```
VAR_GLOBAL
DI_ACK_M1 AT %IX0.0 : BOOL; (* Confirmació Motor M1 *)
DI_ACK_M2 AT %IX0.1 : BOOL; (* Confirmació Motor M2 *)
DI_ACK_M3 AT %IX0.2 : BOOL; (* Confirmació Motor M3 *)
DI_ACK_M4 AT %IX0.3 : BOOL; (* Confirmació Motor M4 *)
DI_ACK_M5 AT %IX0.4 : BOOL; (* Confirmació Motor M5 *)
DI_ACK_M6 AT %IX0.5 : BOOL; (* Confirmació Motor M6 *)
DI_ACK_M7 AT %IX0.6 : BOOL; (* Confirmació Motor M7 *)
DI_ACK_M8 AT %IX0.7 : BOOL; (* Confirmació Motor M8 *)
DI_ACK_M9 AT %IX0.8 : BOOL; (* Confirmació Motor M9 *)
DI_ACK_M10 AT %IX0.9 : BOOL; (* Confirmació Motor M10 *)
DI_ACK_M11 AT %IX0.10 : BOOL; (* Confirmació Motor M11 *)
DI_ACK_M12 AT %IX0.11 : BOOL; (* Confirmació Motor M12 *)
DI_ACK_M13 AT %IX0.12 : BOOL; (* Confirmació Motor M13 *)
DI_ACK_M14 AT %IX0.13 : BOOL; (* Confirmació Motor M14 *)
DI_ACK_M15 AT %IX0.14 : BOOL; (* Confirmació Motor M15 *)
DI_ACK_M16 AT %IX0.15 : BOOL; (* Confirmació Motor M16 *)
DI_ACK_M17 AT %IX0.16 : BOOL; (* Confirmació Motor M17 *)
DI_ACK_M18 AT %IX0.17 : BOOL; (* Confirmació Motor M18 *)
DI_ACK_M19 AT %IX0.18 : BOOL; (* Confirmació Motor M19 *)
DI_ACK_REVOLVER AT %IX0.19 : BOOL; (* Confirmació Motor REVOLVER *)
DI_PARADA_EMERGENCIA AT %IX1.0 : BOOL; (* Pulsador de Parada emergència *)
DI_SS1 AT %IX1.1 : BOOL; (* Sensor presència producte Sitja S1 *)
DI_SS2 AT %IX1.2 : BOOL; (* Sensor presència producte Sitja S2 *)
DI_SS3 AT %IX1.3 : BOOL; (* Sensor presència producte Sitja S3 *)
DI_SS4 AT %IX1.4 : BOOL; (* Sensor presència producte Sitja S4 *)
DI_SS5 AT %IX1.5 : BOOL; (* Sensor presència producte Sitja S5 *)
DI_SS6 AT %IX1.6 : BOOL; (* Sensor presència producte Sitja S6 *)
DI_SS7 AT %IX1.7 : BOOL; (* Sensor presència producte Sitja S7 *)
DI_SSA AT %IX1.8 : BOOL; (* Sensor presència producte Tramuja A *)
DI_SSB AT %IX1.9 : BOOL; (* Sensor presència producte Tramuja B *)
DI_SSM1 AT %IX1.10 : BOOL; (* Sensor presència producte entrada Mescladora *)
DI_SSM2 AT %IX1.11 : BOOL; (* Sensor presència producte Mescladora *)
DI_SSM3 AT %IX1.12 : BOOL; (* Sensor presència producte sortida Mescladora *)
DI_FC1 AT %IX1.13 : BOOL; (* Final de cursa Revolver per omplir Sitja S1 *)
DI_FC2 AT %IX1.14 : BOOL; (* Final de cursa Revolver per omplir Sitja S2 *)
DI_FC3 AT %IX1.15 : BOOL; (* Final de cursa Revolver per omplir Sitja S3 *)
DI_FC4 AT %IX1.16 : BOOL; (* Final de cursa Revolver per omplir Sitja S4 *)
DI_FC5 AT %IX1.17 : BOOL; (* Final de cursa per omplir Sitja S5 *)
DI_FC6 AT %IX1.18 : BOOL; (* Final de cursa per omplir Sitja S6 *)
DI_FC7 AT %IX1.19 : BOOL; (* Final de cursa per omplir Sitja S7 *)
DI_S0 AT %IX1.20 : BOOL; (* Sensor presència producte S0 *)
DI_S1 AT %IX1.21 : BOOL; (* Sensor presència producte S1 *)
DI_S2 AT %IX1.22 : BOOL; (* Sensor presència producte S2 *)
DI_S3 AT %IX1.23 : BOOL; (* Sensor presència producte S3 *)
DI_S4 AT %IX1.24 : BOOL; (* Sensor presència producte S4 *)
DO_EV0 AT %QX0.0 : BOOL; (* Electrovàlvula EV0 *)
DO_EV1 AT %QX0.1 : BOOL; (* Electrovàlvula EV1 *)
DO_EV2 AT %QX0.2 : BOOL; (* Electrovàlvula EV2 *)
DO_EV3 AT %QX0.3 : BOOL; (* Electrovàlvula EV3 *)
DO_EV4 AT %QX0.4 : BOOL; (* Electrovàlvula EV4 *)
DO_EV5 AT %QX0.5 : BOOL; (* Electrovàlvula EV5 *)
DO_EV6 AT %QX0.6 : BOOL; (* Electrovàlvula EV6 *)
DO_EV7 AT %QX0.7 : BOOL; (* Electrovàlvula EV7 *)
DO_EV8 AT %QX0.8 : BOOL; (* Electrovàlvula EV8 *)
DO_EV9 AT %QX0.9 : BOOL; (* Electrovàlvula EV9 *)
DO_EV10 AT %QX0.10 : BOOL; (* Electrovàlvula EV10 *)
DO_EV11 AT %QX0.11 : BOOL; (* Electrovàlvula EV11 *)
DO_EV12 AT %QX0.12 : BOOL; (* Electrovàlvula EV12 *)
DO_H0 AT %QX0.13 : BOOL; (* Senyal lluminosa entrada pel camió H1 *)
DO_M1 AT %QX0.14 : BOOL; (* Motor M1 *)
DO_M2 AT %QX0.15 : BOOL; (* Motor M2 *)
DO_M3 AT %QX0.16 : BOOL; (* Motor M3 *)
DO_M4 AT %QX0.17 : BOOL; (* Motor M4 *)
DO_M5 AT %QX0.18 : BOOL; (* Motor M5 *)
DO_M6 AT %QX0.19 : BOOL; (* Motor M6 *)
DO_M7 AT %QX0.20 : BOOL; (* Motor M7 *)
DO_M8 AT %QX0.21 : BOOL; (* Motor M8 *)
DO_M9 AT %QX0.22 : BOOL; (* Motor M9 *)
DO_M10 AT %QX0.23 : BOOL; (* Motor M10 *)
DO_M11 AT %QX1.0 : BOOL; (* Motor M11 *)
DO_M12 AT %QX1.1 : BOOL; (* Motor M12 *)
DO_M13 AT %QX1.2 : BOOL; (* Motor M13 *)
DO_M14 AT %QX1.3 : BOOL; (* Motor M14 *)
DO_M15 AT %QX1.4 : BOOL; (* Motor M15 *)
DO_M16 AT %QX1.5 : BOOL; (* Motor M16 *)
DO_M17 AT %QX1.6 : BOOL; (* Motor M17 *)
DO_M18 AT %QX1.7 : BOOL; (* Motor M18 *)
DO_M19 AT %QX1.8 : BOOL; (* Motor M19 *)
DO_REVOLVER AT %QX1.9 : BOOL; (* Motor REVOLVER *)
S0 AT %MW100.0 : WORD;
S1 AT %MW100.2 : WORD;
IM1 AT %MW100.4 : WORD;
M1 AT %MW100.6 : WORD;
M2 AT %MW100.8 : WORD;
M3 AT %MW100.10 : WORD;
M4 AT %MW100.12 : WORD;
M5 AT %MW100.14 : WORD;
M6 AT %MW100.16 : WORD;
M7 AT %MW100.18 : WORD;
M8 AT %MW100.20 : WORD;
M9 AT %MW100.22 : WORD;
M10 AT %MW100.24 : WORD;
FC1 AT %MW100.28 : WORD;
FC2 AT %MW100.30 : WORD;
FC3 AT %MW100.32 : WORD;
FC4 AT %MW100.34 : WORD;
H0 AT %MW100.36 : WORD;
SS1 AT %MW100.38 : WORD;
SS2 AT %MW100.40 : WORD;
SS3 AT %MW100.42 : WORD;
SS4 AT %MW100.44 : WORD;
LST1_PRC AT %MW100.48 : WORD;
LST1_VAL AT %MW100.52 : WORD;
LST2_PRC AT %MW100.56 : WORD;
LST2_VAL AT %MW100.60 : WORD;
LST3_PRC AT %MW100.64 : WORD;
LST3_VAL AT %MW100.68 : WORD;
LST4_PRC AT %MW100.72 : WORD;
LST4_VAL AT %MW100.76 : WORD;
PRODUCTE_SELECTED AT %MW100.78 : WORD;
QUANT_PROD_SITJA_S1 AT %MW100.88 : WORD;
QUANT_PROD_SITJA_S2 AT %MW100.90 : WORD;
QUANT_PROD_SITJA_S3 AT %MW100.92 : WORD;
QUANT_PROD_SITJA_S4 AT %MW100.94 : WORD;
QUANT_RESTA_SITJA_S1 AT %MW100.96 : WORD;
QUANT_RESTA_SITJA_S2 AT %MW100.98 : WORD;
QUANT_RESTA_SITJA_S3 AT %MW100.100 : WORD;
QUANT_RESTA_SITJA_S4 AT %MW100.102 : WORD;
ESTAT_SITJA_S1 AT %MW100.104 : WORD;
ESTAT_SITJA_S2 AT %MW100.106 : WORD;
ESTAT_SITJA_S3 AT %MW100.108 : WORD;
ESTAT_SITJA_S4 AT %MW100.110 : WORD;
S2 AT %MW100.112 : WORD;
S3 AT %MW100.114 : WORD;
S4 AT %MW100.116 : WORD;
M11 AT %MW100.118 : WORD;
M12 AT %MW100.120 : WORD;
M13 AT %MW100.122 : WORD;
M14 AT %MW100.124 : WORD;
M15 AT %MW100.126 : WORD;
M16 AT %MW100.128 : WORD;
M17 AT %MW100.130 : WORD;
M18 AT %MW100.132 : WORD;
M19 AT %MW100.134 : WORD;
PAS_CARREGAR AT %MW100.136 : UINT;
```

```
PAS_FABRICAR AT %MW100.138 : UINT;
SSA AT %MW100.140 : WORD;
SSB AT %MW100.142 : WORD;
SSM1 AT %MW100.144 : WORD;
SSM2 AT %MW100.146 : WORD;
SSM3 AT %MW100.150 : WORD;
IM2 AT %MW100.152 : WORD;
IM3 AT %MW100.154 : WORD;
IM4 AT %MW100.156 : WORD;
EV1 AT %MW100.158 : WORD;
EV2 AT %MW100.160 : WORD;
EV3 AT %MW100.162 : WORD;
EV4 AT %MW100.164 : WORD;
EV5 AT %MW100.166 : WORD;
EV6 AT %MW100.168 : WORD;
BTA AT %MW100.170 : WORD;
BTB AT %MW100.172 : WORD;
FC5 AT %MW100.174 : WORD;
FC6 AT %MW100.176 : WORD;
FC7 AT %MW100.178 : WORD;
LST5_PRC AT %MW100.182 : WORD;
LST5_VAL AT %MW100.186 : WORD;
LST6_PRC AT %MW100.190 : WORD;
LST6_VAL AT %MW100.194 : WORD;
LST7_PRC AT %MW100.198 : WORD;
LST7_VAL AT %MW100.202 : WORD;
SS5 AT %MW100.204 : WORD;
SS6 AT %MW100.206 : WORD;
SS7 AT %MW100.208 : WORD;
EV7 AT %MW100.212 : WORD;
EV8 AT %MW100.214 : WORD;
EV9 AT %MW100.216 : WORD;
EV10 AT %MW100.218 : WORD;
EV11 AT %MW100.220 : WORD;
EV12 AT %MW100.222 : WORD;
QUANT_PROD_SITJA_S5 AT %MW100.236 : WORD;
QUANT_PROD_SITJA_S6 AT %MW100.238 : WORD;
QUANT_PROD_SITJA_S7 AT %MW100.240 : WORD;
QUANT_RESTA_SITJA_S5 AT %MW100.242 : WORD;
QUANT_RESTA_SITJA_S6 AT %MW100.244 : WORD;
QUANT_RESTA_SITJA_S7 AT %MW100.246 : WORD;
ESTAT_SITJA_S5 AT %MW100.248 : WORD;
ESTAT_SITJA_S6 AT %MW100.250 : WORD;
ESTAT_SITJA_S7 AT %MW100.252 : WORD;
PARADA_EMERGENCIA AT %MW100.254 : WORD;
CONFIG_CARREGAR_ONOFF AT %MW100.260 : WORD;
CONFIG_FABRICAR_ONOFF AT %MW100.280 : WORD;
CONFIG_EXPEDIR_ONOFF AT %MW100.288 : WORD;
REVOLVER AT %MW100.292 : WORD;
QUANT_BLAT_MORO_SITJES AT %MW100.294 : WORD;
QUANT_BLAT_SITJES AT %MW100.296 : WORD;
QUANT_ORDI_SITJES AT %MW100.298 : WORD;
QUANT_FARINA_SITJES AT %MW100.300 : WORD;
EV0 AT %MW100.336 : WORD;
PAS_GUARDAR AT %MW100.344 : UINT;
PAS_EXPEDIR AT %MW100.346 : UINT;
ACK_M1 AT %MW100.348 : WORD;
ACK_M2 AT %MW100.350 : WORD;
ACK_M3 AT %MW100.352 : WORD;
ACK_M4 AT %MW100.354 : WORD;
ACK_M5 AT %MW100.356 : WORD;
ACK_M6 AT %MW100.358 : WORD;
ACK_M7 AT %MW100.360 : WORD;
ACK_M8 AT %MW100.362 : WORD;
ACK_M9 AT %MW100.364 : WORD;
ACK_M10 AT %MW100.366 : WORD;
ACK_M11 AT %MW100.368 : WORD;
ACK_M12 AT %MW100.370 : WORD;
ACK_M13 AT %MW100.372 : WORD;
ACK_M14 AT %MW100.374 : WORD;
ACK_M15 AT %MW100.376 : WORD;
ACK_M16 AT %MW100.378 : WORD;
ACK_M17 AT %MW100.380 : WORD;
ACK_M18 AT %MW100.382 : WORD;
ACK_M19 AT %MW100.384 : WORD;
ACK_REVOLVER AT %MW100.386 : WORD;
ALARMES_MOTORS1 AT %MW100.388 : WORD;
ALARMES_MOTORS2 AT %MW100.390 : WORD;
ALARMES_CARREGAR AT %MW100.392 : WORD;
ALARMES_FABRICAR AT %MW100.394 : WORD;
ALARMES_GUARDAR AT %MW100.396 : WORD;
ALARMES_EXPEDIR AT %MW100.398 : WORD;
PC_COMANDA_1 AT %MW100.500 : WORD;
PC_COMANDA_2 AT %MW100.502 : WORD;
PC_COMANDA_3 AT %MW100.504 : WORD;
PC_COMANDA_4 AT %MW100.506 : WORD;
END_VAR


VAR_GLOBAL RETAIN
LST1_MAX AT %MW100.46 : WORD;
LST1_MIN AT %MW100.50 : WORD;
LST2_MAX AT %MW100.54 : WORD;
LST2_MIN AT %MW100.58 : WORD;
LST3_MAX AT %MW100.62 : WORD;
LST3_MIN AT %MW100.66 : WORD;
LST4_MAX AT %MW100.70 : WORD;
LST4_MIN AT %MW100.74 : WORD;
PRODUCTE_SITJA_S1 AT %MW100.80 : WORD;
PRODUCTE_SITJA_S2 AT %MW100.82 : WORD;
PRODUCTE_SITJA_S3 AT %MW100.84 : WORD;
PRODUCTE_SITJA_S4 AT %MW100.86 : WORD;
LST5_MAX AT %MW100.180 : WORD;
LST5_MIN AT %MW100.184 : WORD;
LST6_MAX AT %MW100.188 : WORD;
LST6_MIN AT %MW100.192 : WORD;
LST7_MAX AT %MW100.196 : WORD;
LST7_MIN AT %MW100.200 : WORD;
KG_EXPEDITS_SITJA_S5 AT %MW100.224 : WORD;
KG_EXPEDITS_SITJA_S6 AT %MW100.226 : WORD;
KG_EXPEDITS_SITJA_S7 AT %MW100.228 : WORD;
PRODUCTE_SITJA_S5 AT %MW100.230 : WORD;
PRODUCTE_SITJA_S6 AT %MW100.232 : WORD;
PRODUCTE_SITJA_S7 AT %MW100.234 : WORD;
CONFIG_PROD_CARREGAR AT %MW100.256 : WORD;
CONFIG_QUANT_CARREGAR AT %MW100.258 : WORD;
CONFIG_TIMEOUT_S1 AT %MW100.262 : WORD;
CONFIG_TIMEOUT_M1 AT %MW100.264 : WORD;
CONFIG_TIMEOUT_FC1_A_FC4 AT %MW100.266 : WORD;
CONFIG_TIMEOUT_SS1_A_SS7 AT %MW100.268 : WORD;
CONFIG_TIMEOUT_S2 AT %MW100.270 : WORD;
CONFIG_TIMEOUT_S3 AT %MW100.272 : WORD;
CONFIG_TIMEOUT_S4 AT %MW100.274 : WORD;
CONFIG_PROD_FABRICAR AT %MW100.276 : WORD;
CONFIG_QUANT_FABRICAR AT %MW100.278 : WORD;
CONFIG_RENDIMENT_MOLI AT %MW100.282 : WORD;
CONFIG_PROD_EXPEDIR AT %MW100.284 : WORD;
CONFIG_QUANT_EXPEDIR AT %MW100.286 : WORD;
CONFIG_TIMEOUT_FC5_A_FC7 AT %MW100.290 : WORD;
RECEPTA_BLAT_MORO_PROD1 AT %MW100.302 : WORD;
RECEPTA_BLAT_PROD1 AT %MW100.304 : WORD;
RECEPTA_ORDI_PROD1 AT %MW100.306 : WORD;
RECEPTA_FARINA_PROD1 AT %MW100.308 : WORD;
RECEPTA_BLAT_MORO_PROD2 AT %MW100.310 : WORD;
RECEPTA_BLAT_PROD2 AT %MW100.312 : WORD;
RECEPTA_ORDI_PROD2 AT %MW100.314 : WORD;
```

```
RECEPTA_FARINA_PROD2 AT %MW100.316 : WORD;
RECEPTA_BLAT_MORO_PROD3 AT %MW100.318 : WORD;
RECEPTA_BLAT_PROD3 AT %MW100.320 : WORD;
RECEPTA_ORDI_PROD3 AT %MW100.322 : WORD;
RECEPTA_FARINA_PROD3 AT %MW100.324 : WORD;
DOSIS_MESCLADORA AT %MW100.326 : WORD;
DOSIS_MOLI_BLAT_MORO AT %MW100.328 : WORD;
DOSIS_MOLI_BLAT AT %MW100.330 : WORD;
DOSIS_MOLI_ORDI AT %MW100.332 : WORD;
DOSIS_MOLI_FARINA AT %MW100.334 : WORD;
CONFIG_TEMPS_MOLI AT %MW100.338 : WORD;
CONFIG_TEMPS_MESCLADORA AT %MW100.340 : WORD;
CONFIG_TEMPS_SORTIDA_MESCLA AT %MW100.342 : WORD;
VARIADOR_M11_PRC AT %MW100.400 : WORD;
VARIADOR_M14_PRC AT %MW100.402 : WORD;
END_VAR
```

```
VAR
MARXA : BOOL;
CarregarMateriesPrimeres : CarregarMateriesPrimeres;
Aturar : Aturar;
FabricarProductes : FabricarProductes;
ComprovarMateriesPrimeres : ComprovarMateriesPrimeres;
ExpedirProductes : ExpedirProductes;
GuardarProductes : GuardarProductes;
AlarmesMotors : AlarmesMotors;
Simulador : Simulador;
AturarCarrega : AturarCarrega;
AturarExpedicio : AturarExpedicio;
AturarFabricacio : AturarFabricacio;
AturarProductes : AturarProductes;
ConvertirEntrades : ConvertirEntrades;
ConvertirSortides : ConvertirSortides;
ComprovarSitges : ComprovarSitges;
END_VAR
```

```
 1
 2 (* Miro si tinc la parada d'emergencia activa *)
 3 IF (PARADA_EMERGENCIA > 0) THEN
 4     (* Desactivo la marxa *)
 5     MARXA := FALSE;
 6 ELSE
 7     (* Activo la marxa *)
 8     MARXA := TRUE;
 9 END_IF;
10
11 (* Passo les entrades a marques per treballar *)
12 (* ConvertirEntrades(); *)
13
14 (* Simulador *)
15 Simulador();
16
17 (* Comprovo els nivells de les Sitges *)
18 ComprovarSitges();
19
20 (* Comprovo les alarmes dels motors *)
21 AlarmesMotors();
22
23 (* Miro si estic en marxa *)
24 IF (MARXA) THEN
25
26     (* Carrego Matèries Primeres *)
27     CarregarMateriesPrimeres();
28     (* Comprovo les Matèries Primeres *)
29     ComprovarMateriesPrimeres();
30     (* Miro si NO estic carregant materies primeres *)
31     IF CONFIG_CARREGAR_ONOFF = 0 THEN
32         (* Aturo la carrega *)
33         AturarCarrega();
34     END_IF;
35
36     (* Fabrico els Productes *)
37     FabricarProductes();
38     (* Guardo els Productes *)
39     GuardarProductes();
40     (* Miro si NO estic fabricant productes *)
41     IF CONFIG_FABRICAR_ONOFF = 0 THEN
42         (* Aturo la fabricacio *)
43         AturarFabricacio();
44         (* Aturo els productes *)
45         AturarProductes();
46     END_IF;
47
48     (* Expedeixo Productes *)
49     ExpedirProductes();
50     (* Miro si NO estic expedint productes *)
51     IF CONFIG_EXPEDIR_ONOFF = 0 THEN
52         (* Aturo l'expedicio *)
53         AturarExpedicio();
54     END_IF;
55
56 ELSE
57     (* Parada emergencia *)
58     Aturar();
59 END_IF;
60
61 (* Passo les marques a sortides *)
62 (* ConvertirSortides(); *)
63
64
```

FUNCTION_BLOCK ComprovarSitja

```
VAR_INPUT
LST_MAX : WORD; (* Nivell màxim de la Sitja (percentatge) *)
LST_MIN : WORD; (* Nivell minim de la Sitja (percentatge) *)
LST_VAL : WORD; (* Nivell de la Sitja (valor) *)
QUANTITAT_MAX : UINT;
END_VAR

VAR_OUTPUT
LST_PRC : WORD; (* Nivell de la Sitja (percentatge) *)
ESTAT_SITJA : WORD;
QUANTITAT : UINT;
RESTA : UINT;
END_VAR

VAR
PERCENT : REAL;
SITJA_OK : UINT; (* Nivell de la Sitja està entre límits *)
END_VAR
```

```
(* Calculo el percentatge de nivell de la Sitja *)
PERCENT := TO_REAL(LST_VAL) / 65536;
PERCENT := PERCENT * 100;
LST_PRC := TO_UINT(PERCENT);

(* Calculo la quantitat de material de la Sitja *)
QUANTITAT := TO_UINT((QUANTITAT_MAX * TO_UINT(PERCENT)) / 100);

(* Calculo la quantitat restant de material de la Sitja *)
RESTA := TO_UINT(QUANTITAT_MAX - QUANTITAT);

(* Miro el nivell de la Sitja ha superat el límit maxim *)
IF (LST_MAX < LST_PRC) THEN
    (* Activo error MAX Sitja *)
    ESTAT_SITJA := 1;
    SITJA_OK := 0;
ELSE
    IF (LST_MIN > LST_PRC) THEN
        (* Activo error MIN Sitja *)
        ESTAT_SITJA := 2;
        SITJA_OK := 0;
    ELSE
        (* Sitja OK *)
        ESTAT_SITJA := 0;
        SITJA_OK := 1;
    END_IF;
END_IF;
```

```
VAR_EXTERNAL
ALARMES_CARREGAR : WORD;
CONFIG_CARREGAR_ONOFF : WORD;
CONFIG_PROD_CARREGAR : WORD;
CONFIG_TIMEOUT_FC1_A_FC4 : WORD;
CONFIG_TIMEOUT_M1 : WORD;
CONFIG_TIMEOUT_S1 : WORD;
FC1 : WORD;
FC2 : WORD;
FC3 : WORD;
FC4 : WORD;
LST1_MAX : WORD;
LST1_PRC : WORD;
LST2_MAX : WORD;
LST2_PRC : WORD;
LST3_MAX : WORD;
LST3_PRC : WORD;
LST4_MAX : WORD;
LST4_PRC : WORD;
M1 : WORD;
M2 : WORD;
M3 : WORD;
M4 : WORD;
M5 : WORD;
M6 : WORD;
PAS_CARREGAR : UINT;
PRODUCTE_SELECTED : WORD;
PRODUCTE_SITJA_S1 : WORD;
PRODUCTE_SITJA_S2 : WORD;
PRODUCTE_SITJA_S3 : WORD;
PRODUCTE_SITJA_S4 : WORD;
QUANT_RESTA_SITJA_S1 : WORD;
QUANT_RESTA_SITJA_S2 : WORD;
QUANT_RESTA_SITJA_S3 : WORD;
QUANT_RESTA_SITJA_S4 : WORD;
REVOLVER : WORD;
S0 : WORD;
S1 : WORD;
M7 : WORD;
M8 : WORD;
M9 : WORD;
M10 : WORD;
QUANT_PROD_SITJA_S1 : WORD;
QUANT_PROD_SITJA_S2 : WORD;
QUANT_PROD_SITJA_S3 : WORD;
QUANT_PROD_SITJA_S4 : WORD;
CONFIG_QUANT_CARREGAR : WORD;
H0 : WORD;
END_VAR

VAR
ALARMA_REVOLVER : BOOL;
ALARMA_S1 : BOOL;
AturarCarrega : AturarCarrega;
CarregarSitjaS1 : CarregarSitja;
CarregarSitjaS2 : CarregarSitja;
CarregarSitjaS3 : CarregarSitja;
CarregarSitjaS4 : CarregarSitja;
INSTANT_FC : UDINT;
INSTANT_M1 : UDINT;
INSTANT_S1 : UDINT;
OK_CAPACITAT_SITJA_S1 : BOOL;
OK_CAPACITAT_SITJA_S2 : BOOL;
OK_CAPACITAT_SITJA_S3 : BOOL;
OK_CAPACITAT_SITJA_S4 : BOOL;
OK_PROD_S1 : BOOL;
OK_PROD_S2 : BOOL;
OK_PROD_S3 : BOOL;
OK_PROD_S4 : BOOL;
PRODUCTE_ANTERIOR : WORD;
AturarFabricacio : AturarFabricacio;
NIVELL_SITJA : WORD;
ALARMA_M1 : BOOL;
END_VAR
```

```
 1
 2 (* Mostro per pantalla el producte que he seleccionat *)
 3 PRODUCTE_SELECTED := CONFIG_PROD_CARREGAR;
 4
 5 (* Miro si he canviat el producte seleccionat *)
 6 IF PRODUCTE_SELECTED <> PRODUCTE_ANTERIOR THEN
 7     (* Em guardo el producte seleccionat *)
 8     PRODUCTE_ANTERIOR := PRODUCTE_SELECTED;
 9     (* Reset del pas *)
10     PAS_CARREGAR := 0;
11 END_IF;
12
13 (* Reset dels flags *)
14 OK_PROD_S1 := FALSE;
15 OK_PROD_S2 := FALSE;
16 OK_PROD_S3 := FALSE;
17 OK_PROD_S4 := FALSE;
18
19 (* Miro si la Sitja S1 esta buida o te el mateix producte que esta seleccionat per pantalla *)
20 IF PRODUCTE_SITJA_S1 = 0 OR PRODUCTE_SITJA_S1 = PRODUCTE_SELECTED THEN
21
22     (* Carrego la Sitja S1 *)
23     CarregarSitjaS1.FINAL_CARRERA := FC1;
24     CarregarSitjaS1.QUANT_RESTA_SITJA := QUANT_RESTA_SITJA_S1;
25     CarregarSitjaS1();
26     M4 := CarregarSitjaS1.MOTOR_OUT;
27     M5 := M4;
28     OK_CAPACITAT_SITJA_S1 := CarregarSitjaS1.OK_CAPACITAT;
29     (* Miro si carrego la Sitja S1 *)
30     IF OK_CAPACITAT_SITJA_S1 THEN
31         (* Guardo el producte seleccionat per pantalla com el producte de la Sitja S1 *)
32         PRODUCTE_SITJA_S1 := PRODUCTE_SELECTED;
33     END_IF;
34     (* Activo el flag *)
35     OK_PROD_S1 := TRUE;
36
37 ELSE
38
39     (* Miro si la Sitja S1 no esta buida i (la Sitja S2 esta buida o te el mateix producte que esta seleccionat) *)
40     IF PRODUCTE_SITJA_S1 > 0 AND (PRODUCTE_SITJA_S2 = 0 OR PRODUCTE_SITJA_S2 = PRODUCTE_SELECTED) THEN
41
42         (* Carrego la Sitja S2 *)
43         CarregarSitjaS2.FINAL_CARRERA := FC2;
