[redacted]

[redacted]

[redacted]

Title:	SIARM - Control de servomotor
Author:	Alfredo Rodríguez Magdalena
Version:	3.0
Description:	Programa destinado a controlar una de las dos partes del sistema de inyección de adhesivo dentro del proyecto S.I.A.R.M. Para leer detalles de la versión, ir al programa MAIN, y leer las primeras líneas de comentarios.

```
0001 PROGRAM MAIN
0002 VAR
0003 END_VAR
0001 (*
0002  ================================================================================================
0003  Nombre      : PROGRAMA MAESTRO DE GOBIERNO DE SERVOMOTOR
0004  Autor       :  Alfredo Rodríguez Magdalena
0005  Version     : 3.0
0006  Copyright   : Normagrup Technology (c)
0007  Descripcion : Implementación de PLC como unidad intermediaria entre robot de ABB y servomotor de SMC
0008                  para proyecto S.I.A.R.M
0009  Entidad     : Normagrup Technology - Departamento de Ingenieria
0010  ================================================================================================
0011  *)
0012
0013 (*
0014 Se encargará de decidir el movimiento a ejecutar por el servomotor en función del entorno:
0015       En caso de (inicialización), recambio de bote o situación de emergencia -> Volver al origen -> IN<5:0>=0b000000
0016       En caso de flanco de subida en señal de marcha -> Inyectar adhesivo -> IN<5:0>=0b000001
0017       En caso de flanco de bajada en señal de marcha -> Retroceder bástago para detener inyección -> IN<5:0>=0b000010
0018
0019 Se hará una aplicación directa de la guía GEMMA de autómatas programables.
0020 *)
0021
0022 (*
0023 ========================== NOTAS DE VERSIÓN 2.0 ==========================
0024  - Implementado un mejor sistema de control de excepciones, capaz de lidiar con
0025    seta de emergencia y alarmas de la controladora del servomotor.
0026
0027  - Sustituido programa esclavo de control del servomotor en ST por uno en GRAFCET.
0028 ==========================================================================
0029  *)
0030
0031 (*
0032 ========================== NOTAS DE VERSIÓN 3.0 ==========================
0033  - Implementado control de velocidad de empuje del servomotor por control numérico.
0034
0035  - Sistema preparado para ser controlado por agente remoto externo (pyads)
0036 ==========================================================================
0037  *)
0038
0039
0040 (* ---------------------------------------------- SECCIÓN PRELIMINAR ---------------------------------------------- *)
0041
0042 (* Inicialización del programa *)
0043       IF Mx_Inicio THEN
0044             QxS_Output_Port.11:=TRUE; (* Se resetea el bus de comunicaciones para evitar conflictos *)
0045             QxS_Output_Port.10:=FALSE; (* El bit DRIVE se pone a 0 para evitar errores de inicialización *)
0046             QxS_Output_Port.9:=FALSE; (* SVON en estado bajo -> Se apaga el servo para iniciar proceso de arranque *)
0047             (* A continuación, se iniciará el proceso de arranque en condiciones de seguridad del sistema *)
0048             Mx_A6_PuestaEnCI:=TRUE;
0049             Mx_Inicio:=FALSE; (* Con esta instrucción se asegura que todo lo ejecutado en este
0050                                  condicional se haga al principio del programa *)
0051       END_IF;
0052
0053 (* Auxiliar *)
0054       Mx_Recambiar:=(Ix_Recambiar AND NOT Ix_LocDis) OR (IxD_Recambiar AND Ix_LocDis);
0055       Mx_Rearme:=(Ix_Rearme AND NOT Ix_LocDis) OR (IxD_Recambiar AND Ix_LocDis);
0056       Mx_FlagEmergencia:=NOT(Ix_SetaEmer); (* Se recuerda que la seta de emergencia es NC *)
0057       (* LOS BITS IN2, IN3, IN4 E IN5 ESTARÁN SIEMPRE EN ESTADO BAJO, NO SE USARÁN *)
0058       QxS_Output_Port.2:=FALSE;
0059       QxS_Output_Port.3:=FALSE;
0060       QxS_Output_Port.4:=FALSE;
0061       QxS_Output_Port.5:=FALSE;
0062
0063 (* Objetos*)
0064       (* Detectores de flancos *)
```

```
0065        Ready_R_TRIG(CLK:=Ix_Ready);
0066        Ready_F_TRIG(CLK:=Ix_Ready);
0067        Recambiar_R_TRIG(CLK:=Mx_Recambiar);
0068        Rearme_R_TRIG(CLK:=Mx_Rearme);
0069        Emergencia_F_TRIG(CLK:=Mx_FlagEmergencia);
0070        Pasta_F_TRIG(CLK:=TON_TemporizaProceso.IN);
0071        (* Temporizadores *)
0072        TON_ApagaServo(IN:=((Mx_A1_ParadaEnCI OR Mx_A4_ParadaObtenida) AND IxS_Info.9=1),PT:=t#15s);
0073        TON_1s(IN:=NOT(TON_1s.Q),PT:=T#1s); (* Este temporizador no debe dejar de hacer temporizaciones nunca *)
0074        TON_Force(IN:=IxS_CurrentForce>85,PT:=t#500ms);
0075        TON_TemporizaProceso(PT:=T#60s); (* Se pone un tiempo lo suficientemente grande como para que no sature *)
0076        TON_Recambio_Dist(IN:=IxD_Recambiar,PT:=t#500ms);
0077        TON_Rearme_Dist(IN:=IxD_Rearme,PT:=t#500ms);
0078
0079    (* Osciladores *)
0080        IF TON_1s.ET<=t#500ms THEN
0081            Mx_Oscilador_1Hz:=TRUE;
0082        ELSE
0083            Mx_Oscilador_1Hz:=FALSE;
0084        END_IF;
0085
0086    (* Tratamiento de datos *)
0087        (* Cálculo de nivel de bote + Gestion de temporizador *)
0088        IF (IxS_Info.8=1 AND Ix_Ready AND TON_Force.Q AND Mx_F1_Automatico) THEN
0089            TON_TemporizaProceso.IN:=TRUE;
0090        ELSE
0091            TON_TemporizaProceso.IN:=FALSE;
0092        END_IF;
0093
0094    (* Tratamiento de variables a distancia *)
0095        (* IxD_Recambiar *)
0096        IF TON_Recambio_Dist.Q THEN
0097            IxD_Recambiar:=FALSE;
0098        END_IF;
0099        (* IxD_Rearmer *)
0100        IF TON_Rearme_Dist.Q THEN
0101            IxD_Rearme:=FALSE;
0102        END_IF;
0103    (* ---------------------------------------------- A1 - PARADA EN CI  ---------------------------------------------- *)
0104
0105    (* El programa asegura que no haya ningún tipo de movimiento no deseado en el servomotor *)
0106
0107    IF Mx_A1_ParadaEnCI AND TON_ApagaServo.Q THEN
0108        QxS_Output_Port.9:=FALSE; (* SVON en estado bajo -> Se apaga el servo para alargar vida útil y ahorrar energía *)
0109    END_IF;
0110
0111
0112    (* ---------------------------------------------- F1 - PRODUCCIÓN NORMAL  ---------------------------------------------- *)
0113
0114    (* SECCIÓN DE GOBIERNO DE PROGRAMA ESCLAVO *)
0115
0116    IF (Mx_A1_ParadaEnCI OR  Mx_A4_ParadaObtenida) AND Ix_Ready THEN
0117        SR_BorraEstados();
0118        Mx_F1_Automatico:=TRUE;
0119        QxS_Numerical_Data.11:=TRUE; (* Se activa el bit de control numérico de la velocidad de empuje *)
0120        QxS_Numerical_Data.5:=TRUE; (* Se activa el bit de control numérico de la velocidad máxima *)
0121        QxS_Output_Port.9:=TRUE; (* Se pone en funcionamiento el servomotor *)
0122        (* Se selecciona el movimiento de retroceso de vástago IN<1:0>=1d=0b01 *)
0123        QxS_Output_Port.0:=TRUE;
0124        QxS_Output_Port.1:=FALSE;
0125        Mx_Move_0:=TRUE;
0126    END_IF;
0127
0128    IF Mx_F1_Automatico AND Mx_Move_0 THEN
0129        (* Se envia la orden de mover el vástago al programa esclavo *)
0130        SFCInit:=FALSE;
0131        QxS_Pushing_Speed:=Mx_VelocidadEmpuje;
0132        QxS_Speed:=Mx_VelocidadEmpuje;
```

