Escuela Universitaria Ingeniería Técnica Industrial ZARAGOZA

PROYECTO FINAL DE CARRERA

MEMORIA - ANEXO A

DOCUMENTACION SOFTWARE

Automatización de la cúpula de un telescopio de un observatorio astrofísico

AUTOR

LÁZARO GUERRERO, JAVIER

DIRECTOR

ROMEO TELLO, ANTONIO

ESPECIALIDAD

ELECTRÓNICA INDUSTRIAL

CONVOCATORIA

Diciembre 2013

INDICE

8 ANEXO A – Documentación software

8.1 PLC1: código de programación

Estructura de programa

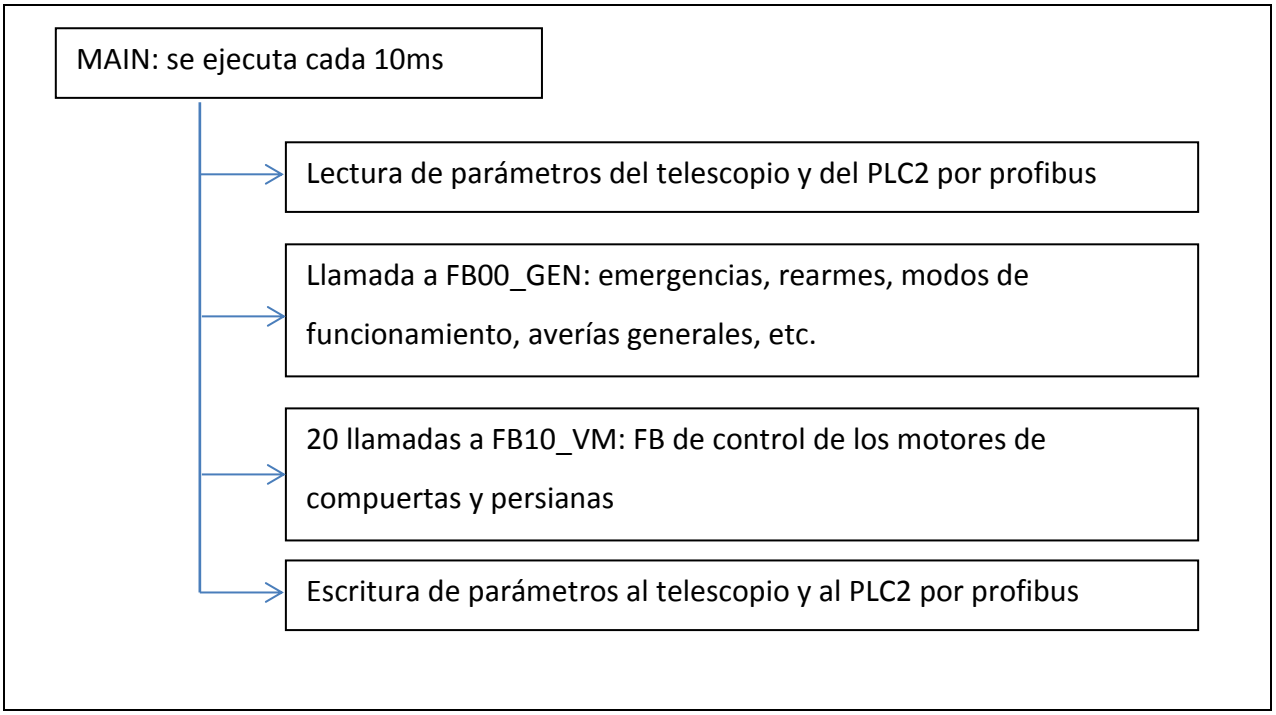

Fig.8.1.1 – PLC1 Estructura de programa

8.1.1 FB00_GEN

```
FUNCTION_BLOCK FB00_GEN
VAR
        Man: BOOL; (* Modo Manual *)
        Aut: BOOL; (* Modo Automático *)
        Def: BOOL; (* Reunión Defectos *)
        D_: Gen_D; (* Defectos Generales *)
        S_: Gen_S; (* Pulsadores Generales *)
        cont_1s: INT; (* General - Contador auxiliar 1s *)
        int_500ms: BOOL; (* General - Intermitencia 500ms *)
END_VAR
VAR_INPUT
        EMG: BOOL; (* Seta de emergencia *)
        SRea1: BOOL; (* Pulsador de rearme 1 *)
        SRea2: BOOL; (* Pulsador de rearme 2 *)
        SRea3: BOOL; (* Pulsador de rearme 3 *)
        SRea4: BOOL; (* Pulsador de rearme 4 *)
        SRea5: BOOL; (* Pulsador de rearme 5 *)
        Q_Cupula: BOOL; (* Térmico cuadro remoto: Cúpula *)
        F220: BOOL; (* Térmico alimentación 220VAC *)
        TM01: BOOL; (* Termostato armario general *)
END_VAR
VAR_OUTPUT
        KV: BOOL; (* Marcha contactor alimentación ventilador armario *)
        KRea: BOOL; (* Rearme general instalación - programa de seguridad *)
        KRea_Remoto: BOOL; (* Rearme general instalación - programa de seguridad remoto *)
        HRea1: BOOL; (* Piloto emergencia general 1 *)
        HRea2: BOOL; (* Piloto emergencia general 2 *)
        HRea3: BOOL; (* Piloto emergencia general 3 *)
        HRea4: BOOL; (* Piloto emergencia general 4 *)
        HRea5: BOOL; (* Piloto emergencia general 5 *)
END_VAR
```

```
(* Gestión de Generales de la instalación *)

(****************************************** Previo ***************************************************)
(* Bits Generales *)
CERO := 0;
UNO := 1;
(* Generación de ciclos de tiempo *)
cont_1s:=cont_1s+1;
IF cont_1s > 100 THEN cont_1s:=0; END_IF;

(* Intermitencia 500ms *)
IF cont_1s <= 50 THEN int_500ms:=1; ELSE int_500ms:=0; END_IF;
```

```
(****************************************** Modos ******************************************************)
(* Modos Manual - Automático *)
Man:=NOT Aut;
Aut:=MAIN.VentanaCompuerta01.Aut AND MAIN.VentanaCompuerta02.Aut AND MAIN.VentanaCompuerta03.Aut AND
MAIN.VentanaCompuerta04.Aut AND
                MAIN.VentanaCompuerta05.Aut AND MAIN.VentanaCompuerta06.Aut AND MAIN.VentanaCompuerta07.Aut AND
MAIN.VentanaCompuerta08.Aut AND
                MAIN.VentanaCompuerta09.Aut AND MAIN.VentanaCompuerta10.Aut AND MAIN.VentanaPersiana01.Aut AND
MAIN.VentanaPersiana02.Aut AND
                MAIN.VentanaPersiana03.Aut AND MAIN.VentanaPersiana04.Aut AND MAIN.VentanaPersiana05.Aut AND
MAIN.VentanaPersiana06.Aut AND
                MAIN.VentanaPersiana07.Aut AND MAIN.VentanaPersiana08.Aut AND MAIN.VentanaPersiana09.Aut AND
MAIN.VentanaPersiana10.Aut;

(************************************* Rearme general ************************************************)
(* Rearme general *)
KRea:=S_.ReaWeb OR SRea1 OR SRea2 OR SRea3 OR SRea4 OR SRea5 OR MAIN.GEN_PLC2_PLC1.ReaPLC2;
KRea_Remoto:=S_.ReaWeb OR SRea1 OR SRea2 OR SRea3 OR SRea4 OR SRea5;
(* Rearme Defectos *)
IF KRea THEN
        D_.Q_Cupula:=0;
        D_.F220:=0;
        D_.TM01:=0;
END_IF;
(****************************************** Defectos ******************************************************)
(* Defectos HW *)
IF NOT Q_Cupula THEN D_.Q_Cupula:=1; END_IF;      (* Defecto térmico alimentación cuadro remoto: Cúpula *)
IF NOT F220 THEN D_.F220:=1; END_IF;                    (* Defecto térmico alimentación 220VAC *)
IF NOT TM01 THEN D_.TM01:=1; END_IF;                    (* Defecto Termostato ventilador armario *)

(* Reunión Defectos *)
Def:=D_.Q_Cupula OR D_.F220 OR D_.TM01;
(********************************** Flags/Habilitaciones ***************************************)
(* Habilitar contactor alimentación 220VAC relé alimentación ventilador armario  *)
KV:=S_.HabilitarKV ;
(* Pilotos Emergencia General *)
HRea1:=EMG AND int_500ms;
HRea2:=EMG AND int_500ms;
HRea3:=EMG AND int_500ms;
HRea4:=EMG AND int_500ms;
HRea5:=EMG AND int_500ms;
```

8.1.2 FB10_VM: Bloque de función para la gestión de motores Compuerta y Persiana

```
FUNCTION_BLOCK FB10_VM
VAR
	Man: BOOL; (* Modo Manual *)
	Aut: BOOL; (* Modo Automático *)
	Def: BOOL; (* Reunión Defectos *)
	AKH: BOOL; (* Habilitar marcha en automático *)
	MKH: BOOL; (* Habilitar marcha en manual *)
	AKM: BOOL; (* Mover en automático: '0' Quitar - '1' Poner *)
	MKM: BOOL; (* Mover en manual: '0' Quitar - '1' Poner *)
	Activo: BOOL; (* Movimiento motor activo *)
	Poniendo: BOOL; (* Poniendo: cerrando compuerta ó bajando persiana *)
	Quitando: BOOL; (* Quitando: abriendo compuerta ó subiendo persiana *)
	AuxKH: BOOL; (* Auxiliar Relé habilitación marcha motor *)
	FPKH: R_TRIG; (* Flanco positivo de habilitación marcha motor *)
	FNKH: F_TRIG; (* Flanco negativo de habilitación marcha motor *)
	TOFFMotor: INT; (* Tiempo a la desconexión del motor *)
	TActMotor: INT; (* Tiempo contaje de activación movimiento motor [cseg] *)
	TActKH: INT; (* Tiempo contaje para activar KH *)
	SPMan: REAL; (* SP Manual % *)
	SPAut_Ant: REAL; (* SP Automático ciclo anterior % *)
	S_: Motor_S; (* Pulsadores *)
	D_: Motor_D; (* Defectos *)
	K_: Motor_K; (* Constantes *)
	kk: BOOL;
END_VAR
VAR_INPUT
	Q: BOOL; (* Térmico alimentación motores *)
	BP: BOOL; (* Detector - Final de carrera puesto, persiana abajo, compuerta cerrada *)
	BQ: BOOL; (* Detector - Final de carrera quitado, persiana arriba, compuerta abierta *)
	SRea: BOOL; (* Pulsador rearme *)
	SMan_Aut: BOOL; (* Pulsador modo Manual/Automático *)
	WDef: BOOL; (* Defecto externo *)
	WAut: BOOL; (* Condiciones de marcha automático *)
	TRecorridox10ms: REAL; (* Tiempo equivalente al recorrido total de posición del motor [cseg] *)
	SPAut: REAL; (* SP Automático *)
END_VAR
VAR_OUTPUT
	KH: BOOL; (* Relé habilitación marcha motor *)
	KM: BOOL; (* Relé movimiento motor: '0' Quitar - '1' Poner *)
END_VAR
VAR PERSISTENT
	ActPos: REAL; (* Posición actual % *)
	ActPos0: REAL; (* Posición antes del inicio del movimiento % *)
END_VAR
```

```
(* Bloque de función para la gestión en Ventanas de motores Compuerta y Persiana *)
(****************************************** Previo ***************************************************)
(* Control límites consigna de automático *)
IF SPAut > 100 THEN SPAut:=100; END_IF;
IF SPAut < 0 THEN SPAut:=0; END_IF;

(* Cálculo de la posición actual en función del tiempo de movimiento *)
IF BP THEN                                          (* Posición de puesto (persiana abajo, compuerta cerrada) *)
        ActPos:=0;
ELSIF BQ THEN                                       (* Posición de quitado (persiana arriba, compuerta abierta) *)
        ActPos:=100;
ELSIF NOT BQ AND NOT BP THEN                        (* No está en fin de recorrido, *)
        IF KH AND KM THEN                           (* y está cerrando/bajando *)
                TActMotor:=TActMotor + 1;           (* Tiempo actual de movimiento motor *)
                ActPos:=ActPos0 - (100 * (TActMotor / TRecorridox10ms));
        ELSIF KH AND NOT KM THEN                    (* y está abriendo/subiendo *)
                TActMotor:=TActMotor + 1;           (* Tiempo actual de movimiento motor *)
                ActPos:=ActPos0 + (100 * (TActMotor / TRecorridox10ms));
        END_IF;
END_IF;

(* Gestión del tiempo a la desconexión del motor *)
(* Si Manual y no hay habilitación o Automático y está en movimiento y fin Tiempo desconexión *)
(*IF (Man AND NOT MKH) OR (Aut AND AKH AND (TActMotor >= TOFFMotor)) THEN *)(* Paro en automático por tiempo a la desconexión *)
(* Paro en automático por tiempo a la desconexión si SP <> 0% ó 100%.   Si SP es 0% ó 100% -> paro si tengo Finales de carrera  *)
IF (Man AND NOT MKH) OR
   (Aut AND AKH AND ((TActMotor >= TOFFMotor AND SPAut<>0 AND SPAut<>100) OR (SPAut=100 AND BQ) OR (SPAut=0 AND BP))) THEN
        TActMotor:=0;                                   (* Inicializar el tiempo actual de movimiento *)
        AKH:=0;                                         (* Quitar habilitación de automático *)
        AKM:=0;                                         (* Relé de movimiento a reposo: '0' Quitar en automático *)
        ActPos0:=ActPos;
        kk:=1;
END_IF;

(****************************************** Modos ***************************************************)

(* Modos Manual - Automático *)
Man:=NOT Aut;
Aut:=SMan_Aut AND NOT Def AND NOT WDef;

(* Gestión Modo Automático - Manual *)

(* Modo AUTOMÁTICO *)
IF Aut THEN
        MKH:=0;                                     (* Borrar habilitación marcha en manual *)
        MKM:=0;                                     (* Relé de movimiento a reposo: '0' Quitar en manual *)
```

```
		(* Cambiar posición si hay variación de Consigna y ha finalizado el movimiento y no tiene bit Arr. secuencial motores *)
		IF ((SPAut <> SPAut_Ant) AND (NOT AKH) AND (NOT G_ARM) AND (G_MotoresActivos <10))THEN
			SPAut_Ant:=SPAut;
								(* Actualizar SP para siguiente ciclo *)
			TOFFMotor:=REAL_TO_INT(((ABS(SPAut - ActPos))/100.0) * TRecorridox10ms);		(* Tiempo a la desconexión del motor *)
			AKH:=1;
						(* Habilitar marcha en automático *)
			IF SPAut < ActPos THEN		(* Poner (cerrar compuerta o bajar persiana ) *)
				AKM:=1;				(* Marcha Poner en automático *)
			ELSE
					(* Quitar (abrir compuerta o subir persiana ) *)
				AKM:=0;				(* Marcha Quitar en automático *)
			END_IF;
		END_IF;

(* Modo MANUAL *)
ELSE
		AKH:=0;							(* Borrar habilitación marcha en automático *)
		AKM:=0;							(* Relé de movimiento a reposo: '0' Quitar en
automático *)
		IF S_.Poner AND NOT S_.Quitar AND NOT BP THEN	(* Poner (cerrar compuerta o bajar persiana ) *)
			MKH:=1;						(* Habilitar marcha en manual *)
			MKM:=1;						(* Marcha Poner en manual *)
		ELSIF S_.Quitar AND NOT S_.Poner AND NOT BQ THEN(* Quitar (abrir compuerta o subir persiana ) *)
			MKH:=1;						(* Habilitar marcha en manual *)
			MKM:=0;						(* Marcha Quitar en manual *)
		ELSE
			MKH:=0;						(* Deshabilitar marcha en manual *)
			MKM:=0;						(* Relé de movimiento a reposo: '0' Quitar
en manual *)
		END_IF;
END_IF;

(************************************ Rearme defectos ***********************************************)
(* Rearme Defectos *)
IF SRea THEN
		D_.Q:=0;
		D_.Detector:=0;
END_IF;

(************************************* Defectos ************************************************)
(* Defectos HW *)
IF NOT Q THEN D_.Q:=1; END_IF;				(* Defecto Térmico *)

(* Defecto Detectores *)
IF BP AND BQ THEN D_.Detector:=1; END_IF;			(*Defecto Detectores de final de carrera activos a la vez *)
```

```
(* Reunión Defectos *)
Def:=D_.Q OR D_.Detector;

(************************************ Flags/Habilitaciones ****************************************)
(* Habilitar relés de marcha *)
(* Gestión Relé Abrir/Subir - Cerrar/Bajar *)
KM:=AKM OR MKM;
                                        (* Mover motor: '0' Quitar - '1' Poner *)
(* Gestión Relé Habilitación marcha motor, activación del relé temporizado para asegurar el movimiento *)
AuxKH:=((AKH AND ((AKM AND NOT BP) OR (NOT AKM AND NOT BQ))) OR          (* Habilitación marcha automático *)
                (MKH AND ((MKM AND NOT BP) OR (NOT MKM AND NOT BQ))));          (* Habilitación marcha manual *)
IF AuxKH AND NOT Def AND NOT KH THEN
                                        (* Auxiliar Relé para temporizar arranque *)
        TActKH:=TActKH + 1;
        IF TActKH >= 5 THEN
                                        (* Esperar 5 ciclos antes de activar relé *)
                TActKH:=0;
                KH:=1;
        END_IF;
END_IF;
IF NOT AuxKH THEN KH:=0; END_IF;

(* Gestión de arranque secuencial motores *)
FPKH(CLK:=AuxKH , Q=> );
                                        (* Flanco positivo Habilitación marcha motor *)
IF FPKH.Q THEN G_ARM:=1; G_MotoresActivos:=G_MotoresActivos+1; END_IF;          (* Poner a '1' bit general de arranque secuencial motores *)
FNKH(CLK:=AuxKH , Q=> );
                                        (* Flanco negativo Habilitación marcha motor *)
IF FNKH.Q THEN G_MotoresActivos:=G_MotoresActivos-1;END_IF;

(* Control límites posición actual *)
IF ActPos >= 100 THEN ActPos:=100; END_IF;
IF ActPos <=0 THEN ActPos:=0; END_IF;
(************************************ Visualizar en Web ****************************************)
Poniendo:=KH AND KM;                                    (* Bit que indica movimiento: cerrando compuerta ó bajando persiana *)
Quitando:=KH AND NOT KM;      (* Bit que indica movimiento: abriendo compuerta ó subiendo persiana *)
```

8.1.3 FB25_ProfibusServo_PLC1_PLC2

```
FUNCTION_BLOCK FB25_ProfibusServo_PLC1_PLC2
VAR_INPUT
        PLC1_Axis_Cmd: WORD; (* Axis bits commands *)
        PLC1_PosSP_Par: INT; (* Axis Position SetPoint (1/100 deg) *)
        PLC1_JogSpd_Par: INT; (* Speed sepoint for jog move (1/100 deg) *)
END_VAR
VAR_OUTPUT
        HabilitarEje: BOOL; (* '1' Habilitar sincronización del Eje con SetPoint / '0' Parar el movimiento *)
        HomeEje: BOOL; (* '1' Ejecutar secuencia de Home / '0' Parar el movimiento *)
        JogFEje: BOOL; (* '1' Mover Jog sentido positivo Eje / '0' Parar el movimiento *)
        JogBEje: BOOL; (* '1' Mover Jog sentido negativo Eje / '0' Parar el movimiento *)
        SP_PosTarget: LREAL; (* Setpoint de posición absoluta objetivo en % *)
        SP_Vel: LREAL; (* Setpoint de velocidad *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC1 -> PLC2 *)

(* Comandos binarios *)

(* Valores de Consigna *)
PLC1_Axis_Cmd:=0;
```

8.1.4 FB26_ProfibusServo_PLC2_PLC1

```
FUNCTION_BLOCK FB26_ProfibusServo_PLC2_PLC1
VAR
        EjePreparado: BOOL; (* '1' Eje habilitado para movimiento y sin fallo / '0' Arranque de eje no finalizado *)
        EjePosHome: BOOL; (* '1' Eje en posición de Home / '0' Eje no está en Home *)
        EjeActivo: BOOL; (* '1' Eje en movimiento / '0' Eje parado *)
        EjeTarget: BOOL; (* '1' Eje alcanza posición de destino / '0' Eje no ha alcanzado la posición de destino *)
        EjeOk: BOOL; (* '1' Eje estado Ok / '0' Eje en error *)
        EjePosAbierto: BOOL; (* '1' Eje en posición de abierto / '0' Eje no está en posición de abierto *)
        EjePosCerrado: BOOL; (* '1' Eje en posición de cerrado / '0' Eje no está en posición de cerrado *)
        EjeFallo1: BOOL; (* '1' Eje en fallo 1 / '0' Eje sin fallo 1 *)
        EjeFallo2: BOOL; (* '1' Eje en fallo 2 / '0' Eje sin fallo 2 *)
        EjeFallo3: BOOL; (* '1' Eje en fallo 3 / '0' Eje sin fallo 3 *)
        EjeFallo4: BOOL; (* '1' Eje en fallo 4 / '0' Eje sin fallo 4 *)
        ActPos: LREAL; (* Eje posición actual [cg] *)
        ActConsumo: REAL; (* Eje consumo actual [A] *)
END_VAR
VAR_INPUT
        PLC1_Axis_Sta: WORD; (* Axis bits status *)
        PLC1_Pos_Mes: INT; (* Axis actual position (1/100 deg) *)
        PLC1_Cur_Mes: INT; (* Actual current (1/10 A) *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC2 -> PLC1 *)

(* Bits de Estados *)
EjePreparado:=PLC1_Axis_Sta.0;
EjePosHome:=PLC1_Axis_Sta.1;
EjeActivo:=PLC1_Axis_Sta.2;
EjeTarget:=PLC1_Axis_Sta.3;
EjeOk:=PLC1_Axis_Sta.4;
EjePosAbierto:=PLC1_Axis_Sta.5;
EjePosCerrado:=PLC1_Axis_Sta.6;
EjeFallo1:=PLC1_Axis_Sta.7;
EjeFallo2:=PLC1_Axis_Sta.8;
EjeFallo3:=PLC1_Axis_Sta.9;
EjeFallo4:=PLC1_Axis_Sta.10;


(* Valores Medidos *)
```

8.1.5 FB27_ProfibusGen_PLC2_PLC1

```
FUNCTION_BLOCK FB27_ProfibusGen_PLC2_PLC1
VAR_INPUT
        PLC2_Gen_Sta1: WORD; (* Generales Bits de estado *)
        PLC2_Gen_Sta2: WORD; (* Generales Bits de estado *)
END_VAR
VAR_OUTPUT
        ReaPLC2: BOOL; (* Rearme desde PLC2 *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC1 -> PLC2 *)

(* Comandos binarios *)
ReaPLC2:=PLC2_Gen_Sta1.0;

(* Valores de Consigna *)
```

8.1.6 FB28_ProfibusGen_PLC1_PLC2

```
FUNCTION_BLOCK FB28_ProfibusGen_PLC1_PLC2
VAR_INPUT
        ReaPLC1: BOOL; (* Rearme desde PLC1 *)
        VentArmarioPLC2: BOOL; (* Marcha ventilador armario PLC2 *)
        AutRemoto: BOOL := 0; (* AutRemoto = 0 ->Modo Local (PLC1) / AutRemoto = 1 -> Modo remoto (PLC2) *)
END_VAR
VAR_OUTPUT
        PLC2_Gen_Cmd1: WORD; (* Generales Bits de Estado *)
        PLC2_Gen_Cmd2: WORD; (* Generales Bits de Estado *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC2 -> PLC1 *)

(* Bits de Estados *)
PLC2_Gen_Cmd1.0:=ReaPLC1;
PLC2_Gen_Cmd1.1:=AutRemoto;


(* Valores Medidos *)
```

