CAPITULO 3

Desarrollo del proyecto

3.1 Introducción

En este capítulo se explicará detalladamente la parte software realizada con ayuda del programa STEP7 V5.3 que suministra el fabricante SIEMENS.

Por una parte, explicaremos detalladamente las pantallas configuradas con la herramienta SIMATIC PROTOOL/PRO CS opcional, que tuvimos que instalar con el programa STEP/ antes mencionado, para el panel táctil SIMATIC TP270 de SIEMENS. Así mismo, también se mostrará algún ejemplo de simulación de las mismas realizado con el paquete SIMATIC PROTOOL/PRO RT que también fue necesario instalar.

Por último, se mostrará la programación realizada en SCL que se introducirá en la memoria de trabajo del PLC que gobierna el sistema, el SIMATIC S7 300 de SIEMENS.

3.2 Introducción a STEP 7

En este primer apartado explicaremos brevemente el programa STEP 7 que usaremos en este proyecto.

¿En qué consiste el software STEP 7?

STEP 7 es el software estándar para configurar y programar los sistemas de automatización SIMATIC. STEP 7 forma parte del software industrial SIMATIC.

Funciones del software estándar

El software estándar le asiste en todas las fases de creación de soluciones de automatización, tales como

- crear y gestionar proyectos

- configurar y parametrizar el hardware y la comunicación
- gestionar símbolos
- crear programas, p.ej. para sistemas de destino S7
- cargar programas en sistemas de destino
- comprobar el sistema automatizado
- diagnosticar fallos de la instalación

El interface de usuario del software STEP 7 ha sido diseñado siguiendo los criterios ergonómicos más avanzados, lo que permite conocer rápidamente sus funciones.

Herramientas auxiliares

El software estándar STEP 7 ofrece toda una serie de herramientas:

Administrador SIMATIC

El Administrador SIMATIC gestiona todos los datos pertenecientes al proyecto de automatización, independientemente del sistema de destino (S7/M7/C7) donde se encuentren. El Administrador SIMATIC arranca automáticamente las herramientas necesarias para tratar los datos seleccionados.

Editor de símbolos

Con el editor de símbolos se gestionan todas las variables globales. Se dispone de las siguientes funciones:

- definir nombres simbólicos y comentarios para las señales del proceso (entradas y salidas), las marcas y los bloques,
- funciones de ordenación,
- importación/exportación de/hacia otros programas de Windows.

Todas las herramientas pueden acceder a la tabla de símbolos creada. Por consiguiente, detectan automáticamente si se ha modificado un parámetro de un símbolo.

Diagnóstico del hardware

El diagnóstico del hardware permite visualizar el estado del sistema de automatización, mostrando una vista general en la que aparece un símbolo cuando alguno de los módulos presenta un fallo o no. Con un doble clic en el módulo averiado se visualizan información detallada sobre el error. El volumen de información disponible depende del módulo en cuestión:

- visualización de informaciones generales sobre el módulo (p.ej. número de referencia, versión, denominación) y sobre su estado (p.ej. fallo),
- visualización de los fallos del módulo (p.ej. errores de canal) de la periferia centralizada y de los esclavos DP,
- visualización de los avisos del búfer de diagnóstico.

En el caso de las CPUs se visualizan además las siguientes informaciones:

- causas de una ejecución errónea del programa de usuario,
- duración del ciclo (máximo, mínimo y último),
- características y grado de utilización de la comunicación MPI,
- datos característicos (cantidad de entradas y salidas, marcas, contadores, temporizadores y bloques posibles).

Lenguajes de programación

Los lenguajes de programación KOP, AWL y FUP para S7-300/400 son parte integrante del software estándar.

- KOP (esquema de contactos) es un lenguaje de programación gráfico. La sintaxis de las instrucciones es similar a la de un esquema de circuitos. KOP permite observar la circulación de la corriente a través de contactos, elementos complejos y bobinas.
- AWL (lista de instrucciones) es un lenguaje de programación textual orientado a la máquina. En un programa creado en AWL, las instrucciones equivalen en gran medida a los pasos con los que la CPU ejecuta el programa. Para facilitar la

programación, AWL se ha ampliado con estructuras de lenguajes de alto nivel (tales como accesos estructurados a datos y parámetros de bloques).

- FUP (diagrama de funciones) es un lenguaje de programación gráfico que utiliza los cuadros del álgebra booleana para representar la lógica. Asimismo, permite representar funciones complejas (p.ej. funciones matemáticas) mediante cuadros lógicos.

Además ofrecemos otros lenguajes de programación opcionales. Como es el caso del lenguaje de programación **SCL** *(Lenguaje de Control Estructurado)*, que se incluye en el paquete opcional S7-SCL instalado para la realización de éste proyecto.

HW-Config: Configuración del hardware

Esta herramienta se utiliza para configurar y parametrizar el hardware de un proyecto de automatización. Se dispone de las siguientes funciones:

- Para configurar el sistema de automatización, se eligen primero los bastidores (racks) de un catálogo electrónico y luego se asignan los módulos seleccionados a los slots de los bastidores.

- La configuración de la periferia descentralizada se efectúa del mismo modo. También se asiste la periferia canal a canal (granular).

- Al parametrizar la CPU se pueden ajustar mediante menús propiedades tales como el comportamiento en el arranque y la vigilancia del tiempo de ciclo. Se asiste el modo multiprocesador. Los datos introducidos se depositan en bloques de datos del sistema.

- Al configurar los módulos, todos los datos se pueden ajustar en cuadros de diálogo. No es preciso efectuar ajustes mediante los interruptores DIP. La parametrización de los módulos se efectúa automáticamente durante el arranque de la CPU. Por consiguiente se puede p.ej. sustituir un módulo sin necesidad de repetir la parametrización.

- La parametrización de módulos de función (FMs) y de procesadores de comunicaciones (CPs) se efectúa con la misma herramienta de configuración del hardware de forma idéntica a como se parametrizan los demás módulos. Para los

FM y CP se dispone de cuadros de diálogo específicos de los módulos (que forman parte del volumen de suministro del paquete de funciones FM/CP). El sistema impide que se efectúen entradas incorrectas, ya que los cuadros de diálogo sólo ofrecen las entradas admisibles.

NetPro

Con NetPro, los datos se pueden transferir de forma cíclica y temporizada a través de MPI, permitiendo

- seleccionar las estaciones que intervienen en la comunicación e
- introducir la fuente y el destino de los datos en una tabla. La creación de todos los bloques a cargar (SDB) y su transferencia completa a todas las CPUs se efectúa de forma automática.

Además, existe la posibilidad de transferir los datos de forma controlada por eventos, pudiéndose

- definir los enlaces de comunicación,
- seleccionar los bloques de comunicación o de función de la librería de bloques integrada,
- parametrizar en el lenguaje de programación habitual los bloques de comunicación o de función seleccionados.

Aparte, existen otras herramientas opcionales muy útiles que se pueden incorporar a estas herramientas estándar. En especial hablaremos de una en concreto:

S7-PLCSIM

La aplicación S7-PLCSIM permite ejecutar y comprobar el programa de usuario en un sistema de automatización (PLC) simulado en un PC, o bien en una unidad de programación (como p.ej. en una PG 740, Power PG o Field PG). Puesto que la simulación se realiza sólo mediante el software STEP 7, no se requiere ninguna conexión con equipos hardware S7 (CPU o módulos de ampliación). El PLC S7

simulado permite probar y depurar programas para las CPUs S7-300 y S7-400, así como programas de WinLC.

S7-PLCSIM incorpora un sencillo interface de usuario para visualizar y modificar diversos parámetros utilizados por el programa (como p.ej. para activar y desactivar las entradas). Además se pueden usar varias aplicaciones del software STEP 7 mientras se va ejecutando el programa en el PLC simulado. Ello permite utilizar herramientas tales como la tabla de variables (VAT) para visualizar y modificar variables.

Mostramos a continuación la pantalla del Administrador SIMATIC (que arranca automáticamente al abrir el programa STEP7), desde donde se puede acceder a todos las herramientas disponibles:

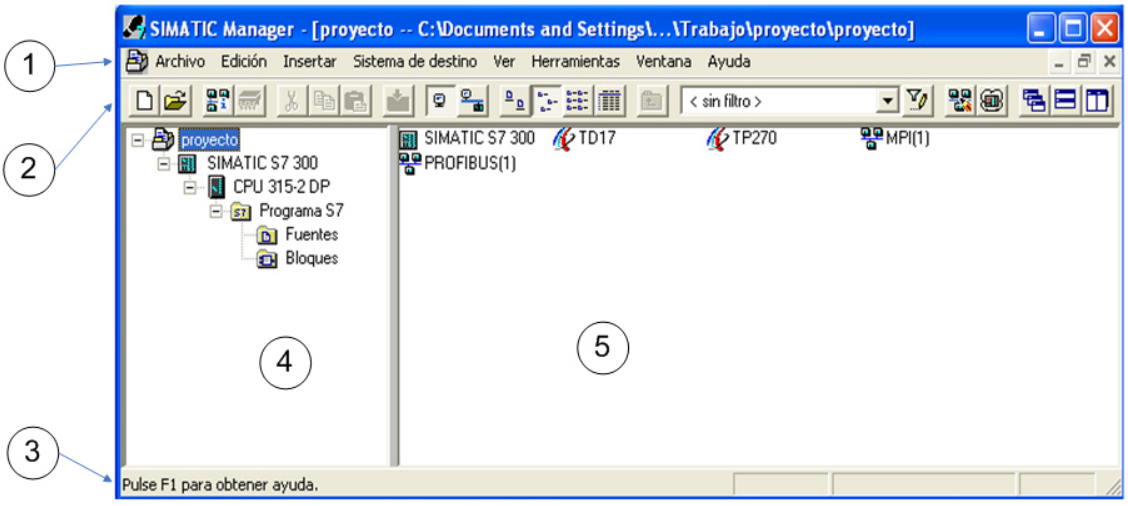


Figura 3.1. Administrador SIMATIC

En ella encontramos la siguiente estructura:

1. Barra de título y barra de menús

La barra de título y la barra de menús se encuentran siempre en el borde superior de la ventana. La barra de título contiene el título de la ventana y los botones para modificar el tamaño de la misma y para cerrarla. La barra de menús contiene todos los menús disponibles en la ventana.

Los menús que aparecen son los siguientes:

- Archivo: En este menú se podrá abrir, guardar, cerrar, crear proyectos nuevos, trabajar en multiproyectos en red, etc…

- Edición: Copiar, pegar, cortar objetos del proyecto, iniciar el RunTime para aplicaciones HMI si es posible, visualizar las propiedades de los objetos, etc..

- Insertar: Insertar programa S7, equipo, subred, Software S7, Bloque S7, tabal de símbolo, etc…

- Sistema de destino: Con el siguiente menú podremos cargar el proyecto realizado en el autómata correspondiente.

- Ver: Para visualizar las herramientas que deseemos, ver el programa online y offline (es decir, ver el programa que se está ejecutando en el equipo destino o ver el programa que se está ejecutando en el equipo local), modo de visualización de los iconos (grandes, pequeños, lista, etc).
- Herramientas: En este menú podremos configurar las preferencias, acceder al simulador S7LCSIM, acceder a NetPro, etc..

- Ventana: Configuramos la forma de ver las distintas ventanas abiertas de nuestro proyecto

- Ayuda: Soporte de ayuda que nos ofrece el propio programa.

2. Barra de herramientas

La barra de herramientas contiene botones con los que es posible ejecutar rápidamente con un clic del ratón los comandos de menú de uso frecuente que estén disponibles en ese momento. Situando el puntero del ratón unos instantes en un botón, se obtiene breve información sobre su función. Además, en la barra de estado se visualiza una explicación adicional.

Si no es posible acceder a la configuración actual, los botones aparecen atenuados.

3. Barra de estado

En la barra de estado se muestran informaciones contextuales.

Los punto **4** y **5** se corresponde con la **ventana de proyecto.**

En **4** se representa la estructura en árbol del proyecto.

En **5** aparece el contenido del objeto seleccionado en 4, conforme a la visualización elegida (iconos grandes, iconos pequeños, lista o detalles).

3.2.1. Estructura del proyecto

Comentamos la estructura del proyecto y los diversos objetos que lo forman:

Objeto proyecto: Es el nivel superior del proyecto. Dentro de éste se encuentran todos los equipos y redes que conforman el mismo y que es necesario configurar. Para nuestro caso en particular, se pueden visualizar en 5 los siguientes objetos:

- *TD17*. Es el Display de Texto de SIEMENS que usaremos como HMI (Interfaz Hombre Máquina) para que el operador pueda gobernar las alarmas y realizar movimientos manuales. Para configurarlo se necesita tener instalado la herramienta SIMATIC PROTOOL/PRO CS. No será objeto de estudio de nuestro proyecto.

- *TP270*. Es la Pantalla Táctil de SIEMENS que usaremos igualmente como HMI para que el operador pueda realizar tareas de supervisión, control, y gestión de alarmas. Para configurarlo es necesario tener instalado la herramienta opcional SIMATIC PROTOOL/PRO CS e igualmente recomendable tener instalado la otra herramienta SIMATIV PROTOLL/PRO RT para poder simular las pantallas tal y como se deben visualizar. En un punto más adelante se detallará su configuración.

- *SIMATIC S7 300*. Es el autómata en cuestión. En los niveles más bajos del proyecto se detalla su configuración.

- *MPI (Multi Point Interface)*. Es una de las interfaces que dispone el autómata S7 300 para comunicarse con otros periféricos. En nuestro caso, no se utilizará, pues la comunicación se realizará mediante PROFIBUS DP. Si se usase, se configura con la herramienta NetPro.

- *PROFIBUS*. Es la red con la que el autómata se comunicará con los diversos equipos. Se puede configurar con NetPro o bien con la herramienta HW Config, que en nuestro caso es el que utilizaremos.

Objeto Equipo. Es el siguiente elemento en la jerarquía del proyecto. El equipo en cuestión es el autómata SIMATIC S7 300. En este nivel podemos visualizar dos partes en la que se compone el mismo:

- *Hardware*. Desde aquí podremos configurar el Hardware del sistema, accediendo mediante un doble clic en éste icono a la herramienta HW Config.
- *CPU 315-2 DP*. Es la CPU del autómata. En los niveles siguientes se configura el mismo.

Objeto Módulo Programable. En este nivel se configura la CPU del autómata. Se compone de dos partes:

- *Enlaces*. Aquí se configura el tipo de enlace a utilizar. Dependerá del tipo de subred elegido y el protocolo de transferencia utilizado para establecer el enlace. En nuestro caso, se ha utilizado comunicación S7 que engloba subredes MPI y PROFIBUS e Industrial Ethernet. Se puede configurar con el programa NetPro.

- *Programa S7*. En esta carpeta se almacena el software que se debe almacenar en el autómata. En los niveles siguientes se detallan los elementos que lo componen.

Objeto Programa S7. En este nivel tenemos acceso al software en sí que se debe almacenar en el autómata. Encontramos 3 elementos en este nivel:

- *Símbolos*. Aquí se accede a la tabla de símbolos del programa. Aquí asignamos símbolos a señales y otras variables (funciones, entradas, salidas, zonas de memoria, etc…).
- *Fuentes*. Esta carpeta da acceso al programa fuente usado, en líneas de texto.
- *Bloques*. Es una carpeta que da acceso en el siguiente nivel a los bloques propiamente dichos que se van a almacenar en el autómata.

Objeto Fuente. Como se ha mencionado, tendrá el código fuente que se ha usado para programar la CPU del autómata.

Objeto Bloques. Como se ha dicho contendrá los bloques del programa. Se diferencia su contenido si la vista es offline (sin conexión con el sistema destino) u online (STEP 7 corriendo en el programa destino):

Una carpeta de bloques de una vista offline puede contener bloques lógicos (OB, FB, FC, SFB, SFC), bloques de datos (DB), tipos de datos de usuario (UDT) y tablas de variables. El objeto "Datos de sistema" representa bloques de datos de sistema.

La carpeta de bloques de una vista online contiene las partes ejecutables del programa residentes en el sistema de destino.

Los tipos de bloques existentes son los siguientes:

- ***Bloque de Función (FB):*** Un bloque de función (FB) es un bloque lógico que contiene una sección del programa y que tiene asignada un área de memoria. Cada vez que se llama a un FB hay que asignarle un DB de instancia.

- ***Función (FC)***: Una función (FC) es un bloque lógico que no tiene asignada ningún área de memoria propia. No necesita bloque de datos de instancia. A diferencia de un FB, una función puede retornar el resultado de la función (valor

de respuesta) al punto de llamada. Por consiguiente, la función se puede utilizar igual que una variable en una expresión. Las funciones del tipo VOID no tienen valor de respuesta.

- ***Bloque de Organización (OB)***: Al igual que un FB o una FC, el bloque de organización es una parte del programa de usuario que el sistema operativo llama cíclicamente o cuando se producen determinados eventos. Constituye el interface entre el programa de usuario y el sistema operativo.

- ***Bloque de Datos (DB)***: Los datos globales de usuario a los que acceden todos los bloques de un programa se depositan en bloques de datos. Cada FB, FC u OB puede leer o escribir estos bloques de datos.

  Existen dos tipos diferentes de bloques de datos:

  - **Bloque de datos** Bloques de datos a los que pueden acceder todos los bloques lógicos del programa S7. Todos los FBs, FCs y OBs pueden leer o escribir los datos contenidos en estos bloques de datos.

  - **Bloque de datos asociado a un FB (DB de instancia)** Los bloques de datos de instancia son bloques de datos asignados a un determinado bloque de función (FB). Incluyen datos locales para el bloque de función asignado.

- ***Tipo de datos de usuario (UDT)***: Los tipos de datos de usuario UDT son estructuras especiales creadas por el usuario. Dado que los tipos de datos de usuario tienen un nombre, pueden reutilizarse. Por definición puede utilizarse en la totalidad del programa de usuario, por lo que son tipos de datos globales. Por consiguiente, estos tipos de datos se pueden:

  - utilizar en bloques como tipos de datos simples o compuestos, o

  - utilizar como plantilla para crear bloques de datos de idéntica estructura.

3.2.2 Configuración de Hardware.

Con la Herramienta HW Config podemos configurar y visualizar la configuración del Hardware.

Mostramos a continuación la ventana del Hardware:

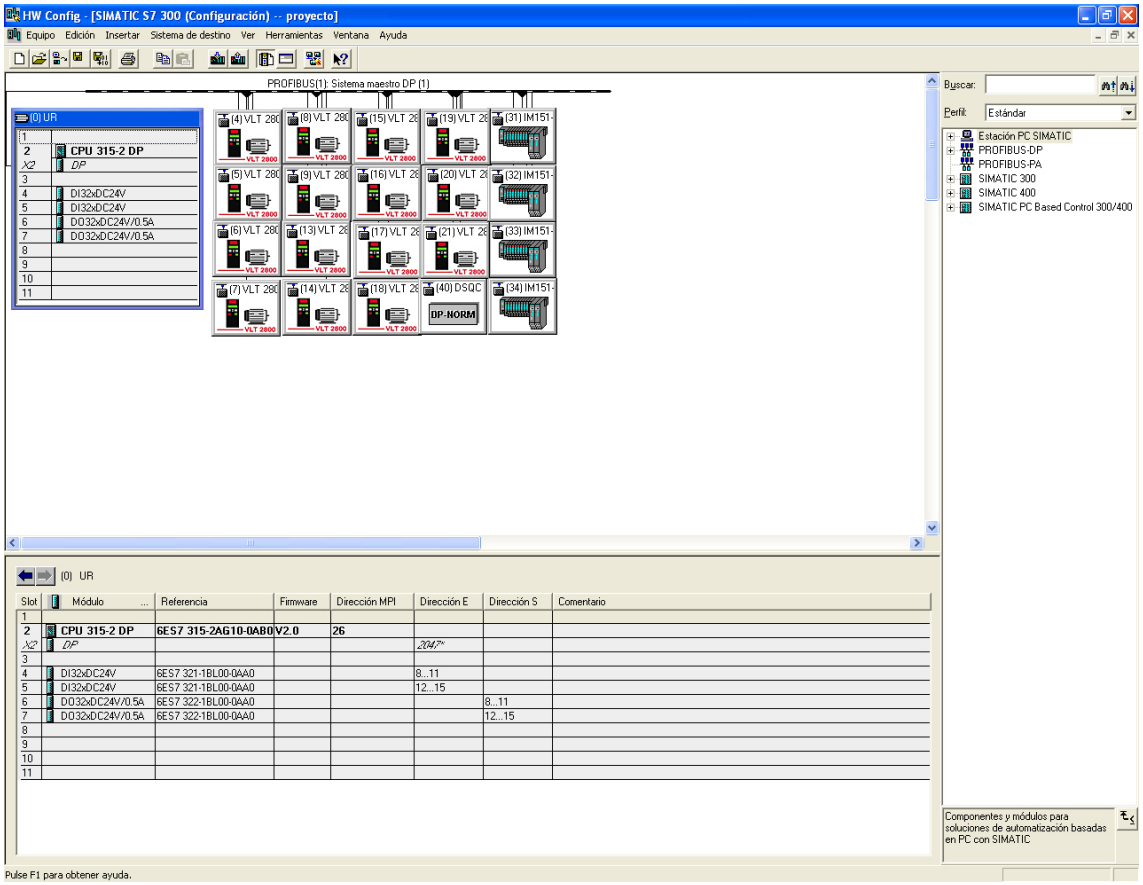


Figura 3.2. HW Config

En la misma observamos los siguientes elementos:

- la ventana del equipo en la que se emplazan los bastidores (ventana superior izquierda)y
- la ventana "Catálogo de hardware" de la que se seleccionan los componentes de hardware requeridos, p. ej. bastidor, módulos y módulos interface. (ventana de la derecha)

En la parte inferior de la ventana del equipo aparece una vista detallada del bastidor que se ha insertado o seleccionado. Allí se visualizan en forma de tabla las referencias y las direcciones de los módulos.

Una vez comentado esto, observamos los elementos que aparecen en la ventana del equipo:

- La ventana UR (0). Hace mención al bastidor 0. Es el bastidor donde se ubicará el PLC. Dentro de esta ventana observamos los distintos módulos que conforman el autómata:

    - CPU315-2DP.
    - Interfaz DP
    - 2 módulos de 32 Entradas Digitales DI32 24V
    - 2 módulos de 32 salidas Digitales DO32 24V/0.5A

- Cable PROFIBUS DP.

- Por último, observamos los diversos equipos conectado al PROFIBUS (aparte del PLC):

    - 16 Variadores de Frecuencia DANFOSS
    - 4 ET200S de SIEMENS
    - 1 DSQC 352 PROI-01, módulo para las comunicaciones con el Robot.

Un apartado que querríamos destacar es la configuración de la CPU. Todos y cada uno de los elementos se pueden configurar como se deseen. Así por ejemplo, a los ET200S se pueden establecer las direcciones de entradas y salidas que deseemos, al igual que a los variadores o al módulo de comunicación del robot. Pero dentro de la CPU, pinchando dos veces se puede configurar algo que nos será de gran utilidad: el tiempo de ciclo y la marca de ciclo. Mostramos la ventana que se observa al pinchar dos veces en el componente de la CPU315-2DP.

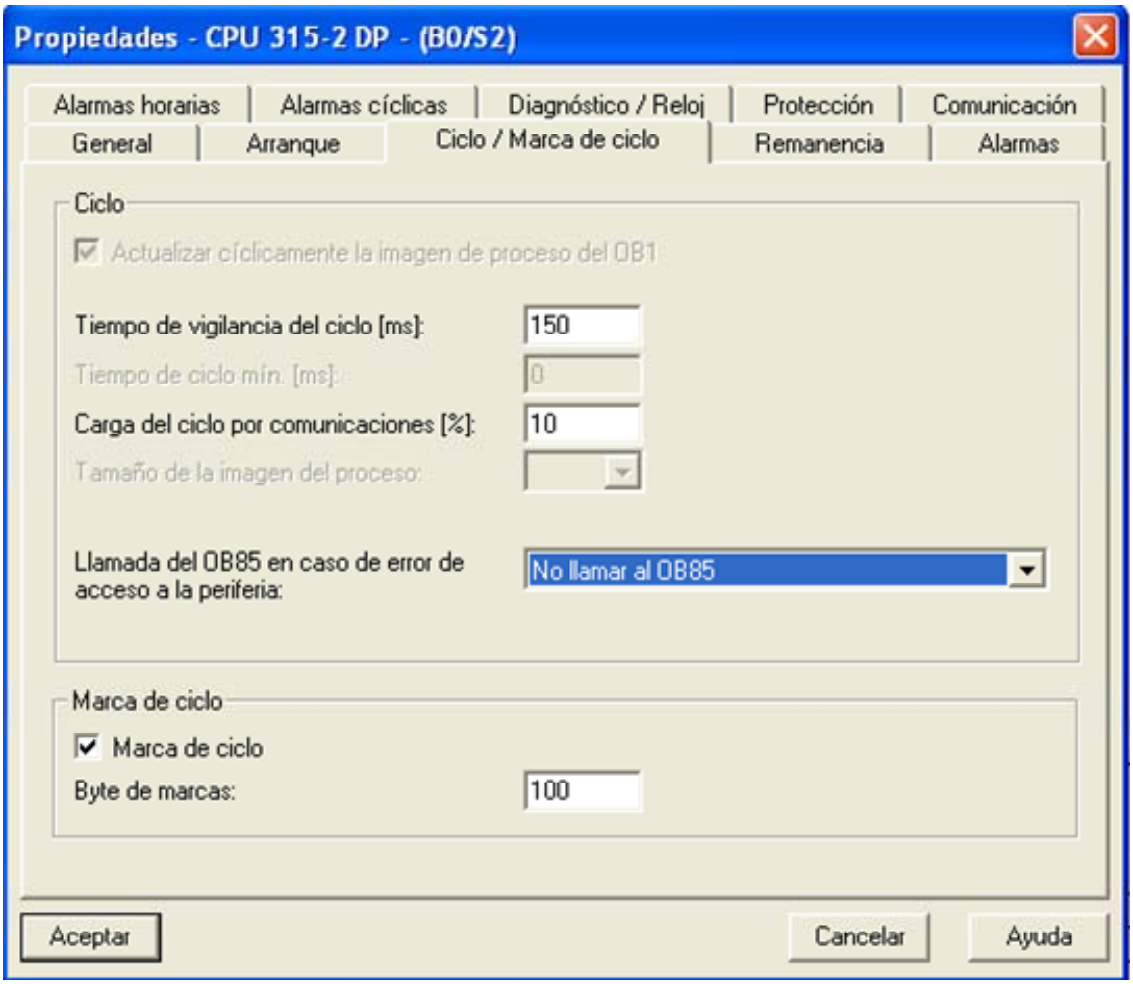


Figura 3.3. Ciclo/Marca Ciclo

Observamos varias pestañas para configurar la CPU:

Alarmas horarias, Alarmas cíclicas, Diagnóstico/Reloj, Protección, Comunicación, General, Arranque, Ciclo/Marca de ciclo, Remanencia, Alarmas.

No entraremos demasiado en detalle en cada una de las opciones. En principio basta con dejar las opciones marcadas por defecto. Nos centraremos en un aspecto que sí nos interesará conocer en profundidad: el tiempo de ciclo y la marca de ciclo.

El tiempo de ciclo es el tiempo que el sistema operativo necesita para ejecutar el programa cíclico, así como todas las partes del programa que interrumpen dicho ciclo (por ejemplo: la ejecución de otros bloques de organización) y las actividades del sistema (por ejemplo: la actualización de las imágenes del proceso). Este tiempo es vigilado por el sistema. El tiempo de ciclo (TZ) no es igual para cada ciclo.

Para regular y controlar el tiempo de ciclo, usamos el **Tiempo de vigilancia del ciclo.**

Con STEP 7 se puede modificar el tiempo de vigilancia del ciclo preajustado. Transcurrido este tiempo, la CPU pasa a STOP o se llama el OB 85, en el cual puede definirse cómo debe reaccionar la CPU al error de tiempo. En nuestro caso, se ha optado por pasar a STOP simplemente. Se ha definido el tiempo de vigilancia de ciclo en 150 ms.

Por otro lado, está la **Marca de ciclo.**

Una marca de ciclo es una marca que modifica su estado binario periódicamente con un ciclo de trabajo de 1:1. Parametrizando la marca de ciclo con STEP 7 se puede definir qué byte de marcas de la CPU se utiliza como byte de marcas de ciclo.

