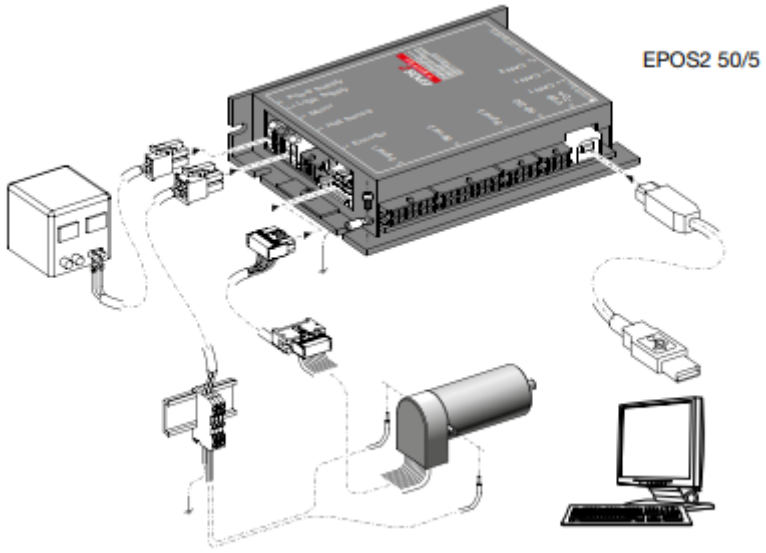

Ilustración 2. Tarjeta de control EPOS, y conexiones a realizar

2.2 Parametrización del servomotor

Primeramente se accederá a la aplicación de Maxon EPOS Studio. Se conectará el motor y se alimentará a 24 V.

Se elegirá la conexión USB y se accederá al asistente (Wizard) de Configuración.

A continuación se muestran las distintas ventanas con las que se parametriza el accionamiento, accesibles a través de la aplicación EPOS Studio.

