Nom du fichier:	Ascenseur_Augustin_FUCHS.lib
Répertoire:	C:\Users\Augustin\Documents\Tp Ascenceur\Elevator\src
Date de la dernière modification:	13.10.22 15:59:22 / V2.3

Titre:
Auteur
Version:
Description:

```
0001 FUNCTION Get_Floor : INT
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005 END_VAR
```

```
0001 (*
0002 Cette fonction renvoie l'étage de l'ascenseur si il est arreté à un étage
0003 renvoie une valeur supérieur si il est en mouvement
0004 *)
0005 IF PRDC=TRUE THEN Get_Floor:=0;
0006 ELSIF PET1=TRUE THEN Get_Floor:=1;
0007 ELSIF PET2=TRUE THEN Get_Floor:=2;
0008 ELSIF PET3=TRUE THEN Get_Floor:=3;
0009 ELSE Get_Floor:=4;
0010 END_IF
0011 (*0n verifie que l'ascenceur n'est pas en cours de mouvement*)
0012 IF CABD OR CABM THEN Get_Floor:=5;END_IF
```

PorteEstFermee (FUN-ST)

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

```
0001 (*Cette fonction permet de savoir si la porte à l'étage ou se trouve l'ascenseur est fermée.
0002 Elle renvoie true si elle est fermée.
0003 *)
0004 PorteEstFermee:=FALSE;
0005 IF Get_Floor() = 0 THEN
0006     IF PRDCfer THEN PorteEstFermee:=TRUE;
0007     ELSE PorteEstFermee:=FALSE;
0008     END_IF
0009 ELSIF Get_Floor() = 1 THEN
0010     IF P1fer  THEN PorteEstFermee:=TRUE;
0011     ELSE PorteEstFermee:=FALSE;
0012     END_IF
0013 ELSIF Get_Floor() = 2 THEN
0014     IF P2fer  THEN PorteEstFermee:=TRUE;
0015     ELSE PorteEstFermee:=FALSE;
0016     END_IF
0017 ELSIF Get_Floor() = 3 THEN
0018     IF P3fer  THEN PorteEstFermee:=TRUE;
0019     ELSE PorteEstFermee:=FALSE;
0020     END_IF
0021 END_IF
```

PorteEstOuverte (FUN-ST)

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

```
0001 (*Cette fonction permet de savoir si la porte à l'étage ou se trouve l'ascenseur est ouverte.
0002 Elle renvoie true si elle est ouverte.
0003 *)
0004 PorteEstOuverte:=FALSE;
0005 IF Get_Floor() = 0 THEN
0006     IF PRDCouv THEN PorteEstOuverte:=TRUE;
0007     ELSE PorteEstOuverte:=FALSE;
0008     END_IF
0009 ELSIF Get_Floor() = 1 THEN
0010     IF P1ouv  THEN PorteEstOuverte:=TRUE;
0011     ELSE PorteEstOuverte:=FALSE;
0012     END_IF
0013 ELSIF Get_Floor() = 2 THEN
0014     IF P2ouv  THEN PorteEstOuverte:=TRUE;
0015     ELSE PorteEstOuverte:=FALSE;
0016     END_IF
```

```
0017 ELSIF Get_Floor() = 3 THEN
0018     IF P3ouv  THEN PorteEstOuverte:=TRUE;
0019     ELSE PorteEstOuverte:=FALSE;
0020     END_IF
0021 END_IF
```

Re_ouverture (FUN-ST)

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

```
0001 (* Cette fonction renvoie une variable permettant la ré-ouverture d'une porte en cas d'appuis cabine ou palier*)
0002
0003 IF (PALRDC OR CABRDC) AND PRDC THEN
0004     RE_OUVRE:=TRUE;
0005     PALRDC:=FALSE;
0006     CABRDC:=FALSE;
0007 ELSIF  (PALD1 OR CABET1)  AND PET1 AND NOT SENS_M THEN
0008     RE_OUVRE:=TRUE;
0009     PALD1:=FALSE;
0010     CABET1:=FALSE;
0011 ELSIF (PALM1 OR CABET1) AND PET1 AND SENS_M THEN
0012     RE_OUVRE:=TRUE;
0013     PALM1:=FALSE;
0014     CABET1:=FALSE;
0015 ELSIF (PALD2 OR CABET2) AND PET2 AND NOT SENS_M THEN
0016     RE_OUVRE:=TRUE;
0017     PALD2:=FALSE;
0018     CABET2:=FALSE;
0019 ELSIF (PALM2 OR CABET2) AND PET2 AND SENS_M THEN
0020     RE_OUVRE:=TRUE;
0021     PALM2:=FALSE;
0022     CABET2:=FALSE;
0023 ELSIF (PAL3 OR CABET3) AND PET3 THEN
0024     RE_OUVRE:=TRUE;
0025     PAL3:=FALSE;
0026     CABET3:=FALSE;
0027 ELSE
0028     RE_OUVRE:=FALSE;
0029 END_IF
0030 IF ReOuvreCabine THEN
0031     RE_OUVRE:=TRUE;
0032 END_IF
```

Appel_Palier (FUN-ST)

```
0001 FUNCTION Appel_Palier : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005     TesteurLocal: BOOL;
0006     Operate_Monte: BOOL;
0007     CompteurLocal: INT;
0008     CompteurForLocal:INT;
0009     Niveau_Actuel: INT;
0010 END_VAR
```

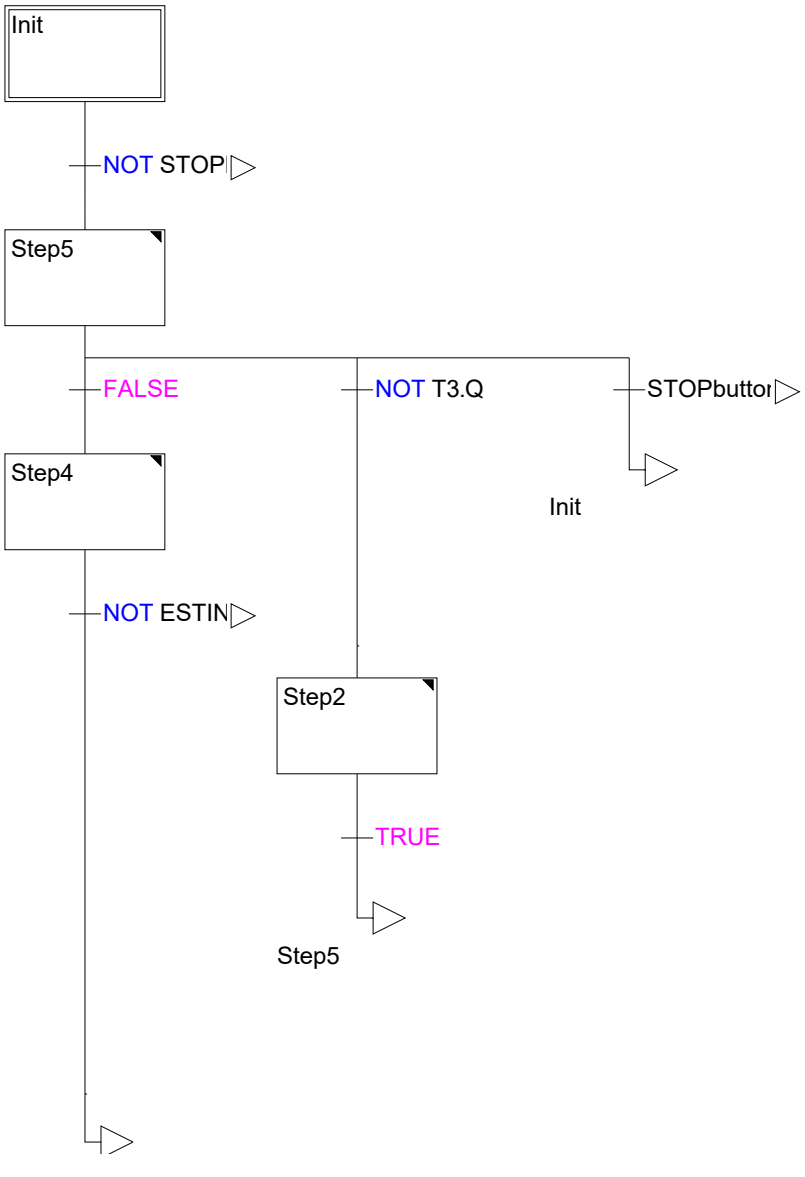
```
0001 (*Cette fonction prend en charge un appel du palier
0002 si l'ascenseur est initialisé elle renvoie l'appel palier,  enregistre la mémo et lance le clignotement*)
0003 IF ESTINIT2  OR StanbyActive THEN
0004     IF PALRDCcligno THEN
0005         PALRDC:=TRUE;
0006     END_IF
0007     IF NOT PALRDC THEN
0008         VRDC:=FALSE;
0009     END_IF
0010
0011     IF PALD1cligno THEN
0012         PALD1:=TRUE;
0013     END_IF
```

```
0014      IF PALM1cligno THEN
0015          PALM1:=TRUE;
0016      END_IF
0017
0018      IF NOT PALD1 AND NOT PALM1 THEN
0019          V1:=FALSE;
0020      END_IF
0021
0022      IF PALD2cligno THEN
0023          PALD2:=TRUE;
0024      END_IF
0025      IF PALM2cligno THEN
0026          PALM2:=TRUE;
0027          COUNTSTEP:=COUNTSTEP+1;
0028      END_IF
0029      IF NOT PALD2 AND NOT PALM2 THEN
0030          V2:=FALSE;
0031      END_IF
0032
0033      IF PAL3cligno THEN
0034          PAL3:=TRUE;
0035          COUNTSTEP:=COUNTSTEP+1;
0036      END_IF
0037
0038      IF NOT PAL3 THEN V3:=FALSE;END_IF
0039      IF PALRDC OR PALD1 OR PALM1 OR PALM2 OR PALD2 OR PAL3 OR CABRDC OR CABET1 OR CABET2 OR CABET3 THEN EXIT_StanbyFu
0040      ELSE EXIT_StanbyFunc:=FALSE; END_IF
0041  END_IF
```

inc:=TRUE;

Clignotement (PRG-SFC)

```
0001 PROGRAM Clignotement
0002 VAR
0003 END_VAR
```



Step2

Clignotement (PRG-SFC).Action Step5 (ST)

```
0001 T3(IN:=TRUE, PT:=t#500ms);
```

Clignotement (PRG-SFC).Action Step4 (ST)

```
0001 T2(IN:=FALSE);
0002 ESTINIT2:=FALSE;
```

Clignotement (PRG-SFC).Action Step2 (ST)

```
0001 (* IF  CABD OR CABM OR ORDC OR O1 OR O2 OR O3 OR NOT PRDCfer OR NOT P1fer OR NOT P2fer OR NOT P3fer OR PAL3 OR PALM2 OR
0002       T2(IN:=TRUE, PT:=t#10s);
0003 ELSE
0004       T2(IN:=FALSE);
0005 END_IF
0006 *)
0007 IF PALRDC THEN
0008       IF VRDC THEN VRDC:=FALSE;
0009       ELSE VRDC:=TRUE;END_IF
0010 END_IF
0011 IF PALD1 OR PALM1 THEN
0012       IF V1 THEN V1:=FALSE;
0013       ELSE V1:=TRUE;END_IF
0014 END_IF
0015 IF PALD2 OR PALM2 THEN
0016       IF V2 THEN V2:=FALSE;
0017       ELSE V2:=TRUE;END_IF
0018 END_IF
0019 IF PAL3 THEN
0020       IF V3 THEN V3:=FALSE;
0021       ELSE V3:=TRUE;END_IF
0022 END_IF
0023 T3(IN:=FALSE);
0024 T3(IN:=TRUE, PT:=t#1ms);
```

PALD2 OR PALD1 OR PALM1 OR PALRDC THEN

Quel_Etage (FUN-ST)

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

```
0001 (*Cette fonction permet d'activer et de désactiver les voyant de présence à l'érage en fonction des capteurs de présence*)
0002 IF PRDC THEN
0003     VPALRDC:=TRUE;
0004     VPAL1:=FALSE;
0005     VPAL2:=FALSE;
0006     VPAL3:=FALSE;
0007 ELSE IF PET1 THEN
0008     VPALRDC:=FALSE;
0009     VPAL1:=TRUE;
0010     VPAL2:=FALSE;
0011     VPAL3:=FALSE;
0012 ELSE IF PET2 THEN
0013     VPALRDC:=FALSE;
0014     VPAL1:=FALSE;
0015     VPAL2:=TRUE;
0016     VPAL3:=FALSE;
0017 ELSE IF PET3 THEN
0018     VPALRDC:=FALSE;
0019     VPAL1:=FALSE;
0020     VPAL2:=FALSE;
0021     VPAL3:=TRUE;
0022 END_IF
0023 END_IF
0024 END_IF
0025 END_IF
0026 Quel_Etage:=TRUE;
```

Reset_Arrets (FUN-ST)

```
0001 FUNCTION Reset_Arrets : BOOL
```

```
0002 VAR_INPUT
0003     ARRET_EN_COURS: INT;
0004 END_VAR
0005 VAR
0006 END_VAR
```

```
0001 (* cette fonction permet de réinitialiser tout les arrets d'un étage pour permettre un nouvel appel lors d'un prochain cycle*)
0002 IF ARRET_EN_COURS=0 THEN
0003     PALRDC:=FALSE;
0004     CABRDC:=FALSE;
0005     ARRETcabRDC:=FALSE;
0006 ELSIF ARRET_EN_COURS=1 THEN
0007     PALD1:=FALSE;
0008     CABET1:=FALSE;
0009     ARRETcabD1:=FALSE;
0010 ELSIF ARRET_EN_COURS=2 THEN
0011     PALM1:=FALSE;
0012     CABET1:=FALSE;
0013     ARRETcabM1:=FALSE;
0014 ELSIF ARRET_EN_COURS=3 THEN
0015     PALD2:=FALSE;
0016     CABET2:=FALSE;
0017     ARRETcabD2:=FALSE;
0018 ELSIF ARRET_EN_COURS=4 THEN
0019     ARRETcabM2:=FALSE;
0020     PALM2:=FALSE;
0021     CABET2:=FALSE;
0022 ELSIF ARRET_EN_COURS=5 THEN
0023     PAL3:=FALSE;
0024     CABET3:=FALSE;
0025     ARRETcabET3:=FALSE;
0026 END_IF
0027 ETAGE_SUIVANT:=9;
```

Reset_Arrets_CAB (FUN-ST)

```
0001 FUNCTION Reset_Arrets_CAB : BOOL
0002 VAR_INPUT
0003     ARRET_EN_COURS: INT;
0004 END_VAR
0005 VAR
0006 END_VAR
```

```
0001 (* cette fonction permet de réinitialiser les boutons d'appels cabine de l'ascenceur *)
0002 IF ARRET_EN_COURS=0 THEN
0003     CABRDC:=FALSE;
0004 ELSIF ARRET_EN_COURS=1 OR ARRET_EN_COURS=2 THEN
0005     IF ARRET_EN_COURS = 1 THEN PALD1:=FALSE;
0006     ELSE PALM1:=FALSE; END_IF
0007     CABET1:=FALSE;
0008 ELSIF ARRET_EN_COURS=3 OR ARRET_EN_COURS= 4 THEN
0009     IF ARRET_EN_COURS = 3 THEN PALD2:=FALSE;
0010     ELSE PALM2:=FALSE; END_IF
0011     CABET2:=FALSE;
0012 ELSIF ARRET_EN_COURS=5 THEN
0013     CABET3:=FALSE;
0014 END_IF
```

