**MANUAL DE USUARIO DE LA LIBRERÍA DE LOS CONTROLADORES DE MOTORES EPOS2**

Una vez instalado y configurado el software de los EPOS2 en Matlab, se puede emplear para controlar varios motores Maxon Motors conectados a su correspondiente controlador Epos2. Es importante tener en cuenta que la librería solo es para hacer uso de los controladores, por lo que la etapa inicial de configuración y sintonización se debe hacer en el software EPOS Studio.

Para hacer uso de la librería se debe saber el número del nodo al cual se quiere comunicar. En los ejemplos se asume que el nodo es el número 1. Si hay más de un controlador conectado al bus USB, se debe configurar en EPOS Studio el primer nodo conectado, para que este actúe como un Gateway permitiendo la comunicación automática de los demás nodos con el PC a través del bus USB.

Una vez esté configurado el nodo, se puede hacer uso de la librería a través de los tres protocolos de comunicación que se tiene: uno procedural a través de las funciones compiladas .mex, un segundo procedimiento basado en objetos a través de la clase ‘*Epos2*’, o por último, a través de simulink usando la s-function “*sfun\_maxon*.c”.

A continuación se muestran las diferentes formas de comunicación usando la librería.

**Comunicación procedural.**

El primer paso para conectarse con el nodo es abrir la comunicación USB, para ello se llama a la función “OpenCommunication”, la cual abre establece la comunicación con el nodo y devuelve un número asociado con el “handle” del sistema operativo que realiza la comunicación a bajo nivel.

Ejemplo de uso,

>> a = OpenCommunication;

Una vez establecida la comunicación se debe proceder a detectar si el nodo se encuentra en modo error o se encuentra trabajando normalmente. En caso de error se debe resetear el mismo. El chequeo del error se hace llamando a la función “GetErrorState” la cual toma como parámetros el “handle” devuelto por “OpenCommunication” y el número del nodo.

Ejemplo de uso con el nodo número 1,

>> b = GetErrorState(a, 1);

Si el nodo se encuentra en error, se debe borrar el error antes de operar, para ello se llama a la función “ClearErrorState”, la cual toma como parámetros el “handle” y el número del nodo.

Ejemplo de uso,

If ( GetErrorState(a,1))

ClearErrorState(a,1);

end

Después de haber borrado los errores del nodo, se procede a habilitarlo de manera que se active la etapa de potencia que mueve el motor. La activación se realiza llamando a la función “EnableNode”, la cual toma como parámetros el “handle” y el número del nodo.

Ejemplo de uso.

>> EnableNode(a, 1);

Antes de hacer que el controlador mueva el motor, es necesario hacer que el nodo trabaje en el modo de operación correcto. Los modos de operación que se han programado son:

1. Modo Homming: para hacer la búsqueda inicial del cero máquina.
2. Modo Position: para mover el motor de un punto a otro.
3. Modo Profile Position: para mover el motor siguiendo un patrón de perfil en posición.
4. Modo Velocity: para mover el motor a velocidad constante sin rampa de aceleración.
5. Modo Profile Velocity: para mover el motor a velocidad constante siguiendo una rampa en aceleración.
6. Modo Current: para mover el motor en el modo corriente, requiere que el computador tenga una baja latencia de operación para poder cerrar los lazos de control en posición o velocidad.

Para el caso de perfil posición, que es el más común a realizar en robots que se programen con la cinemática inversa, se procede a llamar a la función “SetOperationMode” para configurar el modo de operación y posteriormente se programan los parámetros del perfil de posición con un llamado a “SetProfilePositionData”.

“SetOperationMode” se llama con los siguientes parámetros: handle, identificador del nodo, y un número relacionado con el modo de operación deseado, los cuales son:

1. Para modo Homming.
2. Para modo Current.
3. Para modo Velocity.
4. Para modo Position.
5. Para modo Profile Velocity.
6. Para modo Profile Position.

La función “SetProfilePositionData” se emplea para ajustar los parámetros con los cuales se hará el perfil de posición del motor. Los valores que acepta son la velocidad del perfil y la aceleración/desaceleración del perfil.