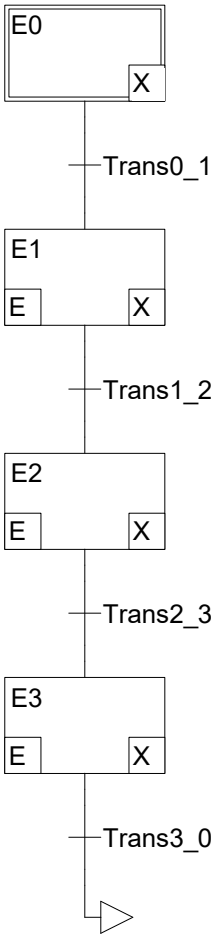
```
END_IF;

(* Si se detecta un flanco de bajada en el estado marcha, se apagará el servo,
y se ejecutará un movimiento de retroceso de 3mm hacia atrás *)
IF Mx_F1_Automatico AND NOT Ix_Ready AND Mx_Move_2 THEN
    Mi_NivelBote:=(21100.0-IxS_Current_Position)/21100.0*100.0;
    Mx_F1_Automatico:=FALSE;
    QxS_Numerical_Data:=0;
    QxS_Start_Flag.0:=FALSE;
    Mx_A3_ParadaPedida:=TRUE;
    QxS_Output_Port.11:=TRUE; (* Paramos el avance del vástago con el bit RESET *)
    SR_ReseteaMovimiento();
    SFCInit:=TRUE;
END_IF;

(* Si el movimiento se termina, es decir, se acaba el bote de adhesivo, se cambia al estado de error D2 *)
IF Mx_F1_Automatico AND Mx_Move_3 THEN
    (* Se resetea el GRAFCET para poder hacer otros movimientos en el futuro *)
    SR_ReseteaMovimiento();
    SFCInit:=TRUE;
    (* Transicion al siguiente estado *)
    QxS_Numerical_Data:=0;
    Mx_F1_Automatico:=FALSE;
    Mx_D2_TratamientoErrores:=TRUE;
    Mx_D2_Error_1:=TRUE;
    QxS_Start_Flag.0:=FALSE;
END_IF;

(* ---------------------------------------------- D1 - PARADA DE EMERGENCIA ---------------------------------------------- *)

IF Mx_FlagEmergencia THEN
    SR_BorraEstados();
    Mx_D1_Emergencia:=TRUE;
    QxS_Output_Port.9:=FALSE; (* Se apaga el servo por seguridad *)
END_IF;

IF Emergencia_F_TRIG.Q AND Mx_D1_Emergencia THEN
    SR_BorraEstados();
    Mx_A5_PreparacionPuestaCI:=TRUE;
    SR_ReseteaMovimiento();
    Mx_Move_0:=TRUE;
    SFCInit:=TRUE;
END_IF;



(* ---------------------------------- D2 - DIAGNÓSTICO Y/O TRATAMIENTO DE ERRORES ---------------------------------- *)

(* ERROR TIPO 1 -> BOTE VACÍO *)

    IF Mx_D2_TratamientoErrores AND Recambiar_R_TRIG.Q AND Mx_D2_Error_1 THEN
        QxS_Output_Port.9:=TRUE; (* Se pone en funcionamiento el servomotor *)
        (* Se selecciona el movimiento de retorno rápido al origen IN<1:0>=0d=0b00 *)
        QxS_Output_Port.0:=FALSE;
        QxS_Output_Port.1:=FALSE;
        Mx_Move_0:=TRUE;
    END_IF;

    IF Mx_D2_TratamientoErrores AND Mx_Move_0 AND Mx_D2_Error_1 THEN
        SFCInit:=FALSE;
    END_IF;

    IF Mx_D2_TratamientoErrores AND Mx_Move_3 AND Mx_D2_Error_1 THEN
        (* Se resetea el GRAFCET para poder hacer otros movimientos en el futuro *)
        SR_ReseteaMovimiento();
        Mx_Move_0:=TRUE;
        SFCInit:=TRUE;
        (* Transición de estado *)
```

```
0201            Mx_D2_TratamientoErrores:=FALSE;
0202            Mx_A5_PreparacionPuestaCI:=TRUE;
0203            Mx_D2_Error_1:=FALSE;
0204        END_IF;
0205
0206 (* ERROR TIPO 2 -> ALARMA GENERADA DESDE LA CONTROLADORA *)
0207
0208        IF IxS_Info.15 THEN
0209            SR_BorraEstados();
0210            Mx_D2_TratamientoErrores:=TRUE;
0211            Mx_D2_Error_2:=TRUE;
0212            QxS_Output_Port.11:=TRUE; (* El bit reset se pone en estado alto para resetear las alarmas *)
0213        END_IF;
0214
0215        IF Mx_D2_TratamientoErrores AND NOT IxS_Info.15 AND Mx_D2_Error_2 THEN
0216            Mx_D2_TratamientoErrores:=FALSE;
0217            Mx_D2_Error_2:=FALSE;
0218            QxS_Output_Port.11:=FALSE;
0219            Mx_A5_PreparacionPuestaCI:=TRUE;
0220        END_IF;
0221
0222 (* ---------- A5 - PREPARACIÓN DE PUESTA EN CI ---------- *)
0223
0224 IF Mx_A5_PreparacionPuestaCI AND Rearme_R_TRIG.Q THEN
0225        Mx_A6_PuestaEnCI:=TRUE;
0226        Mx_A5_PreparacionPuestaCI:=FALSE;
0227 END_IF;
0228
0229 (* ---------- A6 - PUESTA EN CI ---------- *)
0230
0231 IF Mx_A6_PuestaEnCI THEN
0232        QxS_Output_Port.11:=FALSE; (* Se apaga el bit de reseteo del bus de comunicaciones *)
0233 END_IF;
0234
0235 IF Mx_A6_PuestaEnCI AND Mx_Move_0 THEN
0236        SFCInit:=FALSE;
0237        QxS_Output_Port.9:=TRUE; (* Se pone en funcionamiento el servomotor *)
0238 END_IF;
0239
0240 IF Mx_A6_PuestaEnCI AND Mx_Move_3 THEN
0241        (* Se resetea el GRAFCET para poder hacer otros movimientos en el futuro *)
0242        SR_ReseteaMovimiento();
0243        Mx_Move_0:=TRUE;
0244        SFCInit:=TRUE;
0245        (* Transicion al siguiente estado *)
0246        Mx_A6_PuestaEnCI:=FALSE;
0247        Mx_A1_ParadaEnCI:=TRUE;
0248 END_IF;
0249
0250 (* ---------- A3 - PARADA PEDIDA EN ESTADO INTERMEDIO ---------- *)
0251
0252 IF Mx_A3_ParadaPedida AND NOT IxS_Info.8 THEN
0253        QxS_Output_Port.11:=FALSE; (* Se apaga el bit RESET para configurar el movimiento de retroceso *)
0254        (* Se selecciona el movimiento de retroceso de vástago IN<1:0>=2d=0b10 *)
0255        QxS_Output_Port.0:=FALSE;
0256        QxS_Output_Port.1:=TRUE;
0257        Mx_Move_0:=TRUE; (* Se prepara el inicio del movimiento correspondiente *)
0258 END_IF;
0259
0260 IF Mx_A3_ParadaPedida AND Mx_Move_0 THEN
0261        (* Se envia la orden de mover el vástago al programa esclavo *)
0262        SFCInit:=FALSE;
0263 END_IF;
0264
0265 (* ---------- A4 - PARADA OBTENIDA ---------- *)
0266
0267 IF Mx_A3_ParadaPedida AND Mx_Move_3 THEN
0268        Mx_Move_0:=TRUE;
```

```
0269        SR_ReseteaMovimiento();
0270        Mx_A3_ParadaPedida:=FALSE;
0271        Mx_A4_ParadaObtenida:=TRUE;
0272        SFCInit:=TRUE;
0273    END_IF;
0274
0275    IF Mx_A4_ParadaObtenida AND TON_ApagaServo.Q THEN
0276        QxS_Output_Port.9:=FALSE; (* SVON en estado bajo -> Se apaga el servo para alargar vida útil y ahorrar energía *)
0277    END_IF;
```

