UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

AUTOMATIZACIÓN DEL PALETIZADO DE UNA LÍNEA DE CAJAS DE BOTELLAS

Autor: **Ignacio Rodríguez Planas**

Tutor: **Eduardo Fernández Camacho**

A mis padres y mis hermanos, por todo lo que han aguantado.
A mi tío Pedro, que tanto se ha interesado.
Y por supuesto, a mis amigos, que siempre me ayudaron.

ÍNDICE DEL PROYECTO

CAPITULO 1 Introducción y objetivos del proyecto 1

1.1 Introducción a los autómatas programables 3

CAPITULO 2 Memoria Descriptiva 9

2.1 Descripción funcional del sistema 9

2.2 Seguridades de la instalación 14

2.3 Arquitectura del sistema 16

2.4 Armarios, pupitres y cajetines del sistema 26

2.5 Entradas y salidas digitales 32

2.5.1 Entradas y salidas digitales: PLC 33

2.5.2 Entradas y salidas digitales: ET200S 37

2.5.3 Entradas y salidas digitales: Robot ABB 48

2.6 Comunicación entre los periféricos: PROFIBUS-DP 50

2.6.1 Topología de BUS 52

2.6.2 El variador de frecuencia VLT 2800 54

2.7 Datos Técnicos del Hardware 62

2.7.1 PLC 62

2.7.2. ET200 66

2.7.3 VLT2800 71

2.7.4 TP 270 72

2.8. Software utilizado 73

CAPITULO 3 Desarrollo del proyecto 76

3.1 Introducción 76

3.2 Introducción a STEP 7 **76**

3.2.1. Estructura del Proyecto **83**

3.2.2. Configuración del Hardware **87**

3.3. Configuración de la pantalla TP270 **91**

3.3.1. Introducción a PROTOOL **91**

3.3.2. Pantallas del TP270 **98**

3.4. Programa S7 **147**

3.4.1. Introducción a SCL **147**

3.4.2. Estructura del programa S7 **149**

3.4.3. Programación del OB 1 **153**

3.4.4. Programación de secuencias **169**

CAPITULO 4 Conclusiones y propuestas de mejora **188**

BIBLIOGRAFÍA **190**

ANEXO Listado del código **191**

A.1 DB's y UDT's en SCL **191**

A.2 FC's en SCL **212**

ÍNDICE DE FIGURAS

Figura 1.1. Estructura básica de un PLC 4

Figura 2.1. Formatos y formación de las cajas. 10

Figura 2.2. Palet de tamaño europeo. 10

Figura 2.3 Estructura genérica de un palet completado. 12

Figura 2.4. Zonas de la planta de paletizado 13

Figura 2.5. Formación de filas en la mesa de formación. 18

Figura 2.6 Arquitectura completa de la instalación 24

Figura 2.7. Interior del Armario 27

Figura 2.8. Exterior del Armario 28

Figura 2.9. Localización del Armario, Pupitres y Cajetines. 31

Figura 2.10 Transporte de cajas 40

Figura 2.11. Mesa de Formación 40

Figura 2.12. Pinza del robot 43

Figura 2.13. Transporte de palets 45

Figura 2.14. Almacén de cartones 47

Figura 2.15. Esquema PROFIBUS 52

Figura 2.16. Tipos de PPO 56

Figura 2.17. PCA 57

Figura 2.18. CPU 315-2DP 63

Figura 2.19. SM31 65

Figura 2.20. SM322 66

Figura 2.21. ET200 68

Figura 2.22. IM 151-1 Standard 68

Figura 2.23. PM-E 24 V DC 69

Figura 2.24. VLT2800 71

Figura 2.25. Pantalla TP270 de SIEMENS 72

Figura 3.1. Administrador SIMATIC 81

Figura 3.2. HW Config 87

Figura 3.3. Ciclo/Marca Ciclo 89

Figura 3.4. PROTOOL/PRO CS 92

Figura 3.5. Pantalla base del HMI 98

Figura 3.6. Estructura de las pantallas HMI 1 100

Figura 3.7. Estructura de las pantallas HMI 2 101

Figura 3.8. Pantalla de Presentación 102

Figura 3.9. Pantalla de Menú Principal 103

Figura 3.10. Pantalla de Zonas de Movimientos Manuales 104

Figura 3.11. Instalación Neumática 105

Figura 3.12. Pinza Robot 106

Figura 3.13. Portaventosas Cartones 107

Figura 3.14. Pinza Componentes 108

Figura 3.15. Persiana Componentes 109

Figura 3.16. Entrada Cajas 16 y 17 110

Figura 3.17. Empujador 111

Figura 3.18. Preparación 112

Figura 3.19. Entrada Cajas 14 y 15 113

Figura 3.20. Topes de Giro 114

Figura 3.21. Desviador 2 115

Figura 3.22. Entrada Cajas 11, 12 y 13 116

Figura 3.23. Desviador 1 117

Figura 3.24. Tope entrada 118

Figura 3.25. Giro Entrada 119

Figura 3.26. Preparación Palets 35 y 36 120

Figura 3.27. Transferencia de Salida 121

Figura 3.28. Salida de Palet 122

Figura 3.29. Preparación Palets 33 y 34 123

Figura 3.30. Paletizado 124

Figura 3.31. Transferencia de entrada 125

Figura 3.32. Preparación Palets 31 y 32 126

Figura 3.33. Dosificación de Palets 127

Figura 3.34. Carga de Palets 128

Figura 3.35. Almacén Cartones 129

Figura 3.36. Empujador de Cartones 130

Figura 3.37. Centrador del almacén de cartones 131

Figura 3.38. Modos de Trabajo 132

Figura 3.39. Planta de Paletizado 133

Figura 3.40. Transporte de palets 134

Figura 3.41. Liberación de Palet 135

Figura 3.42. Almacén de Cartones 136

Figura 3.43. Dosificador de Entrada 137

Figura 3.44. Pantalla de Introducción de Contraseña 138

Figura 3.45. Modificar Contadores 138

Figura 3.46. Datos de Gestión 139

Figura 3.47. Pantalla de Alarmas del Sistema vacía 141

Figura 3.48. Pantalla de Alarmas del Sistema con Avisos 142

Figura 3.49. Configurar Datos 143

Figura 3.50. Modificación Directa 144

Figura 3.51. Gestión de claves 145

Figura 3.52. Ajuste de Fecha y Hora 146

Figura 3.53. Lista de símbolos 150

Figura 3.54. Llamadas del fcGeneral 154

Figura 3.55. GRAFCET para secuencia 1 175

Figura 3.56. GRAFCET para secuencia 2 176

ÍNDICE DE TABLAS

Tabla 2.1. Resumen de elementos 25

Tabla 2.2. Número de entradas y salidas 33

Tabla 2.3. Entradas PLC 35

Tabla 2.4. Salidas PLC 36

Tabla 2.5. Entradas ET200[1] 39

Tabla 2.6. Salidas ET200[1] 40

Tabla 2.7. Entradas ET200[2] 42

Tabla 2.8. Salidas ET200[2] 42

Tabla 2.9. Entradas ET200[3] 44

Tabla 2.10. Salidas ET200[3] 45

Tabla 2.11. Entradas ET200[4] 46

Tabla 2.12. Salidas ET200[4] 47

Tabla 2.13. Entradas al PLC del Robot 48

Tabla 2.14. Salidas del PLC hacia el Robot 49

Tabla 2.15. Valores del CTW 60

Tabla 2.16. Valores del STW 61

Tabla 3.1 Frecuencias posibles 90

Tabla 3.2. Lista “PARÁMETROS” 140

Tabla 3.3. Lista “LST_FORMATOS” 140

Tabla 3.4. Lista “SI_NO” 141

Tabla 3.5. Bloques S7 152

CAPITULO 1

Introducción y objetivos del proyecto

La realización del siguiente proyecto fin de carrera está realizada en colaboración con la empresa MP Productividad, y tiene como objeto el control y la supervisión de una línea de paletizado de cajas de botellas, de una fábrica del sector, que por razones de confidencialidad, no nos está permitido nombrar al cliente.

A este sistema, del que le llegaran una serie de cajas de botellas de diversos tamaños de una línea exterior, deberá producir 12000 botellas a la hora, y devolverlas todas ellas, en palets ordenados en diversos formatos, según el tamaño que se esté usando de cada caja, a otra línea exterior, con la cual nuestro sistema se deberá comunicar. Los tamaños de dichas cajas varían en función del número de botellas que posean en su interior. Las cajas podrán contener 4, 6 u 8 botellas en su interior.

Para gobernar todo este sistema, se usará el autómata S7-300 de la compañía SIEMENS, mientras que para el control y supervisión del mismo, nos ayudaremos de un panel táctil y otro de texto, que fabrica igualmente SIEMENS: El TP270 y el TD17. Aunque sobre la arquitectura del sistema, hablaremos con más detalle en otro capítulo posterior, también queremos reseñar que la línea contará con tres partes claramente diferenciadas y separadas tanto lógica como físicamente, y para la conexión de ambas partes se utilizará un puesto robotizado de la marca ABB ROBOTICS. Se trata de un S4CPLUS, de la serie 7600, de seis ejes y sin ejes externos conectados.

El sistema estará gobernando por un total de más de 200 señales digitales, distribuidas según una periferia descentralizada a lo largo de toda la instalación. La estructura se dividirá en diversas zonas a saber:

- *Transportes de cajas*: Esta primera zona será una línea de transportes desde donde entrarán las cajas al sistema. Se comunica físicamente con la *mesa de formación*.

- *Mesa de formación*: En esta zona, las cajas serán aceleradas, desplazadas y empujadas para poder ordenarlas convenientemente para formar camadas completas y poder situarlas en los palets.
- *Almacén de palets*: Donde se almacenarán los palets. Físicamente será una zona distinta a la formada por la mesa de formación y el transporte de cajas. Proporciona los palets que transportarán los *transportadores de rodillos*.

- *Transportes de rodillos*: Llevarán los palets hasta el exterior. Se comunicará con el *robot* para que éste pueda depositar las distintas camadas de cajas de botellas para formar un palet completo.

- *Almacén de cartones*: Es otra zona diferenciada, donde se almacenarán los cartones que se incluirán en los palets.

- *Robot*: Como ya hemos mencionado, es el elemento que comunica las distintas zonas. Se encargará de recoger camadas de la *mesa de formación*, coger cartones cuando sea necesario del *almacén de cartones* y depositarlos en los palets que se encuentran en el *transporte de rodillos*.

Aunque el proyecto realizado por la empresa MP Productividad abarca numerosos temas: diseño y producción mecánico de las diversas piezas, diseño y montaje eléctrico, etc… El objeto de este proyecto fin de carrera se centrará en la programación del autómata S7-300 y su comunicación con los diversos elementos que lo forman, así como de las pantallas de supervisión y control que llevará el panel táctil TP 270.

Es necesario indicar por otra parte, que el sistema realizado fue simulado en los talleres de la empresa MP Productividad, pero el desarrollo final del mismo no ha podido ser verificado, pues el destino final del proyecto completo es la fábrica de nuestro cliente anónimo, proyecto que sigue en construcción durante la redacción de éste documento. Es por tanto, que ciertas partes que durante el desarrollo del proyecto detallaremos, no han sido terminadas completamente, es el caso por ejemplo del dar valores iniciales a diversas variables, por ejemplo. Pero repetimos, en el capítulo donde

se explica detalladamente el desarrollo de este proyecto fin de carrera, los explicaremos con mayor detalle.

Con esto, los objetivos que perseguimos cumplir con la realización de éste trabajo serían los siguientes:

- Intensificar al detalle los conocimientos de un PLC comercial común, en este caso, el S7-300 de SIEMENS. Los PLC's son de suma importancia para cualquier aplicación robótica, ya que éstos son los encargados de gobernar el sistema completo. Pensamos que es primordial el conocimiento en este campo para un ingeniero de telecomunicación con la intensificación de robótica.
- Estudio de una planta industrial, en este caso, el de una línea de paletizado, real, con todos los elementos que estos conllevan: motores, válvulas, sensores, cintas transportadoras, etc..
- Estudio de una aplicación HMI, necesarias para cualquier sistema automatizado. En la robótica industrial, el interfaz Hombre-Máquina es indispensable para poder controlar y supervisar adecuadamente el sistema a realizar. Durante la carrera, apenas se ha tratado nada sobre este asunto tan importante. Con el estudio en este caso del panel táctil de la serie 270 de SIEMENS, creemos que conseguiremos adquirir nuevos conocimientos que durante la carrera se nos han privado, quizás por falta de tiempo.
- Por último, con este proyecto pensamos que podremos aplicar diversos conocimientos adquiridos durante la carrera, a una situación real.

1.1 Introducción a los autómatas programables

Definición de autómata programable.

Entendemos por Autómata Programable, o PLC (Controlador Lógico Programable), toda máquina electrónica, diseñada para controlar en tiempo real y en medio industrial procesos secuenciales. Su manejo y programación puede ser realizada por personal eléctrico o electrónico sin conocimientos informáticos. Realiza funciones

lógicas: series, paralelos, temporizaciones, contajes y otras más potentes como cálculos, regulaciones, etc.

Otra definición de autómata programable sería una «caja» en la que existen, por una parte, unos terminales de entrada (o captadores) a los que se conectan pulsadores, finales de carrera, fotocélulas, detectores...; y por otra, unos terminales de salida (o actuadores) a los que se conectarán bobinas de contactores, electroválvulas, lámparas..., de forma que la actuación de estos últimos está en función de las señales de entrada que estén activadas en cada momento, según el programa almacenado.

La función básica de los autómatas programables es la de reducir el trabajo del usuario a realizar el programa, es decir, la relación entre las señales de entrada que se tienen que cumplir para activar cada salida, puesto que los elementos tradicionales (como relés auxiliares, de enclavamiento, temporizadores, contadores...) son internos.

Estructura de un autómata programable.

La estructura básica de un autómata programable es la siguiente:

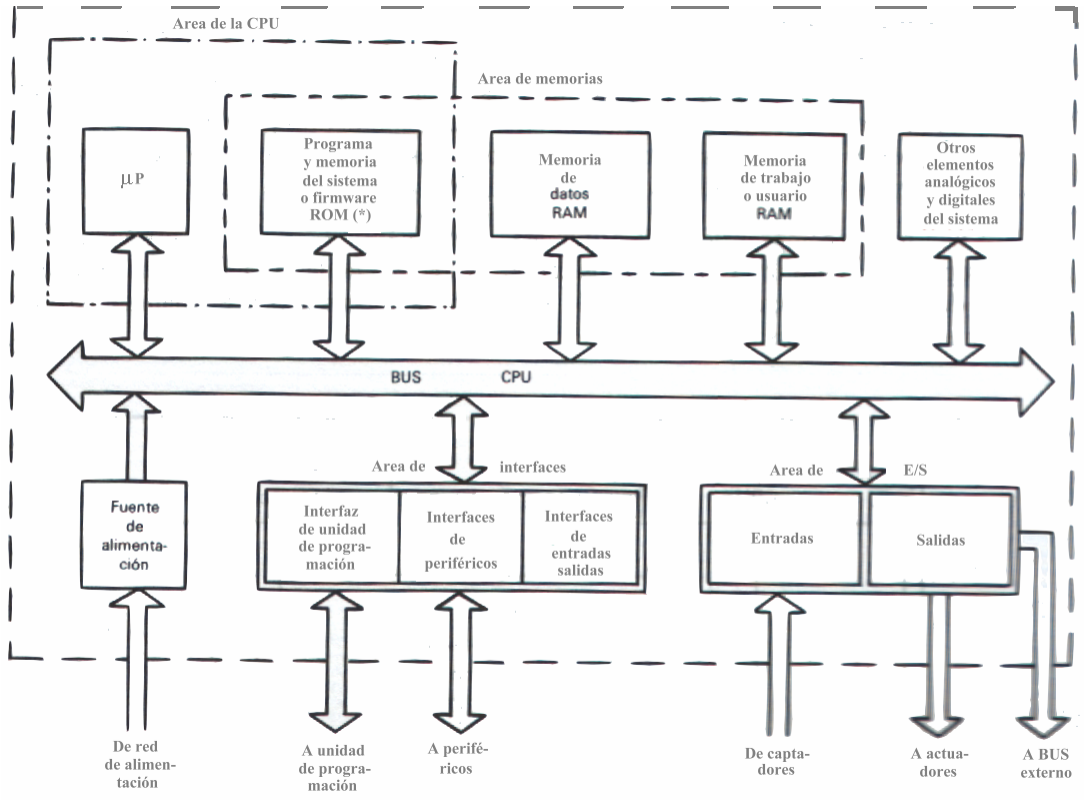


Figura 1.1. Estructura básica de un PLC

- **Fuente de alimentación:**

Es la encargada de convertir la tensión de la red, 220v corriente alterna, a baja tensión de corriente continua, normalmente a 24v. Siendo esta la tensión de trabajo en los circuitos electrónicos que forma el Autómata.

- **Unidad Central de Procesos o CPU:**

Se encarga de recibir las órdenes del operario por medio de la consola de programación y el módulo de entradas. Posteriormente las procesa para enviar respuestas al módulo de salidas. En su memoria se encuentra residente el programa destinado a controlar el proceso.

Contiene las siguientes partes:

  - Unidad central o de proceso
  - Temporizadores y contadores
  - Memoria de programa
  - Memoria de datos
  - Memoria imagen de entrada
  - Memoria de salida

- **Módulo de entrada:**

Es al que se unen los captadores (interruptores, finales de carrera, pulsadores,...).

Cada cierto tiempo el estado de las entradas se transfiere a la memoria imagen de entrada. La información recibida en ella, es enviada a la CPU para ser procesada de acuerdo a la programación.

Se pueden diferenciar dos tipos de captadores conectables al módulo de entradas: los *pasivos* y los *activos*.

Los *captadores pasivos* son los que cambian su estado lógico (activado o no activado) por medio de una acción mecánica. Estos son los interruptores, pulsadores, finales de carrera,...

Los *captadores activos* son dispositivos electrónicos que suministran una tensión al autómata, que es función de una determinada variable.

- **Módulo de salidas:**

Es el encargado de activar y desactivar los actuadores (bobinas de contactores, lámparas, motores pequeños,...)

La información enviada por las entradas a la CPU, una vez procesada, se envía a la memoria imagen de salidas, de donde se envía a la interface de salidas para que estas sean activadas y a la vez los actuadores que en ellas están conectados.

Según el tipo de proceso a controlar por el autómata, podemos utilizar diferentes módulos de salidas. Existen tres tipos bien diferenciados:

  - A relés: son usados en circuitos de corriente continua y corriente alterna. Están basados en la conmutación mecánica, por la bobina del relé, de un contacto eléctrico normalmente abierto.

  - A triac: se utilizan en circuitos de corriente continua y corriente alterna que necesitan maniobras de conmutación muy rápidas.

  - A transistores a colector abierto: son utilizados en circuitos que necesiten maniobras de conexión / desconexión muy rápidas. El uso de este tipo de módulos es exclusivo de los circuitos de corriente continua.

- **Terminal de programación:**

El terminal o consola de programación es el que permite comunicar al operario con el sistema.

Las funciones básicas de éste son las siguientes:

  - Transferencia y modificación de programas.

  - Verificación de la programación.

  - Información del funcionamiento de los procesos.

Como consolas de programación pueden ser utilizadas las construidas específicamente para el autómata, tipo calculadora o bien un ordenador personal, PC,

que soporte un software específicamente diseñado para resolver los problemas de programación y control.

- **Periféricos:**

Los periféricos no intervienen directamente en el funcionamiento del autómata, pero sin embargo facilitan la labor del operario.

Los más utilizados son:

- Grabadoras a cassettes.
- Impresoras.
- Cartuchos de memoria EPROM.
- Visualizadores y paneles de operación OP.
- Memorias EEPROM.

Lenguaje de programación

Cuando surgieron los autómatas programables, lo hicieron con la necesidad de sustituir a los enormes cuadros de maniobra construidos con contactores y relés. Por lo tanto, la comunicación hombre-máquina debería ser similar a la utilizada hasta ese momento. El lenguaje usado, debería ser interpretado, con facilidad, por los mismos técnicos electricistas que anteriormente estaban en contacto con la instalación. Estos lenguajes han evolucionado, en los últimos tiempos, de tal forma que algunos de ellos ya no tienen nada que ver con el típico plano eléctrico a relés.

Los lenguajes más significativos son:

- **Lenguaje a contactos:**

Es el que más similitudes tiene con el utilizado por un electricista al elaborar cuadros de automatismos. Muchos autómatas incluyen módulos especiales de software para poder programar gráficamente de esta forma.

- **Lenguaje por lista de instrucciones:**

En los autómatas de gama baja, es el único modo de programación. Consiste en elaborar una lista de instrucciones o nemónicos que se asocian a los símbolos y su combinación en un circuito eléctrico a contactos. También decir, que este tipo de lenguaje es, en algunos casos, la forma más rápida de programación e incluso la más potente.

- **GRAFCET** (Gráfico Funcional de Etapas y Transiciones):

Ha sido especialmente diseñado para resolver problemas de automatismos secuenciales. Las acciones son asociadas a las etapas y las condiciones a cumplir a las transiciones. Este lenguaje resulta enormemente sencillo de interpretar por operarios sin conocimientos de automatismos eléctricos. Muchos de los autómatas que existen en el mercado permiten la programación en GRAFCET, tanto en modo gráfico o como por lista de instrucciones. También podemos utilizarlo para resolver problemas de automatización de forma teórica y posteriormente convertirlo a plano de contactos.

- **Plano de funciones lógicas:**

Resulta especialmente cómodo de utilizar, a técnicos habituados a trabajar con circuitos de puertas lógicas, ya que la simbología usada en ambos es equivalente.

Forma de funcionamiento del autómata. Concepto de ejecución cíclica.

La mayoría de los autómatas actuales se basan en el concepto de la ejecución cíclica de las instrucciones ubicadas en su memoria.

El programa es una serie de instrucciones grabadas en la memoria, un ciclo de proceso consiste inicialmente en la consideración de una serie de entradas que seguidamente serán fijadas para todo el ciclo. Después, el autómata ejecuta una instrucción tras otra hasta finalizar el programa y finalmente se definen las ordenes a aplicar sobre las salidas. El ciclo se reproduce así indefinidamente.

CAPITULO 2

Memoria Descriptiva

2.1 Descripción funcional del sistema

El sistema estará dividido en varias zonas bien diferenciadas desde el punto de vista funcional que desarrollan las necesidades del proyecto.

Transportes de cajas:

Su función es la de proporcionar la entrada de cajas de un determinado tamaño al sistema. Esta parte estará compuesta físicamente por tres transportadores de rodillos, con un funcionamiento lógico independiente cada una de ellas. Mientras entren cajas al sistema, estos transportadores estarán en funcionamiento, mientras que si dejan de pasar durante un cierto período de tiempo, dejarán de funcionar y por tanto se pararán.

Mesa de formación:

Su función es la de organizar las cajas según el formato seleccionado, para poder formar una camada y almacenarla posteriormente en palets. Hay tres formatos distintos, según el tamaño de las cajas de entrada:

- Formato 2x2: Las cajas tienen un tamaño de 204x204x298 mm. x mm. x mm. Cada una de ellas contiene 4 botellas en su interior.
- Formato 2x3: Las cajas tienen un tamaño de 204x306x298 mm. x mm. x mm. Cada una de ellas contiene 6 botellas en su interior.
- Formato 2x4: Las cajas tienen un tamaño de 204x408x298 mm. x mm. x mm. Cada una de ellas contiene 8 botellas en su interior.

Los palets usados son de tamaño Europeo (800x1200 mm. x mm.) de 4Kg de peso, por lo que las cajas deben organizarse de distinta forma según el formato elegido para poder formar una camada. La estructura de cada una de estas camadas lo podemos visualizar en la figura 2.1.

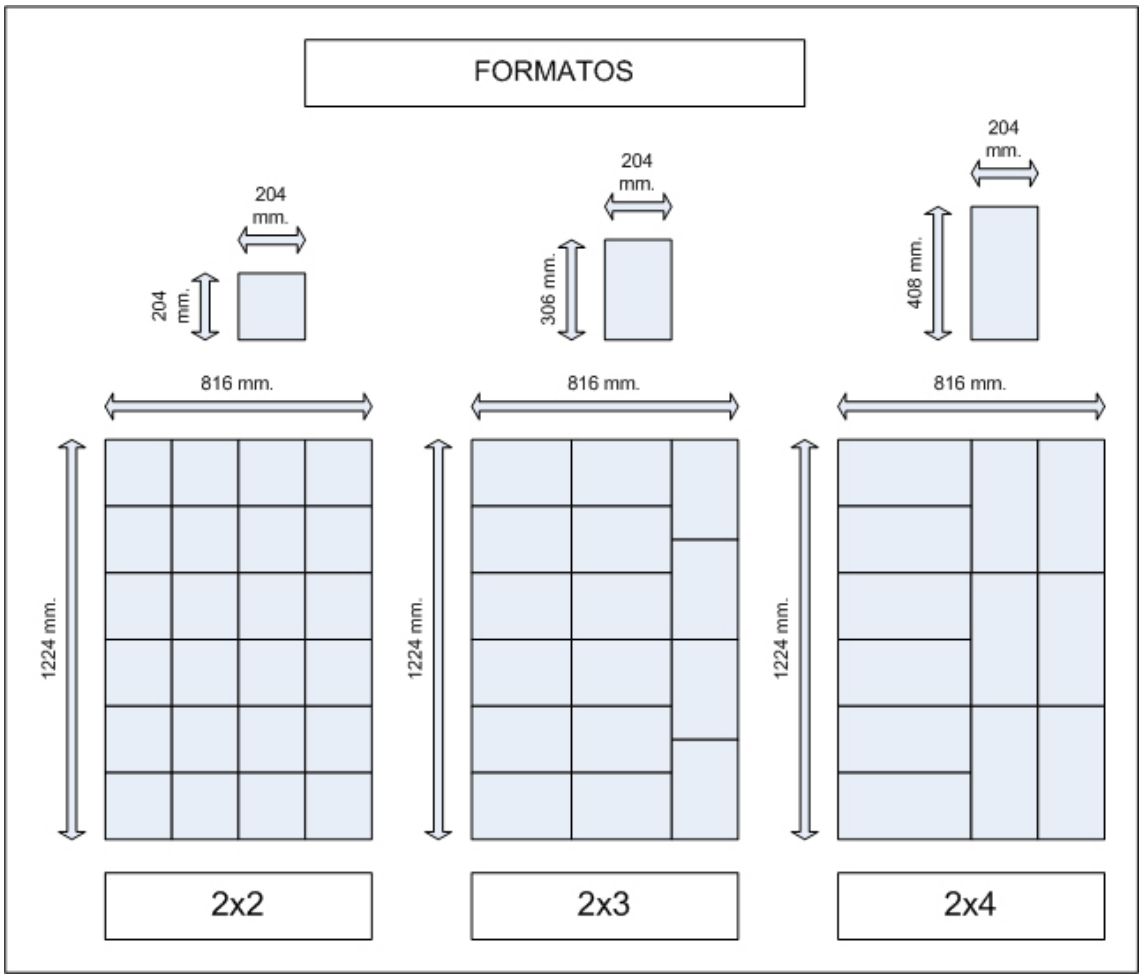


Figura 2.1. Formatos y formación de las cajas.

Figura 2.2. Palet de tamaño europeo.

Dado que las camadas son de mayor tamaño que el palet, se observa que la carga sobrepasará 12 mm. por cada lado de ancho y 8 mm. en cada lado de largo.

Robot

Se encarga de recoger cada camada formada en la mesa de formación y posicionarlas en el palet hasta completarlos. Así mismo, también se encarga de recoger cartones del almacén y posicionarlo adecuadamente según el formato elegido. Como ya se comentó, esta parte está compuesta físicamente por un robot ABB de 400Kg de carga en muñeca y una garra de 175 Kg. dotado de un cabezal para captura de camada completa. El esquema de paletización para cada formato en este caso es el siguiente:

- Formato 2x2: Cada palet llevará 120 cajas, distribuidas en 5 camadas de 24 cajas (la distribución de cada camada es la que se forma en la mesa de formación). Debajo de la primera y la tercera camada se colocará un cartón de tamaño 816x1224 (mm. x mm.) y un espesor de 3 mm., por lo que cada palet completo llevará un total de 2 cartones.

- Formato 2x3: El formato será exactamente igual, pero con un número de cajas inferior al anterior caso. Por tanto, la estructura del palet será de 5 camadas de 16 cajas cada una. Por lo que llevará un total de 80 cajas y 2 cartones.

- Formato 2x4: 60 cajas distribuídas en 5 camadas de 12 cajas cada una. Igualmente, cada palet llevará cartones en las mismas posiciones antes indicadas.

En la figura 2.3 podemos observar gráficamente el perfil de un palet completado genérico, independientemente del formato elegido.

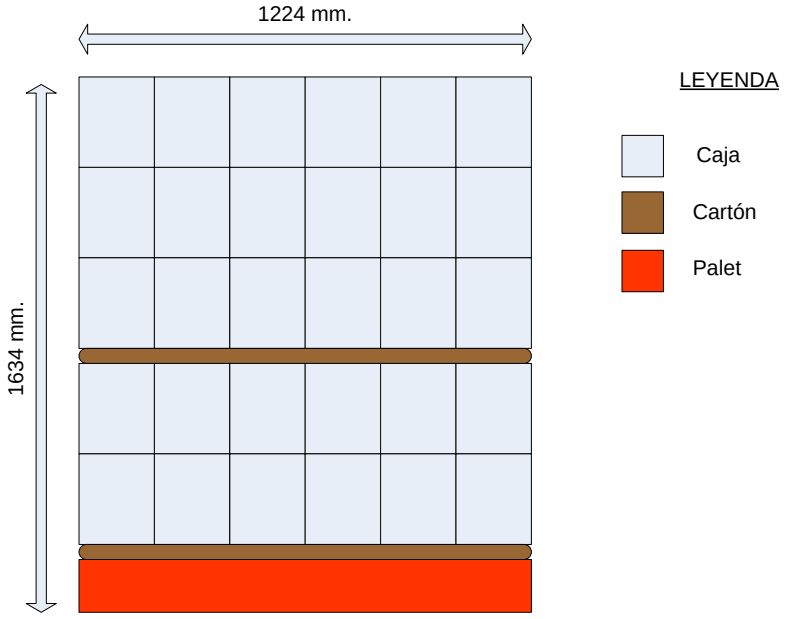


Figura 2.3 Estructura genérica de un palet completado.

Almacén de palets

Se encargar de dosificar los palets. Recoge palets de un almacén exterior y los va dosificando para que el robot vaya colocando las camadas correspondientes en cada uno de ellos hasta completarlos. Reseñar que el mantenimiento de dicho almacén exterior para que haya palets disponibles, es responsabilidad del encargado de mantenimiento del sistema.

Transportes de rodillos

Su función es la de proporcionar la salida de los palets. Esta zona transporta los palets desde el almacén de palets hasta la salida de éstos. Por el camino, los palets sufren una pequeña parada para que el robot los complete con las 5 camadas con las que se forman. Igualmente, es en esta zona donde los palets esperan para poder salir cuando tenemos permiso del exterior.

Almacén de cartones

Se encarga de dosificar los cartones. Dispone de un armario donde se almacenan los cartones. De esta zona, el robot agarra mediante válvulas de vacío que posee en la pinza el robot los cartones para depositarlos posteriormente en el palet correspondiente.

Para localizar su ubicación en la planta ver la figura 2.4.

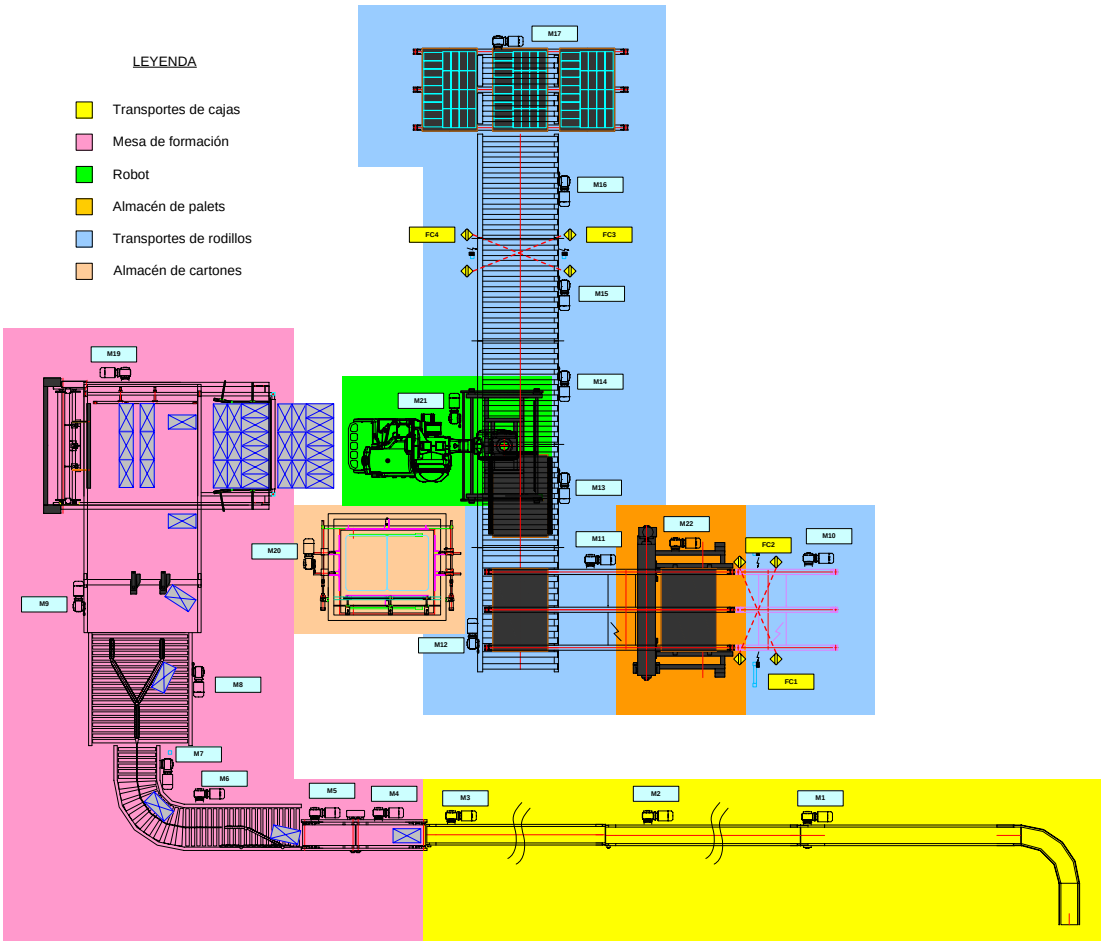


Figura 2.4. Zonas de la planta de paletizado

El control de la máquina se realiza a través de un panel de operador TP270, más adelante se explicará cada una de las pantallas que forman parte de la aplicación de control.

El sistema dispone de dos maneras de control:

- Control automático. Éste será el modo de funcionamiento por defecto. En este modo, el sistema debe funcionar de manera estable y completando palets a una velocidad de 12000 unidades/hora de una manera autónoma. En este modo, la única operación que debería realizar el encargado de mantenimiento es la de supervisión.

- Control manual: Se realizará con ayuda de los dos visualizadores de SIEMENS. Este modo está especialmente pensado para realizar tareas de manteniendo en caso de algún error o imprevisto que se produjese en el modo automático, como por ejemplo, atasco de alguna caja, retirada de objetos, reiniciar algún mecanismo, etc… Es decir, para el tratamiento de las alarmas surgidas. De cualquier manera, también en este modo se

podría usar para realizar tareas de control y supervisión, siempre en este caso, bajo la responsabilidad del encargado de mantenimiento de éste sistema.

Por último, para la comunicación entre los diversos elementos que forman el sistema, se realizará por medio de una red PROFIBUS DP.

2.2 Seguridades de la instalación

En la instalación existen distintos elementos de seguridad como son setas de emergencia, pestillos de puertas o barreras. A continuación describiremos estos elementos así como el procedimiento para rearmarlos.

- Setas de emergencia

  Todas las setas de emergencia de la instalación están seriadas a doble canal y conectados con un relé de seguridad.

  Cuando se abre esta serie se desactiva el relé de seguridad cortando el suministro eléctrico a toda la instalación. Para rearmar el módulo, una vez desenclavada la seta de emergencia correspondiente, tenemos un pulsador luminoso rojo que hay en la puerta del armario principal (más adelante se detallarán los armarios, cajas y pupitres disponibles) etiquetado como "Rearme defecto".

  Cuando una seta está enclavada, aparece su ubicación en la lista de alarmas de la aplicación de la TP270.

  El robot también se para cuando se produce una parada de emergencia.

- Pestillos de las puertas

  Todas las puertas de la instalación tienen un pestillo de seguridad. Estos pestillos se encuentran conectados a un relé de seguridad. Cuando se abre una puerta se desactiva el relé cortando el suministro eléctrico en su zona de influencia. Existen dos zonas de seguridad, el almacén de cartones y el resto de la instalación. Esto significa que podemos tener la puerta del almacén de cartones abierta y el resto de la instalación funcionando en automático y viceversa.

Cerca de cada puerta se encuentra una caja de pulsadores donde se realizan las operaciones de petición de apertura y rearme del módulo.

Los pestillos disponen de una bobina de enclavamiento de forma que la puerta no se puede abrir sin antes haber realizado una petición al sistema. La petición de apertura se realiza con un pulsador luminoso verde etiquetado como "Petición de apertura" si en el momento de la petición el sistema se encuentra en un estado indefinido por ejemplo el robot no está en ningún punto fijo sino que se encuentra realizando una trayectoria, el pulsador luminoso quedará parpadeando indicando que estamos a la espera de que se pueda abrir la puerta. Cuando el sistema se encuentre en un estado definido, desenclavará la bobina, y el pulsador luminoso quedará iluminado fijo, indicando que la puerta se encuentra abierta, aun cuando físicamente no se haya abierto. Permanecerá es ese estado hasta que volvamos a pulsarlo para decirle al sistema que ya puede enclavar la bobina. Después será necesario una pulsación del pulsador luminoso rojo etiquetada como "Rearme defecto" para rearmar el modulo de seguridad de la puerta correspondiente. Este pulsador luminoso rojo se ilumina cuando el módulo de seguridad no está activo, hecho que se produce cuando se abre una puerta. La puerta se considera abierta cuando su bobina de enclavamiento esta activada, pulsador luminoso verde iluminado fijo. Una petición de apertura pendiente, pulsador luminoso verde parpadeando puede ser cancelada con una nueva pulsación del mismo.

- Barreras

Las barreras también se encuentran conectadas a un modulo de seguridad de manera que cuando se cortan se desactiva el módulo cortando el suministro eléctrico a toda la instalación, excepto al almacén de cartones. Cuando esto ocurre sólo tendremos que rearmar el modulo correspondiente pulsando el pulsador luminoso rojo etiquetado como "Rearme defecto" de la caja de pulsadores de la barrera.

Si el fallo de barrera se produce por el propio producto por un fallo de mutting, disponemos de un pulsador luminoso azul etiquetado como "Override" que nos permitirá rearmar el módulo aún con la barrera cortada.

2.3 Arquitectura del sistema

En este apartado ahondaremos en la estructura física que forman las diferentes partes lógicas que comentamos en el apartado anterior.

En primer lugar, disponemos a la entrada del sistema de:

- 1 fotodetector IFM

Este fotodetector nos informará de cuando entran cajas al sistema. Servirá para activar el primer transportador de cajas.

Transporte de cajas.

Estará compuesto por:

- 3 cintas transportadoras de rodillos.
- 3 motores estándar (1 para cada cinta).
- 3 fotodetectores IFM (1 para cada cinta)

Los motores funcionaran siempre a la misma potencia. Los detectores servirán para advertir al sistema si han pasado cajas por la cinta transportadora. Nos servirá para, desde control, activar o desactivar (poner en marcha o parar el motor correspondiente) el siguiente transportador de banda, en función de un temporizador interno al control del sistema.

Deberá moverse cada una de ellas a una velocidad de 12 m/min.

Mesa de formación.

La zona donde se ordenaran las cajas en camadas para entregarla al robot, se compone a su vez de varias zonas. En primer lugar, dispone de:

- 1 fotodetector IFM
- 1 tope mecánico, accionado por 1 cilindro neumático.

Utilizamos este tope y el fotodetector asociado para gestionar la entrada de cajas de forma adecuada a la mesa de formación.

Además, también nos encontramos con:

- Una doble cinta de aceleración de malla LBP de baja fricción, para poder realizar acumulación sin presión.
- 2 motores estándar (1 para cada cinta).
- 2 variadores de frecuencia DANFOSS (1 para cada motor).

Con estos elementos conseguiremos acelerar las cajas que llegan del transportador de cajas, para conseguir separar las cajas suficientemente para poder posteriormente ordenarlas adecuadamente.

La velocidad de cada cinta aceleradora será de 25 m/min la primera y 35 m/min. la segunda, para conseguir de esta forma acelerar las cajas que llegan del transportador de cajas.

De la doble cinta aceleradora pasamos a una línea de rodillos compuesta por:

- 3 motores estándar.
- 3 variadores de frecuencia DANFOSS (1 para cada motor)
- 2 desviadores mecánicos, accionados por 1 cilindro neumático cada uno.
- 2 fotodetectores IFM

En esta línea de rodillos, se utilizará para formar cada caja en su fila correspondiente. Gracias a los desviadores mecánicos, podremos separar las cajas en 4 posibles filas distintas. Los dos fotodetectores nos ayudarán a contar el número de cajas que van pasando por cada desviador y así tener un control adecuado sobre el sistema.

Las 4 opciones de las que se disponen para ordenar las cajas se pueden observar en la siguiente figura:

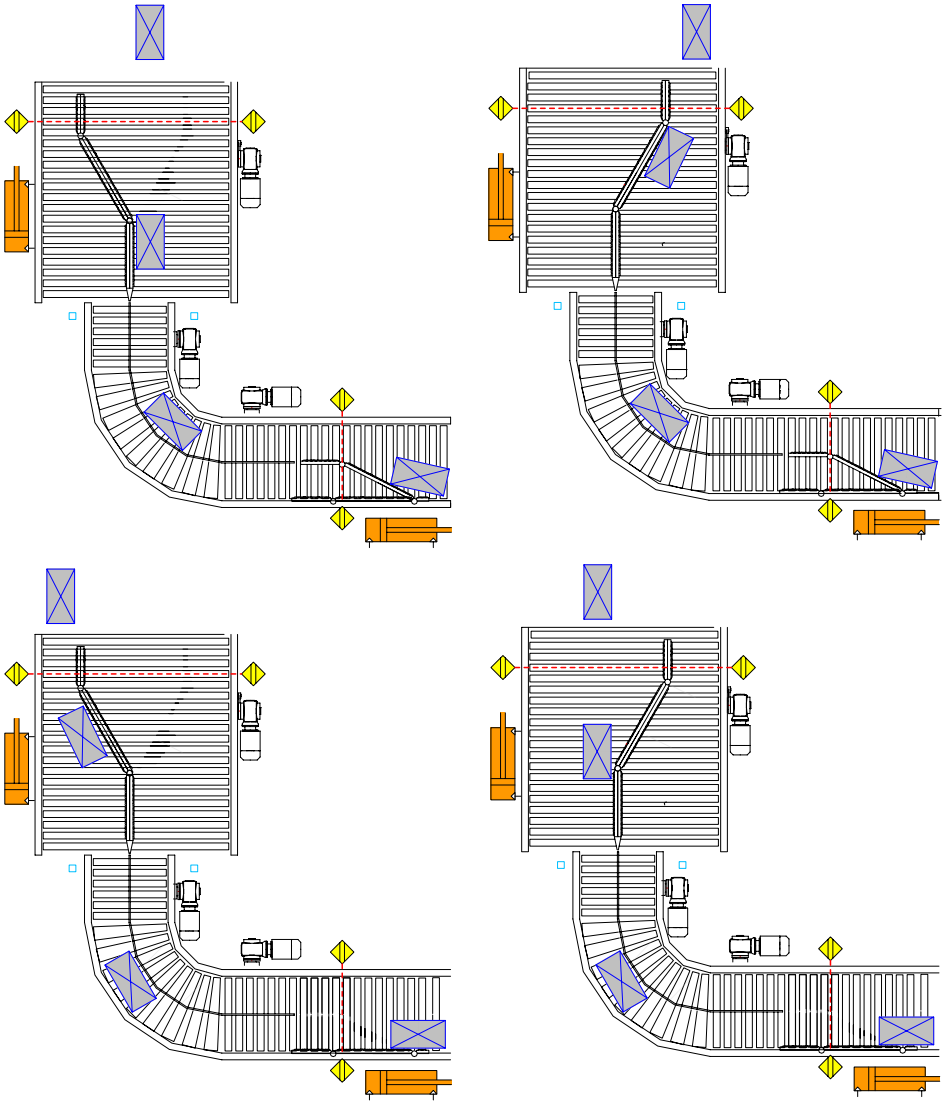


Figura 2.5. Formación de filas en la mesa de formación.

La línea de rodillos alimenta a otra zona definida por:

- 1 cinta transportadora de malla LBP de baja fricción.
- 1 motor estándar
- 1 variador de frecuencia DANFOSS
- 2 giradores mecánicos, accionados por 1 cilindro neumático cada uno.
- 1 tope mecánico, accionado por 1 cilindro neumático.

Es aquí donde se produce el giro de la caja cuando es necesario, en función del formato de caja elegido. Hay que indicar que en la línea de rodillos, las separaciones de cada fila se deberán ir alternando para permitir crear espacio entre cajas para un giro

limpio. Al final de ésta línea, disponemos del tope, para impedir la entrada en el empujador de la mesa de formación.

La velocidad de gestión de cajas deberá ser de 55 cajas/minuto.

Al final de la mesa nos encontramos con el empujador, que se encarga de desplazar las filas, hasta la zona de transferencia, donde compactamos la camada, y de ahí al cabezal del robot. El mencionado empujador constará de:

- 1 motor estándar
- 1 variador de frecuencia DANFOSS
- 5 detectores inductivos IFM
- 4 fotocélula IFM
- 1 empujador mecánico, accionado por 1 cilindro neumático.
- 1 tope mecánico, accionado por 1 cilindro neumático.

El empujador mecánico se podrá subir o bajar para desplazar la carga (si está abajo) o no (empujador arriba). Con el variador de frecuencia controlaremos el movimiento del motor que realiza propiamente el desplazamiento, para que se haga de una forma suave, mientras que los detectores nos advertirán de la posición donde se encuentra el mencionado empujador mecánico. Al final del mismo, disponemos de un tope mecánico que nos servirá para evitar la caída de las cajas al suelo. La estructura de esta zona, se podrá dividir en 2 zonas lógicas:

- zona de formación: donde se acumulan las filas que se han ido ordenando previamente.
- Zona de precarga: Es la zona donde la camada espera a cargarse en la pinza del robot.

La siguiente zona sería ya la pinza del robot propiamente dicha. Los 5 detectores inductivos nos advertirán de la zona donde se encuentra el motor en sí. Mientras que las fotocélulas las usaremos para advertir si hay cajas formadas en la zona de formación del empujador y si hay carga esperando en la zona de precarga.

Robot

Es el encargado de recoger las camadas completas del empujador y los cartones del almacén de cartones y depositarlo en los palets. El robot que se va a utilizar es un

S4CPLUS, de la serie 7600, de seis ejes y sin ejes externos conectados de la marca ABB ROBOTICS.