Para mover el motor se hace un llamado a las funciones “MoveToPosition” en caso de modo perfil posición, “SetPosition” en caso modo posición, “MoveWithVelocity” en caso modo perfil velocidad, “SetVelocity” en caso modo velocidad y “SetCurrent”, en caso modo corriente.

Ejemplo de uso del procedimiento a seguir para mover un motor en el modo perfil posición:

>> SetOperationMode(a, 1, 6); % modo perfil posicion

>> SetProfilePositionData(a, 1, 3000, 3000); % velocidad del perfil de 300 rpm, aceleración de 300 rpm/s

>> MoveToPosition(a, 1, 132000, 1); % mover a 132000 pulsos cuadratura de manera absoluta

Ejemplo de uso para modo perfil velocidad:

>> SetOperationMode(a, 1, 5); % modo perfil velocidad

>> SetProfileVelocityData(a, 1, 2000); % aceleracion de 2000 rpm/s

>> MoveWithVelocity(a, 1, 3000); % mover a 3000 rpm/s

Para detener el motor, ya sea de emergencia, o por un cambio en el valor del destino, se hace un llamado a la función QuickStop,

>> QuickStop(a, 1);

Ejemplo de uso para modo posición:

>> SetOperationMode(a, 1, 4); % modo posicion

>> SetPosition(a, 1, 132000); % mover a 132000 pulsos cuadratura de manera absoluta

Ejemplo de uso para modo velocidad

>> SetOperationMode(a, 1, 3); % modo velocidad

>> SetVelocity(a, 1, 3000); % mover a 3000 rpm

Ejemplo de uso para modo corriente

>> SetOperationMode(a, 1, 2); % modo corriente

>> SetCurrent(a, 1, 2000); % mover hasta alcanzar los 2 Amperios

La función “IsTargetReached” se emplea para detectar si el motor alcanzó el objetivo deseado o “setpoint”. Por ejemplo, para el caso perfil posición, es deseable esperar hasta que el motor alcance la posición antes de enviar un nuevo comando.

>> SetOperationMode(a, 1, 6); % modo perfil posición

>> SetProfilePositionData(a, 1, 3000, 4000); % velocidad de 3000 rpm, aceleracion de 400 rpm/s

>> MoveToPosition(a, 1, 132000, 1) % mover de manera absoluta hasta 132000 pulsos

>> do; a = IsTargetReached(a, 1); while (a ==0); % espera a que termine

La función “WaitForTargetReached” es preferable usarla en vez de la secuencia do .. while del ejemplo anterior. Esta función espera a que se termine la tarea antes de continuar con la siguiente instrucción.

Ejemplo de uso:

>> WaitForTargetReached(a, 1, 5000);

Una vez se ha terminado de operar con el motor se debe deshabilitar el nodo y cerrar la comunicación, esto se hace con los comandos “DisableNode” y “CloseCommunication”.

Ejemplo típico de uso:

>> DisableNode(a, 1);

>> CloseCommuncation(a);

**Comunicación basada en objetos.**

Para la comunicación basada en objetos se crearon las clases “Epos2”, “OperationModes” y “HommingModes”. Estas clases buscan facilitar el uso de los Epos2 para los usuarios más novatos que inician con los epos ya que internamente se hace una serie de validaciones que permiten diagnosticar con facilidad los errores más comunes que se estén cometiendo durante la secuencia lógica de operación de los Epos.

Los objetos se crean llamando al constructor Epos2 y como parámetro de entrada el número de nodo.

Ejemplo:

>> Motor1 = Epos2(1); % creación de la comunicación con el nodo número 1

Una vez creado el objeto, se procede a detectar errores y borrarlos en caso de existencia, para ello se utilizan los métodos “IsInErrorState” y “ClearErrorState”.