44         CarregarSitjaS2.QUANT_RESTA_SITJA := QUANT_RESTA_SITJA_S2;
45         CarregarSitjaS2();
46         M4 := CarregarSitjaS2.MOTOR_OUT;
47         OK_CAPACITAT_SITJA_S2 := CarregarSitjaS2.OK_CAPACITAT;
48         (* Miro si carrego la Sitja S2 *)
49         IF OK_CAPACITAT_SITJA_S2 THEN
50             (* Guardo el producte seleccionat per pantalla com el producte de la Sitja S2 *)
51             PRODUCTE_SITJA_S2 := PRODUCTE_SELECTED;
52         END_IF;
53         (* Activo el flag *)
54         OK_PROD_S2 := TRUE;
55
56     ELSE
57
58         (* Miro si les Sitges S1 i S2 no estan buides i (la Sitja S3 esta buida o te el mateix producte que esta seleccionat) *)
```

```
59          IF PRODUCTE_SITJA_S1 > 0 AND PRODUCTE_SITJA_S2 > 0 AND (PRODUCTE_SITJA_S3 = 0 OR PRODUCTE_SITJA_S3 = PRODUCTE_SELECTED) THEN
60
61              (* Carrego la Sitja S3 *)
62              CarregarSitjaS3.FINAL_CARRERA := FC3;
63              CarregarSitjaS3.QUANT_RESTA_SITJA := QUANT_RESTA_SITJA_S3;
64              CarregarSitjaS3();
65              M4 := CarregarSitjaS3.MOTOR_OUT;
66              OK_CAPACITAT_SITJA_S3 := CarregarSitjaS3.OK_CAPACITAT;
67              (* Miro si carrego la Sitja S3 *)
68              IF OK_CAPACITAT_SITJA_S3 THEN
69                  (* Guardo el producte seleccionat per pantalla com el producte de la Sitja S3 *)
70                  PRODUCTE_SITJA_S3 := PRODUCTE_SELECTED;
71              END_IF;
72              (* Activo el flag *)
73              OK_PROD_S3 := TRUE;
74
75          ELSE
76              (* Miro si les Sitges S1, S2 i S3 no estan buides i (la Sitja S4 esta buida o te el mateix producte que esta seleccionat) *)
77              IF PRODUCTE_SITJA_S1 > 0 AND PRODUCTE_SITJA_S2 > 0 AND PRODUCTE_SITJA_S3 > 0 AND (PRODUCTE_SITJA_S4 = 0 OR PRODUCTE_SITJA_S4 = PRODUCTE_SELECTED) THEN
78
79                  (* Carrego la Sitja S4 *)
80                  CarregarSitjaS4.FINAL_CARRERA := FC4;
81                  CarregarSitjaS4.QUANT_RESTA_SITJA := QUANT_RESTA_SITJA_S4;
82                  CarregarSitjaS4();
83                  M4 := CarregarSitjaS4.MOTOR_OUT;
84                  M6 := M4;
85                  OK_CAPACITAT_SITJA_S4 := CarregarSitjaS4.OK_CAPACITAT;
86                  (* Miro si carrego la Sitja S4 *)
87                  IF OK_CAPACITAT_SITJA_S4 THEN
88                      (* Guardo el producte seleccionat per pantalla com el producte de la Sitja S4 *)
89                      PRODUCTE_SITJA_S4 := PRODUCTE_SELECTED;
90                  END_IF;
91                  (* Activo el flag *)
92                  OK_PROD_S4 := TRUE;
93
94              END_IF;
95          END_IF;
96      END_IF;
97  END_IF;
98
99  (* Selecciono segons el pas de la carrega *)
100 CASE PAS_CARREGAR OF
101
102     (* REPOS *)
103     0:
104         (* Miro si he pulsat boto a la pantalla, per carregar les materies primeres *)
105         IF CONFIG_CARREGAR_ONOFF > 0 THEN
106             (* Miro si estic carregant la Sitja S1 *)
107             IF OK_PROD_S1 THEN
108                 (* Calculo el nivell que haura d'arribar *)
109                 NIVELL_SITJA := QUANT_PROD_SITJA_S1 + CONFIG_QUANT_CARREGAR;
110             END_IF;
111             (* Miro si estic carregant la Sitja S2 *)
112             IF OK_PROD_S2 THEN
113                 (* Calculo el nivell que haura d'arribar *)
114                 NIVELL_SITJA := QUANT_PROD_SITJA_S2 + CONFIG_QUANT_CARREGAR;
115             END_IF;
116             (* Miro si estic carregant la Sitja S3 *)
117             IF OK_PROD_S3 THEN
118                 (* Calculo el nivell que haura d'arribar *)
119                 NIVELL_SITJA := QUANT_PROD_SITJA_S3 + CONFIG_QUANT_CARREGAR;
120             END_IF;
121             (* Miro si estic carregant la Sitja S4 *)
122             IF OK_PROD_S4 THEN
123                 (* Calculo el nivell que haura d'arribar *)
124                 NIVELL_SITJA := QUANT_PROD_SITJA_S4 + CONFIG_QUANT_CARREGAR;
125             END_IF;
126             (* Activo el REVOLVER *)
127             REVOLVER := 1;
128             (* Miro si el revolver esta situat correctament per omplir la Sitja corresponent *)
129             IF (OK_PROD_S1 AND FC1 > 0) OR (OK_PROD_S2 AND FC2 > 0) OR (OK_PROD_S3 AND FC3 > 0) OR (OK_PROD_S4 AND FC4 > 0) THEN
130                 (* Desactivo el REVOLVER *)
131                 REVOLVER := 0;
132                 (* Incremento el pas *)
133                 PAS_CARREGAR := PAS_CARREGAR + 1;
134             ELSE
135                 (* Miro si ha passat prou temps (en segons) per confirmar la posicio del Revolver *)
136                 IF ((SysGetSysTime(TRUE) - INSTANT_FC) >= CONFIG_TIMEOUT_FC1_A_FC4 * 1000000) THEN
137                     (* Activo el flag per indicar que hi ha alarma per timeout Revolver *)
138                     ALARMA_REVOLVER := TRUE;
139                 END_IF;
140             END_IF;
141         ELSE
142             (* Desactivo el REVOLVER *)
143             REVOLVER := 0;
144             (* Desactivo la senyal lluminosa H0 *)
145             H0 := 0;
146             (* Desactivo els motors M1, M2 *)
147             M1 := 0;
148             M2 := 0;
149             (* Guardo l'instant actual *)
150             INSTANT_FC := SysGetSysTime(TRUE);
151         END_IF;
152
153     (* VERIFICO QUE EL CAMIO ESTIGUI EN POSICIO *)
154     1:
155         (* Miro si el camio esta en posicio *)
156         IF S0 > 0 THEN
157             (* Activo la senyal lluminosa H0 per indicar al camio que pot descarregar el material *)
158             H0 := 1;
159             (* Activo els motors M1, M2 *)
160             M1 := 1;
161             M2 := 1;
162             (* Guardo l'instant actual *)
163             INSTANT_S1 := SysGetSysTime(TRUE);
164             (* Incremento el pas *)
165             PAS_CARREGAR := PAS_CARREGAR + 1;
166         END_IF;
167
168     (* VERIFICO QUE LA MATERIA PRIMERA ARRIBA AL SENSOR S1 *)
169     2:
170         (* Miro si el sensor S1 esta activat *)
171         IF S1 > 0 THEN
172             (* Activo els motors M3, M4 *)
173             M3 := 1;
174             M4 := 1;
175             (* Guardo l'instant actual *)
176             INSTANT_M1 := SysGetSysTime(TRUE);
177             (* Incremento el pas *)
178             PAS_CARREGAR := PAS_CARREGAR + 1;
179         ELSE
180             (* Miro si ha passat prou temps (en segons) per confirmar sensor S1 *)
181             IF ((SysGetSysTime(TRUE) - INSTANT_S1) >= CONFIG_TIMEOUT_S1 * 1000000) THEN
182                 (* Alarma timeout S1 *)
183                 ALARMA_S1 := TRUE;
184             END_IF;
185         END_IF;
186
187     (* ESPERO QUE ES CARREGUI TOTA LA MATERIA PRIMERA I CONTROLO EL TIMEOUT *)
188     3:
189         (* Miro si estic carregant la Sitja S1 *)
190         IF OK_PROD_S1 THEN
191             (* Miro si el nivell calculat de la Sitja S1 s'ha assolit *)
192             IF QUANT_PROD_SITJA_S1 >= NIVELL_SITJA THEN
193                 (* Incremento el pas *)
194                 PAS_CARREGAR := PAS_CARREGAR + 1;
```

```
            END_IF;
        END_IF;
        (* Miro si estic carregant la Sitja S2 *)
        IF OK_PROD_S2 THEN
            (* Miro si el nivell calculat de la Sitja S2 s'ha assolit *)
            IF QUANT_PROD_SITJA_S2 >= NIVELL_SITJA THEN
                (* Incremento el pas *)
                PAS_CARREGAR := PAS_CARREGAR + 1;
            END_IF;
        END_IF;
        (* Miro si estic carregant la Sitja S3 *)
        IF OK_PROD_S3 THEN
            (* Miro si el nivell calculat de la Sitja S3 s'ha assolit *)
            IF QUANT_PROD_SITJA_S3 >= NIVELL_SITJA THEN
                (* Incremento el pas *)
                PAS_CARREGAR := PAS_CARREGAR + 1;
            END_IF;
        END_IF;
        (* Miro si estic carregant la Sitja S4 *)
        IF OK_PROD_S4 THEN
            (* Miro si el nivell calculat de la Sitja S4 s'ha assolit *)
            IF QUANT_PROD_SITJA_S4 >= NIVELL_SITJA THEN
                (* Incremento el pas *)
                PAS_CARREGAR := PAS_CARREGAR + 1;
            END_IF;
        END_IF;
        (* Miro si ha passat prou temps (en segons) pel motor M1 *)
        IF ((SysGetSysTime(TRUE) - INSTANT_M1) >= CONFIG_TIMEOUT_M1 * 1000000) THEN
            (* Alarma per exces de temps de carrega *)
            ALARMA_M1 := TRUE;
        END_IF;

    (* ATURO LA CARREGA *)
    4:
        (* Aturo la carrega *)
        AturarCarrega();
        (* Reset del pas *)
        PAS_CARREGAR := 0;

END_CASE;
(* Miro si tinc alarma de timeout FC1..FC4 del Revolver *)
IF ALARMA_REVOLVER THEN
    (* Miro la Sitja que haig d'omplir *)
    IF OK_PROD_S1 THEN
        (* Alarma timeout FC1 *)
        ALARMES_CARREGAR := ALARMES_CARREGAR OR 1;
    END_IF;
    IF OK_PROD_S2 THEN
        (* Alarma timeout FC2 *)
        ALARMES_CARREGAR := ALARMES_CARREGAR OR 2;
    END_IF;
    IF OK_PROD_S3 THEN
        (* Alarma timeout FC3 *)
        ALARMES_CARREGAR := ALARMES_CARREGAR OR 4;
    END_IF;
    IF OK_PROD_S4 THEN
        (* Alarma timeout FC4 *)
        ALARMES_CARREGAR := ALARMES_CARREGAR OR 8;
    END_IF;
    (* Aturo la carrega *)
    PAS_CARREGAR := 4;
    (* Reset del flag *)
    ALARMA_REVOLVER := FALSE;
END_IF;

(* Miro si tinc alarma de timeout S1 *)
IF ALARMA_S1 THEN
    (* Alarma timeout *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 16;
    (* Aturo la carrega *)
    PAS_CARREGAR := 4;
    (* Reset flag *)
    ALARMA_S1 := FALSE;
END_IF;

(* Miro si el nivell de la Sitja que estic carregant ha arribat al limit maxim *)
IF (CONFIG_CARREGAR_ONOFF > 0 AND OK_PROD_S1 AND LST1_MAX < LST1_PRC) THEN
    (* Activo l'alarma per limit maxim Sitja S1 *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 32;
    (* Aturo la carrega *)
    PAS_CARREGAR := 4;
END_IF;
(* Miro si el nivell de la Sitja que estic carregant ha arribat al limit maxim *)
IF (CONFIG_CARREGAR_ONOFF > 0 AND OK_PROD_S2 AND LST2_MAX < LST2_PRC) THEN
    (* Activo l'alarma per limit maxim Sitja S2 *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 64;
    (* Aturo la carrega *)
    PAS_CARREGAR := 4;
END_IF;
(* Miro si el nivell de la Sitja que estic carregant ha arribat al limit maxim *)
IF (CONFIG_CARREGAR_ONOFF > 0 AND OK_PROD_S3 AND LST3_MAX < LST3_PRC) THEN
    (* Activo l'alarma per limit maxim Sitja S3 *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 128;
    (* Aturo la carrega *)
    PAS_CARREGAR := 4;
END_IF;
(* Miro si el nivell de la Sitja que estic carregant ha arribat al limit maxim *)
IF (CONFIG_CARREGAR_ONOFF > 0 AND OK_PROD_S4 AND LST4_MAX < LST4_PRC) THEN
    (* Activo l'alarma per limit maxim Sitja S4 *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 256;
    (* Aturo la carrega *)
    PAS_CARREGAR := 4;
END_IF;

(* Miro si M7 esta activat i el nivell de la Sitja S1 ha arribat a 0 *)
IF M7 > 0 AND LST1_PRC = 0 THEN
    (* Activo l'alarma per Sitja S1 buida *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 512;
    (* Aturo la fabricacio *)
    AturarFabricacio();
END_IF;
(* Miro si M8 esta activat i el nivell de la Sitja S2 ha arribat a 0 *)
IF M8 > 0 AND LST2_PRC = 0 THEN
    (* Activo l'alarma per Sitja S2 buida *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 1024;
    (* Aturo la fabricacio *)
    AturarFabricacio();
END_IF;
(* Miro si M9 esta activat i el nivell de la Sitja S3 ha arribat a 0 *)
IF M9 > 0 AND LST3_PRC = 0 THEN
    (* Activo l'alarma per Sitja S3 buida *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 2048;
    (* Aturo la fabricacio *)
    AturarFabricacio();
END_IF;
(* Miro si M10 esta activat i el nivell de la Sitja S4 ha arribat a 0 *)
IF M10 > 0 AND LST4_PRC = 0 THEN
    (* Activo l'alarma per Sitja S4 buida *)
    ALARMES_CARREGAR := ALARMES_CARREGAR OR 4096;
    (* Aturo la fabricacio *)
    AturarFabricacio();
END_IF;

(* Miro si tinc alarma de timeout per carregar materials *)
IF ALARMA_M1 THEN
```

```
331     (* Alarma timeout *)
332     ALARMES_CARREGAR := ALARMES_CARREGAR OR 8192;
333     (* Aturo la carrega *)
334     PAS_CARREGAR := 4;
335     (* Reset flag *)
336     ALARMA_M1 := FALSE;
337 END_IF;
338
339
```

FUNCTION_BLOCK Aturar

```
VAR
AturarCarrega : AturarCarrega;
AturarFabricacio : AturarFabricacio;
AturarExpedicio : AturarExpedicio;
AturarProductes : AturarProductes;
END_VAR
```

```
 1
 2 (* Aturo la carrega de materies primeres *)
 3 AturarCarrega();
 4 (* Aturo la fabricacio de productes *)
 5 AturarFabricacio();
 6 (* Aturo el magatzem de productes *)
 7 AturarProductes();
 8 (* Aturo l'expedicio de productes *)
 9 AturarExpedicio();
10
11
```

	Project : Pinsos FUNCTION BLOCK : Aturar	
	Release : Pinsos	Ver :1.00
	Author :	Date:14/05/2012
	Note :	Page:1 of 1

FUNCTION_BLOCK CarregarSitja

```
VAR_INPUT
QUANT_RESTA_SITJA : UINT;
FINAL_CARRERA : UINT;
END_VAR

VAR_OUTPUT
MOTOR_OUT : UINT;
OK_CAPACITAT : BOOL;
END_VAR

VAR_EXTERNAL
CONFIG_QUANT_CARREGAR : WORD;
M2 : WORD;
M3 : WORD;
END_VAR
```

```
1
2 (* Miro si la sitja te prou capacitat per carregar la quatitat de producte del camio *)
3 IF QUANT_RESTA_SITJA > CONFIG_QUANT_CARREGAR THEN
4     (* Activo el flag per indicar que la capacitat es correcta *)
5     OK_CAPACITAT := TRUE;
6 END_IF;
7
8 (* Miro si la capacitat es correcta, el revolver esta ben posicionat per carregar la sitja i el motor M3 esta en marxa *)
9 IF (OK_CAPACITAT AND FINAL_CARRERA > 0 AND M3 > 0) THEN
10     (* Activo el motor *)
11     MOTOR_OUT := 1;
12 ELSE
13     (* Desactivo el motor *)
14     MOTOR_OUT := 0;
15 END_IF;
16
17
```

```
VAR_EXTERNAL
ALARMES_FABRICAR : WORD;
CONFIG_FABRICAR_ONOFF : WORD;
CONFIG_PROD_FABRICAR : WORD;
CONFIG_QUANT_FABRICAR : WORD;
CONFIG_TEMPS_MESCLADORA : WORD;
CONFIG_TEMPS_MOLI : WORD;
CONFIG_TEMPS_SORTIDA_MESCLA : WORD;
CONFIG_TIMEOUT_S4 : WORD;
CONFIG_TIMEOUT_SS1_A_SS7 : WORD;
DOSIS_MESCLADORA : WORD;
DOSIS_MOLI_BLAT : WORD;
DOSIS_MOLI_BLAT_MORO : WORD;
DOSIS_MOLI_FARINA : WORD;
DOSIS_MOLI_ORDI : WORD;
EV4 : WORD;
EV5 : WORD;
EV6 : WORD;
M12 : WORD;
M13 : WORD;
M14 : WORD;
M15 : WORD;
M16 : WORD;
M17 : WORD;
M18 : WORD;
M19 : WORD;
PRODUCTE_SITJA_S1 : WORD;
PRODUCTE_SITJA_S2 : WORD;
PRODUCTE_SITJA_S3 : WORD;
PRODUCTE_SITJA_S4 : WORD;
RECEPTA_BLAT_MORO_PROD1 : WORD;
RECEPTA_BLAT_MORO_PROD2 : WORD;
RECEPTA_BLAT_MORO_PROD3 : WORD;
RECEPTA_BLAT_PROD1 : WORD;
RECEPTA_BLAT_PROD2 : WORD;
RECEPTA_BLAT_PROD3 : WORD;
RECEPTA_FARINA_PROD1 : WORD;
RECEPTA_FARINA_PROD2 : WORD;
RECEPTA_FARINA_PROD3 : WORD;
RECEPTA_ORDI_PROD1 : WORD;
RECEPTA_ORDI_PROD2 : WORD;
RECEPTA_ORDI_PROD3 : WORD;
S4 : WORD;
SSM1 : WORD;
SSM2 : WORD;
SSM3 : WORD;
PAS_FABRICAR : UINT;
END_VAR

VAR
ALARMA_S2 : BOOL;
ALARMA_S3 : BOOL;
ALARMA_S4 : BOOL;
ALARMA_SS1 : BOOL;
ALARMA_SS2 : BOOL;
ALARMA_SS3 : BOOL;
ALARMA_SS4 : BOOL;
ALARMA_SSA : BOOL;
ALARMA_SSB : BOOL;
AturarFabricacio : AturarFabricacio;
DosificarBlat : DosificarMateria;
DosificarBlatMoro : DosificarMateria;
DosificarFarina : DosificarMateria;
DosificarOrdi : DosificarMateria;
INSTANT : UDINT;
OK_BLAT : BOOL;
OK_BLAT_MORO : BOOL;
OK_DECREMENTAR : BOOL;
OK_DOSIFICAR : BOOL;
OK_DOSIS_BLAT : BOOL;
OK_DOSIS_BLAT_MORO : BOOL;
OK_DOSIS_FARINA : BOOL;
OK_DOSIS_ORDI : BOOL;
OK_FARINA : BOOL;
OK_ORDI : BOOL;
OK_PRODUCTE1 : BOOL;
OK_PRODUCTE2 : BOOL;
OK_PRODUCTE3 : BOOL;
ReceptaProducte1 : ReceptaProducte;
ReceptaProducte2 : ReceptaProducte;
ReceptaProducte3 : ReceptaProducte;
TancarMateria : TancarMateria;
ALARMA_FALTA_MATERIAL : BOOL;
END_VAR
```

```
 1
 2 (* Selecciono segons el pas en la dosificacio de les materies primeres *)
 3 CASE PAS_FABRICAR OF
 4
 5     (* EN REPOS *)
 6     0:
 7         (* Paro el Moli, les cintes de sortida i l'elevador *)
 8         M12 := 0;
 9         M13 := 0;
10         M14 := 0;
11         M15 := 0;
12         (* Miro si haig de produir el Producte 1 *)
13         IF CONFIG_FABRICAR_ONOFF > 0 AND CONFIG_PROD_FABRICAR = 1 THEN
14             (* Recepta Producte 1 *)
15             ReceptaProducte1.BLAT_MORO := RECEPTA_BLAT_MORO_PROD1;
16             ReceptaProducte1.BLAT := RECEPTA_BLAT_PROD1;
17             ReceptaProducte1.ORDI := RECEPTA_ORDI_PROD1;
18             ReceptaProducte1.FARINA := RECEPTA_FARINA_PROD1;
19             ReceptaProducte1();
20             OK_BLAT_MORO := ReceptaProducte1.OK_BLAT_MORO;
21             OK_BLAT := ReceptaProducte1.OK_BLAT;
22             OK_ORDI := ReceptaProducte1.OK_ORDI;
23             OK_FARINA := ReceptaProducte1.OK_FARINA;
24             OK_PRODUCTE1 := ReceptaProducte1.RESULTAT;
25         ELSE
26             OK_PRODUCTE1 := FALSE;
27         END_IF;
28
29         (* Miro si haig de produir el Producte 2 *)
30         IF CONFIG_FABRICAR_ONOFF > 0 AND CONFIG_PROD_FABRICAR = 2 THEN
31             (* Recepta Producte 2 *)
32             ReceptaProducte2.BLAT_MORO := RECEPTA_BLAT_MORO_PROD2;
33             ReceptaProducte2.BLAT := RECEPTA_BLAT_PROD2;
34             ReceptaProducte2.ORDI := RECEPTA_ORDI_PROD2;
35             ReceptaProducte2.FARINA := RECEPTA_FARINA_PROD2;
36             ReceptaProducte2();
37             OK_BLAT_MORO := ReceptaProducte2.OK_BLAT_MORO;
38             OK_BLAT := ReceptaProducte2.OK_BLAT;
39             OK_ORDI := ReceptaProducte2.OK_ORDI;
40             OK_FARINA := ReceptaProducte2.OK_FARINA;
41             OK_PRODUCTE2 := ReceptaProducte2.RESULTAT;
42         ELSE
43             OK_PRODUCTE2 := FALSE;
44         END_IF;
45
46         (* Miro si haig de produir el Producte 3 *)
47         IF CONFIG_FABRICAR_ONOFF > 0 AND CONFIG_PROD_FABRICAR = 3 THEN
48             (* Recepta Producte 3 *)
```

```
            ReceptaProducte3.BLAT_MORO := RECEPTA_BLAT_MORO_PROD3;
            ReceptaProducte3.BLAT := RECEPTA_BLAT_PROD3;
            ReceptaProducte3.ORDI := RECEPTA_ORDI_PROD3;
            ReceptaProducte3.FARINA := RECEPTA_FARINA_PROD3;