Arret_Urgence (FUN-ST)

```
0001 FUNCTION Arret_Urgence : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005     ReinitAscenseurProcess: BOOL;
0006 END_VAR
```

```
0001 (* Cette fonction va lors d'une impulsion sur l'arret d'urgence, bloquer tous les mouvements des moteurs de l'ascenceur
0002 et attendre une réponse de l'utilisateur quand à l'état de retour à la normal à adopter: reset / Reprise du dernier cycle.*)
0003 IF STOPbuttonVarImpuls THEN
0004     STOPbuttonVar:=TRUE;
0005     ReinitAscenseurProcess:=TRUE;
0006 END_IF
0007 IF STOPbuttonVar THEN
```

```
(*STOP MOTORS *)
    (*on coupe tout les moteurs en action*)
    STOPbuttonVar:=TRUE;

    IF ReinitAscenseurProcess THEN
        (*on sauvegarde l'état de la logique de l'ascenceur
            -boutons
            -voyants
            -variable de prio
*)
        CABDsave:=CABD;
        CABMsave:=CABM;

        PALRDCsave:=PALRDC;
        PALD1save:=PALD1;
        PALM1save:=PALM1;
        PALD2save:=PALD2;
        PALM2save:=PALM2;
        PAL3save:=PAL3;

        CABRDCsave:=CABRDC;
        CABET1save:=CABET1;
        CABET2save:=CABET2;
        CABET3save:=CABET3;

        ARRETcabRDCsave:=ARRETcabRDC;
        ARRETcabM1save:=ARRETcabM1;
        ARRETcabD1save:=ARRETcabD1;
        ARRETcabM2save:=ARRETcabM2;
        ARRETcabD2save:=ARRETcabD2;
        ARRETcabET3save:=ARRETcabET3;

        PROCHAIN_ARRETsave:=PROCHAIN_ARRET;
        ETAGE_SUIVANTsave:=ETAGE_SUIVANT;

        ReinitAscenseurProcess:=FALSE;
    END_IF

    CABD:=FALSE;
    CABM:=FALSE;

    ORDC:=FALSE;
    O1:=FALSE;
    O2:=FALSE;
    O3:=FALSE;

    FRDC:=FALSE;
    F1:=FALSE;
    F2:=FALSE;
    F3:=FALSE;

    ARRETcabRDC:=FALSE;
    ARRETcabM1:=FALSE;
    ARRETcabD1:=FALSE;
    ARRETcabM2:=FALSE;
    ARRETcabD2:=FALSE;
    ARRETcabET3:=FALSE;

    PALRDC:=FALSE;
    PALD1:=FALSE;
    PALM1:=FALSE;
    PALD2:=FALSE;
    PALM2:=FALSE;
    PAL3:=FALSE;

    CABRDC:=FALSE;
    CABET1:=FALSE;
    CABET2:=FALSE;
```

```
0076        CABET3:=FALSE;
0077        AfficheBoutonsReset:=FALSE;
0078        (*Reprise si initiaisé*)
0079        IF ESTINIT2 THEN AfficheBoutonsReset2:=TRUE;END_IF
0080
0081    ELSE
0082        AfficheBoutonsReset:=TRUE;
0083        AfficheBoutonsReset2:=FALSE;
0084    END_IF
0085
0086    (* si l'utilisateur demande un reset de l'ascenceur*)
0087    IF BoutonResetVrai THEN
0088        ESTINIT2:=FALSE;
0089        STOPbuttonVar:=FALSE;
0090        BoutonResetVrai:=FALSE;
0091        ETAGE_SUIVANT:=1;
0092        PROCHAIN_ARRET:=0;
0093
0094    (* si l'utilisateur demande un retour au dernier état de l'ascenceur*)
0095    ELSIF BoutonRepriseVrai THEN
0096        ESTINIT2:=TRUE;
0097        STOPbuttonVar:=FALSE;
0098
0099        IF NOT PRDC AND NOT PET1 AND NOT PET2 AND NOT PET3 THEN
0100            CABD:=CABDsave;
0101            CABM:=CABMsave;
0102        END_IF
0103
0104        PALRDC:=PALRDCsave;
0105        PALD1:=PALD1save;
0106        PALM1:=PALM1save;
0107        PALD2:=PALD2save;
0108        PALM2:=PALM2save;
0109        PAL3:=PAL3save;
0110
0111        CABRDC:=CABRDCsave;
0112        CABET1:=CABET1save;
0113        CABET2:=CABET2save;
0114        CABET3:=CABET3save;
0115
0116        ARRETcabRDC:=ARRETcabRDCsave;
0117        ARRETcabM1:=ARRETcabM1save;
0118        ARRETcabD1:=ARRETcabD1save;
0119        ARRETcabM2:=ARRETcabM2save;
0120        ARRETcabD2:=ARRETcabD2save;
0121        ARRETcabET3:=ARRETcabET3save;
0122
0123        PROCHAIN_ARRET:=PROCHAIN_ARRETsave;
0124        ETAGE_SUIVANT:=ETAGE_SUIVANTsave;
0125        BoutonRepriseVrai:=FALSE;
0126    END_IF
0127
```

Correspondance_Etages (FUN-ST)

```
0001    FUNCTION Correspondance_Etages : BOOL
0002    VAR_INPUT
0003    END_VAR
0004    VAR
0005        CabPosBout: INT;
0006        PosAscActu: INT;
0007    END_VAR
```

```
0001    (*
0002        Cette fonction à pour but de faire correspondre un appui de la cabine à un appel provenant du palier (sans activer de bouton)
0003        le but étant de limiter les variables et de se caller sur le cycle existant fonctionnel
0004
0005        Après avoir récuperer la position de l'ascenseur, nous allons faire correspondre le sens pour que l'étage demandé soit e porchain deservie (dans le cycle)
0006
0007        Si l'etage demandé est dans le sens et n'as pas encore été demandé, alors il sera dans ce sens
0008        Si l'étage demandé est déja passé alors nous nous callons sur l'autre sens pour qu'il soit deservie au porchain cycle
```

```
0009 *)
0010
0011 PosAscActu:= Last_Capt_Pres();
0012 IF CABRDCswitch THEN
0013     IF NOT ARRETcabRDC THEN ARRETcabRDC:=TRUE;END_IF
0014     CABRDC:=TRUE;
0015 END_IF
0016
0017 IF CABET1switch THEN
0018     CabPosBout:=2;
0019     IF PosAscActu < 5 OR  ETAGE_SUIVANT < 9 THEN
0020         IF PosAscActu >= 5 THEN PosAscActu:= ETAGE_SUIVANT;END_IF
0021         IF SENS_M THEN
0022             IF PosAscActu <= CabPosBout THEN ARRETcabM1:=TRUE;
0023             ELSE ARRETcabD1:=TRUE;END_IF
0024         ELSE
0025             IF PosAscActu >= CabPosBout THEN ARRETcabD1:=TRUE;
0026             ELSE ARRETcabM1:=TRUE;END_IF
0027         END_IF
0028     ELSE
0029         IF SENS_M THEN
0030             IF NOT ARRETcabM1 THEN ARRETcabM1:=TRUE;END_IF
0031         ELSE
0032             IF NOT ARRETcabD1 THEN ARRETcabD1:=TRUE;END_IF
0033         END_IF
0034     END_IF
0035     CABET1:=TRUE;
0036 END_IF
0037
0038 IF CABET2switch THEN
0039     CabPosBout:=3;
0040     IF PosAscActu < 5 OR  ETAGE_SUIVANT < 9 THEN
0041         IF PosAscActu >= 5 THEN PosAscActu:= ETAGE_SUIVANT;END_IF
0042         IF SENS_M THEN
0043             IF PosAscActu <= CabPosBout THEN ARRETcabM2:=TRUE;
0044             ELSE ARRETcabD2:=TRUE;END_IF
0045         ELSE
0046             IF PosAscActu >= CabPosBout THEN ARRETcabD2:=TRUE;
0047             ELSE ARRETcabM2:=TRUE;END_IF
0048         END_IF
0049     ELSE
0050         IF SENS_M THEN
0051             IF NOT ARRETcabM2 THEN ARRETcabM2:=TRUE;END_IF
0052         ELSE
0053             IF NOT ARRETcabD2 THEN ARRETcabD2:=TRUE;END_IF
0054         END_IF
0055     END_IF
0056     CABET2:=TRUE;
0057 END_IF
0058
0059 IF CABET3switch THEN
0060     CabPosBout:=4;
0061     IF NOT ARRETcabET3 THEN ARRETcabET3:=TRUE;END_IF
0062     CABET3:=TRUE;
0063 END_IF
0064
0065 (*Si un appel palier est effectué, il est associé à sa mémorisation*)
0066 IF PALRDC THEN ARRETcabRDC:=TRUE;END_IF
0067 IF PALD1 THEN ARRETcabD1:=TRUE;END_IF
0068 IF PALM1 THEN ARRETcabM1:=TRUE;END_IF
0069 IF PALD2 THEN ARRETcabD2:=TRUE;END_IF
0070 IF PALM2 THEN ARRETcabM2:=TRUE;END_IF
0071 IF PAL3 THEN ARRETcabET3:=TRUE;END_IF
```

Last_Capt_Pres (FUN-ST)

```
0001 FUNCTION Last_Capt_Pres : INT
0002 VAR_INPUT
0003 END_VAR
0004 VAR
```

```
0005 END_VAR
```

```
0001 (*cette fonction se rapproche de get_floor mais elle permet d'obtenir le dernier capteur ou était l'ascenseur*)
0002 IF PRDC=TRUE THEN Last_Capt_Pres_Rec:=1;
0003 ELSIF PET1=TRUE THEN Last_Capt_Pres_Rec:=2;
0004 ELSIF PET2=TRUE THEN Last_Capt_Pres_Rec:=3;
0005 ELSIF PET3=TRUE THEN Last_Capt_Pres_Rec:=4; END_IF
0006 IF NOT PRDC AND NOT PET1 AND NOT PET2 AND NOT PET3 THEN
0007     Last_Capt_Pres:=Last_Capt_Pres_Rec;
0008     IF CABM AND NOT CABD THEN
0009         Last_Capt_Pres:= Last_Capt_Pres_Rec+1;
0010     ELSIF CABD AND NOT CABM THEN
0011         Last_Capt_Pres:= Last_Capt_Pres_Rec-1;
0012     END_IF
0013     IF Last_Capt_Pres > 4 THEN Last_Capt_Pres:=4;END_IF
0014     IF Last_Capt_Pres = 0 THEN Last_Capt_Pres:=1;END_IF
0015 ELSE
0016     Last_Capt_Pres:=Last_Capt_Pres_Rec;
0017 END_IF
```

Last_Capt_Pres2 (FUN-ST)

```
0001 FUNCTION Last_Capt_Pres2 : INT
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005 END_VAR
```

```
0001 (*cette fonction se rapproche de get_floor mais elle permet d'obtenir le dernier capteur ou était l'ascenseur*)
0002 IF PRDC=TRUE THEN Last_Capt_Pres2:=1; END_IF
0003 IF PET1=TRUE THEN Last_Capt_Pres2:=2; END_IF
0004 IF PET2=TRUE THEN Last_Capt_Pres2:=3; END_IF
0005 IF PET3=TRUE THEN Last_Capt_Pres2:=4; END_IF
0006 IF NOT PRDC AND NOT PET1 AND NOT PET2 AND NOT PET3 THEN Last_Capt_Pres2:=0;END_IF
```

Ouv_Fer_Etage (FUN-ST)

```
0001 FUNCTION Ouv_Fer_Etage : BOOL
0002 VAR_INPUT
0003     (*True pour ouvrir et False pour fermer*)
0004     OuvrirFermer:BOOL;
0005 END_VAR
0006 VAR
0007 END_VAR
```

```
0001 (*Cette fonction ouvre la porte de l'étage à laquelle il se trouve l'ascenseur si ell est vrai
0002 la ferme si elle est fausse*)
0003 IF OuvrirFermer THEN
0004     IF Get_Floor() = 0 THEN
0005         IF NOT PRDCouv THEN ORDC:=TRUE;END_IF
0006     ELSIF Get_Floor() = 1 THEN
0007         IF NOT P1ouv THEN O1:=TRUE;END_IF
0008     ELSIF Get_Floor() = 2 THEN
0009         IF NOT P2ouv THEN O2:=TRUE;END_IF
0010     ELSIF Get_Floor() = 3 THEN
0011         IF NOT P3ouv THEN O3:=TRUE;END_IF
0012     END_IF
0013 ELSE
0014     IF Get_Floor() = 0 THEN
0015         IF NOT PRDCfer THEN FRDC:=TRUE;END_IF
0016     ELSIF Get_Floor() = 1 THEN
0017         IF NOT P1fer THEN F1:=TRUE;END_IF
0018     ELSIF Get_Floor() = 2 THEN
0019         IF NOT P2fer THEN F2:=TRUE;END_IF
0020     ELSIF Get_Floor() = 3 THEN
0021         IF NOT P3fer THEN F3:=TRUE;END_IF
0022     END_IF
0023 END_IF
```

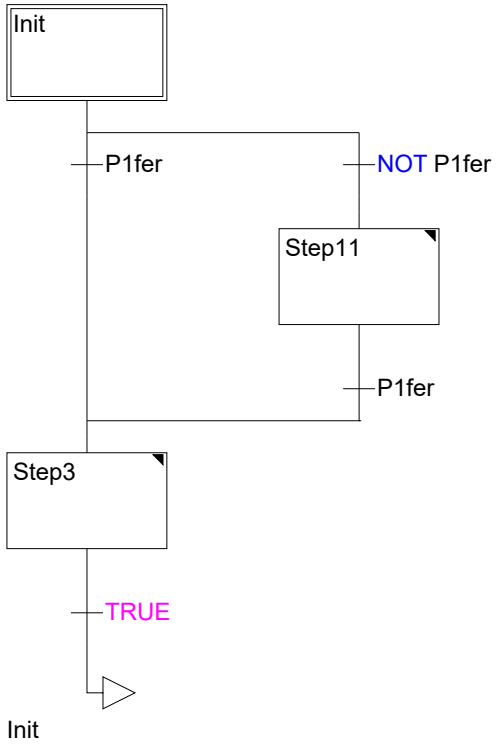
STOP_PORTES (FUN-ST)

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

```
0001 (*Cette fonction est une sécurité qui bloque l'ascenseur tant que toutes les portes ne sont pas fermé*)
0002 IF CABD OR CABM THEN
0003     IF NOT PRDCfer OR NOT P1fer OR NOT P2fer OR NOT P3fer THEN
0004         CABDstatus:= CABD;
0005         CABMstatus:= CABM;
0006         CABD:=FALSE;
0007         CABM:=FALSE;
0008     END_IF
0009 ELSIF NOT CABD AND NOT CABM THEN
0010     IF PRDCfer AND P1fer AND P2fer AND P3fer THEN
0011         IF CABDstatus OR CABDstatus THEN
0012             CABD:= CABDstatus;
0013             CABM:= CABMstatus;
0014             CABDstatus:=FALSE;
0015             CABMstatus:=FALSE;
0016         END_IF
0017     END_IF
0018 END_IF
```

ETAGE_1 (PRG-SFC)

```
0001 PROGRAM ETAGE_1
0002 VAR
0003 END_VAR
```



Init

ETAGE_1 (PRG-SFC).Action Step11 (ST)

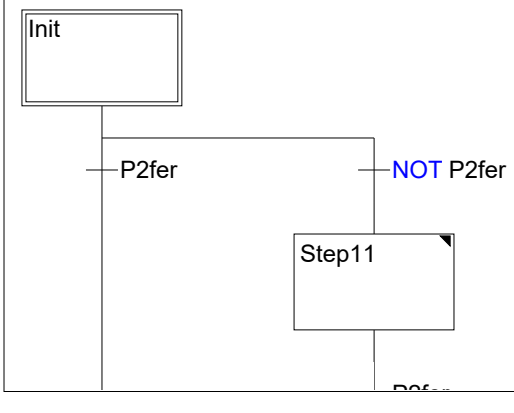
```
0001 F1:=TRUE;
```

ETAGE_1 (PRG-SFC).Action Step3 (ST)

```
0001 F1:=FALSE;
```

ETAGE_2 (PRG-SFC)

```
0001 PROGRAM ETAGE_2
0002 VAR
0003 END_VAR
```



ETAGE_2 (PRG-SFC).Action Step11 (ST)

