Universidad de Oviedo

Trabajo Fin de Máster realizado por

Victoria Casares García

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

Modelado de Transelevador
Automatic Warehouse Studio (AWS)

Julio de 2018

ÍNDICE GENERAL

Universidad de Oviedo

Trabajo Fin de Máster realizado por

Victoria Casares García

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

MEMORIA

Modelado de Transelevador
Automatic Warehouse Studio (AWS)

Julio de 2018

ÍNDICE DE LA MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. - OBJETIVOS Y ALCANCE

La compañía Mecalux Software Solutions S.A. ha solicitado la elaboración del presente proyecto, el cual consiste en desarrollar el modelo de simulación de un transelevador[1] y su programa de control.

Por lo tanto, este trabajo tiene 2 objetivos:

1. Desarrollar el programa de control de un dispositivo transelevador sobre el entorno GALILEO[2], concretamente en la plataforma de desarrollo de programas de control Designer IV en el lenguaje (ST) Texto Estructurado del estándar internacional de programación para PLCs IEC61131. El programa de control deberá cubrir las funcionalidades habituales de este tipo de máquina, entre ellas:
    1.1. Capacidad de recibir órdenes de un sistema externo.
    1.2. Ejecución de movimiento a origen de la forma más óptima posible, controlando los 2 ejes de la máquina en este caso (X, Y) y gestionando el movimiento mediante una rampa en S (típica en este tipo de máquinas).
    1.3. Control de sensores y variadores.
2. Definir el modelo y programa de simulación de la máquina transelevador. El programa de simulación se realizará para la herramienta Automatic Warehouse Studio (AWS) en lenguaje C# combinado con JavaScript. AWS se trata de la plataforma de simulación del entorno GALILEO. El modelo y programa de simulación debe incluir:
    2.1. Representación 3D de los diferentes elementos que componen la máquina.
    2.2. Programa para modelar la electromecánica de la máquina. Especialmente importante el modelado del variador con posicionador que es el elemento que habitualmente controla el movimiento del transelevador en los ejes X e Y y que debe ser el encargado de llevar la máquina al destino.
    2.3. Programa para modelar resto de comportamientos y operativas de un transelevador.

El alcance de este trabajo está definido por los siguientes puntos:

1. Creación del programa de control simplificado de la máquina transelevador completamente operativo a todos los efectos. Incluyendo gestión de averías.

[1] Transelevador: son máquinas creadas para el almacenaje automático de pallets. Se desplazan a lo largo de los pasillos del almacén realizando las funciones de entrada, ubicación y salida de mercancías

[2] GALILEO: Solución integral que se presenta como la alternativa a los PLCs tradiciones. La principal característica de este sistema es que no se trata de un terminal físico concreto, sino que se trata de una herramienta software que corre bajo Windows y que realiza las labores de los PLCs tradicionales, es decir, se trata de un softPLC desarrollado por Mecalux. Es un sistema de control que funciona de coordinador general entre las diferentes plataformas de control, simulación, scada y comunicaciones que engloba.

2. Demostrar que recibiendo una orden realiza el movimiento indicado mediante los desplazamientos precisos y simulando una velocidad y aceleración realistas.
3. Creación del modelo y programa de simulación para representar comportamiento de un transelevador gráficamente.

A continuación se muestra una imagen que representa la herramienta GALILEO y las diferentes plataformas de desarrollo que engloba.

Estas herramientas se explican detalladamente en el apartado "*Herramientas y software de Mecalux*".

Figura 1.1.- *Herramienta Galileo y sus plataformas*

Por lo tanto, lo que se pretende con seguir con este proyecto es desarrollar para el softPLC de Galileo, tal como se puede ver en la siguiente figura:

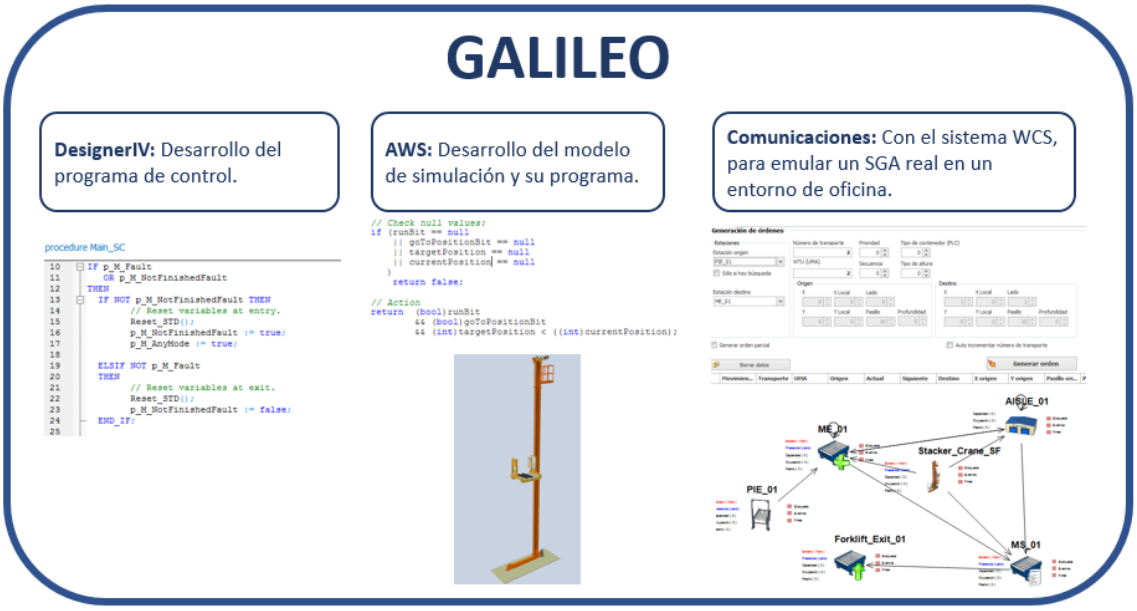


Figura 1.2.- *Objetivos del proyecto*

Con los desarrollos e implementaciones anteriores se pretende conseguir un modelo de transelevador operativo con su programa de control para operar en cualquier instalación:

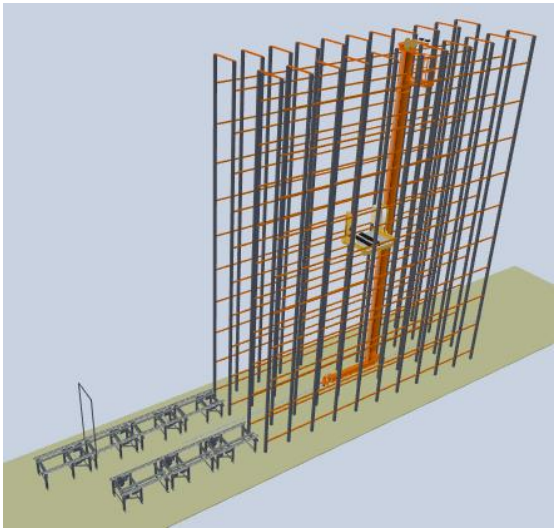


Figura 1.3.- *Ejemplo de instalación simulada con un transelevador.*

1.2. - CONCEPTOS, DEFINICIONES

El concepto principal para comprender este trabajo es el significado de almacén automatizado. Un almacén automatizado o almacén automático se trata de una estructura generalmente de gran altura, donde los elementos de almacenamiento y los elementos de manutención van integrados y controlados por un sistema informático. Los almacenes automatizados permiten gestionar y optimizar los procesos derivados del almacenaje, preparación y expedición de mercancías. (Se detalla más en profundidad todos los aspectos relevantes de los almacenes automatizados y las máquinas que pueden aparecer en ellos en el "ANEXO IV: Logística")

De cara a adentrarse en el tema que concierte al presente proyecto es necesario conceptualizar diversos términos. El primero de ellos es el de simulación que se define como el proceso de diseñar un modelo del sistema real y llevar a cabo experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias, dentro de los límites impuestos por un cierto criterio o un conjunto de ellos, para su funcionamiento. De esta manera al analizar en sí el concepto puede entenderse que una simulación de un modelo lleva consigo un planteamiento, colección de información, análisis, recursos, actividades, eventos, estrategias, conocimientos teóricos y prácticos que conducen a observar cómo influyen ciertos cambios en el modo de operar un sistema, lo cual, sirve para tomar decisiones correctas minimizando el riesgo y maximizando el rendimiento. A continuación, se definen varios términos relativos a este ámbito:

- Modelo: representación del sistema o proceso real, que involucra lógica, matemática, y estructura de cara a su aproximación a la realidad.
- Evento discreto: es aquel que se presenta a intervalos discretos de tiempo.
- Status de un modelo: en la práctica y en la teoría indica el estado del modelo o sistema a lo largo del tiempo en cualquier punto.
- Evento: ocurrencia instantánea que cambia el estado del modelo.
- Actividad: una actividad es una duración de tiempo, es iniciada cuando ocurre un evento y en el cambio de status del sistema.
- Entidad: Una entidad es un objeto en el modelo, que representan un objeto real en el sistema, tienen estándares establecidos, pero también tienen atributos personalizables para hacer para individualizarlas. Se les conoce como entidades dinámicas.
- Recurso: es una entidad no dinámica que provee un servicio a una entidad dinámica.
- Estación: Origen y/o destino de movimientos de contenedores en el almacén.
- Ruta: Trayectoria entre estaciones
- Tarea: Es un proceso para realizar una acción sobre el stock o contenedor dentro de un almacén. Para su ejecución se descompone en uno o varios movimientos.
- Movimiento: es una de las partes para la realización de una tarea. El movimiento puede implicar movimiento de stock o contenedor. En el caso de ser movimientos

que son enviados mediante órdenes al sistema de control (Galileo) implican el movimiento de contenedores.

- Orden: Una orden es un movimiento enviado a Galileo para su ejecución.
- Tracking; El tracking es la información del movimiento reflejada en Galileo. Esta información se "mueve" en memoria de forma que se utiliza la analogía de que dicha información se mueve asociada a un contenedor

Es importante diferenciar el concepto de simulación del término emulación. Se habla de emulación cuando un sistema imita el comportamiento de otro, aunque no necesariamente a la misma velocidad, es decir, un emulador imita la causa o el proceso mientras que un simulador pretende hacerlo sobre el aspecto o las variables "resultado" del sistema.

Aunque diferentes, simulación y emulación comparten diversos términos y procedimientos identificándose las principales diferencias en los resultados buscados con cada una de ellas. En simulación, los modelos son utilizados de cara a exhaustivos experimentos con el fin de probar y desarrollar diversas soluciones basadas en los requerimientos y restricciones del escenario en sí. En la mayoría de los casos, las pruebas que tienen lugar bajo la simulación tienen carácter temporal, esto es, el tiempo es el que determina la dinámica del sistema haciendo que los entes presentes en el escenario interaccionen en función del tiempo. Absolutamente todos los elementos que intervienen en la simulación se encuentran incluidos en este entorno virtual lo que hace posible que la interacción temporal entre ellos pueda acelerarse, presentándose la posibilidad de agilizar la evolución de los escenarios pudiendo obtener información de la respuesta del sistema en horizontes temporales distantes en tiempos cortos de simulación.

Esta última característica citada acerca de la simulación es una de las principales diferencias que ésta presenta frente a la emulación, en cuyo caso no todos los entes que intervienen en el escenario de prueba se encuentran embebidos en un entorno virtual en el que; no se opera en tiempo real. La emulación presenta escenarios en los que parte del entorno es virtual y parte es real y es sobre ésta última parte sobre la que se realizan las pruebas o experimento.

La emulación presenta como motivaciones:

- Someter al sistema a test de capacidad y rendimiento máximo. En emulación un sistema puede ser forzado a trabajar mucho más rápido de lo que lo haría en cualquier situación real dando pie a determinar en qué momento ese sistema se desborda. Esto no sería posible en escenario real ya que presenta limitaciones físicas en su funcionamiento y esto solo permitiría determinar la capacidad de los sistemas de control.
- Conocer las reacciones del sistema real ante imprevistos. La posibilidad de analizar el comportamiento de sistemas reales frente a situaciones improbables hace que se reduzca el riesgo de paradas no programas en instalaciones u otro tipo de incidencias.

Esta última característica deja a la luz una diferencia fundamental entre la simulación y la emulación en lo referente al periodo en el que se emplean:

- Simulación cobra importancia previa a la implantación de un sistema en una planta determinando así condiciones operativas y posibles modificaciones a realizar durante la fase de diseño;
- Emulación se presenta durante la fase post implantación previa a la validación final de la instalación o en cualquier momento de la vida funcional del sistema, obteniendo de ella información de diferente naturaleza.

1.2.1.- PRINCIPALES BENEFICIOS DE LA SIMULACIÓN

Una de las principales características de esta técnica y es la imitación de escenarios y dispositivos a los ojos de los sistemas de gestión que los controlan.

La disposición básica y habitual de una instalación industrial de control en el que una red de servidores situados en el nivel superior controla una red de PLCs que a su vez gestionan dispositivos, electromecánicos en este caso. La *figura 1.4.* muestra la posición que ocuparía dentro de esa instalación una simulación de dichos dispositivos; cabe destacar que el resto de los componentes del sistema no tendrían constancia de estar operando sobre un entorno no real.

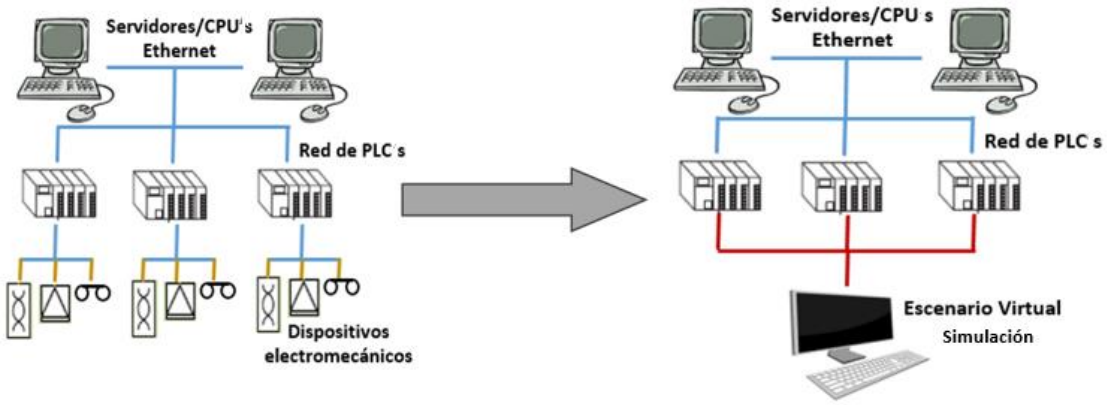


Figura 1.4.- *Sistema de control real Vs sistema de control con simulación.*

Observando los esquemas de la red de control y el lugar que ocupa el escenario virtual creado con objeto de simular los dispositivos electromecánicos a controlar es posible comprender el gran alcance de esta técnica, así como conocer esos tres aspectos principales en los que se fundamenta:

1. Someter al sistema a pruebas realistas de capacidad, rendimiento máximo, control de errores, así como las respuestas ante diversos escenarios operacionales… todo ello utilizando tanto sistemas de control reales previa a su instalación en el cliente o sobre sistemas que se encuentren funcionando sobre los cuales se precise obtener información de esta naturaleza como condiciones operativas habituales. Así, es posible disponer de un amplio espacio muestral de operación habitual del sistema.

2. Emplear técnicas de emulación en el análisis de sistemas atiende a la inevitable relación entre el incremento significativo de la posibilidad de error, fallo o desborde y la operación en periodos de pico de demanda o en aquellos no habituales que presentan un alto nivel de actividad. De acuerdo con estos casos, el escenario virtual es capaz de forzar al sistema a funcionar más rápido de lo que podría hacerlo realmente, así es posible descubrir cuellos de botella o puntos limitantes dentro del sistema.
3. Realizar pruebas sin riesgo útiles para el personal de formación y mantenimiento. De este modo, los operarios pueden adquirir la capacitación práctica de cualquier situación que pueda presentarse durante el desarrollo de sus tareas sin riesgo de daño sobre la instalación.

La necesidad de integrar sistemas automáticos a procesos de manipulación conlleva la implementación de software de control. Diversos módulos e instalaciones forman redes de comunicación con el fin de intercambiar información y excluir en la medida de lo posible la intervención humana de estos procesos.

En definitiva, realizar simulaciones, es un procedimiento empleado de cara a mejorar la fiabilidad de los sistemas y sacar a la luz fallos o identificar cuellos de botella de los procesos. Permitiendo así, examinar todos los parámetros y resultados del sistema. Esto conlleva a poder resolver fallos o dificultades observadas, previamente a que dé comienzo la ejecución y puesta en marcha del sistema real

1.2.2.- GESTIÓN Y SIMULACIÓN DE SISTEMAS DE ALMACENAMIENTO

Un campo de especial relevancia actualmente dentro del terreno de la simulación es la gestión de almacenes y el ámbito logístico. En este sentido, es importante destacar que los principales proveedores de sistemas de gestión de almacén e infraestructura de almacenamiento desarrollan sistemas capaces de validar sus instalaciones y sus operaciones en planta de cara a la minimización de riesgos funcionales.

En este sentido, es importante destacar la relevancia de las pruebas sobre sistemas de gestión de almacenamiento ya que debido a la responsabilidad que se les otorga dentro de la cadena de suministro y el volumen de información que se maneja con los mismos, la dimensión de las consecuencias derivadas de fallos en los sistemas puede ser catastrófico para el correcto funcionamiento de las empresas.

Algunos proveedores de proyectos llave en mano del sector de la gestión automática de almacenes y logística han desarrollado sistemas y protocolos de pruebas que incluyen simulación.

Las simulaciones referentes a este campo hacen posible los análisis del funcionamiento de diversos dispositivos, como transelevadores de almacenamiento y recuperación automáticos presentes en una planta de almacenamiento, transportadores, lanzaderas, vehículos láser guiados, elevadores.

El sistema de almacenamiento se compone de varios dispositivos electromecánicos controlados cada uno por una red de PLC’s que ejecuta órdenes provenientes del sistema

de gestión global del almacén, el Warehouse Management System (WMS), que controla por completo las operaciones que se llevan a cabo en la planta y que a su vez debe interpretar las señales provenientes de los PLCs de control de cada transelevador.

La simulación se inserta en este caso al final del sistema jerárquico de órdenes sustituyendo tanto la red de PLC's como los dispositivos electromecánicos presentes en el sistema. Las simulaciones deben presentar las características del almacén objeto de estudio en lo referente a capacidad de almacenamiento, velocidad de los dispositivos móviles, tiempos de carga y descarga, etc

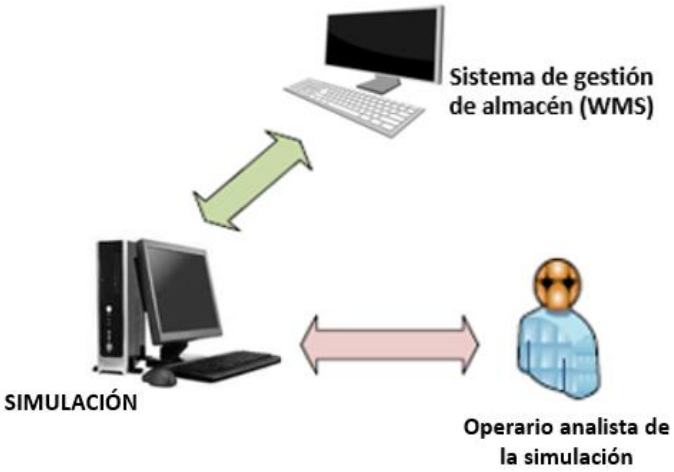


Figura 1.5.- *Red de simulación para sistemas de AS/RS*

Mediante la simulación de parte de la instalación de un sistema AS/RS es posible hacer funcionar al WMS sobre un entorno virtual cuyas respuestas ante órdenes se ajustan a la realidad operativa de la planta tal y como lo harían los dispositivos reales. De cara a test y pruebas es posible forzar al sistema de gestión a realizar operaciones sobre un entorno tan exigente como el analista haya determinado en la emulación, incluso por encima de lo que las limitaciones físicas presentes en el sistema real hubieran permitido

2. CONTEXTO DEL PROYECTO

2.1. - INTRODUCCIÓN A LA GESTIÓN DE ALMACENES. ESTADO ACTUAL

Actualmente la cadena de suministro se ha convertido en un medio para que las empresas aumenten su productividad y competitividad.

La gestión de almacenes es un eslabón tan importante como crítico dentro de la cadena de suministro debido a que se encarga de la administración del stock y, de forma intensiva, interviene directamente en la gestión de las necesidades de los clientes.

El desarrollo de la tecnología y en especial en el campo del tratamiento de la información ha contribuido en la implantación masiva de sistemas dentro de las compañías cuya concepción busca aumentar la eficiencia entre otros. Siendo así, la gestión y manejo de almacenes no es una excepción, de hecho, hoy en día las TIC (Tecnologías de la Información y Comunicaciones) son consideradas como una herramienta indispensable dentro de los medios empresariales y por supuesto dentro del campo de la organización industrial.

La gestión de almacenes es un elemento cuya finalidad pasa por homogeneizar y regular la interacción entre la oferta y la demanda, optimizar los costos de distribución y satisfacer los requerimientos de ciertos procesos productivos. La gestión de almacenes, por tanto, abarca desde los mecanismos destinados a proporcionar la materia prima al proceso de producción hasta la gestión del almacenamiento asociado al trabajo en proceso o WPI (Work In Process) dotando en la medida de lo posible de flexibilidad a los procesos productivos. La gestión de almacenes es a su vez el integrante más importante dentro de la gestión de producto terminado en cuanto a la preparación y el cumplimiento de entrega de los pedidos a los clientes. Es, por tanto, un elemento de especial relevancia dentro de la gestión de la cadena de suministro ya que influye directamente en el intercambio de bienes tanto con proveedores como con clientes.

2.1.1.- PRINCIPIOS Y OBJETIVOS DE LA GESTIÓN DE ALMACENES.

Minimizar	Maximizar
- El espacio destinado como camino hacia la mejora de la rentabilidad	- El potencial de almacenamiento
- Necesidades de inversión y costes de administración de inventarios	- La rotación de bienes y materias
- Los riesgos de personal y bienes físicos	- La atención a pedidos de clientes
- Pérdidas causadas por robos, incidencias y extravíos	- Disponibilidad de bienes para cualquier fin interno
- Las manipulaciones, los recorridos y movimientos de los empleados	- La operatividad y disponibilidad del almacén
- El número de equipos de manejo de los materiales	- La protección de los productos asegurando su integridad y duración

Tabla 2.1.- *Principios y objetivos de la gestión de almacenes*

2.2. - MECALUX SOFTWARE SOLUTIONS S.A.

2.2.1.- PRESENTACIÓN

Mecalux es una de las compañías punteras en el mercado de sistemas de almacenaje. Su actividad consiste en el diseño, fabricación, comercialización y prestación de servicios relacionados con las estanterías metálicas, almacenes automáticos y otras soluciones de almacenamiento. Compañía líder en España, se sitúa en el tercer puesto mundial en el ranking de su sector, con ventas en más de 70 países.

La primacía de la empresa se basa en el uso de las tecnologías más avanzadas de la industria y en una extensa red de distribución implantada en Alemania, Bélgica, Eslovaquia, España, Francia, Holanda, Italia, Reino Unido, República Checa, Polonia, Portugal, Argentina, Uruguay, Perú, Chile, Panamá, Brasil, México, Canadá, Turquía y Estados Unidos.

El Grupo dispone de once centros productivos: en España (Barcelona, Gijón y Palencia), Polonia (Gliwice), Estados Unidos (Chicago, Sumter y Pontiac), México (Tijuana y Matamoros), Brasil (São Paulo) y Argentina (Buenos Aires), que se encuentran ubicados estratégicamente para ofrecer un servicio rápido y ágil a sus mercados actuales y potenciales.

La apertura de nuevas delegaciones, la ampliación de las redes comerciales y de distribución, la dedicación de recursos a I+D+i, la división de almacenaje automatizado y el portal de logística Logismarket son las fuentes de crecimiento y desarrollo del Grupo Mecalux. Todo ello revierte, como siempre y desde hace más de 50 años, en un producto de calidad y en un servicio esmerado para sus clientes.

Los Principios Esenciales de Mecalux transmiten los objetivos y valores básicos que rigen el funcionamiento interno de las empresas del Grupo así como sus relaciones con sus empleados, clientes y proveedores

2.2.2.- ¿QUÉ Y CÓMO FABRICAN?

2.2.2.1.- PROCESO DE FABRICACIÓN. PROCESOS DE MÁXIMA CALIDAD

El Grupo Mecalux cuenta con once plantas de producción: tres situadas en España y las restantes en Argentina, Polonia, México, Estados Unidos y Brasil.

Las líneas de perfilados garantizan una producción continua de material, sometida siempre a los más estrictos controles de calidad. Asimismo, los procesos de soldadura automática que utiliza Mecalux son los más avanzados del mercado.

Todos los elementos de las estanterías están sujetos a exhaustivos ensayos de tracción y doblado de materiales, compresión, rigidez de uniones, resistencias y flexiones

2.2.2.2.- PROTECCIÓN GARANTIZADA

Los materiales son tratados mediante procesos automáticos de pintura, que aportan la protección adecuada. Asimismo, los elementos más expuestos al desgaste y a la corrosión son acabados mediante el proceso de cataforesis, que proporciona un grado de resistencia tres veces superior al pintado convencional. Este proceso de pintado ofrece ventajas importantes respecto a otros métodos tradicionales:

- Mayor resistencia a la corrosión.
- Capa de pintura uniforme.
- Comportamiento al fuego óptimo (temperatura de ignición superior a los 200º C).

2.2.2.3.- SOLUCIONES A MEDIDA

En el momento en que un cliente nos solicita la instalación de un sistema de almacenaje, Mecalux emplea un programa informático único en el mercado que determina, tras la introducción de distintas variables, el sistema más adecuado para lograr la máxima optimización del almacén.

Todos los procesos de cálculo, ensayos y diseño de estanterías metálicas en Mecalux cumplen con la normativa internacional FEM, cuya exigencia respecto a la calidad y seguridad en los productos se traduce en garantías de eficacia para sus clientes clientes.

2.2.3.- EVOLUCIÓN E HISTORIA DE LA COMPAÑÍA

Mecalux es una compañía multinacional de sistema de almacenaje, con sede en Gijón, creada para aplicar la experiencia en el diseño y desarrollo de soluciones de intralogística al Software de Gestión de Almacenes (SGA) Easy WMS. Su actividad está centrada en el diseño, fabricación, comercialización y prestación de servicios relacionados con las estanterías metálicas, almacenes automáticos y otras soluciones de almacenamiento. El resultado es una gestión integrada de la logística en todo tipo de empresas.

1966 – 1980: Nacimiento de Mecalux

- José Luis Carrillo, actual presidente de Mecalux, crea en Barcelona un pequeño taller de 200 m² para fabricar estantería ligera.
- Se establece la red comercial en todo el territorio español.
- La gama de productos se amplía con los sistemas de paletización.

1980 – 1990: **Desarrollo de la capacidad productiva**

- Apertura en Paris de la primera oficina en el extranjero
- Inauguración de oficinas en Reino Unido y Alemania
- Establecimiento de un centro productivo en Argentina (6000 m²)

- Ampliación de la gama de productos con los almacenes autoportantes[3]

1990 – 2000: Expansión internacional

- Salida de la compañía a Bolsa.
- Nuevo almacén logístico en Barcelona de 15.000 m².
- Obtención del certificado ISO 14001 de gestión medioambiental.
- Fuerte expansión internacional: apertura de oficinas en Francia, Reino Unido, Portugal, Chile, México, Estados Unidos, Italia, Polonia, República Checa y Alemania.
- Lanzamiento de la revista de logística Mecalux News.
- Obtención de la certificación de calidad ISO 9001.
- Desarrollo de la primera fase de la fábrica de Tijuana, México, de 30.000 m²

2000 – Actualidad: Integración de los servicios logísticos

- Apertura de nuevas delegaciones en Bélgica, Austria, Polonia, República Checa, Brasil, Singapur y Eslovaquia.
- Entrada en funcionamiento de los nuevos centros productivos en Estados Unidos y Polonia.
- Adquisición de TKINSA (división de almacenes automáticos de Thyssen Krupp) y de Esmena.
- Ampliación de las fábricas de Polonia y Chicago.
- Consolidación de la nueva planta de producción de galvanizados y carpintería en Cornellà.
- Automatización del nuevo centro de almacenaje y expediciones en Cornellà.
- Convenio Mecalux-IESE para la creación del CIIL (Centro Internacional de Investigación Logística).
- Creación del directorio industrial Logismarket.
- Incorporación de la planta productiva de Esmena en São Paulo, de 27.000 m² de superficie.
- Adquisición de activos de la compañía americana UFC Interlake Holding Co. En Estados Unidos y México.
- Lanzamiento del Clasimat, almacén automático vertical, y del Spinblock, carrusel horizontal automático.
- Lanzamiento de la nueva versión del EasyWMS (SGA[4]). Versión demo con funciones básicas descargable gratuitamente durante 90 días.

[3] Almacenes autoportantes: almacenaje a gran altura con el máximo aprovechamiento de la superficie disponible.
[4] SGA: Sistema de gestión de almacenes, es la definición atribuida a programas informáticos destinados a gestionar la operativa de un almacén

- Renovación de la imagen corporativa de la compañía.
- Entrada en funcionamiento de la división de robótica dedicada a la fabricación de sistemas de almacenaje automatizado.
- Lanzamiento de la revista industrial Logismarket.
- Nuevo Centro Tecnológico de Gijón para el desarrollo tecnológico de Software de gestión de almacenes EasyWMS.
- Ampliación de la gama de productos con el nuevo Movirack

Mecalux, gracias a su ambicioso plan de crecimiento, es hoy la tercera compañía de su sector en el mundo. La empresa ha consolidado su estrategia como proveedor integral de soluciones y tecnologías de almacenaje consiguiendo colocarse en la vanguardia de la industria y ofreciendo soluciones que evolucionan de acuerdo con las necesidades de sus clientes.

2.2.4.- HERRAMIENTAS Y SOFTWARE DE MECALUX

Mecalux ha creado diferentes herramientas para tener una solución específica para cada necesidad, por ello se trata de una de las empresas líder en el sector del almacenaje. Desarrolla productos adaptables para dar siempre respuesta a todos los requerimientos.

Estas soluciones desarrolladas por Mecalux facilitan a las empresas una gestión integral de su logística, optimizando la gestión física y documental del flujo de productos, desde su entrada en el almacén hasta su entrega en destino. Y todo ello, garantizando su trazabilidad. Presentan una interfaz muy intuitiva y fácil de usar que facilita el acceso directo a la información, presentando una alta homogeneidad de las pantallas y un cuidadoso diseño de los componentes gráficos.

Las soluciones software de Mecalux se emplean para el desarrollo, análisis, implementación y simulación/emulación, tanto de almacenes automáticos, como almacenes manuales. A continuación, se describen brevemente las herramientas desarrolladas por Mecalux que se emplearán en la definición y desarrollo del presente proyecto

2.2.4.1.- SISTEMA GALILEO

Es el software de control utilizado para gobernar los equipos electromecánicos de las instalaciones que llevan a cabo las tareas de transporte y almacenaje de la mercancía, siguiendo las instrucciones recibidas desde el software de gestión de la instalación.

El sistema de control de Galileo es un softPLC que funciona como un coordinador general. Se encarga del manejo de grafos, del control de componentes hardware, la gestión de logs, las comunicaciones con el Sistema de Gestión del Almacén... Es decir, engloba todas las tareas necesarias para recibir órdenes desde el SGA, enviar las señales correspondientes a las máquinas y, así, mover los contenedores de la instalación.

El sistema integra además distintas capas y aplicaciones:

- Entorno de Desarrollo Designer: Entorno integrado de desarrollo del Sistema de Control de Galileo. Desde esta aplicación, el usuario puede programar la lógica de todas las máquinas de la instalación y realizar la configuración de las comunicaciones con los elementos hardware y con el Sistema de Gestión del Almacén.

  Se trata de un entorno intuitivo, rápido en todas aquellas tareas repetitivas que implican la automatización de muchas máquinas y, sobre todo, robusto ante fallos en la programación o acciones realizadas por el usuario.

- Scada Galileo Status Monitor: Galileo Status Monitor es el Supervisor Control And Data Acquisition (Scada) del Sistema de Control de Galileo. Desde esta visualización, el operario interactúa con la instalación. En función de los permisos que tenga asignados, éste puede, desde activar operativas o movimientos manuales, hasta tener un control de las averías o eventos del almacén. Se trata de un sistema Scada sumamente flexible, con la ventaja de que permite visualizaciones totalmente personalizadas para cada cliente.

- Comunicaciones con el SGA: El Agente de transportes es el módulo encargado de realizar la comunicación con el Sistema de Gestión de Almacén. Su misión es gestionar las búsquedas de órdenes que las máquinas de la instalación solicitan e informar al SGA en cuanto esas órdenes hayan finalizado, así como del estado de los distintos movimientos, estaciones y rutas.

Actualmente este interfaz soporta comunicaciones TCP/IP. De esta forma es posible realizar la conexión con Sistemas de Gestión de Almacén de terceros de forma más sencilla.

Puntos clave del sistema de control Galileo:

- Independiente y compatible con distintos Software de Gestión de Almacenes (SGA).
- Fácil de ampliar gracias a su alto nivel de estandarización.
- Suave curva de aprendizaje.
- Sistema de visualización (Scada) incorporado.
- Rápida solución de incidencias en la robótica de la instalación.
- Uso de módulos ya programados —en continuo desarrollo— para diversos componentes hardware que simplifican notablemente la programación del proyecto

El "*Manual de usuario de Galileo*" incorpora más detalles e información de este software

2.2.4.2.- DESIGNER IV

El entorno de desarrollo Designer IV, como ya se comentó en el apartado anterior, consiste en un entorno integrado de desarrollo del Sistema de Control Galileo. Se trata por tanto de la herramienta que se utiliza para desarrollar aplicaciones sobre Galileo.

Designer IV cumple dos roles principales:

- Servir de entorno integrado de desarrollo (IDE), de manera que, desde la misma aplicación, el programador puede definir la lógica del programa, las variables a utilizar, crear las visualizaciones que los operarios utilizarán y, además, utilizar las herramientas de depuración previstas, de manera que puede depurar y encontrar los errores del sistema.

- Actuar como sistema SCADA, que se comunica con el sistema de control, y muestra la visualización creada por el programador, interactúa con los operarios y envía y recibe los datos hacía y desde el sistema de control.

Para realizar el programa de control y simulación de la máquina transelevador se ha empleado el software Designer IV, las funcionalidades que ofrece este entorno para el desarrollo de modelos de simulación son:

- Entorno colaborativo, esto es un entorno en el que se puede trabajar de forma remota sobre un proyecto único. De esta forma, cada usuario de la aplicación realiza consultas y modificaciones sobre un proyecto ubicado en un servidor.
- Posibilidad de diseño de layout de la instalación sobre su propia plataforma o importación de un layout desde el entorno de EasyS
- Posibilidad de comunicación con herramientas externas de forma bidireccional mediante librerías propias del software, además junto con Galileo forma un sistema extensible, es decir, permite añadir librerías de enlace dinámico con nuevos componentes en cualquier momento. Especialmente es relevante que estas comunicaciones puedan tener lugar durante la ejecución de las simulaciones/emulaciones.
- Capacidad de implementación de tareas mediante programación rutinas lógicas y funciones.
- Capacidad de operar en tiempo real (sistema SCADA)

El "*Manual de usuario de Designer IV*" incorpora más detalles e información de este software

2.2.4.3.- AUTOMATIC WAREHOUSE STUDIO (AWS)

Se trata de un nuevo producto de la compañía, que facilita el trabajo de desarrollo en tanto, que permite simular una instalación logística completa en oficina a nivel electromecánico y de control, sin la necesidad de desplazarse a obra más que para realizar pequeños ajustes. Actualmente, tras varias actualizaciones, AWS se trata de una

aplicación que se encuentra dentro de la plataforma de desarrollo DesignerIV. Entre sus principales características cabe destacar que AWS:

- Sirve como plataforma de simulación y emulación para poder configurar y probar y verificar el correcto funcionamiento de la instalación antes de llevarla a implantar en un entorno real.

- Al proporcionar un escenario de pruebas real, el tiempo de puesta en marcha del proyecto se acortará notablemente, para beneficio del cliente.

- Tareas como la creación de variables, esclavos de bus, etc., ahora se han automatizado, con la consiguiente reducción del tiempo de desarrollo

- Incorpora la posibilidad de programar utilizando el lenguaje estándar internacional de programación de PLCs vigente, el IEC-61131-ST y generar el programa de control para PLC's de la familia SIMATIC S7-1500/1200 y Rockwell ControlLogix.

- Admite un segundo lenguaje de programación estándar aparte de Xana (propiedad de MECALUX)

- Para realizar el proyecto no se parte de un documento (plano de la instalación), sino de un layout de EasyS[5] y desde él y mediante un asistente, se genera todo el esqueleto, incluida la visualización en 3D

2.2.4.4.- EASYWCS

EasyWCS es un Sistema de Control de Almacén capaz de funcionar como un coordinador general que se sitúa entre el programa de control y un software de gestión de alto nivel (ERP o SGA). EasyWCS se comunica, vía base de datos, con el Sistema de Control de Transporte para darle las órdenes de movimiento.

En cuanto a sus características principales:

- Es capaz de comunicar MySQL, SQLServer y base de datos Oracle con sistemas externos.

- Realizar la gestión de tareas mediante preselecciones estándar. Particularización por parte del implementador de casos especiales según necesidades del cliente.

- Puede conectar con PLC o PC con Galileo.

[5] EasyS: Software propio de la compañía Mecalux. Es una herramienta integrada para configuración, diseño gráfico de un almacén fácil y rápidamente. El "Manual de usuario de EasyS" incorpora más detalles e información de este software

- Ofrece la potencialidad de una interfaz gráfica de usuario (GUI): Sistema de supervisión en tiempo real de las estaciones de la instalación. Facilidad de configuración de estaciones y rutas. Visor de tramas y tiempos de ejecución implicados en la comunicación. Logs y diagnosis.

- Integra herramientas para simular las órdenes y validar el sistema y, de esta manera, minimizar los tiempos y los errores y, por otro lado, facilitar las pruebas de las distintas partes que componen la instalación.

El "*Manual de usuario de EasyWCS*" incorpora más detalles e información de este software

3. TRABAJO FIN DE MÁSTER Y MECALUX

El proyecto que la compañía Mecalux ha solicitado llevar a cabo consiste en el desarrollo de un modelo de simulación de un transelevador, más concretamente, solicita la realización de la simulación de la máquina transelevador para la herramienta Automatic Warehouse Studio (AWS) y su programa de control.

El sistema AWS está pensado para poder desarrollar las diferentes máquinas existentes en un almacén automático en el lenguaje estándar de programación de PLCs, en IEC 61131 (se trata del estándar utilizado en la mayoría de las grandes marcas de autómatas programables, como SIEMENS o ROCKWELL).

Dado que ningún fabricante de autómatas programables cumple la norma de manera íntegra, el desarrollo se realizará en DesignerIV (entorno de desarrollo de Mecalux) y este entorno adaptará y generará el programa para cualquiera de los siguientes destinos:

- PLC Rockwell
- PLC Siemens
- Sistema Galileo

El código del proyecto solicitado será creado para el SoftPLC Galileo.

Para su implementación se utilizará el lenguaje IEC 61131 ST (Structured Text) ya que este tipo de lenguaje permite dar un paso más allá en el campo de la programación de PLCs por admitir una fácil adaptación a otras marcas (que no cumplan el estándar).

Se eligió ST por tratarse de un lenguaje literal/textual de alto nivel que permite programación estructurada (división de tareas), facilitando la programación de procesos que requieren instrucciones complejas o grandes cálculos.

3.1. - DESCRIPCIÓN Y NECESIDADES

En este apartado se aborda el entorno del proyecto.

Puede clasificarse el proceso de identificación de necesidades, como uno de los pilares básicos sobre los que se fundamenta la caracterización del presente proyecto.

Las principales necesidades que se identifican están estrechamente ligados a los objetivos que el sistema a desarrollar pretende cubrir. En este sentido, las necesidades a cubrir derivan de la pretensión constante de optimización de procesos, más concretamente de procesos altamente automatizados en los que se destacan tanto el tiempo de funcionamiento de forma autónoma como la gran cantidad de información que procesan y que define su comportamiento y actuaciones.

Se identifica, a su vez, dentro del sector industrial la necesidad de realizar sobre las instalaciones aquellas pruebas y test que aporten la manera de reducir tanto los riesgos operativos como los eslabones ineficientes y limitantes de los procesos.

En este panorama se identifica la necesidad de poder optimizar los procesos de almacenamiento y en concreto los automáticos reduciendo sus incidencias e identificando sus cuellos de botella. Se decide abordar estas tareas desde el campo de la emulación.

3.1.1.- NECESIDADES Y REQUISITOS POR PARTE DE LA EMPRESA MECALUX S.S.

Hasta ahora, las máquinas no simuladas ni implementadas mediante AWS han sido los transelevadores, por ser muy complejas, debido a que son dispositivos que poseen multitud de señales. Por ello, si en una instalación había presencia de estas máquinas, para el sistema AWS, cuando las mercancías llegaban a este punto, desaparecían dejando una zona de la instalación sin simular o emular, es decir, la simulación o emulación de la instalación finalizaba en la mesa de entrada al almacén (mesa de entrada a estantería) y comenzaba en la mesa de salida de almacén (mesa de salida desde la estantería).

Es decir, lo que se veía a nivel de simulación en AWS:

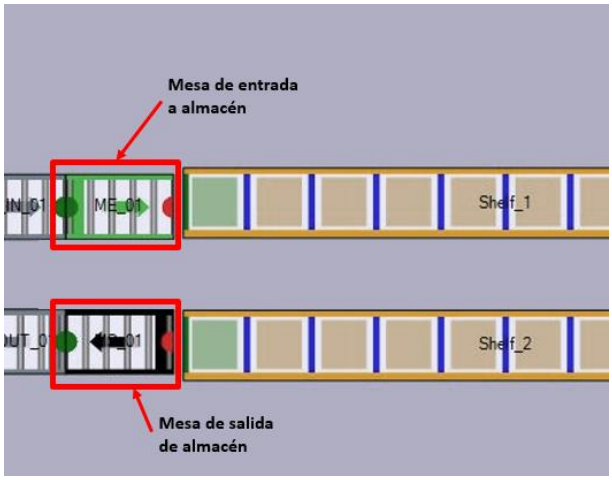


Figura 3.1.- *Representación de un almacén automático previo a la creación del modelo transelevador*

Lo que quiere Mecalux visualizar en sus simulaciones es la máquina transelevador para ver como interactúa entre Mesa de Entrada- Estantería; Estantería-Mesa de Salida; Mesa de Entrada-Mesa de Salida

Por lo tanto, sin estas máquinas, los transelevadores, era difícil poder validar una instalación completa, por la falta de estos elementos, que en muchas ocasiones pueden ser motivo de incumplimiento de ciclos en de instalaciones si no están correctamente diseñadas (los traslos pueden generar cuellos de botella por no poder absorber todo el flujo de contenedores de entrada o salida de almacén, pueden estar configurados a velocidades no adecuadas y que no cumplan los ciclos exigidos por el cliente…) .

Los transelevadores/traslos (*o stacker crane - en inglés*) no se habían implementado hasta ahora por no disponer de un grafo referencia, además en IEC no queremos desarrollar grafos, ya que no se usa en el estándar de programación de autómatas.

Estas máquinas "transelevadores", desde un punto de vista de usuario de una aplicación, gusta verlas como máquinas cerradas ("cajas negras"), por tratarse de máquinas muy complejas en cuanto al número de señales que reciben y transmiten debido a la multitud de sensores y variadores de los que disponen en la realidad, por ello hasta ahora no era viable simular o emular una máquina de estas características y mucho menos aún, su interacción con el resto de la instalación.

Conociendo todo esto, se llegó a la conclusión de que:

- Los clientes requieren ver el proceso completo de funcionamiento de su instalación, y así poder validar diferentes diseños y estructuras de un almacén, seleccionando la más ventajosa para el desarrollo de su actividad.
- La gente que trabaja en las oficinas de Mecalux necesitaban poder simular/emular estas máquinas con los movimientos y velocidades reales, para poder conocer los problemas que podrían surgir y así prevenirlos o plantear posibles soluciones.
- Conocer ciclos y tener una visualización del proceso completo ayudará a la empresa a mostrar una herramienta más completa al cliente, facilitando la tarea de ventas.

Por todo lo descrito anteriormente, se ideó diseñar un modelo de simulación del transelevador con un número limitado/simplificado de señales que permita visualizar en el proceso de interacción con la maquina previa ME (mesa de entrada al almacén) y la estación siguiente que será la estantería/ubicación o en caso de funcionalidad crossdocking[6] con la MS (Mesa de salida del almacén)

Lo que solicita Mecalux desarrollar en este proyecto para visualizar en AWS es:

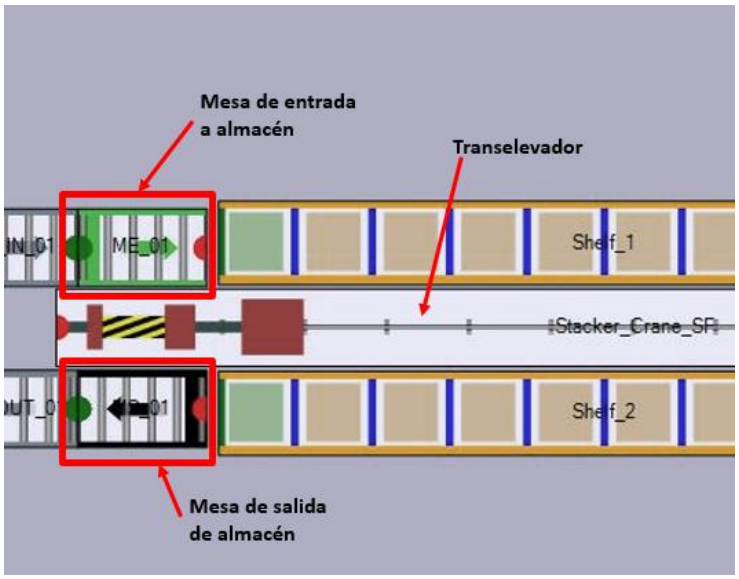


Figura 3.2.- *Representación del almacén automático con el modelo de transelevador*

[6] Crossdocking: operativa o tarea de una instalación de almacenaje en la cual, la mercadería recibida por un depósito o centro de distribución no es almacenada, sino preparada inmediatamente para su próximo envío. Es decir, que la mercadería no hace stock ni ningún otro tipo de almacenaje intermedio.

En otros términos, las operativas, no implementadas hasta ahora en AWS y que se requieren visualizar en una simulación son:

1. Recogida de contenedores de cabecera en la llamada mesa de entrada y ubicación en estanterías del almacén (ME-AISLE)

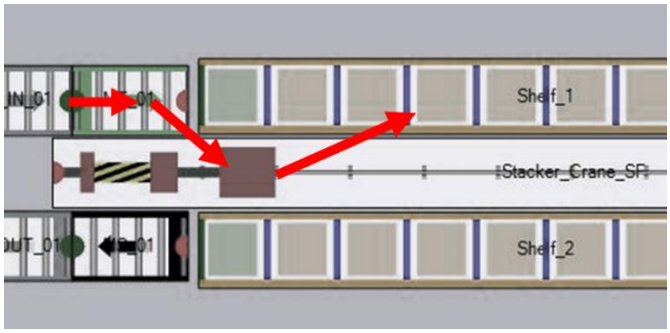


Figura 3.3.- *Tarea de ME – StackerCrane - AISLE*

2. Recogida de contenedores de estanterías del almacén y su extracción y deposición en la mesa de salida de la cabecera (AISLE-MS)

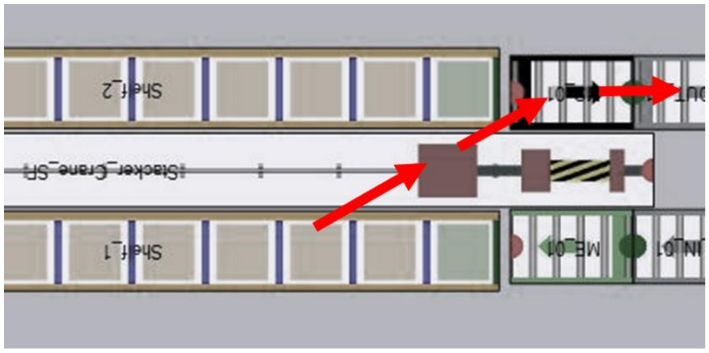


Figura 3.4.- *Tarea AISLE –StackerCrane - MS*

3. Recogida de contenedores de la cabecera, mesa de entrada a la zona de almacenaje y su movimiento directo hacia mesa de salida de la cabecera (ME-MS). Esta tarea define la conocida operativa crossdocking.

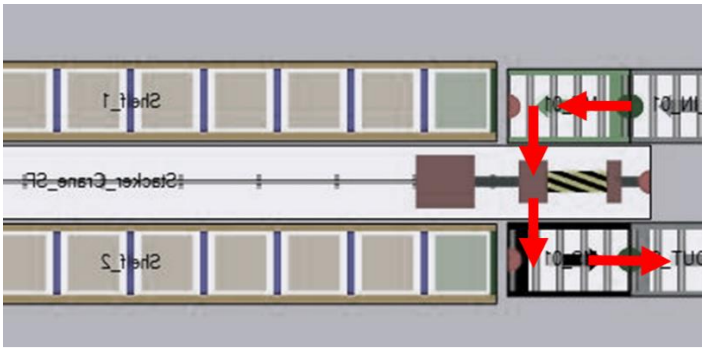


Figura 3.5.- *Tarea ME – StackerCrane – MS*

3.1.2.- DESCRIPCIÓN TECNICA DEL SISTEMA Y AJUSTE A OBJETIVOS.

Como objetivo principal de este proyecto se presenta el desarrollo de un sistema, así como un programa de control que permita realizar tareas de emulación con transelevadores en instalaciones tipo ASRS. Para ello, debemos diferenciar los objetivos generales y los específicos de la herramienta a emplear y ajustarnos a ellos

3.1.2.1.- OBJETIVOS GENERALES.

- Identificar limitaciones y posibilidades del simulador: Instruirse en el manejo de las herramientas necesarias para el desarrollo del modelo de simulación EasyS, DesignerIV, GalileoIV, AWS, EasyWCS identificando claramente las limitaciones que presenten, así como las posibilidades reales de adaptación a su uso como emulador.
- Conocer los factores determinantes presentes en sistemas de gestión y elementos mecánicos de gestión automática de almacenes: Abarca tanto la indagación en el funcionamiento y alcance de los sistemas de gestión como de los dispositivos presentes en este tipo de tecnología
- Desarrollo de entorno para emulación bajo AWS y Designer IV: Se pretende implementar un nuevo modelo de simulación, más concretamente un transelevador, con su sistema de control, capaz de comunicarse con la capa de transporte de un SGA (sistema de gestión de almacenes), a fin de hacer creer a este que lleva a cabo su operación sobre un entorno real. El objetivo abarca diversas tecnologías de gestión de almacenes y logística. El alcance de este proyecto se centra en los transelevadores.
- Interacción del entorno de emulación creado en Designer IV con la capa de transporte de un SGA, es decir con EasyWCS: Tras el desarrollo de un modelo de transelevador con su gestión a través de un programa de control, parte que concierne al presente proyecto, se pretende que exista comunicación con la capa de transporte del SGA para poder realizar la emulación.
- Validación de entorno: Se diseñará un escenario de pruebas orientadas a evaluar cada aspecto funcional del sistema completo. Una vez corregidos los errores y fallos funcionales que puedan detectarse, queda validado el sistema. Las pruebas se

presentan con un objetivo principal, el de evaluar la adecuación del sistema a la lógica y comportamiento que se le presupone.

3.1.2.2.- OBJETIVOS ESPECÍFICOS DE LA HERRAMIENTA DESIGNER IV

- Creación del modelo de transelevador en Designer IV: Empleando como plataforma de desarrollo una versión de administrador de Designer IV, se implementará un modelo que imite el comportamiento de un transelevador con una simplificación moderada de todos los dispositivos electromecánicos que lleva integrados.
- Diseño de sintaxis de comunicaciones: El modelo debe ser capaz de interactuar en tiempo real con aplicaciones externas atendiendo a órdenes y ofreciendo respuestas. Estas comunicaciones se realizarán en base a unos mensajes que integran la sintaxis de comunicaciones.
- Configuración del sistema de emulación en las plataformas de Designer IV y EasyWCS para realizar la comunicación entre modelo de emulación y SGA: Las pruebas sobre Designer (simulador/emulador/scada) empleado como emulador se realizan desde un pseudosistema de gestión que hace de capa de transporte de un WMS, es decir, se empleará un software llamado EasyWCS que se encargará de enviar y recibir señales, órdenes e información del sistema emulado tras haberlo configurado en su plataforma.
- Los elementos empleados en el entorno de emulación, deben estar configurados con los mismos valores e identificadores en las plataformas de DesignerIV y EasyWCS, para que estos se relacionen y conecten para recibir y ejecutar órdenes e intercambiar información.

4. ANÁLISIS Y DESARROLLO DEL PROGRAMA DE CONTROL

Para el desarrollo del modelo de simulación de la máquina transelevador es necesario diseñar un programa de control de la misma, para que con las señales que reciba el programa de control, este ejecute y active diferentes señales que permitirán visualizar la evolución de la simulación en función de las condiciones que ocurran en un momento dado.

4.1. - LÓGICA DEL MODELO TRANSELEVADOR

El diseño de la lógica del modelo pasa por su ajuste a las funcionalidades del simulador, es por ello su concepción sea posterior al análisis del alcance de la herramienta DesignerIV.

De este modo mediante el análisis y estudios entre las acciones y eventos del simulador y un diagrama de estados del transelevador, se define tanto la lógica que tiene lugar en los diferentes sucesos dentro de la simulación como los estados y transiciones de dicho modelo. El siguiente diagrama muestra de manera simplificada la operativa, en un modo de funcionamiento normal, de un transelevador.

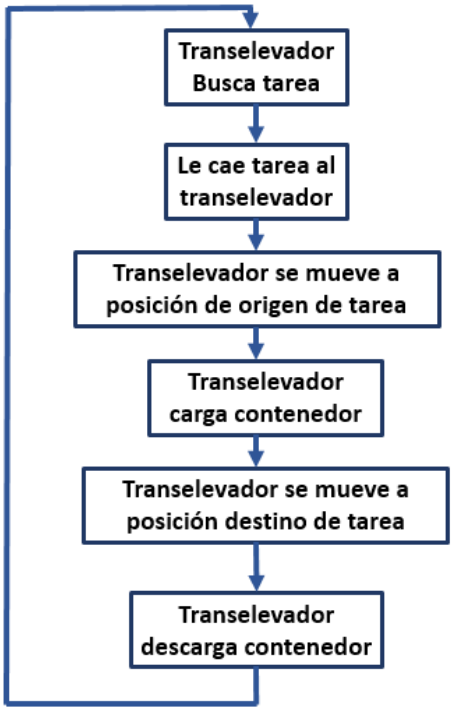


Figura 4.1.- *Operativa de funcionamiento normal de un transelevador*

4.2. - MODOS DE FUNCIONAMIENTO DEL TRANSELEVADOR

4.2.1.- MODO SELECCIÓN

El usuario debe ser capaz de elegir el modo de funcionamiento que desea para la emulación de la máquina.

La programación que rige cómo y cuándo se entra en un modo de funcionamiento u otro y cómo y cuándo se sale es de vital importancia para el correcto funcionamiento de la automatización. Esta tarea de selección del modo de funcionamiento viene dada por el valor que se le asigne a una determinada variable de tipo entero llamada "Modo_Funcionamiento". Se puede encontrar descrita en el "ANEXO I: Variables y parámetros".

Dentro de este programa se realiza una división jerárquica, dada por los valores asignados a unas determinadas variables "Modo_Funcionamiento". Para acceder a cada uno de los diferentes tipos de funcionamiento se ha de activar una variable booleana del mismo nombre: automático, manual y emergencia; cuyo valor se da desde el panel de control de DesignerIV llamado "**inspector**". Se procede a analizar cada uno de estos modos de funcionamiento en detalle:

4.2.1.1.- MODO AUTOMÁTICO

En modo automático el transelevador tienen un comportamiento secuencial, es decir, su comportamiento está definido por diferentes etapas que van evolucionando según determinadas condiciones que reportan los sensores. La máquina parte del reposo, donde no está haciendo ningún movimiento y está a la espera de entrar en funcionamiento, donde realiza su función específica para finalmente volver al reposo hasta que se vuelva a iniciar de nuevo el ciclo.

En el modo automático el transelevador evolucionará de manera autónoma.

Figura 4.2.- *Modo de funcionamiento automático del traslo*

El modo automático también se conoce como modo "sin hombre a bordo", ejecuta las ordenes enviadas mediante una fotocélula de comunicación desde el ordenador de gestión de transportes. En este modo se ejecutan las siguientes operaciones:

- Ubicación
- Extracción
- Cambio de ubicación
- Corrección de errores en el almacén
- Autoaprendizaje de las ubicaciones del almacén

4.2.1.2.- MODO MANUAL

Es posible realizar el movimiento manual de los diferentes conjuntos de movimientos del transelevador. Para hacer uso del movimiento manual, se debe activar la variable asociada a este tipo de funcionamiento, así se inhiben otros modos de funcionamiento.

Estos movimientos manuales se realizarán activando una variable (desde el panel de **inspector**) que ejecutará paso a paso cada grupo de movimientos, es decir, por seguridad, solo se ejecutará un grupo de movimientos cada vez que se active la variable.

NOTA: Los movimientos manuales solo se deben ejecutar cuando no hay pallets encima del transelevador. Si se realizan movimientos manuales con pallets encima se corre el riesgo de producir averías, caída de contenedores, decalaje de pallets, etc.

Este modo también es llamado modo "con hombre a bordo" y permite manipular todos los elementos del transelevador de forma restringida para llevar a cabo tareas de mantenimiento y reparación

4.2.1.3.- MODO SEMIAUTOMÁTICO

Los transelevadores permiten realizar movimientos semiautomáticos, que permiten mover la máquina a posiciones lógicas del almacén específicas. Así, es posible resolver incidencias (como por ejemplo recolocar pallets que estuviesen mal orientados) o probar movimientos.

Este modo de funcionamiento se utiliza para realizar funciones de apoyo como son:

- Acceso automático a una ubicación, posicionando el transelevador automáticamente en el emplazamiento demandado por el operario
- Ciclo de horquillas automático: extrae o deposita automáticamente una unidad de carga en la dirección indicada por el operario
- Reubicaciones de mercancía

4.3. - CODIFICACIÓN DE LAS POSICIONES DE ALMACENAMIENTO

Las posiciones de almacenamiento, estanterías y posiciones o ubicaciones sobre las que opera el transelevador, se referencian mediante 2 sistemas de coordenadas.

Un sistema de coordenadas es una representación del espacio plano basado en un sistema Cartesiano. Se trata de un conjunto de valores y puntos que permiten definir unívocamente la posición de cualquier punto de un espacio.

Para este proyecto, se emplean dos tipos de sistemas de coordenadas:

- Coordenadas de la simulación/emulación: Están definidas dentro de un espacio lógico, es decir, el espacio donde se utilizan las unidades lógicas, que serán las unidades virtuales con las que se definirán las ubicaciones y posiciones de los distintos elementos de la instalación simulada.
- Coordenadas físicas: Están definidas dentro de un espacio euclídeo tridimensional. Estas coordenadas serán empleadas dentro del sistema para acceder a los valores reales que proporcionan los sensores y telémetros.

Internamente, ambos sistemas de coordenadas estarán relacionados para poder relacionar la visualización de la emulación con el programa de control.

4.3.1.- CONTROL DE POSICIÓN

Para realizar el control de posición del conjunto transelevador, de la cuna y de los extractores la simulación se cuenta con telémetros y sensores inductivos. Se trata de parámetros/funciones configurables con el fin de simular lecturas de valores reales.

El transelevador estará definido en un espacio euclideo, cada una de sus posiciones estará mapeada con unas coordenadas lógicas referenciadas a unas coordenadas reales.

Un ejemplo de estos valores es:

```
Dimensions = 3
Offsets = 0:0:0
(1:0:1) = 16000:1175:600
(1:0:2) = 16000:1175:-600
(2:1:1) = 14650:900:600
(2:1:2) = 14650:900:-600
(2:2:1) = 14650:2400:600
(2:2:2) = 14650:2400:-600
(2:3:1) = 14650:3900:600
(2:3:2) = 14650:3900:-600
(2:4:1) = 14650:5400:600
(2:4:2) = 14650:5400:-600
(2:5:1) = 14650:6900:600
(2:5:2) = 14650:6900:-600
```

Figura 4.3.- *Equivalencias de coordenadas lógicas a reales*

Los valores de las posiciones que adopta el transelevador en el eje_X estarán medidas por un telemetro, al igual que las posiciones de la cuna a lo largo del eje_Y.

Las posiciones de los extractores en el eje_Z solo estarán controladas por unos sensores inductivos y chapas metálicas colocadas en los extremos de los extractores que nos darán los valores de las señales TODO/NADA.

4.4. - INFORMACIÓN CONTENIDA EN LAS ENTIDADES DEL MODELO. PARÁMETROS DE INTERÉS

Las entidades que representan valores o datos que maneja el transelevador, los eventos de movimiento con o sin carga o las órdenes recibidas del sistema de gestión, almacenan información en etiquetas, que acompañan a estas entidades a continuación se muestran las más relevantes:

- p_C_DriverX (Component[IStdFrequencyInverter]): Variador de frecuencia al cual le llegan señales que permiten gestionar los movimientos en x del motor del transelevador. Realiza una rampa de aceleración y desaceleración.
- P_C_DriverY (Component[IStdFrequencyInverter]): Variador de frecuencia al cual le llegan las señales que permiten gestionar los movimientos en Y del motor del transelevador. Realiza una rampa de aceleración y desaceleración
- P_C_MaspPosSC (Component [Dimensional map Ex]): Componente que proporciona un mecanismo para almacenar un mapa multidimensional donde el número de dimensiones, en este caso, es 3. Gracias a este objeto es posible almacenar posiciones bidimensionales y tridimensionales en memoria para su consulta y modificación.
- P_Ini_Map_PosSC (String): Variable mediante la cual se accederá a los datos del mapa multidimensional.
- P_I_ExtractorIN (BOOL): Variable relacionada con el sensor que informa de que las horquillas telescópicas se encuentran recogidas en la cuna.
- P_I_ExtractorLeft (BOOL): Variable relacionada con el sensor que informa del estado de que las horquillas telescópicas están extendidas hacia la izquierda.
- P_I_ExtractorRight (BOOL): Variable relacionada con el sensor que informa del estado de que las horquillas telescópicas están extendidas hacia la izquierda
- P_O_Extractor_Positive (BOOL): Variable relacionada con el actuador/motor de las horquillas, esta variable genera el movimiento de las horquillas hacia la derecha. Solo toma dos valores TRUE/FALSE. Activando/desactivando el movimiento hacia la derecha de los extractores
- P_O_Extractor_negative (BOOL): Variable relacionada con el actuador/motor de las horquillas, esta variable genera el movimiento de las horquillas hacia la izquierda. Solo toma dos valores TRUE/FALSE. Activando/desactivando el movimiento hacia la izquierda de los extractores

- P_MW_MachineCycle (INT): Variable del programa que indica qué etapa del programa se está ejecutando
- P_MW_CodeError (INT): Variable a la cual se le van asignando diferentes valores en función de la etapa del programa en la que haya ocurrido el error.
- P_MDW_TransportNumber (DINT): Variable relacionada con el número de máquinas del mismo tipo que existe dentro de la instalación.
- P_MDW_TargetXLogic/Physical (DINT): Variable directamente relacionada con la posición final o destino de una orden generada por el SGA, al tratarse de un valor lógico estará codificado según las posiciones lógicas de las ubicaciones, si se trata de la variable física su valor vendrá en mm. Datos de posición referentes al eje X
- P_MDW_TargetYLogic/Physical (DINT): Variable directamente relacionada con la posición final o destino de una orden generada por el SGA, al tratarse de un valor lógico estará codificado según las posiciones lógicas de las ubicaciones, si se trata de la variable física su valor vendrá en mm.. Datos de posición referentes al eje Y
- P_MDW_TargetZLogic/Physical (DINT): Variable directamente relacionada con la posición final o destino de una orden generada por el SGA, al tratarse de un valor lógico estará codificado según las posiciones lógicas de las ubicaciones, si se trata de la variable física su valor vendrá en mm.. Datos de posición referentes al eje Z
- P_MDW_SourceXLogic/Physical (DINT): Variable directamente relacionada con la posición inicial u origen de una orden generada por el SGA, al tratarse de un valor lógico estará codificado según las posiciones lógicas de las ubicaciones, si se trata de la variable física su valor vendrá en mm. Datos de posición referentes al eje X
- P_MDW_SourceYLogic/Physical (DINT): Variable directamente relacionada con la posición inicial u origen de una orden generada por el SGA, al tratarse de un valor lógico estará codificado según las posiciones lógicas de las ubicaciones, si se trata de la variable física su valor vendrá en mm. Datos de posición referentes al eje Y
- P_MDW_SourceZLogic/Physical (DINT): Variable directamente relacionada con la posición inicial u origen de una orden generada por el SGA, al tratarse de un valor lógico estará codificado según las posiciones lógicas de las ubicaciones, si se trata de la variable física su valor vendrá en mm. Datos de posición referentes al eje Z
- P_MB_StationType (SINT): El valor del número de tipo de estación es único e intransferible para cada uno de los elementos que intervienen en la emulación. Están definidos según unos estándares para el propio software.
- P_MB_StationNumber (SINT): Número configurado en Galileo y establecido por el SGA, en este caso, su valor, será 1.

- P_MB_Occupation (SINT): Variable que indicará si una estación está realizando algún tipo de tarea o tiene algún contenedor encima, es decir, nos indica si la máquina está cargada o se le ha asignado una tarea.

La siguiente tabla muestra todos los parámetros empleados para el programa de control:

Nombre del parámetro	Tipo
IStdFrequencyInverter	COMPONENT
IStdFrequencyInverter	COMPONENT
DimensionalMapEx	COMPONENT
p_I_ExtractorIN	BIT
p_I_ExtractorLeft	BIT
p_I_ExtractorRight	BIT
p_I_ForwardSensor	BIT
p_I_ShelfRight	BIT
p_I_ShelfSensor	BIT
p_I_WeightPallet	BIT
p_Ini_MapPosSC	STRING
p_Left	BIT
p_M_AdvanceMovement	BIT
p_M_AnyMode	BIT
p_M_AutoExtractorIN	BIT
p_M_AutoExtractorNegative	BIT
p_M_AutoExtractorPositive	BIT
p_M_ChangeIN	BIT
p_M_ConveyorPresence	BIT
p_M_CreatePallet	BIT
p_M_EntryConsent	BIT
p_M_EntrySlowSpeed	BIT
p_M_ExitRequest	BIT
p_M_ExtractorIN	BIT
p_M_ExtractorNegative	BIT
p_M_ExtractorPositive	BIT
p_M_Fault	BIT
p_M_IsOK_X	BIT
p_M_IsOK_Y	BIT
p_M_IsTransition	BIT
p_M_LoadFinished	BIT
p_M_LoadFinished_Shelf	BIT
p_M_LoadStart_Shelf	BIT
p_M_ManBackward	BIT
p_M_ManDown	BIT
p_M_ManExtractorIN	BIT

p_M_ManExtractorNegative	BIT
p_M_ManExtractorPositive	BIT
p_M_ManForward	BIT
p_M_ManResetValues	BIT
p_M_ManualMode	BIT
p_M_ManualSlave	BIT
p_M_ManUp	BIT
p_M_MotorRunning	BIT
p_M_MoveManualNext	BIT
p_M_NotFinishedFault	BIT
p_M_OrderInProcess	BIT
p_M_PalletLeft	BIT
p_M_PalletPresence	BIT
p_M_PalletRight	BIT
p_M_Presence_Mem	BIT
p_M_PresenceToNext	BIT
p_M_PreviousRelease	BIT
p_M_SideME	BIT
p_M_SideMS	BIT
p_M_SideShelf	BIT
p_M_SourceTask	BIT
p_M_TargetTask	BIT
p_M_TrackingPresent	BIT
p_M_UnloadFinished	BIT
p_M_UnloadFinished_Shelf	BIT
p_M_UpdateShelfStatus	BIT
p_M_UpdateShelfTime	BIT
p_M_UpdateStationStatus	BIT
p_MB_AisleNumber	BYTE
p_MB_Capacity	BYTE
p_MB_Capacity_Mem	BYTE
p_MB_CurrentStateSC	BYTE
p_MB_LoadConsentToSC_ME	BYTE
p_MB_NewTracking	BYTE
p_MB_Occupation	BYTE
p_MB_StationNumber	BYTE
p_MB_StationType	BYTE
p_MB_Status_Mem	BYTE
p_MB_TargetStationNumber	BYTE
p_MB_TargetStationType	BYTE
p_MDW_CurrentPositionX	DWORD
p_MDW_CurrentPositionY	DWORD
p_MDW_NextConveyor	DWORD

p_MDW_PrevConveyor	DWORD
p_MDW_SourceXLogic	DWORD
p_MDW_SourceXPhysical	DWORD
p_MDW_SourceYLogic	DWORD
p_MDW_SourceYPhysical	DWORD
p_MDW_SourceZLogic	DWORD
p_MDW_SourceZPhysical	DWORD
p_MDW_StatusX	DWORD
p_MDW_StatusY	DWORD
p_MDW_TargetXLogic	DWORD
p_MDW_TargetXPhysical	DWORD
p_MDW_TargetYLogic	DWORD
p_MDW_TargetYPhysical	DWORD
p_MDW_TargetZLogic	DWORD
p_MDW_TargetZPhysical	DWORD
p_MDW_TransportNumber	DWORD
p_MDW_WTUNumber	DWORD
p_MW_CodeError	WORD
p_MW_ExtractorCycle_LoadShelfLeft	WORD
p_MW_ExtractorCycle_LoadShelfRight	WORD
p_MW_ExtractorCycle_ME_Left	WORD
p_MW_ExtractorCycle_ME_Right	WORD
p_MW_ExtractorCycle_MS_Left	WORD
p_MW_ExtractorCycle_MS_Right	WORD
p_MW_ExtractorCycle_UnloadShelfLeft	WORD
p_MW_ExtractorCycle_UnloadShelfRight	WORD
p_MW_MachineCycle	WORD
p_MW_TypeFromLoad	WORD
p_OExtractor_Negative	BIT
p_OExtractor_Positive	BIT
p_OExtractorIN	BIT
p_Right	BIT
p_TON_UpdateShelfStatus	TON
p_TON_UpdateStationStatus	TON

Tabla 4.1.- *Parámetros del programa de control*

4.5. - COMUNICACIONES EN LA ESTACIÓN TRANSELEVADOR

El transelevador (o TRASLO) es una maquina automática que se encarga del procedimiento de ubicación y extracción de contenedores en la estantería mediante un sistema de movimiento en tres ejes (traslación, elevación y profundidad).

Se emplea en pasillos automáticos no transitables

Al recibir una orden un transelevador realiza la siguiente secuencia de movimiento:

1. Desplazamiento a la posición de ubicación del contenedor que debe mover (posición origen). Las coordenadas de esta posición están dadas en los 3 ejes (traslación X, elevación Y y lado Z
2. Carga del contenedor en la máquina mediante un mecanismo denominado sistema extractor.
3. Desplazamiento a la posición de la ubicación destino del contenedor que se está moviendo (posición destino). Las coordenadas de esta posición están dadas en los tres ejes (traslación X, elevación Y y lado Z)
4. Descarga del contenedor en la ubicación indicada mediante el sistema de extracción.

Para el sistema de control, un transelevador es un equipamiento. Se deben conocer:

- Pasillo automático: Código del pasillo
- Transelevador en el pasillo: Indica si el pasillo automático incorporara un transelevador propio. Si no lo incorpora, se deberá indicar cuál de los transelevadores disponibles en los otros pasillos automáticos, tendrá el control sobre este.
- Número de extractores: número de mecanismos encargados de realizar el movimiento de carga y descarga de contenedores en el transelevador.
- Acoplamiento: Define si la orden de movimiento limita o condiciona la orden de movimiento del otro contenedor en el mismo sistema extractor.

La estación transelevador realiza el siguiente protocolo entre Galileo (SCT) y EasyWCS (SGA):

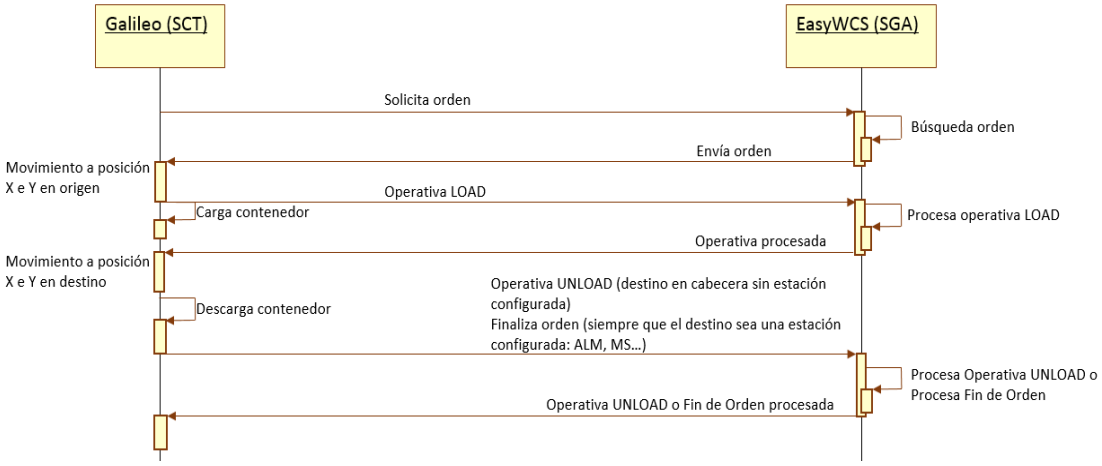


Figura 4.4.- *Protocolo de comunicaciones Galileo –EasyWCS (estación TRASLO)*

A nivel de control los pasos necesarios son:

1º. Galileo solicita una orden de movimiento a EasyWCS, con los siguientes parámetros:

Parámetros Búsqueda de orden	**Valor**
Tipo de estación	Ver "Tabla: Tipos de estaciones" ANEXO II (para el TRASLO toma el valor 1)
Número de estación	Número de transelevador (solo hemos configurado un Traslo, toma valor 1)
Transporte	Se envía siempre 0
X	Coordenada X actual o en "Búsquedas en paralelo a origen" la coordenada origen de la orden que se está ejecutando
Y	Coordenada Y actual o en "Búsquedas en paralelo a origen" la coordenada origen de la orden que se está ejecutando
Pasillo	Pasillo actual o en "Búsquedas en paralelo a origen" el pasillo origen de la orden que se está ejecutando
Estado	En Servicio (S) / Con Corrección a depósito (E) /Movimientos simultáneos a entrada (I)
Ocupación actual	Número de trackings de la máquina
Capacidad actual	Valor de configuración constante
Aux01… Aux08	Custom

Tabla 4.2.- *Parámetros de búsqueda de orden*

Según ciertas condiciones, es posible realizar una operativa que permita realizar Movimientos simultáneos de entrada (carga y descarga simultánea -I-)

En estos casos, a través del programa de control para la emulación, Galileo comprobará de forma estándar que todos los movimientos de salida, si los hubiera, qué están en ejecución en el transelevador están todos ya cargados en cuyo caso buscará

movimientos pendientes en la cabecera destino de los movimientos de salida. Para EasyWCS la mesa de entrada debe estar situada justo en frente de la mesa de salida; si no es así EasyWCS no devolverá ningún movimiento.

NOTA: Las coordenadas indicadas por el programa de control de Galileo, deben existir obligatoriamente en EasyWCS.

2º. Enviar la orden de movimiento generada para el transelevador.

La orden de movimiento tiene la siguiente estructura de datos de tracking:

Campo	Descripción
Número Transporte	Número de Transporte
Prioridad	Prioridad del tracking
Secuencia	Secuencia de tracking
UMA	Identificador del contenedor
Tipo Contenedor	Tipo de contenedor
Tipo Altura	Tipo de altura
DATOS RELATIVOS AL ORIGEN DE MOVIMIENTO	
Número Estación	Estación origen del movimiento (número)
Tipo Estación	Estación origen del movimiento (tipo)
X	Coordenada X del alveolo origen del movimiento
Y	Coordenada Y del alveolo origen del movimiento
X Local	Coordenada X local al alveolo
Y Local	Coordenada Y local al alveolo
Lado	Lado de origen
Pasillo	Pasillo de origen
Profundidad	Profundidad de origen
DATOS RELATIVOS AL DESTINO DE MOVIMIENTO	
Número Estación	Estación destino del movimiento (número)
Tipo Estación	Estación destino del movimiento (tipo)
X	Coordenada X del alveolo destino del movimiento
Y	Coordenada Y del alveolo destino del movimiento
X Local	Coordenada X local al alveolo
Y Local	Coordenada Y local al alveolo
Lado	Lado de destino
Pasillo	Pasillo de destino
Profundidad	Profundidad de destino
DATOS CUSTOM	
Custom Data	Datos auxiliares
PERCENT_V_X	Porcentaje de velocidad en X
PERCENT_V_Y	Porcentaje de velocidad en Y
PERCENT_V_Z	Porcentaje de velocidad en Z

Tabla 4.3.- *Estructura de datos del tracking.*

NOTA: Los datos enviados desde el SGA son ubicados a partir del byte 25 (inclusive) del Custom Data del tracking. Los 25 primeros son utilizados internamente por el sistema de control.

La estructura del tracking dentro del sistema de control tiene su reflejo en EasyWCS en las órdenes de transporte.

3º. El transelevador realiza el movimiento en los ejes X e Y a la ubicación origen donde está el contenedor.

4º. Una vez colocado en ambos ejes y mientras el transelevador carga el contenedor, Galileo envía una operativa LOAD indicando a EasyWCS en qué orden se cargarán los contenedores en la cuna. La operativa LOAD tiene los siguientes parámetros:

Parámetros Operativa LOAD	Valor
Tipo de estación	Ver "Tabla: Tipos de estaciones" ANEXO II (para el TRASLO toma el valor 1)
Número de estación	Número de transelevador (solo hemos configurado un Traslo, toma valor 1)
Transporte	Se envía siempre 0
Tipo Operativa	Operativa de carga equivale al valor 3
Tipo de contenedor	Enviado por TA (datos de tracking)
Tipo de altura	Enviado por TA (datos de tracking)
X Local	Ver "Tabla: Posición local dentro de la ubicación traslo" ANEXO II
Y Local	Ver "Tabla: Posición local dentro de la ubicación traslo" ANEXO II
Aux01… Aux08	Custom

Tabla 4.4.- *Parámetros de la operativa de carga del transelevador.*

En la operativa LOAD, Galileo debe indicar la posición dentro de la cuna del transelevador donde se han cargado los contenedores.

En nuestro caso el transelevador tiene simple fondo, es decir, una sola ubicación. En el capítulo “Posición local dentro de la ubicación de un Traslo (ANEXO II)” se detalla la normativa empleada para designar las diferentes ubicaciones que puede tener la cuna de un traslo

5º. El transelevador realiza el movimiento en los ejes X e Y a la ubicación destino donde tiene que depositar el contenedor.

6º. El transelevador descarga el contenedor en la ubicación destino y lanza una operativa UNLOAD solo en el caso de descargar en cabecera y que esa descarga no se realice en una estación configurada para indicar a EasyWCS que los contenedores no se encuentran en el equipamiento (transelevador). Se debe resaltar que con una operativa UNLOAD no se realiza ningún final de orden.

La operativa UNLOAD tiene los siguientes parámetros:

Parámetros Operativa UNLOAD	Valor
Tipo de estación	Ver "Tabla: Tipos de estaciones" ANEXO II (para el TRASLO toma el valor 1)
Número de estación	Número de transelevador (solo hemos configurado un Traslo, toma valor 1)
Transporte	Enviado por TA (datos de tracking)
Tipo Operativa	Operativa de carga equivale al valor 4
Tipo de contenedor	Enviado por TA (datos de tracking)
Tipo de altura	Enviado por TA (datos de tracking)
X	0
Y	0
Lado	0
Pasillo	0
Profundidad	0
Aux01… Aux07	Custom

Tabla 4.5.- *Parámetros de la operativa de descarga del transelevador.*

Es importante que los parámetros mostrados en la tabla anterior sean correctos. Un error en las coordenadas puede producir efectos indeseados en el sistema, ya que la operativa UNLOAD implica un cambio de ubicación.

Si la descarga se realiza sobre la estación destino se realiza un Fin de Orden.

Parámetros Operativa UNLOAD	Valor
Tipo de estación	Ver "Tabla: Tipos de estaciones" ANEXO I (para el TRASLO toma el valor 1)
Número de estación	Número de transelevador (solo hemos configurado un Traslo, toma valor 1)
Contenedor Id	Enviado por TA (datos de tracking)
Transporte	Enviado por TA (datos de tracking)
Código de error	Tipo de Error (Depósito, en descarga a estantería; Extracción, en recogida de estantería; Altura, en descarga a estantería) o Sin Error
Tipo de plataforma	Enviado por TA (datos de tracking)
Tipo de altura	Enviado por TA (datos de tracking)
X	Coordenada X de la ubicación
Y	Coordenada Y de la ubicación
X local	Coordenada X local de la ubicación
Y local	Coordenada Y local de la ubicación

Lado	Lado de la ubicación
Pasillo	Pasillo de la ubicación
Profundidad	Profundidad de la ubicación
Aux01… Aux07	Custom

Tabla 4.6.- *Parámetros de la operativa de fin de orden del transelevador.*

4.5.2.- CASOS ESPECIALES

- **Errores de depósito:**

  Los errores de depósito se gestionan de forma que Galileo ante un error de depósito de cualquiera de los contenedores, será programado para que se detenga y genere un aviso y se cambie el destino del contenedor/es con error de depósito y puede cambiar también el del resto de los contenedores del transelevador (en caso de existir más de una ubicación en la cuna del traslo).

  Galileo a través del programa de control creado para la emulación, en los errores de depósito, realiza la búsqueda de orden restando de la ocupación el error o errores de depósito.

- **Errores de extracción:**

  En los errores de extracción, Los gestionará Galileo de la misma forma que en el caso anterior, independientemente del número de cargas, cancela las tareas para la ubicación del error de extracción y cualquier tarea que exista para las profundidades posteriores. En el caso de una máquina que tenga que recoger de una misma posición y en diferentes profundidades el comportamiento es el mismo, el Galileo realizará un Fin de Orden (indicando el error de extracción en el código de error) para cada una de las profundidades afectadas, en este caso concreto solo hay una profundidad.

- **Errores de altura:**

  En los errores de altura, Galileo realizará un Fin de Orden indicándolo en el código de error. Este tipo de error solo se indica en descargas a la estantería.

  Los errores de altura son poco frecuentes porque normalmente los contenedores antes de entrar en la máquina transelevador pasan una serie de controles de peso, galibo, de patines y de huecos.

- **Gestión de movimientos:**

  Galileo es el encargado de gestionar la secuencia de ejecución de órdenes, es decir, el programa de control del traslo, se ha configurado de tal forma que realizará las operaciones de movimientos según se detallan a continuación:

  - **Órdenes de Trasvase-Salida** → Galileo ejecutará primeramente el Trasvase. De esta forma, se podrá generar, si es posible, otra orden de Salida y así optimizar movimientos.
  - **Órdenes de Entrada-Salida** → Mediante EasyWCS y a través del programa de control de Galileo en emulación se podrá cambiar el destino de una orden entrada al origen de una orden de Salida. Para ello, el traslo

ha de encontrarse en condiciones de recibir orden (antes de realizar el movimiento a destino, ocupación, etc) y EasyWCS generar dicha orden de Salida. De esta forma, Galileo gestionará extraer primeramente el contenedor cuyo origen es la orden de Salida y luego introducirá el contenedor de la orden de Entrada. Se trata de una optimización para aprovechar la ubicación libre y ahorrar un movimiento.

- **Posiciones inaccesibles para la máquina:**

  EasyWCS será el responsable de enviar las órdenes de movimiento al transelevador. Pero no será el encargado de gestionar si las posiciones origen o destino son inaccesibles para la máquina, de esto se encargará el programa de control creado para la emulación del transelevador, de este modo se gestionará como error la orden hacia o desde una posición inaccesible.

  La accesibilidad o no de una posición viene determinada por la posición en cuestión y por la cuna. Podrían darse varios escenarios:

  - Caso 1: Ubicaciones límite a final de pasillo y el contenedor de la ME con destino la ubicación límite. En este caso si el transelevador no puede acceder a ella se cargará en la variable de error un valor relacionado con dicho fallo y se anulará el movimiento, cancelando la tarea a ejecutar.
  - Caso 2: Ubicaciones límite al inicio del pasillo y el contenedor de la ME con destino la ubicación límite. En este caso si el transelevador no puede acceder a ella se cargará en la variable de error un valor relacionado con dicho fallo y se anulará el movimiento, cancelando la tarea a ejecutar.

4.5.3.- TIPOS DE ESTACIONES

El tipo de estación determinará la funcionalidad estándar que se aplicará en un punto de decisión. Es por lo tanto imprescindible entender su concepto funcional ya que se debe utilizar el tipo de estación adecuada para desempeñar una función determinada.

Campo	Tipo	Descripción
ALM	0	Almacén
TRASLO	1	Máquina transelevador
PK	2	Puesto de picking
PIE	3	Puesto de identificación de entradas
PS	4	Puesto de salidas
REAC	5	Puesto de reacondicionamiento
RECH	7	Puesto de rechazos
ME	9	Mesa de entrada al almacén (carga transelevador)
MS	10	Mesa de salida del almacén (descarga transelevador)
MU	11	Mesa de ubicación
APL	13	Máquina apilador/desapilador
LANZ	14	Máquina lanzadera

MP	16	Puesto de preparación de picking
PKE	18	Puesto de entrada del picking
ET	20	Puesto o estación de transito e identificación
CME	21	Puesto de entrada a pasillo
CH	35	Máquina carrusel horizontal
AV	36	Máquina elevadora del almacén vertical
RETENTION	42	Puesto de retención provisional de contenedores
PCS	53	Puesto de control de secuenciación
PB	54	Buffer de proximidad
ECB	55	Buffer de contenedores vacíos

Tabla 1.1.- Tipos de estaciones.

Los números de los tipos de estación no pueden ser modificados, son el estándar que se emplea en Galileo/Designer/WCS para designar y hacer referencia a cada una de las máquinas de la instalación. Se trata de una relación establecida como estándar para los diferentes softwares que pueden intervenir en la simulación de una instalación.

Se requiere conocer los tipos de estaciones para el presente proyecto para implementar el escenario de prueba con la máquina trasnelevador.

4.5.4.- DATOS DEL TRACKING

Los contenedores que se desplazan por el almacén llevan asociados datos relativos a la orden de movimiento. Estos datos son denominados tracking

Existen dos formas de asignación de tracking a un contenedor:

- Los trackings generados por Galileo: se utilizan cuando Galileo inicia el movimiento sin conocer la información relativa al contenedor que va a transportar.

  ** Cuando galileo genere el tracking para un contenedor ha de hacerlo con valor 1 en el campo número de transporte. Siempre que el campo número de transporte tenga valor 1, el contenedor será ignorado por EasyWCS.

- Los trackings generados por EasyWCS: Trackings generados por EasyWCS cada vez que se genera un movimiento para un contenedor identificado.

4.5.4.1.- ESTRUCTURA DEL TRACKING

La estructura del tracking en Galileo tiene su reflejo como ordenes de movimiento en EasyWCS.

La estructura corresponde a los datos de un solo contenedor que se desplaza por el almacén.

El tracking se desplaza junto con el contenedor por las distintas máquinas de las que consta el almacén. Cada máquina puede almacenar un máximo de 10 trackings en su memoria interna, pero puede utilizar colecciones para almacenarlos en otro.

En las tablas siguientes se detalla la información que contiene el tracking asociada a un contenedor para Galileo:

Campo	Descripción
Número Transporte	Número de Transporte
Prioridad	Prioridad del tracking
Secuencia	Secuencia de tracking
UMA	Identificador del contenedor
Tipo Contenedor	Tipo de contenedor
Tipo Altura	Tipo de altura
DATOS RELATIVOS AL ORIGEN DE MOVIMIENTO	
Número Estación	Estación origen del movimiento (número)
Tipo Estación	Estación origen del movimiento (tipo)
X	Coordenada X del alveolo origen del movimiento
Y	Coordenada Y del alveolo origen del movimiento
X Local	Coordenada X local al alveolo
Y Local	Coordenada Y local al alveolo
Lado	Lado de origen
Pasillo	Pasillo de origen
Profundidad	Profundidad de origen
DATOS RELATIVOS AL DESTINO DE MOVIMIENTO	
Número Estación	Estación destino del movimiento (número)
Tipo Estación	Estación destino del movimiento (tipo)
X	Coordenada X del alveolo destino del movimiento
Y	Coordenada Y del alveolo destino del movimiento
X Local	Coordenada X local al alveolo
Y Local	Coordenada Y local al alveolo
Lado	Lado de destino
Pasillo	Pasillo de destino
Profundidad	Profundidad de destino
DATOS CUSTOM	
Custom Data	Datos auxiliares
PERCENT_V_X	Porcentaje de velocidad en X
PERCENT_V_Y	Porcentaje de velocidad en Y
PERCENT_V_Z	Porcentaje de velocidad en Z

Tabla 1.2.- Información de tracking asociada a un contenedor

4.5.5.- POSICIÓN LOCAL DENTRO DE LA UBICACIÓN TRASLO.

Las posiciones relativas en el interior de la cuna siguen la siguiente estructura (suponiendo una cuna con capacidad para 4 contenedores):

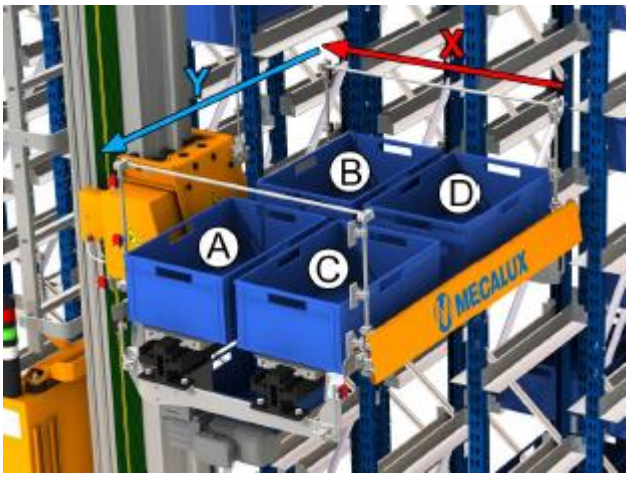


Figura 4.5.- *Posiciones que ocupan los contenedores en la cuna del traslo*

Los ejes considerados son los equivalentes a los utilizados en el movimiento de la máquina y siguen su misma normativa:

- X Local de traslación. Incrementa en el mismo sentido que las coordenadas del pasillo
- Y Local. Se numeran igual que para Z o profundidad (1→profundidad 1y 2→ profundidad 2). Tomando como referencia que siempre lado 1 es el izquierdo y lado 2 es el derecho.

Según estas indicaciones, las bandejas de la imagen anterior indican los valores de la tabla en los parámetros X local e Y local para el evento LOAD

Bandeja	X Local	Y Local
A	2	2
B	2	1
C	1	2
D	1	1

Tabla 4.7.- *Posición local dentro de la ubicación traslo*

Suponiendo una cuna con capacidad para n contenedores, según estas indicaciones, se deben configurar los valores correspondientes:

Número de contenedores	Bandeja
1	D
2 (mismo extractor)	D - C
2 (diferente extractor)	B - D
4	A - B - C - D

Tabla 4.8.- *Posición contenedores – Posición bandeja*

Siguiendo la lógica anterior, se muestra a continuación, las coordenadas lógicas, en función del tipo de transelevador:

Transelevador 1 – 1 (1 extractor – 1 contenedor). Caso que aplica al presente proyecto:

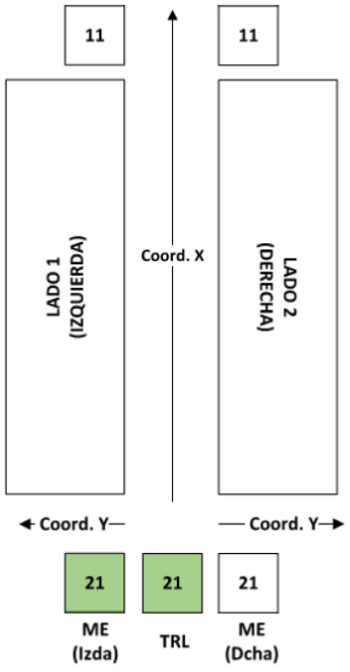


Figura 4.6.- *Transelevador 1 extractor – 1 contenedor*

4.5.6.- FUNCIONES DE COMUNICACIÓN

Existen varios tipos de funciones de comunicación entre EasyWCS y Galileo que será necesario tenerlas en cuenta para el posterior desarrollo del programa de control:

1. Multioperación
2. Evento
3. Búsqueda de orden
4. Fin de orden
5. Cancelación de orden
6. Actualización de estaciones
7. Actualización de rutas
8. Devolución de órdenes

En cada una de las estaciones existentes en las máquinas del presente proyecto pueden aparecer una o varias funciones de comunicación.

4.5.6.1.- MULTIOPERACIÓN

Una multioperación es una función específica para DesignerIV que permite a EasyWCS notificar en una sola comunicación varias funciones. En concreto en una multioperación se pueden notificar:

- Una búsqueda de hasta 10 órdenes,
- y/o 10 eventos
- y/o 10 fines de orden

4.5.6.2.- EVENTO

Un evento es una notificación a iniciativa del SCT hacia EasyWMS® en la que se indica una casuística (indicada a través del campo tipo de evento)

La siguiente lista muestra los diferentes tipos de evento gestionados de forma estándar y sus valores:

Tipo	Descripción
1	Palet en PIE
2	Pulsación de rechazo
3	Load (Carga de contenedor)
4	UnLoad (Descarga de contenedor)
5	Pila llena
6	Pulsación de reintento
7	Simulación picking (reservado)
8	Confirmación de picking
9	Extracción de PS
10	Transferencia de contenedor de ECB a MP (HPKS)

Tabla 4.9.- *Tipos de eventos y sus equivalencia*

4.5.6.3.- BÚSQUEDA DE ORDEN.

La estructura existente en EasyWMS® para la búsqueda de orden, mediante Galileo IV:

Parámetro	Método Galileo
Número estación	**SetMultiOPSearchStationNumber** Byte
Tipo Estación	**SetMultiOPSearchStationType** Byte
Transporte	**SetMultiOPSearchTransport** Dword
X	**SetMultiOPSearchCurrentX** Word
Y	**SetMultiOPSearchCurrentY** Word
Aisle	**SetMultiOPSearchCurrentAisle** Byte
Side	**SetMultiOPSearchCurrentSide** Byte
State	**SetMultiOPSearchCurrentState** Byte
CurrentCount	**SetMultiOPSearchOccupation** Byte
Capacity	**SetMultiOPSearchcapacity** Byte
Aux01/08	**SetMultiOPSearchAuxData** Byte [0…7], Byte

Tabla 4.10.- *Búsqueda de orden mediante Multi-operación*

Parámetros de la búsqueda de orden:

Parámetro	Descripción	Valor
WAREHOUSE	Identificador del almacén	Configurado en el programa de control y establecido por WCS
Station Type	Número de estación	Los tipos de estaciones están estandarizados y pueden consultarse en "Tabla de Tipos de estaciones"
Station_Number	Tipo de estación	Número configurado en EasyWCS y establecido por el programa de control

Transport	Número de transporte	Según el tipo de estación	
X	Coordenada X de la ubicación	Según el tipo de estación	
Y	Coordenada Y de la ubicación	Según el tipo de estación	
Side	Lado de la ubicación	**Valor**	**Lado**
		1	Izquierda
		2	Derecha
AISLE	Pasillo asociado a la ubicación	Según el tipo de estación	
State	Estado en el que se encuentra la máquina	**Valor**	**Estado**
		83 (S)	En servicio
		70 (F)	En defecto
		77(M)	Manual o semiautomático
		73 (I)	Movimientos simultáneos de entrada
Occupation	Ocupación actual de trackings en la máquina	Según la ocupación actual [7]	
Capacity	Capacidad de trackings de la máquina	Número configurado en EasyWCS y en el programa de control	
Aux01	Auxiliar 1	Custom	
Aux02	Auxiliar 2	Custom	
Aux03	Auxiliar 3	Custom	
Aux04	Auxiliar 4	Custom	
Aux05	Auxiliar 5	Custom	
Aux06	Auxiliar 6	Custom	
Aux07	Auxiliar 7	Custom	
Aux08	Auxiliar 8	Custom	

Tabla 4.11.- *Parámetros de la búsqueda de orden*

4.5.6.4.- FIN DE ORDEN

Cuando el Sistema de Control Galileo finaliza una orden de transporte lo notifica a EasyWCS® mediante un procedimiento de final de orden.

Los códigos de finalización de orden estándar establecidos son:

Código	Descripción

[7] En el caso de realizar una Multioperación con un Fin + Búsqueda, la ocupación indicada en el parámetro no ha de notificar los trackings actuales de la máquina, si no que ha de restar los trackings que vayan a ser finalizados en esa misma multioperación. Es decir, la ocupación sería la capacidad menos número de órdenes que se podrían recibir tras el fin de orden.

0	Finalización correcta
1	Error de depósito
2	Error de extracción
7	Error de altura
9	Error de gálibo (obsoleto)

Tabla 4.12.- *Códigos de error estándar de la finalización de orden*

Existe también una multi-operación para el fin de orden que viene dada por los siguientes métodos:

Parámetro	Método Galileo
Número estación	**SetMultiOPEndRealStationNumber** Byte, Byte
Tipo Estación	**SetMultiOPEndRealStationType** Byte, Byte
Movimiento	**PrepareMultiOPCommandToFinish** Byte, Byte, Dword, Dword
Codigo de error	**PrepareMultiOPCommandToFinish** Byte, Byte, Dword, Dword
UMA	**PrepareMultiOPCommandToFinish** Word
Tipo de plataforma	**SetMultiOPEndPlatformType** Byte, Byte
Tipo de altura	**SetMultiOPEndHeightType** Byte, Byte
X	**SetMultiOPSearchCurrentX** Byte, Word
Y	**SetMultiOPSearchCurrentY** Byte, Word
Lado	**SetMultiOPSearchCurrentAisle** Byte, Byte
Pasillo	**SetMultiOPSearchCurrentSide** Byte, Byte
Profundidad	**SetMultiOPSearchCurrentState** Byte, Byte
Xlocal	**SetMultiOPSearchOccupation** Byte, Byte
Ylocal	**SetMultiOPSearchcapacity** Byte, Byte
Aux	**SetMultiOPSearchAuxData** Byte, Byte, Dword, Dword
Aux01/08	**SetMultiOPSearchAuxData**

	Byte[0…7], Byte

Tabla 4.13.- *Métodos multi-operación para el fin de orden*

4.5.6.5.- ACTUALIZACIÓN DE ESTACIONES

La actualización de las estaciones desde el Galileo, referenciando capacidades, ocupaciones, presencias, etc. para que EasyWCS® pueda decidir sobre la forma más correcta de gestionar la instalación, se realiza bajo las siguientes condiciones:

- Si el estado anterior es diferente del actual.
- Si es igual, debe actualizarse de forma periódica tras un tiempo configurable (de base 1000 ms). En el caso Galileo, este tiempo se configura en la consola en el campo "Timeout para la actualización de estaciones/rutas(ms)"

4.5.6.6.- ACTUALIZACIÓN DE RUTAS

La actualización de rutas bajo el sistema de control Galileo se realiza bajo las siguientes condiciones:

- Si el estado anterior es diferente del actual.
- Si es igual, debe actualizarse de forma periódica tras un tiempo configurable (de base 1000 ms). En el caso Galileo, este tiempo se configura en la consola en el campo "Timeout para la actualización de estaciones/rutas(ms)".

El comportamiento de EasyWCS® en función de los valores de estado que envía el sistema de control Galileo para una determinada ruta es el siguiente:

- 0 → Ruta en defecto: EasyWCS® no genera movimientos al destino de la ruta ni estaciones intermedias. Si existe algún movimiento hacia el destino ya en ejecución, el fin de orden devuelve un error.
- 1 → Ruta en servicio (habilitada): Ruta válida para cualquier movimiento.
- 2 →Ruta llena y en defecto: Mismo comportamiento que el valor 0 (se mantiene el valor para comportamientos custom).
- 3 → Ruta llena y en servicio: Si es desde PKE o CME hacia PK o SE ME intenta hacer otros movimientos para no colapsar y que los contenedores recirculen, desde cualquier otro tipo de estación se ignora que esté cargada.

Teniendo en cuenta esta información es importante tener especial cuidado en la habilitación/deshabilitación de cada ruta. Las posibles consecuencias sobre las operaciones realizadas son:

- **Búsqueda de orden**: Si no se encuentra ninguna ruta habilitada al destino, no se genera el movimiento.

Fin de orden: Cuando se notifica un final de orden al EasyWCS®, éste intenta generar un movimiento hacia la estación destino. Si no se encuentra una ruta para llegar EasyWCS® no finalizará al sistema de control Galileo esa orden

4.6. - IMPLEMENTACIÓN DE MOVIMIENTOS. CICLOS DE TRABAJO DE LOS TRANSELEVADORES

En este apartado se van a exponer los casos que aplican al presente proyecto recogidos en la norma FEM 9.851 (06.2003) para realizar el cálculo de tiempos de ciclos de trabajo de los transelevadores (operados bajo control automático)

Los transelevadores, considerados "*Maquinaria de elevación y transporte*" siguen la normativa UNE 58912:2004.

Esta norma establece las reglas que son aplicables exclusivamente a los transelevadores con mando automático y diseñados para la manipulación de unidades de carga.

Esta norma proporciona un método uniforme para el cálculo de tiempos de ciclos de trabajo de los transelevadores y así determinar sus ritmos de manipulación para optimizar los almacenes de gran altura.

Ritmo de manipulación:

El ritmo de manipulación en un almacén de gran altura indica el número de unidades carga que se almacenan y/o se desalmacenan por unidad de tiempo. En el área de almacén ello depende de:

- Del número de transelevadores
- Del número de ciclos por transelevador, cuyos ciclos dentro de un turno dependen de:
    - La secuencia de movimientos
    - La disposición de los puntos de traslado
    - Rendimiento del transelevador

Tiempo de un ciclo:

El tiempo de un ciclo es la duración de una secuencia de movimientos.

En el "ANEXO III: Velocidades y Aceleraciones del sistema transelevador" se detallan los cálculos y pruebas realizadas para llegar a las siguientes ecuaciones, que serán las empleadas para implementar en el programa de control de la simulación, teniendo en cuenta que, los transelevadores, tienen una variable fundamental a tener en cuenta, el JERK, término usado para describir el ritmo con que cambia la aceleración:

Partiendo de unas ecuaciones genéricas:

$$x = \frac{j}{6}(t - t_0)^3 + \frac{a_0}{2}(t - t_0)^2 + v(t - t_0) + x_0$$

Ecuación 4.1.- Ecuación de la posición de un MRUA en un transelevador

$$v(t) = \frac{j}{2}(t - t_0)^2 + a_0(t - t_0) + v_0$$

Ecuación 4.2.- Ecuación de la velocidad de un MRUA en un transelevador

$$a(t) = j(t - t_0) + a_0$$

Ecuación 4.3.- Ecuación de la aceleración de un MRUA en un transelevador

Donde:

x, x_0 : posición del transelevador en un instante dado (x) y en su posición inicial (x_0)

j: jerk del motor del transelevador. Permanece constante.
t, t_0: intervalo de tiempo estudiado
v, v_0: velocidad del transelevador en un instante dado (v) y en el instante inicial (v_0)

a, a_0: aceleración del transelevador en un instante dado (a) y en el instante inicial (a_0)

Simplificando se obtiene:

$$v_1 = \frac{a^2}{2 \cdot j}$$

Ecuación 4.4.- Velocidad proporcionada por el jerk del motor del transelevador

$$d_1 = \frac{a^3}{6 \cdot j^2}$$

Ecuación 4.5.- Distancia recorrida por el transelevador debida al jerk del motor

Las ecuaciones obtenidas son los valores de velocidad y distancia generados por el impulso del motor (JERK)

Según estas ecuaciones pueden darse dos posibles casos:

- Caso 1: Caso en el que la velocidad máxima es 2 veces mayor que la velocidad proporcionada por el jerk ($v > 2v_1$)

    - Caso 1- a). Cuando: $d \geq 4d_1$, el tiempo total que tarda el transelevador en la realización de un ciclo es de:

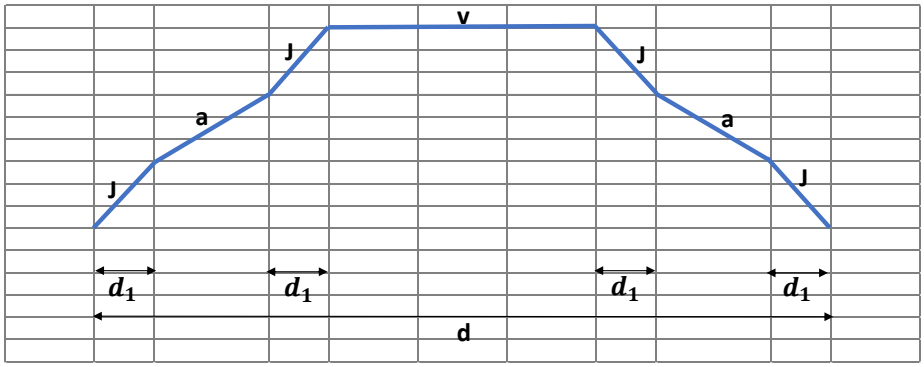


Figura 4.7.- *Caso1- a):* $v > 2v_1$ *, pero* $d \geq 4d_1$

$$t_T = \frac{d - 4 \cdot d_1}{v} + \frac{2}{a}\left(\frac{v}{2} - 2 \cdot v_1 + 2\frac{{v_1}^2}{v}\right) + \frac{4 \cdot a}{j}$$

Ecuación 1.6.- Caso1- a): Tiempo total de un ciclo de trabajo cuando $d \geq 4d_1$

- Caso 1- b). Cuando: $d < 4d_1$, el tiempo total que tarda el transelevador en la realización de un ciclo es de:

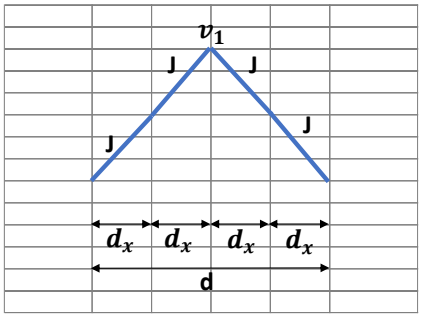


Figura 4.8.- *Caso1 – b):* $v > 2v_1$ *, pero* $d < 4d_1$

$$d_x = \frac{d}{4}$$

Ecuación 1.7.- Caso1 – b): Desplazamiento generado por el Jerk cuando $d < 4d_1$

$$t_x = \left(\frac{6d_x}{j}\right)^{1/3}$$

Ecuación 1.8.- Caso1 – b): Tiempo que tarda el traslo en recorrer d_x *cuando* $d < 4d_1$

$$t_T = 4 \cdot t_x$$

Ecuación 1.9.- Caso1 - b): Tiempo total de un ciclo de trabajo cuando $d < 4d_1$

- Caso 2: Caso en el que la velocidad que proporciona el JERK·2 es mayor que la velocidad nominal, en este caso se debe recalcular el valor del jerk. Se produce en distancias muy pequeñas. ($v < 2v_1$)

$$v_x = \frac{v}{2}$$

Ecuación 4.6.- Caso 2: Velocidad proporcionada por el jerk del motor del transelevador

$$t_x = \sqrt{\frac{2v_x}{j}}$$

Ecuación 4.7.- Caso 2: Tiempo que tarda el traslo en recorrer d_x a v_x

$$d_x = \frac{1}{6} \cdot j \cdot {t_x}^3$$

Ecuación 4.8.- Caso 2: Distancia recorrida por el traslo debido al jerk del motor

  - Caso 2 - a). Cuando: $d < 4d_x$, el tiempo total que tarda un transelevador en la realización de un ciclo es de:

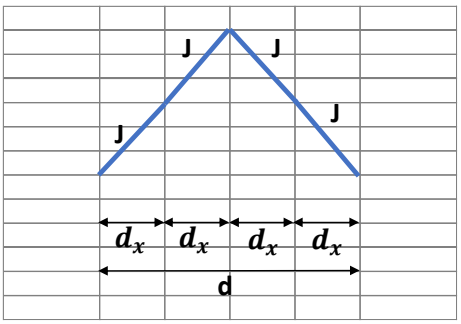


Figura 4.9.- *Caso2 - a): Tiempo total de un ciclo de trabajo cuando $d < 4d_x$*

$$d_x = \frac{d}{4}$$

Ecuación 1.13.- Caso2 - a): Distancia recorrida por el traslo debido al jerk del motor

$$t_x = \left(\frac{6d_x}{j}\right)^{1/3}$$

Ecuación 4.9.- Caso2 - a): Tiempo que tarda el traslo en recorrer d_x

$$t_T = 4 \cdot t_x$$

Ecuación 4.10.- Caso2 – a): Tiempo total de un ciclo de trabajo cuando $d < 4d_x$

- Caso 2 - b). Cuando: $d \geq 4d_x$, el tiempo total que tarda un transelevador en la realización de un ciclo es de:

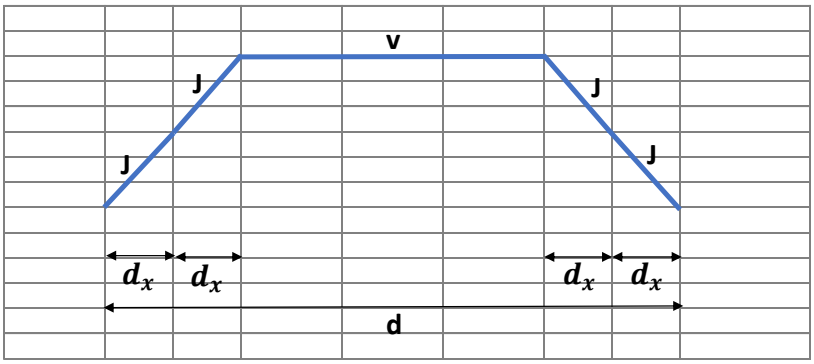


Figura 4.10.- *Caso2 - b): Tiempo total de un ciclo de trabajo cuando* $d \geq 4d_x$

$$t_x = \sqrt{\frac{2v_x}{j}}$$

Ecuación 1.16.- Caso2 - b): Tiempo que tarda el traslo en recorrer d_x *a* v_x

$$v_x = \frac{v}{2}$$

Ecuación 1.17.- Caso2 – b): Velocidad proporcionada por el jerk del motor del transelevador

$$d_x = \frac{1}{6} \cdot j \cdot {t_x}^3$$

Ecuación 4.11.- Caso 2 – b): Desplazamiento generado por el jerk del motor del transelevador

$$t_T = 4 \cdot t_x + \frac{d - 4 \cdot d_x}{v}$$

Ecuación 4.12.- Caso2 – b): Tiempo total de un ciclo de trabajo cuando $d \geq 4d_x$

4.7. - PROGRAMA DE CONTROL

Se ha desarrollado un programa de control para el secuenciador Transelevador

En el “Manual de usuario de DesignerIV” se explica de manera detallada como crear un secuenciador en IEC.

Los siguientes subapartados explican los métodos, parámetros y detalles que se deben conocer para poder implementar el programa de control.

La estructura de un secuenciador en IEC se rige por sus métodos principales, los métodos de acción, definidos en las propiedades del secuenciador. Desde ellos se realiza la invocación al reto de métodos definidos en el secuenciador. El orden de ejecución en funcionamiento normal (excluyendo arranques) en cada ciclo de programa es:

1. Acción anterior
2. Método principal IEC
3. Acción posterior

Los métodos son acciones que actúan siempre sobre el mismo secuenciador, independientemente del modo de funcionamiento activo. A continuación, se definen todos los métodos de las propiedades del secuenciador:

- Método principal IEC: método donde se define el comportamiento principal del secuenciador y desde el que se invocan al resto de métodos del mismo para definir las acciones necesarias.
- Acción anterior: acción que se ejecuta antes de evaluar el estado activo del secuenciador.
- Acción posterior: acción que se ejecuta después de evaluar el estado activo del secuenciador.
- Arranque en frío: método invocado cuando GalileoIV inicia un arranque frío (inicio del sistema). Típicamente se hacen las inicializaciones en esta acción.
- Arranque en caliente: idéntica a la anterior, pero para arranques en caliente (reinicios de GalileoIV que no implican una parada del sistema). Su cometido es muy similar a la anterior y puede usarse en los mismos casos, aunque suele realizar una inicialización más ligera que aquella.