Utilidad

Las marcas de ciclo se pueden utilizar en el programa de usuario, por ejemplo, para controlar avisadores luminosos con luz intermitente o para iniciar procesos que se repitan periódicamente (como la captación de un valor real).

Frecuencias posibles

Cada bit del byte de marcas de ciclo tiene asignada una frecuencia. La tabla siguiente muestra la asignación:

Bit del byte de la marca de ciclo	7	6	5	4	3	2	1	0
Duración del período (s)	2,0	1,6	1,0	0,8	0,5	0,4	0,2	0,1
Frecuencia (Hz)	0,5	0,625	1	1,25	2	2,5	5	10

Tabla 3.1 Frecuencias posibles

En nuestro caso, se ha reservado el byte 100 para marca de ciclo. En el apartado referente al programa S7, se comprobará el uso que se le ha dado, del cual se observará que se hará uso del bit 7, para conseguir una frecuencia de 0,5Hz (2 segundos de duración, para comunicación con el TP270), y del bit 3 (2 Hz) frecuencia que se usará para el parpadeo de luces intermitentes.

3.3. Configuración de la pantalla TP270

En este apartado, detallaremos la configuración de las pantallas para la pantalla táctil TP270 de SIEMENS realizado con ayuda del paquete SIMATIC PROTOOL/PRO CS, y el simulador SIMATIC PROTOOL/PRO RT.

3.3.1. Introducción a PROTOOL

Para configurar la pantalla TP270 de SIEMENS, lo primero que hay que hacer es crear un nuevo equipo "*SIMATIC OP*", desde el Objeto Proyecto, dentro del *Administrador SIMATIC*.

A dicha estación la llamamos TP270. En la figura 1.2 expuesta anteriormente la observamos ya creada. Haciendo doble click sobre el nuevo icono creado se nos abre automáticamente el programa SIMATIC PROTOOL/PRO CS (que debe estar previamente instalado). Y a continuación nos saldría el asistente para crear un nuevo proyecto. No entraremos en demasiados detalles por considerar que este proceso carece de dificultad. Con ayuda del asistente se selecciona el tipo de HMI que se quiere configurar, en nuestro caso, se selecciona el Panel TP270 de 6" que encontramos dentro del menú "Sistemas Basados en Windows"; y el protocolo que se quiere utilizar, esto es, en nuestro caso concreto el SIMATIC S7 – 300/400 V6.0, ya que el autómata que estamos utilizando es un SIMATIC S7 de la serie 300.

Tras seguir los pasos detallados por el asistente llegamos a la siguiente ventana:

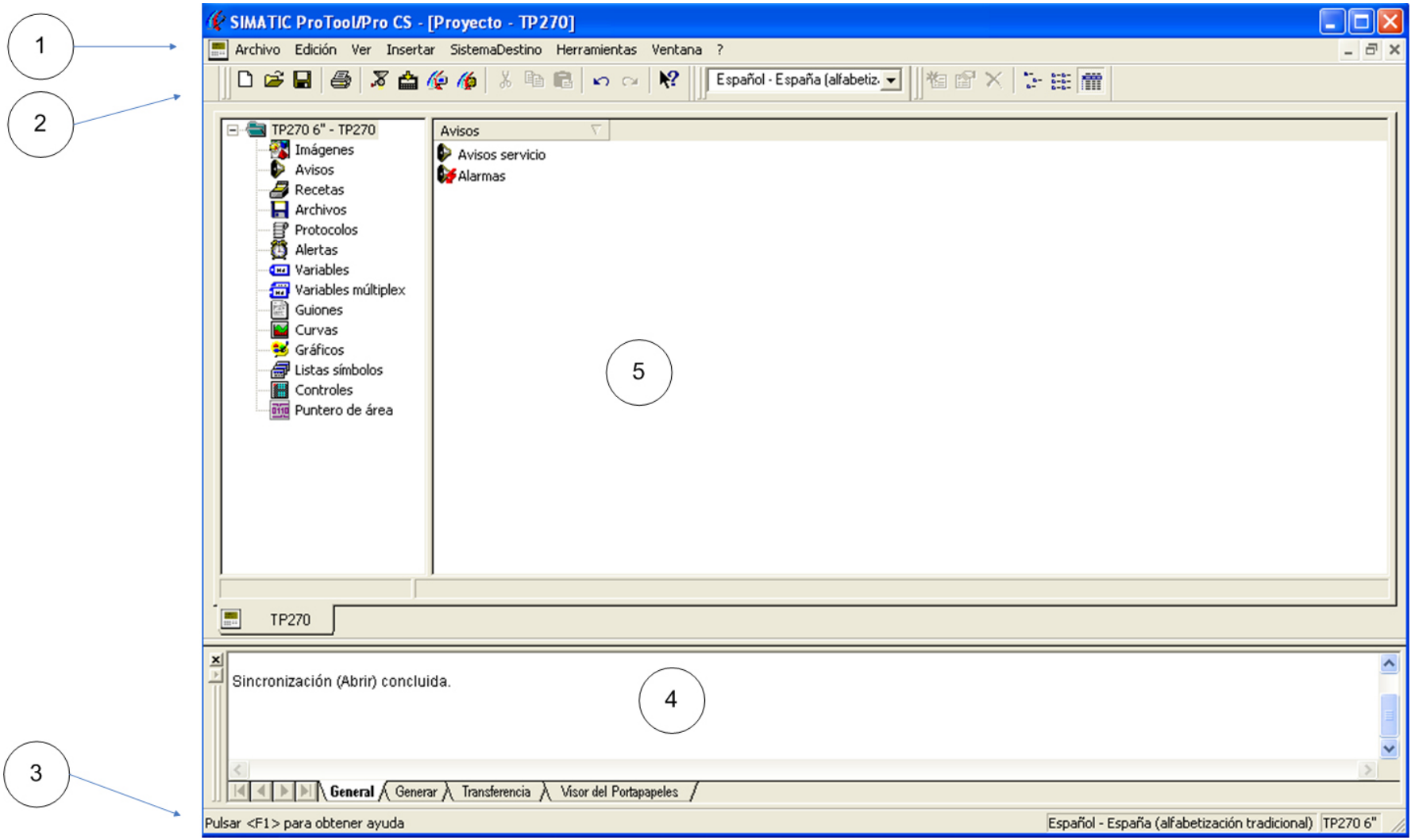


Figura 3.4. PROTOOL/PRO CS

Observamos los siguientes elementos:

1. Barra de título y barra de menús

La barra de título y la barra de menús se encuentran siempre en el borde superior de la ventana. La barra de título contiene el título de la ventana y los botones para modificar el tamaño de la misma y para cerrarla. La barra de menús contiene todos los menús disponibles en la ventana.

Los menús que aparecen son los siguientes:

- Archivo. Aquí se podrá abrir, cerrar, guardar proyectos, crear nuevos proyectos, transferir el proyecto al HMI, Iniciar PROTOOL/PRO RT, Iniciar el Simulador de PROTOOL/PRO RT, etc..

- Edición: Copiar, pegar, cortar objetos del proyectos, visualizar las propiedades de los objetos, etc..

- Ver: Para visualizar las herramientas que deseemos, modo de visualización de los iconos (pequeños, lista o detalle), Referencia cruzada (para las imágenes), etc..

- Insertar: Imagen, Aviso, Receta, etc..

- SistemaDestino: Imagen/teclas (para definir la distribución de la indicación en el display y configurar teclas de funciones globales), Funciones (para definir puntos de entrada donde se ejecutarán funciones globales), Ajustes para la unidad de operación (pasword para el nivel superior, ajustes de los avisos, ajustes de la fuente a utilizar en el proyecto, etc…

- Herramientas: Ajustes OLE (para configurar los programas de edición de gráficos), configuración de los nombres de los distintos objetos del proyecto, etc..

- Ventana: Configuramos la forma de ver las distintas ventanas abiertas de nuestro proyecto

- Ayuda (?): Soporte de ayuda que nos ofrece el propio programa.

2. Barra de herramientas

La barra de herramientas contiene botones con los que es posible ejecutar rápidamente con un clic del ratón los comandos de menú de uso frecuente que estén disponibles en ese momento. Situando el puntero del ratón unos instantes en un botón, se obtiene breve información sobre su función. Además, en la barra de estado se visualiza una explicación adicional.
Si no es posible acceder a la configuración actual, los botones aparecen atenuados.

3. Barra de estado

En la barra de estado se muestran informaciones contextuales.

4. Ventana de avisos del sistema

Aquí se muestra información sobre las siguientes acciones:

- *General*: Avisos de estado y de error
- *Generar*: Aviso de estado y de errores mientras se genera un proyecto
- *Transferencia*: Aviso de estados y de errores mientras se transfiere un proyecto
- *Visor del Portapapeles*: Avisos acerca del contenido del portapapeles.

5. Ventana de proyecto

Los datos de un proyecto de ProTool se depositan en forma de objetos. Los objetos están dispuestos dentro de un proyecto en una estructura de árbol.

En la ventana "Proyecto" se verán los tipos de objetos que pertenecen al proyecto y los que se pueden configurar para la unidad de operación seleccionada. La ventana de proyectos se puede comparar con la del explorador de Windows. Los tipos de objetos contienen objetos con propiedades ajustables.

La ventana de proyectos está estructurada del siguiente modo:

- La línea del título contiene el nombre del proyecto.

- En la mitad izquierda de la ventana se visualizan, dependiendo de la unidad de operación, los tipos de objetos configurables, en la mitad derecha los objetos generados.

Para el caso específico del panel táctil TP270 de 6" de SIEMENS; los tipos de objetos configurables son los siguientes:

Imágenes. En este objeto se almacenan las pantallas del equipo HMI. Desde aquí se puede editar y configurar las visualizaciones del equipo SIMATIC.

Avisos. Para advertir los avisos y alarmas producidos, en este objeto se crea un listado de todas las posibles alarmas que se puedan producir. Para su control, en el objeto Punteros de área, se ha de asociar los avisos de alarmas con un DB (Bloque de Datos) para de esta forma establecer una comunicación con el autómata que lo controla.

Recetas. Una receta es una agrupación de variables para formar una estructura de datos fija. La estructura de datos configurada puede ocuparse con datos en la unidad de operación y se designa como registro de datos. La utilización de recetas asegura que al transferir un registro de datos, lleguen al control todos los datos asignados, conjuntamente y de forma sincrónica. En nuestro proyecto no será necesario la utilización de recetas.

Archivos. Un archivo es un área de memoria en un medio de memoria (una tarjeta de memoria por ejemplo). El tamaño del archivo se define en ProTool. Se usará para almacenar avisos o variables, según nuestras necesidades. Una de sus aplicaciones podría ser la de obtener un histórico con las alarmas producidas. En nuestro proyecto, no se ha hecho uso de archivos.

Protocolos. En este objeto se utiliza para configurar cómo ha de ser la impresión de diversos datos de procesos que nos interesen (avisos, variables, etc..). Para un protocolo se define el contenido, el layout y el evento para el que se debe activar la impresión del protocolo. En nuestro caso concreto, no se dispone de impresora alguna, por lo que no se hará uso de protocolos.

Alertas. Las alertas son alarmas cíclicas que definen un momento que se repite a intervalos regulares en el que se debe ejecutar una determinada función (también se

puede no ejecutar ninguna función y lo único que se realiza es la activación del bit de alerta asociada a la variable configurada a ella).

Hay disponibles los siguientes tipos de alarma cíclica:

- diaria
- diario
- semanal
- anual

En nuestro proyecto, no ha sido necesaria la utilización de alertas.

Variables. En este objeto se especifican las variables que se usarán en el entorno HMI. A cada variable se les asociará una variable de control del autómata, esto es, una variable definida previamente dentro de un Bloque de Datos (DB) del programa software del autómata, siempre y cuando se desee que la variable utilizada esté controlada por el PLC. Se pueden usar variables sin control, para el funcionamiento interno del propio HMI.

Variables multiplex. Son un tipo de variables especiales. Con las variables multiplexadas se consigue modificar una gran cantidad de variables en función del valor de una variable de índice. En nuestro proyecto no se ha hecho uso de este tipo de variables, por lo que no entraremos en muchos más detalles.

Guiones. En este objeto se pueden programar pequeñas funciones (script) que sean de utilidad para el manejo del programa del HMI. Estos programas se pueden utilizar, entre otras, las variables sin control que se definieron en el objeto *Variables*. Los guiones deben ser traducidos correctamente para poder ser posteriormente compilados.

Curvas. En este menú se configuran los gráficos de curvas, asociado a alguna variable, que deseemos representar en alguna de las pantallas del HMI. En nuestro proyecto no se ha hecho uso de ningún grafico de curvas, por lo que no será necesario su utilización.

Gráficos. En este objeto se editan y se insertan los diversos gráficos e imágenes que deseemos utilizar para las pantallas del HMI. Los gráficos a insertar pueden editarse con

ayuda de programas externos. En nuestro caso, muchas de las imágenes utilizadas han sido creadas con el programa MICROSOFT VISIO.

Lista de símbolos. Un texto o un gráfico es a menudo más expresivo que los valores abstractos. Así, p. ej. los textos *lleno* y *vacío* o dos símbolos gráficos ilustran el estado de un tanque de manera más clara que los valores numéricos correspondientes. A tal fin, ProTool dispone de la posibilidad de configurar listas de símbolos. Las listas de símbolos son listas de textos o de gráficos en las que se les puede asignar a cada valor de una variable un elemento de una lista.

Controles. En este objeto se especifica el PLC que se va a usar, indicando el puerto por el cual se va a comunicar, el protocolo y la CPU utilizada. Para el caso de nuestro proyecto, se ha utilizado el protocolo SIMATIC S7 300-400 V6.0, y la comunicación se realiza con la CPU 315-2 DP por PROFIBUS.

Punteros de área. A través de un puntero de área se activa un área de direccionamiento definida en el control, la cual sirve para el intercambio de datos con la unidad de operación. Así pues, aquí se asocian diversos punteros de área a los Bloques de Datos (DB) que nos interese, para poder establecer comunicación entre PLC y HMI.

Existen varios tipos de punteros de área: Acuses OP, Acuses PLC, Alarmas, Avisos de servicio, Buzón de datos, Buzón de órdenes, Coordinación, Fecha/Hora, Número de la imagen, PLC-Fecha/Hora, Solicitud de curvas, Transferencia curva1, Transferencia curva2 y versión de usuario.

Para nuestro proyecto, se ha hecho uso de tan sólo 1 puntero de área:

- Avisos de servicio. Se ha asociado con el DB 91 del proyecto, que como veremos más adelante, es el que se ha denominado en la tabla de símbolo como “dbAlarmas”, y se usará para controlar las alarmas y avisos producidos en el sistema.

3.3.2. Pantallas del TP270

En este apartado, ahondaremos en detalle en las pantallas realizadas para nuestro equipo HMI.

Como se ha dicho anteriormente, en el objeto imagen es donde se configuran las pantallas que se visualizarán. Aquí es donde se relaciona todo lo que se inserta en el resto de objetos que componen el proyecto.

La pantalla base que servirá para crear el prototipo de pantalla se define en la opción del menú "SistemaDestino" → "Imagen/Teclas". Mostramos a continuación la pantalla base:

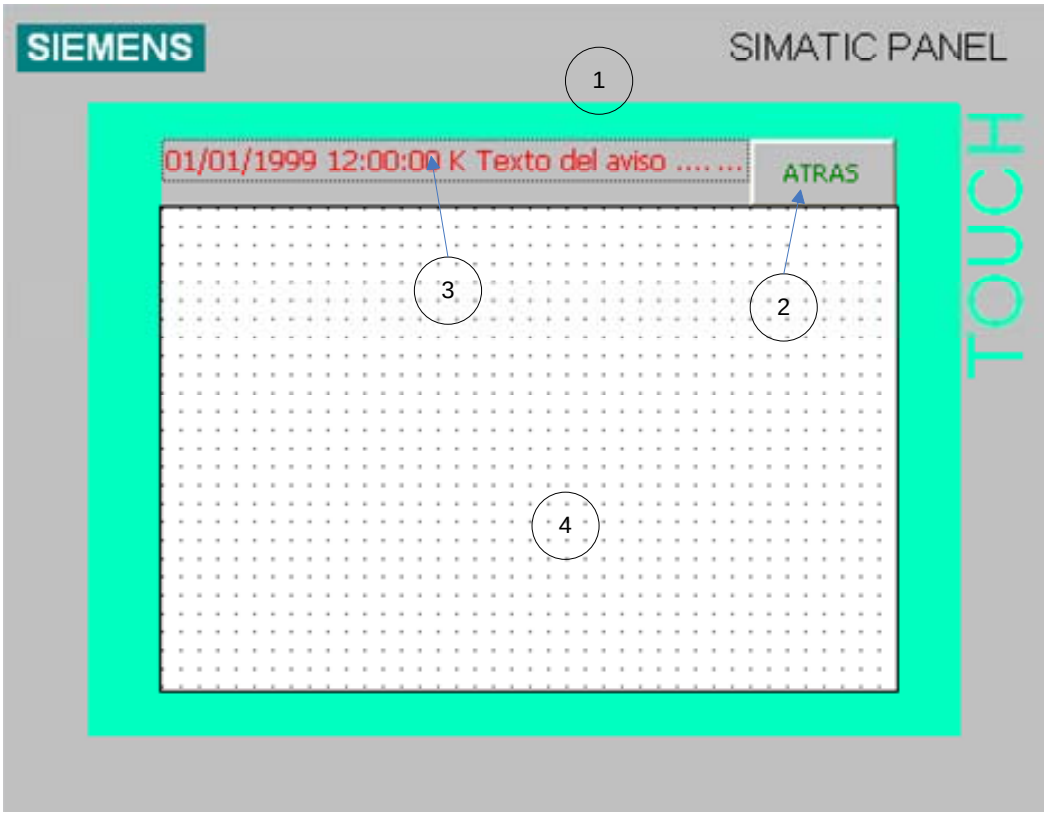


Figura 3.5. Pantalla base del HMI

Se advierten 4 zonas diferenciadas:

La zona 1 (en gris y verde), corresponde al borde de la pantalla. Esta zona es por tanto no editable, pues pertenece a la propia estructura del HMI.

La zona 2, corresponde a un botón con un texto en su interior “ATRÁS”. Se le ha asignado una función al mismo, para que cuando sea pulsado, se pueda volver a la pantalla anterior.

La zona 3 se corresponde con textos de avisos sencillos. El texto que aparece en esta zona anunciará el último aviso activo que se produjo en el sistema HMI.

Y por último, la zona 4 es l el área básica, que contendrá la zona editable, donde se diferenciarán cada una de las pantallas del programa HMI.

Estructura de las pantallas

Una vez comentado lo anterior, pasamos a mostrar la estructura general de las pantallas que conforman el proyecto:

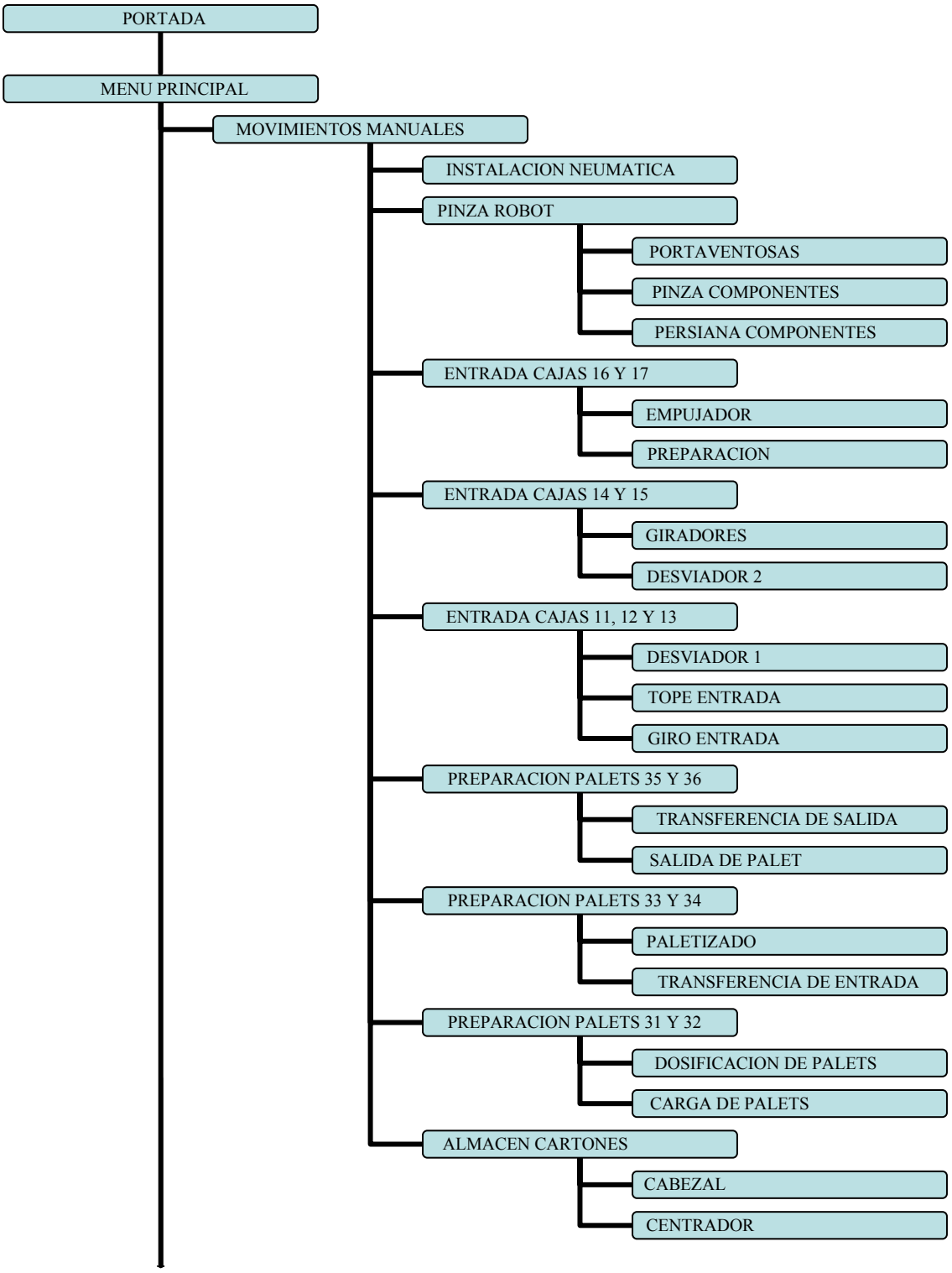


Figura 3.6. Estructura de las pantallas HMI 1

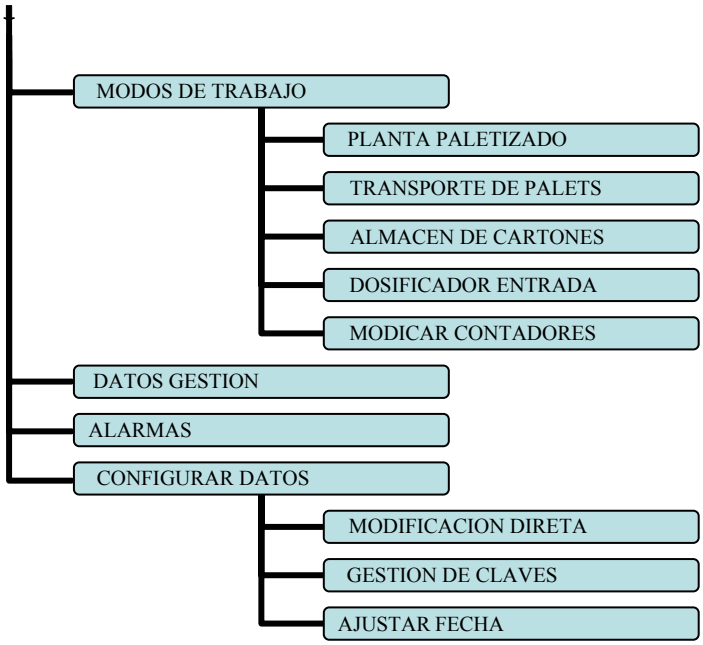


Figura 3.7. Estructura de las pantallas HMI 2

Mostramos a continuación cada una de las pantallas.

Pantalla de presentación (portada)

Esta pantalla es con la que se inicia la aplicación. Consta de un único botón que nos conduce al menú principal. Se puede observar el título del proyecto y un reloj que marca la hora.

Figura 3.8. Pantalla de Presentación

Pantalla de menú principal

Con esta pantalla accedemos a las distintas funcionalidades de la aplicación. Consta de:

Movimientos Manuales, Modos de Trabajo, Datos de Gestión, Alarmas y Configurar Datos. Aparte se ha creado un botón de Rearme, cuya función es la de reponer los bits (poner los bits a 0) "REARME", "REARMAR_BARRERA1" y "REARMAR_BARRERA2" que se encuentran dentro del DB 94 denominado "dbParámetros", que se ha utilizado para configurar diversos parámetros del HMI. Con el botón de Salir Contraseña, podrá salir el usuario, que previamente se había registrado introduciendo su contraseña, del sistema.

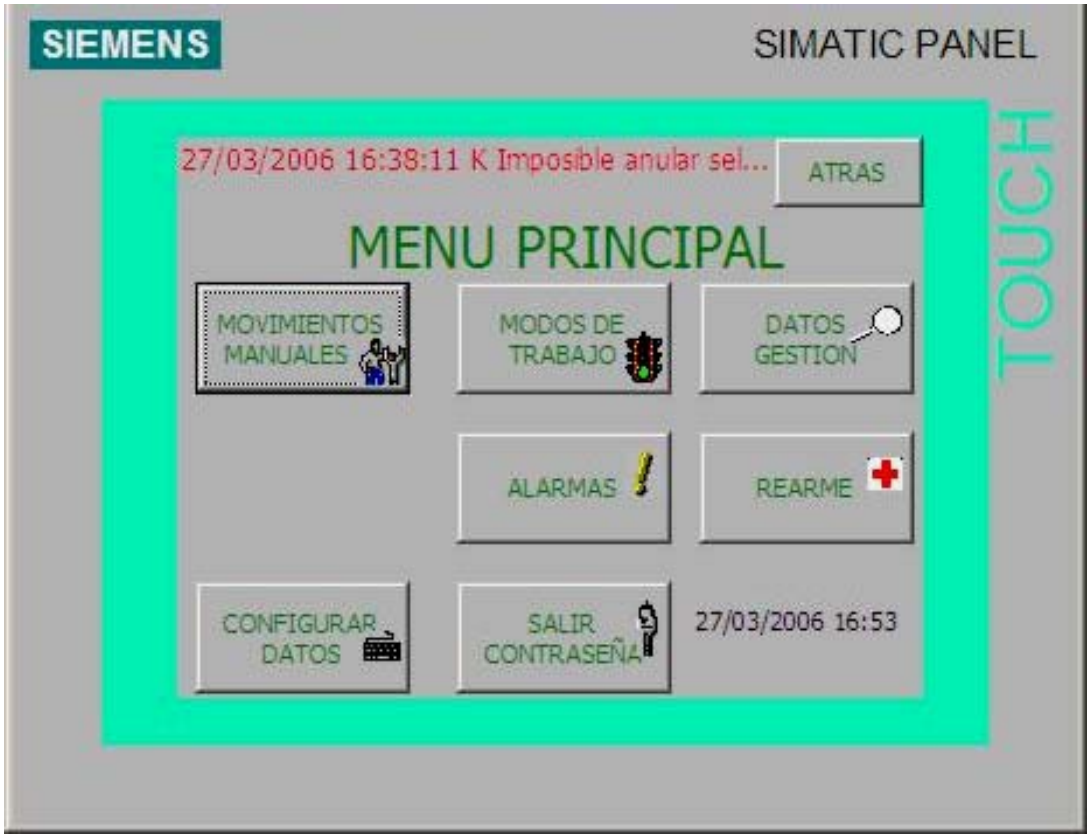


Figura 3.9. Pantalla de Menú Principal

Pantalla de zonas movimientos manuales

En esta pantalla se presenta un sinóptico de la planta con cada una de las zonas representadas por colores para poder llevar a cabo movimientos manuales. A la izquierda y derecha se encuentran los botones para acceder a la zona en detalle que se desee trabajar.