8.1.7 MAIN

```
PROGRAM MAIN
VAR
	CTARM: INT; (* Constante de arranque secuencial motores *)
	CARM: INT; (* Contador tiempo arranque secuencial motores *)
	Generales: FB00_GEN; (* FB de gestión de Generales *)
	VentanaCompuerta01: FB10_VM; (* FB de gestión Ventana Motor - VCM01 *)
	VentanaCompuerta02: FB10_VM; (* FB de gestión Ventana Motor - VCM02 *)
	VentanaCompuerta03: FB10_VM; (* FB de gestión Ventana Motor - VCM03 *)
	VentanaCompuerta04: FB10_VM; (* FB de gestión Ventana Motor - VCM04 *)
	VentanaCompuerta05: FB10_VM; (* FB de gestión Ventana Motor - VCM05 *)
	VentanaCompuerta06: FB10_VM; (* FB de gestión Ventana Motor - VCM06 *)
	VentanaCompuerta07: FB10_VM; (* FB de gestión Ventana Motor - VCM07 *)
	VentanaCompuerta08: FB10_VM; (* FB de gestión Ventana Motor - VCM08 *)
	VentanaCompuerta09: FB10_VM; (* FB de gestión Ventana Motor - VCM09 *)
	VentanaCompuerta10: FB10_VM; (* FB de gestión Ventana Motor - VCM10 *)
	VentanaPersiana01: FB10_VM; (* FB de gestión Ventana Motor - VPM01 *)
	VentanaPersiana02: FB10_VM; (* FB de gestión Ventana Motor - VPM02 *)
	VentanaPersiana03: FB10_VM; (* FB de gestión Ventana Motor - VPM03 *)
	VentanaPersiana04: FB10_VM; (* FB de gestión Ventana Motor - VPM04 *)
	VentanaPersiana05: FB10_VM; (* FB de gestión Ventana Motor - VPM05 *)
	VentanaPersiana06: FB10_VM; (* FB de gestión Ventana Motor - VPM06 *)
	VentanaPersiana07: FB10_VM; (* FB de gestión Ventana Motor - VPM07 *)
	VentanaPersiana08: FB10_VM; (* FB de gestión Ventana Motor - VPM08 *)
	VentanaPersiana09: FB10_VM; (* FB de gestión Ventana Motor - VPM09 *)
	VentanaPersiana10: FB10_VM; (* FB de gestión Ventana Motor - VPM10 *)
	AZ_PLC1_PLC2: FB25_ProfibusServo_PLC1_PLC2; (* FB de interconexión Profibus PLC1 - PLC2 variables Azimut *)
	AZ_PLC2_PLC1: FB26_ProfibusServo_PLC2_PLC1; (* FB de interconexión Profibus PLC2 - PLC1 variables Azimut *)
	GEN_PLC1_PLC2: FB28_ProfibusGen_PLC1_PLC2; (* FB de interconexión Profibus PLC1 - PLC2 variables Generales *)
	GEN_PLC2_PLC1: FB27_ProfibusGen_PLC2_PLC1; (* FB de interconexión Profibus PLC1 - PLC2 variables Generales *)
	VP_B11_TON: TON;
	VP_B12_TON: TON;
	VP_B21_TON: TON;
	VP_B22_TON: TON;
	VP_B31_TON: TON;
	VP_B32_TON: TON;
	VP_B41_TON: TON;
	VP_B42_TON: TON;
	VP_B51_TON: TON;
	VP_B52_TON: TON;
	VP_B61_TON: TON;
	VP_B62_TON: TON;
	VP_B71_TON: TON;
	VP_B72_TON: TON;
	VP_B81_TON: TON;
```

```
        VP_B82_TON: TON;
        VP_B91_TON: TON;
        VP_B92_TON: TON;
        VP_B101_TON: TON;
        VP_B102_TON: TON;
        UPS1: FB_S_UPS; (* Guarda variables persistentes en Compact Flash *)
        kk1: BOOL;
        AZ_VCCorrecciones: INT; (* AZIMUTAL-Contador de correciones en destino *)
        AZ_MF1: BOOL; (* AZIMUTAL-Memoria flanco 1 *)
END_VAR
```

```
(* Main - Llamada a todos los elementos de la Instalación *)
(* Ejecución cíclica del programa cada 10ms *)

(* Gestión tiempo de arranque secuencial motores *)
(* Vigilancia límites *)
IF CTARM < 100 THEN CTARM:=100; END_IF;
IF CTARM > 1000 THEN CTARM:=1000; END_IF;
(* Contador tiempo actual arranque motor *)
IF G_ARM THEN
        CARM:=CARM + 1;
        IF CARM >= CTARM THEN                (* Fin tiempo arranque por motor *)
                CARM:=0;                                (* Inicializar contador *)
                G_ARM:=0;                                        (* Borrar bit de Arr. secuencial motores *)
        END_IF;
END_IF;
(*Número de motores en marcha*)

IF (G_MotoresActivos < 0) THEN G_MotoresActivos:=0; END_IF;
IF (
NOT VentanaCompuerta01.AuxKH AND
NOT VentanaCompuerta02.AuxKH AND
NOT VentanaCompuerta03.AuxKH AND
NOT VentanaCompuerta04.AuxKH AND
NOT VentanaCompuerta05.AuxKH AND
NOT VentanaCompuerta06.AuxKH AND
NOT VentanaCompuerta07.AuxKH AND
NOT VentanaCompuerta08.AuxKH AND
NOT VentanaCompuerta09.AuxKH AND
NOT VentanaCompuerta10.AuxKH AND
NOT VentanaPersiana01.AuxKH AND
NOT VentanaPersiana02.AuxKH AND
NOT VentanaPersiana03.AuxKH AND
NOT VentanaPersiana04.AuxKH AND
NOT VentanaPersiana05.AuxKH AND
NOT VentanaPersiana06.AuxKH AND
```

```
NOT VentanaPersiana07.AuxKH AND
NOT VentanaPersiana08.AuxKH AND
NOT VentanaPersiana09.AuxKH AND
NOT VentanaPersiana10.AuxKH
)THEN
G_MotoresActivos:=0;
END_IF;
(* Llamada a FB de interconexión Profibus de variables "Generales" desde PLC2 a PLC1  *)
GEN_PLC2_PLC1(          PLC2_Gen_Sta1:=PLC2_W.GEN.Gen_Sta1 ,
                        PLC2_Gen_Sta2:=PLC2_W.GEN.Gen_Sta2 ,
        ReaPLC2=> );
(* Llamada a FB de interconexión Profibus de variables "Azimut" desde PLC2 a PLC1  *)
AZ_PLC2_PLC1(           PLC1_Axis_Sta:= ,
                        PLC1_Pos_Mes:= ,
                        PLC1_Cur_Mes:= );
(* Llamada a FB de gestión de Generales *)
Generales(      EMG:=NOT G_EMG ,            (* Programa seguridad HW - Señal Emergencia instalación *)
                SRea1:=G_SH01 ,                     (* Entrada digital - Pulsador rearme instalación 1 *)
                SRea2:=G_SH02 ,                     (* Entrada digital - Pulsador rearme instalación 2 *)
                SRea3:=G_SH03 ,                     (* Entrada digital - Pulsador rearme instalación 3 *)
                SRea4:=G_SH04 ,                     (* Entrada digital - Pulsador rearme instalación 4 *)
                SRea5:=G_SH05 ,                     (* Entrada digital - Pulsador rearme instalación 5 *)
                Q_Cupula:=CU_Q ,                    (* Entrada digital - Térmico alimentación cuadro remoto: Cúpula *)
                F220:=G_F220 ,                      (* Entrada digital - Térmico alimentación 220VAC *)
                TM01:=G_TM01 ,                      (* Entrada digital - Termostato armario *)
        KV=>G_KV ,                                  (* Salida digital - Ventilador armario *)
        KRea=>G_Rea ,                               (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
        KRea_Remoto=> ,                     (* Programa seguridad HW - Señal Rearme Emergencia instalación remoto *)
        HRea1=>G_H01 ,                              (* Salida digital - Piloto Emergencia instalación 1 *)
        HRea2=>G_H02 ,                              (* Salida digital - Piloto Emergencia instalación 2 *)
        HRea3=>G_H03 ,                              (* Salida digital - Piloto Emergencia instalación 3 *)
        HRea4=>G_H04 ,                              (* Salida digital - Piloto Emergencia instalación 4 *)
        HRea5=>G_H05 );                             (* Salida digital - Piloto Emergencia instalación 5 *)
(* Llamada a FB de gestión Motor Ventana Compuerta 01 - VCM01 *)
VentanaCompuerta01(     Q:=VC_Q01 ,                 (* Entrada digital - Térmico alimentación motor *)
                        BP:=VC_FC12 ,               (* Entrada digital - Fin de recorrido compuerta cerrada *)
                        BQ:=VC_FC11 ,               (* Entrada digital - Fin de recorrido compuerta abierta *)
                        SRea:=G_Rea,                (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                        SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                        WDef:=MAIN.Generales.EMG,   (* Programa seguridad HW - Señal Emergencia instalación *)
                        WAut:= ,
                        TRecorridox10ms:=900 ,      (* Tiempo total recorrido motor en cseg *)
                        SPAut:=VC_SPAut ,           (* Consigna de posición en automático *)
        KH=>VC_K11 ,                                (* Salida digital - Habilitar marcha compuerta *)
        KM=>VC_K12 );                               (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
```

```
(* Llamada a FB de gestión Motor Ventana Compuerta 02 - VCM02 *)
VentanaCompuerta02(         Q:=VC_Q02 ,                      (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC22 ,                    (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC21 ,                    (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                     (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,        (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,           (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,                (* Consigna de posición en automático *)
          KH=>VC_K21 ,                                       (* Salida digital - Habilitar marcha compuerta *)
          KM=>VC_K22 );                                      (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
(* Llamada a FB de gestión Motor Ventana Compuerta 03 - VCM03 *)
VentanaCompuerta03(         Q:=VC_Q03 ,                      (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC32 ,                    (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC31 ,                    (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                     (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,        (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,           (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,                (* Consigna de posición en automático *)
          KH=>VC_K31 ,                                       (* Salida digital - Habilitar marcha compuerta *)
          KM=>VC_K32 );                                      (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
(* Llamada a FB de gestión Motor Ventana Compuerta 04 - VCM04 *)
VentanaCompuerta04(         Q:=VC_Q04 ,                      (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC42 ,                    (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC41 ,                    (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                     (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,        (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,           (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,                (* Consigna de posición en automático *)
          KH=>VC_K41 ,                                       (* Salida digital - Habilitar marcha compuerta *)
          KM=>VC_K42 );                                      (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
```

```
(* Llamada a FB de gestión Motor Ventana Compuerta 05 - VCM05 *)
VentanaCompuerta05(         Q:=VC_Q05 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC52 ,                        (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC51 ,                        (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,            (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,               (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,                    (* Consigna de posición en automático *)
          KH=>VC_K51 ,                                           (* Salida digital - Habilitar marcha compuerta *)
          KM=>VC_K52 );                                          (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
(* Llamada a FB de gestión Motor Ventana Compuerta 06 - VCM06 *)
VentanaCompuerta06(         Q:=VC_Q06 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC62 ,                        (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC61 ,                        (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,            (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,               (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,                    (* Consigna de posición en automático *)
          KH=>VC_K61 ,                                           (* Salida digital - Habilitar marcha compuerta *)
          KM=>VC_K62 );                                          (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
(* Llamada a FB de gestión Motor Ventana Compuerta 07 - VCM07 *)
VentanaCompuerta07(         Q:=VC_Q07 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC72 ,                        (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC71 ,                        (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,            (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,               (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,                    (* Consigna de posición en automático *)
          KH=>VC_K71 ,                                           (* Salida digital - Habilitar marcha compuerta *)
          KM=>VC_K72 );                                          (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
```

```
(* Llamada a FB de gestión Motor Ventana Compuerta 08 - VCM08 *)
VentanaCompuerta08(         Q:=VC_Q08 ,                    (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC82 ,                  (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC81 ,                  (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                   (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,      (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,         (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,              (* Consigna de posición en automático *)
                KH=>VC_K81 ,                               (* Salida digital - Habilitar marcha compuerta *)
                KM=>VC_K82 );                              (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
(* Llamada a FB de gestión Motor Ventana Compuerta 09 - VCM09 *)
VentanaCompuerta09(         Q:=VC_Q09 ,                    (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC92 ,                  (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC91 ,                  (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                   (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,      (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,         (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,              (* Consigna de posición en automático *)
                KH=>VC_K91 ,                               (* Salida digital - Habilitar marcha compuerta *)
                KM=>VC_K92 );                              (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
(* Llamada a FB de gestión Motor Ventana Compuerta 10 – VCM10 *)
VentanaCompuerta10(         Q:=VC_Q10 ,                    (* Entrada digital - Térmico alimentación motor *)
                            BP:=VC_FC102 ,                 (* Entrada digital - Fin de recorrido compuerta cerrada *)
                            BQ:=VC_FC101 ,                 (* Entrada digital - Fin de recorrido compuerta abierta *)
                            SRea:=G_Rea,                   (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG,      (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=900 ,         (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VC_SPAut ,              (* Consigna de posición en automático *)
                KH=>VC_K101 ,                              (* Salida digital - Habilitar marcha compuerta *)
                KM=>VC_K102 );                             (* Salida digital - '0' Abrir / '1' Cerrar compuerta *)
```

```
(* Llamada a FB de gestión Motor Ventana Persiana 01 - VPM01 *)
VP_B11_TON(IN:=VP_B11 , PT:=T#600ms , Q=> , ET=> );
VP_B12_TON(IN:=VP_B12 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana01(          Q:=VP_Q01 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B12_TON.Q ,                   (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B11_TON.Q ,                   (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,           (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,              (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,                    (* Consigna de posición en automático *)
        KH=>VP_K11 ,                                             (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K12 );                                            (* Salida digital - '0' Subir / '1' Bajar persiana *)
(* Llamada a FB de gestión Motor Ventana Persiana 02 - VPM02 *)
VP_B21_TON(IN:=VP_B21 , PT:=T#600ms , Q=> , ET=> );
VP_B22_TON(IN:=VP_B22 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana02(          Q:=VP_Q02 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B22_TON.Q ,                   (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B21_TON.Q ,                   (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,           (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,              (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,                    (* Consigna de posición en automático *)
        KH=>VP_K21 ,                                             (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K22 );                                            (* Salida digital - '0' Subir / '1' Bajar persiana *)
(* Llamada a FB de gestión Motor Ventana Persiana 03 - VPM03 *)
VP_B31_TON(IN:=VP_B31 , PT:=T#600ms , Q=> , ET=> );
VP_B32_TON(IN:=VP_B32 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana03(          Q:=VP_Q03 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B32_TON.Q ,                   (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B31_TON.Q ,                   (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,           (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,              (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,                    (* Consigna de posición en automático *)
        KH=>VP_K31 ,                                             (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K32 );                                            (* Salida digital - '0' Subir / '1' Bajar persiana *)
```

```
(* Llamada a FB de gestión Motor Ventana Persiana 04 - VPM04 *)
VP_B41_TON(IN:=VP_B41 , PT:=T#600ms , Q=> , ET=> );
VP_B42_TON(IN:=VP_B42 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana04(          Q:=VP_Q04 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B42_TON.Q ,                   (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B41_TON.Q ,                   (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,           (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,              (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,                    (* Consigna de posición en automático *)
        KH=>VP_K41 ,                                             (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K42 );                                            (* Salida digital - '0' Subir / '1' Bajar persiana *)
(* Llamada a FB de gestión Motor Ventana Persiana 05 - VPM05 *)
VP_B51_TON(IN:=VP_B51 , PT:=T#600ms , Q=> , ET=> );
VP_B52_TON(IN:=VP_B52 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana05(          Q:=VP_Q05 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B52_TON.Q ,                   (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B51_TON.Q ,                   (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,           (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,              (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,                    (* Consigna de posición en automático *)
        KH=>VP_K51 ,                                             (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K52 );                                            (* Salida digital - '0' Subir / '1' Bajar persiana *)
(* Llamada a FB de gestión Motor Ventana Persiana 06 - VPM06 *)
VP_B61_TON(IN:=VP_B61 , PT:=T#600ms , Q=> , ET=> );
VP_B62_TON(IN:=VP_B62 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana06(          Q:=VP_Q06 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B62_TON.Q ,                   (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B61_TON.Q ,                   (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,           (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,              (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,                    (* Consigna de posición en automático *)
        KH=>VP_K61 ,                                             (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K62 );                                            (* Salida digital - '0' Subir / '1' Bajar persiana *)
```

```
(* Llamada a FB de gestión Motor Ventana Persiana 07 - VPM07 *)
VP_B71_TON(IN:=VP_B71 , PT:=T#600ms , Q=> , ET=> );
VP_B72_TON(IN:=VP_B72 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana07(          Q:=VP_Q07 ,                    (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B72_TON.Q ,             (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B71_TON.Q ,             (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                   (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,     (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,        (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,              (* Consigna de posición en automático *)
        KH=>VP_K71 ,                                       (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K72 );                                      (* Salida digital - '0' Subir / '1' Bajar persiana *)
(* Llamada a FB de gestión Motor Ventana Persiana 08 - VPM08 *)
VP_B81_TON(IN:=VP_B81 , PT:=T#600ms , Q=> , ET=> );
VP_B82_TON(IN:=VP_B82 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana08(          Q:=VP_Q08 ,                    (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B82_TON.Q ,             (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B81_TON.Q ,             (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                   (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,     (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,        (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,              (* Consigna de posición en automático *)
        KH=>VP_K81 ,                                       (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K82 );                                      (* Salida digital - '0' Subir / '1' Bajar persiana *)
(* Llamada a FB de gestión Motor Ventana Persiana 09- VPM09 *)
VP_B91_TON(IN:=VP_B91 , PT:=T#600ms , Q=> , ET=> );
VP_B92_TON(IN:=VP_B92 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana09(          Q:=VP_Q09 ,                    (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B92_TON.Q ,             (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B91_TON.Q ,             (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                   (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,     (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,        (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,              (* Consigna de posición en automático *)
        KH=>VP_K91 ,                                       (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K92 );                                      (* Salida digital - '0' Subir / '1' Bajar persiana *)
```

```
(* Llamada a FB de gestión Motor Ventana Persiana 10 – VPM10 *)
VP_B101_TON(IN:=VP_B101 , PT:=T#600ms , Q=> , ET=> );
VP_B102_TON(IN:=VP_B102 , PT:=T#5s , Q=> , ET=> );
VentanaPersiana10(          Q:=VP_Q10 ,                          (* Entrada digital - Térmico alimentación motor *)
                            BP:=VP_B102_TON.Q ,                          (* Entrada digital - Detector persiana abajo *)
                            BQ:=VP_B101_TON.Q ,                          (* Entrada digital - Detector persiana arriba *)
                            SRea:=G_Rea,                         (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                            SMan_Aut:=MAIN.Generales.S_.Man_Aut , (* Pulsador general Manual/Automático *)
                            WDef:=MAIN.Generales.EMG ,           (* Programa seguridad HW - Señal Emergencia instalación *)
                            WAut:= ,
                            TRecorridox10ms:=1900 ,              (* Tiempo total recorrido motor en cseg *)
                            SPAut:=VP_SPAut ,                    (* Consigna de posición en automático *)
        KH=>VP_K101 ,                                            (* Salida digital - Habilitar marcha persiana *)
        KM=>VP_K102 );                                           (* Salida digital - '0' Subir / '1' Bajar persiana *)


(* Llamada a FB de interconexión Profibus de variables "Generales" desde PLC1 a PLC2  *)
(*AZIMUTAL*)
(*Cmd.0=1 -> Se ejecuta el movimiento*)
IF NOT PLC2_X.AZ.Axis_Cmd.0 THEN
        AZ_VCCorrecciones:=0;
END_IF;
IF (PLC2_W.AZ.Axis_Sta.3 AND NOT AZ_MF1) THEN
        AZ_VCCorrecciones:=AZ_VCCorrecciones+1;
END_IF;
AZ_MF1:=PLC2_W.AZ.Axis_Sta.3;
IF (
        (PLC2_X.AZ.Axis_Cmd.0 AND PLC2_W.AZ.Axis_Sta.3 AND (ABS(PLC2_W.AZ.Pos_Mes - PLC2_X.AZ.PosSP_Par) < 2))
        OR (AZ_VCCorrecciones=3)
)THEN
(*
AZ_VCCorrecciones=1 -> Primer movimiento
AZ_VCCorrecciones=2 -> Primera corrección
AZ_VCCorrecciones=3 -> Segunda corrección
*)
        PLC2_X.AZ.Axis_Cmd.0:=0;
END_IF;
(*Azimutal-Comando Home*)
IF PLC2_X.AZ.Axis_Cmd.1 THEN
        PLC2_X.AZ.PosSP_Par:=0;
        PLC2_X.AZ.JogSpd_Par:=150;
        PLC2_X.AZ.Axis_Cmd.1:=0;
        PLC2_X.AZ.Axis_Cmd.0:=1;
END_IF;
```

```
GEN_PLC1_PLC2(                  ReaPLC1:=Generales.KRea_Remoto ,
        PLC2_Gen_Cmd1=>PLC2_X.GEN.Gen_Cmd1 ,
        PLC2_Gen_Cmd2=>PLC2_X.GEN.Gen_Cmd2 );

(*Guarda variables persistentes en Compact Flash*)
(* No es necesario utilizar ninguna patilla de entrada, con llamar el bloque es suficiente *)
UPS1(                                   sNetID:= ,
                                                iPLCPort:= ,
                                                iUPSPort:= ,
                                                tTimeout:= ,
                                                eUpsMode:= ,
                                                ePersistentMode:= ,
                                                tRecoverTime:= ,
        bPowerFailDetect=> ,eState=> );
```

8.1.8 Estructuras de datos

```
TYPE Gen_D :
STRUCT
        Q_Cupula: BOOL; (* Defecto térmico alimentación cuadro remoto: Cúpula *)
        F220: BOOL; (* Defecto térmico alimentación 220VAC *)
        TM01: BOOL; (* Defecto termostato ventilador armario *)
END_STRUCT
END_TYPE
```

```
TYPE Gen_S :
STRUCT
        Man_Aut: BOOL; (* Cambiar modo Manual/Automático *)
        ReaWeb: BOOL; (* Pulsador rearme general instalación - web *)
        HabilitarKV: BOOL; (* Habilitar relé alimentación 220VAC ventilador armario *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Gen_Comandos :
STRUCT
        Gen_Cmd1: WORD; (* Generales - Comandos binarios 1 *)
        Gen_Cmd2: WORD; (* Generales - Comandos binarios 2 *)
        Res_Word03: INT; (* Reserva *)
        Res_Word04: INT; (* Reserva *)
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Gen_Estados :
STRUCT
        Gen_Sta1: WORD; (* Generales - Bits de Estados 1 *)
        Gen_Sta2: WORD; (* Generales - Bits de Estados 2 *)
        Res_Word03: INT; (* Reserva *)
        Res_Word04: INT; (* Reserva *)
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_PLC1_PLC2 :
STRUCT
        GEN: Profibus_Gen_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Generales *)
        AZ: Profibus_Servo_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Azimutal *)
        CR: Profibus_Servo_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Compuerta Rendija *)
        WS: Profibus_Servo_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Windshield *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_PLC2_PLC1 :
STRUCT
        GEN: Profibus_Gen_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Generales *)
        AZ: Profibus_Servo_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Azimutal *)
        CR: Profibus_Servo_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Compuerta Rendija *)
        WS: Profibus_Servo_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Windshield *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Servo_Comandos :
STRUCT
        Axis_Cmd: WORD; (* Axis bits commands *)
        PosSP_Par: UINT; (* Axis Position SetPoint (1/100 deg) *)
        JogSpd_Par: INT; (* Speed sepoint for jog move (1/100 deg) *)
        Res_Word04: INT; (* Reserva *)
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Servo_Estados :
STRUCT
        Axis_Sta: WORD; (* Axis bits status *)
        Pos_Mes: UINT; (* Axis actual position (1/100 deg) *)
        Cur_Mes: INT; (* Actual current (1/10 A) *)
        Res_Word04: INT; (* Reserva *)
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

8.1.9 Variables globales

```
VAR_GLOBAL
    CU_Q AT %I* : BOOL; (* Armario remoto Cúpula - Térmico alimentación acometida - EL1809 *)
    G_F220 AT %I* : BOOL; (* General - Térmico alimentación 220VAC - EL1809 *)
    G_EMG1 AT %I* : BOOL; (* General - Emergencia 1 planta telescopio escalera *)
    G_EMG2 AT %I* : BOOL; (* General - Emergencia 2 planta telescopio montacargas *)
    G_EMG3 AT %I* : BOOL; (* General - Emergencia armario fijo *)
    G_EMG4 AT %I* : BOOL; (* General - Emergencia sala técnica *)
    G_EMG5 AT %I* : BOOL; (* General - Emergencia AMOS telescopio *)
    G_EMG6 AT %I* : BOOL; (* General - Emergencia armario móvil a bordo de la cúpula *)
    VC_Q01 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 1 - EL1809 *)
    VP_Q01 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 1 - EL1809 *)
    VC_Q02 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 2 - EL1809 *)
    VP_Q02 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 2 - EL1809 *)
    VC_Q03 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 3 - EL1809 *)
    VP_Q03 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 3 - EL1809 *)
    VC_Q04 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 4 - EL1809 *)
    VP_Q04 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 4 - EL1809 *)
    VC_Q05 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 5 - EL1809 *)
    VP_Q05 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 5 - EL1809 *)
    VC_Q06 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 6 - EL1809 *)
    VP_Q06 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 6 - EL1809 *)
    VC_Q07 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 7 - EL1809 *)
    VP_Q07 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 7 - EL1809 *)
    VC_Q08 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 8 - EL1809 *)
    VP_Q08 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 8 - EL1809 *)
    VC_Q09 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 9 - EL1809 *)
    VP_Q09 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 9 - EL1809 *)
    VC_Q10 AT %I* : BOOL; (* Ventana Compuerta - Térmico alimentación motor 10 - EL1809 *)
    VP_Q10 AT %I* : BOOL; (* Ventana Persiana - Térmico alimentación motor 10 - EL1809 *)
    G_TM01 AT %I* : BOOL; (* General - Termostato armario general - EL1809 *)
    G_SH01 AT %I* : BOOL; (* General - Pulsador rearme emergencia 1 - EL1809 *)
    G_SH02 AT %I* : BOOL; (* General - Pulsador rearme emergencia 2 - EL1809 *)
    G_SH03 AT %I* : BOOL; (* General - Pulsador rearme emergencia 3 - EL1809 *)
    G_SH04 AT %I* : BOOL; (* General - Pulsador rearme emergencia 4 - EL1809 *)
    G_SH05 AT %I* : BOOL; (* General - Pulsador rearme emergencia 5 - EL1809 *)
    VC_FC11 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 1 compuerta abierta - EL1809 *)
    VC_FC12 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 1 compuerta cerrada - EL1809 *)
    VP_B11 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 1 persiana arriba - EL1809 *)
    VP_B12 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 1 persiana abajo - EL1809 *)
    VC_FC21 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 2 compuerta abierta - EL1809 *)
    VC_FC22 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 2 compuerta cerrada - EL1809 *)
    VP_B21 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 2 persiana arriba - EL1809 *)
    VP_B22 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 2  persiana abajo - EL1809 *)
    VC_FC31 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 3 compuerta abierta - EL1809 *)
```

```
VC_FC32 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 3 compuerta cerrada - EL1809 *)
VP_B31 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 3 persiana arriba - EL1809 *)
VP_B32 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 3 persiana abajo - EL1809 *)
VC_FC41 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 4 compuerta abierta - EL1809 *)
VC_FC42 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 4 compuerta cerrada - EL1809 *)
VP_B41 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 4 persiana arriba - EL1809 *)
VP_B42 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 4 persiana abajo - EL1809 *)
VC_FC51 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 5 compuerta abierta - EL1809 *)
VC_FC52 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 5 compuerta cerrada - EL1809 *)
VP_B51 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 5 persiana arriba - EL1809 *)
VP_B52 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 5 persiana abajo - EL1809 *)
VC_FC61 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 6 compuerta abierta - EL1809 *)
VC_FC62 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 6 compuerta cerrada - EL1809 *)
VP_B61 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 6 persiana arriba - EL1809 *)
VP_B62 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 6 persiana abajo - EL1809 *)
VC_FC71 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 7 compuerta abierta - EL1809 *)
VC_FC72 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 7 compuerta cerrada - EL1809 *)
VP_B71 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 7 persiana arriba - EL1809 *)
VP_B72 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 7 persiana abajo - EL1809 *)
VC_FC81 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 8 compuerta abierta - EL1809 *)
VC_FC82 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 8 compuerta cerrada - EL1809 *)
VP_B81 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 8 persiana arriba - EL1809 *)
VP_B82 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 8 persiana abajo - EL1809 *)
VC_FC91 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 9 compuerta abierta - EL1809 *)
VC_FC92 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 9 compuerta cerrada - EL1809 *)
VP_B91 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 9 persiana arriba - EL1809 *)
VP_B92 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 9 persiana abajo - EL1809 *)
VC_FC101 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 10 compuerta abierta - EL1809 *)
VC_FC102 AT %I* : BOOL; (* Ventana Compuerta - Final de carrera 10 compuerta cerrada - EL1809 *)
VP_B101 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 10 persiana arriba - EL1809 *)
VP_B102 AT %I* : BOOL; (* Ventana Persiana - Detector presencia 10 persiana abajo - EL1809 *)
VC_K11 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 1 - EL2809 *)
VC_K12 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 1 - EL2809 *)
VP_K11 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 1 - EL2809 *)
VP_K12 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 1 - EL2809 *)
VC_K21 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 2 - EL2809 *)
VC_K22 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 2 - EL2809 *)
VP_K21 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 2 - EL2809 *)
VP_K22 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 2 - EL2809 *)
VC_K31 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 3 - EL2809 *)
VC_K32 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 3 - EL2809 *)
VP_K31 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 3 - EL2809 *)
VP_K32 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 3 - EL2809 *)
VC_K41 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 4 - EL2809 *)
VC_K42 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 4 - EL2809 *)
VP_K41 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 4 - EL2809 *)
```