4.7.1.1.- ARRANQUE EN FRÍO

Este método se caracteriza por ser llamado cuando arranca el sistema:

```
// ============================================================================ //
//                                START FUNCTION                                //
// ============================================================================ //

VAR_TEMP
  posX_cabecera: DINT;
  posY_cabecera: DINT;
  posX_backward : DINT;
  posY_backward : DINT;
END_VAR


CALL_INTERFACE(p_C_MapPosSC,SetDimension,3);
CALL_INTERFACE(p_C_MapPosSC,LoadAndTrack,p_Ini_MapPosSC);

posX_cabecera:=16000;
posY_cabecera:=1175;
posX_backward:=3850;
posY_backward:=900 ;


// ==================== //
// RESET MANUAL MEMORIES //
// ==================== //

// Reset movements stacker crane
p_M_ManBackward := FALSE;
p_M_ManForward := FALSE;
p_M_ManUp := FALSE;
p_M_ManDown := FALSE;

//Reset movements extractors
p_M_ManExtractorPositive := FALSE;
p_M_ManExtractorNegative := FALSE;
p_M_ManExtractorIN := FALSE;
p_M_AutoExtractorPositive := FALSE;
p_M_AutoExtractorNegative := FALSE;
p_M_AutoExtractorIN := FALSE;


p_M_NotFinishedFault:= FALSE;

      // ==================== //
      // RESET INITIALIZATION  //
      // ==================== //

      //Reset outputs
      p_OExtractor_Negative := FALSE;
      p_OExtractor_Positive := FALSE;
      p_OExtractorIN := FALSE;

      //Reset other marks
      p_Left := FALSE;
      p_Right := FALSE;

      //para generar palestas y hacerlas desaparecer en estanteria
      p_M_UnloadFinished_Shelf:=FALSE;
      p_M_LoadStart_Shelf:=FALSE;

      //indican los cambios de malla en ME y MS
      p_M_UnloadFinished :=FALSE;
      p_M_UnloadFinished_Shelf:= FALSE;
      p_M_LoadFinished :=FALSE;

      //Indican el valor lógico del lado en el que se encuentra la ME,
      //la MS y la estanteria
      //true: lado derecho; false:lado izquierdo
      p_M_SideME:= false;
      p_M_SideMS := true;
      p_M_SideShelf:=true;

      p_M_UnloadFinished_Shelf:= false;

      // ==================//
      //   RESET  DRIVERS   //
      // ================== //

      CALL_INTERFACE(p_C_DriverX, Reset);
      CALL_INTERFACE(p_C_DriverY, Reset);

     //Reseteo de otras variables del programa
       p_MB_Status_Mem:=0;
       p_M_Presence_Mem:= false;
       p_MB_Capacity_Mem:= 0;
```

```
// ============ //
// START PROGRAM //
// ============ //
//Start in Initialization if is the first program load
IF p_MW_MachineCycle = 0 THEN
  p_MW_MachineCycle:=1;
END_IF;


IF p_MW_ExtractorCycle_ME_Left =0 THEN
  p_MW_ExtractorCycle_ME_Left:=1;
END_IF;

IF p_MW_ExtractorCycle_ME_Right =0 THEN
  p_MW_ExtractorCycle_ME_Right:=1;
END_IF;

IF p_MW_ExtractorCycle_MS_Right =0 THEN
  p_MW_ExtractorCycle_MS_Right:=1;
END_IF;

IF p_MW_ExtractorCycle_MS_Left =0 THEN
  p_MW_ExtractorCycle_MS_Left:=1;
END_IF;


IF p_MW_ExtractorCycle_UnloadShelfRight =0 THEN
  p_MW_ExtractorCycle_UnloadShelfRight:=1;
END_IF;

IF p_MW_ExtractorCycle_UnloadShelfLeft =0 THEN
  p_MW_ExtractorCycle_UnloadShelfLeft:=1;
END_IF;

IF p_MW_ExtractorCycle_LoadShelfRight = 0 THEN
  p_MW_ExtractorCycle_LoadShelfRight:=1;
END_IF;

IF p_MW_ExtractorCycle_LoadShelfLeft = 0 THEN
  p_MW_ExtractorCycle_LoadShelfLeft:=1;
END_IF;
```

4.7.1.2.- ACCIÓN ANTERIOR

Método evaluado antes de conocer el estado activo del transelevador

```
//// ================================================================================ //
////                              PREVIOUS FUNCTION                                   //
//// ================================================================================ //
//
//
// Presence memories
// -----------------

// Tracking presence
p_M_TrackingPresent := TrackingCount()>0;


//AWS
p_MB_Occupation:=TrackingCount();

// Handshake con posterior.
IF (p_MDW_NextConveyor = GetNextNumber(0)) THEN
  p_M_PresenceToNext := p_I_ForwardSensor;
END_IF;


// Presencia de pallet en mesas de entrada y salida.
p_M_PalletPresence :=p_I_WeightPallet; // Code_PresenceME_TO_ST();



//Actualizamos los valores de la estacion para que no este en fallo en WCS
//y ademas deje de salir en "presencia==LLENO"
CU_WMS_Cyclic_Data(p_M_PalletPresence);
```

4.7.1.3.- MÉTODO PRINCIPAL IEC

Este método define el comportamiento principal del secuenciador, desde él se llaman a otros métodos y funciones.

```
// ============================================================================ //
//                              MAIN  FUNCTION                                  //
// ============================================================================ //


// ------ //
// Fault  //
// ------ //

IF p_M_Fault
 OR p_M_NotFinishedFault
THEN
  IF NOT p_M_NotFinishedFault THEN
   // Reset variables at entry.
   Reset_STD();
   p_M_NotFinishedFault := true;
   p_M_AnyMode := true;

  ELSIF NOT p_M_Fault
  THEN
   // Reset variables at exit.
   Reset_STD();
   p_M_NotFinishedFault := false;
  END_IF;


  // -------------- //
  // Automatic mode //
  // -------------- //

  ELSE

    CASE (p_MW_MachineCycle) OF

    //==============================//
    //  REST                        //
    //==============================//
    CASEOPTION 1:
      IF p_M_IsTransition OR p_M_AnyMode THEN
          // Entry function.
          Initialize_StackerCrane();
          p_M_IsTransition:=false;
      END_IF;
        // Previous step function.
        ;
        // Transition.
      IF TRUE THEN
          // Transition function
          ;
          // Exit function.
          Reset_STD();
          // Entry next step function.
          Prepare_MultiOP_Search();
          p_MW_MachineCycle:=10;
      ELSE
          // Step funcion:
          ;
      END_IF;
```

```
//==============================//
//  Receive Task: Search        //
//==============================//
CASEOPTION 10:
  IF p_M_IsTransition THEN
      // Entry function.
      Prepare_MultiOP_Search();
      p_M_IsTransition:= FALSE;
  END_IF;
    // Previous step function.
    ;
    // Transition.
  IF TR_MultiOP_Received() THEN
      // Transition function
      ;
      // Exit function.
      Exit_MultiOP();
      // Entry next step function.
      Tracking_Source_Management();
      p_MW_MachineCycle:= 15;
  ELSE
      // Step funcion:
      Receive_Task();
  END_IF;
```

```
//==============================//
//  Tracking Source             //
//==============================//
CASEOPTION 15:
  IF p_M_IsTransition THEN
      // Entry function.
      Tracking_Source_Management();
      p_M_IsTransition:= FALSE;
  END_IF;
    // Previous step function.
    ;
    // Transition.
  IF TestSource_PosOK() THEN
      // Transition function
      ;
      // Exit function.
      ;
      // Entry next step function.
      ;
      p_MW_MachineCycle:= 20 ;
  ELSE
      // Step funcion:
      GotoSource();
  END_IF;
```

```
//==============================//
//  load task                   //
//==============================//
CASEOPTION 20:
  IF p_M_IsTransition THEN
      // Entry function.
      ;
      p_M_IsTransition:= FALSE;
  END_IF;
    // Previous step function.
    ;
    // Transition.
  IF Source_Task() THEN
      // Transition function
      ;
      // Exit function.
      ;
      // Entry next step function.
      Tracking_Target_Management();
      p_MW_MachineCycle:=24;
  ELSE
      // Step funcion:
      ;
  END_IF;
```

```
//==============================//
//  unload task                 //
//==============================//
CASEOPTION 26:
  IF p_M_IsTransition THEN
      // Entry function.
      ;
      p_M_IsTransition:= FALSE;
  END_IF;
    // Previous step function.
    ;
    // Transition.
  IF Target_Task() THEN
      // Transition function
      ;
      // Exit function.
      ;
      // Entry next step function.
      PrepareMultiOPCommandToFinish(1, 1, 0, 0);
      p_MW_MachineCycle:=28;
  ELSE
      // Step funcion:
      ;
  END_IF;
```

```
//==============================//
//  Tracking target             //
//==============================//
CASEOPTION 24:
  IF p_M_IsTransition THEN
      // Entry function.
      Tracking_Target_Management();
      p_M_IsTransition:= FALSE;
  END_IF;
    // Previous step function.
    ;
    // Transition.
  IF TestTarget_PosOK() THEN
      // Transition function
      ;
      // Exit function.
      ;
      // Entry next step function.
      ;
      p_MW_MachineCycle:=26;
  ELSE
      // Step funcion:
      GotoTarget();
  END_IF;
```

```
//==============================//
//  END and search new task     //
//==============================//
CASEOPTION 28:
  IF p_M_IsTransition THEN
      // Entry function.
      PrepareMultiOPCommandToFinish(1, 1, 0, 0);
      p_M_IsTransition:= FALSE;
  END_IF;
    // Previous step function.
    ;
    // Transition.
  IF TR_End_MultiOP() THEN
      // Transition function
      ;
      // Exit function.
      Exit_MultiOP();
      // Entry next step function.
      Initialize_StackerCrane();
      p_MW_MachineCycle:=1;
  ELSE
      // Step funcion:
      ;
  END_IF;


ELSE// (del caseOPtion)
     p_MW_MachineCycle:=1;
     p_M_IsTransition:=true;


 END_CASE

END_IF
```

4.7.1.4.- FUNCIONES INVOCADAS DESDE LOS DIFERENTES MÉTODOS DEL SECUENCIADOR

En este subapartado se muestran las funciones que ha sido necesario implementar para gestionar las señales recibidas por el transelevador y poder responder en consecuencia a través del programa de control:

Retorno	Nombre	Parámetros
VOID	After_ExtrLeft	
VOID	After_ExtrLeft_Unload	
VOID	After_ExtrRight	
VOID	After_ExtrRight_Unload	
SINT	Code_LoadConsentToSC_ME	
BOOL	Code_PresenceME_TO_ST	
BOOL	Code_PresenceMS_TO_ST	
VOID	Code_SET_EndLoadCycleFromSC	
VOID	Code_SET_EndUnloadCycleFromSC	
VOID	Code_SET_LoadCycleFromSC	
VOID	Code_SET_UnloadCycleFromSC	
SINT	Code_UnloadConsentToSC_MS	
VOID	CU_WMS_Cyclic_Data	Presence: BOOL
VOID	Execute_Task	
VOID	Exit_MultiOP	
VOID	ExtractorLeft_OFF	
VOID	ExtractorLeft_ON	
VOID	ExtractorLeftIn_ON	
VOID	ExtractorRight_OFF	
VOID	ExtractorRight_ON	
VOID	ExtractorRightIn_ON	
VOID	GotoSource	
VOID	GotoTarget	
VOID	Initialize_StackerCrane	
VOID	Load_FromME_ToSC	
VOID	Load_FromShelf_ToSC	
VOID	LoadFromShelf	
VOID	Main_SC	
VOID	MoveLoad_Extractor_LeftME	
VOID	MoveLoad_Extractor_LeftShelf	
VOID	MoveLoad_Extractor_RightME	
VOID	MoveLoad_Extractor_RightShelf	
VOID	MoveUnload_Extractor_Left	
VOID	MoveUnload_Extractor_LeftShelf	
VOID	MoveUnload_Extractor_Right	
VOID	MoveUnload_Extractor_RightShelf	
VOID	Prepare_MultiOP_Search	
VOID	Prepare_Task_To_Finish	
VOID	Previous_SC	
VOID	Receive_Task	
VOID	Reset_ParamsExtractor	
VOID	Reset_STD	
VOID	SobreElev	
BOOL	Source_Task	
VOID	Start_SC	
BOOL	Target_Task	
DINT	TestSobreElev_OK	
BOOL	TestSource_PosOK	
BOOL	TestTarget_PosOK	
BOOL	TR_End_MultiOP	
BOOL	TR_MultiOP_Received	
VOID	Tracking_Source_Management	
VOID	Tracking_Target_Management	
VOID	Unload_FromSC_ToMS	
VOID	Unload_FromSC_ToShelf	
VOID	UnloadToShelf_Left	
VOID	UnloadToShelf_Right	

A continuación, se detallan cada una de las funciones anteriores:

procedure After_ExtrLeft

```
1
2
3   p_Left:=TRUE;
4   p_M_LoadFinished:= TRUE;
5   p_OExtractor_Negative:=FALSE;
```

procedure After_ExtrLeft_Unload

```
1   p_Left:=TRUE;
2   p_M_LoadFinished:=TRUE;
3   p_OExtractor_Negative:=FALSE;
4
5
6
```

procedure After_ExtrRight

```
1   p_Right:=TRUE;
2   p_M_LoadFinished:= TRUE;
3   p_OExtractor_Positive:=FALSE;
4
5
6
```

procedure After_ExtrRight_Unload

```
1   p_Right:=TRUE;
2   p_M_UnloadFinished:= TRUE;
3   p_OExtractor_Positive:=FALSE;
4
5
6
```

procedure CU_WMS_Cyclic_Data (Presence: BOOL)

```
1   // ===============================================================================
2   // Cyclic data to WMS
3   // ===============================================================================
4
5   VAR_TEMP
6     status     : SINT;
7     capacity : SINT;
8   END_VAR
9
10  IF (p_MB_StationNumber <> 0)
11     AND(p_MB_StationType <> 0)
12  THEN
13
14
15  //AWS (Hay que modificar la parte de falls y  manual cuando este hecha)
16      // Station status: 0 -> Error/Manual, 1 -> Automatic mode
17      status :=1;
18
19  //     IF p_M_Fault
20  //       OR p_M_ManualMode
21  //       OR p_M_ManualSlave
22  //     THEN
23  //         status := 0;
24  //     END_IF;
25
26    // Presence -> 1, without presence -> 0
27
28    // CAPACITY
29    // This value is negligible in update station status
30      capacity := 1;
31
```

```
32    // AISLE
33    // Constant value
34
35    // Timer:
36    p_M_UpdateStationStatus:= EvalTON(p_TON_UpdateStationStatus
37                                      ,NOT p_M_UpdateStationStatus
38                                      ,5000);
39
40    // Only call SetStationStatus if some variable has changed:
41  IF (p_MB_Status_Mem <> status
42      OR p_M_Presence_Mem <> Presence
43      OR p_MB_Capacity_Mem <> capacity)
44      OR p_M_UpdateStationStatus
45    THEN
46
47      SetStationStatus(
48                           p_MB_StationType, // byte
49                           p_MB_StationNumber, // byte
50                           status, // byte
51                           NOT Presence, //bool
52                           capacity, //byte
53                           p_MB_Occupation,// byte
54                           p_MB_AisleNumber //byte
55                          );
56
57      END_IF;
58
59      // Store the variables to check the next cycle.
60      p_MB_Status_Mem:= status;
61      p_M_Presence_Mem:= Presence;
62      p_MB_Capacity_Mem:= capacity;
63
```

```
64  END_IF;
65
66    // TRAKING DATA (to show in OP)
67    // ----------------------------
68    p_MDW_TransportNumber := GetTrackingTransportNumber(1);
69    p_MB_TargetStationNumber := GetTrackingTargetNumber(1);
70    p_MB_TargetStationType := GetTrackingTargetType(1);
71
```

procedure Execute_Task

```
1   /// ============================================================================
2   // Set execute task memory
3   // ============================================================================
4
5   p_M_OrderInProcess := true;
6
7   // ============================================================================
8
```

procedure Exit_MultiOP

```
1   // ============================================================================
2   // Exit multiOP protocol
3   // ============================================================================
4
5   ExitMultiOP();
6
7   // ============================================================================
8
```

procedure ExtractorLeft_OFF

```
p_OExtractor_Negative := FALSE;
```

procedure ExtractorLeft_ON

```
p_OExtractor_Negative := NOT p_I_ExtractorLeft;
```

procedure ExtractorLeftIn_ON

```
p_OExtractor_Positive := NOT p_I_ExtractorIN;
```

procedure ExtractorRight_OFF

```
p_OExtractor_Positive:=FALSE;
```

procedure ExtractorRight_ON

```
p_OExtractor_Positive :=NOT p_I_ExtractorRight;
```

procedure GotoSource

```
1
2
3   //Vamos a recoger la orden
4
5   CALL_INTERFACE(p_C_DriverX, GotoPosition,1,p_MDW_SourceXPhysical,100,50);
6   CALL_INTERFACE(p_C_DriverY, GotoPosition,1,p_MDW_SourceYPhysical,100,50);
7
8
```

procedure GotoTarget

```
1   ////Vamos a dejar la orden
2
3
4   CALL_INTERFACE(p_C_DriverX, GotoPosition,1,p_MDW_TargetXPhysical,100,50);
5   CALL_INTERFACE(p_C_DriverY, GotoPosition,1,p_MDW_TargetYPhysical,100,50);
6
7
8
9
```

procedure Initialize_StackerCrane

```
// ================================================================================
// Initialize Stacker Crane sequencer
// ================================================================================

p_OExtractorIN := false;
p_OExtractor_Negative := false;
p_OExtractor_Positive := false;

//Para ME y MS
p_MW_ExtractorCycle_ME_Left:=1;
p_MW_ExtractorCycle_ME_Right:=1;
p_MW_ExtractorCycle_MS_Left := 1;
p_MW_ExtractorCycle_MS_Right := 1;

//Para estanteria
p_MW_ExtractorCycle_UnloadShelfRight:=1;
p_MW_ExtractorCycle_UnloadShelfLeft:=1;

p_MW_ExtractorCycle_LoadShelfRight:=1;
p_MW_ExtractorCycle_LoadShelfLeft:=1;

p_M_SourceTask := false;
p_M_TargetTask := false;

p_MW_MachineCycle:=1;

//Code_SET_EndLoadCycleFromSC();

ClearAllTrackings();

```

procedure Load_FromME_ToSC

```

//comprobamos en que lado esta la mesa de entrada al alamcen

IF (p_MDW_SourceZLogic = 1) THEN //Mesa de entrada en lado izquierdo.
  MoveLoad_Extractor_LeftME();
ELSIF (p_MDW_SourceZLogic = 2) THEN
  MoveLoad_Extractor_RightME();//Mesa de entrada en lado derecho.
ELSE
  p_MW_CodeError:=2;
END_IF;

```

procedure Load_FromShelf_ToSC

```

IF (p_MDW_SourceZLogic=1) THEN //Estanteria en lado izquierdo.
  MoveLoad_Extractor_LeftShelf();
ELSIF (p_MDW_SourceZLogic=2) THEN
  MoveLoad_Extractor_RightShelf();//Estanteria en lado derecho.
ELSE
  p_MW_CodeError:=24;
END_IF;

```

procedure LoadFromShelf

```
1
2
3
4   p_M_LoadFinished_Shelf:=TRUE;
5   p_M_CreatePallet:=TRUE;
6
7   IF(p_MDW_SourceZLogic=2) THEN
8     p_Right:=TRUE;
9     p_OExtractor_Positive:=FALSE;
10  ELSIF (p_MDW_SourceZLogic=1) THEN
11    p_Left:=TRUE;
12    p_OExtractor_Negative:=FALSE;
13  END_IF;
14
```

procedure MoveLoad_Extractor_LeftME

```
1   //--------------------------//
2   // Movimiento Extractor Izq //
3   //--------------------------//
4   VAR_TEMP
5     AuxSobreElev : DINT;
6   END_VAR
7
8     CASE (p_MW_ExtractorCycle_ME_Left) OF
9
10
11    //Detectamos si hay paleta en la mesa de entrada al almacen
12    //y sacamos extractor cuando este disponible
13
14    CASEOPTION 1:
15      IF p_M_IsTransition THEN
16          // Entry function
17          ;
18          p_M_IsTransition:= FALSE;
19      END_IF;
20        // Previous step function.
21        ;
22        // Transition.
23      IF (Code_PresenceME_TO_ST() AND Code_LoadConsentToSC_ME()=1) THEN
24          // Transition function
25          Code_SET_LoadCycleFromSC();
26          // Exit function.
27          Reset_ParamsExtractor();
28          // Entry next step function.
29          ExtractorLeft_ON();
30          p_MW_ExtractorCycle_ME_Left:= 3;
```

```
31      ELSE
32          // Step funcion
33          ExtractorLeft_OFF();
34      END_IF;
35
36
37      //una vez que sacamos extractor finalizamos la orden de carga para SGA
38
39    CASEOPTION 3:
40      IF p_M_IsTransition THEN
41          // Entry function.
42          ExtractorLeft_ON();
43          p_M_IsTransition:= FALSE;
44      END_IF;
45        // Previous step function.
46        ;
47        // Transition.
48      IF (p_I_ExtractorLeft) THEN
49          // Transition function
50          ExtractorLeft_OFF();
51          // Exit function.
52          After_ExtrLeft();
53          // Entry next step function.
54          ;
55          p_MW_ExtractorCycle_ME_Left:= 4;
56      ELSE
57          // Step funcion:
58          ExtractorLeft_ON();
59      END_IF;
60
61
62  //Cuando tenemos los extractores fuera elevamos
63  //un poco la cuna con la paleta por seguridad
64
65  CASEOPTION 4:
66  IF p_M_IsTransition THEN
67      // Entry function.
68      ;
69      p_M_IsTransition:= FALSE;
70  END_IF;
71    // Previous step function.
72    ;
73    // Transition.
74  IF (p_Left) THEN
75      // Transition function
76      SobreElev();
77      // Exit function.
78      p_Left:= FALSE;
79      // Entry next step function.
80      ;
81      p_MW_ExtractorCycle_ME_Left:= 5;
82  ELSE
83      // Step funcion:
84      ;
85  END_IF;
86
87
88  //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
89
```

```
90      CASEOPTION 5:
91        IF p_M_IsTransition THEN
92            // Entry function.
93            ;
94            p_M_IsTransition:= FALSE;
95        END_IF;
96          // Previous step function.
97          ;
98          // Transition.
99        IF (TestSobreElev_OK()=1275) THEN
100           // Transition function
101           ExtractorLeft_OFF();
102           // Exit function.
103           ExtractorLeftIn_ON();
104           // Entry next step function.
105           p_MW_ExtractorCycle_ME_Left:= 6;
106       ELSE
107           // Step funcion:
108           SobreElev();
109       END_IF;
110
111
112       // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
113
114     CASEOPTION 6:
115       IF p_M_IsTransition THEN
116           // Entry function.
117           ;
118           p_M_IsTransition:= FALSE;
119       END_IF;
120         // Previous step function.
121         ;
122         // Transition.
123       IF (p_I_ExtractorIN) THEN
124           // Transition function
125           ExtractorRight_OFF();
126           //Check_PalletLoad_SC(); --AWS reseteo las variables con las que hago cambi
127           // Exit function.
128           p_M_ChangeIN:=true;
129           // Entry next step function.
130           Reset_ParamsExtractor();
131           p_MW_ExtractorCycle_ME_Left:= 7;
132       ELSE
133           // Step funcion
134           ExtractorLeftIn_ON();
135       END_IF;
136
137
138
139
140     CASEOPTION 7:
141       IF p_M_IsTransition THEN
142           // Entry function.
143           Reset_ParamsExtractor();
144           p_M_IsTransition:= FALSE;
145       END_IF;
146         // Previous step function.
147         //Code_SET_EndLoadCycleFromSC();
148         // Transition.
149       IF (Code_LoadConsentToSC_ME() =127) THEN
```

```
150         // Transition function
151         Code_SET_EndLoadCycleFromSC();
152         ME_01.p_MB_EndLoadCycleFromSC:=127;
153         // Exit function.
154         p_M_SourceTask:=TRUE;
155         // Entry next step function.
156         ;
157         p_MW_ExtractorCycle_ME_Left:=1;
158       ELSE
159         // Step funcion
160         ;
161       END_IF;
162
163
164     END_CASE;
165
166
```

procedure MoveLoad_Extractor_LeftShelf

```
1
2   //---------------------------//
3   // Movimiento Extractor Izq //
4   //---------------------------//
5
6   VAR_TEMP
7     AuxSobreElev : DINT;
8   END_VAR
9
10
11
12
13    p_M_UpdateShelfStatus:= EvalTON(p_TON_UpdateShelfStatus
14                                    ,NOT p_M_UpdateShelfStatus
15                                    ,3000);
16    p_M_UpdateShelfTime:=EvalTON(p_TON_UpdateShelfStatus
17                                    ,NOT p_M_UpdateShelfTime
18                                    ,5000);
19
20
21
22    CASE (p_MW_ExtractorCycle_LoadShelfLeft) OF
23
24
25    CASEOPTION 1:
26      IF p_M_IsTransition THEN
27          // Entry function
28          ;
29          p_M_IsTransition:= FALSE;
30      END_IF;
31        // Previous step function.
32        ;
```

```
33        // Transition.
34      IF(NOT p_I_WeightPallet )THEN//AND p_M_UpdateShelfTime) THEN
35          // Transition function
36          Reset_ParamsExtractor();
37          // Exit function.
38          ;
39          // Entry next step function.
40          ExtractorLeft_ON();
41          p_MW_ExtractorCycle_LoadShelfLeft:= 3;
42      ELSE
43          // Step funcion
44          ExtractorLeft_OFF();
45      END_IF;
46
47
48      //una vez que sacamos extractor finalizamos la orden de carga para SGA
49
50      CASEOPTION 3:
51      IF p_M_IsTransition THEN
52          // Entry function.
53          ExtractorLeft_ON();
54          p_M_IsTransition:= FALSE;
55      END_IF;
56        // Previous step function.
57        ;
58        // Transition.
59      IF (p_I_ExtractorLeft) THEN
60        // Transition function
61          ExtractorLeft_OFF();
62          // Exit function.
63          //After_ExtrLeft();
64          // Entry next step function.
65          LoadFromShelf();
66          IF(p_M_UpdateShelfStatus)THEN
67            p_MW_ExtractorCycle_LoadShelfLeft:= 4;
68          END_IF;
69      ELSE
70          // Step funcion:
71          ExtractorLeft_ON();
72      END_IF;
73
74
75      CASEOPTION 4:
76      IF p_M_IsTransition THEN
77          // Entry function.
78          ;
79          p_M_IsTransition:= FALSE;
80      END_IF;
81        // Previous step function.
82        ;
83        // Transition.
84      IF (p_Left) THEN
85          // Transition function
86          ;
87          // Exit function.
88          ;
89          // Entry next step function.
90          ;
91          p_MW_ExtractorCycle_LoadShelfLeft:= 5;
92
93      ELSE
94          // Step funcion:
95          ;
96      END_IF;
```

```
97
98
99      //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
100
101   CASEOPTION 5:
102     IF p_M_IsTransition THEN
103         // Entry function.
104         ;
105         p_M_IsTransition:= FALSE;
106     END_IF;
107       // Previous step function.
108       ;
109       // Transition.
110     IF ( p_M_LoadFinished_Shelf) THEN
111         // Transition function
112         //p_M_LoadFinished_Shelf:=false;
113         // Exit function.
114         ExtractorLeft_OFF();
115         // Entry next step function.
116         ;
117         p_MW_ExtractorCycle_LoadShelfLeft:= 6;
118     ELSE
119         // Step funcion:
120         ;
121     END_IF;
122
123
124     // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
125
126     CASEOPTION 6:
127     IF p_M_IsTransition THEN
128         // Entry function.
```

```
130         p_M_IsTransition:= FALSE;
131     END_IF;
132       // Previous step function.
133       ExtractorLeftIn_ON();
134       // Transition.
135     IF (p_I_ExtractorIN AND p_I_WeightPallet) THEN
136
137         p_Left := false; ////
138
139         // Transition function
140         ExtractorRight_OFF();
141         // Exit function.
142         //Check_PalletLoad_SC();
143         // Entry next step function.
144         Reset_ParamsExtractor();
145         p_M_SourceTask:=TRUE;
146         p_MW_ExtractorCycle_LoadShelfLeft:= 1;
147     ELSE
148         // Step funcion
149         ExtractorLeftIn_ON();
150     END_IF;
151
152   ELSE
153   // Si el caseoption vale 0 u otro valor no comprendido entre 1 y 6 dara error=0;
154   p_MW_CodeError :=0;
155
156   END_CASE;
157
158
159
```

procedure MoveLoad_Extractor_RightME

```
1
2   //--------------------------//
3   //Movimiento Extractor Dcha //
4   //--------------------------//
5
6   VAR_TEMP
7     AuxSobreElev : DINT;
8   END_VAR
9
10
11    CASE (p_MW_ExtractorCycle_ME_Right) OF
12
13
14    //Detectamos si hay paleta en la mesa de entrada al almacen
15    //y sacamos extractor cuando este disponible
16
17    CASEOPTION 1:
18      IF p_M_IsTransition THEN
19          // Entry function
20          ;
21          p_M_IsTransition:= FALSE;
22      END_IF;
23        // Previous step function.
24        ;
25        // Transition.
26      IF (Code_PresenceME_TO_ST() AND Code_LoadConsentToSC_ME()=1) THEN
27          // Transition function
28          Code_SET_LoadCycleFromSC();
29          // Exit function.
30          Reset_ParamsExtractor();
31          // Entry next step function
32          ExtractorRight_ON();
33          p_MW_ExtractorCycle_ME_Right:= 3;
34      ELSE
35          // Step funcion
36          ExtractorRight_OFF();
37      END_IF;
38
39
40      //una vez que sacamos extraztor finalizamos la orden de carga para SGA
41
42      CASEOPTION 3:
43      IF p_M_IsTransition THEN
44          // Entry function.
45          ExtractorRight_ON();
46          p_M_IsTransition:= FALSE;
47      END_IF;
48        // Previous step function.
49        ;
50        // Transition.
51      IF (p_I_ExtractorRight) THEN
52          // Transition function
53          ExtractorRight_OFF();
54          // Exit function.
55          After_ExtrRight();
56          // Entry next step function.
57          ;
58          p_MW_ExtractorCycle_ME_Right:= 4;
59      ELSE
60          // Step funcion:
```

```
61          ExtractorRight_ON();
62      END_IF;
63
64
65      //Cuando tenemos los extractores fuera elevamos un poco la cuna con la paleta p
66
67      CASEOPTION 4:
68      IF p_M_IsTransition THEN
69          // Entry function.
70          ;
71          p_M_IsTransition:= FALSE;
72      END_IF;
73        // Previous step function.
74        ;
75        // Transition.
76      IF (p_Right) THEN
77          // Transition function
78          SobreElev();
79          // Exit function.
80          p_Right:= FALSE;
81          // Entry next step function.
82          ;
83          p_MW_ExtractorCycle_ME_Right:= 5;
84      ELSE
85          // Step funcion:
86          ;
87      END_IF;
88
89
90      //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
```

```
91
92    CASEOPTION 5:
93      IF p_M_IsTransition THEN
94          // Entry function.
95          ;
96          p_M_IsTransition:= FALSE;
97      END_IF;
98        // Previous step function.
99        ;
100       // Transition.
101     IF (TestSobreElev_OK()= 1275) THEN
102         // Transition function
103         ExtractorRight_OFF();
104         // Exit function.
105         ExtractorRightIn_ON();
106         // Entry next step function.
107         p_MW_ExtractorCycle_ME_Right:= 6;
108     ELSE
109         // Step funcion:
110         SobreElev();
111     END_IF;
112
113
114     // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
115
116 CASEOPTION 6:
117     IF p_M_IsTransition THEN
118         // Entry function.
119         ;
120         p_M_IsTransition:= FALSE;
```

```
121     END_IF;
122       // Previous step function.
123       ;
124       // Transition.
125     IF (p_I_ExtractorIN) THEN
126         // Transition function
127         ExtractorLeft_OFF();
128         //Check_PalletLoad_SC(); --AWS reseteo las variables con las que hago cambi
129         // Exit function.
130         ;
131         // Entry next step function.
132         Reset_ParamsExtractor();
133         p_MW_ExtractorCycle_ME_Right:=7;
134     ELSE
135         // Step funcion
136         ExtractorRightIn_ON();
137     END_IF;
138
139
140
141
142   CASEOPTION 7:
143     IF p_M_IsTransition THEN
144         // Entry function.
145         Reset_ParamsExtractor();
146         p_M_IsTransition:=FALSE;
147     END_IF;
148       // Previous step function.
149       //Code_SET_EndLoadCycleFromSC();
150       // Transition.
```

```
151     IF (Code_LoadConsentToSC_ME() =127) THEN
152         // Transition function
153         Code_SET_EndLoadCycleFromSC();
154         ME_01.p_MB_EndLoadCycleFromSC:=127;
155         // Exit function.
156         p_M_SourceTask:=TRUE;
157         // Entry next step function.
158         ;
159         p_MW_ExtractorCycle_ME_Right:=1;
160     ELSE
161         // Step funcion
162         ;
163     END_IF;
164
165
166   END_CASE;
167
168
```

procedure MoveLoad_Extractor_RightShelf

```
1
2   //--------------------------//
3   // Movimiento Extractor Dcha//
4   //--------------------------//
5
6   VAR_TEMP
7     AuxSobreElev : DINT;
8   END_VAR
9
10
11
12
13    p_M_UpdateShelfStatus:=EvalTON(p_TON_UpdateShelfStatus
14                                   ,NOT p_M_UpdateShelfStatus
15                                   ,3000);
16    p_M_UpdateShelfTime:=EvalTON(p_TON_UpdateShelfStatus
17                                   ,NOT p_M_UpdateShelfTime
18                                   ,5000);
19
20
21
22    CASE (p_MW_ExtractorCycle_LoadShelfRight) OF
23
24
25    CASEOPTION 1:
26      IF p_M_IsTransition THEN
27          // Entry function
28          ;
29          p_M_IsTransition:= FALSE;
30      END_IF;
31        // Previous step function.
32        ;
33        // Transition.
34      IF(NOT p_I_WeightPallet )THEN//AND p_M_UpdateShelfTime) THEN
35          // Transition function
36          Reset_ParamsExtractor();
37          // Exit function.
38          ;
39          // Entry next step function.
40          ExtractorRight_ON();
41          p_MW_ExtractorCycle_LoadShelfRight:= 3;
42      ELSE
43          // Step funcion
44          ExtractorRight_OFF();
45      END_IF;
46
47
48      //una vez que sacamos extractor finalizamos la orden de carga para SGA
49
50      CASEOPTION 3:
51      IF p_M_IsTransition THEN
52          // Entry function.
53          ExtractorRight_ON();
54          p_M_IsTransition:= FALSE;
55      END_IF;
56        // Previous step function.
57        ;
58        // Transition.
59      IF (p_I_ExtractorRight) THEN
60        // Transition function
```

```
61          ExtractorRight_OFF();
62          // Exit function.
63          //After_ExtrRight();// -- AWS (revisar, afecta a cambio de mallas)
64          // Entry next step function.
65          LoadFromShelf();
66          IF(p_M_UpdateShelfStatus)THEN
67            p_MW_ExtractorCycle_LoadShelfRight:=4;
68          END_IF;
69      ELSE
70          // Step funcion:
71          ExtractorRight_ON();
72      END_IF;
73
74
75      CASEOPTION 4:
76      IF p_M_IsTransition THEN
77          // Entry function.
78          ;
79          p_M_IsTransition:=FALSE;
80      END_IF;
81        // Previous step function.
82        ;
83        // Transition.
84      IF (p_Right) THEN
85          // Transition function
86          ;
87          // Exit function.
88          ;
89          // Entry next step function.
90          ;
```

```
91          p_MW_ExtractorCycle_LoadShelfRight:=5;
92
93      ELSE
94          // Step funcion:
95          ;
96      END_IF;
97
98
99      //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
100
101   CASEOPTION 5:
102     IF p_M_IsTransition THEN
103         // Entry function.
104         ;
105         p_M_IsTransition:=FALSE;
106     END_IF;
107       // Previous step function.
108       ;
109       // Transition.
110     IF (p_M_LoadFinished_Shelf) THEN
111         // Transition function
112         //p_M_LoadFinished_Shelf:=false;
113         // Exit function.
114         ExtractorRight_OFF();
115         // Entry next step function.
116         ;
117         p_MW_ExtractorCycle_LoadShelfRight:=6;
118     ELSE
119         // Step funcion:
120         ;
```

```
121     END_IF;
122     // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
123     CASEOPTION 6:
124     IF p_M_IsTransition THEN
125         // Entry function.
126         ;
127         p_M_IsTransition:= FALSE;
128     END_IF;
129       // Previous step function.
130       ExtractorRightIn_ON();
131       // Transition.
132     IF (p_I_ExtractorIN AND p_I_WeightPallet) THEN
133         p_Right := false; ////
134
135         // Transition function
136         ExtractorLeft_OFF();
137         // Exit function.
138         //Check_PalletLoad_SC();
139         // Entry next step function.
140         Reset_ParamsExtractor();
141         p_M_SourceTask:=TRUE;
142         p_MW_ExtractorCycle_LoadShelfRight:=1;
143     ELSE
144         // Step funcion
145         ExtractorRightIn_ON();
146     END_IF;
147   ELSE
148   // Si el caseoption vale 0 u otro valor no comprendido entre 1 y 6 dara error=0;
149   p_MW_CodeError :=0;
150   END_CASE;
```

procedure MoveUnload_Extractor_Left

```
1
2   //--------------------------//
3   // Movimiento Extractor Izq //
4   //--------------------------//
5
6   VAR_TEMP
7     AuxSobreElev : DINT;
8   END_VAR
9
10
11    CASE (p_MW_ExtractorCycle_MS_Left) OF
12
13
14    //Detectamos si hay paleta en la mesa de entrada al almacen
15    //y sacamos extractor cuando este disponible
16
17    CASEOPTION 1:
18      IF p_M_IsTransition THEN
19          // Entry function
20          ;
21          p_M_IsTransition:= FALSE;
22      END_IF;
23        // Previous step function.
24        Code_SET_UnloadCycleFromSC();
25        // Transition.
26      IF (NOT Code_PresenceMS_TO_ST() AND Code_UnloadConsentToSC_MS()=1) THEN
27          // Transition function
28          Reset_ParamsExtractor();
29          // Exit function.
30          ;
```

```
31          // Entry next step function.
32          ExtractorLeft_ON();
33          p_MW_ExtractorCycle_MS_Left:=3;
34      ELSE
35          // Step funcion
36          ExtractorLeft_OFF();
37      END_IF;
38
39
40      //una vez que sacamos extractor finalizamos la orden de carga para SGA
41
42      CASEOPTION3:
43      IFp_M_IsTransitionTHEN
44          // Entry function.
45          ExtractorLeft_ON();
46          p_M_IsTransition:=FALSE;
47      END_IF;
48        // Previous step function.
49        ;
50        // Transition.
51      IF(p_I_ExtractorLeft)THEN
52        // Transition function
53          ExtractorLeft_OFF();
54          // Exit function.
55          After_ExtrLeft_Unload();
56          // Entry next step function.
57          ;
58          p_MW_ExtractorCycle_MS_Left:=4;
59      ELSE
60          // Step funcion:
```

```
61          ExtractorLeft_ON();
62      END_IF;
63
64
65      //Cuando tenemos los extractores fuera elevamos un poco la cuna con la paleta p
66
67      CASEOPTION4:
68      IFp_M_IsTransitionTHEN
69          // Entry function.
70          ;
71          p_M_IsTransition:=FALSE;
72      END_IF;
73        // Previous step function.
74        ;
75        // Transition.
76      IFp_LeftTHEN
77          // Transition function
78          ;
79          //p_M_UnloadFinished:= TRUE; --AWS: esta variable va implcita en una funcic
80          // Exit function.
81          p_Left:=FALSE;
82          // Entry next step function.
83          ;
84          p_MW_ExtractorCycle_MS_Left:=5;
85      ELSE
86          // Step funcion:
87          ;
88      END_IF;
89
90
```

```
91      //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
92
93    CASEOPTION 5:
94      IF p_M_IsTransition THEN
95          // Entry function.
96          ;
97          p_M_IsTransition:= FALSE;
98      END_IF;
99        // Previous step function.
100       ;
101       // Transition.
102     IF (Code_UnloadConsentToSC_MS() =127) THEN
103         // Transition function
104         Code_SET_EndUnloadCycleFromSC();
105         // Exit function.
106         ExtractorLeft_OFF();
107         // Entry next step function.
108         ;
109         p_MW_ExtractorCycle_MS_Left:= 6;
110     ELSE
111         // Step funcion:
112         ;
113     END_IF;
114
115
116     // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
117
118     CASEOPTION 6:
119     IF p_M_IsTransition THEN
120         // Entry function.
121         ;
122         p_M_IsTransition:= FALSE;
123     END_IF;
124       // Previous step function.
125       ExtractorLeftIn_ON();
126       // Transition.
127     IF (p_I_ExtractorIN) THEN
128         // Transition function
129         ExtractorRight_OFF();
130         // Exit function.
131         //Check_PalletLoad_SC(); -- AWS: revisar aqui hago el reseteo de variables
132         // Entry next step function.
133         Reset_ParamsExtractor();
134         p_MW_ExtractorCycle_MS_Left:= 7;
135     ELSE
136         // Step funcion
137         ExtractorLeftIn_ON();
138     END_IF;
139
140     CASEOPTION 7:
141     IF p_M_IsTransition THEN
142         // Entry function.
143         Reset_ParamsExtractor();
144         p_M_IsTransition:= FALSE;
145     END_IF;
146       // Previous step function.
147       ;
148       // Transition.
149     IF (TRUE) THEN
150         // Transition function
```

```
151         ;
152         // Exit function.
153         p_M_TargetTask:=TRUE;
154         // Entry next step function.
155         ;
156         p_MW_ExtractorCycle_MS_Left:=1;
157
158       ELSE
159         // Step funcion
160         ;
161       END_IF;
162
163
164     END_CASE;
```

procedure MoveUnload_Extractor_LeftShelf

```
1
2   //--------------------------//
3   // Movimiento Extractor izq //
4   //--------------------------//
5
6   VAR_TEMP
7     AuxSobreElev : DINT;
8   END_VAR
9
10
11
12
13    p_M_UpdateShelfStatus:=EvalTON(p_TON_UpdateShelfStatus
14                                   ,NOT p_M_UpdateShelfStatus
15                                   ,2000);
16    p_M_UpdateShelfTime:=EvalTON(p_TON_UpdateShelfStatus
17                                   ,NOT p_M_UpdateShelfTime
18                                   ,5000);
19
20
21
22    CASE (p_MW_ExtractorCycle_UnloadShelfLeft) OF
23
24
25    CASEOPTION 1:
26      IF p_M_IsTransition THEN
27          // Entry function
28          ;
29          p_M_IsTransition:=FALSE;
30      END_IF;
```

```
31      // Previous step function.
32      ;
33      // Transition.
34    IF(p_I_WeightPallet AND p_M_UpdateShelfTime ) THEN
35        // Transition function
36        Reset_ParamsExtractor();
37        // Exit function.
38        ;
39        // Entry next step function.
40        ExtractorLeft_ON();
41        p_MW_ExtractorCycle_UnloadShelfLeft:=3;
42    ELSE
43        // Step funcion
44        ExtractorLeft_OFF();
45    END_IF;
46
47
48    //una vez que sacamos extractor finalizamos la orden de carga para SGA
49
50    CASEOPTION 3:
51    IF p_M_IsTransition THEN
52        // Entry function.
53        ExtractorLeft_ON();
54        p_M_IsTransition:=FALSE;
55    END_IF;
56      // Previous step function.
57      ;
58      // Transition.
59    IF (p_I_ExtractorLeft) THEN
60      // Transition function
61        ExtractorLeft_OFF();
62        // Exit function.
63        UnloadToShelf_Left();
64        //After_ExtrLeft(); --AWS: estos seteos y reseteos estan en otra funcion
65        // Entry next step function.
66        ;
67        //p_M_UnloadFinished_Shelf:=true; --AWS
68        IF(p_M_UpdateShelfTime) THEN
69          p_MW_ExtractorCycle_UnloadShelfLeft:=4;
70        END_IF;
71    ELSE
72        // Step funcion:
73        ExtractorLeft_ON();
74    END_IF;
75
76
77    CASEOPTION 4:
78    IF p_M_IsTransition THEN
79        // Entry function.
80        ;
81        p_M_IsTransition:=FALSE;
82    END_IF;
83      // Previous step function.
84      ;
85      // Transition.
86    IF (p_Left) THEN
87        // Transition function
88        ;
89        // Exit function.
90        //p_Left:= FALSE;
```

```
91          // Entry next step function.
92          ;
93          p_MW_ExtractorCycle_UnloadShelfLeft:=5;
94
95      ELSE
96          // Step funcion:
97          ;
98      END_IF;
99
100
101     //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
102
103   CASEOPTION 5:
104     IF p_M_IsTransition THEN
105         // Entry function.
106         ;
107         p_M_IsTransition:=FALSE;
108     END_IF;
109       // Previous step function.
110       ;
111       // Transition.
112     IF (p_M_UpdateShelfTime AND p_M_UnloadFinished_Shelf) THEN
113         // Transition function
114         ;
115         // Exit function.
116         ExtractorLeft_OFF();
117         // Entry next step function.
118         ;
119         p_MW_ExtractorCycle_UnloadShelfLeft:=6;
120     ELSE
```

```
121         // Step funcion:
122         ;
123     END_IF;
124
125
126     // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
127
128     CASEOPTION 6:
129     IF p_M_IsTransition THEN
130         // Entry function.
131         ;
132         p_M_IsTransition:=FALSE;
133     END_IF;
134       // Previous step function.
135       ExtractorLeftIn_ON();
136       // Transition.
137     IF (p_I_ExtractorIN) THEN
138
139         p_Left :=false; ////
140
141         // Transition function
142         ExtractorRight_OFF();
143         // Exit function.
144         // Check_PalletLoad_SC(); -- AWS
145         // Entry next step function.
146         Reset_ParamsExtractor();
147         p_M_TargetTask:=TRUE;
148         p_MW_ExtractorCycle_UnloadShelfLeft:=1;
149     ELSE
150         // Step funcion
```

```
151         ExtractorLeftIn_ON();
152       END_IF;
153
154
155     ELSE
156     // Si el caseoption vale 0 u otro valor no comprendido entre 1 y 6 dara error=0;
157     p_MW_CodeError :=0;
158
159
160     END_CASE;
161
162
163
```

procedure MoveUnload_Extractor_Right

```
1
2   //---------------------------//
3   // Movimiento Extractor Dcha//
4   //---------------------------//
5
6   VAR_TEMP
7     AuxSobreElev : DINT;
8   END_VAR
9
10
11    CASE (p_MW_ExtractorCycle_MS_Right) OF
12
13
14    //Detectamos si hay paleta en la mesa de entrada al almacen
15    //y sacamos extractor cuando este disponible
16
17    CASEOPTION 1:
18      IF p_M_IsTransition THEN
19          // Entry function
20          ;
21          p_M_IsTransition:= FALSE;
22      END_IF;
23        // Previous step function.
24        Code_SET_UnloadCycleFromSC();
25        // Transition.
26      IF (NOT Code_PresenceMS_TO_ST() AND Code_UnloadConsentToSC_MS()=1) THEN
27          // Transition function
28          Reset_ParamsExtractor();
29          // Exit function.
30          ;
```

```
31          // Entry next step function.
32          ExtractorRight_ON();
33          p_MW_ExtractorCycle_MS_Right:=3;
34      ELSE
35          // Step funcion
36          ExtractorRight_OFF();
37      END_IF;
38
39
40      //una vez que sacamos extractor finalizamos la orden de carga para SGA
41
42      CASEOPTION 3:
43      IF p_M_IsTransition THEN
44          // Entry function.
45          ExtractorRight_ON();
46          p_M_IsTransition:=FALSE;
47      END_IF;
48        // Previous step function.
49        ;
50        // Transition.
51      IF (p_I_ExtractorRight) THEN
52        // Transition function
53          ExtractorRight_OFF();
54          // Exit function.
55          After_ExtrRight_Unload();
56          // Entry next step function.
57          ;
58          p_MW_ExtractorCycle_MS_Right:=4;
59      ELSE
60          // Step funcion:
```

```
61          ExtractorRight_ON();
62      END_IF;
63
64
65      //Cuando tenemos los extractores fuera elevamos un poco la cuna con la paleta p
66
67      CASEOPTION 4:
68      IF p_M_IsTransition THEN
69          // Entry function.
70          ;
71          p_M_IsTransition:=FALSE;
72      END_IF;
73        // Previous step function.
74        ;
75        // Transition.
76      IF p_Right THEN
77          // Transition function
78          ;
79          //p_M_UnloadFinished:= TRUE; --AWS: esta variable va implcita en una funcic
80          // Exit function.
81          p_Right:=FALSE;
82          // Entry next step function.
83          ;
84          p_MW_ExtractorCycle_MS_Right:=5;
85      ELSE
86          // Step funcion:
87          ;
88      END_IF;
89
90
```

```
91      //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
92
93    CASEOPTION 5:
94      IF p_M_IsTransition THEN
95          // Entry function.
96          ;
97          p_M_IsTransition:= FALSE;
98      END_IF;
99        // Previous step function.
100       ;
101       // Transition.
102     IF (Code_UnloadConsentToSC_MS() = 127) THEN
103         // Transition function
104         Code_SET_EndUnloadCycleFromSC();
105         // Exit function.
106         ExtractorRight_OFF();
107         // Entry next step function.
108         ;
109         p_MW_ExtractorCycle_MS_Right:= 6;
110     ELSE
111         // Step funcion:
112         ;
113     END_IF;
114
115
116     // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
117
118     CASEOPTION 6:
119     IF p_M_IsTransition THEN
120         // Entry function.
121         ;
122         p_M_IsTransition:= FALSE;
123     END_IF;
124       // Previous step function.
125       ExtractorRightIn_ON();
126       // Transition.
127     IF (p_I_ExtractorIN) THEN
128         // Transition function
129         ExtractorLeft_OFF();
130         // Exit function.
131         //Check_PalletLoad_SC(); -- AWS: revisar aqui hago el reseteo de variables
132         // Entry next step function.
133         Reset_ParamsExtractor();
134         p_MW_ExtractorCycle_MS_Right:= 7;
135     ELSE
136         // Step funcion
137         ExtractorRightIn_ON();
138     END_IF;
139
140     CASEOPTION 7:
141     IF p_M_IsTransition THEN
142         // Entry function.
143         Reset_ParamsExtractor();
144         p_M_IsTransition:= FALSE;
145     END_IF;
146       // Previous step function.
147       ;
148       // Transition.
149     IF (TRUE) THEN
150         // Transition function
```

```
151         ;
152         // Exit function.
153         p_M_TargetTask:=TRUE;
154         // Entry next step function.
155         ;
156         p_MW_ExtractorCycle_MS_Right:=1;
157
158       ELSE
159         // Step funcion
160         ;
161       END_IF;
162
163
164     END_CASE;
165
166
```

procedure MoveUnload_Extractor_RightShelf

```
1
2   //--------------------------//
3   // Movimiento Extractor Dcha//
4   //--------------------------//
5
6   VAR_TEMP
7     AuxSobreElev : DINT;
8   END_VAR
9
10
11
12
13    p_M_UpdateShelfStatus:=EvalTON(p_TON_UpdateShelfStatus
14                                   ,NOT p_M_UpdateShelfStatus
15                                   ,2000);
16    p_M_UpdateShelfTime:=EvalTON(p_TON_UpdateShelfStatus
17                                   ,NOT p_M_UpdateShelfTime
18                                   ,5000);
19
20
21
22    CASE (p_MW_ExtractorCycle_UnloadShelfRight) OF
23
24
25    CASEOPTION 1:
26      IF p_M_IsTransition THEN
27          // Entry function
28          ;
29          p_M_IsTransition:=FALSE;
30      END_IF;
```

```
31        // Previous step function.
32        ;
33        // Transition.
34      IF(p_I_WeightPallet AND p_M_UpdateShelfTime ) THEN
35          // Transition function
36          Reset_ParamsExtractor();
37          // Exit function.
38          ;
39          // Entry next step function.
40          ExtractorRight_ON();
41          p_MW_ExtractorCycle_UnloadShelfRight:=3;
42      ELSE
43          // Step funcion
44          ExtractorRight_OFF();
45      //END_IF;
46      END_IF;
47
48
49      //una vez que sacamos extractor finalizamos la orden de carga para SGA
50
51      CASEOPTION 3:
52      IF p_M_IsTransition THEN
53          // Entry function.
54          ExtractorRight_ON();
55          p_M_IsTransition:=FALSE;
56      END_IF;
57        // Previous step function.
58        ;
59        // Transition.
60      IF (p_I_ExtractorRight)THEN// AND p_M_UpdateShelfTime) THEN
```

```
61        // Transition function
62          ExtractorRight_OFF();
63          // Exit function.
64          //After_ExtrRight(); --AWS: estos seteos y reseteos estan en otra funcion
65          // Entry next step function.
66          UnloadToShelf_Right();
67          //p_M_UnloadFinished_Shelf:=true; --AWS
68          IF(p_M_UpdateShelfTime) THEN
69            p_MW_ExtractorCycle_UnloadShelfRight:=4;
70          END_IF;
71      ELSE
72          // Step funcion:
73          ExtractorRight_ON();
74      END_IF;
75
76
77      CASEOPTION 4:
78      IF p_M_IsTransition THEN
79          // Entry function.
80          ;
81          p_M_IsTransition:=FALSE;
82      END_IF;
83        // Previous step function.
84        ;
85        // Transition.
86      IF (p_Right) THEN
87          // Transition function
88          ;
89          // Exit function.
90          //p_Right:= FALSE;
```

```
91          // Entry next step function.
92          ;
93          p_MW_ExtractorCycle_UnloadShelfRight:=5;
94
95      ELSE
96          // Step funcion:
97          ;
98      END_IF;
99
100
101     //fin de sobreelevacion y posicionamos paleta sobre la cuna del SC
102
103   CASEOPTION 5:
104     IF p_M_IsTransition THEN
105         // Entry function.
106         ;
107         p_M_IsTransition:=FALSE;
108     END_IF;
109       // Previous step function.
110       ;
111       // Transition.
112     IF (p_M_UpdateShelfTime AND p_M_UnloadFinished_Shelf) THEN
113         // Transition function
114         ;
115         // Exit function.
116         ExtractorRight_OFF();
117         // Entry next step function.
118         ;
119         p_MW_ExtractorCycle_UnloadShelfRight:=6;
120     ELSE
```

```
121         // Step funcion:
122         ;
123     END_IF;
124
125
126     // Una vez que tenemos la paleta en la cuna finalizamos el ciclo
127
128     CASEOPTION 6:
129     IF p_M_IsTransition THEN
130         // Entry function.
131         ;
132         p_M_IsTransition:=FALSE;
133     END_IF;
134       // Previous step function.
135       ExtractorRightIn_ON();
136       // Transition.
137     IF (p_I_ExtractorIN) THEN
138
139         p_Right := false; ////
140
141         // Transition function
142         ExtractorLeft_OFF();
143         // Exit function.
144         //Check_PalletLoad_SC(); -- AWS
145         // Entry next step function.
146         Reset_ParamsExtractor();
147         p_M_TargetTask:=TRUE;
148         p_MW_ExtractorCycle_UnloadShelfRight:=1;
149     ELSE
150         // Step funcion
```

```
151         ExtractorRightIn_ON();
152       END_IF;
153
154
155     ELSE
156     // Si el caseoption vale 0 u otro valor no comprendido entre 1 y 6 dara error=0;
157     p_MW_CodeError :=0;
158
159
160     END_CASE;
161
162
163
```

procedure Prepare_MultiOP_Search

```
1   //=============================================================================
2   // SEARCH data parameters MultiOP  (PUB_GALILEO40)
3   //=============================================================================
4   ////p_M_Routine_Active := FALSE;
5   //IF(NOT p_Test_M_ActivarRutina)THEN
6   //  p_Test_MDW_RoutineSubTotalMoves := 0;
7   //END_IF;
8   // Receive task with current position values
9   SetMultiOPSearchStationNumber(p_MB_StationNumber); // Station number
10  SetMultiOPSearchStationType (1);        // Station type
11  SetMultiOPSearchTransport  (0);       // Always 0
12  SetMultiOPSearchCurrentX   (p_MDW_SourceXLogic); // Current location X
13  SetMultiOPSearchCurrentY   (p_MDW_SourceYLogic); // Current location Y
14  SetMultiOPSearchCurrentSide (p_MDW_SourceZLogic); // Current location Z
15  SetMultiOPSearchCurrentAisle (10);// (p_MB_AisleNumber);       // Current aisle
16  SetMultiOPSearchCurrentState (83);//(p_MB_CurrentStateSC); // Current state
17  SetMultiOPSearchCapacity   (1);   // Capacity
18  SetMultiOPSearchOccupation  (0);  // Current occupation
19
20
21  //AWS
22  ////Test tiempos Búsqueda de Orden
23  //CU_Reset_TimeStamp_Counters();
24  //p_MDW_GetTimeStamp_SearchOrder   := p_MDW_TimeStamp_10ms_Counter;
25  //p_MDW_GetTimeStamp_OrderReceived := 0;
26  //p_MDW_GetTimeStamp_XY_Mov_Start  := 0;
27  //
28
29  //=============================================================================
```

procedure Prepare_Task_To_Finish

```
// ================================================================================
// Notify extraction error with end task
// ================================================================================
VAR_TEMP
  aux:DINT;// without service
END_VAR;
// To initialize
ExitMultiOP();
aux := 0;
// END data parameters
// PrepareMultiOPCommandToFinish(byte slot, byte tracking, dword errorCode, dword aux)
SetMultiOPEndRealStationType(1, p_MB_StationType);  // Type is LZ
SetMultiOPEndRealStationNumber(1, p_MB_StationNumber); // Number
//SetMultiOPEndPlatformType(1, GetTrackingWTUType(1));
SetMultiOPEndHeightType(1, GetTrackingHeightType(1));
SetMultiOPEndLogicalX(1, p_MDW_CurrentPositionX);
SetMultiOPEndLogicalY(1,p_MDW_CurrentPositionY);

SetMultiOPEndAisle(1, 10);//p_MB_NumAisle);

//SetMultiOPEndLocalX(1, 1);
//SetMultiOPEndLocalY(1, 1);
//SetMultiOPEndAux(1, 0, 0);

IF (p_MW_CodeError <> 0 ) THEN
  PrepareMultiOPCommandToFinish (1, 1, p_MW_CodeError, aux);
END_IF;

// ================================================================================
```

procedure Receive_Task

```
// ================================================================================
// Search order by multioperation process

// Se comprueba si se ha solicitado el acceso a pasillo

// Parámetros Execute MultiOP:
// 1º - Búsqueda de orden
// 2º - Número de órdenes a finalizar
// 3º - Número de eventos

ExecuteMultiOP(true, 0, 0);

// ================================================================================
```

procedure Reset_ParamsExtractor

```
1
2
3   //reseteamos las marcas del movimiento de extractores
4   p_Left:=FALSE;
5   p_Right := FALSE;
6
7   //Reseteamos la varible que nos da si hay paleta en la mesa de entrada, en la Previous
8   //p_M_PalletPresence:=FALSE;
9   //p_M_TargetTask:=false;
10
11
12  //variable que actualiza el estado del proceso
13  p_M_LoadFinished:= false;
14  p_M_UnloadFinished:=false;
15  p_M_UnloadFinished_Shelf:=false;
16  p_M_LoadFinished_Shelf:=false;
17  //p_M_CreatePallet:=false;
18  p_M_ChangeIN:=false;
19
```

procedure Reset_STD

```
1
2
3   SetMultiOPSearchStationNumber(p_MB_StationNumber); // Station number
4   SetMultiOPSearchStationType(p_MB_StationType); // Station type
5   SetMultiOPSearchTransport(0); // Transport number
6   SetMultiOPSearchCurrentX(1); // Current X position
7   SetMultiOPSearchCurrentY(1); // Current Y position
8   SetMultiOPSearchCurrentSide(0); // Current Z position
9   SetMultiOPSearchCurrentAisle(10); // Current aisle
10  SetMultiOPSearchCurrentState(83);  // Current conveyor state
11  SetMultiOPSearchCapacity(1);  // Conveyor capacity
12  SetMultiOPSearchOccupation(0); // Conveyor occupation
13
14
15  //AWS - Occupation
16  //p_MB_Occupation:=TrackingCount();
17
18  ClearAllTrackings();
19
```

procedure SobreElev

```
CALL_INTERFACE(p_C_DriverY, GotoPosition,1,1275,100,50);
```

function Source_Task : BOOL

```
1   // ================================================================================
2   // Evaluate Trackings function type
3   // ================================================================================
4
5   IF (p_MDW_SourceXLogic=1 AND p_MDW_SourceYLogic=0 ) THEN //de mesa de entrada ME
6       Load_FromME_ToSC();
7   ELSIF (p_MDW_SourceXLogic>=2 AND p_MDW_SourceXLogic<=11) AND (p_MDW_SourceYLogic>=1 AND p_
8       Load_FromShelf_ToSC();
9   ELSE
10      p_MW_CodeError:=4;
11  END_IF;
12
13
14  IF (p_M_SourceTask) THEN
15    Source_Task:=TRUE;
16  ELSIF (NOT p_M_SourceTask) THEN
17    Source_Task:=FALSE;
18  END_IF;
```

function Target_Task : BOOL

```
// ================================================================================
// Evaluate Trackings function type
// ================================================================================


IF (p_MDW_TargetXLogic=1 AND p_MDW_TargetYLogic=0 ) THEN //descargamos en MS
//                                                      (mesa de salida de almacen)
  Unload_FromSC_ToMS();
ELSIF (p_MDW_TargetXLogic>=2 AND p_MDW_TargetXLogic<=11) AND (p_MDW_TargetYLogic>=1
    AND p_MDW_TargetYLogic<=10) THEN //descargamos en estanteria
  Unload_FromSC_ToShelf();
ELSE
  p_MW_CodeError:=5;
END_IF;



IF(p_M_TargetTask)THEN
  Target_Task:=TRUE;
ELSIF(NOT p_M_TargetTask) THEN
  Target_Task:=FALSE;
END_IF;
```

function TestSobreElev_OK : DINT

```
TestSobreElev_OK:= CALL_INTERFACE(p_C_DriverY, GetCurrentPosition, 1,1);
```

function TestSource_PosOK : BOOL

```
//================================================================================
// MACHINE + Stacker Crane position ready to process
//================================================================================

//Leemos las variables
p_MDW_CurrentPositionX :=CALL_INTERFACE(p_C_DriverX, GetCurrentPosition, 1,1);
p_MDW_CurrentPositionY :=CALL_INTERFACE(p_C_DriverY, GetCurrentPosition, 1,1);


//Si estamos bien posicionados validamos la operativa
IF((p_MDW_CurrentPositionX=p_MDW_SourceXPhysical)
    AND (p_MDW_CurrentPositionY=p_MDW_SourceYPhysical)) THEN
  TestSource_PosOK:=TRUE;
ELSE
  TestSource_PosOK:=FALSE;
END_IF;
```

function TestTarget_PosOK : BOOL

```
//================================================================================
// MACHINE + Stacker Crane position ready to process
//================================================================================

//Leemos las variables
p_MDW_CurrentPositionX :=CALL_INTERFACE(p_C_DriverX, GetCurrentPosition, 1,1);
p_MDW_CurrentPositionY :=CALL_INTERFACE(p_C_DriverY, GetCurrentPosition, 1,1);

IF((p_MDW_CurrentPositionX=p_MDW_TargetXPhysical)
    AND (p_MDW_CurrentPositionY=p_MDW_TargetYPhysical)) THEN
  TestTarget_PosOK:=true;
ELSE
  TestTarget_PosOK:=false;
END_IF;
```

function TR_End_MultiOP : BOOL

```
// ================================================================================
// End multiOP protocol
// ================================================================================

// Reset memory
p_MB_NewTracking := 0;
ExecuteMultiOP(false, 1, 0); // One FO cradle 1 (END OK or ERROR)
TR_End_MultiOP := IsMultiOPFinished(p_MB_NewTracking);

//p_MB_Occupation:=0;//Reseteamos el valor de la ocupacion
```

function TR_MultiOP_Received : BOOL

```
// ================================================================================
// End multiOP protocol
// ================================================================================

// Reset memory


p_MB_NewTracking := 0;

TR_MultiOP_Received:=IsMultiOPFinished (p_MB_NewTracking);

RETURN;

// ================================================================================

```

procedure Tracking_Source_Management

```
//=====================================================
// Busqueda de orden
//=====================================================
VAR_TEMP
  AuxSource : BOOL;
END_VAR


//Tomamos el valor de X, Y, Z (valores lógicos)
p_MDW_SourceXLogic := GetTrackingSourceX(1);
p_MDW_SourceYLogic := GetTrackingSourceY(1);
p_MDW_SourceZLogic := GetTrackingSourceSide(1);


//Sacamos las equivalencias de el valor de X, Y, Z (valores en mm)
AuxSource := CALL_INTERFACE(p_C_MapPosSC,Position3D, p_MDW_SourceXLogic,
                  p_MDW_SourceYLogic,
                  p_MDW_SourceZLogic,
                  p_MDW_SourceXPhysical,//RackType
                  p_MDW_SourceYPhysical,//RackHeight
                  p_MDW_SourceZPhysical);//RackEnabled

IF (AuxSource) THEN
  p_MW_CodeError:=8;
END_IF;

```

procedure Tracking_Target_Management

```
VAR_TEMP
  AuxTarget : BOOL;
END_VAR



//Tomamos el valor de X, Y, Z (valores lógicos)
p_MDW_TargetXLogic := GetTrackingTargetX(1);
p_MDW_TargetYLogic := GetTrackingTargetY(1);
p_MDW_TargetZLogic := GetTrackingTargetSide(1);



//Sacamos las equivalencias de el valor de X, Y, Z (valores en mm)
AuxTarget := CALL_INTERFACE(p_C_MapPosSC,Position3D, p_MDW_TargetXLogic,
                 p_MDW_TargetYLogic,
                 p_MDW_TargetZLogic,
                 p_MDW_TargetXPhysical,//RackType
                 p_MDW_TargetYPhysical,//RackHeight
                 p_MDW_TargetZPhysical);//RackEnabled

IF (AuxTarget) THEN
  p_MW_CodeError:=4;
END_IF;
```

procedure Unload_FromSC_ToMS

```
IF (p_MDW_TargetZLogic=2) THEN//Mesa de salida en lado derecho.
  MoveUnload_Extractor_Right();
ELSIF (p_MDW_TargetZLogic=1) THEN
  MoveUnload_Extractor_Left();//Mesa de salida en lado Izquierdo.
ELSE
  p_MW_CodeError:=2;
END_IF;
```

procedure Unload_FromSC_ToShelf

```
IF (p_MDW_TargetZLogic=2) THEN//Mesa de salida en lado derecho.
  MoveUnload_Extractor_RightShelf();
ELSIF (p_MDW_TargetZLogic=1) THEN
  MoveUnload_Extractor_LeftShelf();//Mesa de salida en lado Izquierdo.
ELSE
  p_MW_CodeError:=2;
END_IF;
```

procedure UnloadToShelf_Left

```


p_Left:=TRUE;
p_M_UnloadFinished_Shelf:=TRUE;
p_OExtractor_Negative:=FALSE;
```

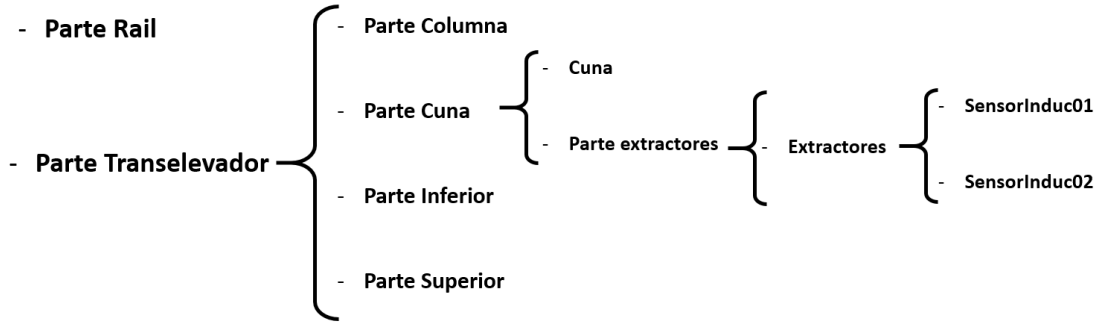
procedure UnloadToShelf_Right

```
p_Right:=TRUE;
p_M_UnloadFinished_Shelf:=TRUE;
p_OExtractor_Positive:=FALSE;


```

5. ANÁLISIS Y DESARROLLO DEL MODELO DE SIMULACIÓN

5.1. - ELEMENTOS PRESENTES EN EL MODELO

Inicialmente, para realizar el modelo de simulación y su visualización, se partirá de un modelo de mallas presentado en el programa AC3D, cada una de las mallas conformarán los cada uno de los elementos del modelo:



La visualización del modelo de mallas en AC3D es la que se presenta a continuación

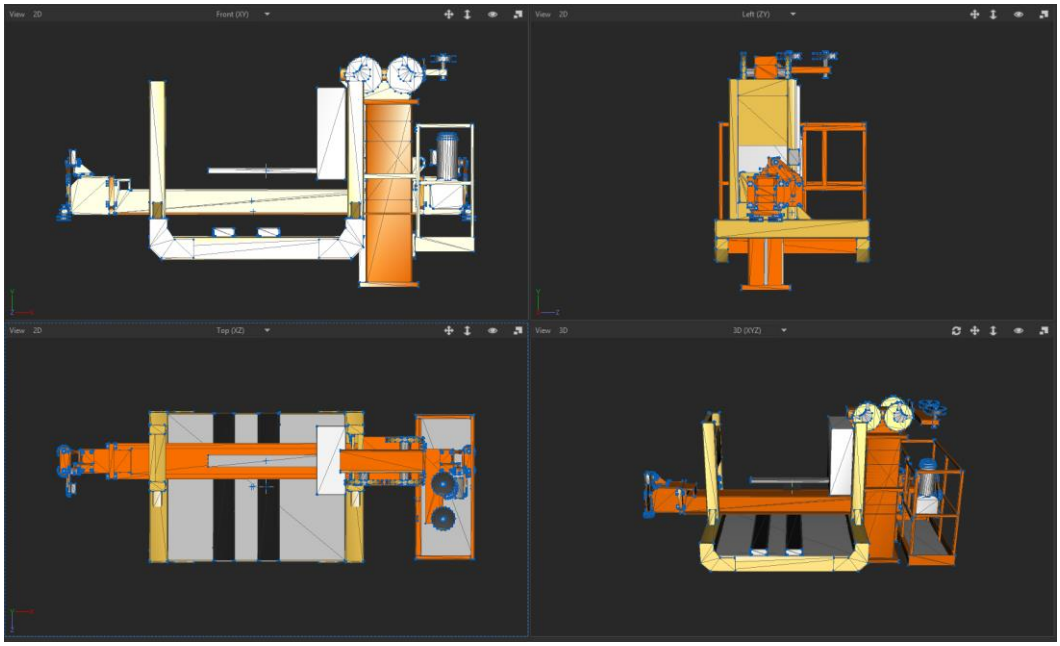


Figura 5.1.- *Modelo del transelevador en el software AC3D*

Mediante la utilización de los elementos citados en el esquema anterior, podremos tener acceso y control sobre ellos a la hora de programar sus movimientos.

1. Parte rail ("Rail"): Se fija a la solera mediante traviesas normalizadas compuestas de placa de apoyo, intercaladores elásticos, clips de fijación y pernos de anclaje. Estas traviesas se distribuyen a lo largo del raíl a intervalos regulares cuya distancia depende de la carga por rueda, del paso entre ruedas y de la longitud de los tramos de raíl. En los extremos del pasillo se colocan, las traviesas, más cercanas con el fin de absorber las elevadas reacciones de las ruedas en el caso de impacto con los amortiguadores hidráulicos. Desde el punto de vista del modelo de simulación es un elemento estático. Pieza del mecanismo con forma de carril por la cual se desliza el transelevador, para impedir que se desvíe, en el eje X
2. Parte transelevador ("StackerCrane"): Se conoce como transelevadores a las modernas grúas apiladoras, generalmente automáticas, que operan en los silos de altura manipulando las unidades de carga: recogiéndolas de cabecera para depositarlas en las ubicaciones previstas y extrayéndolas de la estantería para dejarlas en los transportadores o puestos de salida. Se diseñan siguiendo la normativa internacional aplicable a este tipo de máquinas e instalaciones: Directiva de Máquinas 98/37/CE, normas armonizadas EN y normas FEM. Todas ellas inciden especialmente en calidad, ergonomía y, sobre todo, seguridad para personas e instalaciones. Es especialmente importante la EN 528. Transelevadores. Seguridad, donde se detalla exhaustivamente toda la tipología de riesgos específicos de los transelevadores y las soluciones que deben presentar frente a ellos. Su modelo de simulación consta de un conjunto formado por toda la estructura del transelevador (Se compone de subconjuntos perfectamente diferenciados: columna, cuna, parte superior e inferior), permite al modelo el movimiento en el eje X
    2.1. Parte columna ("ColumnPart"): Las columnas pueden estar formadas por un tubo estructurado bien por vigas cajón. La columna lleva atornillados carriles verticales para el guiado de la cuna de elevación. Es la columna vertebral del transelevador, sirve de carril de movimiento a la cuna, facilitándole el movimiento en el eje Y. Su altura dependerá de la altura de la estantería.
    2.2. Parte cuna ("CradlePart"): es la encargada de desplazar la unidad de carga en sentido vertical, de realizar la extracción y depósito, mediante el mecanismo de manipulación de cargas en este caso, 2 extractores. Es una parte conformada por los extractores y las partes metálicas utilizadas para dar señales a los sensores inductivos
        2.2.1. Sistema extractor ("ExtractorPart"): engloba dos partes claramente diferenciadas, los extractores y las placas metálicas.
            2.2.1.1. Extractores ("Extractor"): los extractores u horquillas telescópicas de simple profundidad, son un mecanismo de manipulación horizontal que permite depositar o extraer unidades de carga en estanterías de simple fondo. La horquilla telescópica está compuesta por dos brazos unidos entre sí mediante un árbol de transmisión, para evitar tensiones.

2.2.1.2. Metal para activación de sensores (“MetalDetection_01” y “MetalDetection_02”): las placas metalizas perimten la activación/desactivación de los sensores inductivos.

2.3. Parte inferior (“InferiorPart”): Testero o bastidor inferior. Es una estructura soldada generalmente en forma de cajón, realizada con chapas de acero, resistente a torsión y flexión.

El guiado longitudinal del transelevador lo realizan dos parejas de rodillos de apoyo en disposición horizontal que trabajan sobre la cabeza del raíl. Cada pareja de rodillos se coloca en el extremo de los cabezales y aseguran una traslación precisa ajustando su holgura con el rail. Por delante de ellas se ubican limpiavías que eliminan la suciedad o elementos extraños de la superfície del raíl – restos de exfoliación, de obra, etc-. En la parte inferior del testero, dos parejas de tenazas de raíl evitan posibles descarrilamientos en caso de rotura o pérdida de alguna rueda de guía

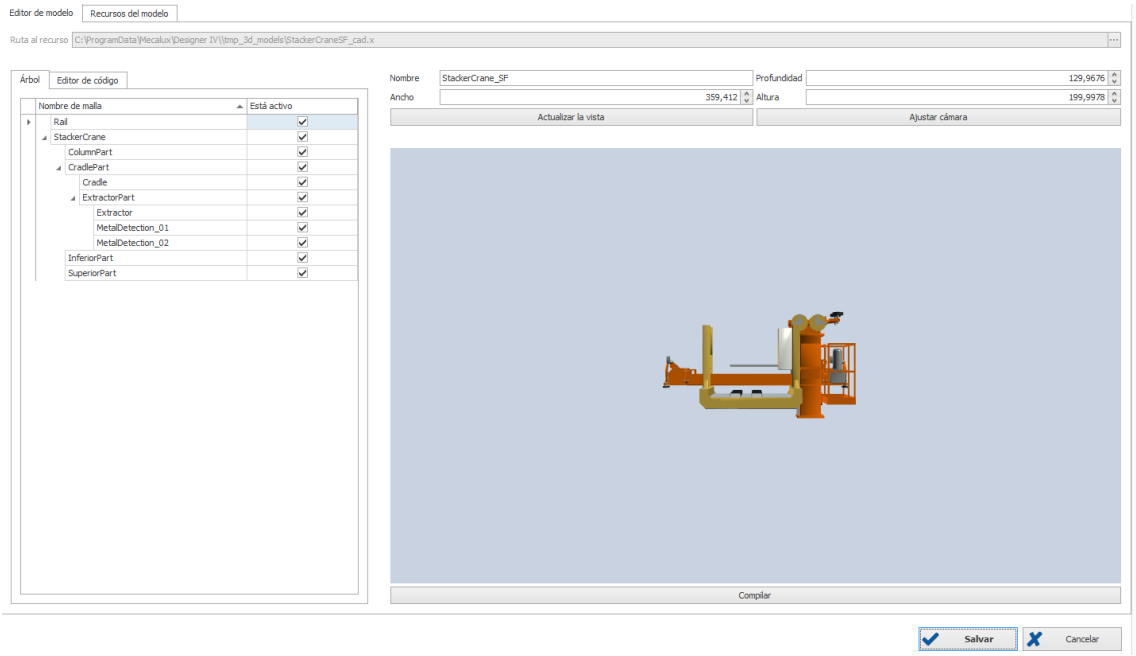
Es decir, se trata de un elemento que estará fijado a la columna central del transelevador por la parte inferior, es la parte que conecta con el rail

2.4. Parte superior (“SuperiorPart”): Testero superior. Es un elemento formado por placas soldadas, situadas en el extremo superior de la columna, que sirven de soporte para ruedas horizontales de gía sobre el carril superior. En este modelo, será un mero elemento característico sin funcionalidad propia.

Mediante la utilización de varios de estos elementos de simulación, se lleva a cabo el diseño de un modelo que pueda imitar el comportamiento de un transelevador.

5.2. - GENERACIÓN DEL MODELO 3D

La generación del modelo 3D del transelevador se hará en DesignerIV, mediante el modelo de mallas creado en AC3D. El archivo generado con la extensión “.x” se cargará en Designer y obtendremos lo siguiente:



Pestaña Árbol:

En esta pestaña se recogen todas las mallas presentes en el modelo. Las mallas que se seleccionen son las que estarán visibles posteriormente en la pestaña "Simulación" de un secuenciador al seleccionar un modelo. Por tanto, si se quiere disponer de una malla determinada para implementar un comportamiento específico del modelo del secuenciador, se debe marcar esa casilla para seleccionarla.

Pestaña editor de código:

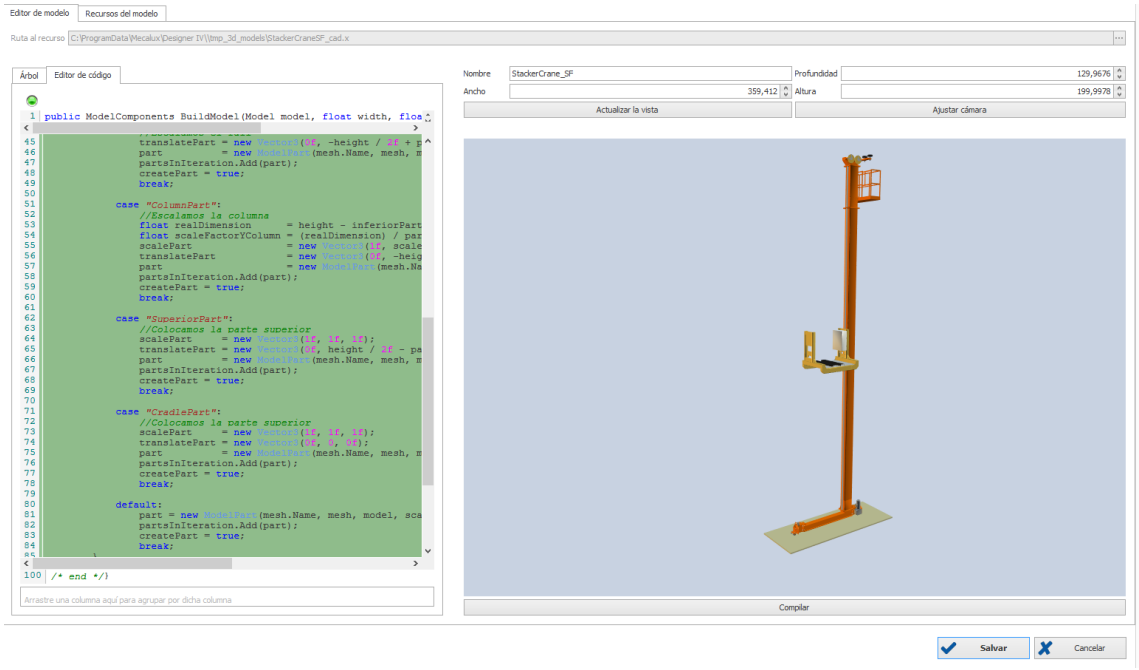
En este editor se encuentra el script de montaje del modelo. Aquí se aplican todas las transformaciones necesarias a las diferentes piezas que conforman el modelo para que tenga la representación y escalado adecuados en la simulación.

A continuación, se muestra el script de montaje del modelo del transelevador:

```
public ModelComponents BuildModel(Model model, float width, float height, float depth, object aditionalData)
{

    height = 1500f;

    List<ModelPart> parts = new List<ModelPart>();

    Matrix[] boneTransforms = new Matrix[model.Bones.Count];
    model.CopyAbsoluteBoneTransformsTo(boneTransforms);

    ///Por defecto usaremos un vertex buffer unicamente.
    ///En los pie se usara un modelo por partes

    ModelMesh partmesh           = model.Meshes.FirstOrDefault(m => m.Name == "InferiorPart");
    Vector3 inferiorPartDimension = VertexIndexBufferBuilder.ExtractDimensions(partmesh);
    partmesh                     = model.Meshes.FirstOrDefault(m => m.Name == "SuperiorPart");
    Vector3 superiorPartDimension = VertexIndexBufferBuilder.ExtractDimensions(partmesh);
    partmesh                     = model.Meshes.FirstOrDefault(m => m.Name == "RailPart");

```

```
20
21      foreach (ModelMesh mesh in model.Meshes)
22      {
23          ModelPart part = null;
24          List<ModelPart> partsInIteration = new List<ModelPart>();
25
26          Vector3 scalePart      = Vector3.One;
27          Vector3 translatePart  = Vector3.Zero;
28          Quaternion rotatePart  = Quaternion.Identity;
29          bool createPart        = false;
30          Vector3 partDimensions = VertexIndexBufferBuilder.ExtractDimensions(mesh);
31          switch (mesh.Name)
32          {
33              case "Rail":
34                  //Escalamos el rail
35                  float scaleFactorXFarSizeable = (width) / partDimensions.X;
36                  scalePart                     = new Vector3(scaleFactorXFarSizeable, 1f, 1f);
37                  translatePart                 = new Vector3(0f, -height / 2f + partDimensions.Y / 2f, 0f);
38                  part = new ModelPart(mesh.Name, mesh, model, scalePart, translatePart, rotatePart);
39                  partsInIteration.Add(part);
40                  createPart = true;
41                  break;
42
43              case "InferiorPart":
44                  //Escalamos el rail
45                  translatePart = new Vector3(0f, -height / 2f + partDimensions.Y / 2f, 0f);
46                  part          = new ModelPart(mesh.Name, mesh, model, scalePart, translatePart, rotatePart);
47                  partsInIteration.Add(part);
48                  createPart = true;
49                  break;
50
51              case "ColumnPart":
52                  //Escalamos la columna
53                  float realDimension      = height - inferiorPartDimension.Y / 2f - superiorPartDimension.Y;
54                  float scaleFactorYColumn = (realDimension) / partDimensions.Y;
55                  scalePart                = new Vector3(1f, scaleFactorYColumn, 1f);
56                  translatePart            = new Vector3(0f, -height / 2f + inferiorPartDimension.Y + realDimension / 2f, 0f);
57                  part                     = new ModelPart(mesh.Name, mesh, model, scalePart, translatePart, rotatePart);
58                  partsInIteration.Add(part);
59                  createPart = true;
60                  break;
61
62              case "SuperiorPart":
63                  //Colocamos la parte superior
64                  scalePart     = new Vector3(1f, 1f, 1f);
65                  translatePart = new Vector3(0f, height / 2f - partDimensions.Y / 2f, 0f);
66                  part          = new ModelPart(mesh.Name, mesh, model, scalePart, translatePart, rotatePart);
67                  partsInIteration.Add(part);
68                  createPart = true;
69                  break;
71              case "CradlePart":
72                  //Colocamos la parte superior
73                  scalePart     = new Vector3(1f, 1f, 1f);
74                  translatePart = new Vector3(0f, 0, 0f);
75                  part          = new ModelPart(mesh.Name, mesh, model, scalePart, translatePart, rotatePart);
76                  partsInIteration.Add(part);
77                  createPart = true;
78                  break;
79
80              default:
81                  part = new ModelPart(mesh.Name, mesh, model, scalePart, translatePart, rotatePart);
82                  partsInIteration.Add(part);
83                  createPart = true;
84                  break;
85          }
86          //Creo un ModelPart con la malla
87          //Siempre se creara con escala uno y con rotacion y posicion Zero. Sera el nodo quien lo posicione o lo escale
88          if (createPart)
89              parts.AddRange(partsInIteration);
90
91      }
92
93      ModelComponents res = new ModelComponents();
94      res.AddModelParts(parts);
95      res.VertexBuffer = null;
96      res.IndexBuffer  = null;
97      res.DrawMode     = DrawMode.Parts;
98      res.GenerateBoneTransforms(model);
99      return res;
100 /* end */}
```

Tras compilar este código se obtiene el modelo del transelevador con sus dimensiones reales:



5.3. - PROGRAMA DEL MODELO DE SIMULACIÓN

El programa de modelo de simulación consiste en crear una simulación o un modelo de simulación asociado a un secuenciador

En el programa de simulación se realizará la definición de los diferentes modelos 3D (de cadenas, rodillos, etc) asociados a un secuenciador, para que cuando éste sea utilizado en visualizaciones con representación 3D, se enlace y aparezca automáticamente dicho modelo con el funcionamiento previamente definido en esta pestaña.

A continuación, se muestran los scripts de simulación del transelevador:

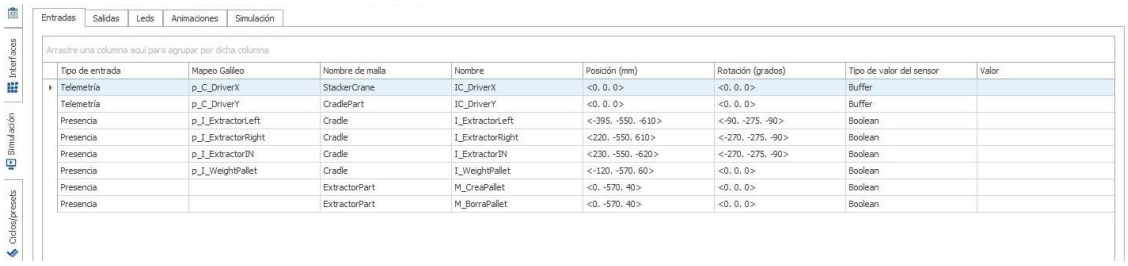
5.3.1.- SCRIPT DE INICIALIZACIÓN DE VALORES DE LAS ANIMACIONES

```
public static void StackerCrane_SF_InitialScript(StackerCrane_SF model)
{
    // Initialize frecuency inverter_Y values:
    FrequencyInverterHelper.SetCtrlInhibit((byte[])model.IC_DriverY.Value, false);
    FrequencyInverterHelper.SetFailure((byte[])model.IC_DriverY.Value, false);
    FrequencyInverterHelper.SetBrakeOn((byte[])model.IC_DriverY.Value, false);
    FrequencyInverterHelper.SetReady((byte[])model.IC_DriverY.Value, true);
    FrequencyInverterHelper.SetEngineSelected0((byte[])model.IC_DriverY.Value, true);
    FrequencyInverterHelper.SetEngineSelected1((byte[])model.IC_DriverY.Value, false);


    // Initialize frecuency inverter_X values:
    FrequencyInverterHelper.SetCtrlInhibit((byte[])model.IC_DriverX.Value, false);
    FrequencyInverterHelper.SetFailure((byte[])model.IC_DriverX.Value, false);
    FrequencyInverterHelper.SetBrakeOn((byte[])model.IC_DriverX.Value, false);
    FrequencyInverterHelper.SetReady((byte[])model.IC_DriverX.Value, true);
    FrequencyInverterHelper.SetEngineSelected0((byte[])model.IC_DriverX.Value, true);
    FrequencyInverterHelper.SetEngineSelected1((byte[])model.IC_DriverX.Value, false);

    //deletePallet

Fin
```

5.3.2.- DEFINICIÓN DE PARÁMETROS SEÑAL DE ENTRADA AL MODELO DE SIMULACIÓN

En la tabla de entradas se recgen los diferentes tipos de sensores definios para el modelo del transelevador. Pueden ser sensores de 2 tipo de telemetría o de presencia.

En la siguiente imagen se muestran los sensores definidos para el transelevdor:



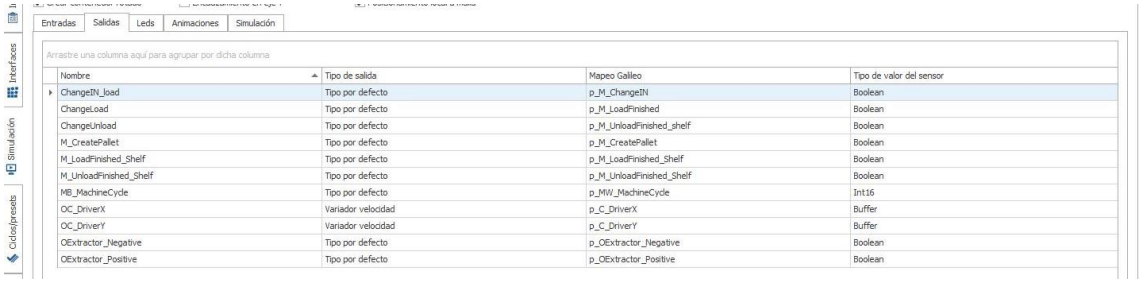
Los sensores de telemetría recogen un valor numérico asignado a una variable que representa la magnitud física medible. De este tipo

Los sensores de presencia están asociados a la detección (o no) de determinada propiedad.

En el “Manual de usuario de DesignerIV” se detalla cómo crear los sensores y cada una de sus características asociadas.

5.3.3.- DEFINICIÓN DE PARÁMETROS SEÑAL DE SALIDA DEL MODELO DE SIMULACIÓN

En la pestaña de salidas se recogen los diferentes actuadores definidos para el modelo transelevador:



Nombre	Tipo de salida	Mapeo Galileo	Tipo de valor del sensor
ChangeIN_load	Tipo por defecto	p_M_ChangeIN	Boolean
ChangeLoad	Tipo por defecto	p_M_LoadFinished	Boolean
ChangeUnload	Tipo por defecto	p_M_UnloadFinished_shelf	Boolean
M_CreatePallet	Tipo por defecto	p_M_CreatePallet	Boolean
M_LoadFinished_Shelf	Tipo por defecto	p_M_LoadFinished_Shelf	Boolean
M_UnloadFinished_Shelf	Tipo por defecto	p_M_UnloadFinished_Shelf	Boolean
MB_MachineCycle	Tipo por defecto	p_MW_MachineCycle	Int16
OC_DriverX	Variador velocidad	p_C_DriverX	Buffer
OC_DriverY	Variador velocidad	p_C_DriverY	Buffer
OExtractor_Negative	Tipo por defecto	p_OExtractor_Negative	Boolean
OExtractor_Positive	Tipo por defecto	p_OExtractor_Positive	Boolean

5.3.4.- DEFINICIÓN DE PARÁMETROS PARA LAS ANIMACIONES DEL MODELO DE SIMULACIÓN

Las animaciones permiten en la visualización del modelo ver la evolución en el tiempo de la situación del almacén, es decir, no se trata de una imagen fija de un momento sino que los elementos van cambiando su estado en función de determinadas condiciones. Dichos cambios son concatenaciones de diferentes animaciones definidas individualmente.

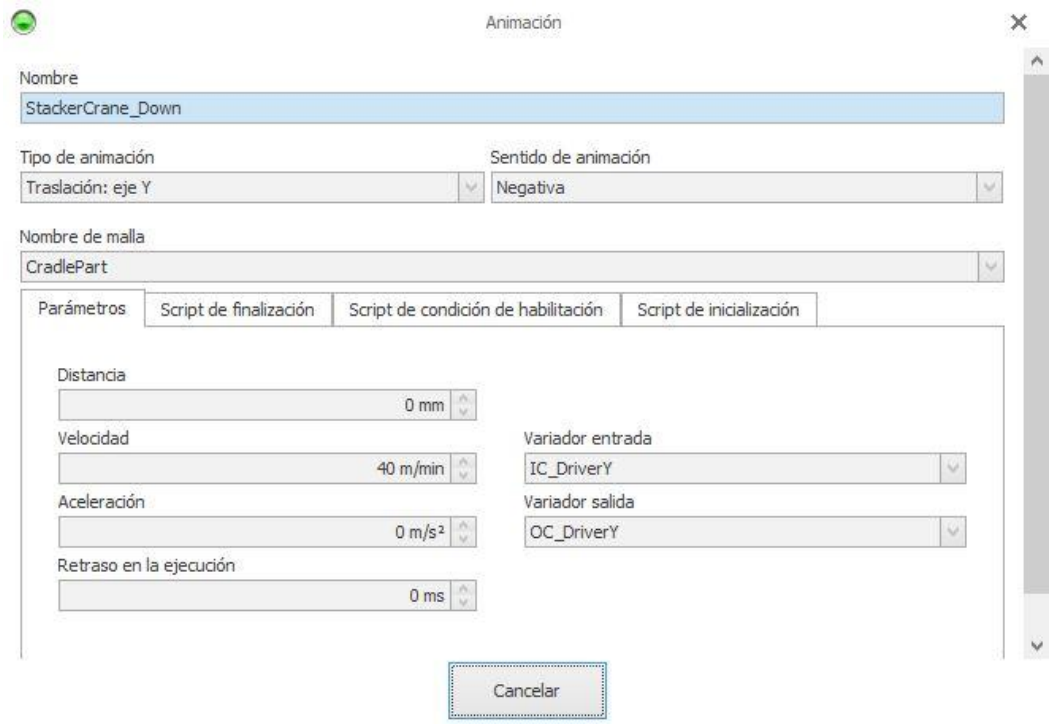
Secuencia	Nombre	Tipo de animación	Script de finalización	Script de inicialización	Script de condición de habilita...	Nombre de malla
1	StackerCrane_Down	Traslación: eje Y	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	CradlePart
2	StackerCrane_Up	Traslación: eje Y	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	CradlePart
3	StackerCrane_Backward	Traslación: eje X	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	StackerCrane
4	StackerCrane_Forward	Traslación: eje X	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	StackerCrane
7	ChangeLoad	Tiempo	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	ExtractorPart
9	OExtractor_Positive	Traslación: eje Z	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	ExtractorPart
10	OExtractor_Negative	Traslación: eje Z	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	ExtractorPart
11	DeletePallet	Eliminación de contenedores	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	Extractor_MovePallets
12	DUMMY	Tiempo	//TODO: Fill script	//TODO: Fill script	return false;	Extractor_MovePallets
13	ChangeUnload	Tiempo	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	ExtractorPart
14	ChangeIN	Tiempo	//TODO: Fill script	//TODO: Fill script	//TODO: Fill script	ExtractorPart

Los scripts de cada una de estas animaciones se definen a continuación.

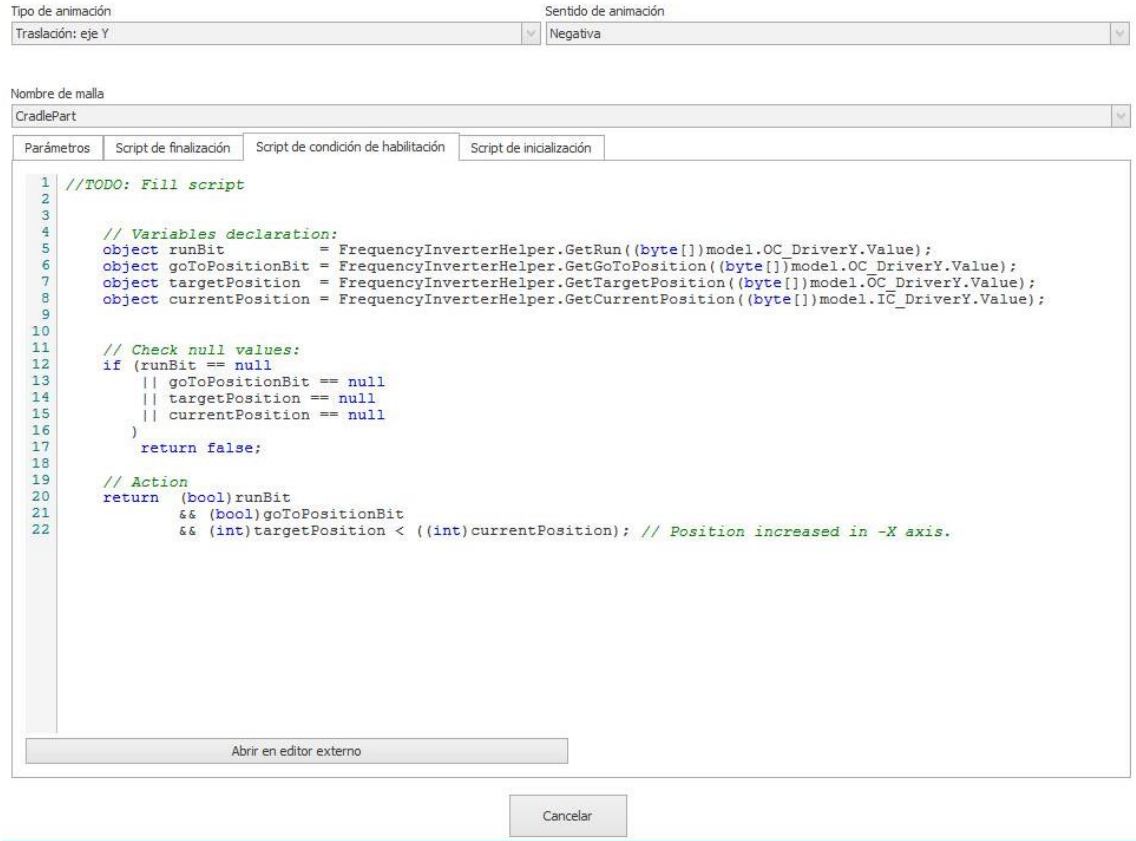
5.3.4.1.- STACKER CRANE DOWN

Animación que permite visualizar el movimiento vertical descendente de la cuna del transelevador.

Viene definida por sus parámetros:



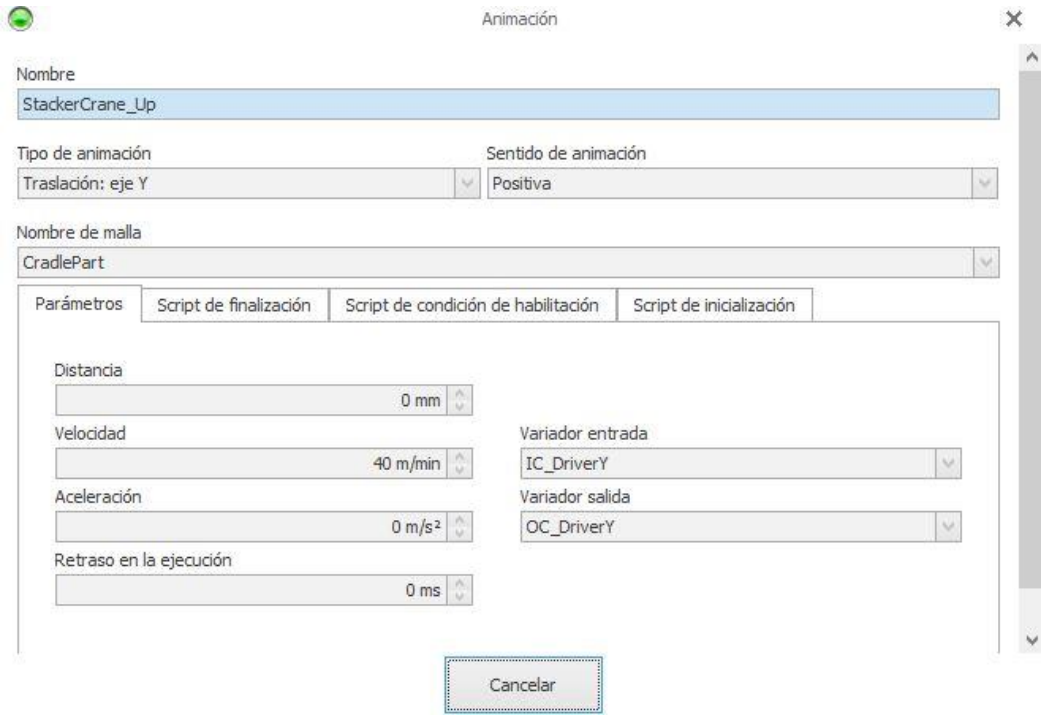
Y su script de condición de habilitación:



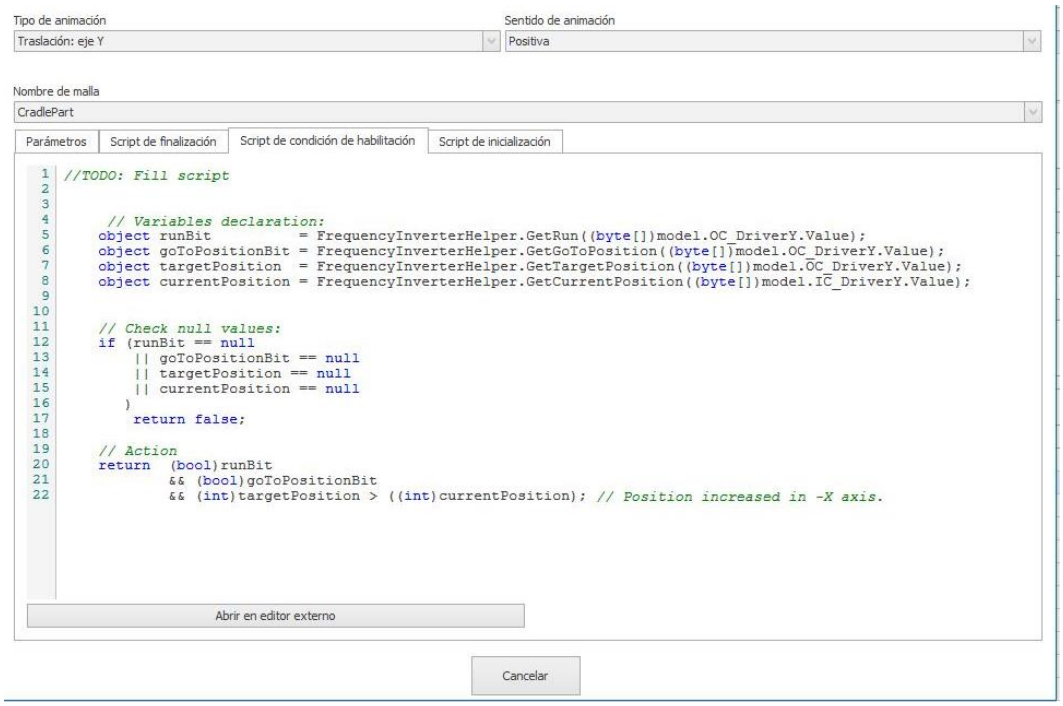
5.3.4.2.- STACKER CRANE UP

Animación que permite visualizar el movimiento vertical ascendente de la cuna del modelo transelevador.

Los parámetros configurados han sido:



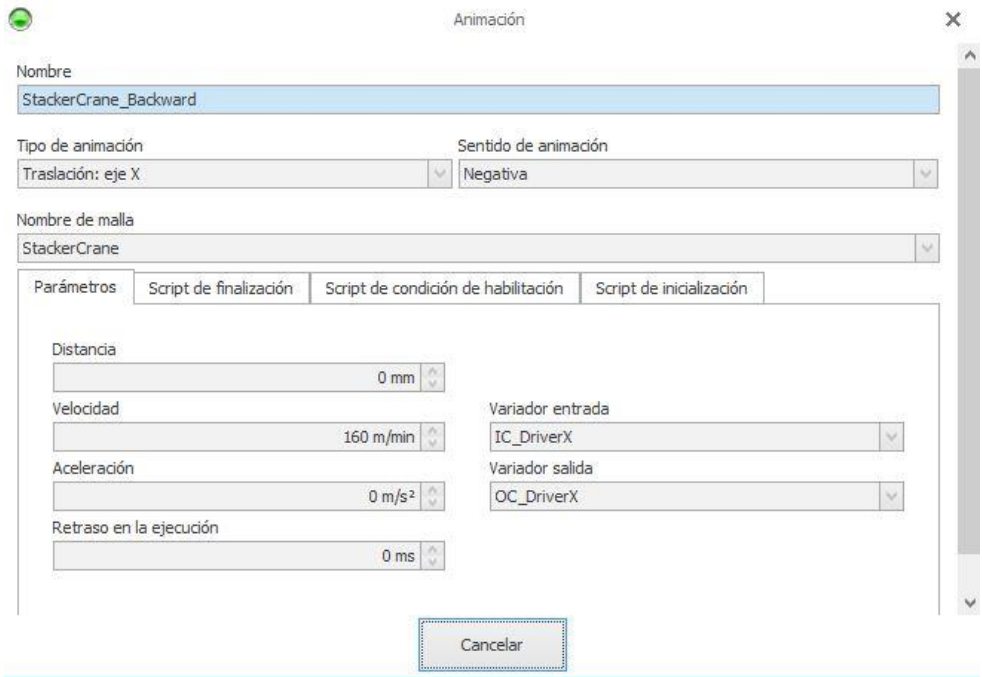
El script de condición de habilitación de la animación viene definido por:



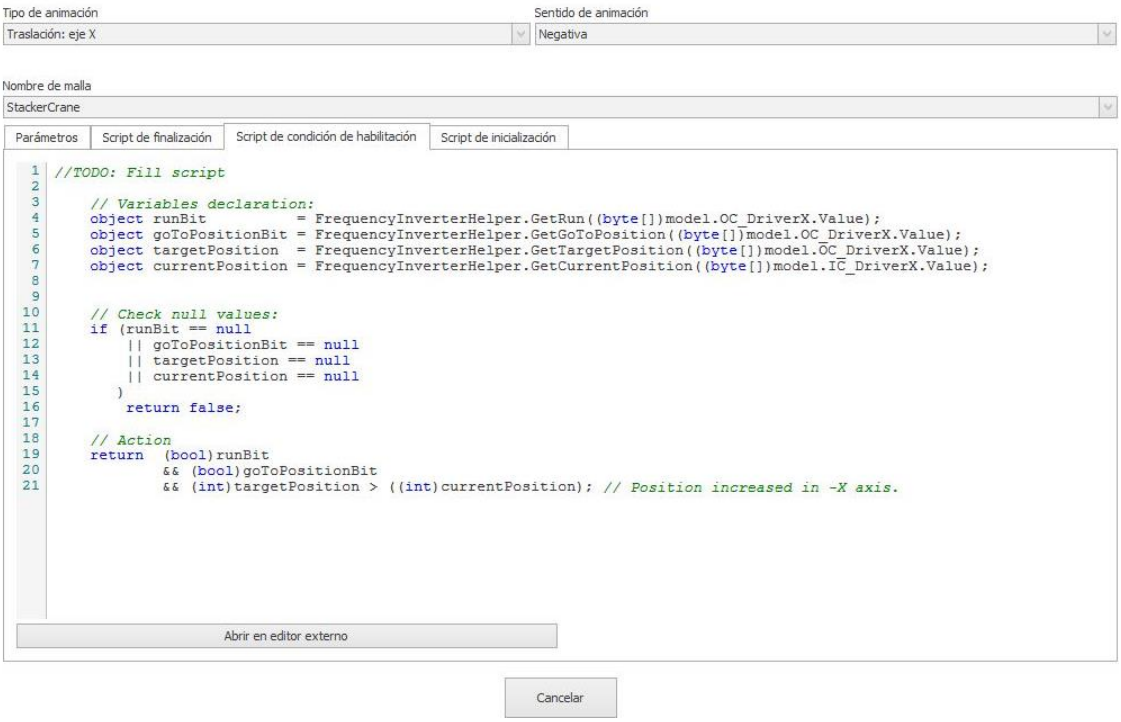
5.3.4.3.- STACKER CRANE BACKWARD

Animación que permite visualizar el movimiento horizontal en el eje X negativo del modelo transelevador.

Los parámetros configurados han sido:



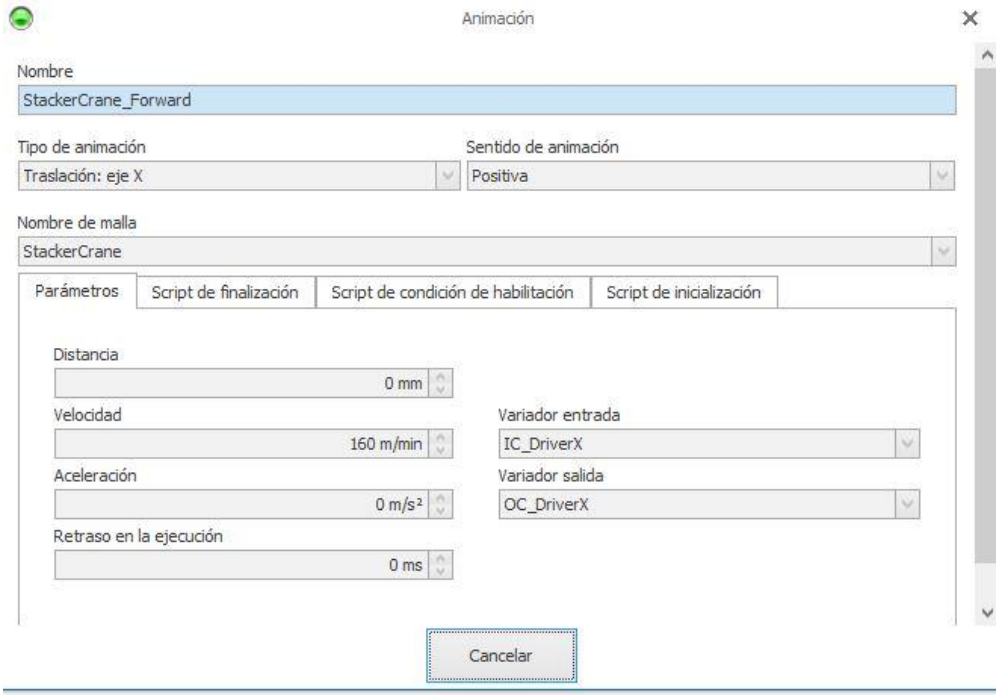
El script de condición de habilitación de la animación viene definido por:



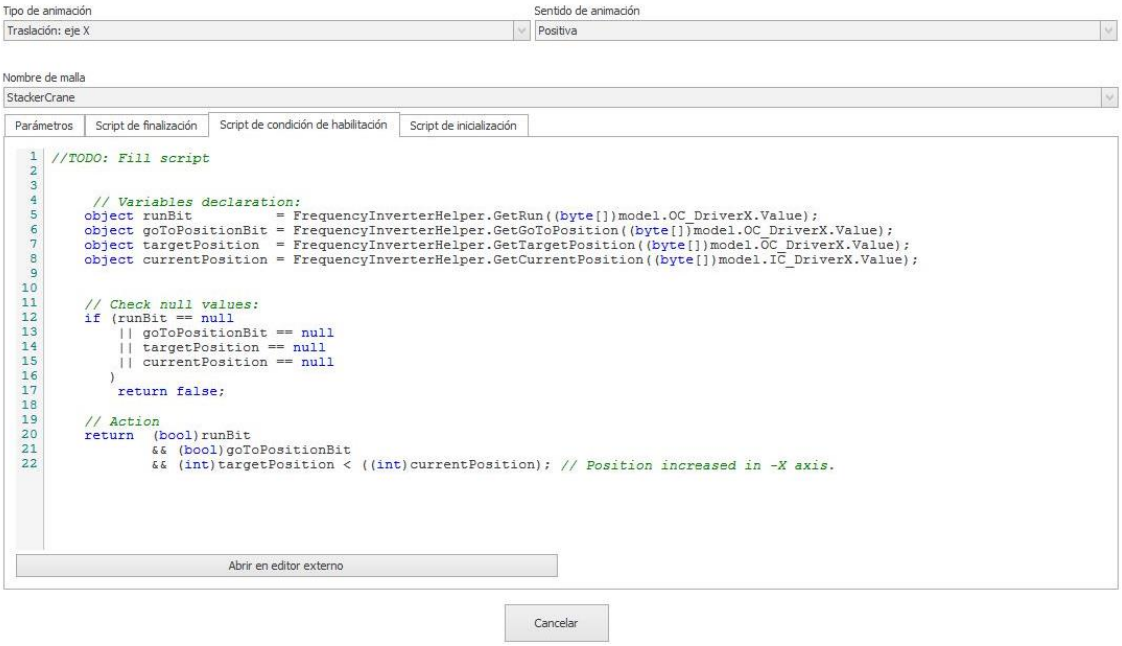
5.3.4.4.- STACKER CRANE FORWARD

Animación que permite visualizar el movimiento horizontal en el eje X positivo del modelo transelevador.

Los parámetros configurados han sido:



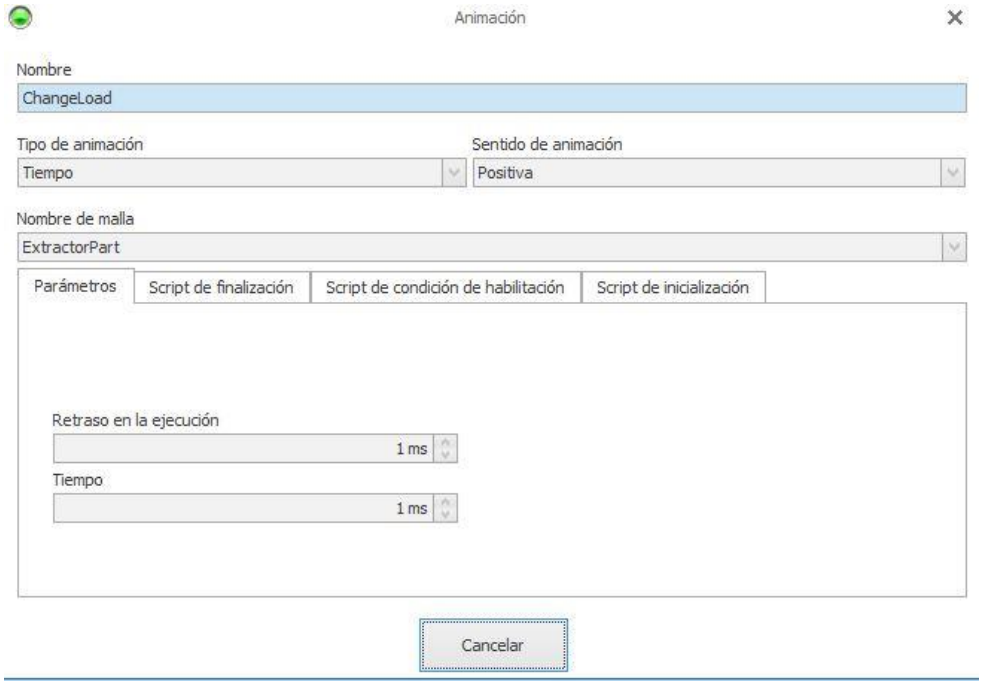
El script de condición de habilitación de la animación viene definido por:



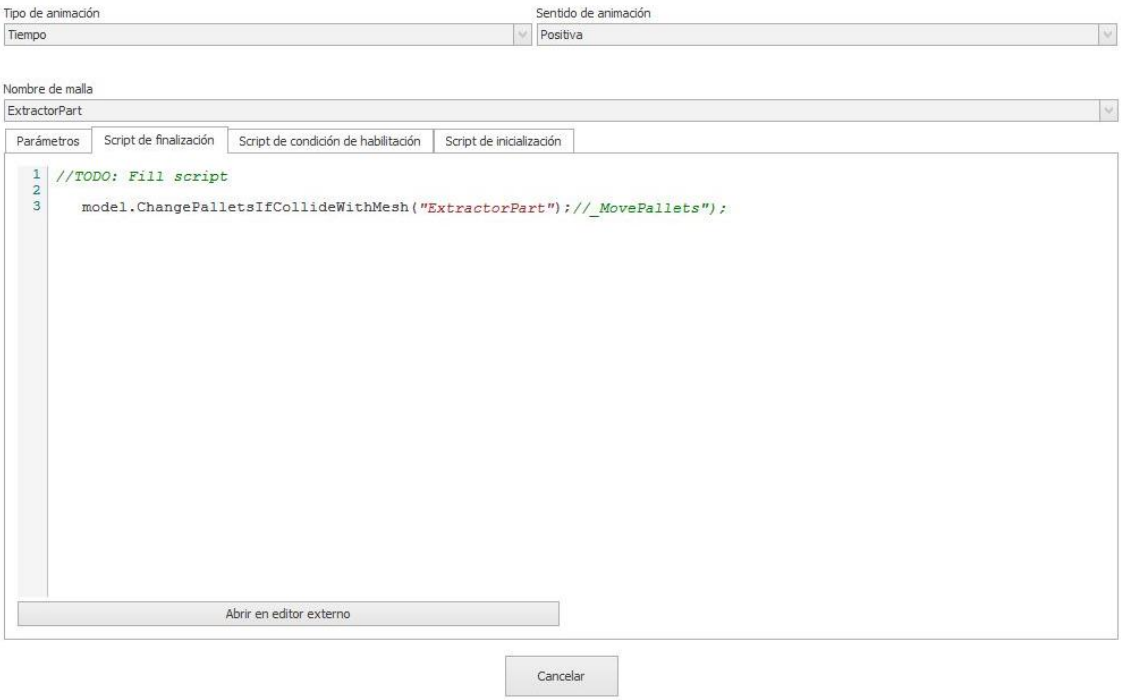
5.3.4.5.- CHANGELOAD

Animación empleada para realizar el cambio de mallas entre 2 máquinas diferentes de la instalación. En este caso para realizar un cambio de malla desde una máquina o una ubicación de estantería hacia los extractores del transelevador.

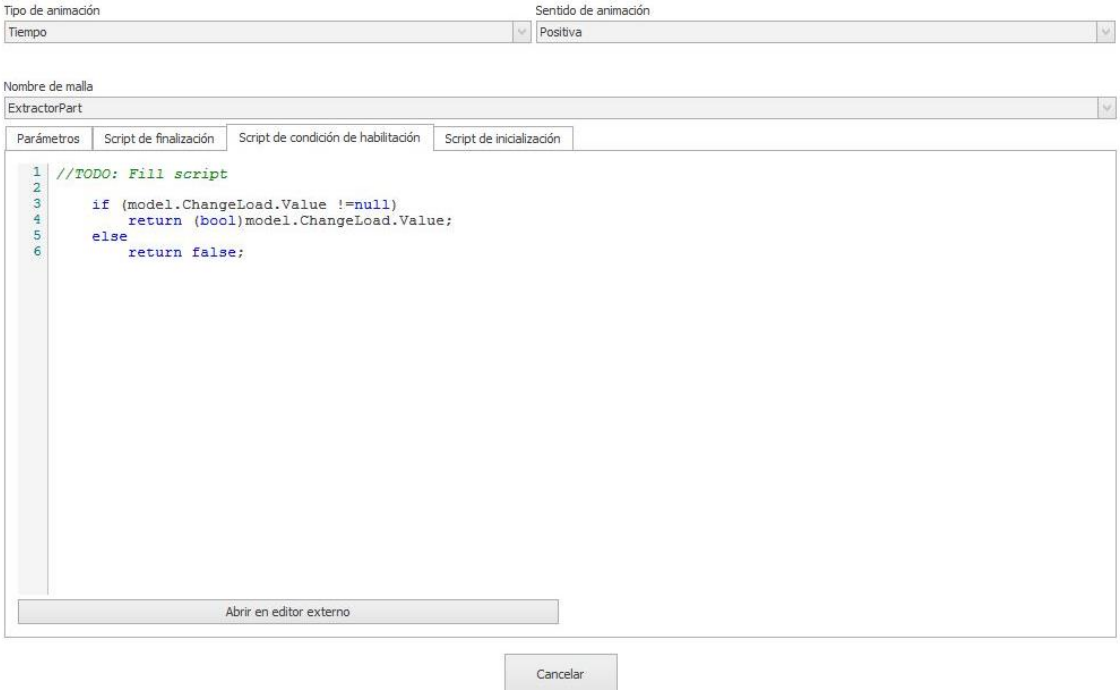
Los parámetros configurados han sido:



El script de finalización, ejecutado cuando finaliza la animación, es:



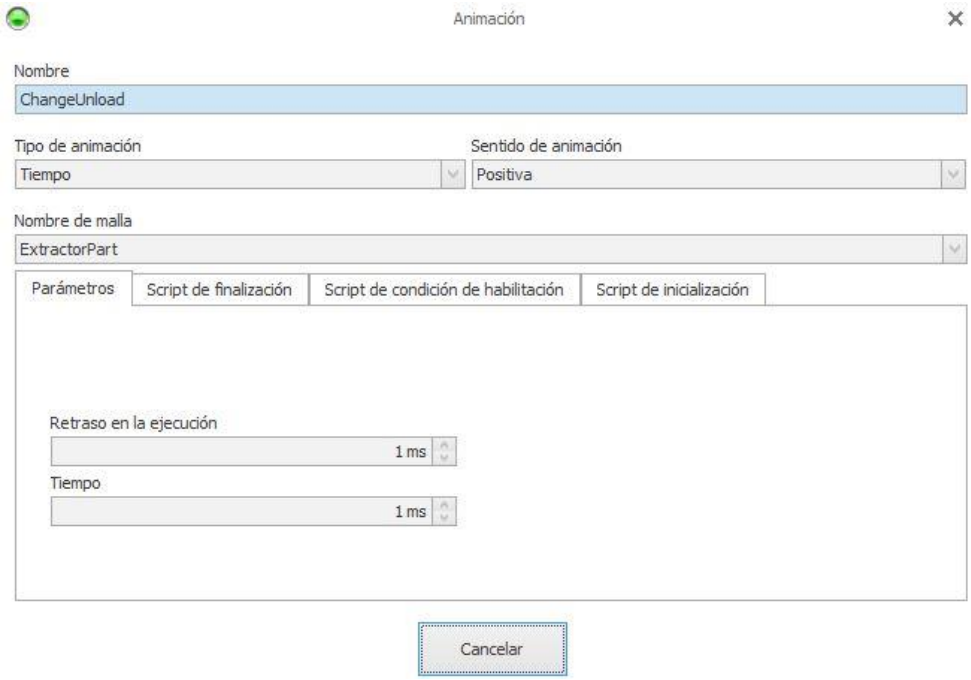
El script de condición de habilitación de la animación viene definido por:



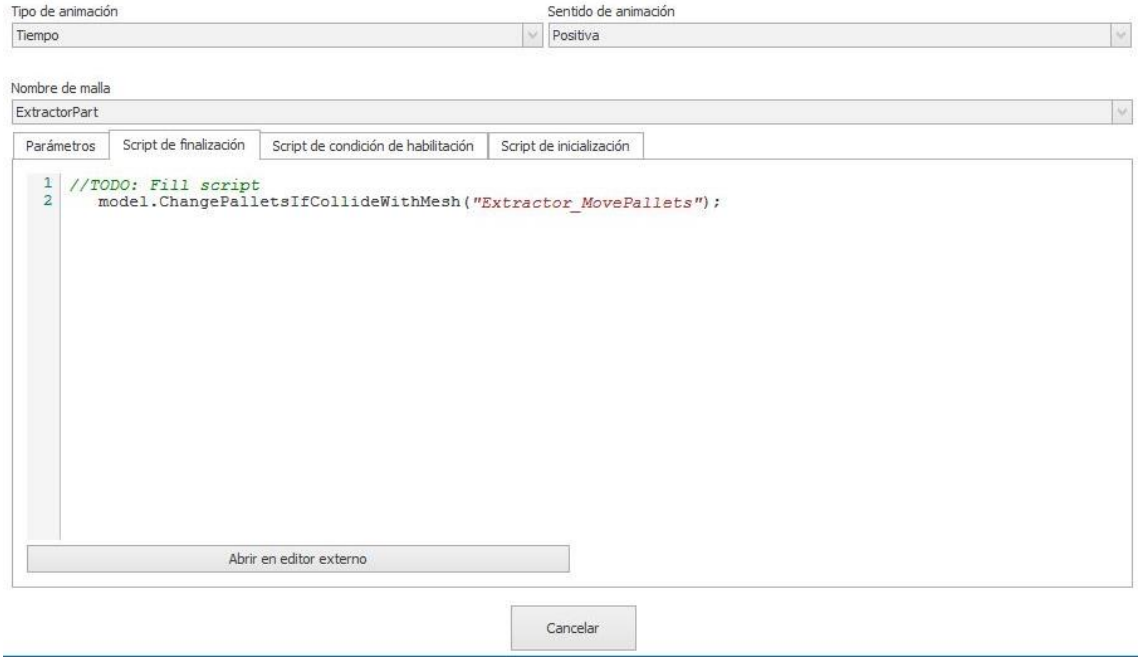
5.3.4.6.- CHANGEUNLOAD

Animación empleada para realizar el cambio de mallas entre 2 máquinas diferentes de la instalación. En este caso para realizar un cambio de malla desde los extractores del transelevador hacia una máquina o una ubicación de la estantería.

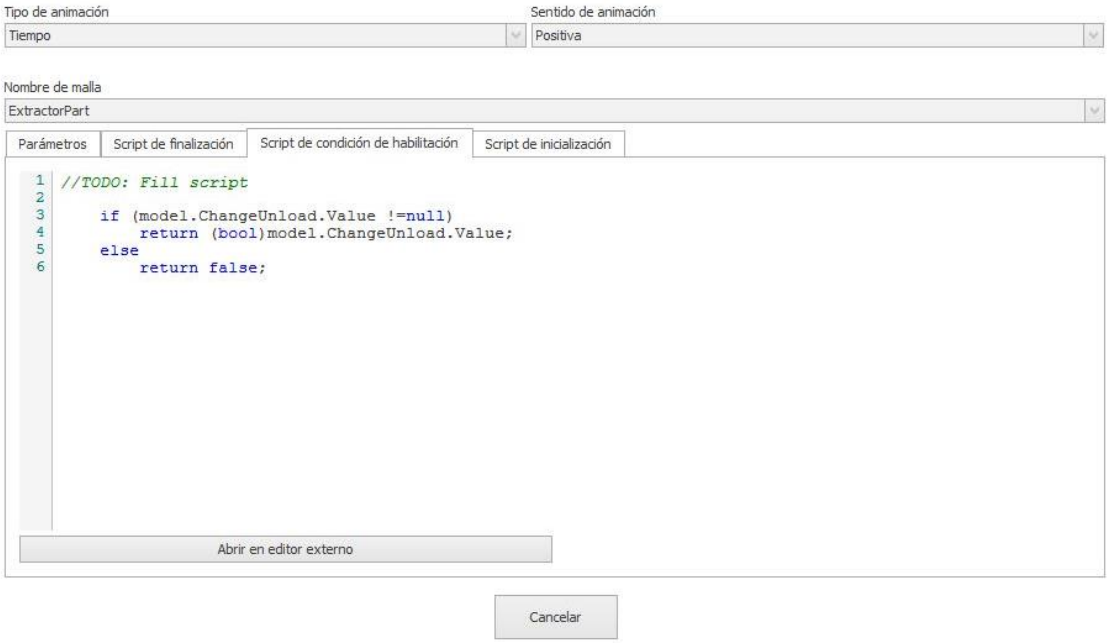
Los parámetros configurados han sido:



El script de finalización, ejecutado cuando finaliza la animación, es:

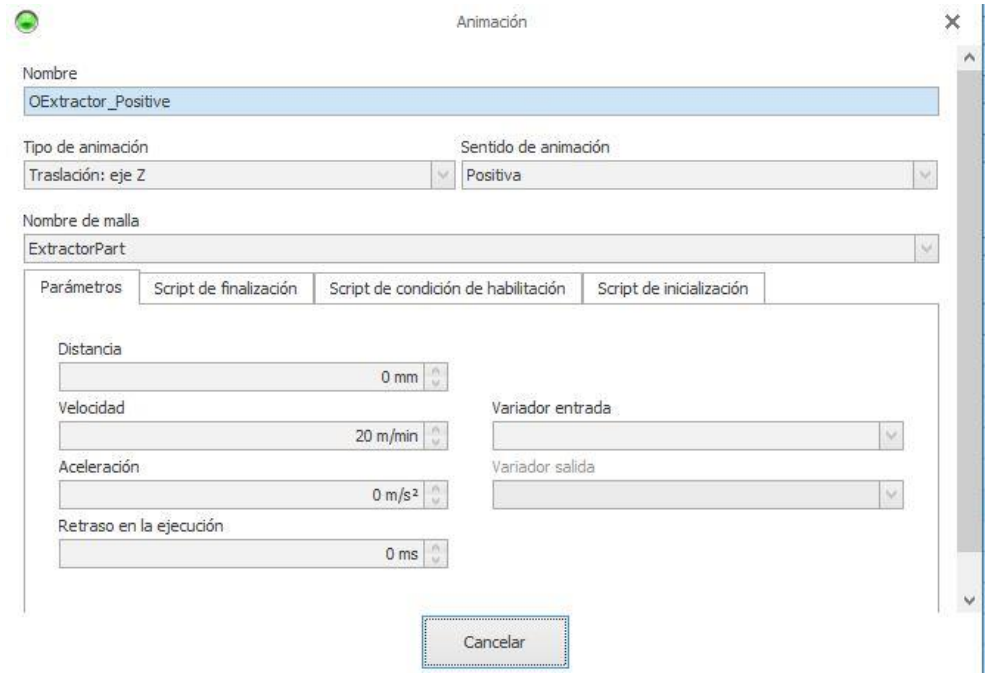


El script de condición de habilitación de la animación viene definido por:

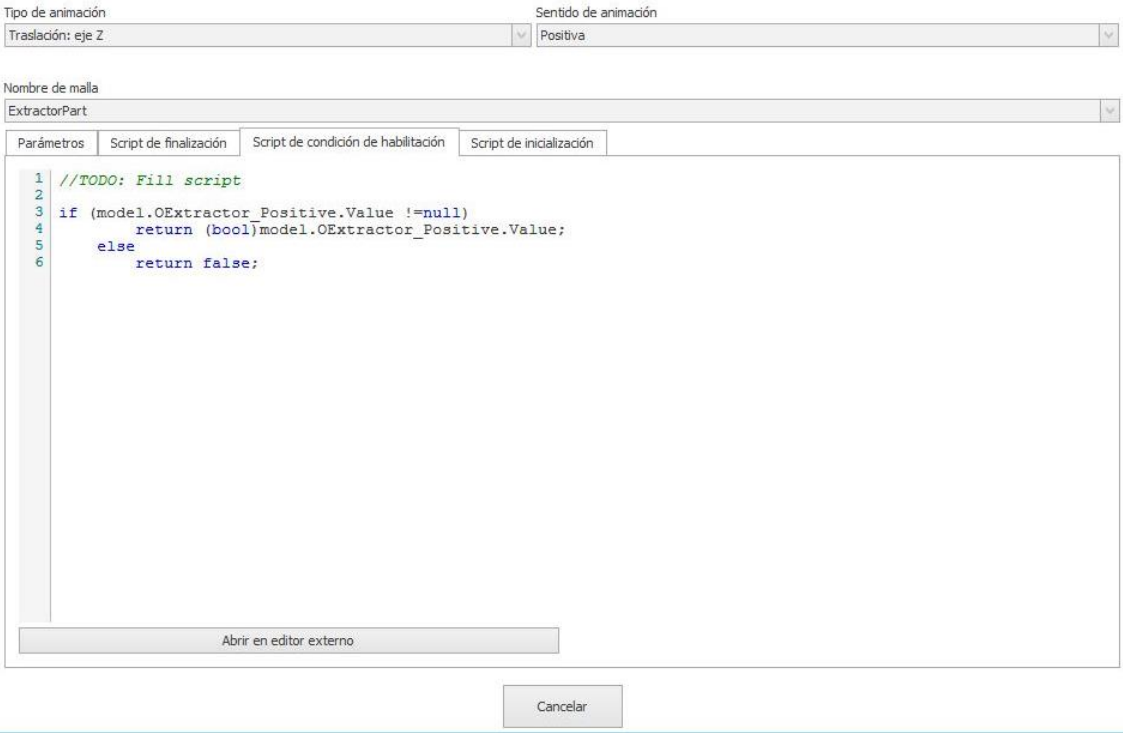


5.3.4.7.- OEXTRACTOR POSITIVE

Animación empleada para visualizar el movimiento de los extractores en el eje Z positive. Los parámetros configurados han sido:



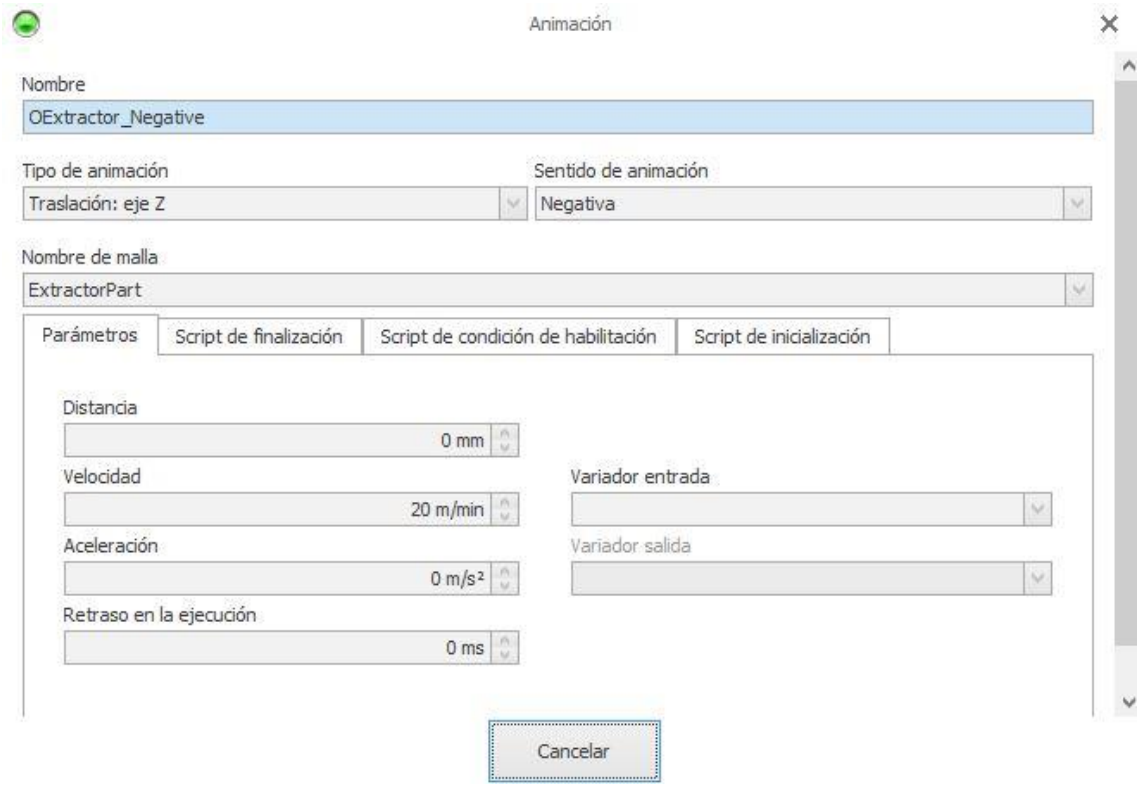
El script de condición de habilitación:



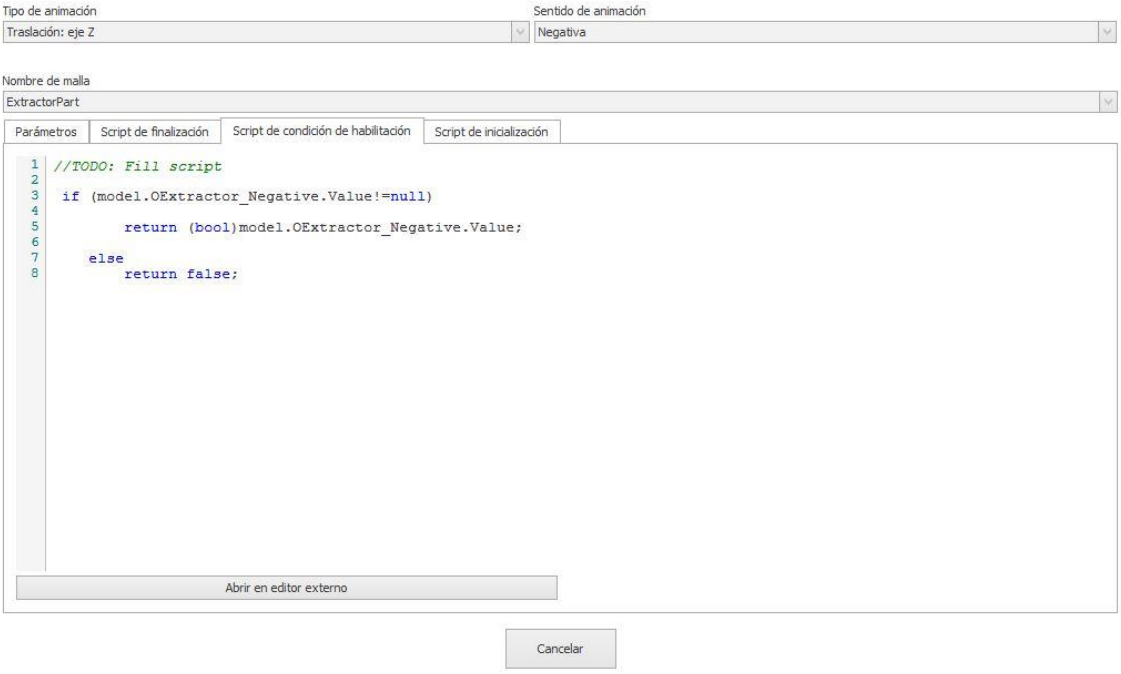
5.3.4.8.- OEXTRACTOR NEGATIVE

Animación empleada para visualizar el movimiento de los extractores en el eje Z positive.