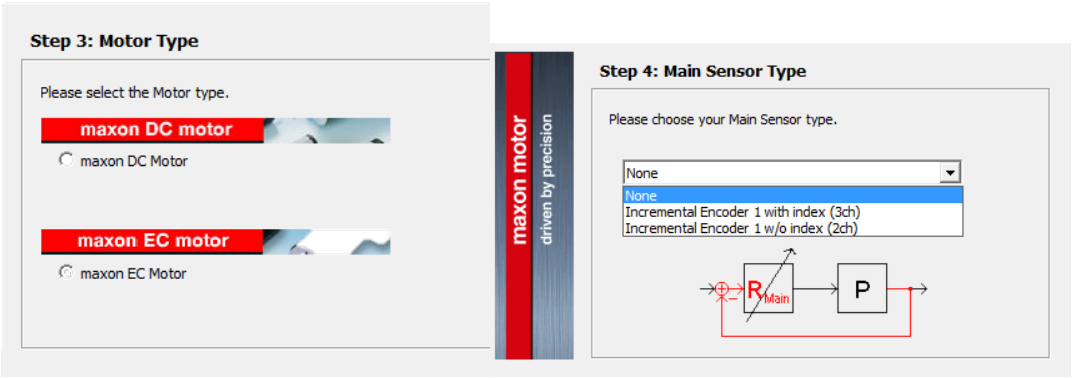

Ilustración 3. a) Selección DC o EC; b) Selección del encóder

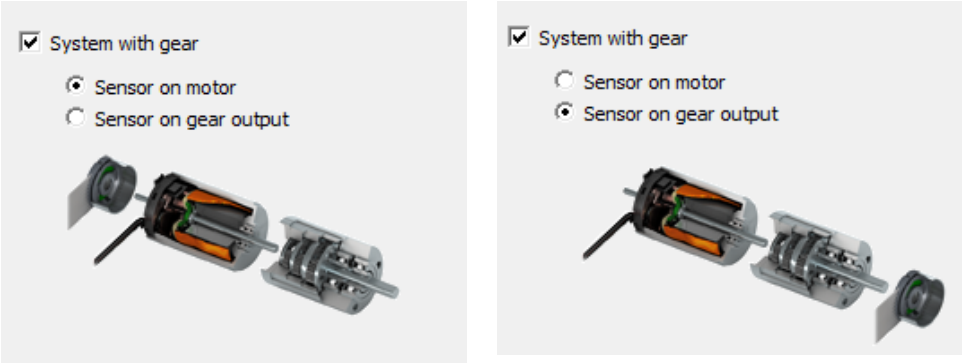

Ilustración 4. Selección de la posición del encóder

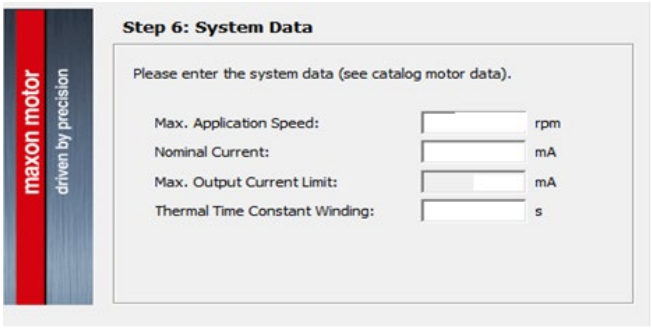

Ilustración 5. Introducción de parámetros

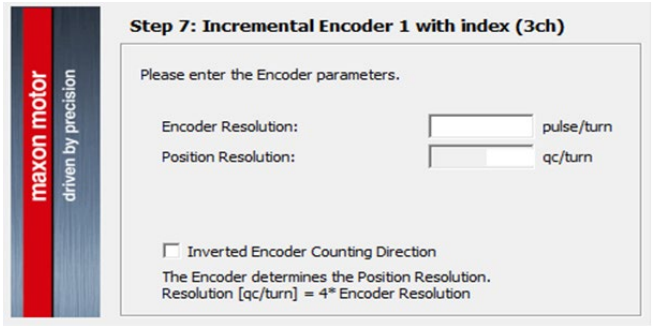

Ilustración 6. Parámetros del encóder

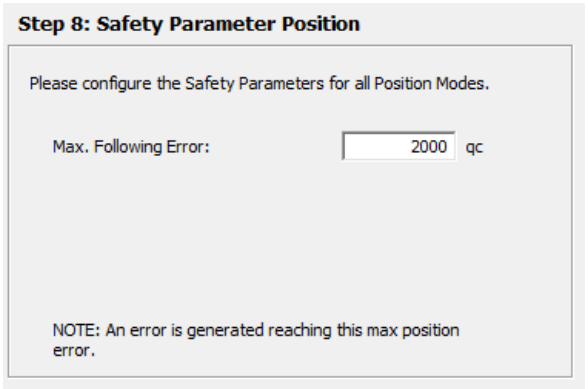

Ilustración 7. Error de seguimiento

A partir de las hojas de características, deducir todos los parámetros de ajuste mostrados en las figuras anteriores y anotar sus valores (para el límite máximo de corriente de salida, utilizar 5 A, y para el error de seguimiento, dejarlo en 2000 quads)

2.3 Ajuste de las ganancias de los controladores PI y PID

Una vez identificados los parámetros del motor y encóder, hay que definir las ganancias del controlador PID de velocidad y PI de posición. Esto puede hacerse manualmente, pero Maxon ofrece un asistente (Wizard) que permite el ajuste de estas ganancias de forma automática.

Una vez ajustadas las ganancias, la tarjeta será capaz de accionar al motor de manera que la velocidad y/o posición siga el perfil prefijado (ver Ilustración 8).