```
VP_K42 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 4 - EL2809 *)
VC_K51 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 5 - EL2809 *)
VC_K52 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 5 - EL2809 *)
VP_K51 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 5 - EL2809 *)
VP_K52 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 5 - EL2809 *)
VC_K61 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 6 - EL2809 *)
VC_K62 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 6 - EL2809 *)
VP_K61 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 6 - EL2809 *)
VP_K62 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 6 - EL2809 *)
VC_K71 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 7 - EL2809 *)
VC_K72 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 7 - EL2809 *)
VP_K71 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 7 - EL2809 *)
VP_K72 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 7 - EL2809 *)
VC_K81 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 8 - EL2809 *)
VC_K82 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 8 - EL2809 *)
VP_K81 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 8 - EL2809 *)
VP_K82 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 8 - EL2809 *)
VC_K91 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 9 - EL2809 *)
VC_K92 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 9 - EL2809 *)
VP_K91 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 9 - EL2809 *)
VP_K92 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 9 - EL2809 *)
VC_K101 AT %Q* : BOOL; (* Ventana Compuerta - Habilitar marcha compuerta 10 - EL2809 *)
VC_K102 AT %Q* : BOOL; (* Ventana Compuerta - '0' Abrir / '1' Cerrar compuerta 10 - EL2809 *)
VP_K101 AT %Q* : BOOL; (* Ventana Persiana - Habilitar marcha persiana 10 - EL2809 *)
VP_K102 AT %Q* : BOOL; (* Ventana Persiana - '0' Subir / '1' Bajar persiana 10 - EL2809 *)
G_KV AT %Q* : BOOL; (* General - Marcha relé ventilador armario - EL2809 *)
G_H01 AT %Q* : BOOL; (* General - Piloto rearme emergencia 1 - EL2809 *)
G_H02 AT %Q* : BOOL; (* General - Piloto rearme emergencia 2 - EL2809 *)
G_H03 AT %Q* : BOOL; (* General - Piloto rearme emergencia 3 - EL2809 *)
G_H04 AT %Q* : BOOL; (* General - Piloto rearme emergencia 4 - EL2809 *)
G_H05 AT %Q* : BOOL; (* General - Piloto rearme emergencia 5 - EL2809 *)
G_Rea AT %Q* : BOOL; (* General - Rearme general instalación - EL6900 *)
G_EMG AT %I* : BOOL; (* General - Emergencia Instalación - EL6900 *)
G_ARM: BOOL; (* General - Arranque secuencial motores *)
G_MotoresActivos: INT; (* General - Número de motores en marcha (máximo 10) *)
VC_SPAut: REAL; (* Ventana Compuerta - SP Automático *)
VP_SPAut: REAL; (* Ventana Persiana - SP Automático *)
UNO: BOOL; (* Variable siempre a '1' *)
CERO: BOOL; (* Variable siempre a '0' *)
PLC2_X AT %Q* : Profibus_PLC1_PLC2; (* Interconexión Profibus PLC1 -> PLC2 Comandos (32 Word) *)
PLC2_W AT %I* : Profibus_PLC2_PLC1; (* Interconexión Profibus PLC2 -> PLC1 Estados (32 Word) *)
_044_Pos1_Mes AT %Q* : INT; (* DCS - Window 1 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos2_Mes AT %Q* : INT; (* DCS - Window 2 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos3_Mes AT %Q* : INT; (* DCS - Window 3 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos4_Mes AT %Q* : INT; (* DCS - Window 4 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos5_Mes AT %Q* : INT; (* DCS - Window 5 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
```

_044_Pos6_Mes AT %Q* : INT; (* DCS - Window 6 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos7_Mes AT %Q* : INT; (* DCS - Window 7 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos8_Mes AT %Q* : INT; (* DCS - Window 8 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos9_Mes AT %Q* : INT; (* DCS - Window 9 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_Pos10_Mes AT %Q* : INT; (* DCS - Window 10 actual aperture (0: Deconocido; 1000: Completely Closed; 2000: Completely Opened) *)
_044_IMV1_Sta AT %Q* : BOOL; (* DCS - '1': Window 1 is moving. '0': Window 1 is stopped *)
_044_IMV2_Sta AT %Q* : BOOL; (* DCS - '1': Window 2 is moving. '0': Window 2 is stopped *)
_044_IMV3_Sta AT %Q* : BOOL; (* DCS - '1': Window 3 is moving. '0': Window 3 is stopped *)
_044_IMV4_Sta AT %Q* : BOOL; (* DCS - '1': Window 4 is moving. '0': Window 4 is stopped *)
_044_IMV5_Sta AT %Q* : BOOL; (* DCS - '1': Window 5 is moving. '0': Window 5 is stopped *)
_044_IMV6_Sta AT %Q* : BOOL; (* DCS - '1': Window 6 is moving. '0': Window 6 is stopped *)
_044_IMV7_Sta AT %Q* : BOOL; (* DCS - '1': Window 7 is moving. '0': Window 7 is stopped *)
_044_IMV8_Sta AT %Q* : BOOL; (* DCS - '1': Window 8 is moving. '0': Window 8 is stopped *)
_044_IMV9_Sta AT %Q* : BOOL; (* DCS - '1': Window 9 is moving. '0': Window 9 is stopped *)
_044_IMV10_Sta AT %Q* : BOOL; (* DCS - '1': Window 10 is moving. '0': Window 10 is stopped *)
_044_Ok1_Sta AT %Q* : BOOL; (* DCS - '1': Window 1 is in good health. '0': Warning - Window 1 needs maintenance *)
_044_Ok2_Sta AT %Q* : BOOL; (* DCS - '1': Window 2 is in good health. '0': Warning - Window 2 needs maintenance *)
_044_Ok3_Sta AT %Q* : BOOL; (* DCS - '1': Window 3 is in good health. '0': Warning - Window 3 needs maintenance *)
_044_Ok4_Sta AT %Q* : BOOL; (* DCS - '1': Window 4 is in good health. '0': Warning - Window 4 needs maintenance *)
_044_Ok5_Sta AT %Q* : BOOL; (* DCS - '1': Window 5 is in good health. '0': Warning - Window 5 needs maintenance *)
_044_Ok6_Sta AT %Q* : BOOL; (* DCS - '1': Window 6 is in good health. '0': Warning - Window 6 needs maintenance *)
_044_Ok7_Sta AT %Q* : BOOL; (* DCS - '1': Window 7 is in good health. '0': Warning - Window 7 needs maintenance *)
_044_Ok8_Sta AT %Q* : BOOL; (* DCS - '1': Window 8 is in good health. '0': Warning - Window 8 needs maintenance *)
_044_Ok9_Sta AT %Q* : BOOL; (* DCS - '1': Window 9 is in good health. '0': Warning - Window 9 needs maintenance *)
_044_Ok10_Sta AT %Q* : BOOL; (* DCS - '1': Window 10 is in good health. '0': Warning - Window 10 needs maintenance *)
_044_Flt1_Sta AT %Q* : BOOL; (* DCS - '1': Fault 1: Window 1 in fault. '0': Fault 1: No fault on window 1 *)
_044_Flt2_Sta AT %Q* : BOOL; (* DCS - '1': Fault 2: Window 2 in fault. '0': Fault 2: No fault on window 2 *)
_044_Flt3_Sta AT %Q* : BOOL; (* DCS - '1': Fault 3: Window 3 in fault. '0': Fault 3: No fault on window 3 *)
_044_Flt4_Sta AT %Q* : BOOL; (* DCS - '1': Fault 4: Window 4 in fault. '0': Fault 4: No fault on window 4 *)
_044_Flt5_Sta AT %Q* : BOOL; (* DCS - '1': Fault 5: Window 5 in fault. '0': Fault 5: No fault on window 5 *)
_044_Flt6_Sta AT %Q* : BOOL; (* DCS - '1': Fault 6: Window 6 in fault. '0': Fault 6: No fault on window 6 *)
_044_Flt7_Sta AT %Q* : BOOL; (* DCS - '1': Fault 7: Window 7 in fault. '0': Fault 7: No fault on window 7 *)
_044_Flt8_Sta AT %Q* : BOOL; (* DCS - '1': Fault 8: Window 8 in fault. '0': Fault 8: No fault on window 8 *)
_044_Flt9_Sta AT %Q* : BOOL; (* DCS - '1': Fault 9: Window 9 in fault. '0': Fault 9: No fault on window 9 *)
_044_Flt10_Sta AT %Q* : BOOL; (* DCS - '1': Fault 10: Window 10 in fault. '0': Fault 10: No fault on window 10 *)
_044_Pos1_Par AT %I* : INT; (* DCS - Window 1 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos2_Par AT %I* : INT; (* DCS - Window 2 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos3_Par AT %I* : INT; (* DCS - Window 3 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos4_Par AT %I* : INT; (* DCS - Window 4 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos5_Par AT %I* : INT; (* DCS - Window 5 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos6_Par AT %I* : INT; (* DCS - Window 6 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos7_Par AT %I* : INT; (* DCS - Window 7 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos8_Par AT %I* : INT; (* DCS - Window 8 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos9_Par AT %I* : INT; (* DCS - Window 9 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_044_Pos10_Par AT %I* : INT; (* DCS - Window 10 aperture setpoint (0 Stop, 1000 Completely Close, 2000 Completely Open) *)
_045_IMV1_Sta AT %Q* : BOOL; (* DCS - '1' Louver 1 is moving. '0' Louver 1 is stopped *)

```
_045_IMV2_Sta AT %Q* : BOOL; (* DCS - '1' Louver 2 is moving. '0' Louver 2 is stopped *)
_045_IMV3_Sta AT %Q* : BOOL; (* DCS - '1' Louver 3 is moving. '0' Louver 3 is stopped *)
_045_IMV4_Sta AT %Q* : BOOL; (* DCS - '1' Louver 4 is moving. '0' Louver 4 is stopped *)
_045_IMV5_Sta AT %Q* : BOOL; (* DCS - '1' Louver 5 is moving. '0' Louver 5 is stopped *)
_045_IMV6_Sta AT %Q* : BOOL; (* DCS - '1' Louver 6 is moving. '0' Louver 6 is stopped *)
_045_IMV7_Sta AT %Q* : BOOL; (* DCS - '1' Louver 7 is moving. '0' Louver 7 is stopped *)
_045_IMV8_Sta AT %Q* : BOOL; (* DCS - '1' Louver 8 is moving. '0' Louver 8 is stopped *)
_045_IMV9_Sta AT %Q* : BOOL; (* DCS - '1' Louver 9 is moving. '0' Louver 9 is stopped *)
_045_IMV10_Sta AT %Q* : BOOL; (* DCS - '1' Louver 10 is moving. '0' Louver 10 is stopped *)
_045_Ok1_Sta AT %Q* : BOOL; (* DCS - '1' Louver 1 is in good health, '0' Warning - Louver 1 needs maintenance *)
_045_Ok2_Sta AT %Q* : BOOL; (* DCS - '1' Louver 2 is in good health, '0' Warning - Louver 2 needs maintenance *)
_045_Ok3_Sta AT %Q* : BOOL; (* DCS - '1' Louver 3 is in good health, '0' Warning - Louver 3 needs maintenance *)
_045_Ok4_Sta AT %Q* : BOOL; (* DCS - '1' Louver 4 is in good health, '0' Warning - Louver 4 needs maintenance *)
_045_Ok5_Sta AT %Q* : BOOL; (* DCS - '1' Louver 5 is in good health, '0' Warning - Louver 5 needs maintenance *)
_045_Ok6_Sta AT %Q* : BOOL; (* DCS - '1' Louver 6 is in good health, '0' Warning - Louver 6 needs maintenance *)
_045_Ok7_Sta AT %Q* : BOOL; (* DCS - '1' Louver 7 is in good health, '0' Warning - Louver 7 needs maintenance *)
_045_Ok8_Sta AT %Q* : BOOL; (* DCS - '1' Louver 8 is in good health, '0' Warning - Louver 8 needs maintenance *)
_045_Ok9_Sta AT %Q* : BOOL; (* DCS - '1' Louver 9 is in good health, '0' Warning - Louver 9 needs maintenance *)
_045_Ok10_Sta AT %Q* : BOOL; (* DCS - '1' Louver 10 is in good health, '0' Warning - Louver 10 needs maintenance *)
_045_Ope1_Sta AT %Q* : BOOL; (* DCS - '1' Louver 1 completely opened. '0' Louver 1 not completely opened *)
_045_Ope2_Sta AT %Q* : BOOL; (* DCS - '1' Louver 2 completely opened. '0' Louver 2 not completely opened *)
_045_Ope3_Sta AT %Q* : BOOL; (* DCS - '1' Louver 3 completely opened. '0' Louver 3 not completely opened *)
_045_Ope4_Sta AT %Q* : BOOL; (* DCS - '1' Louver 4 completely opened. '0' Louver 4 not completely opened *)
_045_Ope5_Sta AT %Q* : BOOL; (* DCS - '1' Louver 5 completely opened. '0' Louver 5 not completely opened *)
_045_Ope6_Sta AT %Q* : BOOL; (* DCS - '1' Louver 6 completely opened. '0' Louver 6 not completely opened *)
_045_Ope7_Sta AT %Q* : BOOL; (* DCS - '1' Louver 7 completely opened. '0' Louver 7 not completely opened *)
_045_Ope8_Sta AT %Q* : BOOL; (* DCS - '1' Louver 8 completely opened. '0' Louver 8 not completely opened *)
_045_Ope9_Sta AT %Q* : BOOL; (* DCS - '1' Louver 9 completely opened. '0' Louver 9 not completely opened *)
_045_Ope10_Sta AT %Q* : BOOL; (* DCS - '1' Louver 10 completely opened. '0' Louver 10 not completely opened *)
_045_Clo1_Sta AT %Q* : BOOL; (* DCS - '1' Louver 1 completely closed. '0' Louver 1 not completely closed *)
_045_Clo2_Sta AT %Q* : BOOL; (* DCS - '1' Louver 2 completely closed. '0' Louver 2 not completely closed *)
_045_Clo3_Sta AT %Q* : BOOL; (* DCS - '1' Louver 3 completely closed. '0' Louver 3 not completely closed *)
_045_Clo4_Sta AT %Q* : BOOL; (* DCS - '1' Louver 4 completely closed. '0' Louver 4 not completely closed *)
_045_Clo5_Sta AT %Q* : BOOL; (* DCS - '1' Louver 5 completely closed. '0' Louver 5 not completely closed *)
_045_Clo6_Sta AT %Q* : BOOL; (* DCS - '1' Louver 6 completely closed. '0' Louver 6 not completely closed *)
_045_Clo7_Sta AT %Q* : BOOL; (* DCS - '1' Louver 7 completely closed. '0' Louver 7 not completely closed *)
_045_Clo8_Sta AT %Q* : BOOL; (* DCS - '1' Louver 8 completely closed. '0' Louver 8 not completely closed *)
_045_Clo9_Sta AT %Q* : BOOL; (* DCS - '1' Louver 9 completely closed. '0' Louver 9 not completely closed *)
_045_Clo10_Sta AT %Q* : BOOL; (* DCS - '1' Louver 10 completely closed. '0' Louver 10 not completely closed *)
_045_Flt1_Sta AT %Q* : BOOL; (* DCS - '1' Fault 1: Louver 1 in fault. '0' Fault 1: No fault on louver 1 *)
_045_Flt2_Sta AT %Q* : BOOL; (* DCS - '1' Fault 2: Louver 2 in fault. '0' Fault 2: No fault on louver 2 *)
_045_Flt3_Sta AT %Q* : BOOL; (* DCS - '1' Fault 3: Louver 3 in fault. '0' Fault 3: No fault on louver 3 *)
_045_Flt4_Sta AT %Q* : BOOL; (* DCS - '1' Fault 4: Louver 4 in fault. '0' Fault 4: No fault on louver 4 *)
_045_Flt5_Sta AT %Q* : BOOL; (* DCS - '1' Fault 5: Louver 5 in fault. '0' Fault 5: No fault on louver 5 *)
_045_Flt6_Sta AT %Q* : BOOL; (* DCS - '1' Fault 6: Louver 6 in fault. '0' Fault 6: No fault on louver 6 *)
_045_Flt7_Sta AT %Q* : BOOL; (* DCS - '1' Fault 7: Louver 7 in fault. '0' Fault 7: No fault on louver 7 *)
```

```
        _045_Flt8_Sta AT %Q* : BOOL; (* DCS - '1' Fault 8: Louver 8 in fault. '0' Fault 8: No fault on louver 8 *)
        _045_Flt9_Sta AT %Q* : BOOL; (* DCS - '1' Fault 9: Louver 9 in fault. '0' Fault 9: No fault on louver 9 *)
        _045_Flt10_Sta AT %Q* : BOOL; (* DCS - '1' Fault 10: Louver 10 in fault. '0' Fault 10: No fault on louver 10 *)
        _045_Ope1_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 1. '0' Stop de movement *)
        _045_Ope2_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 2. '0' Stop de movement *)
        _045_Ope3_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 3. '0' Stop de movement *)
        _045_Ope4_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 4. '0' Stop de movement *)
        _045_Ope5_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 5. '0' Stop de movement *)
        _045_Ope6_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 6. '0' Stop de movement *)
        _045_Ope7_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 7. '0' Stop de movement *)
        _045_Ope8_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 8. '0' Stop de movement *)
        _045_Ope9_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 9. '0' Stop de movement *)
        _045_Ope10_Cmd AT %I* : BOOL; (* DCS - '1' Open the Louver 10. '0' Stop de movement *)
        _045_Clo1_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 1. '0' Stop de movement *)
        _045_Clo2_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 2. '0' Stop de movement *)
        _045_Clo3_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 3. '0' Stop de movement *)
        _045_Clo4_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 4. '0' Stop de movement *)
        _045_Clo5_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 5. '0' Stop de movement *)
        _045_Clo6_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 6. '0' Stop de movement *)
        _045_Clo7_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 7. '0' Stop de movement *)
        _045_Clo8_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 8. '0' Stop de movement *)
        _045_Clo9_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 9. '0' Stop de movement *)
        _045_Clo10_Cmd AT %I* : BOOL; (* DCS - '1' Close the Louver 10. '0' Stop de movement *)
END_VAR
```

8.2 PLC2: código de programación

Estructura de programa

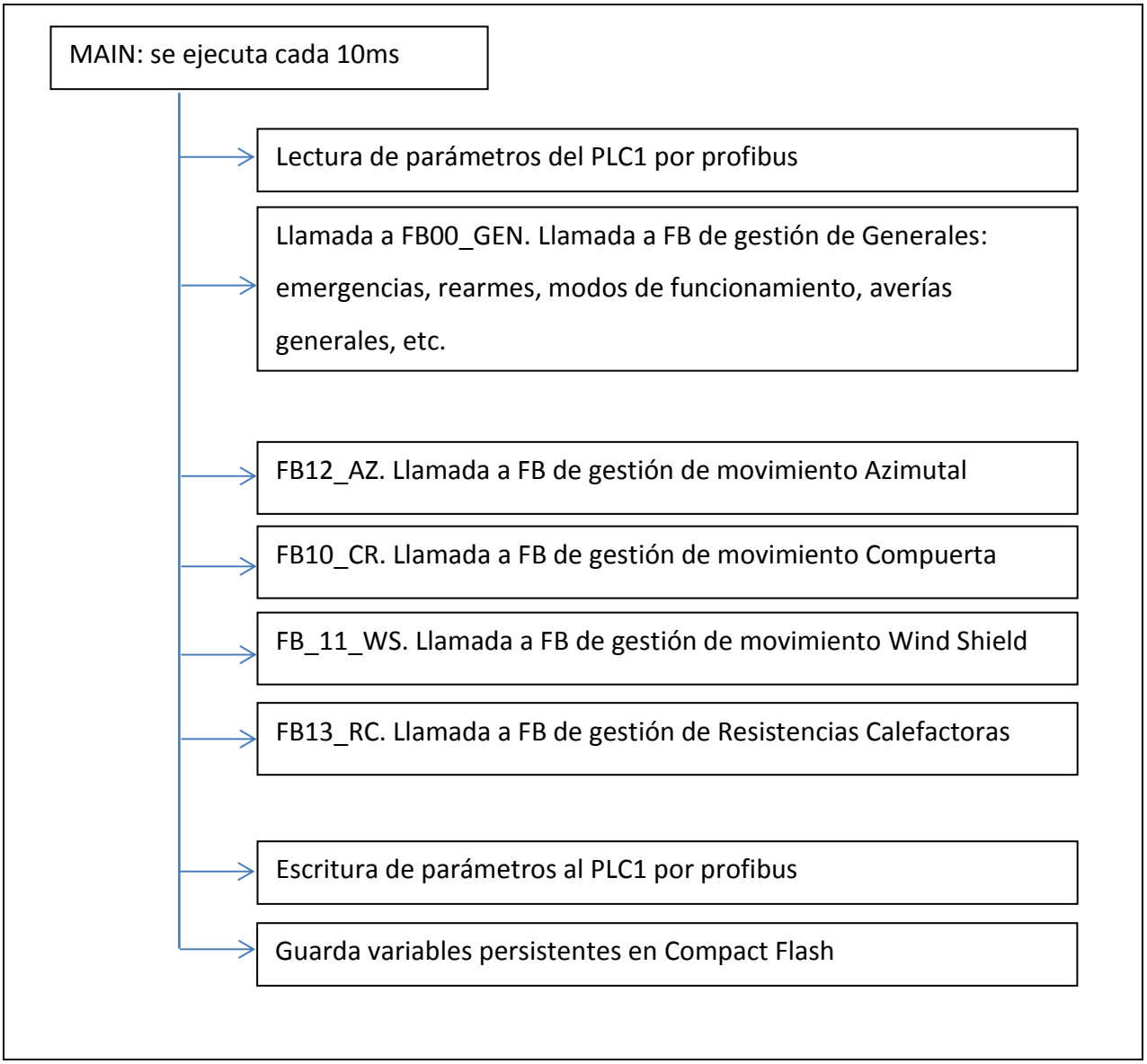

Fig.8.2.1 – PLC2 Estructura de programa

8.2.1 FB00_GEN

```
FUNCTION_BLOCK FB00_GEN
VAR
        Man: BOOL; (* Modo Manual *)
        Aut: BOOL; (* Modo Automático *)
        Def: BOOL; (* Reunión Defectos *)
        int_500ms: BOOL; (* General - Intermitencia 500ms *)
        cont_1s: INT; (* General - Contador auxiliar 1s *)
        D_: Gen_D; (* Defectos Generales *)
        S_: Gen_S; (* Pulsadores Generales *)
END_VAR
VAR_INPUT
        EMG: BOOL; (* Seta de emergencia *)
        SRea: BOOL; (* Pulsador de rearme *)
        TM01: BOOL; (* Termostato armario general *)
END_VAR
VAR_OUTPUT
        KV: BOOL; (* Marcha contactor alimentación ventilador armario *)
        KRea: BOOL; (* Rearme general instalación - programa de seguridad *)
        KRea_Remoto: BOOL; (* Rearme general instalación - programa de seguridad remoto *)
        HRea: BOOL; (* Piloto emergencia general *)
END_VAR
```

```
(* Gestión de Generales de la instalación *)
(****************************************** Previo ************************************************)
(* Bits Generales *)
CERO := 0;
UNO := 1;
(* Generación de ciclos de tiempo *)
cont_1s:=cont_1s+1;
IF cont_1s > 100 THEN cont_1s:=0; END_IF;
(* Intermitencia 500ms *)
IF cont_1s <= 50 THEN int_500ms:=1; ELSE int_500ms:=0; END_IF;

(****************************************** Modos *************************************************)
(* Modos Manual - Automático *)
Man:=NOT Aut;
Aut:=MAIN.Compuerta_Rendija.Aut AND MAIN.Windshield.Aut AND MAIN.Azimutal.Aut;

(************************************** Rearme general ********************************************)
(* Rearme general *)
KRea:=S_.ReaWeb OR SRea OR MAIN.GEN_PLC1_PLC2.ReaPLC1;
KRea_Remoto:=S_.ReaWeb OR SRea;
```

```
(* Rearme Defectos *)
IF KRea THEN
        D_.TM01:=0;
END_IF;

(************************************** Defectos ***************************************************)
(* Defectos HW *)
IF NOT TM01 THEN D_.TM01:=1; END_IF;          (* Defecto Termostato ventilador armario *)
(* Reunión Defectos *)
Def:=D_.TM01;

(********************************** Flags/Habilitaciones ***************************************)
(* Habilitar contactor alimentación 220VAC relé alimentación ventilador armario  *)
KV:=S_.HabilitarKV ;
(* Piloto Emergencia General *)
HRea:=EMG AND int_500ms;
```

8.2.2 FB12_AZ. Llamada a FB de gestión de movimiento Azimutal

```
FUNCTION_BLOCK FB12_AZ
VAR
	Man: BOOL; (* Modo Manual *)
	Aut: BOOL; (* Modo Automático *)
	Def: BOOL; (* Reunión Defectos *)
	AuxDone0: BOOL; (* Auxiliar operación ejecutada con éxito *)
	AuxDone1: BOOL; (* Auxiliar operación ejecutada con éxito *)
	AuxDone2: BOOL; (* Auxiliar operación ejecutada con éxito *)
	HabControlPos: BOOL; (* Habilitar control de Posición: corrección con encoder externo *)
	AlternarMovPos: BOOL; (* Toogle para alternar el flag de habilitación de mover en posición *)
	cont_1: INT; (* Contador auxiliar *)
	cont_2: INT; (* Contador auxiliar *)
	Paso: INT; (* Número de paso en modo Automático *)
	SP_Vel: LREAL; (* Setpoint de velocidad *)
	SP_PosTarget: LREAL; (* Setpoint de posición objetivo *)
	SP_PosActual: LREAL; (* Setpoint de posición actual *)
	SP_Aut: LREAL; (* Setpoint de posición en Automático *)
	SP_Aut_HMI: LREAL; (* Setpoint de posición en Automático *)
	SP_Man: LREAL; (* Setpoint de posición en Manual *)
	PosAct_0_360: LREAL; (* Posición actual encoder Servos 0º - 360º *)
	PosCorr: LREAL; (* Posición a corregir por desfase entre encoder Vahle y encoder Servos *)
	PosError: LREAL; (* Error de posición actual entre encoder Vahle y encoder Servos *)
	AZM01: Servo; (* Control Servomotor Azimutal 1 - AZM01 *)
	AZM02: Servo; (* Control Servomotor Azimutal 2 - AZM02 *)
	D_: Servo_D; (* Defectos Servomotor *)
	S_: Servo_S; (* Pulsadores Servomotor *)
	Vahle: Vahle; (* Encoder absoluto carril Vahle *)
	AZM01_PAR: FB22_FiltroAnal; (* Motor azimut 1 Par filtrado *)
	AZM02_PAR: FB22_FiltroAnal; (* Motor azimut 2 Par filtrado *)
	AZM01_Intensidad: FB22_FiltroAnal; (* Motor azimut 1 Intensidad filtrada *)
	AZM02_Intensidad: FB22_FiltroAnal; (* Motor azimut 2 Intensidad filtrada *)
	CalculoSPAut: FB20_MenorDistancia; (* Función auxiliar para el cálculo del SP Aut por el camino más corto *)
END_VAR
VAR_INPUT
	Q: BOOL; (* Térmico alimentación variadores *)
	WDef: BOOL; (* Defecto externo *)
	SRea: BOOL; (* Pulsador de rearme *)
	WAut: BOOL; (* Condiciones marcha automático *)
END_VAR
VAR_OUTPUT
	K: BOOL; (* Marcha contactor alimentación variadores *)
END_VAR
VAR PERSISTENT
	K_: Servo_K; (* Constantes Servomotor *)
```

```
END_VAR
(* FB control servomotores movimiento Azimutal *)
(****************************************** Previo ***************************************************)
(* Constantes *)
K_.VelTrabajo:=150;
Vahle.PosLMax_mm:=33554430;          (* Cota máxima encoder absoluto de Vahle *)
Vahle.PosLMin_mm:=33516468;          (* Cota mínima encoder absoluto de Vahle *)

(* Consignas *)
IF MAIN.GEN_PLC1_PLC2.AutRemoto THEN
        SP_Aut:=SP_Aut_HMI;                          (* SP de posición desde HMI *)
        K_.VelTrabajo:=150;                          (* Velocidad de trabajo *)
ELSE
        SP_Aut:=MAIN.AZ_PLC1_PLC2.SP_PosTarget;              (* SP de posición desde HMI *)
        K_.VelTrabajo:=MAIN.AZ_PLC1_PLC2.SP_Vel;             (* Velocidad de trabajo *)
END_IF;

(* Filtro señales analógicas *)
AZM01_PAR(AnalIn:=ABS(AZM01_AT.TorqueFeedbackValue));
AZM02_PAR(AnalIn:=ABS(AZM02_AT.TorqueFeedbackValue));
AZM01_Intensidad(AnalIn:=ABS(AZM01_AT.ActualAbsoluteCurrent));
AZM02_Intensidad(AnalIn:=ABS(AZM02_AT.ActualAbsoluteCurrent));

(* Gestión Encoder Carril Vahle *)
Vahle.PosAct_mm:=DWORD_TO_LREAL(AZ_PosVahle)*2;          (* Conversión a mm Encoder Vahle *)
Vahle.PosAct:=((Vahle.PosAct_mm - Vahle.PosLMin_mm) * (36000.0 / (Vahle.PosLMax_mm - Vahle.PosLMin_mm)));    (* Conversión valor de encoder
de banda Vahle mm -> centésima de grado *)