Los parámetros configurados han sido:



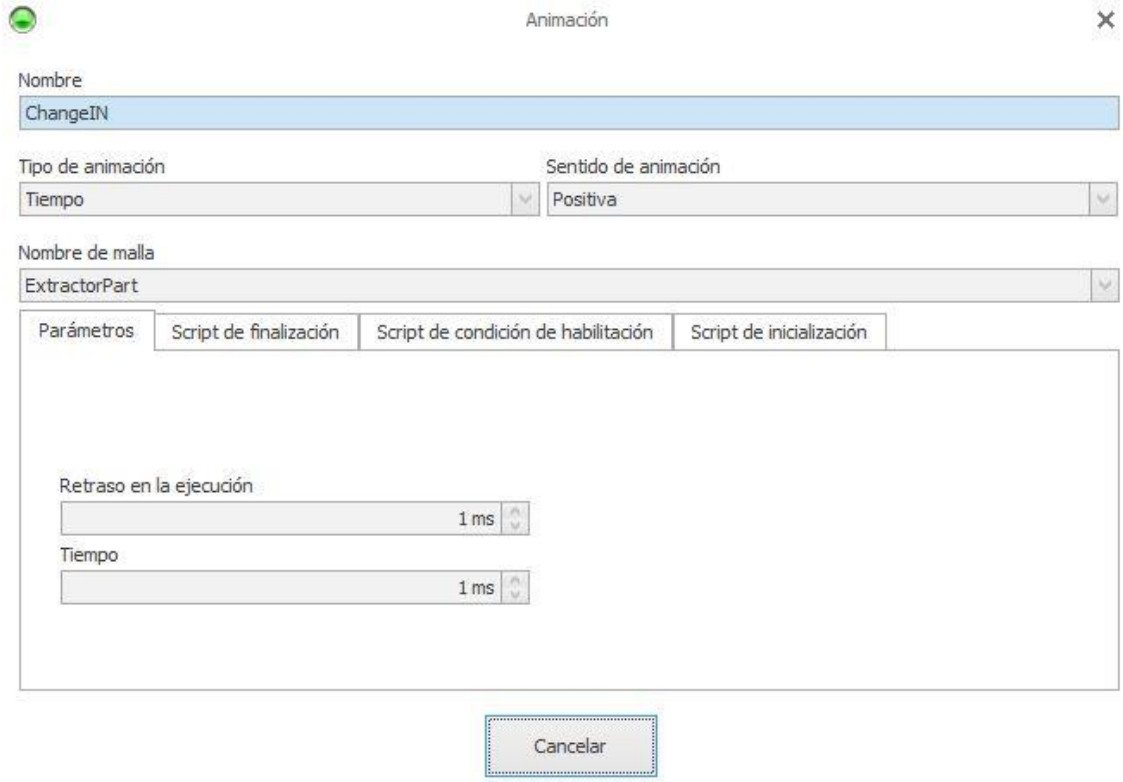
El script de habilitación viene dado por:



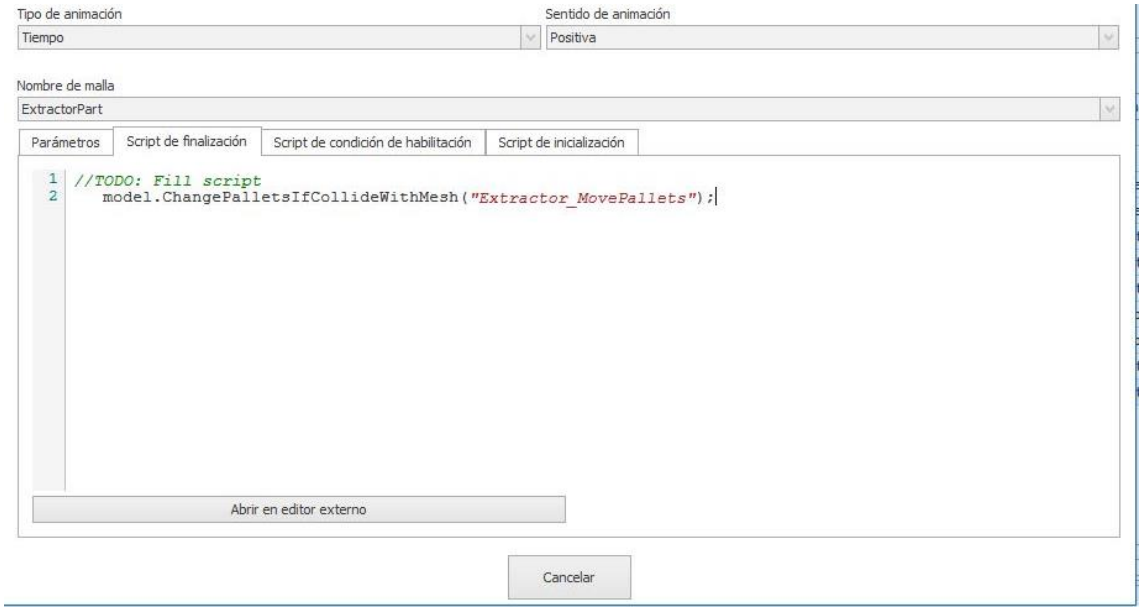
5.3.4.9.- CHANGEIN

Animación empleada para realizar el cambio de la malla extractores hacia la malla cuna.

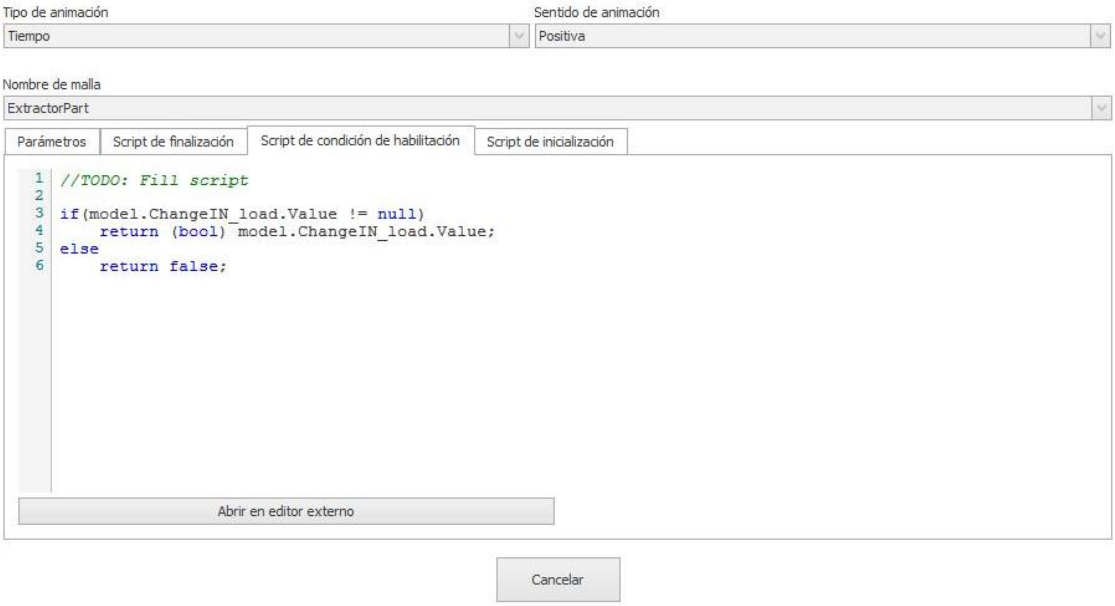
Los parámetros para esta animación son:



El script de finalización se define como:



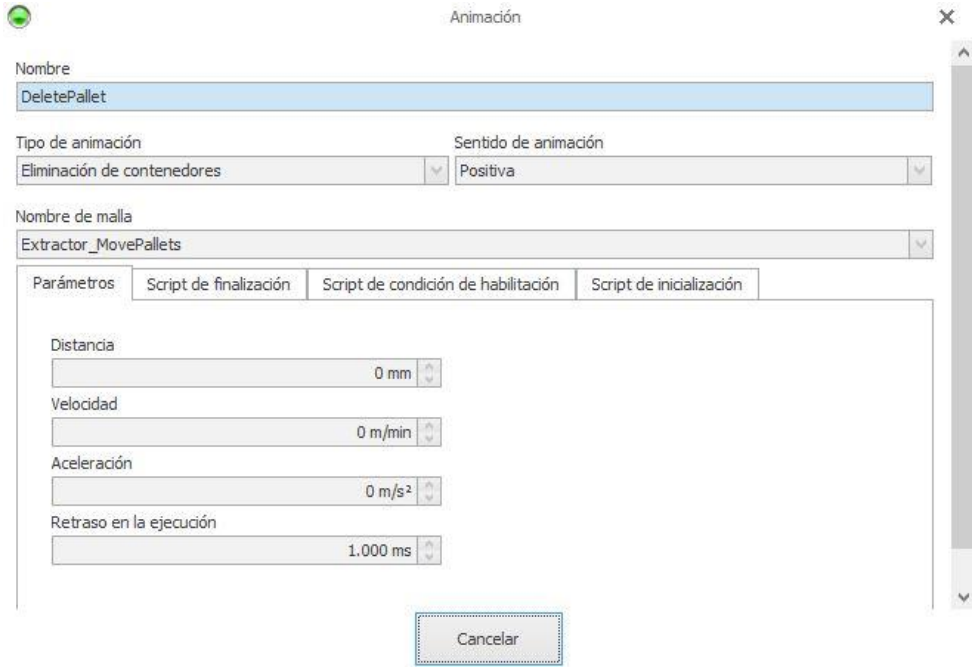
El script de habilitación de la animación es:



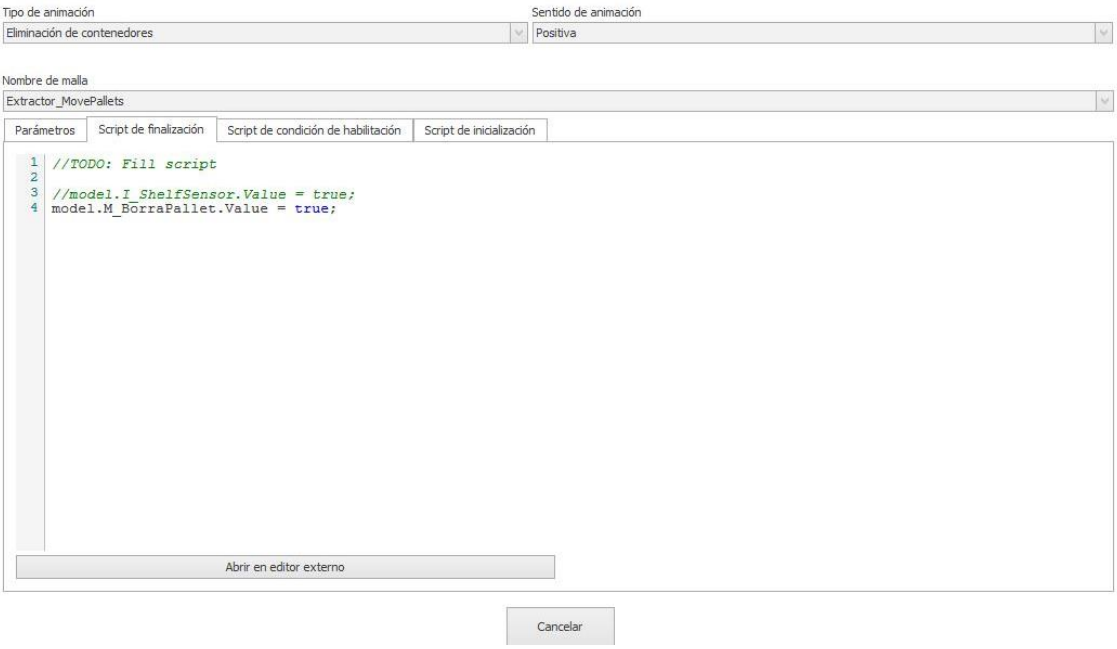
5.3.4.10.- DELETE PALLET

Animación empleada cuando el transelevador deposita un contenedor en la estanteria para eliminar dicho contenedor.

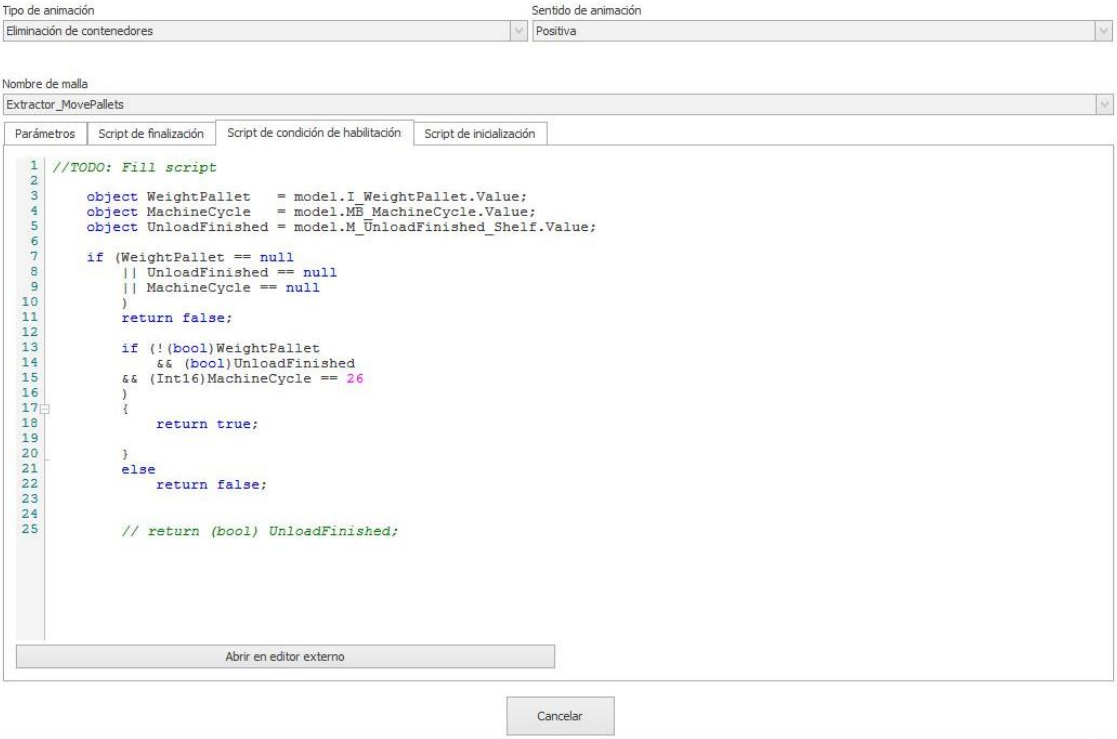
Los parámetros de esta animación son:



El script de finalización:



El script de habilitación de la animación es:



6. DISEÑO DE ENTORNO DE PRUEBAS

6.1. - DISEÑO DEL LAYOUT - EASYS

El layout de un almacén es una zona grafica del configurador de EasyS. Representa una vista en planta del espacio real de almacenamiento. En ella se van añadiendo los distintos elementos que configuran la apariencia final del almacén, con pasillos ubicaciones, muelles, estaciones de trabajo…

Se ha diseñado un entorno de pruebas, basado en una instalación sencilla con una cabecera formada por:

- Un área de recepciones:
    - Un puesto de entrada a la instalación (INPUT)
    - Una estación de identificación de entradas o PIE
    - Una serie de transportadores convencionales (Transportador)
    - Una mesa de entrada a almacén (ME)
- Un área de expediciones:
    - Una mesa de salida de almacén (MS)
    - Una serie de transportadores convencionales (Transportador)
    - Un puesto de salidas (PS)
- Un área de almacenamiento o almacén, donde habrá:
    - Un pasillo automático donde operará el modelo de transelevador creado (AISLE)
    - 2 estanterías de simple profundidad con capacidad para 50 contenedores cada una. (SHELF_1/2)

Este diseño de layout se ha implementado mediante la herramienta EasyS, donde se han configurado todos los elementos, máquinas, rutas, con los valores y las relaciones que deben tener entre ellas. La imagen siguiente muestra, el diseño:

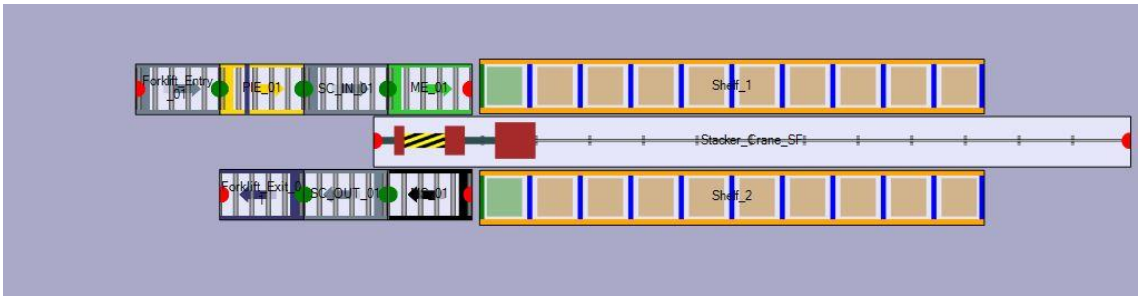


Figura 6.1.- *Layout del diseño/entorno de pruebas del transelevador.*

Identificando cada una de las máquinas citadas anteriormente:

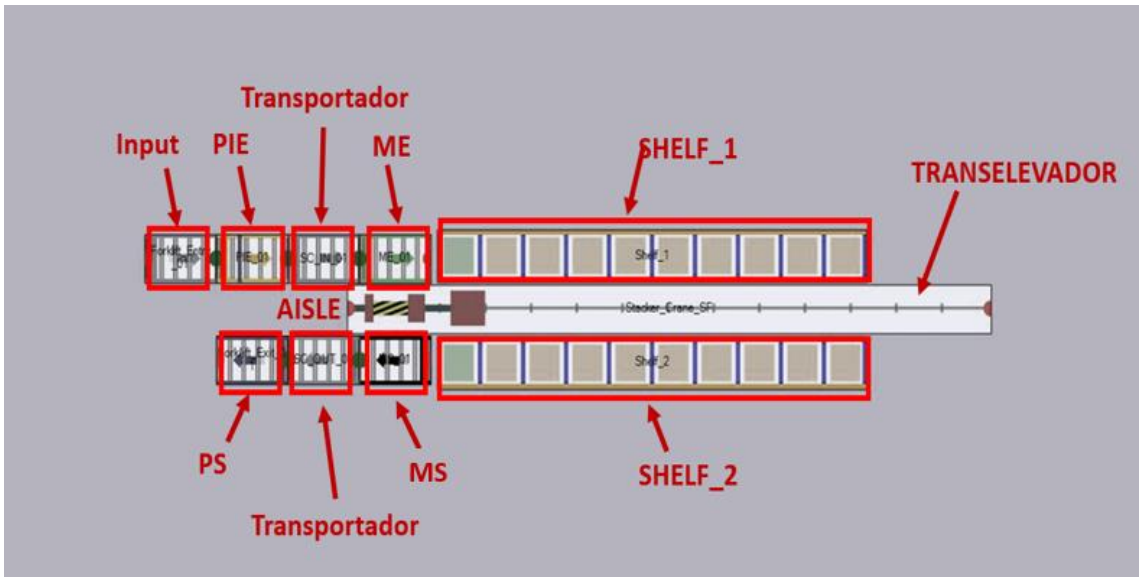


Figura 6.2.- *Identificación de las máquinas del diseño de pruebas.*

Las máquinas del layout anterior, tienen las siguientes finalidades:

- Input o estación de entrada (IN o ES): Estación que representa el punto de entrada al almacén automático, donde los contenedores son depositados y desde la cual son movidos al PIE
- Puesto de identificación de entradas (PIE): Estación donde se realiza el control dimensional e identificativo del contenedor en una instalación automática. El control dimensional es realizado por el sistema de control. Con la información enviada por el sistema de control, EasyWMS® envía el contenedor al puesto de rechazo o al almacén.
- Mesa de entrada (ME): Esta estación el punto de enlace con el Transelevador. Su función es permitir una última optimización antes de entregar el contenedor a la máquina. Esta estación es necesaria en todo caso y permite realizar cambios en función de las características del Transelevador (cantidad de extractores y configuración de los mismos) y de los transportadores de entrega al Transelevador. Se encarga de la optimización de ubicación en función de las características de la máquina.
- Estantería (SHELF): Estación que representa las ubicaciones de un pasillo de un almacén
- Transelevador (STC): Máquina automática encargada de los procedimientos de ubicación y extracción de contenedores en la estantería mediante un sistema de movimiento en tres ejes (traslación, elevación y profundidad). Se emplea en pasillos automáticos no transitables.
- Mesa de salida (MS): Estación de descarga del transelevador.

- Puesto de salida (PS): Estación en la que un contenedor es extraído del almacén automático hacia el exterior mediante una carretilla, transpaleta, etc.

Todas las máquinas citadas se encuentran descritas con mayor detalle en el "ANEXO I: Logística"

6.2. - INTERFACE EASYWCS

EasyWCS se encarga de gestionar los movimientos a través de una base de datos (BBDD) ofreciendo una interfaz sencilla e intuitiva, facilitando la configuración y diagnosis de las tareas en las estaciones de una instalación. Es el encargado de la interconexión entre Galileo y el SGA.

En solo dos pasos se puede comenzar a trabajar con esta herramienta:

- Paso 1: Instalación del diseñador y servicio de EasyWCS. Con ello se tienen todas las herramientas para el arranque y la realización de pruebas.

Figura 1.3.- *Instalador de la herramienta EasyWCS*

- Paso2: Importación directa desde EasyWCS del layout de estaciones y rutas de la instalación previamente definidas en EasyS.

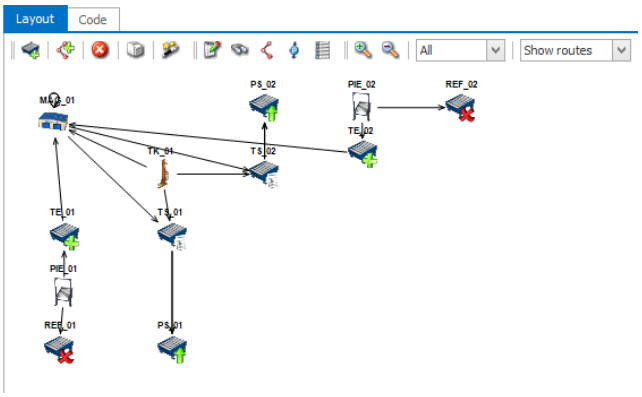


Figura 6.3.- *Ejemplo de layout y rutas configuradas en EasyWCS.*

La imagen siguiente muestra la configuración llevada a cabo para probar el modelo de simulación del transelevador.

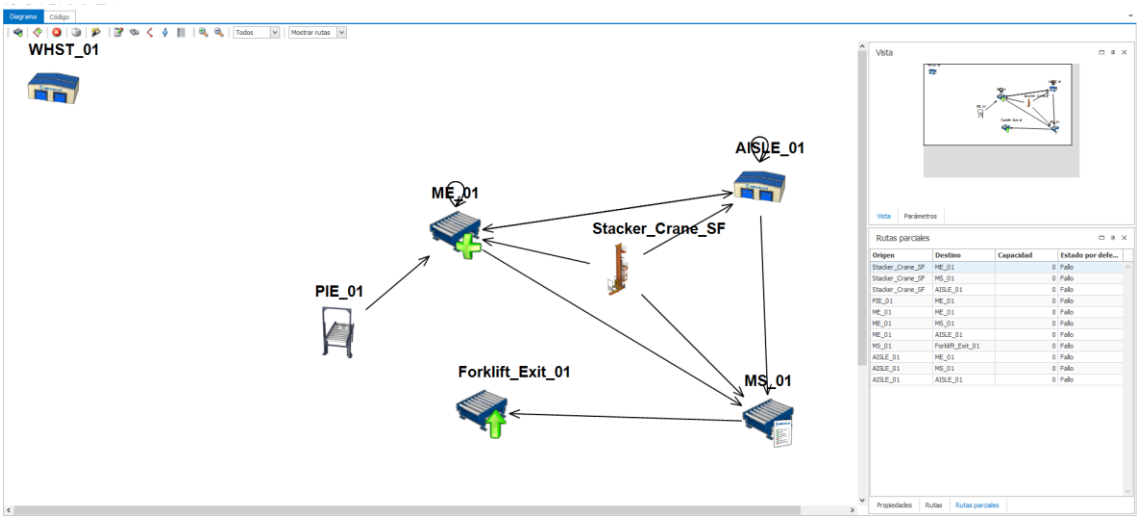


Figura 6.4.- *Configuración de las rutas y maquinas del entorno de pruebas del transelevador*

Como se pude ver en la imagen es imprescindible tener rutas de comunicación entre las estaciones correspondientes, estas rutas se explican más detalladamente en el "Manual de usuario de EasyWCS"

La interfaz gráfica de usuario de EasyWCS, cuenta con:

- Un sistema de supervisión en tiempo real de las estaciones y rutas de movimiento entre estaciones.
- Un visor de tiempo de ejecución de las diferentes operaciones.
- Unas tablas detalladas de las tareas realizadas o en proceso.

EasyWCS ayudará con la realización de las pruebas de funcionamiento del modelo del transelevador porque:

- Nos ofrece la funcionalidad de una interfaz gráfica de usuario, esto es:
    - Supervisión en tiempo real de las estaciones de la instalación emulada, de manera clara e intuitiva.
    - Facilidad de configuración de estaciones y rutas entre estaciones, destacando la rapidez en todas aquellas tareas repetitivas que implican la creación de muchas estaciones y rutas
    - Visor de tramas y tiempos de ejecución implicados en la comunicación
    - Informes y diagnosis
- Nos ofrece la robustez de un sistema de transporte basado en la experiencia de otros productos de Mecalux:
    - Gestión de tareas mediante preselecciones estándar
    - Particularización por parte del implementador de casos especiales según necesidades

El "Manual de usuario de EasyWCS", se detallan los aspectos de la configuración del entorno de pruebas.

6.3. - INTERFACE DESIGNER IV

Con la herramienta DesignerIV. Los pasos a seguir para tener la configuración correcta del diseño para la realización de pruebas se detallan en el "Manual de usuario de DesignerIV". Los pasos simplificados son:

- Paso1: Crear un secuenciador, los secuenciadores son el corazón del sistema de control Galileo, ya que con ellos se definen los comportamientos de los diferentes elementos a controlar. Un secuenciador es un autómata finito y determinista por definición. Sin embargo, dentro del sistema de control Galileo, los secuenciadores extienden esta definición para añadirle diversas propiedades que aumentan sus posibilidades hasta un punto que permiten modelar comportamientos de gran complejidad.
- Paso2: Crear el programa de control del secuenciador transelevador: consiste en generar un programa de control, con todos sus parámetros, métodos y modos de funcionamiento necesarios para el correcto funcionamiento de la máquina transelevador
- Paso3: Crear el modelo de simulación del modelo transelevador, con todas sus variables y todas sus acciones asociadas.

La imagen siguiente muestra la visualización del sistema en la que puede verse tanto el modelo de transelevador como la ventana del inspector donde se ven los valores que van tomando los diferentes parámetros durante la emulación.

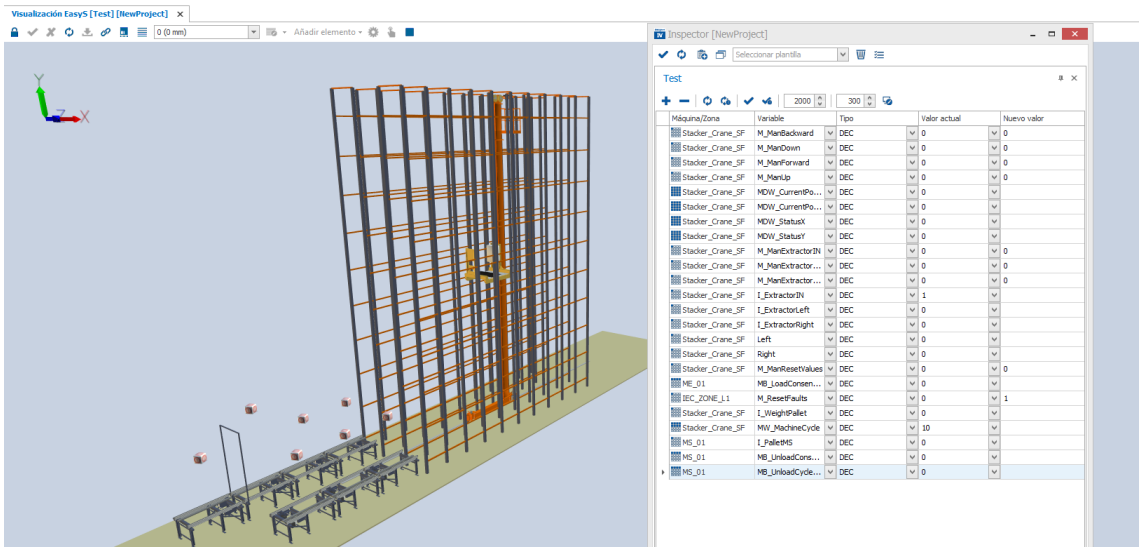


Figura 6.5.- *Visualización de la instalación y ventana "inspector" de valores de variables.*

6.4. - PRUEBAS REALIZADAS SOBRE EL SISTEMA

De cara a analizar la funcionalidad y el ajuste del sistema a las especificaciones, se diseñaron varias pruebas con diferentes objetivos y orientadas a comprobar aspectos diversos tanto del modelo como del resto de herramientas que componen el sistema.

6.4.1.- PRUEBA 1. MOVIMIENTOS DEL TRANSELEVADOR

La primera prueba tiene como objetivo estudiar el ajuste del modelo a la lógica que se le presupone, es decir, que realice correctamente la carga/descarga de contenedores tanto desde transportadores de entrada/salida como desde estantería

- En lo referente la realización de cargas de contenedores de una mesa de entrada, es decir cuando el almacén realiza una recepción de un contenedor y necesita ubicarlo, en la zona de almacenamiento o estantería.

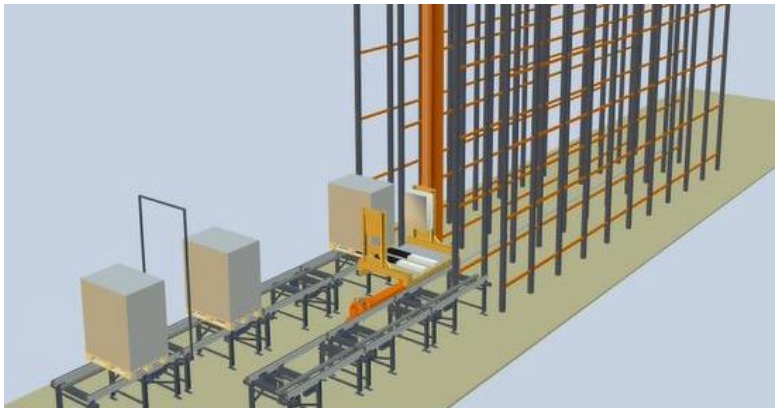


Figura 6.6.- *Recogida de contenedor de ME mediante transelevador*

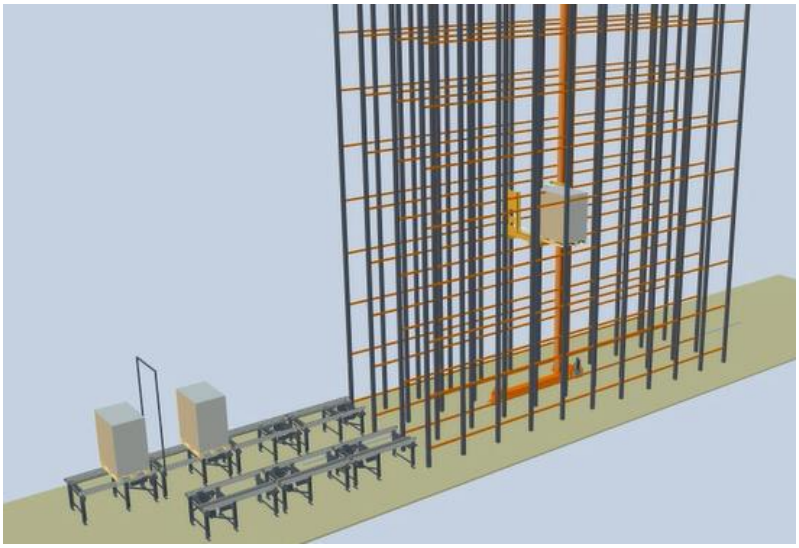


Figura 6.7.- *Descarga de contenedor en estantería*

- Verificación del correcto funcionamiento de la instalación cuando el sistema de SGA, solicita realizar una tarea de extracción de un contenedor de una ubicación de la estantería, para dejarlo sobre el puesto de salida.

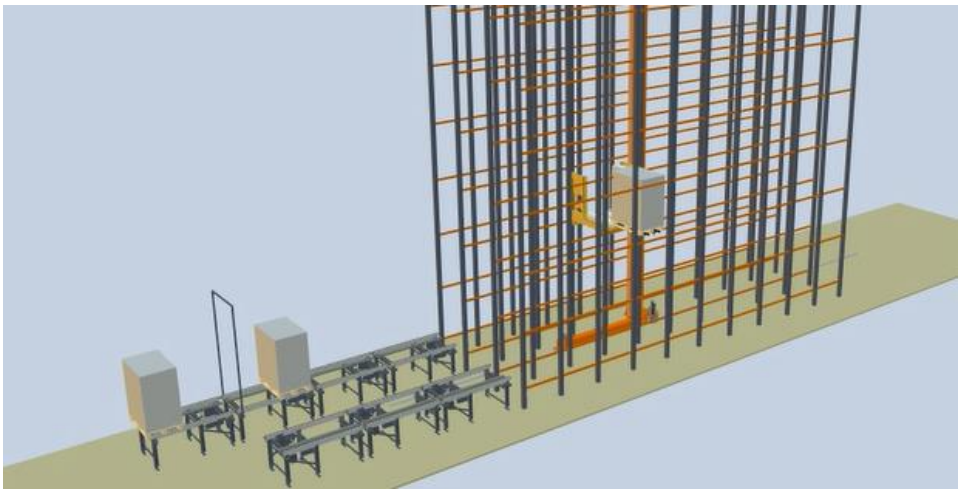


Figura 6.8.- *Carga de contenedor de estantería*

Figura 6.9.- *Descarga del transelevador en MS (mesa de salida)*

- Representación del caso, en que se solicita un contenedor para completar un pedido y este acaba de ser recibido, es decir, el contenedor entrará por la mesa de entrada hacia el puesto de identificación de entradas, posteriormente avanzará hacia la mesa de entrada al almacén, y el transelevador realizará un handshake, para recoger el pallet y posteriormente, efectuará otro handshake con la mesa de salida del almacén para depositar el contenedor.

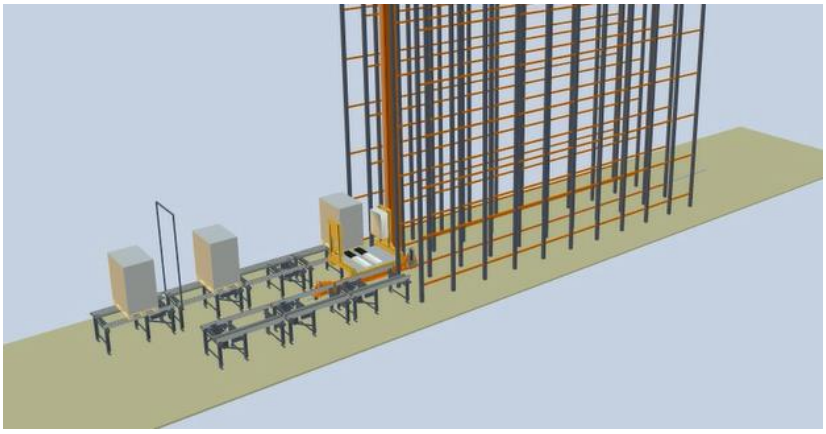


Figura 6.10.- *Carga de contenedor de ME*

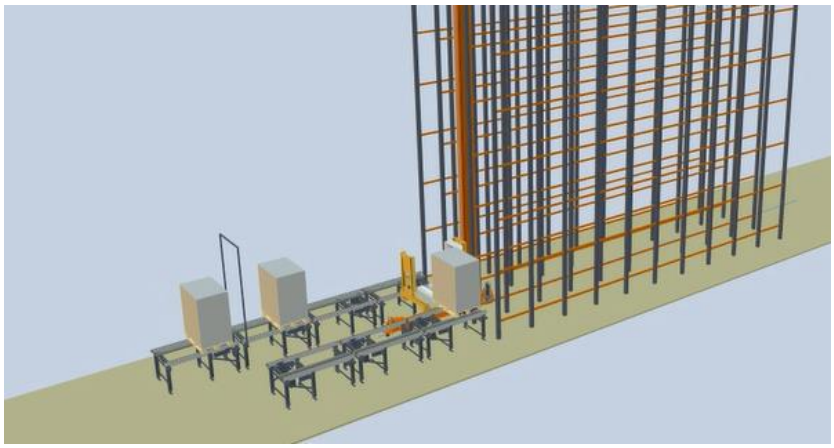


Figura 6.11.- *Descarga de contenedor en MS*

- Pruebas de diferentes modos de funcionamiento
    - Manual
    - Automático
    - Defecto (error de depósito/extracción)

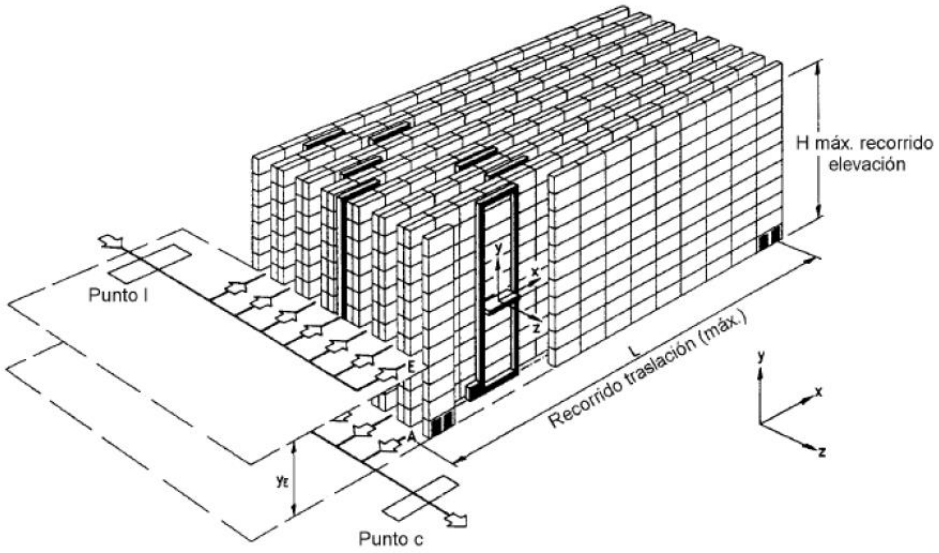
6.4.2.- PRUEBA 2. TIEMPOS Y CICLOS DE TRABAJO

La segunda prueba está determinada por el objetivo de validar la capacidad del sistema a operar en los tiempos reales para realizar los movimientos de un ciclo de carga/descarga de un transelevador.

La concreción real del tiempo de ciclo de trabajo es muy compleja y costosa, en consecuencia, a fin de poder estimar un valor medio estadístico, que sea exportable y que constituya una buena aproximación del valor exacto determinado en la práctica, la norma define una serie de pre-supuestos de base que determinan una seria de pautas a seguir.

A continuación, se indican los supuestos teóricos en los que se basa la norma, y los que son aplicables al modelo de transelevador creado para este proyecto.

- Se distinguen 2 puntos claramente diferenciados en el almacén:
  - Punto E: punto de ubicación de recogida para las mercancías que van a ser almacenadas
  - Punto A: punto de traslado para las mercancías que van a ser desalmacenadas.
  - H: recorrido máximo de elevación en el almacén
  - L: recorrido máximo de traslación en el almacén.



Punto I: Punto de identificación en entradas
Punto C: Punto de control en salidas

Figura 6.12.- *Puntos de interés del almacén*

- Determinación de los puntos teóricos (P1 y P2) de referencia o ensayo, que permitan establecer el valor del tiempo medio de un ciclo de trabajo, que aproxime el valor real exacto. Dichos puntos dependen de la configuración del almacén, en especial, dependen de las siguientes magnitudes:
  - Altura y longitud del almacén. La ubicación de estas longitudes determina el recorrido en elevación y traslación de los transelevadores en el almacén.
  - Posición relativa de los puntos de entrada y salida de mercancías en el almacén. Los puntos teóricos de referencia (P1 y P2) se calcula en función de la posición relativa de los puntos de entrada

6.4.2.1.- DEFINICIÓN DE CICLO SIMPLE

En la definición de un ciclo simple de trabajo, se definen 2 operaciones:

1. **Reposición** o almacenaje

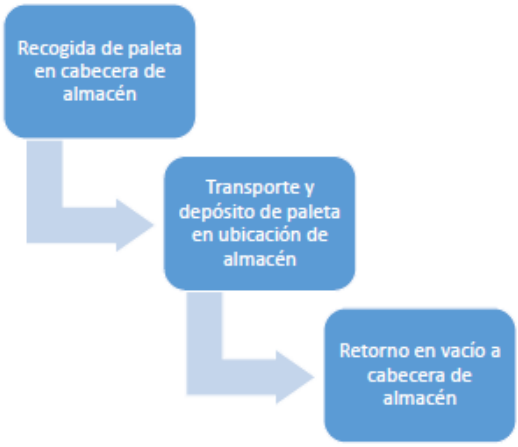


Figura 1.4.- *Operaciones a realizar en un ciclo simple de almacenaje/reposición.*

A continuación, se detallan las operaciones a realizar en un ciclo simple de almacenaje de un transelevador:

- Recogida de paleta en cabecera del almacén
- Transporte y depósito de la carga a ubicación de almacenaje.
- Retorno, sin carga, del transelevador al punto de entrada del almacén (E)

2. **Extracción** o desalmacenaje

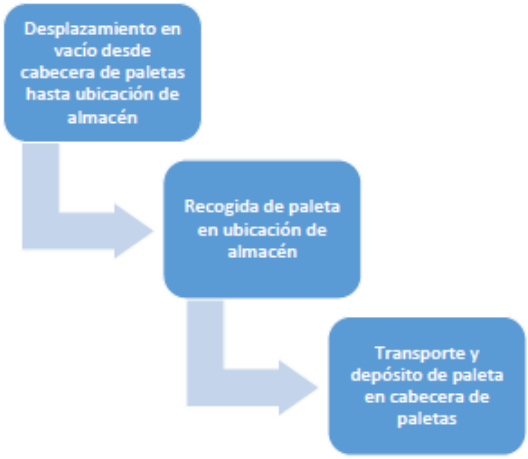


Figura 1.5.- *Operaciones a realizar en un ciclo simple de desalmacenaje/extracción.*

A continuación, se detallan las operaciones a realizar en un ciclo simple de desalmacenaje de un transelevador:

- Desplazamiento, sin carga, del transelevador desde cabecera de paletas hasta ubicación de almacén donde está localizada la carga a desalmacenar
- Recogida de paleta
- Transporte y depósito de la carga a cabecera de paletas (A)

6.4.2.2.- CICLO COMBINADO

En la definición de un ciclo combinado se definen las siguientes operaciones simultaneas: reposición y extracción (almacenaje y desalmacenaje)

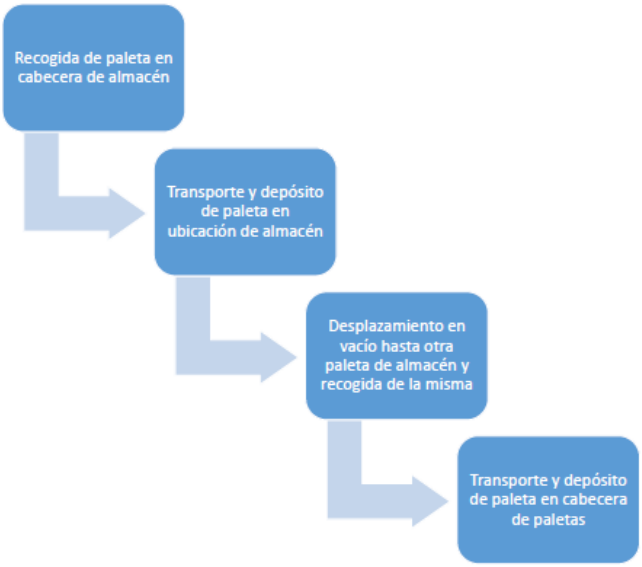


Figura 1.6.- *Operaciones a realizar en un ciclo combinado*

A continuación, se detallan las operaciones a realizar en un ciclo combinado:

- Recogida de paleta en cabecera del almacén
- Transporte y depósito de la carga a ubicación de almacenaje
- Desplazamiento, sin carga, del transelevador, hasta ubicación de almacén donde está localizada la carga a desalmacenar.
- Transporte y depósito de la carga a cabecera de paletas (A)

6.4.2.3.- CICLO MEDIO SIMPLE Y COMBINADO

A continuación, se va a definir tanto un ciclo medio simple como uno combinado. Ambos, se usarán para realizar el cálculo teórico de los ciclos de trabajo. En primer lugar, se definirá un ciclo medio simple de almacenaje o desalmacenaje, en segundo lugar, un ciclo medio combinado. En segundo lugar, se expondrán las ecuaciones matemáticas que describen dichos ciclos.

En un ciclo medio simple de almacenaje, el tiempo de ciclo se calcula como la media aritmética del tiempo necesario para realizar el trayecto de ida y vuelta entre los siguientes puntos:

• E y P1: punto de recogida E y punto de referencia P1

• E y P2: punto de recogida E y punto de referencia P2

Igualmente, en un ciclo medio simple de desalmacenaje, el tiempo de ciclo se calcula como la media aritmética del tiempo requerido para realizar el trayecto de ida y vuelta entre los siguientes puntos:

- A y P1: punto de entrega A y punto de referencia P1
- A y P2: punto de entrega A y punto de referencia P2

Por último, en un ciclo medio combinado, el tiempo de ciclo se calcula como el tiempo necesario para realizar el recorrido entre los siguientes puntos:

- E y P1: desde el punto de recogida E hasta el punto de referencia P1
- P1 y P2: desde el punto de referencia P1 hasta el punto de referencia P2
- P2 y A: desde el punto de referencia P2 hasta el punto de entrega A
- A y E: desde el punto de entrega A hasta el punto de recogida E

En primer lugar, se va a definir un ciclo medio simple. Para ello, se va a definir el tiempo de traslación de un transelevador como el valor temporal mayor necesario para completar el movimiento de desplazamiento en el eje $\overline{XX'}$ y el movimiento de elevación en el eje $\overline{YY'}$

$$t_{E\to P_1} = t_{\max(E\to P_1)} = max\{t_{x,E\to P_1}, t_{y,E\to P_1}\}$$
$$t_{E\to P_2} = t_{\max(E\to P_2)} = max\{t_{x,E\to P_2}, t_{y,E\to P_2}\}$$

Ecuación 6.1.- Tiempo de traslación de un transelevador en un ciclo medio simple

El tiempo de horquillas (t_{01}) se calcula mediante la suma de los siguientes tiempos del transelevador:

$$t_{01} = t_{posicionado} + t_{Control\ de\ ubicación} + t_{ciclo\ de\ horquilla} + t_{procesoconexiones/control}$$

Ecuación 6.2.- Tiempo de horquillas

Como se ha comentado anteriormente, el ciclo de horquilla se calcula mediante la siguiente expresión:

$$t_{ciclo\ horquilla} = t_{extensionHorquilla} + t_{elev/decensoCuna} + t_{rettraccion\ de\ la\ horquilla}$$

Ecuación 1.1.- Ciclo de horquilla

A continuación, se indica las ecuaciones que describen las operaciones de almacenaje de un transelevador entre los puntos P1 y P2:

$$t_{E\to P_1,P_{1\to E}} = t_{(E\to P_1)} + t_{(P_{1\to E})} + 2t_{01}$$
$$t_{E\to P_2,P_{2\to E}} = t_{(E\to P_2)} + t_{(P_{2\to E})} + 2t_{01}$$

Ecuación 6.3.- Tiempos de almacenaje de un traslo entre los puntos 1 y 2

En un ciclo medio simple, el tiempo de ciclo se calcula como la media aritmética de los anteriores:

$$t_{m\ (ciclo\ medio\ simple)} = \frac{t_{(E\to P_1, P_1\to E)} + t_{(E\to P_2, P_2\to E)}}{2} = \frac{t_{(E\to P_1)} + t_{(P_{1\to E})} + t_{(E\to P_2)} + t_{(P_{2\to E})}}{2} + 2t_{01}$$

Ecuación 6.4.- Tiempo de un ciclo medio simple

En segundo lugar, se va a definir un ciclo medio combinado, para ello, como en el caso anterior, el tiempo de traslación del transelevador se calcula como el mayor valor temporal necesario para completar el movimiento de desplazamiento en el eje $\overline{XX'}$ y el movimiento de elevación en el eje $\overline{YY'}$:

$$t_{E\to P_1} = t_{\max(E\to P_1)} = max\{t_{x,E\to P_1}, t_{y,E\to P_1}\}$$
$$t_{P_1\to P_2} = t_{\max(P_1\to P_2)} = max\{t_{x,P_1\to P_2}, t_{y,P_1\to P_2}\}$$
$$t_{P_2\to A} = t_{\max(P_2\to A)} = max\{t_{x,P_2\to A}, t_{y,P_2\to A}\}$$
$$t_{A\to E} = t_{\max(A\to E)} = max\{t_{x,A\to E}, t_{y,A\to E}\}$$

Ecuación 6.5.- Tiempo de traslación del traslo en un ciclo medio combinado

El tiempo de horquillas (t_{02}) se calcula de forma equivalente a la definida anteriormente.

A continuación, se indica los tiempos de ciclo del transelevador entre los puntos P_1 y P_2:

$$t_{E\to P_1} = t_{(E\to P_1)} + t_{02}$$
$$t_{P_1\to P_2} = t_{(P_1\to P_2)} + t_{02}$$
$$t_{P_2\to A} = t_{(P_2\to A)} + t_{02}$$
$$t_{A\to E} = t_{(A\to E)} + t_{02}$$

Ecuación 6.6.- Tiempos de ciclo de un traslo entre los puntos 1 y 2

En un ciclo medio combinado, el tiempo de ciclo se calcula como el tiempo empleado en el trayecto entre el punto de recogida (E) y P_1, en el trayecto de P_1 a P_2, del punto P_2 al punto de entrega A y del punto A al punto de almacenaje (E):

$$t_{m\ (ciclo\ medio\ combinado)} = t_{(E\to P_1)} + t_{(P_1\to P_2)} + t_{(P_2\to A)} t_{(A\to E)} + 4t_{02}$$

Ecuación 6.7.- Tiempo de ciclo medio combinado

El cálculo del tiempo de ciclo de trabajo se basa en los siguientes parámetros:

- Altura y longitud del almacén.
- Posición de los puntos de entrada y salida.

A continuación, se va a definir los casos de ensayo definidos en la norma FEM 9.851 (2003).

Para esta prueba se ha tenido la necesidad de redimensionar el sistema de almacenaje, es decir, se han modificado las dimensiones de la estantería y del transelevador para poder verificar los tiempos de trabajo.

Se han realizado las pruebas que aplican al modelo de transelevador diseñado de la normativa FEM 9.851

6.4.2.4.- CASO 4.1. NORMA FEM 9.851 (06.2003). PUNTOS DE RECOGIDA Y TRASLADO EN EL VÉRTICE INFERIOR.

En las tablas siguientes se indica las ecuaciones que permiten determinar las coordenadas de los valores estadísticos de referencia (P_1, P_2), usados en el cálculo del tiempo de ciclo de trabajo de los transelevadores, de acuerdo a la norma **FEM 9.851 (06.2003).**

Punto	Coordenadas	
	X	Y
E=A	0	0
P_1	$\frac{L}{5}$	$\frac{2H}{3}$
P_2	$\frac{2L}{3}$	$\frac{H}{5}$

Tabla 6.1.- *Norma FEM 9.851 – Caso 4.1 - Coordenadas genéricas.*

En el siguiente gráfico se muestra los puntos P_1 y P_2 calculados según las coordenadas de la tabla anterior:

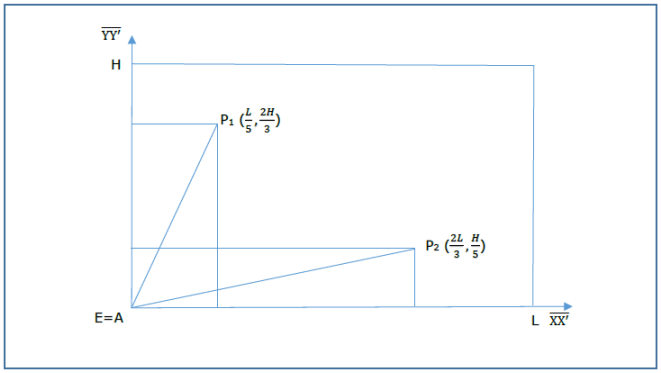


Figura 6.13.- *Norma FEM 9.851 – Caso 4.1 – Puntos P1 y P2*

E	Punto de almacenaje/entrada
A	Punto de desalmacenaje/salida
$\mathbf{P_{1E} \Leftrightarrow P_1}$	Distancia (en valor absoluto) entre el punto de almacén (E) y el punto estadístico de referencia P_1
$\mathbf{P_{1A} \Leftrightarrow P'_1}$	Distancia (en valor absoluto) entre el punto de desalmacenaje (A) y el punto estadístico de referencia P_1
Tiempo para un ciclo medio simple: almacenaje	**Media aritmética** del tiempo empleado en el trayecto de **ida y vuelta** entre los siguientes puntos: • E y P_1: punto de recogida E y punto de referencia P_1 • E y P_2: punto de recogida E y punto de referencia P_2
Tiempo para un ciclo medio simple: desalmacenaje	**Media aritmética** del tiempo empleado en el trayecto de **ida y vuelta** entre los siguientes puntos: • A y P_1: punto de entrega A y punto de referencia P_1 • A y P_2: punto de entrega A y punto de referencia P_2
Tiempo para un ciclo medio combinado	**Tiempo empleado** en el trayecto entre los siguientes puntos: • E y P_1: punto de recogida (E) y punto de referencia P_1 • P_1 y P_2 : puntos de referencia P_1 y P_2 • P_2 y A: punto de referencia P_2 y punto de entrega (A) • A y E: punto A de salida (A) y punto de entrada (E)

Tabla 6.2.- *Norma FEM 9.851 – Caso 4.1 - Definiciones aplicables*

A continuación, se muestra las coordenadas anteriores en un almacén de 100 m de longitud por 100 m de altura:

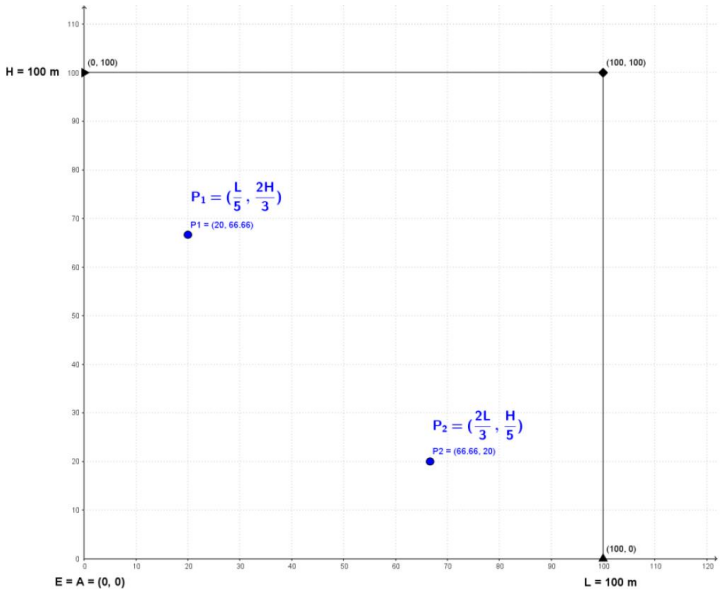


Figura 6.14.- *Norma FEM 9.851 – Caso 4.1 – Coord. P1 y P2 (Almacén DIm. 100m x 100m)*

A continuación, se define un ciclo simple de entrada:

- Almacenaje. En un ciclo medio simple, el tiempo se calcula como la media aritmética de los siguientes tiempos (incluyendo los tiempos constantes de transelevador u horquillas):

- $t_{P1E} =>$ el tiempo, en un ciclo simple de almacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de recogida (E) y el punto de referencia P_1
- $t_{P2E} =>$ el tiempo, en un ciclo simple de almacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de recogida (E) y el punto de referencia P_2

En el siguiente dibujo se detalla la trayectoria de un ciclo medio simple de entrada/almacenaje:

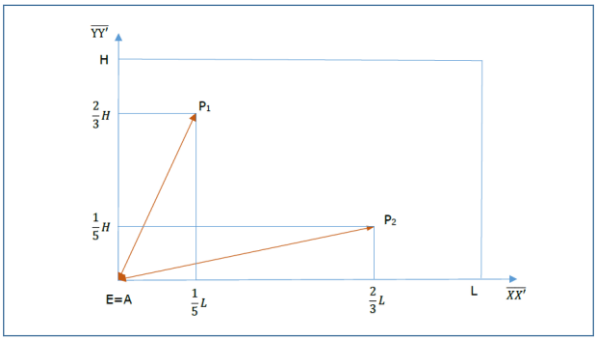


Figura 6.15.- *Norma FEM 9.851 – Caso 4.1 – Trayectoria ciclo medio simple de entrada*

A continuación, se muestra la trayectoria de un ciclo medio simple en un almacén de 100 m de longitud por 100 m de altura

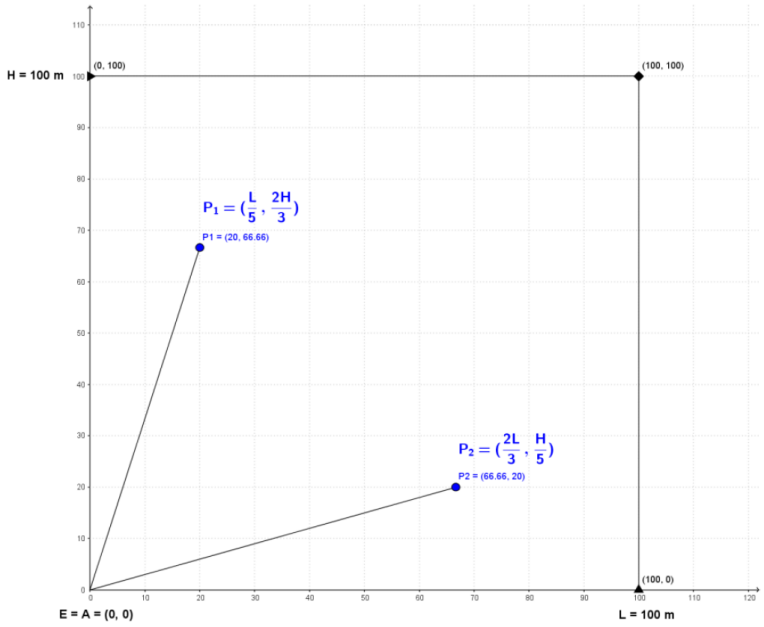


Figura 6.16.- *Norma FEM 9.851 – Caso 4.1 – Trayectoria ciclo medio simple **entrada** (Almacén DIm. 100m x 100m)*

A continuación, se define un ciclo simple de salida:

- Desalmacenaje. El tiempo, en un ciclo medio simple, se calcula como la media aritmética de los siguientes (incluyendo los tiempos constantes de transelevador u horquillas).
  - $t_{P\prime 1} = t_{P1A} =>$ el tiempo, en un ciclo simple en desalmacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de entrega (A) y el punto de referencia P'_1 (medido tomando como eje de coordenadas el punto A).
  - $t_{P\prime 2} = t_{P2A} =>$ el tiempo, en un ciclo simple en desalmacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de entrega (A) y el punto de referencia P'_2 (medido tomando como eje de coordenadas el punto A).

En este caso, coinciden los puntos A y E, por lo tanto, se verifica la siguiente igualdad:

- $P_1 = P'_1 \;\; y \;\; P_2 = P'_2$

Ecuación 6.8.- Igualdades entre los puntos característicos

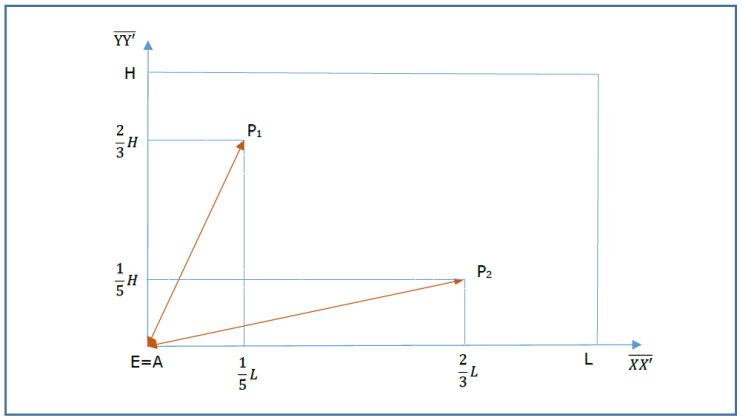


Figura 6.17.- *Norma FEM 9.851 – Caso 4.1 – Trayectoria ciclo medio simple de salida*

A continuación, se define el ciclo combinado:

- Ciclo combinado. En un ciclo medio combinado, el tiempo se calcula como la media aritmética de los siguientes (incluyendo los tiempos constantes de transelevador u horquillas).
  - $t_{P1;\,P2,\;\;A;\,E} \leftrightarrow t_{P1E;\,P2E,\;\;A;\,E} =>$ El tiempo, en un ciclo combinado, se calcula como el tiempo empleado en el trayecto de ida entre los siguientes puntos:
    - Punto de recogida (E) y P_1
    - Punto P_1 y P_2
    - Punto P_2 y punto de entrega (A)
    - Punto de entrega (A) y punto de recogida (E)

- $t_{P\prime 1;\, P\prime 2,\quad A;\, E} \leftrightarrow t_{P1A;\, P2A,\quad A;\, E}$ =>El tiempo, en un ciclo combinado, se calcula como el tiempo empleado en el trayecto de ida entre los siguientes puntos:
  - Punto de recogida (E) y P'_1
  - Punto P'_1 y P'_2
  - Punto P'_2 y punto de entrega (A)
  - Punto de entrega (A) y punto de recogida (E)

En el siguiente dibujo se detalla la trayectoria de un ciclo medio combinado:

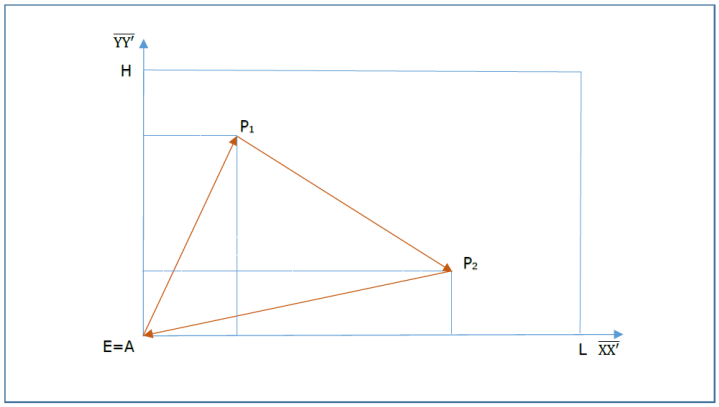


Figura 6.18.- *Trayectoria de un ciclo medio combinado*

A continuación, se detalla la trayectoria de un ciclo medio combinado en un almacén de 100 m de longitud por 100 m de altura:

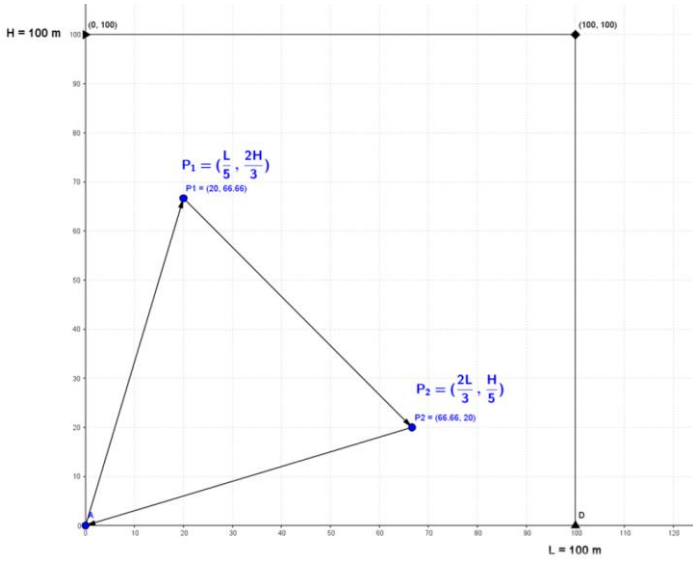
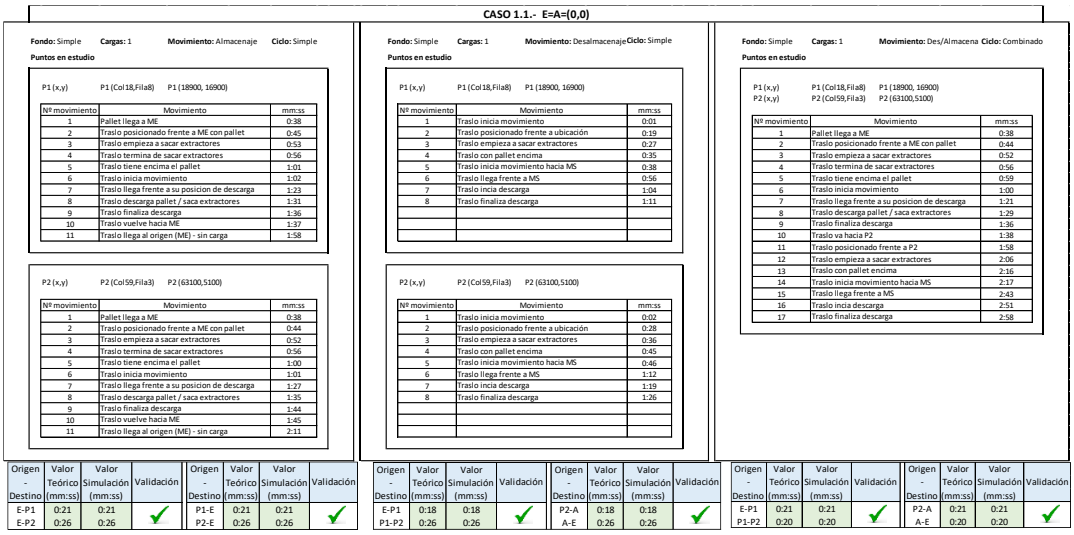


Figura 6.19.- *Norma FEM 9.851 – Caso 4.1 – Trayectoria ciclo medio combinado (Almacén DIm. 100m x 100m)*

Las validaciones a las ruebas realizadas se muestran en la siguiente tabla:



CASO 1.1.- E=A=(0,0)

Fondo: Simple **Cargas:** 1 **Movimiento:** Almacenaje **Ciclo:** Simple

Puntos en estudio

P1 (x,y) P1 (Col18,Fila8) P1 (18900, 16900)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:45
3	Traslo empieza a sacar extractores	0:53
4	Traslo termina de sacar extractores	0:56
5	Traslo tiene encima el pallet	1:01
6	Traslo inicia movimiento	1:02
7	Traslo llega frente a su posicion de descarga	1:23
8	Traslo descarga pallet / saca extractores	1:31
9	Traslo finaliza descarga	1:36
10	Traslo vuelve hacia ME	1:37
11	Traslo llega al origen (ME) - sin carga	1:58

P2 (x,y) P2 (Col59,Fila3) P2 (63100,5100)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:44
3	Traslo empieza a sacar extractores	0:52
4	Traslo termina de sacar extractores	0:56
5	Traslo tiene encima el pallet	1:00
6	Traslo inicia movimiento	1:01
7	Traslo llega frente a su posicion de descarga	1:27
8	Traslo descarga pallet / saca extractores	1:35
9	Traslo finaliza descarga	1:44
10	Traslo vuelve hacia ME	1:45
11	Traslo llega al origen (ME) - sin carga	2:11

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:21	0:21	✓
E-P2	0:26	0:26	
P1-E	0:21	0:21	✓
P2-E	0:26	0:26	

Fondo: Simple **Cargas:** 1 **Movimiento:** Desalmacenaje **Ciclo:** Simple

Puntos en estudio

P1 (x,y) P1 (Col18,Fila8) P1 (18900, 16900)

Nº movimiento	Movimiento	mm:ss
1	Traslo inicia movimiento	0:01
2	Traslo posicionado frente a ubicación	0:19
3	Traslo empieza a sacar extractores	0:27
4	Traslo con pallet encima	0:35
5	Traslo inicia movimiento hacia MS	0:38
6	Traslo llega frente a MS	0:56
7	Traslo incia descarga	1:04
8	Traslo finaliza descarga	1:11

P2 (x,y) P2 (Col59,Fila3) P2 (63100,5100)

Nº movimiento	Movimiento	mm:ss
1	Traslo inicia movimiento	0:02
2	Traslo posicionado frente a ubicación	0:28
3	Traslo empieza a sacar extractores	0:36
4	Traslo con pallet encima	0:45
5	Traslo inicia movimiento hacia MS	0:46
6	Traslo llega frente a MS	1:12
7	Traslo incia descarga	1:19
8	Traslo finaliza descarga	1:26

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:18	0:18	✓
P1-P2	0:26	0:26	
P2-A	0:18	0:18	✓
A-E	0:26	0:26	

Fondo: Simple **Cargas:** 1 **Movimiento:** Des/Almacena **Ciclo:** Combinado

Puntos en estudio

P1 (x,y) P1 (Col18,Fila8) P1 (18900, 16900)
P2 (x,y) P2 (Col59,Fila3) P2 (63100,5100)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:44
3	Traslo empieza a sacar extractores	0:52
4	Traslo termina de sacar extractores	0:56
5	Traslo tiene encima el pallet	0:59
6	Traslo inicia movimiento	1:00
7	Traslo llega frente a su posicion de descarga	1:21
8	Traslo descarga pallet / saca extractores	1:29
9	Traslo finaliza descarga	1:36
10	Traslo va hacia P2	1:38
11	Traslo posicionado frente a P2	1:58
12	Traslo empieza a sacar extractores	2:06
13	Traslo con pallet encima	2:16
14	Traslo inicia movimiento hacia MS	2:17
15	Traslo llega frente a MS	2:43
16	Traslo incia descarga	2:51
17	Traslo finaliza descarga	2:58

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:21	0:21	✓
P1-P2	0:20	0:20	
P2-A	0:21	0:21	✓
A-E	0:20	0:20	

Tabla 6.3.- *Verificación de ciclos en norma FEM 9.851 (06.2003)- caso 4.1*

6.4.2.5.- CASO 4.2. NORMA FEM 9.851 (06.2003). PUNTOS DE RECOGIDA EN VÉRTICE (E) Y TRASLADO AL VÉRTICE (A)

Puntos		Coordenadas	
		X	Y
E		0	0
A		L	0
Almacenaje => distancia entre el punto de almacenaje (E) y los puntos de referencia (P_1, P_2)	P_1	$\frac{L}{5}$	$\frac{2H}{3}$
	P_2	$\frac{2L}{3}$	$\frac{H}{5}$
Desalmacenaje => distancia entre el punto de desalmacenaje (A) y los puntos de referencia (P'_1 y P'_2)	P'_1	$\frac{4L}{5}$ (2)	$\frac{2H}{3}$
	P'_2	$\frac{L}{3}$ (1)	$\frac{H}{5}$

Tabla 6.4.- *Norma FEM 9.851 – Caso 4.2 – Coordenadas genéricas*

A continuación, se define los ciclos simples de reposición y extracción:

- Almacenaje. En un ciclo medio simple, el tiempo se calcula como la media aritmética de los siguientes tiempos (incluyendo los tiempos constantes de horquillas o transelevador):

- $t_{P1E} =>$ el tiempo, en un ciclo simple en almacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de recogida (E) y el punto de referencia P_1
- $t_{P2E} =>$ el tiempo, en un ciclo simple en almacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de recogida (E) y el punto de referencia P_2

En el siguiente dibujo se detalla la trayectoria de un ciclo medio simple de entrada/almacenaje:

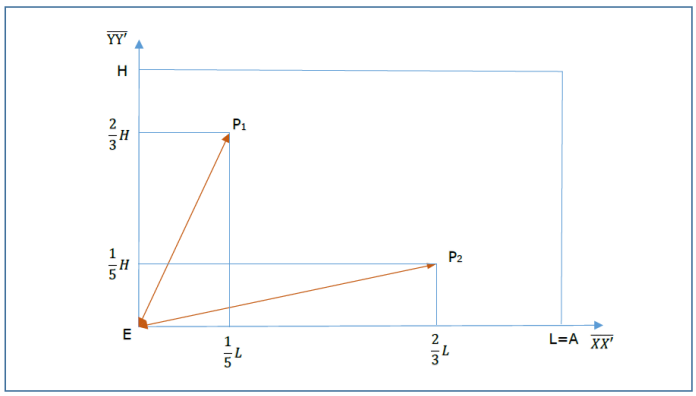


Figura 6.20.- *Norma FEM 9.851 – Caso 4.2 – Trayectoria ciclo medio simple de almacenaje*

A continuación, se muestra las coordenadas de entrada/almacenaje en un almacén de 100m de longitud por 100 m de altura:

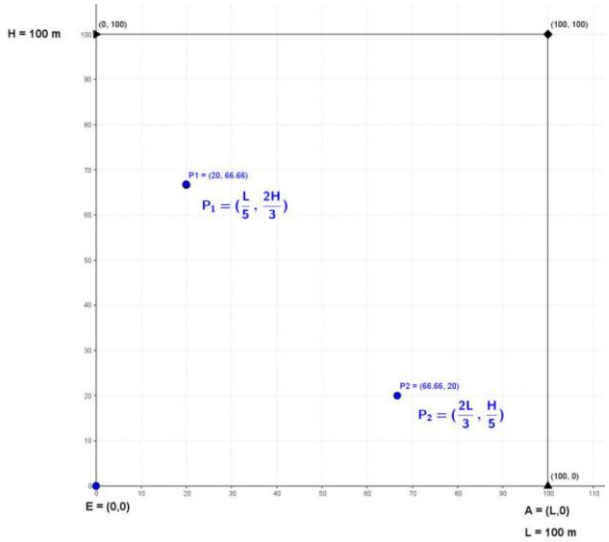


Figura 6.21.- *Norma FEM 9.851 – Caso 4.2- Puntos P1 y P2 de almacenaje (Almacén Dim. 100m x 100m)*

A continuación, se muestra la trayectoria de un ciclo medio simple de entrada en un almacén de 100 m de longitud por 100m de altura:

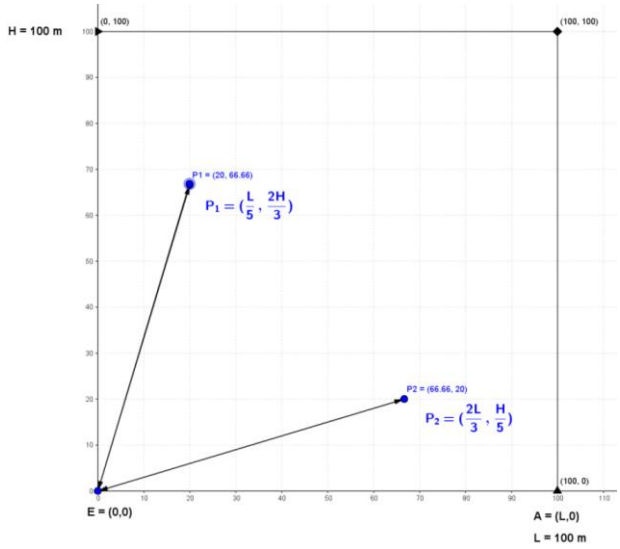


Figura 6.22.- *Norma FEM 9.851 – Caso 4.2- Ciclo medio simple de entrada/almacenaje (Almacén Dim. 100m x 100m)*

- Desalmacenaje. El tiempo, en un ciclo medio simple, se calcula como la media aritmética de los siguientes (incluyendo los tiempos constantes de horquillas o transelevador).
  - $t_{P'1A} =>$ el tiempo, en un ciclo simple en desalmacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de entrega (A) y el punto de referencia P'$_1$, cuyas coordenadas, tomando como punto de referencia el centro de desalmacenaje, han sido calculadas en la tabla anterior $(\frac{4L}{5}, \frac{2H}{3})$.
  - $t_{P'2A} =>$ el tiempo, en un ciclo simple en desalmacenaje, se calcula como el tiempo empleado en el trayecto de ida y vuelta entre el punto de entrega (A) y el punto de referencia P'$_2$, cuyas coordenadas han sido calculadas en la tabla anterior $(\frac{L}{3}, \frac{H}{5})$

En el siguiente dibujo se detalla la trayectoria de un ciclo medio simple de salida/desalmacenaje:

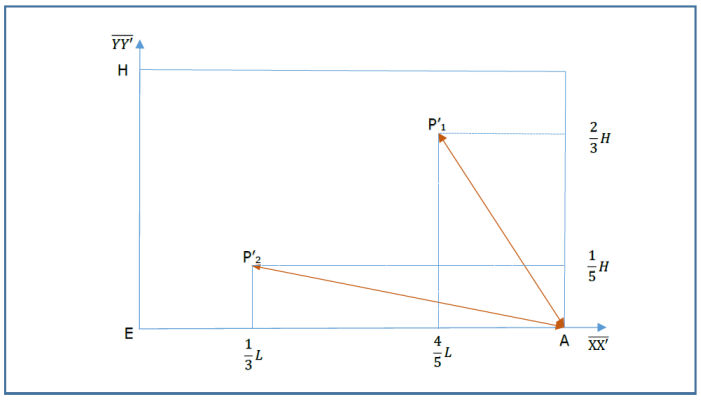


Figura 6.23.- *Norma FEM 9.851 – Caso 4.2- Trayectoria de un ciclo medio simple de salida*

A continuación, se muestra las coordenadas de salida/desalmacenaje en un almacén de 100 m de longitud por 100 m de altura:

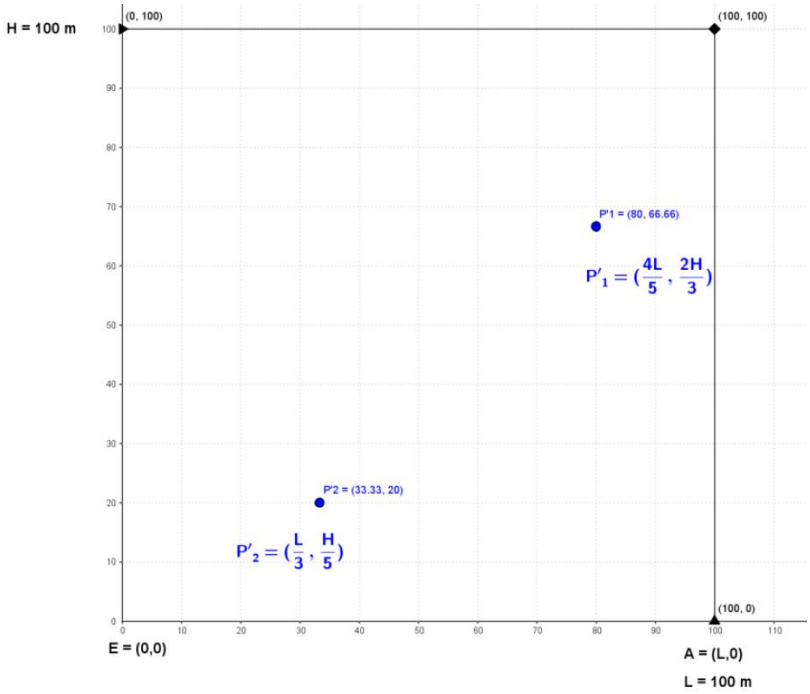


Figura 6.24.- *Norma FEM 9.851 – Caso 4.2- Puntos P1 y P2 de salida (Almacén Dim. 100m x 100m)*

A continuación, se muestra la trayectoria de un ciclo medio simple de salida/desalmacenaje en un almacén de 100 m de longitud por 100 m de altura:

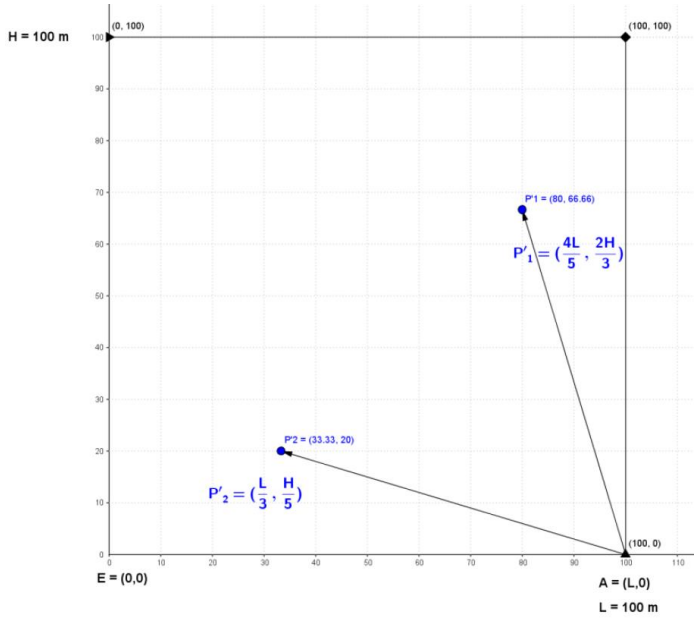


Figura 6.25.- *Norma FEM 9.851 – Caso 4.2- Ciclo medio simple de salida (Almacén Dim. 100m x 100m)*

A continuación, se define el ciclo combinado:

- Ciclo combinado. En un ciclo medio combinado, el tiempo se calcula como la media aritmética de los siguientes (incluyendo los tiempos constantes de horquillas o transelevador).
  - $t_{P1\,;\,P2,\quad A;E} \Leftrightarrow t_{P1E\,;\,P2A,\quad A;E} \Leftrightarrow t_{P1E\,;\,P2E,\quad A;E}$ => El tiempo, en un ciclo combinado, se calcula como el tiempo empleado en el trayecto de ida entre los siguientes puntos:
    - Punto de recogida (E) y P_1
    - Punto P_1 y P_2
    - Punto P_2 y punto de entrega (A)
    - Punto de entrega (A) y punto de recogida (E)
  - $t_{P\prime1\,;\,P\prime2,\quad A;E} \Leftrightarrow t_{P\prime1E\,;\,P\prime2A,\quad A;E} \Leftrightarrow t_{P1A\,;\,P2A,\quad A;E}$ => El tiempo, en un ciclo combinado, se calcula como el tiempo empleado en el trayecto de ida entre los siguientes puntos:
    - Punto de recogida (E) y P'_1
    - Punto P'_1 y P'_2
    - Punto P'_2 y punto de entrega (A)
    - Punto de entrega (A) y punto de recogida (E)

En primer lugar, se va a tomar como referencia el punto de entrada (E). Por lo tanto, en el siguiente dibujo se detalla la trayectoria de un ciclo combinado tomando como referencia el punto de entrada:

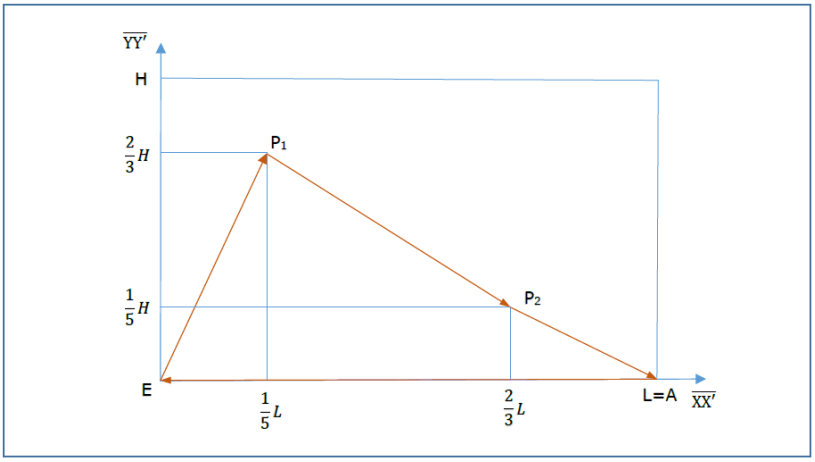


Figura 6.26.- *Norma FEM 9.851 – Caso 4.2- Trayectoria de un ciclo combinado con referencia al punto de entrada*

A continuación, se muestra las coordenadas en un almacén de 100 m de longitud por 100m de altura:

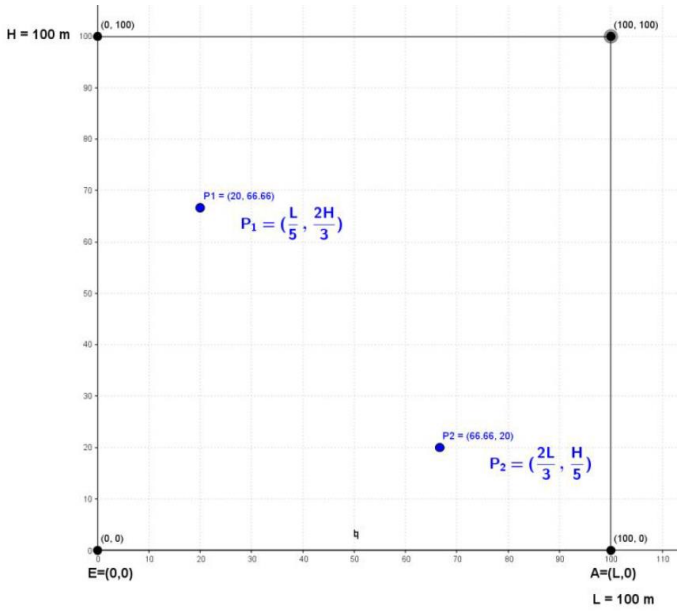


Figura 6.27.- *Norma FEM 9.851 –Caso 4.2 – Puntos P1 y P2 desde la entrada (Almacén Dim. 100m x 100m)*

A continuación, se muestra la trayectoria de un ciclo medio combinado en un almacén de 100 m de longitud por 100 m de altura:

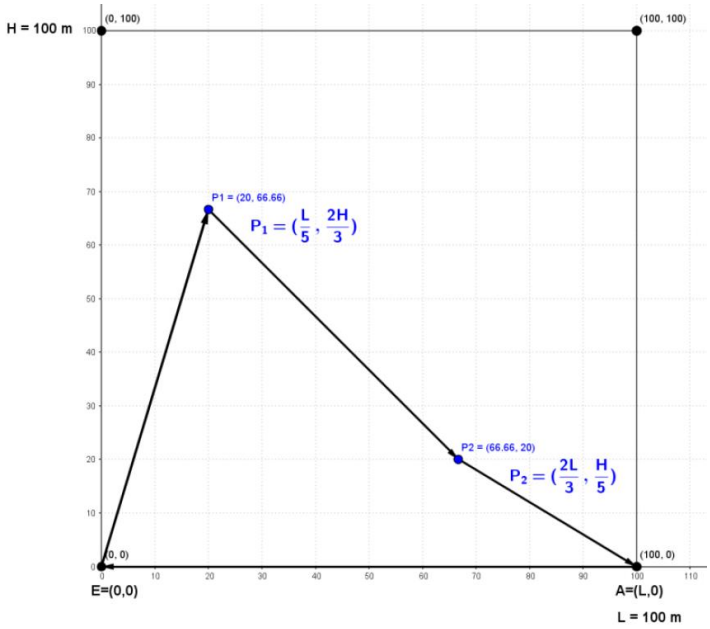


Figura 6.28.- *Norma FEM 9.851 –Caso 4.2 – Trayectoria de ciclo medio combinado (Almacén Dim. 100m x 100m)*

En segundo lugar, se va a tomar como referencia el punto de salida (A). Por lo tanto, en el siguiente dibujo se detalla la trayectoria de un ciclo combinado tomando como referencia el punto de salida:

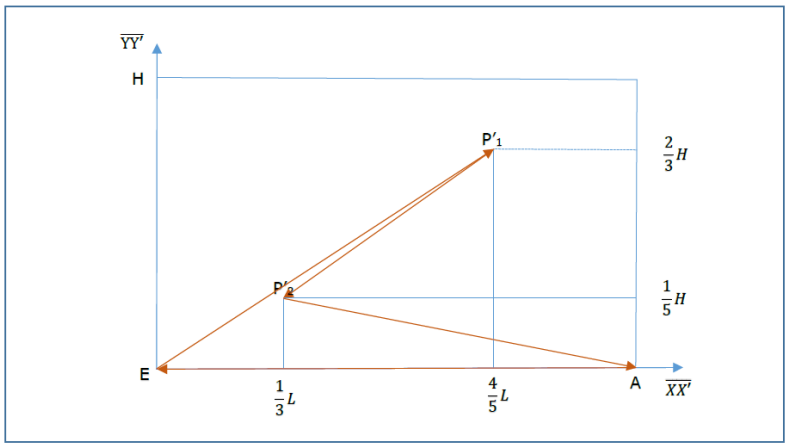


Figura 6.29.- *Norma FEM 9.851 – Caso 4.2- Trayectoria de un ciclo combinado con referencia el punto de salida*

Por lo tanto, a continuación, se muestra las coordenadas en un almacén de 100 m de longitud por 100 m de altura:

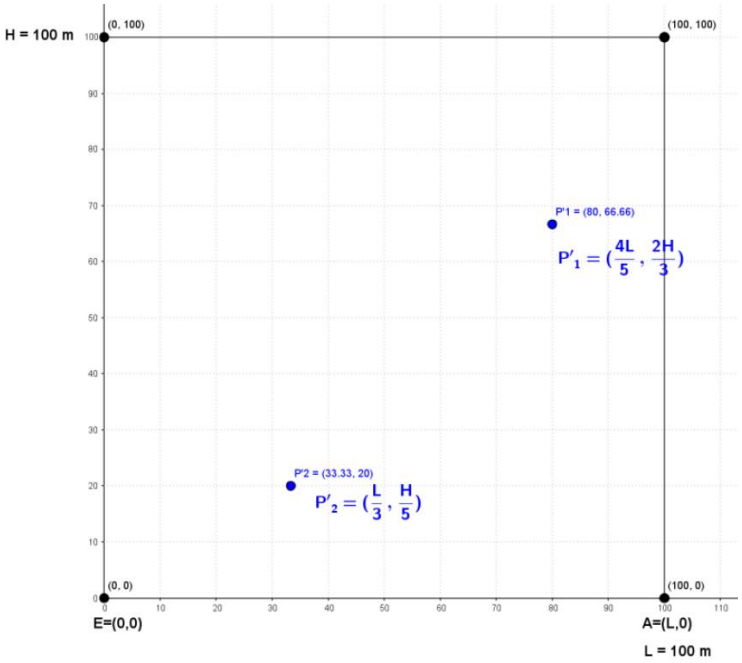


Figura 6.30.- *Norma FEM 9.851 –Caso 4.2 – Puntos P'1 y P'2 desde la salida (Almacén Dim. 100m x 100m)*

A continuación, se muestra la trayectoria de un ciclo medio combinado en un almacén de 100 m de longitud por 100 m de altura

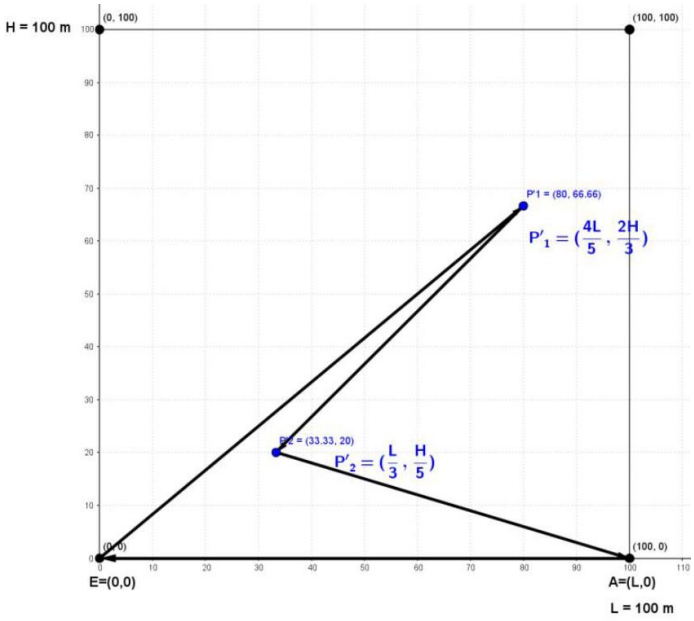
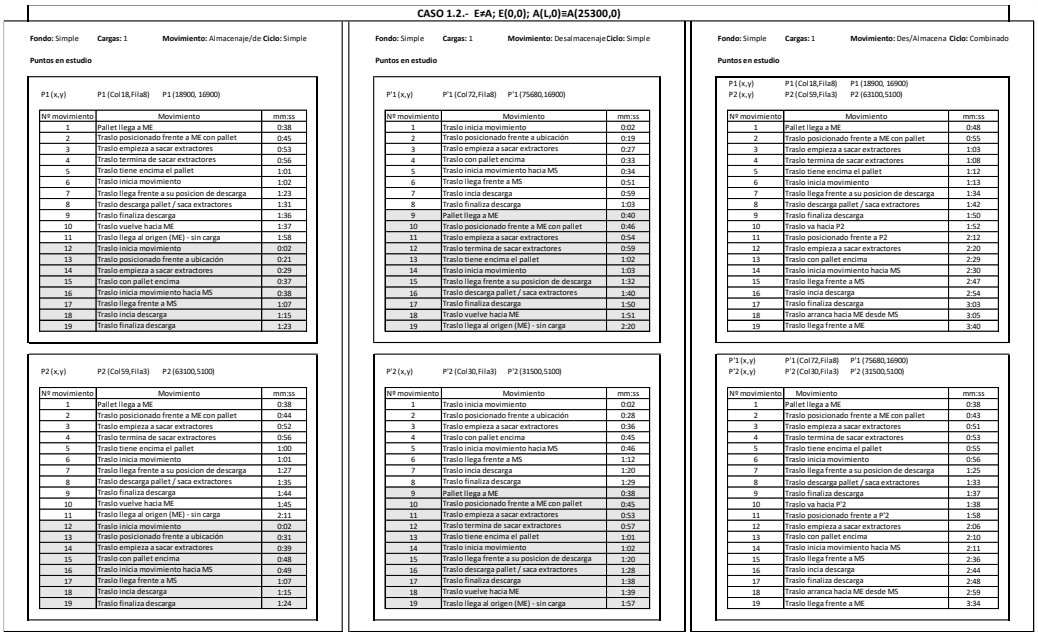


Figura 6.31.- *Norma FEM 9.851 –Caso 4.2 – Trayectoria de un ciclo medio combinado (Almacén Dim. 100m x 100m)*

Las siguientes tablas muestran los datos de las validaciones realizadas para el caso 4.2. de la norma FEM 9.851:



CASO 1.2.- E≠A; E(0,0); A(L,0)≡A(25300,0)

Fondo: Simple **Cargas:** 1 **Movimiento:** Almacenaje/de **Ciclo:** Simple

Puntos en estudio

P1 (x,y) P1 (Col 18,Fila8) P1 (18900, 16900)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:45
3	Traslo empieza a sacar extractores	0:53
4	Traslo termina de sacar extractores	0:56
5	Traslo tiene encima el pallet	1:01
6	Traslo inicia movimiento	1:02
7	Traslo llega frente a su posicion de descarga	1:23
8	Traslo descarga pallet / saca extractores	1:31
9	Traslo finaliza descarga	1:36
10	Traslo vuelve hacia ME	1:37
11	Traslo llega al origen (ME) - sin carga	1:58
12	Traslo inicia movimiento	0:02
13	Traslo posicionado frente a ubicación	0:21
14	Traslo empieza a sacar extractores	0:29
15	Traslo con pallet encima	0:37
16	Traslo inicia movimiento hacia MS	0:38
17	Traslo llega frente a MS	1:07
18	Traslo incia descarga	1:15
19	Traslo finaliza descarga	1:23

P2 (x,y) P2 (Col 59,Fila3) P2 (63100,5100)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:44
3	Traslo empieza a sacar extractores	0:52
4	Traslo termina de sacar extractores	0:56
5	Traslo tiene encima el pallet	1:00
6	Traslo inicia movimiento	1:01
7	Traslo llega frente a su posicion de descarga	1:27
8	Traslo descarga pallet / saca extractores	1:35
9	Traslo finaliza descarga	1:44
10	Traslo vuelve hacia ME	1:45
11	Traslo llega al origen (ME) - sin carga	2:11
12	Traslo inicia movimiento	0:02
13	Traslo posicionado frente a ubicación	0:31
14	Traslo empieza a sacar extractores	0:39
15	Traslo con pallet encima	0:48
16	Traslo inicia movimiento hacia MS	0:49
17	Traslo llega frente a MS	1:07
18	Traslo incia descarga	1:15
19	Traslo finaliza descarga	1:24

Fondo: Simple **Cargas:** 1 **Movimiento:** DesalmacenajeCiclo: Simple

Puntos en estudio

P'1 (x,y) P'1 (Col 72,Fila8) P'1 (75680,16900)

Nº movimiento	Movimiento	mm:ss
1	Traslo inicia movimiento	0:02
2	Traslo posicionado frente a ubicación	0:19
3	Traslo empieza a sacar extractores	0:27
4	Traslo con pallet encima	0:33
5	Traslo inicia movimiento hacia MS	0:34
6	Traslo llega frente a MS	0:51
7	Traslo incia descarga	0:59
8	Traslo finaliza descarga	1:03
9	Pallet llega a ME	0:40
10	Traslo posicionado frente a ME con pallet	0:46
11	Traslo empieza a sacar extractores	0:54
12	Traslo termina de sacar extractores	0:59
13	Traslo tiene encima el pallet	1:02
14	Traslo inicia movimiento	1:03
15	Traslo llega frente a su posicion de descarga	1:32
16	Traslo descarga pallet / saca extractores	1:40
17	Traslo finaliza descarga	1:50
18	Traslo vuelve hacia ME	1:51
19	Traslo llega al origen (ME) - sin carga	2:20

P'2 (x,y) P'2 (Col 30,Fila3) P'2 (31500,5100)

Nº movimiento	Movimiento	mm:ss
1	Traslo inicia movimiento	0:02
2	Traslo posicionado frente a ubicación	0:28
3	Traslo empieza a sacar extractores	0:36
4	Traslo con pallet encima	0:45
5	Traslo inicia movimiento hacia MS	0:46
6	Traslo llega frente a MS	1:12
7	Traslo incia descarga	1:20
8	Traslo finaliza descarga	1:29
9	Pallet llega a ME	0:38
10	Traslo posicionado frente a ME con pallet	0:45
11	Traslo empieza a sacar extractores	0:53
12	Traslo termina de sacar extractores	0:57
13	Traslo tiene encima el pallet	1:01
14	Traslo inicia movimiento	1:02
15	Traslo llega frente a su posicion de descarga	1:20
16	Traslo descarga pallet / saca extractores	1:28
17	Traslo finaliza descarga	1:38
18	Traslo vuelve hacia ME	1:39
19	Traslo llega al origen (ME) - sin carga	1:57

Fondo: Simple **Cargas:** 1 **Movimiento:** Des/Almacena **Ciclo:** Combinado

Puntos en estudio

P1 (x,y) P1 (Col 18,Fila8) P1 (18900, 16900)
P2 (x,y) P2 (Col 59,Fila3) P2 (63100,5100)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:48
2	Traslo posicionado frente a ME con pallet	0:55
3	Traslo empieza a sacar extractores	1:03
4	Traslo termina de sacar extractores	1:08
5	Traslo tiene encima el pallet	1:12
6	Traslo inicia movimiento	1:13
7	Traslo llega frente a su posicion de descarga	1:34
8	Traslo descarga pallet / saca extractores	1:42
9	Traslo finaliza descarga	1:50
10	Traslo va hacia P2	1:52
11	Traslo posicionado frente a P2	2:12
12	Traslo empieza a sacar extractores	2:20
13	Traslo con pallet encima	2:29
14	Traslo inicia movimiento hacia MS	2:30
15	Traslo llega frente a MS	2:47
16	Traslo incia descarga	2:54
17	Traslo finaliza descarga	3:03
18	Traslo arranca hacia ME desde MS	3:05
19	Traslo llega frente a ME	3:40

P'1 (x,y) P'1 (Col 72,Fila8) P'1 (75680,16900)
P'2 (x,y) P'2 (Col 30,Fila3) P'2 (31500,5100)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:43
3	Traslo empieza a sacar extractores	0:51
4	Traslo termina de sacar extractores	0:53
5	Traslo tiene encima el pallet	0:55
6	Traslo inicia movimiento	0:56
7	Traslo llega frente a su posicion de descarga	1:25
8	Traslo descarga pallet / saca extractores	1:33
9	Traslo finaliza descarga	1:37
10	Traslo va hacia P'2	1:38
11	Traslo posicionado frente a P'2	1:58
12	Traslo empieza a sacar extractores	2:06
13	Traslo con pallet encima	2:10
14	Traslo inicia movimiento hacia MS	2:11
15	Traslo llega frente a MS	2:36
16	Traslo incia descarga	2:44
17	Traslo finaliza descarga	2:48
18	Traslo arranca hacia ME desde MS	2:59
19	Traslo llega frente a ME	3:34

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:21	0:21	✓
E-P2	0:26	0:26	
A-P1	0:29	0:29	
A-P2	0:18	0:18	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
P1-E	0:21	0:21	✓
P2-E	0:26	0:26	
P1-A	0:29	0:29	
P2-A	0:18	0:18	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
P'1-A	0:17	0:17	✓
P'2-A	0:26	0:26	
E-P'1	0:29	0:29	
E-P'2	0:18	0:18	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
A-P'1	0:17	0:17	✓
A-P'2	0:26	0:26	
P'1-E	0:29	0:29	
P'2-E	0:18	0:18	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:21	0:21	✓
P1-P2	0:20	0:20	
P2-A	0:17	0:17	
A-E	0:35	0:35	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P'1	0:29	0:29	✓
P'1-P'2	0:20	0:20	
P'2-A	0:25	0:25	
A-E	0:35	0:35	

Tabla 6.5.- *Verificación de ciclos en norma FEM 9.851 (06.2003)- caso 4.2*

6.4.2.6.- CASO 4.3. NORMA FEM 9.851 (06.2003). PUNTOS DE RECOGIDA EN VÉRTICE (E) Y TRASLADO AL VÉRTICE (A) DESPLAZADO EN YY'

Punto	Coordenadas		
	X	Y	
E=A	0	Y_E	
P_1	$\frac{L}{5}$	$0 < yE \leq \frac{H}{2}$	$y1 = \frac{2H}{3} + \frac{yE}{3}$
		$\frac{H}{2} < yE \leq H$	$y1 = \frac{yE}{3}$
P_2	$\frac{2L}{3}$	$0 < yE \leq \frac{H}{2}$	$y2 = \frac{H}{5} + \frac{yE}{3}$
		$\frac{H}{2} < yE \leq H$	$y2 = \frac{7H}{15} + \frac{yE}{3}$

Tabla 6.6.- *Norma FEM 9.851- caso 4.3 – Coordenadas genéricas*

A continuación, se muestra las coordenadas anteriores en un almacén de 100 m de longitud por 100 m de altura, en el que las coordenadas de los puntos de entrada y salida son inferiores o iguales a 50 m:

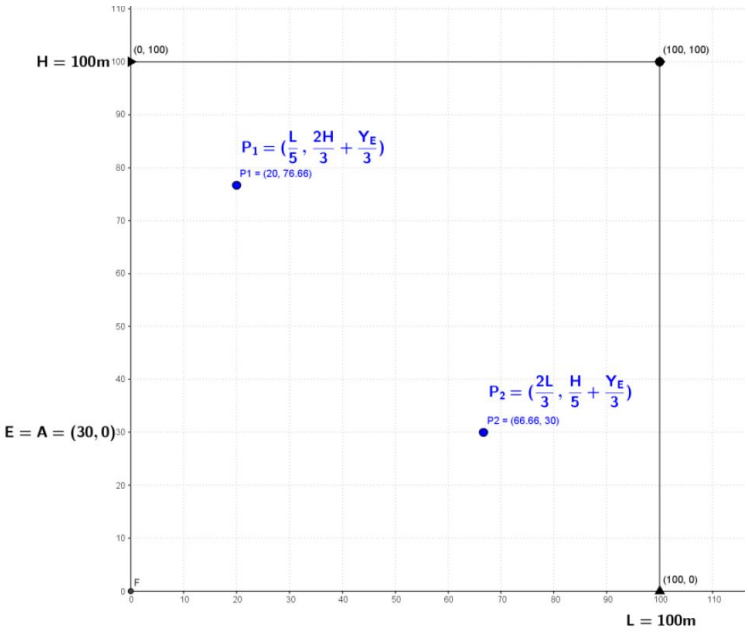


Figura 6.32.- *Norma FEM 9.851 – Caso 4.3 – puntos P1 y P2 (coord. e/s <= 50 m)*

En el siguiente dibujo se muestra la trayectoria de un ciclo medio simple en un almacén de 100 m de longitud por 100 m de altura:

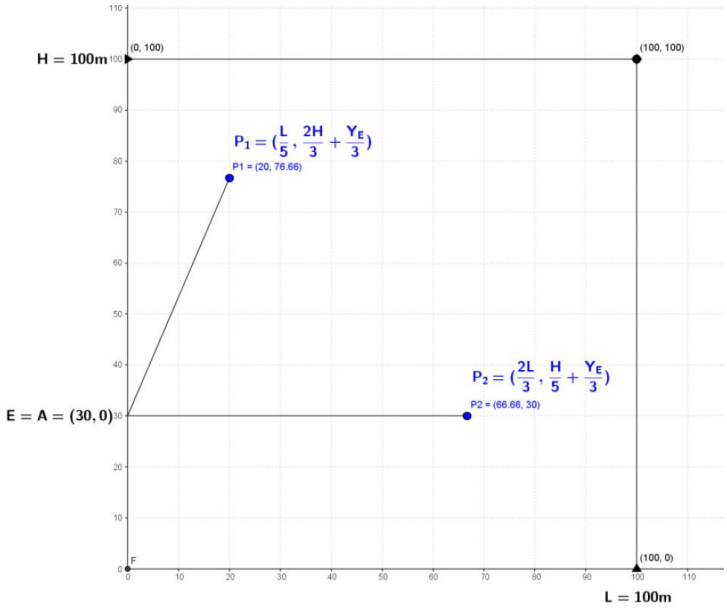


Figura 6.33.- *Norma FEM 9.851 – Caso 4.3 – Trayectoria ciclo medio simple – coord. e/s <= 50 m (Almacén DIm. 100m x 100m)*

En el siguiente dibujo se muestra la trayectoria de un ciclo medio combinado:

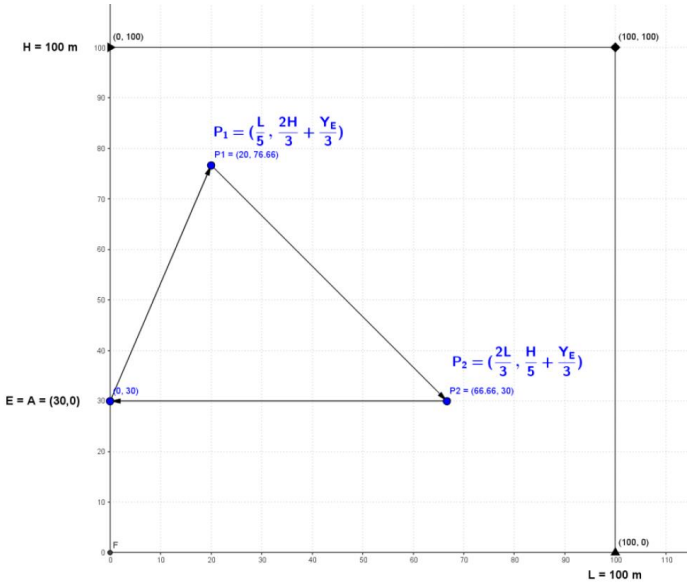


Figura 6.34.- *Norma FEM 9.851 – Caso 4.3 – Trayectoria ciclo medio combinado – coord. e/s <=50m (Almacén DIm. 100m x 100m)*

A continuación, se muestra las coordenadas anteriores en un almacén de 100 m de longitud por 100 m de altura, en el que las coordenadas de los puntos de entrada y salida son superiores a 50 m:

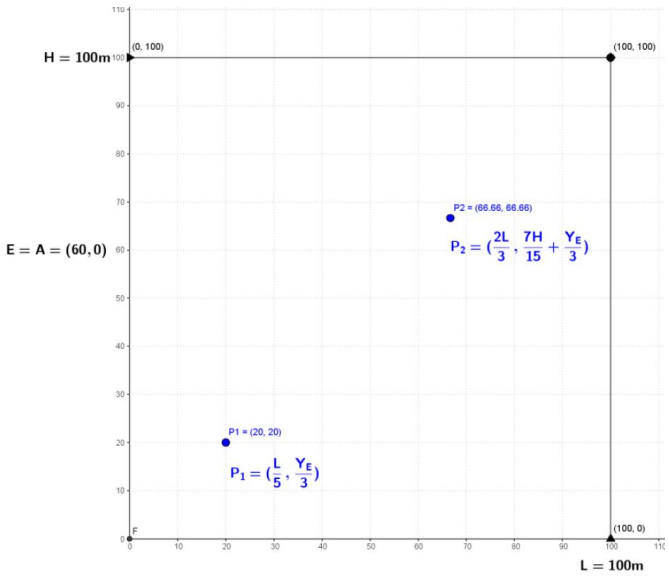


Figura 6.35.- *Norma FEM 9.851 – Caso 4.3 – puntos P1 y P2 (Coord. e/s > 50 m)*

En el siguiente dibujo se muestra la trayectoria de un ciclo medio simple:

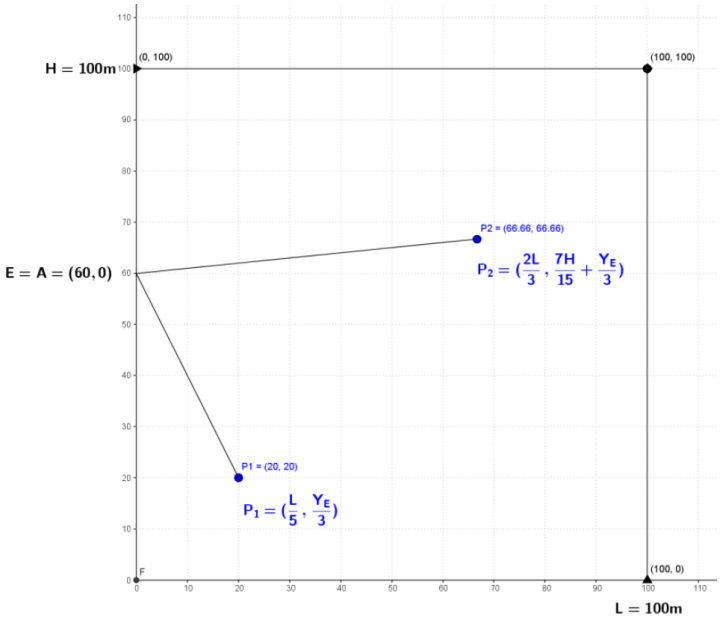


Figura 6.36.- *Norma FEM 9.851 – Caso 4.3 – Trayectoria ciclo medio simple- Coord. e/s > 50 m (Almacén Dlm. 100m x 100m)*

En el siguiente dibujo se muestra la trayectoria de un ciclo medio combinado:

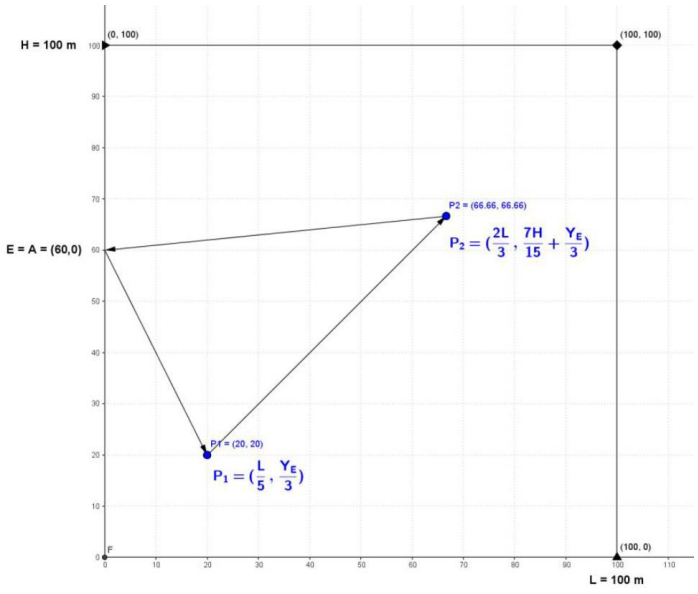


Figura 6.37.- *Norma FEM 9.851 – Caso 4.3 – Trayectoria ciclo medio combinado- coord. e/s > 50 m (Almacén Dlm. 100m x 100m)*

Las siguientes tablas muestran las validaciones del modelo de transelevador sobre el cao 4.3. de la normativa FEM 9.851:

CASO 1.3.- E=A=E(0,Ye)=A(0,Ye)=E(0,12600)=A(0,12600)

Fondo: Simple **Cargas:** 1 **Movimiento:** Almacenaje **Ciclo:** Simple

Puntos en estudio

P1 (x,y) P1 (Col18,Fila10) P1 (18900, 21100)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:53
3	Traslo empieza a sacar extractores	1:01
4	Traslo termina de sacar extractores	1:04
5	Traslo tiene encima el pallet	1:06
6	Traslo inicia movimiento	1:07
7	Traslo llega frente a su posicion de descarga	1:20
8	Traslo descarga pallet / saca extractores	1:28
9	Traslo finaliza descarga	1:33
10	Traslo vuelve hacia ME	1:34
11	Traslo llega al origen (ME) - sin carga	1:47

P2 (x,y) P2 (Col59,Fila5) P2 (63100,9260)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:54
3	Traslo empieza a sacar extractores	1:02
4	Traslo termina de sacar extractores	1:06
5	Traslo tiene encima el pallet	1:10
6	Traslo inicia movimiento	1:11
7	Traslo llega frente a su posicion de descarga	1:37
8	Traslo descarga pallet / saca extractores	1:45
9	Traslo finaliza descarga	1:54
10	Traslo vuelve hacia ME	1:55
11	Traslo llega al origen (ME) - sin carga	2:21

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:13	0:13	✓
E-P2	0:26	0:26	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
P1-E	0:13	0:13	✓
P2-E	0:26	0:26	

Fondo: Simple **Cargas:** 1 **Movimiento:** Desalmacenaje **Ciclo:** Simple

Puntos en estudio

P1 (x,y) P1 (Col18,Fila10) P1 (18900, 21100)

Nº movimiento	Movimiento	mm:ss
1	Traslo inicia movimiento	0:02
2	Traslo posicionado frente a ubicación	0:23
3	Traslo empieza a sacar extractores	0:31
4	Traslo con pallet encima	0:36
5	Traslo inicia movimiento hacia MS	0:37
6	Traslo llega frente a MS	0:50
7	Traslo inicia descarga	0:58
8	Traslo finaliza descarga	1:03
9		
10		
11		

P2 (x,y) P2 (Col59,Fila5) P2 (63100,9260)

Nº movimiento	Movimiento	mm:ss
1	Traslo inicia movimiento	0:02
2	Traslo posicionado frente a ubicación	0:28
3	Traslo empieza a sacar extractores	0:36
4	Traslo con pallet encima	0:45
5	Traslo inicia movimiento hacia MS	0:46
6	Traslo llega frente a MS	1:12
7	Traslo inicia descarga	1:20
8	Traslo finaliza descarga	1:29
9		
10		
11		

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:13	0:13	✓
P1-P2	0:26	0:26	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
P2-A	0:13	0:13	✓
A-E	0:26	0:26	

Fondo: Simple **Cargas:** 1 **Movimiento:** Des/Almacena **Ciclo:** Combinado

Puntos en estudio

P1 (x,y) P1 (Col18,Fila10) P1 (18900, 21100)
P2 (x,y) P2 (Col59,Fila5) P2 (63100,9260)

Nº movimiento	Movimiento	mm:ss
1	Pallet llega a ME	0:38
2	Traslo posicionado frente a ME con pallet	0:53
3	Traslo empieza a sacar extractores	1:01
4	Traslo termina de sacar extractores	1:03
5	Traslo tiene encima el pallet	1:06
6	Traslo inicia movimiento	1:07
7	Traslo llega frente a su posicion de descarga	1:20
8	Traslo descarga pallet / saca extractores	1:28
9	Traslo finaliza descarga	1:32
10	Traslo va hacia P2	1:33
11	Traslo posicionado frente a P2	1:53
12	Traslo empieza a sacar extractores	2:01
13	Traslo con pallet encima	2:06
14	Traslo inicia movimiento hacia MS	2:07
15	Traslo llega frente a MS	2:33
16	Traslo incia descarga	2:31
17	Traslo finaliza descarga	2:46

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
E-P1	0:26	0:26	✓
P1-P2	0:00	0:00	

Origen - Destino	Valor Teórico (mm:ss)	Valor Simulación (mm:ss)	Validación
P2-A	0:26	0:26	✓
A-E	0:00	0:00	

Tabla 6.7.- *Verificación de ciclos en norma FEM 9.851 (06.2003)- caso 4.3*

6.5. - CONCLUSIONES Y POSIBLES AMPLIACIONES

Los resultados obtenidos del desarrollo e implementación del modelo de máquina transelevador creada para la plataforma AWS, mediante Designer IV, han sido los siguientes:

1. Se ha realizado el programa de control del transelevador de manera simplificada, teniendo en cuenta los sensores, variadores, parámetros, variables y modos de funcionamiento necesarios para mantener la seguridad y control de la máquina dentro de la simulación de una instalación.