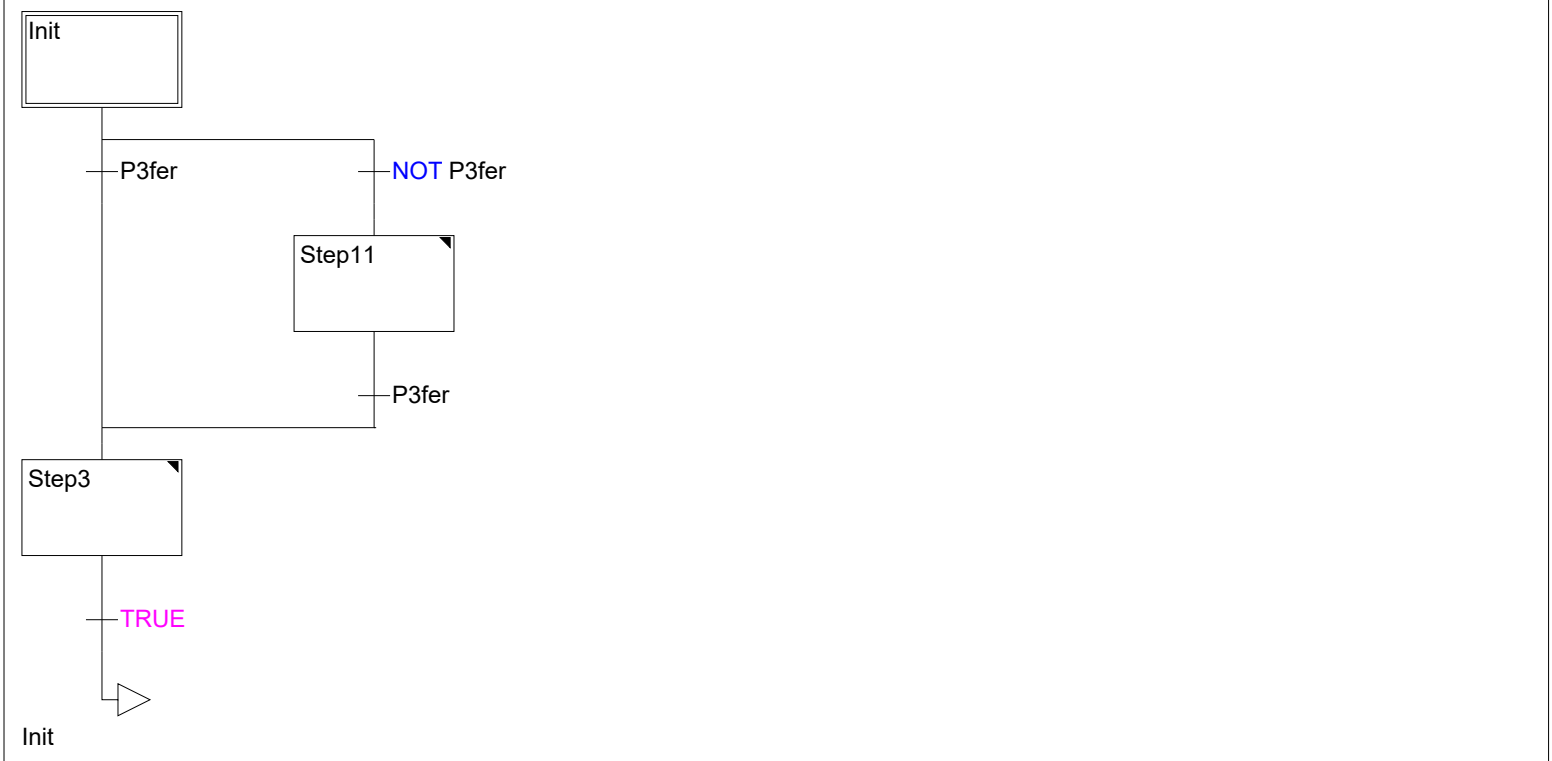
```
0001 F2:=TRUE;
```

ETAGE_2 (PRG-SFC).Action Step3 (ST)

```
0001 F2:=FALSE;
```

ETAGE_3 (PRG-SFC)

```
0001 PROGRAM ETAGE_3
0002 VAR
0003 END_VAR
```



ETAGE_3 (PRG-SFC).Action Step11 (ST)

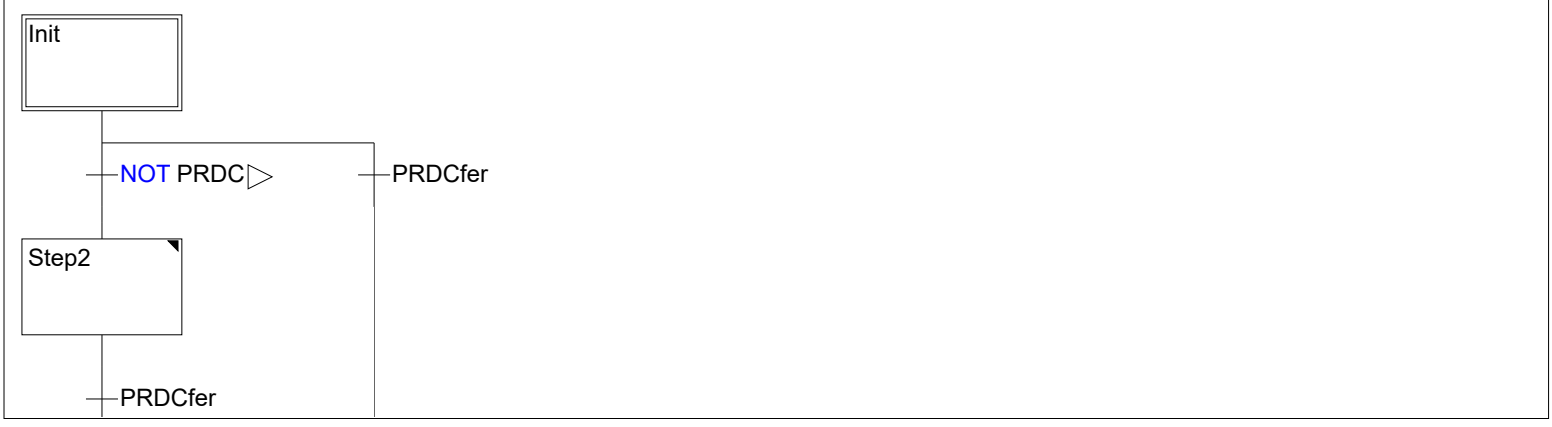
```
0001 F3:=TRUE;
```

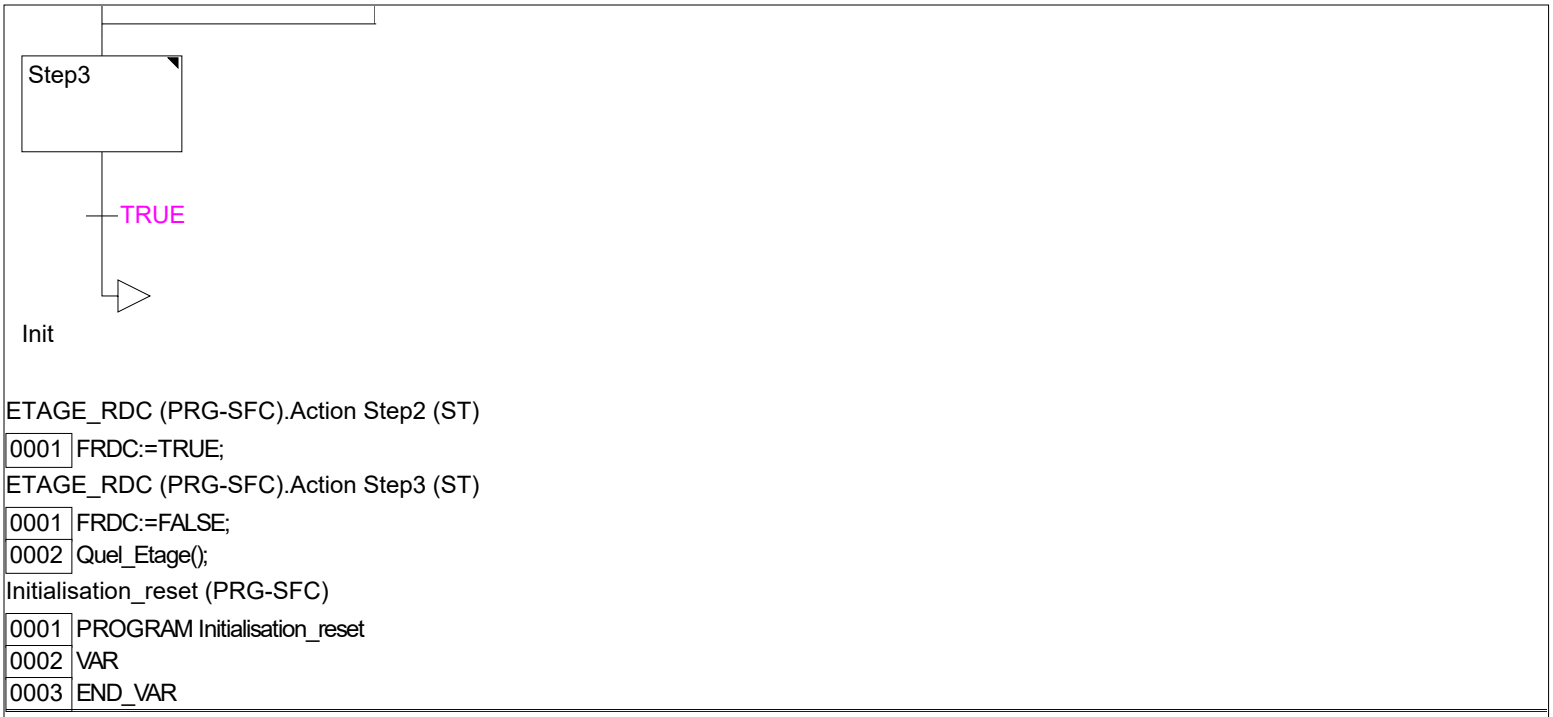
ETAGE_3 (PRG-SFC).Action Step3 (ST)

```
0001 F3:=FALSE;
```

ETAGE_RDC (PRG-SFC)

```
0001 PROGRAM ETAGE_RDC
0002 VAR
0003 END_VAR
```





ETAGE_RDC (PRG-SFC).Action Step2 (ST)

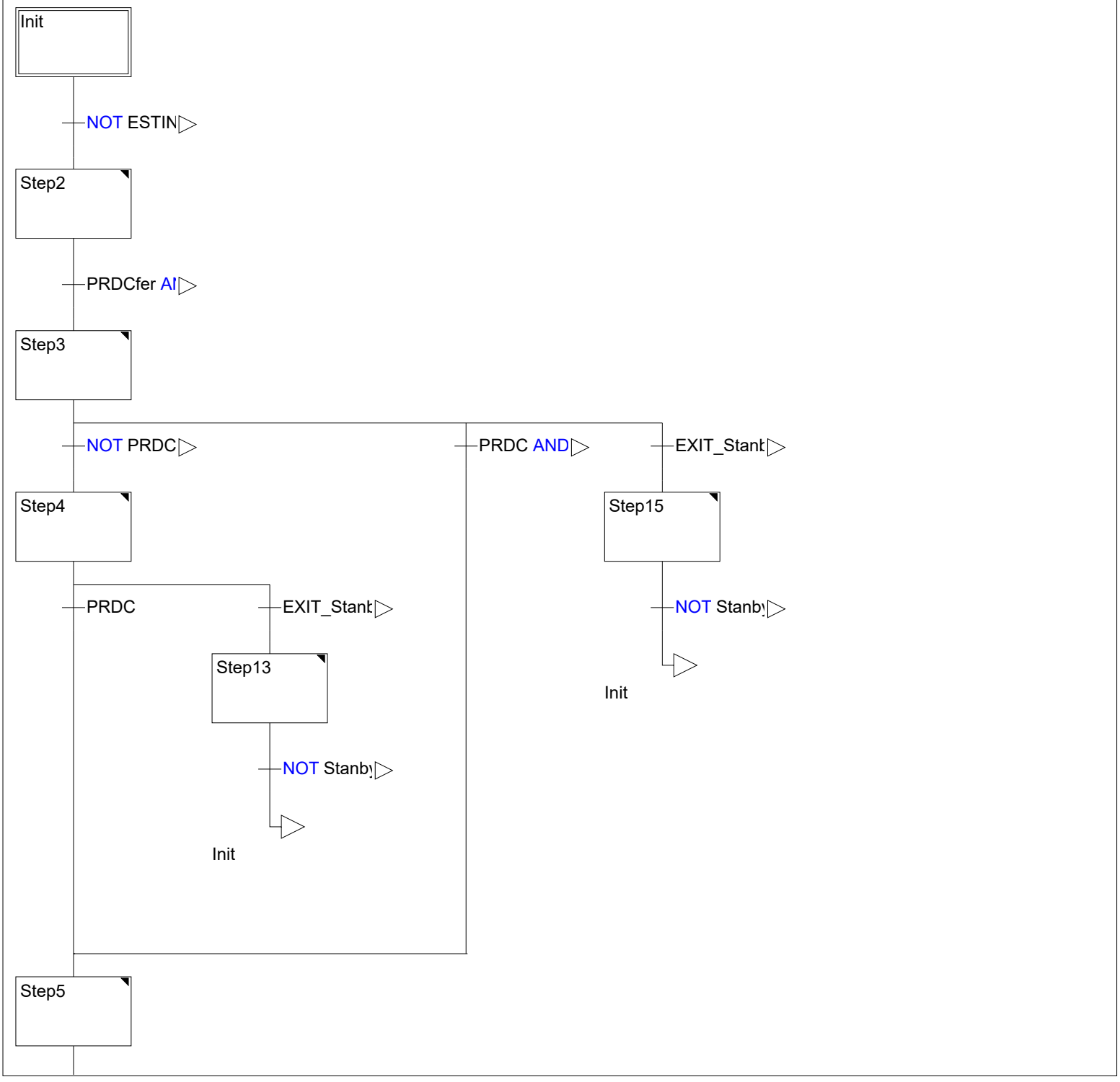
```
0001 FRDC:=TRUE;
```

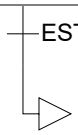
ETAGE_RDC (PRG-SFC).Action Step3 (ST)

```
0001 FRDC:=FALSE;
0002 Quel_Etage();
```

Initialisation_reset (PRG-SFC)

```
0001 PROGRAM Initialisation_reset
0002 VAR
0003 END_VAR
```





Init

Initialisation_reset (PRG-SFC).Action Step2 (ST)

```
0001 ETAGE_RDC;
0002 ETAGE_1;
0003 ETAGE_2;
0004 ETAGE_3;
0005 ETAGE_SUIVANT:=1;
0006 PROCHAIN_ARRET:=0;
```

Initialisation_reset (PRG-SFC).Action Step3 (ST)

```
0001 (*Désactivation de tout les moteurs *)
0002 FRDC:=FALSE;
0003 F1:=FALSE;
0004 F2:=FALSE;
0005 F3:=FALSE;
0006
```

Initialisation_reset (PRG-SFC).Action Step4 (ST)

```
0001 IF NOT PRDC THEN CABD:=TRUE;END_IF
0002 SENS_M:=FALSE;
```

Initialisation_reset (PRG-SFC).Action Step13 (ST)

```
0001 EXIT_StanbyFunc :=FALSE;
0002 ESTINIT2:=TRUE;
0003 CABD:=FALSE;
0004 ETAGE_SUIVANT:=9;
0005 PROCHAIN_ARRET:=9;
0006 StanbyActive:=FALSE;
```

Initialisation_reset (PRG-SFC).Action Step15 (ST)

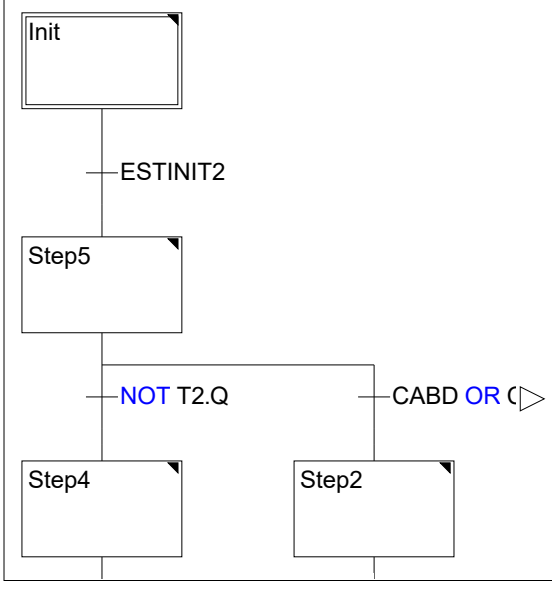
```
0001 EXIT_StanbyFunc:=FALSE;
0002 ESTINIT2:=TRUE;
0003 CABD:=FALSE;
0004 ETAGE_SUIVANT:=9;
0005 PROCHAIN_ARRET:=9;
0006 StanbyActive:=FALSE;
```

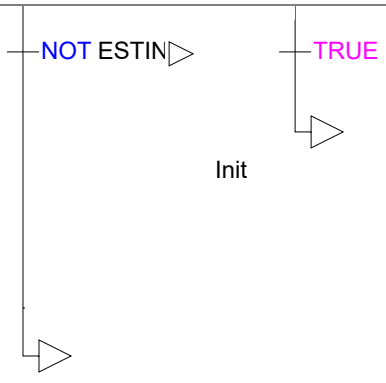
Initialisation_reset (PRG-SFC).Action Step5 (ST)

```
0001 CABD:=FALSE;
0002 ESTINIT2:=TRUE;
0003 StanbyActive:=FALSE;
0004 SENS_M:=TRUE;
0005 ETAGE_SUIVANT:=9;
0006 PROCHAIN_ARRET:=9;
```