En su extremidad dispone de una garra o pinza especialmente diseñada para cargar camadas. Esta pinza está formada por:

- 1 motor estándar
- 1 variador de frecuencia DANFOSS
- 1 fotocélula IFM
- 4 detectores inductivos IFM
- 3 cilindros neumáticos
- 2 vacuostatos

El variador controlará el movimiento del motor, que regula las operaciones de cierre y apertura de una persiana disponible en la pinza, para dejar caer la carga sobre el palet. Igualmente, para agarrar los cartones, dispone de unas pinzas accionadas por un cilindro neumático, con 2 vacuostatos en sus extremos para poder coger los cartones del almacén de cartones. La camada cargada, se centrará con ayuda de 2 centradores accionados por sendos cilindros neumáticos. Con los 4 detectores inductivos detectaremos la posición de la persiana de la pinza (abierta o cerrada). La fotocélula nos indicará si hay carga en la pinza o no.

Almacén de Palets

Está compuesto por:

- 1 motor inversor
- 4 detectores inductivos
- 1 fotocélula IFM
- 1 cilindro neumático.

Usaremos el motor inversor para subir o bajar los palets vacíos. Los detectores inductivos nos ayudarán a determinar la posición en cada momento del almacén de

palets. El cilindro neumático lo usaremos para accionar unas pinzas mecánicas para que agarren los palets que llegan de la línea exterior.

Transporte de rodillos

El transporte de rodillos esta compuesto por varias zonas con una función lógica muy parecida. Se compone en total de:

- 9 motores estándar
- 6 variadores de frecuencia DANFOSS
- 11 fotocélulas IFM
- 4 cilindros neumáticos

La primera parte es la que alimenta al almacén de palets. Con 1 fotocélula que nos informa de la llegada de palets y otras 2 fotocélulas más de mutting, para seguridad por si los palets entran en una posición errónea en éste y para que no entre ningún objeto extraño en esta zona. Esta primera zona estaría compuesta por 2 transportadores de cadenas que funcionan con un motor estándar cada uno.

Después nos encontramos con 2 mesas de transferencia de palets angular, cada una de ellas compuesta por:

- 1 motor estándar
- 1 variador de frecuencia DANFOSS
- 1 fotocélula IFM

La ubicación de ambas se aprecia en la Figura 2.6 de la configuración completa formada por el transporte de rodillos y el almacén de palets.

Entre las mesas de transferencia de palets, nos encontramos con 5 transportadores de rodillos, cada uno de ellos con:

- 1 motor estándar
- 1 variador de frecuencia DANFOSS
- 1 fotocélula IFM

Y en uno de estos, nos encontramos con el dosificador de palets, compuesto a su vez por:

- 1 tope mecánico, accionado por 1 cilindro neumático
- 1 centrador de palets, accionado por 1 cilindro neumático

Que estará en la posición donde el robot irá alojando los cartones y camadas sobre el palet correspondiente.

Por último, mencionar que a la salida de éste nos encontramos con otras 2 fotocélulas de Mutting para seguridad de que el palet está correctamente colocado y no pasa nada extraño por medio.

Almacén de cartones

Como ya se explicó, es la zona donde se almacenan los cartones. Dispone de los siguientes elementos:

- 1 motor estándar
- 1 variador de frecuencia DANFOSS
- 5 fotocélulas IFM
- 4 detectores inductivos
- 2 cilindros neumáticos

Con las fotocélulas, averiguaremos el estado del almacén de cartones (si esta vacío o si llega a un nivel mínimo suficiente) y si están colocados en el sitio

correspondiente para que la pinza del robot las agarre. Los cilindros neumáticos accionaran un centrador mecánico de cartones, y una garra que llevará los cartones del almacén a la posición reservada para que el robot pueda disponer de ellos.

Una vez mencionado esto, ilustramos la configuración global de nuestro sistema:

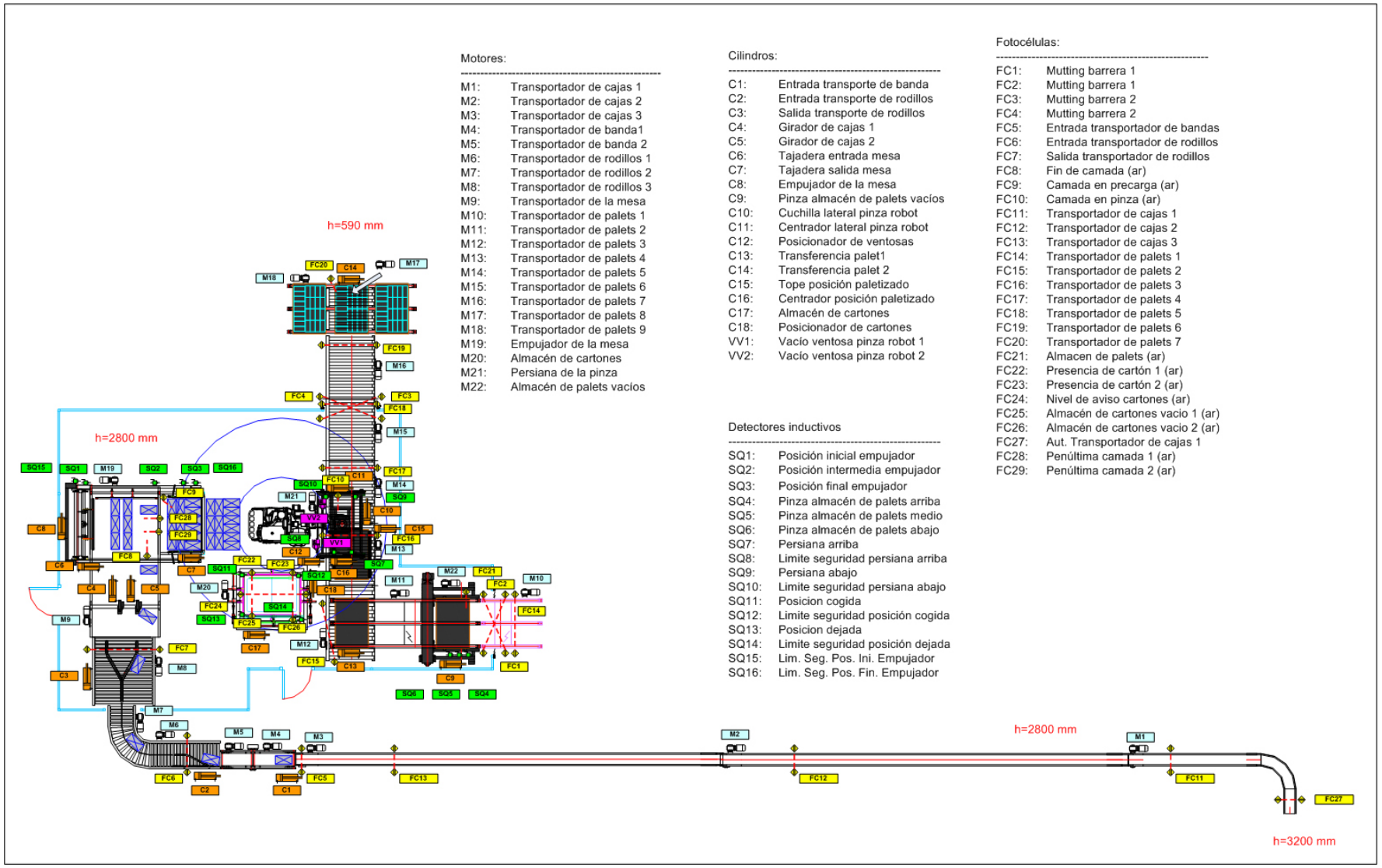


Figura 2.6 Arquitectura completa de la instalación

Localización del sistema

Una vez visto el conjunto, se aprecia en la imagen la altura en la que se colocará el sistema, que se puede dividir en dos zonas:

- Zona 1:

  Compuesta a su vez por:

  - Transporte de cajas
  - Mesa de formación
  - Robot

- Zona 2:

  Compuesta por:

  - Almacén de palets
  - Transporte de rodillos
  - Almacén de cartones

Así pues, la zona 1 recibe cajas de la línea exterior, situada a 3200 mm. y descenderá en rampa con el primer transportador de bandas hasta situar el resto de esta zona a una altura de 2800 mm. mientras que la zona 2 se encuentra a 590 metros. Se aprecia pues que el robot hará de nexo de unión entre la zona 1 y la zona 2.

Resumen de elementos

Por tanto en resumen, los elementos físicos que componen el sistema se resumen en la siguiente tabla:

Unidades	**Elementos**
18	Cilindros
22	Motores
15	Variadores de frecuencia
27	Fotocélulas
16	Detectores inductivos
2	Vacuostatos

Tabla 2.1. Resumen de elementos

2.4 Armarios, pupitres y cajetines del sistema

Una vez mencionado los elementos de campo que controlarán el sistema, exponemos a continuación algo sobre la parte eléctrica en que se compone nuestro sistema, que nos será básico saberlo para poder programar posteriormente correctamente nuestro autómata, uno de los objeto de estudio principales en que se basa éste proyecto. Es por ello por lo que no ahondaremos en muchos detalles en éste apartado.

La parte eléctrica de nuestro sistema está distribuida principalmente por:

- 1 armario principal, donde se encontrará entre otras cosas el PLC.
- 1 pupitre PG de control manual
- 7 pupitres con indicadores y setas
- 1 Armario AR del robot.

Armario Principal

En el interior del armario principal dispondremos de:

- Térmicos y contactores de los motores
- Variadores de frecuencia DANFOSS para los motores.
- Transformadores, interruptores y disyuntores, para la alimentación del sistema.
- PLC de SIEMENS y toma de corriente (para el PLC).
- Relés, contactores y magnetotérmicos para la seguridad del sistema
- Borneros
- Lámpara para la iluminación del armario

A modo de ejemplo, mostramos un esquema del mismo:

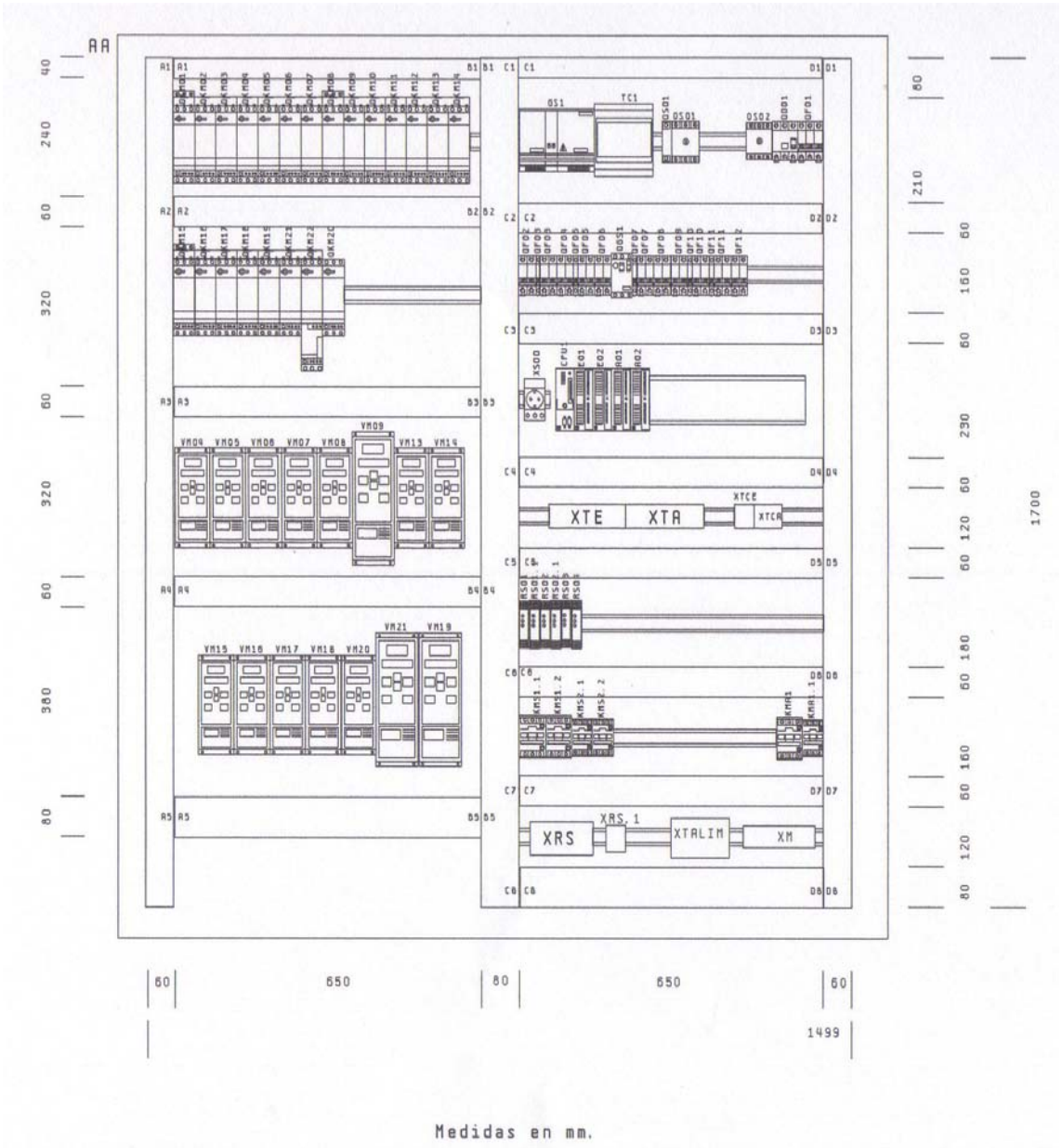


Figura 2.7. Interior del Armario

Por su parte, en el exterior del mismo, disponemos de:

- 1 Panel Táctil TP270, de SIEMENS
- 2 Ventiladores, de RITTAL
- 1 maneta para interruptor seccionador, de TELEMECANIQUE
- 1 Setas de emergencia, de TELEMECANIQUE
- 1 Piloto Luminoso LED blanco, de TELEMECANIQUE
- 2 pulsadores luminosos (1 blanco, 1 rojo), de TELEMECANIQUE
- 1 Pulsador rasante rojo, de TELEMECANIQUE

El TP270 se analizará con profundidad más adelante, ya que éste se usará como interface Hombre-Máquina para el control del sistema.

La maneta activa o desactiva el interruptor QS01, y permitirá o desactivará la corriente de entrada a nuestro sistema.

Los ventiladores, necesarios para el mantenimiento óptimo de los elementos que se encuentran en el interior del armario, para evitar su recalentamiento.

Los pulsadores, serán útiles para que el operario pueda encender el sistema con seguridad y así suministrar corriente a la instalación. Las señales de estos pulsadores serán tratadas de manera eléctrica, por lo que no las estudiaremos en detalle al salirse fuera de los objetivos de éste proyecto (nuestro PLC no tiene control alguno sobre estos pulsadores).

Mostramos esquema de la parte exterior del armario:

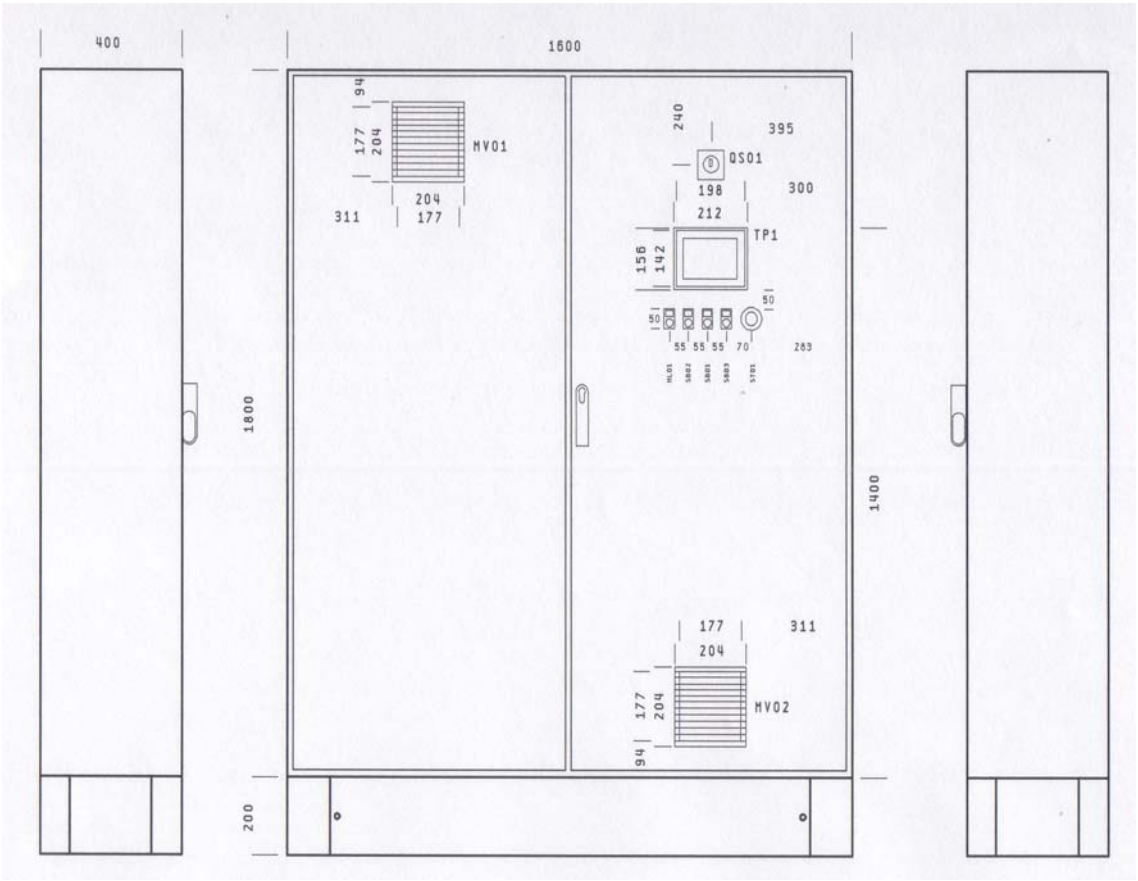


Figura 2.8. Exterior del Armario

Pupitre PG

En su interior dispone de:

- 1 Bornero, de PHOENIX CONTACT

Y en el exterior:

- 1 Seta de emergencia roja, de TELEMECANIQUE
- 1 pulsador luminoso (rojo), de TELEMECANIQUE
- 1 selector de 3 posiciones, de TELEMECANIQUE
- Panel de Texto TD17, de SIEMENS

El panel TD17 se usará únicamente para realizar movimientos manuales y gestionar las alarmas. El pulsador y la seta se utilizarán para gestionar la seguridad de la instalación. El selector será de utilidad para el funcionamiento de la TD17, que no es objeto de estudio en este proyecto.

Cajetines CC1, CC3 y CC5

En el exterior de cada uno disponemos de:

- 1 Seta de emergencia roja, de TELEMECANIQUE
- 2 Pulsadores luminosos (verde y rojo), de TELEMECANIQUE

En el interior de estos cajetines tan sólo hay los cables que le llegan al mismo.

Estos cajetines se usarán como seguridad para poder controlar los pestillos de seguridad que gobiernan las puertas de entrada al sistema. El uso de los pulsadores se explica en el apartado 2.2 de éste mismo capítulo.

Cajetín CC6 y CC7

En el exterior de cada uno disponemos de:

- 1 seta de emergencia roja, de TELEMECANIQUE

En el interior de estos cajetines tan sólo hay los cables que le llegan al mismo.

Estos cajetines se usarán para seguridad del sistema. La seta hará un paro de todo el sistema.

Pupitres CC2 y CC4

En el interior de cada uno de ellos se dispone de:

- 1 Relé de Seguridad de Control de Mutting, de SICK
- 1 Relé de Seguridad, de SICK
- 1 Bornero, de PHOENIX CONTACT

Y en el exterior (en cada uno de ellos):

- 2 pulsadores luminosos (azul y rojo), de TELEMECANIQUE

Se usarán para controlar las dos barreras de muting que tenemos en el sistema (Gobernadas respectivamente por las fotocélulas FC1, FC2 y FC3, FC4). El uso de los pulsadores se explica en el apartado 2.2 de éste mismo capítulo.

Pupitre AR

En él encontramos los elementos necesarios para el control del Robot. Tan sólo nos interesa saber para nuestro estudio, de que dispone en su interior entre otras cosas de:

- 1 Bornero, de PHOENIX CONTACT

A este bornero le llegarán los contactos de seguridad del mismo, para paralizar en caso de que fuese necesario debido a problemas del sistema el funcionamiento del robot (por ejemplo un paro general del sistema).

Mostramos a continuación un gráfico de la localización de cada uno de estos zócalos (Los elementos nombrados como Wxxx son las mangueras que llevan los cables hacia el armario principal):

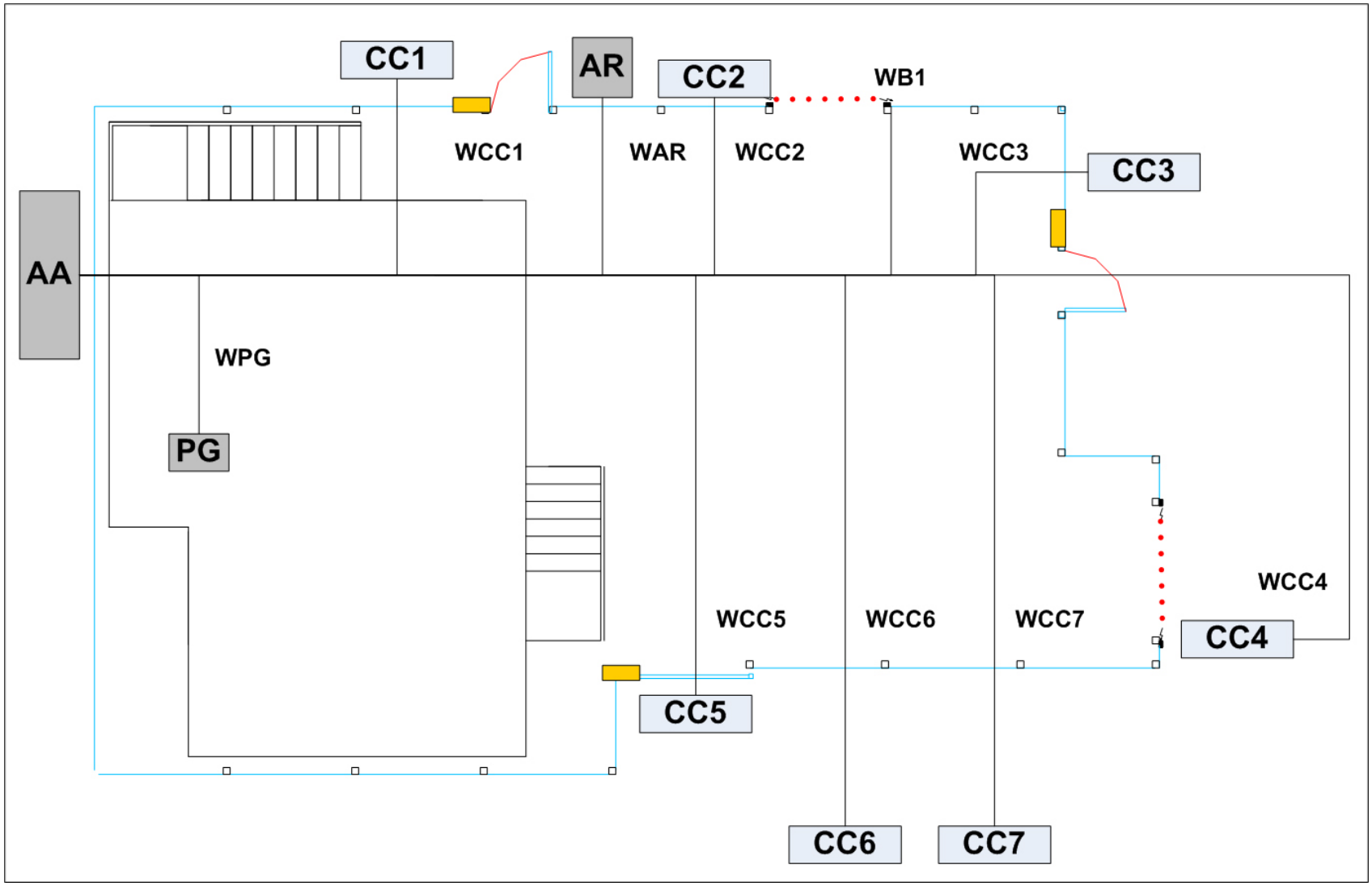


Figura 2.9. Localización del Armario, Pupitres y Cajetines.

2.5 Entradas y salidas digitales

El número de salidas del sistema no es muy numeroso. Es por eso, por lo que ha bastado la utilización de un PLC para gobernar el sistema y no ha sido necesario el utilizar cualquier otro tipo de control, como hubiese podido ser algún SCADA que regulase el total de las entradas y salidas.

Para controlar las entras y salidas (todas serán digitales), se usará lo siguiente:

- 2 módulos de 32 entradas digitales de 24 V. DC de SIEMENS
- 2 módulos de 32 salidas digitales de 24 V. DC y 0.5 A. de SIEMENS
- 4 módulos SIMATIC ET200S de SIEMENS.
- Tarjeta d352 de comunicación ProfibusDP esclavo, con 32 entradas y 32 salidas digitales, de ABB.

Las ET200 se usarán para controlar los elementos de campo que se encuentran dispersos, es decir, los cilindros, vacuostatos, detectores inductivos y fotocélulas. Su uso está justificado pues estos elementos están distribuidos en zonas alejadas del armario donde se encuentra el PLC. Es por ello por lo que nos vemos obligados a utilizar una periferia descentralizada.

Para el resto de elementos, se usarán los módulos de 32 entradas y 32 salidas digitales, que van acoplados en el mismo bastidor del PLC. Se podrá controlar los elementos que se encuentren en el propio armario (relés y contactores de motores, alimentación, etc..), así como el tema de seguridad del sistema, esto es, setas de seguridad, pulsadores y lámparas asociadas a cada uno, pestillos, etc…

La tarjeta d352 que dispone de 128 entradas y 128 salidas digitales se usarán para las señales necesarias para la comunicación con el robot ABB.

El número total de entradas y salidas la reflejamos en la siguiente tabla:

Módulo	Número de señales equipadas (en lista)
Módulos de 32 entradas digitales (PLC)	64 (51)
Módulo de 32 salidas digitales (PLC)	64 (29)
Entradas ET200S	100 (89)
Salidas ET200S	56 (42)
Entradas Robot ABB	128 (17)
Salidas Robot ABB	128 (22)
Total	540 (250)

Tabla 2.2. Número de entradas y salidas

Se comprueba que el total de señales no es muy grande, por lo que la utilización del PLC junto con los elementos antes mencionado, nos basta para poder controlar todo el sistema.

2.5.1 Entradas y salidas digitales: PLC

Como ya se comentó anteriormente, el PLC lleva en su mismo bastidor 2 módulos de 32 entradas digitales y otros 2 de salidas igualmente digitales. Mostramos en la siguiente tabla concretamente los elementos eléctricos que lleva conectados y una breve descripción de cada una de ellos.

Entradas:

Módulo	Dirección	Elemento	Descripción
E01	E8.0	QF03	Magnetotérmico Alimentación 220V
E01	E8.1	QF07	Magnetotérmico Alimentación Lógica cableada
E01	E8.2	QF10	Magnetotérmico Alimentación Entradas
E01	E8.3	QF11	Magnetotérmico Alimentación Salidas
E01	E8.4	QKM01	Térmico Motor 1 Transportador de Cajas 1
E01	E8.5	QKM02	Térmico Motor 2 Transportador de Cajas 2
E01	E8.6	QKM03	Térmico Motor 3 Transportador de Cajas 3

E01	E8.7	QKM04	Térmico Motor 4 Transportador de banda 1
E01	E9.0	QKM05	Térmico Motor 5 Transportador de banda 2
E01	E9.1	QKM06	Térmico Motor 6 Transportador de rodillos 1
E01	E9.2	QKM07	Térmico Motor 7 Transportador de rodillos 2
E01	E9.3	QKM08	Térmico Motor 8 Transportador de rodillos 3
E01	E9.4	QKM09	Térmico Motor 9 Transportador de la mesa
E01	E9.5	QKM10	Térmico Motor 10 Transportador de palets 1
E01	E9.6	QKM11	Térmico Motor 11 Transportador de palets 2
E01	E9.7	QKM12	Térmico Motor 12 Transportador de palets 3
E01	E10.0	QKM13	Térmico Motor 13 Transportador de palets 4
E01	E10.1	QKM14	Térmico Motor 14 Transportador de palets 5
E01	E10.2	QKM15	Térmico Motor 15 Transportador de palets 6
E01	E10.3	QKM16	Térmico Motor 16 Transportador de palets 7
E01	E10.4	QKM17	Térmico Motor 17 Transportador de palets 8
E01	E10.5	QKM18	Térmico Motor 18 Transportador de palets 9
E01	E10.6	QKM19	Térmico Motor 19 Empujador de la mesa
E01	E10.7	QKM20	Térmico Motor 20 Almacén de cartones
E01	E11.0	QKM21	Térmico Motor 21 Persiana de la pinza
E01	E11.1	QKM22	Térmico Motor 22 Almacén de palets vacíos
E01	E11.2	KMA1	Contacto de Puesta en Servicio
E01	E11.3	QF05	Magnetotérmico Ventilación Armario
E01	E11.4	SB03	Rearme Relé de Seguridad de Setas en AA RS01
E01	E11.5	SB14	Rearme Relé de Seguridad de Setas en PG RS01
E01	E11.6	SB11	Pulsador Luminoso Verde CC1
E01	E11.7	SB12	Pulsador Luminoso Verde CC2
E02	E12.0	SB13	Pulsador Luminoso Verde CC3
E02	E12.1	ST01	Seta de seguridad
E02	E12.2	ST02	Seta de seguridad PG
E02	E12.3	ST03	Seta de seguridad CC1
E02	E12.4	ST04	Seta de seguridad CC3
E02	E12.5	ST05	Seta de seguridad CC5
E02	E12.6	ST06	Seta de seguridad CC6

E02	E12.7	ST07	Seta de seguridad CC7
E02	E13.0	RS01	Relé de Seguridad Setas
E02	E13.1	RS02	Relé de Seguridad Puertas
E02	E13.2	RS03	Relé de Seguridad Almacén de Cartones
E02	E13.3	RS09	Relé de Seguridad Muting 1 y 2
E02	E13.4	LIBRE	-
E02	E13.5	SQ03	Interruptor pestillo puerta 1
E02	E13.6	SQ04	Interruptor pestillo puerta 2
E02	E13.7	SQ05	Interruptor pestillo puerta 3
E02	E14.0	LIBRE	-
E02	E14.1	QF12	Magnetotérmico Alimentación Freno Motor Pinza Robot
E02	E14.2	ST08	Decremento movimiento
E02	E14.3	ST09	Incremento movimiento
E02	E14.4	SB15	Pulsador carga almacén de palets
E02	De E14.5 hasta E15.7	LIBRE	-

Tabla 2.3. Entradas PLC

Salidas:

Módulo	Dirección	Elemento	Descripción
A01	A8.0	KM01	Contactor Motor 1 Transportador de cajas 1
A01	A8.1	KM02	Contactor Motor 2 Transportador de cajas 2
A01	A8.2	KM03	Contactor Motor 3 Transportador de cajas 3
A01	A8.3	KM04	Contactor Motor 4 Transportador de banda 1
A01	A8.4	KM05	Contactor Motor 5 Transportador de banda 2
A01	A8.5	KM06	Contactor Motor 6 Transportador de rodillos 1
A01	A8.6	KM07	Contactor Motor 7 Transportador de rodillos 2
A01	A8.7	KM08	Contactor Motor 8 Transportador de rodillos 3
A01	A9.0	KM09	Contactor Motor 9 Transportador de la mesa
A01	A9.1	KM10	Contactor Motor 10 Transportador de palets 1
A01	A9.2	KM11	Contactor Motor 11 Transportador de palets 2

A01	A9.3	KM12	Contactor Motor 12 Transportador de palets 3
A01	A9.4	KM13	Contactor Motor 13 Transportador de palets 4
A01	A9.5	KM14	Contactor Motor 14 Transportador de palets 5
A01	A9.6	KM15	Contactor Motor 15 Transportador de palets 6
A01	A9.7	KM16	Contactor Motor 16 Transportador de palets 7
A01	A10.0	KM17	Contactor Motor 17 Transportador de palets 8
A01	A10.1	KM18	Contactor Motor 18 Transportador de palets 9
A01	A10.2	KM19	Contactor Motor 19 Empujador de la mesa
A01	A10.3	KM20	Contactor Motor 20 Almacén de cartones
A01	A10.4	KM21	Contactor Motor 21 Persiana de la pinza
A01	A10.5	KM22A	Contactor Motor 22a Almacén de palets vacíos Subir
A01	A10.6	KM22B	Contactor Motor 22b Almacén de palets vacíos Bajar
A01	A10.7	LIBRE	-
A01	A11.0	HL11	Lámpara Petición apertura puerta 1
A01	A11.1	HL12	Lámpara Petición apertura puerta 2
A01	A11.2	HL13	Lámpara Petición apertura puerta 3
A01	A11.3	KAP1	Bobina enclavamiento puerta 1
A01	A11.4	KAP2	Bobina enclavamiento puerta 2
A01	A11.5	KAP3	Bobina enclavamiento puerta 3
A01	A11.6	LIBRE	-
A01	A11.7	LIBRE	-
A02	A12.0	HLB0	Sirena Baliza
A02	A12.1	HLB1	Lámpara Baliza Verde
A02	A12.2	HLB2	Lámpara Baliza Naranja
A02	A12.3	HLB3	Lámpara Baliza Roja
A02	De A12.4 hasta A15.7	LIBRE	-

Tabla 2.4. Salidas PLC

2.5.2 Entradas y salidas digitales: ET200S

Dado la envergadura del proyecto y las dimensiones de éste, ya se ha comentado que es necesario utilizar una periferia descentralizada para poder llegar a todos los elementos de campo y no tener que utilizar un cableado demasiado largo, y por tanto costoso, y poco fiable y flexible.

Para ello, se ha utilizado un total de 4 equipos ET200S, de SIEMENS, para satisfacer nuestras necesidades. A éstos equipos se le conectan los diversos elementos de campo, y a su vez esto, se comunican posteriormente con el autómata a través del bus de campo normalizado PROFIBUS DP.

Se les ha denominado, para distinguir uno de otros como ET200[1], ET200[2], ET200[3] y ET200[4]. Y se han distribuido para que cada uno de estos acceda a una zona determinada del sistema.

Es por tanto, que la configuración de cada uno de ellos es distinta, según las necesidades, pues cada uno abarca zonas diversas. Indicamos a continuación la configuración y las señales que se le conectan a cada uno de estos 4 elementos.

ET200[1]:

Está provisto de:

- 9 Módulos de entradas digitales 4DI 24V DC, de SIEMENS.
- 5 Módulos de salidas digitales 4DO 24V DC, de SIEMENS.

A cada uno de estos módulos, se le han conectado los siguientes elementos eléctricos.

Entradas:

ET200[1]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
100				
101	E101.0	C1	SQ1010	Tope Entrada Transporte de Bandas

				Retrocedido
	E101.1	C1	SQ1011	Tope Entrada Transporte de Bandas Avanzado
	E101.2	C2	SQ1012	Desviador 1 Retrocedido
	E101.3	C2	SQ1013	Desviador 1 Avanzado
102	E102.0	C3	SQ1020	Desviador 2 Retrocedido
	E102.1	C3	SQ1021	Desviador 2 Avanzado
	E102.2	C4	SQ1022	Girador de Cajas 1 Retrocedido
	E102.3	C4	SQ1023	Girador de Cajas 1 Avanzado
103	E103.0	C5	SQ1030	Girador de Cajas 2 Retrocedido
	E103.1	C5	SQ1031	Girador de Cajas 2 Avanzado
	E103.2	C6	SQ1032	Tope Entrada Mesa Retrocedido
	E103.4	C6	SQ1033	Tope Entrada Mesa Avanzado
104	E104.0	C7	SQ1040	Tope Salida Mesa Retrocedido
	E104.1	C7	SQ1041	Tope Salida Mesa Avanzado
	E104.2	C8	SQ1042	Empujador Retrocedido
	E104.3	C8	SQ1043	Empujador Avanzado
105	E105.0	FR1	PR1050	Presostato Zona Neumática 1
	E105.1	FC5	SQ1051	Presencia Entrada Transportador de Bandas
	E105.2	FC6	SQ1052	Presencia Salida Desviador 1
	E105.3	FC7	SQ1053	Presencia Salida Desviador 2
106	E106.0	FC8	SQ1060	Fin de Camada
	E106.1	SQ18	SQ1061	Empujador en Precarga 2
	E106.2	FC11	SQ1062	Presencia Transportador de Cajas 1
	E106.3	FC12	SQ1063	Presencia Transportador de Cajas 2
107	E107.0	FC13	SQ1070	Presencia Transportador de Cajas 3
	E107.1	SQ1	SQ1071	Empujador Retrocedido
	E107.2	SQ2	SQ1072	Empujador en Precarga
	E107.3	SQ3	SQ1073	Empujador Avanzado
108	E108.0	FC27	SQ1080	Autorización Transportador de Cajas 1
	E108.1	SQ15	SQ1081	Limite Seguridad Posición Inicial Empujador

	E108.2	SQ16	SQ1082	Limite Seguridad Posición Final Empujador
	E108.3	FC9	SQ1083	Camada en Precarga
109	E109.0	FC28	SQ1090	Penúltima camada 1
	E109.1	FC29	SQ1091	Penúltima camada 2
	E109.2	-	LIBRE	-
	E109.3	-	LIBRE	-

Tabla 2.5. Entradas ET200[1]

Salidas:

ET200[1]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
150				
151	A151.0	C1	EV1510	Tope Entrada Transporte de Bandas Retroceder
	A151.1	C1	EV1511	Tope Entrada Transporte de Bandas Avanzar
	A151.2	C2	EV1512	Desviador 1 Retroceder
	A151.3	C2	EV1513	Desviador 1 Avanzar
152	A152.0	C3	EV1520	Desviador 2 Retroceder
	A152.1	C3	EV1521	Desviador 2 Avanzar
	A152.2	C4	EV1522	Girador de Cajas 1 Retroceder
	A152.3	C4	EV1523	Girador de Cajas 1 Avanzar
153	A153.0	C5	EV1530	Girador de Cajas 2 Retroceder
	A153.1	C5	EV1531	Girador de Cajas 2 Avanzar
	A153.2	C6	EV1532	Tope Entrada Mesa Retroceder
	A153.3	C6	EV1533	Tope Entrada Mesa Avanzar
154	A154.0	C7	EV1540	Tope Salida Mesa Retroceder
	A154.1	C7	EV1541	Tope Salida Mesa Avanzar
	A154.2	C8	EV1542	Tope Empujador Retroceder
	A154.3	C8	EV1543	Tope Empujador Avanzar
155	A155.0	FR1	EV1550	Válvula de Corte Zona Neumática 1
	A155.1	-	LIBRE	-

	A155.2	-	LIBRE	-
	A155.3	-	LIBRE	-

Tabla 2.6. Salidas ET200[1]

Mostramos a continuación sinóptico del sistema de la zona de actuación del ET200[1]:

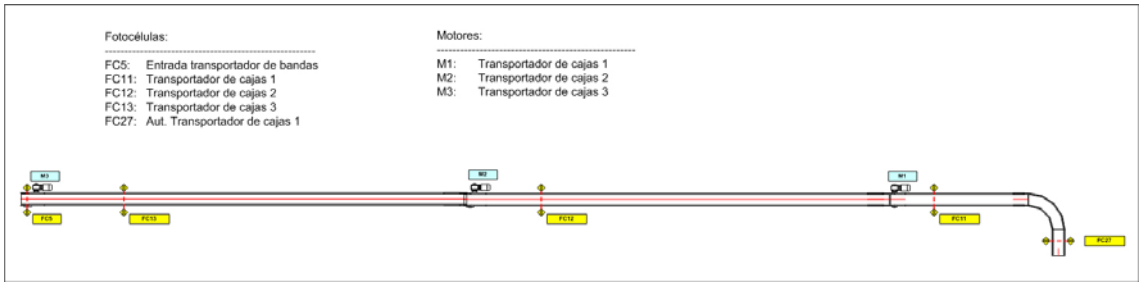


Figura 2.10 Transporte de cajas

.

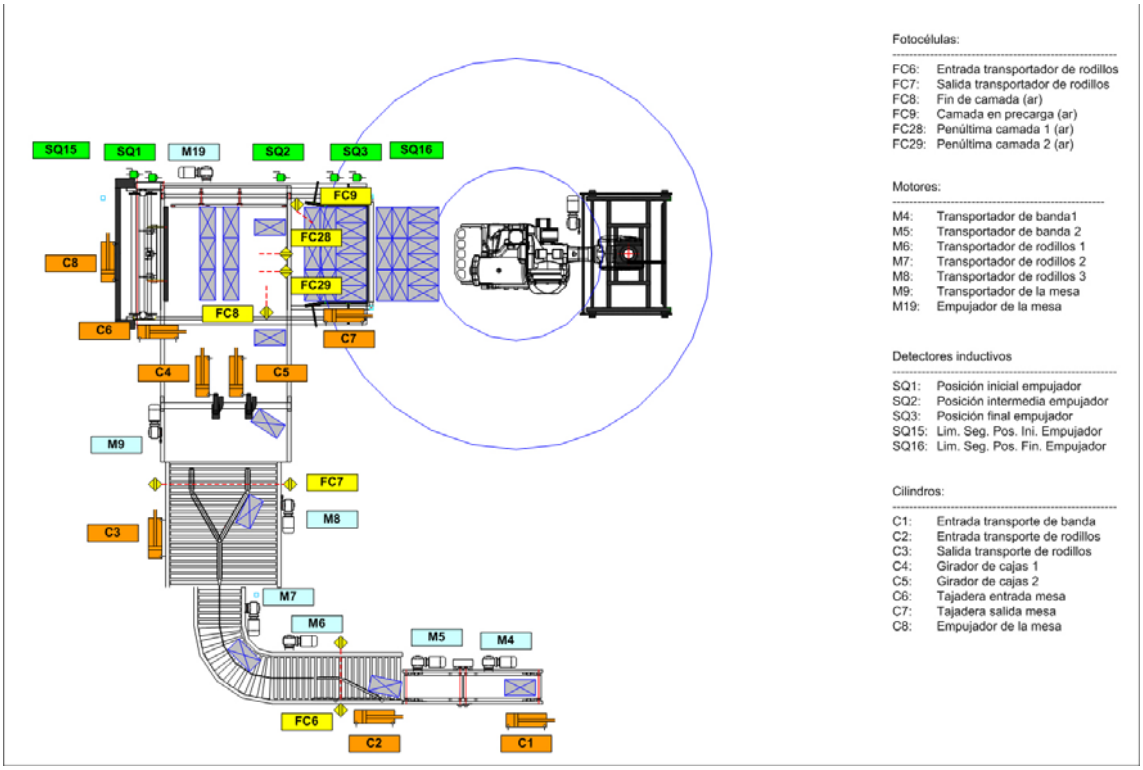


Figura 2.11. Mesa de Formación

ET200[2]

Está provisto a su vez de:

- 5 Módulos de entradas digitales 4DI 24V DC, de SIEMENS.
- 3 Módulos de salidas digitales 4DO 24V DC, de SIEMENS.

Los elementos eléctricos que tiene conectado cada módulo son los siguientes:

Entradas:

ET200[2]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
200				
201	E201.0	C10	SQ2010	Cuchilla Lateral Pinza Robot Retrocedido
	E201.1	C10	SQ2011	Cuchilla Lateral Pinza Robot Avanzado
	E201.2	C11	SQ2012	Centrador Lateral Izq Pinza Robot Retrocedido
	E201.3	C11	SQ2013	Centrador Lateral Izq Pinza Robot Avanzado
202	E202.0	C12	SQ2020	Posicionador Izq de Ventosas Retrocedido
	E202.1	C12	SQ2021	Posicionador Izq de Ventosas Avanzado
	E202.2	VV1	SQ2022	Vacuostato 1
	E202.3	VV2	SQ2023	Vacuostato 2
203	E203.0	SQ7	SQ2030	Persiana Arriba
	E203.1	SQ8	SQ2031	Limite Seguridad Persiana Arriba
	E203.2	SQ9	SQ2032	Persiana Abajo
	E203.3	SQ10	SQ2033	Limite Seguridad Persiana Abajo
204	E204.0	FC10	SQ2040	Presencia de Carga en Pinza
	E204.1	C11	SQ2041	Centrador Lateral Der Pinza Robot Retrocedido
	E204.2	C11	SQ2042	Centrador Lateral Der Pinza Robot Avanzado
	E204.3	-	LIBRE	-
205	E205.0	C12	SQ2050	Posicionador Der de Ventosas Retrocedido
	E205.1	C12	SQ2051	Posicionador Der de Ventosas Avanzado

	E205.2	-	LIBRE	-
	E205.3	-	LIBRE	-

Tabla 2.7. Entradas ET200[2]

Salidas:

ET200[2]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
250				
251	A251.0	C10	EV2510	Cuchilla Lateral Pinza Robot Retroceder
	A251.1	C10	EV2511	Cuchilla Lateral Pinza Robot Avanzar
	A251.2	C11	EV2512	Centrador Lateral Pinza Robot Retroceder
	A251.3	C11	EV2513	Centrador Lateral Pinza Robot Avanzar
252	A252.0	C12	EV2520	Posicionador de Ventosas Retroceder
	A252.1	C12	EV2521	Posicionador de Ventosas Avanzar
	A252.2	VV1	EV2522	Válvula de Vacío 1
	A252.3	VV2	EV2523	Válvula de Vacío 2
253	A253.0	-	LIBRE	-
	A253.1	-	LIBRE	-
	A253.2	-	LIBRE	-
	A253.3	-	LIBRE	-

Tabla 2.8. Salidas ET200[2]

La zona de actuación en este caso corresponde a la siguiente zona:

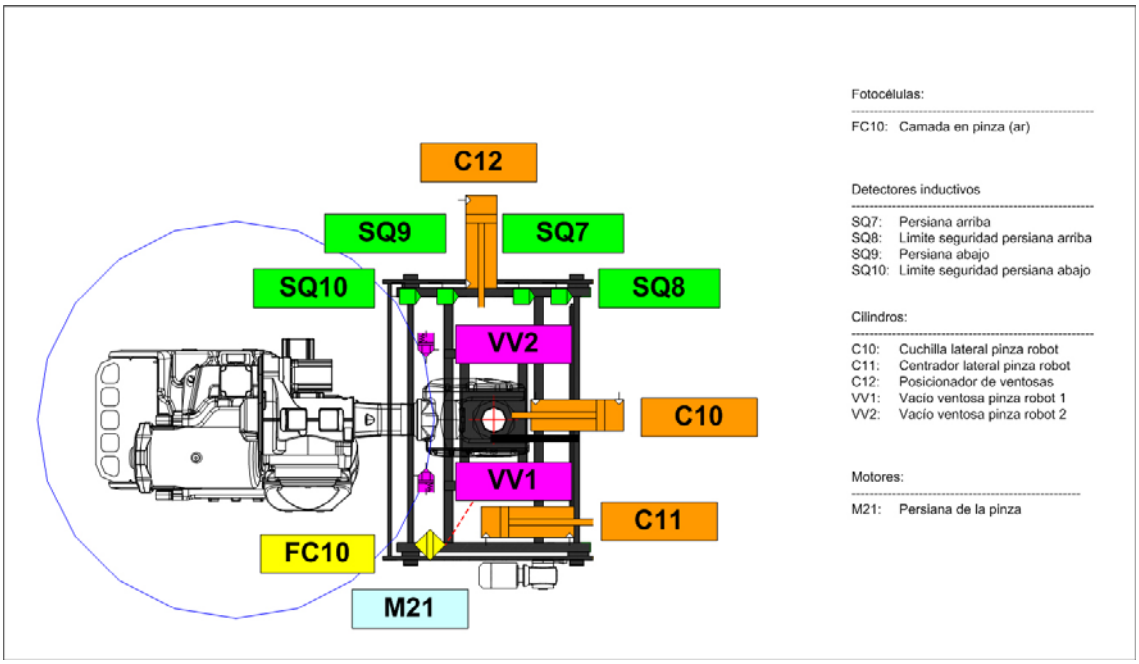


Figura 2.12. Pinza del robot

ET200[3]

Está provisto a su vez de:

- 7 Módulos de entradas digitales 4DI 24V DC, de SIEMENS.
- 4 Módulos de salidas digitales 4DO 24V DC, de SIEMENS.