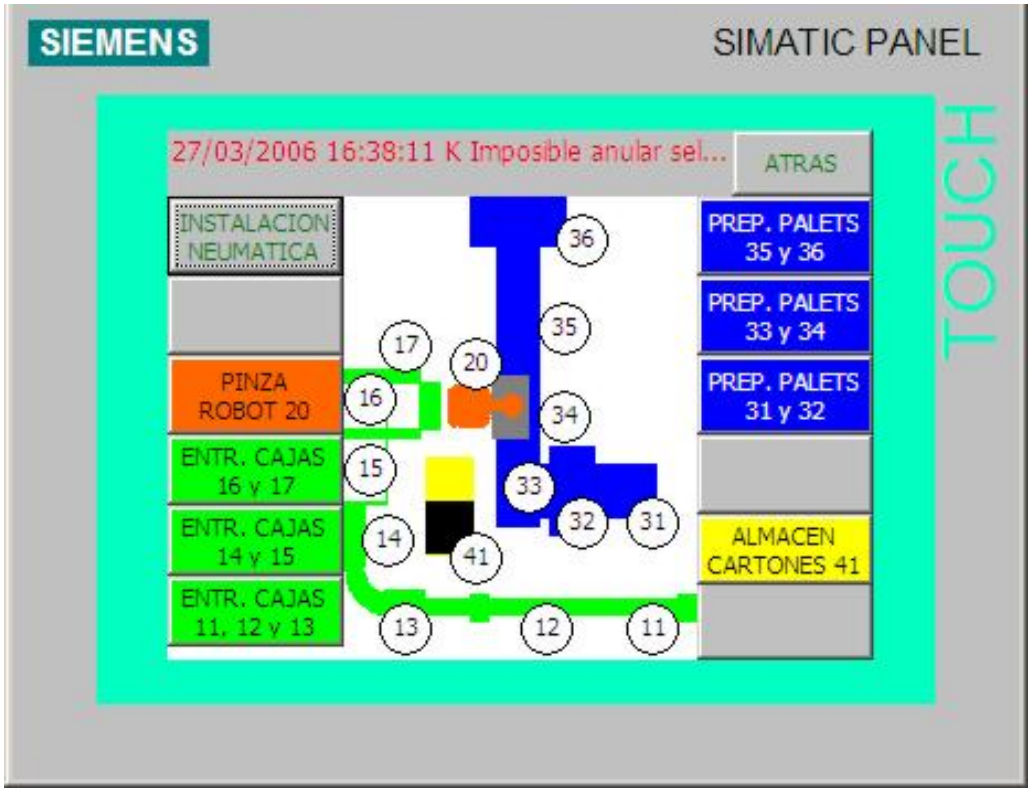


Figura 3.10. Pantalla de Zonas de Movimientos Manuales

Instalación Neumática

Acciones manuales sobre las válvulas de corte de la instalación neumática. Desde esta pantalla podremos activar o desactivar las válvulas de corte de la zona neumática 1 (salida digital A155.0), de la zona neumática 2 (salida digital A353.2) y la válvula de corte TV4 del Almacén de Cartones (salida digital A353.3). Como siempre, lo que se hace realmente desde pantalla es resetear los bits bm155_00, bmA353_02 y bmA353_03 que se encuentran dentro del DB 93 denominado dbManuales, donde registramos los movimientos manuales que se desean realizar con la TP270. Será en el propio autómata donde se leerán estos bits y se actuará propiamente en consecuencia.

Por otra parte, desde esta pantalla se podrá visualizar el estado de los presostatos de la zona neumática 1 y 2 (entradas digitales E105.0 y E303.2 respectivamente). Para ello, se ha asociado a los iconos adjuntos las señales E105_00 y E303_02 respectivamente, que se encuentra en el DB 92 denominado "dbVisualización".

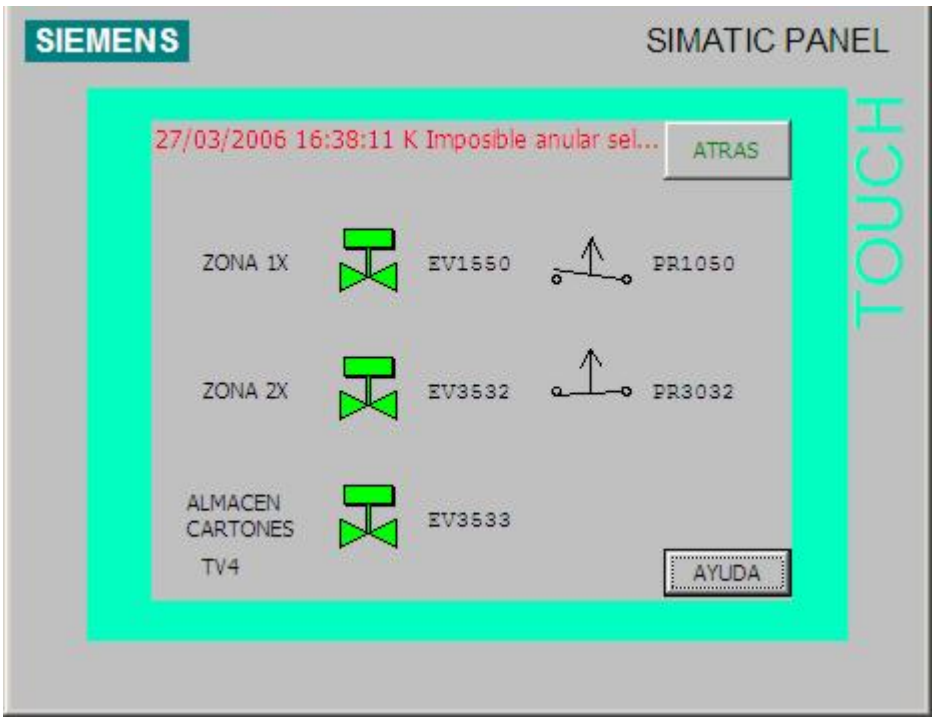


Figura 3.11. Instalación Neumática

Pinza Robot 20

En esta pantalla accedemos a las distintas partes de la pinza del robot para realizar movimientos manuales.

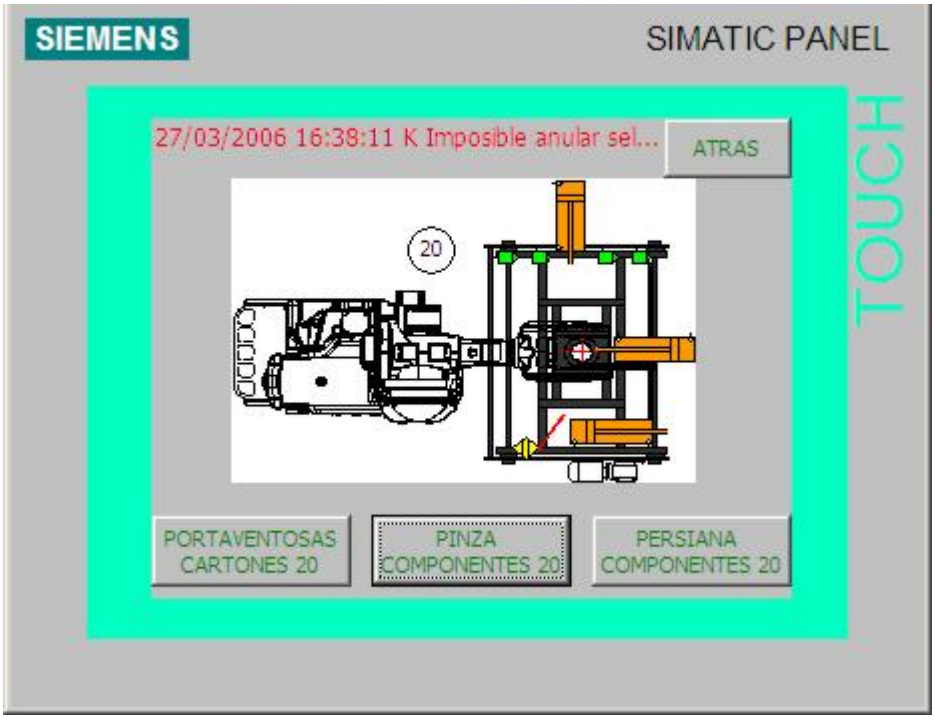


Figura 3.12. Pinza Robot

Portaventosas Cartones 20

Acciones manuales para el portaventosas de la pinza del robot. Disponemos la posibilidad de abrir y cerrar las pinzas (accionando los cilindros mediante pulsanción de las flechas verdes) y generar vacío pulsando sobre los iconos que están al lado de las señales EV2522 y EV2523. Además en la pantalla se nos muestra la información de si se ha realizado el vacío (SQ2022 y SQ2023) y si las pinzas están abiertas o cerradas (mediante los indicadores de los fotodetectores SQ2021 y SQ2020). Recordar nuevamente que en realidad, desde el HMI no se manejas las señales directamente, sino que se activan o desactivan diversos bits, en este caso del DB 93 (dbManuales), que son controlados por el autómata. Así, para la señal EV2522 se regula el bit bmA252_02.

Para las visualizaciones de los fotodetectores se observan las variables E202_00, E202_01, E202_02, E202_03, E205_0 y E205_01 del DB 92 "dbVisualización".

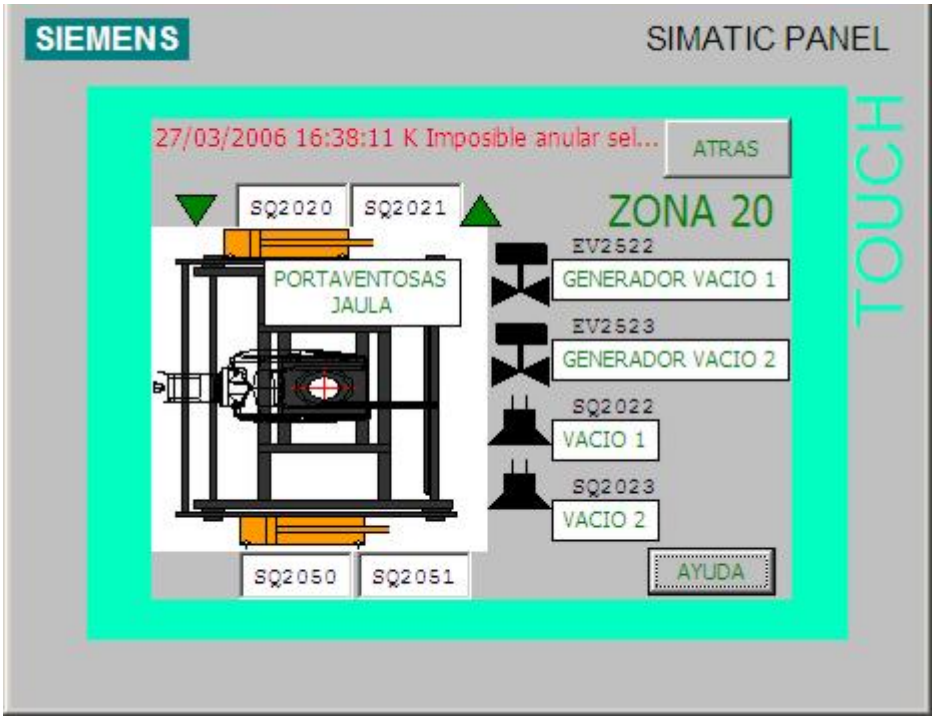


Figura 3.13. Portaventosas Cartones

Pinza Componentes

Acciones manuales sobre los centradores y la tajadera lateral de la pinza del robot. Desde aquí se accionan todos los cilindros y se visualizan su posición y la señal que produce el fotodetector SQ2040.

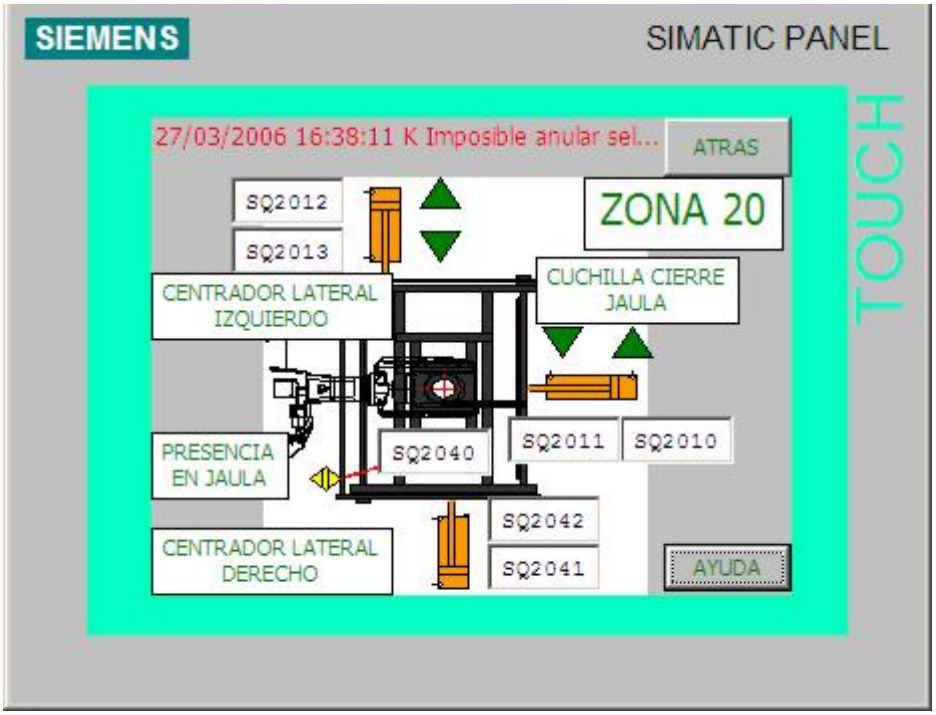


Figura 3.14. Pinza Componentes

Persiana Componentes 20

Acciones manuales sobre el motor de la persiana de la pinza, así como el estado de los detectores inductivos de la misma. En este caso, se visualizan los detectores y el estado del motor (encendido o apagado). El estado del motor se visualiza observando la variable A10_04, del dbVisualización. Para ponerlo en marcha, se presiona sobre las flechas verdes, las cuales activan el bit bmVM21D o bmVM21I (si se quiere que el motor gire en un sentido o en otro), que se encuentra en el DB "dbManuales". También se puede visualizar la velocidad del motor, con la ayuda de guiones. Aunque esto se ha dejado como una posible mejora y no se ha llegado a implementar.

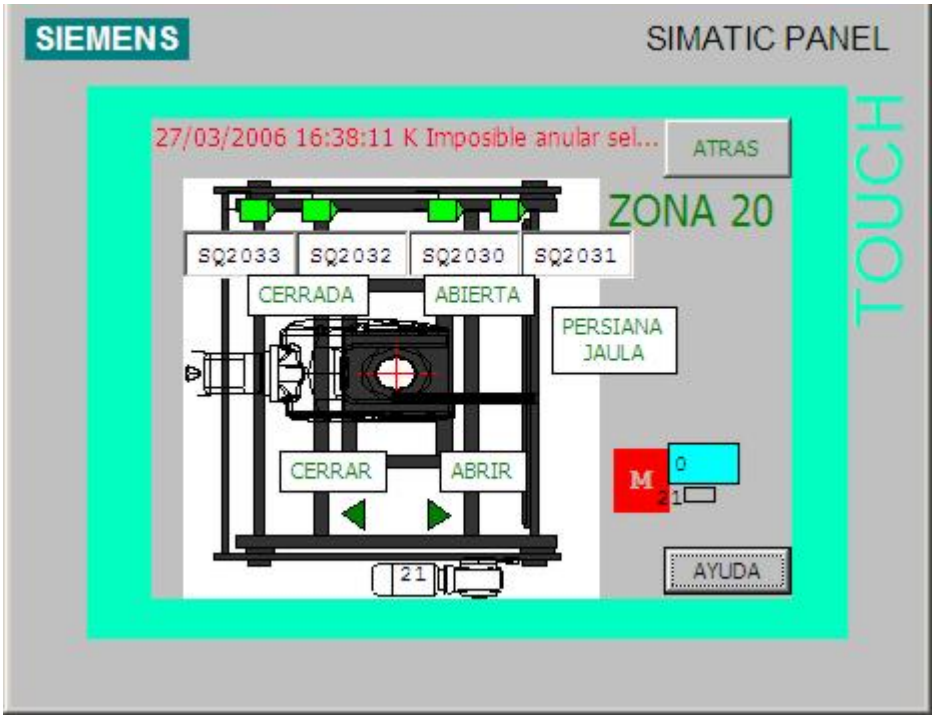


Figura 3.15. Persiana Componentes

Entrada Cajas 16 y 17

Acciones manuales sobre la mesa de formación.

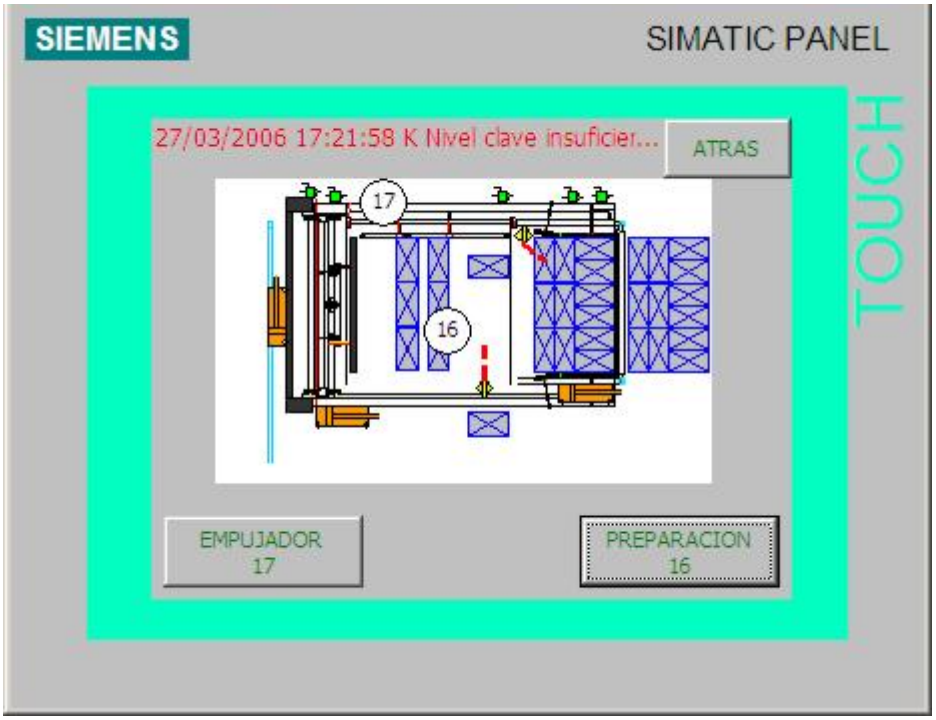


Figura 3.16. Entrada Cajas 16 y 17

Empujador

Acciones manuales sobre el motor del empujador, así como el estado de los detectores inductivos de las distintas posiciones del mismo. El funcionamiento interno es igual que en las pantallas anteriores.

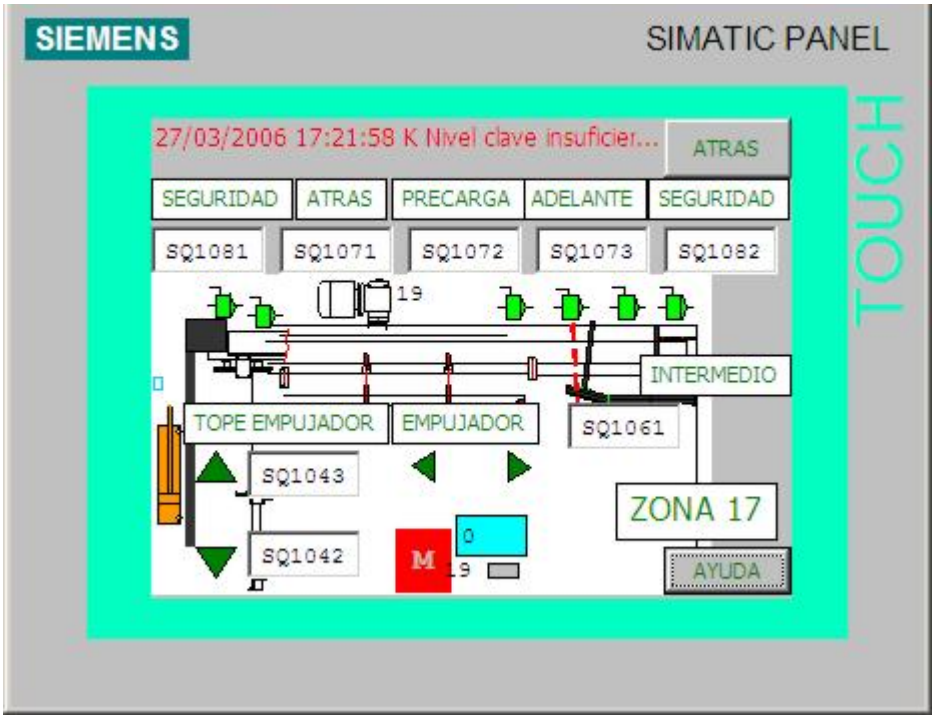


Figura 3.17. Empujador

Preparación

Acciones manuales de las tajaderas de entrada y salida de la mesa, así como el estado de las distintas fotocélulas de presencia.

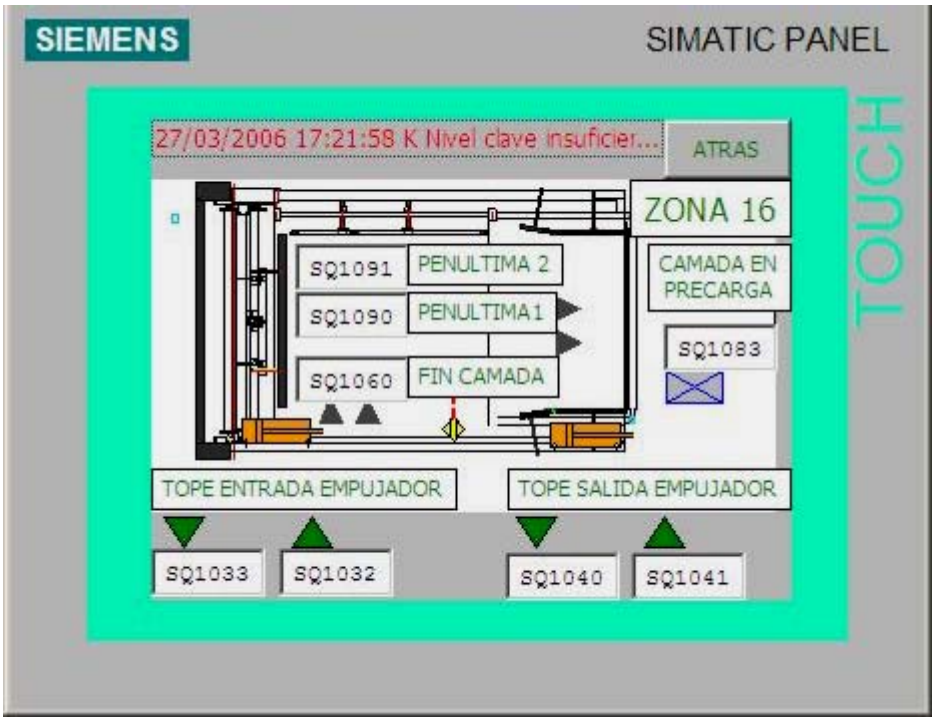


Figura 3.18. Preparación

Entrada Cajas 14 y 15

Acciones manuales sobre los giradores de cajas y sobre el segundo desviador.

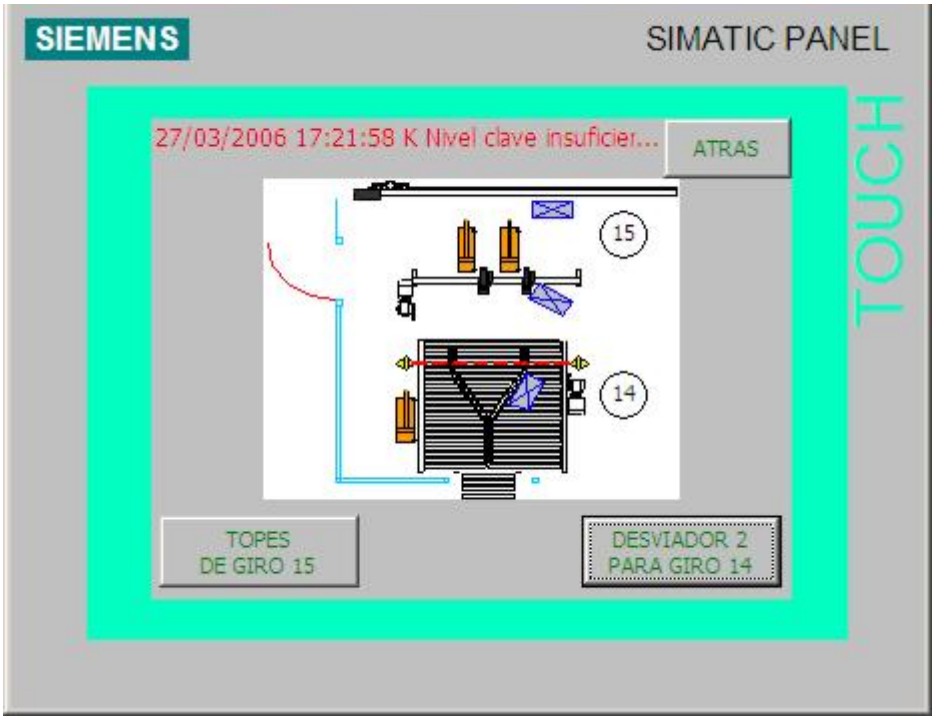


Figura 3.19. Entrada Cajas 14 y 15

Topes de Giro

Acciones manuales sobre los topes de giro, así como sobre el motor de la mesa.

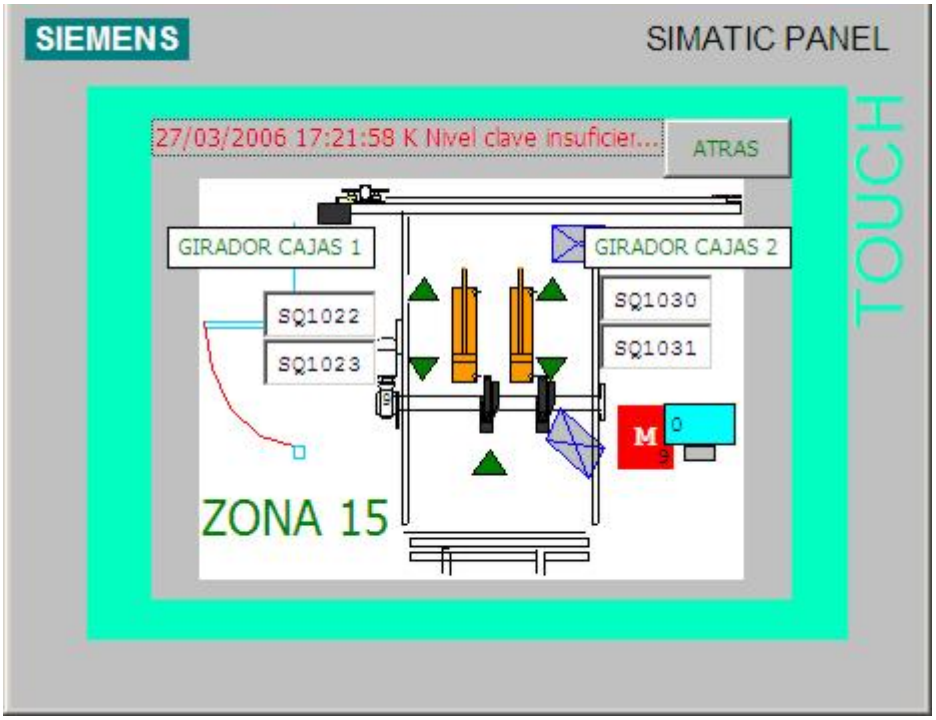


Figura 3.20. Topes de Giro

Desviador 2

Acciones manuales sobre el desviador 2, así como sobre el motor del camino de rodillos correspondiente.