            ReceptaProducte3();
            OK_BLAT_MORO := ReceptaProducte3.OK_BLAT_MORO;
            OK_BLAT := ReceptaProducte3.OK_BLAT;
            OK_ORDI := ReceptaProducte3.OK_ORDI;
            OK_FARINA := ReceptaProducte3.OK_FARINA;
            OK_PRODUCTE3 := ReceptaProducte3.RESULTAT;
        ELSE
            OK_PRODUCTE3 := FALSE;
        END_IF;

        (* Calculo el numero de dosificacions pel Producte 1 *)
        IF OK_PRODUCTE1 THEN
            (* Multiplico el percentatge de cada materia primera per 2500 (quantitat maxima que admet la mescladora) i divideixo per 100 *)
            DOSIS_MOLI_BLAT_MORO := (RECEPTA_BLAT_MORO_PROD1 * 25);
            DOSIS_MOLI_BLAT := (RECEPTA_BLAT_PROD1 * 25);
            DOSIS_MOLI_ORDI := (RECEPTA_ORDI_PROD1 * 25);
            DOSIS_MOLI_FARINA := (RECEPTA_FARINA_PROD1 * 25);
        END_IF;

        (* Calculo el numero de dosificacions pel Producte 2 *)
        IF OK_PRODUCTE2 THEN
            (* Multiplico el percentatge de cada materia primera per 2500 (quantitat maxima que admet la mescladora) i divideixo per 100 *)
            DOSIS_MOLI_BLAT_MORO := (RECEPTA_BLAT_MORO_PROD2 * 25);
            DOSIS_MOLI_BLAT := (RECEPTA_BLAT_PROD2 * 25);
            DOSIS_MOLI_ORDI := (RECEPTA_ORDI_PROD2 * 25);
            DOSIS_MOLI_FARINA := (RECEPTA_FARINA_PROD2 * 25);
        END_IF;

        (* Calculo el numero de dosificacions pel Producte 3 *)
        IF OK_PRODUCTE3 THEN
            (* Multiplico el percentatge de cada materia primera per 2500 (quantitat maxima que admet la mescladora) i divideixo per 100 *)
            DOSIS_MOLI_BLAT_MORO := (RECEPTA_BLAT_MORO_PROD3 * 25);
            DOSIS_MOLI_BLAT := (RECEPTA_BLAT_PROD3 * 25);
            DOSIS_MOLI_ORDI := (RECEPTA_ORDI_PROD3 * 25);
            DOSIS_MOLI_FARINA := (RECEPTA_FARINA_PROD3 * 25);
        END_IF;

        (* Calculo el numero de dosificacions per la mescladora *)
        IF OK_PRODUCTE1 OR OK_PRODUCTE2 OR OK_PRODUCTE3 THEN
            (* Miro si encara no he calculat les dosis *)
            IF DOSIS_MESCLADORA = 0 THEN
                (* Divideixo la quantitat de producte a fabricar per 2500 (quantitat maxima que admet la mescladora) i divideixo per 100 *)
                DOSIS_MESCLADORA := (CONFIG_QUANT_FABRICAR / 2500);
                (* Miro si haig d'afegir una nova dosi per acabar d'arribar a la quantitat total *)
                IF (CONFIG_QUANT_FABRICAR MOD 2500) > 0 THEN
                    (* Incremento la dosi *)
                    DOSIS_MESCLADORA := DOSIS_MESCLADORA + 1;
                END_IF;
                (* Inicio els passos per dosificar les materies primeres *)
                PAS_FABRICAR := 1;
            END_IF;
        ELSE
            (* Reset de les Dosis per la Mescladora *)
            DOSIS_MESCLADORA := 0;
            (* Miro si vull fabricar producte pero em falta material *)
            IF CONFIG_FABRICAR_ONOFF > 0 THEN
                (* Alarma de falta material *)
                ALARMA_FALTA_MATERIAL := TRUE;
            END_IF;
        END_IF;

    (* INICI *)
    1:
        (* Capturo l'instant actual *)
        INSTANT := SysGetSysTime(TRUE);
        (* Poso en marxa el Moli, les cintes de sortida i l'elevador *)
        M12 := 1;
        M13 := 1;
        M14 := 1;
        M15 := 1;
        (* Incremento el pas *)
        PAS_FABRICAR := PAS_FABRICAR + 1;

    (* DOSIS BLAT DE MORO *)
    2:
        (* Dosifico el Blat de Moro *)
        DosificarBlatMoro.TIPUS_MATERIA := 1;
        DosificarBlatMoro.DOSIS_MATERIA := DOSIS_MOLI_BLAT_MORO;
        DosificarBlatMoro();
        OK_DOSIS_BLAT_MORO := DosificarBlatMoro.OK_DOSIS_MATERIA;
        ALARMA_SS1 := DosificarBlatMoro.ALARMA_SS1;
        ALARMA_SS2 := DosificarBlatMoro.ALARMA_SS2;
        ALARMA_SS3 := DosificarBlatMoro.ALARMA_SS3;
        ALARMA_SS4 := DosificarBlatMoro.ALARMA_SS4;
        ALARMA_SSA := DosificarBlatMoro.ALARMA_SSA;
        ALARMA_SSB := DosificarBlatMoro.ALARMA_SSB;
        ALARMA_S2 := DosificarBlatMoro.ALARMA_S2;
        ALARMA_S3 := DosificarBlatMoro.ALARMA_S3;
        (* Miro si ha acabat *)
        IF NOT OK_DOSIS_BLAT_MORO THEN
            (* Capturo l'instant actual *)
            INSTANT := SysGetSysTime(TRUE);
        END_IF;
        (* Miro si ha passat prou temps (en segons) *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TEMPS_MOLI * 1000000) THEN
            (* Incremento el pas *)
            PAS_FABRICAR := PAS_FABRICAR + 1;
        END_IF;

    (* TANCAR BLAT DE MORO *)
    3:
        (* Capturo l'instant actual *)
        INSTANT := SysGetSysTime(TRUE);
        (* Tanco la dosificacio de Blat de Moro *)
        TancarMateria();
        (* Incremento el pas *)
        PAS_FABRICAR := PAS_FABRICAR + 1;

    (* DOSIS BLAT *)
    4:
        (* Dosifico el Blat *)
        DosificarBlat.TIPUS_MATERIA := 2;
        DosificarBlat.DOSIS_MATERIA := DOSIS_MOLI_BLAT;
        DosificarBlat();
        OK_DOSIS_BLAT := DosificarBlat.OK_DOSIS_MATERIA;
        ALARMA_SS1 := DosificarBlat.ALARMA_SS1;
        ALARMA_SS2 := DosificarBlat.ALARMA_SS2;
        ALARMA_SS3 := DosificarBlat.ALARMA_SS3;
        ALARMA_SS4 := DosificarBlat.ALARMA_SS4;
        ALARMA_SSA := DosificarBlat.ALARMA_SSA;
        ALARMA_SSB := DosificarBlat.ALARMA_SSB;
        ALARMA_S2 := DosificarBlat.ALARMA_S2;
        ALARMA_S3 := DosificarBlat.ALARMA_S3;
        (* Miro si ha acabat *)
        IF NOT OK_DOSIS_BLAT THEN
            (* Capturo l'instant actual *)
            INSTANT := SysGetSysTime(TRUE);
        END_IF;
        (* Miro si ha passat prou temps (en segons) *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TEMPS_MOLI * 1000000) THEN
            (* Incremento el pas *)
            PAS_FABRICAR := PAS_FABRICAR + 1;
```

```
        END_IF;

    (* TANCAR BLAT *)
    5:
        (* Capturo l'instant actual *)
        INSTANT := SysGetSysTime(TRUE);
        (* Tanco la dosificacio de Blat *)
        TancarMateria();
        (* Incremento el pas *)
        PAS_FABRICAR := PAS_FABRICAR + 1;

    (* DOSIS ORDI *)
    6:
        (* Dosifico l'Ordi *)
        DosificarOrdi.TIPUS_MATERIA := 3;
        DosificarOrdi.DOSIS_MATERIA := DOSIS_MOLI_ORDI;
        DosificarOrdi();
        OK_DOSIS_ORDI := DosificarOrdi.OK_DOSIS_MATERIA;
        ALARMA_SS1 := DosificarOrdi.ALARMA_SS1;
        ALARMA_SS2 := DosificarOrdi.ALARMA_SS2;
        ALARMA_SS3 := DosificarOrdi.ALARMA_SS3;
        ALARMA_SS4 := DosificarOrdi.ALARMA_SS4;
        ALARMA_SSA := DosificarOrdi.ALARMA_SSA;
        ALARMA_SSB := DosificarOrdi.ALARMA_SSB;
        ALARMA_S2 := DosificarOrdi.ALARMA_S2;
        ALARMA_S3 := DosificarOrdi.ALARMA_S3;
        (* Miro si ha acabat *)
        IF NOT OK_DOSIS_ORDI THEN
            (* Capturo l'instant actual *)
            INSTANT := SysGetSysTime(TRUE);
        END_IF;
        (* Miro si ha passat prou temps (en segons) *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TEMPS_MOLI * 1000000) THEN
            (* Incremento el pas *)
            PAS_FABRICAR := PAS_FABRICAR + 1;
        END_IF;

    (* TANCAR ORDI *)
    7:
        (* Capturo l'instant actual *)
        INSTANT := SysGetSysTime(TRUE);
        (* Tanco la dosificacio d'Ordi *)
        TancarMateria();
        (* Incremento el pas *)
        PAS_FABRICAR := PAS_FABRICAR + 1;

    (* DOSIS FARINA *)
    8:
        (* Dosifico la Farina *)
        DosificarFarina.TIPUS_MATERIA := 4;
        DosificarFarina.DOSIS_MATERIA := DOSIS_MOLI_FARINA;
        DosificarFarina();
        OK_DOSIS_FARINA := DosificarFarina.OK_DOSIS_MATERIA;
        ALARMA_SS1 := DosificarFarina.ALARMA_SS1;
        ALARMA_SS2 := DosificarFarina.ALARMA_SS2;
        ALARMA_SS3 := DosificarFarina.ALARMA_SS3;
        ALARMA_SS4 := DosificarFarina.ALARMA_SS4;
        ALARMA_SSA := DosificarFarina.ALARMA_SSA;
        ALARMA_SSB := DosificarFarina.ALARMA_SSB;
        ALARMA_S2 := DosificarFarina.ALARMA_S2;
        ALARMA_S3 := DosificarFarina.ALARMA_S3;
        (* Miro si ha acabat *)
        IF NOT OK_DOSIS_FARINA THEN
            (* Capturo l'instant actual *)
            INSTANT := SysGetSysTime(TRUE);
        END_IF;
        (* Miro si ha passat prou temps (en segons) *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TEMPS_MOLI * 1000000) THEN
            (* Incremento el pas *)
            PAS_FABRICAR := PAS_FABRICAR + 1;
        END_IF;

    (* TANCAR FARINA *)
    9:
        (* Capturo l'instant actual *)
        INSTANT := SysGetSysTime(TRUE);
        (* Tanco la dosificacio de Farina *)
        TancarMateria();
        (* Incremento el pas *)
        PAS_FABRICAR := PAS_FABRICAR + 1;

    (* ACTIVAR MESCLADORA *)
    10:
        (* Desactivo el Moli, les cintes de sortida i l'elevador *)
        M12 := 0;
        M13 := 0;
        M14 := 0;
        M15 := 0;
        (* Activo la Mescladora, motor M16 *)
        M16 := 1;
        EV4 := 1;
        (* Miro si ha passat prou temps (en segons) *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TEMPS_MESCLADORA * 1000000) THEN
            (* Capturo l'instant actual *)
            INSTANT := SysGetSysTime(TRUE);
            (* Incremento el pas *)
            PAS_FABRICAR := PAS_FABRICAR + 1;
        END_IF;

    (* DESCARREGAR MESCLADORA *)
    11:
        (* Desactivo la Mescladora *)
        M16 := 0;
        EV4 := 0;
        (* Activo la descarrega de la Mescladora *)
        EV5 := 1;
        EV6 := 1;
        M17 := 1;
        (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor S4 *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TIMEOUT_S4 * 1000000) THEN
            (* Miro si el sensor S4 esta desactivat *)
            IF S4 = 0 THEN
                (* Activo el flag per indicar que hi ha alarma per timeout S4 *)
                ALARMA_S4 := TRUE;
            END_IF;
        END_IF;
        (* Miro si ha passat prou temps (en segons) *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TEMPS_SORTIDA_MESCLA * 1000000) THEN
            (* Capturo l'instant actual *)
            INSTANT := SysGetSysTime(TRUE);
            (* Incremento el pas *)
            PAS_FABRICAR := PAS_FABRICAR + 1;
            (* Activo el flag per decrementar la dosis *)
            OK_DECREMENTAR := TRUE;
        END_IF;

    (* DECREMENTAR DOSIS *)
    12:
        (* Desactivo la descarrega de la Mescladora *)
        EV5 := 0;
        EV6 := 0;
        M17 := 0;
        (* Miro si la dosis inicial es valida *)
        IF DOSIS_MESCLADORA > 0 AND OK_DOSIS_BLAT_MORO AND OK_DOSIS_BLAT AND OK_DOSIS_ORDI AND OK_DOSIS_FARINA AND OK_DECREMENTAR THEN
            (* Activo el flag per decrementar la dosis *)
            OK_DECREMENTAR := FALSE;
```

```
            (* Decremento la dosis inicial per calcular la dosis final *)
            DOSIS_MESCLADORA := DOSIS_MESCLADORA - 1;
        END_IF;
        (* Incremento el pas *)
        PAS_FABRICAR := PAS_FABRICAR + 1;

    (* REPETIR DOSIFICACIO PER MESCLADORA? *)
    13:
        (* Reset dels blocs *)
        OK_DOSIS_BLAT_MORO := FALSE;
        OK_DOSIS_BLAT := FALSE;
        OK_DOSIS_ORDI := FALSE;
        OK_DOSIS_FARINA := FALSE;
        OK_DECREMENTAR := FALSE;
        (* Miro si encara tinc alguna dosis per la mescladora *)
        IF DOSIS_MESCLADORA > 0 THEN
            (* Torno al primer pas per completar la dosificacio per la mescladora *)
            PAS_FABRICAR := 1;
        ELSE
            (* Incremento el pas *)
            PAS_FABRICAR := PAS_FABRICAR + 1;
        END_IF;

    (* FINAL *)
    14:
            (* Reset de la dosificacio *)
            DOSIS_MESCLADORA := 0;
            (* Paro la fabricacio *)
            CONFIG_FABRICAR_ONOFF := 0;
            (* Reset dels passos per dosificar les materies primeres *)
            PAS_FABRICAR := 0;
            (* Aturo la fabricacio *)
            AturarFabricacio();

END_CASE;

(* Miro si tinc alarma de timeout SS1 *)
IF ALARMA_SS1 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 1;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_SS1 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout SS2 *)
IF ALARMA_SS2 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 2;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_SS2 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout SS3 *)
IF ALARMA_SS3 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 4;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_SS3 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout SS4 *)
IF ALARMA_SS4 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 8;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_SS4 := FALSE;
END_IF;
(* Miro si tinc alarma de SSA *)
IF ALARMA_SSA THEN
    (* Alarma *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 16;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_SSA := FALSE;
END_IF;
(* Miro si tinc alarma de SSB *)
IF ALARMA_SSB THEN
    (* Alarma *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 32;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_SSB := FALSE;
END_IF;
(* Miro si tinc alarma de timeout S2 *)
IF ALARMA_S2 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 64;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_S2 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout S3 *)
IF ALARMA_S3 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 128;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_S3 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout S4 *)
IF ALARMA_S4 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 256;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
    (* Reset flag *)
    ALARMA_S4 := FALSE;
END_IF;
(* Miro si esta activat el sensor SSM1 *)
IF SSM1 > 0 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 512;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
END_IF;
(* Miro si esta activat el sensor SSM2 *)
IF SSM2 > 0 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 1024;
    (* Aturo la fabricacio *)
    PAS_FABRICAR := 14;
END_IF;
(* Miro si esta activat el sensor SSM3 *)
IF SSM3 > 0 THEN
    (* Alarma timeout *)
    ALARMES_FABRICAR := ALARMES_FABRICAR OR 2048;
    (* Aturo la fabricacio *)
```

```
457     PAS_FABRICAR := 14;
458 END_IF;
459 (* Miro si tinc una alarma de falta de material *)
460 IF ALARMA_FALTA_MATERIAL THEN
461     (* Alarma timeout *)
462     ALARMES_FABRICAR := ALARMES_FABRICAR OR 4096;
463     (* Aturo la fabricacio *)
464     PAS_FABRICAR := 14;
465     (* Reset flag *)
466     ALARMA_FALTA_MATERIAL := FALSE;
467 END_IF;
468
469
```

```
VAR_INPUT
BLAT_MORO : UINT;
BLAT : UINT;
ORDI : UINT;
FARINA : UINT;
END_VAR

VAR_OUTPUT
OK_BLAT_MORO : BOOL;
OK_BLAT : BOOL;
OK_ORDI : BOOL;
OK_FARINA : BOOL;
RESULTAT : BOOL;
END_VAR

VAR_EXTERNAL
CONFIG_QUANT_FABRICAR : WORD;
QUANT_BLAT_MORO_SITJES : WORD;
QUANT_BLAT_SITJES : WORD;
QUANT_FARINA_SITJES : WORD;
QUANT_ORDI_SITJES : WORD;
END_VAR

VAR
BLAT_MORO_QUANT : UINT;
BLAT_QUANT : UINT;
FARINA_QUANT : UINT;
ORDI_QUANT : UINT;
END_VAR
```

```
(* Multiplico la quantitat de producte a fabricar pel percentatge que em passen *)
BLAT_MORO_QUANT := CONFIG_QUANT_FABRICAR * BLAT_MORO / 100;
BLAT_QUANT := CONFIG_QUANT_FABRICAR * BLAT / 100;
ORDI_QUANT := CONFIG_QUANT_FABRICAR * ORDI / 100;
FARINA_QUANT := CONFIG_QUANT_FABRICAR * FARINA / 100;

(* Miro si tinc prou blat de moro a les sitjes de materies primeres *)
IF QUANT_BLAT_MORO_SITJES > BLAT_MORO_QUANT THEN
    (* Activo el flag per indicar que tinc prou blat de moro *)
    OK_BLAT_MORO := TRUE;
END_IF;

(* Miro si tinc prou blat a les sitjes de materies primeres *)
IF QUANT_BLAT_SITJES > BLAT_QUANT THEN
    (* Activo el flag per indicar que tinc prou blat *)
    OK_BLAT := TRUE;
END_IF;

(* Miro si tinc prou ordi a les sitjes de materies primeres *)
IF QUANT_ORDI_SITJES > ORDI_QUANT THEN
    (* Activo el flag per indicar que tinc prou ordi *)
    OK_ORDI := TRUE;
END_IF;

(* Miro si tinc prou farina a les sitjes de materies primeres *)
IF QUANT_FARINA_SITJES > FARINA_QUANT THEN
    (* Activo el flag per indicar que tinc prou farina *)
    OK_FARINA := TRUE;
END_IF;

(* Retorno el resultat *)
RESULTAT := (OK_BLAT_MORO AND OK_BLAT AND OK_ORDI AND OK_FARINA);

```

```
VAR_EXTERNAL
QUANT_BLAT_MORO_SITJES : WORD;
PRODUCTE_SITJA_S1 : WORD;
QUANT_PROD_SITJA_S1 : WORD;
QUANT_BLAT_SITJES : WORD;
QUANT_FARINA_SITJES : WORD;
QUANT_ORDI_SITJES : WORD;
QUANT_PROD_SITJA_S2 : WORD;
QUANT_PROD_SITJA_S3 : WORD;
QUANT_PROD_SITJA_S4 : WORD;
PRODUCTE_SITJA_S2 : WORD;
PRODUCTE_SITJA_S3 : WORD;
PRODUCTE_SITJA_S4 : WORD;
END_VAR
```

```
(* Miro si no tinc producte a la Sitja S1 *)
IF QUANT_PROD_SITJA_S1 = 0 THEN
    (* No tinc cap producte a la Sitja S1 *)
    PRODUCTE_SITJA_S1 := 0;
END_IF;

(* Miro si no tinc producte a la Sitja S2 *)
IF QUANT_PROD_SITJA_S2 = 0 THEN
    (* No tinc cap producte a la Sitja S2 *)
    PRODUCTE_SITJA_S2 := 0;
END_IF;

(* Miro si no tinc producte a la Sitja S3 *)
IF QUANT_PROD_SITJA_S3 = 0 THEN
    (* No tinc cap producte a la Sitja S3 *)
    PRODUCTE_SITJA_S3 := 0;
END_IF;

(* Miro si no tinc producte a la Sitja S4 *)
IF QUANT_PROD_SITJA_S4 = 0 THEN
    (* No tinc cap producte a la Sitja S4 *)
    PRODUCTE_SITJA_S4 := 0;
END_IF;

(* Selecciono en funcio del producte que conte la Sitja S1 *)
CASE PRODUCTE_SITJA_S1 OF

    1:  (* Blat de Moro *)
        (* Em guardo la quantitat de Blat de Moro a les Sitjes *)
        QUANT_BLAT_MORO_SITJES := QUANT_PROD_SITJA_S1;

    2:  (* Blat *)
        (* Em guardo la quantitat de Blat a les Sitjes *)
        QUANT_BLAT_SITJES := QUANT_PROD_SITJA_S1;

    3:  (* Ordi *)
        (* Em guardo la quantitat de Ordi a les Sitjes *)
        QUANT_ORDI_SITJES := QUANT_PROD_SITJA_S1;

    4:  (* Farina *)
        (* Em guardo la quantitat de Farina a les Sitjes *)
        QUANT_FARINA_SITJES := QUANT_PROD_SITJA_S1;

END_CASE;

(* Selecciono en funcio del producte que conte la Sitja S2 *)