Los elementos eléctricos que tiene conectado cada módulo son los siguientes.

Entradas:

ET200[3]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
300				
301	E301.0	C9	SQ3010	Pinza Almacén de Palets Retrocedido
	E301.1	C9	SQ3011	Pinza Almacén de Palets Avanzado
	E301.2	C13	SQ3012	Transferencia de Palets 1 Retrocedido
	E301.3	C13	SQ3013	Transferencia de Palets 1 Avanzado
302	E302.0	C14	SQ3020	Transferencia de Palets 2 Retrocedido
	E302.1	C14	SQ3021	Transferencia de Palets 2 Avanzado
	E302.2	C15	SQ3022	Tope Posición de Paletizado Retrocedido
	E302.3	C15	SQ3023	Tope Posición de Paletizado Avanzado

303	E303.0	C16	SQ3030	Centrador Posición de Paletizado Retrocedido
	E303.1	C16	SQ3031	Centrador Posición de Paletizado Avanzado
	E303.2	FR2	PR3032	Presostato Zona Neumatica 2
	E303.3	FC14	SQ3033	Presencia Palet Almacén 1
304	E304.0	FC15	SQ3040	Presencia Transportador de Palets 2
	E304.1	FC16	SQ3041	Presencia Transportador de Palets 3
	E304.2	FC17	SQ3042	Presencia Transportador de Palets 4
	E304.3	FC18	SQ3043	Presencia Transportador de Palets 5
305	E305.0	FC19	SQ3050	Presencia Transportador de Palets 6
	E305.1	FC20	SQ3051	Presencia Transportador de Palets 7
	E305.2	SQ17	SQ3052	Pinza Almacén de Palets Carga
	E305.3	-	LIBRE	-
306	E306.0	FC21	SQ3060	Almacén de Palets Vacío
	E306.1	SQ4	SQ3061	Pinza Almacén de Palets Arriba
	E306.2	SQ5	SQ3062	Pinza Almacén de Palets Medio
	E306.3	SQ6	SQ3063	Pinza Almacén de Palets Abajo
307	E307.0	-	LIBRE	-
	E307.1	-	LIBRE	-
	E307.2	-	LIBRE	-
	E307.3	-	LIBRE	-

Tabla 2.9. Entradas ET200[3]

Salidas:

ET200[3]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
350				
351	A351.0	C9	EV3510	Pinza Almacén de Palets Retroceder
	A351.1	C9	EV3511	Pinza Almacén de Palets Avanzar
	A351.2	C13	EV3512	Transferencia de Palets 1 Retroceder
	A351.3	C13	EV3513	Transferencia de Palets 1 Avanzar
352	A352.0	C14	EV3520	Transferencia de Palets 2 Retroceder

	A352.1	C14	EV3521	Transferencia de Palets 2 Avanzar
	A352.2	C15	EV3522	Tope Posición de Paletizado Retroceder
	A352.3	C15	EV3523	Tope Posición de Paletizado Avanzar
353	A353.0	C16	EV3530	Centrador Posición de Paletizado Retroceder
	A353.1	C16	EV3531	Centrador Posición de Paletizado Avanzar
	A353.2	FR2	EV3532	Válvula de Corte Zona Neumática 2
	A353.3	-	LIBRE	-
354	A354.0		HL3540	Baliza almacén de palets
	A354.1	-	LIBRE	-
	A354.2	-	LIBRE	-
	A354.3	-	LIBRE	-

Tabla 2.10. Salidas ET200[3]

Y mostramos en el siguiente gráfico la zona que abarca el ET200[3]:

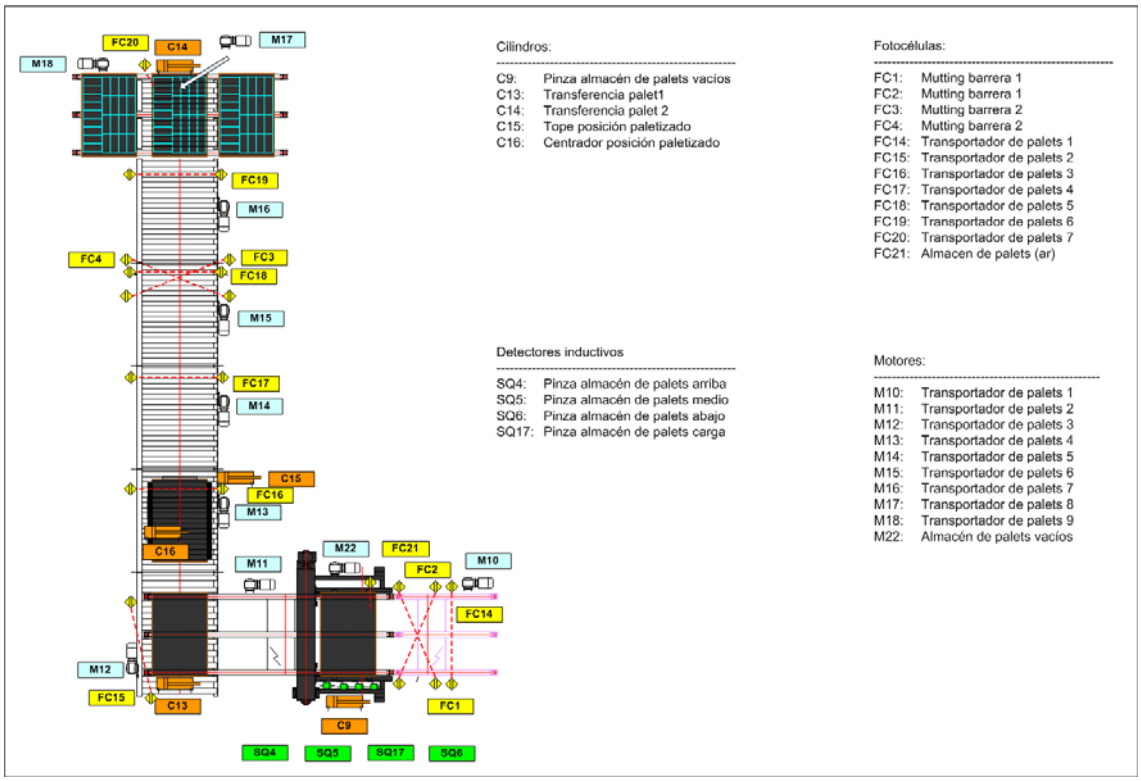


Figura 2.13. Transporte de palets

ET200[4]

Está provisto a su vez de:

- 4 Módulos de entradas digitales 4DI 24V DC, de SIEMENS.
- 2 Módulos de salidas digitales 4DO 24V DC, de SIEMENS.

Los elementos eléctricos que tiene conectado cada módulo son los siguientes:

Entradas:

ET200[4]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
400				
401	E401.0	C17	SQ4010	Dosificador de Cartones Retrocedido
	E401.1	C17	SQ4011	Dosificador de Cartones Avanzado
	E401.2	C18	SQ4012	Centrador Izq de Cartones Retrocedido
	E401.3	C18	SQ4013	Centrador Izq de Cartones Avanzado
402	E402.0	SQ11	SQ4020	Posición Cogida
	E402.1	SQ12	SQ4021	Limite Seguridad Posición Cogida
	E402.2	SQ13	SQ4022	Posición Dejada
	E402.3	SQ14	SQ4023	Limite Seguridad Posición Dejada
403	E403.0	FC22	SQ4030	Presencia de Cartón 1
	E403.1	FC23	SQ4031	Presencia de Cartón 2
	E403.2	FC24	SQ4032	Nivel aviso de cartones
	E403.3	FC25	SQ4033	Almacén de cartones vacío 1
404	E404.0	FC26	SQ4040	Almacén de cartones vacío 2
	E404.1	C18	SQ4041	Centrador Der de Cartones Retrocedido
	E404.2	C18	SQ4042	Centrador Der de Cartones Avanzado
	E404.3	-	LIBRE	-

Tabla 2.11. Entradas ET200[4]

Salidas:

ET200[4]				
Módulo	**Dirección**	**Elemento**	**Nombre**	**Descripción**
450				
451	E451.0	C17	EV4510	Dosificador de Cartones Retroceder
	E451.1	C17	EV4511	Dosificador de Cartones Avanzar
	E451.2	C18	EV4512	Centrador de Cartones Retroceder
	E451.3	C18	EV4513	Centrador de Cartones Avanzar
452	E452.0	VP1	EV4520	Válvula de corte
	E452.1	-	LIBRE	-
	E452.2	-	LIBRE	-
	E452.3	-	LIBRE	-

Tabla 2.12. Salidas ET200[4]

Por último, mostramos el sinóptico para el ET200[4]:

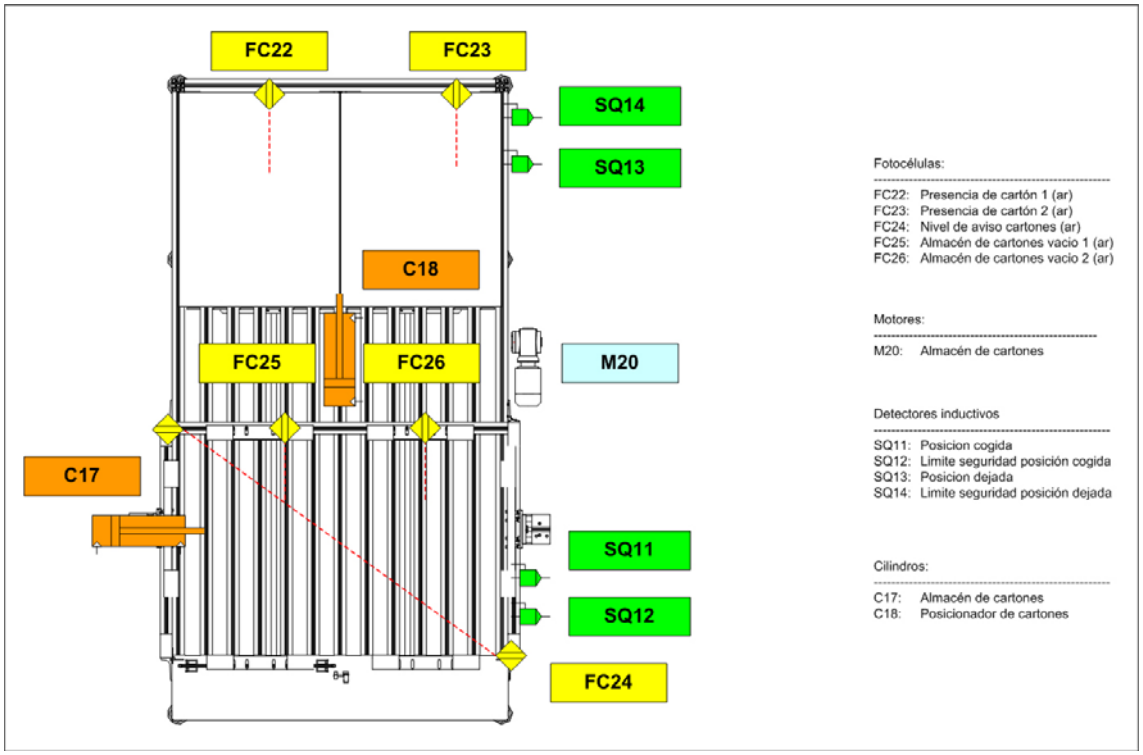


Figura 2.14. Almacén de cartones

2.5.3 Entradas y salidas digitales: Robot ABB

El robot dispone de una tarjeta "d352" que suministra el propio fabricante, ABB, y del cual nos ofrece una posibilidad de 128 entradas y 128 salidas digitales.

En este caso, la asignación que se ha efectuado en la conexión eléctrica para la comunicación del PLC con el ROBOT de ABB es la siguiente:

Entradas al PLC:

Dirección	Señal	Descripción
E40.0	RobotInMon	Robot Motores ON
E40.1	RobotInAut	Robot en Automático
E41.1	RobotInPcp	Robot en Posición Carga Preparación
E41.2	RobotInFcp	Robot Fin Carga Preparación
E41.3	RobotInPcc	Robot en Posición Carga Cartón
E41.4	RobotInFcc	Robot Fin Carga Cartón
E41.5	RobotInPdp	Robot en Posición Descarga Preparación
E41.6	RobotInFdp	Robot Fin Descarga Preparación
E41.7	RobotInPdc	Robot en Posición Descarga Cartón
E42.0	RobotInFdc	Robot Fin Descarga Cartón
E42.1	RobotInHome	Robot en Home
E42.2	RobotInMantenimiento	Robot en posición Mantenimiento
E42.3	RobotInDejadaPinza	Robot en posición Dejada Pinza
E42.4	RobotInEspPcp	Robot Esperando Permiso Coger Preparación
E42.5	RobotInEspPcc	Robot Esperando Permiso Coger Cartón
E42.6	RobotInEspPdc	Robot Esperando Permiso Dejar Cartón
E42.7	RobotInEspPdp	Robot Esperando Permiso Dejar Preparación

Tabla 2.13. Entradas al PLC del Robot

Salidas del PLC:

Dirección	Señal	Descripción
A40.0	RobotOutMon	Motor a ON
A40.1	RobotOutMoff	Motor a OFF
A40.2	RobotOutAon	Modo Automático a ON
A40.3	RobotOutAoff	Modo Automático a OFF
A40.4	RobotOutIniProg	Inicializar Programa
A40.5	RobotOutBit0	Altura a dejar Bit 0
A40.6	RobotOutBit1	Altura a dejar Bit 1
A40.7	RobotOutBit2	Altura a dejar Bit 2
A41.0	RobotOutPonerCarton	Con Cartón
A41.1	RobotOutPcp	Permiso Coger Preparación
A41.2	RobotOutPc	Preparación Cargada
A41.3	RobotOutPcc	Permiso coger cartón
A41.4	RobotOutCc	Cartón cargado
A41.5	RobotOutPdp	Permiso dejar preparación
A41.6	RobotOutPd	Preparación dejada
A41.7	RobotOutPdc	Permiso Dejar Cartón
A42.0	RobotOutCd	Cartón Descargado
A42.1	RobotOutIrHome	Ir a HOME
A42.2	RobotOutIrPm	Robot ir a Posición Mantenimiento
A42.3	RobotOutIrPrp	Ir a Posición de Retirada de Pinza
A42.4	RobotOutDPcp	Presencia Cargada en Pinza
A42.5	RobotOutSegPer	Seguridad de persiana posicionada

Tabla 2.14. Salidas del PLC hacia el Robot

2.6 Comunicación entre los periféricos: PROFIBUS-DP

Dada las dimensiones del sistema, y los diversos periféricos que lo forman, ya se comentó en el punto anterior la necesidad de usar una periferia descentralizada.

Es por ello, que es necesario la utilización de algún tipo de red, donde puedan intercambiarse información cada uno de los periféricos que conforman el sistema.

Para realizar dicha comunicación, estos elementos han de usar un lenguaje conocido por ambos. Se ha buscado un lenguaje fiable, abierto, estándar e independiente del fabricante y que por supuesto, sea viable, y se ha optado por la decisión de usar una red **PROFIBUS-DP** (*DP = Decentralized Peripherals*).

PROFIBUS-DP es el sistema de bus rápido y estandarizado para el nivel de campo. Está normalizado según EN 50170 y IEC 61158-3 Ed2.

A través del bus de campo PROFIBUS-DP es posible interconectar componentes de automatización como autómatas programables SIMATIC, controles numéricos SINUMERIK, sistemas de regulación SIMADYN y soluciones que incluyan SIMATIC M7/PC industriales SIMATIC.

Las características de este protocolo son las siguientes:

- Maestro único
- El PLC se comunica con telegramas de longitud constante
- Se ajusta a los requisitos fundamentales de tiempo
- Transmisión cíclica (PLC)
- Transmisión de valores de consigna
- Realimentación de valor real
- Nuevos valores de consigna calculados
- Nueva transmisión de valores de consigna
- Lectura de parámetros - utilizando el canal PCV
- Escritura de parámetros - utilizando el canal PCV

- Lectura de descripción del parámetro - utilizando el canal PCV

El protocolo DP dispone además de las siguientes funciones:

- Varios fabricantes de PLC lo utilizan para la comunicación de E/S periférica remota.
- Soporta la comunicación cíclica.
- El servicio SRD (Envío/Recepción de datos) proporciona un intercambio cíclico rápido de los datos del proceso entre maestro y esclavos.
- Se admite la función de Mantener y sincronizar.
- Estructura de datos fijos.
- Tamaño de telegrama fijo.
- Ocupa espacio de memoria E/S en PLC proporcional al número de esclavos empleados, lo que puede limitar el número de participantes. Los datos adicionales requieren espacio de memoria de E/S adicional.

DP deberá utilizarse cuando sea necesario un control de proceso cíclico rápido. Este concepto normalmente requeriría un funcionamiento con un sólo maestro y un número limitado de estaciones esclavas Un número elevado de esclavos aumentará la respuesta del sistema. Este caso también podría darse cuando los bucles de control se cierran sobre el bus. Como alternativa muy rápida se puede optar por cerrar el bucle de control fuera del bus.

Aunque actualmente existen varias versiones del protocolo PROFIBUS DP (DP V0, DP V1 y actualmente DP V2), que permiten usar más de un maestro, o llevar una comunicación acíclica, entre otras cosas, para el proyecto se ha decidido usar la primera versión de todas, por lo que no ahondaremos en las características de las nuevas versiones.

2.6.1 Topología de BUS

La red cuenta pues, con un bus de campo, al que se conectan todos los elementos que necesitan enviar y transmitir información digital.

Mostramos a continuación un esquema de la red PROFIBUS que se ha usado en el sistema:

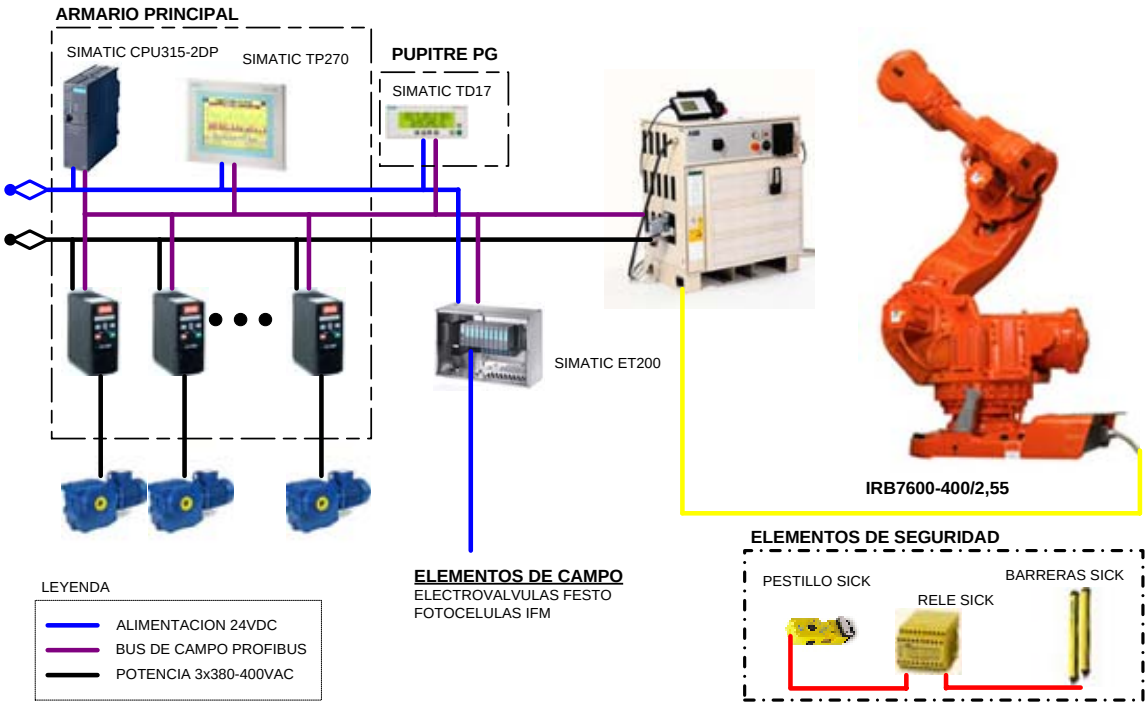


Figura 2.15. Esquema PROFIBUS

Su puede observar los siguientes elementos que lo forman:

Por un lado, disponemos dentro del armario de los siguientes elementos:

- SIMATIC CPU315-2DP. Es la CPU de nuestro PLC, el S7-330 de SIEMENS, como ya se ha mencionado con anterioridad. Actuará como maestro, gobernando y controlando todas las señales que rigen el sistema.

La dirección PROFIBUS asignada es: 02

- SIMATIC TP270. Es una pantalla táctil de HMI, para que el operario pueda realizar fácilmente tareas de control y supervisión del sistema, sin necesidad de tener que estar programando directamente con el PLC. Actúa como esclavo de la CPU del autómata.

  Su dirección asignada para la red PROFIBUS es la 41.

- VLT2800. Son los variadores de frecuencia que controlan los movimientos de los motores. Actúan como esclavos. En el siguiente apartado se explicará con detalle cómo se comunican con el autómata, pues su forma de tratarlo es especial y es necesario tenerlo claro para posteriormente poder programar adecuadamente el autómata para su control.

De los 22 motores que hay en el sistema, han sido necesarios utilizar 15 variadores de frecuencia. Los 8 restantes se arrancan y se paran directamente atacando a los contactores y térmicos que llevan asociados cada uno conectados a las entradas y salidas del PLC.

La dirección PROFIBUS de cada uno de éstos se ha hecho coincidir con la numeración que se le ha dado a cada motor. Es decir:

El variador de frecuencia VM04, que controla el motor M04, tiene asignada la dirección PB: 04.

El variador de frecuencia VM05, que controla el motor M04, tiene asignada la dirección PB: 04.

Y así sucesivamente para los variadores de frecuencia VM06, VM07, VM08, VM09, VM13, VM14, VM15, VM16, VM17, VM18, VM19, VM20 y VM21.

Todos éstos elementos mencionado anteriormente (PLC, TP270 y variadores de frecuencia) están situados en el propio **armario principal**.

Fuera de este, nos encontramos con:

- SIMATIC TD17. Es otra pantalla de HMI, pero más pequeña y más limitada que la pantalla táctil TP270. En este caso, es una pantalla de texto con botones que usaremos para visualizar las alarmas existentes y el poder realizar movimientos manuales. Igualmente, actúa como esclavo en la red PROFIBUS-DP y está situado en el exterior del **pupitre PG**.

  Tiene asignada la dirección PB: 42.

- ET200. Estos periféricos de SIEMENS, están distribuidos por el sistema para controlar los elementos de campo (electroválvulas de FESTO, fotocélulas de IFM, etc…). En el apartado anterior se detallaron qué elementos tiene conectado cada uno de los 4 ET200 que se han utilizado.

La dirección PROFIBUS que tiene asignado cada uno de ellos son:

ET200[1]: 31
ET200[2]: 32
ET200[3]: 33
ET200[4]: 34

- ROBOT ABB. El Robot, cuya referencia es IRB 7600, de la empresa ABB, se comunica con un interfaz definido por la misma empresa con el pupitre que lo controla. Como el funcionamiento de éste elemento no es objeto de estudio de éste proyecto, no ahondaremos más en su funcionamiento. Lo que sí pretendemos reseñar es que, dicho pupitre, es donde se ubica la tarjeta de comunicaciones ProfibusDP esclavo, donde se conectará a la red PROFIBUS-DP.

  Tiene asignada la dirección 40.

2.6.2 El variador de frecuencia VLT 2800

El Fieldbus PROFIBUS se ha diseñado para ofrecer una flexibilidad y un control sin precedentes sobre el sistema controlado. El PROFIBUS actuará como una pieza

integrada del convertidor de frecuencias, dándole acceso a todos los parámetros relevantes para su aplicación. El convertidor de frecuencias siempre actuará como esclavo, y combinado con un maestro puede intercambiar multitud de información y comandos. Señales de control como referencia de velocidad, arranque/parada del motor, operación de cambio de sentido, etc., se transmiten desde el maestro en forma de telegrama. El convertidor de frecuencia acusa recibo transmitiendo al maestro señales de estado, como en funcionamiento, en referencia, motor parado etc. El convertidor de frecuencias puede además transmitir indicaciones de fallos, alarmas y advertencias al maestro, como Sobreintensidad o Pérdida de fase.

El PROFIBUS se comunica según el estándar del field bus PROFIBUS, EN 50170, parte 3. Puede intercambiar datos con todos los maestros que cumplen esta norma; sin embargo, esto no quiere decir que se apoyen todos los servicios disponibles en la norma de perfil de PROFIDRIVE. El perfil de PROFIBUS para los convertidores de frecuencia (versión 2 y en parte versión 3, PNO) es una parte de PROFIBUS que apoya solamente los servicios relacionados con las aplicaciones de control de la velocidad.

Colaboradores de comunicación

En un sistema de control, el convertidor de frecuencia siempre actuará como esclavo, y como tal puede comunicarse con un único maestro o con varios, según la naturaleza de la aplicación. Un maestro puede ser un PLC o un PC que esté equipado con una tarjeta de comunicaciones PROFIBUS. En nuestro caso, es el PLC S7300 de SIEMENS.

Descripción de PPO

Una característica especial del Perfil PROFIBUS para convertidores de frecuencias es el objeto de comunicación llamado PPO, que significa ***Parámetro-Objeto de datos de proceso***.

El PPO está indicado para la transferencia de datos cíclica rápida, y puede, como el nombre indica, transportar datos de proceso y parámetros. La selección del tipo PPO se realiza según la configuración del maestro.

Un PPO puede constar de una parte de parámetro y otra de datos de proceso. El componente de parámetro se puede utilizar para leer o actualizar los parámetros de uno en uno. El componente de datos del proceso consta de una parte fija (4 bytes) y una parte con parámetros (8 o 16 bytes). En la parte fija, el código de control y la referencia de velocidad se transfiere al convertidor de frecuencia mientras que el código de estado y la retroalimentación de la frecuencia de salida real se transmiten desde el convertidor de frecuencia. En la parte con parámetros, el usuario elige los parámetros que deben transferirse al convertidor de frecuencia (parámetro 915) y desde él (parámetro 916).

Hay 8 tipos de PPO para seleccionar, cada uno de ellos con la siguiente estructura:

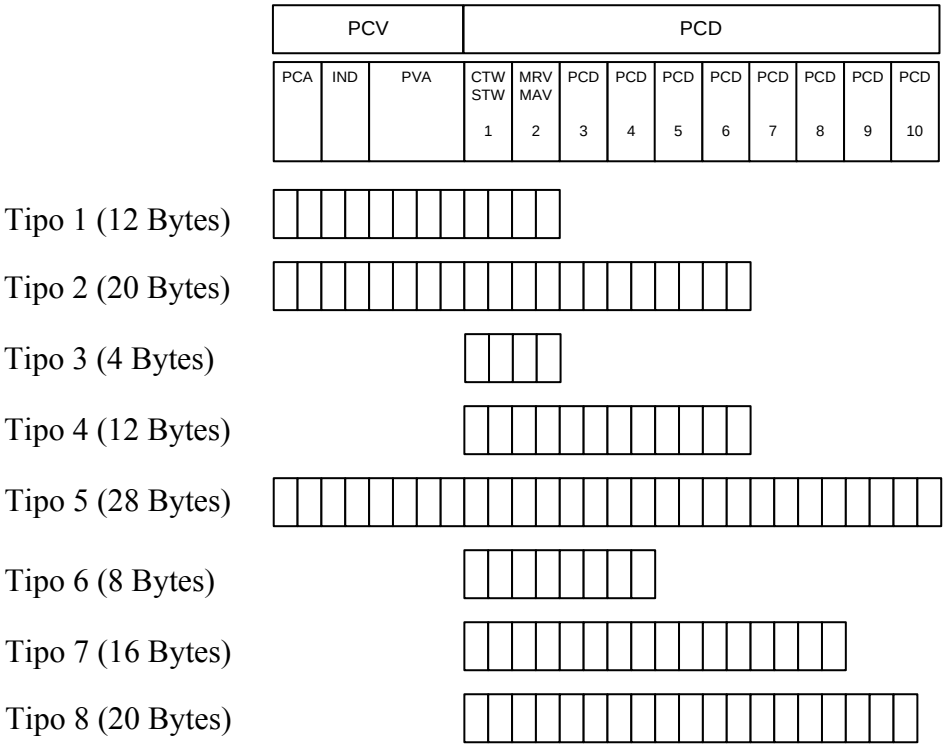


Figura 2.16. Tipos de PPO

Donde:

PCV: Parámetro-Características-Valor

PCD: Datos de proceso

PCA: Características de parámetros (Bytes 1, 2)

IND: Subíndice (Byte 3), (Byte 4 no se utiliza)

PVA: Valor del parámetro (Bytes 5 a 8)

CTW: Código de control

STW: Código de estado

MRV: Valor de referencia principal

MAV: Valor real principal (frecuencia de salida real)

Para nuestro caso de estudio, se eligió el Tipo 1 (el más sencillo de todos que contiene canales PCV y PCD) de longitud 12 bytes, el cual pasamos a detallar.

Como se puede comprobar en la imagen, la trama está compuesta de dos partes diferenciadas:

- El PCV (Parámetro Característica Valor)
- El PCD (Datos de Proceso).

El PCV, que tiene una longitud total de 8 bytes, está compuesto a su vez por tres partes diferenciadas:

- PCA (Característica de parámetros)
- IND (Subíndice)
- PVA (Valor del Parámetro)

El PCA tiene una longitud de 2 bytes y su estructura es la siguiente:

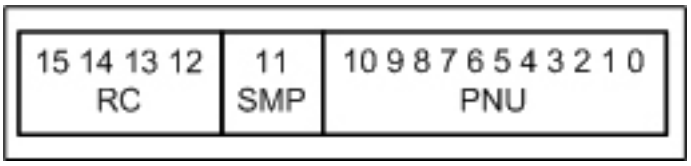


Figura 2.17. PCA

Se observa que se divide a su vez en:

- RC (4 bits): Característica de petición/respuesta
- SMP (1 bit): Cambio de bit para mensajes espontáneos
- PNU (11 bits): Número de Parámetro

El PCA se utiliza para que el maestro pueda controlar y supervisar los parámetros y pedir una respuesta al esclavo, mientras que éste, además de responder a la solicitud, puede transmitir un mensaje espontáneo.

Para ello, en el RC, se define la petición que debe transmitir el master al esclavo, además de qué otros componentes del PCV (IND y PVA) participan.

Cuando se realiza la petición, el RC puede tomar los siguientes valores:

00 Sin petición

01 Leer valor de parámetro

02 Cambiar valor de parámetro (código)

03 Cambiar valor de parámetro (código largo)

06 Leer valor de parámetro (grupo)

07 Cambiar valor de parámetro (código de grupo)

08 Cambiar valor de parámetro (código largo de grupo)

Mientras, cuando el esclavo manda la respuesta al maestro, el RC puede contener los siguientes valores:

0 Sin Respuesta

1 Transferir valor de parámetro (código)

2 Transferir valor de parámetro (código largo)

3 Transferir elemento de descripción

4 Transferir valor de parámetro (código de grupo)

5 Transferir valor de parámetro (código largo de grupo)

6 Transferir número de elementos de grupo

7 Petición rechazada (incluido número de fallos)

8 No se puede realizar tareas de mantenimiento (servicio) mediante la interfaz PCV

El bit SPM se utiliza en el caso de que sea necesario utilizar la función de Mensajes Espontáneos (parámetro 917). No ahondaremos en la función de este bit pues en nuestro caso no será activado y por tanto su valor siempre será 0.

El PNU llevará el número de parámetro a controlar.

Por otro lado, siguiendo con el PCV, si la petición/respuesta contiene elementos de grupo, el IND transportará el Subíndice de grupo. Tiene una longitud de 2 bytes.

Y por último, en el PVA se pondrá el valor a escribir en el parámetro determinado en el PNU (bytes 5 al 8 del PCA) cuando se realiza una petición. En la respuesta se incluirá el valor leido o algún código de error si se dio el caso.

Por otro lado, en el PCD, en la trama de tipo 1, se corresponde con una longitud de 4 bytes compuesta por:

- CTW/STW (2 bytes): Código de contro/Código de Estado
- MRV/MAV (2 bytes): Valor de Referencia Principal/Valor Actual Principal

Los bits del "código de control" (CTW) comunican al convertidor de frecuencia cómo debe reaccionar, mientras que el estado de bit del "código de estado" (STW) comunica al maestro el estado del convertidor de frecuencia.

Así, mientras que los códigos de control se utilizan para enviar comandos de control al convertidor de frecuencia cuando el maestro envía el telegrama. Cuando el convertidor de frecuencia devuelve el marco al maestro, los dos mismos bytes funcionan como estado desde el convertidor de frecuencia.

Los valores del CTW dependen del valor del parámetro 512 (“Tipo de Telegrama”), que en nuestro caso lo tomaremos como valor 0 (perfil PROFIDRIVE). Es por tanto que los valores posibles de estos bits son:

Bit	Bit = 0	Bit = 1
00 (Bit menos significativo)	NO 1	SÍ 1
01	NO 2	SÍ 2
02	NO 3	SÍ 3
03	Inercia del motor	Activar
04	Parada rápida	Rampa
05	Mantener frecuencia de salida	Rampa activa
06	Parada de rampa	Arranque
07	Sin función	Reset
08	Jog 1 DESACTIVADA	SÍ

09	Jog 2 DESACTIVADA	SÍ
10	Dato no válido	Válido
11	Sin función	Enganche abajo
12	Sin función	Enganche arriba
13	Seleccionar ajuste BIT MENOS SIGNIFICATIVO	
14	Seleccionar ajuste BIT MÁS SIGNIFICATIVO	
15 (Bit más significativo)	Sin función	Cambio de sentido

Tabla 2.15. Valores del CTW

Por otro lado, cuando el esclavo responde, los posibles valores del código de estado (STW) pueden ser los siguientes:

Bit	**Bit = 0**	**Bit = 1**
00 (Bit menos significativo)	Control no preparado	Preparado
01	VLT no preparado	Preparado
02	Inercia del motor	Activar
03	Sin fallo	Desconexión
04	SÍ 2	NO 2
05	SÍ 3	NO 3
06	Parada activada	Arranque desactivado
07	Sin advertencia	Advertencia
08	Referencia de velocidad	Veloc. = ref.
09	Funcionamiento local	Control de bus
10	Fuera de rango	Frecuencia OK
11	No está en funcionamiento	En funcionamiento
12		
13	Tensión OK	Límite
14	Par OK	Límite

15 (Bit más significativo)	Sin advertencia térmica	Advertencia térmica

Tabla 2.16. Valores del STW

Por último, en los bits MRV, se describe la referencia de velocidad y de arranque que se desea que tenga el motor. El valor se indica en tanto por ciento en función de la frecuencia máxima (especificada en el parámetro 202). Así, 0 Hex se corresponde con 0% y 4000 Hex con 100%.

En el bit MAV el esclavo indica al maestro (PLC) la referencia de velocidad actual con la que está funcionando el motor.

Para los otros tipos de trama, la estructura difiere claramente, algunos, por ejemplo, no incluyen la interfaz PCA (tipos 3, 4, 6, 7 y 8), por lo que el tratamiento de los parámetros se realiza en exclusiva por el PCD. Pero dado que esto no es objeto de estudio del proyecto, no se va a explicar con más detalles.

Parámetros

Como ya se ha comentado, el VLT2800 se basa en una serie de parámetros fijos que sirven para configurar el dispositivo. Dispone de casi 1000 parámetros distintos, que se pueden clasificar de la siguiente manera:

- Funcionamiento y display (a partir del parámetro 0): Idioma, Control local/Remoto, Referencia local, Menú rápido del usuario, Ajuste del menú rápido, etc…
- Carga y motor (a partir del parámetro 100): Potencia del motor, Tensión del motor, frecuencia del motor, intensidad del motor, velocidad nominal del motor, adaptación automática del motor, etc…
- Referencia y límites (a partir del parámetro 200): Referencia Máxima, Referencia Mínima, Tiempo rampa de aceleración, tiempo rampa desaceleración, etc…
- Señales de entrada y de salidas (a partir del parámetro 300): Entrada digitales, salidas digitales, salida relé, etc…

- Funciones especiales (a partir del parámetro 400): Función de freno, función de reset, ganancia proporcional de PID de velocidad, tiempo de integral de PID de velocidad, tiempo de diferencial de PID de velocidad, tiempo de filtro de paso bajo de PID de proceso, etc…
- Comunicación serie (a partir del parámetro 500): dirección, velocidad en baudios, velocidad fija de bus 1, tipo de telegrama, etc…
- Funciones técnicas (a partir del parámetro 600): horas de funcionamiento, horas ejecutadas, contador de kWh, número puestas en marcha, registro de fallos, etc…
- Parámetros específicos de PROFIBUS (a partir del parámetro 800): selección de protocolo, retardo de bus, selección del tipo de PPO para DP, Mensajes espontáneos activos, configuración de PCD para escritura/lectura, dirección de estación, autoridad de funcionamiento PCV, etc…

Todos estos parámetros se pueden configurar de manera manual y también desde el PLC (maestro) programándolos convenientemente usando el protocolo de comunicación PROFIBUS-DP, tal y como se ha descrito anteriormente.

2.7 Datos Técnicos del Hardware

En este apartado indicaremos la información técnica que nos suministra el fabricante de los elementos Hardware más relevantes, para la programación del autómata.

2.7.1 PLC

Como ya se ha comentado, el autómata que se ha utilizado es el SIMATIC S7 de la serie 300 de SIEMENS.

La composición del autómata está formada por:

- CPU 315-2 DP
- 2 módulos de Entradas Digitales SM321

- 2 módulos de Salidas Digitales SM322

Todo integrado en el mismo bastidor.

CPU 315-2 DP

Figura 2.18. CPU 315-2DP

Los datos técnicos más relevantes de esta CPU son los siguientes:

- Tensión de alimentación de 24 V DC.
- Consumo de corriente de 60 mA al conectar en vacío
- Intensidad de 2,5 A al conectar
- Consumo de intensidad (valor nominal) de 0,8 A
- Potencia disipada de 2,5 W
- Memoria de trabajo de 128 Kbytes no ampliables
- Memoria de carga insertable mediante MMC (máx. 8 Mbytes)
- Tiempos de ejecución de mín. 0,1 µseg para operaciones de bits, mín. 0,2 µseg para operaciones de palabras, mín. 2,0 µseg para aritmética en coma fija y mín. 3,0 µseg para aritmética en coma flotante.
- 256 contadores S7
- 256 Temporizadores S7
- 2048 bytes de marcas

- 1 byte de marca de ciclo
- 1023 bloques de datos de 16Kbytes
- Máx. 1024 bloques cargables
- Máx. 2048 bytes de áreas de direccionamiento, 2000 de ellos descentralizados
- 128 de imagen de proceso E/S
- Máx. 16384 canales digitales (máx. 1024 descentralizados)
- Máx. 1024 canales analógicos (máx. 256 descentralizados)
- Máx. 8 bastidores admisibles (8 módulos máx. por cada bastidor)
- 1 maestro DP integrado
- 1 Interfaz RS 485 integrada para comunicación MPI
- Velocidad de transferencia de 187,5 Kbaudios en comunicación MPI
- 1 Interfez RS 485 integrada para comunicación PROFIBUS DP
- Velocidad de transferencia de hasta 12 Mbaudios en comunicación PROFIBUS DP
- Máx. 124 esclavos DP por estación
- Máx. 244 bytes de área de direccionamiento para comunicación PROFIBUS DP
- Lenguaje de programación admitido: KOP/FUP/AWL

Módulo de 32 Entradas Digitales SM321

Figura 2.19. SM31

El módulo SM 321; DI 32 x DC 24 V se distingue por las propiedades siguientes:

- 32 entradas, separadas galvánicamente en grupos de 16
- Tensión nominal de entrada 24 V c.c.
- Adecuado para conmutadores y detectores de proximidad (BERO) a 2/3/4 hilos

Módulos de 32 Salidas Digitales SM322

Figura 2.20. SM322

El módulo SM322; DO 32 x DC 24 V/0,5 A se distingue por las propiedades siguientes:

- 32 salidas, separadas galvánicamente en grupos de 8
- Intensidad de salida 0,5 A
- Tensión nominal de carga 24 V c.c.
- Adecuado para electroválvulas, contactores de c.c. y lámparas de señalización

2.7.2. ET200

En el sistema a diseñar, se ha hecho uso de 4 equipos ET200 S de SIEMENS. Cada uno compuesto por los siguientes elementos.

ET200[1]

Enumeramos según el orden en el que están conectados en el propio módulo:

- 1 Módulo de interfaz IM 151-1 Standard
- 1 Módulo de potencia PM-E 24 V DC
- 9 Módulos de entradas digitales 4DI 24V DC
- 1 Módulo de potencia PM-E 24 V DC
- 5 Módulos de salidas digitales 4DO 24V DC

ET200[2]

En este caso, dispone de los siguientes elementos:

- 1 Módulo de interfaz IM 151-1 Standard
- 1 Módulo de potencia PM-E 24 V DC
- 5 Módulos de entradas digitales 4DI 24V DC
- 1 Módulo de potencia PM-E 24 V DC
- 3 Módulos de salidas digitales 4DO 24V DC

ET200[3]

- 1 Módulo de interfaz IM 151-1 Standard
- 1 Módulo de potencia PM-E 24 V DC
- 7 Módulos de entradas digitales 4DI 24V DC
- 1 Módulo de potencia PM-E 24 V DC
- 4 Módulos de salidas digitales 4DO 24V DC

ET200[4]

- 1 Módulo de interfaz IM 151-1 Standard
- 1 Módulo de potencia PM-E 24 V DC
- 4 Módulos de entradas digitales 4DI 24V DC
- 1 Módulo de potencia PM-E 24 V DC
- 2 Módulos de salidas digitales 4DO 24V DC

Mostramos a continuación una imagen real del ET200[4], donde se pueden observar los elementos antes comentados:

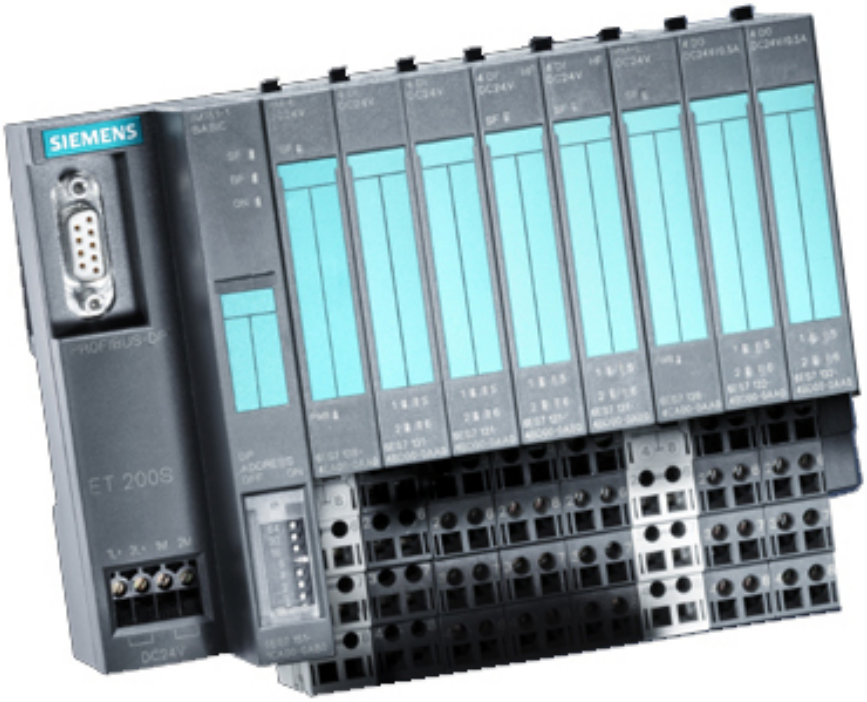


Figura 2.21. ET200

Pasamos a continuación a detallar los datos técnicos más relevantes que nos suministra el fabricante de cada uno de estos componentes.

Módulo de interfaz IM 151-1 Standard

Figura 2.22. IM 151-1 Standard

- Conexión del PROFIBUS DP vía interfaz RS485

- Velocidades de transmisión: 9,6; 19,2; 45,45; 93,75;187,5; 500 kBaudios, 1,5; 3; 6; 12 Mbit/s
- Funcionamiento como esclavo DPV0 o DPV1
- Comunicación directa
- Longitud de parámetros 27 bytes
- Área de direccionamiento de 244 bytes E/S
- Longitud del bus del ET 200S: máx. 2 m (parametrizable)
- Número de módulos enchufables: máx. 63
- Actualizar firmware a través de PROFIBUS DP
- Máxima intensidad de salida de la interfaz PROFIBUS DP de 80mA.
- Tensión nominal de 24 V DC
- Consumo de corriente de la tensión nominal de 200mA.
- Potencia disipada de 3,3 W

Módulo de potencia PM-E 24 V DC

Figura 2.23. PM-E 24 V DC

- Tensión nominal de 24 VDC
- Intensidad de salida de 10 A
- Consumo de corriente de tensión nominal de 4 mA
- Potencia disipada de 100 W

Módulos de entradas digitales 4DI 24V DC

- Tensión nominal de 24 VDC
- Intensidad de salida de 10 A
- Potencia disipada de 0.8 W
- 4 Entradas digitales
- Area de direccionamiento de 4 bits
- Tensión de entrada de 24 V DC
- Intensidad de entrada de 7 mA

Módulos de salidas digitales 4DO 24V DC

- Tensión nominal de 24 VDC
- Consumo de corriente de la tensión nominal de 5 mA
- Consumo de corriente del Bus de 10 mA
- Potencia disipada de 1.4 W
- 4 Salidas digitales
- Area de direccionamiento de 4 bits
- Tensión de salida de 24 V DC
- Intensidad de salida variable de 0.3 mA hasta 0.5 A
- Impendancia de carga variable de 48 Ω hasta 3400 Ω

2.7.3 VLT2800

Figura 2.24. VLT2800

Mostramos en la imagen ejemplo del variador de frecuencia que se ha usado, el VLT 2800 de DANFOSS.