StandBY (PRG-SFC)

```
0001 PROGRAM StandBY
0002 VAR
0003 END_VAR
```





StandBY (PRG-SFC).Action Init (ST)

```
0001 T2(IN:=FALSE);
```

StandBY (PRG-SFC).Action Step5 (ST)

```
0001 T2(IN:=TRUE, PT:=t#7s);
```

StandBY (PRG-SFC).Action Step4 (ST)

```
0001 T2(IN:=FALSE);
0002 ESTINIT2:=FALSE;
0003 StanbyActive:=TRUE;
```

StandBY (PRG-SFC).Action Step2 (ST)

```
0001 (* IF  CABD OR CABM OR ORDC OR O1 OR O2 OR O3 OR NOT PRDCfer OR NOT P1fer OR NOT P2fer OR NOT P3fer OR PAL3 OR PALM2 OR
0002       T2(IN:=TRUE, PT:=t#10s);
0003 ELSE
0004       T2(IN:=FALSE);
0005 END_IF
0006 *)
0007
0008 T2(IN:=FALSE);
0009 T2(IN:=TRUE, PT:=t#1ms);
```

PALD2 OR PALD1 OR PALM1 OR PALRDC THEN

Afficher_Etage_Range (FUN-ST)

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

```
0001 (* Cette fonction sert à l'affichage de l'étage dans lequel se trouve l'asenceur,
0002 ainsi que l'étage qui sera desservie prochainement.*)
0003
0004 (* Utilisé uniquement sur la simulation visuelle de l'écran
0005 n'interfère pas avecx le porgramme principal*)
0006
0007 IF 55 < CABmoveY AND CABmoveY<= 110 THEN
0008     DisplayNumbETRDC:=TRUE;
0009     DisplayNumbET1:=FALSE;
0010 ELSIF -55 < CABmoveY AND CABmoveY<= 55  THEN
0011     DisplayNumbET1:=TRUE;
0012     DisplayNumbETRDC:=FALSE;
0013     DisplayNumbET2:=FALSE;
0014 ELSIF -165 < CABmoveY AND CABmoveY <= -55  THEN
0015     DisplayNumbET2:=TRUE;
0016     DisplayNumbET1:=FALSE;
0017     DisplayNumbET3:=FALSE;
0018 ELSIF -220 < CABmoveY AND CABmoveY <= -165  THEN
0019     DisplayNumbET3:=TRUE;
0020     DisplayNumbET2:=FALSE;
0021 END_IF
0022
0023 IF Get_Floor()+1 <> ETAGE_SUIVANT THEN
0024     IF ETAGE_SUIVANT =  9 THEN
0025         FLECHE_VERS:=FALSE;
0026         DisplayVersRDC:=FALSE;
0027         DisplayVersET1:=FALSE;
0028         DisplayVersET2:=FALSE;
0029         DisplayVersET3:=FALSE;
```

```
0030      ELSIF ETAGE_SUIVANT = 1 THEN
0031          FLECHE_VERS:=TRUE;
0032          DisplayVersET1:=FALSE;
0033          DisplayVersET2:=FALSE;
0034          DisplayVersET3:=FALSE;
0035          DisplayVersRDC:=TRUE;
0036      ELSIF ETAGE_SUIVANT = 2 THEN
0037          FLECHE_VERS:=TRUE;
0038          DisplayVersRDC:=FALSE;
0039          DisplayVersET2:=FALSE;
0040          DisplayVersET3:=FALSE;
0041          DisplayVersET1:=TRUE;
0042      ELSIF ETAGE_SUIVANT = 3 THEN
0043          FLECHE_VERS:=TRUE;
0044          DisplayVersRDC:=FALSE;
0045          DisplayVersET1:=FALSE;
0046          DisplayVersET3:=FALSE;
0047          DisplayVersET2:=TRUE;
0048      ELSIF ETAGE_SUIVANT = 4 THEN
0049          FLECHE_VERS:=TRUE;
0050          DisplayVersRDC:=FALSE;
0051          DisplayVersET1:=FALSE;
0052          DisplayVersET2:=FALSE;
0053          DisplayVersET3:=TRUE;
0054      END_IF
0055  ELSE
0056      FLECHE_VERS:=FALSE;
0057      DisplayVersRDC:=FALSE;
0058      DisplayVersET1:=FALSE;
0059      DisplayVersET2:=FALSE;
0060      DisplayVersET3:=FALSE;
0061  END_IF
0062
0063  IF NOT ARRETcabRDC AND NOT ARRETcabD1 AND  NOT ARRETcabM1 AND  NOT ARRETcabD2 AND NOT ARRETcabM2 AND NOT ARRETcabET
0064      IF PRDC OR PET3 THEN
0065          FinDefinition_SENS:=TRUE;
0066      END_IF
0067  ELSE FinDefinition_SENS:=FALSE;END_IF
0068
0069
0070
0071  Afficher_Etage_Range:=FALSE;
```

'3 THEN

FX_Porte_Vitesse (FUN-ST)

```
0001 FUNCTION FX_Porte_Vitesse : REAL
0002 VAR_INPUT
0003     PosXPorte: INT;
0004 END_VAR
0005 VAR
0006     VarLocalA: REAL;
0007     ValXLocal: REAL;
0008 END_VAR
```

```
0001 (*Cette fonction retourne la position X de la porte gauche en fonction de la position X de la porte droite*)
0002 (* Elle permet une exacte synchronisation des deux portes*)
0003 VarLocalA:=-(59.0/71.0);
0004 ValXLocal:= INT_TO_REAL(PosXPorte);
0005 FX_Porte_Vitesse:=(VarLocalA)*(ValXLocal);
```

FX_PorteD (FUN-ST)

```
0001 FUNCTION FX_PorteD : REAL
0002 VAR_INPUT
0003     PosXPorte: INT;
0004 END_VAR
0005 VAR
0006     VarLocalA: REAL;
0007     ValXLocal: REAL;
0008 END_VAR
```

```
0001 (*Cette fonction retourne la position Y en fonction de l'axe X de la porte droite*)
0002 VarLocalA:=(27.0/71.0);
```

```
0003 ValXLocal:= INT_TO_REAL(PosXPorte);
0004 VarLocalA:=(VarLocalA)*(ValXLocal);
0005 FX_PorteD:=VarLocalA- 2;
```

FX_PorteG (FUN-ST)

```
0001 FUNCTION FX_PorteG : REAL
0002 VAR_INPUT
0003     PosXPorte: REAL;
0004 END_VAR
0005 VAR
0006     VarLocalA: REAL;
0007 END_VAR
```

```
0001 (*Cette fonction retourne la position Y en fonction de l'axe X de la porte gauche*)
0002 VarLocalA:=(20.0/59.0);
0003 VarLocalA:=(VarLocalA)*(PosXPorte);
0004 FX_PorteG:=VarLocalA +1;
```

VisuCabine (FUN-ST)

```
0001 FUNCTION VisuCabine : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005     TimerDeTempo: TP;
0006     CompteurPortesVisu: INT;
0007     CompteurVarVisu: INT;
0008     CompteurDeTemps: INT;
0009     Step: INT;
0010     CompteurLocal: INT;
0011     ListePresence: ARRAY [1..4] OF BOOL;
0012     ListePosCab: ARRAY [1..9,1..2] OF INT;
0013     ListePresenceET: ARRAY [1..9] OF BOOL;
0014     MoveSlowCab: INT;
0015     MoveMediumCab: INT;
0016     MoveFastCab: INT;
0017 END_VAR
```

```
0001 (* Cette fonction a pour but de  faire bouger la cabine en fonction de l'activation des moteurs*)
0002
0003 (* La cabine enclenchera les capteurs de fin de courses si elle se trouve a la position de l'etage défini*)
0004
0005 (* IL y a 3 variable de vitesse défini sur 3 zones pour que la cabine se posiitonne parfaitement à l'étage et sans le dépasser*)
0006
0007 (*VAleurs d'initialisation du visuel*)
0008 ListePresenceET[9]:=FALSE;
0009
0010 (* variable de la zone de ralentissement/ accélérationde la cabine*)
0011 ListePosCab[9,1]:= 130 ;
0012 ListePosCab[9,2]:= 90;
0013
0014 ListePosCab[1,1]:=  130;
0015 ListePosCab[1,2]:= 90;
0016 ListePosCab[2,1]:=  20;
0017 ListePosCab[2,2]:= -20;
0018 ListePosCab[3,1]:= - 90;
0019 ListePosCab[3,2]:= - 130;
0020 ListePosCab[4,1]:= - 200;
0021 ListePosCab[4,2]:= - 240;
0022
0023 ListePresenceET[1]:=PRDC;
0024 ListePresenceET[2]:=PET1;
0025 ListePresenceET[3]:=PET2;
0026 ListePresenceET[4]:=PET3;
0027
0028 (*La variable ZoneDepCab Zone de Départ Cabine sert au départ lent de la cabine*)
0029 IF ZoneDepCab = 0 THEN ZoneDepCab:= 2;END_IF
0030
0031 IF NOT ListePresenceET[ETAGE_SUIVANT] THEN
0032     IF CABD THEN
0033         MoveSlowCab:=1;
0034         MoveMediumCab:=5;
```

```
0035            MoveFastCab:=10;
0036        ELSIF CABM THEN
0037            MoveSlowCab:=-1;
0038            MoveMediumCab:=-5;
0039            MoveFastCab:=-10;
0040        END_IF
0041        IF CABD OR CABM THEN
0042            wait(IN:=TRUE, PT:=T#10ms);
0043            IF NOT wait.Q THEN
0044                IF ZoneDepCab <> ETAGE_SUIVANT THEN
0045                    IF CABmoveY > ListePosCab[ZoneDepCab,2] AND CABmoveY < ListePosCab[ZoneDepCab,1] THEN
0046                        wait(IN:=FALSE);
0047                        CABmoveY:=CABmoveY+MoveMediumCab;
0048                    END_IF
0049                END_IF
0050                IF CABmoveY > ListePosCab[ETAGE_SUIVANT,2] AND CABmoveY < ListePosCab[ETAGE_SUIVANT,1] THEN
0051                    wait(IN:=FALSE);
0052                    CABmoveY:=CABmoveY+MoveSlowCab;
0053
0054                ELSIF CABmoveY > ListePosCab[ETAGE_SUIVANT,2] -20 AND  CABmoveY <= ListePosCab[ETAGE_SUIVANT,2]  OR CABmoveY < ListeP
0055                    wait(IN:=FALSE);
0056                    CABmoveY:=CABmoveY+MoveMediumCab;
0057                ELSE
0058                    wait(IN:=FALSE);
0059                    CABmoveY:=CABmoveY+MoveFastCab;
0060                END_IF
0061            END_IF
0062        END_IF
0063    END_IF
0064
0065    IF CABmoveY = 110 THEN
0066        PRDC:=TRUE;
0067        ZoneDepCab:=1;
0068    ELSE PRDC:=FALSE;END_IF
0069
0070    IF CABmoveY = 0 THEN
0071         PET1:=TRUE;
0072        IF F1 THEN ZoneDepCab:=2;END_IF
0073    ELSE PET1:=FALSE;END_IF
0074
0075    IF CABmoveY = -110 THEN
0076        PET2:=TRUE;
0077        IF F2 THEN ZoneDepCab:=3;END_IF
0078    ELSE PET2:=FALSE;END_IF
0079
0080    IF CABmoveY = -220 THEN
0081        PET3:=TRUE;
0082        IF F3 THEN ZoneDepCab:=4;END_IF
0083    ELSE PET3:=FALSE;END_IF
0084
```

osCab[ETAGE_SUIVANT,1]+20 AND CABmoveY >= ListePosCab[ETAGE_SUIVANT,1] THEN

VisuPortesET1 (FUN-ST)

```
0001 FUNCTION VisuPortesET1 : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005     TimerDeTempo: TP;
0006     CompteurPortesVisu: INT;
0007     CompteurVarVisu: INT;
0008     CompteurDeTemps: INT;
0009     Step: INT;
0010     CompteurLocal: INT;
0011     CONTACTEUR_SEC: INT;
0012 END_VAR
```

```
0001 (* Cette fonction sert à fermer les portes de l'etage 1 en fonction des donné transmises par le porgramme principal*
0002
0003     La fonction va simuler une fermeture sur action d'un moteur, et simuler le capteur  lorsque la coordonée de position fermé est atteinte : x=0 et y=-2*)
0004 IF F1 OR O1 THEN
0005     (*On simule la position physique du contacteur avec des coordonnées*)
```

```
0006      IF F1 THEN CONTACTEUR_SEC:=-71;
0007      ELSE CONTACTEUR_SEC:=0;END_IF
0008      wait(IN:=TRUE, PT:=T#1ms);
0009      FOR CompteurLocal:=1 TO 3 BY 1 DO
0010          IF NOT wait.Q THEN
0011              wait(IN:=FALSE);
0012              (* en fonction de l'état du contacteur nous allons
0013              simuler une ouverture/fermeture de la porte*)
0014
0015              (* nous allons utiliser les fonctions de coordonnées
0016              pour que la simulation suivent une trajectoire excate et soienet parfaitement synchronisé*)
0017              IF VisuET1Dx > CONTACTEUR_SEC THEN
0018                  VisuET1Dx:=VisuET1Dx-1;
0019                  VisuET1Dy:=FX_PorteD(VisuET1Dx);
0020                  VisuET1Gx:=FX_Porte_Vitesse(VisuET1Dx);
0021                  VisuET1Gy:=FX_PorteG(VisuET1Gx);
0022              ELSIF  VisuET1Dx < CONTACTEUR_SEC THEN
0023                  VisuET1Dx:=VisuET1Dx+1;
0024                  VisuET1Dy:=FX_PorteD(VisuET1Dx);
0025                  VisuET1Gx:=FX_Porte_Vitesse(VisuET1Dx);
0026                  VisuET1Gy:=FX_PorteG(VisuET1Gx);
0027              END_IF
0028          END_IF
0029      END_FOR
0030  END_IF
0031
0032  IF VisuET1Dx = - 71  THEN P1fer:=TRUE;
0033  ELSE P1fer:=FALSE;END_IF
0034
0035  IF VisuET1Dx = 0  THEN P1ouv:=TRUE;
0036  ELSE P1ouv:=FALSE;END_IF
```

VisuPortesET2 (FUN-ST)

```
0001  FUNCTION VisuPortesET2 : BOOL
0002  VAR_INPUT
0003  END_VAR
0004  VAR
0005      TimerDeTempo: TP;
0006      CompteurPortesVisu: INT;
0007      CompteurVarVisu: INT;
0008      CompteurDeTemps: INT;
0009      Step: INT;
0010      CompteurLocal: INT;
0011      CONTACTEUR_SEC: INT;
0012  END_VAR
```

```
0001  (* Cette fonction sert à fermer les portes de l'etage 2 en fonction des donné transmises par le porgramme principal*
0002
0003      La fonction va simuler une fermeture sur action d'un moteur, et simuler le capteur  lorsque la coordonée de position fermé est atteinte : x=0 et y=-2*)
0004  IF F2 OR O2 THEN
0005      (*On simule la position physique du contacteur avec des coordonnées*)
0006      IF F2 THEN CONTACTEUR_SEC:=-71;
0007      ELSE CONTACTEUR_SEC:=0;END_IF
0008      wait(IN:=TRUE, PT:=T#1ms);
0009      FOR CompteurLocal:=1 TO 3 BY 1 DO
0010          IF NOT wait.Q THEN
0011              wait(IN:=FALSE);
0012              (* en fonction de l'état du contacteur nous allons
0013              simuler une ouverture/fermeture de la porte*)
0014
0015              (* nous allons utiliser les fonctions de coordonnées
0016              pour que la simulation suivent une trajectoire excate et soienet parfaitement synchronisé*)
0017              IF VisuET2Dx > CONTACTEUR_SEC THEN
0018                  VisuET2Dx:=VisuET2Dx-1;
0019                  VisuET2Dy:=FX_PorteD(VisuET2Dx);
0020                  VisuET2Gx:=FX_Porte_Vitesse(VisuET2Dx);
0021                  VisuET2Gy:=FX_PorteG(VisuET2Gx);
0022              ELSIF  VisuET2Dx < CONTACTEUR_SEC THEN
0023                  VisuET2Dx:=VisuET2Dx+1;
0024                  VisuET2Dy:=FX_PorteD(VisuET2Dx);
```

```
                    VisuET2Gx:=FX_Porte_Vitesse(VisuET2Dx);
                    VisuET2Gy:=FX_PorteG(VisuET2Gx);
                END_IF
            END_IF
        END_FOR
    END_IF

    IF VisuET2Dx = - 71  THEN P2fer:=TRUE;
    ELSE P2fer:=FALSE;END_IF

    IF VisuET2Dx = 0  THEN P2ouv:=TRUE;
    ELSE P2ouv:=FALSE;END_IF
```