CASE PRODUCTE_SITJA_S2 OF

    1:  (* Blat de Moro *)
        (* Em guardo la quantitat de Blat de Moro a les Sitjes *)
        QUANT_BLAT_MORO_SITJES := QUANT_PROD_SITJA_S2;

    2:  (* Blat *)
        (* Em guardo la quantitat de Blat a les Sitjes *)
        QUANT_BLAT_SITJES := QUANT_PROD_SITJA_S2;

    3:  (* Ordi *)
        (* Em guardo la quantitat de Ordi a les Sitjes *)
        QUANT_ORDI_SITJES := QUANT_PROD_SITJA_S2;

    4:  (* Farina *)
        (* Em guardo la quantitat de Farina a les Sitjes *)
        QUANT_FARINA_SITJES := QUANT_PROD_SITJA_S2;

END_CASE;

(* Selecciono en funcio del producte que conte la Sitja S3 *)
CASE PRODUCTE_SITJA_S3 OF

    1:  (* Blat de Moro *)
        (* Em guardo la quantitat de Blat de Moro a les Sitjes *)
        QUANT_BLAT_MORO_SITJES := QUANT_PROD_SITJA_S3;

    2:  (* Blat *)
        (* Em guardo la quantitat de Blat a les Sitjes *)
        QUANT_BLAT_SITJES := QUANT_PROD_SITJA_S3;

    3:  (* Ordi *)
        (* Em guardo la quantitat de Ordi a les Sitjes *)
        QUANT_ORDI_SITJES := QUANT_PROD_SITJA_S3;

    4:  (* Farina *)
        (* Em guardo la quantitat de Farina a les Sitjes *)
        QUANT_FARINA_SITJES := QUANT_PROD_SITJA_S3;

END_CASE;

(* Selecciono en funcio del producte que conte la Sitja S4 *)
CASE PRODUCTE_SITJA_S4 OF

    1:  (* Blat de Moro *)
        (* Em guardo la quantitat de Blat de Moro a les Sitjes *)
        QUANT_BLAT_MORO_SITJES := QUANT_PROD_SITJA_S4;

    2:  (* Blat *)
        (* Em guardo la quantitat de Blat a les Sitjes *)
        QUANT_BLAT_SITJES := QUANT_PROD_SITJA_S4;

    3:  (* Ordi *)
        (* Em guardo la quantitat de Ordi a les Sitjes *)
        QUANT_ORDI_SITJES := QUANT_PROD_SITJA_S4;

    4:  (* Farina *)
        (* Em guardo la quantitat de Farina a les Sitjes *)
        QUANT_FARINA_SITJES := QUANT_PROD_SITJA_S4;

END_CASE;


```

```
VAR_INPUT
TIPUS_MATERIA : UINT;
DOSIS_MATERIA : UINT;
END_VAR

VAR_OUTPUT
OK_DOSIS_MATERIA : BOOL;
ALARMA_SS1 : BOOL;
ALARMA_SS2 : BOOL;
ALARMA_SS3 : BOOL;
ALARMA_SS4 : BOOL;
ALARMA_SSA : BOOL;
ALARMA_SSB : BOOL;
ALARMA_S2 : BOOL;
ALARMA_S3 : BOOL;
END_VAR

VAR_EXTERNAL
PRODUCTE_SITJA_S1 : WORD;
PRODUCTE_SITJA_S2 : WORD;
PRODUCTE_SITJA_S3 : WORD;
PRODUCTE_SITJA_S4 : WORD;
BTA : WORD;
BTB : WORD;
M7 : WORD;
M8 : WORD;
M9 : WORD;
M10 : WORD;
EV0 : WORD;
EV1 : WORD;
EV2 : WORD;
EV3 : WORD;
M11 : WORD;
CONFIG_TEMPS_MOLI : WORD;
CONFIG_TIMEOUT_SS1_A_SS7 : WORD;
SS1 : WORD;
SS2 : WORD;
SS3 : WORD;
SS4 : WORD;
SSA : WORD;
SSB : WORD;
CONFIG_TIMEOUT_S2 : WORD;
CONFIG_TIMEOUT_S3 : WORD;
S2 : WORD;
S3 : WORD;
END_VAR

VAR
OK_ENTRAR_MOLI : BOOL;
OK_BASCULA : BOOL;
INSTANT_M7 : UDINT;
INSTANT_M8 : UDINT;
INSTANT_M9 : UDINT;
INSTANT_M10 : UDINT;
INSTANT_EV3 : UDINT;
INSTANT_EV2 : UDINT;
END_VAR
```

```
(* Reset dels flags *)
OK_DOSIS_MATERIA := FALSE;
OK_ENTRAR_MOLI := FALSE;
OK_BASCULA := FALSE;

(* Miro si tinc la materia a la Sitja S1 *)
IF PRODUCTE_SITJA_S1 = TIPUS_MATERIA THEN
    (* Miro si la bàscula no ha arribat al pes desitjat i la tramuja no esta plena *)
    IF BTA < DOSIS_MATERIA AND SSA = 0 THEN
        (* Activo el motor *)
        M7 := 1;
        (* Capturo l'instant actual *)
        INSTANT_M7 := SysGetSysTime(TRUE);
    ELSE
        (* Desactivo el motor M7 *)
        M7 := 0;
        (* Activo flag per entrar al moli *)
        OK_ENTRAR_MOLI := TRUE;
        (* Activo flag per indicar que bascula ha assolit dosis *)
        OK_BASCULA := TRUE;
    END_IF;
END_IF;
(* Miro si tinc la materia a la Sitja S2 *)
IF PRODUCTE_SITJA_S2 = TIPUS_MATERIA THEN
    (* Miro si la bàscula no ha arribat al pes desitjat i la tramuja no esta plena *)
    IF BTA < DOSIS_MATERIA AND SSA = 0 THEN
        (* Activo el motor M8 *)
        M8 := 1;
        (* Capturo l'instant actual *)
        INSTANT_M8 := SysGetSysTime(TRUE);
    ELSE
        (* Desactivo el motor M8 *)
        M8 := 0;
        (* Activo flag per entrar al moli *)
        OK_ENTRAR_MOLI := TRUE;
        (* Activo flag per indicar que bascula ha assolit dosis *)
        OK_BASCULA := TRUE;
    END_IF;
END_IF;
(* Miro si tinc la materia a la Sitja S3 *)
IF PRODUCTE_SITJA_S3 = TIPUS_MATERIA THEN
    (* Miro si la materia es Farina *)
    IF TIPUS_MATERIA = 4 THEN
        (* Miro si la bàscula no ha arribat al pes desitjat i la tramuja no esta plena *)
        IF BTB < DOSIS_MATERIA AND SSB = 0 THEN
            (* Activo el motor M9 *)
            M9 := 1;
            (* Desactivo la EV0 per NO anar al moli: FARINA NO VA AL MOLI !!!! *)
            EV0 := 0;
            (* Capturo l'instant actual *)
            INSTANT_M9 := SysGetSysTime(TRUE);
        ELSE
            (* Desactivo el motor M9 *)
            M9 := 0;
            (* Desactivo EV0 *)
            EV0 := 0;
            (* Desactivo flag per entrar al moli *)
            OK_ENTRAR_MOLI := FALSE;
            (* Activo flag per indicar que bascula ha assolit dosis *)
            OK_BASCULA := TRUE;
        END_IF;
    (* La materia no es Farina *)
    ELSE
        (* Miro si la bàscula no ha arribat al pes desitjat i la tramuja no esta plena *)
        IF BTA < DOSIS_MATERIA AND SSA = 0 THEN
            (* Activo el motor M9 *)
            M9 := 1;
            (* Activo la EV0 per anar al moli *)
            EV0 := 1;
            (* Capturo l'instant actual *)
            INSTANT_M9 := SysGetSysTime(TRUE);
        ELSE
            (* Desactivo el motor M9 *)
            M9 := 0;
            (* Desactivo EV0 *)
```

```
            EV0 := 0;
            (* Activo flag per entrar al moli *)
            OK_ENTRAR_MOLI := TRUE;
            (* Activo flag per indicar que bascula ha assolit dosis *)
            OK_BASCULA := TRUE;
        END_IF;
    END_IF;
END_IF;
(* Miro si tinc la materia a la Sitja S4 *)
IF PRODUCTE_SITJA_S4 = TIPUS_MATERIA THEN
    (* Miro si la bàscula no ha arribat al pes desitjat i la tramuja no esta plena *)
    IF BTB < DOSIS_MATERIA AND SSB = 0 THEN
        (* Activo el motor M10 *)
        M10 := 1;
        (* Capturo l'instant actual *)
        INSTANT_M10 := SysGetSysTime(TRUE);
    ELSE
        (* Desactivo el motor M10 *)
        M10 := 0;
        (* Desactivo flag per entrar al moli *)
        OK_ENTRAR_MOLI := FALSE;
        (* Activo flag per indicar que bascula ha assolit dosis *)
        OK_BASCULA := TRUE;
    END_IF;
END_IF;

(* Miro si la bascula ha assolit la dosis correcta *)
IF OK_BASCULA THEN
    (* Miro si no tinc els sensors SSA o SSB activats *)
    IF SSA = 0 AND SSB = 0 THEN
        (* Activo la sortida de les bascules *)
        EV1 := 1;
    ELSE
        (* Desactivo la sortida de les bascules *)
        EV1 := 0;
    END_IF;
    (* Miro si he d'activar les vies d'entrada cap al Moli *)
    IF OK_ENTRAR_MOLI THEN
        (* Miro si no tinc el sensor SSA activat *)
        IF SSA = 0 THEN
            (* Activo motor a tramuja A *)
            M11 := 1;
        ELSE
            (* Desactivo motor a tramuja A *)
            M11 := 0;
        END_IF;
        (* Activo/Desactivo electrovalvules *)
        EV2 := 1;
        EV3 := 0;
        (* Capturo l'instant actual *)
        INSTANT_EV2 := SysGetSysTime(TRUE);
    ELSE
        (* Activo/Desactivo electrovalvules *)
        EV2 := 0;
        EV3 := 1;
        (* Capturo l'instant actual *)
        INSTANT_EV3 := SysGetSysTime(TRUE);
    END_IF;
    (* Activo el flag per finalitzar dosificacio *)
    OK_DOSIS_MATERIA := TRUE;
ELSE
    (* Desactivo motors i electrovalvules *)
    EV1 := 0;
    EV2 := 0;
    EV3 := 0;
    M11 := 0;
END_IF;

(* Desactivo el flag per indicar que NO hi ha alarma per timeout SS1 *)
ALARMA_SS1 := FALSE;
(* Miro si he activat el motor M7 *)
IF M7 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor SS1 *)
    IF ((SysGetSysTime(TRUE) - INSTANT_M7) >= CONFIG_TIMEOUT_SS1_A_SS7 * 1000000) THEN
        (* Miro si el sensor SS1 esta desactivat *)
        IF SS1 = 0 THEN
            (* Activo el flag per indicar que hi ha alarma per timeout SS1 *)
            ALARMA_SS1 := TRUE;
        END_IF;
    END_IF;
END_IF;

(* Desactivo el flag per indicar que NO hi ha alarma per timeout SS2 *)
ALARMA_SS2 := FALSE;
(* Miro si he activat el motor M8 *)
IF M8 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor SS2 *)
    IF ((SysGetSysTime(TRUE) - INSTANT_M8) >= CONFIG_TIMEOUT_SS1_A_SS7 * 1000000) THEN
        (* Miro si el sensor SS2 esta desactivat *)
        IF SS2 = 0 THEN
            (* Activo el flag per indicar que hi ha alarma per timeout SS2 *)
            ALARMA_SS2 := TRUE;
        END_IF;
    END_IF;
END_IF;

(* Desactivo el flag per indicar que NO hi ha alarma per timeout SS3 *)
ALARMA_SS3 := FALSE;
(* Miro si he activat el motor M9 *)
IF M9 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor SS3 *)
    IF ((SysGetSysTime(TRUE) - INSTANT_M9) >= CONFIG_TIMEOUT_SS1_A_SS7 * 1000000) THEN
        (* Miro si el sensor SS3 esta desactivat *)
        IF SS3 = 0 THEN
            (* Activo el flag per indicar que hi ha alarma per timeout SS3 *)
            ALARMA_SS3 := TRUE;
        END_IF;
    END_IF;
END_IF;

(* Desactivo el flag per indicar que NO hi ha alarma per timeout SS4 *)
ALARMA_SS4 := FALSE;
(* Miro si he activat el motor M10 *)
IF M10 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor SS4 *)
    IF ((SysGetSysTime(TRUE) - INSTANT_M10) >= CONFIG_TIMEOUT_SS1_A_SS7 * 1000000) THEN
        (* Miro si el sensor SS4 esta desactivat *)
        IF SS4 = 0 THEN
            (* Activo el flag per indicar que hi ha alarma per timeout SS4 *)
            ALARMA_SS4 := TRUE;
        END_IF;
    END_IF;
END_IF;

(* Desactivo el flag per indicar que NO hi ha alarma per timeout SSA *)
ALARMA_SSA := FALSE;
(* Miro si esta activat el sensor SSA *)
IF SSA > 0 THEN
    (* Activo el flag per indicar que hi ha alarma de limit maxim de la tramuja *)
    ALARMA_SSA := TRUE;
END_IF;

(* Desactivo el flag per indicar que NO hi ha alarma per timeout SSB *)
ALARMA_SSB := FALSE;
(* Miro si esta activat el sensor SSB *)
IF SSB > 0 THEN
```

```
212     (* Activo el flag per indicar que hi ha alarma de limit maxim de la tramuja *)
213     ALARMA_SSB := TRUE;
214 END_IF;
215
216 (* Desactivo el flag per indicar que NO hi ha alarma per timeout S2 *)
217 ALARMA_S2 := FALSE;
218 (* Miro si he activat EV3 *)
219 IF EV3 > 0 THEN
220     (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor S2 *)
221     IF ((SysGetSysTime(TRUE) - INSTANT_EV3) >= CONFIG_TIMEOUT_S2 * 1000000) THEN
222         (* Miro si el sensor S2 esta desactivat *)
223         IF S2 = 0 THEN
224             (* Activo el flag per indicar que hi ha alarma per timeout S2 *)
225             ALARMA_S2 := TRUE;
226         END_IF;
227     END_IF;
228 END_IF;
229
230 (* Desactivo el flag per indicar que NO hi ha alarma per timeout S3 *)
231 ALARMA_S3 := FALSE;
232 (* Miro si he activat EV2 *)
233 IF EV2 > 0 THEN
234     (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor S3 *)
235     IF ((SysGetSysTime(TRUE) - INSTANT_EV2) >= CONFIG_TIMEOUT_S3 * 1000000) THEN
236         (* Miro si el sensor S3 esta desactivat *)
237         IF S3 = 0 THEN
238             (* Activo el flag per indicar que hi ha alarma per timeout S3 *)
239             ALARMA_S3 := TRUE;
240         END_IF;
241     END_IF;
242 END_IF;
243
244
245
```

FUNCTION_BLOCK TancarMateria

```
VAR_EXTERNAL
BTA : WORD;
BTB : WORD;
EV0 : WORD;
EV1 : WORD;
EV2 : WORD;
EV3 : WORD;
M7 : WORD;
M8 : WORD;
M9 : WORD;
M10 : WORD;
M11 : WORD;
M13 : WORD;
M14 : WORD;
END_VAR
```

```
1
2 (* Reset de les bascules *)
3 BTA := 0;
4 BTB := 0;
5 (* Desactivo motors i electrovalvules *)
6 EV0 := 0;
7 EV1 := 0;
8 EV2 := 0;
9 EV3 := 0;
10 M7 := 0;
11 M8 := 0;
12 M9 := 0;
13 M10 := 0;
14 M11 := 0;
15
16
```

	Project : Pinsos	
	FUNCTION BLOCK : TancarMateria	
	Release : Pinsos	Ver :1.00
	Author :	Date:14/05/2012
	Note :	Page:1 of 1

FUNCTION_BLOCK AturarCarrega

```
VAR_EXTERNAL
M1 : WORD;
M2 : WORD;
M3 : WORD;
M4 : WORD;
M5 : WORD;
M6 : WORD;
CONFIG_CARREGAR_ONOFF : WORD;
REVOLVER : WORD;
EV0 : WORD;
PAS_CARREGAR : UINT;
H0 : WORD;
END_VAR
```

```
 1
 2 (* Aturo tots els motors *)
 3 M1 := 0;
 4 M2 := 0;
 5 M3 := 0;
 6 M4 := 0;
 7 M5 := 0;
 8 M6 := 0;
 9 REVOLVER := 0;
10 (* Aturo totes les electrovalvules *)
11 EV0 := 0;
12 (* Aturo les senyals lluminoses *)
13 H0 := 0;
14 (* Aturo les ordres de confirmacio per pantalla *)
15 CONFIG_CARREGAR_ONOFF := 0;
16 (* Reset del pas *)
17 PAS_CARREGAR := 0;
18
19
```

FUNCTION_BLOCK AturarFabricacio

```
VAR_EXTERNAL
M7 : WORD;
M8 : WORD;
M9 : WORD;
M10 : WORD;
M11 : WORD;
M12 : WORD;
M13 : WORD;
M14 : WORD;
M15 : WORD;
M16 : WORD;
M17 : WORD;
EV1 : WORD;
EV3 : WORD;
EV4 : WORD;
EV5 : WORD;
EV6 : WORD;
EV7 : WORD;
EV8 : WORD;
EV9 : WORD;
EV10 : WORD;
EV11 : WORD;
EV12 : WORD;
CONFIG_FABRICAR_ONOFF : WORD;
EV2 : WORD;
PAS_FABRICAR : UINT;
END_VAR
```

```
 1
 2 (* Aturo tots els motors *)
 3 M7 := 0;
 4 M8 := 0;
 5 M9 := 0;
 6 M10 := 0;
 7 M11 := 0;
 8 M12 := 0;
 9 M13 := 0;
10 M14 := 0;
11 M15 := 0;
12 M16 := 0;
13 M17 := 0;
14 (* Aturo totes les electrovalvules *)
15 EV1 := 0;
16 EV2 := 0;
17 EV3 := 0;
18 EV4 := 0;
19 EV5 := 0;
20 EV6 := 0;
21 (* Aturo les ordres de confirmacio per pantalla *)
22 CONFIG_FABRICAR_ONOFF := 0;
23 (* Reset del pas *)
24 PAS_FABRICAR := 0;
25
26
```

```
VAR_EXTERNAL
M1 : WORD;
M2 : WORD;
M3 : WORD;
M4 : WORD;
M5 : WORD;
M6 : WORD;
M7 : WORD;
M8 : WORD;
M9 : WORD;
M10 : WORD;
M11 : WORD;
M12 : WORD;
M13 : WORD;
M14 : WORD;
M15 : WORD;
M16 : WORD;
M17 : WORD;
M18 : WORD;
M19 : WORD;
M20 : WORD;
EV1 : WORD;
EV3 : WORD;
EV4 : WORD;
EV5 : WORD;
EV6 : WORD;
EV7 : WORD;
EV8 : WORD;
EV9 : WORD;
EV10 : WORD;
EV11 : WORD;
EV12 : WORD;
H1 : WORD;
CONFIG_CARREGAR_ONOFF : WORD;
CONFIG_EXPEDIR_ONOFF : WORD;
CONFIG_FABRICAR_ONOFF : WORD;
REVOLVER : WORD;
EV2 : WORD;
PAS_EXPEDIR : UINT;
END_VAR
```

```
 1
 2 (* Aturo totes les electrovalvules *)
 3 EV10 := 0;
 4 EV11 := 0;
 5 EV12 := 0;
 6 (* Aturo les ordres de confirmacio per pantalla *)
 7 CONFIG_EXPEDIR_ONOFF := 0;
 8 (* Reset del pas *)
 9 PAS_EXPEDIR := 0;
10
11
```

```
VAR_EXTERNAL
CONFIG_EXPEDIR_ONOFF : WORD;
CONFIG_QUANT_EXPEDIR : WORD;
QUANT_PROD_SITJA_S5 : WORD;
SS5 : WORD;
SS6 : WORD;
SS7 : WORD;
QUANT_PROD_SITJA_S6 : WORD;
QUANT_PROD_SITJA_S7 : WORD;
KG_EXPEDITS_SITJA_S5 : WORD;
KG_EXPEDITS_SITJA_S6 : WORD;
KG_EXPEDITS_SITJA_S7 : WORD;
CONFIG_TIMEOUT_SS1_A_SS7 : WORD;
CONFIG_PROD_EXPEDIR : WORD;
EV10 : WORD;
EV11 : WORD;
EV12 : WORD;
ALARMES_EXPEDIR : WORD;
PAS_EXPEDIR : UINT;
LST5_PRC : WORD;
LST6_PRC : WORD;
LST7_PRC : WORD;
LST5_MAX : WORD;
LST6_MAX : WORD;
LST7_MAX : WORD;
END_VAR

VAR
AturarExpedicio : AturarExpedicio;
NIVELL_SITJA_S5 : UINT;
INSTANT : UDINT;
NIVELL_SITJA_S6 : UINT;
NIVELL_SITJA_S7 : UINT;
ALARMA_SS5 : BOOL;
ALARMA_SS6 : BOOL;
ALARMA_SS7 : BOOL;
ALARMA_NIVELL_S5 : BOOL;
ALARMA_NIVELL_S6 : BOOL;
ALARMA_NIVELL_S7 : BOOL;
END_VAR
```

```
 1
 2 (* Miro si l'expedicio esta aturada *)
 3 IF CONFIG_EXPEDIR_ONOFF = 0 THEN
 4     (* Reset del Pas *)
 5     PAS_EXPEDIR := 0;
 6     (* Aturo l'expedicio *)
 7     AturarExpedicio();
 8     RETURN;
 9 END_IF;
10
11
12 (* Selecciono segons el pas en l'expedicio dels productes *)
13 CASE PAS_EXPEDIR OF
14
15     (* EN REPOS *)
16     0:
17         (* Miro si haig d'expedir el Producte 1 *)
18         IF CONFIG_PROD_EXPEDIR = 1 THEN
19             (* Miro si tinc prou quantitat de Producte 1 a la Sitja S5 *)
20             IF CONFIG_QUANT_EXPEDIR <= QUANT_PROD_SITJA_S5 THEN
21                 (* Controlo que no estigui expedint cap altre producte *)
22                 IF EV11 = 0 AND EV12 = 0 THEN
23                     (* Calculo el nivell al que he d'arribar a la Sitja per completar l'expedicio *)
24                     NIVELL_SITJA_S5 := QUANT_PROD_SITJA_S5 - CONFIG_QUANT_EXPEDIR;
25                     (* Capturo l'instant actual *)
26                     INSTANT := SysGetSysTime(TRUE);
27                     (* Activo la sortida de la Sitja S5 *)
28                     EV10 := 1;
29                     (* Salto al pas per expedir Producte 1 *)
30                     PAS_EXPEDIR := 1;
31                 ELSE
32                     (* Reset del nivell *)
33                     NIVELL_SITJA_S5 := QUANT_PROD_SITJA_S5;
34                     (* Desactivo la sortida de la Sitja S5 *)
35                     EV10 := 0;
36                 END_IF;
37             ELSE
38                 (* Alarma de nivell Sitja S5 *)
39                 ALARMA_NIVELL_S5 := TRUE;
40             END_IF;
41         END_IF;
42
43         (* Miro si haig d'expedir el Producte 2 *)
44         IF CONFIG_PROD_EXPEDIR = 2 THEN
45             (* Miro si tinc prou quantitat de Producte 2 a la Sitja S6 *)
46             IF CONFIG_QUANT_EXPEDIR <= QUANT_PROD_SITJA_S6 THEN
47                 (* Controlo que no estigui expedint cap altre producte *)
48                 IF EV10 = 0 AND EV12 = 0 THEN
49                     (* Calculo el nivell al que he d'arribar a la Sitja per completar l'expedicio *)
50                     NIVELL_SITJA_S6 := QUANT_PROD_SITJA_S6 - CONFIG_QUANT_EXPEDIR;
51                     (* Capturo l'instant actual *)
52                     INSTANT := SysGetSysTime(TRUE);
53                     (* Activo la sortida de la Sitja S6 *)
54                     EV11 := 1;
55                     (* Salto al pas per expedir Producte 2 *)
56                     PAS_EXPEDIR := 2;
57                 ELSE
58                     (* Reset del nivell *)
59                     NIVELL_SITJA_S6 := QUANT_PROD_SITJA_S6;
60                     (* Desactivo la sortida de la Sitja S6 *)
61                     EV11 := 0;
62                 END_IF;
63             ELSE
64                 (* Alarma de nivell Sitja S6 *)
65                 ALARMA_NIVELL_S6 := TRUE;
66             END_IF;
67         END_IF;
68