2. Se ha realizado, programado e implementado el modelo de simulación del transelevador, dejando un modelo operativo que funciona y trabaja correctamente y que puede ser aplicable a cualquier instalación futura, simplemente configurando los parámetros oportunos para cada almacén en el que se aplique.

No obstante, este desarrollo, presenta algunas limitaciones, las cuales vienen dadas por la plataforma sobre la que ha sido creado el modelo de simulación.

La mejora continua del software, la creación de nuevas máquinas para un repositorio y la corrección constante de bugs que han ido surgiendo a lo largo del periodo de ejecución del proyecto, en algunas ocasiones, han ralentizado trabajo a realizar, pero la existencia de un gran equipo de trabajo con altas cualificaciones y disposición a ayudar y colaborar para un buen trabajo de equipo, como el que hay en Meclaux ha hecho posible que haya logrado el cumplimiento de los objetivos que se esperaban para este trabajo fin de master.

Se dejan planteadas mejoras y posibles ampliaciones para futuros trabajos del sistema de recogida y depósito de contenedores de un almacén automático (Transelevador):

1. El sistema de control de la máquina transelevador puede hacerse más complejo si se realiza la programación / simulación de un transelevador con más de un extractor (doble carga o cuádruple carga). Este tipo de transelevadores llevan adicionalmente toda la lógica de optimización de movimiento en función del extractor que complica sustancialmente el programa de control.

2. En desarrollos futuros que este interactúe con un sistema de gestión operativo atendiendo a sus órdenes y respondiendo al igual que lo haría el escenario real.

3. Posibilidad de interacción del sistema emulado con el sistema real. (SCADA)

4. Análisis de las posibles variaciones y puntos sensibles de cara a la optimización del proceso productivo

5. Algunas instalaciones tienen más pasillos que transelevadores, por lo que la ejecución de una orden de movimiento puede llevar implícito un cambio de pasillo además del movimiento en los 3 ejes X, Y y Z.

Universidad de Oviedo

Trabajo Fin de Máster realizado por

Victoria Casares García

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

ANEXOS

Modelado de Transelevador
Automatic Warehouse Studio (AWS)

Julio de 2018

ÍNDICE ANEXOS

Universidad de Oviedo

Trabajo Fin de Máster realizado por

Victoria Casares García

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

ANEXO I: Logística

Modelado de Transelevador
Automatic Warehouse Studio (AWS)

Julio de 2018

ÍNDICE ANEXOI: Logística

1. ANEXO I: LOGÍSTICA

En los siguientes apartados se definirán y explicarán terminología y conceptos referentes al ámbito de la logística, los cuales se consideran imprescindibles para poder entender con claridad el proyecto desarrollado.

1.1. - LOGÍSTICA. DEFINICIONES Y TERMINOLOGÍA

La logística es definida por la Real Academia Española (RAE), como el «*conjunto de medios y métodos necesarios para llevar a cabo la organización de una empresa, o de un servicio, especialmente de distribución*». En el ámbito empresarial existen múltiples definiciones del término logística, que ha evolucionado desde la logística militar hasta el concepto contemporáneo del arte y la técnica que se ocupa de la organización de los flujos de mercancías, energía e información.

La logística es fundamental para el comercio. Las actividades logísticas conforman un sistema que es el enlace entre la producción y los mercados que están separados por el tiempo y la distancia.

Se dice, por lo tanto, que la logística es el puente o el nexo entre la producción y el mercado. La distancia física y el tiempo separan a la actividad productiva del punto de venta: la logística se encarga de unir producción y mercado a través de sus técnicas.

En las empresas, la logística implica tareas de planificación y gestión de recursos. Su función es implementar y controlar con eficiencia los materiales y los productos, desde el punto de origen hasta el consumo, con la intención de satisfacer las necesidades del consumidor al menor coste posible

1.2.- ALMACENES. CONCEPTOS GENERALES

Los almacenes son centros que están estructurados y planificados para llevar a cabo funciones de almacenamiento tales como: conservación, control y expedición de mercancías y productos, recepción, custodia, etc.

El almacén es una instalación que, junto con los equipos de almacenaje, de manipulación, medios humanos y de gestión, nos permite regular las diferencias entre los flujos de entrada de mercancía (la que se recibe de proveedores, centros de fabricación, etc.) y los de salida (aquella mercancía que se envía a la producción, la venta, etc.). Estos flujos suelen no estar coordinados y esa es una de las razones por las que se precisa definir una óptima logística de almacenamiento.

1.3.- TIPOS DE ALMACENES

En ocasiones, la actividad económica de una empresa puede exigir de uno o varios tipos de almacén: de materias primas, de productos semielaborados, de productos terminados,

etc. Todos ellos han de estar ubicados en función de las necesidades específicas de su funcionamiento y de acuerdo con las restricciones o las posibilidades de cada localización y su entorno.

La mejor manera de clasificar los distintos tipos de almacenes que se pueden dar en la actualidad es agrupándolos según sus características comunes:

- **Según la naturaleza del producto** se pueden encontrar almacenes especializados en bobinas, productos inflamables, perfiles, pequeño material, recambios, productos perecederos e incluso almacenes que son de uso general, entre otras posibilidades.
- **El edificio** también puede ser un criterio de clasificación y así se habla de almacenes al aire libre, naves, sótanos, almacenes de gran altura o depósitos, cámaras frigoríficas, almacenes autoportantes (las estanterías conforman el armazón del propio edificio), etc.
- **Dependiendo del flujo de materiales**, las instalaciones pueden agruparse en aquellas destinadas a materias primas, componentes o productos semielaborados, productos acabados, almacenes intermedios, de depósito, para distribución, etc.
- **En cuanto a su localización**, se habla de almacenes centrales, regionales y de tránsito.
- **En cuanto a su mecanización**, pueden ser manuales, convencionales o automáticos

1.4. - ACTIVIDADES QUE SE LLEVAN A CABO EN UN ALMACÉN

Las labores desarrolladas en un almacén principalmente son:

1. Recepción de mercancía
2. Su verificación
3. Transporte interno (entre distintas zonas del almacén)
4. Almacenaje y custodia
5. Preparación de pedidos y la consolidación de cargas
6. Expedición de mercancía
7. Gestión e información relativa a stocks, flujos, demanda, etc

1.5. - ELEMENTOS QUE INTERVIENEN EN UN ALMACÉN

Son varios los factores que hay que tener en cuenta a la hora de estudiar una instalación. Principalmente, se ha de considerar el **producto** que se va a almacenar, el **flujo de materiales** o mercancías, el **espacio disponible** para albergarlos, los **equipos de almacenaje** –tales como las estanterías y los equipos de manutención–, el **factor humano** (el personal), así como el **sistema de gestión** y la política de la empresa.

A partir de todos estos elementos se han de recopilar una serie de datos que influirán en diversos aspectos de la instalación y que se tomarán en cuenta a la hora de desarrollarla.

1.6. - PARTES DE LAS QUE CONSTA UN ALMACÉN

El almacén más simple suele constar de puertas de acceso, una zona libre para maniobra y verificación, una zona de almacenaje para ubicar la mercancía, una oficina de control para la gestión de la planta y unos aseos y vestuarios para el personal.

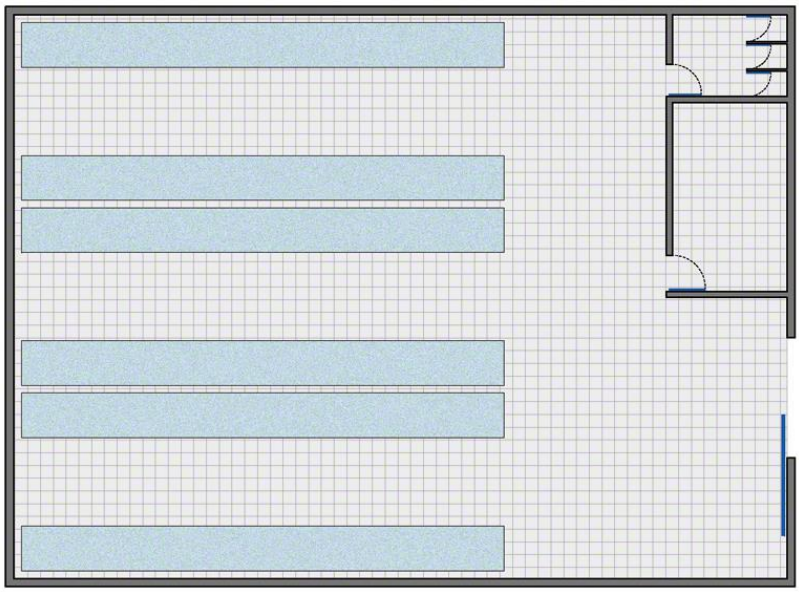


Figura 1.1.- *Almacén con la configuración más simple: zona de almacenaje, zona de gestión y vestuarios y aseos para el personal*

A partir de la configuración más simple, se pueden ir añadiendo otras zonas, tales como la de recepción, la de embalaje y consolidación, de expedición, de recarga de las baterías para las carretillas, así como muelles de carga.

A su vez, el almacén puede estar dividido en sectores en función del producto que se maneje o según la operativa de trabajo. En la siguiente figura se ilustra un ejemplo de este tipo de organización:

1. Edificio de oficinas y servicios.

2. Muelles de carga y descarga.

3. Recepción y verificación.

4. Expediciones.

5. Almacén de alta rotación o producto voluminoso.

6. Picking de alta rotación sobre paletas.

7. Almacén de productos irregulares.

8. Almacén de componentes de media rotación.

9. Almacén de componentes de alta rotación.

10. Almacén de componentes de baja rotación.

11. Almacén de productos de alto valor.

12. Zona de embalaje y consolidación

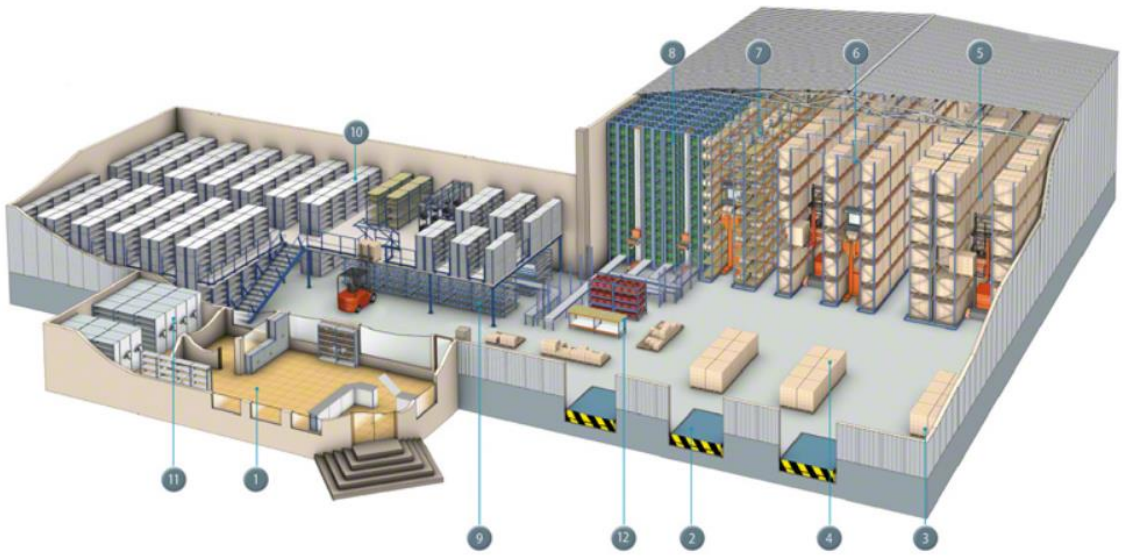


Figura 1.2.- *Mapa de las diferentes operativas que pueden darse en un almacén*

Los espacios asignados a cada zona han de ser los adecuados en función de las dimensiones del terreno o del edificio, la capacidad deseada, las operaciones que se

tengan que realizar, del personal y de los medios necesarios, el flujo de materiales y las posibilidades de crecimiento futuro.

En cualquier caso, la adecuación del proyecto y el diseño de las áreas dentro de la instalación vendrán determinados por un exhaustivo estudio de las necesidades de la empresa –a través de las preguntas previamente planteadas–, así como de la experiencia en la implementación de soluciones logísticas y de almacenaje que tenga el proveedor.

Todo el edificio –su forma, el contenido y los accesos– debe estar en consonancia con las necesidades específicas del cliente y, además, deben preveerse las posibilidades de crecimiento. Un almacén demasiado ajustado y sin capacidad de expansión en el futuro es un error, salvo si se trata de una instalación temporal o de crecimiento estático.

1.7. - GESTIÓN DE ALMACENES

La buena organización y gestión de almacenes permite ofrecer el mejor servicio. Hace que se pueda disponer del stock necesario, tener una alta ocupación del almacén, emplear el menor tiempo en las operaciones internas como el transporte o el picking y controlar el stock, así como optimizar las ubicaciones y los flujos, entre otras cuestiones.

Se debe conseguir que el almacén sea inteligente, es decir, que esté gestionado de manera lógica y eficiente para obtener el mayor rendimiento posible. Hoy en día no se concibe ninguna instalación profesional que no disponga de un software de gestión de almacenes (también conocido por sus siglas, SGA), que garantice estas premisas.

La elección del SGA adecuado es fundamental y ha de permitir realizar, como mínimo y de forma sencilla e intuitiva, las funciones básicas del almacén gestionado.

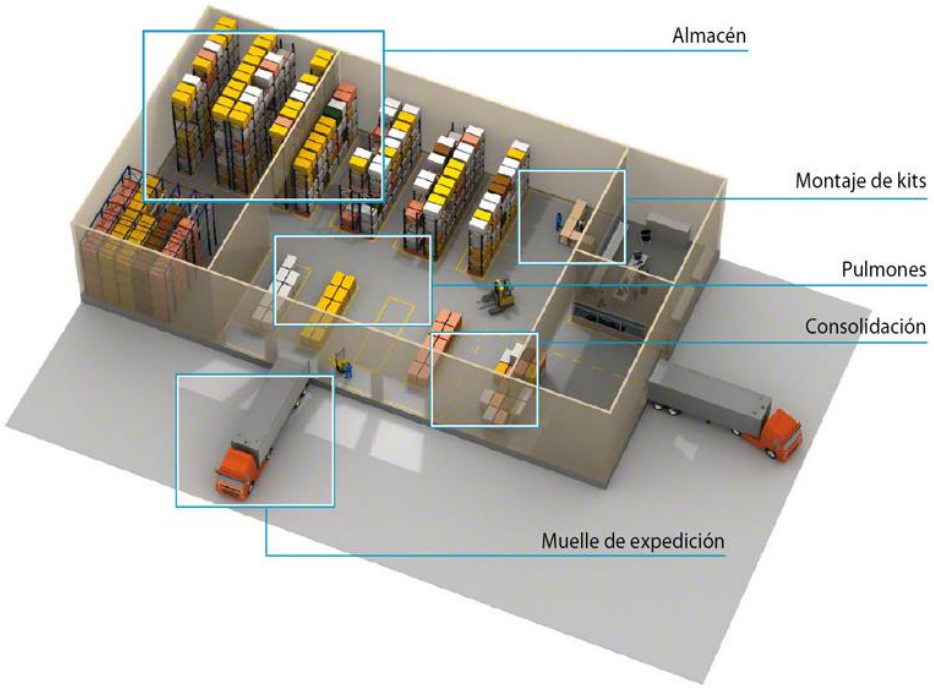


Figura 1.3.- *El SGA controla las diversas operativas de un almacén y colabora así en su óptima gestión.*

1.7.1.- ¿QUIÉN GESTIONA EL ALMACÉN?

La gran mayoría de los almacenes están administrados directamente por la propia empresa (el fabricante o el comercializador) dueña de la instalación. Todas las funciones del sistema pueden estar integradas con el resto de la gestión de la empresa o bien ser anexas a esta y estar coordinadas con ella. El SGA solo se aplica a las funciones propias del almacén y mediante una colección de interfaces establece una comunicación en tiempo real con el sistema de gestión general o central de la empresa.

Por otra parte, cada vez con más frecuencia, las firmas externalizan los servicios logísticos y de almacenaje mediante operadores logísticos, quienes, además de almacenar, pueden ofrecer un servicio global y preparar picking, montar componentes, encargarse del transporte, etc.

Por lo dicho, en un mismo almacén de un operador logístico pueden convivir mezcladas mercancías de varios clientes, o propietarios, que han de ser gestionadas correctamente. En estos casos, el SGA ha de poder ser utilizado con el principio de multipropietario, lo que permite gestionar el stock de terceros.

1.7.2.- CÓMO SE GESTIONA LA MERCANCÍA Y SU UBICACIÓN

El diseño de la gestión del almacén se ha de realizar de acuerdo con un análisis funcional previamente elaborado. Este análisis funcional debe seguir los pasos de los flujos de materiales, así como reflejar las características y tipología de la instalación y sus componentes.

En la ubicación de la mercancía dentro del almacén se ha de tener en cuenta la clasificación de productos A-B-C y colocar los A en los puntos más cercanos y accesibles, dejando los B y los C en un segundo plano según su prioridad.

El criterio de ubicación que se emplee en las estanterías condicionará la forma de trabajar y la capacidad efectiva. Hay tres modos de determinar la posición de cada unidad de carga:

1. Con la **ubicación específica o fija** en la que a cada referencia se le asigna una posición o un número de ubicaciones determinado de antemano. La gran ventaja de este método es la facilidad para localizar las referencias. Las personas que trabajan en el almacén saben dónde está cada una de ellas sin tener que recurrir a ayudas informáticas. La gran desventaja al usar este criterio es la pérdida de capacidad efectiva, que es muy inferior a la física (número de ubicaciones). Solo se debe emplear en almacenes muy pequeños y no necesita un sistema de gestión.
2. Con la **ubicación aleatoria** a la que también se conoce como caótica, libre o variada, la mercancía se ubica en cualquier hueco vacío disponible, siguiendo una lógica previamente establecida y parametrizada (programada) en el SGA. Generalmente se tiene en cuenta la clasificación A-B-C. El sistema, que tiene todos los datos introducidos (incluyendo los huecos vacíos) indica al operario

dónde se ha de colocar la mercancía o dónde se halla esta. Además de la perfecta gestión que supone, la ubicación caótica permite que la capacidad efectiva del almacén se acerque mucho a la capacidad física, pudiendo superar el 92% de ésta.

3. Sin embargo, la **ubicación mixta o semialeatoria** es la que se usa con más frecuencia y combina el sistema específico y el aleatorio, asignando cada uno de ellos en función del tipo de producto o de la operación que se deba realizar. Así, la ubicación específica se utiliza para productos de alto consumo, que generalmente están cerca de los muelles o zonas de picking, mientras que la aleatoria se deja para el resto de los productos y zonas de reserva.

La gestión de los huecos, principalmente en los sistemas específicos, sigue criterios de productividad mediante la optimización de los recorridos, en especial, los de preparación de pedidos.

Además de elegir el criterio adecuado es imprescindible contar con un SGA apropiado para cada caso

1.8. - SGA O SOFTWARE DE GESTIÓN DE ALMACENES

Un SGA o Software de Gestión de Almacenes es una herramienta que se utiliza para controlar, coordinar y optimizar los movimientos, procesos y operativas propios de un almacén.

Aunque el análisis de los flujos y la determinación del tipo de ubicación que se aplicará en el almacén condicionará el ajuste y parametrización específica de su software de gestión, estas son las principales funciones básicas que es imprescindible que realice cualquier SGA:

- Gestión de entradas
- Gestión de la ubicación de las unidades de carga
- Gestión de control de stock
- Gestión del control de las salidas

A continuación, se hace un repaso más en profundidad de todas estas operaciones que deberían estar cubiertas por el SGA.

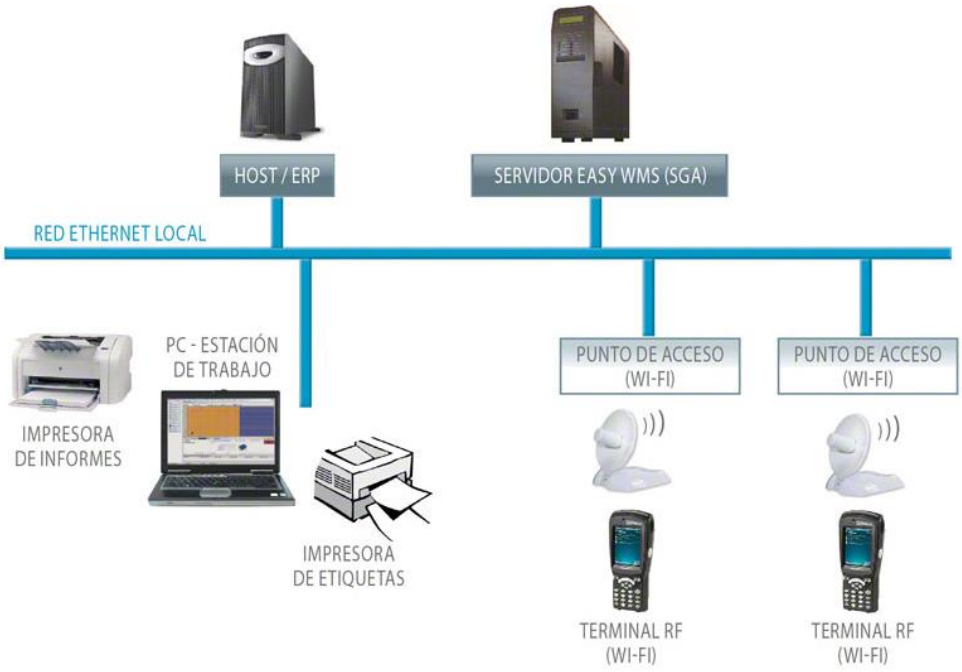


Figura 1.4.- *Ejemplo de arquitectura de un software de gestión de almacenes*

1.8.1.- FUNCIONES DE ENTRADA

Dentro de este grupo destacan tres operaciones gestionables a través del SGA, como son la recepción, la captura de datos logísticos y el etiquetado de los contenedores y la mercancía.

Recepciones

En algunos casos, se reciben los productos sin paletizar y estos son consolidados en distintos contenedores, a la vez que se registran sus características y atributos logísticos para, posteriormente, ser ubicados dentro del almacén.

Otra forma de recibir la mercancía es mediante contenedores en los que la mercancía llega paletizada al almacén, con lo que no se requiere su consolidación y solo es necesario realizar la validación de los atributos logísticos y las cantidades de la mercancía recibida para, después, proceder a su ubicación.

El tercer tipo de recepción básica que se puede dar en un almacén es la de artículos procedentes de devolución. Si bien es similar a cualquiera de las anteriores recepciones comentadas, presenta características particulares como puede ser la aplicación de estados y bloqueos de mercancía en el proceso de entrada a la instalación. Estos estados y bloqueos pueden indicar, por ejemplo, la necesidad de pasar por el control de calidad, quedar pendiente de revisión, etc. Posteriormente habrá que realizar la ubicación dentro del almacén en zonas específicas o determinadas para este tipo de mercancía.

Captura de datos logísticos

Estos datos incluyen distintas informaciones como por ejemplo el lote al que pertenece la carga, su caducidad, su peso, su temperatura, el número de serie, etc. La captura de estos datos en el momento de realizar el proceso de recepción proporciona al stock una trazabilidad.

Los atributos logísticos, como la identificación del lote o el número de serie, permiten conocer a posteriori qué mercancía se ha servido exactamente a cada cliente en concreto.

Etiquetado de contenedores y mercancía

La tercera función en las entradas se complementa con la anterior. Un SGA debe poder generar etiquetas de código de barras para todos los contenedores y mercancías que se almacenen. Gracias a ello, todos los procesos y operativas que se lleven a cabo dentro del almacén se validarán, con exactitud, mediante la lectura de estos códigos, lo que elimina los posibles errores y confusiones que se puedan generar en la manipulación de la mercancía

Figura 1.5.- *La lectura de código de barras agiliza los procesos de identificación*

También pueden ser etiquetados mediante los códigos de barras todos y cada uno de los artículos que se recepcionen de manera que, más tarde, se realicen los procesos de salida con una mayor agilidad y eficiencia.

La documentación de las recepciones permite, además, obtener informes en los que quedan reflejadas las diferencias entre la mercancía prevista y la que realmente se ha recibido, así como otras cuestiones, como pueden ser el cumplimiento de las franjas horarias de recepción.

El cometido del SGA no termina en estas actividades, ya que mediante una interfaz se encarga de transmitir al ERP de la empresa el cierre de recepción, en el que se especifica con exactitud cuántas unidades de las distintas referencias han entrado al almacén, de modo que el ERP pueda llevar a cabo las gestiones administrativas con los proveedores.

1.8.2.- FUNCIONES DE UBICACIÓN

Dentro de este tipo de operaciones, destacan tres gestiones: la gestión de la ubicación mediante reglas y estrategias, el cross-docking y la gestión de la reposición y la consolidación.

La gestión de la ubicación mediante reglas y estrategias

Es la encargada de elegir la localización idónea en el almacén para una mercancía concreta. Para ello, el software tiene en cuenta parámetros como la rotación de artículos (A, B o C), los tipos de contenedores empleados para la consolidación de la mercancía, las familias o tipos de productos que se manejan, la peligrosidad o incompatibilidad de unos productos u otros a fin de que no queden juntos o cercanos, las presentaciones de los artículos, el volumen de los mismos, etc.

'Cross-docking'

Por su parte, mediante el cross-docking se ahorran movimientos con la carga. Si llega al almacén una mercancía de la que se debe extraer producto que formará parte de un pedido que está activo y en el que falta stock, se procede a alojarla en la zona de preparación de pedidos, directamente desde el área de recepción. Una vez que la mercancía necesaria se ha preparado para salida, se procede a la ubicación dentro del almacén de la que sobre.

La gestión de la reposición y la consolidación

También tiene como objetivo el ahorro de movimientos de la mercancía dentro de la instalación. Se aplica, al igual que en el cross-docking, antes de proceder a la ubicación definitiva de la carga. Esta función se lleva a cabo en centros en los que se han establecido posiciones de picking; si en éstas queda poco producto, se ha reponer con la mercancía que acaba de llegar, de manera que haya stock suficiente para la realización de los pedidos. Posteriormente, se procede a la ubicación de la carga que sobre de esta operación.

1.8.3.- FUNCIONES DE CONTROL DE STOCK

En la gestión de la mercancía almacenada, el SGA debe ser capaz de proporcionar información completa y útil sobre el stock. Para ello, se recurre a varias funciones.

- La más intuitiva es la visualización del mapa del almacén, mediante el que se accede, en una pantalla, a una representación gráfica de la instalación en la que se detallan cada una de las ubicaciones y su composición, tanto en lo que respecta al contenedor, como a la propia mercancía.
- La gestión de ubicaciones es otra de las herramientas con las que debe contar un SGA. Con ella se puede obtener y editar la información sobre las posiciones como el tipo de ubicación, los bloqueos que tenga aplicados, sus dimensiones, sus características, las zonas de almacenaje a las que pertenece, etc.
- De la misma manera se debe poder realizar la gestión de estados del stock con el fin de consultar y modificar los datos relativos a las cuarentenas, roturas, pérdidas, bloqueos, reservas, etc.
- El SGA también puede realizar, por sí mismo, operaciones que ayuden en la gestión del stock. Una de ellas es el cálculo de rotación de los artículos. En función de los movimientos realizados durante un periodo de tiempo que se le indique, el sistema puede determinar e informar de cuál debe ser la rotación ideal de un artículo y compararla con la que este tiene asociada en el maestro de artículos. Con esta herramienta se puede recalcular la rotación A-B-C y cambiar la que tiene asignada el artículo en caso de que se considere más eficiente o conveniente.
- Hay una última función indispensable relativa al stock, como es el recuento y el inventario. Con estas tareas programables se puede llevar a cabo desde un inventario global de todo el almacén, a un inventario específico de un artículo, una ubicación o una zona en concreto. En el caso de que se encuentren diferencias de stock se informa de ellas automáticamente al ERP.

1.8.4.- FUNCIONES DE SALIDA

Además de administrar la entrada y ubicación de la mercancía, el sistema de gestión se ocupa también se debe ocupar del control de las salidas de los productos.

- Las funciones principales que se llevan a cabo en esta fase del almacenaje comienzan con la gestión de la preparación de la carga que tiene que salir de la instalación. Esta actividad cubre las agrupaciones de pedidos y las asignaciones de los mismos, entre otras cuestiones. Esto permite el control sobre cómo se ejecutan los pedidos y quién se encarga de ello: la asignación de los muelles de expedición, los operarios que realizan la preparación, la manera de realizar el agrupamiento de los pedidos y la franja horaria en que se produce, etc.
- Dentro de la preparación, el SGA puede gestionar a un nivel muy detallado las operaciones que se tienen que llevar a cabo, como es el caso de los procesos de picking. El sistema se encargará de definir y guiar los recorridos del personal asignado a esta tarea, así como la presentación de los artículos. Una de las ventajas más importantes derivadas de la gestión del picking por parte del SGA es que el sistema es capaz de optimizar el proceso a fin de que éste se ejecute en el más breve tiempo posible y con el menor número de movimientos a la vez que se

respetan los parámetros especificados para el pedido que han sido enviados por el ERP.

- Otra parte de la preparación de la mercancía es la relativa al etiquetado de expediciones, mediante la que se identifican los bultos de pedido, y a través los mismos procedimientos del etiquetado de entrada.
- Igualmente, en la salida se procede a la documentación de las expediciones, lo que facilita la generación de documentos tales como el packing list (listado de los artículos que componen el pedido), la documentación para el transportista y los informes que reflejan las discrepancias que hayan encontrado.
- Tras estas operaciones, el SGA administra el proceso de carga de las expediciones en los vehículos. Gracias a esta función, se controla la calidad del envío, de tal forma que se evitan errores tales como, por ejemplo, el despacho de un material a un cliente que no lo ha solicitado.
- Para terminar, el SGA puede gestionar hasta la última fase de las operaciones de salida, como es la comunicación al ERP del cierre de expedición, lo que se realiza mediante una interfaz entre los dos sistemas. Con esta función, se informa al gestor de recursos de la empresa de cuántas unidades y de qué referencias ha constado la expedición, así como qué bultos se han despachado en cada una de las órdenes de salida que se han ejecutado. Con estos datos, el ERP puede gestionar los procesos administrativos con los clientes.

1.8.5.- OTRAS FUNCIONES

Dependiendo del tipo de almacén y empresa, hay otras tres funciones que pueden ser esenciales para integrar las instalaciones en el resto de la empresa o del centro logístico:

1. La primera es la gestión de los flujos de mercancía (entradas y salidas) a las líneas de producción. Gracias a esta característica, se obtiene un flujo óptimo (tanto en salidas como en entradas) a las líneas de producción o fabricación, lo que agiliza los procesos internos.
2. La segunda función tiene que ver con la administración de varios almacenes con un mismo SGA, lo que se denomina gestión de multialmacenes. Con esta característica, una misma empresa puede gestionar todos sus almacenes de manera única y global y así se optimizan los recursos (así como los sistemas informáticos) y se facilitan los procesos de traspasos de mercancía entre las instalaciones.
3. En tercer lugar, a un nivel superior al anterior se encuentra la gestión multiorganización, por la que, como su nombre indica, distintas organizaciones pueden ser administradas por el mismo SGA

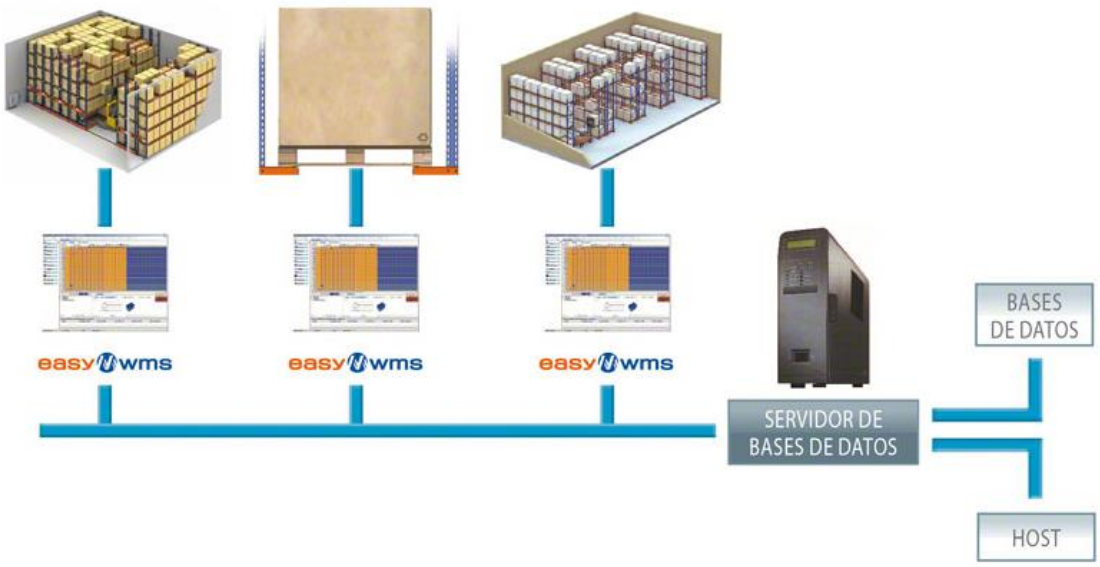


Figura 1.6.- *Un SGA puede llegar a gestionar varios almacenes de manera integrada y global*

Por último, es necesario comentar que en algunos proyectos es preciso un desarrollo específico adaptado al cliente según sus necesidades. Ciertas operativas que llevan a cabo algunas empresas pueden no estar recogidas, a priori, en el software, por lo que es necesario un desarrollo específico tomando como base un SGA ya creado. En este sentido, el sistema de gestión tiene que ser lo suficientemente versátil y abierto para que se puedan programar estas características personalizadas.

1.9. - LOS ALMACENES AUTOMATIZADOS.

Los almacenes automáticos son estructuras, generalmente de gran altura, donde los elementos de almacenamiento y los elementos de manutención van integrados y controlados por un sistema informático, es decir, está formado por estanterías, transelevadores, métodos de transporte automatizados y el sistema de gestión y control (software)que regula todos los movimientos.

Los almacenes automatizados permiten gestionar y optimizar los procesos derivados del almacenaje, preparación y expedición de mercancías.

Los almacenes automáticos se dividen en dos tipos:

1. Almacenes automáticos para pallets: Soluciones automáticas para las operaciones de entrada, almacenaje y salida de los productos paletizados. Permiten mejorar la capacidad, la rapidez y la seguridad del proceso de almacenamiento.
2. Almacenes automáticos para cajas: Sistemas de almacenaje automático para cajas o bandejas que integran en un solo producto las estanterías, la maquinaria y el

software de gestión del almacén. Mejora de la eficiencia gracias al concepto "producto a hombre", evitando los desplazamientos del operario.

Los almacenes automáticos de paletas se recomiendan para empresas con una alta rotación de artículos, muy amplia gama de referencias, de unidades homogéneas de volumen de paleta o superior donde la superficie disponible exija grandes alturas de almacenamiento. Los almacenes Miniload se recomiendan para artículos de poco volumen y elevada cantidad de referencias. Con un muy alto movimiento de artículos

El funcionamiento de un almacén automático, bien sea de pallets o de cajas, es el mismo, el sistema informático ubica los productos en las estanterías mediante el transelevador. Cuando las mercancías son requeridas el sistema informático lanza la orden de recogida.

1.9.1.- CARACTERÍSTICAS GENERALES DE LOS ALMACENES AUTOMÁTICOS

Las técnicas de la logística moderna aportan sistema que permiten gestionar, optimizar y agilizar los procesos derivados del almacenaje, preparación y expedición de todo tipo de mercancías.

Los sistemas de distribución se han convertido en un elemento estratégico dentro de la gestión de la cadena de suministro y, por lo tanto, de creación de valor en la cadena empresarial.

Asimismo, la incorporación de sistemas automáticos en los procesos de manutención facilita a las empresas la diferenciación de su oferta de productos y servicios, gracias a la reducción de costes y al aumento de las prestaciones de la cadena logística.

Estas aplicaciones tecnológicas van adquiriendo cada vez más protagonismo, incorporándose a los sistemas y formas de gestión de los almacenes tradicionales.

Ventajas:

- Productividad y alta disponibilidad
- Disminución de los costes de mantenimiento
- Economía de costes laborales
- Total seguridad del personal
- Absoluta seguridad de la carga
- Inventario permanente
- Máximo aprovechamiento del espacio

Inconvenientes:

- Inversión inicial muy alta
- Necesidad de un sistema informático muy robusto

1.9.2.- TIPOS DE ESTACIONES Y UBICACIONES EN UN ALMACÉN AUTOMÁTICO

- **Estación de Entrada (ES)**: Estación que representa el punto de entrada al almacén automático, donde los contenedores son depositados y desde la cual son movidos al PIE.
- **Apilador / Des-Apilador (DTK):** La estación APL gestiona los contenedores vacíos, como elementos susceptibles de apilar y des-apilar existentes en el maestro de artículos
- **Puesto de Identificación (EIP):** Estación donde se realiza el control dimensional e identificativo del contenedor en una instalación automática. El control dimensional es realizado por el sistema de control. Con la información enviada por el sistema de control, EasyWMS® envía el contenedor al puesto de rechazo o al almacén.
- **Reacondicionamiento (REAC):** Estación en la que se puede depositar un contenedor al que se ha detectado algún tipo de tara, bien sea dimensional (por ejemplo, un gálibo lateral), o identificativa (por ejemplo, peso). Estas estaciones siempre van asociadas a una estación PIE.
- **Estación de Rechazo (REJ):** Estación donde son rechazados los contenedores cuando se decide que no son válidos para ser almacenados.
- **Mesa de ubicación (LC):** Busca ubicación al contenedor. Cuando a la MU llega un contenedor que aún no tiene destino o que tiene como destino la propia MU, en esta ubicación se le busca ubicación. La mesa de ubicación cobra sentido en las instalaciones en las que media gran distancia entre las estaciones PIE y Almacén. Se coloca en un punto cercano al Almacén para afinar la decisión de ubicación.
- **Estación de Tránsito (TS):** Estación intermedia que implica la toma de decisiones sobre los contenedores que pasan por ella. Es personalizada, ya que no tiene funciones standard.
- **Control de Mesa de Entradas (ICS):** Es una estación intermedia. Se encuentra en las intersecciones con la entrada de un pasillo AS/RS. Se comprueba si la ruta elegida está en una condición adecuada para llevar el contenedor o si se debe de buscar una ubicación mejor en otro pasillo.
- **Mesa de Entrada(IC):** Esta estación el punto de enlace con el Transelevador. Su función es permitir una última optimización antes de entregar el contenedor a la máquina. Esta estación es necesaria en todo caso y permite realizar cambios en función de las características del Transelevador (cantidad de extractores y configuración de los mismos) y de los transportadores de entrega al Transelevador. Se encarga de la optimización de ubicación en función de las características de la máquina.
- **Transelevador (STC):** Máquina automática encargada de los procedimientos de ubicación y extracción de contenedores en la estantería mediante un sistema de

movimiento en tres ejes (traslación, elevación y profundidad). Se emplea en pasillos automáticos no transitables.

- **Almacén (AIS):** Estación que representa las ubicaciones de un pasillo de un almacén.
- **Mesa de Salida (OC):** Estación de descarga del transelevador.
- **Mesa de Control de Picking (PKE):** Puesto intermedio, situada a la entrada del pasillo de picking. Decide si el contenedor debe entrar a la zona de picking o seguir circulando por el anillo en función de la capacidad del puesto de picking o el orden en el que se deba de hacer el picking de los diferentes contenedores.
- **PK (Puesto de Picking):** Estación en la que los operarios realizan las operaciones de picking sobre aquellos contenedores asociados a una orden de expedición y de los que no se solicita la cantidad completa.
- **Mesa de Preparación (PS):** La mesa de preparación es donde el operario deposita los artículos mientras hace picking, esto es, mientras prepara órdenes de salida de un mismo artículo.
- **Lanzadera (SHU):** Esta estación sólo gestiona los movimientos. No es una estación de destino. Su propósito es mover los contenedores desde una entrada a varias posibles salidas.
- **Estación de Secuenciación (IPS):** Estación en la que se realizan las comprobaciones de los movimientos de salida para garantizar en todos los casos la secuenciación de los contenedores pertenecientes a una orden de salida.
- **Buffers de Gravedad (GFB):** Los carriles de soporte de los contenedores tienen una pequeña pendiente y están equipados con rodillos, de manera que los contenedores se mueven a través de ellos por gravedad.
- **Puesto de Salida (SC):** Estación en la que un contenedor es extraído del almacén automático hacia el exterior mediante una carretilla, transpaleta, etc.
- **Estación de Consolidación (CS):** Zona del Almacén donde se consolidan diferentes contenedores en una paleta para ser expedidos después.
- **Mostrador (DS):** Puesto de cara al público para venta directa en el almacén.
- **Estación de Montaje de KITs (KIT):** Zona del almacén donde un operario prepara los kits.
- **Advanced Shipping Notice (ASN):** Ubicación virtual utilizada para la notificación pre-avisada de descarga. Es la ubicación en la que se crean los contenedores cuando son preavisados por el ERP.
- **Lost&Found (L&F):** Ubicación virtual utilizada para la ubicación temporal de contenedores perdidos físicamente. El usuario puede enviar contenedores a Lost&Found manualmente y automáticamente desde un puesto de PK.
- **Movimiento (MOV):** Ubicación virtual utilizada para ubicar los contenedores en movimiento. Ejemplo: un contenedor que ha entrado por la estación EIP y se dirige hacia la estación Mesa de Ubicación. El propio recorrido en movimiento es la ubicación virtual MOV.

- **Pulmón (STG):** Utilizada para gestionar pulmones (generalmente ubicaciones de suelo) en los cuales el orden es indiferente. La capacidad puede configurarse ilimitada.
- **Muelle (DOCK):** Utilizada para gestionar muelles tanto de recepción como expedición.
- **Equipamiento (TRF):** Utilizada para gestionar equipamientos, incluyendo los que dispongan de TRF embarcado o de mano.

1.10. - TRANSELEVADORES

ASRS (Automated Storage and Retrieval System) o Transelevador, es un sistema automático que toma y entrega productos dentro de un almacén; ya sea en plantas de proceso, producción o ensamble, o bien en sus centros de distribución. Está constituido por grúas o robots de precisión, que pueden correr grandes distancias y grandes alturas rápidamente, para proveer alta densidad y alta capacidad de almacenamiento, también incluyen el software necesario para gestionar en tiempo real el almacén, por ello, también es conocido como almacén automático.

1.10.1.- QUÉ SON

Los transelevadores son máquinas creadas para el almacenamiento automático de materiales mediante movimientos mecánicos automatizados. Las entradas y salidas del material se ejecutan en un mismo movimiento (ciclo combinado) esto incrementa la productividad de las instalaciones al mismo tiempo que disminuye los recursos requeridos para su funcionamiento.

Para el traslado de cargas en el almacén, los transelevadores pueden realizar 3 tipos de movimientos:

- Longitudinal: sobre un rail a lo largo de un pasillo.
- Vertical: a lo largo de la columna del transelevador
- Transversal: o en profundidad, efectuado por los sistemas de extracción sobre la cuna de la máquina para la extracción o ubicación de la paleta

Principales familias de transelevadores:

- Monocolumna: recomendados para cargas de hasta 1500 kg
- Bicolumna: aconsejados para cargas de más de 1000 kg o grandes dimensiones

1.11. - COMPONENTES DEL TRANSELEVADOR

Figura 1.7.- *Componentes del transelevador*

1. Columnas: las columnas puedes estar formadas por un tubo estructural o bien por vigas cajón. Estas se fabrican con chapas de acero de alta resistencia debidamente conformadas y soldadas entre sí formando un cajón de forma rectangular (viga)

Figura 1.8.- *Columnas*

2. Testero o bastidor inferior: se trata de una estructura en forma de cajón, realizada con perfiles y chapas de acero soldadas entre sí, resistentes a la flexión y a la torsión gracias a las nervaduras de refuerzo soldadas en su interior a intervalos regulares

Figura 1.9.- *Testero o bastidor inferior*

3. Testero superior: está formado por placas soldadas, situadas en el extremo superior de la columna, que sirven de soporte para las ruedas horizontales de guía sobre el carril superior.

Figura 1.10.- *Testero superior*

4. Accionamiento de elevación: el mecanismo de elevación tiene por objeto impulsar el bastidor móvil en su movimiento vertical.

Figura 1.11.- *Accionamiento de elevación*

5. Bastidor móvil de elevación o cuna: tiene la función de desplazar la carga y la cabina en sentido vertical y efectuar los ciclos de recogida y deposito por medio del dispositivo de horquillas extensibles instalado sobre el mismo.

Figura 1.12.- *Cuna de elevación*

6. Sistemas de extracción: elemento determinante en el rendimiento de los transelevadores. En función de los requerimientos de cada instalación se parametriza dicho elemento para obtener mejores resultados.
    a. Horquilla telescópica de simple profundidad: mecanismo de manipulación horizontal que permite depositar o extraer unidades de carga en estanterías de simple fondo.
    b. Horquilla telescópica de doble profundidad: consiste en un mecanismo de manipulación horizontal que ayuda a depositar o extraer unidades de carga en estanterías de doble fondo mediante paletas telescópicas.

Figura 1.13.- *Sistemas de extracción*

7. Armario eléctrico: va a bordo del transelevador. Está colocado en la parte posterior de la columna delantera, y los controles están dispuestos de tal manera que el transelevador pueda ser dirigido como una unidad indicidual desde su plataforma segura

Figura 1.14.- *Armario eléctrico*

8. Transmisión de datos: se utilizan sistemas de comunicación óptica por infrarrojos para establecer comunicación de los terminales de periferia descentralizada con el PC o PLC fijos.

Figura 1.15.- *Transmisión de datos*

1.11.1.- VENTAJAS DEL USO DE TRANSELEVADORES

Ahorro de espacio

La instalación de transelevadores para palets supone un óptimo aprovechamiento del espacio de almacenaje disponible

Ahorro de tiempo

La automatización de las operaciones de ubicación y recogida de los palets supone un considerable ahorro de tiempo.

La posibilidad de ejecutar las entradas y salidas de mercancía en un mismo movimiento (ciclos combinados) reduce los desplazamientos.

Aumento de la productividad

En comparación con los sistemas convencionales, los almacenes automáticos mejoran la productividad en los procesos de entrada y expedición de los productos: incremento del flujo de palets/hora.

Ahorro de costes

Los almacenes automáticos para palets reducen los costes operativos en múltiples aspectos, como reducción del personal necesario para almacenaje, preparación de pedidos, expedición y administración

Versatilidad

El transelevador se puede combinar con estanterías de simple profundidad, de doble profundidad, e incluso de triple profundidad, en función de si se desea conseguir más cantidad de movimientos o más capacidad.

El sistema es flexible y escalable, ya que puede modificarse o ampliarse con el tiempo adaptándose a nuevas necesidades de la empresa.

Control y gestión más eficientes

Los almacenes automáticos para palets de Mecalux están controlados y gestionados por dos tipos de software:

- Software de control Galileo: sistema de control integrado responsable del gobierno de las máquinas. Conjuga de manera eficiente todos los movimientos necesarios para ejecutar las órdenes que provienen del software de gestión del almacén (ya sea Easy WMS de Mecalux o cualquier otro SGA del mercado)
- Software de gestión de almacenes Easy WMS: sistema responsable de la operativa de la instalación, desde los procesos de recepción, a los de almacenaje, preparación y expedición.

Seguridad

En los almacenes automáticos la seguridad de las personas y de la mercancía es absoluta. Además de numerosos dispositivos de seguridad a bordo de los transelevadores, también incorporan elementos de seguridad en el pasillo y un sistema de transmisión inalámbrica de señales de seguridad que activa las eventuales paradas de emergencia en la instalación.

Respecto a los operarios, los sistemas automáticos están concebidos para ser utilizados con una baja intervención humana, ya que sólo es necesario acceder al interior de los pasillos de almacenaje para esporádicas tareas de mantenimiento. Además, el perímetro está asegurado con cerramientos y puertas de seguridad con detectores y acceso restringido.

Respecto a la mercancía, su protección es total ya que sólo el transelevador puede acceder a ella, por lo que se elimina lo que se conoce como “pérdida desconocida” del producto, y se evitan posibles roturas por manipulación incorrecta. Todo ello reduce el número y la necesidad de controles intermedios.

Por último, la precisión de los transelevadores y el control de posicionamiento con telémetros láser evitan cualquier posible choque con la estructura de estanterías, asegurando su estabilidad y durabilidad.

Universidad de Oviedo

Trabajo Fin de Máster realizado por

Victoria Casares García

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

ANEXO II: Manual DesignerIV

Modelado de Transelevador
Automatic Warehouse Studio (AWS)

Julio de 2018

ÍNDICE MANUAL USUARIO DESIGNER IV

1. CAPÍTULO 1: INTRODUCCIÓN

Designer IV es la herramienta que se utiliza para desarrollar aplicaciones sobre Galileo IV. Esta herramienta cumple dos roles principales:

1. Servir de entorno integrado de desarrollo (IDE), de manera que, desde la misma aplicación, el programador puede definir la lógica del programa, las variables a utilizar, crear las visualizaciones que los operarios utilizarán y además, utilizar las herramientas de depuración previstas, de manera que puede depurar y encontrar los errores del sistema.
2. Actuar como sistema SCADA, que se comunica con el sistema de control, y muestra la visualización creada por el programador, interactúa con los operarios y envía y recibe los datos hacía y desde el sistema de control.

2. CAPÍTULO 2: INSTALACIÓN DEL ENTORNO DE DESARROLLO

2.1.- REQUERIMIENTOS Y PROCESO DE INSTALACIÓN

Los requerimientos mínimos para instalar el software Designer IV son:

- PC con Windows XP, Windows 7, Windows 2000 versión Servidor o profesional instalada.
- Si se desea conectar con las estaciones remotas (modo online), es necesario que el ordenador disponga de acceso vía TCP/IP a dicha estación.
- Si se desea compilar los programas que usen llamadas a componentes nativos en el código XanaO2, se debe de instalar también el Sistema de Control Galileo IV, a fin de que las librerías de componentes nativos estén presentes en el sistema o bien usar la opción explicada en “Solapa Opciones de compilación” en la página 18.
- FrameWork NET 2.0 y 4.0 (Requerido en el Gestor de Licencias y la Consola).
- MSVC runtime v10 (incluido con Galileo IV).

2.2.- PROCESO DE INSTALACIÓN

2.2.1.- DESIGNER IV

Para iniciar la instalación haga doble clic sobre el icono correspondiente al instalable de la versión deseada:

Tras las pantallas de bienvenida, acepte el diálogo del contrato de licencia.

Seleccione los módulos a instalar:

- Designer IV: Entorno de desarrollo.

Seleccione un directorio del disco donde instalar la aplicación. El directorio que se muestra por defecto varía dependiendo de dónde tengamos instalado el Sistema Operativo o de la versión del mismo.

Tras ello comienza la instalación. Al final de proceso se instala un acceso directo en el escritorio de la aplicación:

2.2.2.- DATOS Y PROGRAMA

En Designer IV, al igual que en Designer 3.2, se han separado los datos y los binarios de programa.

2.2.2.1.- BINARIOS DE PROGRAMA

Todos aquellos archivos con extensión .EXE, .DLL, .BPL, y .PDB se instalan en el directorio del disco duro que alberga al Program Files que, dependiendo del Sistema Operativo, puede ser Archivos de Programa, Program Files o Program Files (X86).

Internamente, el directorio de programas se guarda como una entrada en el registro del Sistema:

HKEY_LOCAL_MACHINE →SOFTWARE →Mecalux →Designer IV

La entrada se llama install_path, y contiene el directorio donde se encuentra

Designer IV instalado, por ejemplo:

C:\Archivos de Programa\MECALUX\Designer32

Esta cadena debe terminar obligatoriamente en \Datos

Se instalan en:

C:\Documents and Settings\All Users\Datos de programa\Mecalux\Designer IV

Los logs y ficheros .ini de configuración están en dicho directorio (no como en versiones anteriores en C:\Designer31 o C:\Designer 32)

La entrada en el registro para los datos, es:

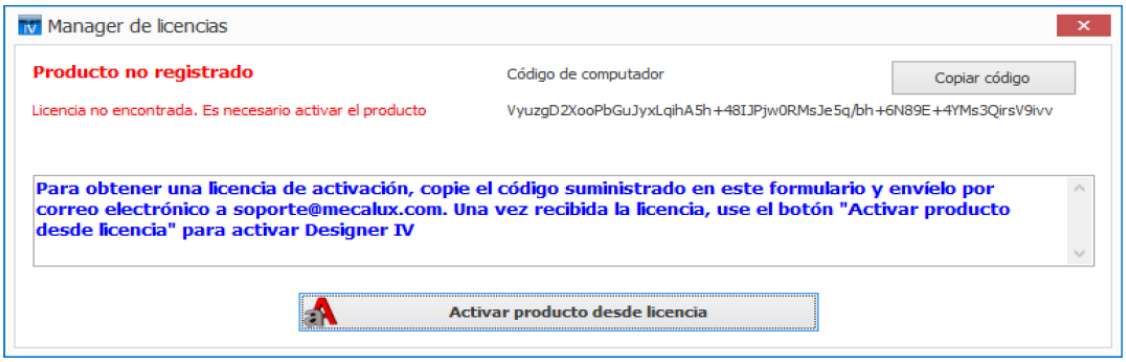
HKEY_CURRENT_USER → Software → MECALUX → Designer IV

2.2.3.- LICENCIA DESIGNER IV

Designer IV dispone de dos tipos de licencia:

- Licencia de Desarrollador: Es la misma licencia que la del Sistema de Control de Galileo IV. Puede obtenerla desde Designer IV o bien desde el manager de licencias de Galileo IV (puede consultar más información en el manual del Manager de Licencias.).
- Licencia de I+D: Licencia destinada a personal de I+D Control. Ofrece las mismas características que la licencia de Desarrollador más la opción del bloqueo de secuenciadores de repositorio (más información en "Pestaña Parámetros" en la página 55). Solo puede obtenerla desde Designer IV.

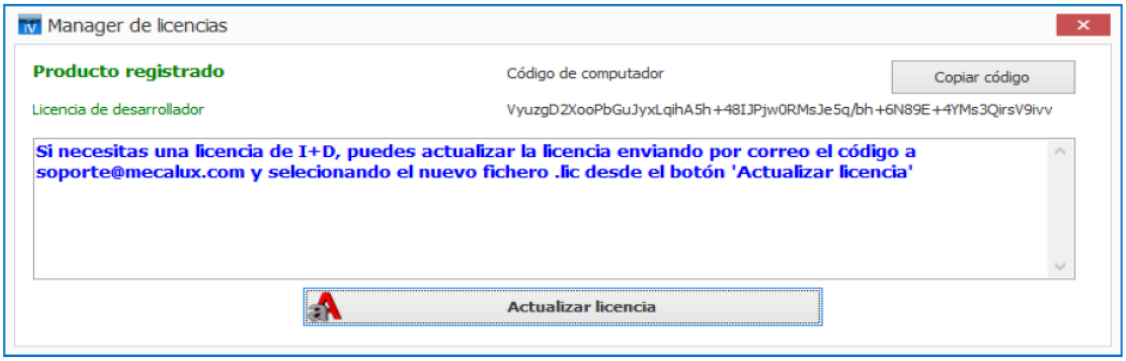
Cuando solicite la licencia a la dirección soporte@mecalux.com especifique el tipo de licencia deseado. Al iniciar la aplicación por primera vez muestra la siguiente ventana:



Los pasos a seguir para solicitar y activar una licencia están detallados en la ventana.

Tras la activación la aplicación actualizará la ventana indicándole si ha habido algún error. Puede consultar los errores típicos en el manual del Manager de Licencias.

Si la activación es correcta:



3. CAPITULO 3: ENTORNO COLABORATIVO

Designer IV es una herramienta que ofrece un entorno colaborativo, esto es, un entorno en el que se puede trabajar de forma remota sobre un proyecto único. De esta forma, cada usuario de la aplicación realiza consultas y modificaciones sobre un proyecto ubicado en un servidor.

La operativa que debe seguir para trabajar en es este entorno colaborativo es la misma si el proyecto se encuentra en un servidor remoto (más información en "¿Cómo abrir un proyecto ubicado en remoto?" en la página 30) o está ubicado en su propio PC.

Designer IV ofrece las garantías suficientes para que no haya cambios no deseados en el proyecto servidor. Para ello bloquea las pantallas u opciones para los distintos usuarios en función de las operativas que se estén realizando.

Las opciones disponibles en cada una de las pantallas son:

- Desbloquea las funcionalidades de la pantalla y la bloquea para otros usuarios en el proyecto. La pantalla no le permite desbloquearla si previamente lo ha sido realizado por otro usuario o tiene desbloqueada otra pantalla que dependa de ésta.
- Guarda en el proyecto todos los cambios realizados en la pantalla
- Descarta los cambios realizados en la pantalla y refresca con los datos originales del proyecto.
- Refresca con los datos originales del proyecto.

4. CAPITULO 4: GUÍA RÁPIDA DE LA APLICACIÓN Y HERRAMIENTAS

En este capítulo se explicarán las distintas pantallas y funcionalidades de la aplicación.

Antes de ello, mostraremos opciones que son comunes a todas las pantallas.

Las opciones más destacables del menú principal son:

- Archivo > Abrir proyecto remoto: más información en "¿Cómo abrir un proyecto ubicado en remoto?" en la página 30.
- Archivo > Exportar a ubicación: más información en "¿Cómo exportar un proyecto?" en la página 37.
- Archivo > Generación automática de proyecto: más información en "¿Cómo se puede generar un proyecto de forma automática?" en la página 31.
- Archivo > Versiones: más información en "¿Cómo se gestionan las Versiones de los proyectos?" en la página 39.
- Archivo > Mezcla de proyectos: más información en "¿Cómo realizar una mezcla de dos proyectos en Designer IV?" en la página 41.
- Archivo > Convertir proyecto de una versión anterior: más información en "¿Cómo convertir proyectos de versiones anteriores?" en la página 37.
- Editar > Buscar: más información en "¿Cómo realizar búsquedas en los proyectos en Designer IV?" en la página 43.
- Comunicaciones > Mostrar Averías: más información en "Ver Averías" en la página 183.
- Comunicaciones > Mostrar Red Galileo: más información en "Ver red Galileo" en la página 181.
- Herramientas >Mostrar Interface: más información en "Mostrar Interfaz" en la página 21.
- Herramientas > Opciones del entorno: más información en "Opciones del entorno" en la página 14.
- Herramientas > Histórico de averías y trasportes: más información en "Histórico de averías y transportes" en la página 21.
- Herramientas > Comparación de secuenciadores: más información en "¿Cómo comparar secuenciadores?" en la página 60.

4.1.- OPCIONES DEL ENTORNO

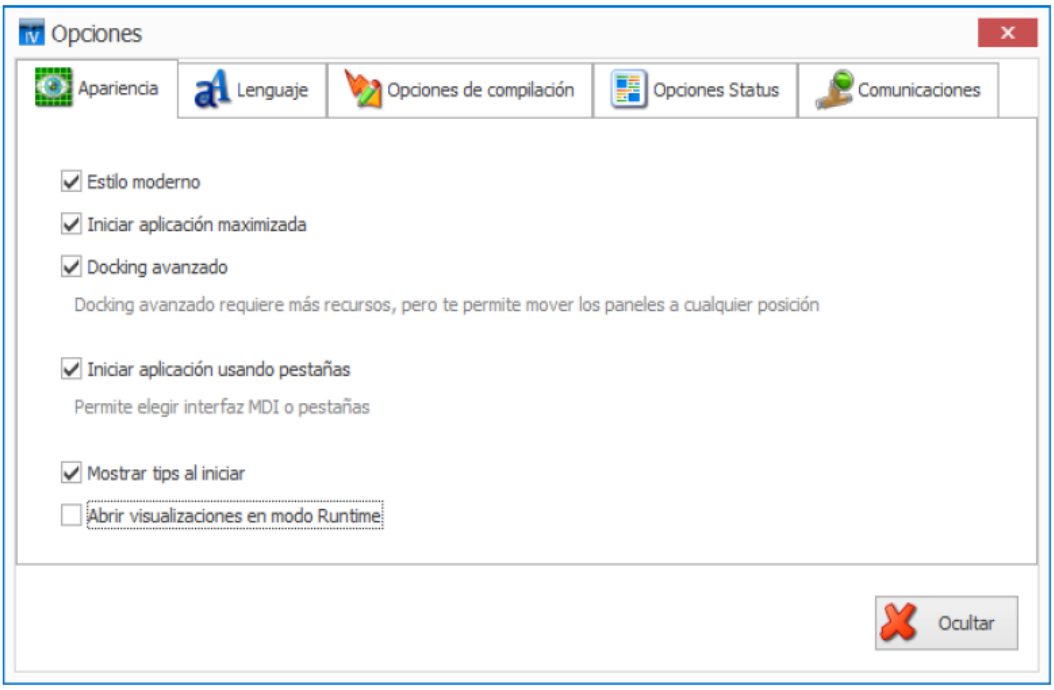
El entorno de desarrollo dispone de opciones configurables que facilitan al usuario la posibilidad de personalizar el entorno o que le permita obtener un mejor rendimiento del mismo.

La ventana tiene distintas solapas:

- Solapa Apariencia.
- Solapa Lenguaje.
- Solapa Opciones de compilación.
- Solapa Opciones de status.
- Solapa Comunicaciones.

4.1.1.- SOLAPA APARIENCIA

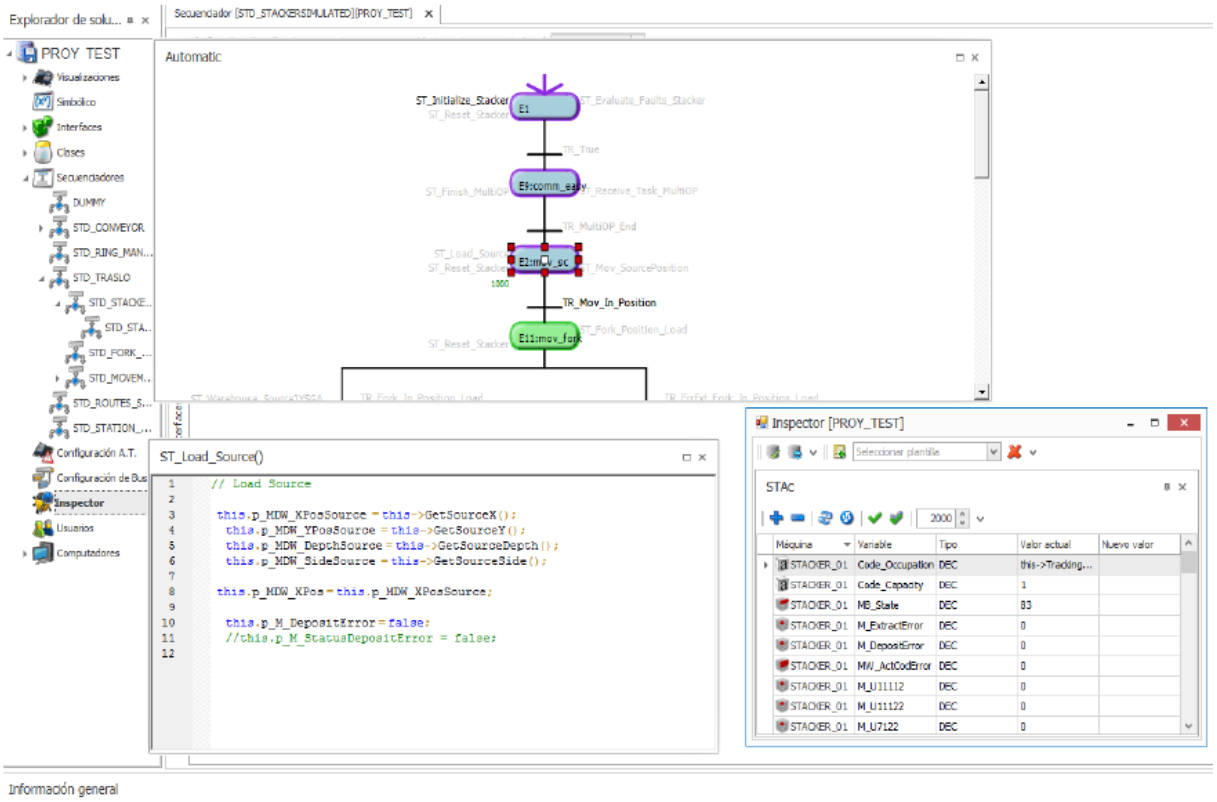
Permite establecer opciones relativas a la apariencia del entorno.



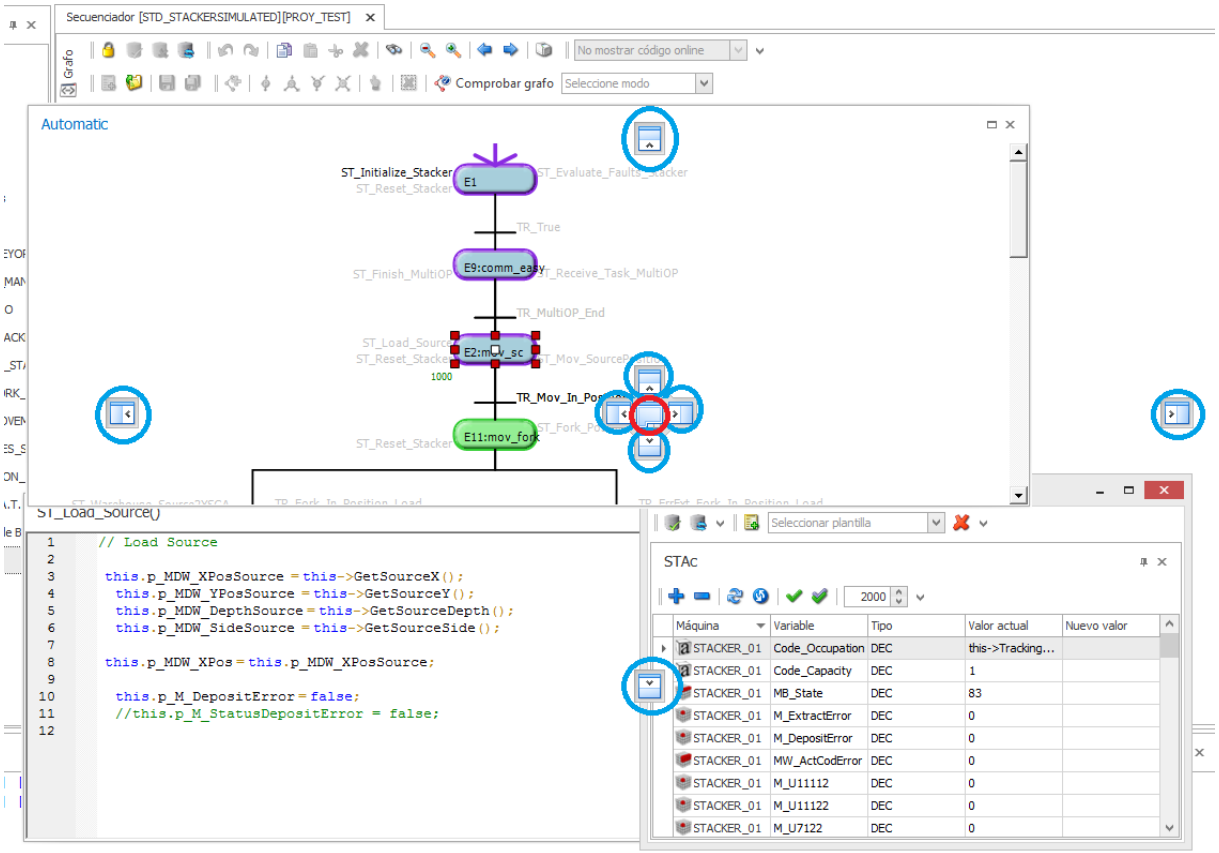
- **Estilo Moderno**: Selección del tema de Windows clásico o estilo moderno del Designer IV.

- **Iniciar la aplicación maximizada**.

- **Docking avanzado**: Permite deslocalizar los paneles de la aplicación. Muy útil para poder visualizar más información de forma simultánea. Requiere más recursos del sistema.
  Un ejemplo de paneles deslocalizados:



Para volver a localizar un panel, dentro de otra ventana o a nivel de pantalla general, debe arrastrar éste sobre las marcas que aparecen en la imagen en azul. Si desea colocar los paneles a modo de pestañas, siempre que sea posible, debe elegir la opción marcada en rojo.



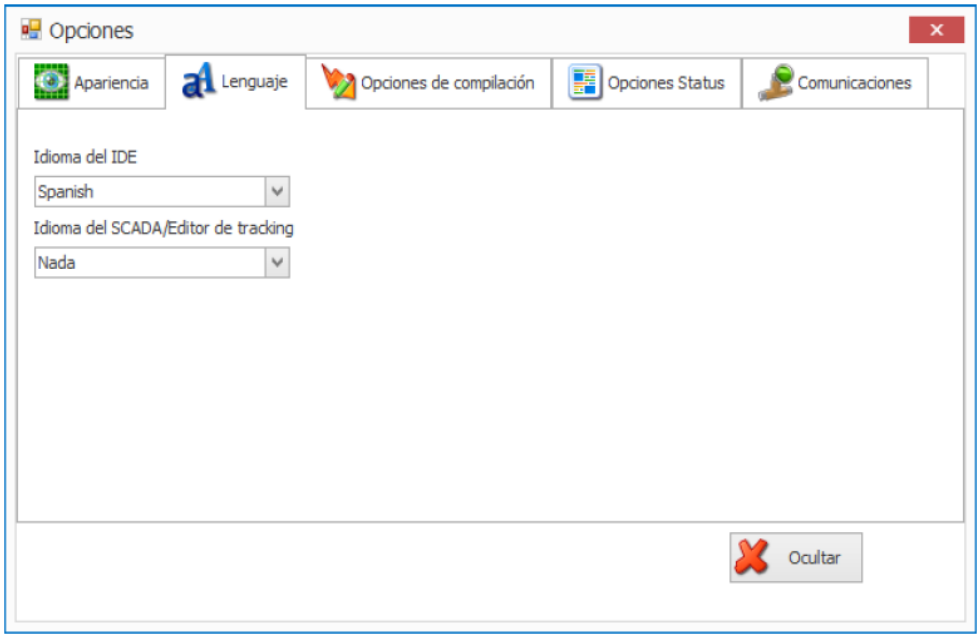
- **Iniciar aplicación usando pestañas**: Permite iniciar la aplicación con las pantallas en pestañas, en caso contrario, las pantallas se cargan en ventanas independientes como en versiones anteriores del Designer IV.

- **Mostrar tips al iniciar**: Habilita que se muestre una ventana, al iniciar Designer IV, que ofrece pequeñas sugerencias sobre cómo utilizar el entorno de desarrollo.

- **Abrir visualizaciones en modo Runtime**: Seleccionado hace que la aplicación abra en modo Runtime las visualizaciones (como en versiones anteriores del Designer). Es útil no seleccionarlo (se abre en modo Diseño) para evitar que cuando se produzcan errores en el Timer de la visualización se empiece a llenar de mensajes de error.

4.1.2.- **SOLAPA LENGUAJE**

Permite acceder a opciones del lenguaje.

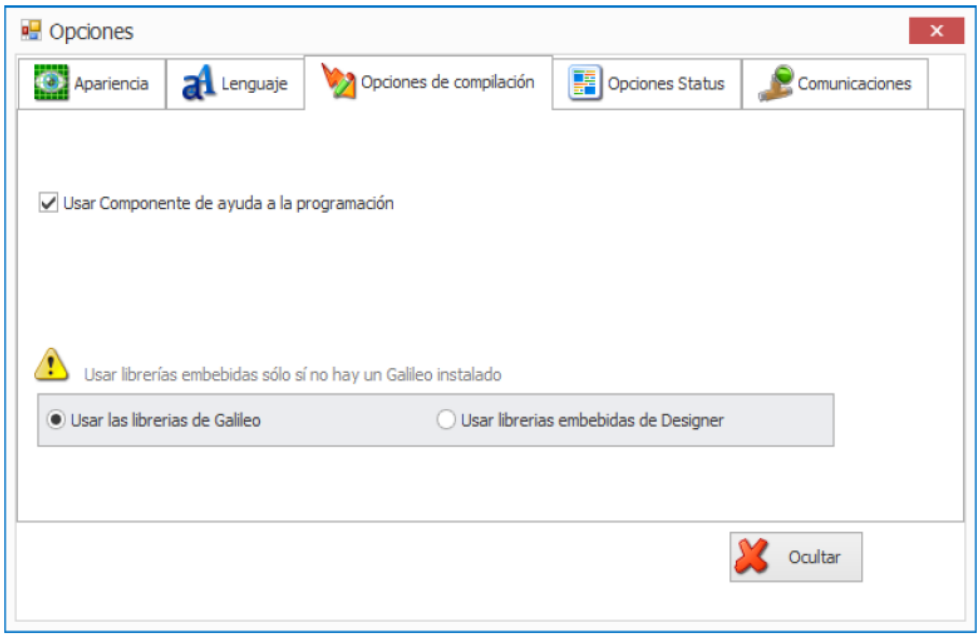


- **Idioma del IDE**: Nombre del idioma utilizado por el entorno de la aplicación. Puede cambiarlo seleccionando otro idioma en el desplegable.

- **Idioma del SCADA/Editor de Tracking**: Nombre del idioma utilizado para la traducción directa de cadenas de texto en la visualización y editor de tracking.

4.1.3.- SOLAPA OPCIONES DE COMPILACIÓN

Permite acceder a opciones de la compilación de proyectos.



- **Usar Componente de ayuda a la programación**: seleccionado se muestran ayudas a la edición de código, más información en la página 79.

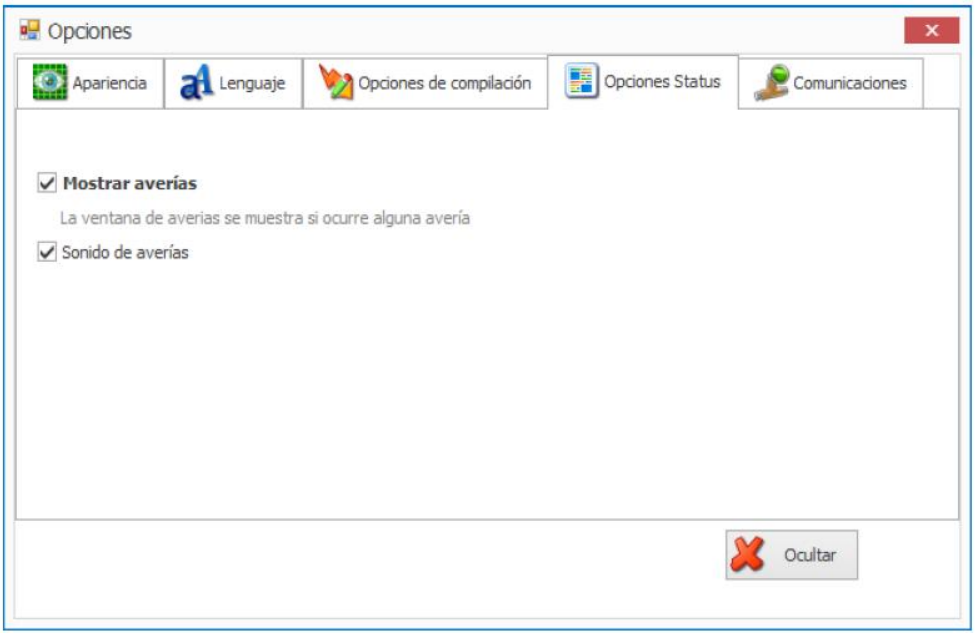
- **Utilizar las librerías de Galileo:** activada por defecto indica que se utilizarán las librerías distribuidas/instaladas por Galileo IV.

- **Utilizar librerías embebidas de Designer**: activada indica que se utilizarán las librerías distribuidas/instaladas por Designer IV.

Esta opción debe utilizarse si se trabaja con Galileo x64 o bien si no hay algún Galileo instalado.

4.1.4.- **SOLAPA OPCIONES DE STATUS**

Permite definir características para cuando Designer IV trabaja como Status:

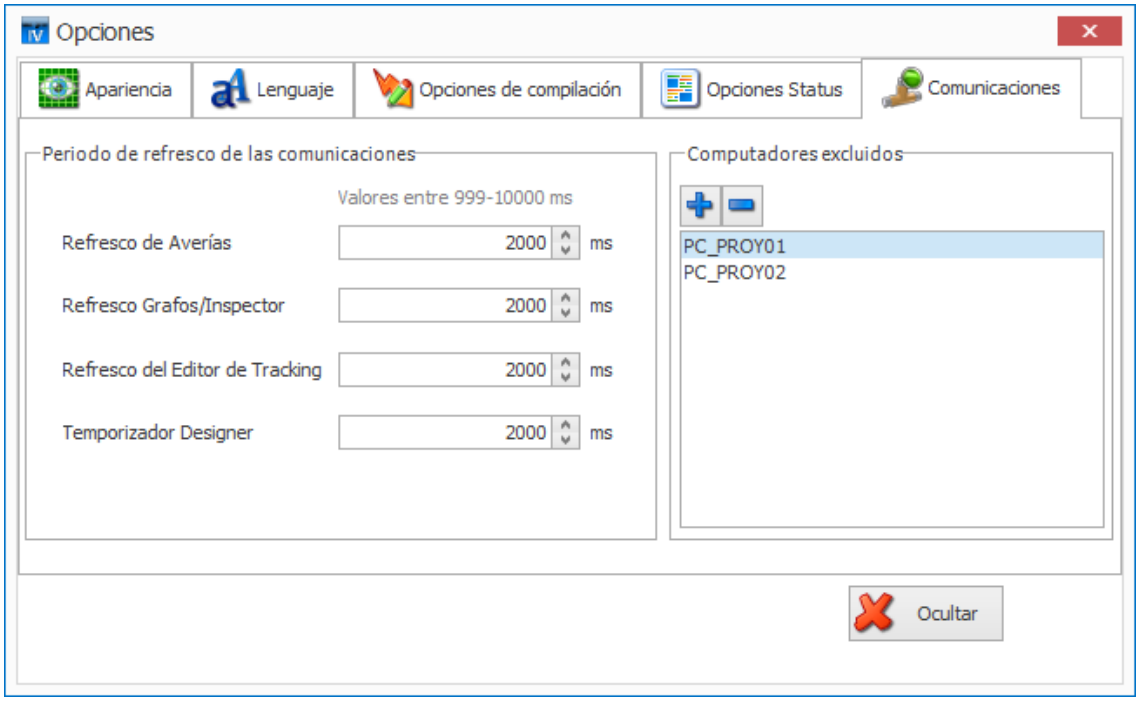


- **Mostrar Averías**: Cuando el Status esté ejecutándose muestra la lista de averías si ocurre alguna de ellas. En caso contrario, la lista esta oculta y tendrá que ser mostrada a petición del usuario desde el menú Comunicaciones > Mostrar Averías.

- **Sonido de averías**: Cuando existan averías en la instalación se activa el sonido de una alarma de aviso.

4.1.5.- SOLAPA COMUNICACIONES

Permite modificar los tiempos de refresco de distintas pantallas:



- **Periodo de refresco de las comunicaciones.**

  - **Refresco de Averías**: Tiempo de refresco de la ventana de averías, más información en "Ver Averías" en la página 183.

  - **Refresco Grafos/Inspector**: Tiempo de refresco de los grafos de los secuenciadores y en la pantalla Inspector de variables, este último puede particularizarlo en "¿Cómo visualizar variables de un secuenciador?" en la página 113.

  - **Refresco del Editor de Tracking**: Tiempo de refresco del Editor de Tracking, más información en "Editor de Tracking" en la página 88.

  - **Temporizador Designer**: Tiempo de refresco de utilizado en algunas opciones genéricas de la aplicación.

- **Computadores excluidos**:
  La configuración en este panel es para casos muy puntuales y hay que diferenciarlo de lo explicado en el caso de exclusión de comunicaciones que se detalla en la "¿Cómo editar un computador?" en la página 46.

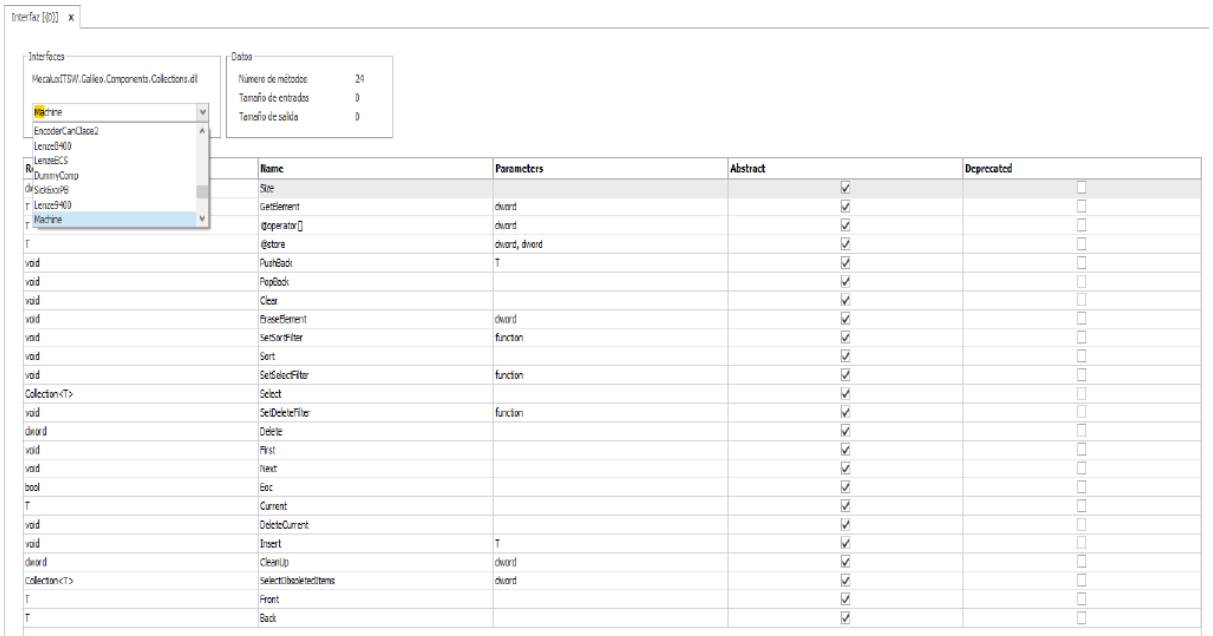
Solo tiene sentido configurar este panel cuando el PC en el cual estamos desarrollando no pertenece a los computadores del proyecto. También puede ser utilizado para un PC solo de visualización y externo al proyecto el cual no queremos que por ejemplo comunique con algunos computadores del proyecto.

4.2.- **MOSTRAR INTERFAZ**

Para la invocación de un método nativo en el lenguaje XANAO2, es muy útil tener una referencia a las funciones contenidas en la librería de componentes nativos instalada en el sistema. Estas referencias están descritas en la documentación sobre componentes en el Manual del Sistema de Control Galileo.

Designer IV dispone de un acceso rápido a estos componentes para facilitar su correcta escritura y definición de parámetros.

Seleccione esta opción desde el menú principal **Herramientas > Mostrar Interfaz**:



Los campos más relevantes que muestra la ventana son:

- **Interfaz**: seleccione en el desplegable o escriba el componente cuyas características quiere consultar.
  La parte superior muestra la dll asociada al componente.

- **Datos**
  - **Número de métodos**: Métodos que tiene el componente seleccionado.

- **Tamaño entradas**: Ocupación del componente en la periferia de entradas si es necesario (en el caso mostrado el componente no ocupa periferia).
- **Tamaño salidas**: Ocupación del componente en la periferia de salidas si es necesario (en el caso mostrado el componente no ocupa periferia).

- **Tabla de métodos**
  En la parte inferior de la ventana la aplicación muestra una tabla con la enumeración de los métodos del componente seleccionado. Para cada método aparecen varios campos:
  - **Tipo de retorno**
  - **Nombre del método**
  - **Parámetros**: Tipo de los parámetros de entrada del método.
  - **Abstracto**: Indica aquellos métodos que no pueden ser instanciados.
  - **Deprecado**: Indica si el método está deprecado por lo que no se recomienda su uso.

 Para que esta opción funcione correctamente, es necesario que el Sistema de Control Galileo esté instalado en la misma computadora del entorno de desarrollo (aunque no esté activo o se encuentre desactivado por no tener licencia) o bien se haya seleccionado el uso de las librerías embebidas del Designer IV.

4.3.- HISTÓRICO DE AVERÍAS Y TRANSPORTES

Al instalar el Designer IV, se instala la utilidad extra HistReport.

Este programa está accesible desde el menú **Herramientas** > **Histórico de averías y transportes**.

Su finalidad es la de poder generar informes sobre los históricos de movimientos producidos en las máquinas gobernados por el Sistema de Control Galileo.

Los criterios para la generación de informes son dos:

- **Trabajar sobre**:
  - **Averías** → Genera un informe con las averías sobre los filtros seleccionados.
  - **Transportes** → Genera un informe con los transportes sobre los filtros seleccionados.

- **Emulación TA 3.0**: Indicar si la instalación sobre la que se realiza el informe trabaja con el Agente de Transportes 3.0

- **Id Almacén**: Indicar el identificador del Almacén proporcionado por el EasyWMS® para realizar las consultas.

- **Fechas:** Indicar en cada uno de los desplegables las fechas inicio y fin entre las cuales se desea generar el informe.

- **Datos transporte**

Este panel aparece si se selecciona la generación de un informe de transportes.

- **Filtrar por número de Palet** → Se deben de introducir cadenas literales, ya que la búsqueda hará que se indiquen todos los transportes que contengan en el *número de Palet* el valor indicado.
- **Filtrar por número de transporte** → Se deben de introducir cadenas literales, ya que la búsqueda hará que se indiquen todos los transportes que contengan en el *número de transporte* el valor indicado.

Una vez establecidos los filtros, es el momento de generar el informe:

- 

Exporta los resultados de la consulta a una BBDD Access, cuyas ubicaciones ha de especificar, a fin de poder ser consultada desde cualquier otro equipo.

- 

Borra de la BBDD los elementos que concuerden con la consulta (para mantenimiento o limpieza de tablas).

- 

Imprime el informe generado de averías o de transportes. Si se marca el check **Visualizar antes de imprimir** se muestra en pantalla el informe antes de ser enviado a la impresora.

Esta herramienta también puede ser invocada desde el JScript de los formularios de visualización. Para ello basta con invocar a la función:

```
ShowReportFailures( TForm Formulario );
```

El formulario necesario suele ser frmEdicionForms.

4.4.- COMPILACIÓN

Tras tener todos los elementos programados y configurados, es necesario compilar el proyecto a fin de generar el archivo .plc que contiene la información necesaria para que Galileo IV pueda manejar la instalación.

Se debe de compilar el proyecto tantas veces como Computadores estén definidos, haciendo una compilación con cada computador. Si todas las compilaciones son correctas, se pueden transferir los ficheros *programa.plc* a la carpeta donde se encuentre instalado Galileo IV de cada computador asociado, y arrancar el servicio.

También se puede realizar esta transferencia desde el propio formulario de compilación, pero se requiere que el computador tenga el servicio de Galileo IV funcionando.

Para gestionar las opciones compilación, haga clic en  en la barra de herramientas. La aplicación muestra la siguiente ventana:

Las opciones disponibles son:

- **Avisar de parámetros sin asignar**: Comprueba que todos los parámetros definidos por el secuenciador tienen una variable asignada en las máquinas cuyo código se está generando, en caso contrario se genera una línea de aviso.

  Esto no tiene por qué provocar el fallo de la compilación, puede darse el caso que dicho parámetro no se esté usando y por lo tanto podemos descartar el aviso de forma segura (esto es normal en el caso de parámetros heredados por un secuenciador del padre, pero que no se usan en su código).

- **Generación segura**: Soluciona posibles errores en los métodos heredados de un secuenciador.

  Es posible que un secuenciador tenga asociado un método perteneciente a un ancestro no directo, pero que ha sido sobrecargada en un ancestro más directo. Como los métodos son sobrecargados por orden de herencia (es decir, la función de un descendiente siempre oculta a la del padre), esto lleva a que la fase de generación de código no es capaz de encontrar el método buscado, y por lo tanto, dicho código no aparecerá en el programa (con los consiguientes problemas que esto puede acarrear). Las causas por las que esta situación puede darse son variadas, y tienen que ver en su mayoría con la forma y orden en que se hayan editado los métodos y secuenciadores.

  Si se sospecha de esta situación, al activar esta comprobación, la fase de generación de código usará un algoritmo diferente para acometer la tarea, y si no encuentra el método referenciado, intentará suplirlo con el más cercano (ignorando la sobrecarga de métodos). En la mayoría de los casos esto solucionará el error, pero es necesario revisar el código de los secuenciadores a continuación para tratar de corregir esa anomalía. Por lo tanto, esta opción no debería usarse de continuo, ya que evitaría que el programador se apercibiera de este error.

- **Servidor**: Seleccione en el desplegable la computadora del proyecto para la que queremos realizar la generación/compilación/transferencia.

  Este paso es importante, porque los ficheros generados serán distintos en función de la computadora seleccionada, dado que cada computadora lleva sus propias máquinas y también sus propios módulos de bus (determinados por las tarjetas que contiene).

- **Salida**
  La compilación de un proyecto se realiza en 2 pasos:
  1. Generar el código XanaO2 fuente del proyecto y compilarlo.
  2. Empaquetar el código máquina resultante en un fichero **.plc** que contiene toda la información necesaria para que Galileo IV pueda ejecutar ese programa. Estos pasos se producen de forma simultánea si hace clic en **Auto compilar**. Provoca que se ejecute el siguiente paso de forma automática si el primero terminó con éxito.

  La generación de código ahora tiene en cuenta la información del simbólico para generar el código de todas las máquinas que pertenezcan a la computadora. Por eso, el fichero generado puede ser de un tamaño mucho mayor que para cualquier secuenciador.

  Si la generación y compilación se realizan con éxito, podremos realizar el paso final de transferir el fichero generado a la computadora de destino.

La **transferencia** del fichero **programa.plc** se realiza **en modo online** y el computador con el servicio de Galileo IV arrancado. Se establece una conexión de red entre Galileo IV y el entorno de desarrollo para transferirle el fichero, localizado en la carpeta **C:\ProgramData\Mecalux\Galileo** del computador de destino.

Si el entorno de desarrollo está en el mismo computador de destino, la **copia local** realiza la copia del fichero de forma local, sin necesidad de estar en modo "online". Si está configurado el uso de sonidos ("Opciones del entorno" en la página 14), se reproduce un sonido para indicar el fin (correcto o no) de un proceso de compilación o generación.

Las opciones de salida son:

- **Solo compilar.**
- **Compilar y copia local.**
- **Compilar y transferencia online.**
- **Solo copia local (sin compilación).**
- **Solo transferencia online (sin compilación).**

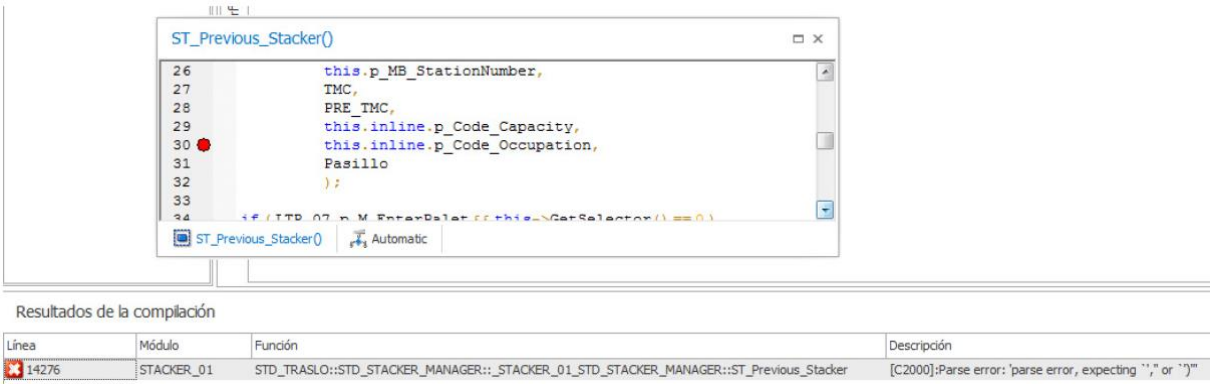
Para realizar la **compilación**, haga clic en  en la barra de herramientas.

En los paneles inferiores se muestra automáticamente el panel "**Resultados de la compilación**". También puede mostrar de forma manual el mismo panel haciendo clic en  en la barra de herramientas.

Si el proceso de generación o compilación termina con error se muestra en el panel anterior el mismo. Haga clic sobre el error o warning para localizar el código donde se produjo el error.

Dependiendo del tipo de error producido, puede que no sea posible localizar este código sobre el proyecto. En estos casos la aplicación muestra de forma automática una ventana con el código xoo generado (no editable) indicándole la línea donde se ha producido el error. Este código xoo puede consultarlo también para otro tipo de error desplegándolo desde el menú contextual ("**Mostrar código XOO**").

 Como el número de warnings puede ser muy elevado, la aplicación crea un log (WARNINGS.txt) donde aparecen todos ellos.



4.5.- **HERRAMIENTAS GALILEO**

Designer IV ofrece al usuario una serie de acciones propias del sistema de Galileo.

Haga clic en  . En el desplegable seleccione la acción a realizar: arranque o paro del servicio de Galileo, mostrar logs o eliminarlos, mostrar la consola de Galileo, etc.

4.6.- **PROYECTOS EN DESIGNER IV**

4.6.1.- **¿CÓMO CREAR UN NUEVO PROYECTO?**

Haga clic en  para crear un nuevo proyecto o bien la opción **Archivo > Nuevo Proyecto** del menú principal.

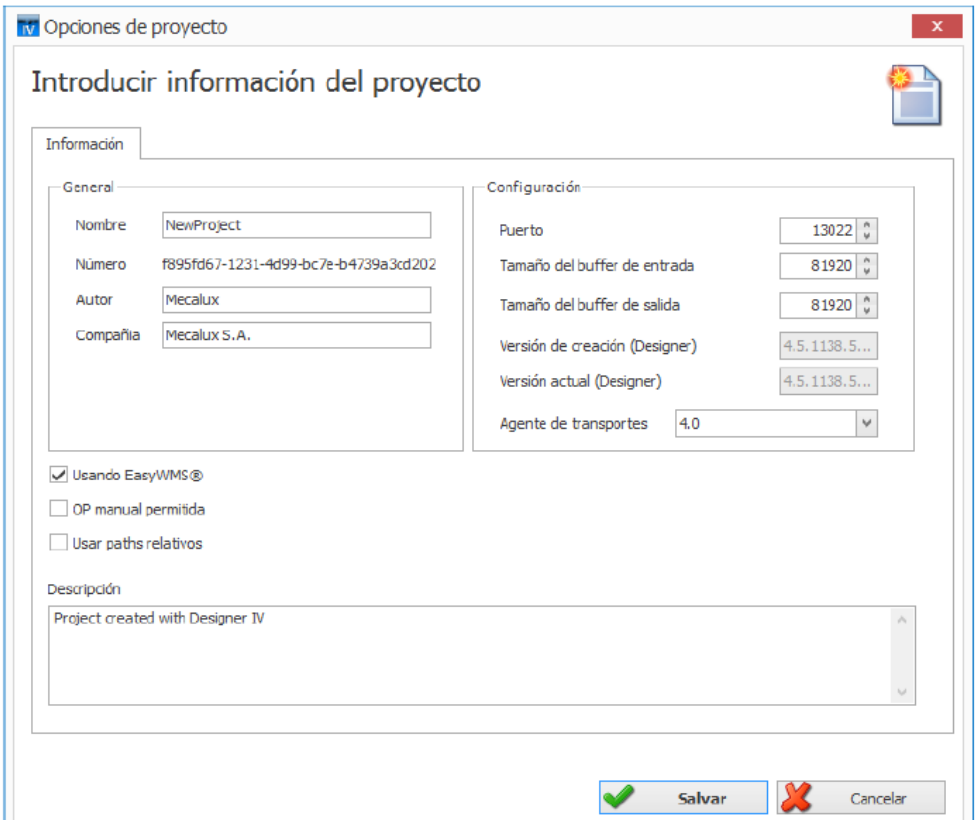
Indique el nombre y la ubicación donde será creado. La extensión del mismo es **.mgp** (Mecalux Galileo Project).

Tras ello la aplicación muestra el árbol general del proyecto.

4.6.1.1.- **OPCIONES DE PROYECTO**

Haga doble clic en 

para introducir la información básica del mismo desde la pantalla **Opciones de proyecto**.



Los campos que aparecen en este formulario son en su mayoría informativos. En cambio, otros como el Agente de Transportes son determinantes a la hora de ejecutar el proyecto. Esta información puede ser modificada a posteriori.

Los campos más destacables que nos encontramos en esta pantalla son:

- **Nombre**: Campo obligatorio. Debería ser un nombre corto y descriptivo del proyecto, puesto que será el indicador que luego aparecerá en el árbol de proyectos y del que colgarán todos los nodos del proyecto.
- **Número**: Identificador que se genera de forma automática y es único para cada proyecto.
- **Puerto**: Puerto de sockets que Galileo IV y el Entorno de Desarrollo usan para comunicarse. Este campo solo debería modificarse si se cambia también el puerto en Galileo IV, y siempre por causas de fuerza mayor.
- **Agente de Transportes**: Desplegable que indica con qué Agente de Transportes trabajará el proyecto actual (3.0-3.1-4.0).
- **Usando EasyWMS®**: Indica si se utiliza el software EasyWMS®, de manera que en el editor de tracking se muestre información adicional.
- **OP manual permitida**: Indica si el proyecto tiene disponible el pupitre manual para los usuarios en el Status Monitor. En caso de no permitir el pupitre manual, éste solo será accesible desde Designer IV.
- **Usar paths relativos**: Si el proyecto utiliza path relativos, la aplicación concatena la cadena fija C:\\ProgramData\Mecalux\Galileo\CustomInis a la variable string correspondiente.

Aparte de los datos anteriores, en esta pantalla se muestra la Versión de Designer IV con la que se creó inicialmente el proyecto y la versión actual con la que se está editando. En caso de que el proyecto haya sido creado con una versión anterior no reconocida se mostrará el mensaje UNKNOWN.

Finalmente, haga clic en 

.

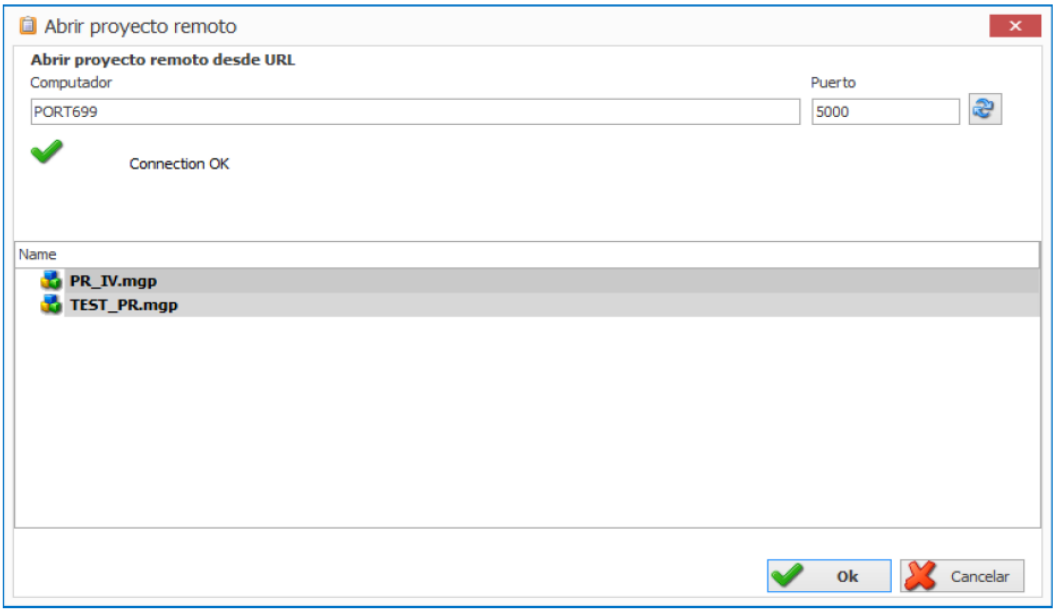
4.6.2.- ¿CÓMO ABRIR UN PROYECTO UBICADO EN REMOTO?

Haga clic en la opción **Archivo > Abrir Proyecto Remoto** del menú principal.

Indique el nombre del PC Servidor del proyecto remoto y su puerto de conexión.

Tendrá acceso a aquellos proyectos ubicados en el Servidor en el directorio configurado para tal efecto.

Finalmente haga clic en para acceder a dicha carpeta y seleccione el proyecto.



A partir de este momento ya está trabajando con el proyecto remoto y con las funcionalidades y seguridades que ofrece la aplicación para este **entorno colaborativo**.

4.6.2.1.- ¿QUÉ ES EL SERVIDOR DE PROYECTOS REMOTOS?

El Servidor de proyectos remotos es un PC que es capaz de funcionar como contenedor de proyectos de Designer IV, de forma que varios usuarios puedan trabajar de forma simultánea sobre el mismo proyecto.

4.6.2.2.- ¿CÓMO SE INSTALA Y CONFIGURA EL SERVIDOR DE PROYECTOS REMOTOS?

Para ello ha de instalar en dicho PC el servicio **Mecalux Designer IV Standalone Project Server** (no se instala por defecto) facilitado en el instalador del Designer IV.

Tras la instalación, y si necesita modificar los valores por defecto, puede editar el fichero de texto **Mecalux.ITSW.Designer.StandaloneProjectServer.exe.config,** ubicado en el directorio de instalación del Designer IV.

En dicho fichero de configuración los valores más destacables son:

- "RepositoryPath" value="**C:\DesignerRepo**": Directorio dónde se ubicarán los proyectos.

   La carpeta aquí indicada no se crea de forma automática con la instalación del servicio.

- "ListenPort" value="**5000**": Puerto de conexión accesible desde el resto de PCs.

Una vez configurados, ha de arrancar el servicio de forma manual desde el panel "Servicios" del S.O.

4.6.3.- ¿CÓMO SE PUEDE GENERAR UN PROYECTO DE FORMA AUTOMÁTICA?

La aplicación facilita la generación de proyectos de forma automática. Para ello se basa en:

• Fichero de definición **.xlsx**, generado desde EasyS.

• Proyecto de Repositorio (**.mgp**), el cual es el proyecto base o de referencia sobre el que se generará el nuevo proyecto. Este proyecto es facilitado por I+D Control.

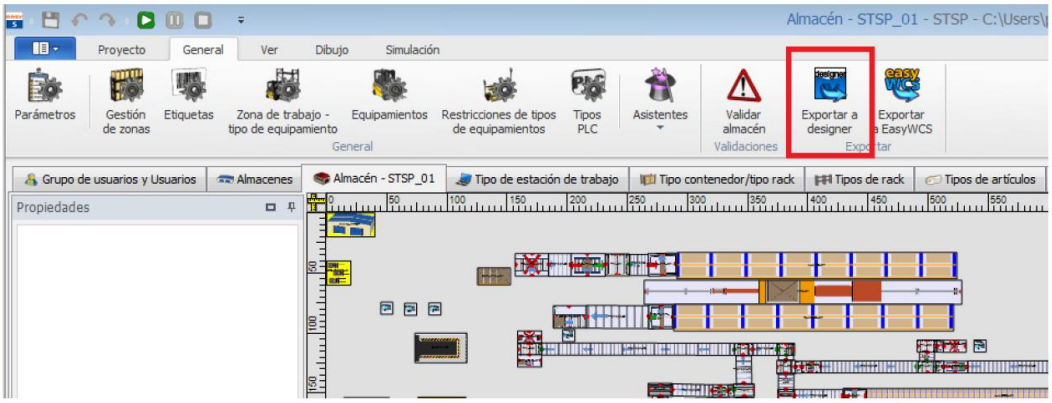
4.6.3.1.- FICHERO DE DEFINICIÓN (.xlsx)

El fichero de definición contiene información de las máquinas del proyecto a generar. Parte de datos proporcionados desde EasyS y posteriormente, completados en el propio fichero por el usuario.

4.6.3.1.1.- ¿CÓMO GENERAR EL FICHERO XLSX?

Los pasos a seguir son:

1. Desde el software **EasyS**, partiendo del proyecto de la instalación, seleccionar en la pestaña **Almacén** la opción **Editar Layout**.
2. Una vez tengamos el layout en la pantalla seleccionar la pestaña **General** y hacer clic en la opción **Exportar a designer**.



3. Esta opción genera el fichero xlsx que contiene todas las máquinas de la instalación e información de las mismas.

   EasyS genera dicho fichero basándose en una plantilla del mismo formato facilitada por I+D Control.

   Para cada máquina, EasyS genera información basándose en los datos configurados en su proyecto. Esta información es:
   - Nombre de la máquina y tipo de transportador (TC, TR, TM, TG).
   - Información si es una estación: número y tipo, tipo de altura y contenedor, operaciones realizadas (Búsqueda, Fin, Evento).
   - Ubicación lógica y ubicación física empleada para la visualización.
   - Definición de anteriores y posteriores.

4.6.3.1.2.- ¿CÓMO EDITAR EL FICHERO XLSX?

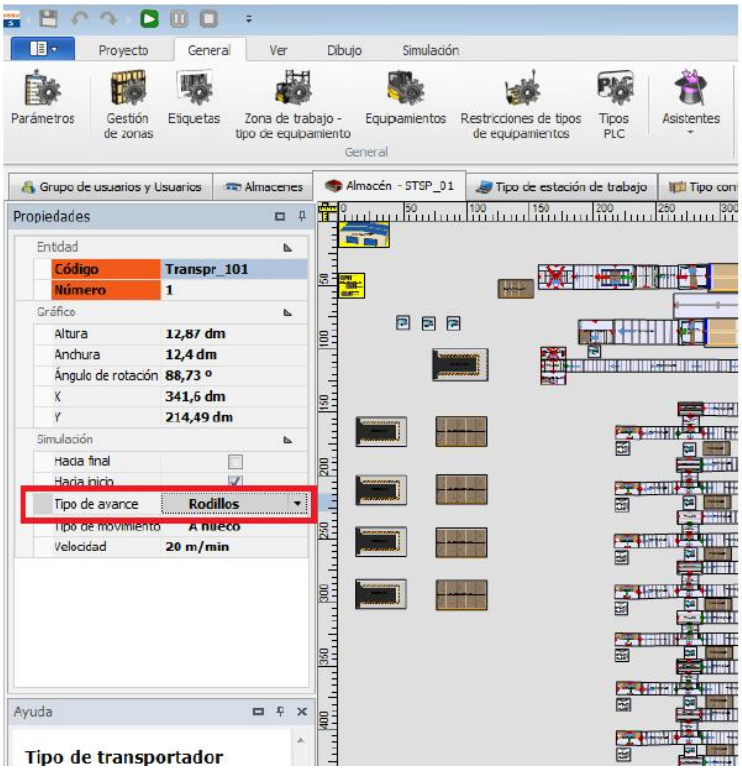
Una vez que el fichero se haya generado es necesario editarlo para indicar más información sobre las máquinas de la instalación. Este paso será realizado por el personal de Control y, por tanto, por el usuario final de Designer IV.

1. Para cada máquina, en la columna **[Machine]**, seleccionar mediante el desplegable la máquina base sobre la que se generará dicha máquina.
   Las opciones disponibles son filtradas en función del dato facilitado por EasyS en la columna **[Base Model]** (TC, TR, TM, TG).

Hay que tener en cuenta que el modelo base también influye directamente en la generación de la visualización. Por ello hay que tener especial cuidado en revisar que los tipos TC y TR coinciden con la máquina real.

Si el modelo base no corresponde con el tipo real de la máquina disponemos de dos opciones:

- Modificar el campo **Tipo de avance** en EasyS y generar de nuevo el xlsx.



- Modificar la columna **[Base Model]**.

2. El fichero xlsx dispone de una zona totalmente configurable donde el personal de control puede añadir más columnas.
   En estas columnas puede configurar variables de máquina utilizadas en el programa de control y asignarle un valor. Este valor puede ser indicado en la celda correspondiente de forma directa o bien puede ser indicado mediante formulación del propio Excel, por ejemplo utilizando el valor de una columna ya facilitada por EasyS.

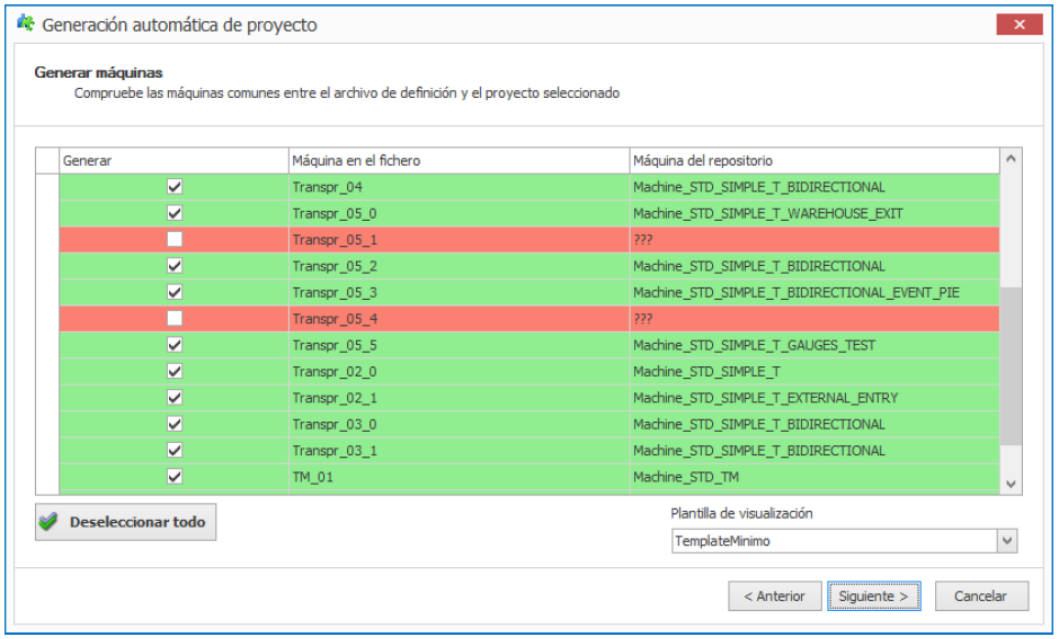
AE	AF	AG	AH
[Pos_4]	K_ConVariador	Code_ProgTipoManual	KB_TipoEvento
	true	MDW_ManualType = 63	0
	true	MDW_ManualType = 63	0
	true	MDW_ManualType = 63	0
	true	MDW_ManualType = 63	0
	true	MDW_ManualType = 63	0
	true	MDW_ManualType = 63	0

Si el usuario decide que estas nuevas columnas han de ser añadidas al exportador de EasyS, para que se generen de forma automática, ha de notificarlo al equipo de I+D Control para que si procede lo tengan en cuenta en la plantilla del fichero de definición.

4.6.3.1.3.- GENERACIÓN DESDE DESIGNER IV

Haga clic en la opción **Archivo > Generación automática de proyecto** del menú principal.

La aplicación muestra un asistente que le guiará durante todo el proceso. Por ejemplo:



Las pantallas más destacables del proceso son:

1. **Generación de datos**
   - **Fichero de definición:** fichero en formato .xlsx mencionado anteriormente.
   - **Crear Visualización**: indica que se ha de generar una visualización a partir de los datos facilitados en dicho fichero y la plantilla del proyecto del Repositorio.

2. **Seleccionar proyecto de repositorio.**

   El proyecto puede estar ubicado en local o remoto (más información en "¿Cómo abrir un proyecto ubicado en remoto?" en la página 29).

3. **Generar máquinas.**

Esta pantalla (mostrada en la imagen anterior) indica la relación existente entre las máquinas del proyecto del archivo de definición y las máquinas base del repositorio.

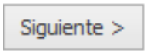
La pantalla le ofrece la posibilidad de generar o no cada una de las máquinas y de, si es necesario, cambiar la máquina predefinida correspondiente al repositorio.

En el desplegable **Plantilla de visualización** seleccione la visualización base a emplear en la generación automática del proyecto.

Esta visualización del proyecto del repositorio ha de ser de tipo **Plantilla**, para seleccionar este tipo consulte "¿Cómo crear y abrir una visualización?" en la página 117. Los LinkedElement creados en esta visualización han de tener configurados el campo **StrTag**, donde se ha de indicar de qué tipo es según corresponda TR, TC, TM, TG (al igual que lo indicado en la columna [Base Model] del fichero xlsx). También ha de configurarse en ellos la propiedad **Scale** (disponible en los MovementX) con el valor 100. Consulte más características en "Propiedades LinkedElement" en la página 133.

Si tras finalizar el proceso, una máquina de las generadas se muestra en la visualización mediante un panel sin imagen, indica que en la plantilla de la visualización no existe un linkedelement para la máquina del Repositorio. Esta situación puede notificarla al equipo de I+D Control para que si procede lo tengan en cuenta en la plantilla de la visualización del proyecto del Repositorio.

Las líneas en rojo indican que la máquina a generar no tiene correspondencia con la máquina base del repositorio, bien porque no está definida en el fichero (o no coincide el nombre) o porque no se encuentra en el propio proyecto del repositorio. Si no selecciona en el desplegable una máquina del repositorio no se generará dicha máquina. Esta situación puede notificarla al equipo de I+D Control para que si procede lo tengan en cuenta en la plantilla del fichero de definición o en el proyecto del Repositorio.

Tras estas pantallas haga clic en 

para crear el nuevo proyecto indicando el nombre y ubicación del mismo.

Con este paso se da por finalizado el proceso de generación automática de proyecto.

4.6.3.1.4.- ¿QUÉ NOS OFRECE EL PROYECTO GENERADO?

El proyecto generado dispone de:

- **Visualización** de la instalación, cada transportador (linkedelement) tiene definido su secuenciador; imágenes; variables asociadas a movimientos, iconos de avería, manual o contenedor; definidos sus anteriores y posteriores.
- **Simbólico**, con todas las zonas y máquinas generadas. El valor de las variables se reflejará si lo hemos indicado anteriormente en el xlsx.
- **Secuenciadores** asociados a las máquinas generadas y toda su jerarquía.
- **Computador** con las tarjetas definidas.
- **Configuración A.T.** con las **averías** y **estaciones**.

 Recuerde que las estaciones ha de comprobarlas a posteriori con las definidas en la BBDD.

- **Esclavos de bus** para las máquinas generadas.

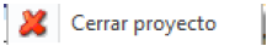
4.6.4.- ¿QUÉ SON LOS FICHEROS BACKUP?

Cada vez que un proyecto es abierto se crea una copia exacta del mismo y en el mismo path dónde se encuentra.

Este fichero tiene el mismo nombre que el fichero original pero con la extensión **mgpBackup** seguida de un índice que indica el número de backup (mgpBackup0...mgpBackup9).

El índice más bajo es el backup mas reciente (como en versiones anteriores de Designer). Los ficheros rotan para mantener siempre el mgpBackup0 como el backup más reciente, de forma que el borrado se realiza siempre sobre el último (mgpBackup9).

4.6.5.- ¿CÓMO CERRAR UN PROYECTO?

Haga clic derecho sobre el nombre del proyecto en el árbol y desde el menú contextual mostrado haga clic en 

.

4.6.6.- ¿CÓMO EXPORTAR UN PROYECTO?

Con esta opción podemos hacer una copia del proyecto actual en la ubicación deseada.

Haga clic en la opción **Archivo > Exportar a ubicación** del menú principal.

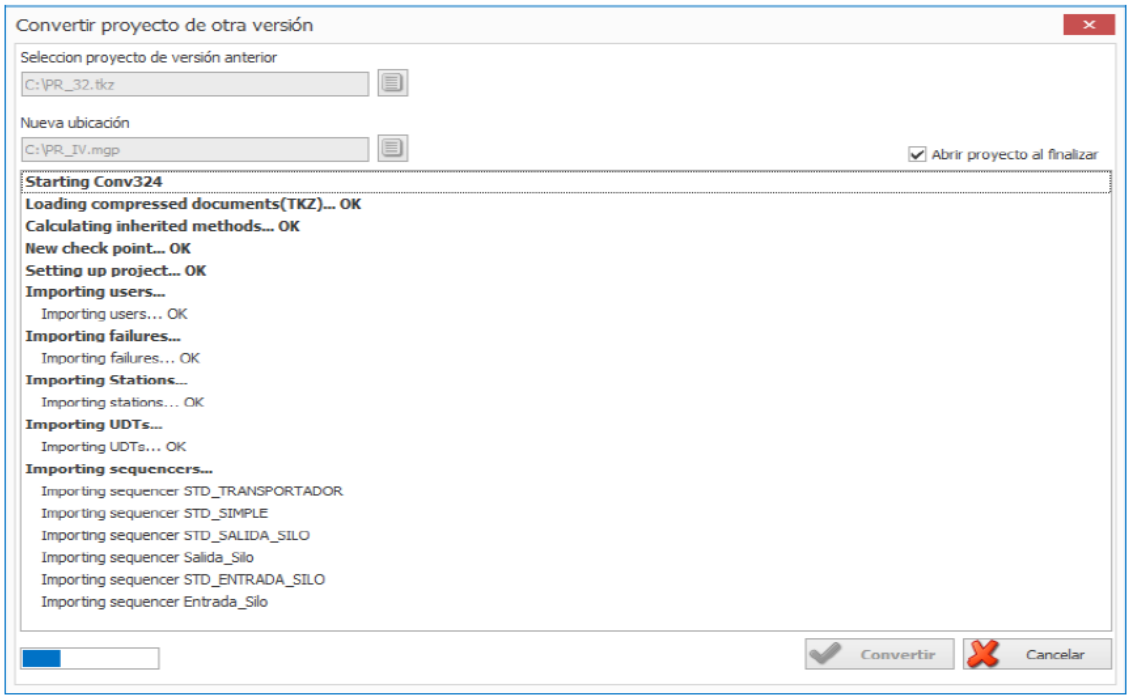
Recuerde guardar previamente, si es necesario, todas las modificaciones realizadas haciendo clic en .

4.6.7.- ¿CÓMO CONVERTIR PROYECTOS DE VERSIONES ANTERIORES?

Haga clic en la opción **Archivo > Convertir proyecto de otra versión** del menú principal.

Indique el path del proyecto original y el path y nombre del proyecto convertido.

Finalmente haga clic en Convertir:



Si durante el proceso de conversión se produce algún error, incompatibilidad, pérdida de información respecto al proyecto antiguo, etc. La aplicación lo muestra mediante warnings/errores en esta misma ventana o desde los logs de Designer IV.

Recuerde que en Designer IV no es posible trabajar con variables Globales. El convertidor le indicará mediante un mensaje si tiene variables globales en su antiguo proyecto. Las "Plantillas de Status" de versiones anteriores se convierten salvo las *plantillas de periferia no definida*.

En la conversión de **proyectos PLC** existen una serie de limitaciones que se han de tener en cuenta:

1. Todos los PLCs se convierten como **Siemens S7** (el más común). La información de los PLCs en Designer 3.1 se almacena fuera del proyecto. Esto hace que durante la conversión no se reconozca el tipo de PLC con exactitud. Una vez convertido puede cambiar el tipo del PLC.
2. Solo se convierten **máquinas asociadas** a un PLC. Si se tienen máquinas NO asociadas no se convertirán.
3. En Designer 3.1 las máquinas PLC no están asociadas a una Zona. En Designer IV esto no es posible, de manera que durante el proceso de conversión **se creará una zona por cada PC que tenga PLCs con máquinas**. Las máquinas PLCs se asociarán a estas zonas.

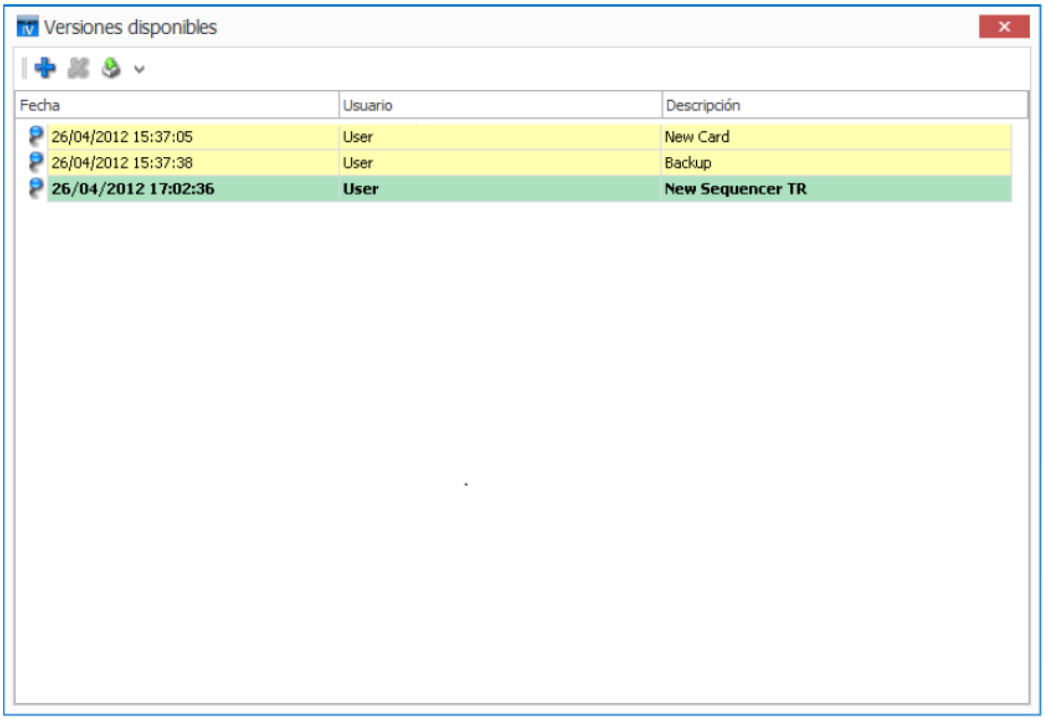
4.6.8.- ¿CÓMO SE GESTIONAN LAS VERSIONES DE LOS PROYECTOS?

La aplicación le permite generar versiones del proyecto sobre el que está trabajando.

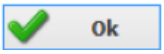
4.6.8.1.- ¿CÓMO CREAR VERSIONES DE UN PROYECTO?

Con el proyecto abierto haga clic en la opción **Archivo > Versiones** del menú principal.

La aplicación muestra la siguiente ventana:



Haga clic en  para añadir una versión y rellene el campo obligatorio "**Descripción**" indicando los cambios de esta versión.

Haga clic en 

para crear la versión. Ésta aparecerá en la ventana anterior en color verde.

 Si a la hora de crear una versión tiene algún cambio en el proyecto que no haya guardado, la aplicación le mostrará un mensaje indicando este hecho.

4.6.8.2.- ¿CÓMO EXTRAER UNA VERSIÓN DE UN PROYECTO?

Con el proyecto abierto haga clic en la opción **Archivo > Versiones** del menú principal. Seleccione la versión a extraer y haga clic en  indicando el nombre del proyecto.

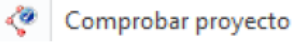
 Si no puede eliminar versiones debido a su importancia, y el tamaño del proyecto con éstas es muy elevado, es aconsejable que extraiga la última versión (por ejemplo cambiando de nombre a la versión actual) y elimine todas las anteriores para disminuir el tamaño de este proyecto.