VisuPortesET3 (FUN-ST)

```
FUNCTION VisuPortesET3 : BOOL
VAR_INPUT
END_VAR
VAR
    TimerDeTempo: TP;
    CompteurPortesVisu: INT;
    CompteurVarVisu: INT;
    CompteurDeTemps: INT;
    Step: INT;
    CompteurLocal: INT;
    CONTACTEUR_SEC: INT;
END_VAR
(* Cette fonction sert à fermer les portes de l'etage 3 en fonction des donné transmises par le porgramme principal*

    La fonction va simuler une fermeture sur action d'un moteur, et simuler le capteur  lorsque la coordonée de position fermé est atteinte : x=0 et y=-2*)
IF F3 OR O3 THEN
    (*On simule la position physique du contacteur avec des coordonnées*)
    IF F3 THEN CONTACTEUR_SEC:=-71;
    ELSE CONTACTEUR_SEC:=0;END_IF
    wait(IN:=TRUE, PT:=T#1ms);
    FOR CompteurLocal:=1 TO 3 BY 1 DO
        IF NOT wait.Q THEN
            wait(IN:=FALSE);
            (* en fonction de l'état du contacteur nous allons
            simuler une ouverture/fermeture de la porte*)

            (* nous allons utiliser les fonctions de coordonnées
            pour que la simulation suivent une trajectoire excate et soienet parfaitement synchronisé*)
            IF VisuET3Dx > CONTACTEUR_SEC THEN
                VisuET3Dx:=VisuET3Dx-1;
                VisuET3Dy:=FX_PorteD(VisuET3Dx);
                VisuET3Gx:=FX_Porte_Vitesse(VisuET3Dx);
                VisuET3Gy:=FX_PorteG(VisuET3Gx);
            ELSIF  VisuET3Dx < CONTACTEUR_SEC THEN
                VisuET3Dx:=VisuET3Dx+1;
                VisuET3Dy:=FX_PorteD(VisuET3Dx);
                VisuET3Gx:=FX_Porte_Vitesse(VisuET3Dx);
                VisuET3Gy:=FX_PorteG(VisuET3Gx);
            END_IF
        END_IF
    END_FOR
END_IF

IF VisuET3Dx = - 71  THEN P3fer:=TRUE;
ELSE P3fer:=FALSE;END_IF

IF VisuET3Dx = 0  THEN P3ouv:=TRUE;
ELSE P3ouv:=FALSE;END_IF
```

VisuPortesRDC (FUN-ST)

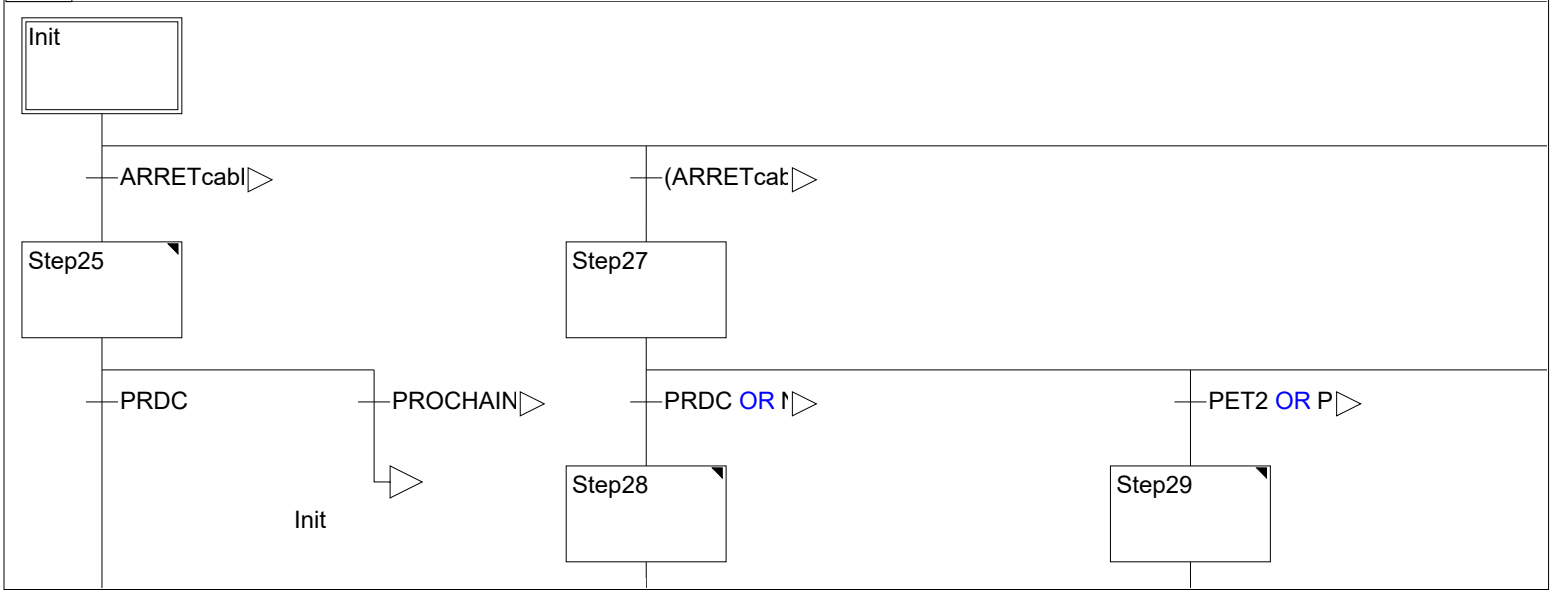
```
FUNCTION VisuPortesRDC : BOOL
VAR_INPUT
END_VAR
VAR
    TimerDeTempo: TP;
    CompteurPortesVisu: INT;
```

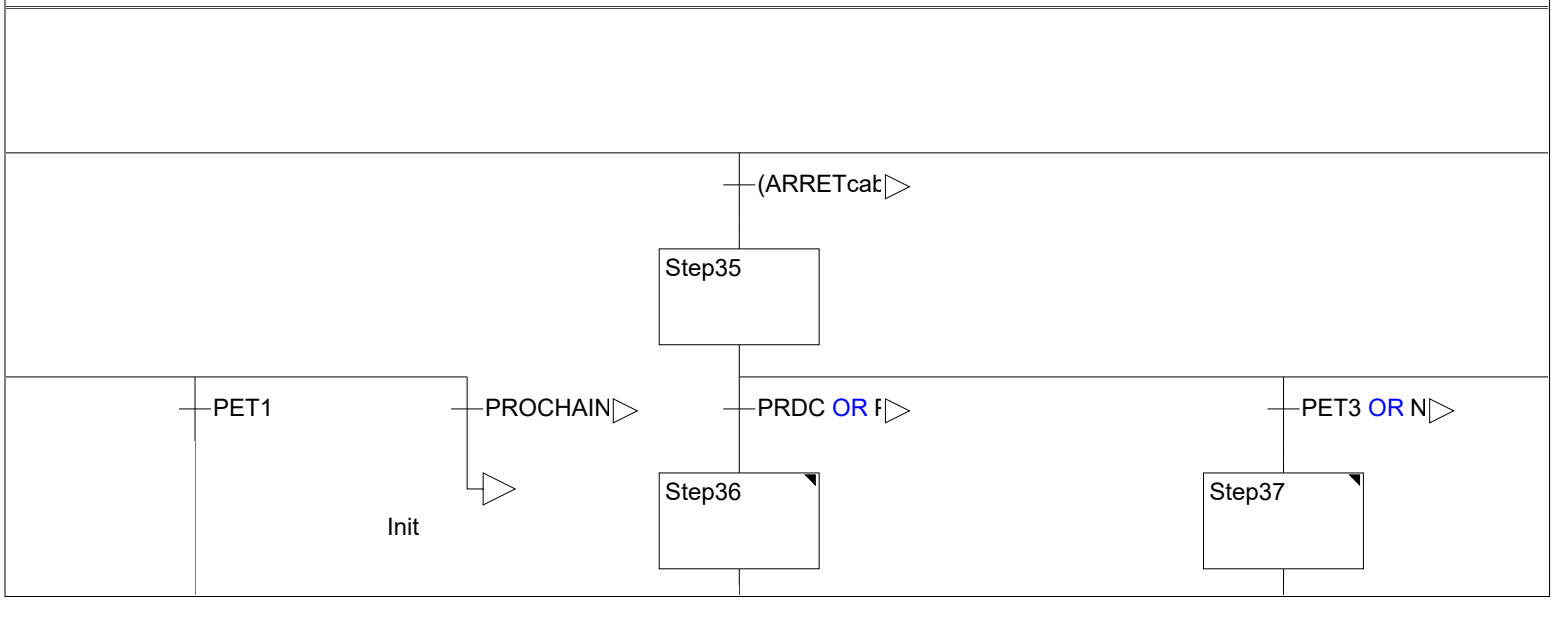
```
0007      CompteurVarVisu: INT;
0008      CompteurDeTemps: INT;
0009      Step: INT;
0010      CompteurLocal: INT;
0011      CONTACTEUR_SEC: INT;
0012  END_VAR
```

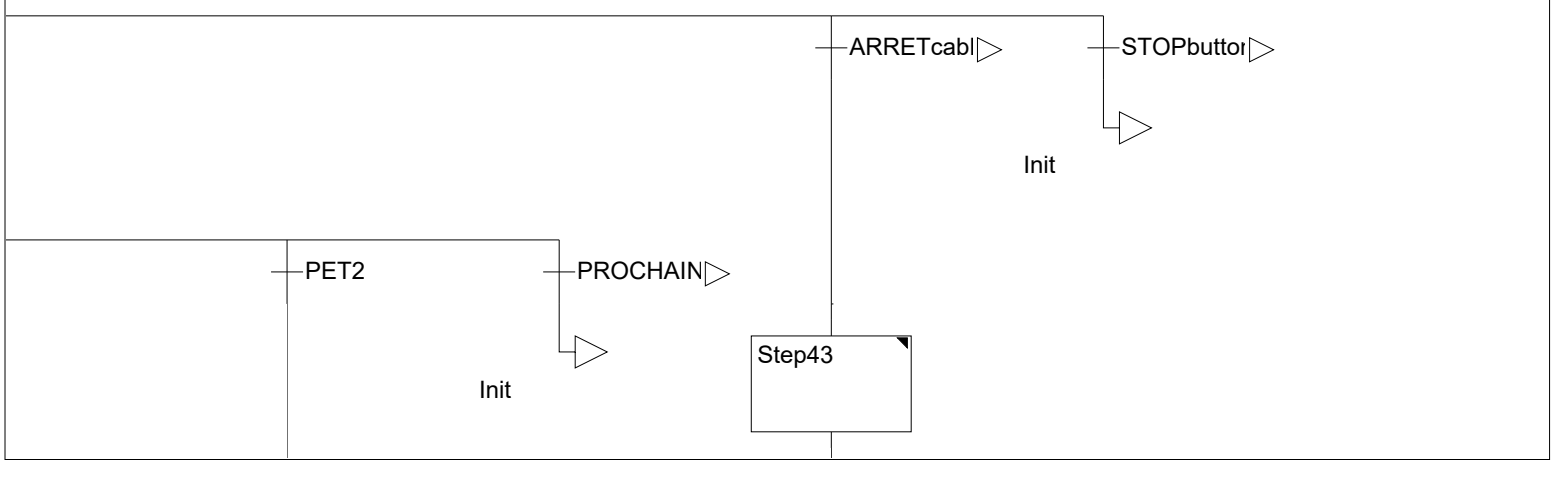
```
0001  (* Cette fonction sert à fermer les portes du rez de chaussé en fonction des donné transmises par le porgramme principal*
0002
0003      La fonction va simuler une fermeture sur action d'un moteur, et simuler le capteur  lorsque la coordonée de position fermé est atteinte : x=0 et y=-2*)
0004  IF FRDC OR ORDC THEN
0005      (*On simule la position physique du contacteur avec des coordonnées*)
0006      IF FRDC THEN CONTACTEUR_SEC:=-71;
0007      ELSE CONTACTEUR_SEC:=0;END_IF
0008      wait(IN:=TRUE, PT:=T#1ms);
0009      FOR CompteurLocal:=1 TO 3 BY 1 DO
0010          IF NOT wait.Q THEN
0011              wait(IN:=FALSE);
0012              (* en fonction de l'état du contacteur nous allons
0013              simuler une ouverture/fermeture de la porte*)
0014
0015              (* nous allons utiliser les fonctions de coordonnées
0016              pour que la simulation suivent une trajectoire excate et soienet parfaitement synchronisé*)
0017              IF VisuRDCDx > CONTACTEUR_SEC THEN
0018                  VisuRDCDx:=VisuRDCDx-1;
0019                  VisuRDCDy:=FX_PorteD(VisuRDCDx);
0020                  VisuRDCGx:=FX_Porte_Vitesse(VisuRDCDx);
0021                  VisuRDCGy:=FX_PorteG(VisuRDCGx);
0022              ELSIF  VisuRDCDx < CONTACTEUR_SEC THEN
0023                  VisuRDCDx:=VisuRDCDx+1;
0024                  VisuRDCDy:=FX_PorteD(VisuRDCDx);
0025                  VisuRDCGx:=FX_Porte_Vitesse(VisuRDCDx);
0026                  VisuRDCGy:=FX_PorteG(VisuRDCGx);
0027              END_IF
0028          END_IF
0029      END_FOR
0030  END_IF
0031
0032  IF VisuRDCDx = - 71  THEN PRDCfer:=TRUE;
0033  ELSE PRDCfer:=FALSE;END_IF
0034
0035  IF VisuRDCDx = 0  THEN PRDCouv:=TRUE;
0036  ELSE PRDCouv:=FALSE;END_IF
```

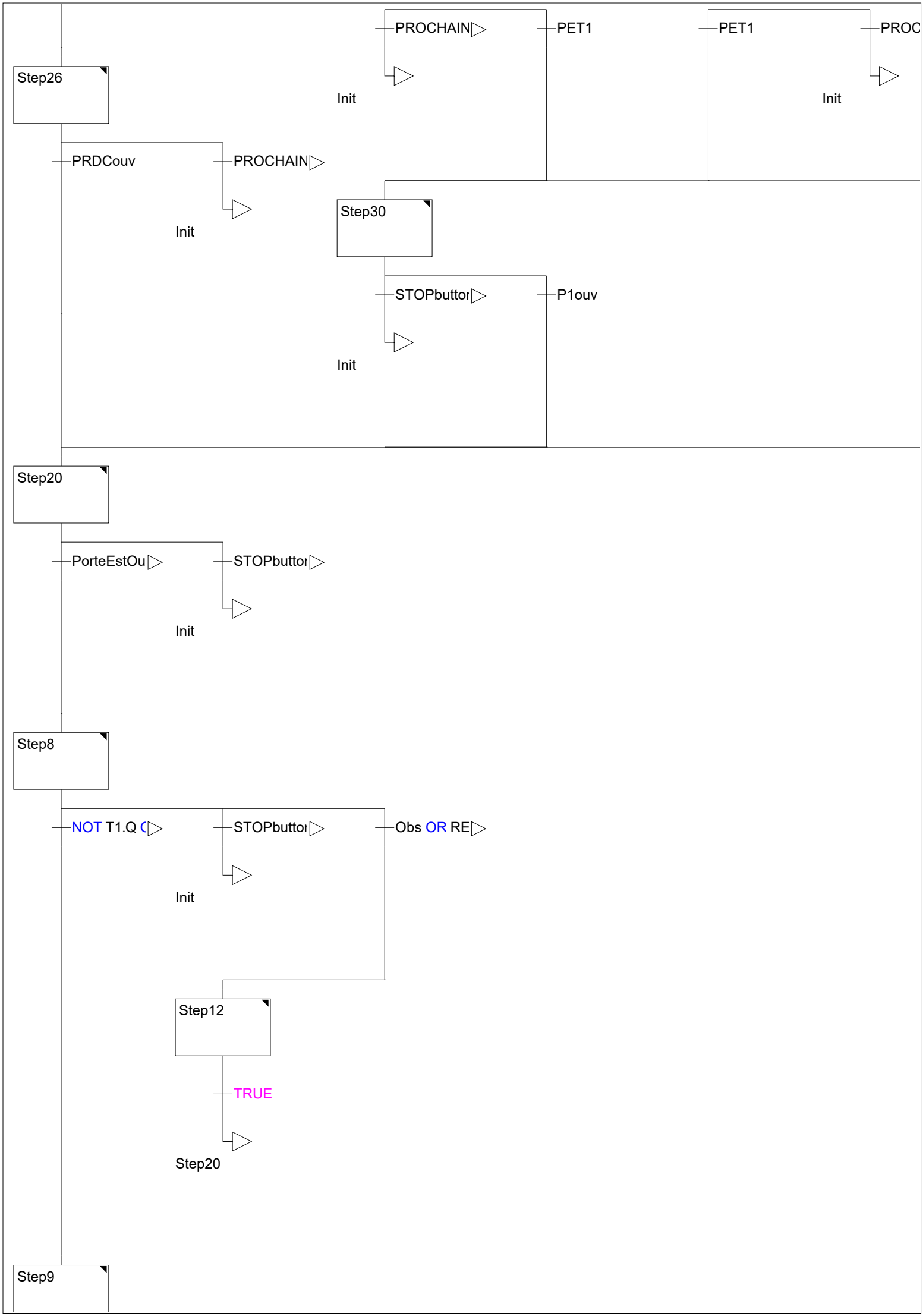
PAL_COMMUN_ETAGE (PRG-SFC)