69         (* Miro si haig d'expedir el Producte 3 *)
70         IF CONFIG_PROD_EXPEDIR = 3 THEN
71             (* Miro si tinc prou quantitat de Producte 3 a la Sitja S7 *)
72             IF CONFIG_QUANT_EXPEDIR <= QUANT_PROD_SITJA_S7 THEN
73                 (* Controlo que no estigui expedint cap altre producte *)
74                 IF EV10 = 0 AND EV11 = 0 THEN
75                     (* Calculo el nivell al que he d'arribar a la Sitja per completar l'expedicio *)
76                     NIVELL_SITJA_S7 := QUANT_PROD_SITJA_S7 - CONFIG_QUANT_EXPEDIR;
77                     (* Capturo l'instant actual *)
78                     INSTANT := SysGetSysTime(TRUE);
79                     (* Activo la sortida de la Sitja S7 *)
80                     EV12 := 1;
81                     (* Salto al pas per expedir Producte 3 *)
82                     PAS_EXPEDIR := 3;
83                 ELSE
84                     (* Reset del nivell *)
85                     NIVELL_SITJA_S7 := QUANT_PROD_SITJA_S7;
86                     (* Desactivo la sortida de la Sitja S7 *)
87                     EV12 := 0;
88                 END_IF;
89             ELSE
90                 (* Alarma de nivell Sitja S7 *)
91                 ALARMA_NIVELL_S7 := TRUE;
92             END_IF;
```

```
        END_IF;

    (* EXPEDIR PRODUCTE 1 *)
    1:
        (* Controlo el nivell de la Sitja fins que arriba al nivell calculat *)
        IF QUANT_PROD_SITJA_S5 <= NIVELL_SITJA_S5 THEN
            (* Salto al pas per finalitzar expedicio *)
            PAS_EXPEDIR := 4;
        END_IF;
        (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor SS *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TIMEOUT_SS1_A_SS7 * 1000000) THEN
            (* Miro si NO s'ha activat el sensor SS *)
            IF SS5 = 0 THEN
                (* Activo el flag per indicar que hi ha alarma per timeout SS5 *)
                ALARMA_SS5 := TRUE;
            END_IF;
        END_IF;
        (* Mostro els Kg expedits per pantalla *)
        KG_EXPEDITS_SITJA_S5 := CONFIG_QUANT_EXPEDIR;
        KG_EXPEDITS_SITJA_S6 := 0;
        KG_EXPEDITS_SITJA_S7 := 0;

    (* EXPEDIR PRODUCTE 2 *)
    2:
        (* Controlo el nivell de la Sitja fins que arriba al nivell calculat *)
        IF QUANT_PROD_SITJA_S6 <= NIVELL_SITJA_S6 THEN
            (* Salto al pas per finalitzar expedicio *)
            PAS_EXPEDIR := 4;
        END_IF;
        (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor SS *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TIMEOUT_SS1_A_SS7 * 1000000) THEN
            (* Miro si NO s'ha activat el sensor SS *)
            IF SS6 = 0 THEN
                (* Activo el flag per indicar que hi ha alarma per timeout SS6 *)
                ALARMA_SS6 := TRUE;
            END_IF;
        END_IF;
        (* Mostro els Kg expedits per pantalla *)
        KG_EXPEDITS_SITJA_S5 := 0;
        KG_EXPEDITS_SITJA_S6 := CONFIG_QUANT_EXPEDIR;
        KG_EXPEDITS_SITJA_S7 := 0;

    (* EXPEDIR PRODUCTE 3 *)
    3:
        (* Controlo el nivell de la Sitja fins que arriba al nivell calculat *)
        IF QUANT_PROD_SITJA_S7 <= NIVELL_SITJA_S7 THEN
            (* Salto al pas per finalitzar expedicio *)
            PAS_EXPEDIR := 4;
        END_IF;
        (* Miro si ha passat prou temps (en segons) per verificar el timeout del sensor SS *)
        IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TIMEOUT_SS1_A_SS7 * 1000000) THEN
            (* Miro si NO s'ha activat el sensor SS *)
            IF SS7 = 0 THEN
                (* Activo el flag per indicar que hi ha alarma per timeout SS7 *)
                ALARMA_SS7 := TRUE;
            END_IF;
        END_IF;
        (* Mostro els Kg expedits per pantalla *)
        KG_EXPEDITS_SITJA_S5 := 0;
        KG_EXPEDITS_SITJA_S6 := 0;
        KG_EXPEDITS_SITJA_S7 := CONFIG_QUANT_EXPEDIR;

    (* FINALITZAR EXPEDICIO PRODUCTES *)
    4:
        (* Desactivo les sortides *)
        EV10 := 0;
        EV11 := 0;
        EV12 := 0;
        (* Reset dels nivells *)
        NIVELL_SITJA_S5 := 0;
        NIVELL_SITJA_S6 := 0;
        NIVELL_SITJA_S7 := 0;
        (* Reset del pas *)
        PAS_EXPEDIR := 0;
        (* Desactivo l'expedicio *)
        CONFIG_EXPEDIR_ONOFF := 0;

END_CASE;

(* Miro si tinc alarma falta de producte Sitja S5 *)
IF ALARMA_NIVELL_S5 THEN
    (* Alarma timeout *)
    ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 1;
    (* Aturo l'expedicio *)
    PAS_EXPEDIR := 4;
    (* Reset flag *)
    ALARMA_NIVELL_S5 := FALSE;
END_IF;
(* Miro si tinc alarma falta de producte Sitja S6 *)
IF ALARMA_NIVELL_S6 THEN
    (* Alarma timeout *)
    ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 2;
    (* Aturo l'expedicio *)
    PAS_EXPEDIR := 4;
    (* Reset flag *)
    ALARMA_NIVELL_S6 := FALSE;
END_IF;
(* Miro si tinc alarma falta de producte Sitja S7 *)
IF ALARMA_NIVELL_S7 THEN
    (* Alarma timeout *)
    ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 4;
    (* Aturo l'expedicio *)
    PAS_EXPEDIR := 4;
    (* Reset flag *)
    ALARMA_NIVELL_S7 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout SS5 *)
IF ALARMA_SS5 THEN
    (* Alarma timeout *)
    ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 8;
    (* Aturo l'expedicio *)
    PAS_EXPEDIR := 4;
    (* Reset flag *)
    ALARMA_SS5 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout SS6 *)
IF ALARMA_SS6 THEN
    (* Alarma timeout *)
    ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 16;
    (* Aturo l'expedicio *)
    PAS_EXPEDIR := 4;
    (* Reset flag *)
    ALARMA_SS6 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout SS7 *)
IF ALARMA_SS7 THEN
    (* Alarma timeout *)
    ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 32;
    (* Aturo l'expedicio *)
    PAS_EXPEDIR := 4;
    (* Reset flag *)
    ALARMA_SS7 := FALSE;
END_IF;

(* Miro si el nivell de la Sitja S5 ha arribat al limit maxim *)
IF (LST5_MAX < LST5_PRC) THEN
```

```
229     (* Activo l'alarma per limit maxim Sitja S5 *)
230     ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 64;
231     (* Aturo l'expedicio *)
232     PAS_EXPEDIR := 4;
233 END_IF;
234 (* Miro si el nivell de la Sitja S6 ha arribat al limit maxim *)
235 IF (LST6_MAX < LST6_PRC) THEN
236     (* Activo l'alarma per limit maxim Sitja S6 *)
237     ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 128;
238     (* Aturo l'expedicio *)
239     PAS_EXPEDIR := 4;
240 END_IF;
241 (* Miro si el nivell de la Sitja S7 ha arribat al limit maxim *)
242 IF (LST7_MAX < LST7_PRC) THEN
243     (* Activo l'alarma per limit maxim Sitja S7 *)
244     ALARMES_EXPEDIR := ALARMES_EXPEDIR OR 256;
245     (* Aturo l'expedicio *)
246     PAS_EXPEDIR := 4;
247 END_IF;
248
249
```

```
VAR_EXTERNAL
M19 : WORD;
M18 : WORD;
M17 : WORD;
LST7_PRC : WORD;
LST7_MAX : WORD;
LST6_PRC : WORD;
LST6_MAX : WORD;
LST5_PRC : WORD;
LST5_MAX : WORD;
FC7 : WORD;
FC6 : WORD;
FC5 : WORD;
EV9 : WORD;
EV8 : WORD;
EV7 : WORD;
CONFIG_TIMEOUT_FC5_A_FC7 : WORD;
CONFIG_PROD_FABRICAR : WORD;
CONFIG_FABRICAR_ONOFF : WORD;
ALARMES_GUARDAR : WORD;
PAS_GUARDAR : UINT;
END_VAR

VAR
INSTANT : UDINT;
AturarProductes : AturarProductes;
ALARMA_PRODUCTE : BOOL;
ALARMA_FC7 : BOOL;
ALARMA_FC6 : BOOL;
ALARMA_FC5 : BOOL;
END_VAR
```

```
 1
 2 (* Miro si NO estic fabricant un producte *)
 3 IF M17 = 0 THEN
 4     (* Aturo els actuadors per guardar productes *)
 5     AturarProductes();
 6     (* Reset del pas *)
 7     PAS_GUARDAR := 0;
 8     RETURN;
 9 ELSE
10     (* Activo els motors de l'elevador i la cinta *)
11     M18 := 1;
12     M19 := 1;
13 END_IF;
14
15 (* Selecciono segons el pas per guardar els productes a les Sitges corresponents *)
16 CASE PAS_GUARDAR OF
17
18     (* EN REPOS *)
19     0:
20         (* Capturo l'instant actual *)
21         INSTANT := SysGetSysTime(TRUE);
22         (* Incremento el pas *)
23         PAS_GUARDAR := PAS_GUARDAR + 1;
24
25     (* INICI GUARDAR PRODUCTE *)
26     1:
27         (* Miro quin Producte estic fabricant *)
28         CASE CONFIG_PROD_FABRICAR OF
29
30             (* Producte 1 *)
31             1:
32                 (* Miro si el sensor FC5 esta activat *)
33                 IF FC5 > 0 THEN
34                     (* Activo l'electrovalvula per guardar el producte a la Sitja *)
35                     EV7 := 1;
36                     EV8 := 0;
37                     EV9 := 0;
38                     (* Incremento el pas *)
39                     PAS_GUARDAR := PAS_GUARDAR + 1;
40                 ELSE
41                     (* Miro si ha passat prou temps (en segons) *)
42                     IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TIMEOUT_FC5_A_FC7 * 1000000) THEN
43                         (* Activo l'alarma per timeout FC5 *)
44                         ALARMA_FC5 := TRUE;
45                     END_IF;
46                 END_IF;
47
48             (* Producte 2 *)
49             2:
50                 (* Miro si el sensor FC6 esta activat *)
51                 IF FC6 > 0 THEN
52                     (* Activo l'electrovalvula per guardar el producte a la Sitja *)
53                     EV7 := 0;
54                     EV8 := 1;
55                     EV9 := 0;
56                     (* Incremento el pas *)
57                     PAS_GUARDAR := PAS_GUARDAR + 1;
58                 ELSE
59                     (* Miro si ha passat prou temps (en segons) *)
60                     IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TIMEOUT_FC5_A_FC7 * 1000000) THEN
61                         (* Activo l'alarma per timeout FC6 *)
62                         ALARMES_GUARDAR := ALARMES_GUARDAR AND 2;
63                         (* Salto al pas per indicar alarma per timeout *)
64                         PAS_GUARDAR := 3;
65                     END_IF;
66                 END_IF;
67
68             (* Producte 3 *)
69             3:
70                 (* Miro si el sensor FC7 esta activat *)
71                 IF FC7 > 0 THEN
72                     (* Activo l'electrovalvula per guardar el producte a la Sitja *)
73                     EV7 := 0;
74                     EV8 := 0;
75                     EV9 := 1;
76                     (* Incremento el pas *)
77                     PAS_GUARDAR := PAS_GUARDAR + 1;
78                 ELSE
79                     (* Miro si ha passat prou temps (en segons) *)
80                     IF ((SysGetSysTime(TRUE) - INSTANT) >= CONFIG_TIMEOUT_FC5_A_FC7 * 1000000) THEN
81                         (* Activo l'alarma per timeout FC7 *)
82                         ALARMES_GUARDAR := ALARMES_GUARDAR AND 4;
83                         (* Salto al pas per indicar alarma per timeout *)
84                         PAS_GUARDAR := 3;
85                     END_IF;
86                 END_IF;
87
88         ELSE
89             (* Activo l'alarma per producte desconegut *)
90             ALARMA_PRODUCTE := TRUE;
91         END_CASE;
92
93
94     (* FINAL GUARDAR PRODUCTE *)
95     2:
96         (* Capturo l'instant actual i espero que s'apagui el motor M17 *)
97         INSTANT := SysGetSysTime(TRUE);
98
99     (* ALARMA *)
100    3:
101        (* Aturo la fabricacio dels productes *)
```

```
        CONFIG_FABRICAR_ONOFF := 0;
        AturarProductes();
        (* Reset del pas *)
        PAS_GUARDAR := 0;

END_CASE;

(* Miro si tinc alarma de timeout FC5 *)
IF ALARMA_FC5 THEN
    (* Alarma timeout *)
    ALARMES_GUARDAR := ALARMES_GUARDAR OR 1;
    (* Aturo la fabricacio *)
    PAS_GUARDAR := 3;
    (* Reset flag *)
    ALARMA_FC5 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout FC6 *)
IF ALARMA_FC6 THEN
    (* Alarma timeout *)
    ALARMES_GUARDAR := ALARMES_GUARDAR OR 2;
    (* Aturo la fabricacio *)
    PAS_GUARDAR := 3;
    (* Reset flag *)
    ALARMA_FC6 := FALSE;
END_IF;
(* Miro si tinc alarma de timeout FC7 *)
IF ALARMA_FC7 THEN
    (* Alarma timeout *)
    ALARMES_GUARDAR := ALARMES_GUARDAR OR 4;
    (* Aturo la fabricacio *)
    PAS_GUARDAR := 3;
    (* Reset flag *)
    ALARMA_FC7 := FALSE;
END_IF;
(* Miro si tinc alarma de producte desconegut *)
IF ALARMA_PRODUCTE THEN
    (* Alarma timeout *)
    ALARMES_GUARDAR := ALARMES_GUARDAR OR 8;
    (* Aturo la fabricacio *)
    PAS_GUARDAR := 3;
    (* Reset flag *)
    ALARMA_PRODUCTE := FALSE;
END_IF;
(* Miro si el nivell de la Sitja que estic carregant ha arribat al limit maxim *)
IF (CONFIG_PROD_FABRICAR = 1 AND LST5_MAX < LST5_PRC) THEN
    (* Activo l'alarma per limit maxim Sitja S5 *)
    ALARMES_GUARDAR := ALARMES_GUARDAR OR 16;
    (* Aturo la fabricacio *)
    PAS_GUARDAR := 3;
END_IF;
(* Miro si el nivell de la Sitja que estic carregant ha arribat al limit maxim *)
IF (CONFIG_PROD_FABRICAR = 2 AND LST6_MAX < LST6_PRC) THEN
    (* Activo l'alarma per limit maxim Sitja S6 *)
    ALARMES_GUARDAR := ALARMES_GUARDAR OR 32;
    (* Aturo la fabricacio *)
    PAS_GUARDAR := 3;
END_IF;
(* Miro si el nivell de la Sitja que estic carregant ha arribat al limit maxim *)
IF (CONFIG_PROD_FABRICAR = 3 AND LST7_MAX < LST7_PRC) THEN
    (* Activo l'alarma per limit maxim Sitja S7 *)
    ALARMES_GUARDAR := ALARMES_GUARDAR OR 64;
    (* Aturo la fabricacio *)
    PAS_GUARDAR := 3;
END_IF;

```

```
VAR_EXTERNAL
EV7 : WORD;
EV8 : WORD;
EV9 : WORD;
M18 : WORD;
M19 : WORD;
CONFIG_FABRICAR_ONOFF : WORD;
PAS_GUARDAR : UINT;
END_VAR
```

```
 1
 2 (* Aturo els motors per guardar productes a les Sitges *)
 3 M18 := 0;
 4 M19 := 0;
 5 (* Aturo totes les electrovalvules *)
 6 EV7 := 0;
 7 EV8 := 0;
 8 EV9 := 0;
 9 (* Reset del pas *)
10 PAS_GUARDAR := 0;
11
12
```

```
VAR_EXTERNAL
M1 : WORD;
M2 : WORD;
M3 : WORD;
M4 : WORD;
M5 : WORD;
M6 : WORD;
M7 : WORD;
M8 : WORD;
M9 : WORD;
M10 : WORD;
M11 : WORD;
M12 : WORD;
M13 : WORD;
M14 : WORD;
M15 : WORD;
M16 : WORD;
M17 : WORD;
M18 : WORD;
M19 : WORD;
REVOLVER : WORD;
ACK_M1 : WORD;
ACK_M10 : WORD;
ACK_M11 : WORD;
ACK_M12 : WORD;
ACK_M13 : WORD;
ACK_M14 : WORD;
ACK_M15 : WORD;
ACK_M16 : WORD;
ACK_M17 : WORD;
ACK_M18 : WORD;
ACK_M19 : WORD;
ACK_M2 : WORD;
ACK_M3 : WORD;
ACK_M4 : WORD;
ACK_M5 : WORD;
ACK_M6 : WORD;
ACK_M7 : WORD;
ACK_M8 : WORD;
ACK_M9 : WORD;
ACK_REVOLVER : WORD;
ALARMES_MOTORS1 : WORD;
ALARMES_MOTORS2 : WORD;
END_VAR

VAR
Aturar : Aturar;
INSTANT_ACK_M1 : UDINT;
INSTANT_ACK_M0 : UDINT;
INSTANT_ACK_M2 : UDINT;
INSTANT_ACK_M3 : UDINT;
INSTANT_ACK_M4 : UDINT;
INSTANT_ACK_M5 : UDINT;
INSTANT_ACK_REVOLVER : UDINT;
INSTANT_ACK_M6 : UDINT;
INSTANT_ACK_M7 : UDINT;
INSTANT_ACK_M8 : UDINT;
INSTANT_ACK_M9 : UDINT;
INSTANT_ACK_M10 : UDINT;
INSTANT_ACK_M11 : UDINT;
INSTANT_ACK_M12 : UDINT;
INSTANT_ACK_M13 : UDINT;
INSTANT_ACK_M14 : UDINT;
INSTANT_ACK_M15 : UDINT;
INSTANT_ACK_M16 : UDINT;
INSTANT_ACK_M17 : UDINT;
INSTANT_ACK_M18 : UDINT;
INSTANT_ACK_M19 : UDINT;
END_VAR
```

```
 1
 2 (* Miro si el motor M1 esta activat *)
 3 IF M1 > 0 THEN
 4     (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
 5     IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M1) >= 2000000) THEN
 6         (* Miro si l'entrada de confirmacio s'ha activat *)
 7         IF ACK_M1 = 0 THEN
 8             (* Activo alarma pel fallada del motor M1 *)
 9             ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 1;
10             (* Aturo *)
11             Aturar();
12         END_IF;
13     END_IF;
14 ELSE
15     (* Guardo l'instant actual *)
16     INSTANT_ACK_M1 := SysGetSysTime(TRUE);
17 END_IF;
18
19 (* Miro si el motor M2 esta activat *)
20 IF M2 > 0 THEN
21     (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
22     IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M2) >= 2000000) THEN
23         (* Miro si l'entrada de confirmacio s'ha activat *)
24         IF ACK_M2 = 0 THEN
25             (* Activo alarma pel fallada del motor M2 *)
26             ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 2;
27             (* Aturo *)
28             Aturar();
29         END_IF;
30     END_IF;
31 ELSE
32     (* Guardo l'instant actual *)
33     INSTANT_ACK_M2 := SysGetSysTime(TRUE);
34 END_IF;
35
36 (* Miro si el motor M3 esta activat *)
37 IF M3 > 0 THEN
38     (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
39     IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M3) >= 2000000) THEN
40         (* Miro si l'entrada de confirmacio s'ha activat *)
41         IF ACK_M3 = 0 THEN
42             (* Activo alarma pel fallada del motor M3 *)
43             ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 4;
44             (* Aturo *)
45             Aturar();
46         END_IF;
47     END_IF;
48 ELSE
49     (* Guardo l'instant actual *)
50     INSTANT_ACK_M3 := SysGetSysTime(TRUE);
51 END_IF;
52
53 (* Miro si el motor M4 esta activat *)
54 IF M4 > 0 THEN
55     (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
56     IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M4) >= 2000000) THEN
57         (* Miro si l'entrada de confirmacio s'ha activat *)
58         IF ACK_M4 = 0 THEN
59             (* Activo alarma pel fallada del motor M4 *)
60             ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 8;
61             (* Aturo *)
62             Aturar();
63         END_IF;
```

```
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M4 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M5 esta activat *)
IF M5 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M5) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M5 = 0 THEN
            (* Activo alarma pel fallada del motor M5 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 16;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M5 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M6 esta activat *)
IF M6 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M6) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M6 = 0 THEN