(* Cálculo auxiliar Posición actual encoder Servos de 0º a 360º *)
IF AZM01.Eje.NcToPlc.ActPos < 0.0 THEN
        PosAct_0_360:=AZM01.Eje.NcToPlc.ActPos + 36000.0;
ELSIF AZM01.Eje.NcToPlc.ActPos > 36000.0 THEN
        PosAct_0_360:=AZM01.Eje.NcToPlc.ActPos - 36000.0;
ELSE
        PosAct_0_360:=AZM01.Eje.NcToPlc.ActPos;
END_IF;

(****************************************** Modos ***************************************************)
(* Modos Manual - Automático *)
Man:=0;
Aut:=1;

(* Gestión Modo Automático - Manual *)
IF Aut THEN
        S_.HabilitarK:=1;                            (* Habilitar contactor de potencia *)
        IF S_.Fin OR (Def AND (Paso>12)) THEN        (* Pulsando Fin o Defecto en cualquier momento forzamos parada segura *)
```

```
                S_.Parar:=1;                    (* Habilitar flag de parada *)
                Paso:=10;                       (* Cambiar a Paso 10 *)
        END_IF;
CASE Paso OF
        0:      (* Inicio - Todo deshabilitado *)
                IF AZM01.ModoAcople<>0 OR AZM02.ModoAcople<>0 THEN
                        S_.Desacoplar:=1;                       (* Si algún motor acoplado -> desacoplar *)
                END_IF;
                IF NOT Def THEN
                        S_.Reset:=1;                            (* Habilitar reset de NC *)
                        Paso:=1;                                (* Cambiar a Paso 1 *)
                END_IF;

        1:      (* Esperar Drive Ready *)
                IF AZM01.ModoAcople<>0 OR AZM02.ModoAcople<>0 THEN
                        S_.Desacoplar:=1;                       (* Si algún motor acoplado -> desacoplar *)
                END_IF;
                (* Habilitar movimiento por consigna *)
                IF (AZM01_AT.DiagCode=16#D012 AND AZM02_AT.DiagCode=16#D012 AND
                        ((S_.Inicio AND MAIN.GEN_PLC1_PLC2.AutRemoto) OR
                         (NOT MAIN.GEN_PLC1_PLC2.AutRemoto AND MAIN.AZ_PLC1_PLC2.HabilitarEje))) OR
                        (AZM01_AT.DiagCode=16#D013 AND AZM02_AT.DiagCode=16#D013) THEN    (* Drive Ready *)
                                S_.Habilitar:=1;                                (* Habilitar variador *)
                                Paso:=2;                                        (* Cambiar a Paso 2 *)
                        END_IF;
                        (* Habilitar movimiento por jog *)
                        IF AZM01_AT.DiagCode=16#D012 AND AZM02_AT.DiagCode=16#D012
                        AND NOT MAIN.GEN_PLC1_PLC2.AutRemoto AND
                        (MAIN.AZ_PLC1_PLC2.JogBEje OR MAIN.AZ_PLC1_PLC2.JogFEje) AND NOT MAIN.AZ_PLC1_PLC2.HabilitarEje
                        AND NOT MAIN.AZ_PLC1_PLC2.HomeEje THEN
                                S_.Habilitar:=1;                                (* Habilitar variador *)
                                Paso:=30;                                       (* Cambiar a Paso 2 *)
                        END_IF;

        2:      (* Servos preparados - PARADOS *) (* Axis OP y Orden de mover *)
                IF AZM01_AT.DiagCode=16#D013 AND AZM02_AT.DiagCode=16#D013 AND
                ((S_.Mover AND MAIN.GEN_PLC1_PLC2.AutRemoto) OR NOT MAIN.GEN_PLC1_PLC2.AutRemoto)  THEN
                        SP_PosActual:=Vahle.PosAct;             (* Actualizar posición con encoder absoluto *)
                        S_.PosActual:=1;                        (* Habilitar cambio de posición actual Servos *)
                        Paso:=3;                                (* Cambiar a Paso 3 *)
                END_IF;

        3:      (* Actualizar posición con encoder Vahle *)
                IF AZM01.SetPos.Done THEN AuxDone0:=1;END_IF;   (* Cambio de posición Ok *)
                IF AZM02.SetPos.Done THEN AuxDone1:=1;END_IF;   (* Cambio de posición Ok *)
```

```
                IF AuxDone0 AND AuxDone1 THEN
                        AuxDone0:=0; AuxDone1:=0;                    (* Borrar flags auxiliares *)
                        S_.Acoplar:=1;                               (* Habilitar flag de acoplar *)
                        Paso:=4;                                     (* Cambiar a Paso 4 *)
                END_IF;

        4:      (* Acoplar servos - Maestro AZM01 de Esclavo AZM02 *)
                IF AZM02.Acople.InGear THEN                          (* Acople Ok *)
                        CalculoSPAut (SP:=SP_Aut, Pos:=MAIN.Azimutal.AZM01.Eje.NcToPlc.ActPos);  (* Cálculo SP por el camino más corto *)
                        SP_PosTarget:=CalculoSPAut.SP_Aux;                   (* Traspaso SP de posición destino *)
                        SP_Vel:=K_.VelTrabajo;                               (* Ajustar velocidad de trabajo *)
                        S_.MovPos:=1;                                        (* Habilitar movimiento en posición *)
                        PosCorr:=0.0;                                        (* Inicializar valor de corrección de posición *)
                        AlternarMovPos:=0;                           (* Inicializar bit alternancia de cambios de posición al vuelo *)
                        Vahle.FalloLectura:=0;                               (* Borrar Fallo lectura encoder Vahle *)
                        Vahle.TiempoFallo:=0;                        (* Inicializar temporizado para fallo lectura encoder Vahle *)
                        Paso:=20;                                                    (* Cambiar a Paso 20 *)
                END_IF;

        10:     (* Secuencia de finalizar movimiento *)
                IF NOT AZM01.Power.Status AND NOT AZM02.Power.Status THEN                (* Si pierden habilitación los drives
mandamos a inicio de secuencia *)
                        Paso:=0;
                END_IF;
                IF AZM01.ParoServo.Done THEN                 (* Parada ok *)
                        S_.Desacoplar:=1;                                    (* Orden de desacoplar *)
                        Paso:=11;                                            (* Cambiar a Paso 11 *)
                END_IF;

        11:     (* Desacoplar Servos *)
                IF NOT AZM01.Power.Status AND NOT AZM02.Power.Status THEN                (* Si pierden habilitación los drives
mandamos a inicio de secuencia *)
                        Paso:=0;
                END_IF;
                IF AZM02.Desacople.Done THEN            (* Desacople Ok *)
                        S_.Reset:=1;                                 (* Ejecutar reset NC y Drive después de desacoplar *)
                        S_.Habilitar:=0;                             (* Quitar habilitación variadores *)
                        Paso:=12;                                    (* Cambiar a Paso 12 *)
                END_IF;

        12:     (* Deshabilitar variador *)
                IF AZM01_AT.DiagCode=16#D012 AND AZM02_AT.DiagCode=16#D012 THEN          (* Drive Ready *)
                        Paso:=1;
                END_IF;
```

```
        20:     (* Servos preparados - POSICIONANDO *)
                IF NOT MAIN.AZ_PLC1_PLC2.HabilitarEje THEN      (* Si no tengo habilitación paro el movimiento sin llegar a destino *)
                        S_.Parar:=1;
                END_IF
                IF AZM01.MovPos.Done OR AZM01.MovPos1.Done OR AZM01.ParoServo.Done THEN
                        S_.Desacoplar:=1;                                   (* Orden para desacoplar *)
                        Paso:=21;                                                   (* Cambiar a Paso 21 *)
                END_IF;

        21:     (* Fin de movimiento - Desacoplar servos *)
                IF AZM02.Desacople.Done THEN            (* Desacople Ok *)
                        S_.Reset:=1;                                        (* Ejecutar reset NC y Drive después de desacoplar *)
                        S_.Habilitar:=0;                                    (* Deshabilitar variador *)
                        Paso:=1;                                            (* Cambiar a Paso 1 *)
                END_IF;

        30:     (* Servos preparados - PARADOS *) (* Axis OP y Orden de mover por jog *)
                IF NOT MAIN.AZ_PLC1_PLC2.JogBEje AND NOT MAIN.AZ_PLC1_PLC2.JogFEje THEN
                        S_.Habilitar:=0;                                    (* Deshabilitar variador *)
                        Paso:=1;                                            (* Cambiar a Paso 1 *)
                END_IF;
                IF AZM01_AT.DiagCode=16#D013 AND AZM02_AT.DiagCode=16#D013 THEN
                        Paso:=31;                                           (* Cambiar a Paso 31 *)
                END_IF;

        31:     (* Servos preparados - MOVIENDO EN JOG *)
                S_.JogB:=MAIN.AZ_PLC1_PLC2.JogBEje;
                S_.JogF:=MAIN.AZ_PLC1_PLC2.JogFEje;
                IF AZM01.Jog.Done THEN AuxDone0:=1;END_IF;      (* jog Ok *)
                IF AZM02.Jog.Done THEN AuxDone1:=1;END_IF;      (* jog Ok *)
                IF AuxDone0 AND AuxDone1 THEN
                        AuxDone0:=0; AuxDone1:=0;                           (* Borrar flags auxiliares *)
                        S_.Habilitar:=0;                                    (* Deshabilitar variador *)
                        Paso:=1;                                            (* Cambiar a Paso 1 *)
                        END_IF;
        END_CASE;
ELSE
        (* Manual *)
        IF D_.EMG THEN                                                      (* Si Emergencia se realiza parada segura *)
                S_.Parar:=1;
        END_IF;
        Paso:=0;
        SP_PosTarget:=SP_Man;                                               (* Actualizar consigna de posición en automático *)
END_IF;
```

```
(************************************ Potencia variadores ***************************************)
(* Habilitar potencia variador Azimutal 1 *)
AZM01.Power(                    Enable:=S_.Habilitar ,
                                                Enable_Positive:=S_.Habilitar ,
                                                Enable_Negative:=S_.Habilitar ,
                                                Override:=100 ,
                                                BufferMode:= ,
                                        Axis:=AZM01.Eje ,
            Status=> ,Busy=> ,Active=> ,Error=> ,ErrorID=> );

(* Habilitar potencia variador Azimutal 2 *)
AZM02.Power(                    Enable:=S_.Habilitar ,
                                                Enable_Positive:=S_.Habilitar ,
                                                Enable_Negative:=S_.Habilitar ,
                                                Override:=100 ,
                                                BufferMode:= ,
                                        Axis:=AZM02.Eje ,
            Status=> ,Busy=> ,Active=> ,Error=> ,ErrorID=> );

(***************************** Rearme general / Reset variadores ********************************)
(* Rearme Defectos *)
IF SRea THEN
            D_.Q:=0;
            D_.Acople:=0;
            D_.DefPos:=0;
            D_.EMG:=0;
    D_.Hab:=0;
END_IF;

(* Reset Servo Azimutal 1, conveniente ejecutar antes que reset de NC *)
AZM01.ResetServo(   NetId:= ,
                                                Execute:= SRea OR S_.Reset,
                                                Timeout:= ,
                                                Axis:= AZM01.Eje,
            Busy=> ,Error=> ,AdsErrId=> ,SercosErrId=> );

(* Reset Servo Azimutal 2, conveniente ejecutar antes que reset de NC *)
AZM02.ResetServo(   NetId:= ,
                                                Execute:= SRea OR S_.Reset,
                                                Timeout:= ,
                                                Axis:= AZM02.Eje,
            Busy=> ,Error=> ,AdsErrId=> ,SercosErrId=> );
```

```
(* Reset NC Azimutal 1, conveniente ejecutar después de reset de Servo *)
AZM01.ResetNC(                  Execute:=SRea OR S_.Reset ,
                                                Axis:=AZM01.Eje,
        Done=> ,Busy=> ,Error=> ,ErrorID=> );

(* Reset NC Azimutal 2, conveniente ejecutar después de reset de Servo *)
IF AZM02.ModoAcople=0 THEN
        AZM02.ResetNC(                  Execute:=SRea OR S_.Reset ,
                                                Axis:=AZM02.Eje ,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(************************************* Defectos ************************************************)
(* Defectos HW *)
IF NOT Q THEN D_.Q:=1; END_IF;                          (* Defecto Térmico *)
IF WDef THEN D_.EMG:=1; END_IF;                         (* Defecto Parada de emergencia *)

(* Defectos SW *)
(*
IF (AZM01.ModoAcople<>1 OR AZM02.ModoAcople <>3) THEN
        D_.Acople:=1; ELSE D_.Acople:=0;                (* Defecto Acople *)
END_IF;
*)
(*
IF AZM02.ModoAcople=0 AND  ABS(AZM01.Eje.NcToPlc.ActPos - AZM02.Eje.NcToPlc.ActPos)>K_.DefPos THEN
        D_.DefPos:=1; ELSE D_.DefPos:=0;                (* Defecto de posición entre servos sólo cuando no están acoplados *)
END_IF;
*)
IF (Paso=20) AND (NOT AZM01.Power.Status OR NOT AZM02.Power.Status) THEN
        D_.Hab:=1;                                      (* Defecto habilitación en motores *)
END_IF;

(* Reunión Defectos *)
Def:=D_.Q OR D_.Acople OR D_.DefPos OR D_.Hab OR D_.EMG;

(* Fallo lectura Encoder Vahle *)
(* Estar en velocidad de trabajo y (lectura cogelada o incremento de lectura excesivo *)
IF (Paso=20)  AND ((ABS(AZM01.Eje.NcToPlc.ActVelo)>(K_.VelTrabajo*0.95))) AND ((Vahle.PosAct=Vahle.PosAnt) OR (ABS(Vahle.PosAct - Vahle.PosAnt)
> 50)) THEN
        Vahle.TiempoFallo:=Vahle.TiempoFallo + 1;
        IF Vahle.TiempoFallo > 5 THEN                   (* Si durante 10 ciclos misma lectua --> fallo *)
                Vahle.FalloLectura:=1;
        END_IF;
END_IF;
IF (Vahle.PosAct <> Vahle.PosAnt) AND (ABS(Vahle.PosAct - Vahle.PosAnt) < 50) THEN
        Vahle.TiempoFallo:=0;
```

```
		Vahle.PosAnt:=Vahle.PosAct;
		Vahle.FalloLectura:=0;
END_IF;
(************************************ Activaciones Pasos ****************************************)
(* Habilitar Control Corrección de Posición *)
(*
HabControlPos:=(Paso = 20);			(* Paso 20 =  Servo preparado posicionando *)
*)
(* Cálculo del error de posición entre encoder Vahle y Servos *)
IF (Paso = 20) THEN
		IF NOT Vahle.FalloLectura THEN (* Filtro para evitar lecturas incorrectas de APOS y origine errores falsos muy grandes*)
			PosError:=PosAct_0_360 - Vahle.PosAct - PosCorr;
		END_IF;
END_IF;
(* Corrección de posición con encoder externo *)
IF HabControlPos AND (ABS(PosError) > 20) AND (ABS(AZM01.Eje.NcToPlc.ActVelo) > (K_.VelTrabajo*0.95))  THEN
			cont_1:=cont_1+1;
			PosCorr:=PosCorr + PosError;		(* Actualizar Posición a corregir *)
			SP_PosTarget:=SP_PosTarget + PosError;	(* Actualizar Posición de destino Servos *)
			IF AlternarMovPos THEN			(* Generar alternancia para cambiar posición de destino al vuelo *)
				S_.MovPos:=1;				(* Habilitar flag de mover en posición *)
				AlternarMovPos:=0;
			ELSE
				S_.MovPosNew:=1;			(* Habilitar flag de mover en posición *)
				AlternarMovPos:=1;
			END_IF;
END_IF;

(************************************ Paro motores ****************************************)
(* Orden de paro servomotor Azimutal 1 *)
AZM01.ParoServo(	Execute:=S_.Parar,
						Deceleration:= ,
						Jerk:= ,
						Options:= ,
						Axis:=AZM01.Eje ,
		Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );

(* Orden de paro servomotor Azimutal 2 (si está desacoplado) *)
IF AZM02.ModoAcople=0 THEN
		AZM02.ParoServo(	Execute:=S_.Parar,
							Deceleration:= ,
							Jerk:= ,
							Options:= ,
							Axis:=AZM02.Eje ,
			Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(****************************************** Mover Jog  ***********************************************)
(* Mover en manual en sentido positivo/negativo servo Azimutal 1 *)
AZM01.Jog(                              JogForward:=S_.JogF ,
                                        JogBackwards:=S_.JogB ,
                                        Mode:=2 ,           (* MC_JOGMODE_CONTINOUS - usa parámetros del PLC *)
                                        Position:= ,
                                        Velocity:=SP_Vel ,
                                        Acceleration:= ,
                                        Deceleration:= ,
                                        Jerk:= ,
                                        Axis:=AZM01.Eje ,
        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );

(* Mover en manual en sentido positivo/negativo servo Azimutal 2 (si está desacoplado) *)
IF AZM02.ModoAcople=0 THEN
        AZM02.Jog(                      JogForward:=S_.JogF ,
                                                JogBackwards:=S_.JogB ,
                                                Mode:=2 ,           (* MC_JOGMODE_CONTINOUS - usa parámetros del PLC *)
                                                Position:= ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                Axis:=AZM02.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(************************************ Mover en Velocidad  ****************************************)
(* Mover en velocidad servomotor Azimutal 1 *)
AZM01.MovVel(                   Execute:=S_.MovVel ,
                                        Velocity:=SP_Vel ,
                                        Acceleration:= ,
                                        Deceleration:= ,
                                        Jerk:= ,
                                        Direction:= ,
                                        BufferMode:= ,
                                        Options:= ,
                                        Axis:=AZM01.Eje ,
        InVelocity=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
```

```
(* Mover en velocidad servomotor Azimutal 2 (si está desacoplado) *)
IF AZM02.MododAcople=0 THEN
        AZM02.MovVel(           Execute:=S_.MovVel ,
                                        Velocity:=SP_Vel ,
                                        Acceleration:= ,
                                        Deceleration:= ,
                                        Jerk:= ,
                                        Direction:= ,
                                                BufferMode:= ,
                                        Options:= ,
                                        Axis:=AZM02.Eje ,
                InVelocity=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
(************************************ Mover en Posición ******************************************)
(* Mover en posición servomotor Azimutal 1 *)
AZM01.MovPos(           Execute:= S_.MovPos,
                                Position:=SP_PosTarget ,
                                Velocity:=SP_Vel ,
                                Acceleration:= ,
                                Deceleration:= ,
                                Jerk:= ,
                                BufferMode:= ,
                                Options:= ,
                                Axis:=AZM01.Eje ,
        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );

(* Mover en posición servomotor Azimutal 2 (si está desacoplado) *)
IF AZM02.MododAcople=0 THEN
        AZM02.MovPos(           Execute:= S_.MovPos,
                                        Position:=SP_PosTarget ,
                                        Velocity:=SP_Vel ,
                                        Acceleration:= ,
                                        Deceleration:= ,
                                        Jerk:= ,
                                        BufferMode:= ,
                                        Options:= ,
                                        Axis:=AZM02.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(******************************** Mover en Posición - Nuevo destino ******************************)
(* Mover en posición servomotor Azimutal 1 *)
AZM01.MovPos1(                  Execute:= S_.MovPosNew,
                                                Position:=SP_PosTarget ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Axis:=AZM01.Eje ,
        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );

(* Mover en posición servomotor Azimutal 2 (si está desacoplado) *)
IF AZM02.ModoAcople=0 THEN
        AZM02.MovPos1(          Execute:= S_.MovPosNew,
                                                Position:=SP_PosTarget ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Axis:=AZM02.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
(******************************** Cambiar posición actual ****************************************)
(* Es posible dar una posición "al vuelo" *)
(* Cambiar la posición actual del servomotor Azimutal 1 *)
AZM01.SetPos(   Execute:=S_.PosActual ,
                                Position:=SP_PosActual ,
                                Mode:= ,
                                Options:= ,
                                Axis:=AZM01.Eje ,
        Done=> ,Busy=> ,Error=> ,ErrorID=> );

(* Cambiar la posición actual del servomotor Azimutal 2 (si está desacoplado) *)
IF AZM02.ModoAcople=0 THEN
        AZM02.SetPos(   Execute:=S_.PosActual ,
                                                Position:=SP_PosActual ,
                                                Mode:= ,
                                                Options:= ,
                                                Axis:=AZM02.Eje ,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(******************************** Acoplar/Desacoplar Ejes ****************************************)
(* Acople de servomotor Azimutal 2 como esclavo a servomotor Azimutal 1 como maestro *)
(* Acoplar ejes con un ratio determinado *)
AZM02.Acople(                   Enable:=S_.Acoplar ,
                                                GearRatio:=AZM02.RatioAcople ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Master:=AZM01.Eje ,
                                                Slave:=AZM02.Eje ,
        InGear=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
(* Desacoplar ejes *)
AZM02.Desacople(    Execute:=S_.Desacoplar ,
                                                Options:= ,
                                                Slave:=AZM02.Eje ,
        Done=> ,Busy=> ,Error=> ,ErrorID=> );

(********************************** Flags/Habilitaciones ****************************************)
(* Habilitar contactor alimentación 380V variadores desde web  *)
K:=S_.HabilitarK;
(* Borrado de flags, habilitaciones, pulsadores *)
S_.Acoplar:=0;                  (* Flag de habilitación Acople *)
S_.Desacoplar:=0;       (* Flag de habilitación Desacoplar *)
S_.PosActual:=0;        (* Flag de habilitación Cambiar la posición actual de Servomotores *)
S_.MovPos:=0;                   (* Flag de habilitación de movimiento *)
S_.MovPosNew:=0;    (* Flag de habilitación para mover en posición con nuevo destino *)
S_.Inicio:=0;                   (* Flag de habilitación inicio de movimiento en automático *)
S_.Fin:=0;                      (* Flag de habilitación fin de movimiento en automático  *)
S_.Parar:=0;                    (* Flag de parar movimiento *)
S_.Mover:=0;                    (* Flag de mover servomotores *)
S_.Reset:=0;                    (* Flag de Reset *)

(************************************ Visualizar en Web ****************************************)
(* Modo de acople *)
AZM01.ModoAcople:=DWORD_TO_INT(AZM01.Eje.NcToPlc.CoupleState);
AZM02.ModoAcople:=DWORD_TO_INT(AZM02.Eje.NcToPlc.CoupleState);
(* Código de diagnóstico *)
AZM01.DiagCode:=DWORD_TO_INT(AZM01_AT.DiagCode);
AZM02.DiagCode:=DWORD_TO_INT(AZM02_AT.DiagCode);
(* Valores variables en unidades reales *)
AZM01.Intensidad_A(Valor_0_1000:=AZM01_Intensidad.VF , FE:=6.3 , VR=> );
AZM01.Par_Nm(Valor_0_1000:=AZM01_PAR.VF , FE:=8.2 , VR=> );
AZM02.Intensidad_A(Valor_0_1000:=AZM02_Intensidad.VF , FE:=6.3 , VR=> );
AZM02.Par_Nm(Valor_0_1000:=AZM02_PAR.VF , FE:=8.2 , VR=> );
```

8.2.3 FB10_CR. Llamada a FB de gestión de movimiento Compuerta

```
FUNCTION_BLOCK FB10_CR
VAR
        Man: BOOL; (* Modo Manual *)
        Aut: BOOL; (* Modo Automático *)
        Def: BOOL; (* Reunión Defectos *)
        AuxDone0: BOOL; (* Auxiliar operación ejecutada con éxito *)
        AuxDone1: BOOL; (* Auxiliar operación ejecutada con éxito *)
        AuxDone2: BOOL; (* Auxiliar operación ejecutada con éxito *)
        Abierto: BOOL; (* Compuerta en posición de abierto *)
        Cerrado: BOOL; (* Compuerta en posición de cerrado *)
        CBAbrir: BOOL; (* Orden de abrir compuertas *)
        CBCerrar: BOOL; (* Orden de cerrar compuertas *)
        HabControlPar: BOOL; (* Habilitar control de Par *)
        TiempoEspera: BOOL; (* En tiempo de espera *)
        ZonaA: BOOL; (* Compuerta en primera mitad de apertura *)
        ZonaB: BOOL; (* Compuerta en primera mitad de cierre *)
        Paso: INT; (* Número de paso en modo Automático *)
        cont_1: INT; (* Auxiliar contador *)
        cont_2: INT; (* Auxiliar contador *)
        VCTiempoEspera: INT; (* Contador ciclos sin aplicar control de Par *)
        VCCorregirDestensar: INT; (* Contador ciclos Corregir Destensar *)
        VCCorregirTensar: INT; (* Contador ciclos Corregir Tensar *)
        SP_Vel: LREAL; (* Setpoint de velocidad *)
        SP_PosTarget: LREAL; (* Setpoint de posición absoluta objetivo *)
        SP_RelTarget: LREAL; (* Setpoint de posición relativa objetivo *)
        SP_Aut: LREAL; (* Setpoint de posición en Automático *)
        SP_Man: LREAL; (* Setpoint de posición en Manual *)
        SP_PosActual: LREAL; (* Setpoint de posición actual *)
        SP_PosActual1: LREAL; (* Setpoint de posición actual *)
        CRM01: Servo; (* Control Servomotor Cierre Compuerta - CRM01 *)
        CRM02: Servo; (* Control Servomotor Apertura Compuerta - CRM02 *)
        D_: Servo_D; (* Defectos Servomotor *)
        S_: Servo_S; (* Pulsadores Servomotor *)
        HabControlParF: R_TRIG; (* Habilitar control de Par Flanco Positivo *)
        CRM01_PAR: FB22_FiltroAnal; (* Motor cierre Par filtrado *)
        CRM02_PAR: FB22_FiltroAnal; (* Motor apertura Par filtrado *)
        CRM01_Intensidad: FB22_FiltroAnal; (* Motor cierre Intensidad filtrada *)
        CRM02_Intensidad: FB22_FiltroAnal; (* Motor apertura Intensidad filtrada *)
        CRM02_Pos: FB24_CompuertaCG; (* Función de conversión cº Tambor Encoder <-> cº Apertura Compuerta *)
END_VAR
VAR_INPUT
        Q: BOOL; (* Térmico alimentación variadores *)
        WDef: BOOL; (* Defecto externo *)
        SRea: BOOL; (* Pulsador de rearme *)
```

```
        WAut: BOOL; (* Condiciones marcha automático *)
END_VAR
VAR_OUTPUT
        K: BOOL; (* Marcha contactor alimentación variadores *)
END_VAR
VAR PERSISTENT
        UltimoMov: BOOL; (* Último movimiento registrado: '1' Abrir - '0' Cerrar *)
        PosAbierto: BOOL; (* Posición de abierto variable dependiendo de las correcciones por Superimposed *)
        PosCerrado: BOOL; (* Posición de cerrado variable dependiendo de las correcciones por Superimposed *)
        DestinoTeoricoAbrir: DINT; (* Destino teórico en maniobra de apertura. Sólo se calcula en posición de cerrado *)
        DestinoTeoricoCerrar: DINT; (* Destino teórico en maniobra de cierre. Sólo se calcula en posición de abierto *)
        AjustePosicionPar: LREAL; (* Ajuste posición dependiendo de las correcciones por Superimposed *)
        K_: Servo_K; (* Constantes Servomotor *)
END_VAR
```