Las características que nos suministra el fabricante en este caso son:

- Tensión de alimentación 380 – 480 VAC (Trifásica)
- Tensión de salida 0-100% de la tensión de red
- Frecuencia de salida de 1-1000 Hz
- Tarjeta de control con 5 Entradas digitales programables
- Tensión de entradas de 24 V DC para las entradas digitales
- Resitencia de entrada de aprox 4 Ω
- Tarjeta de control con 1 entrada analógica de tensión y otra de intensidad
- Tensión de entrada de 0-10 V CC
- Intensidad de entrada de 0/4-20 mA.
- Tarjeta de control con 1 entrada de pulsos programables
- Tarjeta de control con 1 salidad de pulsos programables
- Tarjeta de control con 1 salida analógica programable
- Tajerta de control con 1 salida de 24 V CC, 1 de 10 V CC

- Interfaz RS 485 para comunicación por PROFIBUS
- 1 salida de relé programable
- Tiempo de respuesta menor de 26,6 ms
- Protección IP20

2.7.4 TP 270

Figura 2.25. Pantalla TP270 de SIEMENS

Para el HMI, se ha utilizado el SIMATIC TP 270 de SIEMENS, con pantalla de 6".

Las características técnicas que nos suministra el fabricante son las siguientes:

- Dimensiones externas (ancho por alto en mm.): 212 x 156
- Recorte de montaje (ancho por alto en mm.): 198 x 142
- Protección lateral IP20
- Peso 1 Kg.
- CPU 64 Bit RISC-CPU

- Memoria para configuración de 2 MBytes
- Dispone de ranura para tarjeta CF y PC
- Display de color de 6" tipo CSTN-LCD con Touch
- Tamaño de la imagen visualizable en diagonal 5,7"
- Resolución de 320x240 (QVGA)
- Tensión nominal 24 V DC
- Consumo de 0.9 A de intensidad
- Interfaces disponibles:
  - 2 puertos RS232
  - 1 puerto RS 485 (para comunicación PROFIBUS DP y MPI)
  - 1 puerto USB,

2.8. Software utilizado

Para el desarrollo de este proyecto, será necesario la utilización del siguiente programa software:

- STEP7 V5.3 Es el programa que utilizaremos para programas el PLC SIMATIC S7 300 de SIEMENS.

Se requerirá además cargar los siguientes paquetes opcionales:

- S7-SCL V5.1. Con este paquete tendremos la opción de programar en SCL, lenguaje de programación en que basaremos parte de éste proyecto.
- SIMATIC PROTOOL/PRO CS. Este paquete nos servirá para configurar el HMI (Interfaz Hombre Máquina) que se utilizarán en nuestro proyecto. Esto son las pantallas que gobiernan la TP270 y la TD 17 (que tan sólo visualiza las alarmas).
- SIMATIC PROTOOL/PRO RT (RunTime). Este programa complementa al anterior y se trata de un simulador para comprobar el funcionamiento de las pantallas configuradas con PROTOOL/PRO CS.

Los requisitos para estos programas son los siguientes:

STEP7 V5.3. Según el fabricante, se necesitará un ordenador personal PC Estandar con las siguientes características mínimas:

- Sistema Operativo MS Windows 2000 Professional o MS Windows XP Professional
- Service Pack 3 si se utiliza Windows 2000 Profesional o Service Pack 1 si se utiliza Windows XP instalado.
- Un procesador a 600 Mhz como mínimo
- 256 MB de memoria RAM (recomendables 512 MB)
- Gráficos XGA con una resolución de 1024x768 con 16 Bit profundidad de color
- Disponer de al menos entre 300 y 600 MB de espacio de memoria libre en el disco duro

Para S7-SCL V5.1, los requisitos son los mismos que el programa STEP7 V5.3

Para el paquete SIMATIC PROTOOL/PRO CS, el fabricante nos informa que:

- Sistema Operativo Windows 98 SE y Windows ME (recomendado Windows NT 4.0 Workstation SP 6a, Windows 2000 Professional SP 2, Windows XP Profesional)
- Procesador Pentium II a 233 MHz (recomendando un Pentium III a 500 MHz o superior)
- 64 MB de memoria RAM (recomendado 128 MB)
- Tarjeta gráfica SVGA (recomendado SVGA con aceleración de hardware)
- Resolución 800x600
- Disco duro con más de 360 MBytes libres

Por otro lado, para el paquete SIMATIC PROTOOL/PRO RT, se nos especifica lo siguiente:

- Sistema Operativo Windows 98 SE y Windows ME (recomendado Windows NT 4.0 Workstation SP 6a, Windows 2000 Professional SP 2, Windows XP Profesional)

- Procesador Pentium II a 233 MHz (recomendando un Pentium III a 500 MHz o superior)
- 64 MB de memoria RAM (recomendado 128 MB)
- Tarjeta gráfica VGA (recomendado SVGA con aceleración de hardware)
- Resolución 640x480 (recomendado 800x600)
- Disco duro con al menos 40 MBytes libres (recomendado más de 100 MB).

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:

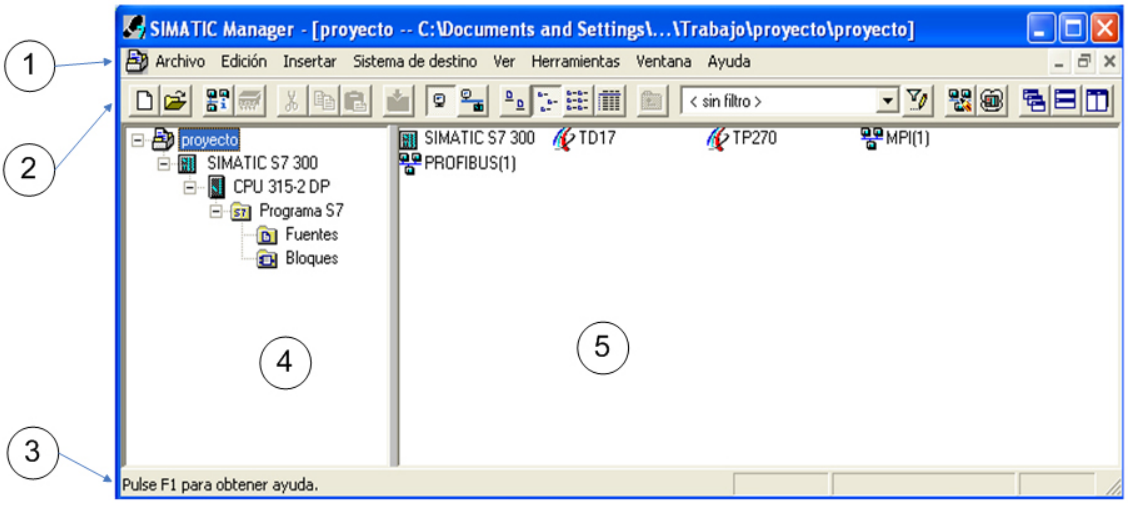


Figura 3.1. Administrador SIMATIC

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:

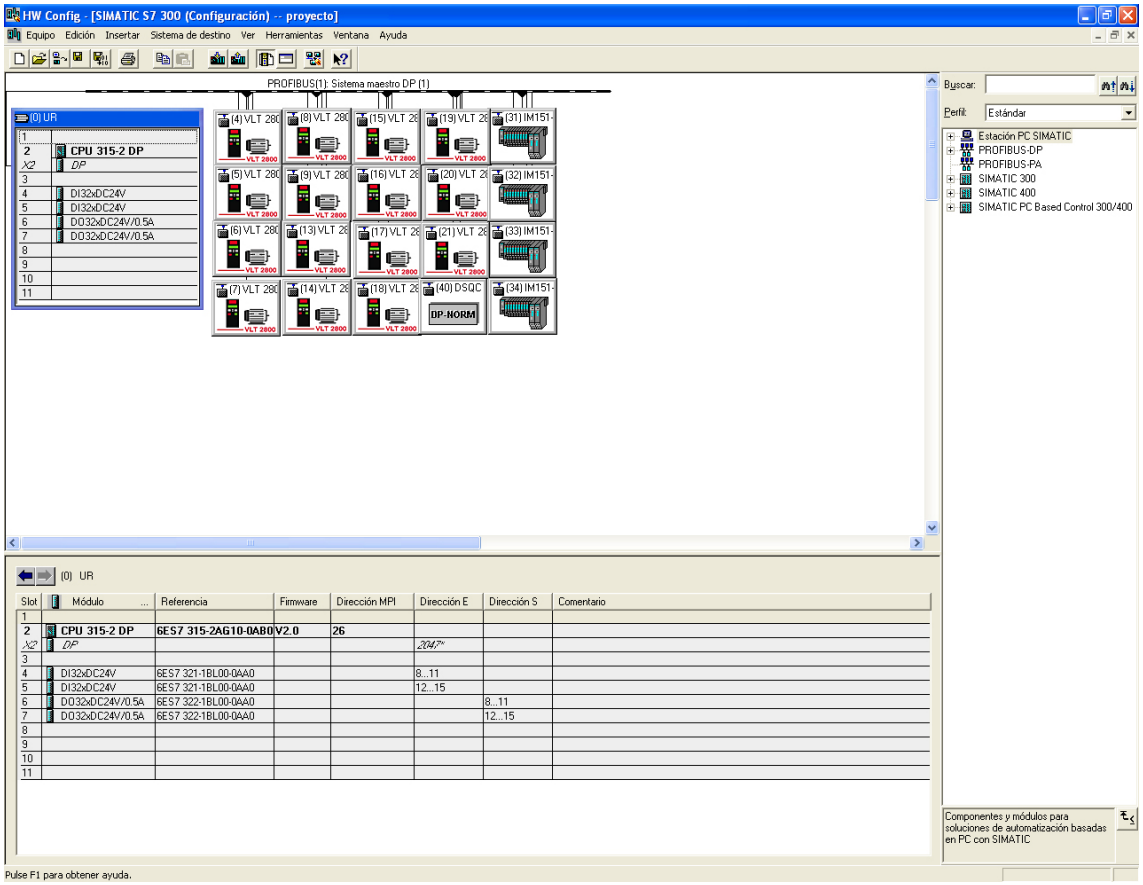


Figura 3.2. HW Config

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.

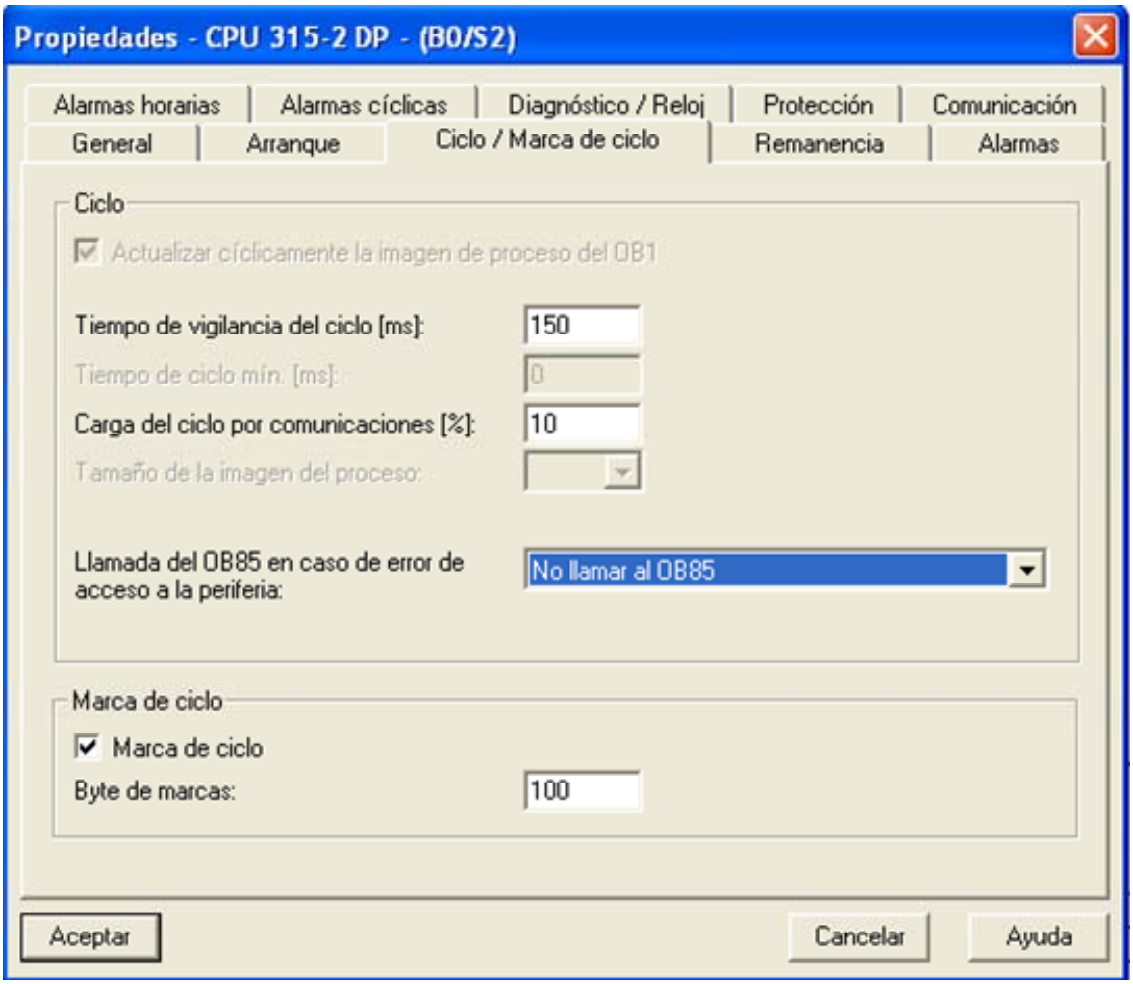


Figura 3.3. Ciclo/Marca Ciclo

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:

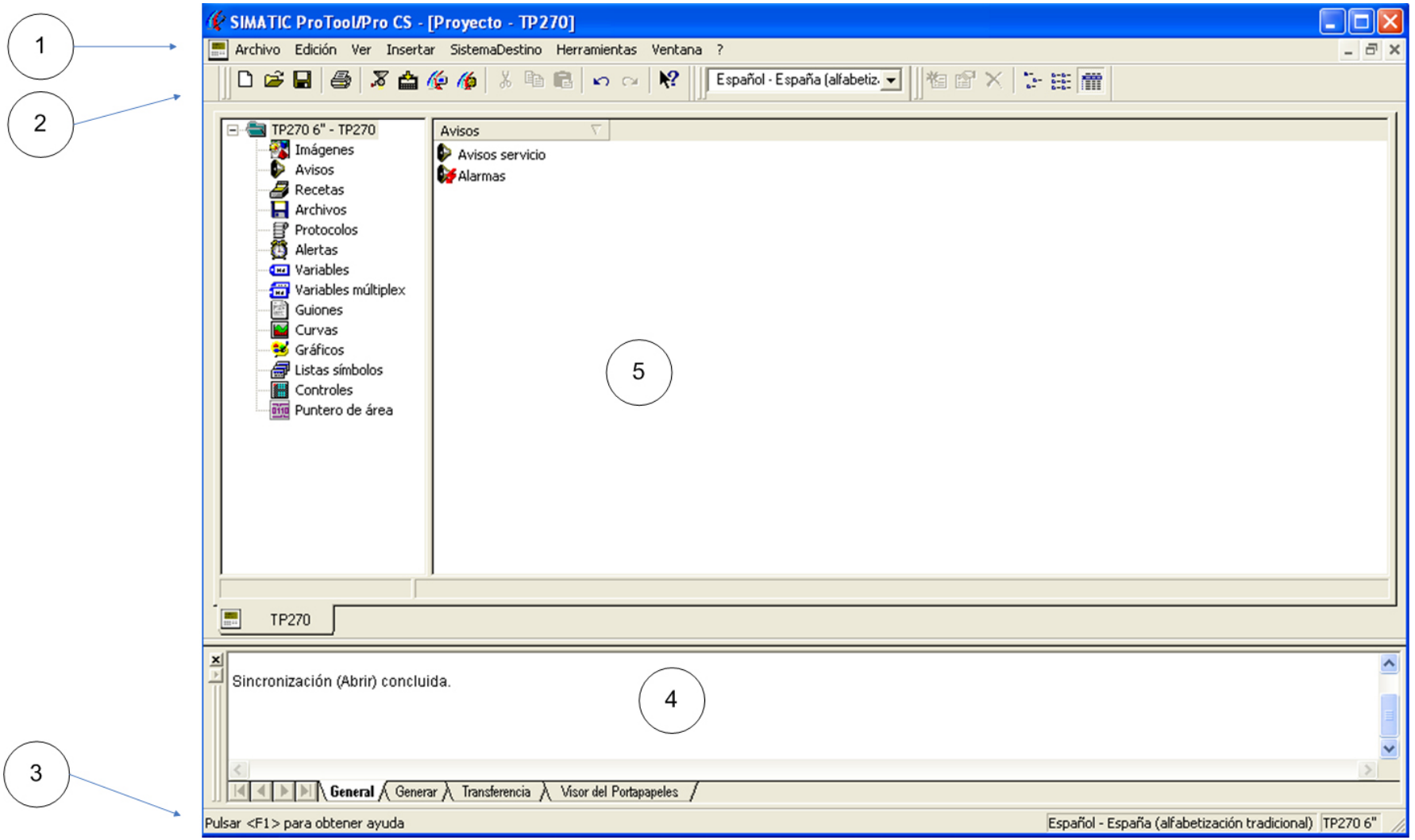


Figura 3.4. PROTOOL/PRO CS

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:

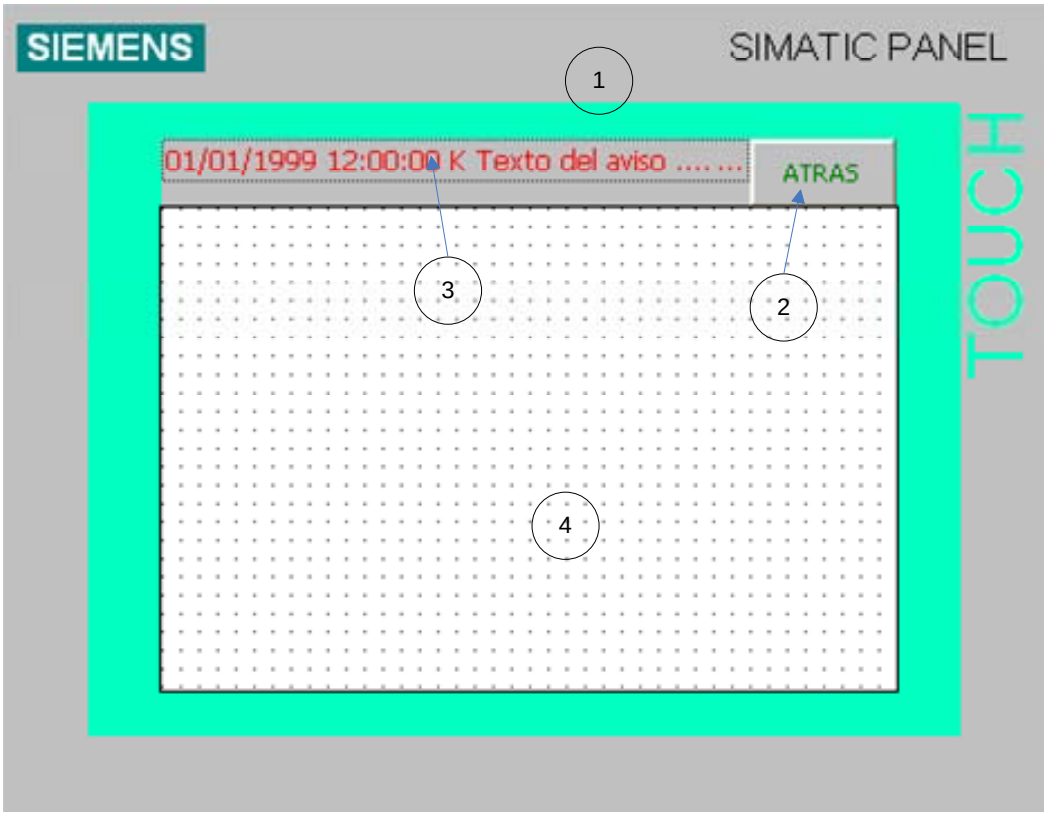


Figura 3.5. Pantalla base del HMI

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:

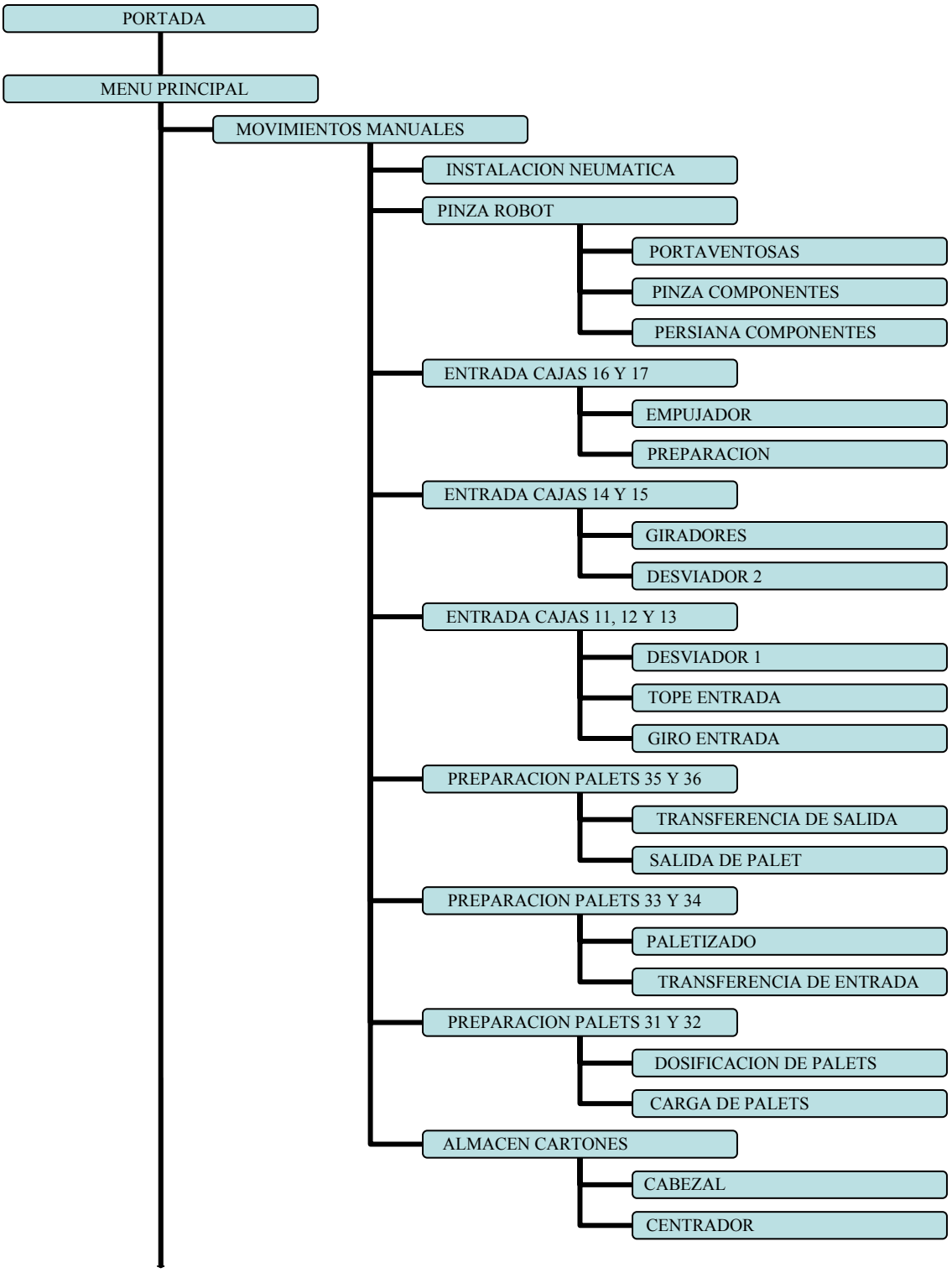


Figura 3.6. Estructura de las pantallas HMI 1

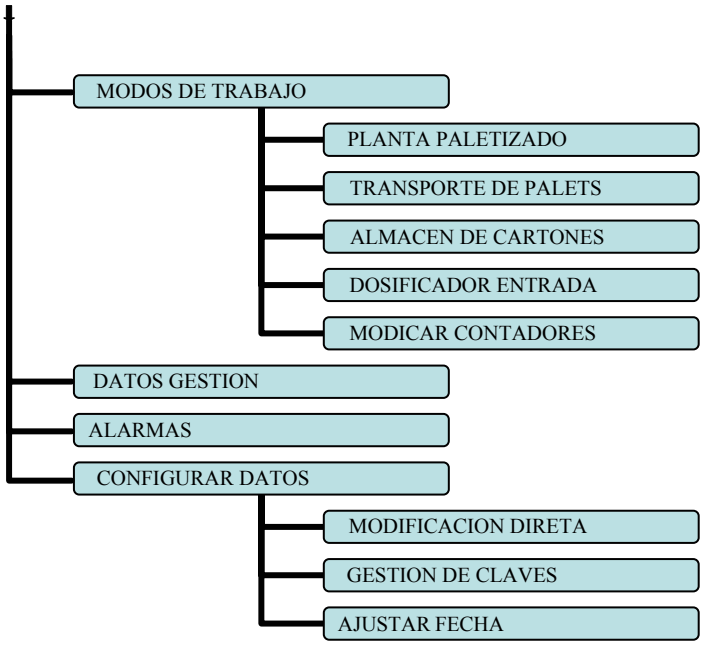


Figura 3.7. Estructura de las pantallas HMI 2

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.

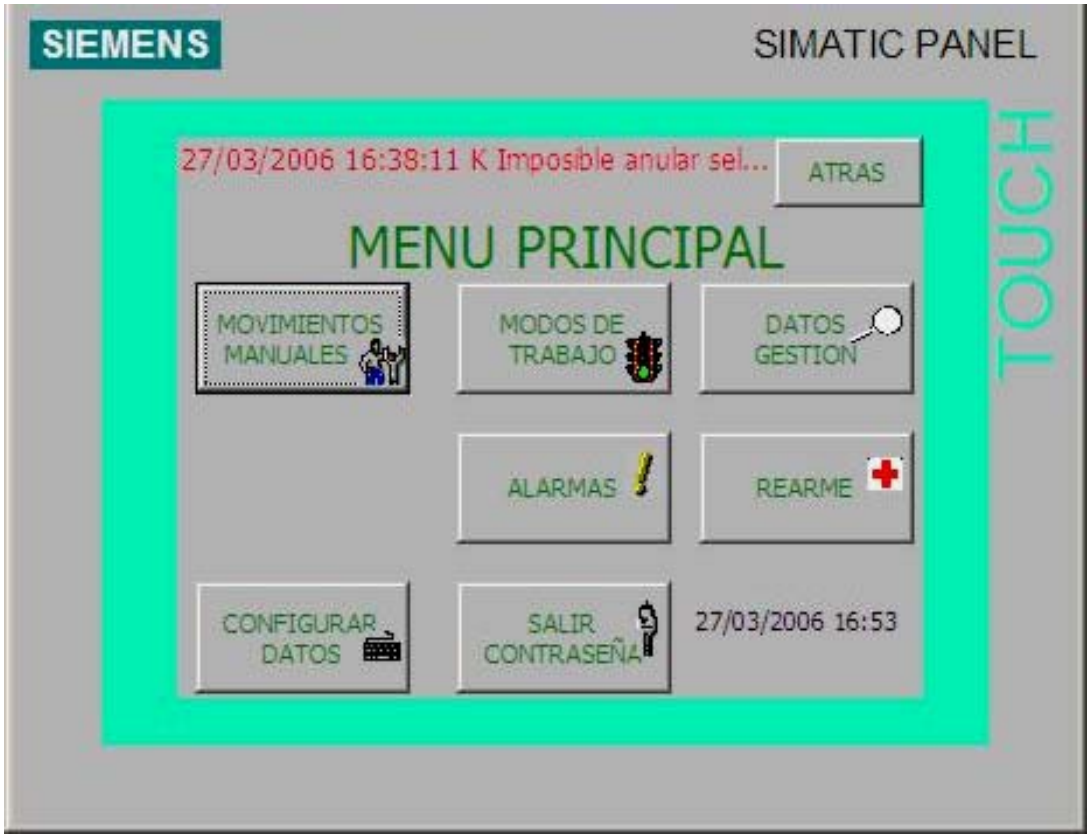


Figura 3.9. Pantalla de Menú Principal

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.

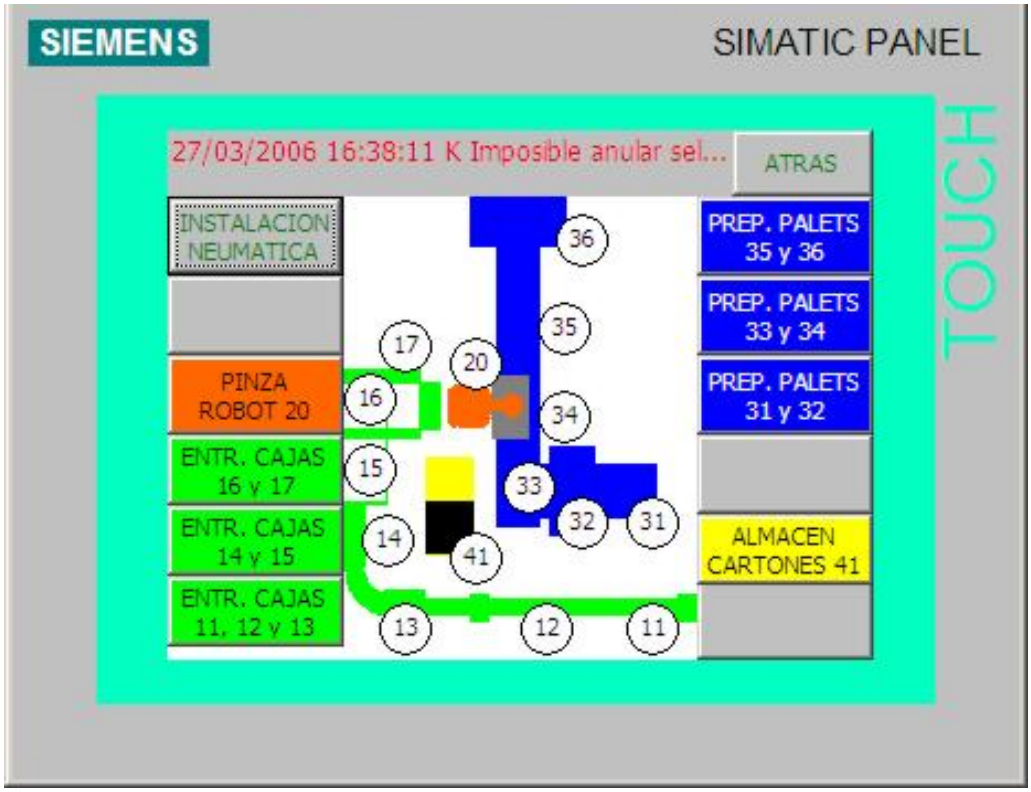


Figura 3.10. Pantalla de Zonas de Movimientos Manuales

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".

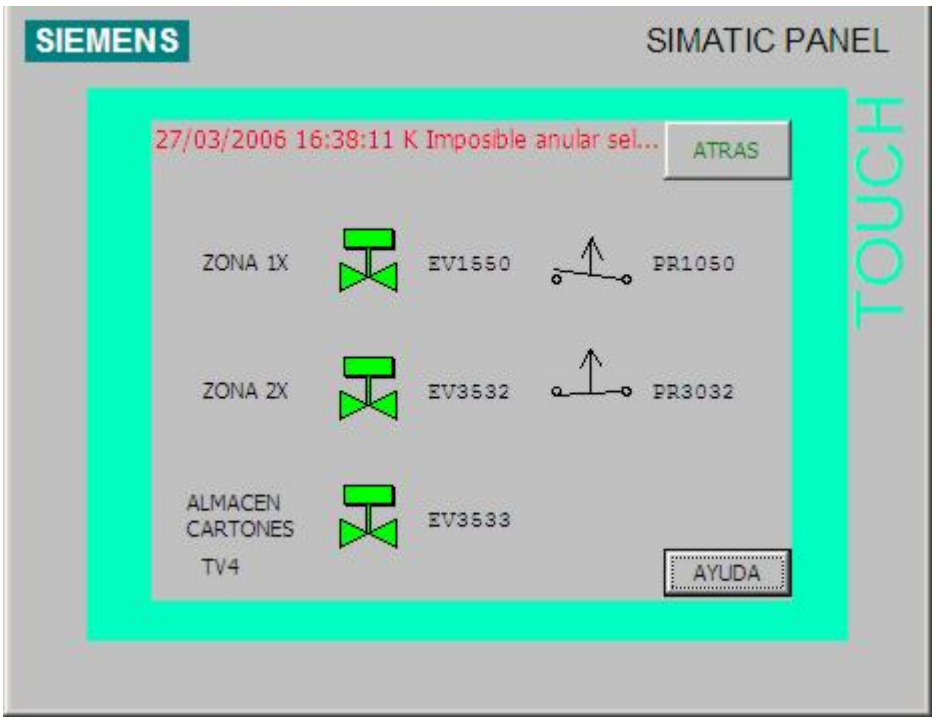


Figura 3.11. Instalación Neumática

Pinza Robot 20

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

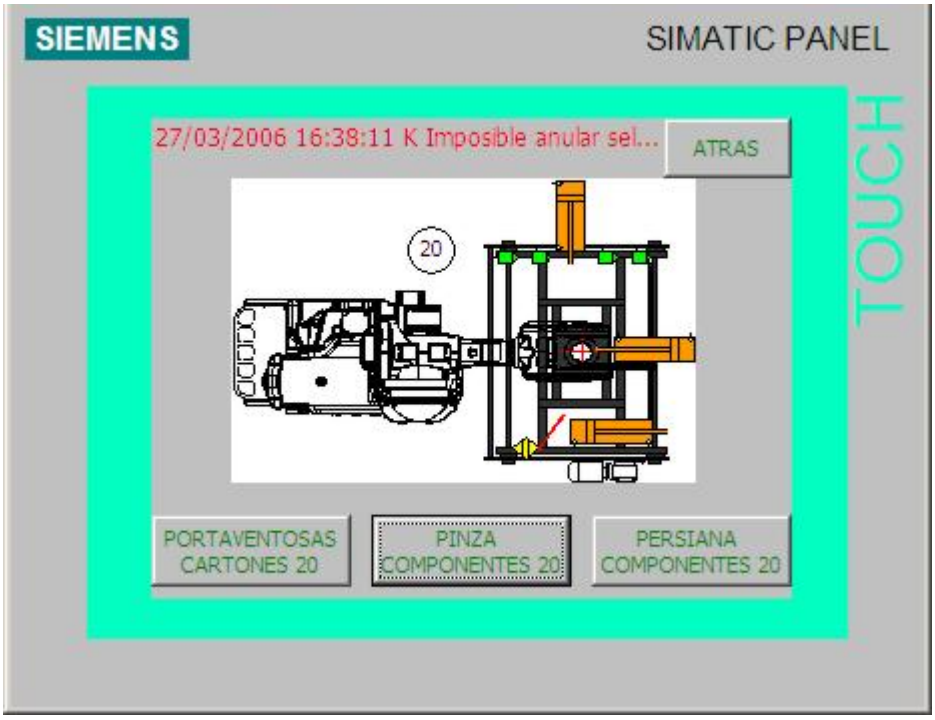


Figura 3.12. Pinza Robot

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".

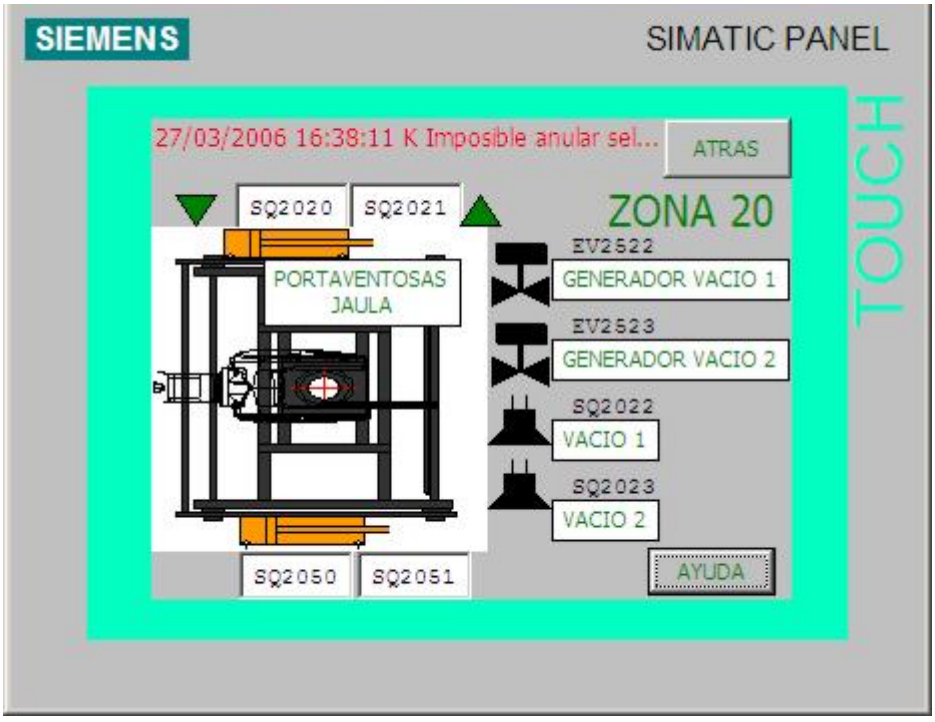


Figura 3.13. Portaventosas Cartones

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.

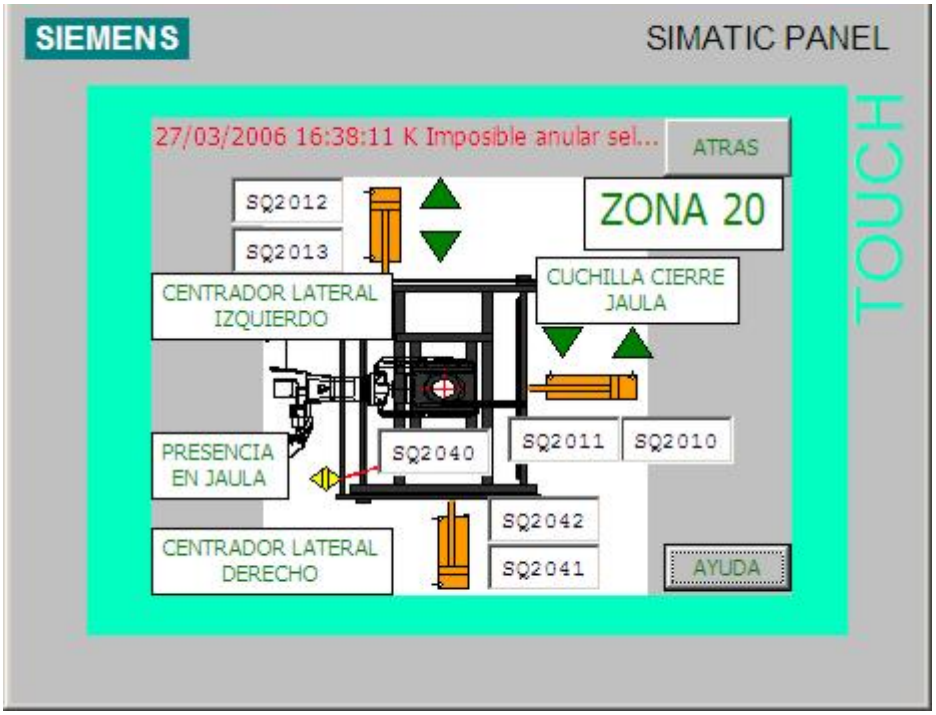


Figura 3.14. Pinza Componentes

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.

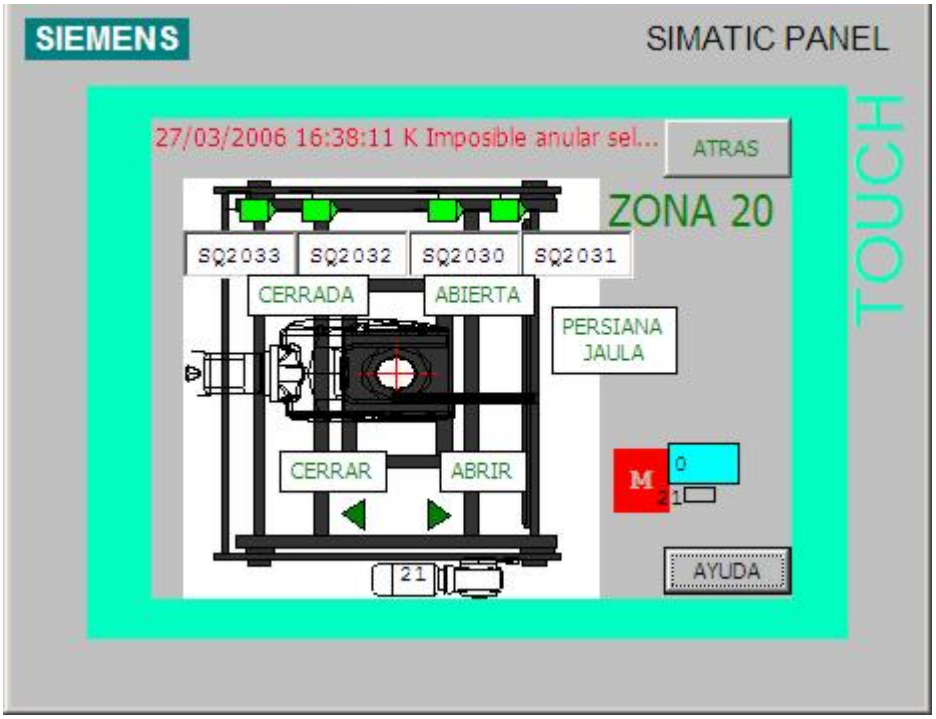


Figura 3.15. Persiana Componentes

Entrada Cajas 16 y 17

Acciones manuales sobre la mesa de formación.

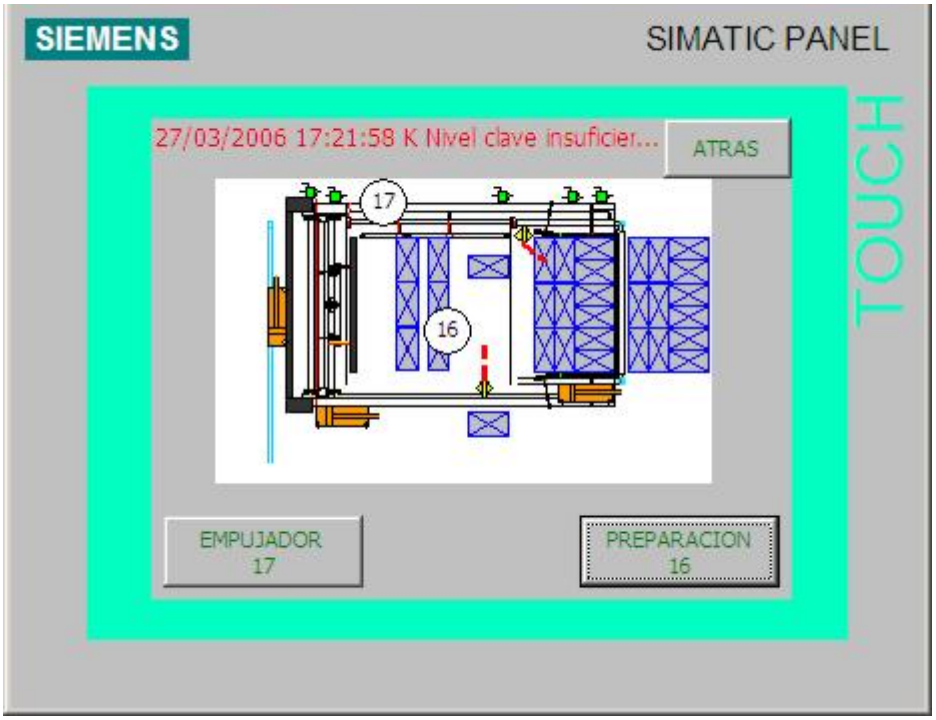


Figura 3.16. Entrada Cajas 16 y 17

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.