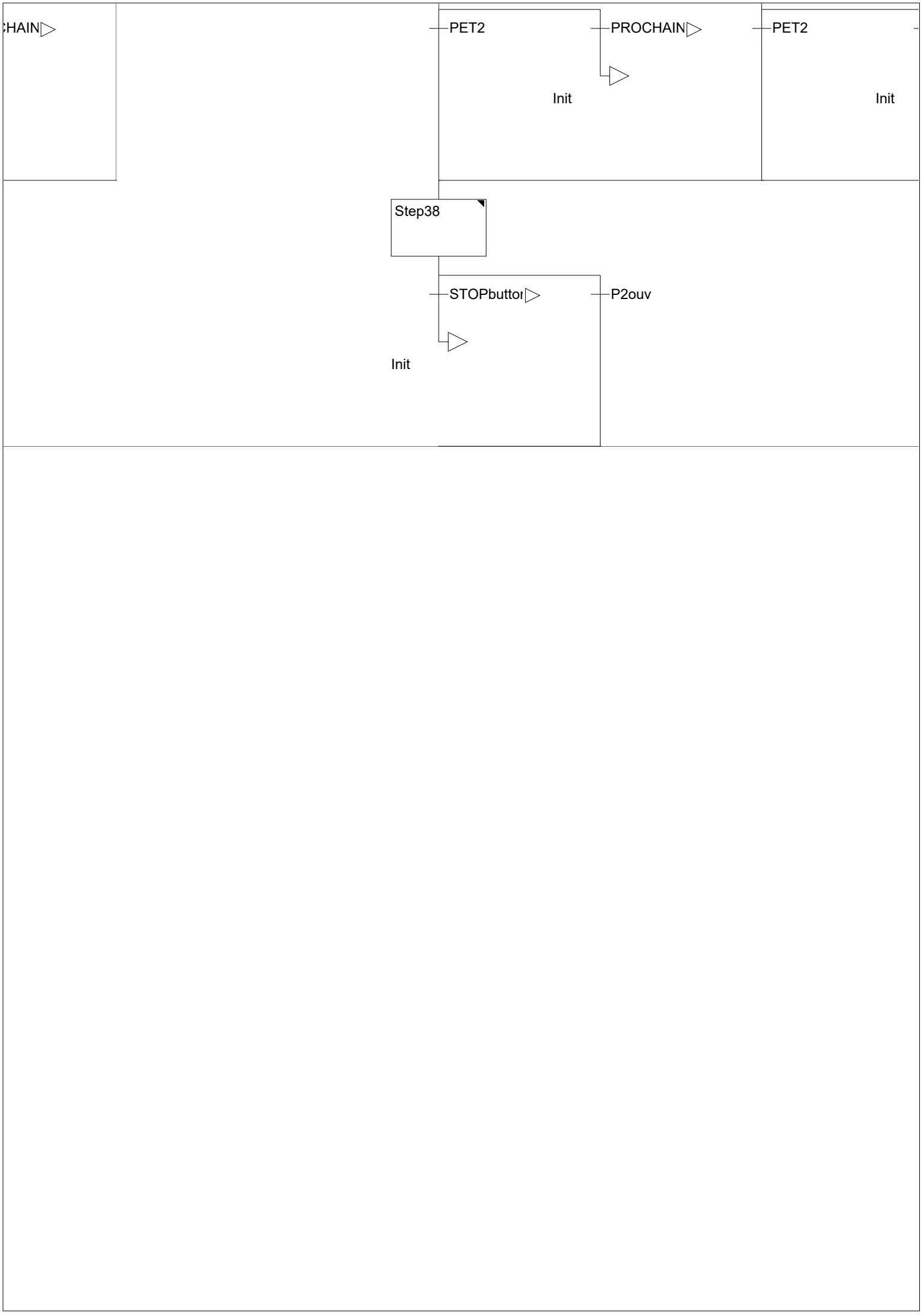
```
0001  PROGRAM PAL_COMMUN_ETAGE
0002  VAR
0003      ARRET_EN_COURS: INT;
0004      ATTEND_UN_PEU: BOOL;
0005      T5: TP;
0006  END_VAR
0007  (*AND ETAGE_SUIVANT = 2 AND NOT SENS_M*)
```

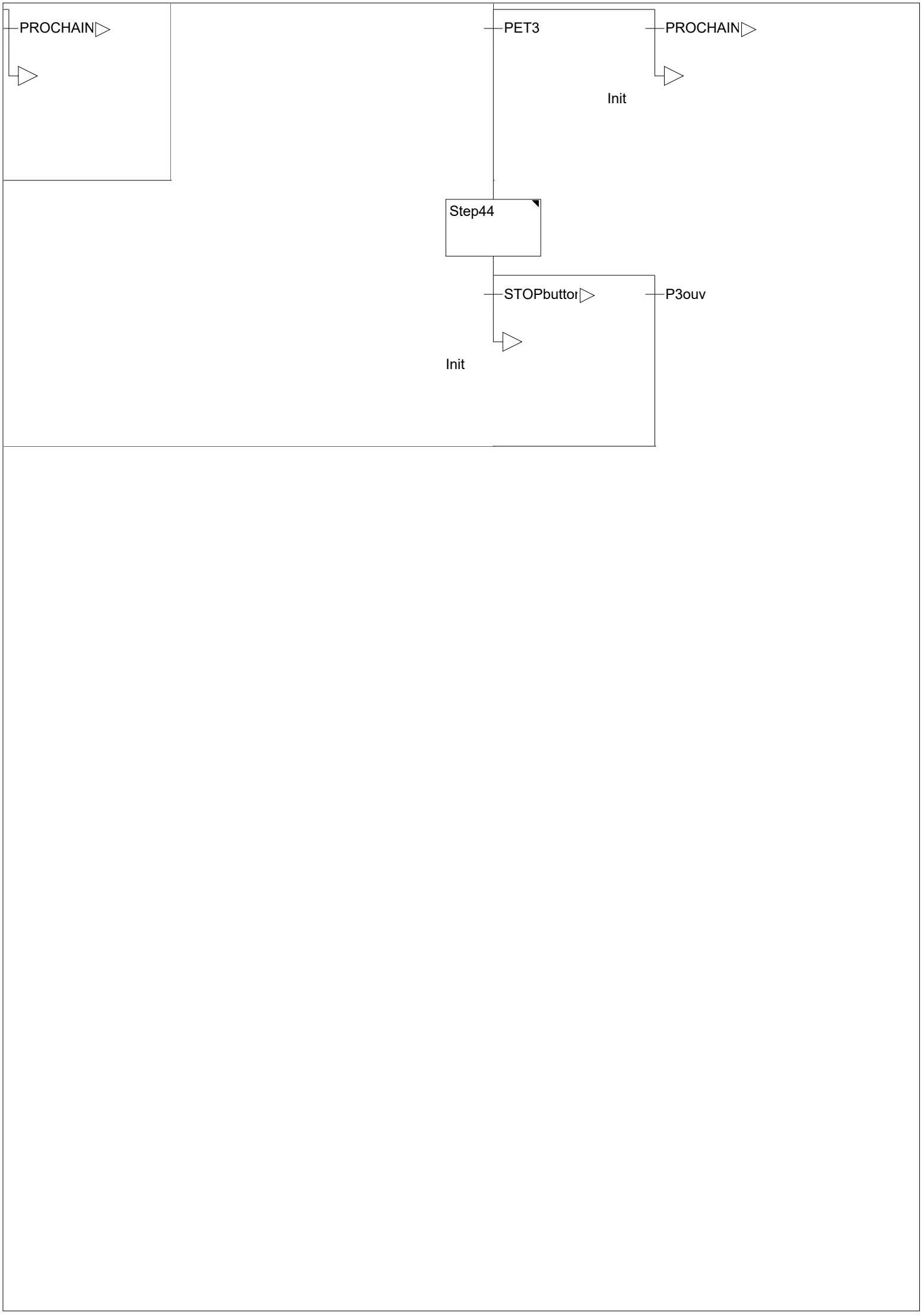


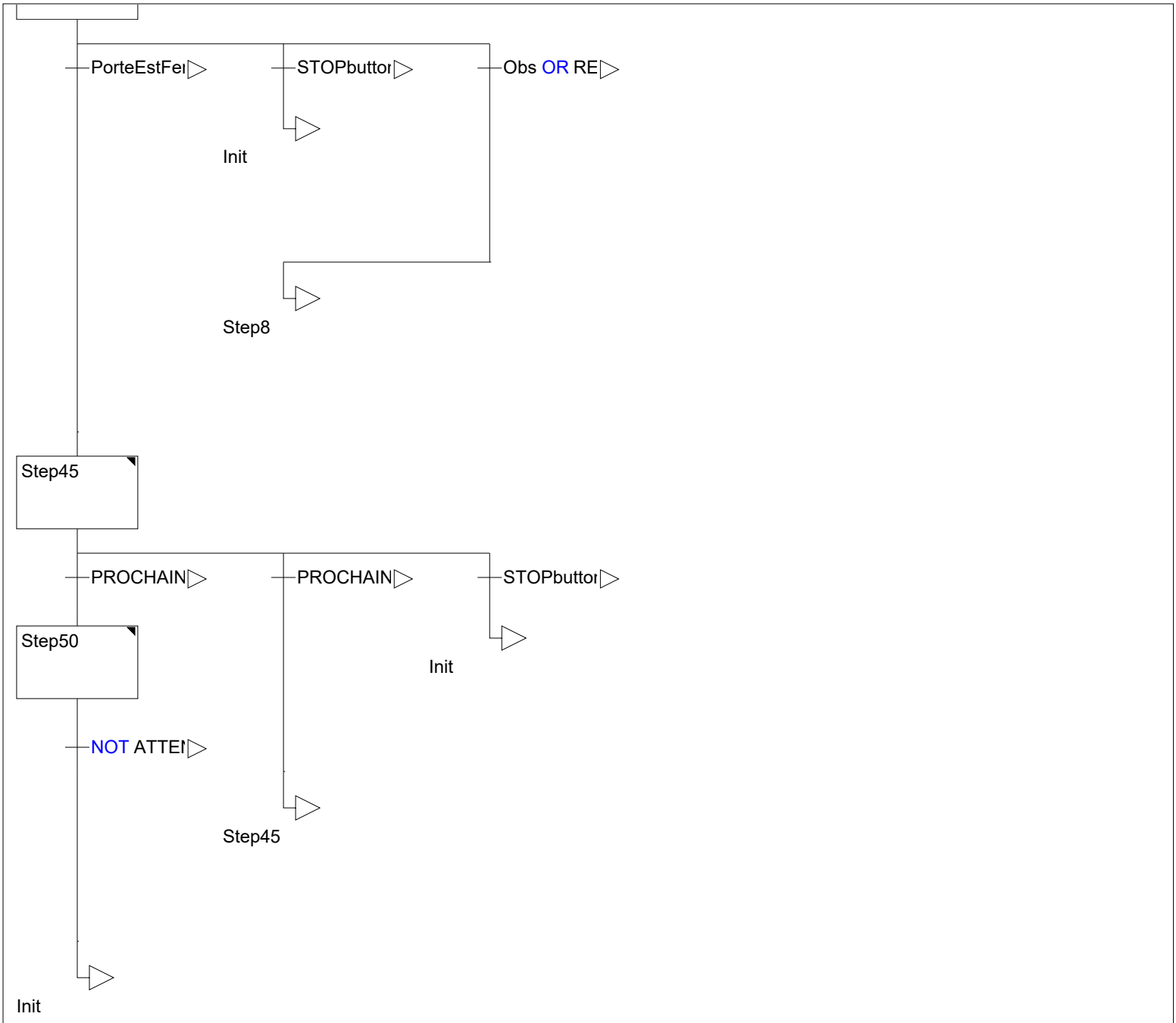

(ARRETcat
Step35
PET1
PROCHAIN
PRDC OR F
PET3 OR N
Step36
Step37
Init


ARRETcabl
STOPbuttor
Init
PET2
PROCHAIN
Step43
Init


PROCHAIN
PET1
Init
PET1
PROC
Init
Step26
PRDCouv
PROCHAIN
Init
Step30
STOPbuttor
P1ouv
Init
Step20
PorteEstOu
STOPbuttor
Init
Step8
NOT T1.Q (
STOPbuttor
Obs OR RE
Init
Step12
TRUE
Step20
Step9


CHAIN
PET2
PROCHAIN
PET2
Init
Init
Step38
STOPbuttor
P2ouv
Init


PROCHAIN
PET3
PROCHAIN
Init
Step44
STOPbuttor
P3ouv
Init


PorteEstFer
STOPbuttor
Obs OR RE
Init
Step8
Step45
PROCHAIN
PROCHAIN
STOPbuttor
Step50
Init
NOT ATTEI
Step45
Init

PAL_COMMUN_ETAGE (PRG-SFC).Action Step25 (ST)

```
0001 CABD:=TRUE;
0002 CABM:=FALSE;
```

PAL_COMMUN_ETAGE (PRG-SFC).Action Step26 (ST)

```
0001 CABD:=FALSE;
0002 ORDC:=TRUE;
0003 CABRDC:=FALSE;
0004 ARRET_EN_COURS:=PROCHAIN_ARRET;
0005 Reset_Arrets_CAB(ARRET_EN_COURS);
```

PAL_COMMUN_ETAGE (PRG-SFC).Action Step28 (ST)

```
0001 CABM:=TRUE;
0002 CABD:=FALSE;
```

PAL_COMMUN_ETAGE (PRG-SFC).Action Step29 (ST)

```
0001 CABD:=TRUE;
0002 CABM:=FALSE;
```

PAL_COMMUN_ETAGE (PRG-SFC).Action Step30 (ST)

```
0001 ARRET_EN_COURS:=PROCHAIN_ARRET;
0002 CABM:=FALSE;
0003 CABD:=FALSE;
0004 O1:=TRUE;
0005 CABET1:=FALSE;
0006 Reset_Arrets_CAB(ARRET_EN_COURS);
```