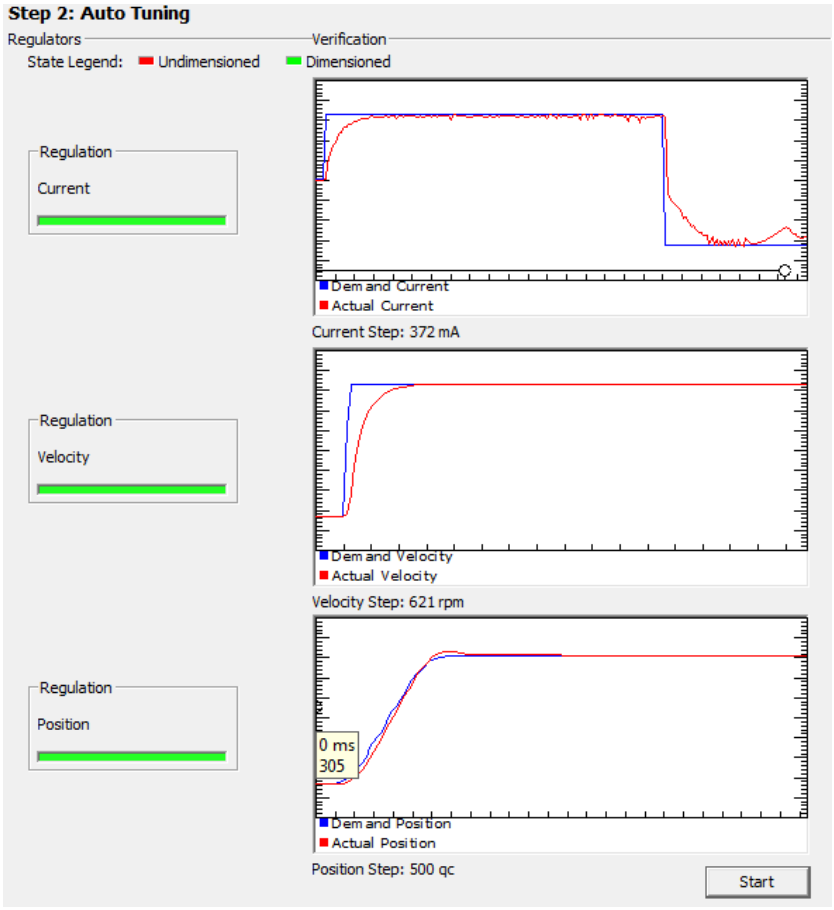

Ilustración 8. Resultado del asistente de ajuste de ganancias.

Tras ejecutar el asistente de ajuste de ganancias es conveniente ejecutar la herramienta (Tools) de movimiento siguiendo un determinado perfil de posición. Esto permite saber el valor de las aceleraciones más apropiadas, así como conocer la relación de reducción del tren de engranajes, si es que esta se desconoce. Después de ejecutar el asistente, hay que deshabilitar el motor para que pueda ser utilizado por otra aplicación distinta de EPOS Studio.

2.4 Programar una secuencia de movimientos

El objetivo es conseguir una secuencia de trayectorias con diferentes perfiles de velocidad y aceleración, para lo que se realizará un pequeño programa en Matlab que efectúe una serie de movimientos de posición.

Para ello, se utilizará la ayuda existente en

https://es.mathworks.com/matlabcentral/fileexchange/53735-commanding-maxon-motors-epos2-motor-controller-from-matlab

Partiendo de la página anterior, y de la ayuda sobre las distintas funciones utilizadas, indicar cuáles son los parámetros de entrada y de salida, así como la finalidad de las funciones que a continuación se relacionan: OpenCommunication; GetErrorState; ClearErrorState; EnableNode; SetOperationMode; SetProfilePositionData; MoveToPosition; WaitForTargetReached

Realizar un pequeño programa que inicie la comunicación con la tarjeta, la habilite, y realice una secuencia de 4 movimientos absolutos previamente definidos en un vector. Se utilizará una constante llamada rel_vel = 50 con la relación del tren de engranajes (su valor exacto se conocerá durante la práctica). Las posiciones se definirán en vueltas haciendo movimientos en ambos sentidos (p.ej 3, 1.5, 10.4, 6.4).

Las velocidades también se indicarán en un vector dado en rpm y al menos dos serán bastante diferentes (p.ej. 30, 30, 120, 120). Como aceleración en todos los casos, se utilizará p.ej. 20000, y como tiempo de espera timeout = 10000 ms.

Al programar el código, hay que tener en cuenta que tanto las posiciones como las velocidades vienen dadas en unas unidades diferentes de las que hay que ingresar como parámetros de las funciones anteriormente listadas. Por ejemplo, el vector de posiciones viene dado **en vueltas en el eje lento** mientras que la función de Matlab a utilizar viene dada en **quads medidos en el encoder**.

Para comprobar que se ha llegado al final del movimiento en cada caso, incluir la siguiente instrucción al finalizar cada movimiento (se supone que i es el índice contador).

```
display(strcat('Alcanzada posición: ',int2str(i)));
```

Nota: el valor del nodo o dirección de la tarjeta se puede imponer activando/desactivando los switches que se encuentran en la parte trasera de la tarjeta (ver Ilustración 9 para una tarjeta con direccion 1). De esta forma, se pueden conectar hasta 127 tarjetas a un mismo equipo de control. Lo usual es que, si sólo se dispone de una tarjeta, esta tenga dirección 0000 0001 (aunque en la tarjeta el switch del bit menos significativo está a la izquierda)

Ilustración 9. Parte trasera de la tarjeta EPOS 25/5

Durante la práctica, y antes de ejecutar el programa creado, es necesario añadir C:\Matlab\Version2 a los path de trabajo (Home --> SetPath --> Add Folder)

Utilizando las herramientas de depuración, variar la relación de velocidad hasta conseguir que en el primer posicionamiento, realmente haga el número de vueltas indicado (3 en el ejemplo).

3 Control de motores paso a paso

3.1 Componentes

En la segunda hora de la práctica se utilizarán motores paso a paso, controlados desde un PLC. Concretamente se utilizará el motor de la Ilustración 10, con un ángulo de paso de 1.8º, accionado por el driver de potencia de la Ilustración 11.

