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
```