4.6.8.3.- ¿CÓMO ELIMINAR UNA VERSIÓN DE UN PROYECTO?

Con el proyecto abierto haga clic en la opción **Archivo > Versiones** del menú principal. Seleccione la versión a eliminar y haga clic en  .

 Tenga en cuenta que eliminar versiones innecesarias implica que el tamaño de su proyecto disminuye.

4.6.9.- ¿CÓMO COMPROBAR SI HAY TRANSICIONES SIN ASIGNAR EN GRAFOS?

Haga clic derecho sobre el nombre del proyecto en el árbol y desde el menú contextual mostrado haga clic en 

.

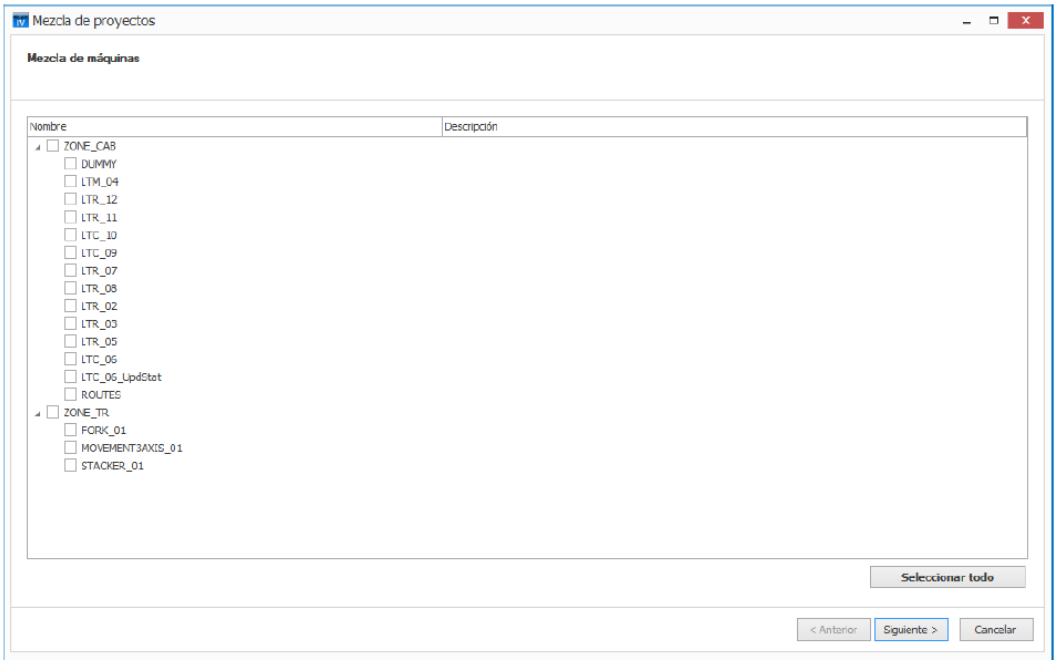
La aplicación le indicará, en el panel de Información General, aquellas transiciones cuya condición no tiene asignada un método.

4.6.10.- ¿CÓMO REALIZAR UNA MEZCLA DE DOS PROYECTOS EN DESIGNER IV?

Su finalidad es integrar partes de un proyecto en el proyecto que está actualmente en edición.

Para comenzar la mezcla seleccione la opción **Archivo > Mezcla de proyectos** del menú principal. Esta opción está habilitada solo si ya tenemos un proyecto en el árbol.

La aplicación muestra un asistente que le guiará durante todo el proceso.



Tras la selección del proyecto a mezclar, el asistente nos muestra la ventana más importante del proceso: "**Mezcla de máquinas**".

La importancia de la selección en esta ventana radica en que, en función de la misma, se seleccionarán de forma automática (y no editable) otros elementos del proyecto a mezclar.

Por poner un ejemplo, al mezclar la máquina STACKER_01, automáticamente en pantallas posteriores del asistente tendremos seleccionado:

- **Zona** a la que pertenece (y por ello el computador asociado a esa zona)
- **Secuenciador** de la máquina (si es heredado, su jerarquía correspondiente). Están representados en forma de árbol para reflejar su estructura de dependencia.
- **Clases** definidas en los parámetros del secuenciador. Están representados en forma de árbol para reflejar su estructura de dependencia.
- **Interface** que implementa (y la herencia de los mismos). En la descripción se indica su dependencia con otros interfaces.
- **Esclavos de bus** de la máquina, en base a la definición de sus variables en el simbólico.

- **Computador** al que pertenece (viene preseleccionado por la propia Zona).

 En las pantallas anteriores siempre puede seleccionar otros elementos.

Tras estas pantallas, el asistente le ofrece mezclar:

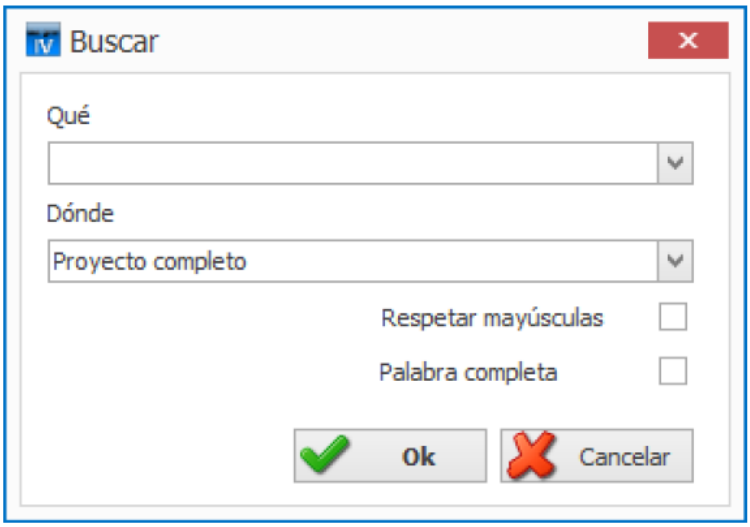
- **Plantillas del Inspector**
- **Configuración** del **Agente** de **Transportes**: Podremos seleccionar por separado las **estaciones**, dicha configuración *se añade* a la actual. Si algún elemento de los anteriores ya existe se sobrescribe.
  Para la lista de **traducciones** sólo se da opción a importarse completamente ("**Mezclar averías y traducciones**"). Con esta opción activa se añaden a la lista de traducciones actuales las cadenas nuevas o se sobrescriben aquellas que coincidan en la misma Cadena Original.
  Para la lista de **averías** también se da opción a importarse completamente ("**Mezclar averías y traducciones**"), siendo añadidas a la lista de averías actuales aquellas que tengan un número de avería nuevo. Si tienen el mismo número de avería, no se sobrescriben a no ser que se active la marca "**Sobrescribir averías por número**".
- **Grupos** de usuarios.
- **Visualizaciones**: las nuevas se renombrarían añadiéndoles un prefijo con la fecha actual.

 A tener en cuenta:

- Los elementos seleccionados serán **añadidos o sustituirán** a los ya existentes. La sustitución **solo** se realiza si éstos tienen el mismo nombre.
- Las **clases** llamadas en el **código** (no las definidas como parámetros) no se detectan y por ello no se mezclan de forma automática al seleccionar la máquina.
- Los **usuarios** no se mezclan para evitar bloquear de forma involuntaria el proyecto original. Sí se mezclan los grupos.
- Las **plantillas** del **Inspector** van ligadas a las variables del otro proyecto, por lo que si no mezclamos las mismas, y éstas no existen en el proyecto actual, las plantillas importadas mostrarán un aviso de ello.
- La mezcla de **esclavos de bus** implica la mezcla del computador asociado al mismo.

4.6.11.- ¿CÓMO REALIZAR BÚSQUEDAS EN LOS PROYECTOS EN DESIGNER IV?

Haga clic en  para mostrar la ventana "**Buscar**" o bien la opción **Editar** > **Buscar** del menú principal.

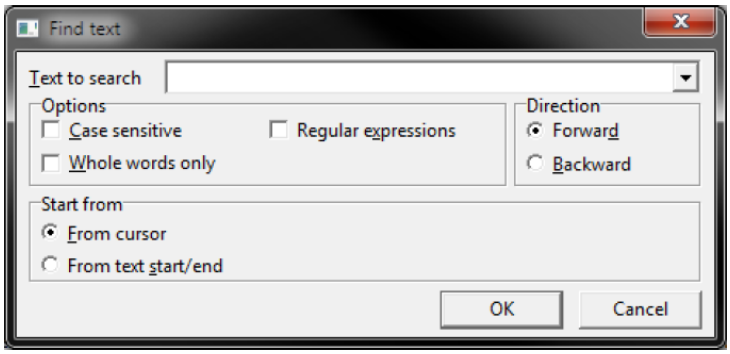


Los campos disponibles son:

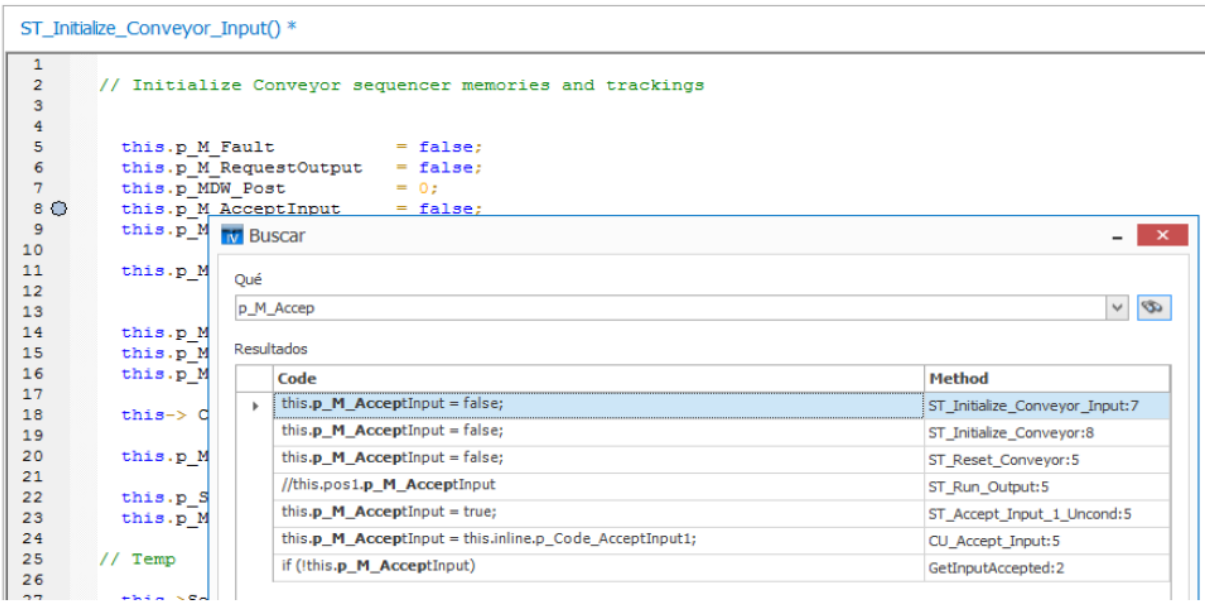
- **Qué**: cadena a buscar
- **Dónde**:
  - **Proyecto completo**: búsqueda global.
  - **Secuenciadores**: búsqueda en todos los secuenciadores. Comprueba si existe coincidencia en:
    Etapas y transiciones (muestra en qué etapa o propiedad es utilizado), parámetros, alias, números de etapas, códigos de error, descripciones de secuenciador...
  - **Código**: búsqueda en el código de todos los métodos definidos.
  - **Variables**: búsqueda en las variables de la pantalla Simbólico.
- **Respetar mayúsculas.**
- **Palabra completa.**

En función de la pantalla en la que nos encontremos, si utilizamos el atajo **ctrl+f**, la opción **Buscar** se comporta de forma dinámica:

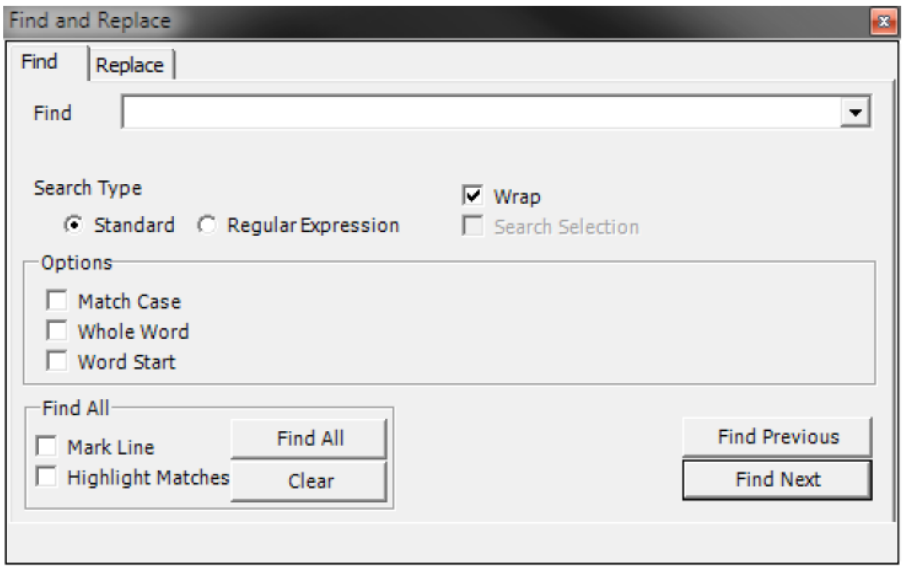
- Pantalla Visualizaciones, pestaña Code:



- Pantalla Secuenciadores: Complete el campo "Qué" con el texto y haga clic en  . En el panel inferior "Resultados" muestra la línea de código donde se encuentra resaltando en negrita el texto buscado. Al lado también le indica el nombre del método que lo contiene y el número de línea de código. Si hace doble clic en la línea, la aplicación le muestra ese método y le indica mediante un icono azul la línea donde se encuentra.



- Pantalla Secuenciadores con código abierto:



En cualquier pantalla, si queremos tener todas las opciones, basta con hacer clic en en la barra de herramientas.

4.7.- PANTALLA "COMPUTADORES" Y "PLC"

En estas pantallas se configuran las opciones de un computador que forme parte del proyecto y de un PLC asociado a un computador.

Haga clic en para habilitar todas las funcionalidades y bloquear la pantalla en el proyecto para el resto de usuarios.

4.7.1.- ¿CÓMO CREAR UN COMPUTADOR?

Haga clic con el botón derecho del ratón sobre el icono Computadores en el árbol del proyecto y seleccione desde el menú contextual la opción **Nuevo Computador.**

Seleccione el computador creado en el árbol de proyecto e indique si es necesario si se trata del **Computador por Defecto**. Toda Zona que no está asociada a un computador se asocia a esta computadora a la hora de compilar el proyecto. El computador por defecto se muestra con el icono .

Desde el menú contextual, sobre el computador creado, puede realizar una prueba de conexión seleccionando la opción Ping.

4.7.2.- ¿CÓMO CREAR UN PLC?

Al igual que las tarjetas, el computador definido también puede tener PLCs asociados.

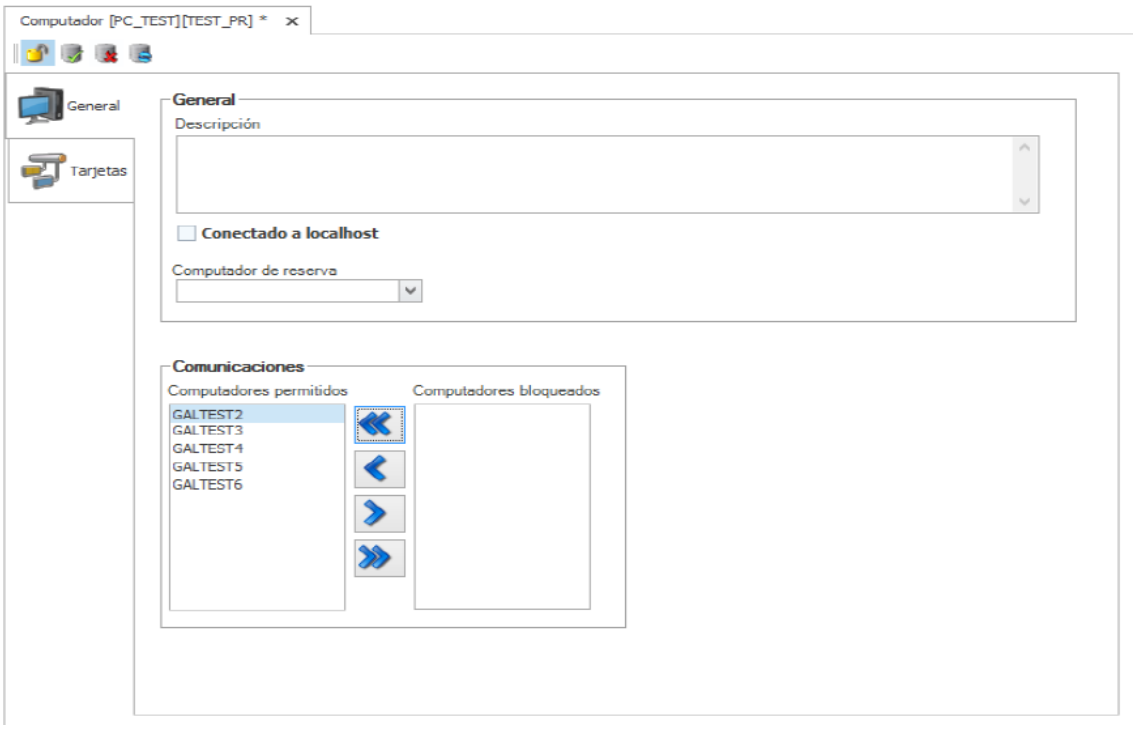
Haga clic con el botón derecho del ratón sobre el computador previamente creado y seleccione desde el menú contextual la opción Nuevo PLC.

4.7.3.- ¿CÓMO EDITAR UN COMPUTADOR?

Para modificar el nombre del computador selecciónelo en el árbol y mediante el menú contextual haga clic en la opción Renombrar.

También, mediante el menú contextual y haciendo clic en la opción Soy yo!, renombra el computador con el nombre del computador actual donde se está ejecutando Designer IV.

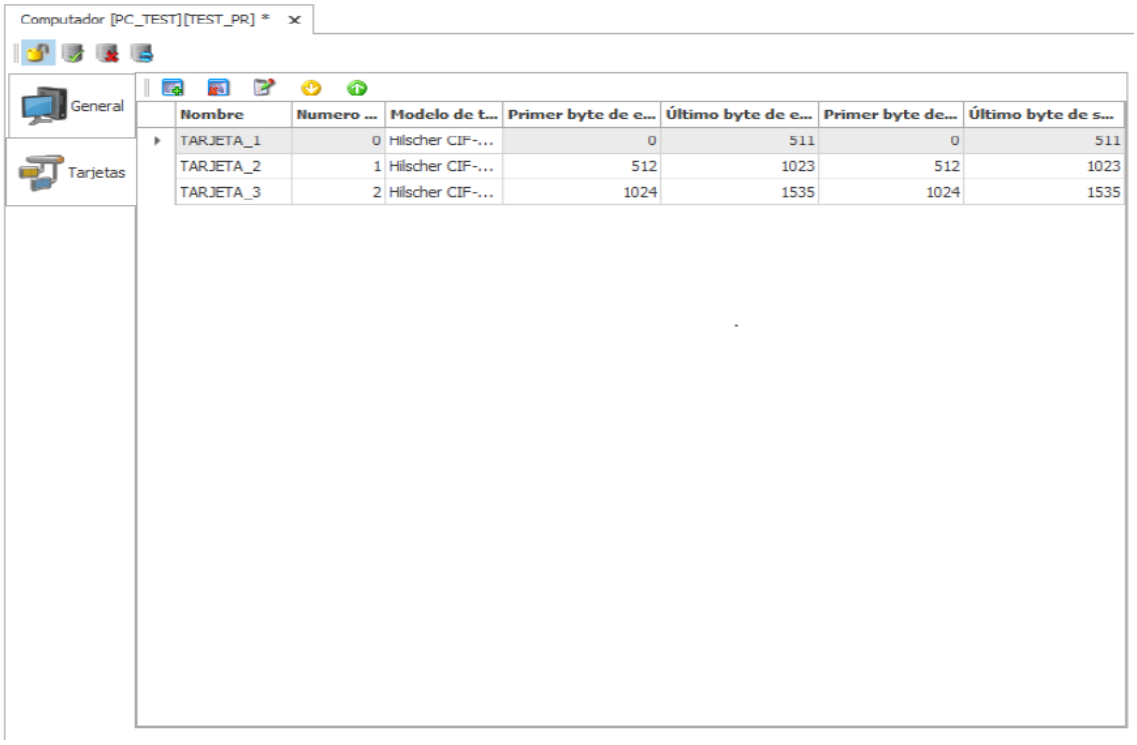
Haga doble clic sobre el computador en el árbol para mostrar su pantalla correspondiente.



Las opciones disponibles por computador son:

- **General**, propiedades básicas de la computadora:
  - **Descripción**: valor informativo para los desarrolladores del proyecto.
  - **Conectado a localhost**: en el Status, las comunicaciones se realizan hacia el localhost en lugar de hacia el propio computador.
  - **Computador de reserva**: nombre del computador configurado como respaldo o espejo en ejecución.
  - **Comunicaciones**: restricción de las comunicaciones con ciertos computadores cuando Designer IV o el Status se ejecuta en este computador de manera que se minimice la carga de red. Existe otro caso distinto de exclusión de comunicaciones que se detalla en la "Solapa Comunicaciones".

- **Tarjetas**, establece las propiedades más elementales asociadas una tarjeta.



Esta pantalla muestra información relevante de la tarjeta:

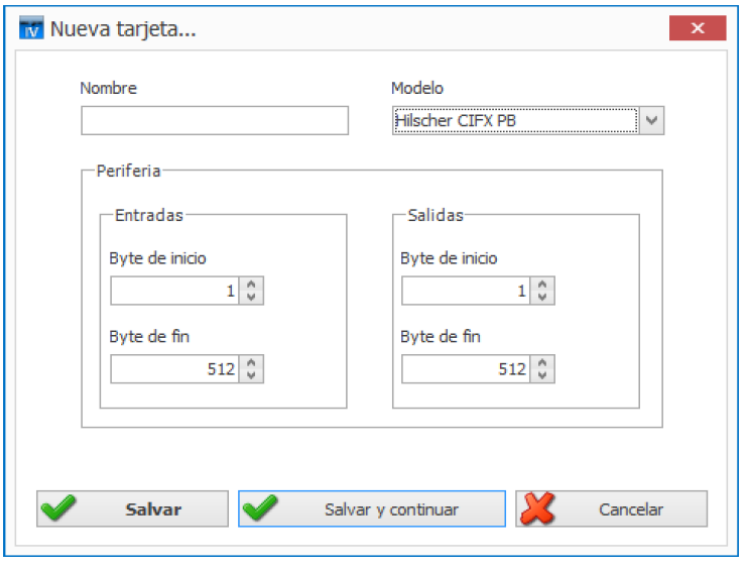
- **Nombre**: por claridad ha de ser único para que no se repita en ninguna otra tarjeta del proyecto.
- **Número de tarjeta**: indica el Slot PCI/PCI Express donde se inserta en el PC de control.
- **Modelo de tarjeta**: tipo de tarjeta del Slot indicado.

- **Posiciones de inicio/fin de las entradas y salidas**: limita el tamaño de los elementos que se pueden insertar en el bus de la tarjeta. Las direcciones de inicio y fin son absolutas.

Mediante los iconos y es posible modificar el orden de las tarjetas para que coincida el definido en el programa con el del slot del PC.

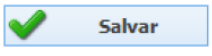
4.7.3.1.- ¿CÓMO CREAR UNA TARJETA?

Haga clic en para crear una nueva tarjeta.



Se pueden crear hasta un máximo de 4 tarjetas por computadora.

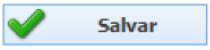
Rellene el campos **"Nombre"** (puede cambiarlo a posteriori), seleccione en el desplegable **"Modelo"** el tipo de tarjeta y, por último, indique el tamaño en bytes de las entradas y salidas de la periferia.

Finalmente, haga clic en 

.

4.7.3.2.- ¿CÓMO EDITAR UNA TARJETA?

Seleccione la tarjeta en la tabla y haga clic en para editarla.

Modifique los datos y haga clic en 

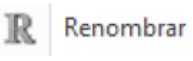
.

4.7.3.3.- ¿CÓMO ELIMINAR UNA TARJETA?

Seleccione la tarjeta en la tabla y haga clic en  para eliminarla.

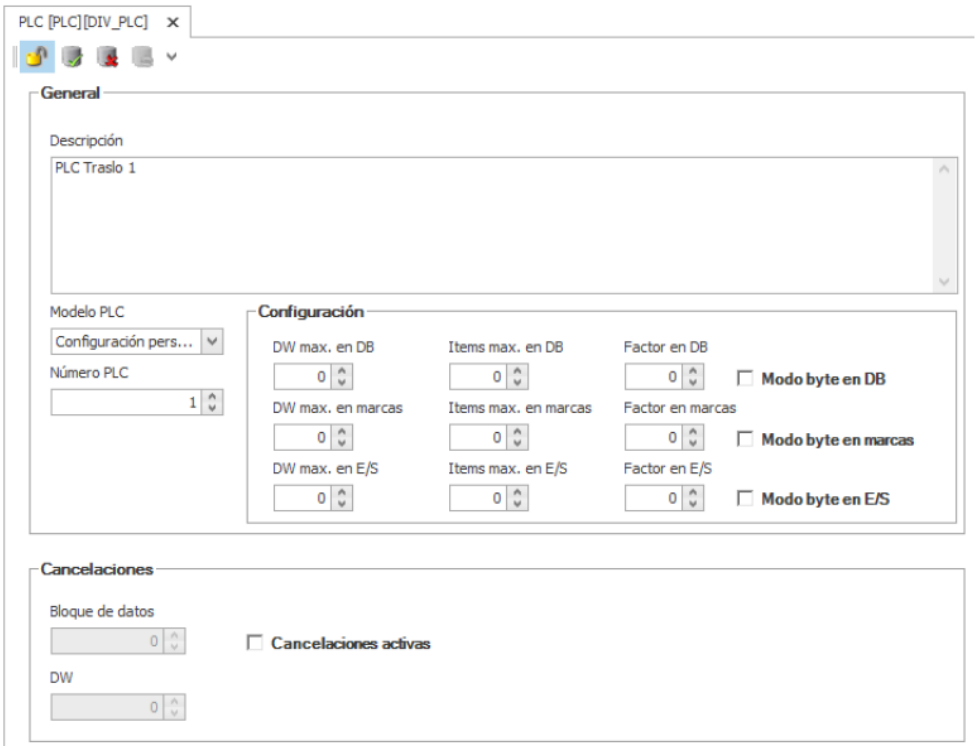
 La eliminación de la tarjeta de un computador implica que cualquier módulo asociado a esa tarjeta queda desligado automáticamente, y por lo tanto debemos volver a ligarlo desde la edición de bus o no podremos generar código que use variables en ese módulo.

4.7.4.- ¿CÓMO EDITAR UN PLC?

Para modificar el nombre del PLC selecciónelo en el árbol y mediante el menú contextual haga clic en la opción 

.

Haga doble clic sobre el PLC en el árbol para mostrar su pantalla correspondiente.



Las opciones disponibles por PLC son:

- **General**, propiedades básicas de la computadora:
  - **Descripción**: valor informativo para los desarrolladores del proyecto.
  - **Modelo PLC**: los modelos disponibles son

Modelo PLC	Descripción
Personalizado	Ver **Configuración**
Simatic S5	Comunicación con un PLC de tipo Simatic S5.
Simatic S7	Comunicación con un PLC de tipo Simatic S7.
Allen-Bradley	Comunicación con PLC de Allen-Bradley

  - **Configuración**
    - **DB**: configuración cuando se realizan peticiones a DBs
    - **Marcas**: configuración cuando se realizan peticiones a Marcas
    - **E/S**: configuración cuando se realizan peticiones a E/S

    Donde:
      - DW max: dirección máxima a partir de la cual se pueden pedir datos.
      - Items max: número máximo de elementos que se pueden pedir en una única petición.
      - Factor: indica el número de bytes que ocupa cada item.
      - Modo byte: indica si el PLC en ese tipo de almacenamiento permite acceder a bytes en posiciones impares (seleccionado) o requiere que siempre se pidan datos a partir de posiciones pares.

  - **Número de PLC**: número unívoco para identificarlo entre todos los PLCs. Este número debe ser el mismo que se le asigne en la configuración de MecaluxNet.

- **Cancelaciones**, permite definir la zona del PLC que se utilizará para comunicar con Galileo las cancelaciones de órdenes. A diferencia de las máquinas normales, que gestionan cada una sus propias cancelaciones, las máquinas PLC delegan está operación en el propio PLC. De esta manera se evita definir una zona para las cancelaciones para cada máquina, definiéndose únicamente para cada PLC.
  - **Cancelaciones activas**, seleccionado permite configurar:
    - **Bloque de datos**: DB en el que se definirá la zona de cancelación.
    - **Byte**: byte en el que se definirá la zona de cancelación.

 Haga clic en  para guardar los cambios en el proyecto.

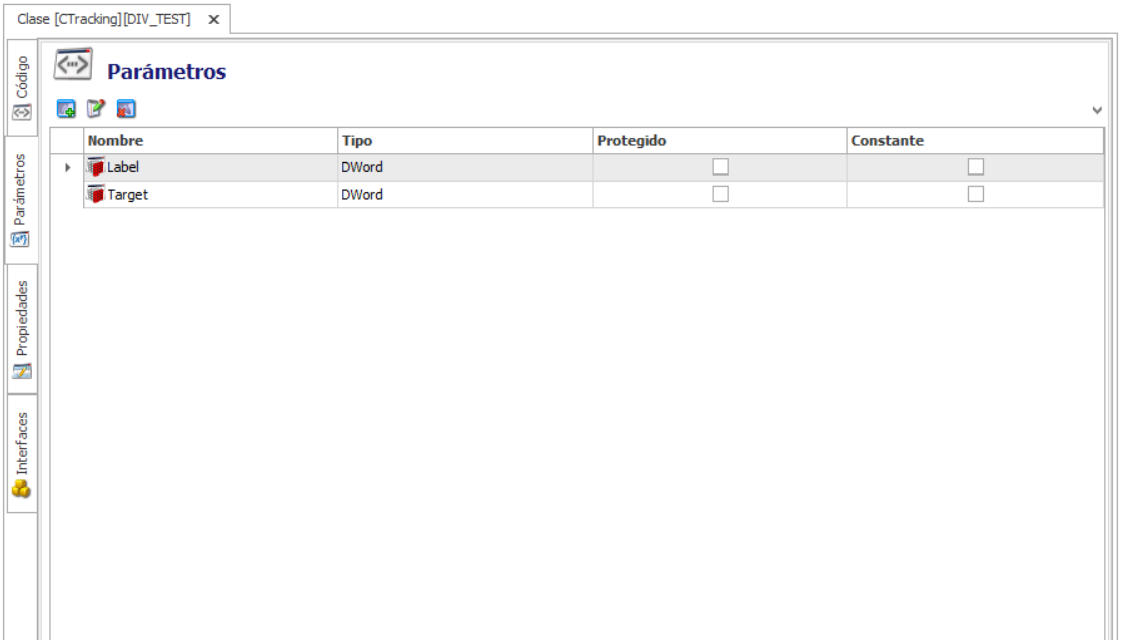
4.8.- PANTALLA "CLASES"

4.8.1.- ¿QUÉ ES UNA CLASE?

Una Clase es una declaración formal de un determinado tipo de datos independiente de la implementación de las funcionalidades, es decir, es una categoría de objetos con propiedades comunes.

En versiones anteriores del Designer IV, esta pantalla estaba limitada solo a la definición de UDTs. Un **UDT** es una clase que define conjunto de variables, de cualquier tipo definido, que se referencia bajo un único nombre, proporcionando un medio eficaz de mantener junta la información relacionada.

Gracias al nuevo lenguaje de programación XanaO2, puede definir estructuras de datos que posteriormente sean utilizadas por las máquinas creadas en el proyecto.



En la zona izquierda de la pantalla se muestran 4 pestañas por cada clase abierta:

- Pestaña **Código**: Creación, eliminación y edición de las funciones con su respectivo código.
- Pestaña **Parámetros**: Creación, eliminación y edición de los parámetros de la clase.
- Pestaña **Propiedades**: descripción de la clase.
- Pestaña **Interfaces**: establece las relaciones con interfaces previamente definidos.

Haga clic en para habilitar todas las funcionalidades y bloquear la pantalla en el proyecto para el resto de usuarios.

4.8.1.1.- ¿CÓMO CREAR UNA CLASE?

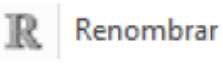
Haga clic con el botón derecho del ratón sobre el icono 

en el árbol del proyecto y seleccione desde el menú contextual la opción **Nuevo Clase**.

Las clases pueden compartir propiedades entre ellas, basándose en una relación de **herencia directa**. Para crear una clase hija haga clic con el botón derecho del ratón sobre una clase ya creado en el árbol del proyecto y seleccione la opción **Nueva hija**.

4.8.1.2.- ¿CÓMO EDITAR UNA CLASE?

Seleccione la clase en el árbol de proyecto y haga doble clic.

Puede renombrarla haciendo clic con el botón derecho, seleccionando desde el menú contextual la opción 

.

4.8.1.3.- ¿CÓMO ELIMINAR UNA CLASE?

Seleccione la clase en el árbol de proyecto, haga clic con el botón derecho y elija la opción 

desde el menú contextual.

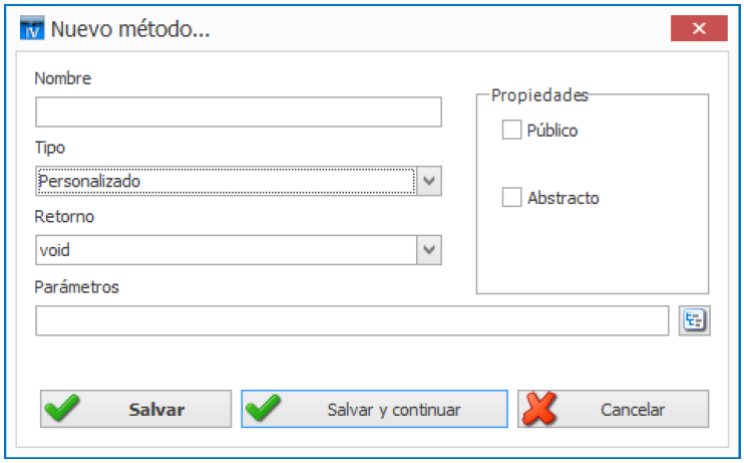
4.8.2.- PESTAÑAS DE LA PANTALLA "CLASES"

4.8.2.1.- PESTAÑA CÓDIGO

4.8.2.1.1.- ¿CÓMO CREAR FUNCIONES?

Haga clic en  en la barra de herramientas.

Rellene en la ventana los datos necesarios:



- **Nombre.**
- **Tipo** > **Personalizado**: función para ser invocada por el programador en cualquier parte del código. Seleccionando la opción **Abstracta**, la función no dispone de implementación, por lo que es obligatorio que la reimplementen sus clases derivadas.
- **Retorno:** desplegable con los tipos de retorno definidos para las funciones de tipo *Custom*.

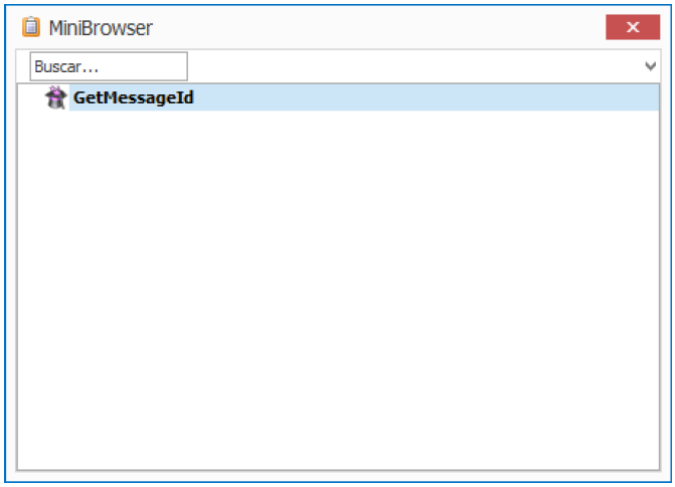
- **Parámetros**: parámetros de entrada definidos para las funciones de tipo *Personalizado (custom)*. Para validar los parámetros haga clic en  (más información en la página 75).

- **Público**: visibilidad del método. Deseleccionado este método es visible solo para el secuenciador y sus hijos.

4.8.2.1.2.- ¿CÓMO ABRIR Y EDITAR LAS FUNCIONES?

Para gestionar las funciones creadas en la clase haga clic en  .

La aplicación muestra el **MiniBrowser**, es una lista de las funciones definidas en la clase hasta el momento:

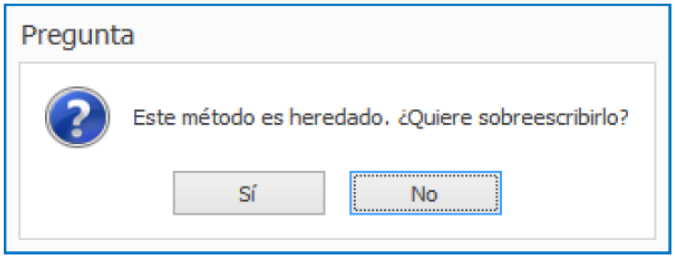


El MiniBrowser le ofrece distintas posibilidades desde el menú contextual:

-  Abrir: Abre las funciones en el editor de código. Permitida la multiselección. También se pueden abrir con doble clic.
-  Mover al padre: Elimina de la clase la función y la agrega al conjunto de funciones de la clase padre. Se emplea si consideramos que la función puede ser utilizada en cualquiera de sus "hermanos".
-  Eliminar: Si la función no está abierta en el editor de código, elimina la función de la clase y de todo el proyecto. La referencia a la función no desaparece del código pero ésto será detectado a la hora de compilar el proyecto.

 Una de las propiedades de las funciones, accesible desde el formulario de detalle al abrir cualquiera de ellas, es **Remota** e indica que la función puede ser consultada por distintos computadores pertenecientes al proyecto.

Al editar una función heredada, la aplicación le indica si desea abrirla en modo lectura o bien puede modificarla (sobrecarga de funciones), dejando de ser ésta *heredada* y se convierte en *propia* de la clase:



La **sobrecarga de funciones** consiste en editar funciones en una clase que ya han sido previamente declaradas en la clase padre (o alguno de sus ancestros).

Estas nuevas funciones, con el mismo prototipo que las de su padre, pueden contener código completamente diferente, y a su vez "ocultan" la visión de la función que sobrecargan para esa clase y para todos sus descendientes. Esto permite especializar los descendientes de una clase sin perder la claridad de concepto que ya se tenía al conocer la clase base.

4.8.2.1.3.- ¿CÓMO TRABAJAR CON EL EDITOR DE CÓDIGO?

Puede consultar "¿Cómo trabajar con el Editor de Código?" en la página 78.

4.8.2.1.4.- PESTAÑA PARÁMETROS

Los *parámetros* especifican el tipo básico de variable, pero no ninguno de sus modificadores. Es decir, podremos definir parámetros de tipos bit, byte, word, dword, qword, double, string, component, class, collection, vector o list, independientemente de si son de tipo Shared, PLC, de entradas o de salidas.

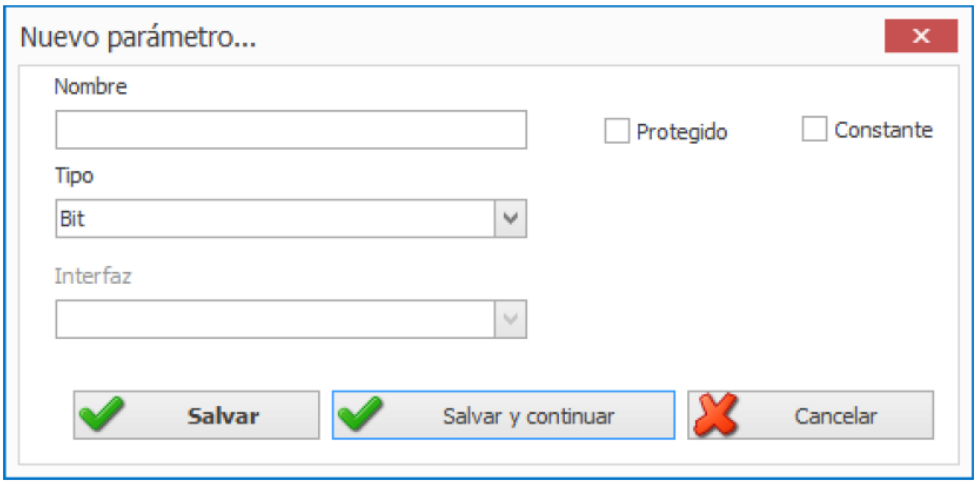
Los parámetros afectan a una instancia de la clase (y en el código de las funciones debe ser usados mediante la fórmula clase.nombre_parametro).

Los parámetros son heredables. Una clase hija tiene acceso a todos los parámetros definidos en el padre o uno de sus ancestros. Pero, al contrario que las funciones, los parámetros no se sobrescriben, es decir, no pueden cambiar de significado entre un padre y sus descendientes.

¿Cómo crear parámetros?

Haga clic en  en la barra de herramientas.

Rellene en la ventana los datos necesarios:



- **Nombre**
- **Tipo**: desplegable con los tipos definidos.
- **Protegido**: seleccionado indica que el parámetro es visible solo desde esta clase y sus hijas.
- **Constante:** seleccionado indica que el parámetro es de solo lectura, durante la compilación se mostrará un error si se intenta modificar su valor.

La aplicación muestra distintos iconos para los parámetros según el tipo definido. Cuando los parámetros son heredados se muestran con letra gris.

 Parámetro de tipo **Bit**

 Parámetro de tipo **Byte**

 Parámetro de tipo **Word**

 Parámetro de tipo **Dword**

 Parámetro de tipo **Qword**

 Parámetro de tipo **Double**

 Parámetro de tipo **String**

 Parámetro de tipo **Component**

 Parámetro de tipo **Class**

 Parámetro de tipo **Collection**

 Parámetro de tipo **Vector**

Parámetro de tipo **List**

4.8.2.1.5.- ¿CÓMO EDITAR PARÁMETROS?

Seleccione el parámetro en la tabla y haga clic en .

4.8.2.1.6.- PESTAÑA INTERFACES

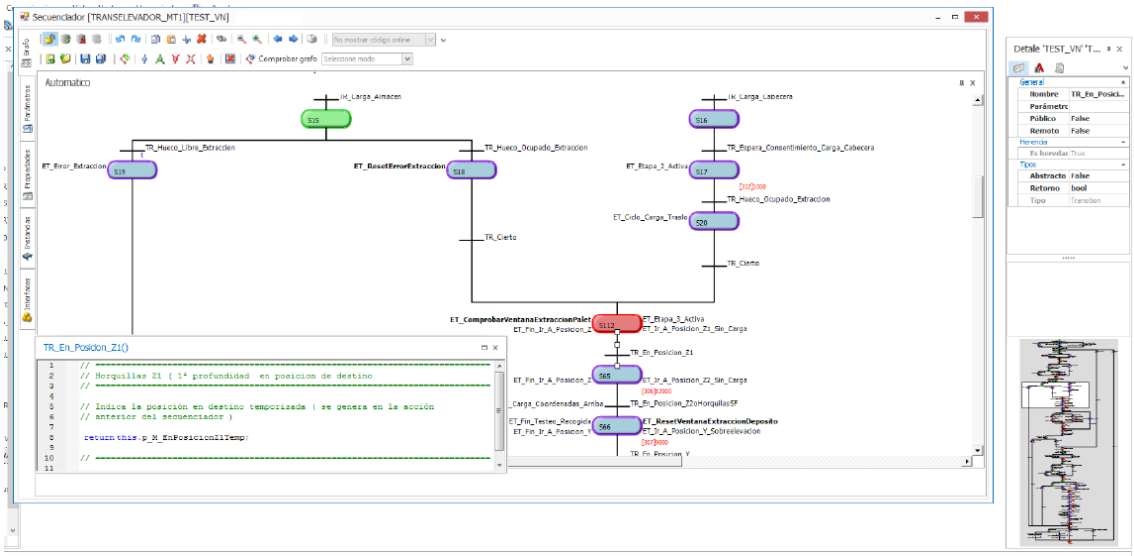
Haga clic en para asociar a esta clase un Interface.

De esta forma, la clase ha de implementar los métodos y/o propiedades definidos en el interface.

Haga clic en para guardar los cambios en el proyecto.

4.9.- PANTALLA "SECUENCIADORES"

En esta pantalla se definen los comportamientos de los diferentes elementos a controlar mediante la edición de los secuenciadores. Es por eso que su edición es sin duda el paso más importante del desarrollo por parte del usuario.



En la zona izquierda de la pantalla se muestran 5 pestañas por cada secuenciador abierto:

- Pestaña **Grafo**: Creación, eliminación y edición de las funciones con su respectivo código y los grafos de cada secuenciador.
- Pestaña **Parámetros**: Creación, eliminación y edición de los parámetros del secuenciador.
- Pestaña **Propiedades**: Descripción del secuenciador, zona para adjuntar ficheros como inis, indicación de las funciones de arranque y anterior/posterior al secuenciador, funciones de depuración y finalmente la asignación de tags (más información en Tags en la página 83)
- Pestaña **Instancias**: Máquinas asociadas a este secuenciador.
- Pestaña **Interfaces**: establece las relaciones con interfaces previamente definidos.

Como puede observar en la imagen de la pantalla, los paneles de la pestaña Grafo pueden moverse, desligarse e incluso visualizarlos en distintos monitores. Esta utilidad permite ver el grafo y código seleccionados simultáneamente.

Un *secuenciador* supone la descripción genérica de un elemento que sigue un patrón de comportamiento cíclico y variable. La potencia de este concepto estriba en definir un patrón que con un único código se podrá aprovechar para definir el comportamiento de todas aquellas máquinas que sigan la misma pauta.

Haga clic en para habilitar todas las funcionalidades y bloquear la pantalla en el proyecto para el resto de usuarios.

4.9.1.- ¿CÓMO CREAR UN SECUENCIADOR?

Haga clic con el botón derecho del ratón sobre el icono Secuenciadores en el árbol del proyecto y seleccione desde el menú contextual la opción **Nuevo Secuenciador**.

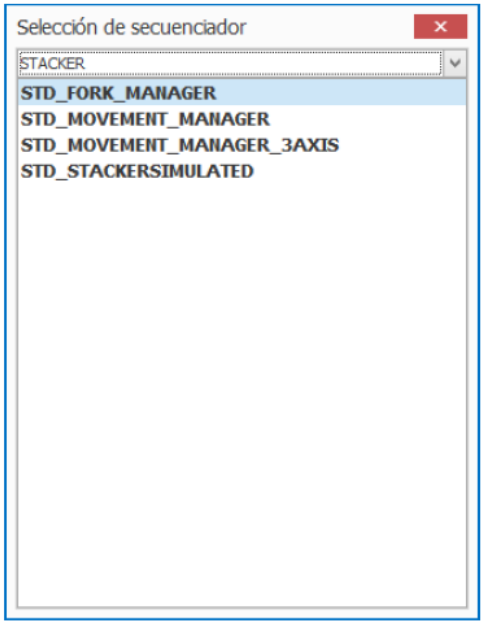
Los secuenciadores pueden compartir propiedades entre ellos, basándose en una relación de **herencia directa**. Para crear un secuenciador hijo haga clic con el botón derecho del ratón sobre un secuenciador ya creado en el árbol del proyecto y seleccione la opción del menú contextual **Nuevo hijo**.

4.9.1.1.- ¿CÓMO BUSCAR UN SECUENCIADOR CREADO (TAGS)?

Si bien puede navegar por el árbol de proyecto sobre los secuenciadores siguiendo la jerarquía entre los mismos, la aplicación le ofrece la posibilidad de realizar una búsqueda de secuenciadores mediante tags, es decir, palabras claves asociadas por el usuario al secuenciador.

Para ello haga clic con el botón derecho del ratón sobre el icono y seleccione desde el menú contextual la opción **Selección de Secuenciador**.

En la ventana seleccione mediante el desplegable el Tag para mostrar aquellos secuenciadores que lo tengan configurado.



4.9.1.2.- ¿CÓMO EDITAR UN SECUENCIADOR?

Seleccione el secuenciador en el árbol de proyecto y haga doble clic.

4.9.1.3.- ¿CÓMO ELIMINAR UN SECUENCIADOR?

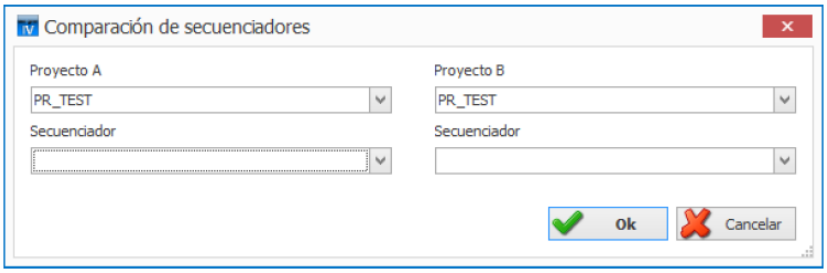
Seleccione el secuenciador en el árbol de proyecto, haga clic con el botón derecho y elija la opción 

desde el menú contextual.

4.9.1.4.- ¿CÓMO COMPARAR SECUENCIADORES?

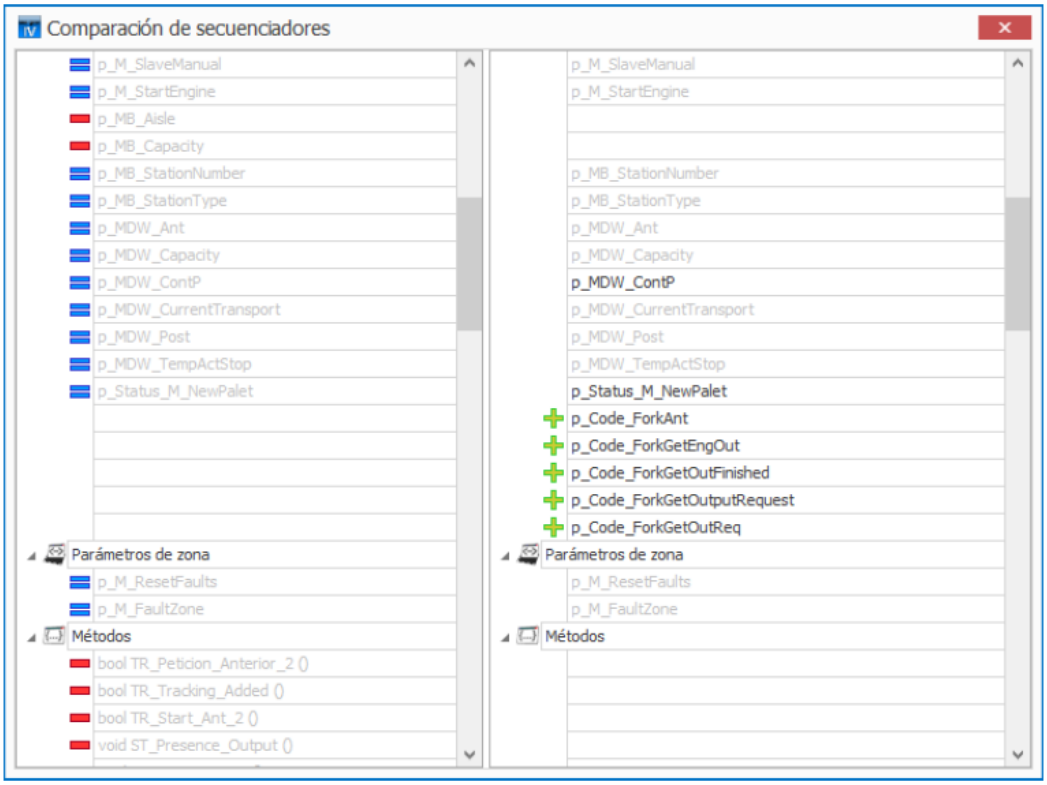
Designer IV permite realizar una comparación de secuenciadores, bien sean del mismo proyecto o de proyectos distintos.

Esta opción es accesible desde el menú **Herramientas** > **Comparación de secuenciadores**.



Seleccione el proyecto y secuenciador a comparar. La aplicación realiza una comparación de: **versión de secuenciador**, **versión de ingeniería**, **parámetros**, **parámetros de zona**, **métodos**, **interfaces**, **modos de funcionamiento** y **propiedades**.

En la siguiente captura mostramos un ejemplo del resultado:



Donde:

- Indica que en ambos secuenciadores el elemento a comparar, en este caso un parámetro, es idéntico.

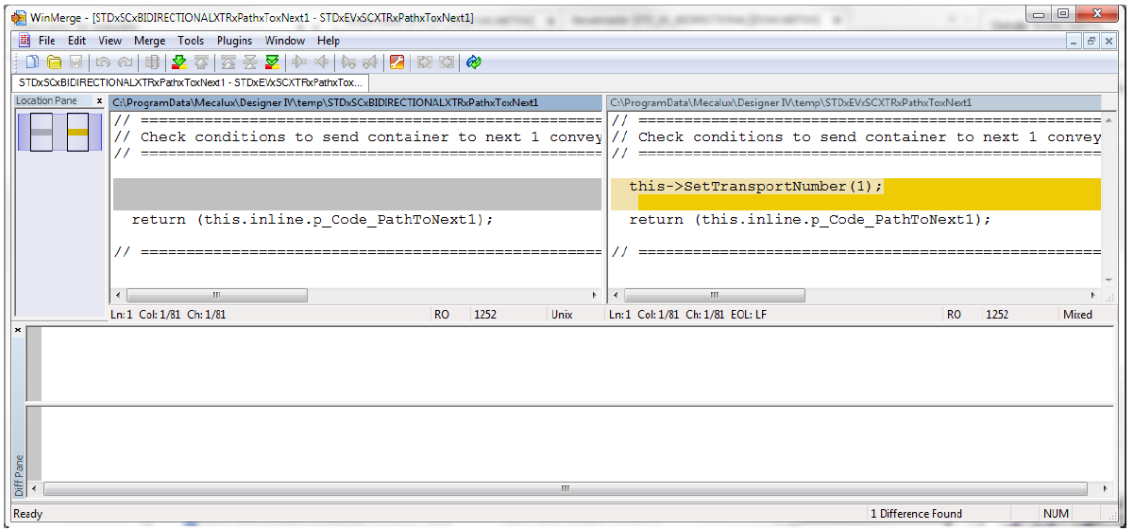
- Indica que el secuenciador original no tiene un elemento, en este caso un parámetro, añadido en el secuenciador a comparar.

- Indica que el secuenciador original tiene un elemento, en este caso un parámetro, que no está presente en el secuenciador a comparar.

– Indica que ambos secuenciadores tienen definido un elemento, en este caso un método, pero éste es distinto.
  - En los **métodos** se comprueba para compararlos el nombre, parámetros de entrada, tipo de retorno y diferencias en el código.

En los métodos es posible **comparar su código**. Para ello haga clic derecho sobre uno de ellos y seleccione la opción 

. Designer IV muestra una herramienta donde se indica de forma clara las diferencias entre ambos métodos.



  - En los **parámetros** se comprueba para compararlos el nombre y el tipo.

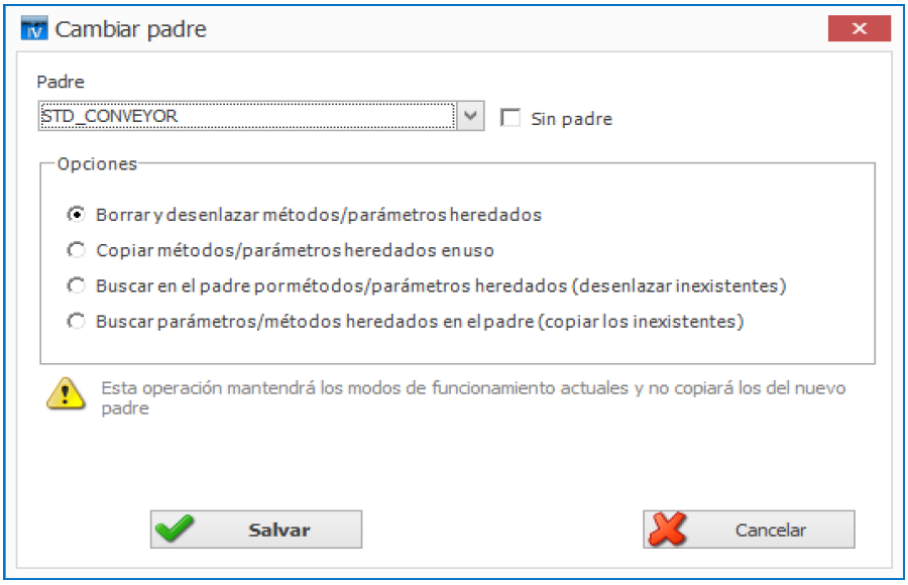
– Indica mediante el color del texto que, el parámetro del secuenciador original se trata de un parámetro heredado (gris claro), y del secuenciador a comparar es un parámetro propio del secuenciador (negro). En ambos casos la comparación los detecta como idénticos ya que tienen el nombre y tipo iguales.

4.9.1.5.- **OTRAS OPCIONES**

Seleccione el secuenciador en el árbol de proyecto, haga clic con el botón derecho mostrándose el menú contextual con las siguientes opciones:

 Nuevo hijo : crea secuenciador hijo. Esta opción copia todo el secuenciador padre.

 Cambiar padre : permite relacionar secuenciadores entre sí, dando lugar a la herencia de funciones y parámetros. El nodo del secuenciador en el árbol de proyectos se desplaza para reflejar el cambio en la jerarquía realizada.



 Esta opción mantiene los modos de funcionamiento actuales y no copia los del padre. Si queremos los modos del funcionamiento del padre hemos de utilizar la opción  Nuevo hijo.

Seleccionaremos en el desplegable "**Padre**" el secuenciador padre. En caso de cambiar a un secuenciador base, seleccionar el check "**Sin padre**":

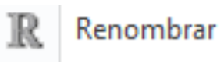
Al cambiar el secuenciador padre, se pueden perder los métodos y parámetros heredados del anterior. Para evitar pérdida de funcionalidad indique con las opciones propuestas como tratar los métodos y parámetros anteriormente heredados.

Las opciones disponibles son:

1. Ignorar todos los parámetros/métodos y copiar solo los parámetros/métodos propios. Esta opción es la más sencilla, es un cambio de padre sin más. Las variables enlazadas a parámetros heredados quedarán desenlazadas.

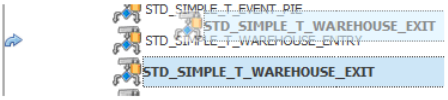
2. Añade a este secuenciador todos los parámetros/métodos del padre en uso. Esta opción solo se selecciona junto a la opción “Sin padre”.
3. Ignorar todos los parámetros/métodos y copiar solo los parámetros/ métodos propios pero además busca los heredados en el nuevo padre y si existen los mantiene. Es como la opción 1 pero en lugar de desenlazar, busca en el padre a ver si existe un parámetro con el mismo nombre.
4. Copiar todos los parámetros/métodos heredados salvo los que existan en el padre. Es como la opción 2 pero solo copia los que no existen en el padre y mantiene enlazados los que puede.

   En resumen:
   Las opciones 1 y 3 son similares. La opción 1 tiene más sentido cuando estamos eliminando el padre (aunque alguien podría querer utilizarla en otro caso) mientras que la 3 tiene más sentido cuando estamos cambiando el padre.
   Las opciones 2 y 4 son similares. La 2 solo tiene sentido al quitar el padre mientras que la 4 solo tiene sentido al cambiarlo.



El secuenciador seleccionado.

El secuenciador seleccionado.



Ctrl+clic ratón y arrastrar el secuenciador.

Permite reordenar los secuenciadores dentro del árbol, siempre dentro de su misma jerarquía.

4.9.2.- PESTAÑAS DE LA PANTALLA “SECUENCIADORES”

4.9.2.1.- PESTAÑA GRAFO

4.9.2.1.1.- ¿CÓMO CREAR UN MODO DE FUNCIONAMIENTO?

Un secuenciador puede tener varios *modos de funcionamiento*, con diferentes prioridades y selectores, que le permiten cambiar su forma de actuar en cuando cambien ciertas condiciones de entrada. Esta posibilidad de cambio le permite adaptarse a situaciones completamente distintas sin necesidad de complicar el grafo hasta niveles incompresibles.

Cuando crea un secuenciador, éste ya tiene un modo de funcionamiento por defecto (**Automatic**).

Para crear un nuevo modo de funcionamiento haga clic en  e indique el nombre y tipo:

En el formulario de detalle, una vez creado el modo, puede modificar el modo de funcionamiento "**Tipo Grafo**" a *Normal* o *Any*.

Los ***Modos de funcionamiento*** **de tipo Normal** llevan asociados un número llamado ***Selector*** que los identifica y una característica llamada ***Prioridad***.

- Debe de existir al menos un modo de funcionamiento cuyo **selector** sea 0 que es el *Modo de Funcionamiento por Defecto* y será el que esté activo si no se especifica otro desde código. Una vez creado el grafo de selector 0, no puede cambiarse dicho valor.
- En todos los modos de funcionamiento normales, la **prioridad** es 0. Los identificadores de selector no pueden repetirse entre los distintos modos de funcionamiento normales.

Además de los modos de funcionamiento normales, existen otros ***Modos de Funcionamiento por Interrupción*** (también llamados **ANY**):

- Su campo prioridad es un número mayor que 0. Dicho número es único e indica su orden de evaluación. Cuanta más alta es la prioridad, antes se evalúan.
- Su selector no es utilizado.
- Sus grafos no tienen Etapas iniciales o finales, ni tampoco transiciones finales. En su lugar, tienen ***Transiciones de Entrada ANY*** **y** ***Transiciones de Salida RETURN***.

 La diferencia fundamental con los *modos de funcionamiento normales* es que los *modos de funcionamiento por interrupción* no necesitan activarse por código, sino que se activan *automáticamente* en cuanto se cumple la condición de su *transición de Entrada ANY*. Asimismo, también dejan de estar activos en cuando se cumple su *transición de Salida RETURN*.

Para integrar ambos modos de funcionamiento (normal y de interrupción) inicialmente se evalúan todas las transiciones de entrada de todos los modos de funcionamiento (siempre por orden de prioridad).

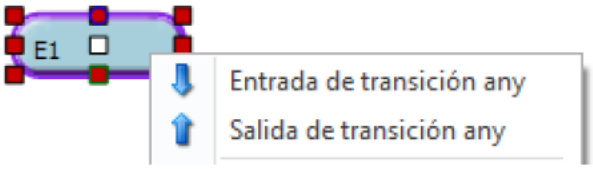
Si no se cumple ninguna de las transiciones se sigue ejecutando el modo de funcionamiento en el que se estaba. En caso de que alguna de las transiciones de entrada se cumpla:

1. Se detiene el modo de funcionamiento actual.
2. Se almacena el estado actual del modo de funcionamiento por selector.
3. Se entra en el modo de funcionamiento por interrupción.
   Dentro de este modo de funcionamiento, el grafo de prioridad se ejecuta normalmente, pero siempre con la salvedad que se siguen evaluando las transiciones de entrada de los modos de funcionamiento cuya prioridad sea más alta que la del actual. Si alguna de estas transiciones se activa, vuelve a producirse un cambio de modo de funcionamiento, con la excepción de que el estado del grafo no se salva en este caso, dado que los funcionamientos por interrupción no recuerdan su estado.

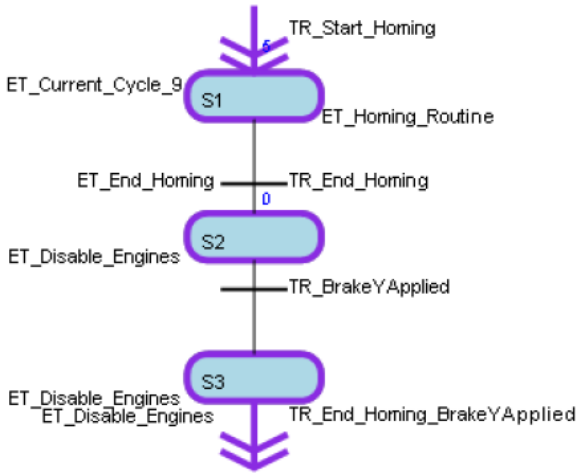
4.9.2.1.2.- ¿CÓMO CREAR TRANSICIONES DE ENTRADA ANY Y DE SALIDA RETURN?

Las transiciones de entrada y de salida de un modo de funcionamiento por interrupción, se representan por una flecha de doble cabeza.

Para crearlas, haga clic derecho sobre ellas para mostrar el menú contextual:

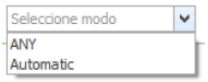


Seleccione el tipo de transición. Automáticamente el tipo de la etapa se convierte en INITIAL_STEP o FINAL_STEP como se explica en la página 68.



4.9.2.1.3.- ¿CÓMO EDITAR UN MODO DE FUNCIONAMIENTO?

Para modificar editar un Modo de Funcionamiento, selecciónelo en el desplegable



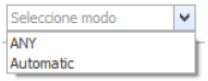
y modifique los campos necesarios en el formulario de detalle.

Los campos editables son:

- **Nombre.**
- **Prioridad**: si el Modo de funcionamiento es una ANY, es el orden de evaluación sobre las transiciones de entrada de todas las ANYs. Si es modo NORMAL su valor es 0.

- **Selector**: si es un modo de funcionamiento NORMAL (su prioridad es 0), es el valor que lo define. Este valor es modificado por código en función del modo que se necesite. Si es una ANY su valor es 0.
- **Tipo Grafo**: normal o por interrupción. Un grafo ANY no puede ser cambiado a NORMAL.

4.9.2.1.4.- ¿CÓMO ELIMINAR UN MODO DE FUNCIONAMIENTO?

Para eliminar un Modo de Funcionamiento, selecciónelo en el desplegable 

y haga clic en  .

 El modo de funcionamiento por defecto (selector y prioridad 0) no puede ser eliminado.

4.9.2.1.5.- ¿CÓMO CREAR UN ETAPAS Y TRANSICIONES?

Cada modo de funcionamiento tiene sus propias ***Etapas*** y ***Transiciones*** y se muestra como un grafo separado del resto.

Los elementos de los que consta un grafo son básicamente dos:

- **Etapas** → Estado que se está desarrollando.
- **Transiciones** → Condición para cambio entre etapas.

¿Cómo crear etapas en un grafo?

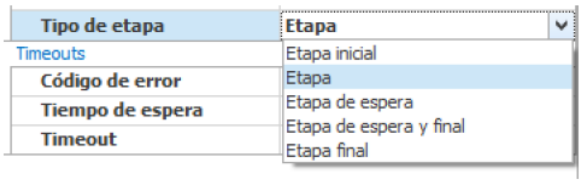
Una **Etapa** consiste en un estado en el que una acción o acciones se están desarrollando.

Las etapas se representan gráficamente como un rectángulo con bordes redondeados.

Para crear una etapa haga clic en  en función del número de transiciones de entrada/salida que necesite:

- A una etapa **PIPE** (de color azul) únicamente le entra una transición y le sale otra.
- Una etapa **FUENTE** (color verde) recibe una sola transición mientras que de ella puede salir más de una transición.
- En una etapa **SUMIDERO** (color rojo) puede entrar más de una transición pero únicamente puede salir una transición de la misma.
- Una etapa **CONCENTRADOR** (color marrón) es aquella que puede recibir más de una transición y de la también pueden salir varias transiciones.

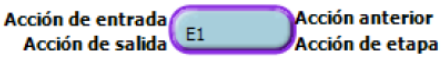
Las etapas pueden tener un atributo extra que les señala una característica especial, estos atributos los puede seleccionar en el formulario de detalle en el campo "**Tipo Etapa**", y son:



- **Etapa Inicial**: indica la etapa por defecto del grafo y a donde se reinicia éste cuando termine un ciclo o sea inicializada por el usuario.
- **Etapa**: Etapa normal, sin más características.
- **Etapa de espera**: La etapa no se activa hasta que se cumplen todas las transiciones que llegan a ella.
- **Etapa de espera y final**: Combinación de una etapa de espera que además es final. No se activa hasta que se cumplan todas las transiciones que llegan a ella, y al cumplirse su transición de salida, el grafo en ejecución regresa a la etapa definida como inicial.
- **Etapa Final**: Al salir de esta etapa, el grafo de ejecución salta de nuevo a la etapa definida como InitialStep.

Cada modo de funcionamiento debe de tener una y solo una etapa definida como inicial, y una y solo una etapa definida como final (puede ser una FinalStep o una WaitFinalStep).

Las etapas pueden tener hasta 4 acciones asociadas a ellas:



Estas acciones las puede seleccionar en el formulario de detalle, tras seleccionar la etapa, en los campos "**Acción de entrada**", "**Acción anterior**", "**Acción de etapa**" y "**Acción de salida**":

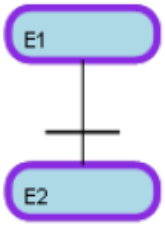
El significado y orden de ejecución de estas acciones se detalla tras explicar las transiciones.

4.9.2.1.6.- ¿CÓMO CREAR TRANSICIONES EN UN GRAFO?

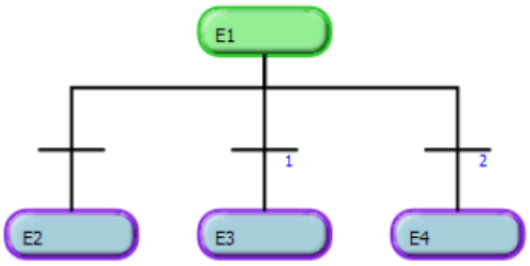
En una **Transición** se definen las condiciones necesarias para el paso de una etapa a otra.

Existen varios tipos de transiciones, veremos ejemplos de cada una de ellas y cómo realizarlas:

- **Sencilla**: Colóquese encima de E1 y haciendo clic en la etapa arrastre hasta la E2.

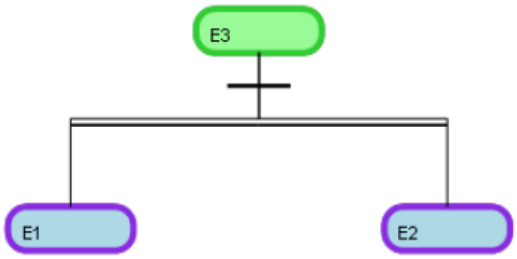


- **Rama divergente OR**: Colóquese encima de E1 (**Fuente**) y haciendo clic en la etapa arrastre hasta la E2. Repita la opción para la transición entre E1 y E3 y E1 y E4 (el orden no importa).

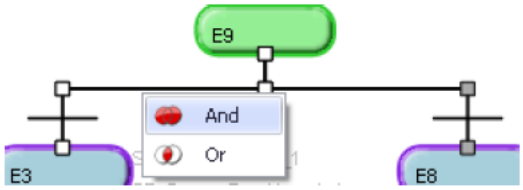


  En este caso se implementa una exclusión entre varias evoluciones posibles a partir de una misma etapa.

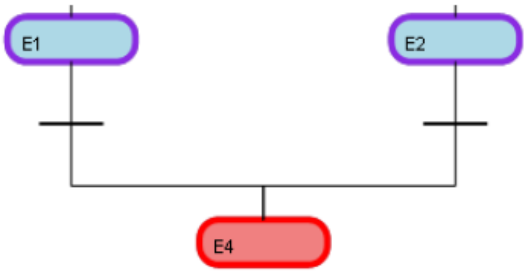
- **Rama divergente AND**: Colóquese encima de E1 (**Fuente**) y haciendo clic en la etapa arrastre hasta la E2. Repita la opción para la transición entre E1 y E3. Finalmente selecciones con el ratón ambas transiciones y teclee la letra **A**.
  Una vez que la transición sea validada, se ejecutarán de forma simultánea las ramas que une.



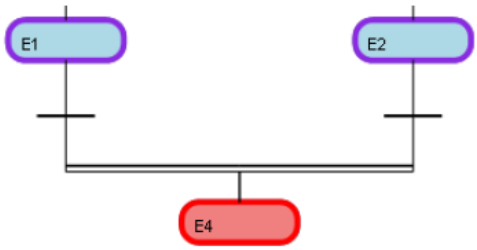
Seleccionando las transiciones deseadas, haga clic con el botón derecho para mostrar el menú contextual y convertirlas en ramas AND/OR directamente:

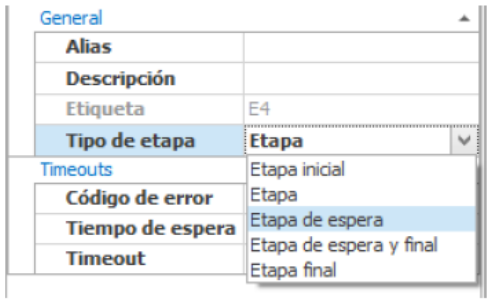


- **Rama convergente OR**: Colóquese encima de E4 (**Sumidero**) y haciendo clic en la etapa arrastre hasta la E2. Repita la opción para la transición entre E4 y E3.



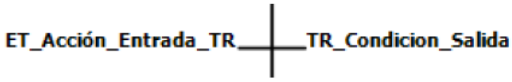
- **Rama convergente AND**: Colóquese encima de E4 (**Sumidero**) y haciendo clic en la etapa arrastre hasta la E2. Repita la opción para la transición entre E4 y E3. Seleccione el **Sumidero** y en el formulario de detalle cambie el campo **"Tipo de etapa"** a **Etapa de Espera.**

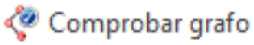




A cada transición va asociada una condición que puede ser evaluada a verdadero o falso, además, y de forma opcional, se le puede asociar una Acción de entrada. Esta Acción de entrada se ejecuta cuando la evaluación de la transición es verdadera.

Estas acciones las puede seleccionar en el formulario de detalle, tras seleccionar la transición, en los campos "**Acción Entrada**" y "**Condición Salida**":

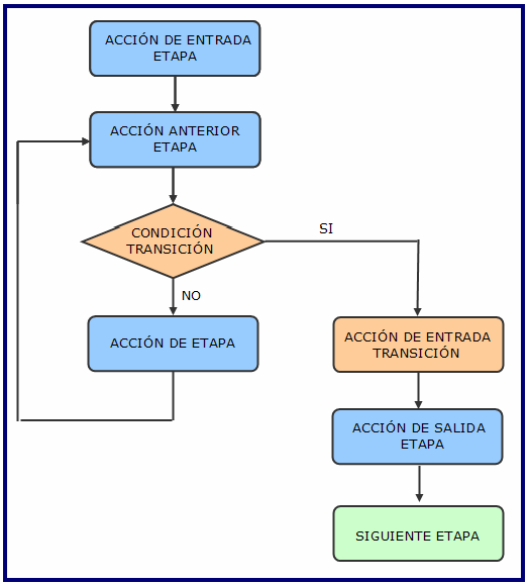


Haga clic en 

para realizar una comprobación de la construcción del grafo, es decir, si tiene creadas y configuradas correctamente los tipos de etapas y transiciones.

4.9.2.1.7.- ¿CUÁL ES EL ORDEN DE EJECUCIÓN DE LAS ETAPAS?

El orden de ejecución de las acciones de etapas y transiciones es:



- ***Acción de entrada de etapa***: Esta acción se ejecuta cuando la etapa en ejecución se activa, bien porque la condición de transición le da paso, bien porque se regresa de un modo de funcionamiento por interrupción (ANY). Su prototipo es un método que no retorna ningún tipo de valor (void) y no requiere de ningún parámetro para ser llamada.
- ***Acción de salida de etapa***: Esta acción se ejecuta cuando la etapa deja paso a la ejecución de cualquier otra etapa, ya sea porque se cumple la transición de salida o porque se cumple la condición de entrada de un modo de funcionamiento por interrupción. Su prototipo es un método que no retorna ningún tipo de valor (void) y no requiere de ningún parámetro para ser llamada.
- ***Acción anterior de etapa***: En cada ciclo de ejecución de Galileo IV, esta acción es llamada antes de evaluar la condición de transición. Su prototipo es un método que no retorna ningún tipo de valor (void) y no requiere de ningún parámetro para ser llamada.
- ***Acción de etapa***: En cada ciclo de ejecución de Galileo IV, esta acción es llamada después de evaluar la condición de transición, siempre que esta última haya

resultado negativa, pues en otro caso, no se llegara a ejecutar. Su prototipo es un método que no retorna ningún tipo de valor (void) y no requiere de ningún parámetro para ser llamada.

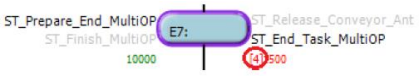
- ***Acción de entrada de transición***: Esta acción es invocada una sola vez por Galileo IV justo después de invocar la condición de transición cuando esta se cumple afirmativamente, y antes de invocar a la acción de salida de la etapa. Su prototipo es un método que no retorna ningún tipo de valor (void) y no requiere de ningún parámetro para ser llamada.
- ***Condición de salida de transición***: Esta es acción que se evalúa en cada ciclo de ejecución. Su prototipo es un método que retorna un booleano indicando si se debe cambiar de etapa (valor true) o si no es así (valor false). No debe llevar parámetros de ningún tipo.

4.9.2.1.8.- ¿CÓMO EDITAR LAS ETAPAS Y TRANSICIONES?

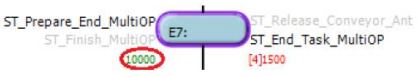
Para editar una etapa o transición, selecciónela en el grafo y modifique los campos necesarios en el formulario de detalle.

Los campos editables son:

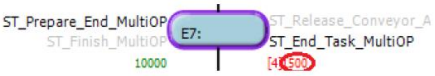
- **Etapas**
  - **Alias**: nemotécnico que podemos asignar a una etapa para que pueda ser referida posteriormente desde el *Editor de Tracking*.
  - **Descripción**: información reflejada solo en este campo.
  - **Etiqueta**: número de etapa.
  - **Tipo etapa**: más información en la página 68.
  - **Código de error**: error no bloqueante, el programador puede programar un modo de funcionamiento por interrupción para detectar este caso. Complementario al campo "**Timeout**".



  - **Tiempo de Espera**: introduce un periodo mínimo de ejecución de la etapa. Si la transición de salida se cumple, no se avanzará a la próxima etapa hasta que la actual haya consumido al menos esetiempo (medido en milisegundos).



  - **Timeout**: tiempo máximo de ejecución de la etapa en milisegundos.
    Si transcurrido ese tiempo la etapa aún no ha cambiado, se disparará un error cuyo número será el indicado en el campo "**Código Error**".

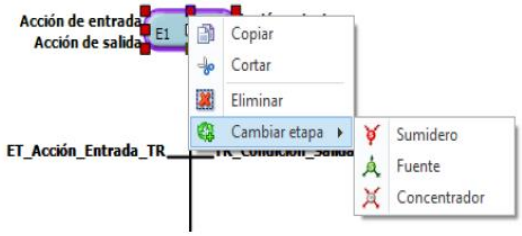


- **Transiciones.**
  - **Acción Entrada/Condición de Salida**: más información en la página 71.
  - **Prioridad**: orden de evaluación en el conjunto formado por la transición y sus transiciones hermanas. Por ejemplo en una divergencia en OR, todas las transiciones hijas de una etapa son evaluadas en el orden especificado por esta prioridad.

A mayor valor de la prioridad, antes se evalúa. Si una de las transiciones se cumple, el resto ya no se siguen evaluando (puesto que implica ya un cambio de etapa).

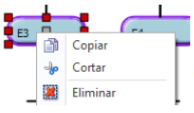
Puede **mover** etapas seleccionándolas (multiselección o ctrl+clic) y arrastrándolas. La aplicación le indica los movimientos no permitidos.

También puede **cambiar el tipo de una etapa**, siempre que cumpla unas condiciones por diseño y que será la propia aplicación quién las valide, mostrando el menú contextual sobre la etapa:



4.9.2.1.9.- ¿CÓMO ELIMINAR ETAPAS Y TRANSICIONES?

Para eliminar etapas selecciónelas, haga clic en  o en la tecla ‘Supr’.

También puede eliminarlas desde el menú contextual 

.

4.9.2.1.10.- OTRAS OPCIONES DISPONIBLES PARA LOS GRAFOS

 Deshacer, rehacer.

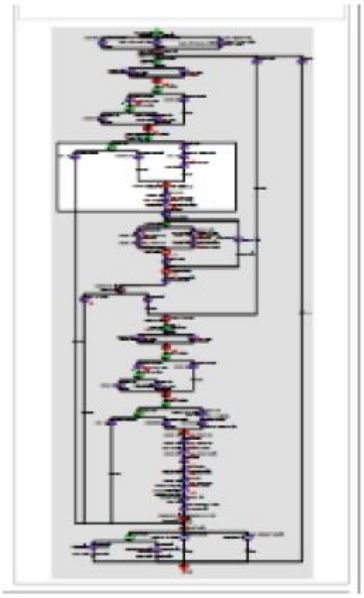
 Copiar, pegar y cortar.

 Elimina el modo de funcionamiento mostrado.

 Zoom.

 Imprime en formato pdf el grafo mostrado.

 Buscar texto en código (más información en "¿Cómo realizar búsquedas en los proyectos en Designer IV?" en la página 43).

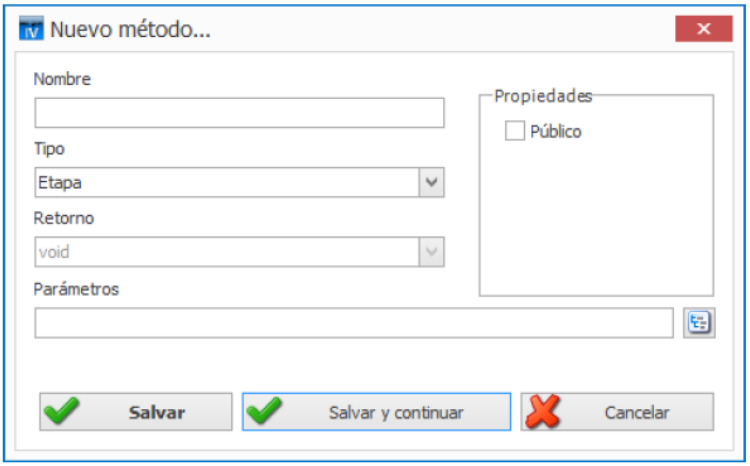


Panel de Navegación: Permite visualizar el grafo completo y moverse por él.

4.9.2.1.11.- ¿CÓMO CREAR MÉTODOS?

Haga clic en  en la barra de herramientas.

Rellene en la ventana los datos necesarios:

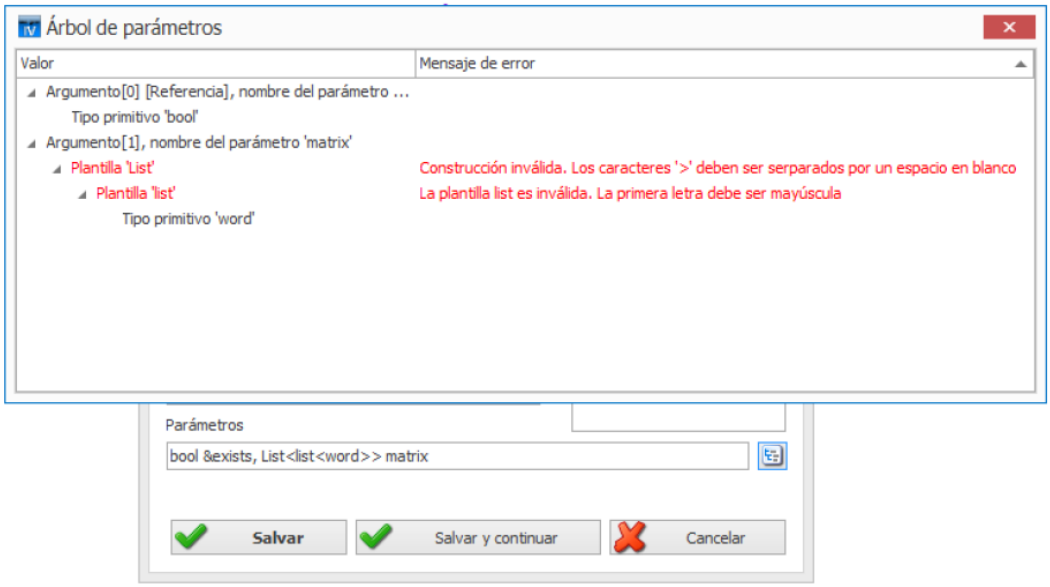


- **Nombre**
- **Tipo**: desplegable con los tipos definidos:
  - **Etapa**: método para asociar a una de las acciones de cualquier etapa del secuenciador o bien a las acciones de arranque, anterior y posterior. También puede ser asociado a la acción de entrada de una transición. *No retorna nada, no tiene parámetros de entrada.*
  - **Transición**: método para asociar en las condición de salida de las transiciones del secuenciador. *Retorna siempre un valor booleano. No tiene parámetros de entrada.*
  - **Personalizado**: método para ser invocado por el programador en cualquier parte del código. Seleccionada la opción **Abstracto**, el método no dispone de implementación, por lo que es obligatorio que lo reimplementen sus clases derivadas.
- **Retorno:** desplegable con los tipos de retorno definidos para los métodos de tipo *Personalizado*.
- **Parámetros**: parámetros de entrada definidos para los métodos de tipo *Custom*.

Desginer IV dispone de un **parser**, es decir, de un analizador sintáctico que le indica si tiene errores en los parámetros introducidos.

Para hacer la validación de los datos introducidos haga clic en .

En la siguiente captura tiene un ejemplo:



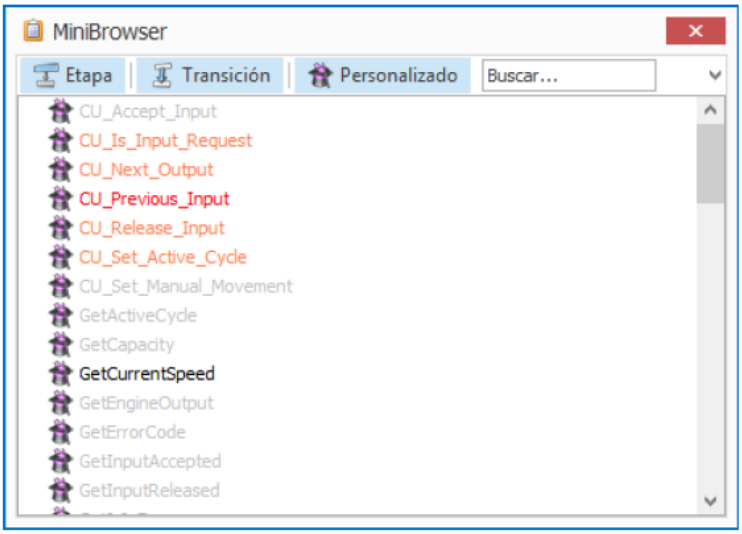
- **Público**: visibilidad del método. Si no se marca ese método es visible solo para el secuenciador y sus hijos.

  Para los métodos que implementen **Interfaces** siempre ha de marcarse esta opción.

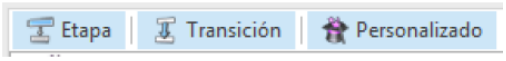
4.9.2.1.12.- ¿CÓMO ABRIR Y EDITAR LOS MÉTODOS?

Para gestionar los métodos creados en el proyecto haga en .

La aplicación muestra el **MiniBrowser**, es una lista con los prototipos XANA de los métodos definidos en el secuenciador hasta el momento:



El MiniBrowser le ofrece distintas posibilidades, bien sea en la misma ventana o desde el menú contextual:

- **Filtrado**:
  - 

: filtra por el tipo de métodos creados.
  - 

: filtrado según complete el campo.

-  Abrir : abre los métodos en el editor de código. Permitida la multiselección. También se pueden abrir con doble clic.

-  Mover al padre : elimina del secuenciador el método, y lo agrega al conjunto de métodos del secuenciador padre. Se emplea si consideramos que el método puede ser utilizado en cualquiera de sus "hermanos".

-  Eliminar : Si el método no está abierto en el editor de código, lo elimina del secuenciador y de todo el proyecto. El método desaparece de todas las etapas o transiciones a las que pudiera estar asociado, sin embargo, las referencias a él desde el código no serán eliminadas.

Los métodos definidos llevan asociados iconos y el nombre en distintos colores que proporcionan la siguiente información:

 Método de tipo **Etapa**

 Método de tipo **Transición**

 Método de tipo **Personalizado (custom)**

 Método de tipo **Personalizado y abstracto**

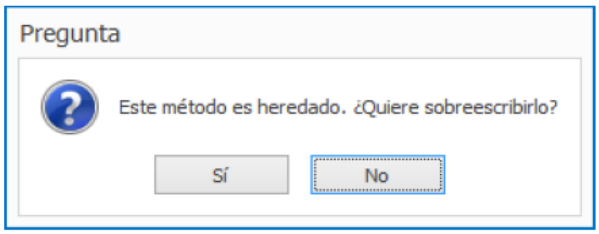
	Propio	Heredado	Sobrescrito
Método **Público**	**ST_Pb_Pr**	ST_Pb_Her	ST_Pb_Hr_Sbrscr *
Método **Privado**	**ST_Prv_Pr**	ST_Prv_Hr	ST_Prv_Hr_Sbrscr *

*→ Método **sobrescrito** (método heredado que ha sido sobrescrito)

 Una de las propiedades de los métodos, accesible desde el formulario de detalle al abrir cualquiera de ellos, es **Remoto** e indica que el método puede ser consultado por distintos computadores pertenecientes al proyecto.

La **sobrescritura de métodos** consiste en editar métodos en un secuenciador que ya han sido previamente declarados en el secuenciador padre (o alguno de sus ancestros). Estos nuevos métodos, con el mismo prototipo que los de su padre, pueden contener código completamente diferente, y a su vez "ocultan" la visión del método que sobrescriben para ese secuenciador y para todos sus descendientes. Esto permite especializar los descendientes de un secuenciador sin perder la claridad de concepto que ya se tenía al conocer al secuenciador base.

Al editar un método heredado, la aplicación le indica si desea abrirlo en modo lectura o bien puede modificarlo (sobrescribirlo), dejando de ser éste *heredado* y se convierte en *propio* del secuenciador:



Si la pantalla no está desbloqueada ( ) y se abre cualquier método heredado la pregunta anterior no se realiza. Si posteriormente la pantalla se desbloquea, todos los métodos heredados ya abiertos muestran el icono . Haciendo clic en él permite sobrescribir el método del padre mostrando la pregunta anterior.

4.9.2.1.13.- ¿CÓMO TRABAJAR CON EL EDITOR DE CÓDIGO?

El lenguaje de control que el usuario programa para efectuar el control de las máquinas en la instalación se basa en el código XanaO2. Este código se divide en métodos que después se invocan desde las acciones de etapa o transición.

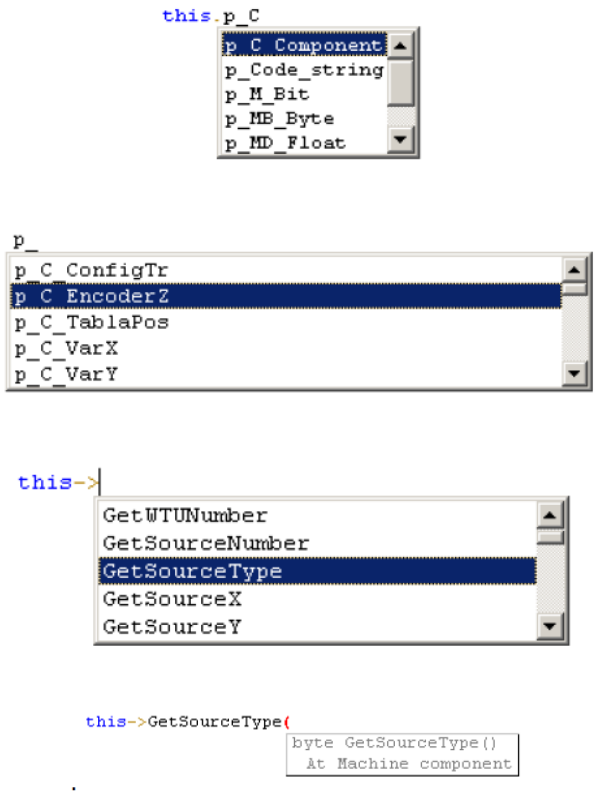
Para editar este código de la forma más cómoda posible, la aplicación incorpora una pantalla con un entorno que ofrece sintaxis resaltada (la del lenguaje XanaO2) y otras opciones.

```
CODE_1()
1   // ------------------------------------------------------------------------------
2   // Lift movement to go to Z task position without tray
3   // ==============================================================================
4   // Check load tray in the lift.
5
6     this.p_M_CheckLoadTray    = true;
7
8   // Disable check claws position
9
10    this.p_M_CheckClawsPosition = false;
11
12  // MOTION SEQUENCE
13  // 1°- Inverter enable -> activate engine power
14  // 2°- Engine selection OK -> movement
15
16    if (
17        this.p_C_Inverter->IsControlInhibit()
18        &&
19        !this.p_E_ExtractorPower
20        &&
21        !this.p_E_LiftPower
22        &&
23        !this.p_E_GatePowerPK01
24        &&
25        !this.p_E_GatePowerPK02
26       )
27    {
28     this.p_S_ExtractorEngine = true;
29    }
30
31    if (this.p_S_ExtractorEngine
32        &&
33        (this.p_MB_EngineSelected != 1))
34    {
35     this.p_C_Inverter->SelectEngine(1);
36    }
CODE_1()   TR_Transition()
```

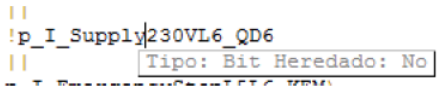
4.9.2.1.14.- COMPONENTE DE AYUDA A LA PROGRAMACIÓN

El editor de código dispone del sistema que, además de completar el resultado a medida que se teclea un parámetro por ejemplo, filtra por los métodos de cada uno de los componentes y sirve como documentación de los mismos.

Mostramos algunos ejemplos:



La aplicación también muestra, al clicar sobre un parámetro, el tipo del mismo y si es o no heredado:



4.9.2.1.15.- OTRAS OPCIONES DISPONIBLES EN EL EDITOR DE CÓDIGO

En la barra de herramientas tenemos:

 Deshacer, rehacer.

 Copiar, pegar y cortar.

 Elimina el método mostrado en el secuenciador de las etapas y transiciones y también en todos sus hijos.

 Buscar. Muestra la ventana “Buscar y reemplazar” con distintas opciones muy útiles.

 Zoom.

 Imprime en formato pdf el método mostrado.

 Guarda los cambios del método mostrado o de todos los abiertos respectivamente.

 Facilita la navegación por el código, pudiendo acceder al punto donde teníamos el cursor con anterioridad o posterioridad.

Haciendo clic derecho del ratón sobre el código se muestra un **menú contextual** con varias de las opciones anteriores y otras nuevas:

- **Ir a definición**: Con esta opción, y teniendo el cursor sobre un método Custom, la aplicación lo abre automáticamente y muestra el código de la mismo.
- **Ir a línea**: La aplicación abre una ventana dónde puede indicar a qué número de línea desea ir.
- **Comentar líneas**: Teniendo seleccionadas una o varias líneas de código y con esta opción, automáticamente les añade al principio de la línea //. También tenemos la posibilidad de comentar varias líneas utilizando **/*** y para cerrar el comentario multilínea ***/**.

4.9.2.2.- **PESTAÑA PARÁMETROS**

Los *parámetros* especifican el tipo básico de variable, pero no ninguno de sus modificadores. Es decir, podremos definir parámetros de tipos bit, byte, word, dword, qword, double, string, component,class, collection, vector o list, independientemente de si son de tipo Shared, PLC, de entradas o de salidas.

Los parámetros se dividen en dos categorías:

- **Parámetros normales**: afectan a una instancia del secuenciador (y en el código de los métodos deben ser usados mediante la fórmula **this**.nombre_parametro o nombre_parametro).
- **Parámetros de zona**: afectan a varias instancias del secuenciador a la vez (en el código deberían aparecer como **this.zone**.nombre_parametro_zona o nombre_parametro_zona). El concepto de zona es explicado en otro apartado de este documento.

 Una correcta realización de un secuenciador implica que en su código no existen referencias a variables del simbólico, sino a sus parámetros. En caso contrario el proyecto no compilaría.

Los parámetros son heredables. Un secuenciador hijo tiene acceso a todos los parámetros definidos en el padre o uno de sus ancestros. Pero, al contrario que los métodos, los parámetros no se sobrecargan, es decir, no pueden cambiar de significado entre un padre y sus descendientes.

4.9.2.2.1.- ¿CÓMO CREAR PARÁMETROS?

Haga clic en  en la barra de herramientas, diferenciando si se trata de un parámetro de zona o no.

Rellene en la ventana los datos necesarios:

- **Nombre**
- **Tipo**: desplegable con los tipos definidos.
  Las señales de tipo bit tienen además las opciones (heredables):
  - **Manual**: sin uso actualmente (era utilizada desde el Pupitre Virtual para movimiento en manual).
  - **Avería**: sin uso actualmente (era utilizada desde el Pupitre Virtual para activar fallo).
- **Protegido**: seleccionado indica que el parámetro es visible solo desde este secuenciador y sus hijos.
- **Constante:** seleccionado indica que el parámetro es de solo lectura, durante la compilación se mostrará un error si se intenta modificar su valor.

La aplicación muestra distintos iconos para los parámetros según el tipo definido. Cuando los parámetros son heredados se muestran con letra gris.

 Parámetro de tipo **Bit**

 Parámetro de tipo **Byte**

 Parámetro de tipo **Word**

 Parámetro de tipo **Dword**

 Parámetro de tipo **Qword**

 Parámetro de tipo **Double**

 Parámetro de tipo **String**

 Parámetro de tipo **Component**

 Parámetro de tipo **Class**

 Parámetro de tipo **Collection**

 Parámetro de tipo **Vector**

 Parámetro de tipo **List**

4.9.2.2.2.- ¿CÓMO EDITAR PARÁMETROS?

Seleccione el parámetro en la tabla y haga clic en  .

4.9.2.3.- PESTAÑA PROPIEDADES

La pestaña se divide en 4 zonas diferenciadas:

- **Descripción**: texto descriptivo del secuenciador.
- **Archivos:** zona para adjuntar archivos a un secuenciador. Únicamente ejemplos o plantillas de archivos necesarios en el secuenciador para ejecutar en una máquina (.ini por ejemplo).

  Haga clic en  para adjuntar, editar o eliminar los archivos respectivamente.

  Haga clic en  para extraer el archivo del proyecto a un fichero real, que más tarde podrá ser utilizado como base en las máquinas que lo necesiten.

   La utilidad de esta funcionalidad no es la permitir que en el propio archivo .mgp estén todos los archivos necesarios para poder ejecutarlo, sino que existan, al menos, un ejemplo de los archivos de configuración.

- **Métodos**: acciones que actúan siempre sobre el secuenciador, independientemente del modo de funcionamiento activo:

- **Acción Anterior**: acción que se ejecuta antes de evaluar el estado activo del secuenciador.
- **Acción Posterior**: acción que se ejecuta después de evaluar el estado activo del secuenciador.
- **Arranque Frío**: método invocado cuando Galileo IV inicia un arranque frío (inicio del sistema). Típicamente se hacen las inicializaciones en esta acción.
- **Arranque Caliente**: Idéntica a la anterior, pero para los arranques en caliente (reinicios de Galileo IV que no implican una parada del sistema). Su cometido es muy similar a la anterior y puede usarse en los mismos casos, aunque suele realizar una inicialización más ligera que aquella.
- **Simulación**: acción que se ejecuta cuando trabajemos en modo simulación (generalmente invocará, entre otras acciones, al método de Galileo SimulatedSignal del componente Machine para la activación/desactivación de señales en función de ciertas condiciones mantenidas con el tiempo.).

- **Tags**: zona para asociar palabras claves al secuenciador y posteriormente poder realizar una búsqueda de los secuenciadores en función de ellas. Más información en "¿Cómo buscar un secuenciador creado (Tags)?" en la página 59.

  Para añadir o eliminar tags haga clic en los iconos  y  respectivamente.

- **Versión del secuenciador**: Indica el número de versión del secuenciador. Si es un secuenciador estándar este campo no es editable.
- **Versión de ingeniería**: Indica el número de versión de la ingeniería eléctrica asociada al secuenciador.
- **Secuenciador de repositorio:** Opción visible solo si el usuario dispone de Licencia de I+D (más información en "Licencia Designer IV" en la página 11). Seleccionado indica que dicho secuenciador es estándar y no podrá ser modificado por usuarios con Licencia de Desarrollado.

 Los secuenciadores de repositorio tiene el icono  .

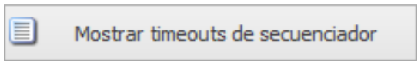
4.9.2.4.- **PESTAÑA INSTANCIAS**

Listado de todas las máquinas del proyecto que usan ese secuenciador como modelo.

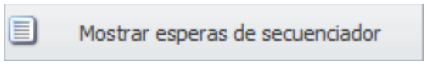
4.9.2.4.1.- SOBRECARGA DE TIMEOUTS O TIEMPOS DE ESPERA

También existe, para cada máquina, la posibilidad de sobrecargar los timeouts y tiempos de espera ya definidos a nivel de secuenciador (más información ver "¿Cómo editar las etapas y transiciones?" en la página 72).

Haga clic en  para añadir una sobrecarga. Para modificar los tiempos, sobrescriba el campo "**Valor**" de la etapa deseada.

Haga clic en 

para mostrar todos los valores de los timeout definidos en el secuenciador.

Haga clic en 

para mostrar todos los valores de tiempos de espera definidos en el secuenciador.

 Recuerde que los tiempos definidos en esta pestaña tienen prioridad sobre los definidos a nivel de secuenciador.

4.9.2.5.- **PESTAÑA INTERFACES**

Haga clic en  para asociar a este secuenciador un Interface.

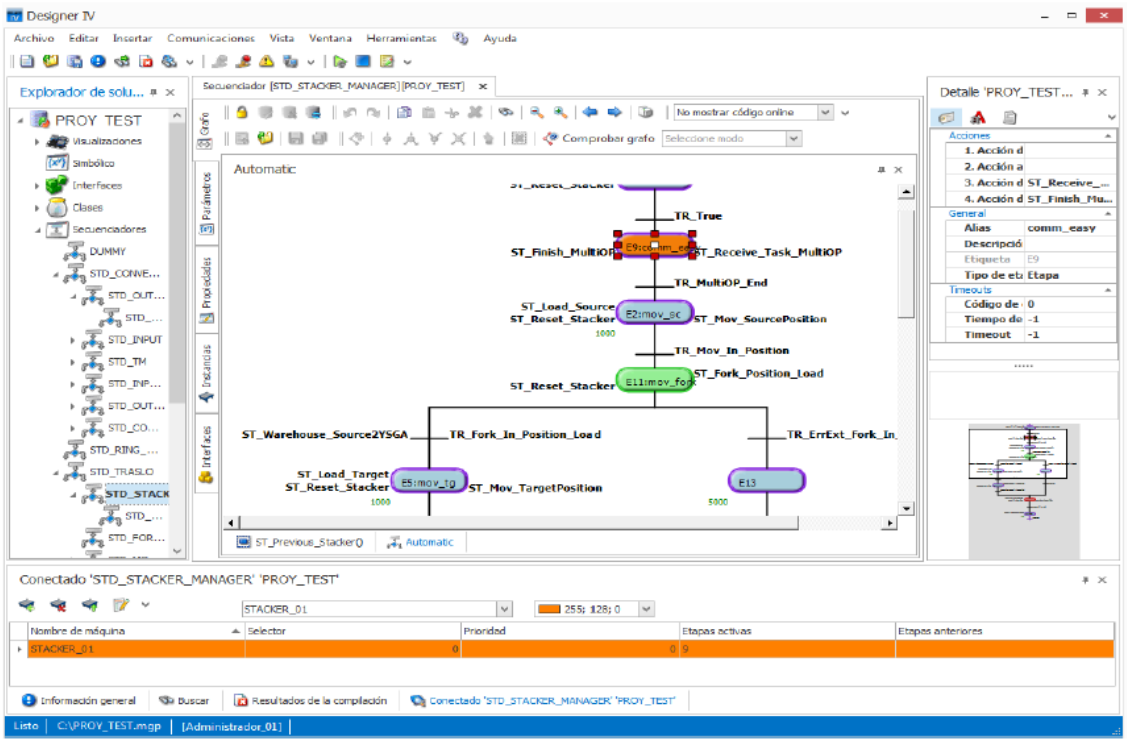
De esta forma, el secuenciador ha de implementar los métodos y/o propiedades definidos en dicho interface.

Haga clic en  para generar automáticamente en el secuenciador todos los métodos declarados en los interfaces asociados.

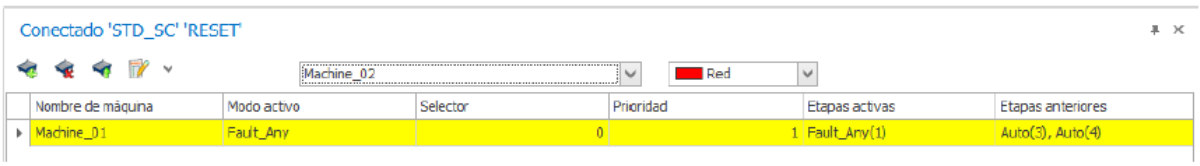
4.9.3.- MODO ONLINE DE LA PANTALLA "SECUENCIADORES"

Siga los pasos de "¿Cómo ejecutar el Modo Conectado?" en la página 180.

Tras ello, y con un modo de funcionamiento del secuenciador activo, la aplicación le muestra el panel inferior **Conectado '*Secuenciador*'.**



Tenemos distintas opciones accesibles desde la barra de herramientas de la pantalla o desde la barra de herramientas del panel **Conectado**.



4.9.3.1.- OPCIONES DEL PANEL CONECTADO

4.9.3.1.1.- AÑADIR UNA MÁQUINA

Dentro de este panel, mediante el desplegable, seleccione la máquina de este secuenciador que se desea visualizar. Puede asignarle un color específico para poder distinguirla de otras máquinas en el grafo.

Haga clic en  . Tras ello, la máquina aparece en la tabla inferior mostrando información sobre ella.

Haga clic en  para eliminar la máquina seleccionada de la tabla y dejar de mostrarse información sobre ella.

Haga clic en  para resetear la máquina seleccionada a cualquier etapa del grafo. Si no se indica la etapa se resetea directamente a la etapa inicial.

La información que se muestra en la tabla para cada máquina es la siguiente:

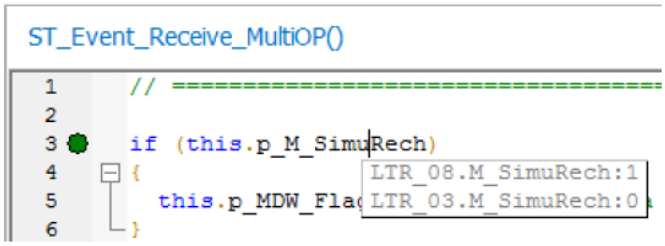
- **Modo activo**: Nombre del modo de funcionamiento activo.
- **Selector**: Si no se encuentra en un modo de prioridad, este número indica el selector (o modo de funcionamiento normal) que se está ejecutando.
- **Prioridad**: Prioridad actualmente activa. Si el valor es distinto de 0, significa que está en un modo de funcionamiento por interrupción.
- **Etapas activas**: Lista de etapas actualmente en ejecución. El número de etapa corresponde a la etapa dentro del modo de funcionamiento activo.
- **Etapas anteriores**: Etapas de retorno, es decir, aquellas a las que la máquina vuelve cuando salga del modo de prioridad.
  En el grafo aparecen marcadas con el borde coloreado para indicar que son el punto de retorno de la ANY.

4.9.3.2.- OTRAS OPCIONES

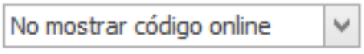
 Tras tener una o varias máquinas añadidas en el panel anterior, la aplicación nos ofrece otras funcionalidades.

4.9.3.2.1.- MOSTRAR EL VALOR DE VARIABLES ONLINE

Situando el puntero del ratón sobre una variable, la aplicación le muestra el valor online para aquellas máquinas añadidas en el panel **Conectado**.



4.9.3.2.2.- MOSTRAR EL CÓDIGO ONLINE

Para mostrar qué líneas de código se están ejecutando (marcadas con un punto verde) seleccione, en el desplegable 

de la barra de herramientas de la pantalla, la máquina de la cual quiere ver dichas líneas.

ST_Event_Receive_MultiOP()

```
1   // ============================================
2
3   if (this.p_M_SimuRech)
4   {                LTR_03.M_SimuRech:0
5     this.p_MDW_Flags = 2;      //Palet a Rechazos
6   }
7   else
8     this.p_MDW_Flags = 65536; //Palet OK
9
10
11  dword CurrentSlot = 1;
```

4.9.3.3.- **EDITOR DE TRACKING**

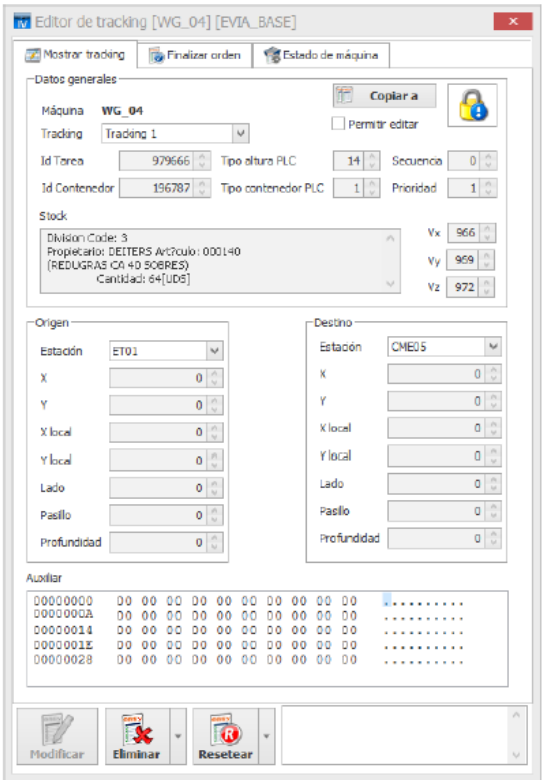
Para acceder al editor de tracking de una máquina en concreto siga los pasos descritos en "Añadir una máquina" en la página 86 en el grafo en modo online.

Tras ello, haga clic en  .

Otra opción es, desde la visualización, hacer doble clic sobre el objeto TlinkedElement (en el caso de que se active esta opción dentro del objeto).

El Editor de Tracking tiene varias pestañas. A continuación explicaremos los campos más relevantes de cada una de ellas.

4.9.3.3.1.- PESTAÑA MOSTRAR TRACKING



- **Datos generales**
  - *Máquina*: Nombre de la máquina de la cual se visualizan los datos.
  - *Secuencia*: Ante un modo de funcionamiento Master indica la posición de la orden dentro del conjunto de órdenes.
  - *Prioridad*: Relacionado con la secuencia. La prioridad tiene dos niveles. A igual prioridad se chequea el orden de la secuencia.
  - *Vx,Vy,Vz*: velocidades relativas (%) que el WMS envía a control para mover la máquina.

  El resto de datos que aparecen son relativos al contenedor que se encuentra en ese instante sobre la máquina.

  - *Tracking*: Número de tracking que se está visualizando. Puede tener un máximo de 10 trackings.
  - *Id Tarea, Id Contenedor, tipo contenedor PLC, tipo Altura PLC, Stock*: Datos relativos al contenedor.

- **Origen**
  Muestra todos los datos del punto de origen del contenedor.
  - *Estación*: Nombre de la estación de la que proviene el contenedor.
  - *X*: Posición del alveolo X en el que se encuentra el contenedor.
  - *Y*: Posición del alveolo Y en el que se encuentra el contenedor.
  - *X lógica*: Posición X dentro del alveolo en la que se encuentra el contenedor.
  - *Y lógica*: Posición Y dentro del alveolo en la que se encuentra el contenedor.
  - *Lado*: Lado del pasillo de la máquina en el que se encuentra el contenedor.
  - *Pasillo*: Pasillo de la máquina en el que se encuentra el contenedor.
  - *Profundidad*: Profundidad de la estantería en la que se encuentra el contenedor.

- **Destino**
  Muestra los datos del destino del contenedor. Los campos son los mismos que han sido descritos en el apartado anterior.

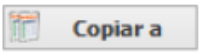
- **Datos auxiliares**
  Se muestran datos adicionales pertenecientes al contenedor.

Otras opciones:

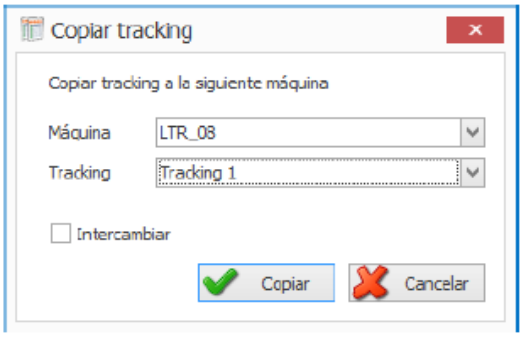
- **Permitir editar**: Check que permite editar los datos del contenedor que aparecen en el Editor de Tracking.

-  **Pestañas avanzadas**: acceso a opciones del Editor de Tracking que se encuentran ocultas y protegidas mediante clave.

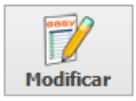
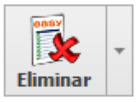
- **Copiar a...**

  Haga clic en 

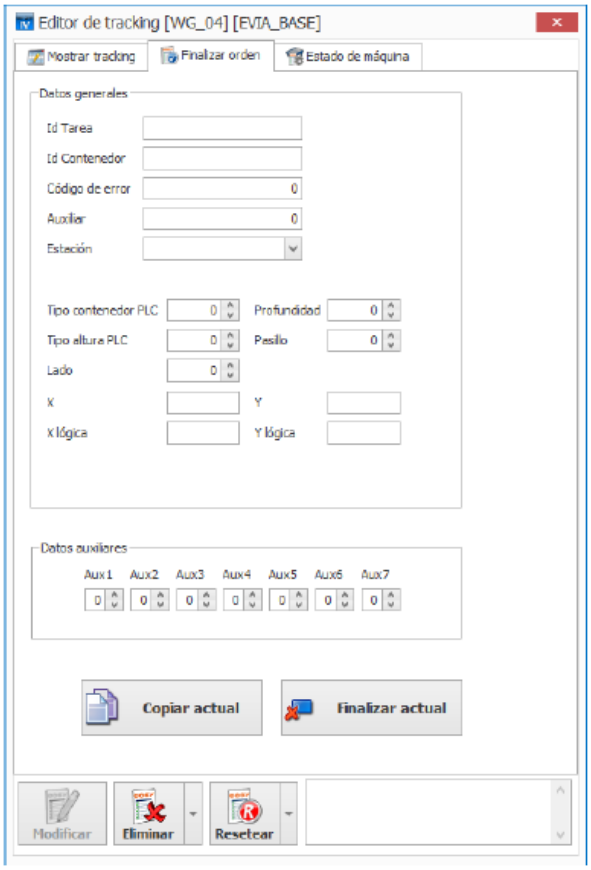
para copiar el tracking seleccionado a otra máquina / tracking. También es posible intercambiar trackings si se marca previamente el check **Intercambiar**.



- **Modificar, Eliminar y Resetear**
  Botones visibles desde todas las pestañas del Editor de Tracking.
  Permiten interactuar con el estado y los datos de la máquina que se muestra:

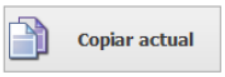
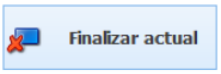
Botón	Descripción
	Permite modificar los datos del tracking que han sido editados. Se habilita cuando se ha activado el check *Permitir edición.*
	Permite borrar tanto el tracking actual que se está visualizando como todos los trackings de la máquina. En la parte derecha del botón aparece un desplegable que habilita ambas opciones.
	Permite resetear la máquina a todas aquellas etapas que hayan sido definidas con *Alias* dentro del grafo del secuenciador de la máquina. En la parte derecha del botón aparece un desplegable que muestra todas las etapas a las que es posible resetear la máquina.

4.9.3.3.2.- PESTAÑA FINALIZAR ÓRDENES



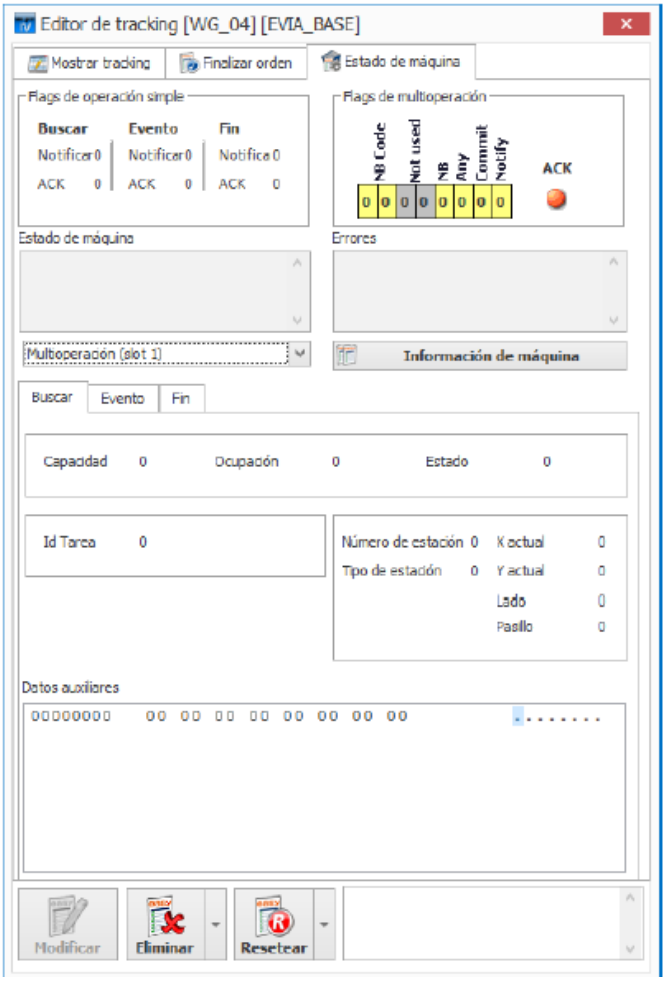
Para finalizar la orden actual, rellene todos los datos e indique el código de error si procede con el que se desea finalizar la orden.

La utilidad de esta pestaña es poder resolver situaciones anómalas de la instalación. En el caso de que se finalice la orden desde el Editor de Tracking, el programador de control ha de tener previsto que el estado de la instalación vuelva a ser correcto sin maniobras complicadas para el operario.

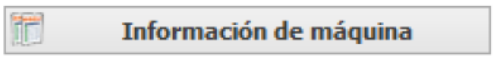
Botón	Descripción
	Copia los datos de tracking actuales a los campos de esta pestaña.
	Inicia el protocolo de fin de orden de los datos introducidos.

4.9.3.3.3.- PESTAÑA ESTADO MÁQUINA

Permite conocer el estado de los flags internos de funcionamiento de una máquina. Muestra los datos referentes al estado de las comunicaciones entre el Sistema de Control y el Sistema de Gestión de Almacén.



- **Datos de la máquina**
  Muestra el estado actual de la máquina en cuanto a comunicaciones con el Sistema de Gestión (finalizando orden, orden finalizada, etc.).
  En el campo **Errores** se muestran errores generados por el Sistema de Gestión en el caso de que se hayan producido en la última comunicación.

  Haga clic en 

  para ver datos de la máquina: Nombre, Número de máquina, Equipamiento asociado y offset de averías.

- **Zonas de comunicaciones**

Consta de una zona de flags y otra con varias pestañas según la selección realizada en el desplegable 

:

- **Operaciones simples**
  - **Búsqueda**
    Contiene datos útiles si la máquina puede recibir órdenes.
    Muestra el estado del **Flag Notify Petición de Orden** (controlado por el Sistema de Control) y el **Flag Ack Orden Recibida** (controlado por el EasyWMS®).
    Si selecciona en el desplegable esta opción, además también se muestran todos los datos con los que el programa de control está realizando la petición de la orden al Sistema de Gestión.

  - **Evento**
    Contiene datos útiles cuando la máquina puede realizar eventos.
    Muestra el estado del **Flag Notify Evento** (controlado por el Sistema de Control) y el **Flag Ack Evento Procesado** (controlado por el EasyWMS®).
    Si selecciona en el desplegable esta opción, además también muestra todos los datos con los que el programa de control está realizando el evento al Sistema de Gestión.

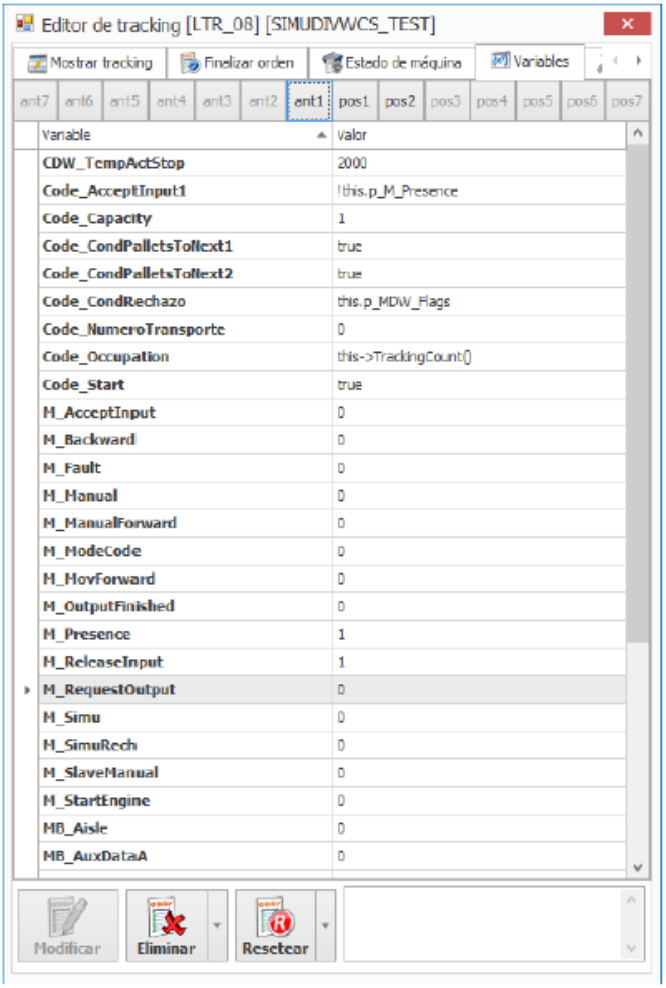
- **Fin de Orden**
  Contiene datos útiles cuando la máquina puede finalizar órdenes.
  Muestra el estado del **Flag Notify Fin de Orden** (controlado por el Sistema de Control) y el **Flag Ack Fin de Orden Procesado** (controlado por el EasyWMS®).
  Si selecciona en el desplegable esta opción, además también muestra todos los datos con los que el programa de control está realizando la finalización de la orden al Sistema de Gestión.