MOVE_SERVO (PRG-SFC)

```
0001    PROGRAM MOVE_SERVO
0002    VAR
0003    END_VAR
0004
0005    (*
0006     ============================================================================================================
0007     Nombre : PROGRAMA ESCLAVO DE GOBIERNO DE SERVOMOTOR
0008              SECUENCIA PARA MOVIMIENTO DE VÁSTAGO
0009     Autor      :  Alfredo Rodríguez Magdalena
0010     Version    : 3.0
0011     Copyright  : Normagrup Technology (c)
0012     Descripcion : Implementación de PLC como unidad intermediaria entre robot de ABB y servomotor de SMC
0013                   para proyecto S.I.A.R.M
0014     Entidad    : Normagrup Technology - Departamento de Ingenieria
0015     ============================================================================================================
0016    *)
0017
0018    (* Este programa se encargará de transferir el movimiento decidido por el programa maestro *)
```



MOVE_SERVO (PRG-SFC).Action E0 - Exit (ST)

```
0001    Mx_Move_0:=FALSE;
```

MOVE_SERVO (PRG-SFC).Action E1 - Entry (ST)

```
0001    (* ---------- E1 - COMENZAR MOVIMIENTO ---------- *)
0002    IF Mx_A6_PuestaEnCI THEN
0003        QxS_Output_Port.12:=TRUE; (* SETUP=1 -> Se inicia la operación de retorno al origen para establecer el eje de coordenadas *)
```

```
0004
0005 ELSIF Mx_F1_Automatico THEN
0006       QxS_Start_Flag.0:=TRUE; (* El bit de control de servomotor por entradas numéricas se pone en estado alto *)
0007
0008 ELSE
0009       QxS_Output_Port.10:=TRUE; (* El bit de control DRIVE se pone en estado alto para comenzar el movimiento *)
0010 END_IF;
0011 Mx_Move_1:=TRUE;
```

MOVE_SERVO (PRG-SFC).Action E1 - Exit (ST)

```
0001 Mx_Move_1:=FALSE;
```

MOVE_SERVO (PRG-SFC).Action E2 - Entry (ST)

```
0001 IF Mx_A6_PuestaEnCI THEN
0002       QxS_Output_Port.12:=FALSE; (* SETUP=0 -> Por decisión del fabricante, este bit debe volver a ponerse a 0 *)
0003
0004 ELSIF Mx_F1_Automatico THEN
0005       QxS_Start_Flag.0:=FALSE; (* El bit de control de servomotor por entradas numéricas se pone en estado bajo *)
0006
0007 ELSE
0008       QxS_Output_Port.10:=FALSE; (* El bit de control DRIVE se pone en estado bajo por decision del fabricante *)
0009 END_IF;
0010 Mx_Move_2:=TRUE;
```