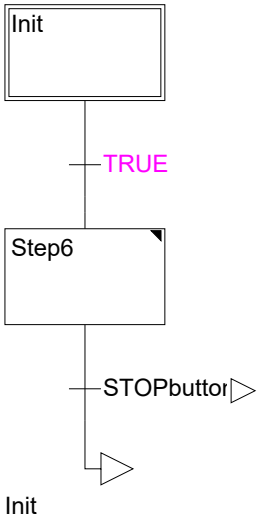
PAL_COMMUN_ETAGE (PRG-SFC).Action Step36 (ST)

```
0001 CABM:=TRUE;
0002 CABD:=FALSE;
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step37 (ST)
```
0001 CABD:=TRUE;
0002 CABM:=FALSE;
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step38 (ST)
```
0001 ARRET_EN_COURS:=PROCHAIN_ARRET;
0002 CABM:=FALSE;
0003 CABD:=FALSE;
0004 O2:=TRUE;
0005 CABET2:=FALSE;
0006 Reset_Arrets_CAB(ARRET_EN_COURS);
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step43 (ST)
```
0001 CABM:=TRUE;
0002 CABD:=FALSE;
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step44 (ST)
```
0001 CABM:=FALSE;
0002 O3:=TRUE;
0003 CABET3:=FALSE;
0004 ARRET_EN_COURS:=PROCHAIN_ARRET;
0005 Reset_Arrets_CAB(ARRET_EN_COURS);
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step20 (ST)
```
0001 T1(IN:=FALSE);
0002 RE_OUVRE:=FALSE;
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step8 (ST)
```
0001 (*Switch Button
0002 IF P1fer THEN P1fer:=FALSE;
0003 END_IF;
0004 *)
0005 ORDC:=FALSE;
0006 O1:=FALSE;
0007 O2:=FALSE;
0008 O3:=FALSE;
0009 T1(IN:=TRUE, PT:=t#2s);
0010 Re_ouverture();
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step12 (ST)
```
0001 IF NOT PorteEstOuverte() THEN Ouv_Fer_Etage(TRUE);
0002 END_IF
0003 FRDC:=FALSE;
0004 F1:=FALSE;
0005 F2:=FALSE;
0006 F3:=FALSE;
0007 T1(IN:=FALSE);
0008 T1(IN:=TRUE, PT:=t#1ms);
0009 RE_OUVRE:=FALSE;
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step9 (ST)
```
0001 (*IF NOT Obs THEN F1:=TRUE;*)
0002 Re_ouverture();
0003 IF NOT Obs THEN
0004     Ouv_Fer_Etage(FALSE);
0005 END_IF
0006 T1(IN:=FALSE);
0007 T5(IN:=FALSE);
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step45 (ST)
```
0001 (* Switch Button
0002 IF P1ouv THEN P1ouv:=FALSE;
0003 END_IF*)
0004 FRDC:=FALSE;
0005 F1:=FALSE;
0006 F2:=FALSE;
0007 F3:=FALSE;
0008 RE_OUVRE:=FALSE;
0009 IF PROCHAIN_ARRET <>  9 THEN Reset_Arrets(ARRET_EN_COURS);END_IF
0010
0011
```
PAL_COMMUN_ETAGE (PRG-SFC).Action Step50 (ST)

```
0001 (*Fonction pour l'attente d'appel dans la cabien avant un changelent de sens*)
0002
0003 ATTEND_UN_PEU:=FALSE;
0004 IF PROCHAIN_ARRET <> 0 AND PROCHAIN_ARRET <> 5 THEN
0005     IF PROCHAIN_ARRET = ARRET_EN_COURS THEN
0006         ATTEND_UN_PEU:=TRUE;
0007     ELSE
0008         IF PROCHAIN_ARRET - ARRET_EN_COURS = 1 OR ARRET_EN_COURS - PROCHAIN_ARRET = 1 THEN
0009             IF PROCHAIN_ARRET MOD 2 = 1 THEN
0010                 IF PROCHAIN_ARRET < ARRET_EN_COURS THEN ATTEND_UN_PEU:=TRUE;END_IF
0011             ELSE
0012                 IF PROCHAIN_ARRET > ARRET_EN_COURS THEN ATTEND_UN_PEU:=TRUE;END_IF
0013             END_IF
0014         END_IF
0015     END_IF
0016 END_IF
0017 T5(IN:=TRUE,PT:=t#5s);
0018
0019 IF ATTEND_UN_PEU THEN
0020     ETAGE_SUIVANT:=9;
0021     PROCHAIN_ARRET:=9;
0022     IF ARRET_EN_COURS MOD 2 = 1 THEN SENS_M:=FALSE;
0023     ELSE SENS_M:=TRUE;END_IF
0024 END_IF
```

PLC_PRG (PRG-SFC)

```
0001 PROGRAM PLC_PRG
0002 VAR
0003 END_VAR
```

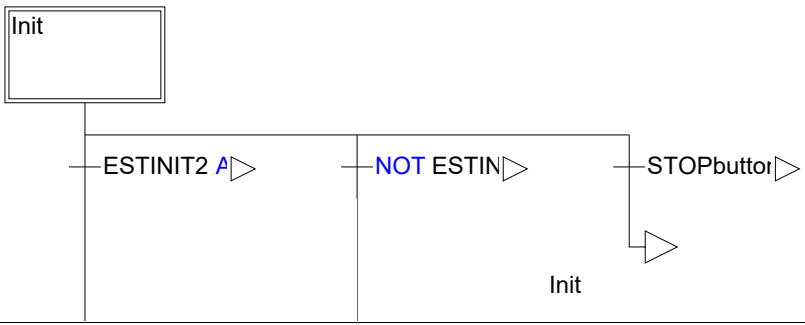


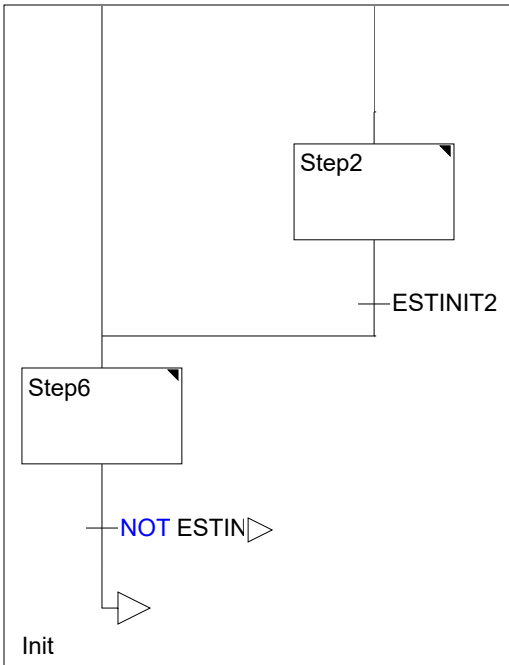
PLC_PRG (PRG-SFC).Action Step6 (ST)

```
0001 PRG_MAIN;
0002 Quel_Etage();
0003 Priorite_Memo();
0004 Appel_Palier();
0005 Arret_Urgence();
0006 ZAuto_RUN();
```

PRG_MAIN (PRG-SFC)

```
0001 PROGRAM PRG_MAIN
0002 VAR
0003 END_VAR
```





PRG_MAIN (PRG-SFC).Action Step2 (ST)

```
0001 IF StanbyActive THEN Correspondance_Etages ();END_IF
0002 Initialisation_reset();
0003 SENS_M:=TRUE;
```

PRG_MAIN (PRG-SFC).Action Step6 (ST)

```
0001 PAL_COMMUN_ETAGE;
0002 StandBY;
0003 Clignotement;
0004 STOP_PORTES();
0005 Correspondance_Etages();
```

Priorite_Memo (FUN-ST)

```
0001 FUNCTION Priorite_Memo : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005     CompteurLocal: INT;
0006     ETAGE_GOOD: INT;
0007 END_VAR
```

```
0001 (*Cette fonction pour but de gérer les priorité en fonction de la mémorisation*)
0002
0003 (* Création d'un tableau pour la mémorisation des appels dans le sens montée*)
0004 TableDeVeriteMontee[1]:=ARRETcabRDC;
0005 TableDeVeriteMontee[2]:=ARRETcabM1;
0006 TableDeVeriteMontee[3]:=ARRETcabM2;
0007 TableDeVeriteMontee[4]:=ARRETcabET3;
0008
0009 (*Création d'un tableau pour a mémorisation des appels dans le sens de la descente*)
0010 TableDeVeriteDesc[1]:=ARRETcabRDC;
0011 TableDeVeriteDesc[2]:=ARRETcabD1;
0012 TableDeVeriteDesc[3]:=ARRETcabD2;
0013 TableDeVeriteDesc[4]:=ARRETcabET3;
0014
0015 (*Correspondance des tableaux précedent avec le numéro du bouton*)
0016 ArretsDesc[1]:=0;(*Rez de chaussé*)
0017 ArretsDesc[2]:=1;(*etage 1 descente*)
0018 ArretsDesc[3]:=3;(*etage 2 descente*)
0019 ArretsDesc[4]:=5;(*etage 3 *)
0020
0021 ArretsMonte[1]:=0;(*Rez de chaussé*)
0022 ArretsMonte[2]:=2;(*etage 1 montée*)
0023 ArretsMonte[3]:=4;(*etage 2 montée*)
0024 ArretsMonte[4]:=5;(*etage 3 *)
0025
0026 IF NOT CABM AND NOT CABD AND NOT PRDC AND NOT PET1 AND NOT PET2 AND NOT PET3 THEN
0027     Niveau_Actuel:=Last_Capt_Pres();
```

```
0028 ELSE
0029     Niveau_Actuel:= Last_Capt_Pres2();
0030 END_IF
0031 (*IF Ancien_Niveau <> 0 THEN Niveau_Actuel:= Ancien_Niveau;END_IF*)
0032
0033 (*On test si un des appels de la mémorisation est true*)
0034 IF ARRETcabRDC OR ARRETcabD1 OR ARRETcabM1 OR ARRETcabD2 OR ARRETcabM2 OR ARRETcabET3 THEN
0035
0036 (*La gestion de la mémorisation suivante va retourner trois variables :
0037     -Le niveau de sortie du palier (1;2;3;4)
0038     -Le niveau de sortie en fonction du bouton (RDC; PALD1,PALM1,PALD2,etc
0039     -Le sens de l'ascenseur
0040
0041 La gestion de la priorité se base sur le sens de l'ascenceur et le niveau auquel il se situe.
0042 Après reflexion les fonctions sont divisés pour couvrir toutes les combinaisons possibles avec 4 CAS Classé par ordre depriorité  (Prio CAS1 > Prio CAS2 etc
0043
0044     CAS 1 : L'étage se situe dans le bon sens et peut etre deservie ensuite (ex: PET1 + PALM2 + SENS montée -> TRUE);
0045     CAS 2 : L'étage se situe dans l'autre sens et l'asenceur doit changer de sens (ex: PET1 + PRDC + SENS montée -> Changement de sens puis CAS 1
0046     CAS 3: L'étage se situe dans le bon sens mais l'ascenceur doit effectuer un demi tour (ex PET2 + PALM2 + SENS montée -> demi tour puis desserte d
0047     CAS 4 :  l'étage se situe dans l'autre sens mais l'asenceur doit effectuer un demi tour pour y acceder ( ex PET1 +SENSmontée + PALD2 -> Demi tour p
0048
0049 Avec ces 4 cas nous pouvons faire face à toutes les situations et gérer les déplacements le plus fluidement possible. *)
0050
0051     IF Niveau_Actuel > 0 THEN
0052         (*————————————*)
0053         (* SENS —— MONTEE*)
0054         (*————————————*)
0055         IF SENS_M THEN
0056             (* On test l'étage le plus proche dans le bon sens*)
0057             FOR CompteurLocal:=4 TO Niveau_Actuel BY-1 DO
0058                 IF TableDeVeriteMontee[CompteurLocal] THEN
0059                     ETAGE_GOOD:= CompteurLocal;
0060                 END_IF
0061             END_FOR
0062             IF ETAGE_GOOD<> 0 THEN ETAGE_SUIVANT:=ETAGE_GOOD;END_IF
0063             (*Si pas de résulats dans le sens*)
0064             (*Alors on teste dans l'autre sens*)
0065             IF ETAGE_SUIVANT = 9 THEN
0066                 FOR CompteurLocal:=1 TO Niveau_Actuel BY 1 DO
0067                     IF TableDeVeriteDesc[CompteurLocal] THEN
0068                         ETAGE_SUIVANT:= CompteurLocal;
0069                         SENS_M:=FALSE;
0070                     END_IF
0071                 END_FOR
0072             END_IF
0073             (*Si toujours pas de résulats dans le sens*)
0074             (*Alors on teste l'ancien sens mais inversé au niveau de l'étage*)
0075             IF ETAGE_SUIVANT = 9 THEN
0076                 FOR CompteurLocal:=1 TO Niveau_Actuel BY 1 DO
0077                     IF TableDeVeriteMontee[CompteurLocal] THEN
0078                         ETAGE_SUIVANT:= CompteurLocal;
0079                     END_IF
0080                 END_FOR
0081             END_IF
0082             (*Si toujours pas de résulats dans le sens mais inversé*)
0083             (*Alors on teste l''autre sens mais inversé au niveau de l'étage*)
0084             IF ETAGE_SUIVANT = 9 THEN
0085                 FOR CompteurLocal:=4 TO Niveau_Actuel BY-1 DO
0086                     IF TableDeVeriteDesc[CompteurLocal] THEN
0087                         ETAGE_SUIVANT:= CompteurLocal;
0088                         SENS_M:=FALSE;
0089                     END_IF
0090                 END_FOR
0091             END_IF
0092         (*————————————*)
0093         (* SENS —— DESCENTE*)
0094         (*————————————*)
0095         ELSIF NOT SENS_M THEN
```

... :

e l'étage
uis changement de sens