- **Operaciones con Multioperación**
  Muestra el estado del **Flag Notificación** (controlado por el Sistema de Control), del **Flag Multioperación procesada** (controlado por el EasyWMS®), del **Flag Any** (controlado por el Agente de Transportes), y el **Flag NB** que indica que la operación es no bloqueante (controlado por el Sistema de Control).
  Destacamos que solo existe una función para lanzar la multioperación por lo que los flags son únicos para la multioperación, independientemente del número de fines/eventos/búsquedas que se realicen.
  Si selecciona en el desplegable esta opción, ha de indicar el **Slot** (zona de memoria) para mostrar los datos con los que el programa de control está realizando el fin/evento/búsqueda al Sistema de Gestión en esa zona de memoria.

Puede encontrar más información detallada sobre este tipo de comunicación entre el sistema de control y EasyWMS® en el documento **Interface de Comunicaciones de Control.**

A partir de este momento se describen el resto de pestañas pertenecientes al Editor de Tracking pero que permanecen ocultas. Si están ocultas, haga doble clic  en para acceder a ellas e indique el usuario y contraseña con los privilegios adecuados para continuar.

4.9.3.3.4.- PESTAÑA VARIABLES



La pestaña muestra un listado de todas las variables de la máquina y sus valores.

Los valores que se muestran en esta tabla no son modificables y se refrescan periódicamente. Sin embargo, como un caso excepcional, puede realizar un Toggle (alternar valor) del valor de las variables booleanas. Para ello seleccione en el menú

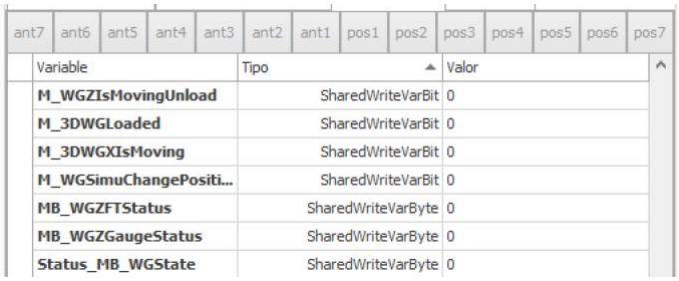
contextual de la variable la opción 

. Al usar esta opción, se realizará un forzado de la variable booleana seleccionada al valor opuesto al que tiene actualmente. También es posible, haciendo doble clic sobre cualquier variable, abrir un panel para visualizarlas de forma más cómoda (variables de tipo String por ejemplo).

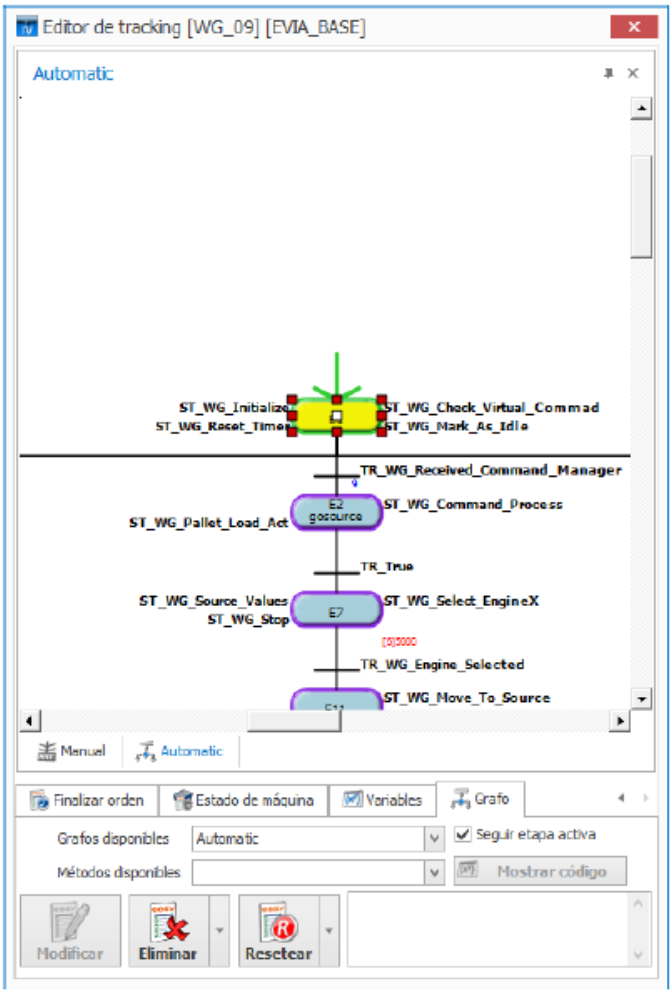
En la parte superior de la pestaña aparecen los siguientes botones:

Los anteriores o posteriores configurados para la máquina del Editor de Tracking se muestran en color negro. Puede hacer clic en cualquiera de ellos para abrir el Editor de Tracking del anterior/posterior correspondiente.

También puede ordenar las variables por **Tipo**. Para ello haga clic derecho sobre la cabecera de la tabla (por ejemplo encima de la celda Variable), seleccione la opción Selector de Columnas, y en la ventana mostrada arrastre la celda Tipo a la cabecera anterior. En este instante se muestra una columna más en el Editor de Tracking con el tipo de las variables.



4.9.3.3.5.- PESTAÑA GRAFO



A modo de lo explicado en "Modo Online de la pantalla Secuenciadores" en la página 85, la pestaña muestra el grafo online seleccionado en el desplegable Grafos disponibles Auto.

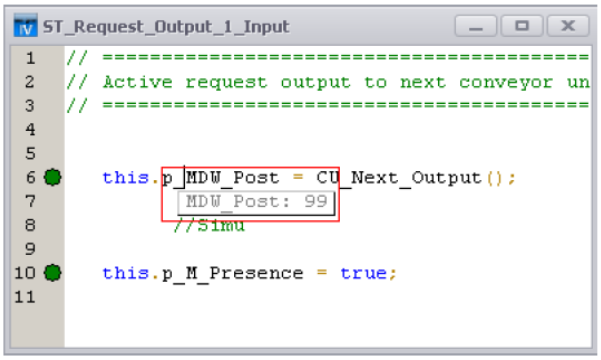
La opción Seguir etapa activa selecciona de forma automática el grafo que contiene la etapa activa, mostrando y centrando esta en el Editor de tracking.

Cuando **existe más de una etapa activa en el grafo** (ramas AND) no se aplica zoom sobre el mismo, **se ajusta todo el grafo al ancho del Editor** de Tracking. En estos casos puede aplicar zoom (no seleccionando la opción anterior) o deslocalizar el grafo, sacándolo fuera del Editor para poder aumentar su tamaño y ver de forma más clara qué etapas activas tenemos.

Haga clic en 

para ver el código online del método seleccionado en el desplegable 

.

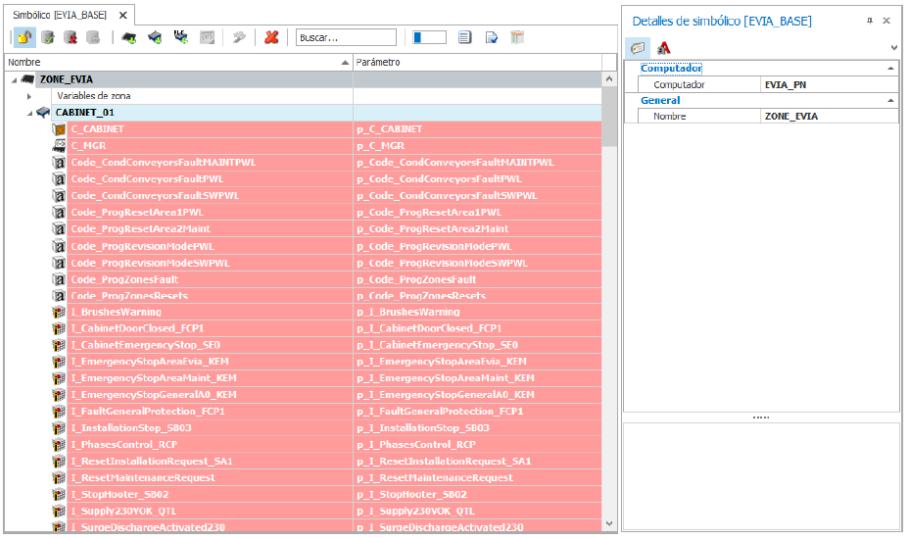


En la pantalla mostrada, puede observar marcadas con un punto verde las líneas de código que se están ejecutando. También, situando el puntero del ratón encima de una variable, la aplicación le muestra su valor online.

4.10.- **PANTALLA "SIMBÓLICO"**

En esta pantalla se configuran, tras definir los comportamientos del sistema mediante los secuenciadores, las zonas y máquinas de la instalación así como las variables de cada tipo a manejar para cada una de ellas.

En el árbol de la izquierda se muestran las zonas, máquinas y variables definidas. Haciendo clic en cualquiera de ellas, los valores que las definen aparecen en el formulario de detalle de la derecha.



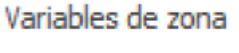
Haga clic en  para habilitar todas las funcionalidades y bloquear la pantalla en el proyecto para el resto de usuarios.

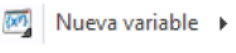
Recuerde que en Designer IV no es posible trabajar con variables Globales.

4.10.1.- ¿CÓMO CREAR UNA ZONA Y VARIABLES DE ZONA?

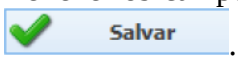
Haga clic en  o bien desde el menú contextual seleccione la opción **Nueva Zona**.

Una vez creada la zona automáticamente se asigna al Computador por defecto. Si lo necesita, modifique el computador en el formulario de detalle.

Para crear variables de zona seleccione 

y en su menú contextual haga clic en 

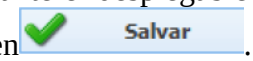
.

Rellene los campos obligatorios para cada tipo de variable y finalmente, haga clic en 

.

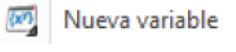
4.10.2.- ¿CÓMO CREAR UNA MÁQUINA Y VARIABLES DE MÁQUINAS?

Seleccione la **zona** de la que quiere que dependa la máquina. Haga clic  o bien desde el menú contextual seleccione la opción **Nueva Máquina** o **Nueva Máquina PLC**.

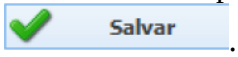
Rellene el campo **"Nombre"** y seleccione mediante el desplegable el **"Secuenciador"** que instancia la máquina. Finalmente, haga clic en 

.

4.10.3.- ¿CÓMO CREAR VARIABLES DE MÁQUINAS?

Seleccione la **máquina** y haga clic en 

.

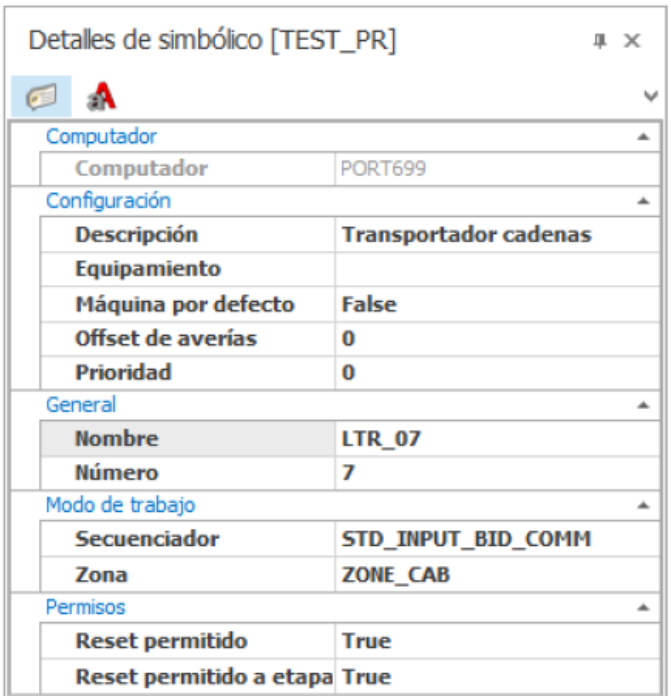
Rellene los campos obligatorios para cada tipo de variable y finalmente, haga clic en 

.

4.10.4.- ¿CÓMO EDITAR UNA MÁQUINA?

Haga clic sobre la máquina a editar 

. La aplicación le muestra un formulario de detalle a la derecha de la pantalla con los siguientes campos:



- **Computador**: nombre del computador, configurado en la Zona, al que pertenece la máquina.
- **Descripción** informativa de la máquina.
- **Equipamiento**: nombre del "Equipamiento" para las máquinas de tipo PIE, transelevador o lanzadera el cual es facilitado por el implantador de EasyWMS®.
- **Máquina por defecto**: indica si es la máquina por defecto del proyecto, es decir, es la máquina utilizada como anterior o posterior de aquellas que no las tengan definidas.
- **Offset de averías:** define un número de avería base que se utiliza en esta máquina para indexar las averías.
  De esta manera el número máximo de averías por máquina no varía (512) pero el número total de averías disponible aumenta considerablemente.
  El siguiente ejemplo ilustra el funcionamiento de este campo:
  - Se define una maquina con un offset de averías 0 (por ejemplo, un transelevador). Se define otra máquina con un offset de averías 512. (por ejemplo un transportador).
  - En la configuración del Agente de Transportes se definen las averías para los transelevadores en el intervalo 0-511, y para los transportadores en el intervalo 512-1023.
  - Cuando por código se pide evaluar la avería 4 en la máquina del transelevador, Galileo IV añade a este número el offset de la máquina (en

este caso 0) para buscar la avería correspondiente en la lista configurada en el TA. Si por el contrario se pide evaluar la avería 4 en la máquina del transportador, la avería buscada será la 516 (512 + 4).

- **Prioridad**: modifica el orden de ejecución de las máquinas de un mismo computador. Por defecto, el orden de ejecución es el orden de creación.
- **Nombre**: nombre de la máquina. Es único por proyecto.
- **Número**: número de máquina. No tiene que ver con el orden en que son manejadas las máquinas por Galileo IV, sino que es un identificador interno usado por el mismo.


El número de máquina no puede repetirse en ninguna otra máquina del proyecto ni puede tener valor 0.

- **Secuenciador**: desplegable con los secuenciadores definidos en el proyecto.


Un cambio en el secuenciador asociado a la máquina implica que los parámetros se pueden desligar de las variables de la máquina.

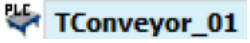
- **Zona:** desplegable con las zonas definidas en el proyecto. Indica a qué zona pertenece la máquina.


Un cambio en la zona asociada a la máquina implica que, si estuvo previamente ligada a otra zona que está asociada a otra computadora, esta máquina se "mueve" de la otra zona a la actual, **pero se ha de recordar que las tarjetas de control no lo hacen**. Al compilar el proyecto se indicará un error por existir módulos de bus que pertenecen a la otra computadora.

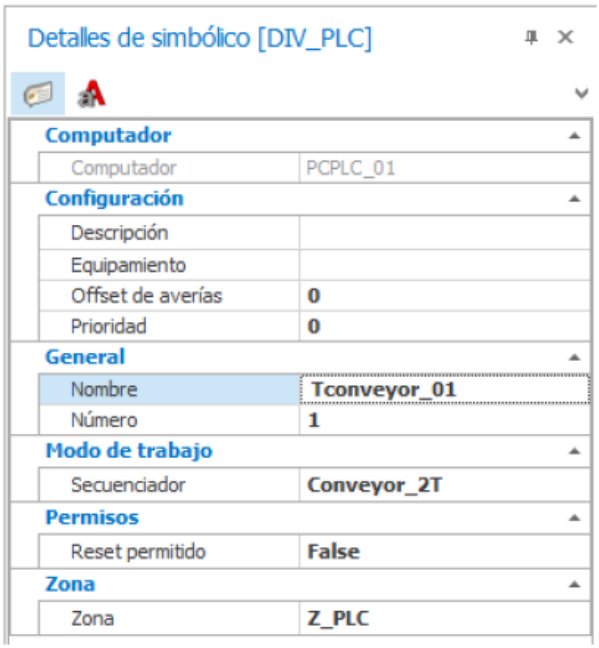
- **Reset permitido**: permite el reset desde el Editor de Tracking a cualquier etapa definida con un alias.
- **Reset permitido a etapa inicial**: permite el reset desde el Editor de Tracking a la etapa inicial (no es necesario que tenga ningún alias definido en el secuenciador).


Para cada máquina puede definir tiempos de sobrecarga de timeout y espera. Para más información ver "Pestaña Instancias" en la página 84.

4.10.5.- ¿CÓMO EDITAR UNA MÁQUINA PLC?

Haga clic sobre la máquina PLC a editar 

. La aplicación le muestra un formulario de detalle a la derecha de la pantalla con los siguientes campos:



- **Computador**: nombre del computador, configurado en la Zona, al que pertenece la máquina PLC.
- **Descripción** informativa de la máquina.
- **Equipamiento**: nombre del "Equipamiento" que de ser necesario es facilitado por el implantador de EasyWMS®.
- **Offset de averías:** define un número de avería base que se utiliza en esta máquina para indexar las averías.
- De esta manera el número máximo de averías por máquina no varía (512) pero el número total de averías disponible aumenta considerablemente.
- **Prioridad**: modifica el orden de ejecución de las máquinas de un mismo computador. Por defecto, el orden de ejecución es el orden de creación.
- **Nombre**: nombre de la máquina PLC. Es único por proyecto.
- **Número**: número de máquina. No tiene que ver con el orden en que son manejadas las máquinas por Galileo IV, sino que es un identificador interno usado por el mismo.

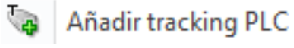
El número de máquina no puede repetirse en ninguna otra máquina del proyecto ni puede tener valor 0.

- **Secuenciador**: desplegable con los secuenciadores PLC definidos en el proyecto.
- **Reset permitido**: permite el reset desde el Editor de Tracking a cualquier etapa definida con un alias.
- **Zona:** desplegable con las zonas definidas en el proyecto. Indica a qué zona pertenece la máquina.

Para que la comunicación entre Galileo y el PLC sea correcta, es necesario definir una serie de estructuras (zonas mapeadas en el PLC) que contienen datos que se intercambian entre el PLC y Galileo.

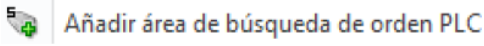
Las estructuras estándar definidas son:

- **Trackings**: Son utilizados para que Galileo pueda entender en todo momento los datos de los transportes que está manejado una máquina PLC. No es obligatorio que una máquina PLC defina estás estructuras, aunque si es necesario si pretende solicitar información a Galileo en forma de órdenes de transporte. El número máximo de trackings que Galileo admite son 10 por máquina de forma simultánea.

  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción 

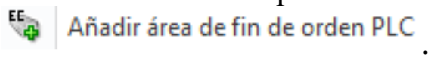
.

- **Área de Búsqueda de orden**: La máquina ha de definir en el PLC una única estructura de este tipo si requiere recibir órdenes.

  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción 

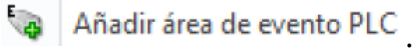
.

- **Área de Fin de orden**: La máquina ha de definir en el PLC una única estructura de este tipo si requiere finalizar órdenes.

  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción 

.

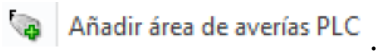
- **Área de Evento**: La máquina ha de definir en el PLC una única estructura de este tipo si requiere realizar un evento.

  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción 

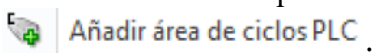
.

- **Área de Averías**: Un campo de 16 bytes es requerido para ubicar las averías que la máquina va notificando. Los números de avería han de estar comprendidos entre 0 y 127, y se marcan activando el bit con ese número dentro del área de memoria. El **Offset** se indica en las propiedades de la máquina PLC.

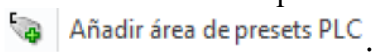
  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción



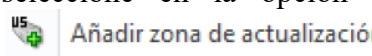
- **Área de Ciclos**: Permiten a la máquina PLC indicar al usuario un punto concreto de la ejecución. Se trata de un valor numérico que es modificado por el PLC cuando alcance el punto de la ejecución que considere oportuno, así el usuario puede saber en qué estado se encuentra la ejecución del secuenciador. Si no están definidos dentro del programa del PLC no hay que añadir este tipo de estructura.

  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción 

- **Área de Presets**: Análogos a los ciclos, pero vistos desde el lado de Galileo. Permiten indicarle al PLC el punto de la ejecución a la que se quiere ir.

  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción 

- **Zona de actualización de Estaciones**: Se define la zona que utiliza la máquina para realizar la actualización de estaciones.

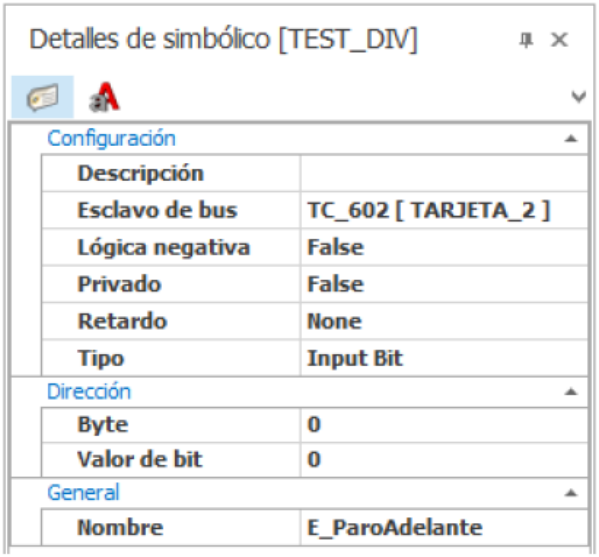
  Para añadir una estructura de este tipo haga clic derecho sobre la máquina PLC y seleccione en la opción "**Añadir zona mapeada en PLC**" la opción 

Las estructuras tienen las mismas propiedades en el formulario de detalle:

- **PLC** → Desplegable con los PLC definidos en la pantalla de Computadores
- **Byte** → Byte donde se inicia la estructura dentro del DB indicado.
- **DB** → DB en el que está mapeada la estructura en el PLC.
- **DW** → DW en el que está mapeada la estructura en el PLC.
- **Nombre**
- **Índice del tracking** → Solo para el área de tipo Tracking indica el índice basado en 0 del tracking (máximo 10).

4.10.6.- ¿CÓMO EDITAR VARIABLES DE MÁQUINA?

Seleccione la variable a editar. La aplicación le muestra un formulario de detalle a la derecha de la pantalla con los distintos campos en función del tipo de variable seleccionada. Por ejemplo:



A continuación vamos a describir, dejando el campo Nombre y Tipo aparte, los distintos campos que nos podemos encontrar a la hora de crear una variable o en el formulario de detalle de la misma:

- **Variables Tipo Shared**:
  - **Privada:** permite especificar la visibilidad de la variable. *True* indica que la variable puede ser accedida desde máquinas del mismo computador. *False* indica que la variable es pública, pueden ser accedidas desde cualquier computador.

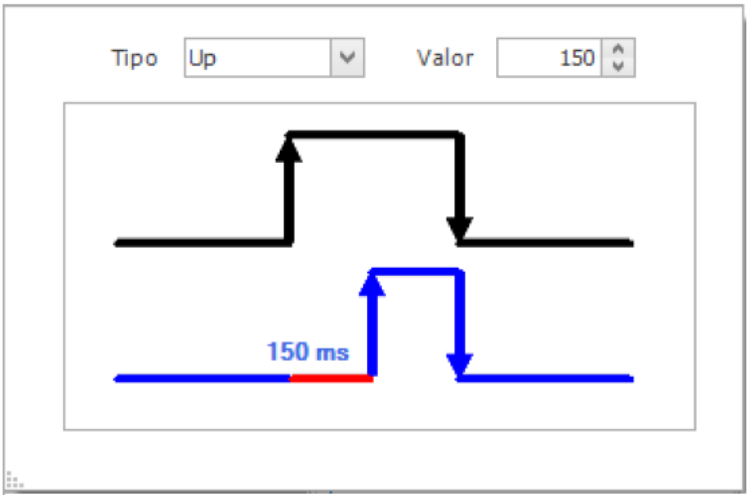
    Es conveniente mantener un número bajo de variables públicas por los siguientes motivos**:**

    - Los secuenciadores con pocas variables públicas son más robustos, al no depender de otros secuenciadores que puedan acceder a sus variables.
    - Se reducen las comunicaciones. Hasta la versión 3.1.3.40 de Galileo, cada computador exportaba su imagen de proceso (de hasta 64 Kb) al resto de computadores del proyecto. Esto producía una sobrecarga de comunicaciones que en algunos casos no era aceptable. Con el uso de variables públicas y privadas Galileo IV conoce qué variables debe enviar para minimizar el tamaño de los mensajes. A modo de ejemplo, para igualar el tamaño de un mensaje de imagen de proceso, un computador tendría que tener 4092 variables públicas.

- **Variables Tipo E/S**: variables asociadas a la periferia física controlada por Galileo. Existen 5 tamaños distintos: Bit, Byte, Word (2 bytes), Dword (4 bytes) o Qword (8 bytes).
  - **Dirección**: dirección donde comienza la estructura de la variable (indexado a través del módulo de bus asociado). Sólo en el caso de que se seleccione el tamaño bit, es necesario especificar qué bit dentro del byte se va a usar.
  - **Esclavo Bus**: desplegable con los esclavos dados de alta en la pantalla de Configuración de Bus.

La aplicación comprueba que el espacio disponible desde el direccionamiento indicado al final del esclavo es suficiente para esa variable. Si no es suficiente se muestra un warning, pero se permite la creación de la variable. En la lista de variables, aquellas que tengan un tamaño superior al del esclavo, se muestran en amarillo para que sea fácil identificarlas.

- **Retardo**:



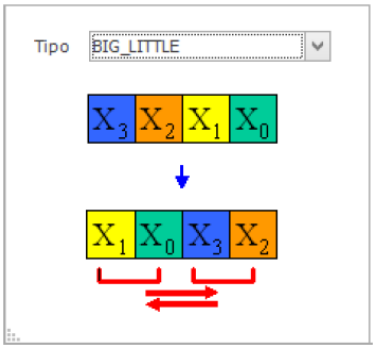
  - **Tipo**: tipo de cambio de señal, flanco positivo (Up) o negativo (Down).
  - **Valor**: periodo de tiempo, medido en milisegundos, que se desea retrasar el cambio de la señal (en subida o en bajada), desde que esta se produce en el bus hasta que el programa de control detecte ese cambio.

Una vez seleccionado el tipo y valor, para aceptarlo haga clic fuera del panel. Para cancelar la operación pulse la tecla *Esc*.

- **Descripción**
- **Lógica Negada**: posibilidad de invertir la lógica de las señales de tipo bit. De esta manera, la señal se procesa en Galileo IV negada con respecto a la señal real. Existen ciertas limitaciones a la hora de trabajar con señales de lógica negativa:

- Solo se puede aplicar a variables input de tipo bit.
- Si la variable se mapea sobre una tarjeta con el driver virtual (*hardware virtual*) o se tiene habilitada la simulación, la simulación de las señales se puede realizar con la lógica negada o bien, a partir de la versión **4.0.1153.9057** de Galileo, activar el check de la consola "Configuración > Simulación > **Permitir lógica negada en la simulación"** y trabajar como si existiese una tarjeta real con intercambio de E/S.

- **Alineación**



Se indica el formato en el que se almacenan los datos de más de un byte.

- Se puede aplicar a variables input de tipo word, dword y qword.

Una vez seleccionado el tipo, para aceptarlo haga clic fuera del panel. Para cancelar la operación pulse la tecla *Esc*.

- **Variables Tipo PLC**: variables asociadas a la periferia física controlada por uno de los PLCs definidos con quien Galileo actúa como interfaz para asignarle órdenes a sus máquinas.
  - **Dirección**: DB donde la variable está almacenada y DW donde comienza la estructura. Sólo en el caso de que se seleccione el tamaño bit, es necesario especificar qué bit dentro del byte se va a usar.
  - **Tipo de almacenamiento**: tipo de memoria que queremos mapear (DATABLOCK, MEMORY, INPUT OUTPUT). Si la memoria es de tipo DB (DataBlock), se ha de indicar también el número de DB al que queremos acceder (máximo 255). En otro caso, se usa el 0 como DB.

A diferencia de las variables E/S, las variables PLC no se definen a partir de un direccionamiento indexado (a través de los módulos de bus), sino que han de ser direccionadas de forma absoluta. Además, van

asociadas de forma implícita al PLC que controla la máquina que las contiene.

- **Variables Tipo Componente**:

  - **Interface**: desplegable con los componentes accesibles en Galileo IV. Un componente es un objeto nativo en lenguaje XanaO2, capaz de realizar unas determinadas tareas de forma automática.
    Típicamente hay componentes que permiten controlar hardware específico o realizar operaciones sólo al alcance del Sistema Operativo donde se ejecuta Galileo IV.
  - **Esclavo Bus**: desplegable (visible según Interface seleccionado) con los esclavos dados de alta.

- **Variables Tipo String**:

  - **Valor**: puede contener fragmentos de código que se asocian con una variable *inline*. Las variables *inline* se comportan como una especie de macro, expandiéndose en el código con los contenidos de la cadena que se asigne a la variable String.

     No está permitido el uso de comentarios de línea "//" en este tipo de variables. El uso de estos puede traer comportamientos indeseados en la aplicación.

- **Variables Tipo Constante**:
  - **Valor**: Debe de tener asignado un valor numérico dentro de los rangos del tamaño seleccionado.

- **Variables Tipo Clase**:
  - **Clase**: desplegable con las definidas en la pantalla **Clases.**

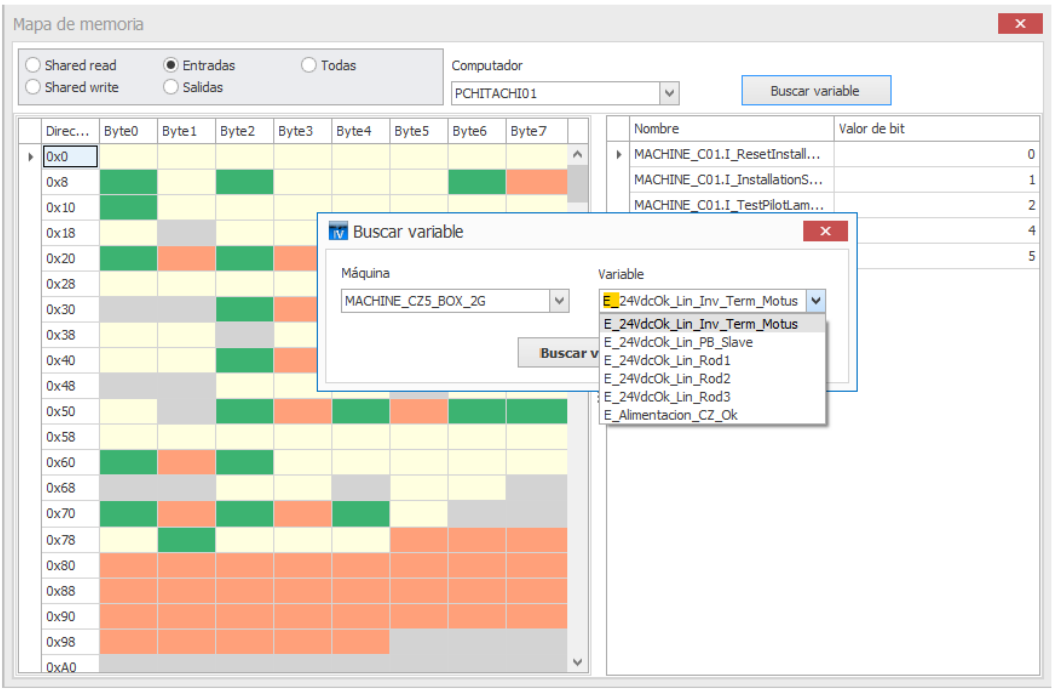
- **Variables Tipo Colección (Vector o Lista)**:

  - **Plantilla**: tipo definido para el Vector o Lista creado**.**

4.10.7.- OPCIONES DE LA PANTALLA

4.10.7.1.- MAPA DE MEMORIA

Desde esta pantalla podemos acceder al **Mapa de Memoria**, para ello haga clic en  en la barra de herramientas.

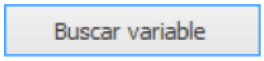


En la ventana mostrada se detalla dónde se sitúan las variables definidas en la memoria de periferia (para las variables I/O) y de intercambio (para variables SHARED).

La leyenda del mapa de memoria es la siguiente (cada celda de memoria corresponde a un byte):

- Celda de memoria ocupada totalmente.
- Celda de memoria ocupada parcialmente.
- Celda de memoria libre.
- Celda de memoria con solapamiento de variables definidas.

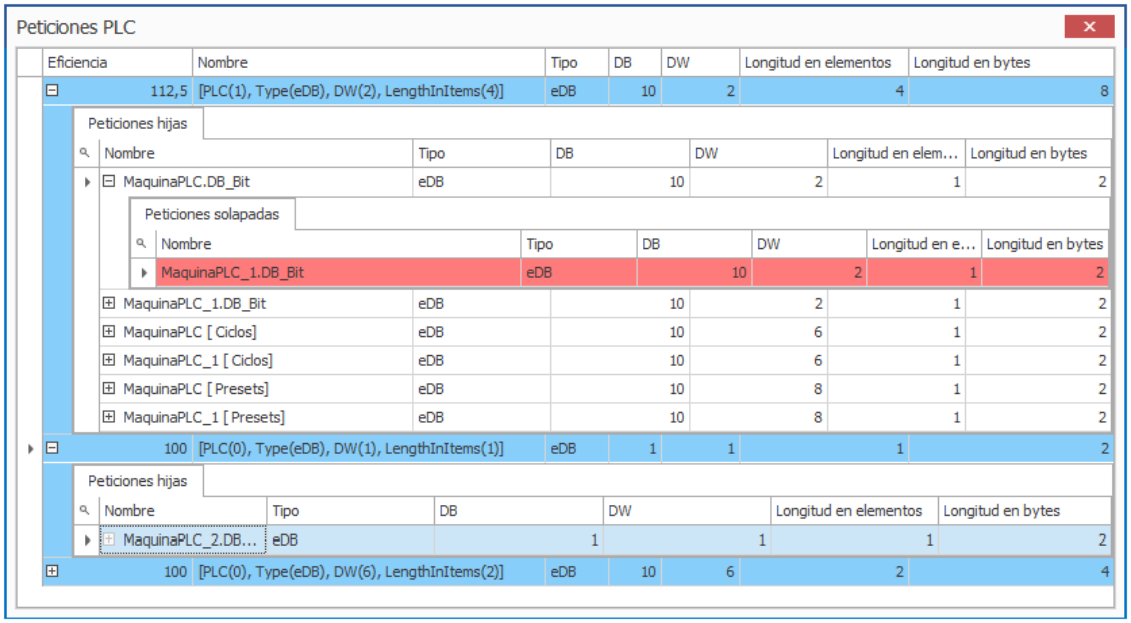
Además de detectar solapamientos, el mapa de variables da una idea bastante aproximada de cómo está de ocupada la memoria de Galileo IV.

Haga clic en 

para localizar en el Mapa de Memoria una variable en concreto. Seleccione la máquina y la variable y la aplicación le indicará la línea y byte dónde está mapeada la variable.

4.10.7.2.- **MOSTRAR PETICIONES PLC**

Desde esta pantalla podemos acceder a la ventana de Peticiones de PLC, para ello haga clic en  en la barra de herramientas.

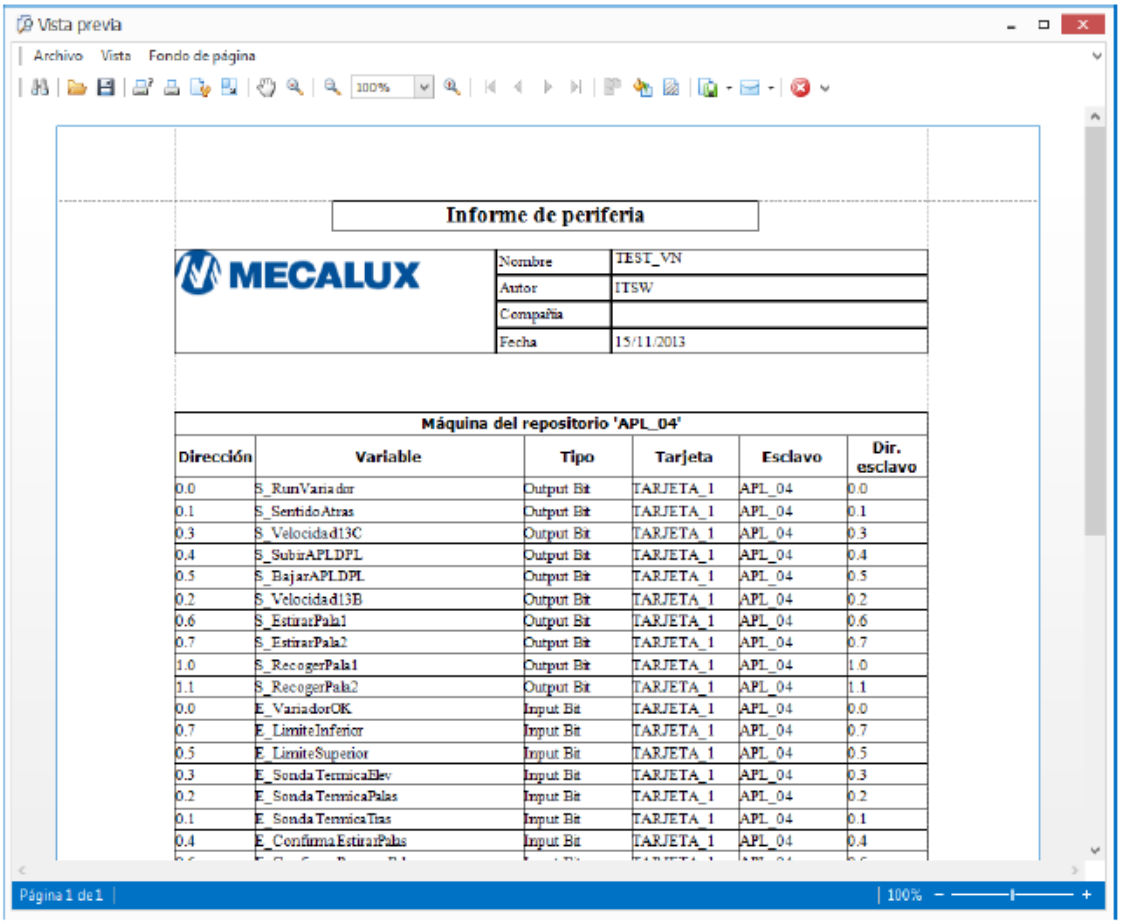


La pantalla ofrece información acerca de la eficiencia y otros datos de las peticiones realizadas al PLC, indicando las zonas de mapeado, variables, si existen peticiones solapadas, etc.

4.10.7.3.- **INFORME DE PERIFERIA**

Desde esta pantalla podemos generar un **Informe de Periferia**, es decir, un documento con todas las variables de periferia definidas en las máquinas (o máquina en concreto). Para ello haga clic en  en la barra de herramientas.

La pantalla ofrece varias opciones como buscar, guardar, imprimir, enviar por correo electrónico, exportación en múltiples formatos, etc...



Dirección	Variable	Tipo	Tarjeta	Esclavo	Dir. esclavo
0.0	S_RunVariador	Output Bit	TARJETA_1	APL_04	0.0
0.1	S_SentidoAtras	Output Bit	TARJETA_1	APL_04	0.1
0.3	S_Velocidad13C	Output Bit	TARJETA_1	APL_04	0.3
0.4	S_SubirAPLDPL	Output Bit	TARJETA_1	APL_04	0.4
0.5	S_BajarAPLDPL	Output Bit	TARJETA_1	APL_04	0.5
0.2	S_Velocidad13B	Output Bit	TARJETA_1	APL_04	0.2
0.6	S_EstirarPala1	Output Bit	TARJETA_1	APL_04	0.6
0.7	S_EstirarPala2	Output Bit	TARJETA_1	APL_04	0.7
1.0	S_RecogerPala1	Output Bit	TARJETA_1	APL_04	1.0
1.1	S_RecogerPala2	Output Bit	TARJETA_1	APL_04	1.1
0.0	E_VariadorOK	Input Bit	TARJETA_1	APL_04	0.0
0.7	E_LimiteInferior	Input Bit	TARJETA_1	APL_04	0.7
0.5	E_LimiteSuperior	Input Bit	TARJETA_1	APL_04	0.5
0.3	E_SondaTermicaElev	Input Bit	TARJETA_1	APL_04	0.3
0.2	E_SondaTermicaPalas	Input Bit	TARJETA_1	APL_04	0.2
0.1	E_SondaTermicaTras	Input Bit	TARJETA_1	APL_04	0.1
0.4	E_ConfirmaEstirarPalas	Input Bit	TARJETA_1	APL_04	0.4

4.10.7.4.- **MENÚ CONTEXTUAL**

Las opciones más destacables de los menús contextuales, accesibles haciendo clic derecho en el árbol sobre una zona, máquina o variable son:

- **Máquina → Crear y enlazar todo**: crea todas las variables, con el mismo nombre que el parámetro eliminando el **p_** y las asocia a su parámetro.
- **Máquina → Cambiar todas a públicas**: convierte todas las variables de la máquina a públicas.
- **Máquina → Cambiar todas a privadas**: convierte todas las variables de la máquina a privadas.
- **Máquina → Ir a secuenciador**: abre la pantalla del secuenciador asociado a la máquina.
- **Variable → Enlazar con**: asocia la variable al parámetro, muestra aquellos parámetros de igual tipo que la variable.
- **Variable → Enlazar y propagar**: asocia la variable al parámetro y la propaga en aquellas máquinas asociadas al mismo secuenciador que la que la contiene.
- **Variable → Eliminar:** elimina la variable.

- **Variable → Propagar:** propaga la variable en aquellas máquinas asociadas al mismo secuenciador que la que la contiene.
- **Variable → Propagar ámbito:** propaga el alcance (pública/privada) de la variable a todas las del mismo nombre en máquinas del mismo secuenciador que la que la contiene.
- **Variable → Renombrar y propagar nombre:** modifica el nombre de la variable y lo propaga en máquinas del mismo secuenciador (o heredado) que la contienen.
- **Variable → Desenlazar parámetro**: desasocia la variable al parámetro.
- **Variable → Valor de la cadena**: para variables de tipo *string*. Visualiza el valor configurado en el formulario de detalle. Puede ser útil cuando se encuentra la pantalla bloqueada. Permite copiar el valor mostrado.
- **Parámetro no asociado a variable → Crear y enlazar variable**: crea una variable, con el mismo nombre que el parámetro eliminando el **p_** y se la asocia.

Otra opción disponible fuera del menú contextual es:

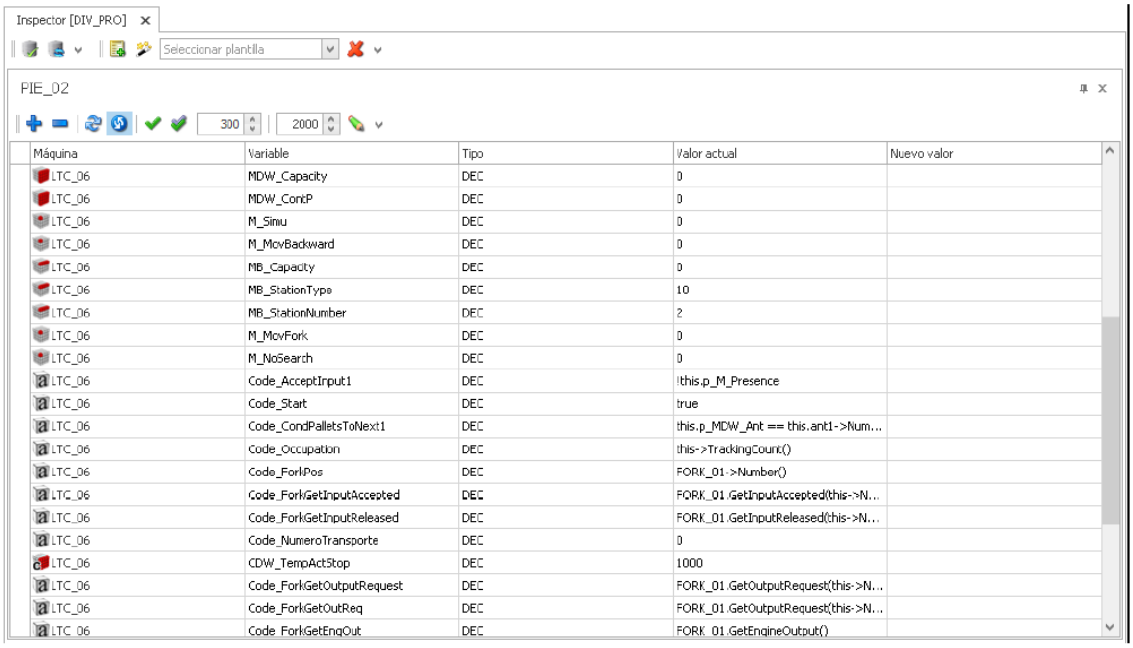
- **Clonar máquinas** → Ctrl+clic ratón y desplazarla sobre otra máquina (o sí misma) de la misma zona u otra. Permite crear máquinas basadas en otras, dando la posibilidad de cambiar los módulos de bus y, en el caso de tipo PLC, permite establecer un offset en DB y DW para que la máquina clonada añada esos offset a la original.

 Recordemos que si se asocia una máquina previamente ligada a otra zona de diferente computadora, esta se "mueve" de la otra computadora a la actual, pero se ha de recordar que las tarjetas de control no lo hacen, de forma que si la máquina requiere de módulos de bus que pertenecen a la otra computadora existen problemas con la configuración del bus. Esto es indicado al compilar el proyecto.

 Recuerde, haga clic en  para guardar en el proyecto todos los cambios realizados en la pantalla.

4.11.- PANTALLA "INSPECTOR"

Esta pantalla permite visualizar los valores de las variables definidas en el proyecto y forzar sus valores.



Máquina	Variable	Tipo	Valor actual	Nuevo valor
LTC_06	MDW_Capacity	DEC	0	
LTC_06	MDW_ContP	DEC	0	
LTC_06	M_Simu	DEC	0	
LTC_06	M_MovBackward	DEC	0	
LTC_06	MB_Capacity	DEC	0	
LTC_06	MB_StationType	DEC	10	
LTC_06	MB_StationNumber	DEC	2	
LTC_06	M_MovFork	DEC	0	
LTC_06	M_NoSearch	DEC	0	
LTC_06	Code_AcceptInput1	DEC	!this.p_M_Presence	
LTC_06	Code_Start	DEC	true	
LTC_06	Code_CondPalletsToNext1	DEC	this.p_MDW_Ant == this.ant1->Num...	
LTC_06	Code_Occupation	DEC	this->TrackingCount()	
LTC_06	Code_ForkPos	DEC	FORK_01->Number()	
LTC_06	Code_ForkGetInputAccepted	DEC	FORK_01.GetInputAccepted(this->N...	
LTC_06	Code_ForkGetInputReleased	DEC	FORK_01.GetInputReleased(this->N...	
LTC_06	Code_NumeroTransporte	DEC	0	
LTC_06	CDW_TempActStop	DEC	1000	
LTC_06	Code_ForkGetOutputRequest	DEC	FORK_01.GetOutputRequest(this->N...	
LTC_06	Code_ForkGetOutReq	DEC	FORK_01.GetOutputRequest(this->N...	
LTC_06	Code_ForkGetEngOut	DEC	FORK_01.GetEngineOutput()	

 Aunque es posible acceder a esta pantalla sin estar en modo Conectado, no es funcional si no se está en ese modo.

4.11.1.- ¿CÓMO CREAR UNA PLANTILLA DE VARIABLES?

Haga clic en  . Indique el nombre y el tipo de tabla a utilizar:

- **Variables**: variables definidas en el Simbólico.
- **Periferia**: valores de periferia no mapeada en las variables de máquina.

Para clonar una plantilla, abra la misma y haga clic en  indicando el nuevo nombre.

4.11.2.- ¿CÓMO VISUALIZAR VARIABLES DE UN SECUENCIADOR?

Seleccione en el desplegable 

la tabla deseada.

La barra de herramientas de la tabla seleccionada tiene las siguientes opciones:

-  Añadir o eliminar filas en la tabla.
-  Actualización y actualización periódica de valores.
-  Forzado y forzado periódico de valores.
- 

Número de ciclos de forzado periódico.
- 

Valor en ms del envío del mensaje de forzado del valor a Galileo.

  Cada vez que se envíe un mensaje de forzado, Galileo mantendrá ese forzado el número de ciclos indicado (solo en el caso del forzado periódico). Por ejemplo, con un tiempo de ciclo de 20ms, si configuramos un número de ciclos muy bajo, 10 por ejemplo, solo forzaremos el valor durante 200ms, el resto del tiempo, hasta que se envíe el siguiente mensaje a los 2sg no se está realizando el forzado de dicho valor.

- Cambiar máquina. En la ventana mostrada seleccione una máquina en el desplegable, de las existentes en la plantilla actual, e indique en el segundo desplegable otra máquina de las definidas en el proyecto. El resultado en la plantilla es la sustitución de una máquina por otra manteniendo las variables. En caso de no existir la variable en la máquina nueva se indicará mediante un icono.

En la tabla **Variables**, y tras añadir una nueva fila, seleccione mediante el desplegable de la columna **"Máquina"** el nombre de la Máquina o Zona.

A continuación, mediante el desplegable de la columna **"Variable"**, seleccione la variable a visualizar.

Puede hacer clic con el botón derecho del ratón para seleccionar desde el menú contextual todas las variables de la máquina o zona de la fila seleccionada.

En la columna **"Tipo"** seleccione mediante el desplegable el formato en el que se van a visualizar los valores en la columna "**Valor actual**". La aplicación le ofrece los siguientes modos:

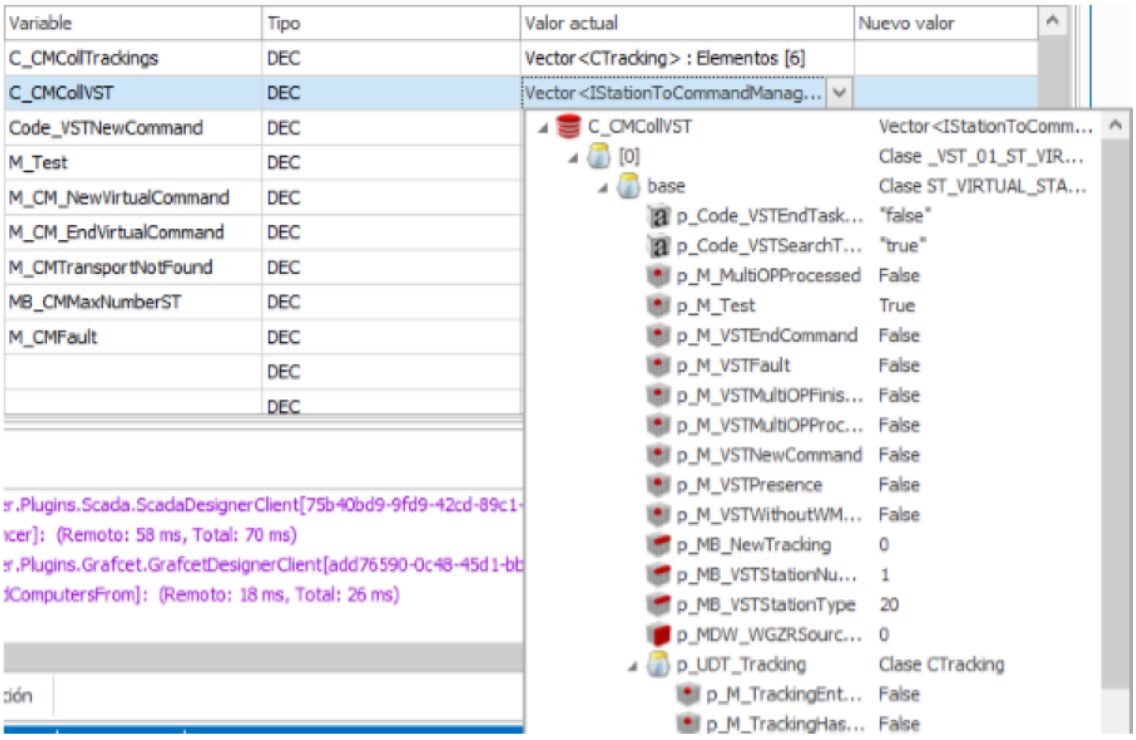
- **DEC**: El valor de la variable se muestra en formato decimal. Este es el formato por defecto, se selecciona automáticamente tras seleccionar una variable.
- **BIN**: El valor de la variable se muestra en formato *binario*, por ejemplo: una variable de tipo byte con valor 4 aparece como "00000100".
- **HEX**: El valor de la variable aparece en formato *hexadecimal*, por ejemplo, una variable de formato byte y valor 15 aparece como "F".
- **RAW**: El valor de la variable se muestra como el valor de sus *bytes* separados por espacios, por ejemplo, la variable de tipo word cuyo valor sea 12000 se mostrará como "046 014".
- **STR**: El valor de la variable se muestra en formato *carácter*. Por ejemplo, la variable de tipo byte cuyo valor sea 88 aparecerá como "X". Si la variable tiene formato word o dword debe tenerse en cuenta que la alineación de sus bytes puede hacer que no se muestre ningún carácter, al ser el primero de la cadena el "0" por tener el byte de más peso ese valor (una cadena que comienza por 0 se considera vacía).

 Las variables de tipo bit se mostrarán como valores 0 o 1, independientemente del formato seleccionado.

4.11.3.- **VISOR DE COLECCIONES**

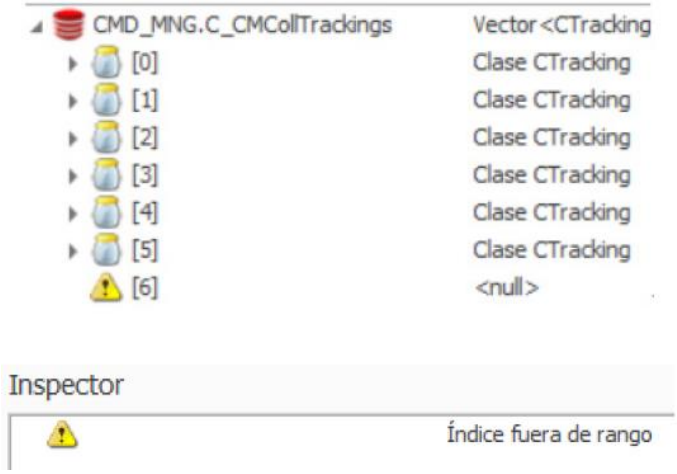
El Inspector facilita la visualización de variables de tipo Colección.

Permite acceder a toda la jerarquía del objeto de manera transparente:



Con ello la aplicación facilita una depuración más rápida e intuitiva de los programas.

También nos permite detectar, en una fase temprana del desarrollo del programa, situaciones potencialmente erróneas. Por ejemplo muestra errores como objetos nulos o accesos fuera de la colección:



El Visor de Colecciones puede integrarse con los elementos del SCADA, puede ver más información en “Uso de colecciones y Visor de colecciones en el Status” en la página 137.

4.11.4.- ¿CÓMO VISUALIZAR VARIABLES DE PERIFERIA NO DEFINIDA?

En la plantilla **Periferia**, seleccione mediante el desplegable de la columna

"**Tarjeta**" el nombre computador y tarjeta asociada que gestiona esa parte de la periferia. Indique, en la columna "**Dirección**", la dirección relativa a la zona a consultar.

La dirección se expresa en el siguiente formato:

#Modo Tipo Byte [.Bit]

- **Modo**: indica si se quiere consultar una entrada o una salida (I para entradas y O para salidas).
- **Tipo**: indica el tipo de dato a consultar X(bit),B(byte), W(word), D(dword).

Los valores posibles son por ejemplo:

- Consulta del *bit* 5 del *byte* 23 de las *entradas*: ***#IX23.5***

- Consulta del *byte* 12 de las *salidas*: ***#OB12***

4.11.5.- ¿CÓMO FOZAR EL VALOR DE UNA VARIABLE?

Además de visualizar los valores de las variables, es posible forzar un cambio en sus valores, a fin de simular condiciones y ver cómo reacciona el programa de control. Visualice, como se ha explicado anteriormente, el valor de la variable que desee modificar.

Indique un valor en la celda de la columna "**Nuevo Valor**" de la tabla. Todas las variables cuyo campo "**Nuevo Valor**" contenga un valor, serán forzadas a la vez.

Si se desea forzar una variable de forma periódica, haga clic en el icono de forzado periódico.

Consulte en "¿Cómo visualizar variables de un secuenciador?" en la página 113 las opciones disponibles en la barra de herramientas.

4.11.6.- ¿CÓMO ELIMINAR UNA VARIABLE DE LA PLANTILLA?

Seleccione la fila que contiene a la variable y haga clic en  para eliminarla.

4.11.7.- ¿CÓMO ELIMINAR UNA PLANTILLA?

Seleccione en el desplegable 

la tabla deseada y haga clic en  para eliminarla.

4.11.8.- ¿CÓMO GUARDAR Y CARGAR PLANTILLAS DE VARIABLES?

Clic  para guardar en el proyecto todos los cambios realizados en la pantalla.

Si desea descartar los cambios y recargar la plantilla original haga clic en  .

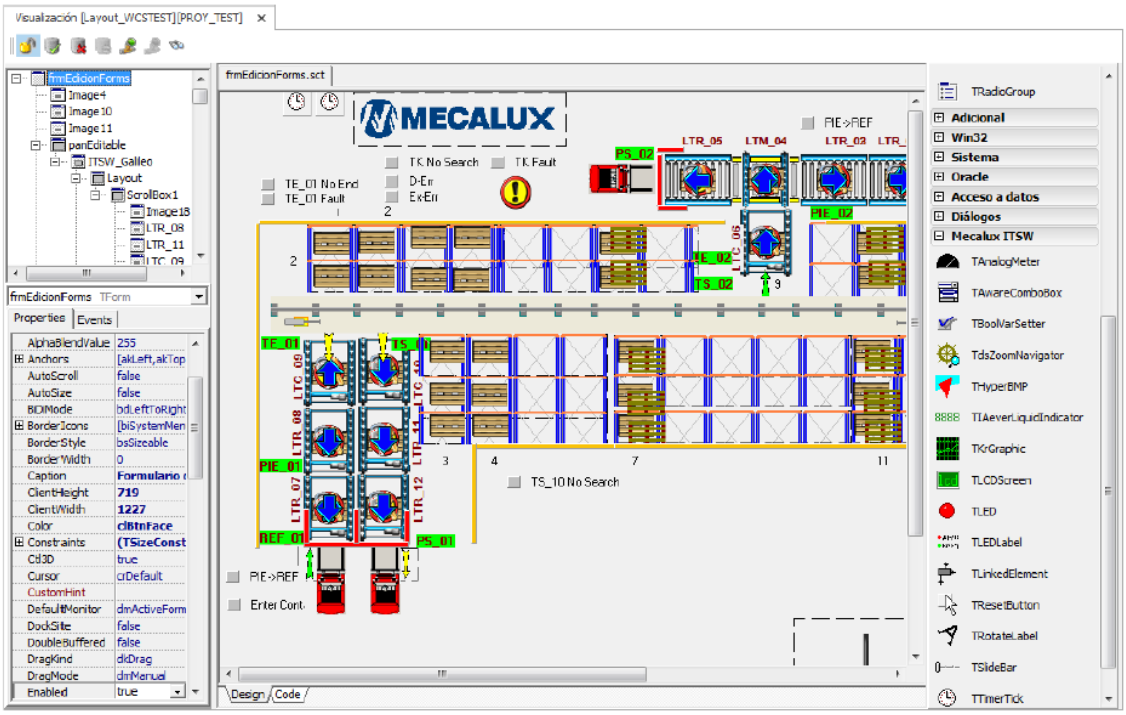
 Recuerde que al guardar la plantilla se almacenan también los valores actuales y de forzado.

4.12.- PANTALLA "VISUALIZACIONES"

Una visualización o layout es una representación esquemática de los elementos de la instalación gobernados por el Sistema de Control Galileo IV. El layout es la versión modificable del formulario de instalación que se usa en la aplicación **Galileo Status Monitor**.

Las visualizaciones son la cara visible de todo el proyecto, pues es la parte que finalmente queda expuesta al uso diario y de la que se obtiene información en tiempo real de la instalación.

La información que contienen tiene un significado especial durante el desarrollo, y es necesario organizar la información de forma ordenada a fin de localizarla posteriormente en un tiempo razonable.



Haga clic en para habilitar todas las funcionalidades y bloquear la pantalla en el proyecto.

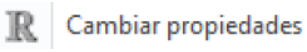
4.12.1.- ¿CÓMO CREAR Y ABRIR UNA VISUALIZACIÓN?

Para crear una visualización haga clic en la opción  accesible con clic derecho del ratón sobre el nodo **Visualizaciones** del árbol del proyecto. Indique el nombre.

Una vez creada, y desde el menú contextual mostrado haciendo clic derecho sobre ella, tiene las siguientes opciones:

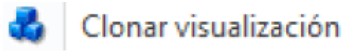
- 

: Elimina la visualización seleccionada.

- 
- **Nombre**: Permite la edición del mismo.
- **Tipo**: Estándar o Plantilla.

El tipo Plantilla es el empleado para aquellas visualizaciones base empleadas en la generación automática de proyectos. Para ver este proceso consulte "¿Cómo se puede generar un proyecto de forma automática?" en la página 31.

- **Nivel de acceso**: Asignación de niveles de acceso predefinidos como en Designer 3.x. Para que puedan ser ejecutados han de ir asociados a los Privilegios asignados al grupo.
- **Traducción automática**: Traducción directa de cadenas de texto en la visualización en el Modo Conectado.

-  : Realiza una copia de la visualización solicitando al usuario el nombre de la nueva.

Para mostrar y/o editar una visualización (su formulario) haga clic sobre ella en el árbol del proyecto.

Para poder editar o eliminar una visualización ha de tener los permisos de usuario necesarios.

Una vez que la visualización aparece en pantalla, puede usarlo en dos modos:

- **Modo Ejecución** → Muestra la visualización de la misma forma que desde la aplicación Galileo Status Monitor.
  En este modo no puede modificar los componentes del formulario, pero sí interactuar con ellos.

   Los cambios que realice en este modo (por ejemplo: cambiar la solapa activa), no son guardados. Al abrir la visualización la siguiente vez vuelven a estar en el estado inicial.

- **Modo Diseño** → Muestra la visualización con una paleta de componentes en la parte derecha, que se utiliza para componer la visualización según las necesidades. En este modo, también aparece una pestaña llamada **Code** donde puede escribir código de script, en el lenguaje JavaScript, que permite programar eventos de los componentes que se inserten. Los cambios que se realicen en este modo sí son guardados.

4.12.2.- ¿CÓMO EDITAR UNA VISUALIZACIÓN?

Al igual que en versiones anteriores del Designer, la aplicación muestra gran cantidad de componentes y opciones a la hora de crear y/o editar visualizaciones.

Una vez que la visualización está abierta, ésta se encuentra por defecto bloqueada y en **modo Diseño**. Por defecto la visualización se encuentra en este modo. En caso contrario, haga clic en  .

Para editar los componentes que contiene haga clic en  para habilitar todas las funcionalidades y bloquear la pantalla en el proyecto para el resto de usuarios.

4.12.2.1.- LA PALETA DE COMPONENTES

Los componentes de edición de la Visualización se agrupan en diferentes paletas según su funcionalidad. La descripción que se realiza es muy básica y, salvo en el caso de los componentes de la paleta "Mecalux ITSW", no se explican las propiedades o métodos inherentes de cada componente particular. Para más información sobre cada caso en particular, se recomienda la lectura de la ayuda incluida en el compilador de Borland C++ Builder, pues son los mismos componentes.

De arriba a abajo, tenemos las siguientes paletas de componentes:

1. Paleta **Estándar**
2. Paleta **Adicional**
3. Paleta **Win32**
4. Paleta **Sistema**
5. Paleta **Oracle**
6. Paleta **Acceso a Datos**
7. Paleta **Diálogos**
8. Paleta **Mecalux ITSW**

 Varios de los elementos disponibles en las paletas pueden ser escalados con "Ctrl + May + flechas dirección".

A continuación comentaremos cada paleta individualmente.

4.12.2.1.1.- PALETA ESTÁNDAR

Estos componentes son los básicos en una aplicación Windows, y son también los más frecuentemente usados.

ICONO	NOMBRE	DESCRIPCIÓN
	TActionList	Sirve para crear colecciones de acciones que centralizan la respuesta del formulario ante acciones del usuario.
	TButton	Crea un botón que el usuario puede pulsar a fin de iniciar acciones.
	TCheckBox	Muestra una opción que el usuario puede activar / desactivar. Su uso más común es el de mostrar grupos de opciones que no son mutuamente exclusivas.
	TComboBox	Muestra una lista de opciones en una combinación de un TListBox y un TEditBox. Los usuarios pueden escribir texto en el campo de edición o seleccionar un ítem de la lista.
	TEdit	Muestra un área de edición donde el usuario puede introducir o modificar una línea de texto simple.
	TGroupBox	Facilita un contenedor de opciones relativas a un grupo o formulario.
	TLabel	Muestra texto que el usuario no puede seleccionar o manipular, como títulos o etiquetas de controles.
	TListBox	Muestra una lista de opciones seleccionables con barra de desplazamiento.
	TMemo	Muestra un área de edición donde el usuario puede introducir múltiples líneas de texto.
	TMenú	Crea una barra con un menú principal para el formulario. Para acceder a sus eventos o diseñar sus contenidos, basta con hacer un doble clic sobre uno de estos elementos, a fin de abrir una ventana de diseño de Menú.
	TPanel	Crea un panel que puede contener otros componentes del formulario. Se pueden usar, por ejemplo, para crear barras de herramientas o líneas de estado.
	TPopupMenu	Crea Menús de tipo Popup que aparecen cuando el usuario pulsa el botón derecho del ratón. Se puede usar el mismo mecanismo que en el anterior componente para diseñarlos.
	TRadioButton	Como el anterior, pero en este caso las opciones son mutuamente excluyentes.
	TRadioGroup	Crea un TGroupBox que contiene TRadioButtons en un formulario.

4.12.2.1.2.- PALETA ADICIONAL

La siguiente paleta son componentes cuyo uso es más raro y especializado que los de la paleta Estándar, pero aun así muy frecuentemente usados en interfaces de aplicaciones Windows.

ICONO	NOMBRE	DESCRIPCIÓN
	TBevel	Crea líneas o cajas de apariencia tridimensional.
	TBitBtn	Crea un botón en el que se puede mostrar un bitmap.
	TCheckListBox	Muestra una lista con barras de desplazamiento, similar a una ListBox, excepto en que cada elemento de la lista tiene un CheckBox a su lado.
	TControlBar	Crea un manager de layouts para los componentes ToolBar.
	TDrawGrid	Crea una tabla donde se pueden mostrar datos en filas y columnas.
	TImage	Muestra una imagen (bitmap, icono o metafile).  Haga clic derecho sobre la visualización y desde el menú contextual mostrado seleccione **Show Images inspector**. En la ventana mostrada se detalla el tamaño de cada una de las imágenes del formulario, así como el tamaño total de todas ellas.
	TMaskEdit	Provee un mecanismo para forzar al usuario a entrar texto en un determinado formato dentro de un TEdit u otro control de texto.
	TScrollBox	Crea un contenedor de tamaño variable que automáticamente muestra barras de desplazamiento cuando es necesario.
	TShape	Dibuja figuras geométricas (elipses, círculos, rectángulos, cuadrados o rectángulos redondeados).
	TSpeedButton	Crea un botón que puede mostrar un icono, pero sin texto. Suelen agruparse en ToolBars.
	TSplitter	Añade un separador al formulario, entre dos controles alineados, permitiendo al usuario cambiar el tamaño que ocupan en tiempo de ejecución, simplemente moviendo el separador.
	TStaticText	Crea una etiqueta de texto no editable, al igual que una Label, con la diferencia de que este componente tiene su propio "handle" a la ventana a la que pertenece.

	TStringGrid	Crea una tabla donde se pueden mostrar cadenas con información organizada en filas y columnas.
		En el siguiente ejemplo, se configura el grid para tener 8 filas y 3 columnas, y después escribe en cada una de las celdas su coordenadas:
		
		`StringGrid1.ColCount = 3;`
		`StringGrid1.RowCount = 8;`
		`for(i = 0; i<StringGrid1.ColCount; i++)`
		`{`
		`for(j = 0; j<StringGrid1.RowCount; j++)`
		`{`
		`StringGrid1.Cells(i, j) = "(" + i.toString() + "," + j.toString() + ")";`
		`}`
		`}`

	TTabSheet	Este componente es una solapa en un componente TPageControl.

4.12.2.1.3.- PALETA WIN32

La siguiente paleta muestra componentes un poco más avanzados, que permiten crear interfaces más vistosos y complejos.

ICONO	NOMBRE	DESCRIPCIÓN
	TAnimate	Este control proporciona un medio para mostrar una animación AVI en una parte del formulario, que se estará reproduciendo cuando lo planifiquemos.
	TCoolBar	Muestra una colección de controles de tipo TCoolBand, cuyas bandas son móviles y de tamaño variable. El usuario puede posicionar los controles arrastrando la etiqueta que tienen en su parte izquierda.
	TDateTimePicker	Muestra un control de edición para introducir fechas o tiempos. El usuario puede seleccionar fechas de un calendario mediante el uso de las teclas arriba, abajo o simplemente escribiendo la fecha. Es necesario disponer de una versión actualizada de COMCTL32.DLL, normalmente localizada en el directorio Windows\System o en Windows\System32
	THeaderControl	Muestra una cabecera de texto o números sobre una lista. Se puede dividir el control en dos o más partes para permitir cabeceras en múltiples columnas. Los títulos se pueden alinear de varias maneras, configuradas por separado.
	THotKey	Hace que una “Hot Key” (combinación de teclado) se asocie a un control en particular.
	TImageList	Este elemento es una colección de imágenes de las mismas dimensiones, cada una de las cuales puede luego ser referida por un índice. El listado de imágenes se gestiona desde el menú contextual haciendo clic con el botón derecho.
	TListView	Permite mostrar una lista en formato de varias columnas. Cada columna mostrando un aspecto del dato que representa la fila.
	TMonthCalendar	Muestra un calendario que representa un mes. Puede ser configurado para permitir navegar entre las fechas.
	TPageControl	Crea un control de páginas que es usado para definir una caja de diálogo multipágina, aunque usando una única ventana.
	TPageScroller	Contiene otros objetos en un área cliente sobre los que se puede hacer desplazamiento horizontal o verticalmente.
	TProgressBar	Una barra rectangular que se puede “rellenar” de izquierda a derecha, indicando un progreso en una operación o proceso.
	TRichEdit	Crea un control que contiene texto enriquecido. Por defecto soporta propiedades tales como: tipo, estilo, tamaño y color para sus fuentes. También soporta alineamiento de texto, numeración y drag & drop.
	TStatusBar	Éste es un área donde colocar texto informativo sobre las acciones que lleva a cabo la aplicación.

	TTabControl	Similar a un divisor en un Notebook, este componente provee un conjunto de solapas mutuamente excluyentes al estilo de un notebook.
	TToolBar	Maneja ToolButtons y otros controles, colocándolos por filas, y ajustando automáticamente su tamaño y posición.
	TTrackBar	Esta es una barra que define una extensión o rango de un ajuste, y un indicador que muestra el valor actual y permite cambiarlo. Se puede establecer su orientación horizontal o verticalmente, definir su longitud, anchura o el indicador de posición o muchos otros parámetros.
	TTreeView	Permite mostrar y controlar un conjunto de objetos basándose en sus relaciones lógicas de jerarquía. El control incluye botones que permiten expandir / contraer ramas. Se usa principalmente para mostrar dichas relaciones entre el conjunto de objetos ordenados de forma jerárquica.
	TUpDown	Crea dos botones, uno con una flecha hacia arriba y el otro hacia abajo, que incrementan o decrementan valores al ser pulsados.

4.12.2.1.4.- PALETA SISTEMA

La siguiente paleta contiene componentes que permiten acceso a componentes propios del sistema operativo.

ICONO	NOMBRE	DESCRIPCIÓN
	TMediaPlayer	Muestra un control con estilo VCR que permite reproducir ficheros multimedia.
	TOleContainer	Crea un área cliente para un Objeto linkado y embebido (OLE).
	TPaintBox	Especifica un área rectangular que provee límites para el repintado de la aplicación.
	TTimer	Este componente es un no-visual y lanza un evento de forma periódica. Tanto el evento como el periodo son definidos por el programador.

4.12.2.1.5.- PALETA ORACLE

En la paleta Oracle hay componentes que permiten interactuar con una base de datos Oracle, permitiendo acceso a sus tablas, conexiones, etc., esto confiere una gran potencia al layout, pudiendo incluso llegar a integrar la aplicación de control con la de gestión del almacén.

ICONO	NOMBRE	DESCRIPCIÓN
	TDataSource	Este componente actúa como un conducto o paso intermedio entre componentes de tipo DataSet y otros componentes dependientes de datos, como un DBGrid.
	TOracleDataSet	Este componente es útil cuando se quiere hacer uso de otros componentes dependientes de datos, puesto que deriva del estándar DataSet, heredando la mayoría de sus propiedades, métodos y eventos, junto con otros derivados del componente TOracleQuery. Para usar un TOracleDataSet, hay que establecer su propiedad "Session" a un componente OracleSession del formulario y establecer la sentencia SELECT que obtendrá el conjunto de datos con el que trabajar.
	TOracleDirectPathLoader	1. Permite a una aplicación acceder al motor del "Direct Path Load" de un servidor Oracle, que permite al servidor cargar datos desde una fuente externa (un fichero, por ejemplo), a la base de datos a gran velocidad, en lugar de usar sentencias SQL INSERT o UPLOAD. 2. Hay varias restricciones para que dicha carga pueda ser llevada a cabo: 3. No se permiten disparadores en la tabla, es necesario desactivarlos temporalmente. 4. No se permiten comprobaciones de restricciones o restricciones de claves ajenas en la tabla, aunque si se permiten restricciones de clave primaria. 5. No se pueden cargar tablas remotas. 6. No se permiten tipos definidos por el usuario para la tabla. 7. Este interface sólo está disponible en las versiones Net8 8.1 cliente y posteriores. 8. La operación de carga no forma parte de una transacción "normal".

	TOracleEvent	Este componente puede ser usado desde una aplicación que necesita reaccionar a señales dbms_alert o mensajes dbms_pipe. Estos son típicamente generados por disparadores en la base de datos que pasan informaciones a otras sesiones de la base de datos. Estos componentes corren en un hilo de ejecución separado en background, sin interferir con el funcionamiento normal del programa. Cuando ocurre un evento, se llama a la función definida en la propiedad "OnEvent", que está sincronizada con el hilo de ejecución principal de la aplicación.
	TOracleLogon	Este componente provee de un cuadro de diálogo estándar para establecer el usuario, contraseña y cadena de conexión a una base de datos.
	TOracleNavigator	Este componente es usado para moverse entre los datos contenidos en un OracleDataSet y permite realizar operaciones sobre dichos datos, tales como insertar nuevos, modificar o borrar datos existentes.
	TOraclePackage	1. Un paquete es una extensión de las bases de datos Oracle que permite el encapsulamiento de funciones, procedimientos, definiciones de tipos, variables y constantes en una única extensión de programa. Esta unidad puede ser luego usada de forma arbitraria desde cualquier punto de un código PL/SQL. 2. Este componente permite el acceso directo a las funciones, procedimientos, etc. Definidas en un paquete. Simplemente hay que establecer la propiedad "Session" y especificar el nombre del paquete en la Base de Datos. Para llamar a procedimientos o funciones de la base de datos, se pueden usar los métodos "CallProcedure" y "CallFunction". Para acceder a variables o constantes, se puede usar los métodos "SetVariable" o "GetVariable".
	TOracleQuery	Se usa este componente para ejecutar sentencias SQL o bloques PL/SQL. Es necesario establecer la propiedad Session a un OracleSession en el cual se ejecutará la consulta. El texto de la consulta puede establecerse en la propiedad SQL y ser ejecutado al invocar al método "Execute". Si la sentencia es de tipo SELECT, se podrá acceder a los campos de un registro mediante el método "Field", y llamar a "Next" para obtener más registros hasta que el retorno de "Eof" indique que ya no hay más registros disponibles. Si ocurriera un error durante la ejecución de la sentencia, se lanzará una excepción "EOracleError".
	TOracleScript	Este componente permite definir y ejecutar un script SQL con múltiples sentencias SQL. Esto puede ser

		muy útil si se necesitara ejecutar muchas sentencias SQL que no pueden ser usadas en bloques PL/SQL, como por ejemplo el caso típico de scripts de instalación.
	TOracleSession	Este componente provee un enlace con una base de datos Oracle. Tras establecer sus propiedades conexión (nombre de usuario, contraseña y base de datos), se puede establecer una conexión con la base de datos llamando al método "LogOn" o estableciendo la propiedad "Connected" a True. Si se desea que sea el usuario final el que especifique los parámetros de conexión, se puede hacer uso del componente OracleLogon.

4.12.2.1.6.- PALETA ACCESO A DATOS

Una vez que existen componentes que acceden a la base de datos, se pueden utilizar los componentes de esta paleta para mostrarlos de muy variadas formas.

ICONO	NOMBRE	DESCRIPCIÓN
	TDBCheckBox	Componente dependiente de datos en forma de CheckBox que muestra o edita campos boléanos de un registro de la Base de Datos.
	TDBComboBox	Componente dependiente de datos en forma de ComboBox que muestra o edita listas de valores de una columna en una tabla de la Base de Datos.
	TDBCtrlGrid	Este componente muestra múltiples campos de múltiples registros en formato de tabla tabulada. Cada celda en la tabla muestra múltiples campos de un sólo registro.
	TDBEdit	Componente dependiente de datos en forma de campo de edición (Edit). Muestra los datos de un campo de un registro de la BD.
	TDBGrid	Componente dependiente de datos, en forma de tabla. Permite visualizar y editar los datos en un formato tabular, similar a una hoja de cálculo.
	TDBImage	Componente dependiente de datos con la misma apariencia de un Image que muestra, corta, copia o pega bitmaps desde o hacia un registro de la Base de Datos.
	TDBListBox	Componente dependiente de datos en forma de lista que muestra una lista con desplazadores de valores de una columna en una tabla de la Base de Datos.
	TDBLookupComboBox	Componente dependiente de datos, en forma de ComboBox, que deriva su lista de objetos a representar de un campo Lookup definido para un Data-Set o un DataSource. En cualquier caso, al usuario se le presenta una lista restringida de valores sobre los que elegir para establecer un campo.

	TDBLookupListBox	Componente dependiente de datos, en forma de ListBox, que deriva su lista de objetos a representar de un campo Lookup definido para un DataSet o un DataSource. En cualquier caso, al usuario se le presenta una lista restringida de valores sobre los que elegir para establecer un campo.
	TDBMemo	Componente dependiente de datos en forma de memo que muestra o edita textos en un registro de la Base de Datos.
	TDBNavigator	Componente dependiente de datos en forma de una barra de botones que permite moverse por los registros de una tabla de la Base de Datos. También puede poner la tabla en modo de inserción, edición o sólo lectura, colocar nuevos registros o refrescar los que se están mostrando.
	TDBRadioGroup	Componente dependiente de datos en forma de grupo de RadioButtons que muestran un conjunto de valores de columnas de una tabla de la Base de Datos.
	TDBRichEdit	Un control de edición multilínea que puede mostrar y editar texto enriquecido de un campo en un Data-Set.
	TDBText	Componente dependiente de datos en forma de etiqueta de texto que muestra los valores de un campo de un registro.

4.12.2.1.7.- PALETA DIÁLOGOS

Crea diálogos que solicitan información de algún tipo al usuario.

ICONO	NOMBRE	DESCRIPCIÓN
	TColorDialog	Este diálogo permite al usuario especificar información sobre un color determinado.
	TFindDialog	Muestra un diálogo de búsqueda de cadenas en un texto dado.
	TFontDialog	Muestra una ventana desde la que el usuario puede especificar el tipo, estilo y tamaño de una fuente y seleccionarla.
	TOpenDialog	Muestra un diálogo de apertura de ficheros, muy común en todas las aplicaciones Windows.
	TOpenPictureDialog	Muestra un diálogo modal para seleccionar y abrir un archivo de imágenes. Es idéntico a un TOpenDialog, excepto que tiene un área donde poder realizar una vista previa de la imagen antes de cargarla.
	TPrintDialog	Muestra un cuadro de diálogo que permite al usuario especificar información sobre una impresión, tal como la impresora donde realizarla y el número de copias a realizar.
	TPrinterSetupDialog	Muestra un diálogo de configuración de impresoras que permite al usuario cambiar y parametrizar sus impresoras.

	TReplaceDialog	Muestra un diálogo que permite realizar las operaciones de buscar y reemplazar cadenas en un texto.
	TSaveDialog	Muestra un diálogo para seleccionar un nombre de fichero para salvar información.
	TSavePictureDialog	Idéntica al anterior, pero el propósito ahora es salvar una imagen.

4.12.2.1.8.- PALETA MECALUX ITSW

La paleta **Mecalux ITSW** contiene elementos tanto visuales como no visuales, y son el verdadero corazón del que partir para crear un layout.

ICONO	NOMBRE	DESCRIPCIÓN
	TAnalogMeter	Este componente emula un medidor analógico y puede ser asociado a una variable definida en el simbólico para mostrar su valor numérico.
	TAwareComboBox	Combobox cuyo elemento seleccionado se controla desde una variable de un secuenciador.
	TBoolVarSetter	Este componente tiene un funcionamiento muy similar al CheckBox de la paleta Estándar, con la diferencia que está ligado a una variable de tipo BOOL definida en el simbólico y al cambiar su estado, el usuario fuerza también la variable que se está viendo desde el Sistema de Control Galileo IV.
	TdsZoomNavigator	Este componente permite incluir un cuadro de navegación rápido para paneles muy grandes. Aparece como un pequeño panel con un área seleccionable que se puede asociar con un Scroll-Box de tamaño mucho mayor que el que permita la pantalla. Al pulsar sobre esta área, nos permitirá desplazarnos de forma rápida por el ScrollBox de forma automática.
	THyperBMP	Este componente es muy similar a los Image, básicamente es un contenedor para una imagen.
	TIAeverLiquidIndicator	Este componente emula una pantalla de cristal líquido, que va mostrando un valor numérico asociado a una variable cualquiera definida en el simbólico.
	TKrGraphic	Componente que permite generar gráficas con variables del proyecto. Requiere tener asociado un componente Timer que establezca cada cuanto se desea actualizar los valores mostrados.
	TLCDScreen	Este componente no está asociado a ninguna variable del simbólico, simplemente sirve para mostrar cadenas o mensajes de forma vistosa debido a la gran variedad de efectos que permite.
	TLED	Un simple LED, asociado a una variable booleana, que aparecerá en color rojo o verde según el valor de la variable sea false o true.

	TLEDLabel	Un LED, con una etiqueta adjunta, asociado a una variable boleana, que aparecerá en color rojo o verde según el valor de la variable sea false o true.
	TLinkedElement	Este componente es esencial para construir el grafo de trayectorias posibles en la estación, mediante su uso, el programador debería ser capaz de reflejar en el layout todas las posibles trayectorias sobre las que se quiere tener control.
	TResetButton	Botón asociado a una máquina tipo que permite realizar un reset de la misma a una determinada etapa.
	TRotateLabel	Este componente no está asociado a variables del simbólico. Permite mostrar texto al igual que un Label, con posibilidades ampliadas de rotarlo en vertical o añadir efectos que lo hagan más vistoso.
	TSlideBar	Barra de desplazamiento (deslizante) asociada a una variable en un secuenciador.
	TTimerTick	1. Este componente es el más sencillo y a la vez más importante de un formulario para el Status Monitor. Mediante él se define el "pulso de refresco" que lleva asociado el formulario, es decir, la frecuencia con la que refrescarán los valores que mantiene en una caché interna y que representan los valores actuales de las variables del simbólico en la tarjeta. El refresco de esta caché provoca el refresco de todos los componentes asociados a variables en el simbólico que se encuentran activos en el formulario. 2. Para activar este refresco periódico, basta con poner la propiedad "Enabled" a true, y pasar a modo runtime. Si ocurriese algún problema en la conexión con el sistema remoto o con la extracción de datos, automáticamente este componente se desactivará, siendo necesario volver a activarlo desde el modo edición si queremos volver a ver los valores que corren en el sistema remoto. Es importante también fijarse en el valor que tiene la propiedad "Interval", ya que define la frecuencia (en milisegundos) con que se refresca la caché (por defecto se establece a 1 segundo). Valores menores pueden llevar a que el sistema no pueda obtener los datos con la suficiente velocidad para alimentar estas peticiones, mientras que valores muy superiores provocarán la falsa apariencia de que el sistema remoto funciona "a saltos". Debe existir al menos un componente de este tipo en el formulario para que las comunicaciones actúen.  Vea también las notas al final de esta tabla.
	TTrasloScroller	Componente que simula un elemento móvil. Tiene capacidad para mostrar hasta 10 señales asociadas a la

		máquina, así como realizar una simulación del movimiento real llevado a cabo por esta.

 En Designer IV cuando se crea una visualización se añade por defecto un TTimerTick habilitado con un intervalo de 1000 msg.

 No puede haber más de un TTimerTick por visualización. Si al importar una visualización de versiones anteriores del Designer, existen varios TTimerTick, Designer IV lo detecta y avisa de ello. En estos casos, de los existentes solo deja uno.

 El TTimerTick no puede tener eventos asociados. Si al importar una visualización de versiones anteriores del Designer, tienen un evento asociado, Designer IV lo detecta y avisa de ello. En estos casos, al salvar la visualización, este evento se desasocia automáticamente.

Los componentes comparten muchas propiedades con los ya conocidos y documentados por Borland. A continuación se detallan aquellas propiedades exclusivas de este grupo o específicas de alguno de ellos.

- El componente **TBoolVarSetter** no aparece listado porque sus únicas características extra son las comunes, que son descritas al principio.
- Con la excepción del **TTimerTick**, el **TLCDScreen**, el **TdsZoomNavigator** y el **TRotateLabel**, todos los componentes de esta paleta tienen las siguientes propiedades comunes:

PROPIEDAD	DESCRIPCIÓN
Clock	Esta es una propiedad especial que no debería ser modificada en ningún momento por el usuario. Se usa en conjunción con el TTimerTick para el refresco de los valores que está monitorizando el componente.
Secuenciador	Puede ser establecida a un nombre válido de Máquina o de Zona definido en el simbólico.  Haga clic derecho sobre la visualización y desde el menú contextual mostrado seleccione **Check layout**. En la ventana mostrada se muestran los errores a nivel de (Variable, Secuenciador), es decir, indica cuando está establecida la variable sin el secuenciador.
Variable	Puede ser establecida a cualquier variable de los tipos BOOL, BYTE, WORD, DWORD, COLLECTION o una CLASE definidas en el simbólico. Las variables disponibles se rellenan automáticamente en el propio desplegable en función del tipo del componente. Si no se establece, se entiende que el componente no está ligado a ninguna variable, y por tanto, no monitorizarán sus valores.

	Para el caso de variables de tipo CLASE (en 3.2 eran UDT), la variable tiene el formato `NombreClase.CampoClase`. También es posible representar anidaciones como por ejemplo `NombreClase.NombreClaseHija.CampoClaseHija` En colecciones la variable puede tener por ejemplo el siguiente formato: C_ColPunto3D[5].Z **Evaluador de Expresiones**: También es posible, en lugar de una variable, establecer una condición, usando operadores de condición AND, OR, XOR u operadores matemáticos. Se ofrece la posibilidad de utilizar el Evaluador con colecciones, por ejemplo: C_CMCollTrackings[5].p_MB_TrackingID div 2. Si no se prefijan las variables, se entenderán que pertenecen a la máquina definida con la propiedad "Secuenciador", mientras que si queremos especificar una variable de otra máquina, deberemos escribirla en su forma completa: `Maquina.Variable` Por ejemplo, si el secuenciador es TRE_01, y en la variable aparece: ActivarEntrada AND ActivarSubir Esto se entiende como que estamos evaluando TRE_01.ActivarEntrada AND TRE_01.ActivarSubir Sin embargo, podemos poner también: ActivarEntrada AND TRE_02.ActivarSalida Que se entenderá como: TRE_01.ActivarEntrada AND TRE_02.ActivarSalida Estas expresiones son susceptibles de parametrización.

- Propiedades de **LED Label**

PROPIEDAD	DESCRIPCIÓN
InvertLogic	Esta propiedad sirve para invertir la lógica de representación del componente, es decir, que si se establece a true esta propiedad, el led se encenderá cuando la variable que monitoriza se encuentre a false, y se apagará cuando la variable contenga el valor de true.
Lit	Esta propiedad sirve para forzar el encendido del led. Cuando se establece a true, el led aparece encendido, sin importar el valor de la variable que monitoriza.

- Propiedades de **AnalogMeter**

PROPIEDAD	DESCRIPCIÓN
AngularRange	Define la forma del medidor analógico. El indicador será un sector angular cuyo arco serán los grados marcados como valor de esta

	propiedad, hasta 360, que corresponderían a una circunferencia completa.
EnableZoneEvents	El indicador permite definir unas zonas mínimas o máximas por debajo o encima de las cuales se pueden disparar eventos que alerten de que la variable se encuentra en dichos umbrales. Estableciendo esta propiedad a true, podremos hacer uso de dichos eventos.

- Propiedades de **HyperBMP**

PROPIEDAD	DESCRIPCIÓN
Image	Corresponde a la imagen que queremos representar, que se puede cargar desde un fichero de imágenes (BMP, ICO o Metafile).
Height	Altura total del área donde se dibujará la imagen. Esta altura no tiene por qué coincidir con el tamaño de la imagen cargada en la propiedad Image.
Width	Anchura total del área donde se dibujará la imagen. Esta altura no tiene por qué coincidir con el tamaño de la imagen cargada en la propiedad Image.
Strech	Si esta propiedad se establece a true, la imagen cargada en la propiedad Image se intentará ajustar al tamaño definido por las propiedades Height y Width, con lo que realizaremos un zoom de dicha imagen.
InvertLogic	El HyperBMP puede llevar asociada una variable definida en el simbólico de tipo BOOL, y si esta variable es true, la imagen se mostrará, en cambio, si es false, la imagen no se mostrará. Si se establece esta propiedad a true, la lógica de visualización se invertirá, al igual que se hacía con los LED.

- Propiedades **IAEverLiquidIndicator**

PROPIEDAD	DESCRIPCIÓN
IndicatorAngle	Si se desea que el indicador este situado en un ángulo inclinado, se puede aumentar el valor de esta propiedad (0 → Horizontal, 900 → Vertical).
IndicatorGap	Indica el espacio (en píxeles) que queda entre el borde del indicador y los indicadores de valor.
IndicatorKind	Según el tipo, podemos usar un indicador 7 segmentos (por defecto), 9 segmentos, o como una matriz de puntos.
Indicatorstring	Esta es la cadena que se está mostrando en el indicador. Si el tipo del indicador es matriz de puntos, se muestra cualquier cadena, pero con los tipos de 7 y 9 segmentos, sólo se pueden mostrar números y los caracteres a, b, c, d, e, f (para números en hexadecimal).
IndicatorStyle	Mediante esta propiedad podemos hacer más gruesas o delgadas las líneas que forman los números o mensajes.
SignNumber	Indica en qué posición va el signo. Como el signo siempre se coloca en la posición más a la izquierda del indicador, esta propiedad también indica cuantos dígitos son visibles en el indicador.

- Propiedades **LCDScreen**

PROPIEDAD	DESCRIPCIÓN
DotMatrix	Mediante esta propiedad podemos elegir entre diferentes tipos de pantallas de matriz de puntos, lo que aumenta las posibilidades de visualización en pantalla.
Intensity	Esta propiedad aumenta la intensidad de brillo de los puntos en la pantalla LCD. A mayor valor, más brillantes aparecerán. El rango varía entre 0 y 255.
Lines	Este es el mensaje que aparecerá en la pantalla LCD, puede estar compuesto por varias líneas, sobre las que se puede ir desplazando de forma periódica.
ScrollActive	Al marcar esta variable a true, la pantalla LCD comienza a realizar el desplazamiento definido por las propiedades correspondientes.
ScrollHorz	Indica la cantidad de desplazamiento que se realiza en horizontal (en píxeles) de cada vez. Un valor positivo implica un desplazamiento de izquierda a derecha, y uno negativo de derecha a Izquierda.
ScrollSpeed	Indica la velocidad del desplazamiento. A mayor valor, más rápidamente se realizará el desplazamiento.

- Propiedades **LED**

PROPIEDAD	DESCRIPCIÓN
InvertLogic	Idéntico al LEDLabel.
Lit	Idéntico al LEDLabel.

- Propiedades **LinkedElement**

PROPIEDAD	DESCRIPCIÓN
Hint	Esta propiedad es el texto que aparecerá flotando sobre el nodo cuando situemos el cursor del ratón sobre dicho nodo.
Entrada1 Entrada2 Entrada3 Entrada4 Entrada5 Entrada6 Entrada7	Un nodo típicamente estará conectado a otros para formar el grafo o camino, y mediante estas 7 propiedades podremos especificar hasta 7 nodos (Componentes LinkedElement también, ya existentes en el formulario), que serán la entrada a este elemento. Esta es una información muy importante, pues es en la que se basa el generador de código para expandir los elementos this, ant1, .. ant7 que coloquemos en nuestro código. Para un nodo configurado en cualquier propiedad (Entrada1/Entrada2...) puede ir viendo, de forma anidada, las propiedades de ese LinkedElement configurado.
Image	Podemos seleccionar un Componente HyperBMP o Image para colocarlo aquí. Es la imagen que queremos que represente ese nodo.
InvertLogic	Idéntico al HyperBMP.
Peso	Esta propiedad está pensada para poder seleccionar diferentes rutas en tiempo real **en un futuro**, en función del peso que asignemos a cada nodo del grafo.

Movement01 Movement02 Movement03 ... ... Movement20	En cada propiedad se pueden agregar una imagen más a la del propio LinkedElement. Cada imagen tiene una serie de propiedades específicas que puede modificar. Cada propiedad de Movimiento tiene además otras propiedades como: **InverLogic**: idéntico al HyperBMP **OffsetX/Y**: desplazamiento respecto al centro del LinkedElement. **Orientation**: rotación de la imagen (independiente a la rotación del propio LinkedElemet) **Tag**: campo meramente informativo, a efectos es una etiqueta. **Variable**: como el HyperBMP puede llevar asociada una variable definida en el simbólico de tipo BOOL,ç **KeepAspect**: permite mantener la relación de aspecto de cada imagen. **Position**: permite establecer la posición (centrada, lateral o en las esquinas) sin necesidad de utilizar offset. Así al redimensionar se mantienen las posiciones. **LockRotation**: permite indicar si durante la generación automática, esa imagen debe rotarse en función de cómo rote todo el elemento, o por el contrario, debe mantener siempre su posición.
MovimientoAvería	Mismas propiedades que Movimiento1/2/3... Normalmente se emplea para la imagen de Defecto.
MovimientoPalet	Mismas propiedades que Movimiento1/2/3... Normalmente se emplea para la imagen del palet.
Salida1 Salida2 Salida3 Salida4 Salida5 Salida6 Salida7	Un nodo típicamente estará conectado a otros para formar el grafo o camino, y mediante estas 7 propiedades podremos especificar hasta 7 nodos (Componentes LinkedElement también, ya existentes en el formulario), que serán la salida a este elemento. Esta es una información muy importante, pues es en la que se basa el generador de código para expandir los elementos this, pos1, .. pos7 que coloquemos en nuestro código. Para un nodo configurado en cualquier propiedad (Salida1/ Salida2...) puede ir viendo, de forma anidada, las propiedades de ese LinkedElement configurado.a
Stretch	Tiene el mismo comportamiento que en el HyperBMP.
Tipo	Los nodos pueden ser de varios tipos según su posición en el grafo: Común, Especial, Inicio o Fin. Esta propiedad, al igual que la de Peso, está diseñada **para futuras implementaciones**, por lo que su uso aún no tiene ningún efecto.

- Propiedades **RotateLabel**

PROPIEDAD	DESCRIPCIÓN
Escapement	Indica el ángulo en el que queremos colocar el texto: 0 → horizontal 90 → vertical
TextStyle	Permite darle algún efecto extra al texto: elevado, incrustado o sin efecto.

- Propiedades **RotateLabel**

PROPIEDAD	DESCRIPCIÓN
Elements	Permite añadir un elemento a mostrar en la gráfica. Al seleccionar uno, aparece una ventana para indicar el secuenciador y la variable a la que desea asociarse.

Lenguajes de Scripting

Los lenguajes de scripting son una generación de lenguajes que implementan un interface COM común de manera que se pueden intercambiar entre sí para permitirlos interactuar con otros objetos COM nativos. Actualmente existe una gran cantidad de lenguajes de Scripting para el Sistema Operativo Windows que cumplan con los interfaces estándar. Entre otros podemos destacar "JavaScript", "VBScript", "Perl", "Phyton", etc.

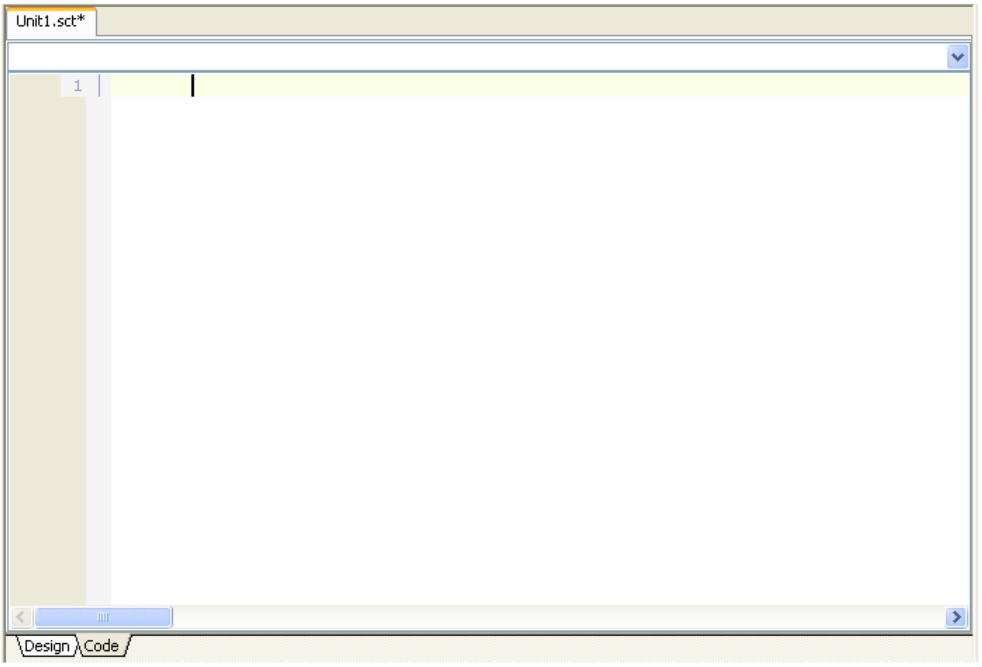
Si el uso de componentes confiere grandes ventajas a la visualización, la posibilidad de usar "Scripts" para programarlos eleva su potencia hasta el punto de que la visualización del layout tiene las mismas posibilidades que el uso de una comercial.

El lenguaje de programación usado es JavaScript, con él se puede hacer uso de los formularios, objetos y componentes que tenemos a nuestra disposición en la paleta de componentes, siguiendo las reglas del lenguaje. No es el objetivo de este manual el explicar los entresijos de JavaScript como lenguaje de programación, por lo que en caso de que el usuario tenga dudas en este aspecto, se recomienda la consulta de cualquier tutorial o manual sobre dicho lenguaje.

La ventana con nuestro código de script se muestra cuando abrimos una visualización y se encuentra habilitada y en modo Diseño (más info en "¿Cómo editar una visualización?" en la página 119).

En ese estado podemos escribir funciones y también, cuando hacemos doble clic sobre alguno de los eventos de un componente, se crea en la ventana una cabecera para la función que responde al evento y que se muestra para rellenar el cuerpo de la función.

Para borrar una función de respuesta a un evento, lo más cómodo es borrar el código de su cuerpo, puesto que al salvar el formulario, se eliminan las funciones cuyo cuerpo se encuentre vacío, y se desligan de los eventos a los que estuvieran asignadas, al igual que pasa desde el IDE de los compiladores de Borland.



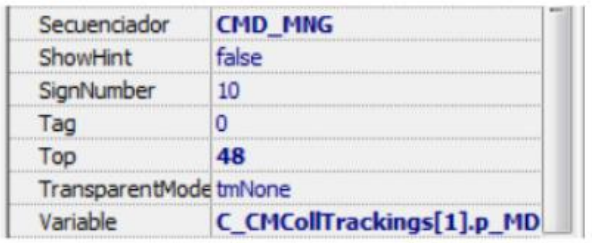
Además de las funciones que incorpora el lenguaje JavaScript, también es posible usar una serie de funciones añadidas desde el entorno:

FUNCIÓN
void **ChangeModeDesigner**(TForm aFormulario); **- Obsoleto -**
void **ShowReportFailures**(TForm aFormulario);
int **GetVarValue**(TForm aFormulario, string aVariable);
void **SetVarValue**(TForm aFormulario, string aVariable, int aValor);
int **GetShowTrackings**(TForm aFormulario);
void **SetShowTrackings**(TForm aFormulario, int aValor);
void **CopyTracking**(TForm formulario, string aSource, string aTarget); - aSource: nombre máquina origen - aTarget: nombre máquina destino Operativa sobre tracking 1
void **MoveTracking**(TForm aFormulario, string aSource, string aTarget);
void **SwapTracking**(TForm aFormulario, string aSource, string aTarget);
bool **GetPermissions** (TForm aFormulario, int aLevel); **- Muestra la ventana de logon para que se introduzca un usuario. Comprueba si tiene permisos suficientes para el nivel requerido.**
bool **QueryPermissions** (TForm aFormulario, string aUser, string aPass,

int aLevel); **- Comprueba si el usuario/password introducidos tiene permisos suficientes para el nivel requerido.** **No muestra ventana de logon.**
bool **QueryAccessLevel** (TForm aFormulario, int aLevel); - Comprueba si el usuario logueado actualmente tiene permisos suficientes para el nivel requerido. No muestra ventana de logon.
void **TheCardInfo.TranslateString**(string aLabel);
void **ShowCollectionViewer**(TForm aFormulario, string aLabel, bool aNew_window); **- Más información en este mismo cápitulo en el "Uso de colecciones y Visor de colecciones en el Status"** en la página 137 **y en el** "Visor de Colecciones" en la página 114 en el Inspector.

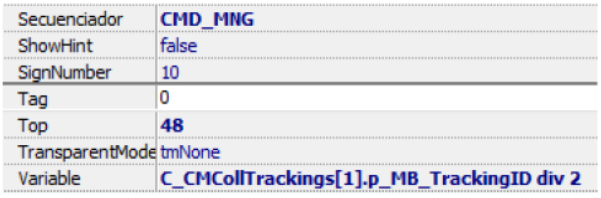
Uso de colecciones y Visor de colecciones en el Status

Cualquier elemento existente puede acceder a los datos de las colecciones:



Desde código, se puede obtener el valor de cualquier campo de una colección mediante el uso de "GetVarValue."

Es compatible con el evaluador de expresiones de Designer IV:

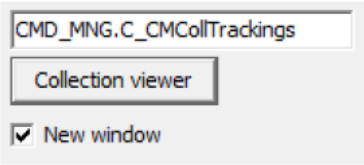


Designer IV ofrece mostrar visores de las colecciones de forma independiente.

El visor no requiere trabajo para configurarlo y actualiza los valores de manera automática, a medida que estos cambian en el programa.

Un ejemplo de un Visor en el Status podría ser:

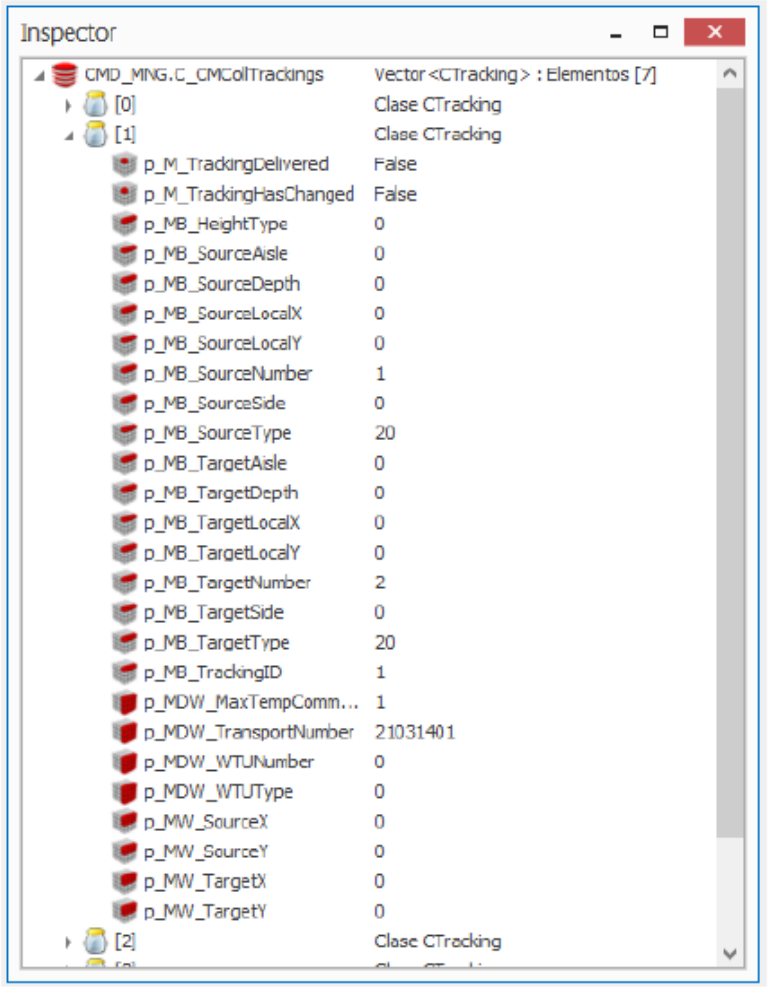
- Elementos en el Status:



- Código de JavaScript asociado:

```
var activeCollectionViewer = 0
function Button1Click(Sender)
{
if (cbNew.Checked)
ShowCollectionViewer(frmEdicionForms, EExpression.Text, 0);
else
activeCollectionViewer = ShowCollectionViewer(
frmEdicionForms,
EExpression.Text,
activeCollectionViewer);
}
```

- Visor de Colecciones mostrado:



4.12.2.2.- JAVASCRIPT, EL LENGUAJE DE PROGRAMACIÓN DEL STATUS MASTER

Aunque realmente Galileo IV soporta todos estos lenguajes, se estandariza la utilización de JavaScript por los siguientes motivos:

- Es un estándar abierto y por lo tanto no está sujeto al control de un único fabricante.
- Es muy similar al lenguaje Xana y por lo tanto no existe un cambio traumático entre la programación de control y la de la visualización.
- La variedad de tipos que soporta es mayor que la de VBScript autolimitado para no competir de forma directa con el lenguaje VBA (Visual Basic for Applications). Mientras que VBScript sólo puede automatizar métodos y propiedades expuestas como Variants, JavaScript puede automatizar interfaces expuestos con tipos complejos.
- Compatibilidad histórica con el antiguo Status Master de Builder.
- *JScript* es la implementación de Microsoft de la especificación de lenguaje ECMA 262 (ECMAScript Edition 3). Aparte de algunas excepciones de poca importancia (para mantener la compatibilidad con versiones anteriores), JScript es una implementación completa del estándar ECMA. Esta introducción pretende ayudarle a comenzar a trabajar con JScript.
- *JScript* es un lenguaje de secuencias de comandos interpretado y basado en objetos. Aunque tiene menos funciones que los lenguajes orientados a objetos de altas prestaciones como C++, JScript es muy eficiente para los propósitos a los que se destina.
- *JScript* no es una versión reducida de cualquier otro lenguaje ni es una simplificación de ningún lenguaje. Sin embargo, es un lenguaje limitado. Por ejemplo, en JScript no se permite escribir aplicaciones independientes ni proporciona compatibilidad integrada para la lectura y escritura de archivos. Además, las secuencias de comandos de JScript sólo pueden ejecutarse con un intérprete o "host", como las páginas Active Server (ASP, Active Server Pages), Internet Explorer, Windows Script Host o Designer y Status Master.
- *JScript* es un lenguaje en el que no se necesita declarar los tipos de datos. De hecho, JScript incluye una mejora en esta característica. No se puede declarar explícitamente los tipos de datos en JScript. Además, en muchos casos JScript realiza conversiones de forma automática cuando es necesario. Por ejemplo, si agrega un número a un elemento que contiene texto (una cadena), el número se convierte en texto.

El código incluido en muchos de los ejemplos siguientes es, de alguna manera, más explicativo y menos denso que el que encuentra en páginas Web sobre este tema. La intención aquí es dejar claros los conceptos, no expresar un estilo óptimo y conciso de codificación. En cualquier caso, incluso es aconsejable escribir código que puede leer y comprender, fácilmente, seis meses después de escribirlo.

Al igual que muchos otros lenguajes de programación, Microsoft JScript se escribe en forma de texto y se organiza en instrucciones, bloques formados por conjuntos de

instrucciones relacionadas y comentarios. En una instrucción puede utilizar variables, datos inmediatos como cadenas y números (denominados "literales"), y expresiones.

4.12.2.3.- **INSTRUCCIONES JSCRIPT**

Un programa de JScript es una colección de instrucciones. Cada instrucción de JScript equivale a una frase completa en español. Las instrucciones de JScript combinan las expresiones de tal forma que realizan una tarea completa.

Una instrucción se compone de una o varias expresiones, palabras claves u operadores (símbolos). Normalmente, cada instrucción se escribe en una sola línea, aunque puede abarcar dos o más líneas. Asimismo, dos o más instrucciones pueden escribirse en la misma línea, aunque deben separarse con un punto y coma. En general, en cada línea nueva comienza una nueva instrucción. Es aconsejable finalizar las instrucciones de una forma explícita.

Para ello se utiliza un punto y coma (;), el carácter de terminación de instrucción de JScript. A continuación se incluyen dos ejemplos de instrucciones de JScript.

→ Ejemplo:

```
aPájaro = "Petirrojo"; // Asignar texto "Petirrojo" a variable
// aPájaro
var hoy = nueva Fecha(); // Asignar fecha actual a la variable hoy
```

Un grupo de instrucciones de JScript rodeadas por llaves ({}) se llama bloque. Por lo general, las instrucciones agrupadas en un bloque pueden tratarse como una sola instrucción. Esto significa que los bloques pueden utilizarse en la mayor parte de los lugares en los que JScript espera encontrar una instrucción. Las excepciones más notables a esta regla son los encabezados de los bucles for y while. Las instrucciones primitivas incluidas en un bloque finalizan con puntos y coma, pero el bloque no.

Los bloques suelen utilizarse en funciones y condiciones. Observe que, al contrario que en C++ y en otros lenguajes, en JScript no se considera un bloque como un ámbito nuevo; sólo las funciones crean ámbitos nuevos. En el siguiente ejemplo la primera instrucción inicia la definición de una función, que está formada por un bloque de cinco instrucciones. Tras el bloque hay tres instrucciones, que no están incluidas en las llaves, no son un bloque y, por lo tanto, no forman parte de la función.

→ Ejemplo:

```
function convert(pulgadas)
{
pies = pulgadas / 12; //Estas cinco instrucciones están en un
// bloque.
millas = pies / 5280;
millasNauticas = pies / 6080;
cm = pulgadas * 2.54;
metros = pulgadas / 39.37;
}
km = metros / 1000; // Estas tres instrucciones no están en un
// bloque.
```

```
kradio = km;
mradio= millas;
```

4.12.2.4.- **COMENTARIOS**

Un comentario de una sola línea en JScript comienza con un par de barras inclinadas (//). A continuación se incluye un ejemplo de comentario de una sola línea.

→ Ejemplo:

```
unaBuenaIdea = "Incluya muchos comentarios en su código.";
// Este es un comentario de una sola línea.
```

Un comentario multilínea de JScript comienza con una barra inclinada seguida por un asterisco (/*) y termina con los mismos elementos en orden inverso (*/).

→ Ejemplo:

```
/*
Este es un comentario de varias líneas que explica la instrucción de
código anterior.
La instrucción asigna un valor a la variable unaBuenaIdea. El valor,
que está encerrado entre comillas, se conoce como literal. Un literal
contiene información de forma directa y explícita, no hace referencia
a la información de forma indirecta. Las comillas no forman parte del
literal.
*/
```

Al intentar incrustar un comentario de varias líneas en otro comentario del mismo tipo, JScript podría interpretar el comentario resultante de una manera inesperada.

Los caracteres */ que marcan el final del comentario incrustado se interpretan como el final de todo el comentario de varias líneas. Esto significa que el texto que sigue al comentario incrustado no se incluye en el comentario sino que se interpreta como código de JScript y genera errores de sintaxis.

Es aconsejable escribir todos los comentarios como bloques de comentarios de una sola línea. Posteriormente, esto permite comentar extensos segmentos de código mediante un comentario de varias líneas.

→ Ejemplo:

```
// Este es otro comentario de varias líneas, escrito como una
// serie de comentarios de una sola línea.
// Después de ejecutar la instrucción, puede hacer referencia al
// contenido de la variable unaBuenaIdea utilizando su nombre,
// como en la siguiente instrucción, en la que se agrega un
// literal de cadena a la variable unaBuenaIdea mediante
// concatenación para crear una nueva variable.
```

```
var ideaAmpliada = unaBuenaIdea
+
"Nunca puede saber cuándo tendrá que averiguar
lo que hace";
```

4.12.2.5.- **ASIGNACIONES E IGUALDAD**

El signo igual (=) se utiliza en las instrucciones de JScript para asignar valores a las variables: es el operador de asignación. El operando situado a la izquierda del operador = siempre es un ValorIzq. Ejemplos de valoresIzq:

- Variables.
- Elementos de matriz.
- Propiedades de objeto.

El operando situado a la derecha del operador = siempre es un valorDcho. Puede utilizarse cualquier tipo de valor arbitrario como valorDcho, incluido el valor de una expresión. A continuación se incluye un ejemplo de instrucción de asignación de JScript.

→ Ejemplo:

```
unEntero = 3;
```

El compilador de JScript interpreta esta instrucción de la siguiente manera: "asigna el valor 3 a la variable unEntero" o "unEntero recibe el valor 3".

Es necesario que comprenda la diferencia entre el operador = (asignación) y el operador == (igualdad). Cuando desee comparar dos valores a fin de determinar si son iguales, use dos signos igual (==). Este tema se analizará con más detalle en Controlar el flujo del programa.

4.12.2.6.- **EXPRESIONES**

Una expresión de JScript es una 'frase' que un intérprete de JScript puede evaluar para generar un valor. El valor puede ser cualquier tipo de JScript válido: un número, una cadena, un objeto, etc. Las expresiones más sencillas son los literales.

A continuación se incluyen algunos ejemplos de expresiones con literales de JScript.

→ Ejemplo:

```
3.9 // literal numérico
"¡Hola!" // literal de cadena
false // literal de tipo Boolean
null // valor null literal
{x:1, y:2} // literal de objeto
[1,2,3] // literal de matriz
function(x){return x*x;} // literal de función
```

Las expresiones más complejas pueden incluir variables, llamadas a la función y otras expresiones. Las expresiones pueden combinarse para crear expresiones complejas mediante operadores. Estos son algunos ejemplos de operadores:

```
+ // suma
- // resta
* // multiplicación
/ // división
```

A continuación se incluyen algunos ejemplos de expresiones complejas de JScript.

→ Ejemplo:

```
var unaExpresion= "3 * (4 / 5) + 6;
var unaSegundaExpresion = Math.PI * radio * radio;
var unaTerceraExpresion = unaSegundaExpresion
+
"%"
+
unaExpresion;
var unaCuartaExpresion = "(" + unaSegundaExpresion + ")
%
(" + unaExpresion + ")";
```

4.12.2.7.- **VARIABLES**

En todos los lenguajes de programación se utilizan segmentos de datos para cuantificar los conceptos.

```
¿Qué edad tengo?
```

En JScript, una variable es el nombre que recibe este concepto. Al utilizar la variable, en realidad se usan los datos que representa.

→ Ejemplo:

```
NúmeroDíasRestantes = FechaLímite – FechaActual;
```

En un sentido mecánico, las variables se emplean para almacenar, recuperar y tratar todos los valores diferentes que aparecen en las secuencias de comandos. Es aconsejable crear siempre nombres de variable descriptivos; esto facilita a los usuarios entender el tipo de tarea que realiza la secuencia de comandos.

4.12.2.8.- **DECLARAR VARIABLES**

Una variable se declara la primera vez que aparece en la secuencia de comandos. Esta primera mención de la variable la configura en la memoria, de modo que se pueda hacer referencia a ella en secciones posteriores de la secuencia de comandos. Para poder utilizar las variables, es necesario declararlas previamente. Para ello utilice la palabra clave **var**.

→ Ejemplo:

```
var contar; // una sola declaración.
var contar, cantidad, nivel; // varias declaraciones con una
//palabra clave var.
var contar = 0, cantidad = 100; // declaración de variable e
// inicialización en
// una instrucción.
```

Si no inicializa la variable en la instrucción **var**, ésta toma automáticamente el valor no definido (undefined) de JScript. Aunque no es aconsejable, la omisión de la palabra clave var en la instrucción de declaración es una sintaxis de JScript válida. Cuando se utilice, el intérprete de JScript proporciona a la variable visibilidad de ámbito global. Sin embargo, cuando se declare una variable en un procedimiento, no ha de ser visible en el ámbito global; en este caso, debe usar la palabra clave **var** en la declaración de variable.

4.12.2.9.- **NOMBRAR VARIABLES**

Un identificador es el nombre de una variable. En JScript, los identificadores se utilizan para:

- Nombrar variables,
- Nombrar funciones,
- Proporcionar etiquetas para los bucles.

JScript es un lenguaje que distingue mayúsculas y minúsculas. Esto significa que un nombre de variable como *miContador* es diferente al nombre *MIContador*. Los nombres de variable pueden tener cualquier longitud. Las normas para crear nombres de variable válidos son las siguientes:

- El primer carácter debe ser una letra ASCII (en mayúsculas o minúsculas) o un carácter de subrayado (_). No puede utilizarse un número como primer carácter.
- Los siguientes caracteres deben ser letras, números o caracteres de subrayado.
- El nombre de una variable no puede ser una **palabra reservada**.

A continuación se incluyen algunos ejemplos de nombres de variables válidos y no válidos:

→ Ejemplo:

```
// Nombres válidos
_numeropaginas
Parte9
Numero_elementos
// Nombres no válidos
99Globos // No puede comenzar con un número.
Smith&Wesson // El signo & no es un carácter válido para los
// nombres de las variables.
```

Si desea declarar una variable e inicializarla, pero no quiere darle ningún valor en particular, asígnele el valor null de JScript.

→ Ejemplo:

```
var edadOptima = null;
var demasiadoMayor = 3 * edadOptima; // demasiadoMayor tiene el
// valor 0.
```

Si se declara una variable sin asignarle un valor, la variable existe, pero tiene el valor no definida de JScript.

→ Ejemplo:

```
var conteoActual;
var conteoFinal = 1 * conteoActual; // conteoFinal tiene el valor
// NaN, ya que conteoActual no
// está definida.
```

La diferencia principal entre **null** y **no definida** en JScript es que **null** se comporta como el número 0, mientras que no definida se comporta como el valor especial **NaN** (No es un número, Not a Number). Al compararlos, los valores **null** y **no definido** siempre se consideran iguales.

Puede definir una variable sin utilizar la palabra clave var en la declaración y asignarle un valor. Esto se denomina declaración implícita.

→ Ejemplo:

```
ningunaCadena = ""; // La variable ningunaCadena se declara de
// forma implícita.
```

No se puede utilizar una variable que nunca se haya declarado.

→ Ejemplo:

```
var volumen = longitud * ancho; // Error, longitud y ancho todavía
// no existen.
```

4.12.2.10.- **CONVERSIÓN**

El intérprete de JScript sólo puede evaluar expresiones en las que los tipos de datos de los operandos coincidan.

Sin la conversión, una expresión que intente llevar a cabo una operación en dos tipos de datos diferentes (un número y una cadena, por ejemplo) produce un resultado erróneo. Pero éste no es el caso en JScript.

JScript es un lenguaje en el que no es necesario declarar los tipos de datos. Esto significa que sus variables no tienen un tipo predeterminado (al contrario que lenguajes en los que se deben declarar los tipos de datos, como C++). En cambio, las variables de JScript tienen un tipo equivalente al tipo del valor que contienen. La ventaja de este comportamiento es que proporciona la flexibilidad necesaria para tratar un valor como si perteneciera a otro tipo.

En JScript, se pueden realizar operaciones en valores de tipos diferentes sin temor a que el intérprete de JScript provoque una excepción. En vez de eso, el intérprete de JScript cambia automáticamente uno de los tipos de datos al otro y realiza después la operación. Por ejemplo:

OPERACIÓN	RESULTADO
Sumar un número y una cadena	El número se convierte en cadena
Sumar un booleano y una cadena	El booleano se convierte en cadena
Sumar un número y un booleano	El booleano se convierte en número

→ Ejemplo:

```
var x = 2000; // Un número
var y = "Hola"; // Una cadena
x = x + y; // el número se convierte en una cadena
document.write(x); // Da como resultado 2000Hola
```

Para convertir de forma explícita una cadena en un entero, utilizar el método parseInt.

Para convertir de forma explícita una cadena en un número, utilizar el método parseFloat.

Hay que tener en cuenta que las cadenas se convierten automáticamente en números equivalentes al realizar una comparación, pero no se modifican durante una suma.

Tipos de datos JScript

En JScript hay tipos de datos principales, compuestos y especiales:

TIPO DE DATO	RESULTADO
Principal	Cadena
	Numérico
	Booleano
Compuesto	Objeto
	Matriz
Espacial	Null
	No definido

Tipo de dato cadena

Un valor de cadena está formado por una cadena de cero o más caracteres Unicode (letras, dígitos y signos de puntuación). Este tipo se utiliza para representar texto en JScript. Para incluir literales de cadena en sus secuencias de comandos, enciérrelos entre pares de comillas simples o dobles. Pueden incluirse comillas dobles en cadenas delimitadas por comillas simples y viceversa.

→ Ejemplo:

```
"Feliz me encuentro; libre de preocupaciones"
'"¡Es imposible!" gritó el hombre.'
"42"
'c'
```

Tenga en cuenta que en JScript no hay ningún tipo que represente un carácter único (como el tipo **char** de C++). Para representar un carácter único en JScript, es necesario crear una cadena que sólo conste de un carácter. Las cadenas que no contienen ningún carácter ("") se denominan cadenas vacías (de longitud cero).

Tipo de dato numérico

En JScript no hay ninguna distinción entre valores enteros y valores de coma flotante; un número de JScript puede ser cualquiera de los dos (internamente, JScript representa todos los números como valores de coma flotante).

4.12.2.10.1.- VALORES ENTEROS

Los valores enteros pueden ser números enteros positivos, números enteros negativos y 0. Se pueden representar en *base 10* (decimal), *base 8* (octal) y *base 16* (hexadecimal). La mayor parte de los números de JScript se escriben en decimal.