MOVE_SERVO (PRG-SFC).Action E2 - Exit (ST)

```
0001 Mx_Move_2:=FALSE;
```

MOVE_SERVO (PRG-SFC).Action E3 - Entry (ST)

```
0001 Mx_Move_3:=TRUE;
```

MOVE_SERVO (PRG-SFC).Action E3 - Exit (ST)

```
0001 Mx_Move_3:=FALSE;
```

SALIDAS (PRG-ST)

```
0001 PROGRAM SALIDAS
0002 VAR
0003 END_VAR
```

```
0001 (*
0002  ========================================================================================================
0003  Nombre     : PROGRAMA DE GOBIERNO DE SALIDAS DEL PLC
0004  Autor      :  Alfredo Rodríguez Magdalena
0005  Version    : 2.0
0006  Copyright  : Normagrup Technology (c)
0007  Descripcion : Implementación de PLC como unidad intermediaria entre robot de ABB y servomotor de SMC
0008                  para proyecto S.I.A.R.M
0009  Entidad    : Normagrup Technology - Departamento de Ingenieria
0010  ========================================================================================================
0011  *)
0012
0013 (* READY *)
0014 IF Mx_A1_ParadaEnCI OR Mx_F1_Automatico OR Mx_A3_ParadaPedida OR Mx_A4_ParadaObtenida THEN
0015       Qx_Ready:=TRUE;
0016 ELSE
0017       Qx_Ready:=FALSE;
0018 END_IF;
0019
0020 (* FAIL *)
0021 IF Mx_D1_Emergencia OR Mx_D2_TratamientoErrores THEN
0022       Qx_Fail:=TRUE;
0023 ELSE
0024       Qx_Fail:=FALSE;
0025 END_IF;
0026
0027 (* PASS *)
0028 IF (IxS_Info.8=1 AND Ix_Ready AND TON_Force.Q AND Mx_F1_Automatico) THEN
0029       Qx_Pass:=TRUE;
0030 ELSE
0031       Qx_Pass:=FALSE;
0032 END_IF;
0033
```

```
0034 (* TESTIGO LUMINOSO DE COLOR ROJO *)
0035 IF Mx_D2_TratamientoErrores AND Mx_D2_Error_2 THEN
0036       Qx_Rojo:=Mx_Oscilador_1Hz;
0037       Mi_EstadoSistema:=3;
0038 ELSIF Mx_D1_Emergencia THEN
0039       Qx_Rojo:=TRUE;
0040       Mi_EstadoSistema:=3;
0041 ELSE
0042       Qx_Rojo:=FALSE;
0043 END_IF;
0044
0045 (* TESTIGO LUMINOSO DE COLOR ÁMBAR *)
0046
0047 IF Mx_D2_TratamientoErrores AND Mx_D2_Error_1 THEN
0048       Qx_Ambar:=TRUE;
0049       Mi_EstadoSistema:=2;
0050 ELSE
0051       Qx_Ambar:=FALSE;
0052 END_IF;
0053
0054 (* TESTIGO LUMINOSO DE COLOR VERDE *)
0055
0056 IF Mx_A1_ParadaEnCI OR Mx_F1_Automatico OR Mx_A3_ParadaPedida OR Mx_A4_ParadaObtenida THEN
0057       Qx_Verde:=TRUE;
0058       Mi_EstadoSistema:=1;
0059 ELSE
0060       Qx_Verde:=FALSE;
0061 END_IF;
```

SR_BorraEstados (FUN-ST)

```
0001 FUNCTION SR_BorraEstados : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005 END_VAR
```

```
0001 (* Subrutina encargada de desactivar todos los estados del proceso *)
0002
0003 Mx_A1_ParadaEnCI:=FALSE;
0004 Mx_F1_Automatico:=FALSE;
0005 Mx_D1_Emergencia:=FALSE;
0006 Mx_D2_TratamientoErrores:=FALSE;
0007 Mx_A5_PreparacionPuestaCI:=FALSE;
0008 Mx_A6_PuestaEnCI:=FALSE;
0009 Mx_A3_ParadaPedida:=FALSE;
0010 Mx_A4_ParadaObtenida:=FALSE;
```

SR_ReseteaMovimiento (FUN-ST)

```
0001 FUNCTION SR_ReseteaMovimiento : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005 END_VAR
```

```
0001 Mx_Move_0:=FALSE;
0002 Mx_Move_1:=FALSE;
0003 Mx_Move_2:=FALSE;
0004 Mx_Move_3:=FALSE;
```

Transiciones (PRG-ST)

```
0001 PROGRAM Transiciones
0002 VAR
0003 END_VAR
```

```
0001 (* TRANSICION E0 -> E1 *)
0002
0003 Trans0_1:=IxS_Info.9; (* SVRE = 1 *)
0004
0005 (* TRANSICIÓN E1 -> E2 *)
0006
0007 Trans1_2:=IxS_Info.11=0 AND IxS_Info.8=1; (* INP = 0 Y BUSY = 1 *)
0008
```

```
0009 (* TRANSICIÓN E2 -> E3 *)
0010
0011 (* Si los bits de control INP y BUSY están en estado bajo y alto respectivamente se lleva a cabo la transición al E3 *)
0012 IF Mx_A6_PuestaEnCI THEN
0013     Trans2_3:=IxS_Info.11=1 AND IxS_Info.8=0 AND IxS_Info.10; (* INP = 1 Y BUSY = 0 Y SETON = 1 *)
0014 ELSE
0015     Trans2_3:=IxS_Info.8=0; (* INP = 1 Y BUSY = 0 *)
0016 (* IxS_Info.11=1 AND *)
0017 END_IF;
0018
0019 (* TRANSICIÓN E3 -> E0 *)
0020 Trans3_0:=FALSE;
```