```
0096            (* On test l'étage le plus proche dans le bon sens*)
0097            FOR CompteurLocal:=1 TO Niveau_Actuel BY 1 DO
0098                IF TableDeVeriteDesc[CompteurLocal] THEN
0099                    ETAGE_GOOD:= CompteurLocal;
0100                END_IF
0101            END_FOR
0102            IF ETAGE_GOOD<> 0 THEN ETAGE_SUIVANT:=ETAGE_GOOD;END_IF
0103            (*Si pas de résulats dans le sens*)
0104            (*Alors on teste dans l'autre sens*)
0105            IF ETAGE_SUIVANT = 9 THEN
0106                FOR CompteurLocal:=4 TO Niveau_Actuel BY -1 DO
0107                    IF TableDeVeriteMontee[CompteurLocal] THEN
0108                        ETAGE_SUIVANT:= CompteurLocal;
0109                        SENS_M:=TRUE;
0110                    END_IF
0111                END_FOR
0112            END_IF
0113            (*Si toujours pas de résulats dans le sens*)
0114            (*Alors on teste l'ancien sens mais inversé au niveau de l'étage*)
0115            IF ETAGE_SUIVANT = 9 THEN
0116                FOR CompteurLocal:=4 TO Niveau_Actuel BY -1 DO
0117                    IF TableDeVeriteDesc[CompteurLocal] THEN
0118                        ETAGE_SUIVANT:= CompteurLocal;
0119                    END_IF
0120                END_FOR
0121            END_IF
0122            (*Si toujours pas de résulats dans le sens mais inversé*)
0123            (*Alors on teste l"autre sens mais inversé au niveau de l'étage*)
0124            IF ETAGE_SUIVANT = 9 THEN
0125                FOR CompteurLocal:=1 TO Niveau_Actuel BY 1 DO
0126                    IF TableDeVeriteMontee[CompteurLocal] THEN
0127                        ETAGE_SUIVANT:= CompteurLocal;
0128                        SENS_M:=TRUE;
0129                    END_IF
0130                END_FOR
0131            END_IF
0132        END_IF
0133        (*————————————————*)
0134        (*ASSIGNATION DU RESULTAT*)
0135        (*————————————————*)
0136 (*Récupération du résultat en fonction des listes prélablemen créer*)
0137        IF ETAGE_SUIVANT < 9 AND SENS_M THEN
0138            PROCHAIN_ARRET:=ArretsMonte[ETAGE_SUIVANT];
0139        ELSIF ETAGE_SUIVANT < 9 AND NOT SENS_M THEN
0140            PROCHAIN_ARRET:=ArretsDesc[ETAGE_SUIVANT];
0141        END_IF
0142    END_IF
0143 END_IF
0144
```

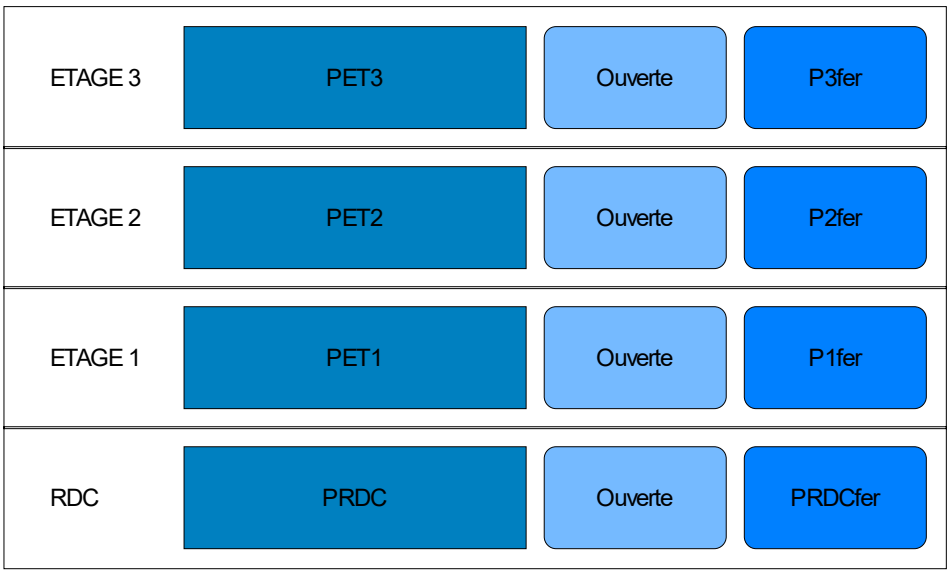
ZAuto_RUN (FUN-ST)

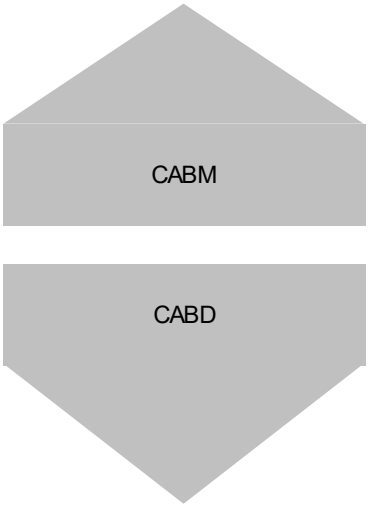
```
0001  FUNCTION ZAuto_RUN : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005     TimerCab:TP;
0006     TimerPortes: TP;
0007 END_VAR
```

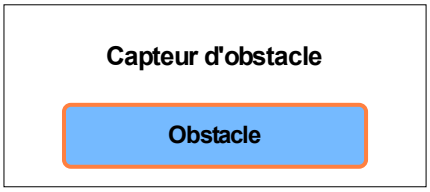
```
0001 (*Cette fonction permet de faire fonctionner les moteurs de la simulation*)
0002
0003 (* Init Porte*)
0004 IF NOT InitPortesAscSalon THEN
0005     (*Fermeture des portes
0006 et Instanciation d'un offset pour aligner le compteur d'étage visuel*)
0007     VisuRDCDx:=-60;
0008     VisuRDCDy:=FX_PorteD(VisuRDCDx);
0009     VisuRDCGx:=FX_Porte_Vitesse(VisuRDCDx);
0010     VisuRDCGy:=FX_PorteG(VisuRDCGx);
0011
```

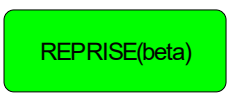
```
0012      VisuET1Dx:=-60;
0013      VisuET1Dy:=FX_PorteD(VisuET1Dx);
0014      VisuET1Gx:=FX_Porte_Vitesse(VisuET1Dx);
0015      VisuET1Gy:=FX_PorteG(VisuET1Gx);
0016
0017      VisuET2Dx:=-60;
0018      VisuET2Dy:=FX_PorteD(VisuET2Dx);
0019      VisuET2Gx:=FX_Porte_Vitesse(VisuET2Dx);
0020      VisuET2Gy:=FX_PorteG(VisuET2Gx);
0021
0022      VisuET3Dx:=-60;
0023      VisuET3Dy:=FX_PorteD(VisuET3Dx);
0024      VisuET3Gx:=FX_Porte_Vitesse(VisuET3Dx);
0025      VisuET3Gy:=FX_PorteG(VisuET3Gx);
0026
0027      InitPortesAscSalon:=TRUE;
0028      DisplayNumbET1:=TRUE;
0029      Offset_ChiffreY:=-5;
0030  END_IF
0031  (*Affichage du compteur d'étage visuel*)
0032  Afficher_Etage_Range();
0033
0034  (*Appel des fonctions de mouvement en focntion des moteurs*)
0035  IF FRDC OR ORDC THEN VisuPortesRDC();END_IF
0036  IF F1 OR O1 THEN VisuPortesET1();END_IF
0037  IF F2 OR O2 THEN VisuPortesET2();END_IF
0038  IF F3 OR O3 THEN VisuPortesET3();END_IF
0039  IF CABD OR CABM THEN VisuCabine();END_IF
```

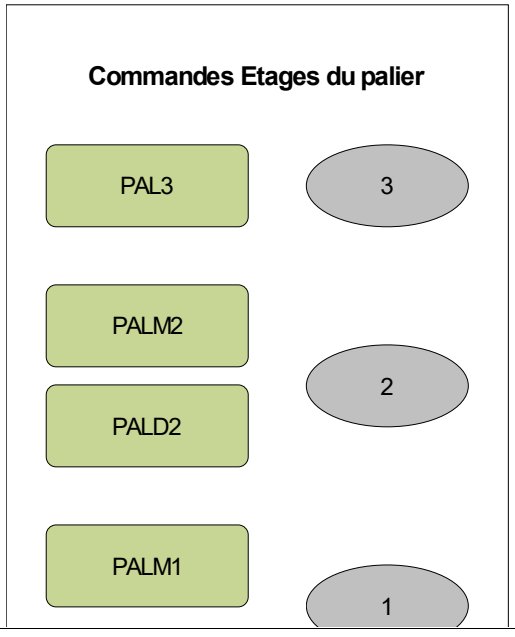
Visu

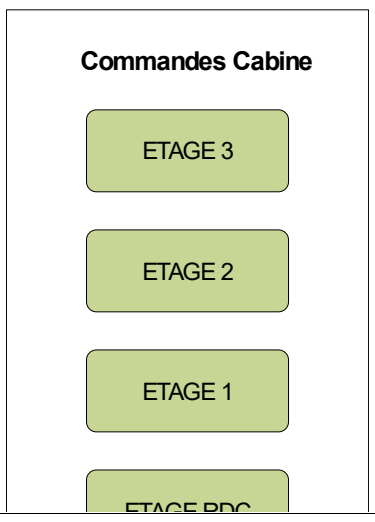












VPAL3

VPAL2

VPAL1

VPALRDC

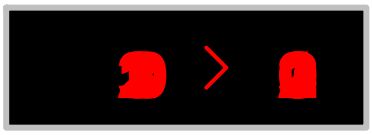
Voyants Etages

Voyants Moteurs Porte

O3 F3

O2 F2

O1 F1

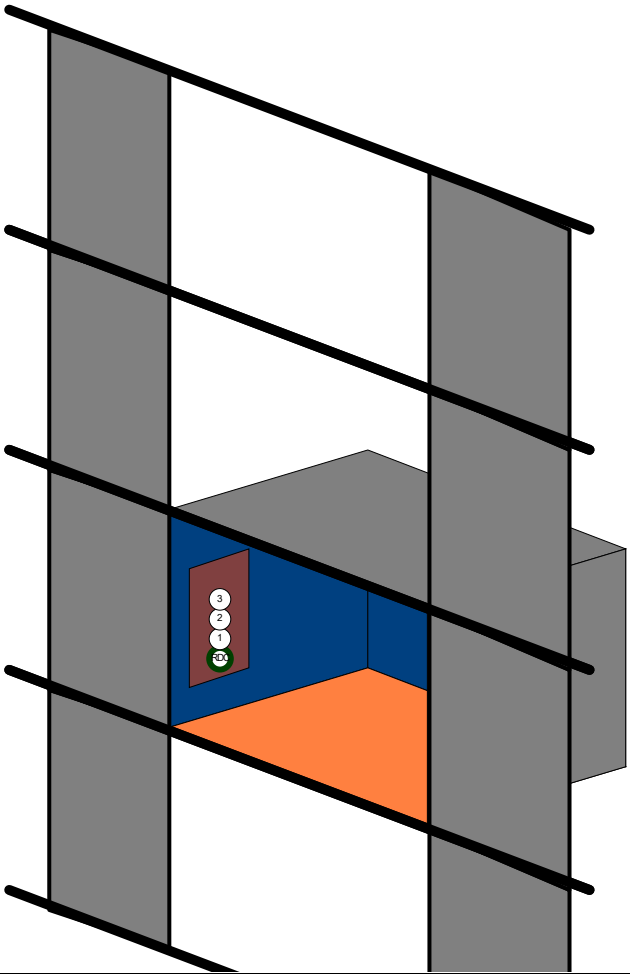


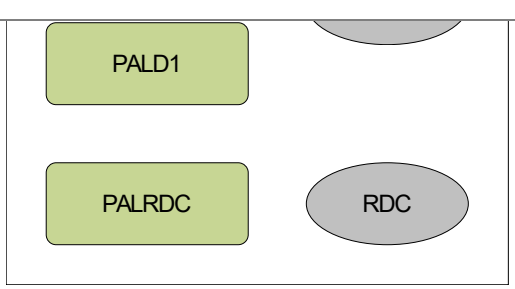
ETAGE 3

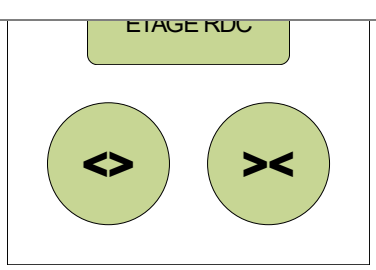
ETAGE 2

ETAGE 1

ETAGE RDC







Globale_Variablen

```
0001 VAR_GLOBAL
0002 END_VAR
```

Variables_Configuration

```
0001 VAR_CONFIG
0002 END_VAR
```

Variables_Globales

```
0001 VAR_GLOBAL
0002 (* ————————————
0003     Entrée
0004 ————————————*)
0005     (*Présence étage*)
0006     PRDC: BOOL;
0007     PET1: BOOL;
0008     PET2: BOOL;
0009     PET3: BOOL;
0010
0011     (*Appel à partir du palier de l'étage*)
0012     PALRDC: BOOL;
0013     PALD1: BOOL;
0014     PALM1: BOOL;
0015     PALD2: BOOL;
0016     PALM2: BOOL;
0017     PAL3: BOOL;
0018
0019     (*Clignotement d'Appel à partir du palier de l'étage*)
0020     PALRDCcligno: BOOL;
0021     PALD1cligno: BOOL;
0022     PALM1cligno: BOOL;
0023     PALD2cligno: BOOL;
0024     PALM2cligno: BOOL;
0025     PAL3cligno: BOOL;
0026
0027     (*Appel à partir de la cabine*)
0028     CABRDC: BOOL;
0029     CABET1: BOOL;
0030     CABET2: BOOL;
0031     CABET3: BOOL;
0032
0033     (*Porte fermée*)
0034     PRDCouv: BOOL;
0035     P1ouv: BOOL;
0036     P2ouv: BOOL;
0037     P3ouv: BOOL;
0038
0039     (*Porte ouverte*)
0040     PRDCfer: BOOL;
0041     P1fer: BOOL;
0042     P2fer: BOOL;
0043     P3fer: BOOL;
0044
0045     (* Obstacle a la fermeture*)
0046     Obs: BOOL;
0047
0048     (* Arret dy systeme *)
0049 (* ————————————
0050     Sorties
```

```
0051 ————————————————*)
0052     (*Ouverture de porte*)
0053     ORDC: BOOL;
0054     O1: BOOL;
0055     O2: BOOL;
0056     O3: BOOL;
0057
0058     (*Fermeture de porte*)
0059     FRDC: BOOL;
0060     F1: BOOL;
0061     F2: BOOL;
0062     F3: BOOL;
0063
0064     (*Moteur cabine*)
0065     CABD: BOOL;
0066     CABM: BOOL;
0067
0068     (*Voyant de présence d'étage*)
0069     VPALRDC: BOOL;
0070     VPAL1: BOOL;
0071     VPAL2: BOOL;
0072     VPAL3: BOOL;
0073
0074     (*Voyant d'apple étage*)
0075     VRDC: BOOL;
0076     V1: BOOL;
0077     V2: BOOL;
0078     V3: BOOL;
0079
0080 (* ————————————————
0081         Autres Variables
0082 ————————————————*)
0083     (*Variable du Timer*)
0084     T1: TP;
0085     (*Variable d'initialization*)
0086     STOPbuttonVar: BOOL;
0087     CABDstatus: BOOL;
0088     CABMstatus: BOOL;
0089     T2: TP;
0090     ESTINIT2: BOOL;
0091     ReinitAscenseur: BOOL;
0092     SFCInit: BOOL;
0093     SFCReset: BOOL;
0094     T3: TP;
0095     COUNTSTEP: INT;
0096     CompteurFor: INT;
0097     ETAGE_SUIVANT: INT;
0098     SENS_M: BOOL;
0099     TableDeVeriteMontee: ARRAY [1..4] OF BOOL;
0100     TableDeVeriteDesc: ARRAY [1..4] OF BOOL;
0101     TestSens: INT;
0102     PROCHAIN_ARRET: INT;
0103     ArretsDesc: ARRAY [1..4] OF INT;
0104     ArretsMonte: ARRAY [1..4] OF INT;
0105     RE_OUVRE: BOOL;
0106     ARRETcabRDC: BOOL;
0107     ARRETcabM1: BOOL;
0108     ARRETcabD1: BOOL;
0109     ARRETcabM2: BOOL;
0110     ARRETcabD2: BOOL;
0111     ARRETcabET3: BOOL;
0112     CABRDCswitch: BOOL;
0113     CABET1switch: BOOL;
0114     CABET2switch: BOOL;
0115     CABET3switch: BOOL;
0116
0117     UpFleche: BOOL;
0118     DownFleche: BOOL;
```

```
0119    RightFleche: BOOL;
0120    LeftFleche: BOOL;
0121    FlecheSwitch:BOOL;
0122
0123    CABmoveY: INT;
0124    VisuRDCDx: INT;
0125    VisuRDCDy: REAL;
0126    VisuRDCGx: REAL;
0127    VisuRDCGy: REAL;
0128    VisuET1Dx: INT;
0129    VisuET1Dy: REAL;
0130    VisuET1Gx: REAL;
0131    VisuET1Gy:REAL;
0132    VisuET2Dx: INT;
0133    VisuET2Dy: REAL;
0134    VisuET2Gx: REAL;
0135    VisuET2Gy: REAL;
0136    VisuET3Dx: INT;
0137    VisuET3Dy: REAL;
0138    VisuET3Gx: REAL;
0139    VisuET3Gy: REAL;
0140    InitPortesAscSalon: BOOL;
0141    T6: TP;
0142    TIMEURblock: BOOL;
0143    CompteActionsTime: INT;
0144    wait: TP;
0145    PALRDCsave: BOOL;
0146    PALD1save: BOOL;
0147    PALM1save: BOOL;
0148    PALD2save: BOOL;
0149    PALM2save: BOOL;
0150    PAL3save: BOOL;
0151    CABRDCsave: BOOL;
0152    CABET1save: BOOL;
0153    CABET2save: BOOL;
0154    CABET3save: BOOL;
0155    AfficheBoutonsReset: BOOL;
0156    BoutonResetVrai: BOOL;
0157    BoutonRepriseVrai: BOOL;
0158    STOPbuttonVarImpuls: BOOL;
0159    ARRETcabRDCsave: BOOL;
0160    ARRETcabM1save: BOOL;
0161    ARRETcabD1save: BOOL;
0162    ARRETcabM2save: BOOL;
0163    ARRETcabD2save: BOOL;
0164    ARRETcabET3save: BOOL;
0165    ReOuvreCabine: BOOL;
0166    ForceFermeCabine: BOOL;
0167    BoutonRepriseStopSwitch: BOOL;
0168    ZoneDepCab: INT;
0169    DisplayNumbET3: BOOL;
0170    DisplayNumbET2: BOOL;
0171    DisplayNumbET1: BOOL;
0172    DisplayNumbETRDC: BOOL;
0173    FLECHE_VERS: BOOL;
0174    DisplayVersRDC: BOOL;
0175    DisplayVersET1: BOOL;
0176    DisplayVersET2: BOOL;
0177    DisplayVersET3: BOOL;
0178    FinDefinition_SENS: BOOL;
0179    Offset_ChiffreY: INT;
0180    CABDsave: BOOL;
0181    CABMsave: BOOL;
0182    ORDCsave: BOOL;
0183    O1save: BOOL;
0184    O2save: BOOL;
0185    O3save: BOOL;
0186    F1save: BOOL;
```

```
0187      FRDCsave: BOOL;
0188      F2save: BOOL;
0189      F3save: BOOL;
0190      PROCHAIN_ARRETsave: INT;
0191      ETAGE_SUIVANTsave: INT;
0192      AfficheBoutonsReset2: BOOL;
0193      StanbyActive: BOOL;
0194      EXIT_StanbyFunc: BOOL;
0195      Niveau_Actuel: INT;
0196      Ancien_Niveau: INT;
0197      Last_Capt_Pres_Rec: INT;
0198  END_VAR
```

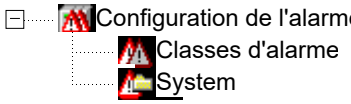
Administration d´espion et des recettes

Standard

Watch0

```
0001  wait.Q
0002
```

Configuration de l'alarme



Configuration de l'automate

*__not_found__ (Id.:)

Numéro de noeud: 0
Adresse d'entrée:%IB0
Adresse de sortie: %QB0
Adresse de diagnose: %MB0
Download: 1
AutoAdr: 1

Configuration des tâches

Espace de travail

Histogramme

Pas lancé d'histogramme

Manager des paramètres

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