```
(* FB control servomotores movimiento Compuerta Rendijas *)
(****************************************** Previo ******************************************************)
(* Constantes *)
K_.RecorridoLineal:=10600;                          (* [cº] *)
K_.PosAbierto:=300000;                              (*280928 - 36000;*)
K_.PosCerrado:=-300000;                             (*-280928 + 36000;*)
K_.RecorridoAjusteAbierta:=500;                     (* Recorrido ajuste abierta en cº de vueltas de tambor *)
K_.RecorridoAjusteCerrada:=500;                     (* Recorrido ajuste cerrada en cº de vueltas de tambor *)
K_.VelTrabajo:=3000;                                (* Velocidad de trabajo *)
K_.VelAjuste:=500;                                  (* Velocidad de ajuste para tensar/destensar sirgas cº/s *)
K_.ParMinimo:=100;                                  (* Par mínimo aplicar corrección Superimposed %o *)
CRM01.RatioAcople:=1;
K_.CRM01_ParVacio:=80;                                          (* Par en vacío de CRM01 *)
(* 27-Agosto K_.CRM02_ParVacio:=50 -> K_.CRM02_ParVacio:=70*)
K_.CRM02_ParVacio:=70;                                          (* Par en vacío de CRM02 *)

(* Asignar posición destino desde pantalla *)
IF S_.Abrir THEN K_.PosDestino:=276200; END_IF;
IF S_.Cerrar THEN K_.PosDestino:=-277488; END_IF;
(* Asignar posición destino desde pantalla: Destinos Teóricos *)
(*
IF S_.Abrir THEN K_.PosDestino:=DestinoTeoricoAbrir; END_IF;
IF S_.Cerrar THEN K_.PosDestino:=DestinoTeoricoCerrar; END_IF;
*)
(* Conversión cº Tambor Encoder <-> cº Apertura Compuerta *)
CRM02_Pos(          E_cg_Compuerta:=K_.PosDestino_cg_Compuerta ,
                    E_cg_Encoder:=CRM02.Eje.NcToPlc.ActPos ,
                    S_cg_Compuerta=> , S_cg_Encoder=> );

(* Filtro señales analógicas *)
CRM01_PAR(AnalIn:=ABS(CRM01_AT.TorqueFeedbackValue));
CRM02_PAR(AnalIn:=ABS(CRM02_AT.TorqueFeedbackValue));
```

```
CRM01_Intensidad(AnalIn:=ABS(CRM01_AT.ActualAbsoluteCurrent));
CRM02_Intensidad(AnalIn:=ABS(CRM02_AT.ActualAbsoluteCurrent));

(*Movimiento Abrir/Cerrar en función de la velocidad*)
CBAbrir := (CRM02.Eje.NcToPlc.ActVelo > 10);
CBCerrar := (CRM02.Eje.NcToPlc.ActVelo < -10);
IF CBAbrir THEN UltimoMov:=1; END_IF;                (* Memoria último movimiento registrado: Abrir *)
IF CBCerrar THEN UltimoMov:=0; END_IF;               (* Memoria último movimiento registrado: Cerrar *)

(* Estados de Posición Cerrado y Abierto *)
Cerrado:=(CRM02.Eje.NcToPlc.ActPos < (K_.PosCerrado + 1));
Abierto:=(CRM02.Eje.NcToPlc.ActPos > (K_.PosAbierto - 1));

(* Función para Tensar sirga compuertas bajo control de Par *)
(* Dos compuertas 20 y 30º *)
ZonaA := (CRM02.Eje.NcToPlc.ActPos < -201339);      (*Compuerta en primera mitad de apertura según geometría*)
ZonaB := (CRM02.Eje.NcToPlc.ActPos > -149801);      (*Compuerta en primera mitad de cierre según geometría*)

(* Compuerta inferior 50 y 60º *)
(*
ZonaA := (CRM02.Eje.NcToPlc.ActPos < -46000);       (*Compuerta en primera mitad de apertura según geometría*)
ZonaB := (CRM02.Eje.NcToPlc.ActPos > 5000);                 (*Compuerta en primera mitad de cierre según geometría*)
*)

(* Cálculo de los destinos teóricos en las posiciones de abrir/cerrar *)
IF PosAbierto THEN
        DestinoTeoricoCerrar:=LREAL_TO_DINT(CRM02.Eje.NcToPlc.ActPos) - K_.RecorridoLineal;
        AjustePosicionPar:=0;
END_IF;
IF PosCerrado THEN
        DestinoTeoricoAbrir:=LREAL_TO_DINT(CRM02.Eje.NcToPlc.ActPos) + K_.RecorridoLineal;
        AjustePosicionPar:=0;
END_IF;

(****************************************** Modos ******************************************************)
(* Modos Manual - Automático *)
Man:=NOT Aut;
Aut:=S_.Man_Aut;

(* Gestión Modo Automático - Manual *)
IF Aut THEN
        S_.HabilitarK:=1;                           (* Habilitar contactor de potencia *)
        IF S_.Fin OR (Def AND (Paso>12)) THEN       (* Pulsando Fin o Defecto en cualquier momento forzamos parada segura *)
                S_.Parar:=1;                        (* Habilitar flag de parada *)
                Paso:=10;                           (* Cambiar a Paso 10 *)
END_IF;
```

```
CASE Paso OF
        0:      (* Inicio - Todo deshabilitado *)
                IF CRM01.ModoAcople<>0 OR CRM02.ModoAcople<>0 THEN
                        S_.Desacoplar:=1;                       (* Si algún motor acoplado -> desacoplar *)
                END_IF;
                IF NOT Def THEN
                        S_.Reset:=1;                            (* Habilitar reset de NC *)
                                Paso:=1;                        (* Cambiar a Paso 1 *)
                END_IF;

        1:      (* Esperar Drive Ready *)
                IF CRM01.ModoAcople<>0 OR CRM02.ModoAcople<>0 THEN
                        S_.Desacoplar:=1;                       (* Si algún motor acoplado -> desacoplar *)
                END_IF;
                IF ((CRM01_AT.DiagCode=16#D012 AND CRM02_AT.DiagCode=16#D012 AND S_.Inicio) OR
                (CRM01_AT.DiagCode=16#D013 AND CRM02_AT.DiagCode=16#D013)) THEN         (* Drive Ready *)
                        S_.Habilitar:=1;                        (* Habilitar variador *)
                        Paso:=2;                                (* Cambiar a Paso 2 *)
                END_IF;

        2:      (* Servos preparados *)
                IF CRM01_AT.DiagCode=16#D013 AND CRM02_AT.DiagCode=16#D013 AND S_.Mover THEN    (* Axis OP y Orden de mover *)
                        Paso:=20;                               (* Cambiar a Paso 20 *)
                END_IF;

        10:     (* Secuencia de finalizar movimiento - Orden de parar *)
                IF NOT CRM01.Power.Status AND NOT CRM02.Power.Status THEN        (* Si pierden habilitación los drives mandamos a
inicio de secuencia *)
                        Paso:=0;
                END_IF;
                IF CRM01.ParoServo.Done OR  CRM02.ParoServo.Done THEN    (* Parada ok *)
                        S_.Desacoplar:=1;                       (* Habilitar flag de desacoplar *)
                        Paso:=11;                               (* Cambiar a Paso 11 *)
                END_IF;

        11:     (* Desacoplar Servos *)
                IF NOT CRM01.Power.Status AND NOT CRM02.Power.Status THEN        (* Si pierden habilitación los drives mandamos a
inicio de secuencia *)
                        Paso:=0;
                END_IF;
                IF CRM01.Desacople.Done OR CRM02.Desacople.Done THEN     (* Desacople Ok *)
                        S_.Reset:=1;                            (* Ejecutar reset NC y Drive después de desacoplar *)
                        S_.Habilitar:=0;                        (* Deshabilitar variador *)
                        Paso:=12;                               (* Cambiar a Paso 12 *)
                END_IF;
```

```
12:     (* Deshabilitar variador *)
        IF CRM01_AT.DiagCode=16#D012 AND CRM02_AT.DiagCode=16#D012 THEN          (* Drive Ready *)
                Paso:=1;
        END_IF;

20:     (* Orden de mover *)
        (* Compuerta en posición Abierto *)
        IF Abierto THEN
                S_.MovPos1:=1;                          (* Habilitar flag de mover en posición *)
                SP_PosTarget:=CRM01.Eje.NcToPlc.ActPos - K_.RecorridoAjusteAbierta;    (* Valor de posición para tensar la sirga *)
                SP_Vel:=K_.VelAjuste;                                                  (* Cargar velocidad de ajuste tensar *)
                Paso:=21;                                                              (* Cambiar a Paso 21 *)
        END_IF;

        (* Compuerta en posición Cerrado *)
        IF Cerrado THEN
                S_.MovPos:=1;                                                          (* Habilitar flag de mover en posición *)
                SP_PosTarget:=CRM02.Eje.NcToPlc.ActPos + K_.RecorridoAjusteCerrada;    (* Valor de posición para tensar la sirga *)
                SP_Vel:=K_.VelAjuste;                                                  (* Cargar velocidad de ajuste tensar *)
                Paso:=22;                                                                      (* Cambiar a Paso 22 *)
        END_IF;

        (* Compuerta no está ni en posición de Cerrado ni Abierto *)
        IF NOT Cerrado AND NOT Abierto THEN
                SP_PosActual:=CRM02.Eje.NcToPlc.ActPos;(* Actualizar posición CRM01 con valor de CRM02 de encoder absoluto *)
                S_.PosActual:=1;                        (* Habilitar flag de cambio de posición absoluta *)
                Paso:=23;                               (* Cambiar a Paso 23 *)
        END_IF;

21:     (* Ajustar tensión en sirga en CRM01 *)
        IF CRM01.MovPos.Done THEN                       (* Alcanza posición Ok *)
                SP_PosActual:=CRM02.Eje.NcToPlc.ActPos;(* Actualizar posición CRM01 con valor de CRM02 de encoder absoluto *)
                S_.PosActual:=1;                        (* Habilitar flag de cambio de posición absoluta *)
                Paso:=23;                               (* Cambiar a Paso 23 *)
        END_IF;

22:     (* Ajustar tensión en sirga en CRM02 *)
        IF CRM02.MovPos.Done THEN                       (* Alcanza posición Ok *)
                SP_PosActual:=CRM02.Eje.NcToPlc.ActPos;(* Actualizar posición CRM01 con valor de CRM02 de encoder absoluto *)
                S_.PosActual:=1;                        (* Habilitar flag de cambio de posición absoluta *)
                Paso:=23;                               (* Cambiar a Paso 23 *)
        END_IF;
```

```
23:     (* Actualizar posición con encoder absoluto de CRM02 *)
        IF CRM01.SetPos.Done THEN                                   (* Cambio de posición Ok *)
                S_.Acoplar:=1;                              (* Habilitar flag de acoplar CRM01 como esclavo de CRM02 *)
                Paso:=24;                                                                   (* Cambiar a Paso 24 *)
        END_IF;

24:     (* Acoplar servos: CRM01 esclavo de CRM02 maestro *)
        IF CRM01.Acople.InGear THEN                                 (* Acople Ok *)
                SP_PosTarget:=K_.PosDestino;                        (* Transferir posición destino *)
                SP_RelTarget:=SP_PosTarget - CRM02.Eje.NcToPlc.ActPos;      (* Transferir posición destino en relativo *)
                SP_Vel:=K_.VelTrabajo;                              (* Ajustar velocidad de trabajo *)
                S_.MovPos:=1;                                       (* Habilitar flag de mover en posición absoluta *)
(*              S_.MovRel:=1;                                       (* Habilitar flag de mover en posición relativa *)*)
                Paso:=25;                                                                   (* Cambiar a Paso 25 *)
        END_IF;

25:     (* Servo preparado - MOVIENDO *)
        IF CRM02.MovPos.Done OR CRM02.ParoServo.Done OR CRM02.MovRel.Done THEN    (* Alcanza posición Ok o Parada Ok *)
                S_.Desacoplar:=1;                                                 (* Habilitar flag de desaclople servos *)
                Paso:=26;                                                         (* Cambiar a Paso 26 *)
        END_IF;

26:     (* Fin de movimiento - Desacoplar servo *)
        IF CRM01.Desacople.Done THEN                        (* Desacople Ok *)
                S_.Reset:=1;                                (* Reset NC *)
                Paso:=27;                                   (* Cambiar a Paso 27 *)
        END_IF;

27:     (* Desacople y Reset *)
        (* Compuerta en posición Abierto *)
        IF Abierto THEN
                S_.MovPos1:=1;                                                    (* Habilitar flag de mover en posición *)
                SP_PosTarget:=CRM01.Eje.NcToPlc.ActPos + K_.RecorridoAjusteAbierta;    (* Posición para destensar la sirga *)
                Paso:=28;                                                         (* Cambiar a Paso 28 *)
        END_IF;

        (* Compuerta en posición Cerrado *)
        IF Cerrado THEN
                S_.MovPos:=1;                                                     (* Habilitar flag de mover en posición *)
                SP_PosTarget:=CRM02.Eje.NcToPlc.ActPos - K_.RecorridoAjusteCerrada;    (* Posición para destensar la sirga *)
                Paso:=29;                                                                   (* Cambiar a Paso 29 *)
        END_IF;

        (* Compuerta no está ni en posición de Cerrado ni Abierto *)
        IF NOT Cerrado AND NOT Abierto THEN
                S_.Habilitar:=0;
```

```
                    Paso:=1;                                    (* Cambiar a Paso 2 *)
                END_IF;

        28:     (* Ajustar Destensión en Sirga de CRM01 *)
                IF CRM01.MovPos.Done THEN
                    S_.Habilitar:=0;
                    Paso:=1;                                    (* Cambiar a Paso 2 *)
                END_IF;

        29:     (* Ajustar Destensión en Sirga de CRM02 *)
                IF CRM02.MovPos.Done THEN
                    S_.Habilitar:=0;
                    Paso:=1;                                    (* Cambiar a Paso 2 *)
                END_IF;

END_CASE;
ELSE
        (* Manual *)
        IF D_.EMG THEN                                          (* Si Emergencia se realiza parada segura *)
            S_.Parar:=1;
        END_IF;
        Paso:=0;
        SP_PosTarget:=SP_Man;                                   (* Actualizar consigna de posición en automático *)
END_IF;
(* Control límites consigna de abierto/cerrado *)
IF SP_PosTarget >= 290000 THEN SP_PosTarget:=290000; END_IF;
IF SP_PosTarget <= -277488 THEN SP_PosTarget:=-277488; END_IF;
(************************************ Potencia variadores ***************************************)
(* Habilitar potencia variador Compuerta cierre *)
CRM01.Power(            Enable:=S_.Habilitar ,
                                Enable_Positive:=S_.Habilitar ,
                                Enable_Negative:=S_.Habilitar ,
                                Override:=100 ,
                                BufferMode:= ,
                            Axis:=CRM01.Eje ,
        Status=> ,Busy=> ,Active=> ,Error=> ,ErrorID=> );

(* Habilitar potencia variador Compuerta apertura *)
CRM02.Power(            Enable:=S_.Habilitar ,
                                Enable_Positive:=S_.Habilitar ,
                                Enable_Negative:=S_.Habilitar ,
                                Override:=100 ,
                                BufferMode:= ,
                            Axis:=CRM02.Eje ,
        Status=> ,Busy=> ,Active=> ,Error=> ,ErrorID=> );
```

```
(*************************** Rearme general / Reset variadores ********************************)
(* Rearme Defectos *)
IF SRea THEN
        D_.Q:=0;
        D_.Vel:=0;
        D_.Hab:=0;
        D_.EMG:=0;
END_IF;

(* Reset Servo Compuerta cierre, conveniente ejecutar antes que reset de NC *)
CRM01.ResetServo(       NetId:= ,
                                                Execute:= SRea OR S_.Reset,
                                                Timeout:= ,
                                                Axis:= CRM01.Eje,
        Busy=> ,Error=> ,AdsErrId=> ,SercosErrId=> );

(* Reset Servo Compuerta apertura, conveniente ejecutar antes que reset de NC *)
CRM02.ResetServo(       NetId:= ,
                                                Execute:= SRea OR S_.Reset,
                                                Timeout:= ,
                                                Axis:= CRM02.Eje,
        Busy=> ,Error=> ,AdsErrId=> ,SercosErrId=> );

(* Reset NC Compuerta cierre, conveniente ejecutar después de reset de Servo *)
IF CRM01.ModoAcople=0 OR CRM01.ModoAcople=1 THEN
        CRM01.ResetNC(          Execute:=SRea OR S_.Reset ,
                                Axis:=CRM01.Eje,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(* Reset NC Compuerta apertura, conveniente ejecutar después de reset de Servo *)
IF CRM02.ModoAcople=0 OR CRM02.ModoAcople=1 THEN
        CRM02.ResetNC(          Execute:=SRea OR S_.Reset ,
                                Axis:=CRM02.Eje,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(************************************* Defectos ************************************************)
(* Defectos HW *)
IF NOT Q THEN D_.Q:=1; END_IF;                  (* Defecto Térmico *)
IF WDef THEN D_.EMG:=1; END_IF;                 (* Defecto Parada de emergencia *)

(* Defectos SW *)
(*IF (CRM01.ModoAcople<>1 OR CRM02.ModoAcople <>3) THEN
        D_.Acople:=1; ELSE D_.Acople:=0;        (* Defecto Acople *)
END_IF;
```

```
*)
(*
IF (Paso=25) AND (ABS(ABS(CRM02.Eje.NcToPlc.ActVelo) - ABS(CRM01.Eje.NcToPlc.ActVelo)) > (SP_Vel*0.5))  THEN
        D_.Vel:=1;                                              (* Defecto control de velocidad *)
END_IF;
*)
IF (Paso=25) AND (NOT CRM01.Power.Status OR NOT CRM02.Power.Status) THEN
        D_.Hab:=1;                                              (* Defecto habilitación en motores *)
END_IF;

(* Reunión Defectos *)
Def:=D_.Q OR D_.Vel OR D_.Hab OR D_.EMG;

(************************************ Activaciones Pasos ****************************************)
(* Habilitación funcionamiento con Control de Par *)
HabControlPar := (Paso = 25);
HabControlParF(CLK:=HabControlPar , Q=> );
(* Gestión par tiempo de espera *)
IF CRM01.SuperImp.Done OR HabControlParF.Q THEN
        TiempoEspera:=1;
        VCTiempoEspera :=0;
END_IF;
IF TiempoEspera THEN
        VCTiempoEspera := VCTiempoEspera + 1;
END_IF;
IF (VCTiempoEspera>=50) THEN
        TiempoEspera:=0;
END_IF;

(*Gestón par mínimo*)
(*Velocidad real dentro de una ventana de trababjo para evitar las rampas de aceleración y decelaración*)
IF HabControlPar AND NOT TiempoEspera AND NOT CRM01.SuperImp.Busy AND (ABS((ABS(CRM02.Eje.NcToPlc.ActVelo)) - SP_Vel) < 50) THEN

        (*Apertura - Zona A - Compuerta 0º-20º de apertura->Corrección decremento velocidad en CRM01 - TENSAR*)
        IF CBAbrir AND (CRM01_PAR.VF > (K_.CRM01_ParVacio - 20)) AND (CRM02_PAR.VF < 470) AND ZonaA THEN
                K_.CorreccionSI := -150;
                K_.LongitudSI := 3000;
                VCCorregirTensar := VCCorregirTensar + 1;
                S_.MovSuper := 1;
                (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
                SP_PosTarget:=DestinoTeoricoAbrir + AjustePosicionPar;          (* Nueva posición de destino motivada por la corrección por
Superimposed *)
                S_.MovPos:=1;*)
        END_IF;
```

```
(*Apertura - Zona A - Compuerta 0º-20º de apertura->Corrección incremento velocidad en CRM01 - DESTENSAR*)
(*26-Agosto: (K_.CRM02_ParVacio + 450) -> (K_.CRM02_ParVacio + 500)*)
IF CBAbrir AND (CRM01_PAR.VF < (K_.CRM01_ParVacio - 35)) AND (CRM02_PAR.VF > (K_.CRM02_ParVacio + 480)) AND ZonaA THEN
        K_.CorreccionSI := 150;
        K_.LongitudSI := 3000;
        VCCorregirDestensar := VCCorregirDestensar + 1;
        S_.MovSuper := 1;
        (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
        SP_PosTarget:=DestinoTeoricoAbrir + AjustePosicionPar;        (* Nueva posición de destino motivada por la corrección por
Superimposed *)
        S_.MovPos:=1;*)
END_IF;
(*Apertura - No Zona A - Compuerta 20º-110º de apertura->Corrección decremento velocidad en CRM01 - TENSAR*)
(*27-Agosto:  (K_.CRM01_ParVacio - 30) ->  (K_.CRM01_ParVacio - 40)*)
(*27-Agosto: (K_.CRM02_ParVacio + 30) -> (K_.CRM02_ParVacio + 50)*)
IF CBAbrir AND (CRM01_PAR.VF > (K_.CRM01_ParVacio - 40)) AND (CRM02_PAR.VF < (K_.CRM02_ParVacio + 30)) AND NOT ZonaA THEN
        K_.CorreccionSI := -150;
        K_.LongitudSI := 3000;
        VCCorregirTensar := VCCorregirTensar + 1;
        S_.MovSuper := 1;
        (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
        SP_PosTarget:=DestinoTeoricoAbrir + AjustePosicionPar;        (* Nueva posición de destino motivada por la corrección por
Superimposed *)
        S_.MovPos:=1;*)
END_IF;
(*Apertura - No Zona A - Compuerta 20º-110º de apertura->Corrección incremento velocidad en CRM01 - DESTENSAR*)
IF CBAbrir AND (CRM01_PAR.VF < (K_.CRM01_ParVacio - 50)) AND (CRM02_PAR.VF > (K_.CRM02_ParVacio + 70)) AND NOT ZonaA THEN
        K_.CorreccionSI := 150;
        K_.LongitudSI := 3000;
        VCCorregirDestensar := VCCorregirDestensar + 1;
        S_.MovSuper := 1;
        (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
        SP_PosTarget:=DestinoTeoricoAbrir + AjustePosicionPar;        (* Nueva posición de destino motivada por la corrección por
Superimposed *)
        S_.MovPos:=1;*)
END_IF;


(*Cierre - Zona B - Compuerta 110º-30º de apertura->Corrección incremento velocidad en CRM01 - TENSAR*)
IF CBCerrar AND (CRM02_PAR.VF > (K_.CRM02_ParVacio - 15)) AND (CRM01_PAR.VF < 350) AND ZonaB THEN
        K_.CorreccionSI := 150;
        K_.LongitudSI := 3000;
        VCCorregirTensar := VCCorregirTensar + 1;
```

```
                    S_.MovSuper := 1;
                    (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
                    SP_PosTarget:=DestinoTeoricoCerrar + AjustePosicionPar;        (* Nueva posición de destino motivada por la corrección por
Superimposed *)
                    S_.MovPos:=1;*)
          END_IF;
          (*Cierre - Zona B - Compuerta 110º-30º de apertura->Corrección decremento velocidad en CRM01 - DESTENSAR*)
          IF CBCerrar AND (CRM02_PAR.VF < (K_.CRM02_ParVacio - 30)) AND (CRM01_PAR.VF > (K_.CRM01_ParVacio + 30)) AND ZonaB THEN
                    K_.CorreccionSI := -150;
                    K_.LongitudSI := 3000;
                    VCCorregirDestensar := VCCorregirDestensar + 1;
                    S_.MovSuper := 1;
                    (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
                    SP_PosTarget:=DestinoTeoricoCerrar + AjustePosicionPar;        (* Nueva posición de destino motivada por la corrección por
Superimposed *)
                    S_.MovPos:=1;*)
          END_IF;
          (*Cierre - No Zona B - Compuerta 30º-0º de apertura->Corrección incremento velocidad en CRM01 - TENSAR*)
          (* Par vacio CRM01= 6.5 -> +30 *)
          (* 27-Agosto: (K_.CRM02_ParVacio - 25) -> (K_.CRM02_ParVacio - 45)*)
          IF CBCerrar AND (CRM02_PAR.VF > (K_.CRM02_ParVacio - 45)) AND (CRM01_PAR.VF < (K_.CRM01_ParVacio + 12)) AND NOT ZonaB THEN
                    K_.CorreccionSI := 150;
                    K_.LongitudSI := 3000;
                    VCCorregirTensar := VCCorregirTensar + 1;
                    S_.MovSuper := 1;
                    (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
                    SP_PosTarget:=DestinoTeoricoCerrar + AjustePosicionPar;        (* Nueva posición de destino motivada por la corrección por
Superimposed *)
                    S_.MovPos:=1;*)
          END_IF;
          (*Cierre - No Zona B - Compuerta 30º-0º de apertura->Corrección decremento velocidad en CRM01 - DESTENSAR*)        (* Par
vacio CRM01= 6.5 -> +40 *)
          (* 27-Agosto (K_.CRM02_ParVacio - 35) -> (K_.CRM02_ParVacio - 55) *)
          IF CBCerrar AND (CRM02_PAR.VF < (K_.CRM02_ParVacio - 55)) AND (CRM01_PAR.VF > (K_.CRM01_ParVacio + 17)) AND NOT ZonaB THEN
                    K_.CorreccionSI := -150;
                    K_.LongitudSI := 3000;
                    VCCorregirDestensar := VCCorregirDestensar + 1;
                    S_.MovSuper := 1;
                    (*AjustePosicionPar:=AjustePosicionPar - K_.CorreccionSI;        (* Invertir signo de Corrección para compensar la posición de
cerrado/abierto *)
                    SP_PosTarget:=DestinoTeoricoCerrar + AjustePosicionPar;        (* Nueva posición de destino motivada por la corrección por
Superimposed *)
                    S_.MovPos:=1;*)
```

```
            END_IF;

END_IF;

(************************************** Paro motores ***********************************************)
(* Orden de paro servomotor Compuerta cierre *)
IF CRM01.ModoAcople=0 OR CRM01.ModoAcople=1 THEN
            CRM01.ParoServo(    Execute:=S_.Parar ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    Options:= ,
                                                    Axis:=CRM01.Eje ,
                        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(* Orden de paro servomotor Compuerta apertura *)
IF CRM02.ModoAcople=0 OR CRM02.ModoAcople=1 THEN
            CRM02.ParoServo(    Execute:=S_.Parar ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    Options:= ,
                                                    Axis:=CRM02.Eje ,
                        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(************************************** Mover Jog  ***********************************************)
(* Mover en manual en sentido positivo/negativo servo Compuerta cierre *)
IF CRM01.ModoAcople=0 OR CRM01.ModoAcople=1 THEN
            CRM01.Jog(                          JogForward:=S_.JogF ,
                                                    JogBackwards:=S_.JogB ,
                                                    Mode:=2 ,           (* MC_JOGMODE_CONTINOUS - usa parámetros del PLC *)
                                                    Position:= ,
                                                    Velocity:=SP_Vel ,
                                                    Acceleration:= ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    Axis:=CRM01.Eje ,
                        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(* Mover en manual en sentido positivo/negativo servo Compuerta apertura *)
IF CRM02.ModoAcople=0 OR CRM02.ModoAcople=1 THEN
            CRM02.Jog(                          JogForward:=S_.JogF ,
                                                    JogBackwards:=S_.JogB ,
                                                    Mode:=2 ,           (* MC_JOGMODE_CONTINOUS - usa parámetros del PLC *)
                                                    Position:= ,
```

```
                                                    Velocity:=SP_Vel ,
                                                    Acceleration:= ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    Axis:=CRM02.Eje ,
                    Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(************************************ Mover en Velocidad  ****************************************)
(* Mover en velocidad servomotor Compuerta cierre *)
IF CRM01.ModoAcople=0 OR CRM01.ModoAcople=1 THEN
        CRM01.MovVel(           Execute:=S_.MovVel ,
                                                    Velocity:=SP_Vel ,
                                                    Acceleration:= ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    Direction:= ,
                                                    BufferMode:= ,
                                                    Options:= ,
                                                    Axis:=CRM01.Eje ,
                    InVelocity=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(* Mover en velocidad servomotor Compuerta apertura *)
IF CRM02.ModoAcople=0 OR CRM02.ModoAcople=1 THEN
        CRM02.MovVel(           Execute:=S_.MovVel ,
                                                    Velocity:=SP_Vel ,
                                                    Acceleration:= ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    Direction:= ,
                                                    BufferMode:= ,
                                                    Options:= ,
                                                    Axis:=CRM02.Eje ,
                    InVelocity=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(********************************** Mover en Posición Absoluta ************************************)
(* Mover en posición absoluta servomotor Compuerta cierre *)
IF CRM01.ModoAcople=0 OR CRM01.ModoAcople=1 THEN
        CRM01.MovPos(       Execute:= S_.MovPos1,
                                                    Position:=SP_PosTarget ,
                                                    Velocity:=SP_Vel ,
                                                    Acceleration:= ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
```