Si se desea denotar enteros octales, anteponer un “0” (cero). Sólo pueden contener dígitos del 0 al 7. Un número precedido por un “0” que contiene los dígitos “8” o “9” se interpreta como un número decimal.

Si se desea denotar enteros hexadecimales (“hex”), anteponer un “0x” (cero y x o X). Sólo pueden contener dígitos del 0 al 9 y letras de la A a la F (mayúsculas o minúsculas). Las letras de la A a la F se utilizan para representar, como dígitos simples, los números que van del 10 al 15 en base 10. Es decir, 0xF equivale a 15 y 0x10 equivale a 16.

Los números octales y hexadecimales pueden ser negativos, pero no pueden contener una parte decimal ni escribirse en notación científica (exponencial).

4.12.2.10.2.- VALORES DE COMA FLOTANTE

Los valores de coma flotante pueden ser números enteros con una parte decimal. Además, pueden expresarse en notación científica. Es decir, se utiliza una "e" en mayúsculas o minúsculas para representar “diez a la potencia de”.

JScript representa números mediante el estándar de coma flotante IEEE 754 de ocho bytes para representación numérica. Esto significa que puede escribir números tan grandes como ±1,7976931348623157x10308 y tan pequeños como ±5x10-324. Los números que comienzan con un sólo "0" y contienen un separador decimal se interpretan como números decimales de coma flotante.

Tenga en cuenta que un número que comience con "0x" o "00" y contenga un separador decimal genera un error. A continuación se incluyen algunos ejemplos de números de JScript.

NÚMERO	DESCRIPCIÓN	EQUIVALECIA DECIMAL
.0001 0.0001 1e-4 1.0e-4	Cuatro números de coma flotante equivalentes.	0.0001
3.45e2	Un número de coma flotante.	345
42	Un entero.	42
0378	Un entero. Aunque parece un número octal (comienza con un cero), 8 no es un dígito octal válido, por lo que el número se interpretará como un decimal.	378
0377	Un entero octal. Observe que aunque en apariencia se trata del número precedente al anterior, su valor real es muy diferente.	255
0.0001	Un número de coma flotante. Aunque este número comienza con un cero, no es octal, ya que tiene un separador decimal.	0.0001
00.0001	Esto es un error. Los dos ceros precedentes marcan el número como octal, pero este tipo numérico no puede incluir un componente decimal.	N/D (error del compilador)
0Xff	Un entero hexadecimal.	255
0x37CF	Un entero hexadecimal.	14287
0x3e7	Un entero hexadecimal. Tenga en cuenta que la 'e' no se interpreta como exponenciación.	999
0x3.45e2	Esto es un error. Los números hexadecimales no pueden tener partes decimales.	N/D (error del compilador)

Además, JScript contiene números con valores especiales. Son los siguientes:

- **NaN** (no es un número) Se utiliza al realizar una operación matemática en datos inapropiados, como cadenas o con el valor no definido.
- **Infinito positivo**. Se utiliza cuando un número positivo es demasiado grande para representarlo en JScript.
- **Infinito negativo**. Se utiliza cuando un número negativo es demasiado grande para representarlo en JScript.
- **0 positivo o negativo**. JScript diferencia entre cero positivo y negativo.

4.12.2.11.- **TIPO DE DATO BOOLEANO**

Mientras que los tipos de datos de cadena y numérico pueden tener un número prácticamente ilimitado de valores diferentes, el tipo de dato booleano sólo puede tener dos: los literales true y false. Un valor booleano expresa la validez de una condición, indicando si es verdadera o no.

Las comparaciones realizadas en las secuencias de comandos siempre tienen un resultado booleano. Observe la siguiente línea de código de JScript.

```
y = (x == 2000);
```

En este caso, el valor de la variable x se prueba para establecer si es igual al número 2000. Si es así, el resultado de la comparación es el valor booleano true, que se asigna a la variable y. En caso contrario, el resultado de la comparación es el valor booleano false.

Este tipo de valores es especialmente útil en las estructuras de control. En este caso, se combina una comparación que crea directamente un valor booleano con una instrucción que lo usa.

→ Ejemplo:

```
if (x == 2000)
z = z + 1;
else
x = x + 1;
```

La instrucción **if/else** de JScript realiza una acción si un valor booleano es true (en este caso, z = z + 1), y una acción alternativa si el valor booleano es false (x = x + 1).

Puede utilizar cualquier tipo de expresión como expresión comparativa. Toda expresión que dé como resultado una cadena 0, null, undefined o vacía se interpreta como false. Las expresiones que den como resultado cualquier otro valor se interpretan como true.

→ Ejemplo:

```
if (x = y + z) // Es posible que no obtenga los resultados
// previstos -- ¡vea a continuación!
```

Observe que la línea anterior no comprueba si x es igual a y + z, ya que únicamente se utiliza un signo igual (asignación). En vez de eso, el código anterior asigna el valor de y + z a la variable x y, a continuación, comprueba si el resultado de toda la expresión (el valor de x) es cero. Para comprobar si x es igual a y + z, use el código siguiente.

→ Ejemplo:

```
if (x == y + z) // ¡Este código es diferente al anterior!
```

Para obtener más información acerca de las comparaciones, consulte Controlar el flujo del programa.

4.12.2.12.- TIPO DE DATO NULL

El tipo de dato **null** sólo tiene un valor en JScript: **null**. La palabra clave **null** no puede utilizarse como nombre de una función o variable.

Una variable que contiene **null** no contiene "ningún valor" ni "ningún objeto". En otras palabras, no incluye ningún número, cadena, tipo booleano, matriz u objeto válido. Para borrar el contenido de una variable (sin eliminar la variable), asígnele el valor **null**.

Observe que en JScript, null no es lo mismo que 0 (al contrario de lo que ocurre en C y C++). Tenga en cuenta también que el operador **typeof** de JScript interpretará los valores **null** como un tipo **Objeto**, no como un tipo **null**. Este comportamiento, que puede provocar confusiones, permite mantener la compatibilidad con versiones anteriores.

4.12.2.13.- TIPO DE DATO NO DEFINIDO

Se devuelve el valor no definido al utilizar:

- Una propiedad de objeto que no existe
- Una variable que se ha declarado, pero a la que no se ha asignado un valor. Tenga en cuenta que no es posible comparar una variable con el valor undefined para comprobar si dicha variable existe, aunque se puede saber si su tipo es "no definido".

→ Ejemplo: el programador está intentando probar si la variable **x** se ha declarado:

```
// Este método no funcionará
if (x == undefined)
{
// hacer algo
}
// Este método tampoco funcionará - debe comprobar
// la cadena "undefined"
if (typeof(x) == undefined)
{
// hacer algo
}
// Este método funcionará
if (typeof(x) == "undefined")
{
// hacer algo
}
```

Piense en la comparación del valor undefined con el valor null.

```
someObject.prop == null;
```

Esta comparación es **true**,

- Si la propiedad someObject.prop contiene el valor null,
- Si la propiedad someObject.prop no existe.
- Para comprobar si una propiedad de objeto existe puede utilizar el nuevo operador **in**:

```
if ("prop" in unObjeto)
// unObjeto tiene la propiedad 'prop'
```

4.12.2.14.- **OPERADORES JSCRIPT**

JScript tiene un amplio conjunto de operadores, entre los que se incluyen operadores aritméticos, lógicos, de bits y de asignación, así como algunos operadores variados.

Cálculo		Lógico		De bits		Asignación		Varios	
Descripción	**Símbolo**	**Descripción**	**Símbolo**	**Descripción**	**Símbolo**	**Descripción**	**Símbolo**	**Descripción**	**Símbolo**
Negación unaria	-	NOT lógico	!	NOT bits	~	Asignación	=	Delete	delete
Incremento	++	Menor que	<	Desplazamiento de bits a izda.	<<	Asignación compuesta	OP=	Typeof	typeof
Decremento	--	Mayor que	>	Desplazamiento de bits a dcha.	>>			Void	void
Multiplicación	*	Menos o igual que	<=	Desplazamiento de bits a dcha. sin signo	>>>			Instanceof	instanceof
División	/	Mayor o igual que	>=	AND bits	&			New	new
Módulo aritmético	%	Igualdad	==	XOR bits	^			In	in
Suma	+	Desigualdad	!=	OR bits	|				
Resta	-	AND lógico	&&						
		OR lógico	||						
		Condicional	?:						
		Coma	,						
		Igualdad estricta	===						
		Desigualdad estricta	!==						

La diferencia entre == (igualdad) e === (igualdad estricta) es que el operador de igualdad convierte los valores de diferentes tipos antes de comprobar la igualdad. Por ejemplo, la comparación entre la cadena "1" y el número 1 se interpreta como true. Por otro lado, el operador de igualdad estricta no convierte los valores a tipos diferentes, por lo que la cadena "1" no se considera igual al número 1.

Las cadenas primitivas, los números y los tipos booleanos se comparan por valor. Si tienen el mismo valor, se consideran iguales. Los objetos (incluidos los objetos **Array**, **Function**, **String**, **Number**, **Boolean**, **Error**, **Date** y **RegExp**) se comparan por

referencia. Aunque dos variables pertenecientes a dichos tipos tengan el mismo valor, sólo se consideran iguales si hacen referencia al mismo objeto.

→ Ejemplo:

```
// Dos cadenas primitivas con el mismo valor
var cadena1 = "Hola":
var cadena2 = "Hola":
// Dos objetos String, con el mismo valor
var ObjetoString1 = new String(cadena1);
var ObjetoString2 = new String(cadena2);
// El resultado en este caso será true
if (cadena1 == cadena2)
{
// hacer algo (se ejecutará)
}
// El resultado en este caso será false
if (ObjetoString1 == ObjetoString2)
{
// hacer algo (no se ejecutará)
}
// Para comparar el valor de los objetos String,
// use los métodos toString() o valueOf()
if (ObjetoString1.valueOf() == ObjetoString2)
{
// hacer algo (se ejecutará)
}
```

4.12.2.15.- CONTROLAR EL FLUJO DE PROGRAMA

Normalmente, las instrucciones de una secuencia de comandos de JScript se ejecutan una detrás de otra, en el orden en que se han escrito. Esto se denomina ejecución secuencial y es la dirección predeterminada del flujo del programa.

Una de las alternativas a este tipo de ejecución transfiere el flujo del programa a otra parte de la secuencia de comandos. Es decir, en lugar de ejecutar la instrucción siguiente de la secuencia, se ejecuta otra.

Para que la secuencia de comandos sea útil, esta transferencia de control debe realizarse de una forma lógica. La transferencia del control del programa se basa en una decisión, cuyo resultado es una instrucción del valor de verdad (que devuelve un tipo booleano **true** o **false**). Se crea una expresión y a continuación se prueba si su resultado es verdadero. Para ello hay dos clases principales de estructuras de programa.

La primera es la estructura de selección. Se utiliza para especificar rutas alternativas para el flujo del programa y crear de este modo un entronque en el programa. En JScript hay disponibles cuatro tipos de estructuras de selección.

- La estructura de selección única (**if**),
- La estructura de selección doble (**if/else**),
- El operador ternario en línea **?:**
- La estructura de selección múltiple (**switch**).

El segundo tipo de estructura de control de programas es la estructura de repetición. Utilícela para especificar que una acción se repita mientras alguna condición siga siendo verdadera. Cuando las condiciones de la instrucción de control se hayan cumplido (normalmente después de un número específico de repeticiones), el control se transfiere a la siguiente instrucción externa a la estructura de repetición. En JScript hay disponibles cuatro tipos de estructuras de repetición.

- La expresión se prueba en la parte superior del bucle (**while**),
- La expresión se prueba en la parte inferior del bucle (**do/while**),
- La estructura opera en cada una de las propiedades de un objeto (**for/in**).
- Repetición controlada mediante contador (**for**).

Es posible crear secuencias de comandos muy complejas mediante la anidación y el apilamiento de las estructuras de control de selección y repetición.

El tercer tipo de flujo de programa estructurado, el control de excepciones, no se trata en este documento.

4.12.2.16.- USAR INSTRUCCIONES CONDICIONALES

JScript admite las instrucciones condicionales if e if...else. En las instrucciones if se prueba una condición y, si satisface la prueba, se ejecuta el código de JScript apropiado. En la instrucción if...else se ejecuta un código distinto si la condición no cumple la prueba. La forma más sencilla de una instrucción if se puede escribir en una sola línea, pero se usan mucho más las instrucciones if e if...else de múltiples líneas.

Los siguientes ejemplos demuestran distintas sintaxis que puede usar con las instrucciones if e if...else.

→ Ejemplo: Si el elemento encerrado entre paréntesis da como resultado true, se ejecuta la instrucción o el bloque de instrucciones que sigue a la instrucción if.

```
// La función estrellar() se define en cualquier otra parte del
// código.
// Prueba de tipo Boolean para saber si barcoNuevo es true.
if (barcoNuevo)
estrellar(botellaCava,proa);
// En este ejemplo, la prueba no se cumple a menos que ambas
// condiciones sean verdaderas.
if (cascara.color == "amarillo oscuro"
&&
cascara.textura == "arrugas grandes y pequeñas")
{
laRespuesta = ("¿Es un melón?");
}
// En este ejemplo, la prueba se cumple si cualquiera de las
// condiciones es verdadera
var laReaccion = "";
if ((dayOfWeek == "Sábado") || (dayOfWeek == "Domingo"))
{
```

```
laReaccion = ("¡Me voy de vacaciones a la playa!");
}
else
{
laRespuesta = ("¡Hi ho, hi ho, a casa a descansar!");}
```

4.12.2.17.- **OPERADOR CONDICIONAL**

JScript también admite una forma condicional implícita. Utiliza un signo de interrogación después de la condición que va a probar, también especifica dos alternativas, una para usarla si la condición se cumple y otra para usarla si no se cumple. Las alternativas deben separarse con dos puntos.

→ Ejemplo:

```
var horas = "";
// Código que especifica que la variable horas tiene el contenido
// de laHora o laHora - 12.
horas += (laHora >= 12) ? " p.m." : " a.m.";
```

Si tiene que probar varias condiciones juntas y sabe que alguna tiene más posibilidades de cumplirse o de no cumplirse que las demás, puede utilizar la característica 'evaluación de cortocircuito' para acelerar la velocidad de ejecución de la secuencia de comandos. Cuando JScript evalúa una expresión lógica, sólo analiza el número de subexpresiones necesario para obtener un resultado.

Por ejemplo, si tiene una expresión 'AND' como ((x == 123) && (y == 42)), JScript comprueba primero si x es igual a 123. Si esta condición no se cumple, la expresión no puede ser verdadera, aun cuando y sea igual a 42. En consecuencia, la comprobación de y no se realiza y JScript devuelve el valor false.

De forma similar, si de varias condiciones sólo una debe ser verdadera (usando operadores ||), la prueba se detiene tan pronto como una condición cumple este requisito. Esto es particularmente efectivo si las condiciones a probar implican la ejecución de llamadas a funciones u otras expresiones complejas. Teniendo esto en cuenta, cuando se escriban expresiones OR, situar en primer lugar las condiciones con mayor probabilidad de ser verdaderas. Cuando escriba expresiones AND, situar en primer lugar las condiciones con mayor probabilidad de ser falsas.

Un ejemplo de las ventajas de diseñar la secuencia de comandos de esta manera es que, en el siguiente ejemplo, ejecutarSegundo() no se ejecuta a menos que ejecutarPrimero() devuelva 0 o false.

→ Ejemplo:

```
if ((ejecutarPrimero() == 0) || (ejecutarSegundo() == 0))
{
// algún código
}
```

4.12.2.18.- **USAR BUCLES**

Hay diversas formas de ejecutar una instrucción o un bloque de instrucciones varias veces. En general, la ejecución repetitiva se llama ejecución en bucle o iteración. Una iteración es simplemente una ejecución única de un bucle.

Normalmente se controla mediante la prueba de una variable cuyo valor cambia cada vez que se ejecuta el bucle. JScript admite cuatro tipos de bucles: los bucles for, for...in, while y do...while.

Bucles for

La instrucción **for** especifica una variable de contador, una condición de prueba y una acción que actualiza el contador. La condición se comprueba antes de cada iteración del bucle. Si la comprobación es correcta, se ejecuta el código interior del bucle. En caso contrario, el código interior del bucle no se ejecuta y el programa continúa por la primera línea de código inmediatamente posterior al bucle. Después de ejecutar el bucle, la variable de contador se actualiza antes de comenzar la siguiente iteración.

Si nunca se cumple la condición del bucle, éste nunca se ejecuta. Si la condición del bucle se cumple siempre, el bucle se convierte en un proceso infinito.

Aunque es posible que lo primero sea necesario en algunos casos, lo segundo raramente lo es, por lo que debe tener cuidado al escribir las condiciones de los bucles.

→ Ejemplo:

```
/*
La expresión de actualización ("contadori++" en los siguientes
ejemplos) se ejecuta al final del bucle, después de que el bloque de
instrucciones que forman el cuerpo del bucle se ejecuta y antes de
comprobar la condición.
*/
var limite = 10; // Establece un límite de 10 en el bucle.
var suma = new Array(limite); // Crea una matriz llamada suma
// con 10 miembros, desde 0 hasta 9.
var laSuma = 0;
suma[0] = 0;
for(var contadori = 0; contadori < limite; contadori++)
{
// En este caso, cuenta desde 0 hasta 9
laSuma += contadori;
suma[contadori] = laSuma;
}
var nuevaSuma = 0;
for(var contadori= 0; contadori > limite; contadori++)
{ // Esto nunca se ejecuta, ya que contadori no es mayor que
// limite
nuevaSuma += contadori;
}
var suma = 0;
for(var contadori = 0; contadori >= 0; contadori++)
{ // Esto es un bucle infinito
suma += contadori;
```

```
}
```

Bucles for … in

JScript proporciona un tipo de bucle especial para examinar una a una todas las propiedades definidas por el usuario de un objeto o todos los elementos de una matriz. El contador del bucle en un bucle for...in es una cadena, no un número.

Contiene el nombre de la propiedad actual o el índice del elemento de matriz actual.

El siguiente ejemplo de código debe ejecutarse desde dentro de Internet Explorer, ya que utiliza el método alert, que no forma parte de JScript.

→ Ejemplo:

```
// Crear un objeto con algunas propiedades
var miObjeto = new Object();
miObjeto.name = "Jaime";
miObjeto.age = "22";
miObjeto.phone = "555 1234";
// Enumere (itere) todas las propiedades del objeto
for (prop in miObjeto)
{ // Se muestra "El 'nombre' de propiedad es Jaime", etc.
window.alert("La propiedad '"
+
prop
+
"' es "
+
miObjeto[prop]);
}
```

Aunque los bucles for...in son similares a los bucles For Each...Next de VBScript, no funcionan de la misma manera.

El bucle for...in de JScript itera sobre propiedades de objetos de JScript. El bucle For Each...Next de VBScript itera sobre elementos de una colección. Para iterar sobre colecciones en JScript, es necesario utilizar el objeto Enumerator. Aunque algunos objetos, como los de Internet Explorer, admiten los bucles For Each...Next de VBScript y for...in de JScript, la mayoría de los objetos no los admiten.

Bucles while

El bucle while es muy parecido a un bucle for. La diferencia es que un bucle while no tiene integrada una variable de contador o una expresión de actualización. Si desea controlar la ejecución repetitiva de una instrucción o bloque de instrucciones, pero necesita una regla más compleja que "ejecutar este código n veces", use un bucle while. En el ejemplo siguiente se emplea el modelo de objetos de Internet Explorer y un bucle while para hacer al usuario una pregunta sencilla.

→ Ejemplo:

```
var x = 0;
while ((x != 42) && (x != null))
{
x = window.prompt("¿Cuál es mi número favorito?", x);
}
if (x == null)
ShowMessage("¡Te has dado por vencido!");
else
ShowMessage("¡Exacto, esa es la respuesta definitiva!");
```

Debido a que los bucles while no tienen variables de contador explícitas integradas, son más vulnerables a crear bucles infinitos que otros tipos de bucles.

Además, debido a que no es fácil descubrir dónde y cuándo se actualiza la condición del bucle, hay muchas posibilidades de escribir accidentalmente un bucle while en el que la condición nunca se actualice. Por este motivo, debe tener precaución al diseñar bucles while.

Como se especificó anteriormente, en JScript también hay un bucle do...while similar al bucle while, con la excepción de que se garantiza que se ejecuta siempre al menos una vez, ya que la condición se comprueba al final del bucle, en lugar de al principio. Por ejemplo, el bucle anterior puede volverse a escribir de la siguiente manera:

```
var x = 0;

do
{
x = window.prompt("¿Cuál es mi número favorito?", x);
}
while ((x != 42) && (x != null));
if (x == null)
window.alert("¡Te has dado por vencido!");
else
window.alert("¡Exacto, esa es la respuesta definitiva!");
```

Instrucciones break y continue

En Microsoft JScript, la instrucción **break** se puede usar para detener la ejecución de un bucle, si se cumple alguna condición. La instrucción break también se utiliza para salir de un bloque **switch**. La instrucción **continue** se puede usar para pasar inmediatamente a la siguiente iteración, saltando el resto del bloque de código y actualizando al mismo tiempo la variable de contador si el bucle es un bucle **for** o **for...in**.

→ Ejemplo:

```
var x = 0;
do
{
x = window.prompt("¿Cuál es mi número favorito?", x);
// ¿Ha cancelado la acción el usuario? Si es así, sale del
// bucle
if (x == null)
break;
// ¿Han escrito un número?
```

```
// Si es así, no es necesario pedirles que escriban un número
if (Number(x) == x)
continue;
// Pide al usuario que sólo escriba números
window.alert("Escriba únicamente números");
}
while (x != 42)
if (x == null)
window.alert("¡Te has dado por vencido!");
else
window.alert("¡Exacto, esa es la respuesta definitiva!");
```

4.12.2.19.- FUNCIONES DE JSCRIPT

Las funciones de Microsoft JScript realizan acciones y también pueden devolver valores. A veces son los resultados de cálculos o comparaciones. Las funciones también se denominan "métodos globales".

Combinan varias operaciones en un único nombre. Esto permite simplificar el código. Para escribir un conjunto de instrucciones, asignarle un nombre y ejecutarlo, tan sólo es necesario llamarlo y pasarle la información que necesite.

Para pasar información a una función hay que encerrar la información entre paréntesis después del nombre de la función. Los elementos de información que se pasan a una función se llaman argumentos o parámetros. Algunas funciones no aceptan ningún argumento, mientras que otras utilizan uno o varios. En algunas funciones el número de argumentos depende de cómo se utilice la función.

JScript admite dos tipos de funciones: las que están integradas en el lenguaje y las creadas por el propio usuario.

4.12.2.19.1.- FUNCIONES ESPECIALES INTEGRADAS

El lenguaje JScript incluye varias funciones integradas. Algunas permiten controlar expresiones y caracteres especiales, mientras que otras convierten cadenas en valores numéricos. Una función integrada que resulta útil es **eval()**. Esta función evalúa cualquier código de JScript válido que se presente en forma de cadena. La función eval() recibe un argumento que es el código que se va a evaluar.

→ Ejemplo:

```
var unaExpresion = "6 * 9 % 7";
var total = eval(unaExpresion); // Asigna el valor 5 a la
// variable total.
var otraExpresionMas = "6 * (9 % 7)";
total = eval(otraExpresionMas) // Asigna el valor 12 a la
// variable total.
// Asigna una cadena a la totalidad (tenga en cuenta las comillas
// anidadas)
```

```
var totalidad = eval("'...rodeado por hectáreas de olivares.'");
```

 Consulte la referencia del lenguaje para obtener más información acerca de éstas y otras funciones integradas.

 La función **Format** de JScript ha sufrido modificaciones respecto a versiones anteriores a Designer IV. Por ello, y para que las visualizaciones importadas de otras versiones funcionen correctamente, es posible solucionar este punto de una manera sencilla creando una función propia que simule a la función **Format**.

4.12.2.19.2.- CREAR FUNCIONES PROPIAS

Es posible crear funciones propias cuando sean necesarias. La definición de una función está formada por una instrucción function y un bloque de instrucciones de JScript.

La función *compruebaTriangulo* del siguiente ejemplo usa como argumentos las longitudes de los lados de un triángulo. Con ellas calcula si los tres números cumplen el teorema de Pitágoras (en un triángulo rectángulo, la suma del cuadrado de los catetos es igual al cuadrado de la hipotenusa) para comprobar si el triángulo es rectángulo. La función *compruebaTriangulo* llama a una función de dos para realizar el cálculo real.

Observe el uso de un número muy pequeño ("epsilon") como variable de prueba en la versión en coma flotante de la comprobación. Debido a las incertidumbres y a los errores de redondeo en los cálculos de coma flotante, no resulta práctico probar directamente si los tres números cumplen el teorema de Pitágoras a menos que se sepa que los valores en cuestión son números enteros. Como es más exacto realizar una prueba directa, el código de este ejemplo determina si este método es apropiado y, si lo es, lo utiliza.

→ Ejemplo:

```
var epsilon = 0.00000000001;// Algún número muy pequeño contra
// el que se realiza la comprobación.
// La función de comprobación para los enteros
function compruebaEntero(a, b, c)
{
// La propia función
if ( (a*a) == ((b*b) + (c*c)) )
return true;
return false;
} // Fin de la función de comprobación de enteros
// La función de comprobación para números de coma flotante
function compruebaFlotante(a, b, c)
{
// Crea el número de comprobación
var delta = ((a*a) - ((b*b) + (c*c)))
// La comprobación requiere el valor absoluto
delta = Math.abs(delta);
// Si la diferencia es menor que epsilon, es muy aproximada
```

```
if (delta < epsilon)
return true;
return false;
} // Fin de la función de comprobación de coma flotante
// La comprobación del teorema
function compruebaTriangulo(a, b, c)
{
// Crea una variable temporal para intercambiar valores
var d = 0;
// Primero pasa el lado mayor a la posición "a"
// Intercambia a y b si es necesario
if (b > a)
{
d = a;
a = b;
b = d;
}
// Intercambia a y c si es necesario
if (c > a)
{
d = a;
a = c;
c = d;
}
// Comprobar los tres valores. ¿Son enteros?
if (((a % 1) == 0) && ((b % 1) == 0) && ((c % 1) == 0))
{
// Si lo son se utiliza la comprobación precisa
return compruebaEntero(a, b, c);
}
else
{
// Si no lo son, se obtienen los valores más próximos
return compruebaFlotante(a, b, c);
}
} // Fin de la función de comprobación del teorema
// Las tres instrucciones siguientes asignan valores de prueba
// para realizar una comprobación
var ladoA = 5;
var ladoB = 5;
var ladoC = Math.sqrt(50.001);
// Llama a la función. Después de la llamada, 'resultado'
// contiene el resultado
var resultado = compruebaTriangulo(ladoA, ladoB, ladoC);
```

4.12.2.20.- **OBJETOS DE JSCRIPT**

Los objetos de JScript son colecciones de propiedades y métodos.

Un método es una función que es miembro de un objeto.

Una propiedad es un valor o un conjunto de valores (en forma de matriz o de objeto) que son miembros de un objeto.

JScript admite cuatro tipos de objetos:

- Objetos intrínsecos.
- Objetos creados por el usuario.
- Objetos de host, proporcionados por el host (como window y document en Internet Explorer).
- Objetos de Active X (componentes externos).

4.12.2.20.1.- OBJETOS COMO MATRICES

En JScript, los objetos y las matrices se tratan de forma casi idéntica. A ambos se les puede asignar propiedades arbitrarias; en realidad, las matrices tan sólo son un tipo especial de objeto. La diferencia entre las matrices y los objetos es que las primeras tienen una propiedad length que es "mágica" y que otros objetos no poseen. Esto significa que si se asigna un valor a un elemento de una matriz que es mayor que el resto de los elementos (por ejemplo, miMatriz[100] = "hola") la propiedad length se actualiza automáticamente a 101 (la nueva longitud). De forma semejante, si modifica la propiedad length de una matriz, se eliminan todos los elementos que ya no formen parte de la misma.

Todos los objetos de JScript admiten propiedades "que se expanden" (expando) o propiedades que pueden agregarse o quitarse dinámicamente en tiempo de ejecución. A dichas propiedades se les puede asignar cualquier nombre, incluidos números. Si el nombre de la propiedad es un identificador simple, puede escribirse después del nombre del objeto con un punto, como en:

→ Ejemplo:

```
var miObj = new Object();
// Agregar dos propiedades expando, 'name' y 'age'
miObj.name = "Alfredo";
miObj.age = "22";
```

Si el nombre de la propiedad no es un identificador simple o no se conoce en el momento de escribir la secuencia de comandos, puede incluir una expresión arbitraria dentro de corchetes para indizar la propiedad. Los nombres de todas las propiedades expando de JScript se convierten en cadenas antes de agregarlos al objeto.

→ Ejemplo:

```
var miObj = new Object();
// Agregar dos propiedades expando que no puedan escribirse en la
// sintaxis de object.property
// La primera contiene caracteres no válidos (espacios), de modo
// que debe escribirse entre corchetes
miObj["identificador no válido"] = "Este es el valor de la
propiedad";
// El segundo nombre que se expande es un número, por lo que
// también debe colocarse entre corchetes
miObj[100] = "100";
```

Normalmente, a los elementos de la matriz se les asignan índices numéricos que empiezan a partir de cero. Estos son los elementos que interactúan con la propiedad length. Puesto

que todas las matrices son objetos, también admiten propiedades expando. Observe, sin embargo, que este tipo de propiedades no pueden interactuar con la propiedad length.

→ Ejemplo:

```
// Una matriz con tres elementos
var miMatriz = new Array(3);
// Agregar algunos datos
miMatriz[0] = "Hola";
miMatriz[1] = 42;
miMatriz[2] = new Date(2000, 1, 1);
// Muestra 3, la longitud de la matriz
window.alert(miMatriz.length);
// Agregar algunas propiedades expando
miMatriz.expando = "¡JScript!";
miMatriz["otra propiedad Expando"] = "Windows";
// Continúa mostrando 3, ya que las dos propiedades expando
// no afectan a la longitud
window.alert(miMatriz.length);
```

Aunque JScript no admite directamente matrices multidimensionales, puede almacenar cualquier tipo de datos dentro de los elementos de la matriz, incluidas otras matrices. Si almacena matrices dentro de los elementos de otra matriz puede conseguir el comportamiento de una matriz multidimensional.

→ Ejemplo: Crear una tabla de multiplicación para los números hasta el 5:

```
// Cambiar este número para una tabla mayor
var iNumMax = 5;
// Contadores del bucle
var i, j;
// Nueva matriz. Lo convierte en iNumMax + 1, ya que las matrices
// comienzan a contar desde cero, no desde 1
var TablaMultiplicacion = new Array(iNumMax + 1);
// Ejecutar el bucle para cada número mayor (cada fila de la
// tabla)
for (i = 1; i <= iNumMax; i++)
{
// Crear las columnas de la tabla
TablaMultiplicacion[i] = new Array(iNumMax + 1);
// Rellenar la fila con los resultados de la multiplicación
for (j = 1; j <= iNumMax; j++)
{
TablaMultiplicación[i][j] = i * j;
}
}
window.alert(TablaMultiplicación[3][4]); // Muestra 12
window.alert(TablaMultiplicación[5][2]); // Muestra 10
window.alert(TablaMultiplicación[1][4]); // Muestra 4
```

Crear sus propios objetos

Para crear instancias de sus propios objetos, primero debe definir una función constructora para ellos. Una función constructora crea un objeto y le asigna propiedades y, si es necesario, métodos. Por ejemplo, el ejemplo siguiente define una función constructora para objetos pasta. Observe el uso de la palabra clave this, que hace referencia al objeto actual.

→ Ejemplo:

```
// pasta es un constructor que utiliza cuatro parámetros
función pasta (cereal, grosor, forma, conHuevo)
{
// ¿De qué tipo de cereal está hecha?
this.cereal = cereal;
// ¿Qué grosor tiene? (número)
this.grosor = grosor;
// ¿Cuál es su forma? (cadena)
this.forma = forma;
// ¿Incluye yema de huevo como aglutinante? (booleano)
this.conHuevo = conHuevo;
}
```

Una vez que se ha definido un constructor de objetos, se crean sus instancias mediante el operador new.

→ Ejemplo:

```
var espagueti = new pasta("trigo", 0,2, "círcular", true);
var linguini = new pasta("trigo", 0,3, "oval", true);
```

Se pueden agregar propiedades a una instancia de un objeto para cambiarla, pero estas propiedades no entran a formar parte de la definición del resto de los objetos creados con el mismo constructor ni aparecen en otras instancias a menos que se agreguen específicamente. Si se desea que las propiedades adicionales se muestren en todas las instancias del objeto, deben de agregarse a la función constructora o al objeto del prototipo del constructor (los prototipos se describen en la documentación de características avanzadas).

→ Ejemplo:

```
// Propiedades adicionales para espagueti
espagueti.color = "amarillo paja pálido";
espagueti.cocciónseco = 7;
espagueti.cocciónfresco = 0,5;
var chowFun = new pasta("arroz", 3, "plana", false);
// Ni el objeto chowFun ni ninguno de los objetos
// pasta existentes tienen las tres propiedades nuevas que se han
// agregado al objeto espagueti
// Al agregar la propiedad 'grupoAlimento' al objeto prototipo
// pasta, ésta queda disponible para todas las instancias de los
```

```
// objetos pasta incluidas aquellas que ya se han creado
pasta.prototipo.grupoAlimento = "carbohidratos"
// ahora todos los objetos espagueti.grupoAlimento,
// grupoAlimento, etc. contienen el valor "carbohidratos"
```

4.12.2.20.2.- INCLUIR MÉTODOS EN LA DEFINICIÓN

Es posible incluir métodos (funciones) en la definición de un objeto. Una manera de hacerlo consiste en agregar una propiedad a la función constructora que haga referencia a una función definida en cualquier otra parte. Por ejemplo, el siguiente ejemplo expande la función constructora de pasta definida anteriormente de modo que incluya un método toString, al que se llama si se muestra el valor del objeto.

```
// pasta es un constructor que utiliza cuatro parámetros
// La primera parte es la misma que en el ejemplo anterior
función pasta (cereal, grosor, forma, conHuevo)
{
// ¿De qué tipo de cereal está hecha?
this.cereal = cereal;
// ¿Qué grosor tiene? (número)
this.grosor = grosor;
// ¿Cuál es su forma? (cadena)
this.forma = forma;
// ¿Incluye yema de huevo como aglutinante? (booleano)
this.conHuevo = conHuevo;
// Aquí agregaremos el método toString (que se define a
// continuación).
// Tenga en cuenta que no ponemos los paréntesis después del
// nombre de la función; no se trata de una llamada a la
// función, sino de una referencia a la propia función
this.toString = pastaToString;
}
// La función real para mostrar el contenido de un objeto pasta
function pastaToString()
{
// devuelve las propiedades del objeto
return "Cereal: "
+
this.cereal
+
"\n"
+
"Grosor: "
+
this.grosor
+
"\n"
+
"Forma: "
+
this.forma
+
"\n"
+
```

```
"¿Huevo?: "
+
Boolean(this.conHuevo);
}
var espagueti = new pasta("trigo", 0,2, "circular", true);
// Esto llama a toString() y muestra las propiedades
// del objeto espagueti (se requiere Internet Explorer)
window.alert(espagueti);
```

4.12.2.20.3.- OBJETOS INTRÍNSECOS

La implementación de JScript utilizada proporciona once objetos intrínsecos (o "integrados"). Se trata de los objetos **Array**, **Boolean**, **Date**, **Function**, **Global**, **Math**, **Number**, **Object**, **RegExp**, **Error** y **String**.

Cada uno de ellos tiene métodos y propiedades asociados que se describen con detalle en la referencia de lenguaje.

Objeto Array

Los subíndices de una matriz pueden considerarse como propiedades de un objeto y se hace referencia a ellos por su índice numérico. Las propiedades con nombre agregadas a una matriz no pueden indizarse por número ya que son independientes de los elementos de la matriz.

→ Ejemplo:

```
var losMeses = new Array(12);
losMeses[0] = "Ene";
losMeses[1] = "Feb";
losMeses[2] = "Mar";
losMeses[3] = "Abr";
losMeses[4] = "May";
losMeses[5] = "Jun";
losMeses[6] = "Jul";
losMeses[7] = "Ago";
losMeses[8] = "Sep";
losMeses[9] = "Oct";
losMeses[10] = "Nov";
losMeses[11] = "Dic";
```

Cuando se crea una matriz mediante la palabra clave **Array**, JScript incluye una propiedad **length**, que registra el número de entradas. Si no especifica ningún número, la longitud se establece en 0 y la matriz no tiene entradas. Si especifica un número, la longitud se establece en dicho número. Si especifica varios parámetros, se utilizan como entradas de la matriz. Además, el número de parámetros se asigna a la propiedad **length.**

→ Ejemplo:

```
var losMeses = new Array(("Ene",
"Feb",
"Mar",
"Abr",
"May",
"Jun",
"Jul",
"Ago",
"Sep",
"Oct",
"Nov",
"Dic");
```

JScript cambia automáticamente el valor de **length** cuando se agregan elementos a una matriz creada con la palabra clave **Array**. Los índices de matriz de JScript siempre empiezan con 0, no con 1, de modo que la propiedad **length** siempre es una unidad mayor que el índice de mayor tamaño de la matriz.

Objeto string

En JScript es posible tratar a las cadenas y a los números como si fueran objetos. El objeto String dispone de varios métodos integrados que pueden utilizarse con las cadenas. Uno de ellos es el método substring, que devuelve parte de la cadena. Utiliza dos números como argumentos.

→ Ejemplo:

```
aString = "0123456789";
var unSegmento = aString.substring(4, 7); // Establece
// unSegmento a "456"
var otroSegmento = aString.substring(7, 4); // Establece
// otroSegmento
// a "456"
// Si se utiliza el ejemplo de creación de matriz anterior
primeraLetra = losMeses[5].substring(0,1); // Establece la
// variable
// primeraLetra a "E"
```

Otra propiedad del objeto String es length. Esta propiedad contiene el número de caracteres de la cadena (0 para una cadena vacía). Se trata de un valor numérico que puede utilizarse directamente en los cálculos.

```
var Duracion = "Hola a todos".length // Establece la variable
// Duracion a 11
```

Objeto Math

El objeto Math tiene varios métodos y propiedades predefinidos. Las propiedades son números específicos. Uno de ellos es el valor de pi (aproximadamente 3,14159...). Esta es la propiedad Math.PI.

→ Ejemplo:

```
// Se declara una variable radio y se le asigna un valor numérico
var AreaCírculo = Math.PI * radio * radio; // Observe las
// mayúsculas de
// Math y PI
```

Uno de los métodos integrados del objeto Math es el método de exponenciación, o pow, que eleva un número a la potencia especificada. En el siguiente ejemplo se emplean los métodos pi y exponenciación.

→ Ejemplo:

```
// Esta fórmula calcula el volumen de una esfera con radio dado
volumen = (4/3)*(Math.PI*Math.pow(radio,3));
```

Objeto Date

El objeto Date puede utilizarse para representar fechas y horas arbitrarias, para obtener la fecha actual del sistema y para calcular las diferencias entre fechas.

Tiene varias propiedades y métodos, todos ellos predefinidos. En general, el objeto Date proporciona el día de la semana, el mes, el día y el año, y la hora en horas, minutos y segundos. Esta información se basa en el número de milisegundos transcurridos desde el 1 de enero de 1970, 00:00:00.000 GMT, que es la hora media de Greenwich (el término más utilizado es el UTC (Universal Coordinated Time), u Horario Universal Coordinado, que hace referencia a las señales emitidas por World time Standard (o estándar de horario mundial). JScript puede tratar fechas comprendidas entre los años 250.000 A.C. y 255.000 D.C., aproximadamente. Para crear un nuevo objeto Date, use el operador new.

→ Ejemplo: Cálculo el número de días transcurridos y restantes del año actual.

```
/*
Este ejemplo utiliza la matriz de nombres de mes definida
anteriormente.
La primera instrucción asigna la fecha actual, en formato "Día
Mes Fecha 00:00:00 Año", a la variable HoyEs.
*/
var hoyEs = new Date();
var Hoy = new Date(); // Captura la fecha actual
// Extrae el año, el mes y el día
var esteAño = Hoy.getFullYear();
var esteMes = losMeses[Hoy.getMonth()];
var esteDia = esteMes + " " + Hoy.getDate() + ", " + esteAño;
```

Objeto Number

Además de las propiedades numéricas especiales disponibles en el objeto Math, Microsoft JScript ofrece varias propiedades adicionales mediante el objeto Number.

PROPIEDAD	DESCRIPCIÓN
MAX_VALUE	El mayor número posible, aproximadamente 1.79E+308; puede ser positivo o negativo. (El valor varía ligeramente entre sistemas).
MIN_VALUE	El menor número posible, aproximadamente 2.22E-308; puede ser positivo o negativo. (El valor varía ligeramente entre sistemas).
NaN	Valor especial no numérico, "*not a number*".
POSITIVE_INFINITY	Cualquier valor positivo mayor que el mayor número positivo (Number.MAX_VALUE) se convierte automáticamente a este valor; se representa como infinito.
NEGATIVE_INFINITY	Cualquier valor negativo mayor que el mayor número negativo (Number.MAX_VALUE) se convierte automáticamente a este valor; se representa como -infinito.

Number.NaN es una propiedad especial que se define como "no es un número". La división por cero, por ejemplo, devuelve **NaN**. Un intento de analizar una cadena que no puede analizarse como un número también devuelve **Number.NaN. NaN** se compara como diferente a cualquier número y a sí misma. Para comprobar un resultado **NaN**, no compare con **Number.NaN**; en lugar de ello, use la función **isNaN().**

4.12.2.21.- PALABRAS RESERVADAS EN JSCRIPT

JScript tiene varias palabras reservadas que no pueden utilizarse como identificadores. Las palabras reservadas tienen un significado específico para el lenguaje de JScript, ya que forman parte de su sintaxis. El uso de una palabra reservada provocará un error de compilación al cargar la secuencia de comandos.

JScript también incluye una lista de futuras palabras reservadas. En la actualidad dichas palabras no forman parte del lenguaje de JScript, aunque se reservan para uso futuro.

PALABRAS RESERVADAS				
break	delete	function	return	typeof
base	do	if	switch	var
batch	else	in	this	void
bontinue	false	instanceof	throw	while
debugger	finally	new	true	with
default	for	null	try	

FUTURAS PALABRAS RESERVADAS				
abstract	double	goto	native	static
boolean	enum	implements	package	super
byte	export	import	private	synchronized
char	extends	int	protected	throws
class	final	interface	public	transient
const	float	long	short	volatile

Al elegir identificadores también debe evitar cualquier palabra que ya sea un nombre de un objeto o una función intrínseca de JScript, como String o parseInt.

4.12.2.22.- CARACTERÍSTICAS AVANZADAS DE JAVASCRIPT

4.12.2.22.1.- CREACIÓN AVANZADA DE OBJETOS : USAR CONSTRUCTORES PARA CREAR OBJETOS

Un constructor es una función a la que se llama para crear instancias e inicializar un tipo determinado de objetos. Se llama a un constructor con la palabra clave **new**.

→ Ejemplo:

```
var miObjeto = new Object(); // Crea un objeto
// genérico sin propiedades
var miCumpleaños = new Date(1961, 5, 10); // Crea un objeto Date
var micoche = new Car(); // Crea un objeto definido por el
// usuario e inicializa sus propiedades
```

Se pasa al constructor una referencia a un objeto vacío recién creado, como el valor de la palabra clave especial **this**. A continuación, el constructor es responsable de llevar a cabo una inicialización apropiada del nuevo objeto (crear propiedades y asignarles valores iniciales). Cuando termina, devuelve una referencia al objeto que ha construido.

4.12.2.22.2.- ESCRIBIR CONSTRUCTORES

Puede crear objetos e inicializarlos mediante el operador **new** junto con funciones de constructor predefinidas, como **Object()**, **Date()** y **Function()**. Una de las características más eficaces de la programación orientada a objetos es la capacidad de definir funciones de constructor personalizadas para crear objetos personalizados y usarlos en secuencias de comandos. La creación de constructores personalizados le permite crear objetos con propiedades ya definidas.

→ Ejemplo:

```
function Circulo (puntox, puntoy, radio)
{
this.x = puntox; // El componente x del centro del círculo
this.y = puntoy; // El componente y del centro del círculo
this.r = radio; // El radio del círculo
}
```

Al invocar al constructor Circulo, se suministran valores para el punto central y el radio del círculo (estos elementos son lo único que se necesita para definir completamente un objeto círculo). Se consigue un objeto Círculo con tres propiedades. Las instancias del objeto Círculo se crean de la siguiente manera.

```
var unCirculo = new Circulo(5, 11, 99);
```

4.12.2.22.3.- USAR PROTOTIPOS PARA CREAR OBJETOS

Cuando se escribe un constructor, es posible utilizar propiedades del objeto prototipo (prototype, que es una propiedad de todos los constructores) para crear propiedades heredadas y métodos compartidos. Las propiedades y métodos del prototipo se copian por referencia en cada objeto de una clase, por lo que todos tienen el mismo valor. Es posible cambiar el valor de una propiedad prototipo en un objeto y el nuevo valor sustituye al predeterminado, pero sólo en esa instancia. El resto de los objetos miembros de esa clase no se ven afectados por el cambio. A continuación se incluye un ejemplo en el que se hace uso del constructor personalizado, Circulo (observe el uso de la palabra clave **this**).

→ Ejemplo:

```
Circulo.prototype.pi = Math.PI;
function AreadeCirculo ()
{
return this.pi * this.r * this.r; // La fórmula para hallar el
// área de un círculo es pi·r2
}
Circulo.prototype.area = AreadeCirculo; // La función que calcula
// el área de un círculo
// es ahora un método del
// objeto prototipo
// Círculo
var a = UnCirculo.area(); // De esta manera se invocaría la función
// area en un objeto Círculo
```

Siguiendo este principio, puede definir propiedades adicionales para funciones de constructor predefinidas (todas las cuales tienen objetos prototype). Por ejemplo, si desea poder quitar espacios iniciales y finales de las cadenas (de forma similar a la función **Trim** de VBScript), puede crear un método propio en el objeto prototype de **String** y todas las cadenas de la secuencia de comandos heredarán automáticamente el método.

→ Ejemplo:

```
// Agregar una función denominada trim como método del objeto
// prototype del constructor String
String.prototype.trim = function()
{
// Use una expresión regular para reemplazar los espacios
// iniciales y finales con una cadena vacía
return this.replace(/(^\s*)|(\s*$)/g, "");
}
// Una cadena con espacios
var s = " espacios iniciales y finales ";
// Muestra " espacios iniciales y finales (35)"
window.alert(s + " (" + s.length + ")");
// Quita los espacios iniciales y finales
s = s.trim();
// Muestra "espacios iniciales y finales (27)"
window.alert(s + " (" + s.length + ")");
```

4.12.2.22.4.- RECURSIVIDAD

La recursividad es una técnica de programación importante. Se utiliza para realizar una llamada a una función desde la misma función. Como ejemplo útil se puede presentar el cálculo de números factoriales. El factorial de 0 es, por definición, 1. Los factoriales de números mayores se calculan mediante la multiplicación de 1 * 2 *..., incrementando el número de 1 en 1 hasta llegar al número para el que se está calculando el factorial.

El siguiente párrafo muestra una función, expresada con palabras, que calcula un factorial.

"Si el número es menor que cero, se rechaza. Si no es un entero, se redondea al siguiente entero. Si el número es cero, su factorial es uno. Si el número es mayor que cero, se multiplica por el factorial del número menor inmediato."

Para calcular el factorial de cualquier número mayor que cero hay que calcular como mínimo el factorial de otro número. La función que se utiliza es la función en la que se encuentra en estos momentos, esta función debe llamarse a sí misma para el número menor inmediato, para poder ejecutarse en el número actual. Esto es un ejemplo de recursividad.

La recursividad y la iteración (ejecución en bucle) están muy relacionadas, cualquier acción que pueda realizarse con la recursividad puede realizarse con iteración y viceversa. Normalmente, un cálculo determinado se prestará a una técnica u otra, sólo necesita elegir el enfoque más natural o con el que se sienta más cómodo.

Claramente, esta técnica puede constituir un modo de meterse en problemas. Es fácil crear una función recursiva que no llegue a devolver nunca un resultado definitivo y no pueda llegar a un punto de finalización. Este tipo de recursividad hace que el sistema ejecute lo que se conoce como bucle "infinito". El siguiente es un ejemplo: omita la primera regla (la de los números negativos) de la descripción verbal del cálculo de un factorial e intente calcular el factorial de un número negativo. Se produce un error, ya que para poder calcular el factorial de, por ejemplo -24, primero hay que calcular el factorial de -25 pero, para hacerlo, primero hay que calcular el factorial de -26 y así sucesivamente. Obviamente, el bucle nunca se detendrá.

Por tanto, es muy importante cuidar al máximo el diseño de las funciones recursivas. Si en alguna ocasión sospecha que existe la posibilidad de que se produzca un bucle infinito, puede hacer que la función cuente el número de veces que se llama a sí misma. Si la función se llama a sí misma demasiadas veces, y es usted quien debe decidir cuántas veces se consideran demasiadas, se sale automáticamente.

→ Ejemplo:

```
// Función para calcular factoriales. Si se pasa un número
// no válido (por ejemplo, uno menor que cero), se devuelve
// -1 para indicar la existencia de un error. De lo contrario, el
// número se convierte al entero más cercano y se devuelve
// su factorial
function factorial(unNumero)
{
unNumero = Math.floor(unNumero); // Si no es un entero, lo
// redondea
if (unNumero < 0)
{ // Si el número es menor que cero, lo rechaza
return -1;
}
if (unNumero == 0)
{ // Si el número es 0, su factorial es 1
return 1;
}
else
return (unNumero * factorial(unNumero - 1));
// De lo contrario, la función se llama a sí misma
// recursivamente hasta terminar
}
```

4.12.2.22.5.- ALCANCE DE VARIABLES

Microsoft JScript tiene dos alcances: global y local. Si declara una variable fuera de la definición de una función, será una variable global y se podrá tener acceso a su valor y se podrá modificar desde todo el programa. Si declara una variable dentro de la definición de una función, esa variable será local. Se crea y se destruye cada vez que se ejecuta la función, no se tiene acceso a ella desde fuera de la función.

Lenguajes como C++ también tienen un "alcance de bloque". En este caso, cualquier conjunto de llaves "{}" define un nuevo alcance. JScript no admite alcances de bloque.

Una variable local puede tener el mismo nombre que una variable global, pero es totalmente distinta e independiente. En consecuencia, si se cambia el valor de una variable no se producirá ningún efecto sobre la otra. Dentro de la función en la que se declara la variable local sólo tiene significado la versión local.

→ Ejemplo:

```
var unCentauro = "un caballo con jinete"; // Definición global de
// unCentauro.
// Código de JScript, omitido por razones de brevedad
```

```
function antiguedades() // En esta función se define una
// variable local unCentauro
{ // Código de JScript, omitido por razones de brevedad
var unCentauro = "Un centauro es, probablemente, un guerrero
Escita a caballo";
// Código de JScript, omitido por razones de brevedad

unCentauro += ", mal entendido; es decir, "; // Se suma a la
// variable local
// Código de JScript, omitido por razones de brevedad
} // Fin de la función
var nadaEnParticular = antiguedades();
unCentauro += " tal y como es visto a cierta distancia por una
persona inocente.";
/*
Dentro de la función, la variable contiene "Un centauro es,
probablemente, un guerrero Escita a caballo, mal entendido, es
decir, "; fuera de la función, la variable contiene el resto de
la frase:
"un caballo con jinete tal y como es visto a cierta distancia por
una persona inocente."
*/
```

Es importante tener en cuenta que las variables actúan como si se declararan al principio de cualquier alcance en el que existen. Por este motivo en ocasiones se producen resultados inesperados.

→ Ejemplo:

```
tweak();
var unNumero = 100;
function tweak()
{
var nuevoElemento = 0; // Declaración explícita de la variable
// nuevoElemento
// Esta instrucción asigna el valor no definido (undefined) a la
// variable nuevoElemento, ya que existe una variable local con
// el nombre unNúmero
nuevoElemento = unNumero;
// La siguiente instrucción asigna el valor 42 a la variable
// local unNumero
unNumero = 42;
if (false)
{
var unNumero; // Esta instrucción nunca se ejecuta
unNumero = 123; // Esta instrucción nunca se ejecuta
} // Fin de la condición
} // Fin de la definición de la función
```

Cuando JScript ejecuta una función, en primer lugar busca todas las declaraciones de variable,

```
var algunaVariable;
```

y crea las variables con el valor inicial undefined. Si se declara una variable con un valor,

```
var algunaVariable = "algo";
```

continuará teniendo inicialmente el valor undefined y sólo tomará el valor declarado si la línea que contiene la declaración llega a ejecutarse.

JScript procesa las declaraciones de variable antes de ejecutar cualquier código, por lo que es irrelevante que la declaración se encuentre dentro de un bloque condicional o de alguna otra construcción. Una vez que JScript encuentra todas las variables, ejecuta el código en la función. Si una variable se declara implícitamente dentro de una función (es decir, si aparece en el lado izquierdo de una expresión de asignación pero no se ha declarado con var), se crea como una variable global.

4.12.2.22.6.- COPIAR, PASAR Y COMPARAR DATOS

En JScript, la forma en que se controlan los datos depende de su tipo.

4.12.2.22.7.- POR VALOR O POR REFERENCIA

Los valores numéricos o Boolean (true y false) se copian, pasan y comparan por valor. Al copiar o pasar por valor, se asigna un espacio en la memoria del equipo y se copia el valor del original en ese espacio. Si después cambia el original, la copia no se modifica (y viceversa), porque son entidades independientes.

Los objetos, matrices y funciones se copian, pasan y comparan por referencia. Al copiar o pasar por referencia, básicamente se crea un puntero al elemento original y se utiliza como si fuera una copia. Si cambia el original, modifica el original y la copia (y viceversa). En realidad, sólo hay una entidad; la "copia" realmente no es una copia, es sólo otra referencia a los datos.

Al comparar por referencia, ambas variables deben hacer referencia exactamente a la misma entidad para que la comparación se realice correctamente. Por ejemplo, el resultado de una comparación entre dos objetos Array siempre será diferente, aunque contengan los mismos elementos. Para que la comparación se realice correctamente, una de las variables debe ser una referencia a la otra. Para comprobar si los dos objetos Array contienen los mismos elementos, compare los resultados del método toString().

Por último, las cadenas se copian y pasan por referencia, pero se comparan por valor. Tenga en cuenta que dos objetos String, creados con new String("algo"), se compararán por referencia, pero si uno de los dos valores es un valor de cadena, se compararán por valor.

Debido a la forma en que se construyen los juegos de caracteres ASCII y ANSI, las letras mayúsculas preceden a las letras minúsculas en orden de secuencia. Por ejemplo, "Zoo" es menor que "alianza". Puede llamar a toUpperCase() o toLowerCase() en ambas cadenas si desea realizar una comparación en la que no se distingan mayúsculas y minúsculas.

Pasar parámetros a funciones

Al pasar un parámetro a una función por valor, está haciendo una copia independiente del parámetro y sólo existe dentro de la función. Aunque los objetos y matrices se pasen por referencia, si los sobrescribe directamente con un nuevo valor en la función, el nuevo valor no se reflejará fuera de la función. Sólo los cambios en las propiedades de los objetos o de los elementos de las matrices son visibles fuera de la función.

→ Ejemplo:

```
// Esto sobrescribe su parámetro, por lo que el cambio
// no se refleja en el código de llamada
function Sobrescribir(param)
{
// sobrescribir el parámetro; esto no se verá en
// el código de llamada
param = new Object();
param.message = "Esto no funcionará";
}
// Esta acción modifica una propiedad del parámetro, que
// puede verse en el código de llamada
function Actualizar(param)
{
// Modifica la propiedad del objeto; esto se verá en
// el código de llamada
param.message = "Me han cambiado";
}
// Crea un objeto y le asigna una propiedad
var obj = new Object();
obj.message = "Este es el original";
// Llama a Sobrescribir e imprime obj.message. Observe que no ha
// cambiado
Sobrescribir(obj);
window.alert(obj.message); // Todavía muestra "Este es el
// original"
// Llama a Actualizar e imprime obj.message. Observe que ha
// cambiado
Actualizar(obj);
window.alert(obj.message); // Muestra "Me han cambiado"
```

4.12.2.22.8.- USAR MATRICES

Indización de matrices

Las matrices en JScript son dispersas.

Es decir, si tiene una matriz con tres elementos, numerados como 0, 1 y 2, puede crear un elemento 50 sin preocuparse por los elementos que van del 3 al 49. Si la matriz tiene una variable de longitud automática (consulte objetos intrínsecos para obtener una explicación acerca del control automático de la longitud de una matriz), la longitud de la variable tendría el valor 51, en lugar de 4. Se pueden crear matrices en las que no haya espacios vacíos en la numeración de los elementos, pero no es necesario.

En JScript, los objetos y las matrices son casi idénticos. Las dos diferencias principales estriban en que los objetos normales no tienen una propiedad de longitud automática y las matrices no tienen las propiedades y métodos de un objeto.

Referencias a matrices

Debe hacer referencia a las matrices mediante el uso de corchetes "[]". Los corchetes encierran un valor numérico o una expresión que se evalúa como un número entero. En el siguiente ejemplo se supone que la variable numEntr se define en alguna parte de la secuencia de comandos y se le asigna un valor.

```
laLista = libretaDirecciones[numEntr];
laPrimeraLinea = laLista [1];
```

Objetos como matrices asociativas

Por lo general, el operador punto "." se usa para tener acceso a las propiedades de un objeto. Por ejemplo:

```
miObjeto.unaPropiedad
```

En este caso el nombre de la propiedad es un identificador. También puede tener acceso a las propiedades de un objeto mediante el operador índice "[]". En este caso, el objeto se trata como una matriz asociativa. Una matriz asociativa es una estructura de datos que permite asignar dinámicamente valores de datos arbitrarios con cadenas arbitrarias. Por ejemplo:

```
miObjeto["unaPropiedad"] // Como en el ejemplo anterior
```

Aunque el uso del operador de índice se suele asociar con el acceso a elementos de una matriz, cuando se usa con objetos, el índice siempre es el nombre de la propiedad expresado como un literal de cadena.

Observe la evidente diferencia entre las dos maneras de tener acceso a las propiedades de objeto.

Operador	El nombre de la propiedad se trata como …	Indica que el nombre de la propiedad …
Punto "."	un identificador	no puede tratarse como datos
Índice "[]"	un literal de cadena	puede tratarse como datos

Esta diferencia resulta útil cuando no se sabe cuáles serán los nombres de propiedad hasta el momento de la ejecución (por ejemplo, cuando se construyen objetos basados en información escrita por el usuario). Para extraer todas las propiedades de una matriz asociativa, debe usar el bucle for … in.

Comprobar datos

Al realizar una comprobación por valor, se comparan dos elementos para establecer si son iguales. Normalmente, esta comparación se realiza byte por byte. Al comprobar por referencia, está determinando si dos elementos son punteros a un único elemento original. Si es así, el resultado es que son iguales; si no, aunque contengan los mismos valores exactos, byte por byte, el resultado es que no lo son.

Copiar y pasar cadenas por referencia ahorra memoria; pero como no puede cambiar cadenas una vez creadas, es posible compararlas por valor. Esto permite comprobar si dos cadenas tienen el mismo contenido aunque una se haya generado totalmente aparte de la otra.

Caracteres especiales

JScript admite caracteres especiales que le permiten incluir en las cadenas algunos caracteres que no se pueden teclear directamente. Cada uno de estos caracteres comienza con una barra invertida. La barra invertida es un carácter de escape que se utiliza para indicar al intérprete de JScript que el siguiente carácter es especial.

Secuencia de escape	Caracter
\b	Retroceso
\f	Avance de página
\n	Avance de línea (nueva línea)
\r	Retorno de carro
\t	Tabulador horizontal (Ctrl-I)
\'	Marca de comillas simples
\"	Marca de comillas dobles
\\	Barra invertida

Observe que, dado que la barra invertida se utiliza como carácter de escape, no se puede utilizar directamente en la secuencia de comandos. Si desea escribir una barra invertida debe escribirla dos veces seguidas (\\).

→ Ejemplo:

```
document.write('La ruta de la imagen es
C:\\web\\mipagina\\gifs\\jardin.gif.');
document.write('El texto dice: "Después de la nevada del \'97, la
casa de la abuela se cubrió de nieve."');
```

5. CAPITULO 5: COMUNICACIONES

Las comunicaciones son el núcleo del Sistema de Control Galileo VI.

Una de las partes fundamentales del desarrollo de un sistema de control es la depuración del funcionamiento una vez programados los secuenciadores apropiados.

Para ello es necesario que exista una comunicación entre el entorno de desarrollo y los computadores donde se está ejecutando el sistema de control.

Esta comunicación, además de servir para depurar, permite el acceso a formularios que realizan operaciones que de otra manera no se podrían realizar y muestran información de lo que está ocurriendo.

5.1.- MODO CONECTADO

5.1.1.- ¿CÓMO EJECUTAR EL MODO CONECTADO?

Las opciones del modo conectado de cada pantalla, si las tuviera, están detalladas en la ayuda de la propia pantalla.

Para iniciar la comunicación de un proyecto, selecciónelo en el árbol de proyectos (si tiene varios abiertos en el entorno) y haga clic en  o bien desde el menú principal Comunicaciones > Modo conectado.

 El proyecto seleccionado para el Modo conectado muestra el icono  .

Desde este momento la aplicación intenta establecer las comunicaciones de forma puntual (P2P con las computadoras definidas en el proyecto). Las computadoras deben estar accesibles por TCP/IP desde la red donde se está ejecutando el entorno de desarrollo. Los nombres de computadora son sensibles al uso de mayúsculas y minúsculas, por lo que es necesario tener cuidado a la hora de nombrarlas.

El panel de Información general muestra un mensaje de si la conexión es correcta o ha fallado.

Para terminar la comunicación de un proyecto, haga clic en  o bien desde el menú principal Comunicaciones > Modo desconectado.

5.1.2.- ¿QUÉ PODEMOS EJECUTAR EN EL MODO CONECTADO?

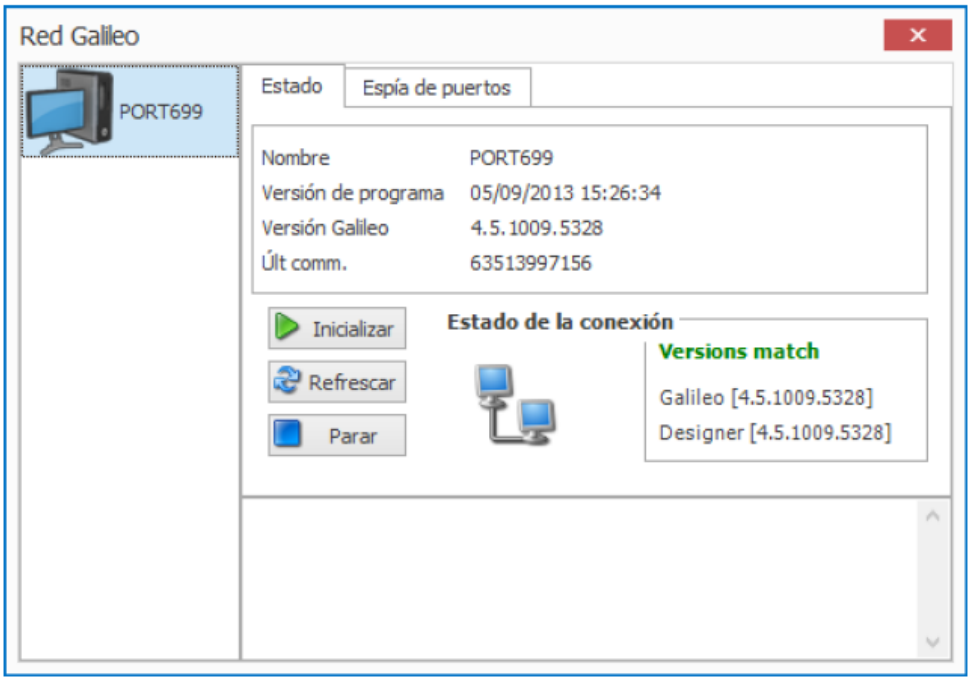
Opciones, como “Modo Online de la pantalla Secuenciadores” en la página 85 y el “Editor de Tracking” en la página 88, o las ofrecidas en la “¿Cómo visualizar variables de un secuenciador?” en la página 113, están explicadas con detalle en la ayuda las pantallas “**Secuenciadores**” e “**Inspector**”.

Además de todo esto, también son accesibles las siguientes opciones en el proyecto solo si estamos en el **Modo conectado**:

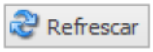
- Ver Red Galileo.
- Ver Averías.
- Restringir comunicaciones.

5.1.2.1.- VER RED GALILEO

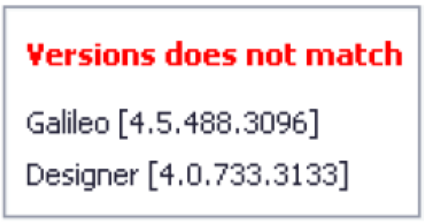
Opción accesible haciendo clic en  o desde el menú principal **Comunicaciones** > **Mostrar Red Galileo**.



Pestaña Estado > Las opciones disponibles son:

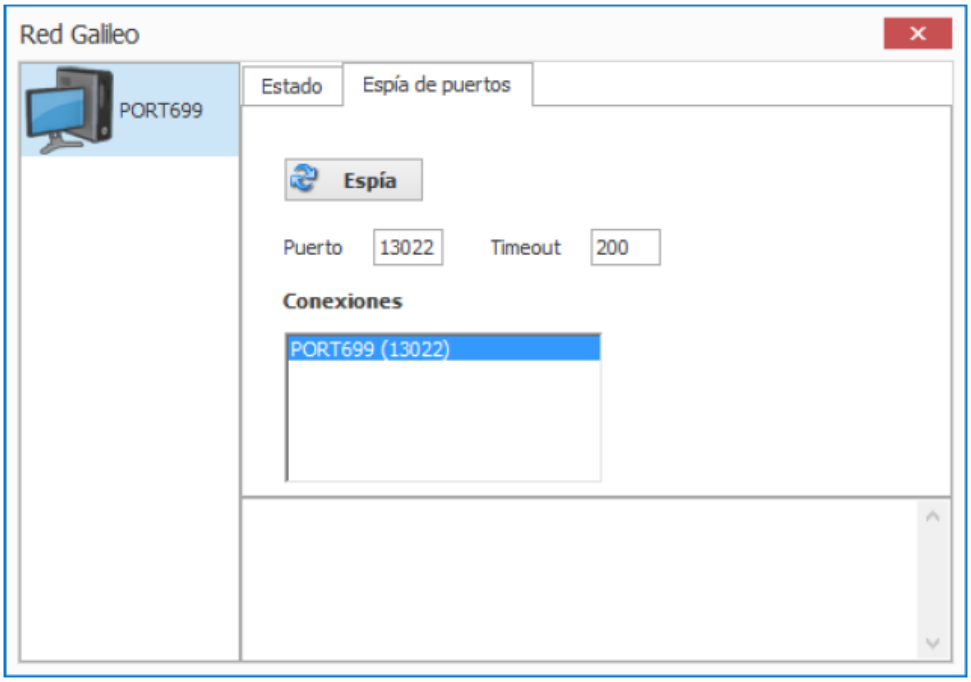
BOTÓN	DESCRIPCIÓN
	Envía un comando de control al servicio de Galileo IV indicando que inicie la ejecución del programa de control actual. ▪ Si el programa de control del computador remoto está parado, se inicia, ejecutando la acción de Arranque Caliente del programa. ▪ Si el programa de control estaba iniciado, primeramente se para y luego se continúa en el modo mencionado ejecutando la acción de Arranque Caliente del programa.
	Envía un comando de control al servicio de Galileo IV indicándole que detenga la ejecución del programa de control actual. ▪ El bus conectado a sus tarjetas es desactivado, y las salidas físicas se resetean. También se descarga el programa de control activo en ese momento de la memoria. ▪ Si el programa de control ya estaba parado, esta opción no tiene ningún efecto significativo si Galileo IV ya está parado.
	Refresca la información de varios campos que indican lo siguiente: ▪ *Versión del Programa*: Versión del programa cargado en memoria por el sistema de control. Es distinta para cada programa compilado por el entorno de desarrollo y puede servir para identificar una de ellas. ▪ *Versión Galileo:* Versión del Sistema de control usada. No cambia a menos que instalemos una nueva versión de Galileo IV. ▪ *Última Comunicación*: Fecha y hora de la última comunicación que se produjo entre el entorno de desarrollo y el programa de control.
 	Estado de la conexión: ▪ Conexión correcta: computador remoto funcionando y ejecutando un programa de control que se muestra en las etiquetas correspondientes. ▪ Conexión errónea: no hay comunicación con el computador remoto (imposible acceder a él, posiblemente esté desconectado o el servicio de Galileo IV no puede responder a las peticiones que se le hacen).

En el caso de que las versiones de Galileo IV y Designer IV sean distintas (situación errónea) se le indicará mediante el siguiente mensaje:



Pestaña Espía de puertos > Permite conocer las conexiones en un determinado puerto. Útil para conocer qué PCs tienen el servicio arrancado en dicho puerto.

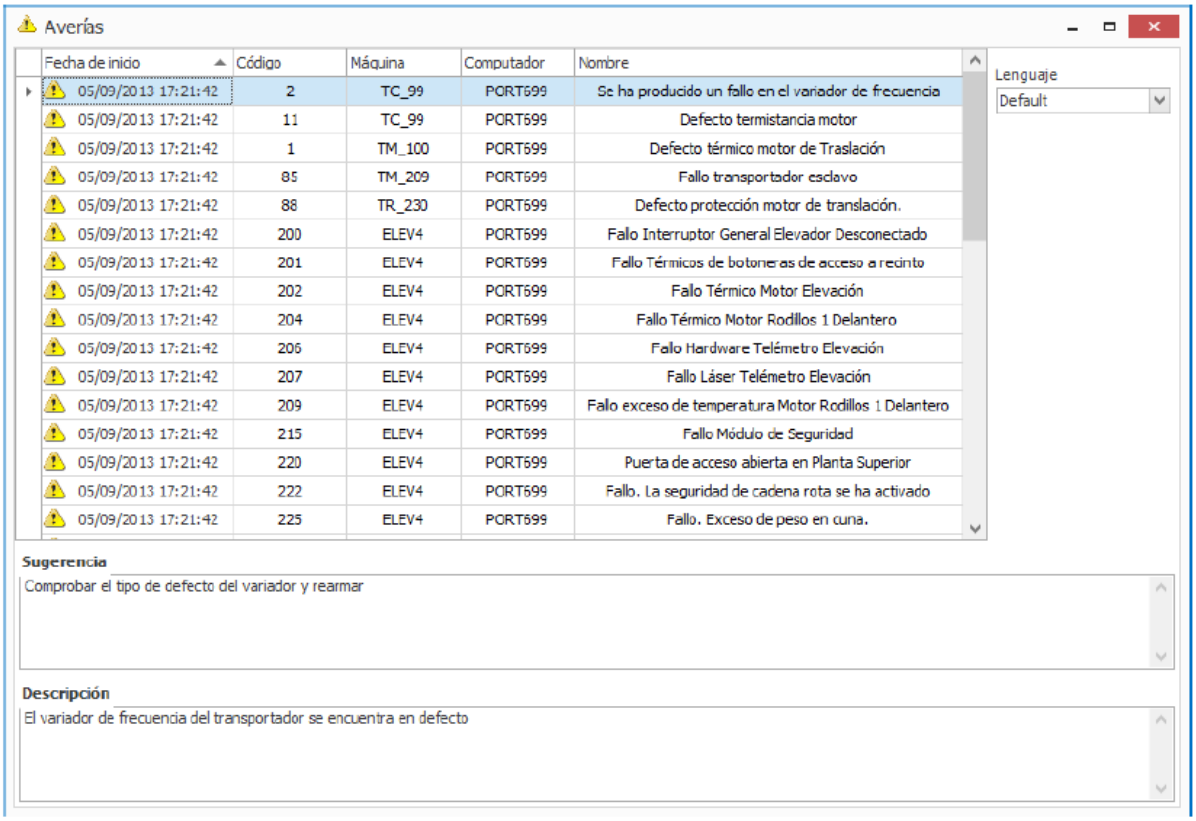
También puede realizar una prueba de conexión seleccionando la opción **Ping** desde el menú contextual sobre el computador deseado en el árbol del proyecto.



5.1.2.2.- **VER AVERÍAS**

Opción accesible haciendo clic en  o desde el menú principal **Comunicaciones** > **Mostrar Averías**.

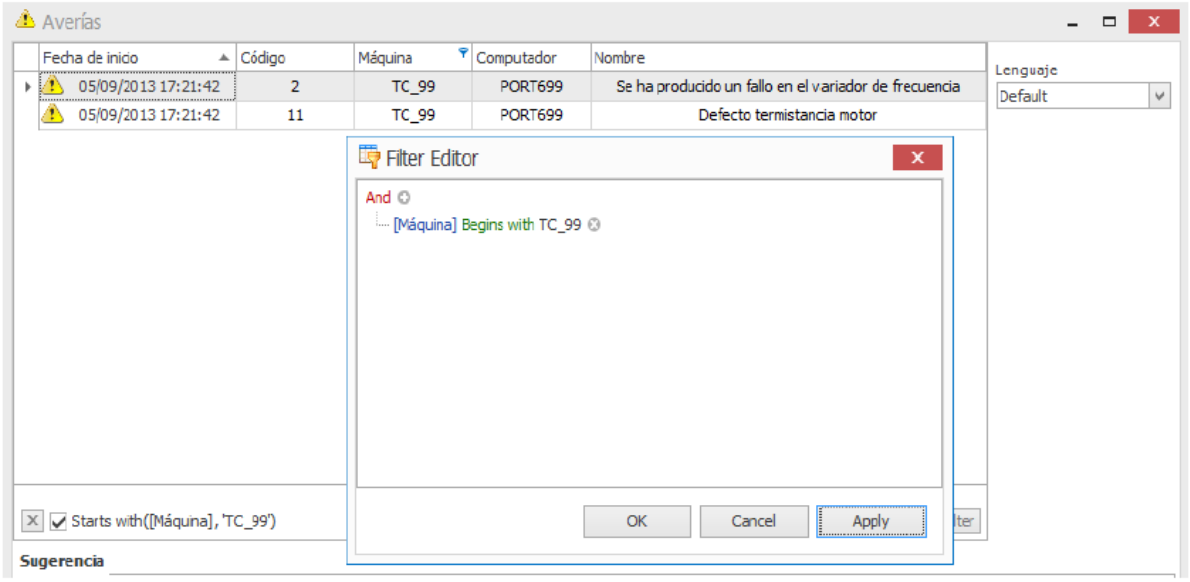
Esta ventana puede funcionar como emergente cuando se active cualquiera de las averías definidas en la pantalla "**Configuración TA**" y también puede disparar una alarma sonora. Para activar estas opciones puede consultar "Solapa Opciones de Status" en la página 19.



Esta ventana se actualiza periódicamente cuando se produzca un cambio en la información referente a las averías y mientras no se cierre o se pase a Modo Offline.

Para modificar el idioma en el que se muestran las averías, siempre y cuando se hayan definido distintos lenguajes, selecciónelo en el desplegable "**Lenguajes**". Esta configuración se salvará para posteriores ejecuciones.

Como en el resto de tablas de la aplicación puede ordenar o aplicar filtros, por ejemplo:



5.1.2.3.- **RESTRINGIR COMUNICACIONES**

Pueden existir casos en los que es necesario o recomendable el restringir las comunicaciones entre ciertos computadores. Un caso típico se produce cuando la red de comunicaciones es wireless y existen problemas de conexión, velocidad, etc.

En estos casos, es probable que no necesitemos que Designer IV se comunique con ciertos computadores. Sin embargo, por defecto, Designer IV está continuamente pidiendo datos a todos los Galileos de la instalación. En este caso es recomendable limitar la comunicación con ciertos computadores con el fin de minimizar la carga de la red.

Es importante tener en cuenta las diferentes comunicaciones que se tienen en sistema Galileo IV:

- **Comunicaciones Galileo – Galileo**: los distintos Galileos de la instalación intercambian continuamente información. Esta información es la que utilizan durante la ejecución del programa de control, por ejemplo, variables, trackings, etc. Para restringir este tipo de comunicaciones es necesario hacerlo desde la consola de Galileo IV (consulte el manual del **Manual de Galileo IV**).

- **Comunicaciones Galileo – Designer**: Designer IV pide información acerca de la imagen de proceso y averías a todos los computadores incorporados en el proyecto. Se puede configurar el proyecto de Designer IV para que cada computador tenga su propia lista de comunicaciones excluidas. De esta manera, el mismo proyecto, dependiendo del computador en el que se esté ejecutando Designer IV en ese momento se comunica con unos u otros computadores. Para restringir las comunicaciones en un determinado computador, puede consultar el apartado "¿Cómo editar un computador?" en la página 46.
  También puede configurar un PC, el cual solo es para desarrollo, de forma que no pertenezca a los computadores del proyecto, más información en la "Solapa Comunicaciones".

Universidad de Oviedo

Trabajo Fin de Máster realizado por

Victoria Casares García

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

ANEXO III: Manual EasyWCS

Modelado de Transelevador
Automatic Warehouse Studio (AWS)

Julio de 2018

ÍNDICE MANUAL USUARIO EASWCS

1. CAPÍTULO 1: INTRODUCCIÓN

La numerosa combinación existente en el mercado para automatizar y gestionar procesos con distintas Bases de Datos ha creado la necesidad de disponer de una herramienta capaz de agilizar y controlar estas posibles combinaciones. El sistema de MECALUX EasyWCS funciona como un coordinador general. Se encarga de gestionar los transportes a través de una base de datos (BBDD) ofreciendo una interfaz sencilla e intuitiva, facilitando la configuración, supervisión y control de las tareas en las estaciones de una instalación.

Esta herramienta cumple dos roles principales:

1. Nos ofrece la robustez de un sistema de transporte basado en la experiencia de otros productos de Mecalux:
   - Servicio de Windows que se ejecuta durante el uso de la instalación y que permite la comunicación entre los sistemas implicados
   - Gestión de tareas mediante preselecciones estándar.
   - Particularización por parte del implementador de casos especiales según necesidades del cliente.

2. Nos ofrece la potencialidad de una interfaz gráfica de usuario (GUI), esto es:
   - Sistema de supervisión en tiempo real de las estaciones de la instalación claro e intuitivo.
   - Facilidad de configuración de estaciones y rutas, destacando la rapidez en todas aquellas tareas repetitivas que implican la creación de muchas estaciones y rutas.
   - Visor de tramas y tiempos de ejecución implicados en la comunicación. Logs y diagnosis.

2. CAPÍTULO 2: INSTALACIÓN DE EASYWCS

Es recomendable instalar siempre la versión más reciente de EasyWCS. Dicha versión está disponible en la web http://itswservices.mecalux.com/downloads.aspx

2.1.- REQUERIMIENTOS Y PROCESO DE INSTALACIÓN

2.1.1.- REQUERIMIENTOS

Los requerimientos mínimos para instalar el software EasyWCS son:

- PC con Windows XP, Windows 7, Windows 8.1, Windows 2000 (version servidor o profesional) o Windows Server 2012 R2.
- Si se desea conectar con las estaciones remotas (modo online), es necesario que el ordenador disponga de acceso vía TCP/IP a dicha estación.
- FrameWork NET 4 o superior

2.1.2.- PROCESO DE INSTALACIÓN

Para iniciar la instalación haga doble clic sobre el icono correspondiente al instalable de la versión deseada:

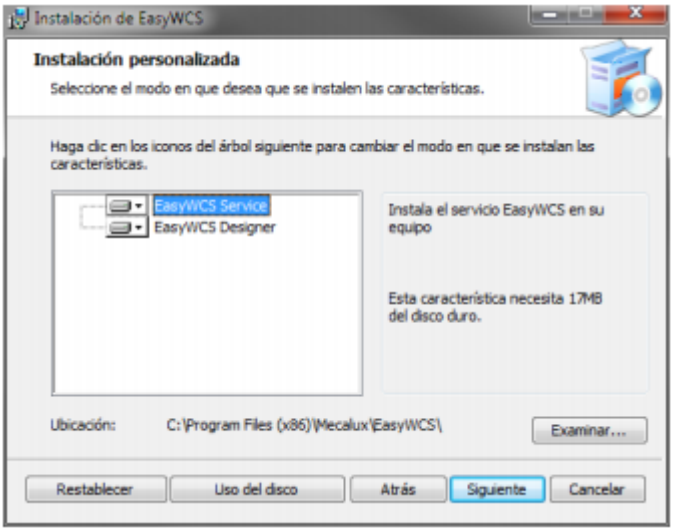
Tras las pantallas de bienvenida, acepte el diálogo del contrato de licencia.

Seleccione un directorio del disco donde instalar EasyWCS. Por defecto el directorio que se muestra es C:\Archivos de programa\Mecalux\EasyWCS\.

Dependiendo de donde tengamos instalado el Sistema Operativo o de la versión del mismo puede cambiar.

Seleccione las opciones de instalación:

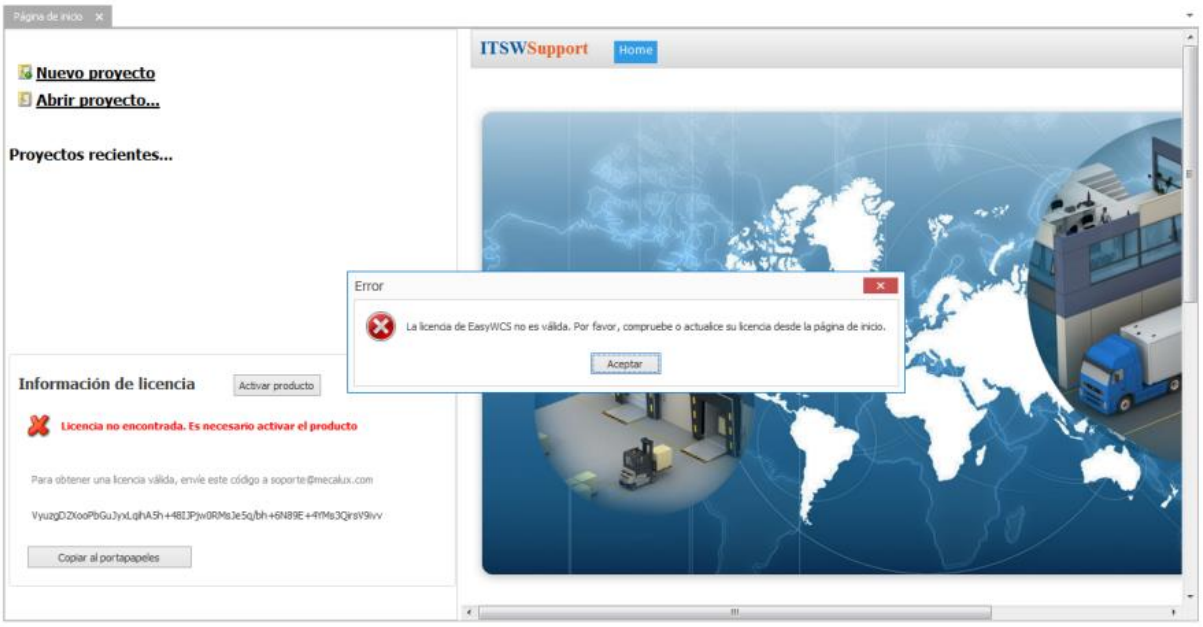
- EasyWCS Service: Servicio de comunicación entre el sistema de control (TMS) y el sistema de gestión de almacén (WMS).
- EasyWCS Designer: Diseñador o Interfaz gráfica de usuario.

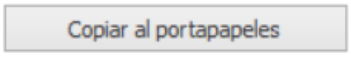


Con esto comienza la instalación.

Al final de proceso, EasyWCS instala un acceso directo del Diseñador en el escritorio:

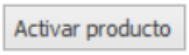
Si se trata de la primera instalación del software, EasyWCS muestra en la pantalla “Página de inicio” que no existe licencia activada del mismo:



Para solicitar la licencia copie el código mostrado (clave que identifica de forma única el computador de instalación del software) haciendo clic en 

y envíelo a soporte@mecalux.com.

Tras ello, recibirá un fichero denominado "**EasyWCS.lic**".

Para instalar la licencia haga clic en 

y seleccione el fichero de licencia recibido.

El resultado de la activación se indicará mediante un mensaje.

A continuación, mostramos algunos mensajes posibles relacionados con la licencia:

- "Licencia no encontrada. Es necesario activar el producto": EasyWCS no es capaz de localizar el archivo de licencia. Este fichero debería estar localizado en (C:\Documents and Settings\All Users\ApplicationData\Mecalux\EasyWCS).

  Si el archivo de licencia no aparece en esta ubicación, puede ser que el proceso de activación haya fallado. En ese caso, puede tratar de copiar manualmente el archivo .lic a la localización correcta.

- "Producto inválido. La licencia no es para un producto EasyWCS": La licencia es válida, pero no para EasyWCS, es decir, es una licencia de otro producto de MECALUX.

- "Licencia inválida. No está firmada por Mecalux": La licencia no es válida, es decir, no está firmada por MECALUX. Se debe comprobar que el fichero es el original enviado por MECALUX y que no ha sufrido alteración alguna.

- "Licencia inválida. La licencia no es para esta máquina": La licencia es válida (está firmada por MECALUX), pero para otro computador. Se debe comprobar que no se ha modificado el nombre del equipo, o sustituido componentes hardware.

- "Esta licencia es una versión de prueba. Días restantes...": Ciertas licencias pueden tener un máximo de días de prueba.

- "Error desconocido en la licencia": La licencia no es válida, pero no se conoce el motivo. Póngase en contacto con el departamento de ITSW.

 En cualquier momento puede consultar el estado de la licencia instalada haciendo clic en la opción "**Acerca de**" del menú principal.

3. CAPITULO 3: GUÍA RÁPIDA DEL DISEÑADOR

En este capítulo se explicarán las distintas pantallas y funcionalidades del Diseñador.

Antes de ello, mostraremos opciones que son comunes a todas las pantallas.

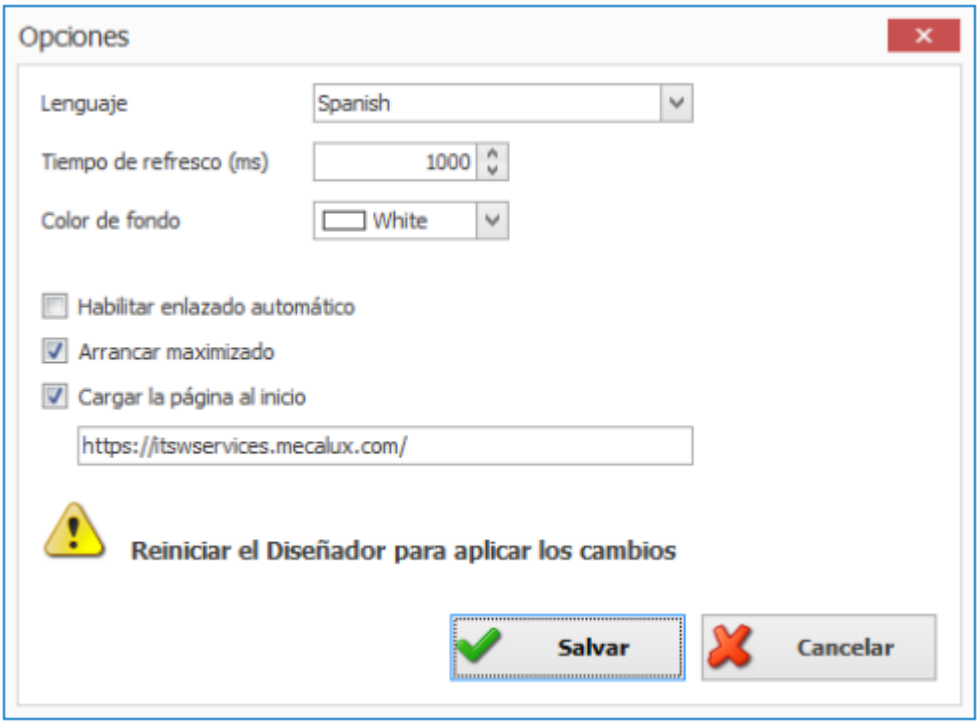
Las opciones más destacables del menú principal son:

- Archivo > Propiedades del proyecto: más información en "**Propiedades del proyecto**" en la página 12.
- Archivo > Importar desde XML: más información en **"¿Cómo importar un nuevo proyecto?**" en la página 12.
- Archivo > Generar informe: más información en "**Generador de informes**" en la página 40.
- Edición >Estaciones: más información en "**¿Cómo crear una estación?"** en la página 19.
- Edición >Opciones: más información en "**Opciones del entorno**" en la página 9
- Edición >Configuración de rutas: más información en **"¿Cómo crear rutas entre las estaciones del layout?**" en la página 24.
- Compilación > Comprobar compilación: más información en "**Compilación del código**" en la página 31.
- Compilación > Mostrar código generado: más información en "**Compilación del código**" en la página 31.
- Comunicaciones > Modo Online: más información en "**Modo Online**" en la página 34.
- Comunicaciones > Red: más información en "**Ver Red**" en la página 38.
- Comunicaciones > Gestión de la BBDD interna: más información en "**Gestión de la BBDD interna**" en la página 39.
- Vista >Mostrar Mensajes: más información en "**Compilación del código**" en la página 31.
- Vista >Mostrar Código: más información en "**Pantalla "Código**"" en la página 30.
- Vista >Mostrar Diagrama: más información en "**Pantalla "Diagrama"**" en la página 18.
- Vista >Mostrar Órdenes: más información en **"¿Cómo generar una orden?**" en la página 41.
- Vista >Mostrar Tablas: más información en "**Pantalla Tablas**" en la página 43.
- Servicio EasWCS Local > Configurar Servicio EasyWCS: más información en "**Herramienta de configuración**" en la página 58.

- Servicio EasWCS Local > Comprobar configuración de las comunicaciones: más información en "**Asistente de comprobación de las comunicaciones**" en la página 64.

3.1.- OPCIONES DEL ENTORNO

El entorno de desarrollo dispone de opciones configurables que facilitan al usuario la posibilidad de personalizar el entorno o que le permita obtener un mejor rendimiento del mismo.



- **Lenguaje**: Puede cambiarlo seleccionando el idioma en el desplegable.
- **Tiempo de refresco**: Periodo de actualización (ms) de los datos
- mostrados en la pantalla "Diagrama".
- **Color de fondo**.
- **Habilitar enlazado automático**: Permite modificar las líneas entre estaciones pudiendo así evitar solapes entre los distintos elementos mostrados en la pantalla de "Diagrama".
- **Arrancar maximizado**: El Diseñador arranca a pantalla completa.
- **Cargar la página al inicio**: Seleccionado, carga en la pantalla "Página de inicio" (más información ver "Página de inicio" en la página 11) la página web indicada. Por defecto es https://itswservices.mecalux.com/

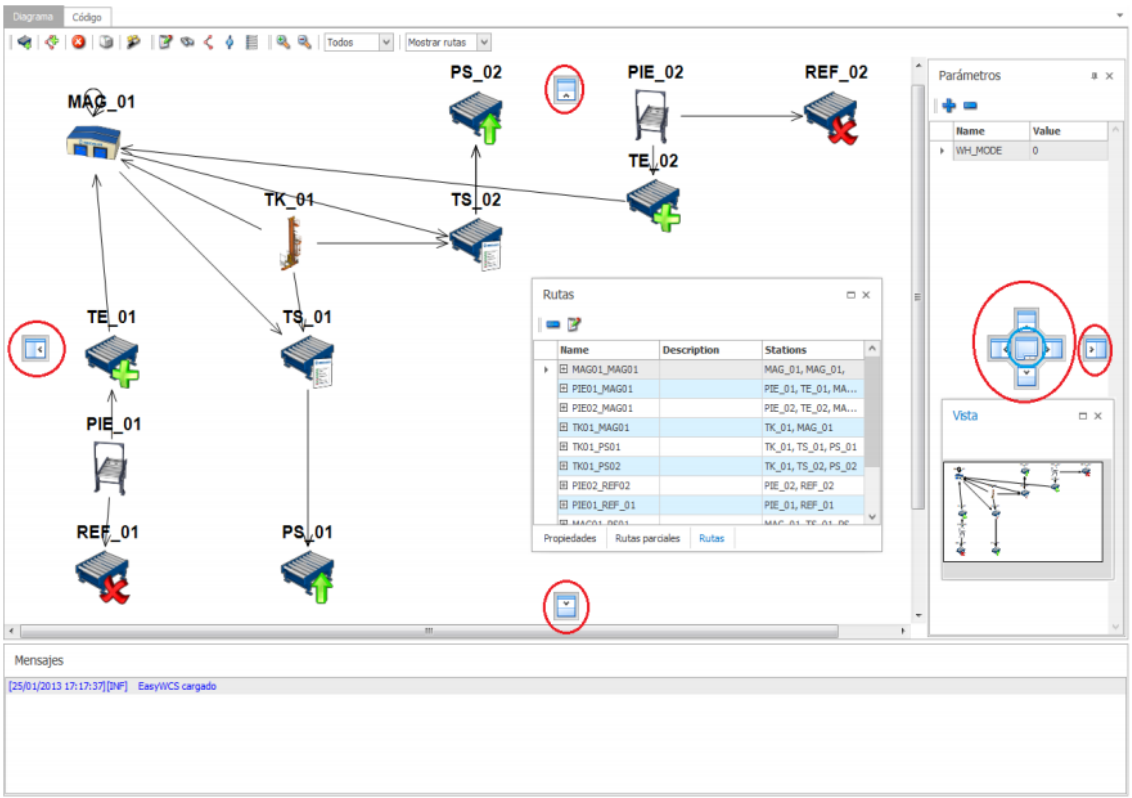
Haga clic en 

para guardar los cambios.

Para que se apliquen los cambios es necesario cerrar y abrir el Diseñador.

El Diseñador ofrece un docking avanzado, esto es, permite deslocalizar los paneles. Muy útil para poder visualizar más información de forma simultánea (requiere más recursos del sistema).

Un ejemplo de paneles deslocalizados:



Para volver a localizar un panel, dentro de otra ventana o a nivel de pantalla general, debe arrastrar éste sobre las marcas que aparecen en la imagen en rojo. Si desea colocar los paneles a modo de pestañas, siempre que sea posible, debe elegir la opción marcada en azul.

3.2.- PÁGINA DE INICIO

La pantalla “Página de inicio” se muestra al inciar el Diseñador de EasyWCS.

Desde ella se pueden realizar las siguientes opciones:

- **Nuevo proyecto**: más información en “¿Cómo crear un nuevo proyecto?” en la página 12.
- **Abrir proyecto**: más información “¿Cómo abrir un proyecto ya creado?” en la página 16.
- **Proyectos recientes...**
- **Infomación de licencia**: solicitud, activación y consulta de la licencia de EasyWCS (ver más información en el “Proceso de instalación” en la página 5).
- **Navegador web:** Permite mostar la página definida en “Opciones del entorno” en la página 9. Desde este navegador el usuario puede loguearse en el Portal para realizar disintas gestiones como por ejemplo descargar el setup, consultar la documentación, abrir un item sobre la aplicación, etc.

La página de inicio se oculta cuando se crea o abre un proyecto.

3.3.- PROPIEDADES DEL PROYECTO

Recuerde, haga clic en situado en la barra de herramientas, para guardar las modificaciones realizadas en el proyecto. Es imprescindible reiniciar el servicio para que los cambios surtan efecto.

3.3.1.- ¿CÓMO CREAR UN PROYECTO NUEVO?

Para crear un nuevo proyecto haga clic en la opción Archivo > Nuevo proyecto del menú principal o bien seleccione esta misma opción desde la "Página de inicio".

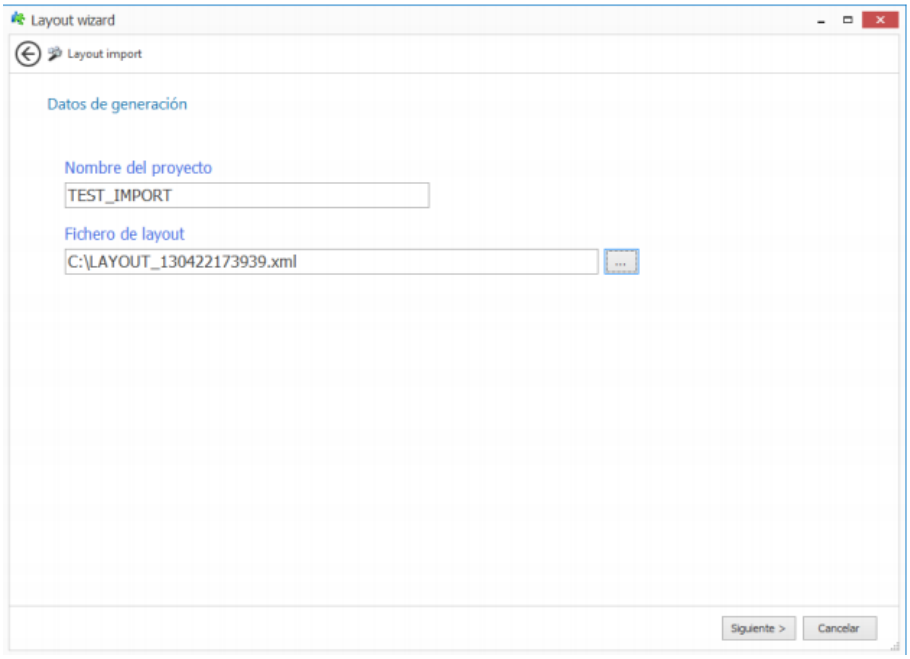
Indique el nombre y la ubicación donde será creado. La extensión del mismo es .wcsproj (Warehouse Control System Project).

3.3.2.- ¿CÓMO IMPORTAR UN NUEVO PROYECTO?

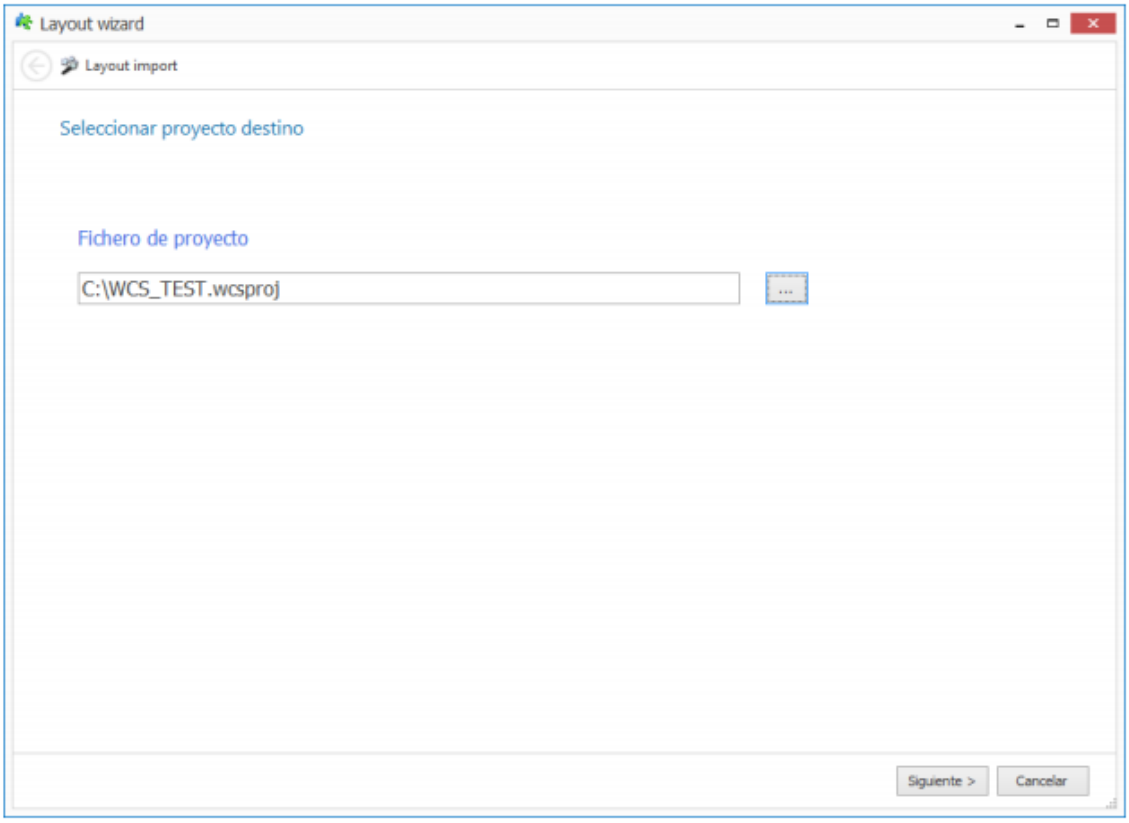
EasyWCS permite realizar la importación de proyectos en formato XML previamente generados desde EasyS. Este XML contiene información sobre estaciones y rutas.

Para importar un proyecto en formato XML haga clic en la opción Archivo > Importar desde XML del menú principal.

Tras ello, EasyWCS le ofrece un asistente que le guiará en la importación del proyecto. Las pantallas más significativas son:

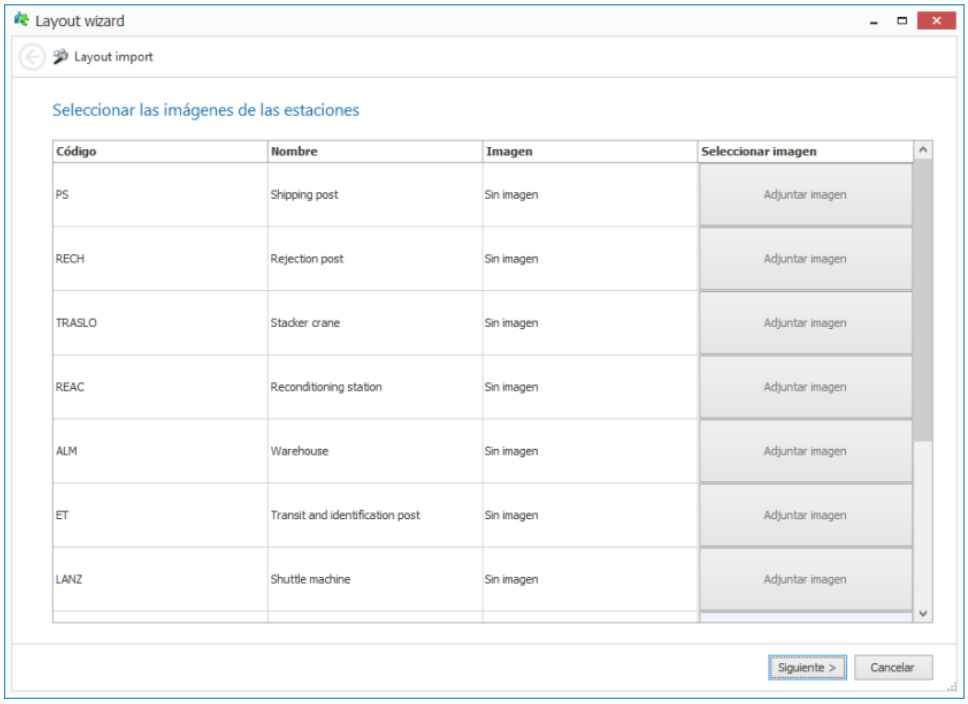


Indique el "Nombre del proyecto" y seleccione, haciendo clic en ..., el fichero en formato XML. Tras seleccionarlo haga clic en Siguiente >.

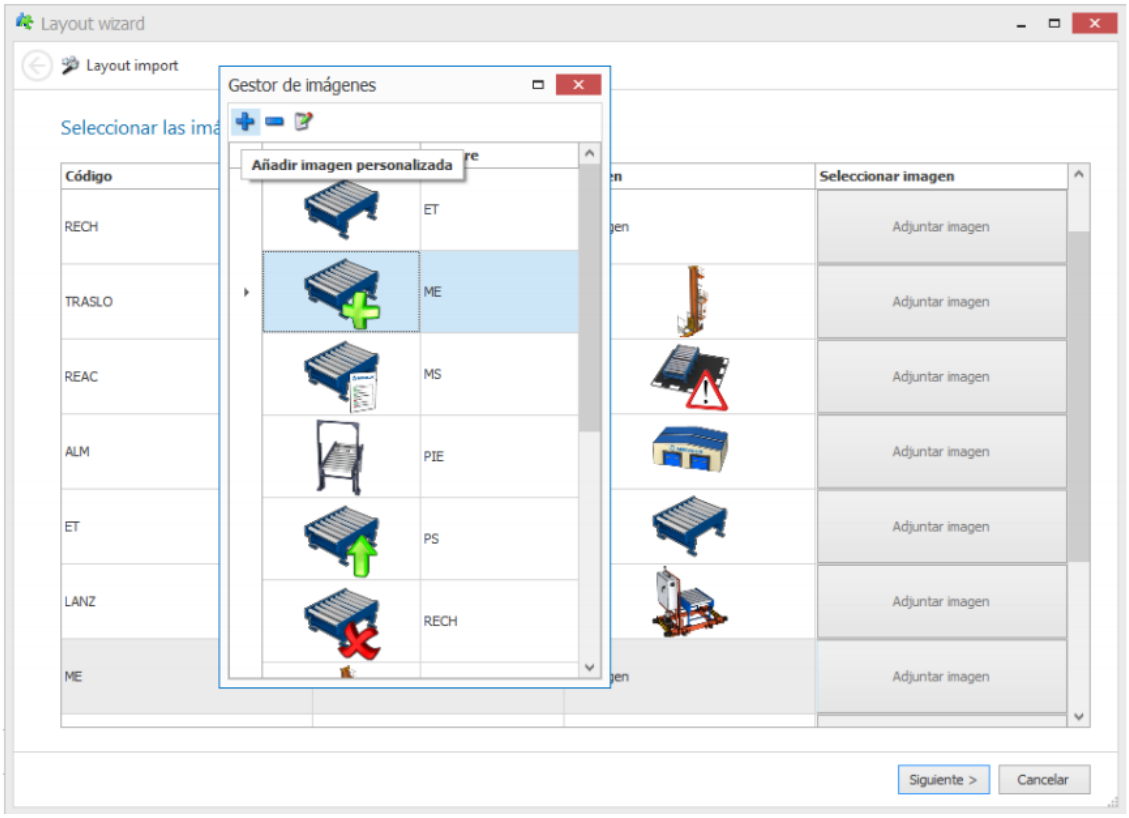


Haga clic en ... e indique el nombre y la ubicación donde será creado el proyecto. La extensión del mismo es .wcsproj (Warehouse Control System Project).

No seleccione un proyecto .wcsproj creado con anterioridad ya que este se sobreescribiría.

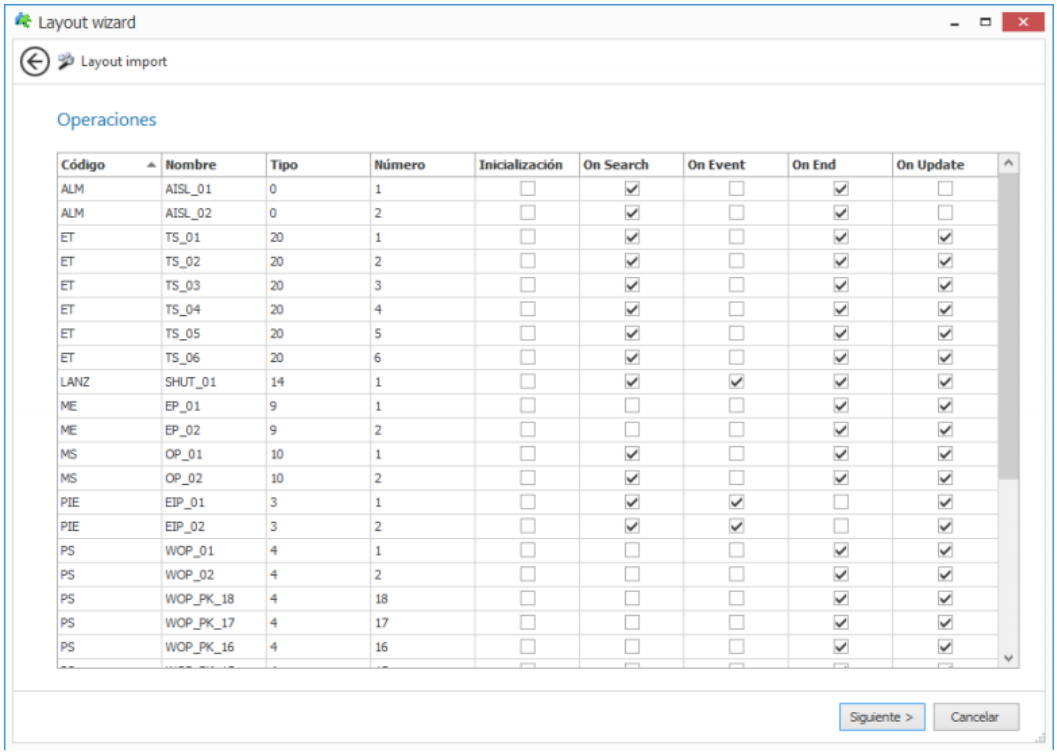


En esta pantalla EasyWCS le ofrece la posibilidad de asignar una imagen a cada tipo de estación de los existentes en el XML importado.

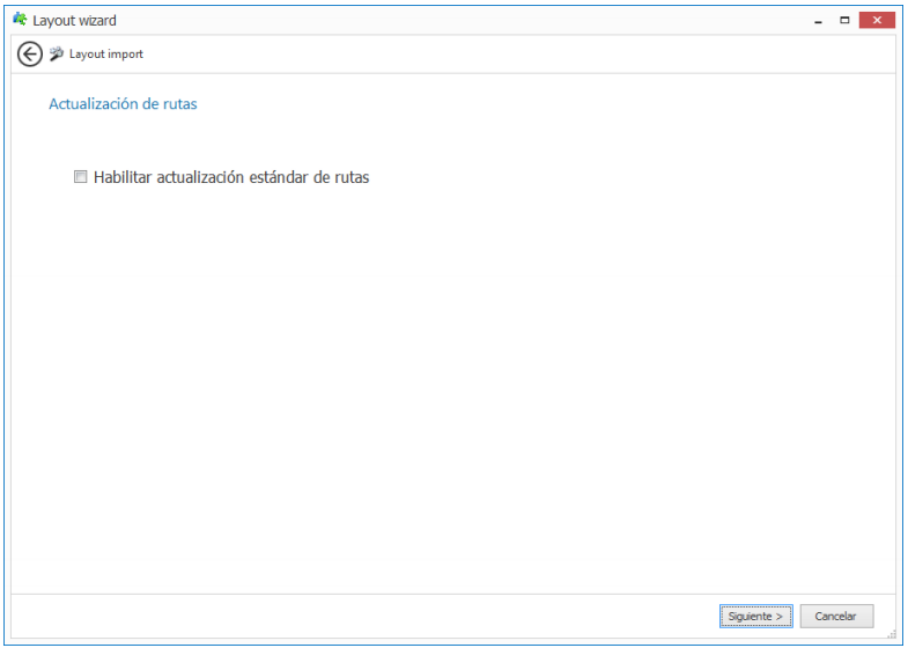


Haga clic en 

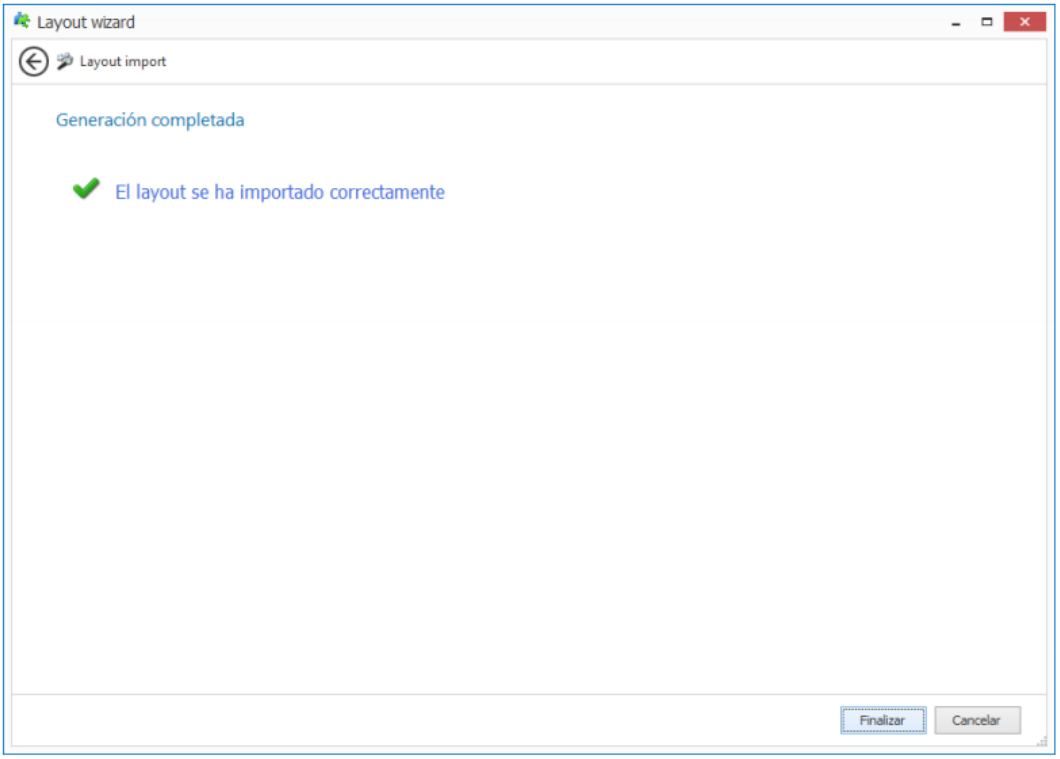
para mostrar el "Gestor de Imágenes" donde podrá visualizar, importar y eliminar imágenes (en formato PNG).



En esta pantalla EasyWCS muestra todas las estaciones existentes en el layout, indicando el nombre, número y tipo. Para cada una de ellas se preseleccionan las operaciones estándar. Puede cambiar esta preselección en función de sus necesidades.



De forma similar a las operaciones en las estaciones, para las rutas es posible asociar una función estándar común para todas ellas.



Tras estos pasos se da por finalizada la importación, generándose en el proyecto indicado el layout y la información facilitada en el XML

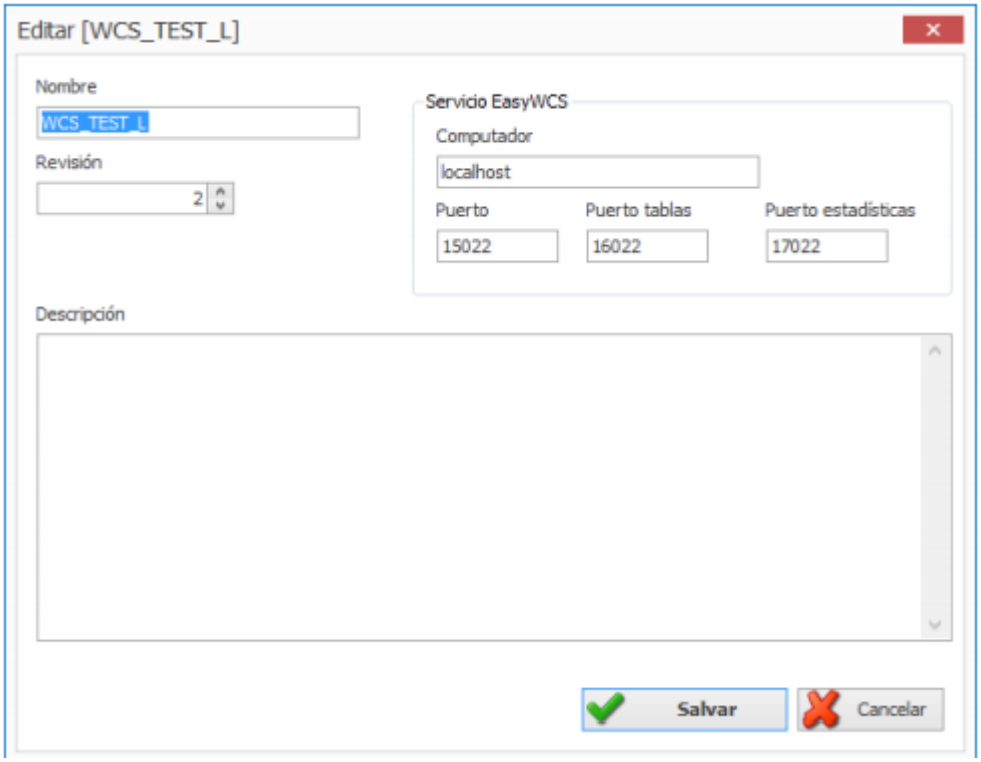
3.3.3.- CÓMO ABRIR UN PROYECTO YA CREADO

Para abrir un proyecto ya creado haga clic en la opción Archivo → Abrir del menú principal o bien seleccione esta misma opción desde la "**Página de inicio**".

La barra inferior del Diseñador le indica el proyecto abierto y el estado del mismo, es decir, si está modificado o no.

3.3.4.- CÓMO EDITAR LAS PROPIEDADES DE UN PROYECTO

Haga clic en la opción Fichero →  Propiedades de proyecto del menú principal para introducir la información del mismo.



Los campos que aparecen en este formulario son determinantes a la hora de ejecutar el proyecto. Esta información puede ser modificada a posteriori.

Los campos más destacables que nos encontramos en esta pantalla son:

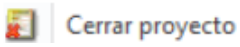
- **Nombre**: Debería ser un nombre corto y descriptivo del proyecto, puesto que será el indicador que luego aparecerá en la parte superior del Diseñador.
- **Revisión**: Campo informativo sobre la revisión del proyecto.
- **Servicio EasyWCS**
  - **Computador**: Nombre o dirección IP donde está el servicio de EasyWCS.
  - **Puerto**: Puerto de comunicación entre el Diseñador y el servicio.
  - **Puerto de tablas**: Puerto de comunicación con el servicio de datos de EasyWCS que permite mostrar online los datos de las tablas ubicadas en la BBDD del cliente.
  - **Puerto estadísticas**: Puerto de comunicación con el servicio de datos de EasyWCS que permite mostrar online estadísticas de las operaciones realizadas.
- **Descripción**: Anotaciones del proyecto.

Finalmente, haga clic en 

para guardar los cambios.

 Es imprescindible reiniciar el servicio para que los cambios en el mismo surtan efecto

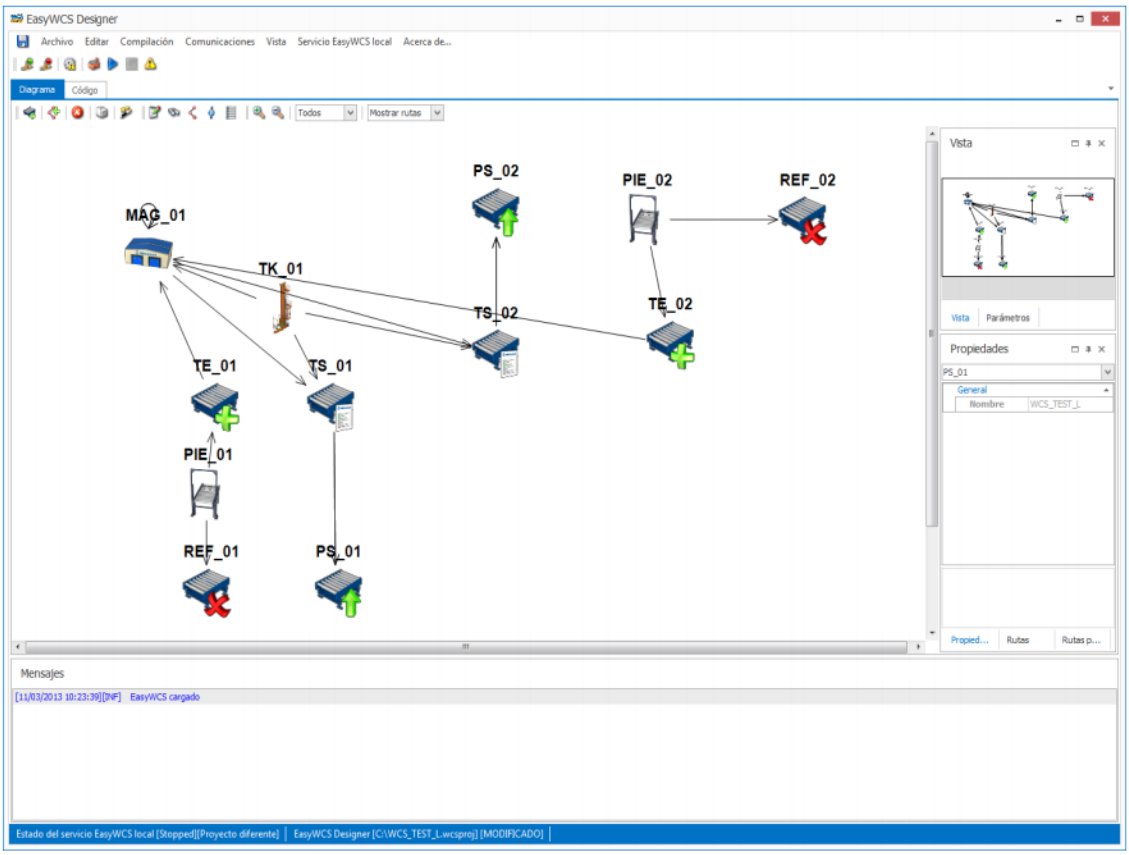
3.3.5.- CÓMO CERRAR UN PROYECTO

Haga clic en la opción Fichero → 

del menú principal para cerrar el proyecto.

3.4.- PANTALLA DIAGRAMA

En la pantalla principal **Diagrama** se visualizan fácilmente las rutas que están configuradas y todas las estaciones del almacén. Gráficamente se muestran los enlaces entre estaciones.

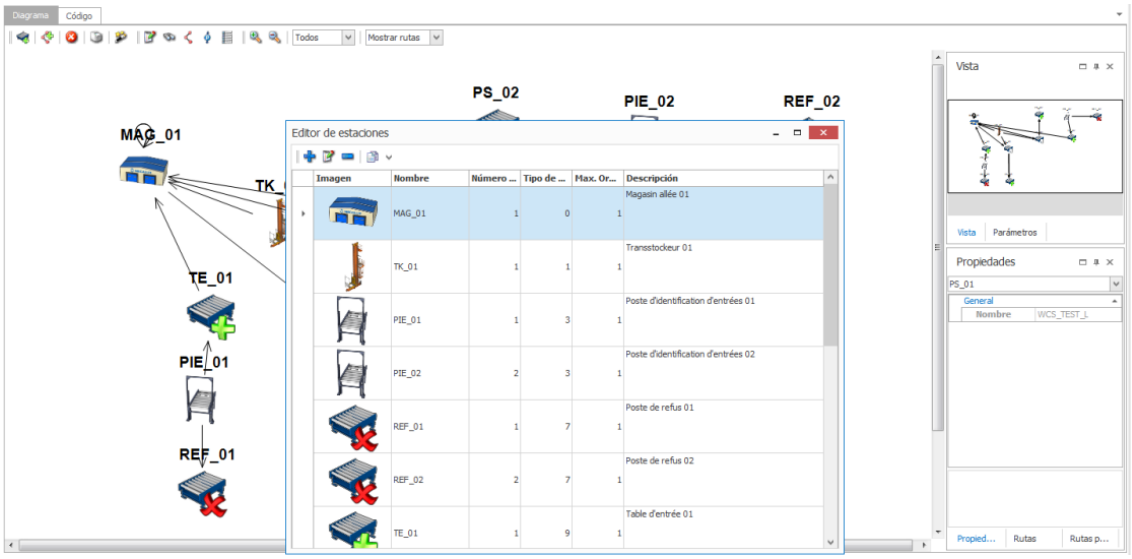


Uno de los primeros pasos en la realización del proyecto es partir de un proyecto importado desde XML (más información en "¿Cómo importar un nuevo proyecto?" en la página 12) o comenzar a crear las **Estaciones** existentes en la instalación.