Ejemplo:

If (Motor1.IsInErrorState)

Motor1.ClearErrorState;

End

Para habilitar el nodo se llama al método EnableNode,

>> Motor1.EnableNode;

Para deshabilitarlo se llama al método DisableNode,

>> Motor1.DisableNode;

Para cambiar el modo de operación se llama al método “SetOperationMode” con parámetro a la clase “OperationModes”. Ejemplo:

>> Motor1.SetOperationMode( OperationModes.ProfilePositionMode ); % para perfil posición

>> Motor1.SetOperationMode( OperationModes.ProfileVelocityMode ); % para perfil velocidad

>> Motor1.SetOperationMode( OperationModes.PositionMode ); % para modo posición

>> Motor1.SetOperationMode( OperationModes.VelocityMode ); % para modo posición

>> Motor1.SetOperationMode( OperationModes.CurrentMode ); % para modo corriente

>> Motor1.SetOperationMode( OperationModes.HommingMode ); % para modo home

Para el modo de home, el cual se selecciona con “OperationModes.HommingMode”, se debe seleccionar el modo de hacer home llamando al método “SetHommingMethod”, el cual permite varios parámetros de entrada, según el tipo de home a utilizar. Las opciones de home se agrupan en la clase “HommingMethods”, la cual define una enumeración que consta de las siguientes definiciones:

1. “HomeSwitchNegSpeed”, para hacer un home basado en un interruptor de final de carrera (Limit Switch) donde el motor se mueve en sentido de giro negativo para realizar su búsqueda.
2. “HomeSwitchNegSpeedIndex”, similar al caso anterior, solo que una vez se alcanza el final de carrera, se realiza una búsqueda del canal “Index” del encoder para ubicarlo en cero.
3. “HomeSwitchPosSpeed”, similar al caso a), pero el motor gira con velocidad angular positiva para realizar la búsqueda del final de carrera.
4. “HomeSwitchPosSpeedIndex”, similar al caso c), pero el motor se mueve en sentido positivo.
5. “CurrentNegSpeed”, realiza la búsqueda del cero máquina moviendo el motor en sentido de giro negativo hasta que se alcance un bloqueo mecánico que haga que la corriente supere un valor límite.
6. “CurrentNegSpeedIndex”, similar al caso e), pero al alcanzar el tope mecánico y superar la corriente límite se realiza una búsqueda del cero del canal “índice” del encoder.
7. “CurrentPosSpeed”, similar al caso e), pero moviendo el motor en sentido positivo.
8. “CurrentPosSpeedIndex”, similar al caso g), agregando la búsqueda de cero del canal índice del encoder.
9. “ActualPosition”, toma la posición actual del encoder como el cero máquina, lo que causa que se resetee el contador del encoder a cero.

Una vez se haya seleccionado el método de la búsqueda a cero, se procede a realizar dicha búsqueda llamando al método “DoHomming”.

A continuación se visualizan algunos ejemplos de las formas más comunes de realizar la búsqueda a cero de una máquina, o procedimiento home.

Para posición actual,

>> Motor1.SetOperationMode( OperationModes.HommingMode ); % Modo de operación homming

>> Motor1.SetHommingMethod( HommingMethods.ActualPosition ); % Modo home a posición actual

>> Motor1.DoHomming; % realiza el procedimiento home

Para velocidad negativa sin índice,

>> Motor1.SetOperationMode( OperationModes.HommingMode );

>> Motor1.SetHommingMethod( HommingMethods.HomeSwitchNegSpeed , 3000, 6000 ); % búsqueda del cero con velocidad de 3000 rpm y aceleracion de 6000 rpm/s

>> Motor1.DoHomming; % empieza la búsqueda del cero

Para velocidad positiva y con índice,

>> Motor1.SetOperationMode( OperationModes.HommingMode );

>> Motor1.SetHommingMethod( HommingMethods.HomeSwitchPosSpeedIndex, 3000, 100, 6000); % búsqueda del cero con velocidad de 3000 rpm, búsqueda del índice con velocidad de 100 rpm y aceleración para ambos casos de 6000 rpm/s

>> Motor1.DoHomming; % empieza la búsqueda del cero

Para búsqueda a cero con límite de corriente,

>> Motor1.SetOperationMode( OperationModes.HommingMode );

>> Motor1.SetHommingMethod( HommingMethods.CurrentPosSpeed, 600 ); % búsqueda del cero con motor girando en sentido positivo y límite de corriente de 600 mA.