Global_Variables

```
0001 VAR_GLOBAL
0002     (* Variables I/O *)
0003         (* Variables de entrada *)
0004         Ix_Ready AT %I*: BOOL; (* No tendrá pulsador asociado, es una señal directamente conectada al robot *)
0005                                 (* También denominado MARCHA *)
0006         Ix_Recambiar AT %I*: BOOL; (* Tiene pulsador asociado *)
0007         Ix_Rearme AT %I*: BOOL; (* Tiene pulsador asociado *)
0008         Ix_SetaEmer AT %I*: BOOL; (* Compartirá seta de emergencia con el robot, se recuerda que es un contacto NC *)
0009
0010         Ix_LocDis AT %I*: BOOL; (* Selector de control del sistema mediante pulsadores mecánicos o pantalla de operador *)
0011
0012         (* Variables de salida *)
0013         Qx_Ready AT %Q*: BOOL; (* Indica si el servo está preparado para hacer el funcionamiento normal *)
0014         Qx_Fail AT %Q*: BOOL; (* Indica si el servo está en alguna situación de error o alarma *)
0015         Qx_Pass AT %Q*: BOOL; (* Indica si el servomotor está funcionando *)
0016         Qx_Rojo AT %Q*: BOOL; (* Indicador de emergencia *)
0017         Qx_Ambar AT %Q*: BOOL; (* Indicador de fin de vida de bote de adhesivo *)
0018         Qx_Verde AT %Q*: BOOL:=TRUE; (* Indicador de funcionamiento correcto del sistema *)
0019
0020         (* Variables de entrada a distancia *)
0021         IxD_Recambiar: BOOL; (* Botón virtual de recambio de bote *)
0022         IxD_Rearme: BOOL; (* Botón virtual de rearme del sistema *)
0023
0024     (* Variables de memoria *)
0025         (* Eventos *)
0026         Mx_Recambiar: BOOL; (* Variable de memoria que almacena el boton recambiar fisico y virtual *)
0027         Mx_Rearme: BOOL; (* Variable de memoria que almacena el boton rearme fisico y virtual *)
0028
0029         (* Estados *)
0030         Mx_A1_ParadaEnCI: BOOL:=FALSE; (* Estado en el que el servomotor está listo para usarse,
0031                                           estando a la espera de órdenes.*)
0032         Mx_F1_Automatico: BOOL:=FALSE; (* Estado en el que el servomotor debe estar inyectando el adhesivo *)
0033         Mx_D1_Emergencia: BOOL:=FALSE; (* Estado de emergencia *)
0034         Mx_D2_TratamientoErrores: BOOL:=FALSE; (* Estado que indica al sistema la falta de material adhesivo
0035                                                   y otros tipos de errores *)
0036         Mx_A5_PreparacionPuestaCI: BOOL:=FALSE; (* Estado en el que se espera al rearme del sistema *)
0037         Mx_A6_PuestaEnCI: BOOL:=FALSE; (* Estado que indica que el sistema está en CI *)
0038         Mx_A3_ParadaPedida: BOOL:=FALSE; (* Estado que prepara el sistema para ser parado *)
0039         Mx_A4_ParadaObtenida: BOOL:=FALSE; (* Estado que indica que el sistema está parado *)
0040
0041         Mx_Move_0: BOOL:=TRUE; (* Indica si la etapa E0 del GRAFCET está activa *)
0042         Mx_Move_1: BOOL:=FALSE; (* Indica si la etapa E1 del GRAFCET está activa *)
0043         Mx_Move_2: BOOL:=FALSE; (* Indica si la etapa E2 del GRAFCET está activa *)
0044         Mx_Move_3: BOOL:=FALSE; (* Indica si la etapa E3 del GRAFCET está activa *)
0045
0046         (* Errores *)
0047         Mx_D2_Error_1: BOOL:=FALSE; (* Error referente a bote vacío *)
0048         Mx_D2_Error_2: BOOL:=FALSE; (* Error referente  a activación de alguna alarma del servomotor *)
0049
0050         (* Transiciones *)
0051         Trans0_1: BOOL;
0052         Trans1_2: BOOL;
0053         Trans2_3: BOOL;
0054         Trans3_0: BOOL;
0055
```

```
0056            (* Auxiliar *)
0057            Mx_FlagEmergencia: BOOL; (* Flag que usa el sistema para saber si hay una emergencia *)
0058            Mx_Inicio: BOOL:=TRUE; (* Variable que se pone a 1 sólo en el inicio del programa *)
0059            Mx_VelocidadEmpuje: UINT:=12; (* Variable auxliar donde se almacena la velocidad de empuje a asignar
0060                                             cada vez que se invoca el estado F1 *)
0061                                           (* Se pretende que el valor asignado a esta variable se cambie mediante pyads *)
0062            (* Objetos *)
0063                (* Detectores de flancos *)
0064                Ready_R_TRIG: R_TRIG; (* Se encarga de detectar flancos de subida en la señal de marcha *)
0065                Ready_F_TRIG: F_TRIG; (* Se encarga de detectar flancos de bajada en la señal de marcha *)
0066                Recambiar_R_TRIG: R_TRIG; (* Se encarga de detectar flancos de subida en la señal de recambiar *)
0067                Rearme_R_TRIG: R_TRIG; (* Se encarga de detectar flancos de subida en la señal de rearme *)
0068                Emergencia_F_TRIG: F_TRIG; (* Se encarga de detectar flancos de bajada en la señal de emergencia *)
0069                Pasta_F_TRIG: F_TRIG; (* Detecta cuando se deja de echar pasta *)
0070
0071                (* Temporizadores *)
0072                TON_ApagaServo: TON; (* Si se está en estado de parada durante X segundos,
0073                                        el temporizador apagará el servomotor *)
0074                TON_1s: TON; (* Temporizador destinado a crear infinitas temporizaciones de 1 s de duración *)
0075                TON_Force: TON; (* Temporizador destinado a evitar ruido en detección de bote *)
0076                TON_TemporizaProceso: TON; (* Temporizador encargado de calcular el tiempo
0077                                              que tarda en inyectarse una línea de adhesivo *)
0078                TON_Recambio_Dist: TON; (* Temporizador para pone variable IxD_Recabiar
0079                                           en estado bajo por software *)
0080                TON_Rearme_Dist: TON; (* Temporizador para pone variable IxD_Rearme
0081                                         en estado bajo por software *)
0082
0083            (* Osciladores *)
0084                Mx_Oscilador_1Hz: BOOL; (* Booleano que oscila entre TRUE y FALSE a 1 Hz simétricamente *)
0085
0086            (* Flags de SFC *)
0087                SFCInit: BOOL:=TRUE; (* Variable interna del PLC para resetear y poner un SFC en el estado inicial *)
0088
0089        (* Variables del servomotor *)
0090            (* Entradas *)
0091            IxS_Current_Position AT %I*: DINT; (* Devuelve la posición instantánea que tiene el servomotor *)
0092            IxS_CurrentSpeed AT %I*: UINT; (* Devuelve la velocidad instantánea que tiene el servomotor *)
0093            IxS_Info AT %I*: UINT; (* Bits de control por parte del servomotor - SOLO LECTURA *)
0094            IxS_CurrentForce AT %I*:UINT; (* Devuelve la fuerza instantánea ejercida por el vástago del servimotor *)
0095
0096            (* Salidas *)
0097            QxS_Start_Flag AT %Q*: USINT:=0; (* Este bit interesa tenerlo a 0 todo el programa, no se va a usar *)
0098            QxS_Numerical_Data AT %Q*: UINT:=0; (* Se va a emplear step data, no habrá configuración numérica de parámetros *)
0099            QxS_Output_Port AT %Q*: UINT:=0; (* Desde aquí se accede a los bits de contro por parte del PLC
0100                                                y a los de selección (IN) - SOLO ESCRITURA *)
0101            QxS_Pushing_Speed AT %Q*: UINT; (* Variable donde se va a cargar la velocidad del vástago del servomotor
0102                                               en operación de empuje *)
0103            QxS_Speed AT %Q*: UINT; (* Variable donde se va a cargar la velocidad máxima del vástago del servomotor *)
0104
0105        (* Variables de almacenaje de información *)
0106        (* El PLC tendrá la doble función de ejecutar el proceso de inyección, y de recoger información del mismo
0107        para enviarla a la pantalla de operador de streamlit *)
0108        Mt_Temp:TIME; (* Variable donde se almacena el tiempo de duración del proceso de inyección de plástico por tirada *)
0109        Mi_EstadoSistema: INT; (* Variable donde se almacenará de forma numérica el estado del sistema,
0110                                  para ser interpretada en Python *)
0111        Mi_NivelBote: REAL; (* Variable donde se almacenará el resultado del cálculo del nivel de adhesivo restante en el bote *)
0112    END_VAR
```

TwinCAT_Configuration

```
0001 (* Generated automatically by TwinCAT - (read only) *)
0002 VAR_CONFIG
0003     .Ix_Ready AT %IX8.0 : BOOL;  (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 2 (EL1809)^Channel 1^
0004     .Ix_Recambiar AT %IX8.1 : BOOL;  (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 2 (EL1809)^Chann
0005     .Ix_Rearme AT %IX8.2 : BOOL;     (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 2 (EL1809)^Chann
0006     .Ix_SetaEmer AT %IX8.3 : BOOL;   (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 2 (EL1809)^Chann
0007     .Ix_LocDis AT %IX14.0 : BOOL;    (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 2 (EL1809)^Chann
0008     .Qx_Ready AT %QX6.0 : BOOL;      (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 3 (EL2809-0015)^C
0009     .Qx_Fail AT %QX6.1 : BOOL;   (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 3 (EL2809-0015)^Chan
```