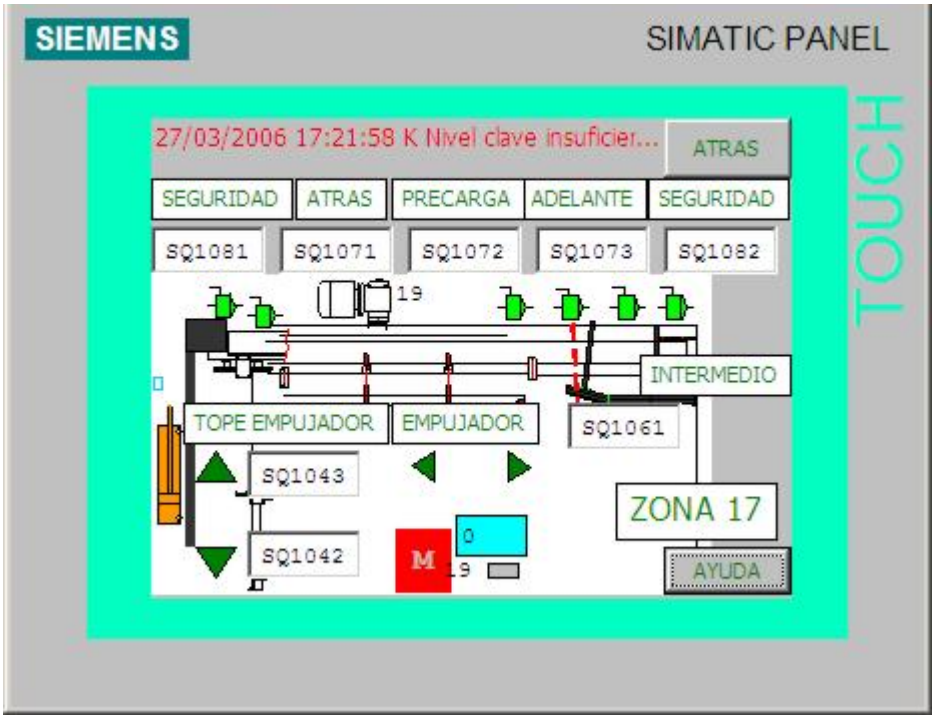


Figura 3.17. Empujador

Preparación

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

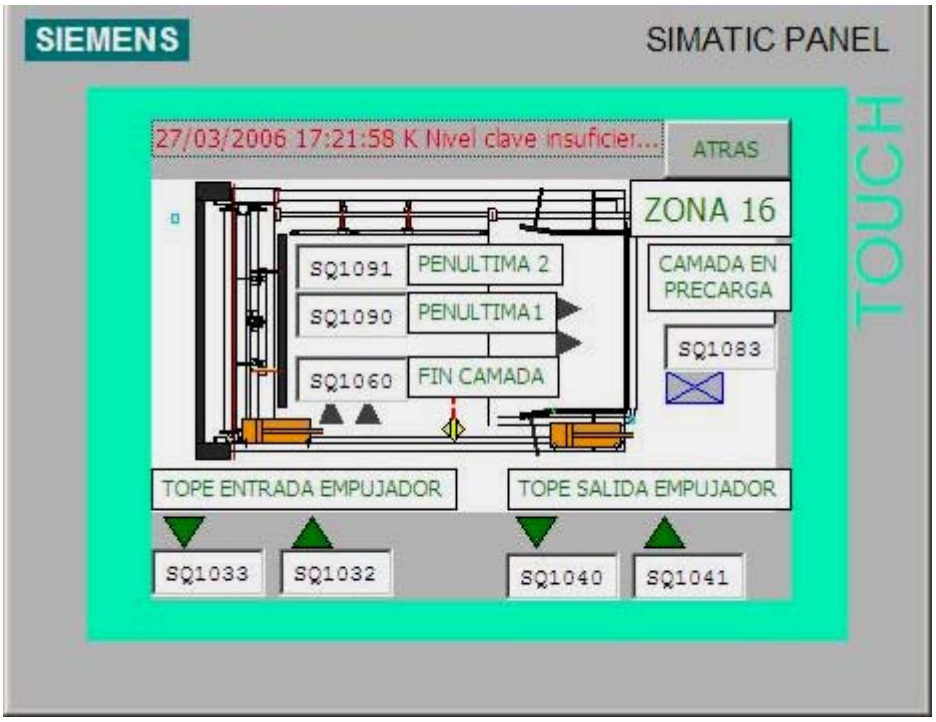


Figura 3.18. Preparación

Entrada Cajas 14 y 15

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

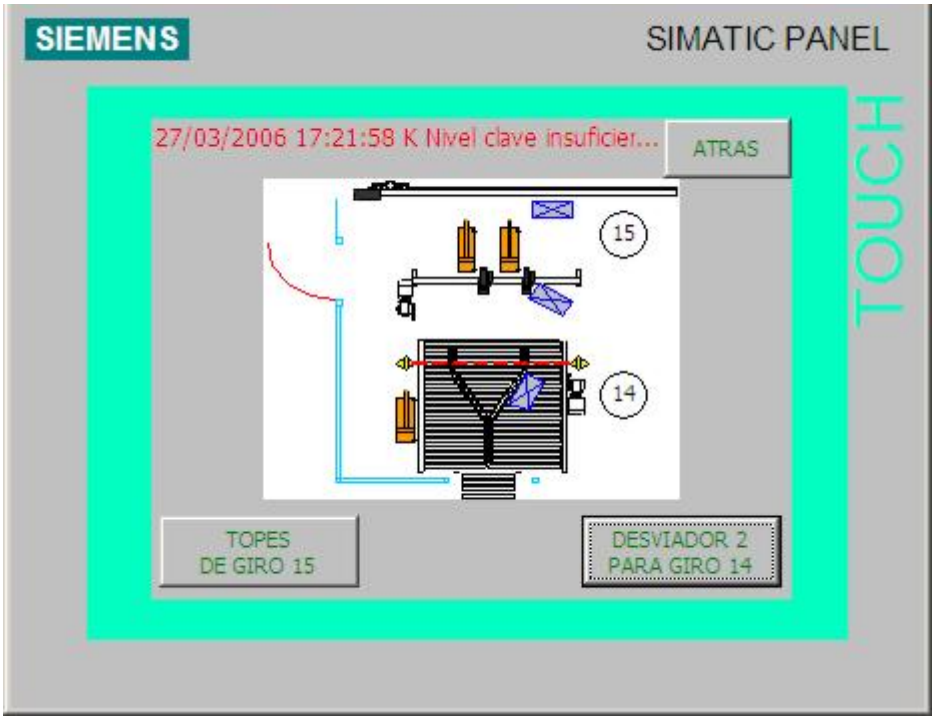


Figura 3.19. Entrada Cajas 14 y 15

Topes de Giro

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

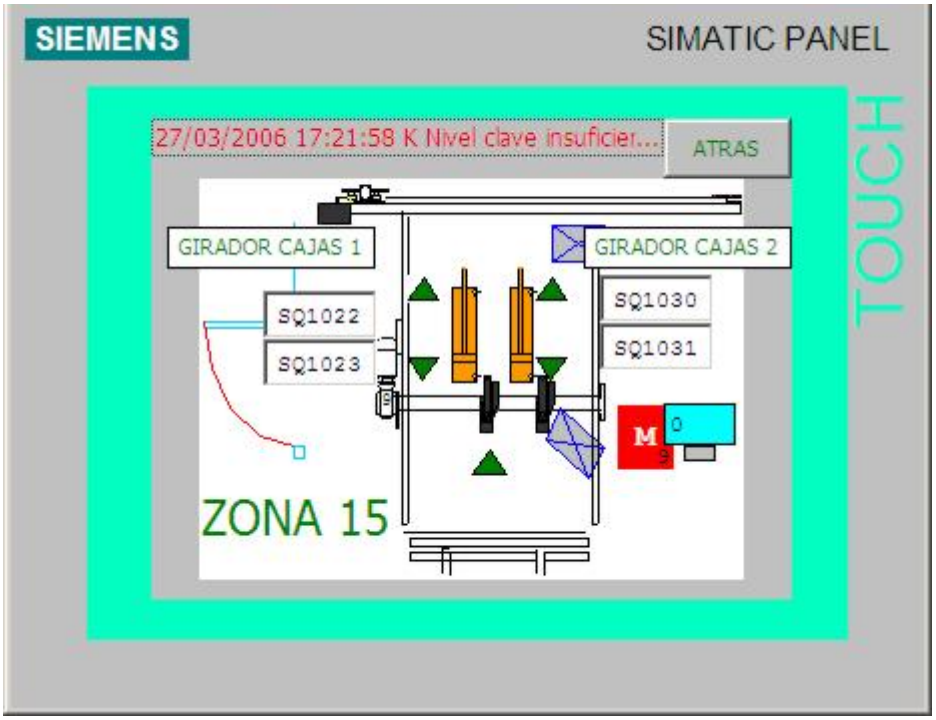


Figura 3.20. Topes de Giro

Desviador 2

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

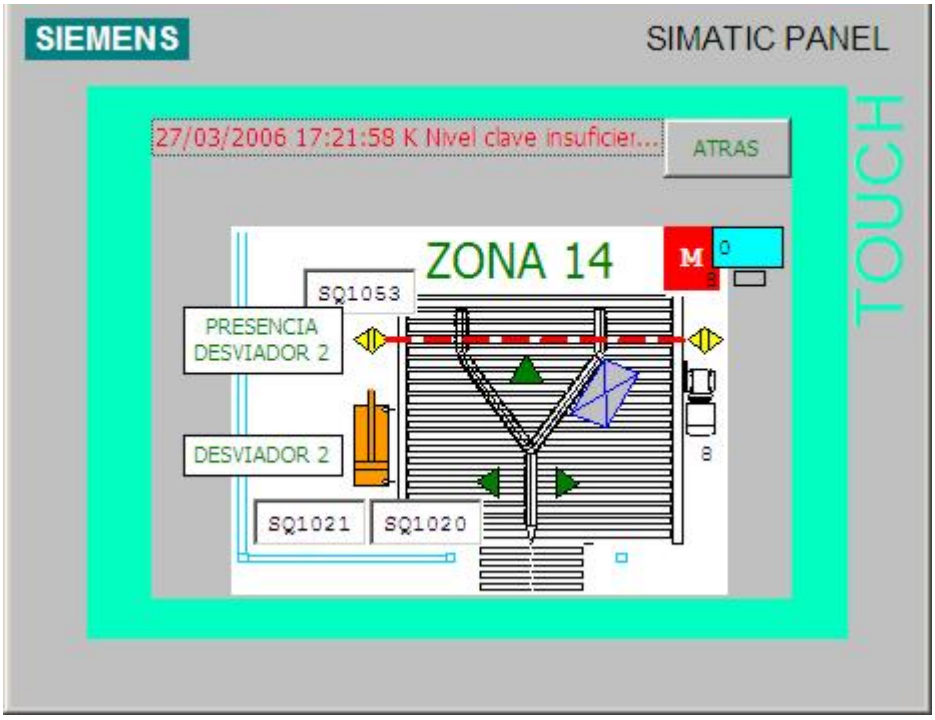


Figura 3.21. Desviador 2

Entrada Cajas 11,12 y 13

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

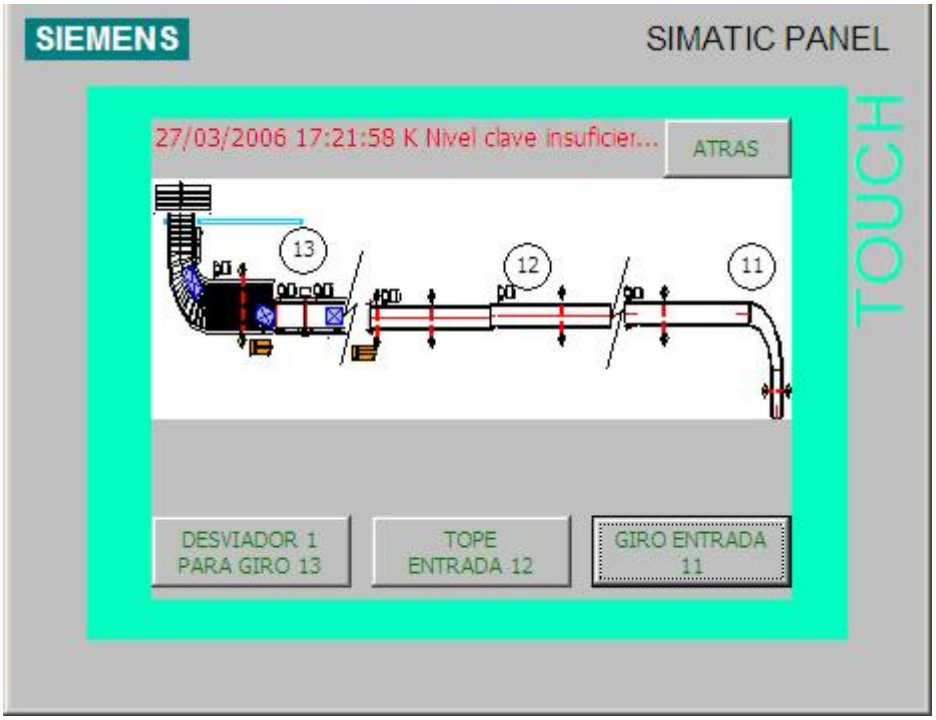


Figura 3.22. Entrada Cajas 11, 12 y 13

Desviador 1

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

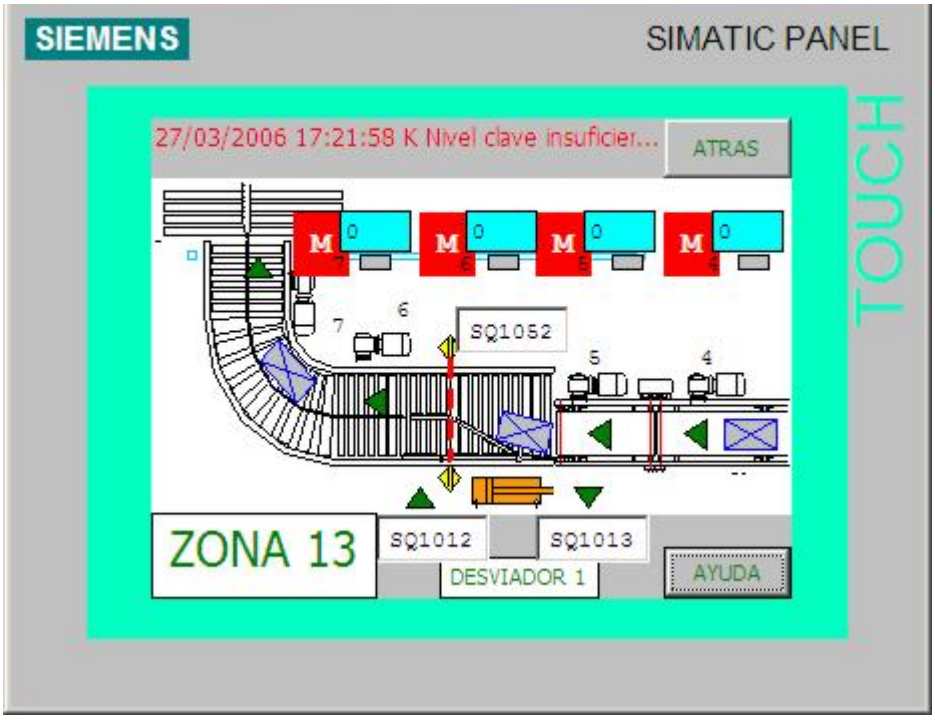


Figura 3.23. Desviador 1

Tope Entrada

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

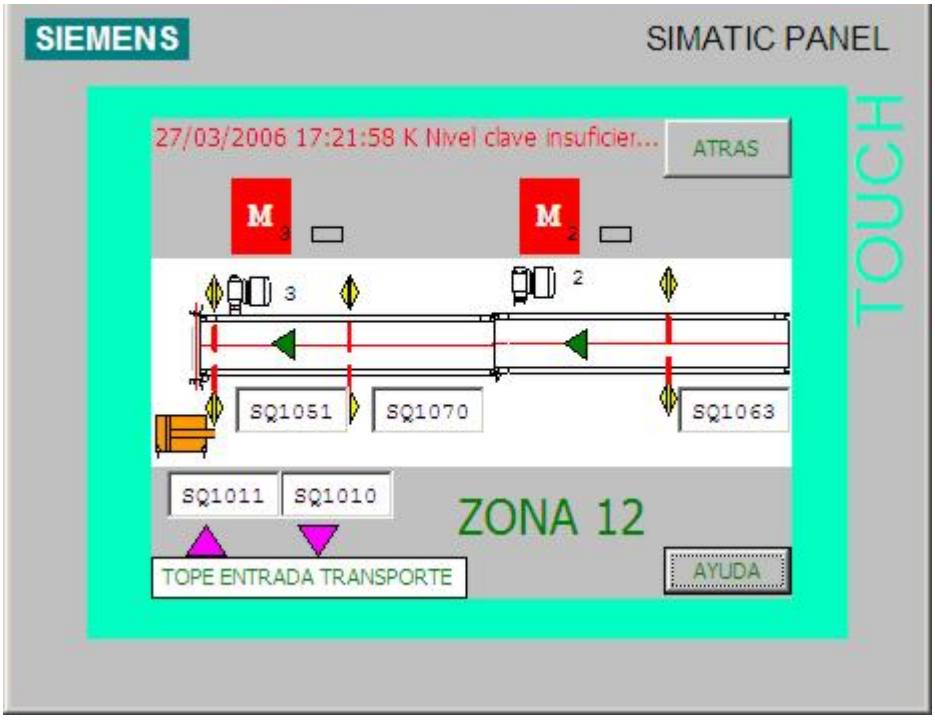


Figura 3.24. Tope entrada

Giro Entrada

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

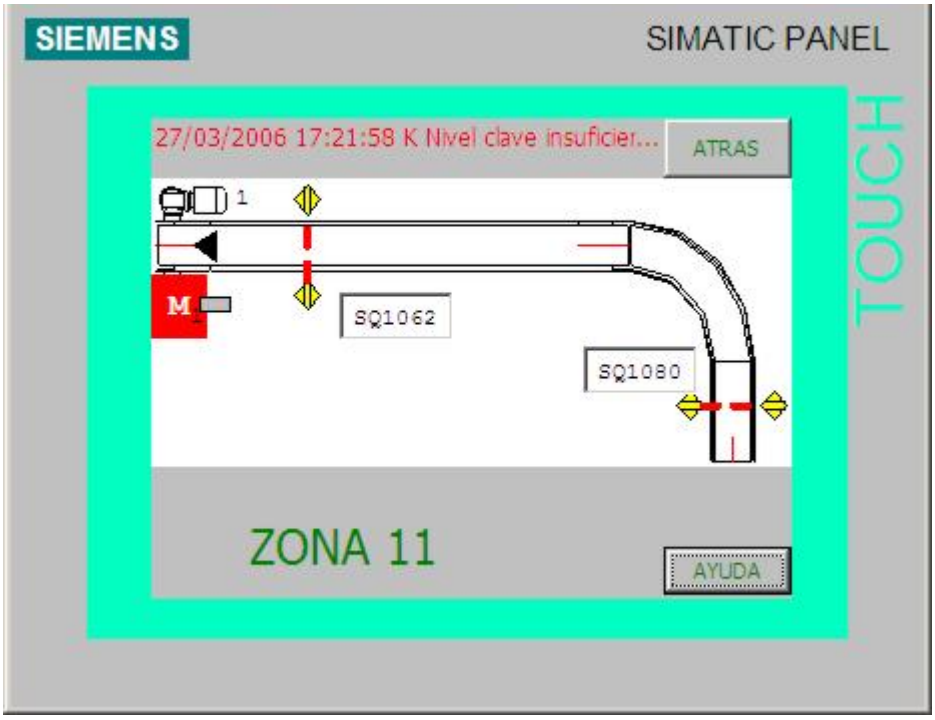


Figura 3.25. Giro Entrada

Preparación Palets 35 y 36

Acciones sobre la segunda transferencia de palets.

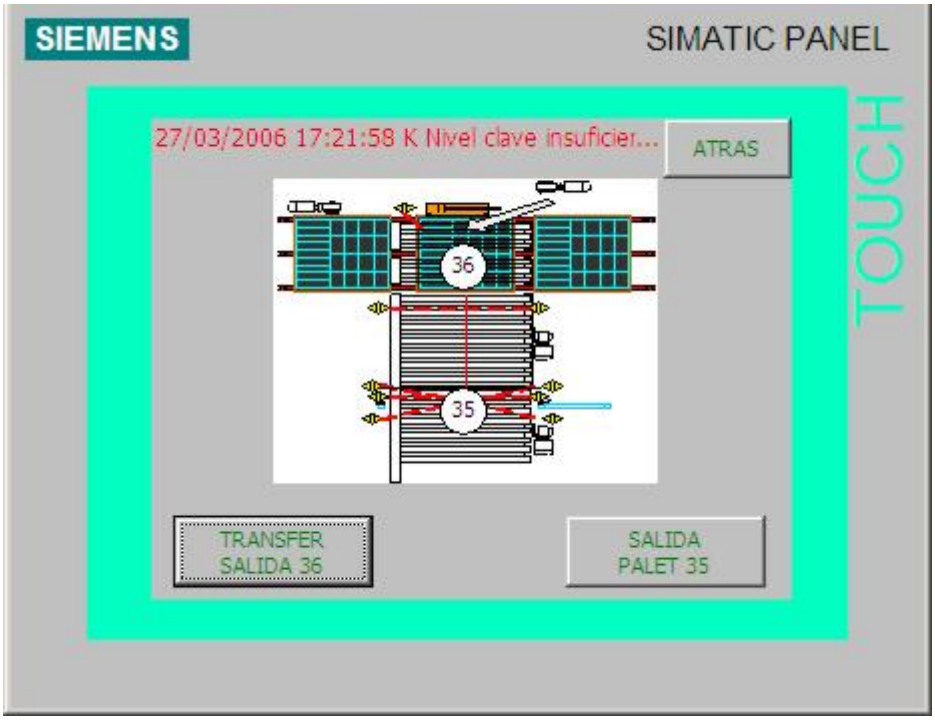


Figura 3.26. Preparación Palets 35 y 36

Transferencia de salida

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

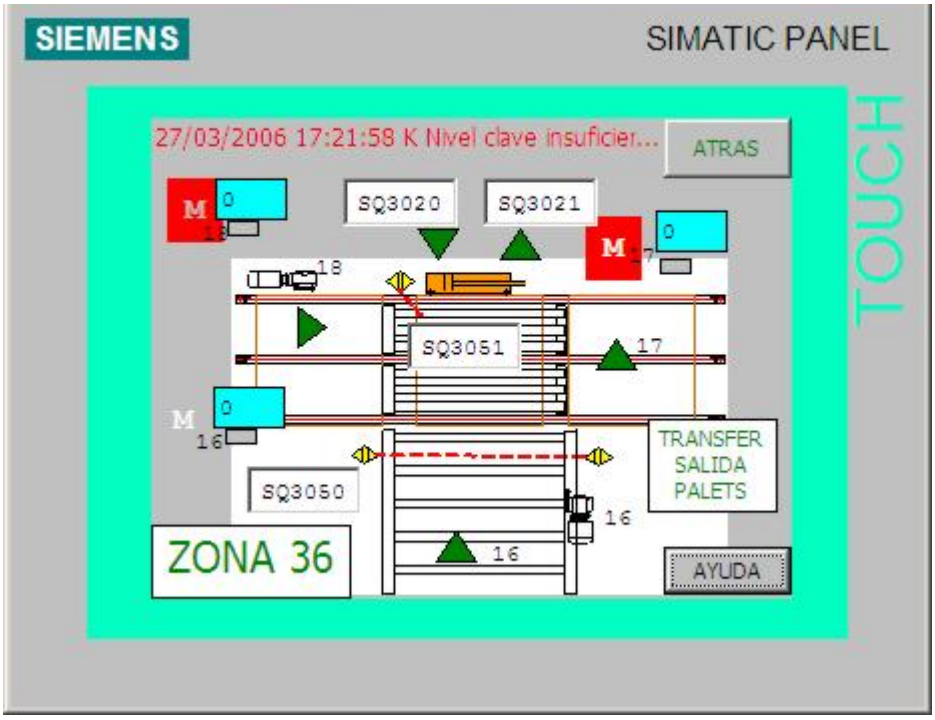


Figura 3.27. Transferencia de Salida

Salida de palet

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

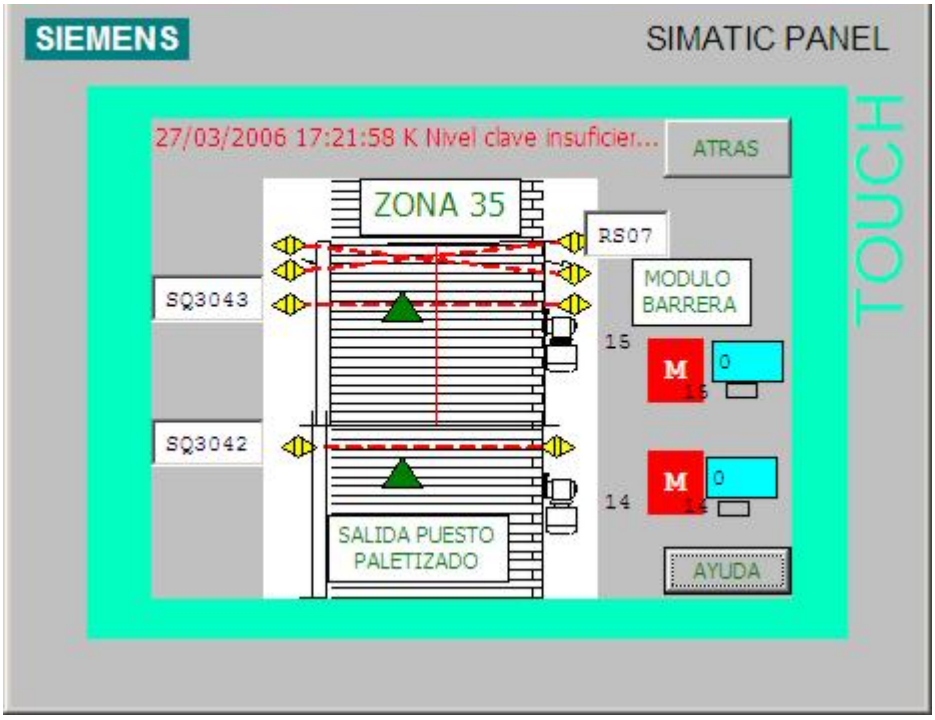


Figura 3.28. Salida de Palet

Preparación Palets 33 y 34

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

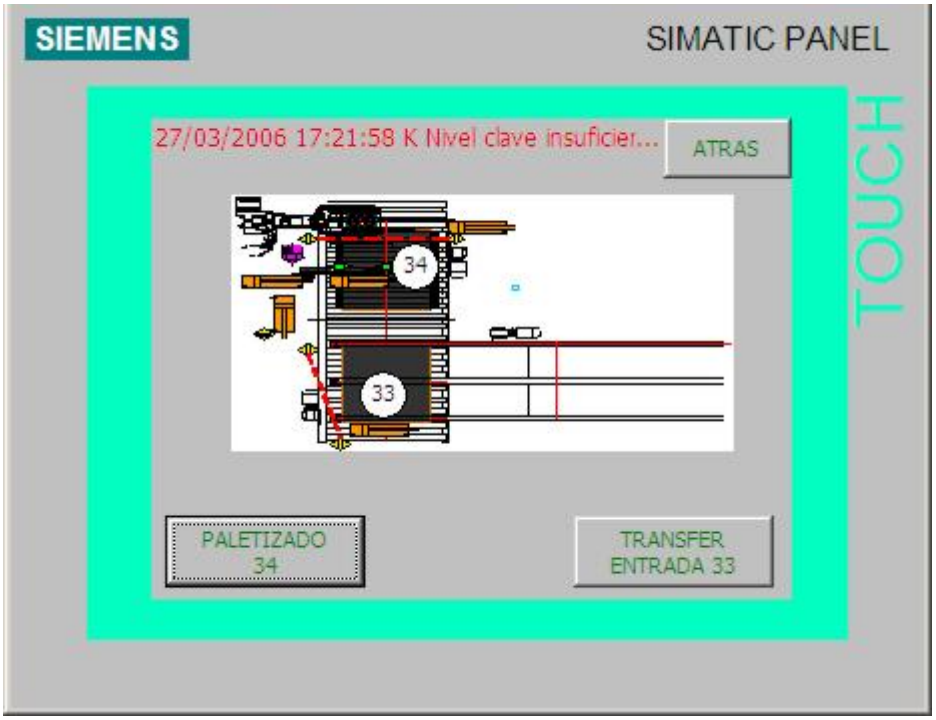


Figura 3.29. Preparación Palets 33 y 34

Paletizado

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

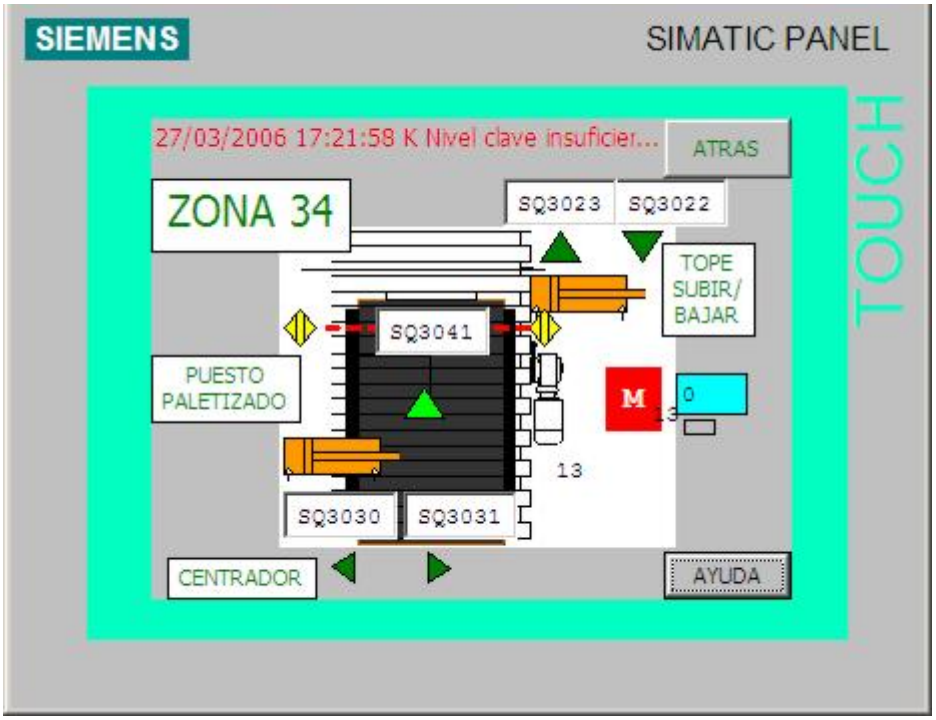


Figura 3.30. Paletizado

Transferencia de entrada

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

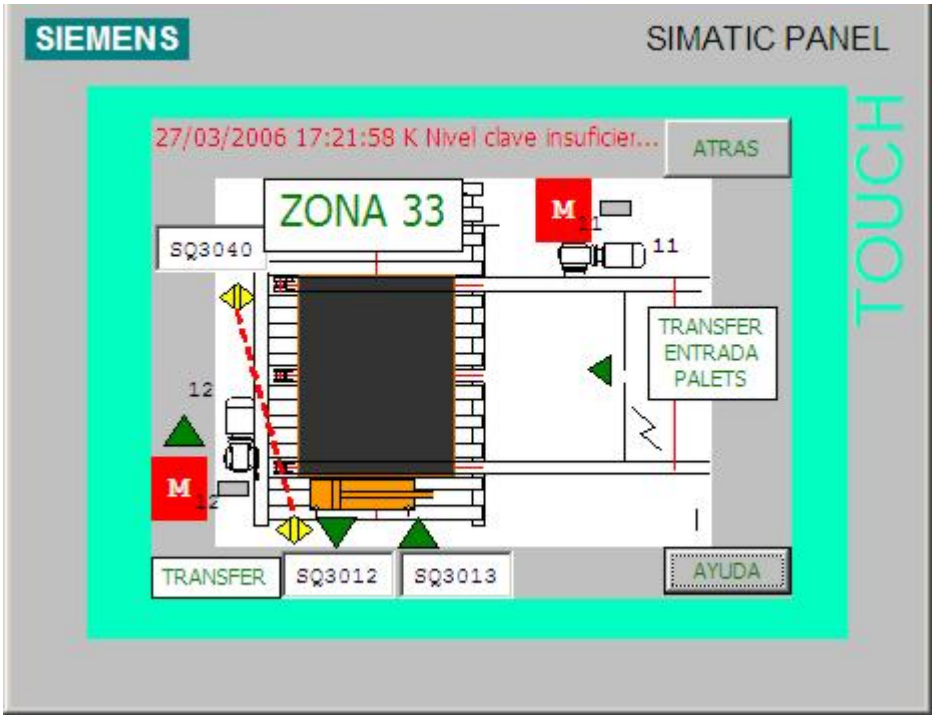


Figura 3.31. Transferencia de entrada

Preparación Palets 31 y 32

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

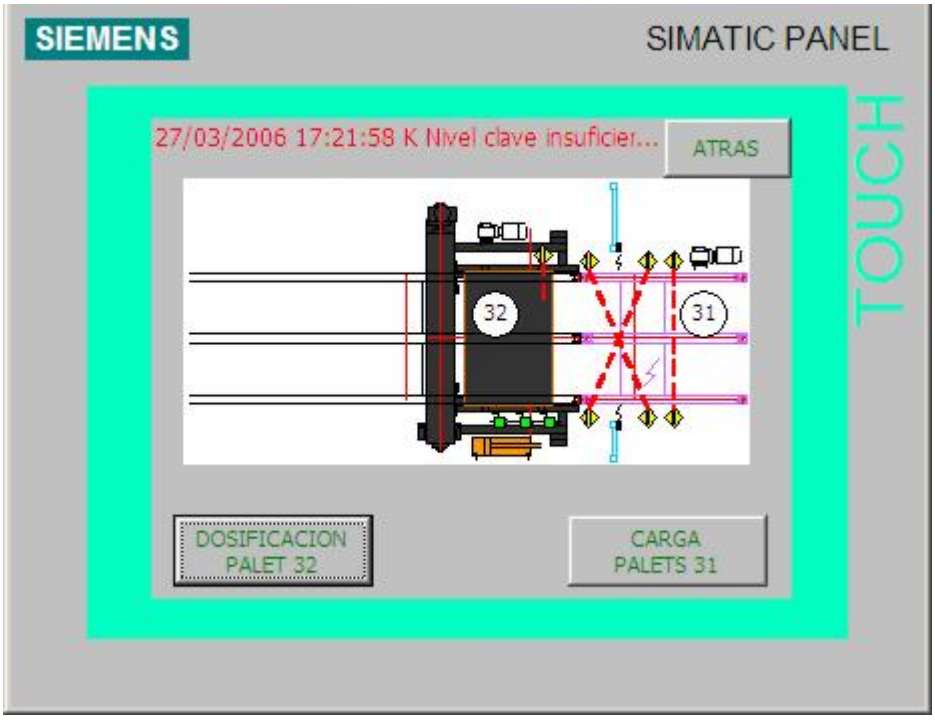


Figura 3.32. Preparación Palets 31 y 32

Dosificación palet

Acciones manuales sobre los elementos del almacén de palets.

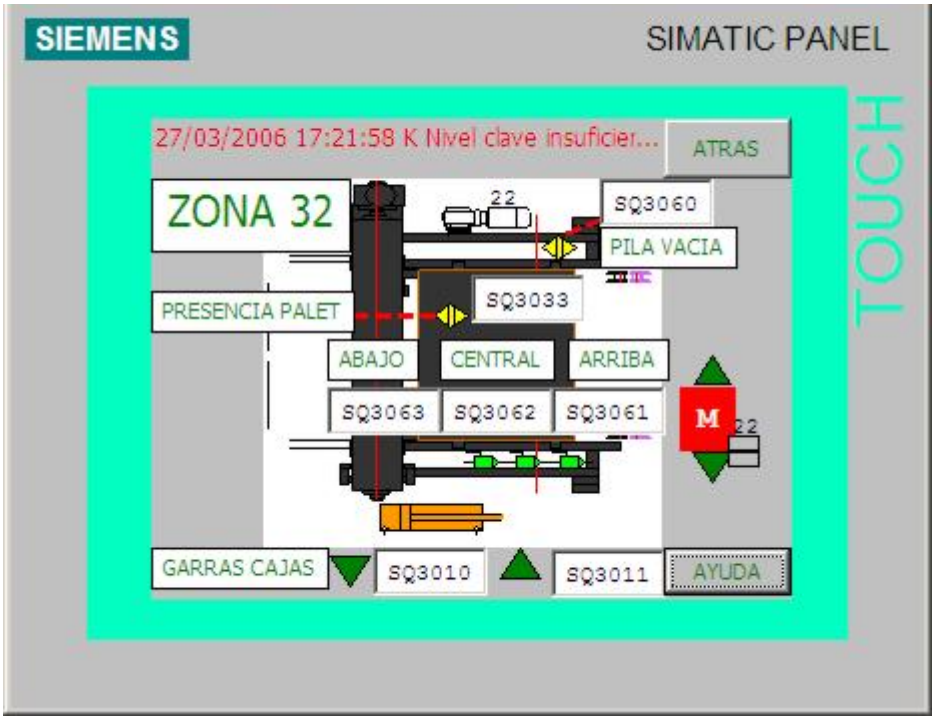


Figura 3.33. Dosificación de Palets

Carga de palets

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

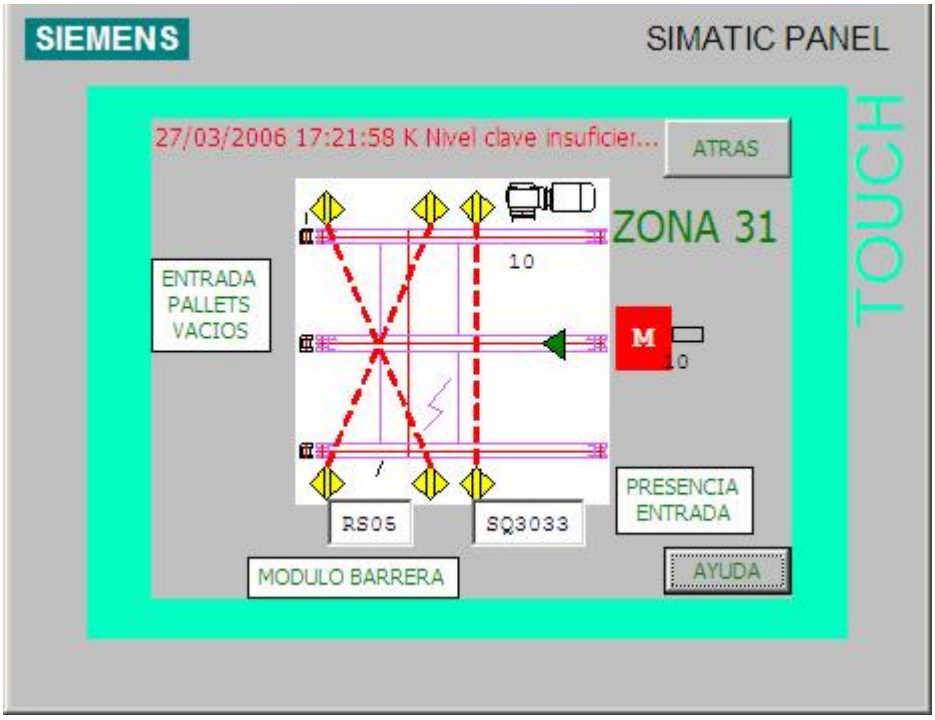


Figura 3.34. Carga de Palets

Almacén de Cartones

Acciones manuales sobre el almacén de cartones.

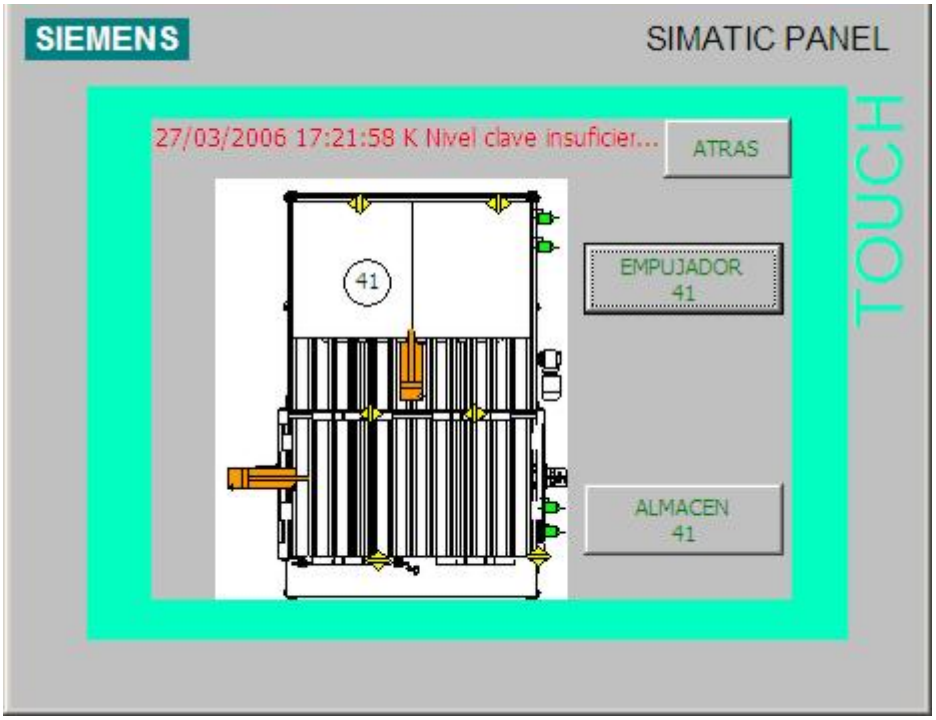


Figura 3.35. Almacén Cartones

Cabezal del almacén de cartones

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

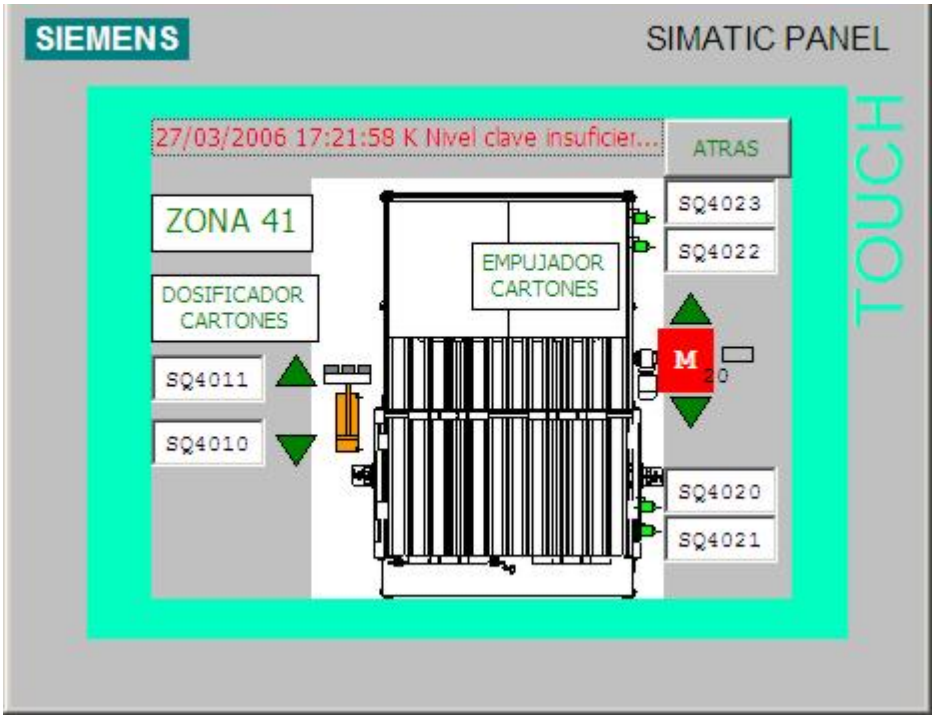


Figura 3.36. Empujador de Cartones

Centrador del almacén de cartones

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

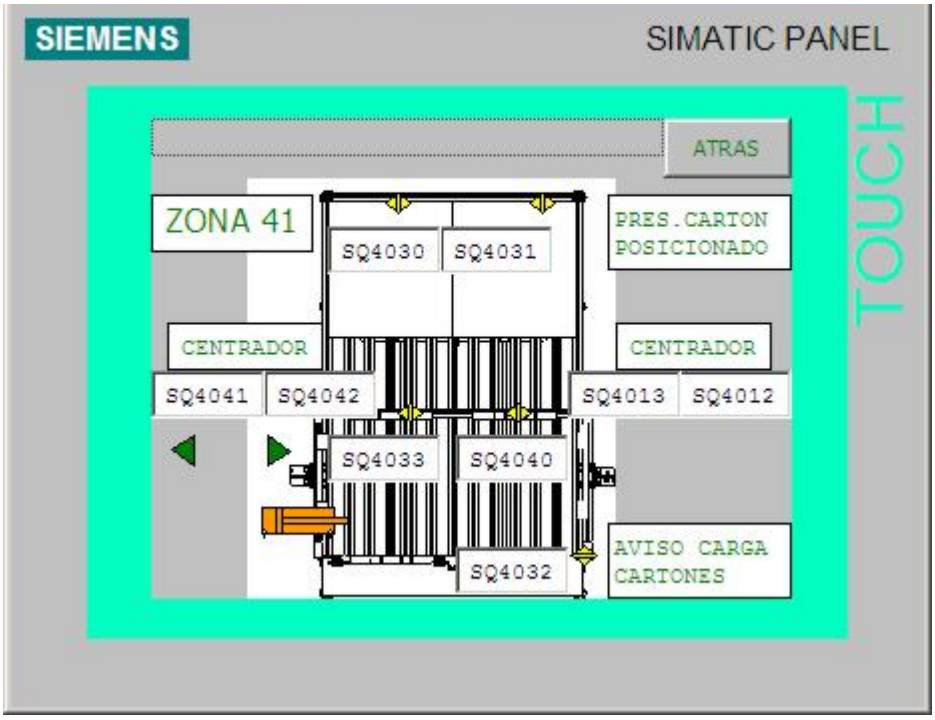


Figura 3.37. Centrador del almacén de cartones

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 **Figura 2.4. Zonas de la planta de paletizado**) 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

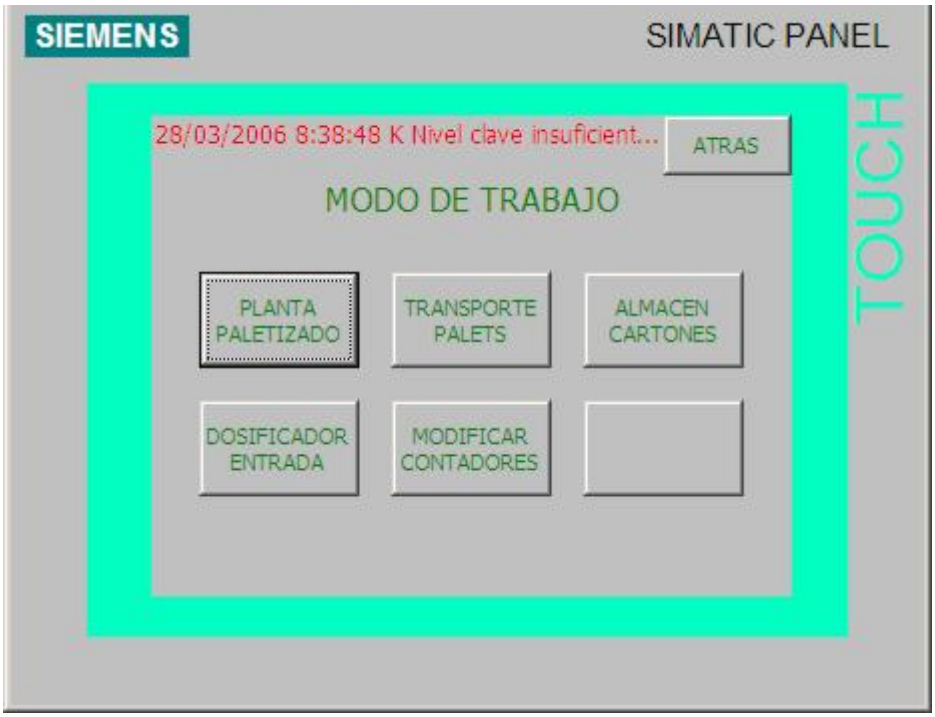


Figura 3.38. Modos de Trabajo

Planta de Paletizado

Cada una de las secciones consta de:

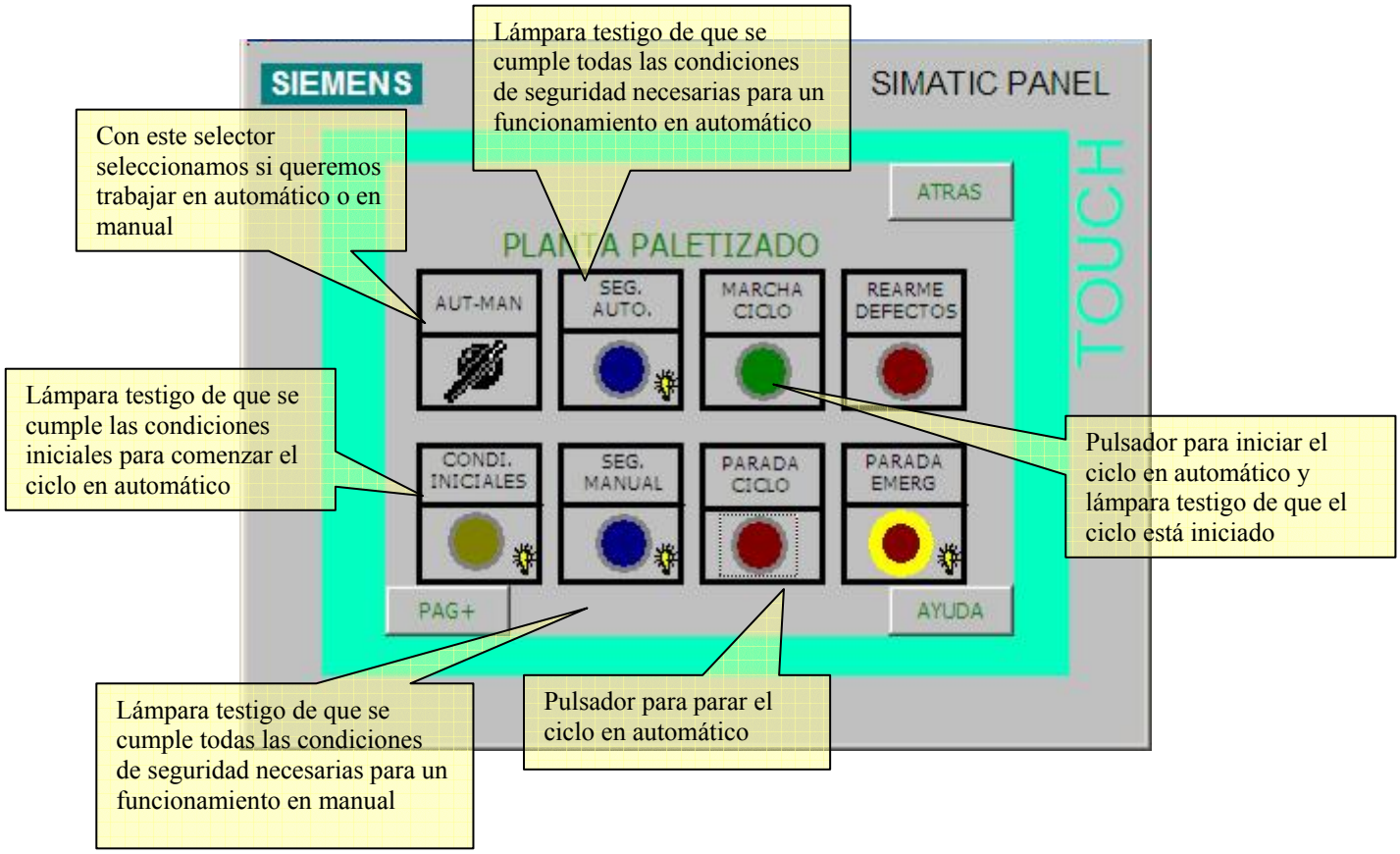


Figura 3.39. Planta de Paletizado

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.