>> Motor1.DoHomming; % empieza la búsqueda del cero

Para mover el motor en modo perfil posición y modo posición, basta con llamar al método “MotionInPosition”. Esta función tiene dos formas de llamarse. En el modo posición solo toma el valor deseado de la posición absoluta medida en encoders cuadratura. En el modo perfil posición puede tomar adicionalmente los valores deseados de velocidad y aceleración del perfil, además de especificar si el movimiento es absoluto o relativo.

Ejemplo de uso,

>> Motor.SetOperationMode( OperationModes.ProfilePositionMode );

>> Motor1.MotionInPosition( 132000 ); % para modo posición o perfil posición

>> Motor1.MotionInPosition( -132000, 4000, 6000, 1); % moverse a la posición -132000 con una velocidad de perfil de 4000 rpm y una aceleración de 6000 rpm/s, posición absoluta.

Para los modos “Velocity” y “ProfileVelocity”, el motor se mueve llamando a la función “MotionInVelocity”, en el caso del modo “Velocity” solo se especifica la velocidad deseada, para el caso “ProfileVelocity” se puede especificar la aceleración deseada.

Ejemplo de uso,

>> Motor1.SetOperationMode( OperationModes.ProfileVelocityMode );

>> Motor1. MotionInVelocity( 2000 ); % mover a 2000 rpm en el modo velocidad o perfil velocidad

>> Motor1.MotionInVelocity( 2000, 3000); % mover a 2000 rpm en modo perfil velocidad con aceleración de 3000 rpm/s.

El motor se detiene llamando al método “Stop”,

>> Motor1.Stop;

Para el modo corriente, el motor se mueve con la función “MotionInCurrent”, la cual acepta como parámetro de entrada el valor deseado de corriente del motor.

>> Motor1.SetOperationMode( OperationModes.CurrentMode );

>> MotionInCurrent( 2000 ); % mover hasta alcanzar una corriente de 2 Amperios

Una vez finalizado con el motor, se procede a destruir el objeto con un llamado a delete.

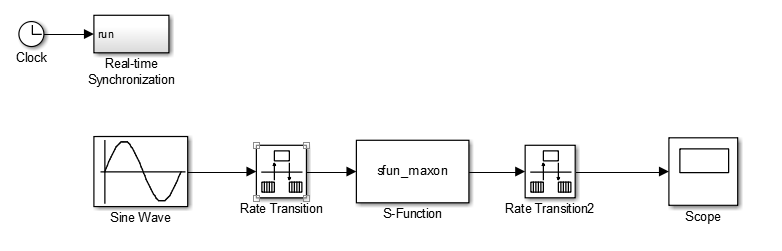
Ejemplo:

>> delete( Motor1 );

El destructor, delete, hace que se deshabilite el motor y se cierre la comunicación con el nodo.

**Comunicación desde Simulink.**

Para realizar la comunicación con simulink se creó la “s-function” llamada “sfun\_maxon.c” la cual se compila automáticamente cuando se llama a “Make”. El ejemplo de uso es el modelo llamado “testMaxon.slx”, el cual se reproduce en la figura a continuación.



El bloque “sfun\_maxon” acepta tres parámetros que son: NodeID, para denotar el identificador del nodo; DeltaT, para el muestreo con el puerto USB y Operational Mode, que es el modo de operación de trabajo del Epos2. Se aceptan los valores: 1 para modo corriente (OperationModes.CurrentMode), 2, para modo velocidad (OperationModes.VelocityMode) y 3 para modo posición (OperationModes.PositionMode). Solo se permiten estos modos básicos porque se supone que el control de alto nivel estará programado en simulink.

Una característica importante del ejemplo es la sincronización con el reloj de la máquina para crear un efecto de tiempo real. Si bien estrictamente hablando no es tiempo real, se puede dar la sensación de tiempo real haciendo que el tiempo de simulación de simulink no vaya más rápido que el reloj de la máquina. Esta sincronización se realiza llamando al bloque “RealTime Syncronization”.