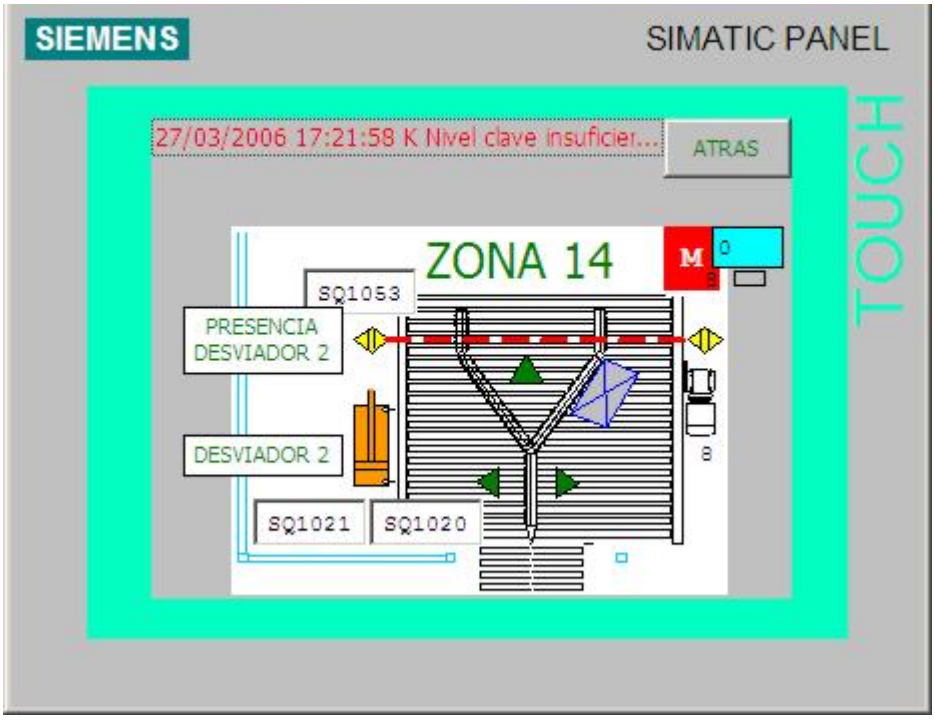


Figura 3.21. Desviador 2

Entrada Cajas 11,12 y 13

Accione sobre lo transportes de cajas a la entrada de la mesa.

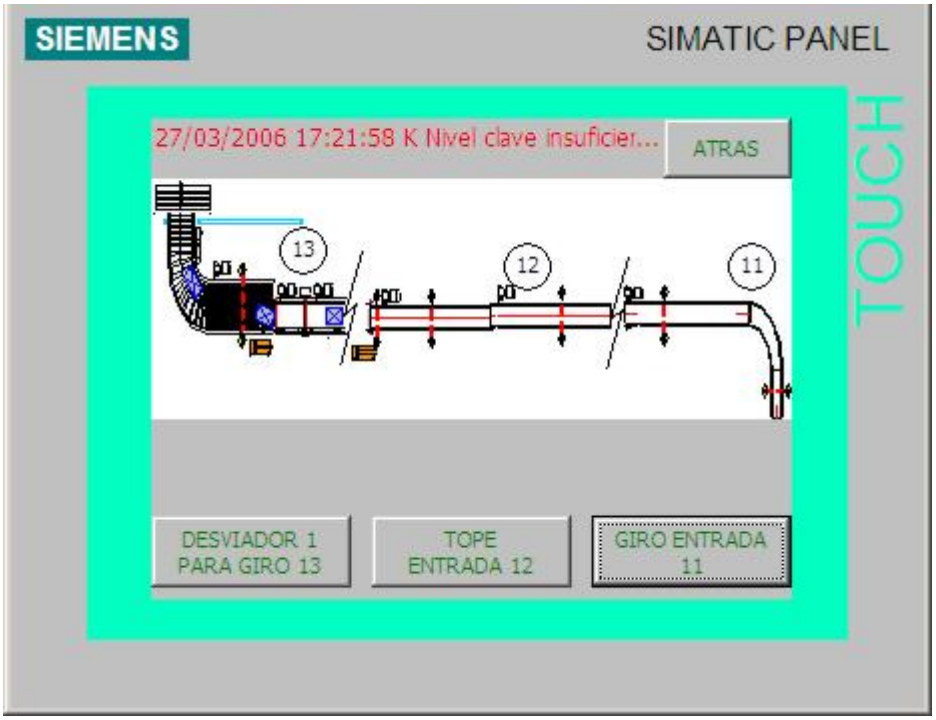


Figura 3.22. Entrada Cajas 11, 12 y 13

Desviador 1

Acciones manuales sobre el desviador 1, así como sobre los distintos motores de los caminos de rodillos implicados.

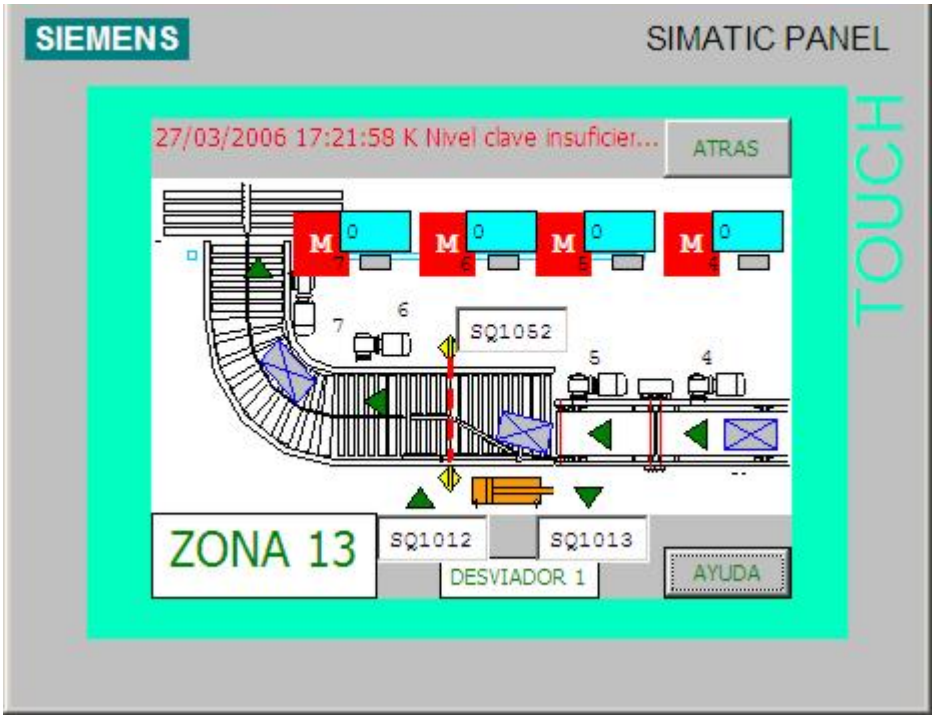


Figura 3.23. Desviador 1

Tope Entrada

Acciones manuales sobre el tope de entrada a la mesa, así como sobre los motores de los dos transportes de cajas.

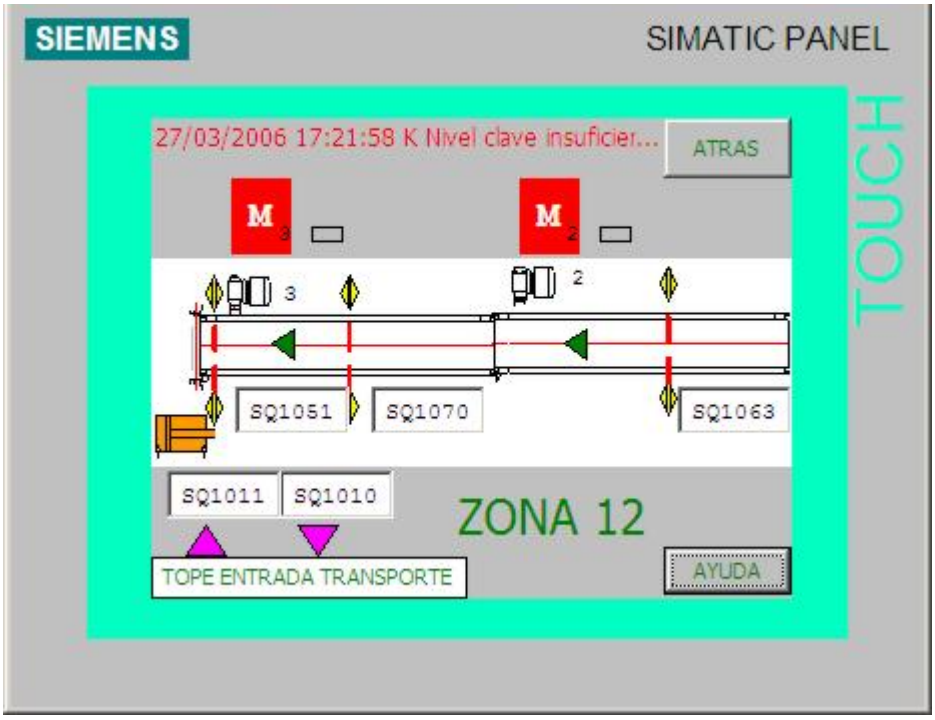


Figura 3.24. Tope entrada

Giro Entrada

Acciones sobre el motor del primer transporte de cajas, así como el estado de fotocélulas de saturación.

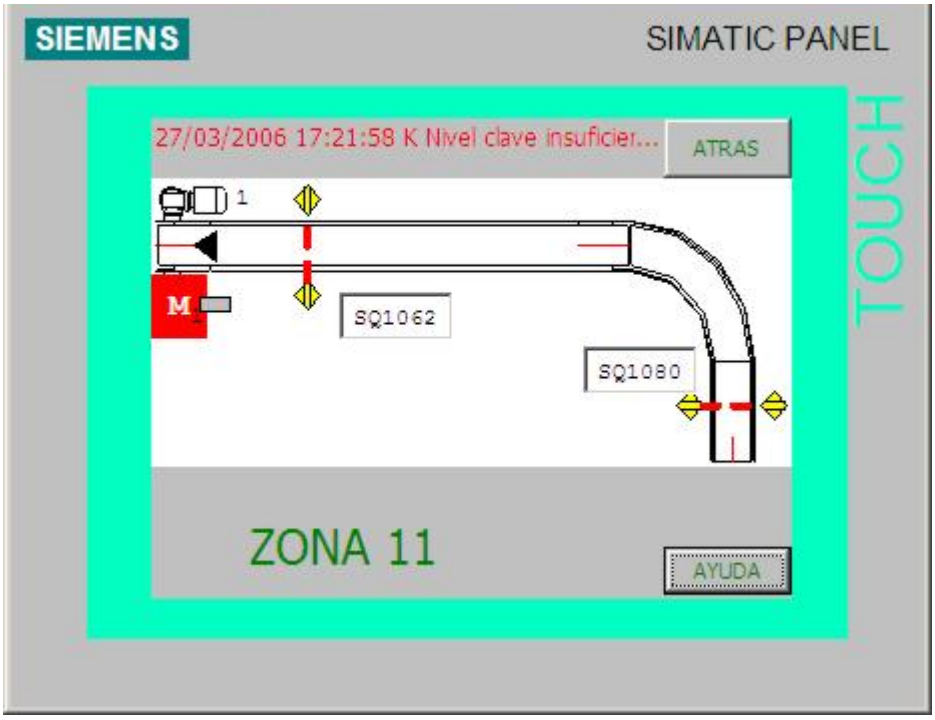


Figura 3.25. Giro Entrada

Preparación Palets 35 y 36

Acciones sobre la segunda transferencia de palets.

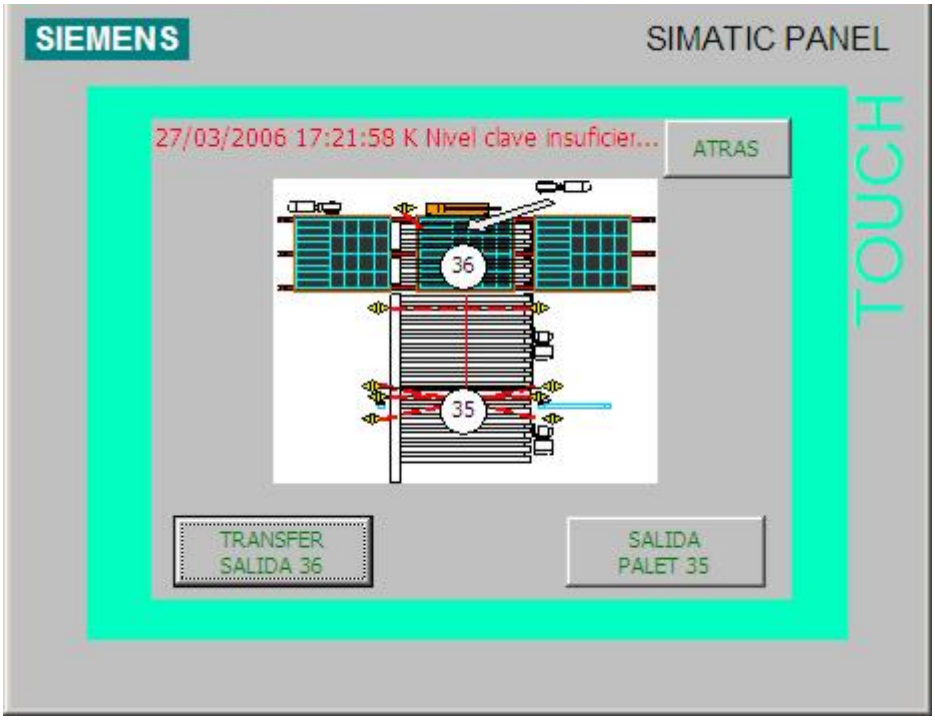


Figura 3.26. Preparación Palets 35 y 36

Transferencia de salida

Acciones sobre los motores de la transferencia de salida, así como con el cilindro neumático y el estado de los detectores.

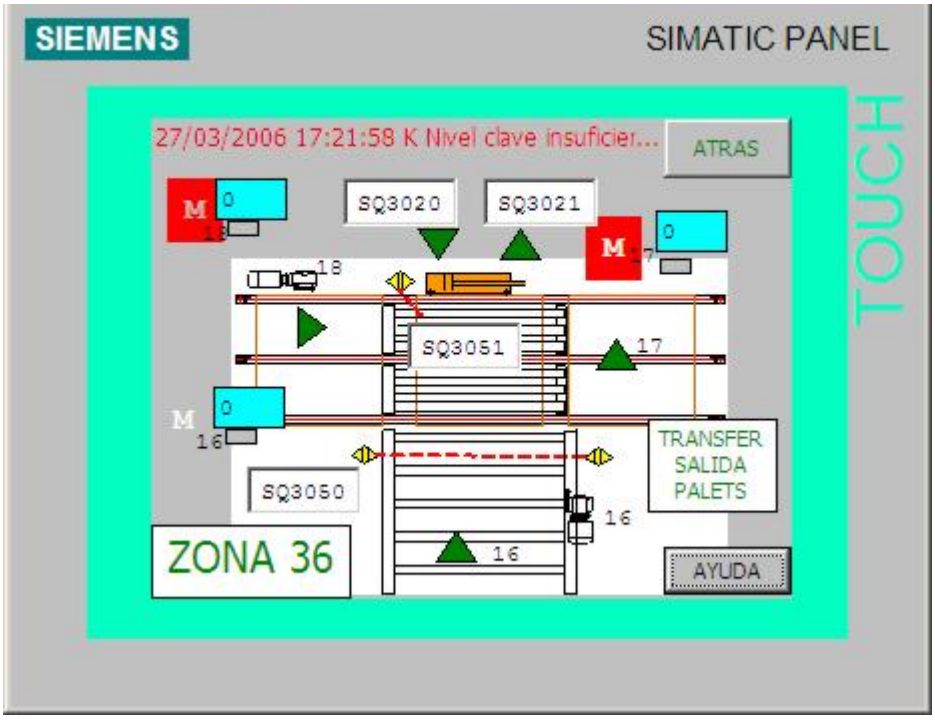


Figura 3.27. Transferencia de Salida

Salida de palet

Acciones sobre los motores de los transportes de palet y de las fotocélulas de presencia de palet.

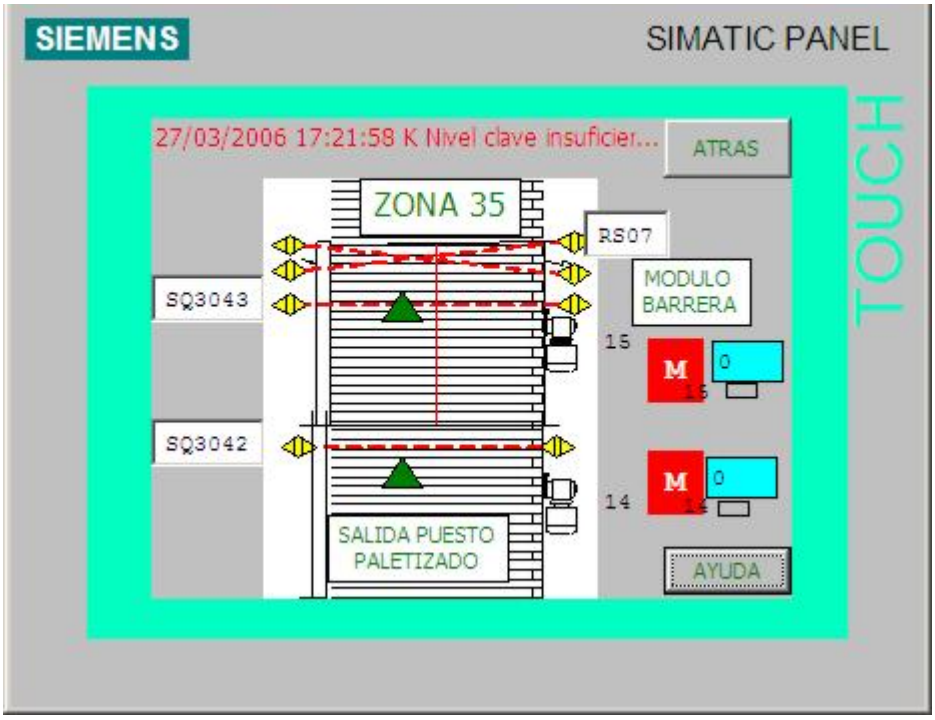


Figura 3.28. Salida de Palet

Preparación Palets 33 y 34

Acciones manuales sobre la transferencia de entrada y la zona de paletizado.

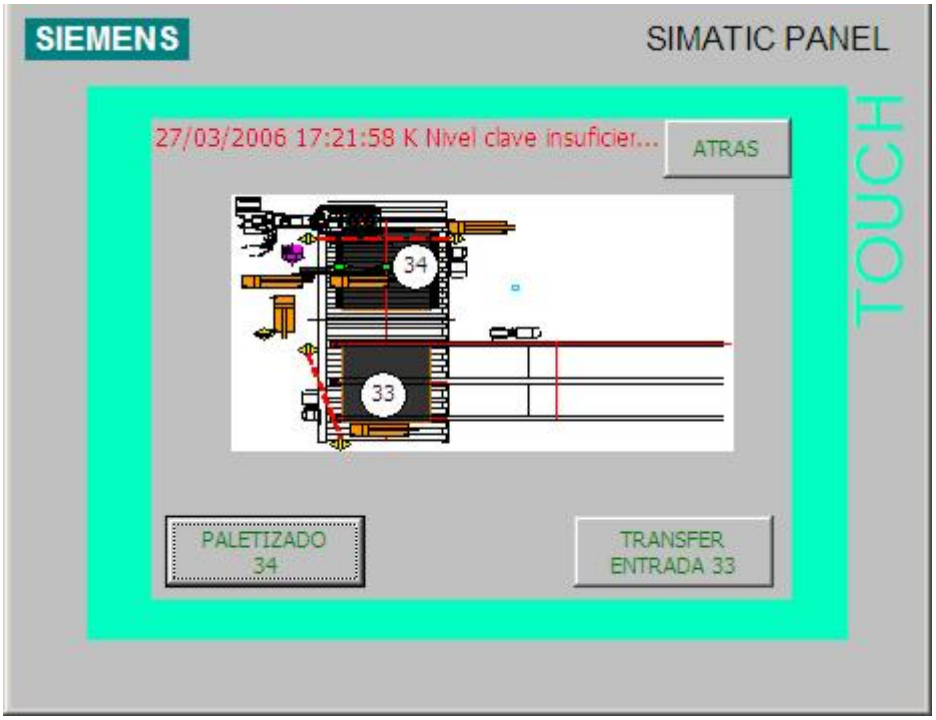


Figura 3.29. Preparación Palets 33 y 34

Paletizado

Acciones sobre el motor, el tope y el centrador de la zona de paletizado.

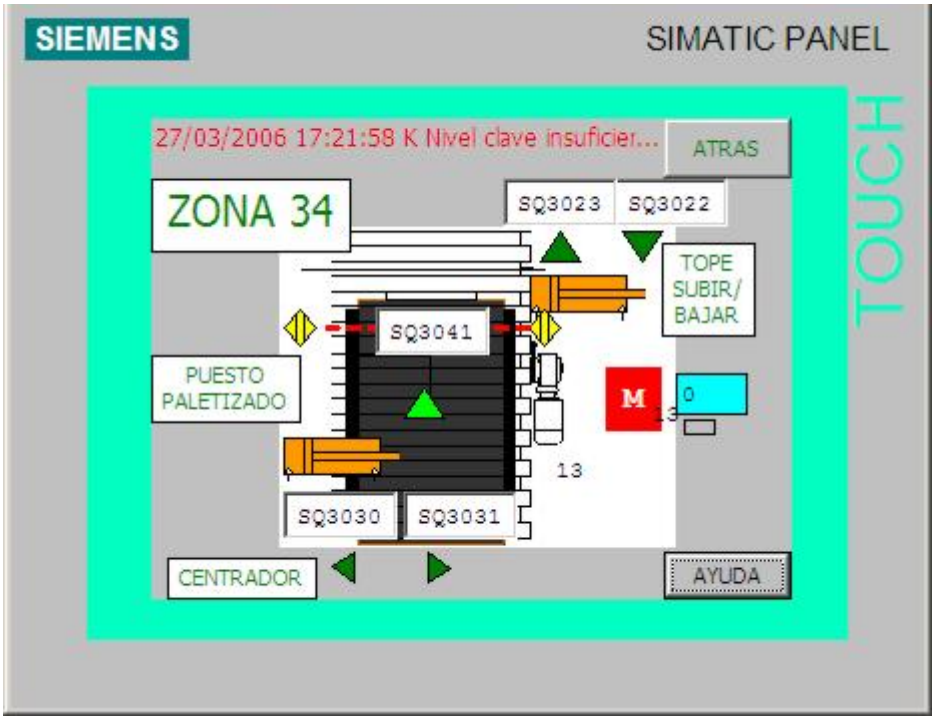


Figura 3.30. Paletizado

Transferencia de entrada

Acciones sobre los motores y el cilindro neumático de la transferencia de entrada.

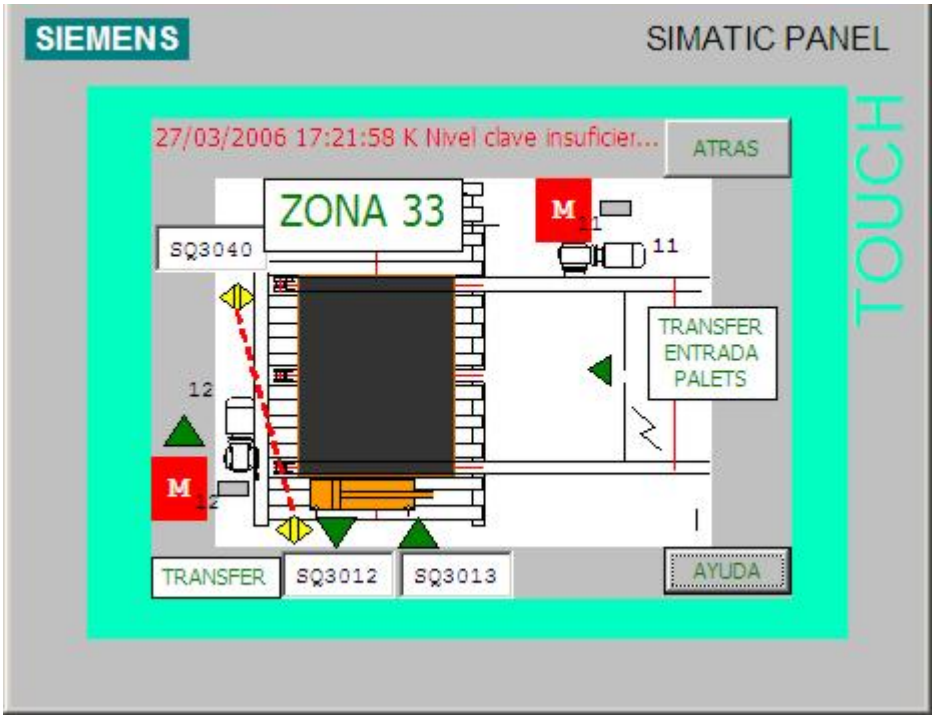


Figura 3.31. Transferencia de entrada

Preparación Palets 31 y 32

Acciones manuales sobre el dosificador de palet y la carga de los mismos en el almacén.

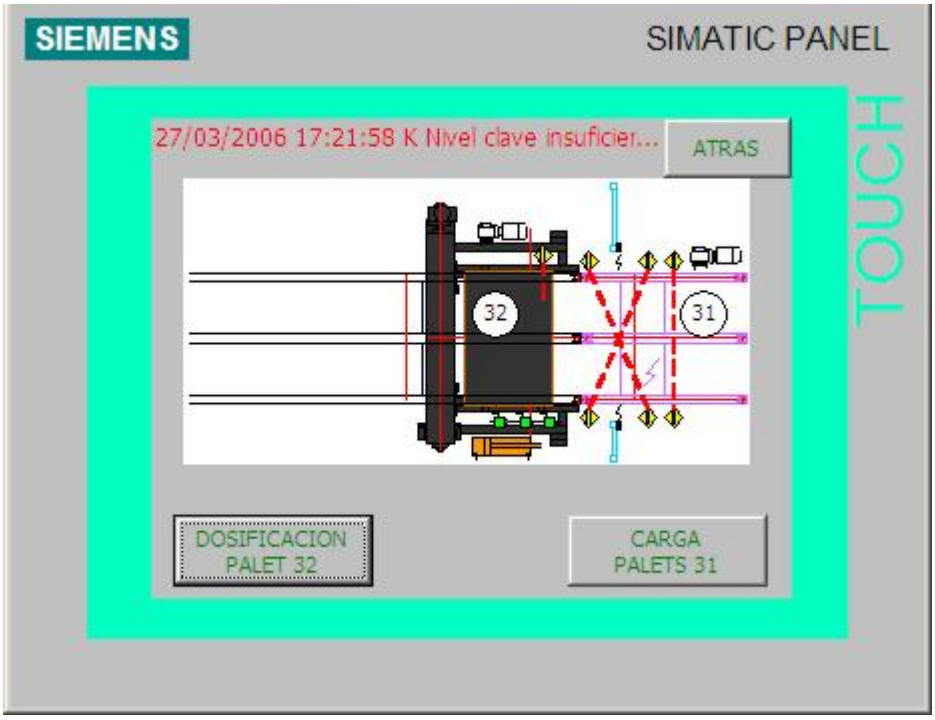


Figura 3.32. Preparación Palets 31 y 32

Dosificación palet

Acciones manuales sobre los elementos del almacén de palets.