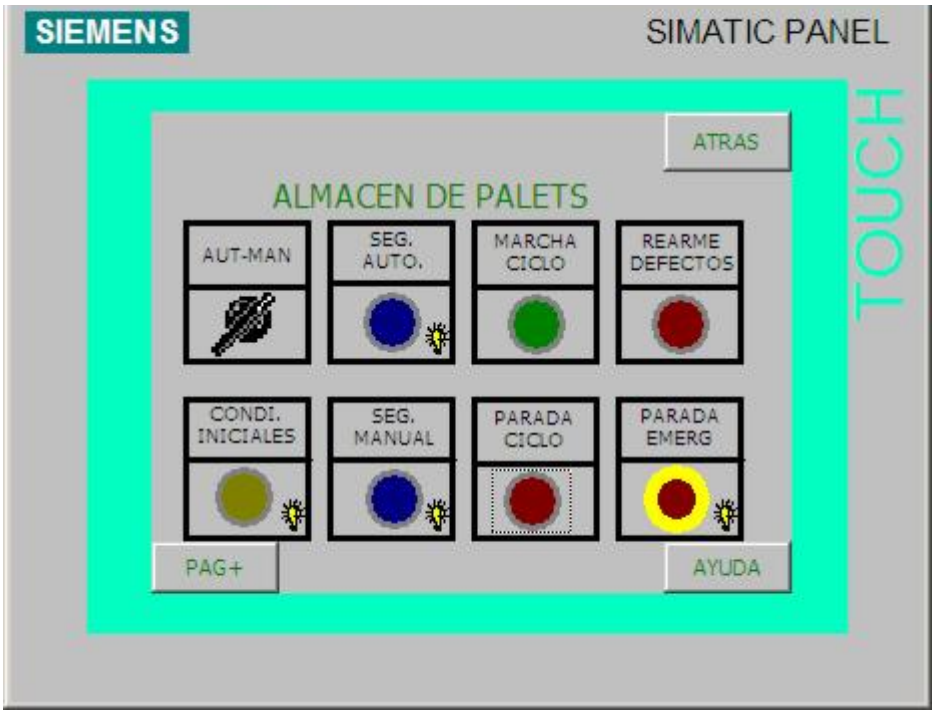


Figura 3.40. Transporte de palets

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”.

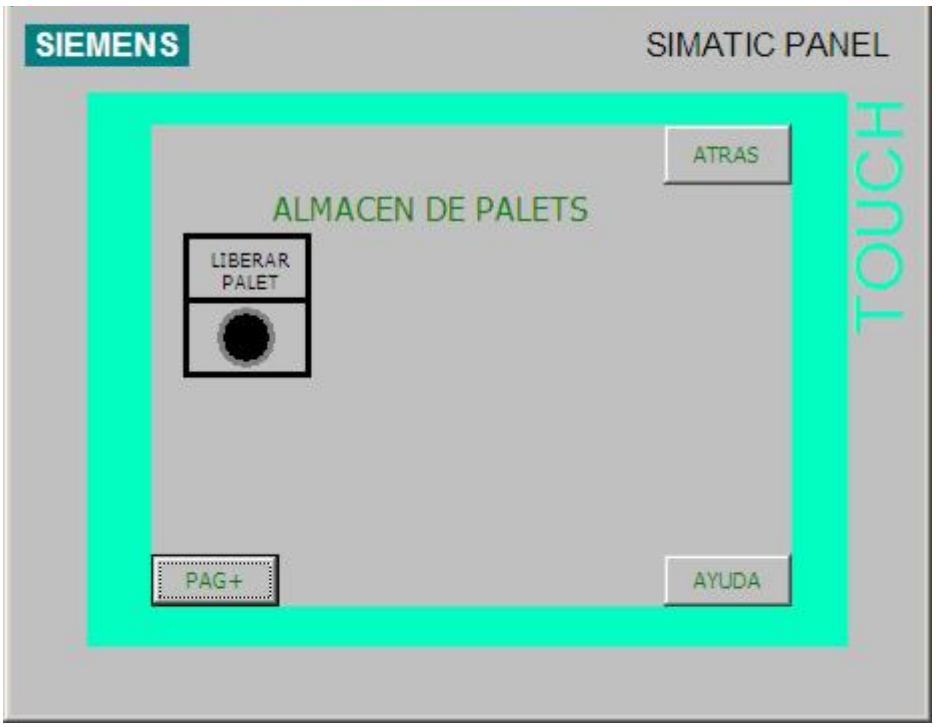


Figura 3.41. Liberación de Palet

Almacén de Cartones

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

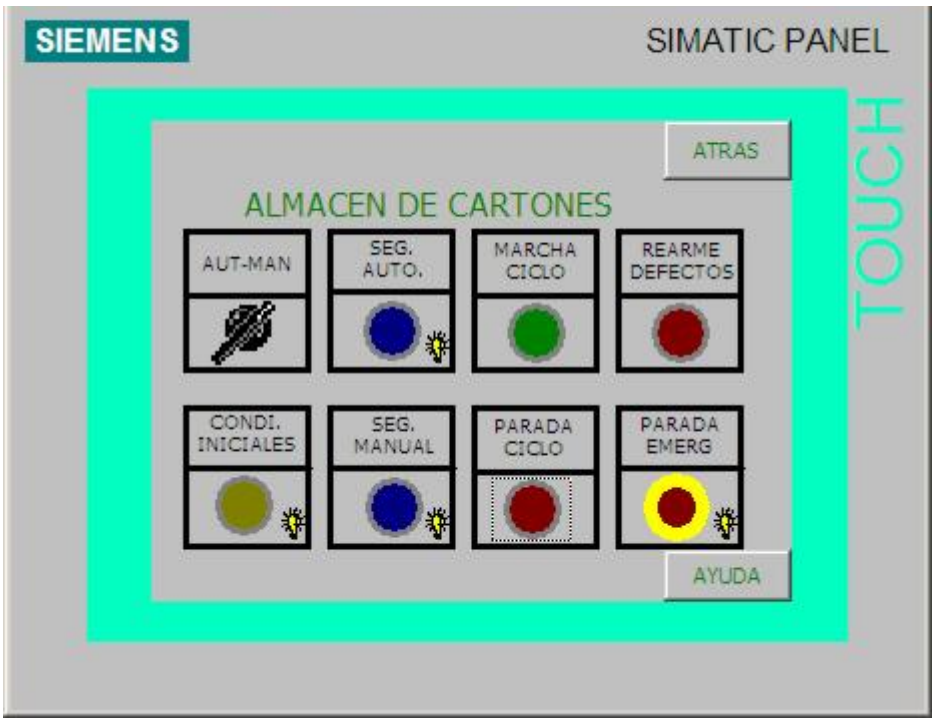


Figura 3.42. Almacén de Cartones

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.

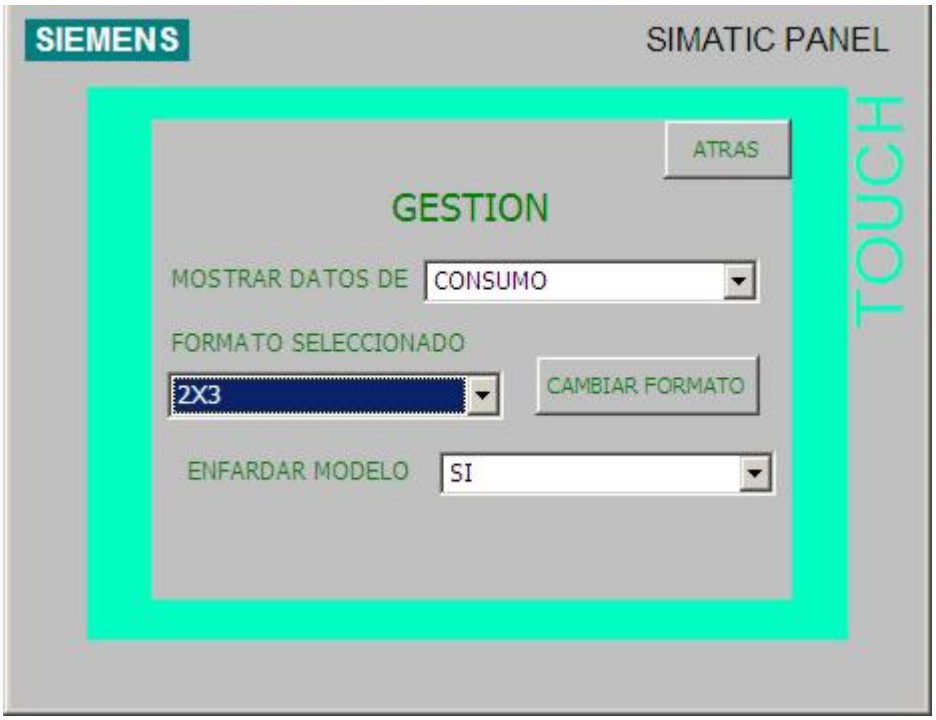


Figura 3.46. Datos de Gestión

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.

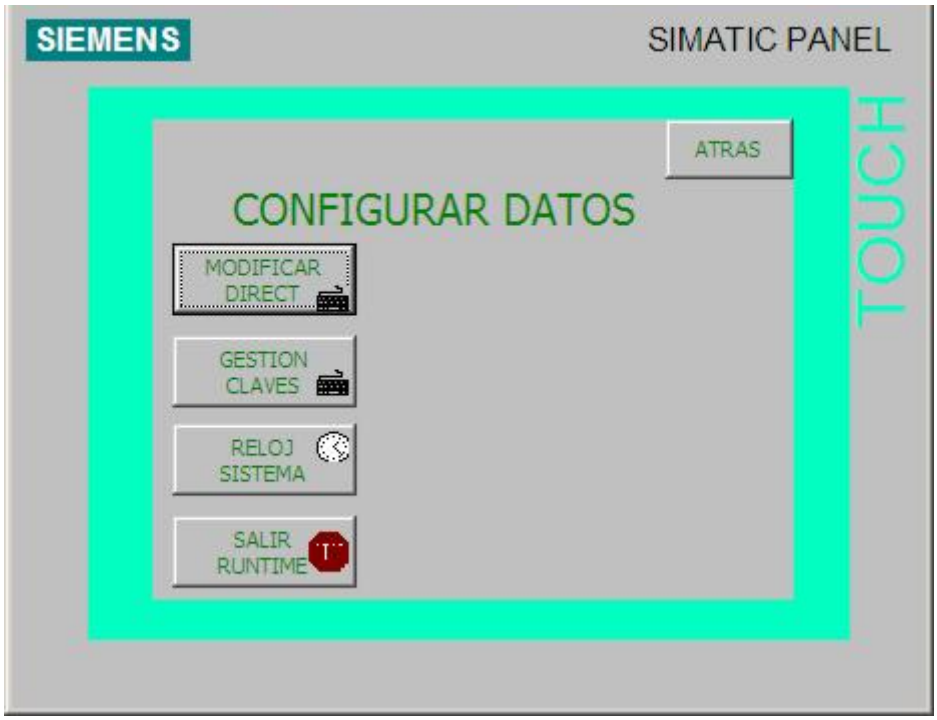


Figura 3.49. Configurar Datos

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.

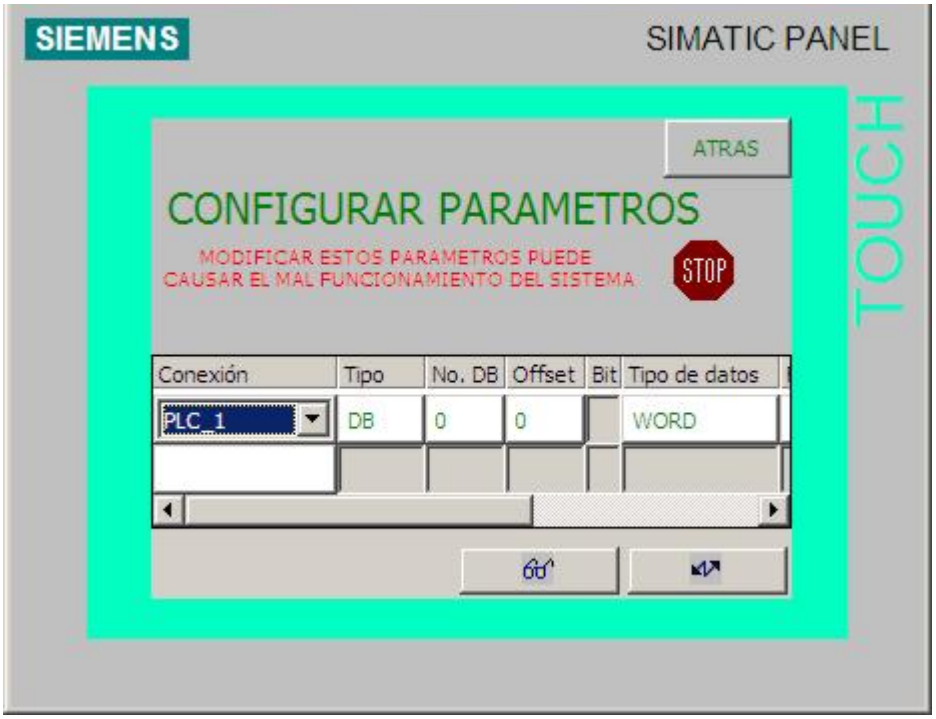


Figura 3.50. Modificación Directa

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.

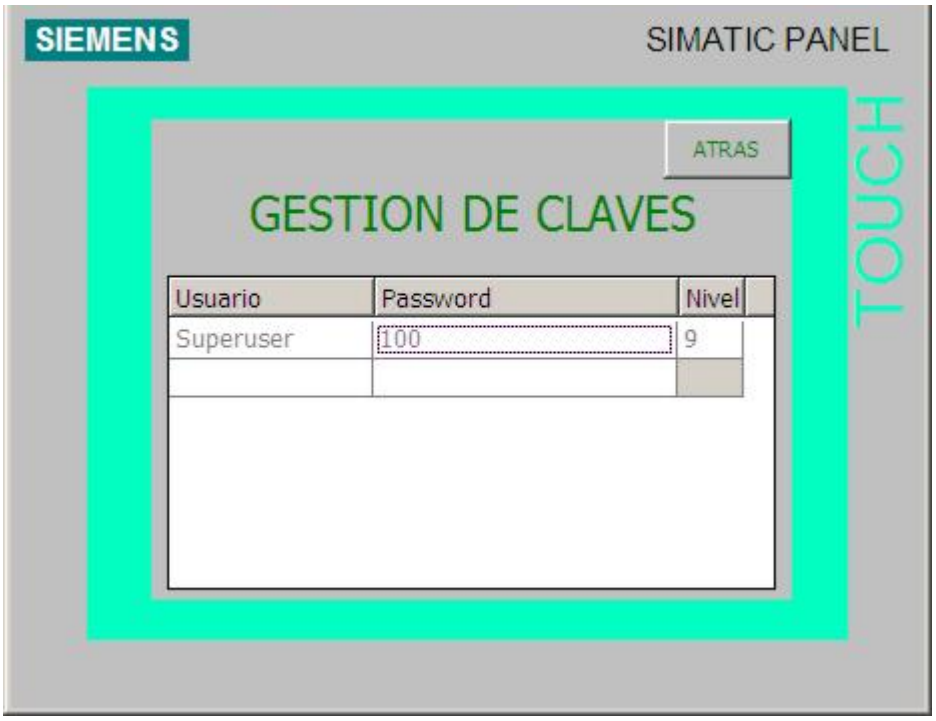


Figura 3.51. Gestión de claves

Ajustar fecha y hora

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

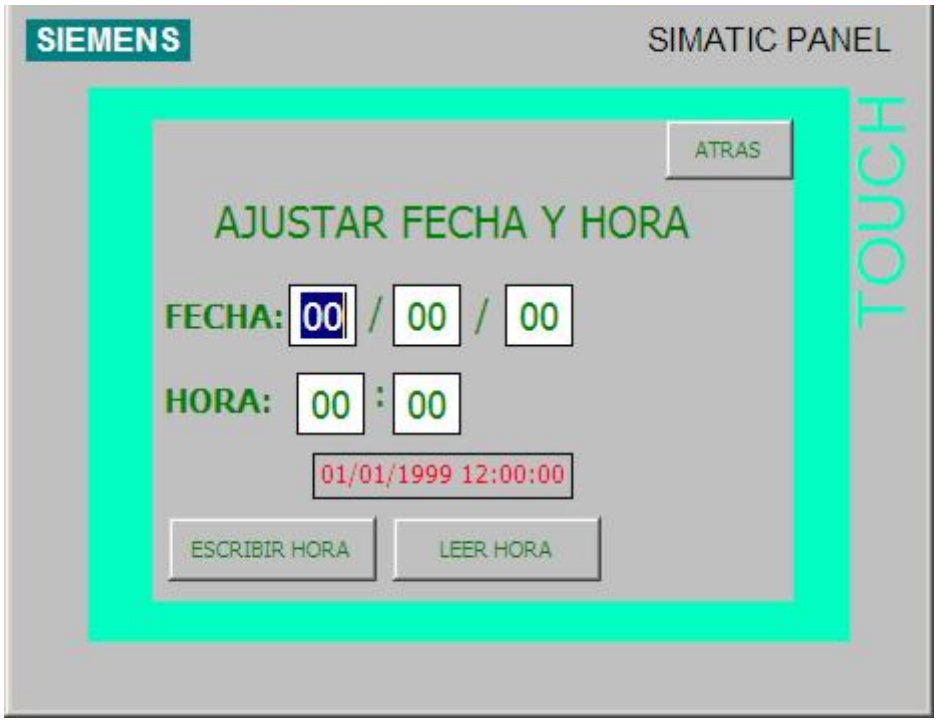


Figura 3.52. Ajuste de Fecha y Hora

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.

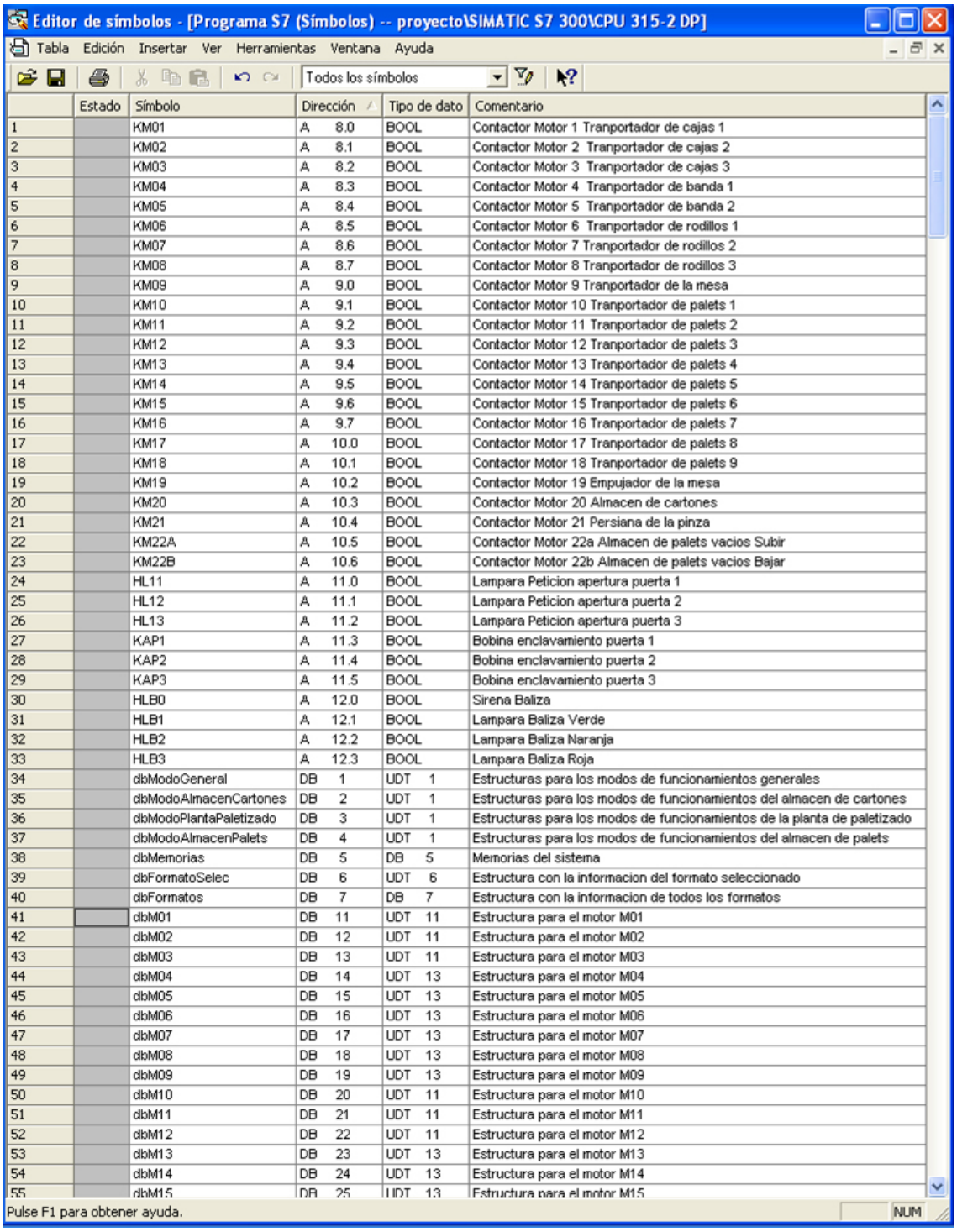
- Al crear sus programas, el programador dispone de cómodas funciones para editar el texto fuente.
- En el proceso de compilación se generan bloques a partir del programa editado que pueden ejecutarse en todas las CPU del sistema de automatización S7-300/400 a partir de la CPU 314.
- Las funciones de 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:

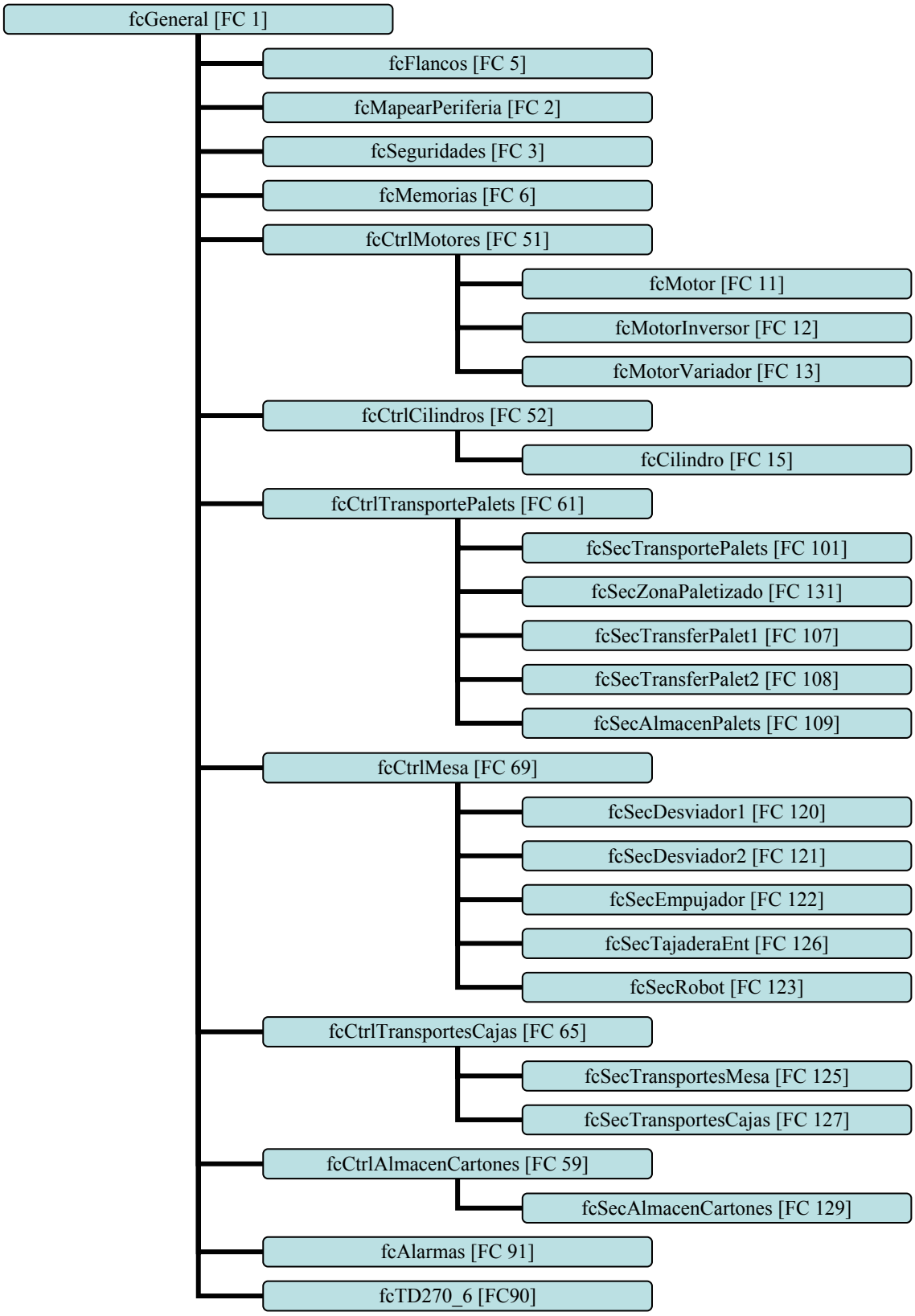


Figura 3.54. Llamadas del fcGeneral

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:

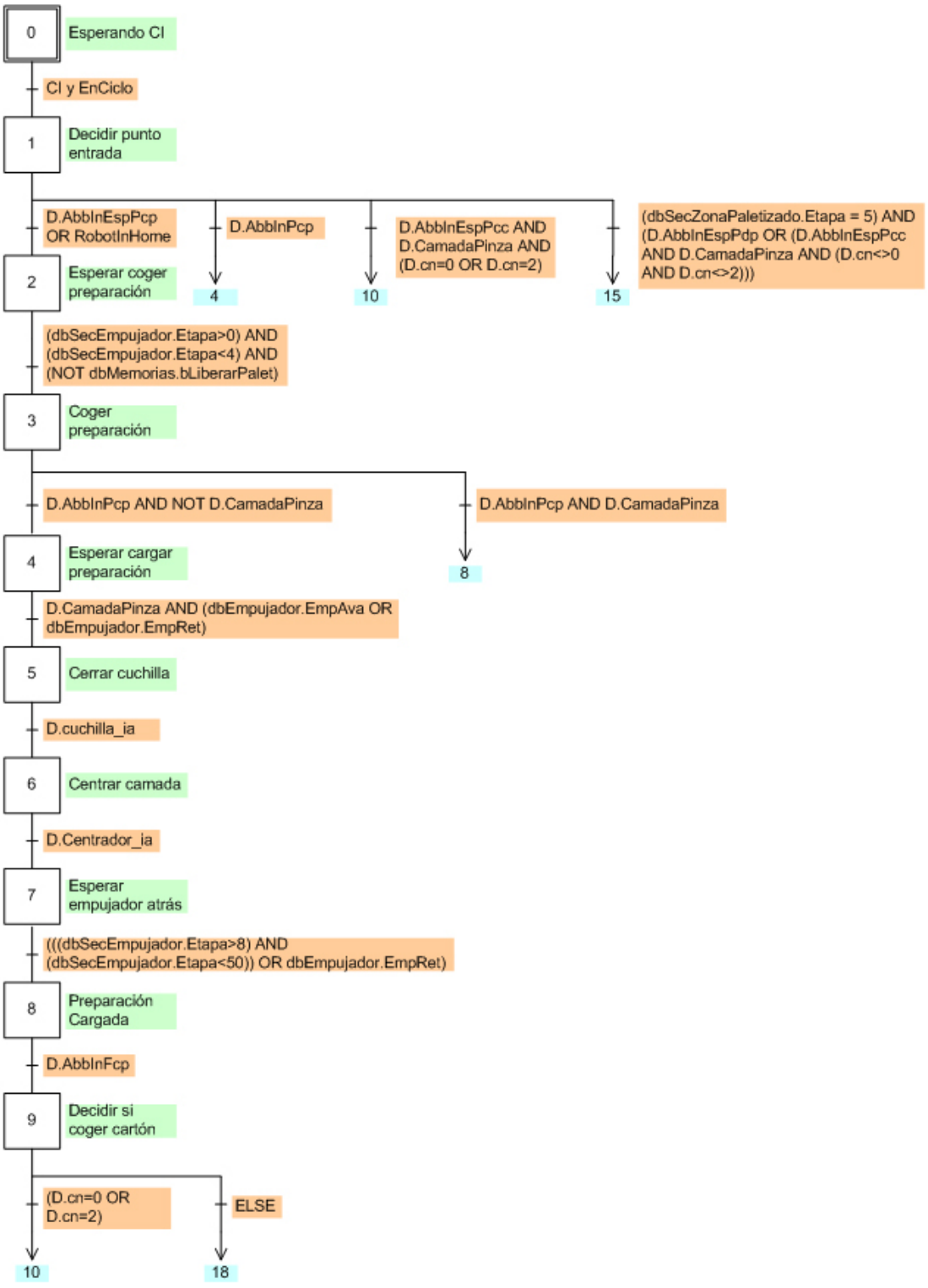


Figura 3.55. GRAFCET para secuencia 1

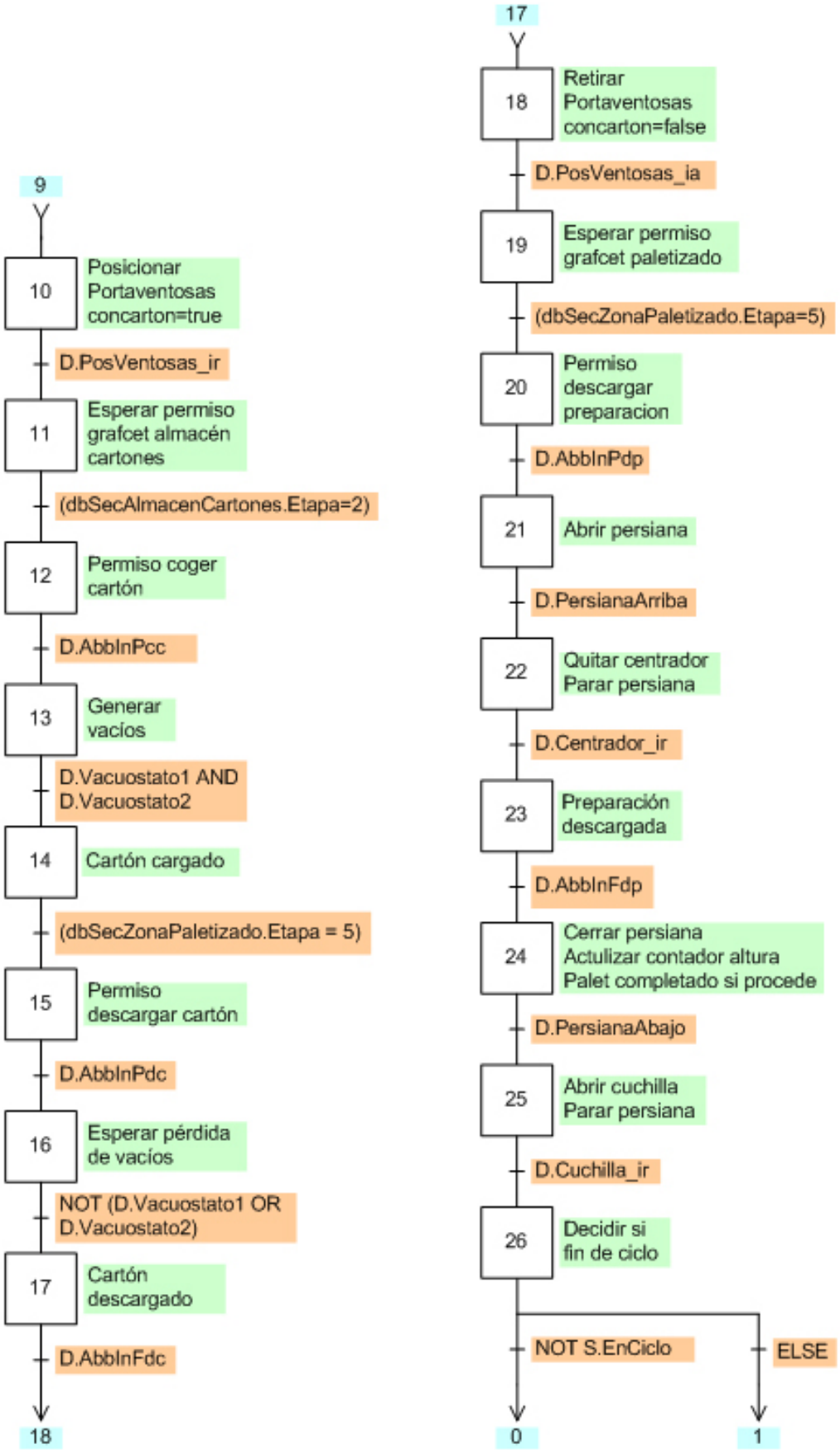


Figura 3.56. GRAFCET para secuencia 2

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.

.

CAPITULO 4

Conclusiones y propuestas de mejora

En este proyecto hemos conseguido poder programar con cierta eficiencia un autómata industrial bastante extendido, como es el que fabrica la empresa SIEMENS. Así mismo, también se ha podido configurar uno de los aspectos básicos que requiere siempre cualquier instalación robótica, como es el interfaz Hombre - Máquina.

Por otro lado, gracias a la realización de éste proyecto se ha podido adquirir nuevos conocimientos relativos a los de cualquier instalación robótica básica, ya que se ha hecho uso y se han nombrados elementos eléctricos tales como son motores, cilindros neumáticos, relés, fotocélulas, vacuostatos, etc… Aspectos muy importantes que cualquier ingeniero con especialidad robótica debe conocer a la perfección.

El proyecto, como ya se ha comentado, ha sido un proyecto real, realizado en la empresa MP Productividad y por tanto, ha tenido los inconvenientes de los plazos exigidos para su conclusión, que ha repercutido en imposibilidad de ampliación de ciertas partes del mismo. Es por ello, que como mejoras y propuestas de ampliación al mismo existen varias opciones, como pueden ser:

- Ampliación de las posibilidades de la pantalla HMI, tales como poder modificar parámetros de los variadores desde las mismas, sin tener que hacerlo manualmente desde el armario, tal y como está pensado actualmente. Esto requeriría cambios en la configuración del HMI y por supuesto, de la programación del propio autómata.
- Estudiar la forma de poder enfardar los palets completados, para una mejor compactación de las cajas. Para ello haría falta poder adquirir una nueva máquina especializada en ello y añadirla en algún punto a la instalación.
- Estudiar la manera de introducir alguna tarjeta de memoria donde refleje diversas estadísticas de la producción realizada.
- Por último, se podría estudiar si hubiese alguna manera de poder realizar el control vía Internet. Aunque la posibilidad de esto repercutiría posiblemente

en la sustitución del propio autómata por un ordenador personal, con el inconveniente del mayor gasto que esto conllevaría.

En resumen, la realización de este proyecto nos ha servido para ampliar y descubrir nuevos conocimientos bastante útiles para cualquier Ingeniero de Telecomunicación. Conocimientos que algunos sólo se han visto muy por encima durante la carrera, otros ni se han llegado incluso a tratar, como es el caso del interfaz Hombre - Máquina, realizando con esto un caso práctico de gran utilidad.

BIBLIOGRAFÍA

Libros de consulta:

- Martínez Domínguez, Fernando. **Tecnología Eléctrica**. Ed. Paraninfo. 2003.
- Piedrafita Moreno, Ramón. **Ingeniería de la Automatización Industrial**. Ed. RA-MA, 2004.
- Creus Solé, Antonio. **Instrumentación Industrial**. Ed. Marcombo Boixareu 1997.

Material complementario:

- **Automatismos.** Apuntes de clase, 5º curso Ing. Telecomunicación
- **Control por Computador.** Apuntes de clase, 4º curso Ing. Telecomunicación
- **Manual de SIMATIC de SIEMENS**

Páginas web:

- Página oficial de SIEMENS:
  http://www.siemens.es

- Página oficial de Danfoss:
  http://www.danfoss.com

ANEXO

Listado del código

A.1 DB's y UDT's en SCL

dbMemorias (DB 5)

```
DATA_BLOCK dbMemorias
 STRUCT
   EmpujadorEnPrecarga : BOOL;
   EmpZonaSegSubida: BOOL;

   ccd1Ant: INT;
   ccd2Ant: INT;
   FinCiclo: BOOL;

   PetPuerta1: BOOL;
   PetPuerta2: BOOL;
   PetPuerta3: BOOL;
   CamadaCompleta: BOOL;   //SecDesviador2 => SecTajaderaEnt
   CamadaEmpujador: BOOL;  //SecTajaderaEnt => Empujador

   PenultimaTemp: BOOL;

   TmpFCPenultima2: S5TIME;

   CamadaEnPinza: BOOL;
   NoCamadaEnPinza: BOOL;

   ccm: INT;              //Contador de cajas de la mesa

   //Zona de paletizado
```

```
    zpSinPalet: BOOL;      //Zona de paletizado SIN palet
    zpConPalet: BOOL;      //Zona de paletizado CON palet
    PaletCompletado: BOOL; //Palet completado por el robot

    apPeticionCarga: BOOL; //Almacen de palet peticion de carga

    bEnfardarModelo: BOOL;        //Enfardar modelo actual
    bPeticionCambioFormato: BOOL; //Peticion cambiar formato
    iFormatoSeleccionado: INT;    //Formato seleccionado
    bLiberarPalet: BOOL;          //Peticion liberar palet

    //Memorias para el paso a paso ----------------------------
    bPasoAPaso: BOOL;          //Selector paso a paso
    bDosificar: BOOL;          //Memoria dosificar para el paso a paso
    ccpp: INT;                 //Contador de cajas para el paso a paso
    ncpp: INT;                 //Nº de cajas a dosificar por el paso a paso

//Zona de paletizado
    EmpujadorAvanzando: BOOL;      //Empujador avanzando, memoria para seguir
avanzando



    //Transportadores de cajas
    TdpcTC3: BOOL;
    TdpcTC2: BOOL;
    TdpcTC1: BOOL;
    TretTajadera: BOOL; //Retardo de tajadera de entrada mesa

    ContTC3: INT;
    ContTC2: INT;
    ContTC1: INT;
  END_STRUCT
BEGIN
```

```
  TmpFCPenultima2 := S5T#2S;
END_DATA_BLOCK
```

dbFormatoSelec (DB 6)

```
DATA_BLOCK dbFormatoSelec  tdFormato
BEGIN
END_DATA_BLOCK
```

tdFormato (UDT 6)

```
TYPE tdFormato
  STRUCT
  //  Descripción de tipo
    nf: INT;      //Nº de formato: 2x2(1), 2x3(2), 2x4(3)
    nCajas: INT;  //Nº de cajas del formato

    nc1: INT;     //Nº de cajas de la primera calle
    cg1: BOOL;    //Giramos en calle 1

    nc2: INT;     //Nº de cajas de la segunda calle
    cg2: BOOL;    //Giramos en calle 2

    nc3: INT;     //Nº de cajas de la tercera calle
    cg3: BOOL;    //Giramos en calle 3

    nc4: INT;     //Nº de cajas de la cuarta calle
    cg4: BOOL;    //Giramos en calle 4
  END_STRUCT
END_TYPE
```

dbFormatos (DB 7)

```
DATA_BLOCK dbFormatos
  STRUCT
    nfs: INT;     //Nº de formato seleccionado
    tf: ARRAY [1..3] OF tdFormato;  //Tabla de formatos
  END_STRUCT
BEGIN
  nfs := 2;

  //Formato 2x2
  tf[1].nf := 1;
  tf[1].nCajas := 24;

  tf[1].nc1 := 6;
  tf[1].cg1 := false;

  tf[1].nc2 := 6;
  tf[1].cg2 := false;

  tf[1].nc3 := 6;
  tf[1].cg3 := false;

  tf[1].nc4 := 6;
  tf[1].cg4 := false;

  //Formato 2x3
  tf[2].nf := 2;
  tf[2].nCajas := 16;

  tf[2].nc1 := 6;
  tf[2].cg1 := true;

  tf[2].nc2 := 6;
```

```
  tf[2].cg2 := true;

  tf[2].nc3 := 0;
  tf[2].cg3 := false;

  tf[2].nc4 := 4;
  tf[1].cg4 := false;

  //Formato 2x4
  tf[3].nf := 3;
  tf[3].nCajas := 12;

  tf[3].nc1 := 6;
  tf[3].cg1 := true;

  tf[3].nc2 := 3;
  tf[3].cg2 := false;

  tf[3].nc3 := 3;
  tf[3].cg3 := false;

  tf[3].nc4 := 0;
  tf[3].cg4 := false;
END_DATA_BLOCK
```

dbTP1 (DB 52)

```
DATA_BLOCK dbTP1 tdTransportePalet
BEGIN
END_DATA_BLOCK
```

tdTransportePalet (UDT 61)

```
TYPE tdTransportePalet
```

```
STRUCT
   dpSalida : BOOL;          //Entrada. Barrera de Salida
   coINpalet: BOOL;          //Entrada. Señal de petición de entrada de Palet (del
Elemento anterior)
   coOUTmotorArrancado: BOOL;  //Salida. Permiso para entrar palet (se ha arrancado
el motor.
   coOUTpalet: BOOL;         //Salida. Señal de petición de Salida del Palet (al
siguiente elemento)
   coINautSalida: BOOL;      //Entrada. Señal de Autorización de salida del palet
   Marcha: BOOL;             //Salida. Señal de marcha al motor del camino
   Paro: BOOL;               //Salida. Señal de parada al motor de salida
   Contactor: BOOL;          //Entrada. Señal de confirmación del motor
END_STRUCT
END_TYPE
```

dbDesviador1 (DB 60)

```
DATA_BLOCK dbDesviador1  tdDesviador1
BEGIN
END_DATA_BLOCK
```

tdDesviador 1 (UDT 120)

```
TYPE tdDesviador1
STRUCT
   FB_Fotocelula : BOOL;     //Entrada. Flanco de Bajada de SQ1053.
   ccd1 : INT:=0;            //Salida. Contador de Cajas del Desviador 2.

   Desviador_ia: BOOL;       //Entrada. Informacion Desviador a Izq.
   Desviador_ir: BOOL;       //Entrada. Informacion Desviador a Der.

   Desviador_aa : BOOL;      //Salida. Orden Mover Desviador a Izq.
   Desviador_ar : BOOL;      //Salida. Orden Mover Desviador a Der.
END_STRUCT
```

```
END_TYPE
```

dbDesviador2 (DB 61)

```
DATA_BLOCK dbDesviador2  tdDesviador2
BEGIN
END_DATA_BLOCK
```

tdDesviador2 (UDT 121)

```
TYPE tdDesviador2
STRUCT
   FB_Fotocelula : BOOL;     //Entrada. Flanco de Bajada de SQ1053.
   ccd2 : INT:=0;            //Salida. Contador de Cajas del Desviador 2.

   Desviador_ia: BOOL;       //Entrada. Informacion Desviador a Izq.
   Desviador_ir: BOOL;       //Entrada. Informacion Desviador a Der.
   Girador1_ia : BOOL;       //Entrada. Informacion Girador 1 Abajo
   Girador1_ir : BOOL;       //Entrada. Informacion Girador 1 Arriba
   Girador2_ia : BOOL;       //Entrada. Informacion Girador 2 Abajo
   Girador2_ir : BOOL;       //Entrada. Informacion Girador 2 Arrib

   Desviador_aa : BOOL;      //Salida. Orden Mover Desviador a Izq.
   Desviador_ar : BOOL;      //Salida. Orden Mover Desviador a Der.
   Girador1_aa : BOOL;       //Salida. Orden de bajar Girador 1
   Girador1_ar : BOOL;       //Salida. Orden de subir Girador 1
   Girador2_aa : BOOL;       //Salida. Orden de bajar Girador 2
   Girador2_ar : BOOL;       //Salida. Orden de subir Girador 2
END_STRUCT
END_TYPE
```

dbEmpujador (DB 62)

```
DATA_BLOCK dbEmpujador  tdEmpujador
```

```
BEGIN
END_DATA_BLOCK
```

tdEmpujador (UDT 122)

```
TYPE tdEmpujador
STRUCT
   FinCamada : BOOL;          //Entrada. Fotocelula Fin de Camada.
   CamadaEnPrecarga : BOOL;   //Entrada. Fotocelula Camada en Precarga
   CamadaenPinza : BOOL;      //Salida. Camada colocada correctamente en pinza.

   EmpRet : BOOL;             //Entrada. Señal Empujador Retrocedido (SQ1)
   EmpPre : BOOL;             //Entrada. Señal Empujador en Precarga (SQ2)
   EmpPre2: BOOL;             //Entrada. Señal Empujador en Precarga 2
   EmpAva : BOOL;             //Entrada. Señal Empujador Avanzado (SQ3)

   OAvaEmp : BOOL;            //Salida. Orden de avance del Empujador
   ORetEmp : BOOL;            //Salida. Orden de retroceso del Empujador
   OParoEmp : BOOL;           //Salida.Orden de paro del Empujador

   //C07
   TajaderaSal_ia: BOOL;
   TajaderaSal_ir: BOOL;
   TajaderaSal_aa: BOOL;
   TajaderaSal_ar: BOOL;

   //C08
   Emp_ia: BOOL;
   Emp_ir: BOOL;
   Emp_aa: BOOL;
   Emp_ar: BOOL;

   //Cuchilla Pinza
   Cuchilla_ia: BOOL;
```

```
   Cuchilla_ir: BOOL;
END_STRUCT
END_TYPE
```