^Input} *)
el 2^Input} *)
el 3^Input} *)
el 9^Input} *)
el 16^Input} *)
Channel 1^Output} *)
nnel 2^Output} *)

```
    .Qx_Pass AT %QX6.2 : BOOL; (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 3 (EL2809-0015)^Chan
    .Qx_Rojo AT %QX6.3 : BOOL; (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 3 (EL2809-0015)^Chan
    .Qx_Ambar AT %QX6.4 : BOOL;    (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 3 (EL2809-0015)^C
    .Qx_Verde AT %QX6.5 : BOOL;    (* ~{LinkedWith:TIID^Dispositivo 1 (EtherCAT)^Terminal 1 (EK1200)^Terminal 3 (EL2809-0015)^C
    .IxS_Current_Position AT %IB0 : DINT;   (* ~{LinkedWith:TIID^Dispositivo 3 (EtherCAT)^Box 5 (JXCE1)^Transmit PDO1 Mapping^Cu
    .IxS_CurrentSpeed AT %IB16 : UINT;    (* ~{LinkedWith:TIID^Dispositivo 3 (EtherCAT)^Box 5 (JXCE1)^Transmit PDO1 Mapping^Cu
    .IxS_Info AT %IB4 : UINT;   (* ~{LinkedWith:TIID^Dispositivo 3 (EtherCAT)^Box 5 (JXCE1)^Transmit PDO1 Mapping^Signal allocated t
    .IxS_CurrentForce AT %IB6 : UINT;  (* ~{LinkedWith:TIID^Dispositivo 3 (EtherCAT)^Box 5 (JXCE1)^Transmit PDO1 Mapping^Current
    .QxS_Start_Flag AT %QB0 : USINT;(* ~{LinkedWith:TIID^Dispositivo 3 (EtherCAT)^Box 5 (JXCE1)^Receive PDO1 Mapping^Start flag
    .QxS_Numerical_Data AT %QB2 : UINT;(* ~{LinkedWith:TIID^Dispositivo 3 (EtherCAT)^Box 5 (JXCE1)^Receive PDO1 Mapping^Cor
    .QxS_Output_Port AT %QB4 : UINT;    (* ~{LinkedWith:TIID^Dispositivo 3 (EtherCAT)^Box 5 (JXCE1)^Receive PDO1 Mapping^Out
    .QxS_Pushing_Speed AT %QB8 : UINT;
    .QxS_Speed AT %QB12 : UINT;
END_VAR
```

nnel 3^Output} *)
nnel 6^Output} *)
Channel 5^Output} *)
Channel 4^Output} *)
rrent position} *)
rrent speed} *)
to the input port} *)
: pushing force} *)
g} *)
ntrolling of the numerical data flag} *)
tput port to which signals are allocated} *)

Variable_Configuration

```
0001 VAR_CONFIG
0002 END_VAR
```

Global Variables 0

```
0001 VAR_GLOBAL
0002 END_VAR
```

Alarm configuration

- Alarm configuration
  - Alarm classes
  - System

PLC Configuration

- Hardware-Configuration

Sampling Trace

No trace loaded

Task configuration

- Task configuration
  - Standard (PRIORITY := 0, INTERVAL := T#10ms)
    - MOVE_SERVO();
    - MAIN
    - Transiciones();
    - SALIDAS();

Watch- and Recipe Manager

Standard

Workspace

Parameter Manager

```
0001 Parameter-Manager
0002 ===============
```

Cross Reference List

Trans3_0		
	MOVE_SERVO (8)	Global Read
	TRANSICIONES (20)	Global Write
	_global_init (33)	Global Write
Trans2_3		
	MOVE_SERVO (6)	Global Read
	TRANSICIONES (13)	Global Write
	TRANSICIONES (15)	Global Write
	_global_init (32)	Global Write
Trans1_2		
	MOVE_SERVO (4)	Global Read
	TRANSICIONES (7)	Global Write
	_global_init (31)	Global Write
TON_Force		
	MAIN (74)	Global Write
	MAIN (88)	Global Read
	SALIDAS (28)	Global Read

_global_init (45)	Global Read
Ready_R_TRIG	
MAIN (65)	Global Write
_global_init (37)	Global Read
Trans0_1	
MOVE_SERVO (2)	Global Read
TRANSICIONES (3)	Global Write
_global_init (30)	Global Write
TON_TemporizaProceso	
MAIN (70)	Global Read
MAIN (75)	Global Write
MAIN (89)	Global Write
MAIN (91)	Global Write
_global_init (46)	Global Read
TON_Rearme_Dist	
MAIN (77)	Global Write
MAIN (100)	Global Read
_global_init (48)	Global Read
TON_Recambio_Dist	
MAIN (76)	Global Write
MAIN (96)	Global Read
_global_init (47)	Global Read
TON_ApagaServo	
MAIN (72)	Global Write
MAIN (107)	Global Read
MAIN (275)	Global Read
_global_init (43)	Global Read
TON_1s	
MAIN (73)	Global Read
MAIN (73)	Global Write
MAIN (80)	Global Read
_global_init (44)	Global Read
SFCInit	
MAIN (130)	Global Write
MAIN (145)	Global Write
MAIN (152)	Global Write
MAIN (174)	Global Write
MAIN (192)	Global Write
MAIN (199)	Global Write
MAIN (236)	Global Write
MAIN (244)	Global Write
MAIN (262)	Global Write
MAIN (272)	Global Write
MOVE_SERVO (0)	Global Read
_global_init (50)	Global Write
Recambiar_R_TRIG	
MAIN (67)	Global Write
MAIN (183)	Global Read
_global_init (39)	Global Read
QxS_Pushing_Speed%QB8	
MAIN (131)	Global Write
_global_init (58)	Global Write
QxS_Start_Flag%QB0	
MAIN (141)	Global Write
MAIN (158)	Global Write
MOVE_SERVO (6)	Global Write
MOVE_SERVO (5)	Global Write
_global_init (55)	Global Write
QxS_Speed%QB12	
MAIN (132)	Global Write
_global_init (59)	Global Write
QxS_Output_Port%QB4	
MAIN (44)	Global Write
MAIN (45)	Global Write
MAIN (46)	Global Write
MAIN (58)	Global Write
MAIN (59)	Global Write
MAIN (60)	Global Write