            (* Activo alarma pel fallada del motor M6 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 32;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M6 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor REVOLVER esta activat *)
IF REVOLVER > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_REVOLVER) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_REVOLVER = 0 THEN
            (* Activo alarma pel fallada del motor REVOLVER *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 64;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_REVOLVER := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M7 esta activat *)
IF M7 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M7) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M7 = 0 THEN
            (* Activo alarma pel fallada del motor M7 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 128;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M7 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M8 esta activat *)
IF M8 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M8) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M8 = 0 THEN
            (* Activo alarma pel fallada del motor M8 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 256;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M8 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M9 esta activat *)
IF M9 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M9) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M9 = 0 THEN
            (* Activo alarma pel fallada del motor M9 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 512;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M9 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M10 esta activat *)
IF M10 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M10) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M10 = 0 THEN
            (* Activo alarma pel fallada del motor M10 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 1024;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M10 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M11 esta activat *)
IF M11 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M11) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M11 = 0 THEN
            (* Activo alarma pel fallada del motor M11 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 2048;
            (* Aturo *)
            Aturar();
        END_IF;
```

```
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M11 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M12 esta activat *)
IF M12 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M12) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M12 = 0 THEN
            (* Activo alarma pel fallada del motor M12 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 4096;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M12 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M13 esta activat *)
IF M13 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M13) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M13 = 0 THEN
            (* Activo alarma pel fallada del motor M13 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 8192;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M13 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M14 esta activat *)
IF M14 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M14) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M14 = 0 THEN
            (* Activo alarma pel fallada del motor M14 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 16384;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M14 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M15 esta activat *)
IF M15 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M15) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M15 = 0 THEN
            (* Activo alarma pel fallada del motor M15 *)
            ALARMES_MOTORS1 := ALARMES_MOTORS1 OR 32768;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M15 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M16 esta activat *)
IF M16 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M16) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M16 = 0 THEN
            (* Activo alarma pel fallada del motor M16 *)
            ALARMES_MOTORS2 := ALARMES_MOTORS2 OR 1;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M16 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M17 esta activat *)
IF M17 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M17) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M17 = 0 THEN
            (* Activo alarma pel fallada del motor M17 *)
            ALARMES_MOTORS2 := ALARMES_MOTORS2 OR 2;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M17 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M18 esta activat *)
IF M18 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M18) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M18 = 0 THEN
            (* Activo alarma pel fallada del motor M18 *)
            ALARMES_MOTORS2 := ALARMES_MOTORS2 OR 4;
            (* Aturo *)
            Aturar();
        END_IF;
    END_IF;
ELSE
    (* Guardo l'instant actual *)
    INSTANT_ACK_M18 := SysGetSysTime(TRUE);
END_IF;

(* Miro si el motor M19 esta activat *)
IF M19 > 0 THEN
    (* Miro si ha passat prou temps (en segons) per confirmar l'activacio del motor *)
    IF ((SysGetSysTime(TRUE) - INSTANT_ACK_M19) >= 2000000) THEN
        (* Miro si l'entrada de confirmacio s'ha activat *)
        IF ACK_M19 = 0 THEN
            (* Activo alarma pel fallada del motor M19 *)
            ALARMES_MOTORS2 := ALARMES_MOTORS2 OR 8;
            (* Aturo *)
            Aturar();
        END_IF;
```

```
336     END_IF;
337 ELSE
338     (* Guardo l'instant actual *)
339     INSTANT_ACK_M19 := SysGetSysTime(TRUE);
340 END_IF;
341
342
```

```
VAR_EXTERNAL
M1 : WORD;
M10 : WORD;
M11 : WORD;
M12 : WORD;
M13 : WORD;
M14 : WORD;
M15 : WORD;
M16 : WORD;
M17 : WORD;
M18 : WORD;
M19 : WORD;
M2 : WORD;
M3 : WORD;
M4 : WORD;
M5 : WORD;
M6 : WORD;
M7 : WORD;
M8 : WORD;
M9 : WORD;
ACK_M1 : WORD;
ACK_M10 : WORD;
ACK_M11 : WORD;
ACK_M12 : WORD;
ACK_M13 : WORD;
ACK_M14 : WORD;
ACK_M15 : WORD;
ACK_M16 : WORD;
ACK_M17 : WORD;
ACK_M18 : WORD;
ACK_M19 : WORD;
ACK_M2 : WORD;
ACK_M3 : WORD;
ACK_M4 : WORD;
ACK_M5 : WORD;
ACK_M6 : WORD;
ACK_M7 : WORD;
ACK_M8 : WORD;
ACK_M9 : WORD;
ACK_REVOLVER : WORD;
REVOLVER : WORD;
PC_COMANDA_1 : WORD;
PC_COMANDA_2 : WORD;
PC_COMANDA_3 : WORD;
PC_COMANDA_4 : WORD;
FC1 : WORD;
FC2 : WORD;
FC3 : WORD;
FC4 : WORD;
FC5 : WORD;
FC6 : WORD;
FC7 : WORD;
EV3 : WORD;
EV2 : WORD;
EV6 : WORD;
S1 : WORD;
S2 : WORD;
S3 : WORD;
S4 : WORD;
SS1 : WORD;
SS2 : WORD;
SS3 : WORD;
SS4 : WORD;
SS5 : WORD;
SS6 : WORD;
SS7 : WORD;
CONFIG_PROD_FABRICAR : WORD;
EV7 : WORD;
EV8 : WORD;
EV9 : WORD;
EV10 : WORD;
EV11 : WORD;
EV12 : WORD;
IM1 : WORD;
IM2 : WORD;
IM3 : WORD;
IM4 : WORD;
LST1_VAL : WORD;
LST2_VAL : WORD;
LST3_VAL : WORD;
LST4_VAL : WORD;
LST5_VAL : WORD;
LST6_VAL : WORD;
LST7_VAL : WORD;
BTA : WORD;
BTB : WORD;
EV0 : WORD;
EV1 : WORD;
END_VAR

VAR
PAS_PC2_16 : UINT;
INSTANT_PC2_16 : UDINT;
INSTANT_PC2_32 : UDINT;
INSTANT_PC3_64 : UDINT;
INSTANT_PC3_128 : UDINT;
INSTANT_PC3_256 : UDINT;
INSTANT_PC3_4 : UDINT;
INSTANT_PC3_8 : UDINT;
INSTANT_PC3_16 : UDINT;
INSTANT_PC3_32 : UDINT;
END_VAR
```

```
 1
 2 (* Miro si haig de simular la confirmacio del motor M1 *)
 3 IF (PC_COMANDA_1 AND 1) > 0 THEN
 4     (* Activo o desactivo la confirmacio segons l'estat del motor *)
 5     ACK_M1 := M1;
 6 END_IF;
 7
 8 (* Miro si haig de simular la confirmacio del motor M2 *)
 9 IF (PC_COMANDA_1 AND 2) > 0 THEN
10     (* Activo o desactivo la confirmacio segons l'estat del motor *)
11     ACK_M2 := M2;
12 END_IF;
13
14 (* Miro si haig de simular la confirmacio del motor M3 *)
15 IF (PC_COMANDA_1 AND 4) > 0 THEN
16     (* Activo o desactivo la confirmacio segons l'estat del motor *)
17     ACK_M3 := M3;
18 END_IF;
19
20 (* Miro si haig de simular la confirmacio del motor M4 *)
21 IF (PC_COMANDA_1 AND 8) > 0 THEN
22     (* Activo o desactivo la confirmacio segons l'estat del motor *)
23     ACK_M4 := M4;
24 END_IF;
25
26 (* Miro si haig de simular la confirmacio del motor M5 *)
27 IF (PC_COMANDA_1 AND 16) > 0 THEN
28     (* Activo o desactivo la confirmacio segons l'estat del motor *)
29     ACK_M5 := M5;
30 END_IF;
```

```
31
32 (* Miro si haig de simular la confirmacio del motor M6 *)
33 IF (PC_COMANDA_1 AND 32) > 0 THEN
34     (* Activo o desactivo la confirmacio segons l'estat del motor *)
35     ACK_M6 := M6;
36 END_IF;
37
38 (* Miro si haig de simular la confirmacio del motor M7 *)
39 IF (PC_COMANDA_1 AND 64) > 0 THEN
40     (* Activo o desactivo la confirmacio segons l'estat del motor *)
41     ACK_M7 := M7;
42 END_IF;
43
44 (* Miro si haig de simular la confirmacio del motor M8 *)
45 IF (PC_COMANDA_1 AND 128) > 0 THEN
46     (* Activo o desactivo la confirmacio segons l'estat del motor *)
47     ACK_M8 := M8;
48 END_IF;
49
50 (* Miro si haig de simular la confirmacio del motor M9 *)
51 IF (PC_COMANDA_1 AND 256) > 0 THEN
52     (* Activo o desactivo la confirmacio segons l'estat del motor *)
53     ACK_M9 := M9;
54 END_IF;
55
56 (* Miro si haig de simular la confirmacio del motor M10 *)
57 IF (PC_COMANDA_1 AND 512) > 0 THEN
58     (* Activo o desactivo la confirmacio segons l'estat del motor *)
59     ACK_M10 := M10;
60 END_IF;
61
62 (* Miro si haig de simular la confirmacio del motor M11 *)
63 IF (PC_COMANDA_1 AND 1024) > 0 THEN
64     (* Activo o desactivo la confirmacio segons l'estat del motor *)
65     ACK_M11 := M11;
66 END_IF;
67
68 (* Miro si haig de simular la confirmacio del motor M12 *)
69 IF (PC_COMANDA_1 AND 2048) > 0 THEN
70     (* Activo o desactivo la confirmacio segons l'estat del motor *)
71     ACK_M12 := M12;
72 END_IF;
73
74 (* Miro si haig de simular la confirmacio del motor M13 *)
75 IF (PC_COMANDA_1 AND 4096) > 0 THEN
76     (* Activo o desactivo la confirmacio segons l'estat del motor *)
77     ACK_M13 := M13;
78 END_IF;
79
80 (* Miro si haig de simular la confirmacio del motor M14 *)
81 IF (PC_COMANDA_1 AND 8192) > 0 THEN
82     (* Activo o desactivo la confirmacio segons l'estat del motor *)
83     ACK_M14 := M14;
84 END_IF;
85
86 (* Miro si haig de simular la confirmacio del motor M15 *)
87 IF (PC_COMANDA_1 AND 16384) > 0 THEN
88     (* Activo o desactivo la confirmacio segons l'estat del motor *)
89     ACK_M15 := M15;
90 END_IF;
91
92 (* Miro si haig de simular la confirmacio del motor REVOLVER *)
93 IF (PC_COMANDA_1 AND 32768) > 0 THEN
94     (* Activo o desactivo la confirmacio segons l'estat del motor *)
95     ACK_REVOLVER := REVOLVER;
96 END_IF;
97
98 (* Miro si haig de simular la confirmacio del motor M16 *)
99 IF (PC_COMANDA_2 AND 1) > 0 THEN
100     (* Activo o desactivo la confirmacio segons l'estat del motor *)
101     ACK_M16 := M16;
102 END_IF;
103
104 (* Miro si haig de simular la confirmacio del motor M17 *)
105 IF (PC_COMANDA_2 AND 2) > 0 THEN
106     (* Activo o desactivo la confirmacio segons l'estat del motor *)
107     ACK_M17 := M17;
108 END_IF;
109
110 (* Miro si haig de simular la confirmacio del motor M18 *)
111 IF (PC_COMANDA_2 AND 4) > 0 THEN
112     (* Activo o desactivo la confirmacio segons l'estat del motor *)
113     ACK_M18 := M18;
114 END_IF;
115
116 (* Miro si haig de simular la confirmacio del motor M19 *)
117 IF (PC_COMANDA_2 AND 8) > 0 THEN
118     (* Activo o desactivo la confirmacio segons l'estat del motor *)
119     ACK_M19 := M19;
120 END_IF;
121
122 (* Miro si haig d'activar FC1..FC4 sequencialment quan el REVOLVER esta activat *)
123 IF (PC_COMANDA_2 AND 16) > 0 THEN
124     (* Miro si REVOLVER esta activat *)
125     IF REVOLVER > 0 THEN
126         (* Selecciono en funcio del pas *)
127         CASE PAS_PC2_16 OF
128             0:
129                 FC1 := 1;
130                 FC2 := 0;
131                 FC3 := 0;
132                 FC4 := 0;
133                 INSTANT_PC2_16 := SysGetSysTime(TRUE);
134                 PAS_PC2_16 := PAS_PC2_16 + 1;
135             1:
136                 (* Miro si ha passat prou temps (en segons) per canviar la posicio del Revolver *)
137                 IF ((SysGetSysTime(TRUE) - INSTANT_PC2_16) >= 2000000) THEN
138                     FC1 := 0;
139                     FC2 := 1;
140                     FC3 := 0;
141                     FC4 := 0;
142                     INSTANT_PC2_16 := SysGetSysTime(TRUE);
143                     PAS_PC2_16 := PAS_PC2_16 + 1;
144                 END_IF;
145             2:
146                 (* Miro si ha passat prou temps (en segons) per canviar la posicio del Revolver *)
147                 IF ((SysGetSysTime(TRUE) - INSTANT_PC2_16) >= 2000000) THEN
148                     FC1 := 0;
149                     FC2 := 0;
150                     FC3 := 1;
151                     FC4 := 0;
152                     INSTANT_PC2_16 := SysGetSysTime(TRUE);
153                     PAS_PC2_16 := PAS_PC2_16 + 1;
154                 END_IF;
155             3:
156                 (* Miro si ha passat prou temps (en segons) per canviar la posicio del Revolver *)
157                 IF ((SysGetSysTime(TRUE) - INSTANT_PC2_16) >= 2000000) THEN
158                     FC1 := 0;
159                     FC2 := 0;
160                     FC3 := 0;
161                     FC4 := 1;
162                     INSTANT_PC2_16 := SysGetSysTime(TRUE);
163                     PAS_PC2_16 := PAS_PC2_16 + 1;
164                 END_IF;
165             4:
166                 (* Miro si ha passat prou temps (en segons) per canviar la posicio del Revolver *)
```

```
                IF ((SysGetSysTime(TRUE) - INSTANT_PC2_16) >= 2000000) THEN
                    PAS_PC2_16 := 0;
                END_IF;
        END_CASE;
    END_IF;
END_IF;

(* Miro si haig d'activar S1 3 segons despres que M2 activat *)
IF (PC_COMANDA_2 AND 32) > 0 THEN
    (* Miro si M2 esta activat *)
    IF M2 > 0 THEN
        (* Miro si ha passat prou temps (en segons) *)
        IF ((SysGetSysTime(TRUE) - INSTANT_PC2_16) >= 3000000) THEN
            (* Activo S1 *)
            S1 := 1;
        END_IF;
    ELSE
        (* Desactivo S1 *)
        S1 := 0;
        INSTANT_PC2_32 := SysGetSysTime(TRUE);
    END_IF;
END_IF;

(* Miro si haig d'omplir la Sitja S1 quan M4 i FC1 activats *)
IF (PC_COMANDA_2 AND 64) > 0 THEN
    (* Miro si M4 i FC1 estan activats *)
    IF M4 > 0 AND FC1 > 0 THEN
        (* Omplo la Sitja S1 *)
        LST1_VAL := LST1_VAL + 1;
    END_IF;
END_IF;

(* Miro si haig d'omplir la Sitja S2 quan M4 i FC2 activats *)
IF (PC_COMANDA_2 AND 128) > 0 THEN
    (* Miro si M4 i FC2 estan activats *)
    IF M4 > 0 AND FC2 > 0 THEN
        (* Omplo la Sitja S2 *)
        LST2_VAL := LST2_VAL + 1;
    END_IF;
END_IF;

(* Miro si haig d'omplir la Sitja S3 quan M4 i FC3 activats *)
IF (PC_COMANDA_2 AND 256) > 0 THEN
    (* Miro si M4 i FC3 estan activats *)
    IF M4 > 0 AND FC3 > 0 THEN
        (* Omplo la Sitja S3 *)
        LST3_VAL := LST3_VAL + 1;
    END_IF;
END_IF;

(* Miro si haig d'omplir la Sitja S4 quan M4 i FC4 activats *)
IF (PC_COMANDA_2 AND 512) > 0 THEN
    (* Miro si M4 i FC4 estan activats *)
    IF M4 > 0 AND FC4 > 0 THEN
        (* Omplo la Sitja S4 *)
        LST4_VAL := LST4_VAL + 1;
    END_IF;
END_IF;

(* Miro si haig d'activar SS1 quan M7 activat *)
IF (PC_COMANDA_2 AND 1024) > 0 THEN
    (* Miro si M7 esta activat *)
    IF M7 > 0 THEN
        (* Activo SS1 *)
        SS1 := 1;
    ELSE
        (* Desactivo SS1 *)
        SS1 := 0;
    END_IF;
END_IF;

(* Miro si haig d'activar SS2 quan M8 activat *)
IF (PC_COMANDA_2 AND 2048) > 0 THEN
    (* Miro si M8 esta activat *)
    IF M8 > 0 THEN
        (* Activo SS2 *)
        SS2 := 1;
    ELSE
        (* Desactivo SS2 *)
        SS2 := 0;
    END_IF;
END_IF;

(* Miro si haig d'activar SS3 quan M9 activat *)
IF (PC_COMANDA_2 AND 4096) > 0 THEN
    (* Miro si M9 esta activat *)
    IF M9 > 0 THEN
        (* Activo SS3 *)
        SS3 := 1;
    ELSE
        (* Desactivo SS3 *)
        SS3 := 0;
    END_IF;
END_IF;

(* Miro si haig d'activar SS4 quan M10 activat *)
IF (PC_COMANDA_2 AND 8192) > 0 THEN
    (* Miro si M10 esta activat *)
    IF M10 > 0 THEN
        (* Activo SS4 *)
        SS4 := 1;
    ELSE
        (* Desactivo SS4 *)
        SS4 := 0;
    END_IF;
END_IF;

(* Miro si haig de buidar Sitja S1 quan M7 activat *)
IF (PC_COMANDA_2 AND 16384) > 0 THEN
    (* Miro si M7 esta activat *)
    IF M7 > 0 THEN
        (* Buido la Sitja S1 *)
        LST1_VAL := LST1_VAL - 1;
    END_IF;
END_IF;

(* Miro si haig de buidar Sitja S2 quan M8 activat *)
IF (PC_COMANDA_2 AND 32768) > 0 THEN
    (* Miro si M8 esta activat *)
    IF M8 > 0 THEN
        (* Buido la Sitja S2 *)
        LST2_VAL := LST2_VAL - 1;
    END_IF;
END_IF;

(* Miro si haig de buidar Sitja S3 quan M9 activat *)
IF (PC_COMANDA_3 AND 1) > 0 THEN
    (* Miro si M9 esta activat *)
    IF M9 > 0 THEN
        (* Buido la Sitja S3 *)
        LST3_VAL := LST3_VAL - 1;
    END_IF;
END_IF;

(* Miro si haig de buidar Sitja S4 quan M10 activat *)
IF (PC_COMANDA_3 AND 2) > 0 THEN
```

```
303     (* Miro si M10 esta activat *)
304     IF M10 > 0 THEN
305         (* Buido la Sitja S4 *)
306         LST4_VAL := LST4_VAL - 1;
307     END_IF;
308 END_IF;
309
310 (* Miro si haig d'omplir bascula quan M7 activat *)
311 IF (PC_COMANDA_3 AND 4) > 0 THEN
312     (* Miro si M7 esta activat *)
313     IF M7 > 0 THEN
314         (* Miro si ha passat prou temps (en segons) per incrementar la bascula *)
315         IF ((SysGetSysTime(TRUE) - INSTANT_PC3_4) >= 100000) THEN
316             (* Omplo la bascula BTA *)
317             BTA := BTA + 10;
318             (* Em guardo l'instant actual *)
319             INSTANT_PC3_4 := SysGetSysTime(TRUE);
320         END_IF;
321     END_IF;
322 END_IF;
323
324 (* Miro si haig d'omplir bascula quan M8 activat *)