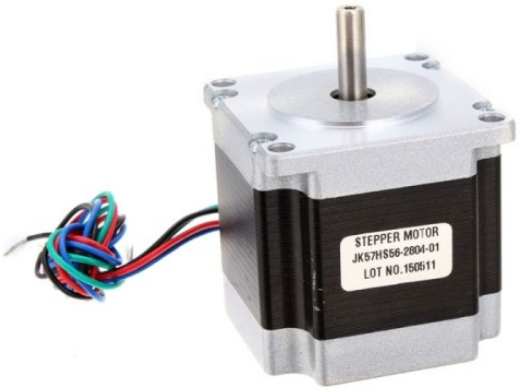

Ilustración 10. Motor paso a paso

Indicar las características principales del motor: tipo, alimentación unipolar/bipolar, par de retención, intensidad por fase.

En relación a las tarjetas de accionamiento de motores paso a paso, éstas suelen tener como entradas un tren de impulsos y la dirección del movimiento; y como salidas, los niveles de tensión a aplicar a las bobinas para mover el rotor. También han de ser alimentados con una fuente de tensión continua del nivel apropiado.

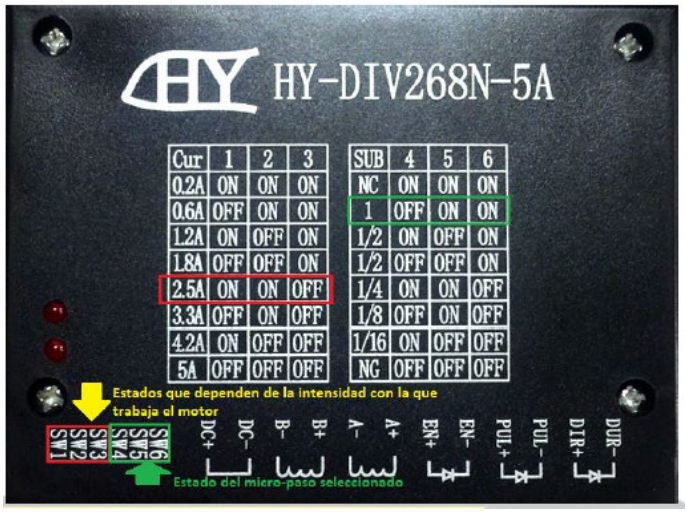

Ilustración 11. Tarjeta de alimentación del motor paso a paso

Tal como se ve en la Ilustración 11, la tarjeta permite obtener rotaciones menores que el ángulo de paso original (1.8º) dado por el fabricante del motor. Con los switches 4-5-6 se puede escoger que, por cada impulso suministrado, la rotación sea igual a 1, ½, ¼, 1/8 o 1/16 veces dicho ángulo. Cuando el divisor es 1 (como en el caso de la figura), un impulso dado en la entrada Pul+Pul- equivale a una rotación igual 1.8º.

Partiendo del ángulo de paso (α) y del divisor (1, 2, 4, 8, 16), obtener una relación entre el número de impulsos dados a la tarjeta y la rotación en grados. Obtener también la constante de proporcionalidad entre la frecuencia de impulsos y la velocidad en rpm.

Si la configuración es SW1 = SW2 = ON, SW3 = OFF, SW4 = SW5 = ON, SW6 = OFF, indicar a cuánto equivale un impulso.

3.2 Configuración del módulo tecnológico Motion Control

El módulo funcional de utilización de motores paso a paso es uno de los módulos tecnológicos que nos ofrece TIA Portal.

Para su configuración y puesta a punto se seguirán los siguientes pasos:

Objetos Tecnológicos → Agregar Objeto → Motion Control

Aquí se puede seleccionar TO_CommandTable (no se hará en esta práctica), que permite la definición de secuencias de movimientos y situaciones de espera. Dicha secuencia sería ejecutada posteriormente. Es un recurso rápido y cómodo aunque menos versátil. En su lugar, se programará la secuencia de movimientos y esperas mediante un Grafcet.