MAIN (61)	Global Write
MAIN (108)	Global Write
MAIN (121)	Global Write
MAIN (123)	Global Write
MAIN (124)	Global Write
MAIN (143)	Global Write
MAIN (166)	Global Write
MAIN (184)	Global Write
MAIN (186)	Global Write
MAIN (187)	Global Write
MAIN (212)	Global Write
MAIN (218)	Global Write
MAIN (232)	Global Write
MAIN (237)	Global Write
MAIN (253)	Global Write
MAIN (255)	Global Write
MAIN (256)	Global Write
MAIN (276)	Global Write
MOVE_SERVO (3)	Global Write
MOVE_SERVO (9)	Global Write
MOVE_SERVO (2)	Global Write
MOVE_SERVO (8)	Global Write
_global_init (57)	Global Write
QxS_Output_Port%QB4	
MAIN (44)	Global Write
MAIN (45)	Global Write
MAIN (46)	Global Write
MAIN (58)	Global Write
MAIN (59)	Global Write
MAIN (60)	Global Write
MAIN (61)	Global Write
MAIN (108)	Global Write
MAIN (121)	Global Write
MAIN (123)	Global Write
MAIN (124)	Global Write
MAIN (143)	Global Write
MAIN (166)	Global Write
MAIN (184)	Global Write
MAIN (186)	Global Write
MAIN (187)	Global Write
MAIN (212)	Global Write
MAIN (218)	Global Write
MAIN (232)	Global Write
MAIN (237)	Global Write
MAIN (253)	Global Write
MAIN (255)	Global Write
MAIN (256)	Global Write
MAIN (276)	Global Write
MOVE_SERVO (3)	Global Write
MOVE_SERVO (9)	Global Write
MOVE_SERVO (2)	Global Write
MOVE_SERVO (8)	Global Write
_global_init (57)	Global Write
Qx_Verde%QX6.5	
SALIDAS (57)	Global Write
SALIDAS (60)	Global Write
_global_init (11)	Global Write
Ready_F_TRIG	
MAIN (66)	Global Write
_global_init (38)	Global Read
QxS_Numerical_Data%QB2	
MAIN (119)	Global Write
MAIN (120)	Global Write
MAIN (140)	Global Write
MAIN (154)	Global Write
_global_init (56)	Global Write
Qx_Rojo%QX6.3	
SALIDAS (36)	Global Write

SALIDAS (39)	Global Write
SALIDAS (42)	Global Write
_global_init (9)	Global Write
Qx_Rojo%QX6.3	
SALIDAS (36)	Global Write
SALIDAS (39)	Global Write
SALIDAS (42)	Global Write
_global_init (9)	Global Write
Qx_Ready%QX6.0	
SALIDAS (15)	Global Write
SALIDAS (17)	Global Write
_global_init (6)	Global Write
Qx_Ready%QX6.0	
SALIDAS (15)	Global Write
SALIDAS (17)	Global Write
_global_init (6)	Global Write
SR_ReseteaMovimiento	
MAIN (144)	Local Read
MAIN (151)	Local Read
MAIN (172)	Local Read
MAIN (197)	Local Read
MAIN (242)	Local Read
MAIN (269)	Local Read
SR_RESETEAMOVIMIENTO (1)	Local Write
QxS_Start_Flag%QB0	
MAIN (141)	Global Write
MAIN (158)	Global Write
MOVE_SERVO (6)	Global Write
MOVE_SERVO (5)	Global Write
_global_init (55)	Global Write
QxS_Speed%QB12	
MAIN (132)	Global Write
_global_init (59)	Global Write
Rearme_R_TRIG	
MAIN (68)	Global Write
MAIN (224)	Global Read
_global_init (40)	Global Read
QxS_Pushing_Speed%QB8	
MAIN (131)	Global Write
_global_init (58)	Global Write
QxS_Numerical_Data%QB2	
MAIN (119)	Global Write
MAIN (120)	Global Write
MAIN (140)	Global Write
MAIN (154)	Global Write
_global_init (56)	Global Write
Qx_Verde%QX6.5	
SALIDAS (57)	Global Write
SALIDAS (60)	Global Write
_global_init (11)	Global Write
Qx_Pass%QX6.2	
SALIDAS (29)	Global Write
SALIDAS (31)	Global Write
_global_init (8)	Global Write
Qx_Fail%QX6.1	
SALIDAS (22)	Global Write
SALIDAS (24)	Global Write
_global_init (7)	Global Write
Qx_Fail%QX6.1	
SALIDAS (22)	Global Write
SALIDAS (24)	Global Write
_global_init (7)	Global Write
Ix_SetaEmer%IX8.3	
MAIN (56)	Global Read
_global_init (4)	Global Write
SR_BorraEstados	
MAIN (117)	Local Read
MAIN (164)	Local Read

MAIN (170)	Local Read
MAIN (209)	Local Read
SR_BORRAESTADOS (1)	Local Write
Qx_Pass%QX6.2	
SALIDAS (29)	Global Write
SALIDAS (31)	Global Write
_global_init (8)	Global Write
Qx_Ambar%QX6.4	
SALIDAS (48)	Global Write
SALIDAS (51)	Global Write
_global_init (10)	Global Write
Pasta_F_TRIG	
MAIN (70)	Global Write
_global_init (42)	Global Read
Mx_VelocidadEmpuje	
MAIN (131)	Global Read
MAIN (132)	Global Read
_global_init (36)	Global Write
Mx_Oscilador_1Hz	
MAIN (81)	Global Write
MAIN (83)	Global Write
SALIDAS (36)	Global Read
_global_init (49)	Global Write
Mx_Recambiar	
MAIN (54)	Global Write
MAIN (67)	Global Read
_global_init (14)	Global Write
Mx_Move_3	
MAIN (149)	Global Read
MAIN (195)	Global Read
MAIN (240)	Global Read
MAIN (267)	Global Read
MOVE_SERVO (1)	Global Write
SR_RESETEAMOVIMIENTO (4)	Global Write
_global_init (27)	Global Write
Mx_Rearme	
MAIN (55)	Global Write
MAIN (68)	Global Read
_global_init (15)	Global Write
Mx_Move_1	
MOVE_SERVO (1)	Global Write
MOVE_SERVO (11)	Global Write
SR_RESETEAMOVIMIENTO (2)	Global Write
_global_init (25)	Global Write
Mx_Inicio	
MAIN (43)	Global Read
MAIN (49)	Global Write
_global_init (35)	Global Write
Mx_FlagEmergencia	
MAIN (56)	Global Write
MAIN (69)	Global Read
MAIN (163)	Global Read
_global_init (34)	Global Write
Mx_F1_Automatico	
MAIN (88)	Global Read
MAIN (118)	Global Write
MAIN (128)	Global Read
MAIN (137)	Global Read
MAIN (139)	Global Write
MAIN (149)	Global Read
MAIN (155)	Global Write
MOVE_SERVO (5)	Global Read
MOVE_SERVO (4)	Global Read
SALIDAS (14)	Global Read
SALIDAS (28)	Global Read
SALIDAS (56)	Global Read
SR_BORRAESTADOS (4)	Global Write
_global_init (17)	Global Write