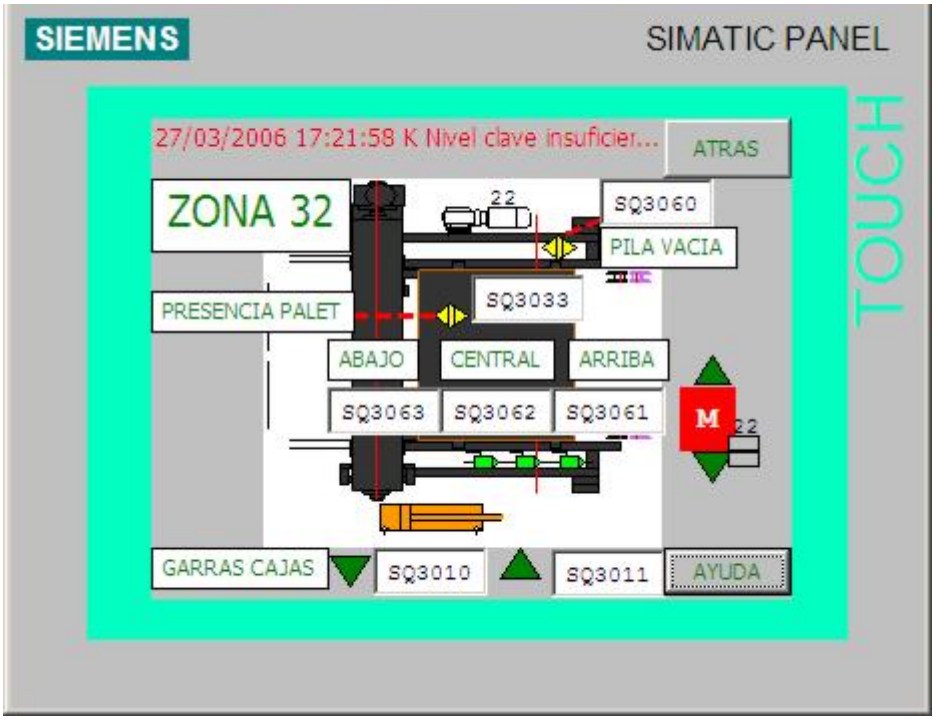


Figura 3.33. Dosificación de Palets

Carga de palets

Acciones sobre el motor y la fotocélula del transporte inicial de carga de palets en el almacén.

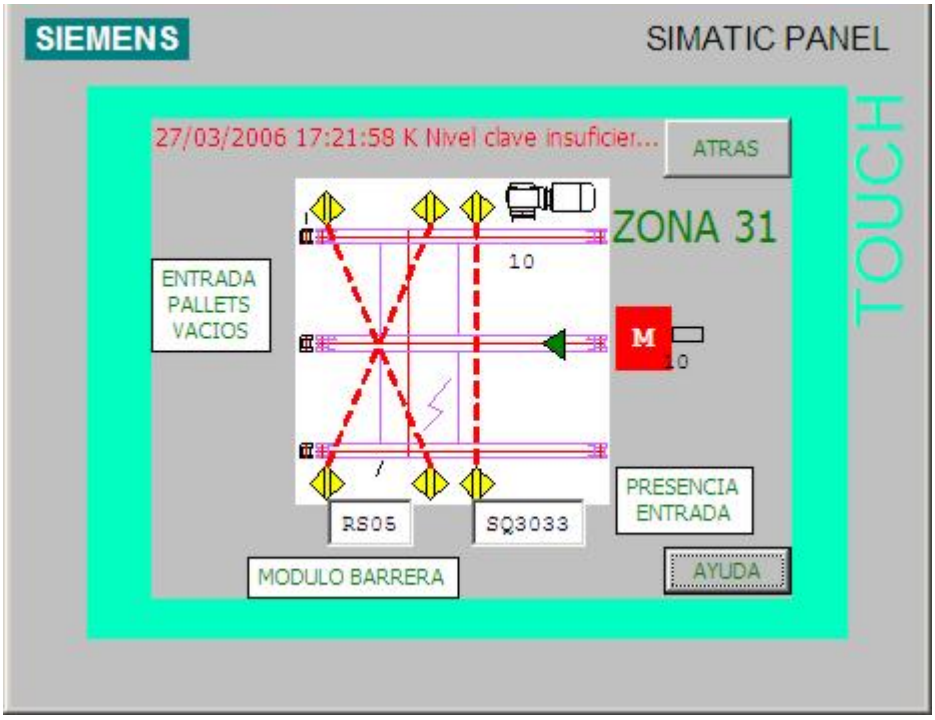


Figura 3.34. Carga de Palets

Almacén de Cartones

Acciones manuales sobre el almacén de cartones.

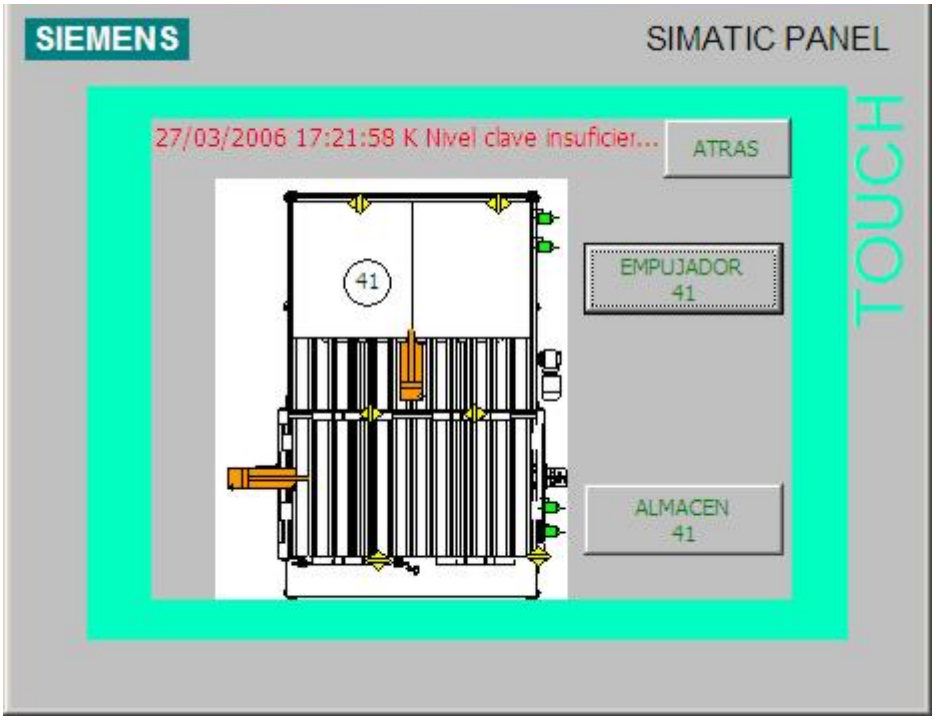


Figura 3.35. Almacén Cartones

Cabezal del almacén de cartones

Acciones manuales sobre el cabezal del empujador de cartones y el motor de arrastre.

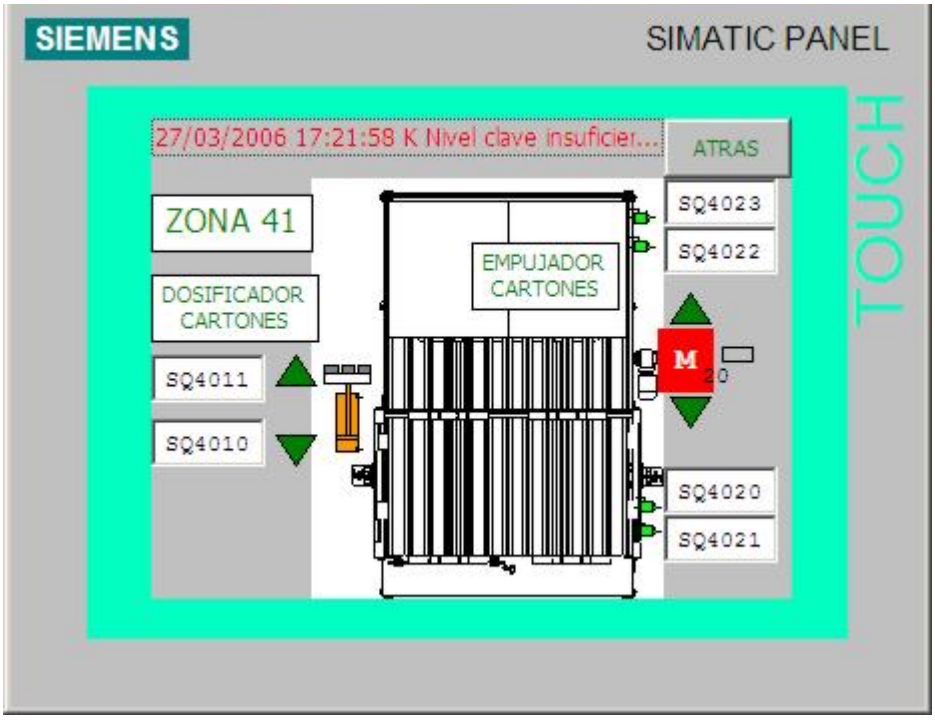


Figura 3.36. Empujador de Cartones

Centrador del almacén de cartones

Acciones sobre el centrador de cartones del almacén y el estado de las fotocélulas de presencia.

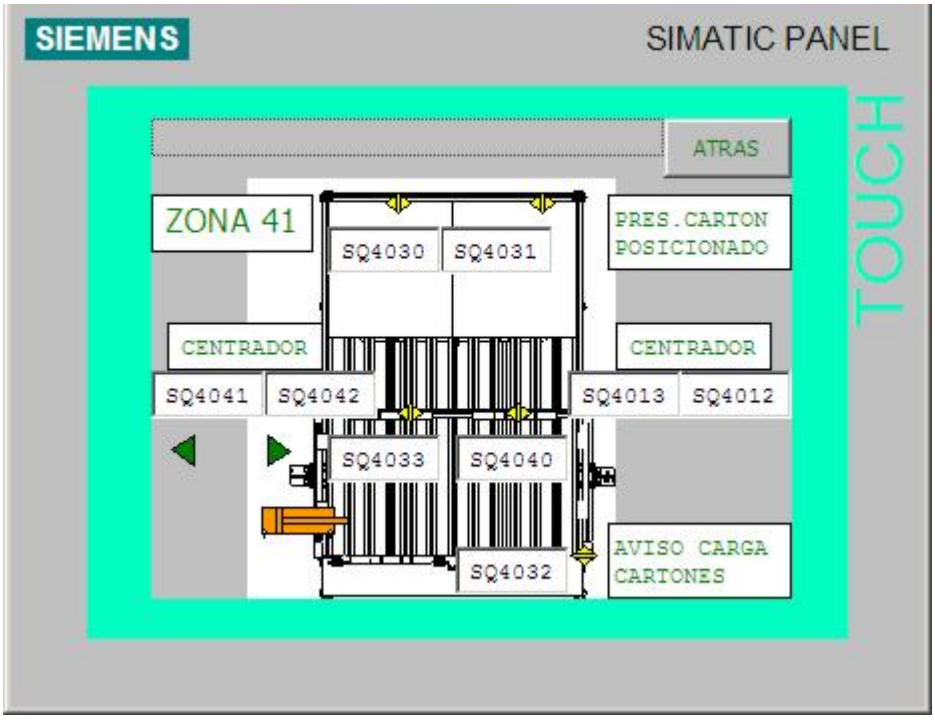


Figura 3.37. Centrador del almacén de cartones

Modos de trabajo

Desde esta pantalla accedemos a los distintos modos de funcionamiento de la máquina. Las distintas zonas de la planta de paletizado (ver **¡Error! No se encuentra el origen de la referencia.**) están agrupadas en tres secciones desde el punto de vista funcional de la siguiente manera:

- Planta Paletizado: Transportadores de bandas, Mesa de formación y Robot
- Transporte de palets: Almacén de palets y Transportes de rodillos.
- Almacén de Cartones: Almacén de Cartones

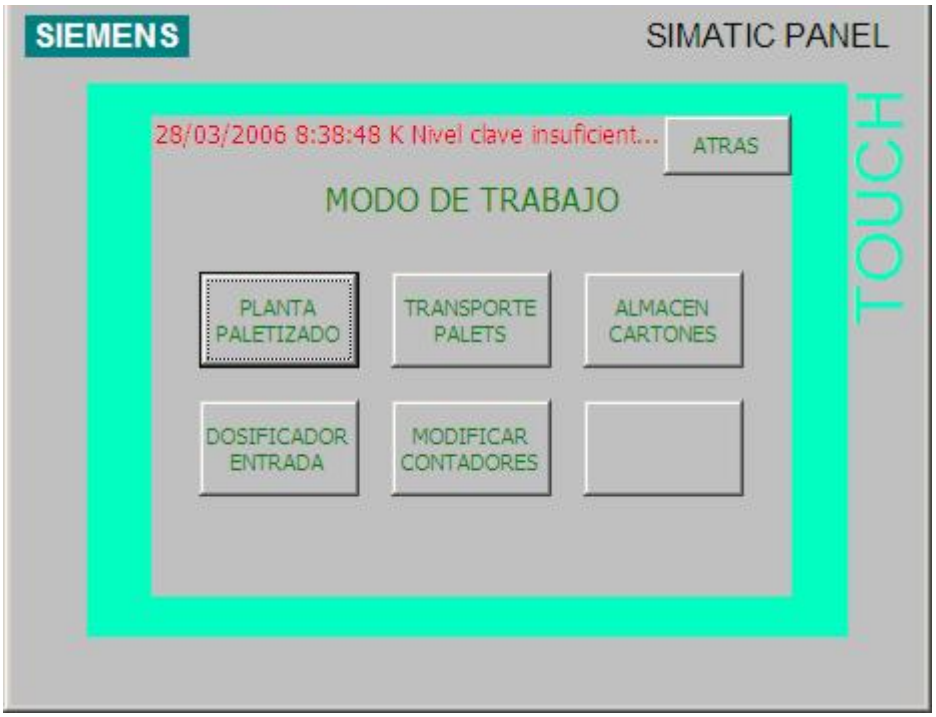


Figura 3.38. Modos de Trabajo

Planta de Paletizado

Cada una de las secciones consta de:

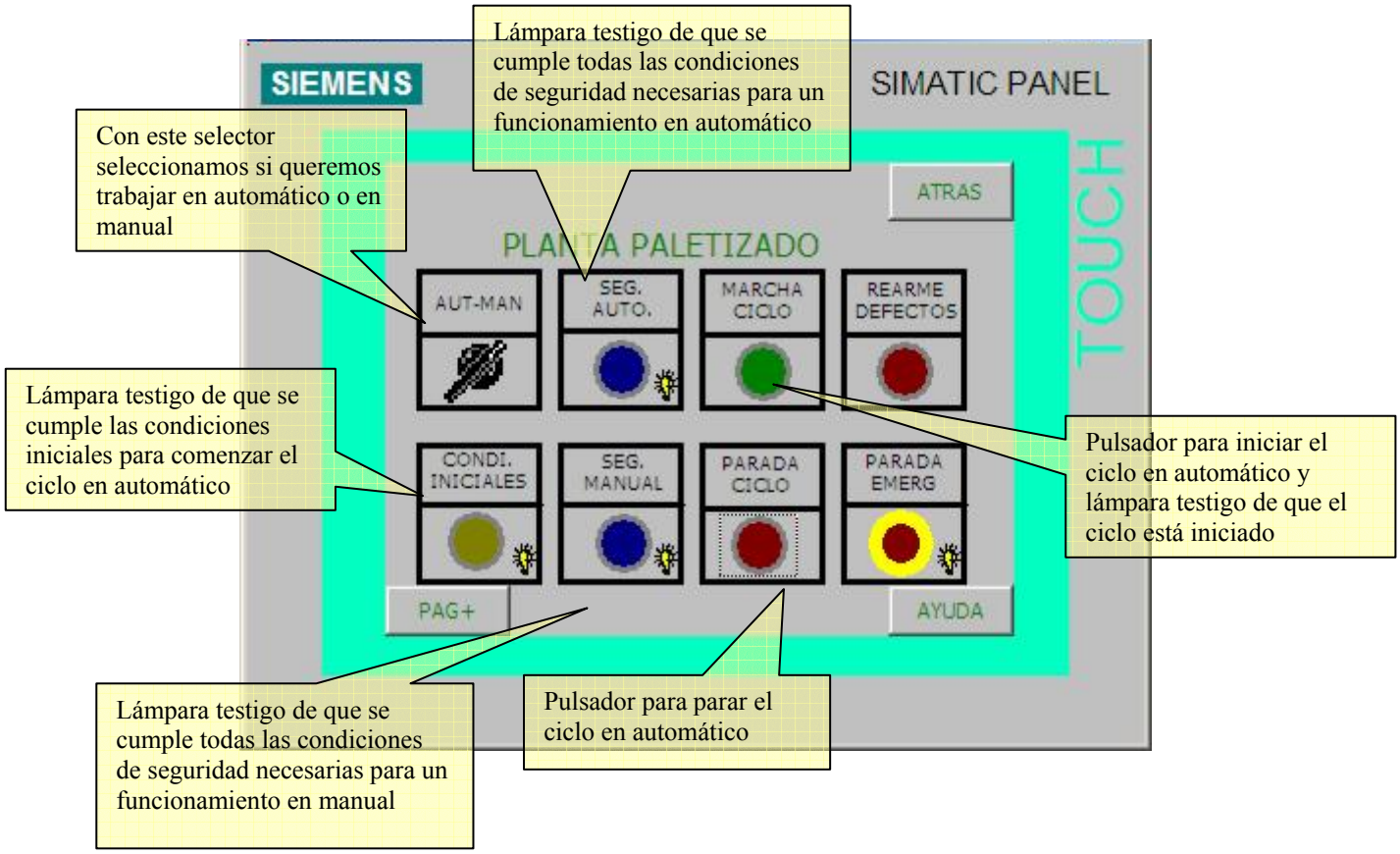


Figura 3.39. Planta de Paletizado

Las variables que manejan se encuentran dentro del DB 200 denominado "dbTP270_Modos". Como se indica en los recuadros, respectivamente con cada pulsador/selector se consigue lo siguiente:

- AUT-MAN: SELECCIÓN DEL MODO DE FUNCIONAMIENTO
- SEG. AUTO: INSTALACIÓN CON SEGURIDAD DE AUTOMATICO
- MARCHA CICLO: INCIAR CICLO DE PRODUCCION / INSTALACIÓN EN MARCHA (lámpara asociada en el mismo pulsador)

- REARME DEFECTO: REARME DE LA INSTALACION / INFORMACION DE QUE SE DEBE REARMAR LA INSTALACION (lámpara asociada en el mismo pulsador)
- CONDI. INICIALES: INSTALACIÓN EN CONDICIONES INICIALES
- SEG. MANUAL: INSTALACIÓN CON SEGURIDAD DE MANUAL
- PARADA CICLO: FINALIZAR CICLO DE PRODUCCION
- PARADA EMERG: INSTALACIÓN EN PARADA DE EMERGENCIA

Transporte de Palets

Igual que el punto anterior pero para la sección Transporte de Palets.

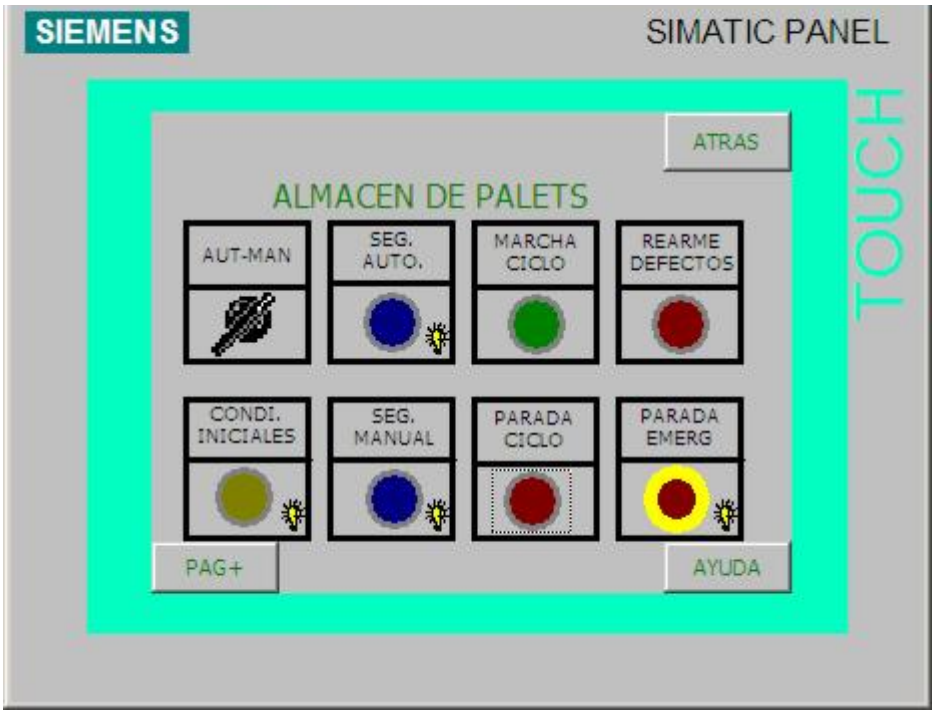


Figura 3.40. Transporte de palets

Liberación de palet

La sección Transporte de Palets consta de una pantalla adicional donde con un pulsador podemos liberar el palet independientemente del número de capas completadas. Para ello se modifica el bit bLiberarPalet, que se encuentra en el DB 5 denominado "dbMemorias".

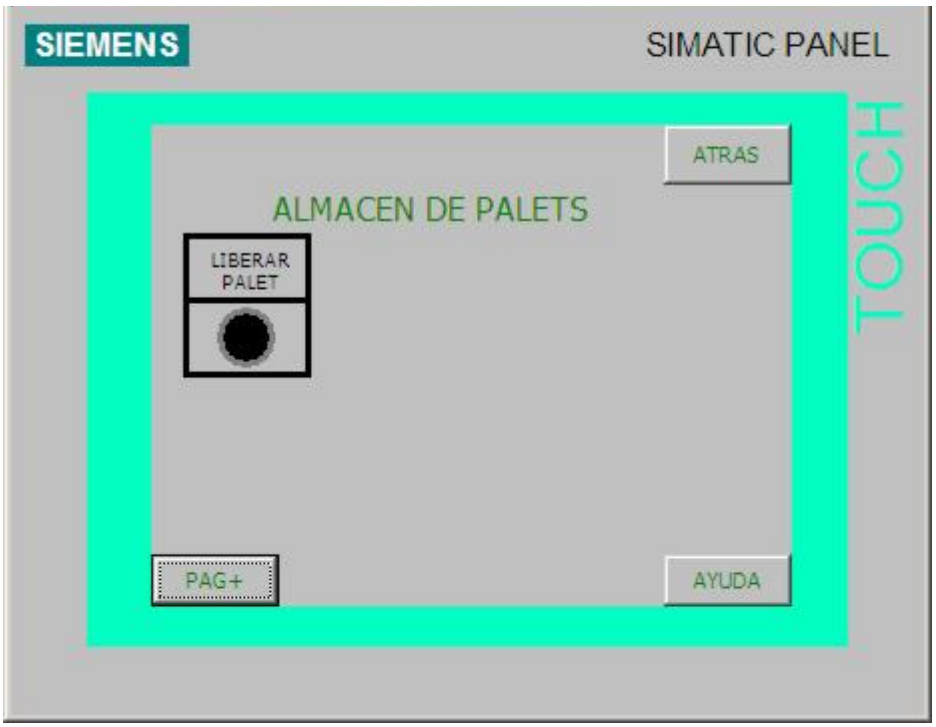


Figura 3.41. Liberación de Palet

Almacén de Cartones

Igual que el punto 3.6.1 pero para la sección del Almacén de Palets.

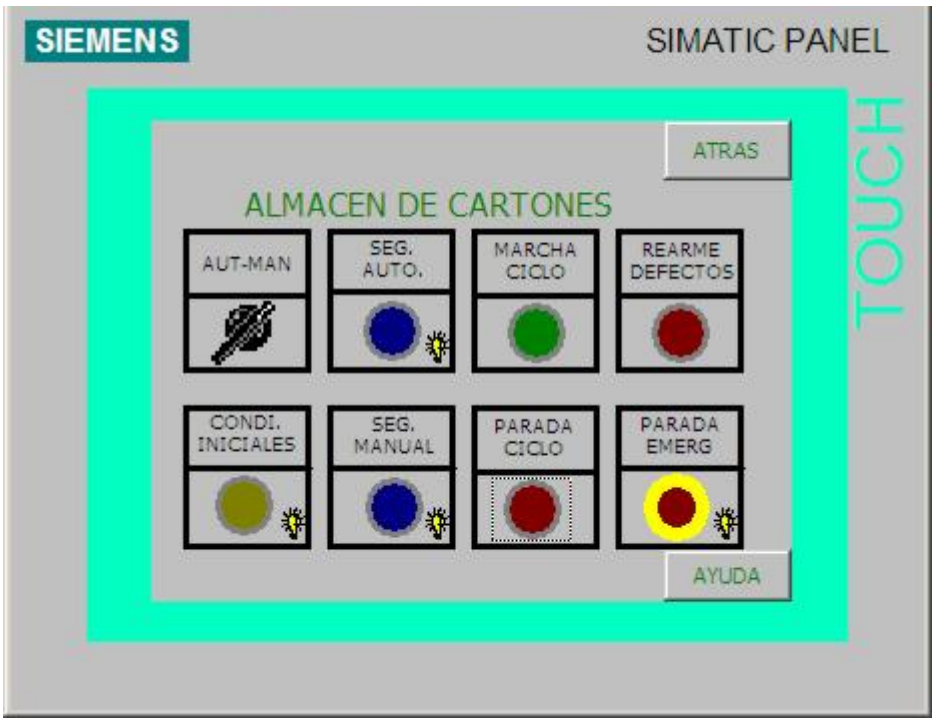


Figura 3.42. Almacén de Cartones

Dosificador de Entrada

Desde esta pantalla podemos controlar el número de cajas que queremos dejar pasar hacia la mesa de formación. Disponemos de un selector donde elegimos si queremos trabajar de forma continua o en modo paso a paso. En este último modo el número de cajas que pasarán a la mesa de formación depende del valor que introduzcamos en el campo de edición, siempre hay que validarlo con el pulsador dosificar.

Figura 3.43. Dosificador de Entrada

Para comunicar al autómata que se desea pasar al modo paso a paso, se modifica el bit bPasoAPaso del “dbMemorias”. Para la validación, se modifica el bit bDosificar, también del “dbMemorias”. El número de cajas a dosificar se almacenaría en la variable ncpp de tipo INT, que se encuentra en dbMemorias.

Modificar Contadores

En esta pantalla podemos modificar los contadores de los desviadores, de la mesa y la altura de dejada del robot a través de unos campos de edición. Esta pantalla esta protegida por contraseña. La contraseña por defecto es "100" queda a cargo del personal de mantenimiento la posibilidad de cambiarla, así como de crear otras nuevas. En la Figura 3.44. Pantalla de Introducción de Contraseña podemos ver el aspecto de dicha ventana. Para cambiar la clave inicial, se cambia dentro del menú "SistemaDestino" → "Ajustes…" → "Nivel superior", de SIMATIC PROOTOL/PRO CS.

Figura 3.44. Pantalla de Introducción de Contraseña

Figura 3.45. Modificar Contadores

Las variables que se controlan en este caso son las siguientes:

Altura Dejada: Variable "cn", de tipo INT, que se encuentra en el DB 63 denominado "dbRobot".
Contador Cajas Desviador 1: Variable "ccd1", de tipo INT, que se encuentra en el DB 60 "dbDesviador1".
Contador Cajas Desviador 2: Variable "ccd2", de tipo INT, que se encuentra en el DB 61 "dbDesviador2".
Contador Cajas Mesa: Variable "ccm", de tipo INT, que se encuentra en el DB 5 "dbMemorias".

Datos de Gestión

Desde esta pantalla podemos seleccionar el formato de cajas que se va a paletizar, el resto de campos no están operativos y se han incluido para futuras ampliaciones.

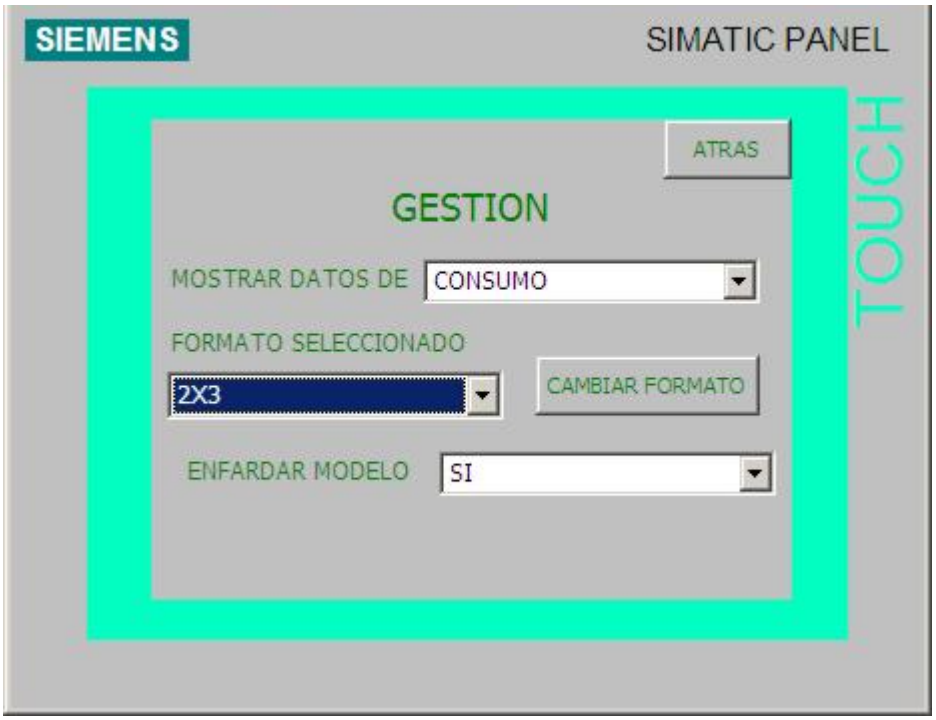


Figura 3.46. Datos de Gestión

Para las opciones posible, se ha hecho uso de Listas.