dbTransportesMesa (DB 65)

```
DATA_BLOCK dbTransportesMesa  tdTransportesMesa
BEGIN
END_DATA_BLOCK
```

tdTransportesMesa (UDT 125)

```
TYPE tdTransportesMesa
STRUCT
  //Secuencias
  SecDes1: BOOL;
  SecDes2: BOOL;
  SecEmp:  BOOL;

  //Motores
  M04_Marcha: BOOL;
  M04_Paro:   BOOL;
  M05_Marcha: BOOL;
  M05_Paro:   BOOL;
  M06_Marcha: BOOL;
  M06_Paro:   BOOL;
  M07_Marcha: BOOL;
  M07_Paro:   BOOL;
  M08_Marcha: BOOL;
  M08_Paro:   BOOL;
  M09_Marcha: BOOL;
  M09_Paro:   BOOL;
END_STRUCT
END_TYPE
```

dbTajaderaEnt (DB 66)

```
DATA_BLOCK dbTajaderaEnt  tdTajaderaEnt
BEGIN
END_DATA_BLOCK
```

tdTajaderaEnt (UDT 126)

```
TYPE tdTajaderaEnt
STRUCT
   //C06
   Tajadera_ia: BOOL;
   Tajadera_ir: BOOL;
   Tajadera_aa: BOOL;
   Tajadera_ar: BOOL;
   PenultimaCamada1: BOOL;
   PenultimaCamada2: BOOL;
   FinCamada: BOOL;
END_STRUCT
END_TYPE
```

dbAlmacenCartones (DB 69)

```
DATA_BLOCK dbAlmacenCartones  tdAlmacenCartones
BEGIN
END_DATA_BLOCK
```

tdAlmacenCartones (UDT 129)

```
TYPE tdAlmacenCartones
STRUCT
  HayCarton   : BOOL;      //Entrada.
  NoHayCarton : BOOL;      //Entrada.
  RobotFcc    : BOOL;      //Entrada. Robot Fin Coger Carton
```

```
  RobotPcc    : BOOL;      //Entrada.

  Marcha      : BOOL;      //Salida. Orden de avance del Empujador
  MarchaInv   : BOOL;      //Salida. Orden de avance del Empujador
  Paro        : BOOL;      //Salida. Orden de paro del Empujador
  Cabezal_ia : BOOL;       //Entrada.
  Cabezal_ir : BOOL;       //Entrada.

  //Cilindros
  //C17
  Dientes_ia: BOOL;        //Entrada.
  Dientes_ir: BOOL;        //Entrada.
  Dientes_aa: BOOL;        //Salida.
  Dientes_ar: BOOL;        //Salida.

END_STRUCT
END_TYPE
```

dbTransferPalet1 (DB 77)

```
DATA_BLOCK dbTransferPalet1 tdTransferPalet1
BEGIN
END_DATA_BLOCK
```

tdTransferPalet1 (UDT 107)

```
TYPE tdTransferPalet1
STRUCT
   dpSalida : BOOL;          //Entrada. Barrera de Salida
   coINpalet: BOOL;          //Entrada. Señal de petición de entrada de Palet (del Elemento anterior)
   coOUTautDosificacion: BOOL; //Salida. Aitorización para dosificar al Almacen
   coOUTpalet: BOOL;         //Salida. Señal de petición de Salida del Palet (al siguiente elemento)
```

```
   coINautSalida: BOOL;       //Entrada. Señal de Autorización de salida del palet
   dpp: BOOL;                 //Entrada. Detector presencia palet
   dpPilaPalet: BOOL;         //Entrada. Señal de existencia de Pila
   coINpila : BOOL;           //Entrada. Señal de petición de entrada de Pila
   coOUTautPila: BOOL;        //Salida. Autorización entrada de Pila

   MarchaCadena: BOOL;        //Salida. Señal de marcha al motor del camino
   ParoCadena: BOOL;          //Salida. Señal de parada al motor de salida
   ContactorCadena: BOOL;     //Entrada. Señal de confirmación del motor
   MarchaRodillos: BOOL;      //Salida. Señal de marcha al motor del camino
   ParoRodillos: BOOL;        //Salida. Señal de parada al motor de salida
   Cilindro_ia : BOOL;        //Entrada. Cilindro de elevación
   Cilindro_ir : BOOL;        //Entrada.
   Cilindro_aa : BOOL;        //Salida.
   Cilindro_ar : BOOL;        //Salida.
END_STRUCT
END_TYPE
```

dbTransferPalet2 (DB 78)

```
DATA_BLOCK dbTransferPalet2 tdTransferPalet2
BEGIN
END_DATA_BLOCK
```

tdTransferPalet2 (UDT 108)

```
TYPE tdTransferPalet2
STRUCT
  Dpp           : BOOL;    //Entrada.
  STORK_A_MP    : BOOL;    //Entrada.
  coINPalet     : BOOL;    //Entrada.
  MP_A_STORK    : BOOL;    //Salida.
  coOUTautPalet : BOOL;    //Salida.
```

```
  Marcha     : BOOL;       //Salida. Orden de avance del Empujador
  Paro       : BOOL;       //Salida. Orden de paro del Empujador

  //Cilindros
  //C14
  Cilindro_ia: BOOL;       //Entrada.
  Cilindro_ir: BOOL;       //Entrada.
  Cilindro_aa: BOOL;       //Salida.
  Cilindro_ar: BOOL;       //Salida.

END_STRUCT
END_TYPE
```

dbAlmacenPalets (DB 79)

```
DATA_BLOCK dbAlmacenPalets tdAlmacenPalets
BEGIN
END_DATA_BLOCK
```

tdAlmacenPalets (UDT 109)

```
TYPE tdAlmacenPalets
STRUCT
   dpPila : BOOL;            //Entrada. Detección de Pila
   dpPosicion1: BOOL;        //Entrada. Posición Baja. Cogida Pila Completa
   dpPosicion2: BOOL;        //Entrada. Posición Media. Cogida Pila -1 palet
   dpPosicion3:BOOL;         //Entrada. Posición Alta. Salida de Palet
   dpp: BOOL;                //Entrada. Detector presencia palet

   coINautDosificar: BOOL;     //Entrada. Autorización para dosificar
   coOUTpaletDosificado: BOOL; //Salida. Indicación de palet dosificado
   coOUTpeticionPila: BOOL;    //Salida. Petición de Nueva Pila de Palets

   // Motor de elevación
```

```
  Marcha    : BOOL;          //Salida
  MarchaInv : BOOL;          //Salida
  Paro      : BOOL;          //Salida

  // Cilindro de la pinza
  Pinza_ia : BOOL;           //Entrada
  Pinza_ir : BOOL;           //Entrada
  Pinza_aa : BOOL;           //Salida
  Pinza_ar : BOOL;           //Salida

END_STRUCT
END_TYPE
```

dbZonaPaletizado (DB 81)

```
DATA_BLOCK dbZonaPaletizado tdZonaPaletizado
BEGIN
END_DATA_BLOCK
```

tdZonaPaletizado (UDT 131)

```
TYPE tdZonaPaletizado
STRUCT
  Dpp :    BOOL;       //Entrada. Fotocelula presencia palet
  Marcha : BOOL;       //Salida. Orden de avance
  Paro :   BOOL;       //Salida.Orden de paro

  //Comunicaciones
  coINautSalida: BOOL; //Entrada. Autorizacion para sacar el palet
  coINpalet:     BOOL; //Entrada. El transporte anterior esta metiendo un palet
  coOUTsalida:   BOOL; //Salida. Sacando el palet
  coOUTautPalet: BOOL; //Salida. Permiso para meter un palet

  //Cilindros
```

```
  //C15
  Tope_ia: BOOL;       //Entrada.
  Tope_ir: BOOL;       //Entrada.
  Tope_aa: BOOL;       //Salida.
  Tope_ar: BOOL;       //Salida.

  //C16
  Centrador_ia: BOOL;  //Entrada.
  Centrador_ir: BOOL;  //Entrada.
  Centrador_aa: BOOL;  //Salida.
  Centrador_ar: BOOL;  //Salida.

END_STRUCT
END_TYPE
```

dbAlarmas (DB 91) (*)

```
DATA_BLOCK dbAlarmas
  STRUCT
   alm_Z00 : BOOL ; //ALARMA GENERAL DE SISTEMA
   alm_Z11 : BOOL ; //ALARMA EN ZONA 11
   alm_Z12 : BOOL ; //ALARMA EN ZONA 12
   alm_Z13 : BOOL ; //ALARMA EN ZONA 13

   […]

  END_STRUCT ;
BEGIN
END_DATA_BLOCK
```

dbVisualizacion (DB 92) (*)

```
DATA_BLOCK "dbVisualizacion"
  STRUCT
```

```
  E8_00 : BOOL ;   //Magnetotermico Alimentacion Logica cableada
  E8_01 : BOOL ;   //Magnetotermico Alimentacion Entradas
  E8_02 : BOOL ;   //Magnetotermico Alimentacion Salidas
  E8_03 : BOOL ;   //Termico Motor 1 Transportador de Cajas 1
  E8_04 : BOOL ;   //Termico Motor 2 Transportador de Cajas 2
  E8_05 : BOOL ;   //Termico Motor 3 Transportador de Cajas 3
  E8_06 : BOOL ;   //Termico Motor 4 Tranportador de banda 1
  E8_07 : BOOL ;   //Magnetotermico Alimentacion 220V
  E9_00 : BOOL ;   //Termico Motor 5 Tranportador de banda 2

  […]

 END_STRUCT ;
BEGIN
END_DATA_BLOCK
```

dbManuales (DB 93) (*)

```
 STRUCT
  bmA8_00 : BOOL ;
  bmA8_01 : BOOL ;
  bmA8_02 : BOOL ;

  […]

  bmVM21D : BOOL ;
  bmVM21I : BOOL ;
  bmVM22D : BOOL ;
  bmVM22I : BOOL ;
 END_STRUCT ;
BEGIN
END_DATA_BLOCK
```

DB_CAMBIO_HORA (DB 98)

```
DATA_BLOCK "DB_CAMBIO_HORA"
 STRUCT
  CAMBIAR_HORA : BOOL ;    //HABILITAR CAMBIO DE HORA DESDE TP
  FECHAYHORA_NUEVA : DATE_AND_TIME ;   //N.O.-HORA
  FECHAYHORA_BCD : INT ;   //MINUTO-SEGUNDO
  FECHAYHORA_BCD1 : INT ;  //N.O.-N.O.
  FECHAYHORA_BCD2 : INT ;  //N.O.-DIA SEMANA
  FECHAYHORA_BCD3 : INT ;  //DIA-MES
  FECHAYHORA_BCD4 : INT ;  //ANO-N.O.
  FECHAYHORA_BCD5 : INT ;  //ANO-N.O.ANO-N.O.
 END_STRUCT;
BEGIN
  CAMBIAR_HORA := FALSE;
  FECHAYHORA_NUEVA := DT#90-1-1-0:0:0.000;
  FECHAYHORA_BCD := 0;
  FECHAYHORA_BCD1 := 0;
  FECHAYHORA_BCD2 := 0;
  FECHAYHORA_BCD3 := 0;
  FECHAYHORA_BCD4 := 0;
  FECHAYHORA_BCD5 := 0;
END_DATA_BLOCK
```

dbTC1 (DB 141)

```
DATA_BLOCK dbTC1  tdTC
BEGIN
END_DATA_BLOCK
```

tdTC (UDT 140)

```
TYPE tdZonaPaletizado
STRUCT
 Marcha : BOOL;              //Salida. Orden de avance.
 Paro :  BOOL;               //Salida.  Orden de paro.
```

```
  coINcaja : BOOL;          //Entrada. El transporte anterior esta metiendo una caja
  coINautCaja : BOOL;       //Entrada. Autorización para sacar la caja
  coOUTcaja : BOOL;         //Salida. Sacando caja
  coOUTautCaja : BOOL;      //Salida. Permiso para meter una caja
  dpc : BOOL;               //Entrada. Detector presencia de caja
  dpcAnt : BOOL;            //Entrada. Detector presencia de caja del tramo anterior
END_STRUCT
END_TYPE
```

dbTP270_Modos (DB 200)

```
DATA_BLOCK dbTP270_Modos
  STRUCT

  //---------------------------------------- MODO GENERAL
    MG_AUT_MAN : BOOL ; //SELECCIÓN DEL MODO DE FUNCIONAMIENTO
    MG_MARCHA_CICLO : BOOL ; //INCIAR CICLO DE PRODUCCION
    MG_PARADA_CICLO : BOOL ; //FINALIZAR CICLO DE PRODUCCION
    MG_REARME_DEFECTO : BOOL ; //REARME DE LA INSTALACION

    MG_LAMP_COND_INICIO : BOOL ; //INSTALACIÓN EN CONDICIONES INICIALES
    MG_LAMP_SEG_AUTO : BOOL ; //INSTALACIÓN CON SEGURIDAD DE AUTOMATICO
    MG_LAMP_SEG_MANUAL : BOOL ; //INSTALACIÓN CON SEGURIDAD DE MANUAL
    MG_LAMP_MARCHA_CICLO : BOOL ; //INSTALACIÓN EN MARCHA
    MG_LAMP_REARME_DEFECTO : BOOL ; //INFORMACION DE QUE SE DEBE REARMAR LA INSTALACION
    MG_LAMP_PARADA_EMERG : BOOL ; //INSTALACIÓN EN PARADA DE EMERGENCIA

    //---------------------------------------- PLANTA PALETIZADO
    PP_AUT_MAN : BOOL ; //SELECCIÓN DEL MODO DE FUNCIONAMIENTO
```

```
   PP_MARCHA_CICLO : BOOL ; //INCIAR CICLO DE PRODUCCION
   PP_PARADA_CICLO : BOOL ; //FINALIZAR CICLO DE PRODUCCION
   PP_REARME_DEFECTO : BOOL ; //REARME DE LA INSTALACION

   PP_LAMP_COND_INICIO : BOOL ; //INSTALACIÓN EN CONDICIONES INICIALES
   PP_LAMP_SEG_AUTO : BOOL ; //INSTALACIÓN CON SEGURIDAD DE AUTOMATICO
   PP_LAMP_SEG_MANUAL : BOOL ; //INSTALACIÓN CON SEGURIDAD DE MANUAL
   PP_LAMP_MARCHA_CICLO : BOOL ; //INSTALACIÓN EN MARCHA
   PP_LAMP_REARME_DEFECTO : BOOL ; //INFORMACION DE QUE SE DEBE REARMAR LA INSTALACION
   PP_LAMP_PARADA_EMERG : BOOL ; //INSTALACIÓN EN PARADA DE EMERGENCIA

   //---------------------------------------- ALMACEN DE PALETS
   AP_AUT_MAN : BOOL ; //SELECCIÓN DEL MODO DE FUNCIONAMIENTO
   AP_MARCHA_CICLO : BOOL ; //INCIAR CICLO DE PRODUCCION
   AP_PARADA_CICLO : BOOL ; //FINALIZAR CICLO DE PRODUCCION
   AP_REARME_DEFECTO : BOOL ; //REARME DE LA INSTALACION

   AP_LAMP_COND_INICIO : BOOL ; //INSTALACIÓN EN CONDICIONES INICIALES
   AP_LAMP_SEG_AUTO : BOOL ; //INSTALACIÓN CON SEGURIDAD DE AUTOMATICO
   AP_LAMP_SEG_MANUAL : BOOL ; //INSTALACIÓN CON SEGURIDAD DE MANUAL
   AP_LAMP_MARCHA_CICLO : BOOL ; //INSTALACIÓN EN MARCHA
   AP_LAMP_REARME_DEFECTO : BOOL ; //INFORMACION DE QUE SE DEBE REARMAR LA INSTALACION
   AP_LAMP_PARADA_EMERG : BOOL ; //INSTALACIÓN EN PARADA DE EMERGENCIA
```

```
   //---------------------------------------- ALMACEN DE CARTONES
   AC_AUT_MAN : BOOL ; //SELECCIÓN DEL MODO DE FUNCIONAMIENTO
   AC_MARCHA_CICLO : BOOL ; //INCIAR CICLO DE PRODUCCION
   AC_PARADA_CICLO : BOOL ; //FINALIZAR CICLO DE PRODUCCION
   AC_REARME_DEFECTO : BOOL ; //REARME DE LA INSTALACION

   AC_LAMP_COND_INICIO : BOOL ; //INSTALACIÓN EN CONDICIONES INICIALES
   AC_LAMP_SEG_AUTO : BOOL ; //INSTALACIÓN CON SEGURIDAD DE AUTOMATICO
   AC_LAMP_SEG_MANUAL : BOOL ; //INSTALACIÓN CON SEGURIDAD DE MANUAL
   AC_LAMP_MARCHA_CICLO : BOOL ; //INSTALACIÓN EN MARCHA
   AC_LAMP_REARME_DEFECTO : BOOL ; //INFORMACION DE QUE SE DEBE REARMAR LA INSTALACION
   AC_LAMP_PARADA_EMERG : BOOL ; //INSTALACIÓN EN PARADA DE EMERGENCIA

  END_STRUCT
BEGIN

END_DATA_BLOCK
```

(*) De estas funciones tan sólo se muestran un extracto para no extender en demasía el contenido del capítulo.

A.2 FC's en SCL

fcMapearPeriferia (FC 2)

```
FUNCTION fcMapearPeriferia : VOID
  //Robot
  MD40  := PED40;
  MD44  := PED44;
  MD48  := PED48;
  MD52  := PED52;

  PAD40 := MD56;
  PAD44 := MD60;
  PAD48 := MD64;
  PAD52 := MD68;

  // ET200[1]
  MD101 := PED101;
  MD105 := PED105;
  MB109 := PEB109;

  PAD151 := MD151;
  PAB155 := MB155;

  // ET200[2]
  MD201 := PED201;
  MB205 := PEB205;

  PAW251 := MW251;
  PAB253 := MB253;

  // ET200[3]
  MD301 := PED301;
```

```
  MW305 := PEW305;
  MB307 := PEB307;

  PAD351 := MD351;

  // ET200[4]
  MD401 := PED401;

  PAW451 := MW451;

END_FUNCTION
```

fcSeguridades (FC 3)

```
FUNCTION fcSeguridades : VOID
VAR_TEMP
  CAP1 : BOOL;   //Condiciones para la apertura de la puerta 1
  CAP2 : BOOL;   //Condiciones para la apertura de la puerta 2
  CAP3 : BOOL;   //Condiciones para la apertura de la puerta 3
END_VAR
BEGIN
  // Seguridades de los modos de funcionamiento
  dbModoGeneral.SegMando := QF07 AND QF10 AND QF11 AND QF12 AND
EnServicio;
  dbModoGeneral.SegAuto := dbModoGeneral.SegMando
    AND dbModoPlantaPaletizado.SegAuto
    AND dbModoAlmacenCartones.SegAuto
    AND dbModoAlmacenPalets.SegAuto;
    //and variadores AND robot;

  dbModoPlantaPaletizado.SegMando := dbModoGeneral.SegMando;
  dbModoPlantaPaletizado.SegAuto := dbModoPlantaPaletizado.SegMando AND
PR1050 AND PR3032
```

```
  AND QKM01 AND QKM02 AND QKM03 AND QKM04 AND QKM05 AND
QKM06 AND QKM07
  AND QKM08 AND QKM09 AND QKM19 AND QKM21 AND RS02 AND RSB;

 dbModoAlmacenCartones.SegMando := dbModoGeneral.SegMando;
 dbModoAlmacenCartones.SegAuto := dbModoAlmacenCartones.SegMando AND
PR3032
  AND QKM20 AND RS03;

 dbModoAlmacenPalets.SegMando := dbModoGeneral.SegMando;
 dbModoAlmacenPalets.SegAuto := dbModoAlmacenPalets.SegMando AND PR3032
  AND QKM10 AND QKM11 AND QKM12 AND QKM13 AND QKM14 AND
QKM15 AND QKM16 AND QKM17
  AND (QKM18 OR true) AND QKM22 AND RS02 AND RSB;

 // Peticion de apertura de puertas
 CAP1 := dbModoPlantaPaletizado.Manual OR (
    RobotInEspPcp OR RobotInEspPcc OR RobotInEspPdc OR RobotInEspPdp OR
    RobotInHome OR RobotInPcp OR RobotInPcc OR RobotInPdp OR RobotInPdc)
    OR RS01 OR (NOT RSB);
 CAP2 := CAP1;

 CAP3 := dbModoAlmacenCartones.Manual OR RS01 OR
       (dbAlmacenCartones.Cabezal_ia OR dbAlmacenCartones.Cabezal_ir);

 IF FS_SB11 THEN
  dbMemorias.PetPuerta1 := NOT dbMemorias.PetPuerta1;
 END_IF;
 IF FS_SB12 THEN
  dbMemorias.PetPuerta2 := NOT dbMemorias.PetPuerta2;
 END_IF;
 IF FS_SB13 THEN
  dbMemorias.PetPuerta3 := NOT dbMemorias.PetPuerta3;
 END_IF;
```

```
  KAP1 := dbMemorias.PetPuerta1 AND CAP1;
  HL11 := KAP1 OR ((dbMemorias.PetPuerta1 AND NOT CAP1) AND Inter);

  KAP2 := dbMemorias.PetPuerta2 AND CAP2;
  HL12 := KAP2 OR ((dbMemorias.PetPuerta2 AND NOT CAP2) AND Inter);

  KAP3 := dbMemorias.PetPuerta3 AND CAP3;
  HL13 := KAP3 OR ((dbMemorias.PetPuerta3 AND NOT CAP3) AND Inter);
END_FUNCTION
```

fcFlancos (FC 5)

```
//-------------------------------------------------------------------------
// Estructura del byte de control:
//   7   6   5   4   3   2   1   0
// ---------------------------------
// |   |   |   |   | b | s | a | v |
// ---------------------------------
// LEYENDA:
// v: Valor actual de la variable.
// a: Valor anterior de la variable.
// s: Flanco de subida.
// b: Flanco de bajada.
//
// Rango de direcciones para bytes de control
// MB1000 - MB1099
// Primero libre: 1007
//-------------------------------------------------------------------------
FUNCTION fcFlancos : VOID
   // MB1000: Flancos de la marca de ciclo de 0'1 seg
   // Simbolo reMarcaDecimas = M1000.2
   M1000.0 := M100.0;
   IF NOT M1000.1 AND M1000.0 THEN    //Flanco de subida
      M1000.2 := true;
```

```
ELSE
  M1000.2 := false;
END_IF;
IF M1000.1 AND NOT M1000.0 THEN    //Flanco de bajada
  M1000.3 := true;
ELSE
  M1000.3 := false;
END_IF;
M1000.1 := M1000.0;

// MB1001: Flancos de la FC6
// Simbolo FC6 = SQ1052
M1001.0 := SQ1052;
IF NOT M1001.1 AND M1001.0 THEN    //Flanco de subida
  M1001.2 := true;
ELSE
  M1001.2 := false;
END_IF;
IF M1001.1 AND NOT M1001.0 THEN    //Flanco de bajada
  M1001.3 := true;
ELSE
  M1001.3 := false;
END_IF;
M1001.1 := M1001.0;

// MB1002: Flancos de la FC7
// Simbolo FC7 = SQ1053
M1002.0 := SQ1053;
IF NOT M1002.1 AND M1002.0 THEN    //Flanco de subida
  M1002.2 := true;
ELSE
  M1002.2 := false;
END_IF;
IF M1002.1 AND NOT M1002.0 THEN    //Flanco de bajada
```

```
  M1002.3 := true;
ELSE
  M1002.3 := false;
END_IF;
M1002.1 := M1002.0;

// MB1003: Flancos de la FC5
// Simbolo FC5 = SQ1051
M1003.0 := SQ1051;
IF NOT M1003.1 AND M1003.0 THEN    //Flanco de subida
  M1003.2 := true;
ELSE
  M1003.2 := false;
END_IF;
IF M1003.1 AND NOT M1003.0 THEN    //Flanco de bajada
  M1003.3 := true;
ELSE
  M1003.3 := false;
END_IF;
M1003.1 := M1003.0;

// MB1004: Flancos del SB11
M1004.0 := SB11;
IF NOT M1004.1 AND M1004.0 THEN    //Flanco de subida
  M1004.2 := true;
ELSE
  M1004.2 := false;
END_IF;
IF M1004.1 AND NOT M1004.0 THEN    //Flanco de bajada
  M1004.3 := true;
ELSE
  M1004.3 := false;
END_IF;
M1004.1 := M1004.0;
```

```
  // MB1005: Flancos del SB12
  M1005.0 := SB12;
  IF NOT M1005.1 AND M1005.0 THEN    //Flanco de subida
    M1005.2 := true;
  ELSE
    M1005.2 := false;
  END_IF;
  IF M1005.1 AND NOT M1005.0 THEN    //Flanco de bajada
    M1005.3 := true;
  ELSE
    M1005.3 := false;
  END_IF;
  M1005.1 := M1005.0;

  // MB1006: Flancos del SB13
  M1006.0 := SB13;
  IF NOT M1006.1 AND M1006.0 THEN    //Flanco de subida
    M1006.2 := true;
  ELSE
    M1006.2 := false;
  END_IF;
  IF M1006.1 AND NOT M1006.0 THEN    //Flanco de bajada
    M1006.3 := true;
  ELSE
    M1006.3 := false;
  END_IF;
  M1006.1 := M1006.0;

END_FUNCTION
```

fcMemorias (FC 6)

```
FUNCTION fcMemorias : VOID
```

```
BEGIN
  //Memoria empujador en precarga
  IF SQ1072 AND dbM19.Contactor THEN
    dbMemorias.EmpujadorEnPrecarga := true;
  END_IF;
  IF SQ1072 AND dbM19.ContactorInverso THEN
    dbMemorias.EmpujadorEnPrecarga := false;
  END_IF;

  //Memoria empujador en zona segura de subida
  IF SQ1072 AND dbM19.Contactor THEN
    dbMemorias.EmpZonaSegSubida := true;
  END_IF;
  IF SQ1072 AND dbM19.ContactorInverso THEN
    dbMemorias.EmpZonaSegSubida := false;
  END_IF;

  IF SQ1061 AND dbM19.Contactor THEN
    dbMemorias.EmpZonaSegSubida := false;
  END_IF;
  IF SQ1061 AND dbM19.ContactorInverso THEN
    dbMemorias.EmpZonaSegSubida := true;
  END_IF;

  //Memoria peticion de carga del almacen de palets
  IF SB15 THEN
    dbMemorias.apPeticionCarga := true;
  END_IF;
  IF dbMemorias.apPeticionCarga AND dbM10.Contactor THEN
    dbMemorias.apPeticionCarga := false;
  END_IF;

  //Memorias de camada en pinza
```

```
 dbMemorias.CamadaEnPinza := dbRobot.CamadaPinza AND dbC10.ia AND
dbC11.ia;
 dbMemorias.NoCamadaEnPinza := (NOT dbRobot.CamadaPinza) AND dbC10.ir
AND dbC11.ir;

 //Memorias de zona paletizado
 dbMemorias.zpSinPalet := NOT SQ3041 AND dbC15.ia AND dbC16.ir;
 dbMemorias.zpConPalet :=     SQ3041 AND dbC15.ia AND dbC16.ia;

 dbMemorias.TmpFCPenultima2 := T#1S250MS;

 IF dbModoPlantaPaletizado.Manual THEN
   dbMemorias.EmpujadorAvanzando := false;
   DBMemorias.CamadaEmpujador := false;
 END_IF;

END_FUNCTION
```

fcMotor (FC 11)

```
FUNCTION fcMotor : VOID
VAR_INPUT
   MF : tdModoFuncionamiento;
END_VAR
VAR_IN_OUT
   M : tdMotor;    //Instancia de Motor
END_VAR
BEGIN

M.Contactor := M.termico AND M.SeguridadMecanica AND NOT M.Paro AND
          ((MF.Automatico AND MF.EnCiclo AND (M.Marcha OR M.Contactor)) OR
          (MF.Manual AND M.OrdenManual));
//Para implementar un pulso, en la secuencia automática se activan y aquí se desactivan
M.Marcha := FALSE;
```

```
M.Paro:= FALSE;

M.Error:=M.marcha AND M.paro OR (NOT M.marcha AND NOT M.paro);
M.Alarma:=NOT M.termico;
END_FUNCTION
```

fcMotorInversor (FC 12)

```
FUNCTION fcMotorInversor : VOID
VAR_INPUT
   MF : tdModoFuncionamiento;
END_VAR
VAR_IN_OUT
   M : tdMotorInversor;    //Instancia de Motor inversor
END_VAR
BEGIN

M.Contactor := M.termico AND M.SeguridadMecanica AND NOT M.Paro AND
        ((MF.Automatico AND MF.EnCiclo AND (M.Marcha OR M.Contactor)) OR
        (MF.Manual AND M.OrdenManual));

M.ContactorInverso := M.termico AND M.SeguridadMecanicaInversa AND NOT
M.Paro AND
        ((MF.Automatico AND MF.EnCiclo AND (M.MarchaInversa OR
M.ContactorInverso)) OR
        (MF.Manual AND M.OrdenManualInversa));

//Para implementar un pulso, en la secuencia automática se activan y aquí se desactivan
M.Marcha := FALSE;
M.MarchaInversa := FALSE;
M.Paro:= FALSE;

M.Error:=(M.marcha OR M.marchaInversa) AND M.paro OR (NOT M.marcha AND
NOT M.marchaInversa AND NOT M.paro);
```

```
M.Alarma:=NOT M.Termico;
END_FUNCTION
```

fcMotorVariador (FC 13)

```
FUNCTION fcMotorVariador : VOID
VAR_INPUT
   MF : tdModoFuncionamiento;
END_VAR
VAR_IN_OUT
   M : tdMotorVariador;    //Instancia de Motor Variador
END_VAR
VAR_TEMP
  CAnt: BOOL;
  CAntInv: BOOL;
  error: INT;
END_VAR
BEGIN
CAnt := M.Contactor;
M.Contactor := M.termico AND M.SeguridadMecanica AND NOT M.Paro AND
          ((MF.Automatico AND MF.EnCiclo AND (M.Marcha OR M.Contactor)) OR
          (MF.Manual AND M.OrdenManual));
IF NOT CAnt AND M.Contactor THEN //Arranque
  ErrorVariador:=fcVLTRun(dir:=M.NumVar,
vel:=INT_TO_WORD(M.ConsignaVelocidad));
ELSIF CAnt AND NOT M.Contactor THEN //Parada;
  ErrorVariador:=fcVLTStop(dir:=M.NumVar); //dir
END_IF;

CAntInv := M.ContactorInverso;
M.ContactorInverso := M.termico AND M.SeguridadMecanicaInversa AND NOT
M.Paro AND
          ((MF.Automatico AND MF.EnCiclo AND (M.MarchaInversa OR
M.ContactorInverso)) OR
```

```
        (MF.Manual AND M.OrdenManualInversa));
IF NOT CAntInv AND M.ContactorInverso THEN //Arranque
  ErrorVariador:=fcVLTRunI(dir:=M.NumVar,
vel:=INT_TO_WORD(M.ConsignaVelocidad));
ELSIF CAntInv AND NOT M.ContactorInverso THEN //Parada;
  ErrorVariador:=fcVLTStop(dir:=M.NumVar);
END_IF;

IF CAnt AND CAntInv THEN
   ErrorVariador:=fcVLTStop(dir:=M.NumVar);
END_IF;

//Para implementar un pulso, en la secuencia automática se activan y aquí se desactivan
M.Marcha := FALSE;
M.MarchaInversa := FALSE;
M.Paro:= FALSE;

M.Error:=(M.marcha OR M.marchaInversa) AND M.paro OR (NOT M.marcha AND
NOT M.marchaInversa AND NOT M.paro);
M.Alarma:=NOT M.termico;
END_FUNCTION
```

fcCilindro (FC 15)

```
FUNCTION fcCilindro : VOID
VAR_INPUT
   MF : tdModoFuncionamiento;
END_VAR
VAR_IN_OUT
   C : tdCilindro;    //Instancia de Cilindro
END_VAR

BEGIN
//Informaciones
```

```
C.ia:= C.da AND NOT C.dr;
C.ir:= C.dr AND NOT C.da;

//Autorizaciones
C.oav:=C.sa AND ((MF.Automatico AND MF.EnCiclo AND C.aa) OR (MF.Manual
AND C.oma)) AND NOT C.ia;
C.ore:=C.sr AND ((MF.Automatico AND MF.EnCiclo AND C.ar) OR (MF.Manual
AND C.omr)) AND NOT C.ir;

//Control de tiempo
IF C.oav THEN
  IF reMarcaDecimas THEN
    C.cta := C.cta + 1;
  END_IF;
ELSE
  C.cta := 0;
END_IF;

IF C.ore THEN
  IF reMarcaDecimas THEN
    C.ctr := C.ctr + 1;
  END_IF;
ELSE
  C.ctr := 0;
END_IF;

IF NOT C.da AND NOT C.dr THEN
  IF reMarcaDecimas THEN
    C.ctd := C.ctd + 1;
  END_IF;
ELSE
  C.ctd := 0;
END_IF;
```

```
//Alarmas
C.ama:= C.oav AND (C.cta >= C.ta);
C.amr:= C.ore AND (C.ctr >= C.tr);
C.afd:= (C.ctd >= C.td);

//Errores
C.error:= C.aa AND C.ar;
END_FUNCTION
```

fcCtrlAlmacenCartones (FC 59)

```
FUNCTION fcCtrlAlmacenCartones : VOID
//*****************************************************************\\
//* Almacen de Cartones                                           *\\
//*****************************************************************\\
  dbAlmacenCartones.HayCarton   := (SQ4030 AND SQ4031);
  dbAlmacenCartones.NoHayCarton := (NOT SQ4030) AND (NOT SQ4031);
  dbAlmacenCartones.RobotFcc    := RobotInFcc;
  dbAlmacenCartones.RobotPcc    := RobotInPcc;

  dbAlmacenCartones.Cabezal_ia := (SQ4022 OR SQ4023);
  dbAlmacenCartones.Cabezal_ir := (SQ4020 OR SQ4021);

  dbAlmacenCartones.Dientes_ia := dbC17.ia;
  dbAlmacenCartones.Dientes_ir := dbC17.ir;


  dbSecAlmacenCartones.CondicionesIniciales := true;
  dbSecAlmacenCartones.EnCiclo := dbModoAlmacenCartones.EnCiclo;
  dbSecAlmacenCartones.Automatico := dbModoAlmacenCartones.Automatico;

  fcSecAlmacenCartones(S:=dbSecAlmacenCartones, D:=dbAlmacenCartones);

END_FUNCTION
```

fcCtrlTransportesPalets (FC 61)

```
FUNCTION fcCtrlTransportesPalets : VOID
BEGIN
//*************************************************************\\
//* Transporte de Palet 1. Entrada desde Carretilla           *\\
//*************************************************************\\
  dbTP1.dpSalida := dbMemorias.apPeticionCarga;
  dbTP1.coINpalet := dbMemorias.apPeticionCarga AND
dbTransferPalet1.ContactorCadena;
  dbTP1.coINautSalida := dbTransferPalet1.coOUTautPila AND
dbTransferPalet1.ContactorCadena;
  dbTP1.Contactor := dbM10.Contactor;

  dbSecTranspPalet1.CondicionesIniciales := TRUE;
  dbSecTranspPalet1.EnCiclo := dbModoAlmacenPalets.EnCiclo;
  dbSecTranspPalet1.Automatico := dbModoAlmacenPalets.Automatico;

  fcSecTransportePalet(S:=dbSecTranspPalet1, D:=dbTP1);

//*******************************************************\\
//* Almacen de palets                                   *\\
//*******************************************************\\
  dbAlmacenPalets.dpPila      := SQ3060;
  dbAlmacenPalets.dpPosicion1 := SQ3063;
  dbAlmacenPalets.dpPosicion2 := SQ3052;
  dbAlmacenPalets.dpPosicion3 := SQ3061 OR SQ3062;
  dbAlmacenPalets.dpp         := SQ3033 OR SQ3053;

  dbAlmacenPalets.coINautDosificar := dbTransferPalet1.coOUTautDosificacion;
  dbAlmacenPalets.Pinza_ia := dbC09.ia;
  dbAlmacenPalets.Pinza_ir := dbC09.ir;
```

```
  dbSecAlmacenPalets.CondicionesIniciales := dbAlmacenPalets.Pinza_ia OR
dbAlmacenPalets.Pinza_ir;
  dbSecAlmacenPalets.EnCiclo := dbModoAlmacenPalets.EnCiclo;
  dbSecAlmacenPalets.Automatico := dbModoAlmacenPalets.Automatico;

  fcSecAlmacenPalets(S:=dbSecAlmacenPalets, D:=dbAlmacenPalets);

//*********************************************************\\
//* Transferencia 1                                       *\\
//*********************************************************\\
  dbTransferPalet1.dpSalida := SQ3040;
  dbTransferPalet1.coINpalet := dbAlmacenPalets.coOUTpaletDosificado;
  dbTransferPalet1.coINautSalida := dbZonaPaletizado.coOUTautPalet;
  dbTransferPalet1.dpPilaPalet := SQ3060;
  dbTransferPalet1.dpp := SQ3033;
  dbTransferPalet1.coINpila := (dbSecAlmacenPalets.Etapa=2);

  dbTransferPalet1.ContactorCadena := KM11;
  dbTransferPalet1.Cilindro_ia := dbC13.ia;
  dbTransferPalet1.Cilindro_ir := dbC13.ir;

  dbSecTransferPalet1.CondicionesIniciales := TRUE;
  dbSecTransferPalet1.EnCiclo := dbModoAlmacenPalets.EnCiclo;
  dbSecTransferPalet1.Automatico := dbModoAlmacenPalets.Automatico;

  dbSecTransferPalet1.TiempoAlarma[0] := 0; //Tiempo salida de palet
  dbSecTransferPalet1.TiempoAlarma[1] := 0; //Tiempo de estabilizacion arranque
cadenas
  dbSecTransferPalet1.TiempoAlarma[2] := 0; //Tiempo de estabilizacion subir rodillos

  fcSecTransferPalet1(S:=dbSecTransferPalet1, D:=dbTransferPalet1);

//*********************************************************\\
//* Zona Paletizado                                       *\\
```

```
//*****************************************************\\
  dbZonaPaletizado.Dpp := SQ3041;
  dbZonaPaletizado.coINautSalida := dbTP4.coOUTmotorArrancado;
  dbZonaPaletizado.coINpalet := dbTransferPalet1.coOUTpalet;
  dbZonaPaletizado.Tope_ia := dbC15.ia;
  dbZonaPaletizado.Tope_ir := dbC15.ir;
  dbZonaPaletizado.Centrador_ia := dbC16.ia;
  dbZonaPaletizado.Centrador_ir := dbC16.ir;

  dbSecZonaPaletizado.CondicionesIniciales := true;
  dbSecZonaPaletizado.EnCiclo := dbModoAlmacenPalets.EnCiclo;
  dbSecZonaPaletizado.Automatico := dbModoAlmacenPalets.Automatico;

  fcSecZonaPaletizado(S:=dbSecZonaPaletizado, D:=dbZonaPaletizado);

//*************************************************************\\
//* Transporte de Palet 3. Salida Posición de Apilado         *\\
//*************************************************************\\
   dbTP3.dpSalida := SQ3043;
  dbTP3.coINpalet := dbZonaPaletizado.coOUTsalida;
  dbTP3.coINautSalida := dbTP4.coOUTmotorArrancado;
  dbTP3.Contactor := dbM14.Contactor;

  dbSecTranspPalet3.CondicionesIniciales := TRUE;
  dbSecTranspPalet3.EnCiclo := dbModoAlmacenPalets.EnCiclo;
  dbSecTranspPalet3.Automatico := dbModoAlmacenPalets.Automatico;

  fcSecTransportePalet(S:=dbSecTranspPalet3, D:=dbTP3);

//*****************************************************\\
//* Transporte de Palet 4.                            *\\
//*****************************************************\\
  dbTP4.dpSalida := SQ3043; //FC18;
  dbTP4.coINpalet :=  dbZonaPaletizado.coOUTsalida;
```

```
  dbTP4.coINautSalida := dbTP5.coOUTmotorArrancado;
  dbTP4.Contactor := dbM15.Contactor;

  dbSecTranspPalet4.CondicionesIniciales := TRUE;
  dbSecTranspPalet4.EnCiclo := dbModoAlmacenPalets.EnCiclo;
  dbSecTranspPalet4.Automatico := dbModoAlmacenPalets.Automatico;

  fcSecTransportePalet(S:=dbSecTranspPalet4, D:=dbTP4);

//*********************************************************\\
//* Transporte de Palet 5                                 *\\
//*********************************************************\\
  dbTP5.dpSalida := SQ3050; //FC19;
  dbTP5.coINpalet :=  dbTP4.coOUTpalet;
  dbTP5.coINautSalida := dbTransferPalet2.coOUTautPalet;
  dbTP5.Contactor := dbM16.Contactor;

  dbSecTranspPalet5.CondicionesIniciales := TRUE;
  dbSecTranspPalet5.EnCiclo := dbModoAlmacenPalets.EnCiclo;
  dbSecTranspPalet5.Automatico := dbModoAlmacenPalets.Automatico;

  fcSecTransportePalet(S:=dbSecTranspPalet5, D:=dbTP5);

//*********************************************************\\
//* Transferencia 2                                       *\\
//*********************************************************\\
  dbTransferPalet2.Dpp := SQ3051;
  dbTransferPalet2.STORK_A_MP := "DATOS STORK_A_MP".Permiso_entrada;
  dbTransferPalet2.coINpalet := dbTP5.coOUTpalet;
  dbTransferPalet2.Cilindro_ia := dbC14.ia;
  dbTransferPalet2.Cilindro_ir := dbC14.ir;

  dbSecTransferPalet2.CondicionesIniciales := dbC14.ir AND (NOT dbC14.ia);
  dbSecTransferPalet2.EnCiclo := dbModoAlmacenPalets.EnCiclo;
```

```
  dbSecTransferPalet2.Automatico := dbModoAlmacenPalets.Automatico;

  dbSecTransferPalet2.TiempoAlarma[0] := 0; //Tiempo inserccion arranque de rodillos

  fcSecTransferPalet2(S:=dbSecTransferPalet2, D:=dbTransferPalet2);

END_FUNCTION
```

fcCtrlTransportesCajas (FC 65)

```
FUNCTION fcCtrlTransportesCajas : VOID
BEGIN
//*********************************************************\\
//* Transportes de la mesa                                *\\
//*********************************************************\\
  dbSecTransportesMesa.CondicionesIniciales := TRUE;
  dbSecTransportesMesa.EnCiclo := dbModoPlantaPaletizado.EnCiclo;
  dbSecTransportesMesa.Automatico := dbModoPlantaPaletizado.Automatico ;

  dbTransportesMesa.SecDes1 := (dbSecDesviador1.Etapa>0);
  dbTransportesMesa.SecDes2 := (dbSecDesviador2.Etapa>0);
  dbTransportesMesa.SecEmp  := (dbSecEmpujador.Etapa>0);

  fcSecTransportesMesa(S:=dbSecTransportesMesa, D:=dbTransportesMesa);

//*********************************************************\\
//* Transportes de cajas                                  *\\
//*********************************************************\\
  dbTC1.Marcha := dbC01.ir AND (dbSecTransportesMesa.Etapa=8);
  dbTC1.Paro := dbC01.ia OR dbMemorias.TdpcTC1;

  dbTC2.Marcha := dbC01.ir AND (dbSecTransportesMesa.Etapa=8);
  dbTC2.Paro := dbC01.ia OR dbMemorias.TdpcTC2;
```

```
  dbTC3.Marcha := dbC01.ir AND (dbSecTransportesMesa.Etapa=8);
  dbTC3.Paro := dbC01.ia OR dbMemorias.TdpcTC3;
END_FUNCTION
```

fcCtrlMesa (FC 69)

```
FUNCTION fcCtrlMesa : VOID
BEGIN
//************************************************************\\
//* Tope entrada transportadores de banda                    *\\
//************************************************************\\
  IF dbMemorias.bDosificar THEN
    dbMemorias.ccpp := 0;
  END_IF;

  IF FB_SQ1051 THEN
    dbMemorias.ccm := dbMemorias.ccm + 1;

    dbMemorias.ccpp := dbMemorias.ccpp + 1;
    dbMemorias.bDosificar := false;
  END_IF;

  dbC01.aa := ((dbMemorias.ccm >= (dbFormatoSelec.nCajas*2)) AND NOT
dbMemorias.bPasoAPaso) OR
          ((dbMemorias.ccpp >= dbMemorias.ncpp) AND dbMemorias.bPasoAPaso)
OR
          dbMemorias.TretTajadera;
  dbC01.ar := (((dbMemorias.ccm < (dbFormatoSelec.nCajas*2)) AND NOT
dbMemorias.bPasoAPaso) OR
          ((dbMemorias.ccpp < dbMemorias.ncpp) AND dbMemorias.bPasoAPaso))
AND NOT dbMemorias.TretTajadera;

  //dbC01.aa := (dbMemorias.ccm >= (dbFormatoSelec.nCajas*1));
  //dbC01.ar := (dbMemorias.ccm <  (dbFormatoSelec.nCajas*1));
```

```
//*****************************************************\\
//* Desviador 1                                       *\\
//*****************************************************\\
  dbDesviador1.Desviador_ia  := dbC02.ia;
  dbDesviador1.Desviador_ir  := dbC02.ir;
  dbDesviador1.FB_Fotocelula := FB_SQ1052;

  dbSecDesviador1.CondicionesIniciales := true;
  dbSecDesviador1.EnCiclo := dbModoPlantaPaletizado.EnCiclo;
  dbSecDesviador1.Automatico := dbModoPlantaPaletizado.Automatico ;

  fcSecDesviador1(S:=dbSecDesviador1, D:=dbDesviador1);

//*****************************************************\\
//* Desviador 2                                       *\\
//*****************************************************\\
  dbDesviador2.Desviador_ia  := dbC03.ia;
  dbDesviador2.Desviador_ir  := dbC03.ir;
  dbDesviador2.Girador1_ia   := dbC04.ia;
  dbDesviador2.Girador1_ir   := dbC04.ir;
  dbDesviador2.Girador2_ia   := dbC05.ia;
  dbDesviador2.Girador2_ir   := dbC05.ir;
  dbDesviador2.FB_Fotocelula := FB_SQ1053;

  dbSecDesviador2.CondicionesIniciales := true;
  dbSecDesviador2.EnCiclo := dbModoPlantaPaletizado.EnCiclo;
  dbSecDesviador2.Automatico := dbModoPlantaPaletizado.Automatico;

  fcSecDesviador2(S:=dbSecDesviador2, D:=dbDesviador2);

//*****************************************************\\
//* Tajadera Entrada de Camadas                       *\\
//*****************************************************\\
  dbTajaderaEnt.FinCamada        := SQ1060; //FC8
```

```
  dbTajaderaEnt.PenultimaCamada1 := SQ1090;
  dbTajaderaEnt.PenultimaCamada2 := SQ1091;
  dbTajaderaEnt.Tajadera_ia := dbC06.ia;
  dbTajaderaEnt.Tajadera_ir := dbC06.ir;

  dbSecTajaderaEnt.CondicionesIniciales := (DBMemorias.CamadaEmpujador AND
dbC06.ia) OR
                                (NOT DBMemorias.CamadaEmpujador AND dbC06.ir);
  dbSecTajaderaEnt.EnCiclo := dbModoPlantaPaletizado.EnCiclo;
  dbSecTajaderaEnt.Automatico := dbModoPlantaPaletizado.Automatico;

  fcSecTajaderaEnt(S:=dbSecTajaderaEnt, D:=dbTajaderaEnt);

//*****************************************************\\
//* Empujador                                           *\\
//*****************************************************\\
  dbEmpujador.FinCamada       := SQ1060; //FC8
  dbEmpujador.CamadaEnPrecarga := SQ1083; //FC9
  dbEmpujador.CamadaenPinza    := SQ2040; //FC10

  dbEmpujador.EmpRet          := SQ1071 OR SQ1081;
  dbEmpujador.EmpPre          := SQ1072;
  dbEmpujador.EmpPre2         := SQ1061;
  dbEmpujador.EmpAva          := SQ1073 OR SQ1082;

  dbEmpujador.TajaderaSal_ia := dbC07.ia;
  dbEmpujador.TajaderaSal_ir := dbC07.ir;
  dbEmpujador.Emp_ia        := dbC08.ia;
  dbEmpujador.Emp_ir        := dbC08.ir;
  dbEmpujador.Cuchilla_ia    := dbC10.ia;
  dbEmpujador.Cuchilla_ir    := dbC10.ir;

  dbSecEmpujador.CondicionesIniciales := (dbEmpujador.EmpRet AND
```

```
 (NOT dbEmpujador.CamadaEnPrecarga) AND (dbEmpujador.TajaderaSal_ia OR
true)AND dbC06.ir) OR
 (dbMemorias.EmpujadorEnPrecarga AND dbC06.ia AND
dbEmpujador.CamadaEnPrecarga) OR
 ((dbEmpujador.EmpRet OR dbMemorias.EmpujadorAvanzando)AND
(DBEmpujador.TajaderaSal_ia OR true) AND
 dbMemorias.CamadaEmpujador AND  dbC06.ia);
//modificar para considerar preparacion efectuada

 dbSecEmpujador.EnCiclo := dbModoPlantaPaletizado.EnCiclo;
 dbSecEmpujador.Automatico := dbModoPlantaPaletizado.Automatico;

 fcSecEmpujador(S:=dbSecEmpujador, D:=dbEmpujador);

//*****************************************************\\
//* 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);

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
```