Mx_D2_TratamientoErrores	
MAIN (156)	Global Write
MAIN (183)	Global Read
MAIN (191)	Global Read
MAIN (195)	Global Read
MAIN (201)	Global Write
MAIN (210)	Global Write
MAIN (215)	Global Read
MAIN (216)	Global Write
SALIDAS (21)	Global Read
SALIDAS (35)	Global Read
SALIDAS (47)	Global Read
SR_BORRAESTADOS (6)	Global Write
_global_init (19)	Global Write
Qx_Ambar%QX6.4	
SALIDAS (48)	Global Write
SALIDAS (51)	Global Write
_global_init (10)	Global Write
Mx_Move_2	
MAIN (137)	Global Read
MOVE_SERVO (1)	Global Write
MOVE_SERVO (10)	Global Write
SR_RESETEAMOVIMIENTO (3)	Global Write
_global_init (26)	Global Write
Mx_D2_Error_2	
MAIN (211)	Global Write
MAIN (215)	Global Read
MAIN (217)	Global Write
SALIDAS (35)	Global Read
_global_init (29)	Global Write
Mx_Move_0	
MAIN (125)	Global Write
MAIN (128)	Global Read
MAIN (173)	Global Write
MAIN (188)	Global Write
MAIN (191)	Global Read
MAIN (198)	Global Write
MAIN (235)	Global Read
MAIN (243)	Global Write
MAIN (257)	Global Write
MAIN (260)	Global Read
MAIN (268)	Global Write
MOVE_SERVO (1)	Global Write
SR_RESETEAMOVIMIENTO (1)	Global Write
_global_init (24)	Global Write
Mx_D2_Error_1	
MAIN (157)	Global Write
MAIN (183)	Global Read
MAIN (191)	Global Read
MAIN (195)	Global Read
MAIN (203)	Global Write
SALIDAS (47)	Global Read
_global_init (28)	Global Write
Mx_A6_PuestaEnCI	
MAIN (48)	Global Write
MAIN (225)	Global Write
MAIN (231)	Global Read
MAIN (235)	Global Read
MAIN (240)	Global Read
MAIN (246)	Global Write
MOVE_SERVO (2)	Global Read
MOVE_SERVO (1)	Global Read
SR_BORRAESTADOS (8)	Global Write
TRANSICIONES (12)	Global Read
_global_init (21)	Global Write
Mx_D1_Emergencia	
MAIN (165)	Global Write
MAIN (169)	Global Read

SALIDAS (21)	Global Read
SALIDAS (38)	Global Read
SR_BORRAESTADOS (5)	Global Write
_global_init (18)	Global Write
Mx_A5_PreparacionPuestaCI	
MAIN (171)	Global Write
MAIN (202)	Global Write
MAIN (219)	Global Write
MAIN (224)	Global Read
MAIN (226)	Global Write
SR_BORRAESTADOS (7)	Global Write
_global_init (20)	Global Write
Mt_Temp	
_global_init (60)	Global Write
Mi_EstadoSistema	
SALIDAS (37)	Global Write
SALIDAS (40)	Global Write
SALIDAS (49)	Global Write
SALIDAS (58)	Global Write
_global_init (61)	Global Write
Mx_A4_ParadaObtenida	
MAIN (72)	Global Read
MAIN (116)	Global Read
MAIN (271)	Global Write
MAIN (275)	Global Read
SALIDAS (14)	Global Read
SALIDAS (56)	Global Read
SR_BORRAESTADOS (10)	Global Write
_global_init (23)	Global Write
IxS_Info%IB4	
MAIN (72)	Global Read
MAIN (88)	Global Read
MAIN (208)	Global Read
MAIN (215)	Global Read
MAIN (252)	Global Read
SALIDAS (28)	Global Read
TRANSICIONES (3)	Global Read
TRANSICIONES (7)	Global Read
TRANSICIONES (13)	Global Read
TRANSICIONES (15)	Global Read
_global_init (53)	Global Write
IxS_Info%IB4	
MAIN (72)	Global Read
MAIN (88)	Global Read
MAIN (208)	Global Read
MAIN (215)	Global Read
MAIN (252)	Global Read
SALIDAS (28)	Global Read
TRANSICIONES (3)	Global Read
TRANSICIONES (7)	Global Read
TRANSICIONES (13)	Global Read
TRANSICIONES (15)	Global Read
_global_init (53)	Global Write
Mx_A3_ParadaPedida	
MAIN (142)	Global Write
MAIN (252)	Global Read
MAIN (260)	Global Read
MAIN (267)	Global Read
MAIN (270)	Global Write
SALIDAS (14)	Global Read
SALIDAS (56)	Global Read
SR_BORRAESTADOS (9)	Global Write
_global_init (22)	Global Write
IxS_CurrentSpeed%IB16	
_global_init (52)	Global Write
Mx_A1_ParadaEnCI	
MAIN (72)	Global Read
MAIN (107)	Global Read

MAIN (116)	Global Read
MAIN (247)	Global Write
SALIDAS (14)	Global Read
SALIDAS (56)	Global Read
SR_BORRAESTADOS (3)	Global Write
_global_init (16)	Global Write
Mi_NivelBote	
MAIN (138)	Global Write
_global_init (62)	Global Write
lxS_CurrentSpeed%IB16	
_global_init (52)	Global Write
lxS_CurrentForce%IB6	
MAIN (74)	Global Read
_global_init (54)	Global Write
lxS_CurrentForce%IB6	
MAIN (74)	Global Read
_global_init (54)	Global Write
lxS_Current_Position%IB0	
MAIN (138)	Global Read
_global_init (51)	Global Write
lxS_Current_Position%IB0	
MAIN (138)	Global Read
_global_init (51)	Global Write
lxD_Recambiar	
MAIN (54)	Global Read
MAIN (55)	Global Read
MAIN (76)	Global Read
MAIN (97)	Global Write
_global_init (12)	Global Write
lx_SetaEmer%IX8.3	
MAIN (56)	Global Read
_global_init (4)	Global Write
lxD_Rearme	
MAIN (77)	Global Read
MAIN (101)	Global Write
_global_init (13)	Global Write
lx_Recambiar%IX8.1	
MAIN (54)	Global Read
_global_init (2)	Global Write
lx_Recambiar%IX8.1	
MAIN (54)	Global Read
_global_init (2)	Global Write
lx_Rearme%IX8.2	
MAIN (55)	Global Read
_global_init (3)	Global Write
lx_Rearme%IX8.2	
MAIN (55)	Global Read
_global_init (3)	Global Write
lx_Ready%IX8.0	
MAIN (65)	Global Read
MAIN (66)	Global Read
MAIN (88)	Global Read
MAIN (116)	Global Read
MAIN (137)	Global Read
SALIDAS (28)	Global Read
_global_init (1)	Global Write
lx_Ready%IX8.0	
MAIN (65)	Global Read
MAIN (66)	Global Read
MAIN (88)	Global Read
MAIN (116)	Global Read
MAIN (137)	Global Read
SALIDAS (28)	Global Read
_global_init (1)	Global Write
lx_LocDis%IX14.0	
MAIN (54)	Global Read
MAIN (55)	Global Read
_global_init (5)	Global Write

Ix_LocDis%IX14.0

MAIN (54)	Global Read
MAIN (55)	Global Read
_global_init (5)	Global Write

Emergencia_F_TRIG

MAIN (69)	Global Write
MAIN (169)	Global Read
_global_init (41)	Global Read

Call Tree of MAIN (PRG-ST)