Tanto si se utiliza TO_CommandTable como si se programan uno a uno los movimientos, hay que configurar el accionamiento driver-motor. Para esto, se seleccionará Motion Control → TO_PositioningAxis, y se rellenarán todos sus campos:

Parámetros básicos. General.	**Parámetros avanzados. Mecánica**
Accionamiento por tren de pulsos	Impulsos por vuelta del motor: ¿?
Unidad de medida de posición: impulsos.	Movimiento de carga por vuelta del motor: 10
Parámetros básicos. Accionamiento.	Sentido de giro permitido: Ambos sentidos.
Generador de impulsos: Pulse_1	**Parámetros avanzados. Límites de posición**
Tipo de señal: PTO (impulso A, sentido B)	Activar final de carrera por SW.
Salida de impulsos: Pulsos en Q0.0	Resto de valores: los datos por defecto
Activar salida de impulsos: Sí	**Dinámica. General**
Salida de sentido: dirección. Q0.1	Unidad de los límites de velocidad: impulsos/s
Selección de salida de Habilitación: seleccionar la variable Eje_Habilitado: Q0.2	Velocidad máxima 25.000
Selección de entrada de disponibilidad: TRUE	Velocidad de arranque/parada: 100
	Aceleración y deceleración: 5000
	Dinámica. Parada de emergencia.
	Valores por defecto.

En relación al homing o referenciado, no se utilizarán sensores físicos. De todas formas, en la ventana de referenciado activo, rellenarlo tal como se muestra en la Ilustración 12.

Para el resto de campos, se dejarán los valores por defecto. Igualmente, para el referenciado pasivo, se dejarán todos valores por defecto.

Una vez configurado el eje, conviene probar el driver, el motor paso a paso y sus conexiones. Esta comprobación puede realizarse a través de la opción Objetos Tecnológicos→Eje_1→Puesta

en marcha (commissioning). En este asistente se elegirá Asumir → Habilitar y se definirá algún movimiento.

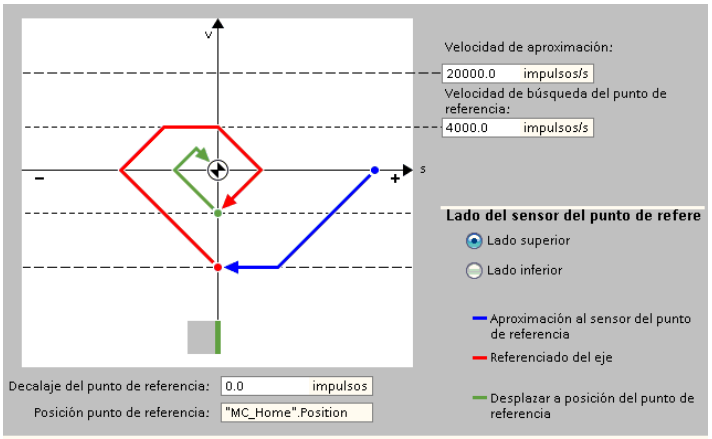

Ilustración 12. Parametrización del referenciado (homing) activo

3.3 Programación del código de posicionamiento

En Docencia Virtual se ha subido un fichero con el código del bloque OB1.

Realizar el Grafcet correspondiente.

El bloque funcional Calcula_Target está definido de la siguiente forma

```
#Divisor_del_paso := 1;
IF #SW4 AND #SW5 AND NOT #SW6 THEN
    #Divisor_del_paso := 4;
END_IF;
IF NOT #SW5 AND #SW6 THEN
    #Divisor_del_paso := 2;
END_IF;

#impulsos_por_vuelta := Expresión en función de #Divisor_del_paso  y #Paso_del_motor;
#rpm_2_imp_per_vuelta := Expresión en función de #Divisor_del_paso  y #Paso_del_motor;

#Pos_imp := "Targets".Posiciones_vueltas[#Indice] * #impulsos_por_vuelta;
#Vel_imp_s := "Targets".Velocidades_rpm[#Indice] * #rpm_2_imp_per_vuelta;
```

Como expresiones, se pondrían las obtenidas en el apartado 3.1.

Las posiciones y velocidades de crucero se han definido en el bloque de datos Targets

Posiciones_vueltas	**Array[1..8] of Real**		**Velocidades_rpm**	**Array[1..8] of DInt**	
Posiciones_vueltas[1]	Real	5	Velocidades_rpm[1]	DInt	600
Posiciones_vueltas[2]	Real	3	Velocidades_rpm[2]	DInt	600
Posiciones_vueltas[3]	Real	2.7	Velocidades_rpm[3]	DInt	600
Posiciones_vueltas[4]	Real	3.3	Velocidades_rpm[4]	DInt	6000
Posiciones_vueltas[5]	Real	10	Velocidades_rpm[5]	DInt	6000
Posiciones_vueltas[6]	Real	90	Velocidades_rpm[6]	DInt	6000
Posiciones_vueltas[7]	Real	20	Velocidades_rpm[7]	DInt	6000
Posiciones_vueltas[8]	Real	30	Velocidades_rpm[8]	DInt	6000