```
                                                    BufferMode:= ,
                                                    Options:= ,
                                                    Axis:=CRM01.Eje ,
                    Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(* Mover en posición absoluta servomotor Compuerta apertura *)
IF CRM02.ModoAcople=0 OR CRM02.ModoAcople=1 THEN
        CRM02.MovPos(       Execute:= S_.MovPos,
                                                    Position:=SP_PosTarget ,
                                                    Velocity:=SP_Vel ,
                                                    Acceleration:= ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    BufferMode:= ,
                                                    Options:= ,
                                                    Axis:=CRM02.Eje ,
                    Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(********************************** Mover en Posición Relativa ************************************)
(* Mover en posición relativa servomotor Compuerta apertura *)
IF CRM02.ModoAcople=0 OR CRM02.ModoAcople=1 THEN
        CRM02.MovRel(               Execute:=S_.MovRel ,
                                                    Distance:=SP_RelTarget ,
                                                    Velocity:=SP_Vel ,
                                                    Acceleration:= ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    BufferMode:= ,
                                                    Options:= ,
                                                    Axis:=CRM02.Eje ,
                    Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(********************************** Cambiar posición actual ***************************************)
(* Es posible dar una posición "al vuelo" *)
(* Cambiar la posición actual del servomotor Compuerta cierre *)
IF CRM01.ModoAcople=0 OR CRM01.ModoAcople=1 THEN
        CRM01.SetPos(               Execute:=S_.PosActual ,
                                                    Position:=SP_PosActual ,
                                                    Mode:= ,
                                                    Options:= ,
                                                    Axis:=CRM01.Eje ,
                    Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(* Cambiar la posición actual del servomotor Compuerta apertura *)
IF CRM02.ModoAcople=0 OR CRM02.ModoAcople=1 THEN
        CRM02.SetPos(           Execute:=S_.PosActual1 ,
                                        Position:=SP_PosActual ,
                                        Mode:= ,
                                        Options:= ,
                                        Axis:=CRM02.Eje ,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(********************************* Acoplar/Desacoplar Ejes ***************************************)
(* Acople de servomotor de cierre como esclavo a servomotor apertura como maestro *)
(* Acoplar ejes con un ratio determinado - ABRIR *)
CRM01.Acople(           Enable:=S_.Acoplar ,
                                GearRatio:=CRM01.RatioAcople ,
                                Acceleration:= ,
                                Deceleration:= ,
                                Jerk:= ,
                                BufferMode:= ,
                                Options:= ,
                                Master:=CRM02.Eje ,
                                Slave:=CRM01.Eje ,
        InGear=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );

(* Desacoplar ejes *)
CRM01.Desacople(    Execute:=S_.Desacoplar ,
                                Options:= ,
                                Slave:=CRM01.Eje ,
        Done=> ,Busy=> ,Error=> ,ErrorID=> );

(********************************** Mover con SuperImposed ***************************************)
CRM01.SuperImp(     Execute:=S_.MovSuper ,
                                Mode:= ,
                                Distance:=K_.CorreccionSI ,
                                VelocityDiff:=SP_Vel ,
                                Acceleration:= ,
                                Deceleration:= ,
                                Jerk:= ,
                                VelocityProcess:=SP_Vel ,
                                Length:=K_.LongitudSI ,
                                Options:= ,
                                Axis:=CRM01.Eje ,
        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> ,Warning=> ,WarningId=> ,
        ActualVelocityDiff=> ,ActualDistance=> ,ActualLength=> ,ActualAcceleration=> ,ActualDeceleration=> );
```

```
(********************************** Flags/Habilitaciones ****************************************)
(* Habilitar contactor alimentación 380V variadores desde web  *)
K:=S_.HabilitarK;

(* Borrado de flags, habilitaciones, pulsadores *)
S_.Acoplar:=0;                  (* Flag de habilitación Acople *)
S_.Acoplar1:=0;                 (* Flag de habilitación Acople *)
S_.Desacoplar:=0;       (* Flag de habilitación Desacoplar *)
S_.PosActual:=0;        (* Flag de habilitación Cambiar la posición actual de Servomotores *)
S_.PosActual1:=0;       (* Flag de habilitación Cambiar la posición actual de Servomotores *)
S_.MovPos:=0;                   (* Flag de habilitación de movimiento en posición absoluta *)
S_.MovPos1:=0;          (* Flag de habilitación de movimiento en posición absoluta *)
S_.MovRel:=0;                   (* Flag de habilitación de movimiento en posición relativa *)
S_.MovVel:=0;                   (* Flag de habilitación de mover en velocidad *)
S_.Inicio:=0;                   (* Flag de habilitación inicio de movimiento en automático *)
S_.Fin:=0;                      (* Flag de habilitación fin de movimiento en automático  *)
S_.Parar:=0;                    (* Flag de parar movimiento *)
S_.Mover:=0;                    (* Flag de mover servomotores *)
S_.Abrir:=0;                            (* Flag de habilitación abrir *)
S_.Cerrar:=0;                   (* Flag de habilitación cerrar *)
S_.MovSuper:=0;         (* Flag de habilitación opción SuperImposed *)
S_.MovSuper1:=0;        (* Flag de habilitación opción SuperImposed *)
S_.Reset:=0;                    (* Flag de Reset *)

(************************************ Visualizar en Web ******************************************)
(* Modo de acople *)
CRM01.ModoAcople:=DWORD_TO_INT(CRM01.Eje.NcToPlc.CoupleState);
CRM02.ModoAcople:=DWORD_TO_INT(CRM02.Eje.NcToPlc.CoupleState);
(* Código de diagnóstico *)
CRM01.DiagCode:=DWORD_TO_INT(CRM01_AT.DiagCode);
CRM02.DiagCode:=DWORD_TO_INT(CRM02_AT.DiagCode);
(* Valores variables en unidades reales *)
CRM01.Intensidad_A(Valor_0_1000:=CRM01_Intensidad.VF , FE:=35.2 , VR=> );
CRM01.Par_Nm(Valor_0_1000:=CRM01_PAR.VF , FE:=45.76 , VR=> );
CRM02.Intensidad_A(Valor_0_1000:=CRM02_Intensidad.VF , FE:=17.6 , VR=> );
CRM02.Par_Nm(Valor_0_1000:=CRM02_PAR.VF , FE:=22.88 , VR=> );
(************************************** Contadores ***************************************************)
IF CRM02.SuperImp.Done THEN cont_1:=cont_1+1; END_IF;
IF CRM01.SuperImp.Done THEN cont_2:=cont_2+1; END_IF;
```

8.2.4 FB11_WS. Llamada a FB de gestión de movimiento Wind Shield

```
FUNCTION_BLOCK FB11_WS
VAR
        Man: BOOL; (* Modo Manual *)
        Aut: BOOL; (* Modo Automático *)
        DEF: BOOL; (* Reunión Defectos *)
        AuxDone0: BOOL; (* Auxiliar operación ejecutada con éxito *)
        AuxDone1: BOOL; (* Auxiliar operación ejecutada con éxito *)
        AuxDone2: BOOL; (* Auxiliar operación ejecutada con éxito *)
        CBAbrir: BOOL; (* Orden de abrir windshield *)
        CBCerrar: BOOL; (* Orden de cerrar windshield *)
        HabControlPar: BOOL; (* Habilitar control de Par *)
        TiempoEspera: BOOL; (* En tiempo de espera *)
        Paso: INT; (* Número de paso en modo Automático *)
        cont_1: INT;
        cont_2: INT;
        VCTiempoEspera: INT; (* Contador ciclos sin aplicar control de Par *)
        VCCorregirDestensar: INT; (* Contador ciclos Corregir Destensar *)
        VCCorregirTensar: INT; (* Contador ciclos Corregir Tensar *)
        SP_Vel: LREAL; (* Setpoint de velocidad *)
        SP_PosTarget: LREAL; (* Setpoint de posición objetivo *)
        SP_Aut: LREAL; (* Setpoint de posición en Automático *)
        SP_Man: LREAL; (* Setpoint de posición en Manual *)
        SP_PosActual: LREAL; (* Setpoint de posición actual *)
        SP_PosActual1: LREAL; (* Setpoint de posición actual *)
        HabControlParF: R_TRIG; (* Habilitar control de Par Flanco Positivo *)
        WSM01: Servo; (* Control Servomotor Windshield inferior - WSM01 *)
        WSM02: Servo; (* Control Servomotor Windshield superior 1 - WSM02 *)
        WSM03: Servo; (* Control Servomotor Windshield superior 2 - WSM03 *)
        D_: Servo_D; (* Defectos Servomotor *)
        S_: Servo_S; (* Pulsadores Servomotor *)
        AuxPosLineal: FB21_DesarrolloLineal;
        AuxPosLineal1: FB21_DesarrolloLineal;
        WSM01_PAR: FB22_FiltroAnal; (* Motor inferior Par filtrado *)
        WSM02_PAR: FB22_FiltroAnal; (* Motor superior 1 Par filtrado *)
        WSM03_PAR: FB22_FiltroAnal; (* Motor superior 2 Par filtrado *)
        WSM01_Intensidad: FB22_FiltroAnal; (* Motor inferior Intensidad filtrada *)
        WSM02_Intensidad: FB22_FiltroAnal; (* Motor superior 1 Intensidad filtrada *)
        WSM03_Intensidad: FB22_FiltroAnal; (* Motor superior 2 Intensidad filtrada *)
END_VAR
VAR_INPUT
        Q: BOOL; (* Térmico alimentación variadores *)
        WDef: BOOL; (* Defecto externo *)
        SRea: BOOL; (* Pulsador de rearme *)
        WAut: BOOL; (* Condiciones marcha automático *)
```

```
END_VAR
VAR_OUTPUT
        K: BOOL; (* Marcha contactor alimentación variadores *)
END_VAR
VAR PERSISTENT
        K_: Servo_K; (* Constantes Servomotor *)
        Sick: Sick; (* Encoder absoluto Sick *)
END_VAR
```

```
(* FB control servomotores movimiento Windshield *)
(****************************************** Previo ***************************************************)
(* Constantes *)
K_.RecorridoLineal:=3960;                       (* [mm] *)
K_.ParMaxAbrir:=200;
K_.ParMaxCerrar:=200;
K_.ParControlAbrir:=-75;
K_.ParControlCerrar:=50;
K_.VelTrabajo:=3000;                            (* Velocidad de trabajo *)
K_.PosAbierto:=0;
K_.PosCerrado:=K_.RecorridoLineal;
K_.Ri_0:=205.9; K_.Ri_1:=205.9; K_.Ri_2:=205.9; K_.Ri_3:=205.9; K_.Ri_4:=205.9;     (* Radios tambor inferior. Despreciamos arrollamientos [mm] *)
K_.Rs_0:=104;                                   (* Radio tambor superior [mm] *)
WSM03.RatioAcople:=1;                           (* Ratio de acople Windshield superior esclavo *)
WSM01.RatioAcople:=0.505;                       (* Ratio de acople Windshield inferior esclavo *)
K_.WSM01_ParVacio:=50;                          (* Par en vacío de WSM01 - Tambor inferior *)
K_.WSM02_ParVacio:=30;                          (* Par en vacío de WSM02 - Tambor superior maestro *)
Sick.Offset :=1523079175;                       (* Offset encoder Sick - Valor fijo equivalente a cero vueltas, punto inicial *)

(* Asignar posición destino desde pantalla *)
IF S_.Abrir THEN K_.PosDestino:=0; END_IF;
IF S_.Cerrar THEN K_.PosDestino:=180000; END_IF;

(*Movimiento Abrir/Cerrar en función de la velocidad*)
CBCerrar := (WSM02.Eje.NcToPlc.ActVelo > 10);
CBAbrir := (WSM02.Eje.NcToPlc.ActVelo < -10);

(* Filtro señales analógicas *)
WSM01_PAR(AnalIn:=ABS(WSM01_AT.TorqueFeedbackValue));
WSM02_PAR(AnalIn:=ABS(WSM02_AT.TorqueFeedbackValue));
WSM03_PAR(AnalIn:=ABS(WSM03_AT.TorqueFeedbackValue));
WSM01_Intensidad(AnalIn:=ABS(WSM01_AT.ActualAbsoluteCurrent));
WSM02_Intensidad(AnalIn:=ABS(WSM02_AT.ActualAbsoluteCurrent));
WSM03_Intensidad(AnalIn:=ABS(WSM03_AT.ActualAbsoluteCurrent));

 (* Cálculo de la posición real mediante Encoder Absoluto conectado a Windshield inferior *)
Sick.PosAct := (DINT_TO_REAL(WSM01_AT.FeedbackValue2 - Sick.Offset) / Sick.K_Escala);          (* [vueltas tambor inferior] *)
```

```
(* Cálculo del desarrollo lineal mediante Encoder Absoluto conectado a Windshield inferior *)
AuxPosLineal(		Vueltas:=Sick.PosAct ,
					Ri_0:=K_.Ri_0 ,
					Ri_1:=K_.Ri_1 ,
					Ri_2:=K_.Ri_2 ,
					Ri_3:=K_.Ri_3 ,
					Ri_4:=K_.Rs_0 ,
		Rat=> ,
		Posicion_mm=>Sick.PosAct_mm );
(****************************************** Modos ******************************************************)
(* Modos Manual - Automático *)
Man:=NOT Aut;
Aut:=S_.Man_Aut;

(* Modos Velocidad - Par en WSM01 *)
S_.Velo_Par:=0;		(* Forzamos de momento trabajar siempre en modo Velocidad *)
IF S_.Velo_Par THEN
		WSM01_MDT.ControlWord:=WSM01_ControlWord_NCtoPLC OR 16#01;	(* Esto pone a 1 el bit 0 respetando el resto de bits *)
ELSE
		WSM01_MDT.ControlWord:=WSM01_ControlWord_NCtoPLC AND 16#F8;	(* Esto pone a 0 los bits 0, 1 y 2 respetando el resto de bits *)
END_IF;

(* Gestión Modo Automático - Manual *)
IF Aut THEN
		S_.HabilitarK:=1;				(* Habilitar contactor de potencia *)
		IF S_.Fin OR (Def AND (Paso>12)) THEN		(* Pulsando Fin o Defecto en cualquier momento forzamos parada segura *)
		S_.Parar:=1;				(* Habilitar flag de parada *)
		Paso:=10;				(* Cambiar a Paso 10 *)
END_IF;
CASE Paso OF
		0:		(* Inicio - Todo deshabilitado *)
			IF WSM01.ModoAcople<>0 OR WSM02.ModoAcople<>0 OR WSM03.ModoAcople<>0 THEN
				S_.Desacoplar:=1;		(* Si algún motor acoplado -> desacoplar *)
			END_IF;
			IF NOT Def THEN
				S_.Reset:=1;			(* Habilitar reset de NC *)
				Paso:=1;			(* Cambiar a Paso 1 *)
			END_IF;

		1:		(* Esperar Drive Ready *)
			IF WSM01.ModoAcople<>0 OR WSM02.ModoAcople<>0 OR WSM03.ModoAcople<>0 THEN
				S_.Desacoplar:=1;		(* Si algún motor acoplado -> desacoplar *)
			END_IF;
```

```
                IF ((WSM01_AT.DiagCode=16#D012 AND WSM02_AT.DiagCode=16#D012 AND WSM03_AT.DiagCode=16#D012 AND S_.Inicio)
                OR (WSM01_AT.DiagCode=16#D013 AND WSM02_AT.DiagCode=16#D013 AND WSM03_AT.DiagCode=16#D013))
                 THEN      (* Drive Ready *)
                           S_.Habilitar:=1;                    (* Habilitar variador *)
                           Paso:=2;                            (* Cambiar a Paso 2 *)
                END_IF;

        2:      (* Servos preparados *)
                IF WSM01_AT.DiagCode=16#D013 AND WSM02_AT.DiagCode=16#D013 AND WSM03_AT.DiagCode=16#D013 AND S_.Mover
                THEN      (* Axis OP y Orden de mover *)
                           SP_PosActual:=(Sick.PosAct_mm / AuxPosLineal.Ds) * 36000;      (* Actualizar posición con encoder absoluto *)
                           SP_PosActual1:=Sick.PosAct * 36000;                  (* Actualizar posición con encoder absoluto tambor inferior *)
                           S_.PosActual:=1;                                     (* Habilitar cambio de posición actual Servos *)
                           S_.PosActual1:=1;
                           Paso:=3;                                             (* Cambiar a Paso 3 *)
                END_IF;

        3:      (* Orden de Mover - Actualizar posición con encoder Sick *)
                IF WSM01.SetPos.Done THEN AuxDone0:=1;END_IF;   (* Cambio de posición Ok *)
                IF WSM02.SetPos.Done THEN AuxDone1:=1;END_IF;   (* Cambio de posición Ok *)
                IF WSM03.SetPos.Done THEN AuxDone2:=1;END_IF;   (* Cambio de posición Ok *)
                IF AuxDone0 AND AuxDone1 AND AuxDone2 THEN
                           AuxDone0:=0; AuxDone1:=0; AuxDone2:=0;                   (* Borrar flags auxiliares *)
                           S_.Acoplar:=1;                                           (* Habilitar flag de acoplar *)
                           Paso:=4;                                                 (* Cambiar a Paso 4 *)
                END_IF;

        4:      (* Acoplar servos - Maestro WSM02 de Esclavo WSM03 y WSM01 *)
                IF WSM01.Acople.InGear THEN AuxDone0:=1;END_IF; (* Cambio de posición Ok *)
                IF WSM03.Acople.InGear THEN AuxDone1:=1;END_IF; (* Cambio de posición Ok *)
                IF AuxDone0 AND AuxDone1 AND                    (* Cambios de posición Ok y Límites de encoder Sick Ok *)
                 ((K_.PosDestino > WSM01.Eje.NcToPlc.ActPos AND (Sick.PosAct < 2.526)) OR (K_.PosDestino < WSM01.Eje.NcToPlc.ActPos AND
                (Sick.PosAct > -0.002)))THEN
                           AuxDone0:=0; AuxDone1:=0;                            (* Borrar flags auxiliares *)
                           SP_PosTarget:=K_.PosDestino;                         (* Transferir posción destino *)
                           SP_Vel:=K_.VelTrabajo;                               (* Transferir velocidad de trabajo *)
                           S_.MovPos:=1;                                        (* Habilitar movimiento en posición  *)
                           Paso:=20;                                            (* Cambiar a Paso 20 *)
                END_IF;

        10:     (* Secuencia de finalizar movimiento - Orden de parar *)
                IF NOT WSM01.Power.Status AND NOT WSM02.Power.Status AND NOT WSM03.Power.Status THEN                    (* Si
pierden habilitación los drives mandamos a inicio de secuencia *)
                           Paso:=0;
                END_IF;
                IF WSM01.ParoServo.Done OR  WSM02.ParoServo.Done OR WSM03.ParoServo.Done THEN        (* Parada ok *)
```

```
                    S_.Desacoplar:=1;                                   (* Habilitar flag de desacoplar *)
                    Paso:=11;                                           (* Cambiar a Paso 11 *)
                END_IF;

        11:     (* Desacoplar Servos *)
                IF NOT WSM01.Power.Status AND NOT WSM02.Power.Status AND NOT WSM03.Power.Status THEN     (* Si
pierden habilitación los drives mandamos a inicio de secuencia *)
                    Paso:=0;
                END_IF;
                IF WSM01.Desacople.Done THEN AuxDone0:=1;END_IF;        (* Desacople Ok *)
                IF WSM03.Desacople.Done THEN AuxDone1:=1;END_IF;        (* Desacople Ok *)
                IF AuxDone0 AND AuxDone1 THEN
                    AuxDone0:=0; AuxDone1:=0;                           (* Borrar flags auxiliares *)
                    S_.Reset:=1;                                        (* Ejecutar reset NC y Drive después de desacoplar *)
                    S_.Habilitar:=0;                                    (* Deshabilitar variador *)
                    Paso:=12;                                           (* Cambiar a Paso 12 *)
                END_IF;

        12:     (* Deshabilitar variador *)
                IF WSM01_AT.DiagCode=16#D012 AND WSM02_AT.DiagCode=16#D012 AND WSM03_AT.DiagCode=16#D012 THEN    (* Drive
Ready *)
                    Paso:=1;
                END_IF;

        20:     (* Servos preparados - MOVIENDO *)
                IF ((Sick.PosAct > 2.526) AND CBCerrar) OR ((Sick.PosAct < -0.002) AND CBAbrir) THEN   (* Control límites encoder absoluto *)
                    cont_1:=cont_1+1;
                    S_.Parar:=1;                                        (* Ejecutar orden de parar si se sobrepasa los límites *)
                END_IF;
                IF WSM02.MovPos.Done OR WSM02.ParoServo.Done THEN                   (* Parada ok *)
                    S_.Desacoplar:=1;                                   (* Habilitar flag de desacoplar *)
                    Paso:=21;                                           (* Cambiar a Paso 11 *)
                END_IF;

        21:     (* Fin de movimiento - Desacoplar servos *)
                IF WSM01.Desacople.Done THEN AuxDone0:=1;END_IF;        (* Desacople Ok *)
                IF WSM03.Desacople.Done THEN AuxDone1:=1;END_IF;        (* Desacople Ok *)
                IF AuxDone0 AND AuxDone1 THEN
                    AuxDone0:=0; AuxDone1:=0;                           (* Borrar flags auxiliares *)
                    S_.Reset:=1;                                        (* Ejecutar reset NC y Drive después de desacoplar *)
                    S_.Habilitar:=0;                                    (* Deshabilitar variador *)
                    Paso:=1;                                            (* Cambiar a Paso 1 *)
                END_IF;

END_CASE;
ELSE
```

```
		(* Manual *)
		IF D_.EMG THEN 					(* Si Emergencia se realiza parada segura *)
			S_.Parar:=1;
		END_IF;
		Paso:=0;
		SP_PosTarget:=SP_Man;				(* Actualizar consigna de posición en automático *)
END_IF;

(* Control límites consigna de abierto/cerrado *)
IF SP_PosTarget >= 180000 THEN SP_PosTarget:=180000; END_IF;
IF SP_PosTarget <= 0 THEN SP_PosTarget:=0; END_IF;
(************************************ Potencia variadores ***************************************)
(* Habilitar potencia variador Windshield inferior *)
WSM01.Power(		Enable:=S_.Habilitar ,
					Enable_Positive:=S_.Habilitar ,
					Enable_Negative:=S_.Habilitar ,
					Override:=100 ,
					BufferMode:= ,
				Axis:=WSM01.Eje ,
		Status=> ,Busy=> ,Active=> ,Error=> ,ErrorID=> );

(* Habilitar potencia variador Windshield superior 1 *)
WSM02.Power(		Enable:=S_.Habilitar ,
					Enable_Positive:=S_.Habilitar ,
					Enable_Negative:=S_.Habilitar ,
					Override:=100 ,
					BufferMode:= ,
				Axis:=WSM02.Eje ,
		Status=> ,Busy=> ,Active=> ,Error=> ,ErrorID=> );

(* Habilitar potencia variador Windshield superior 2 *)
WSM03.Power(		Enable:=S_.Habilitar ,
					Enable_Positive:=S_.Habilitar ,
					Enable_Negative:=S_.Habilitar ,
					Override:=100 ,
					BufferMode:= ,
				Axis:=WSM03.Eje ,
		Status=> ,Busy=> ,Active=> ,Error=> ,ErrorID=> );
(*************************** Rearme general / Reset variadores ********************************)
(* Rearme Defectos *)
IF SRea THEN
		D_.Q:=0;
		D_.Acople:=0;
		D_.Hab:=0;
		D_.EMG:=0;
END_IF;
```

```
(* Reset Servo Windshield inferior, conveniente ejecutar antes que reset de NC *)
WSM01.ResetServo(  NetId:= ,
                                                    Execute:= SRea OR S_.Reset,
                                                    Timeout:= ,
                                                    Axis:= WSM01.Eje,
        Busy=> ,Error=> ,AdsErrId=> ,SercosErrId=> );

(* Reset Servo Windshield superior 1, conveniente ejecutar antes que reset de NC *)
WSM02.ResetServo(  NetId:= ,
                                                    Execute:= SRea OR S_.Reset,
                                                    Timeout:= ,
                                                    Axis:= WSM02.Eje,
        Busy=> ,Error=> ,AdsErrId=> ,SercosErrId=> );

(* Reset Servo Windshield superior 2, conveniente ejecutar antes que reset de NC *)
WSM03.ResetServo(  NetId:= ,
                                                    Execute:= SRea OR S_.Reset,
                                                    Timeout:= ,
                                                    Axis:= WSM03.Eje,
        Busy=> ,Error=> ,AdsErrId=> ,SercosErrId=> );

(* Reset NC Windshield inferior, conveniente ejecutar después de reset de Servo *)
IF WSM01.ModoAcople=0 OR WSM01.ModoAcople=1 THEN
        WSM01.ResetNC(          Execute:=SRea OR S_.Reset ,
                                                    Axis:=WSM01.Eje,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(* Reset NC Windshield superior 1, conveniente ejecutar después de reset de Servo *)
IF WSM02.ModoAcople=0 OR WSM02.ModoAcople=1 THEN
        WSM02.ResetNC(          Execute:=SRea OR S_.Reset ,
                                                    Axis:=WSM02.Eje,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(* Reset NC Windshield superior 2, conveniente ejecutar después de reset de Servo *)
IF WSM03.ModoAcople=0 OR WSM03.ModoAcople=1 THEN
        WSM03.ResetNC(          Execute:=SRea OR S_.Reset ,
                                                    Axis:=WSM03.Eje ,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(************************************* Defectos ************************************************)
(* Defectos HW *)
IF NOT Q THEN D_.Q:=1; END_IF;                          (* Defecto Térmico *)
IF WDef THEN D_.EMG:=1; END_IF;                         (* Defecto Parada de emergencia *)

(* Defectos SW *)
(*IF (WSM02.ModoAcople<>1 OR WSM03.ModoAcople <>3) THEN
        D_.Acople:=1; ELSE D_.Acople:=0;                (* Defecto Acople *)
END_IF;
*)
(*
IF WSM03.ModoAcople=0 AND  ABS(WSM02.Eje.NcToPlc.ActPos - WSM03.Eje.NcToPlc.ActPos)>K_.DefPos THEN
        D_.DefPos:=1; ELSE D_.DefPos:=0;                (* Defecto de posición entre servos sólo cuando no están acoplados *)
END_IF;
*)
IF (Paso=20) AND (NOT WSM01.Power.Status OR NOT WSM02.Power.Status OR NOT WSM03.Power.Status) THEN
        D_.Hab:=1;                                                      (* Defecto habilitación en motores *)
END_IF;

(* Reunión Defectos *)
Def:=D_.Q OR D_.Acople OR D_.DefPos OR D_.Hab OR D_.EMG;

(************************************* Activaciones Pasos ******************************************)
(* Habilitación funcionamiento con Control de Par *)
(*
HabControlPar := (Paso = 20);
*)
HabControlParF(CLK:=HabControlPar , Q=> );
(* Gestión par tiempo de espera *)
IF WSM01.SuperImp.Done OR HabControlParF.Q THEN
        TiempoEspera:=1;
        VCTiempoEspera :=0;
END_IF;
IF TiempoEspera THEN
        VCTiempoEspera := VCTiempoEspera + 1;
END_IF;
IF (VCTiempoEspera>=50) THEN
        TiempoEspera:=0;
END_IF;
```