Para la opción "MOSTRAR DATOS DE", se ha hecho uso de la lista "PARÁMETROS". Cuyas opciones son las siguientes:

Valor	Texto
0	NO VALIDO
1	VELOCIDAD
2	ACELERACIÓN
3	DECELERACIÓN
4	CONSUMO

Tabla 3.2. Lista "PARÁMETROS"

Se le ha asignado la variable índice de control TIPOVISUALIZACIÓN, pero no se ha asociado a ningún DB por el momento, pues ésta sería una opción de ampliación de éste proyecto.

Para la opción "FORMATO SELECCIONADO", se ha hecho uso de la lista LST_FORMATOS. Las opciones posibles son:

Valor	Texto
1	2x2
2	2x3
3	2x4

Tabla 3.3. Lista "LST_FORMATOS"

La variable índice que controla es el bit iFormatoSeleccionado, que se encuentra en el "dbMemorias". Para confirmar, hay que pulsar el botón "CAMBIAR FORMATO", que modifica el bit bPeticiónCambiarFormato, que se encuentra en el dbMemorias.

Por último, para la opción de ENFARDAR MODELO, se ha hecho uso de la lista SI_NO, cuyas opciones son:

Valor	Texto
0	NO
1	SI

Tabla 3.4. Lista "SI_NO"

La variable índice que se ha usado es el bit bEnfardarModelo. Tampoco se le ha asociado ningún DB dejando esto como futura ampliación del proyecto.

Alarmas

En esta pantalla podemos ver un listado de las alarmas y avisos del sistema. Estas alarmas son de mucha utilidad sobre todo para el personal de mantenimiento. El operario también puede ver el estado de estas alarmas para ver si ha saltado un térmico y si no tenemos presión de aire, por ejemplo.

Figura 3.47. Pantalla de Alarmas del Sistema vacía

Mostramos otro ejemplo de lista de avisos:

Figura 3.48. Pantalla de Alarmas del Sistema con Avisos

Se advierte que el cilindro 14 (que regula la Transferencia de Palet 2) está retrocedido y que en el transportador de caja 3, hay presencia de cajas. En este caso, cuando esta situación desaparece, desaparece automáticamente de la lista de avisos.

Configurar Datos

En esta pantalla se pueden modificar distintos parámetros, esta pantalla está protegida por contraseña y queda restringido su uso sólo para el personal de mantenimiento. Desde ella se pueden modificar palabras del autómata, gestionar las claves, modificar la fecha y hora o salir de la aplicación.

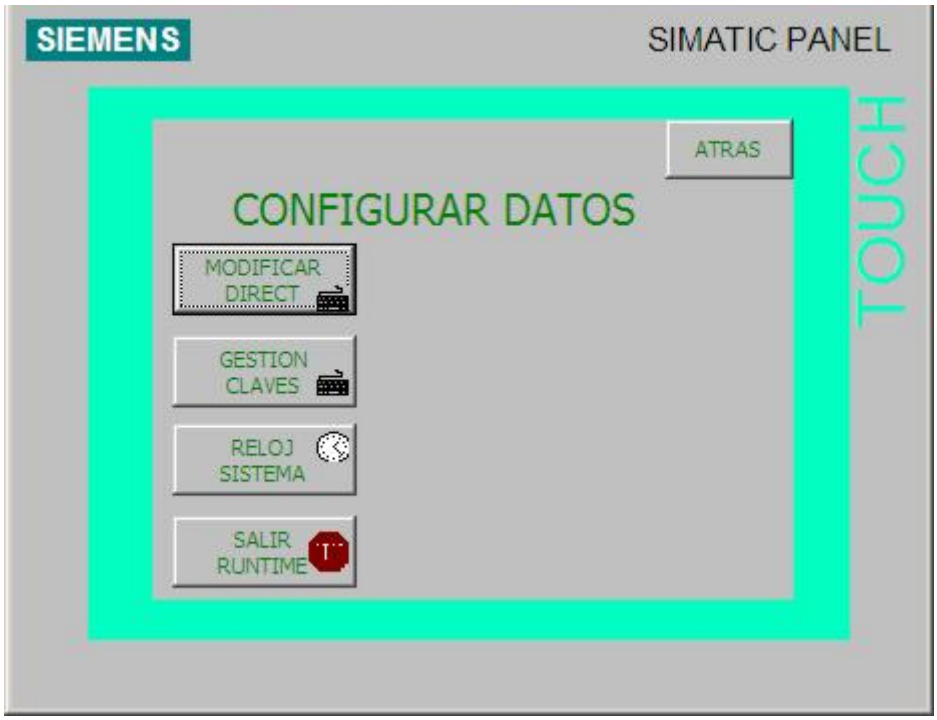


Figura 3.49. Configurar Datos

Modificación Directa

Desde esta pantalla podemos cambiar el valor de las palabras y las variables del autómata. Es imprescindible conocer el uso de estás palabras por el programa de autómata antes de cualquier modificación así como el propio programa pues un uso no adecuado de las mismas podría ocasionar daños en la instalación. Se disponen de botones para forzar valores a variables y visualizarlos.

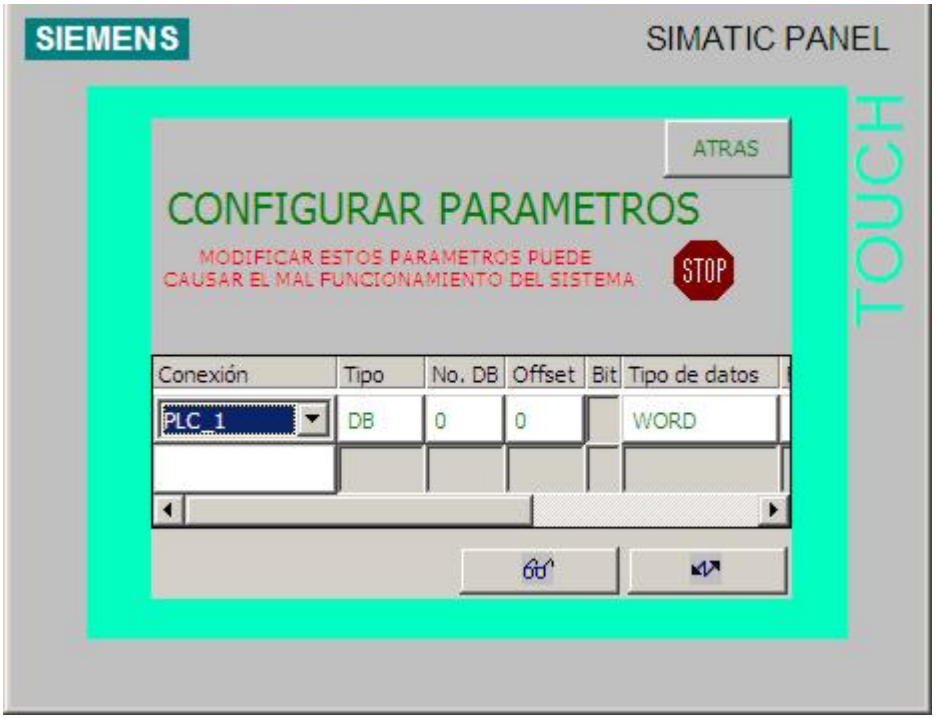


Figura 3.50. Modificación Directa

Gestión de claves

Desde esta pantalla se pueden gestionar las distintas contraseñas de la aplicación, es decir, altas, bajas y modificaciones de las mismas. Inicialmente, se tiene configurada la clave 100 para el superuser. Esta clave se puede modificar una vez iniciado el programa, tal y como se ha comentado anteriormente.

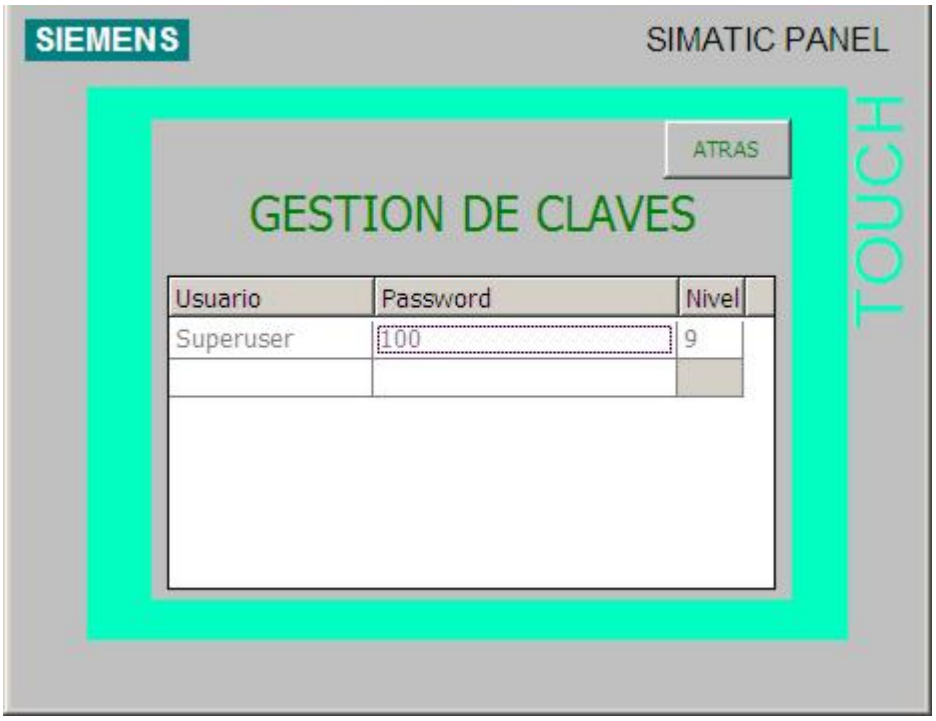


Figura 3.51. Gestión de claves

Ajustar fecha y hora

En esta pantalla podemos modificar la fecha y hora del autómata.

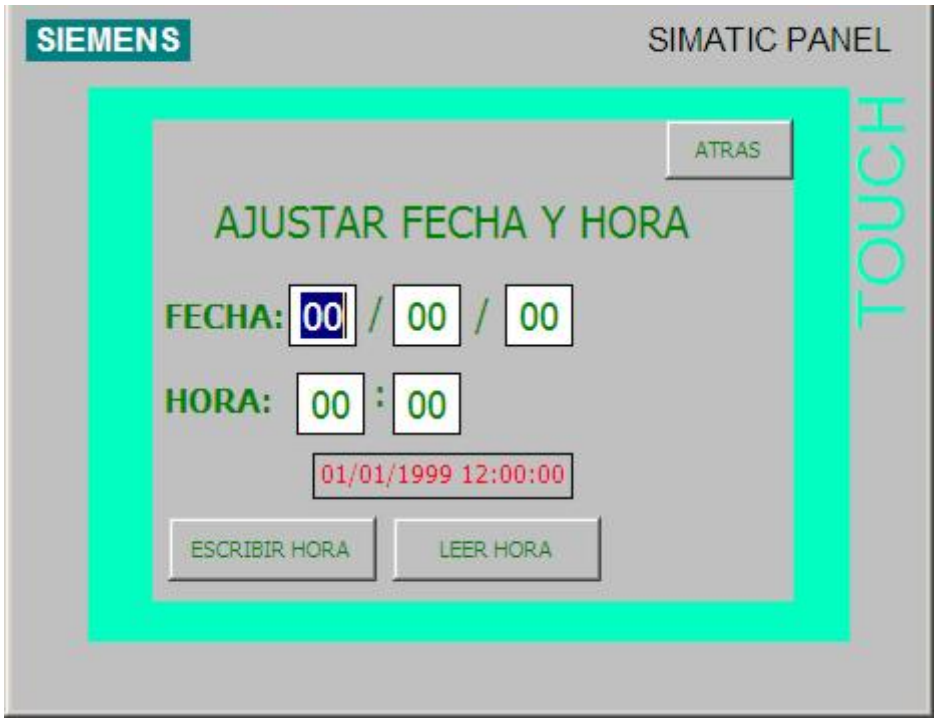


Figura 3.52. Ajuste de Fecha y Hora

El botón ESCRIBIR HORA activa el guión ESCRIBIR_FECHA_HORA. Cuyo código interno es el siguiente:

AÑO_DB=AÑO_TP
MES_DB=MES_TP
DIA_DB=DIA_TP
HORA_DB=HORA_TP
MINUTOS_DB=MINUTOS_TP
SEGUNDOS_DB=SEGUNDOS_TP
CAMBIAR_HORA=1

Donde AÑO_DB,MES_DB, DIA_DB, HORA_DB, MINUTOS_DB, SEGUNDOS_DB y CAMBIAR_HORA son variables de control que están incluidas en el DB 98 "DB_CAMBIO_HORA", que se ha creado para gestionar desde el PLC la hora y la fecha. Mientras que por otra parte, AÑO_TP, MES_TP, DIA_TP, HORA_TP,

MINUTOS_TP y SEGUNDOS_TP son variables sin control por parte del autómata, sino para uso del propio HMI.

Por otra parte, el botón LEER HORA ejecuta el guión LEER_FECHA_HORA. Su código es el siguiente:

```
AÑO_TP=AÑO_DB
MES_TP=MES_DB
DIA_TP=DIA_DB
HORA_TP=HORA_DB
MINUTOS_TP=MINUTOS_DB
SEGUNDOS_TP=SEGUNDOS_DB
```

3.4. Programa S7

En el siguiente capitulo, incluiremos y explicaremos la parte de programación realizada que se ha de incluir en el PLC para el correcto funcionamiento del sistema.

El lenguaje de programación utilizado es SCL, que se ha usado junto con AWL para pequeños módulos de control.

3.4.1. Introducción a SCL

SCL (*Structured Control Language*) es un lenguaje de programación de alto nivel orientado a PASCAL. El lenguaje se basa en una norma para PLC (autómatas programables).

La norma DIN EN-61131-3 (int. IEC 1131-3) normaliza los lenguajes de programación para autómatas programables. El lenguaje de programación SCL cumple el PLCopen Basis Level del lenguaje ST (texto estructurado) definido en esta norma.

Además de elementos de lenguaje de alto nivel, SCL incluye también elementos típicos del PLC, como entradas, salidas, temporizadores, marcas, llamadas a bloques, etc., que son elementos del propio lenguaje. Es decir, SCL completa y amplía el software de programación STEP 7 con sus lenguajes de programación KOP, FUP y AWL.

¿Qué ventajas ofrece SCL?

SCL ofrece todas las ventajas de un lenguaje de programación de alto nivel. Además, SCL ofrece características diseñadas expresamente asistir la programación estructurada, p.ej.:

- SCL asiste el diseño de bloques de STEP 7, por lo que permite, junto a AWL, KOP y FUP programar bloques conforme a la norma.
- No es necesario crear cada una de las funciones requeridas, sino que se puede recurrir a bloques preconfeccionados, tales como las funciones de sistema (FC) o los bloques de función de sistema (SFB) que residen en el sistema operativo de la unidad central de procesamiento.
- Los bloques programados con SCL se pueden alternar con bloques programados en AWL, KOP y FUP. Esto significa que un bloque programado con SCL puede llamar a otro bloque programado en AWL, en KOP o en FUP. Asimismo, los bloques SCL también pueden ser llamados en programas AWL, KOP y FUP. En definitiva, los lenguajes de programación de STEP 7 y SCL (paquete opcional) se complementan de forma óptima.
- Salvo contadas excepciones, los objetos fuente creados con SCL para STEP 5 son compatibles con versiones superiores; es decir, estos programas también se pueden editar, compilar y testear con S7-SCL.
- En aplicaciones concretas, los bloques SCL se pueden descompilar al lenguaje de programación de STEP 7 AWL (lista de instrucciones). No es posible descompilar a SCL.
- Si ya se dispone de una cierta experiencia en los lenguajes de programación de alto nivel, SCL resulta aprender con facilidad.

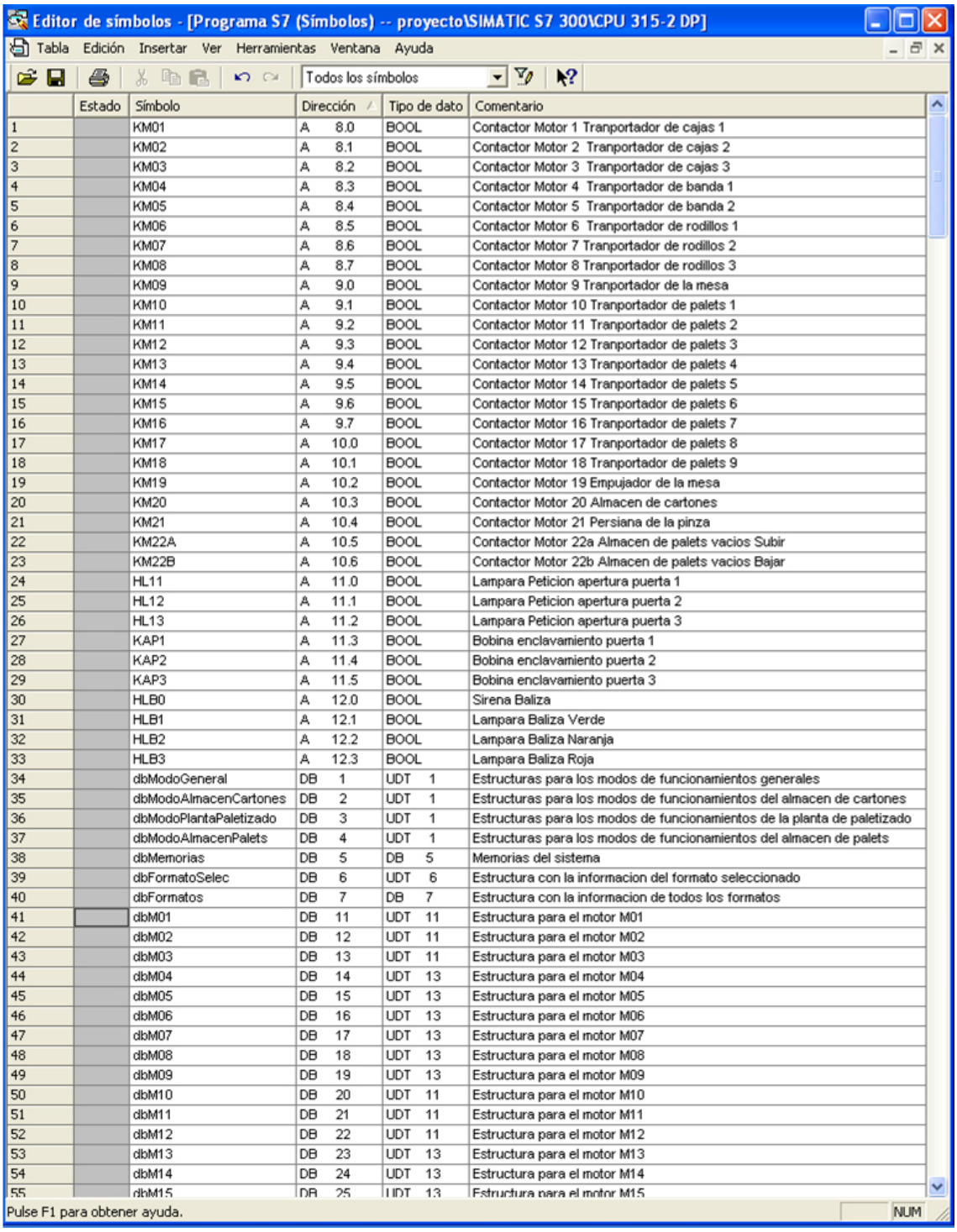
- Al crear sus programas, el programador dispone de cómodas funciones para editar el texto fuente.
- En el proceso de compilación se generan bloques a partir del programa editado que pueden ejecutarse en todas las CPU del sistema de automatización S7-300/400 a partir de la CPU 314.
- Las funciones de test de SCL permiten localizar errores lógicos de programación en los bloques compilados sin errores. La búsqueda de errores se realiza en lenguaje fuente.

3.4.2. Estructura del programa S7

Antes de abordar directamente con el código y los GRAFCET utilizados para su programación, pasamos a explicar brevemente la estructura que se ha utilizado para programar convenientemente nuestro sistema.

Lo primero que hay que detallar es la tabla de símbolos utilizada, donde se asignan a cada número de bloque (UDT, DB, FC, etc.) un nombre simbólico que le hace referencia.

Mostramos la pantalla donde se observan todos los elementos:



	Estado	Símbolo	Dirección	Tipo de dato	Comentario
1		KM01	A 8.0	BOOL	Contactor Motor 1 Tranportador de cajas 1
2		KM02	A 8.1	BOOL	Contactor Motor 2 Tranportador de cajas 2
3		KM03	A 8.2	BOOL	Contactor Motor 3 Tranportador de cajas 3
4		KM04	A 8.3	BOOL	Contactor Motor 4 Tranportador de banda 1
5		KM05	A 8.4	BOOL	Contactor Motor 5 Tranportador de banda 2
6		KM06	A 8.5	BOOL	Contactor Motor 6 Tranportador de rodillos 1
7		KM07	A 8.6	BOOL	Contactor Motor 7 Tranportador de rodillos 2
8		KM08	A 8.7	BOOL	Contactor Motor 8 Tranportador de rodillos 3
9		KM09	A 9.0	BOOL	Contactor Motor 9 Tranportador de la mesa
10		KM10	A 9.1	BOOL	Contactor Motor 10 Tranportador de palets 1
11		KM11	A 9.2	BOOL	Contactor Motor 11 Tranportador de palets 2
12		KM12	A 9.3	BOOL	Contactor Motor 12 Tranportador de palets 3
13		KM13	A 9.4	BOOL	Contactor Motor 13 Tranportador de palets 4
14		KM14	A 9.5	BOOL	Contactor Motor 14 Tranportador de palets 5
15		KM15	A 9.6	BOOL	Contactor Motor 15 Tranportador de palets 6
16		KM16	A 9.7	BOOL	Contactor Motor 16 Tranportador de palets 7
17		KM17	A 10.0	BOOL	Contactor Motor 17 Tranportador de palets 8
18		KM18	A 10.1	BOOL	Contactor Motor 18 Tranportador de palets 9
19		KM19	A 10.2	BOOL	Contactor Motor 19 Empujador de la mesa
20		KM20	A 10.3	BOOL	Contactor Motor 20 Almacen de cartones
21		KM21	A 10.4	BOOL	Contactor Motor 21 Persiana de la pinza
22		KM22A	A 10.5	BOOL	Contactor Motor 22a Almacen de palets vacios Subir
23		KM22B	A 10.6	BOOL	Contactor Motor 22b Almacen de palets vacios Bajar
24		HL11	A 11.0	BOOL	Lampara Peticion apertura puerta 1
25		HL12	A 11.1	BOOL	Lampara Peticion apertura puerta 2
26		HL13	A 11.2	BOOL	Lampara Peticion apertura puerta 3
27		KAP1	A 11.3	BOOL	Bobina enclavamiento puerta 1
28		KAP2	A 11.4	BOOL	Bobina enclavamiento puerta 2
29		KAP3	A 11.5	BOOL	Bobina enclavamiento puerta 3
30		HLB0	A 12.0	BOOL	Sirena Baliza
31		HLB1	A 12.1	BOOL	Lampara Baliza Verde
32		HLB2	A 12.2	BOOL	Lampara Baliza Naranja
33		HLB3	A 12.3	BOOL	Lampara Baliza Roja
34		dbModoGeneral	DB 1	UDT 1	Estructuras para los modos de funcionamientos generales
35		dbModoAlmacenCartones	DB 2	UDT 1	Estructuras para los modos de funcionamientos del almacen de cartones
36		dbModoPlantaPaletizado	DB 3	UDT 1	Estructuras para los modos de funcionamientos de la planta de paletizado
37		dbModoAlmacenPalets	DB 4	UDT 1	Estructuras para los modos de funcionamientos del almacen de palets
38		dbMemorias	DB 5	DB 5	Memorias del sistema
39		dbFormatoSelec	DB 6	UDT 6	Estructura con la informacion del formato seleccionado
40		dbFormatos	DB 7	DB 7	Estructura con la informacion de todos los formatos
41		dbM01	DB 11	UDT 11	Estructura para el motor M01
42		dbM02	DB 12	UDT 11	Estructura para el motor M02
43		dbM03	DB 13	UDT 11	Estructura para el motor M03
44		dbM04	DB 14	UDT 13	Estructura para el motor M04
45		dbM05	DB 15	UDT 13	Estructura para el motor M05
46		dbM06	DB 16	UDT 13	Estructura para el motor M06
47		dbM07	DB 17	UDT 13	Estructura para el motor M07
48		dbM08	DB 18	UDT 13	Estructura para el motor M08
49		dbM09	DB 19	UDT 13	Estructura para el motor M09
50		dbM10	DB 20	UDT 11	Estructura para el motor M10
51		dbM11	DB 21	UDT 11	Estructura para el motor M11
52		dbM12	DB 22	UDT 11	Estructura para el motor M12
53		dbM13	DB 23	UDT 13	Estructura para el motor M13
54		dbM14	DB 24	UDT 13	Estructura para el motor M14
55		dbM15	DB 25	UDT 13	Estructura para el motor M15

Figura 3.53. Lista de símbolos

Como la lista es enorme, y consideramos innecesario el mostrarla al completo (tan sólo son asignaciones de bloques a nombres simbólicos) pasamos a exponer un breve resumen de la numeración que se ha realizado para las UDTs, DBs y FCs:

Tipos de datos:

tdXXXX	[11..x]

Ejemplo de algunos UDTs

tdModoFuncionamiento	1 //UDT especial. Tipo de dato modos de funcionamiento.
tdFormato	6 //UDT Especial. Tipo de dato para un formato de palet.
tdMotor	11
tdMotorInversor	12
tdMotorVariador	13
tdCilindro	15

Bloques de datos:

dbXXXX	[1..10]	Generales
dbXXXX	[11..50]	Sistemas basicos(cilindros, motores, …)
dbXXXX	[51..90]	Sistemas complejos
dbXXXX	[91..100]	Comunicación HMI
dbXXXX	[101..x]	Secuencias automaticas

Ejemplo de algunos DBs utilizados

dbModoGeneral	1
dbModoAlmacenCartones	2
dbModoPlantaPaletizado	3
dbAlarmas	91
dbVisualizacion	92
dbManuales	93
dbParametros	94

Funciones:

fcXXXX	[1..10]	Generales

fcXXXX	[11..50]	Sistemas basicos
fcXXXX	[51..100]	Llamadas a sistemas basicos
fcXXXX	[101..x]	Secuencias automaticas

Ejemplo de algunas FCs utilizadas:

fcGeneral	1
fcCtrlMotor	11
fcCtrlMotorInversor	12
fcCtrlMotorVariador	13
fcCtrlCilindro	15
fcMotores	51
fcCilindros	52

Las entradas y salidas no se van a nombrar, pues son las mismas que ya se nombraron en el capítulo 2. Los espacios de memoria (Mx xx.x) se verán según se vaya exponiendo el código el símbolo elegido.

Para resumir, mostramos a continuación el número total de bloques que se han usado para este proyecto:

Tipo de bloque	Número
DB	97
OB	1
FC	34
UDT	19
TOTAL	151

Tabla 3.5. Bloques S7

Dada la magnitud del proyecto y del código escrito, no presentaremos la totalidad del código utilizado en este capítulo, sino que explicaremos la forma que se ha seguido para realizar toda la programación, y en el anexo adjunto mostraremos aquellos

códigos relevantes necesarios para comprender adecuadamente cómo se ha programado nuestro sistema.