fcAlarmas (FC 91)

```
FUNCTION fcAlarmas : VOID
BEGIN
 // Alarmas de zonas
 dbAlarmas.alm_Z00 := false;
 dbAlarmas.alm_Z11 := (dbAlarmas.alm1062 OR dbAlarmas.alm1080) OR dbM01.alarma;
 dbAlarmas.alm_Z12 := (dbAlarmas.alm1051 OR dbAlarmas.alm1063 OR dbAlarmas.alm1070) OR
          (dbC01.ama OR dbC01.amr OR dbC01.afd) OR (dbM02.alarma OR dbM03.alarma);
 dbAlarmas.alm_Z13 := (dbM04.alarma OR dbM05.alarma OR dbM06.alarma OR dbM07.alarma) OR
          (dbC02.ama OR dbC02.amr OR dbC02.afd) OR dbAlarmas.alm1052;
 dbAlarmas.alm_Z14 := dbAlarmas.alm1053 OR dbM08.alarma OR (dbC03.ama OR dbC03.amr OR dbC03.afd);
 dbAlarmas.alm_Z15 := dbM09.alarma OR (dbC04.ama OR dbC04.amr OR dbC04.afd) OR
          (dbC05.ama OR dbC05.amr OR dbC05.afd);
 dbAlarmas.alm_Z16 := dbAlarmas.alm1060 OR dbAlarmas.alm1083 OR //dbAlarmas.alm1090 OR dbAlarmas.alm1091 OR
          (dbC06.ama OR dbC06.amr OR dbC06.afd) OR (dbC07.ama OR dbC07.amr OR dbC07.afd);
 dbAlarmas.alm_Z17 := dbAlarmas.alm1061 OR dbAlarmas.alm1071 OR dbAlarmas.alm1072 OR
          dbAlarmas.alm1073 OR dbAlarmas.alm1081 OR dbAlarmas.alm1082 OR
          dbM19.alarma OR (dbC08.ama OR dbC08.amr OR dbC08.afd);
```

```
 dbAlarmas.alm_Z20 := dbM21.alarma OR (dbC10.ama OR dbC10.amr OR
dbC10.afd) OR
        (dbC11.ama OR dbC11.amr OR dbC11.afd) OR (dbC12.ama OR
dbC12.amr OR dbC12.afd) OR
        dbAlarmas.alm2030 OR dbAlarmas.alm2031 OR dbAlarmas.alm2032
OR dbAlarmas.alm2033 OR
        dbAlarmas.alm2022 OR dbAlarmas.alm2023 OR dbAlarmas.alm2040
OR
        dbAlarmas.alm2522 OR dbAlarmas.alm2523;
 dbAlarmas.alm_Z31 := dbM10.alarma;
 dbAlarmas.alm_Z32 := dbAlarmas.alm3033 OR dbAlarmas.alm3053 OR
dbAlarmas.alm3060 OR
        dbAlarmas.alm3061 OR dbAlarmas.alm3062 OR dbAlarmas.alm3063
OR
        dbM22.alarma OR (dbC09.ama OR dbC09.amr OR dbC09.afd);
 dbAlarmas.alm_Z33 := dbM11.alarma OR dbM12.alarma OR dbAlarmas.alm3040 OR
        (dbC13.ama OR dbC13.amr OR dbC13.afd);
 dbAlarmas.alm_Z34 := dbM13.alarma OR dbAlarmas.alm3041 OR
        (dbC15.ama OR dbC15.amr OR dbC15.afd) OR (dbC16.ama OR
dbC16.amr OR dbC16.afd);
 dbAlarmas.alm_Z35 := dbAlarmas.alm3042 OR dbAlarmas.alm3043 OR
dbM14.alarma OR dbM15.alarma;
 dbAlarmas.alm_Z36 := dbAlarmas.alm3050 OR dbAlarmas.alm3051 OR
dbM16.alarma OR dbM17.alarma OR
        (dbC14.ama OR dbC14.amr OR dbC14.afd);
 dbAlarmas.alm_Z41 := dbAlarmas.alm4020 OR dbAlarmas.alm4021 OR
dbAlarmas.alm4022 OR
        dbAlarmas.alm4023 OR dbAlarmas.alm4030 OR dbAlarmas.alm4031
OR
        dbAlarmas.alm4032 OR dbAlarmas.alm4033 OR dbAlarmas.alm4040
OR
        (dbC17.ama OR dbC17.amr OR dbC17.afd) OR (dbC18.ama OR
dbC18.amr OR dbC18.afd) OR
        dbM20.alarma;
```

```
  //Alarmas de hardware
dbAlarmas.alm_CPU    := false;
dbAlarmas.alm_DP     := false;
dbAlarmas.alm_P3     := false;
dbAlarmas.alm_E8_11  := false;
dbAlarmas.alm_E12_15 := false;
dbAlarmas.alm_S8_11  := false;
dbAlarmas.alm_S12_15 := false;
dbAlarmas.alm_P8     := false;
dbAlarmas.alm_P9     := false;
dbAlarmas.alm_P10    := false;
dbAlarmas.alm_P11     := false;

//Alarmas de protecciones
dbAlarmas.alm0080 := NOT QF03; //MT alimentacion 220
dbAlarmas.alm0113 := NOT QF05; //MT ventilacion armario
dbAlarmas.alm0081 := NOT QF07; //MT alim logica cableada
dbAlarmas.alm0082 := NOT QF10; //MT alim entradas
dbAlarmas.alm0083 := NOT QF11; //MT alim salidas

//Alarmas de seguridades
dbAlarmas.alm0121 := ST01; //Seta AA
dbAlarmas.alm0122 := ST02; //Seta PG
dbAlarmas.alm0123 := ST03; //Seta CC1
dbAlarmas.alm0124 := ST04; //Seta CC3
dbAlarmas.alm0125 := ST05; //Seta CC5
dbAlarmas.alm0126 := ST06; //Seta CC6
dbAlarmas.alm0127 := ST07; //Seta CC7
dbAlarmas.alm0130 := RS01; //Rele de setas
dbAlarmas.alm0131 := NOT RS02; //Rele de puertas
dbAlarmas.alm0132 := NOT RS03; //Rele de cartones
dbAlarmas.alm0133 := NOT RSB; //Rele mutting 1
dbAlarmas.alm0134 := NOT RSB; //Rele mutting 2
```

```
  //Alarmas comunes
  dbAlarmas.alm1050 := NOT PR1050; //Presostato Zona Neumatica 1
  dbAlarmas.alm3032 := NOT PR3032; //Presostato Zona Neumatica 2
  dbAlarmas.alm3060 := NOT SQ3060; //Almacen de Palets Vacio
  dbAlarmas.alm4033 := NOT SQ4033; //Almacen de cartones vacío 1
  dbAlarmas.alm4040 := NOT SQ4040; //Almacen de cartones vacío 2

  dbAlarmas.alm3063 := (dbSecAlmacenPalets.Etapa=901);

//Baliza del armario AA
HLB0 := (dbSecEmpujador.Etapa=120) AND
      (dbSecEmpujador.TiempoEtapa>20) AND
      (SQ1092 OR SQ1093); //Maquina en automatico pero a la espera de alguna
validacion del operario
HLB1 := dbModoPlantaPaletizado.EnCiclo AND dbModoAlmacenPalets.EnCiclo AND
dbModoAlmacenCartones.EnCiclo;
HLB2 := dbAlarmas.alm3060 OR dbAlarmas.alm4033 OR dbAlarmas.alm4040;
HLB3 := NOT HLB1;

//Baliza del almacen de palets
HL3540 := dbAlarmas.alm3060;

END_FUNCTION
```

fcSecTransportePalet (FC 101)

```
FUNCTION fcSecTransportePalet : VOID
VAR_IN_OUT
   S : tdSecuencia;       //Instancia de la Secuencia
   D : tdTransportePalet;   //Instancia de los Elementos de Transporte
END_VAR

BEGIN
//Preliminar
```

```
//-------------------------------------------------------------------------

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

//Acciones a la Activación
//-------------------------------------------------------------------------

//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.Paro := (S.Etapa = 1) OR (S.Etapa = 2) OR (S.Etapa = 7);
D.Marcha := (S.Etapa = 3) OR (S.Etapa = 8);

D.coOUTmotorArrancado := (S.Etapa=4);
D.coOUTpalet := (S.Etapa=5) OR (S.Etapa=6) OR (S.Etapa=7) OR (S.Etapa=8);

//Transiciones
//-------------------------------------------------------------------------

//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 NOT D.dpSalida THEN
      S.EtapaSiguiente := 2;     //No hay palet en el Transporte
   ELSIF D.dpSalida THEN
      S.EtapaSiguiente := 7;     //Hay palet en el transporte
   END_IF;
END_IF;

IF S.Etapa = 2 AND D.coINpalet THEN
   S.EtapaSiguiente := 3;     //Va a entrar un palet
END_IF;

IF S.Etapa = 3 AND D.Contactor THEN
   S.EtapaSiguiente := 4;     //Motor en Regimen
END_IF;

IF S.Etapa = 4 AND D.dpSalida THEN
   S.EtapaSiguiente := 5;     //Palet en Posición
END_IF;

IF S.Etapa = 5 THEN
   S.EtapaSiguiente := 6;     //Pasamos Siempre
END_IF;
```

```
IF S.Etapa = 6 THEN
   IF D.coINautSalida THEN
      S.EtapaSiguiente := 8;      //Continuamos la salida del palet
   ELSIF NOT D.coINautSalida THEN
      S.EtapaSiguiente := 7;      //No continuar con la salida del palet
   END_IF;
END_IF;

IF S.Etapa = 7 AND D.coINautSalida THEN
   S.EtapaSiguiente := 8;      //Autorización para sacar el palet
END_IF;

IF S.Etapa = 8 THEN
   IF NOT D.dpSalida AND D.coINpalet THEN
      S.EtapaSiguiente := 4;      //Continuamos con una entrada de Palet
   ELSIF NOT D.dpSalida AND NOT D.coINpalet THEN
      S.EtapaSiguiente := 2;      //Sale palet y no hay petición de entrada
   END_IF;
END_IF;

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

//Posterior
//----------------------------------------------------------------------------

END_FUNCTION
```

fcSecTransferPalet1 (FC 107)

```
FUNCTION fcSecTransferPalet1 : VOID
VAR_IN_OUT
   S : tdSecuencia;        //Instancia de la Secuencia
   D : tdTransferPalet1;   //Instancia de los Elementos de Transporte
```

```
END_VAR

BEGIN
//Preliminar
//---------------------------------------------------------------------------

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

//Acciones a la Activación
//---------------------------------------------------------------------------

//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.MarchaCadena := (S.Etapa=8) OR (S.Etapa=9) OR (S.Etapa=11);
D.ParoCadena := (S.Etapa=10) OR (S.Etapa=12);

D.MarchaRodillos := (S.Etapa=4);
D.ParoRodillos := (S.Etapa=5);

D.Cilindro_ar := (S.Etapa=5);
```

```
D.Cilindro_aa := (S.Etapa=3);

D.coOUTautDosificacion := (S.Etapa = 2) OR (S.Etapa = 10);
D.coOUTpalet := (S.Etapa=2) OR (S.Etapa=3);
D.coOUTautPila := (S.Etapa=9);

//Transiciones
//------------------------------------------------------------------------

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

//Comprobación de Condiciones Iniciales (Siempre Igual)
IF (S.Etapa = 0) THEN
   IF S.CondicionesIniciales AND NOT S.FinCiclo THEN
      S.EtapaSiguiente := 1;         //Test Palet en transfer
   END_IF;
END_IF;



IF (S.Etapa = 1) THEN
   IF D.dpSalida THEN
      S.EtapaSiguiente := 2;      //Peticion de Salida y Aut. Dosificar
   ELSIF NOT D.dpSalida THEN
      S.EtapaSiguiente := 6;      //Test Pila de Palets y Aut. Dosificar
   END_IF;
END_IF;

IF (S.Etapa = 2) AND D.coINautSalida THEN
   S.EtapaSiguiente := 3;      //Subimos Rodillos y Aut.Dosificar
END_IF;
```

```
IF (S.Etapa = 3) AND D.Cilindro_ia THEN
   S.EtapaSiguiente := 4;      //Arrancamos rodillos y Aut. Dosificar
END_IF;

IF (S.Etapa = 4) AND NOT D.coINautSalida THEN
   S.EtapaSiguiente := 5;      //Bajamos Rodillos y Aut. Dosificar
END_IF;

IF (S.Etapa = 5) AND D.Cilindro_ir THEN
   S.EtapaSiguiente := 6;      //Test Pila de Palets y Aut. Dosificar
END_IF;

IF (S.Etapa = 6) THEN
   IF (NOT D.dpPilaPalet AND NOT D.dpp) OR D.coINpila THEN
      S.EtapaSiguiente := 7;      //Esperar Petición Entrada Pila
   ELSIF (D.dpPilaPalet OR D.dpp) THEN
      S.EtapaSiguiente := 10;      //Espera Dosificación  y Aut. Dosificar
   END_IF;
END_IF;

IF (S.Etapa = 7) AND D.coINpila THEN
   S.EtapaSiguiente := 8;      //Arrancar Cadenas
END_IF;

IF (S.Etapa = 8) AND D.ContactorCadena THEN
   S.EtapaSiguiente := 9;      //Autorización entrada de Pila
END_IF;

IF (S.Etapa = 9) AND D.dpPilaPalet THEN
   S.EtapaSiguiente := 910;      //Autorización para dosoficar
END_IF;

IF (S.Etapa = 910) AND (S.TiempoEtapa>20) THEN
  S.EtapaSiguiente := 10;
```

```
END_IF;

IF (S.Etapa = 10) AND D.coINpalet THEN
   S.EtapaSiguiente := 11;      //Arrancar las cadenas
END_IF;

IF (S.Etapa = 11) AND D.dpSalida THEN
   S.EtapaSiguiente := 12;      //Espera de estabilización
END_IF;

IF (S.Etapa = 12) AND (S.TiempoEtapa>S.TiempoAlarma[1]) THEN
   S.EtapaSiguiente := 13;      //Subir transfer
END_IF;

IF (S.Etapa = 13) THEN
   IF S.EnCiclo THEN
      S.EtapaSiguiente := 2;      //Petición de Salida y Aut. Dosicicación
   ELSE
      S.EtapaSiguiente := 0;
   END_IF;
END_IF;

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

//Posterior
//-------------------------------------------------------------------------

END_FUNCTION
```

fcSecTransferPalet2 (FC 108)

```
FUNCTION fcSecTransferPalet2 : VOID
VAR_IN_OUT
```

```
   S : tdSecuencia;       //Instancia de la Secuencia
   D : tdTransferPalet2;  //Instancia de los Elementos de la transferencia de palets 2
END_VAR

BEGIN
//Preliminar
//--------------------------------------------------------------------------

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

//Acciones a la Activación
//--------------------------------------------------------------------------

//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.Marcha := (S.Etapa=4);
D.Paro := (S.Etapa=5);

D.MP_A_SALIDA := (S.Etapa=2) OR (S.Etapa=3) OR (S.Etapa=4) OR (S.Etapa=5)
OR (S.Etapa=6);
```

```
D.coOUTautPalet := (S.Etapa=4);

D.Cilindro_aa := (S.Etapa=3);
D.Cilindro_ar := (S.Etapa=6);

//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.coINpalet) THEN
   S.EtapaSiguiente := 2;
  END_IF;
END_IF;

IF (S.Etapa = 2) AND (D.STORK_A_MP) THEN // AND NOT D.Dpp) THEN
   S.EtapaSiguiente := 3;
END_IF;

IF (S.Etapa = 3) AND D.Cilindro_ia THEN
   S.EtapaSiguiente := 4;
END_IF;
```

```
IF (S.Etapa = 4) AND (D.Dpp) THEN
   S.EtapaSiguiente := 5;
END_IF;

IF (S.Etapa = 5) AND (S.TiempoEtapa > S.TiempoAlarma[0]) THEN
   S.EtapaSiguiente := 6;
END_IF;

IF (S.Etapa = 6) AND (D.Cilindro_ir) THEN
   S.EtapaSiguiente := 7;
END_IF;

IF (S.Etapa = 7) AND (NOT D.STORK_A_MP) THEN
   S.EtapaSiguiente := 8;
END_IF;

IF (S.Etapa = 8) THEN
  IF S.EnCiclo THEN
    S.EtapaSiguiente := 1;
  ELSE
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

END_IF;
//Acciones a la Desactivación
//---------------------------------------------------------------------------

//Posterior
//---------------------------------------------------------------------------

END_FUNCTION
```

fcSecAlmacenPalets (FC109)

```
FUNCTION fcSecAlmacenPalets : VOID
VAR_IN_OUT
   S : tdSecuencia;       //Instancia de la Secuencia
   D : tdAlmacenPalets;   //Instancia de los Elementos de Transporte
END_VAR

BEGIN
//Preliminar
//---------------------------------------------------------------------------

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

//Acciones a la Activación
//---------------------------------------------------------------------------

//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.coOUTpaletDosificado := (S.Etapa=7);
D.coOUTpeticionPila := (S.Etapa=2) OR (S.Etapa=23);

//Cerrar Pînas
D.Pinza_aa := (S.Etapa=4);
//Abrir Pinzas
D.Pinza_ar := (S.Etapa=9);

//Elevador Subir
D.Marcha := (S.Etapa=3) OR (S.Etapa=5);
//Elevador Bajar
D.MarchaInv := (S.Etapa=8);
//Elevador parada inicial
D.Paro := (S.Etapa=4) OR (S.Etapa=6) OR (S.Etapa=9);

//Transiciones
//--------------------------------------------------------------------------

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

//Comprobación de Condiciones Iniciales (Siempre Igual)
IF S.Etapa = 0 THEN
   IF S.CondicionesIniciales AND S.EnCiclo THEN
      S.EtapaSiguiente := 1;         //Test Palet dosificado
   END_IF;
END_IF;



IF S.Etapa = 1 THEN
   IF NOT D.dpPila THEN
     IF D.dpPosicion1 THEN
```

```
      S.EtapaSiguiente := 2;      //Esperar Pila
     ELSIF D.dpPosicion3 THEN
      S.EtapaSiguiente := 12;
     END_IF;
   ELSIF D.dpPila AND D.Pinza_ia THEN
      S.EtapaSiguiente := 11;      //Ir a posicion 3
   ELSIF D.dpPila AND D.Pinza_ir AND D.dpPosicion1 THEN
      S.EtapaSiguiente := 3;

   END_IF;
END_IF;

IF (S.Etapa = 2) AND D.dpPila THEN
   S.EtapaSiguiente := 23;      //Ir a posicion 2
END_IF;

IF (S.Etapa = 23) AND (S.TiempoEtapa=40) THEN
  S.EtapaSiguiente := 3;
END_IF;

IF S.Etapa = 3 AND D.dpPosicion2 THEN
  IF D.dpPila THEN
   S.EtapaSiguiente := 4;      //Cerrar pinza
  ELSIF NOT D.dpPila THEN
   S.EtapaSiguiente := 5;      //No cerramos pinza porque es el ultimo palet
  END_IF;
END_IF;

IF S.Etapa = 4 AND D.Pinza_ia THEN
   S.EtapaSiguiente := 5;      //Ir a posicion 3
END_IF;

IF S.Etapa = 5 AND D.dpPosicion3 THEN
   S.EtapaSiguiente := 6;      //Test dpp
```

```
END_IF;

IF S.Etapa = 6 THEN
   IF D.dpp THEN
      S.EtapaSiguiente := 7;      //Esperar aut dosificar
   ELSIF NOT D.dpp THEN
      S.EtapaSiguiente := 100;
   END_IF;
END_IF;

IF S.Etapa = 7 AND D.coINautDosificar AND NOT D.dpp THEN
   S.EtapaSiguiente := 8;      //Bajar Elevador a Posición 1
END_IF;

IF S.Etapa = 8 THEN
  IF D.dpPosicion1 THEN
    S.EtapaSiguiente := 9;      //Abrir Pinzas
  ELSIF (S.TiempoEtapa>=55) THEN
    S.EtapaSiguiente := 901;    //Alarma detector posicion 1
  END_IF;
END_IF;

IF S.Etapa = 9 AND D.Pinza_ir THEN
   S.EtapaSiguiente := 10;      //Subir Elevador a Posición 2
END_IF;

IF S.Etapa = 10 THEN
   IF S.FinCiclo THEN
      S.EtapaSiguiente := 0;
   ELSE
      S.EtapaSiguiente := 2;
   END_IF;
END_IF;
```

```
IF S.Etapa = 11 AND D.dpPosicion3 THEN
   S.EtapaSiguiente := 12;      //Test dpp
END_IF;

IF S.Etapa = 12 THEN
   IF D.dpp THEN
      S.EtapaSiguiente := 7;      //Test
   ELSIF NOT D.dpp THEN
      S.EtapaSiguiente := 8;
   END_IF;
END_IF;

IF S.Etapa = 100 AND SB03 THEN
   S.EtapaSiguiente := 1;      //Decidir punto entrada
END_IF;

IF S.Etapa = 901 AND SB03 THEN
   S.EtapaSiguiente := 1;      //Decidir punto entrada
END_IF;

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

//Posterior
//------------------------------------------------------------------------

END_FUNCTION
```

fcSecDesviador1 (FC 120)

```
FUNCTION fcSecDesviador1 : VOID
VAR_IN_OUT
   S : tdSecuencia;    //Instancia de la Secuencia
   D : tdDesviador1;   //Instancia de los Elementos de Transporte
```

```
END_VAR
VAR_TEMP
  cambio: BOOL;
END_VAR
BEGIN
//Preliminar
//--------------------------------------------------------------------------
IF D.FB_Fotocelula  AND S.EnCiclo AND S.Automatico THEN
  D.ccd1 := D.ccd1 + 1;
END_IF;
IF dbMemorias.ccd1Ant<>D.ccd1 THEN
  cambio := true;
  dbMemorias.ccd1Ant := D.ccd1;
ELSE
  cambio := false;
END_IF;
IF D.ccd1 >= dbFormatoSelec.nCajas THEN
  D.ccd1 := 0;
  dbMemorias.ccd1Ant := 0;
END_IF;

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

//Acciones a la Activación
//--------------------------------------------------------------------------

//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.Desviador_aa := (S.Etapa=4);
  D.Desviador_ar := (S.Etapa=5);

//Transiciones
//--------------------------------------------------------------------------

//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 AND true THEN //cambio THEN
   S.EtapaSiguiente := 2;
END_IF;

IF S.Etapa = 2 THEN

  CASE dbFormatoSelec.nf OF
    1 : S.EtapaSiguiente := 3;
```

```
    2 : S.EtapaSiguiente := 6;
    3 : S.EtapaSiguiente := 7;
  END_CASE;
END_IF;

IF S.Etapa = 3 THEN
  IF (D.ccd1>=0 AND D.ccd1<12) THEN
    S.EtapaSiguiente := 4;
  ELSIF (D.ccd1>=12 AND D.ccd1<24)THEN
    S.EtapaSiguiente := 5;
  END_IF;
END_IF;

IF S.Etapa = 4 AND D.Desviador_ia THEN
    S.EtapaSiguiente := 8;
END_IF;

IF S.Etapa = 5 AND D.Desviador_ir THEN
    S.EtapaSiguiente := 8;
END_IF;

IF S.Etapa = 6 THEN
  IF D.ccd1<12 THEN
    S.EtapaSiguiente := 4;
  ELSE
    S.EtapaSiguiente := 5;
  END_IF;
END_IF;

IF S.Etapa = 7 THEN
  IF D.ccd1<6 THEN
    S.EtapaSiguiente := 4;
  ELSE
    S.EtapaSiguiente := 5;
```

```
  END_IF;
END_IF;

IF S.Etapa = 8 THEN
  IF S.EnCiclo THEN
    S.EtapaSiguiente := 1;
  ELSE
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

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

//Posterior
//----------------------------------------------------------------------------

END_FUNCTION
```

fcSecDesviador2 (FC 121)

```
FUNCTION fcSecDesviador2 : VOID
VAR_IN_OUT
   S : tdSecuencia;    //Instancia de la Secuencia
   D : tdDesviador2;   //Instancia de los Elementos de Transporte
END_VAR
VAR_TEMP
  cambio: BOOL;
END_VAR
BEGIN
//Preliminar
//----------------------------------------------------------------------------
IF D.FB_Fotocelula AND S.EnCiclo AND S.Automatico THEN
  D.ccd2 := D.ccd2 + 1;
```

```
  IF dbMemorias.ccd2Ant<>D.ccd2 THEN
    cambio := true;
  ELSE
    cambio := false;
  END_IF;
  dbMemorias.ccd2Ant := D.ccd2;
END_IF;

IF (D.ccd2 >= dbFormatoSelec.nCajas) THEN
  D.ccd2 := 0;
  dbMemorias.ccd2Ant := 0;
  dbMemorias.CamadaCompleta := true;
END_IF;

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

//Acciones a la Activación
//---------------------------------------------------------------------------

//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.Desviador_aa := (S.Etapa=6);
  D.Desviador_ar := (S.Etapa=7);
  D.Girador1_aa  := (S.Etapa=8);
  D.Girador1_ar  := (S.Etapa=2);
  D.Girador2_aa  := (S.Etapa=8);
  D.Girador2_ar  := (S.Etapa=2);

//Transiciones
//-----------------------------------------------------------------------------

//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 dbFormatoSelec.nf=1 THEN
    S.EtapaSiguiente := 2;
  ELSE
    S.EtapaSiguiente := 8;
  END_IF;
END_IF;

IF S.Etapa = 2 AND D.Girador1_ir AND D.Girador2_ir THEN
   S.EtapaSiguiente := 3;
END_IF;
```

```
IF S.Etapa = 3 AND true THEN //cambio THEN
   S.EtapaSiguiente := 4;
END_IF;

IF S.Etapa = 4 THEN

  CASE dbFormatoSelec.nf OF
    1 : S.EtapaSiguiente := 5;
    2 : S.EtapaSiguiente := 9;
    3 : S.EtapaSiguiente := 10;
  END_CASE;
END_IF;

IF S.Etapa = 5 THEN
  IF (D.ccd2>=0 AND D.ccd2<6) OR (D.ccd2>=12 AND D.ccd2<18) THEN
    S.EtapaSiguiente := 6;
  ELSIF (D.ccd2>=6 AND D.ccd2<12) OR (D.ccd2>=18 AND D.ccd2<24)THEN
    S.EtapaSiguiente := 6;
  END_IF;
END_IF;

IF S.Etapa = 6 AND D.Desviador_ia THEN
   S.EtapaSiguiente := 11;
END_IF;

IF S.Etapa = 7 AND D.Desviador_ir THEN
   S.EtapaSiguiente := 11;
END_IF;

IF S.Etapa = 8 AND D.Girador1_ia AND D.Girador2_ia THEN
   S.EtapaSiguiente := 3;
END_IF;

IF S.Etapa = 9 THEN
```

```
  IF D.ccd2<6 THEN
    S.EtapaSiguiente := 6;
  ELSE
    S.EtapaSiguiente := 7;
  END_IF;
END_IF;

IF S.Etapa = 10 THEN
  IF D.ccd2<9 THEN
    S.EtapaSiguiente := 6;
  ELSE
    S.EtapaSiguiente := 7;
  END_IF;
END_IF;

IF S.Etapa = 11 THEN
  IF S.EnCiclo THEN
    S.EtapaSiguiente := 3;
  ELSE
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

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

//Posterior
//-------------------------------------------------------------------------

END_FUNCTION
```

fcSecEmpujador (FC 122)

```
FUNCTION fcSecEmpujador : VOID
```

```
VAR_IN_OUT
   S : tdSecuencia;    //Instancia de la Secuencia
   D : tdEmpujador;   //Instancia de los Elementos de Empujador
END_VAR

BEGIN
//Preliminar
//-------------------------------------------------------------------------

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

//Acciones a la Activación
//-------------------------------------------------------------------------
IF S.EtapaSiguiente = 11 AND S.Etapa <> 11 THEN
   dbMemorias.CamadaEmpujador := false;
   dbMemorias.ccm := dbMemorias.ccm - dbFormatoSelec.nCajas;
   IF dbMemorias.ccm < 0 THEN
     dbMemorias.ccm := 0;
   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.TajaderaSal_aa := (S.Etapa=11); //C07
 D.TajaderaSal_ar := (S.Etapa=4) OR (dbSecRobot.Etapa=4); //C07
 D.Emp_aa         := (S.Etapa=9); //C08
 D.Emp_ar         := (S.Etapa=12); //C08

 //D.OAvaEmp := (S.Etapa=2) OR (S.Etapa=5);
 D.ORetEmp := (S.Etapa=8) OR (S.Etapa=9) OR (S.Etapa=11);
 //D.OParoEmp :=(S.Etapa=3) OR (S.Etapa=6) OR (S.Etapa=10);

 //Mejora del ciclo del empujador para evitar que se pare en posicion intermedia
 D.OAvaEmp := (S.Etapa=2) OR (S.Etapa=5) OR
   (((S.Etapa=3) OR (S.Etapa=4)) AND (dbSecRobot.Etapa=4) AND D.TajaderaSal_ir
);

 D.OParoEmp :=((S.Etapa=3) AND (dbSecRobot.Etapa<>4)) OR (S.Etapa=6) OR
(S.Etapa=10);

//Transiciones
//------------------------------------------------------------------------------
IF S.Enciclo THEN

//Activación de la Secuencia
IF (S.Etapa = -1) AND S.Automatico THEN
   S.EtapaSiguiente := 0;
   //jbrl ver cuando borrar la memoria de empujador avanzando
dbMemorias.EmpujadorAvanzando
END_IF;

//Comprobación de Condiciones Iniciales (Siempre Igual)
IF (S.Etapa = 0) THEN
   IF S.CondicionesIniciales AND S.EnCiclo THEN
      S.EtapaSiguiente := 100;      //Comenzamos a ejecutar la Secuencia
```

```
   END_IF;
END_IF;

IF (S.Etapa = 100) THEN
   IF dbMemorias.EmpujadorEnPrecarga THEN
     S.EtapaSiguiente := 3;      //Camada en precarga
   ELSIF dbMemorias.EmpujadorAvanzando THEN
     S.EtapaSiguiente := 2;      //Empujador estaba empujando
   ELSE
     S.EtapaSiguiente := 1;      //camada sin completar
   END_IF;
END_IF;

IF (S.Etapa = 1) AND dbMemorias.CamadaEmpujador THEN
   S.EtapaSiguiente := 120;
   dbMemorias.EmpujadorAvanzando:= TRUE;
END_IF;

IF (S.Etapa = 120) AND (NOT SQ1092) AND (NOT SQ1093) THEN //Error
fotocelulas de camada bien posicionadas
   S.EtapaSiguiente := 2;
END_IF;

IF (S.Etapa = 2) AND D.EmpPre AND D.CamadaEnPrecarga THEN
   S.EtapaSiguiente := 3;
   dbMemorias.EmpujadorAvanzando:= FALSE;
END_IF;

IF (S.Etapa = 3) AND (dbSecRobot.Etapa=4) THEN
   S.EtapaSiguiente := 4;
END_IF;

IF (S.Etapa = 4) AND  D.TajaderaSal_ir THEN
   S.EtapaSiguiente := 5;
```

```
END_IF;

IF (S.Etapa = 5) AND D.EmpAva THEN
   S.EtapaSiguiente := 6;
END_IF;

IF (S.Etapa = 6) AND D.CamadaEnPinza AND NOT D.CamadaEnPrecarga THEN
   S.EtapaSiguiente := 7;
END_IF;

IF (S.Etapa = 7) AND D.Cuchilla_ia THEN
   S.EtapaSiguiente := 8;
END_IF;

IF (S.Etapa = 8) AND D.EmpPre2 THEN
   S.EtapaSiguiente := 9;
END_IF;

IF (S.Etapa = 9) THEN
  IF D.Emp_ia AND dbMemorias.EmpujadorEnPrecarga THEN
    S.EtapaSiguiente := 11;
  ELSIF NOT dbMemorias.EmpujadorEnPrecarga THEN
    S.EtapaSiguiente := 10;
  END_IF;
END_IF;

IF (S.Etapa = 10) AND D.Emp_ia THEN
   S.EtapaSiguiente := 11;
END_IF;

IF (S.Etapa = 11) AND D.EmpRet THEN
   S.EtapaSiguiente := 12;
END_IF;
```

```
IF (S.Etapa = 12) AND D.EmpRet AND D.TajaderaSal_ia THEN
   S.EtapaSiguiente := 13;
END_IF;

IF (S.Etapa = 13) THEN
  IF S.EnCiclo THEN
    S.EtapaSiguiente := 1;
  ELSE
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

END_IF;

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

//Posterior
//------------------------------------------------------------------------

END_FUNCTION
```

fcSecTransportesMesa (FC 125)

```
FUNCTION fcSecTransportesMesa : VOID
VAR_IN_OUT
   S : tdSecuencia;         //Instancia de la Secuencia
   D : tdTransportesMesa;   //Instancia de los Elementos de los Transportes de la mesa
END_VAR

BEGIN
//Preliminar
//------------------------------------------------------------------------
```

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

//Acciones a la Activación
//---------------------------------------------------------------------------



//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.M04_Marcha := (S.Etapa= 7);
 D.M04_Paro   := (S.Etapa= 9);

 D.M05_Marcha := (S.Etapa= 6);
 D.M05_Paro   := (S.Etapa=10);

 D.M06_Marcha := (S.Etapa= 5);
 D.M06_Paro   := (S.Etapa=11);

 D.M07_Marcha := (S.Etapa= 4);
 D.M07_Paro   := (S.Etapa=12);
```

```
 D.M08_Marcha := (S.Etapa= 3);
 D.M08_Paro   := (S.Etapa=13);

 D.M09_Marcha := (S.Etapa= 2);
 D.M09_Paro   := (S.Etapa=14);

//Transiciones
//------------------------ ----------------------------------------------------
//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) AND (D.SecDes1 AND D.SecDes2 AND D.SecEmp) THEN
  S.EtapaSiguiente := 2;
END_IF;

IF (S.Etapa = 2) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 3;
  END_IF;
END_IF;

IF (S.Etapa = 3) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 4;
  END_IF;
```

```
END_IF;

IF (S.Etapa = 4) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 5;
  END_IF;
END_IF;

IF (S.Etapa = 5) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 6;
  END_IF;
END_IF;

IF (S.Etapa = 6) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 7;
  END_IF;
END_IF;

IF (S.Etapa = 7) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 8;
  END_IF;
END_IF;

IF (S.Etapa = 8) AND (NOT S.EnCiclo) THEN
    S.EtapaSiguiente := 9;
END_IF;

IF (S.Etapa = 9) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 10;
  END_IF;
```

```
END_IF;

IF (S.Etapa =10) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 11;
  END_IF;
END_IF;

IF (S.Etapa =11) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 12;
  END_IF;
END_IF;

IF (S.Etapa =12) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 13;
  END_IF;
END_IF;

IF (S.Etapa =13) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 14;
  END_IF;
END_IF;

IF (S.Etapa =14) THEN
  IF S.TiempoEtapa>35 THEN
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

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

```
//Posterior
//------------------------------------------------------------------------------

END_FUNCTION
```

fcSecTajaderaEnt (FC 126)

```
FUNCTION fcSecTajaderaEnt : VOID
VAR_IN_OUT
   S : tdSecuencia;    //Instancia de la Secuencia
   D : tdTajaderaEnt;   //Instancia de los Elementos de Empujador
END_VAR

BEGIN
//Preliminar
//------------------------------------------------------------------------------

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

//Acciones a la Activación
//------------------------------------------------------------------------------


//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.Tajadera_aa := (S.Etapa=3); //C06
 D.Tajadera_ar := (S.Etapa=5); //C06

//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;      //analisis del punto de entrada
   END_IF;
END_IF;

IF S.Etapa = 1 THEN
   IF  NOT DBMemorias.CamadaEmpujador THEN
      S.EtapaSiguiente := 2;
   ELSIF DBMemorias.CamadaEmpujador THEN
      S.EtapaSiguiente := 4;
   END_IF;
END_IF;

IF (S.Etapa = 2) AND DBMemorias.PenultimaTemp AND D.PenultimaCamada1 AND
(NOT D.FinCamada) THEN
```

```
   S.EtapaSiguiente := 3;
END_IF;

IF (S.Etapa = 3) AND D.Tajadera_ia THEN
   S.EtapaSiguiente := 4; //La tajadera de Entrada mesa ya ha bajado
END_IF;

IF (S.Etapa = 4) AND NOT DBMemorias.CamadaEmpujador THEN
   S.EtapaSiguiente := 5;
END_IF;

IF (S.Etapa = 5) AND D.Tajadera_ir THEN
   S.EtapaSiguiente := 6;
END_IF;

IF (S.Etapa = 6) THEN
  IF (S.EnCiclo) THEN
    S.EtapaSiguiente := 2;
  ELSE
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

END_IF;
//Acciones a la Desactivación
//-------------------------------------------------------------------------
IF (S.Etapa = 3) AND (S.EtapaSiguiente = 4) THEN
   DBMemorias.CamadaEmpujador := true;
   dbMemorias.CamadaCompleta := false;
END_IF;

//Posterior
//-------------------------------------------------------------------------
```

```
END_FUNCTION
```

fcSecAlmacenCartones (FC 129)

```
FUNCTION fcSecAlmacenCartones : VOID
VAR_IN_OUT
   S : tdSecuencia;         //Instancia de la Secuencia
   D : tdAlmacenCartones;   //Instancia de los Elementos del Almacen de cartones
END_VAR

BEGIN
//Preliminar
//-------------------------------------------------------------------------

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

//Acciones a la Activación
//-------------------------------------------------------------------------

//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.Marcha := (S.Etapa=5);
D.MarchaInv := (S.Etapa=8);
D.Paro := (S.Etapa=6) OR (S.Etapa=9);

D.Dientes_aa := (S.Etapa=4);
D.Dientes_ar := (S.Etapa=7);

//Transiciones
//-----------------------------------------------------------------------------
//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.HayCarton) THEN
    S.EtapaSiguiente := 2;
  ELSIF (D.NoHayCarton) THEN
    S.EtapaSiguiente := 3;
  END_IF;
END_IF;

IF (S.Etapa = 2) AND D.RobotFcc THEN
   S.EtapaSiguiente := 3;
END_IF;
```

```
IF (S.Etapa = 3) AND (NOT D.RobotPcc) THEN
   S.EtapaSiguiente := 4;
END_IF;

IF (S.Etapa = 4) AND (D.Dientes_ia) THEN
   S.EtapaSiguiente := 5;
END_IF;

IF (S.Etapa = 5) AND (D.Cabezal_ia) THEN
   S.EtapaSiguiente := 6;
END_IF;

IF (S.Etapa = 6) AND (D.HayCarton) THEN
   S.EtapaSiguiente := 7;
END_IF;

IF (S.Etapa = 7) AND (D.Dientes_ir) THEN
   S.EtapaSiguiente := 8;
END_IF;

IF (S.Etapa = 8) AND (D.Cabezal_ir) THEN
   S.EtapaSiguiente := 9;
END_IF;

IF (S.Etapa = 9) THEN
  IF S.EnCiclo THEN
    S.EtapaSiguiente := 1;
  ELSE
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

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

```
//Posterior
//----------------------------------------------------------------------------

END_FUNCTION
```

fcSecZonaPaletizado (FC 131)

```
FUNCTION fcSecZonaPaletizado : VOID
VAR_IN_OUT
   S : tdSecuencia;        //Instancia de la Secuencia
   D : tdZonaPaletizado;   //Instancia de los Elementos de la Zona de paletizado
END_VAR

BEGIN
//Preliminar
//----------------------------------------------------------------------------

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

//Acciones a la Activación
//----------------------------------------------------------------------------

//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.Marcha := ((S.Etapa=2 OR S.Etapa=3) AND (D.coINpalet)) OR (S.Etapa=9);
D.Paro := (S.Etapa=4) OR (S.Etapa=10);

D.Centrador_aa := (S.Etapa=4);
D.Centrador_ar := (S.Etapa=7);

D.Tope_aa := (S.Etapa=10);
D.Tope_ar := (S.Etapa=8);

D.coOUTsalida := (S.Etapa=6) OR (S.Etapa=7) OR (S.Etapa=8) OR (S.Etapa=9);
D.coOUTautPalet := (S.Etapa=2);

//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 (dbMemorias.zpSinPalet) THEN
```

```
    S.EtapaSiguiente := 2;
  ELSIF (dbMemorias.zpConPalet) THEN
    S.EtapaSiguiente := 5;
  END_IF;
END_IF;

IF (S.Etapa = 2) AND D.Dpp THEN
    S.EtapaSiguiente := 3;
END_IF;

IF (S.Etapa = 3) AND (S.TiempoEtapa>10) THEN
    S.EtapaSiguiente := 4;
END_IF;

IF (S.Etapa = 4) AND (D.Centrador_ia) THEN
    S.EtapaSiguiente := 5;
END_IF;

IF (S.Etapa = 5) AND (dbMemorias.PaletCompletado OR dbMemorias.bLiberarPalet)
THEN
    S.EtapaSiguiente := 6;
END_IF;

IF (S.Etapa = 6) AND (D.coINautSalida) THEN
    S.EtapaSiguiente := 7;
END_IF;

IF (S.Etapa = 7) AND (D.Centrador_ir) THEN
    S.EtapaSiguiente := 8;
END_IF;

IF (S.Etapa = 8) AND (D.Tope_ir) THEN
    S.EtapaSiguiente := 9;
END_IF;
```

```
IF (S.Etapa = 9) AND (NOT D.Dpp) AND (S.TiempoEtapa>15) AND (SQ3042) THEN
   S.EtapaSiguiente := 10;
END_IF;

IF (S.Etapa = 10) AND (NOT D.Tope_ia) THEN
   S.EtapaSiguiente := 11;
END_IF;

IF (S.Etapa =11) THEN
  IF S.EnCiclo THEN
    S.EtapaSiguiente := 2;
  ELSE
    S.EtapaSiguiente := 0;
  END_IF;
END_IF;

END_IF;
//Acciones a la Desactivación
//-------------------------------------------------------------------------
IF (S.Etapa = 9) AND (S.EtapaSiguiente = 10) THEN
    DBMemorias.PaletCompletado := false;
    DBMemorias.bLiberarPalet := false;
    dbRobot.cn := 0;
END_IF;

//Posterior
//-------------------------------------------------------------------------

END_FUNCTION
```

fcVLTDir (FC 200)

```
FUNCTION fcVLTDir : WORD
```

```
VAR_INPUT
  dir: INT;   //Direccion del esclavo profibus
END_VAR
VAR_TEMP
  DirVar: ARRAY[1..22] OF WORD;
END_VAR
 // Mapeo de las direcciones de periferia con el nº de esclavo
  DirVar[01] := 16#0000;
  DirVar[02] := 16#0000;
  DirVar[03] := 16#0000;
  DirVar[04] := INT_TO_WORD(1000);
  DirVar[05] := INT_TO_WORD(1012);
  DirVar[06] := INT_TO_WORD(1024);
  DirVar[07] := INT_TO_WORD(1036);
  DirVar[08] := INT_TO_WORD(1048);
  DirVar[09] := INT_TO_WORD(1060);
  DirVar[10] := 16#0000;
  DirVar[11] := 16#0000;
  DirVar[12] := 16#0000;
  DirVar[13] := INT_TO_WORD(1072);
  DirVar[14] := INT_TO_WORD(1084);
  DirVar[15] := INT_TO_WORD(1096);
  DirVar[16] := INT_TO_WORD(1108);
  DirVar[17] := INT_TO_WORD(1120);
  DirVar[18] := INT_TO_WORD(1132);
  DirVar[19] := INT_TO_WORD(1144);
  DirVar[20] := INT_TO_WORD(1156);
  DirVar[21] := INT_TO_WORD(1168);
  DirVar[22] := 16#0000;

  fcVLTDir := DirVar[dir];
END_FUNCTION
```

fcVLTRun (FC 202)

```
FUNCTION fcVLTRun : INT
VAR_INPUT
  dir: INT;
  vel: WORD;
END_VAR
VAR_TEMP
   error: INT;
END_VAR

  error := 0;

  MW810 := W#16#0000;
  MW812 := W#16#0000;
  MW814 := W#16#0000;
  MW816 := W#16#0000;
  MW818 := W#16#047F;
  MW820 := vel;

  error := fcVLTCmd(dir:=dir, esc:=true, lec:=true);

  fcVLTRun := error;
END_FUNCTION
```

fcVLTStop (FC 203)

```
FUNCTION fcVLTStop : INT
VAR_INPUT
  dir: INT;
END_VAR
VAR_TEMP
   error: INT;
END_VAR
```

```
  error := 0;

  MW810 := W#16#0000;
  MW812 := W#16#0000;
  MW814 := W#16#0000;
  MW816 := W#16#0000;
  MW818 := W#16#043F;
  MW820 := W#16#0000;

  error := fcVLTCmd(dir:=dir, esc:=true, lec:=true);

  fcVLTStop := error;
END_FUNCTION
```

fcVLTRunI (FC 204)

```
FUNCTION fcVLTRunI : INT
VAR_INPUT
  dir: INT;
  vel: WORD;
END_VAR
VAR_TEMP
   error: INT;
END_VAR

  error := 0;

  MW810 := W#16#0000;
  MW812 := W#16#0000;
  MW814 := W#16#0000;
  MW816 := W#16#0000;
  MW818 := W#16#847F;
  MW820 := vel;
```

```
  error := fcVLTCmd(dir:=dir, esc:=true, lec:=true);


    fcVLTRunI := error;
END_FUNCTION
```