325 IF (PC_COMANDA_3 AND 8) > 0 THEN
326     (* Miro si M8 esta activat *)
327     IF M8 > 0 THEN
328         (* Miro si ha passat prou temps (en segons) per incrementar la bascula *)
329         IF ((SysGetSysTime(TRUE) - INSTANT_PC3_8) >= 100000) THEN
330             (* Omplo la bascula BTA *)
331             BTA := BTA + 10;
332             (* Em guardo l'instant actual *)
333             INSTANT_PC3_8 := SysGetSysTime(TRUE);
334         END_IF;
335     END_IF;
336 END_IF;
337
338 (* Miro si haig d'omplir bascula quan M9 activat i EV0 activada *)
339 IF (PC_COMANDA_3 AND 16) > 0 THEN
340     (* Miro si M9 esta activat *)
341     IF M9 > 0 THEN
342         (* Miro si ha passat prou temps (en segons) per incrementar la bascula *)
343         IF ((SysGetSysTime(TRUE) - INSTANT_PC3_16) >= 100000) THEN
344             (* Miro si EV0 esta activada *)
345             IF EV0 > 0 THEN
346                 (* Omplo la bascula BTA *)
347                 BTA := BTA + 10;
348             ELSE
349                 (* Omplo la bascula BTB *)
350                 BTB := BTB + 10;
351             END_IF;
352             (* Em guardo l'instant actual *)
353             INSTANT_PC3_16 := SysGetSysTime(TRUE);
354         END_IF;
355     END_IF;
356 END_IF;
357
358 (* Miro si haig d'omplir bascula quan M10 activat *)
359 IF (PC_COMANDA_3 AND 32) > 0 THEN
360     (* Miro si M10 esta activat *)
361     IF M10 > 0 THEN
362         (* Miro si ha passat prou temps (en segons) per incrementar la bascula *)
363         IF ((SysGetSysTime(TRUE) - INSTANT_PC3_32) >= 100000) THEN
364             (* Omplo la bascula BTB *)
365             BTB := BTB + 10;
366             (* Em guardo l'instant actual *)
367             INSTANT_PC3_32 := SysGetSysTime(TRUE);
368         END_IF;
369     END_IF;
370 END_IF;
371
372 (* Miro si haig d'activar S2 3 segons despres que EV3 activat *)
373 IF (PC_COMANDA_3 AND 64) > 0 THEN
374     (* Miro si EV3 esta activada *)
375     IF EV3 > 0 THEN
376         (* Miro si ha passat prou temps (en segons) *)
377         IF ((SysGetSysTime(TRUE) - INSTANT_PC3_64) >= 3000000) THEN
378             (* Activo S2 *)
379             S2 := 1;
380         END_IF;
381     ELSE
382         (* Desactivo S2 *)
383         S2 := 0;
384         INSTANT_PC3_64 := SysGetSysTime(TRUE);
385     END_IF;
386 END_IF;
387
388 (* Miro si haig d'activar S3 3 segons despres que EV2 activat *)
389 IF (PC_COMANDA_3 AND 128) > 0 THEN
390     (* Miro si EV2 esta activada *)
391     IF EV2 > 0 THEN
392         (* Miro si ha passat prou temps (en segons) *)
393         IF ((SysGetSysTime(TRUE) - INSTANT_PC3_128) >= 3000000) THEN
394             (* Activo S3 *)
395             S3 := 1;
396         END_IF;
397     ELSE
398         (* Desactivo S3 *)
399         S3 := 0;
400         INSTANT_PC3_128 := SysGetSysTime(TRUE);
401     END_IF;
402 END_IF;
403
404 (* Miro si haig d'activar S4 3 segons despres que EV6 activat *)
405 IF (PC_COMANDA_3 AND 256) > 0 THEN
406     (* Miro si EV6 esta activada *)
407     IF EV6 > 0 THEN
408         (* Miro si ha passat prou temps (en segons) *)
409         IF ((SysGetSysTime(TRUE) - INSTANT_PC3_256) >= 3000000) THEN
410             (* Activo S4 *)
411             S4 := 1;
412         END_IF;
413     ELSE
414         (* Desactivo S4 *)
415         S4 := 0;
416         INSTANT_PC3_256 := SysGetSysTime(TRUE);
417     END_IF;
418 END_IF;
419
420 (* Miro si haig d'activar FC5 quan M19 i fabrico Producte1 *)
421 IF (PC_COMANDA_3 AND 512) > 0 THEN
422     (* Miro si M19 activat i fabrico Producte1 *)
423     IF M19 > 0 AND CONFIG_PROD_FABRICAR = 1 THEN
424         (* Activo FC5 *)
425         FC5 := 1;
426     ELSE
427         (* Desactivo FC5 *)
428         FC5 := 0;
429     END_IF;
430 END_IF;
431
432 (* Miro si haig d'activar FC6 quan M19 i fabrico Producte2 *)
433 IF (PC_COMANDA_3 AND 1024) > 0 THEN
434     (* Miro si M19 activat i fabrico Producte2 *)
435     IF M19 > 0 AND CONFIG_PROD_FABRICAR = 2 THEN
436         (* Activo FC6 *)
437         FC6 := 1;
438     ELSE
```

```
439         (* Desactivo FC6 *)
440         FC6 := 0;
441     END_IF;
442 END_IF;
443
444 (* Miro si haig d'activar FC7 quan M19 i fabrico Producte3 *)
445 IF (PC_COMANDA_3 AND 2048) > 0 THEN
446     (* Miro si M19 activat i fabrico Producte3 *)
447     IF M19 > 0 AND CONFIG_PROD_FABRICAR = 3 THEN
448         (* Activo FC7 *)
449         FC7 := 1;
450     ELSE
451         (* Desactivo FC7 *)
452         FC7 := 0;
453     END_IF;
454 END_IF;
455
456 (* Miro si haig d'omplir Sitja S5 quan EV7 *)
457 IF (PC_COMANDA_3 AND 4096) > 0 THEN
458     (* Miro si EV7 esta activat *)
459     IF EV7 > 0 THEN
460         (* Omplo la Sitja S5 *)
461         LST5_VAL := LST5_VAL + 1;
462     END_IF;
463 END_IF;
464
465 (* Miro si haig d'omplir Sitja S6 quan EV8 *)
466 IF (PC_COMANDA_3 AND 8192) > 0 THEN
467     (* Miro si EV8 esta activat *)
468     IF EV8 > 0 THEN
469         (* Omplo la Sitja S6 *)
470         LST6_VAL := LST6_VAL + 1;
471     END_IF;
472 END_IF;
473
474 (* Miro si haig d'omplir Sitja S7 quan EV9 *)
475 IF (PC_COMANDA_3 AND 16384) > 0 THEN
476     (* Miro si EV9 esta activat *)
477     IF EV9 > 0 THEN
478         (* Omplo la Sitja S7 *)
479         LST7_VAL := LST7_VAL + 1;
480     END_IF;
481 END_IF;
482
483 (* Miro si haig d'activar SS5 quan EV10 activat *)
484 IF (PC_COMANDA_3 AND 32768) > 0 THEN
485     (* Miro si EV10 esta activat *)
486     IF EV10 > 0 THEN
487         (* Activo SS5 *)
488         SS5 := 1;
489     END_IF;
490 END_IF;
491
492 (* Miro si haig d'activar SS6 quan EV11 activat *)
493 IF (PC_COMANDA_4 AND 1) > 0 THEN
494     (* Miro si EV11 esta activat *)
495     IF EV11 > 0 THEN
496         (* Activo SS6 *)
497         SS6 := 1;
498     END_IF;
499 END_IF;
500
501 (* Miro si haig d'activar SS7 quan EV12 activat *)
502 IF (PC_COMANDA_4 AND 2) > 0 THEN
503     (* Miro si EV12 esta activat *)
504     IF EV12 > 0 THEN
505         (* Activo SS7 *)
506         SS7 := 1;
507     END_IF;
508 END_IF;
509
510 (* Miro si haig de buidar Sitja S5 quan EV10 *)
511 IF (PC_COMANDA_4 AND 4) > 0 THEN
512     (* Miro si EV10 esta activat *)
513     IF EV10 > 0 THEN
514         (* Buido la Sitja S5 *)
515         LST5_VAL := LST5_VAL - 1;
516     END_IF;
517 END_IF;
518
519 (* Miro si haig de buidar Sitja S6 quan EV11 *)
520 IF (PC_COMANDA_4 AND 8) > 0 THEN
521     (* Miro si EV11 esta activat *)
522     IF EV11 > 0 THEN
523         (* Buido la Sitja S6 *)
524         LST6_VAL := LST6_VAL - 1;
525     END_IF;
526 END_IF;
527
528 (* Miro si haig de buidar Sitja S7 quan EV12 *)
529 IF (PC_COMANDA_4 AND 16) > 0 THEN
530     (* Miro si EV12 esta activat *)
531     IF EV12 > 0 THEN
532         (* Buido la Sitja S7 *)
533         LST7_VAL := LST7_VAL - 1;
534     END_IF;
535 END_IF;
536
537 (* Miro si haig d'activar IM1 quan M3 *)
538 IF (PC_COMANDA_4 AND 32) > 0 THEN
539     (* Miro si M3 esta activat *)
540     IF M3 > 0 THEN
541         (* Assigno un valor a IM1 *)
542         IM1 := 55;
543     ELSE
544         (* Reset IM1 *)
545         IM1 := 0;
546     END_IF;
547 END_IF;
548
549 (* Miro si haig d'activar IM2 quan M15 *)
550 IF (PC_COMANDA_4 AND 64) > 0 THEN
551     (* Miro si M15 esta activat *)
552     IF M15 > 0 THEN
553         (* Assigno un valor a IM2 *)
554         IM2 := 55;
555     ELSE
556         (* Reset IM2 *)
557         IM2 := 0;
558     END_IF;
559 END_IF;
560
561 (* Miro si haig d'activar IM3 quan M18 *)
562 IF (PC_COMANDA_4 AND 128) > 0 THEN
563     (* Miro si M18 esta activat *)
564     IF M18 > 0 THEN
565         (* Assigno un valor a IM3 *)
566         IM3 := 55;
567     ELSE
568         (* Reset IM3 *)
569         IM3 := 0;
570     END_IF;
571 END_IF;
572
573 (* Miro si haig d'activar IM4 quan M12 *)
574 IF (PC_COMANDA_4 AND 256) > 0 THEN
```

```
575     (* Miro si M12 esta activat *)
576     IF M12 > 0 THEN
577         (* Assigno un valor a IM4 *)
578         IM4 := 55;
579     ELSE
580         (* Reset IM4 *)
581         IM4 := 0;
582     END_IF;
583 END_IF;
584
585
```

```
VAR_EXTERNAL
ACK_M1 : WORD;
ACK_M2 : WORD;
ACK_M3 : WORD;
ACK_M4 : WORD;
ACK_M5 : WORD;
ACK_M6 : WORD;
ACK_M7 : WORD;
ACK_M8 : WORD;
ACK_M9 : WORD;
ACK_M10 : WORD;
ACK_M11 : WORD;
ACK_M12 : WORD;
ACK_M13 : WORD;
ACK_M14 : WORD;
ACK_M15 : WORD;
ACK_M16 : WORD;
ACK_M17 : WORD;
ACK_M18 : WORD;
ACK_M19 : WORD;
ACK_REVOLVER : WORD;
BTA : WORD;
BTB : WORD;
IM1 : WORD;
IM2 : WORD;
IM3 : WORD;
IM4 : WORD;
LST1_VAL : WORD;
LST2_VAL : WORD;
LST3_VAL : WORD;
LST4_VAL : WORD;
LST5_VAL : WORD;
LST6_VAL : WORD;
LST7_VAL : WORD;
PARADA_EMERGENCIA : WORD;
S0 : WORD;
S1 : WORD;
S2 : WORD;
S3 : WORD;
S4 : WORD;
SS1 : WORD;
SS2 : WORD;
SS3 : WORD;
SS4 : WORD;
SS5 : WORD;
SS6 : WORD;
SS7 : WORD;
SSA : WORD;
SSB : WORD;
SSM1 : WORD;
SSM2 : WORD;
SSM3 : WORD;
FC1 : WORD;
FC2 : WORD;
FC3 : WORD;
FC4 : WORD;
FC5 : WORD;
FC6 : WORD;
FC7 : WORD;
DI_ACK_M1 : BOOL; (* Confirmació Motor M1 *)
DI_ACK_M2 : BOOL; (* Confirmació Motor M2 *)
DI_ACK_M3 : BOOL; (* Confirmació Motor M3 *)
DI_ACK_M4 : BOOL; (* Confirmació Motor M4 *)
DI_ACK_M5 : BOOL; (* Confirmació Motor M5 *)
DI_ACK_M6 : BOOL; (* Confirmació Motor M6 *)
DI_ACK_M7 : BOOL; (* Confirmació Motor M7 *)
DI_ACK_M8 : BOOL; (* Confirmació Motor M8 *)
DI_ACK_M9 : BOOL; (* Confirmació Motor M9 *)
DI_ACK_M10 : BOOL; (* Confirmació Motor M10 *)
DI_ACK_M11 : BOOL; (* Confirmació Motor M11 *)
DI_ACK_M12 : BOOL; (* Confirmació Motor M12 *)
DI_ACK_M13 : BOOL; (* Confirmació Motor M13 *)
DI_ACK_M14 : BOOL; (* Confirmació Motor M14 *)
DI_ACK_M15 : BOOL; (* Confirmació Motor M15 *)
DI_ACK_M16 : BOOL; (* Confirmació Motor M16 *)
DI_ACK_M17 : BOOL; (* Confirmació Motor M17 *)
DI_ACK_M18 : BOOL; (* Confirmació Motor M18 *)
DI_ACK_M19 : BOOL; (* Confirmació Motor M19 *)
DI_ACK_REVOLVER : BOOL; (* Confirmació Motor REVOLVER *)
DI_FC1 : BOOL; (* Final de cursa Revolver per omplir Sitja S1 *)
DI_FC2 : BOOL; (* Final de cursa Revolver per omplir Sitja S2 *)
DI_FC3 : BOOL; (* Final de cursa Revolver per omplir Sitja S3 *)
DI_FC4 : BOOL; (* Final de cursa Revolver per omplir Sitja S4 *)
DI_FC5 : BOOL; (* Final de cursa per omplir Sitja S5 *)
DI_FC6 : BOOL; (* Final de cursa per omplir Sitja S6 *)
DI_FC7 : BOOL; (* Final de cursa per omplir Sitja S7 *)
DI_PARADA_EMERGENCIA : BOOL; (* Pulsador de Parada emergència *)
DI_S0 : BOOL; (* Sensor presència producte S0 *)
DI_S1 : BOOL; (* Sensor presència producte S1 *)
DI_S2 : BOOL; (* Sensor presència producte S2 *)
DI_S3 : BOOL; (* Sensor presència producte S3 *)
DI_S4 : BOOL; (* Sensor presència producte S4 *)
DI_SS1 : BOOL; (* Sensor presència producte Sitja S1 *)
DI_SS2 : BOOL; (* Sensor presència producte Sitja S2 *)
DI_SS3 : BOOL; (* Sensor presència producte Sitja S3 *)
DI_SS4 : BOOL; (* Sensor presència producte Sitja S4 *)
DI_SS5 : BOOL; (* Sensor presència producte Sitja S5 *)
DI_SS6 : BOOL; (* Sensor presència producte Sitja S6 *)
DI_SS7 : BOOL; (* Sensor presència producte Sitja S7 *)
DI_SSA : BOOL; (* Sensor presència producte Tramuja A *)
DI_SSB : BOOL; (* Sensor presència producte Tramuja B *)
DI_SSM1 : BOOL; (* Sensor presència producte entrada Mescladora *)
DI_SSM2 : BOOL; (* Sensor presència producte Mescladora *)
DI_SSM3 : BOOL; (* Sensor presència producte sortida Mescladora *)
END_VAR

VAR
BTA_SysGetAnInp : SysGetAnInp;
BTB_SysGetAnInp : SysGetAnInp;
LST1_SysGetAnInp : SysGetAnInp;
LST2_SysGetAnInp : SysGetAnInp;
LST3_SysGetAnInp : SysGetAnInp;
LST4_SysGetAnInp : SysGetAnInp;
LST5_SysGetAnInp : SysGetAnInp;
LST6_SysGetAnInp : SysGetAnInp;
LST7_SysGetAnInp : SysGetAnInp;
IM1_SysGetAnInp : SysGetAnInp;
IM2_SysGetAnInp : SysGetAnInp;
IM3_SysGetAnInp : SysGetAnInp;
IM4_SysGetAnInp : SysGetAnInp;
OK_BTA : BOOL;
ERR_BTA : BOOL;
BTA_ANA : REAL;
OK_BTB : BOOL;
ERR_BTB : BOOL;
BTB_ANA : REAL;
OK_LST1 : BOOL;
ERR_LST1 : BOOL;
LST1_ANA : REAL;
OK_LST2 : BOOL;
ERR_LST2 : BOOL;
LST2_ANA : REAL;
OK_LST3 : BOOL;
ERR_LST3 : BOOL;
```

```
      LST3_ANA : REAL;
      OK_LST4 : BOOL;
      ERR_LST4 : BOOL;
      LST4_ANA : REAL;
      OK_LST5 : BOOL;
      ERR_LST5 : BOOL;
      LST5_ANA : REAL;
      OK_LST6 : BOOL;
      ERR_LST6 : BOOL;
      LST6_ANA : REAL;
      OK_LST7 : BOOL;
      ERR_LST7 : BOOL;
      LST7_ANA : REAL;
      OK_IM1 : BOOL;
      ERR_IM1 : BOOL;
      IM1_ANA : REAL;
      OK_IM2 : BOOL;
      ERR_IM2 : BOOL;
      IM2_ANA : REAL;
      OK_IM3 : BOOL;
      ERR_IM3 : BOOL;
      IM3_ANA : REAL;
      OK_IM4 : BOOL;
      ERR_IM4 : BOOL;
      IM4_ANA : REAL;
      END_VAR
```

```
 1
 2 (* ----------------- *)
 3 (* ENTRADES DIGITALS *)
 4 (* ----------------- *)
 5
 6 (* Confirmacio Motor M1 *)
 7 IF DI_ACK_M1 THEN
 8     ACK_M1 := 1;
 9 ELSE
10     ACK_M1 := 0;
11 END_IF;
12
13 (* Confirmacio Motor M2 *)
14 IF DI_ACK_M2 THEN
15     ACK_M2 := 1;
16 ELSE
17     ACK_M2 := 0;
18 END_IF;
19
20 (* Confirmacio Motor M3 *)
21 IF DI_ACK_M3 THEN
22     ACK_M3 := 1;
23 ELSE
24     ACK_M3 := 0;
25 END_IF;
26
27 (* Confirmacio Motor M4 *)
28 IF DI_ACK_M4 THEN
29     ACK_M4 := 1;
30 ELSE
31     ACK_M4 := 0;
32 END_IF;
33
34 (* Confirmacio Motor M5 *)
35 IF DI_ACK_M5 THEN
36     ACK_M5 := 1;
37 ELSE
38     ACK_M5 := 0;
39 END_IF;
40
41 (* Confirmacio Motor M6 *)
42 IF DI_ACK_M6 THEN
43     ACK_M6 := 1;
44 ELSE
45     ACK_M6 := 0;
46 END_IF;
47
48 (* Confirmacio Motor M7 *)
49 IF DI_ACK_M7 THEN
50     ACK_M7 := 1;
51 ELSE
52     ACK_M7 := 0;
53 END_IF;
54
55 (* Confirmacio Motor M8 *)
56 IF DI_ACK_M8 THEN
57     ACK_M8 := 1;
58 ELSE
59     ACK_M8 := 0;
60 END_IF;
61
62 (* Confirmacio Motor M9 *)
63 IF DI_ACK_M9 THEN
64     ACK_M9 := 1;
65 ELSE
66     ACK_M9 := 0;
67 END_IF;
68
69 (* Confirmacio Motor M10 *)
70 IF DI_ACK_M10 THEN
71     ACK_M10 := 1;
72 ELSE
73     ACK_M10 := 0;
74 END_IF;
75
76 (* Confirmacio Motor M11 *)
77 IF DI_ACK_M11 THEN
78     ACK_M11 := 1;
79 ELSE
80     ACK_M11 := 0;
81 END_IF;
82
83 (* Confirmacio Motor M12 *)
84 IF DI_ACK_M12 THEN
85     ACK_M12 := 1;
86 ELSE
87     ACK_M12 := 0;
88 END_IF;
89
90 (* Confirmacio Motor M13 *)
91 IF DI_ACK_M13 THEN
92     ACK_M13 := 1;
93 ELSE
94     ACK_M13 := 0;
95 END_IF;
96
97 (* Confirmacio Motor M14 *)
98 IF DI_ACK_M14 THEN
99     ACK_M14 := 1;
100 ELSE
101     ACK_M14 := 0;
102 END_IF;
103
104 (* Confirmacio Motor M15 *)
105 IF DI_ACK_M15 THEN
106     ACK_M15 := 1;
107 ELSE
108     ACK_M15 := 0;
```

```
END_IF;

(* Confirmacio Motor M16 *)
IF DI_ACK_M16 THEN
    ACK_M16 := 1;
ELSE
    ACK_M16 := 0;
END_IF;

(* Confirmacio Motor M17 *)
IF DI_ACK_M17 THEN
    ACK_M17 := 1;
ELSE
    ACK_M17 := 0;
END_IF;

(* Confirmacio Motor M18 *)
IF DI_ACK_M18 THEN
    ACK_M18 := 1;
ELSE
    ACK_M18 := 0;
END_IF;

(* Confirmacio Motor M19 *)
IF DI_ACK_M19 THEN
    ACK_M19 := 1;
ELSE
    ACK_M19 := 0;
END_IF;

(* Confirmacio Motor REVOLVER *)
IF DI_ACK_REVOLVER THEN
    ACK_REVOLVER := 1;
ELSE
    ACK_REVOLVER := 0;
END_IF;

(* Pulsador Parada emergencia *)
IF DI_PARADA_EMERGENCIA THEN
    PARADA_EMERGENCIA := 1;
ELSE
    PARADA_EMERGENCIA := 0;
END_IF;

(* Sensor presencia producte S0 *)
IF DI_S0 THEN
    S0 := 1;
ELSE
    S0 := 0;
END_IF;

(* Sensor presencia producte S1 *)
IF DI_S1 THEN
    S1 := 1;
ELSE
    S1 := 0;
END_IF;

(* Sensor presencia producte S2 *)
IF DI_S2 THEN
    S2 := 1;
ELSE
    S2 := 0;
END_IF;

(* Sensor presencia producte S3 *)
IF DI_S3 THEN
    S3 := 1;
ELSE
    S3 := 0;
END_IF;

(* Sensor presencia producte S4 *)
IF DI_S4 THEN
    S4 := 1;
ELSE
    S4 := 0;
END_IF;

(* Sensor presencia producte Sitja S1 *)
IF DI_SS1 THEN
    SS1 := 1;
ELSE
    SS1 := 0;
END_IF;

(* Sensor presencia producte Sitja S2 *)
IF DI_SS2 THEN
    SS2 := 1;
ELSE
    SS2 := 0;
END_IF;

(* Sensor presencia producte Sitja S3 *)
IF DI_SS3 THEN
    SS3 := 1;
ELSE
    SS3 := 0;
END_IF;

(* Sensor presencia producte Sitja S4 *)
IF DI_SS4 THEN
    SS4 := 1;
ELSE
    SS4 := 0;
END_IF;

(* Sensor presencia producte Sitja S5 *)
IF DI_SS5 THEN
    SS5 := 1;
ELSE
    SS5 := 0;
END_IF;

(* Sensor presencia producte Sitja S6 *)
IF DI_SS6 THEN
    SS6 := 1;
ELSE
    SS6 := 0;
END_IF;

(* Sensor presencia producte Sitja S7 *)
IF DI_SS7 THEN
    SS7 := 1;
ELSE
    SS7 := 0;
END_IF;

(* Sensor presencia producte Tramuja A *)
IF DI_SSA THEN
    SSA := 1;
ELSE
    SSA := 0;
END_IF;

(* Sensor presencia producte Tramuja B *)
```

```
IF DI_SSB THEN
    SSB := 1;
ELSE
    SSB := 0;
END_IF;

(* Sensor presencia producte entrada Mescladora *)
IF DI_SSM1 THEN
    SSM1 := 1;
ELSE
    SSM1 := 0;
END_IF;

(* Sensor presencia producte Mescladora *)
IF DI_SSM2 THEN
    SSM2 := 1;
ELSE
    SSM2 := 0;
END_IF;

(* Sensor presencia producte sortida Mescladora *)
IF DI_SSM3 THEN
    SSM3 := 1;
ELSE
    SSM3 := 0;
END_IF;

(* Final de cursa Revolver per omplir Sitja S1 *)
IF DI_FC1 THEN
    FC1 := 1;
ELSE
    FC1 := 0;
END_IF;

(* Final de cursa Revolver per omplir Sitja S2 *)
IF DI_FC2 THEN
    FC2 := 1;
ELSE
    FC2 := 0;
END_IF;

(* Final de cursa Revolver per omplir Sitja S3 *)
IF DI_FC3 THEN
    FC3 := 1;
ELSE
    FC3 := 0;
END_IF;

(* Final de cursa Revolver per omplir Sitja S4 *)
IF DI_FC4 THEN
    FC4 := 1;
ELSE
    FC4 := 0;
END_IF;

(* Final de cursa Revolver per omplir Sitja S5 *)
IF DI_FC5 THEN
    FC5 := 1;
ELSE
    FC5 := 0;
END_IF;

(* Final de cursa Revolver per omplir Sitja S6 *)
IF DI_FC6 THEN
    FC6 := 1;
ELSE
    FC6 := 0;
END_IF;

(* Final de cursa Revolver per omplir Sitja S7 *)
IF DI_FC7 THEN
    FC7 := 1;
ELSE
    FC7 := 0;
END_IF;

(* -------------------- *)
(* ENTRADES ANALOGIQUES *)
(* -------------------- *)
(* Llegeixo l'entrada analogica de la bascula BTA: en el el 1er canal del modul 2, mode 4-20mA  *)
BTA_SysGetAnInp(Address:=2, Channel:=0, Mode:=AD_CURR_4_20_COMMON);
OK_BTA := BTA_SysGetAnInp.Done;
ERR_BTA := BTA_SysGetAnInp.Fault;
BTA_ANA := BTA_SysGetAnInp.Value;
(* Miro si la lectura ha estat correcta *)
IF OK_BTA THEN
    (* Converteixo la lectura a Kg de producte 0-20mA -> 0-2500Kg *)