3.4.3. Programación del OB 1

En este punto, mostraremos la programación realizada del bloque OB 1 y el FC al cual llama éste para controlar el sistema:

OB 1

Es el bloque que se ejecuta en cada ciclo de procesamiento del autómata. Este bloque se llegó a comprobar que no admitía SCL, por lo que se creó de manera que su única tarea es llamar a la función fcGeneral, que hará de "main" del programa.

fcGeneral (FC 1)

Es la función que se encargará de llamar a todas las funciones que se usan para el funcionamiento del programa software. Mostramos en primer momento la estructura de las llamadas que se ha usado:

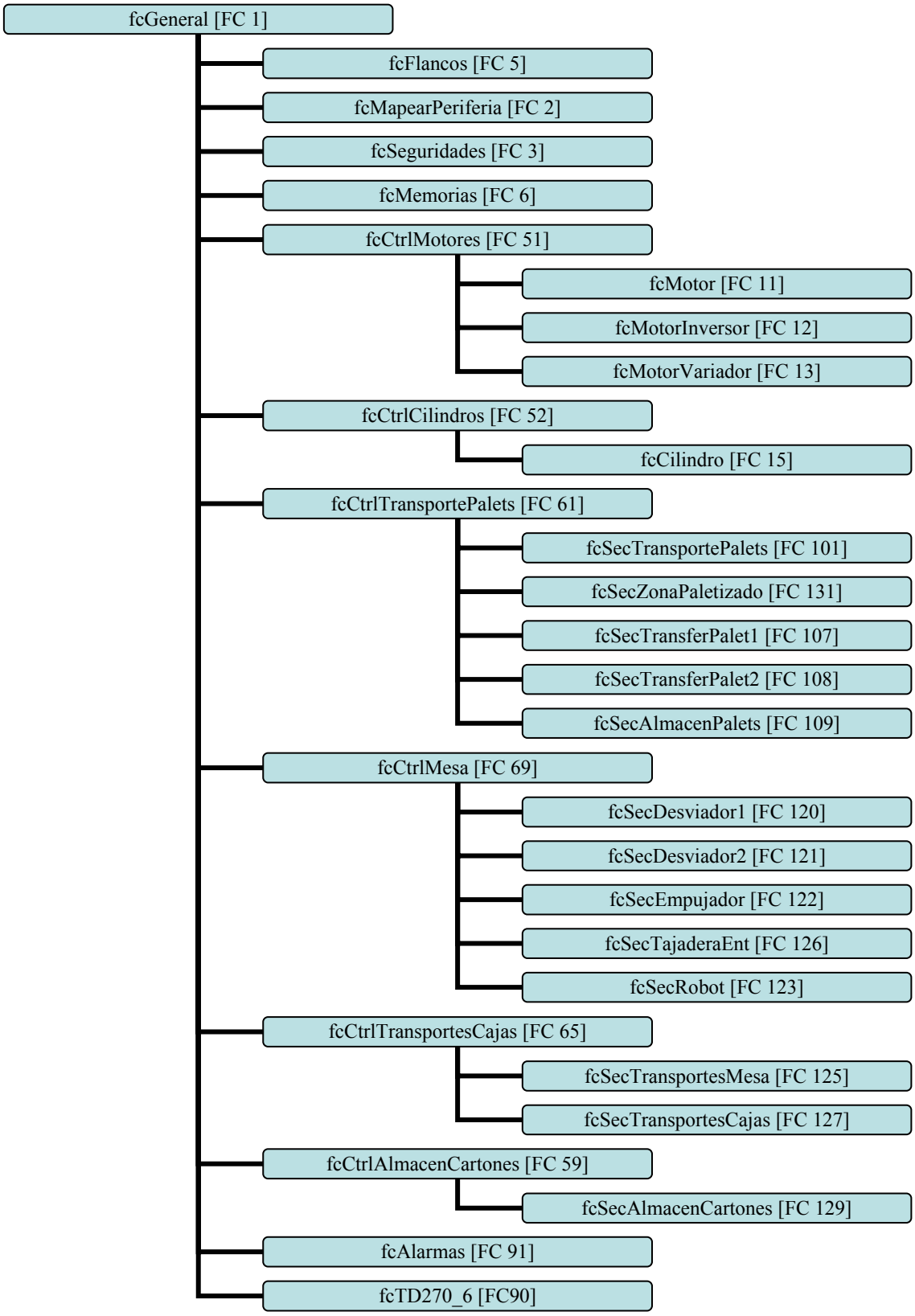


Figura 3.54. Llamadas del fcGeneral

Mostramos a continuación el código SCL usado:

```
FUNCTION fcGeneral : VOID
BEGIN
  Ba0 := false;
  Ba1 := true;
  Inter := M100.3;

  //Cargar formato
  IF dbMemorias.bPeticionCambioFormato THEN
    dbFormatos.nfs := dbMemorias.iFormatoSeleccionado;
    dbFormatoSelec := dbFormatos.tf[dbFormatos.nfs];

    dbMemorias.bPeticionCambioFormato := false;
  END_IF;

  //-----------------------------------------------------------------------------------------
  //Control de flancos de señales
  fcFlancos();

  //Mapeo de la periferia
  fcMapearPeriferia();

  //Control de seguridades
  fcSeguridades();

  //Construccion de memorias
  fcMemorias();

  //-----------------------------------------------------------------------------------------
  //Modos de funcionamiento
  dbModoGeneral.SL_AUT_MAN := dbParametros.AUT_MAN;
```

```
 dbModoGeneral.Manual := (NOT dbModoGeneral.SL_AUT_MAN) AND
dbModoGeneral.SegMando;
 dbModoGeneral.Automatico := dbModoGeneral.SL_AUT_MAN AND
dbModoGeneral.SegAuto;

 dbModoPlantaPaletizado.Manual := (NOT dbModoPlantaPaletizado.SL_AUT_MAN)
AND
   dbModoPlantaPaletizado.SegMando;
 dbModoPlantaPaletizado.Automatico := dbModoPlantaPaletizado.SL_AUT_MAN
AND
   dbModoPlantaPaletizado.SegAuto;

 dbModoAlmacenCartones.Manual := (NOT
dbModoAlmacenCartones.SL_AUT_MAN) AND
   dbModoAlmacenCartones.SegMando;
 dbModoAlmacenCartones.Automatico := dbModoAlmacenCartones.SL_AUT_MAN
AND
   dbModoAlmacenCartones.SegAuto;

 dbModoAlmacenPalets.Manual := (NOT dbModoAlmacenPalets.SL_AUT_MAN)
AND
   dbModoAlmacenPalets.SegMando;
 dbModoAlmacenPalets.Automatico := dbModoAlmacenPalets.SL_AUT_MAN AND
   dbModoAlmacenPalets.SegAuto;

 //Activación de contactores de servicios para variadores
  km04 := QF07;
  km05 := QF07;
  km06 := QF07;
  km07 := QF07;
  km08 := QF07;
  km09 := QF07;
  km13 := QF07;
  km14 := QF07;
```

```
  km15 := QF07;
  km16 := QF07;
  km17 := QF07;
  km18 := QF07;
  km19 := QF07;
  km20 := QF07;
  km21 := QF07;

 //Activacion de las valvulas de corte neumaticas
 EV1550 := QF07;
 EV3532 := QF07;
 EV4520 := QF07 AND RS03;

 //Inicio de ciclo ModoGeneral
 dbModoGeneral.CI := dbModoPlantaPaletizado.CI AND dbModoAlmacenCartones.CI
AND dbModoAlmacenPalets.CI;
 dbModoGeneral.EnCiclo :=
   ((dbModoGeneral.IniCiclo AND dbModoGeneral.CI) OR dbModoGeneral.EnCiclo)
   AND (NOT dbModoGeneral.FinCiclo) AND dbModoGeneral.Automatico;
 dbModoGeneral.FinCiclo := dbModoGeneral.EnCiclo AND dbModoGeneral.CI
   AND dbModoGeneral.MemoriaFinCiclo;
 IF NOT dbModoGeneral.EnCiclo THEN
   dbModoGeneral.MemoriaFinCiclo := false;
 END_IF;

 //Inicio de ciclo ModoPlantaPaletizado
 dbModoPlantaPaletizado.CI := true;
 dbModoPlantaPaletizado.EnCiclo :=
   ((dbModoPlantaPaletizado.IniCiclo AND dbModoPlantaPaletizado.CI) OR
dbModoPlantaPaletizado.EnCiclo)
   AND (NOT dbModoPlantaPaletizado.FinCiclo) AND
dbModoPlantaPaletizado.Automatico;
 dbModoPlantaPaletizado.FinCiclo := dbModoPlantaPaletizado.EnCiclo AND
dbModoPlantaPaletizado.CI
```

```
    AND dbModoPlantaPaletizado.MemoriaFinCiclo;
  IF NOT dbModoPlantaPaletizado.EnCiclo THEN
    dbModoPlantaPaletizado.MemoriaFinCiclo := false;
  END_IF;

  //Inicio de ciclo ModoAlmacenCartones
  dbModoAlmacenCartones.CI := true;
  dbModoAlmacenCartones.EnCiclo :=
    (((dbModoAlmacenCartones.IniCiclo OR RS03)
    AND dbModoAlmacenCartones.CI) OR dbModoAlmacenCartones.EnCiclo)
    AND (NOT dbModoAlmacenCartones.FinCiclo) AND
dbModoAlmacenCartones.Automatico;
  dbModoAlmacenCartones.FinCiclo := dbModoAlmacenCartones.EnCiclo AND
dbModoAlmacenCartones.CI
    AND dbModoAlmacenCartones.MemoriaFinCiclo;
  IF NOT dbModoAlmacenCartones.EnCiclo THEN
    dbModoAlmacenCartones.MemoriaFinCiclo := false;
  END_IF;

  //Inicio de ciclo ModoAlmacenPalets
  dbModoAlmacenPalets.CI := dbSecTranspPalet1.CondicionesIniciales AND
(dbSecAlmacenPalets.CondicionesIniciales) AND
  dbSecTransferPalet1.CondicionesIniciales AND
dbSecZonaPaletizado.CondicionesIniciales AND
  dbSecTranspPalet4.CondicionesIniciales AND
dbSecTranspPalet5.CondicionesIniciales AND
dbSecTransferPalet2.CondicionesIniciales;
  dbModoAlmacenPalets.EnCiclo :=
    ((dbModoAlmacenPalets.IniCiclo AND dbModoAlmacenPalets.CI) OR
dbModoAlmacenPalets.EnCiclo)
    AND (NOT dbModoAlmacenPalets.FinCiclo) AND
dbModoAlmacenPalets.Automatico;
  dbModoAlmacenPalets.FinCiclo := dbModoAlmacenPalets.EnCiclo AND
dbModoAlmacenPalets.CI
```

```
  AND dbModoAlmacenPalets.MemoriaFinCiclo;
IF NOT dbModoAlmacenPalets.EnCiclo THEN
  dbModoAlmacenPalets.MemoriaFinCiclo := false;
END_IF;

//-----------------------------------------------------------------------------------------
//Llamada a los motores
fcCtrlMotores();

//Llamada a los cilindros
fcCtrlCilindros();

//-----------------------------------------------------------------------------------------
//Llamada a la secuencia de los transportes de cajas
fcCtrlTransportesCajas();

//Llamada a las secuencias de los desviadores, tajadera, empujador y robot
fcCtrlMesa();

//Llamada a las secuencias de transportes de palets
fcCtrlTransportesPalets();

//Llamada a la secuencia de los transportes de la mesa
fcCtrlAlmacenCartones();

//-----------------------------------------------------------------------------------------
//Alarmas
fcAlarmas();


//Llamada a visualizaciones
IF (Marca05Hz AND NOT bFlancoLlamadasP) THEN
  fcTD270_6();
  bFlancoLlamadasP := TRUE;
```

```
 ELSIF NOT Marca05Hz THEN
   bFlancoLlamadasP := FALSE;
 END_IF;

END_FUNCTION;
```

Explicamos a continuación el código SCL:

En primer lugar se activan los bits Ba0 (M0.0) y Ba1 (M0.1), bits auxiliares que usaremos para mantenerlos siempre a 0 y a 1 respectivamente. Nos servirán para saber cuando se produce el primer ciclo de programa o un reinicio, ya que ambos inicialmente valen 0. También actualizamos el bit Inter (M0.2) según el valor del bit M100.3, que si recordamos, tal y como dijimos con anterioridad, es el bit que se usará para conseguir el efecto de parpadeo de luces intermitentes.

En segundo lugar pasamos a cargar el formato seleccionado (2x2, 2x3 o 2x4) siempre y cuando el bit "bPeticionCambioFormato" que se encuentra en el "dbMemorias" (DB 5) esté a TRUE. Este bit, como se comprobó anteriormente, es modificado en las pantallas de la TP270 (ver Figura 3.50). Para fijar el formato seleccionado, se actualiza el bit "nfs" que se encuentra en el dbFormatos (DB 7) con el valor del bit "iFormatoSeleccionado", cuyo valor se obtiene igualmente desde las pantallas del TP270. También se actualiza la estructura con la información asociada al formato seleccionado, asignando el "dbFormatoSelec" (DB 6) a la tabla de formatos preconfigurada que contiene los datos necesarios para todos los formatos, el "tf" que se encuentra dentro del DB 7 "dbFormatos". La estructura de estos DB's se muestran en el ANEXO del proyecto.

Tras realizar todo esto, se realizan una serie de llamadas a diversas funciones, cuyo código en SCL mostramos en el ANEXO, por no tener mucho más que comentar.

Tras dichas llamadas, se configura el modo de funcionamiento del sistema en general y de las distintas zonas en las que se ha dividido el mismo: en automático o en manual. Para ello, el estado se almacena en diversas DB's creadas para tal efecto:

- *dbModoGeneral* (DB 1): Estructuras para los modos de funcionamientos generales
- *dbModoAlmacenCartones* (DB 2): Estructuras para los modos de funcionamientos del almacen de cartones.
- *dbModoPlantaPaletizado* (DB 3): Estructuras para los modos de funcionamientos de la planta de paletizado.
- *dbModoAlmacenPalets* (DB 4): Estructuras para los modos de funcionamientos del almacen de palets

Cada DB contiene el mismo UDT, el "tdModoFuncionamiento" (UDT 1), cuya estructura es la siguiente (código SCL):

```
TYPE tdModoFuncionamiento
STRUCT
   Automatico: BOOL;    //Entrada/Salida. Sistema en Automático
   Manual: BOOL;        //Entrada/Salida. Sistema en Manual
   SegMando: BOOL;
   SegAuto: BOOL;

   SL_AUT_MAN: BOOL;
   EnCiclo: BOOL;
   IniCiclo: BOOL;
   FinCiclo: BOOL;
   MemoriaFinCiclo: BOOL;
   CI: BOOL;
END_STRUCT
END_TYPE
```

Por tanto, la definición de cada DB es la siguiente:

```
DATA_BLOCK dbModoGeneral tdModoFuncionamiento
BEGIN
END_DATA_BLOCK

DATA_BLOCK dbModoAlmacenCartones tdModoFuncionamiento
BEGIN
```

```
END_DATA_BLOCK

DATA_BLOCK dbModoAlmacenPalets tdModoFuncionamiento
BEGIN
END_DATA_BLOCK

DATA_BLOCK dbModoPlantaPaletizado tdModoFuncionamiento
BEGIN
END_DATA_BLOCK
```

Tras realizar todo esto, se activan los contactores de los motores y las válvulas de corte neumáticas. Y a continuación se configura las condiciones iniciales para el funcionamiento en modo automático de cada secuencia que controlará cada una de las partes en las que se compone el proyecto.

Seguidamente, se llama a las funciones que controlan el funcionamiento de los motores y cilindros. En el ANEXO mostraremos el código SCL de cada una de estas 2 funciones. A modo de ejemplo, mostramos a continuación un extracto de los códigos de las funciones “fcCtrlMotores” y “fcCtrlCilindros”:

```
FUNCTION fcCtrlMotores : VOID
BEGIN
  //Motor M01 ----------------------------------------------------------(Marcha/Paro)
  dbM01.Termico := QKM01;
  dbM01.SeguridadMecanica := true;
  dbM01.OrdenManual := dbManuales.bmA8_00;

  dbM01.Marcha := dbTC1.Marcha;
  dbM01.Paro   := dbTC1.Paro;

  fcMotor(MF:=dbModoPlantaPaletizado, M:=dbM01);

  KM01 := dbM01.Contactor;
  dbAlarmas.alm0084 := dbM01.Alarma;
```

```
//Motor M05 ----------------------------------------------------------(Variador)
  dbM05.Termico := QKM05;
  dbM05.SeguridadMecanica := true;
  dbM05.SeguridadMecanicaInversa := true;
  dbM05.OrdenManual := dbManuales.bmVM05D;
  dbM05.OrdenManualInversa := dbManuales.bmVM05I;

  dbM05.Marcha := dbTransportesMesa.M05_Marcha;
  dbM05.Paro   := dbTransportesMesa.M05_Paro;

  fcMotorVariador(MF:=dbModoPlantaPaletizado, M:=dbM05);

  dbAlarmas.alm0090 := dbM05.Alarma;
//Motor M22 ----------------------------------------------------------(Inversor)
  dbM22.Termico := QKM22;
  dbM22.SeguridadMecanica := NOT SQ3061;
  dbM22.SeguridadMecanicaInversa := NOT SQ3063;
  dbM22.OrdenManual := dbManuales.bmIM22D;
  dbM22.OrdenManualInversa := dbManuales.bmIM22I;

  dbM22.Marcha := dbAlmacenPalets.Marcha;
  dbM22.MarchaInversa := dbAlmacenPalets.MarchaInv;
  dbM22.Paro := dbAlmacenPalets.Paro;

  fcMotorInversor(MF:=dbModoAlmacenPalets, M:=dbM22);

  KM22B := dbM22.Contactor;
  KM22A := dbM22.ContactorInverso;
  dbAlarmas.alm0111 := dbM22.Alarma;

END_FUNCTION
```

En este caso, hemos mostrado ejemplo para los 3 tipos de motores que hay: marcha/paro, con variador o inversor. El código seguiría la misma estructura para el

resto de motores. Se observa que para cada motor, se ha creado previamente un DB para cada uno de ellos con los datos necesarios. Para ello también se creó un UDT que definía dichos datos. Así pues, mostramos cómo serían éstos DB's:

```
DATA_BLOCK dbM01 tdMotor
BEGIN
END_DATA_BLOCK

DATA_BLOCK dbM05 tdMotorVariador
BEGIN
   NumVar:= 5;
END_DATA_BLOCK

DATA_BLOCK dbM22 tdMotorInversor
BEGIN
END_DATA_BLOCK
```

Y los UDT's:

```
TYPE tdMotor
STRUCT
   Marcha:BOOL:=FALSE;     //Entrada. 'Pulsador' de Marcha
   Paro:BOOL:=TRUE;   //Entrada. 'Pulsador' de Parada. Inicialmente, motores parados
   Termico:BOOL;          //Entrada. Disparo Magnetotérmico
   SeguridadMecanica: BOOL;  //Entrada. Condición para la activación del motor
   OrdenManual : BOOL;     //Entrada. Orden de activación manual.
   Contactor:BOOL;         //Salida. Contactor de activación del motor.
   Alarma:BOOL;           //Salida. Información de disparo para HMI.
   Error:BOOL;           //Salida. Error de programacion
END_STRUCT
END_TYPE

TYPE tdMotorVariador
STRUCT
   NumVar:INT:=0;         //Numero del variador
```

```
    Marcha:BOOL:=FALSE;     //Entrada. 'Pulsador' de Marcha en Sentido Directo
    MarchaInversa:BOOL:=FALSE;  //Entrada. 'Pulsador' de Marcha en Sentido Inverso
    Paro:BOOL:=TRUE;   //Entrada. 'Pulsador' de Parada. Inicialmente, motores parados
    Termico:BOOL;          //Entrada. Disparo Magnetotérmico
    SeguridadMecanica: BOOL;//Entrada. Condición para la activación del motor en
sentido directo
    SeguridadMecanicaInversa: BOOL;//Entrada. Condición para la activación del motor
en sentido inverso
    OrdenManual : BOOL;     //Entrada. Orden de activación manual en sentido directo
    OrdenManualInversa : BOOL;      //Entrada. Orden de activación manual en sentido
inverso
    ConsignaVelocidad:INT;      //Entrada. Valor de la consigna de velociadad
    RampaAceleracion:DINT; //Entrada. Tiempo de rampa de aceleración (SEGUNDOS)
    RampaDesaceleracion:DINT;       //Entrada. Tiempo de rampa de desaceleración
(SEGUNDOS)
    Contactor:BOOL;         //Salida. Contactor de activación del motor en sentido directo
    ContactorInverso:BOOL; //Salida. Contactor de activación del motor en sentido
inverso
    Alarma:BOOL;          //Salida. Información de disparo para HMI
    Error:BOOL;           //Salida. Error de programacion
END_STRUCT
END_TYPE

TYPE tdMotorInversor
STRUCT
    Marcha:BOOL:=FALSE;     //Entrada. 'Pulsador' de Marcha en Sentido Directo
    MarchaInversa:BOOL:=FALSE; //Entrada. 'Pulsador' de Marcha en Sentido Inverso
    Paro:BOOL:=TRUE;  //Entrada. 'Pulsador' de Parada. Inicialmente, motores parados
    Termico:BOOL;          //Entrada. Disparo Magnetotérmico
    SeguridadMecanica: BOOL;//Entrada. Condición para la activación del motor en
sentido directo
    SeguridadMecanicaInversa: BOOL;//Entrada. Condición para la activación del motor
en sentido inverso
    OrdenManual : BOOL;     //Entrada. Orden de activación manual en sentido directo
```

```
   OrdenManualInversa : BOOL;     //Entrada. Orden de activación manual en sentido inverso
   Contactor: BOOL;        //Salida. Contactor de activación del motor en sentido directo
   ContactorInverso: BOOL; //Salida. Contactor de activación del motor en sentido inverso
   Alarma:BOOL;          //Salida. Información de disparo para HMI
   Error:BOOL;           //Salida. Error de programacion
END_STRUCT
END_TYPE
```

Las FC's a las que llaman la función "fcCtrlMotores" las mostramos en el ANEXO.

Y respecto a los cilindros:

```
FUNCTION fcCtrlCilindros : VOID
BEGIN
//Cilindro C02 ------------------------------------------------------------
  dbC02.sr := true;
  dbC02.sa := true;
  dbC02.dr := SQ1012;
  dbC02.da := SQ1013;

  dbC02.omr := dbManuales.bmA151_02;
  dbC02.oma := dbManuales.bmA151_03;

  dbC02.aa := dbDesviador1.Desviador_aa;
  dbC02.ar := dbDesviador1.Desviador_ar;

  dbC02.ta := 30;
  dbC02.tr := 30;
  dbC02.td := 80;

  fcCilindro(MF:=dbModoPlantaPaletizado, C:=dbC02);

  EV1512 := dbC02.ore;
```

```
  EV1513 := dbC02.oav;

  dbAlarmas.alm1512 := dbC02.amr;
  dbAlarmas.alm1513 := dbC02.ama;
  dbAlarmas.alm1012 := dbC02.afd;
  dbAlarmas.alm1013 := dbC02.afd;
END_FUNCTION
```

Donde igualmente se ha creado un DB para cada cilindro, con su correspondiente UDT:

```
DATA_BLOCK dbC02 tdCilindro
BEGIN
END_DATA_BLOCK

TYPE tdCilindro
STRUCT
   sa : BOOL; //Entrada. Seguridad Avance
   sr : BOOL; //Entrada. Seguridad Retroceso

   ia : BOOL; //Salida. Informacion Avance
   ir : BOOL; //Salida. Informacion Retroceso
   da : BOOL; //Entrada. Detector avance
   dr : BOOL; //Entrada. Detector retroceso

   aa  : BOOL; //Entrada. Autorizacion Avance
   ar  : BOOL; //Entrada. Autorizacion Retroceso
   oma : BOOL; //Entrada. Orden Manual de Avance
   omr : BOOL; //Entrada. Orden Manual de Retroceso
   oav : BOOL; //Salida. Orden de Avance
   ore : BOOL; //Salida. Orden de Retroceso

   ta    : INT;  //Temporizador de avance. Hay que inicializarlo
   tr    : INT;  //Temporizador de retroceso Hay que inicializarlo
   td    : INT;  //Temporizador de detectores. Hay que inicializarlo
```

```
   cta   : INT;  //Contador Temporizacion Avance
   ctr   : INT;  //Contador Temporizacion Retroceso
   ctd   : INT;  //Contador Temporizacion Detectores
   ama   : BOOL; //Alarma Movimiento Avance
   amr   : BOOL; //Alarma Movimiento Retroceso
   afd   : BOOL; //Alarma Fallo Detectores
   error : BOOL; //Error de programacion
END_STRUCT
END_TYPE
```

Y al igual que antes, la FC "fcCilindro" la mostraremos en el ANEXO.

Tras poner en marcha motores y cilindros, se realizan por fin las llamadas a las secuencias que controlan propiamente las partes que componen la instalación en modo automático. En el siguiente capítulo mostraremos ejemplo de cómo se ha programado una de éstas secuencias, y en el ANEXO mostraremos el resto.

Tras esto, se llama a la función que hace control de las Alarmas que se producen en el sistema.

Y por último, se hace llamada a la función que gobierna la comunicación con el TP270 de 6". Esta función tiene la peculiaridad que no se puede llamar en todos los ciclos de funcionamiento, ya que el proceso de comunicación requiere cierto retardo. Es por ello que se realiza la llamada cada 4 segundos (según la marca de 0.5 Hz, que permanece 2 segundos activa y 2 segundos desactivada) y durante 1 único ciclo. Para ello se hace uso del bit bFlancoLlamadasP (M0.3), bit para flanco de llamadas de bloques no fundamentales, dejándolo 1 ciclo activo.

3.4.4. Programación de secuencias

En este capítulo explicaremos con detalle la forma que se ha seguido para programar las secuencias propias que controlan las diversas partes de la instalación en el modo automático.

En primer lugar, tal y como se ha visto en el código de la FC "fcGeneral", se ha estructurado en cuatro partes diferenciadas físicamente:

- Control del transportador de cajas: mediante la FC "fcCtrlTransportesCajas"
- Control de la mesa de formación y el robot: "fcCtrlMesa". Donde se controla el funcionamiento de los desviadores, tajadera, empujador y el robot.
- Control de los transportadores de rodillos, por donde circulan los palets: "fcCtrlTransportesPalets".
- Control del almacén de cartones: "fcCtrlAlmacenCartones"

Explicaremos a continuación en detalle el funcionamiento de una de las secuencias. En concreto nos centraremos por ejemplo en la secuencia que controla los movimientos del robot. Para ello, lo primero que hacemos es mostrar extracto del código SCL de la función "fcCtrlMesa" (FC 69), donde se llama a ésta función:

```
FUNCTION fcCtrlMesa : VOID

[…]

//************************************************\\
//* Robot                                        *\\
//************************************************\\
  dbRobot.AbbInPcp := RobotInPcp;
  dbRobot.AbbInFcp := RobotInFcp;
  dbRobot.AbbInPcc := RobotInPcc;
  dbRobot.AbbInFcc := RobotInFcc;
  dbRobot.AbbInPdc := RobotInPdc;
```

```
dbRobot.AbbInFdc := RobotInFdc;
dbRobot.AbbInPdp := RobotInPdp;
dbRobot.AbbInFdp := RobotInFdp;
dbRobot.AbbInEspPcp := RobotInEspPcp;
dbRobot.AbbInEspPcc := RobotInEspPcc;
dbRobot.AbbInEspPdp := RobotInEspPdp;

dbRobot.CamadaPinza := SQ2040;
dbRobot.Vacuostato1 := SQ2022;
dbRobot.Vacuostato2 := SQ2023;
dbRobot.DpZp := dbZonaPaletizado.Dpp;
dbRobot.PersianaArriba := SQ2030 OR SQ2031;
dbRobot.PersianaAbajo := SQ2032 OR SQ2033;

dbRobot.Cuchilla_ia := dbC10.ia;
dbRobot.Cuchilla_ir := dbC10.ir;
dbRobot.Centrador_ia := dbC11.ia;
dbRobot.Centrador_ir := dbC11.ir;
dbRobot.PosVentosas_ia := dbC12.ia;
dbRobot.PosVentosas_ir := dbC12.ir;

dbSecRobot.CondicionesIniciales := (RobotInHome OR RobotInEspPcp OR
  RobotInPcp OR RobotInFcp) AND (dbMemorias.CamadaEnPinza OR
  dbMemorias.NoCamadaEnPinza);

dbSecRobot.EnCiclo := dbModoPlantaPaletizado.EnCiclo;
dbSecRobot.Automatico := dbModoPlantaPaletizado.Automatico;

RobotOutMon := dbModoPlantaPaletizado.Automatico;

fcSecRobot(S:=dbSecRobot, D:=dbRobot);  //Llamada a la secuencia del Robot

RobotOutPcp := dbRobot.AbbOutPcp;
RobotOutPc := dbRobot.AbbOutPc;
```

```
  RobotOutPcc := dbRobot.AbbOutPcc;
  RobotOutCc := dbRobot.AbbOutCc;
  RobotOutPdp := dbRobot.AbbOutPdp;
  RobotOutPd := dbRobot.AbbOutPd;

  RobotOutBit0 := dbRobot.bit0;
  RobotOutBit1 := dbRobot.bit1;
  RobotOutBit2 := dbRobot.bit2;
  RobotOutPonerCarton := dbRobot.concarton;
  EV2522 := dbRobot.Vacio1;
  EV2523 := dbRobot.Vacio2;

  RobotOutSegPer := dbRobot.SegPersiana;
END_FUNCTION
```

Nuevamente, se han hecho 2 DB's distintos para regular el funcionamiento:

- dbRobot (DB 63). Aquí se almacenan los datos del robot.
- dbSecRobot (DB123). Para almacenar los datos de la secuencia del robot.