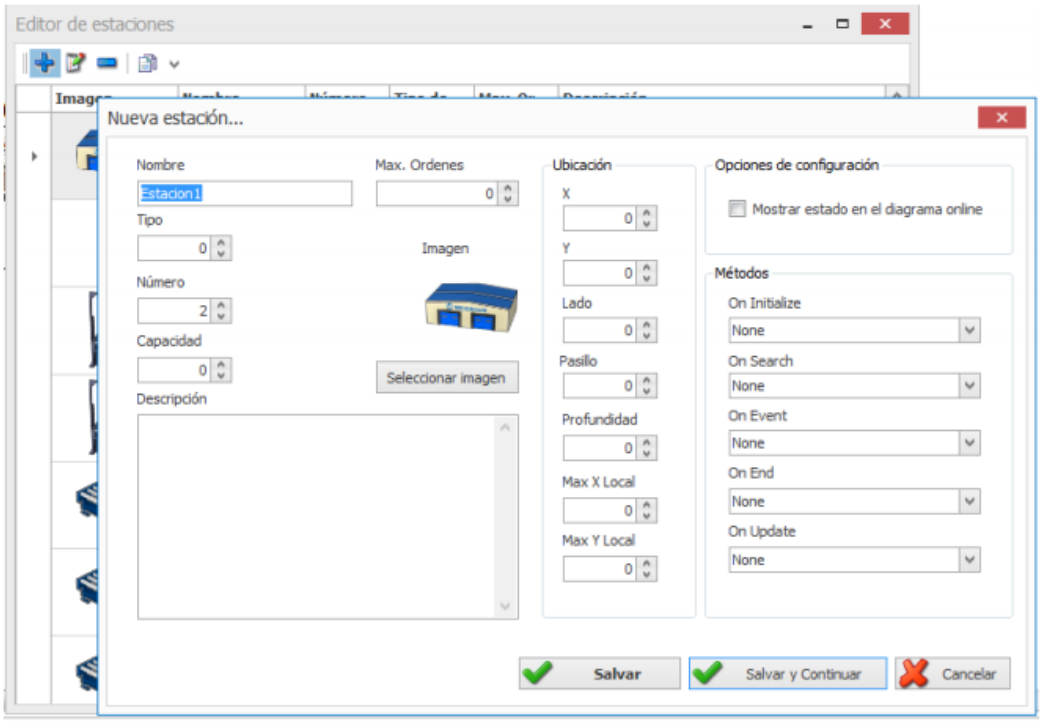
Recuerde, haga clic en situado en la barra de herramientas, para guardar las modificaciones realizadas en el proyecto. Es imprescindible reiniciar el servicio para que los cambios surtan efecto.

3.4.1.- CÓMO CREAR UNA ESTACIÓN

Haga clic en la opción Edición → Estaciones del menú principal.



En la ventana **Editor de Estaciones**, haga clic en para crear una estación.

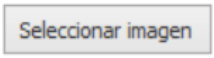


Tenga en cuenta que parte de esta información ha de reflejarse también en el WMS.

Los campos obligatorios son:

- **Nombre, Número, Tipo**: Para más información sobre las estaciones existentes en una instalación consulte el manual de Interface de Comunicaciones de Control.
- **Capacidad**: Número de trackings admisibles en la estación.
- **Máximo de Órdenes**: Número máximo de órdenes que pueden ser asignadas para una estación. El valor "0" indica que no hay número máximo.

Otros campos:

- **Imagen**: Es posible asociar una imagen a cada estación. Haga clic en 

y el Diseñador muestra el Gestor de Imágenes donde podrá visualizar, importar y eliminar las imágenes (en formato PNG).

Datos posicionales:

- **X, Y, Lado, Pasillo, Profundidad.**
- **X e Y local máxima**: Coordenadas X e Y Local relativa a la ubicación. Consulte en el manual de **Interface de Comunicaciones de Control** los valores predefinidos según el tipo de estación.

Opciones de configuración:

- **Mostrar estado en el diagrama online**: Seleccionado puede visualizar el campo estado. En las estaciones de Tipo 0 no está seleccionado por defecto, en el resto sí.

Métodos, es decir, funciones que se van a ejecutar en cada uno de los eventos configurables.

- **Inicialización, Búsqueda, Evento, Fin y Actualización de estación**.

  En ellos, seleccione en el desplegable el tipo de método a utilizar. Por defecto se muestra la opción "**Ninguno**" y el equivalente para cada operación del método **Estándar**, indicado éste entre [].

  La opción **estándar** implementa la lógica común para la operación seleccionada. Esta implementación puede consultarla, (más información en "¿**Cómo consultar el Código Estándar**?" en la página 31.

La implementación estándar podrá ser utilizada de varias maneras:

- Como implementación final de las operaciones a realizar.
- Como implementación a utilizar durante el proceso de puesta en marcha.
- Como implementación base a extender por el código custom de usuario:

  Si desea implementar otra lógica diferente, necesitará crear una implementación Custom (más información en "Pantalla "Código"" en la página 30) tras lo cual le aparecerá en el desplegable junto a las 2 opciones anteriores.

En la ventana **Edición de Estaciones**, haga clic en  para clonar una estación de las anteriormente creadas. La estación clonada mantiene todas las características definidas de la anterior.

 Recuerde, para más información sobre las estaciones existentes en una instalación consulte el manual de **Interface de Comunicaciones de Control.**

3.4.2.- CÓMO EDITAR UNA ESTACIÓN

Haga doble clic en la tabla Editor de Estaciones sobre la estación que desea editar o bien selecciónela y haga clic en  .

También puede seleccionar la estación en el desplegable del panel **Propiedades** (panel inferior derecho de la pantalla) y cambiar en el mismo sus propiedades.

 La pantalla **Diagrama** permite **Multiselección**. Puede cambiar las propiedades de varias estaciones seleccionándolas en el Diagrama y modificando los valores en el panel **Propiedades**. Observe que el desplegable permanece en blanco si existe Multiselección.

3.4.3.- CÓMO ELIMINAR UNA ESTACIÓN

Seleccione la estación en la tabla y haga clic en  para eliminarla.

También puede eliminar una estación seleccionándola en el diagrama y haciendo clic en  en la barra de herramientas o directamente con la tecla Supr del teclado o desde el menú contextual haciendo clic derecho en el ratón.

El Diseñador no permite eliminar una estación hasta que no se hayan eliminado previamente todas las rutas que pertenezcan a la misma.

Tras la creación de las estaciones comenzamos a crear el de estaciones de la Instalación.

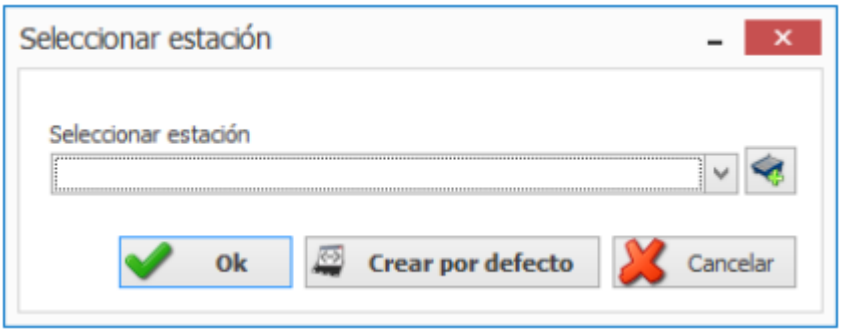
3.4.4.- CÓMO CREAR UN LAYOUT DE ESTACIONES

En una instalación pueden existir varios niveles en el layout.

Si no parte de un proyecto importado desde XML (más información en "¿Cómo importar un nuevo proyecto?" en la página 12) es necesario crear estos niveles.

Para ello añada una de las estaciones haciendo clic en en la barra de herramientas o bien, sobre el fondo de la pantalla, haga clic derecho y elija en el menú contextual esta opción.

Ubique en la pantalla la estación (la posición puede modificarse a posteriori). El Diseñador muestra la siguiente ventana:



En la ventana **Seleccionar Estación** elija mediante el desplegable una de las estaciones definidas con anterioridad. Si desea crear una estación nueva haga clic en .

A continuación, en el panel inferior derecho de la pantalla, **Propiedades**, modifique el valor "**Nivel**". A partir de este momento el nivel está creado.

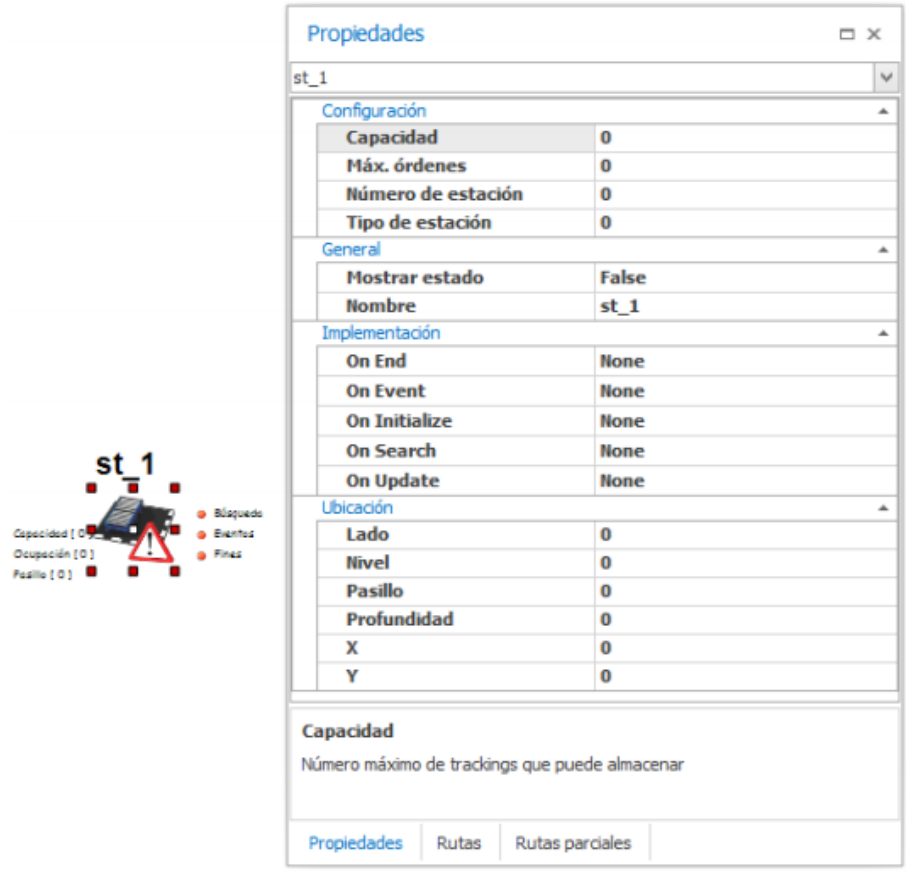
Si desea añadir más niveles repita el paso anterior. Si ya están todos los niveles definidos seleccione mediante el desplegable Todos el nivel del layout donde crear otra estación haciendo de nuevo haciendo clic en .

Por defecto, si tiene la opción "Todos" seleccionada, el nivel donde se crea la estación es el más bajo de los niveles definidos.

Haga clic en 

para que el Diseñador cree una estación por defecto.

Tanto para esta estación, como para el resto, puede cambiar sus propiedades en el panel inferior derecho de la pantalla, **Propiedades**, seleccionando la estación en el Diagrama o en el desplegable del mismo panel.

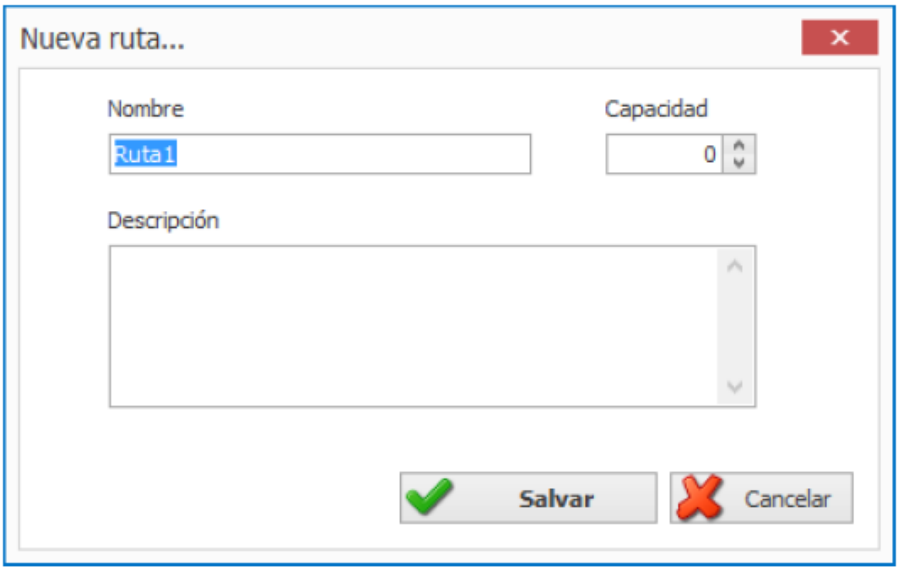


3.4.5.- CÓMO CREAR RUTAS ENTRE LAS ESTACIONES DEL LAYOUT

La creación de rutas puede realizarse desde el propio layout o mediante un formulario específico.

Para comenzar haga clic en  en la barra de herramientas o bien, sobre el fondo de la pantalla, haga clic derecho y elija en el menú contextual la opción "**Nueva Ruta**".

El Diseñador muestra la ventana:



Los campos disponibles son:

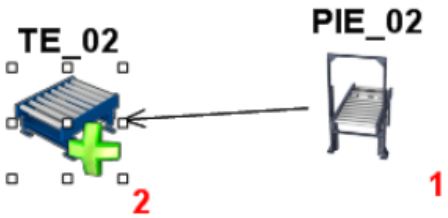
- **Nombre.**
- **Capacidad**: Número de contenedores que es capaz de gestionar la ruta. Este parámetro está relacionado a la limitación física de la ruta.

  Este campo es meramente informativo. El parámetro que tiene efecto en la gestión de EasyWCS es el referente a la capacidad de las rutas parciales (más información en "¿Qué son las rutas parciales?" en la página 28)

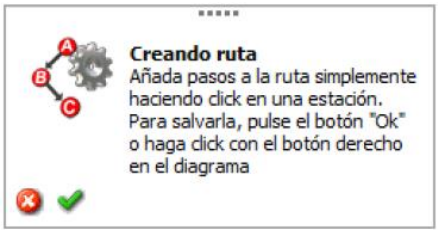
- **Descripción.**

A continuación, sobre el cursor del ratón, el Diseñador muestra el icono .

Haga clic sobre la estación origen de la ruta. Así tantas veces como estaciones quiera incluir en la ruta. Mediante un número le indicará el orden en la misma.

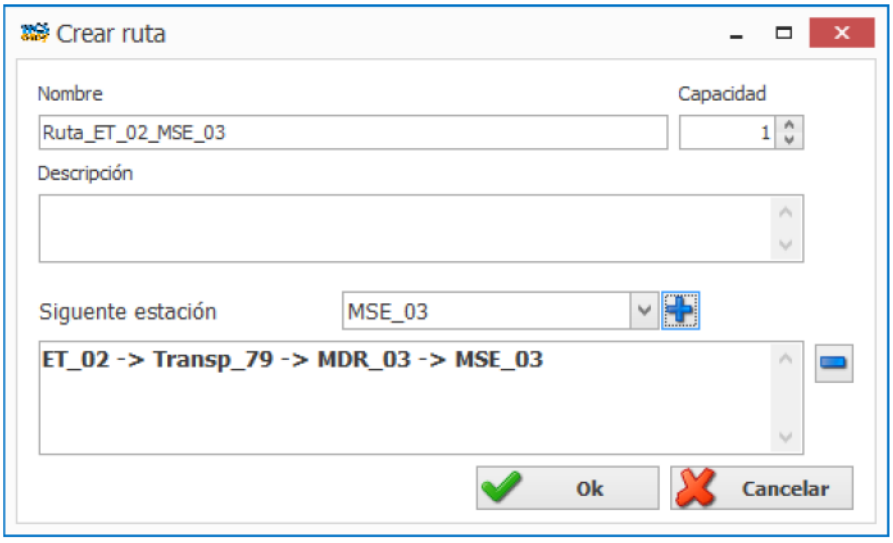


Para finalizar la creación de la ruta, siga las indicaciones de la siguiente captura:



También es posible crear rutas desde un formulario. Para ello haga clic derecho sobre el fondo de la pantalla y elija en el menú contextual la opción "**Crear ruta desde formulario**".

El Diseñador muestra la ventana:



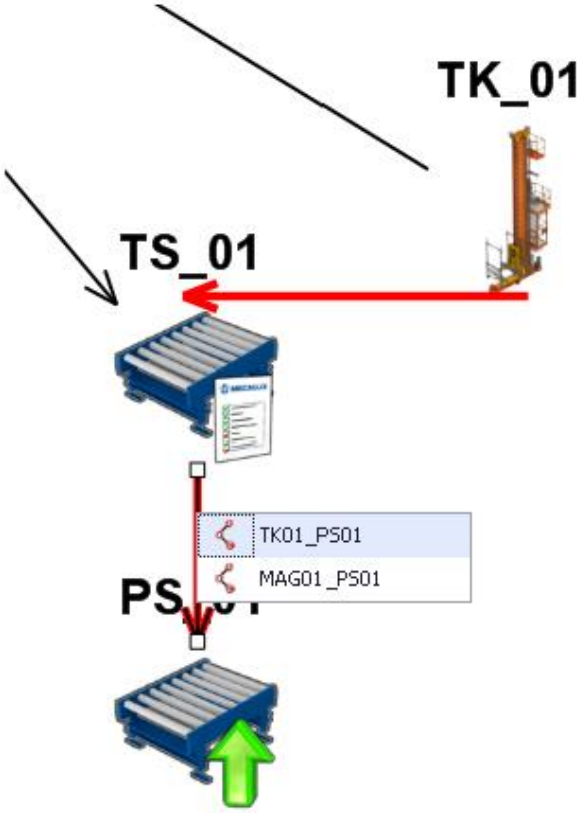
Los campos disponibles son:

- **Nombre**, **Capacidad**, **Descripción.**

- **Siguiente estación**: Añada mediante el icono  la estación seleccionada en el desplegable. En el panel inferior se detallan las estaciones añadidas a la ruta. Para eliminar la última estación añadida haga clic en  .

En un diagrama creado con varias rutas, entre dos estaciones puede existir gráficamente un enlace pero, en realidad, este enlace pertenece a varias rutas.

Para distinguir estos casos haga clic derecho sobre el enlace. Como puede observar en la imagen inferior, el Diseñador le indica las rutas a las que pertenece el enlace. Si hace doble clic sobre una de ellas todos los enlaces de esa ruta son mostrados en rojo.



En el desplegable 

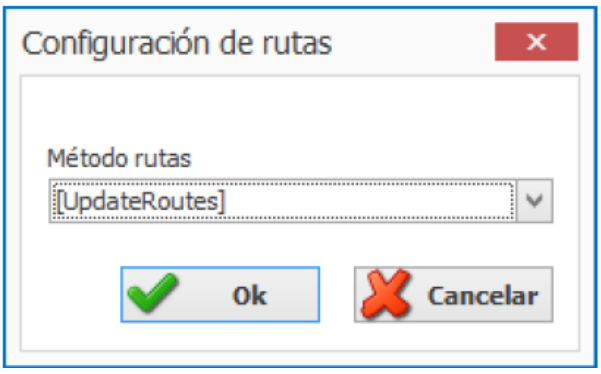
puede seleccionar que se muestren o no las rutas en la pantalla Diagrama.

Seleccionando un nivel en el desplegable 

, el Diseñador muestra las rutas de ese nivel, no las rutas que existan entre varios niveles.

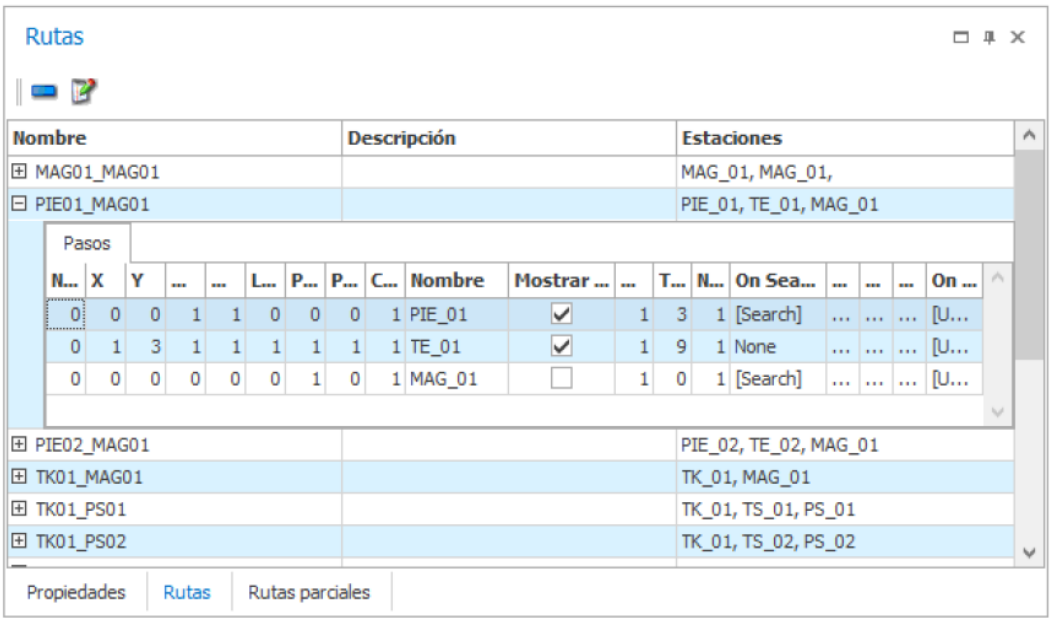
3.4.6.- ¿CÓMO ASOCIAR UN MÉTODO A LAS RUTAS CREADAS?

De forma similar a lo explicado en los métodos de las estaciones, para las rutas es posible asociar una función común para todas ellas.



3.4.7.- ¿CÓMO EDITAR UNA RUTA?

En el panel inferior derecho de la pantalla principal, haga clic sobre la pestaña **Rutas.**



Seleccione la ruta en la tabla y haga clic en  . Haga doble clic sobre una ruta y ésta se mostrará en rojo en el Diagrama. Esta tabla le ofrece varias opciones similares a las de la pantalla “Tablas”, más información en “Herramientas de selección y filtrado” en la página 44.

3.4.8.- ¿CÓMO ELIMINAR UNA RUTA?

En el panel inferior derecho de la pantalla principal, haga clic sobre la pestaña **Rutas.**

Seleccione la ruta en la tabla y haga clic en  para eliminarla.

3.4.9.- ¿QUÉ SON LAS RUTAS PARCIALES?

Las rutas parciales son tramos entre dos estaciones y son generadas automáticamente cuando definimos las Rutas completas en el diagrama (más información en "¿Cómo crear rutas entre las estaciones del layout?" en la página 24).

En el panel inferior derecho de la pantalla principal, haga clic sobre la pestaña **Rutas Parciales.**

Rutas parciales

Origen	Destino	Capacidad	Estado por defecto
MAG_01	MAG_01	0	Fallo
MAG_01	TS_01	0	Fallo
MAG_01	TS_02	0	Fallo
TK_01	MAG_01	0	Fallo
TK_01	TS_01	0	Fallo
TK_01	TS_02	0	Fallo
PIE_01	REF_01	0	Fallo
PIE_01	TE_01	0	Fallo
PIE_02	REF_02	0	Fallo
PIE_02	TE_02	0	Fallo
TE_01	MAG_01	0	Fallo
TE_02	MAG_01	0	Fallo
TS_01	PS_01	0	Fallo
TS_02	PS_02	0	Fallo

En esta tabla se muestran los tramos (rutas parciales) que componen una Ruta.

Esta tabla le ofrece varias opciones similares a las de la pantalla "Tablas", más información en "Herramientas de selección y filtrado" en la página 44. Las columnas mostradas son:

- **Origen:** estación inicio del tramo. Campo no editable y autogenerado en la definición de las Rutas.
- **Destino**: estación final del tramo. Campo no editable y autogenerado en la definición de las Rutas.

- **Capacidad**: Número de contenedores que es capaz de gestionar esa ruta parcial. Este parámetro está relacionado a la limitación física del tramo.

  Con el valor "0", EasyWCS no tiene en cuenta la capacidad de las rutas a la hora de seleccionar órdenes.

- **Estado por defecto**: permite definir un estado inicial de la ruta (Fallo, En servicio, Llena y en fallo, Llena y en servicio).

  Dicho estado prevalece hasta que sea actualizado por el TMS.

3.4.10.- PARÁMETROS DEL DIAGRAMA

En el panel superior derecho de la pantalla principal, haga clic sobre la pestaña **Parámetros.**

WMS comunica al servicio EasyWCS los parámetros que considere oportuno mediante la "Tabla Parámetros (WCS_CONFIG_PARAMETERS)" en la página 54. La finalidad de la tabla mostrada en la pantalla Diagrama es indicar estos parámetros y su valor por defecto.

Para añadir un parámetro haga clic en y rellene sobre la tabla el campo **Nombre** y el campo **Valor**. Para eliminar uno selecciónelo y haga clic en .

3.4.11.- OTRAS OPCIONES DE LA PANTALLA DIAGRAMA

El Diseñador incluye, en la barra de herramientas, opciones como imprimir, zoom, reorganizar las líneas de rutas, etc.

La pantalla **Diagrama** muestra información adicional cuando se encuentra en modo Online. Puede obtener más información en "¿Qué podemos ejecutar en el Modo Online?" en la página 34.

3.5.- PANTALLA CÓDIGO

La pantalla **Código** ofrece al usuario la posibilidad de crear y editar métodos que se asociarán a las estaciones del diagrama.

Cada una de las estaciones tiene asociada una operación genérica que realiza las siguientes operaciones básicas: Inicialización, búsqueda de orden, evento, fin de orden, actualización de estación, actualización de ruta.

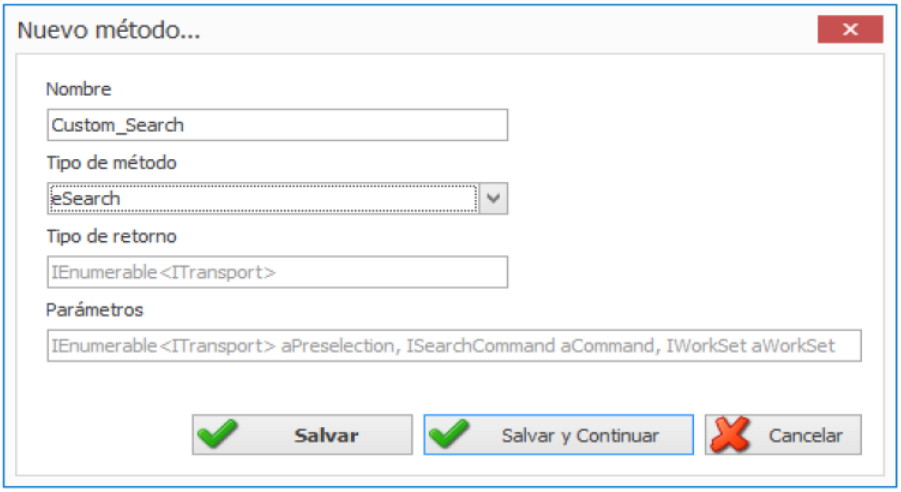
Sin embargo, esto no es suficiente en algunos casos y puede requerir de código adicional. Para ello se permite asociar a cada operación una función custom (programada en C#) que serán ejecutadas durante la operación genérica.

Por ejemplo, durante una búsqueda de orden, este código será el encargado de seleccionar, de entre las órdenes generadas, cuáles son las seleccionadas para transferir al TMS.

Recuerde, haga clic en situado en la barra de herramientas, para guardar las modificaciones realizadas en el proyecto. Es imprescindible reiniciar el servicio para que los cambios surtan efecto.

3.5.1.- ¿CÓMO CREAR UN MÉTODO?

Haga clic en en la barra de herramientas. El Diseñador muestra la ventana **Nuevo Método**:



Los campos disponibles son:

- **Nombre.**
- **Tipo de método**: Seleccione mediante el desplegable una de las operaciones predefinidas para las estaciones: Inicialización, búsqueda de orden, evento, fin de orden, actualización de estación, actualización de ruta.

- **Tipo de Retorno**: retorno en función del tipo de método seleccionado (no editable).
- **Parámetros**: definición de los parámetros en función del tipo de método seleccionado (no editable).

EasyWCS incorpora un potente editor de código que permite realizar las tareas de programación. Incluye un resaltado de sintaxis para código C#.

3.5.2.- ¿CÓMO EDITAR UN MÉTODO?

Haga clic en  en la barra de herramientas. El Diseñador muestra la ventana **Buscador de Método**. Seleccione el método y haga clic en  en la barra de herramientas.

3.5.3.- ¿CÓMO ELIMINAR UN MÉTODO?

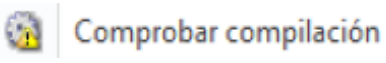
Haga clic en  en la barra de herramientas. El Diseñador muestra la ventana **Buscador de Método**. Seleccione el método a eliminar y haga clic en  .

3.5.4.- ¿CÓMO CONSULTAR EL CÓDIGO ESTÁNDAR?

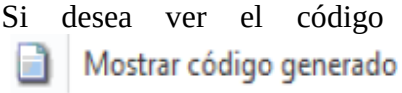
Haga clic en  en la barra de herramientas. El Diseñador muestra la ventana **Code** donde puede consultar el código estándar de las operaciones.

3.5.5.- COMPILACIÓN DEL CÓDIGO

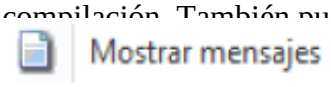
Para diagnosticar errores en el código realice una compilación del mismo.

Haga clic la opción Compilación > 

o directamente en el icono  de la barra de herramientas.

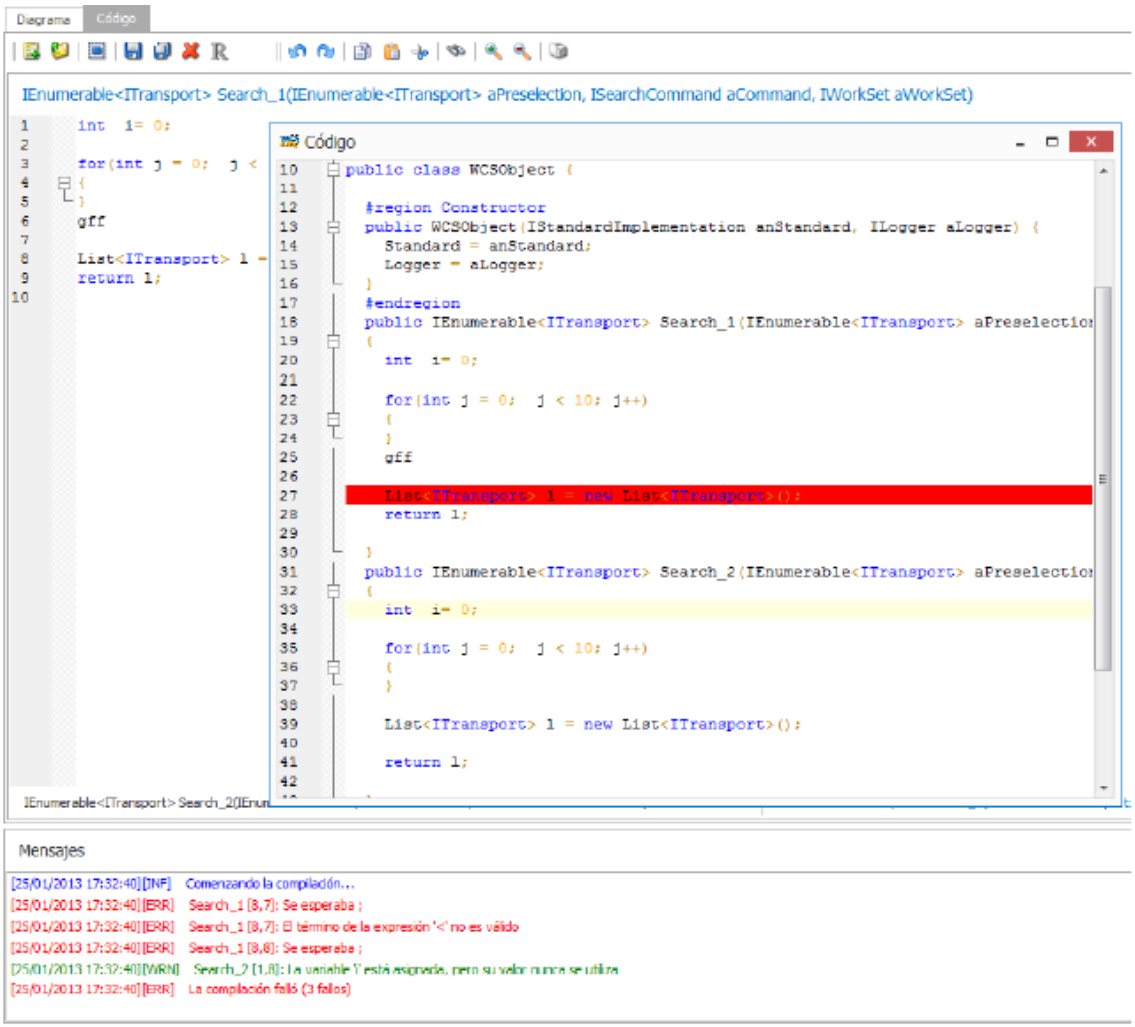
Si desea ver el código generado haga clic la opción Compilación > 

.

En el panel inferior "**Mensajes**" se muestra automáticamente el resultado de la compilación. También puede mostrar de forma manual este panel haciendo clic en Ver > 

en la barra de herramientas.

Si el proceso de compilación termina con error se muestra el motivo en el panel anterior indicado con un breve texto. Haga clic sobre el error o warning para localizar el código donde se produjo el error.



3.5.6.- **OTRAS OPCIONES DE LA PANTALLA CÓDIGO**

El Diseñador incluye, en la barra de herramientas, opciones como cortar, pegar, deshacer/rehacer, buscar texto, imprimir, etc.

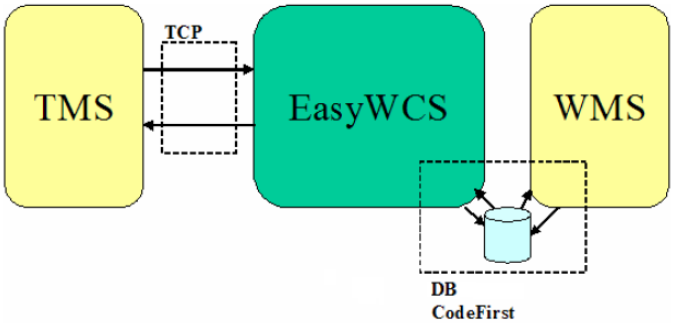
4. CAPITULO 4: COMUNICACIONES

En este capítulo detallamos de forma general las comunicaciones existentes en una instalación que utilice EasyWCS y las funcionalidades que ofrece el Diseñador cuando está comunicando con el servicio EasyWCS.

4.1.- COMUNICACIONES EN EASYWCS

En las comunicaciones se pueden diferenciar dos partes:

- **Comunicaciones TMS - EasyWCS**: Estas comunicaciones se realizan sobre una red TCP/IP utilizando el protocolo de EasyWMS Gateway (más información sobre este protocolo consulte la *Especificación del protocolo de comunicaciones EasyWMS Gateway*).
- **Comunicaciones WMS - EasyWCS**: Estas comunicaciones se realizan utilizando una interface de tablas de base de datos. Esta base de datos puede ser tanto la propia del WMS (modificada para que incorpore estas tablas) o una base de datos independiente que actúe como puente.



Según esto, el funcionamiento básico del sistema es el siguiente:

1. EasyWCS atiende las **peticiones** que le lleguen desde el **TMS**. Cada petición que requiera pasar alguna información al WMS lo hará mediante inserciones en la base de datos.
2. EasyWCS sondea la base de datos en busca de **peticiones** desde el **WMS**. Cada petición que requiera informar al TMS lo hará enviándole el mensaje adecuado.

4.2.- MODO ONLINE

4.2.1.- CONSIDERACIONES PREVIAS

Para comunicar el Diseñador con el servicio de EasyWCS es necesario:

- Diseñador: indique el nombre del PC y puertos del servicio EasyWCS (consulte "¿Cómo editar las propiedades de un proyecto?" en la página 17).
- Servicio EasyWCS: realice las configuraciones detalladas en "Herramienta de configuración" en la página 58.

4.2.2.- ¿CÓMO EJECUTAR EL MODO ONLINE?

Para iniciar la comunicación de un proyecto entre el Diseñador y el servicio EasyWCS, haga clic en  o bien desde el menú principal Comunicaciones > Modo Online.

Para terminar la comunicación de un proyecto, haga clic en  o bien desde el menú principal Comunicaciones > Modo Offline.

Cuando la comunicación se establece entre el Diseñador y el servicio muestra el mensaje EasyWCS está online.

4.2.3.- ¿QUÉ PODEMOS EJECUTAR EN EL MODO ONLINE?

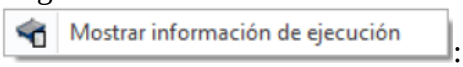
4.2.3.1.- INFORMACIÓN DE LA ESTACIÓN

La pantalla **Diagrama** muestra información adicional para cada una de las estaciones:

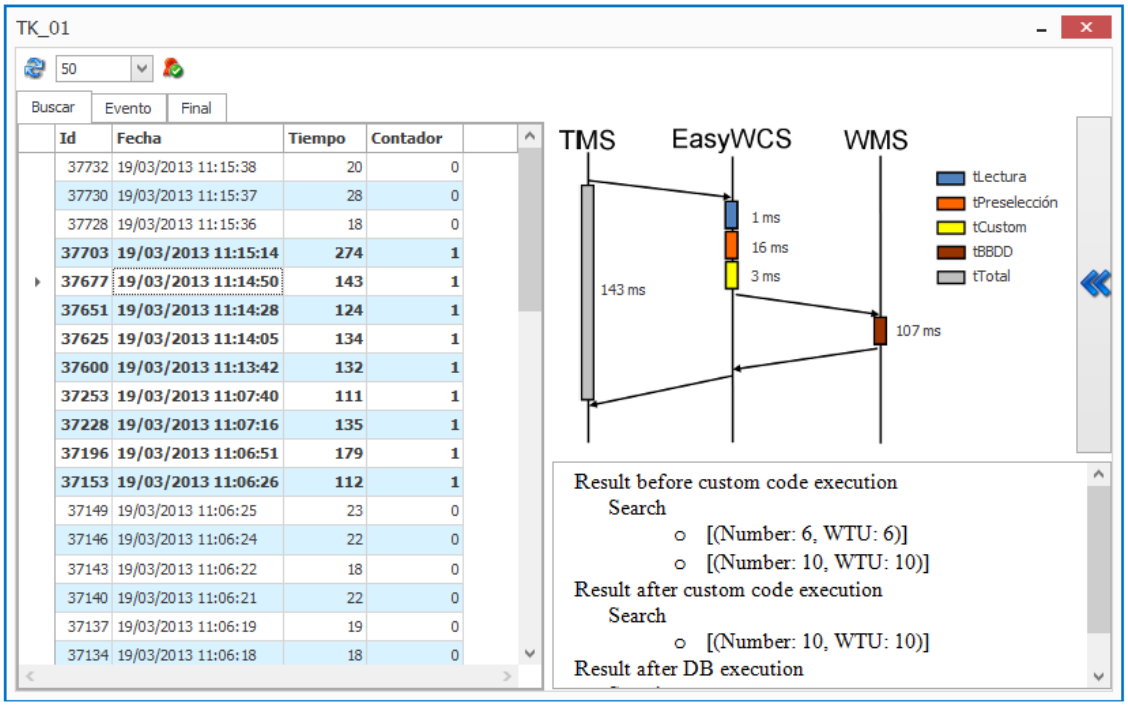
- **Estado**: último estado notificado desde el TMS.
- **Presecia**: última notificación desde el TMS de si la estación se encuantra vacía o llena.
- **Capacidad**: última capacidad de la de estación notificada desde el TMS.

- **Ocupación**: última ocupación de la estación notificada desde el TMS.
- **Búsqueda de orden**: El TMS está notificando al servicio EasyWCS una búsqueda.
- **Evento**: El TMS está notificando al servicio EasyWCS un evento.
- **Fin de orden**: El TMS está notificando al servicio EasyWCS un fin de orden.

Recuerde, para más información sobre las estaciones existentes en una instalación consulte el manual ***Interface de Comunicaciones de Control***.

Haga clic derecho en el ratón sobre una estación y seleccione 

:



La ventana ofrece información acerca de las funciones de comunicación realizadas en dicha estación. Muestra tantas operaciones como tengamos configuradas en el desplegable 

:

- **Id**: identificador global a todas las operaciones.
- **Fecha**
- **Tiempo empleado**: haga doble clic sobre la operación para desplegar un panel gráfico con los tiempos relativos empleados en la operación.

- **tLectura**: Tiempo de lectura del mensaje enviado desde el TMS al servicio.
- **tPreselección**: Tiempo empleado por el servicio en adquirir del WMS las órdenes candidatas.
- **tCustom**: Tiempo empleado por el servicio para seleccionar la orden en función de las condiciones implementadas.
- **tBBDD**: Tiempo empleado en base de datos.
- **tTotal**: Tiempo total.

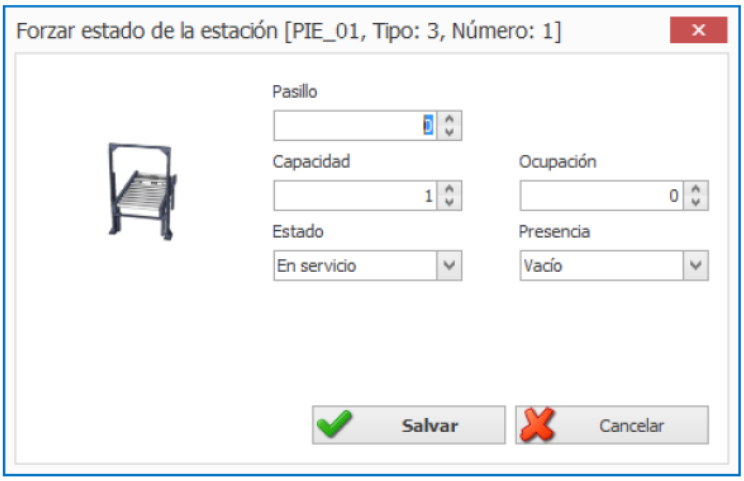
  En la parte inferior se detalla, por ejemplo para una búsqueda:
  - Resultado de la operación **antes** de ejecutar el código estándar/custom.
    En el ejemplo, existen 2 órdenes candidatas.
  - Resultado de la operación **después** de ejecutar el código estándar/custom.
    En el ejemplo, tras ejecutar la implementación estándar de la búsqueda, el servicio selecciona solo una de las candidatas. En nuestro caso, era prioritario ejecutar la orden de más profundidad para una misma localización por lo que fue esa la orden seleccionada.
    - Resultado de la operación en BBDD.

- **Contador**: Indica el número de trackings devueltos por el WMS en las búsquedas de orden. Si el valor es distinto de 0 la línea de la operación se muestra en negrita. Para mostrar solo los valores con resultado haga clic en  .

- **Error**: La línea de la operación se muestra en rojo si se ha producido un error en la operación debido a tener datos incorrectos en la misma.

4.2.3.2.- FORZAR VALORES DE UNA ESTACIÓN

Haga clic derecho en el ratón sobre una estación y seleccione  Forzar estado de la estación.



La ventana ofrece la posibilidad de realizar un forzado de varios valores de la estación como el **Pasillo**, la **Capacidad** y **Ocupación**, el **Estado** (En servicio o Fallo) y por último la **Presecia** (Vacío o Lleno).

Dicho forzado prevalece hasta que sea actualizado por el TMS.

4.2.3.3.- **FORZAR VALORES DE UNA RUTA PARCIAL**

La pantalla **Diagrama** muestra también información adicional para cada una de las **Rutas Parciales.** En el panel inferior derecho de la pantalla principal, haga clic sobre la pestaña Rutas Parciales. En el modo online el Diseñador muestra dos nuevas columnas.

- **Ocupación**: Número de contenedores que se encuentran en esa ruta parcial. Este campo es indicado desde el TMS.

  En la actualidad no es posible, mediante Galileo, indicar este parámetro ya que para la actualización de rutas solo se indica el Estado de la misma (Fallo, En servicio, Llena y en fallo, Llena y en servicio).

- **Estado**:
  - **Desconocido**: el servicio no tiene información sobre el estado.
  - **Fallo**: la ruta parcial está en defecto.
  - **En servicio:** la ruta parcial no está en defecto y no ha completado su capacidad máxima definida.
  - **Llena y en defecto:** la ruta parcial ha completado su capacidad y está en fallo.
  - **Llena y en servicio:** la ruta parcial no ha completado su capacidad máxima definida y no está en fallo.

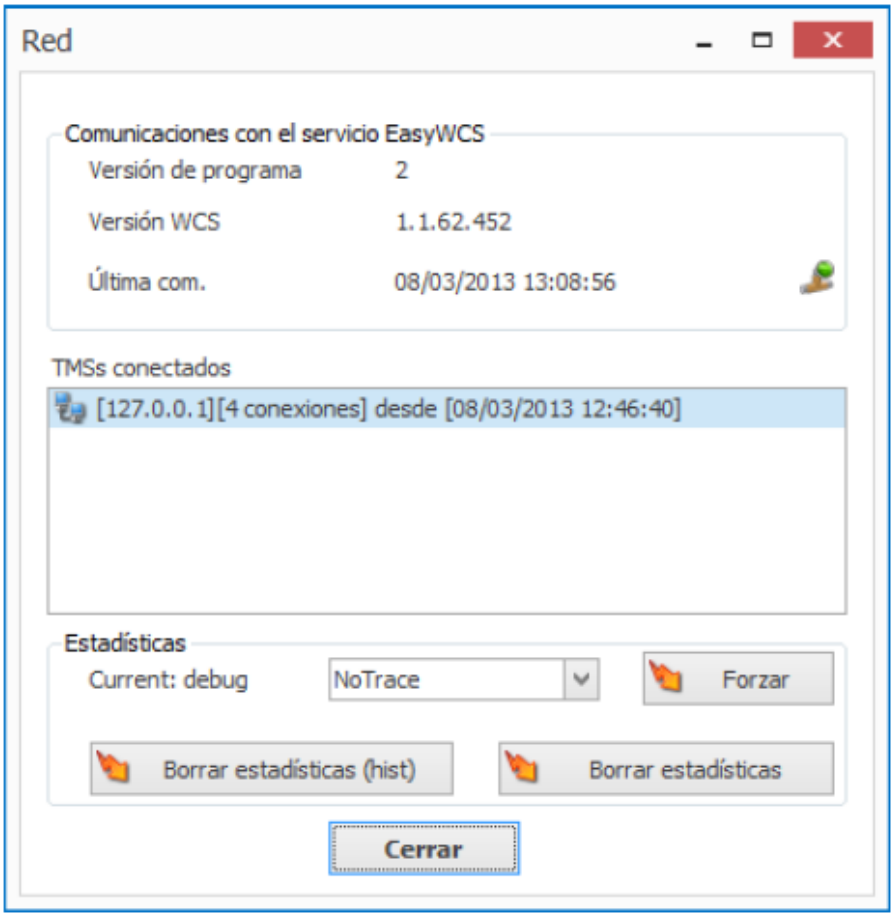
4.2.3.4.- **OTRAS PANTALLAS**

El Diseñador muestra nuevas pantallas cuando se encuentra en modo online:

- **Simulación de Órdenes**: Permite crear una orden de forma virtual pero perfectamente funcional para comunicar tanto con el TMS como con el servicio de EasyWCS. Más información en “Pantalla “Simulación de órdenes”” en la página 41.
- **Tablas**: establece una serie de tablas que muestran información acerca de las comunicaciones y mensajería entre cualquier WMS con un TMS a través de EasyWCS. Más información en “Pantalla “Tablas”” en la página 43.

4.2.4.- **VER RED**

Opción accesible desde el menú principal **Comunicaciones** >  Red .



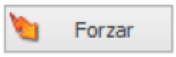
La información disponible es:

- **Comunicaciones con el servicio EasyWCS**: Informa sobre la versión del Diseñador, versión del Servicio y fecha de la última comunicación entre ambos.

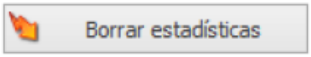
Si existe comunicación entre el Diseñador y el servicio muestra el icono  , en caso contrario muestra  .

- **TMSs conectados**: Informa, para cada TMS, desde que IP se conecta, número de conexiones desde el arranque del servicio y última comunicación entre ambos.

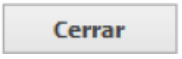
 En el panel inferior "**Mensajes**" se muestra automáticamente la conexión o desconexión al servicio de EasyWCS de cualquier TMS.

- **Estadísticas**: Información visible en "Tabla Estadísticas" en la página 56.
  - **Actual**: Nivel de traza **actual** seleccionada.
    Seleccione mediante el desplegable el nivel de traza de las estadísticas. Según el nivel de traza la información en el log es:
    No Trace: No se traza ninguna información
    Error: Se traza solo errores.
    Info: Se trazan errores y mensajes de información acerca de las operaciones.
    Debug: Se traza lo anterior y mensajes de depuración.
  - 

    : Aplica el nivel de traza seleccionado.
  - 

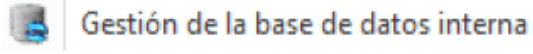
    : Borra toda la información de la tabla Statistics.
  - 

    : Borra toda la información de la tabla Statistics.

Para cerrar la ventana haga clic en 

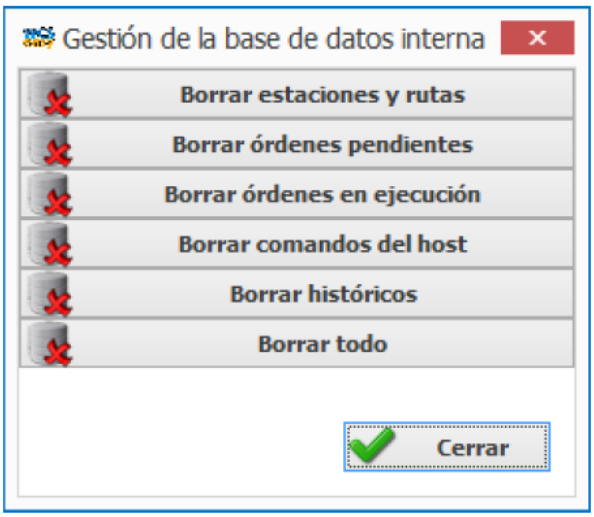
.

4.2.5.- GESTIÓN DE LA BBDD INTERNA

Opción accesible desde el menú principal **Comunicaciones** > 

.

EasyWCS puede utilizar una BBDD interna destinada a pruebas en puesta en marcha.



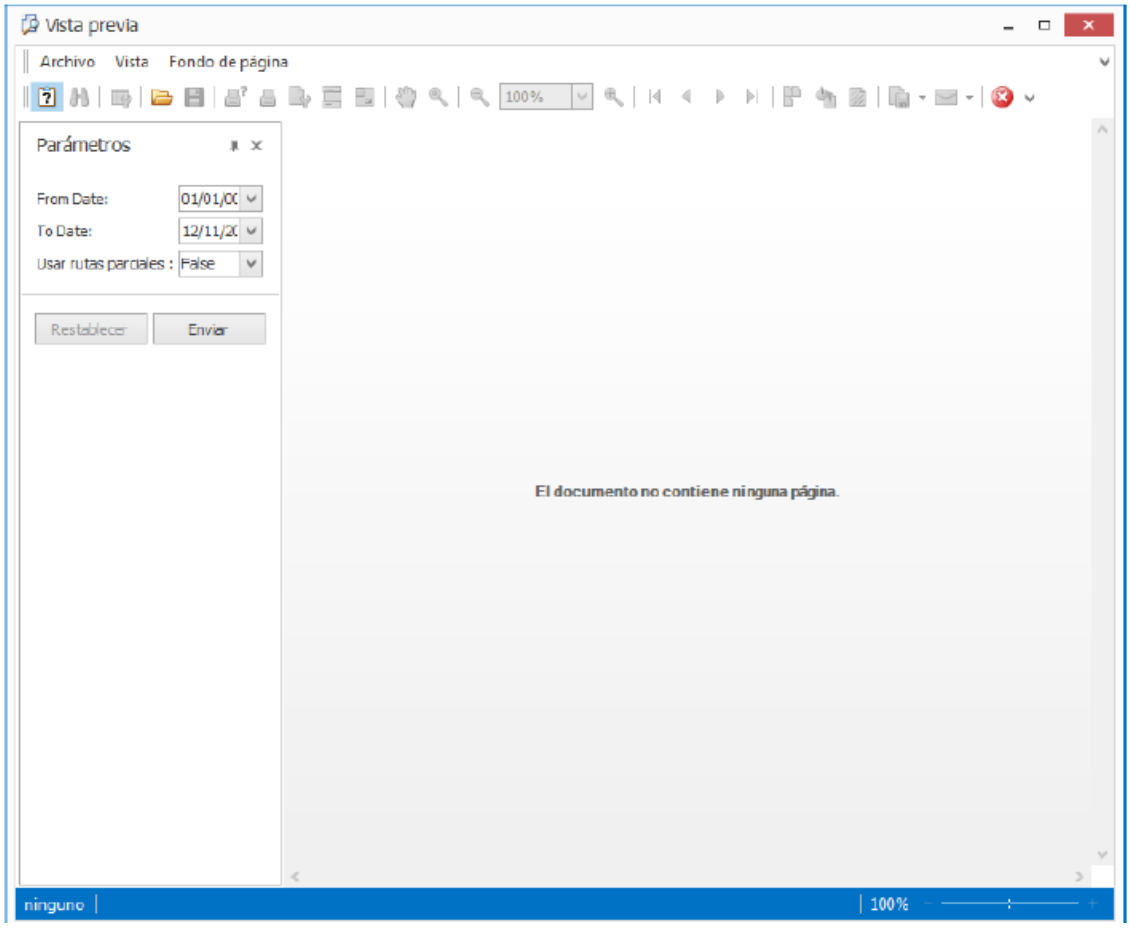
Desde esta ventana puede gestionar la BBDD interna. Le ofrece el borrado total de la misma o el borrado independiente de la información de cada una de las tablas.

4.3.- **GENERADOR DE INFORMES**

EasyWCS ofrece la posibilidad de generar un informe con distintos datos obtenidos del servicio. Este informe no deja de ser un resumen "más amigable" de los principales datos u operaciones disponibles en las Tablas.

Para generar un informe el Diseñador ha de estar en modo online (más información en "¿Cómo ejecutar el Modo Online?" en la página 34).

Para generar un informe haga clic en Archivo > Generar informe. EasyWCS muesta la siguiente ventana:



1. Seleccione el intervalo de fechas en los que se generará el informe.
2. Seleccione si desea o no que se generen en el informe las rutas parciales.
3. Haga clic en Enviar.

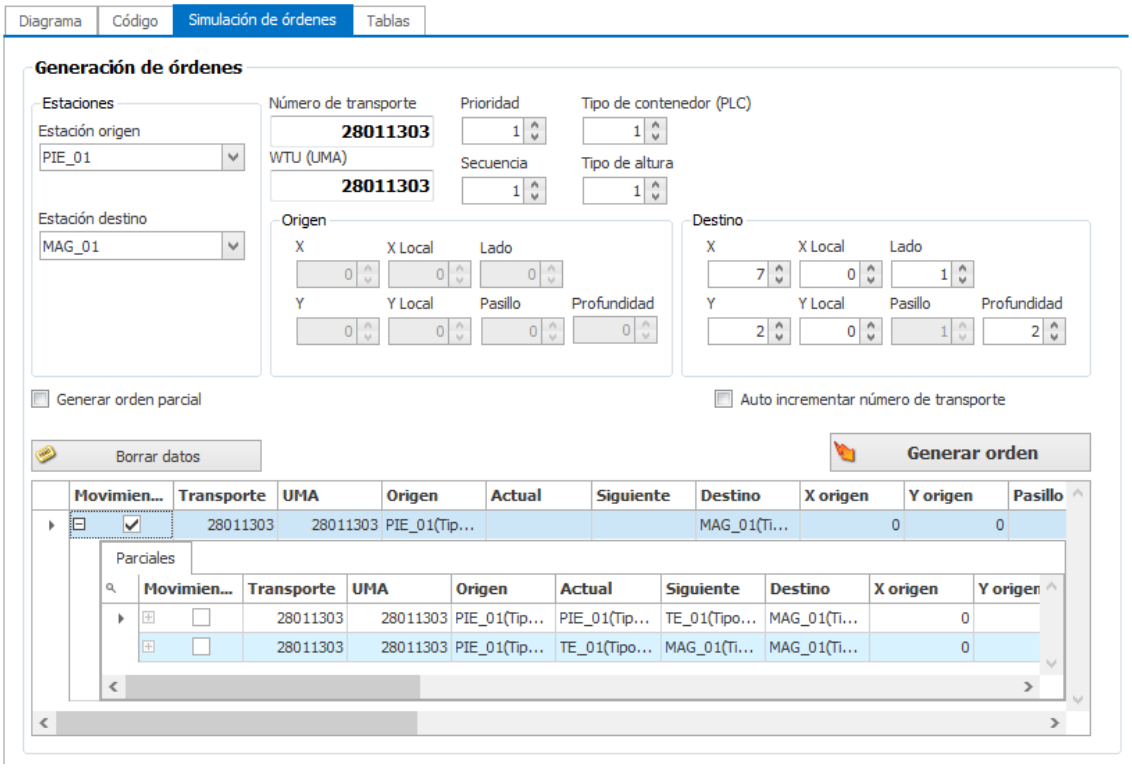
En el caso de que EasyWCS sea utilizado para realizar un test de las comunicaciones de EasyWMS®, las rutas parciales son las mismas que las rutas completas por lo que recomendamos no seleccionar el checK para evitar duplicidades en los datos del informe.

4.4.- PANTALLA "SIMULACIÓN DE ÓRDENES"

La pantalla **Simulación de Órdenes** es accesible solo si está el modo Online activado. Para activar el modo Online consulte "¿Cómo ejecutar el Modo Online?" en la página 34.

Una de las principales ventajas de EasyWCS es que el usuario implementador del TMS pueda realizar un test básico de las comunicaciones que existirán a posteriori en la instalación sin contar con recursos del WMS.

Esta pantalla facilita la creación de órdenes.



4.4.1.- ¿CÓMO GENERAR UNA ORDEN?

Desde esta pantalla puede generar órdenes completas o parciales. A continuación detallamos la operativa para ambas.

4.4.1.1.- ÓRDENES COMPLETAS

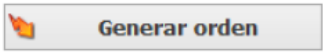
4.4.1.1.1.- ¿QUÉ ES UNA ORDEN COMPLETA?

Una orden completa es aquella orden cuyo origen y destino corresponde al definido en una de las Rutas configuradas en el Diagrama.

4.4.1.1.2.- ¿CÓMO GENERAR UNA ORDEN COMPLETA?

Complete los campos habilitados que aparecen en la pantalla.

Para más información sobre los campos consulte el manual de ***Interface de Comunicaciones de Control*** en el apartado *Estructura del Tracking (Galileo IV)*.

Haga clic en 

para crearla.

En la parte inferior de la pantalla, como puede observar en la captura superior, el Diseñador muestra una tabla con los datos de las órdenes generadas.

Cuando la columna **Movimiento Completo** muestra un check, haga clic en y el Diseñador le muestra las órdenes parciales existentes para esa orden. Estas órdenes parciales son generadas automáticamente.

Si desea consultar la orden generada puede hacerlo en la "Tabla Pendientes (local) (WCS_MSG_TR_LOCAL)" en la página 49.

4.4.1.2.- ÓRDENES PARCIALES

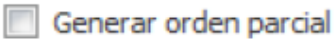
4.4.1.2.1.- ¿QUÉ ES UNA ORDEN PARCIAL?

Una orden parcial es aquella orden cuyo origen y destino corresponde al definido en una de las rutas parciales generadas automáticamente por el Diseñador.

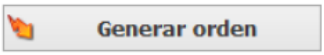
4.4.1.2.2.- CÓMO GENERAR UNA ORDEN PARCIAL

El Diseñador ofrece dos opciones:

- **Partiendo de una orden completa**:
  Tras generar una orden completa como se detalló anteriormente, haga clic sobre cualquiera de las órdenes parciales mostradas en la tabla inferior. Automáticamente se actualizan todos los campos.
- **Creando la orden parcial desde cero**:

  Haga clic en 

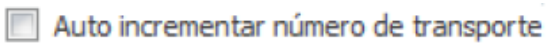
. Complete todos los campos que aparecen en la pantalla de la misma forma que la orden completa, añadiendo además la estación desde la que se inicia la orden parcial.

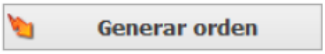
Finalmente, en ambos casos, haga clic en 

para crear la orden parcial.

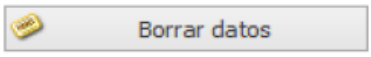
Si desea consultar la orden parcial generada puede hacerlo en la "Tabla Pendientes (local) (WCS_MSG_TR_LOCAL)" en la página 49.

4.4.2.- **OTRAS OPCIONES**

Haga clic en 

para que automáticamente incremente el número de transporte y el WTU (UMA) en una unidad cada vez que haga clic en 

.

Haga clic en 

para eliminar todas las órdenes de la tabla inferior. Si por el contrario solo quiere eliminar registros de forma individual, haga clic derecho en el ratón sobre cada registro y seleccione la opción desde el menú contextual.

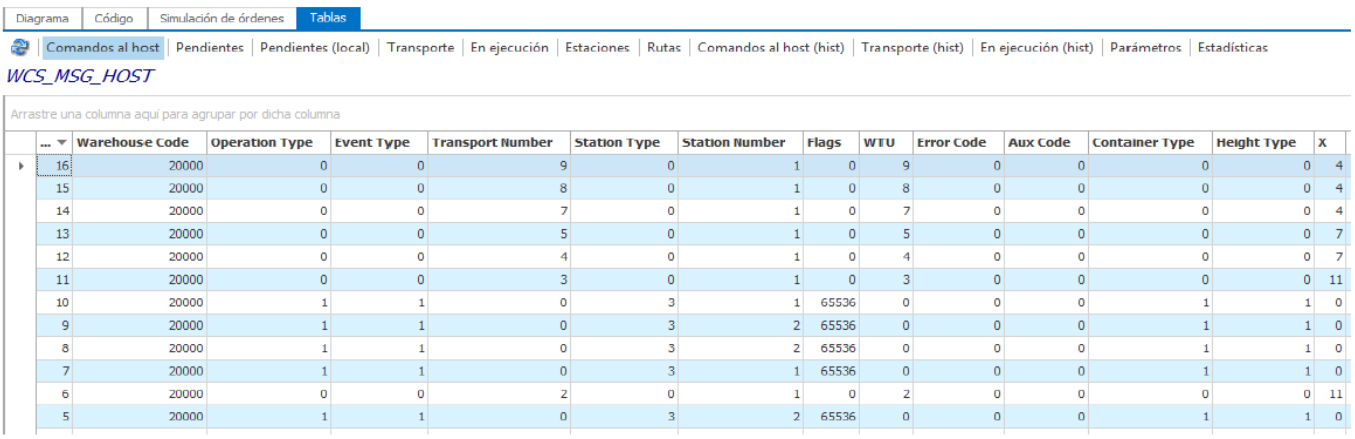
4.5.- **PANTALLA "TABLAS"**

La pantalla **Tablas** es accesible solo si está el modo Online activado. Para activar el modo Online consulte "¿Cómo ejecutar el Modo Online?" en la página 34.

Esta pantalla establece una serie de tablas que muestran información acerca de las comunicaciones y mensajería entre cualquier WMS con un TMS a través del servicio EasyWCS.

Las tablas se dividen en grupos:

- Comunicación **hacia el WMS**: permite traspasar datos al WMS > Tabla Comandos al Host.
- Comunicación **desde el WMS**: permite obtener datos desde el WMS > Tabla Pendientes.
- Tablas de **Configuración**: reflejan definiciones de estaciones, rutas y parámetros.
- Tablas de **Históricos** y **Estadísticas**: reflejan un registro de las distintas operaciones realizadas.

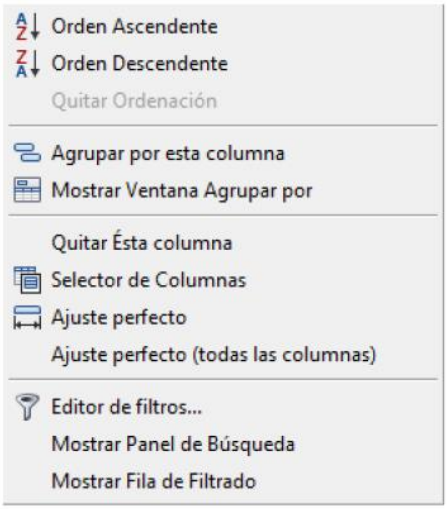


...	Warehouse Code	Operation Type	Event Type	Transport Number	Station Type	Station Number	Flags	WTU	Error Code	Aux Code	Container Type	Height Type	X
16	20000	0	0	9	0	1	0	9	0	0	0	0	4
15	20000	0	0	8	0	1	0	8	0	0	0	0	4
14	20000	0	0	7	0	1	0	7	0	0	0	0	4
13	20000	0	0	5	0	1	0	5	0	0	0	0	7
12	20000	0	0	4	0	1	0	4	0	0	0	0	7
11	20000	0	0	3	0	1	0	3	0	0	0	0	11
10	20000	1	1	0	3	1	65536	0	0	0	1	1	0
9	20000	1	1	0	3	2	65536	0	0	0	1	1	0
8	20000	1	1	0	3	2	65536	0	0	0	1	1	0
7	20000	1	1	0	3	1	65536	0	0	0	1	1	0
6	20000	0	0	2	0	1	0	2	0	0	0	0	11
5	20000	1	1	0	3	2	65536	0	0	0	1	1	0

Herramientas de selección y filtrado

Todas las tablas presentes en EasyWCS disponen de herramientas de selección y filtrado y se permite copiar el contenido tanto de una fila como de una celda.

Haga clic derecho sobre la cabecera de una de las columnas. Las opciones mostradas son:



- **Orden ascendente/descendente**: Seleccione esta opción o bien haga clic directamente sobre la cabecera de la columna para realizar la ordenación.
- **Quitar ordenación**: Cancela la ordenación anterior.
- **Agrupar por esta columna**: Muestra el resto de columnas agrupadas por la columna seleccionada. Seleccione esta opción o bien arrastre la/s columna/s sobre el área de agrupación que se encuentra sobre las cabeceras de columna.
- **Mostrar ventana agrupar por**: Indica, a modo de diagrama de árbol, las columnas en la que se ha aplicado la opción "Agrupar por esta columna".
- **Quitar esta columna**: Seleccione esta opción o bien arrastre la cabecera de la columna fuera del grid de tablas para eliminarla.
- **Selector de columnas**: Por defecto las tablas muestran todas las columnas. Puede arrastrarlas a esta ventana para eliminarlas de forma momentánea.
- **Ajuste perfecto**: Si modifica el tamaño por defecto (calculado para mostrar correctamente el nombre), con esta opción se ajusta de nuevo.
- **Ajuste perfecto (todas las columnas)**: Ajuste al tamaño por defecto de todas las columnas.
- **Editor de filtros**: Ventana que permite realizar un filtrado de los datos de la columna seleccionada ayudándose de varios criterios predefinidos.
- **Mostrar panel de búsqueda**: Permite realizar la búsqueda en todas las tablas de texto o número indicado en el desplegable. Los resultados son marcados en amarillo.
- **Mostrar fila de filtrado**: Muestra una fila debajo de las cabeceras para realizar un filtrado de forma individual por columna.

4.5.1.- COMUNICACIÓN HACIA EL WMS

Siempre que TMS necesite indicar algún dato al WMS lo hará siguiendo esta secuencia de pasos:

1. Enviará un mensaje al servicio EasyWCS.
2. El servicio EasyWCS evaluará el mensaje y lo insertará en la tabla correspondiente en BBDD.
3. El WMS debe consultar esta tabla en busca de datos.

Los mensajes posibles que pueden ser enviados desde el TMS al WMS, son:

- Enviar fin de orden: Indica al WMS que una orden en curso ha sido finalizada.
- Notificar evento: Notifica al WMS un evento (p.e. palet en PIE) determinado.

4.5.1.1.- TABLA COMANDOS AL HOST (WCS_MSG_HOST)

Haga clic en la pestaña Comandos al host de la pantalla **Tablas**.

El servicio de EasyWCS es el responsable de la inserción de nuevos registros, mientras que el WMS es el responsable de su consulta y eliminación.

La información mostrada es:

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero	Identificador del mensaje.
WarehouseCode	Entero	Identificador de almacén.
OperationType	Entero	Tipo de operación a realizar con el WMS. Los valores posibles son: • 0 - Finalización de movimiento. • 1 - Notificar un evento. • 2 - Ack del comando de borrado. • 3 - Ack del comando de reset.
EventType	Entero	Tipo de evento que se notifica de desde el programa de control. Los eventos definidos son: • 1 - Palet en PIE. • 2 - Pulsación de botón. • 3 - Evento de carga. • 4 - Evento de descarga. • 5 - Apilador/Desapilador.
TransportNumber	Entero	Número de transporte.
StationType	Entero	Tipo de estación.
StationNumber	Entero	Número de estación.
Flags	Entero	Flags de control del evento PIE.
WTU	Entero	Código UMA.
ErrorCode	Entero	Código de error. Los valores posibles son: • 0 – OK. • 1 - Error de depósito. • 2 - Error de extracción. • 7 - Error de altura. • 101 - Error en comando de borrado. El transporte está en la tabla WCS_MSG_TR_RUNNING.

		• 102 - Error en el comando de Borrado/Reset. El transporte no existe en el sistema.
AuxCode	Entero	Código auxiliar con el que se finaliza el movimiento. Su significado depende de la interpretación por parte del servicio EasyWCS.
ContainerType	Entero	Tipo de contenedor.
HeightType	Entero	Tipo de altura.
X	Entero	Posición X (lógica).
Y	Entero	Posición Y (lógica).
LocalX	Entero	Posición local X.
LocalY	Entero	Posición local Y.
Weight	Entero	Peso.
Side	Entero	Lado.
Depth	Entero	Profundidad.
Aisle	Entero	Pasillo.
AuxData	Cadena	Datos auxiliares necesarios (lectura de escáner, dimensiones, etc).
Aux1	Entero	Auxiliar del Evento/Fin de orden/Búsqueda (1).
…	…	…
Aux10	Entero	Auxiliar del evento/Fin de orden/Búsqueda (10).
Custom1	Cadena	Dato custom.
…	…	…
Custom20	Cadena	Dato custom.

Cada tipo de orden que nos indica el WMS (más información en "Tabla Pendientes (WCS_MSG_TR)" en la página 48) tiene una respuesta que EasyWCS escribirá en esta tabla **Comandos al host**. Esta respuesta depende del tipo de orden:

TIPO DE ORDEN	TIPO DE RESPUESTA	CÓDIGOS DE ERROR POSIBLES
1 - Orden de movimiento.	0 - Finalización de movimiento.	• 0 - Ok (sin error). • 1 - Error de depósito. • 2 - Error de extracción. • 7 - Error de altura.
2 – Operación de borrado.	2 - Ack de la operación de borrado.	• 0 - Ok (sin error). • 101 - Error en la operación de borrado. El transporte está en la "Tabla En ejecución (WCS_MSG_TR_RUNNING)" en la página 51. • 102 -Error en la operación de borrado/reset. El transporte no existe en el sistema.
3 – Operación de Reset.	3 - Ack de la operación de reset.	• 0 - Ok (sin error). • 102 -Error en la operación de borrado/reset. El transporte no existe en el sistema.
4 – Sobreescritura.	4 - Ack de la operación de Sobreescritura.	• 0 - Ok (Sin error). • 101 - Error la operación de sobreescritura. El transporte está en la "Tabla En ejecución (WCS_MSG_TR_RUNNING)" en la página 51.

		• 102 - Error en la operación de sobre-escritura. El transporte no existe en el sistema. • 103 - Error la operación de sobreescritura. El transporte está en “Tabla Transporte (WCS_MSG_TRANSPORT)” en la página 51.
5 - Cambio de prioridad/secuencia.	5 - Ack de la operación cambio de prioridad/secuencia.	• 0 - Ok (sin error). • 102 -Error en la operación de cambio de prioridad/secuencia. El transporte no existe en el sistema.

4.5.2.- COMUNICACIÓN DESDE EL WMS

Siempre que WMS necesite indicar algún dato al TMS lo hará siguiendo esta secuencia de pasos:

1. Añadirá un registro en la tabla correspondiente de BBDD.
2. El servicio EasyWCS sondea la tabla en busca de nuevos registros (bien sea bajo demanda o de manera periódica, dependiendo de la tabla).
3. En caso de ser necesario, enviará un mensaje al TMS.

Los mensajes posibles que pueden ser enviados desde el WMS al TMS, son:

- Orden generada: Se informa de una nueva orden generada. Este mensaje no llega al TMS, simplemente se almacena en la BBDD para su uso posterior.
- Nueva orden en curso: Se informa al TMS del transporte a ejecutar.
- Cancelación de orden generada: Permite eliminar una orden generada, aun no ejecutada.

El servicio EasyWCS consume órdenes desde la esta tabla. Para ello se basa en dos campos importantes:

1. OperationType: Indica el tipo de operación a realizar:
   - Ejecutar una orden.
   - Cancelar una orden generada, pero no ejecutada.

2. Priority: Indica la prioridad del mensaje.

Es importante tener en cuenta que el tipo de operación establece de por sí una prioridad, de manera que las operaciones de cancelación tienen más prioridad que las de ejecución.

A continuación se muestra el formato de cada una de las tablas involucradas:

Tabla Pendientes (WCS_MSG_TR)

Haga clic en la pestaña Pendientes de la pantalla **Tablas**.

Es responsabilidad del WMS insertar cada nueva orden generada en esta tabla, mientras que el servicio EasyWCS será el encargado de su eliminación.

Esta tabla puede ser actualizada por el servicio EasyWCS de la siguiente manera:

1. El WMS inserta una nueva orden. Esta orden será una ruta completa, especifica el origen y fin del movimiento, sin especificar las rutas parciales, es decir, las estaciones intermedias por las que pasa.
2. EasyWCS traspasa esta orden a la tabla Pendientes (local), actualizando los valores de las columnas *CurrentStation* y *NextStation*.

La información mostrada es:

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (clave)	Identificador del mensaje.
TransportNumber	Entero	Número de transporte
WarehouseCode	Entero	Identificador de almacén
Priority	Entero	Prioridad
Sequence	Entero	Secuencia
OperationType	Entero	Tipo de operación a realizar. Los valores posibles son: • 1 - Orden un movimiento: WMS informa que quiere realizar un movimiento con los datos indicados. • 2 - Borrado de un transporte: WMS informa de ello indicado **si aún no está siendo ejecutado**. En caso de estar siendo ejecutado o no existir el transporte, esta operación producirá un error que se informará al WMS. • 3 - Reset de un transporte: WMS informa de ello **indicado incluso si está siendo ejecutado**. Esto produce la eliminación del transporte de todo el sistema. En caso de no existir el transporte, produce un error que se informará al WMS.
WTU	Entero	Código UMA.
ContainerType	Entero	Tipo de contenedor.
HeightType	Entero	Tipo de altura.
SourceStationType	Entero	Tipo de estación origen.
SourceStationNumber	Entero	Número de estación origen.
SourceX	Entero	Posición X (lógica) inicial.
SourceY	Entero	Posición Y (lógica) inicial.
SourceLocalX	Entero	Posición local X inicial.
SourceLocalY	Entero	Posición local Y inicial.
SourceSide	Entero	Lado inicial.
SourceAisle	Entero	Pasillo inicial.

SourceDepth	Entero	Profundidad inicial.
TargetStationType	Entero	Tipo de estación destino.
TargetStationNumber	Entero	Número de estación destino.
TargetX	Entero	Posición X (lógica) final.
TargetY	Entero	Posición Y (lógica) final.
TargetLocalX	Entero	Posición local X final.
TargetLocalY	Entero	Posición local Y final.
TargetSide	Entero	Lado final.
TargetAisle	Entero	Pasillo final.
TargetDepth	Entero	Profundidad final.
AuxData	Cadena	Datos auxiliares.
SpeedX	Entero	Velocidad X inicial (en %).
SpeedY	Entero	Velocidad Y inicial (en %).
SpeedZ	Entero	Velocidad Z inicial (en %).
Custom1	Cadena	Dato custom.
…	…	…
Custom20	Cadena	Dato custom.

Las órdenes insertadas por el WMS en esta tabla disponen de dos campos que permiten al sistema EasyWCS priorizar y ordenar las órdenes entregadas al TMS:

- **Priority**: Establece el orden en que se seleccionarán las órdenes a entregar al TMS. La prioridad más alta es 1, a medida que aumenta menos prioridad tiene la orden.
- **Sequence**: Permite un segundo nivel de ordenación para órdenes de la misma prioridad. Al igual que el campo Priority, el valor 1 indica la prioridad más alta, a medida que aumenta este valor disminuye el orden.

Cada tipo de orden tiene una respuesta que EasyWCS escribirá en la tabla Comandos al Host. Esta respuesta depende del tipo de orden. Más información en “Tabla Comandos al host (WCS_MSG_HOST)” en la página 45.

Si EasyWCS no traspasa esta orden a la tabla Pendientes (local) es debido a que esta ruta completa no es interpretable por EasyWCS por lo que deberá ser revisada.

Tabla Pendientes (local) (WCS_MSG_TR_LOCAL)

Haga clic en la pestaña Pendientes (local) de la pantalla **Tablas**.

Esta tabla es la utilizada por el servicio EasyWCS para almacenar las órdenes pendientes de ejecutar. Estas órdenes pueden llegar a esta tabla de dos maneras:

1. WMS inserta en la tabla **Pendientes**. Cuando EasyWCS detecta esta orden, identifica la ruta a la que pertenece y la copia en **Pendientes (local)**, actualizando los campos *CurrentStation* y *NextStation*.
2. Tras un fin de orden, si los campos *NextStation* y *TargetStation* no coinciden, significa que se trata de una orden de una ruta parcial, por lo que el servicio

EasyWCS inserta en **Pendientes (local)**, actualizando los campos *CurrentStation* y *NextStation*.

El formato de la tabla es el mismo que **Pendientes**, pero añade los campos *CurrentStation* y *NextStation*:

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (clave)	Identificador del mensaje.
TransportNumber	Entero	Número de transporte.
WarehouseCode	Entero	Identificador de almacén.
CurrentStationType	Entero	Tipo de estación actual.
CurrentStationNumber	Entero	Número de estación actual.
NextStationType	Entero	Tipo de la próxima estación.
NextStationNumber	Entero	Número de la próxima estación.
Priority	Entero	Prioridad.
Sequence	Entero	Secuencia.
OperationType	Entero	Tipo de operación a realizar. Los valores posibles son: • 1 - Orden un movimiento. • 2 - Borrado de un transporte. • 3 - Reset de un transporte.
WTU	Entero	Código UMA.
ContainerType	Entero	Tipo de contenedor.
HeightType	Entero	Tipo de altura.
SourceStationType	Entero	Tipo de estación origen.
SourceStationNumber	Entero	Número de estación origen.
SourceX	Entero	Posición X (lógica) inicial.
SourceY	Entero	Posición Y (lógica) inicial.
SourceLocalX	Entero	Posición local X inicial.
SourceLocalY	Entero	Posición local Y inicial.
SourceSide	Entero	Lado inicial.
SourceAisle	Entero	Pasillo inicial.
SourceDepth	Entero	Profundidad inicial.
TargetStationType	Entero	Tipo de estación destino.
TargetStationNumber	Entero	Número de estación destino.
TargetX	Entero	Posición X (lógica) final.
TargetY	Entero	Posición Y (lógica) final.
TargetLocalX	Entero	Posición local X final.
TargetLocalY	Entero	Posición local Y final.
TargetSide	Entero	Lado final.
TargetAisle	Entero	Pasillo final.
TargetDepth	Entero	Profundidad final.
AuxData	Cadena	Datos auxiliares.
SpeedX	Entero	Velocidad X inicial (en %).
SpeedY	Entero	Velocidad Y inicial (en %).
SpeedZ	Entero	Velocidad Z inicial (en %).
Custom1	Cadena	Dato custom.
…	…	…
Custom20	Cadena	Dato custom.

Tabla Transporte (WCS_MSG_TRANSPORT)

Haga clic en la pestaña Transporte de la pantalla **Tablas**.

Esta tabla es la utilizada por el servicio EasyWCS para almacenar las órdenes completas que han comenzado su ejecución (tabla **En Ejecución**).

El WMS puede consular esta tabla, antes de insertar una nueva orden en la tabla **Pendientes,** para comprobar si ya existe en el sistema.

EasyWCS es el encargado de insertar el registro y eliminarlo de la tabla cuando se finaliza la orden completa.

El formato de la tabla es el mismo que la tabla de **Pendientes**.

Tabla En ejecución (WCS_MSG_TR_RUNNING)

Haga clic en la pestaña En ejecución de la pantalla **Tablas**.

Esta tabla almacena las órdenes que han sido entregadas al TMS por parte del servicio EasyWCS. EasyWCS es el responsable de la inserción de un nuevo registro y la eliminación de registros antiguos, mientras que el WMS únicamente realizará consultas sobre esta tabla.

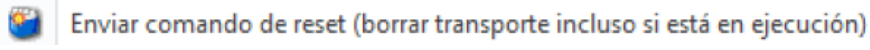
 En las tablas anteriores el campo **CreationDate** indica la fecha de creación del registro. El campo **UpdateDate** indica la fecha de la última modificación del registro.

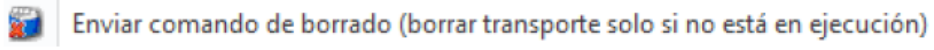
4.5.3.- OPERACIONES SOBRE TABLAS

En las tablas **Pendientes (local)**, **Transporte** y **En ejecución** el Diseñador permite eliminar transportes. Para ello seleccione un registro en la tabla y haga clic derecho del ratón. Seleccione la opción  Eliminar transporte .

 Esta acción elimina la orden solo de la tabla en la que realiza la selección, por lo tanto, tenga en cuenta que el sistema puede quedar inestable.

Para evitar el problema anterior, la tabla **Transporte** ofrece dos nuevas opciones:

- 

Elimina el registro de la tabla **Transporte** y de la tabla **En ejecución**.
- 

Elimina el registro de la tabla **Transporte** solo si el transporte no está **En ejecución**.

4.5.4.- TABLAS DE CONFIGURACIÓN

4.5.4.1.- TABLA ESTACIONES (WCS_STATIONS)

Haga clic en la pestaña Estaciones de la pantalla **Tablas**.

Esta tabla almacena las definiciones y estados de las estaciones. El servicio EasyWCS es el encargado de actualizar los estados ante la recepción de determinados mensajes por parte del TMS.

La información mostrada es la siguiente:

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (clave)	Identificador de la estación.
WarehouseCode	Entero	Identificador de almacén.
StationType	Entero	Tipo de estación.
StationNumber	Entero	Número de estación.
Name	Cadena	Nombre de la estación.
Status	Entero	Estado de la estación. Los valores posibles son: • 0 - Defecto/Manual/Esclavo-Manual. • 1 – Automático.
Loaded	Entero	Presencia. Los valores posibles son: • 0 – Presencial. • 1 - Sin presencia.
Capacity	Entero	Capacidad de la estación.
Occupation	Entero	Ocupación de la estación.
Aisle	Entero	Pasillo.
MaxCommands	Entero	Número máximo de órdenes generadas permitidas para esta estación.
Side	Entero	Lado en el que se encuentra la estación.
X	Entero	Posición X (lógica).
Y	Entero	Posición Y (lógica).
MaxLocalX	Entero	Número máximo de posiciones locales (en la coordenada X).
MaxLocalY	Entero	Número máximo de posiciones locales (en la coordenada Y).
Custom1	Cadena	Dato custom.
…	…	…
Custom20	Cadena	Dato custom.

4.5.4.2.- TABLA RUTAS (WCS_ROUTES)

Haga clic en la pestaña Rutas de la pantalla **Tablas**.

Esta tabla almacena las definiciones y estados de las rutas. El servicio EasyWCS es el encargado de actualizarlos ante la recepción de determinados mensajes.

La información mostrada es la siguiente:

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (clave)	Identificador de la estación.
WarehouseCode	Entero	Identificador de almacén.
SourceStationType	Entero	Tipo de estación origen de la ruta.
SourceStationNumber	Entero	Número de estación origen de la ruta.
TargetStationType	Entero	Tipo de estación destino de la ruta.
TargetStationNumber	Entero	Número de estación destino de la ruta.
RouteType	Entero	Tipo de ruta. Los valores posibles son: • 0 - Ruta completa. • 1 - Ruta parcial.
Name	Cadena	Nombre de la ruta.
Status	Entero	Estado de la ruta. Los valores posibles son: • 0 - Defecto. • 1 - En servicio. • 2 - Llena y en defecto. • 3 - Llena y en servicio.
Capacity	Entero	Capacidad de la ruta.
Occupation	Entero	Ocupación de la ruta.
Custom1	Cadena	Dato custom.
…	…	…
Custom20	Cadena	Dato custom.

Las nuevas estaciones o rutas creadas son automáticamente copiadas en la base de datos del WMS si el usuario tiene configurada esta opción en el servicio (más información en “Herramienta de configuración” en la página 58).

Estas tablas pueden ser sobreescritas por el WMS por lo que es recomendable revisar en las mismas los parámetros de las estaciones y rutas. Los valores aquí definidos han de ser los mismos que los configurados en EasyWCS.

4.5.4.3.- TABLA PARÁMETROS (WCS_CONFIG_PARAMETERS)

Haga clic en la pestaña Parámetros de la pantalla **Tablas**.

Esta tabla almacena parámetros que WMS considera oportuno comunicar al servicio EasyWCS.

Estos parámetros han de ser definidos en el Diseñador (más información en "Parámetros del Diagrama" en la página 29).

El WMS puede modificar los valores de estos parámetros en esta tabla, de manera que el servicio EasyWCS pueda realizar consultas sobre ella.

La información mostrada en esta tabla:

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (clave)	Identificador de la estación.
WarehouseCode	Entero	Identificador de almacén.
ParameterName	Cadena	Nombre del parámetro.
Value	Cadena	Valor del parámetro.

A modo de ejemplo, le mostramos un parámetro candidato a ser configurado en esta tabla:

Parámetro	Valor	Descripción
WH_MODE	"0"	*Modo de funcionamiento normal*: Realiza movimientos de entrada y salida alternativamente.
	"1"	*Modo de entrada*: Realiza movimientos de entrada. En caso de no disponer de ellos, realiza salidas.
	"2"	*Modo de salidas*: Realiza movimientos de salida. En caso de no disponer de ellos, realiza entradas.
	"3"	*Modo de reorganización*: Prioriza los movimientos de reorganización.

4.5.5.- TABLAS DE HISTÓRICOS Y ESTADÍSTICAS

4.5.5.1.- TABLA COMANDOS AL HOST (HIST) (WCS_MSG_HOST_HIST)

Haga clic en la pestaña Comandos al host (hist) de la pantalla **Tablas**.

El servicio EasyWCS inserta un registro en esta tabla cada vez que inserta uno nuevo en la tabla **Comandos al host**. Permite tener una confirmación de los mensajes entregados a la informática.

La información que se muestra es la que se indica al realizar el TMS estas operaciones.

4.5.5.2.- TABLA TRANSPORTE (HIST) (WCS_MSG_TRANSPORT_HIST)

Haga clic en la pestaña Transportes (hist) de la pantalla **Tablas**.

El servicio EasyWCS inserta un registro en esta tabla cada vez que elimina un registro en la tabla **Transporte**. Permite tener una confirmación de las órdenes finalizadas o eliminadas desde la propia tabla.

La tabla **Transporte (hist)** dispone de un campo adicional **Reason** que indica el motivo de la inserción en el histórico.

4.5.5.3.- TABLA EN EJECUCIÓN (HIST) (WCS_MSG_TR_RUNNING_HIST)

Haga clic en la pestaña En ejecución (hist) de la pantalla **Tablas**.

El servicio EasyWCS es el encargado de realizar el paso al histórico de las órdenes que consuma indicando el motivo del paso al histórico. EasyWCS inserta en la tabla **En ejecución (hist)** bajo las siguientes condiciones:

TABLA ORIGEN	ACCIÓN	DESCRIPCIÓN
En ejecución	Finalización del transporte.	Al finalizar un transporte, se insertará en el histórico
En ejecución	Reset del transporte.	Se insertará el transporte reseteado y la orden de reseteo.
Pendientes (local)	Reset del transporte.	Se insertará el transporte reseteado y la orden de reseteo.
Pendientes	Reset del transporte.	Se insertará el transporte reseteado y la orden de reseteo.
Pendientes (local)	Borrado del transporte.	Se insertará el transporte borrado y la orden de borrado.
Pendientes	Borrado del transporte.	Se insertará el transporte borrado y la orden de borrado.

La tabla **En ejecución (hist)** contiene tanto registros que provienen de **En ejecución** como de **Pendientes** o **Pendientes (local)**. Esto es así porque un registro aun no ejecutado que esté en Pendientes o Pendientes (local) puede ser reseteado o borrado, y se debe mantener un registro de ello para poder identificar correctamente las acciones realizadas.

La tabla **En ejecución (hist)** dispone de un campo adicional **Reason** que indica el motivo de la inserción en el histórico.

 En las tablas de históricos el campo **CreationDate** tiene el mismo valor que en el registro original e indica la fecha de creación del registro original. El campo **UpdateDate** contiene la fecha de paso al histórico.

4.5.5.4.- **TABLA ESTADÍSTICAS**

Haga clic en la pestaña Estadísticas de la pantalla **Tablas**.

Esta tabla almacena las estadísticas de las operaciones previamente activadas en el Diseñador. Para más información "¿Qué podemos ejecutar en el Modo Online?" en la página 34.

4.5.6.- GESTIÓN DE AVERÍAS

EasyWCS mantendrá una serie de tablas para almacenar las averías que se producen en la instalación.

4.5.6.1.- **TABLA WCS_FAULT_DEF**

En esta tabla se almacena el maestro de averías, es decir, las definiciones de todas las averías que pueden darse en la instalación.

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (Clave)	Id de la avería.
WarehouseCode	Entero	Identificador del almacén.
Code	Entero	Código de la avería.
Name	Cadena	Nombre de la avería.
Description	Cadena	Descripción de la avería.
Suggestion	Cadena	Sugerencia para resolver la avería.

4.5.6.2.- **TABLA WCS_FAULT**

Esta tabla almacena cada ocurrencia de una avería en la instalación.

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (Clave)	Id de la avería.
WarehouseCode	Entero	Identificador del almacén.
WcsFaultDefFk	Entero	Clave ajena al elemento de WCS_FAULT_DEF que representa la avería.
Machine	Cadena	Máquina en la que se produce la avería.
FaultDate	Fecha	Fecha/hora en la que se produce la avería.
ACKUser	Cadena	Usuario que acusa la avería (no tiene por qué resolverla)).
ACKDate	Fecha	Fecha a la se acusa la avería.
Computer	Cadena	Computador en el que se produce la avería.
FaultEndDate	Fecha	Fecha/hora a la que la avería se cierra (se resuelve).

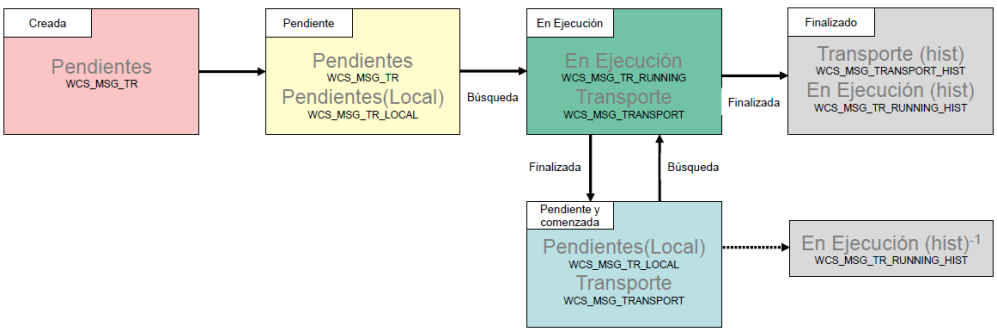
4.5.6.3.- **TABLA WCS_FAULT_INT**

Tabla que almacena las cadenas internacionalizadas de las averías.

CAMPO	TIPO	DESCRIPCIÓN
Id	Entero (Clave)	Id de la avería.
WarehouseCode	Entero	Identificador del almacén.
LanguageCode	Entero	Código del idioma.
Name	Cadena	Nombre de la avería.
Description	Cadena	Descripción de la avería.
Suggestion	Cadena	Sugerencia para resolver la avería.
WcsFaultDefFk	Fecha	Clave ajena al elemento de WCS_FAULT_DEF que representa la avería.

4.5.7.- **RESUMEN GRÁFICO DE LAS COMUNICACIÓN EASYWCS - WMS**

A continuación le mostramos un gráfico donde se explica el flujo que sufre una orden entre las tablas.



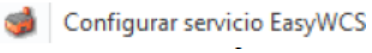
5. CAPITULO 5: GUÍA RÁPIDA DEL SERVICIO

5.1.- ¿QUÉ ES EL SERVICIO EASYWCS?

El servicio EasyWCS se encarga de gestionar los movimientos a través de una base de datos (BBDD). Ofrece la robustez de un sistema de transporte basado en la experiencia de otros productos de MECALUX.

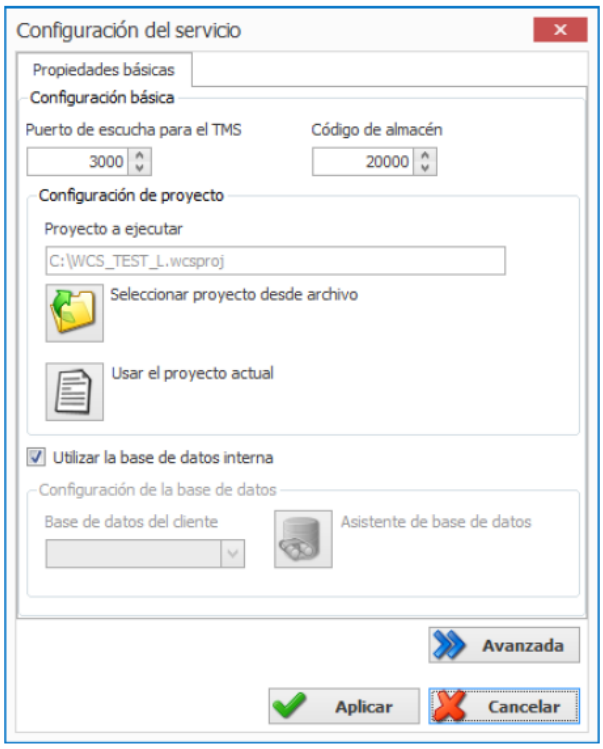
5.2.- HERRAMIENTA DE CONFIGURACIÓN

Debido a la gran cantidad de parámetros disponibles en EasyWCS (en su fichero de configuración) le indicamos en este mismo capítulo una "Guía rápida de configuración" en la página 67 para que, según las necesidades de pruebas o producción, siga los puntos que realmente son necesarios de configurar tras instalar EasyWCS. No obstante, EasyWCS le ofrece un completo asistente de configuración.

Haga clic en la opción Servicio EasyWCS Local > 

del menú principal. El Diseñador muestra la ventana "**Configuración del servicio**".

Recuerde, es imprescindible reiniciar el servicio para que los cambios surtan efecto.



Los campos de la pestaña **Propiedades básicas** son:

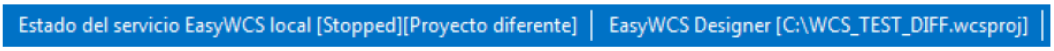
- **Puerto de escucha para el TMS**: Puerto en el que se escucharán las conexiones entrantes.

  Debe configurar un puerto válido entre 1024 y 65535.

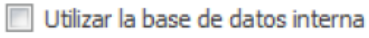
- **Código de almacén**: Indica el número de almacén con el que se trabajará.
- **Proyecto a ejecutar**: path del proyecto utilizado en la ejecución del servicio EasyWCS.

  – Seleccionar proyecto desde archivo.
  – Usar el proyecto actual.

La barra inferior del Diseñador le indica si el proyecto abierto en el Diseñador es diferente al configurado en el servicio.



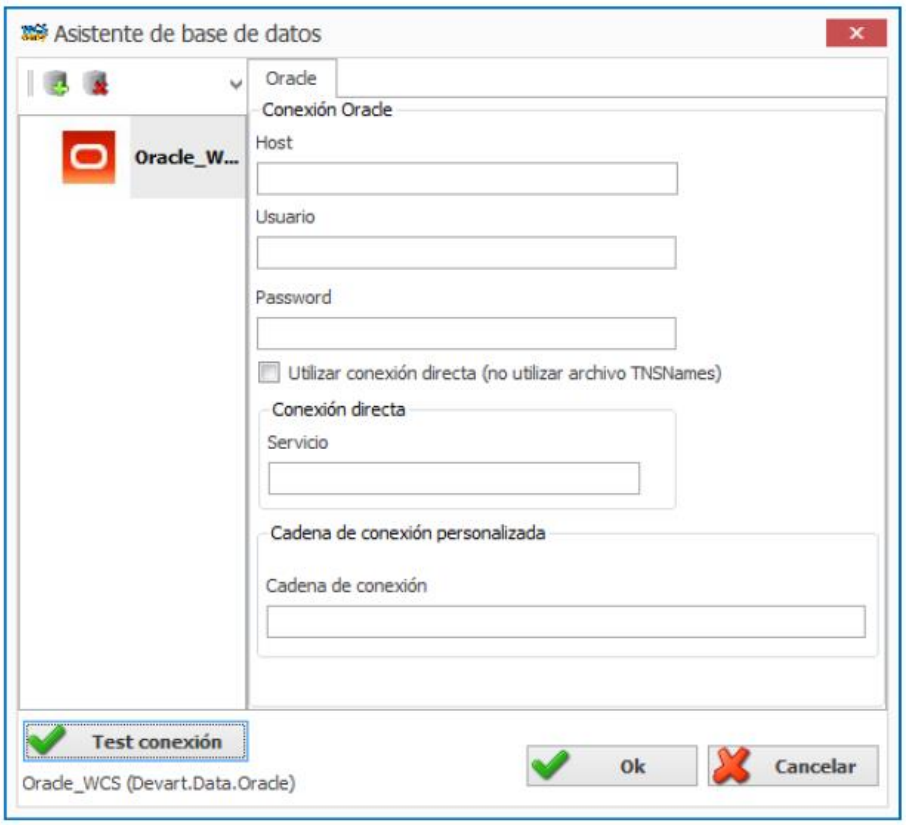
- Utilizar la base de datos interna : seleccione esta opción si desea utilizar la BBDD interna disponible para pruebas. La gestión de la misma la puede realizar consultando “Gestión de la BBDD interna” en la página 39.

Gracias al sistema de "tablas" de intercambio que permite la interconexión de EasyWCS con otros sistemas es posible configurar distintas bases de datos. Oracle, SQLServer y MySQL están soportadas por EasyWCS. Para configurarlas desactive la opción anterior 

. El Diseñador habilita los siguientes campos:

- **Base de datos Cliente**: seleccione mediante el desplegable una de las conexiones a BBDD ya creadas.
-  **Asistente de bases de datos**

  Haga clic sobre el icono. El diseñador muestra el asistente:



  Haga clic en  para configurar una nueva BBDD.

  En la ventana **Nueva Conexión** indique el Nombre y seleccione una de las BBDD soportadas por EasyWCS.

  A continuación detallamos los campos disponibles dependiendo de la

  BBDD seleccionada:

  - ***ORACLE***
    Host: Nombre o dirección IP donde está la BBDD.
    Usuario, **Password.**
    Utilizar conexión directa: Nombre o dirección IP donde está la BBDD si no se utiliza el archivo TNSNames.

Cadena de conexión personalizada: puede indicar la cadena directamente, pe, `"Server=127.0.0.1; Database=WCS_SCH; Uid=myUser; Pwd=myPass;"`

- **MySQL** y **SQLSERVER**
  Host: Nombre o dirección IP donde está la BBDD.
  Base de datos: Nombre del esquema.
  Usuario, Password.
  Cadena de conexión personalizada: puede indicar la cadena directamente, pe, `"Server=127.0.0.1; Database=WCS_SCH; Uid=myUser; Pwd=myPass;"`

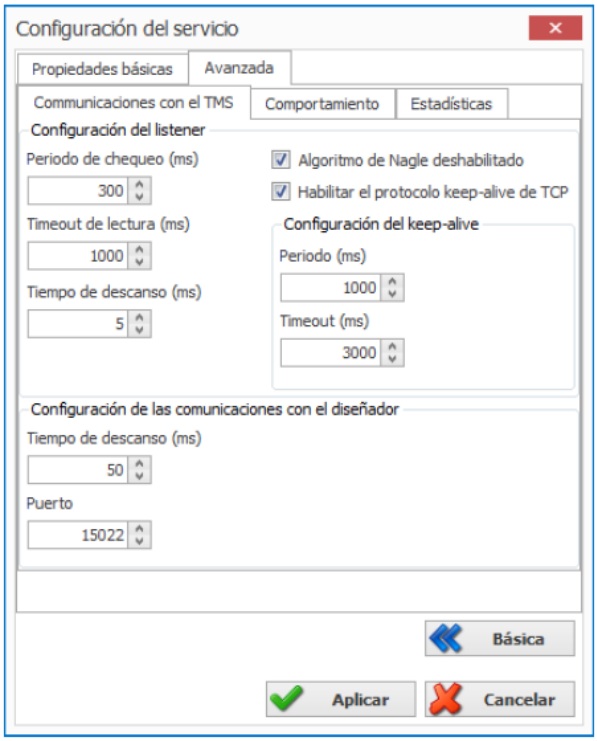
Haga clic en 

para mostrar en la ventana **Configuración del servicio** la pestaña **Avanzada**. En esta pestaña se realiza una configuración del servicio más personalizada.

Los parámetros por defecto ofrecidos son suficientes en la mayoría de los casos.

La pestaña **Avanzada** dispone a su vez de otras 4 pestañas:

- Pestaña **Comunicaciones con el TMS.**



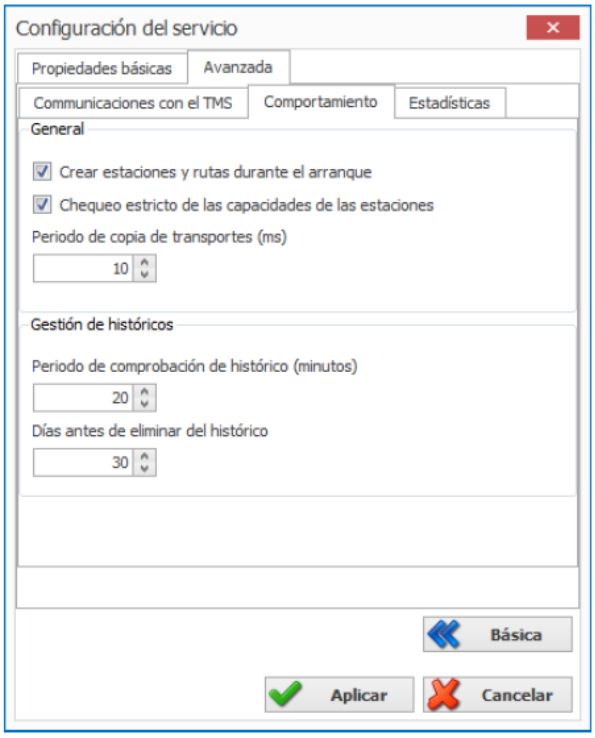
  – *Configuración del listener*: configura las características del módulo que acepta las conexiones desde el TMS.
  Periodo de chequeo (ms): cada cuanto tiempo se está chequeando si hay nuevas conexiones.

Timeout de lectura (ms): Tiempo tras el cual se da por perdida una trama y se da la conexión como desconectada.
Tiempo de descanso (ms): Tiempos entre lectura de tramas. Tiempos altos reducen el consumo de CPU pero aumentan el tiempo de respuesta ante una nueva conexión.
Algoritmo de Nagle deshabilitado: Indica si se quiere desactivar el protocolo de Nagle para el control de la congestión de red. Para aumentar la velocidad en la transmisión de tramas de pequeño tamaño, este protocolo debe ser desactivado.
Habilitar el procolo keep-alive: Activación el protocolo keep-alive de TCP el cual permite detectar errores de conexión de manera rápida y fiable.
Periodo (ms): Tiempo entre cada trama de consulta del keep-alive. Tiempos bajos disminuyen el tiempo necesario para detectar una desconexión pero aumentan la carga de red.
Timeout (ms): Tiempo tras el cual se da por perdida una trama de keep-alive y se da la conexión como desconectada. Tiempos bajos disminuyen el tiempo necesario para detectar una desconexión pero pueden dar falsos positivos.

- *Configuración de las comunicaciones con el diseñador*: configura las características del servidor de comunicaciones el cual permite comunicar el servicio EasyWCS con el Diseñador.
  Tiempo de descanso (ms): Tiempo en ms de descanso. Tiempos altos reducen el consumo de CPU pero aumentan el tiempo de respuesta ante una nueva conexión.
  Puerto: Puerto en el que se escucharán las conexiones desde el diseñador.

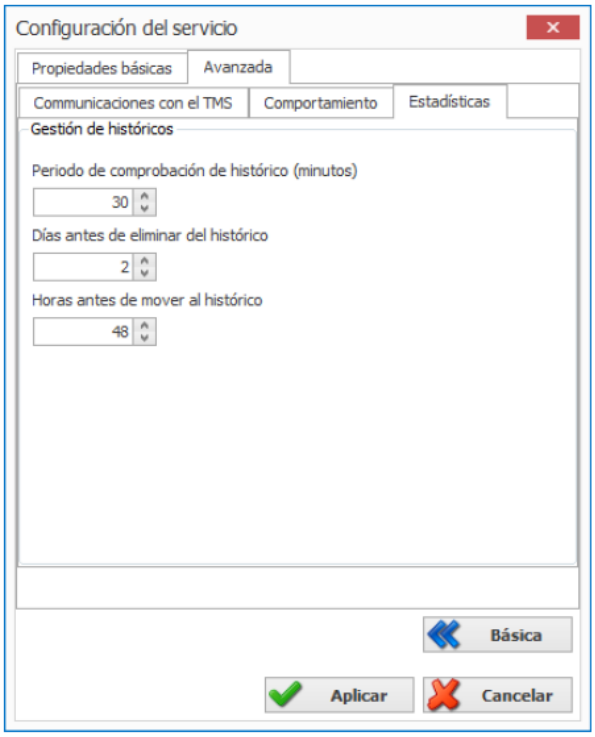
   Debe configurar un puerto valido entre 1024 y 65535, aunque se recomienda el establecido por defecto (15022).

- Pestaña **Comportamiento**:

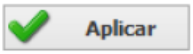


- *General.*
  Crear estaciones y rutas durante el arranque: Indica a EasyWCS que durante el arranque debe comprobar si existen y/o están modificadas las rutas y estaciones en la BBDD y en caso contrario crearlos.
  Chequeo estricto de las capacidades de las estaciones: es una comprobación que indica que la capacidad definida en la estación es la que utilizará el EasyWCS para realizar las operaciones.
  La no activación de esta opción implica, por ejemplo, que una estación definida en EasyWCS con capacidad 2 y ocupación 0, donde el WMS genere en la tabla de Pendientes 15 órdenes, si a posteriori el EasyWCS el que tratara estas órdenes mediante código custom podría llegar a devolver estas 15 órdenes a dicha estación. Como en el código estándar se realiza un chequeo siempre de la capacidad/ocupación esta opción ya va implícita. Esta opción está pensada como seguridad cuando se utiliza código custom.
  Periodo de copia de transportes (ms): periodo de copia de la tabla de Pendientes a la tabla Pendientes (local).
- *Gestión de históricos.*
  Periodo de comprobación de histórico (minutos): Indica cada cuantos minutos se comprobará el histórico en busca de entradas que haya que eliminar. Si el valor configurado es 0 no se eliminan nunca del historial.
  Días antes de eliminar del histórico: Días tras los que una entrada será eliminada del histórico.

- Pestaña **Estadísticas**:



– *Gestión de históricos.*
Periodo de comprobación de histórico (minutos): Indica cada cuantos minutos se comprobará el histórico de estadísticas en busca de entradas que haya que eliminar. Si el valor configurado es 0 no se eliminan nunca del historial.
Días antes de eliminar del histórico: Días tras los que una entrada será eliminada del histórico de estadísticas.
Horas antes de mover del histórico: Horas tras los que una entrada es movida de la tabla de estadísticas al histórico de estadísticas.

Finalmente haga clic en 

.

Recuerde, es imprescindible reiniciar el servicio para que los cambios surtan efecto.

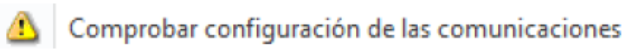
5.3.- ARRANQUE DEL SERVICIO

Una vez que hemos configurado correctamente el servicio de EasyWCS, procedemos al arranque del mismo. Para ello haga clic en o bien desde Windows >Servicios > "**EasyWCS**".

5.4.- HERRAMIENTAS DE DIAGNOSIS

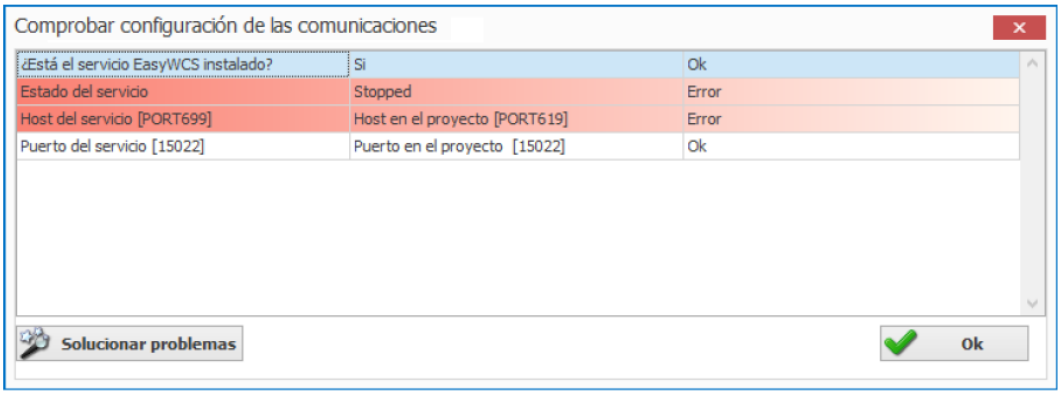
5.4.1.- ASISTENTE DE COMPROBACIÓN DE LAS COMUNICACIONES

EasyWCS dispone de un asistente que nos facilita hacer una comprobación básica de las comunicaciones entre el Diseñador y el servicio EasyWCS.

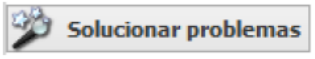
Haga clic en la opción Servicio EasyWCS Local > 

del menú principal o bien desde el icono  de la barra de herramientas.

La comprobación mostraría la siguiente ventana:



En este ejemplo muestra como Error el estado del servicio ya que está parado y también que el host en el proyecto es distinto al configurado en el proyecto actual del Diseñador.

Si desea solucionar cualquiera de las opciones de forma automática, haga clic en 

.

5.4.2.- LOGS

Los logs se crean en el directorio de datos de EasyWCS cuando tengan algo que trazar, es decir, no existirán logs de 0 Kb.

Dependiendo del Sistema Operativo puede ser C:\Documents and Settings\All Users\Application Data\Mecalux\EasyWCS\Logs o bien C:\ProgramData\Mecalux\EasyWCS\Logs.

5.4.2.1.- **DATASERVICECONTEXT.LOG**

En este log se puede encontrar la siguiente información:

- Operaciones realizadas sobre la base de datos del cliente.
- Errores en estas operaciones.

5.4.2.2.- **EASYWCSSERVICE.LOG**

En este log se puede encontrar la siguiente información:

- Inicio de la ejecución de EasyWCS.
- Modo de ejecución de EasyWCS.
- Excepciones no controladas.

5.4.2.3.- **ODATASERVICE.LOG**

En este log se puede encontrar la siguiente información:

- Información y errores en la inicialización y mantenimiento del servicio OData que publica las tablas de la BBDD del cliente.

5.4.2.4.- **SERVERPROTOCOLMANAGER.LOG**

En este log se puede encontrar la siguiente información:

- Tramas recibidas.
- Errores en las tramas.

5.4.2.5.- **SERVICEOBJECT.LOG**

En este log se puede encontrar la siguiente información:

- Inicio de la ejecución de EasyWCS.
- Errores en la carga de los módulos de EasyWCS.

5.4.2.6.- **SOCKETLISTENER.LOG**

En este log se puede encontrar la siguiente información:

- Aceptación de nuevas conexiones.

- Configuración de las nuevas conexiones.
- Recepción de mensajes de registro (SALI).
- Perdida de conexión.

5.4.2.7.- **STATISTICS.LOG**

En este log se puede encontrar la siguiente información:

- Información y errores en la inicialización y mantenimiento de las estadísticas.

5.4.2.8.- **USERACTIONS.LOG**

En este log se puede encontrar la siguiente información:

- Acciones que el usuario ha realizado desde el Diseñador.

5.4.2.9.- **WCSPROJECTMANAGER.LOG**

En este log se puede encontrar la siguiente información:

- Información acerca de la carga del fichero de configuración.
- Errores en la carga del fichero de configuración.

5.5.- GUÍA RÁPIDA DE CONFIGURACIÓN

Para la realización de pruebas en las que no es necesario trabajar con la BBDD real, EasyWCS dispone de una BBDD interna, seleccionada por defecto, por lo que tan solo es necesario:

1. Establecer el puerto de escucha para el sistema de control si no desea trabajar con el valor por defecto (3000). Más información en **Propiedades básicas** en la página 59.

Recuerde realizar en el sistema de control las configuraciones necesarias para comunicar con este puerto. Por ejemplo para el sistema de control de Galileo, ha de configurar en la consola la "Cadena de conexión del Cliente" con "PCx: 3000" donde PCx es el computador donde está instalado el servicio. El resto de campos a completar en la consola puede consultarlos en el **Manual de Configuración del EasyWMS Gateway** ya que siguen la misma filosofía al trabajar con el Agente de Transportes 4.0. Por ejemplo:

2. Seleccionar en el Diseñador el proyecto a ejecutar en el servicio EasyWCS. Más información en **Proyecto a ejecutar** en la página 59.

Para trabajar con la BBDD real es necesario, **además de los puntos anteriores**:

3. Configurar la conexión a la BBDD, más información en "Asistente de bases de datos".

Universidad de Oviedo

Trabajo Fin de Máster realizado por

Victoria Casares García

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

PLANIFICACIÓN Y PRESUPUESTO

Modelado de Transelevador
Automatic Warehouse Studio (AWS)

Julio de 2018

ÍNDICE PLANIFICACIÓN Y PRESUPUESTO

1. INTRODUCCIÓN

1.1. - VISIÓN GENERAL DEL DOCUMENTO

En el presente documento se incluye una estimación del coste que supone llevar a cabo el proyecto desarrollado, una descripción de la planificación temporal y fases del mismo.

En este presupuesto se detallan los recursos de hardware, software y humanos necesarios para el desarrollo e implementación del control automatizado de las fases involucradas en el proceso de desarrollo del programa de control y modelo de simulación de un transelevador con la herramienta DesignerIV, haciendo un presupuesto desglosado del proyecto y un presupuesto final del mismo.

En la planificación temporal, se indican las tareas y fases que constituyen este proyecto.

2. PLANIFICACIÓN

2.1. - APRECIACIÓN DE RECURSOS NECESARIOS

2.1.1.- RECURSOS HARDWARE

MECALUX Software Solutions		Recursos Hardware	
ID UNIDAD	DESCRIPCIÓN	UNIDAD DE MEDICIÓN	NÚMERO DE UNIDADES
HW01	Ordenador de sobremesa	Uds.	1
HW02	Monitor LCD 18"	Uds.	1
HW03	Tecaldo	Uds.	1
HW04	Ratón	Uds.	1

Tabla 2.1.- *Mediciones de recursos hardware del proyecto*

2.1.2.- RECURSOS SOFTWARE

MECALUX Software Solutions		Recursos Software	
ID UNIDAD	DESCRIPCIÓN	UNIDAD DE MEDICIÓN	NÚMERO DE UNIDADES
SW01	Microsoft Windows 10	Uds.	1
SW02	DesignerIV	Uds.	1
SW03	GalileoIV	Uds.	1
SW04	EasyWCS	Uds.	1
SW05	Microsoft Office 10	Uds.	1

Tabla 2.2.- *Mediciones de recursos software del proyecto*

2.1.3.- RECURSOS HUMANOS

La persona encargada del desarrollo e implementación del proyecto debe disponer de experiencia en desarrollo de programas de control, familiarizado con programación IEC61131-ST y debe tener conocimientos de logística.

Debe estar familiarizado también con el uso del software DesignerIV para el diseño del secuenciador del transelevador, con AWS (Automatic Warehouse Studio) para el diseño del modelo de simulación y con EasyWCS para gestionar las ordenes hacia DesignerIV

Para el cálculo de los costes de los recursos humanos se han estimado unas seis horas diarias en las fases de “Análisis” y “Requerimientos del proceso”, así como las fases de diseño y programación, pruebas y documentación.

MECALUX Software Solutions			Recursos Humanos
ID UNIDAD	DESCRIPCIÓN	UNIDAD DE MEDICIÓN	NÚMERO DE UNIDADES
HU01	Definicion: Analisis y Requerimientos	Horas	600
HU02	Implemtentación: Desarrollo de programación	Horas	700
HU03	Pruebas y Documentación	Horas	120

Tabla 2.3.- *Mediciones de recursos humanos del proyecto*

2.2. - PLANIFICACIÓN TEMPORAL

2.2.1.- ETAPAS DEL PROYECTO

Las etapas que apareen en el siguiente cuadro guardan relación co las bases para el desarrollo del proyecto señaladas en el documento "Memoria" durante el desarrollo de un proceso.

Nombre Etapa	Inicio Etapa	Fin etapa
Análisis y descripción del programa	12/02/2018	23/03/2018
Análisis y requerimientos	24/03/2018	30/03/2018
Análisis y desarrollo	01/04/2018	24/06/2018
Simulacón y pruebas	25/06/2018	10/07/2018

Tabla 2.4.- *Etapas del proyecto*

3. PRESUPUESTO

3.1. - CUADRO DE PRECIOS

3.1.1.- RECURSOS HARDWARE

MECALUX Software Solutions			Recursos Hardware
ID UNIDAD	DESCRIPCIÓN	UNIDADES	PRECIO UNITARIO (Euros)
HW01	Ordenador de sobremesa	1	3.200,00
HW02	Monitor LCD 18"	1	230,00
HW03	Tecaldo	1	35,00
HW04	Ratón	1	15,00

Tabla 3.1.- *Tabla de precios de los recursos hardware*

3.1.2.- RECURSOS SOFTWARE

MECALUX Software Solutions			Recursos Software
ID UNIDAD	DESCRIPCIÓN	NÚMERO DE UNIDADES	PRECIO UNITARIO (Euros)
SW01	Microsoft Windows 10	1	160,25
SW02	DesignerIV	1	3.200,00
SW03	GalileoIV	1	1.675,00
SW04	EasyWCS	1	2.300,00
SW05	Microsoft Office 10	1	85,00

Tabla 3.2.- *Tabla de precios de los recursos software*

3.1.3.- RECURSOS HUMANOS

MECALUX Software Solutions			Recursos Humanos
ID UNIDAD	DESCRIPCIÓN	UNIDAD DE MEDICIÓN	PRECIO UNITARIO (Euros)
HU01	Definicion: Analisis y Requerimientos	Horas	20,1
HU02	Implemtentación: Desarrollo de programación	Horas	14,5
HU03	Pruebas y Documentación	Horas	60,1

Tabla 3.3.- *Tabla de precios de los recursos humanos*

3.2. - PRESUPUESTOS PARCIALES

En este capítulo se ha realizado la imputación de una parte proporcional tanto del software como del hardware empleado en función del tiempo de uso. Este proyecto se ha realizado en 6 meses.

3.2.1.- RECURSOS HARDWARE

MECALUX Software Solutions		Recursos Hardware
ID UNIDAD	**DESCRIPCIÓN**	**IMPORTE (Euros)**
HW01	Ordenador de sobremesa	400,00
HW02	Monitor LCD 18"	28,75
HW03	Tecaldo	4,37
HW04	Ratón	1,87
	TOTAL	434,99

Tabla 3.4.- *Presupuesto parcial de recursos hardware*

3.2.2.- RECURSOS SOFTWARE

MECALUX Software Solutions		Recursos Software
ID UNIDAD	DESCRIPCIÓN	IMPORTE (Euros)
SW01	Microsoft Windows 10	20,56
SW02	DesignerIV	400,00
SW03	GalileoIV	209,37
SW04	EasyWCS	287,50
SW05	Microsoft Office 10	10,62
	TOTAL	928,05

Tabla 3.5.- *Presupuesto parcial de recursos Software*

3.2.3.- RECURSOS HUMANOS

MECALUX Software Solutions		Recursos Humanos
ID UNIDAD	DESCRIPCIÓN	IMPORTE (Euros)
HU01	Definicion: Analisis y Requerimientos	12.060,00
HU02	Implemtentación: Desarrollo de programación	10.150,00
HU03	Pruebas y Documentación	7.212,00
	TOTAL	29.422,00

Tabla 3.6.- *Presupuesto parcial de recursos humanos*

3.2.4.- PRESUPUESTO FINAL

MECALUX Software Solutions	Presupuesto Final
CAPITULO	**IMPORTE TOTAL (euros)**
Recursos Hardware	434,99
Recursos Software	928,01
Recursos Humanos	29.422,00
TOTAL	30.785,00

Tabla 3.7.- *Presupuesto final*

		EUROS
Presupuesto de ejecución de Material	=	30.785,00
Beneficio Industrial (6%)	=	1.847,10
Costes Generales (15%)	=	4.617,75
Suma de Gastos y Beneficios	=	37.249,85
I.V.A. (21%)	=	7822.47
Presupuesto de Ejecución por Contrata	=	45.072,32

Asciende el presupuesto de ejecución por contrata a la expresada cantidad de cuarenta y cinco mil setenta y dos con treinta y dos euros (45.072,32)

Gijón, Julio de 2018