```
(*Gestón par mínimo*)
(*Velocidad real dentro de una ventana de trababjo para evitar las rampas de aceleración y decelaración*)
(*
IF HabControlPar AND NOT TiempoEspera AND NOT WSM01.SuperImp.Busy AND (ABS((ABS(WSM02.Eje.NcToPlc.ActVelo)) - SP_Vel) < 10) THEN
	(*Apertura -> Corrección incremento velocidad en WSM01 - TENSAR*)
	IF CBAbrir AND (WSM02_PAR.VF > (K_.WSM02_ParVacio - 20)) AND (WSM01_PAR.VF < 200) THEN
		K_.CorreccionSI := 50;
		K_.LongitudSI := 1000;
		VCCorregirTensar := VCCorregirTensar + 1;
		S_.MovSuper := 1;
	END_IF;
	(*Apertura - > Corrección decremento velocidad en WSM01 - DESTENSAR*)
	IF CBAbrir AND (WSM02_PAR.VF < (K_.WSM02_ParVacio - 30)) AND (WSM01_PAR.VF > (K_.WSM01_ParVacio + 30)) THEN
		K_.CorreccionSI := -50;
		K_.LongitudSI := 1000;
		VCCorregirDestensar := VCCorregirDestensar + 1;
		S_.MovSuper := 1;
	END_IF;
	(*Cierre -> Corrección decremento velocidad en WSM01 - TENSAR*)
	IF CBCerrar AND (WSM01_PAR.VF > (K_.WSM02_ParVacio - 20)) AND (WSM02_PAR.VF < 150) THEN
		K_.CorreccionSI := -50;
		K_.LongitudSI := 1000;
		VCCorregirTensar := VCCorregirTensar + 1;
		S_.MovSuper := 1;
	END_IF;
	(*Cierre -> Corrección incremento velocidad en WSM01 - DESTENSAR*)
	IF CBCerrar AND (WSM01_PAR.VF < (K_.WSM01_ParVacio - 30)) AND (WSM02_PAR.VF > (K_.WSM02_ParVacio + 20)) THEN
		K_.CorreccionSI := 50;
		K_.LongitudSI := 1000;
		VCCorregirDestensar := VCCorregirDestensar + 1;
		S_.MovSuper := 1;
	END_IF;
END_IF;
*)

(****************************************** Paro motores ***********************************************)
(* Orden de paro servomotor Windshield inferior *)
IF WSM01.ModoAcople=0 OR WSM01.ModoAcople=1 THEN
	WSM01.ParoServo(    Execute:=S_.Parar ,
                                                    Deceleration:= ,
                                                    Jerk:= ,
                                                    Options:= ,
                                                    Axis:=WSM01.Eje ,
                    Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(* Orden de paro servomotor Windshield superior 1 *)
IF WSM02.ModoAcople=0 OR WSM02.ModoAcople=1 THEN
        WSM02.ParoServo(    Execute:=S_.Parar ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                Options:= ,
                                                Axis:=WSM02.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
(* Orden de paro servomotor Windshield superior 2 *)
IF WSM03.ModoAcople=0 OR WSM03.ModoAcople=1 THEN
        WSM03.ParoServo(    Execute:=S_.Parar ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                Options:= ,
                                                Axis:=WSM03.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
(****************************************** Mover Jog  **********************************************)
(* Mover en manual en sentido positivo/negativo servo Windshield inferior *)
IF WSM01.ModoAcople=0 OR WSM01.ModoAcople=1 THEN
        WSM01.Jog(                  JogForward:=S_.JogF ,
                                                JogBackwards:=S_.JogB ,
                                                Mode:=2 ,           (* MC_JOGMODE_CONTINOUS - usa parámetros del PLC *)
                                                Position:= ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                Axis:=WSM01.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(* Mover en manual en sentido positivo/negativo servo Windshield superior 1 *)
IF WSM02.ModoAcople=0 OR WSM02.ModoAcople=1 THEN
        WSM02.Jog(                  JogForward:=S_.JogF ,
                                                JogBackwards:=S_.JogB ,
                                                Mode:=2 ,           (* MC_JOGMODE_CONTINOUS - usa parámetros del PLC *)
                                                Position:= ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                Axis:=WSM02.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(* Mover en manual en sentido positivo/negativo servo Windshield superior 2 *)
IF WSM03.ModoAcople=0 OR WSM03.ModoAcople=1 THEN
        WSM03.Jog(              JogForward:=S_.JogF ,
                                        JogBackwards:=S_.JogB ,
                                        Mode:=2 ,       (* MC_JOGMODE_CONTINOUS - usa parámetros del PLC *)
                                        Position:= ,
                                        Velocity:=SP_Vel ,
                                        Acceleration:= ,
                                        Deceleration:= ,
                                        Jerk:= ,
                                        Axis:=WSM03.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(************************************ Mover en Velocidad  ****************************************)
(* Mover en velocidad servomotor Windshield inferior *)
IF WSM01.ModoAcople=0 OR WSM01.ModoAcople=1 THEN
        WSM01.MovVel(   Execute:=S_.MovVel ,
                                        Velocity:=SP_Vel ,
                                        Acceleration:= ,
                                        Deceleration:= ,
                                        Jerk:= ,
                                        Direction:= ,
                                        BufferMode:= ,
                                        Options:= ,
                                        Axis:=WSM01.Eje ,
                InVelocity=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(* Mover en velocidad servomotor Windshield superior 1 *)
IF WSM02.ModoAcople=0 OR WSM02.ModoAcople=1 THEN
        WSM02.MovVel(   Execute:=S_.MovVel ,
                                        Velocity:=SP_Vel ,
                                        Acceleration:= ,
                                        Deceleration:= ,
                                        Jerk:= ,
                                        Direction:= ,
                                        BufferMode:= ,
                                        Options:= ,
                                        Axis:=WSM02.Eje ,
                InVelocity=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(* Mover en velocidad servomotor Windshield superior 2 *)
IF WSM03.ModoAcople=0 OR WSM03.ModoAcople=1 THEN
        WSM03.MovVel(   Execute:=S_.MovVel ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                Direction:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Axis:=WSM03.Eje ,
                InVelocity=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(************************************ Mover en Posición ****************************************)
(* Mover en posición servomotor Windshield inferior *)
IF WSM01.ModoAcople=0 OR WSM01.ModoAcople=1 THEN
        WSM01.MovPos(   Execute:= S_.MovPos1,
                                                Position:=SP_PosTarget ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Axis:=WSM01.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;

(* Mover en posición servomotor Windshield superior 1 *)
IF WSM02.ModoAcople=0 OR WSM02.ModoAcople=1 THEN
        WSM02.MovPos(   Execute:= S_.MovPos,
                                                Position:=SP_PosTarget ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Axis:=WSM02.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(* Mover en posición servomotor Windshield superior 2 *)
IF WSM03.ModoAcople=0 OR WSM03.ModoAcople=1 THEN
        WSM03.MovPos(   Execute:= S_.MovPos,
                                                Position:=SP_PosTarget ,
                                                Velocity:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Axis:=WSM03.Eje ,
                Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );
END_IF;
(******************************** Cambiar posición actual ****************************************)
(* Es posible dar una posición "al vuelo" *)
(* Cambiar la posición actual del servomotor Windshield inferior *)
IF WSM01.ModoAcople=0 OR WSM01.ModoAcople=1 THEN
        WSM01.SetPos(   Execute:=S_.PosActual1 ,
                                                Position:=SP_PosActual ,
                                                Mode:= ,
                                                Options:= ,
                                                Axis:=WSM01.Eje ,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(* Cambiar la posición actual del servomotor Windshield superior 1 *)
IF WSM02.ModoAcople=0 OR WSM02.ModoAcople=1 THEN
        WSM02.SetPos(   Execute:=S_.PosActual ,
                                                Position:=SP_PosActual ,
                                                Mode:= ,
                                                Options:= ,
                                                Axis:=WSM02.Eje ,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;

(* Cambiar la posición actual del servomotor Windshield superior 2 *)
IF WSM03.ModoAcople=0 OR WSM03.ModoAcople=1 THEN
        WSM03.SetPos(   Execute:=S_.PosActual ,
                                                Position:=SP_PosActual ,
                                                Mode:= ,
                                                Options:= ,
                                                Axis:=WSM03.Eje ,
                Done=> ,Busy=> ,Error=> ,ErrorID=> );
END_IF;
```

```
(******************************** Acoplar/Desacoplar Ejes ****************************************)
(* Acople de servomotor Windshield superior 2 como esclavo a servomotor Windshield superior 1 como maestro *)
(* Acoplar ejes con un ratio determinado - CERRAR y ABRIR *)
WSM03.Acople(                   Enable:=S_.Acoplar ,
                                                GearRatio:=WSM03.RatioAcople ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Master:=WSM02.Eje ,
                                                Slave:=WSM03.Eje ,
        InGear=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );

(* Acople de servomotor Windshield inferior como esclavo a servomotor Windshield superior 1 como maestro *)
(* Acoplar ejes con un ratio determinado - CERRAR y ABRIR *)
WSM01.Acople(                   Enable:=S_.Acoplar ,
                                                GearRatio:=WSM01.RatioAcople ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                BufferMode:= ,
                                                Options:= ,
                                                Master:=WSM02.Eje ,
                                                Slave:=WSM01.Eje ,
        InGear=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> );

(* Desacoplar ejes *)
WSM01.Desacople(    Execute:=S_.Desacoplar ,
                                                Options:= ,
                                                Slave:=WSM01.Eje ,
        Done=> ,Busy=> ,Error=> ,ErrorID=> );
WSM02.Desacople(    Execute:=S_.Desacoplar ,
                                                Options:= ,
                                                Slave:=WSM02.Eje ,
        Done=> ,Busy=> ,Error=> ,ErrorID=> );
WSM03.Desacople(    Execute:=S_.Desacoplar ,
                                                Options:= ,
                                                Slave:=WSM03.Eje ,
        Done=> ,Busy=> ,Error=> ,ErrorID=> );
```

```
(********************************** Mover con SuperImposed ************************************)
WSM01.SuperImp(    Execute:=S_.MovSuper ,
                                                Mode:= ,
                                                Distance:=K_.CorreccionSI ,
                                                VelocityDiff:=SP_Vel ,
                                                Acceleration:= ,
                                                Deceleration:= ,
                                                Jerk:= ,
                                                VelocityProcess:=SP_Vel ,
                                                Length:=K_.LongitudSI ,
                                                Options:= ,
                                                Axis:=WSM01.Eje ,
        Done=> ,Busy=> ,Active=> ,CommandAborted=> ,Error=> ,ErrorID=> ,Warning=> ,WarningId=> ,
        ActualVelocityDiff=> ,ActualDistance=> ,ActualLength=> ,ActualAcceleration=> ,ActualDeceleration=> );

(********************************** Flags/Habilitaciones ***************************************)
(* Habilitar contactor alimentación 380V variadores desde web  *)
K:=S_.HabilitarK;

(* Borrado de flags, habilitaciones, pulsadores *)
S_.Acoplar:=0;              (* Flag de habilitación Acople *)
S_.Acoplar1:=0;             (* Flag de habilitación Acople *)
S_.Desacoplar:=0;           (* Flag de habilitación Desacoplar *)
S_.PosActual:=0;            (* Flag de habilitación Cambiar la posición actual de Servomotores *)
S_.PosActual1:=0;           (* Flag de habilitación Cambiar la posición actual de Servomotores *)
S_.MovPos:=0;               (* Flag de habilitación de movimiento *)
S_.MovPos1:=0;              (* Flag de habilitación de movimiento *)
S_.Inicio:=0;                       (* Flag de habilitación inicio de movimiento en automático *)
S_.Fin:=0;                  (* Flag de habilitación fin de movimiento en automático  *)
S_.Parar:=0;                        (* Flag de parar movimiento *)
S_.Mover:=0;                        (* Flag de mover servomotores *)
Sick.S_Offset := 0;         (* Flag de habilitación referencia encoder de Sick *)
S_.Abrir:=0;                        (* Flag de habilitación abrir *)
S_.Cerrar:=0;                       (* Flag de habilitación cerrar *)
S_.Reset:=0;                        (* Flag de Reset *)
S_.MovSuper:=0;             (* Flag de movimiento con Superimposed *)

(********************************** Visualizar en Web ******************************************)
(* Modo de acople *)
WSM01.ModoAcople:=DWORD_TO_INT(WSM01.Eje.NcToPlc.CoupleState);
WSM02.ModoAcople:=DWORD_TO_INT(WSM02.Eje.NcToPlc.CoupleState);
WSM03.ModoAcople:=DWORD_TO_INT(WSM03.Eje.NcToPlc.CoupleState);
(* Código de diagnóstico *)
WSM01.DiagCode:=DWORD_TO_INT(WSM01_AT.DiagCode);
WSM02.DiagCode:=DWORD_TO_INT(WSM02_AT.DiagCode);
WSM03.DiagCode:=DWORD_TO_INT(WSM03_AT.DiagCode);
```

```
(* Valores variables en unidades reales *)
WSM01.Intensidad_A(Valor_0_1000:=WSM01_Intensidad.VF , FE:=80.9 , VR=> );
WSM01.Par_Nm(Valor_0_1000:=WSM01_PAR.VF , FE:=108 , VR=> );

WSM02.Intensidad_A(Valor_0_1000:=WSM02_Intensidad.VF , FE:=50.9 , VR=> );
WSM02.Par_Nm(Valor_0_1000:=WSM02_PAR.VF , FE:=52.5 , VR=> );

WSM03.Intensidad_A(Valor_0_1000:=WSM03_Intensidad.VF , FE:=50.9 , VR=> );
WSM03.Par_Nm(Valor_0_1000:=WSM03_PAR.VF , FE:=52.5 , VR=> );
```

8.2.5 FB13_RC. Llamada a FB de gestión de Resistencias Calefactoras

```
FUNCTION_BLOCK FB13_RC
VAR
        Def: BOOL; (* Reunión Defectos *)
        D_: Res_D; (* Defectos Resistencias calefactoras *)
        S_: Res_S; (* Pulsadores Resistencias calefactoras *)
END_VAR
VAR_INPUT
        Q: BOOL; (* Térmico alimentación variadores *)
        TM01: BOOL; (* Termostato Azimut y Compuerta fija *)
        TM02: BOOL; (* Termostato Compuerta superiror *)
        TM03: BOOL; (* Termostato Compuerta inferior *)
        WDef: BOOL; (* Defecto externo *)
        SRea: BOOL; (* Pulsador de rearme *)
        WAut: BOOL; (* Condiciones marcha automático *)
END_VAR
VAR_OUTPUT
        K: BOOL; (* Marcha contactor alimentación resistencias *)
END_VAR
```

```
(* FB control Resistencias Calefactoras *)
(* Marcha / Paro manual y tratamiento de defectos únicamente como avisos *)

(************************************ Rearme general ************************************************)
(* Rearme Defectos *)
IF SRea THEN
        D_.Q:=0;                    (* Térmico *)
        D_.TM01:=0;                 (* Termostato compuerta fija y anillo azimutal *)
        D_.TM02:=0;                 (* Termostato compuerta superior y guiado *)
        D_.TM03:=0;                 (* Termostato compuerta inferior y guiado *)
END_IF;

(************************************** Defectos ************************************************)
(* Defectos HW *)
IF NOT Q THEN D_.Q:=1; END_IF;                          (* Defecto Térmico *)
IF NOT TM01 THEN D_.TM01:=1; END_IF;            (* Defecto Termostato compuerta fija y anillo azimutal *)
IF NOT TM02 THEN D_.TM02:=1; END_IF;            (* Defecto Termostato compuerta superior y guiado *)
IF NOT TM03 THEN D_.TM03:=1; END_IF;            (* Defecto Termostato compuerta inferior y guiado *)

(* Reunión Defectos *)
Def:=D_.Q OR D_.TM01 OR D_.TM02 OR D_.TM03;

(********************************** Flags/Habilitaciones ****************************************)
(* Habilitar contactor alimentación 380V resistencias desde web  *)
K:=S_.HabilitarK AND NOT WDef;
```

8.2.6 FB20_MenorDistancia

```
FUNCTION_BLOCK FB20_MenorDistancia
VAR
        D1: LREAL; (* Distancia 1 *)
        D2: LREAL; (* Distancia 2 *)
        SP_Aux: LREAL; (* Cálculo del SP auxiliar *)
END_VAR
VAR_INPUT
        SP: LREAL; (* Setpoint de posición entrada *)
        Pos: LREAL; (* Posción actual *)
END_VAR
```

```
* Función para el cálculo del SP para el servo por el camino más corto  *)

(* Cálculo de distancias entre posición actual y SP *)
D1:=36000-ABS(SP-Pos);
D2:=Pos-SP;

(* Determinar que distancia es más corta *)
(* Distancia 2 menor que distancia 1 *)
IF ABS(D1) >= ABS(D2) THEN
        SP_Aux:=Pos-D2;
(* Distancia 1 menor que distancia 2 *)
ELSIF ABS(D1) < ABS(D2) THEN
        IF D2 < 0 THEN
                SP_Aux:=Pos-D1;
        ELSE
                SP_Aux:=Pos+D1;
        END_IF;
END_IF;
```

8.2.7 FB21_DesarolloLineal

```
FUNCTION_BLOCK FB21_DesarrolloLineal
VAR
        i: INT; (* auxiliar contaje *)
        Ds: REAL; (* Desarrollo lineal tambor superior *)
        Di: ARRAY [0..4] OF REAL; (* Desarrollo lineal del tambor inferior *)
        Ri: ARRAY [0..4] OF REAL; (* Radio del tambor inferior *)
END_VAR
VAR_INPUT
        Vueltas: REAL; (* Número de vueltas actual *)
        Ri_0: REAL; (* Radio primitivo *)
        Ri_1: REAL; (* Radio 1 vuelta *)
        Ri_2: REAL; (* Radio 2 vueltas *)
        Ri_3: REAL; (* Radio 3 vueltas *)
        Ri_4: REAL; (* Radio 4 vueltas *)
        Rs_0: REAL; (* Radio primitivo *)
END_VAR
VAR_OUTPUT
        Rat: REAL;
        Posicion_mm: REAL; (* Posición actual en mm *)
END_VAR
```

```
(* Función para el cálculo de la posición por medio del desarrollo lineal del tambor  *)
(* Constantes Radio variable dependiendo del arrollamiento Tambor inferior *)
Ri[0]:=Ri_0; Ri[1]:=Ri_1; Ri[2]:=Ri_2; Ri[3]:=Ri_3; Ri[4]:=Ri_4;
(* Desarrollo lineal tambor inferior *)
FOR i:=0 TO 4 DO
        Di[i]:=2 * PI * Ri[i];
END_FOR;
(* Desarrollo lineal tambor superior *)
Ds:=2 * PI * Rs_0;
(* Dependiendo de la vuelta actual se aplica diferente radio del tambor *)
IF Vueltas<1 THEN
        Rat:=Ds / Di[0];        Posicion_mm:=Vueltas * Di[0];                                   (* [mm] *)
ELSIF Vueltas>=1 AND Vueltas<2 THEN
        Rat:=Ds / Di[1];        Posicion_mm:=(2 * PI * Ri_1 * (Vueltas - 1)) + Di[0];           (* [mm] *)
ELSIF Vueltas>=2 AND Vueltas<3 THEN
        Rat:=Ds / Di[2];        Posicion_mm:=(2 * PI * Ri_2 * (Vueltas - 2)) + Di[0] + Di[1];   (* [mm] *)
ELSIF Vueltas>=3 AND Vueltas<4 THEN
        Rat:=Ds / Di[3];        Posicion_mm:=(2 * PI * Ri_3 * (Vueltas - 3)) + Di[0] + Di[1] + Di[2];                   (* [mm] *)
ELSIF Vueltas>=4 AND Vueltas<5 THEN
        Rat:=Ds / Di[4];        Posicion_mm:=(2 * PI * Ri_4 * (Vueltas - 4)) + Di[0] + Di[1] + Di[2] + Di[3];           (* [mm] *)
END_IF;
```

8.2.8 FB22_FiltroAnalogico

```
FUNCTION_BLOCK FB22_FiltroAnal
VAR
        I: INT;
        VF: INT;
        SUMATORIO: REAL;
        BUFFER: ARRAY[1..100] OF INT;
END_VAR
VAR_INPUT
        AnalIn: INT; (* Variable analogica de entrada en bruto *)
END_VAR
```

```
(*MOVER BUFFER 1 POSICION PARA GENERAR HUECO EN BUFFER[1]*)
FOR I:=99 TO 1 BY -1 DO
        BUFFER[I+1] := BUFFER[I];
END_FOR;

BUFFER[1] := AnalIn;

(*HACER LA MEDIA DE LOS ULTIM OS 100 VALORES EN VALOR FILTRADO*)
SUMATORIO := 0;
FOR I :=1 TO 100 DO
                SUMATORIO := SUMATORIO + INT_TO_REAL(BUFFER[I]);
END_FOR;
VF := REAL_TO_INT(SUMATORIO/100);
```

8.2.9 FB24_CompuertaCG

```
FUNCTION_BLOCK FB24_CompuertaCG
VAR
        K_cgmin_Encoder: LREAL; (* Constante posición en cº mínimo de Encoder  *)
        K_cgmax_Encoder: LREAL; (* Constante posición en cº máximo de Encoder  *)
        K_cgmin_Compuerta: LREAL; (* Constante posición en cº mínimo de Compuerta *)
        K_cgmax_Compuerta: LREAL; (* Constante posición en cº máximo de Compuerta *)
END_VAR
VAR_INPUT
        E_cg_Compuerta: LREAL; (* Entrada - Posición Compuertas en cº *)
        E_cg_Encoder: LREAL; (* Entrada - Posición Encoder en cº *)
END_VAR
VAR_OUTPUT
        S_cg_Compuerta: LREAL; (* Salida - Posición Compuertas en cº *)
        S_cg_Encoder: LREAL; (* Salida - Posición Encoder en cº *)
END_VAR
```

```
(* Función auxiliar para la conversión cº Tambor Encoder <-> cº Apertura Compuerta *)

(* Constantes *)
K_cgmin_Encoder:=-257000.0;
K_cgmax_Encoder:=280895.0;
K_cgmin_Compuerta:=400.0;
K_cgmax_Compuerta:=11000.0;

(* Cálculo de cº tambor encoder -> cº Apertura Compuerta *)
S_cg_Compuerta:=((E_cg_Encoder - K_cgmin_Encoder) * ((K_cgmax_Compuerta - K_cgmin_Compuerta) / (K_cgmax_Encoder - K_cgmin_Encoder))) +
K_cgmin_Compuerta;

(* Cálculo de cº Apertura Compuerta -> cº tambor encoder *)
S_cg_Encoder:=((E_cg_Compuerta - K_cgmin_Compuerta) / ((K_cgmax_Compuerta - K_cgmin_C
```

8.2.10 FB25_ProfibusServo_PLC1_PLC2

```
FUNCTION_BLOCK FB25_ProfibusServo_PLC1_PLC2
VAR_INPUT
        PLC1_Axis_Cmd: WORD; (* Axis bits commands *)
        PLC1_PosSP_Par: UINT; (* Axis Position SetPoint (1/100 deg) *)
        PLC1_JogSpd_Par: INT; (* Speed sepoint for jog move (1/100 deg) *)
END_VAR
VAR_OUTPUT
        HabilitarEje: BOOL; (* '1' Habilitar sincronización del Eje con SetPoint *)
        HomeEje: BOOL; (* '1' Ejecutar secuencia de Home / '0' Parar el movimiento *)
        JogFEje: BOOL; (* '1' Mover Jog sentido positivo Eje / '0' Parar el movimiento *)
        JogBEje: BOOL; (* '1' Mover Jog sentido negativo Eje / '0' Parar el movimiento *)
        SP_PosTarget: LREAL; (* Setpoint de posición absoluta objetivo en % *)
        SP_Vel: LREAL; (* Setpoint de velocidad *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC1 -> PLC2 *)

(* Comandos binarios *)
HabilitarEje:=PLC1_Axis_Cmd.0;
HomeEje:=PLC1_Axis_Cmd.1;
JogFEje:=PLC1_Axis_Cmd.2;
JogBEje:=PLC1_Axis_Cmd.3;

(* Valores de Consigna *)
SP_PosTarget:=UINT_TO_LREAL(PLC1_PosSP_Par);
SP_Vel:=INT_TO_LREAL(PLC1_JogSpd_Par);
```

8.2.11 FB26_ProfibusServo_PLC2_PLC1

```
FUNCTION_BLOCK FB26_ProfibusServo_PLC2_PLC1
VAR
        kk: REAL;
END_VAR
VAR_INPUT
        EjePreparado: BOOL; (* '1' Eje habilitado para movimiento y sin fallo / '0' Arranque de eje no finalizado *)
        EjePosHome: BOOL; (* '1' Eje en posición de Home / '0' Eje no está en Home *)
        EjeActivo: BOOL; (* '1' Eje en movimiento / '0' Eje parado *)
        EjeTarget: BOOL; (* '1' Eje alcanza posición de destino / '0' Eje no ha alcanzado la posición de destino *)
        EjeOk: BOOL; (* '1' Eje estado Ok / '0' Eje en error *)
        EjePosAbierto: BOOL; (* '1' Eje en posición de abierto / '0' Eje no está en posición de abierto *)
        EjePosCerrado: BOOL; (* '1' Eje en posición de cerrado / '0' Eje no está en posición de cerrado *)
        EjeFallo1: BOOL; (* '1' Eje en fallo 1 / '0' Eje sin fallo 1 *)
        EjeFallo2: BOOL; (* '1' Eje en fallo 2 / '0' Eje sin fallo 2 *)
        EjeFallo3: BOOL; (* '1' Eje en fallo 3 / '0' Eje sin fallo 3 *)
        EjeFallo4: BOOL; (* '1' Eje en fallo 4 / '0' Eje sin fallo 4 *)
        ActPos: LREAL; (* Eje posición actual [cg] *)
        ActConsumo: REAL; (* Eje consumo actual [A] *)
END_VAR
VAR_OUTPUT
        PLC1_Axis_Sta: WORD; (* Axis bits status *)
        PLC1_Pos_Mes: UINT; (* Axis actual position (1/100 deg) *)
        PLC1_Cur_Mes: INT; (* Actual current (1/10 A) *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC2 -> PLC1 *)
(* Bits de Estados *)
PLC1_Axis_Sta.0:=EjePreparado;
PLC1_Axis_Sta.1:=EjePosHome;
PLC1_Axis_Sta.2:=EjeActivo;
PLC1_Axis_Sta.3:=EjeTarget;
PLC1_Axis_Sta.4:=EjeOk;
PLC1_Axis_Sta.5:=EjePosAbierto;
PLC1_Axis_Sta.6:=EjePosCerrado;
PLC1_Axis_Sta.7:=EjeFallo1;
PLC1_Axis_Sta.8:=EjeFallo2;
PLC1_Axis_Sta.9:=EjeFallo3;
PLC1_Axis_Sta.10:=EjeFallo4;

(* Valores Medidos *)
PLC1_Pos_Mes:=LREAL_TO_UINT(ActPos);
PLC1_Cur_Mes:=REAL_TO_INT(ActConsumo);
```

8.2.12 FB27_ProfibusGen_PLC1_PLC2

```
FUNCTION_BLOCK FB27_ProfibusGen_PLC1_PLC2
VAR_INPUT
        PLC1_Gen_Cmd1: WORD; (* Axis bits commands *)
        PLC1_Gen_Cmd2: WORD; (* Axis bits commands *)
END_VAR
VAR_OUTPUT
        ReaPLC1: BOOL; (* Rearme desde PLC1 *)
        AutRemoto: BOOL; (* AutRemoto=0 Modo Local; AutRemoto=1 Modo Remoto *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC1 -> PLC2 *)