Donde:

```
DATA_BLOCK dbRobot  tdRobot
BEGIN
END_DATA_BLOCK
```

Y el UDT (123) correspondiente:

```
TYPE tdRobot
STRUCT
   AbbInPcp : BOOL;        //Entrada. Robot en Posición Carga Preparación
   AbbInFcp : BOOL;        //Entrada. Fin de operación Carga Preparación
   AbbInPcc : BOOL;        //Entrada. Robot en Posición Carga Cartón
   AbbInFcc : BOOL;        //Entrada. Fin operación Carga Cartón
   AbbInPdc : BOOL;        //Entrada. Robot en Posición Descarga Cartón
   AbbInFdc : BOOL;        //Entrada. Fin operación Descarga Cartón
```

```
AbbInPdp : BOOL;        //Entrada. Robot en Posición Descarga Preparación
AbbInFdp : BOOL;        //Entrada. Fin de operación Descarga Preparación

AbbInEspPcp : BOOL; //Entrada. Robot Esperando Permiso Coger Preparación.
AbbInEspPcc : BOOL; //Entrada. Robot Esperando Permiso Coger Cartón
AbbInEspPdp : BOOL; //Entrada. Robot Esperando Permiso Dejar Preparación

AbbOutPcp : BOOL;       //Salida. Permiso Coger Preparación
AbbOutPc : BOOL;        //Salida. Preparación Cargada
AbbOutPcc : BOOL;       //Salida. Permiso Coger Cartón
AbbOutCc : BOOL;        //Salida. Cartón Cargado
AbbOutPdp : BOOL;       //Salida. Permiso Dejar Preparación
AbbOutPd : BOOL;        //Salida. Preparación Dejada
AbbOutPdc : BOOL;       //Salida. Permiso descargar cartón
AbbOutCd : BOOL;        //Salida. Cartón descargado
bit0 : BOOL;            //Salida. Altura a dejar Bit 0
bit1 : BOOL;            //Salida. Altura a dejar Bit 1
bit2 : BOOL;            //Salida. Altura a dejar Bit 2
concarton : BOOL;       //Salida. Indica al robot que lleva cartón.
cn : INT:=0;            //Salida. Contador de Niveles
Vacio1:BOOL;            //Salida. Válvula de vacío 1
Vacio2:BOOL;            //Salida. Válvula de vacío 2
CamadaPinza : BOOL;     //Entrada. Camada colocada correctamente en pinza.
Vacuostato1 : BOOL;     //Entrada. Vacuostato de la Ventosa 1
Vacuostato2 : BOOL;     //Entrada. Vacuostato de la Ventosa 2
DpZp : BOOL;            //Entrada. Detector presencia de Palet en Zona Paletizado
PersianaArriba:BOOL;    //Entrada. Persiana arriba
PersianaAbajo:BOOL;     //Entrada. Persiana abajo

Cuchilla_ia : BOOL;     //Entrada. Cuchilla lateral pinza robot información avance
Cuchilla_ir : BOOL;     //Entrada. Cuchilla lateral pinza robot información
retroceso
Cuchilla_aa : BOOL;     //Salida. Cuchilla lateral pinza robot autorización avance
Cuchilla_ar : BOOL;     //Salida. Cuchilla lateral pinza robot autorización retroceso
```

```
   Centrador_ia : BOOL;      //Entrada. Centrador lateral pinza robot información
avance
   Centrador_ir : BOOL;      //Entrada. Centrador lateral pinza robot información
retroceso
   Centrador_aa : BOOL;      //Salida. Centrador lateral pinza robot autorización avance
   Centrador_ar : BOOL;      //Salida. Centrador lateral pinza robot autorización
retroceso

   PosVentosas_ia : BOOL;   //Entrada. Portaventosas lateral pinza robot info. avance
   PosVentosas_ir : BOOL;   //Entrada. Portaventosas lateral pinza robot info. retroceso
   PosVentosas_aa : BOOL;  //Salida. Portaven. lateral pinza robot autorización avance
   PosVentosas_ar : BOOL;   //Salida. Portaven. lateral pinza robot autorización
retroceso

   SegPersiana : BOOL;       //Salida. Persiana bien posicionada

   OAvaPer: BOOL;            //Salida. Orden avance Persiana
   ORetPer: BOOL;            //Salida. Orden retroceso Persiana
   OParoPer: BOOL;           //Salida. Orden paro Persiana

END_STRUCT
END_TYPE
```

Respecto al otro DB, “dbSecRobot”, es un aspecto que tendrán todas y cada unas de las secuencias que hemos programado. La estructura será siempre la misma, y para éste caso concreto es la siguiente:

```
DATA_BLOCK dbSecRobot  tdSecuencia
BEGIN
END_DATA_BLOCK
```

Donde "tdSecuencia" es un tipo de dato (UDT 101) para las secuencias:

```
TYPE tdSecuencia
STRUCT
   Etapa:INT:=-1;              //Salida. Etapa Actual de la Secuencia
   EtapaSiguiente:INT:=-1;     //Salida. Siguiente Etapa de la secuencia
   TemporizadorEtapa: INT;     //Entrada. Nº de temporizador a usar por la secuencia
   TiempoEtapa: INT;           //Salida. Tiempo de actividad de la etapa actual
   TiempoCiclo : TIME;         //Salida. Ultimo tiempo de ciclo

   TiempoAlarma : ARRAY [0..9] OF INT;          //Entrada. Matriz para los tiempos
de alamas
   Alarma : ARRAY [0..9] OF BOOL;               //Salida. Alarmas de Secuencia

   CondicionesIniciales : BOOL;    //Entrada. Condiciones Iniciales de la Secuencia
   FinCiclo: BOOL;                 //Entrada. Indicación de Fin de Ciclo
   EnCiclo: BOOL;                  //Entrada. Indicación de Ciclo en curso
   Automatico: BOOL;               //Secuencia en automático
END_STRUCT
END_TYPE
```

Así pues, una vez dicho lo anterior y siguiendo con la función "fcCtrlMesa", se observa, que antes de llamar a la función que ejecuta la secuencia que controla al robot propiamente dicho, lo primero que se hace es preparar las entradas (asignandolas al DB correspondiente), activar si procede la secuencia que controla el robot, indicando las condiciones iniciales que se deben cumplir y poner el motor del robot a on si también procede (RobotOutMon). Después, tras realizar la mencionada llamada, se asignan las salidas escritas en el DB del robot (dbRobot) a las salidas físicas del sistema.

fcSecRobot

A continuación mostramos en primer lugar el GRAFCET realizado para programar posteriormente la secuencia en SCL:

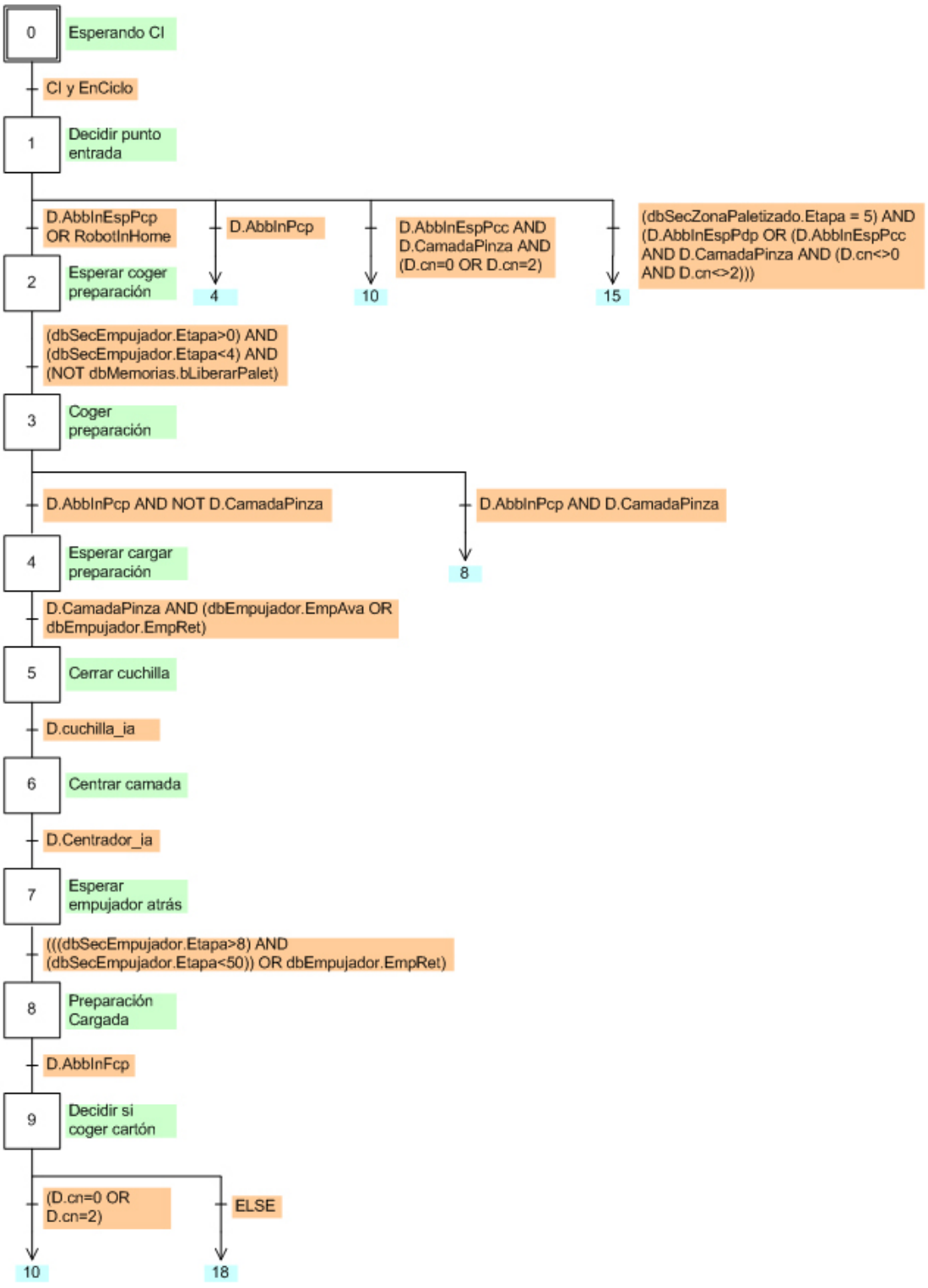


Figura 3.55. GRAFCET para secuencia 1

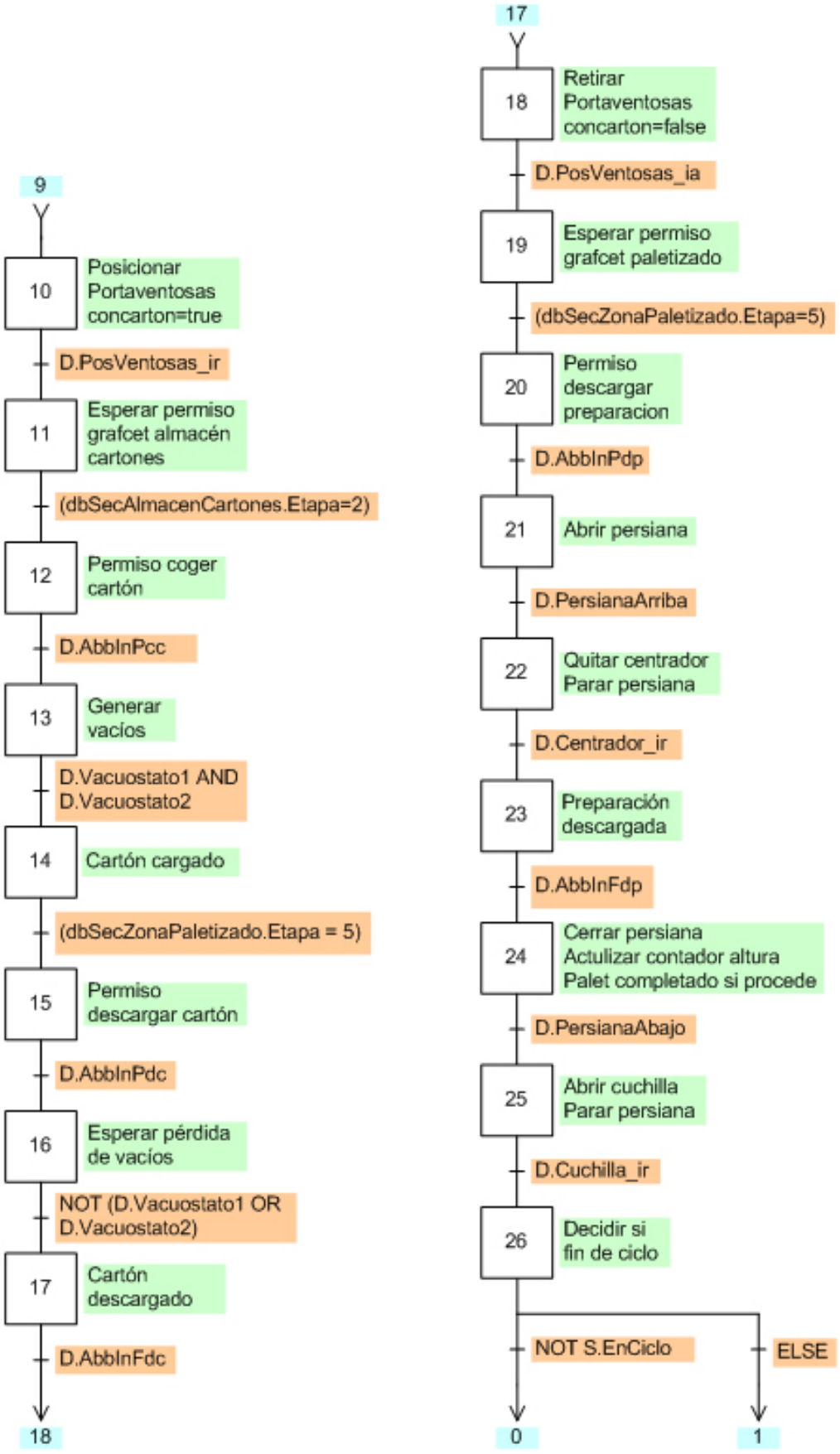


Figura 3.56. GRAFCET para secuencia 2

Para programar esto en SCL, se ha seguido la siguiente estructura genérica para secuencias:

```
FUNCTION fcSecNombre: VOID
VAR_IN_OUT
   S : tdSecuencia;    //Instancia de la Secuencia
   D : tdNombre;   //Instancia de los Elementos de Nombre
END_VAR
VAR_TEMP
  Variables temporales;
END_VAR
BEGIN
//Preliminar
//-------------------------------------------------------------------------
Acciones preliminares.
//Control de la Secuencia
IF (NOT S.Automatico) THEN
   S.Etapa := -1;
   S.EtapaSiguiente := -1;
END_IF;
//Acciones a la Activación
//-------------------------------------------------------------------------
Acciones a la activación de una etapa
//Acciones en continuo
//-------------------------------------------------------------------------
Acciones que ocurren durante la etapa activa
//Transiciones
//-------------------------------------------------------------------------
IF S.EnCiclo THEN

//Activación de la Secuencia
IF (S.Etapa = -1) AND S.Automatico THEN
   S.EtapaSiguiente := 0;
END_IF;
```

```
//Comprobación de Condiciones Iniciales (Siempre Igual)
IF S.Etapa = 0 THEN
   IF S.CondicionesIniciales AND S.EnCiclo THEN
      S.EtapaSiguiente := 1;     //Comenzamos a ejecutar la Secuencia
   END_IF;
END_IF;

….

IF S.Etapa = X THEN
   IF xxxx THEN
      S.EtapaSiguiente := X;
   END_IF;
END_IF;
…

END_IF;
//Acciones a la Desactivación
//-----------------------------------------------------------------------------
Acciones a la desactivación de la etapa
//Posterior
//-----------------------------------------------------------------------------
Acciones posteriores a la ejecución de la secuencia
END_FUNCTION
```

Para la secuencia del robot, el código SCL que le corresponde es el siguiente:

```
FUNCTION fcSecRobot : VOID
VAR_IN_OUT
   S : tdSecuencia;    //Instancia de la Secuencia
   D : tdRobot;   //Instancia de los Elementos de Robot
END_VAR
VAR_TEMP
  altura: INT;
END_VAR
BEGIN
//Preliminar
//-------------------------------------------------------------------------
altura := D.cn + 1;

D.bit0 := (altura=1) OR (altura=3) OR (altura=5);
D.bit1 := (altura=2) OR (altura=3);
D.bit2 := (altura=4) OR (altura=5);

RobotOutDPcp := D.CamadaPinza;

//Control de la Secuencia
IF (NOT S.Automatico) THEN
   S.Etapa := -1;
   S.EtapaSiguiente := -1;
END_IF;

//Acciones a la Activación
//-------------------------------------------------------------------------
IF (S.EtapaSiguiente=24) AND (S.Etapa <> 24) THEN
   D.cn := D.cn+1;
   IF D.cn>=5 THEN
     D.cn := 0;
     dbMemorias.PaletCompletado := true;
```

```
   END_IF;
END_IF;

//Actualización de Etapa
//-------------------------------------------------------------------------
IF S.Etapa <> S.EtapaSiguiente THEN
   S.Etapa := S.EtapaSiguiente;
   S.TiempoEtapa := 0;
END_IF;
IF reMarcaDecimas THEN
   S.TiempoEtapa := S.TiempoEtapa + 1;
END_IF;

//Acciones en continuo
//-------------------------------------------------------------------------
D.AbbOutPcp := (S.Etapa=3);                  //Permiso cargar preparación

D.Cuchilla_aa := (S.Etapa=5);                //Cerrar cuchilla
D.Cuchilla_ar := (S.Etapa=25);               //Abrir cuchilla

D.Centrador_aa := (S.Etapa=6);               //Centrar camada
D.Centrador_ar := (S.Etapa=22);              //Quitar centrajes camada

D.AbbOutPc := (S.Etapa=8);                   //Preparación cargada

D.PosVentosas_aa := (S.Etapa=18);            //Retirar Portaventosas
D.PosVentosas_ar := (S.Etapa=10);            //Posicionar Portaventosas

D.AbbOutCc := (S.Etapa=14);                  //Cartón cargado
D.AbbOutPd := (S.Etapa=23);                  //Preparación descargada
D.AbbOutPcc := (S.Etapa=12);                 //Permiso cargar cartón
D.AbbOutPdp := (S.Etapa=20);                 //Permiso descargar preparacion

D.AbbOutPdc := (S.Etapa=15);                 //Permiso descargar cartón
```

```
D.AbbOutCd := (S.Etapa=17);                    //Cartón descargado

IF S.Etapa = 10 THEN
   D.concarton := TRUE;
ELSIF S.Etapa = 18 THEN
   D.concarton := FALSE;
END_IF;

IF S.Etapa = 13 THEN
   D.Vacio1 := TRUE;
   D.Vacio2 := TRUE;
ELSIF S.Etapa = 16 THEN
   D.Vacio1 := FALSE;
   D.Vacio2 := FALSE;
END_IF;

D.OAvaPer := (S.Etapa=21);
D.ORetPer := (S.Etapa=24);
D.OParoPer:= (S.Etapa=22) OR (S.Etapa=25);

//Transiciones
//--------------------------------------------------------------------------
IF S.EnCiclo THEN

//Activación de la Secuencia
IF (S.Etapa = -1) AND S.Automatico THEN
   S.EtapaSiguiente := 0;
END_IF;

//Comprobación de Condiciones Iniciales (Siempre Igual)
IF S.Etapa = 0 THEN
   IF S.CondicionesIniciales AND S.EnCiclo THEN
      S.EtapaSiguiente := 1;      //Comenzamos a ejecutar la Secuencia
   END_IF;
```

```
END_IF;

IF S.Etapa = 1 THEN
  IF D.AbbInEspPcp OR RobotInHome THEN
    S.EtapaSiguiente := 2;          //Robot esperando una carga de preparación
  ELSIF D.AbbInPcp THEN
    S.EtapaSiguiente := 4;          //Robot en posición carga de preparación
  ELSIF D.AbbInEspPcc AND D.CamadaPinza AND (D.cn=0 OR D.cn=2) THEN
    S.EtapaSiguiente := 10;         //Robot esperando cargar cartón y se debe cargar
  ELSIF (dbSecZonaPaletizado.Etapa = 5) AND (D.AbbInEspPdp OR (D.AbbInEspPcc
AND D.CamadaPinza AND (D.cn<>0 AND D.cn<>2))) THEN
    S.EtapaSiguiente := 15;         //Llevar robot a posición descargar camada
  END_IF;
END_IF;

IF S.Etapa = 2 THEN
    IF (dbSecEmpujador.Etapa>0) AND (dbSecEmpujador.Etapa<4) AND (NOT
dbMemorias.bLiberarPalet) THEN
        S.EtapaSiguiente := 3;      //Permiso a robot para cargar preparación
    END_IF;
END_IF;

IF S.Etapa = 3 THEN
    IF D.AbbInPcp AND NOT D.CamadaPinza THEN
        S.EtapaSiguiente := 4;   //Abrir cuchilla.
    ELSIF D.AbbInPcp AND D.CamadaPinza THEN
        S.EtapaSiguiente := 8;   //Jaula ya estaba cargada seguir con la secuencia
    END_IF;
END_IF;

IF S.Etapa = 4 THEN
    IF D.CamadaPinza AND (dbEmpujador.EmpAva OR dbEmpujador.EmpRet) THEN
        S.EtapaSiguiente := 5;      //Cerrar cuchilla
    END_IF;
```

```
END_IF;

IF S.Etapa = 5 THEN
   IF D.cuchilla_ia THEN
      S.EtapaSiguiente := 6;     //Cerrar centradores
   END_IF;
END_IF;

IF S.Etapa = 6 THEN
   IF D.centrador_ia THEN
      S.EtapaSiguiente := 7;     //Esperar permiso grafcet empujador para sacar jaula
   END_IF;
END_IF;

IF S.Etapa = 7 THEN
   IF (((dbSecEmpujador.Etapa>8) AND (dbSecEmpujador.Etapa<50)) OR
dbEmpujador.EmpRet) THEN
      S.EtapaSiguiente := 8;     //La carga se ha efectuado
   END_IF;
END_IF;

IF S.Etapa = 8 THEN
   IF D.AbbInFcp THEN
      S.EtapaSiguiente := 9;     //Decidir si hay que coger cartón
   END_IF;
END_IF;

IF S.Etapa = 9 THEN
   IF (D.cn=0 OR D.cn=2) THEN
      S.EtapaSiguiente := 10;     //Posicionar portaventosas
   ELSE
      S.EtapaSiguiente := 18;     //Retirar portaventosas
   END_IF;
END_IF;
```

```
IF S.Etapa = 10 THEN
   IF D.PosVentosas_ir THEN
      S.EtapaSiguiente := 11;     //Esperar permiso grafcet almacén cartones
   END_IF;
END_IF;

IF S.Etapa = 11 THEN
   IF (dbSecAlmacenCartones.Etapa=2) THEN
      S.EtapaSiguiente := 12;     //Permiso coger cartón
   END_IF;
END_IF;

IF S.Etapa = 12 THEN
   IF D.AbbInPcc THEN
      S.EtapaSiguiente := 13;     //Generar vacíos
   END_IF;
END_IF;

IF S.Etapa = 13 THEN
   IF D.Vacuostato1 AND D.Vacuostato2 THEN
      S.EtapaSiguiente := 14;    //Esperar Fin de Operación carga cartón
   END_IF;
END_IF;

IF S.Etapa = 14 THEN
   IF (dbSecZonaPaletizado.Etapa = 5) THEN
      S.EtapaSiguiente := 15;    //Permiso descargar cartón
   END_IF;
END_IF;

IF S.Etapa = 15 THEN
   IF D.AbbInPdc THEN
      S.EtapaSiguiente := 16;     //Esperar pérdida de vacíos
   END_IF;
```

```
END_IF;

IF S.Etapa = 16 THEN
   IF NOT (D.Vacuostato1 OR D.Vacuostato2) THEN
      S.EtapaSiguiente := 17;     //Espera Fin descarga cartón del robot
   END_IF;
END_IF;

IF S.Etapa = 17 THEN
   IF D.AbbInFdc THEN
      S.EtapaSiguiente := 18;     //Retirar portaventosas
   END_IF;
END_IF;

IF S.Etapa = 18 THEN
   IF D.PosVentosas_ia THEN
      S.EtapaSiguiente := 19;     //Esperar permiso grafcet paletizado
   END_IF;
END_IF;

IF S.Etapa = 19 THEN
   IF (dbSecZonaPaletizado.Etapa=5) THEN
      S.EtapaSiguiente := 20;     //Esperar robot en posición descarga preparación
   END_IF;
END_IF;

IF S.Etapa = 20 THEN
   IF D.AbbInPdp THEN
      S.EtapaSiguiente := 21;     //Abrir persiana
   END_IF;
END_IF;


IF S.Etapa = 21 THEN
```

```
   IF D.PersianaArriba THEN
      S.EtapaSiguiente := 22;     //Quitar centrador
   END_IF;
END_IF;

IF S.Etapa = 22 THEN
   IF D.Centrador_ir THEN
      S.EtapaSiguiente := 23;     //Espera robot en posición segura para reposicionar la
persiana
   END_IF;
END_IF;

IF S.Etapa = 23 THEN
   IF D.AbbInFdp THEN
      S.EtapaSiguiente := 24;     //Cerrar persiana
   END_IF;
END_IF;

IF S.Etapa = 24 THEN
   IF D.PersianaAbajo THEN
      S.EtapaSiguiente := 25;     //Cerrar cuchilla
   END_IF;
END_IF;

IF S.Etapa = 25 THEN
   IF D.Cuchilla_ir THEN
      S.EtapaSiguiente := 26;     //Decidir si fin de ciclo
   END_IF;
END_IF;

IF S.Etapa = 26 THEN
   IF NOT S.EnCiclo THEN
      S.EtapaSiguiente := 0;     //Detener secuencia
   ELSE
```

```
      S.EtapaSiguiente := 1;      //Continuar secuencia
   END_IF;
END_IF;

END_IF;

//Acciones a la Desactivación
//----------------------------------------------------------------------------

//Posterior
//----------------------------------------------------------------------------
D.SegPersiana := (S.Automatico) AND (S.Etapa <> -1) AND (D.PersianaAbajo);
END_FUNCTION
```

Poco más que comentar en este aspecto. Se comprueba que gracias a la realización de DB's independientes, se puede acceder desde cualquier módulo a cualquiera de ellos para de ésta forma estar interconectados. Es el ejemplo que se puede visualizar en esta función donde en más de una ocasión es necesario comprobar el estado donde se encuentra ejecutándose el grafcet de otra secuencia para poder actuar en consecuencia. Por otro lado, para el resto de secuencias, la forma de programar es exactamente igual, por lo que el código del resto de funciones lo dejamos expuesto en el ANEXO y así no extendernos demasiado en algo innecesario.

.