    BTA := 125 * TO_UINT(BTA_ANA);
ELSE
    (* Reset del valor de la bascula BTA *)
    BTA := 0;
END_IF;

(* Llegeixo l'entrada analogica de la bascula BTB: en el el 2on canal del modul 2, mode 4-20mA  *)
BTB_SysGetAnInp(Address:=2, Channel:=1, Mode:=AD_CURR_4_20_COMMON);
OK_BTB := BTB_SysGetAnInp.Done;
ERR_BTB := BTB_SysGetAnInp.Fault;
BTB_ANA := BTB_SysGetAnInp.Value;
(* Miro si la lectura ha estat correcta *)
IF OK_BTB THEN
    (* Converteixo la lectura a Kg de producte 0-20mA -> 0-2500Kg *)
    BTB := 125 * TO_UINT(BTB_ANA);
ELSE
    (* Reset del valor de la bascula BTB *)
    BTB := 0;
END_IF;

(* Llegeixo l'entrada analogica de nivell de la Sitja S1: en el el 3er canal del modul 2, mode 4-20mA  *)
LST1_SysGetAnInp(Address:=2, Channel:=2, Mode:=AD_CURR_4_20_COMMON);
OK_LST1 := LST1_SysGetAnInp.Done;
ERR_LST1 := LST1_SysGetAnInp.Fault;
LST1_ANA := LST1_SysGetAnInp.Value;
(* Miro si la lectura ha estat correcta *)
IF OK_LST1 THEN
    (* Converteixo la lectura a valor digital entre 0-65535 *)
    LST1_VAL := 3276 * TO_UINT(LST1_ANA);
ELSE
    (* Reset del valor de nivell de Sitja S1 *)
    LST1_VAL := 0;
END_IF;

(* Llegeixo l'entrada analogica de nivell de la Sitja S2: en el el 4rt canal del modul 2, mode 4-20mA  *)
LST2_SysGetAnInp(Address:=2, Channel:=3, Mode:=AD_CURR_4_20_COMMON);
OK_LST2 := LST2_SysGetAnInp.Done;
ERR_LST2 := LST2_SysGetAnInp.Fault;
LST2_ANA := LST2_SysGetAnInp.Value;
(* Miro si la lectura ha estat correcta *)
IF OK_LST2 THEN
    (* Converteixo la lectura a valor digital entre 0-65535 *)
    LST2_VAL := 3276 * TO_UINT(LST2_ANA);
ELSE
    (* Reset del valor de nivell de Sitja S2 *)
    LST2_VAL := 0;
END_IF;

(* Llegeixo l'entrada analogica de nivell de la Sitja S3: en el el 5e canal del modul 2, mode 4-20mA  *)
```

```
381 LST3_SysGetAnInp(Address:=2, Channel:=4, Mode:=AD_CURR_4_20_COMMON);
382 OK_LST3 := LST3_SysGetAnInp.Done;
383 ERR_LST3 := LST3_SysGetAnInp.Fault;
384 LST3_ANA := LST3_SysGetAnInp.Value;
385 (* Miro si la lectura ha estat correcta *)
386 IF OK_LST3 THEN
387     (* Converteixo la lectura a valor digital entre 0-65535 *)
388     LST3_VAL := 3276 * TO_UINT(LST3_ANA);
389 ELSE
390     (* Reset del valor de nivell de Sitja S3 *)
391     LST3_VAL := 0;
392 END_IF;
393
394 (* Llegeixo l'entrada analogica de nivell de la Sitja S4: en el el 6e canal del modul 2, mode 4-20mA  *)
395 LST4_SysGetAnInp(Address:=2, Channel:=5, Mode:=AD_CURR_4_20_COMMON);
396 OK_LST4 := LST4_SysGetAnInp.Done;
397 ERR_LST4 := LST4_SysGetAnInp.Fault;
398 LST4_ANA := LST4_SysGetAnInp.Value;
399 (* Miro si la lectura ha estat correcta *)
400 IF OK_LST4 THEN
401     (* Converteixo la lectura a valor digital entre 0-65535 *)
402     LST4_VAL := 3276 * TO_UINT(LST4_ANA);
403 ELSE
404     (* Reset del valor de nivell de Sitja S4 *)
405     LST4_VAL := 0;
406 END_IF;
407
408 (* Llegeixo l'entrada analogica de nivell de la Sitja S5: en el el 7e canal del modul 2, mode 4-20mA  *)
409 LST5_SysGetAnInp(Address:=2, Channel:=6, Mode:=AD_CURR_4_20_COMMON);
410 OK_LST5 := LST5_SysGetAnInp.Done;
411 ERR_LST5 := LST5_SysGetAnInp.Fault;
412 LST5_ANA := LST5_SysGetAnInp.Value;
413 (* Miro si la lectura ha estat correcta *)
414 IF OK_LST5 THEN
415     (* Converteixo la lectura a valor digital entre 0-65535 *)
416     LST5_VAL := 3276 * TO_UINT(LST5_ANA);
417 ELSE
418     (* Reset del valor de nivell de Sitja S5 *)
419     LST5_VAL := 0;
420 END_IF;
421
422 (* Llegeixo l'entrada analogica de nivell de la Sitja S6: en el el 8e canal del modul 2, mode 4-20mA  *)
423 LST6_SysGetAnInp(Address:=2, Channel:=7, Mode:=AD_CURR_4_20_COMMON);
424 OK_LST6 := LST6_SysGetAnInp.Done;
425 ERR_LST6 := LST6_SysGetAnInp.Fault;
426 LST6_ANA := LST6_SysGetAnInp.Value;
427 (* Miro si la lectura ha estat correcta *)
428 IF OK_LST6 THEN
429     (* Converteixo la lectura a valor digital entre 0-65535 *)
430     LST6_VAL := 3276 * TO_UINT(LST6_ANA);
431 ELSE
432     (* Reset del valor de nivell de Sitja S6 *)
433     LST6_VAL := 0;
434 END_IF;
435
436 (* Llegeixo l'entrada analogica de nivell de la Sitja S7: en el el 9e canal del modul 2, mode 4-20mA  *)
437 LST7_SysGetAnInp(Address:=2, Channel:=8, Mode:=AD_CURR_4_20_COMMON);
438 OK_LST7 := LST7_SysGetAnInp.Done;
439 ERR_LST7 := LST7_SysGetAnInp.Fault;
440 LST7_ANA := LST7_SysGetAnInp.Value;
441 (* Miro si la lectura ha estat correcta *)
442 IF OK_LST7 THEN
443     (* Converteixo la lectura a valor digital entre 0-65535 *)
444     LST7_VAL := 3276 * TO_UINT(LST7_ANA);
445 ELSE
446     (* Reset del valor de nivell de Sitja S7 *)
447     LST7_VAL := 0;
448 END_IF;
449
450 (* Llegeixo l'entrada analogica de l'elevador 1: en el el 1er canal del modul 3, mode 4-20mA  *)
451 IM1_SysGetAnInp(Address:=3, Channel:=0, Mode:=AD_CURR_4_20_COMMON);
452 OK_IM1 := IM1_SysGetAnInp.Done;
453 ERR_IM1 := IM1_SysGetAnInp.Fault;
454 IM1_ANA := IM1_SysGetAnInp.Value;
455 (* Miro si la lectura ha estat correcta *)
456 IF OK_IM1 THEN
457     (* Converteixo la lectura a valor entre 0-100% *)
458     IM1 := 4 * TO_UINT(IM1_ANA);
459 ELSE
460     (* Reset del valor *)
461     IM1 := 0;
462 END_IF;
463
464 (* Llegeixo l'entrada analogica de l'elevador 2: en el el 2on canal del modul 3, mode 4-20mA  *)
465 IM2_SysGetAnInp(Address:=3, Channel:=1, Mode:=AD_CURR_4_20_COMMON);
466 OK_IM2 := IM2_SysGetAnInp.Done;
467 ERR_IM2 := IM2_SysGetAnInp.Fault;
468 IM2_ANA := IM2_SysGetAnInp.Value;
469 (* Miro si la lectura ha estat correcta *)
470 IF OK_IM2 THEN
471     (* Converteixo la lectura a valor entre 0-100% *)
472     IM2 := 4 * TO_UINT(IM2_ANA);
473 ELSE
474     (* Reset del valor *)
475     IM2 := 0;
476 END_IF;
477
478 (* Llegeixo l'entrada analogica de l'elevador 3: en el el 3on canal del modul 3, mode 4-20mA  *)
479 IM3_SysGetAnInp(Address:=3, Channel:=2, Mode:=AD_CURR_4_20_COMMON);
480 OK_IM3 := IM3_SysGetAnInp.Done;
481 ERR_IM3 := IM3_SysGetAnInp.Fault;
482 IM3_ANA := IM3_SysGetAnInp.Value;
483 (* Miro si la lectura ha estat correcta *)
484 IF OK_IM3 THEN
485     (* Converteixo la lectura a valor entre 0-100% *)
486     IM3 := 4 * TO_UINT(IM3_ANA);
487 ELSE
488     (* Reset del valor *)
489     IM3 := 0;
490 END_IF;
491
492 (* Llegeixo l'entrada analogica de l'elevador 4: en el el 4rt canal del modul 3, mode 4-20mA  *)
493 IM4_SysGetAnInp(Address:=3, Channel:=3, Mode:=AD_CURR_4_20_COMMON);
494 OK_IM4 := IM4_SysGetAnInp.Done;
495 ERR_IM4 := IM4_SysGetAnInp.Fault;
496 IM4_ANA := IM4_SysGetAnInp.Value;
497 (* Miro si la lectura ha estat correcta *)
498 IF OK_IM4 THEN
499     (* Converteixo la lectura a valor entre 0-100% *)
500     IM4 := 4 * TO_UINT(IM4_ANA);
501 ELSE
502     (* Reset del valor *)
503     IM4 := 0;
504 END_IF;
505
506
```

```
VAR_EXTERNAL
DO_EV0 : BOOL; (* Electrovàlvula EV0 *)
DO_EV1 : BOOL; (* Electrovàlvula EV1 *)
DO_EV2 : BOOL; (* Electrovàlvula EV2 *)
DO_EV3 : BOOL; (* Electrovàlvula EV3 *)
DO_EV4 : BOOL; (* Electrovàlvula EV4 *)
DO_EV5 : BOOL; (* Electrovàlvula EV5 *)
DO_EV6 : BOOL; (* Electrovàlvula EV6 *)
DO_EV7 : BOOL; (* Electrovàlvula EV7 *)
DO_EV8 : BOOL; (* Electrovàlvula EV8 *)
DO_EV9 : BOOL; (* Electrovàlvula EV9 *)
DO_EV10 : BOOL; (* Electrovàlvula EV10 *)
DO_EV11 : BOOL; (* Electrovàlvula EV11 *)
DO_EV12 : BOOL; (* Electrovàlvula EV12 *)
DO_H1 : BOOL; (* Senyal lluminosa entrada pel camió H1 *)
DO_M1 : BOOL; (* Motor M1 *)
DO_M2 : BOOL; (* Motor M2 *)
DO_M3 : BOOL; (* Motor M3 *)
DO_M4 : BOOL; (* Motor M4 *)
DO_M5 : BOOL; (* Motor M5 *)
DO_M6 : BOOL; (* Motor M6 *)
DO_M7 : BOOL; (* Motor M7 *)
DO_M8 : BOOL; (* Motor M8 *)
DO_M9 : BOOL; (* Motor M9 *)
DO_M10 : BOOL; (* Motor M10 *)
DO_M11 : BOOL; (* Motor M11 *)
DO_M12 : BOOL; (* Motor M12 *)
DO_M13 : BOOL; (* Motor M13 *)
DO_M14 : BOOL; (* Motor M14 *)
DO_M15 : BOOL; (* Motor M15 *)
DO_M16 : BOOL; (* Motor M16 *)
DO_M17 : BOOL; (* Motor M17 *)
DO_M18 : BOOL; (* Motor M18 *)
DO_M19 : BOOL; (* Motor M19 *)
DO_REVOLVER : BOOL; (* Motor REVOLVER *)
EV0 : WORD;
EV1 : WORD;
EV2 : WORD;
EV3 : WORD;
EV4 : WORD;
EV5 : WORD;
EV6 : WORD;
EV7 : WORD;
EV8 : WORD;
EV9 : WORD;
EV10 : WORD;
EV11 : WORD;
EV12 : WORD;
H1 : WORD;
M1 : WORD;
M2 : WORD;
M3 : WORD;
M4 : WORD;
M5 : WORD;
M6 : WORD;
M7 : WORD;
M8 : WORD;
M9 : WORD;
M10 : WORD;
M11 : WORD;
M12 : WORD;
M13 : WORD;
M14 : WORD;
M15 : WORD;
M16 : WORD;
M17 : WORD;
M18 : WORD;
M19 : WORD;
REVOLVER : WORD;
VARIADOR_M11_PRC : WORD;
VARIADOR_M14_PRC : WORD;
END_VAR

VAR
VariadorM11_SysSetAnInp : SysSetAnOut;
VariadorM11_ANA : REAL;
OK_VariadorM11 : BOOL;
ERR_VariadorM11 : BOOL;
VariadorM14_ANA : REAL;
OK_VariadorM14 : BOOL;
ERR_VariadorM14 : BOOL;
END_VAR
```

```
 1
 2 (* ----------------- *)
 3 (* SORTIDES DIGITALS *)
 4 (* ----------------- *)
 5
 6 (* Electrovalvula EV0 *)
 7 IF EV0 > 0 THEN
 8     DO_EV0 := TRUE;
 9 ELSE
10     DO_EV0 := FALSE;
11 END_IF;
12
13 (* Electrovalvula EV1 *)
14 IF EV1 > 0 THEN
15     DO_EV1 := TRUE;
16 ELSE
17     DO_EV1 := FALSE;
18 END_IF;
19
20 (* Electrovalvula EV2 *)
21 IF EV2 > 0 THEN
22     DO_EV2 := TRUE;
23 ELSE
24     DO_EV2 := FALSE;
25 END_IF;
26
27 (* Electrovalvula EV3 *)
28 IF EV3 > 0 THEN
29     DO_EV3 := TRUE;
30 ELSE
31     DO_EV3 := FALSE;
32 END_IF;
33
34 (* Electrovalvula EV4 *)
35 IF EV4 > 0 THEN
36     DO_EV4 := TRUE;
37 ELSE
38     DO_EV4 := FALSE;
39 END_IF;
40
41 (* Electrovalvula EV5 *)
42 IF EV5 > 0 THEN
43     DO_EV5 := TRUE;
44 ELSE
45     DO_EV5 := FALSE;
46 END_IF;
47
48 (* Electrovalvula EV6 *)
49 IF EV6 > 0 THEN
50     DO_EV6 := TRUE;
```

```
ELSE
    DO_EV6 := FALSE;
END_IF;

(* Electrovalvula EV7 *)
IF EV7 > 0 THEN
    DO_EV7 := TRUE;
ELSE
    DO_EV7 := FALSE;
END_IF;

(* Electrovalvula EV8 *)
IF EV8 > 0 THEN
    DO_EV8 := TRUE;
ELSE
    DO_EV8 := FALSE;
END_IF;

(* Electrovalvula EV9 *)
IF EV9 > 0 THEN
    DO_EV9 := TRUE;
ELSE
    DO_EV9 := FALSE;
END_IF;

(* Electrovalvula EV10 *)
IF EV10 > 0 THEN
    DO_EV10 := TRUE;
ELSE
    DO_EV10 := FALSE;
END_IF;

(* Electrovalvula EV11 *)
IF EV11 > 0 THEN
    DO_EV11 := TRUE;
ELSE
    DO_EV11 := FALSE;
END_IF;

(* Electrovalvula EV12 *)
IF EV12 > 0 THEN
    DO_EV12 := TRUE;
ELSE
    DO_EV12 := FALSE;
END_IF;

(* Sortida lluminosa H1 *)
IF H1 > 0 THEN
    DO_H1 := TRUE;
ELSE
    DO_H1 := FALSE;
END_IF;

(* Motor M1 *)
IF M1 > 0 THEN
    DO_M1 := TRUE;
ELSE
    DO_M1 := FALSE;
END_IF;

(* Motor M2 *)
IF M2 > 0 THEN
    DO_M2 := TRUE;
ELSE
    DO_M2 := FALSE;
END_IF;

(* Motor M3 *)
IF M3 > 0 THEN
    DO_M3 := TRUE;
ELSE
    DO_M3 := FALSE;
END_IF;

(* Motor M4 *)
IF M4 > 0 THEN
    DO_M4 := TRUE;
ELSE
    DO_M4 := FALSE;
END_IF;

(* Motor M5 *)
IF M5 > 0 THEN
    DO_M5 := TRUE;
ELSE
    DO_M5 := FALSE;
END_IF;

(* Motor M6 *)
IF M6 > 0 THEN
    DO_M6 := TRUE;
ELSE
    DO_M6 := FALSE;
END_IF;

(* Motor M7 *)
IF M7 > 0 THEN
    DO_M7 := TRUE;
ELSE
    DO_M7 := FALSE;
END_IF;

(* Motor M8 *)
IF M8 > 0 THEN
    DO_M8 := TRUE;
ELSE
    DO_M8 := FALSE;
END_IF;

(* Motor M9 *)
IF M9 > 0 THEN
    DO_M9 := TRUE;
ELSE
    DO_M9 := FALSE;
END_IF;

(* Motor M10 *)
IF M10 > 0 THEN
    DO_M10 := TRUE;
ELSE
    DO_M10 := FALSE;
END_IF;

(* Motor M11 *)
IF M11 > 0 THEN
    DO_M11 := TRUE;
ELSE
    DO_M11 := FALSE;
END_IF;

(* Motor M12 *)
IF M12 > 0 THEN
    DO_M12 := TRUE;
ELSE
    DO_M12 := FALSE;
END_IF;
```

```
187
188 (* Motor M13 *)
189 IF M13 > 0 THEN
190     DO_M13 := TRUE;
191 ELSE
192     DO_M13 := FALSE;
193 END_IF;
194
195 (* Motor M14 *)
196 IF M14 > 0 THEN
197     DO_M14 := TRUE;
198 ELSE
199     DO_M14 := FALSE;
200 END_IF;
201
202 (* Motor M15 *)
203 IF M15 > 0 THEN
204     DO_M15 := TRUE;
205 ELSE
206     DO_M15 := FALSE;
207 END_IF;
208
209 (* Motor M16 *)
210 IF M16 > 0 THEN
211     DO_M16 := TRUE;
212 ELSE
213     DO_M16 := FALSE;
214 END_IF;
215
216 (* Motor M17 *)
217 IF M17 > 0 THEN
218     DO_M17 := TRUE;
219 ELSE
220     DO_M17 := FALSE;
221 END_IF;
222
223 (* Motor M18 *)
224 IF M18 > 0 THEN
225     DO_M18 := TRUE;
226 ELSE
227     DO_M18 := FALSE;
228 END_IF;
229
230 (* Motor M19 *)
231 IF M19 > 0 THEN
232     DO_M19 := TRUE;
233 ELSE
234     DO_M19 := FALSE;
235 END_IF;
236
237 (* Motor REVOLVER *)
238 IF REVOLVER > 0 THEN
239     DO_REVOLVER := TRUE;
240 ELSE
241     DO_REVOLVER := FALSE;
242 END_IF;
243
244 (* -------------------- *)
245 (* SORTIDES ANALÒGIQUES *)
246 (* -------------------- *)
247
248 (* Calculo el valor analogic que haig d'enviar al variador del motor M11 entre 0-10 *)
249 VariadorM11_ANA := TO_REAL(VARIADOR_M11_PRC / 10);
250 (* Assigno la sortida analogica del variador del motor M11 d'entrada al Molí: en el el 1er canal del modul 3, mode 0-10V  *)
251 VariadorM11_SysSetAnInp.Value := VariadorM11_ANA;
252 VariadorM11_SysSetAnInp(Address:=3, Channel:=0, Mode:=DA_VOLT_0_10);
253 OK_VariadorM11 := VariadorM11_SysSetAnInp.Done;
254 ERR_VariadorM11 := VariadorM11_SysSetAnInp.Fault;
255
256 (* Calculo el valor analogic que haig d'enviar al variador del motor M14 entre 0-10 *)
257 VariadorM14_ANA := TO_REAL(VARIADOR_M14_PRC / 10);
258 (* Assigno la sortida analogica del variador del motor M14 de sortida del Molí: en el el 2on canal del modul 3, mode 0-10V  *)
259 VariadorM14_SysSetAnInp.Value := VariadorM11_ANA;
260 VariadorM14_SysSetAnInp(Address:=3, Channel:=1, Mode:=DA_VOLT_0_10);
261 OK_VariadorM14 := VariadorM14_SysSetAnInp.Done;
262 ERR_VariadorM14 := VariadorM14_SysSetAnInp.Fault;
263
264
```

FUNCTION_BLOCK ComprovarSitges

```
VAR_EXTERNAL
LST1_MAX : WORD;
LST1_MIN : WORD;
LST2_MAX : WORD;
LST2_MIN : WORD;
LST3_MAX : WORD;
LST3_MIN : WORD;
LST4_MAX : WORD;
LST4_MIN : WORD;
LST5_MAX : WORD;
LST5_MIN : WORD;
LST6_MAX : WORD;
LST6_MIN : WORD;
LST7_MAX : WORD;
LST7_MIN : WORD;
LST1_PRC : WORD;
LST1_VAL : WORD;
LST2_PRC : WORD;
LST2_VAL : WORD;
LST3_PRC : WORD;
LST3_VAL : WORD;
LST4_PRC : WORD;
LST4_VAL : WORD;
LST5_PRC : WORD;
LST5_VAL : WORD;
LST6_PRC : WORD;
LST6_VAL : WORD;
LST7_PRC : WORD;
LST7_VAL : WORD;
ESTAT_SITJA_S1 : WORD;
ESTAT_SITJA_S2 : WORD;
ESTAT_SITJA_S3 : WORD;
ESTAT_SITJA_S4 : WORD;
ESTAT_SITJA_S5 : WORD;
ESTAT_SITJA_S6 : WORD;
ESTAT_SITJA_S7 : WORD;
QUANT_PROD_SITJA_S1 : WORD;
QUANT_PROD_SITJA_S2 : WORD;
QUANT_PROD_SITJA_S3 : WORD;
QUANT_PROD_SITJA_S4 : WORD;
QUANT_PROD_SITJA_S5 : WORD;
QUANT_PROD_SITJA_S6 : WORD;
QUANT_PROD_SITJA_S7 : WORD;
QUANT_RESTA_SITJA_S1 : WORD;
QUANT_RESTA_SITJA_S2 : WORD;
QUANT_RESTA_SITJA_S3 : WORD;
QUANT_RESTA_SITJA_S4 : WORD;
QUANT_RESTA_SITJA_S5 : WORD;
QUANT_RESTA_SITJA_S6 : WORD;
QUANT_RESTA_SITJA_S7 : WORD;
END_VAR

VAR
ComprovarSitjaS1 : ComprovarSitja;
ComprovarSitjaS0 : ComprovarSitja;
ComprovarSitjaS2 : ComprovarSitja;
ComprovarSitjaS3 : ComprovarSitja;
ComprovarSitjaS4 : ComprovarSitja;
ComprovarSitjaS5 : ComprovarSitja;
ComprovarSitjaS6 : ComprovarSitja;
ComprovarSitjaS7 : ComprovarSitja;
END_VAR
```

```
 1
 2 (* Comprovo el nivell de la Sitja de Matèries Primeres S1 *)
 3 ComprovarSitjaS1.QUANTITAT_MAX := 25000;
 4 ComprovarSitjaS1.LST_MAX := LST1_MAX;
 5 ComprovarSitjaS1.LST_MIN := LST1_MIN;
 6 ComprovarSitjaS1.LST_VAL := LST1_VAL;
 7 ComprovarSitjaS1();
 8 LST1_PRC := ComprovarSitjaS1.LST_PRC;
 9 ESTAT_SITJA_S1 := ComprovarSitjaS1.ESTAT_SITJA;
10 QUANT_PROD_SITJA_S1 := ComprovarSitjaS1.QUANTITAT;
11 QUANT_RESTA_SITJA_S1 := ComprovarSitjaS1.RESTA;
12
13 (* Comprovo el nivell de la Sitja de Matèries Primeres S2 *)
14 ComprovarSitjaS2.QUANTITAT_MAX := 25000;
15 ComprovarSitjaS2.LST_MAX := LST2_MAX;
16 ComprovarSitjaS2.LST_MIN := LST2_MIN;
17 ComprovarSitjaS2.LST_VAL := LST2_VAL;
18 ComprovarSitjaS2();
19 LST2_PRC := ComprovarSitjaS2.LST_PRC;
20 ESTAT_SITJA_S2 := ComprovarSitjaS2.ESTAT_SITJA;
21 QUANT_PROD_SITJA_S2 := ComprovarSitjaS2.QUANTITAT;
22 QUANT_RESTA_SITJA_S2 := ComprovarSitjaS2.RESTA;
23
24 (* Comprovo el nivell de la Sitja de Matèries Primeres S3 *)
25 ComprovarSitjaS3.QUANTITAT_MAX := 25000;
26 ComprovarSitjaS3.LST_MAX := LST3_MAX;
27 ComprovarSitjaS3.LST_MIN := LST3_MIN;
28 ComprovarSitjaS3.LST_VAL := LST3_VAL;
29 ComprovarSitjaS3();
30 LST3_PRC := ComprovarSitjaS3.LST_PRC;
31 ESTAT_SITJA_S3 := ComprovarSitjaS3.ESTAT_SITJA;
32 QUANT_PROD_SITJA_S3 := ComprovarSitjaS3.QUANTITAT;
33 QUANT_RESTA_SITJA_S3 := ComprovarSitjaS3.RESTA;
34
35 (* Comprovo el nivell de la Sitja de Matèries Primeres S4 *)
36 ComprovarSitjaS4.QUANTITAT_MAX := 25000;
37 ComprovarSitjaS4.LST_MAX := LST4_MAX;
38 ComprovarSitjaS4.LST_MIN := LST4_MIN;
39 ComprovarSitjaS4.LST_VAL := LST4_VAL;
40 ComprovarSitjaS4();
41 LST4_PRC := ComprovarSitjaS4.LST_PRC;
42 ESTAT_SITJA_S4 := ComprovarSitjaS4.ESTAT_SITJA;
43 QUANT_PROD_SITJA_S4 := ComprovarSitjaS4.QUANTITAT;
44 QUANT_RESTA_SITJA_S4 := ComprovarSitjaS4.RESTA;
45
46 (* Comprovo el nivell de la Sitja de Productes S5 *)
47 ComprovarSitjaS5.QUANTITAT_MAX := 25000;
48 ComprovarSitjaS5.LST_MAX := LST5_MAX;
49 ComprovarSitjaS5.LST_MIN := LST5_MIN;
50 ComprovarSitjaS5.LST_VAL := LST5_VAL;
51 ComprovarSitjaS5();
52 LST5_PRC := ComprovarSitjaS5.LST_PRC;
53 ESTAT_SITJA_S5 := ComprovarSitjaS5.ESTAT_SITJA;
54 QUANT_PROD_SITJA_S5 := ComprovarSitjaS5.QUANTITAT;
55 QUANT_RESTA_SITJA_S5 := ComprovarSitjaS5.RESTA;
56
57 (* Comprovo el nivell de la Sitja de Productes S6 *)
58 ComprovarSitjaS6.QUANTITAT_MAX := 25000;
59 ComprovarSitjaS6.LST_MAX := LST6_MAX;
60 ComprovarSitjaS6.LST_MIN := LST6_MIN;
61 ComprovarSitjaS6.LST_VAL := LST6_VAL;
62 ComprovarSitjaS6();
63 LST6_PRC := ComprovarSitjaS6.LST_PRC;
64 ESTAT_SITJA_S6 := ComprovarSitjaS6.ESTAT_SITJA;
65 QUANT_PROD_SITJA_S6 := ComprovarSitjaS6.QUANTITAT;
66 QUANT_RESTA_SITJA_S6 := ComprovarSitjaS6.RESTA;
67
68 (* Comprovo el nivell de la Sitja de Productes S7 *)
69 ComprovarSitjaS7.QUANTITAT_MAX := 25000;
70 ComprovarSitjaS7.LST_MAX := LST7_MAX;
```

```
71 ComprovarSitjaS7.LST_MIN := LST7_MIN;
72 ComprovarSitjaS7.LST_VAL := LST7_VAL;
73 ComprovarSitjaS7();
74 LST7_PRC := ComprovarSitjaS7.LST_PRC;
75 ESTAT_SITJA_S7 := ComprovarSitjaS7.ESTAT_SITJA;
76 QUANT_PROD_SITJA_S7 := ComprovarSitjaS7.QUANTITAT;
77 QUANT_RESTA_SITJA_S7 := ComprovarSitjaS7.RESTA;
78
79
```