(* Comandos binarios *)
ReaPLC1:=PLC1_Gen_Cmd1.0;
AutRemoto:=PLC1_Gen_Cmd1.1;

(* Valores de Consigna *)
```

8.2.13 FB28_ProfibusGen_PLC2_PLC1

```
FUNCTION_BLOCK FB28_ProfibusGen_PLC2_PLC1
VAR_INPUT
        ReaPLC2: BOOL; (* Rearme desde PLC2 *)
END_VAR
VAR_OUTPUT
        PLC1_Gen_Sta1: WORD; (* Generales Bits de Estado *)
        PLC1_Gen_Sta2: WORD; (* Generales Bits de Estado *)
END_VAR
```

```
(* Tratamiento y adecuación de los variables de interconexión profibus PLC2 -> PLC1 *)

(* Bits de Estados *)
PLC1_Gen_Sta1.0:=ReaPLC2;


(* Valores Medidos *)
```

8.2.14 MAIN

```
PROGRAM MAIN
VAR
        Generales: FB00_GEN; (* FB de gestión de Generales *)
        Compuerta_Rendija: FB10_CR; (* FB de gestión de movimiento de Compuerta Rendija *)
        Windshield: FB11_WS; (* FB de gestión de movimiento de Windshield *)
        Azimutal: FB12_AZ; (* FB de gestión de movimiento Azimutal *)
        Resistencias: FB13_RC; (* FB de gestión de Resistencias Calefactoras *)
        AZ_PLC1_PLC2: FB25_ProfibusServo_PLC1_PLC2; (* FB de interconexión Profibus PLC1 -> PLC2 variables Azimut *)
        CR_PLC1_PLC2: FB25_ProfibusServo_PLC1_PLC2; (* FB de interconexión Profibus PLC1 -> PLC2 variables Compuerta *)
        WS_PLC1_PLC2: FB25_ProfibusServo_PLC1_PLC2; (* FB de interconexión Profibus PLC1 -> PLC2 variables Generales *)
        AZ_PLC2_PLC1: FB26_ProfibusServo_PLC2_PLC1; (* FB de interconexión Profibus PLC2 -> PLC1 variables Azimut *)
        CR_PLC2_PLC1: FB26_ProfibusServo_PLC2_PLC1; (* FB de interconexión Profibus PLC2 -> PLC1 variables Compuerta *)
        WS_PLC2_PLC1: FB26_ProfibusServo_PLC2_PLC1; (* FB de interconexión Profibus PLC2 -> PLC1 variables Generales *)
        GEN_PLC1_PLC2: FB27_ProfibusGen_PLC1_PLC2; (* FB de interconexión Profibus PLC1 - PLC2 variables Generales *)
        GEN_PLC2_PLC1: FB28_ProfibusGen_PLC2_PLC1; (* FB de interconexión Profibus PLC2 - PLC1 variables Generales *)
        UPS1: FB_S_UPS; (* Guarda variables persistentes en Compact Flash *)
END_VAR
```

```
(* Main - Llamada a todos los elementos de la Instalación *)
(* Ejecución cíclica del programa cada 10ms *)
(*Guarda variables persistentes en Compact Flash*)
(* No es necesario utilizar ninguna patilla de entrada, con llamar el bloque es suficiente *)
UPS1(                                   sNetID:= ,
                                                iPLCPort:= ,
                                                iUPSPort:= ,
                                                tTimeout:= ,
                                                eUpsMode:= ,
                                                ePersistentMode:= ,
                                                tRecoverTime:= ,
        bPowerFailDetect=> ,eState=> );
(* Llamada a FB de interconexión Profibus de variables "Generales" desde PLC1 a PLC2  *)
GEN_PLC1_PLC2(  PLC1_Gen_Cmd1:=PLC1_W.GEN.Gen_Cmd1 ,
                PLC1_Gen_Cmd2:=PLC1_W.GEN.Gen_Cmd2 ,
        ReaPLC1=>,
        AutRemoto=> );
(* Llamada a FB de interconexión Profibus de variables "Servos Azimut" desde PLC1 a PLC2  *)
AZ_PLC1_PLC2(   PLC1_Axis_Cmd:=PLC1_W.AZ.Axis_Cmd ,
                PLC1_PosSP_Par:=PLC1_W.AZ.PosSP_Par ,
                PLC1_JogSpd_Par:=PLC1_W.AZ.JogSpd_Par ,
        HabilitarEje=> ,
        HomeEje=> ,
        JogFEje=> ,
        JogBEje=> ,
        SP_PosTarget=> ,
        SP_Vel=> );
```

```
(* Llamada a FB de gestión de Generales *)
Generales(          SRea:=G_SH01 ,                  (* Entrada digital - Pulsador rearme instalación *)
                    TM01:=G_TM01 ,                  (* Entrada digital - Termostato armario *)
                    EMG:=NOT G_EMG ,                (* Programa seguridad HW - Señal Emergencia instalación *)
          KV=>G_KV ,                                (* Salida digital - Ventilador armario *)
          KRea=>G_Rea ,                             (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
          KRea_Remoto=> ,                           (* Programa seguridad HW - Señal Rearme Emergencia instalación remoto *)
          HRea=>G_H01 );                            (* Salida digital - Piloto Emergencia instalación *)

(* Llamada a FB de gestión de movimiento Azimutal *)
Azimutal(           Q:=AZ_Q ,                       (* Entrada digital - Térmico motores Azimutales *)
                    WDef:=NOT G_EMG ,               (* Programa seguridad HW - Señal Emergencia instalación *)
                    SRea:=G_Rea,                    (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                    WAut:= ,
          K=>AZ_K );                                (* Salida digital - Marcha alimentación 380V Variadores Azimutales *)

(* Llamada a FB de gestión de movimiento Compuerta Rendija *)
Compuerta_Rendija(  Q:=CR_Q ,                       (* Entrada digital - Térmico motores Compuerta Rendija *)
                    WDef:=NOT G_EMG ,               (* Programa seguridad HW - Señal Emergencia intalación *)
                    SRea:=G_Rea,                    (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                    WAut:= ,
          K=>CR_K );                                (* Salida digital - Marcha alimentación 380V Variadores Compuertas Rendijas *)

(* Llamada a FB de gestión de movimiento Windshield *)
Windshield(         Q:=WS_Q ,                       (* Entrada digital - Térmico motores Windshield *)
                    WDef:=NOT G_EMG ,               (* Programa seguridad HW - Señal Emergencia intalación *)
                    SRea:=G_Rea,                    (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                    WAut:= ,
          K=>WS_K );                                (* Salida digital - Marcha alimentación 380V Variadores Windshield *)

(* Llamada a FB de gestión de Resistencias Calefactoras *)
Resistencias(       Q:=RC_Q ,                       (* Entrada digital - Térmico alimentación Resistencias Calefactoras *)
                    TM01:=RC_TM01,                  (* Entrada digital - Termostato Azimut y Compuerta parte fija *)
                    TM02:=RC_TM02 ,                 (* Entrada digital - Termostato Compuerta superior *)
                    TM03:=RC_TM03 ,                 (* Entrada digital - Termostato Compuerta inferior *)
                    WDef:=NOT G_EMG ,               (* Programa seguridad HW - Señal Emergencia instalación *)
                    SRea:=G_Rea ,                   (* Programa seguridad HW - Señal Rearme Emergencia instalación *)
                    WAut:= ,
          K=>RC_K );                                (* Salida digital - Marcha alimentación 380V Resistencias Calefactoras *)

(* Llamada a FB de interconexión Profibus de variables "Generales" desde PLC2 a PLC1  *)
GEN_PLC2_PLC1(      ReaPLC2:=Generales.KRea_Remoto ,
          PLC1_Gen_Sta1=>PLC1_X.GEN.Gen_Sta1 ,
          PLC1_Gen_Sta2=>PLC1_X.GEN.Gen_Sta2 );
```

```
(* Llamada a FB de interconexión Profibus de variables "Servo Azimut" desde PLC2 a PLC1 *)
AZ_PLC2_PLC1(       EjePreparado:=MAIN.Azimutal.AZM01.Power.Status AND MAIN.Azimutal.AZM02.Power.Status ,
                                                EjePosHome:= ,
                                                EjeActivo:=(MAIN.Azimutal.Paso=20) ,
                                                EjeTarget:=(MAIN.Azimutal.Paso=21) ,
                                                EjeOk:= ,
                                                EjePosAbierto:= ,
                                                EjePosCerrado:= ,
                                                EjeFallo1:= ,
                                                EjeFallo2:= ,
                                                EjeFallo3:= ,
                                                EjeFallo4:= ,
                                                ActPos:=MAIN.Azimutal.Vahle.PosAct ,
                                                ActConsumo:= ,
        PLC1_Axis_Sta=>PLC1_X.AZ.Axis_Sta ,
        PLC1_Pos_Mes=>PLC1_X.AZ.Pos_Mes ,
        PLC1_Cur_Mes=>PLC1_X.AZ.Cur_Mes );
```

8.2.15 Estructuras de datos

```
TYPE Drive_AT :
STRUCT
        DiagCode: DWORD; (* Número de diagnóstico S-0-0390 *)
        VelocityFeedbackValue1: DINT; (* Valor 1 feedback de velocidad S-0-0040 *)
        TorqueFeedbackValue: INT; (* Valor feedback de par % - S-0-0084 *)
        ActualMotorCurrentU: INT; (* Valor actual corriente motor fase U % - P-0-0456 *)
        ActualMotorCurrentV: INT; (* Valor actual corriente motor fase V % - P-0-0457 *)
        ActualMotorCurrentW: INT; (* Valor actual corriente motor fase W % - P-0-0458 *)
        ActualTorqueGeneratingCurrent: INT; (* Valor actual par corriente generada % - P-0-0459 *)
        ActualFluxGeneratingCurrent: INT; (* Valor actual corriente de flujo % - P-0-0460 *)
        ActualAbsoluteCurrent: INT; (* Valor actual corriente absoluta % - P-0-0461 *)
        ActualPeakCurrentLimit: INT; (* Valor actual límite corriente de pico % - P-0-0462 *)
        ActualCurrentLimit: INT; (* Valor actual límite corriente % - P-0-0463 *)
        FeedbackValue2: DINT; (* Valor 2 feedback de posición encoder externo *)
END_STRUCT
END_TYPE
```

```
TYPE Drive_Encoder :
STRUCT
        Offset: LREAL; (* Offset valor *)
        PosAct: LREAL; (* Posición actual vueltas tambor inferior *)
        PosAct_mm: LREAL; (* Posición actual en desarrollo lineal mm *)
        S_Offset: BOOL; (* Pulsador orden offset *)
        K_Escala: LREAL; (* Cte de escalado *)
END_STRUCT
END_TYPE
```

```
TYPE Drive_MDT :
STRUCT
        ParMax: UINT := 1000; (* Par máximo ‰ *)
        ParControl: INT := 0; (* Control de Par *)
        ControlWord: BYTE; (* Byte alto vinculado con la Master Control Word del AX5000 *)
END_STRUCT
END_TYPE
```

```
TYPE Gen_D :
STRUCT
        TM01: BOOL; (* Defecto termostato ventilador armario *)
END_STRUCT
END_TYPE
```

```
TYPE Gen_S :
STRUCT
        Man_Aut: BOOL; (* Cambiar modo Manual/Automático *)
        ReaWeb: BOOL; (* Pulsador rearme general instalación - web *)
        HabilitarKV: BOOL; (* Habilitar relé alimentación 220VAC ventilador armario *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Gen_Comandos :
STRUCT
        Gen_Cmd1: WORD; (* Generales - Comandos binarios 1 *)
        Gen_Cmd2: WORD; (* Generales - Comandos binarios 2 *)
        Res_Word03: INT; (* Reserva *)
        Res_Word04: INT; (* Reserva *)
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Gen_Estados :
STRUCT
        Gen_Sta1: WORD; (* Generales - Bits de Estados 1 *)
        Gen_Sta2: WORD; (* Generales - Bits de Estados 2 *)
        Res_Word03: INT; (* Reserva *)
        Res_Word04: INT; (* Reserva *)
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_PLC1_PLC2 :
STRUCT
        GEN: Profibus_Gen_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Generales *)
        AZ: Profibus_Servo_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Azimutal *)
        CR: Profibus_Servo_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Compuerta Rendija *)
        WS: Profibus_Servo_Comandos; (* Interconexión Profibus (8 Word): Comandos PLC1 -> PLC2 Windshield *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_PLC2_PLC1 :
STRUCT
        GEN: Profibus_Gen_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Generales *)
        AZ: Profibus_Servo_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Azimutal *)
        CR: Profibus_Servo_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Compuerta Rendija *)
        WS: Profibus_Servo_Estados; (* Interconexión Profibus (8 Word): Comandos PLC2 -> PLC1 Windshield *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Servo_Comandos :
STRUCT
        Axis_Cmd: WORD; (* Axis bits commands *)
        PosSP_Par: UINT; (* Axis Position SetPoint (1/100 deg) *)
        JogSpd_Par: INT; (* Speed sepoint for jog move (1/100 deg) *)
        Res_Word04: INT; (* Reserva *)
```

```
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

```
TYPE Profibus_Servo_Estados :
STRUCT
        Axis_Sta: WORD; (* Axis bits status *)
        Pos_Mes: UINT; (* Axis actual position (1/100 deg) *)
        Cur_Mes: INT; (* Actual current (1/10 A) *)
        Res_Word04: INT; (* Reserva *)
        Res_Word05: INT; (* Reserva *)
        Res_Word06: INT; (* Reserva *)
        Res_Word07: INT; (* Reserva *)
        Res_Word08: INT; (* Reserva *)
END_STRUCT
END_TYPE
```

```
TYPE Res_D :
STRUCT
        Q: BOOL; (* Defecto térmico alimentación cable calefactor *)
        TM01: BOOL; (* Defecto termostato Azimut y Compuerta fija *)
        TM02: BOOL; (* Defecto termostato Compuerta superiror *)
        TM03: BOOL; (* Defecto termostato Compuerta inferior *)
END_STRUCT
END_TYPE
```

```
TYPE Res_S :
STRUCT
        HabilitarK: BOOL; (* Habilitar contactor alimentación 380VAc *)
END_STRUCT
END_TYPE
```

```
TYPE Servo :
STRUCT
        Eje: AXIS_REF; (* Parámetros Eje AZM02 Plc <-> Nc *)
        ResetServo: FB_SoEReset; (* Reset Servomotor *)
        ResetNC: MC_Reset; (* Reset NC *)
        Power: MC_Power; (* Habilitar potencia Servomotor *)
        ParoServo: MC_Stop; (* Paro Servomotor *)
        Jog: MC_Jog; (* Mover en Jog *)
        MovVel: MC_MoveVelocity; (* Mover en velocidad Servomotor *)
        MovPos: MC_MoveAbsolute; (* Mover en posición absoluta Servomotor *)
        MovRel: MC_MoveRelative; (* Mover en posición relativa Servomotor *)
        SP_Vel: LREAL; (* Setpoint de velocidad *)
        SP_PosActual: LREAL; (* Setpoint de posición actual *)
        SetPos: MC_SetPosition; (* Ajustar posición actual *)
        SP_PosTarget: LREAL; (* Setpoint de posición objetivo *)
```

```
        Acople: MC_GearInDyn; (* Acoplar dos ejes *)
        Acople1: MC_GearInDyn; (* Acoplar dos ejes *)
        Desacople: MC_GearOut; (* Desacoplar dos ejes *)
        RatioAcople: REAL := 1; (* Ratio de acople entre dos ejes *)
        RatioAcopleAnt: REAL := 1; (* Ratio de acople entre dos ejes anterior *)
        ModoAcople: Servo_ModoAcople; (* Modo de acople Servomotor *)
        DiagCode: Servo_DiagCode; (* Código de Diagnóstico en texto para web *)
        Intensidad_A: FB23_Escalado_0_1000; (* Intensidad actual [A] *)
        Par_Nm: FB23_Escalado_0_1000; (* Par [Nm] *)
        Velocidad_rpm: FB23_Escalado_0_1000; (* Velocidad [rpm] *)
        SuperImp: MC_MoveSuperImposed; (* Mover en posición con opción superimposed Servomotor *)
        MovPos1: MC_MoveAbsolute; (* Mover en posición absoluta Servomotor. Introducir nueva posición destino *)
END_STRUCT
END_TYPE
```

```
TYPE Servo_D :
STRUCT
        Q: BOOL; (* Defecto térmico alimentación variadores *)
        Acople: BOOL; (* Defecto acople *)
        DefPos: BOOL; (* Defecto máxima diferencia posición entre Servos *)
        Vel: BOOL; (* Defecto control velocidad *)
        Hab: BOOL; (* Defecto pérdida de habilitación en algún motor *)
        EMG: BOOL; (* Defecto por parada seta de emergencia *)
END_STRUCT
END_TYPE
```

```
TYPE Servo_DiagCode :
(FD43_PowerSupplyFailure:=64835,        (* 16#fd43 *)
ED43_UndervoltageDCLink:=60739,         (* 16#ed43 *)
D011_ControlSectionReady:=53265,        (* 16#d011 *)
D012_DriveReady:=53266,                 (* 16#d012 *)
D013_AxisOP:=53267,                     (* 16#d013 *)
D014_AxisHalt:=53268);                  (* 16#d014 *)
END_TYPE
END_TYPE
```

```
TYPE Servo_K :
STRUCT
        DefPos: INT := 10; (* Diferencia posición entre Servos máxima para defecto *)
        DifPos_Ext: INT := 100; (* Diferencia posición Servos con Encoder Externo *)
        PosCerrado: DINT; (* Posición de cerrado *)
        PosAbierto: DINT; (* Posición de abierto *)
        ParMaxCerrar: UINT; (* Par Límite máximo de cerrar *)
        ParMaxAbrir: UINT; (* Par Límite máximo de abrir *)
        ParControlCerrar: INT; (* Par Control de cerrar *)
        ParControlAbrir: INT; (* Par Control de abrir *)
        RecorridoLineal: DINT; (* Recorrido total lineal *)
        Ri_0: REAL; (* Radio tambor inferior primitivo *)
        Ri_1: REAL; (* Radio tambor inferior 1 vuelta *)
```

```
        Ri_2: REAL; (* Radio tambor inferior 2 vueltas *)
        Ri_3: REAL; (* Radio tambor inferior 3 vueltas *)
        Ri_4: REAL; (* Radio tambor inferior 4 vueltas *)
        Rs_0: REAL; (* Radio tambor superior primitivo *)
        CorreccionSI: LREAL; (* Distancia de corrección SuperImposed *)
        LongitudSI: LREAL; (* Longitud SuperImposed *)
        RecorridoAjusteAbierta: DINT; (* Recorrido ajuste abierta en cº de vueltas de tambor *)
        RecorridoAjusteCerrada: DINT; (* Recorrido ajuste cerrada en cº de vueltas de tambor *)
        VelTrabajo: LREAL; (* Velocidad trabajo cº/s *)
        VelAjuste: LREAL; (* Velocidad de ajuste para tensar/destensar sirgas cº/s *)
        PosDestino: DINT; (* Posición Destino *)
        PosDestino_cg_Compuerta: LREAL; (* Posición Destino *)
        ParMinimo: INT; (* Par mínimo para aplicar corrección para tensar sirga *)
        CRM01_ParVacio: INT; (* Par de vacío CRM01 *)
        CRM02_ParVacio: INT; (* Par de vacío CRM02 *)
        WSM01_ParVacio: INT; (* Par de vacío WSM01 *)
        WSM02_ParVacio: INT; (* Par de vacío WSM02 *)
END_STRUCT
END_TYPE
```

```
TYPE Servo_ModoAcople :
(EjeNoAcoplado,
EjeAcopladoMaestro,
EjeAcopladoMaestroEsclavo,
EjeAcopladoEsclavo);
END_TYPE
END_TYPE
```

```
TYPE Servo_S :
STRUCT
        Parar: BOOL; (* Parar servomotor *)
        JogF: BOOL; (* Mover manual sentido positivo *)
        JogB: BOOL; (* Mover manual sentido negativo *)
        JogF1: BOOL; (* Mover manual sentido positivo *)
        JogB1: BOOL; (* Mover manual sentido negativo *)
        Habilitar: BOOL; (* Habilitar potencia variador *)
        MovVel: BOOL; (* Mover en velocidad *)
        MovPos: BOOL; (* Mover en posición absoluta *)
        MovPos1: BOOL; (* Mover en posición absoluta *)
        MovRel: BOOL; (* Mover en posición relativa *)
        PosActual: BOOL; (* Ajustar posición actual *)
        PosActual1: BOOL; (* Ajustar posición actual *)
        Acoplar: BOOL; (* Acoplar ejes *)
        Acoplar1: BOOL; (* Acoplar ejes *)
        Desacoplar: BOOL; (* Desacoplar ejes *)
        Man_Aut: BOOL; (* Cambiar modo Manual/Automático *)
        HabilitarK: BOOL; (* Habilitar contactor alimentación 380VAc *)
        Inicio: BOOL; (* Inicio servomotores para mover *)
```

```
        Fin: BOOL; (* Fin servomotores de mover *)
        Mover: BOOL; (* Orden de mover servomotores *)
        Cerrar: BOOL; (* Orden de cerrar *)
        Abrir: BOOL; (* Orden de abrir *)
        Velo_Par: BOOL; (* Cambiar entre modo Velocidad = '0' y modo Par = '1' *)
        MovSuper: BOOL; (* Mover en posición con SuperImposed *)
        MovSuper1: BOOL; (* Mover en posición con SuperImposed *)
        Reset: BOOL; (* Ejecutar Reset NC y Drive *)
        MovPosNew: BOOL; (* Mover en posición absoluta - nuevo destino *)
END_STRUCT
END_TYPE
```

```
TYPE Sick :
STRUCT
        Offset: DINT; (* Offset valor *)
        PosAct: REAL; (* Posición actual vueltas tambor inferior *)
        PosAct_mm: REAL; (* Posición actual en desarrollo lineal mm *)
        S_Offset: BOOL; (* Pulsador orden offset *)
        K_Escala: REAL := 1048576; (* Cte de escalado (32768*32) *)
END_STRUCT
END_TYPE
```

```
TYPE Vahle :
STRUCT
        PosAct_mm: LREAL; (* Posción actual encoder Vahle SSI (mm) *)
        PosAct: LREAL; (* Posción actual encoder Vahle SSI (centésima de grado) *)
        PosAnt: LREAL; (* Posción ciclo anterior encoder Vahle SSI (centésima de grado) *)
        PosLMax_mm: LREAL; (* Longitud máxima banda encoder Vahle (mm) *)
        PosLMin_mm: LREAL; (* Longitud mínima banda encoder Vahle (mm) *)
        TiempoFallo: INT; (* Tiempo para Fallo lectura encoder Vahle *)
        FalloLectura: BOOL; (* Fallo lectura encoder Vahle *)
END_STRUCT
END_TYPE
```

8.2.16 Variables globales

```
VAR_GLOBAL
	CR_Q AT %I* : BOOL; (* Compuerta Rendija - Térmico alimentación variadores - EL1809 *)
	WS_Q AT %I* : BOOL; (* Windshield - Térmico alimentación variadores - EL1809 *)
	AZ_Q AT %I* : BOOL; (* Azimutal - Térmico alimentación variadores - EL1809 *)
	RC_Q AT %I* : BOOL; (* Resistencia Calefactora - Térmico alimentación resistencias - EL1809 *)
	CR_FC01 AT %I* : BOOL; (* Compuerta Rendija - Final de carrera sobrerrecorido abierta - EL1809 *)
	CR_FC02 AT %I* : BOOL; (* Compuerta Rendija - Final de carrera sobrerrecorido cerrada - EL1809 *)
	WS_FC01 AT %I* : BOOL; (* Windshield - Final de carrera sobrerrecorrido abajo - EL1809 *)
	WS_FC02 AT %I* : BOOL; (* Windshield - Final de carrera sobrerrecorrido arriba - EL1809 *)
	RC_TM01 AT %I* : BOOL; (* Resistencia Calefactora - Termostato Azimut y Compuerta fija - EL1809 *)
	RC_TM02 AT %I* : BOOL; (* Resistencia Calefactora - Termostato Compuerta superiror - EL1809 *)
	RC_TM03 AT %I* : BOOL; (* Resistencia Calefactora - Termostato Compuerta inferior - EL1809 *)
	G_SH01 AT %I* : BOOL; (* General - Pulsador rearme emergencia - EL1809 *)
	G_TM01 AT %I* : BOOL; (* General - Termostato armario general - EL1809 *)
	CR_K AT %Q* : BOOL; (* Compuerta Rendija - Marcha contactor alimentación variadores - EL2809 *)
	WS_K AT %Q* : BOOL; (* Windshield - Marcha contactor alimentación variadores - EL2809 *)
	AZ_K AT %Q* : BOOL; (* Azimutal - Marcha contactor alimentación variadores - EL2809 *)
	RC_K AT %Q* : BOOL; (* Resitencia Calefactora - Marcha contactor alimentación resistencias - EL2809 *)
	G_H01 AT %Q* : BOOL; (* General - Piloto rearme emergencia - EL2809 *)
	G_KV AT %Q* : BOOL; (* General - Marcha relé ventilador armario - EL2809 *)
	UNO: BOOL; (* Variable siempre a '1' *)
	CERO: BOOL; (* Variable siempre a '0' *)
	G_Rea AT %Q* : BOOL; (* General - Rearme general instalación - EL6900 *)
	G_EMG AT %I* : BOOL; (* General - Emergencia Instalación - EL6900 *)
	AZ_PosVahle AT %I* : DWORD; (* Azimutal - Posición Encoder Vahle SSI - EL5001 *)
	AZM01_AT AT %I* : Drive_AT; (* Azimutal 1 - AT Drive 11 - AX5112 *)
	AZM02_AT AT %I* : Drive_AT; (* Azimutal 2 - AT Drive 12 - AX5112 *)
	CRM01_AT AT %I* : Drive_AT; (* Compuerta cierre - AT Drive 13 - AX5125 *)
	CRM02_AT AT %I* : Drive_AT; (* Compuerta apertura - AT Drive 14 - AX5112 *)
	WSM01_AT AT %I* : Drive_AT; (* Windshield inferior - AT Drive 14 - AX5125 *)
	WSM02_AT AT %I* : Drive_AT; (* Windshield superior 1 - AT Drive 16 - AX5112 *)
	WSM03_AT AT %I* : Drive_AT; (* Windshield superior 2 - AT Drive 17 - AX5112 *)
	WSM01_ControlWord_NCtoPLC AT %I* : BYTE; (* Windshield inferior - Byte alto vincualdo con nCtrl1 del eje de NC *)
	AZM01_MDT AT %Q* : Drive_MDT; (* Azimutal 1 - MDT Drive 11 - AX5112 *)
	AZM02_MDT AT %Q* : Drive_MDT; (* Azimutal 2 - MDT Drive 12 - AX5112 *)
	CRM01_MDT AT %Q* : Drive_MDT; (* Compuerta cierre - MDT Drive 13 - AX5125 *)
	CRM02_MDT AT %Q* : Drive_MDT; (* Compuerta apertura - MDT Drive 14 - AX5112 *)
	WSM01_MDT AT %Q* : Drive_MDT; (* Windshield inferior - MDT Drive 15 - AX5118 *)
	WSM02_MDT AT %Q* : Drive_MDT; (* Windshield superior 1 - MDT Drive 16 - AX5112 *)
	WSM03_MDT AT %Q* : Drive_MDT; (* Windshield superior 2 - MDT Drive 17 - AX5112 *)
	PLC1_X AT %Q* : Profibus_PLC2_PLC1; (* Interconexión Profibus PLC2 -> PLC1 Estados (32 Word) *)
	PLC1_W AT %I* : Profibus_PLC1_PLC2; (* Interconexión Profibus PLC1 -> PLC2 Comandos (32 Word) *)
END_VAR
```